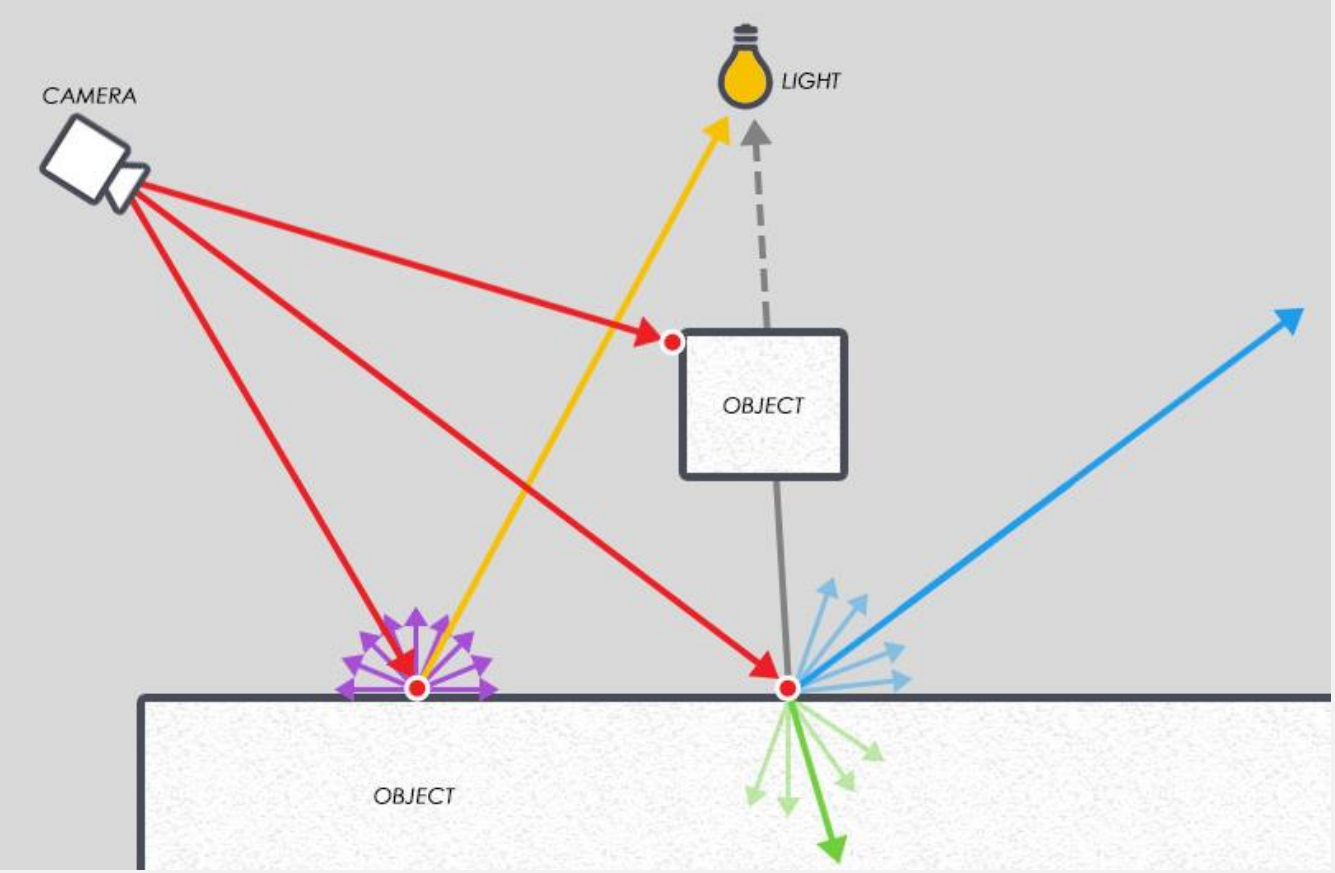
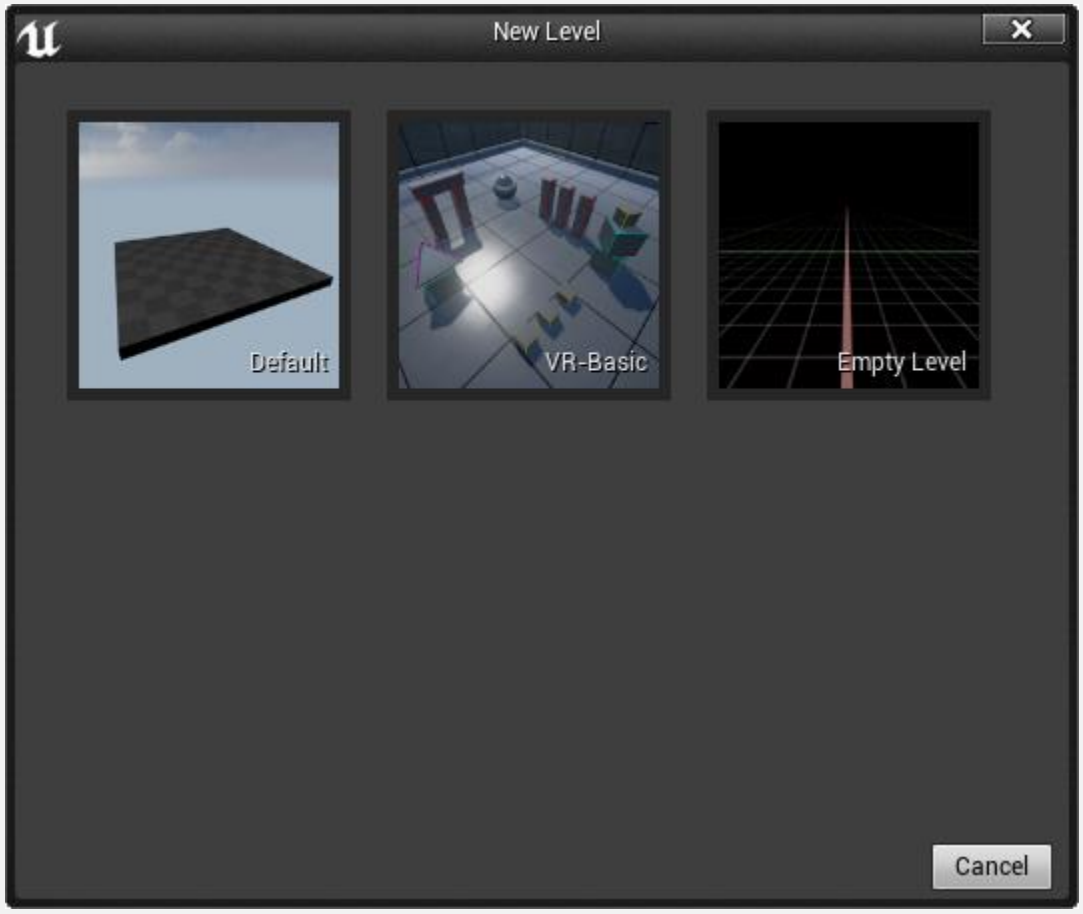
Lab #2.
블루프린트 기초

3D 렌더링 기본 구성 요소

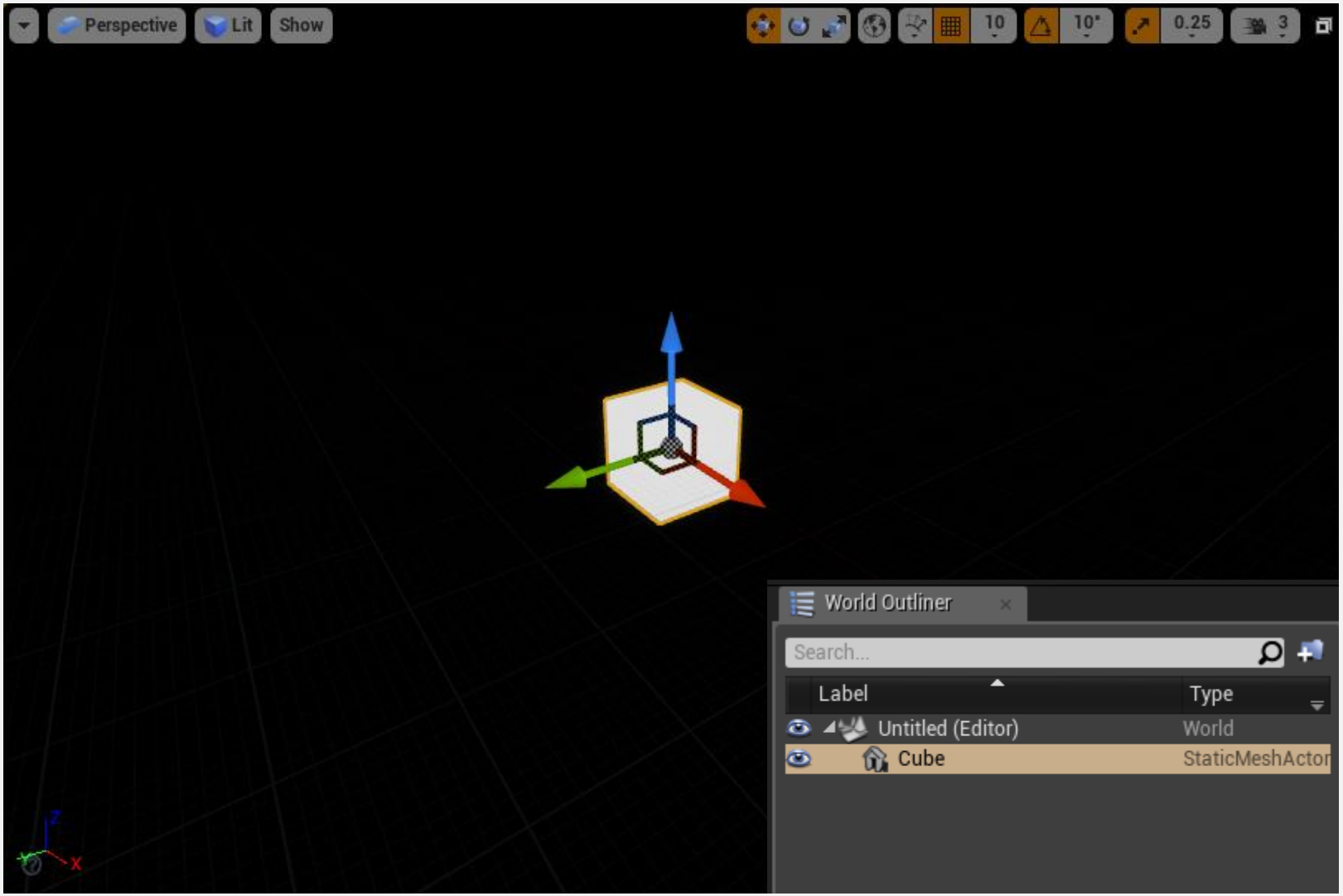
- 씬(Scene) - 여러개의 객체(Object)들로 구성
- 광원(Light)
- 카메라(Camera)



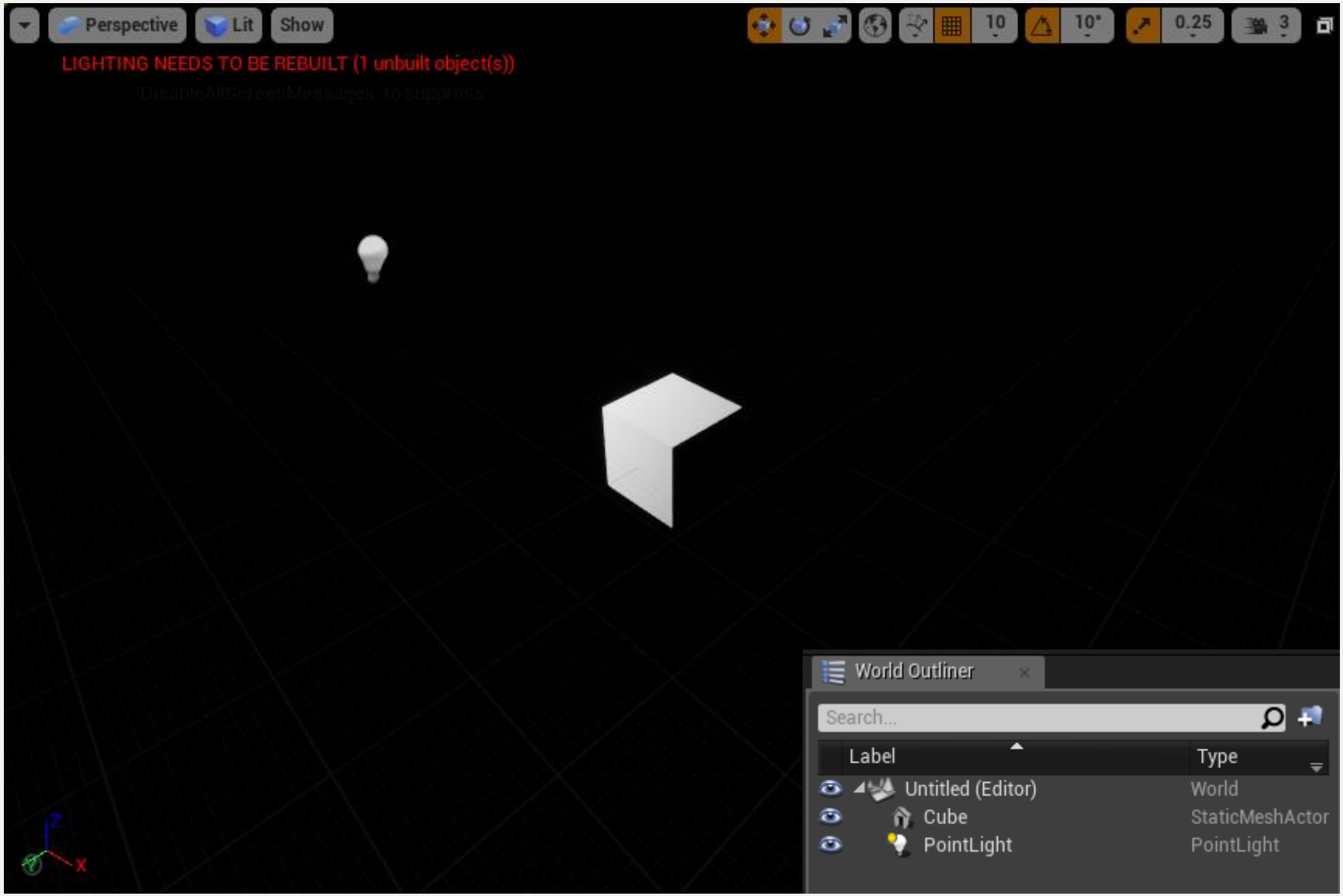
Empty Level 로 새로운 레벨 생성

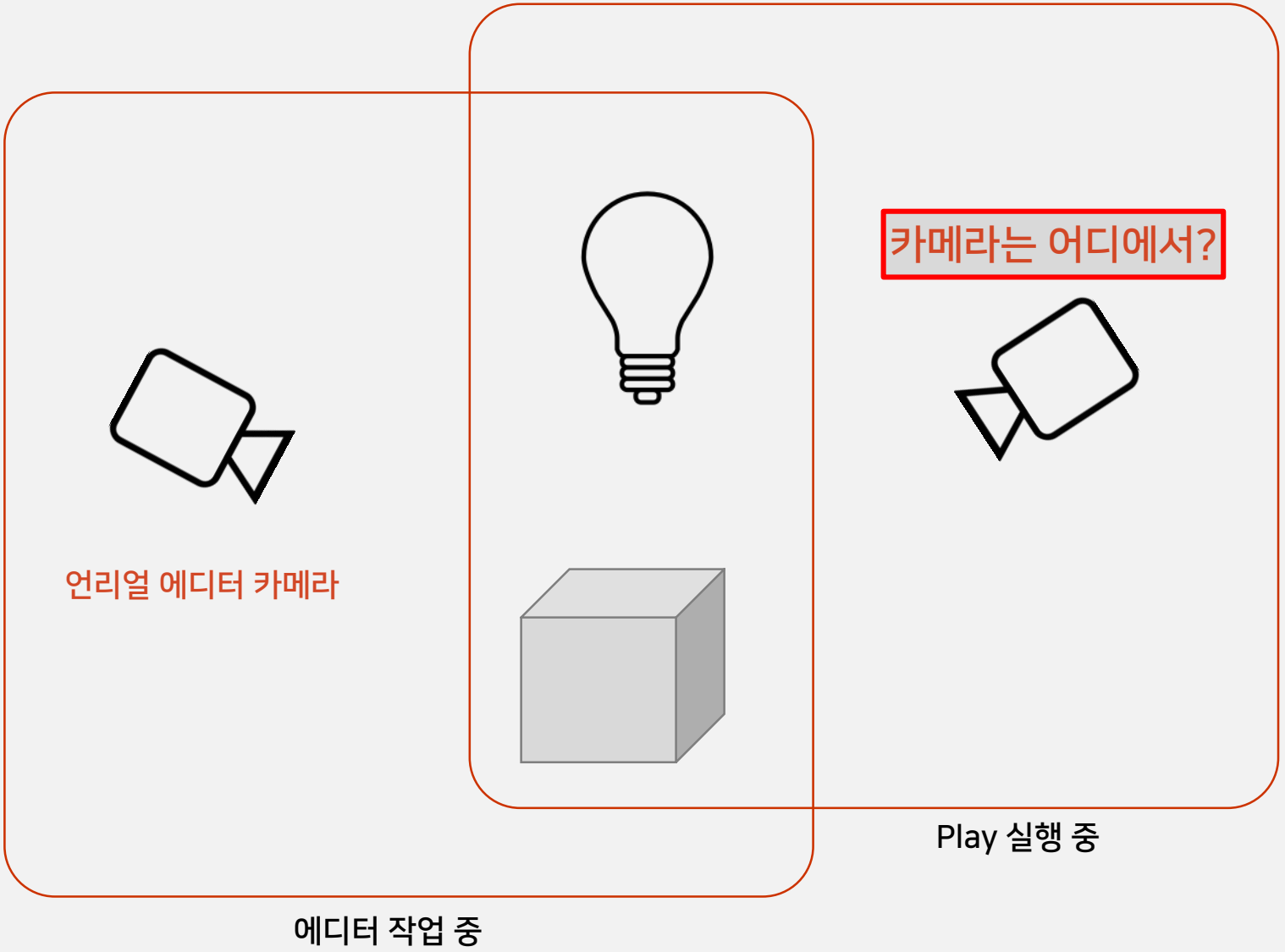


Cube 를 원점에 배치하고 Play !

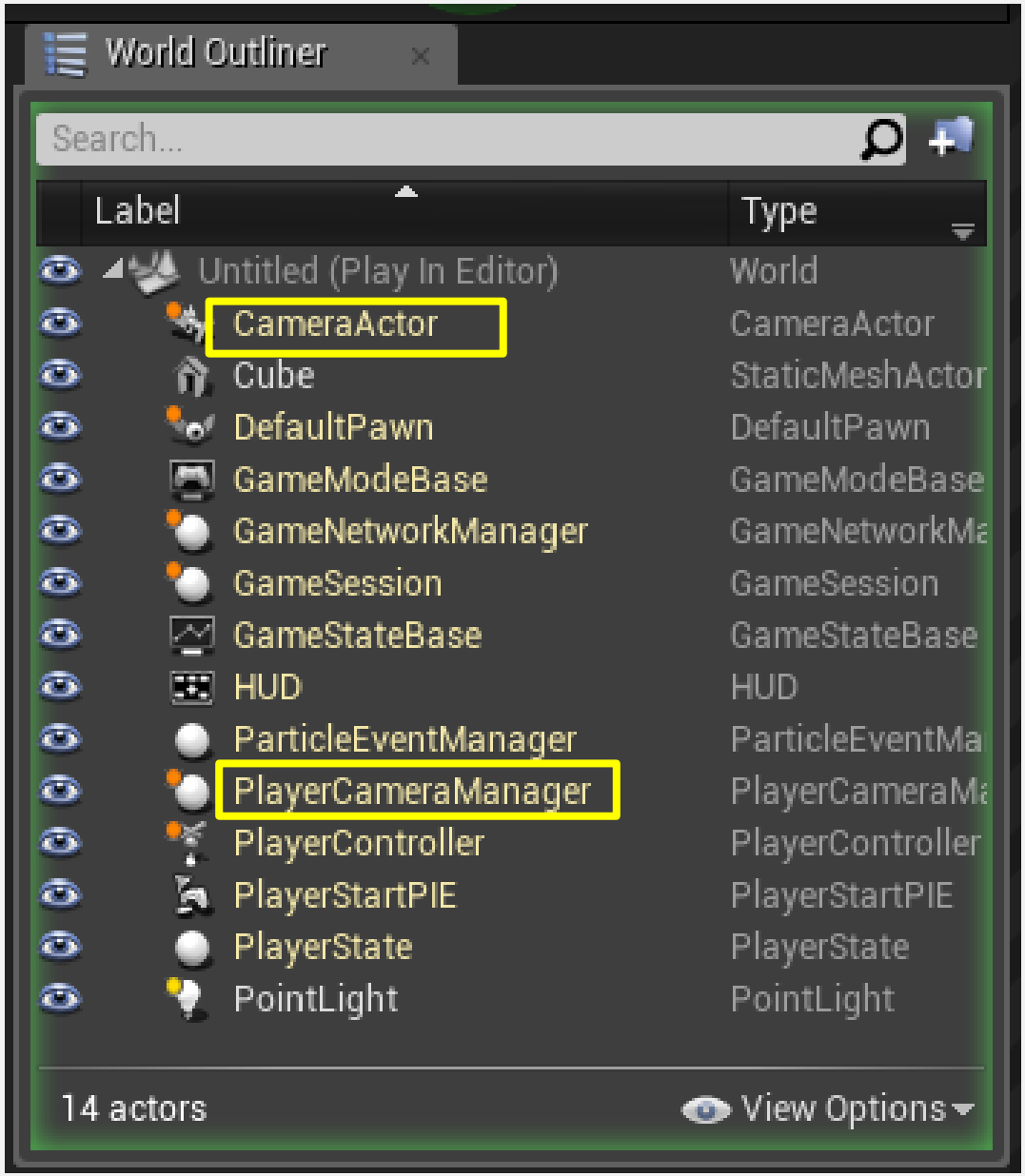


포인트 라이트(Point Light)의 배치하고 Play !!

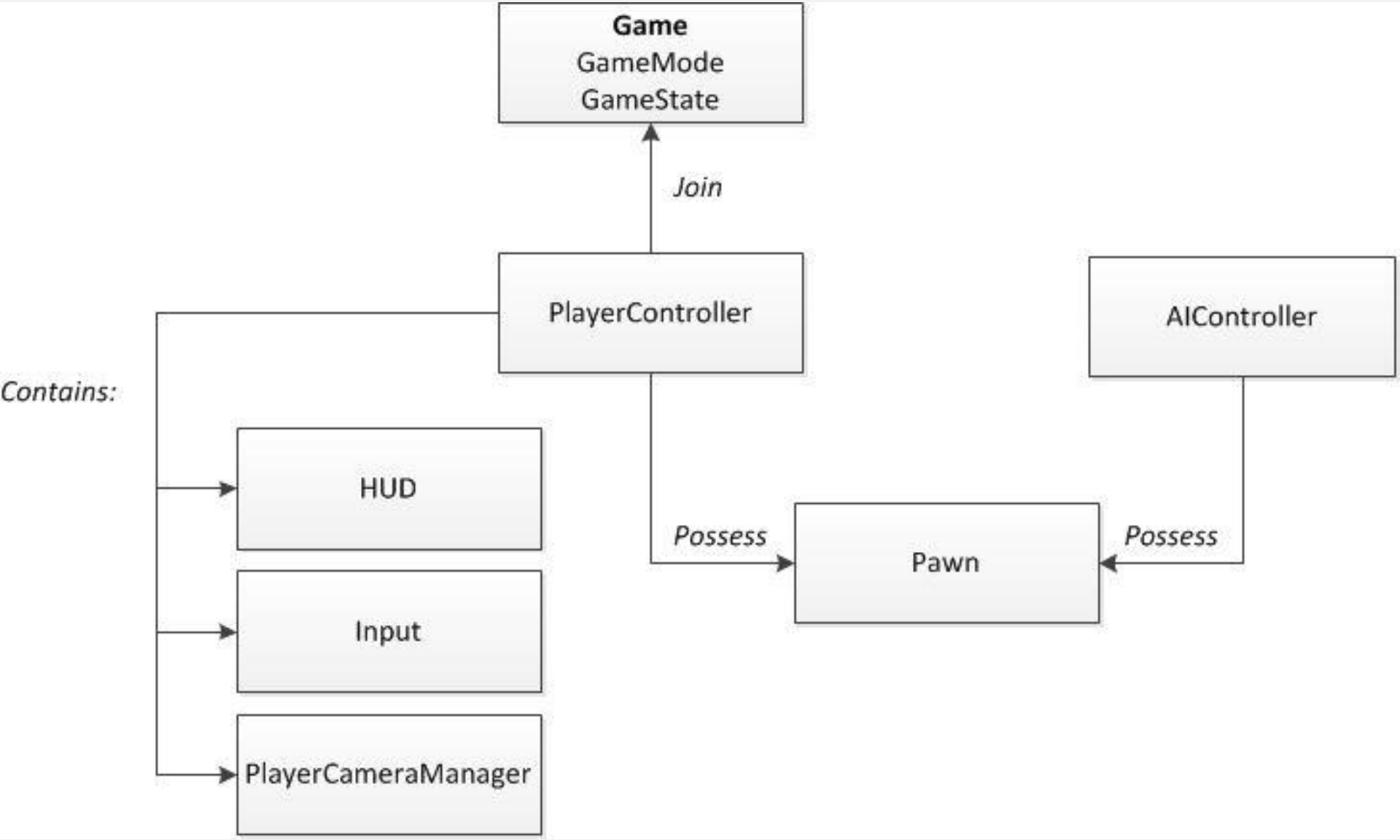


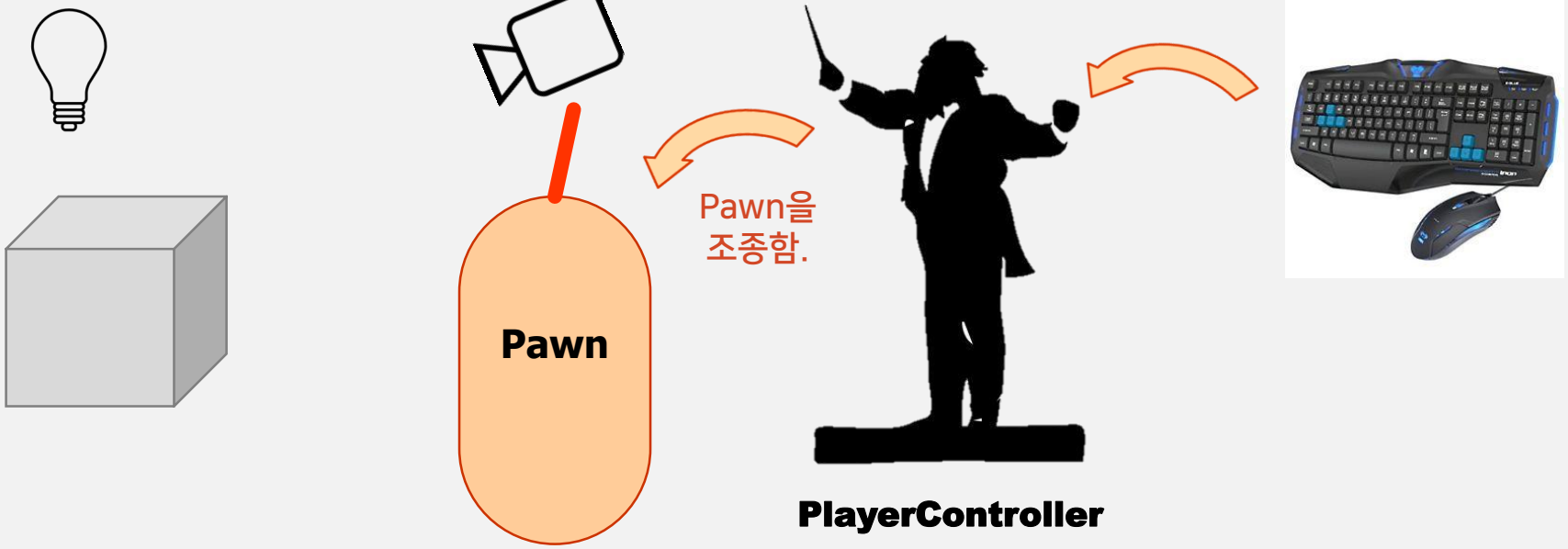

카메라는 어디에서?
언리얼 에디터 카메라
Play 실행 중
에디터 작업 중

플레이 실행하면, 많은 액터들이 생성된다. – 엔진이 생성하는 것임.



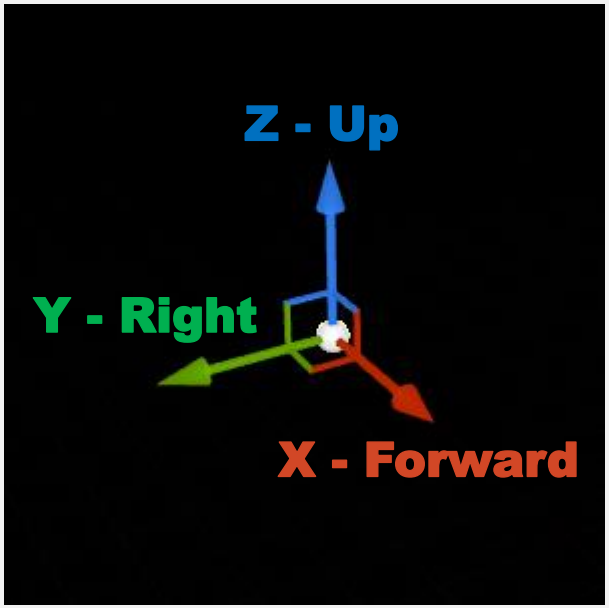
게임 프레임워크 클래스 관계도




카메라는
Pawn에
붙어있음.
Camera
PlayerController는
입력을 체크.
Pawn을
조종함.
Pawn
PlayerController
Pawn은 게임유저의
아바타 역할.

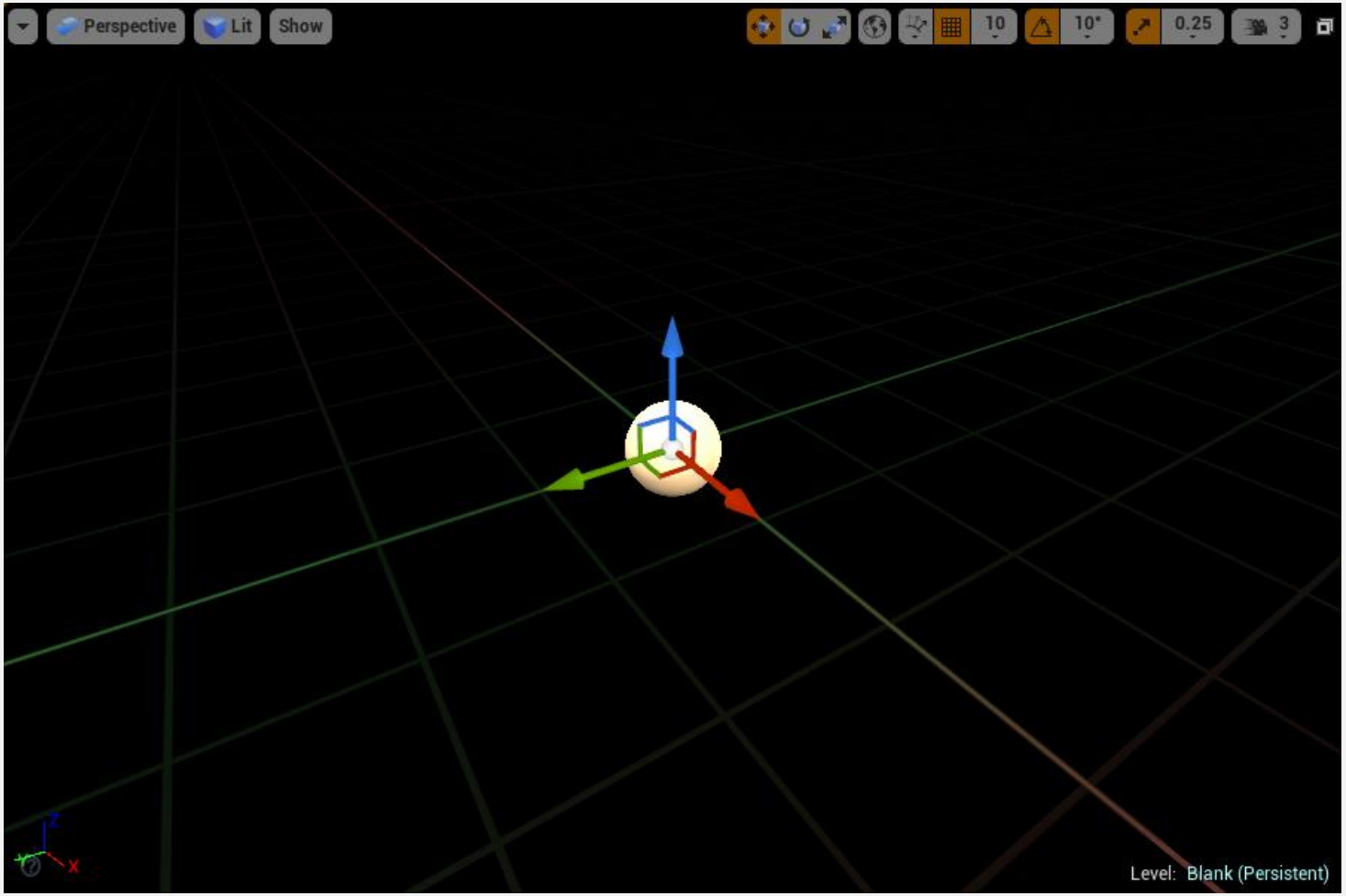
언리얼 엔진 좌표계 : 왼손 좌표계

- Z : Normal Vector의 방향으로 설정하는 것이 일반적
- X : 3D 모델이 바라보는(Facing) 방향으로 설정하는 것이 일반적



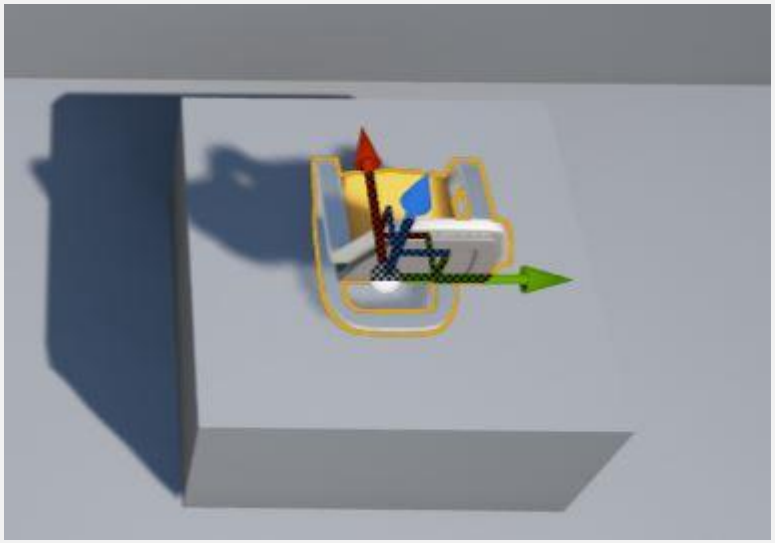
회전의 경우, Z 축 회전은 Left-Handed, 나머지는 Right-Handed

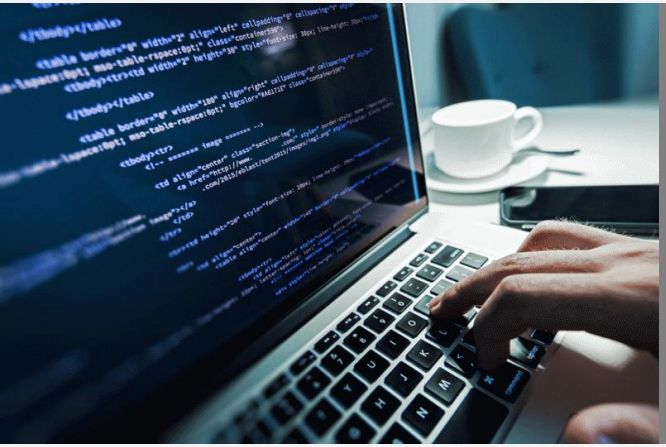
언리얼 엔진 격자 단위 – 1uu(unreal unit) = 1cm



Pivot

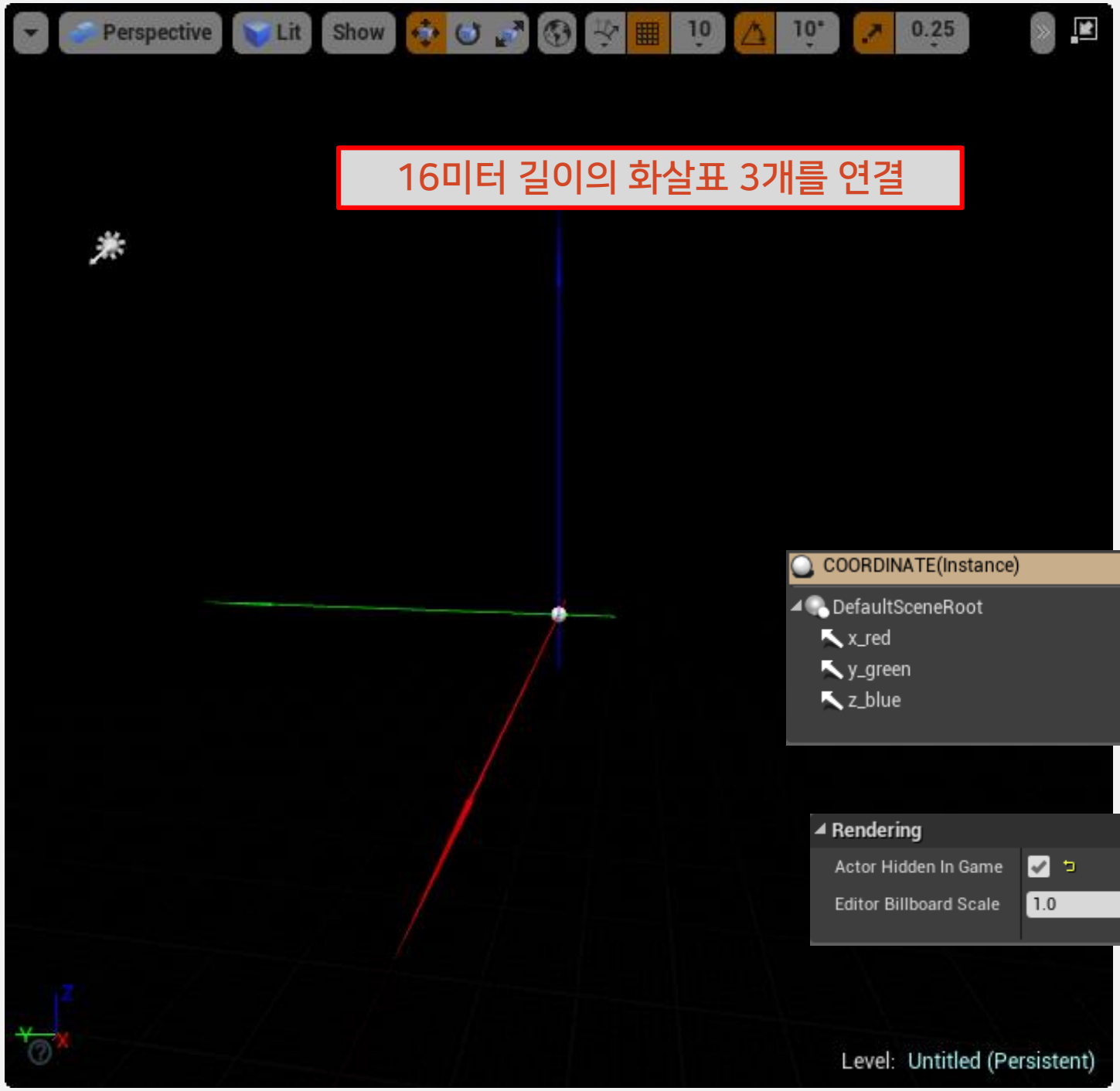
- 객체를 공간에 배치할 때, 기준이 되는 위치
- 아티스트와 개발자에 사전에 약속하는 것이 좋음.





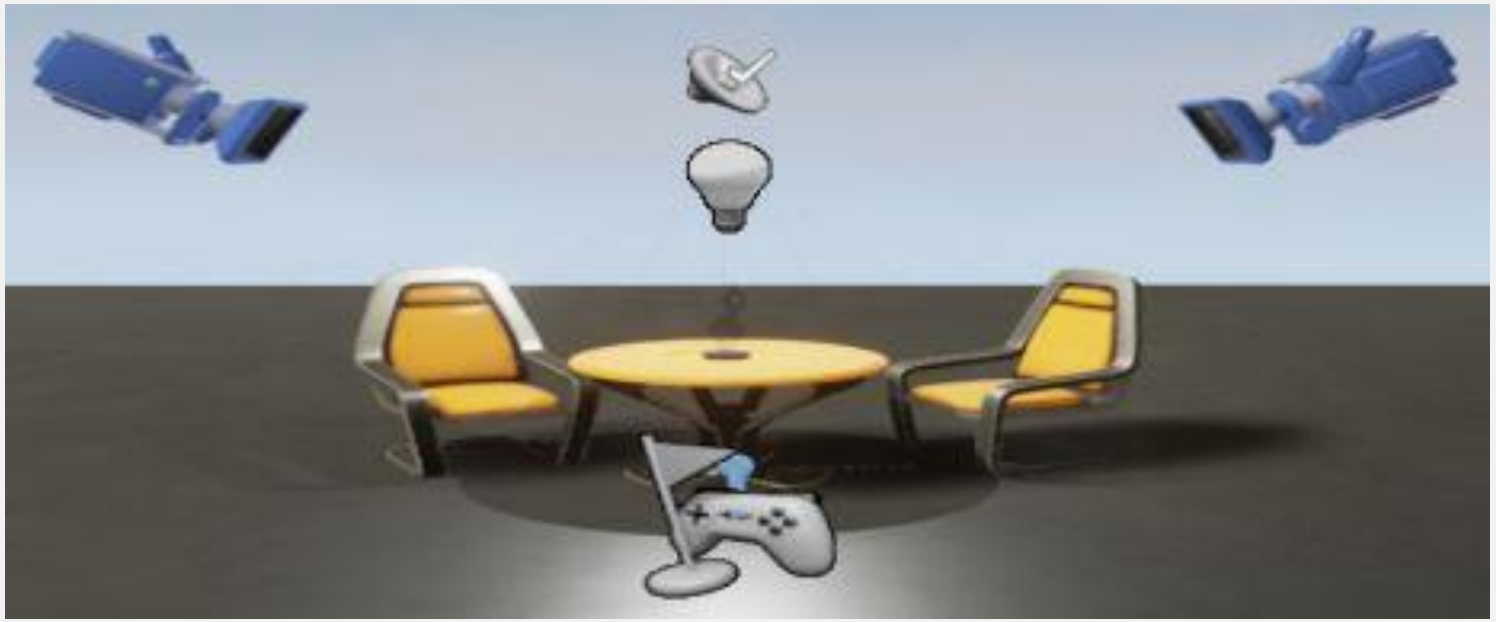
실습

좌표축 가이드 만들기


Perspective
Lit
Show
10
10°
0.25
16미터 길이의 화살표 3개를 연결
COORDINATE(Instance)
DefaultSceneRoot
x_red
y_green
z_blue
Rendering
Actor Hidden In Game
Editor Billboard Scale
1.0
Level: Untitled (Persistent)

Actor (액터)

- 레벨 안에 배치할 수 있는 오브젝트
- 이동, 회전, 스케일과 같은 3D 트랜스폼 지원. 즉, 3차원 공간에 배치될 수 있음.
- 게임플레이코드(C++ 또는 블루프린트)로 생성(Spawn) 및 소멸이 가능
- C++ 이름: AActor
- 대표적 액터: StaticMeshActor, CameraActor, PlayStartActor



Component (컴포넌트)

- 액터에 추가시킬 수 있는 부품.
- “A Piece of Functionality” : 어떤 기능을 구현하는 부품.
- 독립적으로는 존재할 수 없으며, 액터 안에 추가시켜서 액터를 다양한 꾸밀 수 있음.

AI 컴포넌트 ›

AI 인지에 사용되는 AI 관련 컴포넌트 및 폰 감각에 대한 설명입니다.

오디오 컴포넌트 ›

AudioComponent 는 사운드 인스턴스의 생성 및 제어에 사용됩니다.

카메라 컴포넌트 ›

카메라 컴포넌트와 스프링 암 컴포넌트에 대한 설명입니다.

라이트 컴포넌트 ›

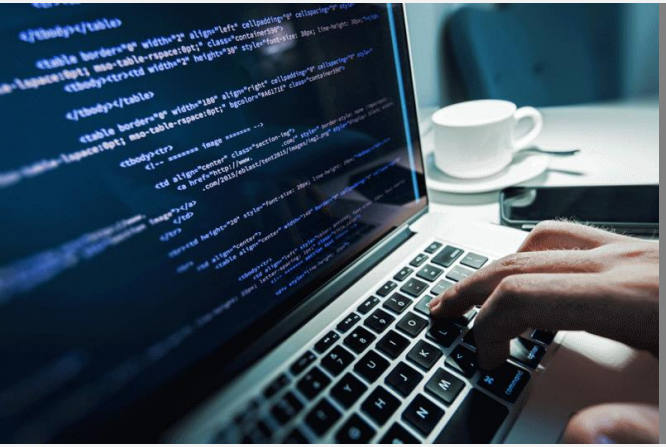
언리얼 엔진 4 에서 사용할 수 있는 여러가지 Light 컴포넌트에 대한 설명입니다.

무브먼트 컴포넌트 ›

캐릭터든 프로젝타일이든, 이동에 관련된 모든 것은 무브먼트 컴포넌트를 사용합니다.

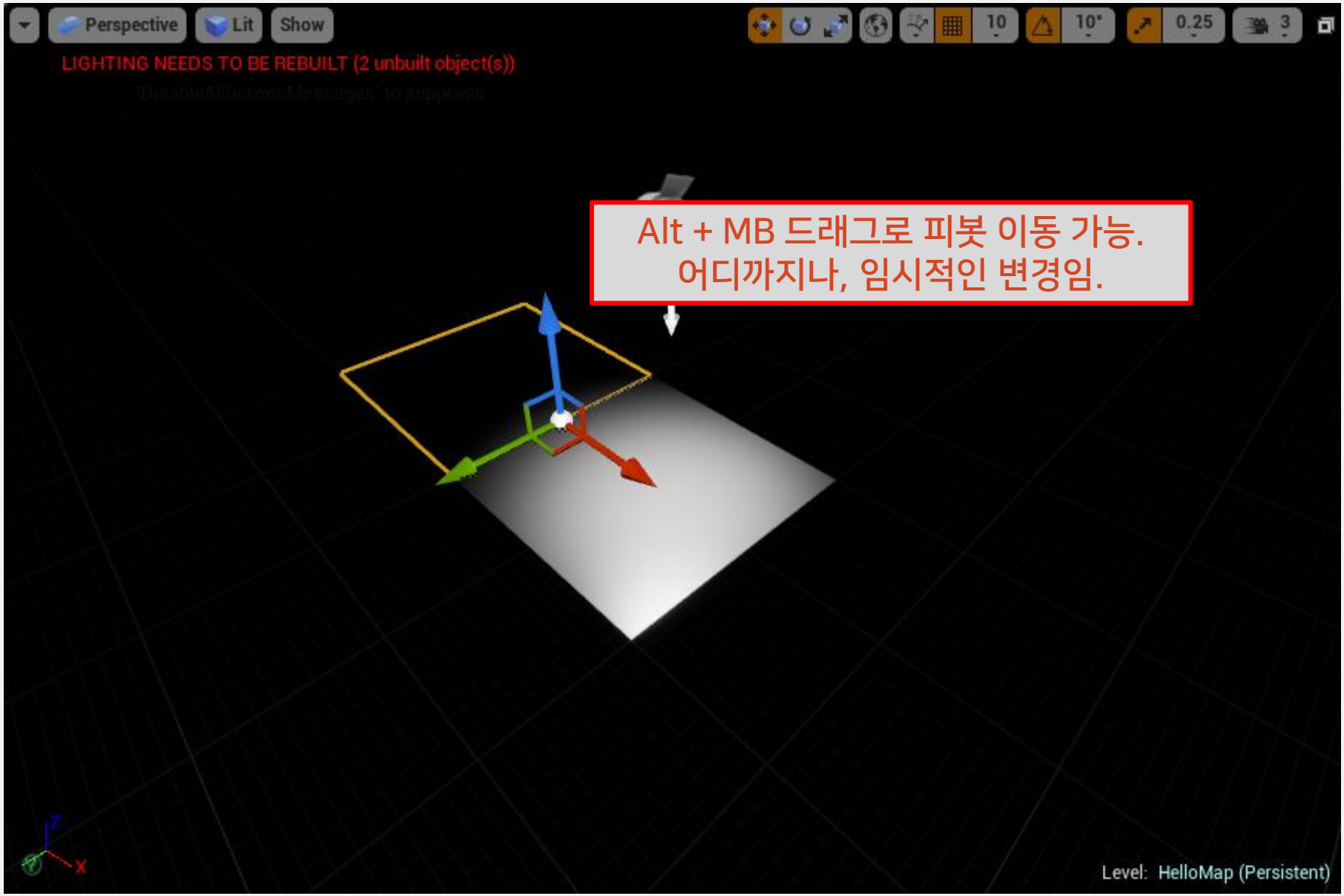
내비게이션 컴포넌트 ›

볼륨의 모양을 사용하여 선택된 AreaClass 를 내비메시에 적용할 수 있습니다.

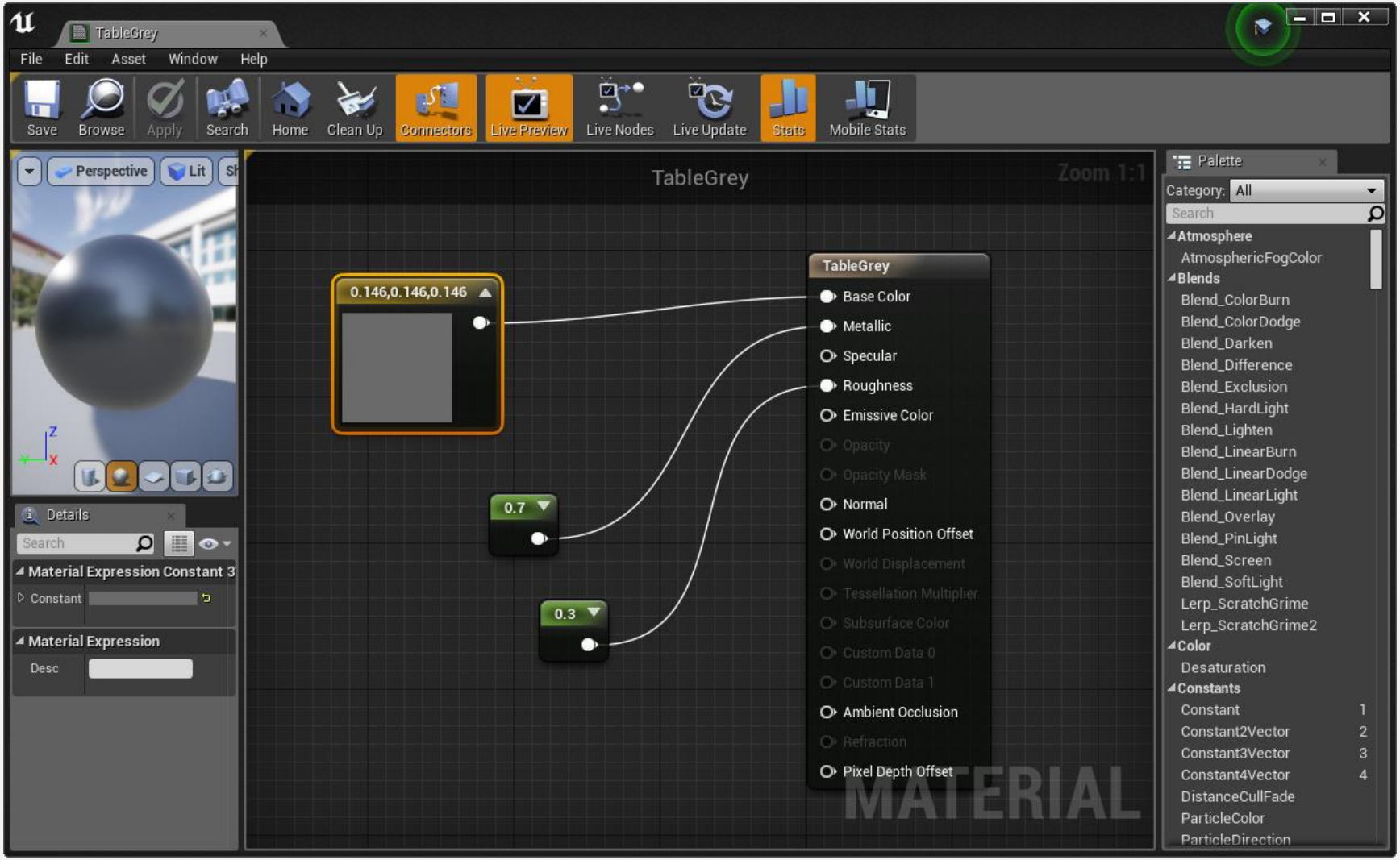


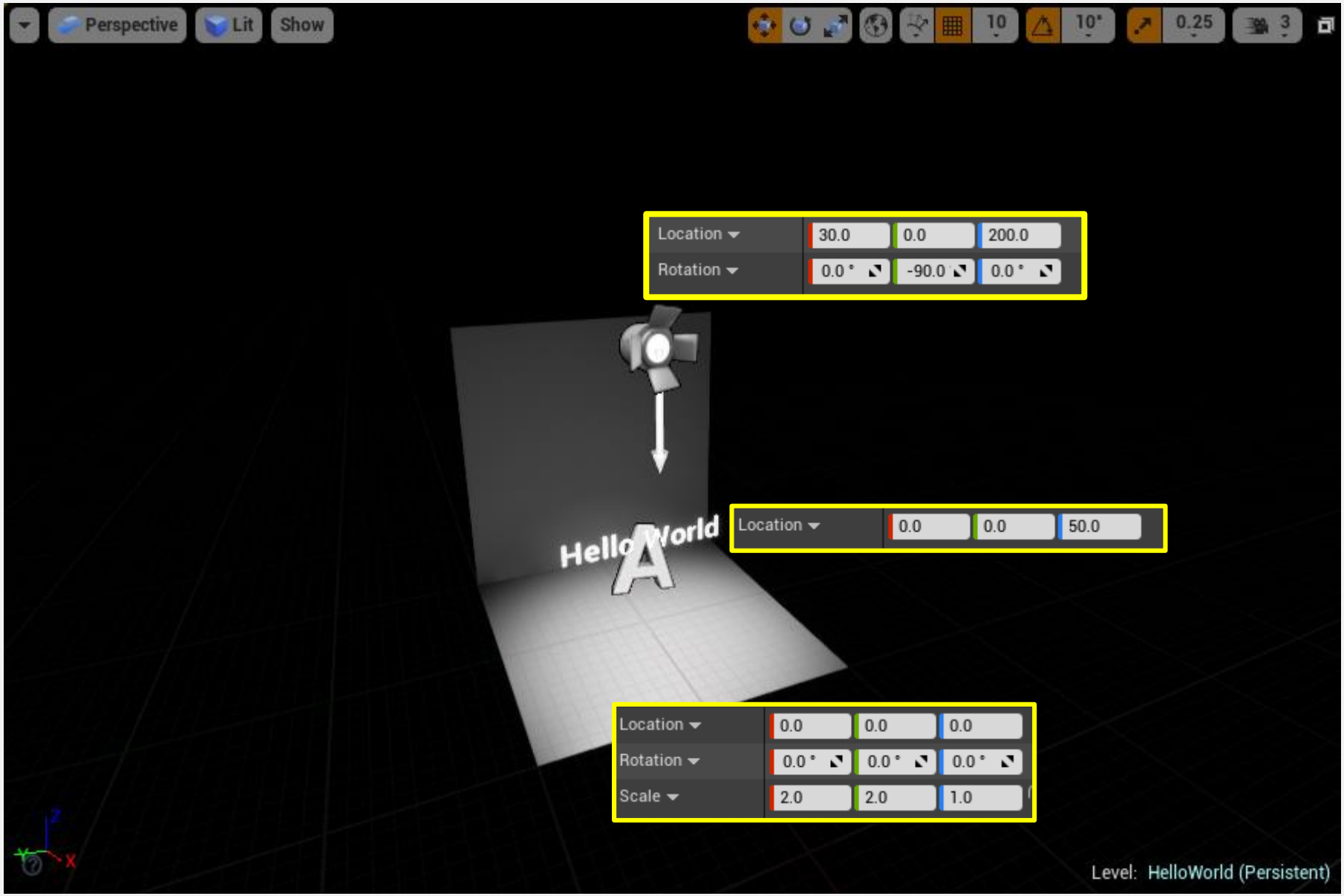
실습

Hello World 표시

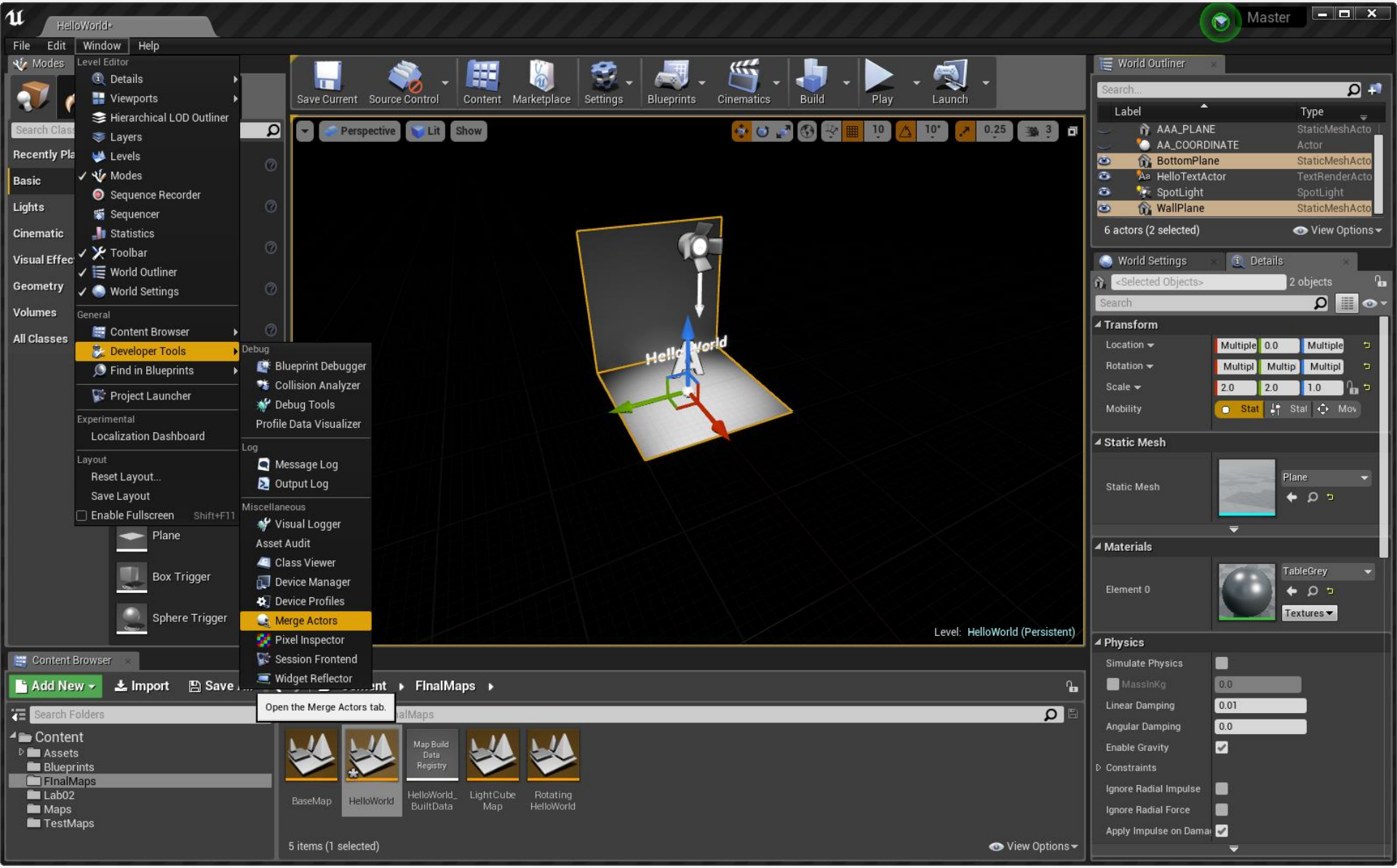

Perspective
Lit
Show
10
10°
0.25
3
LIGHTING NEEDS TO BE REBUILT (2 unbuilt object(s))
Alt + MB 드래그로 피봇 이동 가능.
어디까지나, 임시적인 변경임.
Level: HelloMap (Persistent)

TableGrey 재질(Material) 생성

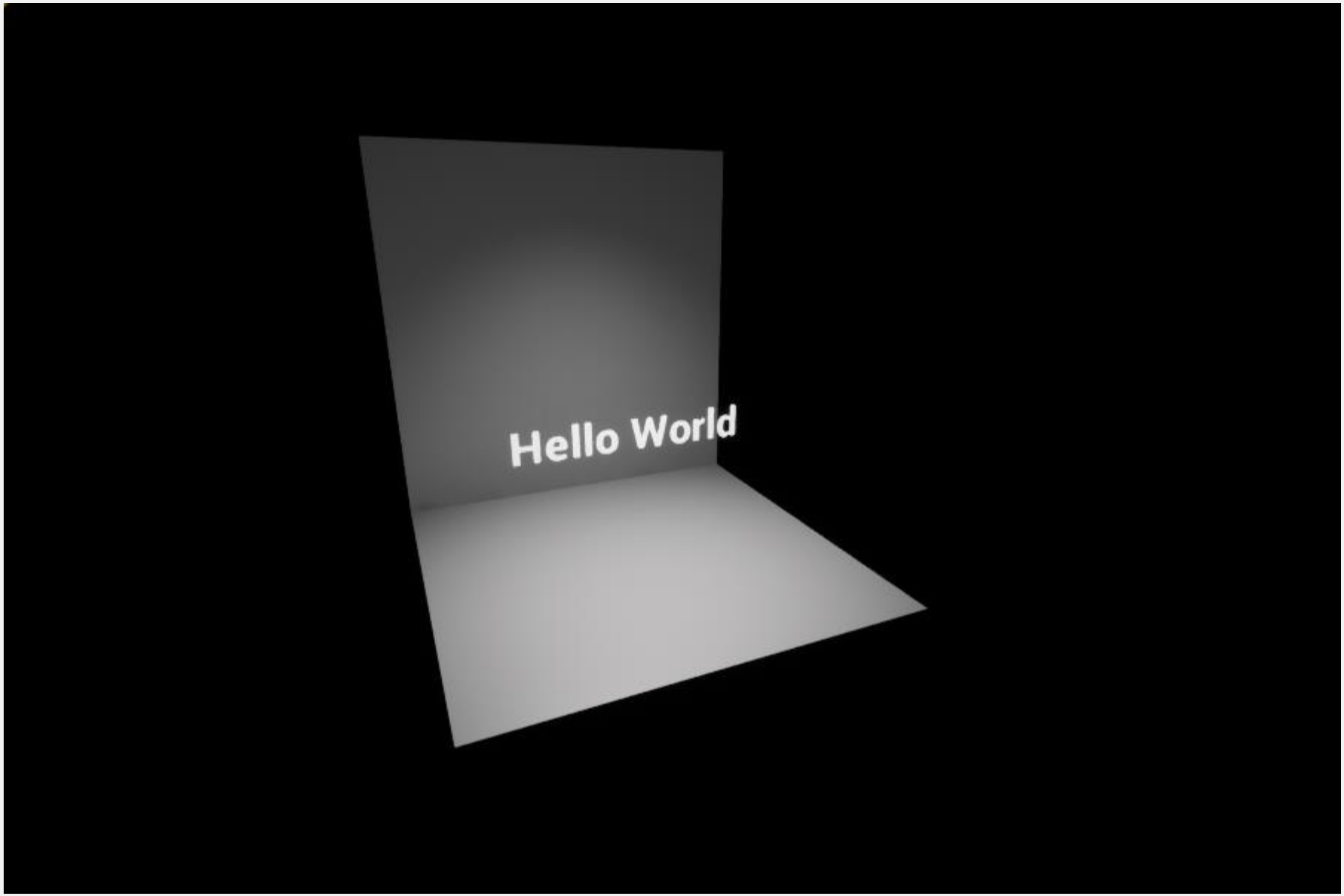


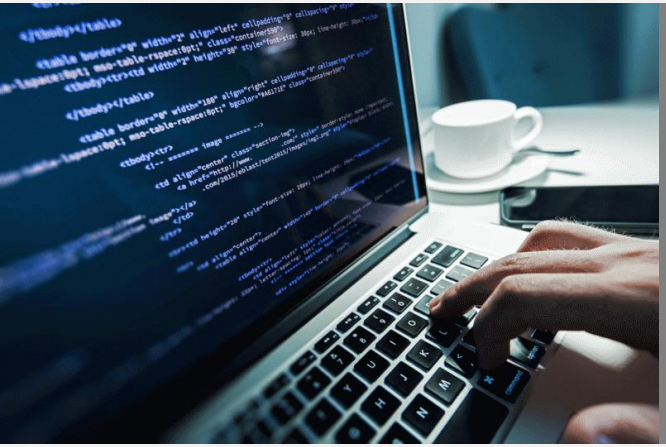

Perspective
Lit
Show
10
10°
0.25
3
Location
30.0
0.0
200.0
Rotation
0.0 °
-90.0
0.0 °
Location
0.0
0.0
50.0
Location
0.0
0.0
0.0
Rotation
0.0 °
0.0 °
0.0 °
Scale
2.0
2.0
1.0
Level: HelloWorld (Persistent)

Actor Merging



Acter Merging

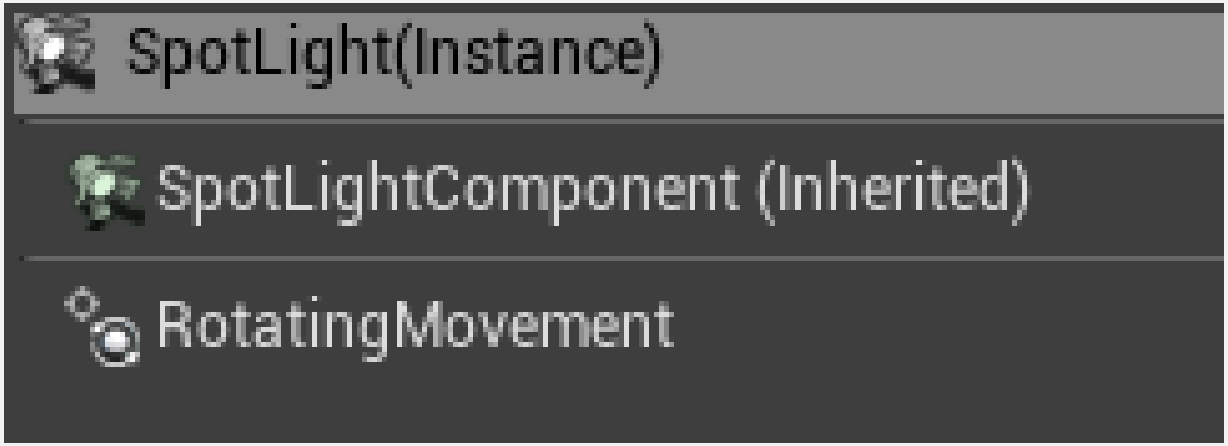

Hello World



실습

라이트 회전

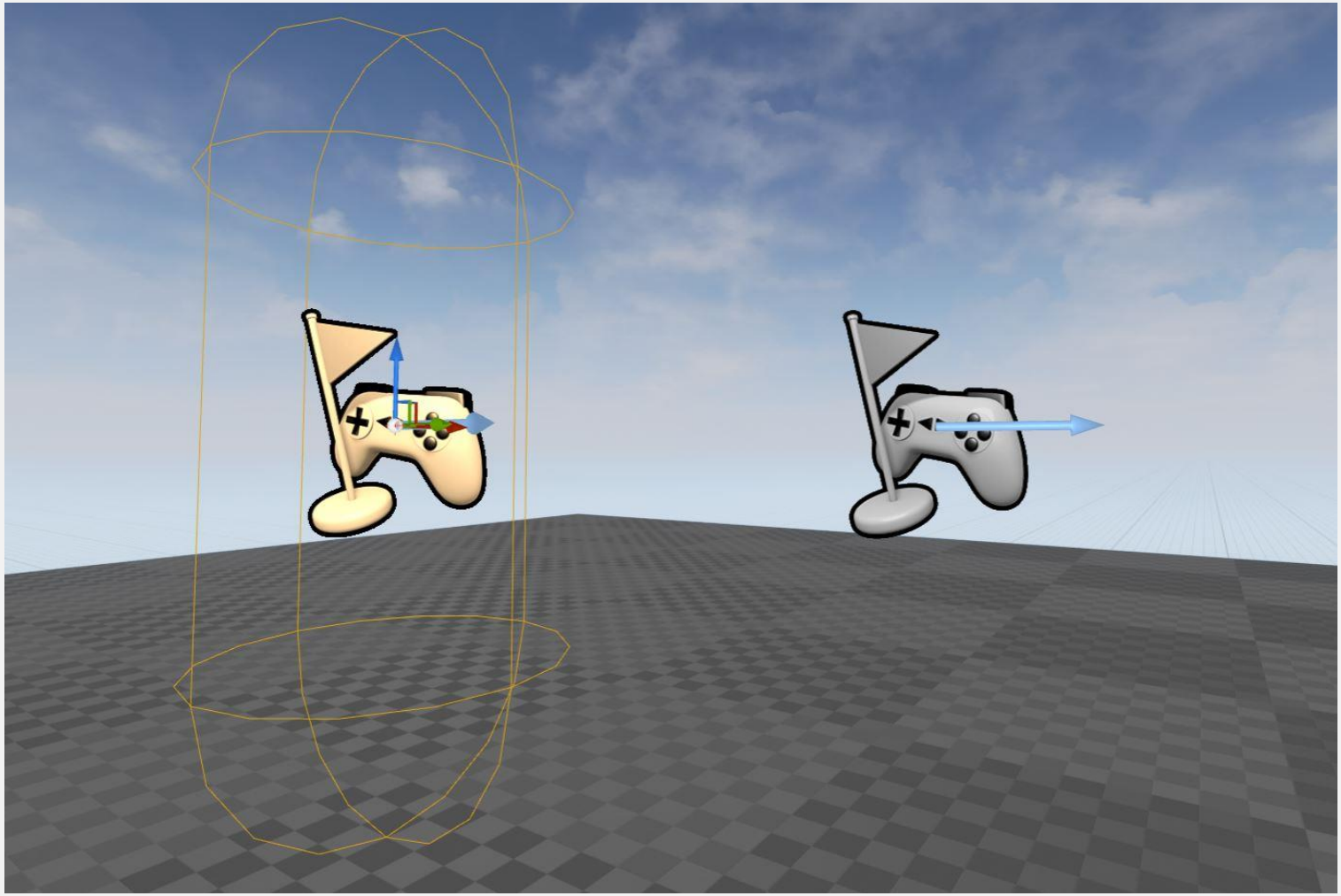
RotatingMovement 컴포넌트 추가

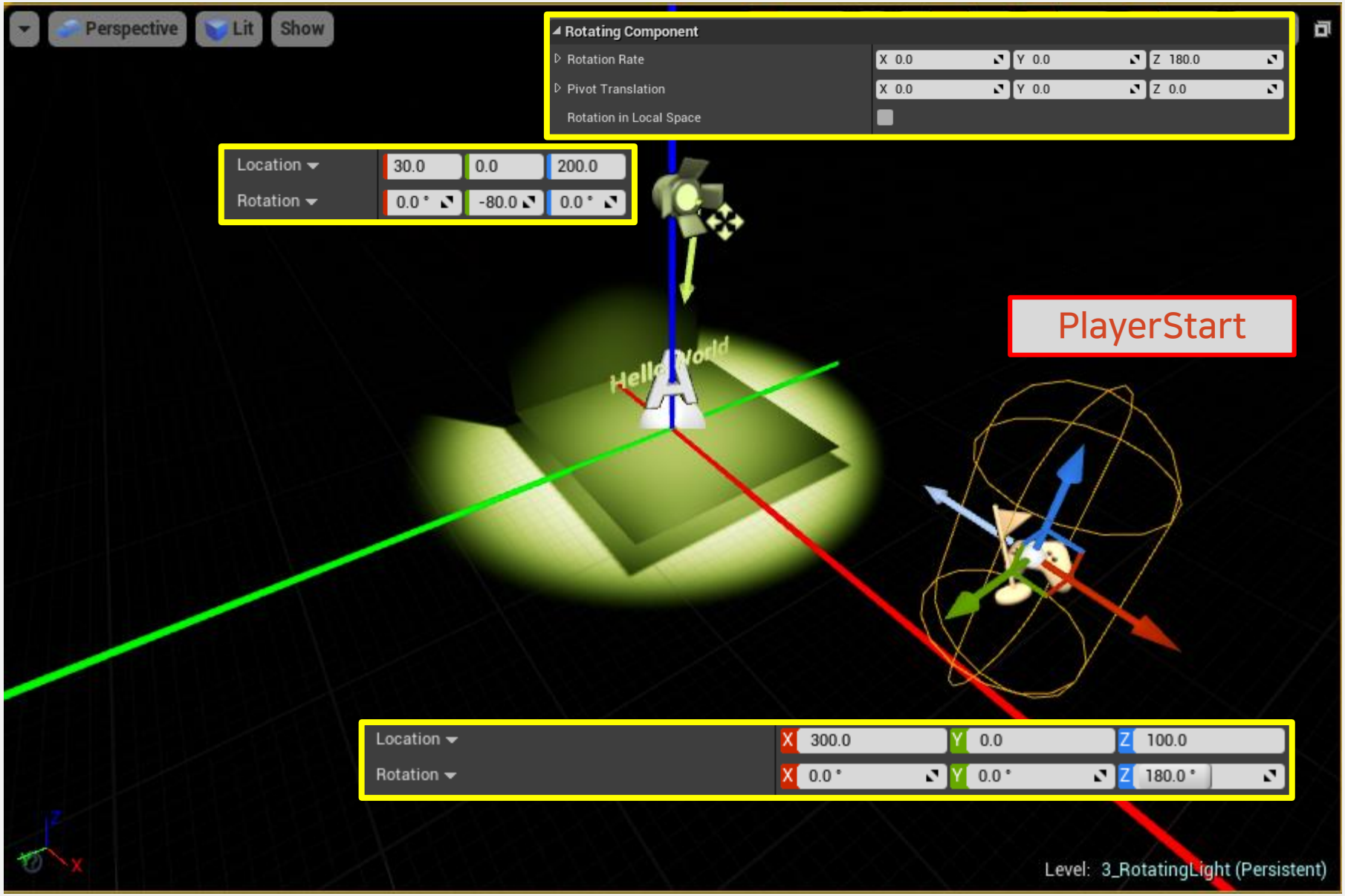


Light를 “Movable” 설정해야 움직일 수 있음.

PlayerStart

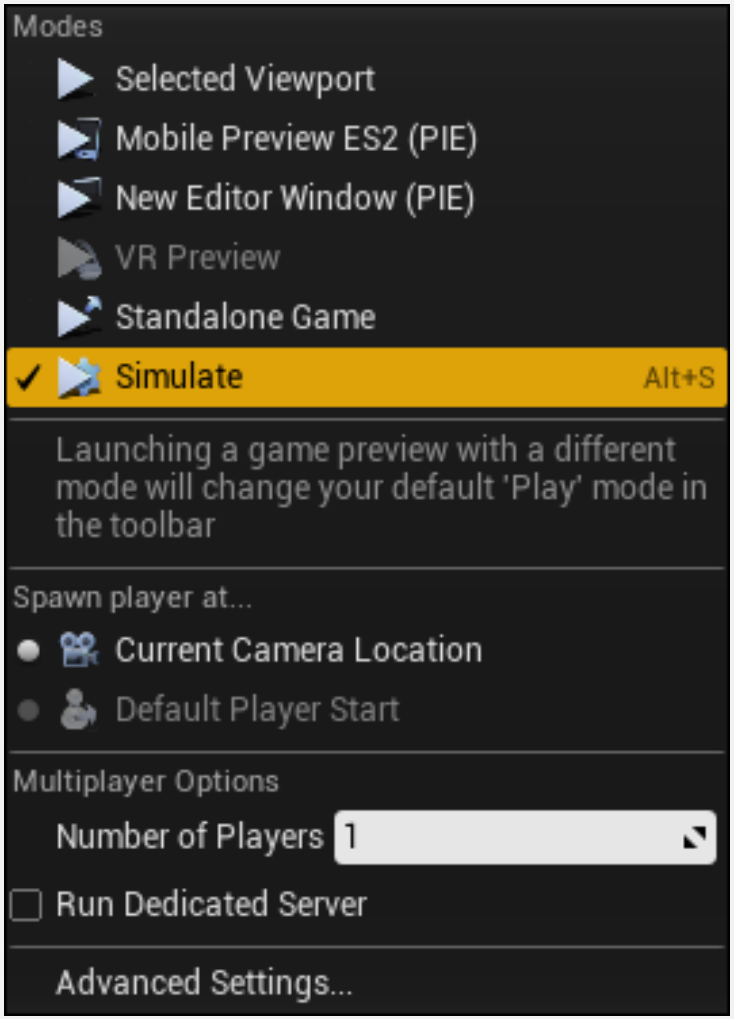
- Pawn 의 시작 위치를 지정




Perspective
Lit
Show
Rotating Component
Rotation Rate
X 0.0
Y 0.0
Z 180.0
Pivot Translation
X 0.0
Y 0.0
Z 0.0
Rotation in Local Space
Location
30.0
0.0
200.0
Rotation
0.0 °
-80.0
0.0 °
PlayerStart
Location
X 300.0
Y 0.0
Z 100.0
Rotation
X 0.0 °
Y 0.0 °
Z 180.0 °
Level: 3_RotatingLight (Persistent)

Simulate

- 액터 들의 움직임을 미리 시뮬레이션하고 살펴볼 수 있다.
- 에디터 모드에서 액터들을 시뮬레이션 함.

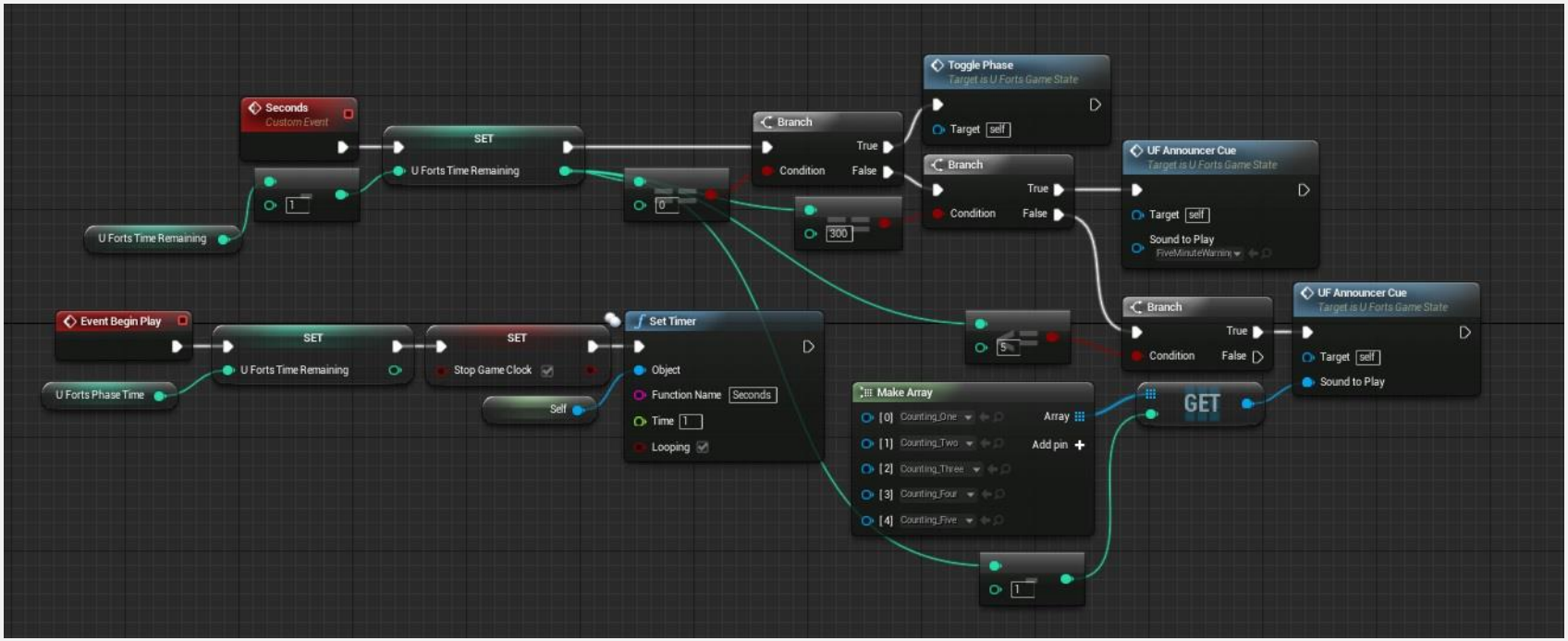


Light를 여러 개 배치하려면?

- 액터를 여러 개 Copy & Paste, 또는 …
- 클래스 블루프린트로 만들어서 인스턴스화

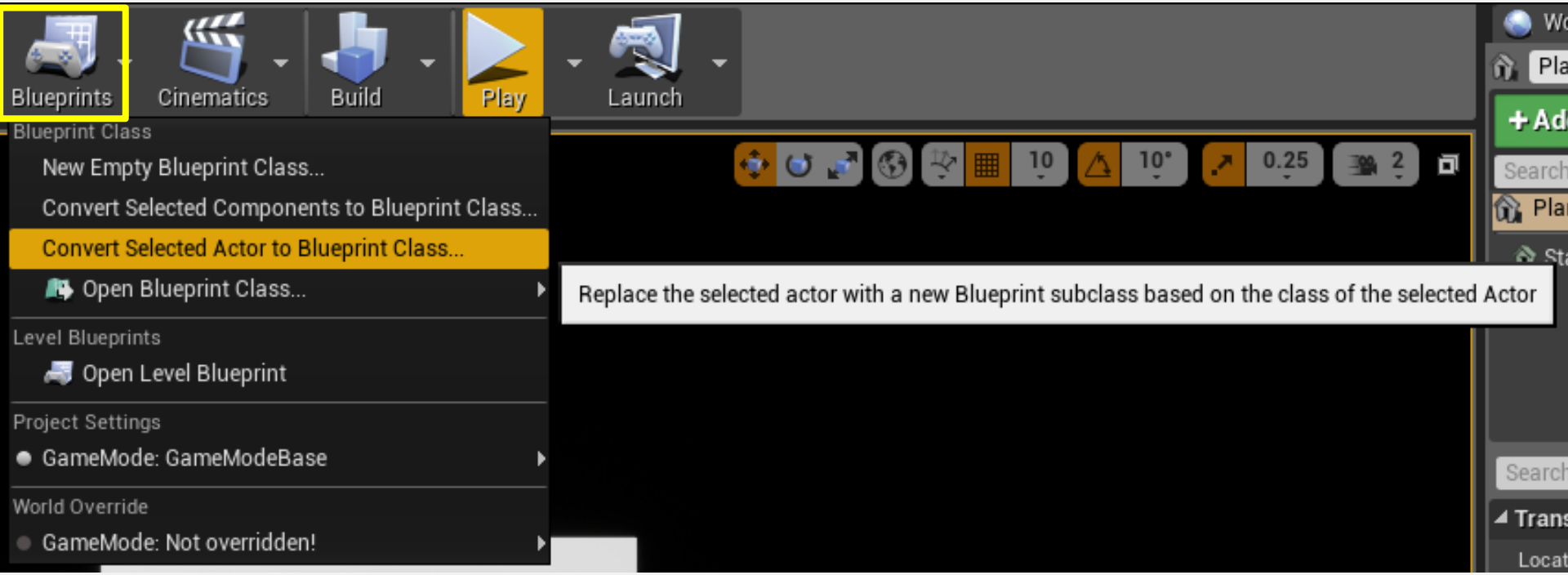
블루프린트 (Blueprint)

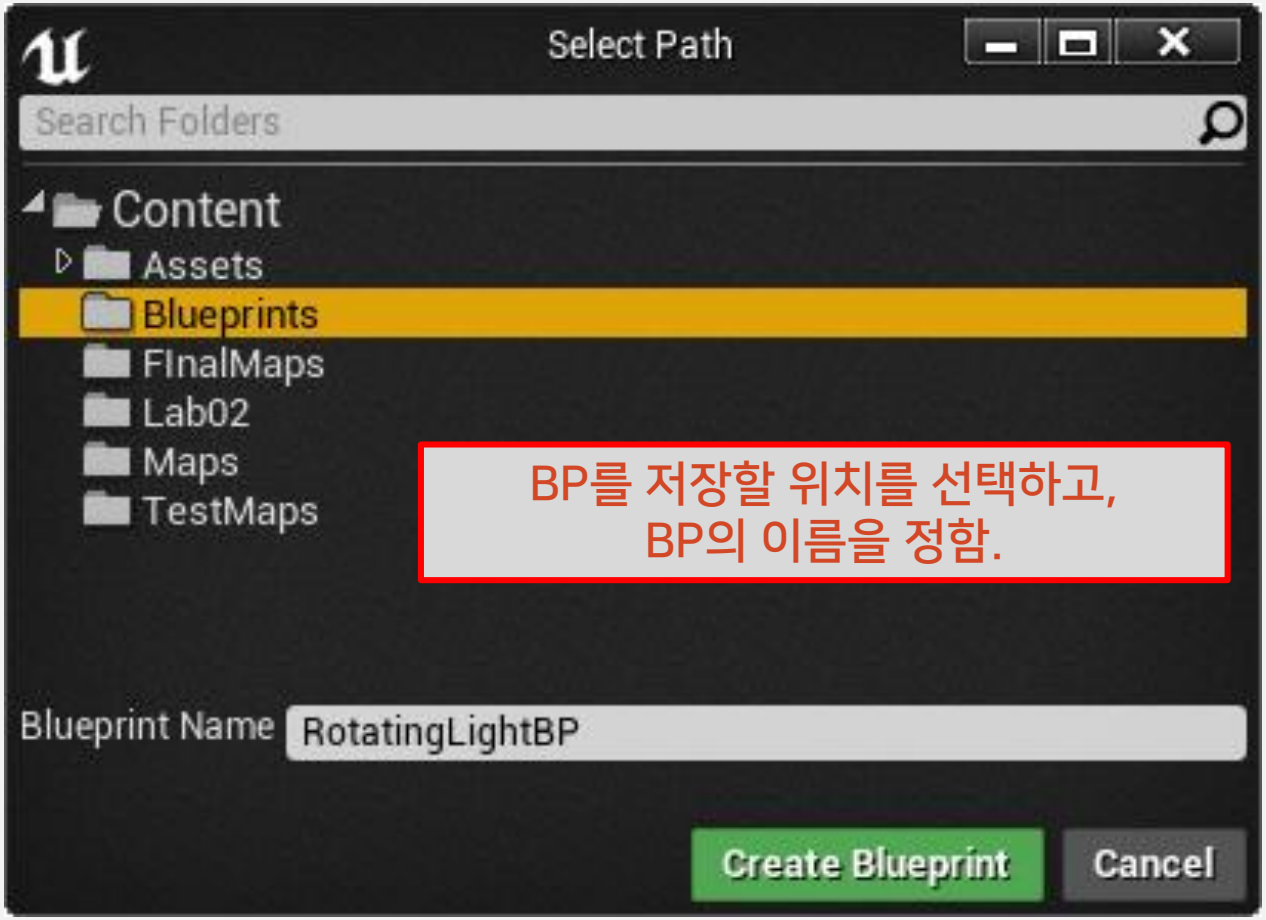
- 언리얼 엔진의 비주얼 스크립팅 언어
- 블루프린트 유형(https://docs.unrealengine.com/latest/KOR/Engine/Blueprints/UserGuide/Types/index.html)
  - Level Blueprint
  - Blueprint Class – 그냥 Blueprint 라고도 함.
  - Data Only Blueprint
  - Blueprint Interface
  - Blueprint Macro Libray



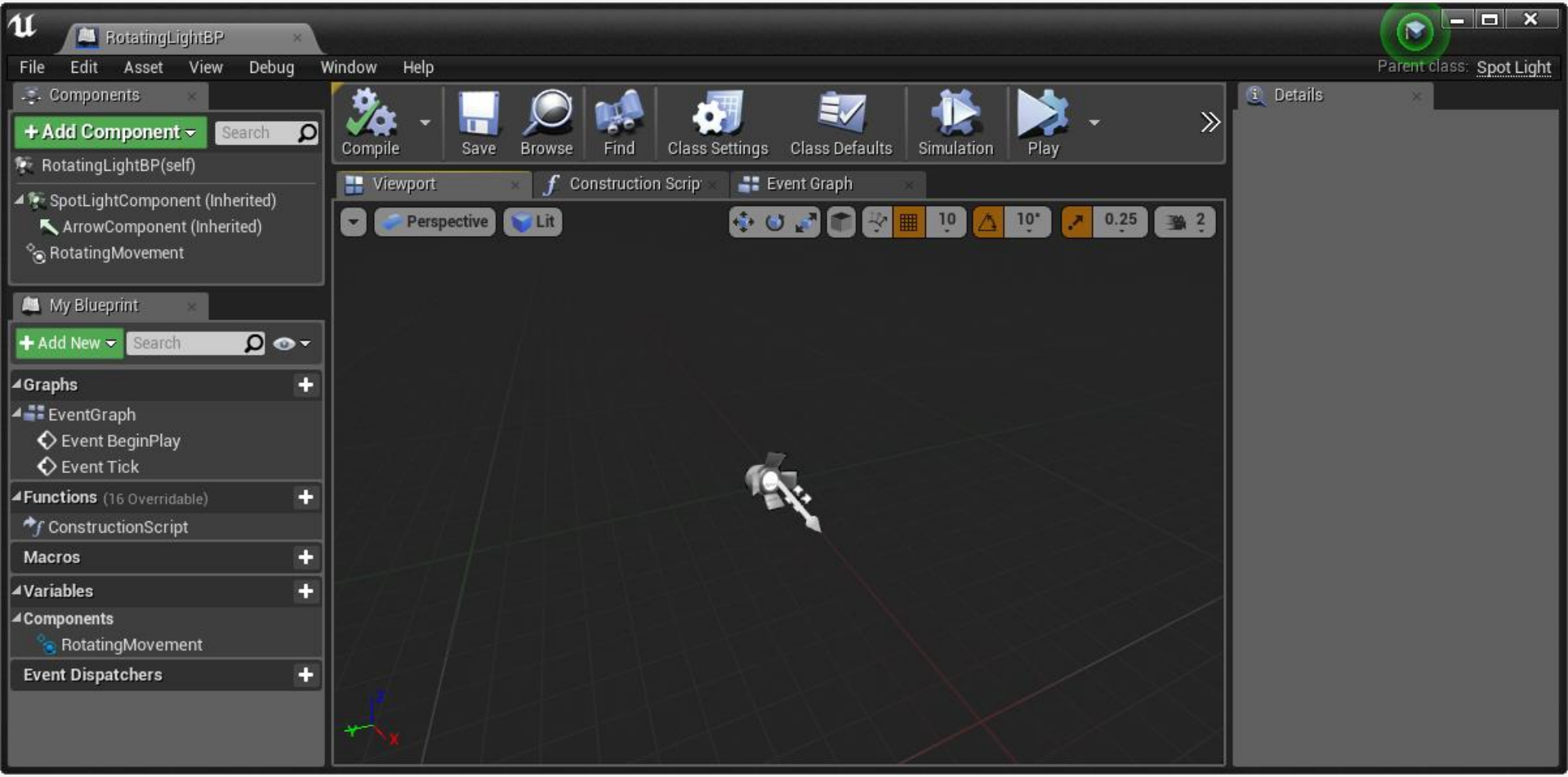
Actor를 Blueprint로 변환법 #1: Subclass 로 만드는 방법

- 선택된 액터의 서브 클래스로 변환함. – 상속이라고 볼 수 있음.
- Transform 이외의 Property를 유지(초기값으로 자동 설정)
- 변환되면서, 기존 액터는 새로운 액터로 바뀜 – 이 때 기존 transform이 반영됨.
- 새로운 인스턴스를 생성하면, transform은 default 로.

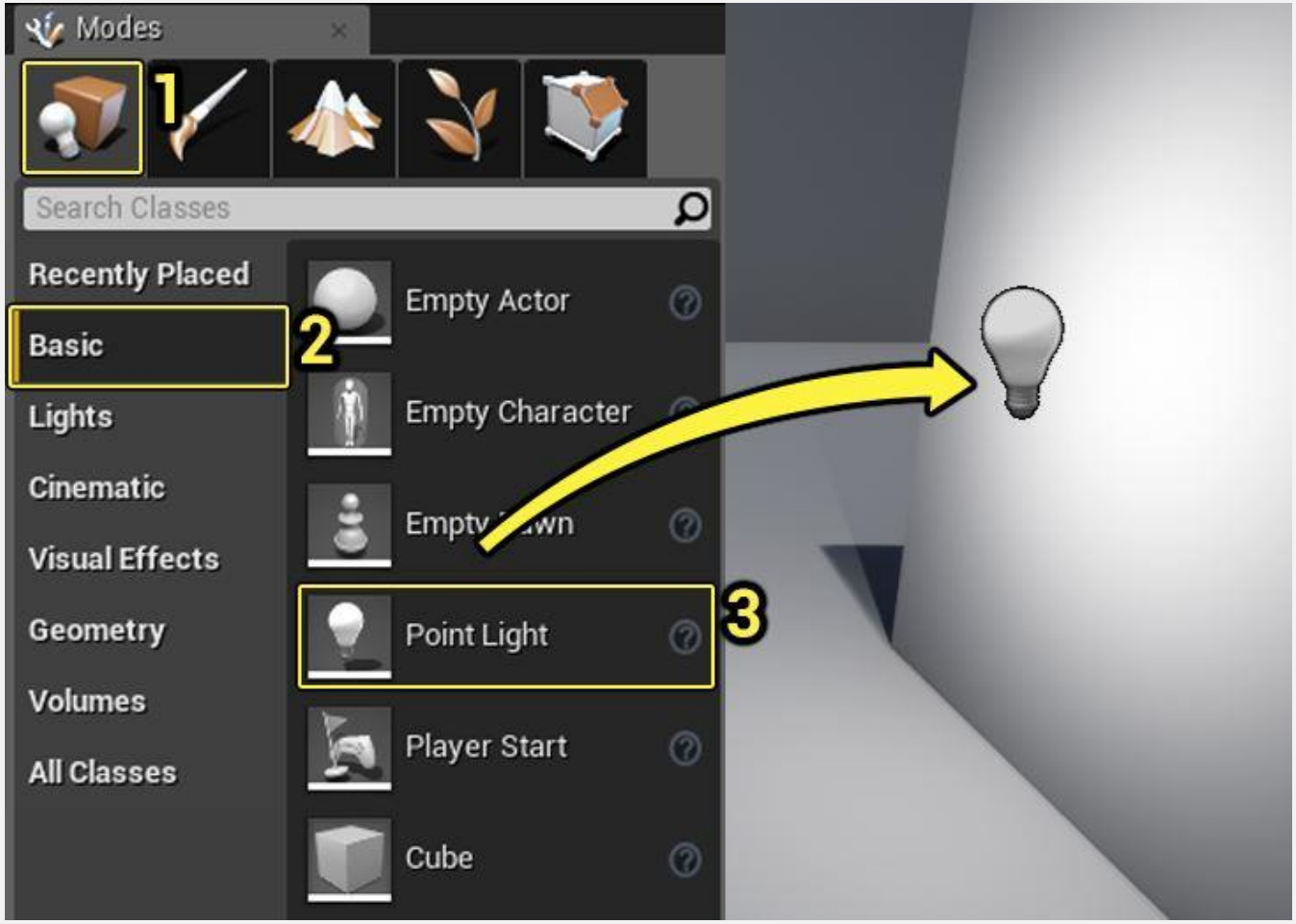



Select Path
Search Folders
Content
Assets
Blueprints
FInalMaps
Lab02
Maps
TestMaps
BP를 저장할 위치를 선택하고,
BP의 이름을 정함.
Blueprint Name
RotatingLightBP
Create Blueprint
Cancel

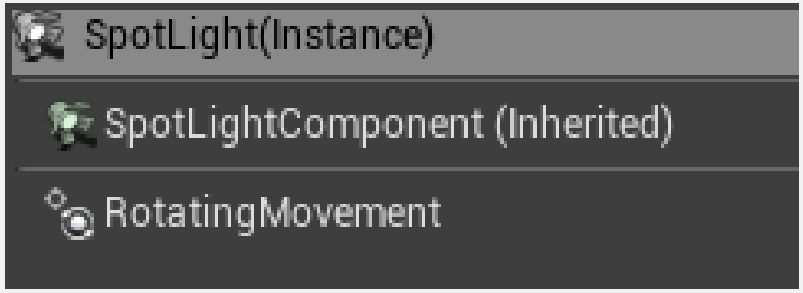
RotatingLightBP



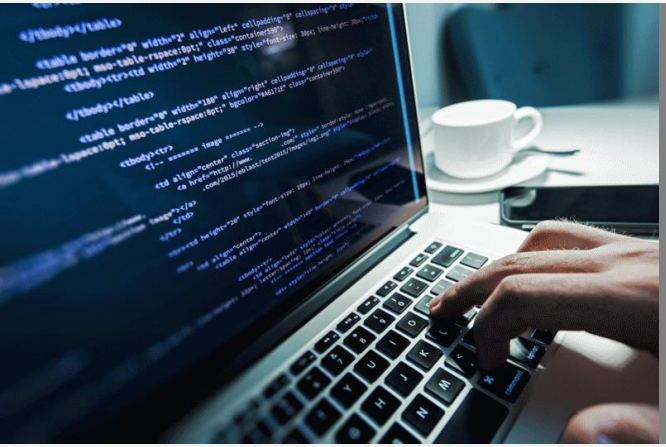
액터 배치 = Object Instantiation



Point Light 를 Drag해서, 뷰포트로 가져오면 Point Light Actor가 생성됨과 동시에 월드에 배치됨.
➔ Point Light 클래스의 오브젝트 인스턴스를 생성하여, 월드에 배치

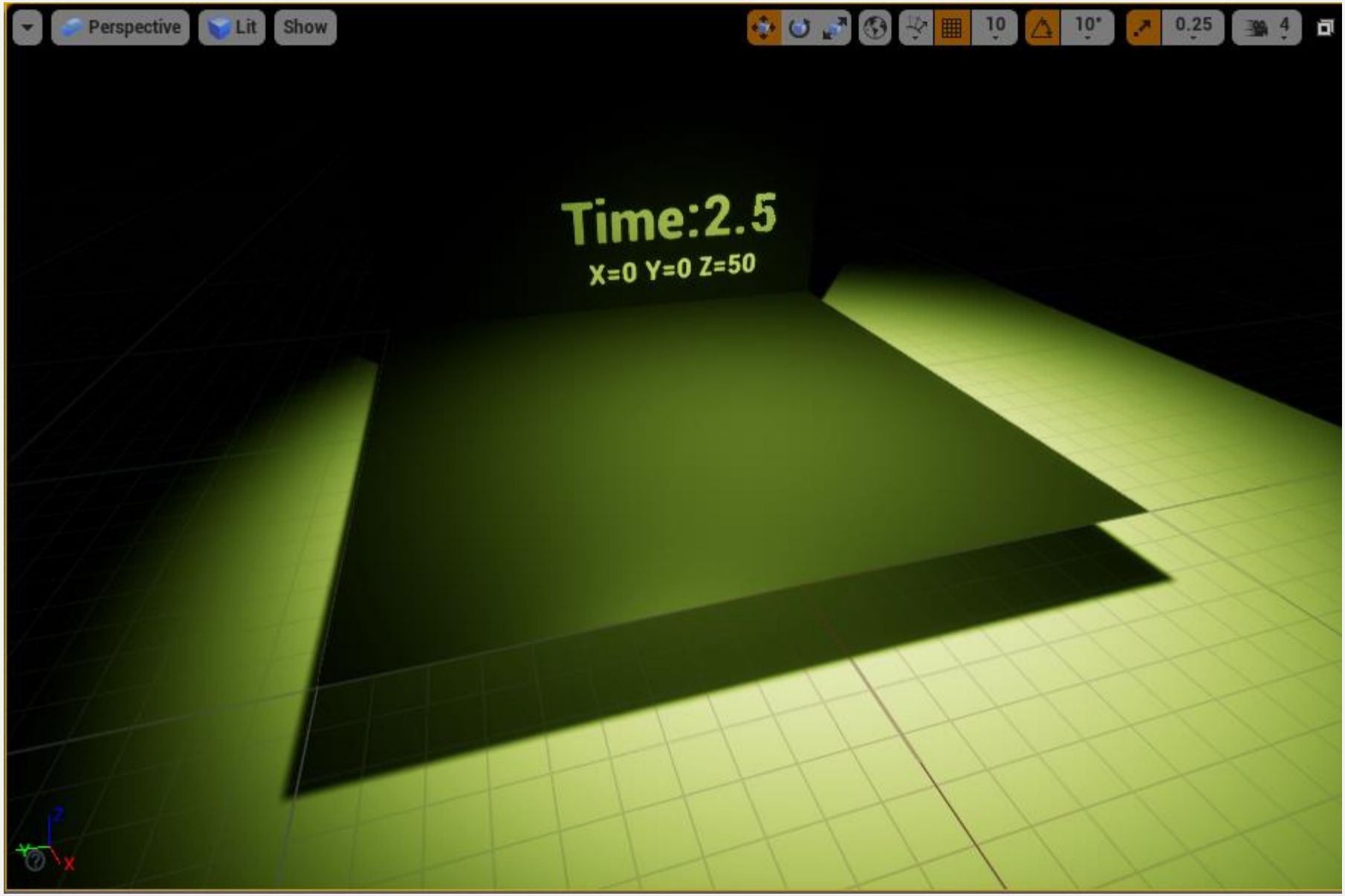



```
class SpotLight
{
    SpotLightComponent *comp;
…
};
```



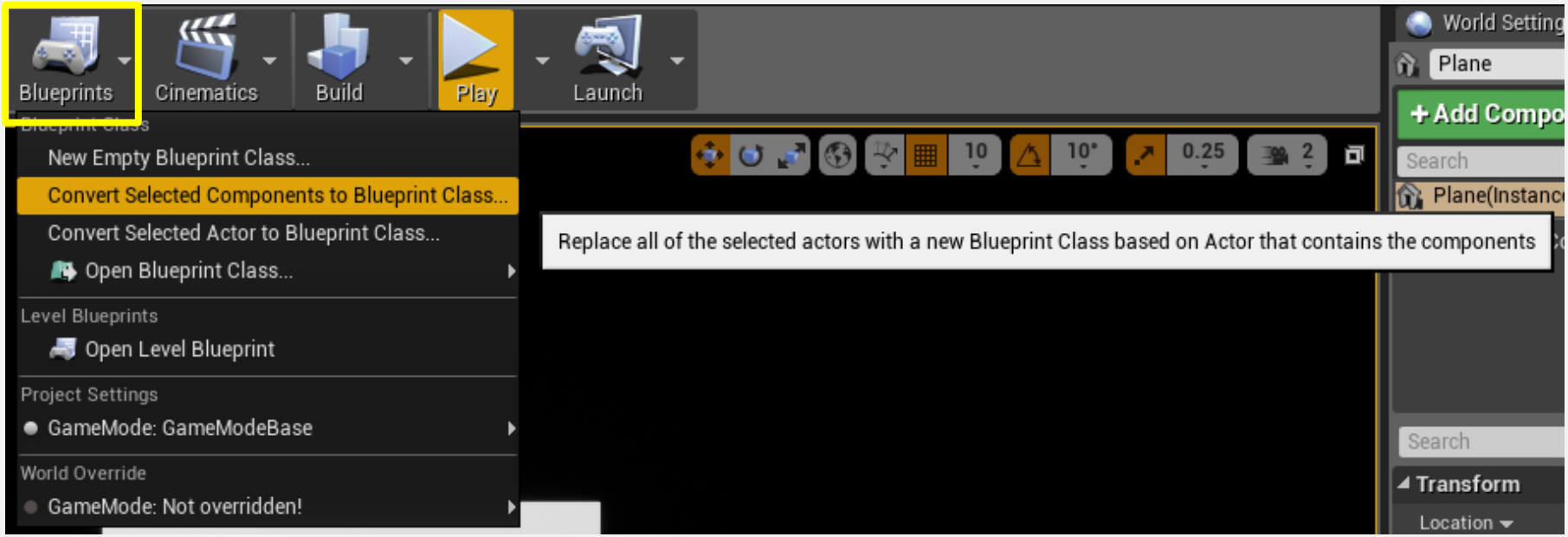
실습

위치 및 경과 시간 표시 텍스트


Perspective
Lit
Show
10
10°
0.25
4
Time:2.5
X=0 Y=0 Z=50

Actor를 Blueprint로 변환법 #2: 컴포넌트로 구성한 BP

- 선택된 액터가 포함하고 있는 component만을 취하는, 새로운 BP class 를 만듬.
- 기존 액터의 component 만을 활용한다는 의미 – compostion 으로 볼 수 있음.
- 변환되면서, 기존 액터는 새로운 액터로 바뀜 – 이 때 기존 transform은 무시되고, transform은 default 값으로 초기화.

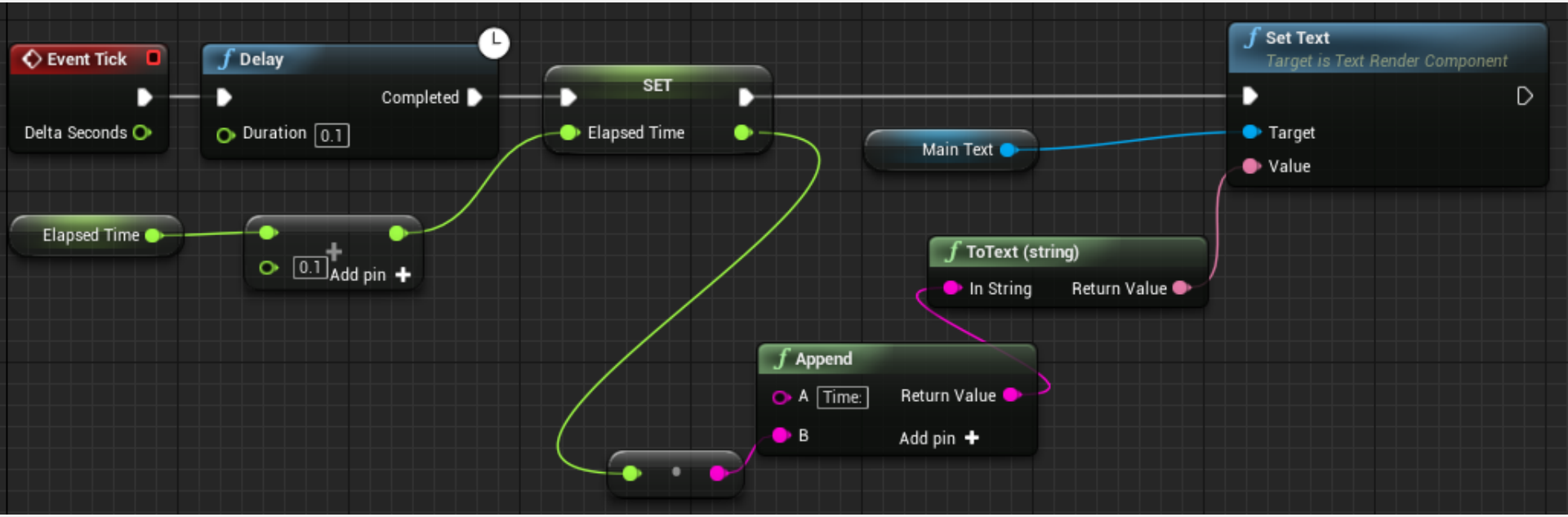


Unreal Engine Game Framework

create actors

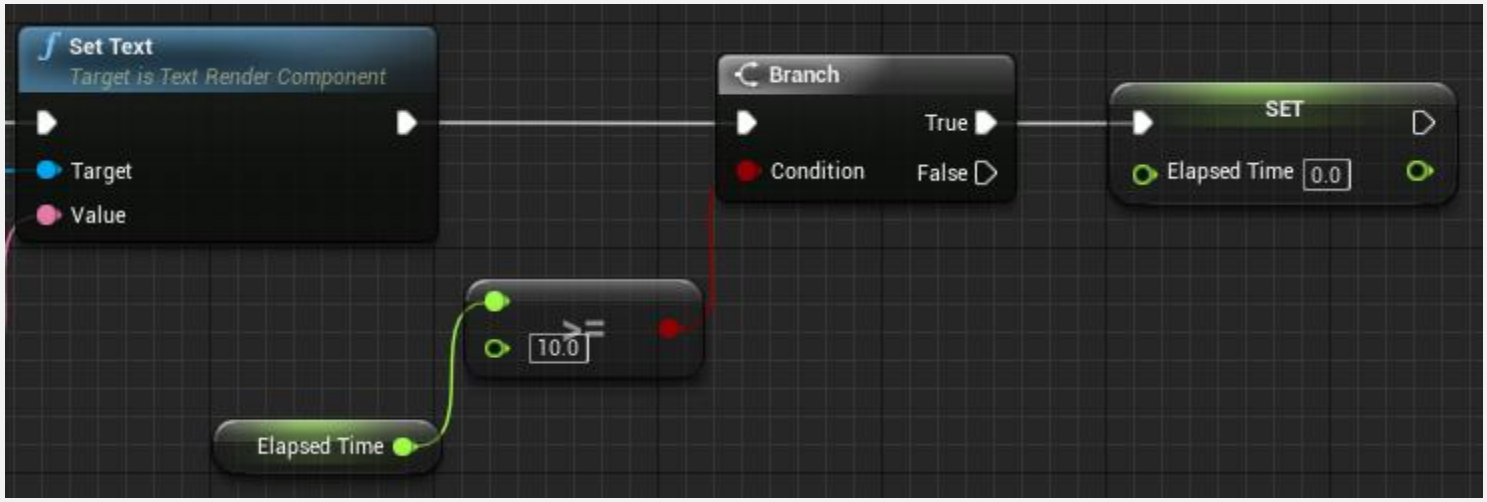
for each actor:
  actor->BeginPlay()

loop
{
  get inputs
  for each actor:
    actor->Tick()
  render()
}

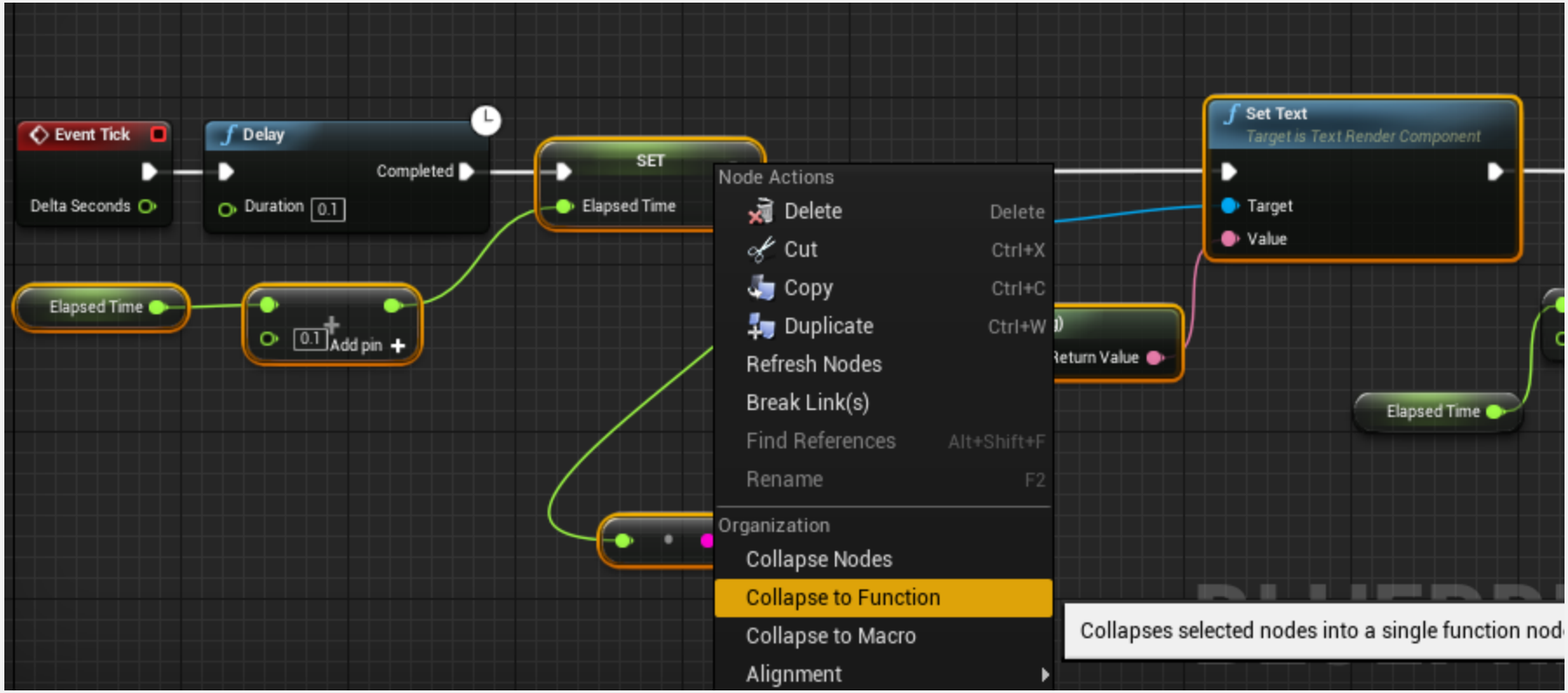


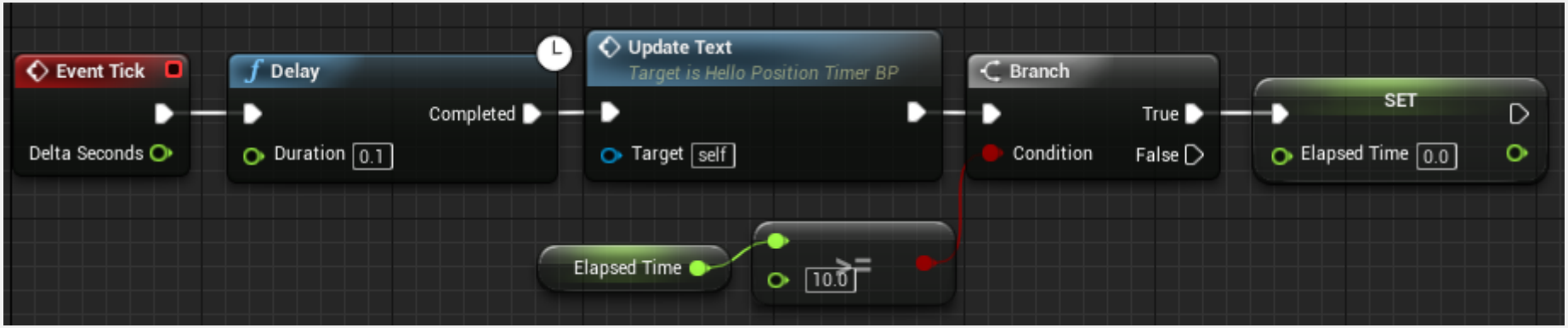
ElapsedTime을 0.1 씩 증가시키고,
그 결과로 Text 를 업데이트

10초 지나면 다시 타이머 리셋



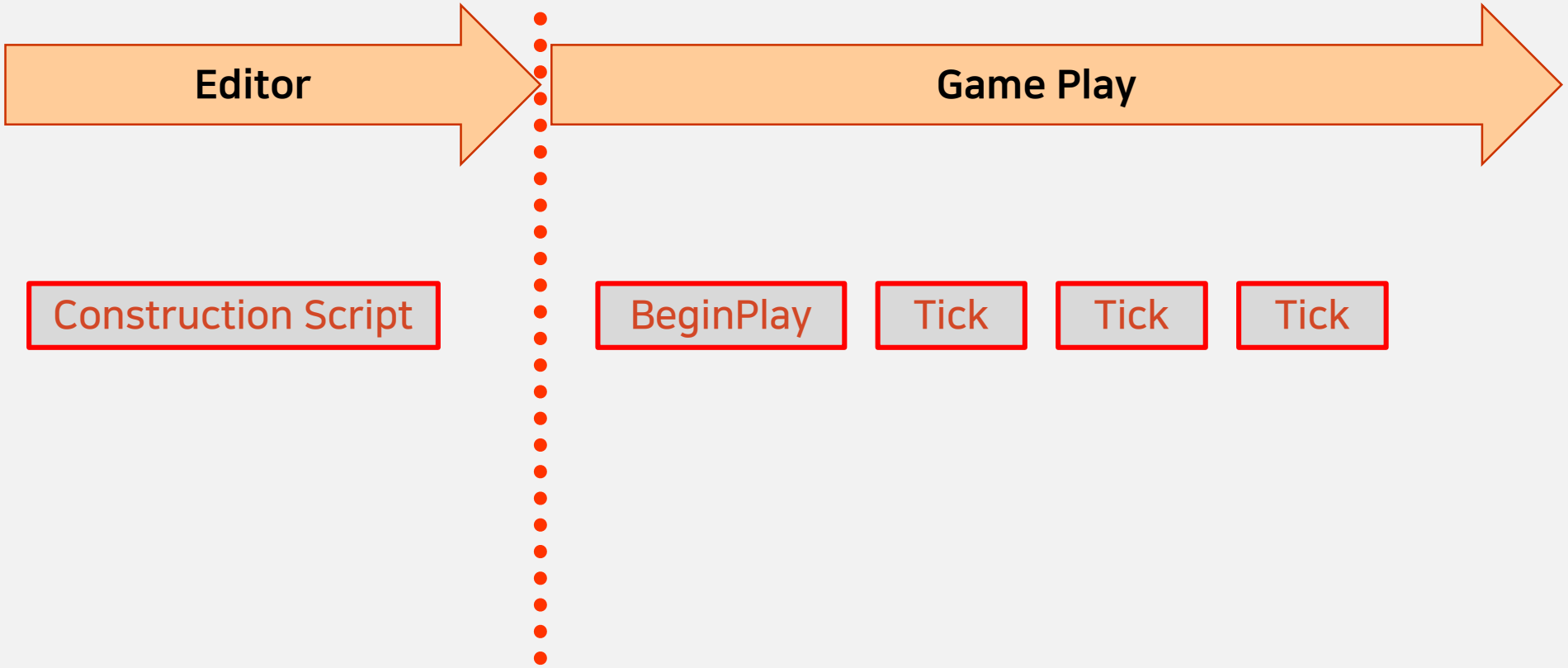
Function 으로 변환



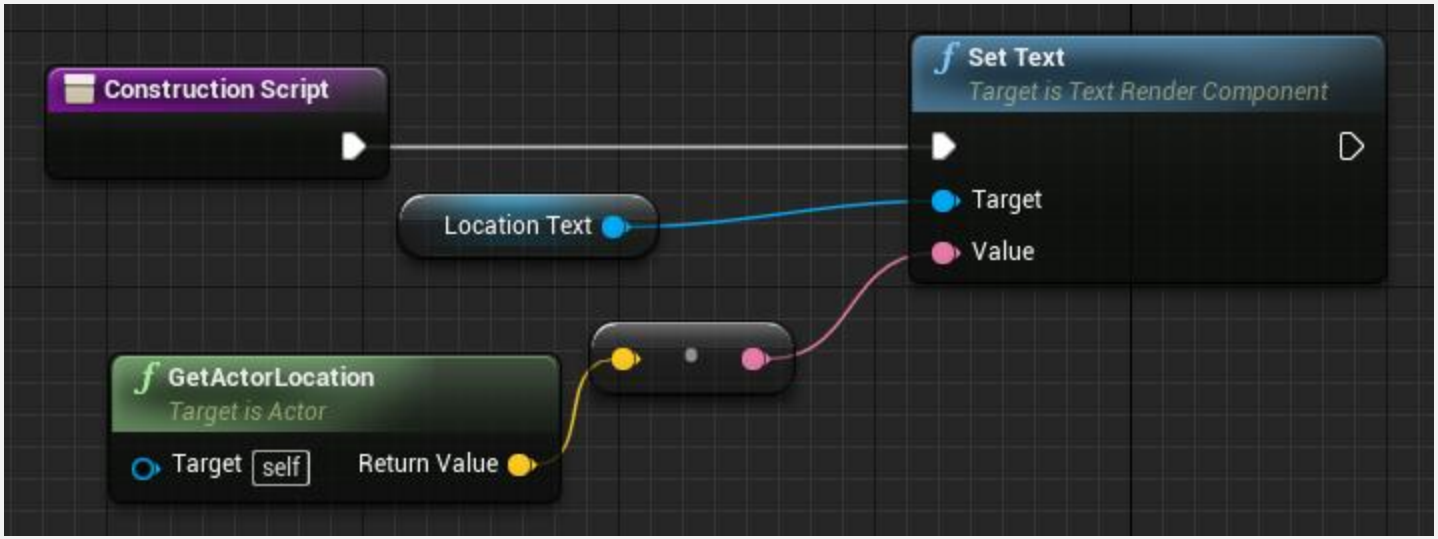

Event Tick
Delta Seconds
Delay
Duration 0.1
Completed
Update Text
Target is Hello Position Timer BP
Target self
Branch
Condition
True
False
SET
Elapsed Time 0.0
Elapsed Time
10.0

Construction Script 와 Event Graph의 실행 관계

- **Construction Script의 Editor 상에서 곧바로 반영**
  - 액터가 생성되는 시점, 또는 액터의 속성 변경이 일어날 때마다 construction script가 실행되어서, 액터 생성 초기화를 수행.
- **Event Graph**
  - 실제로 게임 플레이가 시작되면, BeginPlay event 가 호출되어, 기본적인 초기화를 수행하고, 이어서 게임 루프에서 반복적으로 계속해서 Tick event를 호출함.



Construction Script



Text 와 String

FText

FText represents a display string. Any text you want to display to the user should be handled with FText. The FText class has built-in support for localization, and can handle text content that is localized and stored in a lookup table, as well as text that is localized at runtime, such as numbers, dates, times, and formatted text. Even text that does not need to be localized can be handled with FText. This includes user-entered content such as a player's name, and any text displayed with Slate. FText does not offer any mutation functions, because making changes to display strings is a very unsafe operation.

- FText Reference Guide

FString

Unlike FName and FText, **FString** is the only string class that allows for manipulation. There are many methods available for string manipulation, including case changing, excerpting substrings, and reversing. FStrings can be searched, modified, and compared against other strings. However, these manipulations can make FStrings more expensive than the immutable string classes.

- FString Reference Guide