

# 언리얼 프로그래밍 Part1-14

## 제목:언리얼 오브젝트 관리 II-패키지

**\*\*강의 내용 :** 언리얼 엔진의 패키지와 애셋 개념을 이해하고 언리얼 에디터에 애셋 데이터를 저장하고 불러들이기

**\*\*강의 목표 :**

### 강의 목표

- 언리얼 엔진의 애셋과 이를 포함한 패키지의 개념 이해
- 언리얼 에디터에서 볼 수 있도록 애셋을 저장하고 불러들이는 방법의 이해
- 오브젝트 패스를 사용해 다양한 방식으로 애셋을 로딩하는 방법의 이해



1138208

이득우의 언리얼 프로그래밍 Part1 - 언리얼 C++의 이해  
이득우

이번 강의의 목표는 다음과 같습니다

### \*\*\*언리얼 오브젝트 패키지

### \*\*언리얼 오브젝트 패키지

## 언리얼 오브젝트 패키지

- 단일 언리얼 오브젝트가 가진 정보는 저장할 수 있지만, 오브젝트들이 조합되어 있다면?
  - 저장된 언리얼 오브젝트 데이터를 효과적으로 찾고 관리하는 방법은?
  - 복잡한 계층 구조를 가진 언리얼 오브젝트를 효과적으로 저장과 불러들이는 방법을 통일해야 함.
- 언리얼 엔진은 이를 위해 패키지(UPackage)단위로 언리얼 오브젝트를 관리함.
- 패키지의 중의적 개념
  - 언리얼 엔진은 다양한 곳에서 단어 패키지를 사용하고 있음.
  - 언리얼 오브젝트를 감싼 포장 오브젝트를 의미함. **(이번 강의의 주제)**
  - 또한 개발된 최종 콘텐츠를 정리해 프로그램으로 만드는 작업을 의미함. (예) 게임 패키징)
  - DLC와 같이 향후 확장 콘텐츠에 사용되는 별도의 데이터 묶음을 의미하기도 함 (예) pkg 파일)

구분을 위해 언리얼 오브젝트 패키지로 부르는 것도 고려

어떻게 해야 될까요?

-언리얼 엔진엔 다양한 곳에 '패키지' 라는 단어를 사용함

-이번 강의에서는 언리얼 오브젝트를 감싼 포장 오브젝트를 의미함

### \*\*패키지(Package)와 애셋(Asset)

## 패키지(Package)와 애셋(Asset)

- 언리얼 오브젝트 패키지는 다수의 언리얼 오브젝트를 포장하는데 사용하는 언리얼 오브젝트.
  - 모든 언리얼 오브젝트는 패키지에 소속되어 있음. (예) Transient Package)
- 언리얼 오브젝트 패키지의 서브 오브젝트를 애셋(Asset)이라고 하며 에디터에는 이들이 노출됨
- 구조상 패키지는 다수의 언리얼 오브젝트를 소유할 수 있으나, 일반적으로는 하나의 애셋만 가짐.
- 애셋은 다시 다수의 서브오브젝트를 가질 수 있으며, 모두 언리얼 오브젝트 패키지에 포함됨.
  - 하지만 에디터에는 노출되지 않음.



-우리가 그동안 사용한 언리얼 오브젝트는 사실 패키지(트랜지언트[임시 패키지])에 속해있다.

-다양한 언리얼 오브젝트 중에서 서브 오브젝트를 애셋이라고 한다.

-에디터에는 패키지 정보가 노출되는 것이 아니라 애셋 정보가 노출 된다.

-구조상 패키지는 여러개의 애셋들을 소유할 수 있다.(일반적으로는 1:1로 설계함)

**\*\*실습**

-1.13 강의 프로젝트를 이어서 진행함

-우리가 패키지를 사용하기 위해서는 패키지와 패키지가 담고 있는 대표 에셋을 설정해 줘야 한다.

-우리가 언리얼 엔진 프로젝트를 기동하게 되면 각각에 대해서 고유한 경로를 가지게 된다.

-그 고유한 경로중 하나가 /Game 이라는 경로인데 게임에서 사용되는 에셋들을 모아두는 대표 폴더를 의미한다.

-앞서 살펴본 Saved폴더는 Temp라는 폴더에 매핑이 되어 있다.

-Student라고 하는 패키지 이름을 설정했으면 Student 패키지가 메인으로 관리할 에셋이름을 하나 지정을 추가로 해주자.(TopStudent)

-코딩을 다하고 첫 번째 실행을 하게 되면 Content에 TopStudent가 생긴다.

-에디터에 보여지는 이름은 우리가 첫 번째 패키지가 소유하고 있는 에셋 이름인 TopStudent가 나온다. 그래서 게임 제작에서 에셋을 사용한다고 이야기를 보통 하는데 패키지 정보는 사실 알기가 어렵다.(잘보이지가 않음)

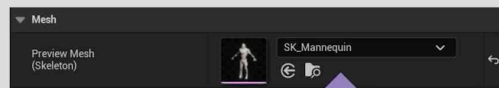
-하지만 패키지가 속한 패키지 바로 아래에 있는 서브 오브젝트인, 에셋에 대한 정보는 에디터에서 자세하게 볼 수 있다.

\*\*\*에셋 참조와 로딩

\*\*에셋 정보의 저장과 로딩 전략

## 에셋 정보의 저장과 로딩 전략

- 게임 제작 단계에서 에셋 간의 연결 작업을 위해 직접 패키지를 불러 할당하는 작업은 부하가 큼.
  - 에셋 로딩 대신 패키지와 오브젝트를 지정한 문자열을 대체해 사용. 이를 오브젝트 경로라고 함.
  - 프로젝트 내에 오브젝트 경로 값은 유일함을 보장함.
  - 그렇기에 오브젝트 간의 연결은 오브젝트 경로 값으로 기록될 수 있음.
  - 오브젝트 경로를 사용해 다양한 방법으로 에셋을 로딩할 수 있음.
- 에셋의 로딩 전략
  - 프로젝트에서 에셋이 반드시 필요한 경우 : 생성자 코드에서 미리 로딩
  - 런타임에서 필요한 때 바로 로딩하는 경우 : 런타임 로직에서 정적 로딩
  - 런타임에서 비동기적으로 로딩하는 경우 : 런타임 로직에서 관리자를 사용해 비동기 로딩



빈번하게 많이 이루어질 텐데요

에디터에서 에셋을 지정할 때마다 항상 로딩해야 하는가?

-에셋 로딩 3가지

1. 먼저 프로젝트에서 에셋이 반드시 필요한 경우 : 엔진이 초기화되는 생성자 코드에서 해당 에셋을 미리 로딩해 두는 것이 좋다.
2. 런타임에서 필요할 때만 : 런타임 로직에서 정적으로 로딩해준다.
3. 정적으로 로딩하면 다른 프로세스의 실행을 막아 게임이 멈추게 되는데 기존의 게임 로직을 진행하면서 비동기적으로 로딩이 필요한 경우 : 런타임 로직에서 관리자를 사용하여 비동기 방식으로 로딩하자.

## \*\*오브젝트 경로

### 오브젝트 경로(Object Path)

- 패키지 이름과 애셋 이름을 한 데 묶은 문자열
- 애셋 클래스 정보는 생략할 수 있음.
- 패키지 내 데이터를 모두 로드하지 않고 오브젝트 경로를 사용해 필요한 애셋만 로드할 수 있음.

{애셋클래스정보}'{패키지이름}.{애셋이름}'

또는

{패키지이름}.{애셋이름}

이것은 패키지 이름과

-애셋을 로딩하기 위해서 오브젝트 경로를 지정해줘야 하는데 이것은 패키지 이름과 애셋이름을 한 데 묶은 문자열이다.

-애셋 클래스 정보가 들어갈 때 패키지 이름과 애셋이름에는 작은 따옴표(')가 들어간다.

-애셋 클래스 정보는 사실 생략이 가능하다.(없어도 로딩 가능)

-패키지이름, 애셋이름 형태만 알 수 있다면 바로 특별한 패키지 안에 있는 애셋을 로딩할 수가 있다.

-오브젝트 경로를 우리가 안다면 패키지 내에 있는 모든 애셋을 로드하지 않고, 필요한 애셋만 개별적으로 로드가 가능하다.

**\*\*애셋 참조 공식 홈페이지**

-링크: [https://dev.epicgames.com/documentation/ko-kr/unreal-engine/referencing-assets-in-unreal-engine?application\\_version=5.1](https://dev.epicgames.com/documentation/ko-kr/unreal-engine/referencing-assets-in-unreal-engine?application_version=5.1)

-직접 프로퍼티 참조:

(1) 언리얼 오브젝트의 헤더에서 언리얼 오브젝트를 참조할 때 타입을 명시적으로 지정하는 것을 의미함 ex: `USoundCue* ConstructionStartStinger;`

-생성 시간 참조

(1)강참조를 진행할 때 해당 오브젝트가 가리키고 있는 애셋을 생성자 코드에서 생성해주는 것을 의미한다. (생성자 코드는 엔진이 초기화 될 때 실행이 된다. 따라서 게임이 실행되기 전에 해당 애셋이 로딩이 된다. 라고 이해하자.)

-간접프로퍼티 참조

(1)TSoftObjectPtr을 사용하는 것으로 LoadObject()나 StaticLoadObject()아니면 FStreamingManager를 사용하여 오브젝트를 필요할 때 로딩할 수 있다고 설명한다.

(2) FSoftObjectPath는 앞서 설명했던 오브젝트 패스를 의미한다. 이 값을 사용하여 우리가 스트리밍 매니저를 사용해 비동기적이나 동기적으로 애셋 로딩이 가능하다.

-오브젝트 검색/로드

(1)유프로퍼티를 기반으로 레퍼런스를 구한다고 했을 때 생성 또는 로드된 오브젝트만 사용하려고 할땐 FindObject()를 사용하면 된다.

(2) 이것은 우리가 사용하는 패키지들이 이미 메모리에 로딩되어있을 때 FindObject()를 사용하여 고유한 이름으로 원하는 언리얼 오브젝트를 찾을수가 있다.

(3) 만약 로드가 안되어 있다면 LoadObject()를 사용하여 원하는 패키지를 로딩 가능하다.

**\*\*애셋 스트리밍 관리자(Streamable Manager)**

## 애셋 스트리밍 관리자(Streamable Manager)

---

- 애셋의 비동기 로딩을 지원하는 관리자 객체
- 콘텐츠 제작과 무관한 싱글턴 클래스에 FStreamableManager를 선언해두면 좋음.
  - GameInstance는 좋은 선택지
- FStreamableManager를 활용해 애셋의 동기/비동기 로딩을 관리할 수 있음.
- 다수의 오브젝트 경로를 입력해 다수의 애셋을 로딩하는 것도 가능함.

이것은 애셋의 비동기 로딩을 지원해 주는 관리자 객체입니다

-이 관리자 객체는 FStreamableManager라는 매니저 타입으로 되어 있다.\

## **\*\*실습**

-생성자에서 에셋을 로딩하고 실행하면 실행하자마자 로그가 찍히고 실행하면 한번 더 로그가 찍힌다. 첫 번째는 에디터가 로딩할 때 찍히고 두 번째는 게임을 실행할 때 컨스트럭터에 관련된 함수들이 자동으로 호출되기 때문이다.

-생성자에서 에셋을 로딩하는 경우는 반드시 그 에셋이 있다는 가정하에 진행하게 된다. 따라서 우리가 Content폴더에 있는 Student.uaaset을 삭제하고 실행하면 강력한 경고가 뜨기 때문에 생성자에서 에셋을 로딩하는 경우 에셋이 빠지지 않도록 미리 잘 세팅해둬야 한다.



\*\*정리

## 언리얼 오브젝트 패키지

---

1. 언리얼 오브젝트 패키지 구조의 이해
2. 패키지 클래스를 사용한 애셋 데이터의 관리
3. 오브젝트 경로의 설계와 이를 활용한 다양한 애셋 로딩 방법의 이해



1138208

이득우의 언리얼 프로그래밍 Part1 - 언리얼 C++의 이해  
이득우

이번 시간에는 언리얼 오브젝트 패키지에 대해서 설명을 드렸었는데요