2단계_10. 게임 데이터 관리

수강일	@2023/07/12
이름	전영재
멘토	Min-Kang Song,현웅 최
작성 상태	Done
단계	2단계
강의 시청 여부	☑

Contents

게임 데이터 관리

💡 **강의 목표**

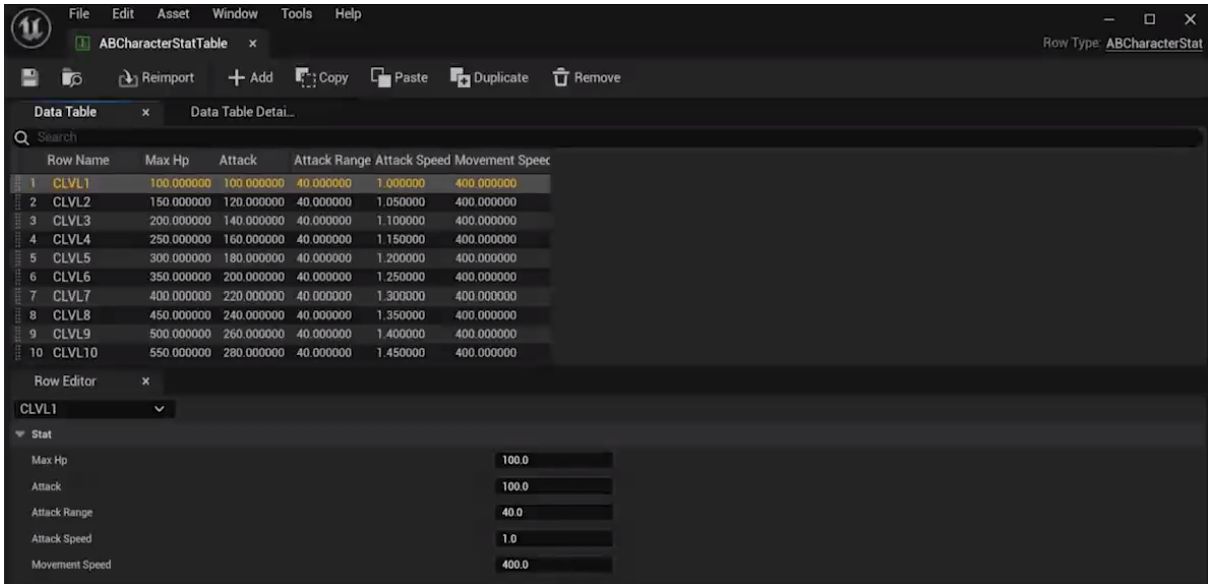
- 게임 데이터를 관리하는 싱글톤 객체의 등록
- 엑셀 데이터 및 INI 파일을 활용한 게임 데이터의 관리
- 액터의 초기화를 위한 지연 생성 기능의 이해와 활용

---

[1] 데이터 기반의 게임 시스템 구축

엑셀 파일을 **csv**형식으로 저장하고 소스파일 내에 해당 파일의 이름과 동일한 이름의 **구조체 헤더 파일**을 작성하면, 에디터 내에 해당 파일을 **임포트** 할 수 있게 된다.

데이터 애셋과 유사하게 `FTableRowBase`를 상속받은 구조체를 선언하고, 액셀의 Name컬럼을 제외한 컬럼과 동일하게 `UPROPERTY`속성을 선언해주어야 한다.




```
#pragma once

#include "CoreMinimal.h"
#include "Engine/DataTable.h"
#include "ABCharacterStat.generated.h"

USTRUCT(BlueprintType)
struct FABCharacterStat : public FTableRowBase
{
  GENERATED_BODY()

public:
  FABCharacterStat() : MaxHp(0.0f), Attack(0.0f), AttackRange(0.0f), AttackSpeed(0.0f) {}

  UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Stat)
  float MaxHp;
  UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Stat)
  float Attack;
  UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Stat)
  float AttackRange;
  UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Stat)
  float AttackSpeed;
  UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = Stat)
  float MovementSpeed;

  FABCharacterStat operator+(const FABCharacterStat& Other) const
  {
    const float* const ThisPtr = reinterpret_cast<const float* const>(this);
    const float* const OtherPtr = reinterpret_cast<const float* const>(&Other);

    FABCharacterStat Result;
    float* ResultPtr = reinterpret_cast<float*>(&Result);
    int32 StatNum = sizeof(FABCharacterStat) / sizeof(float);
    for (int32 i = 0; i < StatNum; i++)
    {
      ResultPtr[i] = ThisPtr[i] + OtherPtr[i];
    }

    return Result;
  }
};
```

이렇게 데이터 테이블을 만들었다면 이들을 관리하는 단 하나의 **싱글톤** 객체를 만드는 것이 좋다.

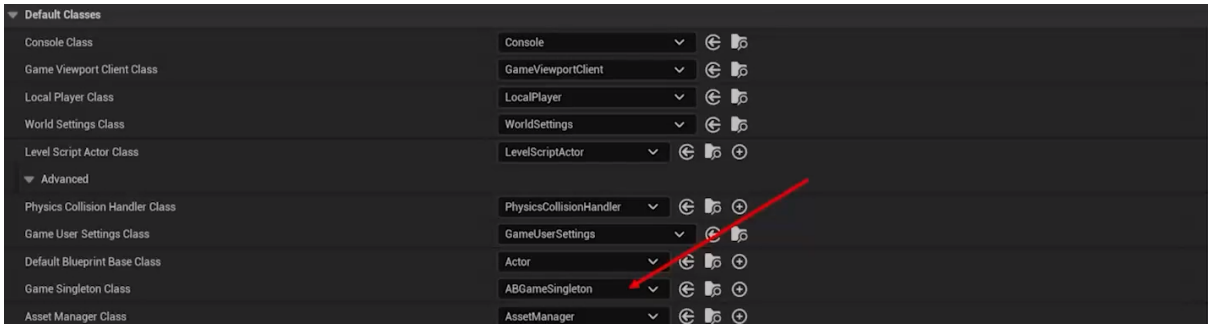
싱글톤 클래스

언리얼 엔진에서는 게임을 관리하기 위해서 다양한 **싱글톤** 클래스를 제공한다. 이들은 언리얼 오브젝트 생성자에서 사용하지 않는다.

- **게임 인스턴스**
  - 컨텐츠에 관여하기 보다 어플리케이션의 데이터를 관리하는 용도로 사용되기에, 컨텐츠를 관리하는 별도의 싱글톤 클래스를 만드는 것이 좋다.

- **애셋 매니저**
- **게임 플레이 관련 액터** (게임모드, 게임 스테이트)
- **프로젝트에 싱글톤으로 등록한 언리얼 오브젝트**

프로젝트 세팅 - 일반 설정에 가면, 우리가 만든 싱글톤 클래스를 엔진이 초기화 될 때**GEnigne**이라고 하는 전역변수에서 **자동**으로 만들어진다.



이러한 게임 싱글톤은 `UObject`를 상속받아 만들면 된다.

```
#pragma once

#include "CoreMinimal.h"
#include "UObject/NoExportTypes.h"
#include "ABCharacterStat.h"
#include "ABGameSingleton.generated.h"

DECLARE_LOG_CATEGORY_EXTERN(LogABGameSingleton, Error, All);

UCLASS()
class ARENABATTLE_API UABGameSingleton : public UObject
{
  GENERATED_BODY()

public:
  UABGameSingleton();
  static UABGameSingleton& Get(); //모든 코드에서 해당 싱글톤에 접근할 수 있게 Get함수 생성

// Character Stat Data Section
public:
  FORCEINLINE FABCharacterStat GetCharacterStat(int32 InLevel) const { return CharacterStatTable.IsValidIndex(InLevel - 1) ? Character
  UPROPERTY()
  int32 CharacterMaxLevel; //몇개의 레벨이 있는지 알기위해

private:
  TArray<FABCharacterStat> CharacterStatTable;

};
```

```
#include "GameData/ABGameSingleton.h"
DECLARE_LOG_CATEGORY_EXTERN(LogABGameSingleton);
...
UABGameSingleton& UABGameSingleton::Get()
{
  UABGameSingleton* Singleton = CastChecked< UABGameSingleton>(GEngine->GameSingleton);
  if (Singleton)
  {
    return *Singleton;
  }

  UE_LOG(LogABGameSingleton, Error, TEXT("Invalid Game Singleton"));
  return *NewObject<UABGameSingleton>(); //코드의 흐름을 위해서지 실제로 사용하는 값은 아니다. 애초에 여기까지 오면 안된다.
}
```

위와 같이 싱글톤에서 캐릭터 스탯에 대한 데이터를 관리한다면 이를 일반 코드에서 불러와 사용할 수 있게된다.

```
#include "CharacterStat/ABCharacterStatComponent.h"
#include "GameData/ABGameSingleton.h"
...
void UABCharacterStatComponent::SetLevelStat(int32 InNewLevel)
{
```

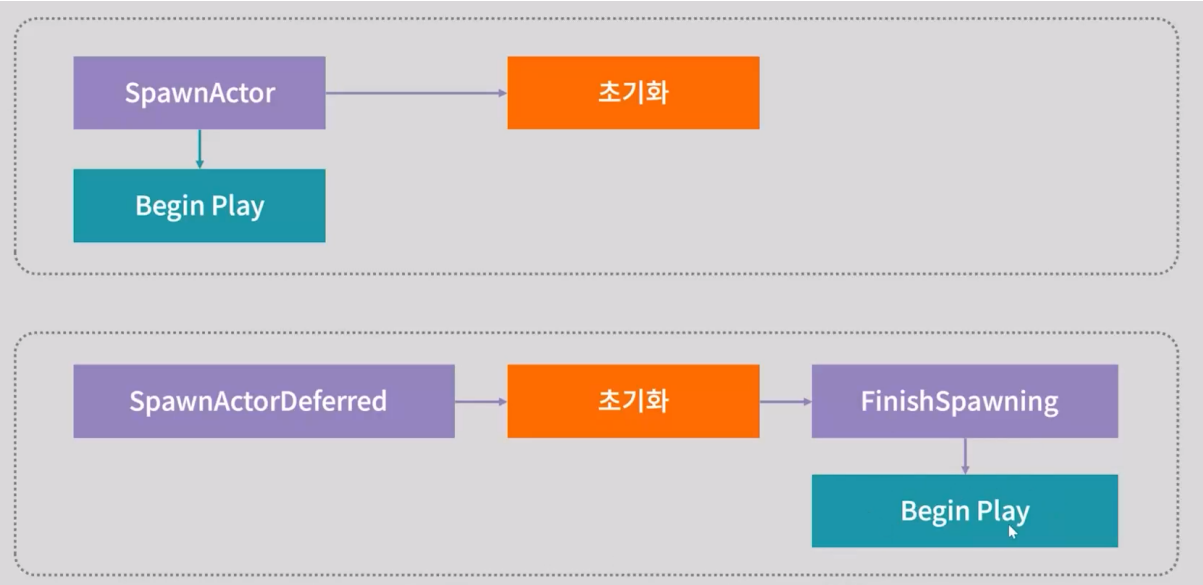
```
  CurrentLevel = FMath::Clamp(InNewLevel, 1, UABGameSingleton::Get().CharacterMaxLevel);
  BaseStat = UABGameSingleton::Get().GetCharacterStat(CurrentLevel);
  check(BaseStat.MaxHp > 0.0f);
}
```

[2] 지연생성 SpawnActorDeferred

`SpawnActor`를 사용하게 되면 **초기화와 동시에** `BeginPlay`가 작동한다. 그렇기에 해당 강의에서는 게임싱글톤이 CurrentLevel을 올리기도 전에 NPC의 초기화가 이루어져 CurrentHP가 이전 레벨의 MaxHP로 설정되는 글리치가 발생했다.

이것을 해결하기 위해서는 생성 시점을 뒤로 미루는 **지연 생성**을 사용해야 한다.

`SpawnActorDeferred`를 사용한다면 `FinishSpawning`함수를 호출한 후에 액터의 `PostInitializeComponents`와 `BeginPlay`가 작동하므로 우리가 원하는 타이밍에 액터의 체력설정을 해줄 수 있다.



[3] INI파일을 사용하여 다른 애셋 사용

NPC가 스폰될 때 각기 다른 메쉬를 사용하는 기능을 추가해보자.

프로젝트의 Config폴더 내에 **ini**파일을 생성하면 특정 UObject를 위한 데이터를 추가할 수 있다. 위 csv파일을 사용하는 것보다 생성은 간단하지만 사용법(ini를 위한 메타 및 클래스가 존재한다.)과 제작법(순수 텍스트로 생성해야 한다.)이 **특이**하기에 사용에 주의를 기하자.

```
#pragma once

#include "CoreMinimal.h"
#include "Character/ABCharacterBase.h"
#include "Engine/StreamableManager.h"
#include "ABCharacterNonPlayer.generated.h"

UCLASS(config=ArenaBattle) //config내 DefaultArenaBattle.ini를 사용하겠다는
class ARENABATTLE_API AABCharacterNonPlayer : public AABCharacterBase
{
  GENERATED_BODY()

public:
  AABCharacterNonPlayer();

protected:
  virtual void PostInitializeComponents() override;

protected:
  void SetDead() override;
  void NPCMeshLoadCompleted();

  UPROPERTY(config) //config파일로부터 데이터를 불러온다는 메타지정자
  TArray<FSoftObjectPath> NPCMeshes; //FSogtObjectPath를 사용했다
```

```
  TSharedPtr<FStreamableHandle> NPCMeshHandle; //비동기방식을 위한 핸
};
```

```
#include "Character/ABCharacterNonPlayer.h"
#include "Engine/AssetManager.h"

AABCharacterNonPlayer::AABCharacterNonPlayer()
{
  GetMesh()->SetHiddenInGame(true);
}

void AABCharacterNonPlayer::PostInitializeComponents()
{
  Super::PostInitializeComponents();

  ensure(NPCMeshes.Num() > 0);
  int32 RandIndex = FMath::RandRange(0, NPCMeshes.Num() - 1);
  NPCMeshHandle = UAssetManager::Get().GetStreamableManager().RequestAsyncLoad(
  NPCMeshes[RandIndex],
  FStreamableDelegate::CreateUObject(this, &AABCharacterNonPlayer::NPCMeshLoadCompleted)
  );
}

void AABCharacterNonPlayer::NPCMeshLoadCompleted()
{
  if (NPCMeshHandle.IsValid())
  {
    USkeletalMesh* NPCMesh = Cast<USkeletalMesh>(NPCMeshHandle->GetLoadedAsset());
    if (NPCMesh)
    {
      GetMesh()->SetSkeletalMesh(NPCMesh);
      GetMesh()->SetHiddenInGame(false);
    }
  }

  NPCMeshHandle->ReleaseHandle();
}

...
```

📌 **Summary**

- csv, ini등 외부 데이터 파일로부터 게임 데이터를 관리하는 방법
- 게임 데이터를 관리하는 싱글톤 클래스 이해
- 지연생성을 활용한 액터 초기화 방법

0n. 게임 데이터 관리 강의 과제

❓ **Q1. 현재 프로젝트에서 싱글톤 오브젝트를 활용할 수 있는 방법에 대해 생각하시오.**

게임 내에는 포탑들에게 버프를 준다던가 플레이어에게 제약을 거는 **기믹**이 존재한다.

이들에게 접근하기 위해서는 게임을 총괄하는 게임 매니저가 필요한데, 별도의 액터 매니저를 만들지 않고 게임 **싱글톤**을 생성해 플레이어 스탯이나 포탑 스테이트에 일괄적으로 영향을 끼치는 구조를 만들수 있다.

❓ **Q2. 액셀이나 INI파일을 사용해 게임에 사용되는 데이터를 로드하고 이를 적용한 기믹을 고안하시오.**

몬스터는 스테이지가 넘어가면 넘어갈수록 강해져야 한다. 이것은 일반적인 게임의 레벨 시스템과 유사하며, 스테이지 별 몬스터의 AttackDamage, HP 등을 데이터 테이블화 시켜 사용할 수 있다.