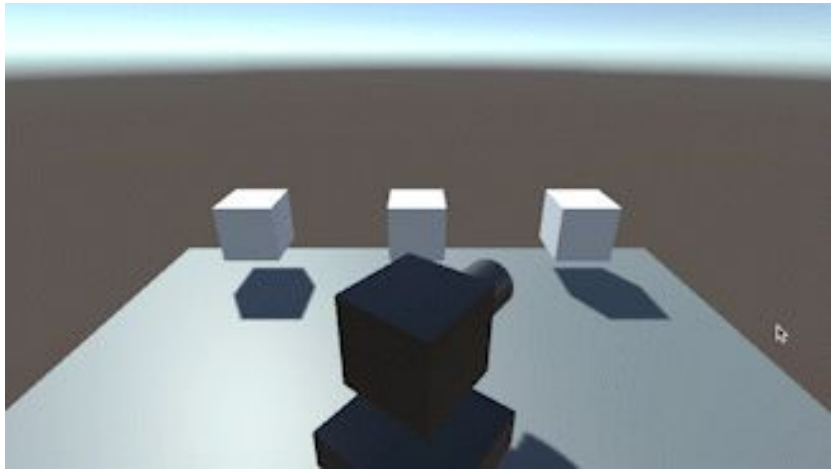
3D Shooter 製作

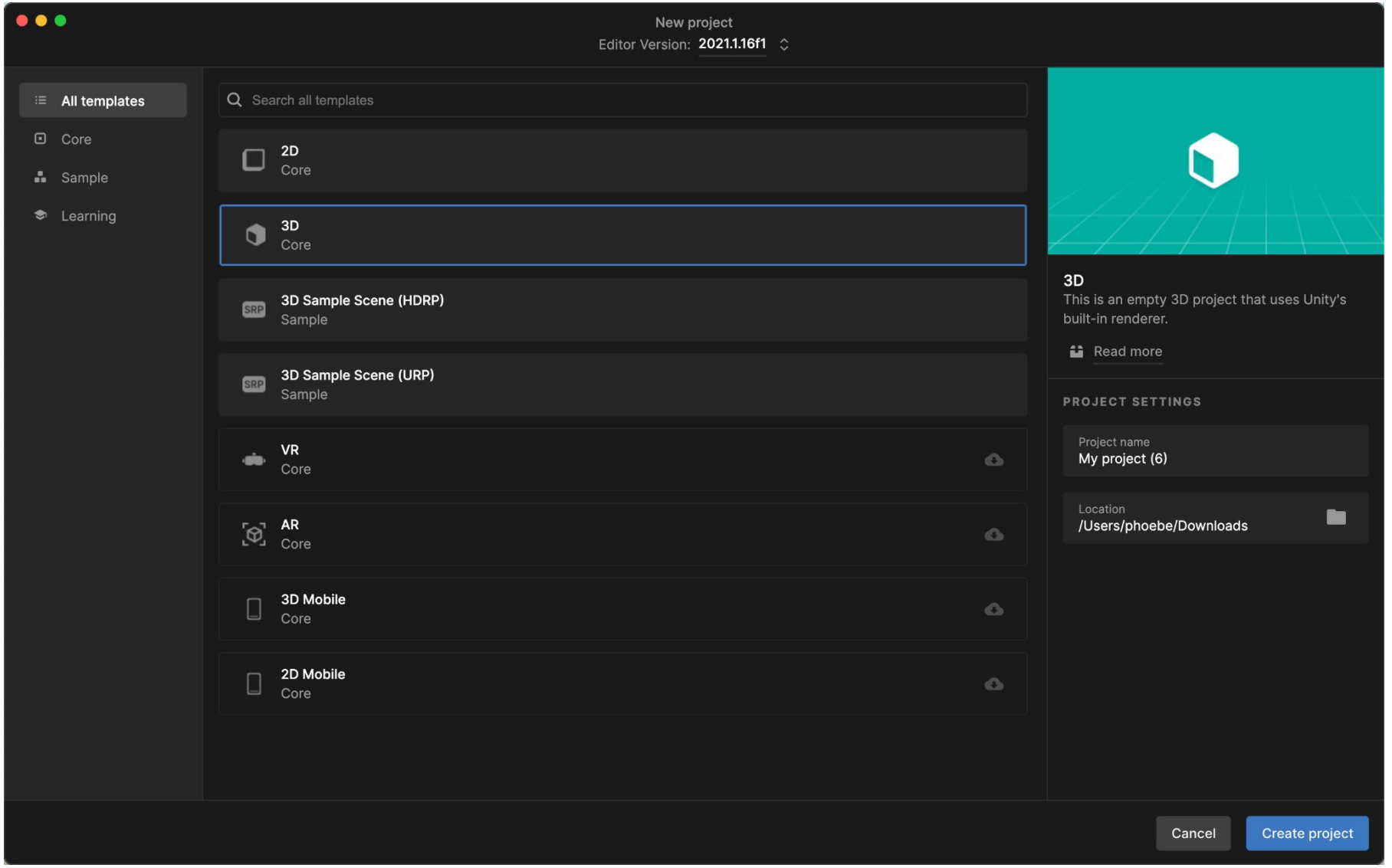
用多個基本元件製作砲台, 並發射砲彈

Outline



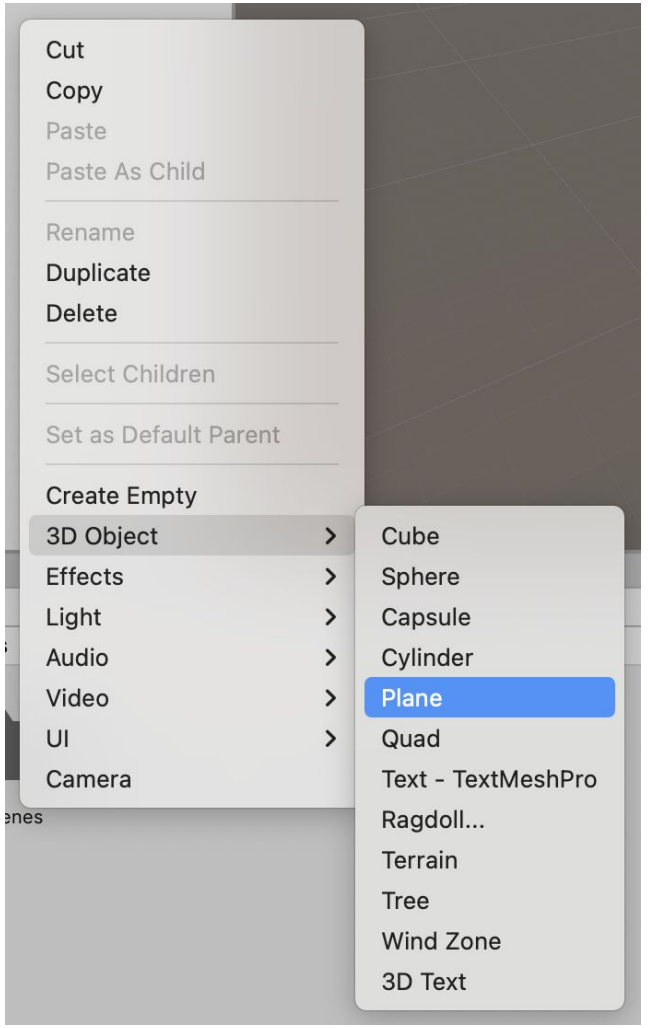
- 創建 3D scene
- **創建砲台 - Material / Object with hierarchy sub-object** / 設定旋轉
- **子彈設定 - Material / Prefab / 物理性質 / 飛行**
- 碰撞事件 - 子彈碰到目標消失
- 未碰到目標的子彈自行消失

建立3D專案


New project
Editor Version: 2021.1.16f1
All templates
Core
Sample
Learning
Search all templates
2D
Core
3D
Core
3D Sample Scene (HDRP)
Sample
3D Sample Scene (URP)
Sample
VR
Core
AR
Core
3D Mobile
Core
2D Mobile
Core
3D
This is an empty 3D project that uses Unity's built-in renderer.
Read more
PROJECT SETTINGS
Project name
My project (6)
Location
/Users/phoebe/Downloads
Cancel
Create project

場景設置

- 加入3D Object 裡的Plane作為場地地板
- 調整攝影機位置跟角度，能看到完整畫面即可

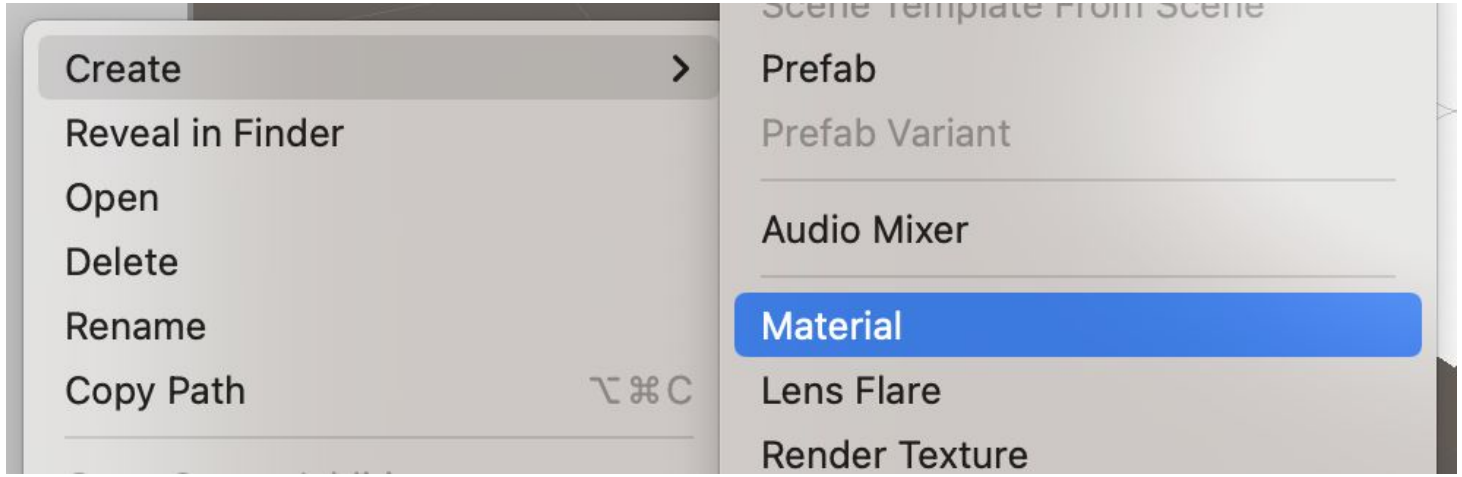


重要！！

- 3D物件生成時請養成reset Position的習慣

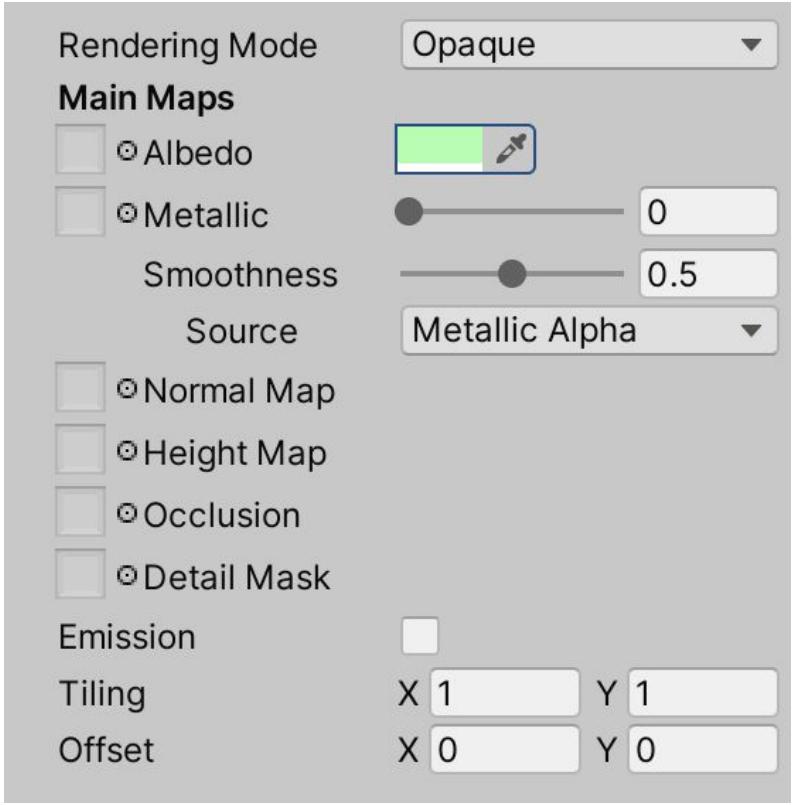
材質(顏色)製作

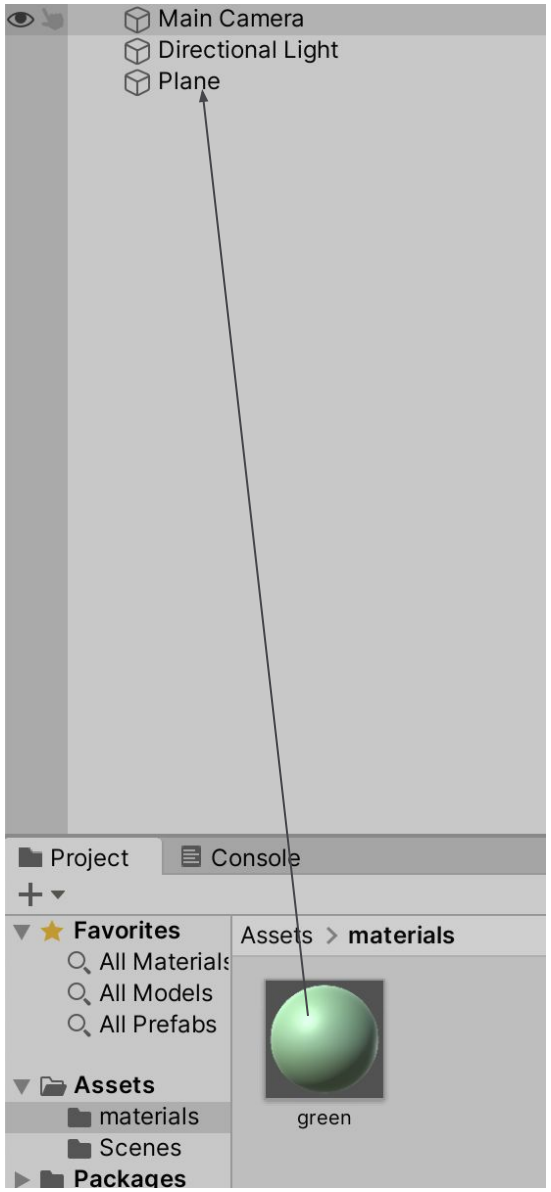
- 在 Assets 底下右鍵新增一個資料夾
- 在裡面右鍵新增 Material



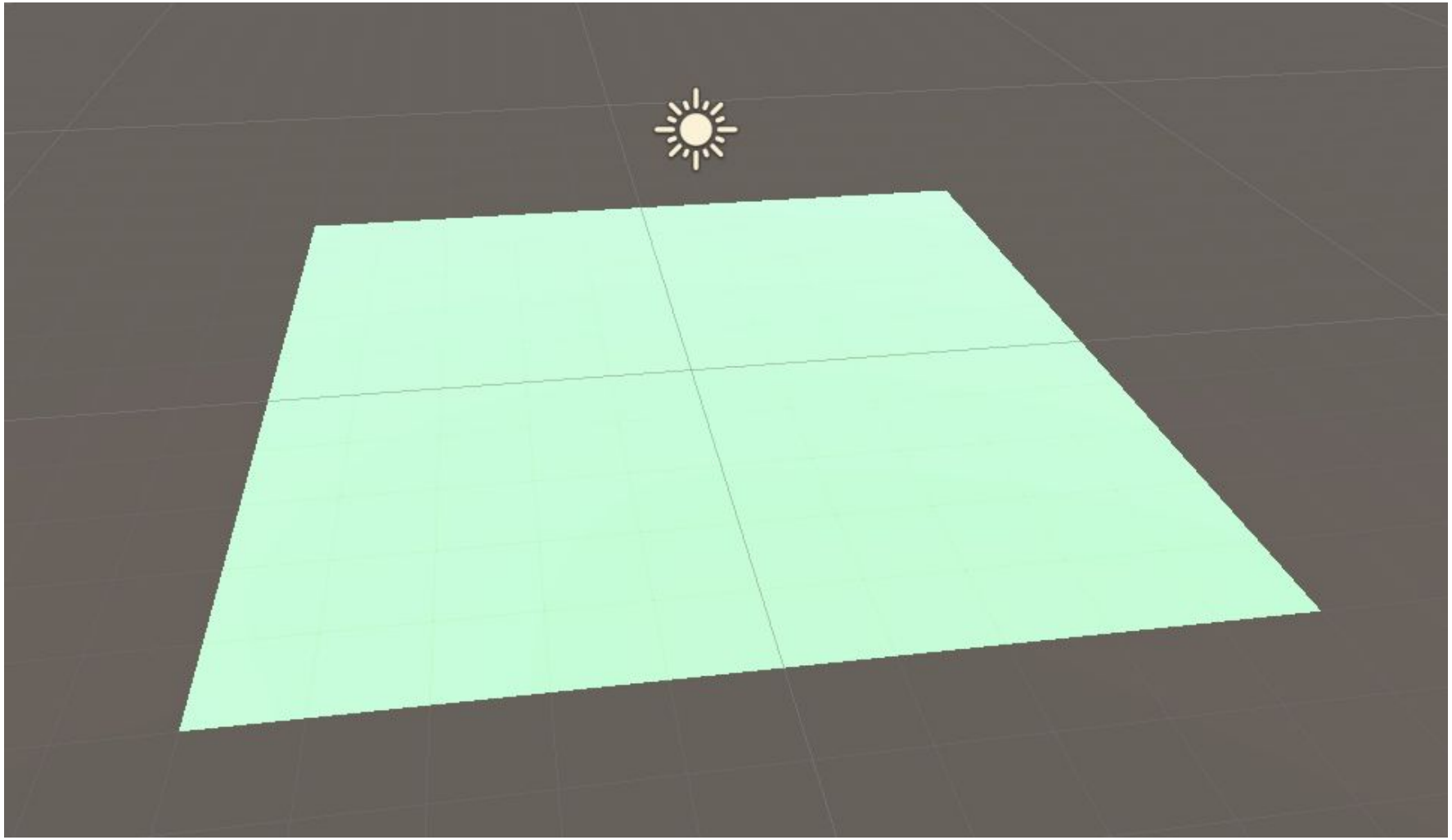
材質(顏色)製作

- 設定想要的顏色與材質
- 將材質拖到物件上即可使用





材質(顏色)製作

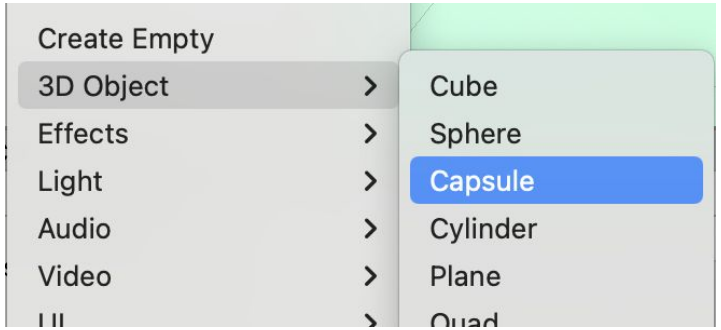


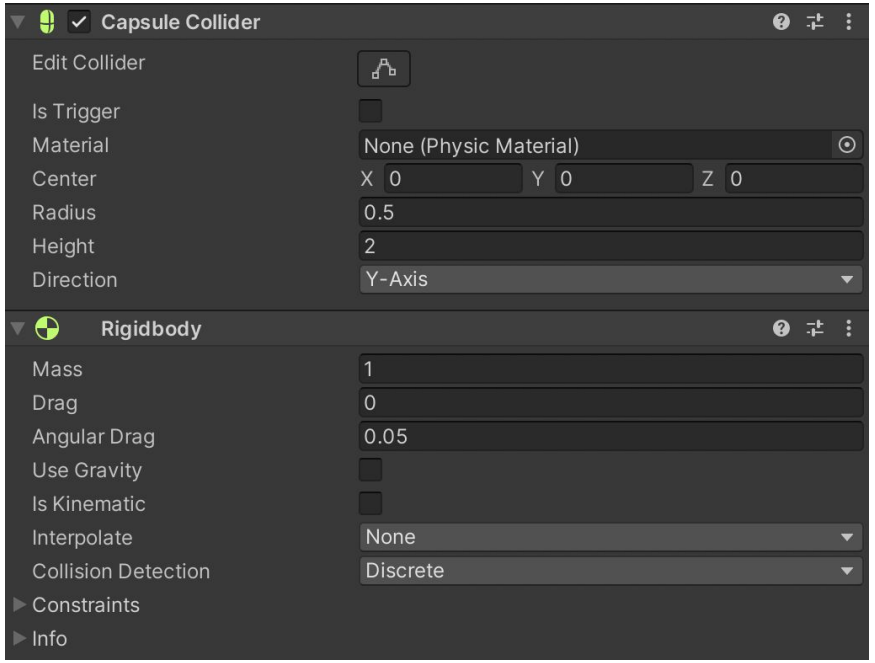
實作時間(5min)

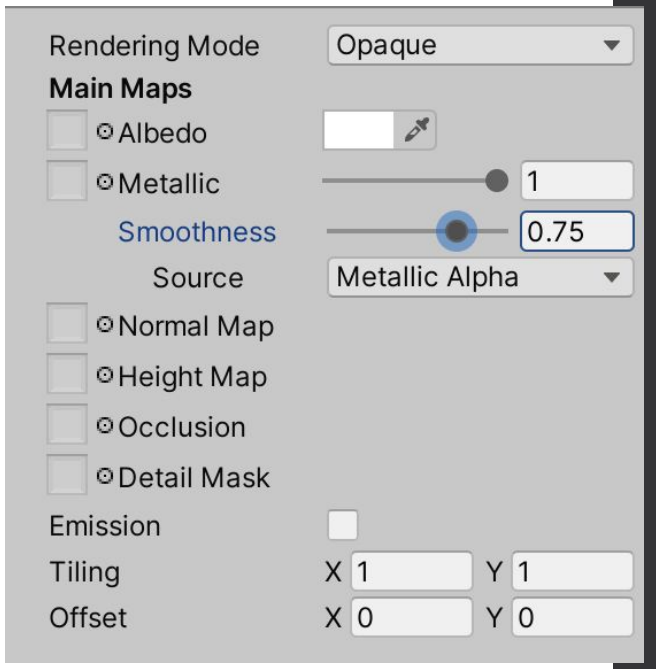
1. 建立3D專案
2. 設置場景:新增地板
3. 新增材質:地板材質

製作子彈

- 製作Object:右鍵Create Capsule
- 增加物理性質: Rigidbody (without gravity) + Capsule Collider
- 新增子彈material並調整



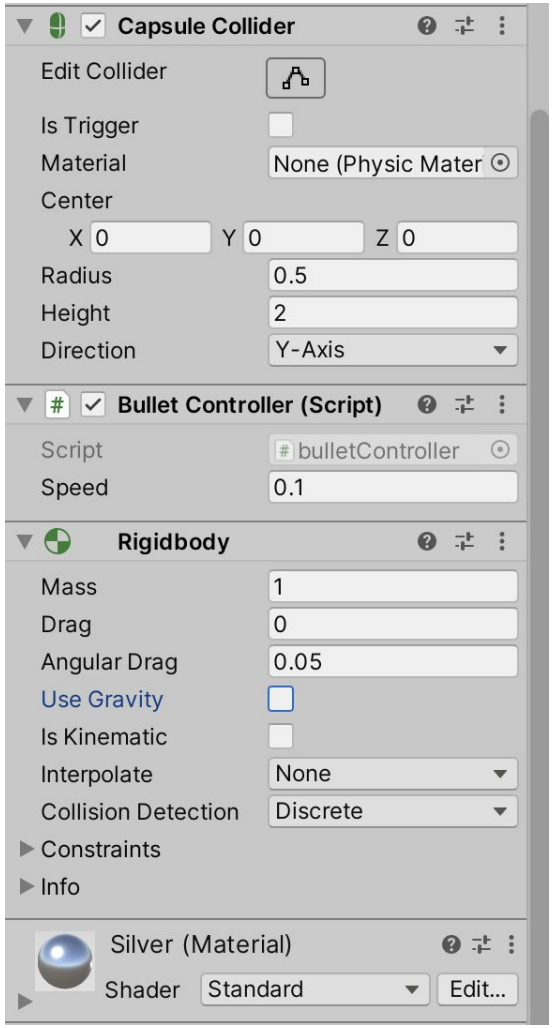




製作子彈scripts

- 在子彈上加入Script讓子彈會飛行
- 宣告一個變數為每次Update時移動的距離
- 讓子彈朝Y軸飛行
- Tip : 變數設成public可以在unity上調整

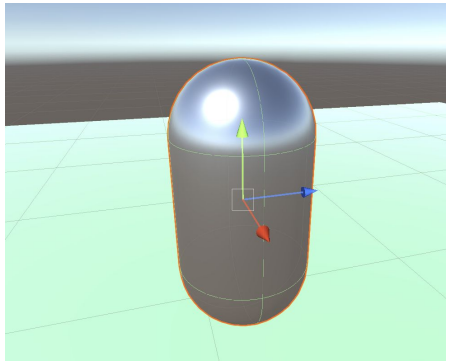
```
1 reference
public float speed = 0.1f;
// Use this for initialization
0 references
void Start() {

}

// Update is called once per frame
0 references
void Update() {
    this.transform.Translate(new Vector3(0, -speed, 0));
}
```

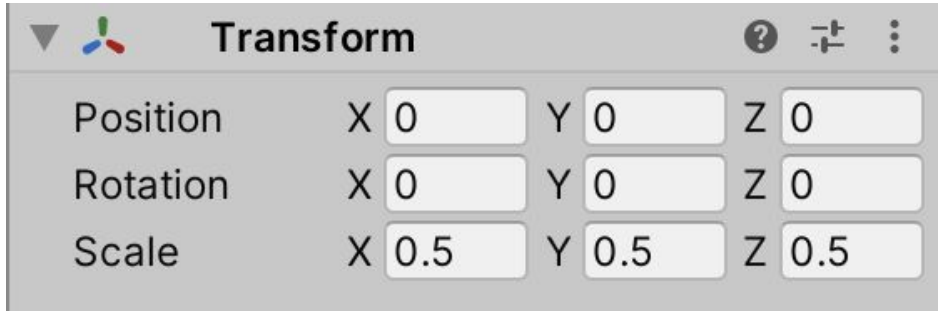


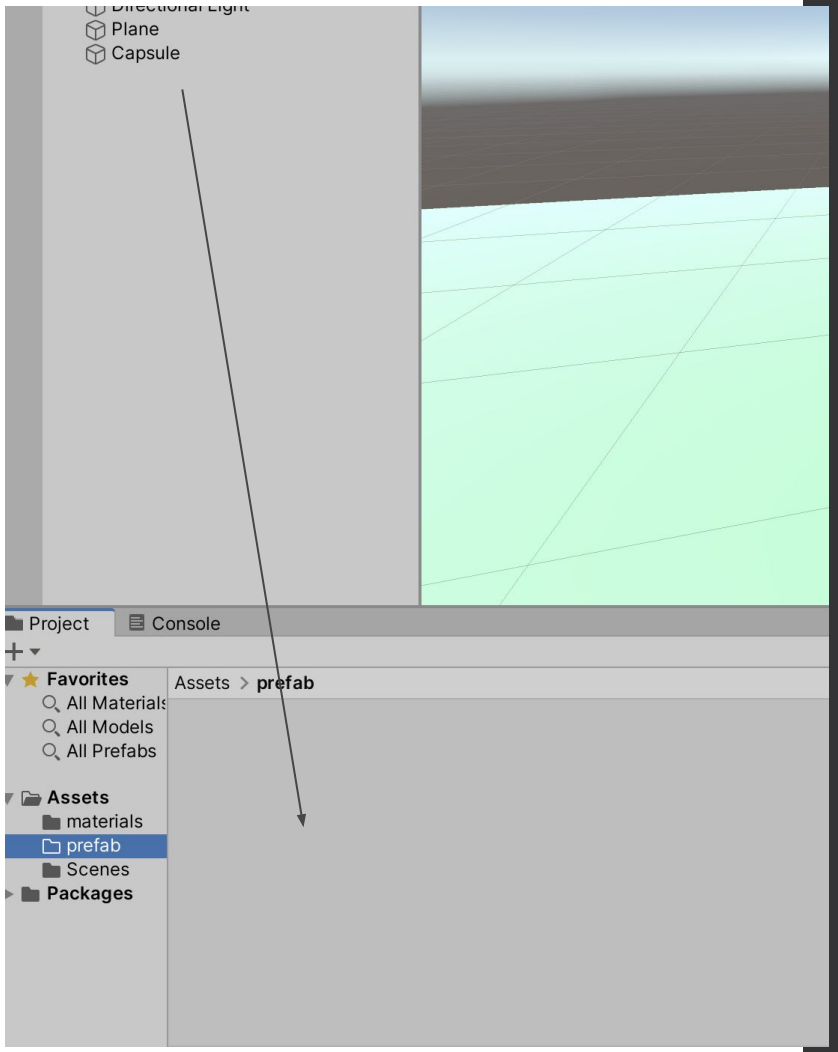
製作Prefabs

Prefab：預先製作的物件，可以動態生成並重複使用。完成後只要直接修改裡面的內容就會套用到所有生成的物件

- 調整一下子彈大小
- 把子彈丟到project tab
- 把子彈Rotate x -90





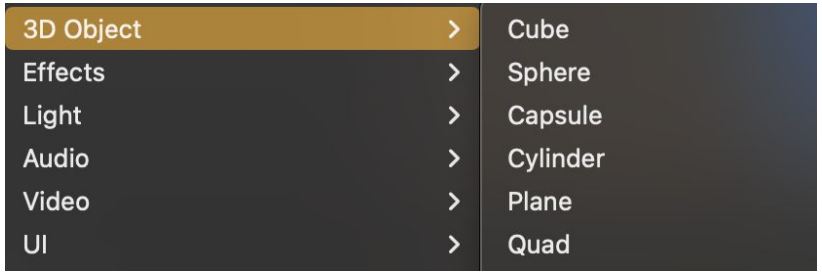


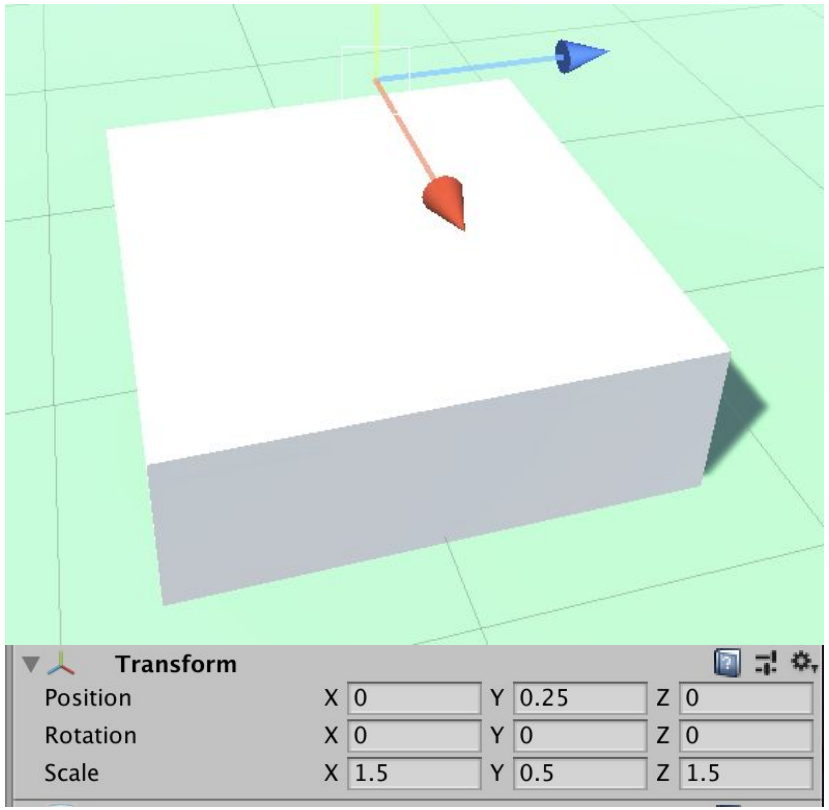
實作時間(5min)

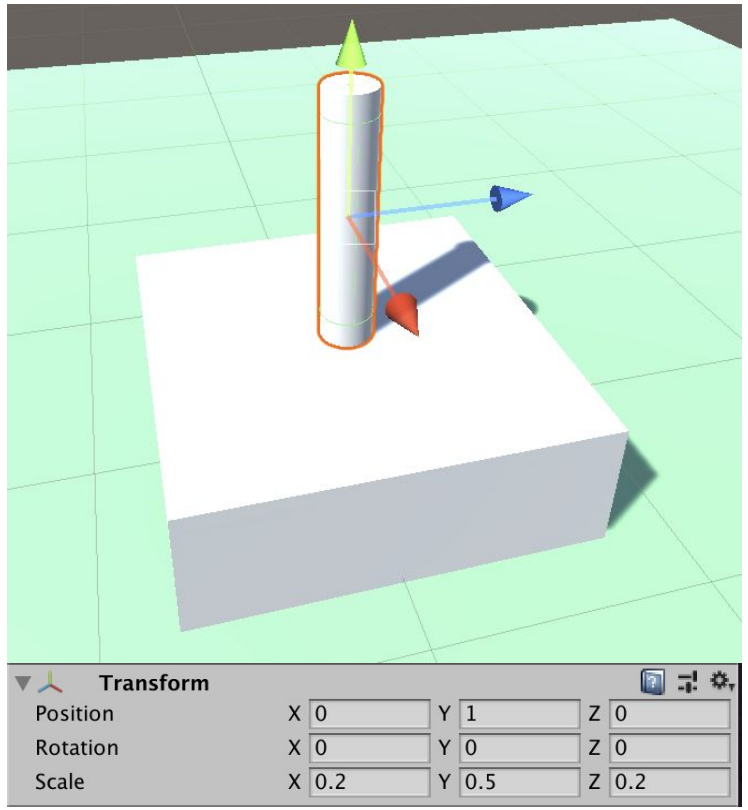
1. 製作子彈
2. 設定子彈物理性質
3. 設定子彈材質
4. 替子彈設定速度
5. 將子彈設定成Prefab

製作砲台

- 新增Cube：製作砲台底座
- 新增Cylinder：製作基座

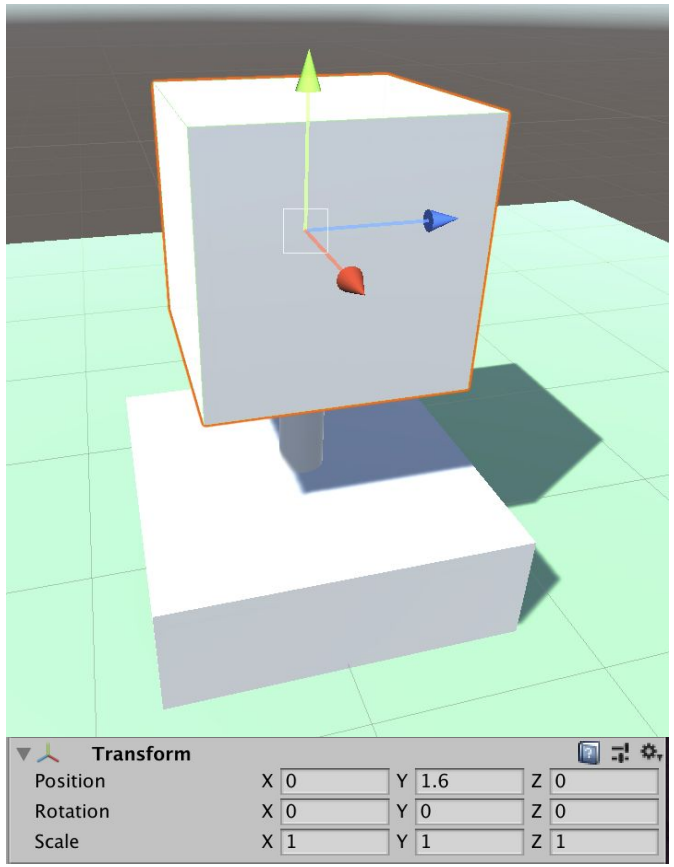


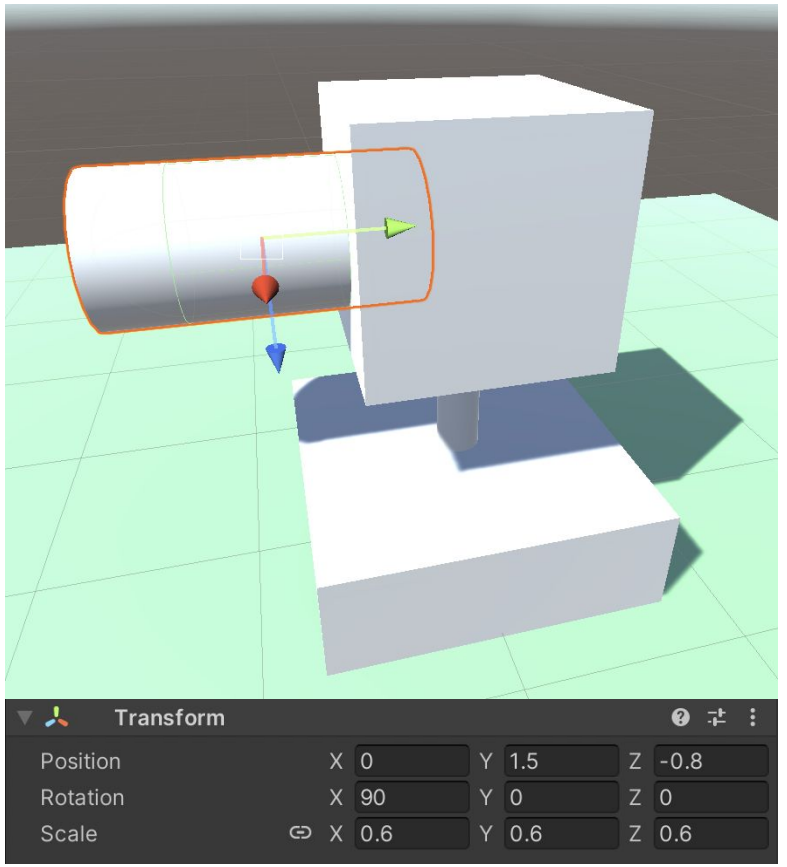




製作砲台

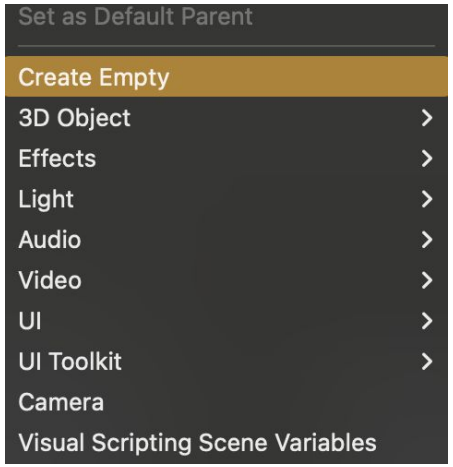
- 新增Cube:製作砲身
- 新增Cylinder:製作砲管

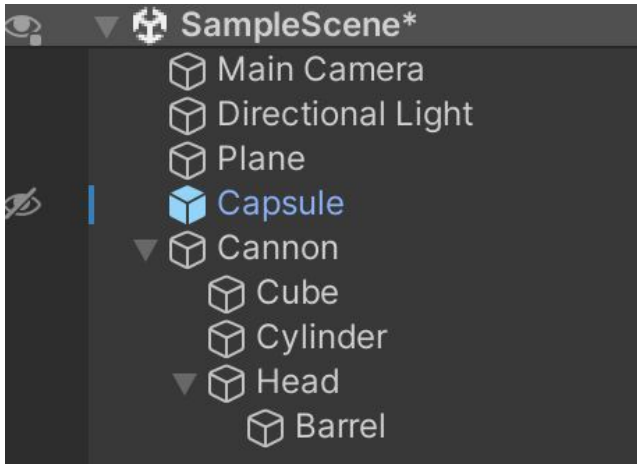


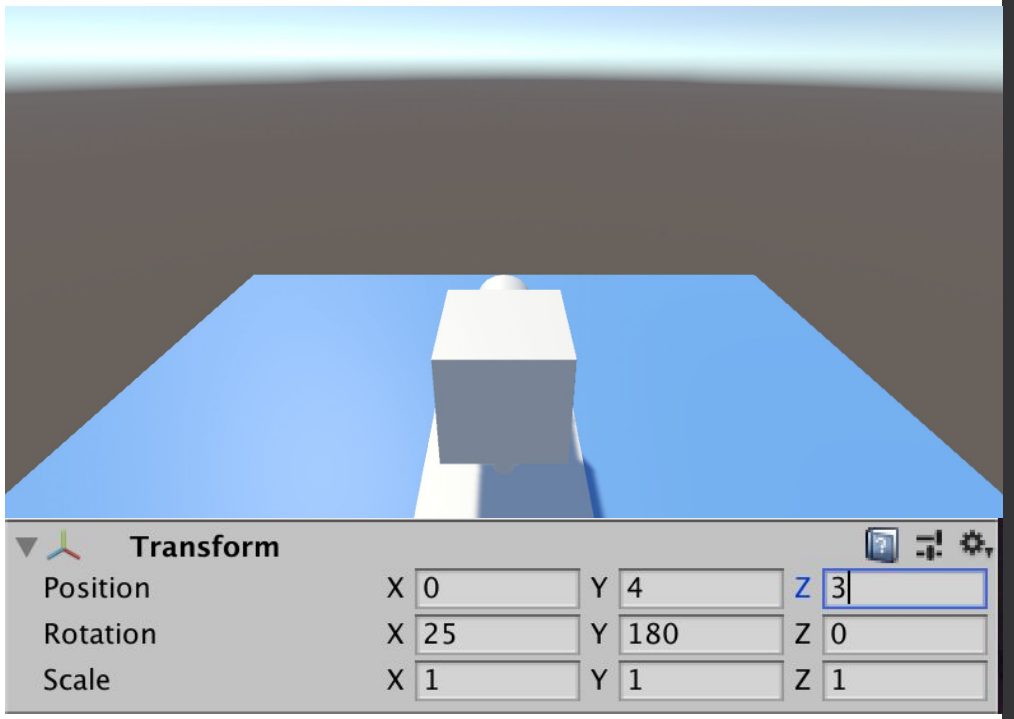


製作砲台

- Concept：
  - ☺ 製造物件時，移動旋轉通常都會發生在物件的中心點，如果要以邊旋轉就讓邊成為另外一個物件的中心即可
  - ☺ 如果有巢狀或多關節的物體，位移跟旋轉用相對位置去計算
- 在Hierarchy中設定砲台物件層級

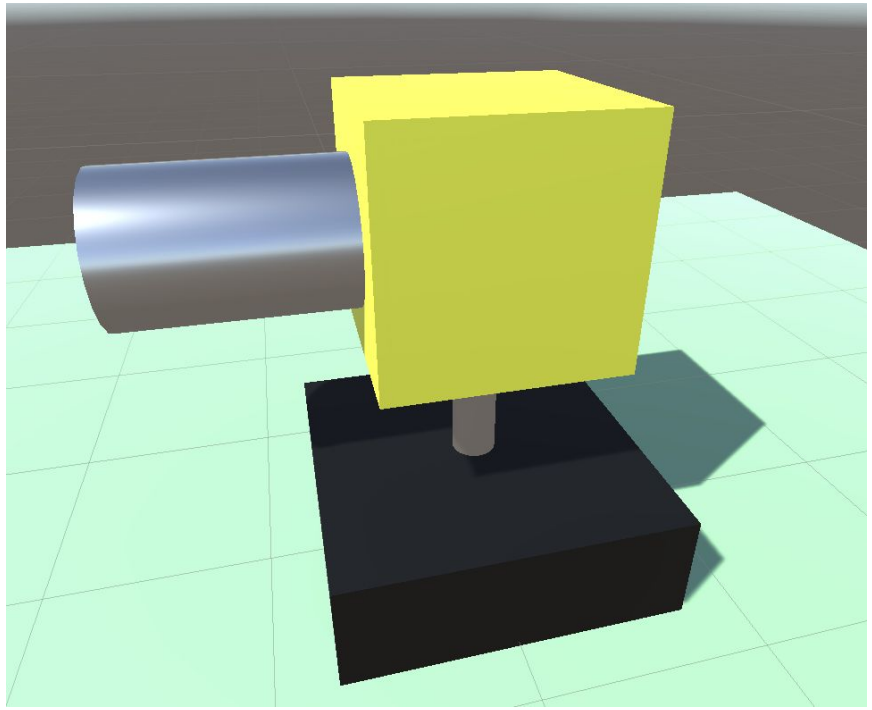


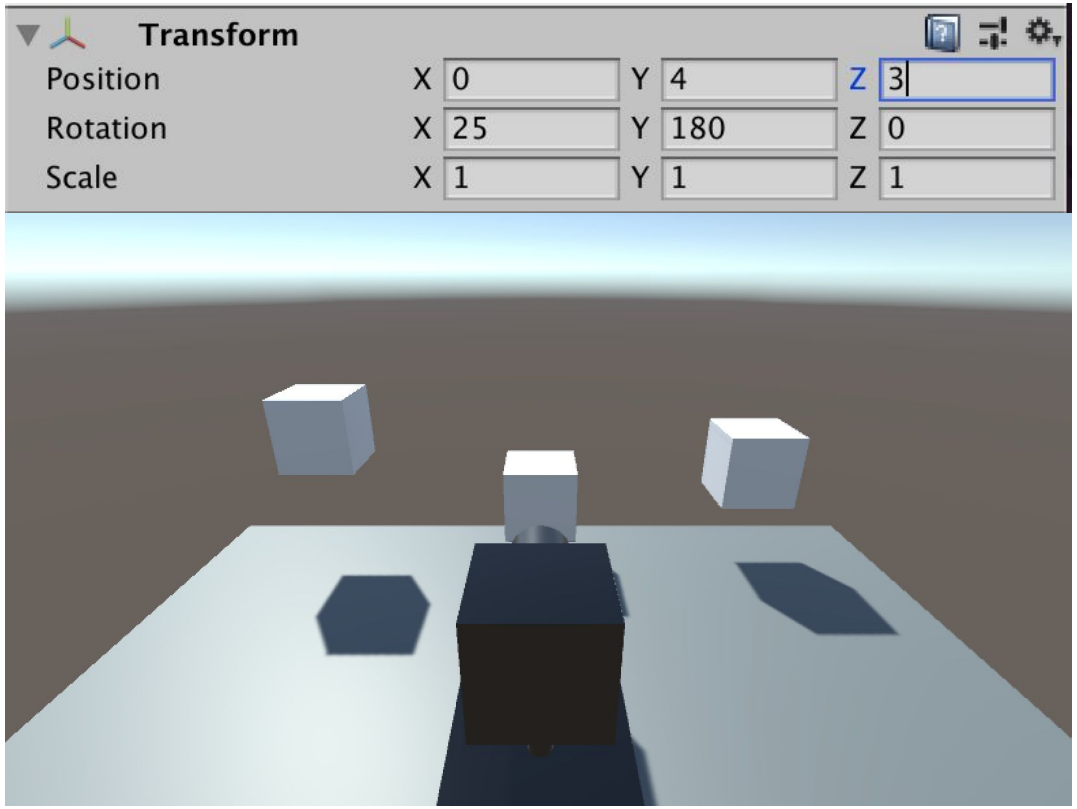




砲台材質

- 替砲台加上Materials
- 調整Camera





實作時間（5min）

1. 建造砲台
2. 調整砲台物件層級
3. 設置砲台Materials

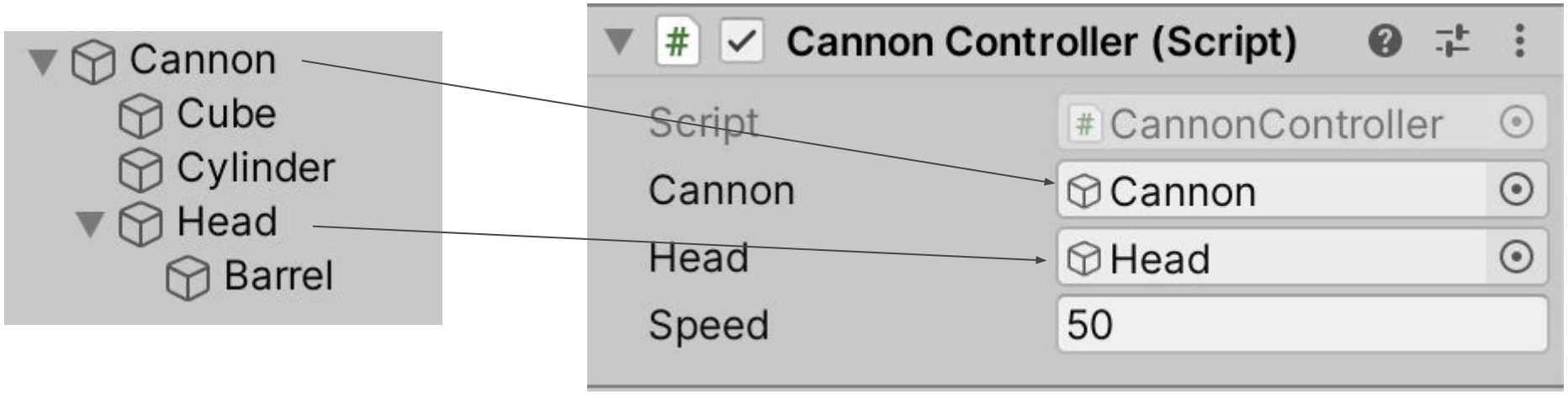
旋轉砲台

- 在砲台加入上下/左右旋轉(沿著X/Y軸)的Script
- 將此Script加入砲台(Cannon)中

```
1 reference
public GameObject Cannon;
1 reference
public GameObject Head;
2 references
public float speed = 50f;
// Use this for initialization
0 references
void Start() {

}

// Update is called once per frame
0 references
void Update() {
    float h = Input.GetAxisRaw("Horizontal") * speed * Time.deltaTime;
    float v = Input.GetAxisRaw("Vertical") * speed * Time.deltaTime;
    Head.transform.Rotate(new Vector3(v, 0, 0));
    Cannon.transform.Rotate(new Vector3(0, h, 0));
}
```

旋轉砲台

- 將Script加入砲台裡，並將Cannon和Head物件拉到剛才Script設為Public的地方

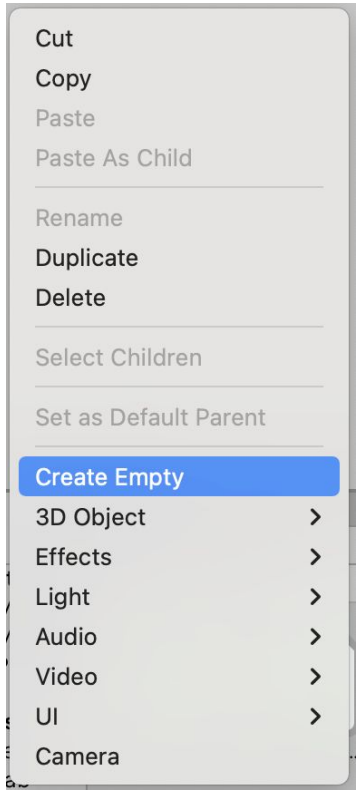


實作時間(3min)

1. 設定砲台旋轉

製作發射點

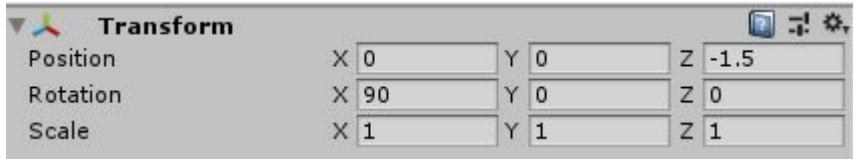
- 先將發射點放到砲台之中
- 設定發射點位置



Hierarchy

GameObject為發射點

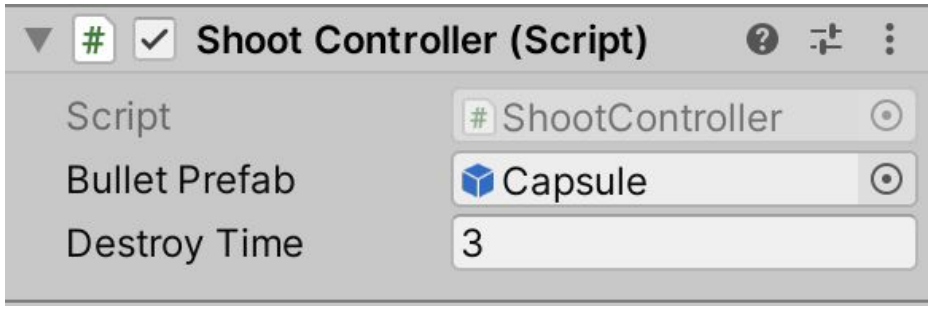
發射點位置



間隔時間發射

使用time來讓發射有冷卻時間

- 發射後歸零
- 將Script加入剛才的發射點
- 設定發射的物件(必為Prefab




```
1 reference
public GameObject BulletPrefab;
3 references
float time = 0f;
0 references
public float destroyTime = 3f;
// Use this for initialization
0 references
void Start() {

}

// Update is called once per frame
0 references
void Update() {
    time += Time.deltaTime;
    if (Input.GetKeyUp(KeyCode.Space)) {
        if (time > 0.5f) {
            Instantiate(BulletPrefab, this.transform.position, this.transform.rotation);
            time = 0;
        }

    }
}
```

製作發射點

- 在砲管head上新增一個Empty，移動到發射砲彈的位置
- if (Input.GetKeyUp(KeyCode.Space))

  > 當空白鍵被按下放開的時候觸發

  - GetKey:按著一直觸發
  - GetKeyDown:按下觸發
- Instantiate(Object, Position, Rotation)

  > 在指定位置跟角度生成該物件

- 物件要設成public才能在unity決定要生成甚麼物件

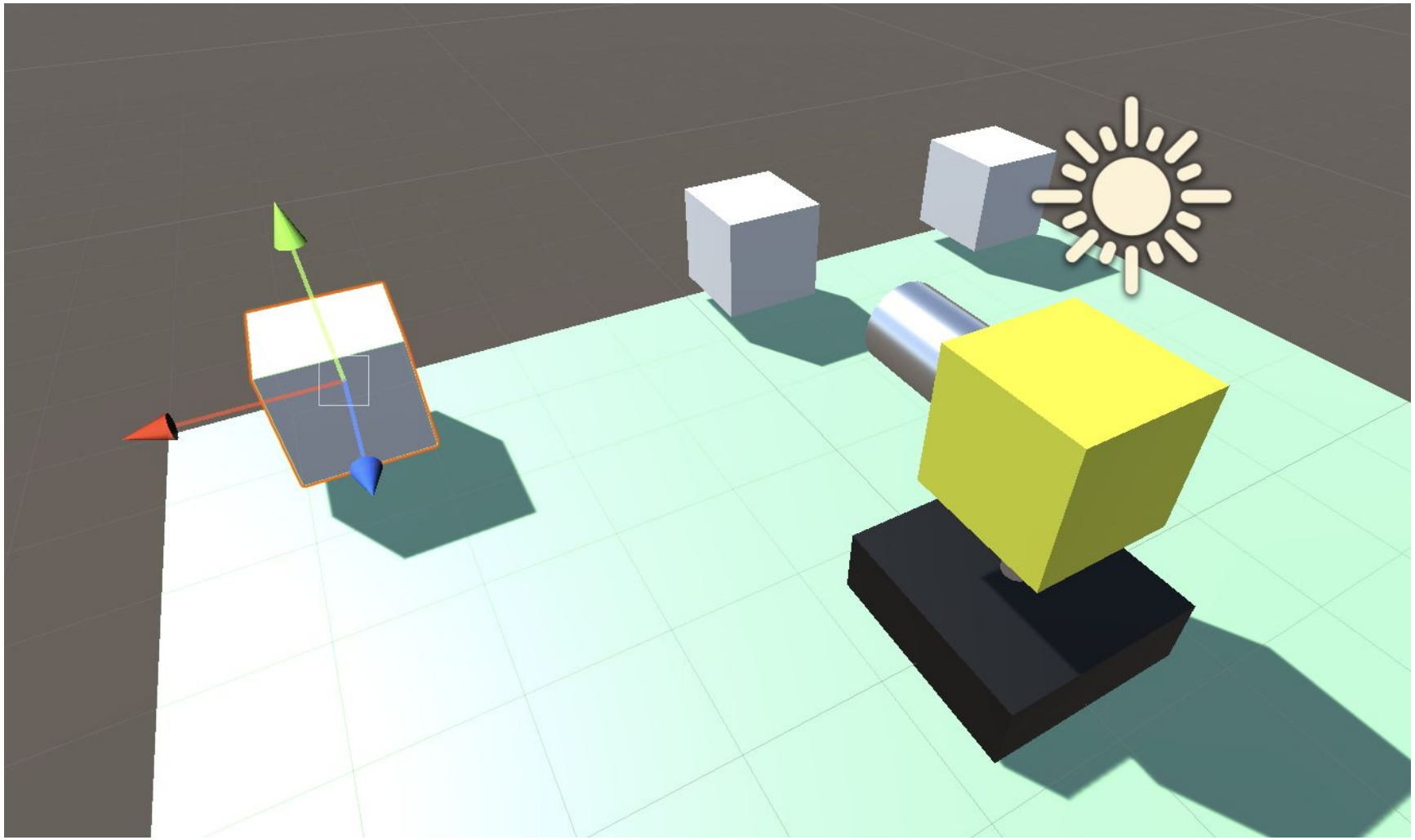
實作時間(3min)

1. 新增空物件
2. 設定發射位置
3. 指定發射物件
4. 設定間隔時間

碰撞與消失

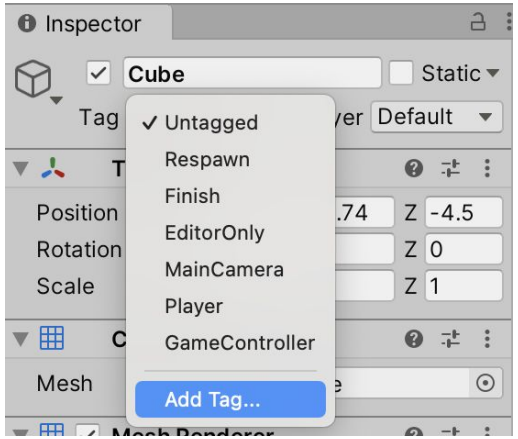
- 子彈生成後會一直存在，所以必須要摧毀它
- 如果沒有發生碰撞，子彈飛一段時間後消失
- 如果發生碰撞，可能消滅目標一起消失或者撞到牆自我毀滅等等
  - Destroy(gameObject): 馬上摧毀
  - Destroy(gameObject, 秒數): 幾秒後摧毀

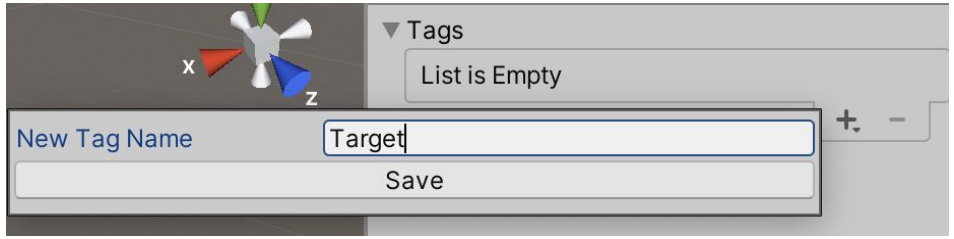
製作目標物件

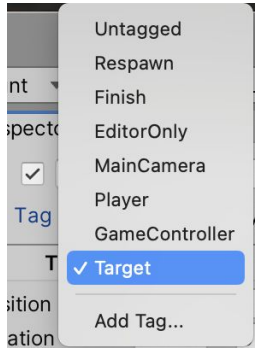


目標物件加入tag

- 把子彈跟想要碰撞的物體(像是邊界、方塊等)有可以被碰撞的屬性並在被碰撞到的物體加入tag分類即可
- 要碰撞的物體要加入RigidBody
- 新增Tag - Target
- 將目標物加入Target的Tag



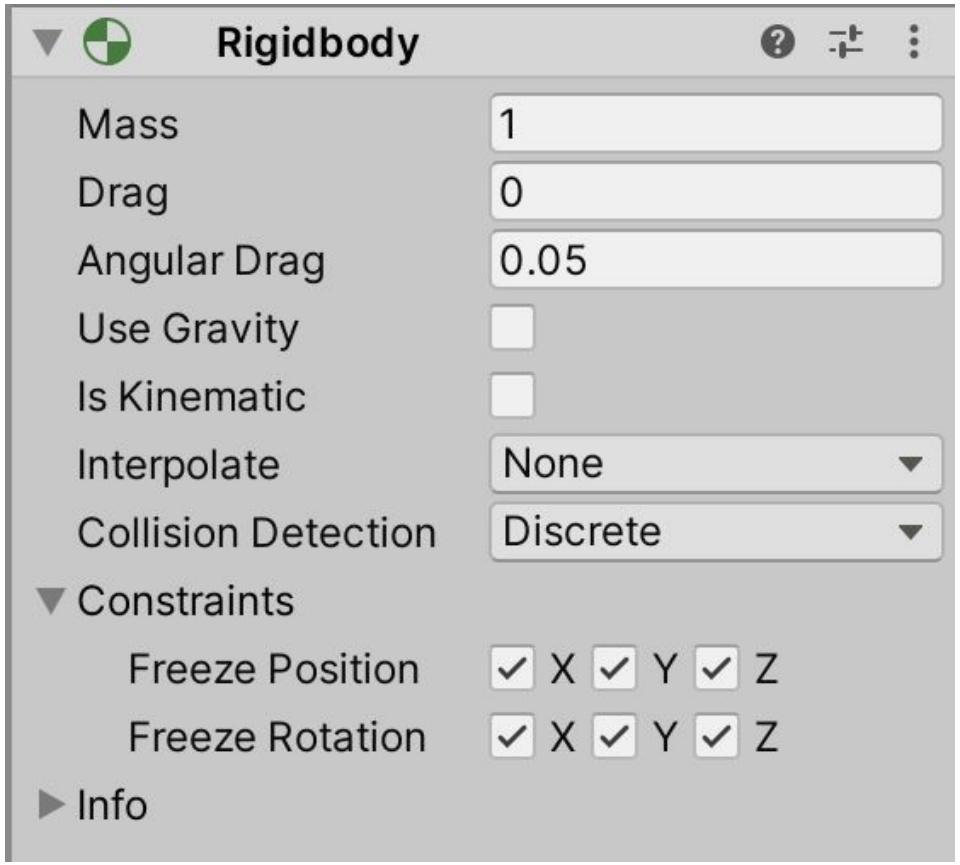




目標物加入Target的Tag

子彈物理屬性

- 設定Bullet的物理屬性
  - ☺ 加入Constraints避免子彈旋轉與不必要的亂動



子彈碰撞事件

在Bullet的Script裡加入碰撞事件
（注意！！不是加在Update裡）

- private void OnCollisionEnter(Collision collision){ }:
  - 當碰撞發生的時候觸發，後面參數為碰撞到的目標
- 在裡面加入條件判定就可以決定碰到不同物體有不一樣的動作

```
private void OnCollisionEnter(Collision collision) {
    if (collision.gameObject.tag == "Target") {
        Destroy(this.gameObject);
        Destroy(collision.gameObject);
    }
}
```

Garbage collection

沒有打到目標的子彈會一直留在hierarchy上

- 在子彈的script中start()加入
  - Destroy(this.gameObject, 3.0f);

```
1 reference
public float speed = 0.1f;
// Use this for initialization
0 references
void Start() {
    Destroy(this.gameObject, 3.0f);
}

// Update is called once per frame
0 references
void Update() {
    this.transform.Translate(new Vector3(0, -speed, 0));
}
```

實作時間(8min)

1. 新增目標物件
2. 設定Tag
3. 設定子彈物理屬性
4. 碰撞事件:消失
5. 未碰撞的子彈自行消失

新增更多功能

可以自行新增不同功能，例如

- 砲管旋轉限制
- 前後移動
- 相機視角
- 套用Asset Store的模型

Thanks For Listening