인터랙티브 콘텐츠 응용
(Chapter 4)

Jin-Mo Kim
jinmo.kim@hansung.ac.kr

Tower Game

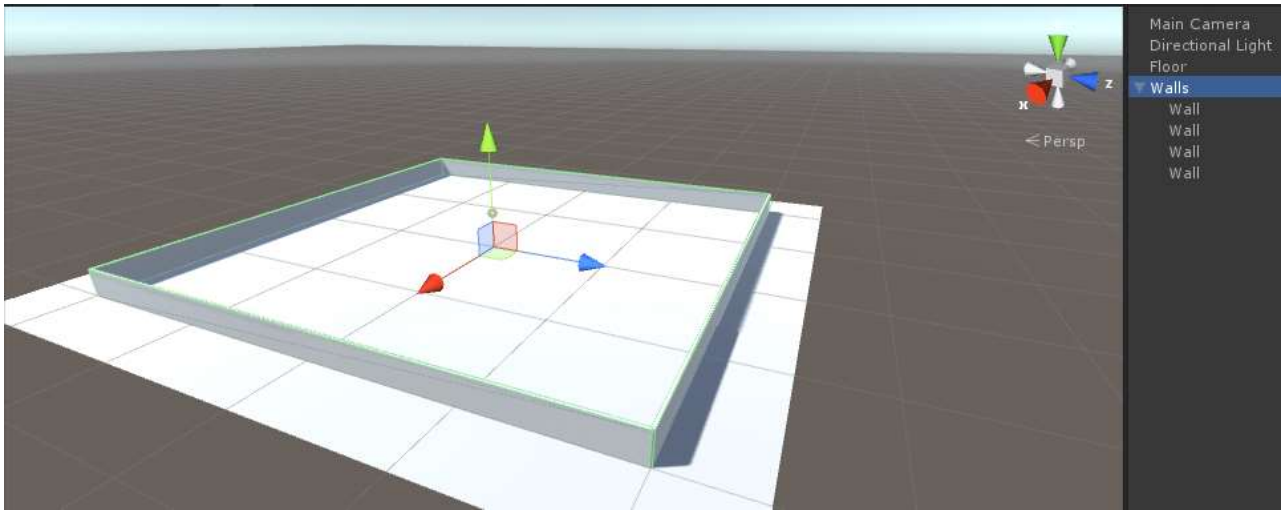
- Unity3D의 응용 기능
  - Prefab : 동일한 객체의 반복적 사용
  - Rigidbody : 물리 기반의 움직임
  - Collider : 충돌처리를 위한 콤포넌트
  - Intantiate : Prefab 객체의 복사
  - Collision / Trigger : 충돌과 관통

Tower Game

- 무대 설정
  - GameObject → Plane 생성
    - name : Floor
    - Scale : (5, 1, 5)
  - GameObject → Cube 생성
    - name : Wall
    - Ctrl + D → 4개 복사
  - GameObject → Create Empty
    - name : Walls
    - Wall 객체들을 그룹으로 설정

Tower Game

- 무대 설정



Transform

Position	X 0	Y 1	Z -20
Rotation	X 0	Y 0	Z 0
Scale	X 40	Y 2	Z 0.2

Transform

Position	X 0	Y 1	Z 20
Rotation	X 0	Y 0	Z 0
Scale	X 40	Y 2	Z 0.2

Transform

Position	X 20	Y 1	Z 0
Rotation	X 0	Y 90	Z 0
Scale	X 40	Y 2	Z 0.2

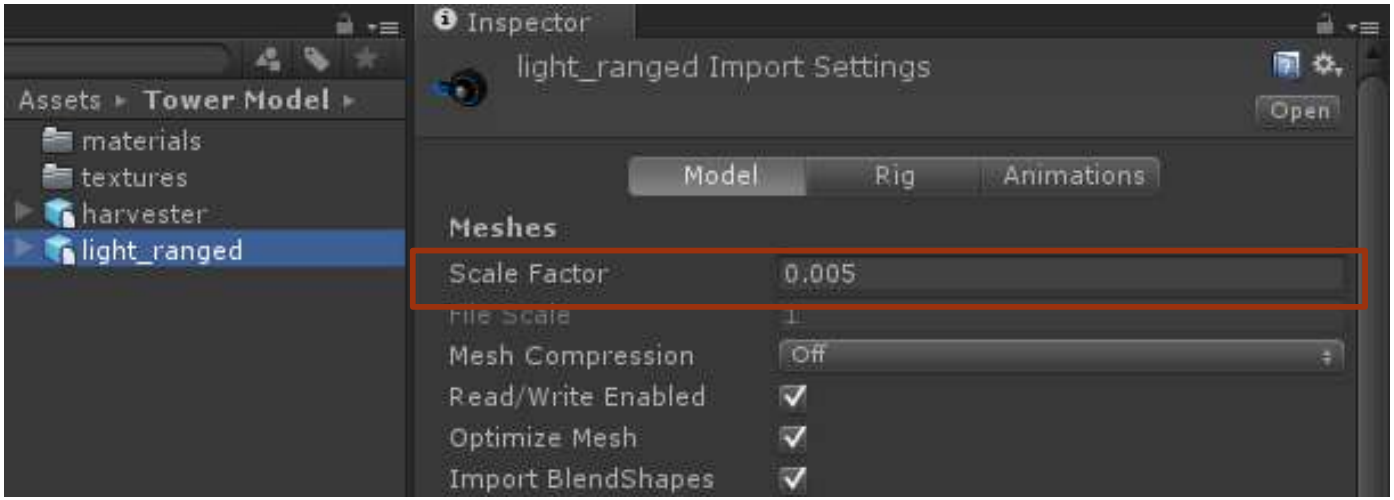
Transform

Position	X -20	Y 1	Z 0
Rotation	X 0	Y 90	Z 0
Scale	X 40	Y 2	Z 0.2

Scale Factor → 객체크기를 변화시키는 시점은 불러오기 전에

Tower Game 파일을 클릭하여 하는 것임

- 타워 캐릭터 배치
  - Tower Model 폴더를 Asset 폴더 안에 복사
  - .fbx 모델의 크기를 조절
    - Inspector → Scale Factor : 0.005

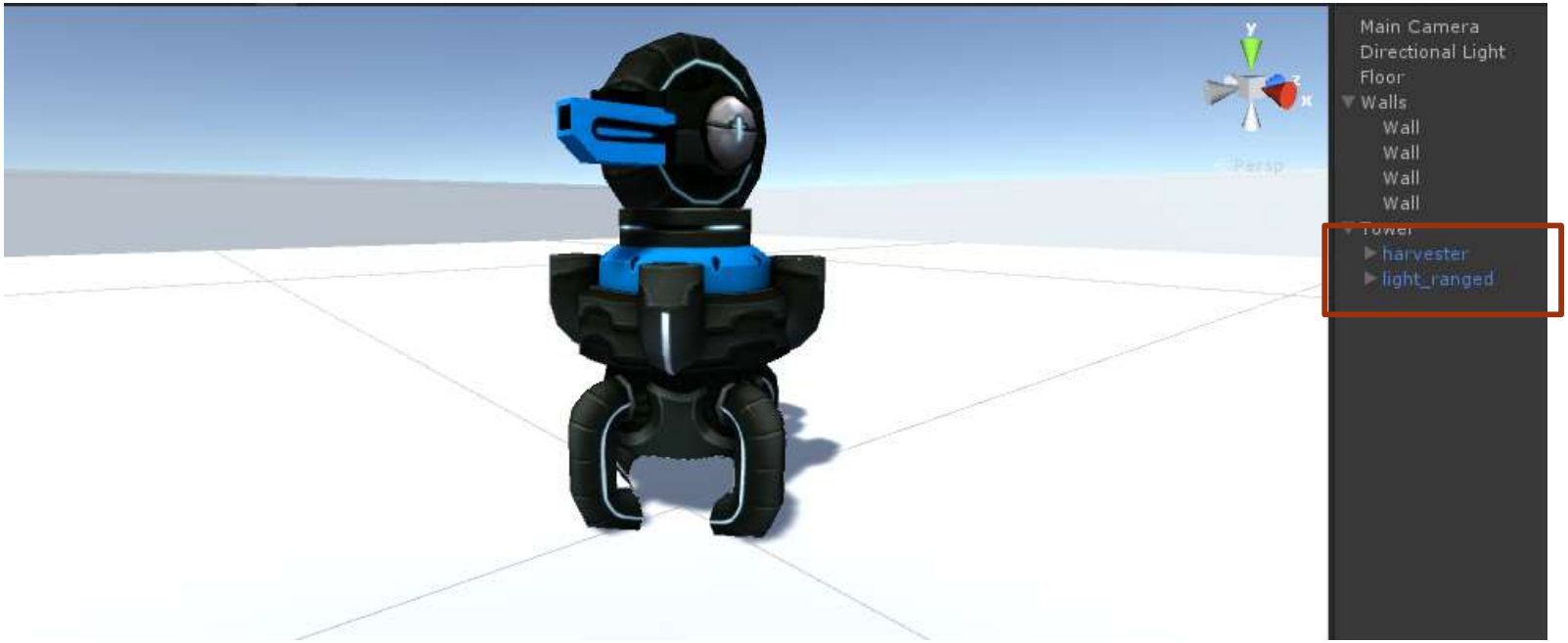


Tower Game

- 타워 캐릭터 배치
  - 모델 위치 조절
    - harvester : position(0, 0.3, 0)
    - light_ranged : position(0, 1, 0)
      : rotation(0, 180, 0)
  - GameObject → CreateEmpty
    - name : Tower
    - harvester, light_ranged 객체를 자식으로 등록

Tower Game

- 타워 캐릭터 배치
  - Tower → position : (5, 0, 0)
    rotation : (0, -90, 0)



pivot : 디자이너가 설정하는 객체 중심의 좌표
local : 객체 기준 회전

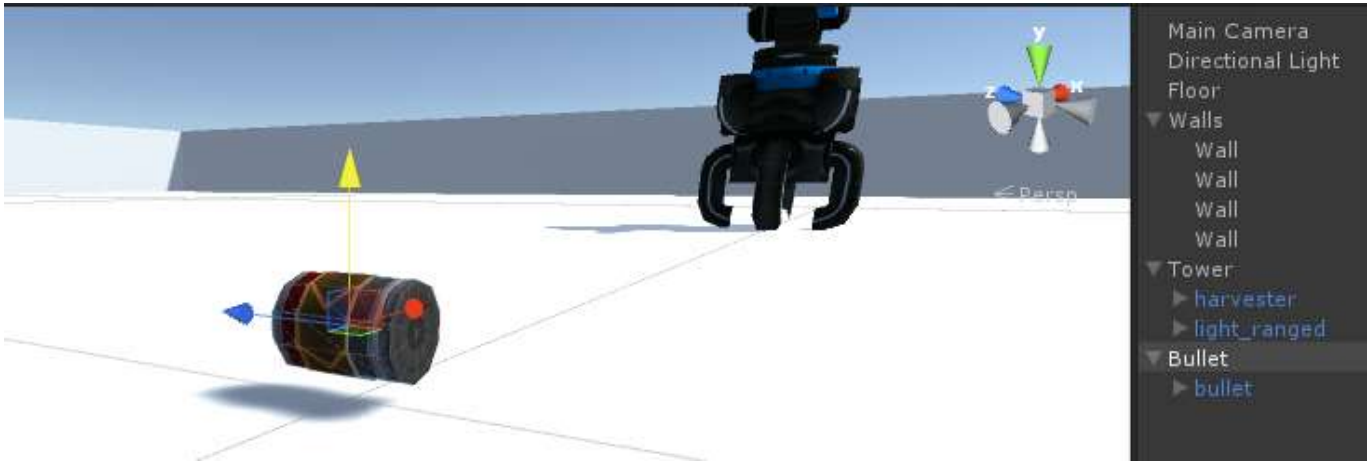
Tower Game

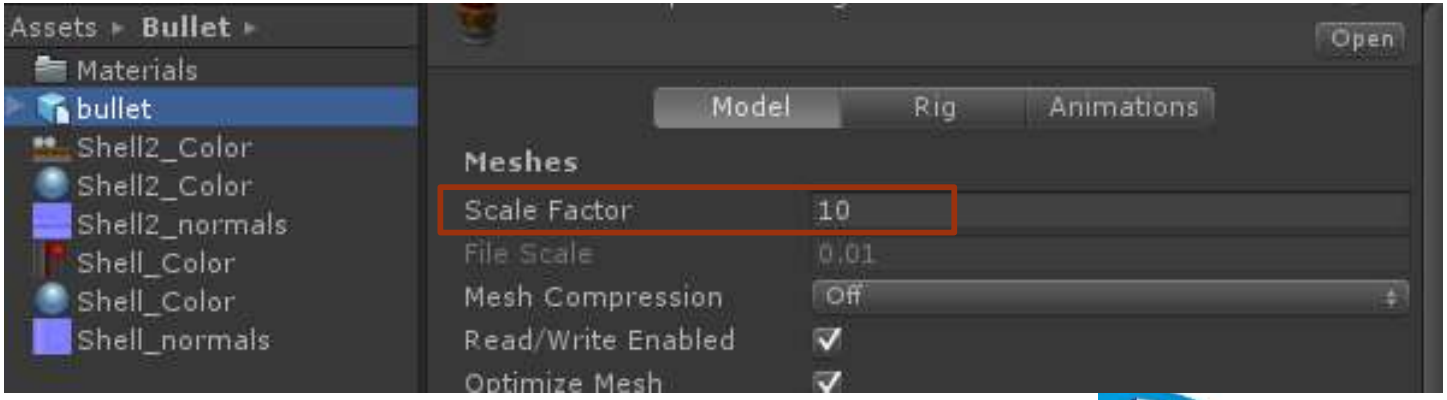
- 총알 움직임 만들기
  - Asset → Bullet 폴더 추가
    - Bullet → bullet.fbx 의 Scale Factor : 10으로 설정
    - bullet.fbx 를 Scene에 등록
  - Create Empty 생성 후 name : Bullet
    - bullet.fbx 를 Bullet 모델의 자식으로 등록
    - bullet.fbx → rotation : (90, 0, 0)
      - z축을 향하도록 변경

pivot을 바꾸고 싶다면 Create Empty 에 넣고
자식을 바꾸고, 명령은 부모인 Create Empty 에 주면됨

Tower Game

- 총알 움직임 만들기

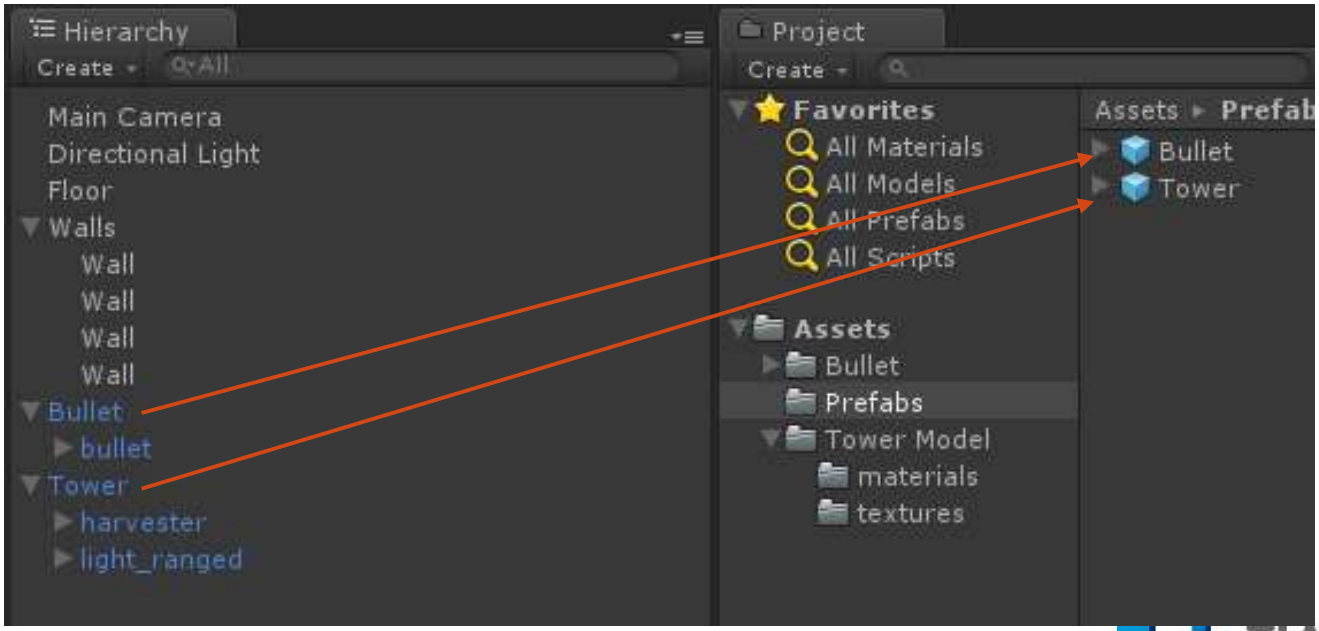




Tower Game

쏘는 주체 : 타워
날아가는 주체 : 불릿
→ 코드를 따로 분리해야함

- 총알 움직임 만들기
  - 프리팹 생성
    - Project → Create → Folder : Prefabs
    - Project → Create → Prefab : Bullet
      : Tower



Tower Game

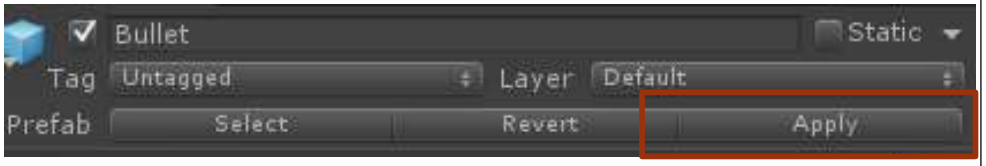
- 총알 움직임 만들기
  - 스크립트 생성
    - Project → Create → Folder : Scripts
    - Project → Create → C# script : cshFire.cs

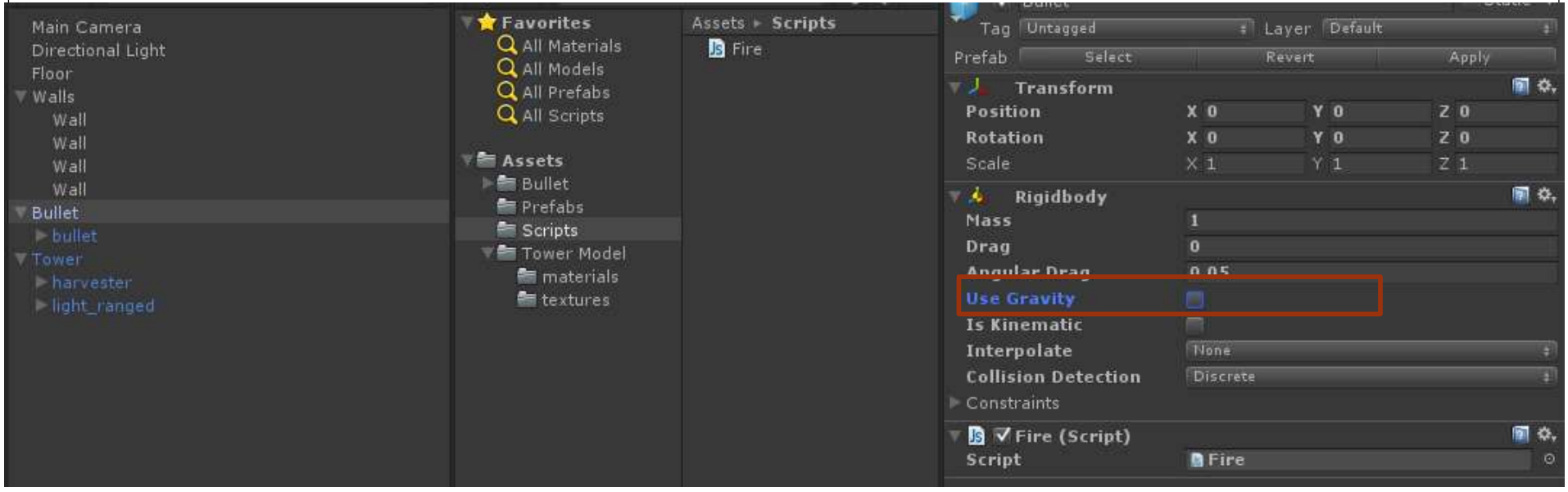
```
public class cshFire : MonoBehaviour {
        private float bulletSpeed = 1000.0f;
        private Transform thisTransform;
        // Use this for initialization
        void Start () {
                thisTransform = GetComponent<Transform> ();
                FireBullet ();
        }

        void FireBullet(){
                GetComponent<Rigidbody> ().AddForce (thisTransform.forward * bulletSpeed);
        }
}
```

Tower Game

- 총알 움직임 만들기
  - cshFire.cs 스크립트를 Bullet에 등록
  - Bullet Prefab에 Rigidbody 추가
    - 중력(Use Gravity) 비활성화
  - Prefab 갱신!!

원본인 프리팹에
객체의 설정을 적용하려면,
객체의 설정에서
Overrides → Apply





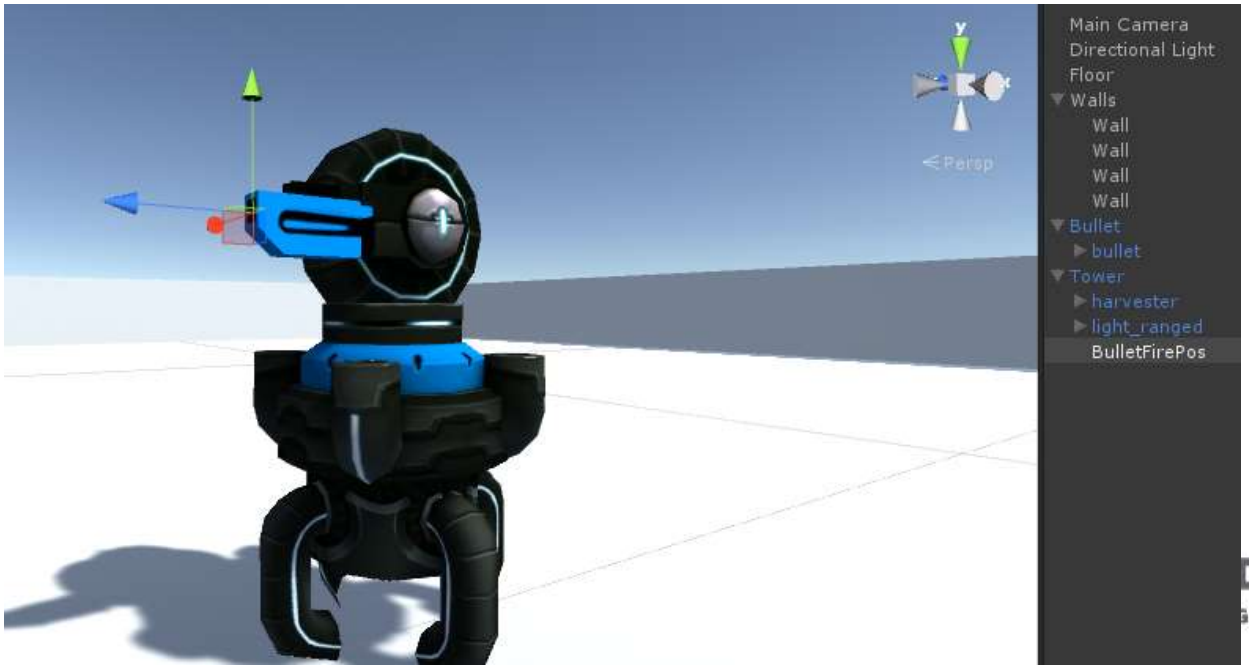
Tower Game

- 총알 생성
  - cshSpawnBullet.cs 스크립트 생성

```
public class cshSpawnBullet : MonoBehaviour {
        public Transform Bullet;
        public float fireTime = 1.0f;
        public float firePassTime = 0.0f;
        public Transform BulletFirePos;

        // Update is called once per frame
        void Update () {
                if (firePassTime >= fireTime) {
                        Instantiate (Bullet, BulletFirePos.position, BulletFirePos.rotation);
                        firePassTime = 0.0f;
                } else {
                        firePassTime += Time.deltaTime;
                }
        }
}
```

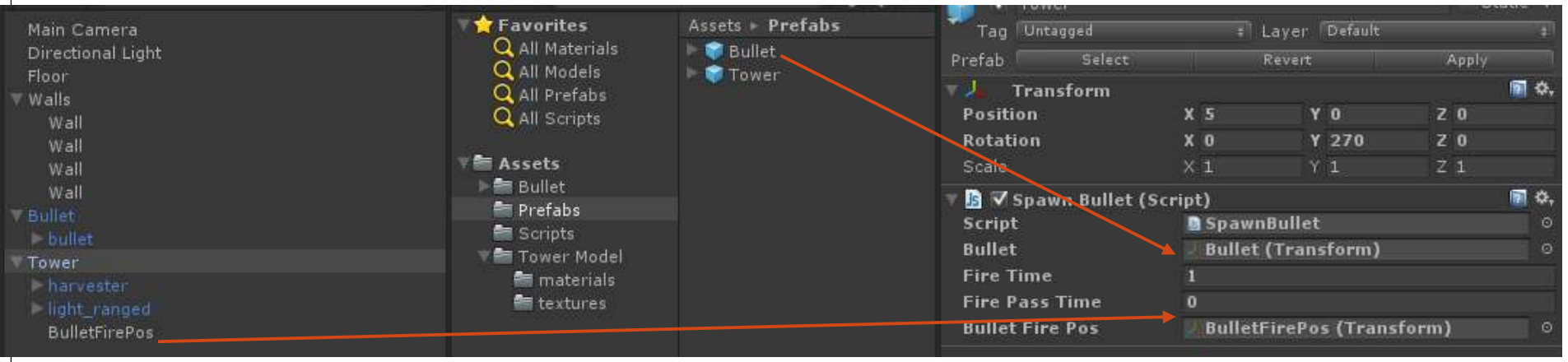
Tower Game

- 총알 생성
  - Tower 객체에 cshSpawnBullet.cs 등록
  - GameObject → Create Empty 생성
    - name : BulletFirePos
    - Tower 총알 발사 위치에 적절히 배치
    - Tower의 자식으로 등록



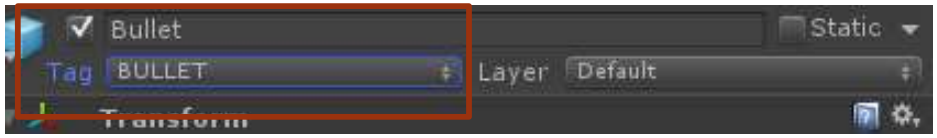
Tower Game

- 총알 생성



Bullet Collision

- Bullet 충돌처리
  - Tag 설정 Tag : 같은 목적인 객체들을 카테고리에 등록
    - Edit → Project Settings → Tags and Layers
    - Tags : BULLET ← Tag추가



    - Prefab → Apply (프리팹 수정)

Bullet Collision

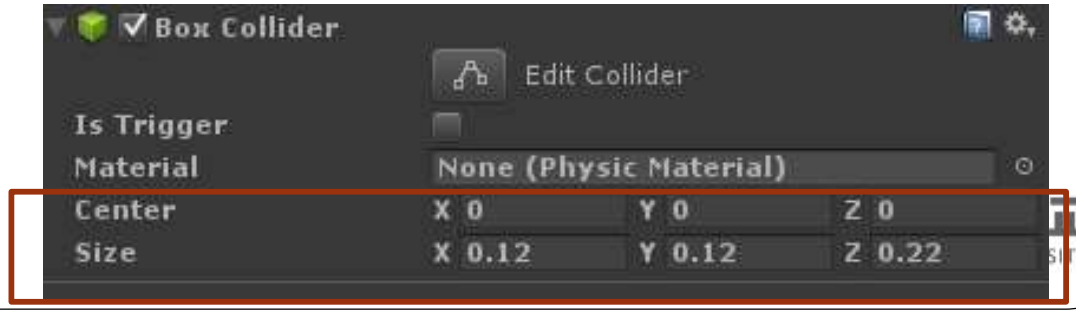
객체가 여럿이 생겼으면 Collider를 여러개 붙이기도 함

- Bullet 충돌처리
  - 충돌 스크립트 생성
    - cshHitBullet.cs

```
void OnCollisionEnter(Collision coll){
        if (coll.gameObject.tag == "BULLET") {
            Destroy (coll.gameObject);
        }
}
```

Destroy(11, 3.0f); 원래는 바로 사라지는데 이건 3초뒤 사라짐

  - Wall 객체에 cshHitBullet.cs 스크립트 등록
  - Bullet 프리팹에 BoxCollider 추가
    - Component → Physics → Box Collider

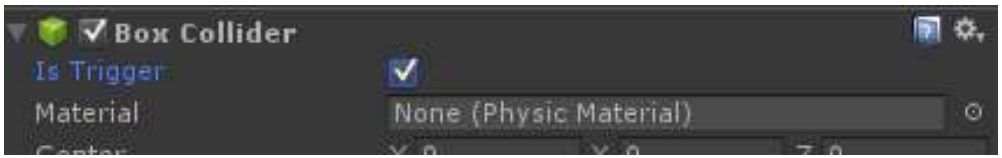


Bullet Collision

- Bullet 충돌처리 충돌은 감지하고 싶지만 진짜로 튕겨져 나가진 않았으면 좋겠을 때...
  - 벽(Wall)이 아닌 다른 물체와 충돌할 경우
    - 충돌 이후 물리 현상이 일어나는 문제가 발생

  - Trigger 개념 도입
    - Trigger Collider는 관통하는 개념
    - 일반적 충돌은 튕김과 같은 반사가 일어나지만 Trigger로 설정 시 물리가 적용되지 않고 충돌만 처리하여 관통

  객체
  - Bullet 의 BoxColler 속성
    - is Trigger 를 활성화



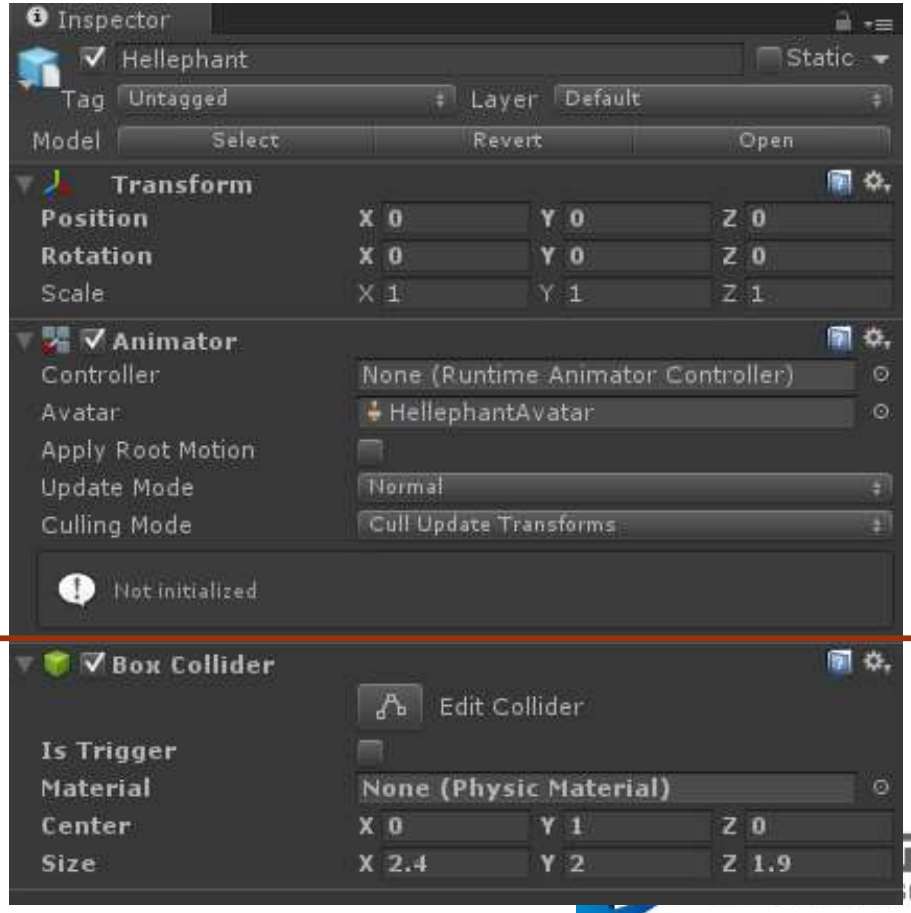
Bullet Collision

- Bullet 충돌처리 아군은 관통하는데 벽은 관통하면 곤란하므로
  - HitBullet.js 수정

```
void OnTriggerEnter(Collider coll){
        if (coll.gameObject.tag == "BULLET") {
                Destroy (coll.gameObject);
        }
}
```

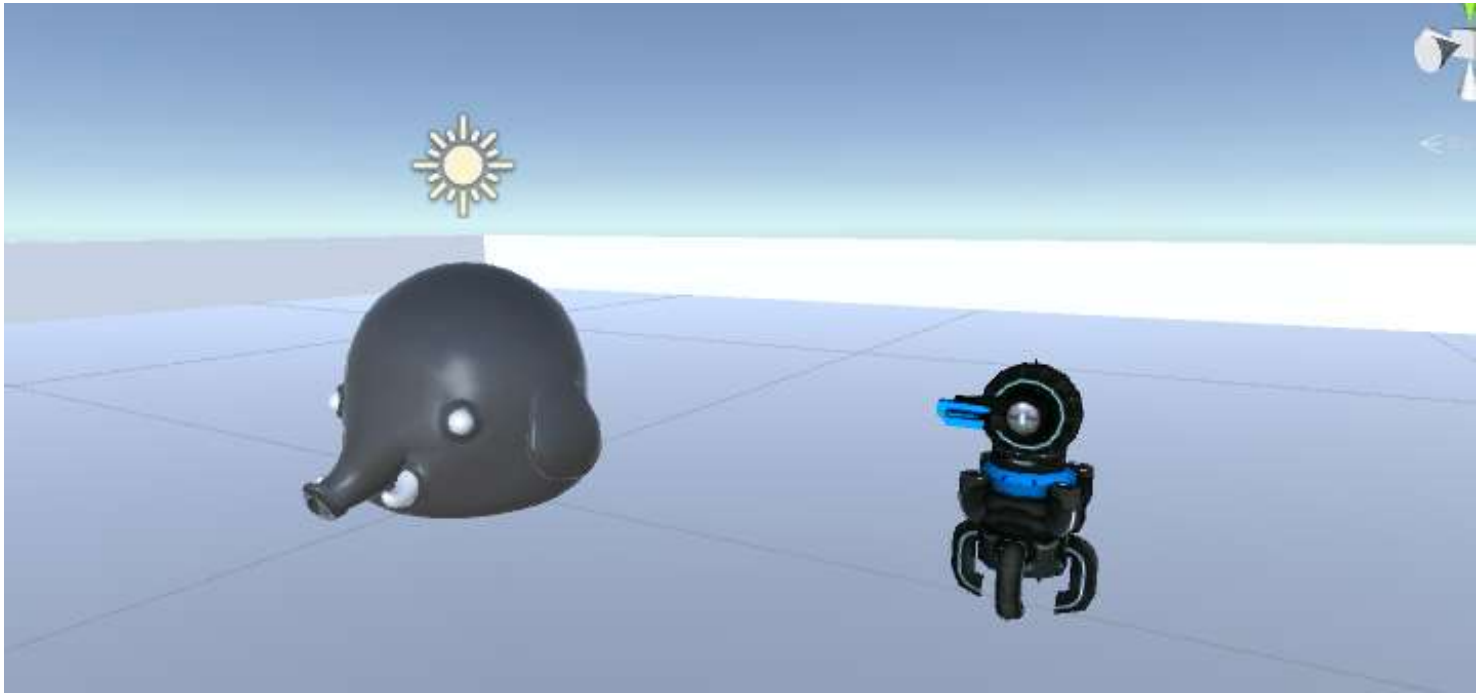
Bullet Collision

- 적군 배치
  - Hellephant 객체를 화면에 적당한 위치에 배치
    - Box Collider 추가



Bullet Collision

- 적군 배치



Bullet Collision

- 적군 배치
  - Hellephant과 Bullet과의 충돌 처리
    - cshDamageEnemy.cs

```
void OnTriggerEnter(Collider coll){
        if (coll.gameObject.tag == "BULLET") {
                Destroy (gameObject);
        }
}
```

    - Hellephant 객체에 등록

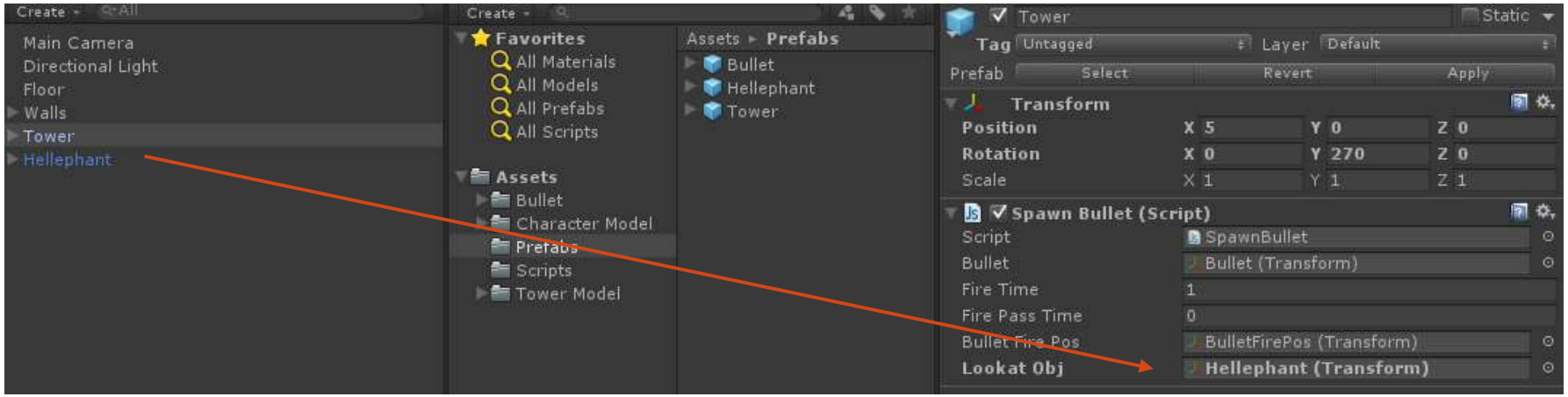
Bullet Collision

- Tower 객체가 항상 Hellephant를 바라 보도록 수정
  - cshSpawnBullet.cs 스크립트 수정

```
public class cshSpawnBullet : MonoBehaviour {
        public Transform Bullet;
        public float fireTime = 1.0f;
        public float firePassTime = 0.0f;
        public Transform BulletFirePos;

        public Transform LookatObj;

        // Update is called once per frame
        void Update () {
                if (firePassTime >= fireTime) {
                        Instantiate (Bullet, BulletFirePos.position, BulletFirePos.rotation);
                        firePassTime = 0.0f;
                } else {
                        firePassTime += Time.deltaTime;
                }

                transform.LookAt (LookatObj);
        }
}
```

Bullet Collision

- Tower 객체가 항상 Hellephant를 바라 보도록 수정
  - cshSpawnBullet.cs 스크립트 수정



- Hellephant의 cshDamageEnemy.cs 속성 삭제

참고로 colider 만 있다고해서 무조건 충돌되어 부딪히는 이벤트가 발생하는 건 아님, rigidbody 가 둘중하나라도 있어야 함 안그러면 관통함