

Unity3D Basic

Unity GameObject

목차

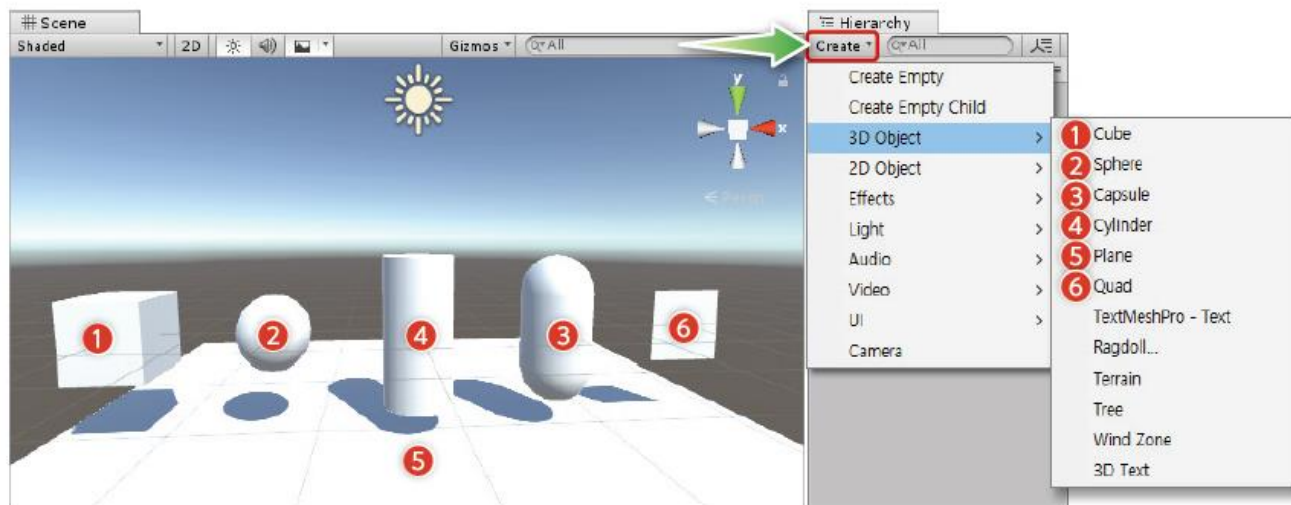
1. Unity GameObject

- Game Object
- Transform

유니티 기본 오브젝트

- 오브젝트 추가

- 게임 오브젝트 : 게임에 필요한 모든 요소 (캐릭터, 총, 조명 등)
- 게임의 일정 부분은 유니티가 제공하는 **기본 오브젝트** (Primitive Object) 를 이용하여 만들 수 있음
- 하이라키 [Create - 3D Object] 또는 메뉴 [GameObject - 3D Object] 클릭하여 해당 오브젝트를 씬에 추가



- Cube (정육면체), Sphere (구), Capsule (캡슐), Cylinder (원기둥), Plane (눌혀진 평면), Quad (세워진 평면)



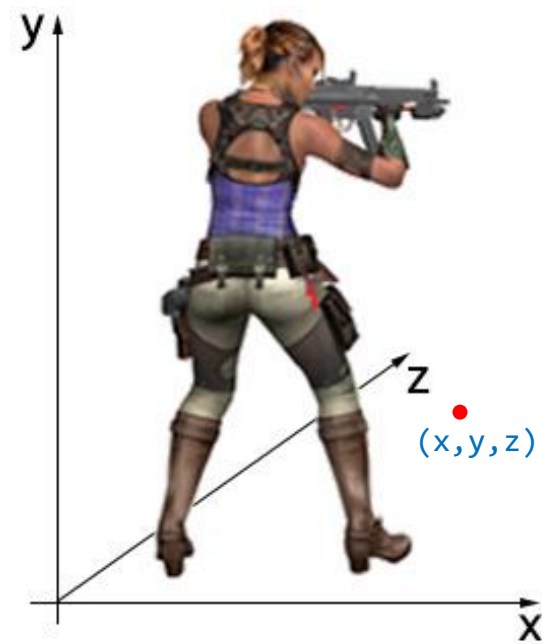
유닛(Unit)

- 3D 그래픽스 오브젝트는 벡터 그래픽이므로, 확대하더라도 경계면이 거칠어지지 않음. 오브젝트의 스케일을 임의로 설정할 수 있기 때문에, 오브젝트의 절대적인 크기는 중요하지 않음
- 오브젝트 크기를 설정하는 유닛이라는 가상 단위를 사용 (유닛 단위는 작업 환경에 맞게 임의로 설정해서 사용)
- 유니티는 유닛 단위를 m로 설정(3DS Max, Maya 등은 유닛을 cm로 설정)

- 유니티 좌표계
 - 공간상에 있는 오브젝트의 위치를 정하는 방법



평면벡터 Vector2(x,y)

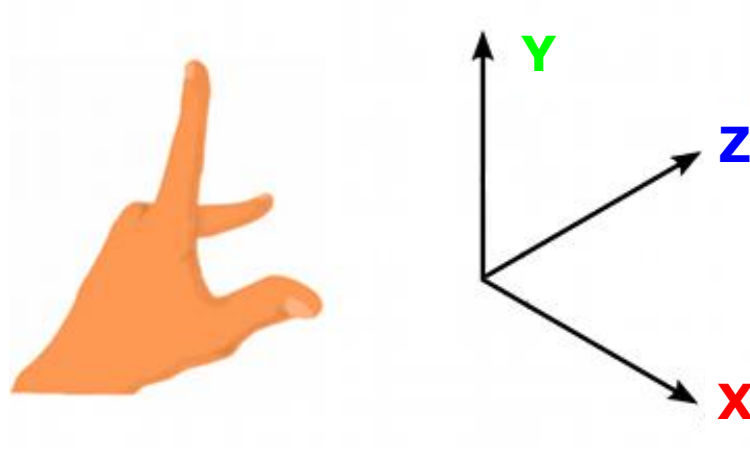


공간벡터 Vector3(x,y,z)



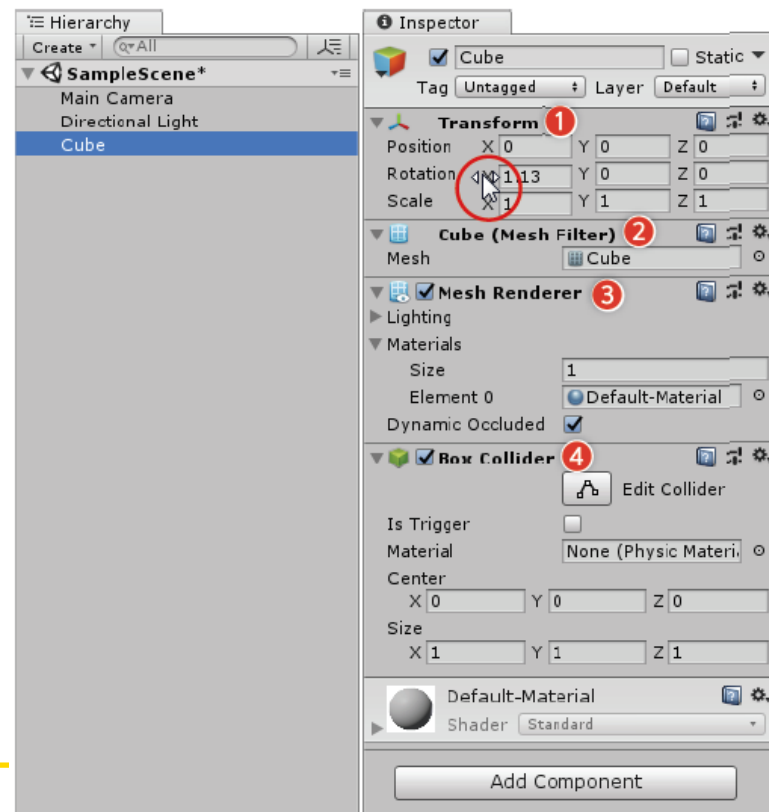
- 유니티는 왼손 좌표계를 사용 (Z축이 전진 방향)

RGB 로 표시 : X 빨강, Y 초록, Z 파랑



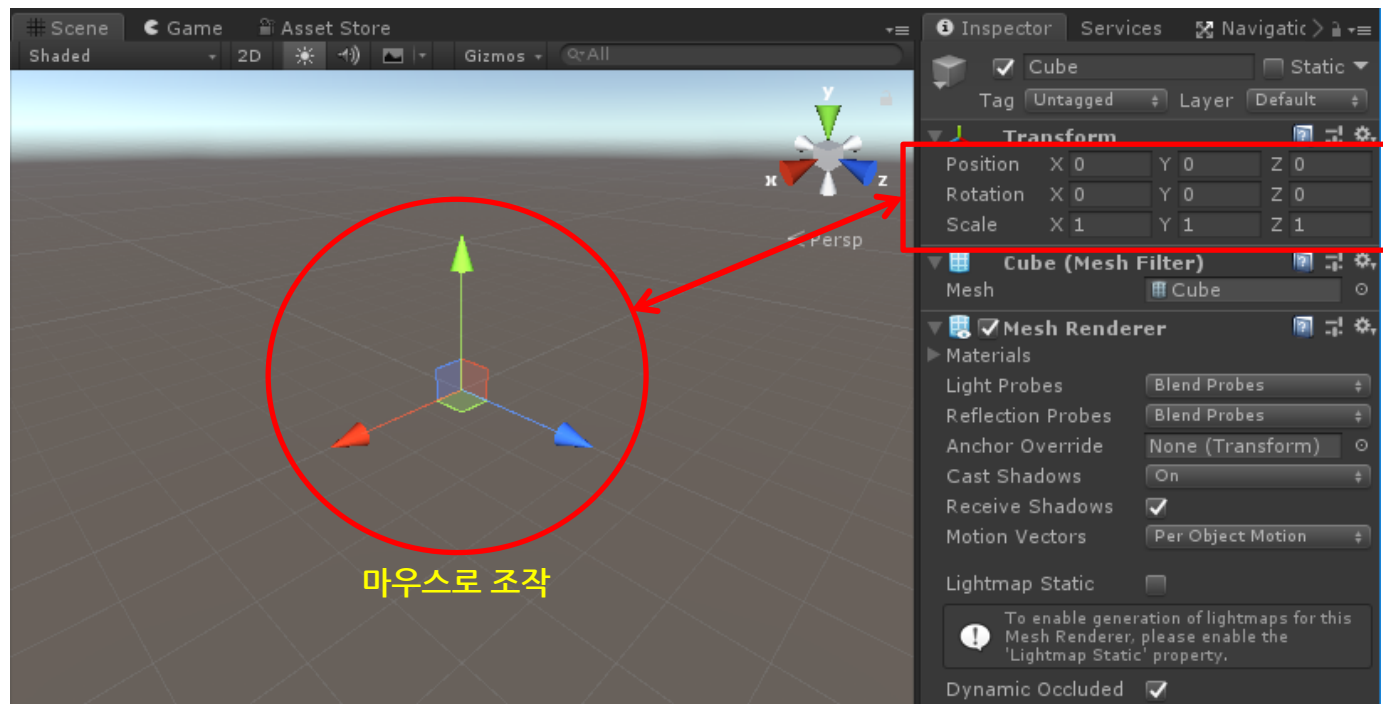
- 기본 컴포넌트

- 오브젝트를 씬에 추가하면, 기본적인 컴포넌트가 생성됨
- 컴포넌트 속성은 인스펙터(Inspector)에서 수정
(볼드체로 표시된 것이 컴포넌트)



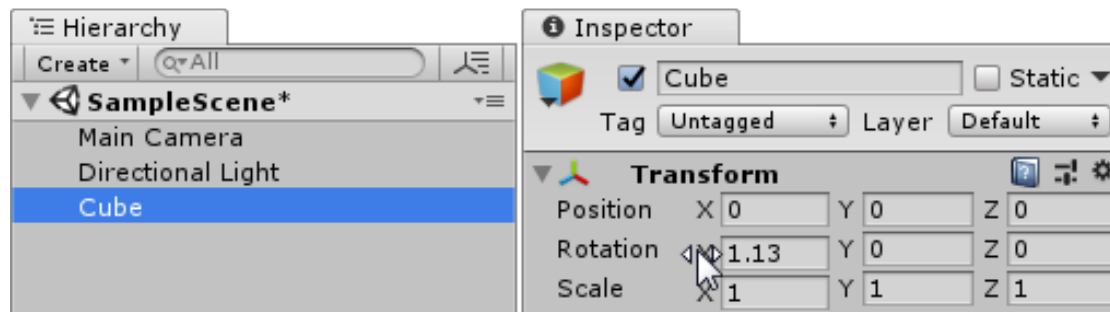
– 컴포넌트 속성 변경

- 인스펙터에서 컴포넌트의 속성을 변경하면 씬 뷰에 즉각적으로 반영됨
- 씬 뷰에서 오브젝트에 대한 이동/회전/스케일 등을 조작하면, 곧바로 인스펙터의 속성에 결과가 반영됨




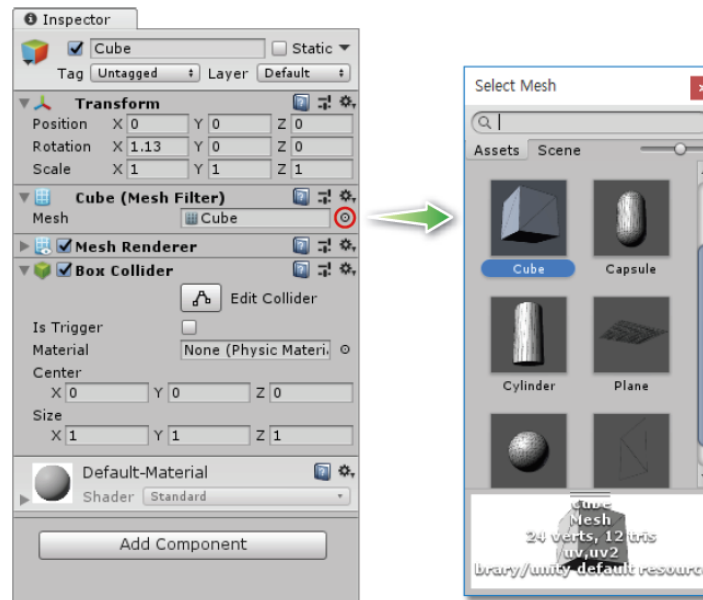
– Transform 컴포넌트

- 오브젝트의 위치, 방향, 크기를 나타내는 기본 컴포넌트
- 씬에 설치된 모든 오브젝트가 보유
 - Position : 기준점으로 부터의 거리
 - Rotation : 각 축에 대한 회전각 (60분법). 시계방향이 (+) 방향
 - Scale : 각 축에 대한 확대/축소 비율
- 게임 화면의 원점이 기준이며, 오브젝트가 계층적으로 구성된 경우는 부모로부터의 상대적인 값을 표시
- 오브젝트의 회전은 시계 방향이 (+) 방향
- 게임 화면은 오브젝트의 뒷면이 표시되므로, z축은 시계 반대 방향이 (+) 방향



– Mesh Filter (메쉬 필터)

- 3D 오브젝트에만 존재하며, 오브젝트의 메쉬(Mesh) 데이터를 보여줌
- Mesh를 수정하려면, 속성 오른쪽 끝에 있는 [] 버튼을 클릭



– Mesh Renderer (메쉬 렌더러)

- 3D 오브젝트에만 존재하며, 오브젝트의 질감(Material) 데이터를 보여줌



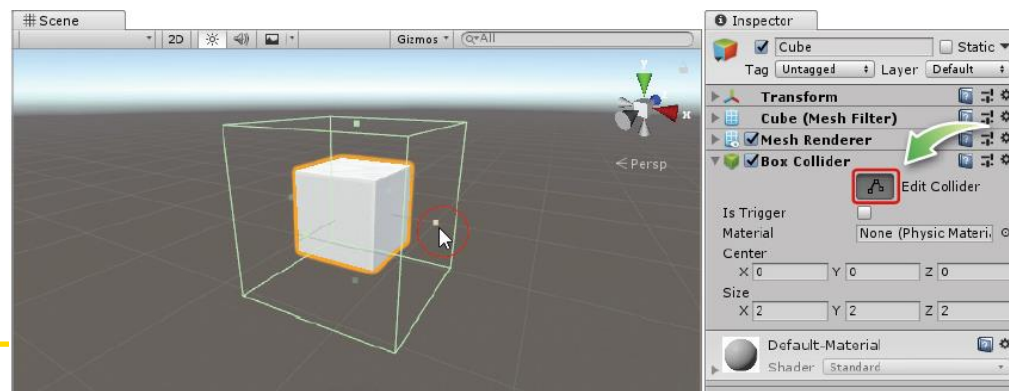
- Collider (콜라이더)

- 오브젝트의 충돌을 판정하기 위한 영역
- 두 물체의 콜라이더가 서로 접촉하면 충돌 이벤트가 발생

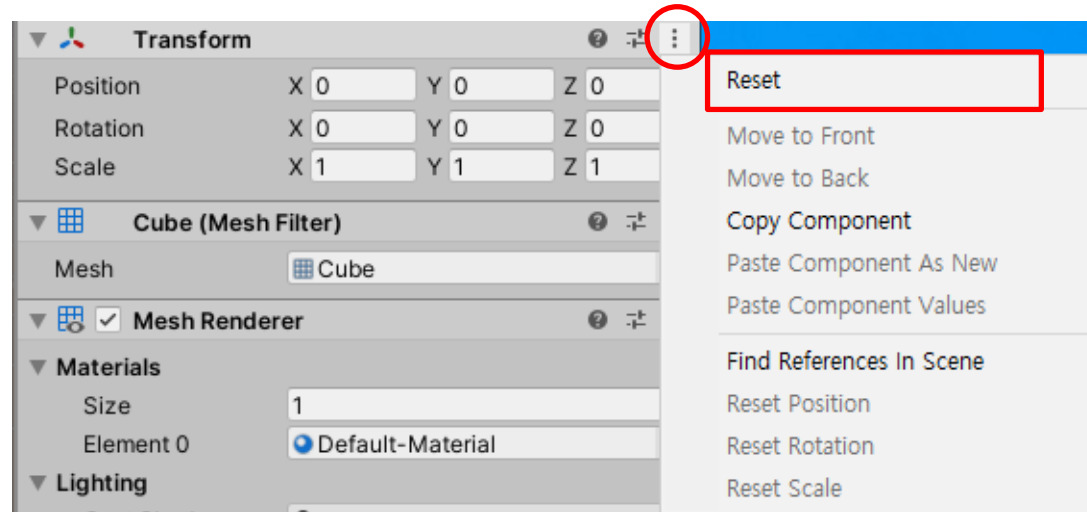
→ 충돌에 반응하려는 오브젝트는 반드시 콜라이더가 있어야 함

콜라이더가 없는 오브젝트는 충돌이 발생하지 않으므로, 물체가 이동할 때 다른 오브젝트를 뚫고 지나감

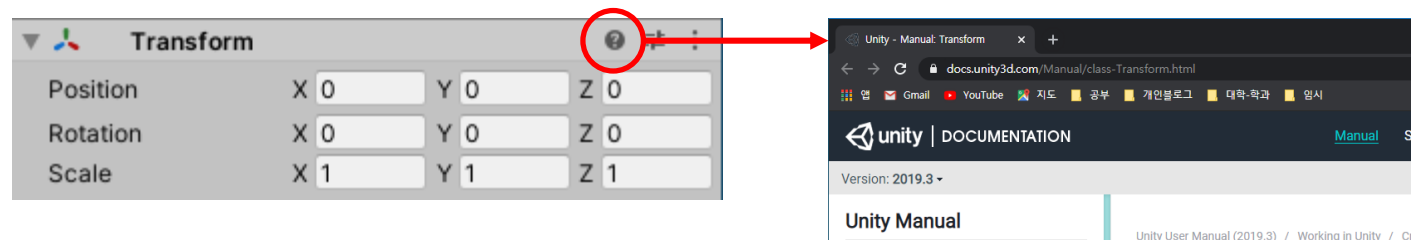
- 콜라이더는 오브젝트 형태에 따라 Box, Sphere, Capsule, Mesh, Wheel, Terrain 등
- 하나의 오브젝트에 여러 개의 콜라이더를 부여가 가능
- [Component - Physics] 메뉴에서 추가
- 콜라이더의 크기, 위치를 수정하려면 선택한 오브젝트에 씬뷰에 콜라이더가 연두색으로 표시됨.
[Edit Collider] 버튼을 클릭하여 콜라이더의 조정점을 표시하여 수정 (드래그)



- 컴포넌트 오른쪽 점 3개의 추가 메뉴 버튼 - 컴포넌트의 부메뉴를 호출
 - (예) Reset 시키면, Position, Rotation, Scale 가 기본값으로 초기화



- 컴포넌트 오른쪽 도움말 버튼 클릭 (유니티 다큐먼트 사이트로 연결)



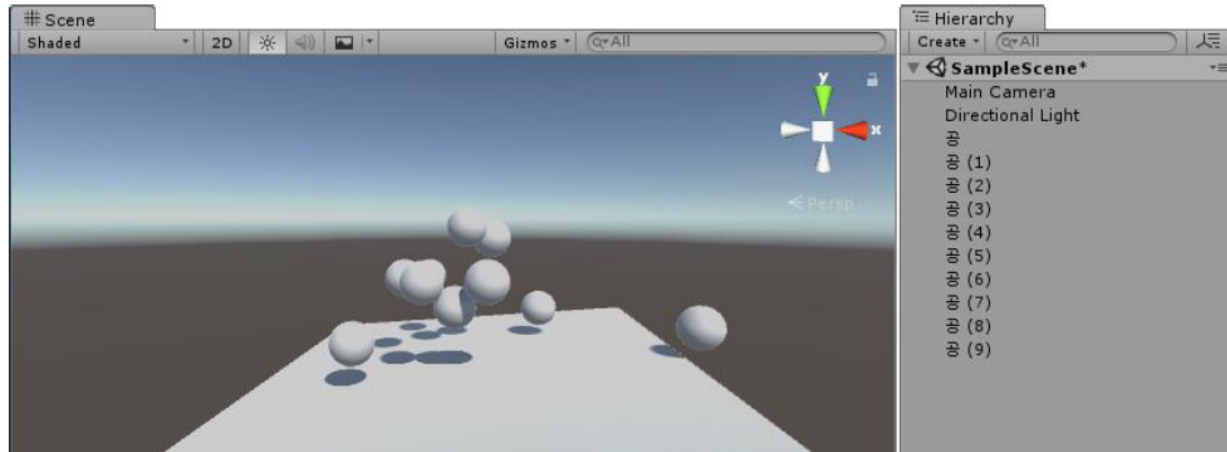
– Rigidbody (리지드바디)

- 물리엔진을 컴포넌트로 만든 것
- 중력, 마찰, 충돌의 판정 등에 관여
- 물체가 충돌 반응을 일으키려면, 두 물체 중에 적어도 어느 하나는 리지드바디가 있어야 하며, 충돌 이벤트는 리지드바디가 있는 오브젝트에만 발생

실습



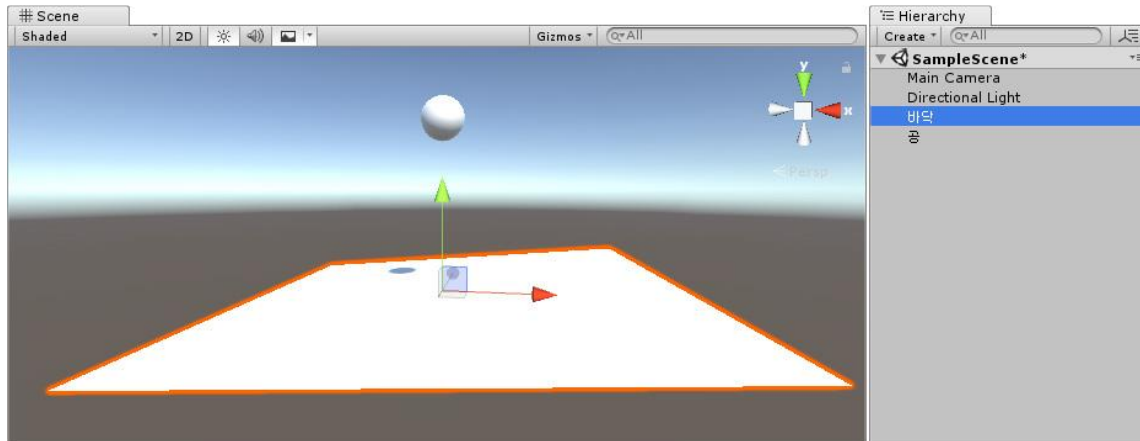
1. Sphere에 리지드바드 추가
2. Sphere 와 Plane 오브젝트에 텍스처를 입힘 (매핑)
3. 오브젝트 반사(반발력 부여) - Physics Material을 생성하고 매핑
4. (반발력에 의해) Sphere가 띄어 오르게 되며, 이때 사운드를 추가하여 효과음을 부여
오디오 클립을 오브젝트에 드래그하여 AudioSource를 추가
충돌 판정 스크립트를 작성 (다음 슬라이드 참조)
스크립트를 Sphere에 드래그
5. Sphere를 복제(^D)하여 여러 개를 배치하고 게임을 실행





[실습] New Project 생성, 씬에 Plane와 Sphere 오브젝트 추가

- 오브젝트 이름은 한글 사용이 가능



- 오브젝트 속성

Object	Name	Position	Rotation	Scale
Plane	바닥	0,0,0	0,0,5	1,1,1
Sphere	공	0,4,0	0,0,0	1,1,1



게임 실행

- 바닥과 공에 모두 물리적 영향을 받지 않는 무중력 상태이므로, 아무런 변화가 없음

- 공에 리지드바디를 추가

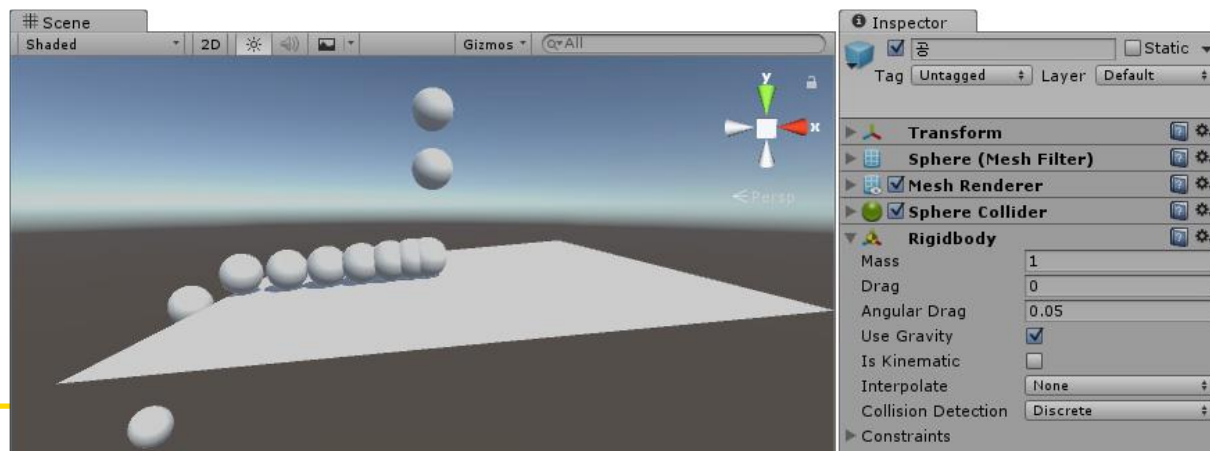
- 공을 선택한 후, [Component - Physics - Rigidbody] 메뉴 클릭
- (또는) 컴포넌트를 추가하는 다른 방법
 - 인스펙터의 [Add Component] 버튼 클릭하고 검색어를 입력하여 해당 컴포넌트를 찾아서 추가



공에 리지드바디를 추가한 후, 다시 게임을 실행

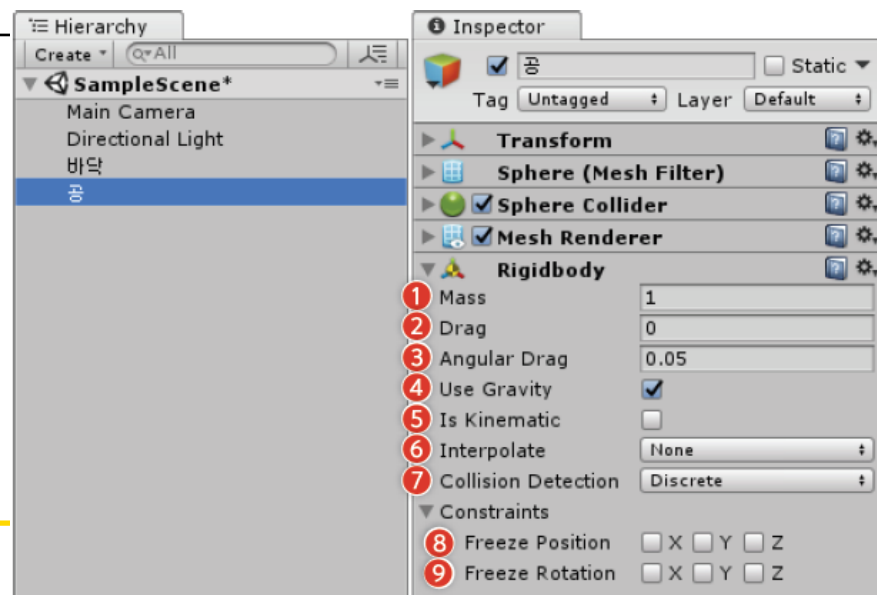
- 중력에 영향을 받아 공이 바닥으로 떨어진다
- 바닥을 회전시켜서 기울인 다음, 다시 실행해 본다. 어떤 현상이 발생 ?

공에 중력가속도가 적용되어 시간의 흐름에 따라 공이 이동하는 거리가 달라짐



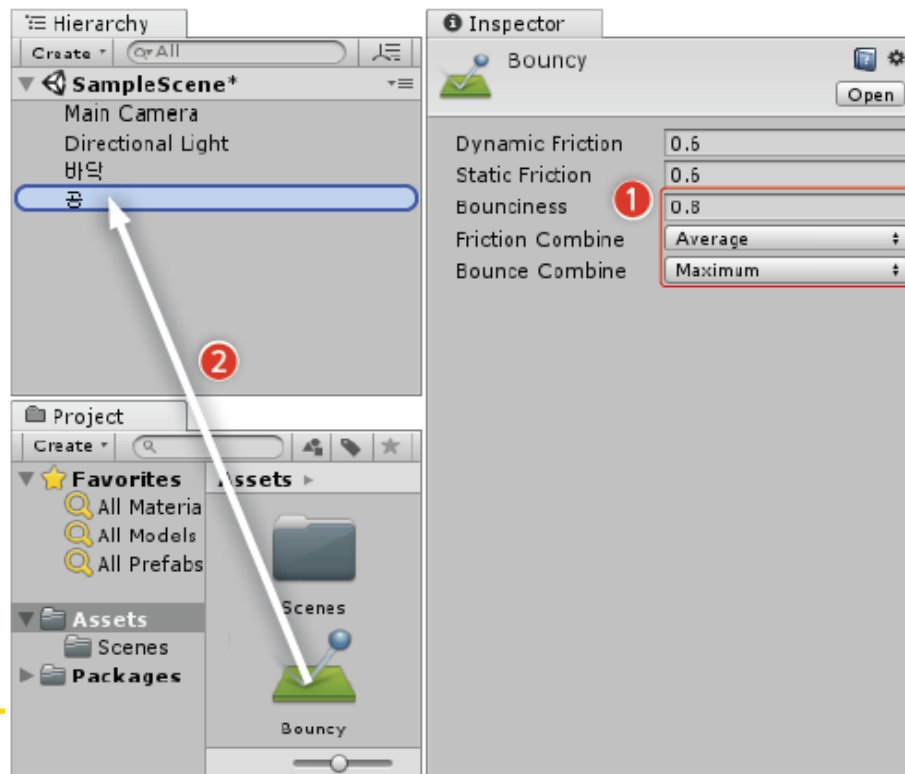
– 리지드바디 속성

Mass	오브젝트의 질량(무게) [단위:kg]
Drag	오브젝트의 운동 저항 (값이 0이면 영원히 날아감)
Angular Drag	오브젝트의 굴림 저항 (값이 0이면 영원히 굴러감)
Use Gravity	중력 적용 여부 - 발사한 물체에 중력을 적용하면 포물선 형태로, 적용하지 않으면 직선 형태로 날아감
Is Kinematic	옵션을 설정하면 충돌 판정만 하고 물리 처리는 하지 않음 (중력도 무시됨) (예) UI 버튼 등은 터치 이벤트만 필요하고 물리 처리는 필요하지 않음
Interpolate	이전 프레임을 참조(interpolate)하거나 다음 프레임을 추정(extrapolate) 해서 오브젝트의 움직임을 부드럽게 처리
Collision Detection	충돌 판정 방법 - 총알과 같이 작은 물체가 빠르게 이동할 때, 충돌 검사가 제대로 이뤄지지 않으면 이 옵션을 변경
Freeze Position	충돌이 발생할 때, 이동(팅김)을 방지하는 축
Freeze Rotation	충돌이 발생할 때, 회전(스핀)을 방지하는 축



- 매터리얼 속성

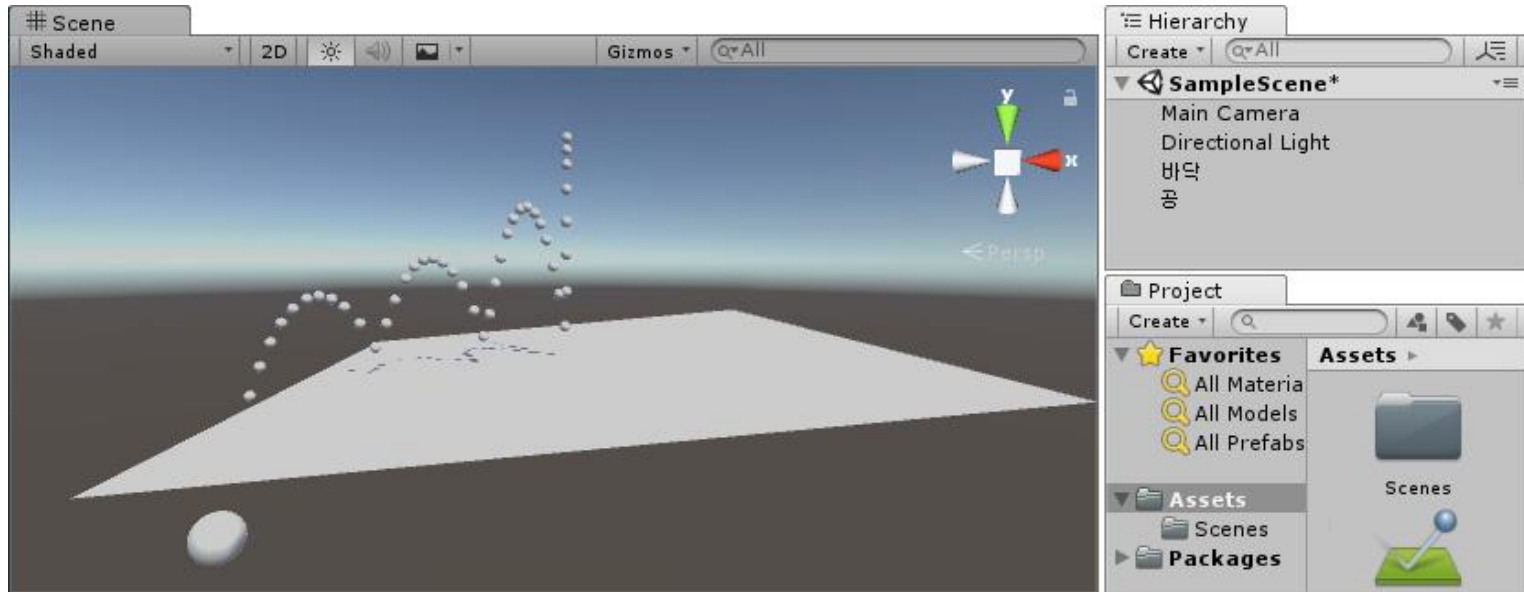
- 물리 매터리얼에 적당한 이름(Bouncy)을 부여하고 속성을 설정
- Bounciness와 Bounce Combine을 설정
마찰력(Friction), 반발력(Bounciness)
- 속성 설정 이후, Bouncy를 하이라키의 공으로 드래그하여 반발력을 적용




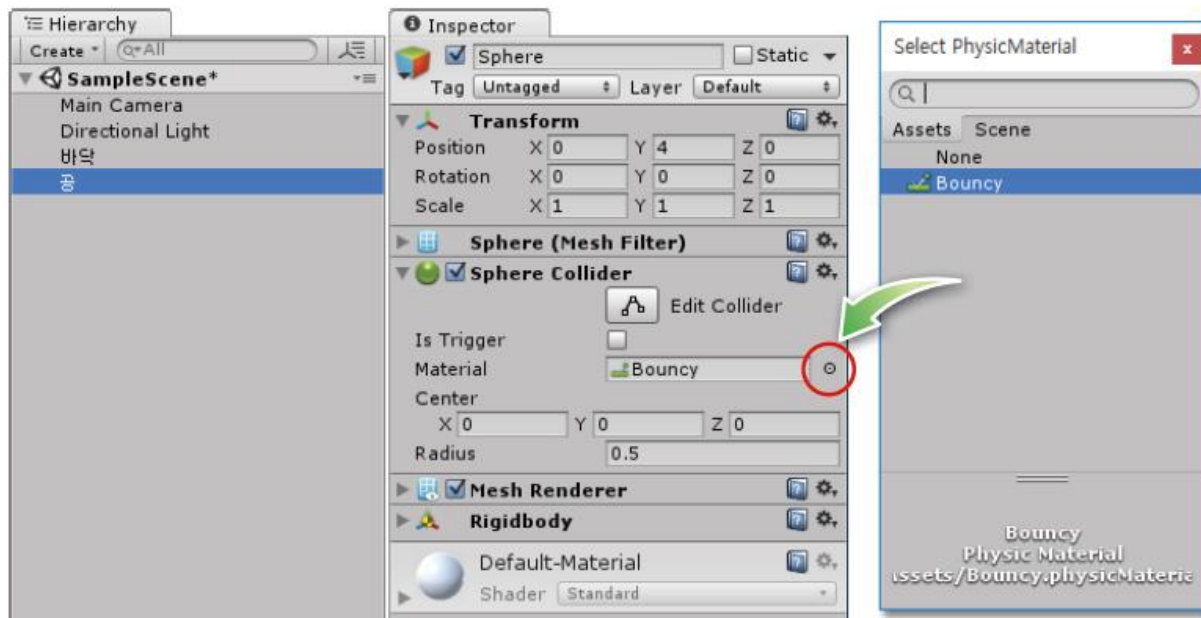


게임 실행

- 공이 바닥과 충돌한 후 튀어 오르는 것을 확인
- Bouncy의 Bounciness를 0~1까지 변화시켜가면서 게임을 실행해 보세요

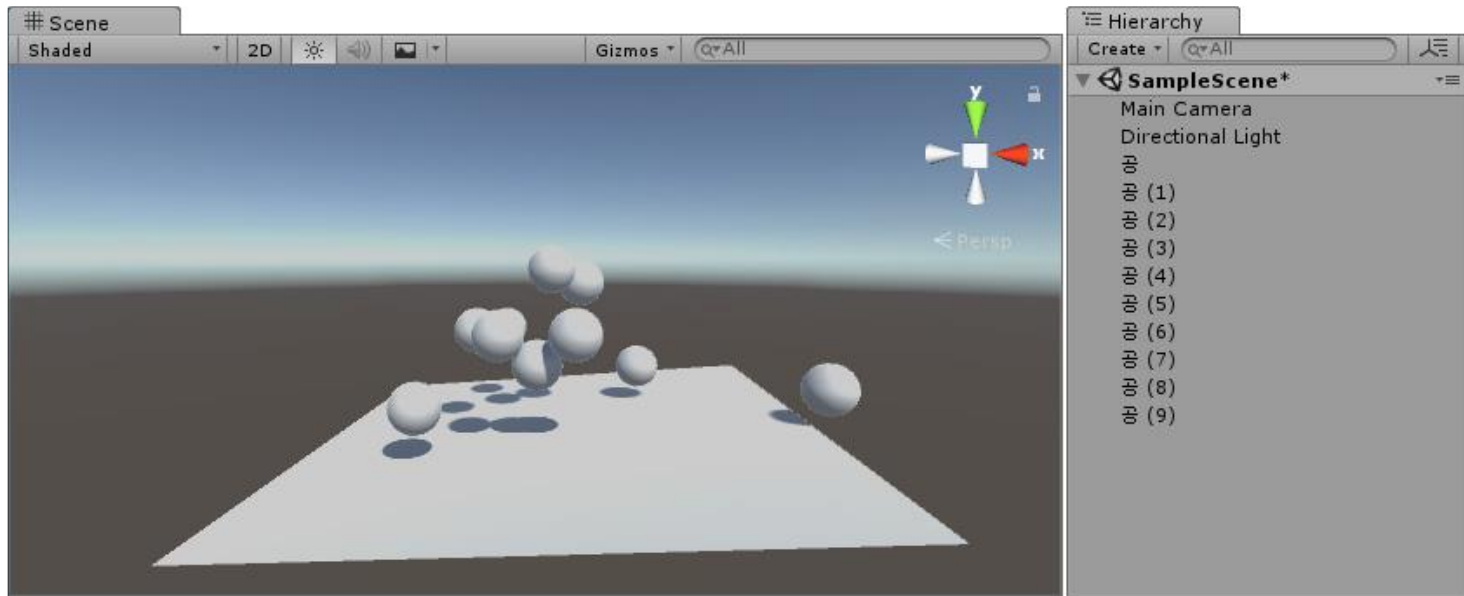


- Physics Material은 Collider 속성이므로, 콜라이더가 없는 오브젝트는 Physics Material을 적용할 수 없음
- 인스펙터 Sphere Collider/Material에 현재 적용된 Physics Material이 표시되는데, [] 버튼을 클릭하여 Physics Material을 변경하거나 제거





오브젝트를 복제(오브젝트 선택 후, Ctrl+D)하여 여러 개를 배치하고 게임을 실행하면, 공이 바닥이나 다른 공과 충돌한 후 반사하여 랜덤하게 흩어지는 것을 확인

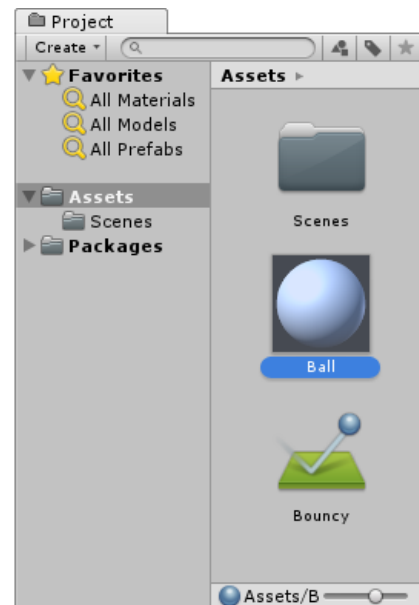
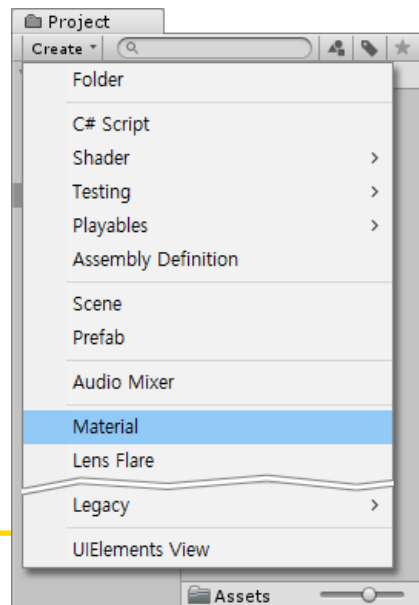


- 매핑

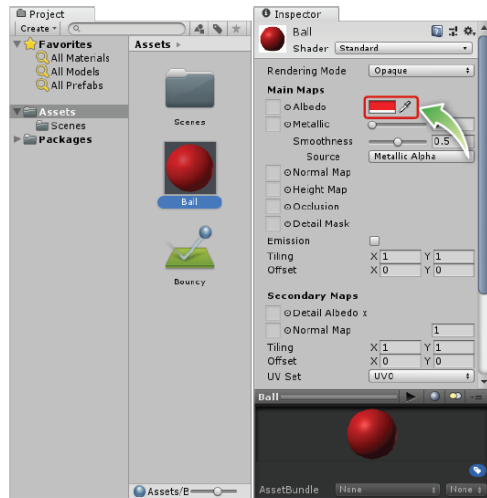
- 오브젝트의 표면을 컬러나 이미지로 씩우는 작업
- 매핑하기 위한 매터리얼(Material)이 필요

(1) 매터리얼 만들기

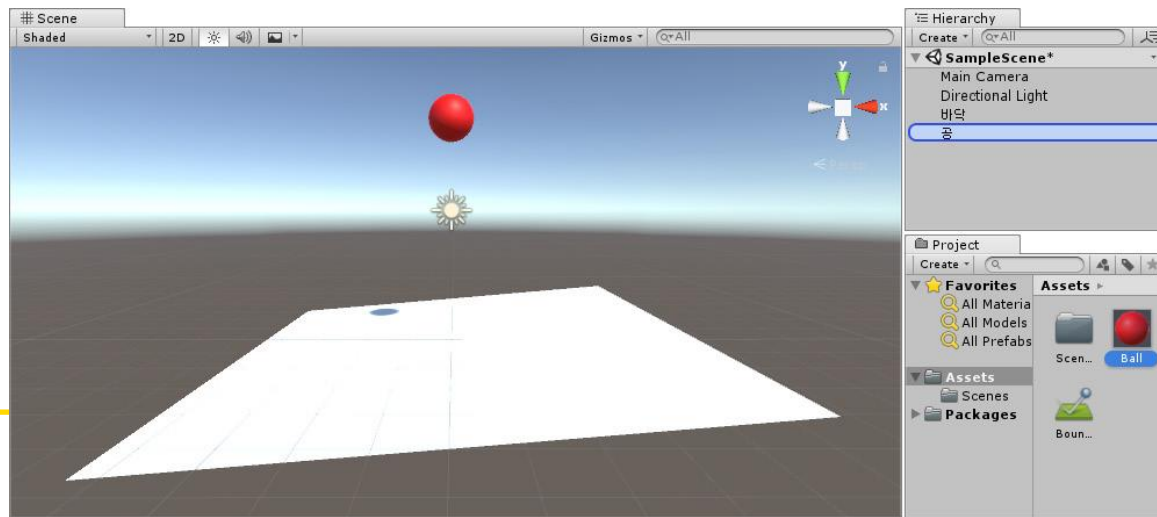
- 프로젝트 브라우저 [Create] 버튼 (또는 오른쪽 마우스 클릭하여 [Create – Material])
- New Material을 만들고 적당한 이름을 부여



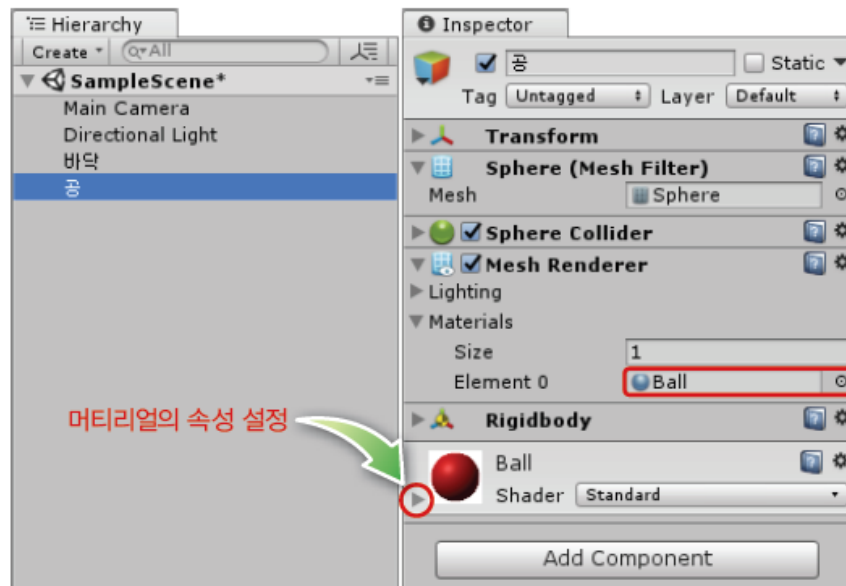
- Albedo 오른쪽에 있는 색상 선택 버튼을 클릭하여 적당한 색을 설정



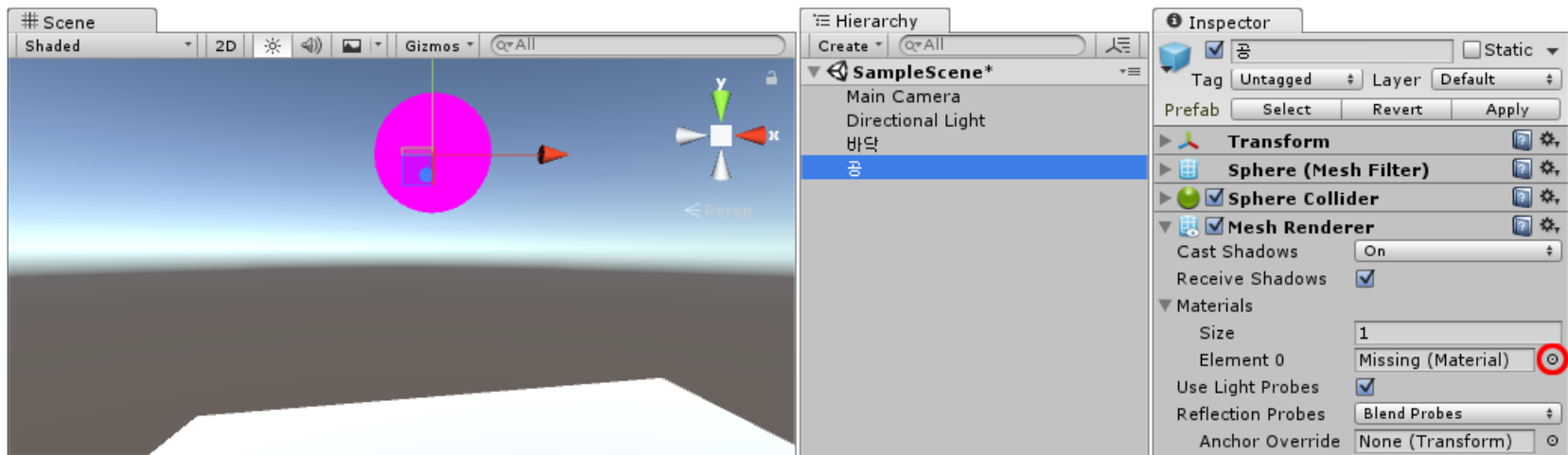
- 생성한 매터리얼을 해당 오브젝트(공)에 드래그




- 오브젝트에 적용한 매터리얼은 인스펙터의 Mesh Renderer/Material에 표시되며, 매터리얼을 수정하거나 속성을 설정



- 오브젝트에 적용된 매터리얼은 삭제하면 안됨
 - 오브젝트의 매터리얼을 삭제하면, 오브젝트의 색상이 음영 없는 **자주색**으로 바뀌고 Mesh Renderer/Materials/Element 속성은 Missing으로 표시됨

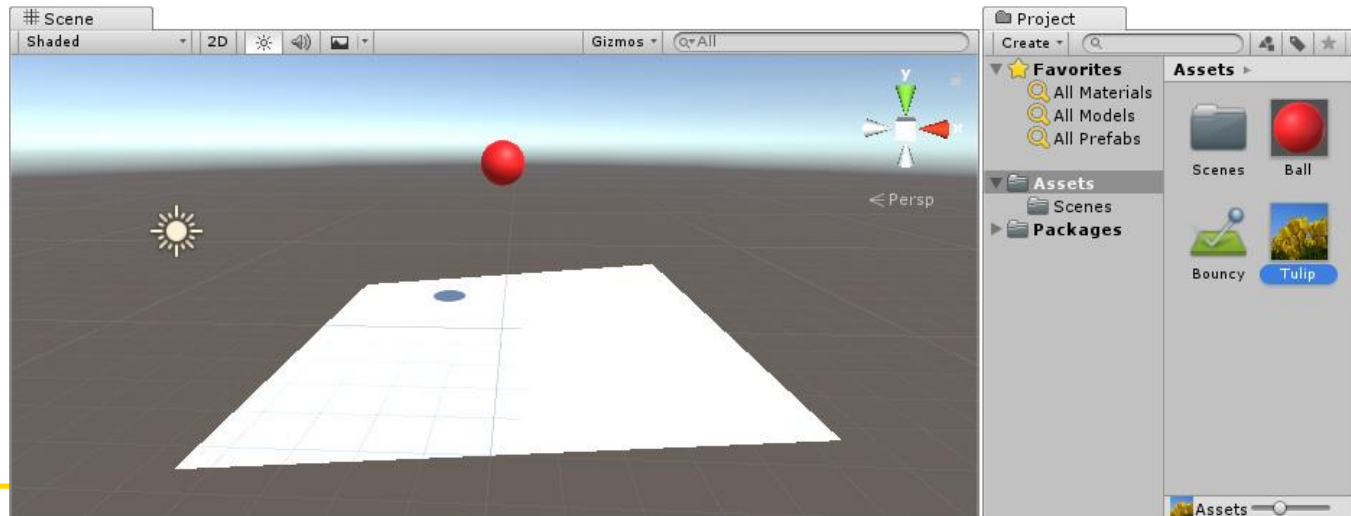


- 이럴 경우, 새로운 매터리얼을 만든 후, 다시 적용하거나 혹은 오른쪽에 있는 [] 버튼을 클릭하여 다른 매터리얼을 적용

(2) Texture로 Mapping하기

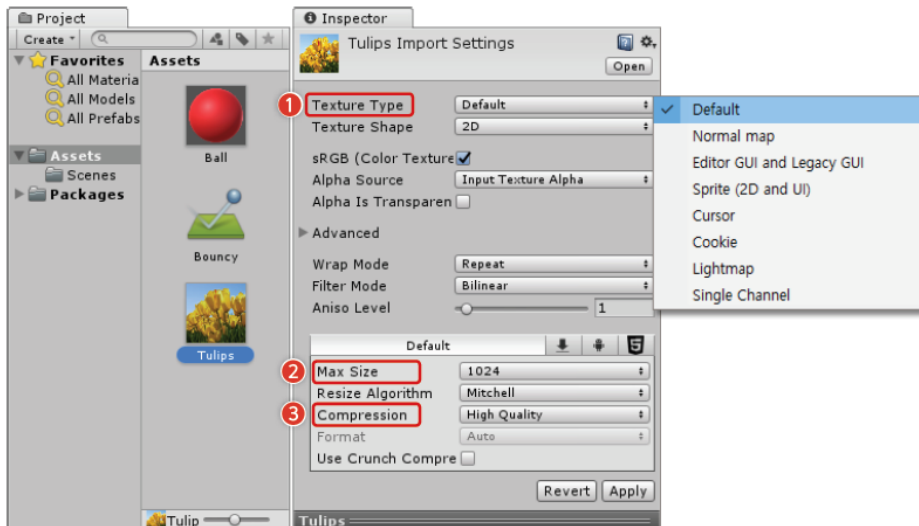
– Texture (텍스처, 무늬)

- 오브젝트의 표면 처리를 위해 다양한 포맷으로 가공한 이미지
(이미지는 텍스처 외에 스프라이트, UI 버튼 등 다양한 용도로 사용됨)
- 오브젝트에 매핑하려면, 매핑하려는 이미지(Mapping Source)가 프로젝트에 추가되어야 함
 - 윈도우 탐색기에서 프로젝트 브라우저에 추가하려는 자원을 드래그
 - (또는) 프로젝트 브라우저에서 오른쪽 마우스 클릭하여 [Import New Asset] 메뉴 클릭하여 윈도우 탐색기에서 추가하려는 자원을 선택



- 이미지 속성 설정

- 추가한 이미지를 선택하여 인스펙터에서 이미지 속성을 설정
- Texture Type (Default), Wrap Mode (Repeat)로 설정



Texture Type	이미지 사용 방식
Max Size	이미지 최대 크기 실제 크기보다 작게 설정하면 이 크기에 맞게 이미지를 Resizing
Compression	이미지 압축 방식



- 유니티에서 이미지 크기는 2제곱수로 리사이즈한 후, 특정 포맷으로 압축하여 저장
- 모바일 플랫폼은 이미지 최대 크기가 2048*2048로 제한된 경우가 있으므로, 이보다 큰 값은 사용하지 않는 것이 좋음

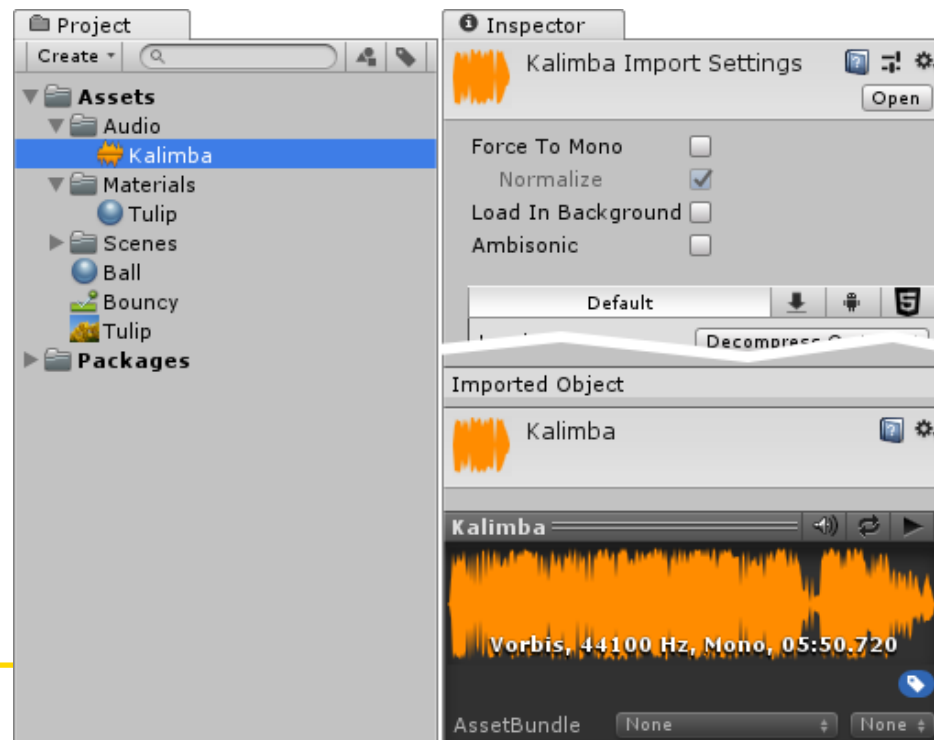
사운드 처리 (Audio)

- 게임에서 배경음악, 각종 효과음 처리는 필수
- 모노, 스테레오, 멀티 채널의 오디오 파일을 지원
(PC/모바일 단말기에서 사용하는 대부분의 사운드 파일 사용 가능)
- 사운드 처리하려면, 3가지 요소가 필요
 - Audio Clip (사운드 파일)
 - Audio Source (사운드 재생용 컴포넌트)
 - Audio Listener (재생한 사운드를 스피커로 전송하는 컴포넌트)



– 오디오 클립(Audio Clip)

- 사용하는 각종 사운드 파일 (aif, wav, mp3, ogg, xm, mod, it, s3m 등)
배경음악은 압축된 형식의 오디오 파일을, 짧은 음향효과는 비압축 오디오 파일을 권장
- 오디오 파일을 프로젝트 브라우저에 추가
- 인스펙터에서 오디오 클립의 압축 포맷, 품질 등 속성을 설정

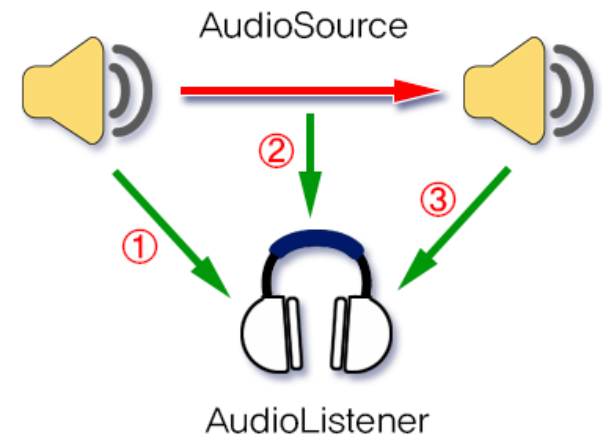


3.8.5 효과음

- 효과음 처리

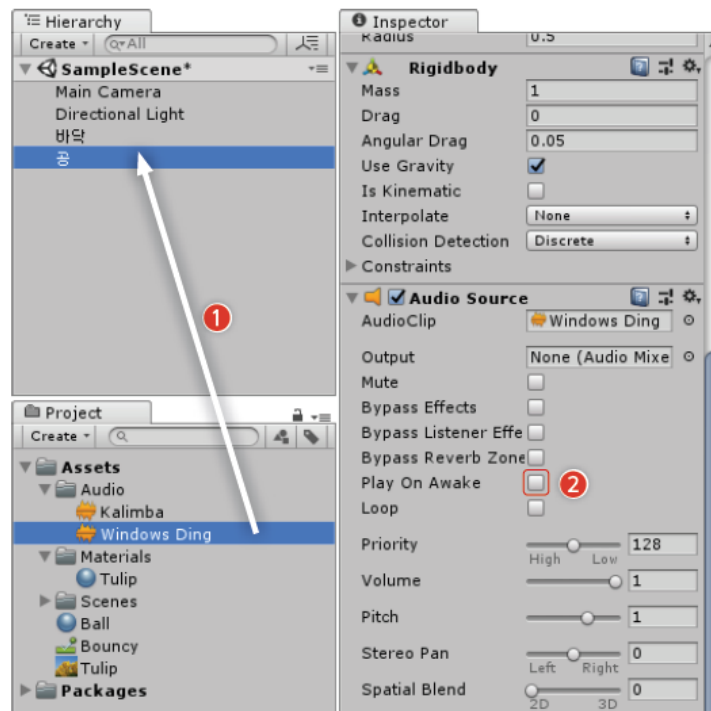
- 공이 다른 물체와 충돌할 때마다 "땡" 하는 효과음을 발생

- 1) Audio Clip 추가
- 2) 충돌 판정 스크립트
- 3) 스크립트 실행



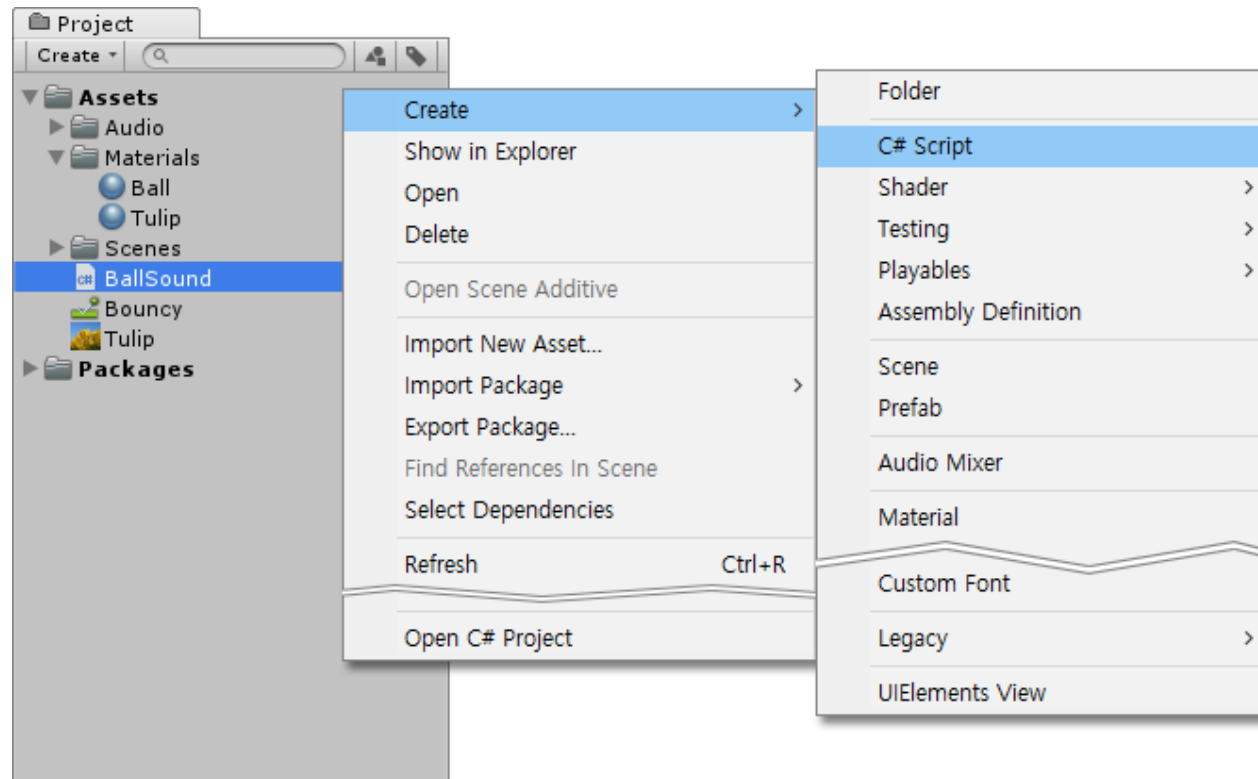
– Audio Clip 추가 :

- 효과음으로 사용할 오디오 클립을 추가
- 추가한 Audio Clip를 공 객체에 드래그하여 연결
- 인스펙터에서 Play on Awake를 OFF로 설정하여 자동 연주 실행을 해제



– 충돌 판정 Script 추가

- 메뉴 [Create – C# Script]를 클릭하여 스크립트를 추가





```
public class BallCollision : MonoBehaviour
{
    // AudioSource를 저장할 변수 선언
    AudioSource ballAudio;

    // Start is called before the first frame update
    void Start() {
        // 시작할 때 컴포넌트 열기
        ballAudio = GetComponent<AudioSource>(); // 씬이 로딩되면 AudioSource를 변수로 읽어 들임
    }

    // Update is called once per frame
    void Update() {

    }

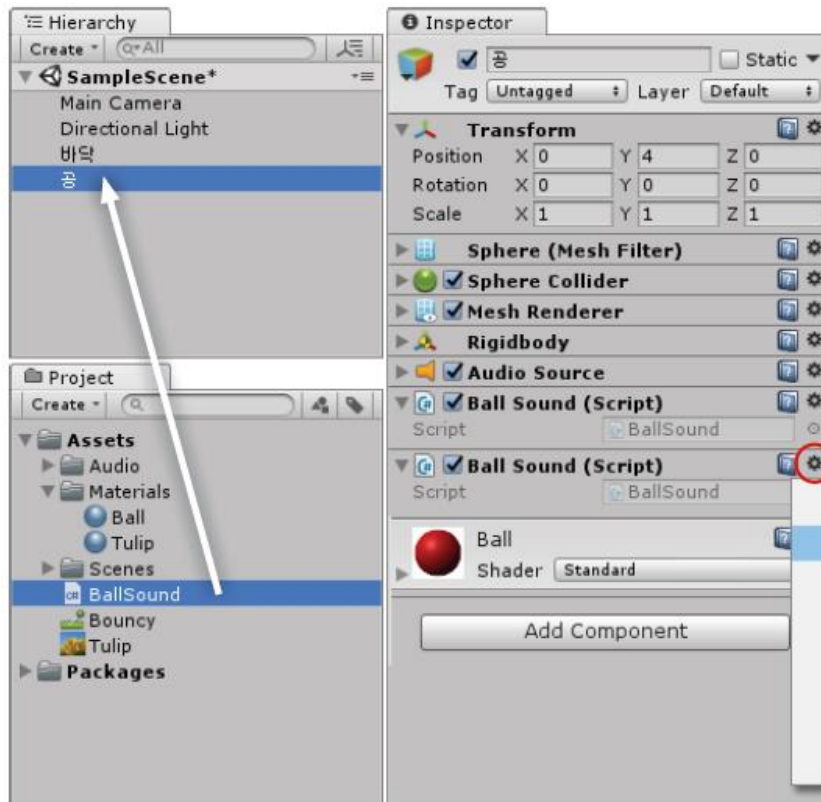
    // 충돌 처리 함수
    void OnCollisionEnter(Collision other) { // 오브젝트 충돌이 발생하면 호출되는 이벤트 함수
        // 충돌의 상대편 정보가 매개변수 other로 전달됨
        ballAudio.Play(); // AudioSource에 할당된 오디오 클립을 재생
    }
}
```

– Script 실행

- 스크립트를 씌운 오브젝트에 드래그하여 연결

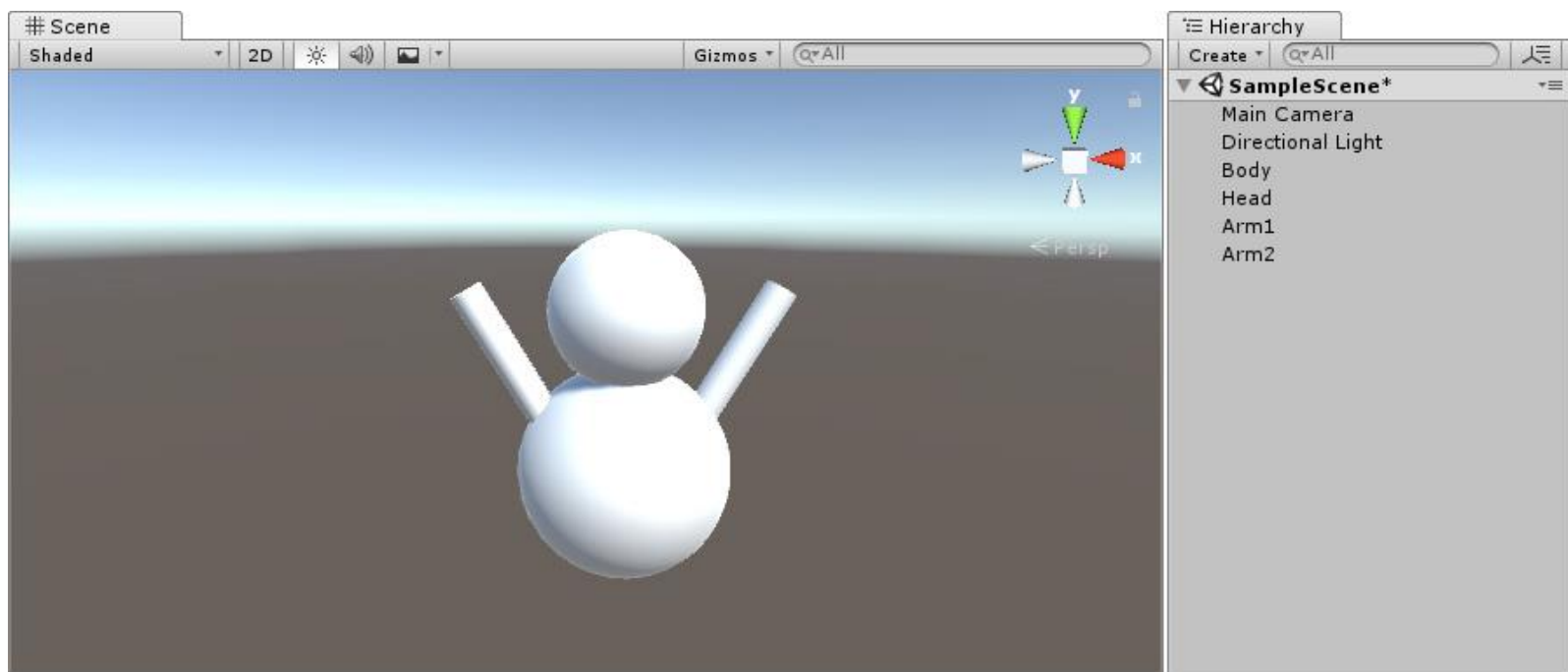


게임을 실행하여 확인



스크립트가
여러 개 중복하여 설정되면
Remove Component 로 삭제

- Object Parenting (Grouping)
 - 여러 개의 오브젝트를 하나로 묶는 작업



– Transform 속성

Object	이름	Position	Rotation	Scale
Sphere	Body	0, 0, 0	0, 0, 0	1, 1, 1
	Head	0, 0.75, 0	0, 0, 0	0.7, 0.7, 0.7
Cylinder	Arm1	-0.5, 0.5, 0	0, 0, 30	0.15, 0.4, 0.15
	Arm2	0.5, 0.5, 0	0, 0, -30	0.15, 0.4, 0.15

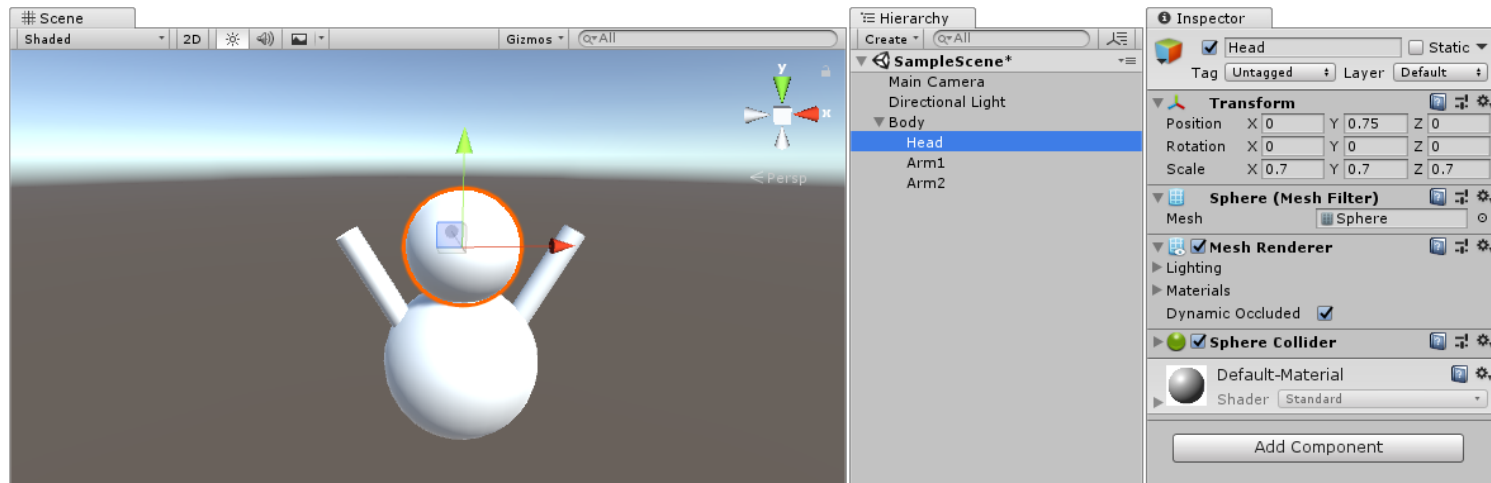
- 오브젝트가 독립된 요소로 분리되어 있으면 이동, 충돌 판정 등의 처리가 번거로워 오브젝트를 하나의 객체로 묶는 것이 편리

– 그룹핑 방법

- 1) 주체가 되는 오브젝트를 정하고, 나머지를 (해당 오브젝트의) 하위 오브젝트로 설정
 - 2) 빈 오브젝트를 새로 만들고, 오브젝트 전체를 빈 오브젝트의 하위 오브젝트로 설정
- 그룹핑 해제 방법 : 해당 오브젝트를 하이라키의 빈 곳으로 드래그

(1번 방법)

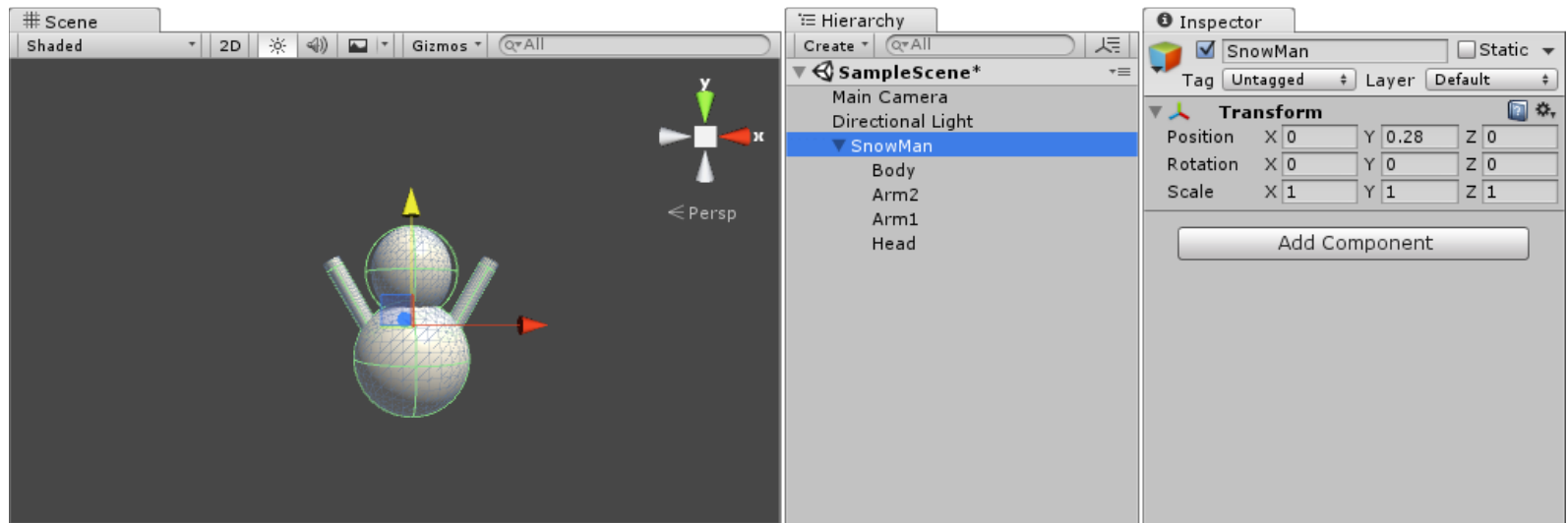
- 핵심 오브젝트를 결정하고 하이라키에서 나머지 오브젝트를 드래그하여 핵심 오브젝트의 하위로 이동시킴
- 그룹핑 해제는 해당 오브젝트를 하이라키에서 빈 곳으로 드래그



- 오브젝트가 계층적으로 구성되어 있을 때, 다른 오브젝트를 포함하는 오브젝트는 Parent, 하위에 있는 오브젝트는 Child
- 오브젝트가 여러 계층으로 구성된 경우, 맨 위의 오브젝트는 Root
- 인스펙터에 표시되는 하위 오브젝트의 Transform은 부모로부터의 상대적인 값 (Local 좌표)

(2번 방법)

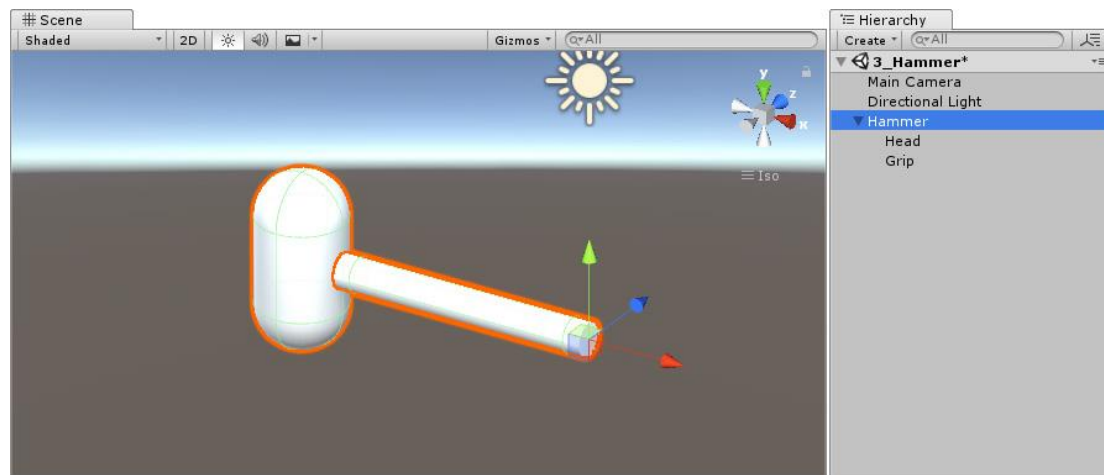
- 메뉴 [**Create – Create Empty**]를 실행하여 빈 오브젝트를 생성
 - 이름은 SnowMan, Position은 (0, 0.28, 0)으로 설정
- 하이라키에서 나머지 오브젝트(Body, Head, Arm1, Arm2)를 드래그&드롭하여 SnowMan 오브젝트의 하위로 이동시킴
- 그룹핑 해제는 해당 오브젝트를 하이라키에서 빈 곳으로 드래그&드롭



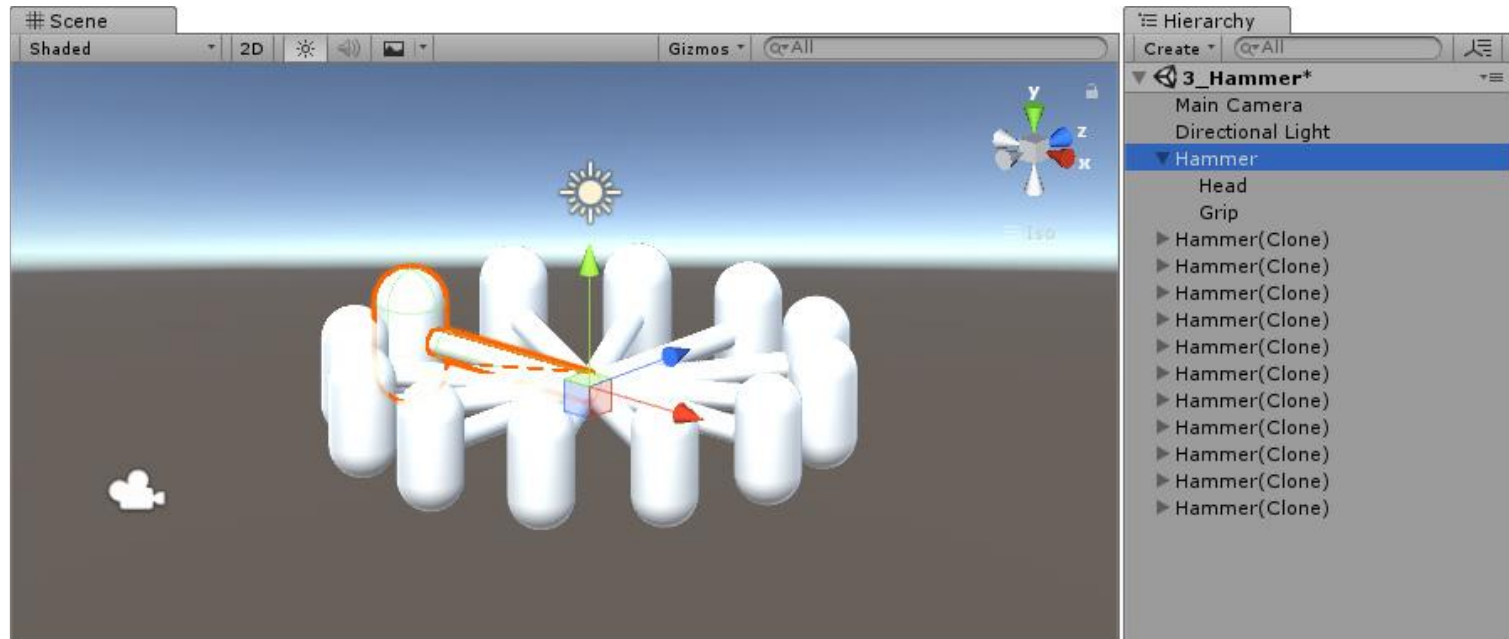
- 빈 오브젝트를 이용하여 그룹핑할 경우, 오브젝트의 피벗을 설정하여 동작시킬 때 유용하게 활용될 수 있음
 - 오브젝트가 회전할 때는 피벗이 회전축이 되며, 유니티에서는 피벗 이동 기능이 없음
 - 피벗의 이동이 필요한 오브젝트는 중심축(회전축)의 위치에 빈 오브젝트를 만들고, 원래의 오브젝트를 빈 오브젝트의 하위 오브젝트로 설정하여 피벗을 이동시키는 효과와 유사하게 만들 수 있음
 - (예) Hammer 오브젝트 생성 및 회전

Object	이름	Position	Rotation	Scale
Empty Object	Hammer	0, 0, 0	0, 0, 0	1, 1, 1
Capsule	Head	-3.5, 0, 0	0, 0, 0	1, 1, 1
Cylinder	Grip	-1.5, 0, 0	0, 0, 90	0.4, 1.5, 0.4

- Head와 Grip을 Hammer로 드래그하여 하위 오브젝트로 만들
- 이때 툴바의 버튼은 [Pivot]으로 설정해야 기즈모의 위치가 제대로 표시됨



- Hammer가 30도 간격으로 y축으로 회전할 경우, Hammer가 손잡이 끝을 회전축으로 회전함

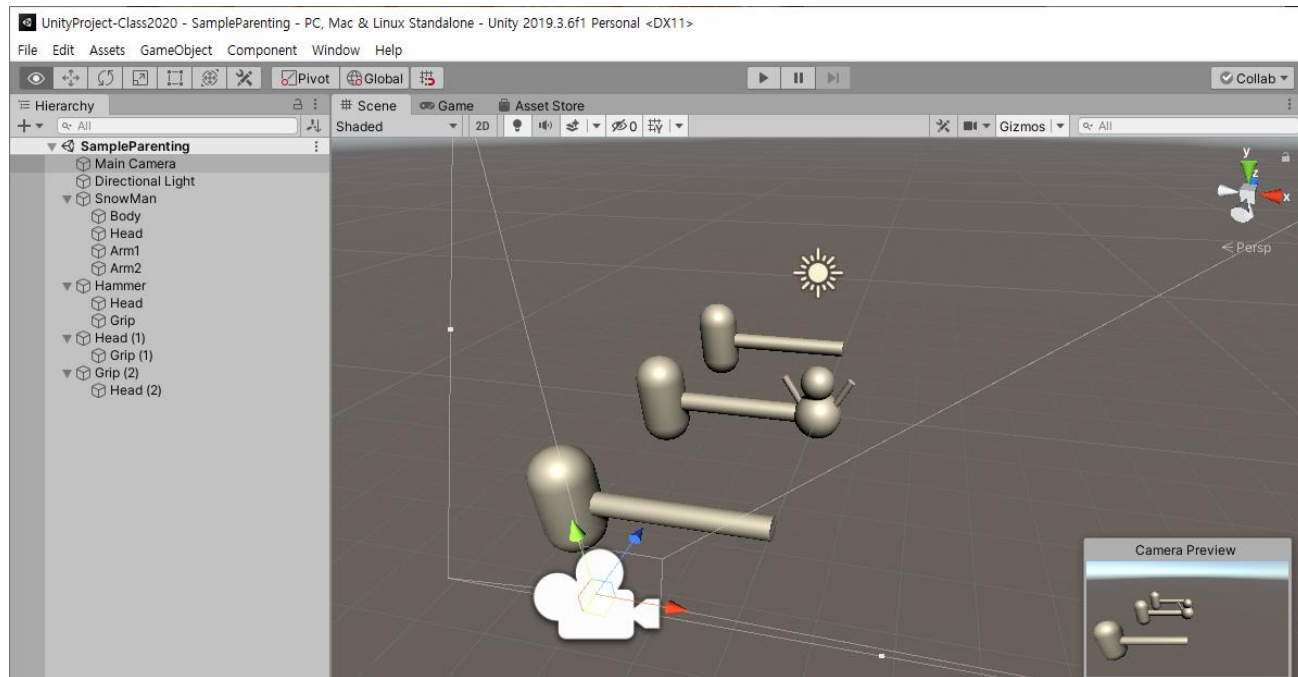




게임을 실행하여 오브젝트 회전을 확인



```
transform.Rotate(new Vector3(0,30,0) * Time.deltaTime);
```



3개의 해머에 대해 Y축을 기준으로 회전시켜 보세요. → Hammer1, Head (1), Grip (2)

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

public class HammerRotate : MonoBehaviour
{
    GameObject objHammer1 = null;
    GameObject objHammer2 = null;
    GameObject objHammer3 = null;

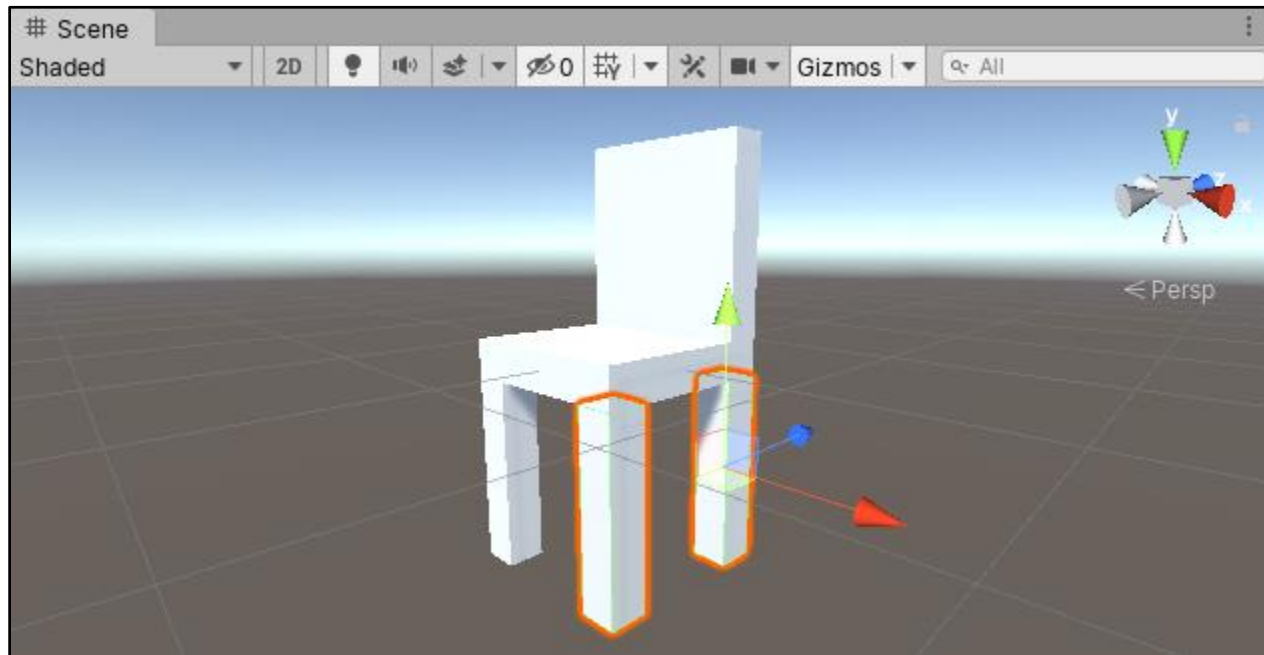
    void Start() {
        objHammer1 = GameObject.Find("Hammer");
        objHammer2 = GameObject.Find("Head (1)");
        objHammer3 = GameObject.Find("Grip (2)");
    }

    void Update() {
        if(objHammer1 != null && objHammer1.name == "Hammer") {
            objHammer1.transform.Rotate(new Vector3(0, 30, 0) * Time.deltaTime);
        }
        if (objHammer2 != null && objHammer2.name == "Head (1)") {
            objHammer2.transform.Rotate(new Vector3(0, 30, 0) * Time.deltaTime);
        }
        if (objHammer3 != null && objHammer3.name == "Grip (2)") {
            objHammer3.transform.Rotate(new Vector3(0, 30, 0) * Time.deltaTime);
        }
    }
}
```

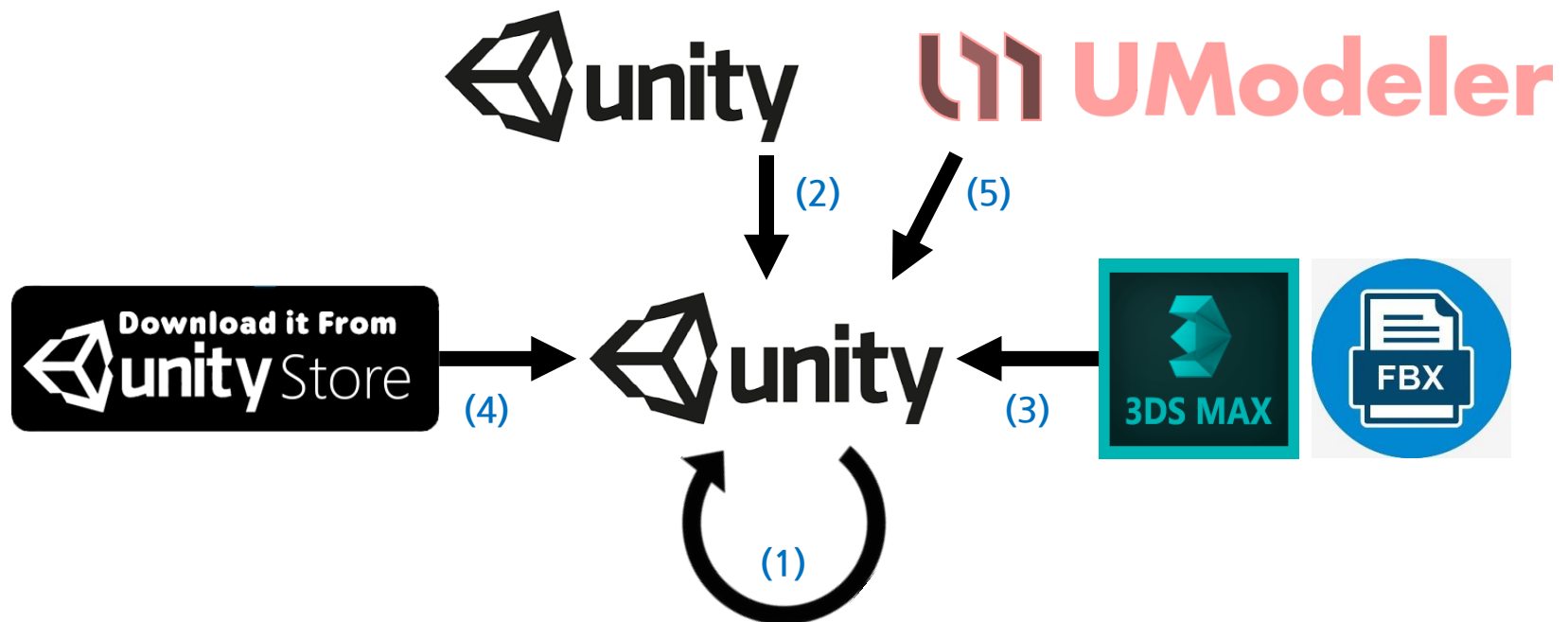


Practical Test (1)

- 아래의 그림을 보고, 기본 오브젝트를 활용하여 게임 오브젝트를 만들어 봅니다



- 게임 오브젝트 생성 방법



- 게임 오브젝트 생성 방법

- 1) Unity 에서 제공하는 기본 오브젝트를 활용하여 생성
- 2) 다른 프로젝트에서 импорт
- 3) 외부 3D 모델을 импорт
- 4) Asset Store 에서 다운로드
- 5) UModeler (3D 모델링을 위한 Unity 플러그인) 에서 제작

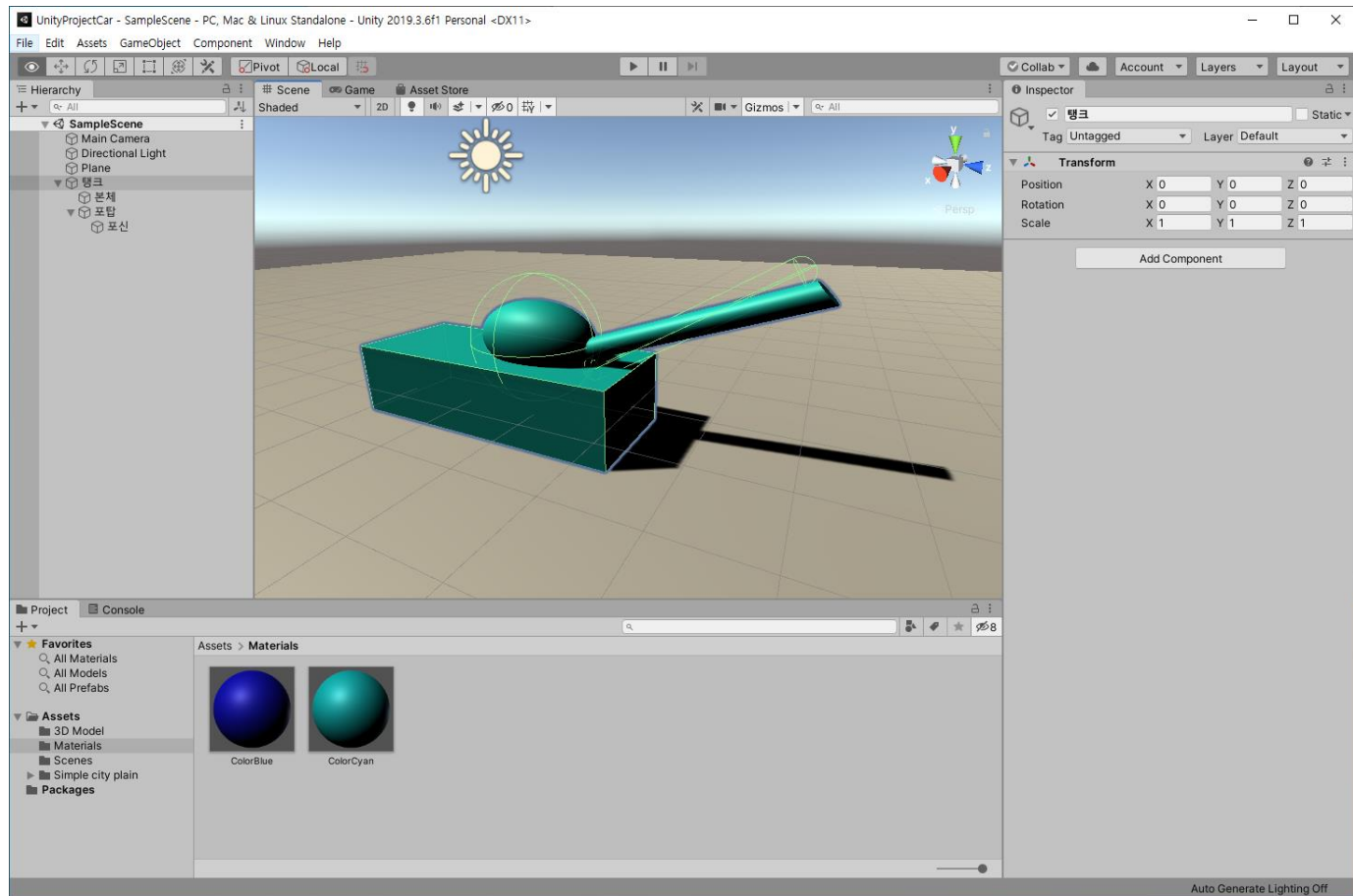
(1) 기본 오브젝트를 활용하여 탱크 모양 만들기

- 계층적인 오브젝트

- 탱크 (Empty Object) : 본체, 포탑, 포신으로 구성
- 포신 : 포탑과 결합되어 함께 회전
- 포탑 : 탱크와 함께 이동, 본체와 독립적으로 회전

- 씬 뷰에서 오브젝트 생성

오브젝트	이름	속성	값	비고
Plane	바닥	Position x, y, z Scale x, y, z	0, -0.5, 0 3, 1, 3	매터리얼 적용
Cube	본체	Position x, y, z Scale x, y, z	0, 0, 0 1.5, 1, 4	
Sphere	포탑	Position x, y, z Scale x, y, z	0, 0.75, 0.6 1, 1, 1.7	
Cylinder	포신	Position x, y, z Rotation x, y, z Scale x, y, z	0, 1.1, 2.6 60, 0, 0 0.3, 1.5, 0.3	
Directional Light				탱크가 잘 보이게 적절하게 조정



– 탱크 결합 (하위 객체로 만들기)

- 별도의 빈 오브젝트를 생성 (메뉴 GameObject → Create Empty)

- 빈 오브젝트 속성

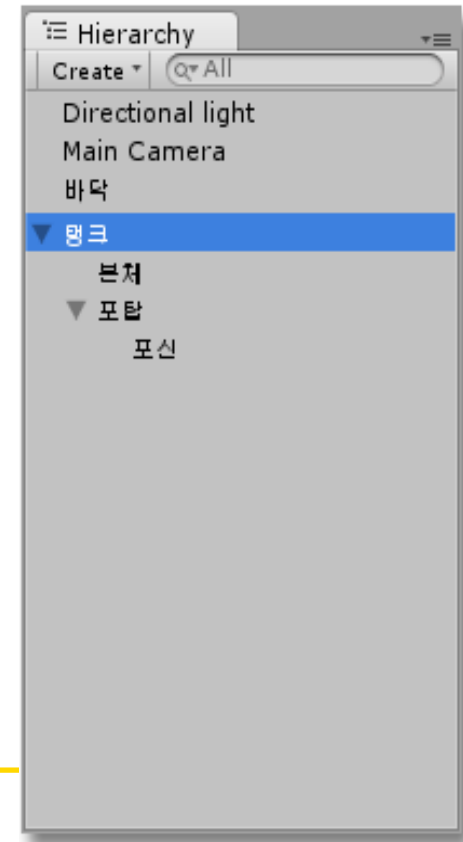
- 이름(탱크), 좌표(0, 0, 0)

- 하이라키 창에서 본체를 빈 오브젝트로 드래그

- 포신을 포탑으로 드래그

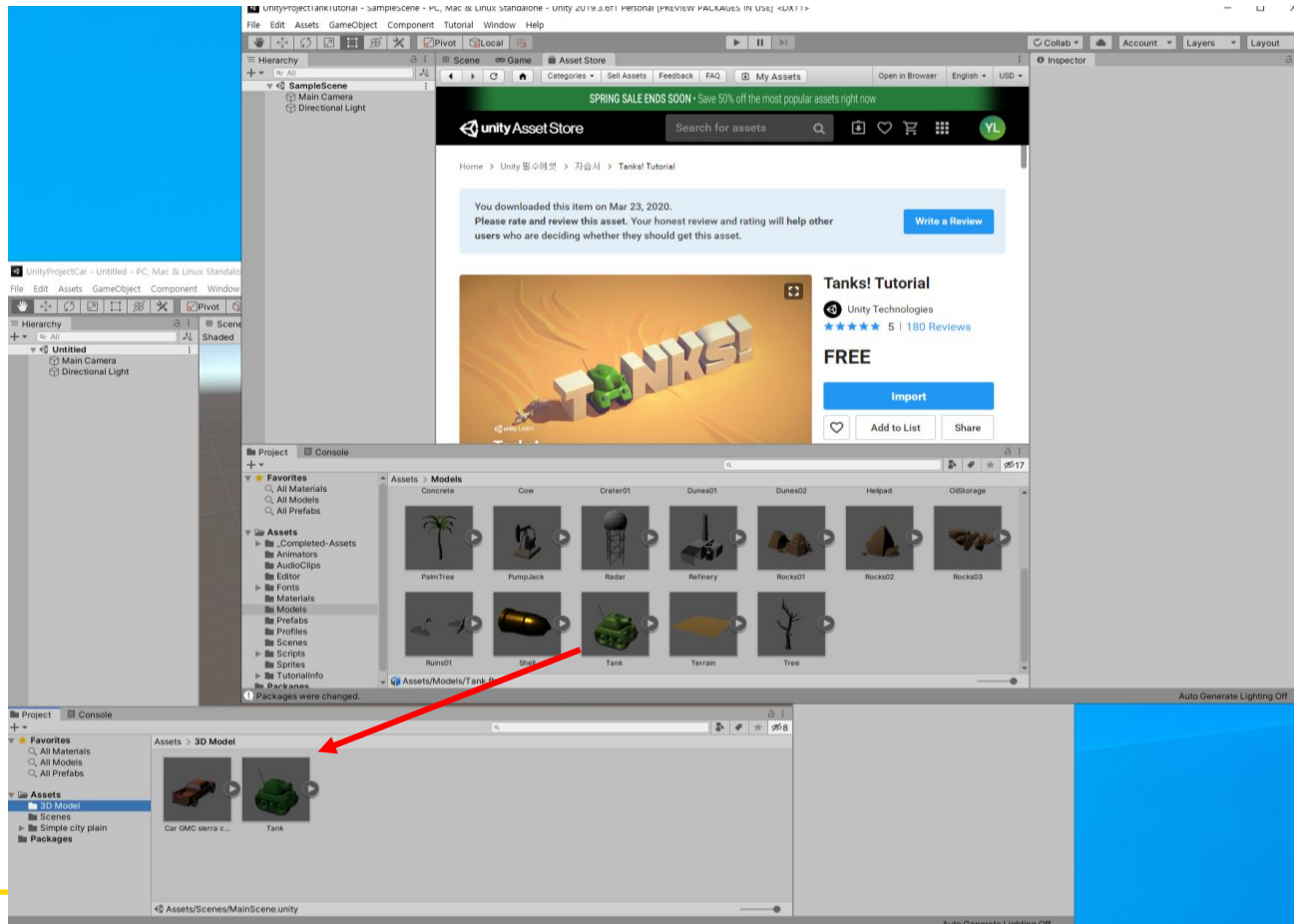
- 포탑을 빈 오브젝트로 드래그

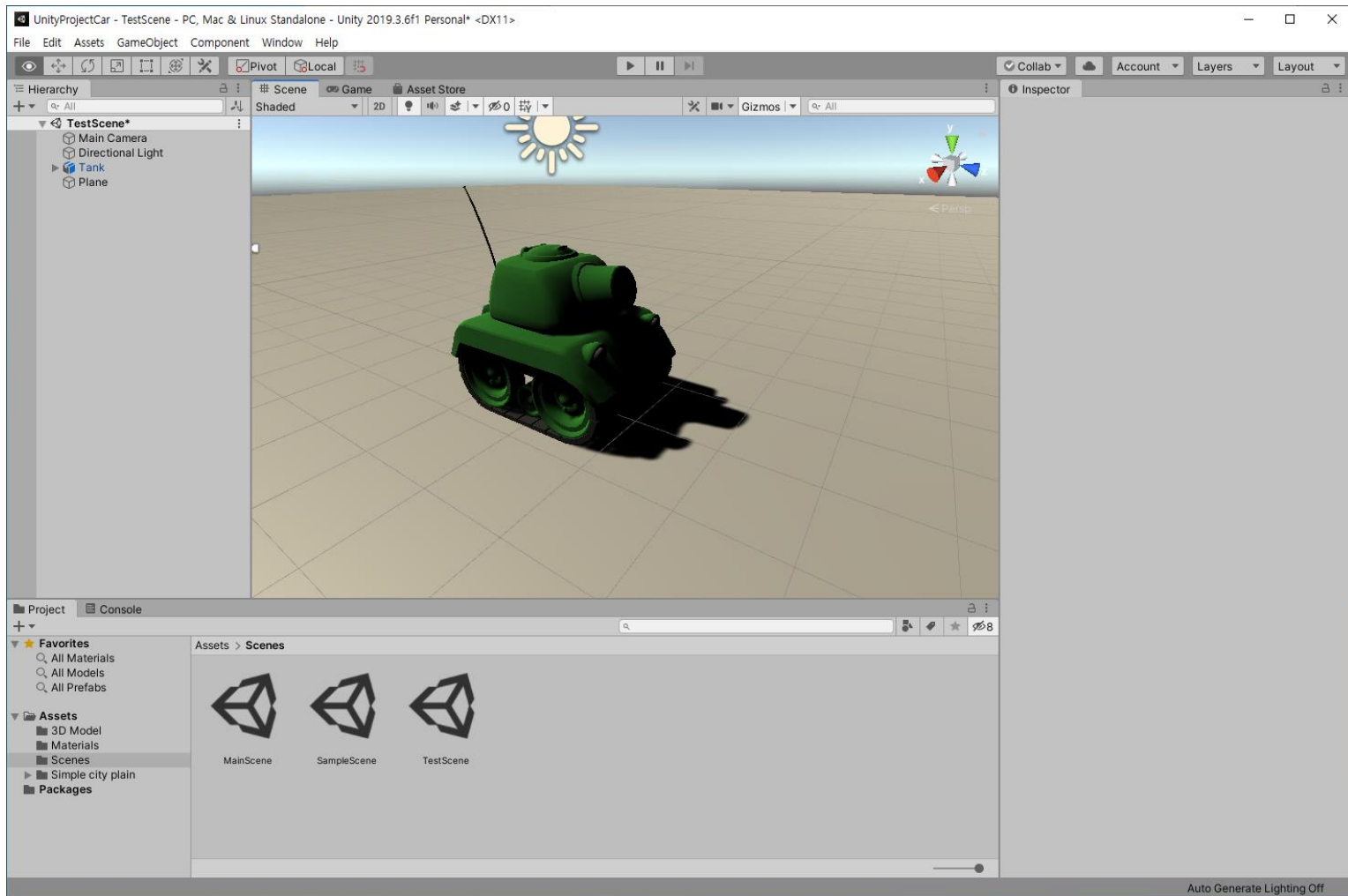
- 탱크를 선택하고 오브젝트를 움직여 본다.



(2) 탱크 오브젝트 가져오기

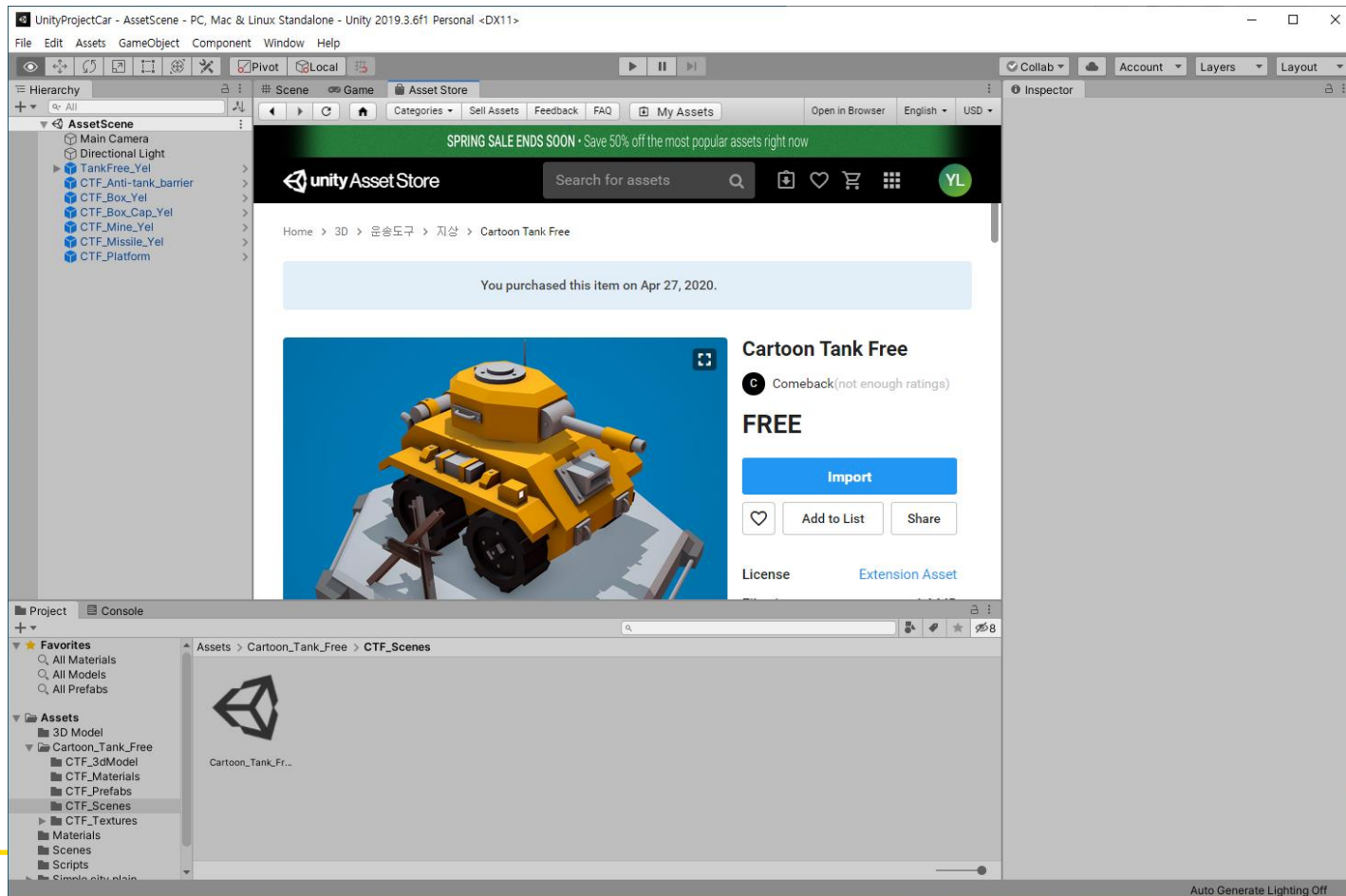
- 다른 프로젝트에서 복사
 - 필요 오브젝트를 드래그 (또는 Export and Import Package)

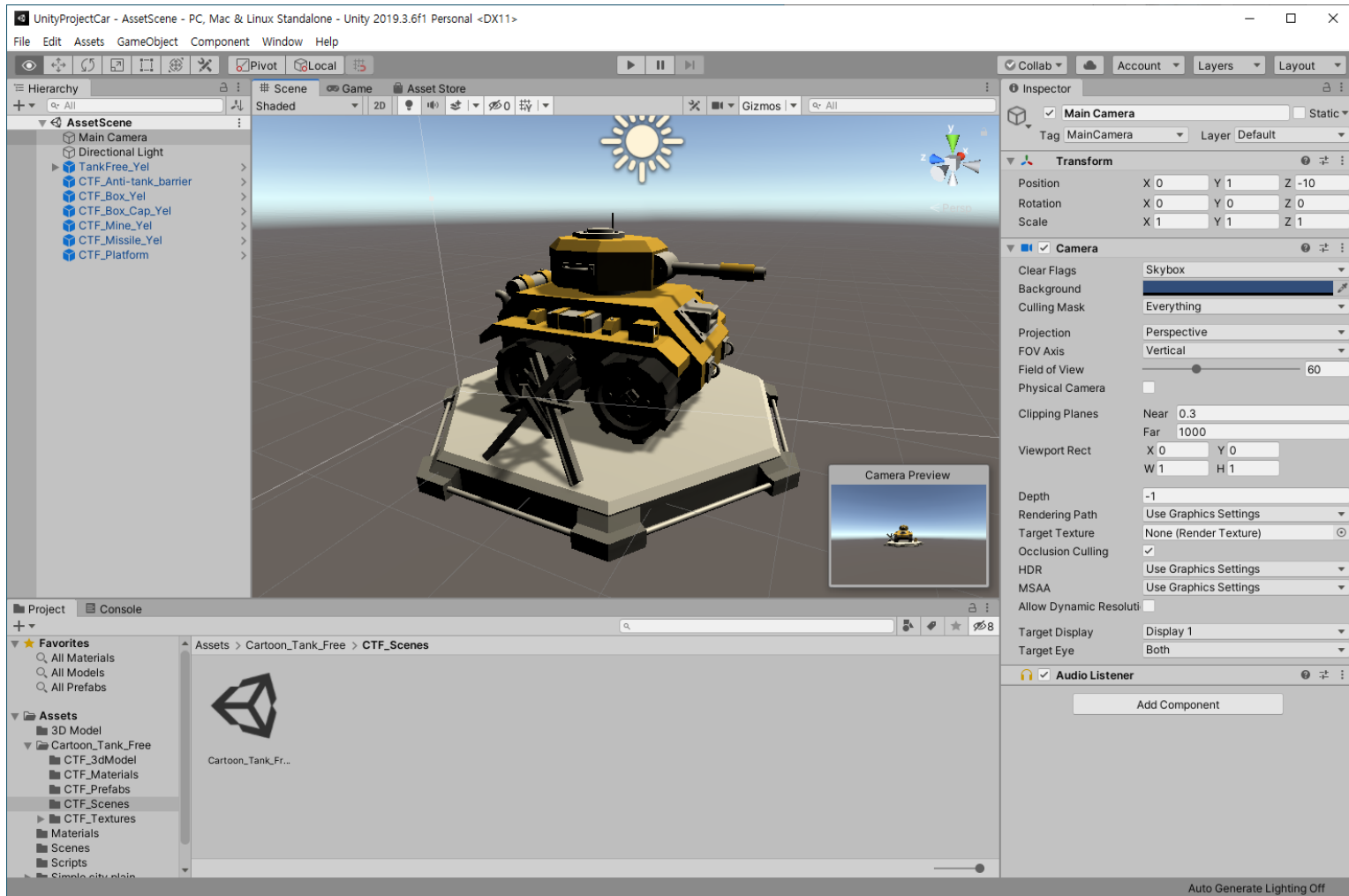




(4) 애셋스토어에서 탱크 다운로드 받기

- Asset Store 에서 다운로드 (무료 또는 유료)
 - 오브젝트를 검색하여, 다운로드 + импорт



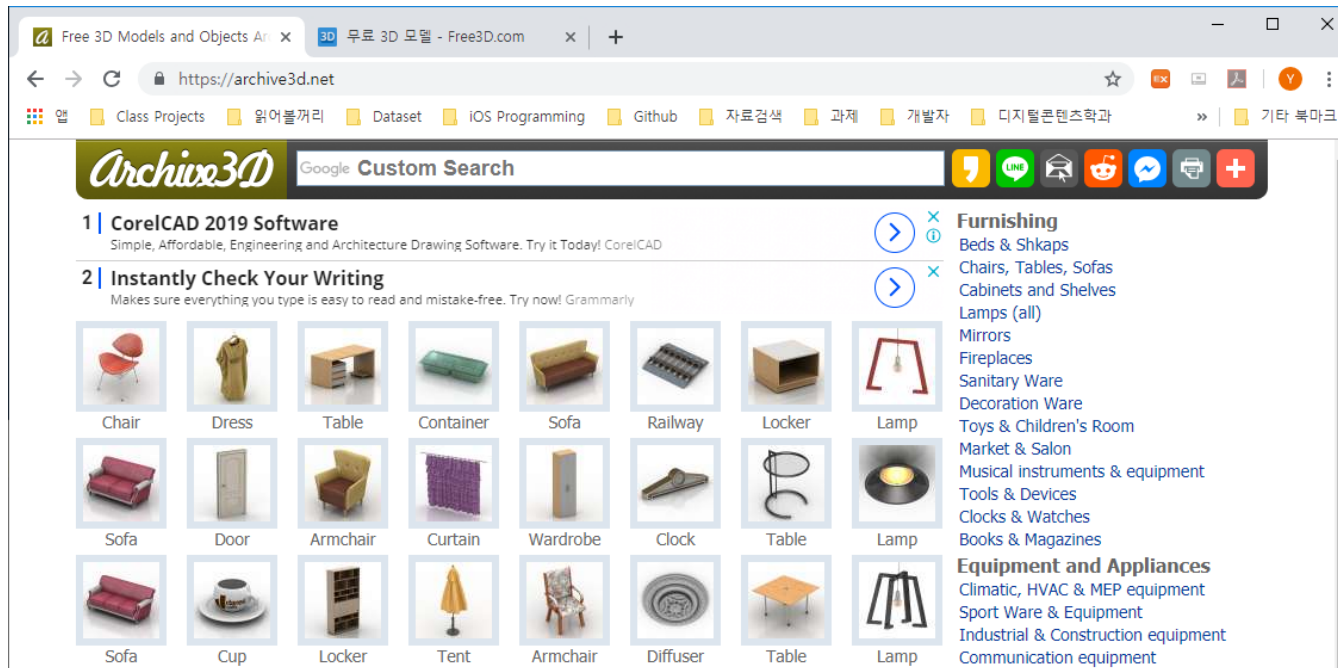


(3) 3D Modelling 파일 가져오기

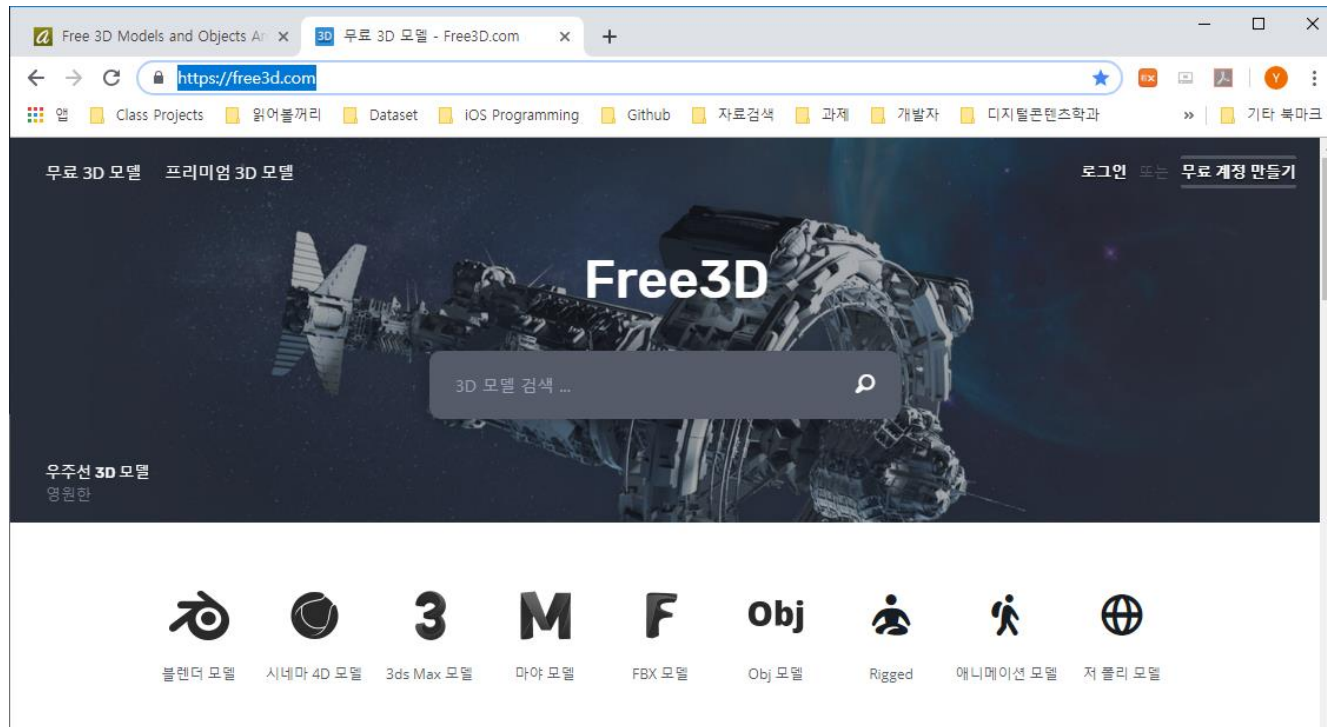
- 3D Model Import

- 무료 다운로드 사이트

- <https://archive3d.net/> “Car 3D Model” 로 검색



- <https://free3d.com/>

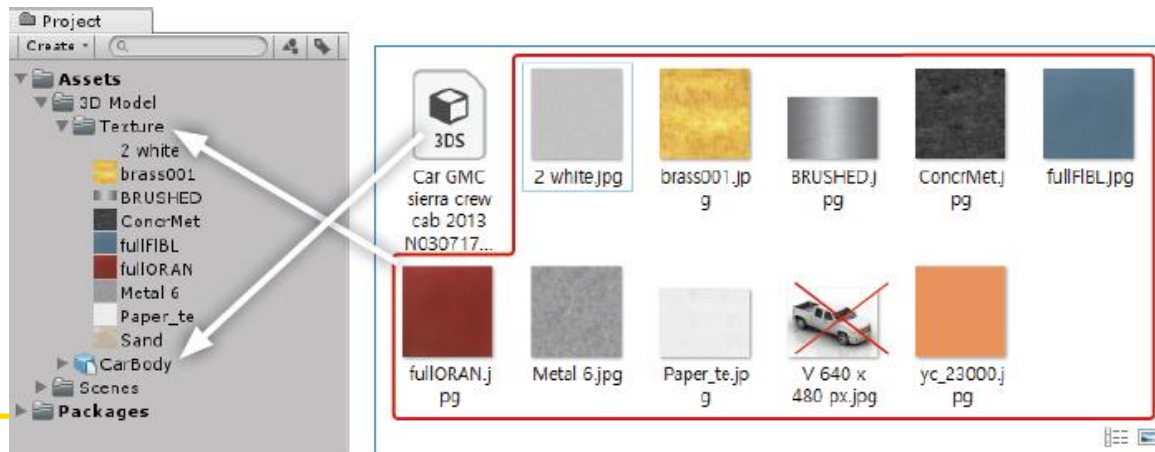


– 3D 모델을 다운로드하고 씬에 추가

- 적절한 3D 모델 파일을 다운로드



- 압축을 풀고, 이미지와 모델파일을 남기고 삭제 (불필요한 파일 삭제)
- 프로젝트 브라우저에서 3DS파일을 드래그 하여 추가 (이름을 적절하게 변경)
- Texture 폴더를 생성하고 이미지 파일들을 드래그 하여 추가



- 3D 모델을 하이라키키로 드래그 하여 씬에 추가

