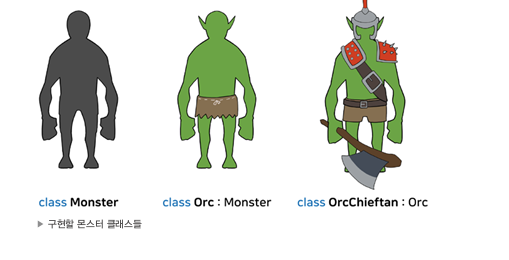
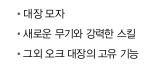
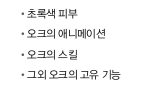
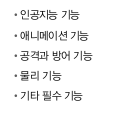
3.1 상속과 재사용

유니티는 c#기반으로 돌아가기 때문에 객체지향을 지원한다.

또한 이미 완성된 게임엔진 코드를 제공하기 때문에 이를 상속받아 재사용하는 것이 생산성을 올리기 좋다.

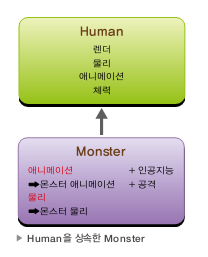
상속을 몬스터 만들기로 예를 들어보자





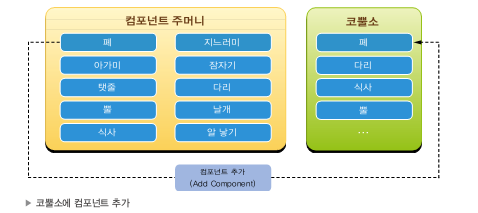
오크대장은 오크를 오크는 몬스터를 상속받도록 되어있다.

하지만 상속도 만능은 아니다. 부모객체가 가지고있는 것을 상속객체에서는 제거해야하는 작업이 생기면 오히려 생산성이 저하 되는 일이 발생하기도 한다.

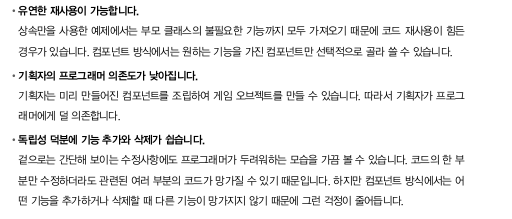


그렇기 때문에 유니티는 상속과 컴포넌트 패턴을 섞어서 사용한다.

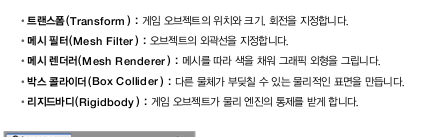
컴포넌트패턴이란 미리 어떤 기능을 구현되있는 조각들을 게임오브젝트라는 틀에다가 삽입시켜서 만들어 기능을 가진 오브젝트로 만들어 내는것이다.\



컴포넌트의 특징



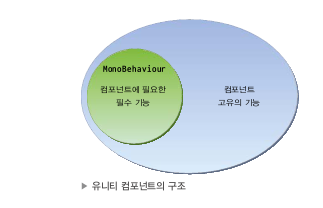
유니티 에디터에서 컴포넌트



이것 이외에도 많지만 대표적으로 자주 쓰이는것들이다.

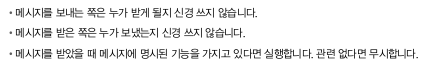
메시지와 브로드 캐스팅

컴포넌트 구조에서는 전체방송을 통해 기능을 간접적 실행한다.(브로드 캐스팅 방식)

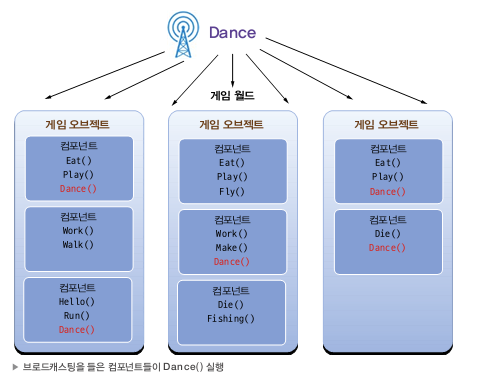
유니티의 모든 컴포넌트는 MonoBehaviour클래스를 상속한다. MonoBehaviour은 유니티에서 제공하는 기본 클래스이며 이 클래스를 기본으로 다른 컴포넌트들이 추가가 되는 것.

또한 MonoBehaviour을 상속받았기 때문에 유니티의 제어를 따르게 된다.

메시지 기반 방식 : 컴포넌트 사이에는 직접적인 연관성이 별로 없어 서로가 찾아가는 방식이 아니라 게임 오브젝트 전부에 메시지를 보내서 응답할 오브젝트를 찾는방식



브로드 캐스팅

그림과 같이 특정 기능을 실행하고 싶다면 그 기능을 메시지로 뿌리면 기능을 가지고있는 오브젝트들은 전부 그기능을 활성화함