Chapter 05: 게임 오브젝트 제어하기

레트로의 유니티 게임프로그래밍 에센스

이 제 민 지음

Contents

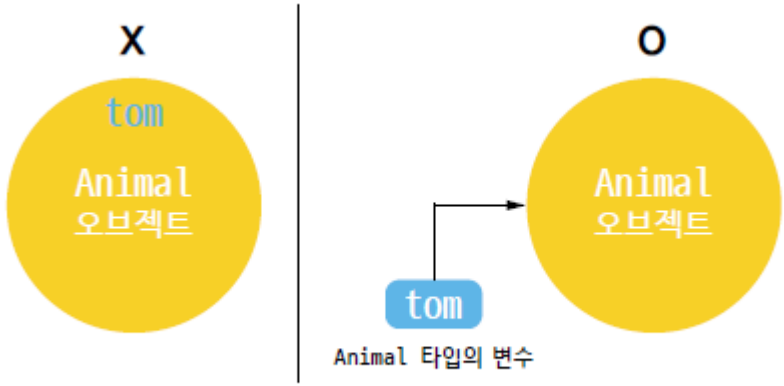
CHAPTER 05 게임 오브젝트 제어하기

5.3 클래스에 대한 추가 설명

- 1. 배열을 왜 동적으로 생성하는지?
- 2. Zoo 에서 Animal Tom = new Animal(); 이렇게 사용하는 것이 상속인가?
- 3. 왜 클래스를 만들때 동적으로 생성을 할까요?
- 4. Animal()이 함수의 형식인가?

5.3 참조 타입

- 참조 타입의 변수는 실체화된 오브젝트가 아님.
- 참조 타입의 변수를 선언하는 것만으로는 오브젝트가 생성되지 않기 때문에 new를 사용해 오브젝트를 개별적으로 생성해야 함.



▶ 변수 tom은 Animal 오브젝트를 가리키는 참조

5.3 참조 타입

◦ 5.3.1 두 마리의 동물 오브젝트

- Zoo 스크립트에서 Animal 타입의 오브젝트를 추가 생성해서 참조 타입이 무엇인지 살펴보겠음.

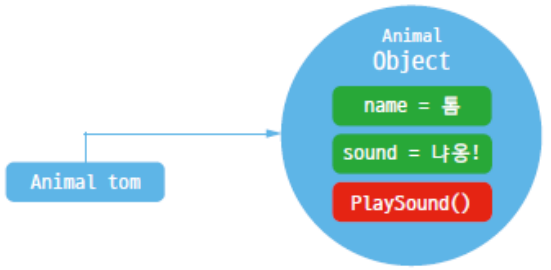
[과정 01] Animal 오브젝트를 두 개 생성하기

① Zoo 스크립트 다시 열기

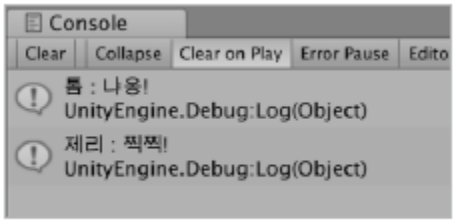
② Zoo 스크립트의 Start() 메서드 부분을 다음과 같이 수정

③ 수정 후 [Ctrl+S]로 스크립트 저장

```
void Start() {
    Animal tom = new Animal();
    tom.name = "톰";
    tom.sound = "냐옹!";

    Animal jerry = new Animal();
    jerry.name = "제리";
    jerry.sound = "찍찍!";

    tom.PlaySound();
    jerry.PlaySound();
}
```



▶ 메모리 상태



▶ 수정된 Zoo 스크립트의 실행 결과

5.3 참조 타입

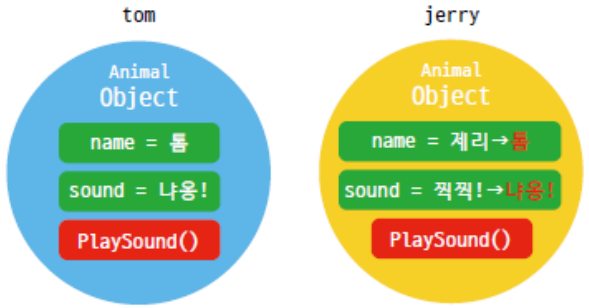
- 5.3.2 참조값 변경하기
- - 변수 jerry에 변수 tom의 값을 대입하면 어떤 일이 일어나는지 볼 것임.

[과정 01] jerry에 tom 대입하기

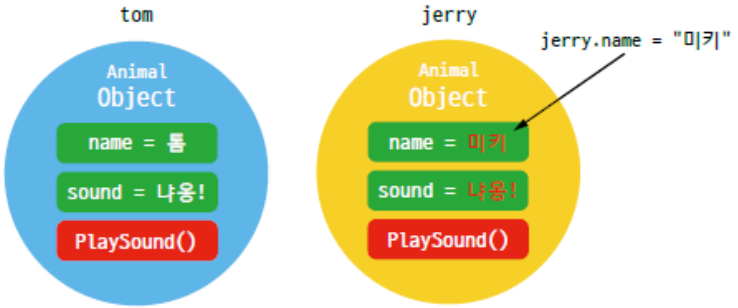
① Zoo 클래스의 Start() 메서드 부분을 다음과 같이 수정

② [Ctrl+S]로 수정된 스크립트 저장

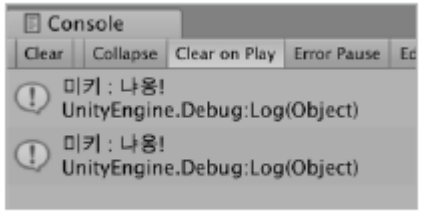
```
void Start() {
    Animal tom = new Animal();
    tom.name = "톰";
    tom.sound = "냐옹!";

    Animal jerry = new Animal();
    jerry.name = "제리";
    jerry.sound = "찍찍!";

    jerry = tom;
    jerry.name = "미키";

    tom.PlaySound();
    jerry.PlaySound();
}
```



▶ jerry = tom의 결과



▶ jerry.name = "미키"의 결과

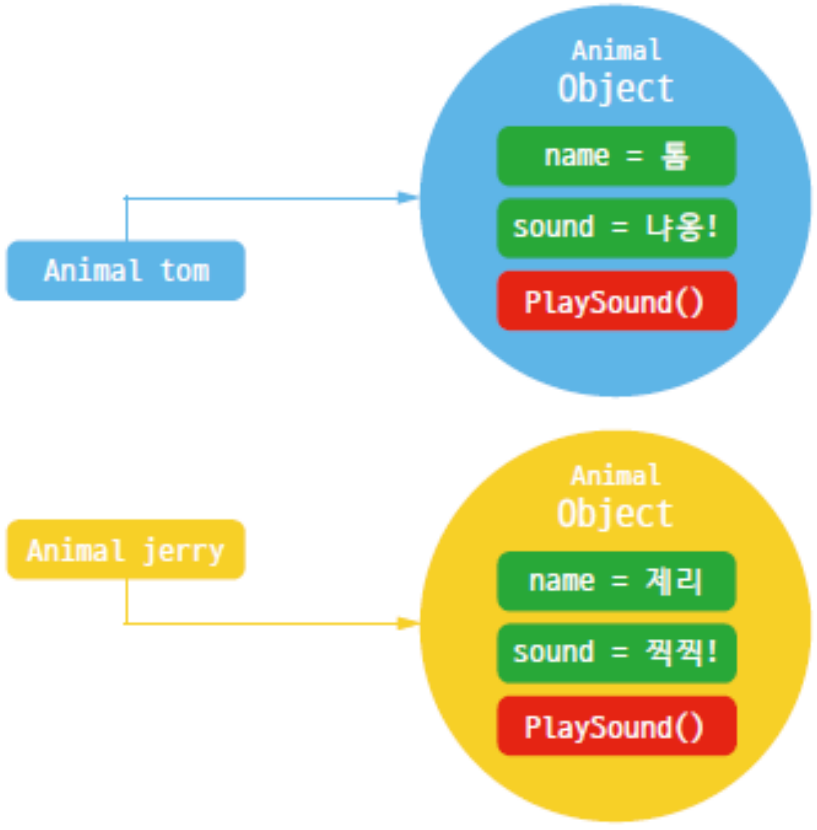


▶ 수정된 Zoo 스크립트의 실행 결과

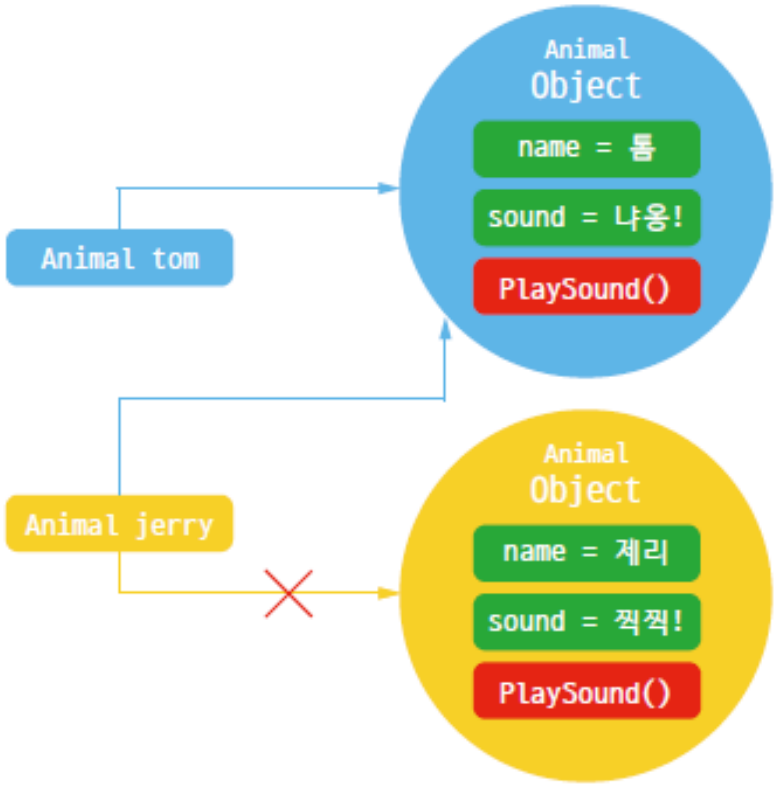
5.3 참조 타입

◦ 5.3.3 참조 타입의 동작

- jerry = tom;이 실행되기 직전의 메모리 상태를 표현하면 다음과 같음.



▶ tom과 jerry의 상태

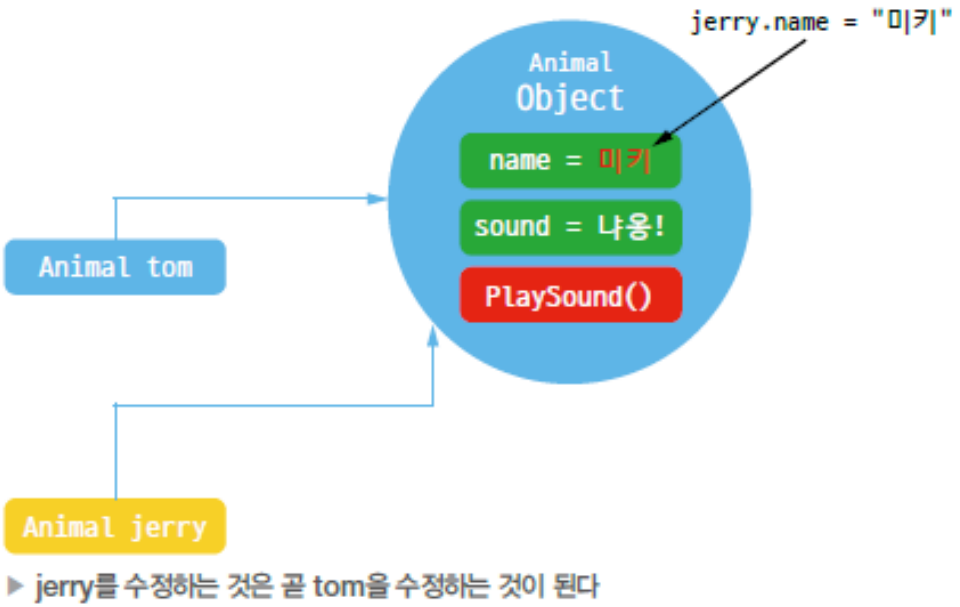


▶ tom과 jerry가 같은 오브젝트를 가리킨다

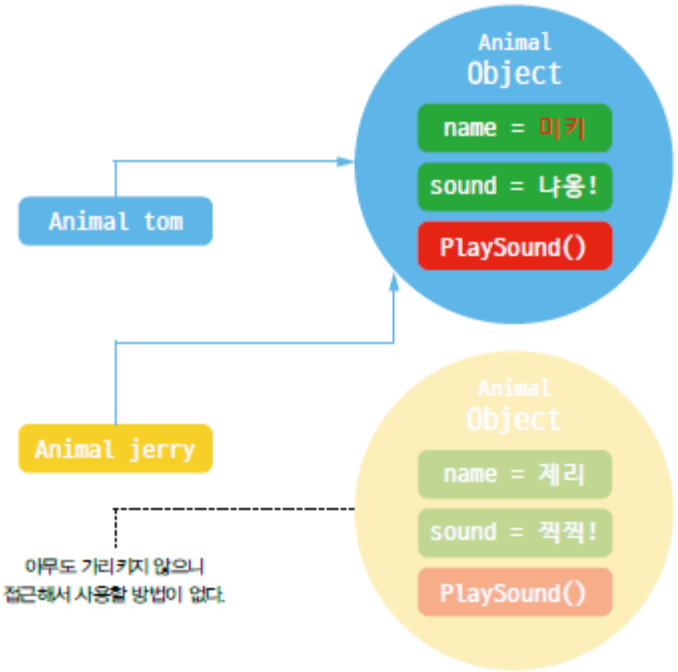
5.3 참조 타입

◦ 5.3.3 참조 타입의 동작

- jerry = tom;이 실행되기 직전의 메모리 상태를 표현하면 다음과 같음.



▶ jerry를 수정하는 것은 곧 tom을 수정하는 것이 된다



▶ 미아가 된 오브젝트는 사용할 수 없다

5.3 참조 타입

- 5.3.4 하나의 실체와 여러 개의 참조 변수
  - 변수에 실체가 아니라 실체로 향하는 참조가 할당되고, 변수에 접근하면 참조를 통해 실체에 접근하는 변수를 참조 타입이라고 함.
  - 참조 타입은 '한 사람을 여러 개의 별명으로 부르는' 상황을 만들 수 있음.
  - 한 사람을 다양한 별명으로 부를 수 있지만, 그 모든 별명이 가리키는 실체는 하나임.
  - 즉, 오브젝트는 하나지만 그것을 여러 개의 참조 변수가 동시에 가리킬 수 있음.

5.3 참조 타입

◦ 5.3.5 값 타입과 참조 타입

- 모든 변수가 참조로 동작하는 것은 아님.
- float, int, string 등의 C# 내장변수는 참조로 동작하지 않음. 이런 타입을 값 타입이라고 함.
- 참조 타입 변수는 값(실체)으로 향하는 참조를 저장하고, 값 타입의 변수는 해당 변수 공간에 값 자체를 저장함.

▶ 값 타입과 참조 타입의 비교

▶ a = b에서 값 타입과 참조 타입의 동작 차이

5.3 참조 타입

참조 타입	값 타입
class 타입 유니티의 모든 컴포넌트 우리가 작성할 C# 스크립트 (MonoBehaviour를 상속받는 클래스)	C# 내장 변수 • bool • int • float • char • double • string (immutable로 선언된 class) • ... struct(구조체) 타입 • Vector3 • Color • ...

5.4 변수로 컴포넌트 사용하기

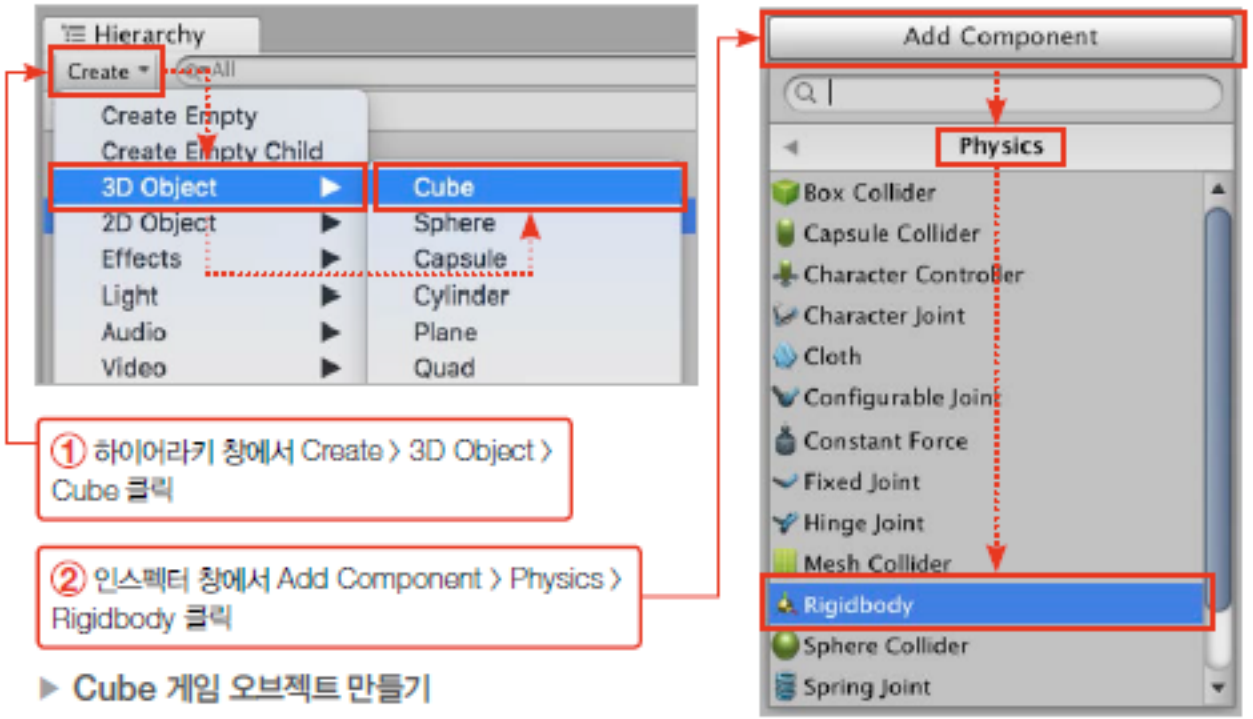
◦ 5.4.1 물리 큐브 만들기

- 변수로 게임 오브젝트를 직접 제어해 봄.

[과정 01] Cube 게임 오브젝트 만들기

①큐브 Cube 게임 오브젝트 생성(하이어라키 창에서 Create > 3D Object > Cube 클릭)

②인스펙터 창에서 Add Component > Physics > Rigidbody 클릭



▶ Cube 게임 오브젝트 만들기

5.4 변수로 컴포넌트 사용하기

◦ 5.4.2 변수로 리지드바디 컴포넌트 사용하기

- Rigidbody 타입의 변수는 Rigidbody 오브젝트 그 자체는 아니지만 실제 Rigidbody 오브젝트(리지드바디 컴포넌트)를 가리킬 수 있음. 따라서 해당 변수로 실제 리지드바디 컴포넌트에 접근해 조종할 수 있음.

[과정 01] Jumper 스크립트 만들기

① **프로젝트** 창에서 Create 〉 **C# Script** 클릭

② 생성된 스크립트 이름을 **Jumper**로 변경 〉 스크립트를 **더블 클릭**으로 열기

[과정 02] Jumper 스크립트 완성하기

① Jumper 스크립트의 전체 코드를 다음과 같이 수정

② **[Ctrl+S]**로 작성한 스크립트 저장

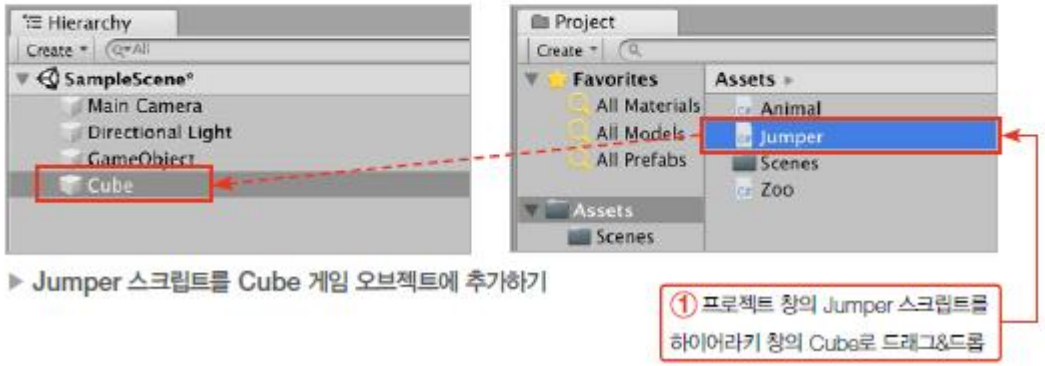
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Jumper : MonoBehaviour {
    public Rigidbody myRigidbody;

    void Start() {
        myRigidbody.AddForce(0, 500, 0);
    }
}
```

5.4 변수로 컴포넌트 사용하기

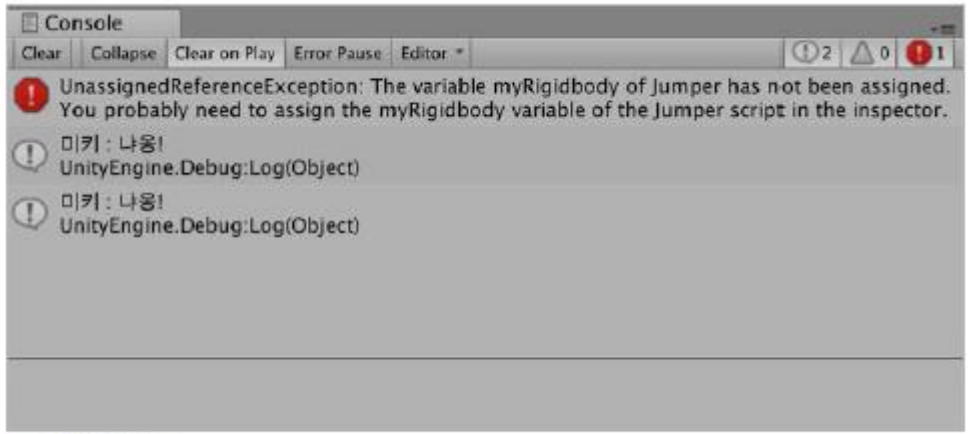
[과정 03] Jumper 스크립트를 Cube 게임 오브젝트에 추가하기

① **프로젝트** 창의 **Jumper** 스크립트를 **하이어라키** 창의 **Cube**로 **드래그&드롭**

→ **Jumper** 스크립트가 **Cube** 게임 오브젝트에 추가됨



▶ Jumper 스크립트를 Cube 게임 오브젝트에 추가하기

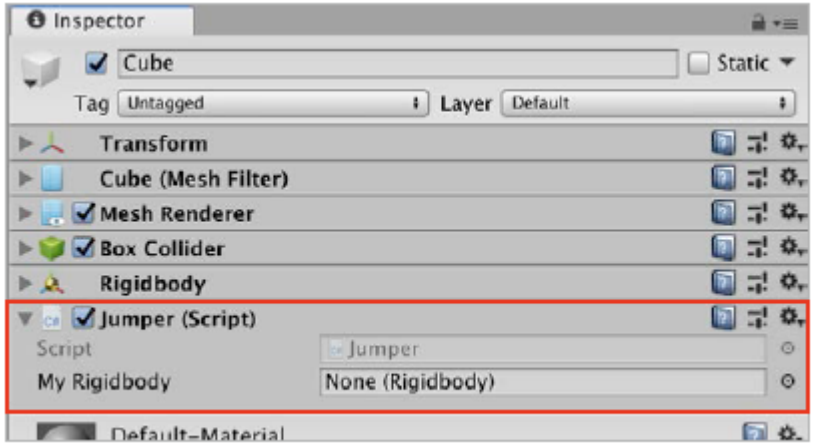


▶ 출력된 에러 로그

5.4 변수로 컴포넌트 사용하기

- 5.4.3 컴포넌트를 변수에 연결하기

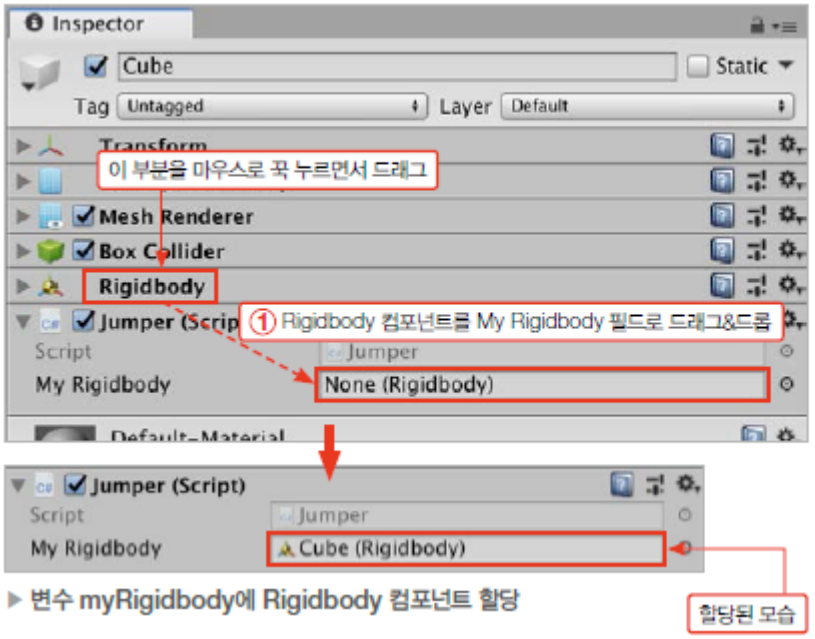
  - 인스펙터 창에서 Cube 게임 오브젝트의 Jumper 컴포넌트(스크립트)를 살펴봄.



▶ Jumper 스크립트의 My Rigidbody 필드

[과정 01] 변수 myRigidbody에 리지드바디 컴포넌트 할당하기

① Cube 게임 오브젝트의 Rigidbody 컴포넌트를 Jumper 컴포넌트의 My Rigidbody 필드로 드래그&드롭

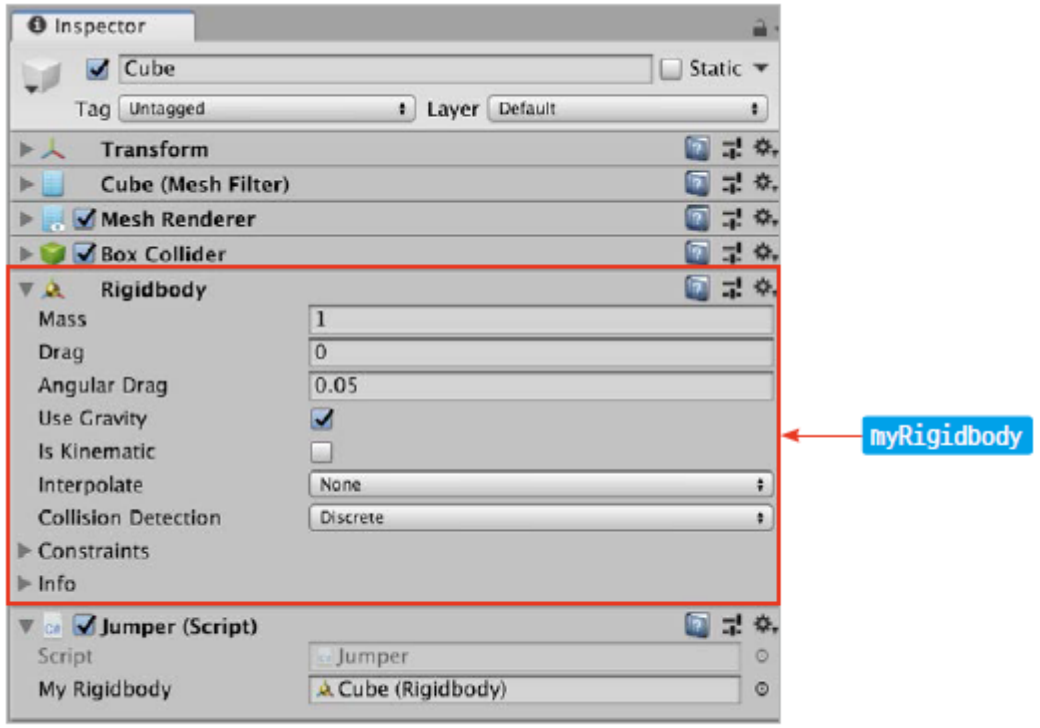


▶ 변수 myRigidbody에 Rigidbody 컴포넌트 할당

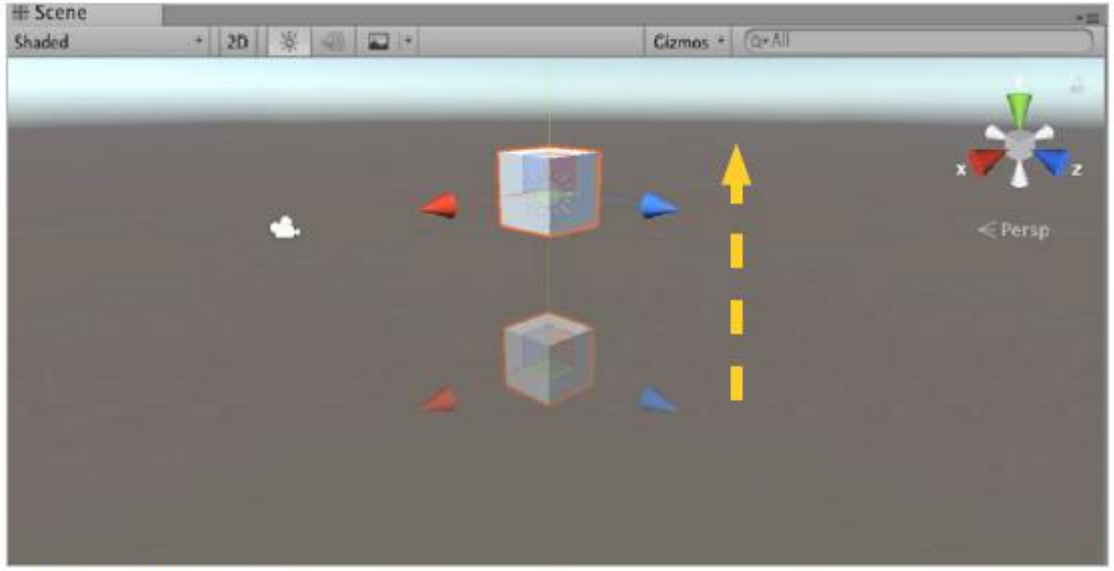
드래그하는 방법

인스펙터 창에서 Rigidbody 컴포넌트의 이름 위에 마우스 커서를 두고 마우스 왼쪽 버튼을 누른 상태에서 My Rigidbody 필드로 드래그

5.4 변수로 컴포넌트 사용하기



▶ 변수 myRigidbody는 Cube 게임 오브젝트의 리지드바디 컴포넌트를 가리킨다



▶ 위쪽으로 점프하는 큐브

5.5 마치며

- 이 장에서 배운 내용 요약

- 클래스는 묘사하려는 사물(오브젝트)을 정의하는 틀임.
- 클래스를 통해 대상에 관한 변수와 메서드를 정의함.
- new 연산자로 클래스로부터 오브젝트를 생성할 수 있음.
- 접근 제한자를 사용해 클래스의 멤버를 선택적으로 외부에 공개할 수 있음.
- C# 기본 내장 변수는 값 타입임.
- 값 타입의 변수는 변수가 값 자체를 저장함
- 클래스 타입의 변수는 참조 타입임.
- 참조 타입의 변수는 오브젝트로 향하는 참조를 저장함.
- 참조 타입에서는 여러 개의 변수가 하나의 오브젝트를 가리키는 것이 가능함.
- 컴포넌트는 클래스로 만들어져 있음.
- 씬에 있는 게임 오브젝트나 컴포넌트의 참조를 변수에 할당하면 코드 상에서 변수로 해당 게임 오브젝트와 컴포넌트를 사용할 수 있음.