가상현실 및 실습

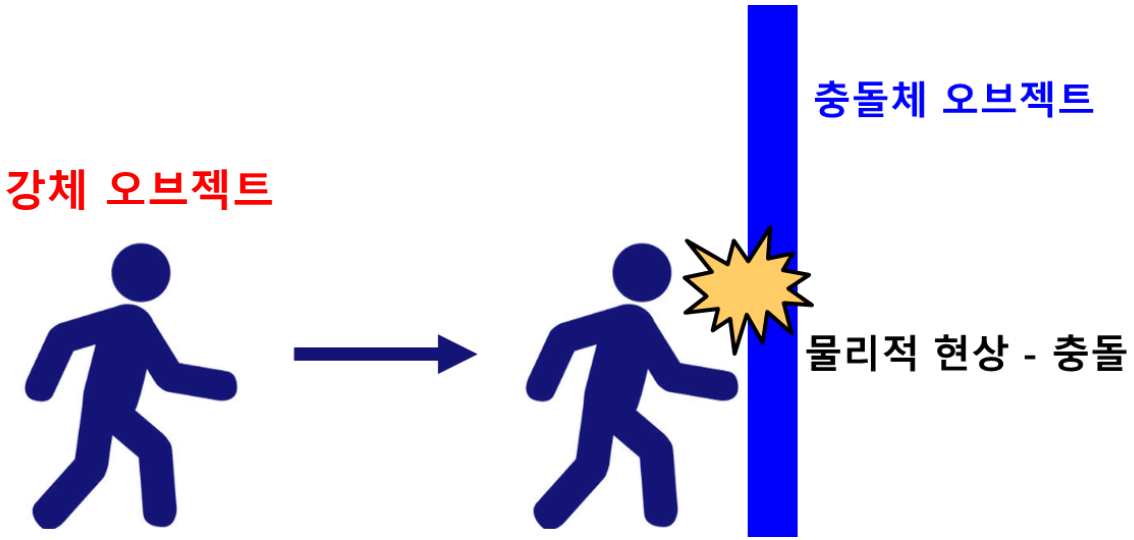
충돌처리

유니티의 물리 엔진

2. 유니티의 물리 엔진

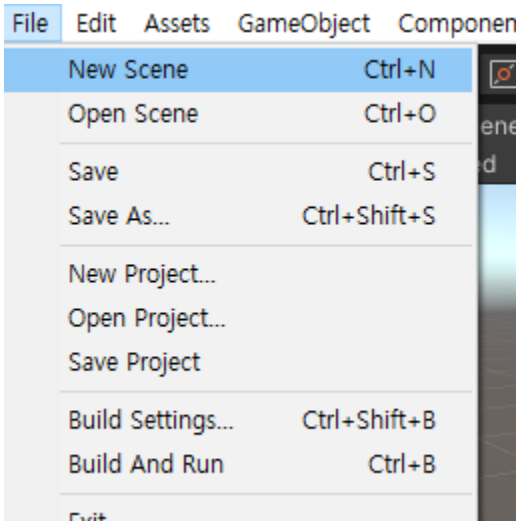
1) 물리엔진

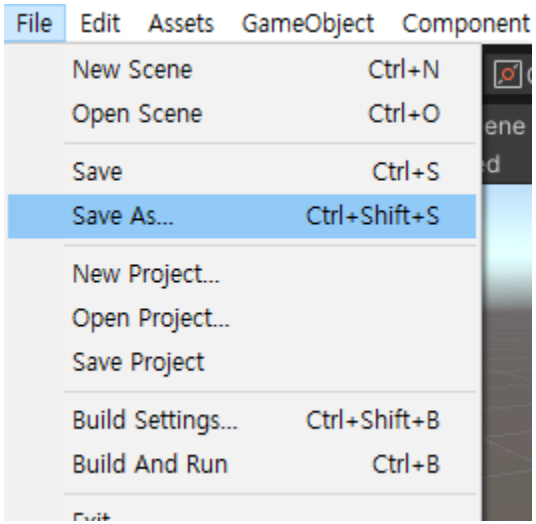
- 물리 엔진은 오브젝트의 중력 또는 오브젝트 간의 충돌과 같은 물리적 현상을 시뮬레이션할 수 있음
- 유니티의 물리 엔진을 사용하기 위해서는 다음의 2가지 조건이 필요함
  - ➢ **물리적 현상이 적용될 오브젝트는 강체(Rigidbody) 컴포넌트가 필요**함
  - ➢ 강체는 **충돌체(Collider) 컴포넌트를 가진 오브젝트**에게만 충돌 등이 발생함

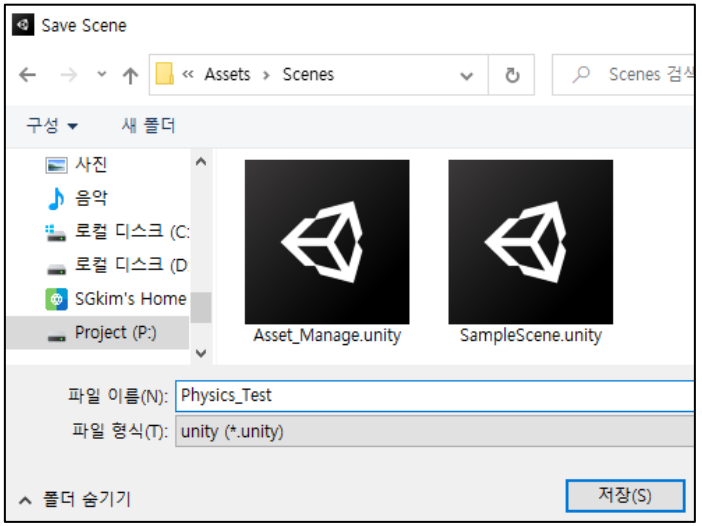


2) 씬 설정

- **메뉴 > File > New Scene을 선택**하여 새로운 씬을 생성함
- **메뉴 > File > Save As...를 선택**하여, 생성한 씬을 폴더 "Scenes"에 저장함
- **저장 시, 파일 이름은 "Physics_Test.unity"**



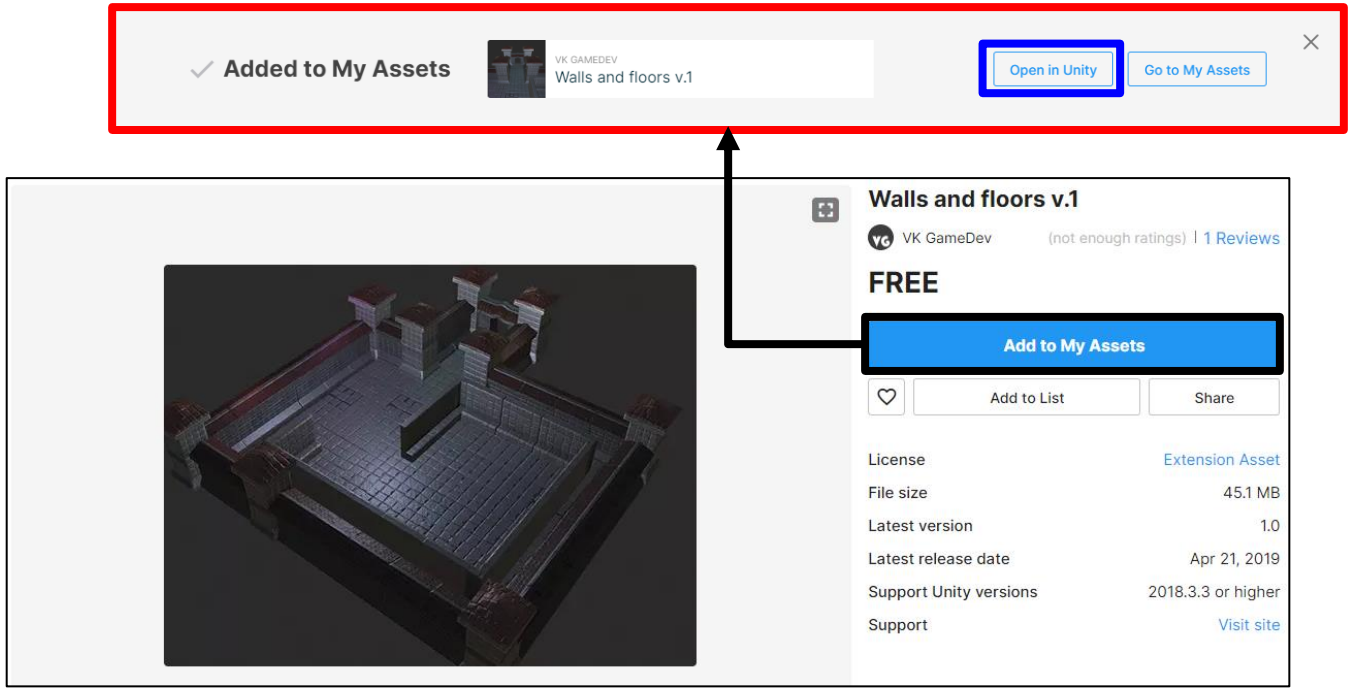




2) 에셋 스토어 활용 방법

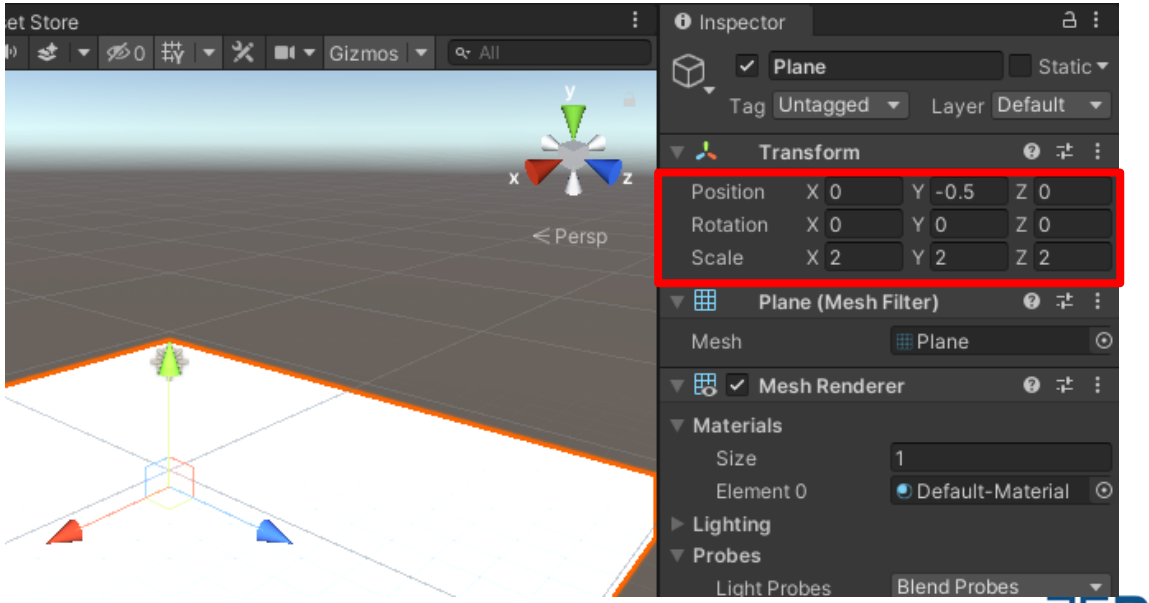
- **Add to My Assets 버튼을 선택**
- **Added to My Assets 창이 나타나면, Open in Unity 버튼을 선택**

  (단, 유니티에서 프로젝트를 실행 중이어야 연동이 가능함)



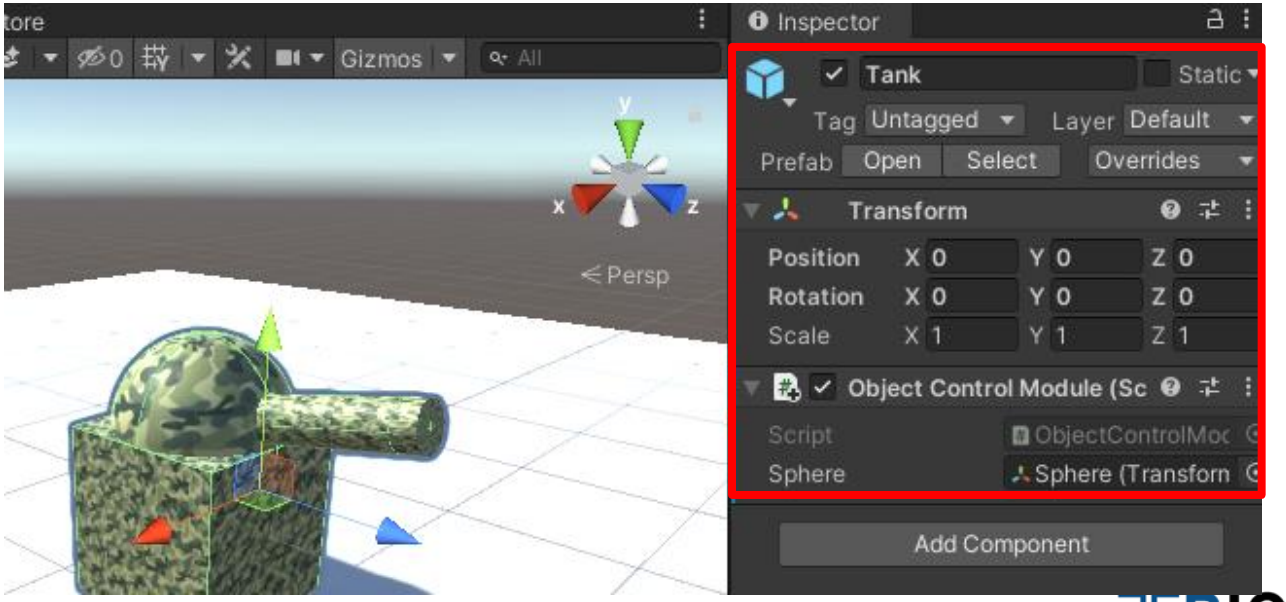
2) 씬 설정

- Hierarchy > 마우스 우 클릭 > 3D Object > Plane을 선택함
- 생성된 **오브젝트 "Plane"의 Transform 값을 다음과 같이 설정**함
  - **Position: (0, -0.5, 0)**
  - **Rotation: (0, 0, 0)**
  - **Scale: (2, 2, 2)**



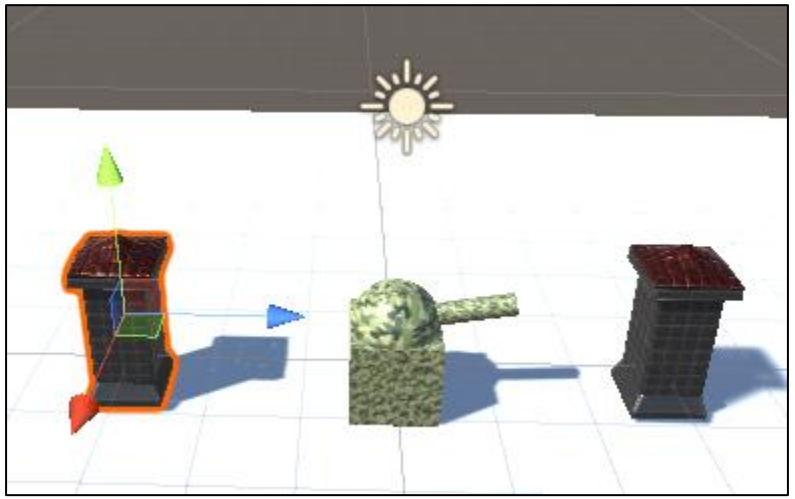
2) 씬 설정

- 프리팹 " 탱크 " 와 스크립트 "Fire", "PlayerController"을 이용하여 키보드를 통해 움직이는 오브젝트 "탱크"를 생성함
  - **Position: (0, 0, 0)**
  - **Rotation: (0, 0, 0)**
  - **Scale: (1, 1, 1)**



2) 씬 설정

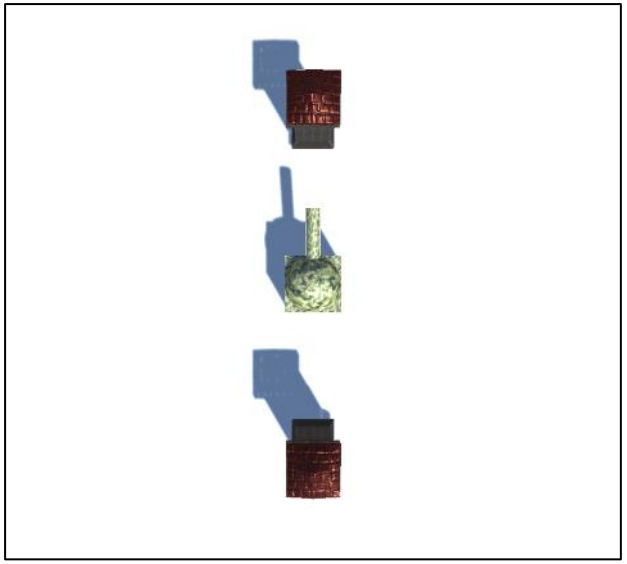
- 프로젝트 창 > Assets > Walls and floors v.1 > Prefabs > 프리팹 "Column"을 이용하여 **다음과 같이 기둥 오브젝트 2개를 생성함**
- 생성된 기둥 오브젝트는 아래 표와 같은 Tansform 값을 가짐



구분	첫번째 기둥	두번째 기둥
위치	0, 0, -3	0, 0, 3
회전	-90, 0 , 0	-90, 0 , 0
크기	40, 40, 40	40, 40, 40

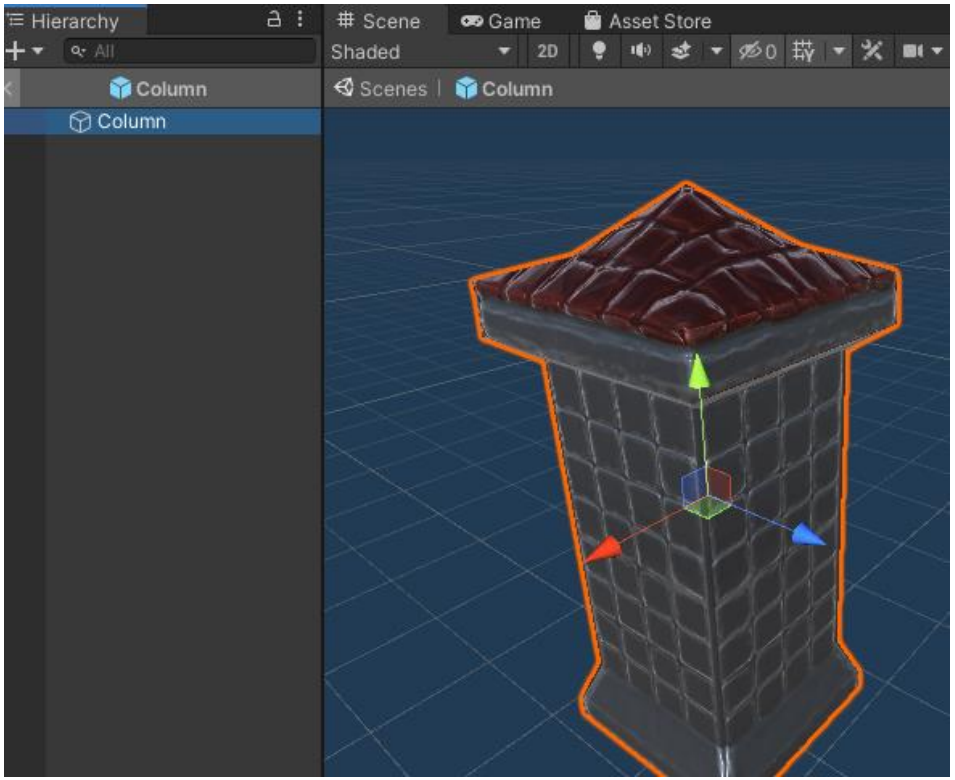
2) 씬 설정

- Hierarchy > **Main Camera의 Transform 값을 다음과 같이 설정**하여, Game 창을 선택하였을 때 위에서 바라볼 수 있도록 함
  - Position: (0, 10, 0)
  - Rotation: (90, 0, 0)
  - Scale: (1, 1, 1)



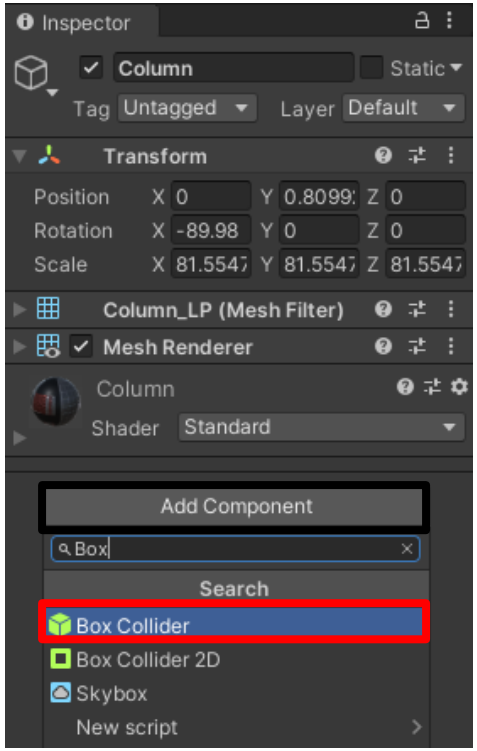
3) 물리엔진 적용: 충돌체 설정

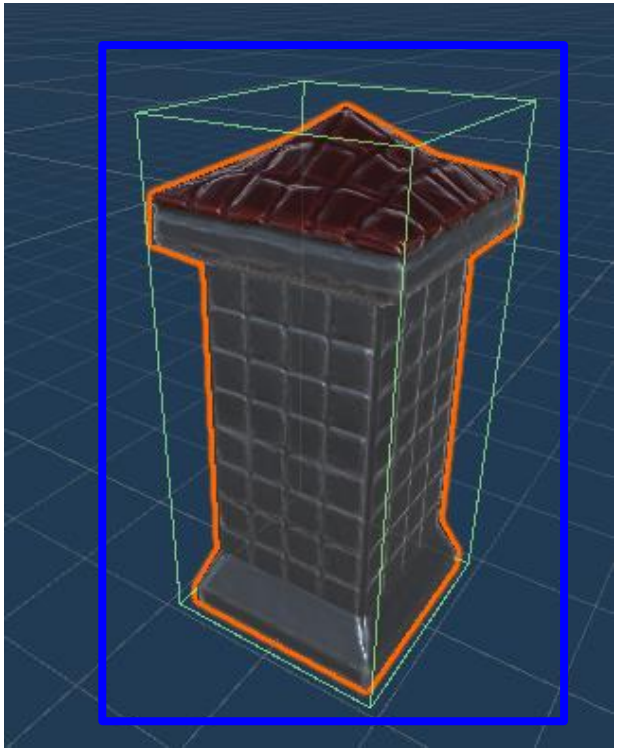
- 프로젝트 창 > Assets > Walls and floors v.1 > Prefabs > **프리팹 "Column"을 더블 클릭함**
- Hierarchy > **충돌체를 부여할 오브젝트를 선택**



3) 물리엔진 적용: 충돌체 설정

- Inspector > **Add Component 버튼을 선택** > **Box Collider 입력 후 선택**
- Box Collider 컴포넌트가 추가되면서 **충돌 영역이 생성된 것을 확인**할 수 있음

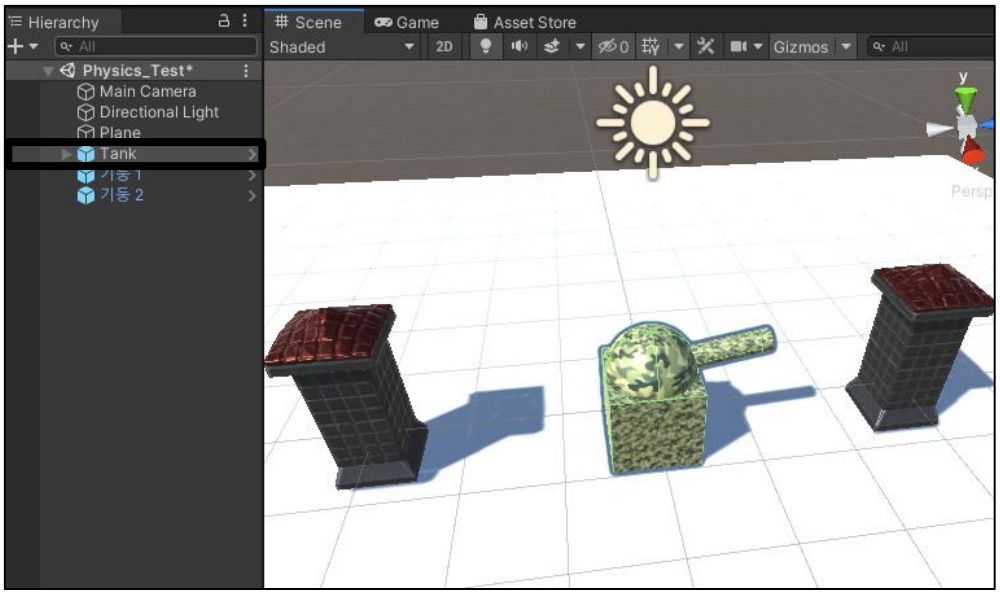


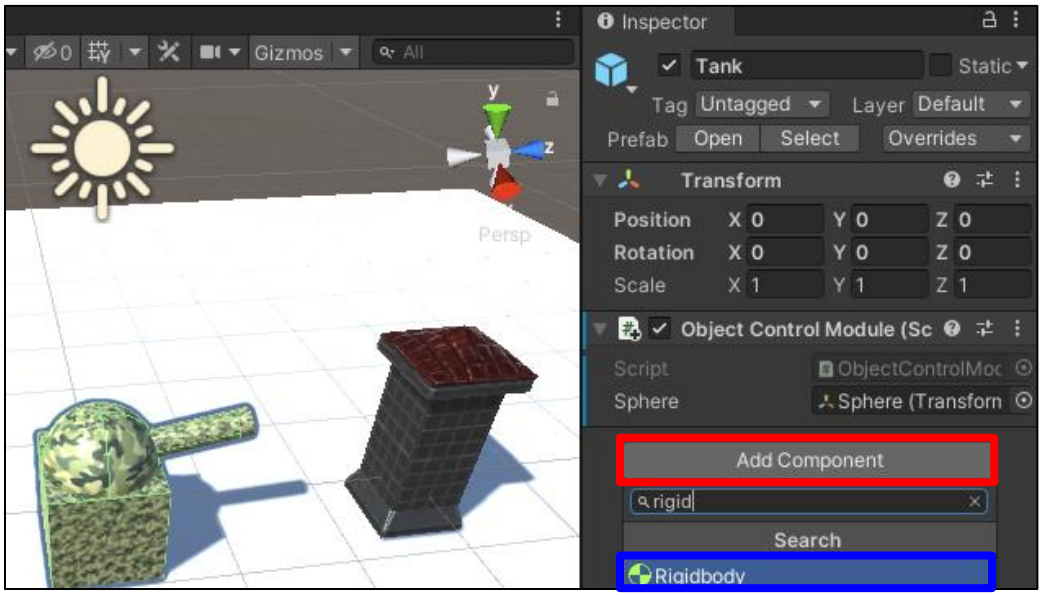


2. 유니티의 물리 엔진

3) 물리엔진 적용: 강체 설정

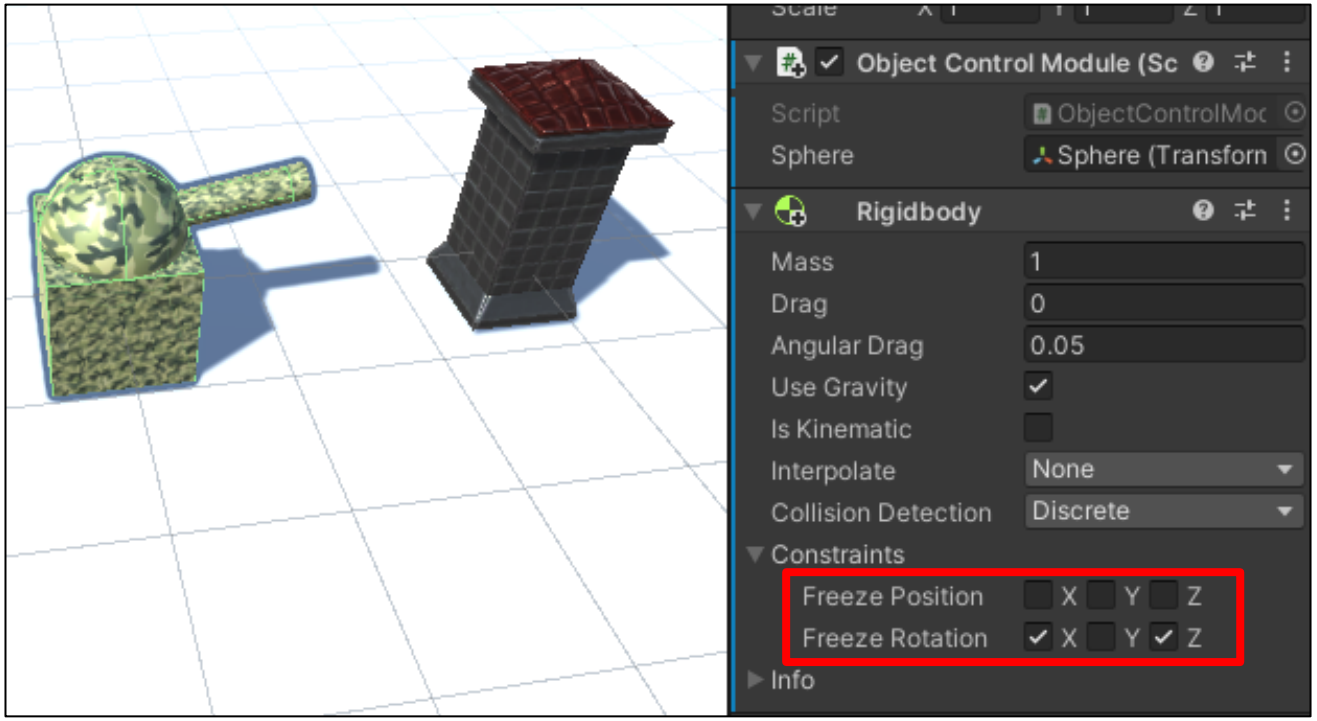
- Hierarychy > **오브젝트 "탱크"를 선택**
- Inspector 창 > **Add Component 버튼을 선택** > **Rigidbody 입력 후 선택**





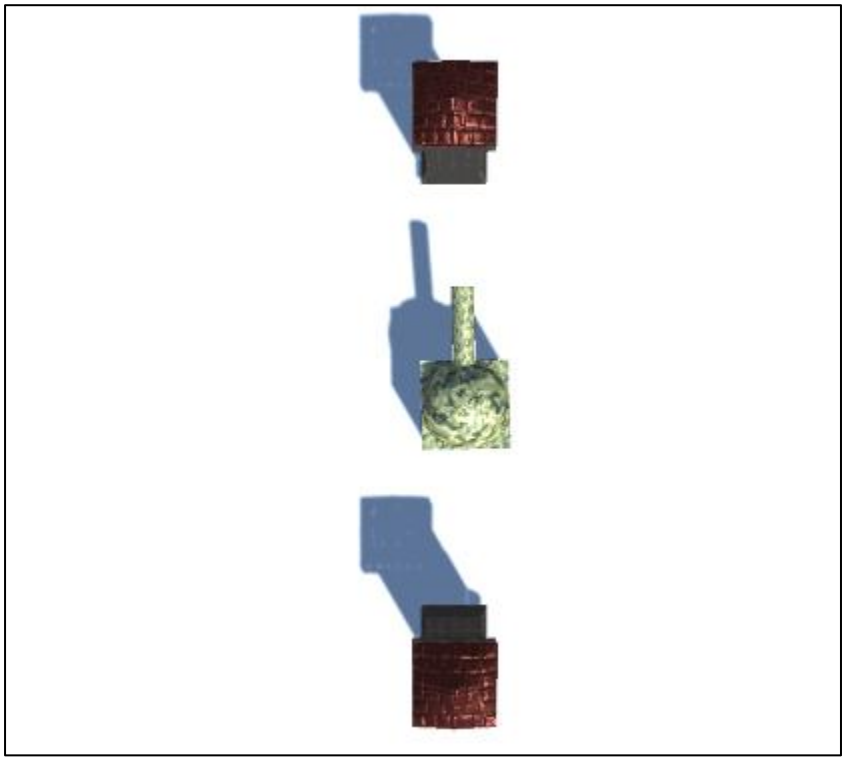
3) 물리엔진 적용: 강체 설정

- 강체 컴포넌트는 매우 다양한 물리적 현상(예. 충돌, 변형, 힘, 중력 등)을 포함함
- 본 실습에는 강체가 충돌체와 부딪혀서 넘어지는 것을 방지하기 위해,
  좌우 회전 (Y)를 제외한 나머지 회전 (X, Z)을 고정함



3) 물리엔진 적용: 강체 설정 - 실행 결과

- 콘텐츠를 재생 후 키보드를 통해 오브젝트 "Tank"를 제어할 때, 기둥에 부딪히면 멈추는 것을 확인



스크립트 기반 충돌 처리

3. 스크립트 기반 충돌 처리

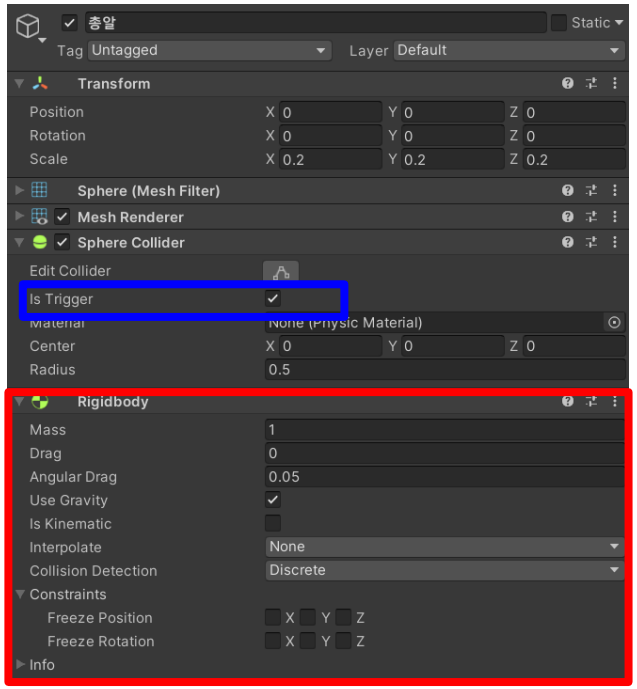
1) 강체의 모드에 따른 물리 엔진

- 강체와 충돌체가 충돌하였을 때, 충돌체의 모드에 따라서 충돌 이벤트가 처리됨
  - ➢ Not Trigger 모드: 유니티에 내장된 물리 엔진을 사용하여 물리 연산 수행
    (예: 중력, 반동 등)
  - ➢ Trigger 모드: 프로그래머가 직접 충돌 이벤트를 처리함
    (예: 박스에 부딪힐 때 텔레포트 기능, 피격시 체력 감소 기능 등)

3. 스크립트 기반 충돌 처리

2) 씬 설정

- 기존에 프리팹 "총알"에서 Bullet Script를 제거 후, Rigidbody Component 추가
- 대포를 발사하여 벽과 부딪혔을 때 충돌 이벤트를 처리하기 위해, Hierarchy > Bullet을 선택 후, **Rigidbody 컴포넌트를 추가**
- 충돌 이벤트를 직접 처리하기 위해 Sphere Collider 컴포넌트의 **Is Trigger 체크**

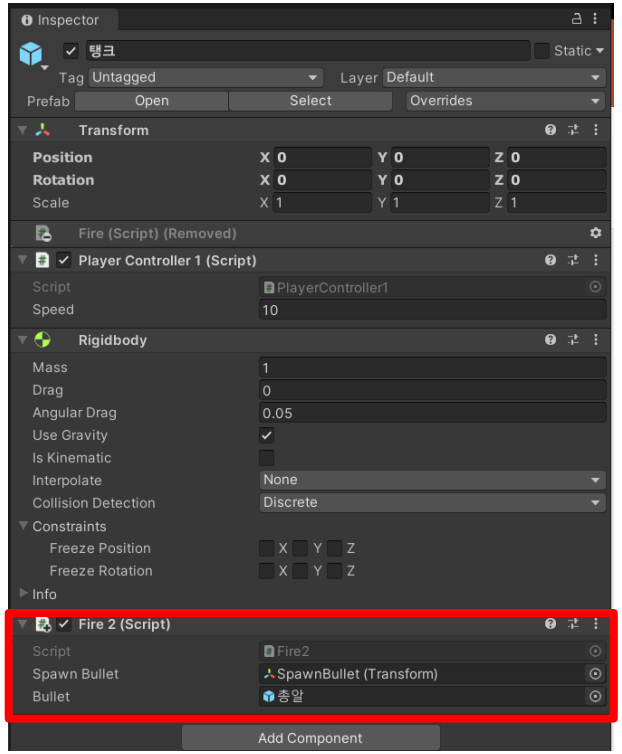


3. 스크립트 기반 충돌 처리

3) 강체에서의 이벤트 처리

- 프로젝트 창 > Assets > Scripts > 새로운 스크립트 "Fire2" 생성
- "Fire"스크립트 제거 후, "Fire2" 스크립트를 오브젝트 "탱크"에 적용

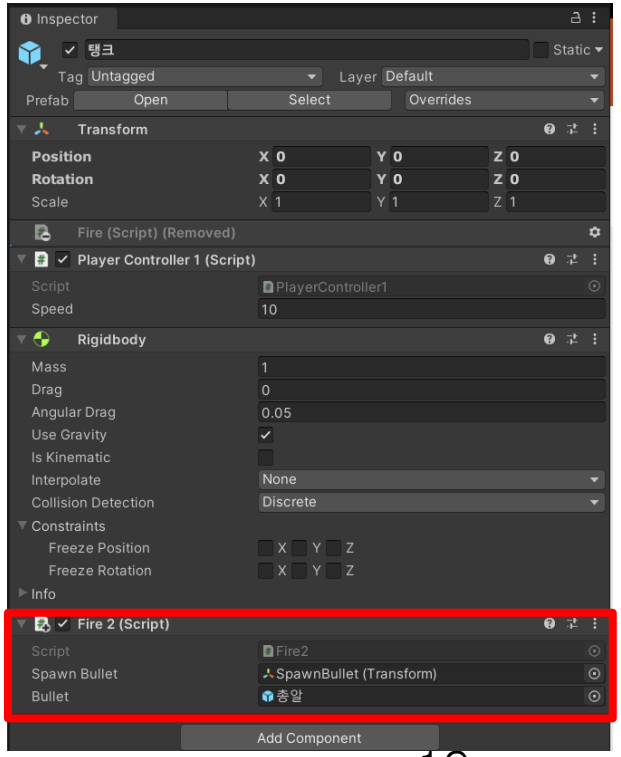
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity 스크립트(자산 참조 1개)|참조 0개
public class Fire2 : MonoBehaviour
{
    private int bulletPower = 1000;
    public Transform spawnBullet;
    public GameObject bullet;

    Unity 메시지|참조 0개
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Space))
        {
            //총알 생성
            GameObject bullet_Instance = Instantiate(bullet, spawnBullet.position, spawnBullet.rotation);
            //생성된 총알의 Rigidbody 컴포넌트 불러옴
            Rigidbody bullet_Rigidbody = bullet_Instance.GetComponent<Rigidbody>();
            //물리적인 힘을 부여
            bullet_Rigidbody.AddForce(bullet_Rigidbody.transform.forward * bulletPower);
        }
    }
}
```



3. 스크립트 기반 충돌 처리

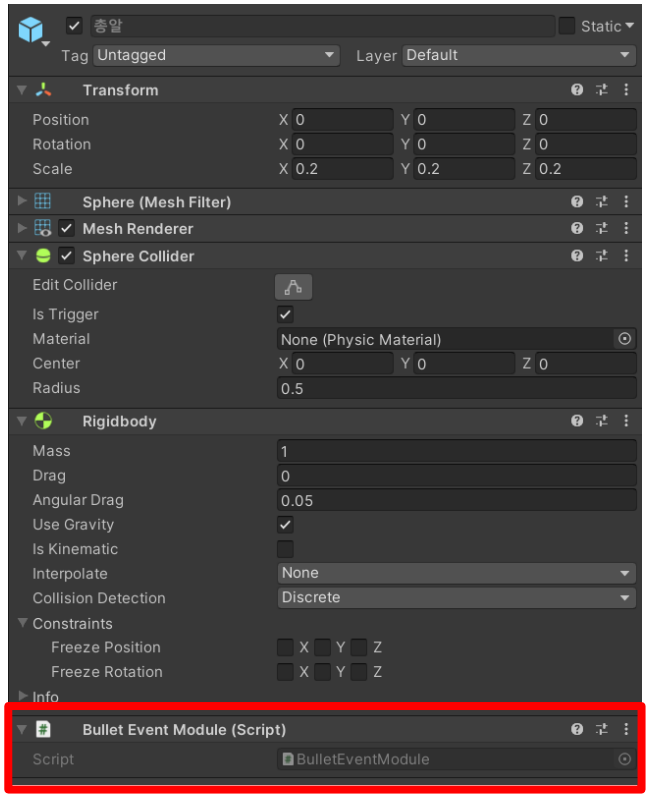
3) 강체에서의 이벤트 처리

- Hierarchy > 오브젝트 "탱크" 선택 후, Fire 스크립트에 다음과 같이 설정
  - ➢ Spawn Bullet 변수 = Hierarchy > 탱크 > 포탑> 포신 > SpawnBullet
  - ➢ Bullet 변수 = 프로젝트 창 > Assets > Prefabs > Bullet



3) 강체에서의 이벤트 처리

- 프로젝트 창 > Assets > Scripts > 새로운 스크립트 "BulletEventModule" 생성
- **프리팹 “총알"을 열은 뒤, 스크립트 "BulletEventModule"을 적용**



3) 강체에서의 이벤트 처리

- 스크립트 "BulletEventModule"을 더블 클릭하여, 코드 편집기를 실행함
- 강체에서의 충돌 이벤트는 다음과 같은 3가지 함수를 통해 처리할 수 있음

함수	파라미터	용도
void OnTriggerEnter();	Collider (충돌체)	강체가 충돌체와 충돌이 시작할 때를 감지
void OnTriggerStay();	Collider (충돌체)	강체가 충돌체와 충돌 중인 상태를 감지
void OnTriggerExit();	Collider (충돌체)	강체가 충돌체와 충돌이 종료될 때를 감지

3) 강체에서의 이벤트 처리

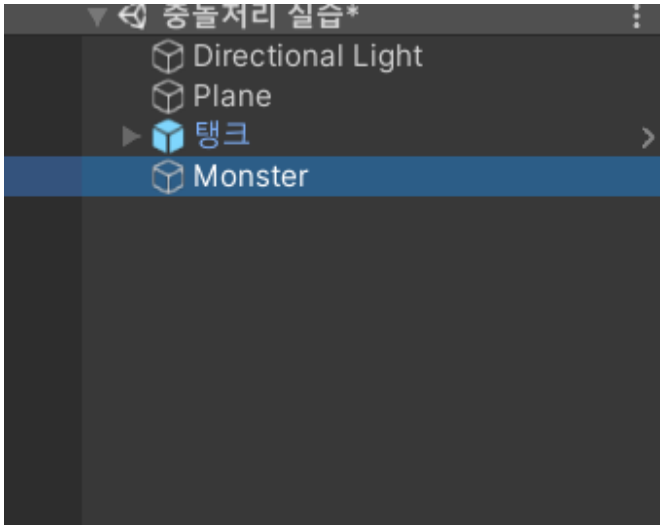
- 스크립트 "BulletEventModule"을 다음과 작성

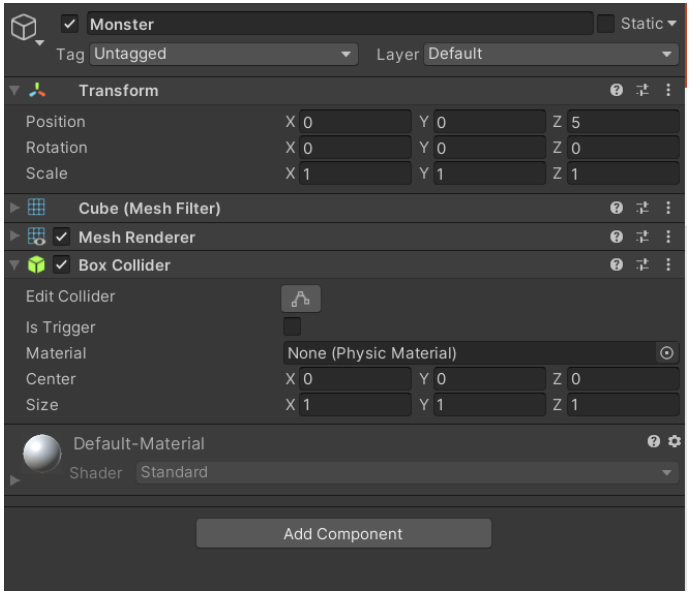
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

Unity 스크립트(자산 참조 1개)|참조 0개
public class BulletEventModule : MonoBehaviour
{

    Unity 메시지|참조 0개
    private void OnTriggerEnter(Collider other)
    {
        //충돌한 오브젝트의 이름이 Monster일 경우
        if (other.gameObject.name == "Monster")
        {
            //충돌한 오브젝트를 파괴
            Destroy(other.gameObject);
            //자기 자신도 파괴
            Destroy(this.gameObject);
        }
    }
}
```

3. 스크립트 기반 충돌 처리

3) 강체에서의 이벤트 처리

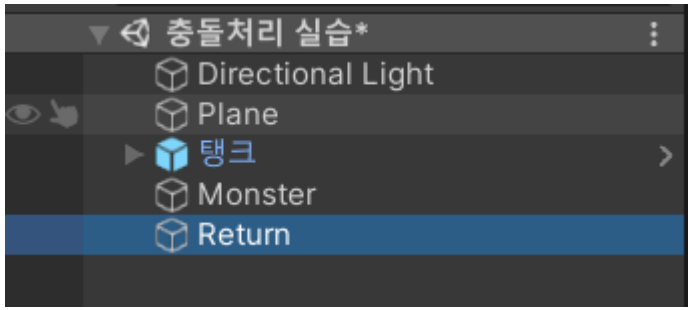
- Hierarchy > 마우스 우 클릭 > 3D Object > Cube를 선택함
- Cube 이름을 "Monster"로 변경
  - **Position: (0, 0, 5)**
  - **Rotation: (0, 0, 0)**
  - **Scale: (1, 1, 1)**

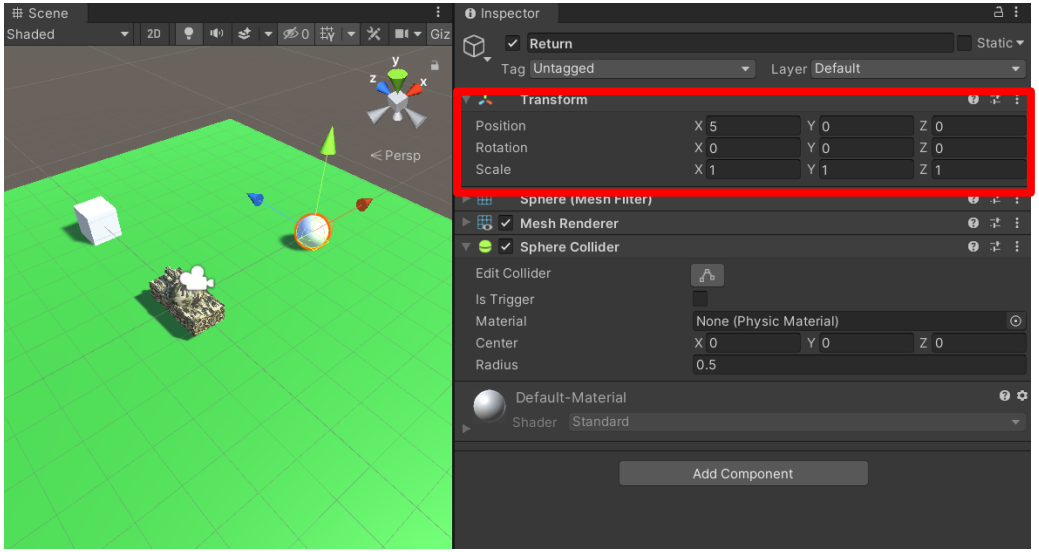




4) 충돌체에서의 이벤트 처리

- Hierarchy > 마우스 우 클릭 > 3D Object > **Sphere를 선택 후 다음과 같이 설정**
  - ➢ 이름: Return
  - ➢ Transform > Position: (5, 0, 0)
  - ➢ Transform > Rotation: (0, 0, 0)
  - ➢ Transform > Scale: (1, 1, 1)

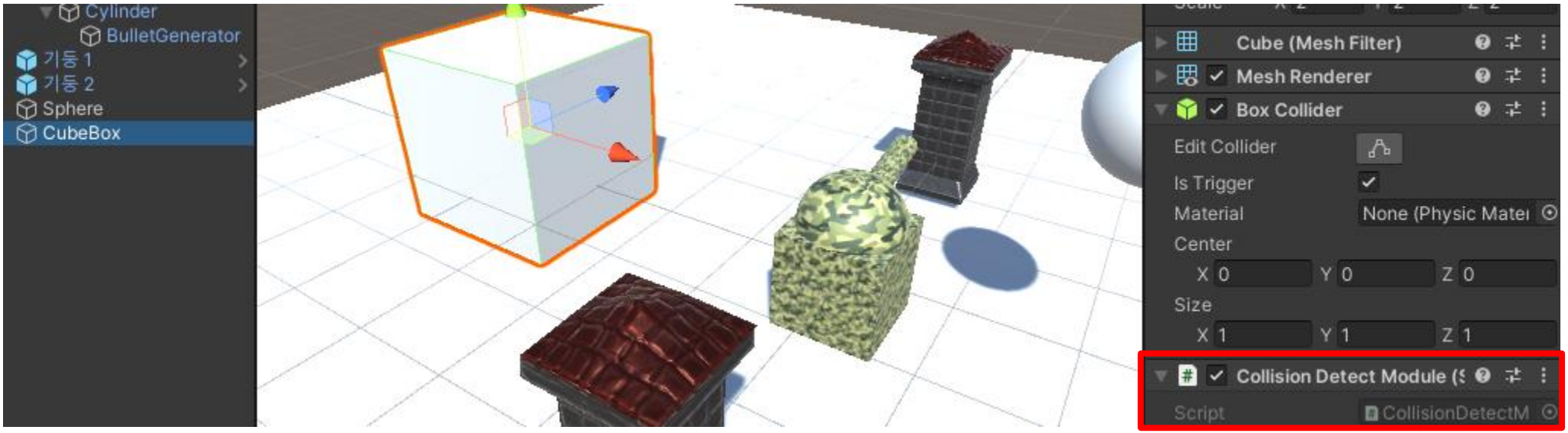




3. 스크립트 기반 충돌 처리

4) 충돌체에서의 이벤트 처리

- 프로젝트 창 > Assets > Scripts > 새로운 스크립트 "CollisionDetectModule" 생성
- **스크립트 "CollisionDetectModule"를 오브젝트 “Return"에 적용**



4) 충돌체에서의 이벤트 처리

- 스크립트 "CollisionDetectModule"을 더블 클릭하여, 코드 편집기를 실행함
- 충돌체에서의 충돌 이벤트는 다음과 같은 3가지 함수를 통해 처리할 수 있음

함수	파라미터	용도
void OnCollisionEnter();	Collision (충돌 정보)	충돌체로 강체가 들어올 때를 감지
void OnCollisionStay();	Collision (충돌 정보)	충돌체에 강체가 머물 때를 감지
void OnCollisionExit();	Collision (충돌 정보)	충돌체로부터 강체가 나갈 때를 감지

4) 충돌체에서의 이벤트 처리

- 스크립트 "CollisionDetectModule"을 다음과 작성

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

Unity 스크립트|참조 0개
public class CollisionDetectModule : MonoBehaviour
{
    Unity 메시지|참조 0개
    private void OnCollisionEnter(Collision collision)
    {
        //충돌한 게임 오브젝트가 탱크일 경우
        if(collision.gameObject.name == "탱크")
        {
            //충돌한 게임 오브젝트의 위치를 원점으로 이동
            collision.transform.position = Vector3.zero;
        }
    }
}
```