



2주차 (2023.09.18)

1주차 회고

▼ 자세히

잔소리!

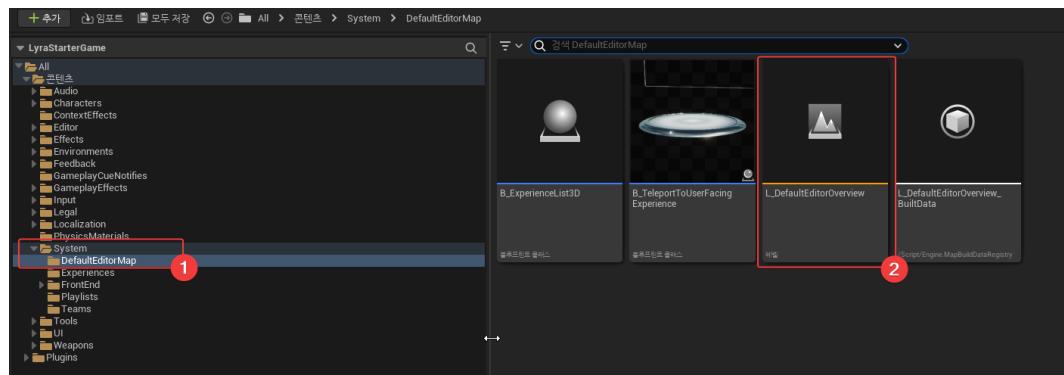
- 저번주 내용을 복습하신 분?
 - 사실 **복습은 기본!**
 - 더 나아가, **.bat** 파일들을 분석해보았어야 함:
 - 여러분들은 이미 **.bat** 파일 분석 방법을 알고 있기에!!!
- 해당 강의는 여러분들의 포트폴리오보다 엔진을 어떻게 읽고 분석하고 활용하는지 여러분의 **본질적인 실력 향상에 초점이 맞추어져 있음**
 - 이는 곧, 여러분들이 **강의 내용이 아닌** 강의에서 배운 방법으로 새로운 것을 분석하고 이해하는 힘을 가지는 것이 중요
- 여러분의 **본질적인 실력이 올라간다면, 여러분만의 포트폴리오를 어떻게 만들지 그 포트폴리오를 어떻게 엣지 있게 만들지 길이 보일 것임:**
 - 필자는 똑같은 포트폴리오보다 여러분만의 포트폴리오를 만들면 **그에 대한 피드백으로** 더 엣지있게 만드는게 더 효율적이라고 판단함
- **강의로서, 인연이 시작될거고**, 이 인연을 잘 활용하고, 필자가 강의에서 전달하고 싶은 **방법에 집중하여**, **새로운 것을 더 도전 해보았으면 함!**

DefaultEditorOverview 맵

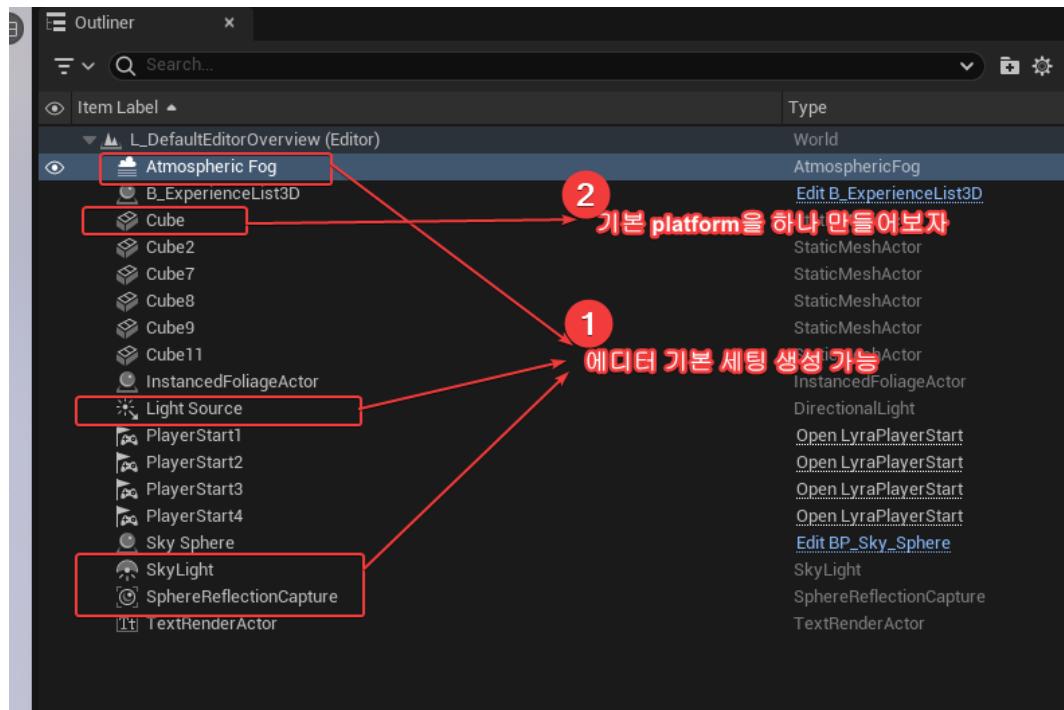
▼ 자세히

Lyra의 DefaultEditorOverview 맵 확인

- Content에서 DefaultEditorOverview 맵으로 검색하여 위치 찾기:

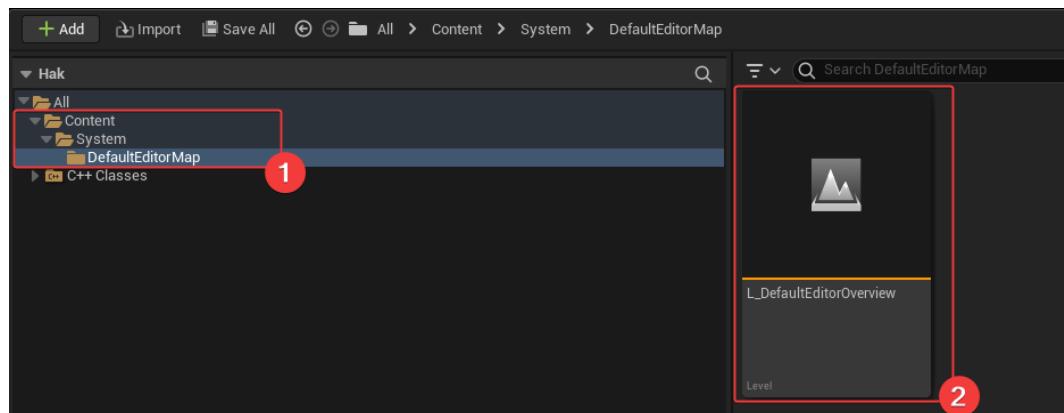


- 맵의 Outliner를 확인해보자:

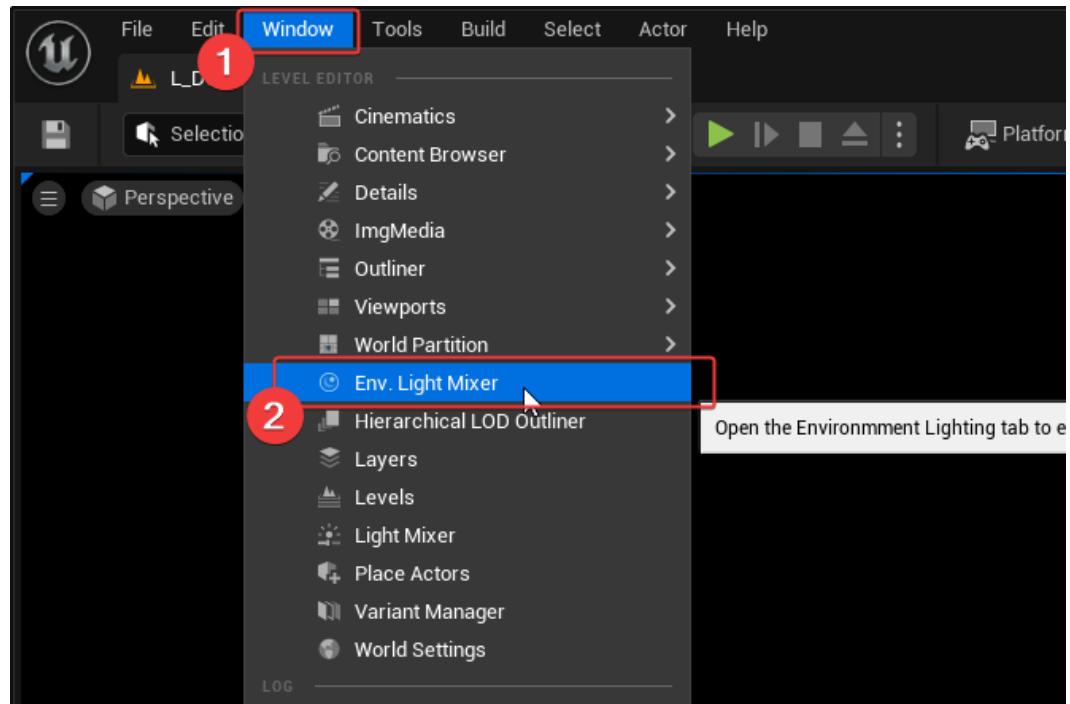


- 앞서 확인한 맵 구성요소를 하나씩 만들어보자:

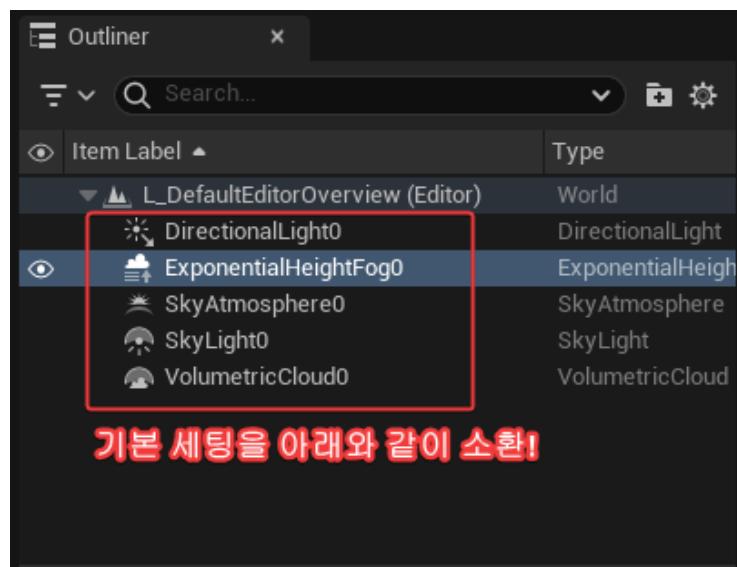
- DefaultEditorOverview와 똑같의 폴더 구조로 만들어보자:



- 앞서, 이야기했던 에디터 기본 세팅인 Atmospheric Fog, Light Source, SkyLight와 SphereReflectionCapture를 만들어보자:

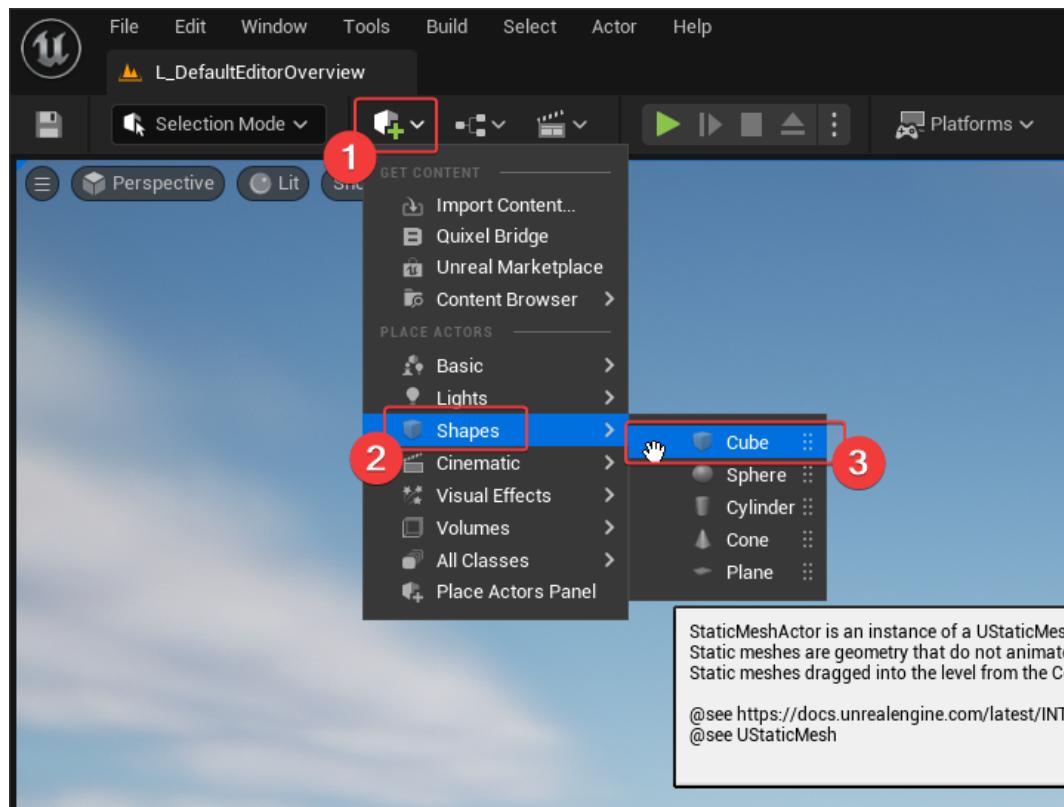


- 아래와 같이 만들어져야 함:

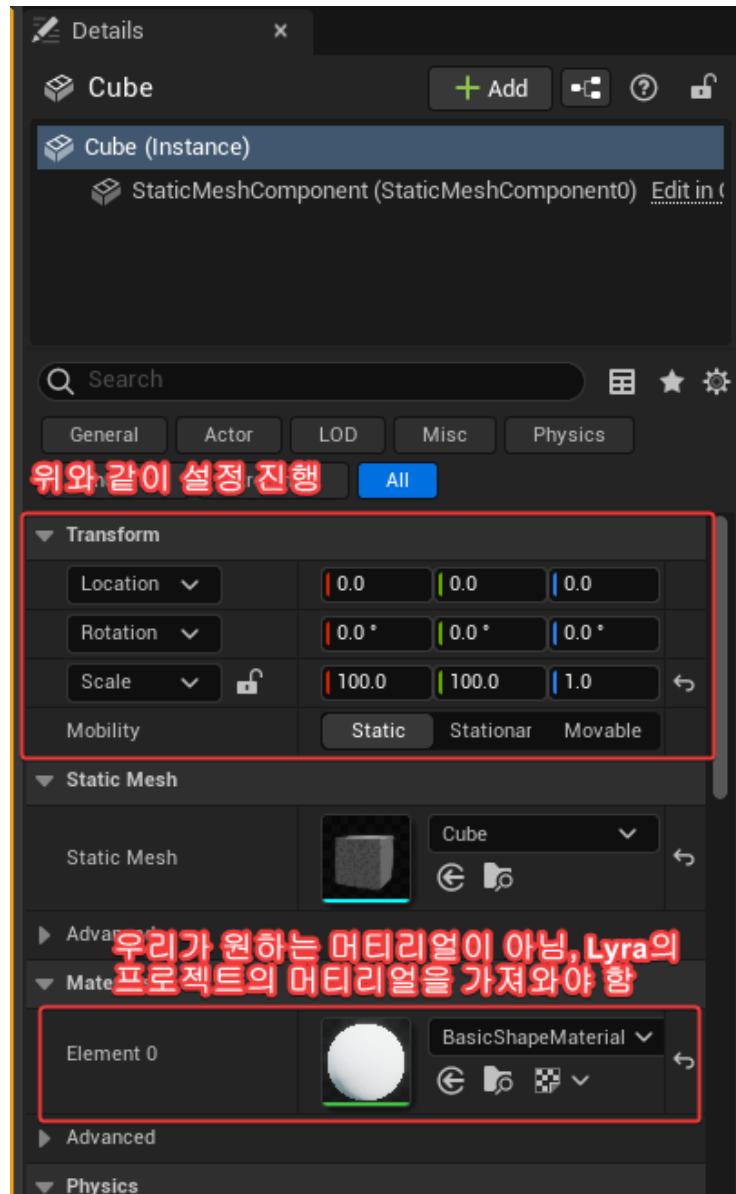


- Platform을 구성할 Cube를 생성해보자:

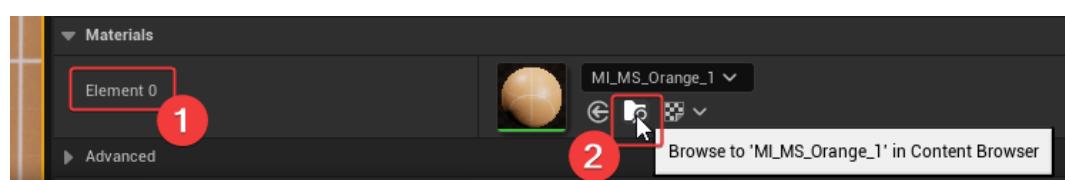
- 아래와 같이 생성 진행:



- 생성된 Cube의 속성은 아래와 같다:

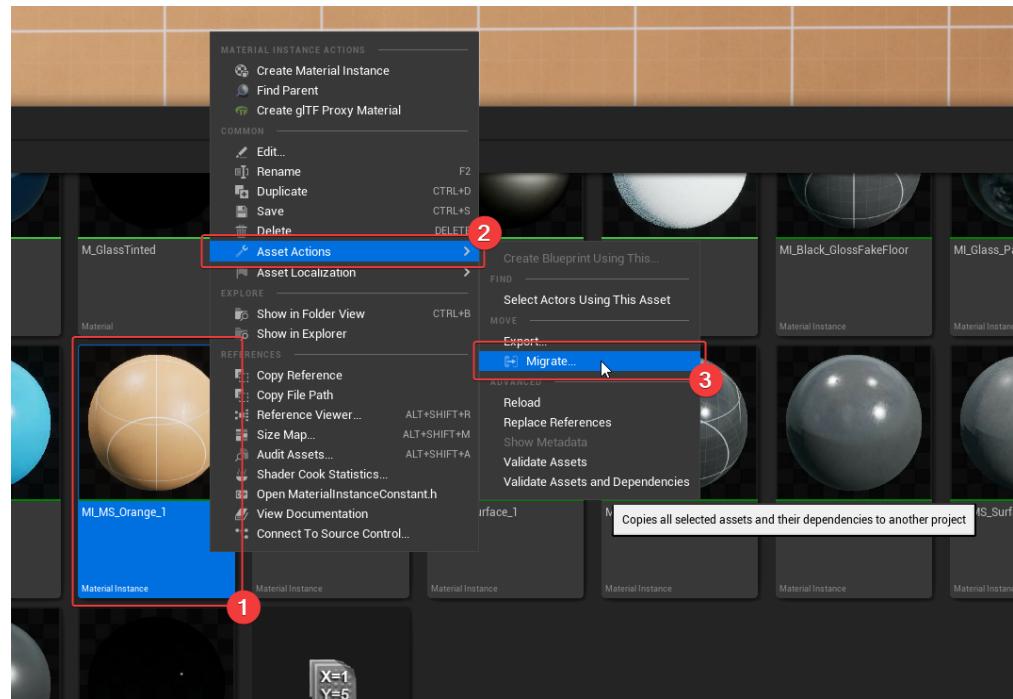


- 우리는 BasicShapeMaterial을 격자 무늬의 Material로 변경해야 한다:

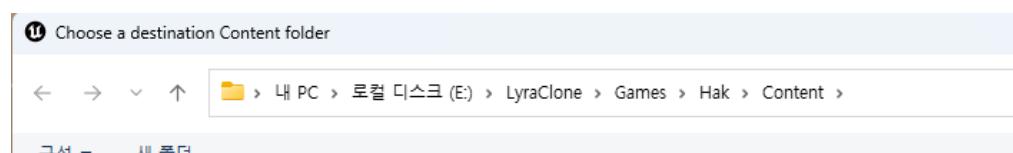
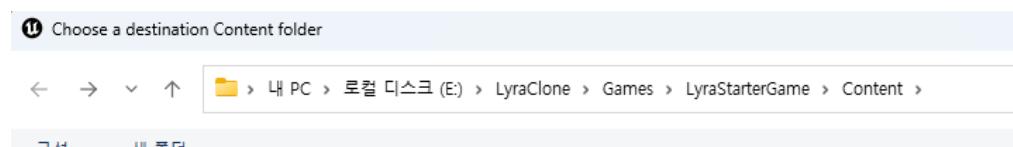


- 격자 무늬의 머티리얼은 아래와 같다:

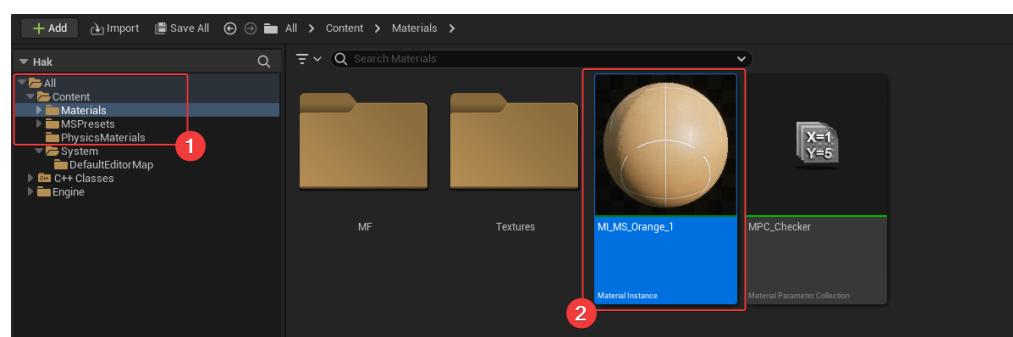
이를 Migrate를 진행해보자



- 아래의 LtraStarterGame/Content에서 Hak/Content로 변경하자:



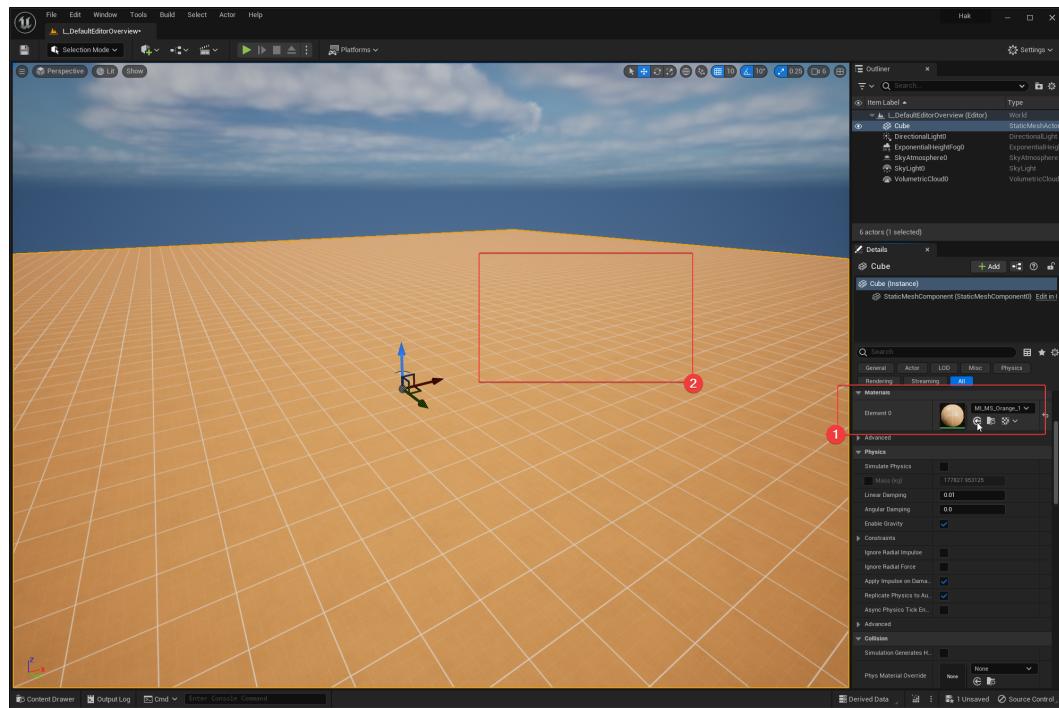
- 아래와 같이 잘 Hak로 잘 Migrate되었음을 확인할 수 있다:





우리가 앞서 풀더 구조를 최대한 맞춘다고 하였는데, 이는 여기서 보았듯이 풀더 구조를 최대한 유지시켜야 Lyra의 에셋을 쉽게 Migrate 가능하기 때문이다!~

- Migrate된 Material을 활용하여, Platform에 세팅해주자:



이제 uasset과 umap도 생성하였으니, 이를 VCS(Version Control System)을 활용하여, 어떻게 관리할지 고민해보자.

Git과 GitLFS 설정

▼ 자세히

Git과 GitLFS 설정

- Git과 GitLFS에 대한 간단한 설명:
 - Git 은 Version Control System(VCS)의 대표적인 예시
 - 그렇다면, Git을 이용한 exe 파일이나 dll 파일과 같은 바이너리 파일을 이를 통해 관리하면, 매우 비효율적이다:
 - 사실상 diff에 의미가 없기 때문!
 - Commit에 따른 파일의 History를 관리하게 만들어진 것이 GitLFS이다:
 - 쉽게 생각하면, diff가 의미없는 바이너리 파일을 관리하기 위해 만든 것이 GitLFS라고 기억하면 된다.

- Unreal Engine에는 대표적으로 uasset과 umap 파일을 바이너리 파일로 관리할 필요가 있다:
 - 이를 GitLFS로 관리해보자
- GitLFS에 대한 간단한 정리:
 - GitLFS는 사용하고자 하는 repo에 관리할 파일을 track을 추가해야 한다:

```
git lfs track {파일 경로}
```

- GitLFS는 Git과 같이 staged/unstaged의 상태가 있다:

- add해둔 파일을 lfs로 관리해야하는 상황이라면 `git rm --cache`로 먼저 unstaged한 후, 다시 `git lfs track` 해야 한다.

```
git rm --cached {파일 경로}
```

- 관리할 경로를 제거하기 위해서는 아래와 같이 untrack을 활용하면 된다:

```
git lfs untrack {파일 경로}
```

- 위의 모든 명령어의 결과물은 `.gitattribute` 파일에 있다:

```

.gitattributes x 1
E: > LyraClone > ◆ gitl Follow link (ctrl + click)
1   # refer to https://forums.unrealengine.com/t/official-git-lfs-tracking-list-for-ue4/131597/7
2
3   ## Unreal Engine ##
4   ## Auto detect text files and perform LF normalization ##
5
6   * text=auto
7
8   # UE file types
9   *.uasset filter=lfs diff=lfs merge=lfs -text
10  *.umap filter=lfs diff=lfs merge=lfs -text
11  *.udk filter=lfs diff=lfs merge=lfs -text
12  *.upk filter=lfs diff=lfs merge=lfs -text
13
14 -----
15
  
```

- 간단히 아래의 명령어를 통해 실습해보기 및 `.gitattribute` 파일 확인:

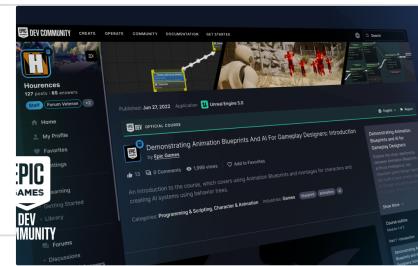
```
git lfs track *.hak
git lfs untrack *.hak
```

- 필자는 `.gitattribute` 파일을 아래의 경로에서 참고했다:

Official Git LFS Tracking List For UE4

Hello, Is this still the correct way to manage unreal projects with git? Is there a quicker/easier way to do this than adding each file type one at a time using the console, because this

 <https://forums.unrealengine.com/t/official-git-lfs-tracking-list-for-ue4/131597/7>



참고 레퍼런스:

Git LFS (Large File Storage) 사용하기

Git의 용량제한과 LFS기본적으로 git은 여러개의 작은 소스코드 파일들을 위한 버전 컨트롤 시스템(VCS)이다. 따라서 Github의 경우 50Mb부터 Warning이 표시되고, 100Mb부터는 push시 Error가 발

 <https://newsight.tistory.com/330>



Git의 `.gitignore` 파일을 활용하여, 우리의 프로젝트를 원하는 형태대로 제어해보자:

- 우선 `.gitignore` 파일에 대해 간단히 설명하면 아래와 같다:
 - VCS를 활용하여, **특정 경로 혹은 특정 파일에 대해 파일의 버전을 관리하고 싶은 않은 리스트** 가 있을 것인데, 이를 **등록할 수 있는 기능**이라고 생각하면 된다.
- **UnrealEngine**의 예시를 보면서 이해해보자
 - 우리는 현재 프로젝트를 진행하면서, 아래의 목적을 가지고 있다:
 1. **LyraStarterGame**은 Clone의 대상으로 버전 관리가 필요 **없다**
 2. Clone이 대상이 되는 프로젝트만 관리되면 된다
 - 기본적으로 Unreal의 `.gitignore`에는 **uasset/umap과 같은 파일이 무시되도록 설정되어있고 이를 예외 처리해야 한다**
 - 또, **Content 경로**에 대해서 예외 처리가 되어있고 이를 해제할 필요가 있다
 - 하나씩, 위의 조건을 만족시키도록 하나씩 진행해보자:
 - 그 중에 우선 현재 우리 프로젝트를 추가하여, LyraStarterGame이 문제가 되는지 확인해보자:

```
git add Game/Hak
```

```

tkdgu@hsh6679-h MINGW64 /e/LyraClone (LyraClone-2_week)
$ git status
On branch LyraClone-2_week
Your branch is up to date with 'origin/LyraClone-2_week'.

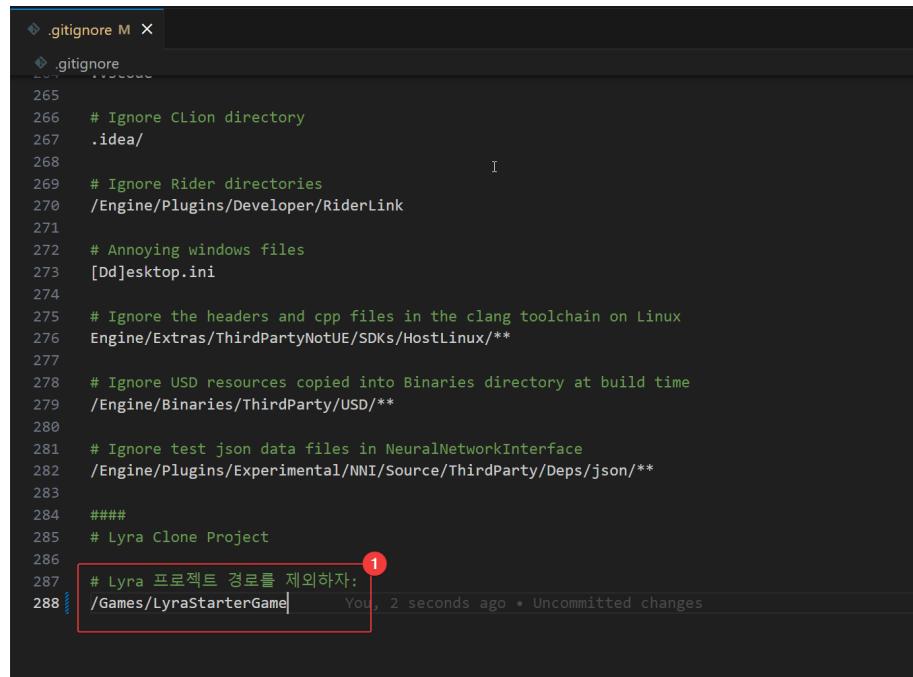
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file: Games/Hak/Config/DefaultEditor.ini
    new file: Games/Hak/Config/DefaultEngine.ini
    new file: Games/Hak/Config/DefaultGame.ini
    new file: Games/Hak/Config/DefaultInput.ini
    new file: Games/Hak/Hak.uproject
    new file: Games/Hak/Source/Hak.Target.cs
    new file: Games/Hak/Source/Hak/Hak.Build.cs
    new file: Games/Hak/Source/Hak/Hak.cpp
    new file: Games/Hak/Source/Hak/Hak.h
    new file: Games/Hak/Source/Hak/HakGameModeBase.cpp
    new file: Games/Hak/Source/Hak/HakGameModeBase.h
    new file: Games/Hak/Source/HakEditor.Target.cs

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Games/LyraStarterGame/

```

- 위와 같이 LyraStarterGame이 추가되어 있음을 볼 수가 있다:

- 이를 제외하기 위해 아래와 같이 `.gitignore`에서 **LyraStarterGame을 제외해보자:**



```

. .gitignore M X
. .gitignore
265
266 # Ignore CLion directory
267 .idea/
268
269 # Ignore Rider directories
270 /Engine/Plugins/Developer/RiderLink
271
272 # Annoying windows files
273 [Dd]esktop.ini
274
275 # Ignore the headers and cpp files in the clang toolchain on Linux
276 Engine/Extras/ThirdPartyNotUE/SDKs/HostLinux/**
277
278 # Ignore USD resources copied into Binaries directory at build time
279 /Engine/Binaries/ThirdParty/USD/**
280
281 # Ignore test json data files in NeuralNetworkInterface
282 /Engine/Plugins/Experimental>NNI/Source/ThirdParty/Deps/json/**
283
284 #####
285 # Lyra Clone Project
286
287 # Lyra 프로젝트 경로를 제외하자:
288 /Games/LyraStarterGame| You, 2 seconds ago * Uncommitted changes

```

- 아래와 같이 `.gitignore`를 수정되며, LyraStarterGame이 제외되었음을 확인할 수 있다:

```

tkdgu@hsh6679-h MINGW64 /e/LyraClone (LyraClone-2_week)
$ git status
On branch LyraClone-2_week
Your branch is ahead of 'origin/LyraClone-2_week' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

```

- ✓ .uasset과 .umap과 같은 gitfs로 버전관리할 에셋을 등록해주자:

- `.gitignore` 파일 추가하장:

```

# 언리얼 에셋 확장자 예외 처리 진행
!*.uasset
!*.umap

```

- 아직 우리가 추가한 DefaultEditorOverview.umap이 보이지 않는다:

```

tkdgu@hsh6679-h MINGW64 /e/LyraClone (LyraClone-2_week)
$ git status
On branch LyraClone-2_week
Your branch is ahead of 'origin/LyraClone-2_week' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   .gitignore

no changes added to commit (use "git add" and/or "git commit -a")

```

- 혹시, Root인 Content 폴더가 예외처리되어 있는가 확인해보자:

```

233 # Ignore exp files in Engine and Plugin Binaries directories as they aren't C/C++ source files
234 /Engine/**/Binaries/**/*.exp
235
236 # Ignore temporary code folders generated by live coding
237 enc_temp_folder/
238
239 # Ignore content folders
240 Content/ Ben Marsh, 8 years ago * Switch Git repository to whitelist repository file...
241
242 # Ignore DDC
243 /Engine/DerivedDataCache/**
244

```

- 그럼, 우리 게임 프로젝트에 대한 Content 폴더를 예외처리해주자:

```

#####
# Lyra Clone Project

# Lyra 프로젝트 경로를 제외하자:
/Games/LyraStarterGame

# 언리얼 에셋 확장자 예외 처리 진행
!*.uasset

```

```
!* .umap

# Content 폴더 예외 처리
!/Games/Hak/Content/
```

- 아래와 같이 에셋이 보여짐을 확인할 수 있다:

```
tkdgu@hsh6679-h MINGW64 /e/LyraClone (LyraClone-2_week)
$ git status
On branch LyraClone-2_week
Your branch is ahead of 'origin/LyraClone-2_week' by 1 commit.
  (use "git push" to publish your local commits)

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Games/Hak/Content/

no changes added to commit (use "git add" and/or "git commit -a")
```

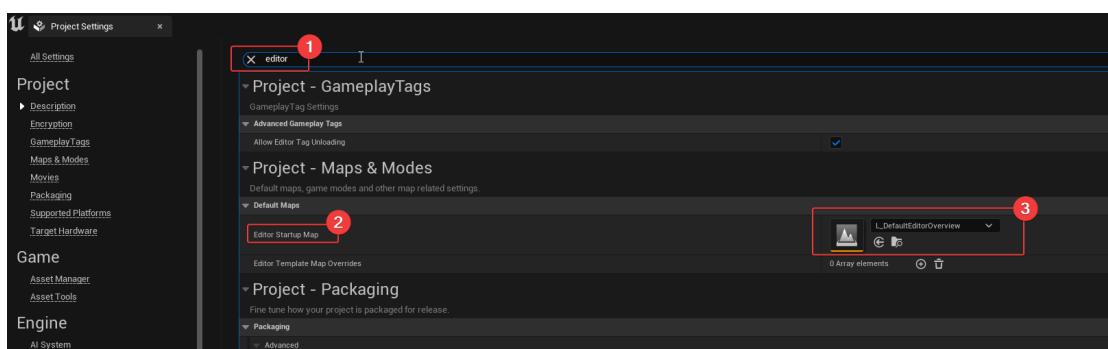
```
On branch LyraClone-2_week
Your branch is ahead of 'origin/LyraClone-2_week' by 1 commit.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   .gitignore
    new file:  Games/Hak/Content/MSPresets/MSTextures/BlackPlaceholder.usset
    new file:  Games/Hak/Content/MSPresets/MSTextures/Placeholder_Normal.usset
    new file:  Games/Hak/Content/MSPresets/MSTextures/WhitePlaceholder.usset
    new file:  Games/Hak/Content/MSPresets/M_MS_Surface_Material/Functions/MF_MapAdjustments.usset
    new file:  Games/Hak/Content/MSPresets/M_MS_Surface_Material/Functions/MF_Tiling.usset
    new file:  Games/Hak/Content/MSPresets/M_MS_Surface_Material/M_MS_Surface_Material_TriPlanar.usset
    new file:  Games/Hak/Content/Materials/MF/MF_2DGrid.usset
    new file:  Games/Hak/Content/Materials/MF/MF_TripleGrid.usset
    new file:  Games/Hak/Content/Materials/MF/MF_WorldGrid.usset
    new file:  Games/Hak/Content/Materials/MF/WorldAlignedTextureMip.usset
    new file:  Games/Hak/Content/Materials/MI_MS_Orange_1.usset
    new file:  Games/Hak/Content/Materials/MPC_Checker.usset
    new file:  Games/Hak/Content/Materials/Textures/T_Paint_Diffuse.usset
    new file:  Games/Hak/Content/Materials/Textures/T_Paint_Glossiness.usset
    new file:  Games/Hak/Content/Materials/Textures/T_Paint_Normal.usset
    new file:  Games/Hak/Content/PhysicsMaterials/PM_Concrete.usset
    new file:  Games/Hak/Content/System/DefaultEditorMap/L_DefaultEditorOverview.umap 1
```

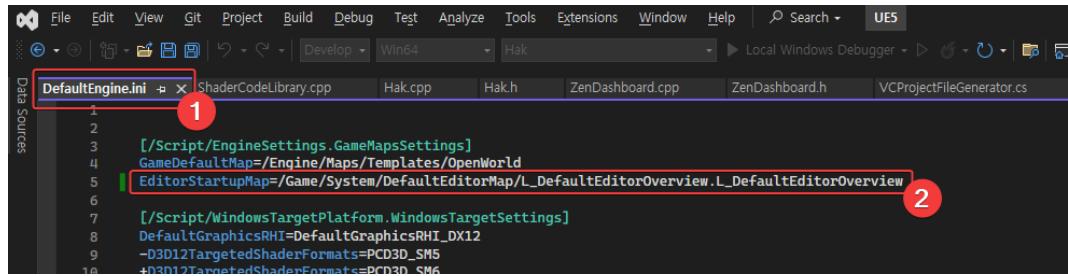
DefaultEditorOverview 맵

▼ 자세히

- 우선, Editor의 Default 맵을 L_DefaultEditorOverview로 설정해보자:



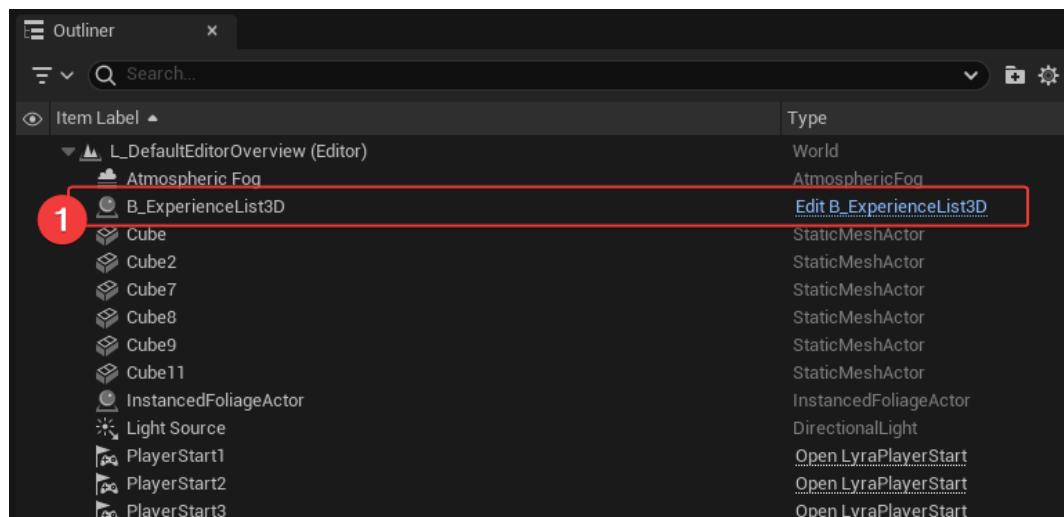
□ 그리고 DefaultEngine.ini를 확인해보자:



```
1
2
3 [ /Script/EngineSettings.GameMapsSettings ]
4 GameDefaultMap=/Engine/Maps/Templates/OpenWorld
5 EditorStartupMap=/Game/System/DefaultEditorMap/_DefaultEditorOverview.L_DefaultEditorOverview
6
7 [ /Script/WindowsTargetPlatform.WindowsTargetSettings ]
8 DefaultGraphicsRHI=DefaultGraphicsRHI_DX12
9 -D3D12TargetedShaderFormats=PCD3D_SMS
10 +D3D12TargetedShaderFormats=PCD3D_SMS
```

- ✓ LyraStarterGame의 L_DefaultEditorOverview를 열어 다음 목표를 살펴보자:

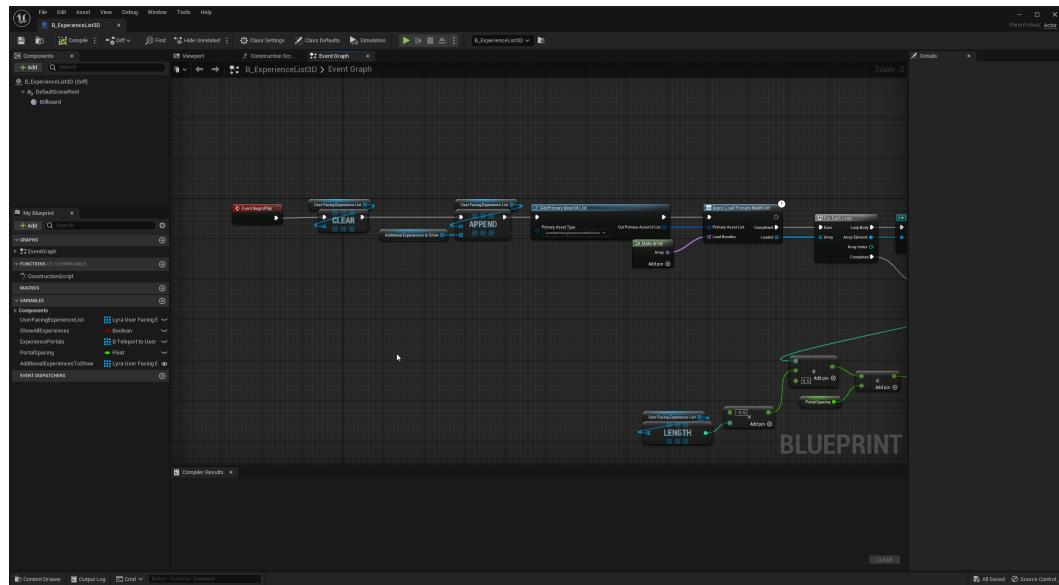
- B_ExperienceList3D를 Clone하는 것이 다음 목표:



- BP의 노드를 기억하는 것은 쉽지 않다...



사실 지금 형태의 VS Solution 구조는 코드에 대한 Clone은 편한 구조로 되어 있으나, BP는 Clone하기 쉽지 않다...

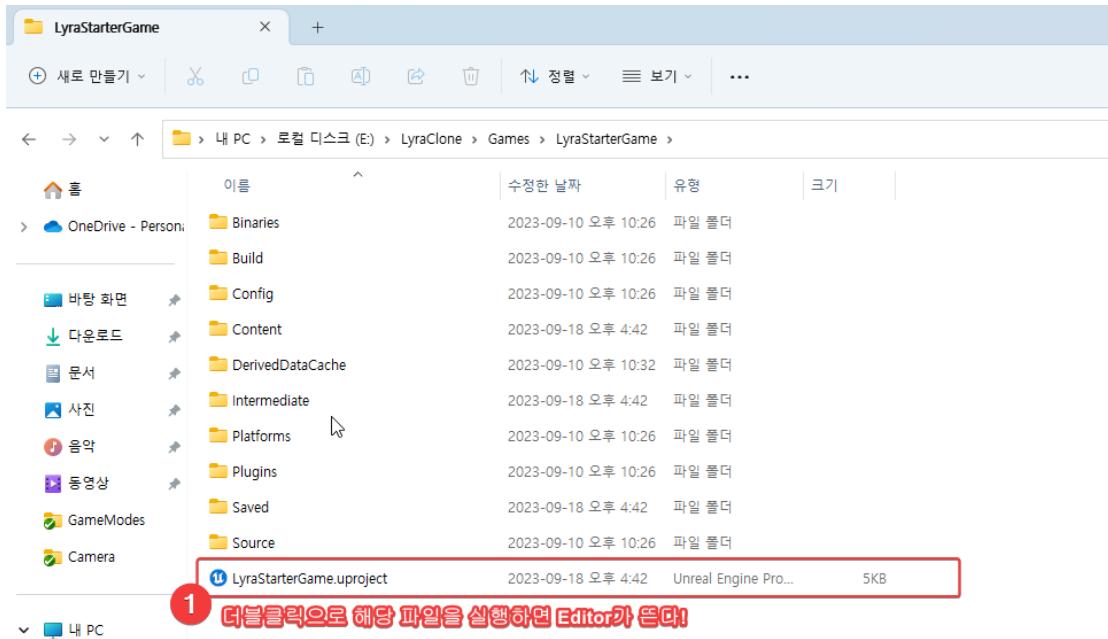


- 그렇다고, 이렇게 Screenshot을 찍어서.. 보기에는 불편하다...

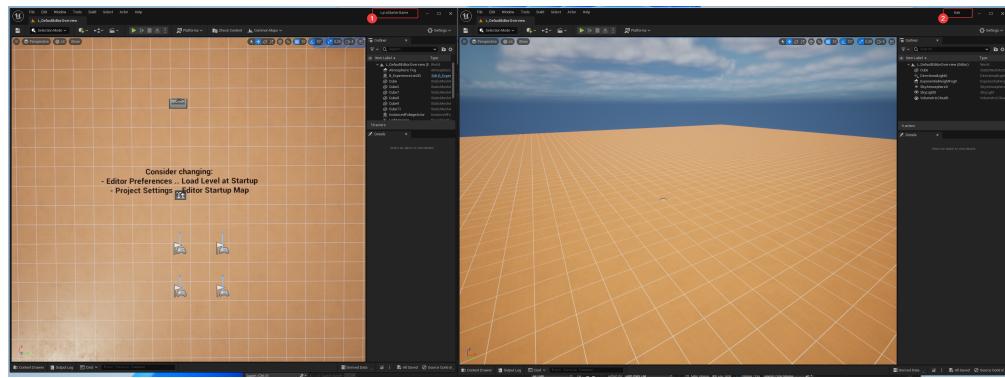
B_ExperienceList3D

▼ 자세히

- LyraStarterGame의 에디터를 또 다른 프로세스로서 실행하여, BP Clone을 손쉽게 환경세팅을 만들어보자.



- 그리고 Hak 프로젝트는 VS로 실행하자:
 - 그럼 아래와 같이 LyraStarterGame과 Hak의 에디터 두 개를 띄울 수 있다:



- 여기서 잠깐? 이게 어떻게 가능할까?

만약 여러분들 중 아래의 질문을 할 수 있다면, 신입으로써 높은 수준이 있다고 생각할 수 있습니당!



허나, 이미 듣고 나서 아는 것은 크게 의미 없죠...

항상 왜? 이게 가능할까? 가능하다면 왜 되는거지? 이런 의문을 가져보도록 합시다! (물론 **과유불급!** → 여러분이 지금은 이해할 수 없을 수도 있습니다)

▼ 정답 숨기기

현재 Hak와 LyraStarterGame은 같은 Engine을 공유하고 있고, 이는 dll을 공유하고 있을건데 어떻게 두 개의 프로세스가 같은 dll을 로딩할 수 있는걸까요?

▼ 정답

- 파일과 메모리의 차이
- dll 파일 → dll 메모리로 안착:
 - dll이란?

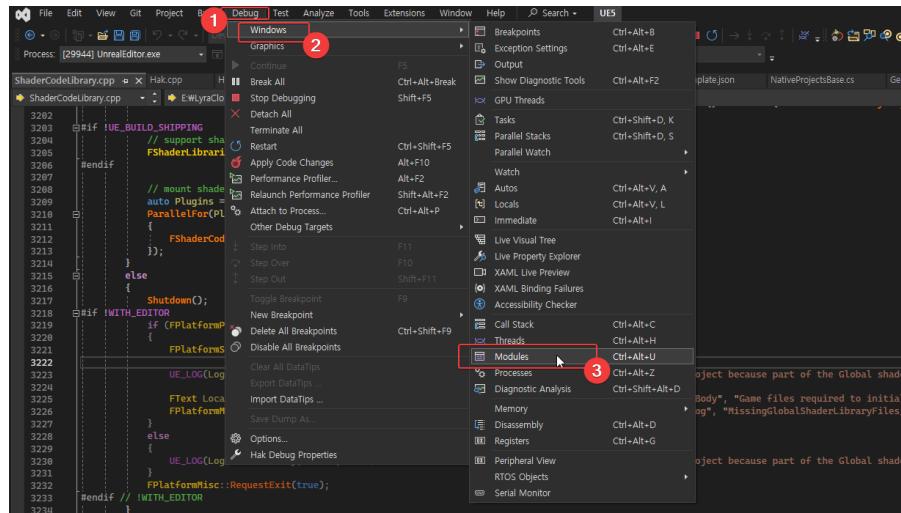
dynamic link library:

■ MS에만 존재:

- 물론 linux에서는 prx와 같은 다른 타입으로 똑같은 컨셉이 존재

■ static library와의 차이 공부하기!

- 각 프로세스는 각각의 메모리 영역을 가지고 있음:
 - 스레드 vs 프로세스의 가장 큰 차이
- VS Module Window를 이용하여, 확인해보자:



Hak 프로젝트:

Modules	core.dll	Path	Optimized	User Code	Symbol Status	Symbol File	Version	Timestamp	Address	Process	AppDomain
UnrealEditor	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	8	5.01.1.0	2023-09-10 오 .. 00007FFB03CC0000-00007FFBD4FBC000	[29944] UnrealEditor.exe	[29944]
UnrealEditor-Signature	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	12	5.01.1.0	2023-09-10 오 .. 00007FFB03CC0000-00007FFBD4FBC000	[29944] UnrealEditor.exe	[29944]
UnrealEditor-RenderCore	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	26	5.01.1.0	2023-09-10 오 .. 00007FFC3F80000-00007FFC3C396000	[29944] UnrealEditor.exe	[29944]
UnrealEditor-ApplicationCore	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	30	5.01.1.0	2023-09-10 오 .. 00007FFC3C80000-00007FFC2CC21000	[29944] UnrealEditor.exe	[29944]

LyraStarterGame 프로젝트:

Modules	core.dll	Path	Optimized	User Code	Symbol Status	Symbol File	Version	Timestamp	Address	Process	AppDomain
UnrealEditor	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	12	5.01.1.0	2023-09-10 오 .. 00007FFB03CC0000-00007FFBD4FBC000	[39348] UnrealEditor.exe	[39348]
UnrealEditor-Signature	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	14	5.01.1.0	2023-09-10 오 .. 00007FFC570000-00007FFC5B77000	[39348] UnrealEditor.exe	[39348]
UnrealEditor-RenderCore	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	27	5.01.1.0	2023-09-10 오 .. 00007FFC3B0000-00007FFC2C396000	[39348] UnrealEditor.exe	[39348]
UnrealEditor-ApplicationCore	core.dll	E:\LyraClone\Engine@InnoSw..	N/A	Yes	Symbols loaded.	E:\LyraClone\Engine@B..	29	5.01.1.0	2023-09-10 오 .. 00007FFC3C80000-00007FFC2CC21000	[39348] UnrealEditor.exe	[39348]

어? Address 영역이 같다!

- 저 Address는 **Virtual Memory Address**이다:

■ 각각의 프로세스는 같은 Virtual Memroy Address 영역을 가질 수 있다

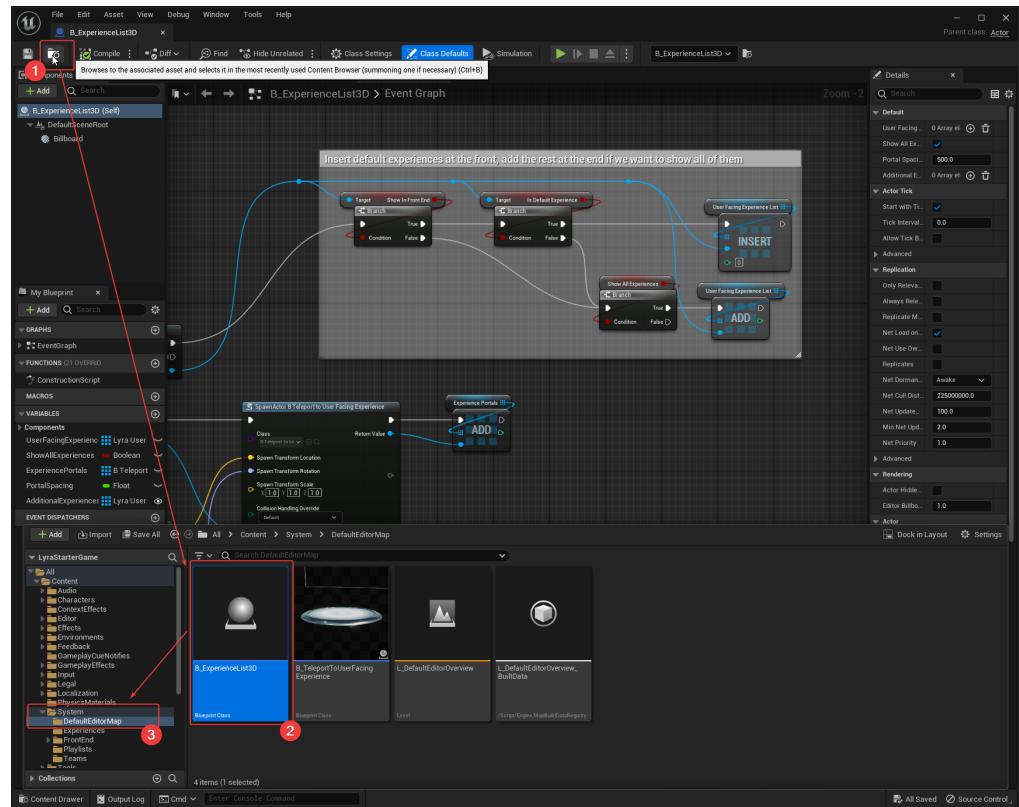
■ 허나, Physical Memory Address는 다르게 매핑되어 있다

- 이는 프로그램이 같고, **DLL을 호출하는 순서가 같기에 메모리 할당 주소 영역도 똑같이 매핑**되었다.

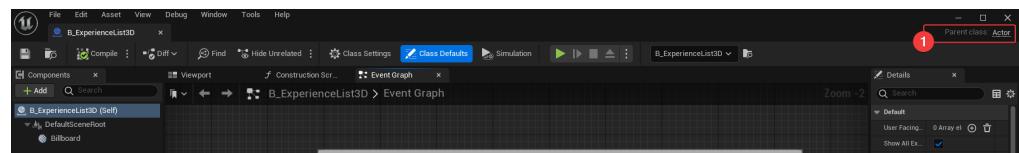
BP부터 Clone 해보자 (자신있게!)

B_ExperienceList3D를 살펴보자:

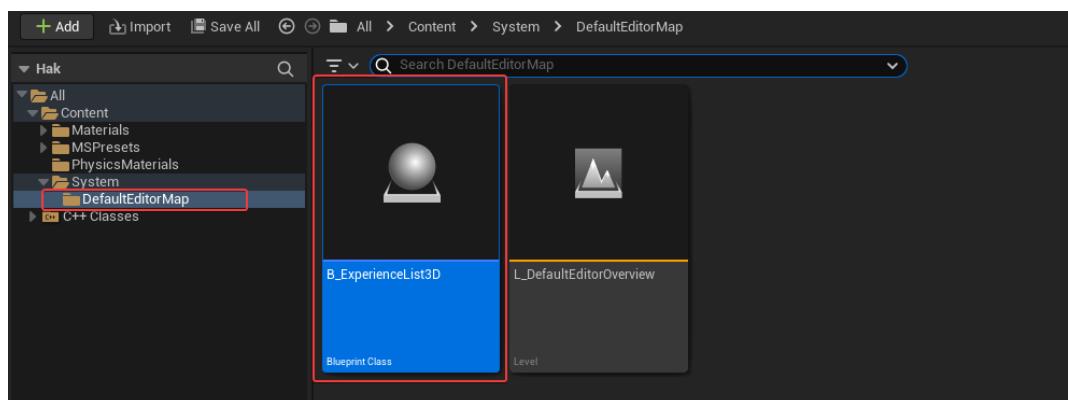
- 우선 에셋의 위치는?



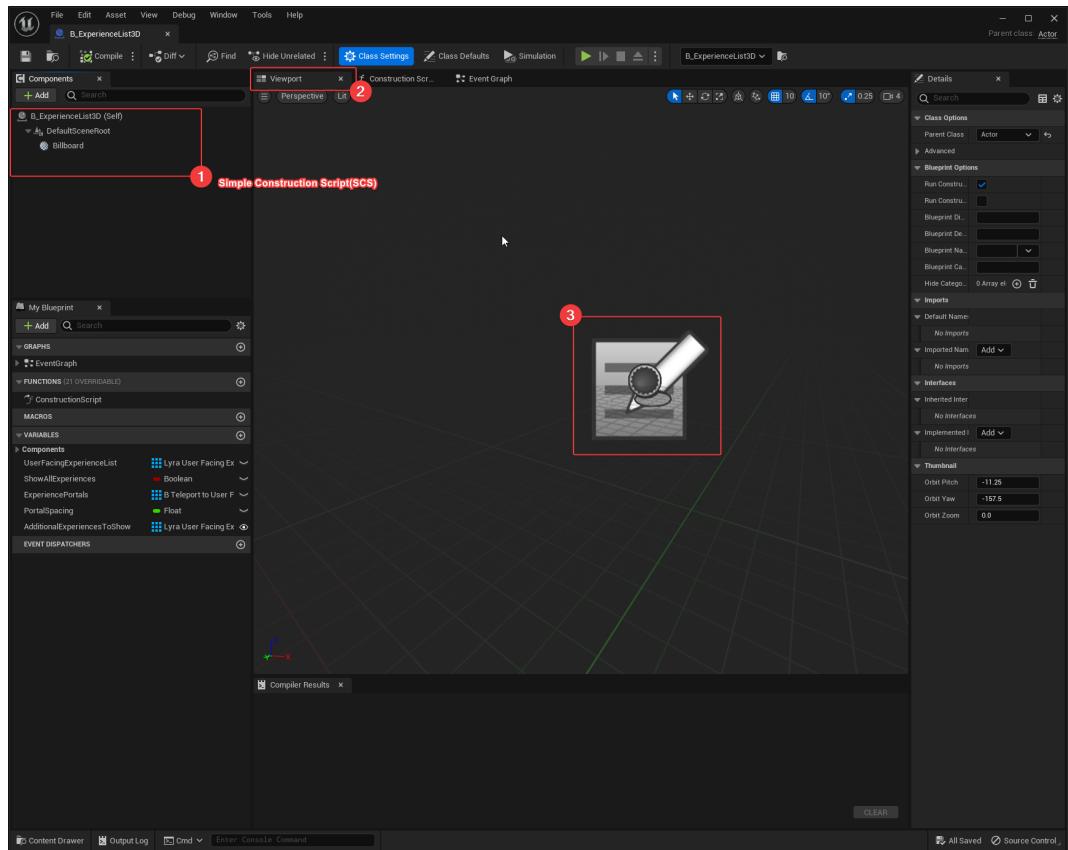
- 무엇을 상속 받는가?



아래와 같이 BP 생성해주자:



BP의 Viewport를 통해 **scs(Simple Construction Script)** 를 살펴보자:



- SCS는 BP를 구성하는 하나의 요소이다:
 - 정확히는 BP를 구성하는 요소로 볼 수 있다:

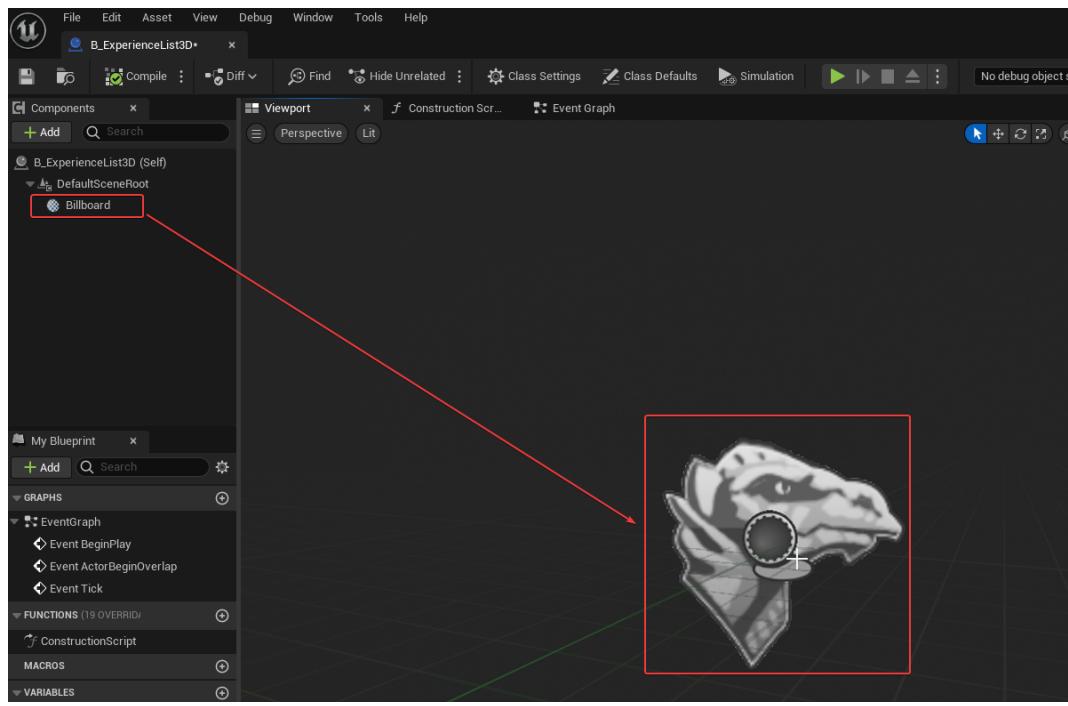
```

SimpleConstructionScript.h
1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "UObject/ObjectMacros.h"
7 #include "UObject/Object.h"
8 #include "Misc/Guid.h"
9 #include "GameFramework/Actor.h"
10 #include "SimpleConstructionScript.generated.h"
11
12 class USCS_Node;
13
14 UCLASS(MinimalAPI)
15 class USimpleConstructionScript : public UObject
16 {
17     GENERATED_UCLASS_BODY()
18
19     /** Suffix used for component template object name. */
20     ENGINE_API static const FString ComponentTemplateNameSuffix;
21
22     //~ Begin UObject Interface
23     virtual void Serialize(FArchive& Ar) override;
24     virtual void PostLoad() override;
25     //~ End UObject Interface
26
27     void PreloadChain();
28
29     /** Ensures that all root node parent references are still valid and clears the reference if not */
30     ENGINE_API void FixupRootNodeParentReferences();
31
32     /** Helper method to register instanced components post-construction */
33     static void RegisterInstancedComponent(UActorComponent* Component);
34
35 };

```

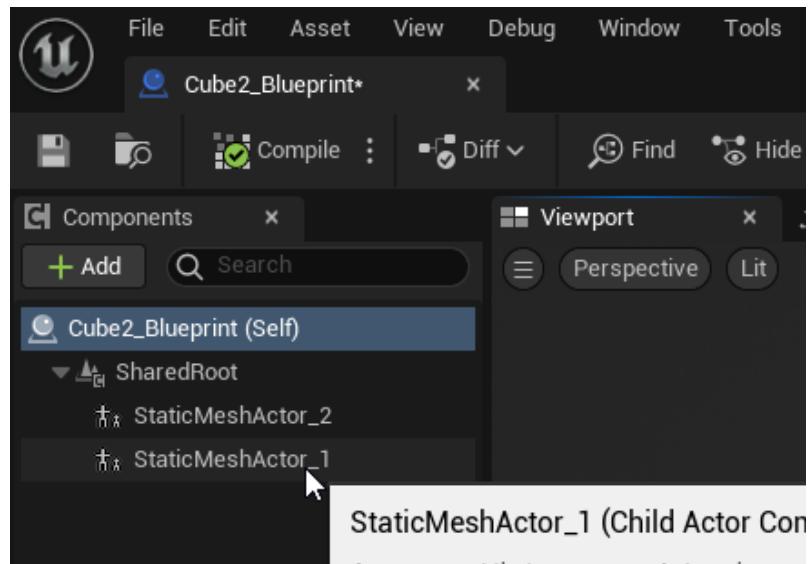
- 쉽게 이해하려면, Viewport에서 비주얼적으로 Component들을 Hierarchical하게 구성할 수 있는 시스템이라고 생각하면 된다

간단하게 BP의 계층구조를 활용하여, Billboard을 추가하였다:



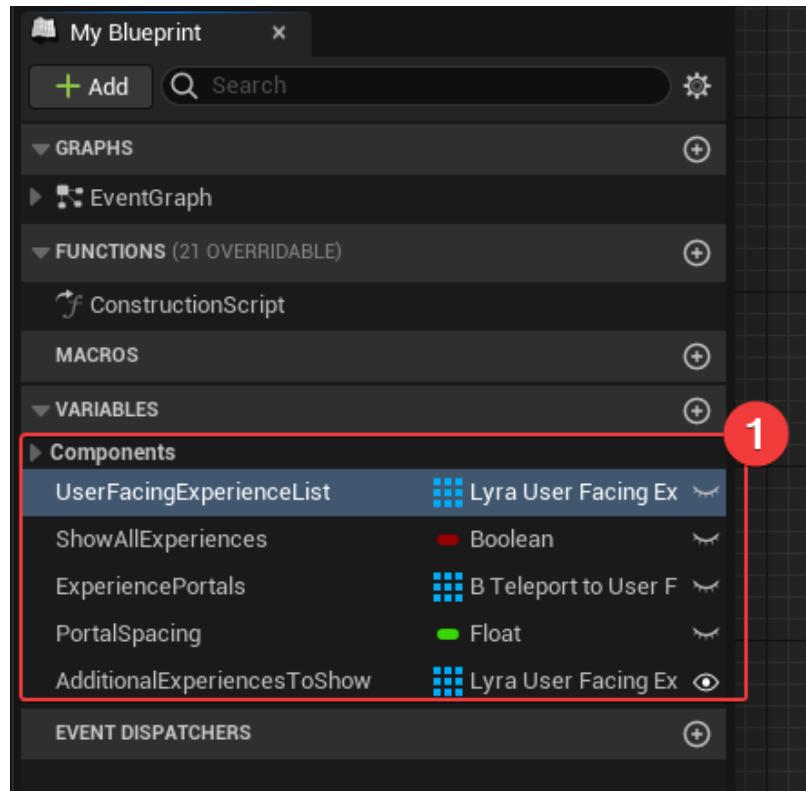
[숙제]

SCS에 대한 이해를 위해, 아래와 같이 Cube를 조합하여, BP로 변환하면서 SCS가 어떻게 생성되고 만들어지는지 확인해보자:

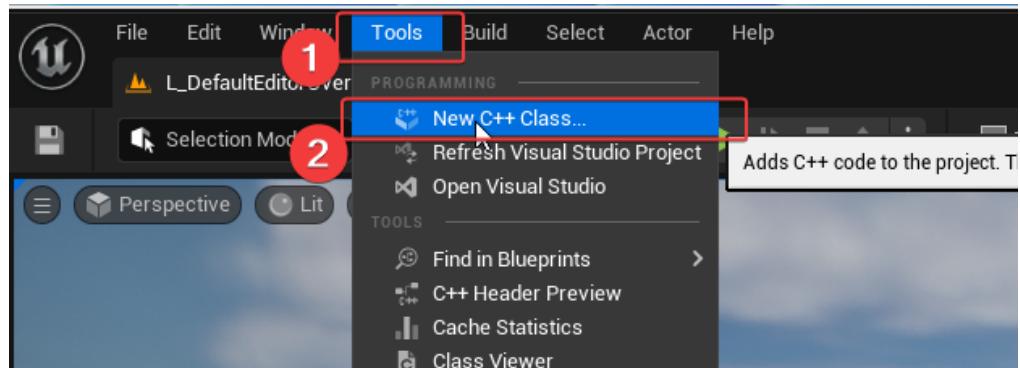


```
void CreateChildActorComponentsForActors(const FBlueprintAssemblyProps& AssemblyProps)
```

일단 멤버 변수들부터 구현해보자:



- 우리에게 `LyraUserFacingExperienceDefinition`이 필요하다:



LyraStarterGames에서 `ULyraUserFacingExperienceDefinition`은 어떻게 구성되었는지 간단히 살펴보자:

```


    /**
     * Description of settings used to display experiences in the UI and start a new session */
    UCLASS(BlueprintType)
    class ULyraUserFacingExperienceDefinition : public UPrimaryDataAsset
    {
        GENERATED_BODY()

    public:
        /** The specific map to load */
        UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience, meta=(AllowedTypes="Map"))
        FPrimaryAssetId MapID; 1

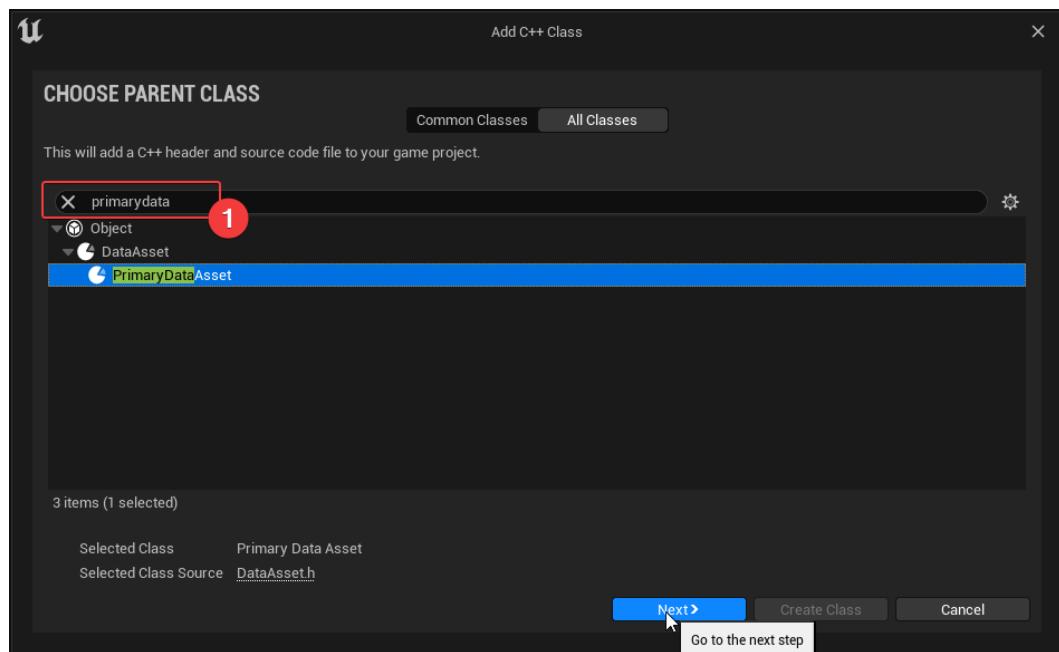
        /** The gameplay experience to load */
        UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience, meta=(AllowedTypes="LyraExperienceDefinition"))
        FPrimaryAssetId ExperienceID; 2

        /** Extra arguments passed as URL options to the game */
        UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience)
        TMap<FString, FString> ExtraArgs;

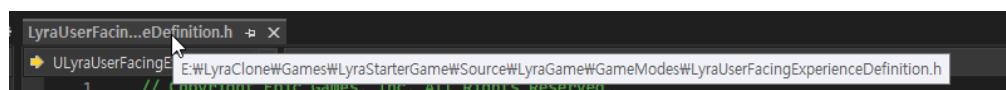
        /** Primary title in the UI */
        UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience)
        FText TileTitle;
    };

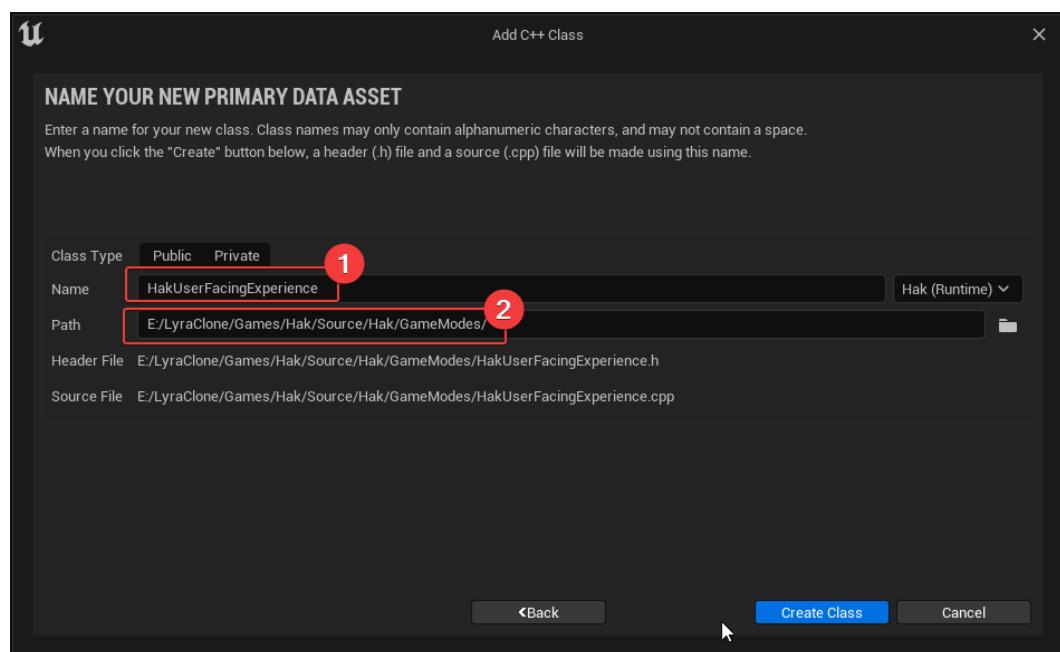
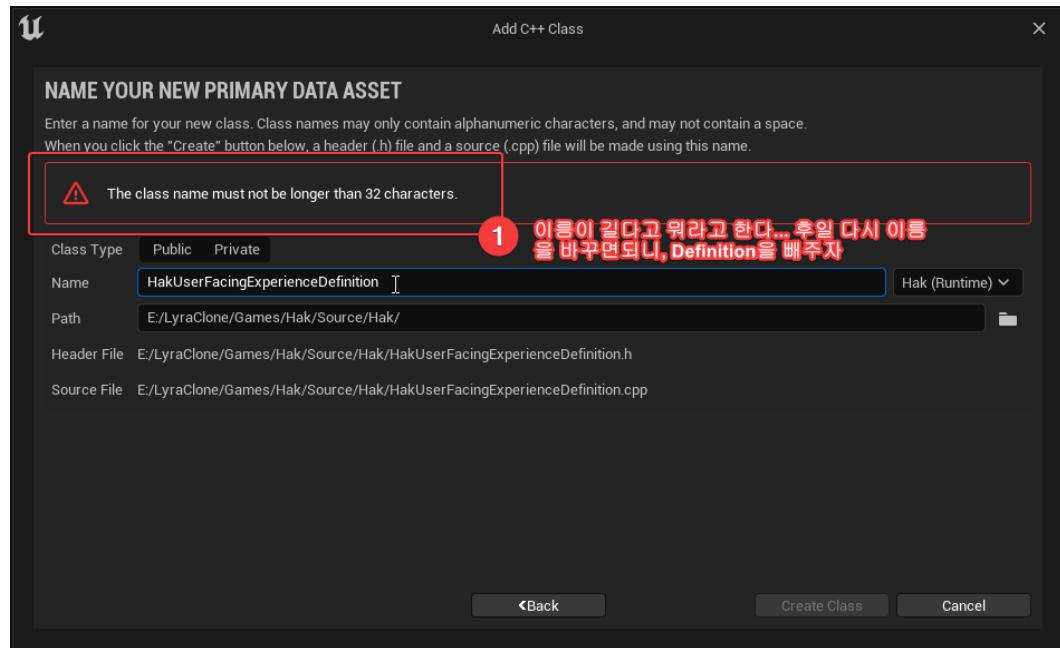

```

- 우선 `UPrimaryDataAsset`을 상속받고 있네?
- `FPrimaryAssetId`를 통해 다른 `UPrimaryDataAsset`을 가리키고 있다:
 - `FPrimaryAssetId`는 Database의 Primary Key와 같은 느낌으로 생각하면 쉽다 → 더 쉽게는 DataAsset의 ID라고 생각해도 무방하다!
- 아래와 같이 C++ 클래스를 만들어주자:



- 참고로 경로는:





- 앞서 못 넣었던 이름을 추가해주자:

- 아래와 같이 우선 파일 변경하자:

내 PC > 로컬 디스크 (E:) > LyraClone > Games > Hak > Source > Hak > GameModes				
이름	▲	수정한 날짜	유형	크기
HakUserFacingExperienceDefinition.cpp		2023-09-18 오후 6:26	CPP 파일	1KB
HakUserFacingExperienceDefinition.h		2023-09-18 오후 6:26	C Header 원본 파일	1KB

- 그리고, 변경된 파일을 VS Project에 적용하기 위해서는 저번주에 배웠던 `GeneratedProjectFiles.bat`를 활용하면 된다!

내 PC > 로컬 디스크 (E:) > LyraClone >

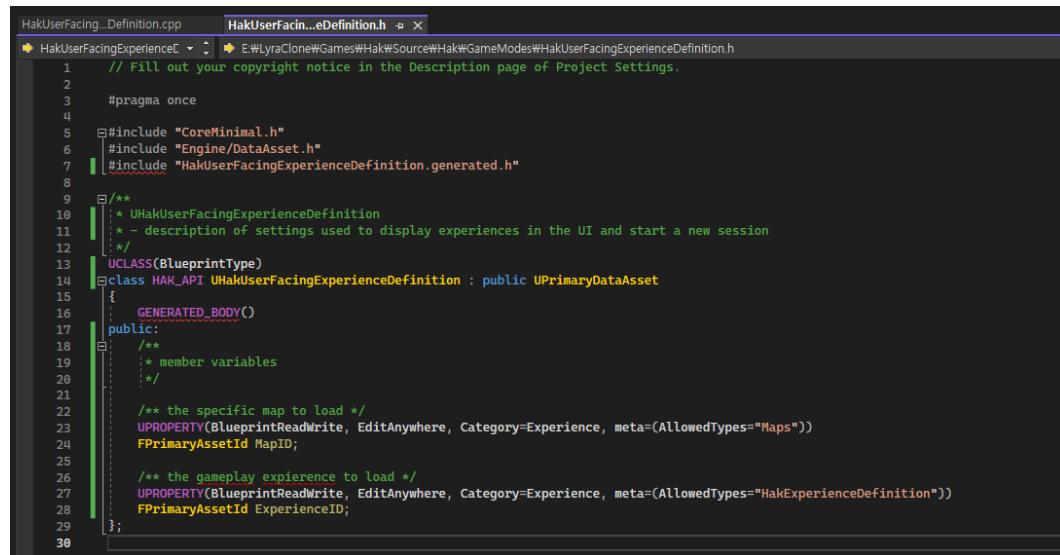
	이름	수정한 날짜	유형	크기
.	.git	2023-09-18 오전 4:55	파일 폴더	
.	.vs	2023-09-10 오후 10:10	파일 폴더	
1	Engine	2023-09-10 오후 9:31	파일 폴더	
1	FeaturePacks	2023-09-10 오후 8:24	파일 폴더	
3	Games	2023-09-11 오후 9:40	파일 폴더	
12	Samples	2023-09-10 오후 2:23	파일 폴더	
12	Templates	2023-09-10 오후 8:09	파일 폴더	
12	.editorconfig	2023-09-10 오후 2:22	Editor Config 파일	1KB
12	.gitattributes	2023-09-18 오전 3:54	Git Attributes 파일	5KB
12	.gitignore	2023-09-18 오전 4:48	Git Ignore 원본 파일	9KB
12	.tgitconfig	2023-09-10 오후 8:10	TGITCONFIG 파일	1KB
12	.uedependencies	2023-09-18 오전 3:55	UEDEPENDENCIES 파일	24,246KB
12	.vsconfig	2023-09-10 오후 8:37	VSCONFIG 파일	1KB
12	cpp.hint	2023-09-10 오후 8:12	HINT 파일	1KB
12	Default.uprojectdirs	2023-09-18 오전 3:54	UPROJECTDIRS 파일	1KB
1	GenerateProjectFiles.bat	2023-09-10 오후 2:23	Windows 배치 파일	1KB
1	GenerateProjectFiles.command	2023-09-10 오후 2:23	COMMAND 파일	1KB
1	GenerateProjectFiles.sh	2023-09-10 오후 2:23	sh_auto_file	1KB
1	git-add-commit.sh	2023-09-10 오후 3:35	sh_auto_file	1KB
1	git-push.sh	2023-09-10 오후 3:35	sh_auto_file	1KB
1	LICENSE.md	2023-09-10 오후 2:23	Markdown 원본 파일	1KB
1	README.md	2023-09-10 오후 2:23	Markdown 원본 파일	13KB

- 이제 알맞게 Class 이름도 변경해주자:

```

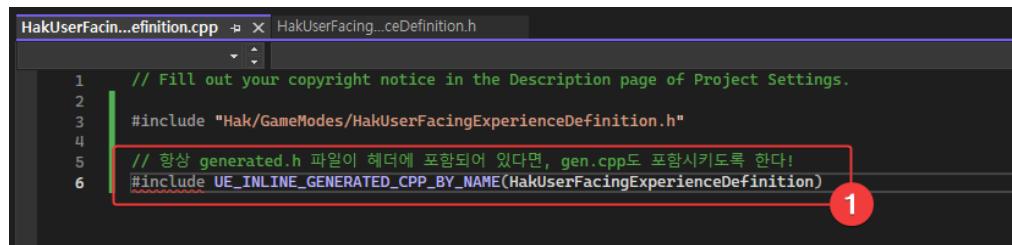
HakUserFacing...Definition.cpp   HakUserFacin...eDefinition.h ✘ ×
HakUserFacingExperienceC ↗   E:\LyraClone\Games\Hak\Source\Hak\GameModes\HakUserFacingExperienceDefinition.h
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "Engine/DataAsset.h"
7 #include "HakUserFacingExperienceDefinition.generated.h" [1]
8
9 /**
10 * 
11 */
12 UCLASS()
13 class HAK_API UHakUserFacingExperienceDefinition : public UPrimaryDataAsset
14 {
15     GENERATED_BODY()
16
17 };
18

```



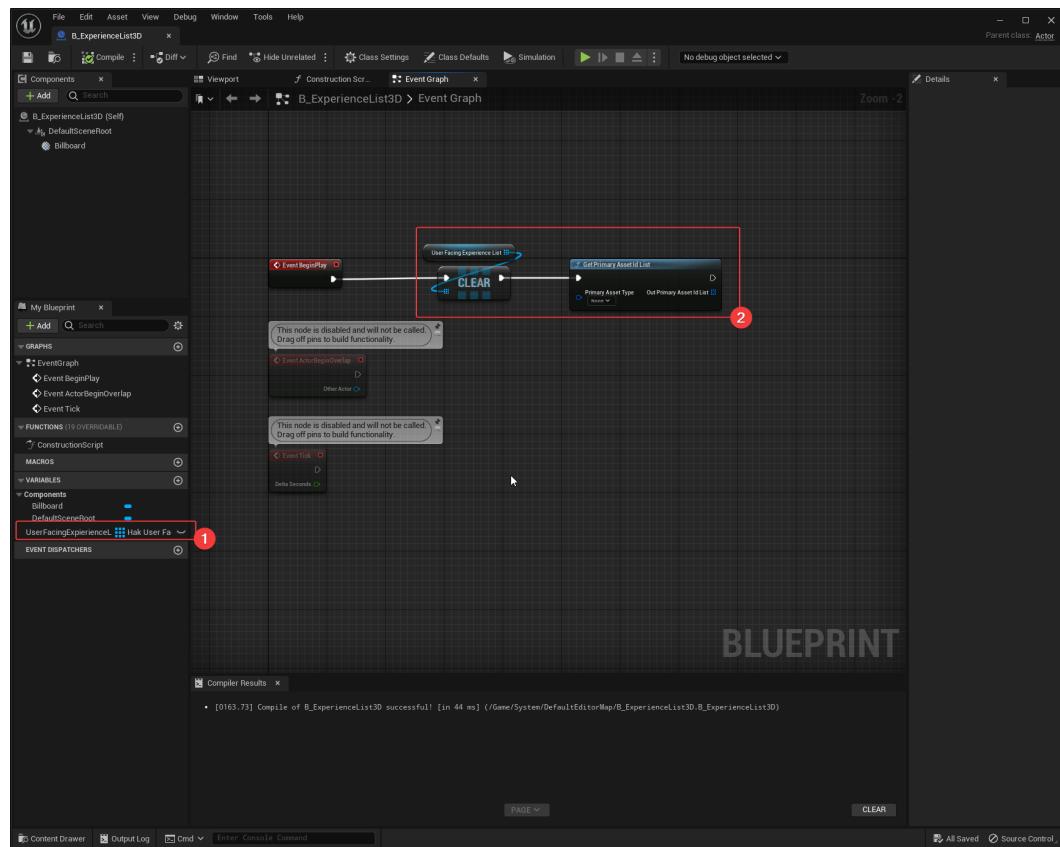
```
HakUserFacing...Definition.cpp | HakUserFacin...eDefinition.h ✘
HakUserFacingExperienceE ... E:\LyraClone\Games\Hak\Source\Hak\GameModes\HakUserFacingExperienceDefinition.h
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "Engine/DataAsset.h"
7 #include "HakUserFacingExperienceDefinition.generated.h"
8
9 /**
10  * UHakUserFacingExperienceDefinition
11  * - description of settings used to display experiences in the UI and start a new session
12  */
13 UCLASS(BlueprintType)
14 class HAK_API UHakUserFacingExperienceDefinition : public UPrimaryDataAsset
15 {
16     GENERATED_BODY()
17 public:
18     /**
19      * member variables
20     */
21
22     /** the specific map to load */
23     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience, meta=(AllowedTypes="Maps"))
24     FPrimaryAssetId MapID;
25
26     /** the gameplay experience to load */
27     UPROPERTY(BlueprintReadWrite, EditAnywhere, Category=Experience, meta=(AllowedTypes="HakExperienceDefinition"))
28     FPrimaryAssetId ExperienceID;
29 };
30
```

✓ 해당 부분 꼭 설명하기:

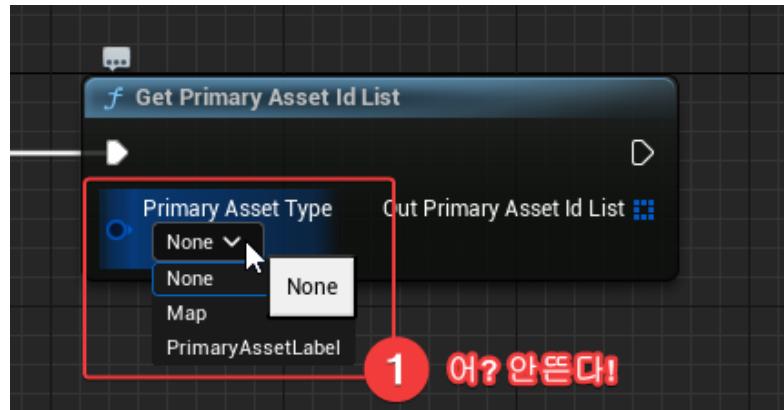


```
HakUserFacin...efinition.cpp ✘ | HakUserFacing...ceDefinition.h
1 // Fill out your copyright notice in the Description page of Project Settings.
2
3 #include "Hak/GameModes/HakUserFacingExperienceDefinition.h"
4
5 // 항상 generated.h 파일이 헤더에 포함되어 있다면, gen.cpp도 포함시키도록 한다!
6 #include UE_INLINED_GENERATED_CPP_BY_NAME(HakUserFacingExperienceDefinition)
```

- `LyraUserFacingExperienceDefinition` 를 참고하여, Clone하고, 다시 BP를 작성 시작해보자:'



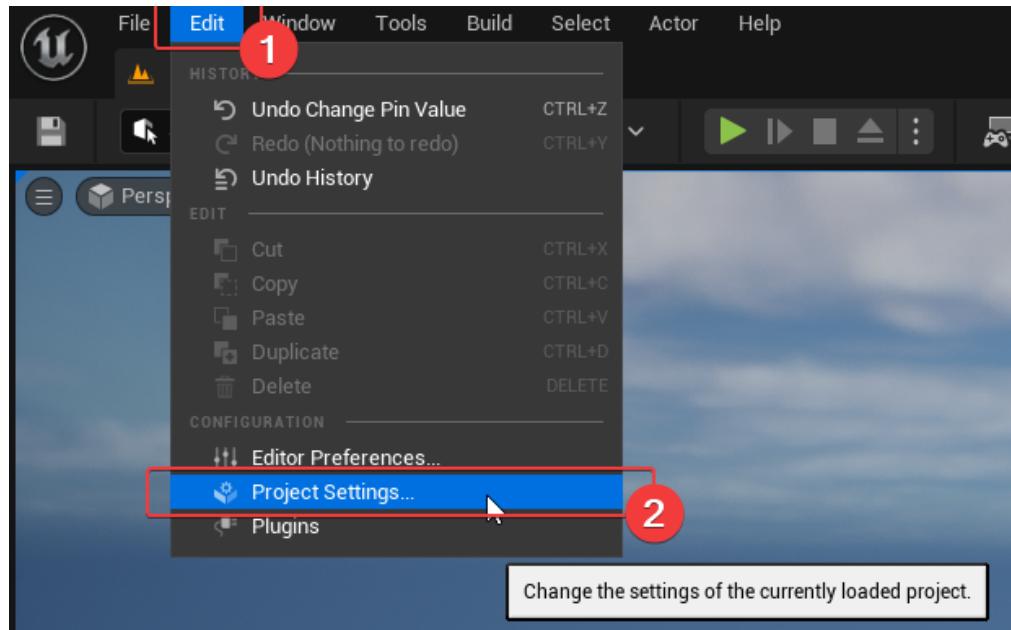
조금 작성하다보니, 막힌다!



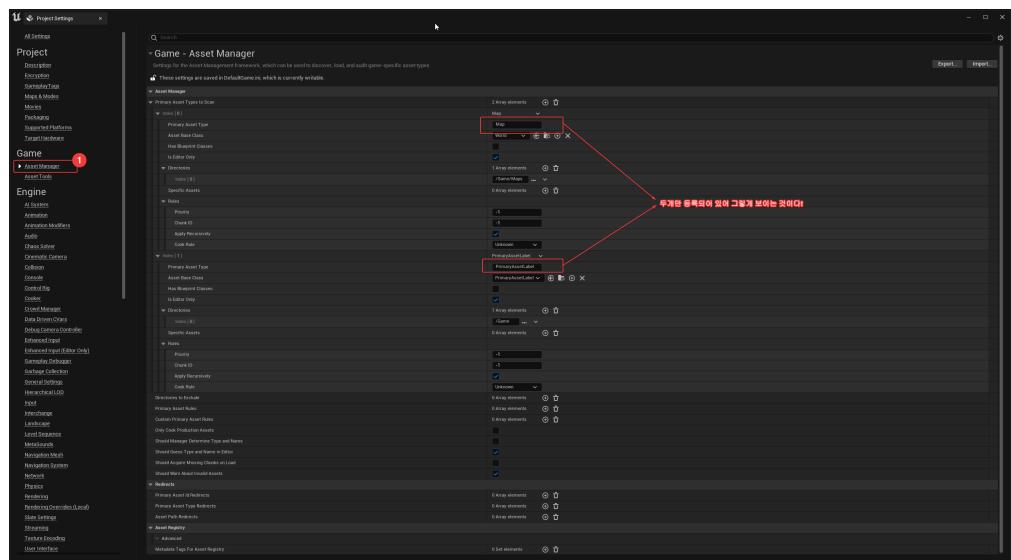
- Primary Asset Type에 보이지 않는다!

이는 우리의 UPrimaryDataAsset을 상속받은 HakUserFacingExperienceDefinition이 AssetManager에 등록되어있지 않아서 그렇다!

- Project Settings에서:



- 아래와 같이 앞서 보았던 Map과 PrimaryAssetLabel만 등록되어 있어서 딱 그것만 보이던 것이다:



우리는 UAssetManager가 아닌 ULyraAssetManager처럼 독립적인 AssetManager를 만들어서 진행하려고 한다:

```

1 // Copyright Epic Games, Inc. All Rights Reserved.
2
3 #pragma once
4
5 #include "CoreMinimal.h"
6 #include "Engine/AssetManager.h"
7 #include "Engine/DataAsset.h"
8 #include "LyraAssetManagerStartupJob.h"
9 #include "LyraAssetManager.generated.h"
10
11 class ULyraGameData;
12 class ULyraPawnData;
13
14 struct FLyraBundles
15 {
16     static const FName Equipped;
17 };
18
19 /**
20 * UlyraAssetManager
21 */
22
23 /**
24 * Game implementation of the asset manager that overrides functionality and stores game-specific types.
25 * It is expected that most games will want to override AssetManager as it provides a good place for game-specific loading logic.
26 */
27 UCLASS(Config = Game)
28 class ULYRAASSETMANAGER : public UAssetManager
29 {
30     GENERATED_BODY()
31
32     public:
33         ULYRAASSETMANAGER();
34

```

정리

▼ 자세히

- 오늘은 BP의 Cloning에 대한 방법을 익혔다:
 - 이를 활용하여, Lyra의 다른 BP도 읽어보고, 따라 만들어보자!
- GenerateProjectFiles.bat를 통해, 소스파일에 대한 프로젝트 반영을 업데이트 가능하다!
- 공부할 내용들:
 - Dynamic Link Library(DLL)와 Static Library의 차이점
 - 파일 로딩을 통한 메모리 로딩에 대한 개념 이해
 - 프로세스와 스레드의 차이점
 - Virtual Memory와 Physical Memory 차이
 - Virtual Memory Table 무엇일까?
 - Virtual Memory Mapping 과정?
 - SCS 동작 원리 디버깅해보기