

# 언리얼 CPP

3강

# 배울 거

- struct
- datatable
- c++ 와 블루프린트의 유기적 활용 2 - UPROPERTY 옵션

# 데이터 테이블이란

- 말그대로 데이터가 테이블 형태로 모여있는거다.
- 각 행은 **Struct** 로 되어있는데 **FTableRowBase** 를 상속해야함.

# 구조체

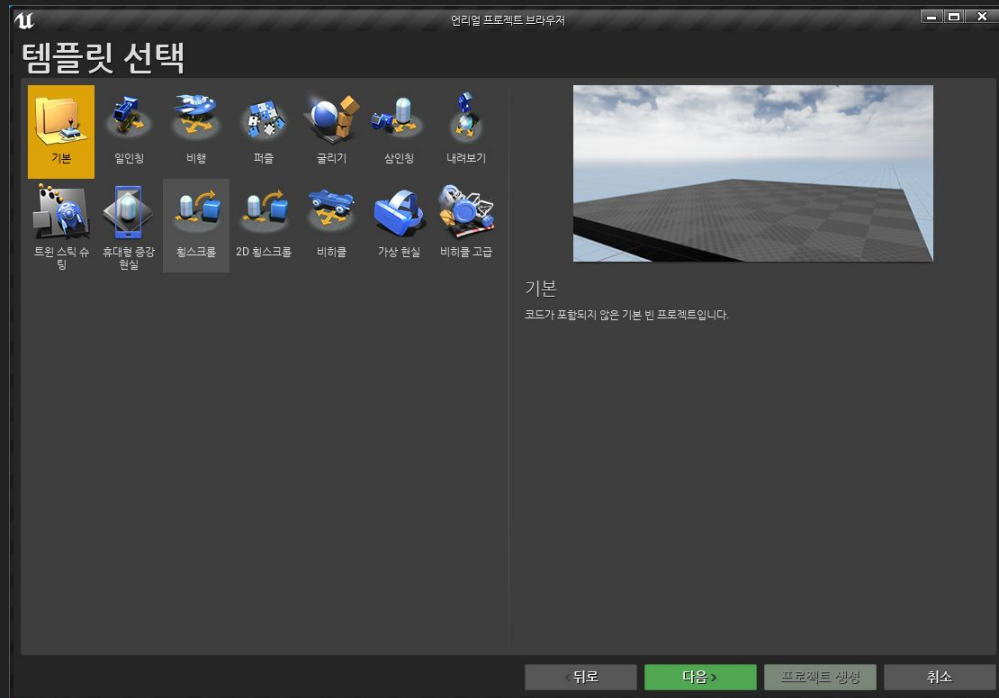
- `struct`
- 아는 그거.
- 언리얼에선 `class`와 확실히 다르다.
- 메소드를 못넣음.

# 만들 거


1. 게임을 키면
2. 내가 넣어준 데이터 테이블을 읽어서
3. 출력하는
4. GameMode

# 새로운 프로젝트


이번에 기본 으로.




# 프로젝트 세팅

  
C++


블루프린트 또는 C++ 프로젝트 생성 여부를 선택합니다.

  
데스크톱 / 콘솔


타겟 플랫폼과 가장 가까운 플랫폼을 선택합니다. 걱정하지 마세요.  
프로젝트 세팅의 타겟 하드웨어 섹션에서 나중에 변경할 수 있습니다.

  
최대 퀄리티

프로젝트의 성능 특성을 선택합니다.

  
시작용 콘텐츠 포함

켜면 기본적인 머티리얼 및 텍스처와 함께 간단히 배치가능한 메시가  
들어있는 부가 콘텐츠 팩을 포함시킵니다.  
포함시키는 옵션을 끄면 선택된 프로젝트 템플릿에 꼭 필요한 것들로만  
된 프로젝트를 만들 수 있습니다.

  
레이 트레이싱 비활성화됨

실시간 레이 트레이싱을 새 프로젝트에서 활성화해야 하는지  
선택합니다.

프로젝트를 저장할 위치를 선택하세요.

C:\Users\LJH\Documents\Unreal Projects

c2020s

폴더

이름

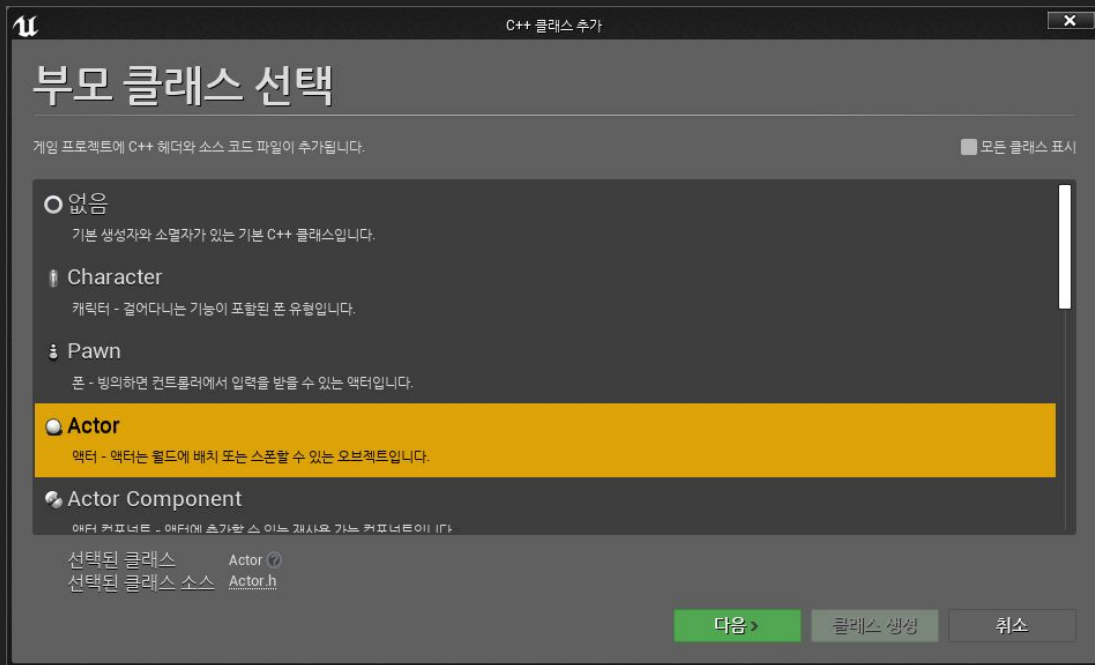
< 뒤로

프로젝트 생성

취소


# Struct 만들기 - 일단 actor로

- struct 는 바로 못만듬.





# 이름은 편하게

 C++ 클래스 추가

## 새 Actor 이름

새 클래스의 이름을 입력해 주세요. 클래스 이름은 공백을 제외한 알파벳과 숫자로만 이루어져야 합니다.  
아래 "생성" 버튼을 누르면 이 이름을 사용하여 헤더(.h) 파일과 소스(.cpp) 파일이 만들어집니다.

|       |   |                    |       |      |
|-------|---|--------------------|-------|------|
| 이름    | <input type="text" value="MyCustomStructs"/>  | c2020s (Runtime) ▼ | 퍼블릭   | 프라이빗 |
| 경로    | <input type="text" value="C:/Users/LJH/Documents/Unreal Projects/c2020s/Source/c2020s/"/> |                    | 폴더 선택 |      |
| 헤더 파일 | C:/Users/LJH/Documents/Unreal Projects/c2020s/Source/c2020s/MyCustomStructs.h             |                    |       |      |
| 소스 파일 | C:/Users/LJH/Documents/Unreal Projects/c2020s/Source/c2020s/MyCustomStructs.cpp           |                    |       |      |

◀ 뒤로

클래스 생성

취소

## .h 에 struct 를 만든다.

```
#include "CoreMinimal.h"
#include "Engine/DataTable.h"
#include "GameFramework/Actor.h"
#include "MyCosomeStructs.generated.h"

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

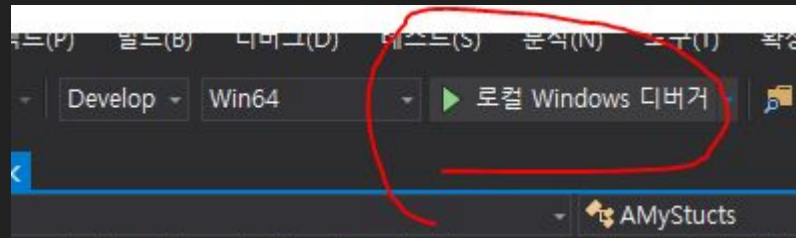
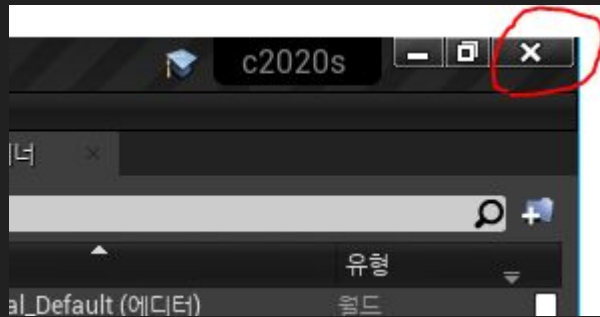
    public:

    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "int")
    int32 num1;

    UPROPERTY(EditAnywhere, BlueprintReadWrite, Category = "int")
    int32 num2;
};
```

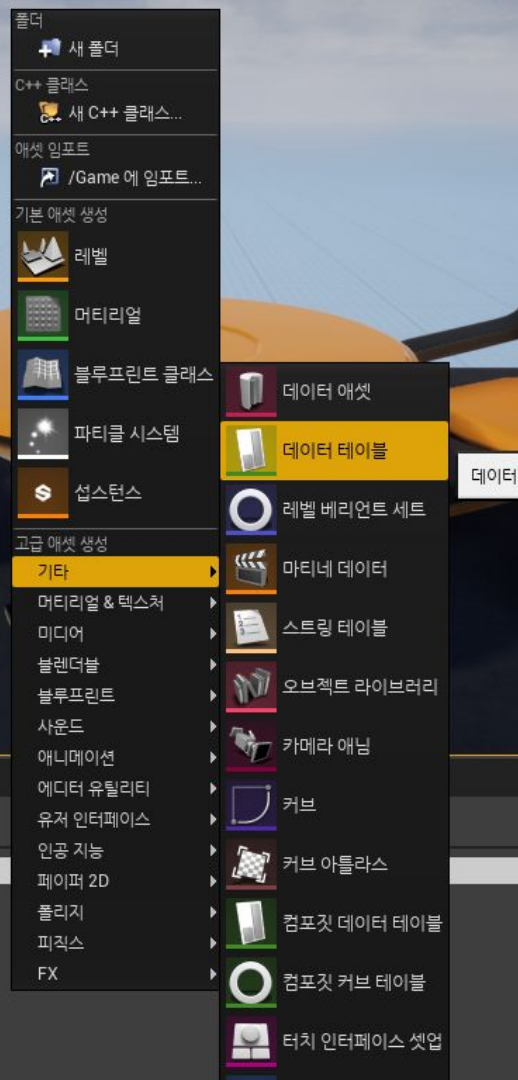
- **BlueprintReadWrite** : 블루프린트에서 읽고 쓸수 있다.
- **Category** : 블루프린트 표시에 카테고리가 있는데 카테고리를 지정한다
- 서브 카테고리는 | 로 구분한다. ex int|subcategory

엔진 끄고 빌드.

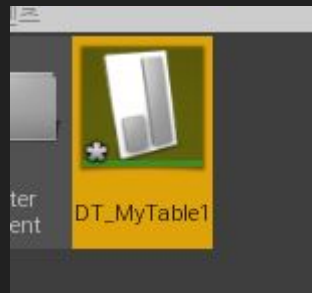
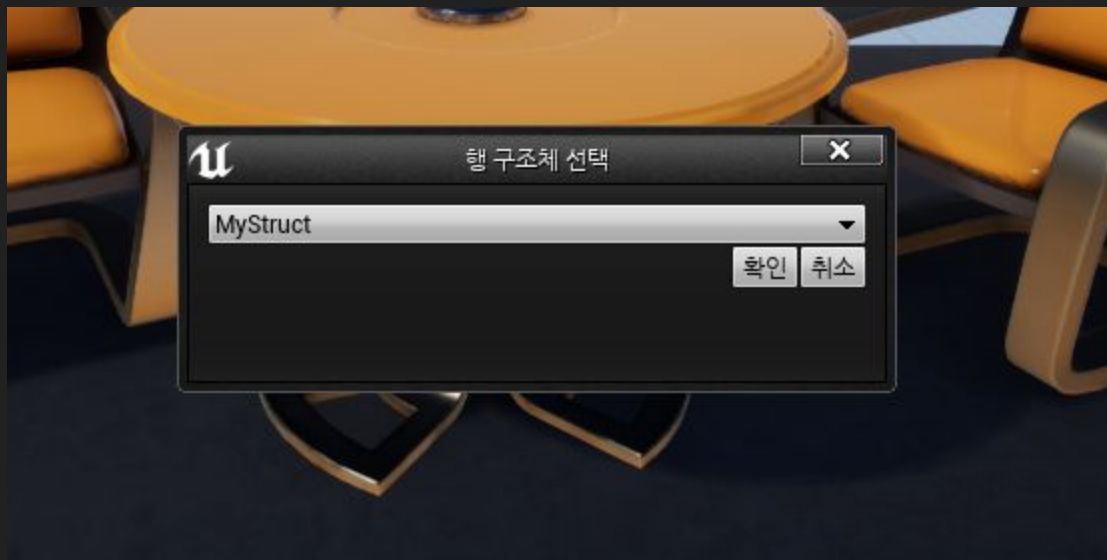


# 데이터 테이블 만들기

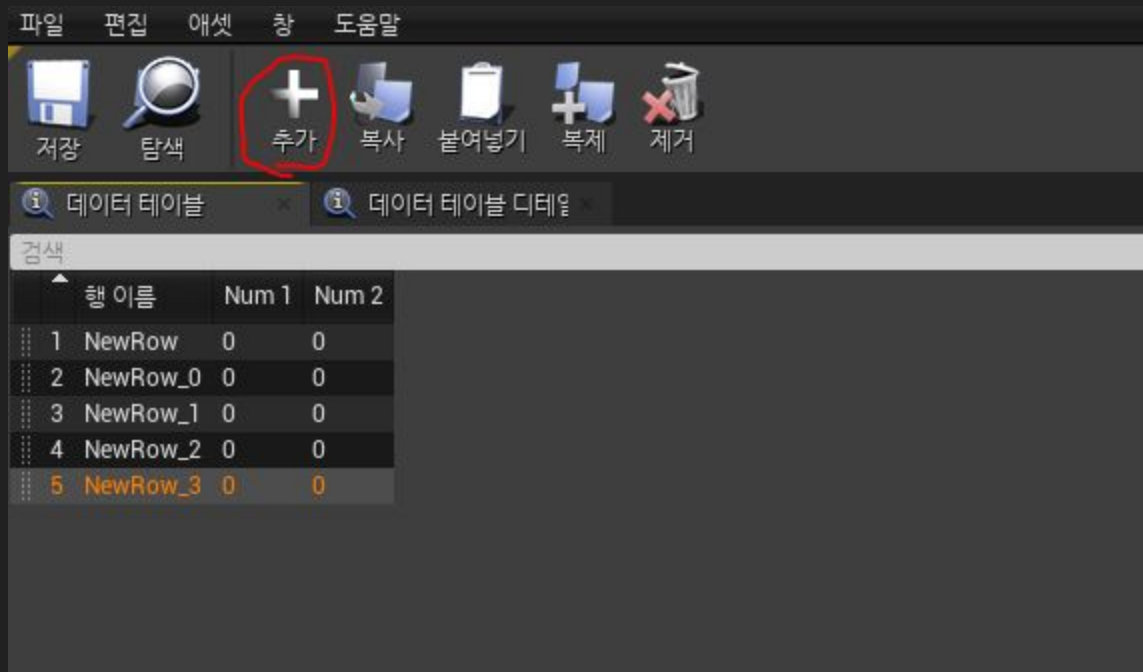
만드는 방법은 2가지 있다.



# 1. 직접 만들기.



# 1. 직접 만들기



## 2. csv 파일 импорт

미리 올려둔 파일을 받자.



11 임포트

← → ↕ ↑ > 내 PC > 문서 > 2020SummerUnrealCpp > 자료

자료 검색

구성 새 폴더

문서

사진

OneDrive

내 PC

3D 개체

다운로드

동영상

문서

바탕 화면

사진

음악

로컬 디스크 (C:)

새 볼륨 (D:)

네트워크

이름

수정된 날짜

유형

크기

3th.csv

2020-08-10 오후 4:28

CSV 파일

1KB

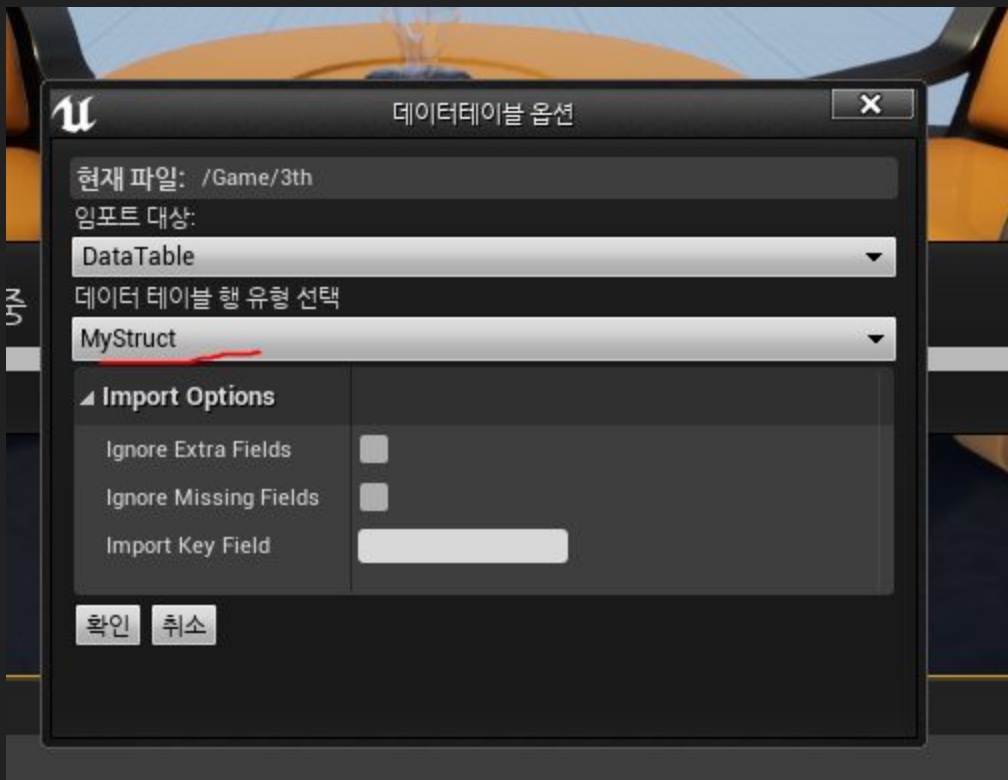
파일 이름(N):

All Files (\*.3g2;\*.3gp;\*.3gpp;\*.3

열기(O)

취소





숫자가 잘 들어가 있다.



저장    탐색    추가    복사    붙여넣기    복제    제거

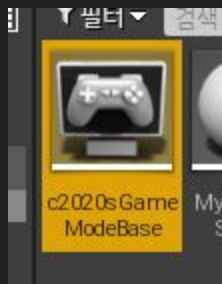
데이터 테이블    데이터 테이블 디테일

검색

| 행 | Num 1 | Num 2 |
|---|-------|-------|
| 1 | 0     | 0     |
| 2 | 1     | 10    |
| 3 | 2     | 20    |
| 4 | 3     | 30    |
| 5 | 4     | 40    |

# 이제 이걸 출력하는 GameMode 를 만들자.

사실 클래스를 이미 만들어져있다.



엔진을 다시 달자.



c2020sGameModeBase.h ✕ c2020sGameModeBase.cpp

c2020s

```
1 // Copyright Epic Games, Inc. All Rights Reserved
2
3 #pragma once
4
5 #include "CoreMinimal.h"
```

## header 에 추가

```
#include "CoreMinimal.h"
#include "GameFramework/GameStateBase.h"
#include "Engine/DataTable.h"
#include "c2020sGameStateBase.generated.h"

/**
 *
 */
UCLASS()
class C2020S_API Ac2020sGameStateBase : public AGameStateBase
{
    GENERATED_BODY()

public:

    UPROPERTY(EditAnywhere, BlueprintReadWrite)
    UDataTable* MyDataTable;

    virtual void BeginPlay() override;
};
```

# .cpp 에 구현

```
#include "c2020sGameModeBase.h"
#include "MyCostomeStructs.h"

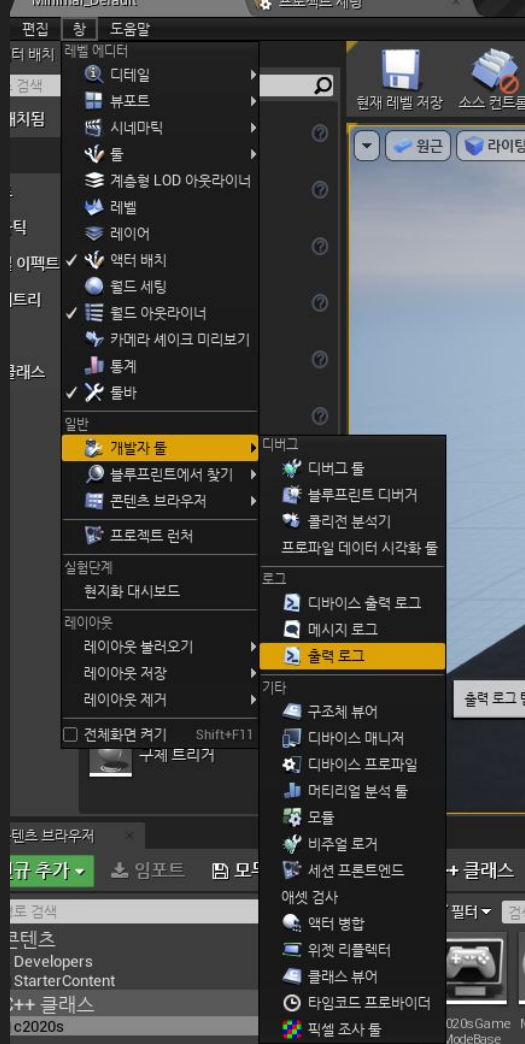
void Ac2020sGameModeBase::BeginPlay()
{
    for(FName& Name : MyDataTable->GetRowNames())
    {
        FMyStruct* Row = MyDataTable->FindRow<FMyStruct>(Name,FString(""));

        //UE_LOG
        UE_LOG(LogTemp,Log,TEXT("num1 : %d , num2 : %d"),Row->num1,Row->num2)
    }
}
```

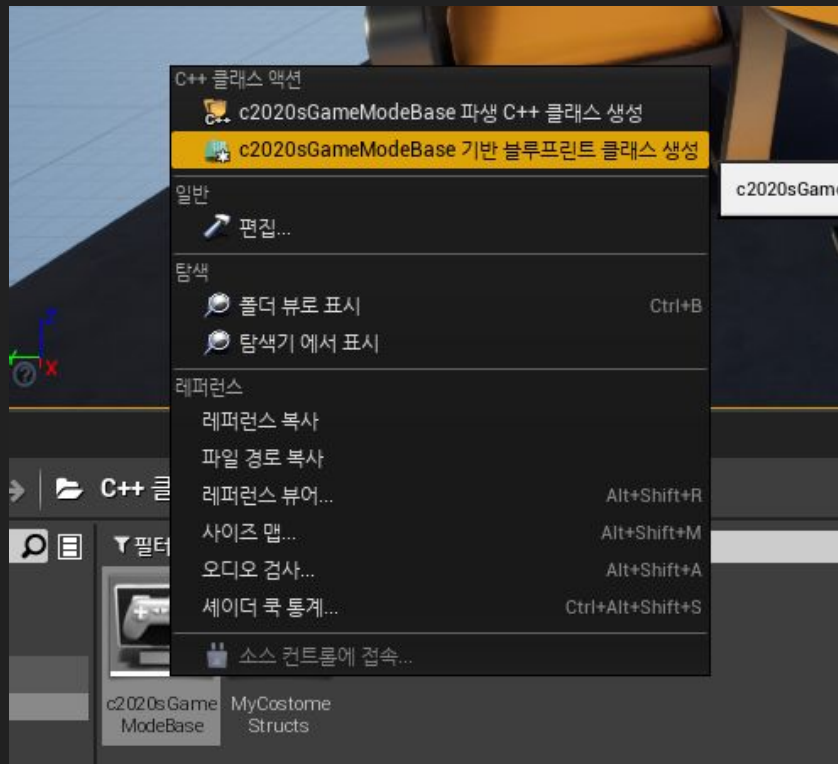
실행



# 로그창 보는법



# 게임모드베이스를 상속한 블루프린트 만들기



# 프로젝트 설정 > 맵 모드 > 만든 블루프린트 모드로.

## 프로젝트 - 맵 & 모드

기본 맵, 게임 모드, 기타 맵 관련 세팅입니다.

 이 세팅은 DefaultEngine.ini 에 저장되었으며, 현재 쓰기가능 입니다.

### Default Modes

기본 게임모드

#### 선택된 GameMode

Default Pawn Class

HUD Class

Player Controller Class

Game State Class

Player State Class

Spectator Class

Myc2020sGameModr

DefaultPawn

HUD

PlayerController

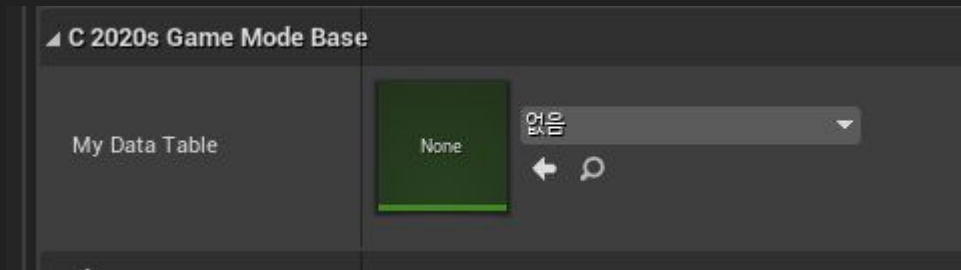
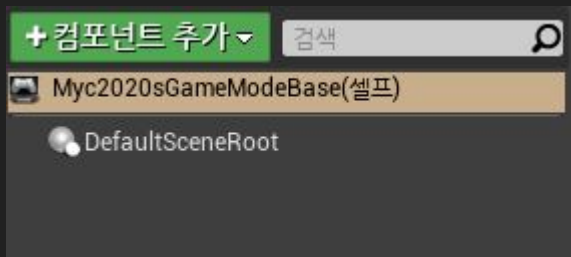
GameStateBase

PlayerState

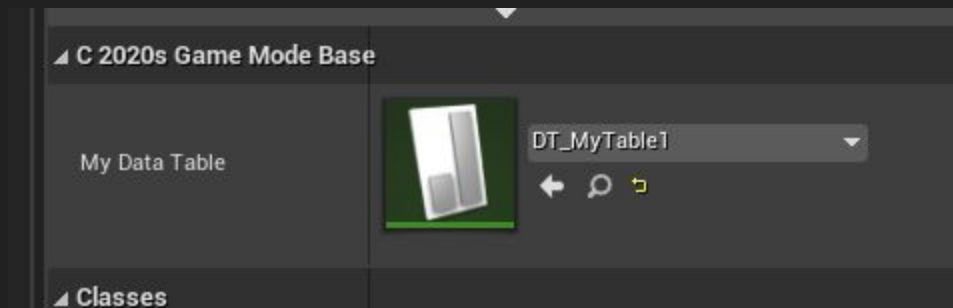
SpectatorPawn

# 만든 블루프린트를 편집하러 들어오면

데이터 테이블 넣는 칸이 있다.



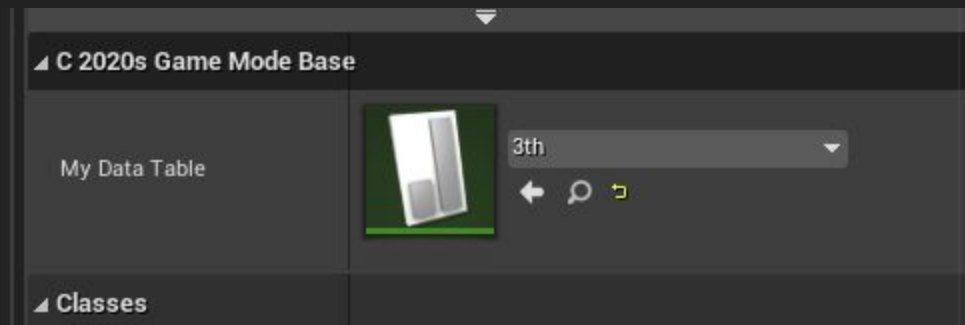
# 처음 만든거부터 넣어서 실행



# 처음거

```
LogContentBrowser: Native class hierarchy updated for 'Mo
LogTemp: num1 : 0 , num2 : 0
LogTemp: num1 : 0 , num2 : 0
LogTemp: num1 : 0 , num2 : 0
LogTemp: num1 : 0 , num2 : 0
LogTemp: num1 : 0 , num2 : 0
PIE: 서버가 로그인했습니다.
```

# 두번째거



잘 쓴다.

```
LogOnline: OSS: Bringing up level for play look: 0.0000  
LogOnline: OSS: Creating online subsystem instance  
LogTemp: num1 : 0 , num2 : 0  
LogTemp: num1 : 10 , num2 : 10  
LogTemp: num1 : 20 , num2 : 20  
LogTemp: num1 : 30 , num2 : 30  
LogTemp: num1 : 40 , num2 : 40  
PIE: 서버가 로그인했습니다.  
PIE: 에디터에서 플레이 총 시작 시간 0.168초입니다.
```



# GameMode란

게임 자체를 관리하는거라고 생각하면 편함.

원래는 게임의 규칙을 정할때 활용함.

할 수 있게 된거.

1. `cpp` 클래스를 상속해서 쓸수 있게 됨.
2. 데이터 테이블이랑 구조체를 만들수 있음.