언리얼 프로그래밍 Part1-1

**언리얼 엔진 기본 세팅

- 디버깅을 위한 편집기 기호는 엔진 소스코드 디버깅에 필요한 심볼 기호들이 다 저장이 되어 있다 따라서 엔진 레벨에서 어떻게 언리얼 엔진이 동작하는지 알기 위해서는 체크해야 한다.

**visual studio 세팅

- 게임 회사에서는 보통 유료 플러그인을 사용하지만 강의에서는 따로 사용하지는 않는다.

- 최근 급부상한 플랫폼으로는 라이더를 이용하여 빠르게 개발하기도 하는데 속도가 중요한 경우에 한번 고려해보도록 하자.

- 강의 환경에서는 영어를 주로 사용하니 꼭 참고하자. (한국어 자료가 별로 없기 때문이다.)

- 언리얼 엔진 공식 환경에서는 “C++를 사용한 게임 개발”만 다운로드가 되어있어도 문제가 없다고 한다.

- 밑줄 그은 거 끄자(에러리스트 보여주는거임[원래 기본값이 해제임])

-Solution Configurations 값을 200으로 바꾸자.

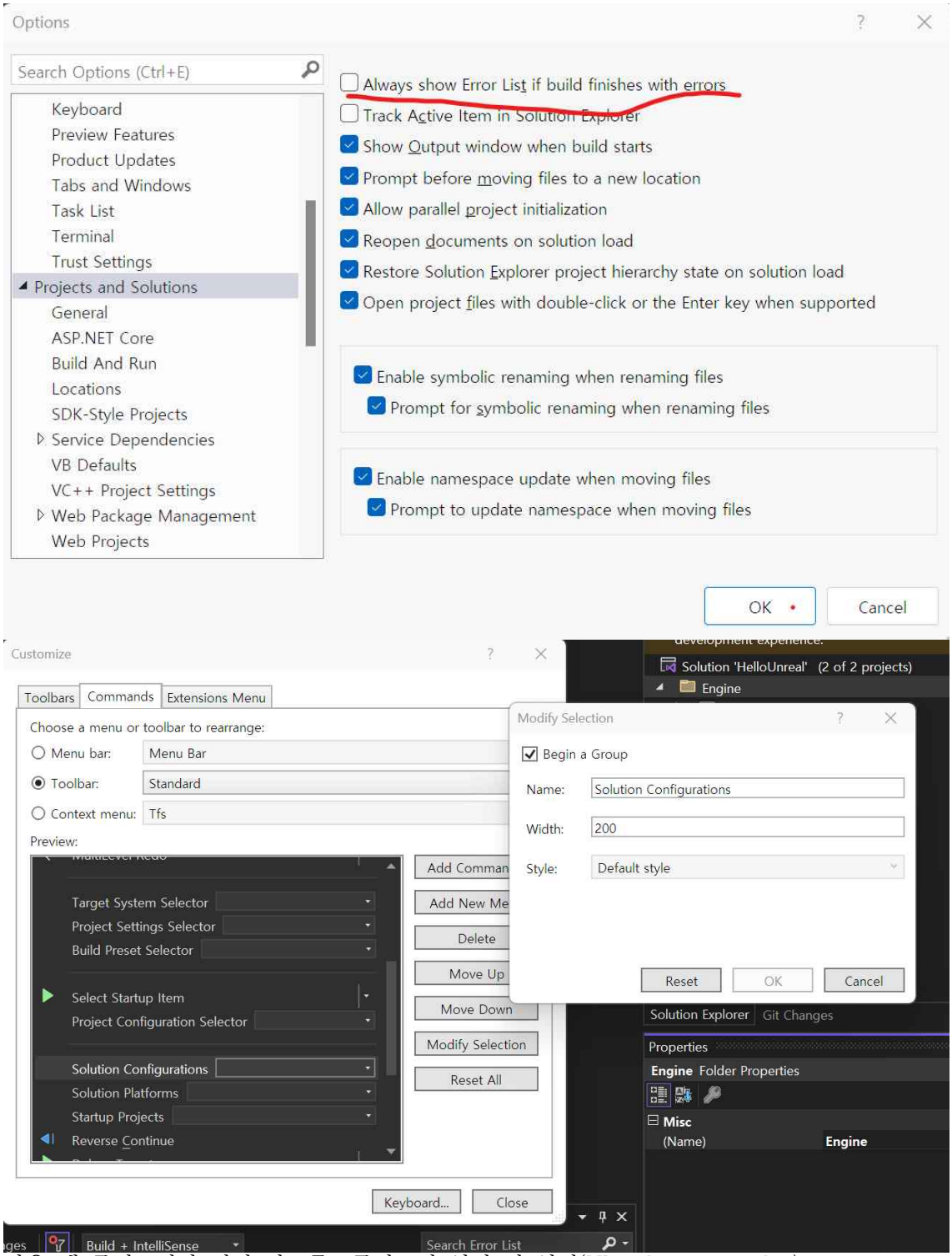
**Unreal Engine(game instance 만들기)

- GameInstance를 상속받아 우리만의 Gameinstance를 만들기

- Tool->new c++ class ->

-에디터가 상속받은 클래스를 파악하여 코딩까지 진행하고. 에디어테어 솔루션 구조를 새로운 구조가 들어갈 수 있게 바꿔준다. (Reload All)

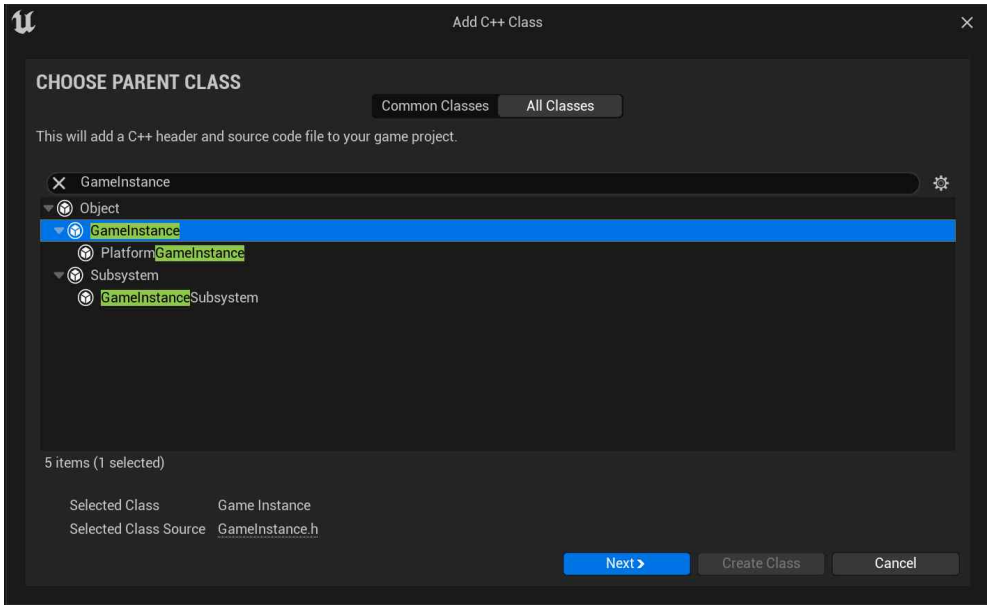
-우리가 언리얼을 사용할 때 매크로를 많이 쓰고 따라서 많은 선언을 하게 됨. 매크로가 들어



갔을 때 들여쓰기가 되지 않도록 플러그인 설치 및 설정(VIsual commander)

**언리얼 코드 컴파일 방법

-헤더 파일에 변경이 발생하면 - 에디터를 끄고 비주얼 스튜디오에서 컴파이 한다.



-소스 파일에만 변경이 발생하면 - 라이브 코딩으로 컴파일 한다.(ctrl +Alt+F11 단축키)

-비주얼 스튜디오에서 수동으로 클래스를 추가하지 말 것!

-언리얼 엔진이 켜져있다면 무조건 라이브 코딩으로만 컴파일 한다.

-라이브 코딩 같은 경우 헤더파일에서 변경이 밭갱하면 잘못 동작할 수 있다. 즉 설계를 바꾼다면 에디터를 끄고 편집하자.

-언리얼 엔진을 하기전에 게임인스턴스를 만들고 초기화를 위해 init()를 호출하는데 이번 강의에서는 init()를 오버라이딩해서 재구성함

-언리얼 엔진이 제공하는 부모클래스를 상속받아서 가상함수를 오버라이드 하여 재구현 하는 경우에는 거의 대부분 언리얼 엔진이 미리 작업해둔 코드를 우리가 실행해줘야 한다.(전체 엔진 플로우가 동작 되게끔 엔지니어들이 만들어 놓았기 때문이다.)[상위 클래스의 가상함수를 실행해야 함을 꼭 잊지 말자.]

-언리얼 엔진에서는 컴파일 하면서 자체적으로 클래스 구조를 파악할 수 있는 프레임웍이 내장 되어 있다.

-부모클래스로부터 상속 받을때는 Super키워드랑 함수 이름을 꼭 써줘야한다. (언리얼 엔진 코딩에서 기본임)

-에픽게임즈 엔지니어들이 구축한 중요한 로직을 다 실행하고 그 다음에 우리가 원하는 로직을 밑에 추가해준다.

-UE_LOG(로그카테고리, 로그 수준
(1) 로그 카테고리: 로그를 분류하기 위함
(2) 로그 수준: 로그 수준을 정하는 것이고 로그, 워닝, 에러 등이 있음. (색상 구분 가능)
(3) printf와 가장 유사하며, 언리얼 엔진에서 string을 다룰땐 TEXT()라는 매크로를 제공함.
-게임 옵션에서 project -Maps&Modes에서 GameInstance는 우리가 컨텐츠를 담는 어플리케이션의 뼈대라고 생각하면 된다. 어플리케이션마다 싱글톤이라고 하는 단일 인스턴스로 관리가 된다. (만약 이컴퓨터에서 이게임을 세 개를 띄우면 게임 인스턴스가 각각 프로그램별로 하나씩 게임 인스턴스로 뜨게 된다.)

-우리가 만든 Init() 함수의 경우 Super키워드를 사용해서 부모 클래스의 모든 로직을 수행하였기 때문에 아무런 문제가 없다.

**정리

헬로 언리얼을 출력하기까지의 여정

1. 언리얼 엔진 설치와 프로그래밍 환경 구축
2. 언리얼 에디터에서의 클래스 추가
3. 언리얼 C++의 클래스 상속 및 오버라이딩 구현
4. 언리얼 엔진의 문자열 처리의 이해
5. 게임인스턴스 클래스의 적용과 카테고리를 활용한 로그 확인