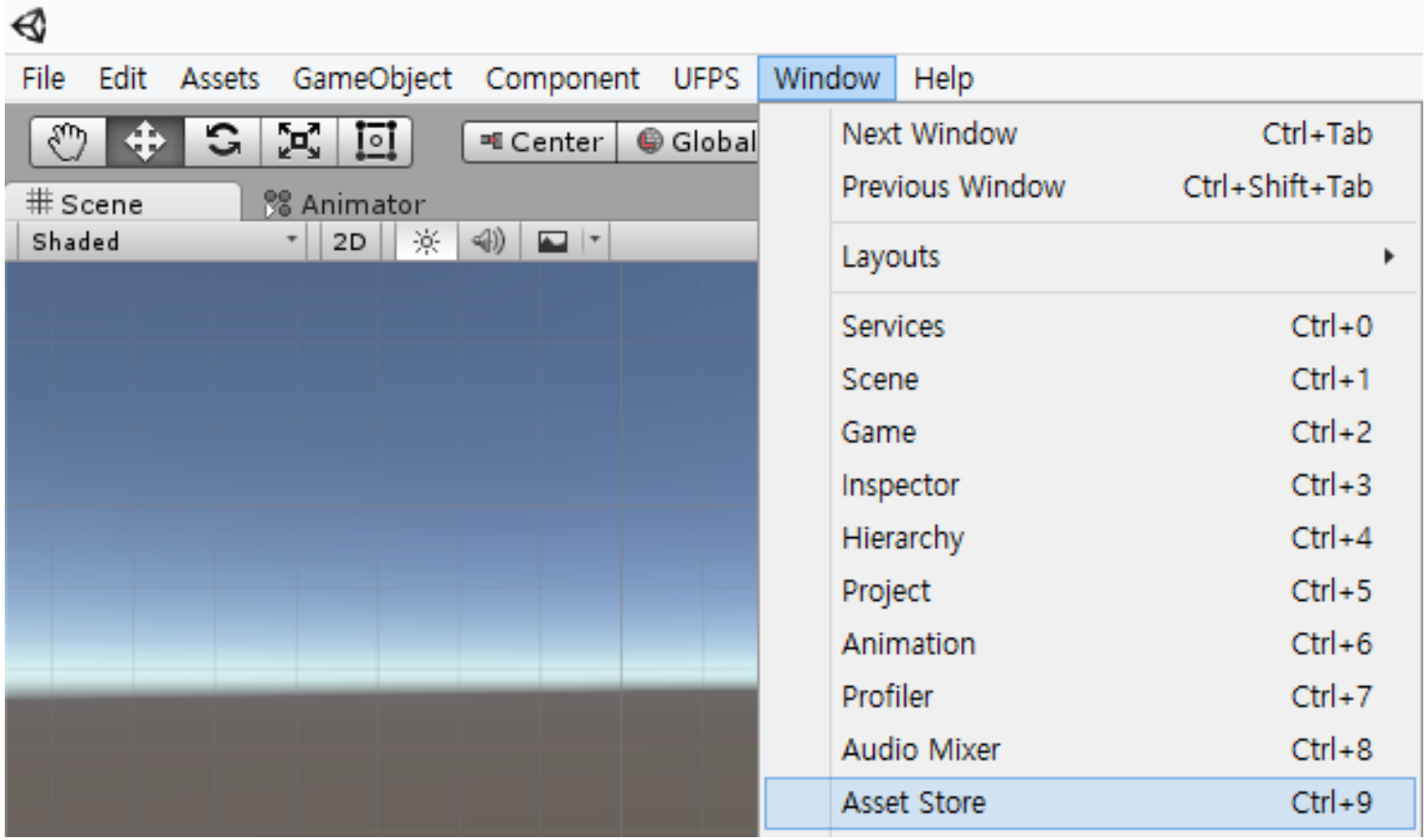
Unity – Fire Ball

NHN NEXT
서형석

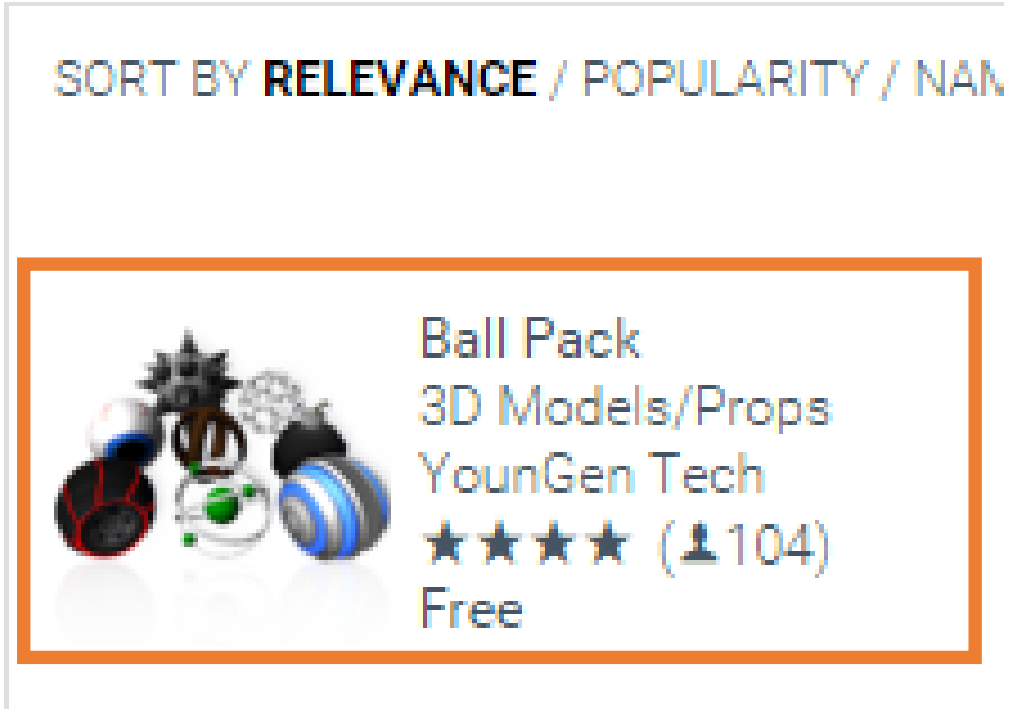
Unity

- Asset Import
  : Asset Store에서 **Ball Pack** 획득



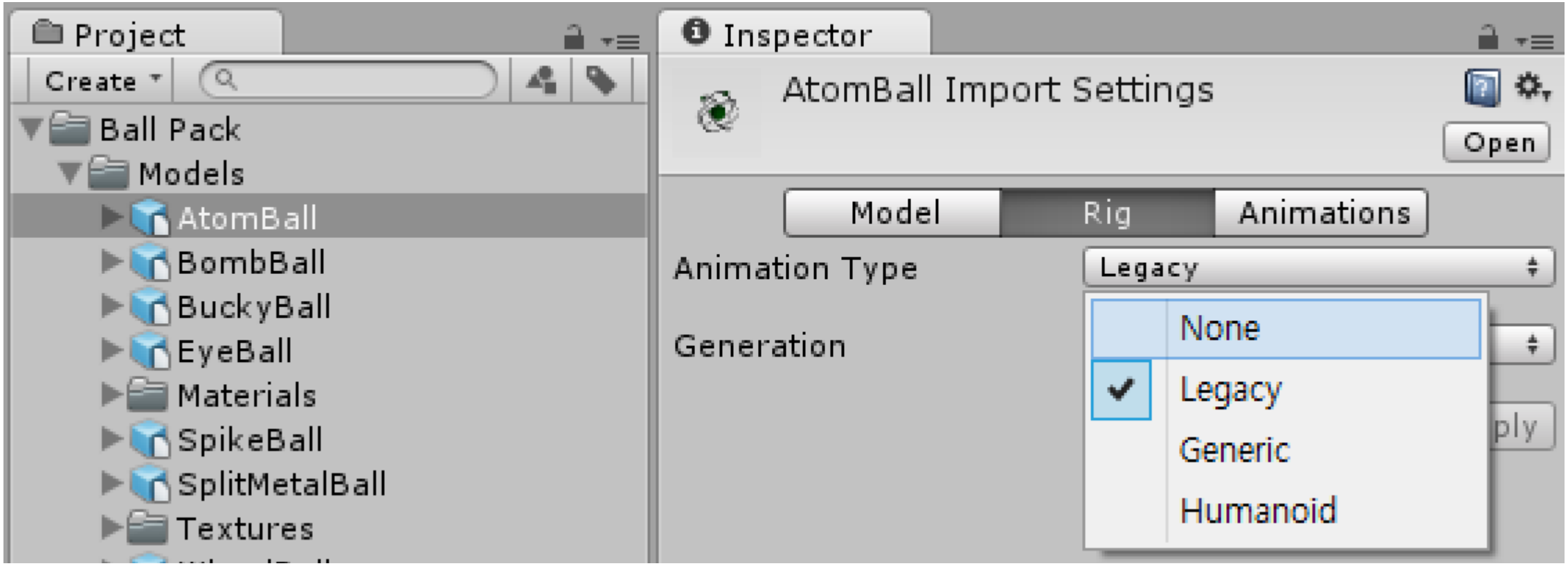
Unity

- Asset Import
  : Asset Store에서 Ball Pack 획득



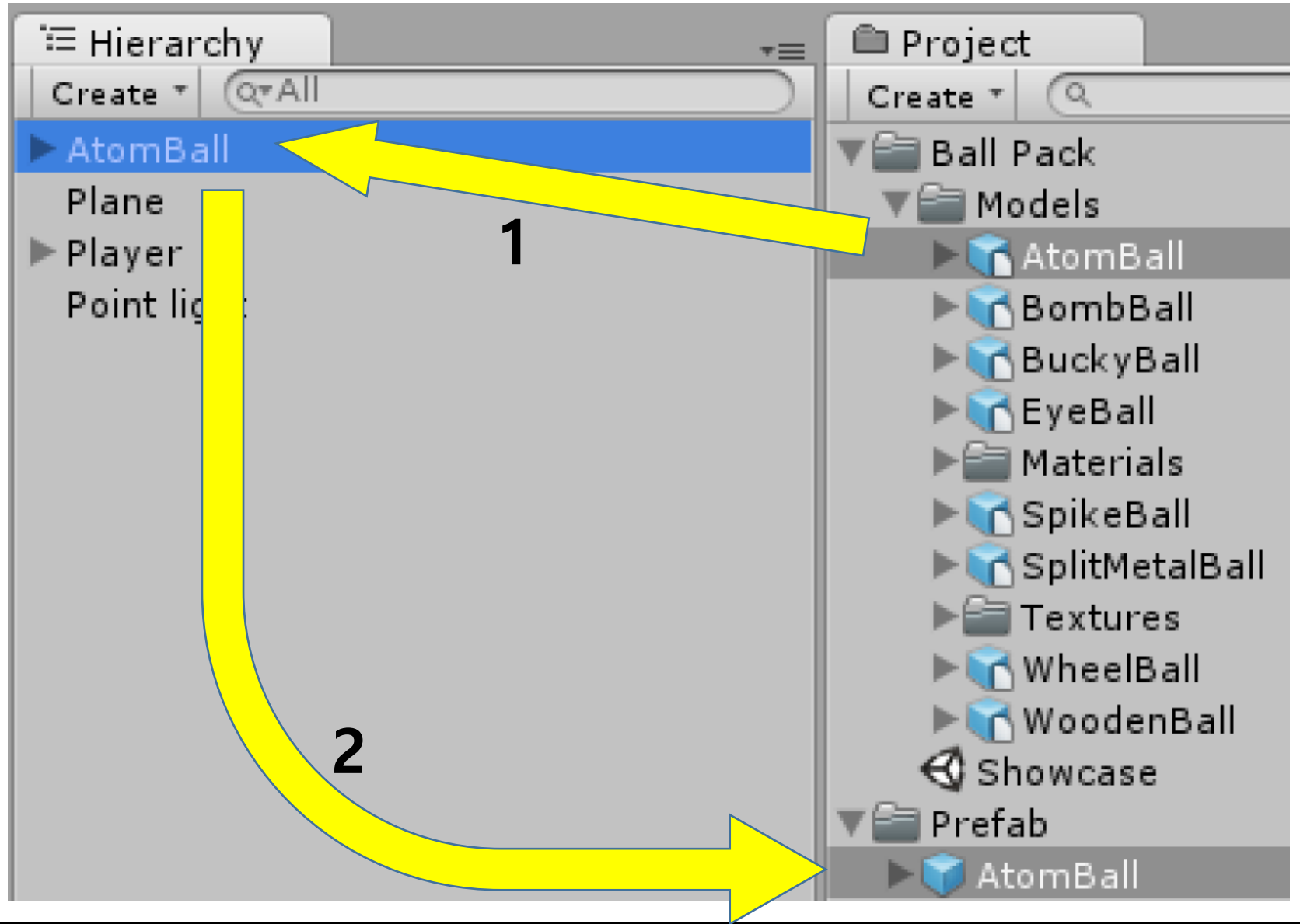
Unity

- 3d Object Import Setting
  : 불 필요한 정보를 제거



Unity

- Prefab 생성



Unity

- 충돌 처리

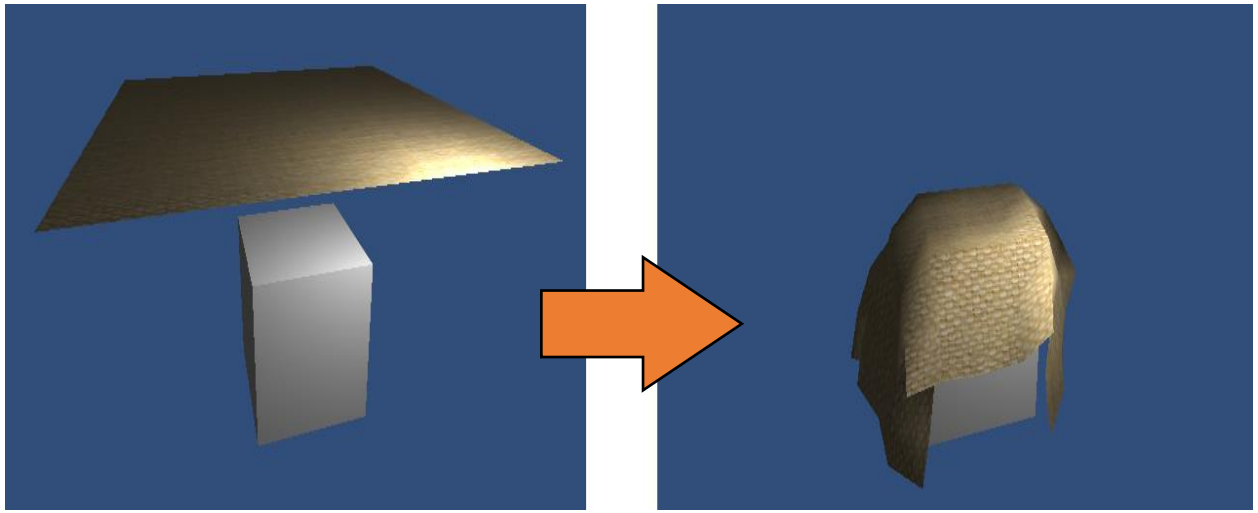
Unity

유니티가 사용하는 물리 시스템

Unity

유니티의 물리 시스템 활용 예시

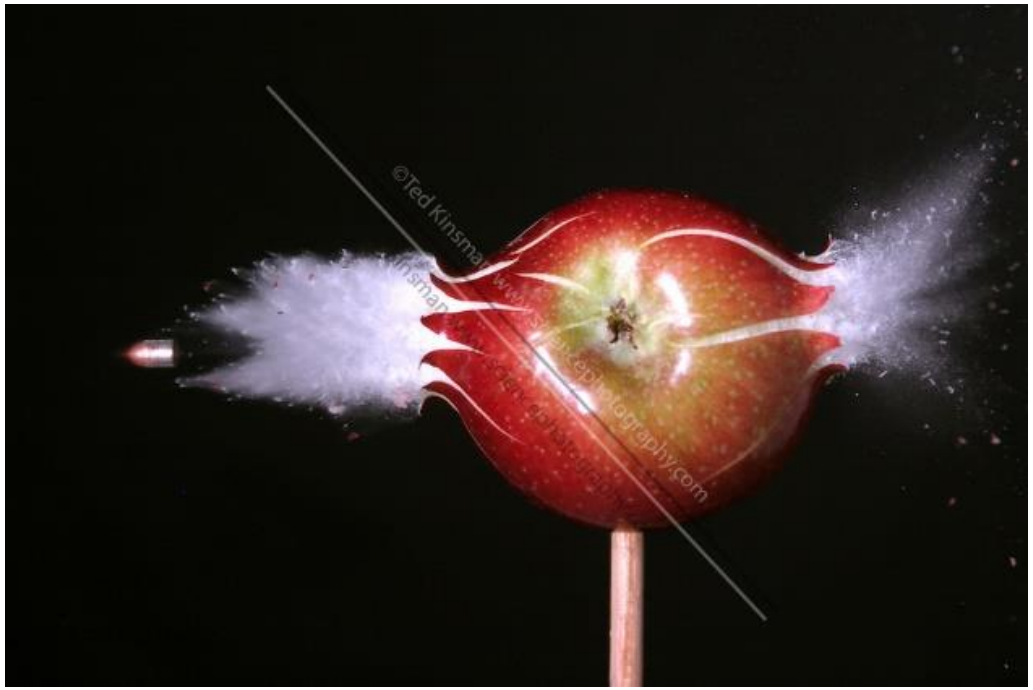
- 사실적인 물리 시스템 사용
  ex) 천 물리, 폭파 등



Unity

유니티의 물리 시스템 활용 예시

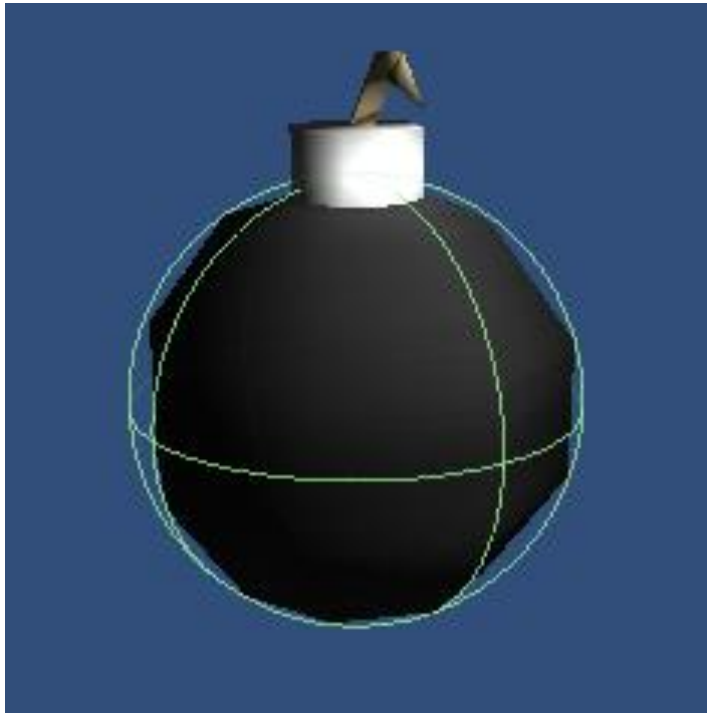
- 게임내 이벤트 처리를 위해
  ex) 총알이 물체와 부딪히는 다양한 처리 : 소리, 이펙트, 피격



Unity

충돌을 검출하기 위한 모양

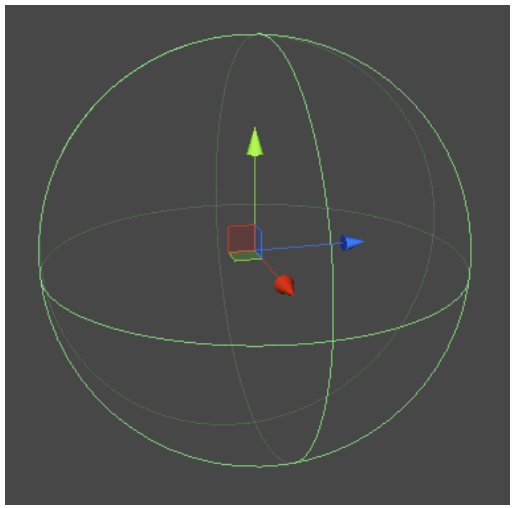
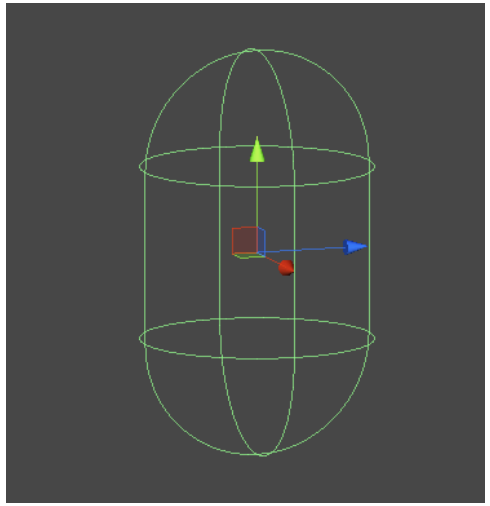
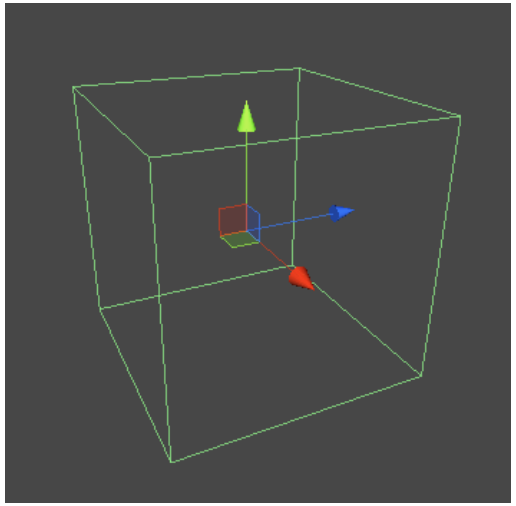
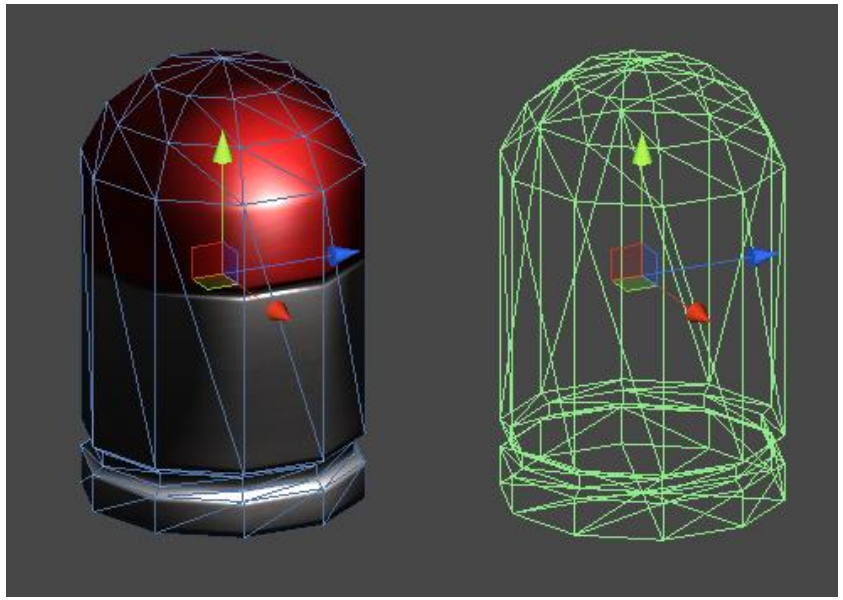
- 충돌체(Collider)



Unity

충돌을 검출하기 위한 모양

- 충돌체(Collider)

Unity

충돌체의 비용

- 충돌 로직에 따라 비용이 다르기 때문에 적절한 충돌영역을 사용해야 함

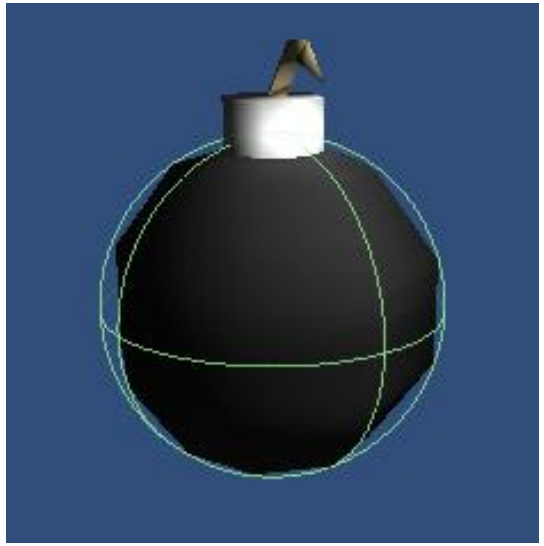
원형(Sphere) < 캡슐(Capsule) < 박스(Box) < 메쉬(Mesh)

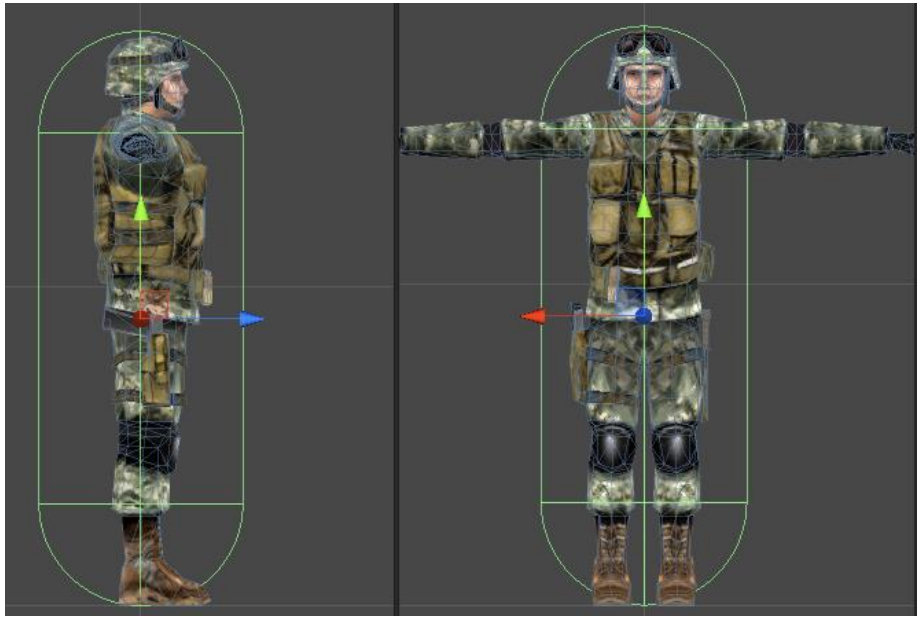
- 되도록 Mesh 충돌체는 사용을 하지 않도록 해야 한다

Unity

충돌을 검출하기 위한 모양의 결정

- 충돌체(Collider)





Unity

문제발생 – 충돌체 설정 후 충돌 이벤트가 발생하지 않는 경우

- 이유 : 충돌하는 주체에
강체(Rigidbody)가 존재하지 않기 때문에

Unity

강체(Rigidbody)란 무엇인가?

- 물리학적 개념.
  외력을 가해도 크기나 형태가 변하지 않는 이상적인 물체를 뜻함.
  물리 엔진에서는 이동, 회전에 관한 운동을 표현하는데 사용.

Unity

강체가 필요한 이유?

- 물리 엔진의 특성
  : 이동, 회전에 관한 운동을 표현하기 위해

- 동적(Dynamic) / 정적(Static) 오브젝트의 구분을 위해
  : 물리 시스템 최적화를 위하여

Unity

Q) 게임에는 움직이는 오브젝트가 많은가?
움직이지 않는 오브젝트가 많은가?

Unity

A) 움직이지 않는 것들이 더 많다.

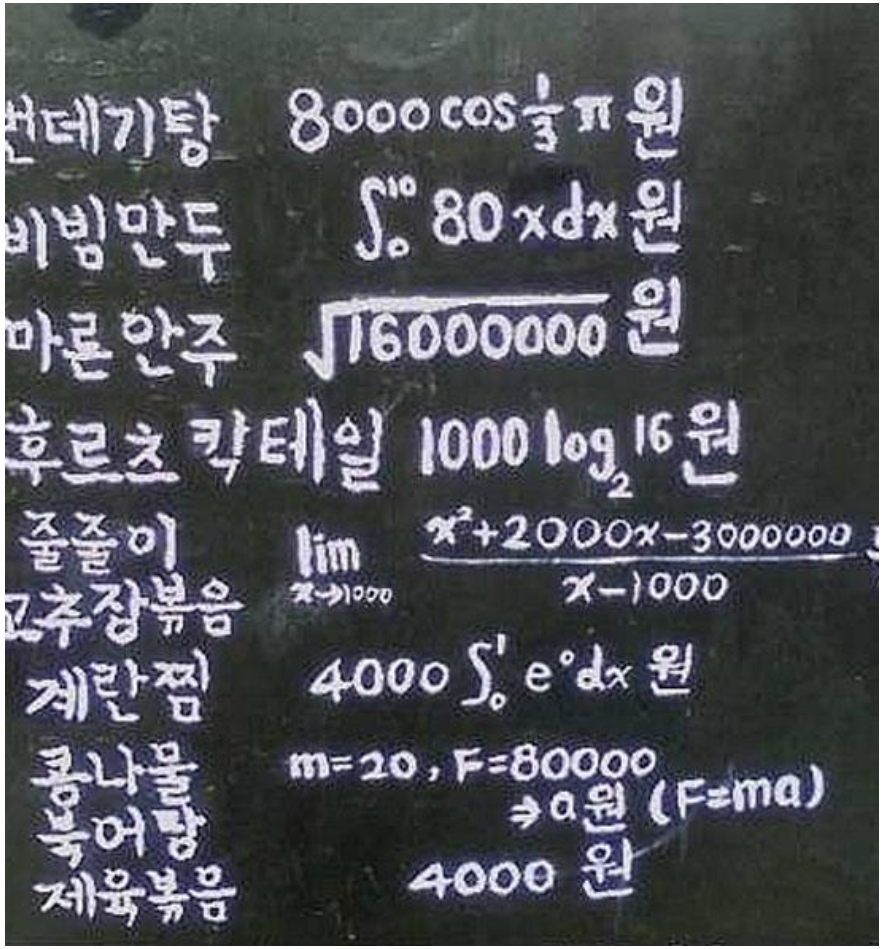
Q) 이것이 게임의 물리 처리에 영향을 끼치는 요소인 이유?

Unity

A) 물리 계산 최적화의 중요한 포인트이기 때문

Unity

포인트 : 물리 계산은 결코 쉽지 않다.(= 비용이 비싸다.)

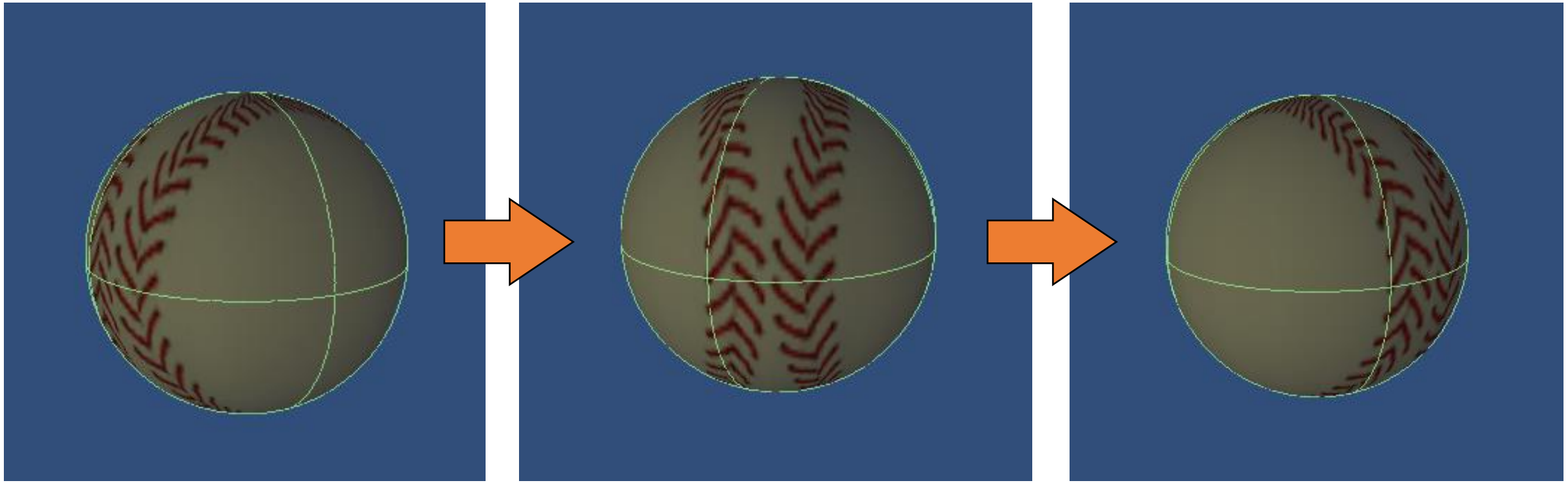


Unity

해법 - 미리 계산할 수 있는 것은 미리 계산하고
실시간 계산할 녀석들은 따로 계산하자!

Unity

생각해 봅시다.



Unity

정리 - 정적(Static) 오브젝트와
동적(Dynamic) 오브젝트의 명확한 구분이 필요

Unity

물리 최적화 방안

- 동적(Dynamic) / 정적(Static) 오브젝트의 구분

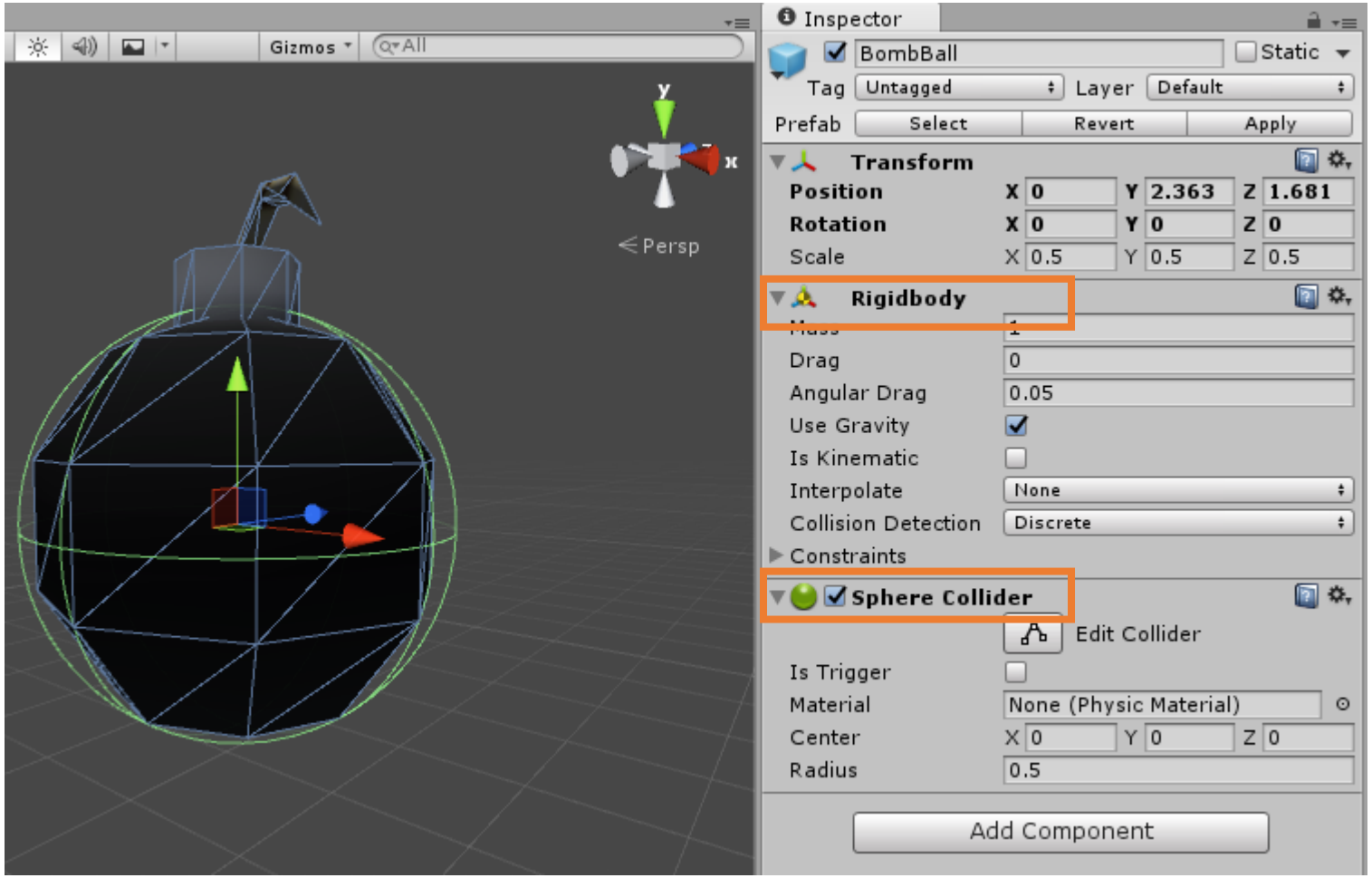
- 물리 갱신 타이밍 조정
  : 게임에 따라 물리 계산 시간을 조정하여 계산량 감소.

Unity

- 포탄 설정

  : Collider + Rigidbody

  scale : 0.5



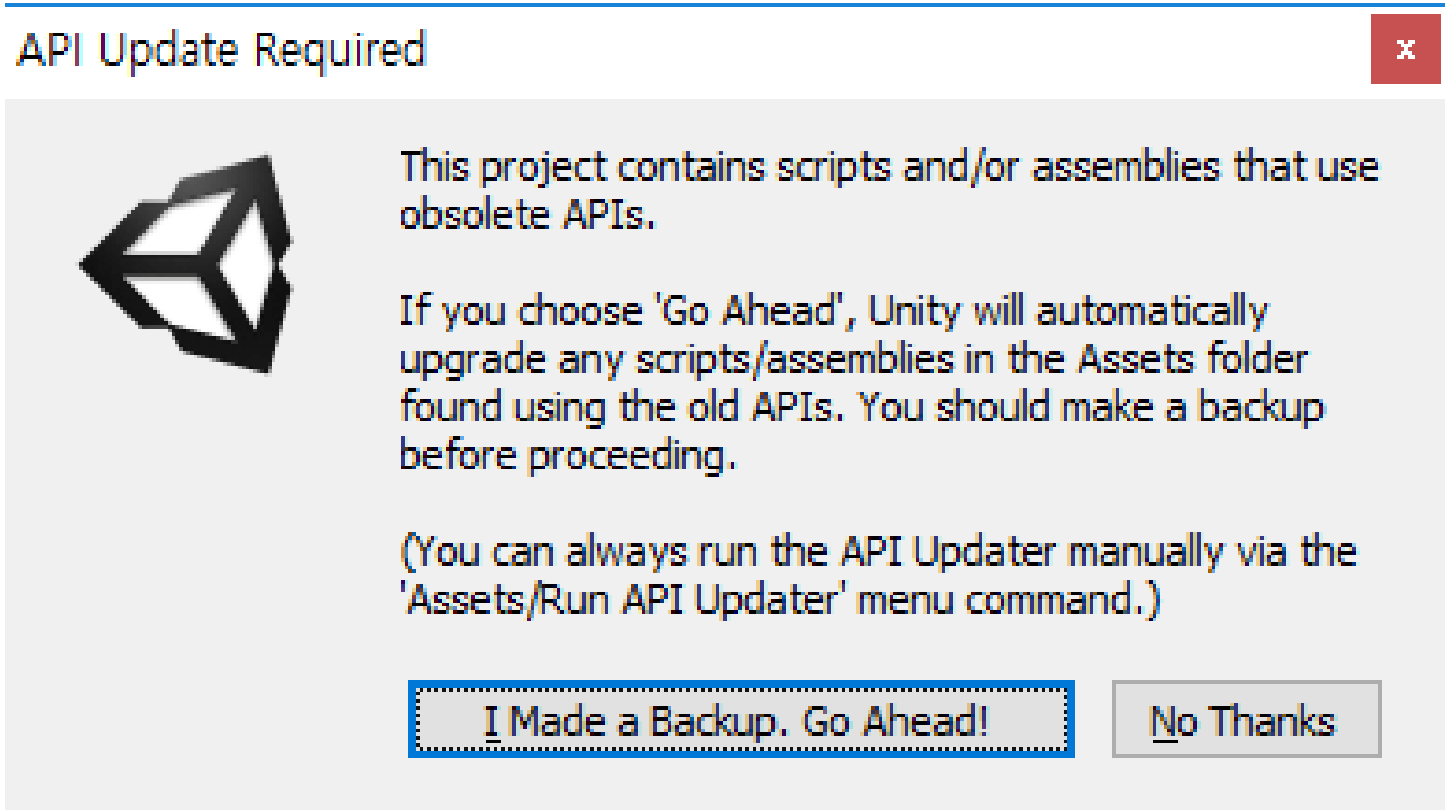
```
public class FireBall : MonoBehaviour
{
    public  Transform  cameraTransform;
    public  GameObject  fireObject;
    public  float  forwardPower = 20.0f;
    public  float  upPower = 1.0f;

    void Update ()
    {
        if(  Input.GetButtonDown("Fire1")  )
        {
            GameObject   obj  =  Instantiate( fireObject ) as GameObject;

            obj.transform.position = transform.position;
            obj.rigidbody.velocity =
                                        (cameraTransform.forward * forwardPower)
                                        + (Vector3.up * upPower);
        }
    }
}
```

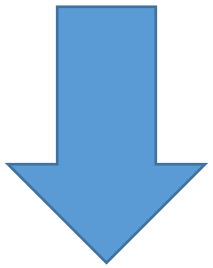
Unity

- Error 확인



Unity

rigidbody



GetComponent< *Rigidbody* >()

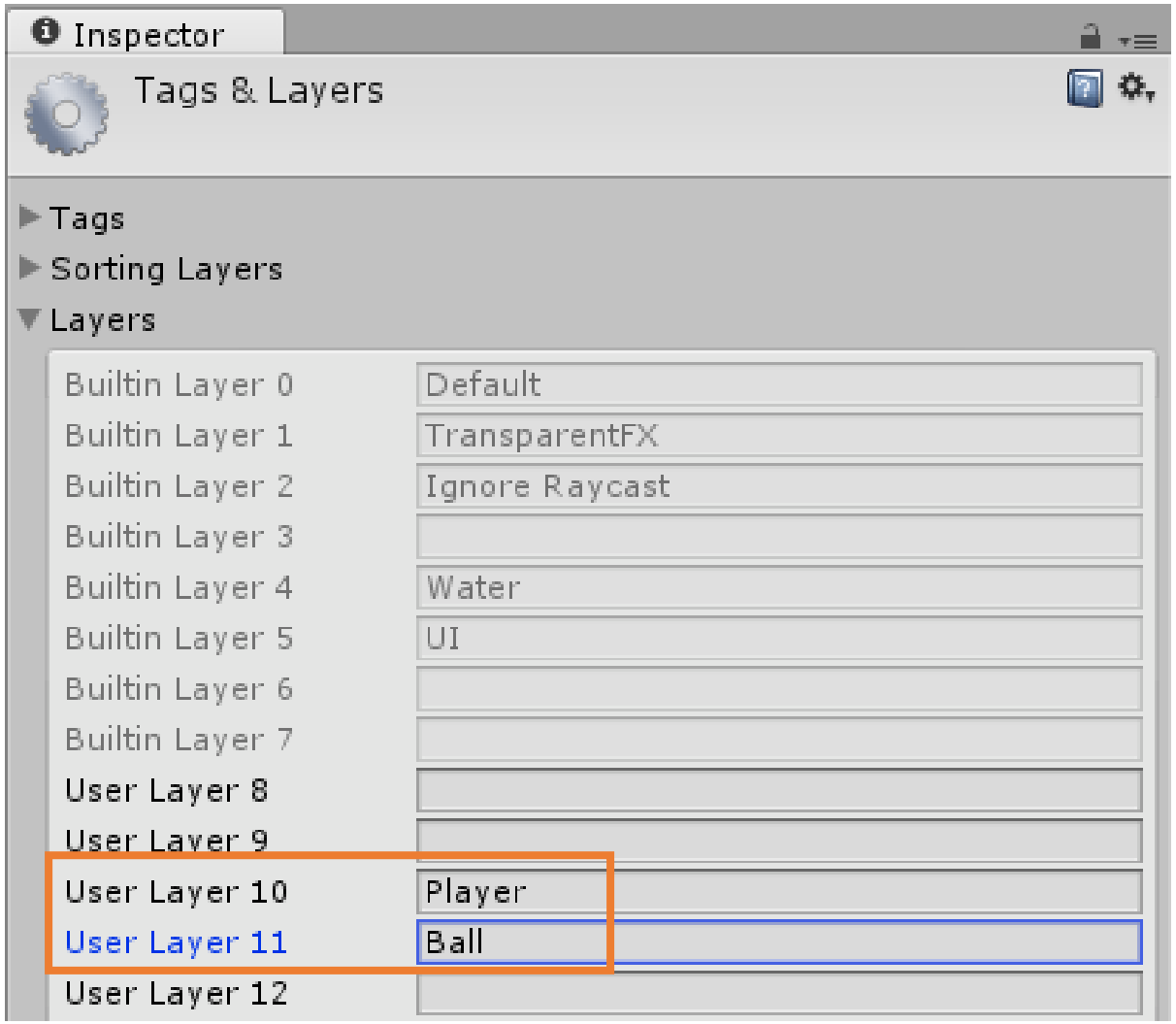
Unity

- 문제점 파악
  : *특정 방향으로는 잘 날아가지만*
  특정 방향으로는 날아가지 못한다.
  또한 캐릭터가 발사할 때 마다 튀어오른다.
  왜 그럴까?

  [Hint]
  1. Scene 뷰를 잘 관찰해볼 것

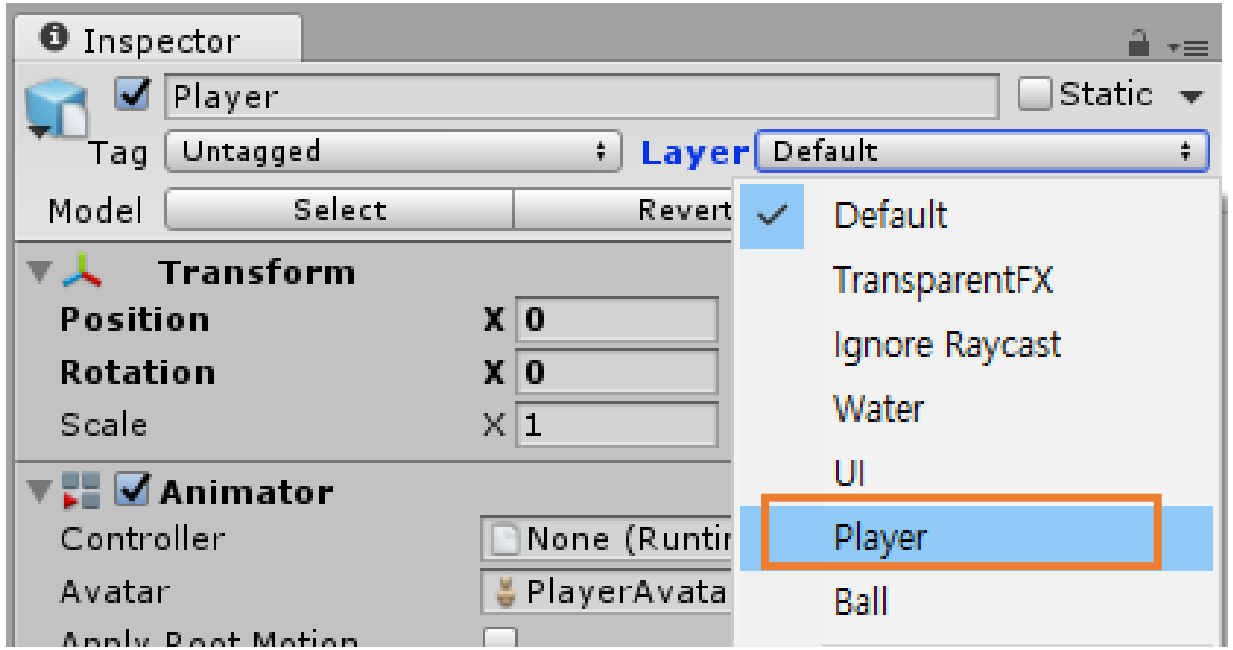
Unity

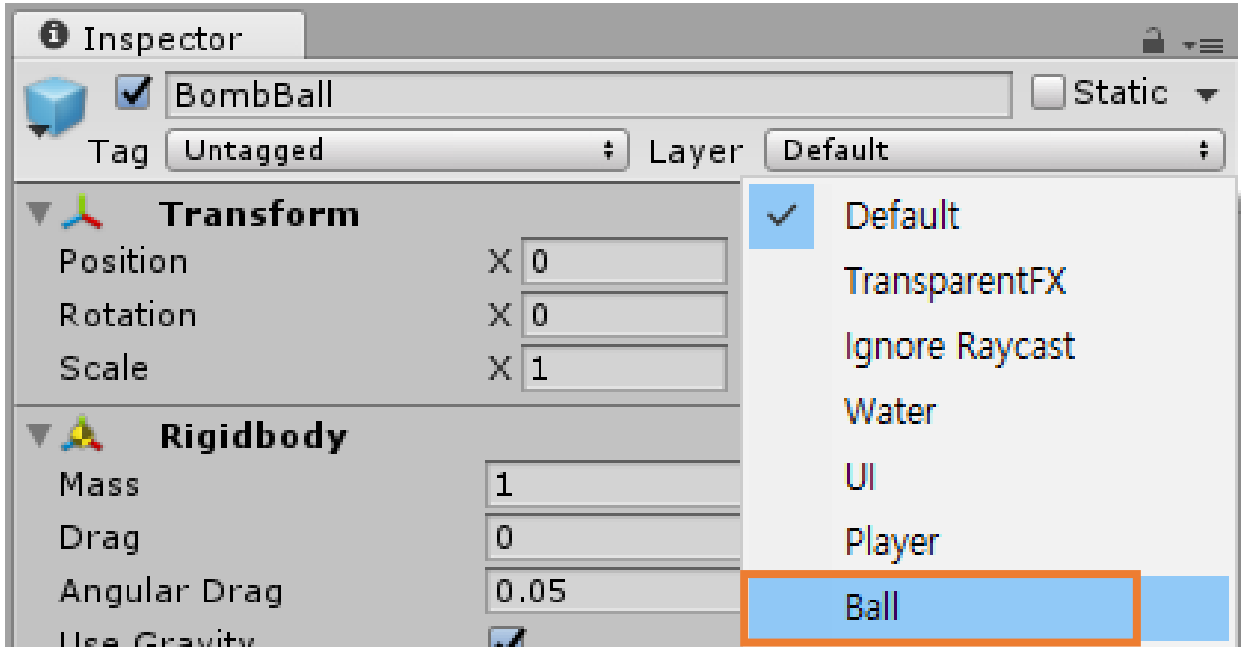
- Layer를 통한 구분



Unity

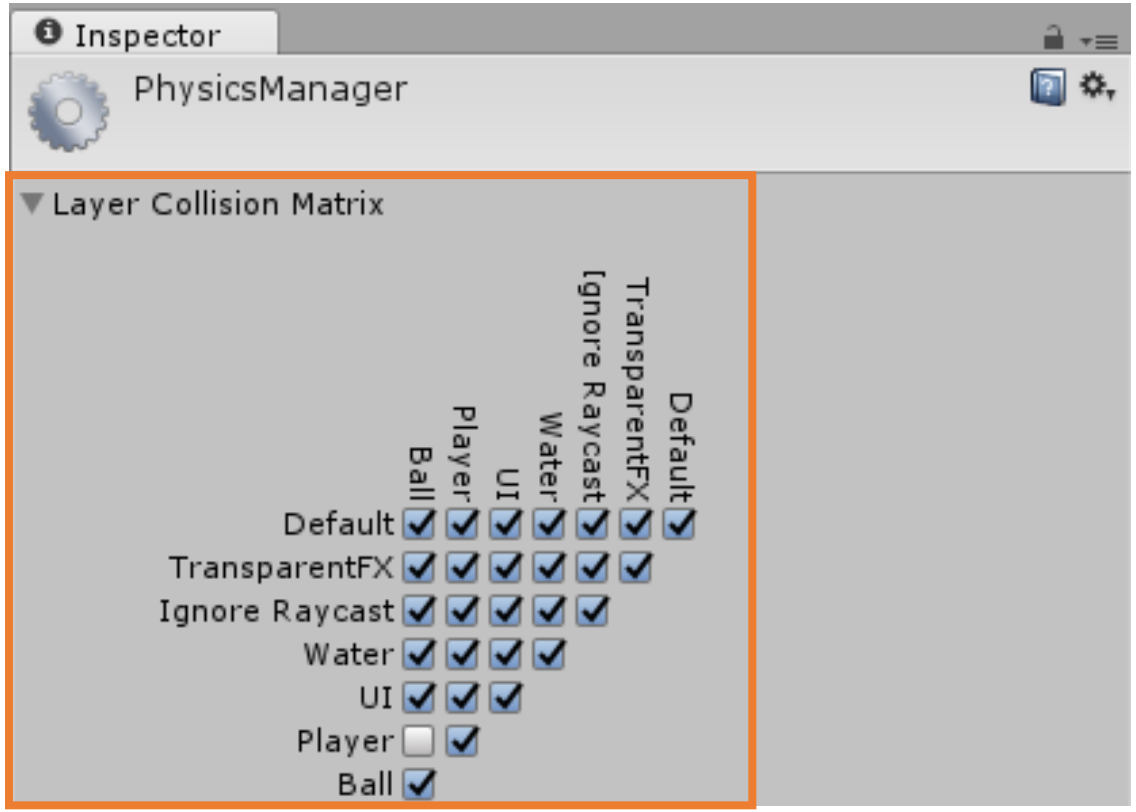
- 각 오브젝트에 적용





Unity

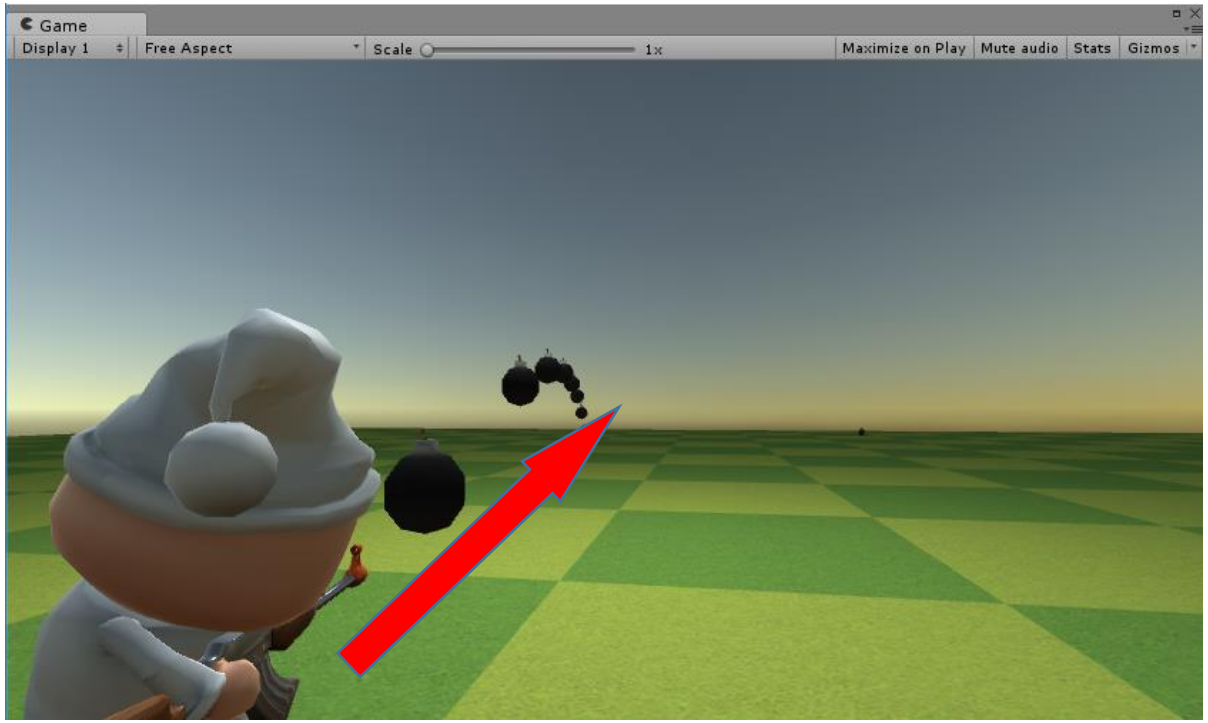
- Physics Collision Layer Matrix



Unity – 컴퍼넌트 구현

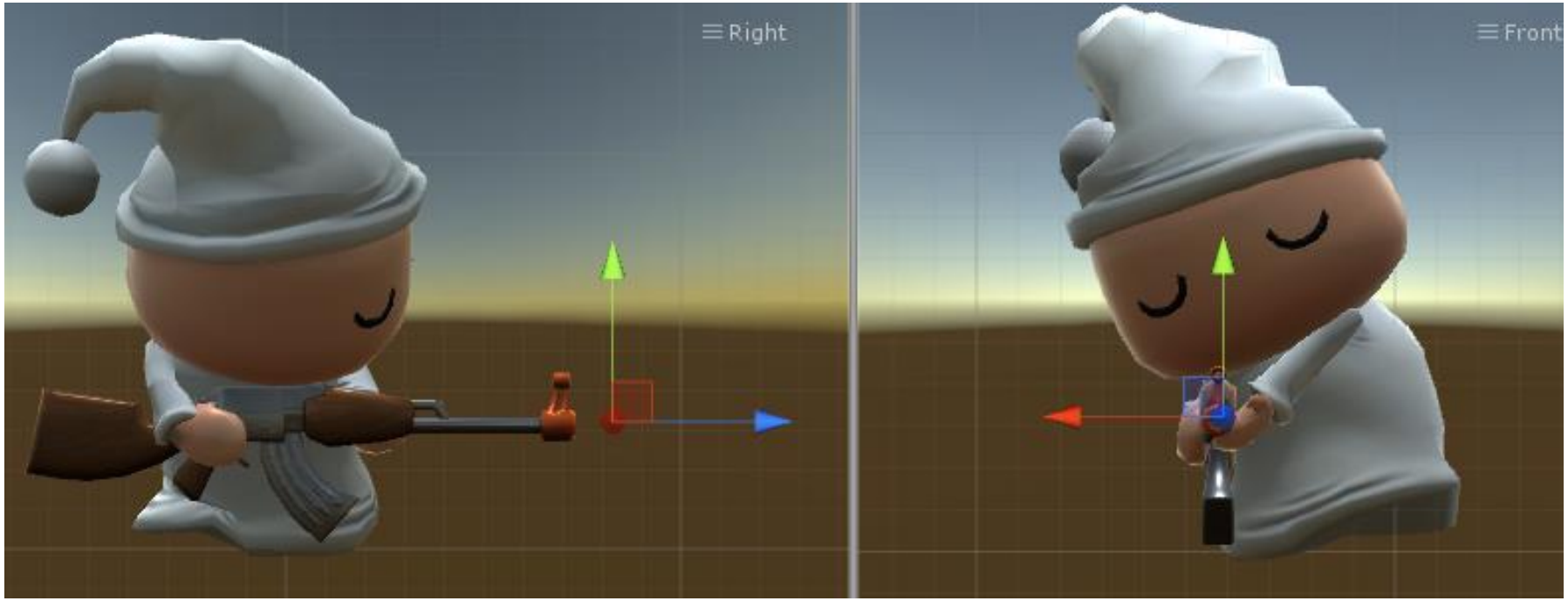
- 포탄이 날아가는 시작 위치가 화면의 정 중앙이기 때문에 이상함을 느낄 수 있다.

- FPS 게임처럼
  총구에서 포탄이 날아가도록 구현해 보자.



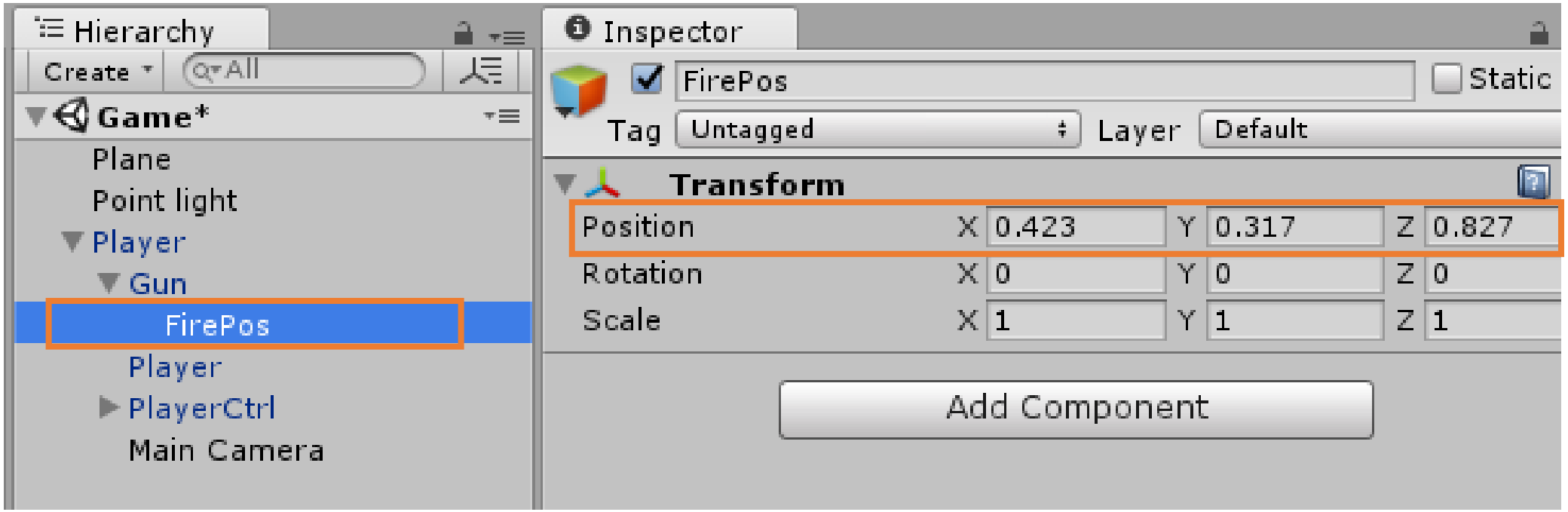
Unity

- dummy object의 position 설정



Unity

- dummy object의 position 설정



Unity

- dummy object의 position 활용

```
public class GunFire : MonoBehaviour
{
    public  Transform  cameraTransform;
    public  GameObject  fireObject;
    public  float  forwardPower = 20.0f;
    public  float  upPower = 5.0f;

    public  Transform   firePosTransform;

    void Update ()
    {
        if(  Input.GetButtonDown("Fire1")  )
        {
            GameObject   obj  =  Instantiate( fireObject ) as GameObject;

            obj.transform.position = firePosTransform.position;
            obj.GetComponent<RigidBody>().velocity = cameraTransform.forward * forwardPower + Vector3.up * upPower;
        }
    }
}
```

Unity

- 문제점 파악 : 포탄이 날아가는 도중에 캐릭터를 움직이면
  포탄이 끊기는 느낌이 난다. 왜 그럴까?

[Hint]

1. 포탄의 동작 방식과 움직임의 차이점

Unity

- QUIZ : 포탄이 초기 발사될 때 회전이 없어 밋밋하다.
  무작위로 회전하며 날아가도록 하자.

  [Hint]

  1. 강체