

언리얼 프로그래밍 Part1-13

제목:언리얼 오브젝트 관리 I - 직렬화

**강의 내용 : 언리얼 엔진이 제공하는 직렬화 기능을 이해하고 언리얼 오브젝트의 데이터를 저장하고 불러들이기

**강의 목표 :

강의 목표

- 언리얼 엔진이 제공하는 직렬화 기능의 이해
- 언리얼 오브젝트를 직렬화하고 이를 저장하고 불러들이는 방법의 이해



1138208

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

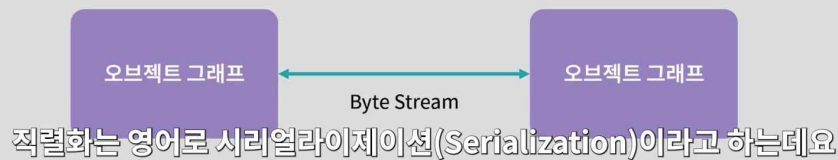
이번 강의의 목표는 다음과 같습니다

***언리얼 엔진의 직렬화

**직렬화(Serialization)란?

직렬화(Serialization)란?

- 오브젝트나 연결된 오브젝트의 묶음(오브젝트 그래프)을 바이트 스트림으로 변환하는 과정
 - 복잡한 데이터를 일렬로 세우기 때문에 직렬화
- 거꾸로 복구시키는 과정도 포함해서 의미
 - 시리얼라이제이션(Serialization) : 오브젝트 그래프에서 바이트 스트림으로
 - 디시리얼라이제이션(Deserialization) : 바이트 스트림에서 오브젝트 그래프로
- 직렬화가 가지는 장점
 - 현재 프로그램의 상태를 저장하고 필요한 때 복원할 수 있다. (게임의 저장)
 - 현재 객체의 정보를 클립보드에 복사해서 다른 프로그램에 전송할 수 있다.
 - 네트워크를 통해 현재 프로그램의 상태를 다른 컴퓨터에 복원할 수 있다. (멀티플레이어 게임)
 - 데이터 압축, 암호화를 통해 데이터를 효율적이고 안전하게 보관할 수도 있음.



-직렬화란 복잡한 데이터를 한 줄로 변환 작업 뿐만 아니라 거꾸로 한 줄로된 데이터를 복잡한 오브젝트로 복구하는 작업까지 포함한다.

-복잡한 오브젝트들 간에 데이터를 연동할 때 한줄로된 바이트스트림으로 변환하거나 복구하는 작업을 직렬화라고 한다.

**직렬화 구현시 고려할 점

직렬화 구현시 고려할 점

- 이러한 직렬화를 직접 구현할 경우 다양한 상황을 고려해야 함.
 - 데이터 레이아웃 : 오브젝트가 소유한 다양한 데이터를 변환할 것인가?
 - 이식성 : 서로 다른 시스템에 전송해도 이식될 수 있는가?
 - 버전 관리 : 새로운 기능이 추가될 때 이를 어떻게 확장하고 처리할 것인가?
 - 성능 : 네트워크 비용을 줄이기 위해 어떤 데이터 형식을 사용할 것인가?
 - 보안 : 데이터를 어떻게 안전하게 보호할 것인가?
 - 에러 처리 : 전송 과정에서 문제가 발생할 경우 이를 어떻게 인식하고 처리할 것인가?

이러한 상황을 모두 감안해 직렬화 모델을 만드는 것은 쉬운일이 아님

일관성 있게 동작하는

**언리얼 엔진의 직렬화 시스템

언리얼 엔진의 직렬화 시스템

- 언리얼 엔진은 이러한 상황을 모두 고려한 직렬화 시스템을 자체적으로 제공하고 있음
- 직렬화 시스템을 위해 제공하는 클래스 FArchive와 연산자
 - 아카이브 클래스 (FArchive)
 - Shift(<<) operator
- 다양한 아카이브 클래스의 제공
 - 메모리 아카이브 (FMemoryReader, FMemoryWriter)
 - 파일 아카이브 (FArchiveFileReaderGeneric , FArchiveFileWriterGeneric)
 - 기타 언리얼 오브젝트와 관련된 아카이브 클래스(FArchiveUObject)
- Json 직렬화 기능 : 별도의 라이브러리를 통해 제공하고 있음

FArchive라고 하는 클래스입니다

-FArchive 클래스에 쉬프트 연산자를 통해 전송해주면 원하는 형태로 바이트 스트림 저장이 가능하다.

-FArchive를 상속받는 다양한 아카이브 클래스를 제공한다.(사진에서 참고)

****실습**

-Student클래스처럼 언리얼엔진 오브젝트의 기본 설정을 맞춰주면 저장하거나 불러들일 때 즉 직렬화를 진행할 때 시리얼 라이즈 함수를 오버라이드 해서 구현해주면 된다.

-언리얼 오브젝트에서 시리얼라이즈 함수가 구현이 되어 있다.

-파일을 읽고 저장할때 번거로운 delete와 null값을 사용하는데이를 한번에 처리 가능한 스마트포인터를 사용해보자(TUniquePtr)

-직렬화 데이터들은 Saved에 ObjectData.bin, RawData.bin에 저장되어 있다.

****Json 직렬화**

Json 직렬화

- Json(JavaScript Object Notation)의 약자
- 웹 환경에서 서버와 클라이언트 사이에 데이터를 주고받을 때 사용하는 텍스트 기반 데이터 포맷
- Json 장점
 - 텍스트임에도 데이터 크기가 가벼움.
 - 읽기 편해서 데이터를 보고 이해할 수 있음.
 - 사실 상 웹 통신의 표준으로 널리 사용됨.
- Json의 단점
 - 지원하는 타입이 몇 가지 안됨. (문자, 숫자, 불리언, 널, 배열, 오브젝트만 사용 가능)
 - 텍스트 형식으로만 사용할 수 있음.

언리얼 엔진의 Json, JsonUtilities 라이브러리 활용

-우리가 Json을 통해 복잡한 데이터를 전송할때는 지원타입이 별로 없어서 자료형을 구분하기가 불가능하다 따라서 우리가 데이터를 받아 적절히 처리해야 한다.

-텍스트로만 처리가 가능하여 네트워크 상에서 극도의 효율을 추구할때는 단점으로 적용된다.

****Json데이터 예시**

Json 데이터 예시

- Json 데이터 유형
 - 오브젝트 : {}
 - 오브젝트 내 데이터는 키, 밸류 조합으로 구성됨. 예) { "key" : 10 }
 - 배열 : []
 - 배열 내 데이터는 밸류로만 구성됨. 예) ["value1", "value2", "value3"]
 - 이외 데이터
 - 문자열 ("string"), 숫자 (10 또는 3.14), 불리언 (true 또는 false), 널 (null)로 구성

Student 오브젝트

Name : 이득우
Order : 59



```
{  
  "Name" : "이득우",  
  "Order" : 59  
}
```

Json 데이터들은 오브젝트, 배열

-배열은 밸류로만 구성되서 같은 종류의 값들을 쭉 넣어주면 된다.

**엔리얼 스마트 포인터 라이브러리 개요

엔리얼 스마트 포인터 라이브러리 개요

- 일반 C++ 오브젝트의 포인터 문제를 해결해주는 엔리얼 엔진의 라이브러리
- TUniquePtr(유니크포인터) : 지정한 곳에서만 메모리를 관리하는 포인터.
 - 특정 오브젝트에게 명확하게 포인터 해지 권한을 주고 싶은 경우.
 - delete 구문 없이 함수 실행 후 자동으로 소멸시키고 싶을 때
- TSharedPtr(공유포인터) : 더 이상 사용되지 않으면 자동으로 메모리를 해지하는 포인터
 - 여러 로직에서 할당된 오브젝트가 공유해서 사용되는 경우
 - 다른 함수로부터 할당된 오브젝트를 Out으로 받는 경우.
 - Null 일 수 있음.
- TSharedPtrRef(공유레퍼런스) : 공유포인터와 동일하지만, 유효한 객체를 항상 보장받는 레퍼런스
 - 여러 로직에서 할당된 오브젝트가 공유해서 사용되는 경우
 - Not Null을 보장받으며 오브젝트를 편리하게 사용하고 싶은 경우

미리 알아 두면 좋습니다

-TSharedPtr : 더이상 참조되지 않으면(더이상 사용되지 않으면) 자동으로 메모리 해제

-유니크포인터의 경우 어떤 특정 오브젝트에게 해당 오브젝트의 포인터 메모리 해지 권한을 주고싶을 때 사용하면 된다.

-유니크 포인터를 사용해서 해당 동적 할당된 오브젝트를 자동으로 소멸시키고 싶을 때 사용하면 좋다.

-공유 포인터의 경우 여러 로직에서 이 공유포인터가 가리키는 동적 할당된 오브젝트를 사용할 때 그 오브젝트가 더 이상 사용되지 않으면 자동으로 소멸되도록 지정할 때 유용하다.

-공유 포인터는 기본 값이 널값으로 진행을 하는 데 다른 함수로부터 할당된 오브젝트를 Out으로 받고싶은 경우 공유 포인터를 사용해주면 된다.

-공유 레퍼런스는 공유포인터와 다르게 레퍼런스 방식으로 동작하기에 Not Null을 보장받을 수 있어서 널인지 아닌지 점검할 필요가 없다.

-공유 레퍼런스가 사용이 간편하고 편리하다.

**정리

언리얼 오브젝트 직렬화

1. 언리얼 엔진이 제공하는 직렬화 시스템에 대한 이해
2. FArchive 클래스를 활용한 메모리 아카이브와 파일 아카이브의 활용
3. JSonSerializer를 사용한 JSON 형식의 직렬화 기능의 활용
4. 일반 C++ 객체 관리를 위한 언리얼 스마트 포인터 라이브러리의 활용

inflearn

1138208

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

언리얼 오브젝트 직렬화에 대해서 살펴보았는데요