



5주차 (2023.10.16)

Module과 Plugin 그리고 Project

▼ 펼치기

조교님께서 너무 잘 정리해주셨는데 아래의 그림에 잘못된 전달이 될 수 있을까봐.

다시 한번 Module과 Plugin 그리고 Project를 정리하고자 합니다:

Module

- Module은 하나의 dll 단위로 라이브러리로 생각하도록 합시다.
- `.h/.cpp` 를 묶는 하나의 최소 단위로 생각할 수 있습니다.
- Unreal Engine은 Module 정의를 위해 필요한 구성 요소는 아래와 같습니다:

우선 Module의 이름을 HakFeature라고 합시다:

1. HakFeature.Build.cs
2. FHakFeatureModule.h/.cpp

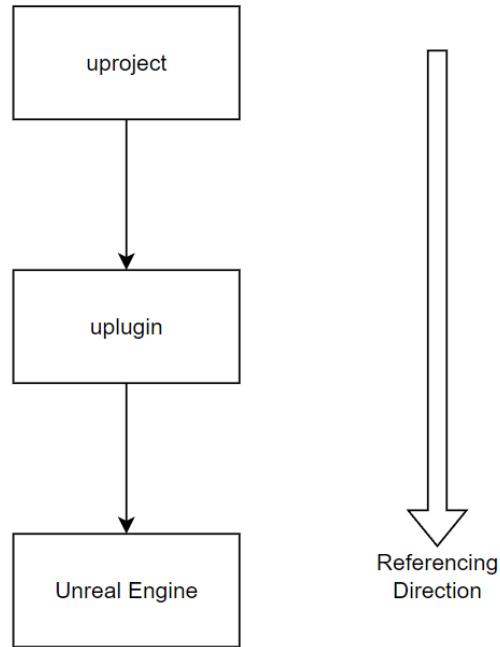
- `IMPLEMENT_PRIMARY_GAME_MODULE`

- 참고로, Module은 단독으로 Unreal Engine에서 구성할 수 없습니다:

- Plugin 혹은 Project에 종속되어 활성화 되어야 합니다!

Plugin [Code]

- Unreal Engine을 확장하는 단위:
- 그림으로 설명하면 아래와 같이 uni-direction한 참조 구조를 가지게 된다:



- 반대로, **Unreal Engine에서 Plugin을 참조할 수 없다!**

- 보통 Unreal Engine에서도 Experimental한 Plugin은 따로 외부 Plugin으로 구성한 이후, 충분히 검증되면(Production-Ready), 내부 Module로 통합한다.

- **알아두면 좋은 특징들:**

- 복수 개의 Module로 구성:
 - **.h/.cpp로 구성할 수 없음!**
 - 복수 개의 Plugin을 참조 가능
 - description 파일로 **.uplugin** 을 참고
 - **독립적인 Content 폴더** 구성 가능

- Plugin의 구성 요소는 아래와 같습니다:

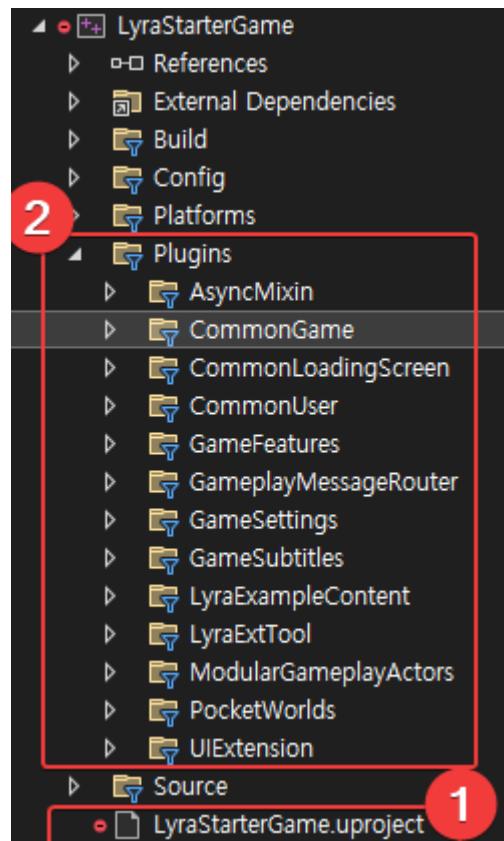
우선 Plugin의 이름을 HakPlugin이라고 합시다:

1. HakPlugin.uplugin
2. 1개 이상의 Module들 (**없어도 됨; 자세한 건 아래에서 설명**)

Project

- Unreal Engine의 활용 대상이 되는 프로젝트
- **알아두면 좋은 특징들:**

- 복수 개의 Module로 구성:
 - Plugin과 마찬가지로 .h/.cpp로 구성할 수 없음!
- 복수 개의 Plugin 참조 및 구성 가능:
 - 아래의 그림과 같이, 하나의 uproject에 대해 여러 개의 Plugin을 포함도 가능하다는 것을 알 수 있다!



- 물론 구성하지 않는 외부의 Plugin도 참조 가능하다!
 - description 파일로 .uproject 를 참고
- Project의 구성 요소는 아래와 같습니다:

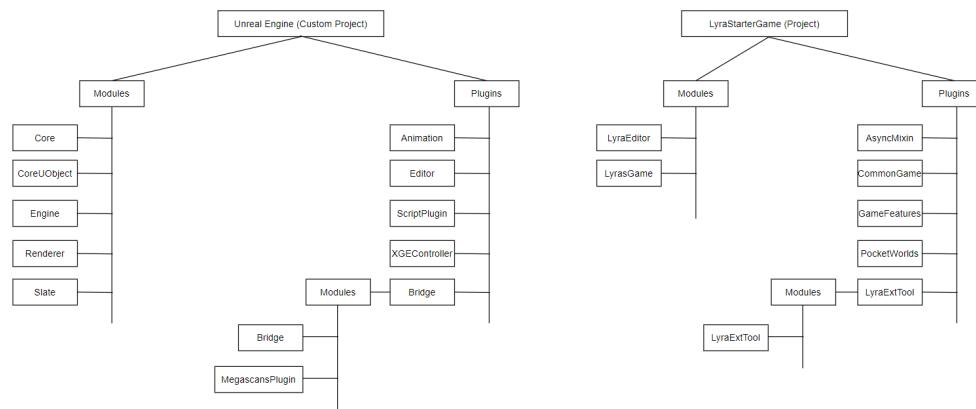
우선 Project의 이름을 Hak라고 하자:

 1. Hak.uproject
 2. Hak[Editor|Game|Server|Client].Target.cs:
 - 빌드 대상을 나누고 싶은 개수대로 추가 가능
 3. 1개 이상 Module들

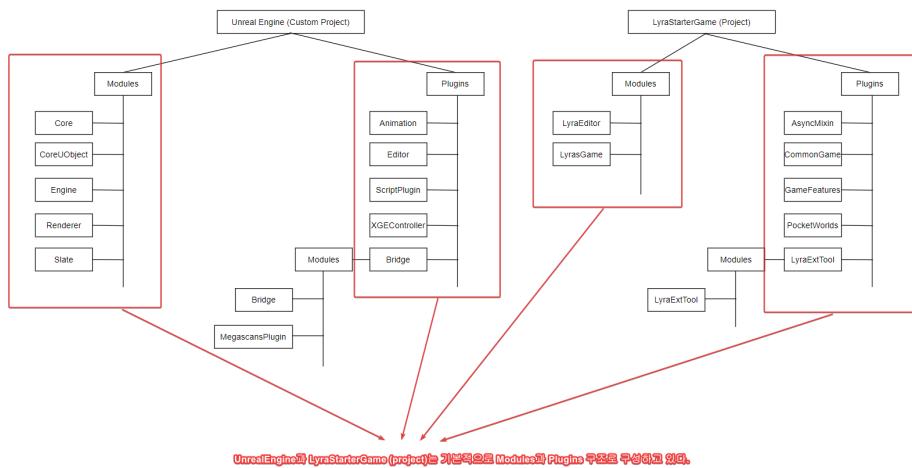
Plugin [Content]

- Unreal Engine은 BP(Blueprint)가 있고, 이를 통해서 코드 없이 에셋으로 Plugin을 구성할 수 있다.
- 앞서 언급했듯이, Plugin은 독립적인 Content 구조를 가져갈 수 있다:
 - 해당 경로에 BP와 같은 로직 관련 에셋을 통해 Unreal Engine을 확장하는 구조를 가져갈 수 있다!
- 이 경우, Editor를 통해 Plugin 생성을 추천한다.

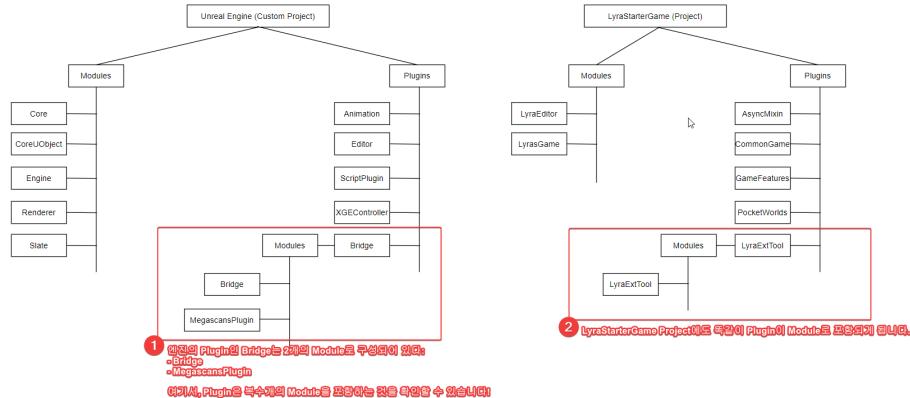
□ 정리하면, Unreal Engine과 LyraStarterGame을 통해 아래의 구조를 확인할 수 있다:



- 여기서 주목해야 할 부분을 좀 더 살펴보자:
 1. Unreal Engine과 Project(uproject)는 Module과 Plugin으로 구성되어 있다:



2. Plugin은 Module로 구성되어 있다:



사실 여러분들이 실제로 **Plugin과 Module**는 **Build.cs** 분석 및 **uplugin과 uproject** 파일에 대한 속성 값들을 이해해야 제대로 활용 가능하다:

- Circular Referencing에 대한 이해도 있어야 한다...
- 헌데, 이는 해당 과정에서 좀 벗어난 이야기이니, 다음에 기회가 되면 또 깊게 다루어보기로 하고, 여기까지 Module ↔ Plugin, Project에 대해 이해하고 넘어가도록 하자

GameMode 및 구성 요소들 구현

▼ 펼치기

GameMode 클래스를 구현해보자:

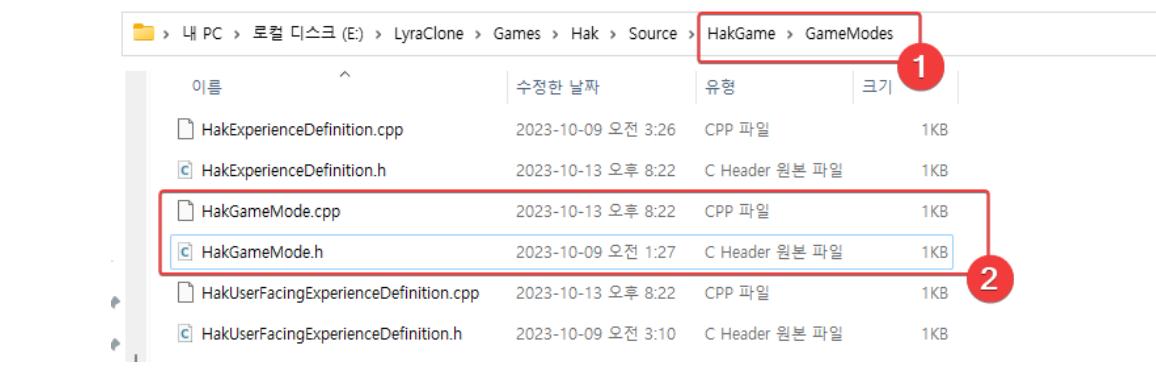
이미 우리는 UE5.sln로 프로젝트를 생성할 당시, 자동적으로 GameMode 클래스가 담긴 HakGameModeBase.h/.cpp가 생성되어 있다:

이름	수정한 날짜	유형	크기
Character	2023-10-13 오후 8:22	파일 폴더	
GameModes	2023-10-13 오후 8:22	파일 폴더	
System	2023-10-13 오후 8:22	파일 폴더	
HakGame.Build.cs	2023-10-09 오전 1:22	CS 파일	1KB
HakGame.cpp	2023-10-13 오후 8:22	CPP 파일	1KB
HakGame.h	2023-10-08 오후 8:49	C Header 원본 파일	1KB
HakGameModeBase.cpp	2023-10-13 오후 8:22	CPP 파일	1KB
HakGameModeBase.h	2023-10-09 오전 1:27	C Header 원본 파일	1KB
HakLogChannels.cpp	2023-10-13 오후 8:22	CPP 파일	1KB
HakLogChannels.h	2023-10-09 오전 1:43	C Header 원본 파일	1KB

□ 이를 이름을 변환하고, GameModes의 폴더로 옮겨주자:



□ 해당 이름을 HakGameMode.h/.cpp로 변경시켜주자:



✓ 아래와 같이 빠르게 리펙토링 해보자:

```
#pragma once

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "HakGameMode.generated.h" 1

/** 
 * HakGameMode
 */
UCLASS()
class AHakGameMode : public AGameModeBase
{
    GENERATED_BODY() 2
public:
    AHakGameMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get()); 3
};
```

```

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

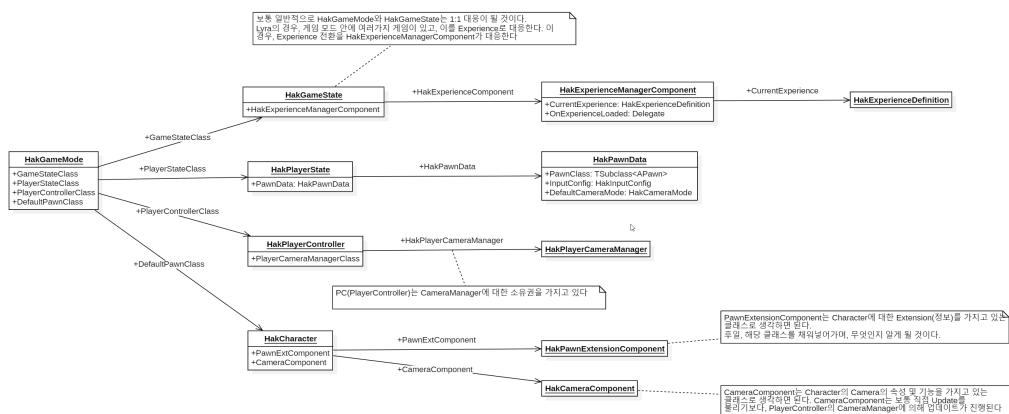
AHakGameMode::AHakGameMode(const FObjectInitializer& ObjectInitializer)
: Super(ObjectInitializer)
{
    // 해당 클래스의 Clone 대상이 되는 LyraGameMode를 살펴보자:
    // - 우리는 첫번째로 당장 필요한 GameState, PlayerController, PlayerState와 Character를 구현해보자:
    GameStateClass = AHakGameState::StaticClass();
    PlayerControllerClass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

```

- 여기서 우리는 빠르게 GameMode에 필요한 기본 구성 요소를 파악할 수 있다:

- AGameState
- AHakPlayerController
- AHakPlayerState
- AHakCharacter

이를 간단히 설명해보면 아래와 같다:



- 우선 해당 클래스들만 간단히 구현해놓자:

- HakGameState → [GameModes]
- HakPlayerController → [Player]
- HakPlayerState → [Player]
- HakCharacter → [Character]
- GameMode의 Class 정의를 헤더 파일을 통해, 마무리하자:

```

1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #include "HakGameMode.h"
4 #include "HakGameState.h"
5 #include "HakGame/Player/HakPlayerController.h"
6 #include "HakGame/Player/HakPlayerState.h"
7 #include "HakGame/Character/HakCharacter.h"
8
9
10 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakGameMode)
11
12 AHakGameMode::AHakGameMode(const FObjectInitializer& ObjectInitializer)
13 : Super(ObjectInitializer)
14 {
15     // 해당 클래스의 Clone 대상이 되는 LyraGameMode를 살펴보자:
16     // - 우리는 첫번째로 당장 필요한 GameState, PlayerController, PlayerState와 Character를 구현해보자:
17     GameStateClass = AHakGameState::StaticClass();
18     PlayerControllerClass = AHakPlayerController::StaticClass();
19     PlayerStateClass = AHakPlayerState::StaticClass();
20     DefaultPawnClass = AHakCharacter::StaticClass();
21 }
22
23
24

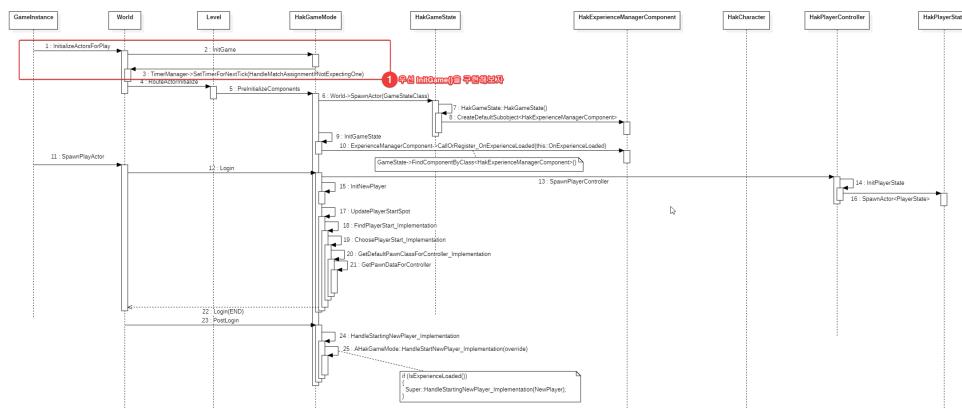
```

HakGameMode::InitGame()

▼ 펼치기

□ 하나씩, 위에서 아래로 구현해보자:

우선 `HakGameMode::InitGame()` 을 보자



□ 아래를 유의하며 보자:

```

void AHakGameMode::InitGame(const FString& MapName, const FString& Options, FString& ErrorMessage)
{
    Super::InitGame(MapName, Options, ErrorMessage);
    // 아직 GameInstance를 통해, 초기화 작업이 진행되므로, 현 프레임에는 Lyra의 Concept인 Experience 처리를 진행할 수 없다:
    // - 이를 처리하기 위해, 한프레임 뒤에 이벤트를 받아 처리를 이어서 진행한다
    GetWorld()->GetTimerManager().SetTimerForNextTick(this, &ThisClass::HandleMatchAssignmentIfNotExpectingOne);
}

```

□ 우선 HandleMatchAssignmentIfNotExpectingOne을 비워놓고 Delegate를 걸어놓자:

```

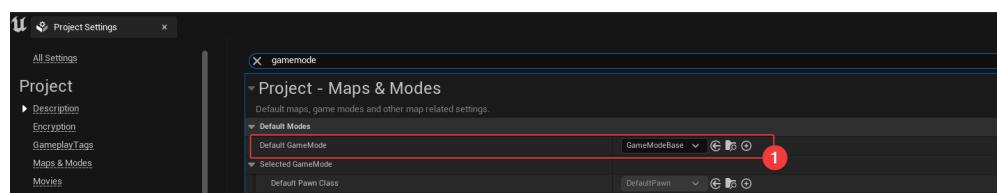
/*
* member methods
*/
void AHakGameMode::HandleMatchAssignmentIfNotExpectingOne()
{
}

```

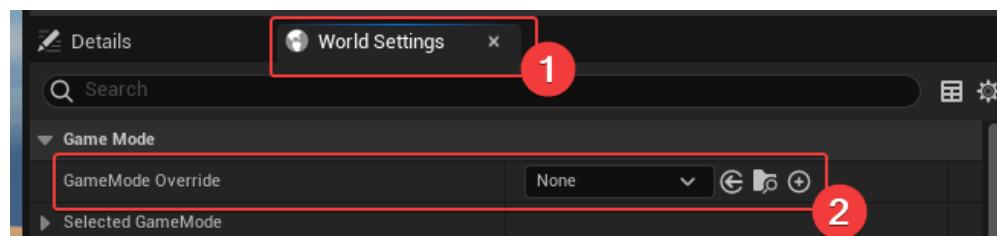
□ 한번 InitGame()의 호출의 Callstack을 확인해보자:

□ 근데 InitGame()에서 Breakpoint가 호출이 안된다:

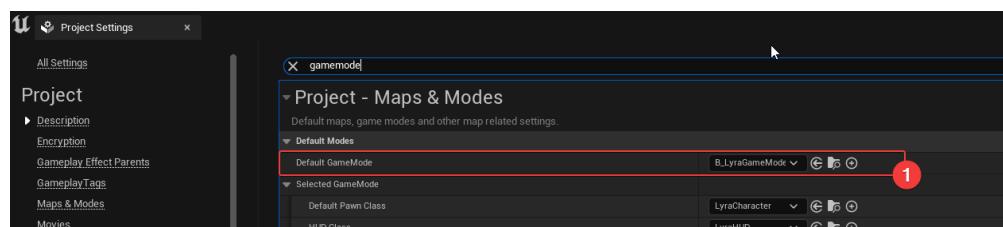
- 아! GameMode0 | HakGameMode로 Override가 안되었는가보다:



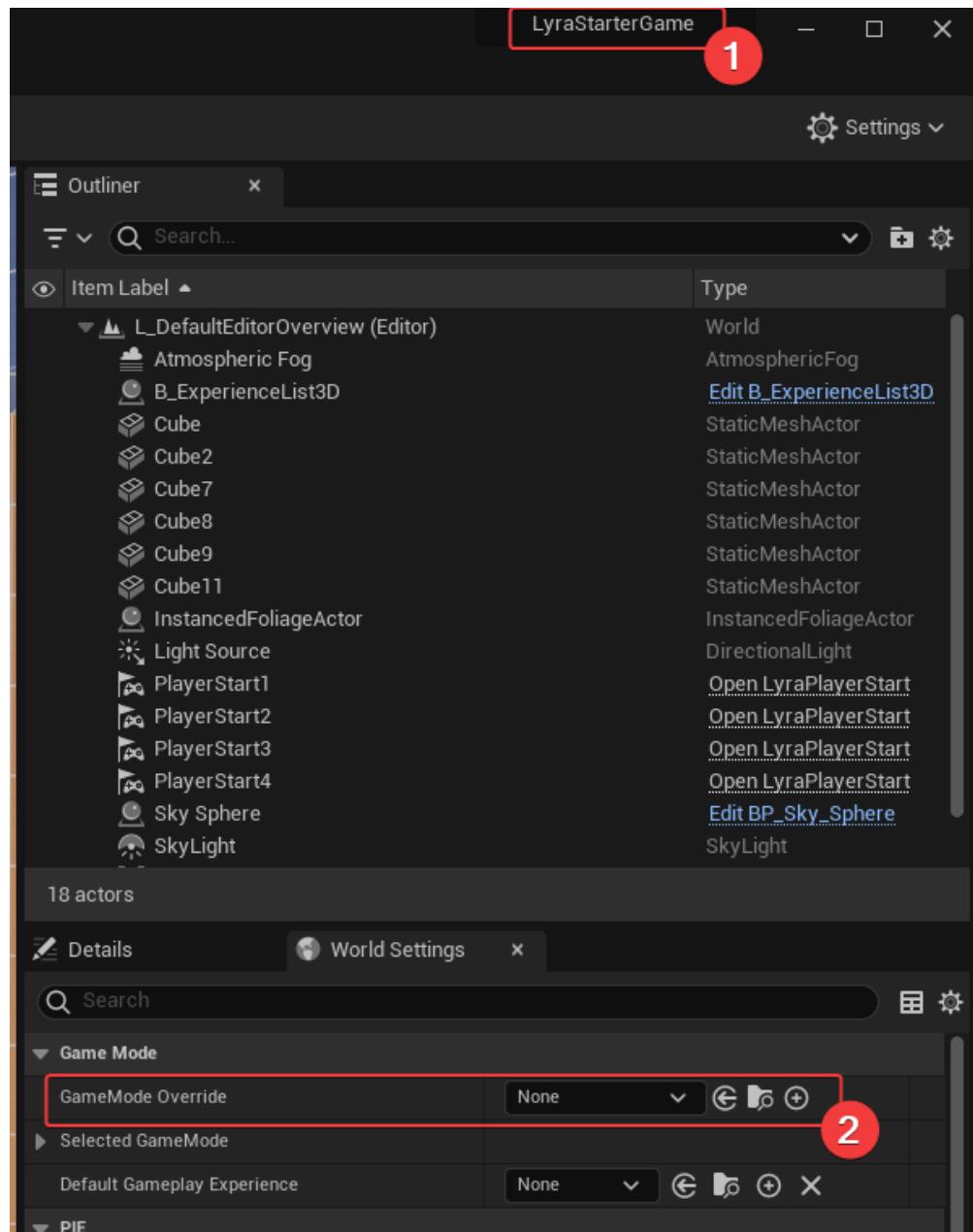
그런데 게임 모드는 아래와 같이 Persistent Level에 대해서 WorldSetting에서도 설정이 가능하다:



□ 그럼 Lyra에 대해서 어떻게 설정되었는지 보자:



그렇다면 World Settings은?



Project Settings에서 DefaultGameMode를 HakGameMode로 설정 해주자:

- 아마 다음과 같은 이유가 있지 않을까 생각해본다:
 - Lyra는 WorldSettings을 통한 GameMode 변경으로 게임의 장르를 바꾸지 않고, Experience를 통해 게임의 장르를 바꾸는 방식 이기에, GameMode는 항상 맵 관계 없이 동일해지기 때문에 그런 게 아닌가 싶다

그럼 이제 Breakpoint가 잡힌다:

```

23  /**
24  * AGameModeBase interface
25  */
26
27 void AHakGameMode::InitGame(const FString& MapName, const FString& Options, FString& ErrorMessage)
28 {
29     Super::InitGame(MapName, Options, ErrorMessage);
30
31     // 아직 GameInstance를 통해, 초기화 작업이 진행되므로, 현 프레임에는 Lyra의 Concept인 Experience 처리를 진행할 수 없다:
32     // - 이를 처리하기 위해, 한프레임 뒤에 이벤트를 받아 처리를 이어서 진행한다
33     GetWorld()>>GetTimerManager().SetTimerForNextTick(this, &ThisClass::HandleMatchAssignmentIfNotExpectingOne);
34 }
35

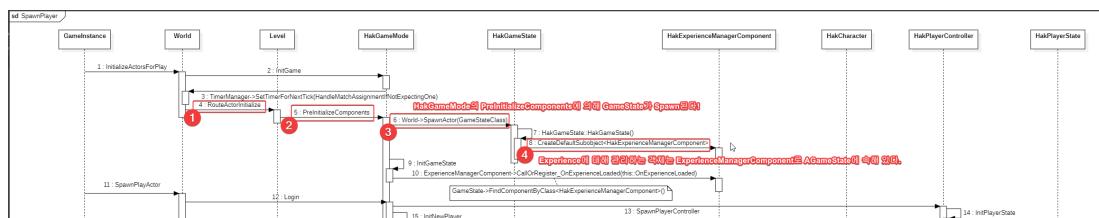
```

- 호출의 큰 흐름을 보면: GameInstance → World → GameMode이다
- 기억해두셔야 할 점은 GameMode의 초기화가 Unreal의 GameFramework 시작점으로 이해하면, 관련되어 디버깅이 필요할 경우 여기를 진입점으로 삼으시면 되겠다!

HakGameState

▼ 펼치기

- 아래의 시퀀서를 보며, HakGameState 초기화 과정을 살펴보자:



- 참고로, 우리가 HakGameState에서 따로 Override나 구현할 부분은 없다:
 - 앞서, GameStateClass를 우리가 Override했기 때문에, 위와 같은 과정을 거쳐 초기화가 진행된다

- LyraExperienceManagerComponent 확인:

- LyraExperienceManagerComponent는 GameStateComponent를 상속 받음:

```

1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "Components/GameStateComponent.h"
7 #include "GameFeaturePlugin/OperationResult.h"
8 #include "LoadingProcessInterface.h"
9
10 #include "LyraExperienceManagerComponent.generated.h"
11
12 class ULYRAExperienceDefinition;
13
14 DECLARE_MULTICAST_DELEGATE_OneParam(FOnLyraExperienceLoaded, const ULYRAExperienceDefinition* /*Experience*/);
15
16 ELYRAExperienceLoadState
17 {
18     Unloaded,
19     Loading,
20     LoadingGameFeatures,
21     LoadingChaosTestingDelay,
22     ExecutingActions,
23     Loaded,
24     Deactivating
25 };
26
27 UCCLASS()
28 ECClass ULYRAExperienceManagerComponent final : public UGameStateComponent, public ILoadingProcessInterface
29 {
30     GENERATED_BODY()
31
32 public:
33
34     ULYRAExperienceManagerComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
35
36     //~UActorComponent interface
37     virtual void EndPlay(const EEndPlayReason::Type EndPlayReason) override;
38     //~-UActorComponent interface
39
40     //~ILoadingProcessInterface interface

```

- GameStateComponent는 ModularGameplay에 있음:

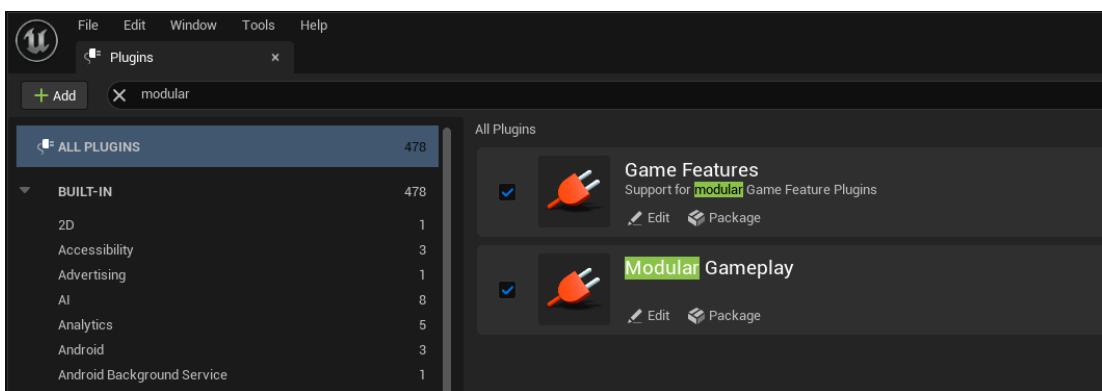
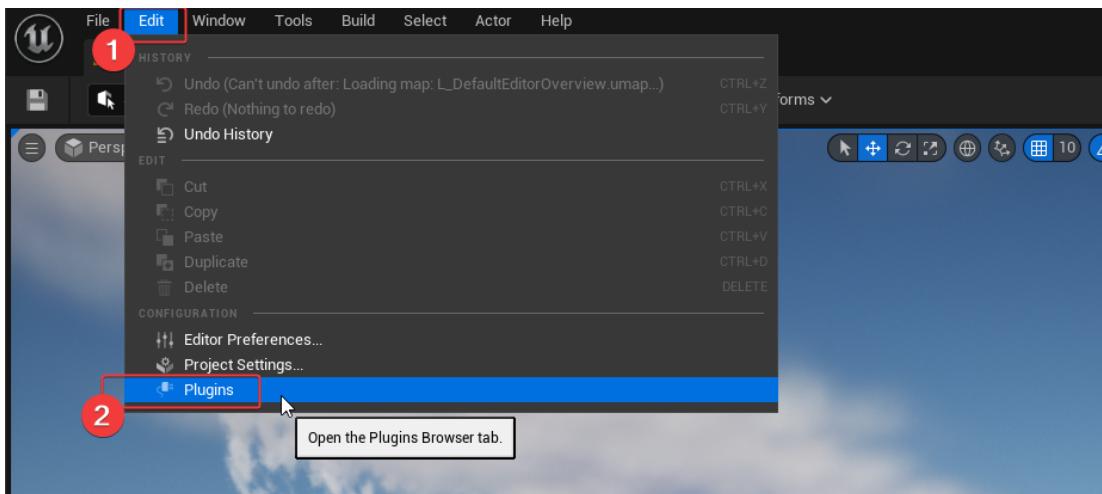
```

1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
4
5 #include "GameFramework/GameState.h"
6 #include "GameFramework/GameMode.h"
7 #include "GameFramework/Component.h"
8 #include "GameStateComponent.generated.h"
9
10 /**
11  * GameStateComponent is an actor component made for AGameStateBase and receives GameState events.
12  */
13 UCCLASS()
14 ECClass MODULARGAMEPLAY_API UGameStateComponent : public UGameFrameworkComponent
15 {
16     GENERATED_BODY()
17
18 public:
19
20     UGameStateComponent(const FObjectInitializer& ObjectInitializer);
21
22     /** Gets the game state that owns the component, this will always be valid during gameplay but can return null in the editor */
23     template <class T>
24     T* GetGameState() const
25     {
26         static_assert(TPointerIsConvertibleFromTo<T, AGameStateBase>::Value, "'T' template parameter to GetGameState must be derived from AGameStateBase");
27         return Cast<T>(GetOwner());
28     }

```

- 우선 우리에게 필요한 3개의 Plugin을 활성화를 진행해주자:

- GameFeatures
- ModularGameplay
- GameplayAbilities



아래와 같이 Hak.uproject 파일에 3개의 Plugin이 잘 활성화된 것을 확인할 수 있다:

```
1  "FileVersion": 3,
2  "EngineAssociation": "",
3  "Category": "",
4  "Description": "",
5  "Modules": [
6    {
7      "Name": "HakGame",
8      "Type": "Runtime",
9      "LoadingPhase": "Default",
10     "AdditionalDependencies": [
11       "Engine"
12     ]
13   }
14 ],
15 ],
16 "Plugins": [
17   {
18     "Name": "ModelingToolsEditorMode",
19     "Enabled": true,
20     "TargetAllowList": [
21       "Editor"
22     ]
23 },
24   {
25     "Name": "GameFeatures",
26     "Enabled": true
27   },
28   {
29     "Name": "ModularGameplay",
30     "Enabled": true
31   },
32   {
33     "Name": "GameplayAbilities",
34     "Enabled": true
35   }
36 ],
37 }
```

□ 우리에게 필요한 Plugin을 활성화해주었으니, HakGame.Build.cs에 우리가 필요한 Module을 추가할 준비가 되었다:

- 현재 우리에게 필요한 ModularGameplay Module을 추가해주자:

```

Hak.uproject LyraGame.Build.cs ModularGameplay.Build.cs + X GameStateComponent.h HakGame.Target.cs HakExperienceManagerComponent.h
ModularGameplay.Modula ▾ 1 new string[]
1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 using UnrealBuildTool;
4
5 1 reference
6 ▾ public class ModularGameplay : ModuleRules
7 {
8     0 references
9     public ModularGameplay(ReadOnlyTargetRules Target) : base(Target)
10    {
11         PCHUsage = ModuleRules.PCHUsageMode.UseExplicitOrSharedPCHs;
12
13         PrivateIncludePaths.AddRange(
14             new string[]
15             {
16                 "ModularGameplay/Private"
17             }
18         );
19
20         PublicDependencyModuleNames.AddRange(
21             new string[]
22             {
23                 "Core",
24                 "Engine",
25                 "GameplayTags",
26             }
27         );
28
29         PrivateDependencyModuleNames.AddRange(
30             new string[]
31             {
32                 "CoreUObject",
33             }
34         );
35
36         DynamicallyLoadedModuleNames.AddRange(
37             new string[]
38             {
39             }
40         );
41     }
42 }

```

항상 Module을 추가하기 전에, {Module}.Build.cs를 확인하여, 추가적으로 종속성이 있는 Module을 항상 체크해서, 추가적인 Module을 종속성을 연결해줘야 한다.

□ 아래와 같이 HakGame.Build.cs에 Module을 추가해주자:

```

1 reference
ModularGameplay.Modula ▾ 0 references
public class HakGame : ModuleRules
{
    0 references
    public HakGame(ReadOnlyTargetRules Target) : base(Target)
    {
        PCHUsage = PCHUsageMode.UseExplicitOrSharedPCHs;

        PublicDependencyModuleNames.AddRange(new string[] {
            "Core",
            "CoreUObject",
            "Engine",
            "InputCore",
            // GAS
            "GameplayTags",
            // Game Features
            "ModularGameplay",
        });
1
        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
    }
}

```

□ ExperienceManagerComponent 구현:

□ HakExperienceManagerComponent에 대한 설명:

```

1 #pragma once
2
3 #include "Components/GameStateComponent.h"
4 #include "HakExperienceManagerComponent.generated.h"
5
6 /** forward declaration */
7 class UHakExperienceDefinition;
8
9 /**
10  * HakExperienceManagerComponent
11  * - 말 그대로, 해당 component는 game state를 owner로 가지면서, experience의 상태 정보를 가지고 있는 component이다
12  * - 뿐만 아니라, manager라는 단어가 포함되어 있어서, experience 보딩 상태 업데이트 및 이벤트를 관리한다
13 */
14 UCLASS()
15 class UHakExperienceManagerComponent : public UGameStateComponent
16 {
17     GENERATED_BODY()
18 public:
19     UHakExperienceManagerComponent(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
20
21     /**
22      * member variables
23     */
24
25     /**
26      * 참고로 해당 멤버 변수는 Lyra에서는 'ReplicatedUsing='으로 선언되어있다:
27      * - 현재 우리는 아직 Replication을 신경쓰지 않을 것이기에, 최대한 네트워크 서버 코드를 배제하도록 하겠다
28     */
29     UPROPERTY()
30     TObjectPtr<const UHakExperienceDefinition> CurrentExperience;
31 };

```

- HakExperienceManagerComponent.h/.cpp 추가 → [GameModes]
- CurrentExperience 추가:
 - LyraExperienceManagerComponent와 차이:
 - Replication을 최대한 배제하면서 진행:
 - OnRep_ 함수 및 추가적인 GetLifetimeReplicatedProps도 신경 써야 함
 - 후일, LyraClone에서 네트워크 항목을 진행할 때 할듯 (아직 미정!)
- HakGameState에 ExperienceManagerComponent 추가:

```

1 #pragma once
2
3 #include "GameFramework/GameStateBase.h"
4 #include "HakGameState.generated.h"
5
6 /** forward declaration */
7 class UHakExperienceManagerComponent;
8
9 /**
10  * AHakGameState
11  */
12 UCLASS()
13 class AHakGameState : public AGameStateBase
14 {
15     GENERATED_BODY()
16 public:
17     AHakGameState(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
18
19     UPROPERTY()
20     TObjectPtr<UHakExperienceManagerComponent> ExperienceManagerComponent;
21 };

```

```

1 #include "HakGameState.h"
2 #include "HakExperienceManagerComponent.h" 1
3 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakGameState)
4
5 AHakGameState::AHakGameState(const FObjectInitializer& ObjectInitializer)
6 : Super(ObjectInitializer)
7 {
8     ExperienceManagerComponent = CreateDefaultSubobject<UHakExperienceManagerComponent>(TEXT("ExperienceManagerComponent")); 2
9 }

```

□ 한번 해당 Callstack을 확인해보자:

□ Breakpoint가 안잡히니, PRAGMA_DISABLE_OPTIMIZATION을 넣어주자:

```

1 #include "HakGameState.h"
2 #include "HakExperienceManagerComponent.h"
3 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakGameState)
4
5 PRAGMA_DISABLE_OPTIMIZATION 1
6 AHakGameState::AHakGameState(const FObjectInitializer& ObjectInitializer)
7 : Super(ObjectInitializer)
8 {
9     ExperienceManagerComponent = CreateDefaultSubobject<UHakExperienceManagerComponent>(TEXT("ExperienceManagerComponent"));
10 }
11 PRAGMA_ENABLE_OPTIMIZATION 2

```

□ 확인해보면 앞서 시퀀서 다이어그램에서 확인했듯이, 같은 Callstack이 보인다:

```

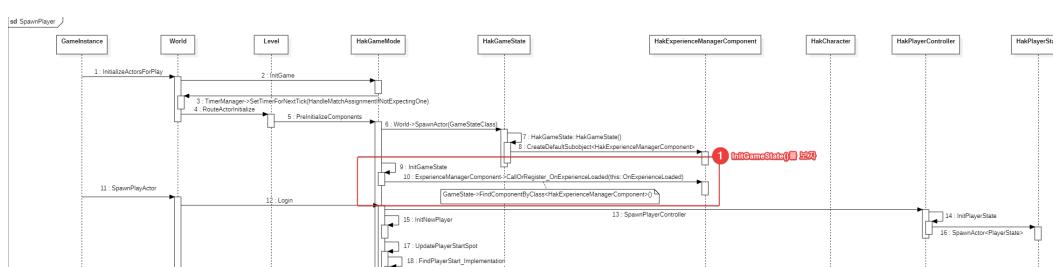
1 #include "HakGameState.h"
2 #include "HakExperienceManagerComponent.h"
3 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakGameState)
4
5 PRAGMA_DISABLE_OPTIMIZATION 1
6 AHakGameState::AHakGameState(const FObjectInitializer& ObjectInitializer)
7 : Super(ObjectInitializer)
8 {
9     ExperienceManagerComponent = CreateDefaultSubobject<UHakExperienceManagerComponent>(TEXT("ExperienceManagerComponent"));
10 }
11 PRAGMA_ENABLE_OPTIMIZATION 2

```

HakGameMode::InitGameState()

▼ 펼치기

□ 다음은 InitGameState()을 볼 차례이다:



- 앞서, 우리는 HakGameState를 준비했고, ExperienceManagerComponent를 정의했다

- 준비된 **HakGameState**와 **HakExperienceManagerComponent**를 이용해, Experience 비동기 로딩을 통한 Delegate Binding 준비를 해놓는다.

□ AHakGameMode::InitGameState()

- 아래와 같이 HakExperienceManagerComponent를 GameState에서 가져온다:

```
void AHakGameMode::InitGameState()
{
    Super::InitGameState();

    // Experience 비동기 로딩을 위한 Delegate를 준비한다;
    UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
    check(ExperienceManagerComponent);
}
```

- ExperienceManagerComponent의 Experience 로딩 관리에 대한 멤버 변수와 메서드를 먼저 구현하자:

```
#include "CoreMinimal.h"
#include "Components/GamestateComponent.h"
#include "HakExperienceManagerComponent.generated.h"

/** forward declaration */
class UHakExperienceDefinition;

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

DECLARE_MULTICAST_DELEGATE_OneParam(FOnHakExperienceLoaded, const UHakExperienceDefinition);

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

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

    /**
     * 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;
};
```

```

    /**
     * member methods
     */
    void UHakExperienceManagerComponent::CallOrRegister_OnExperienceLoaded(FOnHakExperienceLoaded::FDelegate&& Delegate)
    {
        // IsExperienceLoaded() 구현
        if (IsExperienceLoaded())
        {
            Delegate.Execute(CurrentExperience);
        }
        else
        {
            /**
             * 참고로, 여러분들은 Delegate 객체를 자세히 살펴보면, 내부적으로 필요한 변수들은 메모리 할당해놓는다:
             * TArray<int> a = {1, 2, 3, 4};
             * delegate_type delegate = [a](){
             *     return a.Num();
             * }
             * a는 delegate_type 내부에 new로 할당되어 있다. 복사 비용을 낮추기 위해 Move를 통해 하는 것을 잊지 말자!
             */
            OnExperienceLoaded.Add(MoveTemp(Delegate));
        }
    }
}

```

- Delegate에 대한 간단한 설명
- AHakGameMode 구현:

```

#pragma once

#include "CoreMinimal.h"
#include "GameFramework/GameModeBase.h"
#include "HakGameMode.generated.h"

/**
 * forward declaration
 */
class UHakExperienceDefinition; ①

/**
 * HakGameMode
 */
UCLASS()
class AHakGameMode : public AGameModeBase
{
    GENERATED_BODY()
public:
    AHakGameMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /**
     * AGameModeBase interface
     */
    virtual void InitGame(const FString& MapName, const FString& Options, FString& ErrorMessage) final;
    virtual void InitGameState() final;

    /**
     * member methods
     */
    void HandleMatchAssignmentIfNotExpectingOne();
    void OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience); ②
};

```

```

#include "HakGameMode.h"
#include "HakGameState.h"
#include "HakExperienceManagerComponent.h"
#include "HakGame/Player/HakPlayerController.h" ①
#include "HakGame/Player/HakPlayerState.h"
#include "HakGame/Character/HakCharacter.h"

#include UE_INLINE_GENERATED_CPP_BY_NAME(HakGameMode)

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();
}

/** 
 * AGameModeBase interface
 */

void AHakGameMode::InitGame(const FString& MapName, const FString& Options, FString& ErrorMessage)
{
    Super::InitGame(MapName, Options, ErrorMessage);

    // 아직 GameInstance를 통해, 초기화 작업이 진행되므로, 현 프레임에는 Lyra의 Concept의 Experience 처리를 진행할 수 없다:
    // - 이를 처리하기 위해, 한프레임 뒤에 이벤트를 받아 처리를 이어서 진행한다
    GetWorld()->GetTimerManager().SetTimerForNextTick(this, &ThisClass::HandleMatchAssignmentIfNotExpectingOne);
}

void AHakGameMode::InitGameState()
{
    Super::InitGameState();

    // Experience 버동기 로딩을 위한 Delegate를 준비한다:
    UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
    check(ExperienceManagerComponent);

    // OnExperienceLoaded 등록
    ExperienceManagerComponent->CallOrRegister_OnExperienceLoaded(FOnHakExperienceLoaded::FDelegate::CreateUObject(this, &ThisClass::OnExperienceLoaded));
}

/** 
 * member methods
 */
void AHakGameMode::HandleMatchAssignmentIfNotExpectingOne()

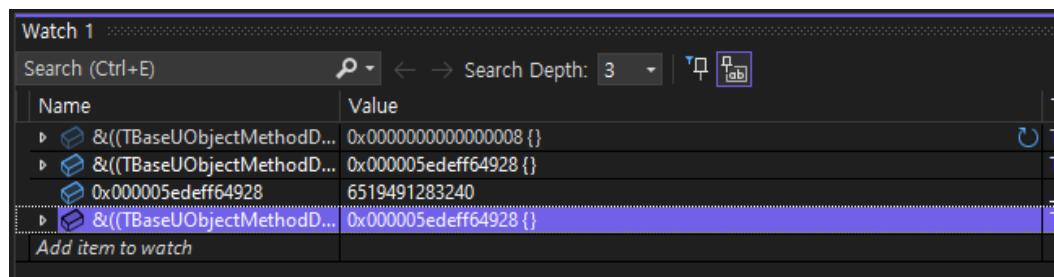
void AHakGameMode::OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience) ③

```

[번외]

- CallOrRegister_OnExperienceLoaded 디버깅을 통해 TDelegate 간단히 확인:

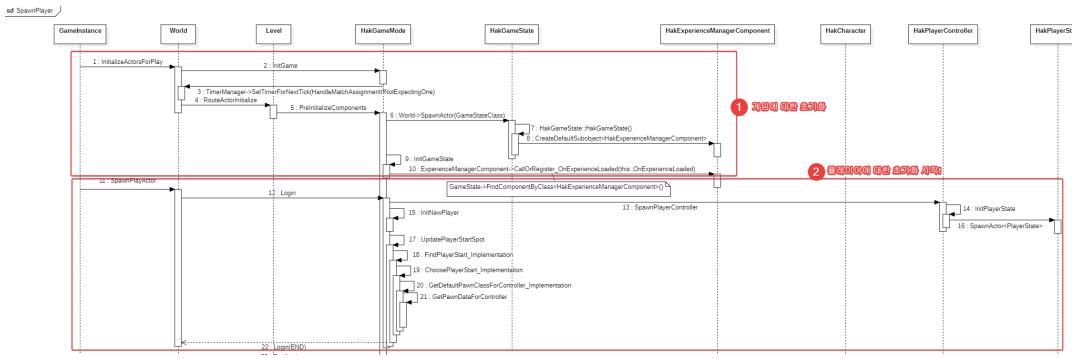
Payload의 Pointer를 이용해서 확인해보자:



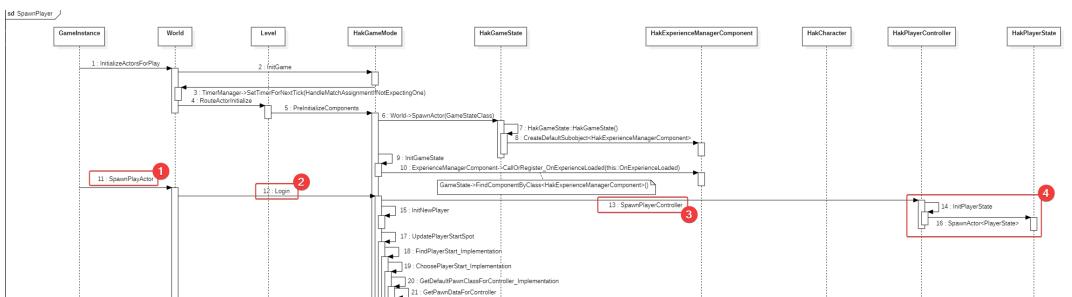
HakPlayerController와 HakPlayerState

▼ 펼치기

- 지금까지 앞서, Game에 대한 초기화 과정이었다면, 이제 Player에 대한 초기화를 진행한다:



□ 아래와 같은 순서로, GameInstance로부터 Player 관련 PlayerController와 PlayerState들을 초기화되기 시작한다:



- 여기서 우리가 주목해야 할 점은 PlayerController와 PlayerState와 관계이다
- PlayerController는 PlayerState를 생성하는 주체이다:
 - 그럼 PlayerController가 PlayerState를 가지고 있을까?

```

UCLASS(Abstract, notplaceable, NotBlueprintable, HideCategories=(Collision, Rendering, Transformation))
class ENGINE_API AController : public AActor, public INavAgentInterface
{
    GENERATED_BODY()

public:
    /* Default Constructor */
    AController(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());

    /** PlayerState containing replicated information about the player using this controller (only exists for players, not NPCs). */
    UPROPERTY(ReplicatedUsing = OnRep_PlayerState, BlueprintReadOnly, Category=Controller)
    TObjectPtr<APlayerState> PlayerState;

    /** Actor marking where this controller spawned in. */
    TWeakObjectPtr<class AActor> StartSpot;

    /** Called when the controller has instigated damage in any way */
    UPROPERTY(BlueprintAssignable)
    FOnInstigatedAnyDamageSignature OnInstigatedAnyDamage;

    /** Called on both authorities and clients when the possessed pawn changes (either OldPawn or NewPawn might be nullptr) */
    UPROPERTY(BlueprintAssignable, Category=Pawn)
    FOnPossessedPawnChanged OnPossessedPawnChanged;

    /** Current gameplay state this controller is in */
    UPROPERTY()
    FName StateName;
}

```

- 이를 어디서 볼 수 있을까?

- InitPlayerState()에서 볼 수 있다:

```

void APlayerController::PostInitializeComponents()
{
    Super::PostInitializeComponents();

    if ( IsValid(this) && (GetNetMode() != NM_Client) )
    {
        // create a new player replication info
        InitPlayerState(); ①
    }

    SpawnPlayerCameraManager();
    ResetCameraMode();

    if ( GetNetMode() == NM_Client )
    {
        SpawnDefaultHUD();
    }

    AddCheats();

    bPlayerIsWaiting = true;
    StateName = NAME_Spectating; // Don't use ChangeState, because we want to defer spawning the SpectatorPawn until the Player is received
}

```

- 정확한 흐름은 `APlayerController::PostInitializeComponents` →

`AController::InitPlayerState()`

```

void AController::InitPlayerState() ①
{
    if ( GetNetMode() != NM_Client )
    {
        UWorld* const World = GetWorld();
        const AGameModeBase* GameMode = World ? World->GetAuthGameMode() : NULL;

        // If the GameMode is null, this might be a network client that's trying to
        // record a replay. Try to use the default game mode in this case so that
        // we can still spawn a PlayerState.
        if (GameMode == NULL)
        {
            const AGameStateBase* const GameState = World ? World->GetGameState() : NULL;
            GameMode = GameState ? GameState->GetDefaultGameMode() : NULL;
        }

        if (GameMode != NULL)
        {
            FActorSpawnParameters SpawnInfo;
            SpawnInfo.Owner = this;
            SpawnInfo.Instigator = GetInstigator();
            SpawnInfo.SpawnCollisionHandlingOverride = ESpawnActorCollisionHandlingMethod::AlwaysSpawn;
            SpawnInfo.ObjectFlags |= RF_Transient; // We never want player states to save into a map

            TSubClassOf<APlayerState> PlayerStateClassToSpawn = GameMode->PlayerStateClass;
            if (PlayerStateClassToSpawn.Get() == nullptr)
            {
                UE_LOG(LogPlayerController, Log, TEXT("AController::InitPlayerState: the PlayerStateClass of game mode %s is null, falling back to APlayerState"));
                PlayerStateClassToSpawn = APlayerState::StaticClass();
            }

            PlayerState = World->SpawnActor<APlayerState>(PlayerStateClassToSpawn, SpawnInfo); ②
            // force a default player name if necessary
            if (PlayerState && PlayerState->GetPlayerName().IsEmpty())
            {
                // don't call SetPlayerName() as that will broadcast entry messages but the GameMode hasn't had a chance
                // to potentially apply a player/bot name yet

                PlayerState->SetPlayerNameInternal(GameMode->DefaultPlayerName.ToString());
            }
        }
    }
}

```

- `HakPlayerState`는 `HakPawnData`가 필요하다:

- 당연히 PlayerController가 Possess할 Character Spawn에 대한 정보가 필요하기 때문이다.
- `HakPawnData는 ExperienceDefinition에 있었다:`

```

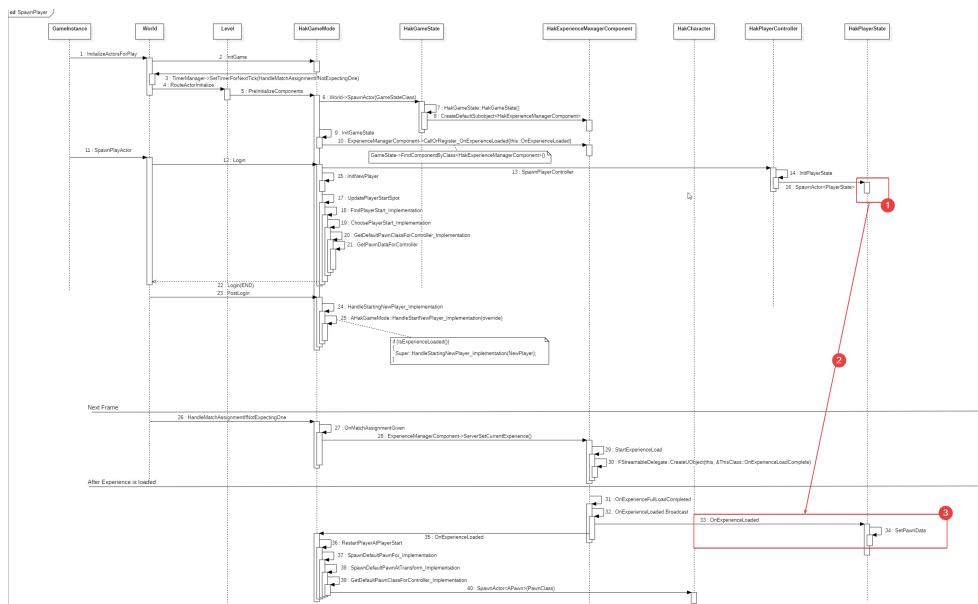
/** forward declaration */
class UHakPawnData;

/**
 * - 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 링크 및 기억용으로 남겨둘도록 하겠다:
    // - GameMode에 따라 필요한 GameFeature plugin들을 로딩하는데 이에 대한 연결고리로 생각하면 된다 (현재는 쓰지 않음)
    UPROPERTY(EditDefaultsOnly, Category=Gameplay)
    TArray< FString > GameFeaturesToEnable;
};

```

- 그럼 Experience가 로딩 완료되었을 경우, 이벤트를 받아, Experience의 DefaultPawnData를 PlayerState에 설정할 필요가 있다:

- 앞서 HakExperienceManagerComponent의 OnExperienceLoaded에 등록할 필요가 있다!



□ HakPlayerState.h/.cpp 구현:

```

1 #pragma once
2
3 #include "GameFramework/PlayerState.h"
4 #include "HakPlayerState.generated.h"
5
6 /** forward declaration */
7 class UHakPawnData;
8 class UHakExperienceDefinition; 1
9
10 UCLASS()
11 class AHakPlayerState : public APlayerState
12 {
13     GENERATED_BODY()
14 public:
15     AHakPlayerState(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
16
17     /**
18      * AActor's interface
19      */
20     virtual void PostInitializeComponents() final;
21
22     /**
23      * member methods
24      */
25     void OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience); 2
26
27     UPROPERTY()
28     TObjectPtr<const UHakPawnData> PawnData;
29 };

```

```

1 #include "HakPlayerState.h"
2 #include "HakGame/GameNodes/HakGameMode.h"
3 #include "HakGame/GameNodes/HakGameState.h"
4 #include "HakGame/GameNodes/HakExperienceManagerComponent.h"
5 #include "HakGame/GameNodes/HakExperienceDefinition.h"
6 #include UE_INLINE_GENERATED_CPP_BY_NAME(HakPlayerState) 1
7
8 AHakPlayerState::AHakPlayerState(const FObjectInitializer& ObjectInitializer)
9     : Super(ObjectInitializer)
10 {
11 }
12
13 /**
14  * AActor's interface
15 */
16 void AHakPlayerState::PostInitializeComponents()
17 {
18     Super::PostInitializeComponents();
19
20     AGameStateBase* GameState = GetWorld()->GetGameState();
21     check(GameState);
22
23     UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
24     check(ExperienceManagerComponent);
25
26     ExperienceManagerComponent->CallOrRegister_OnExperienceLoaded(FOnHakExperienceLoaded::FDelegate::CreateUObject(this, &ThisClass::OnExperienceLoaded));
27 }
28
29 /**
30  * member methods
31 */
32 void AHakPlayerState::OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience) 2
33 {
34 }

```

디버깅을 해보며 살펴보자:

우선 HakPlayerState()에 Breakpoint를 넣고 Callstack을 살펴보자:

```

Name
UnrealEditor-HakGame.dll!AHakPlayerState::AHakPlayerState(const FObjectInitializer & ObjectInitializer) Line 11
UnrealEditor-CoreUObject.dll!StaticConstructObject_Internal(const FStaticConstructObjectParameters & Params) Line 4205
[Inline Frame] UnrealEditor-Engine.dll!NewObject(UObject *) Line 1623
UnrealEditor-Engine.dll!UWorld::SpawnActor(UClass * Class, const UE::Math::TTransform<double> * UserTransformPtr, const FActorS...
UnrealEditor-Engine.dll!UWorld::SpawnActor(UClass * Class, const UE::Math::TVector<double> * Location, const UE::Math::TRotator<...
[Inline Frame] UnrealEditor-Engine.dll!UWorld::SpawnActor(UClass *) Line 3508
UnrealEditor-Engine.dll!AController::InitPlayerState() Line 620
UnrealEditor-Engine.dll!APlayerController::PostInitializeComponents() Line 1016
UnrealEditor-Engine.dll!AActor::PostActorConstruction() Line 3823
UnrealEditor-Engine.dll!AActor::FinishSpawning(const UE::Math::TTransform<double> & UserTransform, bool bIsDefaultTransform, c...
UnrealEditor-Engine.dll!UGameplayStatics::FinishSpawningActor(AActor * Actor, const UE::Math::TTransform<double> & SpawnTrans...
UnrealEditor-Engine.dll!AGameModeBase::SpawnPlayerControllerCommon(ENetRole InRemoteRole, const UE::Math::TVector<double> & Spaw...
UnrealEditor-Engine.dll!AGameModeBase::SpawnPlayerController(ENetRole InRemoteRole, const UE::Math::TVector<double> & Spaw...
UnrealEditor-Engine.dll!AGameModeBase::Login(UPlayer * NewPlayer, ENetRole InRemoteRole, const FString & Portal, const FString ...
UnrealEditor-Engine.dll!UWorld::SpawnPlayActor(UPlayer * NewPlayer, ENetRole RemoteRole, const FURL & InURL, const FURL & InNet...
UnrealEditor-Engine.dll!ULocalPlayer::SpawnPlayActor(const FString & URL, FString & OutError, UWorld * InWorld) Line 292
UnrealEditor-Engine.dll!UGameInstance::StartPlayInEditorGameInstance(ULocalPlayer * LocalPlayer, const FGamedataPIEParamet...
UnrealEditor-UnrealEd.dll!UEditorEngine::CreateInnerProcessPIEGameInstance(FRequestPlaySessionParams & InParams, const FGame...
UnrealEditor-UnrealEd.dll!UEditorEngine::OnLoginPIEComplete_Deferred(int LocalUserNum, bool bWasSuccessful, FString ErrorString, ...
UnrealEditor-UnrealEd.dll!UEditorEngine::CreateNewPlayInEditorInstance(FRequestPlaySessionParams & InRequestParams, const boo...

```

간단한 디버깅을 통해 OnExperienceLoaded에 잘 등록되었는지 확인:

```

14  /**
15  * AActor's interface
16  */
17 void AHakPlayerState::PostInitializeComponents()
18 {
19     Super::PostInitializeComponents();
20
21     AGameStateBase* GameState = GetWorld()->GetGameState();
22     check(GameState);
23
24     UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
25     check(ExperienceManagerComponent);
26
27     ExperienceManagerComponent->CallOrRegister_OnExperienceLoaded(FOnHakExperienceLoaded::FDelegate::CreateUObject(this, &ThisClass::OnExperienceLoaded));
28 }
29

```

HakPawnData와 SimpleHeroPawn 생성

▼ 펼치기

HakCharacter를 Spawn하기 위해 HakPawnData에 PawnClass를 가지도록 추가하자:

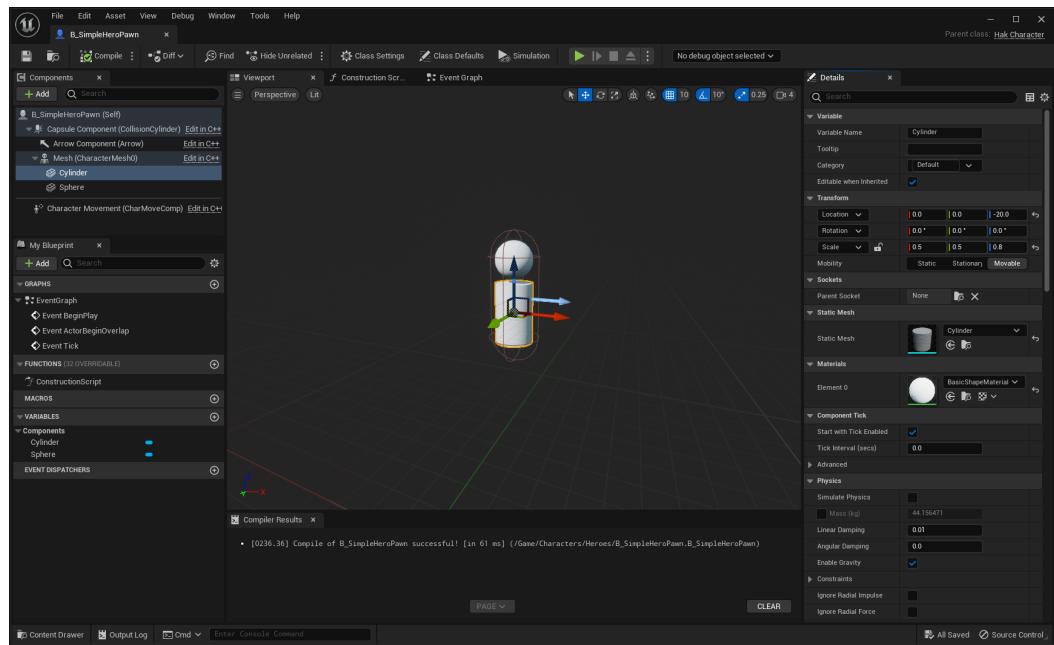
```

1 #pragma once
2
3 #include "CoreMinimal.h"
4 #include "Engine/DataAsset.h"
5 #include "HakPawnData.generated.h"
6
7 /**
8  * UHakPawnData
9  * - non-mutable data asset that contains properties used to define a pawn
10 */
11 UCLASS(BlueprintType)
12 class UHakPawnData : public UPrimaryDataAsset
13 {
14     GENERATED_BODY()
15 public:
16     UHakPawnData(const FObjectInitializer& ObjectInitializer);
17
18     /** @TODO - 일단 단순히 클래스의 형태만 만들어놓도록 하자 */
19
20     /** Pawn의 Class */
21     UPROPERTY(EditDefaultsOnly, BlueprintReadOnly, Category="Hak|Pawn")
22     TSubclassOf<APawn> PawnClass;
23 }

```

□ B_SimpleHeroPawn 생성:

- `Characters/Heroes` 폴더 생성
- HakCharacter 상속 + B_SimpleHeroPawn 이름
- Mesh 생성

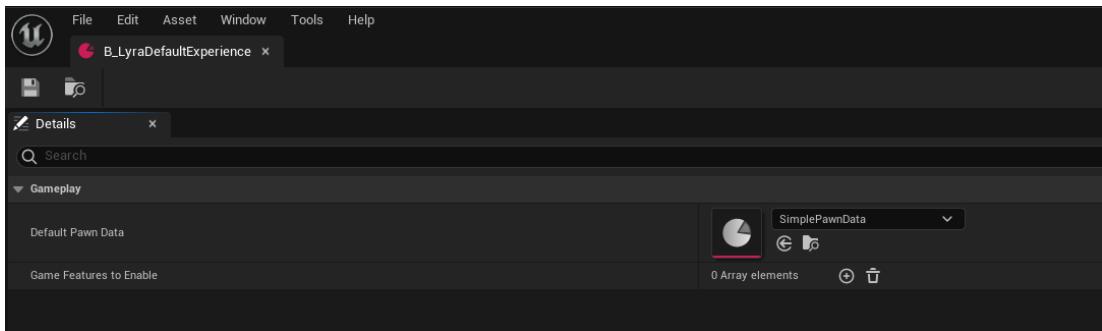


- Sphere:
 - Location: (0, 0, 50)
 - Scale (0.5, 0.5, 0.5)
- Cylinder:
 - Location: (0, 0, -20)
 - Scale: (0.5, 0.5, 0.8)

□ SimplePawnData 생성:

- `Characters/Heroes/SimplePawnData`
- HakPawnData 상속
- PawnClass에 `B_SimpleHeroPawn` 설정

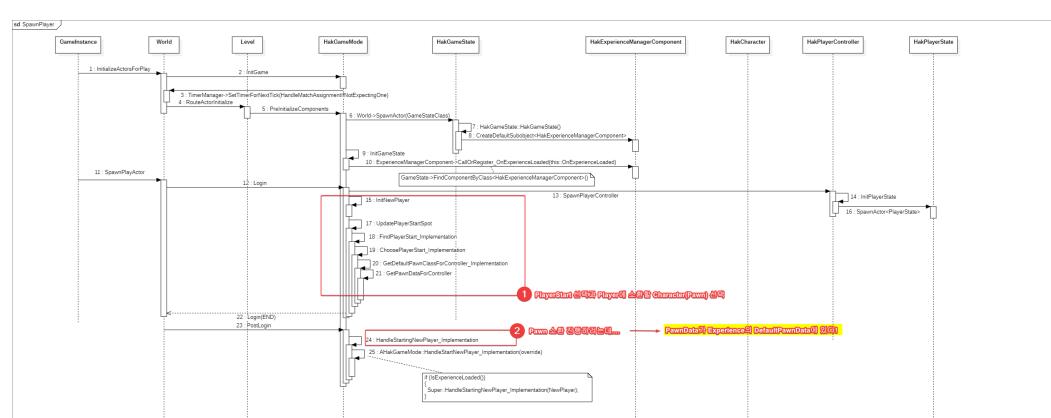
□ 생성한 SimplePawnData를 System/Experiences/B_LyraDefaultExperience에 설정:



Experience 로딩 전까지 NewPlayer 시작 막기!

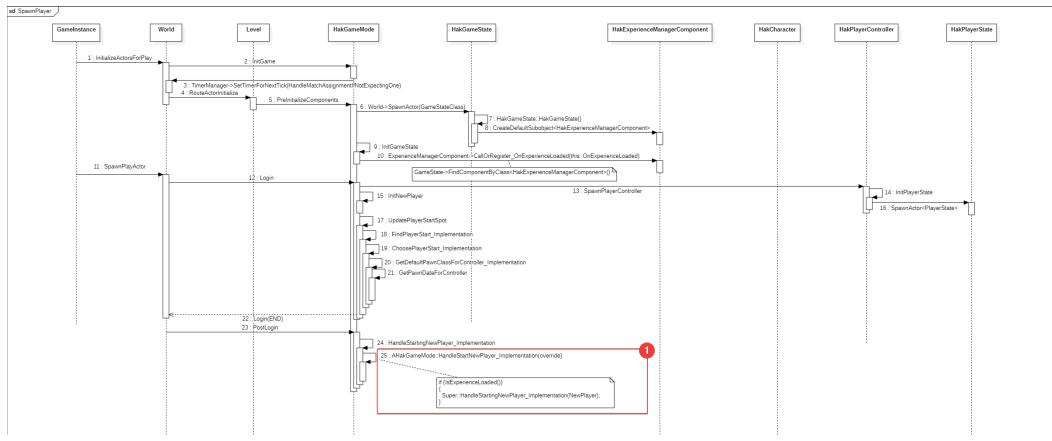
▼ 펼치기

- PlayerController와 PlayerState 생성이 끝났다면, 우리는 NewPlayer 시작할 준비가 되었다:
 - 아래와 같이 PlayerStart로 시작할 지점을 선택하고:



- 우리는 Experience가 로딩되기 전까지 NewPlayer 실행을 멈추어야 한다:

- 이를 위해, `AHalGameMode::HandleStartingNewPlayer_Implementation` 을 Override 한다:



- 실제 게임 모드에 `SpawnDefaultPawn`을 호출하는 것은 `SpawnDefaultPawnAtTransform`이다:

- 해당 함수는 아래와 같이 선언되어 있다:

```
/*
 * Called during RestartPlayer to actually spawn the player's pawn, when using a transform
 * @param NewPlayer - Controller for whom this pawn is spawned
 * @param SpawnTransform - Transform at which to spawn pawn
 * @return a pawn of the default pawn class
 */
UFUNCTION(BlueprintNativeEvent, Category=Game)
APawn* SpawnDefaultPawnAtTransform(AController* NewPlayer, const FTransform& SpawnTransform);
```

- `BlueprintNativeEvent`로 우리가 C++에서 함수를 구현할 예정이지만, 필요하다면 Blueprint에서 재정의 가능하다는 것을 의미

```
// 아래와 같은 느낌으로 이해하면 된다
UFUNCTION(BlueprintNativeEvent)
void Func();
virtual void Func_Implementation(); // [FunctionName]

// 내부적 작동 원리 (아래는 코드가 UnrealHeaderTool에서 생성
void Func()
{
    if(블루프린트에서 오버라이딩 되었다면)
        // 오버라이딩된 블루프린트 함수 호출
    else
        Func_Implementation();
}
```

- 아래와 같이 구현되어 있음:

```

1 APawn* AGameModeBase::SpawnDefaultPawnAtTransform_Implementation(AController* NewPlayer, const FTransform& SpawnTransform)
2 {
3     FActorSpawnParameters SpawnInfo;
4     SpawnInfo.Instigator = GetInstigator();
5     SpawnInfo.ObjectFlags |= RF_Transient; // We never want to save default player pawns into a map
6     UClass* PawnClass = GetDefaultPawnClassForController(NewPlayer);
7     APawn* ResultPawn = GetWorld()->SpawnActor<APawn>(PawnClass, SpawnTransform, SpawnInfo);
8     if (!ResultPawn)
9     {
10         UE_LOG(LogGameMode, Warning, TEXT("SpawnDefaultPawnAtTransform: Couldn't spawn Pawn of type %s at %s"), *GetNameSafe(PawnClass), *Sp
11     }
12     return ResultPawn;
13 }

```

- 우리는 해당 함수를 오버라이드하여, SpawnDefaultPawnAtTransform을 우리가 원하는 동작으로 변환 가능하다.

□ 우리는 이후, 해당 함수는 오버라이드 진행할 예정이다:

□ 지금은 `AHakGameMode::HandleStartingNewPlayer_Implementation` 구현이 잘 작동하는지 확인하기 위해 오버라이드를 진행하고, 해당 함수가 불리는지 확인하자:

- `SpawnDefaultPawnAtTransform_Implementation` 구현:

```

1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "GameFramework/GameModeBase.h"
7 #include "HakGameMode.generated.h"
8
9 /**
10  * Forward declaration
11 */
12 class UHakExperienceDefinition;
13
14 /**
15  * HakGameMode
16 */
17 UCLASS()
18 class AHakGameMode : public AGameModeBase
19 {
20     GENERATED_BODY()
21 public:
22     AHakGameMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
23
24     /**
25      * AGameModeBase interface
26      */
27     virtual void InitGame(const FString& MapName, const FString& Options, FString& ErrorMessage) final;
28     virtual void InitGameState() final;
29
30     /**
31      * SpawnDefaultPawnAtTransform */
32     virtual APawn* SpawnDefaultPawnAtTransform_Implementation(AController* NewPlayer, const FTransform& SpawnTransform) final; 1
33
34     /**
35      * member methods
36      */
37     void HandleMatchAssignmentIfNotExpectingOne();
38     void OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience);
39 };

```

```

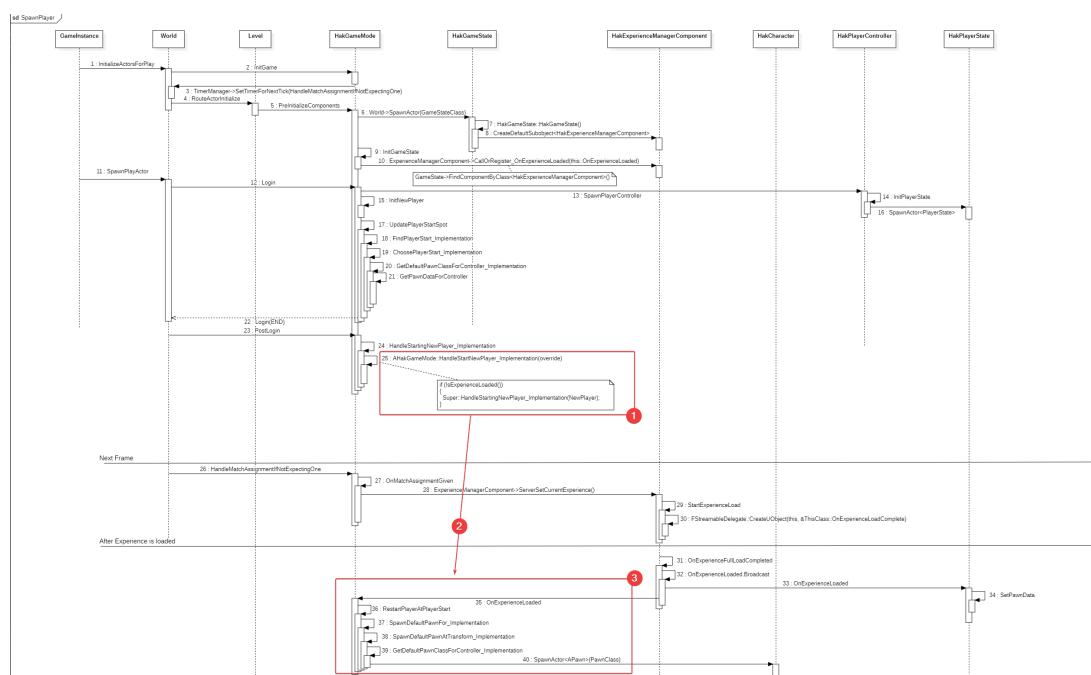
49
50     APawn* AHakGameMode::SpawnDefaultPawnAtTransform_Implementation(AController* NewPlayer, const FTransform& SpawnTransform)
51     {
52         UE_LOG(LogHak, TEXT("SpawnDefaultPawnAtTransform_Implementation is called!"));
53     }
54     return Super::SpawnDefaultPawnAtTransform_Implementation(NewPlayer, SpawnTransform);
55 }

```

□ 해당 함수가 불리는지 보자:

- 앞선 시퀀서에 의해면, PostLogin에서 불려야 한다:

- 우리는 아래와 같이 Experience 로딩 이후에 RestartPlayerAtPlayerStart를 호출하여 다시 SpawnDefaultPawnAtTransform을 호출할 예정이므로, 앞서 언급했던 HandleStartingNewPlayer_Implementation 를 오버라이드하여, PostLogin에서 진행하지 않도록 하자:



HandleStartingNewPlayer_Implementation 구현:

□ 우선 해당 함수를 오버라이딩하여 어떤 효과를 얻을 수 있는지 보자:

```

void AGameModeBase::HandleStartingNewPlayer_Implementation(APlayerController* NewPlayer)
{
    // If players should start as spectators, leave them in the spectator state
    if (!bStartPlayersAsSpectators && !MustSpectate(NewPlayer) && PlayerCanRestart(NewPlayer))
    {
        // Otherwise spawn their pawn immediately
        RestartPlayer(NewPlayer);
    }
}

```

1 RestartPlayer 호출 유무에 대한 조건문을 변환하여
우리가 원하는 흐름을 만들 수 있다!

□ 아래와 같이 구현하자:

```

/**
 * HakGameMode
 */
UCLASS()
class AHakGameMode : public AGameModeBase
{
    GENERATED_BODY()
public:
    AHakGameMode(const FObjectInitializer& ObjectInitializer = FObjectInitializer::Get());
    /**
     * AGameModeBase interface
     */
    virtual void InitGame(const FString& MapName, const FString& Options, FString& ErrorMessage) final;
    virtual void InitGameState() final;

    /** SpawnDefaultPawnAtTransform */
    virtual APawn* SpawnDefaultPawnAtTransform_Implementation(AController* NewPlayer, const FTransform& SpawnTransform) final;
    /** HandleStartingNewPlayer */
    virtual void HandleStartingNewPlayer_Implementation(APlayerController* NewPlayer) final; 1

    /**
     * member methods
     */
    void HandleMatchAssignmentIfNotExpectingOne();
    bool IsExperienceLoaded() const; 2
    void OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience);
};

```

```

50 EPawn* AHakGameMode::SpawnDefaultPawnAtTransform_Implementation(AController* NewPlayer, const FTransform& SpawnTransform)
51 {
52     UE_LOG(LogHak, Log, TEXT("SpawnDefaultPawnAtTransform_Implementation is called!"));
53     return Super::SpawnDefaultPawnAtTransform_Implementation(NewPlayer, SpawnTransform);
54 }

55 void AHakGameMode::HandleStartingNewPlayer_Implementation(APlayerController* NewPlayer) 1
56 {
57     if (IsExperienceLoaded())
58     {
59         Super::HandleStartingNewPlayer_Implementation(NewPlayer);
60     }
61 }

62 /**
63 * member methods
64 */
65 void AHakGameMode::HandleMatchAssignmentIfNotExpectingOne()
66 {
67 }

68 bool AHakGameMode::IsExperienceLoaded() const 2
69 {
70     check(GameState);
71     UHakExperienceManagerComponent* ExperienceManagerComponent = GameState->FindComponentByClass<UHakExperienceManagerComponent>();
72     check(ExperienceManagerComponent);
73     return ExperienceManagerComponent->IsExperienceLoaded();
74 }

75 void AHakGameMode::OnExperienceLoaded(const UHakExperienceDefinition* CurrentExperience)
76 {
77 }

78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95

```

□ 디버깅을 해보며 의도대로 작동하는지 확인해보자:

□ 로그가 호출되지도 않고 Breakpoint도 작동하지 않는다:

```

LogInit: AudioDevice initialized.
LogAudio: Display: Audio Device (ID: 4) registered with world 'L_DefaultEditorOverview'.
LogLoad: Game class is 'HakGameMode'
LogWorld: Bringing up level for play took: 0.000557
LogOnline: OSS: Created online subsystem instance for: :Context
DTE: Server logged in
PIE: Play in editor total start time 6.693 seconds. 1 로그가 없다

```

- 과거 로그가 남는것:

```

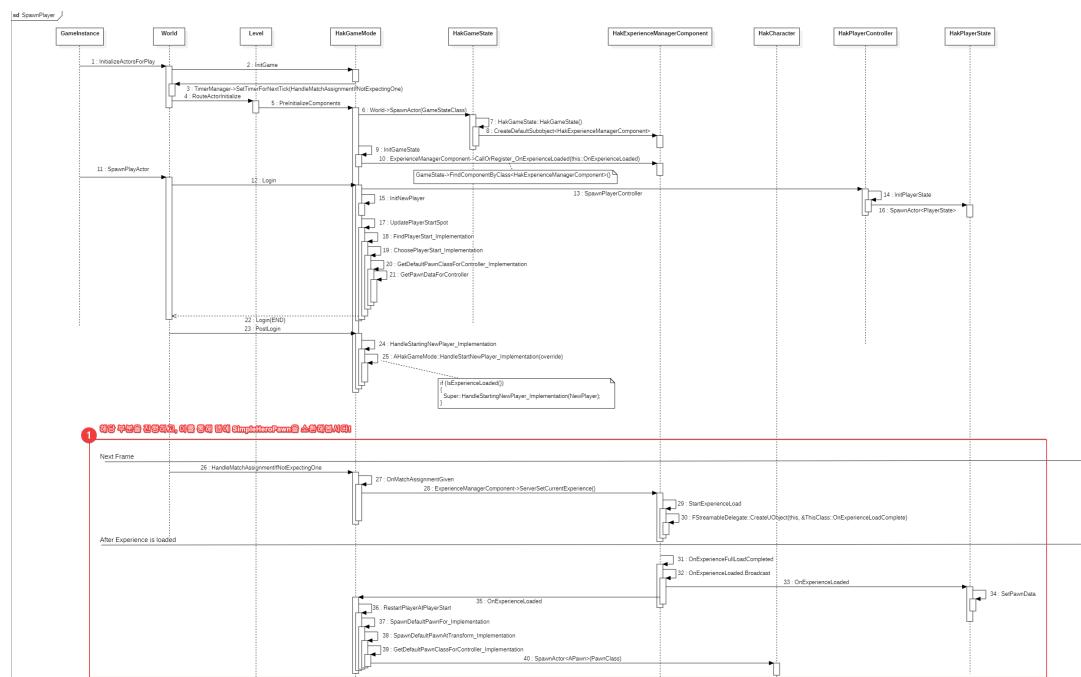
Search Log
LogLoad: Game class is 'HakGameMode'
LogWorld: Bringing World /Game/System/DefaultEditorMap/UEDPIE_0_L_DefaultEditor
LogWorld: Bringing up level for play took: 0.000755
LogOnline: OSS: Created online subsystem instance for: Context_1
LogHak: SpawnDefaultPawnAtTransform_Implementation is called!
PIE: Server logged in
PIE: Play in editor total start time 8.183 seconds.

```

다음 시간은?

▼ 펼치기

- 모든 전체적인 큰 구조에 필요한 요소들을 생성하고, 맵에 SimpleHeroPawn을 소환해보도록 합시다!



자료

▼ 펼치기

HakUserExperience.mdj