Unity LAB 6

Basic UI &
Scene

綱要

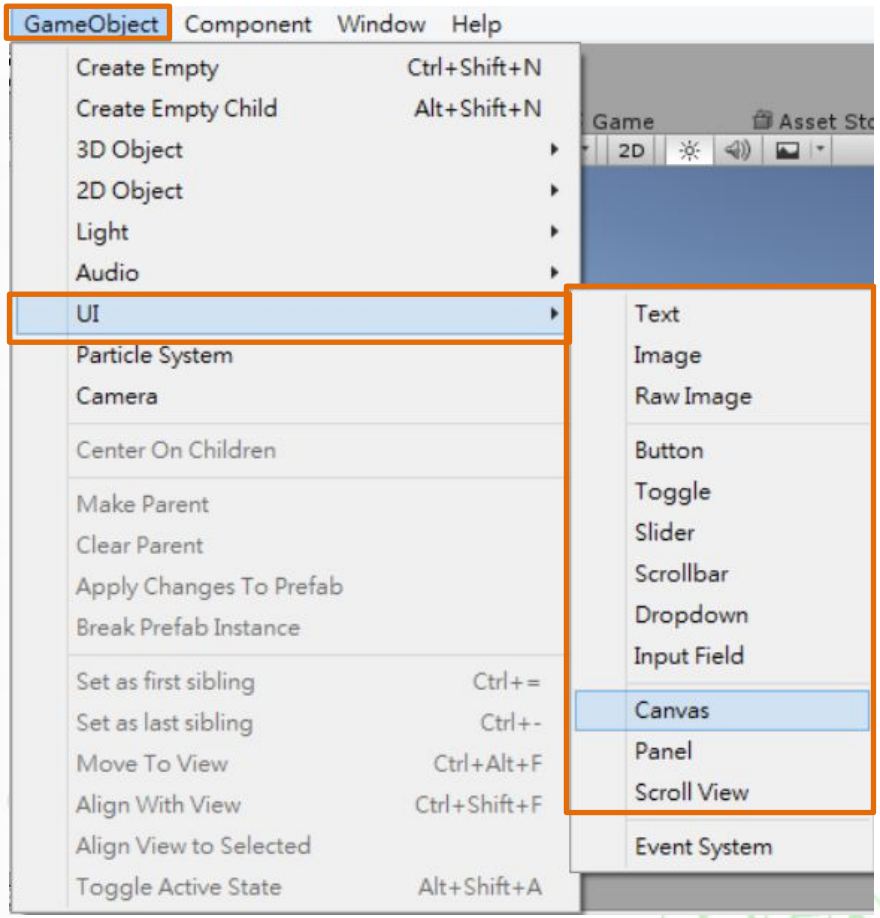
- UI
  - UI基本概念
  - UI範例
- 場景切換
- PlayerPrefs儲存變數
- 輸出成執行檔

UI基本概念

- 常用UI類型:
  - Text : 文字
  - Image : 圖片(Sprite) / Raw Image : 圖片(Texture)
  - Button : 按鈕
  - Slider : 卷軸
  - Panel : 面板

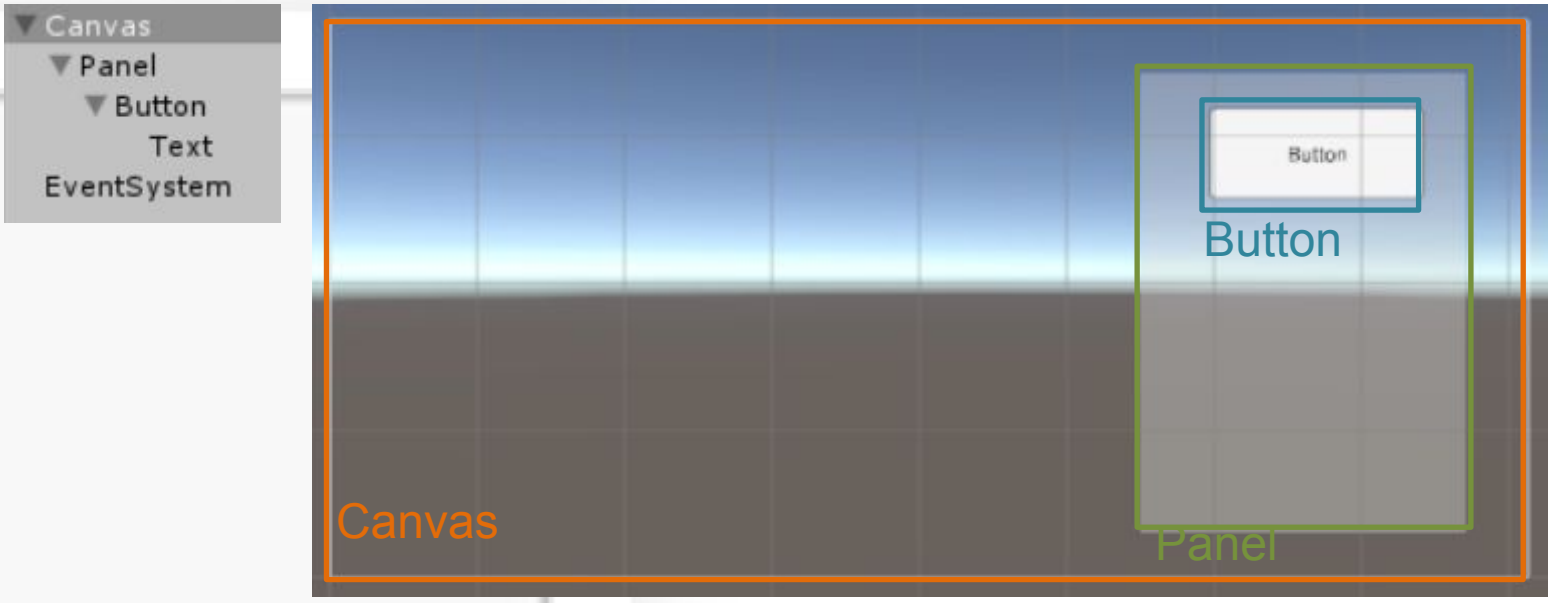
UI基本概念

- 新增一個 UI 元素 : [GameObject] -> [UI] -> [UI元素名稱]



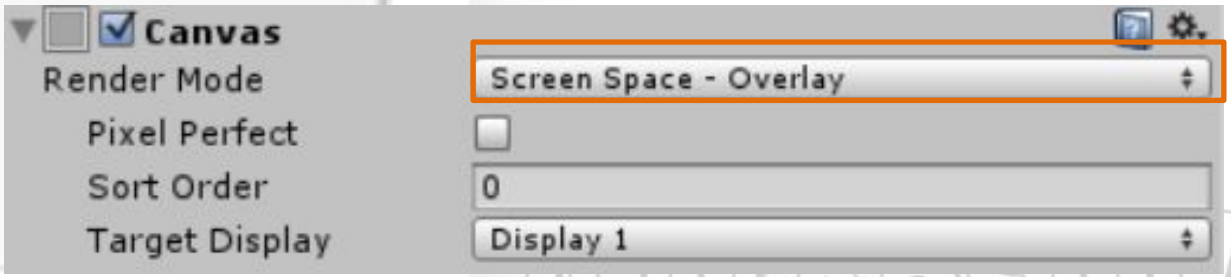
UI基本概念

- 所有的UI元素都必須為Canvas的子物件

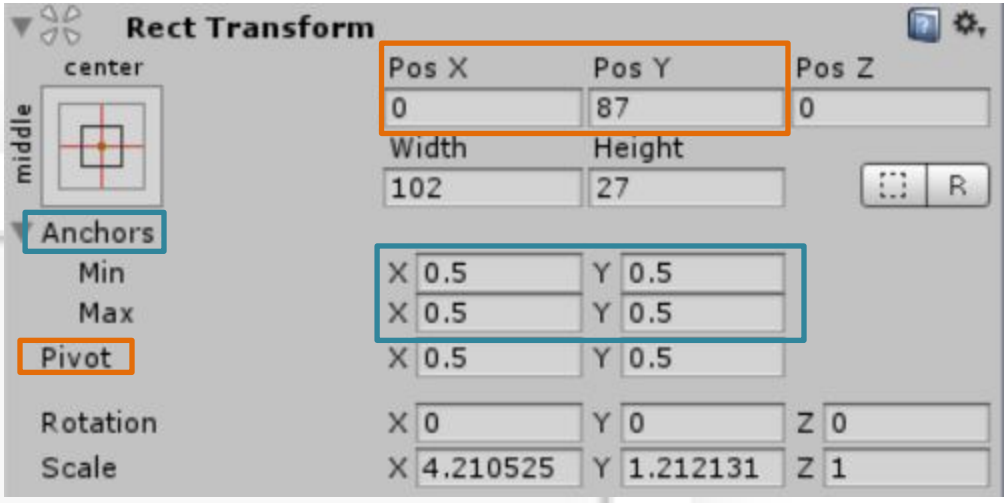


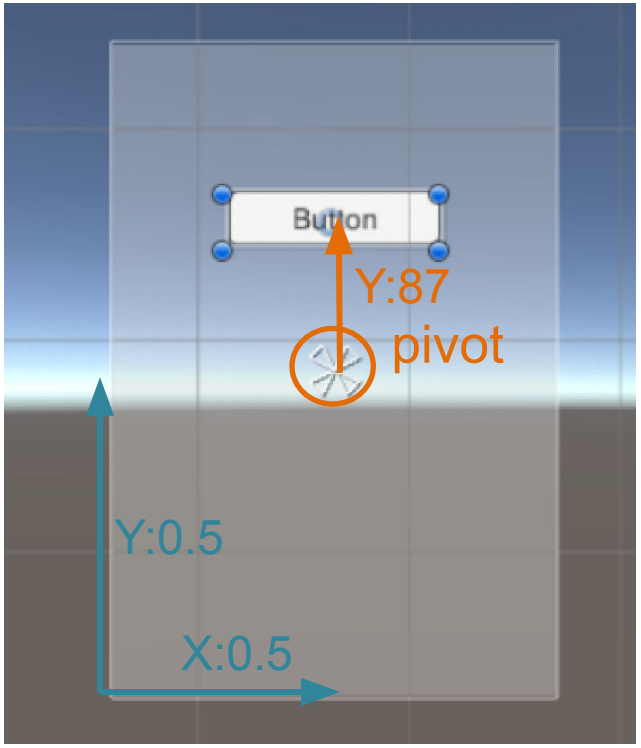
UI基本概念 - Canvas

- Screen Space - Overlay
  - Canvas會直接貼在螢幕上, 不需要額外調整它的位置
- Screen Space - Camera
  - 使用camera來固定canvas, 跟overlay差別在 camera場景物件可以出現在canvas前面(根據距離的遠近), 而overlay是強制設定canvas在最近的平面上
- World space
  - 直接將UI當3D的物件來使用



UI基本概念-Rect Transform



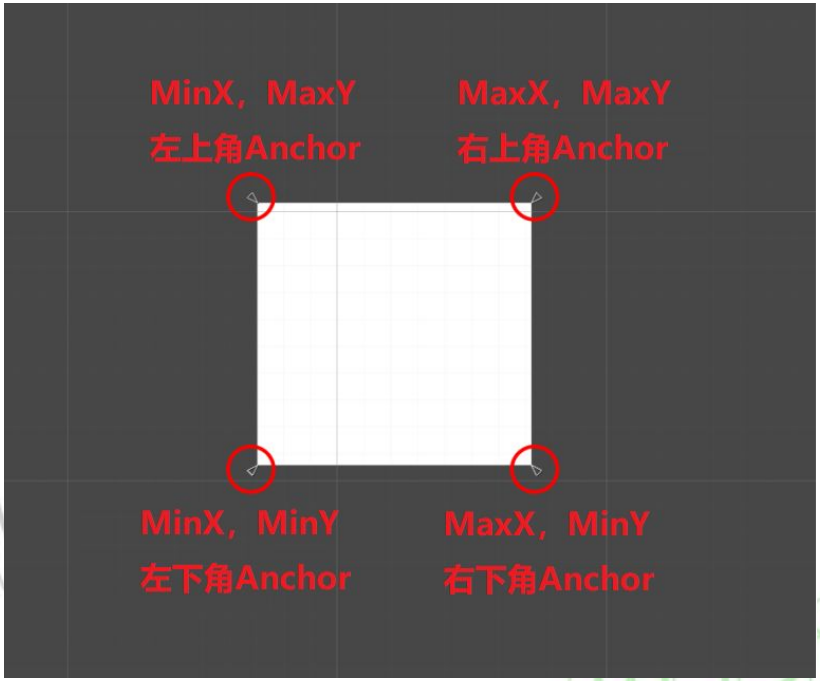


Position : UI物件的中心離**Pivot**的距離
Width、**Height** : UI的寬和高

UI基本概念-Anchor Presets

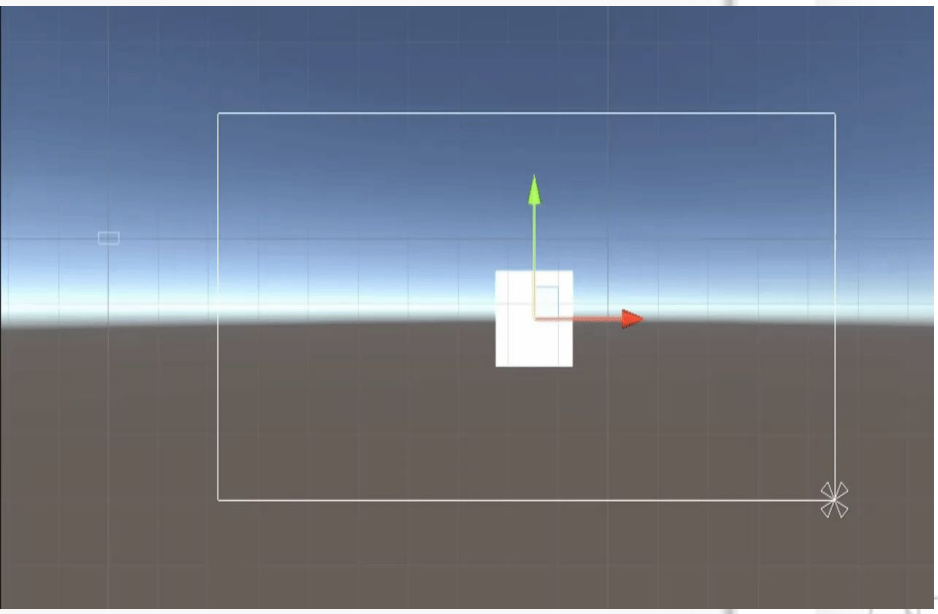
Anchors(錨點)：UI物件與父物件的對齊設定，定義了當父物件(Canvas)伸縮時子物件(UI)該如何對齊，一般會有四個Anchor。

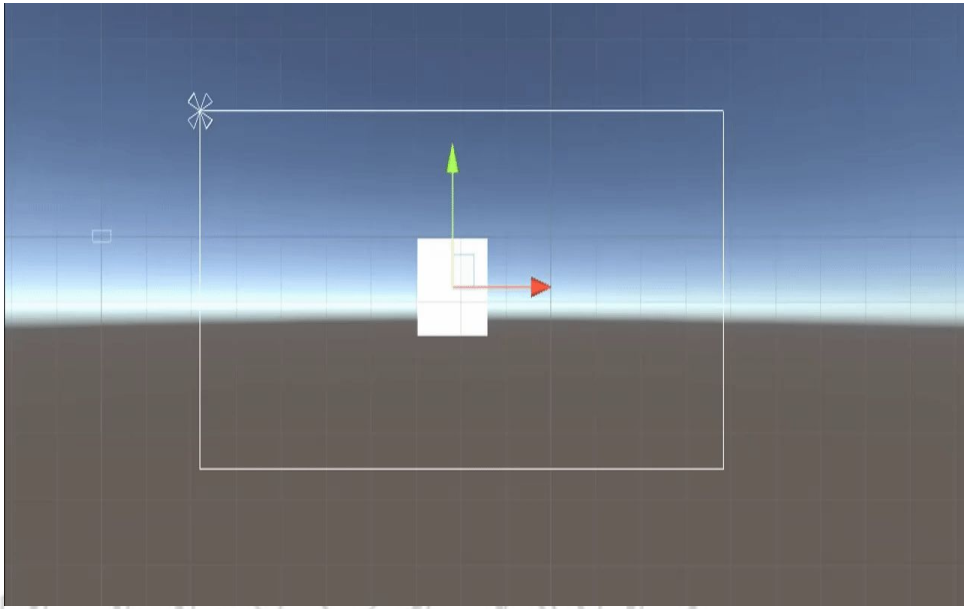
Anchors 的位置會影響子物件的位置，而 Anchors之間的相對距離會影響子物件的長寬



UI基本概念-Anchor Presets

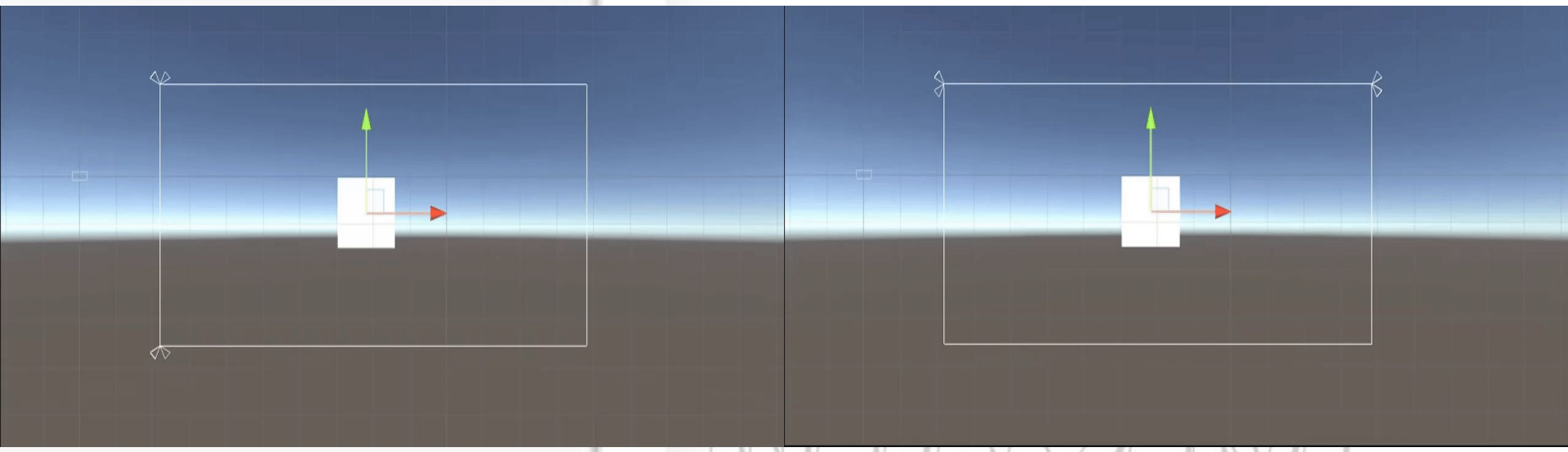
在四個Anchor為同一個位置時，Anchor會隨著父物件的長寬而改變位置，而子物件會隨著 Anchor的移動進行等比例的移動。





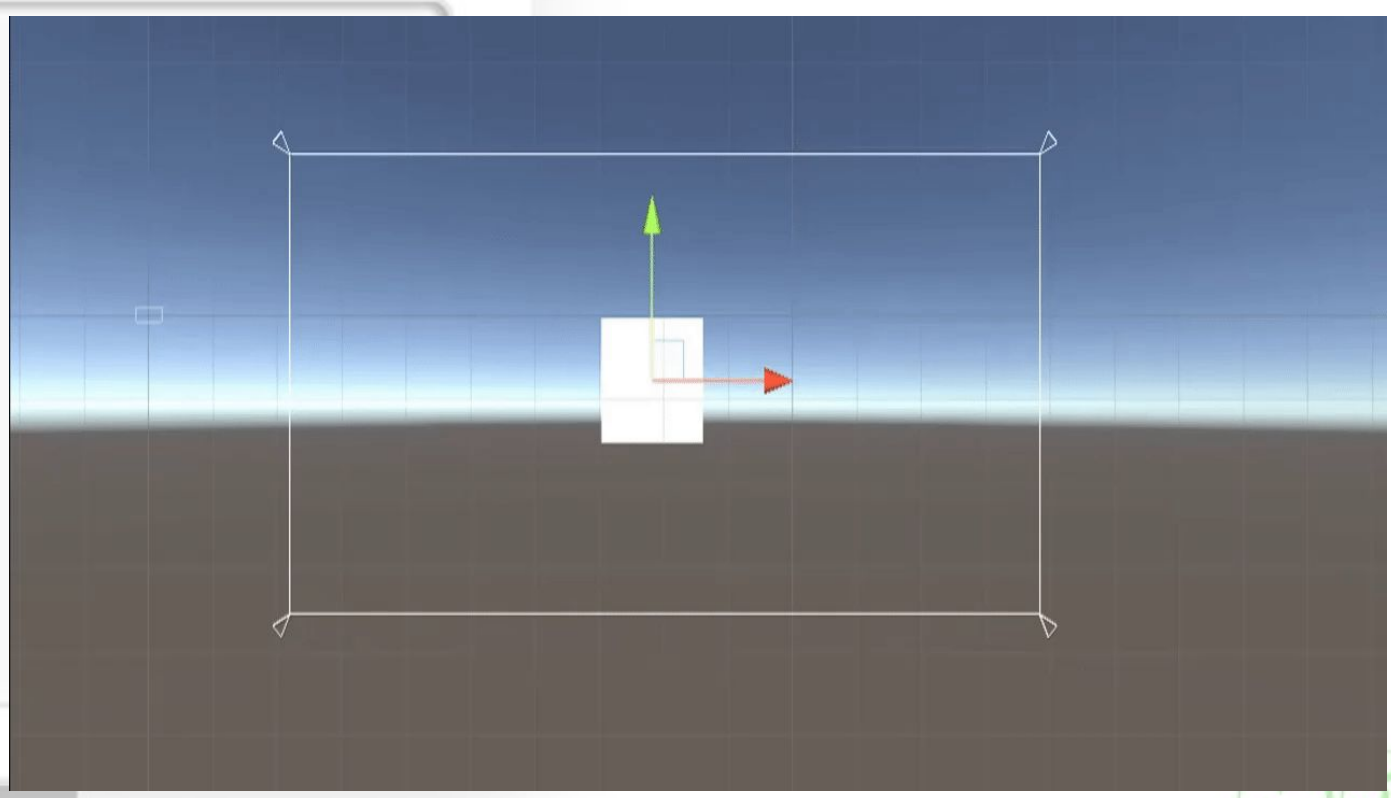
UI基本概念-Anchor Presets

在四個Anchor分成兩組時，Anchor會因爲父物件的長寬而移動，改變兩組 Anchor之間的距離，而子物件的長度和寬度會跟著Anchor之間的距離做等比例增加或減少



UI基本概念-Anchor Presets

在四個Anchor位置都不同時, 代表 Anchors之間的距離和位置都會跟隨父物件變化, 子物件的大小和長寬都會受到影響。
如果Anchors剛好都位於角落, 則代表子物件縮放位移完全跟父物件相同。

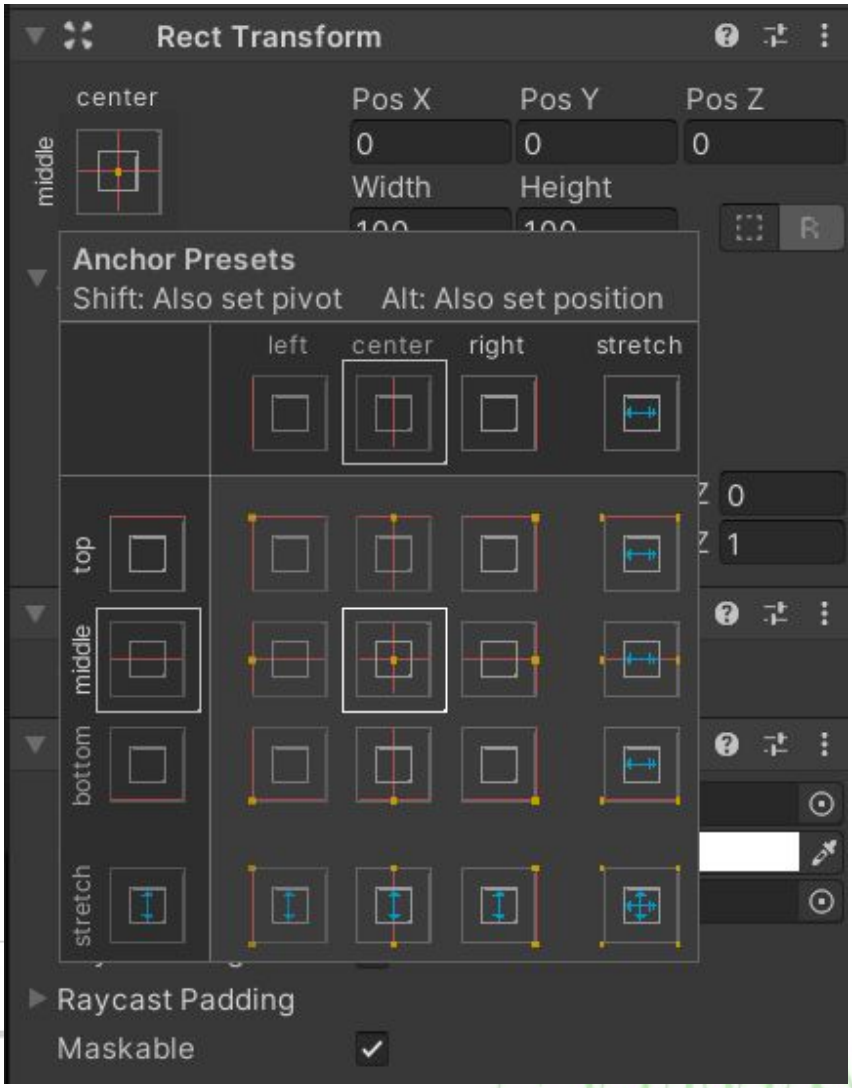


UI基本概念-Anchor Presets

可以從Inspector的畫面調整對齊設定或者直接設定Anchor的值

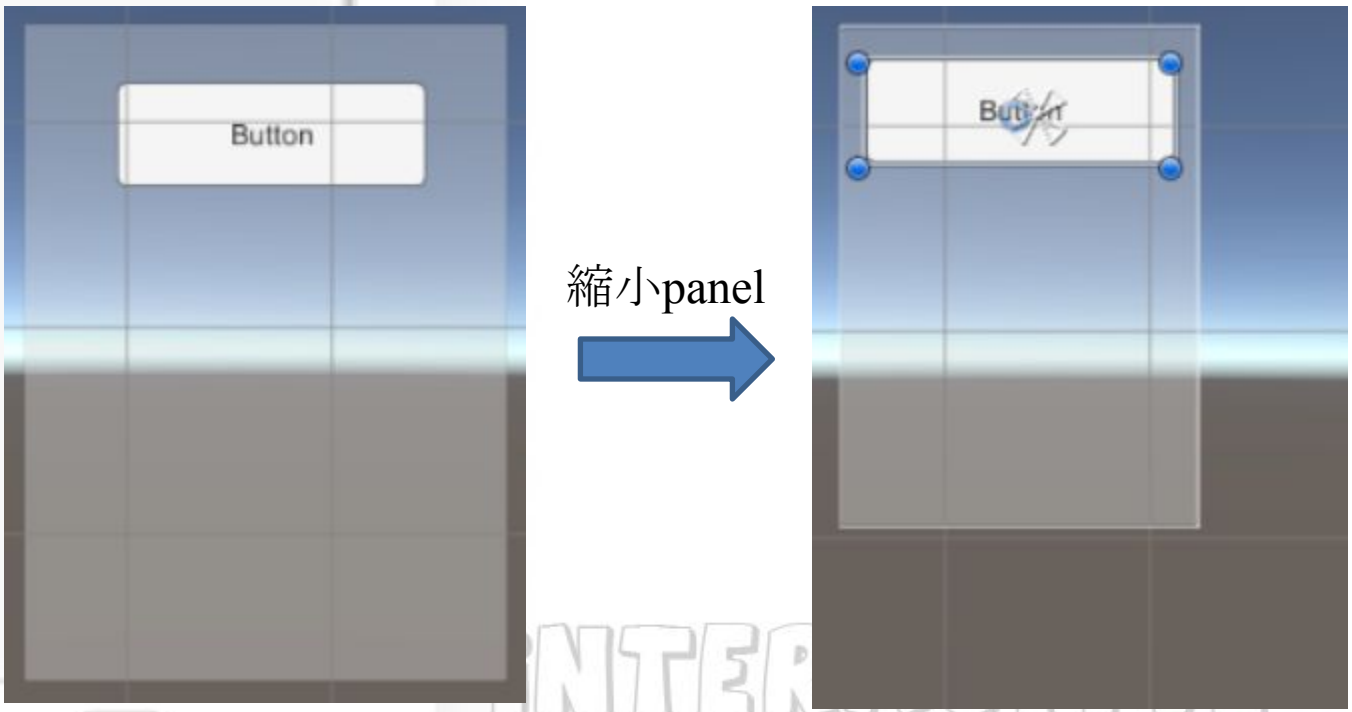
中間的九宮格:當父物件Width/Height改變時子物件移動的對齊位置(不會改變子物件大小)

旁邊的七格:當父物件的Width/Height改變時子物件的長寬伸縮和位置設定



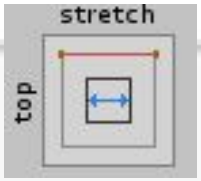
UI基本概念-Anchors

- 類型一：四個Anchor為同一位置(presets九宮格), UI不會隨著panel縮放

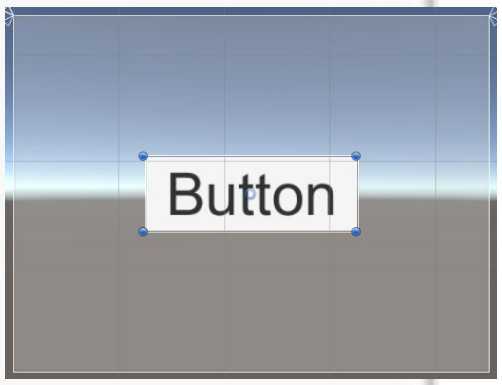


UI基本概念-Anchors

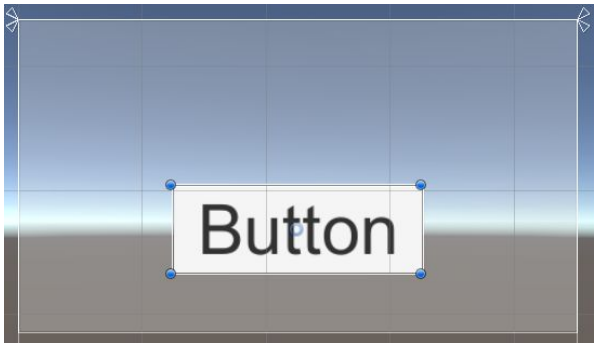
- 類型二：四個Anchor 分成兩組 (presets 右與下的六個選項)

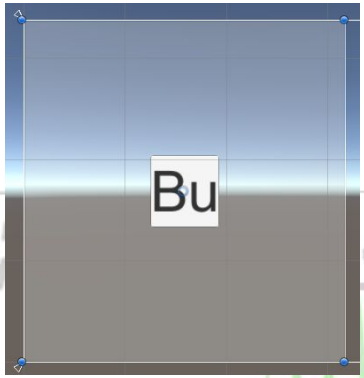


:物件長寬會隨著兩組Anchors間的距離改變



縮小panel



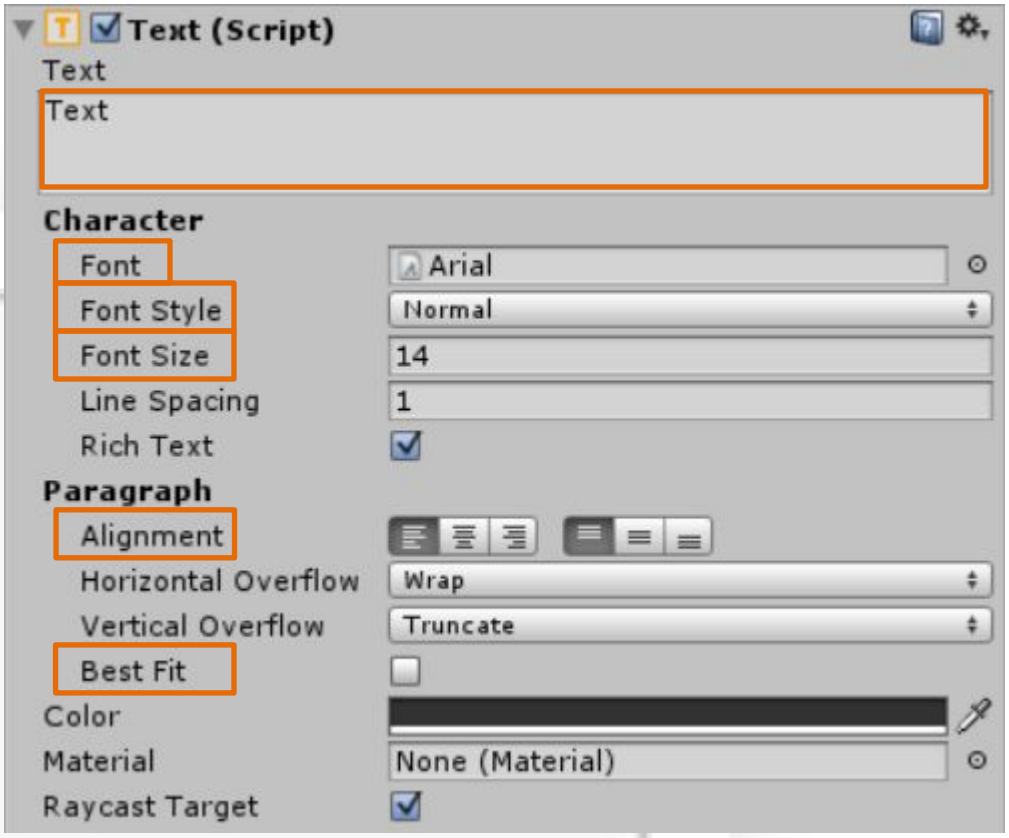


UI基本概念-Anchors

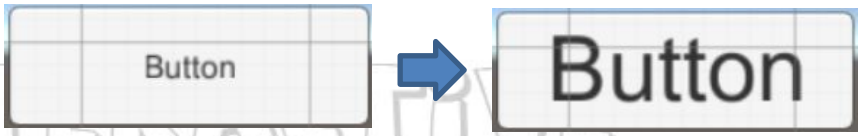
- 類型三 :四個Anchor位置都不同-> Anchors對齊整個Canvas或自訂anchors

不管是從X軸或Y軸改變Canvas大小都會影響到UI物件

UI基本概念-Text



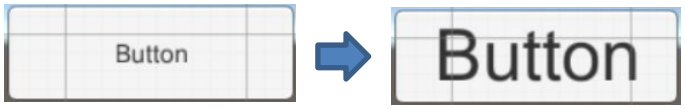
- 文字內容
- 字型
- 粗體/斜體
- 大小
- 對齊方式
- 讓字體填滿整個空間(下圖)



實作時間（3min）

1. 建立 3D專案
2. 設置 Canvas - Panel - Button
3. 嘗試調整 Anchor Presets

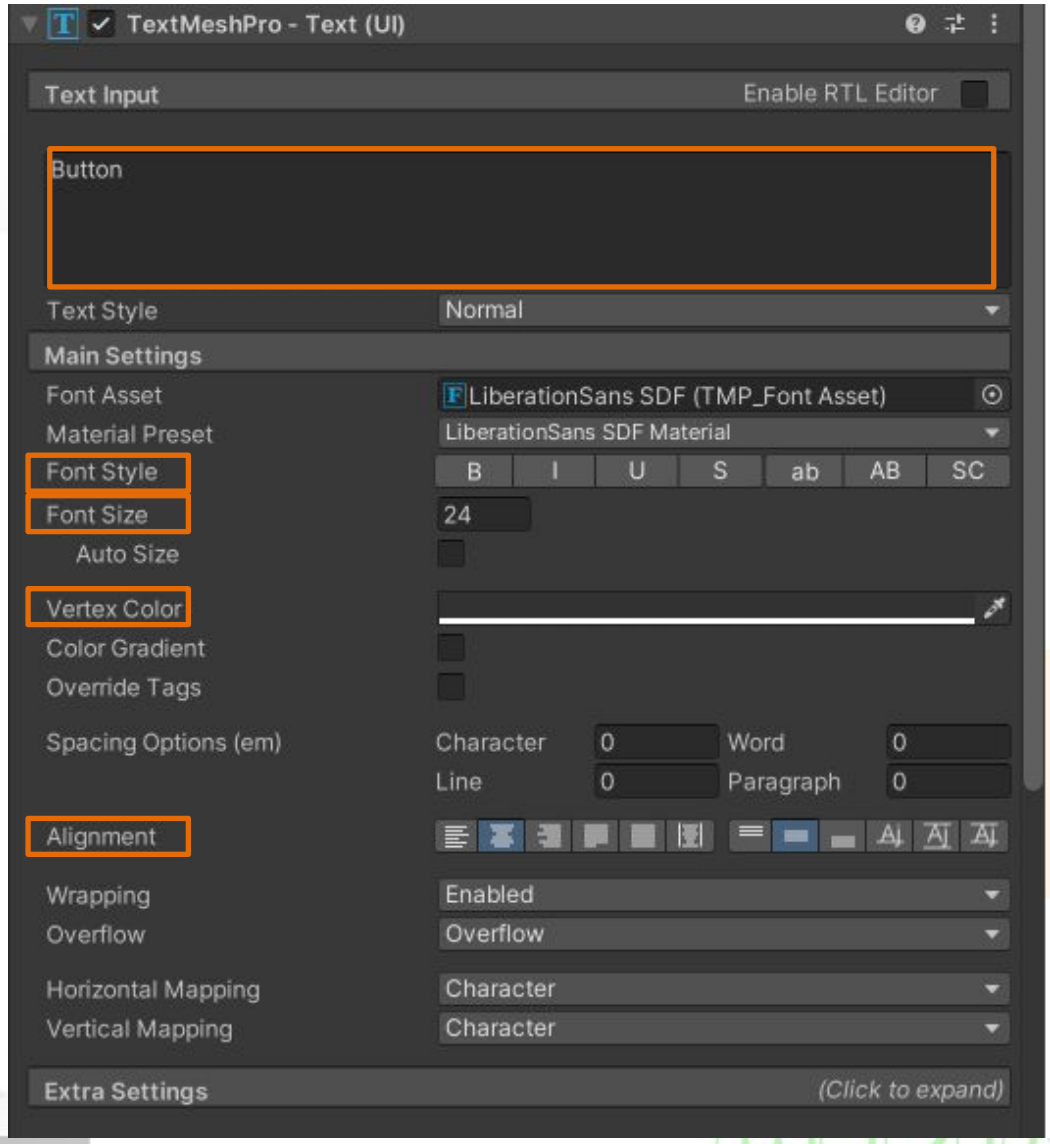
UI基本概念 - Text (TMP)



- 文字內容

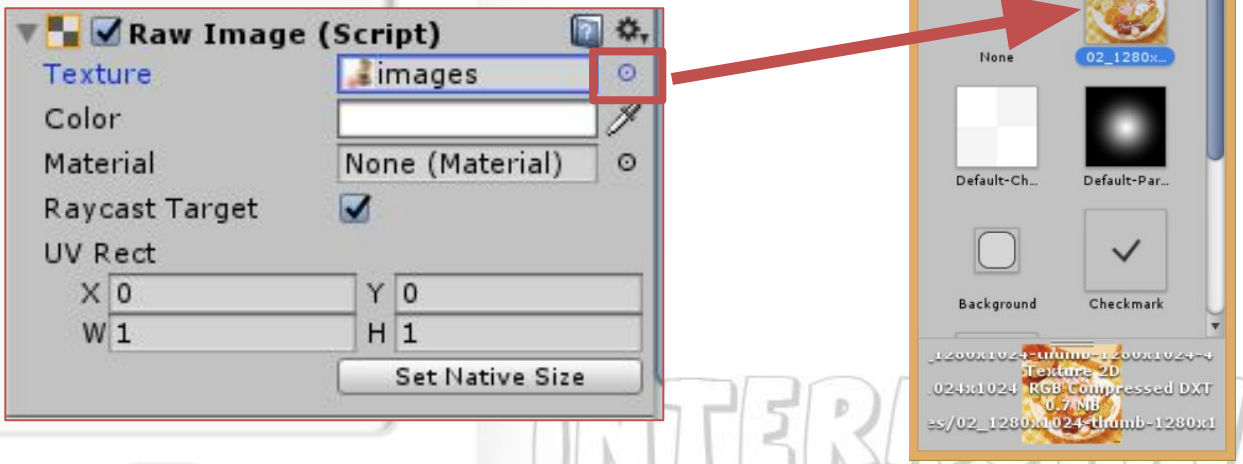
- 粗體 / 斜體
- 大小
- 顏色

- 對齊方式



UI Image & Raw image

- Raw image是當將圖片匯入Asset就可以直接使用，而Image只有將圖片改为Sprite(2D and UI)後才可讀取
- Raw image僅能修改UV屬性、Image則有image type、layout...等
- 選擇圖片:[　] -> [選圖片點兩下]

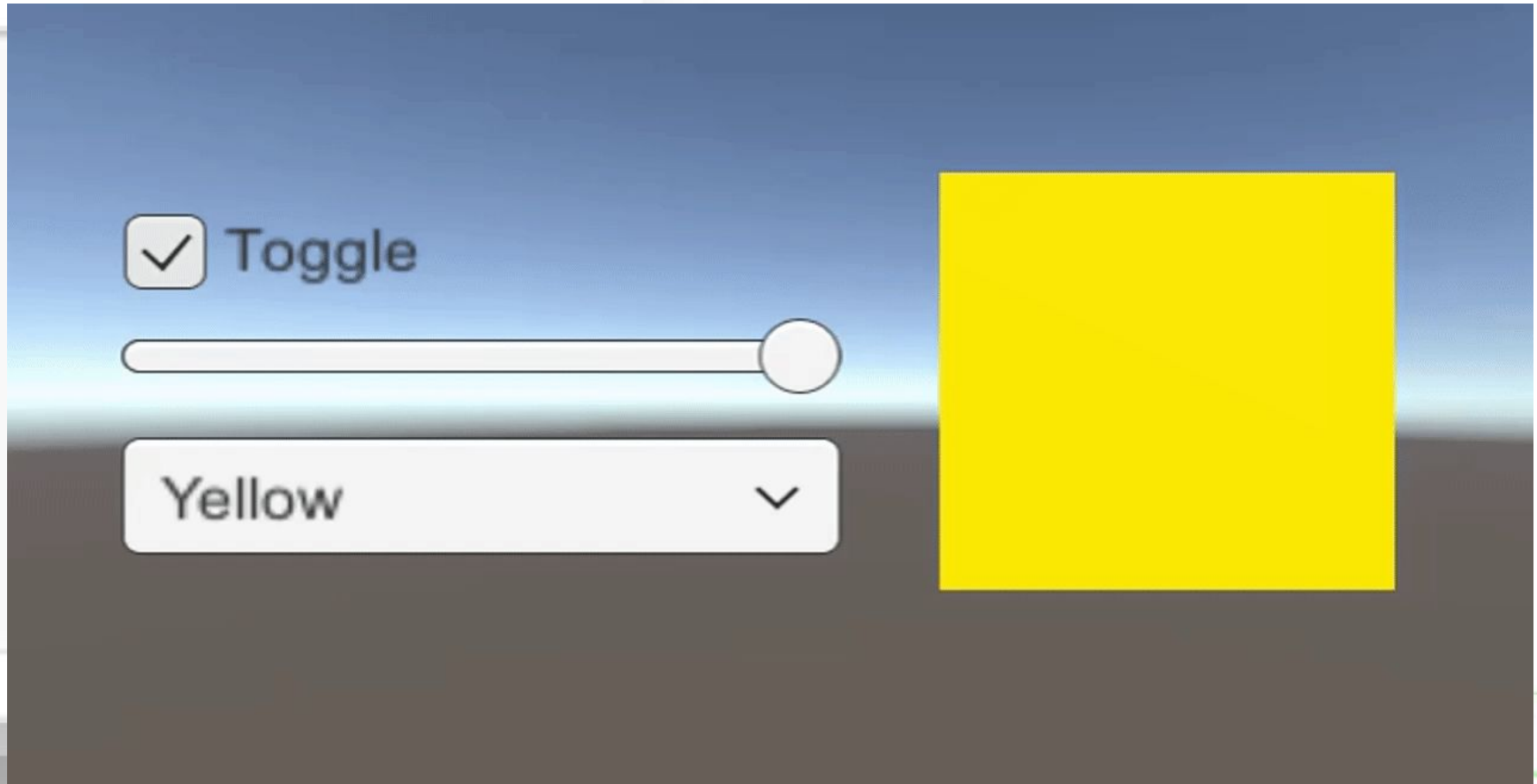


互動式UI

Toggle：一個開或關的選項

Slider：一個可移動的滑條, 左邊值為0, 右邊值為1

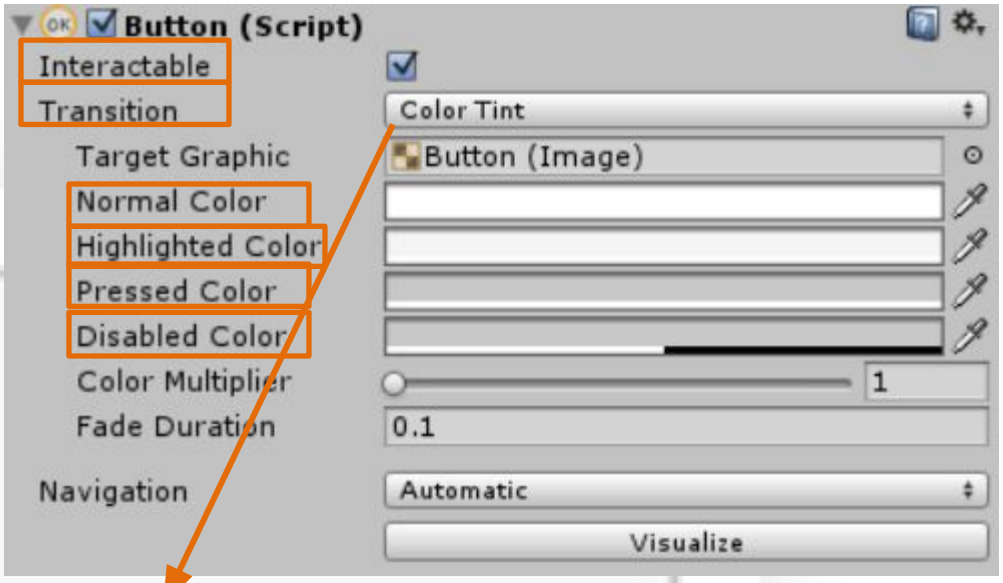
Dropdown：一個可以展開的選單



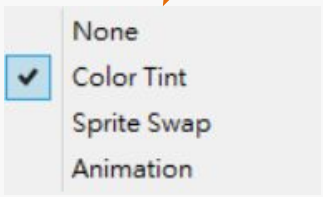
實作時間（5min）

1. 嘗試修改Text
2. 嘗試使用 Image
3. UI - Toggle / Slider / Dropdown

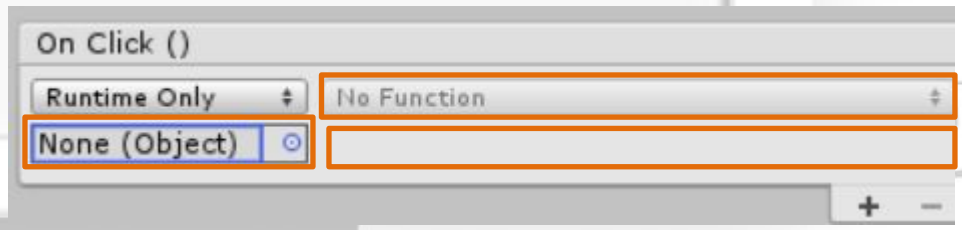
UI基本概念-Button



- 打勾，才能按這個Button
- 平常的顏色
- 滑鼠移動過去的顏色
- 按下Button的顏色
- Interactable不打勾的顏色

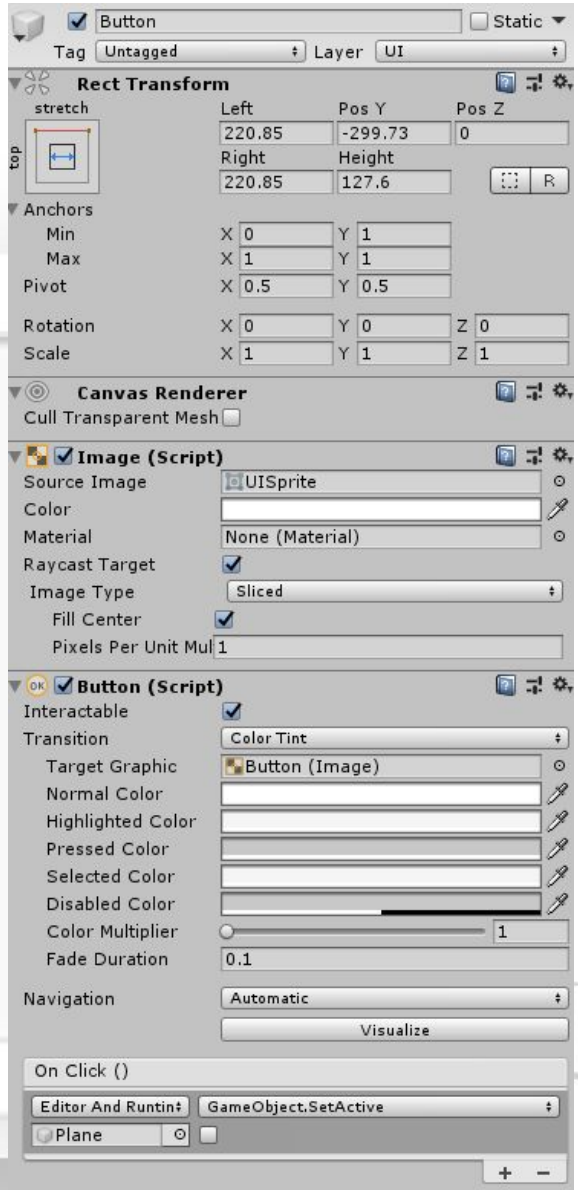


Transition的種類:
顏色
圖片
動畫



選擇要用的函式(**出現的選項全是物體的屬性**)
若有參數需要assign對象物件(**付有語法的物體**)

UI範例 - Editor控制Button



UI範例 - 程式控制Button

- 一定要import UnityEngine.UI

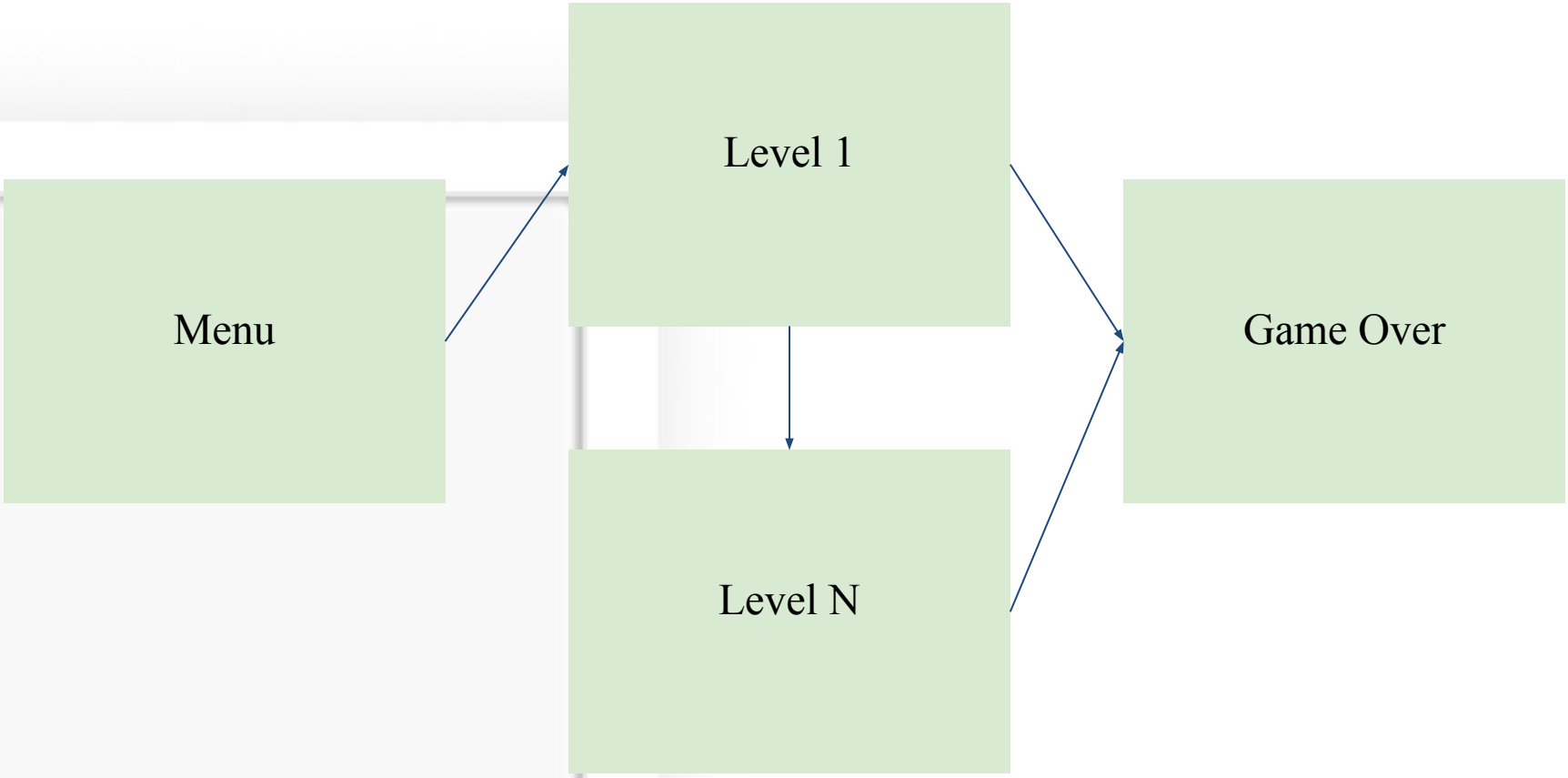
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class ButtonFunction : MonoBehaviour {
    // Use this for initialization
    void Start () {
        this.GetComponent<Button>().onClick.AddListener(DoSomething1);
        //如果這支程式碼附在一個有Button component的物件上
        //他會讀取Button，並且在Button上面加一個method
        //在你點擊該該Button的時候執行你設定的method
    }

    // Update is called once per frame
    void Update () {

    }
    public void DoSomething1() { print("this button is on clicked."); }
}
```

實作時間（3min）

1. Button - Editor
2. Button - 程式控制

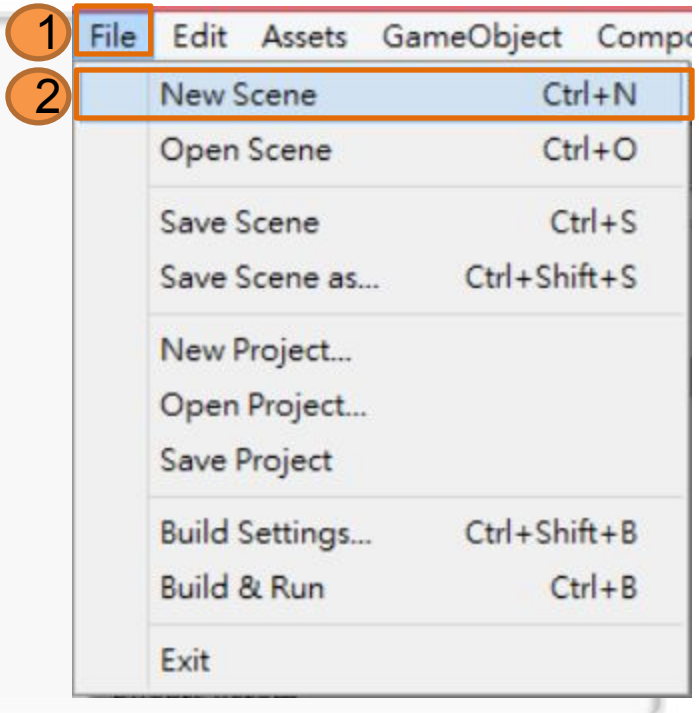
場景切換

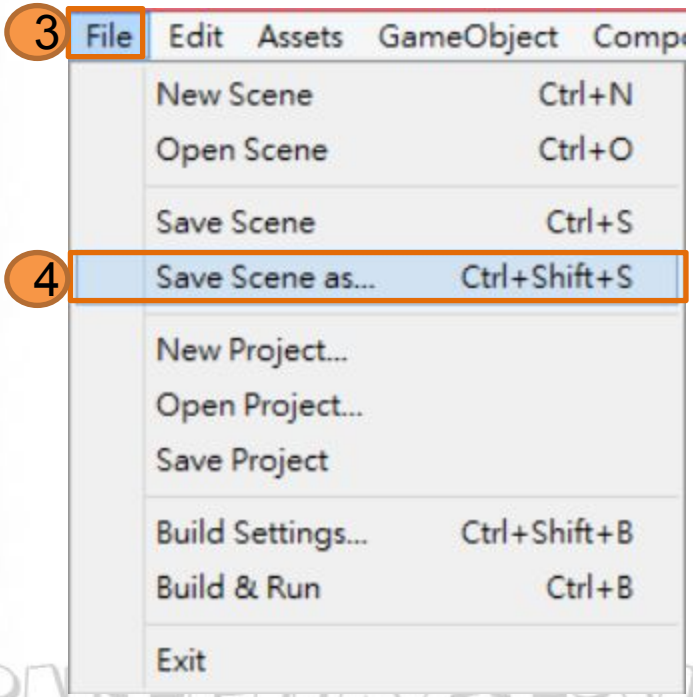
游戲流程圖



場景切換

- 新增一個場景並儲存(剛剛的那個場景記得也要儲存)

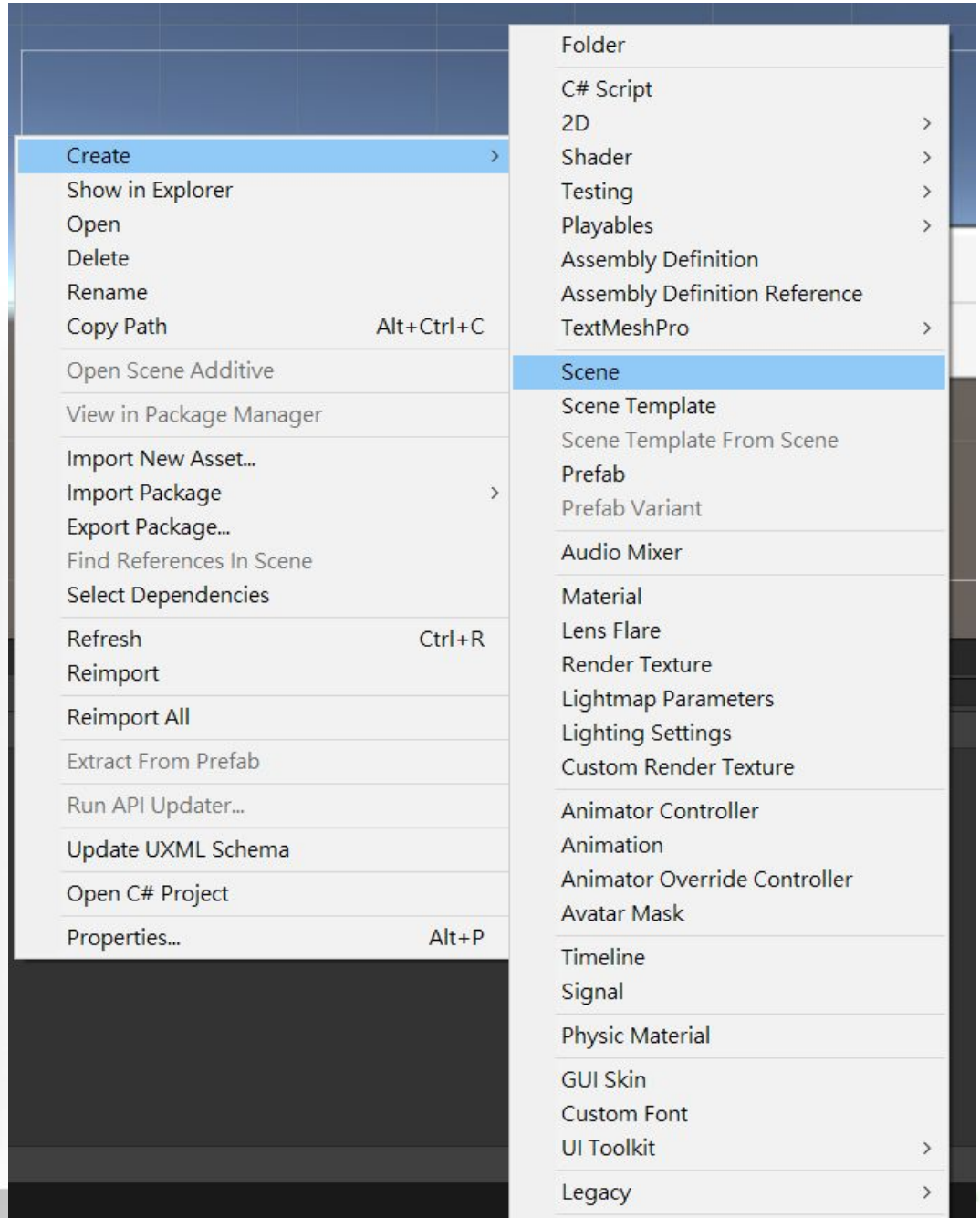




場景切換

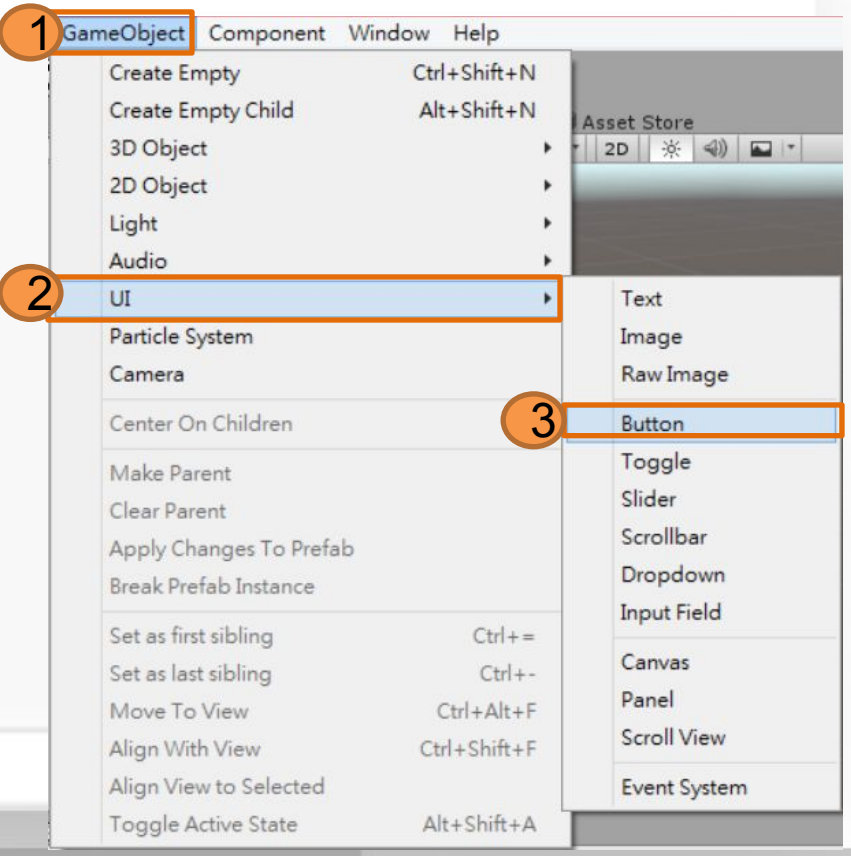
新增兩個場景

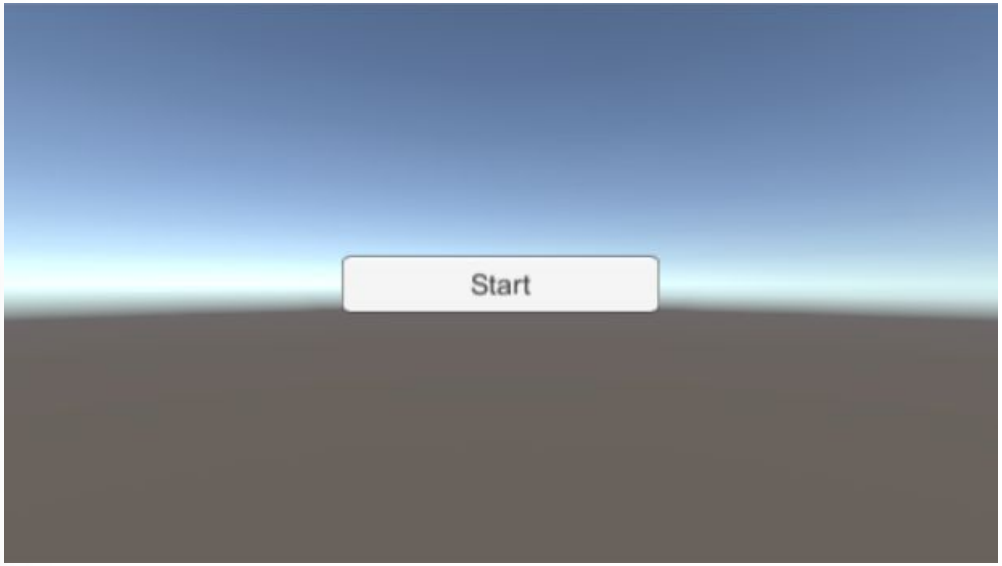
- Assets欄按右鍵
- Create
- Scene



場景切換

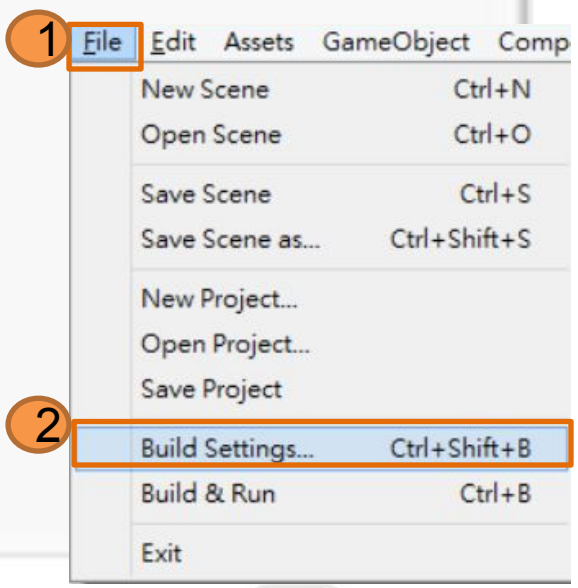
- 在兩個場景中新增一個Button
- 並且移到適當的位置

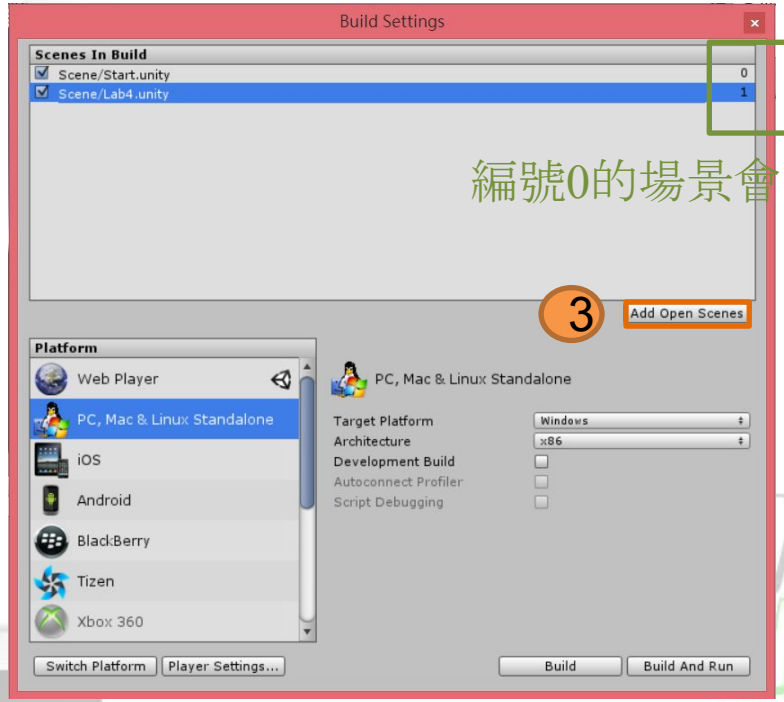




場景切換

- 把兩個場景一一加進來
- [File]->[Build Settings]　->把游戲會用到的場景拉到Scenes In Build
- 有幾個場景就要做幾次這個步驟





編號0的場景會是開始的場景！

場景切換

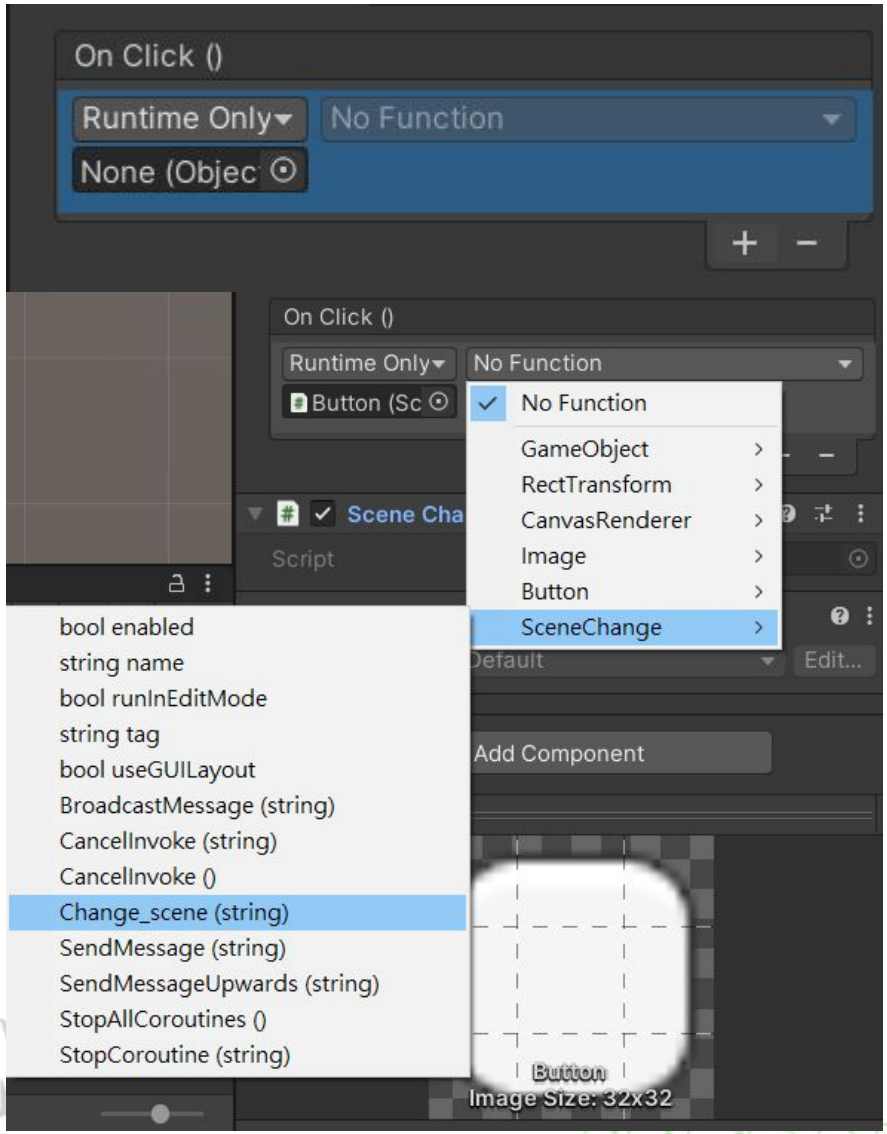
- 將這個script附在兩個要切換場景的Button上

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
Unity 指令碼|0 個參考
public class SceneChange : MonoBehaviour
{
    // Start is called before the first frame update
    Unity Message|0 個參考
    void Start()
    {

    }

    0 個參考
    public void Change_scene(string Scenename)
    {
        SceneManager.LoadScene(Scenename);
    }

    // Update is called once per frame
    Unity Message|0 個參考
    void Update()
    {

    }
}
```

場景切換

回到Button的Inspector，有個On Click，按下加號，此時可以選擇當按鈕按下時要被呼叫的Function。

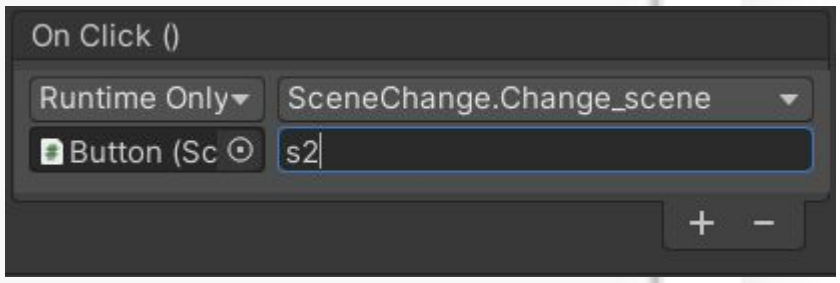
把Button拖到Object的選項，然後選擇剛剛創的Script名字，選擇Change_scene



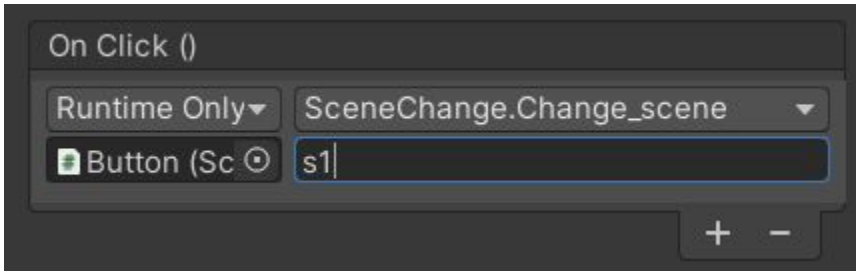
場景切換

可以看到Change_scene需要一個string當作參數，也就是要載入的場景名稱。

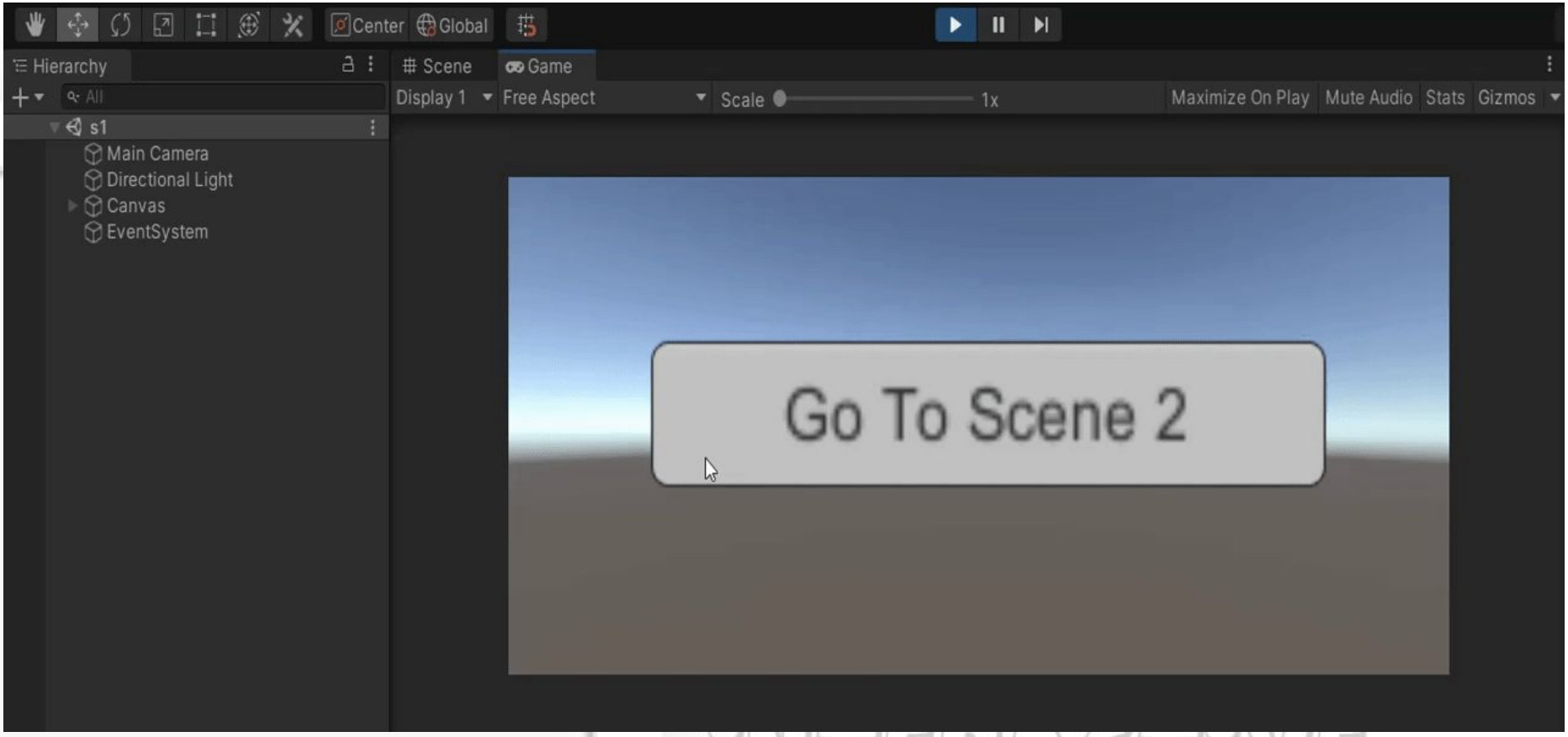
s1場景:按下載入s2場景



s2場景:按下載入s1場景



切換測試



場景切換

- 將這個script附在Button上
- void **LoadScene**(string **sceneName)**

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class ButtonFunction : MonoBehaviour
{
    // Use this for initialization
    void Start()
    {
        this.GetComponent<Button>().onClick.AddListener(SwitchScene);
        //如果這支程式碼附在一個有Button component的物件上
        //他會讀取Button，並且在Button上面加一個method
        //在你點擊該該Button的時候執行你設定的method
    }

    // Update is called once per frame
    void Update()
    {

    }
    public void DoSomething1()
    {
        print("this button is on clicked.");
    }
    public void SwitchScene()
    {
        SceneManager.LoadScene("OtherScene");
    }
}
```

場景切換

- 兩個方法
- 投影片　13頁 editor
- 　　14頁 程式內

場景切換

建立好索引值後

void **LoadScene**(int **sceneBuildIndex**, SceneManagement.LoadSceneMode **mode** = LoadSceneMode.Single);

e.g.載入開始的場景:

SceneManager.LoadScene(0);

場景管理 進階

- Restart current scene
  - SceneManager.LoadScene(SceneManager.GetActiveScene().name) ;

- AsyncOperation **LoadSceneAsync**(string **sceneName**)
  - Loads the Scene asynchronously in the background.

- SceneManager.**UnloadSceneAsync**()
  - Destroys all GameObjects associated with the given Scene and removes the Scene from the SceneManager.

- LoadSceneMode
  - **Single** mode loads a standard Unity Scene which then appears on its own in the Hierarchy window.
  - **Additive** loads a Scene which appears in the Hierarchy window while another is active.

實作時間（5min）

1. 建立新的Scenes
2. 新增按鈕
3. 新增script
4. 切換場景

PlayerPrefs(儲存變數)

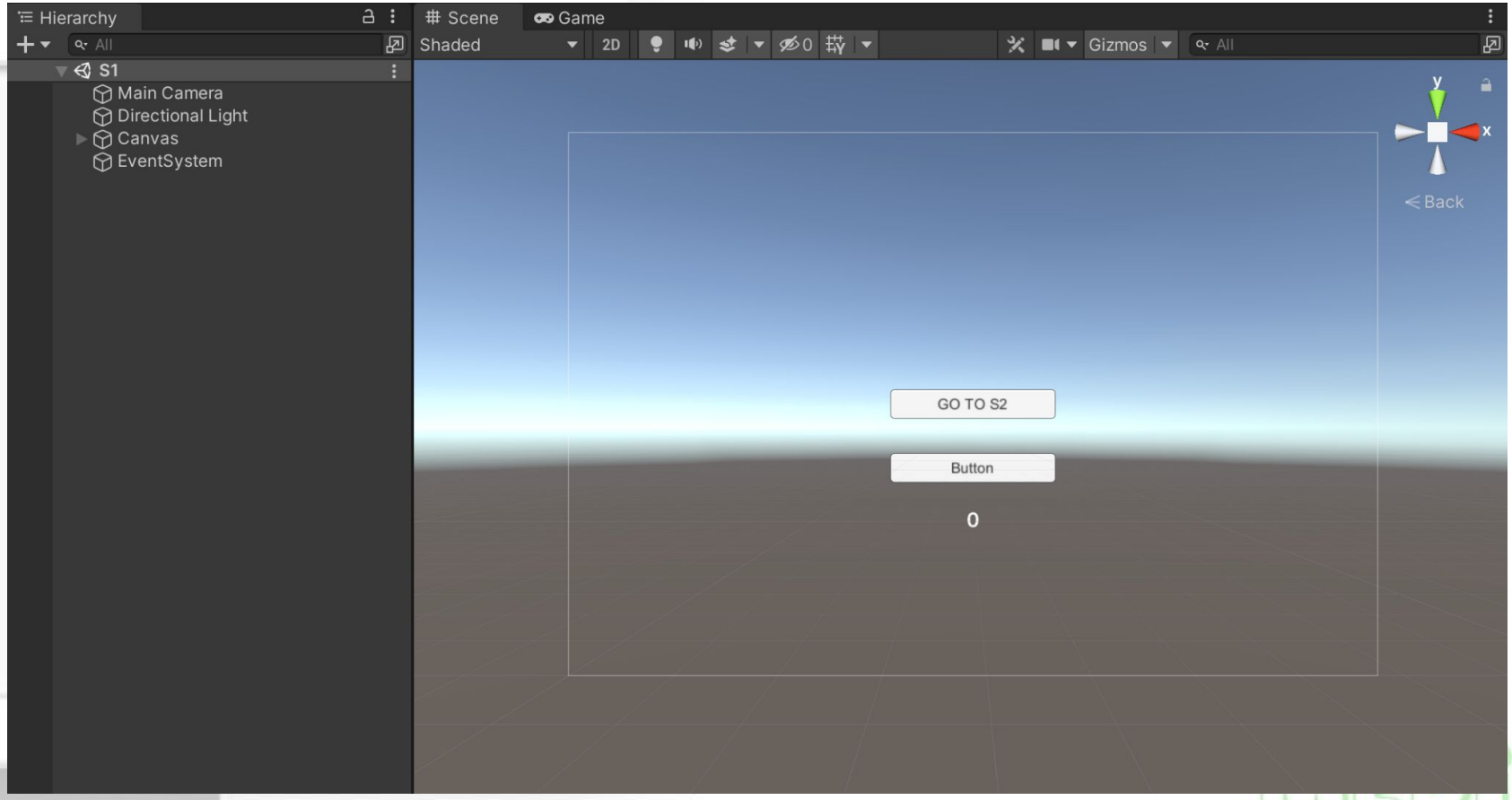
PlayerPrefs可以在Unity本地端儲存字串、整數、浮點數等資料

```
PlayerPrefs.SetInt("YourKey", "Your_Value");
PlayerPrefs.GetInt("YourKey");
```

- SetInt(); - 儲存整數
- GetInt(); - 讀取整數
- SetFloat(); - 儲存浮點數
- GetFloat(); - 讀取浮點數
- SetString(); - 儲存字串
- GetString(); - 讀取字串

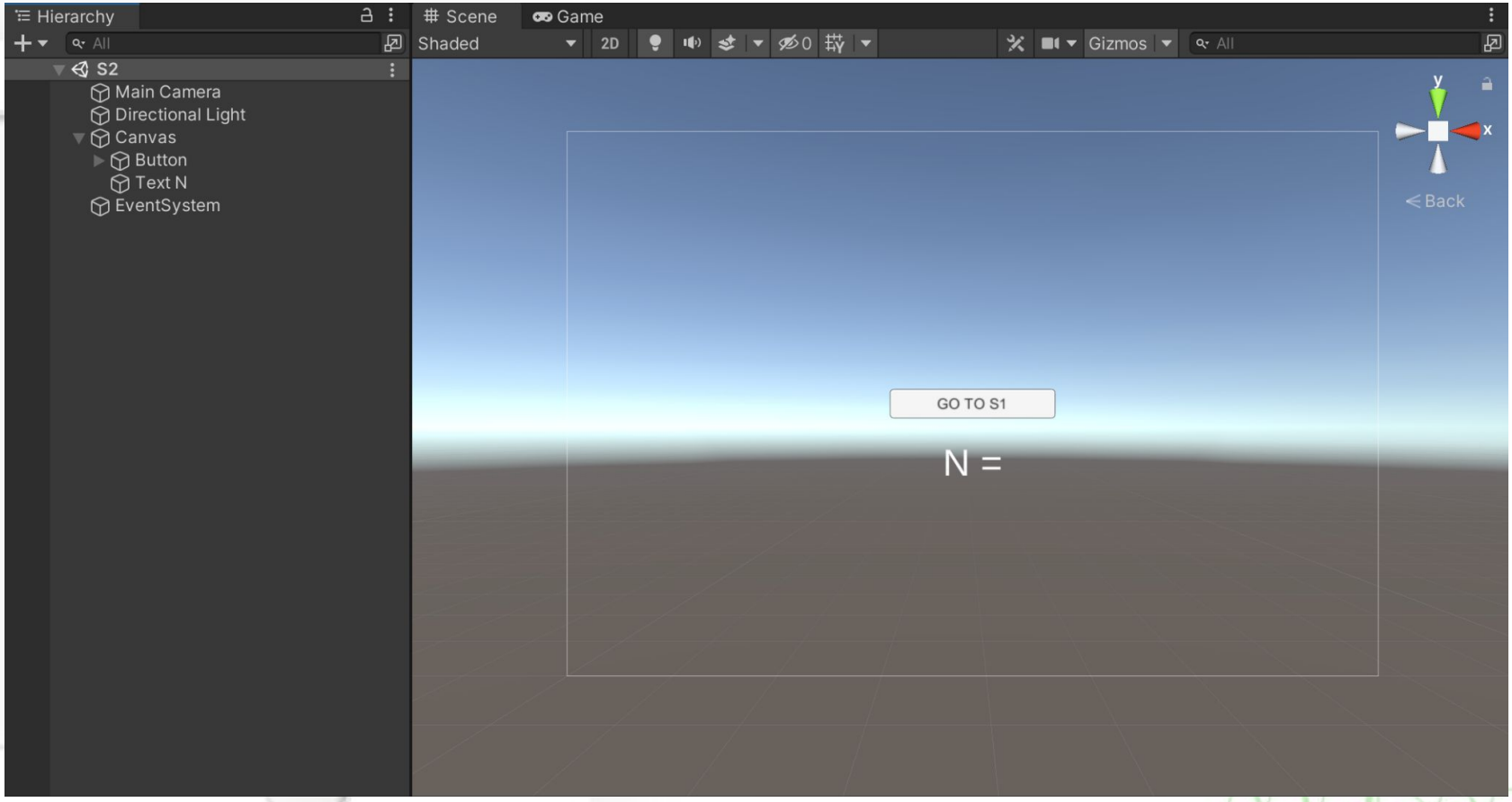
PlayerPrefs(儲存變數)

- 設定一個按鈕，每按一次下面的數字+1
- 將數字利用PlayerPrefs儲存



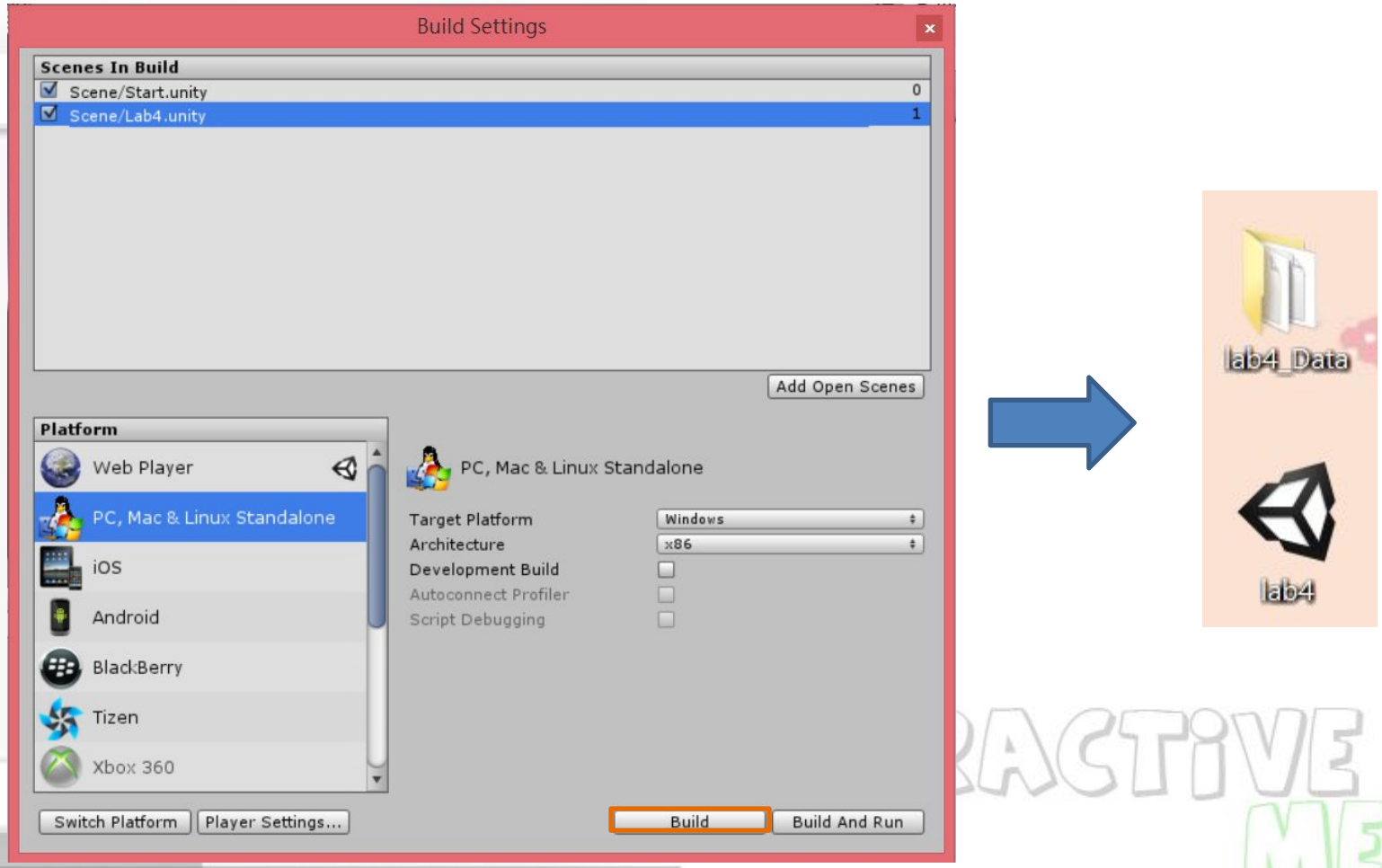
PlayerPrefs(儲存變數)

- 在Scene2中利用PlayerPrefs讀取並顯示數值



輸出成執行檔

- [平台及輸出格式]->[Build]->[輸出路徑]



實作時間（5min）

1. 新增S1的按鈕與文字UI
2. 寫script改變數值並除存數值
3. 新增S2文字UI
4. 寫script讀取數值並顯示