SpaceShooter2D Online 版实作范例

八风吹不动，一屁打过江

《佛印禅师公案》

射击游戏是实时要求非常高的游戏，为了让程序设计师很快能学会设计在线游戏程序，所以我们选择这个简单的游戏做为范例。

依续着上一章的单机版太空射击游戏，在改造为多人联机后，比较单机与在线的版本，您会发现用 arcalet 设计在线游戏真的很简单，原本如如不动的远程场景对象，在一弹指间竟然随着远程玩家的动作同步动了起来。

1-1 Online 基本概念

1-2 arcalet 程序设计基础

1-3 登入游戏

登入游戏指的就是联机到游戏服务器，经过账号密码等安全验证后，游戏程序便开始可以使用游戏服务器的种种功能。

制作玩家登入界面

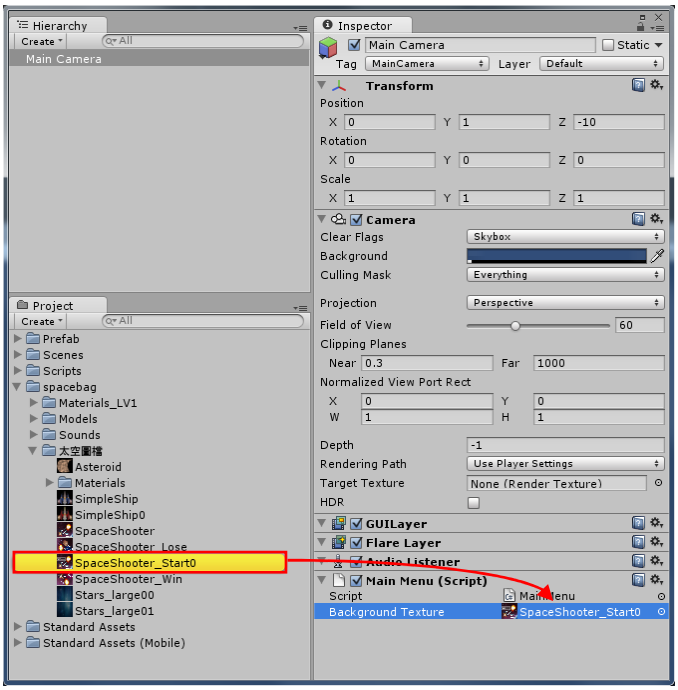
一般来说，玩家登入接口有以下三种做法：

一、 游戏开始前要求玩家输入账号密码，然后按下「登入键」进入游戏，这是 PC 在线游戏最常见的做法。

二、玩家先设定并储存账号密码，下一次启动游戏程序时自动登入，这种方式常见于行动装置的 APP。

三、游戏程序进行到需要联机登入时才要求玩家进行输入账号密码，这种作法也常出现在行动平台上，尤其是具有单机与联机两种游戏模式的程序，在单机模式时并不需要联机到游戏服务器，可避免行动设备因为长时间处于联机的状态而消耗过多的电力。

我们选择第一种做法，并将输入账号密码的接口建立在 MainMenu 场景的 GUI 画面中，为了配合画面要出现账号密码输入栏，我们也要修改 MainMenu 场景的底图，请将图档改为「SPACE SHOOTER ONLINE」。



以下是我们修改过 MainMenu 脚本后的 `OnGUI()`函式，增加了账号密码的提示卷标与输入字段。

```
// MainMenu.cs

using UnityEngine;
using System;
using System.Collections;

public class MainMenu : MonoBehaviour
{
    public Texture backgroundTexture;
    private string instructionText =
            "Instructions:\nPress Left and Right Arrows to move.\nPress.Spacebar to fire.";
    private GUIStyle NormalText;
    private string UserId="预设提示的玩家账号";
    private string Passwd="默认提示的玩家密码";

    public static AGCC agcc=null;

    void Awake() {
        NormalText = new GUIStyle();
        NormalText.fontSize = 17;
        NormalText.normal.textColor=new Color(1.0f,1.0f,1.0f,1.0f);
        agcc=(AGCC)FindObjectOfType(typeof(AGCC));
    }

    void OnGUI() {
        GUI.DrawTexture(new Rect(0, 0, Screen.width, Screen.height),backgroundTexture);
        GUI.Label(new Rect(5, 5, 250, 200), instructionText);

        // 输入玩家账号
        Rect R1=new Rect(82.0f / 300.0f * Screen.width,
                         70.0f / 320.0f * Screen.height,
                         260.0f / 480.0f * Screen.width,
                         18.0f / 320.0f * Screen.height);
        // 输入玩家密码
        Rect R2=new Rect(82.0f / 300.0f * Screen.width,
                         95.0f / 320.0f * Screen.height,
                         260.0f / 480.0f * Screen.width,
                         18.0f / 320.0f * Screen.height);
        // 登入按钮
        Rect R3=new Rect(82.0f / 280.0f * Screen.width,
                         140.0f / 320.0f * Screen.height,
                         60.0f / 480.0f * Screen.width,
                         20.0f / 320.0f * Screen.height);

        UserId = GUI.TextArea(R1, UserId, 24, NormalText);
        Passwd = GUI.PasswordField(R2,Passwd,"*"[0], 24,NormalText);

        if (GUI.Button(R3,"Start")) {
            agcc.ArcaletStartup(UserId,Passwd);
        }
    }

}
```

程序开始执行后，程序代码中 UserID 与 Passwd 变量的初始值将会出现在输入字段上，开发者未来若要实作「记忆账号密码」的功能，可以将账号密码加密保存，游戏开始时则解密取出后做为中 UserID 与 Passwd 变数的初始值，玩家就不必每次登入游戏都得辛苦的输入账号密码了。

登入成功与失败

当 Launch()在背景执行连线作业后，ArcaletGame 最后会透过 OnCompletion 事件通知开发者程序最后的执行结果，所以我们现在要自己设计一个 OnCompletion 事件处理程序，用来取代 ArcaletGame 内建的处理程序。

取代的方法是直接将我们设计的事件处理函式名称指派给 ArcaletGame 的 onCompletion 属性，所以我们修改 agcc.cs 如下：

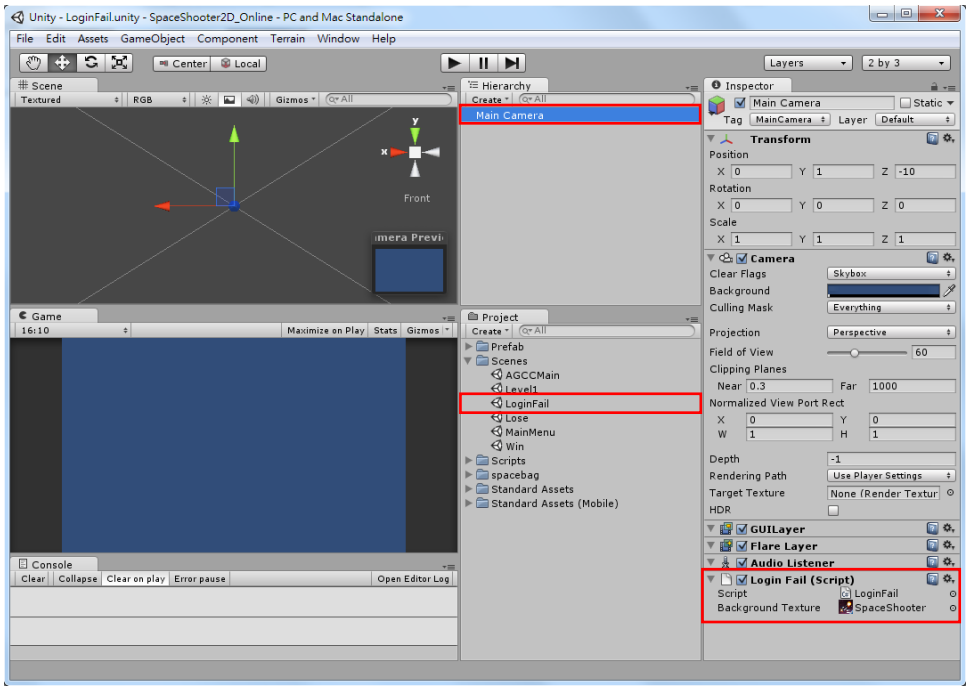
```
// agcc.cs
        ... (略) ...
public void ArcaletStartup(string userid,string passwd) {
    ag=new ArcaletGame(userid,passwd,gguid,sguid,gcert);
    this.userid=userid;
    ag.onCompletion=OnArcaletLaunchCompletion;
    ag.STALaunch();
    ArcaletGameHasLaunched=true;
}

void OnArcaletLaunchCompletion(int code,ArcaletGame game) {     // arcalet 连线作业完成

    if (code==0) { // 联机成功
        Debug.Log("login OK");
    }
    else {          // 联机失败
        Debug.Log("login fail, code="+code);
```

```
        Application.LoadLevel("LoginFail");
    }
}
```

我们自定义的 OnCompletion 事件处理的函式为 `OnArcaletLaunchCompletion()`，它只有一个参数，用来判定联机是否成功，参数值若为 0 表示成功，反之就是失败，如果失败了，就以 `Application.LoadLever()`把游戏转向一个新的 Scene「LoginFail」，但由于「LoginFail」还不存在，所以我们现在先来建立这个 Scene：



一、 建立名为「LoginFail」的 Scene

二、 再新增「LoginFail.cs」脚本，然后把它指定给「Main Camera」，脚本中以 GUI 显示一个「Try Again」的按钮，玩家按下按钮后则回到「MainMenu」Scene，完整的脚本如下：

```
// LoginFail.cs

using UnityEngine;
using System.Collections;

public class LoginFail : MonoBehaviour {

    private string instructionText="Login Fail!!!";
    private GUIStyle NormalText;
    public Texture backgroundTexture;         // It has been assigned in UNITY3D's IDE.

    void Awake()
    {
        NormalText = new GUIStyle();
        NormalText.fontSize = 32;
        NormalText.normal.textColor=new Color(1.0f,1.0f,1.0f,1.0f);
    }

    void OnGUI()
```

```
    {
        GUI.DrawTexture(new Rect(0, 0, Screen.width, Screen.height),backgroundTexture);

        // Label
        Rect R1=new Rect(53 / 480.0f * Screen.width,  72.0f / 320.0f * Screen.height,
                    260.0f / 480.0f * Screen.width,
                    80.0f / 320.0f * Screen.height);

        GUI.Label(R1, instructionText,NormalText);

        // button
        Rect R2=new Rect(82.0f / 480.0f * Screen.width,
                    130.0f / 320.0f * Screen.height,
                    100.0f / 480.0f * Screen.width,
                    50.0f / 320.0f * Screen.height);

        if (GUI.Button(R2,"Try Again?")) {
            Application.LoadLevel("MainMenu");
        }
    }
}
```

三、 为脚本中的公共变量 backgroundTexture 指定图档，在此我们仍旧使用与主画面相同的背景图「SpaceShooter」

四、 至「主选单」→「Build Settings」点选「Add Current」，把「LoginFail」Scene 加入「Build In Scene」中。

「LoginFail」Scene 建造完成。

联机状态发生变化

现在我们已经知道游戏登入成功与失败如何处理了，不过由于网络通讯是一件很复杂的事情，为了让我们的游戏程序能够确实掌握目前与服务器联机的状态，ArcaletGame 会在状态改变时触发 OnStateChanged 事件，开发者可以修改 ArcaletGame 对象中的 `onStateChanged` 属性重新指定事件处理程序，以下是重新指定 OnStateChanged 事件后的 `agcc` 脚本：

```
// agcc.cs

    ... (略) ...

public void ArcaletStartup(string userid,string passwd)
{
    ag=new ArcaletGame(userid,passwd,gguid,sguid,gcert);
    this.userid=userid;
    ag.onCompletion=OnArcaletLaunchCompletion;
    ag.onStateChanged=OnArcaletStateChanged;
    ag.STALaunch();
    ArcaletGameHasLaunched=true;
}

void OnArcaletStateChanged(int state,int code,ArcaletGame game)
{
    // 开发者自定义的事件处理程序
    Debug.Log("State: " + state.ToString());
}
```

OnStateChanged 的 `state` 参数用来指示游戏与服务器之间的状态，我们也可以把它视为 ArcaletGame 对象的状态，在游戏正常启动并登入到游戏服务器的过程中，ArcaletGame 的状态会

从 0、100、200、… 直到 600，每当状态改变时，`OnStateChanged` 函式就会被呼叫一次，最后在状态变成 600 时，同时也会触发 OnCompletion 事件。

Note

以后如果我们讲到「ArcaletGame 的状态」，指的就是 OnStateChanged 事件的状态代码，或者简称为「状态」、「联机状态」，请读者自行从上下文中判断。

以下是 `state` 参数说明:

state (状态代码)	说明
0	尚未联机
100	正要联机
200	已联机
300	已完成握手协议
400	已登入
500	已进入私人场景
600	已进入此游戏的预设场景
901	接收数据时发生错误而断线，参数 `code` 为系统错误码
902	传送数据时发生错误而断线，参数 `code` 为系统错误码
903	侦测网络质量时发生错误而断线，参数 `code` 为系统错误码
904	接收数据时发生线程异常终止的错误
905	传送数据时发生线程异常终止的错误
906~999	发生其它错误，参数 `code` 为联机错误码。

如果 OnStateChaneged 事件触发时的状态代码为 9XX，表示与服务器的联机发生问题了，在联机游戏设计实务中，我们应该把发生联机错误处理方式分为两类：

- 联机失败

  所谓联机失败就是系统从未联机到已联机的过程中发生错误，状态代码为 100 到 500。联机失败也会导致触发 OnCompletion 事件，触发时的 `Code` 参数值为非零。

- 断线

  断线指的是已经联机，但在发生某些错误后导致联机中止。断线也会触发 OnStateChanged 事件，状态代码为 9XX。

虽然「断线」和「联机失败」对开发者而言都是和游戏服务器的网络联机发生错误，但是对玩家体验来说这两者是有区别的。

「断线」发生在游戏已经成功联机，玩家也可能玩得正开心，此时发生断线，游戏势必无法进行

下去，最简单的方法就是让游戏停止，然后在画面上出现错误讯息，请玩家按下「重新联机」按钮。

「联机失败」的部份在登入游戏时我们就已经处理过了，当时我们利用 OnCompletion 事件的参数就可以判断是否为联机失败，如果是联机失败就将场景转换到「LoginFail」 Scene，因此并不需要以 OnStateChange 事件处理「联机失败」的状况。

当游戏已经正常进行，也就是和服务器已经成功联机，这时候若是发生断线状况，游戏程序还是要做必要的处理，所以我们在 OnStateChange 事件处理程序中加上一段程序代码，当状态代码大于 900 时表示发生断线状况，这时候我们借用「Login Fail」场景，让使用者知道已经联机失败。

```
// agcc.cs

    ... (略) ...

public void ArcaletStartup(string userid,string passwd)
{
    ag=new ArcaletGame(userid,passwd,gguid,sguid,gcert);
    this.userid=userid;
    ag.onCompletion=OnArcaletLaunchCompletion;
    ag.onStateChanged=OnArcaletStateChanged;
    ag.STALaunch();
    ArcaletGameHasLaunched=true;
}

void OnArcaletStateChanged(int state,int code,ArcaletGame game)
{
    // 开发者自定义的事件处理程序
    Debug.Log("State: " + state.ToString());
    if (state>=900) {
        Application.LoadLevel("LoginFail");
        ag.Dispose();
    }
}
```

登入成功

玩家登入成功后要做以下两件事：

- 告诉在线其它的玩家说「我来了」

  让其它玩家知道有人刚进到游戏里，同时让游戏程序做必要的同步处理。

- 决定是否成为游戏主控者

  第一个登入游戏的人必须担任游戏主控者，因此要知道在线是否已经有游戏主控者了，如果没有，就让游戏进入主控者模式。

当玩家登入游戏服务器后，**arcalet** 会自动让玩家进入主大厅，这也是为什么 ArcaletGame 对象的建构子(Constructor)会有主大厅 GGUID 的原因。主大厅是游戏中的默认场景，由于 **arcalet** 讯息传递是以场景为单元，任何玩家把讯息传送到指定的场景时，场景中的每一个玩家都会收到这个

讯息，当然也包括发出此讯息的玩家，所以玩家登入成功后，我们就要使用主大厅场景的讯息收发功能来进行相关的处理。

讯息通常是为了要控制远程的游戏程序做某些事情，所以我们制定每一个讯息都有对应的讯息指令，再配合指令的参数，指令与参数中间以「:」隔开，多个参数则以「/」分隔，就像这个样子：

指令：参数 1/参数 2/… …/参数 *n*

这样的设计是为了接收到讯息可以很快的解析(Parse)，有关收到讯息后的处理在后面的章节中会再详细说明。

玩家登入成功后的第一件事就是送出「我来了」的讯息到主大厅，讯息内容必须包含「我的账号」。另外，**arcalet** 的服务器对于每个登入的联机都会给予一个唯一的标识符，称之为 poid。poid 被存放在 ArcaletGame 对象的 `poid` 属性中，所以我们把「我来了」的讯息定义成这个样子：

`new:poid/userid`

因为要在登入成功后立刻送出「我来了」讯息，所以我们就在 OnCompletion 事件处理函式 `OnArcaletLaunchCompletion()`中以 `ag.send()`传送这个讯息：

```
// agcc.cs

    ... (略) ...

void OnArcaletLaunchCompletion(int code,ArcaletGame game)
    {
        if (code==0) { // 联机成功
           Debug.Log("login OK");
            // 告诉其他玩家说: 我进来了
            ag.Send("new:"+ag.poid.ToString()+"/"+userid);
    }
    else {  // 联机失败
            // 进入联机失败画面
            Debug.Log("login fail, code="+code);
            Application.LoadLevel("LoginFail");
    }
}

    ... (略) ...
```

ArcaletGame 对象中的 `send()`函式可以将讯息传送到主大厅，讯息的型态为字符串，所以我们得把讯息包成字符串传送出去。

Note

玩家登入后送出「new」指令讯息，其它玩家收到这个讯息该要做什么样的处理，我们将会在稍候的内容中探讨。

接下来我们要决定是否进入游戏主控者模式，既然我们订定的游戏机制是让第一个进入游戏的玩家成为游戏主控者，直觉的做法登入游戏后马上传送一个讯息到主大厅，请游戏主控者响应「我是主控者」，如果没有人响应，表示目前还没有游戏主控者，此时就可以让本地玩家成为主控者了。

这个方法听起来不错，也很容易理解，但是我们得考虑网络传输数据延迟的状况，如果没有人响

应，就真的是没有主控者吗？就好像电影里的核爆情节，军方对着在核爆范围内的村庄广播核爆就要开始了，请大家离开，结果真的有行动不便的人无法及时离开，遗憾就此发生，所以这样的设计绝对不是好点子。

现在我们要介绍 **arcalet** 服务器的一个特殊功能「设定与查询玩家状态」，我们可以透过这个功能把主控玩家的状态值设为「0」，当然一开始还是要先知道在线有没有主控者，这只要向系统询问有没有状态值为「0」的玩家就可以了。没有，表示目前没有主控者，此时就可以把自己设定为主控者，相反的，主控者若是已经存在，则进入一般玩家模式。

玩家状态是由游戏开发者依照游戏程序逻辑所设定的，玩家只能设定自己的状态，但是可以根据状态值查询目前符合此状态值的所有玩家。

依照状态值查询玩家可以呼叫这个 method:

```
ArcaletGame.FindPlayersByStatus(int status, OnCallCompletionWithData cb, object token);
```

它有两个参数 *status* 与 *cb*，第一个参数 status 要查询的状态值；第二个参数 *cb* 是 callback 函式，`FindPlayersByStatus()`执行结束会呼叫这个函式；第三个参数 *token* 则是用来记录与此次呼叫有关的数据，它会原封不动的传递给 *cb* 函式。*cb* 定义如下：

```
cb(int code, object obj, object token)
```

参数 code 为查询结果，0 表示成功，也就是已经找到符合状态值的玩家；非 0 表示失败(也就是找不到的意思)。若成功找到，则将符合的玩家信息放在第二个参数 *obj*；呼叫 `FindPlayersByStatus()` 所传入的参数 *token* 则原封不动的传到此处的第三个参数 *token*。

obj 的真实型态为 `List<HashTable>`，所有符合状态值的玩家数据都放在这个 `List` 里。HashTable 包函两个 Key：「`userid`」与「`poid`」，其中 `userid` 是玩家账号，`poid` 是玩家的私人 ID。

最后，我们还要在 AGCC 对象中加上一个 public 变量 isMaster，如果确定目前游戏没有主控者，我们就要使用 `SetPlayerStatus()`把自己的状态设为「0」，也就是向服务器登录本地的登入者是游戏主控者，同时将 *isMaster* 设为 `true`，这样就完成了登入成功后的确定游戏主控者工作了。

以下本文所完成的 agcc 脚本程序代码：

```
// agcc.cs

    ... (略) ...

public bool isMaster=false;

void OnArcaletLaunchCompletion(int code,ArcaletGame game)
{
    if (code==0) { // 联机成功
        Debug.Log("login OK");
        // 告诉其他玩家说: 我进来了
        ag.Send("new:"+ag.poid.ToString()+"/"+userid);

        // 找看看有没有主控者
        // 只要找status为0的玩家，找得到，就是有主控者，反之，就是没有
        ag.FindPlayersByStatus(0,FindMasterCallback,null);

    }
```

```
    else {  // 联机失败
        // 进入联机失败画面
        Debug.Log("login fail, code="+code);
        Application.LoadLevel("LoginFail");
    }
}

// 找status为0的玩家
void FindMasterCallback(int code,object s,object token)
{
    Bool Found;

    if (code==0) {
        List<Hashtable> p=(List<Hashtable>)s;
        // 传回的List是空的，表示没有找到
        if (p.Count==0) Found=false;
        else Found=true;
    }
    else {
        Found=false;
    }

    if (Found) {
        // 有找到主控者(有找到status=0的玩家)
    }
    else {
        // 没有找到主控者(有找到status=0的玩家)
        // 就把本地的玩家当成主控者(也就是把status设为0)
        // 这样子其他的玩家就可用FindPlayersByStatus来寻找主控者
        ag.SetPlayerStatus(0,null,null);

        // 这个变量要让其它对象参考
        isMaster=true;
    }

    // 已经确认谁主控者了
    // 接着进入游戏的level 1
    Application.LoadLevel("Level1");
}

    ... (略) ...
```

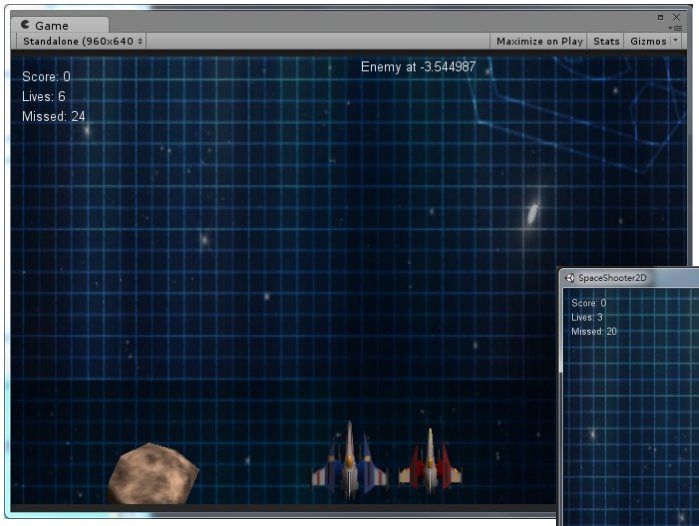
Note

为了测试方便，避免游戏很快的进入 Win 场景，请修改 Projectile.cs，把这行程序代码取消：

```
if (Player.Score>1000) Application.LoadLevel(2);
```

1-4 处理讯息

从无到有－实作运行结果



玩家 1

玩家 2

完整专案下载请至：

http://developer.arcalet.com