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

1-3 敌人系统

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

1-4 碰撞处理

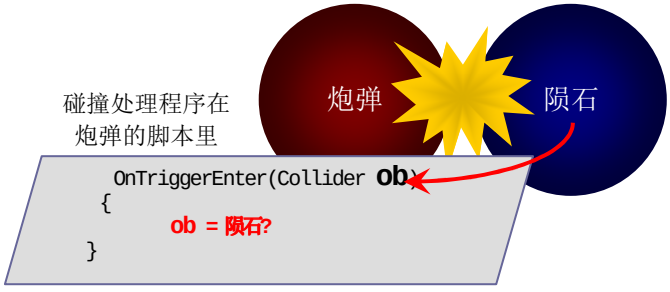
本文我们要介绍射击游戏中最重要的碰撞侦测处理，这里我们要处理的碰撞类型有两种，一种是炮弹与陨石的碰撞，另外一种则是陨石与玩家的碰撞，并且以最简单的 `OnTriggerEnter()` 事件函式进行侦测处理。

炮弹与陨石碰撞

炮弹与陨石互相碰撞之后，它们两个就应该都要消失不见。不过因为两个物体碰撞，碰撞侦测程序只会出现在其中一个物体的控制程序中，所以如果碰撞侦测程序放在炮弹的脚本控制程序里，它本身只要呼叫 `Destroy(gameObject)` 就可以把自己销毁，但是陨石是另外的 GameObject，在炮弹控制脚本里要处理与控制陨石，等于是要间接存取别的物体。

在上一节内容中我们已经介绍过陨石控制脚本中用来摧毁陨石的函式 `SetPositionAndSpeed()`，所以现在只要能够从炮弹脚本里呼叫陨石脚本 Enemy 的 `SetPositionAndSpeed()`函式，同样也可以销毁陨石，如此一来炮弹与陨石的碰撞处理程序就大致完成了。

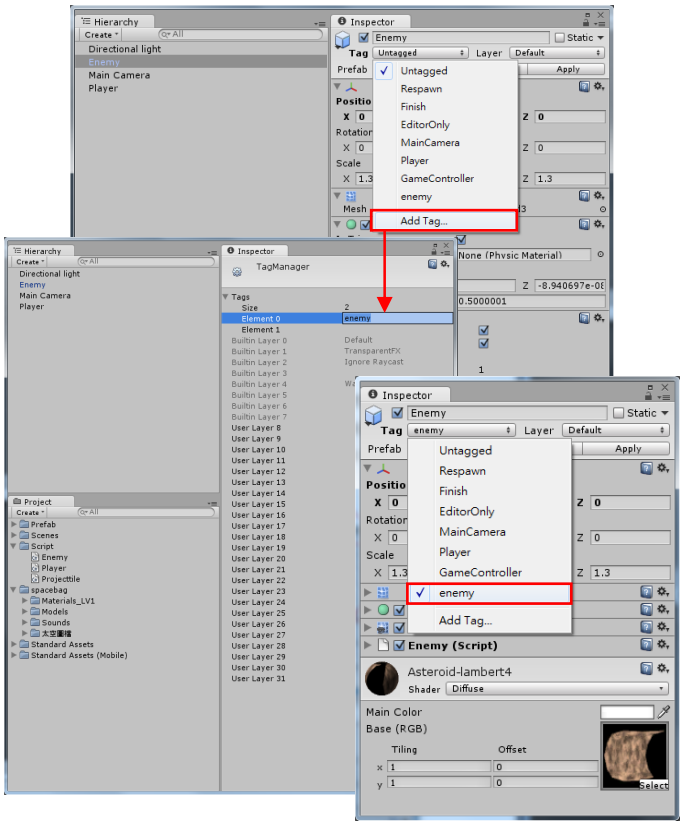
现在我们来介绍 `OnTriggerEnter()`事件函式的使用方法，这个函式有一个参数，型别为 `Collider`。当碰撞发生时，系统就会触发 `OnTriggerEnter()`事件，这个参数代表的就是和我们发生碰撞的物体，为了方便解说，我们用下面这个图来说明：



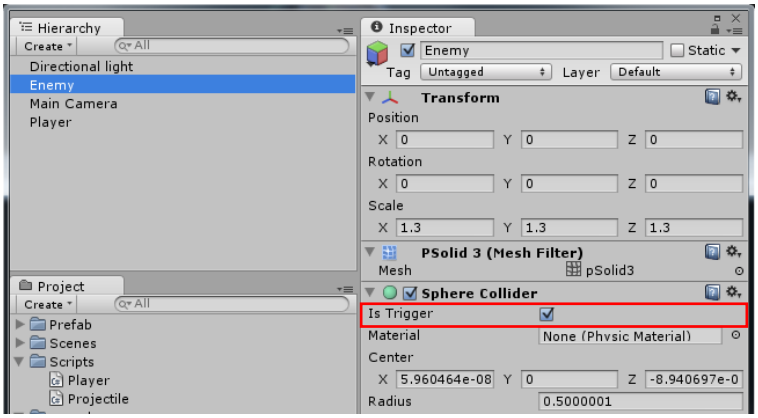
由于和炮弹发生碰撞的物体不一定是陨石，为了确定碰撞的对象是陨石可以使用`ob.tag`判断，不过在使用前要记得设定陨石的卷标(tag)，卷标的设定方法稍候再介绍，假设我们设定好陨石的标签为`"enemy"`，就可以使用这样的判断式：

```
if (ob.tag="enemy") { … }
```

了解`OnTriggerEnter()`的用法后，接着我们继续完成陨石与炮弹碰撞的处理程序。首先设定陨石(Enemy)的标签为`"enemy"`：



接着把 Collider 属性中的「Is Trigger」勾选起来，表示陨石发生碰撞触发时会触发 `OnTriggerEnter()` 事件：



接着修改炮弹脚本 Projectile，加入一段 `OnTriggerEnter()`事件函式。另外，因为这段程序代码会用到 `Enemy` 对象，所以在 `Start()`里面加了一段程序代码，用来寻找 `Enemy` 对象：

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

public class Projectile : MonoBehaviour
{
    public float ProjectileSpeed;
    private Transform myTransform;
    private Enemy enemy;

    void Start() {
        myTransform = transform;
        enemy = (Enemy)GameObject.Find("Enemy").GetComponent("Enemy");
    }

            ... (略) ...

    void OnTriggerEnter(Collider otherObject) {
       if(otherObject.tag == "enemy") {
            enemy.SetPositionAndSpeed();
            Destroy(gameObject);
        }
    }

}
```

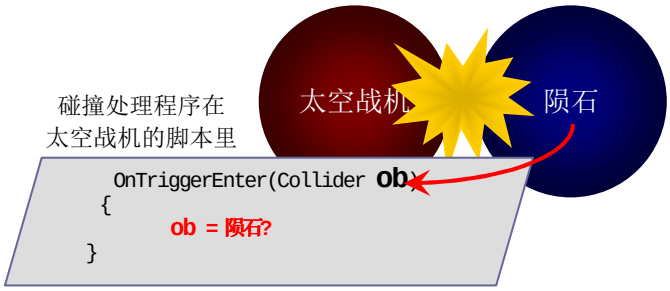
请注意，上面的程序代码会呼叫 Enemy 脚本里的 `SetPositionAndSpeed()`函式，所以要到 Enemy.cs 把它改为公开(`public`)函式：

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

public class Enemy : MonoBehaviour
{

            ... (略) ...

    public 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);

    }

}
```

程序完成后赶紧来体验一下，果然炮弹打到陨石后，炮弹和陨石两者都消失不见了，接着陨石又再从画面上方出现，玩家又可以继续发射炮弹射击新的陨石。

陨石与太空战机碰撞

陨石和太空战机碰撞后两者也都要消失不见，程序概念跟前一节陨石和炮弹的碰撞很类似，我们打算把碰撞侦测程序放在太空战机的控制脚本里，这样子就可以沿用上一节让陨石消失的程序代码，然后再另外设计一段程序代码让太空战机消失。



由于炮弹是从 prefab 动态生成的物体，要摧毁它直接呼叫 `Destroy()`就可以了，但是太空战机和陨石都是原本就存在场景中的物体，陨石消失只是让它离开摄影机的视像范围，那么太空战机被击中后消失是不是也可以这么做呢？

太空战机被陨石击中后，玩家的生命值就会减少，减到零之后就游戏就结束了。有关游戏的开始与结束我们稍后再讨论，在还没有生命值的设计之前，太空战机就好像拥有永恒的生命一样，被击中后可以无限次重生，游戏也就永远不会结束。

事实上，太空战机的重生只不过是把它移到画面中间的位置，也就是游戏一开始的位置，这样的思考逻辑跟陨石消失再出现很像，差别只是陨石先被移到游戏画面外面，而太空战机则是直接变换坐标的新的起始位置。

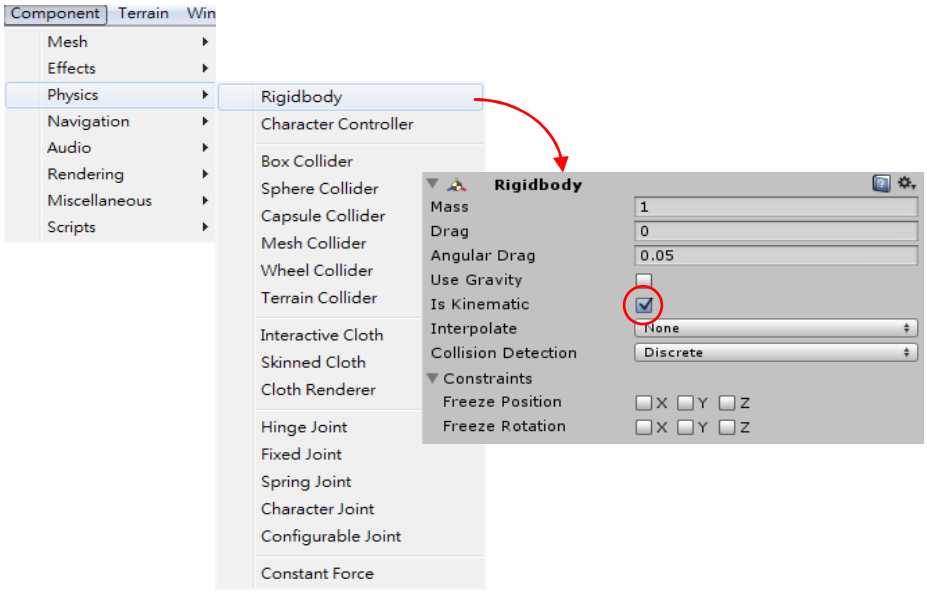
了解这样的概念后我们开始着手修改太空战机的控制脚本 Player.cs：

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

public class Player : MonoBehaviour {
    public float PlayerSpeed;
    public GameObject ProjectilePrefab;

    private Enemy enemy;

    void Start() {
        enemy = (Enemy)GameObject.Find("Enemy").GetComponent("Enemy");
    }

                ... (略) ...

    void OnTriggerEnter(Collider otherObject) {
        if(otherObject.tag == "enemy") {
            enemy.SetPositionAndSpeed();
            transform.position = new Vector3(0f, transform.position.y, transform.position.z);
        }
    }

}
```

修改之后执行却发现这段程序代码没有达到预期的效果，太空战机和陨石碰撞之后似乎没有触发`OnTriggerEnter()`事件，这是怎么回事呢？

原来我们忘了一件事，`OnTriggerEnter()`要有作用，碰撞的双方至少有一个必须被附加上 Rigidbody Component，在这里我们选择将太空战机加上 Rigidbody，操作方法大家应该都很熟练了，如果还是不熟悉，就请参考下面的操作步骤。

首先点选 Hierarchy 窗口的 Player，然后点选「主选单」→「Component」→「Physics」→「Rigidbody」，之后 Player 的 Inspector 窗口中就会出现「Rigidbody」属性了，就像下图一样：



现在我们再执行一次，果然顺利触发了碰撞事件，太空战机被陨石击中后马上消失不见，但同时也立刻出现在画面下方的中间位置。

太空战机消失后延迟出现

太空战机在被陨石击中后马上消失并出现在别的地方，这会让人觉得好像太空飞机没有被击中，而是突然闪躲跳到别的地方，为了让太空战机有被击中再重生的感觉，我们必须想办法让它延迟个几秒钟再出现。

现在我们介绍另外一种让太空战机消失的方法：

```
gameObject.renderer.enabled = false;
```

这个方法是将物体的渲染功能关闭，也就是告诉游戏引擎不要把太空战机「画」到屏幕上，所以我们就看不到它了。不过请注意，太空战机仍然还在原来的位置上，除了我们看不到以外，如果陨石再度击中它，碰撞侦测还是会有效用的。

接着我们就可以使用这个方法，让太空战机的 `gameObject.renderer.enabled` 属性先设定为 false，然后过个几秒钟后再设定为 true，如此一来太空战机就会消失并延迟数秒后才出现。

在计算机执行速度很慢又是单工操作系统的那个年代，很多程序设计师都会使用 `for` 循环达到延迟的效果，后来计算机变快了，操作系统也有多任务能力了，程序语言也有了 sleep 指令，延迟效果对程序设计师而言并不困难，不过这件看似简单的工作，对于 Unity3D 的程序设计师来说并没有想象中的那么简单。

首先我们必须先来厘清一个很重要的概念，由于 Unity3D 的脚本程序代码都是在事件触发后被执行，不只如此，这些事件函式的程序代码都在同一个线程中执行，因此程序设计师必须遵守协同多任务的要求。简单来说就是我们所撰写的程序代码不可以耗时过久，在目前的事件过程尚未结束之前，其它事件是不可能被触发的，如果耗时过久，玩家就会感觉到游戏程序就被冻结了。也因为如此，要让程序代码延迟执行就必须使用协同多任务的技巧，不过在此我们不深入探讨其中的机制，请大家跟着做就可以了。

首先我们得把相关的程序代码写成一个 `IEnumerator` 函式，呼叫 `IEnumerator` 函式则使用 `StartCoroutine()`，如果要在 `IEnumerator` 函式延迟 3 秒钟再执行下一个指令，只要在两行指令中间加上一行

```
yield return new WaitForSeconds(3);
```

就可以了，就像这个样子：

```
IEnumerator DestroyShip() {
    gameObject.renderer.enabled = false;
    yield return new WaitForSeconds(3);
    gameObject.renderer.enabled = true;
}
```

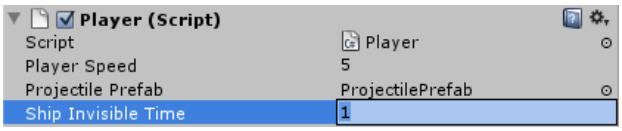
`yield return` 是将程序代码的控制权交还给系统，跟 `return` 最大的不同是 `yield return` 可以让我们回

到下一行指令继续执行，而 return 则是将程序代码控制权交还给呼叫者函式 (caller)后就不再回来了，就算重新呼叫本函式也是重头来过，不会回到交出控制权的那行指令。

接下来我们继续完成太空战机和陨石碰撞的处理程序，以下是完成后的 Player.cs 程序代码内容：

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

public class Player : MonoBehaviour {
    public float PlayerSpeed;
    public GameObject ProjectilePrefab;
    public int shipInvisibleTime;

        ... (略) ...

    void OnTriggerEnter(Collider otherObject) {
        if(otherObject.tag == "enemy") {
            enemy.SetPositionAndSpeed();
            StartCoroutine("DestroyShip");
        }
    }

    IEnumerator DestroyShip() {
        gameObject.renderer.enabled = false;

        yield return new WaitForSeconds(shipInvisibleTime);   } 把控制权交给系统，等到指定的秒数后会再返回

        transform.position = new Vector3(0.0f, transform.position.y, transform.position.z);
        gameObject.renderer.enabled = true;
    }
}
```

为了让延迟秒数成为太空战机的属性之一，方便我们随时修改与调整，所以我们在程序中将延迟时间定义成公共变量，变量名称为 shipInvisibleTime，所以最后要记得到 Inspector 窗口中指定太空战机的 shipInvisibleTime 属性值：



完成之后马上测试看看，果然太空战机有了比较真实被击中后消失再重生的效果了。

制造爆炸效果

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。