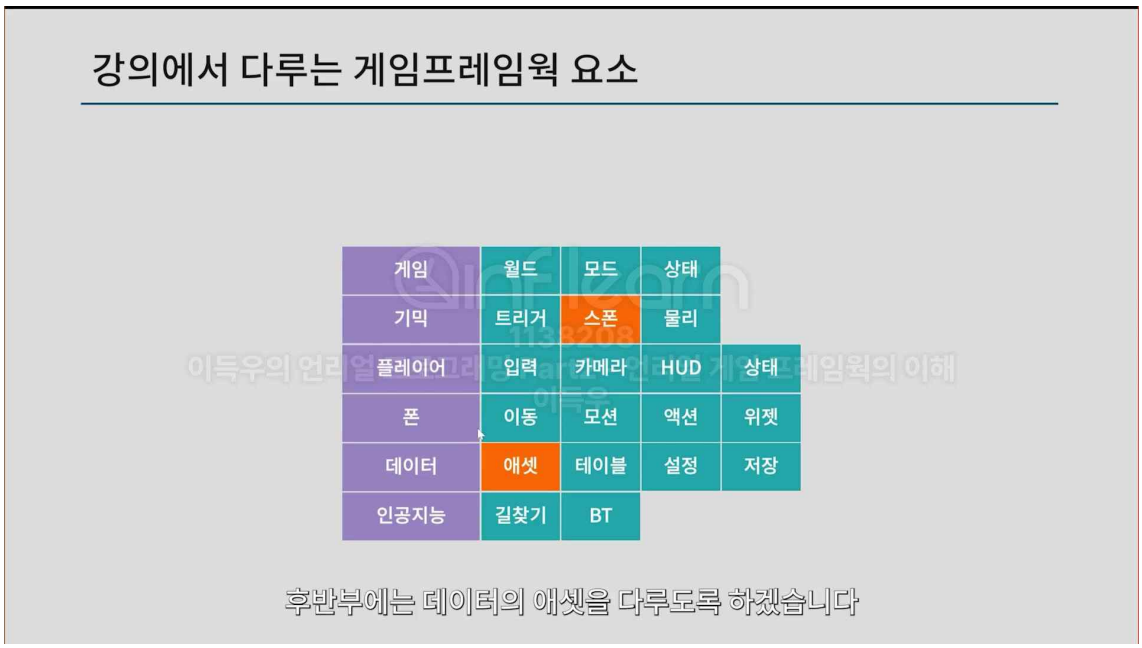
언리얼 프로그래밍 Part2-9

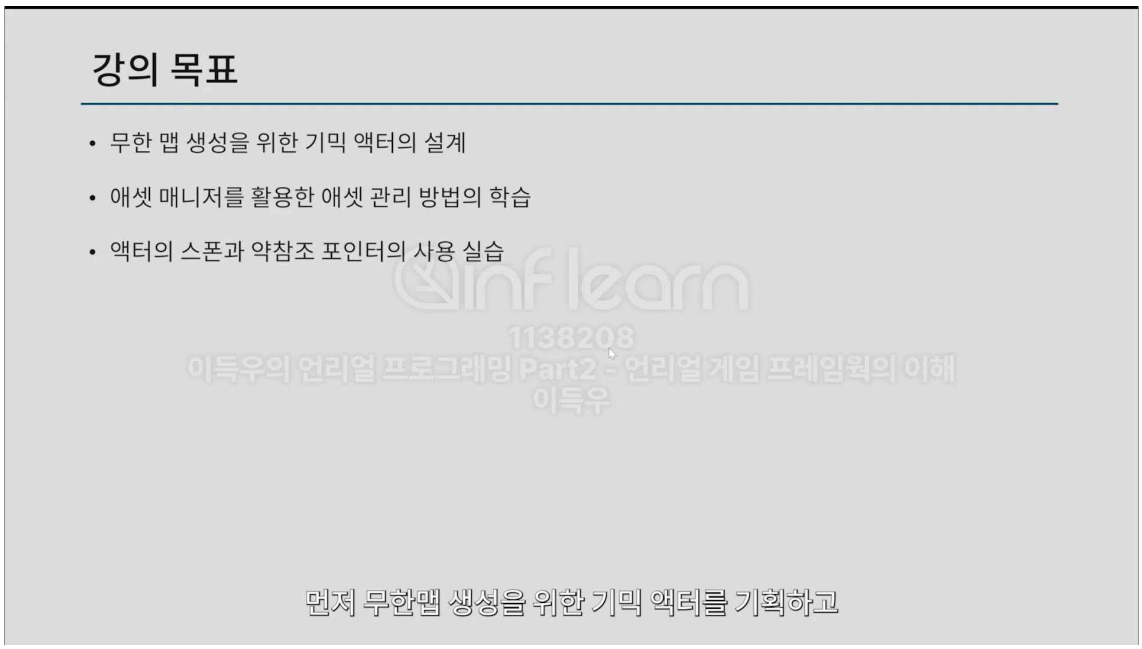
제목: 무한 맵의 제작

**강의 내용 :

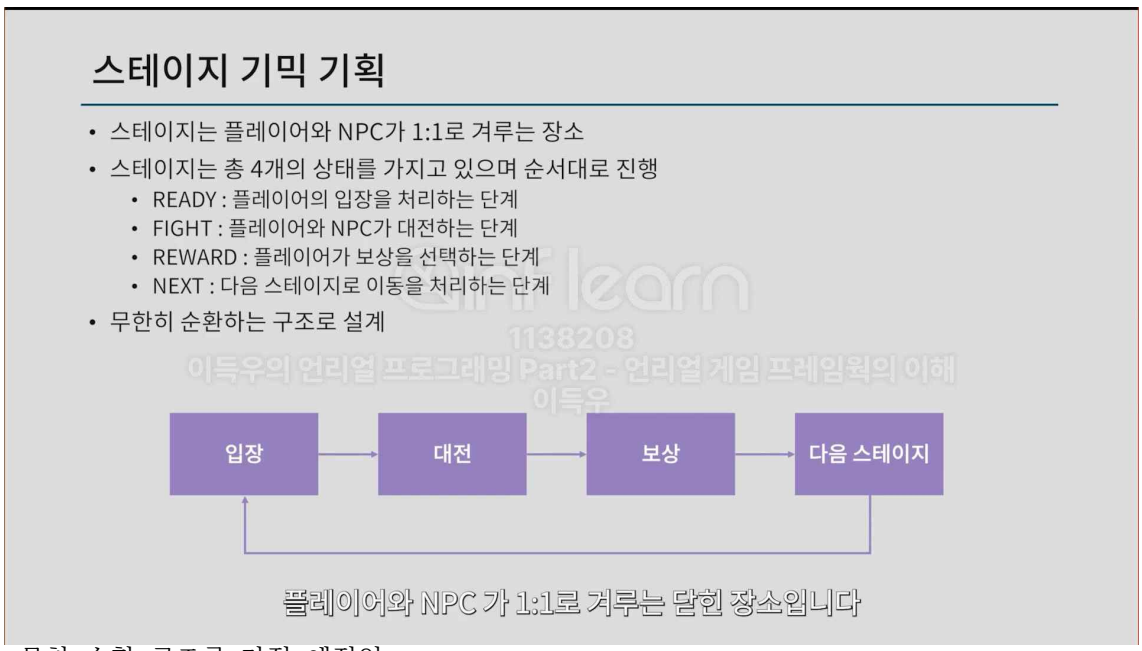
**강의에서 다루는 게임프레임웍 요소



**강의 목표



***스테이지 기믹의 설계



-무한 순환 구조를 가질 예정임.
-첫번째:Ready상태로 플레이어의 입장을 대기하는 상태
-두번째:플레이어와 대전을 NPC를 소환시켜서 플레이어와 NPC가 대전이 일어나도록 설정하는 단계
-세번째:플레이어가 NPC를 이겼을 때 보상을 주도록 아이템 상자를 스폰시키는 상태
-마지막:플레이어가 다음 스테이지로 이동할 수 있도록 처리하는 단계

스테이지 준비 단계

- 스테이지 중앙에 위치한 트리거 볼륨을 준비
- 플레이어가 트리거 볼륨에 진입하면 대전 단계로 이동

-준비 단계의 경우 스테이지 중앙에 트리거 불륨을 배치 하고 플레이어가 트리거불륨에 진입하여 오버랩 이벤트가 발생하면 다음 상태인 '대전' 상태로 이동되도록 설정할 예정임.

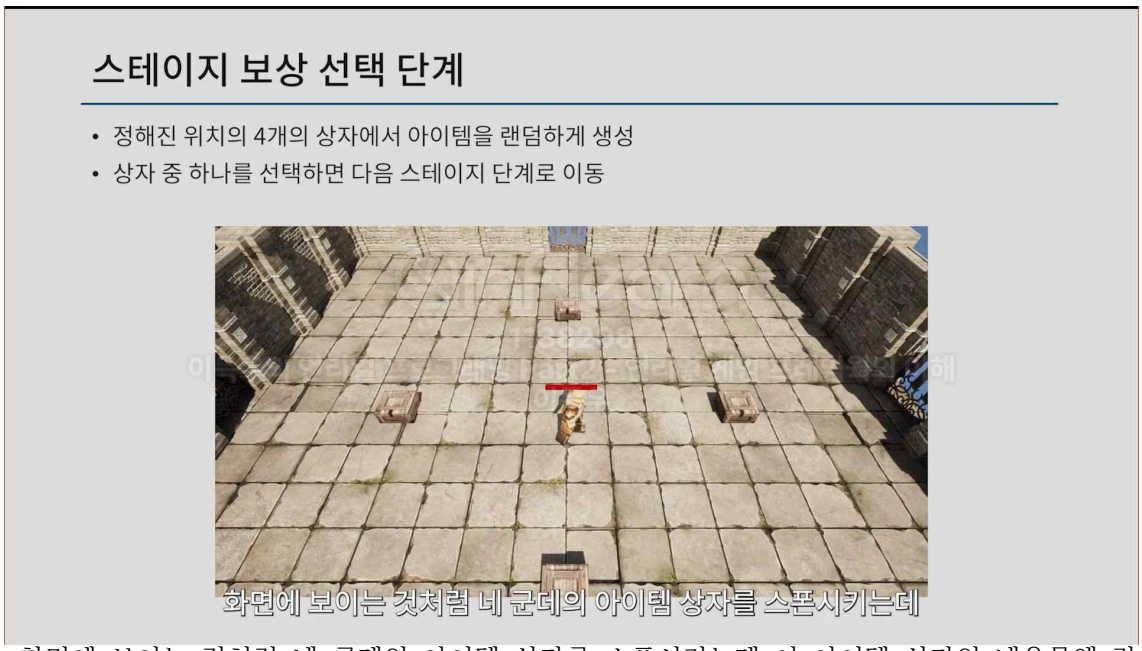
**스테이지 대전 단계

스테이지 대전 단계

- 플레이어가 못 나가게 스테이지의 모든 문을 닫고 대전할 NPC를 스폰
- NPC가 없어지면 보상 단계로 이동

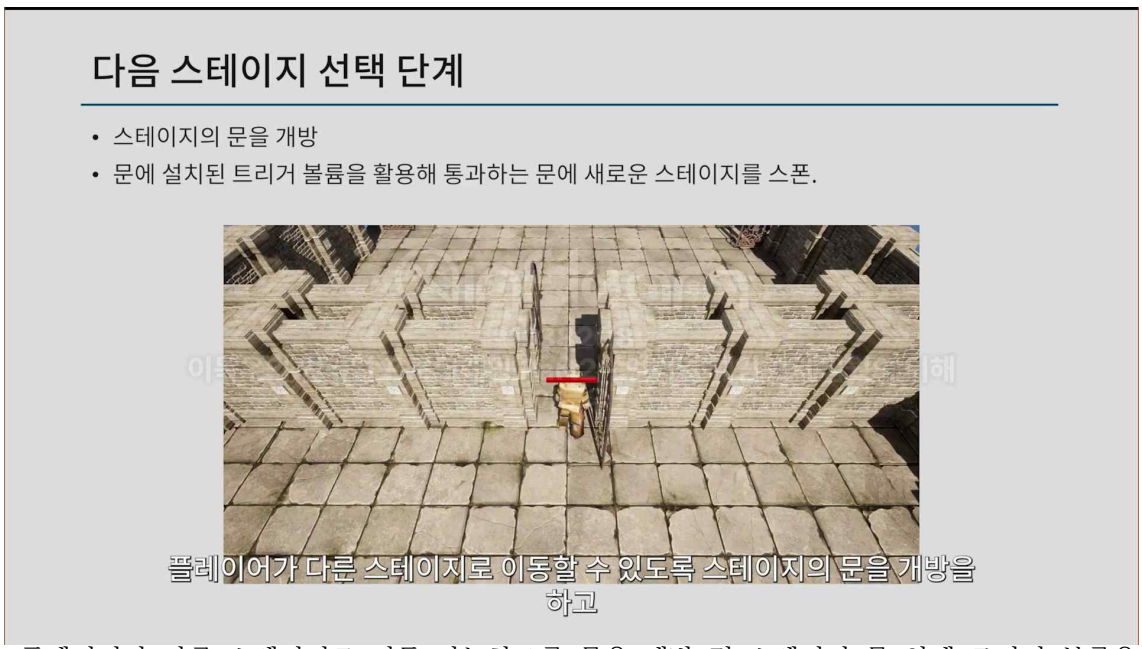
-대전 단계에 진입하면 플레이어가 나가지 못하도록 스테이지의 모든 문을 닫고, 대전할 NPC를 스폰을 시킨다. 이렇게 스폰된 NPC를 플레이어가 이기면 NPC가 없어지는데 NPC가 없어지면 이벤트를 감지하여 다음 단계인 보상 단계로 이동한다.

**스테이지 보상 선택 단계



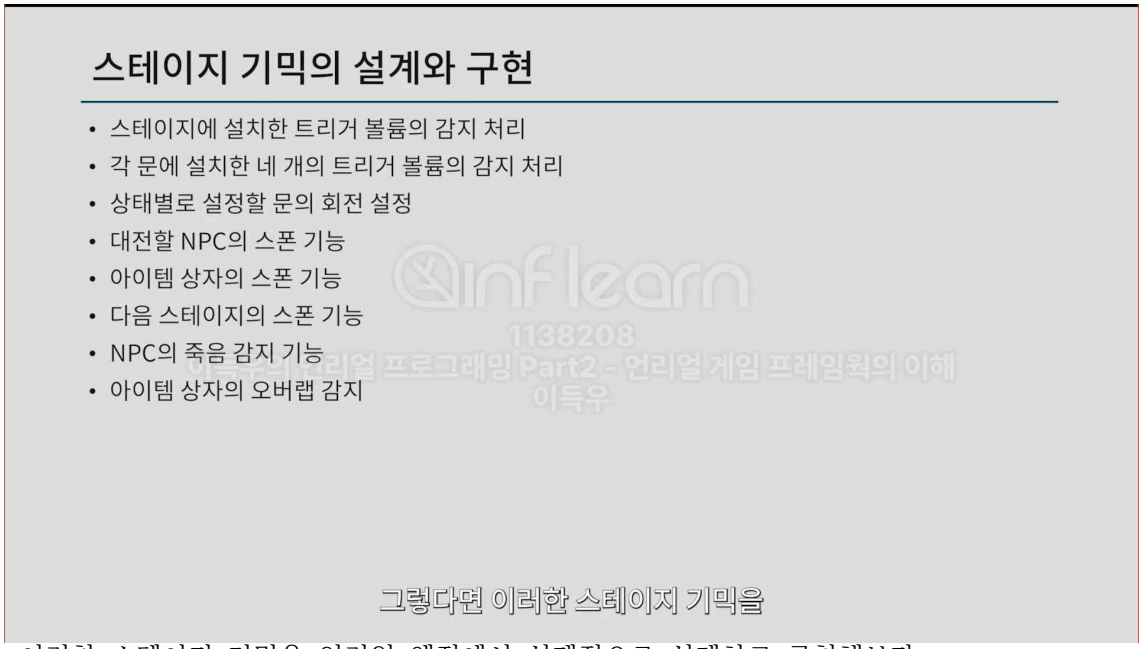
-화면에 보이는 것처럼 네 군데의 아이템 상자를 스폰시키는데 이 아이템 상자의 내용물엔 각각 다른 아이템들이 랜덤으로 설정이 된다. 이중에 하나를 설정하면 다음 스테이지 단계로 이동하도록 설정할 것이다.

**다음 스테이지 선택 단계



-플레이어가 다른 스테이지로 이동 가능하도록 문을 개방 및 스테이지 문 앞에 트리거 볼륨을 장착하여 플레이어가 이동했을 때 오버랩 이벤트가 발동 되면 그 오버랩 이벤트를 사용하여 다음 스테이지 영역에 새로운 스테이지 액터를 스폰시키도록 설정 한다.
-이렇게 되면 플레이어가 이동하면서 새로운 스테이지를 계속해서 생성이 가능하기 때문에 무한맵의 제작이 가능하다.

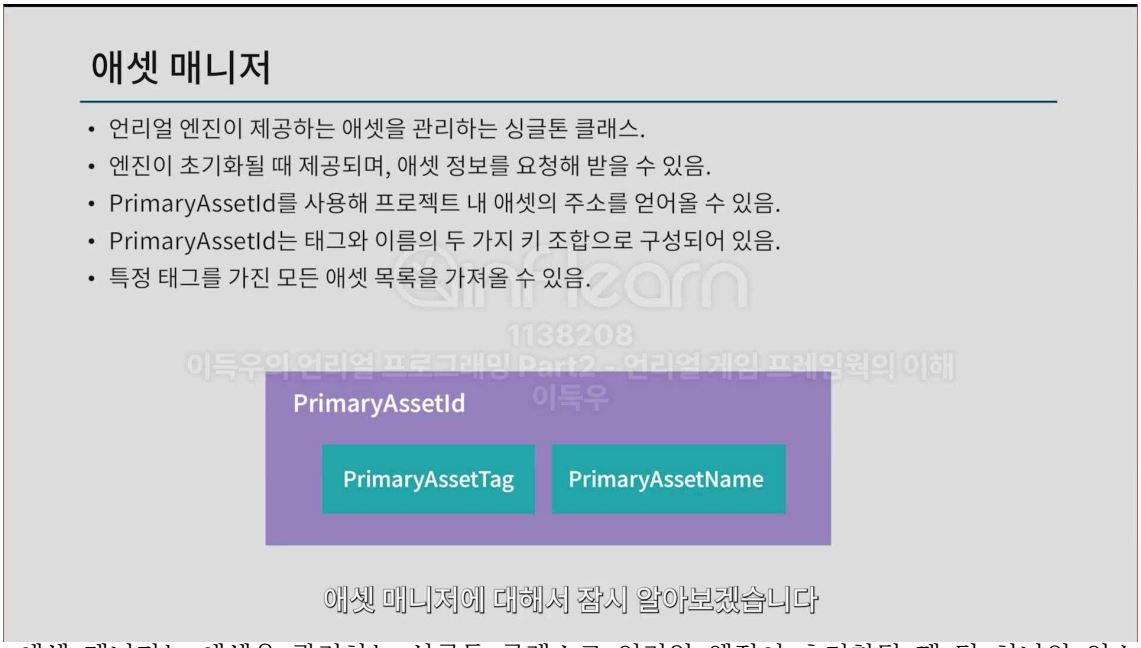
**스테이지 기믹의 설계와 구현



-이러한 스테이지 기믹을 언리얼 엔진에서 실제적으로 설계하고 구현해보자.
-먼저 스테이지 가운데 설치한 트리거 볼륨에 캐릭터가 들어와서 오버랩 이벤트가 발생했을 때 이것을 처리하기 위한 로직이 필요함.
-각 문에 설치한 네 개의 트리거 볼륨에도 플레이어가 진입해 오버랩 이벤트를 발생 시킬 때 이것을 처리하는 로직이 필요하다.
-상태별로 문이 '열렸다', '닫혔다' 하는 문의 회전 설정도 필요하다.
-플레이어가 스테이지에 스테이지에 들어왔을 때 대전할 npc를 지정하고 이것을 스폰시키는 기능이 필요하다.
-NPC를 이겼을 때 보상으로 주어지는 아이템 상자의 타입을 지정하여 이것을 스폰하는 기능이 필요함
-각문에 설치된 네 개의 트리거 볼륨중 하나에 진입하여 다음 스테이지를 스폰시키는 기능이 필요함.
-추가로 NPC가 죽었을 때 이를 감지하여 아이템 상자를 스폰하는 단계로 넘기는 기능이 필요하다.
-스폰된 아이템 상자중에 하나의 오버랩을 감지하여 하나를 선택하면 나머지 상자들을 모두 다 없애는 기능이 필요하다.

***아이템 상자의 랜덤 보상

**애셋 매니저



-애셋 매니저는 애셋을 관리하는 싱글톤 클래스고 언리얼 엔진이 초기화될 때 단 하나의 인스턴스만을 보장받도록 설계 됨.
-이 애셋 매니저를 사용하여 게임이 실행되는동안 애셋정보를 요청하여 받을 수 있음(이때 'PrimaryAssetID'라고 하는 특별한 아이디가 사용된다.)

-'PrimaryAssetID'라는 특별한 아이디를 사용해서 프로젝트 내의 애셋의 주소를 얻어올 수가 있는데 지금 까지 우리는 레퍼런스 복사를 사용하여 각각에 대한 애셋의 주소를 수동으로 복사하여 붙여넣어 로딩 했는데 이제는 PrimaryAssetID를 통하여 편리하게 애셋을 찾아서 주소를 가져올 수가 있다.

-PrimaryAssetID'는 우리가 지정한 태그와 애셋 이름의 두 가지 키의 조합으로 구성이 되어 있다.

-애셋 매니저를 통해 특정 태그를 가진 모든 애셋 목록을 가져올 수도 있다.

**랜덤 보상 설정

-이것을 활용하여 랜덤 보상을 설정할 것이다.

-우리가 지난번에 작업한 아이템 데이터에 'ABItemData'라고 하는 특별한 애셋 태그를 설정할 예정이다.

-프로젝트 설정에서는 우리가 지난시간에 만들었던 포션, 웨폰, 스크롤 이러한 애셋들이 담긴 폴더를 /Game/ArenaBattle/Item 이렇게 지정하고 해당 애셋 태그를 설정하여 해당 폴더에 담긴 모든 애셋의 경로정보를 애셋 매니저에게 요청하여 받아올 수 있다.

-받아온 것들 중에 하나를 랜덤으로 선택하고 이를 상자에 할당하여 플레이어가 먹었을 때 보상으로 지급하도록 로직을 구성해보겠다.

**실습
-우리가 구현된 맵을 테스트 해야 되는데 만약 구현된 스테이지를 선택하고 State값을 변경해도 State값이 변경되는 것이지 해당 스테이트 값의 변경에 따라서 지정된 다양한 SetState함수들이 실행되는건 아니라서 의미가 없다.

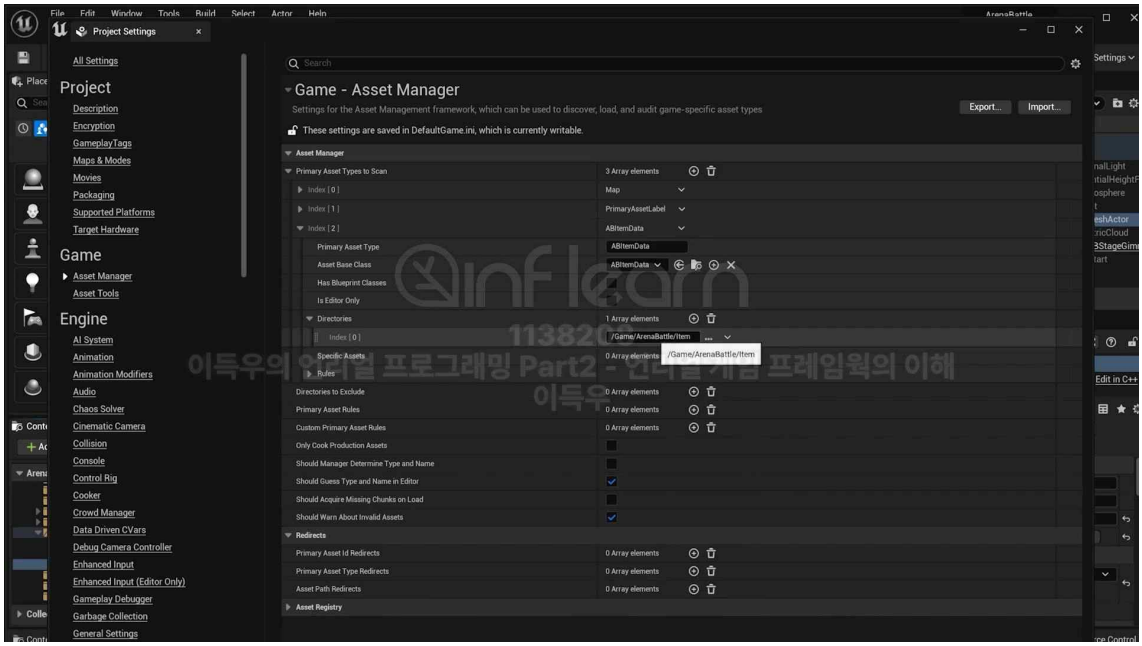
-에디터에서 속성들을 변경할 때 연동된 것들이 같이 시뮬레이션 되게 설정하면 레벨 디자인 할 때 굉장히 편리하게 사용이 가능하다.

-언리얼 엔진은 이것을 위해서 OnConstruction이란 함수를 제공함. 이를 사용하여 각각 상태에 따른 변화를 확인해보자.

*TWeakObjectPtr
-타입을 보면 TObjectPtr이 아닌 TWeakObjectPtr을 통해 선언했는데 이것을 약참조라고 한다.
-RewardBoxes라는 변수는 박스는 스폰된 상자를 관리하기 위함
-스폰된 상자는 우리가 작업하고 있는 스테이지 기믹 액터와는 무관하게 자기 스스로 동작함.
-그래서 외부의 영향을 받았던가 내부의 로직에 의해 스스로 소멸 가능.
-우리가 이것들을 관리할때 TObjectPtr이라고하는건 강참조임
-언리얼 엔진은 강참조로 걸린 상자들을 보면 '아 이 상자들은 아직도 사용하구 있구나'라고 판단하여 메모리에서 소멸 안시킬 수도 있음
-따라서 나는 얘와 무관하지만 얘를 어쨋든 관리해야 된다 라는 측면에서 TWeakObjectPtr라는 약참조로 선언하는게 좋음.
-지금까지 우리는 액터의 선언에서 멤버변수로 가지고 있는 언리얼 오브젝트들을 모두다 강참조로 선언했는데
-이것은 액터가 소멸될때 같이 소멸되는것이 맞기 때문에 강참조를 걸어주는게 좋음
-하지만 지금처럼 액터와 무관하게 동작해야 하는 다른 액터들은 가급적 약참조를 통해 관리하자.

-아이템 폴더의 애셋들의 특징은 ABItemData애셋을 부모로 가지는 애셋들인데 이를 모두 불러들일 수 있는 효과적인 방법은 프로젝트 세팅에서 Asset Manager라고 하는 싱글톤 클래스가 있다. 이것은 엔진이 초기화될 때 반드시 활성화 되는 단 하나만 존재하는 인스턴스임

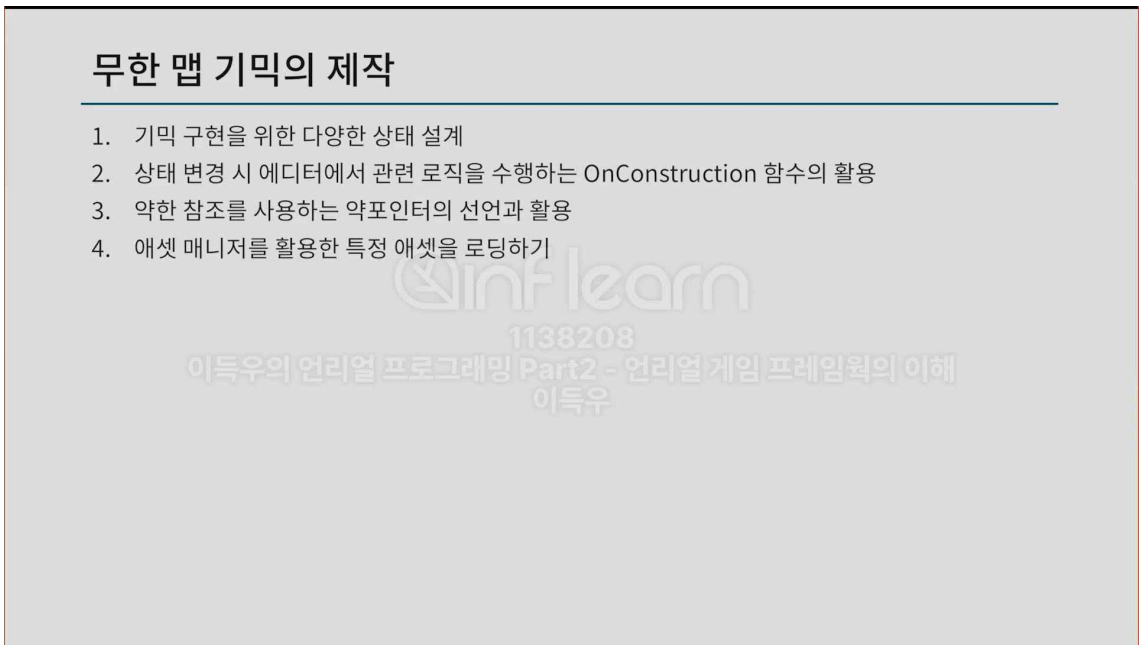
-여기에 Primary Asset Type이라는 것으로 ABItemData를 관리하도록 지정해주고 이 애셋 매니저로부터 모든 애셋의 목록을 불러오도록 설정하는 것이다.



-우리가 제작한 클래스의 선언에서 PrimaryAssetId라고 하는 값을 앞서서 지금 살펴본 ABItemData라는 것으로 태그를 달아주면 된다.

-실습에서 코딩부분을 다 작성하고 나서 엔진이 로딩될 때 애셋 매니저는 우리가 지정한 ABItem이라고하는 태그값을 보고 거기에 해당되는 애셋들은 특별하게 관리하기 시작함.

**정리


무한 맵 기믹의 제작
1. 기믹 구현을 위한 다양한 상태 설계
2. 상태 변경 시 에디터에서 관련 로직을 수행하는 OnConstruction 함수의 활용
3. 약한 참조를 사용하는 약포인터의 선언과 활용
4. 애셋 매니저를 활용한 특정 애셋을 로딩하기
inflearn
1138208
이득우의 언리얼 프로그래밍 Part2 - 언리얼 게임 프레임웍의 이해
이득우