



# 6주차 (2023.10.23)

## 저번주 게시판에 나온 질문들

### ▼ 펼치기

- `UE_INLINE_GENERATED_CPP_BY_NAME`:

- 아마 제 기억으로 강의에서는 아래의 코드가 같다고 언급드렸던거 같습니다:  
(만약 제가 그렇지 않았다면, 혼란을 드린거 같습니다... 다시 정정하겠습니다~  
(\_ \_))

```
#include UE_INLINE_GENERATED_CPP_BY_NAME(Name)
#include "Name.gen.cpp"
```

- 뭔가 `UE_INLINE_GENERATED_CPP_BY_NAME()`이 컴파일 시간을 개선한다고 나와있는데 전 물음표가 있습니다
- 전 단순 `gen.cpp`를 포함시키는데 Lyra의 클론하는 입장에서 일관성 유지로 사용하고 있습니다.
- 정리하면, 전 두가지 방식에서 큰 의미 없다라고 생각합니다. 중요한건 `gen.cpp`를 포함시킨다는 것입니다.
- 제 기억으로 컴파일 타임 최적화는 아래와 같이 헤더에 #include 대신 forward declaration을 통해, 해당 헤더가 포함되는 cpp에 불필요한 #include로 추가적인 컴파일 시간을 늘리지 않음에 따라, 컴파일 시간이 최적화 됨을 알려드렸습니다:

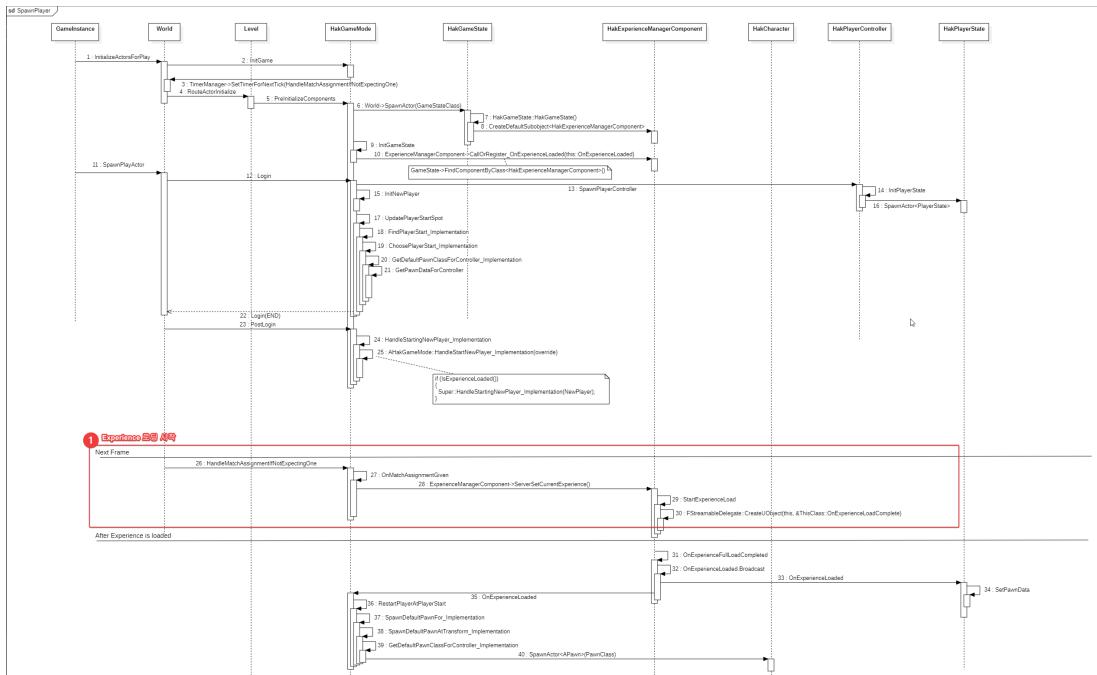
```
/** Name.h */
class NameA;
class B
{
    TObjectPtr<NameA> C;
};

/** Name.cpp */
#include "A.h" // NameA를 포함하고 있음
```

# Experience 로딩

## ▼ 펼치기

- 아래의 시퀀서를 활용하여, Experience 로딩 과정 설명:



- HandleMatchAssignmentIfNotExpectingOne 구현:

```

/*
 * member methods
 */
void AHakGameMode::HandleMatchAssignmentIfNotExpectingOne()
{
    // 해당 함수에서는 우리가 로딩할 Experience에 대해 PrimaryAssetId를 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 설정하자
    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);
}

void AHakGameMode::OnMatchAssignmentGiven(FPrimaryAssetId ExperienceId)
{
    // 해당 함수는 ExperienceManagerComponent을 활용하여 Experience을 로딩하기 위해, ExperienceManagerComponent의 ServerSetCurrentExperience를 호출한다
    check(ExperienceId.IsValid());

    UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
    check(ExperienceManagerComponent);
    ExperienceManagerComponent->ServerSetCurrentExperience(ExperienceId);
}
  
```

- 아직 직관적으로 해당 함수들의 이름으로 이해는 힘들다:

- 중요한 것은 GameMode에서 다음 프레임에 Experience 로딩을 진행하고, 로딩 완료 이후, Callback을 통해 캐릭터 생성을 처리할 것이다.

- HakExperienceComponent::ServerSetCurrentExperience 구현:
  - UGameFeaturesSubsystemSettings::LoadState[Client|Server]를 위한 GameFeatures 모듈:

```

1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 using UnrealBuildTool;
4
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       "CoreUObject",
16       "Engine",
17       "InputCore",
18       // GAS
19       "GameplayTags",
20       // Game Features
21       "ModularGameplay",
22       "GameFeatures",
23     });
24
25     PrivateDependencyModuleNames.AddRange(new string[] { });
26
27     // Uncomment if you are using Slate UI
28     // PrivateDependencyModuleNames.AddRange(new string[] { "Slate", "SlateCore" });
29
30     // Uncomment if you are using online features
31     // PrivateDependencyModuleNames.Add("OnlineSubsystem");
32
33     // To include OnlineSubsystemSteam, add it to the plugins section in your uproject file with the Enabled attribute set to true
34   }
35 }
```

- 전체적인 코드 작성 이후, 코드 디버깅 및 설명 진행
  - 전체적인 코드 작성: OnExperienceFullLoadCompleted()까지 작성

```

void UHakExperienceManagerComponent::ServerSetCurrentExperience(FPrimaryAssetId ExperienceId)
{
    UHakAssetManager& AssetManager = UHakAssetManager::Get();

    TSubclassOf<UHakExperienceDefinition> AssetClass;
    {
        FSoftObjectPath AssetPath = AssetManager.GetPrimaryAssetPath(ExperienceId);
        AssetClass = Cast<UClass>(AssetPath.TryLoad());
    }

    const UHakExperienceDefinition* Experience = GetDefault<UHakExperienceDefinition>(AssetClass);
    check(Experience != nullptr);
    check(CurrentExperience == nullptr);
    {
        CurrentExperience = Experience;
    }

    StartExperienceLoad();
}

void UHakExperienceManagerComponent::StartExperienceLoad()
{
    check(CurrentExperience);
    check(LoadState == EHakExperienceLoadState::Unloaded);

    LoadState = EHakExperienceLoadState::Loading;

    UHakAssetManager& AssetManager = UHakAssetManager::Get();

    TSet<FPrimaryAssetId> BundleAssetList;
    BundleAssetList.Add(CurrentExperience->GetPrimaryAssetId());

    // load assets associated with the experience
    TArray< FName > BundlesToLoad;
    {
        const ENetMode OwnerNetMode = GetOwner()->GetNetMode();
        bool bLoadClient = GIsEditor || (OwnerNetMode != NM_DedicatedServer);
        bool bLoadServer = GIsEditor || (OwnerNetMode != NM_Client);
        if (bLoadClient)
        {
            BundlesToLoad.Add(UGameFeaturesSubsystemSettings::LoadStateClient);
        }
        if (bLoadServer)
        {
            BundlesToLoad.Add(UGameFeaturesSubsystemSettings::LoadStateServer);
        }
    }

    FStreamableDelegate OnAssetsLoadedDelegate = FStreamableDelegate::CreateUObject(this, &ThisClass::OnExperienceLoadComplete);
}

void UHakExperienceManagerComponent::OnExperienceLoadComplete()
{
    OnExperienceFullLoadCompleted();
}

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

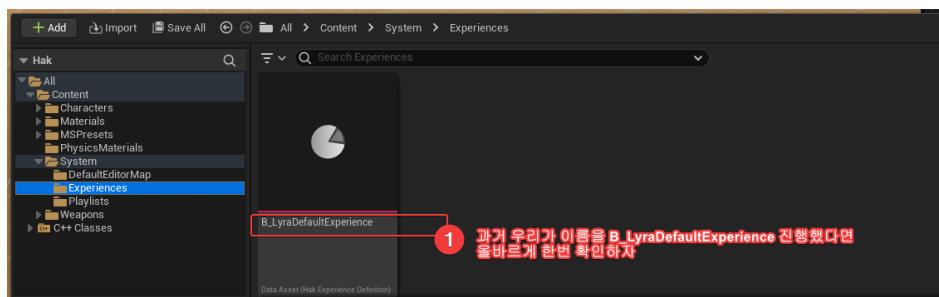
    LoadState = EHakExperienceLoadState::Loaded;
    OnExperienceLoaded.Broadcast(CurrentExperience);
    OnExperienceLoaded.Clear();
}

```

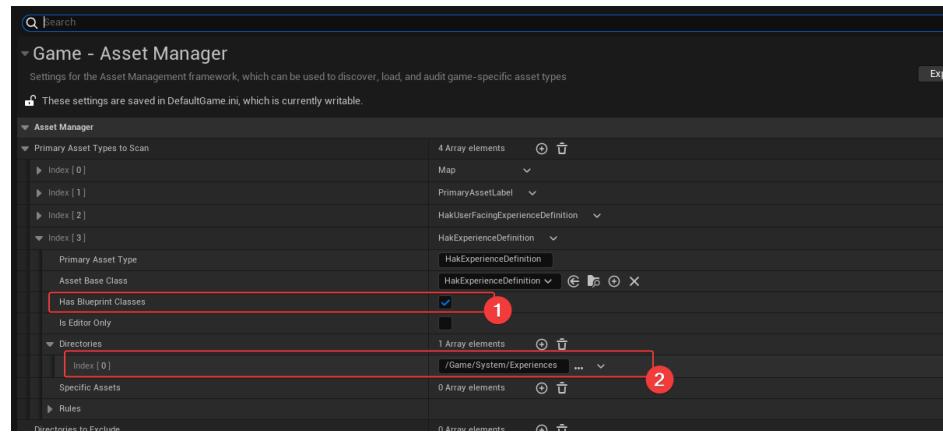
## □ 디버깅을 위한 준비:

- 혹시 B\_HakDefaultExperience가 준비되었는지 확인하자:

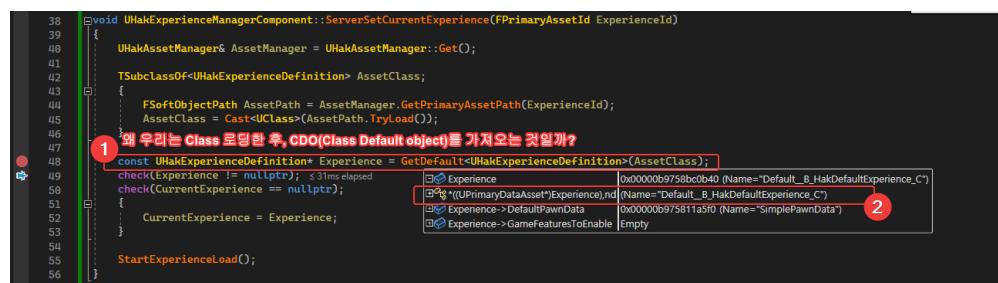
Game/System/Experiences



- 그리고 Project Settings에서 HakUserDefinition을 한번 확인해보자:



- 디버깅을 하며, ServerSetCurrentExperience 함수를 우선 보자:



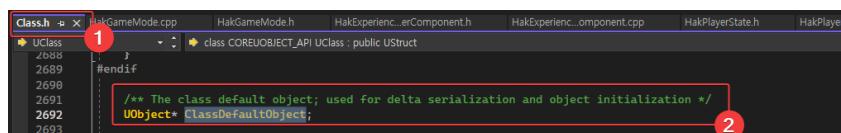
- `Class Default Object`에 대한 설명:

- 쉽게, Blueprint의 default constructor라고 보면 된다!:
  - 정확하게 말하면, Class의 초기화 값으로 생성된 UObject!
- BP는 C++가 아닌 스크립트 → Default Constructor의 동작을 만들기 위해, Class Default Object를 도입하여, 그 동작을 구현하고 있다.
- 사실 `Class Default Object`는 UClass의 초기화 값으로 이루어진 UObject로 객체화 된 인스턴스이다.

- 간단히 디버깅하며, CDO를 살펴보자:

Name	Value	Type
AssetClass	{Class=0x00000b9758252a00 (Name="B_HakDefaultExperience_C") }	TSubclassOf<UHakE...
Class	0x00000b9758252a00 (Name="B_HakDefaultExperience_C")	UClass * (UnrealEdit...
UObject	(Name="B_HakDefaultExperience_C")	UnrealEditor-Engine...
UObject	0x00007ff93fb42d10 (UnrealEditor-HakGame.dll!InternalConstruct...	UObject (*) (FVTable...
ClassConstructor	0x00007ff93fb42d50 (UnrealEditor-HakGame.dll!InternalTableHelp...	FUObjectCppClassSt...
ClassTableHelperCtorCaller	(AddReferencedObjects=0x00007ff9b20fcae0 {UnrealEditor-CoreUO...	int
CppClassStaticFunctions	0	int
ClassUnique	0	bool
FirstOwnedClassRep	false	bool
bCooked	CLASS_ReplicationDatalSetup   CLASS_CompiledFromBlueprint   CL...	ECClassFlags
bLayoutChanging	CASTCLASS_None (0)	ECClassCastFlags
ClassFlags	0x00000b96fea4e00 (Name="Object")	UClass * (UnrealEdit...
ClassCastFlags	0x00000b971c3eea00 (Name="B_HakDefaultExperience")	UObject {*} (UnrealEd...
ClassWithin	0x0000000000000000 <NULL>	FField *
ClassGeneratedBy	"Engine"	FName
PropertiesPendingDestruction	Empty	TArray<FRepRecord,...
ClassConfigName	Empty	TArray<UField * TSiz...
ClassReps	Empty	
NetFields	Empty	
ClassDefaultObject	0x00000b9758bc0b40 (Name="Default_B_HakDefaultExperience_C")	UObject {*} (UHakExp...
UObject	(Name="Default_B_HakDefaultExperience_C")	UHakExperienceDefi...
UObjectBaseUtility	0x0000000000000000	UObjectBaseUtility
SparseClassData	id *	UObject
SparseClassDataStruct	0x0000000000000000	UObject
CppClassTypeInfo	<NULL>	UObject
FuncMap	Set: {{Info=0x00007ff9b255477c {UnrealEditor-CoreUObject.dll!FCpp...}}	UObject
SuperFuncMap	Empty	UObject
SuperFuncMapLock	Empty	UObject
Interfaces	Empty	UObject
ReferenceTokenStream	<Tokens=Num=10 StackSize=1 TokenType=NonNative (1) ...>	UObject
ReferenceTokenStreamCritical	CriticalSection={Opaque1=0x00000b9758252c70 {0xffffffffffff...}}...	UObject
NativeFunctionLookupTable	Empty	UObject

- 헤더를 보면 아래와 같다:



- 코드의 주석에서 또 다른 CDO의 특성을 알 수 있는데:

- Serialization 과정에서 CDO를 기반으로 Delta Serialization을 지원하기 위한 목적도 있다:

- 파일 혹은 네트워크를 통한 Delta 압축을 통한 메모리/네트워크 대역폭 이점이 있을 것이다.

□ 한 가지 더!

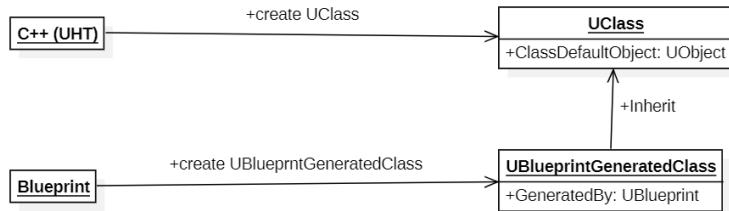
□ Blueprint 란 무엇일까?

- Blueprint는 사실 Class 생성기(Generator)이다:

- 좀 더 프로그래머스럽게, Unreal의 Script Object라고 볼 수 있다:

- Script Object로서, 클래스 정의와 클래스 멤버 함수/이벤트 함수 정의를 담당한다

- Class 생성에는 아래와 같이 두 가지 방법이 있다:



## 1. C++로 정의

- 정확히는 UHT(Unreal Header Tool)로 생성된 UClass!

## 2. Blueprint로 정의

- Blueprint로 생성된 Class는 UBlueprintGeneratedClass로 알 수 있다

□ AssetClass를 통해 보면 (B\_HakExperienceDefinition):

Name	Type
AssetClass	TSubclassOf<UHakE...
Class	UClass * {UnrealEdit...
[UBlueprintGeneratedClass]	UnrealEditor-Engine...
(Name="B_HakDefaultExperience_C")	UObject
1 UStruct	void(*) (const Objec...
CppClassStaticFunctions	FUObjectCppClassSt...
ClassUnique	int
FirstOwnedClassRep	int
bCooked	bool
bLayoutChanging	bool
ClassFlags	EClassFlags
ClassCastFlags	EClassCastFlags
ClassWithin	UClass * {UnrealEdit...
2 ClassGeneratedBy	UObject * {UnrealEdi...
(Name="B_HakDefaultExperience")	UnrealEditor-Engine...
(Name="B_HakDefaultExperience")	UnrealEditor-Engine...
{...}	UnrealEditor-Engine...
ParentClass	UnrealEditor-Engine...
BlueprintType	BPType_Normal (0)
bRecompileOnLoad	unsigned char
bHasBeenRegenerated	unsigned char
bIsRegeneratingOnLoad	unsigned char



혹시나 싶어, 이야기 드리지만! 이런건 외우기 보다 여러번 읽어보고, 이런게 있어지! 만으로 충분하다! (여러번 읽고 기억해두고 그때마다 끄집어내며, 머리에 자연스럽게 녹아들도록 하자!)

이어서 보면, 앞서 그림에서 언급했듯이 왜 Class Default Object로 로딩하는건지 의문으로 생각해봐야 한다:

- 왜 그런거지?...

- 일단 왜라는 의문점만 풀고, 코드를 표면적으로 읽어보자:

```

void UHakExperienceManagerComponent::ServerSetCurrentExperience(FPrimaryAssetId ExperienceId)
{
    UHakAssetManager& AssetManager = UHakAssetManager::Get();
    TSubclassOf<UHakExperienceDefinition> AssetClass;
    {
        FSoftObjectPath AssetPath = AssetManager.GetPrimaryAssetPath(ExperienceId);
        AssetClass = Cast<UClass>(AssetPath.TryLoad());
    }

    // 왜 CDO를 가져오는 걸까?
    const UHakExperienceDefinition* Experience = GetDefault<UHakExperienceDefinition>(AssetClass);
    check(Experience != nullptr);
    check(CurrentExperience == nullptr);
    {
        // 그리고 CDO로 CurrentExperience를 설정한다!
        // 어떤 의도로 이렇게 코드를 작성한지는 코드를 쭉 읽어보고(StartExperienceLoad까지 읽어보자) 다시 생각해보자.
        CurrentExperience = Experience;
    }

    StartExperienceLoad();
}

```

## □ 이어서, StartExperienceLoad를 읽어보자:

```

void UHakExperienceManagerComponent::StartExperienceLoad()
{
    check(CurrentExperience);
    check(LoadState == EHakExperienceLoadState::Unloaded);

    LoadState = EHakExperienceLoadState::Loading;

    UHakAssetManager& AssetManager = UHakAssetManager::Get();

    TSet<FPrimaryAssetId> BundleAssetList;
    // 이미 앞서, ServerSetCurrentExperience에서 우리는 ExperienceId를 넘겨주었는데, 여기서 CDO를 활용하여, GetPrimaryAssetId를 로딩할 대상으로 넣는다!
    // – 왜 이렇게 하는 걸까?
    // – GetPrimaryAssetId를 좀 더 자세히보자;
    BundleAssetList.Add(CurrentExperience->GetPrimaryAssetId()); ①

    // load assets associated with the experience
    TArray< FName > BundlesToLoad;
    {
        const ENetMode OwnerNetMode = GetOwner()->GetNetMode();
        bool bLoadClient = GIsEditor || (OwnerNetMode != NM_DedicatedServer);
        bool bLoadServer = GIsEditor || (OwnerNetMode != NM_Client);
        if (bLoadClient)
        {
            BundlesToLoad.Add(UGameFeaturesSubsystemSettings::LoadStateClient);
        }
        if (bLoadServer)
        {
            BundlesToLoad.Add(UGameFeaturesSubsystemSettings::LoadStateServer);
        }
    }

    FStreamableDelegate OnAssetsLoadedDelegate = FStreamableDelegate::CreateUObject(this, &ThisClass::OnExperienceLoadComplete);
}

```

## □ GetPrimaryAssetId()를 읽어보면:

```

/**
 * A DataAsset that implements GetPrimaryAssetId and has asset bundle support, which allows it to be manually loaded/unloaded from the AssetManager.
 * Instances of native subclasses can be created directly as Data Assets and will use the name of the native class as the PrimaryAssetType.
 * Or, blueprint subclasses can be created with a PrimaryAssetType equal to the name of the first native class going up the hierarchy, or the highest level blueprint class.
 * IE, if you have [UPrimaryDataAsset -> UParentNativeClass -> UChildNativeClass -> UDataOnlyBlueprintClass] the type will be ChildNativeClass.
 * Whereas if you have [UPrimaryDataAsset -> ParentBlueprintClass -> DataOnlyBlueprintClass] the type will be ParentBlueprintClass.
 * To change this behavior, override GetPrimaryAssetId in your native class or copy those functions into a different native base class.
 */

/**
 * UPrimaryDataAsset에 대한 설명이 위의 주석에 남겨져 있다. 정리하면 아래와 같다 (GetPrimaryAssetId()을 보고 정리하려면 – 사실 위의 코멘트로 아래의 내용 정리는 불가능하다 **):
 * – UPrimaryDataAsset은 GetPrimaryAssetId() 구현과 Asset Bundles 대상 구현이 주기된 DataAsset이다.
 * – 해당 기능은 AssetManager를 통한 load/unload 관련 메서드로 구현이 가능하다.
 * – GetPrimaryAssetId()를 통해 자신의 두가지 대상 구현이 가능하다:
 *   1. UPrimaryAssetID를 통해 (Native Class)로 접근 가능하다.
 *   2. UPrimaryDataAsset을 통해 (Blueprint Class)로 접근 가능하다.
 * – 또는 (Parent Class)를 거슬러 올라가며 가장 처음으로 만난 Native C++ 클래스
 * – 또는 가장 부모인 (the most ancestor) BP 클래스 (여기 해당 BP 클래스는 UPrimaryDataAsset을 바로 상속 받았을거임)
 * – 아래 같은 등장 방식을 바꾸고 싶다면, 이를 override하면 된다
 */
UCLASS(Abstract, MinimalAPI, Blueprintable)
class UPrimaryDataAsset : public UDataAsset
{
    GENERATED_BODY()

public:
    // UObject interface
    ENGINE_API virtual FPrimaryAssetId GetPrimaryAssetId() const override;
    ENGINE_API virtual void PostLoad() override;
    #if WITH_EDITORONLY_DATA
    /** This scans the class for AssetBundles metadata on asset properties and initializes the AssetBundleData with InitializeAssetBundlesFromMetadata */
    ENGINE_API virtual void UpdateAssetBundleData();
    #endif
    /** Updates AssetBundleData */
    #pragma disable_WARNING_C9036 // Suppress compiler warning on override of deprecated function
    #pragma disable_DEPRECATED_WARNINGS // Use version that takes FObjectPreSaveContext instead.
    ENGINE_API virtual void PreSave(const class ITargetPlatform* TargetPlatform) override;
    #pragma enable_DEPRECATED_WARNINGS
    ENGINE_API virtual void PreSave(FObjectPreSaveContext ObjectSaveContext) override;

protected:
    /** Asset Bundle data computed at save time. In cooked builds this is accessible from AssetRegistry */
    UPROPERTY()
    FAssetBundleData AssetBundleData;
    #endif
};

```

```

#pragma_DISABLE_OPTIMIZATION
#include "UPrimaryAsset.h"
#include "UPrimaryAssetPrivate.h"

UPrimaryAsset UPrimaryDataSet::SetPrimaryAssetId() const
{
    // 우선 해당 함수를 위하여 이전에, UPrimaryDataSet의 클래스를 한번 보자:
    // - 보았다면, 해당 로직이 그와 똑같이 구현되어 있는지 코드를 보면 확인해보자.

    // 우선 해당 클래스가 CDO이며, 이는 코드를 봐서는 그 특별한 특성을 알기 어렵다.
    // 즉, CDO이며, 역시도 HalExperienceDefinition을 Class를 통해 CDO를 설정한 이유가 여기에 있다!**
    if (HasAnyFlags(RF_ClassDefaultObject))
    {
        // BestPrimaryAssetTypeClass 결과 값은 개별작성
        UClass* BestPrimaryAssetTypeClass = nullptr;
        UClass* SearchPrimaryAssetTypeClass = GetClass();
        while (SearchPrimaryAssetTypeClass == GetSuperClass())
        {
            // If this is a native class or immediate child of PrimaryDataSet, return invalid as we are a type ourselves
            /**
             * 해당 클래스가 UPrimaryDataSet을 상속받은 C++ 클래스라면, 그가 default로 UPrimaryAsset로 리턴한다:
             * - 이는 UPrimaryAsset가 그 클래스를 상속 받은 구체적인 클래스가 필요하다는 것이다:
             *     - UPrimaryAsset --> NativeClass --> NativeClassA_Instance(여기서 this여야 함)
             */
            if (SearchPrimaryAssetTypeClass->HasAnyClassFlags(CLASS_Native | CLASS_Intrinsic) || SearchPrimaryAssetTypeClass == UPrimaryAsset::StaticClass())
            {
                BestPrimaryAssetTypeClass = UPrimaryAsset::StaticClass();
            }
            else if (SearchPrimaryAssetTypeClass->HasAnyClassFlags(CLASS_Native | CLASS_Intrinsic))
            {
                // Our parent is the first native class found that
                BestPrimaryAssetTypeClass = SearchPrimaryAssetTypeClass;
                break;
            }
            else
            {
                SearchPrimaryAssetTypeClass = SearchPrimaryAssetTypeClass->GetSuperClass();
            }
        }

        // BestPrimaryAssetTypeClass가 찾았다면
        // - 찾은 클래스 --> UPrimaryAsset으로 업캐스팅(여기선 이정표는 마지막에 있다!)
        if (BestPrimaryAssetTypeClass)
        {
            // If this is a native class and the one we're looking for, use the package name as it will be missing .
            if (SearchPrimaryAssetTypeClass->HasAnyClassFlags(CLASS_Native | CLASS_Intrinsic))
            {
                // Blueprint으로 .C 파일 때기 위해, GetName()을 통해 이름을 가져온다.
                FName PrimaryAssetType = BestPrimaryAssetTypeClass->HasAnyClassFlags(CLASS_Native | CLASS_Intrinsic) ? BestPrimaryAssetTypeClass->GetName() : FPackageName::GetShortName(BestPrimaryAssetTypeClass->GetOutermost())->GetName();
                return FPrimaryAssetId(PrimaryAssetType, FPackageName::GetShortName(GetOutermost())->GetName());
            }
        }
    }

    // We could never find a class, return invalid
    return FPrimaryAssetId();
}

// CDO가 아니라면, 그냥 바로 클래스 이름으로 PrimaryAssetType을 선택하고, 예전의 이름은 GetName()으로 받는다
if (GetClass()->GetName() == UPrimaryDataSet::GetPrimaryAssetId())
{
    return FPrimaryAssetId(GetClass()->GetName(), GetName());
}

```

## □ Intrinsic이란?

```

FPPrimaryAssetId UPrimaryDataSet::GetPrimaryAssetId() const
{
    // 우선 해당 함수를 위하여 이전에, UPrimaryDataSet의 클래스를 한번 보자:
    // - 보았다면, 해당 로직이 그와 똑같이 구현되어 있는지 코드를 보면 확인해보자.

    // 우선 해당 클래스가 CDO이며, 이는 조건이 붙는다!
    // 즉, CDO이며, **앞서 보았던 HalExperienceDefinition을 Class를 통해 CDO를 설정한 이유가 여기에 있다!**
    if (HasAnyFlags(RF_ClassDefaultObject))
    {
        // BestPrimaryAssetTypeClass 결과 값과 같 개성이겠지?
        UClass* BestPrimaryAssetTypeClass = nullptr;

        // 우선 기본적으로 한단계 상위 Parent부터 시작하네
        UClass* SearchPrimaryAssetTypeClass = GetClass()->GetSuperClass();

        // If this is a native class or immediate child of PrimaryDataSet, return invalid as we are a type ourselves
        /**
         * 해당 클래스가 UPrimaryDataSet을 상속받은 C++ 클래스라면, 원가 이상하니 그냥 default로 FPrimaryAssetId로 리턴한다:
         * - 이는 UPrimaryDataSet을 상속받은 그 클래스를 상속 받은 구체적인 클래스가 필요하다는 것이다:
         *     - UPrimaryDataSet --> NativeClassA --> NativeClassA_Instance(여기서 this여야 함)
         */
        if (GetClass()->HasAnyClassFlags(CLASS_Native | CLASS_Intrinsic) || SearchPrimaryAssetTypeClass == UPrimaryDataSet::StaticClass())
        {
            return FPrimaryAssetId();
        }

        // CLASS_Intrinsic 확인해보자
        bool bHasIntrinsic = UObjectProperty::StaticClass()->HasAnyClassFlags(CLASS_Intrinsic);
        if (bHasIntrinsic)
        {
            // Searching with parent, look up the hierarchy for a class that is either native, or a blueprint class immediately below
            // 여기가 앞서 보았던 로직에 대한 흔적
            while (SearchPrimaryAssetTypeClass) ≤ 13ms elapsed
            {
                if (SearchPrimaryAssetTypeClass->GetSuperClass() == UPrimaryDataSet::StaticClass())

```

- 우리는 앞서, 이제! GetDefault를 활용하여, CDO를 가져온 이유를 알 수 있다:
  - GetPrimaryAssetId()를 의도대로 작동시키기 위함이다
  - 의도는 UPrimaryDataSet을 상속받는 클래스를 찾기 위해!



조교분들께서는 엔진 관련 주석이 있다면 해당 파일도 같이 제공해주시길 바랍니다~ (혹시 까먹었다면 이 글을 읽는 여러분들이 요청해주셔도 될 것 같다!)



필자도 생각해보면, 이러한 왜 그런거지는 경험을 통한 시니어의 시야인거 같다. 여러분들은 저렇게 항상 생각해봐야지라고 다짐을 하되, 본 강의를 통해 필자의 인사이트(Insight)를 통한 왜를 한번 더 생각해보는 것으로 충분한거 같다~ 😊

조급해하지 말자, 경험과 지속적인 공부를 통해 여러분들도 이런 날이 올 것이라 생각한다!

## □ 다음 코드를 같이 읽어보자:

```

void UHakExperienceManagerComponent::StartExperienceLoad()
{
    check(CurrentExperience);
    check(LoadState == EHakExperienceLoadState::Unloaded);

    LoadState = EHakExperienceLoadState::Loading;

    UHakAssetManager& AssetManager = UHakAssetManager::Get();

    TSet<FPrimaryAssetId> BundleAssetList;

    // 이미 앞서 ServerSetCurrentExperience에서 우리는 ExperienceId를 넘겨주었는데, 여기서 CDO를 활용하여, GetPrimaryAssetId를 로딩할 대상으로 넣는다!
    // - 왜 이렇게 하는걸까?
    // - GetPrimaryAssetId를 좀 더 자세히보자:
    // - GetPrimaryAssetId를 살펴봄으로써, 아래의 두가지를 알 수 있다:
    //   1. 우리는 B_HakDefaultExperience를 BP로 만든 이유
    //   2. CDO를 가져와서, GetPrimaryAssetId를 호출한 이유
    // 우리는 앞서 이미 CDO로 로딩하여, CDO를 사용하지 않고 CDO를 사용하여 로딩할 애셋을 지정하여, BundleAssetList에 추가해준다!
    BundleAssetList.Add(CurrentExperience->GetPrimaryAssetId());

    // load assets associated with the experience
    // 아래는 우리가 후일 GameFeature를 사용하여, Experience에 바인딩된 GameFeature Plugin을 로딩할 Bundle 이름을 추가한다:
    // - Bundle이라는게 후일 우리가 로딩할 애셋의 카테고리 이름이라고 생각하면 된다 (앞단 '지금은 남아가자 후일, 또 다른 것이다!')
    TArray< FName > BundlesToLoad;
    {
        // 여기서 조작해야 할 부분은 OwnerNetMode가 NM_Standalone이면? Client/Server 둘다 로딩에 추가된다!
        const ENetMode OwnerNetMode = GetOwner()->GetNetMode();
        bool bLoadClient = GISeditor || (OwnerNetMode != NM_DedicatedServer);
        bool bLoadServer = GISeditor || (OwnerNetMode != NM_Client);
        if (bLoadClient)
        {
            BundlesToLoad.Add(UGameFeaturesSubsystemSettings::LoadStateClient);
        }
        if (bLoadServer)
        {
            BundlesToLoad.Add(UGameFeaturesSubsystemSettings::LoadStateServer);
        }
    }

    FStreamableDelegate OnAssetsLoadedDelegate = FStreamableDelegate::CreateUObject(this, &ThisClass::OnExperienceLoadComplete);

    // 아래도, 후일 Bundle을 우리가 GameFeature에 연동하면서 더 깊게 알아보기로 하고, 지금은 앞서 B_HakDefaultExperience를 로딩해주는 함수로 생각하자
    TSharedPtr< FStreamableHandle > Handle = AssetManager.ChangeBundleStateForPrimaryAssets(
        BundleAssetList.Array(),
        BundlesToLoad,
        {}, false, FStreamableDelegate(), FStreamableManager::AsyncLoadHighPriority);

    if (!Handle.IsValid() || Handle->HasLoadCompleted())
    {
        // 로딩이 완료되었으면, ExecuteDelegate를 통해 OnAssetsLoadedDelegate를 호출하자:
        // 아래의 함수를 확인해보자.
        FStreamableHandle::ExecuteDelegate(OnAssetsLoadedDelegate);
    }
    else
    {
        Handle->BindCompleteDelegate(OnAssetsLoadedDelegate);
        Handle->BindCancelDelegate(FStreamableDelegate::CreateLambda([OnAssetsLoadedDelegate]
        {
            OnAssetsLoadedDelegate.ExecuteIfBound();
        }));
    }
}

```

① 해당 코드를 좀 보도록 하자

- GStreamableDelegateDelayFrames != 0

```

void FStreamableHandle::ExecuteDelegate(const FStreamableDelegate& Delegate, TSharedPtr<FStreamableHandle> AssociatedHandle, const FStreamableDelegate& CancelDelegate)
{
    if (Delegate.IsBound())
    {
        // 해당 함수의 GStreamableDelegateDelayFrames는 현재 1로 세팅되어 있으므로, 바로 Delegate를 호출하지 않는다
        if (GStreamableDelegateDelayFrames == 0)
        {
            // Execute it immediately
            Delegate.Execute();
        }
        else
        {
            // Add to execution queue for next tick
            if (!GStreamableDelegateDelayHelper)
            {
                StreamableDelegateDelayHelper = new FStreamableDelegateDelayHelper();
            }

            // 여기에 Delegate를 추가하여, 풀링 호출한다
            // 그리고 FStreamableDelegateDelayHelper는 FTickableGameObject를 상속받아 매틱 불리는 핵심이다
            // 따라서 해당 Delegate는 다른 프레임 혹은 최대 프레임 마지막에 불릴 예정이다. (World Tick 이후)
            StreamableDelegateDelayHelper->AddDelegate(Delegate, CancelDelegate, AssociatedHandle);
        }
    }
}

```

- StreamableDelegateDelayHelper 를 주목해서 보자

□ GFrameNumber 를 주목해서 OnExperienceLoadComplete를 보자:

- 우선 StartExperienceLoad는 163
- OnExperienceLoadComplete도 똑같이 163

즉, StreamableDelegateDelayHelper는 같은 프레임에 World의 Tick 이후에 불림을 확인할 수 있다:

Call Stack
Search (Ctrl+E)
Name
1 UnrealEditor-HakGame.dll!UHakExperienceManagerComponent::OnExperienceLoadComplete() Line 134
[Inline Frame] UnrealEditor-HakGame.dll!Invoke(void(UHakExperienceManagerComponent::*)()) Line 66
[Inline Frame] UnrealEditor-HakGame.dll!UE::Core::Private::Tuple<TTupleBase<TIntegerSequence<unsigned int>>::ApplyAfter(void(UHakExperienceManagerComponent::*)())
UnrealEditor-HakGame.dll!TBaseUObjectMethodDelegateInstance<UHakExperienceManagerComponent,void __cdecl(void),FDefaultDelegateUserPolicy>::ExecuteIfSafe()
[Inline Frame] UnrealEditor-Engine.dll!ITDelegate::void __cdecl(void),FDefaultDelegateUserPolicy>::ExecuteIfBound() Line 639
UnrealEditor-Engine.dll!FStreamableDelegateDelayHelper::Tick(float DeltaTime) Line 157
2 UnrealEditor-Engine.dll!FTickableGameObject::Tick(float DeltaSeconds, bool bIsPaused, float DeltaSeconds) Line 153
UnrealEditor-UnrealEd.dll!UEditorEngine::Tick(float DeltaSeconds, bool bIdleMode) Line 1943
UnrealEditor-UnrealEd.dll!UEditorEngine::Tick(float DeltaSeconds, bool bIdleMode) Line 517
UnrealEditor-exe!EngineLoop::Tick() Line 5369
[Inline Frame] UnrealEditor-exe!EngineTick() Line 66
UnrealEditor-exe!GuardedMain(const wchar_t * CmdLine) Line 202
UnrealEditor-exe!LaunchWindowsStartup(HINSTANCE__ * hInstance, HINSTANCE__ * hPrevInstance, char * __formal, int nCmdShow, const wchar_t * CmdLine) Line 233
UnrealEditor-exe!WinMain(HINSTANCE__ * hInstance, HINSTANCE__ * hPrevInstance, char * pCmdLine, int nCmdShow) Line 282
[External Code]

- UEditorEngine::Tick()을 보면:

```

    // Update the level.
    {
        // So that hierarchical stats work in PIE
        SCOPE_CYCLE_COUNTER(STAT_FrameTime);

        FKismetDebugUtilities::NotifyDebuggerOfStartOfGameFrame(PieContext.World());
        1 // tick the level
        PieContext.World()->Tick( LEVELTICK_All, TickDeltaSeconds );
    }

    #if WITH_EDITOR
    #endiff
    bWorldTicked = true;
    TickType = LEVELTICK_All;

    // Block on async loading if requested.
    if (PlayWorld->bRequestedBlockOnAsyncLoading)
    {
        BlockKillLevelStreamingCompleted(PlayWorld);
        PlayWorld->bRequestedBlockOnAsyncLoading = false;
    }

    if (!bFirstTick)
    {
        // Update sky light first because sky diffuse will be visible in reflection capture indirect specular
        USkyLightComponent::UpdateSkyCaptureContents(PlayWorld);
        UReflectionCaptureComponent::UpdateReflectionCaptureContents(PlayWorld, nullptr, false, false, bInsideTick);
    }

    FKismetDebugUtilities::NotifyDebuggerOfEndOfGameFrame(PieContext.World());
}

// Tick the viewports.
if (GameViewport != NULL)
{
    GameViewport->Tick(TickDeltaSeconds);
    bHasPIEViewport = true;
}

// Pop the world
RestoreEditorWorld( OldWorld );
}

#elseif IUE_SERVER
// If we do not tick anything of the above, we need to still make sure to issue the pre-tick to MediaModule
// (unless the Editor context had a movie sequence tick active and hence already did so)
if ((MediaModule != nullptr) && !bMediaModulePreEngineTickDone)
{
    MediaModule->TickPreEngine();
}
#endif */

if (bWorldTicked)
{
2 여기서 FTickableGameObject를 상속받은 객체들이 Tick을 한다.
- 월드 World->Tick() 이 호출되는 것을 확인할 수 있다
    FTickableGameObject::TickObjects(nullptr, TickType, false, DeltaSeconds);
}

```

- 이어서, OnExperienceFullLoadComplete()를 보면, 아래와 같이 앞서 우리가 OnExperienceLoaded에 Bind했던 두개의 Delegate를 호출함을 볼 수 있다:

```

136
137 void UHakExperienceManagerComponent::OnExperienceFullLoadCompleted()
138 {
139     check(LoadState != EHakExperienceLoadState::Loaded);
140
141     1 LoadState = EHakExperienceLoadState::Loaded; _1ms elapsed
142     OnExperienceLoaded.Broadcast(CurrentExperience);
143     UnexperienceLoaded.Clear();
144 }

```

2 OnExperienceLoaded Num=2
 

- OnExperienceLoaded[0] 0x000008670187e600 (Name="HakGameMode"\_0)
- OnExperienceLoaded[1] 0x00000866fcfc38a00 (Name="HakPlayerState"\_0)

- 참고로 저 Multicast의 Delegate의 호출 순서는 어떻게 될까?

- 역순이다:

```

template<typename DelegateInstanceInterfaceType, typename DelegateBaseType, typename... ParamTypes>
void Broadcast(ParamTypes... Params) const
{
    bool NeedsCompaction = false;

    LockInvocationList();
    {
        const InvocationListType& LocalInvocationList = GetInvocationList();

1     // call bound functions in reverse order, so we ignore any instances that may be added by callees
for (int32 InvocationListIndex = LocalInvocationList.Num() - 1; InvocationListIndex >= 0; --InvocationListIndex)
    {
        // this down-cast is OK! allows for managing invocation list in the base class without requiring virtual functions
        const DelegateBaseType& DelegateBase = (const DelegateBaseType&)LocalInvocationList[InvocationListIndex];

        IDelegateInstance* DelegateInstanceInterface = GetDelegateInstanceProtectedHelper(DelegateBase);
        if (DelegateInstanceInterface == nullptr || !((DelegateInstanceInterfaceType*)DelegateInstanceInterface)->ExecuteIfSafe(Params...))
        {
            NeedsCompaction = true;
        }
    }

    UnlockInvocationList();
}

if (NeedsCompaction)
{
    const_cast<TMulticastDelegateBase*>(this)->CompactInvocationList();
}

```

- 따라서, PlayerState → GameMode 순서로 호출된다!

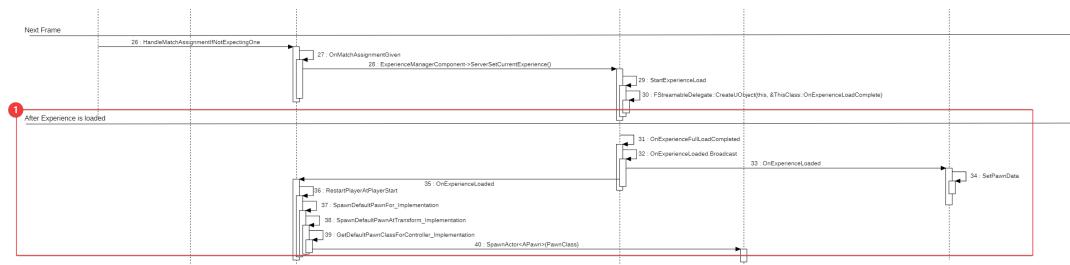
□ Wrap-up:

- 우리는 Experience 로딩 과정을 통해 아래의 사실들을 알 수 있게 되었다:
  - UPrimaryDataAsset의 정의와 역할
  - B\_HakDefaultExperience가 Blueprint로 만들어진 이유
  - Blueprint, Class Default Object, BlueprintGeneratedClass/Class 차이

## OnExperienceLoaded 구현

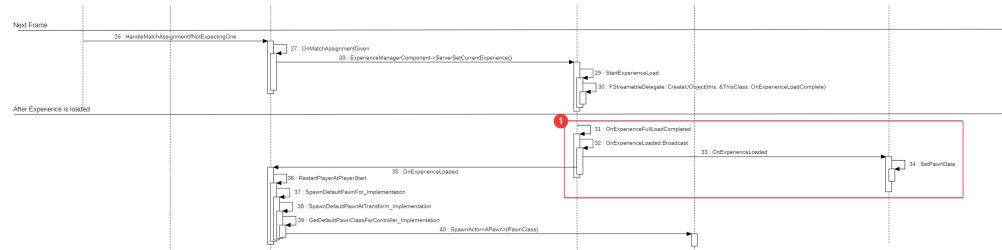
### ▼ 펼치기

□ 우리는 아래 과정을 진행한다:



□ PlayerState의 OnExperienceLoaded:

- 우선 PlayerState의 과정은 아래와 같다:



- 아래와 같이 PlayerState에 대한 OnExperienceLoaded 구현:

```

#pragma once

#include "GameFramework/PlayerState.h"
#include "HakPlayerState.generated.h"

/** forward declaration */
class UHakPawnData;
class UHakExperienceDefinition;

UCLASS()
class AHakPlayerState : public APlayerState
{
    GENERATED_BODY()
public:
    AHakPlayerState(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /**
     * AActor's interface
     */
    virtual void PostInitializeComponents() final;

    /**
     * member methods
     */
    void OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience);
    void SetPawnData(const UHakPawnData* InPawnData);

    UPROPERTY()
    TObjectPtr<const UHakPawnData> PawnData;
};

```

1

```

/**
 * member methods
 */
void AHakPlayerState::SetPawnData(const UHakPawnData* InPawnData)
{
    check(InPawnData);

    // PawnData가 두번 설정되는 것은 원하지 않음!
    check(!PawnData);

    PawnData = InPawnData;
}

void AHakPlayerState::OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience)
{
    if (AHakGameMode* GameMode = GetWorld()->GetAuthGameMode<AHakGameMode>())
    {
        // AHakGameMode에서 GetPawnDataForController를 구현해야 함
        const UHakPawnData* NewPawnData = GameMode->GetPawnDataForController(GetOwningController());
        check(NewPawnData);

        SetPawnData(NewPawnData);
    }
}

```

GetPawnDataForController 구현:

```

const UHakPawnData* AHakGameMode::GetPawnDataForController(const AController* InController) const
{
    // 게임 도중에 PawnData가 오버라이드 되었을 경우, PawnData는 PlayerState에서 가져오게 될
    if (InController)
    {
        if (const AHakPlayerState* HakPS = InController->GetPlayerState<AHakPlayerState>())
        {
            // GetPawnData 구현
            if (const UHakPawnData* PawnData = HakPS->GetPawnData<UHakPawnData>())
                return PawnData;
        }
    }

    // fall back to the default for the current experience
    // 아직 PlayerState에 PawnData가 설정되어 있지 않은 경우, ExperienceManagerComponent의 CurrentExperience로부터 가져와서 설정
    check(GameState);
    UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
    check(ExperienceManagerComponent);

    if (ExperienceManagerComponent->IsExperienceLoaded())
    {
        // GetExperienceChecked 구현
        const UHakExperienceDefinition* Experience = ExperienceManagerComponent->GetCurrentExperienceChecked();
        if (Experience->DefaultPawnData)
        {
            return Experience->DefaultPawnData;
        }
    }

    // 어떠한 케이스에도 핸들링 안되었으면 nullptr
    return nullptr;
}

```

□ HakPawnData 헤더 포함시키기 (`HakGameMode.cpp`)

- 대표적인 예시로 컴파일 타입 최적화의 예시이다!
- 헤더 포함이 아닌 cpp 포함으로 필요한 소스파일에만 헤더를 포함하면서 불필요한 헤더 포함을 막을 수 있다!

□ 디버깅 해보면서 아래를 확인해보자:

- GetAuthGameMode:

```

/**
 * Returns the current Game Mode instance cast to the template type.
 * This can only return a valid pointer on the server and may be null if the cast fails. Will always return null on a client.
 */
template< class T >
T* GetAuthGameMode() const
{
    return Cast<T>(AuthorityGameMode);
}

```

```

void AHakPlayerState::OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience)
{
    if (AHakGameMode* GameMode = GetWorld()->GetAuthGameMode<AHakGameMode>())
    {
        // AHakGameMode에서 GetPawnDataForController를 구현해야 할
        // - GetPawnDataForController에서 우리는 아직 PawnData를 설정하지 않았으므로, ExperienceManagerComponent의 DefaultPawnData로 설정한다
        const UHakPawnData* NewPawnData = GameMode->GetPawnDataForController(GetOwningController()); // 3ms elapsed
        check(NewPawnData);
        SetPawnData(NewPawnData);
    }
}

```

- 이 부분을 통해, UnrealEngine은 어떻게 네트워크 모드와 싱글 모드를 두 개 지원하도록 만드는지 유추가능하다:
  - 싱글모드는 사실상 Server + Client로 볼 수 있다.

- GetPawnDataForController:

```

Const UHakPawnData* AHakGameMode::GetPawnDataForController(const AController* InController) const
{
    // 게임 도중에 PawnData가 오버라이드 되었을 경우, PawnData는 PlayerState에서 가져오게 됨
    if (InController)
    {
        if (const AHakPlayerState* HakPS = InController->GetPlayerState<AHakPlayerState>())
        {
            // GetPawnData 구현
            if (const UHakPawnData* PawnData = HakPS->GetPawnData<UHakPawnData>())
            {
                return PawnData;
            }
        }
    }

    // fall back to the default for the current experience
    // 아직 PlayerState에 PawnData가 설정되어 있지 않은 경우, ExperienceManagerComponent의 CurrentExperience로부터 가져와서 설정
    check(GameState);
    UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
    check(ExperienceManagerComponent);

    if (ExperienceManagerComponent->IsExperienceLoaded())
    {
        // GetExperienceChecked 구현
        const UHakExperienceDefinition* Experience = ExperienceManagerComponent->GetCurrentExperienceChecked();
        if (Experience->DefaultPawnData)
        {
            return Experience->DefaultPawnData; ≤ 2ms elapsed
            [Experience->DefaultPawnData|0x000004f137170cd0 (Name="SimplePawnData")]
        }
    }
}

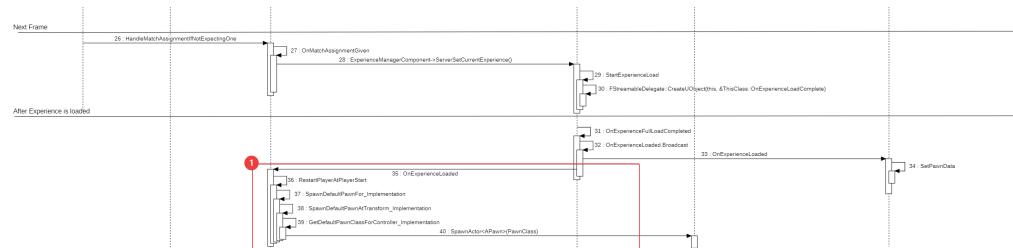
// 어떠한 케이스에도 햄들링 안되었으면 nullptr
return nullptr;

```

1 SimplePawnData를 잘 로딩하고 있다

## □ HakGameMode의 OnExperienceLoaded 구현:

- 우리의 과정은 아래와 같다:



- 코드는 아래와 같다:

```

void AHakGameMode::OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience)
{
    // PlayerController를 순회하며
    for (FConstPlayerControllerIterator Iterator = GetWorld()->GetPlayerControllerIterator(); Iterator; ++Iterator)
    {
        APlayerController* PC = Cast<APlayerController>(*Iterator);

        // PlayerController가 Pawn을 Possess하지 않았다면, RestartPlayer를 통해 Pawn을 다시 Spawn한다
        // - 한번 OnPossess를 보도록 하자:
        if (PC && PC->GetPawn() == nullptr)
        {
            if (PlayerCanRestart(PC))
            {
                RestartPlayer(PC);
            }
        }
    }
}

```

## □ 여기서 Pawn은 언제 Controller에 설정되는건가?

```

void APlayerController::OnPossess(APawn* PawnToPossess)
{
    if ( PawnToPossess != NULL &&
        (PlayerState == NULL || !PlayerState->IsOnlyASpectator() ) )
    {
        const bool bNewPawn = (GetPawn() != PawnToPossess);

        if (GetPawn() && bNewPawn)
        {
            Unpossess();
        }

        PawnToPossess->Controller = NULL;
        PawnToPossess->Controller->Unpossess();

        PawnToPossess->PossessedBy(this);
        // update rotation to match possessed pawn's rotation
        SetControlRotation( PawnToPossess->GetActorRotation() );
    }

    SetPawn(PawnToPossess);
    check(GetPawn() != NULL);

    if (GetPawn() && GetPawn() ->PrimaryActorTick.bStartWithTickEnabled)
    {
        GetPawn() ->SetActorTickEnabled(true);
    }

    INetworkPredictionInterface* NetworkPredictionInterface = GetPawn() ? Cast<INetworkPredictionInterface>(GetPawn() ->GetMovementComponent()) : NULL;
    if (NetworkPredictionInterface)
    {
        NetworkPredictionInterface->ResetPredictionData_Server();
    }

    AcknowledgedPawn = NULL;

    // Local PCs will have the Restart() triggered right away in ClientRestart (via PawnClientRestart()), but the server should call Restart() locally for remote PCs.
    // We're really just trying to avoid calling Restart() multiple times.
    if (!isLocalPlayerController())
    {
        GetPawn() ->DispatchRestart(false);
    }

    ClientRestart(GetPawn());
    ChangeState(NAME_Playing);
    if (!bAutoManageActiveCameraTarget)
    {
        AutoManageActiveCameraTarget(GetPawn());
        ResetCameraMode();
    }
}

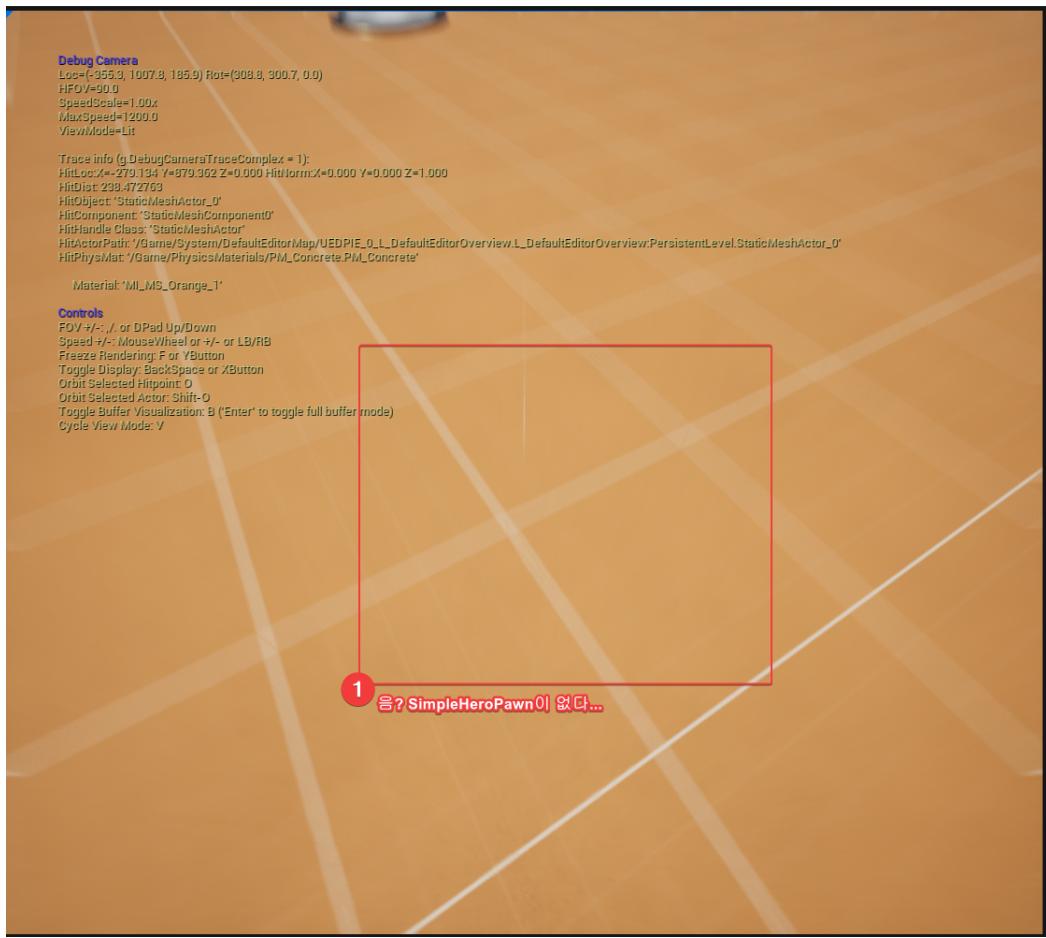
```

□ 뭔가 카메라가 움직이지 않는다?

- 뭔가 PlayerController가 Possess를 잘한거 같다 (그래서 움직이지 않는다):

Name
UnrealEditor-Engine.dll!APlayerController::OnPossess(APawn * PawnToPossess) Line 817
UnrealEditor-Engine.dll!ACController::Possess(APawn * InPawn) Line 333
UnrealEditor-Engine.dll!AGameModeBase::FinishRestartPlayer(AController * NewPlayer, const FMath::TRotator & StartRotation) Line 1343
UnrealEditor-Engine.dll!AGameModeBase::RestartPlayerAtPlayerStart(AController * NewPlayer, AActor * StartSpot) Line 1287
UnrealEditor-Engine.dll!AGameModeBase::RestartPlayer(AController * NewPlayer) Line 1238
UnrealEditor-HakGame.dll!AHakGameMode::OnExperienceLoaded(const UHakExperienceDefinition * CurrentExperience) Line 152
[Inline Frame] UnrealEditor-HakGame.dll!Invoke<void(UHakGameMode::*)(const UHakExperienceDefinition *)> Line 66
[Inline Frame] UnrealEditor-HakGame.dll!UE::Core::Private::Tuple<FTupleBase<FTupleSequence<unsigned int> >::ApplyAfter(void(UHakGameMode::*)(const UHakExperienceDefinition *)) (const UHakExperienceDefinition & _cdecl(UHakGameMode::*,FDefaultDelegateUserPolicy)::ExecuteIfSafe)
UnrealEditor-HakGame.dll!TBaseUObjectMethodDelegateInstance<0,UHakGameMode,void __cdecl(UHakExperienceDefinition const ),FDefaultDelegateUserPolicy>::ExecuteIfSafe() Line 142
[Inline Frame] UnrealEditor-HakGame.dll!TMulticastDelegateBase<FDefaultDelegateUserPolicy>::Broadcast(const UHakExperienceDefinition *) Line 178
[Inline Frame] UnrealEditor-HakGame.dll!UMulticastDelegate<void __cdecl(UHakExperienceDefinition const ),FDefaultDelegateUserPolicy>::Broadcast(const UHakExperienceDefinition *) Line 64
[Inline Frame] UnrealEditor-HakGame.dll!Invoke<void(UHakExperienceManagerComponent::*)()> Line 66
[Inline Frame] UnrealEditor-HakGame.dll!UE::Core::Private::Tuple<FTupleBase<FTupleSequence<unsigned int> >::ApplyAfter(void(UHakExperienceManagerComponent::*)() &) Line 618
UnrealEditor-HakGame.dll!TBaseUObjectMethodDelegateInstance<0,UHakExperienceManagerComponent,void __cdecl(void),FDefaultDelegateUserPolicy>::ExecuteIfSafe() Line 618
[Inline Frame] UnrealEditor-Engine.dll!TDelegate<void __cdecl(void)>::ExecuteIfBound() Line 639

□ 우선 ToggleDebugCamera를 통해, 카메라를 떼서 보자:



□ Pawn이 잘 생성되는지 한번 디버깅해보자:

- 앞서, 우리는 `SpawnDefaultPawnAtTransform_Implementation`을 Override했다
- 여기에 Breakpoint를 넣고 디버깅해보자:

```

52
53     APawn* AHakGameMode::SpawnDefaultPawnAtTransform_Implementation(AController* NewPlayer, const FTransform& SpawnTransform)
54     {
55         UE_LOG(LogHak, Log, TEXT("SpawnDefaultPawnAtTransform_Implementation is called!"));
56         return Super::SpawnDefaultPawnAtTransform_Implementation(NewPlayer, SpawnTransform);
57     }

```

- PawnClass가 제대로 반환 안된다:

```

1201     APawn* AHakGameModeBase::SpawnDefaultPawnAtTransform_Implementation(AController* NewPlayer, const FTransform& SpawnTransform)
1202     {
1203         FActorSpawnParameters SpawnInfo;
1204         SpawnInfo.Instigator = GetInstigator();
1205         SpawnInfo.ObjectFlags = RF_Transient; // We never want to save
1206         UClass* PawnClass = GetDefaultObject(UClass, GetDefaultObject(AController, NewPlayer));
1207         APawn* ResultPawn = GetWorld()>>SpawnActor<APawn>(PawnClass, SpawnTransform, SpawnInfo); // 17ms elapsed
1208         if (!ResultPawn)
1209         {
1210             UE_LOG(LogHakGameMode, Warning, TEXT("SpawnDefaultPawnAtTransformId %d couldn't spawn Pawn of type %s at %s"), >GetNameSafe(PawnClass), >SpawnTransform.ToString());
1211         }
1212     }

```

- 당연히 앞서 우리는 AHakGameMode에 DefaultPawnClass를 AHakCharacter로 했다:

```

AHakGameMode::AHakGameMode(const FObjectInitializer& ObjectInitializer)
    : Super(ObjectInitializer)
{
    // 해당 클래스의 Clone 대상이 되는 LyraGameMode를 살펴보자:
    // - 우리는 첫번째로 당장 필요한 GameState, PlayerController, PlayerState와 Character를 구현해보자:
    GameStateClass = AHakGameState::StaticClass();
    PlayerControllerClass = AHakPlayerController::StaticClass();
    PlayerStateClass = AHakPlayerState::StaticClass();
    DefaultPawnClass = AHakCharacter::StaticClass(); ①
}

```

- 이를 오버라이드하기 위해, PlayerState의 SetPawn의 대상으로 만들었던 `GetPawnDataForController` 함수를 활용하여, `GetDefaultPawnClassForController` 함수를 오버라이드해서 구현해주자:

```

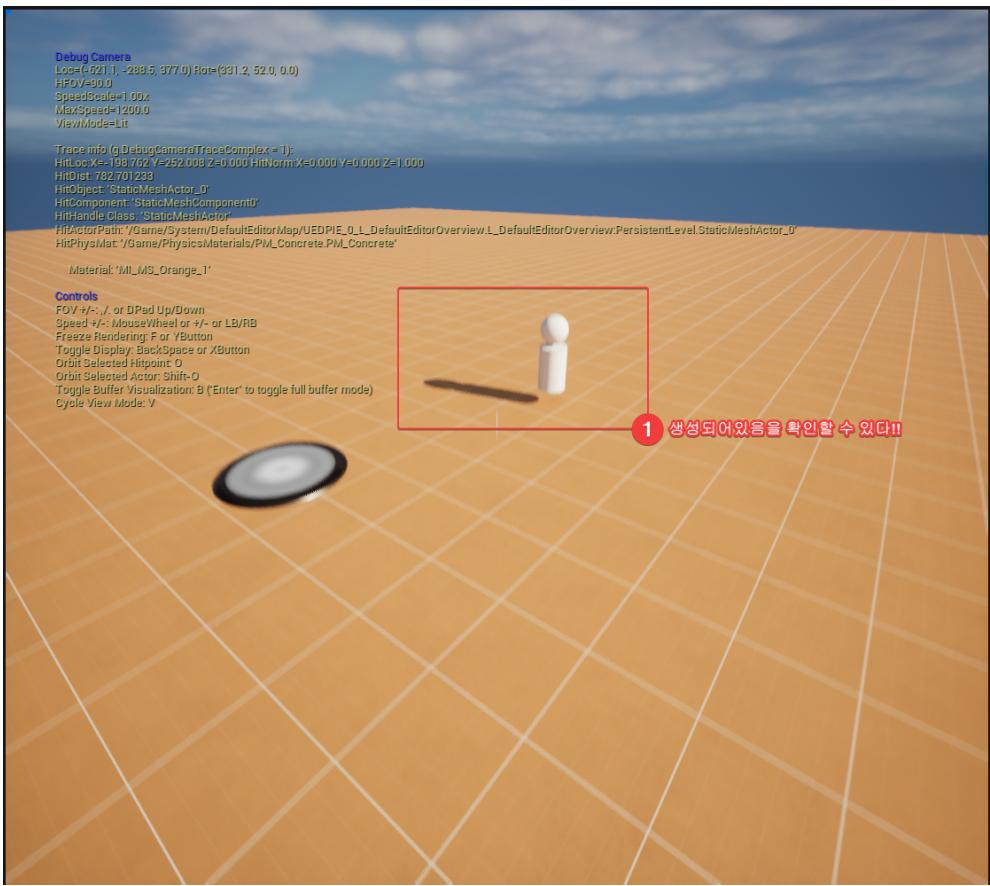
UClass* AHakGameMode::GetDefaultPawnClassForController_Implementation(AController* InController)
{
    // GetPawnDataForController를 활용하여, PawnData로부터 PawnClass를 유도하자
    if (const UHakPawnData* PawnData = GetPawnDataForController(InController))
    {
        if (PawnData->PawnClass)
        {
            return PawnData->PawnClass;
        }
    }
    return Super::GetDefaultPawnClassForController_Implementation(InController);
}

```

- 그럼 아래와 같이 뭔가 보인다:



- `ToggleDebugCamera`를 통해 보자:



## 중간 정리

### ▼ 펼치기

- 지금까지 우리는 Lyra 프로젝트에서 Experience의 로딩까지 클론 코딩해보았다
- 이제 앞서 만들었던 [SimpleHeroPawn의 HakCharacter를 채워나가며, 조작\(카메라와 입력\)에 대해 클론 코딩을 진행한다](#)

## HakGameplayTags

### ▼ 펼치기

- HakGameplayTags.h/.cpp 파일을 추가해주자
- 아래와 같이 HakGameplayTags를 간단히 구현:

```

#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 디제를 의미한다;
     * - GameInstances의 초기화 과정에 UGameFrameworkComponentManager의 RegisterInitState로 등록되어 선형적으로(Linear)하게 업데이트 된다
     * - 이 초기화 GameplayTags는 개체의 Actor 사이에 공유되며, GameframeworkInitStateInterface를 상속받은 클래스는 초기화 상태(Init State)를 상태미신(State Machine)과 같이 관리 가능한 인터페이스를 제공한다
     */
    FGameplayTag InitState_Spawned;
    FGameplayTag InitState_DataAvailable;
    FGameplayTag InitState_DataInitialized;
    FGameplayTag InitState_GameplayReady;

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

```

```

#include "HakGameplayTags.h"
#include "HakLogChannels.h"
#include "GameplayTagsManager.h"

FHakGameplayTags FHakGameplayTags::GameplayTags;

void FHakGameplayTags::InitializeNativeTags()
{
    UGameplayTagsManager& Manager = UGameplayTagsManager::Get();
    GameplayTags.AddAllTags(Manager);
}

void FHakGameplayTags::AddTag(FGameplayTag& OutTag, const ANSICHAR* TagName, const ANSICHAR* TagComment)
{
    OutTag = UGameplayTagsManager::Get().AddNativeGameplayTag(FName(TagName), FString(TEXT("Native ")) + FString(TagComment));
}

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 ready for full gameplay");
    AddTag(InitState_GameplayReady, "InitState.GameplayReady", "4: The actor/component is fully ready for active gameplay");
}

```

## □ HakGameplayTags의 초기화 (HakAssetManager에서!):

```

/**
 * UAssetManager's interfaces
 */
void UHakAssetManager::StartInitialLoading()
{
    // 오버라이드할 경우, Super의 호출은 꼭 깨끗이 맡자
    Super::StartInitialLoading();

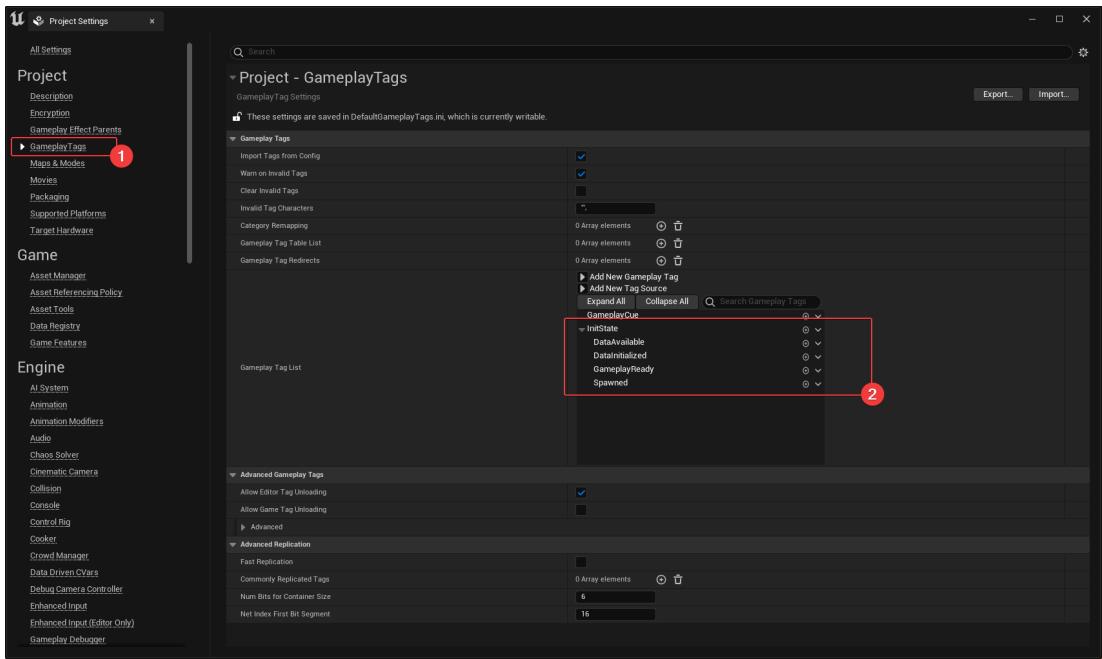
    // HakGameplayTags 초기화
    // - Lyra와의 차이점을 한번 보고 가자:
    // - STARTUP_JOB() 매크로를 활용하고 있으나, 현재 우리는 로딩과 관계 없으므로 간단하게 구현
    FHakGameplayTags::InitializeNativeTags();
}

```

## □ 왜 HakGameplayTags 초기화가 AssetManager 초기화 함수를 이용할 만큼 빨라야 했을까?

- 우선 GameInstance 오버라이드를 진행하면 알게 된다

## □ 잘 GameplayTag가 추가되었을까?



# HakGameInstance

## ▼ 펼치기

- HakGameInstance의 오버라이드가 필요한 이유는?

- GameFrameworkComponentManager에 InitState의 GameplayTags를 추가시켜줘야 한다
- 이는 공식 스펙:

## Init States

Init States are implemented as

[Gameplay Tags](making-interactive-experiences\interactive-framework\Tags) and must be registered with the subsystem by calling `RegisterInitState` during GameInstance initialization. These states are registered in order and shared by all Actors in a game. For instance, a game could support a simple 2 state system with `InitState.Spawning` and `InitState.Ready` or a more complex system like the Lyra example below.

- HakGameInstance.h/.cpp 추가
- 아래와 같이 구현:

```

1 #include "Engine/GameInstance.h"
2 #include "HakGameInstance.generated.h"
3
4 /**
5  * GameInstance는 게임 프로세스(.exe)에서 하나만 존재하는 객체로 생각하면 된다 (= High-level Manager Object)
6  * - 게임이 켜질때, 만들어지고, 게임이 꺼지기 전까지 살아있다
7  * - Editor 상에서는 PIE로 실행 될때마다 하나씩 생성된다: 즉, 에디터에서는 복수개의 GameInstance가 존재 가능하다!
8 */
9 UCLASS()
10 class UHakGameInstance : public UGameInstance
11 {
12     GENERATED_BODY()
13 public:
14     UHakGameInstance(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
15
16     /**
17      * UGameInstance's interfaces
18      */
19     virtual void Init() override;
20     virtual void Shutdown() override;
21 };

```

1

## • GameInstance에 대한 설명

```

1 #include "HakGameInstance.h"
2 #include "HakGame/HakGamePlayTags.h"
3 #include "Components/GameFrameworkComponentManager.h"
4 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakGameInstance)
5
6 UHakGameInstance::UHakGameInstance(const FObjectInitializer& ObjectInitializer)
7     : Super(ObjectInitializer)
8 {
9 }
10
11 void UHakGameInstance::Init()
12 {
13     Super::Init();
14
15     // 앞서 정의한 InitState의 GameplayTags 등록:
16     UGameFrameworkComponentManager* ComponentManager = GetSubsystem<UGameFrameworkComponentManager>(this);
17     if (ensure(ComponentManager))
18     {
19         const FHakGameplayTags& GameplayTags = FHakGameplayTags::Get();
20
21         ComponentManager->RegisterInitState(GameplayTags.InitState_Spawned, false, FGameplayTag());
22         ComponentManager->RegisterInitState(GameplayTags.InitState_DataAvailable, false, GameplayTags.InitState_Spawned);
23         ComponentManager->RegisterInitState(GameplayTags.InitState_DataInitialized, false, GameplayTags.InitState_DataAvailable);
24         ComponentManager->RegisterInitState(GameplayTags.InitState_GameplayReady, false, GameplayTags.InitState_DataInitialized);
25     }
26 }
27
28 void UHakGameInstance::Shutdown()
29 {
30     Super::Shutdown();
31 }

```

1

## • GameFrameworkComponentManager에 대한 간단한 설명:

- Game Feature Plugin을 사용하기 위해 설계된 Game Instance Subsystem으로 생각하면 된다
- 크게, 아래 두 가지 기능을 제공한다:

### 1. Extension Handlers

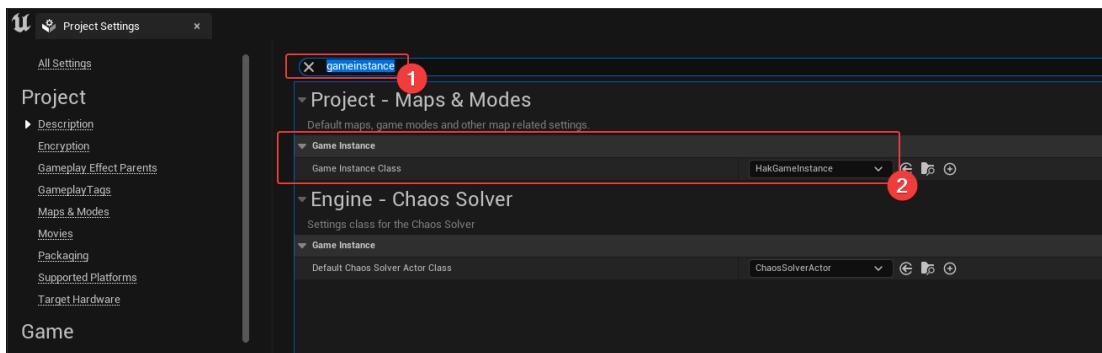
- 해당 기능은 Game Feature가 활성화될 경우, Component와 같은 Action을 추가/제거/수정을 가능하게 한다
- 아직 확 와닫지 않는다... 후일 우리가 Game Feature 기능을 클론할 때, 더 보도록 하자

### 2. Initialization States

- 네트워크 게임의 Component 초기화는 복잡하다

- 복잡한 단계를 좀 더 깔끔하게 나누어 관리할 수 있도록 기능 제공 한다
- 앞서 추가한 Gameplay Tag는 2번째 기능인 Initialization State를 사용자가 정의할 수 있도록 제공하는 함수 RegisterInitState를 통해 등록하였다.

□ HakGameInstance 오버라이드:



## 자료

### ▼ 펼치기

[HakUserExperience.mdj](#)