|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **주차** | 7주차 | **기간** | 8.7 - 8.14 | **지도교수** | (서명) |
| 이번주 한일 요약 | 유데미 인터넷 강의보고 공부하기 (섹션 3)  액터 관련 소스 코드 분석 | | | | |

<상세 수행내용>

이번주부터는 본격적으로 C++코드를 언리얼에 활용하는 방법을 배웠다.

구체적으로 언리얼에서 액터를 C++코드로 만들어보는 것을 공부했는데 부모 클래스를 선택해서 여러가지 파일을 만들 수 있었다. 이를 통해 캐릭터, 폰, 액터, 컨트롤러 등등 만들 수 있는 것이 많은데 이번 강의에서는 액터에 관련된 것을 C++로 만들어 보았다. 언리얼에서 이미 제공하고 있는 액터 클래스를 상속받아 오브젝트를 만들어 보는 것인데 이를 생성해 보면 헤더파일과 cpp파일 2가지를 다룰 수 있게 된다. 이 두가지 파일을 분석해 보았는데 먼저 헤더 파일을 살펴보면

**#include "CoreMinimal.h"**

**#include "GameFramework/Actor.h"**

**#include "MovingPlatform.generated.h"**

이 3가지 헤더 파일을 포함한다.

순서대로 언리얼의 기본적인 기능을 다루는 헤더파일, 액터 클래스를 정의한 헤더파일, 클래스의 속성, 함수등 런타임에 제대로 사용할 수 있도록 도와주는 코드 파일을 도와주는 코드 파일을 포함한다.

이를 하나하나 정의를 들어가 봤지만 너무 방대하고 복잡해서 어떤 역할을 하는지만 이해했다.

**UCLASS()**

이는 언리얼엔진의 클래스를 정의할 때 사용되는 매크로이다.

이를 선언해줌으로서 밑에 정의하는 클래스로 만들어진 객체들을 동적으로 생성, 수정 및 관리를 할 수 있게 해준다. 이 매크로를 통해 언리얼을 고급기능과 연동하여 클래스를 게임 엔진과 통합하고 활용할 수 있게 해주는 것이다.

**class LEC2\_API AMovingPlatform : public AActor**

앞에 A를 붙여 Actor라는 것을 나타내고 AActor를 상속받는 AMovingPlatform 클래스의 정의이다.

LEC2\_API를 사용하여 프로젝트 이름 ‘LEC2’와 함께 클래스나 함수등을 정의하면 해당 코드가 프로젝트 LEC2의 일부로 사용되며 다른 모듈에서 해당 기능을 활용할 수 있도록 가시성을 위해 달아 놓은 것이다. 보통 API를 포함하는 매크로는 언리얼에서 가시성과 관련된 설정에서 사용되는 관례가 있다.

이를 이용해 코드가 어떤 역할을 하는지 파악하기 쉽다.

**{**

**GENERATED\_BODY()**

이는 언리얼에서 클래스 선언내에 사용되는 매크로인데 언리얼의 코드 생성 시스템에서 멤버 변수 초기화, 가상 함수 재정의, 내부함수 구현등등 필요한 코드를 자동으로 생성해준다.

**public:**

**AMovingPlatform();**

클래스의 생성자

**protected:**

**virtual void BeginPlay() override;**

게임이 시작될 때 호출되는 함수, 주로 초기화를 담당한다.

**public:**

**virtual void Tick(float DeltaTime) override;**

게임 프레임마다 호출되는 함수, 게임 로직의 주된 처리부분이다.

이 함수는 프레임 간의 시간간격을 나타내는 델타 타임을 매개변수로 이용하는데

이를 통해 게임의 속도가 프레임율에 의존하지 않고 시간간격에 따라 변화하는 물리량을 정확하게 업데이트 할 수 있다.

여기까지가 기본 헤더파일의 구조이고 이 밑으로 내가 사용할 변수를 정의해 사용한다.

**UPROPERTY(EditAnywhere, Category = "Moving Platform")**

**FVector PlatformVelocity = FVector(100, 0, 0);**

첫번째 줄은 언리얼에서 사용되는 매크로 중 하나로 클래스 멤버 변수의 속성을 지정하기 위해 사용되는데 클래스 멤버변수가 에디터에서 수정 가능하게 해준다. 또한 에디터에서 카테고리를 설정해 줄 수 있다. 두번째 줄은 변수 설정인데 벡터 변수를 선언하고 초기화 해주는 과정이다.

**UPROPERTY(VisibleAnywhere)**

**float DistanceMoved = -1;**

이런식으로 선언해주면 이를 읽기전용 변수로 정의해준다.

**FVector StartLocation;**

**};**

C++만을 위한 벡터는 UPRORERTY 안붙여도 된다.

다음으로 cpp파일을 살펴보면

**#include "MovingPlatform.h"**

위의 헤더파일을 포함시키고 있다.

**AMovingPlatform::AMovingPlatform()**

**{**

**PrimaryActorTick.bCanEverTick = true;**

**}**

클래스의 생성자,

이 생성자에서는 객체가 매 프레임마다 Tick()함수를 호출하는 것을 참으로 하고 있다. PrimaryActorTick은 액터의 기본적인 Tick설정을 다루는 구조체이고, bCanEverTick은 Tick 함수를 사용할지 여부를 설정하는 불변수이다.

**void AMovingPlatform::BeginPlay()**

**{**

**Super::BeginPlay();**

**}**

게임이 시작될 때 호출되는 함수로 부모클래스인 AActor의 BeginPlay()함수를 호출하는 코드를 담고 있다. 이를 통해 자식 클래스의 BeginPlay()함수에서 실행되어야 하는 기본적인 작업을 처리하거나 설정을 초기화하는 등의 역할을 한다.

**void AMovingPlatform::Tick(float DeltaTime)**

**{**

**Super::Tick(DeltaTime);**

**}**

매 프레임마다 호출되는 함수이다. 이 함수의 정의도 부모 클래스의 Tick함수를 호출하는 코드를 담고 있다. 이를 통해 부모 클래스의 기본동작을 유지하면서 자식 클래스는 추가적인 동작을 할수 있게끔 한다. 보통 여기에 게임의 로직관련 코드를 작성한다.

강의를 보며 이 두 파일을 이용해 언리얼 내에서 여러 기능을 가진 액터를 만들어 보았다.

회전하는 액터, 움직이는 액터등을 만들어 기본적인 게임을 구현해 보았다.

|  |  |  |  |
| --- | --- | --- | --- |
| **문제점 정리** |  | | |
| **해결방안** |  | | |
| **다음주차** | 8주차 | **다음기간** | 8.14 - 8.20 |
| **다음주 할일** | 유데미 강의 섹션4  관련 소스 코드 분석 | | |
| **지도 교수**  **Comment** |  | | |