# SpaceShooter2D 单机版实作范例



**大风起兮云飞扬**
**威加海内兮归故乡**
**安得猛士兮守四方**

《大风歌》

本文将以 Unit3D 的 3D 场景制作一个 2D 太空射击游戏，透过实作让读者更能了解如何以 Unity3D 开发游戏。除了与美术有关的模型与素材外，这个游戏项目将会从无到有，一步一步带领读者实际操作，完成太空射击游戏。

接着我们就开始进行，让大家体会开发游戏的乐趣吧！

## 1-1 建立游戏项目基础架构

## 1-2 玩家角色

完整教程请至 http://developer.arcalet.com 进行下载。

## 1-3 敌人系统

太空战机的敌人就是由上而下移动的陨石，玩家必须操控太空战机左右移动以避开陨石，或是发射炮弹摧毁陨石。本文的敌人系统除了要制作陨石，还要处理物体碰撞问题，包括炮弹与陨石的碰撞，以及陨石与太空战机的碰撞。

### 制作陨石

制作陨石的步骤跟上一节的太空战机制作方式一样，都是先建立一个基本的几何模型，然后修改它的 Mesh Filter 成为我们指定的陨石模型，最后再加以贴图让外观变成陨石的样子就可以了，详细的操作步骤请看下面的说明。

一、 建立陨石 GameObject

操作步骤：

(1) 点选主选单→「Create Other」→「Sphere」

(2) 更名为 Enemy，之后我们都称呼它为 Enemy

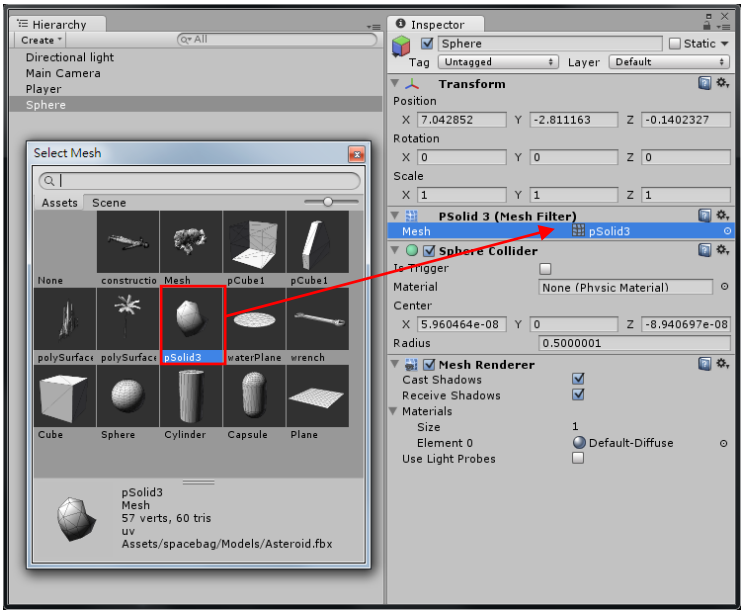
(3) 设定 Position 与 Rotation 均为(0, 0, 0)

(4) 调整 Scale 为(1.3, 1.3, 1.3)

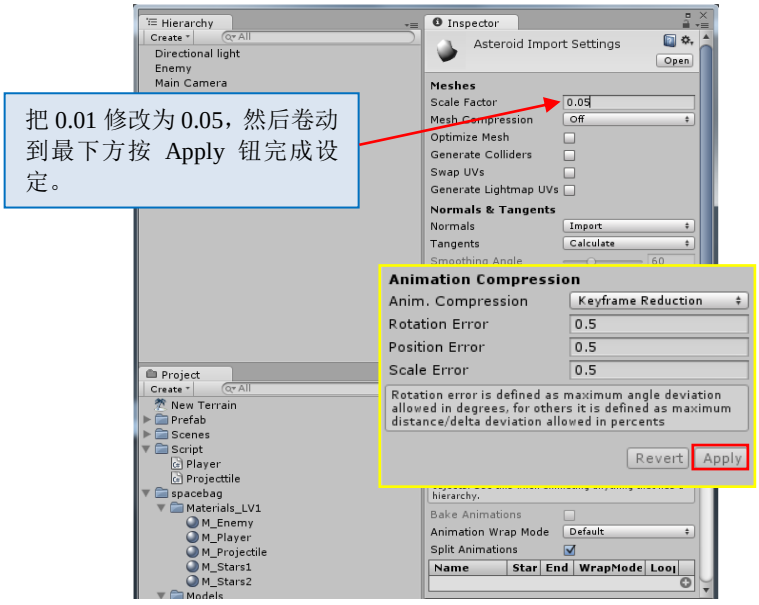
二、 改变 Enemy 的外型

操作步骤：

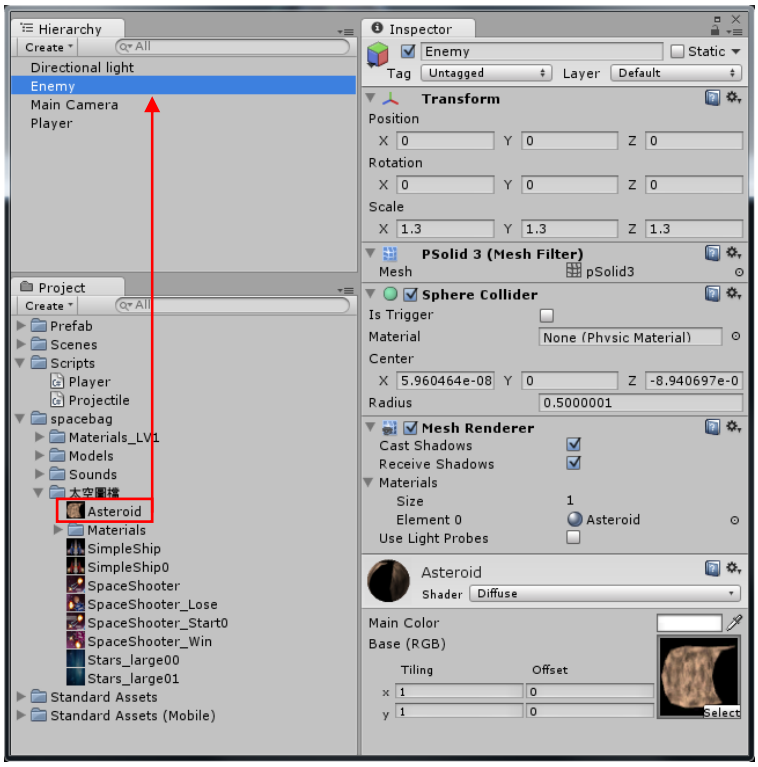
(1) 把属性「Sphere (Mesh Filter)」改为「pSolid3」




(2) 到 Project 窗口的 spacebag 中找到模型「Asteroid」，把它的「Scale Factor」改为 0.05，记得改完后要按 Apply 钮。




(3) 把 Project 窗口的「spacebag→太空图文件→Asteroid」加到 Enemy 里，成为 Enemy 的属性之一。




完成这些步骤之后，陨石敌人的外型就完成了。

陨石的 Collider 还是原来的球体

还记得我们制作 Player 时是先把它建成一个立方体，然后再套用太空战机的 3D 模型，当时我们还有一个步骤，就是修改 Player 的 Collider，把我们事先做好的外壳模型套用成为 Player 的 Collider。

可是现在我们制作陨石并没有这个动作，这是为什么呢？原因很简单，Collider 的目的是为了碰撞侦测而设的，一般而言，游戏中的物体是否相互碰撞不是依照我们看到的物体外型，例如有时候我们要射击某个目标，只要「够接近」目标就算击中了，因此我们在制作游戏物体时，除了物体本身的外观模型外，还要另外指定一个专为碰撞侦测而设的模型，这个模型在 Unity3D 里称为「Collider」。通常这个 Collider 模型会比物体本身大一些，免得彼此之间很难碰撞，因而降低玩家的体验感。

由于陨石的外型和球体很接近，但是毕竟陨石还是比起球体多了一些凹凸面，在游戏进行碰撞侦测时是可以不必考虑到这些凹凸面的，就像太空战机的 Colider 外型也很像太空战机，只是它的表面比较单纯，如果我们把陨石的凹凸面弄得平整一点，它就长得跟原来的球体很接近了，也就是因为这样，我们就

直接沿用球体做为陨石的 Collider。

### 控制陨石

控制陨石的脚本档案为 Enemy.cs，同样是放在 Project 窗口的 Scripts 数据匣内。由于这个游戏是要让玩家操控太空战机以避开陨石，或是发射炮弹摧毁陨石，为了让游戏有趣，陨石出现的位置必须以随机产生，也就是以随机数生成陨石的 *x* 坐标，程序代码就像这样：

```
x = Random.Range(-7.0f, 7.0f);
```

另外，陨石的速度也应该是随机的，所以我们先定义 `MinSpeed` 与 `MaxSpeed` 两个变量，代表陨石速度的最大值与最小值，然后以随机数生成速度值：

```
currentSpeed = Random.Range(MinSpeed,MaxSpeed);
```

陨石的移动也没有什么特别之处，先前在控制炮弹时也写过了：

```
float outToMove = currentSpeed * Time.deltaTime;
transform.Translate(Vector3.down * outToMove);
```

当陨石移动到画面的下方后就逐渐远离我们的视野了，不过在实务上，这个 Enemy 对象并没有消失，所以我们只要重新以随机数生成新的坐标和速度，让它再从画面上方出现，玩家就会感觉又出现一颗新的陨石了：

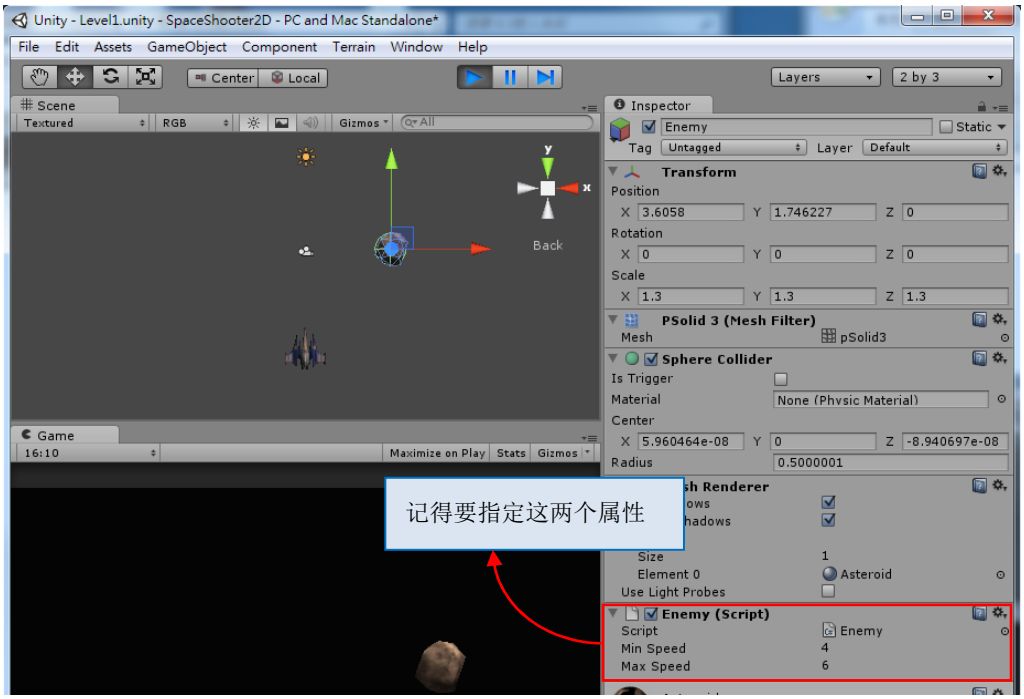
```
if (transform.position.y <=-5)   { ... }
```

现在我们把上面所讨论的概念整理一下，然后完成 Enemy 脚本：

```
// Enemy.cs
using UnityEngine;
using System.Collections;

public class Enemy : MonoBehaviour {
    public float MinSpeed;
    public float MaxSpeed;

    private float currentSpeed;
    private float x, y, z;

    void Start() {
        SetPositionAndSpeed();
    }

    void Update () {
        float outToMove = currentSpeed * Time.deltaTime;
        transform.Translate(Vector3.down * outToMove);

        if (transform.position.y <=-5) {
            SetPositionAndSpeed();
        }
    }

    void SetPositionAndSpeed() {
        currentSpeed = Random.Range(MinSpeed,MaxSpeed);
        x = Random.Range(-7.0f, 7.0f);
        y = 7.0f;
        z = 0.0f;
```

```
        transform.position = new Vector3(x, y, z);
    }
}
```

别忘了脚本中有两个公共变量 `MinSpeed` 与 `MaxSpeed` 得指定初值，透过 Inspector 窗口的属性栏设定之后，敌人系统就大功告成了。



## 1-4 碰撞处理

完整教程请至 http://developer.arcalet.com 进行下载。


**教程从「单机模式」改编成「Online」多人实时都有详细的教程，千万别错过。**