



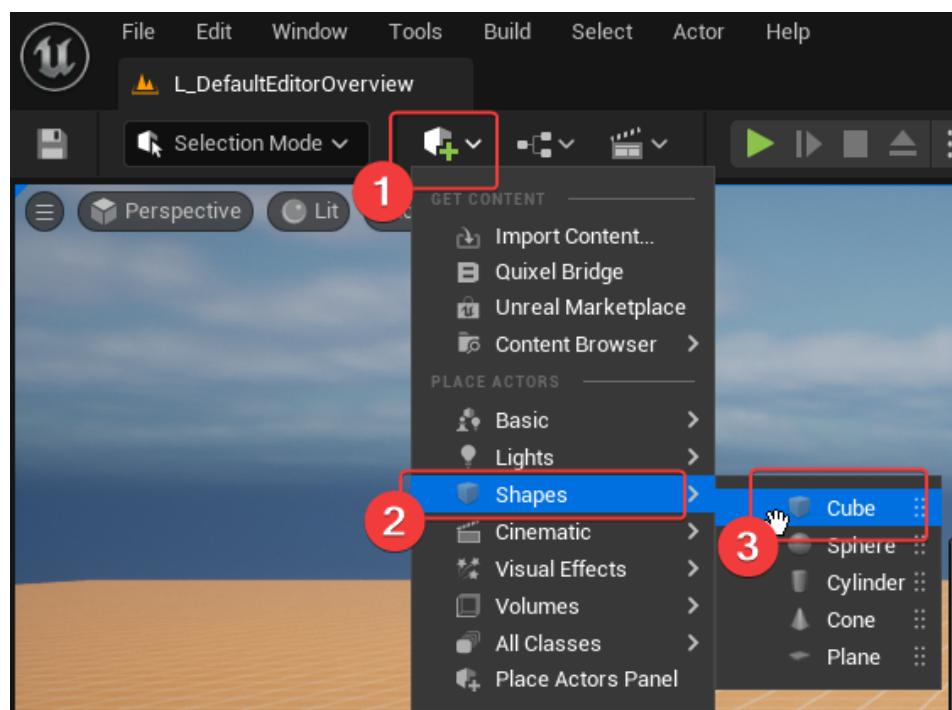
9주차 (2023.11.20)

B_TeleportToUserFacingExperience

▼ 펼치기

우선 간단한 외벽을 하나를 만들어주자:

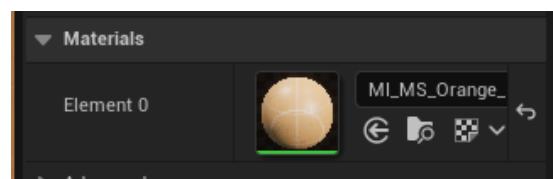
Cube Shape를 추가해주자:



- 해당 Cube 속성값들을 아래와 같이 맞추어주자:



- 그리고 Material도 업데이트해주자:



- 간단히 이에 맞추어 Platform Cube도 Scale을 재조정해주자:

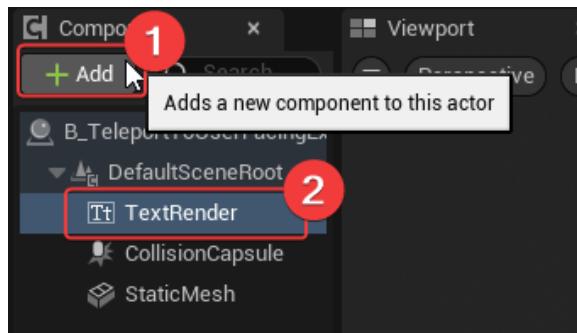


[B_TeleportToUserFacingExperience]

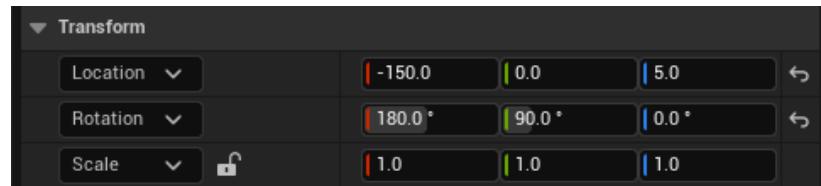
현재 해당 BP의 우리의 구현은 미-구현 상태이다:

- 앞서, 우리의 기억을 회고해보면, 캐릭터가 해당 BP에 설정된 Capsule Collision에 Overlap이 일어나면 알맞는 Experience로서 Teleport 하는 것을 기대한다

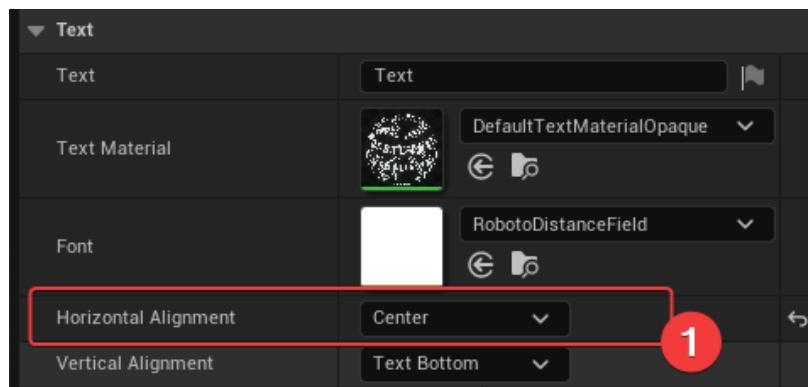
- 우선 해당 Platform 메시에 대해 어떤 Experience가 염여 있는지 간단히 TextRender로 만들자



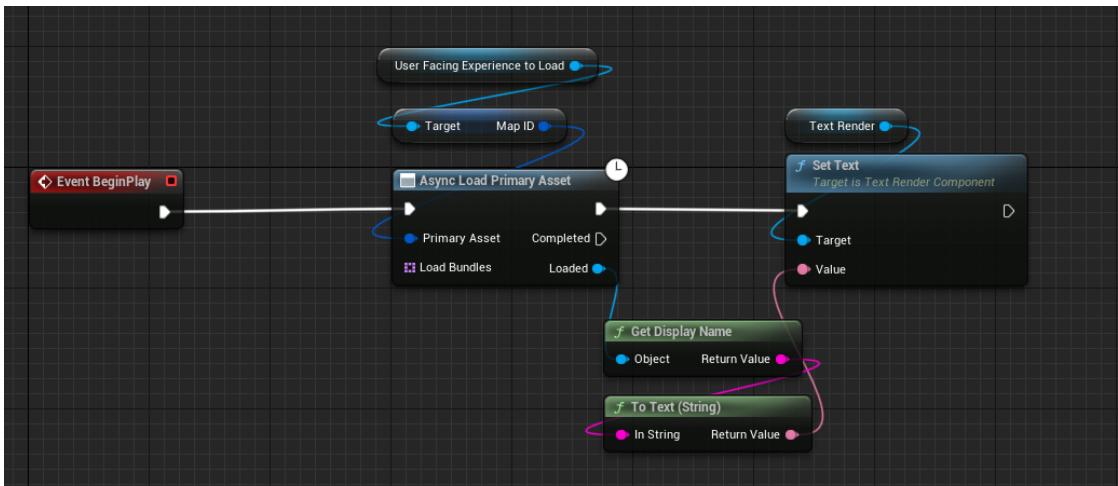
- 아래와 같이 Transform을 설정해주자:



- Text Alignment도 Center로 변경하자:



- 그럼 TextRender의 Text를 BeginPlay에서 변경해주자:



□ 그럼 아래와 같이 손쉽게 나온다:

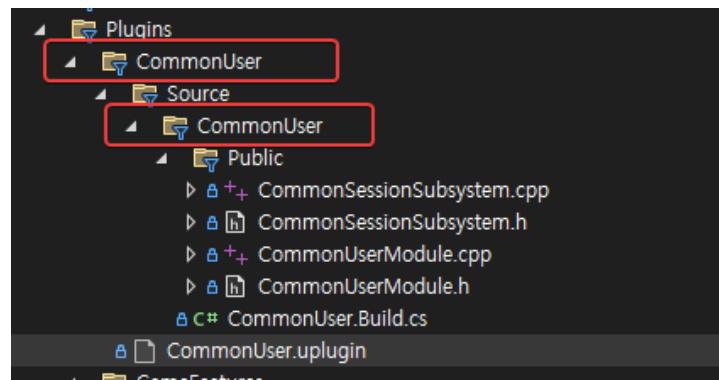


- Event ActorBeginOverlap을 BP에서 구현하기에 앞서, Map Transition 관련 기능을 제공하는 CommonUser Plugin을 구현해야 한다

CommonUser

▼ 펼치기

- GenerateProjectFiles.bat를 활용하여, CommonUser Plugin의 윤곽을 잡아주자:



- 위의 CommonUser 폴더를 포함하여 전체적인 계층적 구조를 유의해서 잡아주자

- CommonUser.uplugin

```
1 { "FileVersion": 3,
2   "Version": 1,
3   "VersionName": "1.0",
4   "FriendlyName": "CommonUser",
5   "Description": "Provides gameplay code and blueprint wrappers for online and platform operations",
6   "Category": "Gameplay",
7   "CanContainContent": false,
8   "IsBetaVersion": false,
9   "IsExperimentalVersion": false,
10  "Installed": false,
11  "Modules": [
12    {
13      "Name": "CommonUser",
14      "Type": "Runtime",
15      "LoadingPhase": "Default"
16    }
17  ],
18  "Plugins": []
19 }
20 }
```

- CommonUser.Build.cs

```
CommonUser.Build.cs
1  using UnrealBuildTool;
2
3  public class CommonUser : ModuleRules
4  {
5      public CommonUser(ReadOnlyTargetRules Target) : base(Target)
6      {
7          PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
8
9          PublicDependencyModuleNames.AddRange(
10             new string[]
11             {
12                 "Core",
13                 "CoreUObject",
14                 "Engine",
15                 // ... add other public dependencies that you statically link with here ...
16             });
17     }
18 }
19
```

□ Public/CommonUserModule[.h/.cpp]

```
CommonUserModule.h
1 #pragma once
2
3 #include "Modules/ModuleInterface.h"
4
5 class FCommonUserModule : public IModuleInterface
6 {
7 public:
8     virtual void StartupModule() override;
9     virtual void ShutdownModule() override;
10};

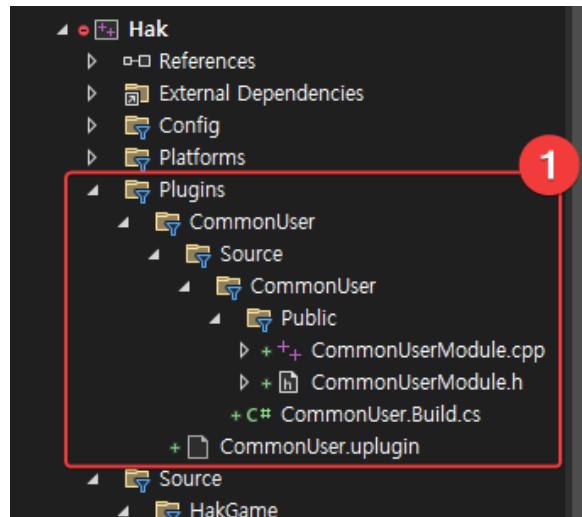
```

```
CommonUserModule.cpp
1 #include "CommonUserModule.h"
2 #include "Modules/ModuleManager.h"
3
4 void FCommonUserModule::StartupModule()
5 {
6 }
7
8 void FCommonUserModule::ShutdownModule()
9 {
10 }
11
12 IMPLEMENT_MODULE(FCommonUserModule, CommonUser);
```



Plugin을 생성할 경우, 위의 세가지 요소를 정의하지 않으면,
GenerateProjectFiles.bat에서 실패하니, 폴더 구조와 파일 내용을 꼼꼼히 살피자.

□ 아래와 같이 VS에 잘 추가되면 성공이다:



- 이제 CommonSessionSubsystem을 구현하자:

[간단한 설명]

CommonSessionSubsystem은 Experience에 속해있는 맵 로딩과 ExperienceManagerComponent에 Experience 변경에 대한 CmdArgs를 전달하는 역할을 담당한다.

- CommonSessionSubsystem.h/.cpp를 추가하자:

```

#pragma once

#include "CoreMinimal.h"
#include "Subsystems/GameInstanceSubsystem.h"
#include "CommonSessionSubsystem.generated.h"

UCLASS()
class COMMONUSER_API UCommonSessionSubsystem : public UGameInstanceSubsystem
{
    GENERATED_BODY()
public:
    UCommonSessionSubsystem() {}

    /**
     * member variables
     */
    /** PendingTravelURL은 흔히 맵의 경로로 생각하면 된다 */
    FString PendingTravelURL;
};

```

```

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

```

- HostSession을 구현하자:

- 간단히 Backbone을 구현하자:

```

#pragma once
#include "CoreMinimal.h"
#include "Subsystems/GameInstanceSubsystem.h"
#include "CommonSessionSubsystem.generated.h"

/** forward declaration */
class APlayerController;

/**
 * UCommonSessionSubsystem은 HakGame에서 사용되야 하기 때문에, Module Export를 해줘야하고 그래서 COMMONUSER_API를 추가해줘야 한다!
 * - 여러분들이 {ModuleName}_API의 추가는 ***다른 모듈에서 사용할 경우, 추가해주면 된다***
 */
UCLASS()
class COMMONUSER_API UCommonSessionSubsystem : public UGameInstanceSubsystem
{
    GENERATED_BODY()
public:
    UCommonSessionSubsystem() {}

    UFUNCTION(BlueprintCallable, Category=Session)
    void HostSession(APlayerController* HostingPlayer, UCommonSession_HostSessionRequest* Request);

    /**
     * member variables
     */
    /** PendingTravelURL은 흔히 맵의 경로로 생각하면 된다 */
    FString PendingTravelURL;
};

```

```

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

void UCommonSessionSubsystem::HostSession(APlayerController* HostingPlayer, UCommonSession_HostSessionRequest* Request)
{
}

```

- UCommonSession_HostSessionRequest를 구현하자:

```

void UCommonSessionSubsystem::HostSession(APlayerController* HostingPlayer, UCommonSession_HostSessionRequest* Request)
{
    ULocalPlayer* LocalPlayer = (HostingPlayer != nullptr) ? HostingPlayer->GetLocalPlayer() : nullptr;
    if (!LocalPlayer)
    {
        return;
    }

    // HostSessionRequest에서 MapID와 ExtraArgs를 통해 URL을 생성하여, MapLoad를 시작한다
    GetWorld()->ServerTravel(Request->ConstructTravelURL());
}

```

- ConstructTravelURL() 구현:

```

FString UCommonSession_HostSessionRequest::ConstructTravelURL() const
{
    FString CombinedExtraArgs;

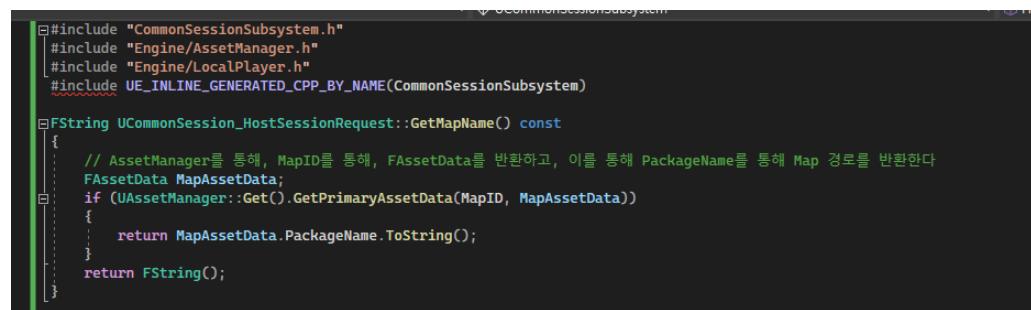
    for (const auto& ExtraArg : ExtraArgs)
    {
        if (ExtraArg.Key.IsEmpty())
        {
            continue;
        }

        /**
         * ?를 separate로 복수개의 ExtraArg를 추가함:
         * - Key 값 유무에 따라, =(assignment)를 통해 알맞는 CmdArg를 생성
         */
        if (ExtraArg.Value.IsEmpty())
        {
            CombinedExtraArgs += FString::Printf(TEXT("%s"), *ExtraArg.Key);
        }
        else
        {
            CombinedExtraArgs += FString::Printf(TEXT("%s=%s"), *ExtraArg.Key, *ExtraArg.Value);
        }
    }

    // Map 경로 앞에 추가하여, 최종 TravelURL 생성
    return FString::Printf(TEXT("%s%s"), *GetMapName(), *CombinedExtraArgs);
}

```

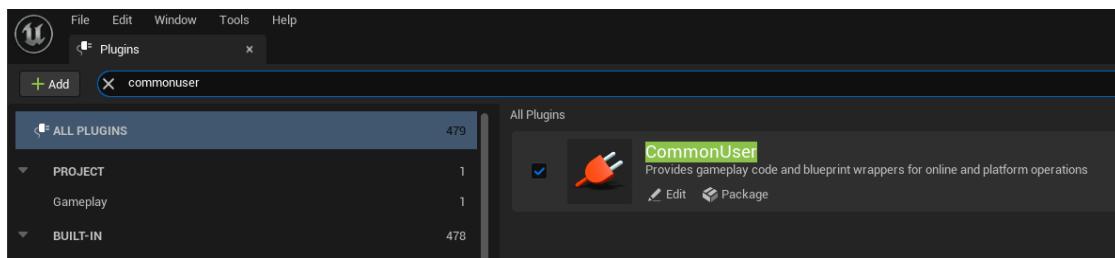
□ GetMapName() 구현:



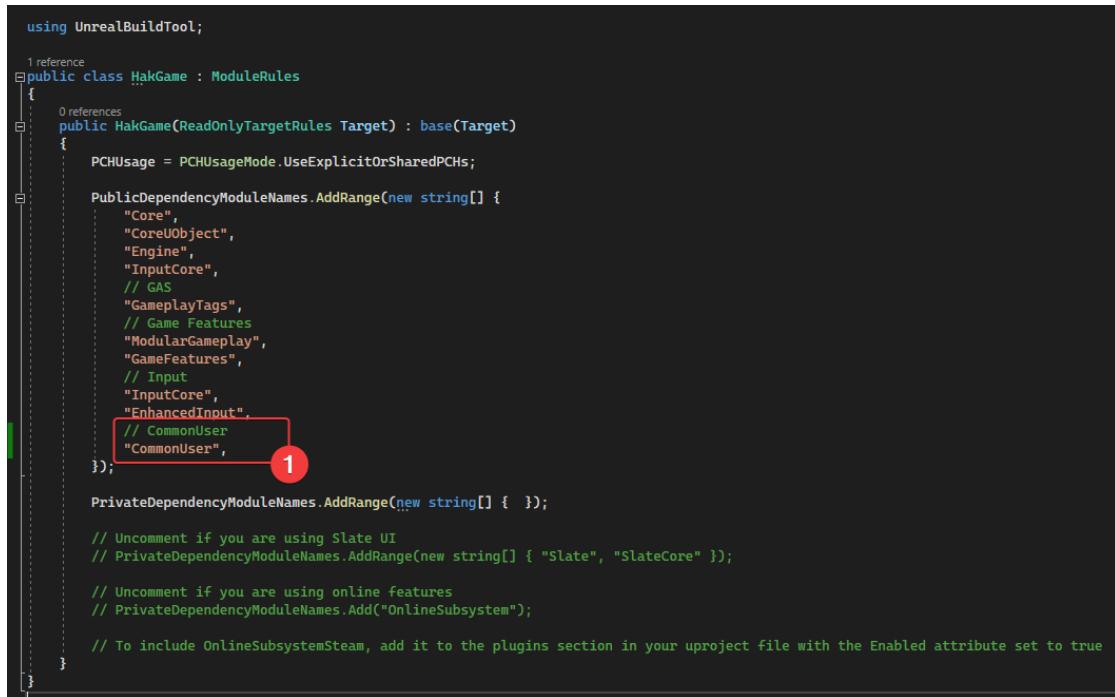
```
#include "CommonSessionSubsystem.h"
#include "Engine/AssetManager.h"
#include "Engine/LocalPlayer.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(CommonSessionSubsystem)

 FString UCommonSession::GetMapName() const
{
    // AssetManager를 통해 MapID를 통해 FAssetData를 반환하고, 이를 통해 PackageName을 통해 Map 경로를 반환한다
    FAssetData MapAssetData;
    if (UAssetManager::Get().GetPrimaryAssetData(MapID, MapAssetData))
    {
        return MapAssetData.PackageName.ToString();
    }
    return FString();
}
```

□ CommonUser의 Plugin을 HakGame에 활성화해주자:



□ HakGame.Build.cs에 CommonUser를 추가하자:



```
using UnrealBuildTool;

public class HakGame : ModuleRules
{
    public HakGame(ReadOnlyTargetRules Target) : base(Target)
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

        PublicDependencyModuleNames.AddRange(new string[] {
            "Core",
            "CoreObject",
            "Engine",
            "InputCore",
            // GAS
            "GameplayTags",
            // Game Features
            "ModularGameplay",
            "GameFeatures",
            // Input
            "InputCore",
            "EnhancedInput",
            // CommonUser
            "CommonUser",
        });
    }

    PrivateDependencyModuleNames.AddRange(new string[] { });

    // Uncomment if you are using Slate UI
    // PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });

    // Uncomment if you are using online features
    // PrivateDependencyModuleNames.Add("OnlineSubsystem");

    // To include OnlineSubsystemSteam, add it to the plugins section in your uproject file with the Enabled attribute set to true
}
```

□ HakUserFacingExperienceDefinition에 CreateHostingRequest()를 구현하자:

```

#pragma once

#include "CoreMinimal.h"
#include "Engine/DataAsset.h"
#include "HakUserFacingExperienceDefinition.generated.h"

/** Forward declarations */
class UCommonSession_HostSessionRequest; 1

/**
 * UHakUserFacingExperienceDefinition
 * - description of settings used to display experiences in the UI and start a new session
 */
UCLASS(BlueprintType)
class HAKGAME_API UHakUserFacingExperienceDefinition : public UPrimaryDataAsset
{
    GENERATED_BODY()
public:
    /**
     * Map 로딩 및 Experience 전환을 위해, MapID와 ExperienceID를 활용하여, HostSessionRequest 생성
     */
    UCommonSession_HostSessionRequest* CreateHostingRequest() const; 2

    /**
     * member variables
     */

    /** the specific map to load */
    UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience, meta=(AllowedTypes="Map"))
    FPrimaryAssetId MapID;

    /** the gameplay experience to load */
    UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience, meta=(AllowedTypes="HakExperienceDefinition"))
    FPrimaryAssetId ExperienceID;
};

```

```

#include "HakGame/GameModes/HakUserFacingExperienceDefinition.h"
#include "CommonSessionSubsystem.h"

// 항상 generated.h 파일이 헤더에 포함되어 있다면, gen.cpp도 포함시키도록 한다!
#include UE_INLINE_GENERATED_CPP_BY_NAME(HakUserFacingExperienceDefinition)

UCommonSession_HostSessionRequest* UHakUserFacingExperienceDefinition::CreateHostingRequest() const
{
    const FString ExperienceName = ExperienceID.PrimaryAssetName.ToString();

    // 잠깐 한걸음 길이 생각해보기:
    // - UCommonSession_HostSessionRequest는 UObject로 생성해놓고, 알아서 GC가 된다:
    //   - 해당 객체는 현재 프레임에서 사용하기 때문에, GC에 대한 염려가 필요없다: 만약 다음 프레임이든 추가적인 프레임 상에서 해당 객체를 사용할 경우, Lifetime 관리 필요!
    //   - 그렇지 않으면 dangling 난다!
    UCommonSession_HostSessionRequest* Result = NewObject<UCommonSession_HostSessionRequest>();
    Result->MapID = MapID;
    Result->ExtraArgs.Add(TEXT("Experience"), ExperienceName);

    return Result;
}

```

자, 이제 BP를 완성할 준비가 되었다! Experience를 통한 맵간 이동을 진행해보자.

B_TeleportToUserFacingExperience

▼ 펼치기

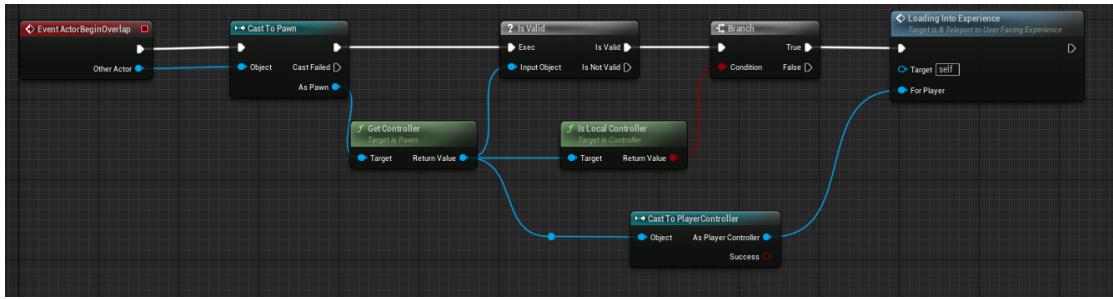
- CreateHostingRequest() 함수를 BlueprintCallable로 정의:

```

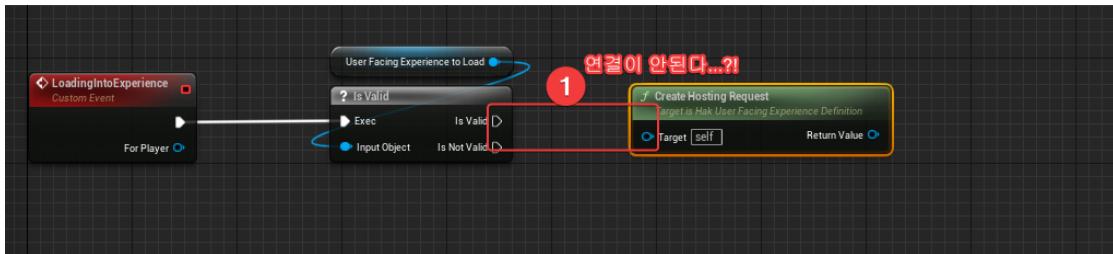
/**
 * UHakUserFacingExperienceDefinition
 * - description of settings used to display experiences in the UI and start a new session
 */
UCLASS(BlueprintType)
class HAKGAME_API UHakUserFacingExperienceDefinition : public UPrimaryDataAsset
{
    GENERATED_BODY()
public:
    /**
     * Map 로딩 및 Experience 전환을 위해, MapID와 ExperienceID를 활용하여, HostSessionRequest 생성
     */
    UFUNCTION(BlueprintCallable)
    UCommonSession_HostSessionRequest* CreateHostingRequest() const; 1

```

□ Event Actor Begin Overlap 구현:



□ LoadingIntoExperience 구현:



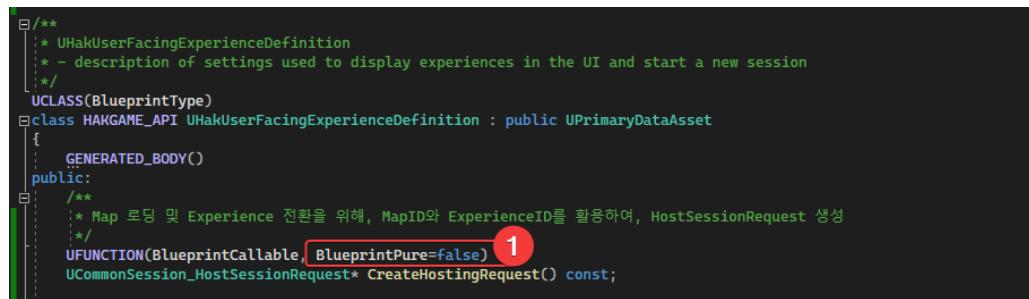
□ 앞서, CreateHostingRequest()에 추가적인 BP 옵션을 넣어줘야 한다:

BlueprintPure

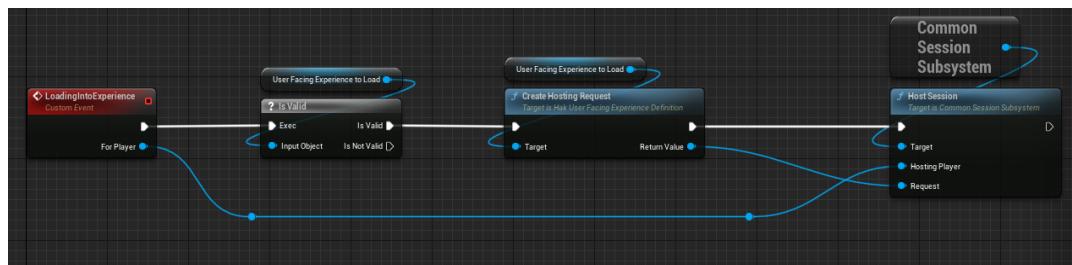
The function does not affect the owning object in any way and can be executed in a Blueprint or Level Blueprint graph.

- 간단히, 이야기하면, Owning Object 상관없이 BP에서 함수 호출이 아닌 하나의 Node 단위 실행이 가능한지 여부라고 생각하면 된다.

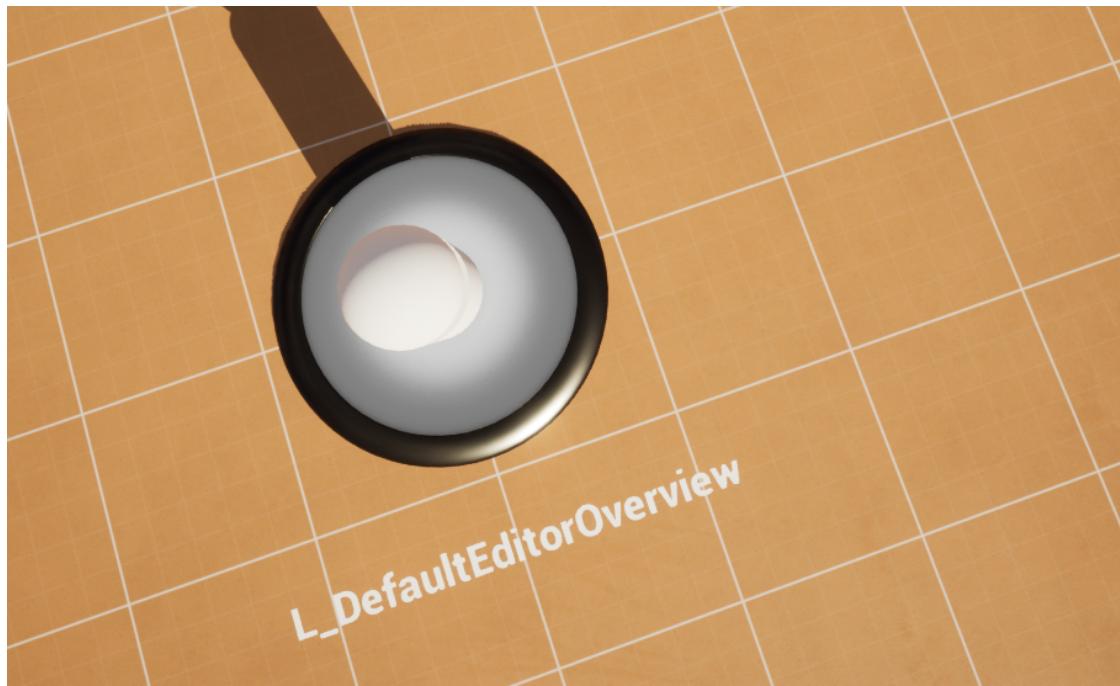
□ 아래와 같이 추가해주자:



□ 이어서, 구현하면 아래와 같다:

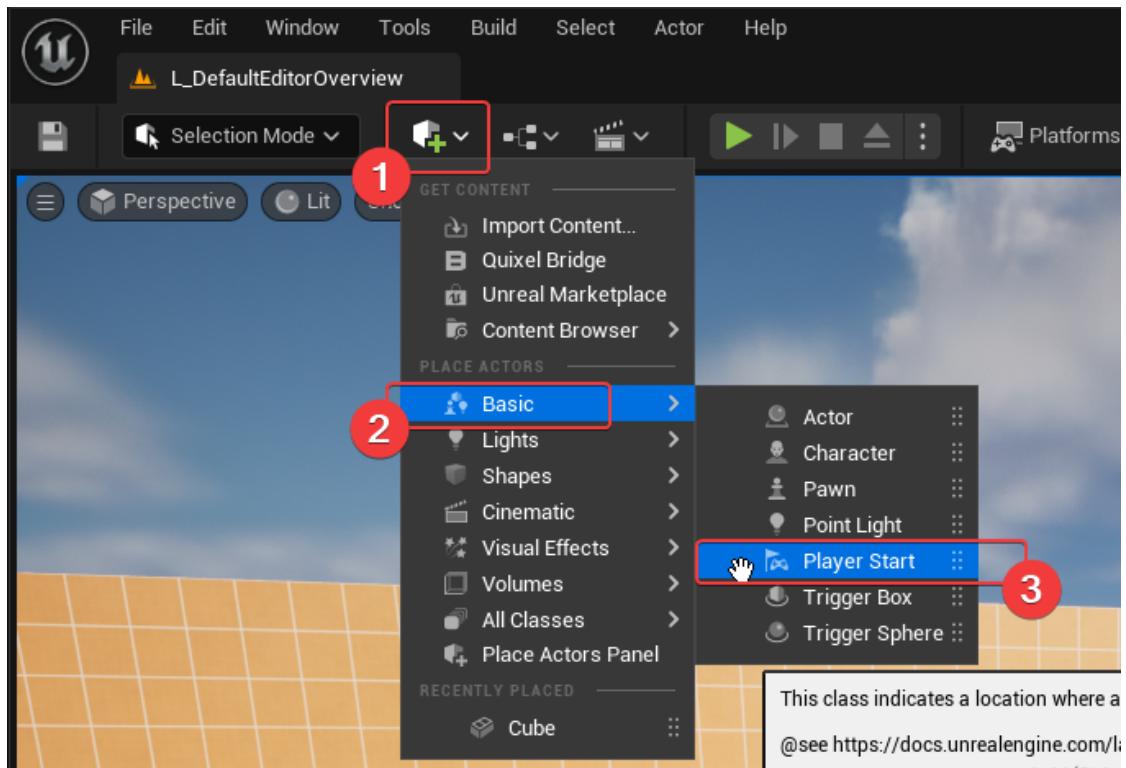


- 실행해보면, 뭔가 우리의 의도대로 원래 처음 Spawn Location에서 시작되는거 같지 않다...



- 이는 우리가 PlayerStart를 생성하지 않았기 때문이다

□ PlayerStart를 추가하자:



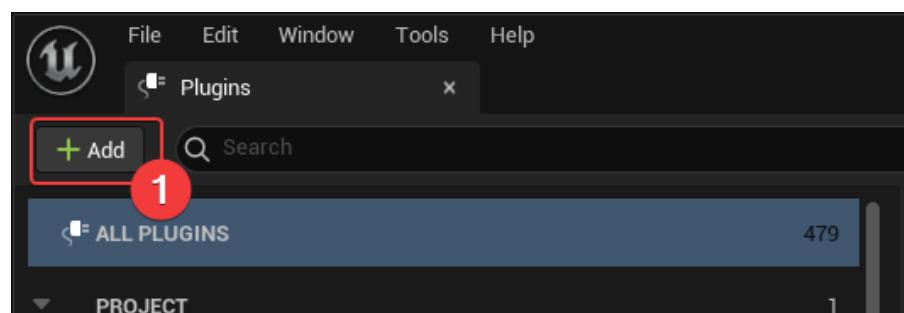
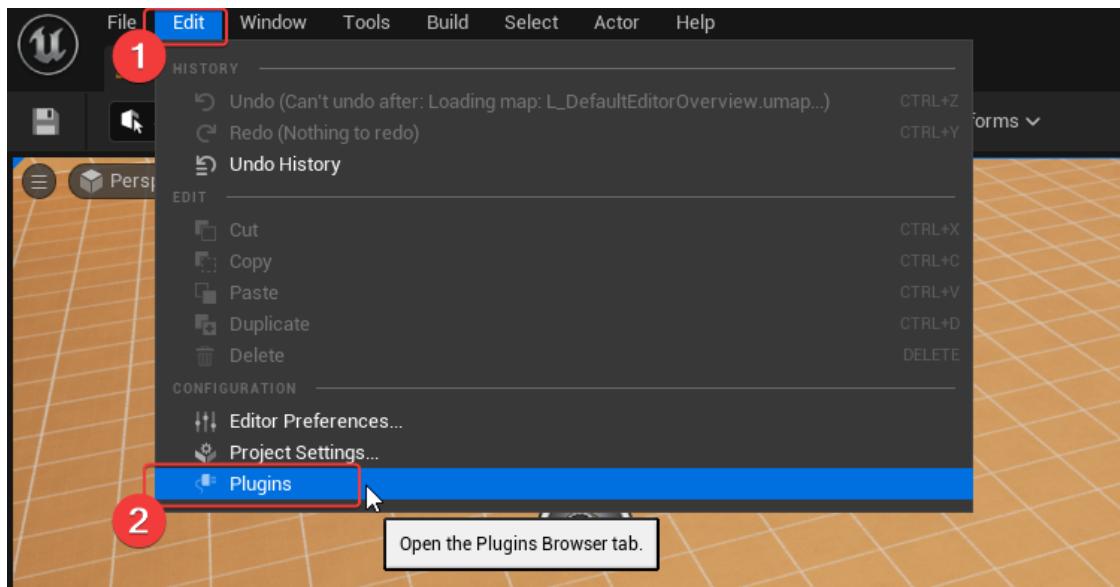
- 아래와 같이 Transform 속성을 입력하자:

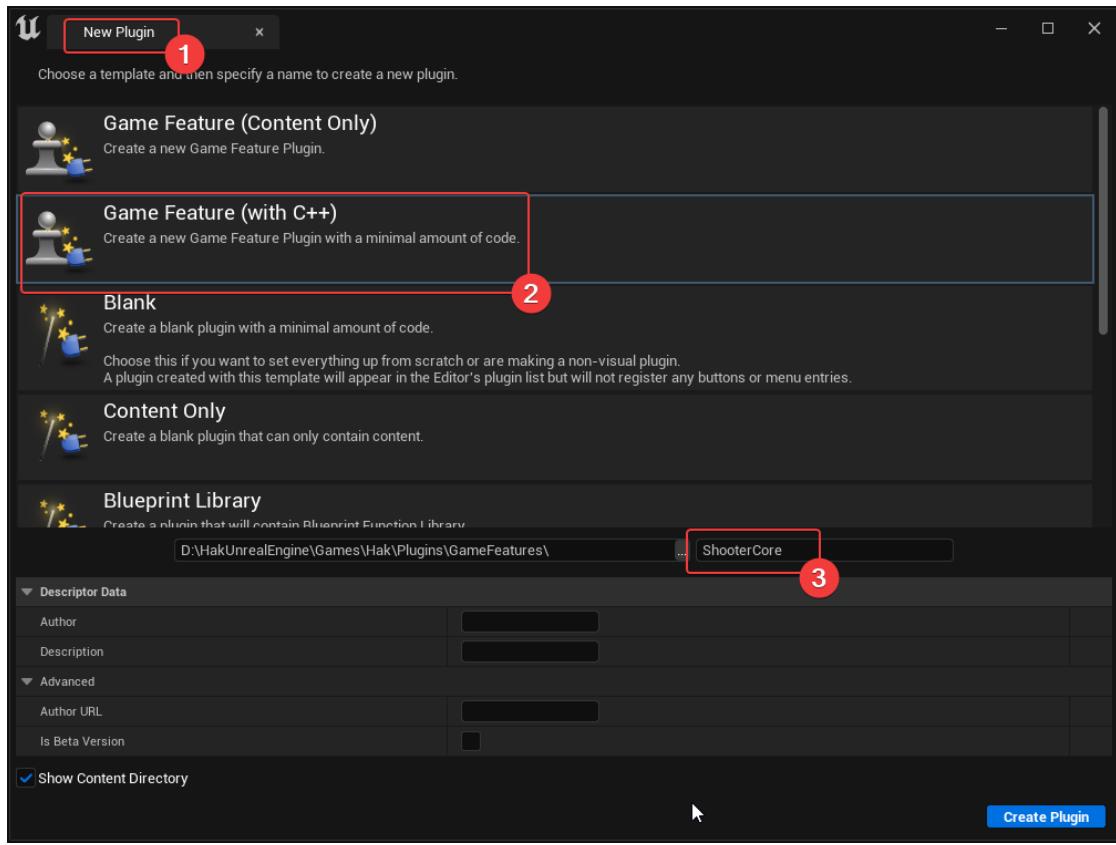


ShooterCore

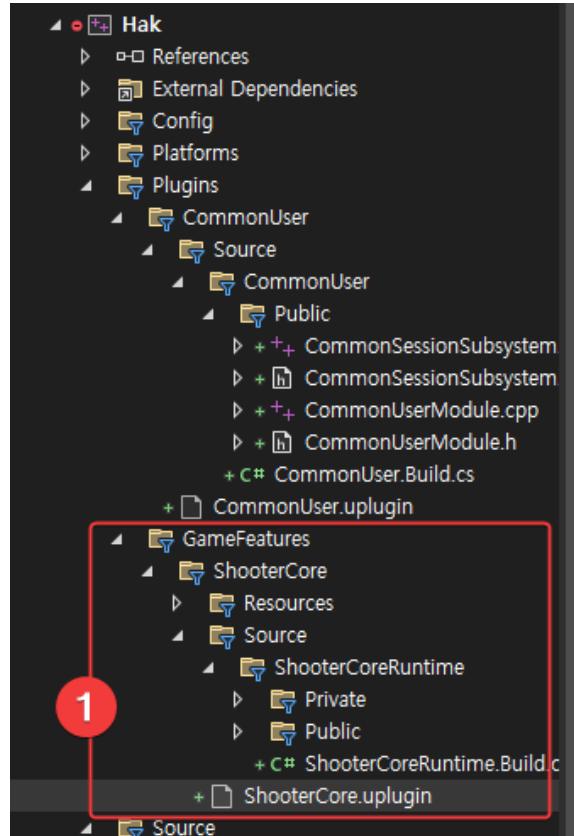
▼ 펼치기

- 아래의 목표로 진행해보자:
 - 우선, 기존 L_DefatuleEditorOverview와 B_HakDefaultExperience와 형태를 같게 Map을 구성하고, 비슷하게 Experience로 만들어보자
- 아래와 같이 GameFeature를 ShooterCore로 추가해주자:

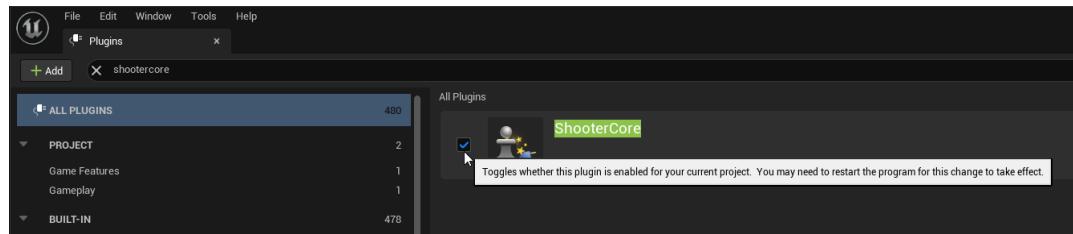




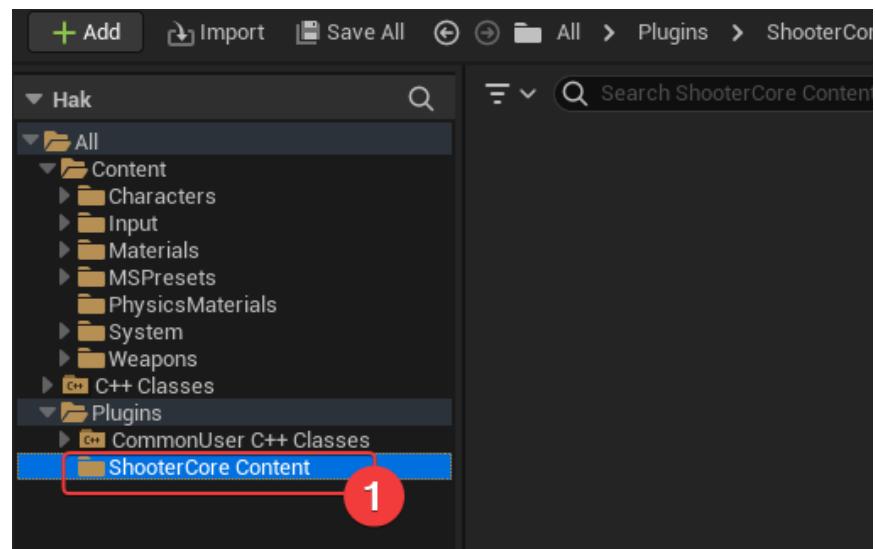
- 아래와 같이 ShooterCore가 추가되었음을 알 수 있다:



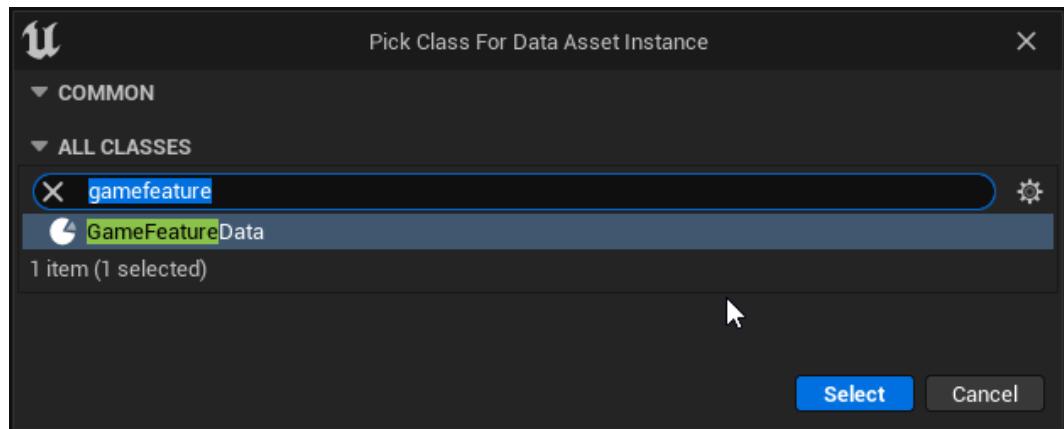
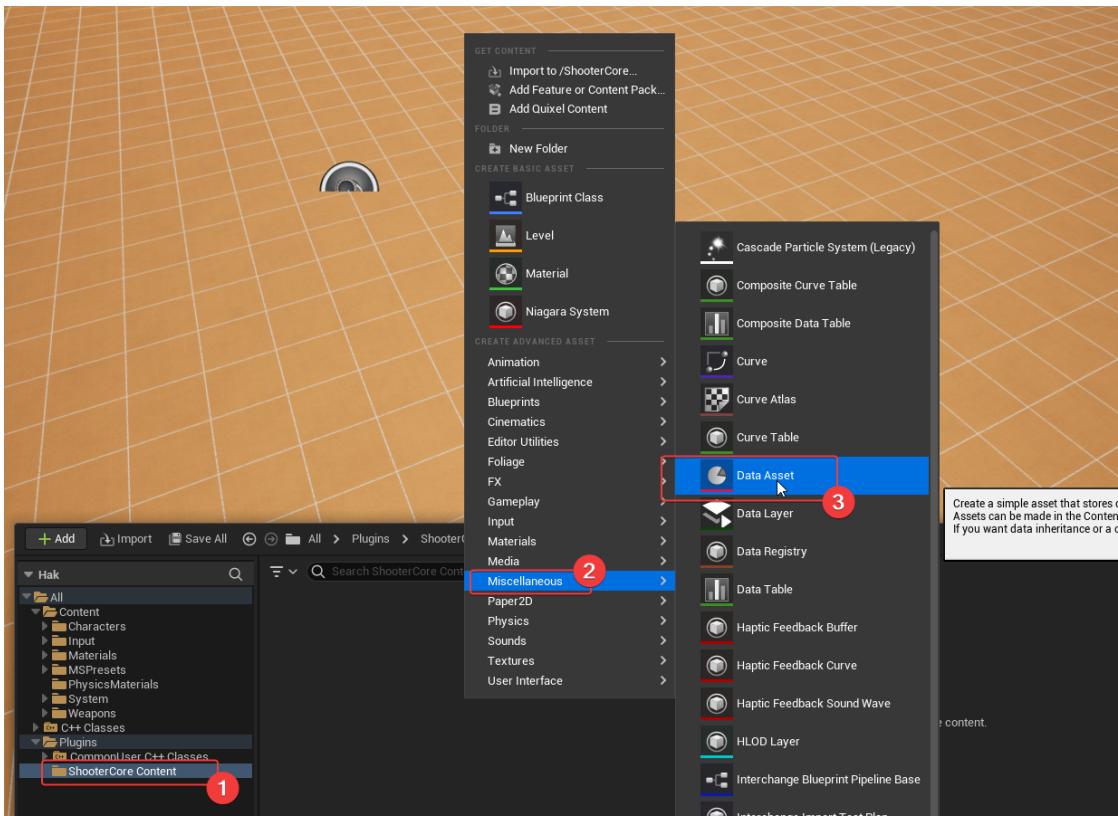
우선 해당 GameFeature를 활성화해주자:

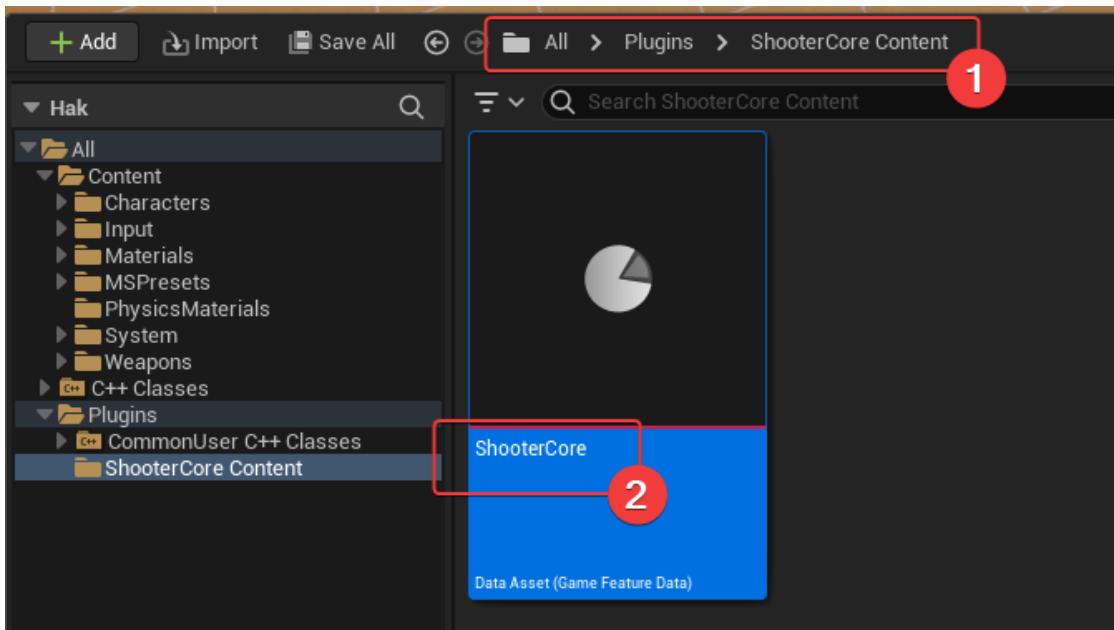


- 아래와 같이 제대로 활성화되었다면, 보인다:

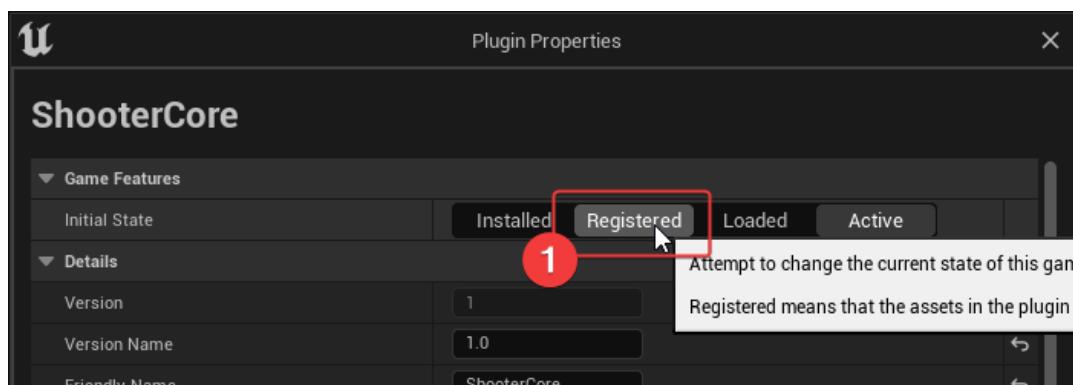
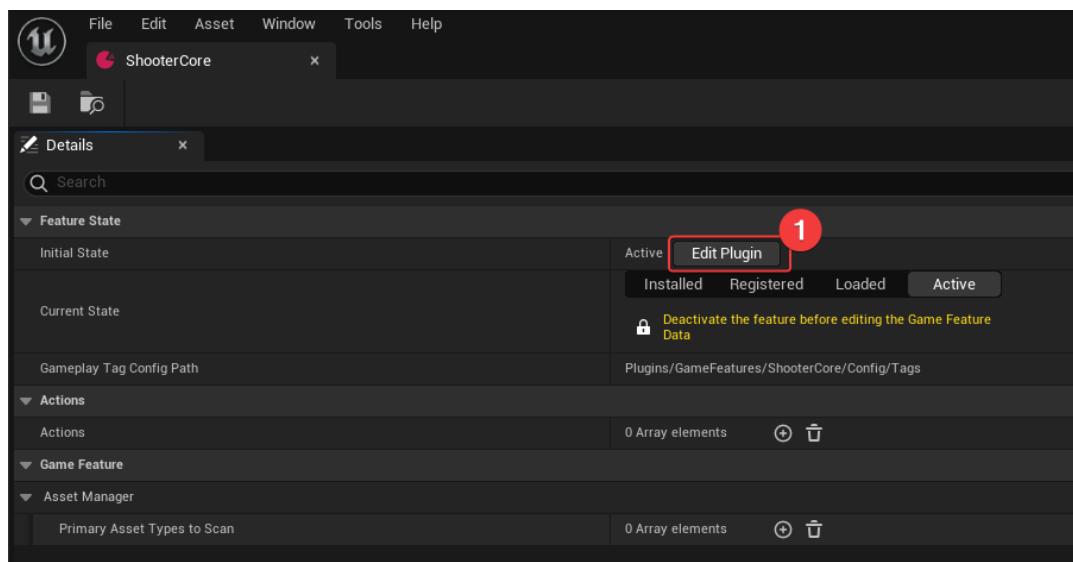


GameFeatureData를 추가해주자:





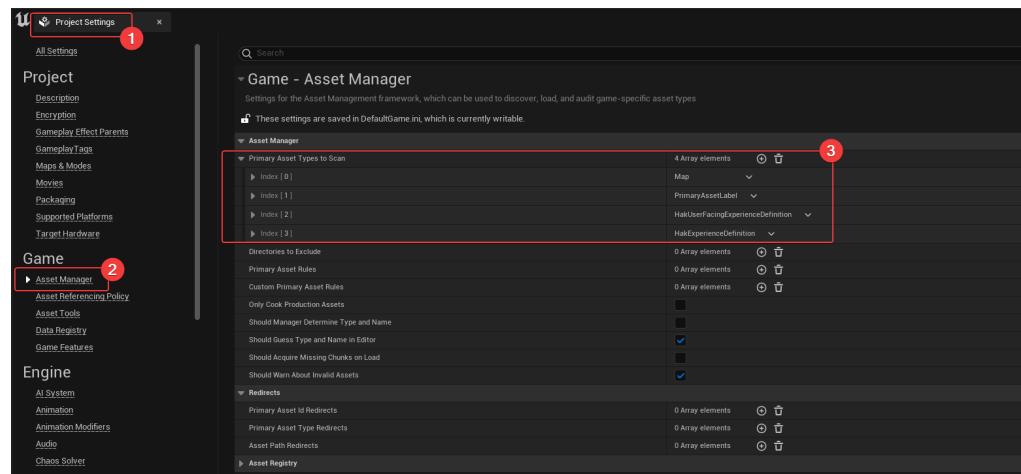
아래와 같이 ShooterCore의 InitState를 바꾸어주자:



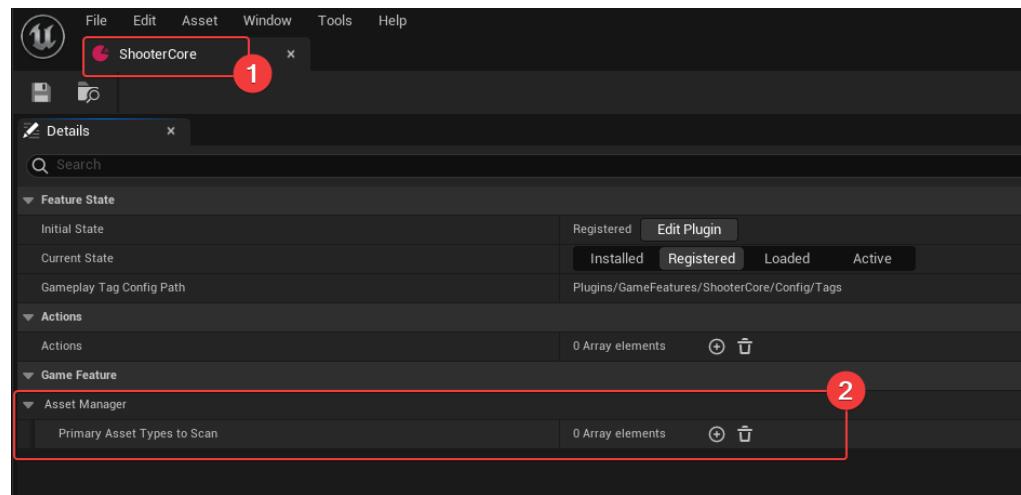
- 우리는 ShooterCore를 항상 활성화가 아닌 필요할 때, 활성화를 진행할 예정이다!

□ GameFeatureData를 간단히 이해해보자:

- GameFeature Plugin의 설정을 담당하는 DataAsset으로 생각하면 됨
- 대표적으로, 앞서, 우리가 Project Setting에서 진행했던 PrimaryDataAsset 설정이 Plugin별로 GameFeatureData에서 설정 가능:
 - 기존 Project Settings:



◦ GameFeatureData:

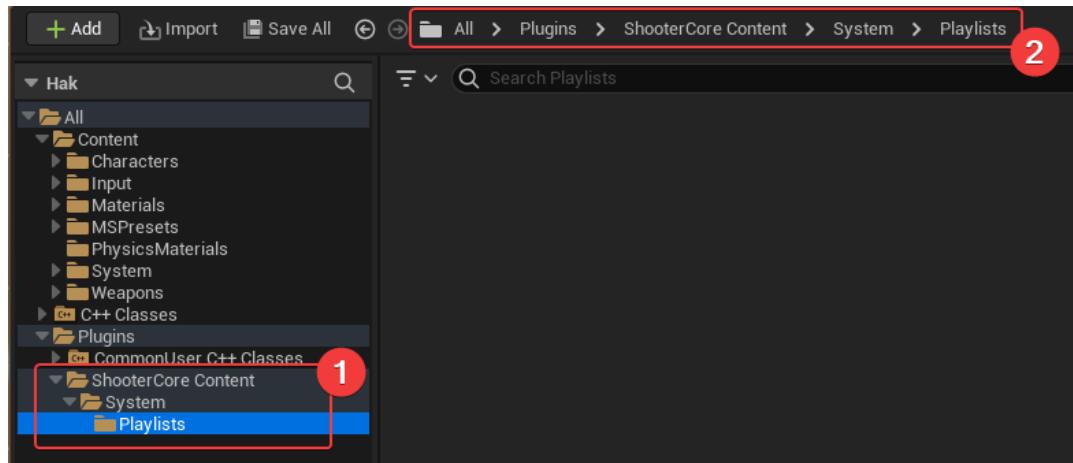


GameFeatureData를 통해 PrimaryAssetType을 추가하면, 자동적으로 Scan할 대상으로 추가된다.

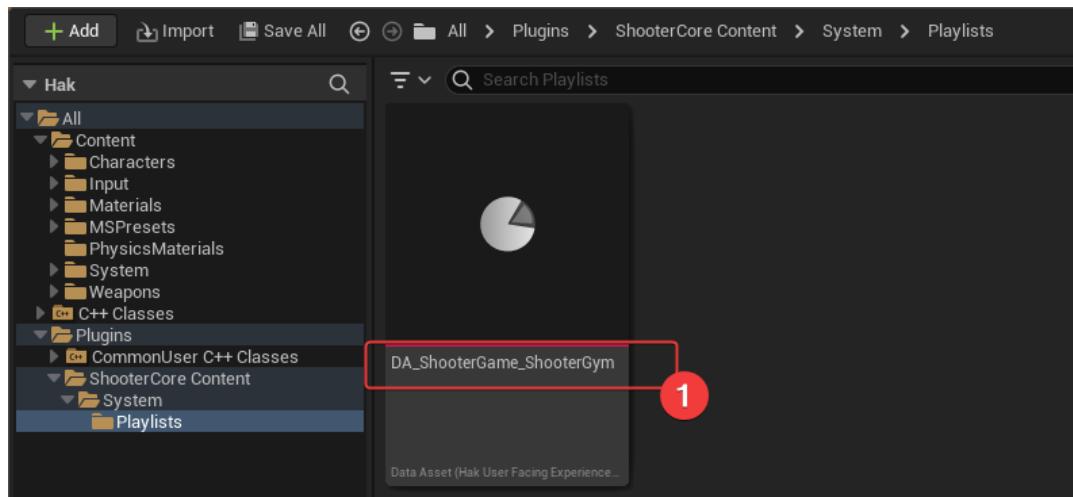
- 이제 과거의 기억을 회고해보며, ShooterCore에 해당하는 새로운 Experience를 추가해보자:

□ UserFacingExperience에 해당하는 DA_ShooterGame_ShooterGym을 추가하자:

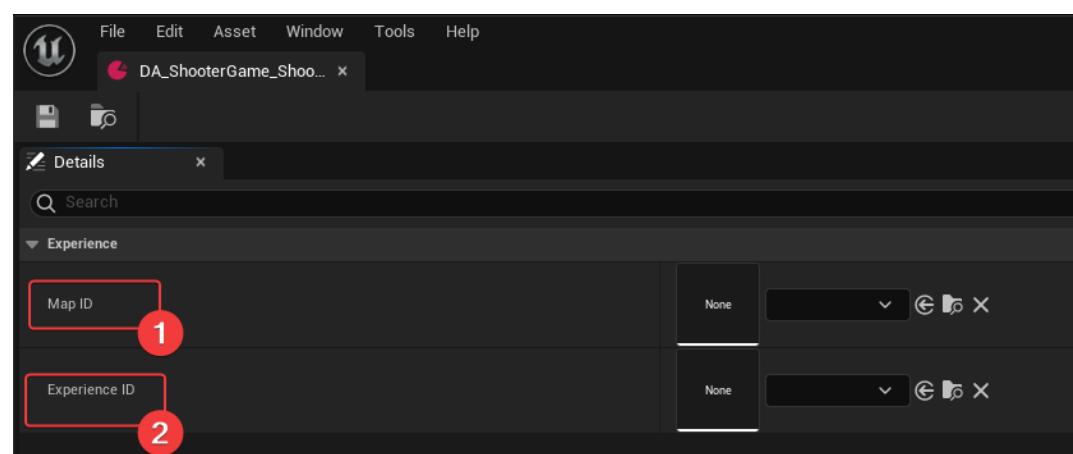
□ 아래와 같이 폴더를 추가하자:



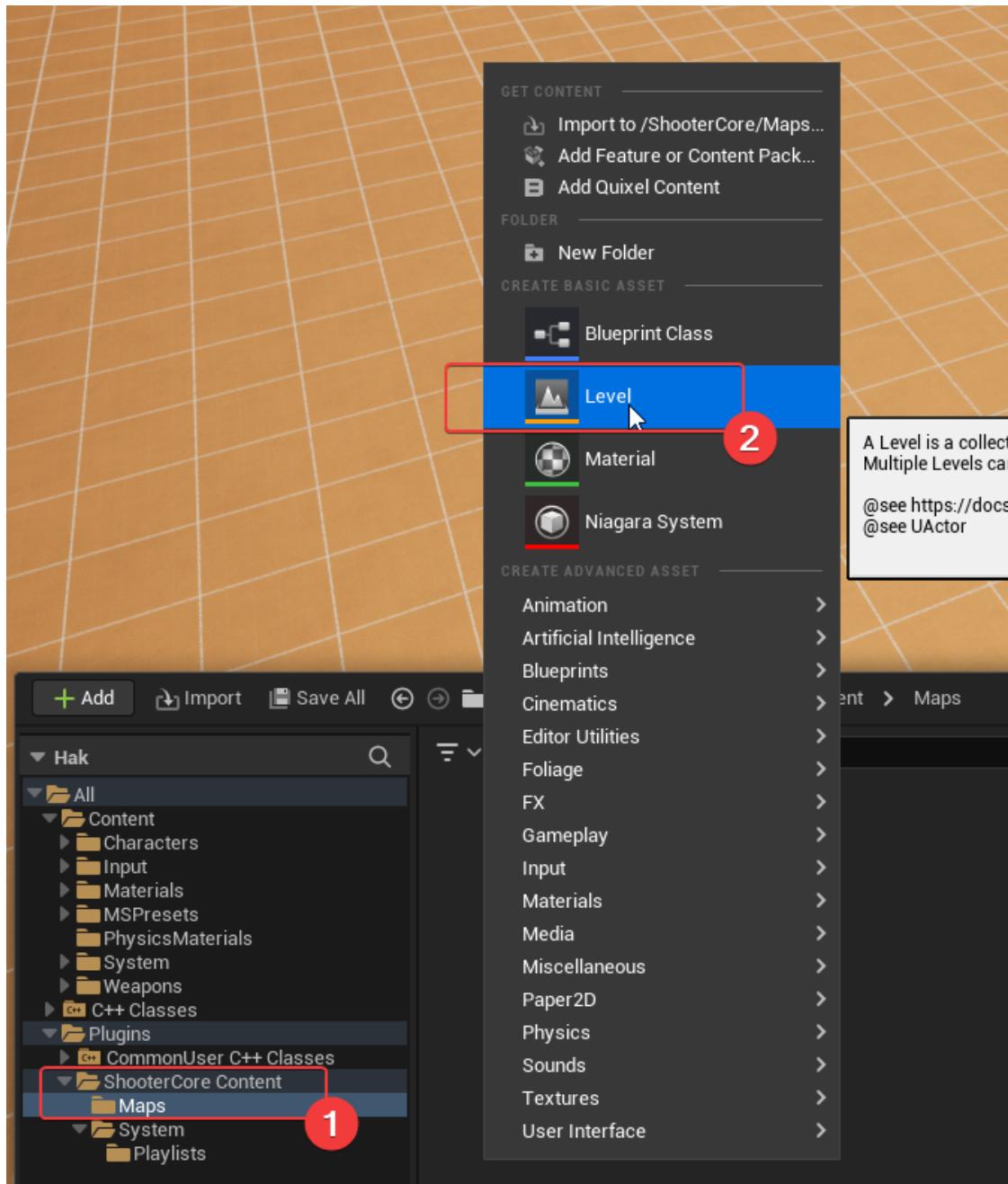
DA_ShooterGame_ShooterGym을 추가하자:

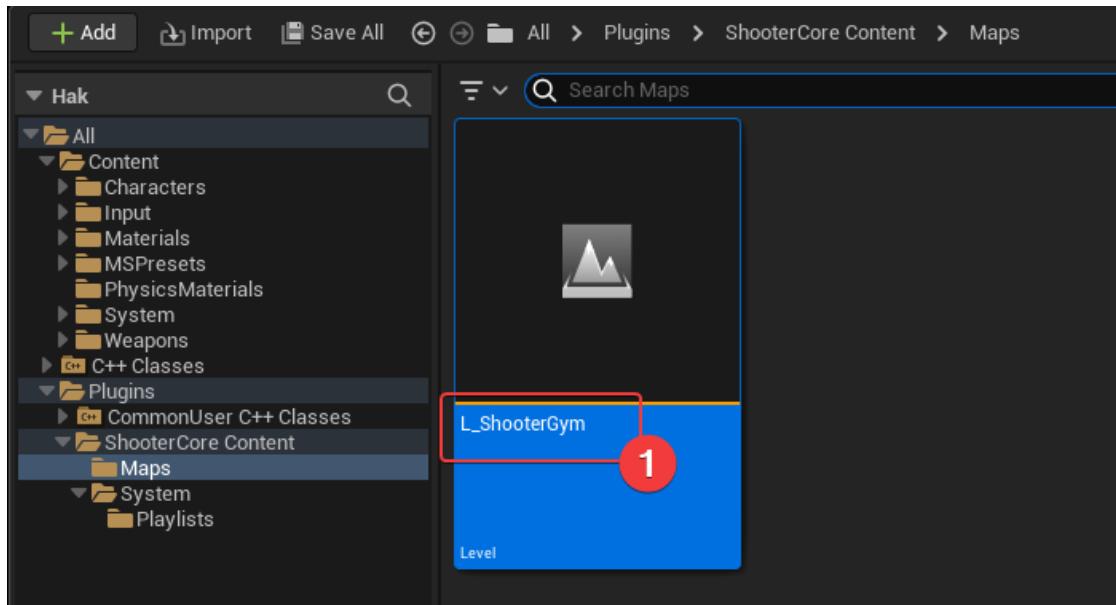


- 아래와 같이 Experience와 Map을 추가해야 한다:



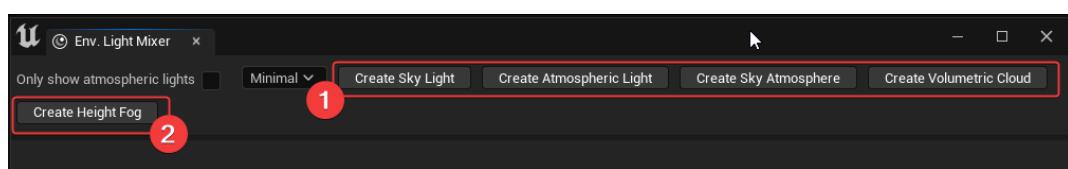
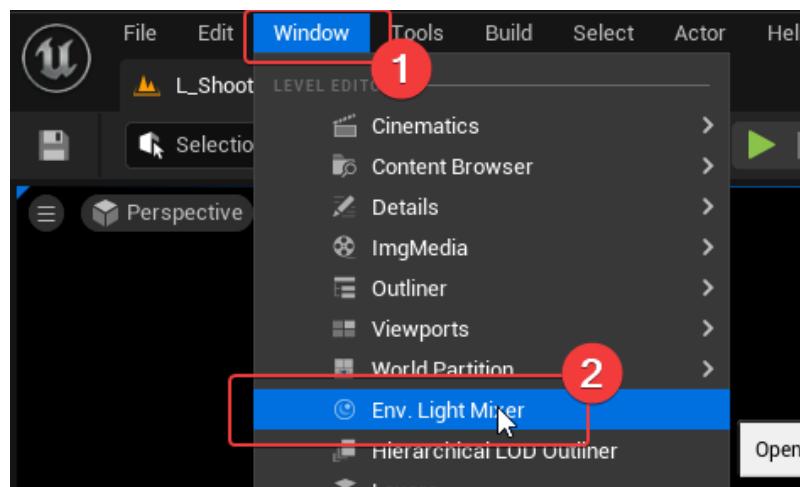
Map을 추가해주자:





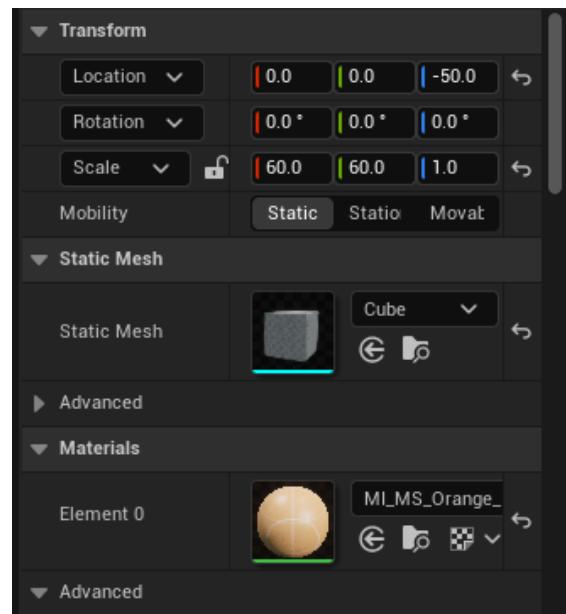
L_ShooterGym을 L_DefaultEditorOverview와 똑같이 구성해주자:

- Env.Light Mixer를 활용한 필수적인 환경세팅 진행하기:

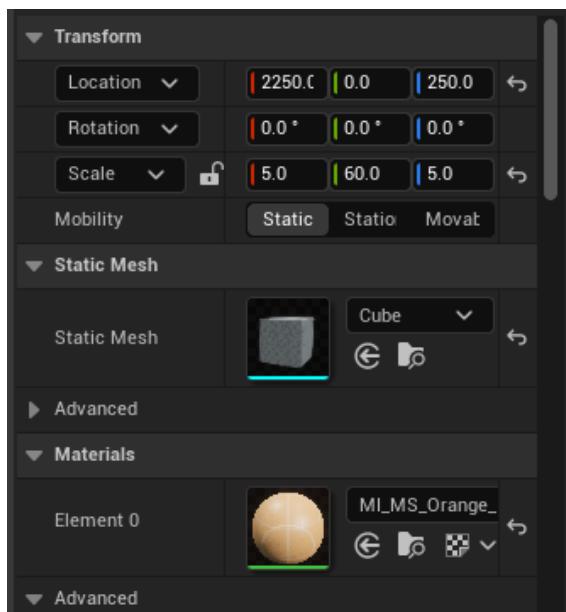


- Cube, Cube2을 추가하여 Platform Shape 만들어주기:

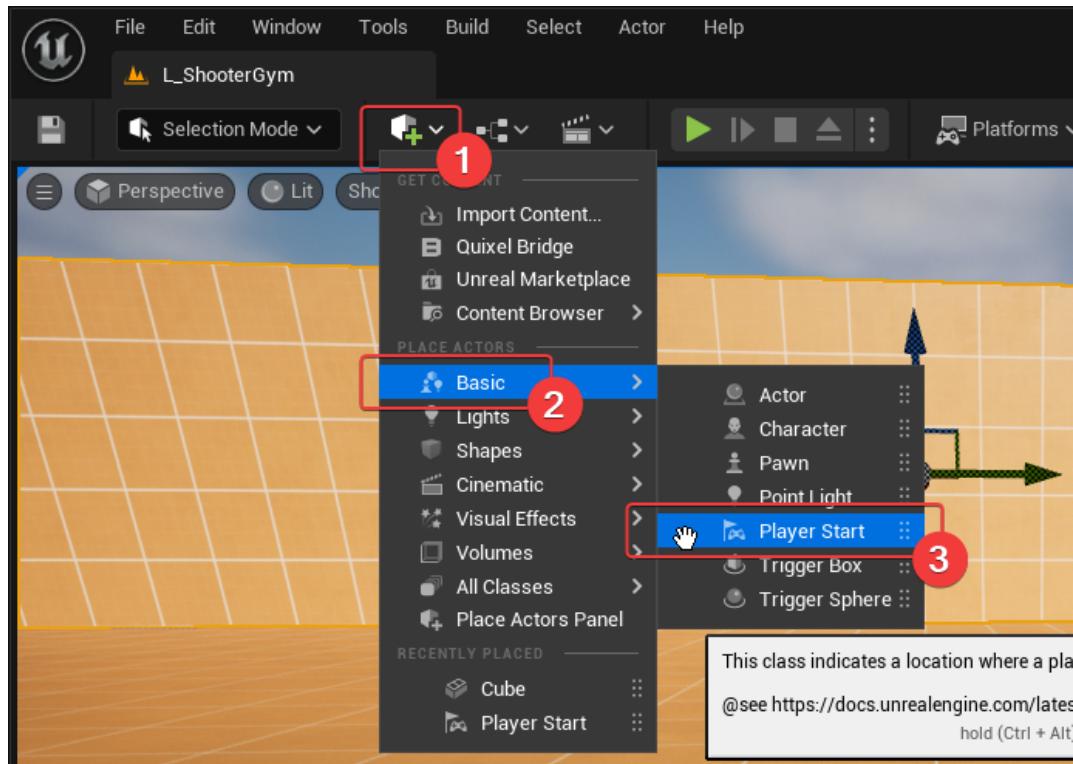
- Cube:



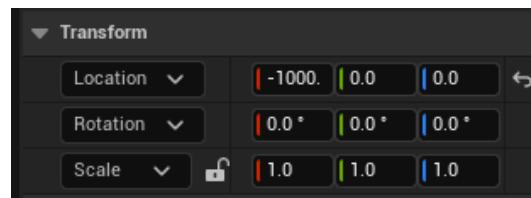
- Cube2:



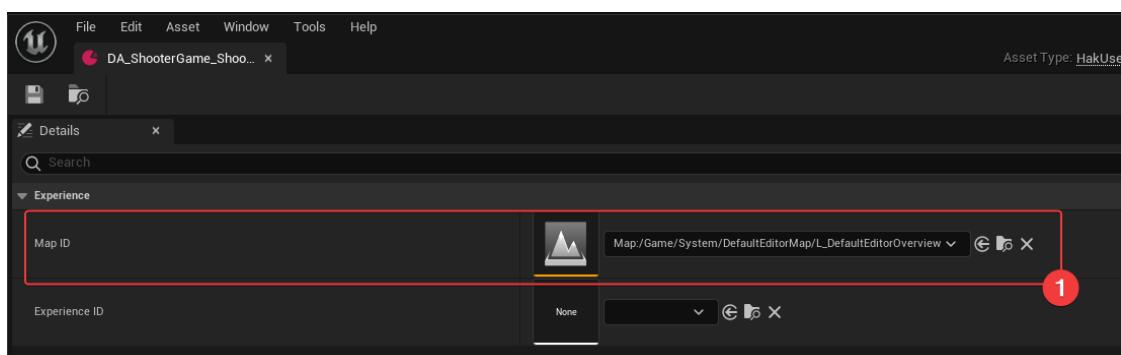
- PlayerStart 추가하기:



- 아래와 같이 속성값 설정:

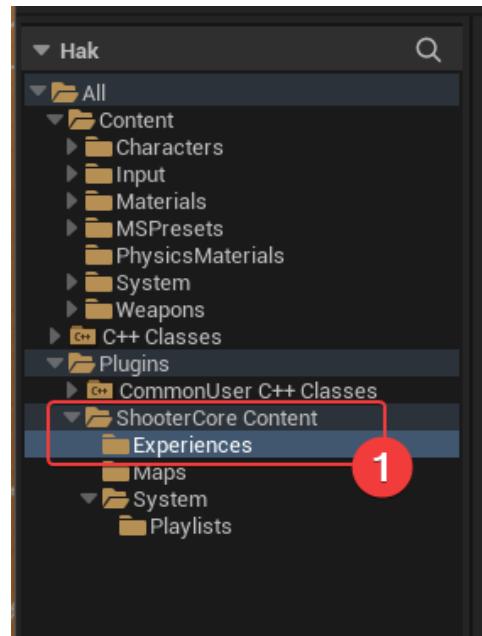


- L_ShooterGym을 DA_ShooterGame_ShooterGym에 추가하기:

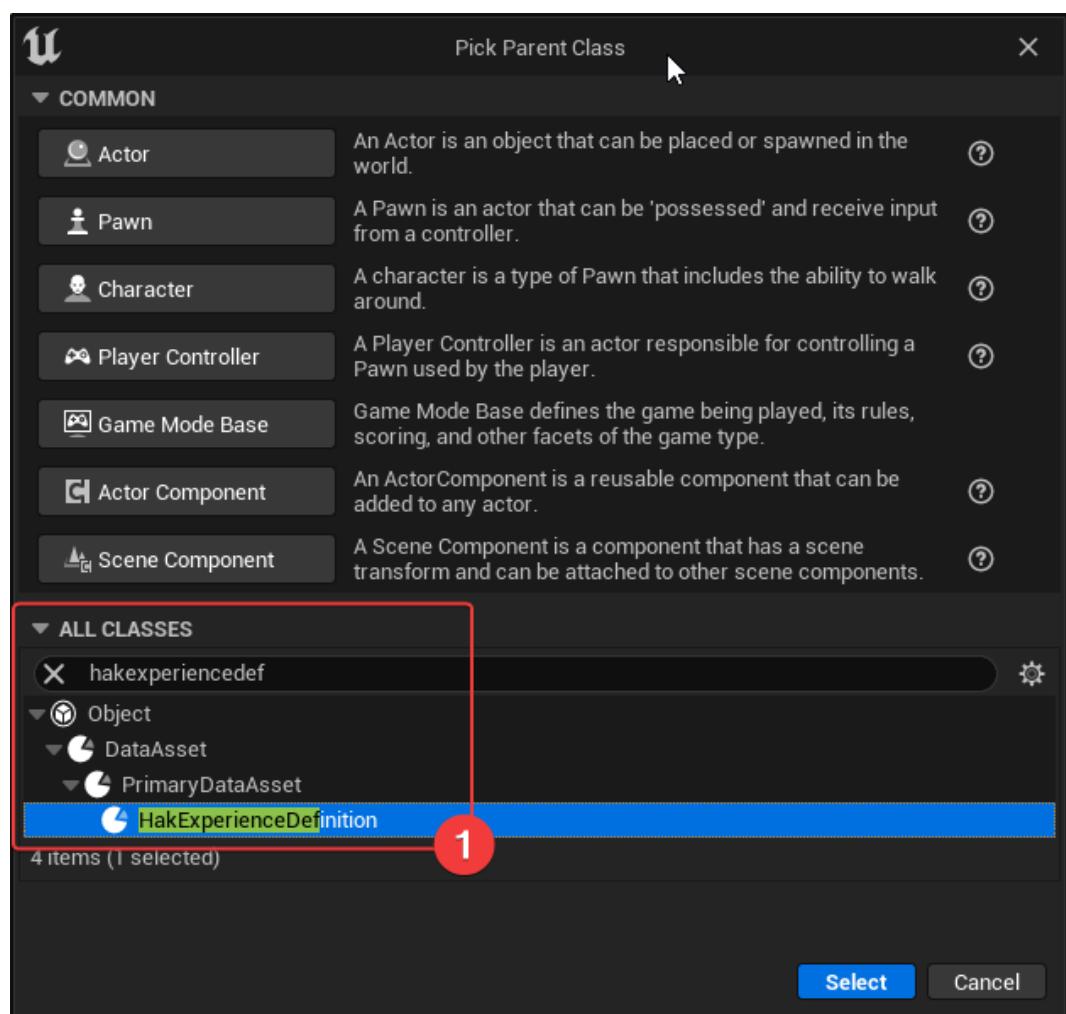


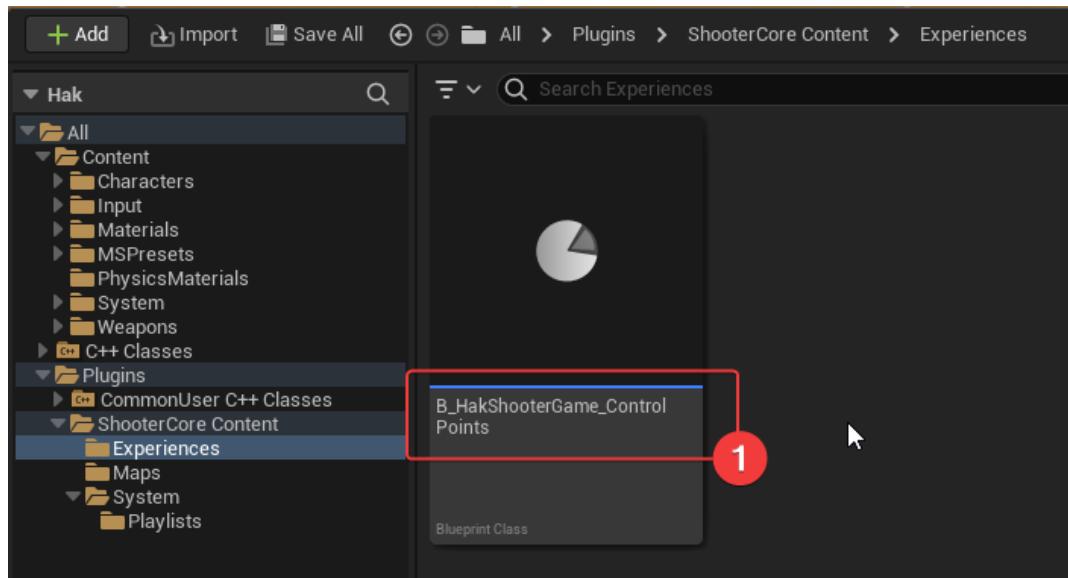
- HakExperienceDefinition의 B_HakShooterGame_ControlPoints 생성:

- Experiences 폴더 생성:

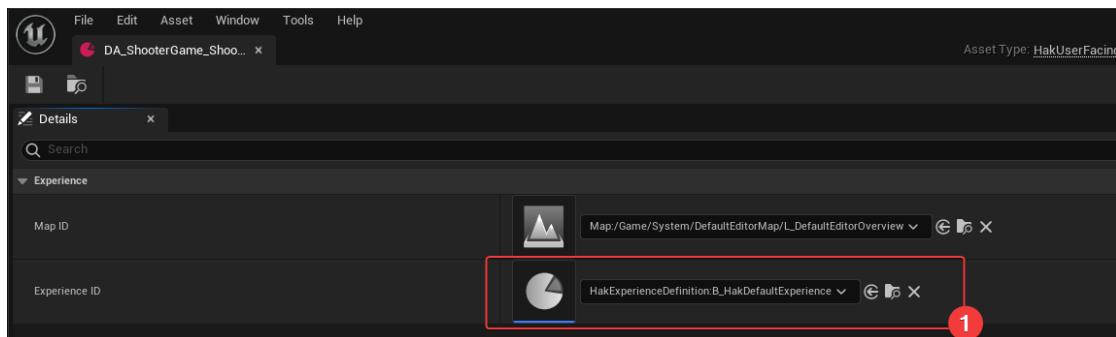


- B_HakShooterGame_ControlPoints 생성:

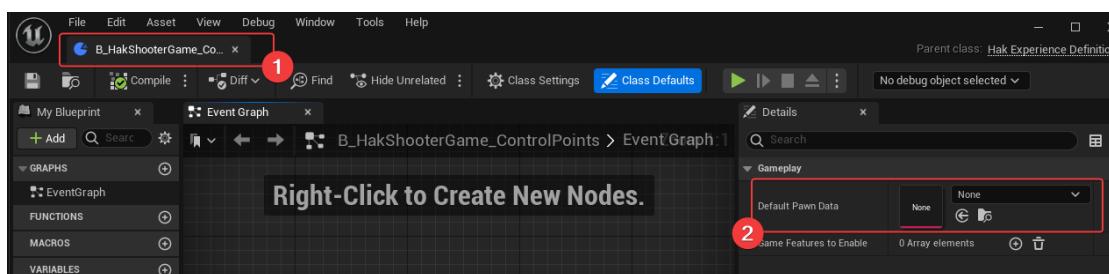




- DA_ShooterGame_ShooterGym에 B_HakShooterGame_ControlPoints를 추가해주자:

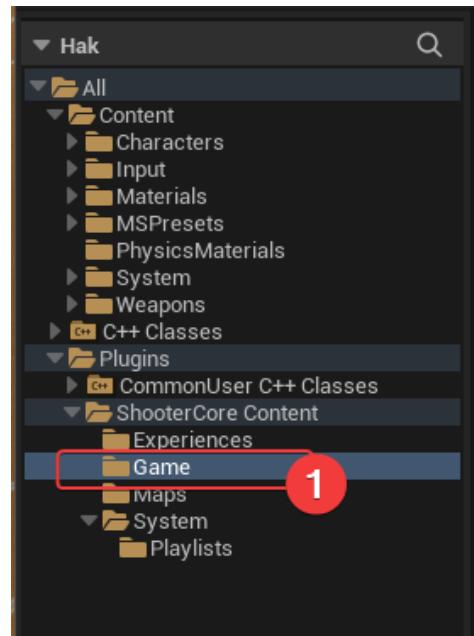


- B_HakShooterGame_ControlPoints에 우리는 DefaultPawnData를 설정해야 한다:



- HakPawnData인 HeroData_ShooterGame을 /Game 경로에 생성하자:

- /Game 폴더 생성:



- HeroData_ShooterGame 생성:

Top Panel: Pick Class For Data Asset Instance

Search Bar: hakpawn

Result List:

- HakPawnData

Bottom Panel: Cancel

Content Browser:

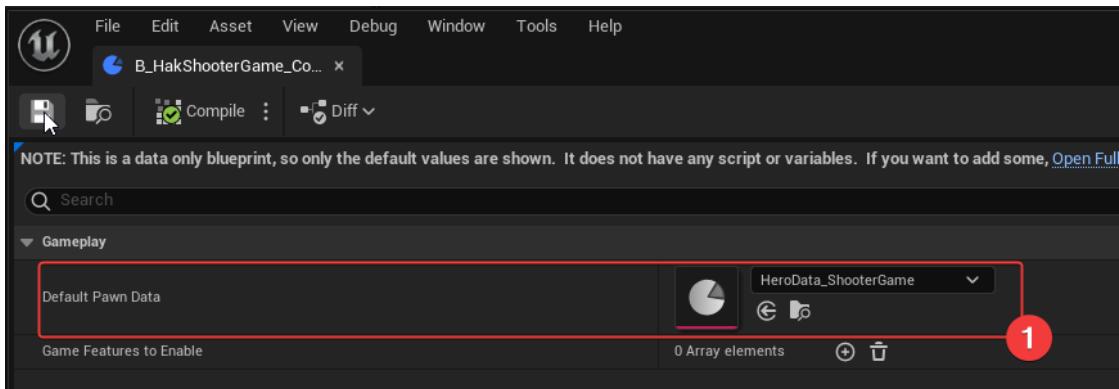
Path: All > Plugins > ShooterCore Content > Game

Selected Item: HeroData_ShooterGame (Data Asset (Hak Pawn Data))

Annotations:

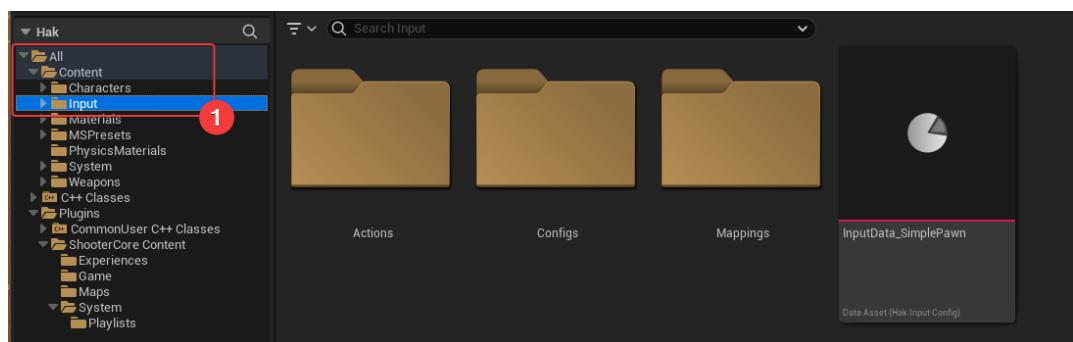
- Red circle with number 1 points to the 'Game' folder path in the Content Browser.
- Red circle with number 2 points to the 'HeroData_ShooterGame' asset in the Content Browser.

HeroData_ShooterGame을 B_HakShooterGame_ControlPoints에 설정하자:

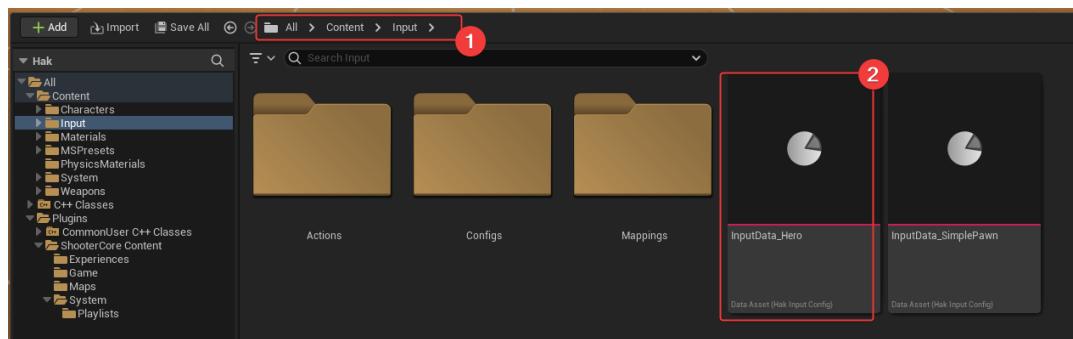


HeroData_ShooterGame에 설정할 InputConfig를 추가하자:

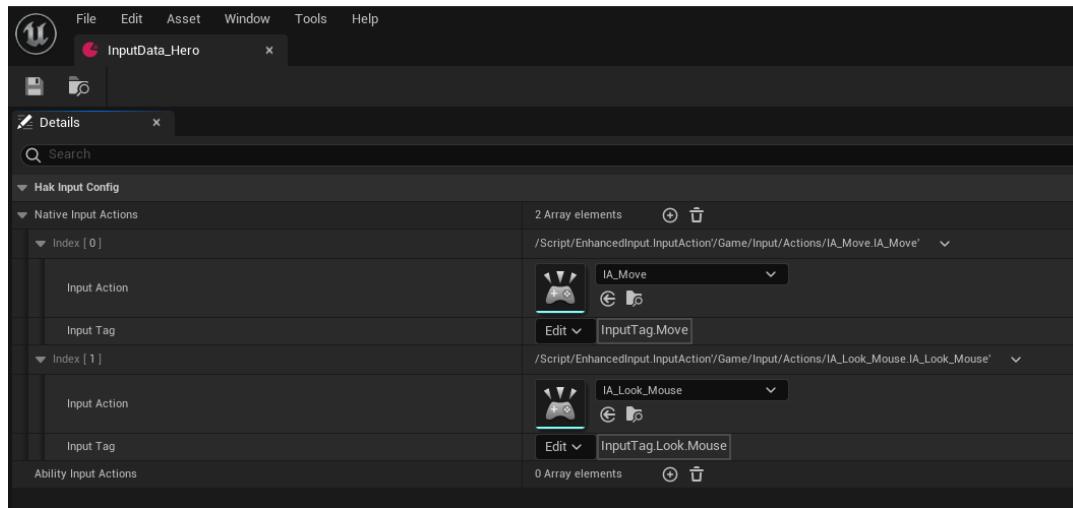
- /Content/Input0|| InputData_Hero를 추가해주자 (해당 에셋은 HakGame에 있다):



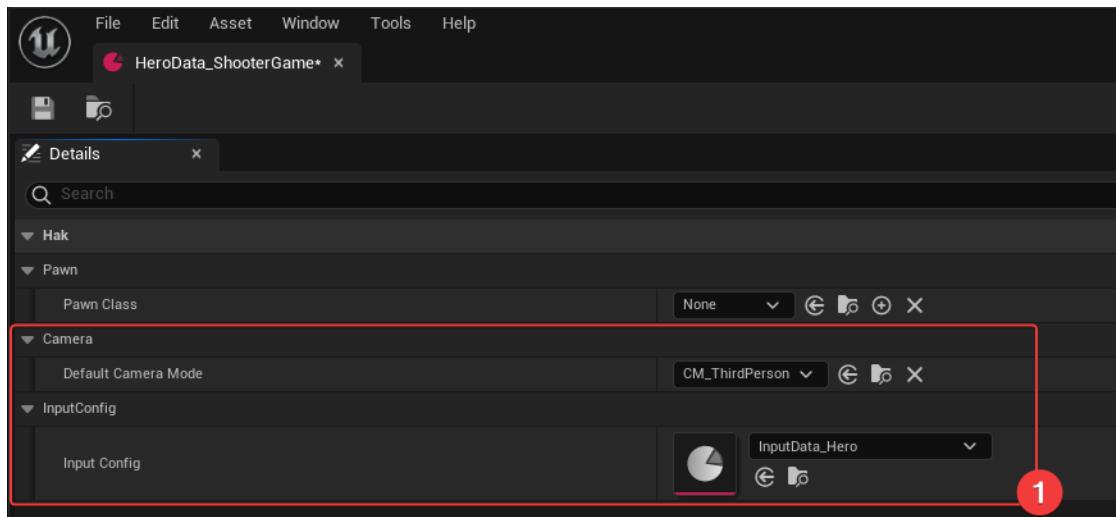
- HakInputConfig를 추가하자:



InputData_Hero에 속성값을 설정하자:



- HeroData_ShooterGame의 속성값을 설정하자 (PawnClass만 제외하고):

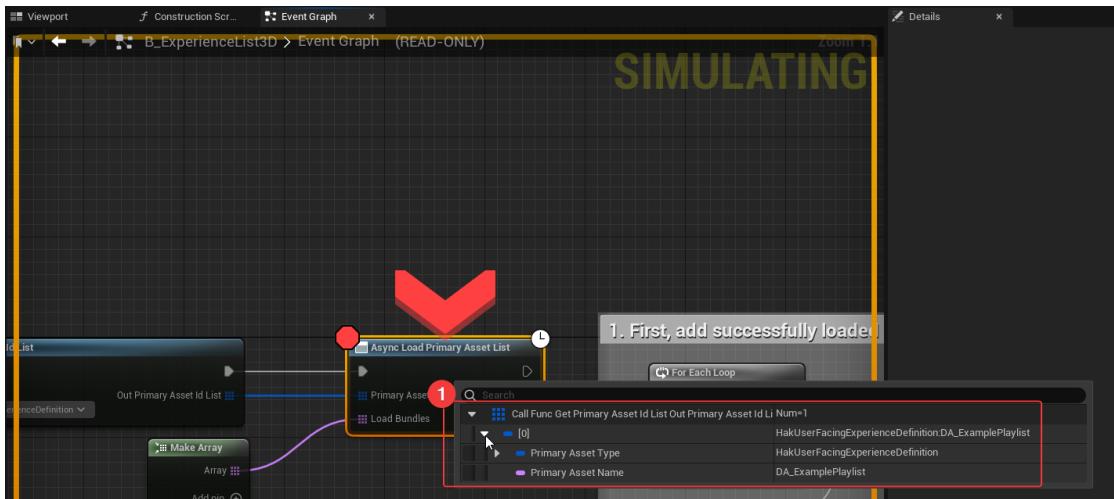


이제 ShooterCore의 큰 형태를 갖추었고, CommonSessionSubsystem을 활용하여, Traversal을 진행 할 수 있게 되었다.

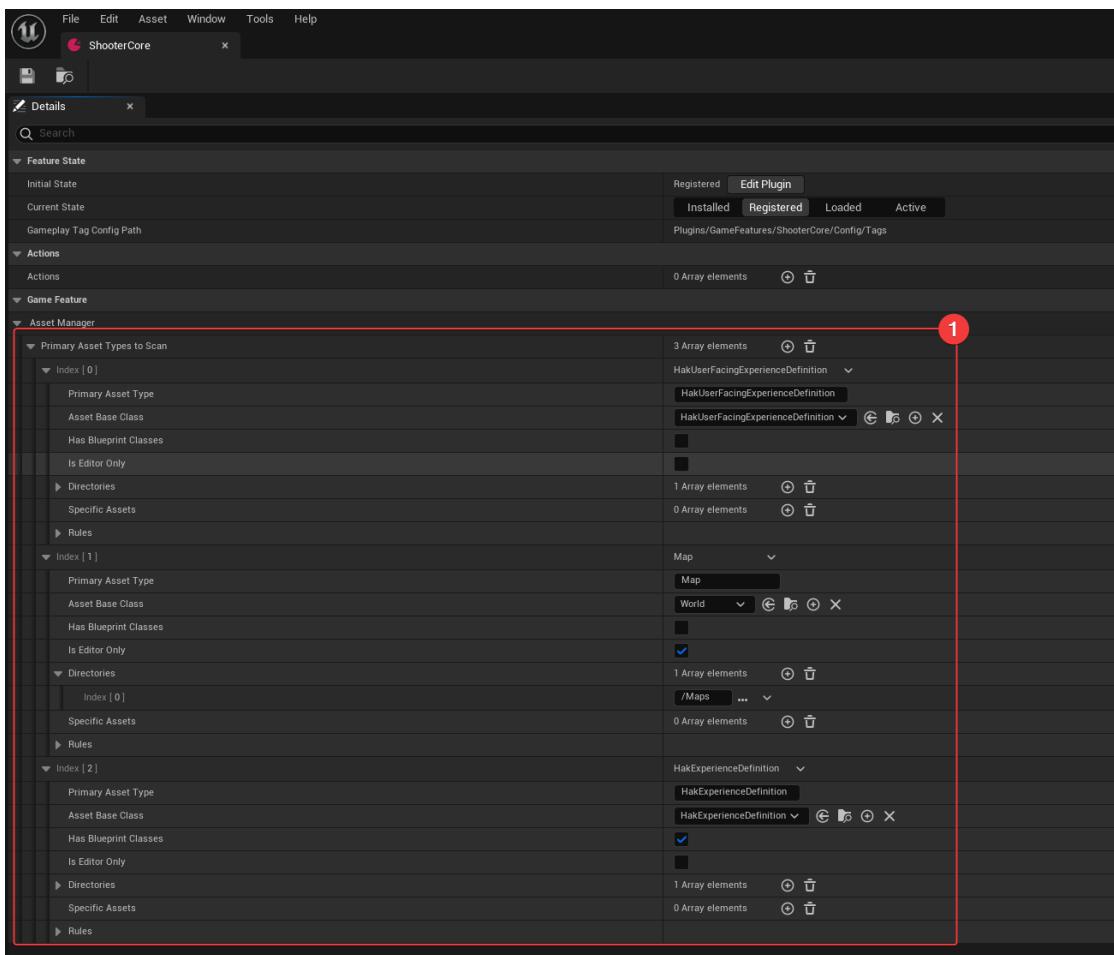
ShooterCore 맵 이동

▼ 펼치기

- 우리가 추가한 DA_ShooterGame_ShooterGym이 보이지 않는다:



□ 아래와 같이 PrimaryAssetTypes를 추가해주자:



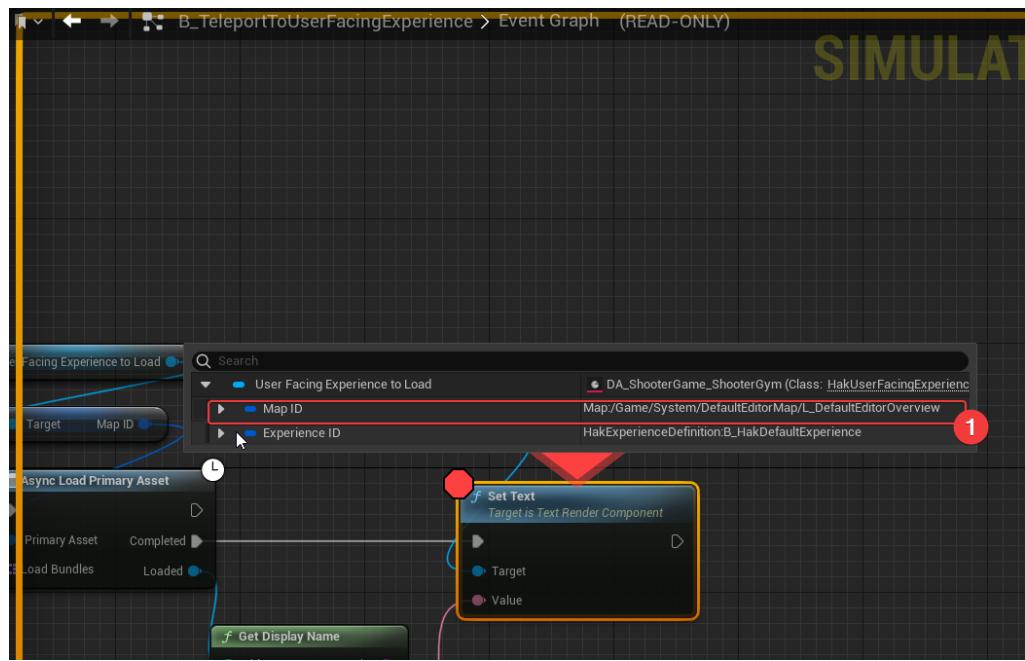
위 설정 이후, 에디터를 껐다가 켜야 다시 Scan을 진행하여, AssetManager 내부에 PrimaryAssetTypes들의 경로가 캐싱된다.

□ 아래와 같이 2개의 Platform이 생기지만 두가지 문제가 있다:

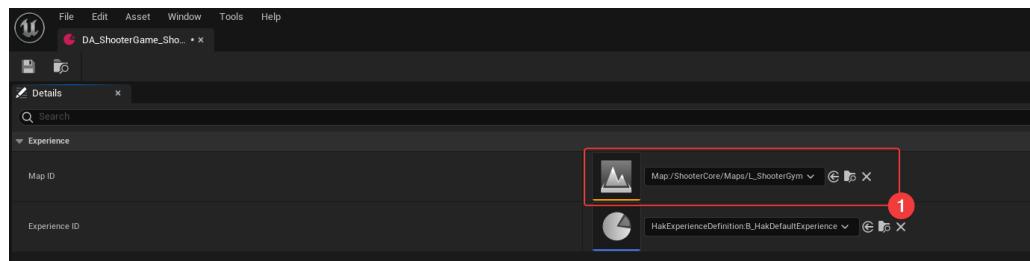


1. 텍스트가 똑같이 나온다...

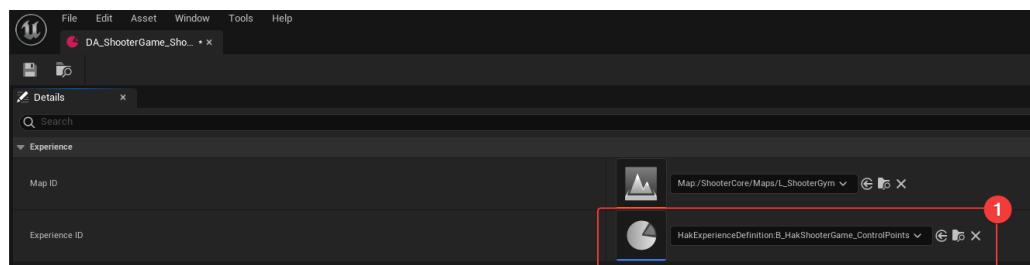
- 디버깅을 해보면, 아래와 같이 우리가 DA_ShooterGame_ShooterGym에 MapID를 잘못 설정하였다:



- DA_ShooterGame_ShooterGym을 잘 설정해주자:



- ExperienceID도 제대로 설정해주자:

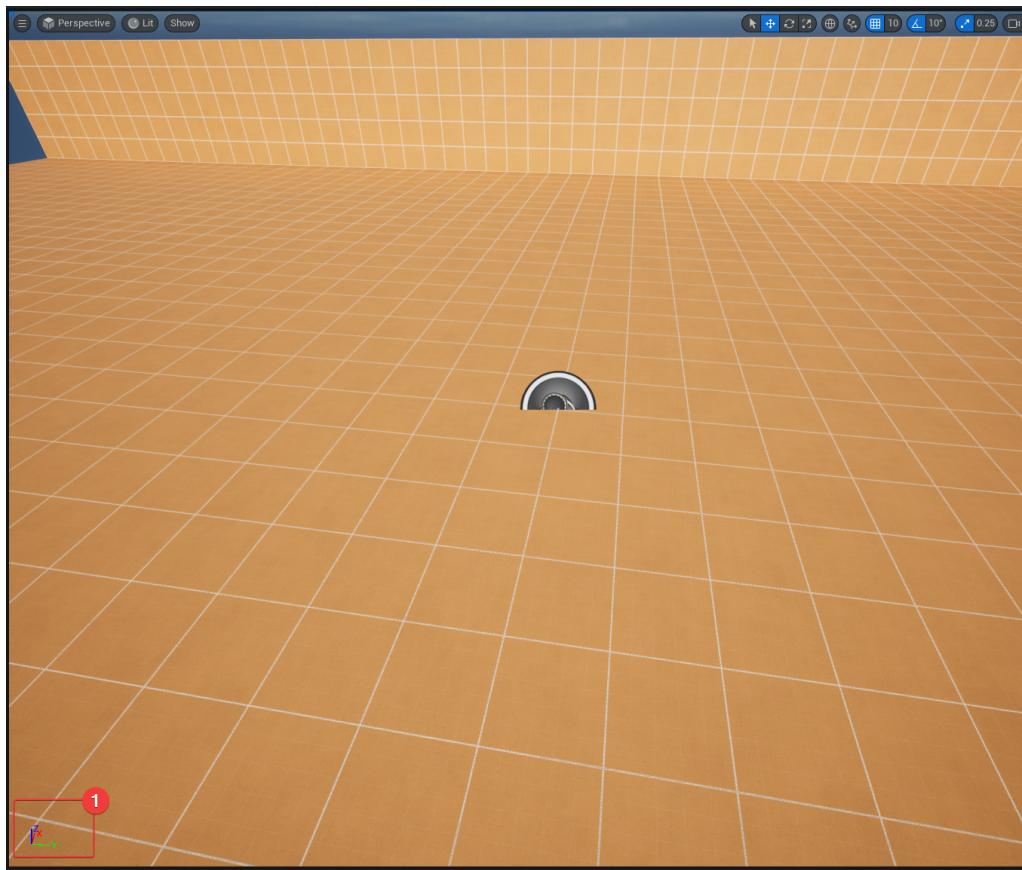


- 그럼 아래의 결과와 같다:

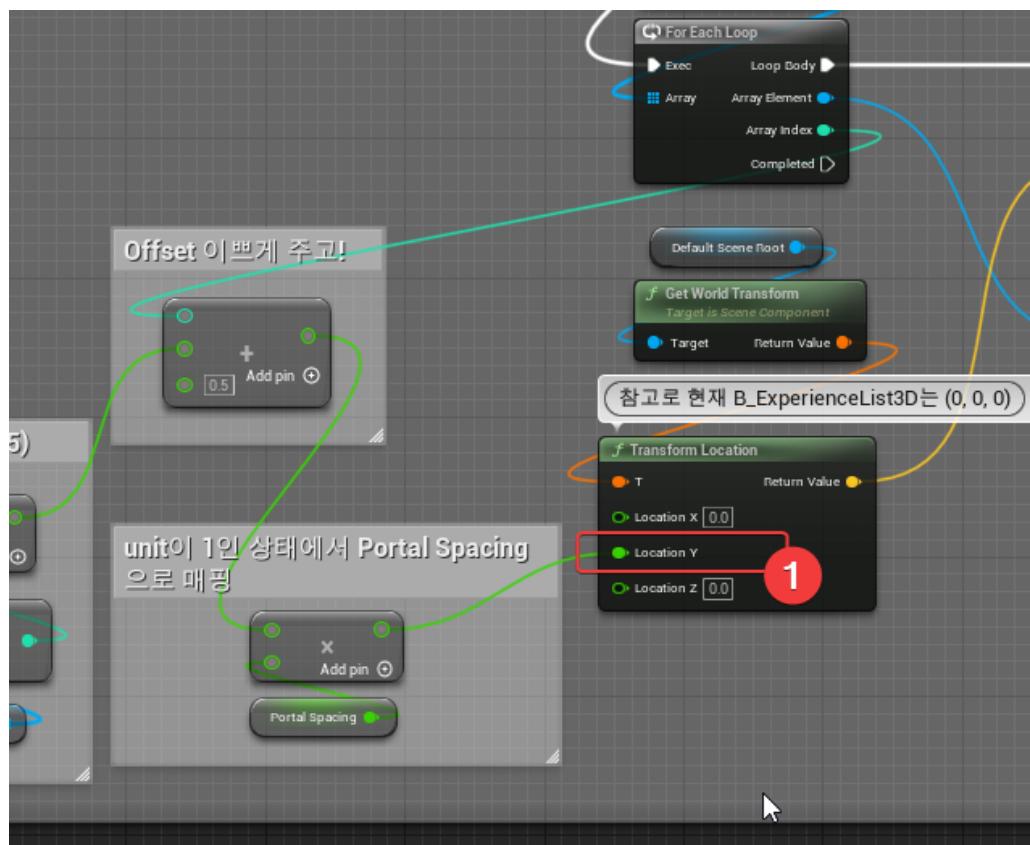


2. Platform의 위치 나열이 세로로 되어있다:

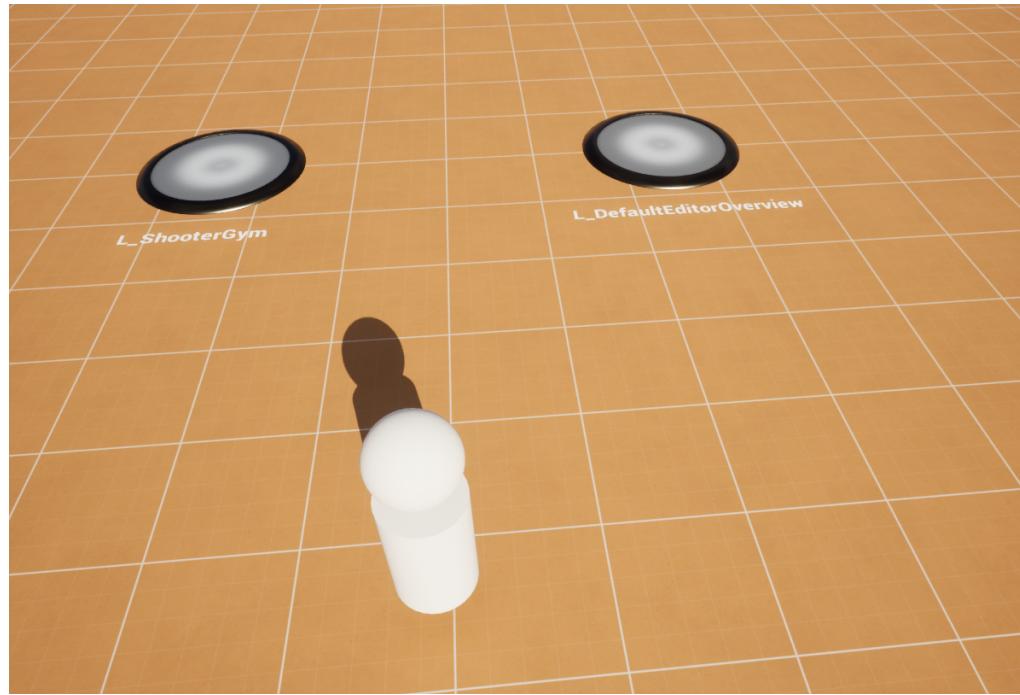
- 아래를 보면 위의 방향이 X로, 우리는 Y축으로 나열되길 기대한다:



□ 아래의 BP의 로직을 X에서 Y로 변경해주자:

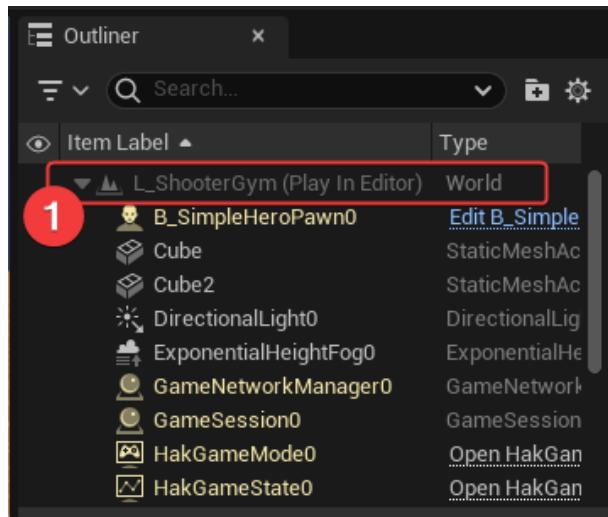


- 아래와 같이 잘 나열되어 Platform이 생성되었음을 확인할 수 있다:



□ 이동을 해보자:

- 우선 맵은 아래와 같이 L_ShooterGym으로 설정되었음을 확인할 수 있다:



- 그러나, SimplePawn이 조정도 가능하다?! (사실 우리는 PawnClass를 설정하지 않았다):



- 그럼 Experience0의 [DA_ShooterGame_ShooterGym](#)에 설정된 ExperienceID로 로딩이 안된것이다

□ 아래의 코드를 보도록 하자:

```

void AHakGameMode::HandleMatchAssignmentIfNotExpectingOne()
{
    // 해당 함수에서는 우리가 로딩할 Experience에 대해 FPrimaryAssetId를 생성하여, OnMatchAssignmentGiven으로 넘겨준다

    FPrimaryAssetId ExperienceId;

    // precedence order (highest wins)
    // - matchmaking assignment (if present)
    // - default experience

    UWorld* World = GetWorld();

    // fall back to the default experience
    // 일단 기본 옵션으로 default하게 B_HakDefaultExperience로 설정놓자
    if (!ExperienceId.IsValid())
    {
        ExperienceId = FPrimaryAssetId(FPrimaryAssetType("HakExperienceDefinition"), FName("B_HakDefaultExperience"));
    }

    // 필자가 이해한 HandleMatchAssignmentIfNotExpectingOne과 OnMatchAssignmentGiven()은 아직 직관적으로 이름이 와닿지 않는다고 생각한다
    // - 후일, 어느정도 Lyra가 구현되면, 해당 함수의 명을 더 이해할 수 있을 것으로 예상한다
    OnMatchAssignmentGiven(ExperienceId);
}

```

1 현재 우리는 무조건 B_HakDefaultExperience를 설정했다...

□ 우리는 앞서 CommonSessionSubsystem에서 Experience에 대한 정보를 ExtraArgs를 통해 CmdArgs로 URL에 포함시켜 넘겨주었다:

- 이를 활용하여, 아래와 같이 구현하면 된다:

□ HandleMatchAssignmentIfNotExpectingOne() 수정:

```

1 void AHakGameMode::HandleMatchAssignmentIfNotExpectingOne()
2 {
3     // 해당 함수에서는 우리가 로딩할 Experience에 대해 PrimaryAssetId를 생성하여, OnMatchAssignmentGiven으로 넘겨준다
4     FPrimaryAssetId ExperienceId;
5
6     // precedence order (highest wins)
7     // - matchmaking assignment (if present)
8     // - default experience
9
10    UWorld* World = GetWorld();
11
12    // 우리가 앞서, URL과 함께 ExtraArgs로 넘겼던 정보는 OptionsString에 저장되어 있다.
13    if (!ExperienceId.IsValid() && UGameplayStatics::HasOption(OptionsString, TEXT("Experience")))
14    {
15        // Experience의 Value를 가져와서, PrimaryAssetId를 생성해준다: 이때, HakExperienceDefinition의 Class 이름을 사용한다
16        const FString ExperienceFromOptions = UGameplayStatics::ParseOption(OptionsString, TEXT("Experience"));
17        ExperienceId = FPrimaryAssetId(FPrimaryAssetType(UHakExperienceDefinition::StaticClass()->GetFName()), FName(*ExperienceFromOptions));
18    }
19
20    // fall back to the default experience
21    // 일단 기본 옵션으로 default하게 B_HakDefaultExperience로 설정놓자
22    if (!ExperienceId.IsValid())
23    {
24        ExperienceId = FPrimaryAssetId(FPrimaryAssetType("HakExperienceDefinition"), FName("B_HakDefaultExperience"));
25    }
26
27    // 뭔가 이해한 HandleMatchAssignmentIfNotExpectingOne과 OnMatchAssignmentGiven()은 아직 직관적으로 이름이 와닿지 않는다고 생각한다
28    // - 후일, 어느정도 Lyra가 구현되면, 해당 함수의 명을 더 이해할 수 있을 것으로 예상한다
29    OnMatchAssignmentGiven(ExperienceId);
30 }

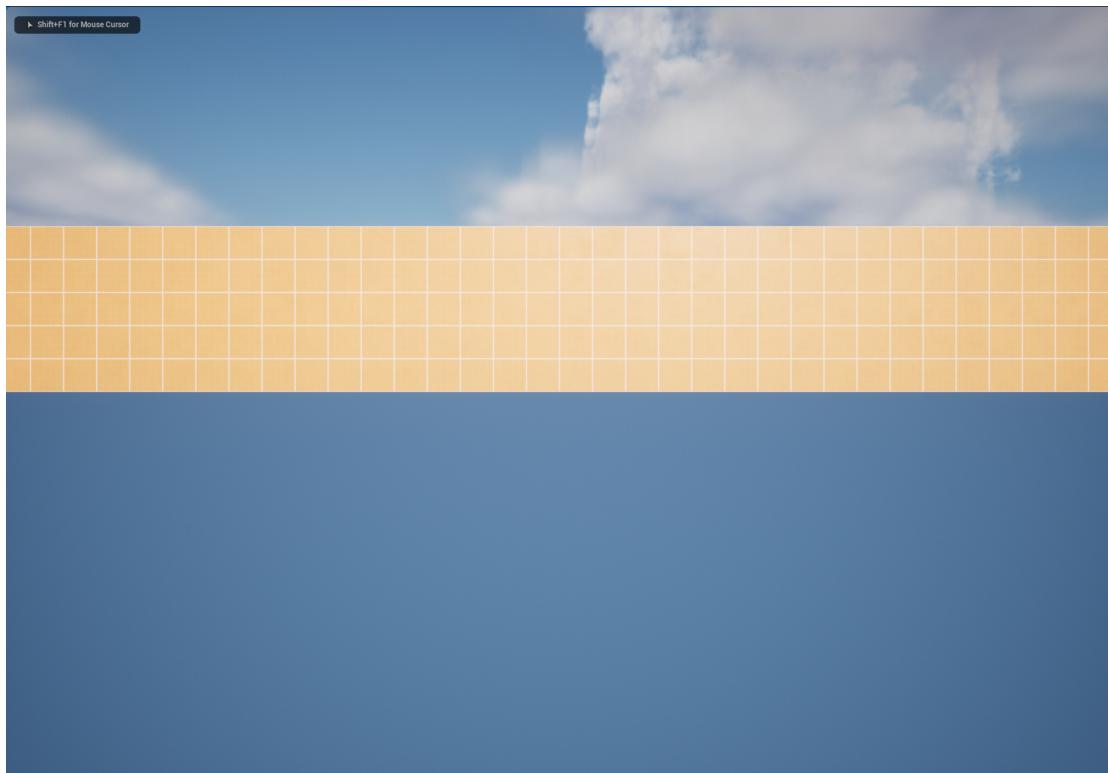
```

□ 해당 코드 디버깅해보기:

```

105   void AHakGameMode::HandleMatchAssignmentIfNotExpectingOne()
106   {
107       // 해당 함수에서는 우리가 로딩할 Experience에 대해 PrimaryAssetId를 생성하여, OnMatchAssignmentGiven으로 넘겨준다
108       FPrimaryAssetId ExperienceId;
109
110       // precedence order (highest wins)
111       // - matchmaking assignment (if present)
112       // - default experience
113
114       UWorld* World = GetWorld();
115
116       // 우리가 앞서, URL과 함께 ExtraArgs로 넘겼던 정보는 OptionsString에 저장되어 있다.
117       if (!ExperienceId.IsValid() && UGameplayStatics::HasOption(OptionsString, TEXT("Experience")))
118       {
119           // Experience의 Value를 가져와서, PrimaryAssetId를 생성해준다: 이때, HakExperienceDefinition의 Class 이름을 사용한다
120           const FString ExperienceFromOptions = UGameplayStatics::ParseOption(OptionsString, TEXT("Experience"));
121           ExperienceId = FPrimaryAssetId(FPrimaryAssetType(UHakExperienceDefinition::StaticClass()->GetFName()), FName(*ExperienceFromOptions));
122       }
123
124       // fall back to the default experience
125       // 일단 기본 옵션으로 default하게 B_HakDefaultExperience로 설정놓자
126       if (!ExperienceId.IsValid())
127       {
128           ExperienceId = FPrimaryAssetId(FPrimaryAssetType("HakExperienceDefinition"), FName("B_HakDefaultExperience"));
129       }
130
131       // 뭔가 이해한 HandleMatchAssignmentIfNotExpectingOne과 OnMatchAssignmentGiven()은 아직 직관적으로 이름이 와닿지 않는다고 생각한다
132       // - 후일, 어느정도 Lyra가 구현되면, 해당 함수의 명을 더 이해할 수 있을 것으로 예상한다
133       OnMatchAssignmentGiven(ExperienceId);
134   }
135 }
```

- 결과는 아래와 같다:



- 카메라 조정도 되지 않고, 우리가 의도한 Experience로 로딩이 잘 되었다.

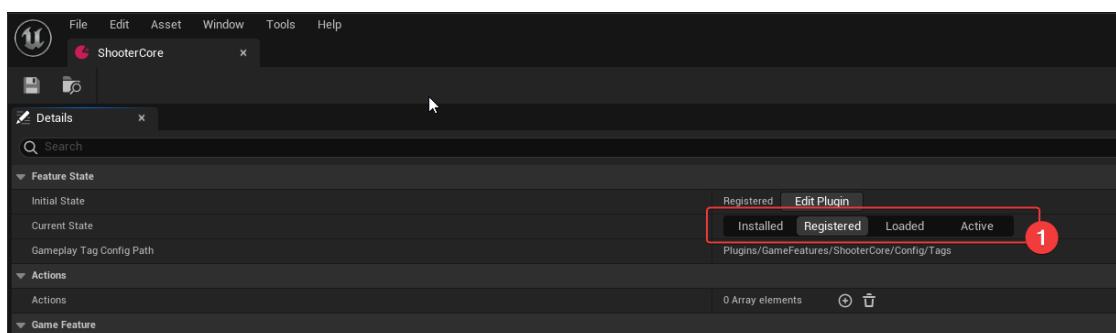


여러분들이 코드에서 직접 어떤 Experience가 로딩되었는지 확인해도 된다!

GameFeature 활성화

▼ 펼치기

- 아래와 같이 ShooterCore의 Map과 Experience도 들어가도, ShooterCore의 Game Feature는 활성화되지 않는다:



- ExperienceManagerComponent.h/.cpp에 아래와 같이 추가 구현을 해줘야 한다:

- OnExperienceLoadComplete:

□ ExperienceManagerComponent의 멤버 변수를 추가해주자:

```
/**  
 * HakExperienceManagerComponent  
 * - 말 그대로, 해당 component는 game state를 owner로 가지면서, experience의 상태 정보를 가지고 있는 component이다  
 */  
UCLASS()  
class UHakExperienceManagerComponent : public UGameStateComponent  
{  
    GENERATED_BODY()  
public:  
    UHakExperienceManagerComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());  
  
    /**  
     * member methods  
     */  
    bool IsExperienceLoaded() { return (LoadState == EHakExperienceLoadState::Loaded) && (CurrentExperience != nullptr); }  
  
    /**  
     * 아래의 OnExperienceLoaded에 바인딩하거나, 이미 Experience 로딩이 완료되었다면 바로 호출함  
     */  
    void CallOrRegister_OnExperienceLoaded(FOnHakExperienceLoaded::FDelegate&& Delegate);  
  
    void ServerSetCurrentExperience(FPrimaryAssetId ExperienceId);  
    void StartExperienceLoad();  
    void OnExperienceLoadComplete();  
    void OnExperienceFullLoadCompleted();  
    const UHakExperienceDefinition* GetCurrentExperienceChecked() const;  
  
    /**  
     * member variables  
     */  
  
    /**  
     * 참고로 해당 멤버 변수는 Lyra에서는 'ReplicatedUsing='으로 선언되어있다:  
     * - 현재 우리는 아직 Replication을 신경쓰지 않을 것이기에, 최대한 네트워크 서버 코드를 배제하도록 하겠다  
     */  
    UPROPERTY()  
    TSharedPtr<const UHakExperienceDefinition> CurrentExperience;  
  
    /** Experience의 로딩 상태를 모니터링 */  
    EHakExperienceLoadState LoadState = EHakExperienceLoadState::Unloaded;  
  
    /** Experience 로딩이 완료된 이후, Broadcasting Delegate */  
    FOnHakExperienceLoaded OnExperienceLoaded;  
  
    /** 활성화된 GameFeature Plugin들 */  
    int32 NumGameFeaturePluginsLoading = 0;  
    TArray< FString> GameFeaturePluginURLs;  
};
```

1

□ EHakExperienceLoadState에 LoadingGameFeatures 추가하기:

```
enum class EHakExperienceLoadState  
{  
    Unloaded,  
    Loading,  
    LoadingGameFeatures,  
    Loaded,  
    Deactivating,  
};
```

1

□ OnGameFeaturePluginLoadComplete 추가:

```

#include "CoreMinimal.h"
#include "Components/GameStateComponent.h"
#include "GameFeaturePluginOperationResult.h" ①
#include "HakExperienceManagerComponent.generated.h"

/** Forward declaration */
class UHakExperienceDefinition;

enum class EHakExperienceLoadState
{
    Unloaded,
    Loading,
    LoadingGameFeatures,
    Loaded,
    Deactivating,
};

DECLARE_MULTICAST_DELEGATE_OneParam(FOnHakExperienceLoaded, const UHakExperienceDefinition*);

/**
 * HakExperienceManagerComponent
 * - 말 그대로, 해당 component는 game state를 owner로 가지면서, experience의 상태 정보를 가지고 있는 component이다
 * - 물론 아니라, manager라는 단어가 포함되어 있듯이, experience 토큰 상태 업데이트 및 이벤트를 관리한다
 */
UCLASS()
class UHakExperienceManagerComponent : public UGameStateComponent
{
    GENERATED_BODY()
public:
    UHakExperienceManagerComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /**
     * member methods
     */
    bool IsExperienceLoaded() { return (LoadState == EHakExperienceLoadState::Loaded) && (CurrentExperience != nullptr); }

    /**
     * 아래의 OnExperienceLoaded에 바인딩하거나, 이미 Experience 로딩이 완료되었다면 바로 호출함
     */
    void CallOrRegister_OnExperienceLoaded(FOnHakExperienceLoaded::FDelegate&& Delegate);

    void ServerSetCurrentExperience(FPrimaryAssetId ExperienceId);
    void StartExperienceLoad();
    void OnExperienceLoadComplete();
    void OnGameFeaturePluginLoadComplete(const UE::GameFeatures::FResult& Result); ②
    void OnExperienceFullLoadCompleted();
    const UHakExperienceDefinition* GetCurrentExperienceChecked() const;

    /**
     * member variables
     */
};

```

```

void UHakExperienceManagerComponent::OnGameFeaturePluginLoadComplete(const UE::GameFeatures::FResult& Result)
{
}

```

□ OnExperienceLoadComplete 변경:

```

void UHakExperienceManagerComponent::OnExperienceLoadComplete()
{
    // FrameNumber를 증명해서 보자
    static int32 OnExperienceLoadComplete_FrameNumber = GFrameNumber;

    check(LoadState == EHakExperienceLoadState::Loading);
    check(CurrentExperience);

    // 아직 활성화된 GameFeature Plugin의 URL을 끌리어준다
    GameFeaturePluginURLs.Reset();

    auto CollectGameFeaturePluginURLs = [this = this](const UPrimaryDataAsset* Context, const TArray< FString>& FeaturePluginList)
    {
        // FeaturePluginList를 순회하여, PluginURL을 ExperienceManagerComponent의 GameFeaturePluginURLs에 추가해준다
        for (const FString& PluginName : FeaturePluginList)
        {
            FString PluginURL;
            if (UGameFeaturesSubsystem::Get().GetPluginURLByName(PluginName, PluginURL))
            {
                This->GameFeaturePluginURLs.AddUnique(PluginURL);
            }
        };
    };

    // GameFeaturesToEnable에 있는 Plugin만 일단 활성화할 GameFeature Plugin 후보군으로 등록
    CollectGameFeaturePluginURLs(CurrentExperience, CurrentExperience->GameFeaturesToEnable);

    // GameFeaturePluginURLs에 등록된 Plugin을 로딩 및 활성화:
    NumGameFeaturePluginsLoading = GameFeaturePluginURLs.Num();
    if (NumGameFeaturePluginsLoading)
    {
        LoadState = EHakExperienceLoadState::LoadingGameFeatures;
        for (const FString& PluginURL : GameFeaturePluginURLs)
        {
            // 해당 Plugin이 로드 및 활성화 이후, OnGameFeaturePluginLoadComplete 콜백 함수 등록
            // 해당 함수를 실행하도록 하자
            UGameFeaturesSubsystem::Get().LoadAndActivateGameFeaturePlugin(PluginURL, FGameFeaturePluginLoadComplete::Create UObject(this, &ThisClass::OnGameFeaturePluginLoadComplete));
        }
    }
    else
    {
        // 해당 함수가 불리는 것은 앞서 보았던 StreamableDelegateDelayHelper 의해 불림
        OnExperienceFullLoadCompleted();
    }
}

```

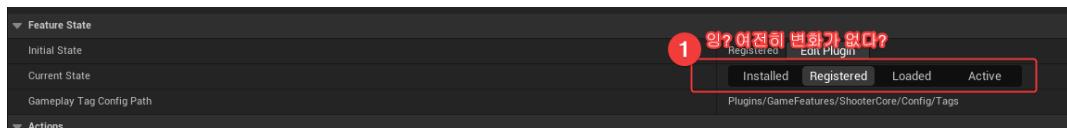
□ OnGameFeaturePluginLoadComplete() 구현:

```

void UHakExperienceManagerComponent::OnGameFeaturePluginLoadComplete(const UE::GameFeatures::FResult& Result)
{
    // 매 GameFeature Plugin이 로딩될 때, 해당 함수가 콜백으로 불린다
    NumGameFeaturePluginsLoading--;
    if (NumGameFeaturePluginsLoading == 0)
    {
        // GameFeaturePlugin 로딩이 다 끝났을 경우, 기준대로 Loaded로서, OnExperienceFullLoadCompleted 호출한다
        // GameFeaturePlugin 로딩과 활성화가 끝났다면? UGameFeatureAction을 활성화해야겠지 (조금만 있다가 하장)
        OnExperienceFullLoadCompleted();
    }
}

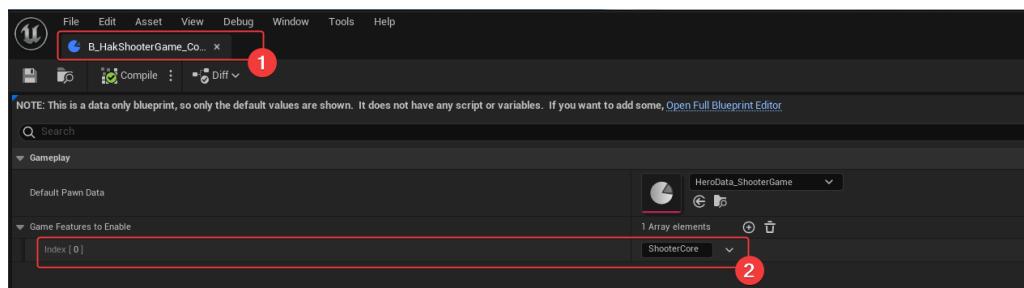
```

- 그럼 아래와 같이 ShooterCore로 텔레포트 하였을 경우, Plugin이 활성화 됨을 확인:



□ 앞서, 코드에서 보았듯이, Experience에 대상이 되는 Game Feature를 지정해야 한다:

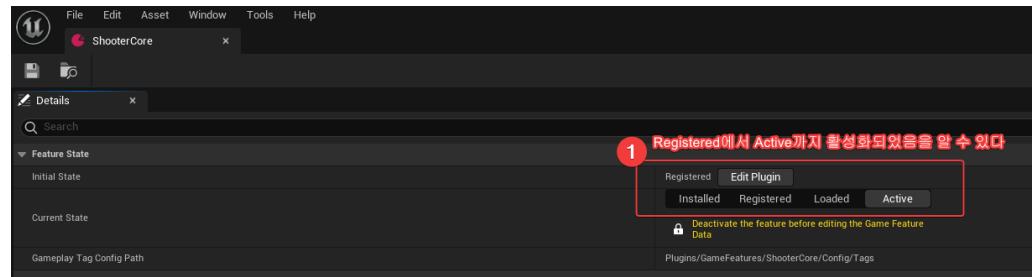
- 아래와 같이 B_HakShooterGame_ControlPoints에 Game Features to Enable에 ShooterCore를 추가해주자:



□ 그리고 디버깅:

```

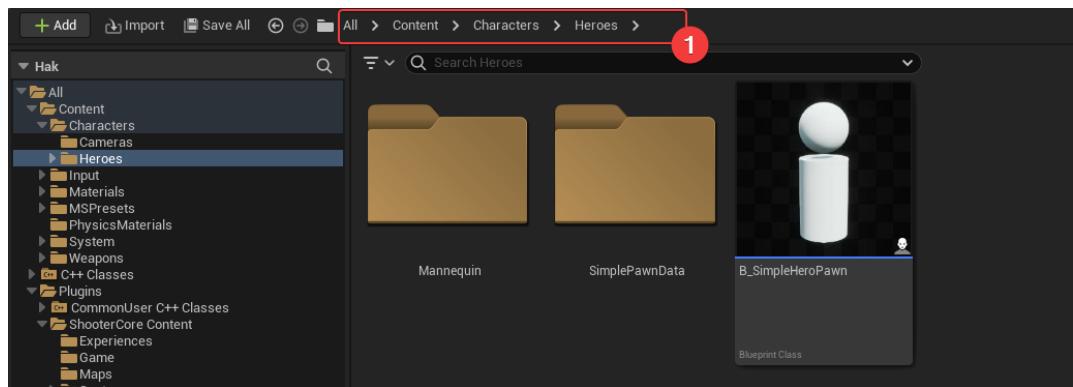
129 void UHakExperienceManagerComponent::OnExperienceLoadComplete()
130 {
131     // FrameNumber를 주목해서 보자
132     static int32 OnExperienceLoadComplete_FrameNumber = GFrameNumber;
133
134     check(LoadState == EHakExperienceLoadState::Loading);
135     check(CurrentExperience);
136
137     // 이전 활성화된 GameFeature Plugin의 URL을 클리어해준다
138     GameFeaturePluginURLs.Reset();
139
140     auto CollectGameFeaturePluginURLs = [This = this](const UPrimaryDataAsset* Context, const TArray< FString>& FeaturePluginList)
141     {
142         // FeaturePluginList를 순회하며, PluginURL을 ExperienceManagerComponent의 GameFeaturePluginURLs에 추가해준다
143         for (const FString& PluginName : FeaturePluginList)
144         {
145             FString PluginURL;
146             if (UGameFeaturesSubsystem::Get().GetPluginURLByName(PluginName, PluginURL))
147             {
148                 This->GameFeaturePluginURLs.AddUnique(PluginURL);
149             }
150         }
151     };
152
153     // GameFeaturesToEnable에 있는 Plugin만 일단 활성화할 GameFeature Plugin 후보군으로 등록
154     CollectGameFeaturePluginURLs(CurrentExperience, CurrentExperience->GameFeaturesToEnable); 1
155
156     // GameFeaturePluginURLs에 등록된 Plugin을 토底 및 활성화
157     NumGameFeaturePluginsLoading = GameFeaturePluginURLs.Num();
158     if (NumGameFeaturePluginsLoading)
159     {
160         LoadState = EHakExperienceLoadState::LoadingGameFeatures;
161         for (const FString& PluginURL : GameFeaturePluginURLs)
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
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
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
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
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1189
1190
1191
1192
1193
1194
1195
1196
1197
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2159
2160
2161
2162
2163
2164
2165
21
```



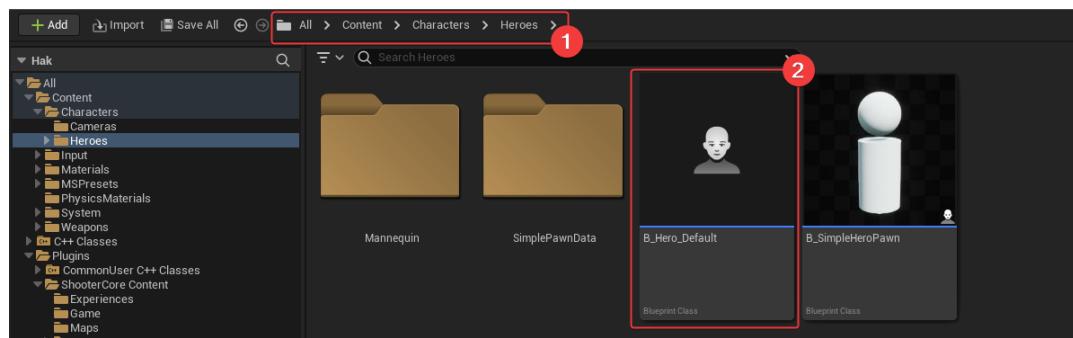
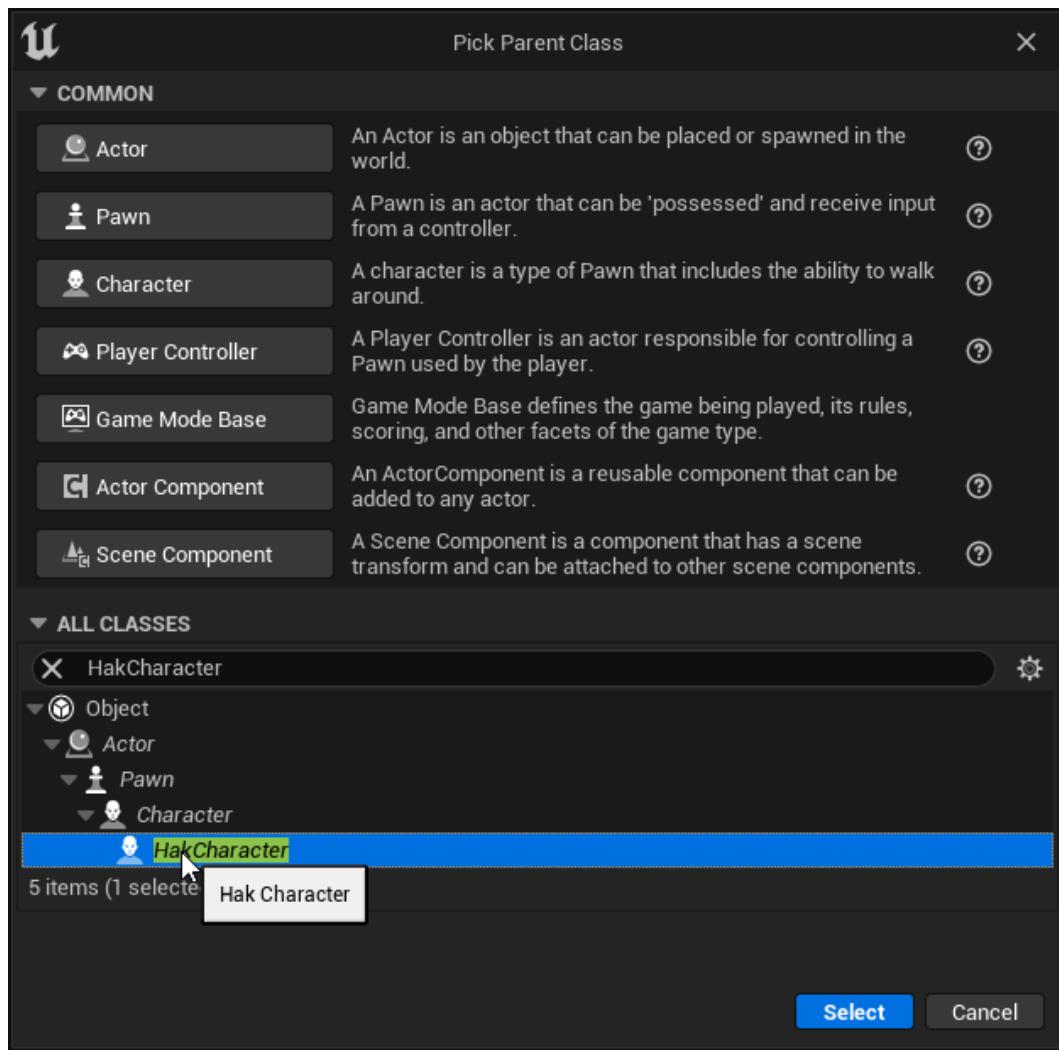
Input 활성화

▼ 펼치기

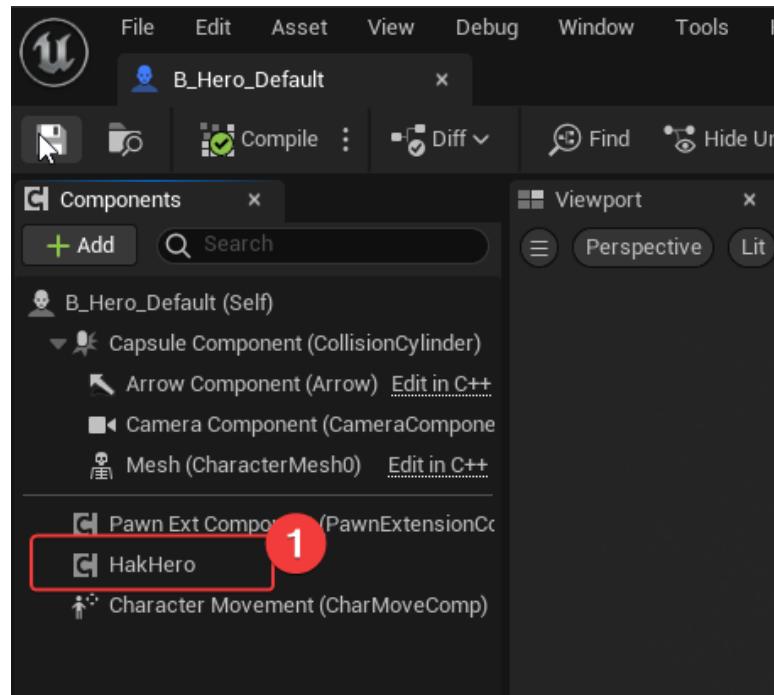
- 앞서, 우리가 SimpleHeroPawn에서 진행했듯이, Input은 HeroComponent에서 활성화가 진행된다:
 - ShooterCore의 HakCharacter를 만들어주자: `B_Hero_ShooterMannequin`
- `B_Hero_ShooterMannequin`가 상속 받을 `B_Hero_Default` 생성:
 - 해당 경로로 위치하자:



- `HakCharacter`를 상속받는 `B_Hero_Default`를 만들어주자:

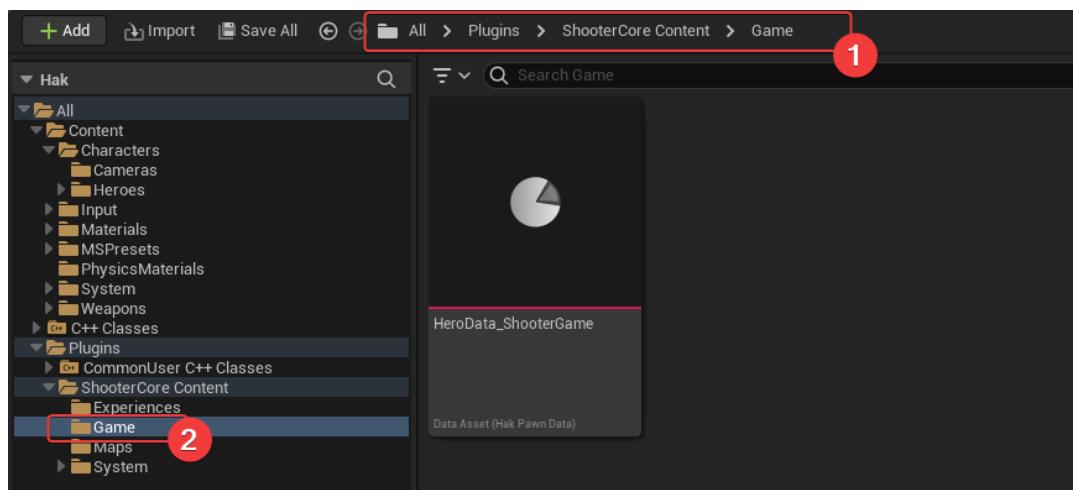


□ 기존 B_SimpleHerPawn과 같이 HeroComponent를 B_Hero_Default에 추가시켜주자:

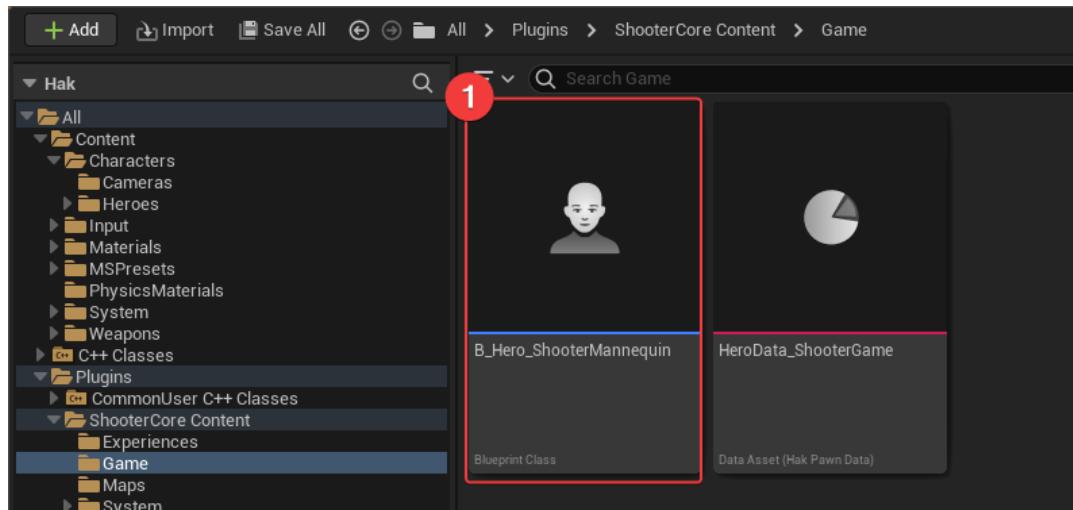


B_Hero_ShooterMannequin을 생성하자:

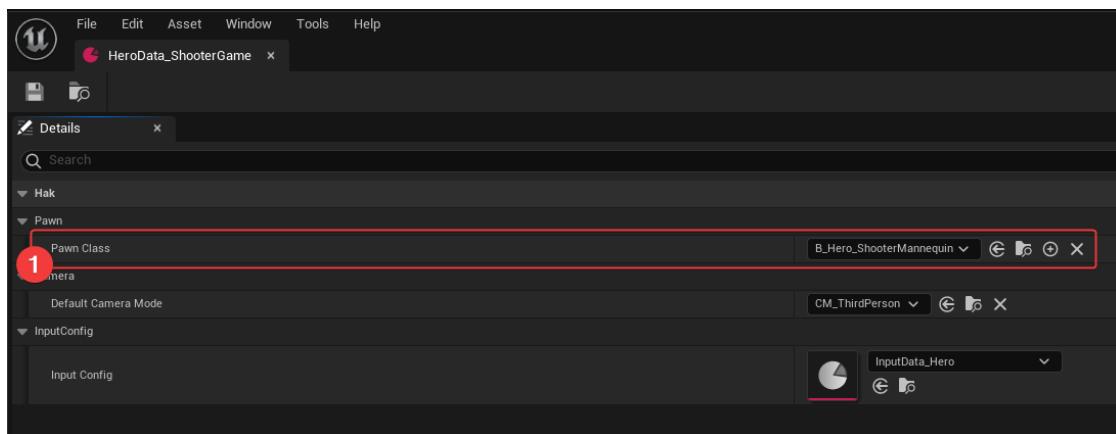
- 아래의 경로에 위치하자:



B_Hero_Default를 상속받아 B_Hero_ShooterMannequin을 생성하자:



□ B_Hero_ShooterMannequin을 PawnClass로 HeroData_ShooterGame에 추가하자:



□ 기존 SimpleHeroPawn을 회고해보면, HeroComponent에서 Input을 설정하는데, 디버깅해보자:

```

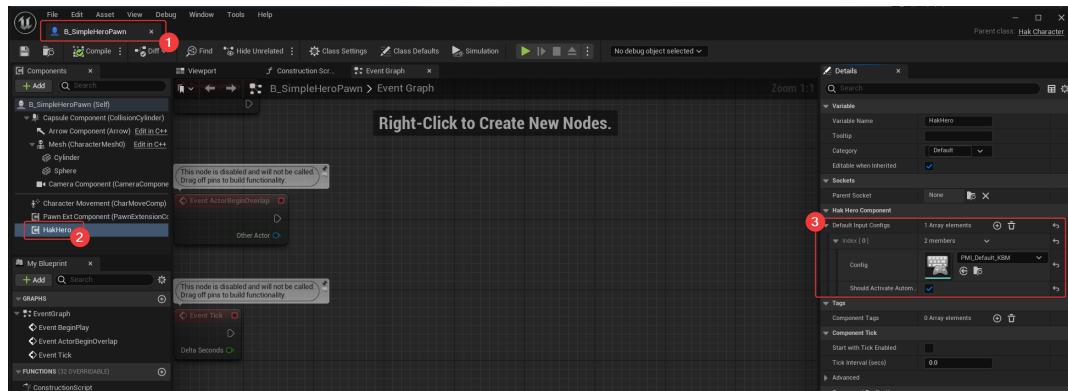
199 void UHakHeroComponent::InitializePlayerInput(UInputComponent* PlayerInputComponent)
200 {
201     check(PlayerInputComponent);
202
203     const APawn* Pawn = GetPawn<APawn>();
204     if (!Pawn)
205     {
206         return;
207     }
208
209     // LocalPlayer를 가져오기 위해
210     const APlayerController* PC = GetController<APlayerController>();
211     check(PC);
212
213     // EnhancedInputLocalPlayerSubsystem 가져오기 위해
214     const ULocalPlayer* LP = PC->GetLocalPlayer();
215     check(LP);
216
217     UEnhancedInputLocalPlayerSubsystem* Subsystem = LP->GetSubsystem<UEnhancedInputLocalPlayerSubsystem>();
218     check(Subsystem);
219
220     // EnhancedInputLocalPlayerSubsystem의 MappingContext를 비워준다:
221     Subsystem->ClearAllMappings();
222
223     // PawnExtensionComponent -> PawnData -> InputConfig 존재 유무 판단:
224     if (const UHakPawnExtensionComponent* PawnExtComp = UHakPawnExtensionComponent::FindPawnExtensionComponent(Pawn))
225     {
226         if (const UHakPawnData* PawnData = PawnExtComp->GetPawnData<UHakPawnData>())
227         {
228             if (const UHakInputConfig* InputConfig = PawnData->InputConfig)
229             {
230                 const FHakGameplayTags& GameplayTags = FHakGameplayTags::Get();
231
232                 // HeroComponent 가지고 있는 Input Mapping Context를 순회하며, EnhancedInputLocalPlayerSubsystem에
233                 // DefaultInputConfigs가 넣어있다.
234                 for (const FHakMappableConfigPair& Pair : DefaultInputConfigs)
235                 {
236                     if (Pair.bShouldActivateAutomatically)
237                     {
238                         FModifyContextOptions Options = {};
239                         Options.bIgnoreAllPressedKeysUntilRelease = false;
240
241                         // 내부적으로 Input Mapping Context를 추가한다:
242                         // - AddPlayerMappableConfig를 간단히 보는 것을 추천
243                         Subsystem->AddPlayerMappableConfig(Pair.Config.LoadSynchronous(), Options);
244                     }
245                 }
246             }
247         }
248     }
249 }

```

1 DefaultInputConfigs가 넣어있다.

2 PlayerMappableConfig가 추가되지 않는다...

- 이전에는 아래와 같이 우리가 임의적으로 아래와 같이 DefaultInputConfigs에 추가해줬다:



- 우리는 이전과 달리, 이번에는 GameFeatureAction을 활용하여, InputConfig를 추가한다

GameFeatureAction

▼ 펼치기

- ExperienceDefinition에 GameFeatureAction을 추가하자:

```

#pragma once

#include "Engine/DataAsset.h"
#include "HakExperienceDefinition.generated.h"

/** forward declaration */
class UHakPawnData;
class UGameFeatureAction;
class UHakExperienceActionSet;

/**
 * UHakExperienceDefinition
 * - definition of an experience
 */
UCLASS()
class UHakExperienceDefinition : public UPrimaryDataAsset
{
    GENERATED_BODY()

public:
    UHakExperienceDefinition(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /** the default pawn class to spawn for players */
    UPROPERTY(EditDefaultsOnly, Category=Gameplay)
    TObjectPtr<UHakPawnData> DefaultPawnData;

    /** lost if game feature plugins this experience wants to have active */
    // 해당 property는 단순히 마킹 및 기억용으로 남겨놓도록 하겠다;
    // - GameMode에 따라 필요한 GameFeature plugin들을 로딩하는데 이에 대한 연결고리로 생각하면 된다 (현재는 쓰지 않음)
    UPROPERTY(EditDefaultsOnly, Category=Gameplay)
    TArray< FString> GameFeaturesToEnable;

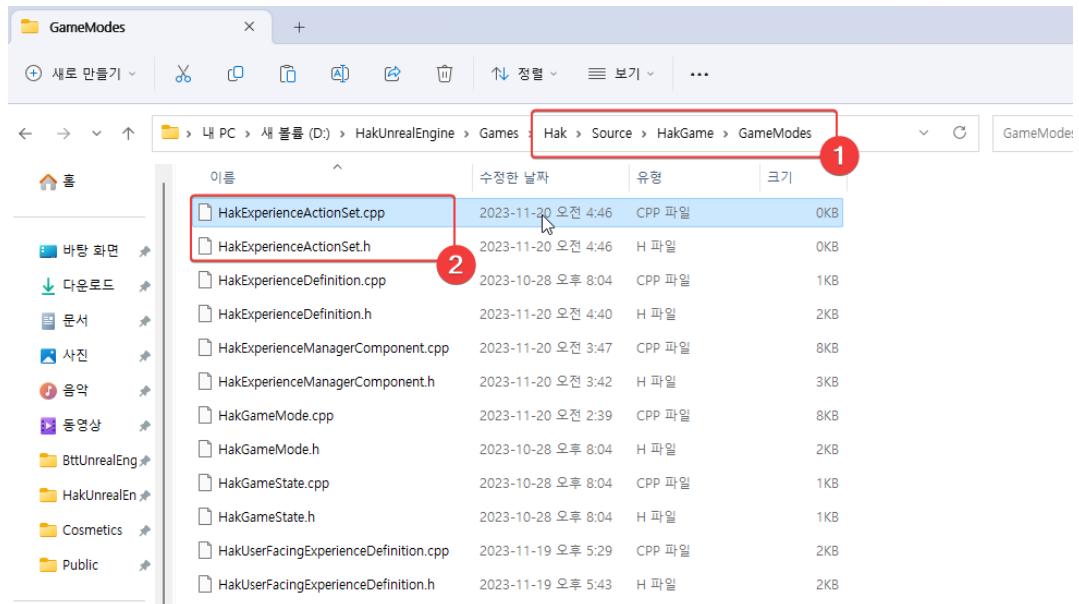
    /**
     * ExperienceActionSet은 UGameFeatureAction의 Set이며, Gameplay 용도에 맞게 분류의 목적으로 사용한다 */
    UPROPERTY(EditDefaultsOnly, Category = Gameplay)
    TArray< TObjectPtr<UHakExperienceActionSet>> ActionSets;

    /**
     * 일반적인 GameFeatureAction으로서 추가 */
    UPROPERTY(EditDefaultsOnly, Category="Actions")
    TArray< TObjectPtr<UGameFeatureAction>> Actions;
};

```

HakExperienceActionSet 추가:

- 아래의 경로에 파일 추가:



HakExperienceActionSet.h/.cpp:

```

#pragma once

#include "CoreMinimal.h"
#include "Engine/DataAsset.h"
#include "HakExperienceActionSet.generated.h"

class UGameFeatureAction;

/**
 * DataAsset로서 UGameFeatureAction을 카테고리화 시킬 때, 유용한 클래스
 */
UCLASS(BlueprintType)
class UHakExperienceActionSet : public UPrimaryDataAsset
{
    GENERATED_BODY()
public:
    UHakExperienceActionSet();

    /**
     * member variables
     */
    UPROPERTY(EditAnywhere, Category="Actions to Perform")
    TArray< TObjectPtr<UGameFeatureAction>> Actions;
};

```

```

#include "HakExperienceActionSet.h"
#include "GameFeatureAction.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(HakExperienceActionSet)

UHakExperienceActionSet::UHakExperienceActionSet()
    : Super()
{
}

```

- HakExperienceDefinition.cpp에 헤더파일 추가:

```

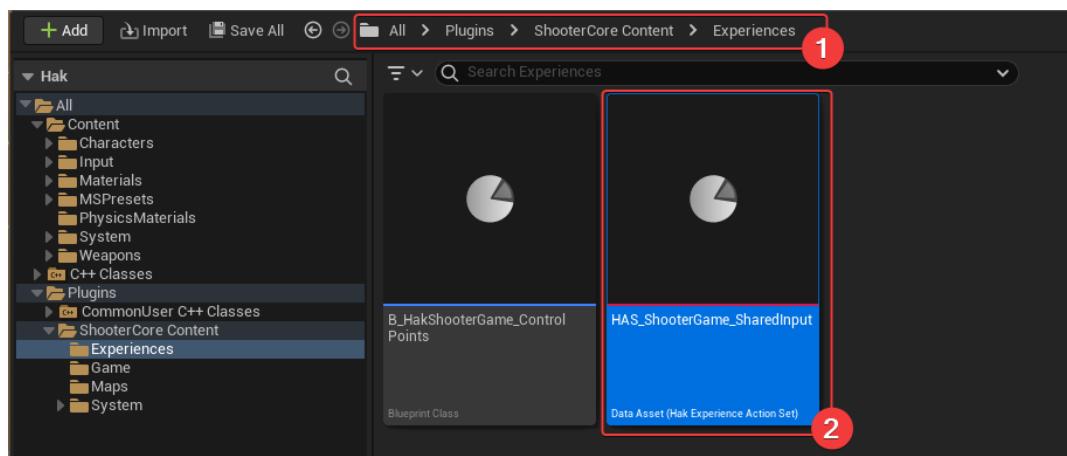
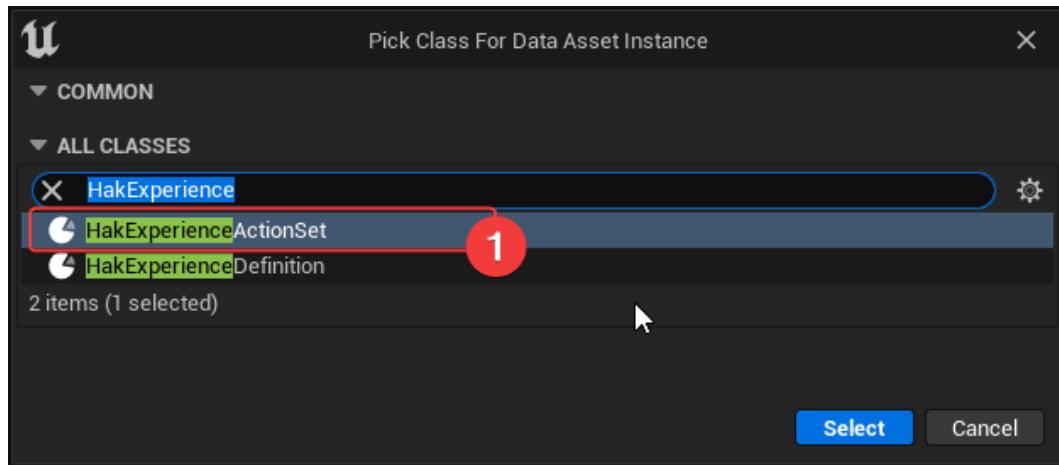
#include "HakExperienceDefinition.h"
#include "HakExperienceActionSet.h" ①
#include "GameFeatureAction.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(HakExperienceDefinition)

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

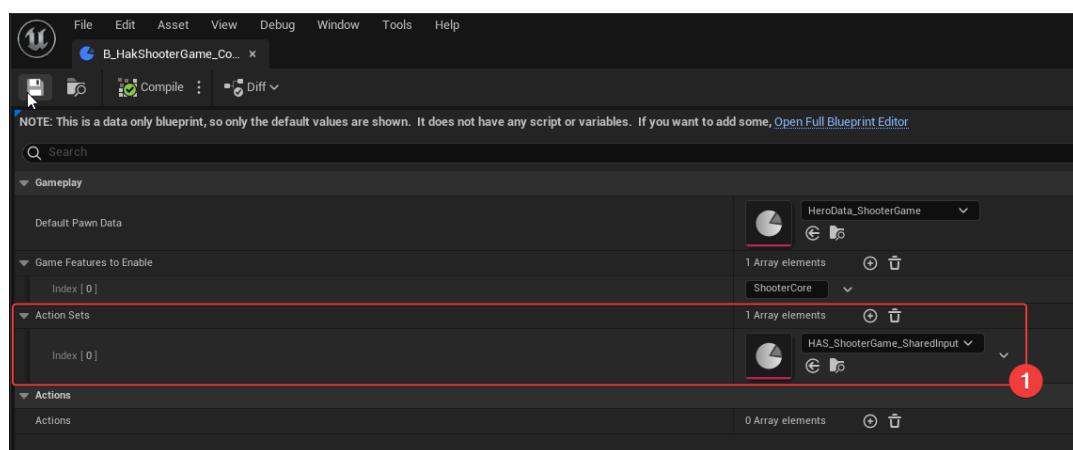
```

- ExperienceActionSet을 추가하자:

- 아래의 경로에 HAS_ShooterGame_SharedInput을 추가하자:



B_HakShooterGame_ControlPoints에 HAS_ShooterGame_SharedInput을 추가:



- InputConfig의 커스텀 GameFeatureAction 타입을 만들기에 앞서, GameFeatureAction을 활성화하는 로직부터 추가하자:
 - 앞서, GameFeature Plugin 로딩하고 활성화하는 로직을 구현하였다
 - GameFeatureAction 활성화는 이 과정 이후, 명시적으로 진행하면 된다

아래와 같이, OnExperienceFullLoadCompleted()에 추가 구현해주자:

HakExperienceManagerComponent.cpp에 헤더 파일 추가시키기:

```

#include "HakExperienceManagerComponent.h"
#include "HakExperienceDefinition.h" 1
#include "HakExperienceActionSet.h"
#include "GameFeaturesSubsystem.h"
#include "GameFeaturesSubsystemSettings.h"
#include "HakGame/System/HakAssetManager.h"
#include "Net/UnrealNetwork.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(HakExperienceManagerComponent)

```

- 마저 구현하기:

```

void UHakExperienceManagerComponent::OnExperienceFullLoadCompleted()
{
    check(LoadState != EHakExperienceLoadState::Loaded);

    // GameFeature Plugin의 토딩과 활성화 이후, GameFeature Action들을 활성화 시키자:
    {
        LoadState = EHakExperienceLoadState::ExecutingActions;

        // GameFeatureAction 활성화를 위한 Context 준비
        FGameFeatureActivatingContext Context;
        {
            // 월드의 핸들을 세팅해준다
            const FWorldContext* ExistingWorldContext = GEngine->GetWorldContextFromWorld(GetWorld());
            if (ExistingWorldContext)
            {
                Context.SetRequiredWorldContextHandle(ExistingWorldContext->ContextHandle);
            }
        }

        auto ActivateListOfActions = [&Context](const TArray<UGameFeatureAction*>& ActionList)
        {
            for (UGameFeatureAction* Action : ActionList)
            {
                // 명시적으로 GameFeatureAction에 대해 Registering -> Loading -> Activating 순으로 호출한다
                if (Action)
                {
                    Action->OnGameFeatureRegistering();
                    Action->OnGameFeatureLoading();
                    Action->OnGameFeatureActivating(Context);
                }
            }
        };

        // 1. Experience의 Actions
        ActivateListOfActions(CurrentExperience->Actions);

        // 2. Experience의 ActionSets
        for (const TObjectPtr<UHakExperienceActionSet>& ActionSet : CurrentExperience->ActionSets)
        {
            ActivateListOfActions(ActionSet->Actions);
        }
    }

    LoadState = EHakExperienceLoadState::Loaded;
    OnExperienceLoaded.Broadcast(CurrentExperience);
    OnExperienceLoaded.Clear();
}

```

GameFeatureAction__AddInputConfig

▼ 펼치기

- GameFeatureAction_WorldActionBase와 _AddInputConfig 파일 추가:

- 아래의 경로에 추가하기:

이름	수정한 날짜	유형	크기
GameFeatureAction_WorldActionBase.h	2023-11-20 오전 6:03	CPP 파일	0KB
GameFeatureAction_WorldActionBase.cpp	2023-11-20 오전 6:03	H 파일	0KB
GameFeatureAction_WorldActionBase.h	2023-11-20 오전 6:03	CPP 파일	0KB
GameFeatureAction_WorldActionBase.cpp	2023-11-20 오전 6:03	H 파일	0KB

GameFeatureAction_WorldActionBase

```
#pragma once

#include "Containers/Map.h"
#include "GameFeatureAction.h"
#include "GameFeaturesSubsystem.h"
#include "GameFeatureAction_WorldActionBase.generated.h"

class FDelegateHandle;
class UGameInstance;
struct FGameFeatureActivatingContext;
struct FGameFeatureDeactivatingContext;
struct FWorldContext;

UCLASS(Abstract)
class UGameFeatureAction_WorldActionBase : public UGameFeatureAction
{
    GENERATED_BODY()
public:
    /**
     * UGameFeatureAction's interface
     */
    virtual void OnGameFeatureActivating(FGameFeatureActivatingContext& Context) override;

    /**
     * interface
     */
    virtual void AddToWorld(const FWorldContext& WorldContext, const FGameFeatureStateChangeContext& ChangeContext) PURE_VI
};
```

```
#include "GameFeatureAction_WorldActionBase.h"
#include UE_INLINE_GENERATED_CPP_BY_NAME(GameFeatureAction_WorldActionBase)

void UGameFeatureAction_WorldActionBase::OnGameFeatureActivating(FGameFeatureActivatingContext& Context)
{
    // 월드를 순회하면서,
    for (const FWorldContext& WorldContext : GEngine->GetWorldContexts())
    {
        // 앞서, ExperienceManagerComponent에서 GameFeatureAction을 활성화하면서, Context에 World를 넣어줌
        // - 이를 통해 적용할 대상인지 판단
        if (Context.ShouldApplyToWorldContext(WorldContext))
        {
            // WorldActionBase의 Interface인 AddToWorld 호출
            AddToWorld(WorldContext, Context);
        }
    }
}
```

GameFeatureAction_AddInputConfig:

Reset() 함수 구현:

```

void UGameFeatureAction_AddInputConfig::Reset(FPerContextData& ActiveData)
{
    // ExtensionRequestHandles을 초기화
    ActiveData.ExtensionRequestHandles.Empty();

    // PawnsAddedTo에 대해서, 하나씩 Stack 방식으로 위에서 아래로 직접 InputConfig를 제거 진행
    while (!ActiveData.PawnsAddedTo.IsEmpty())
    {
        TWeakObjectPtr<APawn> PawnPtr = ActiveData.PawnsAddedTo.Top();
        if (PawnPtr.IsValid())
        {
            RemoveInputConfig(PawnPtr.Get(), ActiveData);
        }
        else
        {
            // WeakObjectPtr로 PawnsAddedTo를 관리하고 있기 때문에, GC되었다면, nullptr일 수 있음
            ActiveData.PawnsAddedTo.Pop();
        }
    }
}

```

□ AddInputConfig/RemoveInputConfig 구현:

```

void UGameFeatureAction_AddInputConfig::AddInputConfig(APawn* Pawn, FPerContextData& ActiveData)
{
    APlayerController* PlayerController = Cast<APlayerController>(Pawn->GetController());
    if (ULocalPlayer* LP = PlayerController ? PlayerController->GetLocalPlayer() : nullptr)
    {
        if (UEnhancedInputLocalPlayerSubsystem* Subsystem = LP->GetSubsystem<UEnhancedInputLocalPlayerSubsystem>())
        {
            FModifyContextOptions Options = {};
            Options.bIgnoreAllPressedKeysUntilRelease = false;

            // 추가된 InputConfigs를 순회하며, EnhancedInputSubsystem에 PlayerMappableConfig를 직접 추가
            for (const FHakMappableConfigPair& Pair : InputConfigs)
            {
                if (Pair.bShouldActivateAutomatically)
                {
                    Subsystem->AddPlayerMappableConfig(Pair.Config.LoadSynchronous(), Options);
                }
            }

            // ActiveData에 Pawn을 관리대상으로 등록
            ActiveData.PawnsAddedTo.AddUnique(Pawn);
        }
    }
}

void UGameFeatureAction_AddInputConfig::RemoveInputConfig(APawn* Pawn, FPerContextData& ActiveData)
{
    APlayerController* PlayerController = Cast<APlayerController>(Pawn->GetController());
    if (ULocalPlayer* LP = PlayerController ? PlayerController->GetLocalPlayer() : nullptr)
    {
        if (UEnhancedInputLocalPlayerSubsystem* Subsystem = LP->GetSubsystem<UEnhancedInputLocalPlayerSubsystem>())
        {
            // InputConfigs를 순회하며, Config를 제거 진행
            for (const FHakMappableConfigPair& Pair : InputConfigs)
            {
                Subsystem->RemovePlayerMappableConfig(Pair.Config.LoadSynchronous());
            }

            ActiveData.PawnsAddedTo.Remove(Pawn);
        }
    }
}

```

□ OnGameFeatureActivating()

```

void UGameFeatureAction_AddInputConfig::OnGameFeatureActivating(FGameFeatureActivatingContext& Context)
{
    FPerContextData& ActiveData = ContextData.FindOrAdd(Context);
    if (!ensure(ActiveData.ExtensionRequestHandles.IsEmpty()) ||
        !ensure(ActiveData.PawnsAddedTo.IsEmpty()))
    {
        Reset(ActiveData);
    }

    // UGameFeatureAction_WorldActionBase를 호출하면서, AddToWorld() 호출!
    Super::OnGameFeatureActivating(Context);
}

```

□ AddToWorld

□ HandlePawnExtension 구현:

```
void UGameFeatureAction_AddInputConfig::HandlePawnExtension(Actor* Actor, FName EventName, FGameFeatureStateChangeContext ChangeContext)
{
    APawn* AsPawn = CastChecked<APawn>(Actor);
    FPerContextData* ActiveData = ContextData.FindOrAdd(ChangeContext);

    if (EventName == UGameFrameworkComponentManager::NAME_ExtensionAdded)
    {
        AddInputConfig(AsPawn, ActiveData);
    }
    else if (EventName == UGameFrameworkComponentManager::NAME_ExtensionRemoved)
    {
        RemoveInputConfig(AsPawn, ActiveData);
    }
}
```

□ AddToWorld 구현:

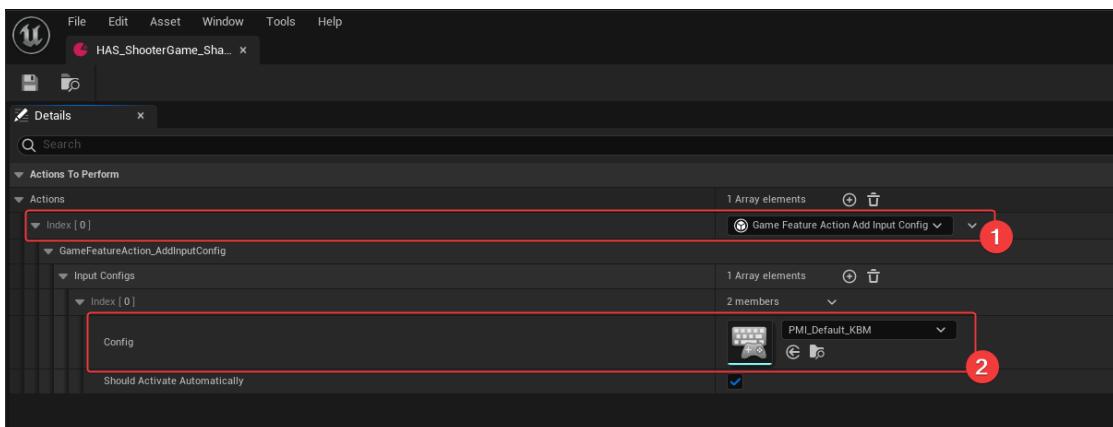
```
void UGameFeatureAction_AddInputConfig::AddToWorld(const FWorldContext& WorldContext, const FGameFeatureStateChangeContext& ChangeContext)
{
    UWorld* World = WorldContext.World();
    UGameInstance* GameInstance = WorldContext.OwningGameInstance;
    FPerContextData* ActiveData = ContextData.FindOrAdd(ChangeContext);
    if (GameInstance && World && World->IsGameWorld())
    {
        // GFCM을 이용하여, ExtensionHandler를 추가하여 등록 실행:
        // - HandlePawnExtension 풀백 함수로 연결
        if (UGameFrameworkComponentManager* ComponentMan = GameInstance->GetSubsystem<UGameFrameworkComponentManager>(GameInstance))
        {
            UGameFrameworkComponentManager::FExtensionHandlerDelegate AddConfigDelegate =
                UGameFrameworkComponentManager::FExtensionHandlerDelegate::Create UObject(this, &ThisClass::HandlePawnExtension, ChangeContext);

            // 등록된 풀백 함수의 핸들을 ActiveData의 ExtensionRequestHandles에 등록
            TSharedPtr<FComponentRequestHandle> ExtensionRequestHandle = ComponentMan->AddExtensionHandler(APawn::StaticClass(), AddConfigDelegate);
            ActiveData.ExtensionRequestHandles.Add(ExtensionRequestHandle);
        }
    }
}
```

□ OnGameFeatureDeactivating()

```
void UGameFeatureAction_AddInputConfig::OnGameFeatureDeactivating(FGameFeatureDeactivatingContext& Context)
{
    Super::OnGameFeatureDeactivating(Context);
    FPerContextData* ActiveData = ContextData.Find(Context);
    if (ensure(ActiveData))
    {
        Reset(*ActiveData);
    }
}
```

□ 앞서 정의했던 HAS_ShooterGame_SharedInput에 AddInputConfig를 추가해주자:



□ 아직 우리에게 보이는 캐릭터가 없으니, 로그를 남기도록 해보자:

- HeroComponent의 Input_Move에 로그를 임시적으로 추가하자:

```
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>();

        bool bLogging = true;
        if (bLogging)
        {
            UE_LOG(LogHak, Log, TEXT("Input_Move[X=%.2f][Y=%.2f]"), Value.X, Value.Y);
        }

        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 우리는 Forward 적용을 위해 swizzle input modifier를 사용했다~
        {
            // 앞서 Left/Right와 마찬가지로 Forward/Backward를 적용한다
            const FVector MovementDirection = MovementRotation.RotateVector(FVector::ForwardVector);
            Pawn->AddMovementInput(MovementDirection, Value.Y);
        }
    }
}
```

로그가 생성되는지 확인하자:

```
256 void UHakHeroComponent::Input_Move(const FInputActionValue& InputActionValue)
257 {
258     APawn* Pawn = GetPawn<APawn>();
259     AController* Controller = Pawn ? Pawn->GetController() : nullptr;
260
261     if (Controller)
262     {
263         FVector2D Value = InputActionValue.Get<FVector2D>();
264
265         bool bLogging = true;
266         if (bLogging)
267         {
268             UE_LOG(LogHak, Log, TEXT("Input_Move[X=%.2f][Y=%.2f]"), Value.X, Value.Y);
269         }
270
271         const FRotator MovementRotation(0.0f, Controller->GetControlRotation().Yaw, 0.0f);
272
273         if (Value.X != 0.0f)
274         {
275             Controller->SetControlRotation(FRotator(0.0f, MovementRotation.Pitch, 0.0f));
276
277             if (Value.Y != 0.0f)
278             {
279                 Controller->AddMovementInput(FVector(0.0f, Value.Y, 0.0f));
280             }
281         }
282     }
283 }
```

하나씩 순차적으로 디버깅해보자:

□ 우선 GameFeatureAction을 활성화하는 ExperienceManagerComponent부터 보자:

```

void UHakExperienceManagerComponent::OnExperienceFullLoadCompleted()
{
    check(LoadState != EHakExperienceLoadState::Loaded);

    // GameFeature Plugin의 로딩과 활성화 이후, GameFeature Action들을 활성화 시키자:
    {
        LoadState = EHakExperienceLoadState::ExecutingActions;

        // GameFeatureAction 활성화를 위한 Context 준비
        FGameFeatureActivatingContext Context;
        {
            // 월드의 핸들을 세팅해준다
            const FWorldContext* ExistingWorldContext = GEngine->GetWorldContextFromWorld(GetWorld());
            if (ExistingWorldContext)
            {
                Context.SetRequiredWorldContextHandle(ExistingWorldContext->ContextHandle);
            }
        }

        auto ActivateListOfActions = [&Context](const TArray<UGameFeatureAction*>& ActionList)
        {
            for (UGameFeatureAction* Action : ActionList)
            {
                // 명시적으로 GameFeatureAction에 대해 Registering -> Loading -> Activating 순으로 호출한다
                if (Action)
                {
                    Action->OnGameFeatureRegistering();
                    Action->OnGameFeatureLoading();
                    Action->OnGameFeatureActivating(Context);
                }
            };
        };

        // 1. Experience의 Actions
        ActivateListOfActions(CurrentExperience->Actions);

        // 2. Experience의 ActionSets
        for (const TObjectPtr<UHakExperienceActionSet>& ActionSet : CurrentExperience->ActionSets)
        {
            ActivateListOfActions(ActionSet->Actions);
        }
    }

    LoadState = EHakExperienceLoadState::Loaded;
    OnExperienceLoaded.Broadcast(CurrentExperience);
    OnExperienceLoaded.Clear();
}

```

- PRAGMA_DISABLE_OPTIMIZATION을 넣어서 디버깅에 용의하게 하자:
- OnExperienceFullLoadCompleted():

```

PRAGMA_DISABLE_OPTIMIZATION
void UHakExperienceManagerComponent::OnExperienceFullLoadCompleted()
{
    check(LoadState != EHakExperienceLoadState::Loaded);

    // GameFeature Plugin의 로딩과 활성화 이후, GameFeature Action들을 활성화 시키자:
    {
        LoadState = EHakExperienceLoadState::ExecutingActions;

        // GameFeatureAction 활성화를 위한 Context 준비
        FGameFeatureActivatingContext Context;
        {
            // 월드의 핸들을 세팅해준다
            const FWorldContext* ExistingWorldContext = GEngine->GetWorldContextFromWorld(GetWorld());
            if (ExistingWorldContext)
            {
                Context.SetRequiredWorldContextHandle(ExistingWorldContext->ContextHandle);
            }
        }

        auto ActivateListOfActions = [&Context](const TArray<UGameFeatureAction*>& ActionList)
        {
            for (UGameFeatureAction* Action : ActionList)
            {
                // 명시적으로 GameFeatureAction에 대해 Registering -> Loading -> Activating 순으로 호출한다
                if (Action)
                {
                    Action->OnGameFeatureRegistering();
                    Action->OnGameFeatureLoading();
                    Action->OnGameFeatureActivating(Context);
                }
            }
        };

        // 1. Experience의 Actions
        ActivateListOfActions(CurrentExperience->Actions);

        // 2. Experience의 ActionSets
        for (const TObjectPtr<UHakExperienceActionSet>& ActionSet : CurrentExperience->ActionSets)
        {
            ActivateListOfActions(ActionSet->Actions);
        }
    }

    LoadState = EHakExperienceLoadState::Loaded;
    OnExperienceLoaded.Broadcast(CurrentExperience);
    OnExperienceLoaded.Clear();
}
PRAGMA_ENABLE_OPTIMIZATION

```

□ UGameFeatureAction_AddInputConfig::OnGameFeatureActivating():

```

7 void UGameFeatureAction_AddInputConfig::OnGameFeatureActivating(FGameFeatureActivatingContext& Context)
8 {
9     FPerContextData& ActiveData = ContextData.FindOrAdd(Context); ≤ 2ms elapsed
10    if (!ensure(ActiveData.ExtensionRequestHandles.IsEmpty()) || 
11        !ensure(ActiveData.PawnsAddedTo.IsEmpty()))
12    {
13        Reset(ActiveData);
14    }
15
16    // UGameFeatureAction_WorldActionBase를 호출하면서, AddToWorld() 호출!
17    Super::OnGameFeatureActivating(Context);
18 }

```

□ UGameFeatureAction_WorldActionBase::OnGameFeatureActivating():

```

4 void UGameFeatureAction_WorldActionBase::OnGameFeatureActivating(FGameFeatureActivatingContext& Context)
5 {
6     // 월드를 순회하면서,
7     for (const FWorldContext& WorldContext : GEngine->GetWorldContexts()) ≤ 2ms elapsed
8     {
9         // 앞서, ExperienceManagerComponent에서 GameFeatureAction을 활성화하면서, Context에 World를 넣어줌
10        // 이를 통해 적용할 대상인지 판단
11        if (Context.ShouldApplyToWorldContext(WorldContext))
12        {
13            // WorldActionBase의 Interface인 AddToWorld 호출
14            AddToWorld(WorldContext, Context);
15        }
16    }
17 }

```

□ UGameFeatureAction_AddInputConfig::AddToWorld():

```
30  void UGameFeatureAction_AddInputConfig::AddToWorld(const FWorldContext& WorldContext, const FGameFeatureStateChangeContext& ChangeContext)
31  {
32      UWorld* World = WorldContext.World; // ≤ 4ms elapsed
33      UGameInstance* GameInstance = WorldContext.GetingGameInstance;
34      FPerContextData& ActiveData = ContextData.FindOrAdd(ChangeContext);
35      if (GameInstance && World && World->IsGameWorld)
36      {
37          // GFCM을 이용하여, ExtensionHandler를 추가하여 등록 진행:
38          // - HandlePawnExtension 플랫폼으로 연결
39          if (UGameFrameworkComponentManager* ComponentMan = GameInstance->GetSubsystem<UGameFrameworkComponentManager>(GameInstance))
40          {
41              UGameFrameworkComponentManager::FExtensionHandlerDelegate AddConfigDelegate =
42                  UGameFrameworkComponentManager::FExtensionHandlerDelegate::Create UObject(this, &ThisClass::HandlePawnExtension, ChangeContext);
43
44          // 등록된 플랫폼의 핸들을 ActiveData의 ExtensionRequestHandles에 등록
45          TSharedPtr<FComponentRequestHandle> ExtensionRequestHandle = ComponentMan->AddExtensionHandler(APawn::StaticClass(), AddConfigDelegate);
46          ActiveData.ExtensionRequestHandles.Add(ExtensionRequestHandle);
47      }
48  }
```

□ UGameFeatureAction_AddInputConfig::HandlePawnExtension():

```
115 void UGameFeatureAction_AddInputConfig::HandlePawnExtension(AActor* Actor, FName EventName, FGameFeatureStateChangeContext ChangeContext)
116 {
117     APawn* AsPawn = CastChecked<APawn>(Actor);
118     FPerContextData& ActiveData = ContextData.FindOrAdd(ChangeContext); // ≤ 1ms elapsed
119
120     if (EventName == UGameFrameworkComponentManager::NAME_ExtensionAdded)
121     {
122         AddInputConfig(AsPawn, ActiveData);
123     }
124     else if (EventName == UGameFrameworkComponentManager::NAME_ExtensionRemoved)
125     {
126         RemoveInputConfig(AsPawn, ActiveData);
127     }
128 }
```

- 우리가 원하는 Actor인 B_Hero_ShooterMannequin이 아니다...

- UGameFrameworkComponentManager::SendGameFrameworkComponentExtensionEvent()를 활용하여, 직접적으로 _AddInputConfig의 GameFeatureAction의 AddInputConfig를 활성화 해야 한다!

□ NAME_BindInputsNow 정의하자:

```
/** 
 * component that sets up input and camera handling for player controlled pawns (or bots that simulate players)
 * - this depends on a PawnExtensionComponent to coordinate initialization
 *
 * 카메라, 입력 등 플레이어가 제어하는 시스템의 초기화를 처리하는 컴포넌트
 */
UCLASS(Blueprintable, Meta=(BlueprintSpawnableComponent))
class UHakHeroComponent : public UPawnComponent, public IGameFrameworkInitStateInterface
{
    GENERATED_BODY()
public:
    UHakHeroComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /** FeatureName 정의 */
    static const FName NAME_ActorFeatureName;

    /** Extension Event 이름 정의 */
    static const FName NAME_BindInputsNow; // 1
};

/**
```

```

#include "HakHeroComponent.h"
#include "HakPawnExtensionComponent.h"
#include "EnhancedInputSubsystems.h"
#include "PlayerMappableInputConfig.h"
#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)

/** FeatureName 정의: static member variable 초기화 */
const FName UHakHeroComponent::NAME_ActorFeatureName("Hero");

/** InputConfig의 GameFeatureAction 활성화 ExtensioEvent 이름 */
const FName UHakHeroComponent::NAME_BindInputsNow("BindInputsNow");

```

□ SendGameFrameworkComponentExtensionEvent() 추가:

```

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);

    // EnhancedInputLocalPlayerSubsystem에 MappingContext을 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 FHakInputConfig* 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);

                if (HakIC)
                {
                    // InputTag_Move@ InputTag_Look_Mouse에 대해 각각 Input_Move()와 Input_LookMouse() 명령 함수에 바인딩시킨다:
                    // - 배경당한 이후, Input 이벤트에 따라 엘바 함수가 같은 키가된다
                    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);
                }
            }
        }
    }
}

// GameFeatureAction_AddInputConfig@ HandlePawnExtension 블록 함수 전달
+ UGameFrameworkComponentManager::SendGameFrameworkComponentExtensionEvent(const_cast<APawn>(Pawn), NAME_BindInputsNow);

```

□ HandlePawnExtension 설정:

```
void UGameFeatureAction_AddInputConfig::HandlePawnExtension(AActor* Actor, FName EventName, FGameFeatureStateChangeContext ChangeContext)
{
    APawn* AsPawn = CastChecked<APawn>(Actor);
    FPerContextData& ActiveData = ContextData.FindOrAdd(ChangeContext);

    if (EventName == UGameFrameworkComponentManager::NAME_ExtensionAdded || EventName == UHakHeroComponent::NAME_BindInputsNow)
    {
        AddInputConfig(AsPawn, ActiveData);
    }
    else if (EventName == UGameFrameworkComponentManager::NAME_ExtensionRemoved || EventName == UHakHeroComponent::NAME_BindInputsNow)
    {
        RemoveInputConfig(AsPawn, ActiveData);
    }
}
```

- 카메라가 잘 움직이는 것을 확인하자!