

# Photon Cloud(一)

## 基本架構

主講：紀曲峰

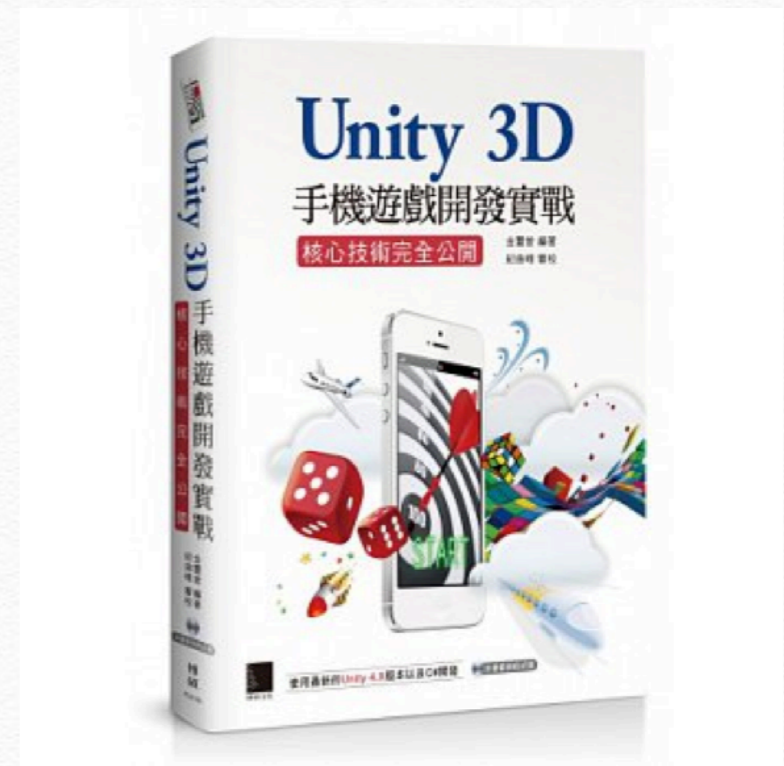
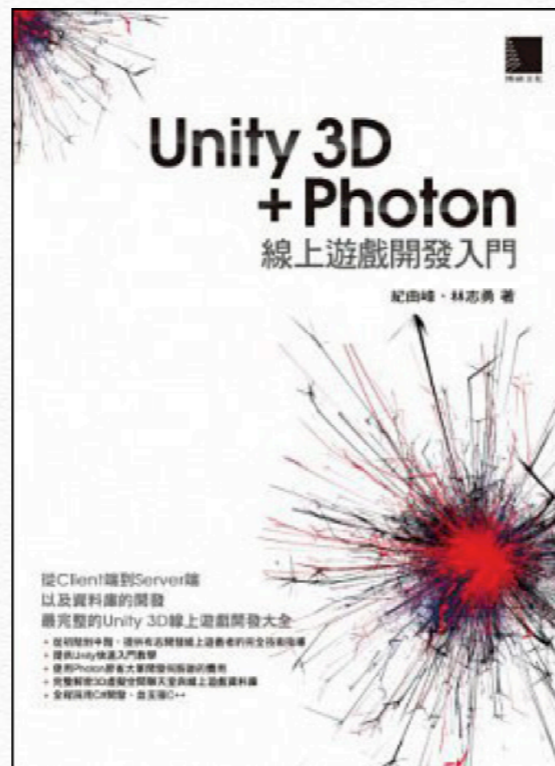
# 講師介紹

- 姓名：紀曲峰
- 資歷：資深程式設計師  
Unity3D專業作家  
Photon Server作家



# 紀曲峰的著作

- Unity 3D + Photon 線上遊戲開發入門
- Unity3D手機開發實戰 審校



# 課程內容概要

# 課程內容

- ❖ Photon Cloud開發入門
- ❖ Unity 的 www通訊
- ❖ 弱聯網遊戲
- ❖ 資料庫說明

# Photon Cloud 開發入門

# 什麼是Photon

- ❖ Photon 是 Exit Games 公司開發的套裝遊戲伺服器

# Photon提供的服務

- ❖ Photon Realtime
- ❖ Photon Unity networking
- ❖ Photon Chat
- ❖ Photon Turnbased
- ❖ Photon Server



# Photon的產Ⓕ

- ❖ Photon Realtime SDK (雲端Game Server)
- ❖ Photon Server SDK (主機Game Server)

# Photon Cloud VS Photon Server

- ❖ <http://doc.exitgames.com/en/realtime/current/getting-started/onpremise-or-saas>

Photon Cloud	Photon Server
不需管理伺服器	需自行管理伺服器
訂閱方式	提供訂閱與買斷方式
遊戲邏輯寫在Client端	遊戲邏輯寫在Server端

# Photon Cloud的優勢

- ❖ 效能高
- ❖ 支援平台多
- ❖ 若不敷使用時可以無痛轉移至Photon Server

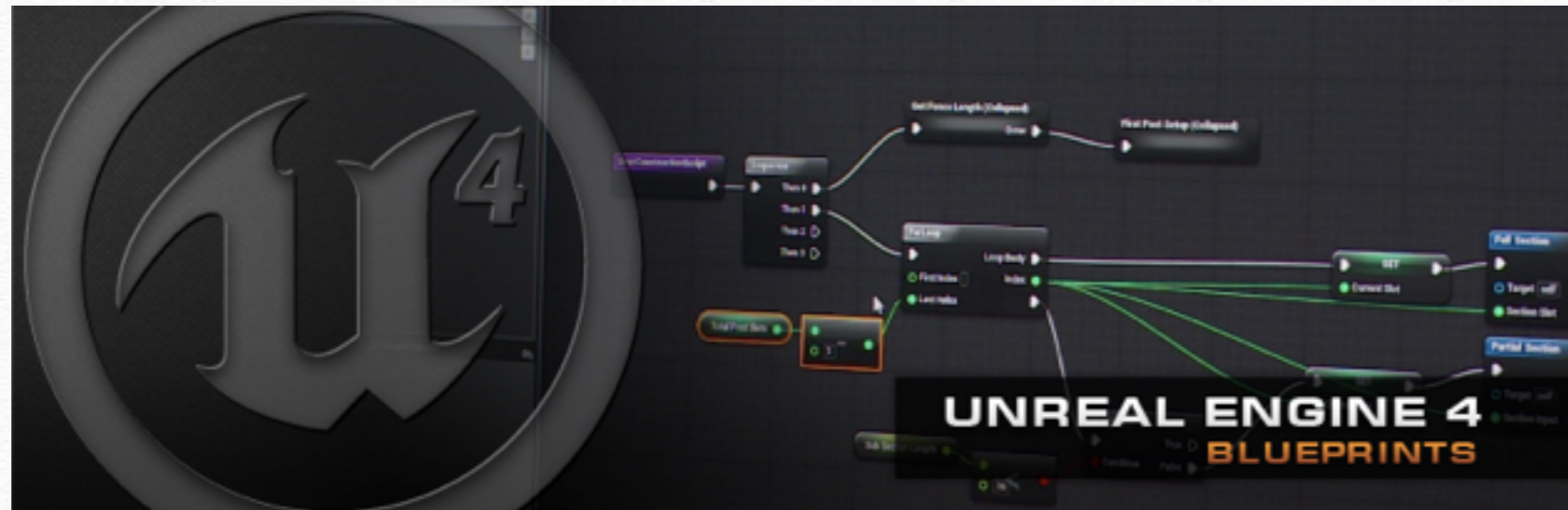
# Photon Cloud支援的平台

- ❖ PC/Mac/Linux
- ❖ iOS/Android/Win8/Blackberry
- ❖ Unity3D/flash
- ❖ C++/.Net/JavaScript

	<a href="#">Request the Playstation SDK.</a>
	<a href="#">Photon-AndroidNDK_v3-2-5-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Android_v3-2-0-0_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-BlackberryNDK_v3-2-2-0_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Corona_v3-2-1-5_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-DotNet_v3-2-2-1_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Flash_v3-2-1-4_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-iOS_v3-2-5-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Javascript_v3-2-1-4_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Linux_v3-2-5-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-MacOSX_v3-2-5-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Marmalade_v3-2-5-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-PSMobile_v3-2-0-1_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Silverlight_v3-2-1-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Unity3D_v3-2-2-1_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Win8Phone_v3-2-2-1_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Win8RT_v3-2-2-1_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-WindowsPhone_v3-2-1-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Windows_v3-2-5-3_Cloud_SDK.zip</a> <a href="#">[Details]</a>
	<a href="#">Photon-Xamarin_v3-2-2-1_Cloud_SDK.zip</a> <a href="#">[Details]</a>

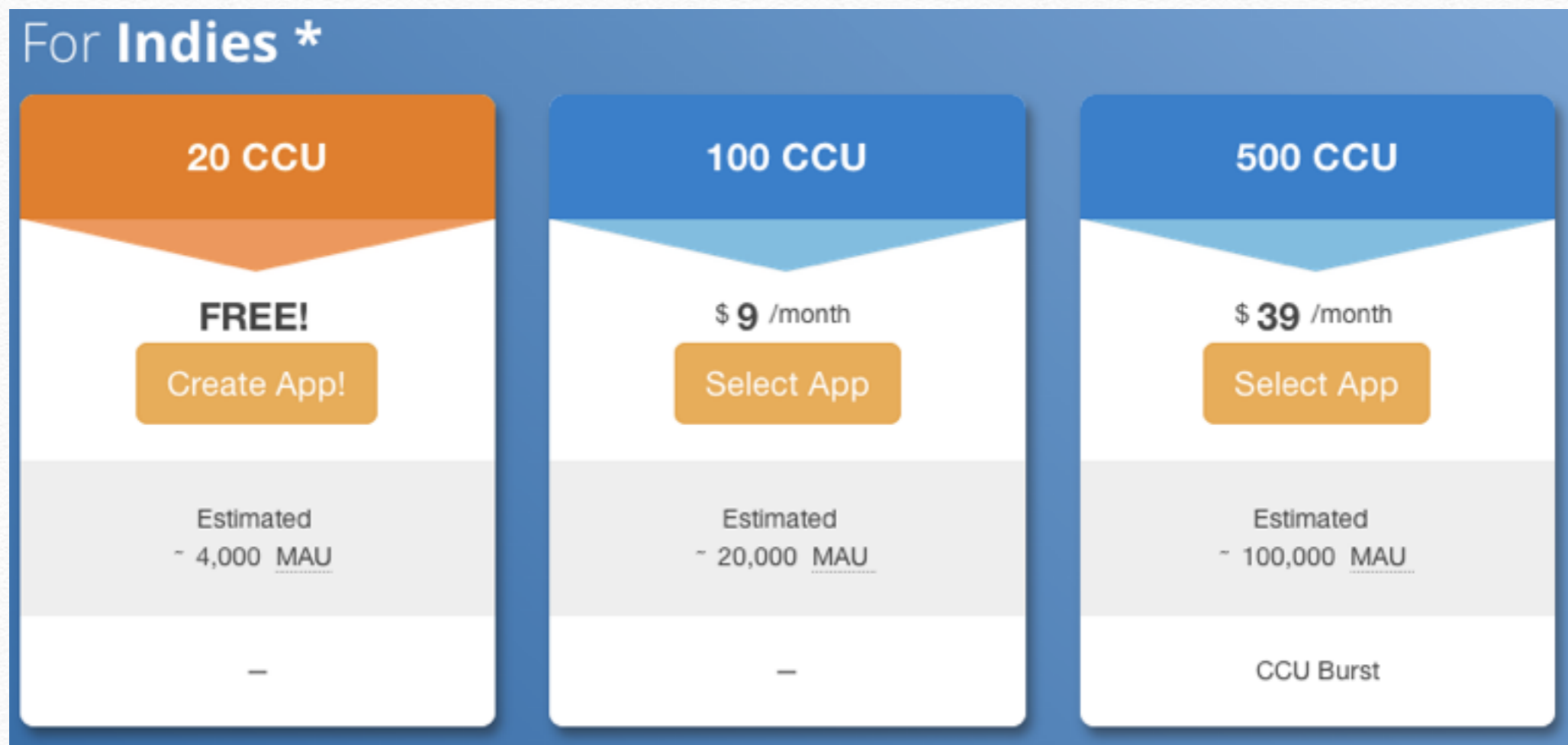
# 尚未支援但值得注意的引擎

❖ Unreal 4, CryEngine



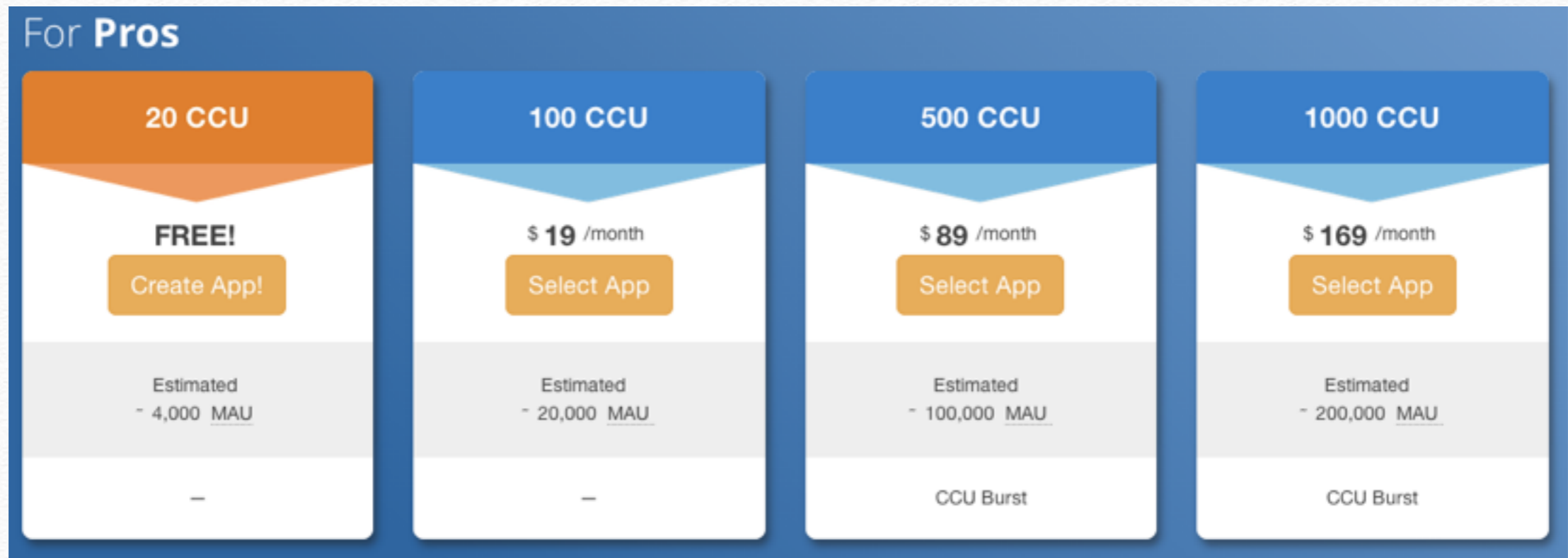
# Photon Realtime的費用

## ❖ 獨立製作團隊



# Photon Realtime的費用

## ❖ 公司的費用



# 獨立製作的資格

- ❖ 月收入在10k(美金)以下



# Photon Cloud for Unity

## 開發基礎認知

### ❖ 開發語言

Ans：基本上必須使用C#開發，但可以用C#包裝元件後再給js呼叫

### ❖ 遊戲類型

Ans：比較適合休閒遊戲和小型的射擊遊戲及賽車遊戲，因為邏輯存在Client，有相當風險

# 如何取得Photon Realtime

- ❖ 官方網站：<https://www.exitgames.com>
- ❖ 點選 Sign Up 輸入mail取得密碼
- ❖ 用Email和密碼登入後到網址  
<https://www.exitgames.com/en/Realtime/Download>  
下載SDK

# Photon for Unity

❖ 下載時僅需下載

Photon-Unity3D\_VXXX\_Cloud\_SDK.zip

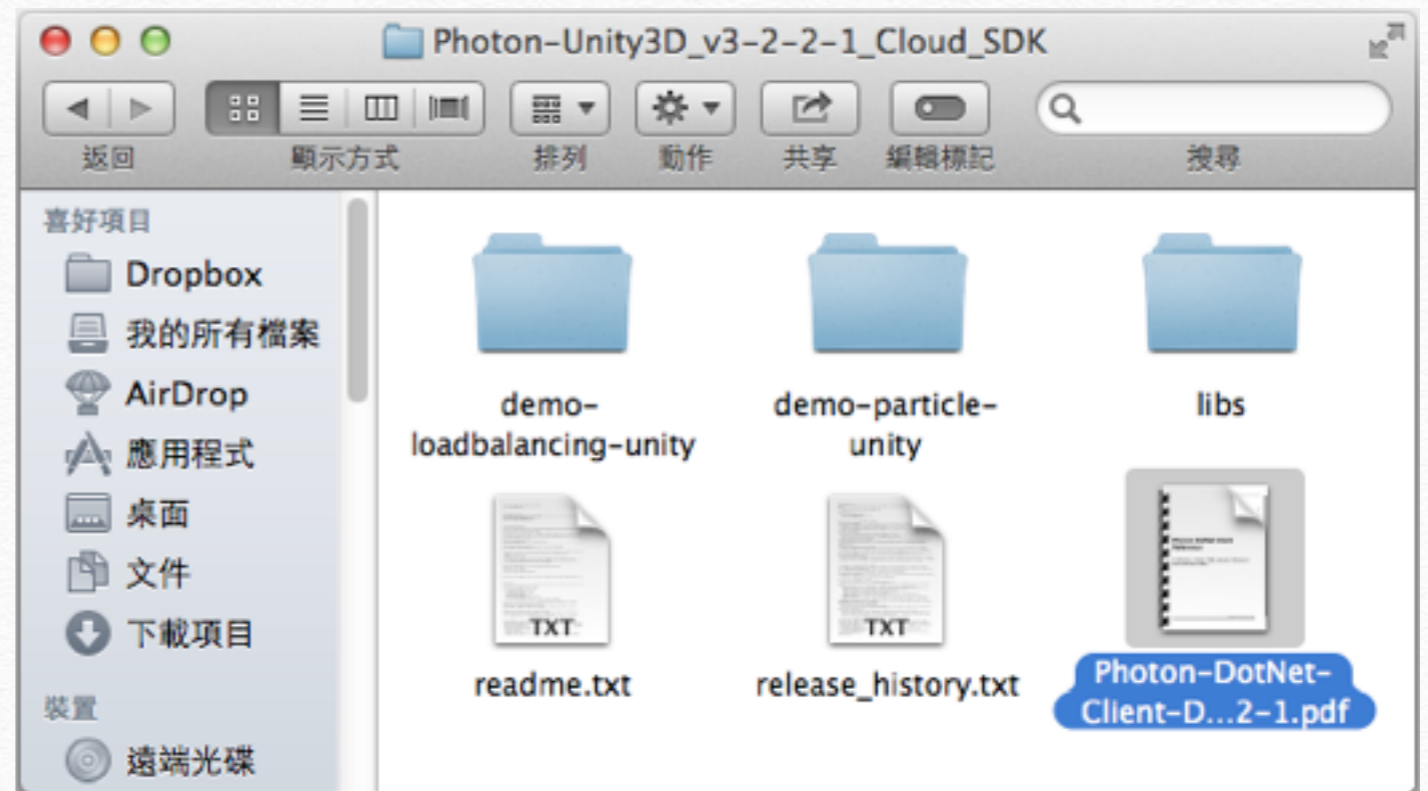
下載後解壓縮，內含SDK與範例專案



[Photon-Unity3D\\_v3-2-2-1\\_Cloud\\_SDK.zip](#) [Details]

# 文件內容

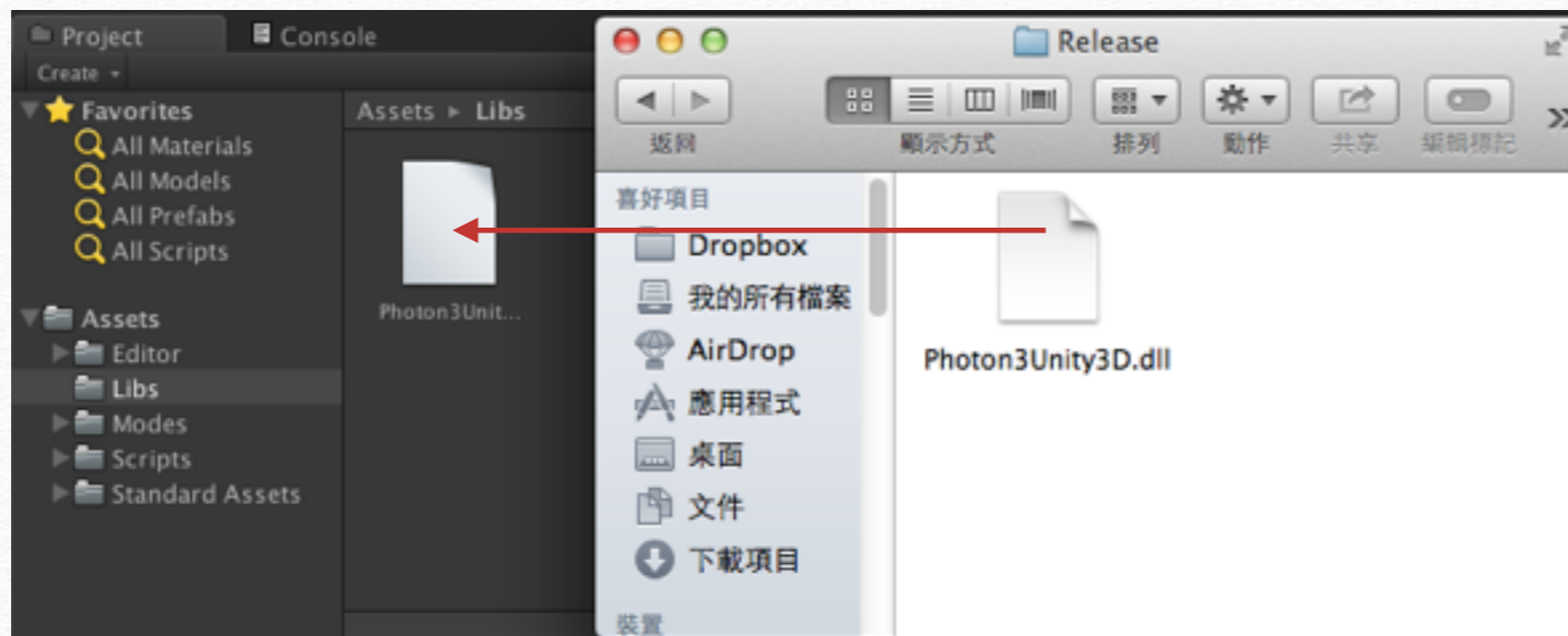
- ❖ 範例專案
- ❖ Photon Cloud SDK(Libs)
- ❖ 文件說明
- ❖ 更新歷程
- ❖ .Net函式說明手冊



# Photon Cloud 開發入門

# 建立專案

- ❖ 1. 建立新專案
- ❖ 2. 新增 Libs 資料夾
- ❖ 3. 將 Photon-Unity3D\_vXXX\_Cloud\_SDK\libs\ 底下  
Debug\Photon3Unity3D.dll或Release\Photon3Unity3D.dll放到Libs  
資料夾內



# 建立基本框架

- ❖ 到Unity建立名稱為「EZCloudClient.cs」腳本
- ❖ 加入 `using ExitGames.Client.Photon;`
- ❖ 把MonoBehaviour改成IPhotonPeerListener
- ❖ 將Start()和Update()移除

# 完成後外觀

❖ 完成後外觀如下

```
using UnityEngine;
using System.Collections;
using ExitGames.Client.Photon;

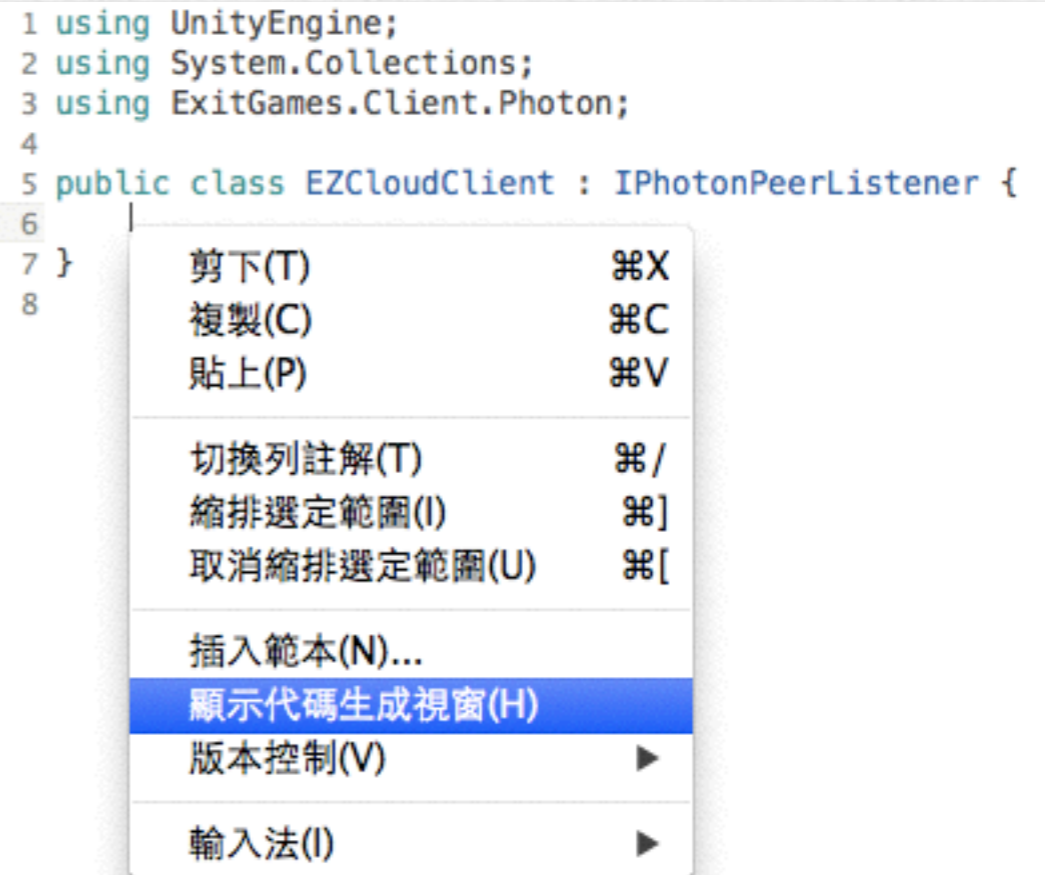
public class EZCloudClient : IPhotonPeerListener {
}
```



# 加入實作成員

- ❖ IPhotonPeerListener是Photon SDK的一個Interface，必須實作成員
- ❖ 將游標移到類別內
- ❖ 滑鼠右鍵選擇「Show Code Generation Window」或「顯示代碼生成視窗」

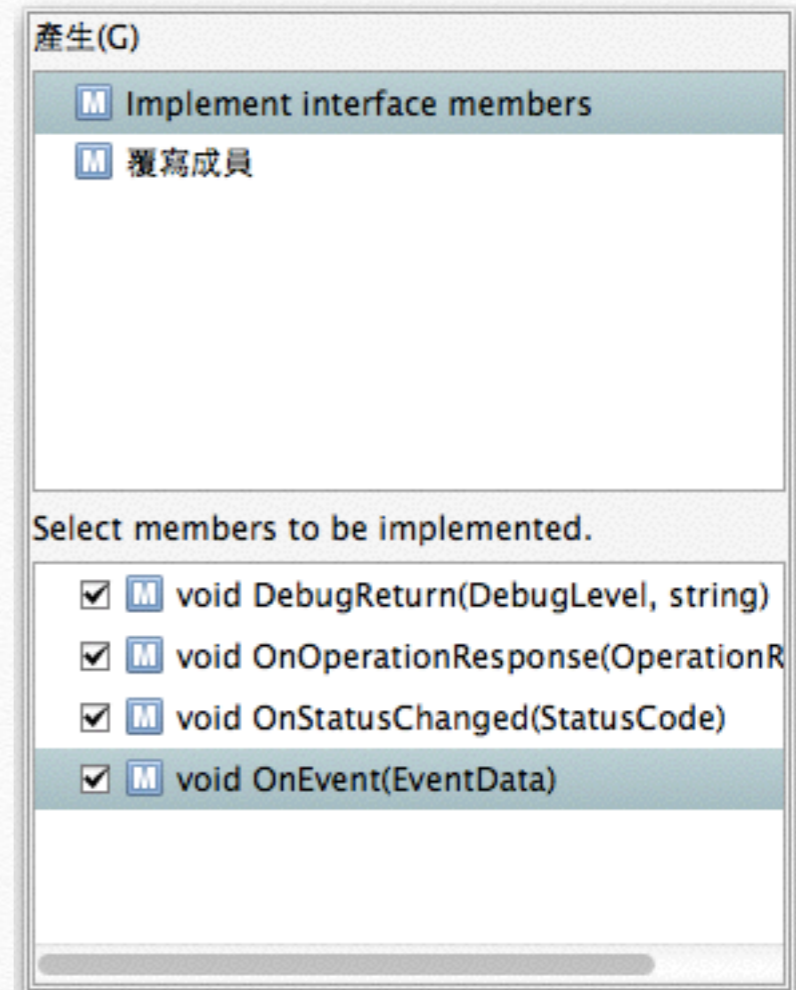
```
1 using UnityEngine;
2 using System.Collections;
3 using ExitGames.Client.Photon;
4
5 public class EZCloudClient : IPhotonPeerListener {
6
7 }
8
```



剪下(T)	⌘X
複製(C)	⌘C
貼上(P)	⌘V
切換列註解(T)	⌘/
縮排選定範圍(I)	⌘]
取消縮排選定範圍(U)	⌘[
插入範本(N)...	
顯示代碼生成視窗(H)	
版本控制(V)	▶
輸入法(I)	▶

# 建立Interface成員

- ❖ 選取所有Interface成員
- ❖ 按下Enter建立成員框架



# 成員完成後框架

## ❖ 完成後框架如下

```
public class EZCloudClient : IPhotonPeerListener {
    public void DebugReturn (DebugLevel level, string message)
    {
        throw new System.NotImplementedException ();
    }

    public void OnOperationResponse (OperationResponse operationResponse)
    {
        throw new System.NotImplementedException ();
    }

    public void OnStatusChanged (StatusCode statusCode)
    {
        throw new System.NotImplementedException ();
    }

    public void OnEvent (EventData eventData)
    {
        throw new System.NotImplementedException ();
    }
}
```

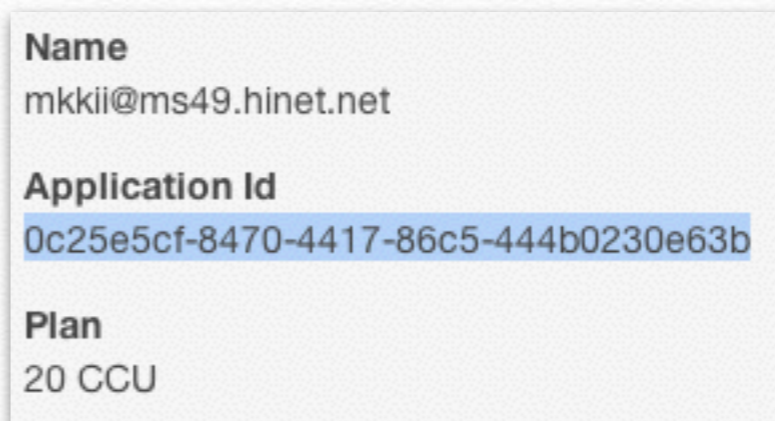
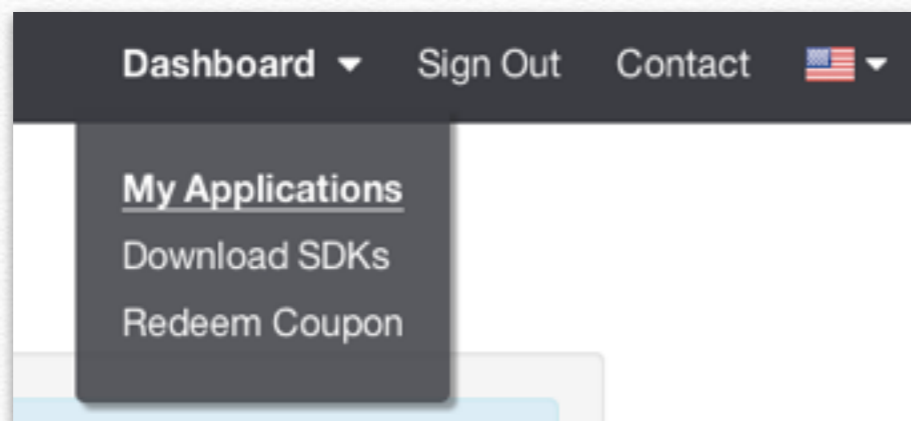
# 加入連線方法

- ❖ 加入一個 Connect() 方法供連線用

```
public class EZCloudClient : IPhotonPeerListener {  
  
    PhotonPeer peer;  
  
    public bool Connect()  
    {  
        peer = new PhotonPeer(this, ConnectionProtocol.Udp);  
        if( peer.Connect("app.exitgamescloud.com:5055", "<你的AppID>"))  
        {  
            return true;  
        }  
  
        return false;  
    }  
  
    . . . . .  
}
```

# 取得AppID

- ❖ 登入Exit Games網站
- ❖ 網頁上方選「Dashboard > My Applications」
- ❖ 顯示內容找到「Application Id」



# 貼入App ID

- ❖ 將App ID貼入自己申請的
- ❖ 請自己申請AppID，勿直接用筆者的這組

```
PhotonPeer peer;  
  
public bool Connect()  
{  
    peer = new PhotonPeer(this, ConnectionProtocol.Udp);  
    if( peer.Connect("app.exitgamescloud.com:5055",  
                    "0c25e5cf-8470-4417-86c5-444b0230e63b"))  
    {  
        return true;  
    }  
    return false;  
}
```

# 更改 OnStatusChanged 取得連線狀態

## ❖ 複寫 OnStatusChanged 取得連線狀態

```
public void OnStatusChanged (StatusCode statusCode)
{
    switch (statusCode)
    {
        case StatusCode.Connect:           // 連線成功
            break;

        case StatusCode.Disconnect:        // 斷線
            break;

        case StatusCode.DisconnectByServerUserLimit: // 人數達上線
            break;

        case StatusCode.ExceptionOnConnect:           // 連線時例外錯誤
            break;

        case StatusCode.DisconnectByServer:           // 被Server強制斷線
            break;

        case StatusCode.TimeoutDisconnect:           // 逾時斷線
            break;

        case StatusCode.Exception:                   // 其他例外
        case StatusCode.ExceptionOnReceive:
            break;
    }
}
```

# 加入斷線及呼叫服務方法

- ❖ 加入Disconnect方法提供遊戲斷線功能
- ❖ 加入Service供遊戲Update()叫用以提供網路存取及廣播服務

```
public void Disconnect()  
{  
    peer.Disconnect();  
}  
  
public void Service()  
{  
    peer.Service();  
}
```



# 加入狀態Log

## ❖ 加入讀取狀態的Log

```
public void OnStatusChanged (StatusCode statusCode)
{
    Debug.Log(statusCode.ToString());

    switch (statusCode)
    {
        . . . . .
    }
}
```

# 加入switch case內的Log

```
case StatusCode.Connect:           // 連線成功
    Debug.Log("連線成功");
    break;

case StatusCode.Disconnect:        // 斷線
    Debug.Log("斷線");
    break;

case StatusCode.DisconnectByServerUserLimit: // 人數達上線
    Debug.Log("人數達上線");
    break;

case StatusCode.ExceptionOnConnect:           // 連線時例外錯誤
    Debug.Log("連線時例外錯誤");
    break;

case StatusCode.DisconnectByServer:          // 被Server強制斷線
    Debug.Log("被Server強制斷線");
    break;

case StatusCode.TimeoutDisconnect:          // 逾時斷線
    Debug.Log("逾時斷線");
    break;

case StatusCode.Exception:                  // 其他例外
case StatusCode.ExceptionOnReceive:
    Debug.Log("其他例外");
    break;
```

# 新增測試腳本

- ❖ 加入新腳本取名為「EZCloudDemo.cs」

# 加入連線用變數

- ❖ 加入兩個變數作為連線呼叫之用

```
public class EZCloudDemo : MonoBehaviour {  
    EZCloudClient ezClient;  
    private bool peerActive;
```

# 在Start()用初始化物件

- ❖ 在 Start () {} 裡實體化物件，並將遊戲設為runInBackground以免失去焦點時就斷線。

```
void Start () {  
    Application.runInBackground = true;  
  
    ezClient = new EZCloudClient();  
    peerActive = false;  
}
```

# 在Update()裡叫用Service

- ❖ 因為網路服務必須不斷叫用Service以取得網路訊號，因此在Update中加入此方法的叫用內容。

```
void Update () {  
    if( peerActive )  
    {  
        ezClient.Service();  
    }  
}
```

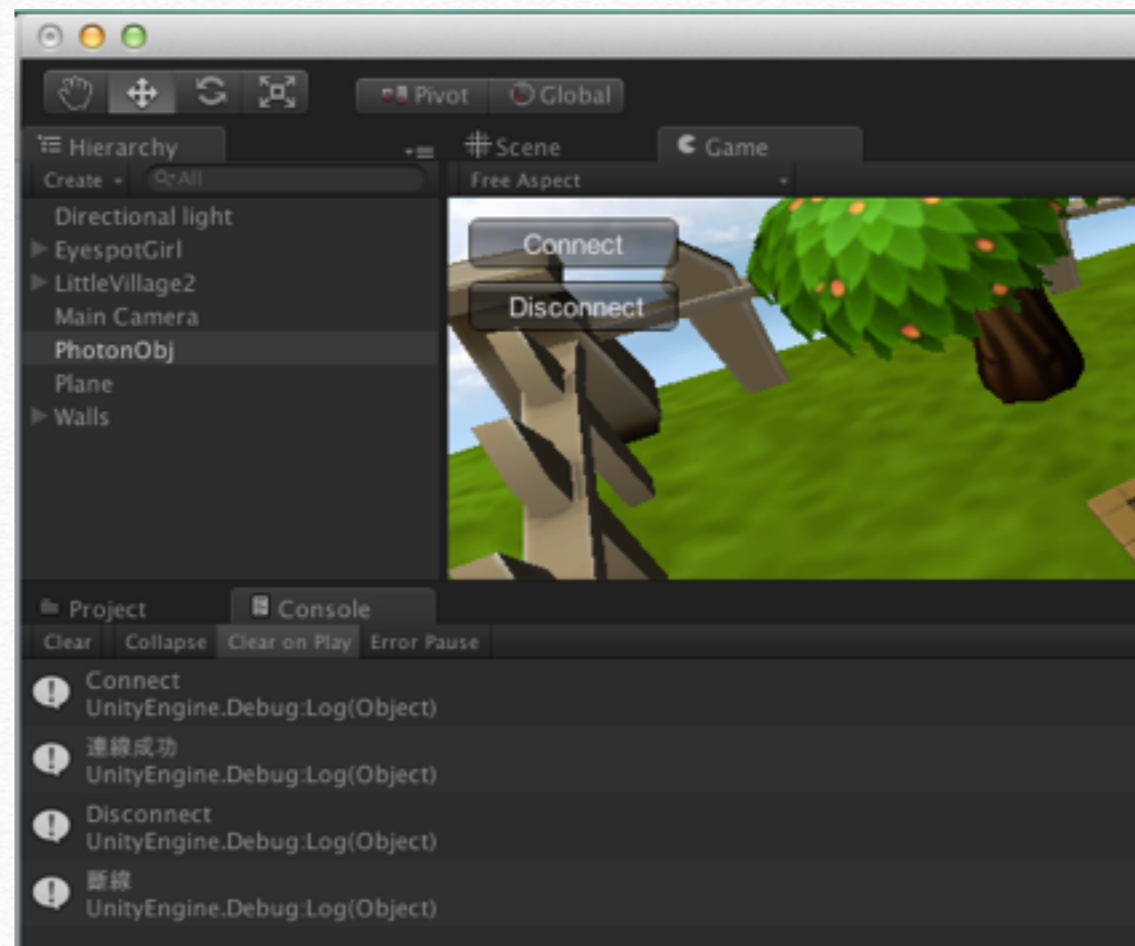
# 加入測試用GUI

## ❖ 加入測試用的GUI內容

```
void OnGUI () {  
    if( GUI.Button(new Rect(10,10,100,24), "Connect" ))  
    {  
        peerActive = ezClient.Connect();  
    }  
  
    if( GUI.Button(new Rect(10,40,100,24), "Disconnect" ))  
    {  
        ezClient.Disconnect();  
    }  
  
}
```

# 測試程式

- ❖ 將EZCloudDemo.cs拉到場景中的物件即可測試最基本的連線





# LoadBalancing架構

- ❖ LoadBalancingClient : 複寫進入點框架
- ❖ LoadBalancingPeer : 複寫連線物件(Peer)

End