

언리얼 프로그래밍 Part1-5

제목:언리얼 오브젝트 리플렉션 시스템 I

**강의 내용 : 언리얼 오브젝트 시스템의 이해

**강의 목표 :

강의 목표

- 언리얼 오브젝트의 특징과 리플렉션 시스템의 설명
- 언리얼 오브젝트의 처리 방식의 이해



1138208

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

언리얼 오브젝트의 특징과 언리얼 오브젝트의 핵심을 구성하는

****언리얼 오브젝트의 특징**

- 공식 링크: <https://www.unrealengine.com/ko/blog/unreal-property-system-reflection>

- 원래대로 라면 '리플렉션 시스템'으로 이야기하는 것이 보통이지만, 모든 사람을 대상으로 글을 만들다 보니 리플렉션이라는 '반사'를 의미하는 그래픽용어와 혼동을 피하기 위해 '프로퍼티 시스템'이라고도 불림.
- 리플렉션은 프로그램이 실행시간에 자기자신을 조사하는 기능임.
- c++언어는 리플렉션을 지원하지 않기 때문에 언리얼이 자체적으로 구축함.
- 언리얼 엔진에서는 일반 c++ 객체와 언리얼 오브젝트 객체를 두가지로 나눠야 하며 리플리케이션 시스템은 우리가 옵션으로 사용할 수 있다.
- 이러한 리플렉션 시스템에 보이도록 하는 유형이나 프로퍼티에 주석을 달아주면 언리얼 헤더툴이 프로젝트를 컴파일 할 때 본견적인 컴파일전 관련된 헤더파일(generated.h)과 cpp 파일을 저장하는데 여기에 리플렉션 시스템도 들어가 있음.
- 이것을 진행하기 위해 리플렉션 유형이 있는 언리얼 오브젝트 타입에는 generated.h와 같은 특수한 헤더파일을 추가해줘야함.
- 우리가 사용하는 개체에다가 UENUM(), UCLASS(), USTRUCT(), UFUNCTION(), UPROPERTY()와 같은 매크로를 앞에다가 집어넣어주면 언리얼 헤더툴이 이것을 보고 분석하여 자동으로 필요한 소스코드(리플렉션 시스템 같은거)를 구성해준다고 이해하자.

**** 공식 홈페이지 소스코드 분석**

-클래스 위에 매크로 선언

UCLASS(Abstract)

class AStrategyChar : public ACharacter, public IStrategyTeamInterface

-멤버변수 위에 매크로 선언

UPROPERTY(EditAnywhere, Category=Pawn)

int32 ResourcesToGather;

-멤버 함수에도 매크로 선언

UFUNCTION(BlueprintCallable, Category=Attachment)

void SetWeaponAttachment(class UStrategyAttachment* Weapon);

-이런 매크로들을 추가하면 UHT가 보고 적절한 코드들을 Intermediate폴더 내부에 생성해 줌.

-예전 코드라 GENERATED_UCLASS_BODY() / GENERATED_USTRUCT_BODY() 는 GENERATED_BODY()로 통합됨.

-매크로 안에 추가적인 메타데이터들은 에디터와 직접 연동하여 게임을 실제 제작할 때 사용 되는 옵션이다.

ex)EditAnywhere, Category=Pawn

-uint8 MyTeamNum; 매크로가 없는 멤버변수인데 모든 멤버변수가 UPROPERTY로 선언될 필요가 없다. 하지만 UPROPERTY가 아닌 멤버변수들은 언리얼엔진 시스템에서 관리받지 않는다는 것만 알아두자.

-언리얼 엔진도 가비지컬렉터와 같은 자동 메모리 관리 기능을 지원하는데 UPROPERTY를 선언해줘야 자동으로 메모리를 관리해줌. 없으면 우리가 직접 관리해줘야 한다.

-UHT는 실제 C++ 언어를 언리얼 헤더툴이 분석하여 컴파일을 진행하는 것이 아님. 그렇기에 정교하지는 않음

-너무 복잡한 유형은 언리얼 헤더툴이 읽지 못함.

-리플렉션 데이터를 사용할때 다음과 같은 계층 구조를 알아야 한다.(기본적으로 UStruct라고 하는 클래스부터 리플렉션이 시작됨을 알 수 있다.)

UField (가장 상위)

UStruct(UField 상속)

UClass (C++ class) (UStruct 상속)

UScriptStruct (C++ struct) (UStruct 상속)

UFunction (C++ function) (UStruct 상속)

UEnum (C++ enumeration) (UField 상속)

UPROPERTY (C++ member variable or function parameter) (UField 상속)

(Many subclasses for different types)

-UClass라고 하는 언리얼 오브젝트를 정의하는 클래스는 UStruct를 상속받아 여기에 함수 정보를 추가하여 전체 클래스 정보를 구축한다 이렇게 이해해보자.

- 구축된 언리얼 오브젝트 클래스는 StaticClass() 나 GetClass()를 통해 접근이 가능하다.
- 이함수를 호출하면 리플렉션, 언리얼 헤더툴이 분석하여 만들어놓은 리플렉션 정보들을 보관한 특정 객체에 접근 가능함을 뜻함.
- 이렇게 객체에 접근하게 되면 UPROPERTY라는 언리얼 오브젝트 사용해서 순회를 하여 언리얼 오브젝트의 속성들을 조회하여 값이나 정보들을 빼내올 수 있다.
- 이러한 시스템이 언리얼 엔진에서 구축한 게임 시스템의 근간을 이루게 된다.
- 꼭 쓸필요는 없지만 이해해야지만 언리얼 엔진을 더 잘 이해할 수 있게 된다.
- 각 유형에는 플래그 세트와 같은 부가적인 정보가 들어가 있어 특정 정보를 얻어내는 것도 필요한 것들만 필터링 해서 얻을 수 있다.
- 리플렉션 데이터를 사용하면 여러 가지 많은 작업이 가능하다.

- 커튼 안쪽 살펴보기

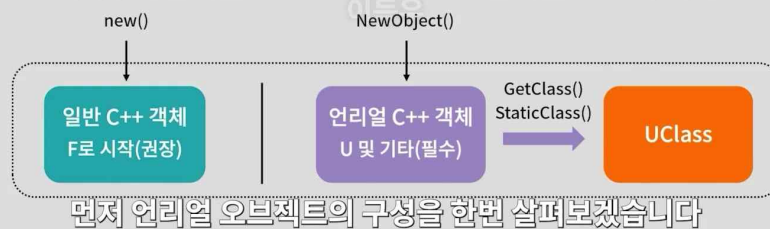
(1) Unreal Build Tool(UBT)와 Unreal Header Tool(UHT)가 함께 컴파일 되기 전에 우리가 만든 소스를 분석하여 전체 시스템이 자동으로 구축된다.

(2) StaticClass()함수들은 컴파일 타임에서 리플렉션 정보에 직접 접근 가능한 함수인데 모두 generated.h에 선언되어 있어서 실제 선언할때는 함수 지정을 안했음지언정 언리얼헤더툴이 자동으로 생성해준다 라고 이해하자.

**언리얼 오브젝트의 구성

언리얼 오브젝트의 구성

- 언리얼 오브젝트에는 특별한 프로퍼티와 함수를 지정할 수 있음.
 - 관리되는 클래스 멤버 변수 : UPROPERTY
 - 관리되는 클래스 멤버 함수 : UFUNCTION
 - 에디터와 연동되는 메타데이터를 심을 수 있음
- 모든 언리얼 오브젝트는 클래스 정보와 함께 함.
 - 클래스를 사용해 자신이 가진 프로퍼티와 함수 정보를 컴파일 타임과 런타임에서 조회할 수 있음.
- 이렇게 다양한 기능을 제공하는 언리얼 오브젝트는 NewObject API를 사용해 생성해야 함.



-UPROPERTY와 UFUNCTION이라는 매크로 안에는 여러 메타데이터를 심을 수 있고 이는 게임 콘텐츠를 제작할 때 에디터와 연동되어 유용한 기능을 만들 수 있다.

-이러한 모든 언리얼 오브젝트는 항상 클래스 정보(UClass)와 함께 한다.

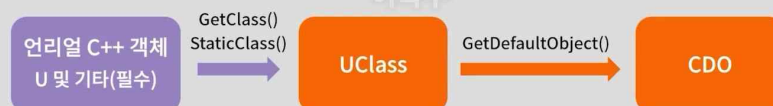
-UClass를 사용하여 자신의 프로퍼티, 클래스이름, 함수정보와 같은 다양한 정보를 컴파일 타임과 런타임에서 조회가 가능하다.

-언리얼 오브젝트는 다양한 기능을 제공하기 때문에 c++의 new 키워드가 아닌 NewObject()라는 API 키워드를 사용하여 생성해야 한다.

**언리얼 오브젝트의 클래스 기본 오브젝트

언리얼 오브젝트의 클래스 기본 오브젝트

- 언리얼 클래스 정보에는 클래스 기본 오브젝트(Class Default Object)가 함께 포함되어 있음.
- 클래스 기본 오브젝트는 줄여서 CDO라고 부름.
- CDO는 언리얼 객체가 가진 기본 값을 보관하는 템플릿 객체임.
- 한 클래스로부터 다수의 물체를 생성해 게임 콘텐츠에 배치할 때 일관성 있게 기본 값을 조정하는데 유용하게 사용됨.
- CDO는 클래스 정보로부터 GetDefaultObject 함수를 통해 얻을 수 있음.
- UClass 및 CDO는 엔진 초기화 과정에서 생성되므로 콘텐츠 제작에서 안심하고 사용할 수 있음.



이 CDO는 언리얼 객체가 가진 기본값을

****언리얼 오브젝트 처리**

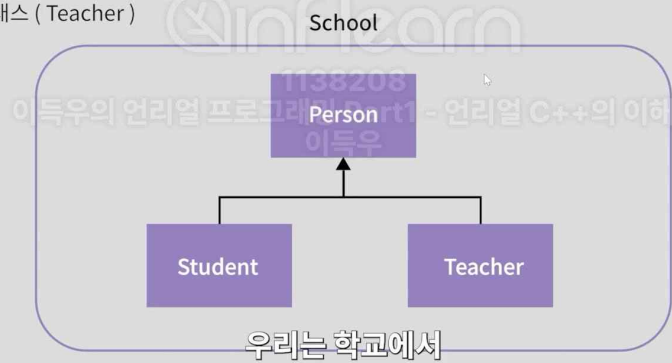
-링크: https://dev.epicgames.com/documentation/ko-kr/unreal-engine/unreal-object-handling-in-unreal-engine?application_version=5.1

- 클래스, 프로퍼티, 함수에 적합한 매크로로 마킹해 주면 UClass, UProperty, UFunction 으로 변환됩니다. 그러면 언리얼 엔진이 접근할 수 있어, 다수의 내부적인 처리 기능을 구현할 수 있습니다.
- 자동 프로퍼티 초기화 : UObject 는 생성자 호출 전 초기화시 자동으로 0 으로 채워집니다. 클래스, UProperty, 네이티브 멤버 모두에게 전체적으로 벌어지는 일입니다. 그 이후 멤버는 클래스 생성자의 커스텀 값으로 초기화 가능합니다.
- Serialization직렬화는 언리얼 오브젝트의 객체를 우리가 지정된 포맷에 맞게 저장하거나 불러들이는 것을 일괄적으로 진행할 수 있는데 저장되는 정보들은 UProperty라고 명시한 것들만 언리얼 오브젝트에서 빼워서 디스크에 저장하고 불러들일수 있다.
- 클래스 디폴트 오브젝트를 활용하여 하나의 클래스를 게임에 대해 여러 가지를 배치했을 때 기본 값을 효과적인 관리가 가능하다.
- 매크로 안에 메타데이터를 넣어주면 에디터와 통합되면서 유용한 기능들을 우리가 사용할 수 있게 된다.
- 이러한 리플렉션 시스템을 사용하면 런타임에서 정보를 얻을수 있고 안정한 형변환(캐스팅)이 가능해진다.
- 어떤 함수를 오버라이드 받을 때 super같은 키워드들도 리플렉션 시스템이 있기에 이 정보 설정이 가능한 것이다.
- 형변환이 실패하는 경우 null을 완벽히 보장해주기 때문에 if문을 사용하여 안전한 코딩 진행이 가능하다.
- 더 이상 사용하지 않는 언리얼 오브젝트들은 자동으로 회수하여 메모리에서 소멸시킨다==>가비지 컬렉션
- 네트워크 리플리케이션==>직렬화와 비슷한 기능인데 UProperty를 지정하면 자동으로 디스크에 저장하고 불러들일 수 있는것처럼 네트워크 통신을 통해서도 해당 UProperty를 자동으로 전송하고 받을 수가 있게 되는 시스템이 언리얼 내부적으로 자동으로 구축이 되어 있다.

**예제를 위한 클래스 다이어그램

예제를 위한 클래스 다이어그램

- 어떤 학교에서 학생과 교수가 함께 수업하는 상황의 구현
- 학교 정보는 GameInstance에서 지정
- 인물 클래스 (Person)
 - 학생 클래스 (Student)
 - 선생 클래스 (Teacher)



-학생과 선생은 인물클래스를 상속받도록 만들어 볼것임.

-예제: 클래스 이름 가져오기

// Fill out your copyright notice in the Description page of Project Settings.

```
#include "MyGameInstance.h"

UMyGameInstance::UMyGameInstance()
{
}


void UMyGameInstance::Init()
{
    Super::Init();
    UE_LOG(LogTemp, Log, TEXT("====="));

    UClass* ClassRuntime = GetClass();
    //GetClass라는 함수를 통해UMyGameInstance에 대한 클래스 정보를 런타임에서 얻을 수 있다.
    UClass* ClassCompile = UMyGameInstance::StaticClass();
    check(ClassRuntime == ClassCompile);
    /*
    컴파일 타임에서 얻기 위해서는UMyGameInstance의
    스태틱클래스라는 함수를 통해서 해당 클래스 정보를 얻어올 수 있음

    위 두 클래스는 동일해야 한다. check()라는 함수 사용
    */

    UE_LOG(LogTemp, Log, TEXT("학교를 담당하는 클래스 이름 : %s"),
    *ClassRuntime->GetName());
    //체크를 통과해야 학교 담당 클래스 이름을 볼 수 있다.
    //코드에서 한글을 쓰기 때문에 encoding 방식 바꾸자.

    UE_LOG(LogTemp, Log, TEXT("====="));
}
```



ObjectRefletion Crash Reporter

An Unreal process has crashed: UE-ObjectRefletion

We are very sorry that this crash occurred. Our goal is to prevent crashes like this from occurring in the future. Please help us track down and fix this crash by providing detailed information about what you were doing so that we may reproduce the crash and fix it quickly. You can also log a Bug Report with us using the [Bug Submission Form](#) and work directly with support staff to report this issue.

Thanks for your help in improving the Unreal Engine.

Please provide detailed information about what you were doing when the crash occurred.

Crash reports comprise diagnostics files (click here to view directory) and the following summary information:

LoginId:3ef36fed478c88da7251b2bd809454dd
EpicAccountId:f46a8ee72c97429e938b99f65e414149

Assertion failed: ClassRuntime != ClassCompile [File:E:\HighProject\ObjectRefletion\Source\ObjectRefletion\MyGameInstance.cpp] [Line: 18]

UnrealEditor_ObjectRefletion!UMyGameInstance::Init() [E:\HighProject\ObjectRefletion\Source\ObjectRefletion\MyGameInstance.cpp:18]
UnrealEditor_Engine!UGameInstance::InitializeForPlayInEditor() [D:\build\++UE5\Svc\Engine\Source\Runtime\Engine\Private\GameInstance.cpp:18]
UnrealEditor_UnrealEd!UEditorEngine::CreateInnerProcessPIEGameInstance() [D:\build\++UE5\Svc\Engine\Source\Editor\UnrealEd\Private\PlayLevelEditor.cpp:18]
UnrealEditor_UnrealEd!UEditorEngine::OnLoginPIEComplete_Deferred() [D:\build\++UE5\Svc\Engine\Source\Editor\UnrealEd\Private\PlayLevelEditor.cpp:18]
UnrealEditor_UnrealEd!UEditorEngine::CreateNewPlayInEditorInstance() [D:\build\++UE5\Svc\Engine\Source\Editor\UnrealEd\Private\PlayLevelEditor.cpp:18]
UnrealEditor_UnrealEd!UEditorEngine::StartPlayInEditorSession() [D:\build\++UE5\Svc\Engine\Source\Editor\UnrealEd\Private\PlayLevelEditor.cpp:18]
UnrealEditor_UnrealEd!UEditorEngine::StartPlayInEditorSession() [D:\build\++UE5\Svc\Engine\Source\Editor\UnrealEd\Private\PlayLevelEditor.cpp:18]

☒ Include log files with submission. I understand that logs contain some personal information such as my system and user name.
 ☒ I agree to be contacted by Epic Games via email if additional information about this crash would help fix it.

Close Without Sending

Send and Close

Send and Restart

-우리가 코딩할 때 돌다리도 두들긴다는 마음으로 check구문을 꾸준히 넣어줘야 한다. 실제 게임에서는 check구문은 모두 사라짐.

`ensure(ClassRuntime != ClassCompile);` 와 같이 `ensure` 함수를 통해 에디터 뺏는거 없이 결과를 볼 수 있는데 로그를 보면 위에 빨간색 로그로, `Ensure condition failed`라고 하는 에러 로그가 출력 됨을 확인할 수 있다.

`ensureMsgf(ClassRuntime != ClassCompile, TEXT("일부러 에러 발생 시킨 코드"))`; 과 같이 쓰자.

****Class Default Object(CDO) 예제**

// Fill out your copyright notice in the Description page of Project Settings.

```
#include "MyGameInstance.h"
```

```
UMyGameInstance::UMyGameInstance()
```

```
{
```

```
    SchoolName = TEXT("기본학교");
```

```
    //이 기본값은 CDO라고 하는 템플릿 객체에 저장되어있음
```

```
}
```

```
void UMyGameInstance::Init()
```

```
{
```

```
    Super::Init();
```

```
    UE_LOG(LogTemp, Log, TEXT("====="));
```

```
    SchoolName = TEXT("청강문화산업대학교");
```

//기본 객체와 무관하게 생성된 MyGameInstance에는 "청강문화산업대학교"라는 학교 이름이 설정 됨.

```
    UE_LOG(LogTemp, Log, TEXT("학교 이름 : %s"), *SchoolName);
```

```
    UE_LOG(LogTemp, Log, TEXT("학교 이름 기본값 : %s"),
```

```
*GetClass()->GetDefaultObject<UMyGameInstance>()->SchoolName);
```

```
    /*
```

이상태로 컴파일 하면 글자가 안나올수도 있는데

CDO는 이 에디터가 활성화 되기 이전에 초기화 되는 순서를 가지고 있기 때문에
에디터에서 인지를 못하는 경우가 종종있다.

따라서 CDO를 고쳐주는 (기본값을 고쳐주는) 생성자 코드를 고치는 경우에는
에디터를 꺼줘야 한다.!

```
    */
```

```
    UE_LOG(LogTemp, Log, TEXT("====="));
```

```
}
```

-규칙을 정리하자면

헤더파일의 리플렉션 정보의 구조를 변경하거나 생성자 코드에서 CDO의 기본값을 변경하는 경우
에디터를 끄고 컴파일에서 다시 실행을 해줘야 안전하다.

-CDO가 실행되는 시점을 보기위해 생성자 초기화 부분에 브레이크를 걸고 디버깅을 하면 에디터가 75%가 지난시점에서 브레이킹이 걸리는데 즉 엔진이 초기화가 되는 과정에서 CDO나 UClass 정보들이 만들어지는것이며 이것들이 다 만들어진 이후에 에디터나 게임과 같은 어플리케이션이 가동되는 순서로 진행이 된다.

**이번 강의 정리

언리얼 오브젝트 시스템

1. 언리얼 오브젝트에는 항상 클래스 정보를 담은 UClass 객체가 매칭되어 있다.
2. UClass로부터 언리얼 오브젝트의 정보를 파악할 수 있음.
3. UClass에는 클래스 기본 오브젝트(CDO)가 연결되어 있어 이를 활용해 개발의 생산성을 향상시킬 수 있음.
4. 클래스 정보와 CDO는 엔진 초기화 과정에서 생성되므로 게임 개발에서 안전하게 사용 가능.
5. 헤더 정보를 변경하거나 생성자 정보를 변경하면 에디터를 끄고 컴파일하는것이 안정적임.

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

앞으로 강의에서 언리얼 오브젝트가 가지고 있는