UNREAL ENGINE

레퍼런스만 알면 언리얼 엔진이 제대로 보인다.

부제 : 남이 만든 코드 복붙은 이제 그만

프롤로그(Prologue)

레퍼런스(Reference)란 무엇인가?

- 프로그래밍 : 언리얼 오브젝트를 참조하는 안전한 C++ 포인터
- 블루프린트 : 물체에 명령을 전달할 수 있는 핀을 가진 노드

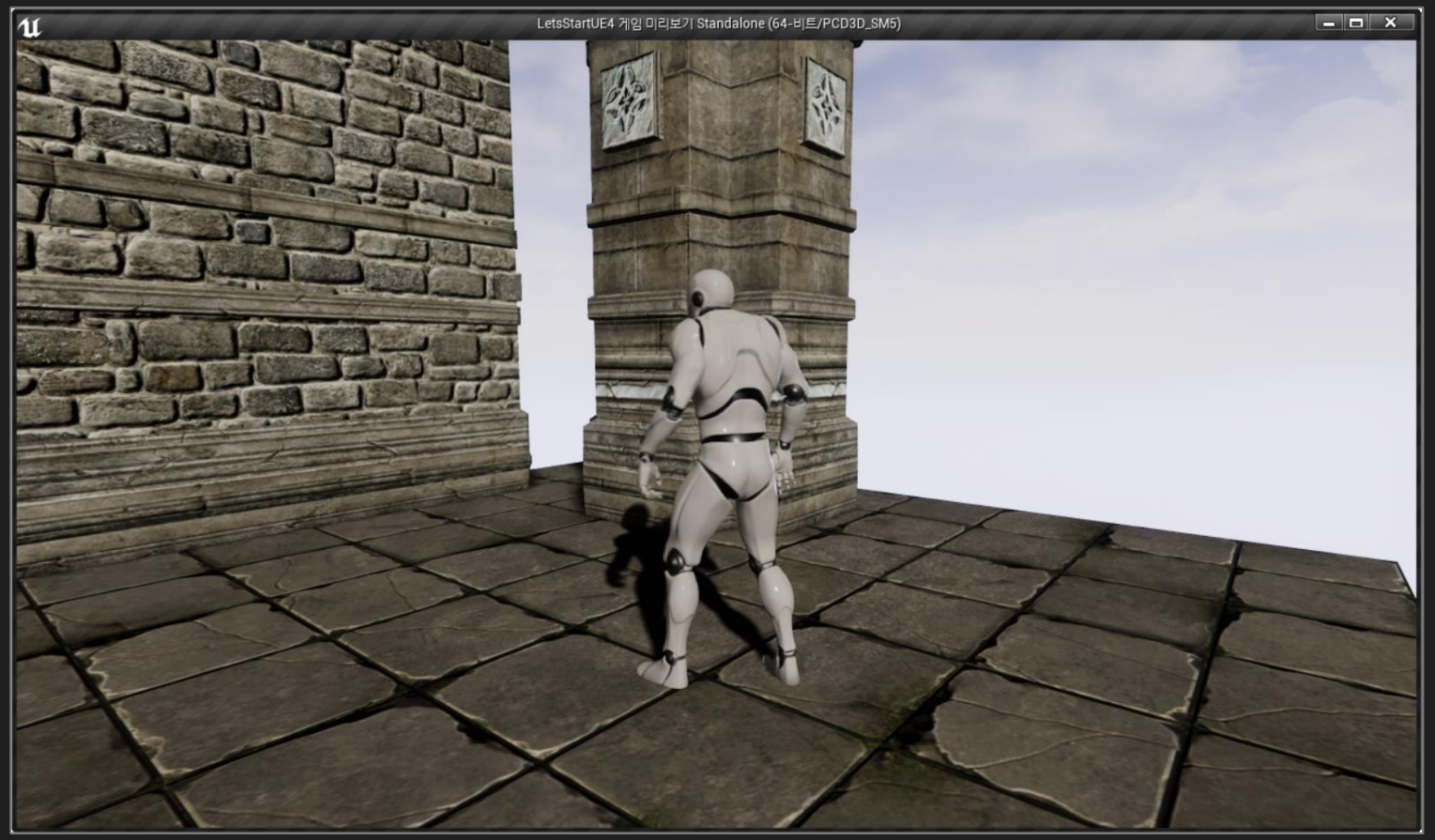
어떻게 명령을 내리는 것 보다
먼저 누구에게 명령을 내리는지가 중요.

강연의 전체 구성

1. 월드와 레벨 – 1교시
2. 액터의 설계 – 2교시
3. 뷰 타겟과 컨트롤 회전 – 2교시
4. 애니메이션 블루프린트 – 3교시
5. 애니메이션 노티파이 – 3교시

게임 콘텐츠의 구성 요소

튜토리얼 시작 화면



게임을 시작하면 무슨 일이 벌어지는가?

1. 월드가 생성된다.
2. 월드에 레벨을 로딩한다.
3. 지정한 규칙에 따라 로컬 플레이어를 입장시킨다.
4. 플레이어에 관련된 다양한 물체가 생성된다.

언리얼 엔진은 기본적으로 제공하는 기능이 많다.

콘텐츠 제작 환경 설정 – 월드

공간 (Transform)

시간 (Time)

물리 (Physics , Collision)

언리얼 엔진에서 월드(World)란?

1. 게임 콘텐츠를 담는 가상 공간.
2. 가상 공간에 시간과 물리법칙을 부여
3. 가상 공간에 적용할 여러 기능들을 설정 (길찾기 / 라이트맵 등등.)

콘텐츠 제작의 시작 – 액터의 배치

월드에 배치된 바닥 액터의 역할

1. 월드의 어딘가에 존재해야 한다. (트랜스폼)
2. 바닥의 비주얼을 보여준다. (비주얼)
3. 다른 물체가 떨어지지 않도록 받쳐준다. (충돌)

언리얼 엔진에서 액터(Actor)란?

1. 가상 공간에 존재하는 물체 = 트랜스폼을 가지는 물체

   모든 액터는 트랜스폼을 가진다.

2. 하나의 독립된 역할을 부여 받아 스스로 동작한다.

   모든 액터는 설계된 역할이 있으며, 월드에 배치되면 스스로 동작한다.

트랜스폼이 없는 액터는 존재하지 않는다.

콘텐츠의 고도화 – 레벨의 형성

다양한 역할의 액터를 배치해 게임에 필요한 스테이지를 생성한다.
언리얼 엔진에서는 이러한 액터의 집합을 레벨이라고 한다.

언리얼 엔진에서 레벨(Level)이란?

1. 개발자가 게임 제작을 위해 배치한 액터의 묶음
2. 축구장이 없으면 축구 게임을 못 하듯이, 레벨이 없으면 게임을 할 수가 없다!
3. 언리얼 엔진은 월드가 생성될 때, 기본 레벨을 로딩하도록 설계되어 있다.
4. 이를 퍼시스턴트 레벨(Persistent Level) 이라고 함.

언리얼 엔진에서 월드는 항상 퍼시스턴트 레벨을 가진다.

언리얼 엔진의 레벨 시스템

1. 퍼시스턴트 레벨과 서브(스트리밍) 레벨로 나뉜다.
    1. 퍼시스턴트 레벨은 월드에 포함되는 고정 레벨
    2. 서브 레벨은 필요에 따라 붙였다 뗏다 할 수 있는 레벨
2. 퍼시스턴트 레벨에 여러 개의 서브 레벨을 부착할 수 있다.
    1. 필요에 따라 초기 로딩하도록 설정이 가능.
    2. 서브 레벨의 원점 트랜스폼을 조절하면 다양하게 활용하는 것이 가능

레벨 시스템 데모

언리얼 엔진의 액터의 구성

1. 액터는 컴포넌트들로 구성되어 있다.
   - 컴포넌트는 언리얼 엔진의 기능을 규격화한 모듈
   - 다양한 종류의 컴포넌트를 결합해 액터의 역할을 설계
2. 루트 컴포넌트(Root Component)가 액터를 대표한다.
   - 액터의 위치 = 루트 컴포넌트의 위치
   - 액터의 위치와 물리적인 설정을 루트 컴포넌트가 대표한다

액터의 제작 데모

액터에 움직임을 부여하는 여러 가지 방법

1. 틱에 따라 트랜스폼을 수동으로 조정
2. 물리 시뮬레이션에 움직임을 위임
3. 길찾기에 움직임을 위임
4. 애니메이션 루트 모션에 따라 이동
5. 무브먼트 컴포넌트를 활용

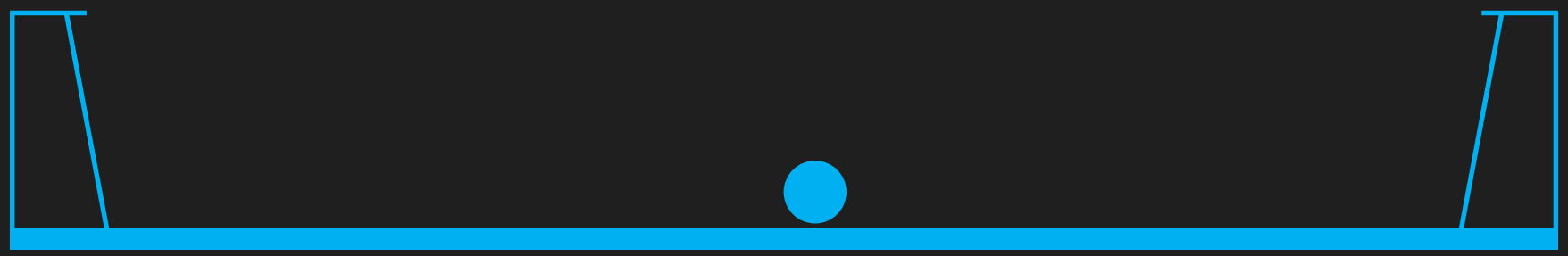
무브먼트 컴포넌트 데모

월드, 레벨, 액터, 컴포넌트의 관계

게임 콘텐츠의 시작 – 게임 규칙의 설정

게임의 승패를 가리기 위한 게임의 규칙이 있어야 함

(전반 45분, 후반 45분, 골 많이 넣는 사람이 Win)

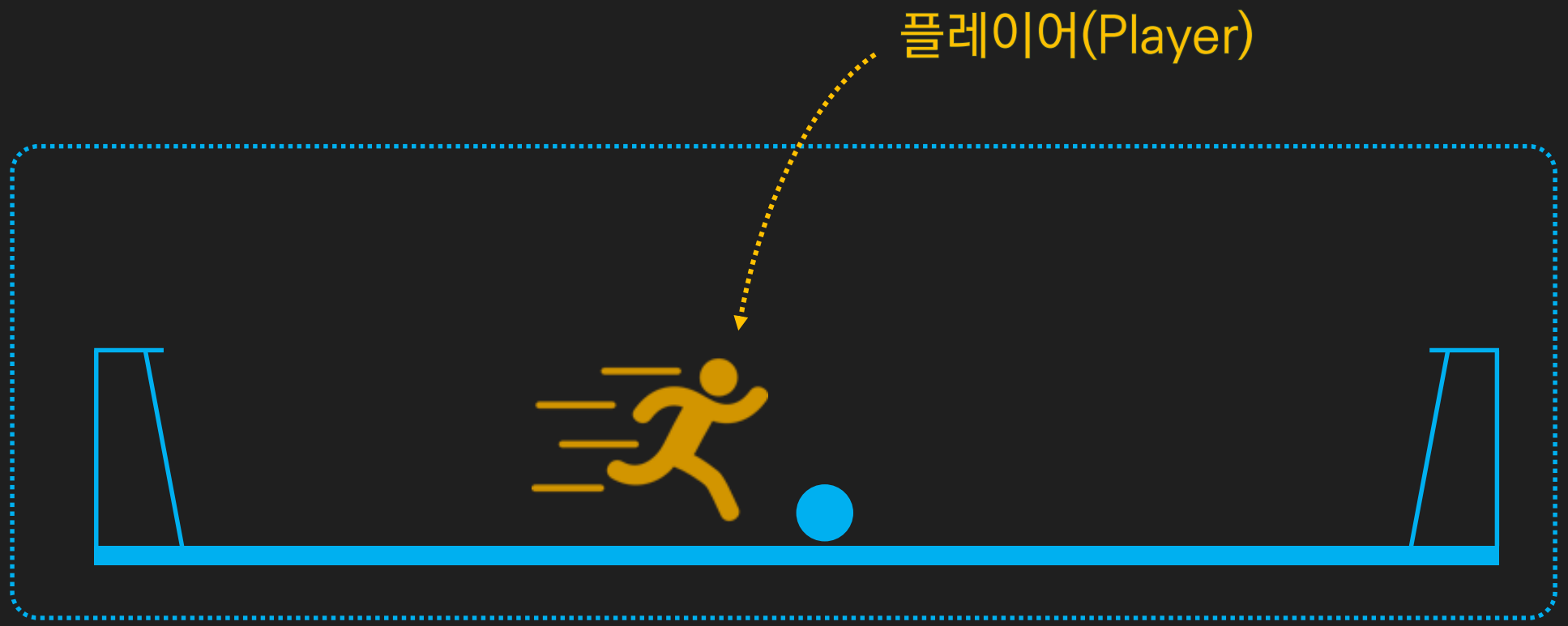


언리얼 엔진에서 게임모드(GameMode)란?

1. 게임의 시작과 끝을 설정해주는 관리자.
2. 게임 중 발생하는 문제를 조정해주는 보이지 않는 심판.
3. 하나의 스테이지에 다른 룰을 적용할 수 있다.

시뮬레이션 콘텐츠에서 게임 컨텐츠로 발전하는 첫 관문!

게임의 시작 – 플레이어의 입장



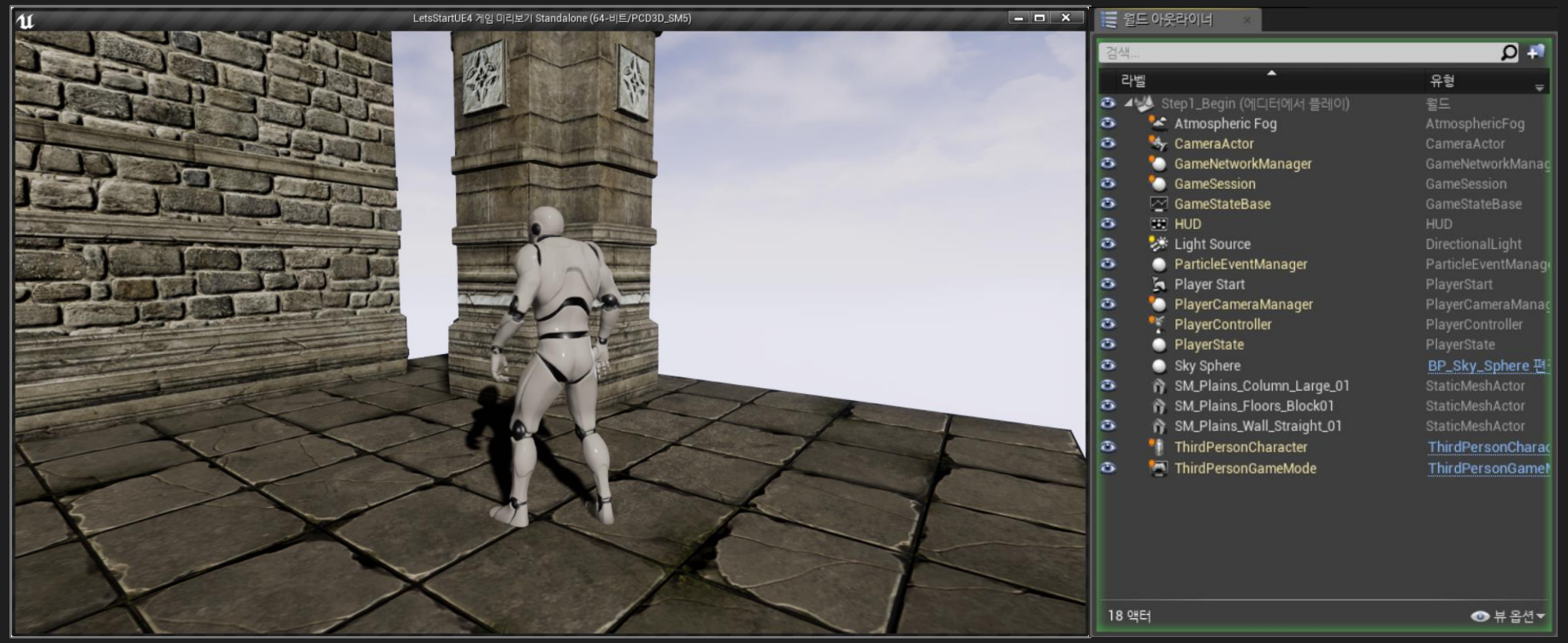
언리얼 엔진에서 플레이어(Player)란?

1. 컨텐츠를 이용하기 위해 접속한 **현실 세계**의 사람
2. 게임에 설정된 입력 시스템을 매개로 실시간으로 콘텐츠를 조작하고 생성.
3. 언리얼 엔진은 플레이어 별로 독립된 디바이스를 가지도록 설계되어 있음.
4. 게임에 입장하는 플레이어들은 모두 평등하고 같은 조건을 가진다.

   예) U-17 월드컵과 일반 월드컵은 따로 열린다.

플레이어를 대변할 수 있는 액터가 필요

플레이 버튼을 누른 이후의 변화

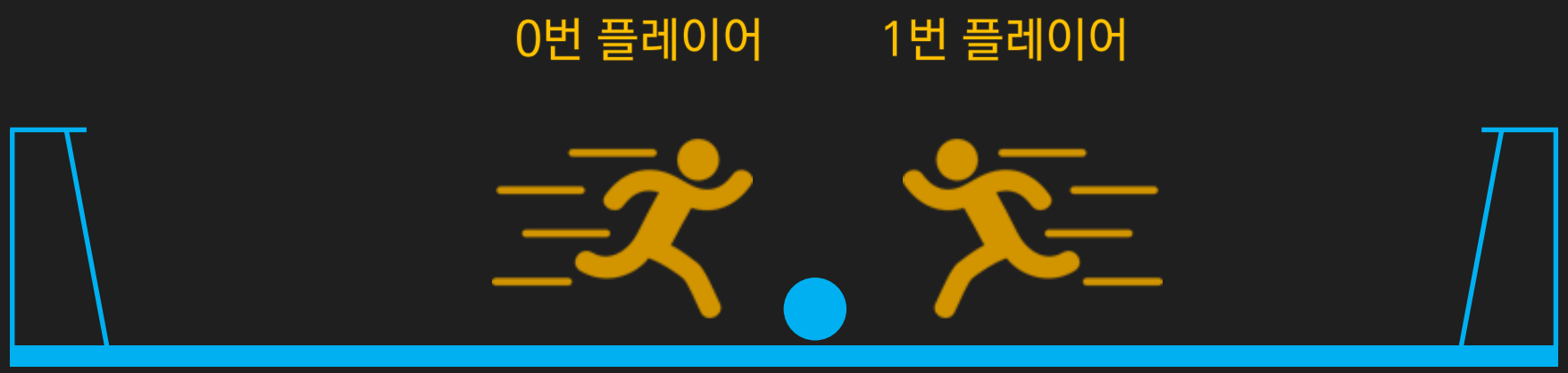


흰색은 레벨 액터 / 노란색은 게임 플레이 관련 액터

플레이어의 추가 입장

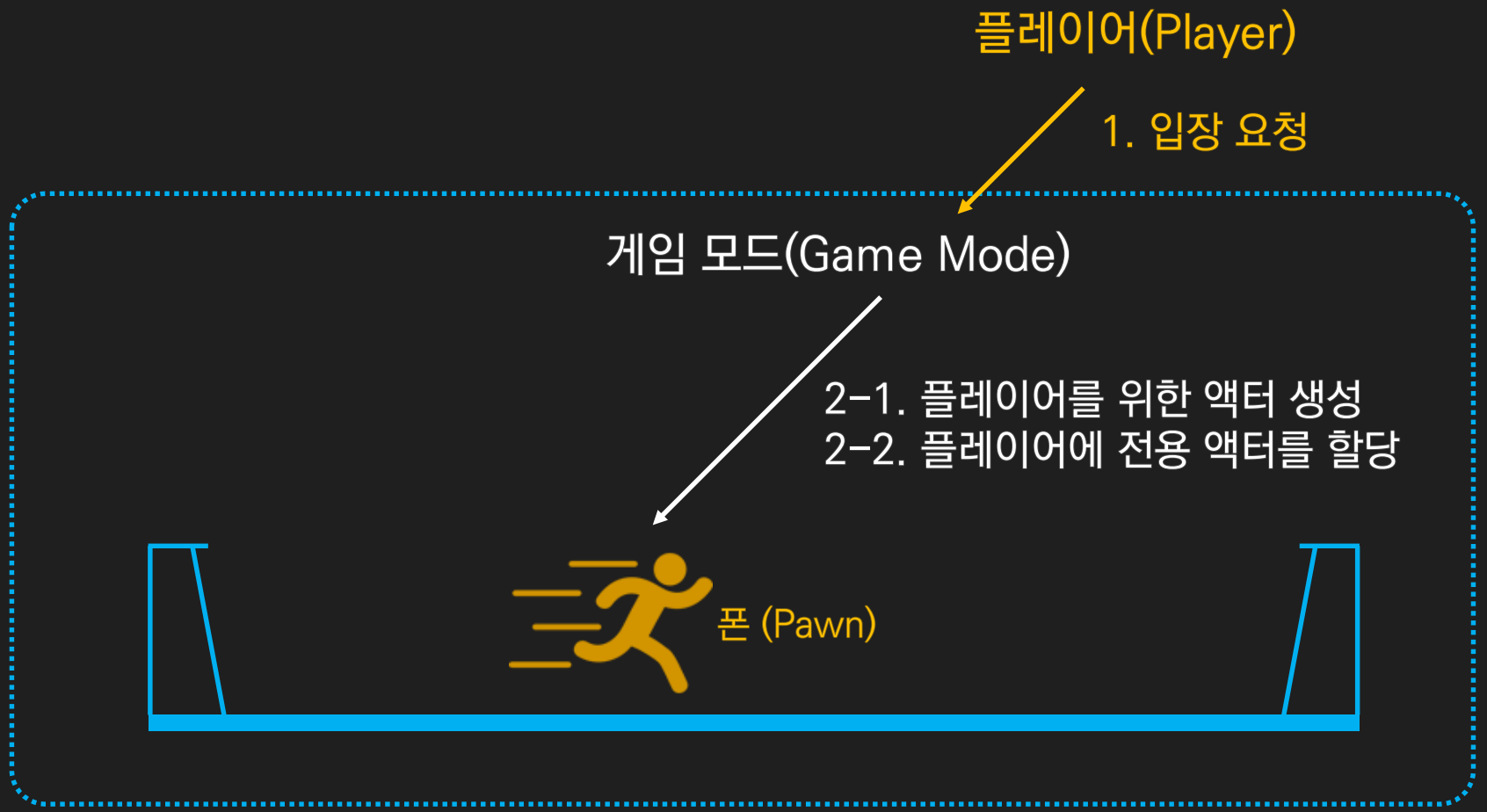
게임에 처음 접속한 플레이어는 0번을 부여받는다.

로컬/스탠드얼론/싱글플레이 게임은 항상 0번 플레이어가 입장한다.



멀티 플레이어 입장 데모

플레이어 입장 시 발생하는 일

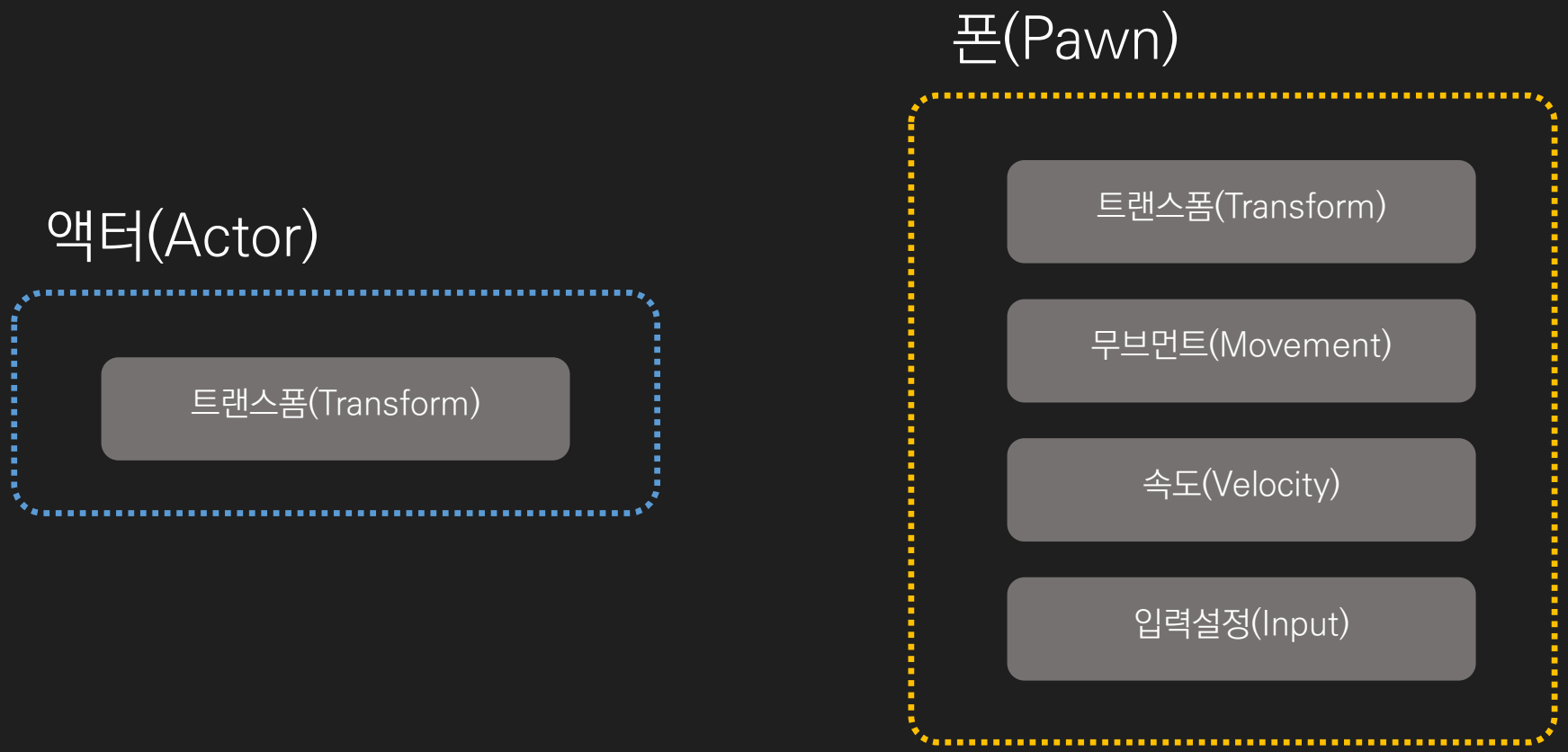


언리얼 엔진에서 폰(Pawn)이란?

1. 플레이어가 조종할 수 있는 액터
2. 플레이어의 명령을 충실히 이행 (입력)
3. 가상 세계의 시간 / 물리적 제약을 받으며 행동
4. 가상 세계에서 벌어진 일을 플레이어에게 리포트 (출력)

폰은 플레이어를 게임 세계에 투영시킨 물체. 아바타

액터 모델과 폰 모델의 비교



폰의 생성 데모

플레이어 폰의 무브먼트 시스템

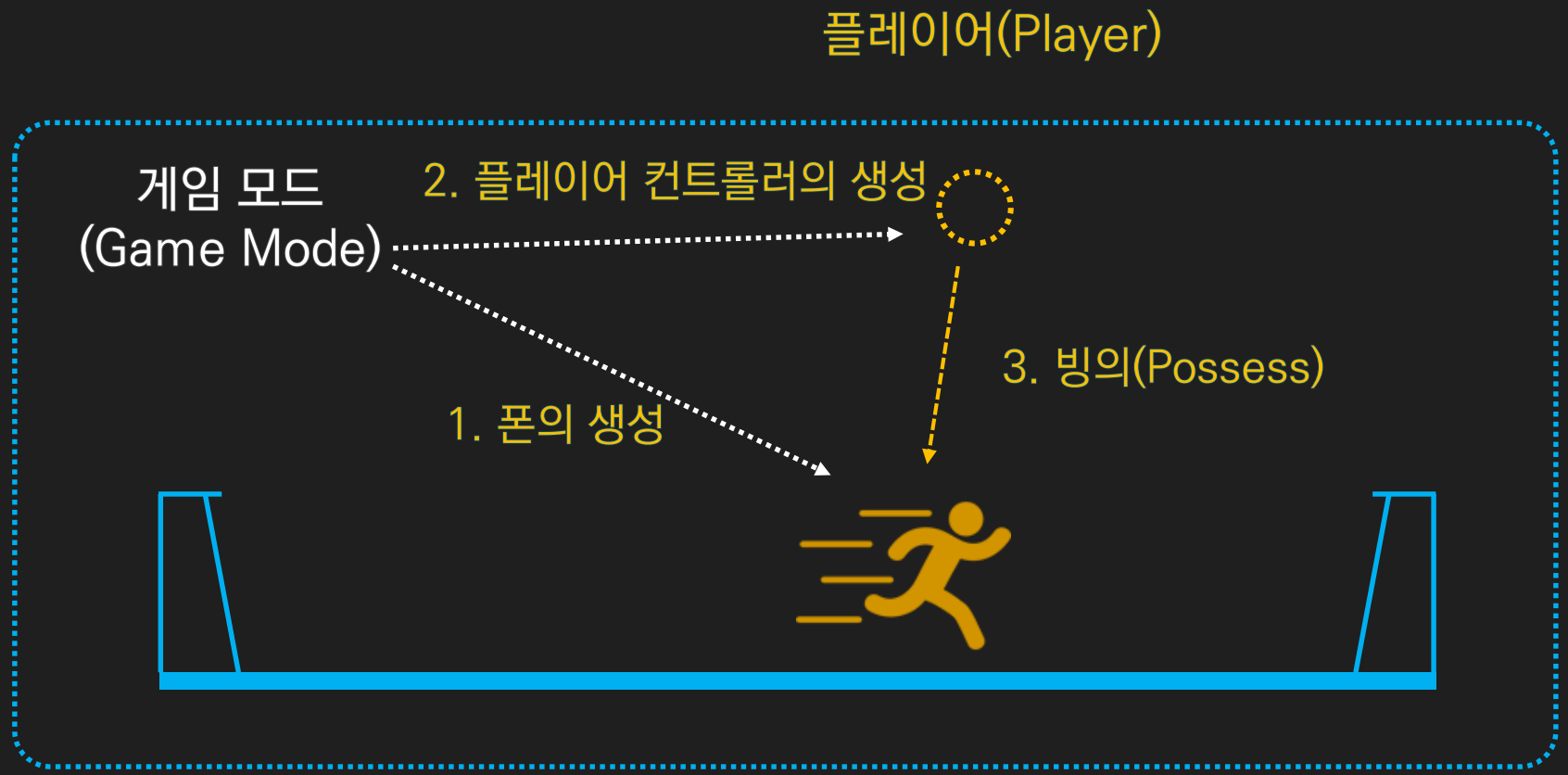
AddMovementInput = 악셀과 핸들

무브먼트 컴포넌트 = 자동차 엔진

루트 컴포넌트 = 자동차 프레임

폰의 무브먼트 시스템 데모

플레이어 컨트롤러와 빙의



언리얼 엔진에서 플레이어 컨트롤러(Player Controller)란?

1. 플레이어를 대변하는 가상 세계의 무형의 존재 (물리적 제약이 없음)
2. 플레이어의 의지를 받아 현재 빙의 중인 폰에게 전달.
3. 빙의 중인 폰이 느끼는 감각을 플레이어에게 전달. (대표적인 감각 : 시각)

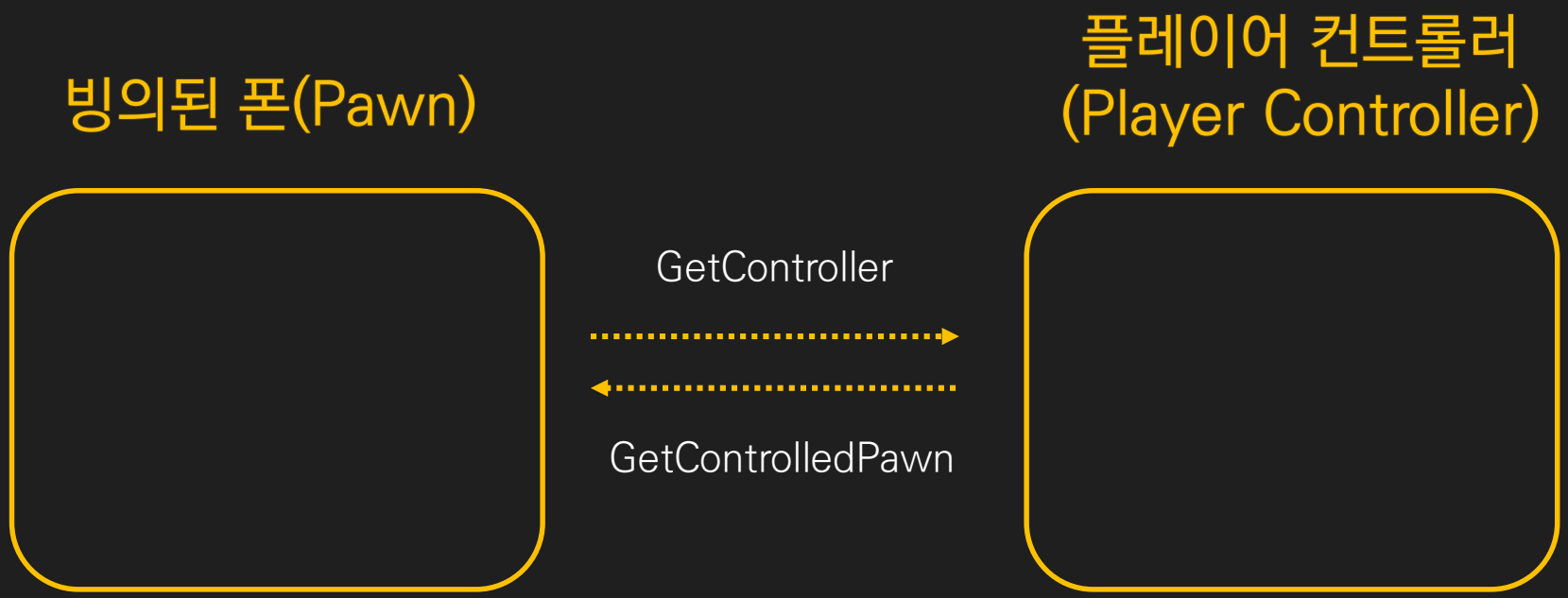
현실 세계 플레이어와의 실질적 커넥션

언리얼 엔진에서 빙의(Possess)란?

1. 월드에 존재하는 폰은 모두 플레이어가 조종할 수 있음.
2. 플레이어는 이 중 하나만 조종이 가능.
3. 내가 원할 때 전지 전능하게 다른 폰으로 변경해 조종이 가능한 시스템.

다양한 유형의 게임을 빙의 시스템을 통해 구현 가능

빙의된 폰과 플레이어 컨트롤러와의 관계



빙의 시스템 데모

언리얼 엔진에서 캐릭터(Character)란?

1. 근사화 된 충돌 영역
2. 캐릭터 애니메이션
3. 리얼한 움직임

캐릭터는 리얼한 무브먼트와 애니메이션 기능을 가진 폰

캐릭터 제작 데모

언리얼 엔진에서 컨트롤 회전(Control Rotation)이란?

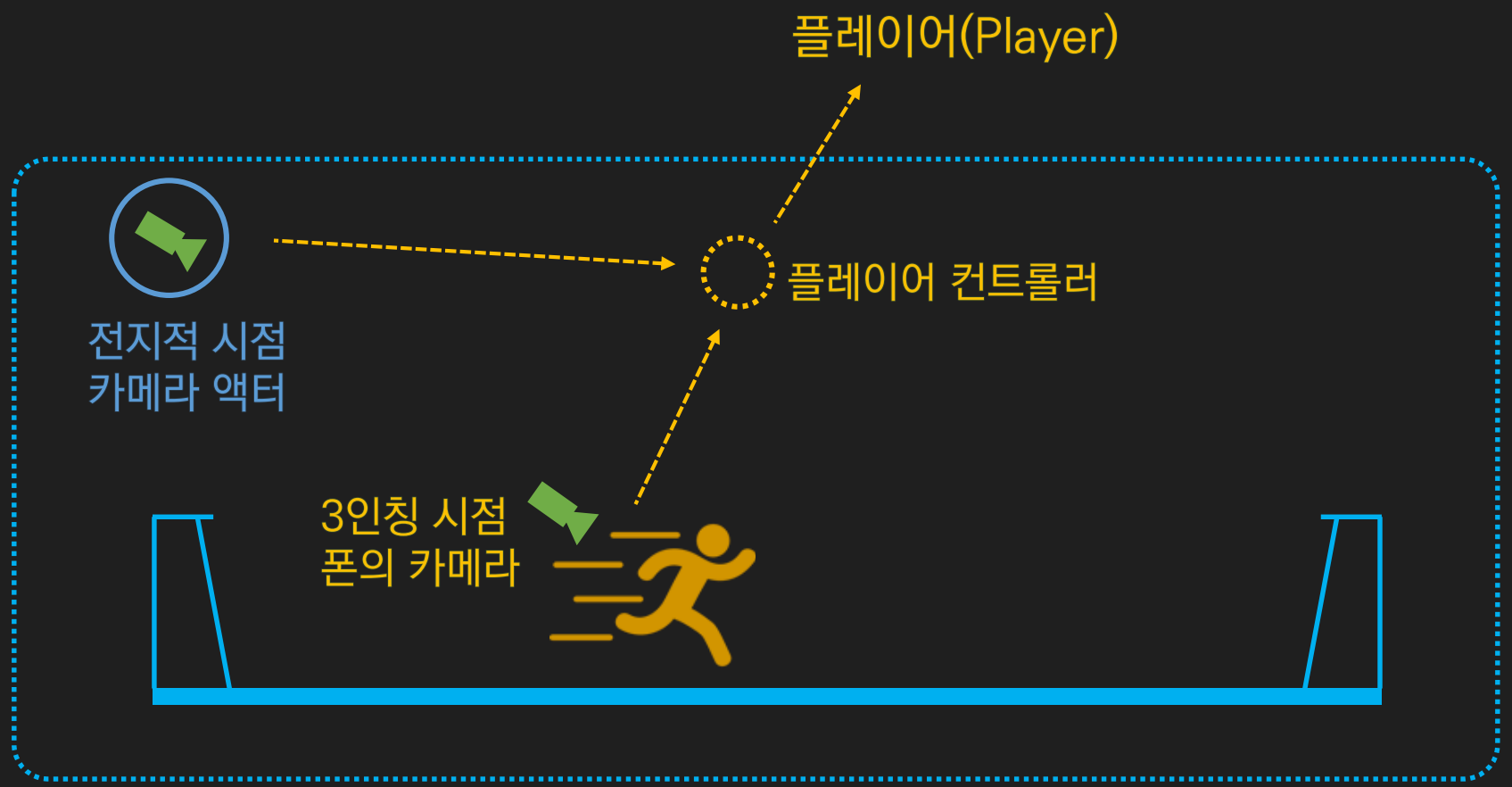
1. 플레이어의 의지를 보관하는데 사용
2. 폰은 물리적인 제약을 받고 있기 때문에 의지를 바로 반영할 수 없다.
3. 의지는 플레이어 컨트롤러에서, 폰은 물리적인 상황에 따라 수용하는 설계

플레이어의 입력 방향을 컨트롤 회전에 저장하고,
폰은 컨트롤 회전을 참고해 물리적 움직임에 반영

콘솔 커맨드 : displayall PlayerController Rotation

컨트롤 회전 데모

게임 컨텐츠 제작을 위한 카메라 시스템

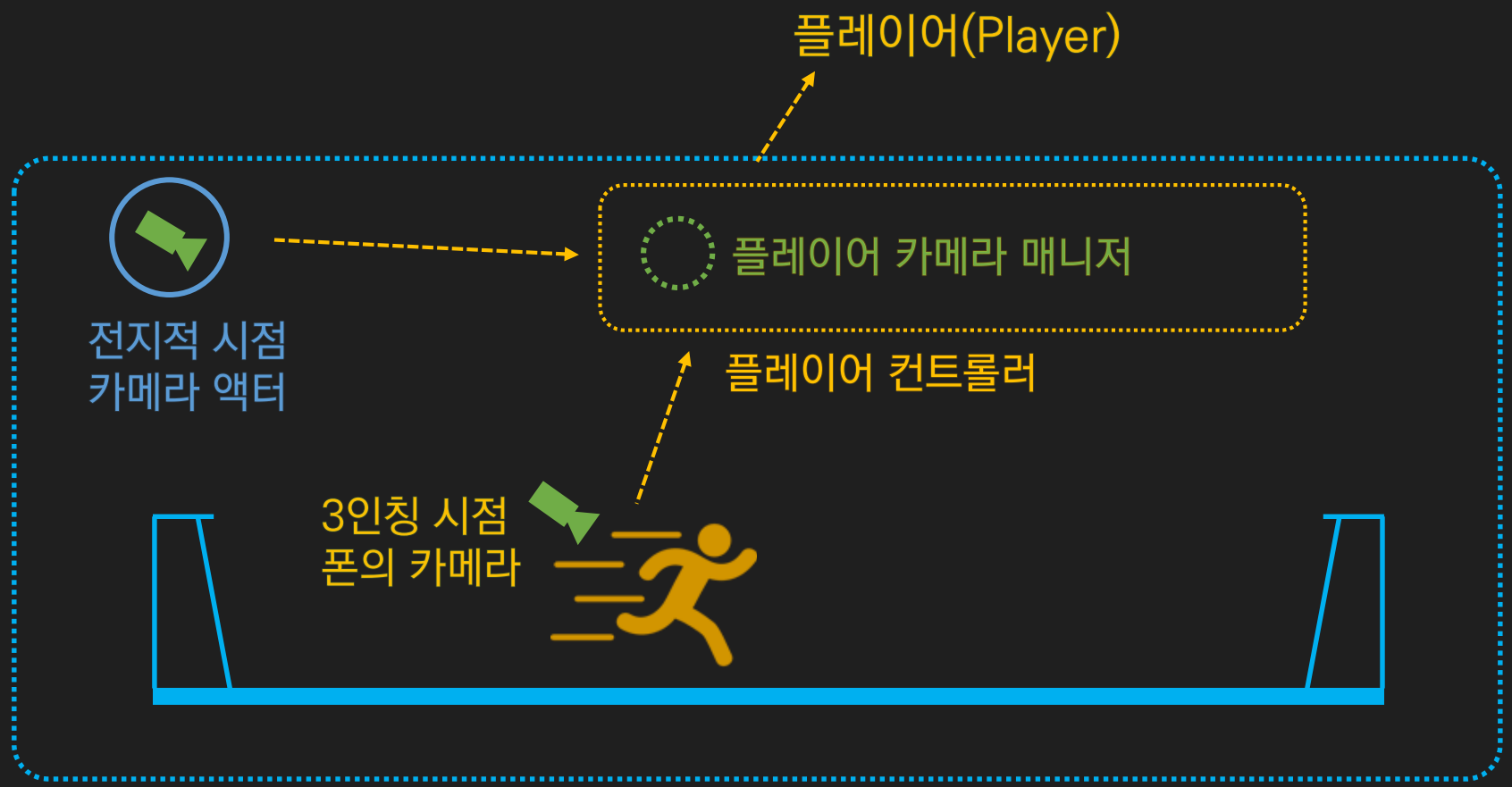


언리얼 엔진에서 뷰 타겟(View Target)이란?

1. 1인칭 / 3인칭 게임에서는 플레이어의 화면은 항상 폰을 따라 다님
2. 폰과 무관하게 전지적 시점을 가져야 할 필요가 종종 존재
3. 폰의 조종은 그대로 가져가되, 시점은 카메라 기능을 가진 액터로 대체
4. 화면을 뿌려주는 역할을 가진 액터를 뷰 타겟(View Target)이라고 함

빙의는 입력과 출력(뷰 타겟)을 지정한 폰에게 부여하는 기능

플레이어 카메라 매니저



언리얼 엔진에서 플레이어 카메라 매니저(Camera Manager)란?

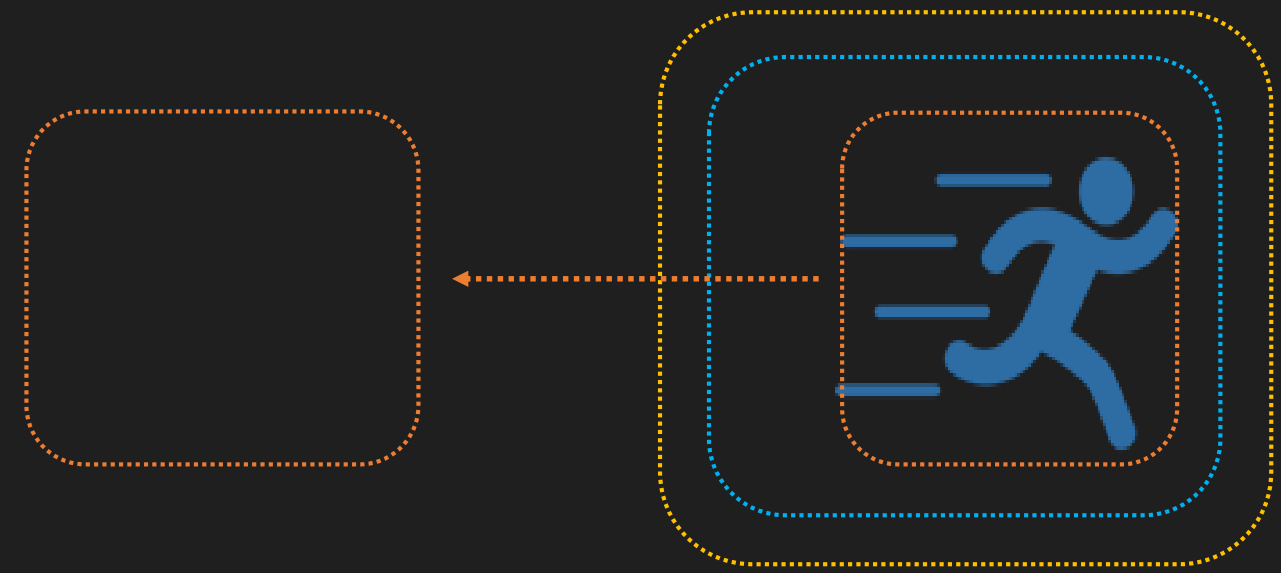
1. 플레이어 화면(Client Viewport)를 담당하는 카메라를 관리
2. 카메라가 지정된 범위 내에서만 움직이도록 제어
3. 현재 카메라의 위치 / 방향을 알 수 있음
4. 카메라의 애니메이션 / 페이드 / 셰이크등의 효과를 처리할 수 있음.
   (이미 다 만들어져 있음)

카메라 매니저를 통해 현재 카메라의 정보를 얻을 수 있음

플레이어 카메라 매니저 데모

애니메이션 시스템의 개요

액터에 독립적인
애니메이션 시스템의 생성
(애님 인스턴스)



물리적 충돌 기능
(캡슐 컴포넌트)

시각적인 기능
(스켈레탈메시 컴포넌트)

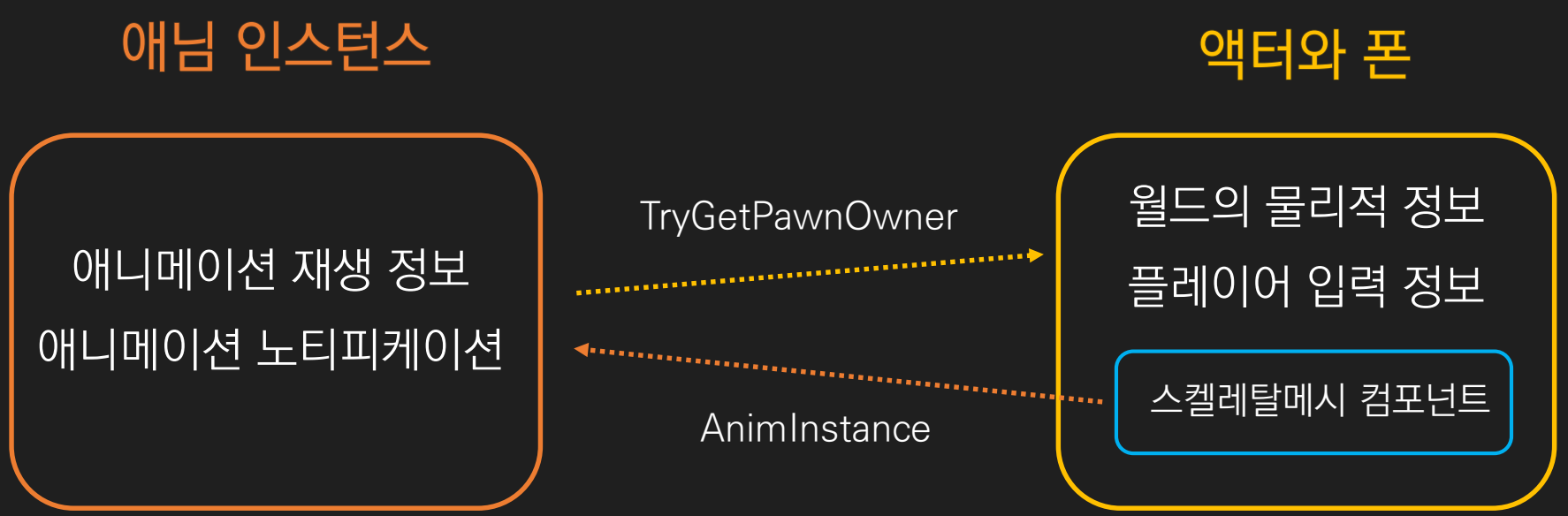
애니메이션 기능
(애니메이션 블루프린트)

언리얼 엔진에서 애님 인스턴스(Anim Instance)란?

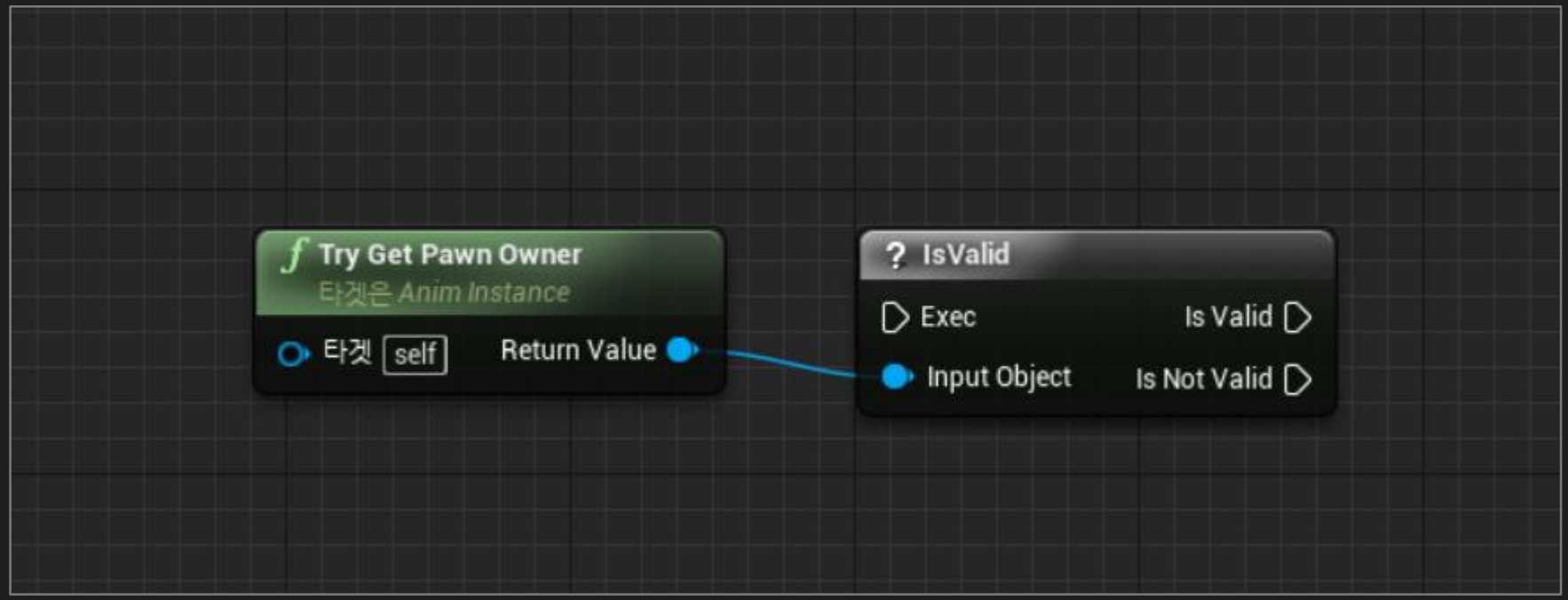
1. 애니메이션 블루프린트 설계도에서 생성된 애니메이션을 관리하는 시스템
2. 시스템을 소유하는 액터(Owner)가 존재하지만 독립적으로 동작한다.
3. 소유자인 액터로부터 물리적/공간적 정보를 얻을 수 있음.
4. 소유자에게 애니메이션 재생 정보와 신호(Notification)을 보낼 수 있음.

애니메이션 시스템과 폰 시스템은 독립적으로 동작하며
상호 필요한 정보만 교환

애니메이션 시스템의 설계



Try(?)GetPawnOwner 노드



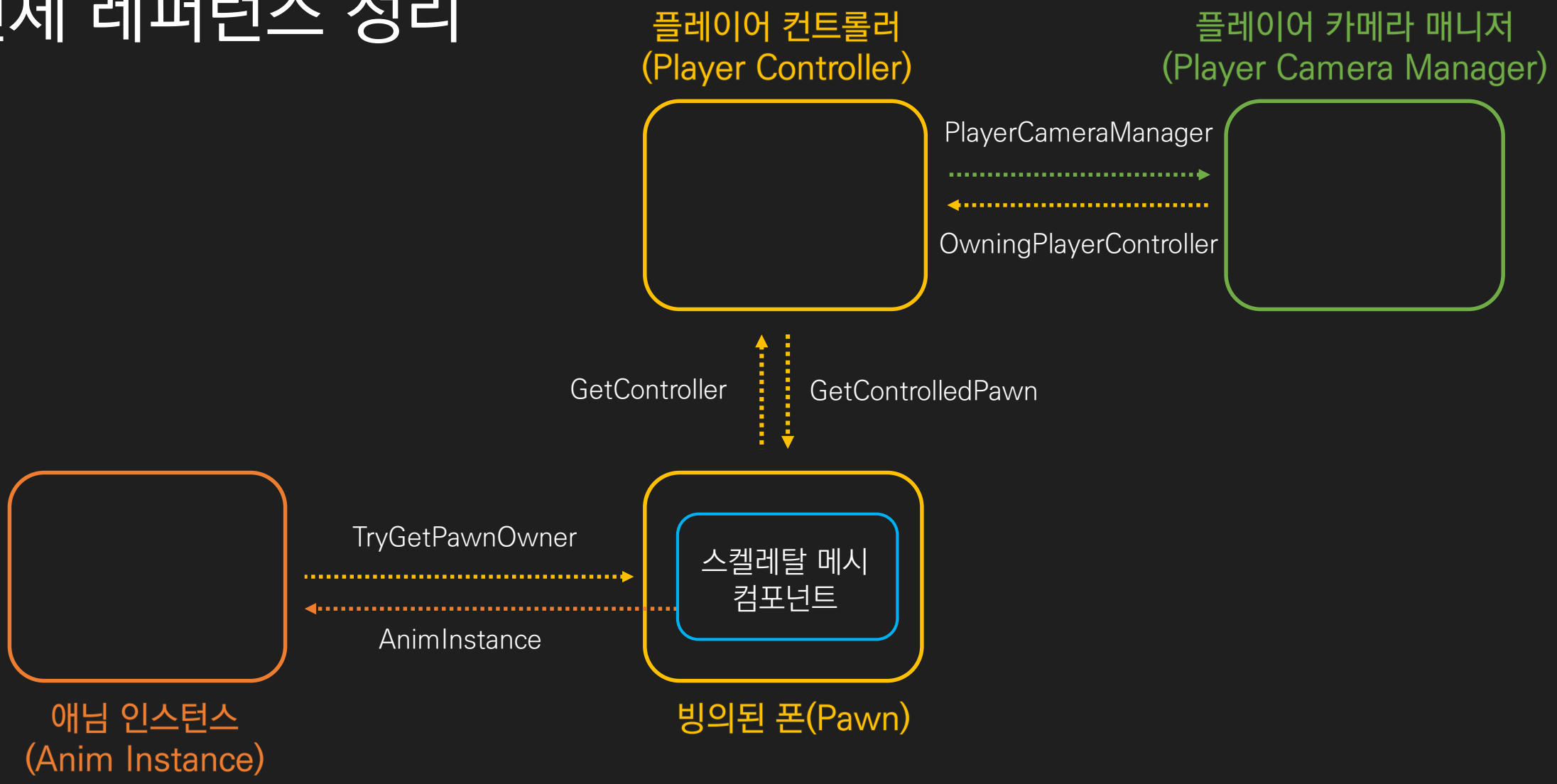
언리얼 엔진의 실행 순서

1. 플레이어 입력을 가장 먼저 처리 (VR은 예외) – 플레이어 의지의 전달
2. 액터를 구성하는 컴포넌트의 Tick 처리 – 구성 요소의 반응
3. 액터의 Tick 처리 – 최종 액터가 취할 행동 결정
4. 애니메이션 블루프린트 업데이트 – 액터의 행동에 따른 알맞은 애니메이션 재생

애니메이션 로직은 액터 로직보다 늦게 실행된다.
액터 로직에서 소유자인 자신을 없앨 수 있기 때문에 항상 체크.

애니메이션 노티파이 스테이트 데모

전체 레퍼런스 정리


플레이어 컨트롤러
(Player Controller)
플레이어 카메라 매니저
(Player Camera Manager)
PlayerCameraManager
OwningPlayerController
GetController
GetControlledPawn
TryGetPawnOwner
AnimInstance
스켈레탈 메시
컴포넌트
애님 인스턴스
(Anim Instance)
빙의된 폰(Pawn)

감사합니다!