

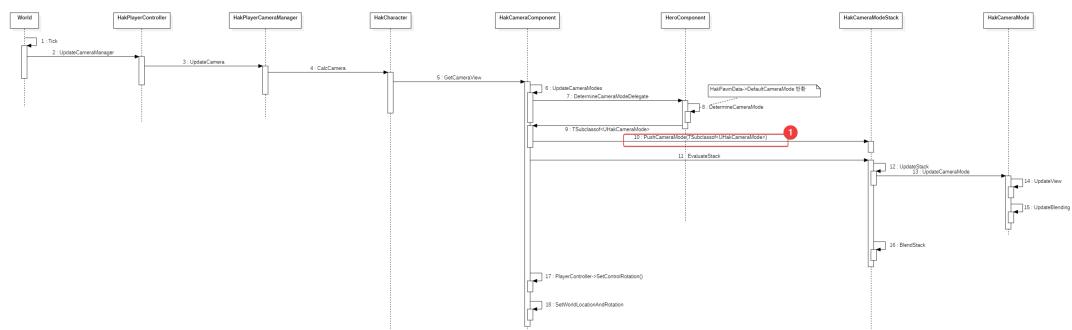


8주차 (2023.11.13)

CameraModeStack

▼ 펼치기

- PushCameraMode를 구현해보자:



- 아래와 같이 이전에 우리는 주석 처리를 하였다:

```
void UHalCameraComponent::UpdateCameraModes()
{
    check(CameraModeStack);

    // DetermineCameraModeDelegate는 CameraMode Class를 반환한다:
    // - 해당 delegate는 HeroComponent의 멤버 함수로 바인딩되어 있다
    if (DetermineCameraModeDelegate.IsBound())
    {
        if (const TSubclassOf<UHalCameraMode> CameraMode = DetermineCameraModeDelegate.Execute())
        {
            // CameraModeStack->PushCameraMode(CameraMode); 1
        }
    }
}
```

- 아래와 같이 PushCameraMode()를 .h/cpp에 추가해주자:

```

#pragma once

#include "HakCameraMode.generated.h"

/** Camera Blending 대상 유닛 */
UCLASS(Abstract, NotBlueprintable)
class UHakCameraMode : public UObject
{
    GENERATED_BODY()
public:
    UHakCameraMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
};

/** Camera Blending을 담당하는 객체 */
UCLASS()
class UHakCameraModeStack : public UObject
{
    GENERATED_BODY()
public:
    UHakCameraModeStack(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /**
     * member methods
     */
    UHakCameraMode* GetCameraModeInstance(TSubclassOf<UHakCameraMode>& CameraModeClass);
    void PushCameraMode(TSubclassOf<UHakCameraMode>& CameraModeClass);

    /**
     * member variables
     */

    /** 생성된 CameraMode를 관리 */
    UPROPERTY()
    TArray<TObjectPtr<UHakCameraMode>> CameraModeInstances;

    /** Camera Matrix Blend 업데이트 진행 큐 */
    UPROPERTY()
    TArray<TObjectPtr<UHakCameraMode>> CameraModeStack;
};

```

```

void UHakCameraModeStack::PushCameraMode(TSubclassOf<UHakCameraMode>& CameraModeClass)
{
    if (!CameraModeClass)
    {
        return;
    }

    UHakCameraMode* CameraMode = GetCameraModeInstance(CameraModeClass);
}

```

□ GetCameraModeInstance를 구현하자:

```

UHakCameraMode* UHakCameraModeStack::GetCameraModeInstance(TSubclassOf<UHakCameraMode>& CameraModeClass)
{
    check(CameraModeClass);

    // CameraModeInstances에서 먼저 생성되어있는지 확인 후, 반환한다:
    for (UHakCameraMode* CameraMode : CameraModeInstances)
    {
        // CameraMode는 UClass를 비교한다:
        // - 즉, CameraMode는 클래스 타입에 하나만 생기게된다
        if ((CameraMode != nullptr) && (CameraMode->GetClass() == CameraModeClass))
        {
            return CameraMode;
        }
    }

    // CameraModeClass에 알맞는 CameraMode의 인스턴스가 없다면 생성한다:
    UHakCameraMode* NewCameraMode = NewObject<UHakCameraMode>(GetOuter(), CameraModeClass, NAME_None, RF_NoFlags);
    check(NewCameraMode);

    // 여기서 우리는 CameraModeInstances는 CameraModeClass에 맞는 인스턴스를 가지고(관리하는) 컨테이너이라는 것을 알 수 있다
    CameraModeInstances.Add(NewCameraMode);

    return NewCameraMode;
}

```

□ CameraMode에 멤버 변수들을 추가해주자:

```

    /** Camera Blending 대상 유닛 */
    UCLASS(Abstract, NotBlueprintable)
    class UHakCameraMode : public UObject
    {
        GENERATED_BODY()
    public:
        UPROPERTY(EditDefaultsOnly, Category="Blending")
        float BlendTime;

        /** 선형적인 Blend 값 [0, 1] */
        float BlendAlpha;

        /**
         * 해당 CameraMode의 최종 Blend 값
         * 앞서 BlendAlpha의 선형 값을 매핑하여 최종 BlendWeight를 계산 (코드를 보면, 이해해보자)
         */
        float BlendWeight;
    };

```

1

□ PushCameraMode를 먼저 구현해보자:

```

void UHakCameraModeStack::PushCameraMode(TSubclassOf<UHakCameraMode> CameraModeClass)
{
    if (!CameraModeClass)
    {
        return;
    }

    UHakCameraMode* CameraMode = GetCameraModeInstance(CameraModeClass);
    check(CameraMode);

    int32 StackSize = CameraModeStack.Num();
    if ((StackSize > 0) && (CameraModeStack[0] == CameraMode))
    {
        // CameraModeStack[0] 가장 최근에 이미 CameraMode가 Stacking되었으므로 그냥 리턴!
        return;
    }

    // ExistingStackIndex는 CameraModeStack에서 CameraMode에 맞는 Index를 찾음
    // ExistingStackContribution은 위에서 아래로 최종 BlendWeight 값을 찾기 위해 초기값으로 1.0으로 설정
    int32 ExistingStackIndex = INDEX_NONE;
    float ExistingStackContribution = 1.0f;

    /**
     * BlendWeight | ExistingStackContribution | ExistingStackContribution (accumulated)
     * 0.1f       | (1.0f) * 0.1f = 0.1f      | (1.0f - 0.1f) = 0.9f
     * 0.3f       | (0.9f) * 0.3f = 0.27f      | (1.0f - 0.3f) * 0.9f = 0.63f
     * 0.6f       | (0.63f) * 0.6f = 0.378f    | (1.0f - 0.6f) * 0.63f = 0.252f
     * 1.0f       | (0.252f) * 1.0f = 0.252f   |
     *             | 0.1f + 0.27f + 0.378f + 0.252f = 1.0f!
    */
    for (int32 StackIndex = 0; StackIndex < StackSize; ++StackIndex)
    {
        if (CameraModeStack[StackIndex] == CameraMode)
        {
            ExistingStackIndex = StackIndex;
            // BlendWeight를 CameraMode에 추가해준다:
            // ~ 여기서 ExistingStackContribution 우리가 찾는 CameraMode를 찾았으니까, 누적된 BlendWeight와 함께 BlendWeight를 곱하여, 루프를 빠져나온다
            ExistingStackContribution *= CameraMode->BlendWeight;
            break;
        }
        else
        {
            // 당연히 우리가 원하는 CameraMode가 아니니까, InvBlendWeight = (1.0 - BlendWeight)을 곱해줘야, 값이 누적되겠지?
            ExistingStackContribution *= (1.0f - CameraModeStack[StackIndex]->BlendWeight);
        }
    }

    // 해당 루프의 동작 원리는 앞서 적어놓은 표를 확인해보며 이해해보자.

    // 우리는 CameraMode를 Top으로 반영하기 위해, 당연히 중간에 있다면, 제거하여 다시 Push 해줘야 한다
    if (ExistingStackIndex != INDEX_NONE)
    {
        CameraModeStack.RemoveAt(ExistingStackIndex);
        StackSize--;
    }
    else
    {
        // 없었으니까 당연히 Contribution은 0으로 세팅해줘야겠지?
        ExistingStackContribution = 0.0f;
    }

    // BlendTime이 보다 크다는 것은 Blend를 얼마 시간동안 진행할을 의미 따라서, ExistingStackContribution을 적용
    // 따라서 Blend하지 않는다면, BlendWeight를 1.0을 넣어 새로 넣는 CameraMode만 적용될 것이다
    const bool bShouldBlend = ((CameraMode->BlendTime > 0.0f) && (StackSize > 0));
    const float BlendWeight = (bShouldBlend ? ExistingStackContribution : 1.0f);
    CameraMode->BlendWeight = BlendWeight;

    // 좀.. Array를 Stack처럼 사용하는것은 알지만, Index 0에 넣는건 비효율적인데..
    // ~ Push, Pop 메시드와 같이 그냥 last에 넣는게... 아까 싶음
    CameraModeStack.Insert(CameraMode, 0);

    // 항상 마지막에는 100%로 해놓는다? (아직은 이해가 안됨)
    // ~ 전전히 토록 보면 이해될거임
    CameraModeStack.Last()->BlendWeight = 1.0f;
}

```

□ PushCameraMode의 주석을 풀어주자:

```

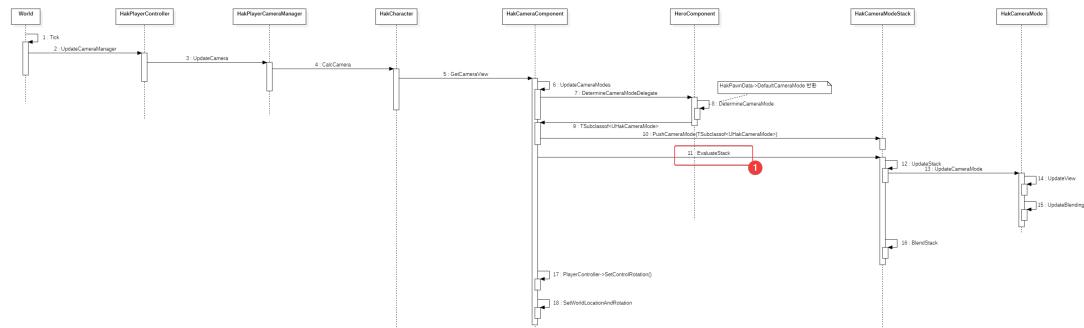
void UHakCameraComponent::UpdateCameraModes()
{
    check(CameraModeStack);

    // DetermineCameraModeDelegate는 CameraMode Class를 반환한다;
    // - 해당 delegate는 HeroComponent의 멤버 함수로 바인딩되어 있다
    if (DetermineCameraModeDelegate.IsBound())
    {
        if (TSubclassOf<UHakCameraMode> CameraMode = DetermineCameraModeDelegate.Execute())
        {
            CameraModeStack->PushCameraMode(CameraMode);
        }
    }
}

```

1

□ EvaluateStack을 구현해보자:



□ GetCameraView()에서 EvaluateStack 관련 코드를 추가하자:

```

void UHakCameraComponent::GetCameraView(FVector& DesiredView)
{
    check(CameraModeStack);

    // 해당 함수는 HeroComponent에서 PawnData에서 DefaultCameraMode를 가져와서 CameraModeStack에 추가하여, CameraMode 블렌딩 준비한다
    UpdateCameraModes();

    FHakCameraModeView CameraModeView;
    CameraModeStack->EvaluateStack(DeltaTime, CameraModeView);
}

```

1

□ UpdateCameraMode() 함수에 대한 간단한 정리 설명

□ FHakCameraModeView를 구현하자:

```

#pragma once

#include "HakCameraMode.generated.h"

/**
 * FHakCameraModeView
 */
struct FHakCameraModeView
{
    FHakCameraModeView();

    FVector Location;
    FRotator Rotation;
    FRotator ControlRotation;
    float FieldOfView;
};

```

```

#include "HakCameraMode.h"
#include "HakPlayerCameraManager.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(HakCameraMode)

FHakCameraModeView::FHakCameraModeView()
    : Location(ForceInit)
    , Rotation(ForceInit)
    , ControlRotation(ForceInit)
    , FieldOfView(HAK_CAMERA_DEFAULT_FOV)
{ }

UHakCameraMode::UHakCameraMode(const FObjectInitializer& ObjectInitializer)
    : Super(ObjectInitializer)
{ }

UHakCameraModeStack::UHakCameraModeStack(const FObjectInitializer& ObjectInitializer)

```

□ EvaluateStack() .h/.cpp에 정의

```

/** Camera Blending을 담당하는 객체 */
UCLASS()
class UHakCameraModeStack : public UObject
{
    GENERATED_BODY()
public:
    UHakCameraModeStack(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /**
     * member methods
     */
    UHakCameraMode* GetCameraModeInstance(TSubclassOf<UHakCameraMode>& CameraModeClass);
    void PushCameraMode(TSubclassOf<UHakCameraMode>& CameraModeClass);
    void EvaluateStack(float DeltaTime, FHakCameraModeView& OutCameraModeView); 1

    /**
     * member variables
     */

    /** 생성된 CameraMode를 관리 */
    UPROPERTY()
    TArray<TObjectPtr<UHakCameraMode>> CameraModeInstances;

    /** Camera Matrix Blend 업데이트 진행 큐 */
    UPROPERTY()
    TArray<TObjectPtr<UHakCameraMode>> CameraModeStack;
};


```

```

void UHakCameraModeStack::EvaluateStack(float DeltaTime, FHakCameraModeView& OutCameraModeView)
{
}

```

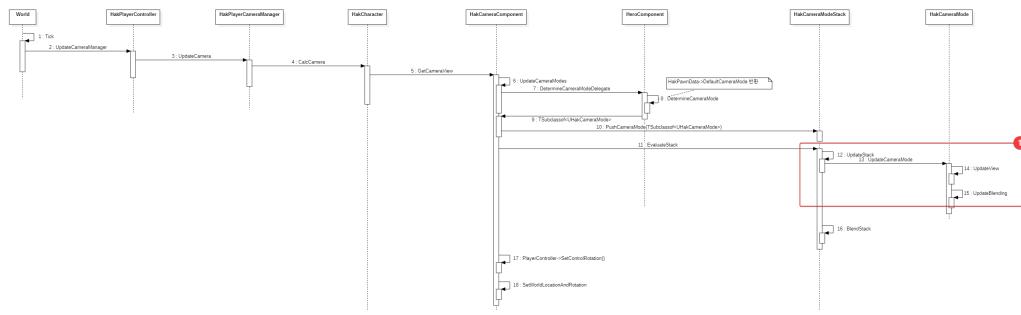
- Evaluate는 크게 2개로 구분된다:

```

void UHakCameraModeStack::EvaluateStack(float DeltaTime, FHakCameraModeView& OutCameraModeView)
{
    UpdateStack(DeltaTime); 1
    BlendStack(OutCameraModeView); 2
}

```

□ UpdateStack()을 구현해보자:



□ UpdateStack()의 .h/.cpp 정의하자:

```

/** Camera Blending을 담당하는 객체 */
UCLASS()
class UHakCameraModeStack : public UObject
{
    GENERATED_BODY()
public:
    UHakCameraModeStack(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /**
     * member methods
     */
    UHakCameraMode* GetCameraModeInstance(TSubclassOf<UHakCameraMode>& CameraModeClass);
    void PushCameraMode(TSubclassOf<UHakCameraMode>& CameraModeClass);
    void UpdateStack(float DeltaTime); 1
    void EvaluateStack(float DeltaTime, FHakCameraModeView& OutCameraModeView);

    /**
     * member variables
     */

    /** 생성된 CameraMode를 관리 */
    UPROPERTY()
    TArray<TObjectPtr<UHakCameraMode>> CameraModeInstances;

    /** Camera Matrix Blend 업데이트 진행 큐 */
    UPROPERTY()
    TArray<TObjectPtr<UHakCameraMode>> CameraModeStack;
};

```

□ UpdateStack의 대략적인 구현:

```

void UHakCameraModeStack::UpdateStack(float DeltaTime)
{
    const int32 StackSize = CameraModeStack.Num();
    if (StackSize <= 0)
    {
        return;
    }

    int32 RemoveCount = 0;
    int32 RemoveIndex = INDEX_NONE;
    for (int32 StackIndex = 0; StackIndex < StackSize; ++StackIndex)
    {
        UHakCameraMode* CameraMode = CameraModeStack[StackIndex];
        check(CameraMode);

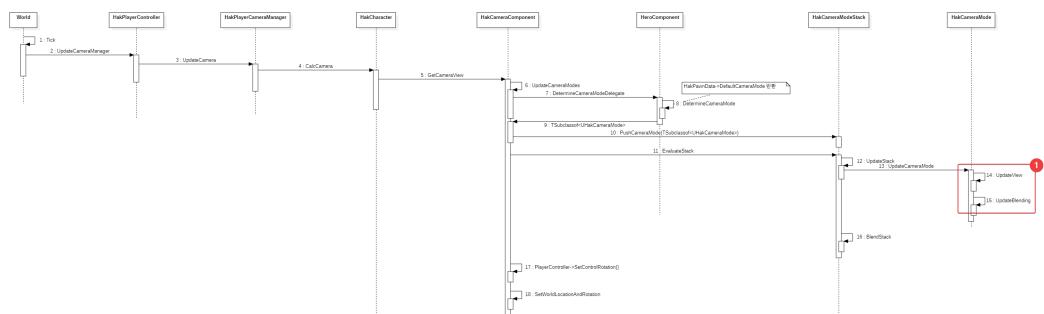
        // UpdateCameraMode를 확인해보자:
        CameraMode->UpdateCameraMode(DeltaTime); ①

        if (CameraMode->BlendWeight >= 1.0f)
        {
            RemoveIndex = (StackIndex + 1);
            RemoveCount = (StackSize - RemoveIndex);
            break;
        }
    }

    if (RemoveCount > 0)
    {
        // 생각해보면 이거 때문에 Pop하고 Push를 안한거일지도?
        CameraModeStack.RemoveAt(RemoveIndex, RemoveCount);
    }
}

```

□ UpdateCameraMode를 구현하자:



□ 간단한 정의를 .h/.cpp에 구현:

```

/** Camera Blending 대상 유닛 */
UCLASS(Abstract, NotBlueprintable)
class UHakCameraMode : public UObject
{
    GENERATED_BODY()
public:
    UHakCameraMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /**
     * member methods
     */
    void UpdateCameraMode(float DeltaTime); ①

    /** 얼마동안 Blend를 진행할지 시간을 의미 */
    UPROPERTY(EditDefaultsOnly, Category="Blending")
    float BlendTime;

    /** 선형적인 Blend 값 [0, 1] */
    float BlendAlpha;

    /**
     * 해당 CameraMode의 최종 Blend 값
     * 앞서 BlendAlpha의 선형 값을 매핑하여 최종 BlendWeight를 계산 (코드를 보며, 이해해보자)
     */
    float BlendWeight;
};

```

```

void UHakCameraMode::UpdateCameraMode(float DeltaTime)
{
    UpdateView(DeltaTime);
    UpdateBlending(DeltaTime);
}

```

□ UpdateView()를 구현하자:

□ 우선 아래 마킹했듯이 멤버 변수를 추가하자:

```

void UHakCameraMode::UpdateView(float DeltaTime)
{
    FVector PivotLocation = GetPivotLocation();
    FRotator PivotRotation = GetPivotRotation();

    PivotRotation.Pitch = FMath::ClampAngle(PivotRotation.Pitch, ViewPitchMin, ViewPitchMax); 1

    2 View.Location = PivotLocation;
    View.Rotation = PivotRotation;
    View.ControlRotation = View.Rotation;
    View.FieldOfView = FieldOfView; 3
}

```

- ViewPitch[Min|Max], View, FieldOfView 변수 추가:

```

/** Camera Blending 대상 유닛 */
UCLASS(Abstract, NoBlueprintable)
class UHakCameraMode : public UObject
{
    GENERATED_BODY()
public:
    UHakCameraMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /** 
     *+ member methods
     */
    void UpdateView(float DeltaTime);
    void UpdateCameraMode(float DeltaTime);

    /** CameraMode에 의해 생성된 CameraModeView */
    FHakCameraModeView* View;

    /** Camera Mode의 FOV */
    UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "5.0", UIMax = "170", ClampMin = "5.0", Clampmax = "170.0"))
    float FieldOfView;

    /** View에 대한 Pitch [Min, Max] */
    UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "-89.9", UIMax = "89.9", ClampMin = "-89.9", Clampmax = "89.9"))
    float ViewPitchMin;

    UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "-89.9", UIMax = "89.9", ClampMin = "-89.9", Clampmax = "89.9"))
    float ViewPitchMax;

    /** 얼마동안 Blend를 진행할지 시간을 의미 */
    UPROPERTY(EditDefaultsOnly, Category="Blending")
    float BlendTime;

    /** 선형적인 Blend 값 [0, 1] */
    float BlendAlpha;

    /**
     *+ 해당 CameraMode의 최종 Blend 값
     *+ 앞서 BlendAlpha의 선형 값을 매핑하여 최종 BlendWeight를 계산 (코드를 보며, 이해해보자)
     */
    float BlendWeight;
};

```

- Constructor에 추가된 변수 초기화:

```

UHakCameraMode::UHakCameraMode(const FObjectInitializer& ObjectInitializer)
    : Super(ObjectInitializer)
{
    FieldOfView = HAK_CAMERA_DEFAULT_FOV;
    ViewPitchMin = HAK_CAMERA_DEFAULT_PITCH_MIN;
    ViewPitchMax = HAK_CAMERA_DEFAULT_PITCH_MAX;

    BlendTime = 0.0f;
    BlendAlpha = 1.0f;
    BlendWeight = 1.0f;
}

```

□ GetPivotLocation()과 GetPivotRotation() 구현:

- GetPivotLocation()

- GetTargetActor()를 구현하자:

```
UActor* UHakCameraMode::GetTargetActor() const
{
    const UHakCameraComponent* HakCameraComponent = GetHakCameraComponent();
    return HakCameraComponent->GetTargetActor();
}
```

- GetHakCameraComponent 구현:

```
UHakCameraComponent* UHakCameraMode::GetHakCameraComponent() const
{
    // 우리가 앞서 UHakCameraMode를 생성하는 곳은 UHakCameraModeStack이었다;
    // - 해당 코드를 보면, GetOuter()를 HakCameraMode로 HakCameraComponent로 설정하였다
    // - UHakCameraModeStack::GetCameraModeInstance() 확인
    return CastChecked<UHakCameraComponent>(GetOuter());
}
```

- UHakCameraComponent::GetTargetActor() 구현:

```
UCLASS()
class UHakCameraComponent : public UCameraComponent
{
    GENERATED_BODY()
public:
    UHakCameraComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
    static UHakCameraComponent* FindCameraComponent(const Actor* Actor) { return (Actor ? Actor->FindComponentByClass<UHakCameraComponent>() : nullptr); }

    /**
     * member methods
     */
    Actor* GetTargetActor() const { return GetOwner(); } 1
    void UpdateCameraNodes();

    /**
     * CameraComponent interface
     */
    virtual void OnRegister() final;
    virtual void GetCameraView(Float DeltaTime, FMiminalViewInfo& DesiredView) final;

    /**
     * member variables
     */
    /** 카메라의 blending 기능을 지원하는 stack */
    UPROPERTY()
    TObjectPtr<UHakCameraModeStack> CameraModeStack;

    /** 현재 CameraMode를 가지오는 Delegate */
    FHakCameraModeDelegate DetermineCameraModeDelegate;
};


```

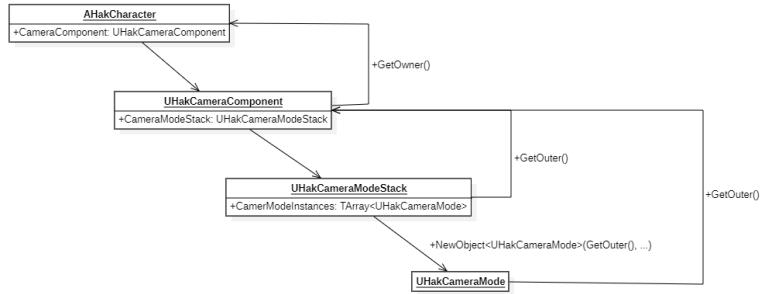
- 여기서, CameraComponent는 HakCharacter에 생성했었다:

```
UHakCharacter::UHakCharacter(const FObjectInitializer& ObjectInitializer)
: Super(ObjectInitializer)
{
    // Tick을 비활성화
    PrimaryActorTick.bCanEverTick = false;
    PrimaryActorTick.bStartWithTickEnabled = false;

    // PawnExtComponent 생성
    PawnExtComponent = CreateDefaultSubobject<UHakPawnExtensionComponent>(TEXT("PawnExtensionComponent"));

    // CameraComponent 생성
    {
        CameraComponent = CreateDefaultSubobject<UHakCameraComponent>(TEXT("CameraComponent"));
        CameraComponent->SetRelativeLocation(FVector(-300.0f, 0.0f, 75.0f));
    } 2
}
```

- 여기서 잠깐! CameraMode → CameraModeStack → CameraComponent → Character간의 관계를 UML로 잠깐 정리하고 가자:



- GetPivotRotation()

```

FRotator UHakCameraMode::GetPivotRotation() const
{
    const AActor* TargetActor = GetTargetActor();
    check(TargetActor);

    if (const APawn* TargetPawn = Cast<APawn>(TargetActor))
    {
        // GetViewRoation() 확인, 보통 Pawn의 ControlRotation을 반환
        return TargetPawn->GetViewRotation();
    }

    return TargetActor->GetActorRotation();
}

```

□ UpdateView 코드 리뷰:

```

void UHakCameraMode::UpdateView(float DeltaTime)
{
    // CameraMode를 가지고 있는 CameraComponent의 Owner인 Character(Pawn)을 활용하여, PivotLocation/Rotation을 반환함
    FVector PivotLocation = GetPivotLocation();
    FRotator PivotRotation = GetPivotRotation();

    // Pitch 값에 대해 Min/Max를 Clamp시킴
    PivotRotation.Pitch = FMath::ClampAngle(PivotRotation.Pitch, ViewPitchMin, ViewPitchMax);

    // FHakCameraModeView의 PivotLocation/Rotation 설정
    View.Location = PivotLocation;
    View.Rotation = PivotRotation;

    // PivotRotation을 똑같이 ControlRotation으로 활용
    View.ControlRotation = View.Rotation;
    View.FieldOfView = FieldOfView;

    // 정리하면, Character의 Location과 ControlRotation을 활용하여, View를 업데이트함
}

```

□ UpdateBlending 구현:

□ 우선 EHakCameraModeBlendFunction을 구현하자:

```

    /**
     * [0,1]을 BlendFunction에 맞게 재매핑을 위한 타입
     */
    UENUM(BlueprintType)
    enum class EHakCameraModeBlendFunction : uint8
    {
        Linear,
        /**
         * EaseIn/Out은 exponent 값에 의해 조절된다:
         */
        EaseIn,
        EaseOut,
        COUNT
    };

```

- 해당 Enum을 사용하여, CameraMode에 BlendFunction을 멤버 변수로 넣어주자:

- 그리고 Exponent도!

```

    /** Camera Blending 대상 유닛 */
    UCLASS(Abstract, NotBlueprintable)
    class UHakCameraMode : public UObject
    {
        GENERATED_BODY()
    public:
        UHakCameraMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
        /**
         * member methods
         */
        UHakCameraComponent* GetHakCameraComponent() const;
        AActor* GetTargetActor() const;
        FVector GetPivotLocation() const;
        FRotator GetPivotRotation() const;
        void UpdateView(float DeltaTime);
        void UpdateBlending(float DeltaTime);
        void UpdateCameraMode(float DeltaTime);

        /** CameraMode에 의해 생성된 CameraModeView */
        FHakCameraModeView View;

        /** Camera Mode의 FOV */
        UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "5.0", UIMax = "170", ClampMin = "5.0", Clampmax = "170.0"))
        float FieldOfView;

        /** View에 대한 Pitch [Min, Max] */
        UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "-89.9", UIMax = "89.9", ClampMin = "-89.9", Clampmax = "89.9"))
        float ViewPitchMin;

        UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "-89.9", UIMax = "89.9", ClampMin = "-89.9", Clampmax = "89.9"))
        float ViewPitchMax;

        /** 얼마동안 Blend를 진행할지 시간을 의미 */
        UPROPERTY(EditDefaultsOnly, Category="Blending")
        float BlendTime;

        /** 선형적인 Blend 값 [0, 1] */
        float BlendAlpha;

        /**
         * 해당 CameraMode의 최종 Blend 값
         * 앞서 BlendAlpha의 선형 값을 매핑하여 최종 BlendWeight를 계산 (코드를 보며, 이해해보자)
         */
        float BlendWeight;

        /**
         * EaseIn/Out에 사용한 Exponent
         */
        UPROPERTY(EditDefaultsOnly, Category = "Blending")
        float BlendExponent;

        /** Blend function */
        EHakCameraModeBlendFunction BlendFunction;
    };

```

- 당연히 Constructor도 업데이트 해주자:

```

    UHakCameraMode::UHakCameraMode(const FObjectInitializer& ObjectInitializer)
        : Super(ObjectInitializer)

        FieldOfView = HAK_CAMERA_DEFAULT_FOV;
        ViewPitchMin = HAK_CAMERA_DEFAULT_PITCH_MIN;
        ViewPitchMax = HAK_CAMERA_DEFAULT_PITCH_MAX;

        BlendTime = 0.0f;
        BlendAlpha = 1.0f;
        BlendWeight = 1.0f;

        BlendFunction = EHakCameraModeBlendFunction::EaseOut;
        BlendExponent = 4.0f;
    }

```

1

□ 이어서, UpdateBlending 구현:

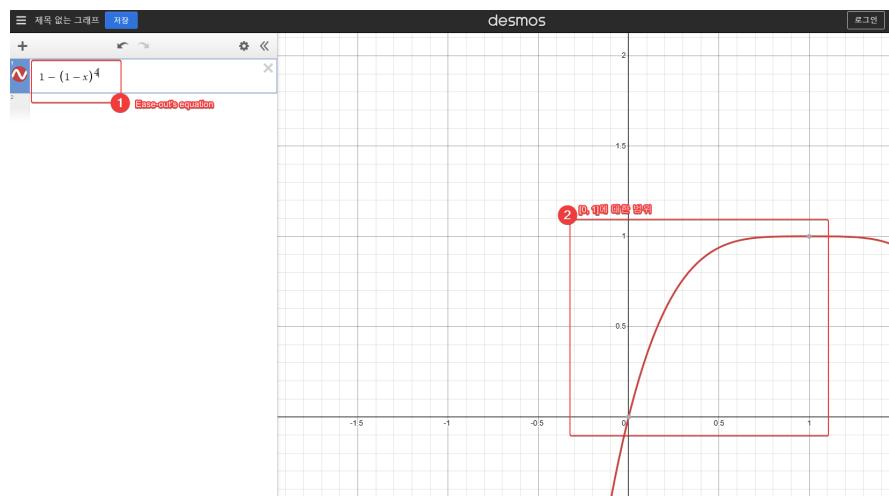
```

void UHakCameraMode::UpdateBlending(float DeltaTime)
{
    // BlendAlpha를 DeltaTime을 통해 계산
    if (BlendTime > 0.f)
    {
        // BlendTime은 Blending 과정 총 시간(초)
        // - BlendAlpha는 0 -> 1로 변화하는 과정:
        // - DeltaTime을 활용하여, BlendTime을 1로 블 경우, 진행 정도를 누적
        BlendAlpha += (DeltaTime / BlendTime);
    }
    else
    {
        BlendAlpha = 1.0f;
    }

    // BlendWeight를 [0,1]를 BlendFunction에 맞게 재매핑
    const float Exponent = (BlendExponent > 0.0f) ? BlendExponent : 1.0f;
    switch (BlendFunction)
    {
        case EHakCameraModeBlendFunction::Linear:
            BlendWeight = BlendAlpha;
            break;
        case EHakCameraModeBlendFunction::EaseIn:
            BlendWeight = FMath::InterpEaseIn(0.0f, 1.0f, BlendAlpha, Exponent);
            break;
        case EHakCameraModeBlendFunction::EaseOut:
            BlendWeight = FMath::InterpEaseOut(0.0f, 1.0f, BlendAlpha, Exponent);
            break;
        case EHakCameraModeBlendFunction::EaseInOut:
            BlendWeight = FMath::InterpEaseInOut(0.0f, 1.0f, BlendAlpha, Exponent);
            break;
        default:
            checkf(false, TEXT("UpdateBlending: Invalid BlendFunction [%d]\n"), (uint8)BlendFunction);
            break;
    }
}

```

- 해당 함수를 그래프를 그려보면 아래와 같다:



- UpdateCameraMode를 정리하면 아래와 같다:

```
void UHakCameraMode::UpdateCameraMode(float DeltaTime)
{
    // Actor를 활용하여, Pivot[Location|Rotation]을 계산하여, View를 업데이트
    UpdateView(DeltaTime);

    // BlendWeight를 DeltaTime을 활용하여, BlendAlpha 계산 후, BlendFunction에 맞게 재-매핑하여 최종 계산
    UpdateBlending(DeltaTime);
}
```

- UpdateStack 정리:

```
void UHakCameraModeStack::UpdateStack(float DeltaTime)
{
    const int32 StackSize = CameraModeStack.Num();
    if (StackSize <= 0)
    {
        return;
    }

    // CameraModeStack을 순회하며, CameraMode를 업데이트한다
    int32 RemoveCount = 0;
    int32 RemoveIndex = INDEX_NONE;
    for (int32 StackIndex = 0; StackIndex < StackSize; ++StackIndex)
    {
        UHakCameraMode* CameraMode = CameraModeStack[StackIndex];
        check(CameraMode);

        // UpdateCameraMode를 확인해보자:
        CameraMode->UpdateCameraMode(DeltaTime);

        // 만약 하나라도 CameraMode가 BlendWeight가 1.0에 도달했다면, 그 이후 CameraMode를 제거한다
        if (CameraMode->BlendWeight >= 1.0f)
        {
            RemoveIndex = (StackIndex + 1);
            RemoveCount = (StackSize - RemoveIndex);
            break;
        }
    }

    if (RemoveCount > 0)
    {
        // 생각해보면 이거 때문에 Pop하고 Push를 안한거일지도?
        CameraModeStack.RemoveAt(RemoveIndex, RemoveCount);
    }
}
```

- BlendStack을 구현하자:

```

void UHakCameraModeStack::BlendStack(FHakCameraModeView& OutCameraModeView) const
{
    const int32 StackSize = CameraModeStack.Num();
    if (StackSize <= 0)
    {
        return;
    }

    // CameraModeStack은 Bottom -> Top 순서로 Blend를 진행한다
    const FHakCameraMode* CameraMode = CameraModeStack[StackSize - 1];
    check(CameraMode);

    // 참고로 가장 아래 (Bottom)은 BlendWeight가 1.0이다!
    OutCameraModeView = CameraMode->View;

    // 이미 Index = [StackSize - 1] 이미 OutCameraModeView로 지정했으므로, StackSize - 2부터 순회하면 된다
    for (int32 StackIndex = (StackSize - 2); StackIndex >= 0; --StackIndex)
    {
        CameraMode = CameraModeStack[StackIndex];
        check(CameraMode);

        // HakCameraModeView Blend 함수를 구현하자:
        OutCameraModeView.Blend(CameraMode->View, CameraMode->BlendWeight);
    }
}

```

□ UHakCameraModeView::Blend 함수를 구현하자:

```

void FHakCameraModeView::Blend(const FHakCameraModeView& Other, float OtherWeight)
{
    if (OtherWeight <= 0.0f)
    {
        return;
    }
    else if (OtherWeight >= 1.0f)
    {
        // Weight가 1.0이면 Other를 덮어쓰면 된다
        *this = Other;
        return;
    }

    // Location + OtherWeight * (Other.Location - Location);
    Location = FMath::Lerp(Location, Other.Location, OtherWeight);

    // Location과 같은 방식 Lerp (ControlRotation과 FieldOfView도 같음)
    const FRotator DeltaRotation = (Other.Rotation - Rotation).GetNormalized();
    Rotation = Rotation + (OtherWeight * DeltaRotation);

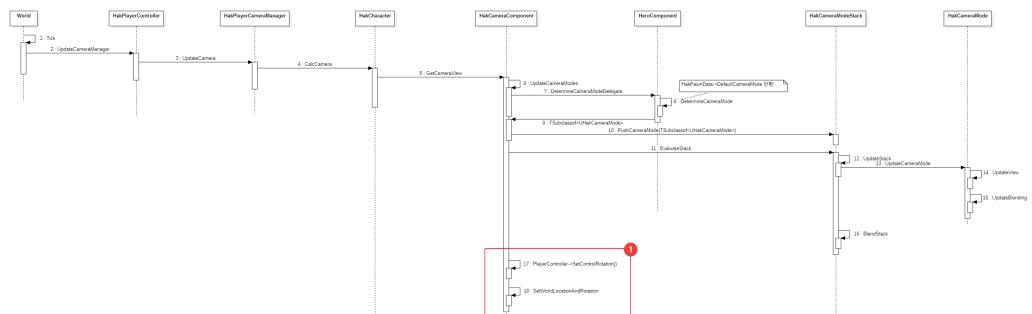
    const FRotator DeltaControlRotation = (Other.ControlRotation - ControlRotation).GetNormalized();
    ControlRotation = ControlRotation + (OtherWeight * DeltaControlRotation);

    FieldOfView = FMath::Lerp(FieldOfView, Other.FieldOfView, OtherWeight);
}

```

□ 다시 돌아와서, UHakCameraComponent::GetCameraView() 구현을 진행하자:

- 우선 우리가 진행해야 할 부분을 UML로 다시보자:



□ UHakCameraComponent::GetCameraView() 구현:

```
void UHakCameraComponent::GetCameraView(float DeltaTime, FMinimalViewInfo& DesiredView)
{
    check(CameraModeStack);

    // 해당 함수는 HeroComponent에서 PawnData에서 DefaultCameraMode를 가져와서 CameraModeStack에 추가하여, CameraMode 블랜딩 준비한다
    UpdateCameraModes();

    // EvaluateStack은 CameraModeStack에 있는 CameraMode를 업데이트(+블랜딩)하고 CameraModeStack을 Bottom-Top까지 업데이트된 CameraMode들을 Lerp를 진행해준다.
    // 이에 대한 결과는 CameraModeView에 설정된다
    FHakCameraModeView CameraModeView;
    CameraModeStack->EvaluateStack(DeltaTime, CameraModeView);

    if (APawn* TargetPawn = Cast<APawn>(GetTargetActor()))
    {
        if (APlayerController* PC = TargetPawn->GetController<APlayerController>())
        {
            // PlayerController의 ControlRotation을 계산된 CameraModeView의 ControlRotation으로 업데이트해주자
            // - SetControlRotation을 외우기보다 한번 코드를 보자:
            //   - 해당 함수는 PC가 Possess하고 있는 Pawn의 RootComponent의 ControlRotation을 반영한다 (= 조정하고 있는 캐릭터에 회전을 시키겠지?)
            PC->SetControlRotation(CameraModeView.ControlRotation);
        }
    }

    // Camera의 Location과 Rotation을 반영하자:
    // - 시간 있으면, SetWorldLocationAndRotation을 보면서 -> UpdateChildTransform까지 보는 것을 추천한다
    // - SceneGraph 관계 업데이트를 이해할 수 있다
    // 간단하게 정리하면, CameraComponent에 대해 Parent의 SceneGraph 관계를 고려하여 업데이트 진행한다
    SetWorldLocationAndRotation(CameraModeView.Location, CameraModeView.Rotation);

    // FOV 업데이트
    FieldOfView = CameraModeView.FieldOfView;

    /**
     * 여기서 우리는 ControlRotation vs Rotation을 자이점을 이해할 수 있다:
     * - ControlRotation은 PC가 조정하는 Pawn의 Rotation을 적용할 것이다
     * - 그에 반해, Rotation은 Camera에 반영하는 Rotation이라는 것이다
     */

    // FMinimalViewInfo를 업데이트 해준다:
    // - CameraComponent의 범위 사항을 반영해서 최종 렌더링까지 반영하게 됨
    DesiredView.Location = CameraModeView.Location;
    DesiredView.Rotation = CameraModeView.Rotation;
    DesiredView.FOV = CameraModeView.FieldOfView;
    DesiredView.OrthoWidth = OrthoWidth;
    DesiredView.OrthoNearClipPlane = OrthoNearClipPlane;
    DesiredView.OrthoFarClipPlane = OrthoFarClipPlane;
    DesiredView.AspectRatio = AspectRatio;
    DesiredView.bConstrainAspectRatio = bConstrainAspectRatio;
    DesiredView.bUseFieldOfViewForLOD = bUseFieldOfViewForLOD;
    DesiredView.ProjectionMode = ProjectionMode;
    DesiredView.PostProcessBlendWeight = PostProcessBlendWeight;
    if (PostProcessBlendWeight > 0.0f)
    {
        DesiredView.PostProcessSettings = PostProcessSettings;
    }
}
```

□ 관련 코드 주석 설명 진행

CameraMode_ThirdPerson

▼ 펼치기

□ ThirdPerson용으로 CurveData를 활용하여, UpdateView를 오버라이드하자:

```

1  #pragma once
2
3  #include "HakCameraMode.h"
4  [#include "HakCameraMode_ThirdPerson.generated.h"]
5
6  /** forward declarations */
7  class UCurveVector;
8
9  UCLASS(Abstract, Blueprintable)
10 class UHakCameraMode_ThirdPerson : public UHakCameraMode
11 {
12     GENERATED_BODY()
13     public:
14     UHakCameraMode_ThirdPerson(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
15
16     /**
17     * UHakCameraMode's interface
18     */
19     virtual void UpdateView(float DeltaTime) override;
20
21     /**
22     * member variables
23     */
24     UPROPERTY(EditDefaultsOnly, Category="Third Person")
25     TObjectPtr<const UCurveVector> TargetOffsetCurve;
26 };

```

1 HakCameraMode의 UpdateView()를
virtual 함수화 하자!

2

□ UHakCameraMode::UpdateView()의 virtual 함수화

```

/** Camera Blending 대상 유닛 */
UCLASS(Abstract, NotBlueprintable)
class UHakCameraMode : public UObject
{
    GENERATED_BODY()
public:
    UHakCameraMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
    /**
    * member methods
    */
    UHakCameraComponent* GetHakCameraComponent() const;
    AActor* GetTargetActor() const;
    FVector GetPivotLocation() const;
    FRotator GetPivotRotation() const;
    void UpdateBlending(float DeltaTime);
    void UpdateCameraMode(float DeltaTime);

    /**
    * HakGameMode's interface
    */
    virtual void UpdateView(float DeltaTime);
    /**
    * member variables
    */

    /** Camera Mode에 의해 생성된 CameraModeView */
    FHakCameraModeView View;

    /** Camera Mode의 FOV */
    UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "5.0", UIMax = "170", ClampMin = "5.0", Clampmax = "170.0"))
    float FieldOfView;

    /** View에 대한 Pitch [Min, Max] */
    UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "-89.9", UIMax = "89.9", ClampMin = "-89.9", Clampmax = "89.9"))
    float ViewPitchMin;

    UPROPERTY(EditDefaultsOnly, Category = "View", Meta = (UIMin = "-89.9", UIMax = "89.9", ClampMin = "-89.9", Clampmax = "89.9"))
    float ViewPitchMax;

    /** 얼마동안 Blend를 진행할지 시간을 의미 */
    UPROPERTY(EditDefaultsOnly, Category="Blending")
    float BlendTime;

    /** 선형적인 Blend 값 [0, 1] */
    float BlendAlpha;

    /**
    * 해당 CameraMode의 최종 Blend 값
    * 앞서 BlendAlpha의 선형 값을 매핑하여 최종 BlendWeight를 계산 (코드를 보면, 이해해보자)
    */
    float BlendWeight;

    /**
    * EaseIn/Out에 사용한 Exponent
    */
    UPROPERTY(EditDefaultsOnly, Category = "Blending")
    float BlendExponent;

    /** Blend function */
    EHakCameraModeBlendFunction BlendFunction;
};

```

□ CameraMode_ThirdPerson::UpdateView() 구현:

```
#include "HakCameraMode_ThirdPerson.h"
#include "Curves/CurveVector.h"
#include "UE_INLINE_GENERATED_CPP_BY_NAME(HakCameraMode_ThirdPerson)

UHakCameraMode_ThirdPerson::UHakCameraMode_ThirdPerson(const FObjectInitializer& ObjectInitializer)
    : Super(ObjectInitializer)

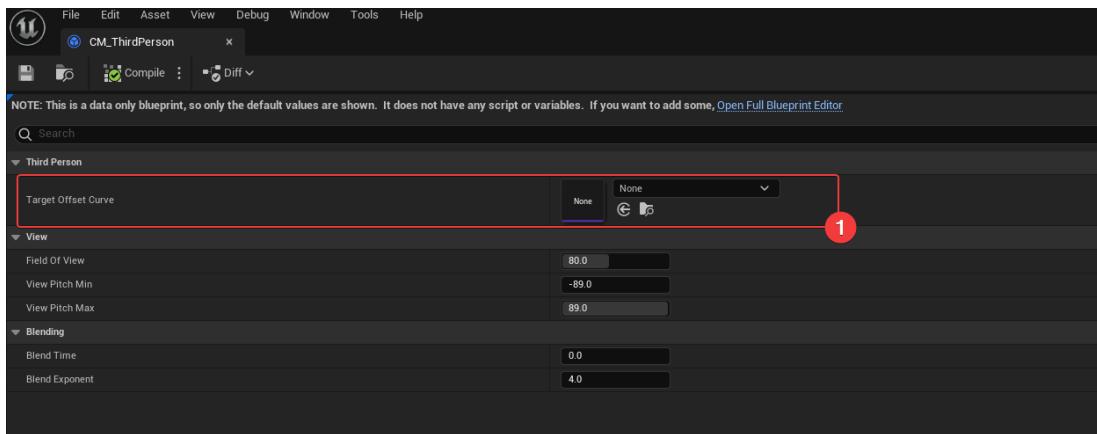
void UHakCameraMode_ThirdPerson::UpdateView(float DeltaTime)
{
    FVector PivotLocation = GetPivotLocation();
    FRotator PivotRotation = GetPivotRotation();

    PivotRotation.Pitch = FMath::ClampAngle(PivotRotation.Pitch, ViewPitchMin, ViewPitchMax);

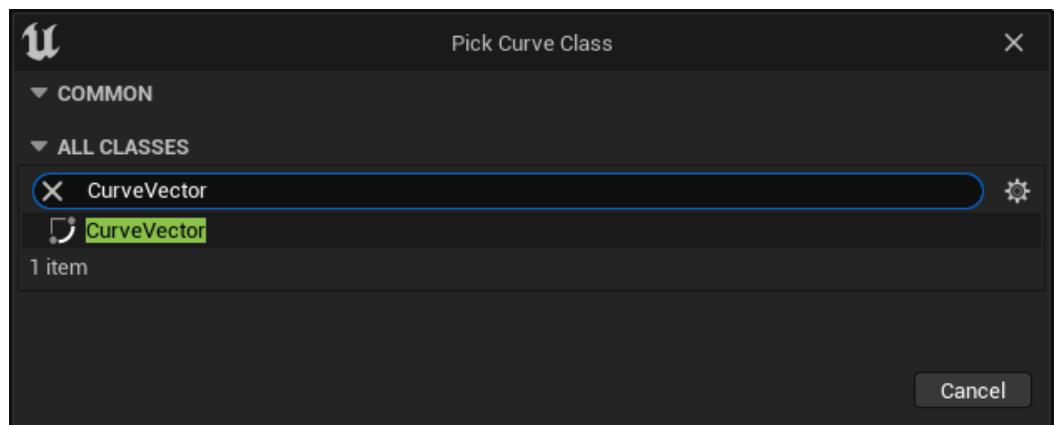
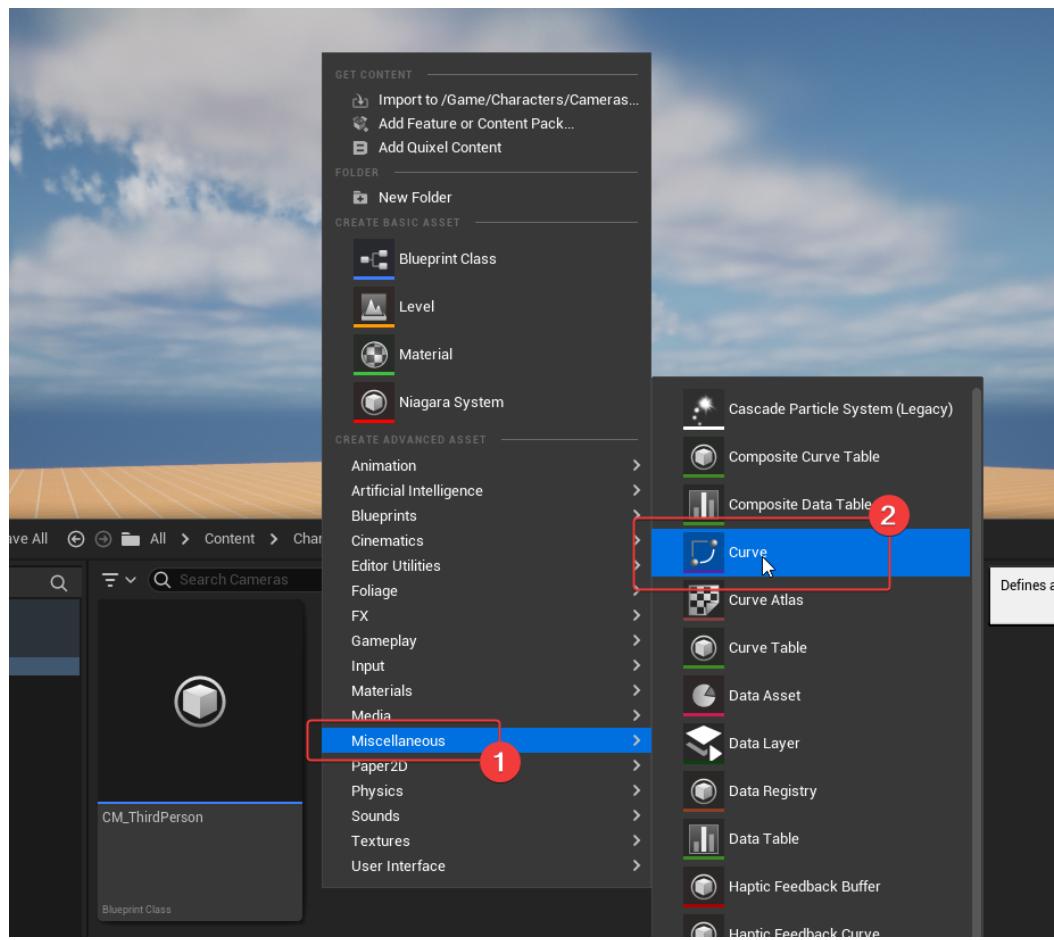
    View.Location = PivotLocation;
    View.Rotation = PivotRotation;
    View.ControlRotation = View.Rotation;
    View.FieldOfView = FieldOfView;

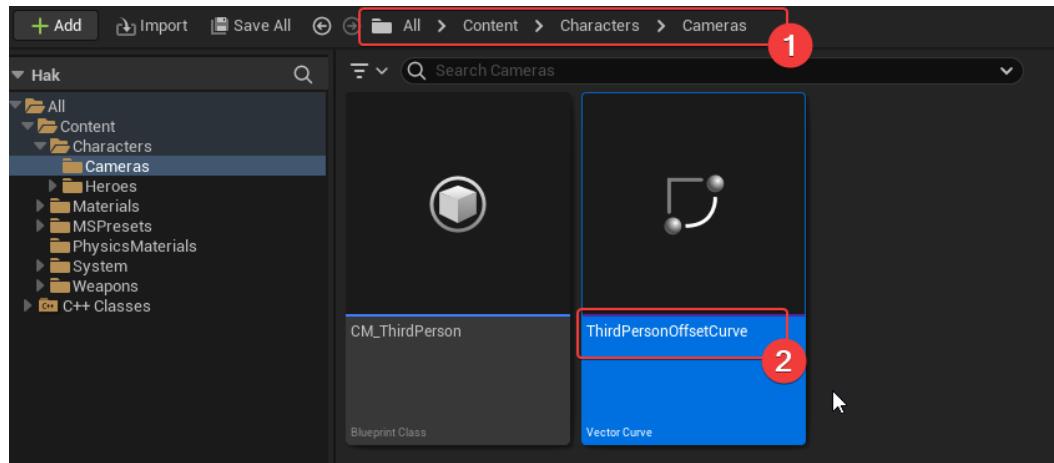
    // TargetOffsetCurve가 오버라이드되어 있다면, Curve에 값을 가져와서 적용 진행
    // - Camera 관점에서 Charater의 어느 부분에 Target으로 할지 결정하는 것으로 이해하면 됨
    if (TargetOffsetCurve)
    {
        const FVector TargetOffset = TargetOffsetCurve->GetVectorValue(PivotRotation.Pitch);
        View.Location = PivotLocation + PivotRotation.RotateVector(TargetOffset);
    }
}
```

□ CM_ThirdPerson BP 파일을 열어보자:

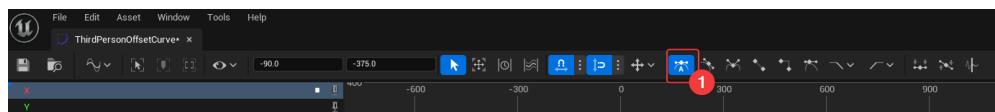


□ Target Offset Curve에 넣어줄 Curve를 만들어주자:

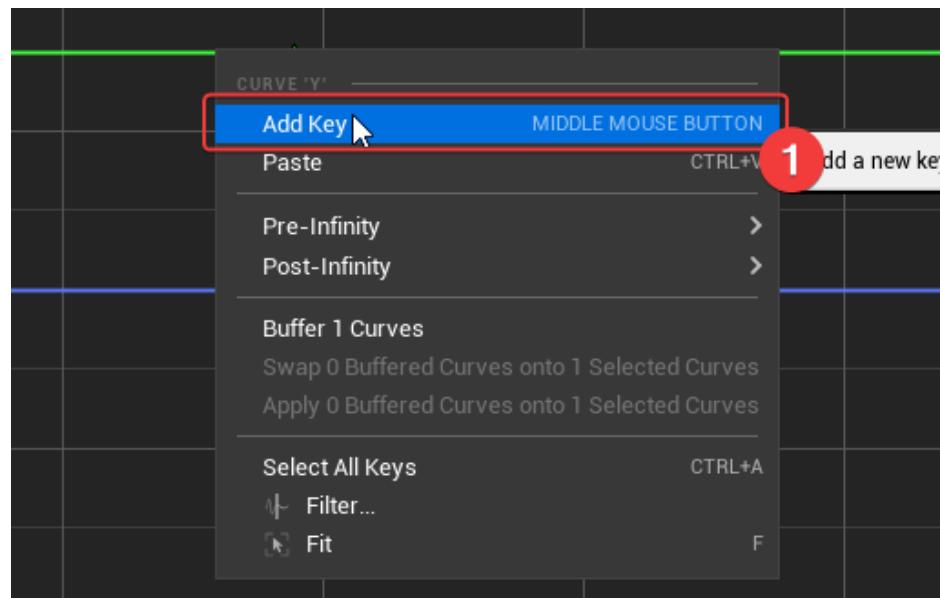




- 각 모든 Control Point는 Cubic Interpolation:



- Control Point 추가 방법은 아래와 같다:



- X:

- (-90, -375)
- (90, -250)
- (-40, -325) → 자연스러운 효과를 위해 추가적인 Control Point!

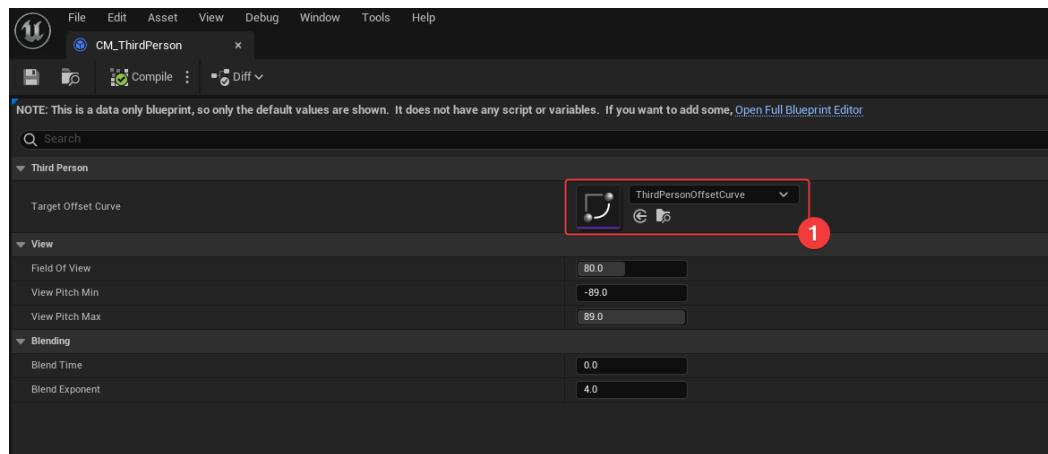
- Y:

- (-90, 60)

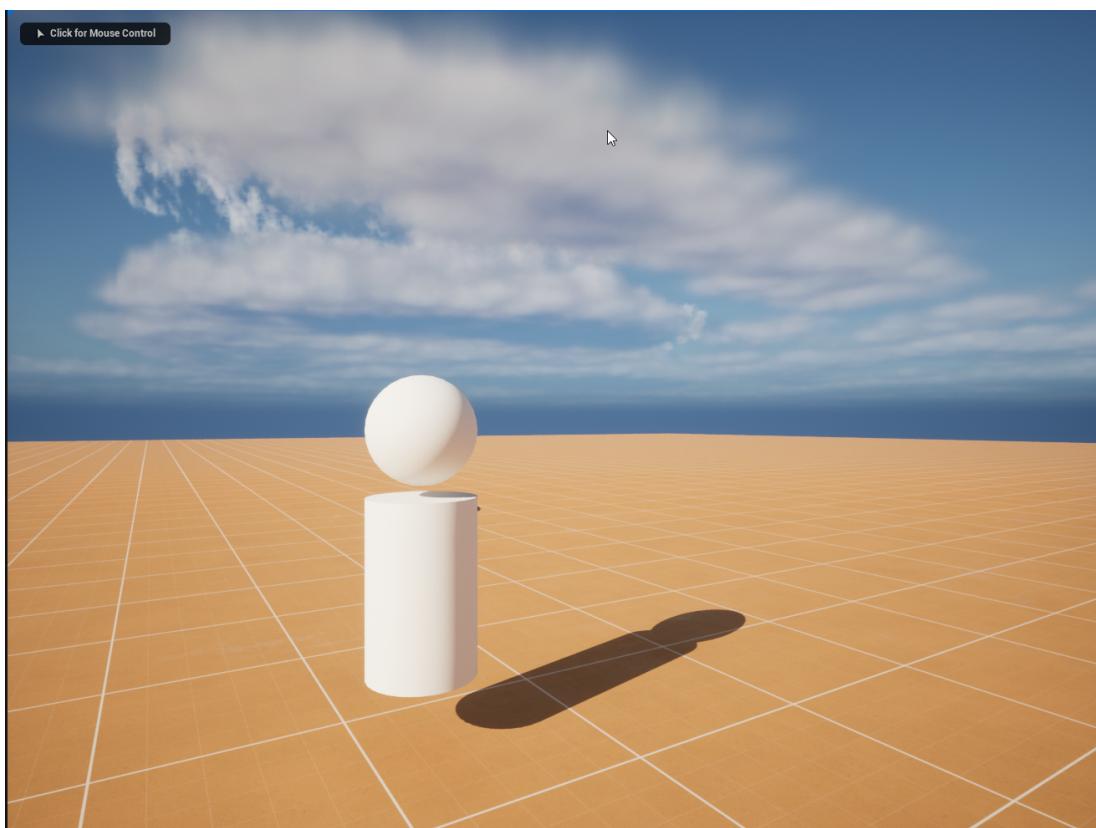
- Z:

- (-90, 60)
- (0, -10)

□ ThirdPersonOffsetCurve를 CM_ThirdPerson에 적용해주자:



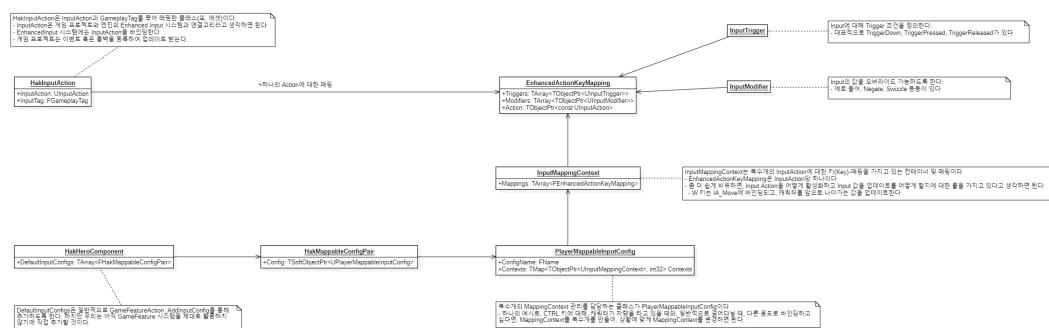
□ 그럼 아래와 같이 올바른 카메라 세팅으로 잘 렌더링 됨을 확인할 수 있다:



EnhancedInput

▼ 펼치기

□ EnhancedInput에 대한 간단한 설명:



□ GameplayTag 정의:

```
#pragma once

#include "Containers/UnrealString.h"
#include "Containers/Map.h"
#include "GameplayTagContainer.h"

/** forward declaration */
class UGameplayTagsManager;

/**
 * HakGameplayTags
 * - singleton containing native gameplay tags
 */
struct FHakGameplayTags
{
    /**
     * static methods
     */
    static const FHakGameplayTags& Get() { return GameplayTags; }
    static void InitializeNativeTags();

    /**
     * member methods
     */
    void AddTag(FGameplayTag& OutTag, const ANSICHAR* TagName, const ANSICHAR* TagComment);
    void AddAllTags(UGameplayTagsManager& Manager);

    /**
     * 아래의 GameplayTag는 초기화 과정 단계를 의미한다:
     * - GameInstance의 초기화 과정에 UGameFrameworkComponentManager의 RegisterInitState로 등록되어 선형적으로(linear)하게 업데이트 된다
     * - 이 초기화 GameplayTag는 게임의 Actor 사이에 공유되며, GameFrameworkInitStateInterface 상속받은 클래스는 초기화 상태(Init State)를 상태머신(State Machine)과 관리한다
     */
    FGameplayTag InitState_Spawned;
    FGameplayTag InitState_DataAvailable;
    FGameplayTag InitState_DataInitialized;
    FGameplayTag InitState_GameplayReady;

    /**
     * Enhanced Input Tags
     */
    FGameplayTag InputTag_Move;
    FGameplayTag InputTag_Look_Mouse;
};

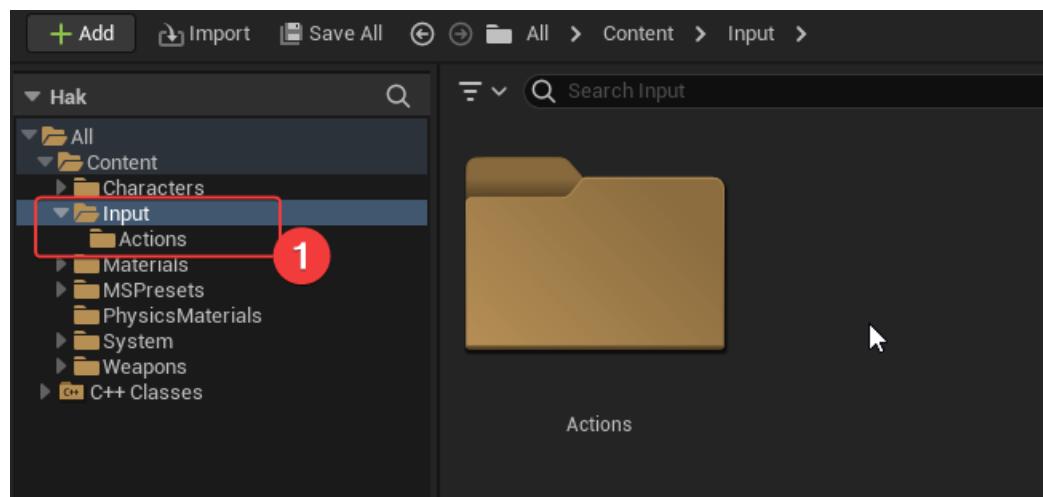
private:
    // static 변수 초기화는 .cpp에 해주는 것을 잊지 말기!
    static FHakGameplayTags GameplayTags;
};
```

```
void FHakGameplayTags::AddAllTags(UGameplayTagsManager& Manager)
{
    /**
     * GameFrameworkComponentManager init state tags
     */
    AddTag(InitState_Spawned, "InitState.Spawned", "1: Actor/Component has initially spawned and can be extended");
    AddTag(InitState_DataAvailable, "InitState.DataAvailable", "2: All required data has been loaded/replicated and is ready for initialization");
    AddTag(InitState_DataInitialized, "InitState.DataInitialized", "3: The available data has been initialized for this actor/component, but it is not yet fully initialized");
    AddTag(InitState_GameplayReady, "InitState.GameplayReady", "4: The actor/component is fully ready for active gameplay");

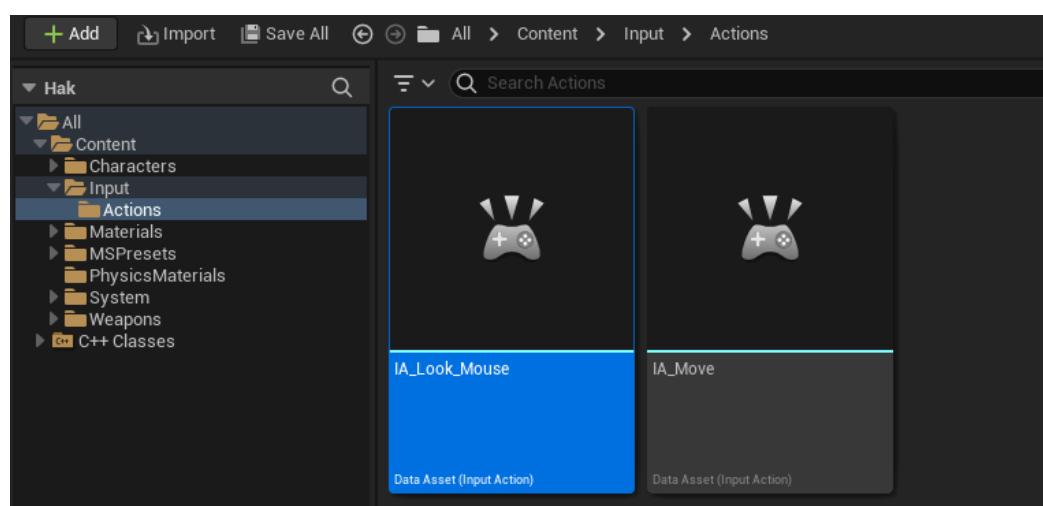
    /**
     * Enhanced Input Tags
     */
    AddTag(InputTag_Move, "InputTag.Move", "");
    AddTag(InputTag_Look_Mouse, "InputTag.Look.Mouse", "");
}
```

□ InputAction 에셋 생성:

- 폴더 생성:

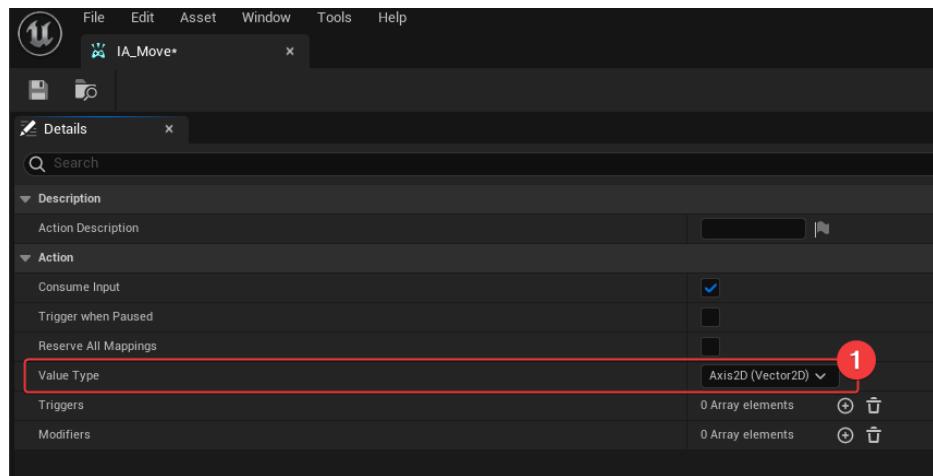


- IA_Move와 IA_LookMouse. 생성:

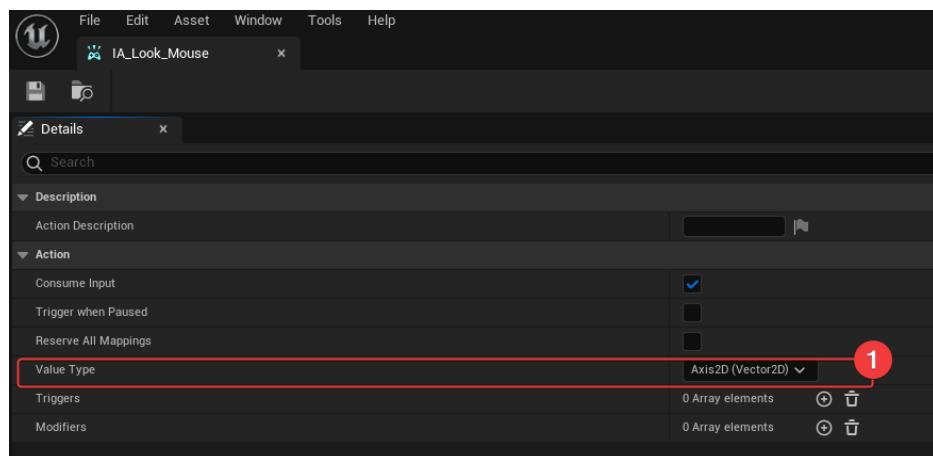


□ 각 InputAction에 대한 속성 값 설정:

- IA_Move:

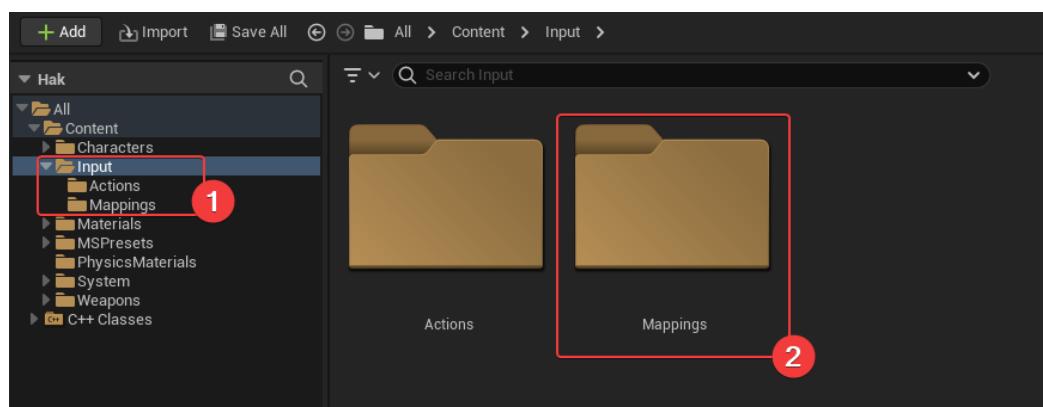


- IA_Look_Mouse:

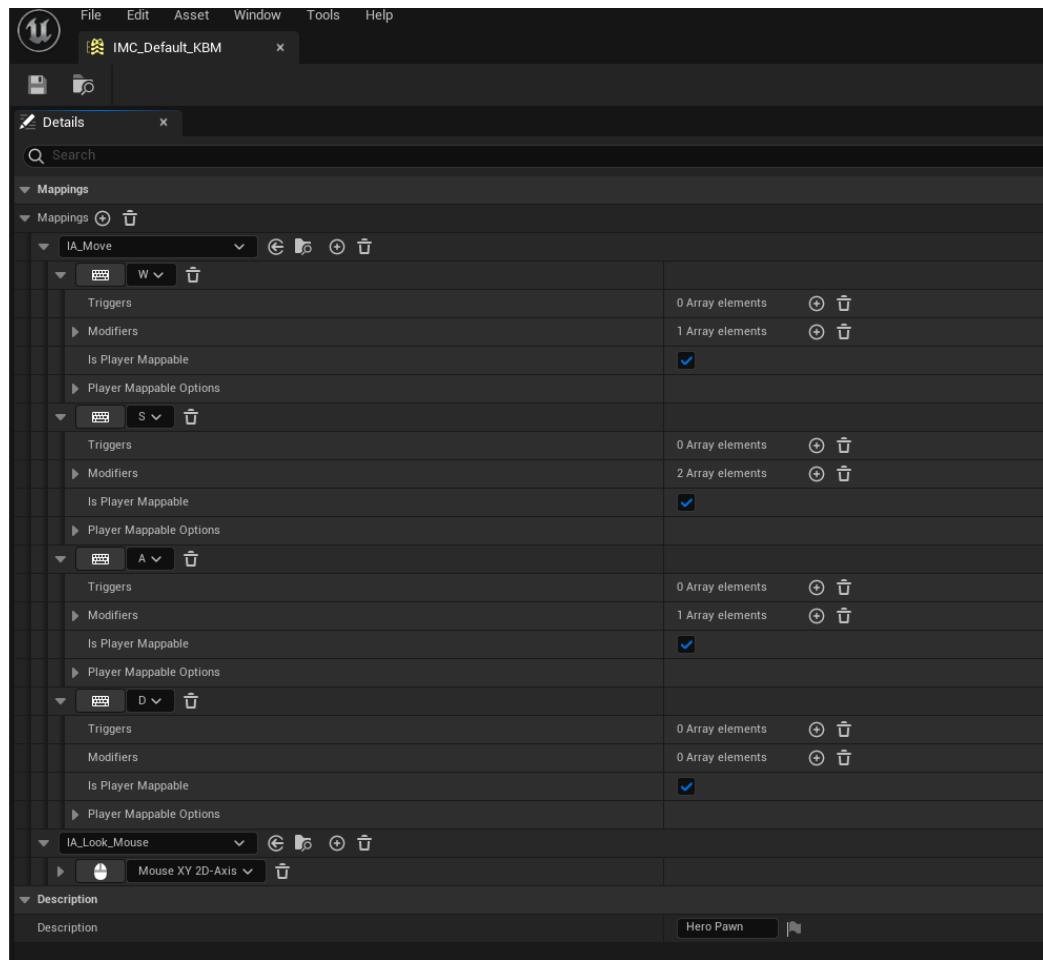


- Input Mapping Context 생성:

- 폴더 생성:

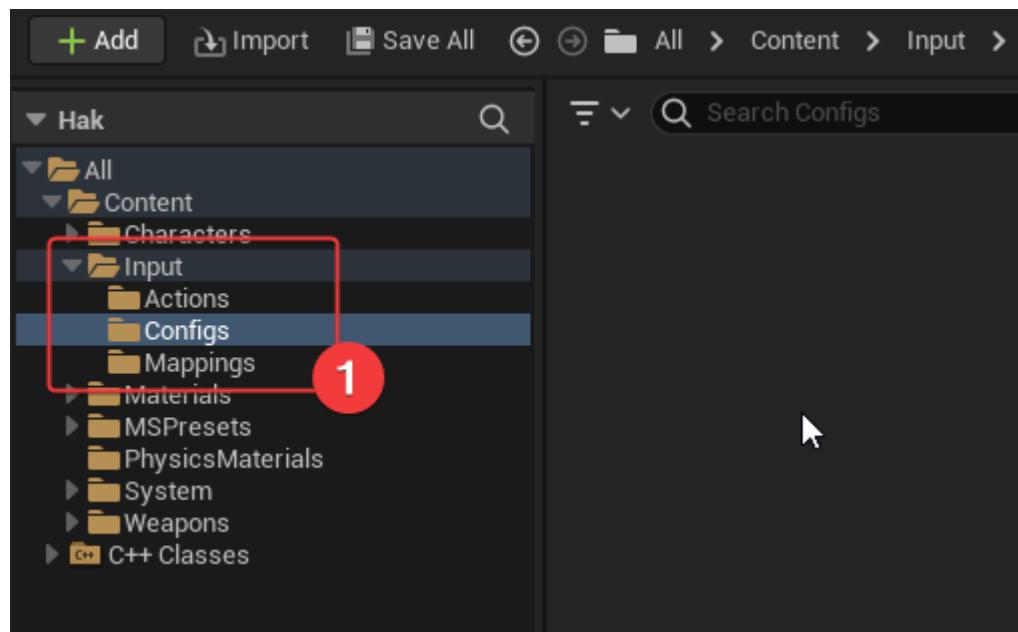


- IMC_Default_KBM 생성:

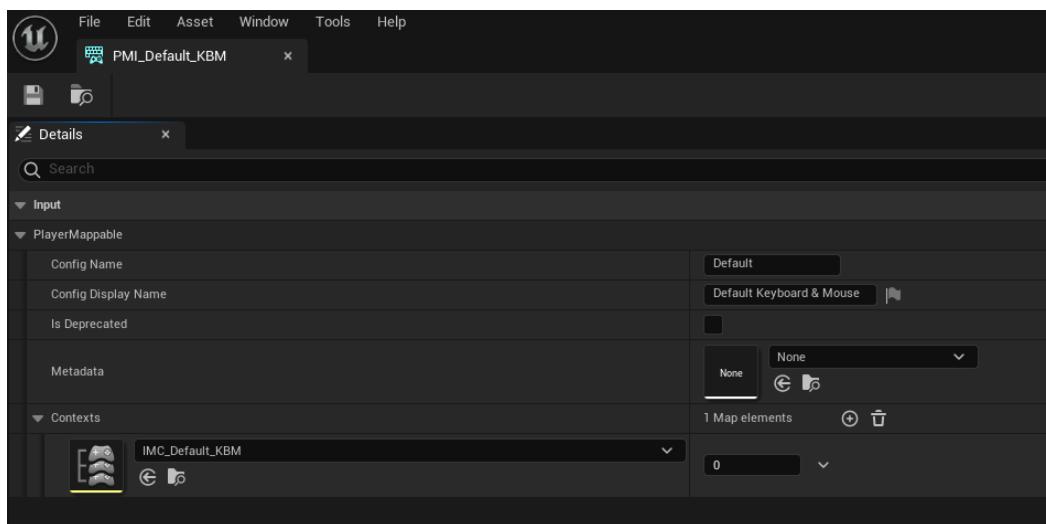
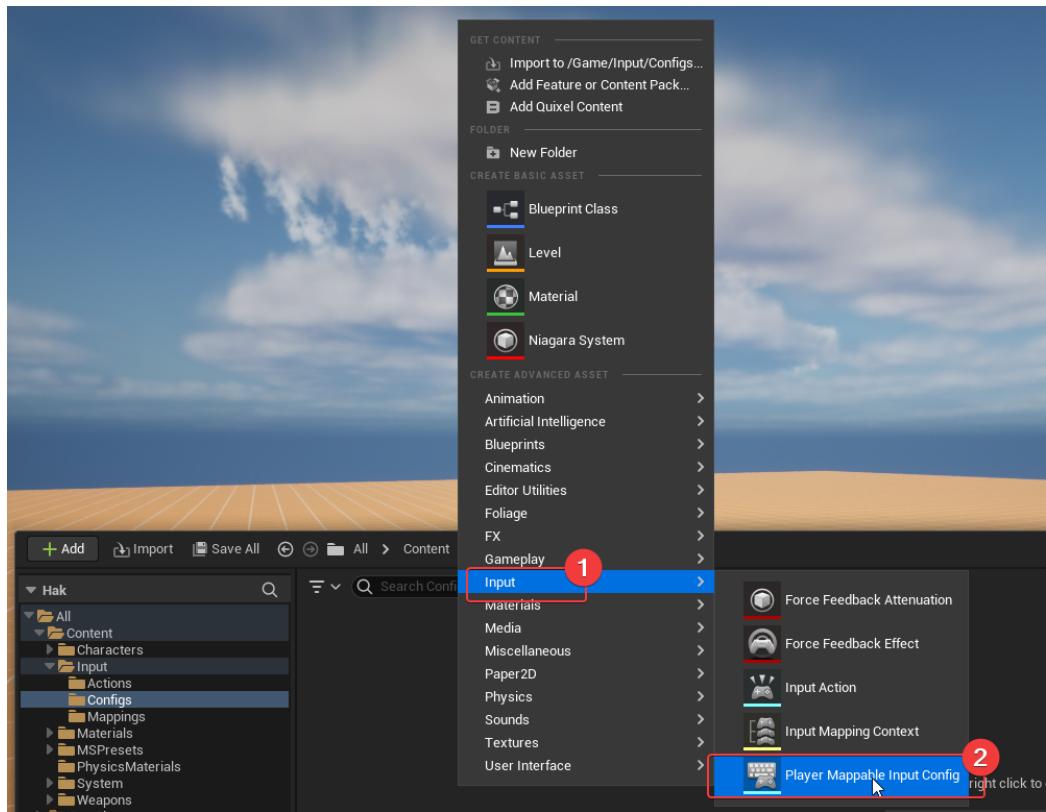


PlayerMappableInputConfig

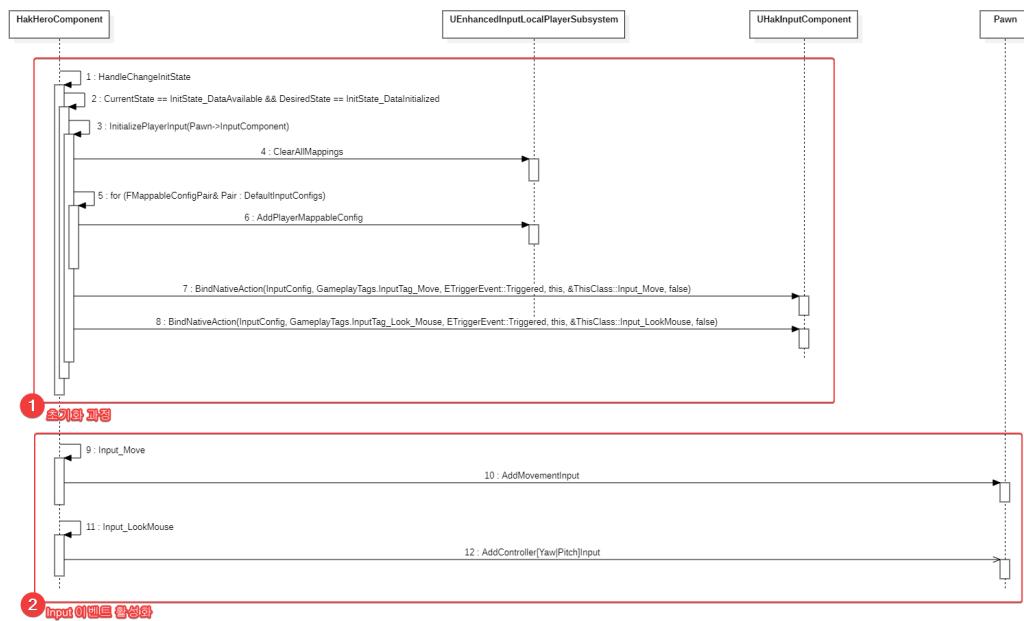
- 폴더 생성:



- PMI_Default_KBM 생성:

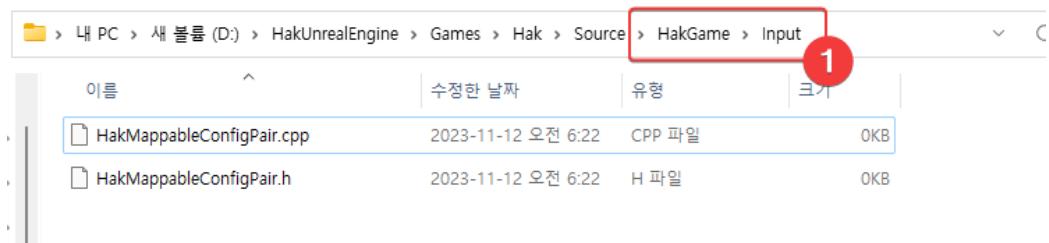


- 전체적인 Lyra에서 Input 초기화 및 업데이트 과정에 대한 순서도:



□ FHakMappableConfigPair 구현:

- .h/.cpp 생성:



- 꼭, Input 폴더 생성하기

- 코드 작성:

```

1 #pragma once
2
3 #include "HakMappableConfigPair.generated.h"
4
5 /** forward declaration */
6 class UPlayerMappableInputConfig;
7
8 USTRUCT()
9 struct FHakMappableConfigPair
10 {
11     GENERATED_BODY()
12 public:
13     FHakMappableConfigPair() = default;
14
15     UPROPERTY(EditAnywhere)
16     TSoftObjectPtr<UPlayerMappableInputConfig> Config;
17
18     UPROPERTY(EditAnywhere)
19     bool bShouldActivateAutomatically = true;
20 };

```

```

1 #include "HakMappableConfigPair.h"
2 #include "PlayerMappableInputConfig.h"
3 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakMappableConfigPair)

```

EnhancedInput 모듈 포함하기:

```

5 1 reference
6  public class HakGame : ModuleRules
7  {
8      0 references
9      public HakGame(ReadOnlyTargetRules Target) : base(Target)
10     {
11         PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;
12
13         PublicDependencyModuleNames.AddRange(new string[] {
14             "Core",
15             "CoreObject",
16             "Engine",
17             "InputCore",
18             // GAS
19             "GameplayTags",
20             // Game Features
21             "ModularGameplay",
22             "GameFeatures",
23             // Input
24             "InputCore",
25             "EnhancedInput",
26         });
27         1
28
29         PrivateDependencyModuleNames.AddRange(new string[] { });
30
31         // Uncomment if you are using Slate UI
32         // PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });
33
34         // Uncomment if you are using online features
35         // PrivateDependencyModuleNames.Add("OnlineSubsystem");
36
37         // To include OnlineSubsystemSteam, add it to the plugins section in your uproject file with the Enabled attribute set to true
38     }
39
40 }
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
746
746
747
748
748
749
749
750
751
752
753
754
755
756
756
757
758
758
759
759
760
761
762
763
764
765
766
766
767
768
768
769
769
770
771
772
773
774
775
776
776
777
778
778
779
779
780
781
782
783
784
785
786
786
787
788
788
789
789
790
791
792
793
794
795
796
796
797
798
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
846
847
848
848
849
849
850
851
852
853
854
855
856
856
857
858
858
859
859
860
861
862
863
864
865
866
866
867
868
868
869
869
870
871
872
873
874
875
876
876
877
878
878
879
879
880
881
882
883
884
885
886
886
887
888
888
889
889
890
891
892
893
894
895
895
896
897
897
898
898
899
899
900
901
902
903
904
905
905
906
907
907
908
908
909
909
910
911
912
913
913
914
914
915
915
916
916
917
917
918
918
919
919
920
921
922
923
923
924
924
925
925
926
926
927
927
928
928
929
929
930
931
932
933
933
934
934
935
935
936
936
937
937
938
938
939
939
940
941
942
943
943
944
944
945
945
946
946
947
947
948
948
949
949
950
951
952
953
953
954
954
955
955
956
956
957
957
958
958
959
959
960
961
962
962
963
963
964
964
965
965
966
966
967
967
968
968
969
969
970
971
972
972
973
973
974
974
975
975
976
976
977
977
978
978
979
979
980
981
982
982
983
983
984
984
985
985
986
986
987
987
988
988
989
989
990
991
992
992
993
993
994
994
995
995
996
996
997
997
998
998
999
999
1000
1000
1001
1001
1002
1002
1003
1003
1004
1004
1005
1005
1006
1006
1007
1007
1008
1008
1009
1009
1010
1010
1011
1011
1012
1012
1013
1013
1014
1014
1015
1015
1016
1016
1017
1017
1018
1018
1019
1019
1020
1020
1021
1021
1022
1022
1023
1023
1024
1024
1025
1025
1026
1026
1027
1027
1028
1028
1029
1029
1030
1030
1031
1031
1032
1032
1033
1033
1034
1034
1035
1035
1036
1036
1037
1037
1038
1038
1039
1039
1040
1040
1041
1041
1042
1042
1043
1043
1044
1044
1045
1045
1046
1046
1047
1047
1048
1048
1049
1049
1050
1050
1051
1051
1052
1052
1053
1053
1054
1054
1055
1055
1056
1056
1057
1057
1058
1058
1059
1059
1060
1060
1061
1061
1062
1062
1063
1063
1064
1064
1065
1065
1066
1066
1067
1067
1068
1068
1069
1069
1070
1070
1071
1071
1072
1072
1073
1073
1074
1074
1075
1075
1076
1076
1077
1077
1078
1078
1079
1079
1080
1080
1081
1081
1082
1082
1083
1083
1084
1084
1085
1085
1086
1086
1087
1087
1088
1088
1089
1089
1090
1090
1091
1091
1092
1092
1093
1093
1094
1094
1095
1095
1096
1096
1097
1097
1098
1098
1099
1099
1100
1100
1101
1101
1102
1102
1103
1103
1104
1104
1105
1105
1106
1106
1107
1107
1108
1108
1109
1109
1110
1110
1111
1111
1112
1112
1113
1113
1114
1114
1115
1115
1116
1116
1117
1117
1118
1118
1119
1119
1120
1120
1121
1121
1122
1122
1123
1123
1124
1124
1125
1125
1126
1126
1127
1127
1128
1128
1129
1129
1130
1130
1131
1131
1132
1132
1133
1133
1134
1134
1135
1135
1136
1136
1137
1137
1138
1138
1139
1139
1140
1140
1141
1141
1142
1142
1143
1143
1144
1144
1145
1145
1146
1146
1147
1147
1148
1148
1149
1149
1150
1150
1151
1151
1152
1152
1153
1153
1154
1154
1155
1155
1156
1156
1157
1157
1158
1158
1159
1159
1160
1160
1161
1161
1162
1162
1163
1163
1164
1164
1165
1165
1166
1166
1167
1167
1168
1168
1169
1169
1170
1170
1171
1171
1172
1172
1173
1173
1174
1174
1175
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1180
1181
1181
1182
1182
1183
1183
1184
1184
1185
1185
1186
1186
1187
1187
1188
1188
1189
1189
1190
1190
1191
1191
1192
1192
1193
1193
1194
1194
1195
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1200
1201
1201
1202
1202
1203
1203
1204
1204
1205
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1210
1211
1211
1212
1212
1213
1213
1214
1214
1215
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1220
1221
1221
1222
1222
1223
1223
1224
1224
1225
1225
1226
1226
1227
1227
1228
1228
1229
1229
1230
1230
1231
1231
1232
1232
1233
1233
1234
1234
1235
1235
1236
1236
1237
1237
1238
1238
1239
1239
1240
1240
1241
1241
1242
1242
1243
1243
1244
1244
1245
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1250
1251
1251
1252
1252
1253
1253
1254
1254
1255
1255
1256
1256
1257
1257
1258
1258
1259
1259
1260
1260
1261
1261
1262
1262
1263
1263
1264
1264
1265
1265
1266
1266
1267
1267
1268
1268
1269
1269
1270
1270
1271
1271
1272
1272
1273
1273
1274
1274
1275
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1280
1281
1281
1282
1282
1283
1283
1284
1284
1285
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1290
1291
1291
1292
1292
1293
1293
1294
1294
1295
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1300
1301
1301
1302
1302
1303
1303
1304
1304
1305
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1310
1311
1311
1312
1312
1313
1313
1314
1314
1315
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1320
1321
1321
1322
1322
1323
1323
1324
1324
1325
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1330
1331
1331
1332
1332
1333
1333
1334
1334
1335
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1340
1341
1341
1342
1342
1343
1343
1344
1344
1345
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1350
1351
1351
1352
1352
1353
1353
1354
1354
1355
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1360
1361
1361
1362
1362
1363
1363
1364
1364
1365
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1370
1371
1371
1372
1372
1373
1373
1374
1374
1375
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1380
1381
1381
1382
1382
1383
1383
1384
1384
1385
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1390
1391
1391
1392
1392
1393
1393
1394
1394
1395
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1400
1401
1401
1402
1402
1403
1403
1404
1404
1405
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1410
1411
1411
1412
1412
1413
1413
1414
1414
1415
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1420
1421
1421
1422
1422
1423
1423
1424
1424
1425
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1430
1431
1431
1432
1432
1433
1433
1434
1434
1435
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1440
1441
1441
1442
1442
1443
1443
1444
1444
1445
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1450
1451
1451
1452
1452
1453
1453
1454
1454
1455
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1460
1461
1461
1462
1462
1463
1463
1464
1464
1465
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1470
1471
1471
1472
1472
1473
1473
1474
1474
1475
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1480
1481
1481
1482
1482
1483
1483
1484
1484
1485
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1490
1491
1491
1492
1492
1493
1493
1494
1494
1495
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1500
1501
1501
1502
1502
1503
1503
1504
1504
1505
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1510
1511
1511
1512
1512
1513
1513
1514
1514
1515
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1520
1521
1521
1522
1522
1523
1523
1524
1524
1525
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1530
1531
1531
1532
1532
1533
1533
1534
1534
1535
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1540
1541
1541
1542
1542
1543
1543
1544
1544
1545
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1550
1551
1551
1552
1552
1553
1553
1554
1554
1555
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1560
1561
1561
1562
1562
1563
1563
1564
1564
1565
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1570
1571
1571
1572
1572
1573
1573
1574
1574
1575
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1580
1581
1581
1582
1582
1583
1583
1584
1584
1585
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1590
1591
1591
1592
1592
1593
1593
1594
1594
1595
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1600
1601
1601
1602
1602
1603
1603
1604
1604
1605
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1610
1611
1611
1612
1612
1613
1613
1614
1614
1615
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1620
1621
1621
1622
1622
1623
1623
```

```

1  #pragma once
2
3 #include "Components/PawnComponent.h"
4 #include "Components/GameFrameworkInitStateInterface.h"
5 #include "HakHeroComponent.generated.h"
6
7 /** forward declaration */
8 class UHakCameraMode;
9 template<class TClass> class TSubclassOf;
10 struct FHakMappableConfigPair;
11
12 /**
13 * component that sets up input and camera handling for player controlled pawns (or bots that simulate players)
14 * - this depends on a PawnExtensionComponent to coordinate initialization
15 *
16 * 카메라, 입력 등 플레이어가 제어하는 시스템의 초기화를 처리하는 컴포넌트
17 */
18 UCLASS(Blueprintable, Meta=(BlueprintSpawnableComponent))
19 class UHakHeroComponent : public UPawnComponent, public IGameFrameworkInitStateInterface
20 {
21     GENERATED_BODY()
22     public:
23         UHakHeroComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
24
25         /** FeatureName 정의 */
26         static const FName NAME_ActorFeatureName;
27
28         /**
29         * UPawnComponent interface
30         */
31         virtual void OnRegister() final;
32         virtual void BeginPlay() final;
33         virtual void EndPlay(const EEndPlayReason::Type EndPlayReason) final;
34
35         /**
36         * IGameFrameworkInitStateInterface
37         */
38         virtual FName GetFeatureName() const final { return NAME_ActorFeatureName; }
39         virtual void OnActorInitStateChanged(const FActorInitStateChangedParams& Params) final;
40         virtual bool CanChangeInitState(UGameFrameworkComponentManager* Manager, FGameplayTag CurrentState, FGameplayTag DesiredState) const final;
41         virtual void HandleChangeInitState(UGameFrameworkComponentManager* Manager, FGameplayTag CurrentState, FGameplayTag DesiredState) final;
42         virtual void CheckDefaultInitialization() final;
43
44         /**
45         * member methods
46         */
47         TSubclassOf<UHakCameraMode> DetermineCameraMode() const;
48
49         /**
50         * member variables
51         */
52         UPROPERTY(EditAnywhere)
53         TArray<FHakMappableConfigPair> DefaultInputConfigs;
54 };

```

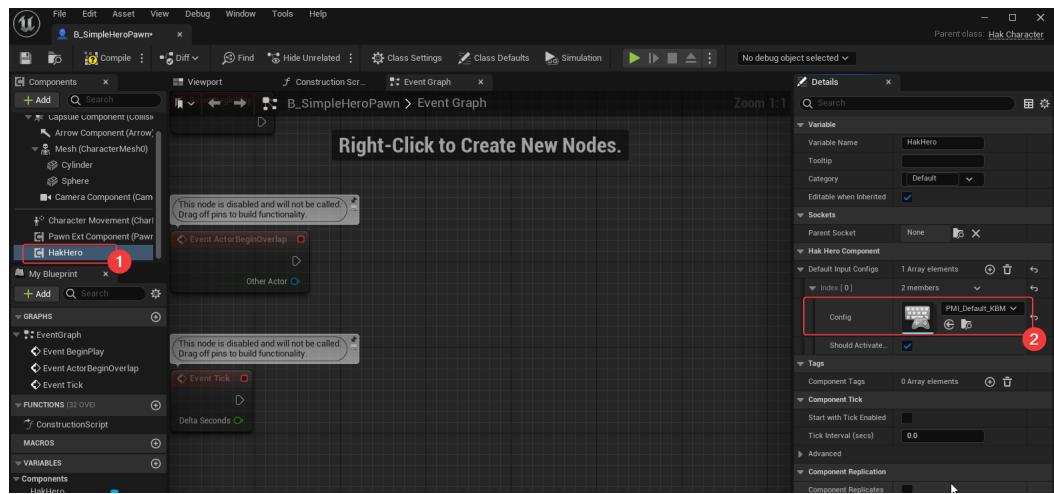
헤더 파일을 추가해주기:

```

1  #include "HakHeroComponent.h"
2  #include "HakPawnExtensionComponent.h"
3  #include "HakGame/HakLogChannels.h"
4  #include "HakGame/HakGameplayTags.h"
5  #include "HakGame/Player/HakPlayerState.h"
6  #include "HakGame/Character/HakPawnData.h"
7  #include "HakGame/Camera/HakCameraComponent.h"
8  #include "HakGame/Input/HakMappableConfigPair.h" 1
9  #include "Components/GameFrameworkComponentManager.h"
10 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakHeroComponent)
11
12 // FeatureName 정의, static member variables 추가하기

```

HeroComponent에서 DefaultInputConfigs를 추가해주기:



□ HakInputConfig.h/.cpp 구현:

```

#pragma once

#include "Containers/Array.h"
#include "GameplayTagContainer.h"
#include "Engine/DataAsset.h"
#include "HakInputConfig.generated.h"

/** forward declaration */
class UInputAction;

/** HakInputAction은 GameplayTag와 InputAction을 연결하는 래퍼 클래스이다 */
USTRUCT(BlueprintType)
struct FHakInputAction
{
    GENERATED_BODY()
public:
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly)
    TObjectPtr<const UInputAction> InputAction = nullptr;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Meta=(Categories="InputTag"))
    FGameplayTag InputTag;
};

/** 
 * HakInputConfig는 활성화 가능한 InputAction을 가지고 있는 클래스이다
 * - 이는 코드를 보면서, 무슨 설명인지 와닿을 것이다
 */
UCLASS(BlueprintType)
class UHakInputConfig : public UDataAsset
{
    GENERATED_BODY()
public:
    UHakInputConfig(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    const UInputAction* FindNativeInputActionForTag(const FGameplayTag& InputTag, bool bLogNotFound = true) const;
    const UInputAction* FindAbilityInputActionForTag(const FGameplayTag& InputTag, bool bLogNotFound = true) const;

    /**
     * member variables
     */
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Meta=(TitleProperty="InputAction"))
    TArray<FHakInputAction> NativeInputActions;

    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Meta = (TitleProperty = "InputAction"))
    TArray<FHakInputAction> AbilityInputActions;
};

```

```

#include "HakInputConfig.h"
#include "HakGame/HakLogChannels.h"

#include UE_INLINE_GENERATED_CPP_BY_NAME(HakInputConfig)

UHakInputConfig::UHakInputConfig(const FObjectInitializer& ObjectInitializer)
: Super(ObjectInitializer)
{ }

const UInputAction* UHakInputConfig::FindNativeInputActionForTag(const FGameplayTag& InputTag, bool bLogNotFound) const
{
    // NativeInputActions를 순회하며, Input으로 들어온 InputTag가 있는지 체크한다:
    // - 있으면, 그에 따른 InputAction을 반환하지만, 없다면, 그냥 nullptr을 반환한다.
    for (const FHakInputAction& Action : NativeInputActions)
    {
        if (Action.InputAction && (Action.InputTag == InputTag))
        {
            return Action.InputAction;
        }
    }

    if (bLogNotFound)
    {
        UE_LOG(LogHak, Error, TEXT("can't find NativeInputAction for InputTag [%s] on InputConfig [%s]."), *InputTag.ToString(), *GetNameSafe(this));
    }

    return nullptr;
}

const UInputAction* UHakInputConfig::FindAbilityInputActionForTag(const FGameplayTag& InputTag, bool bLogNotFound) const
{
    for (const FHakInputAction& Action : AbilityInputActions)
    {
        if (Action.InputAction && (Action.InputTag == InputTag))
        {
            return Action.InputAction;
        }
    }

    if (bLogNotFound)
    {
        UE_LOG(LogHak, Error, TEXT("Can't find AbilityInputAction for InputTag [%s] on InputConfig [%s]."), *InputTag.ToString(), *GetNameSafe(this));
    }

    return nullptr;
}

```

□ PawnData에 추가시켜주자:

```

/**
 * UHakPawnData
 * - non-mutable data asset that contains properties used to define a pawn
 */
UCLASS(BlueprintType)
class UHakPawnData : public UPrimaryDataAsset
{
    GENERATED_BODY()
public:
    UHakPawnData(const FObjectInitializer& ObjectInitializer);

    /** @TODO - 일단 단순히 클래스의 형태만 만들어놓도록 하자 */

    /** Pawn의 Class */
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category="Hak|Pawn")
    TSubclassOf<APawn> PawnClass;

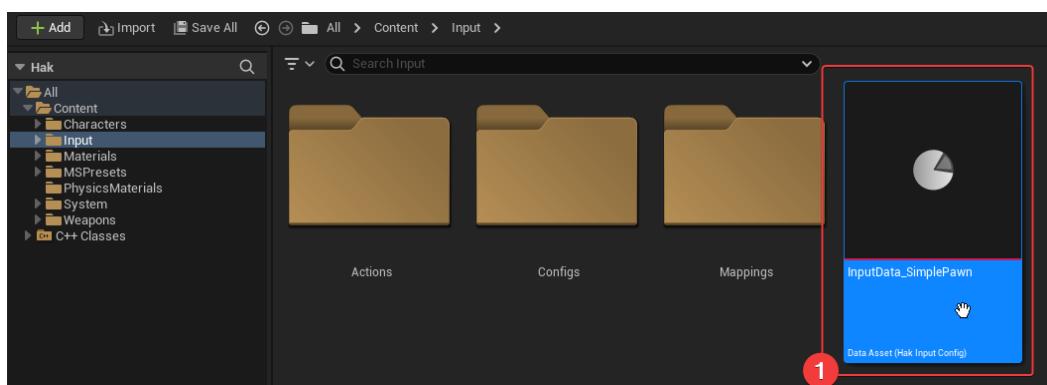
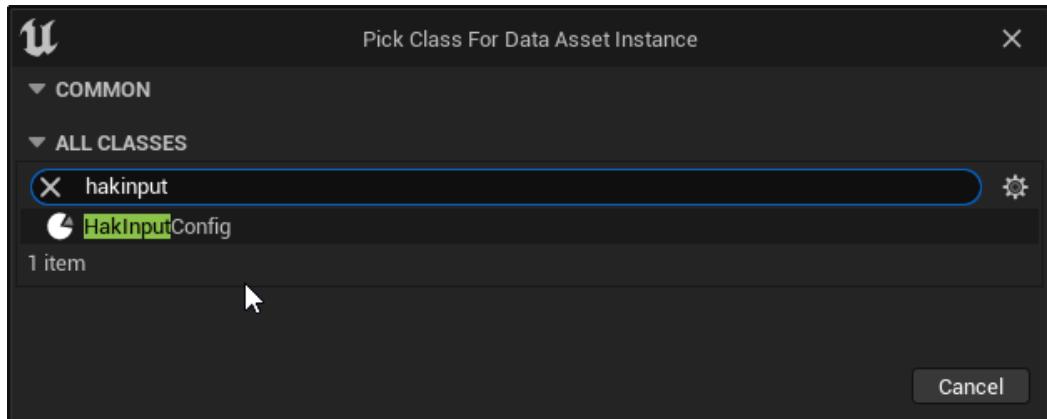
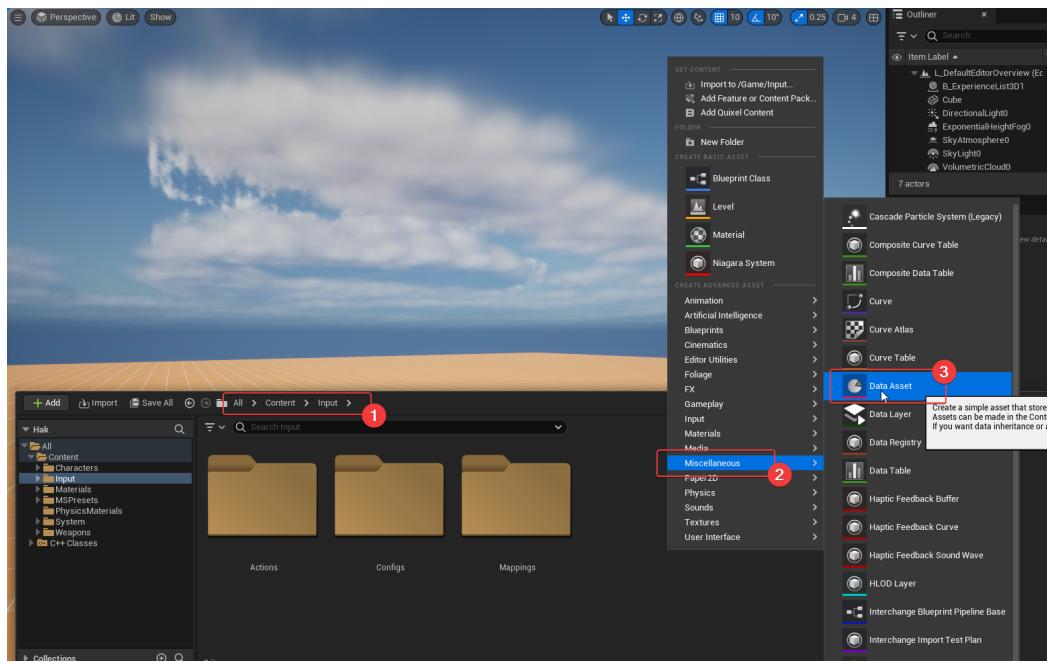
    /** Camera Mode */
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category="Hak|Camera")
    TSubclassOf<UHakCameraMode> DefaultCameraMode;

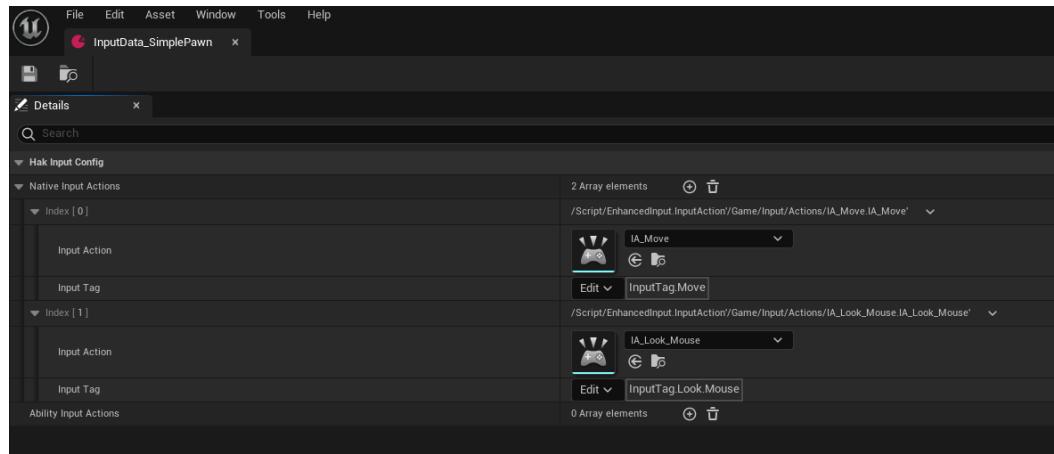
    /** input configuration used by player controlled pawns to create input mappings and bind input actions */
    UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category="Hak|InputConfig")
    TObjectPtr<UHakInputConfig> InputConfig;
};

```

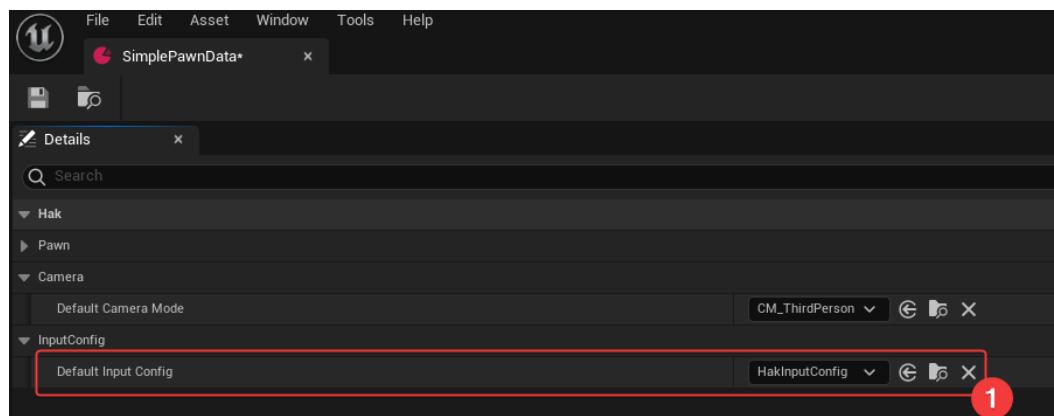
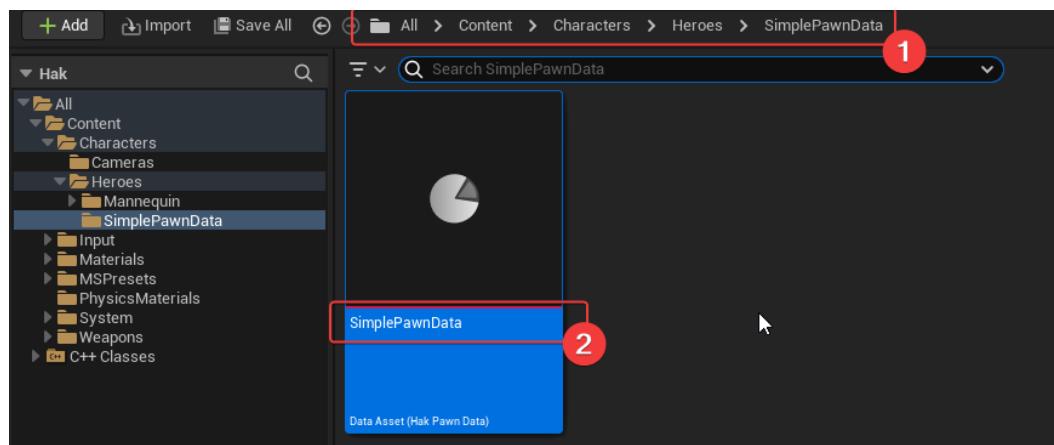
- InputConfig는 허용된 InputAction을 들고 있다.
- 이는 PawnData에 초기화 값으로서 존재한다.

□ InputConfig 에셋을 만들자:





□ SimplePawnData에 업데이트 해주자:



□ HakInputComponent.h/.cpp 구현:

```

#pragma once
#include "EnhancedInputComponent.h"
#include "InputTriggers.h"
#include "InputActionValue.h"
#include "HakInputConfig.h"
#include "HakInputComponent.generated.h"

UCLASS()
class UHakInputComponent : public UEnhancedInputComponent
{
    GENERATED_BODY()
public:
    UHakInputComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
    /**
     * Member methods
     */
    template <class UserClass, typename FuncType>
    void BindNativeAction(const UHakInputConfig* InputConfig, const FGameplayTag& InputTag, ETriggerEvent TriggerEvent, UserClass* Object, FuncType Func, bool bLogIfNotFound);
    template <class UserClass, typename PressedFuncType, typename ReleasedFuncType>
    void BindAbilityActions(const UHakInputConfig* InputConfig, UserClass* Object, PressedFuncType PressedFunc, ReleasedFuncType ReleasedFunc, TArray<uint32>& BindHandles);
};

template <class UserClass, typename FuncType>
void UHakInputComponent::BindNativeAction(const UHakInputConfig* InputConfig, const FGameplayTag& InputTag, ETriggerEvent TriggerEvent, UserClass* Object, FuncType Func, bool bLogIfNotFound)
{
    check(InputConfig);
    // 여기서 알 수 있듯이, InputConfig는 활성화 가능한 InputAction을 담고 있다.
    // 만약 InputConfig에 있는 InputAction은 Binding시키면, nullptr를 반환하여, 바인딩하는데 실패한다!
    if (const UHakInputAction* IA = InputConfig->FindNativeInputActionForTag(InputTag, bLogIfNotFound))
    {
        BindAction(IA, TriggerEvent, Object, Func);
    }
}

template <class UserClass, typename PressedFuncType, typename ReleasedFuncType>
void UHakInputComponent::BindAbilityActions(const UHakInputConfig* InputConfig, UserClass* Object, PressedFuncType PressedFunc, ReleasedFuncType ReleasedFunc, TArray<uint32>& BindHandles)
{
    check(InputConfig);
    // AbilityAction에 대해서는 그냥 모든 InputAction에 다 바인딩 시킨다!
    for (const FHakInputActions::Action Action : InputConfig->AbilityInputActions)
    {
        if (Action.InputAction && Action.InputTag.IsValid())
        {
            if (PressedFunc)
            {
                BindHandles.Add(BindAction(Action.InputAction, ETriggerEvent::Triggered, Object, PressedFunc, Action.InputTag.GetHandle()));
            }

            if (ReleasedFunc)
            {
                BindHandles.Add(BindAction(Action.InputAction, ETriggerEvent::Completed, Object, ReleasedFunc, Action.InputTag.GetHandle()));
            }
        }
    }
}

```

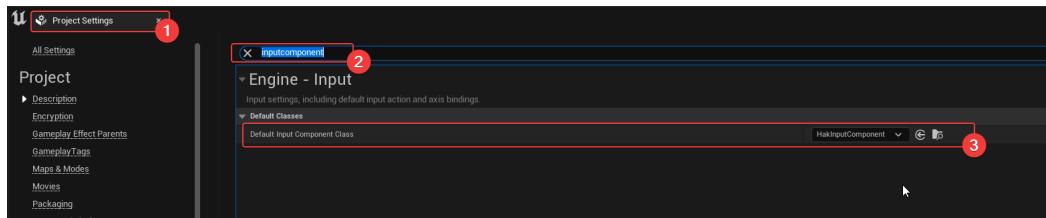
```

#include "HakInputComponent.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(HakInputComponent)

UHakInputComponent::UHakInputComponent(const FObjectInitializer& ObjectInitializer)
    : Super(ObjectInitializer)
{
}

```

InputComponent 오버라이드:

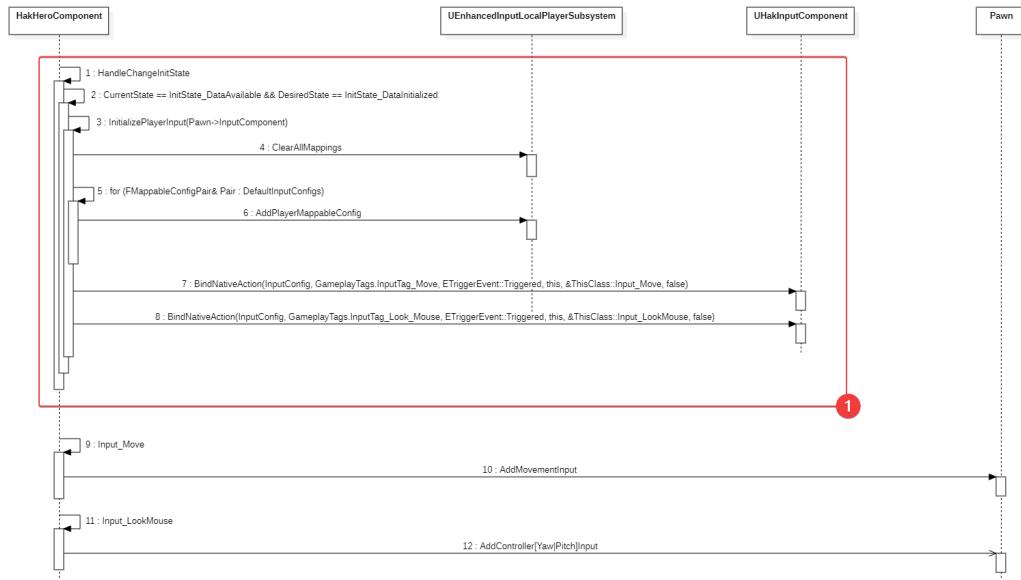


주석에 대한 설명하기

HeroComponent에 Input을 추가하자

▼ 펼치기

Input을 초기화 해주자:



HeroComponent::HandleChangeInitState()

```

void UHakHeroComponent::HandleChangeInitState(UGameFrameworkComponentManager* Manager, FGameplayTag CurrentState, FGameplayTag DesiredState)
{
    const FHakGameplayTags& InitTags = FHakGameplayTags::Get();

    // DataAvailable -> DataInitialized 단계
    if (CurrentState == InitTags.InitState_DataAvailable && DesiredState == InitTags.InitState_DataInitialized)
    {
        APawn* Pawn = GetPawn<APawn>();
        AHakPlayerState* HakPS = GetPlayerState<AHakPlayerState>();
        if (!ensure(Pawn && HakPS))
        {
            return;
        }

        // Input과 Camera에 대한 핸들링... (TODO)

        const bool bIsLocallyControlled = Pawn->IsLocallyControlled();
        const UHakPawnData* PawnData = nullptr;
        if (UHakPawnExtensionComponent* PawnExtComp = UHakPawnExtensionComponent::FindPawnExtensionComponent(Pawn))
        {
            PawnData = PawnExtComp->GetPawnData<UHakPawnData>();
        }

        if (bIsLocallyControlled && PawnData)
        {
            // 현재 HakCharacter이 Attach된 CameraComponent를 찾음
            if (UHakCameraComponent* CameraComponent = UHakCameraComponent::FindCameraComponent(Pawn))
            {
                CameraComponent->DetermineCameraModeDelegate.BindUObject(this, &ThisClass::DetermineCameraMode);
            }
        }

        if (AHakPlayerController* HakPC = GetController<AHakPlayerController>())
        {
            if (Pawn->InputComponent != nullptr)
            {
                InitializePlayerInput(Pawn->InputComponent);
            }
        }
    }
}

```

1 InitializePlayerInput() 구현하자

HeroComponent::InitializePlayerInput()

- HakInputComponent.h 헤더 파일 포함시키기
- #include "EnhancedInputSubsystems.h" 포함시키기
- 코드 구현 및 설명:

```

void UHakHeroComponent::InitializePlayerInput(UInputComponent* PlayerInputComponent)
{
    check(PlayerInputComponent);

    const APawn* Pawn = GetPawn<APawn>();
    if (!Pawn)
    {
        return;
    }

    // LocalPlayer를 가져오기 위해
    const APlayerController* PC = GetController<APlayerController>();
    check(PC);

    // EnhancedInputLocalPlayerSubsystem 가져오기 위해
    const ULocalPlayer* LP = PC->GetLocalPlayer();
    check(LP);

    UEnhancedInputLocalPlayerSubsystem* Subsystem = LP->GetSubsystem<UEnhancedInputLocalPlayerSubsystem>();
    check(Subsystem);

    // EnhancedInputLocalPlayerSubsystem의 MappingContext를 비워준다:
    Subsystem->ClearAllMappings();

    // PawnExtensionComponent -> PawnData -> InputConfig 존재 유무 판단:
    if (const UHakPawnExtensionComponent* PawnExtComp = UHakPawnExtensionComponent::FindPawnExtensionComponent(Pawn))
    {
        if (const UHakPawnData* PawnData = PawnExtComp->GetPawnData<UHakPawnData>())
        {
            if (const UHakInputConfig* InputConfig = PawnData->InputConfig)
            {
                const FHakGameplayTags& GameplayTags = FHakGameplayTags::Get();

                // HeroComponent 가지고 있는 Input Mapping Context를 순회하며, EnhancedInputLocalPlayerSubsystem에 추가한다
                for (const FHakMappableConfigPair& Pair : DefaultInputConfigs)
                {
                    if (Pair.bShouldActivateAutomatically)
                    {
                        FModifyContextOptions Options = {};
                        Options.IgnoreAllPressedKeysUntilRelease = false;

                        // 내부적으로 Input Mapping Context를 추가한다:
                        // - AddPlayerMappableConfig를 간단히 보는 것을 추천
                        Subsystem->AddPlayerMappableConfig(Pair.Config.LoadSynchronous(), Options);
                    }
                }

                UHakInputComponent* HakIC = CastChecked<UHakInputComponent>(PlayerInputComponent);
                {
                    // InputTag_Move와 InputTag_Look_Mouse에 대해 각각 Input_Move()와 Input_LookMouse() 명령 함수에 바인딩시킨다:
                    // - 바인딩 이후, Input 이벤트에 따라 명령 함수가 트리거된다
                    HakIC->BindNativeAction(InputConfig, GameplayTags.InputTag_Move, ETriggerEvent::Triggered, this, &ThisClass::Input_Move, false);
                    HakIC->BindNativeAction(InputConfig, GameplayTags.InputTag_Look_Mouse, ETriggerEvent::Triggered, this, &ThisClass::Input_LookMouse, false);
                }
            }
        }
    }
}

```

□ Input_Move()와 Input_LookMouse() 구현:



- `Input_Move()`:

```
void UHakHeroComponent::Input_Move(const FInputActionValue& InputActionValue)
{
    APawn* Pawn = GetPawn<APawn>();
    AController* Controller = Pawn ? Pawn->GetController() : nullptr;

    if (Controller)
    {
        const FVector2D Value = InputActionValue.Get<FVector2D>();
        const FRotator MovementRotation(0.0f, Controller->GetControlRotation().Yaw, 0.0f);

        if (Value.X != 0.0f)
        {
            // Left/Right -> X 값에 들어있음:
            // MovementDirection은 현재 카메라의 RightVector를 의미함 (World-Space)
            const FVector MovementDirection = MovementRotation.RotateVector(FVector::RightVector);

            // AddMovementInput 함수를 한번 보자:
            // - 내부적으로 MovementDirection * Value.X를 MovementComponent에 적용(더하기)해준다
            Pawn->AddMovementInput(MovementDirection, Value.X);
        }

        if (Value.Y != 0.0f)
        {
            // 앞서 Left/Right와 마찬가지로 Forward/Backward를 적용한다
            const FVector MovementDirection = MovementRotation.RotateVector(FVector::ForwardVector);
            Pawn->AddMovementInput(MovementDirection, Value.Y);
        }
    }
}
```

- Input_LookMouse():

```
void UHakHeroComponent::Input_LookMouse(const FInputActionValue& InputActionValue)
{
    APawn* Pawn = GetPawn<APawn>();
    if (!Pawn)
    {
        return;
    }

    const FVector2D Value = InputActionValue.Get<FVector2D>();
    if (Value.X != 0.0f)
    {
        // X에는 Yaw 값이 있음:
        // - Camera에 대해 Yaw 적용
        Pawn->AddControllerYawInput(Value.X);
    }

    if (Value.Y != 0.0f)
    {
        // Y에는 Pitch 값!
        double AimInversionValue = -Value.Y;
        Pawn->AddControllerPitchInput(AimInversionValue);
    }
}
```

- 헤더파일 추가 잊지 말기:

```
#include "HakHeroComponent.h"
#include "HakPawnExtensionComponent.h"
#include "EnhancedInputSubsystems.h"
#include "PlayerMappableInputConfig.h" 1
#include "HakGame/HakLogChannels.h"
#include "HakGame/HakGameplayTags.h"
#include "HakGame/Player/HakPlayerState.h"
#include "HakGame/Player/HakPlayerController.h"
#include "HakGame/Character/HakPawnData.h"
#include "HakGame/Camera/HakCameraComponent.h"
#include "HakGame/Input/HakMappableConfigPair.h"
#include "HakGame/Input/HakInputComponent.h"
#include "Components/GameFrameworkComponentManager.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(HakHeroComponent)
```

- 잘 적용됨을 확인할 수 있다:

https://prod-files-secure.s3.us-west-2.amazonaws.com/ecba3054-6b52-40da-ba34-e88eb287722c/22107c65-8763-415a-b69f-fdcccfb0868c/UnrealEditor_yn8RDeS0pd.mp4

자료

▼ 펼치기

[PawnExtension_Hero.mdj](#)