Unity 3D 練習手冊

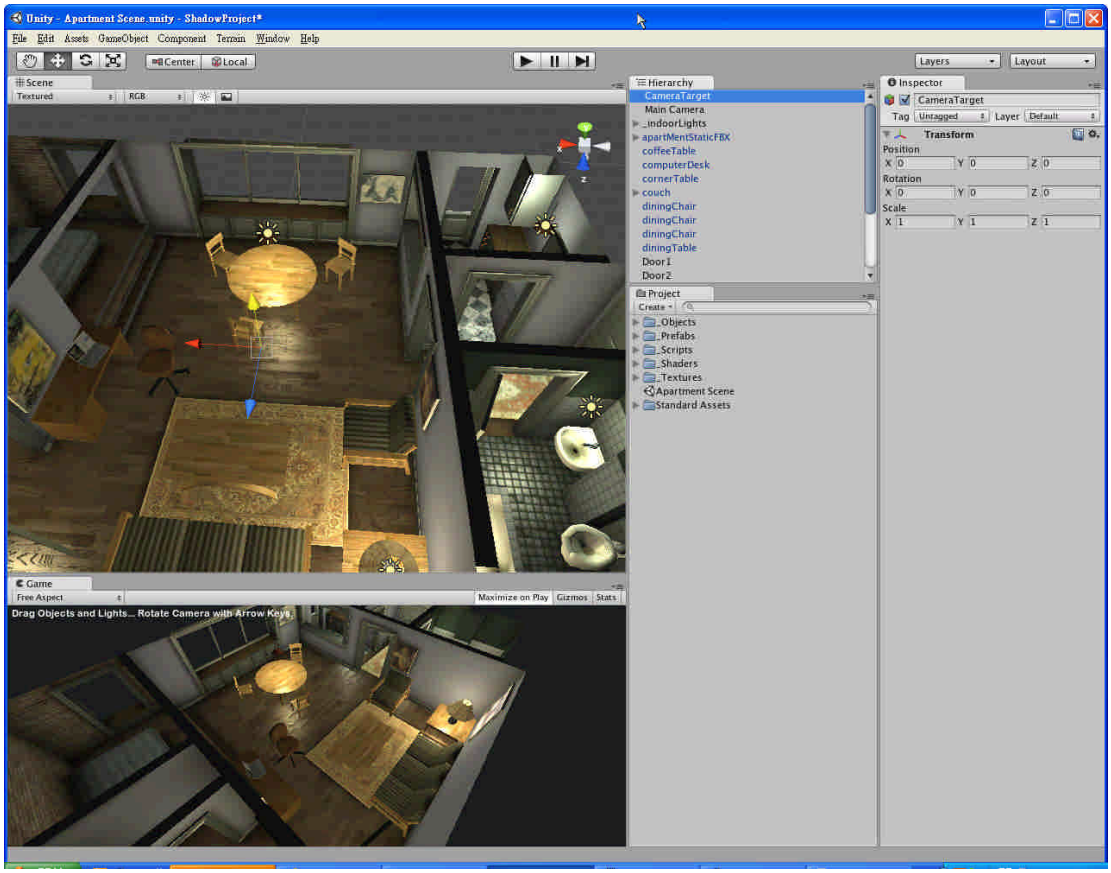
作者：游峰碩

崑山科技大學資訊管理系

1. 3D Game 製作-瞭解 Unity 3D 環境

1.1. 認識 Unity 3D 環境

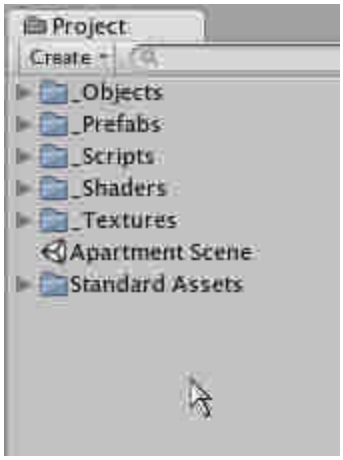
到 Unity 3D 網站下載 Shadow 這個計畫。

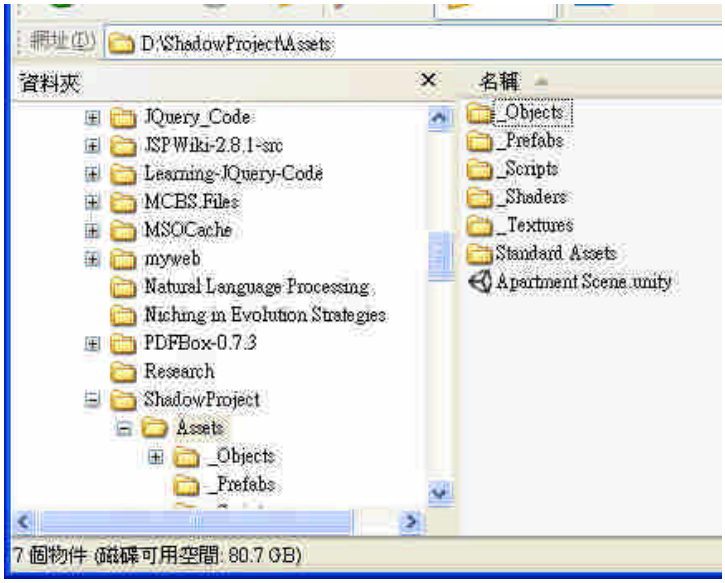


打開此計畫。(雙擊 D:\ShadowProject\Assets\Apartment Scene.unity 這個檔案即可。)

Project 面版

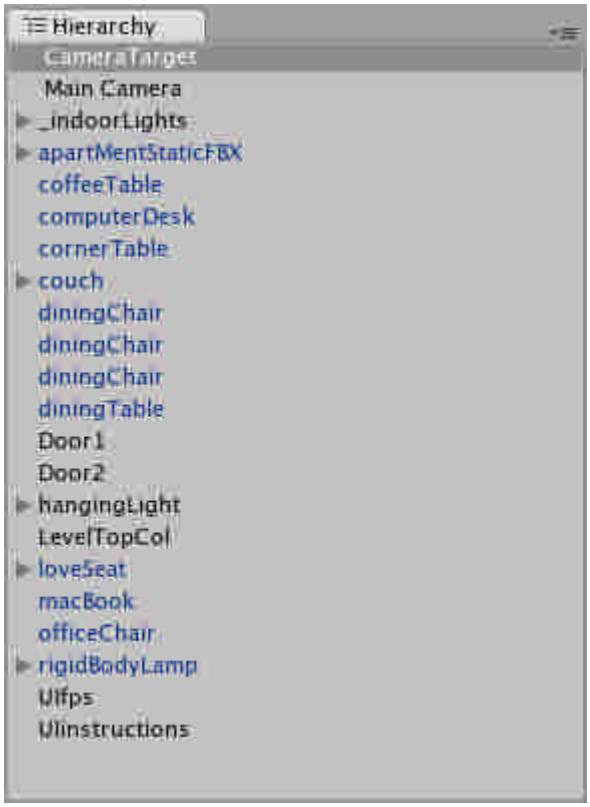
基本上，Project 面版所顯示的就是外部檔案目錄的結構。





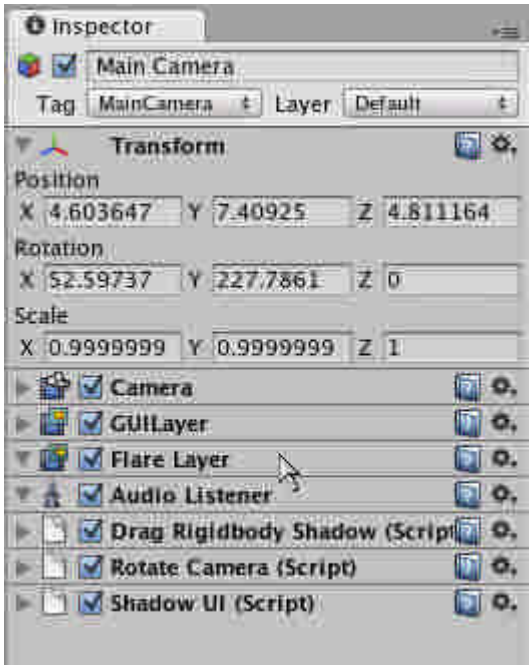
Hierarchy 面版

Hierarchy 面版包含遊戲中所有的 **Game Object** (GO) 。點選任一個 GO，右邊的 Inspector 面版會顯示該 GO 的屬性。這些屬性在 Unity 3D 叫做 **Component**。



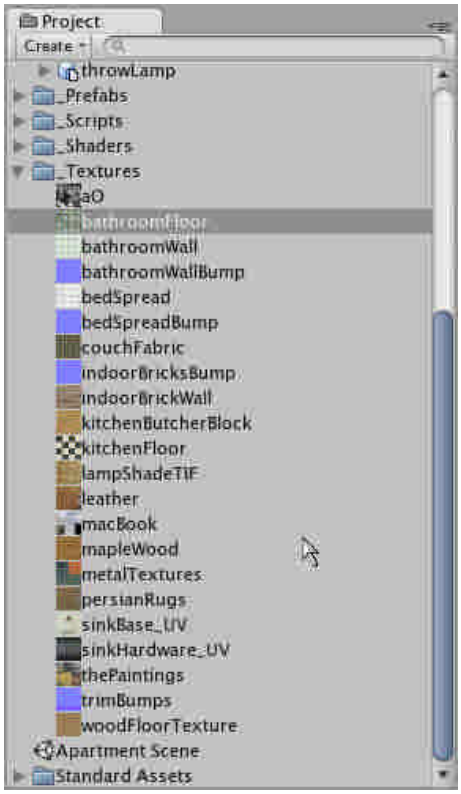
Inspector 面版

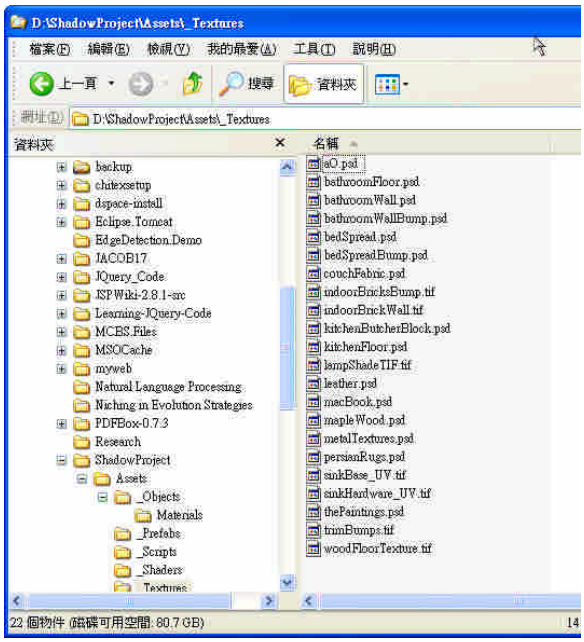
Inspector 可以想像成是 Game Object 的 component(屬性)面版。下圖顯示的是 Main Camera 的 Component。



在 Project 面版展開 _Textures

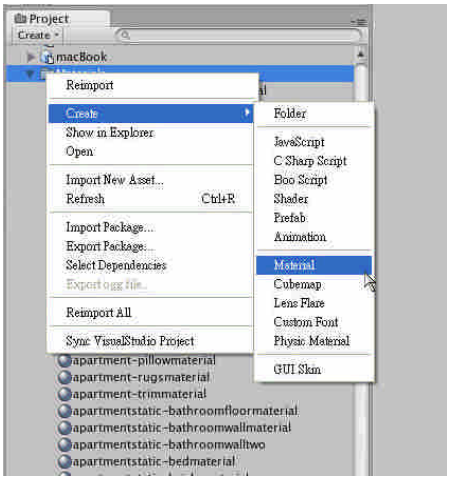
這裡面包含了很多的檔案，從檔案總管中可以看到，這些都是一般的圖片檔案。Texture 裡面放的都是可以用來貼皮用的圖片檔。(貼到 Mesh 上)



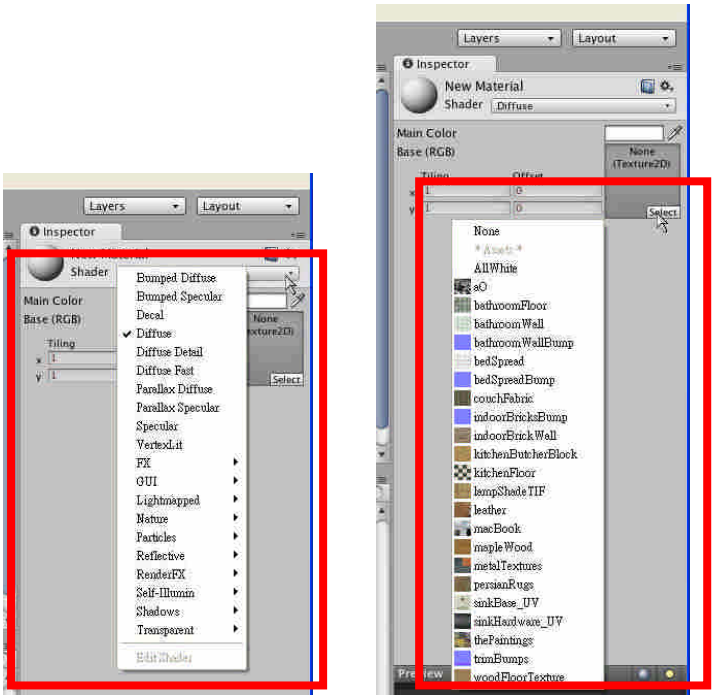


Material

在 material 上，我們可以使用 texture。建構 material 可以在 Project 面版的任意地方，點擊右鍵，接著選擇 create -> Material。



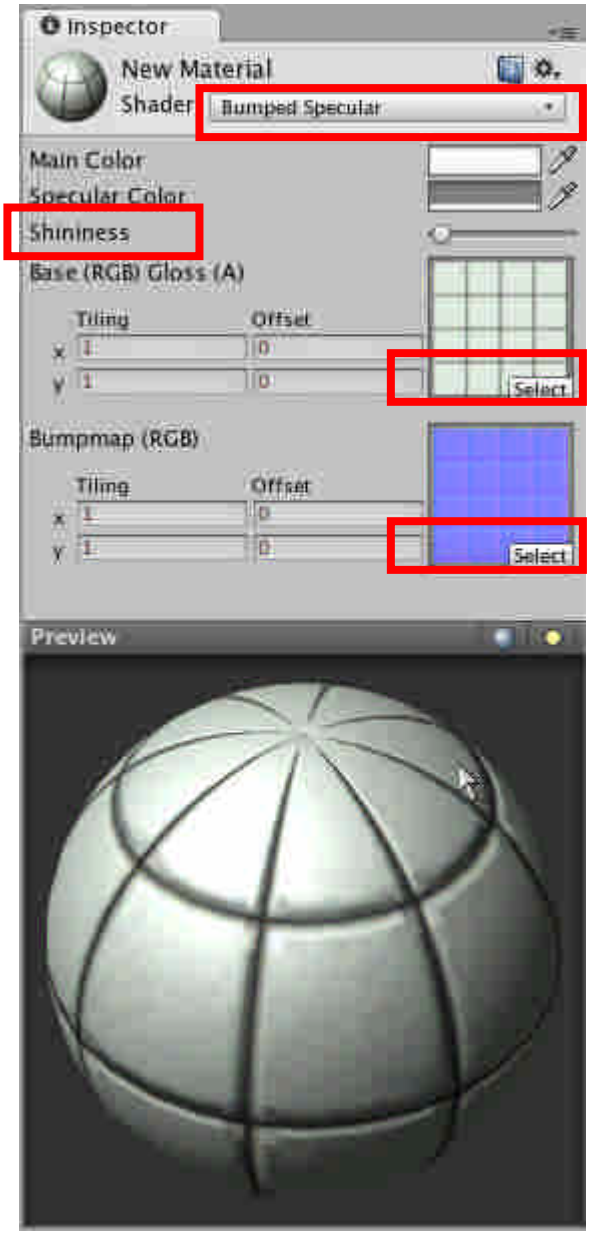
在新的 material 上，我們可以設定不同的 Shader。以及設定不同的 Texture。



範例中，我們選擇了「Bumped Specular」做爲這個 material 的 Shader，選定後，下方的屬性中，我們可以調整亮度 (Shininess) 以及選擇要貼上的皮。

在 Preview 視窗中的右上角有兩個小的按鈕。第一個按鈕可以用來觀看不同幾何形狀在貼上皮之後的樣子；另一個按鈕可以用來開關光源。

註：根據所選擇的 Shader 不同，相關可設定之屬性也會不同。



Mesh

Unity 3D 不是用來建構 Mesh 的。一個物體的 Mesh 一般來說都是使用其他軟體例如 Maya，3D Max，Blender 來設計。Unity 3D 可以讀入 .fbx, .dae (Collada), .3DS, .dxf and .obj 副檔名的檔案。

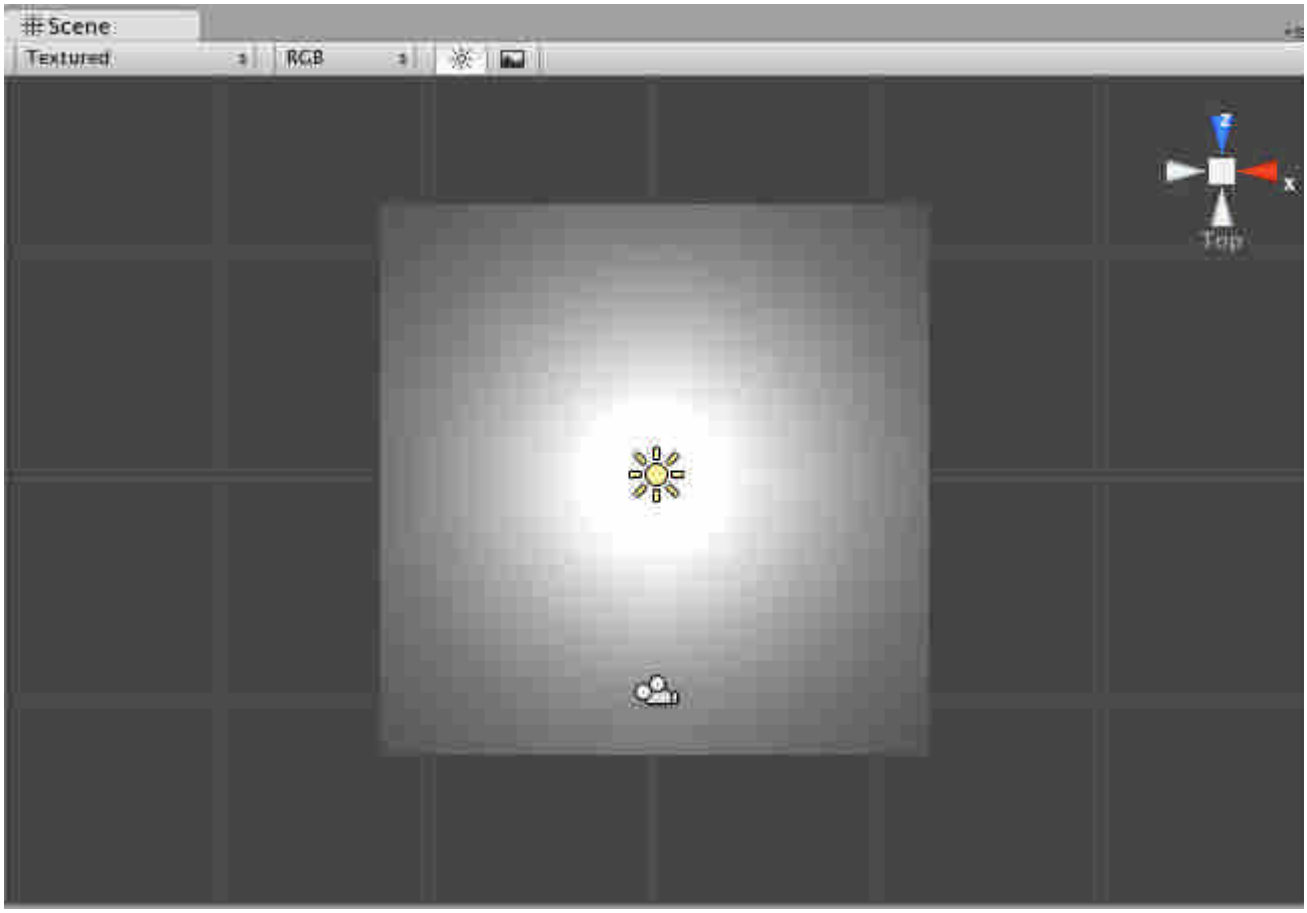
2. 3D Game 製作-FollowUp 製作

2.1. 配置場景

先把基本場景布置好。

GO 的初始位置都要重設爲 0

1. Cube：X=25,y=0.3,z=25
2. Point Light



放一個 Cube 命名爲 Actor。離地面一點距離。

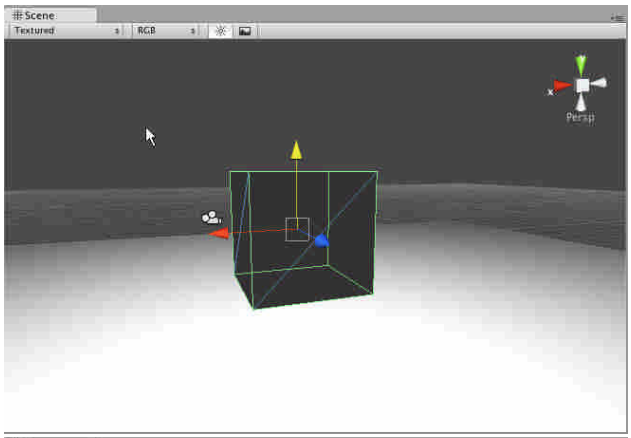
然後 attach 一個 MoveAround 的 script 給這個 Actor。

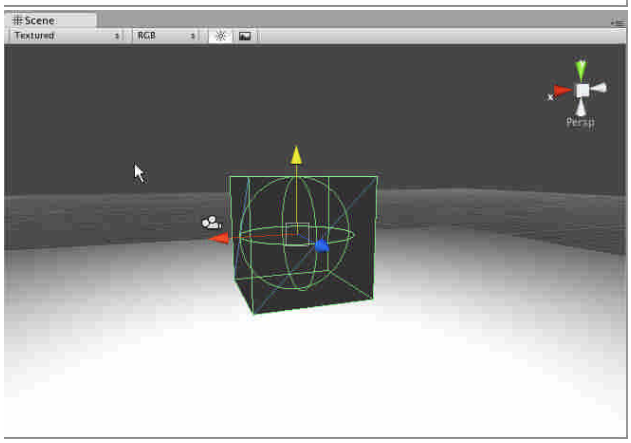
Script：MoveAround

```
var speed = 10.0;
var rotateSpeed = 10.0;
function Update ()
{
var controller : CharacterController =
  GetComponent(CharacterController);
```

```
transform.Rotate(0, Input.GetAxis ("Horizontal") * rotateSpeed, 0);

var forward = transform.TransformDirection(Vector3.forward);
var curSpeed = speed * Input.GetAxis ("Vertical");
controller.SimpleMove(forward * curSpeed); // 向 forward 這個方向移
  動
}
```

程式碼中
GetComponent(CharacterController);
的意思是：取出 attached 到這個 GO 的 CharacterController (這是一個 Type)。因此，在 Editor 中，我們先去連結一個 CharacterController 給 Actor。先點選好 Actor，然後 Component -> Physics -> Character Controller



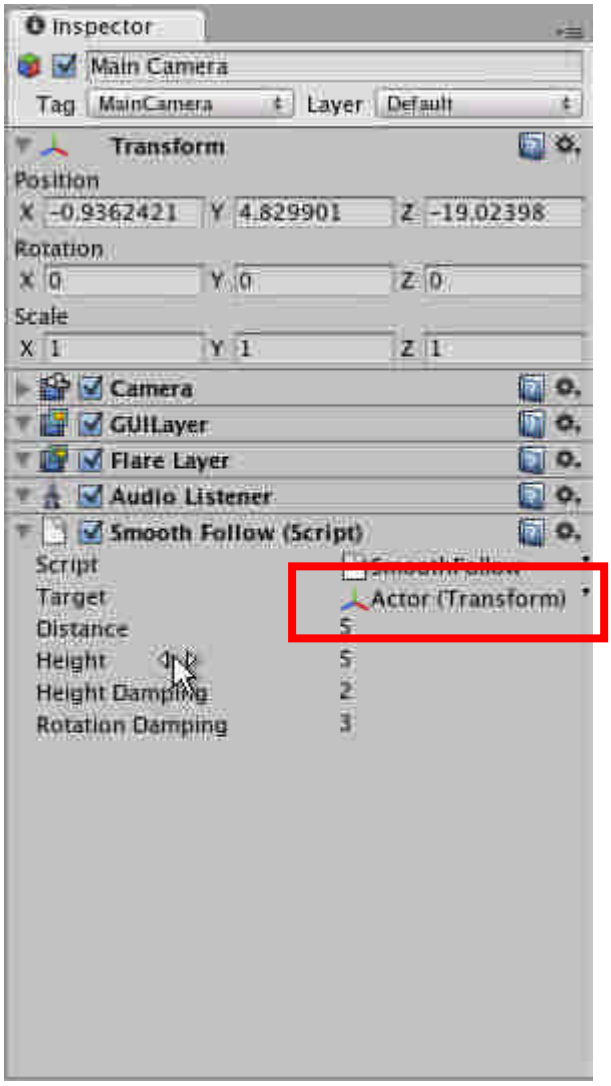


有加了 Character Controller 時，GO 的裡面會多出了一個圓形的線。那個圓形或橢圓型的線代表的就是角色控制器。測試一下程式應該可以前後左右移動我們的 Actor。

如果你有先點選 Actor，那麼你會發現到 Actor 前進的方向都是指向 Y 軸(藍

色)，旋轉時，都是相對於 Z-軸(黃色)旋轉。

稍微操作一下，你會發現到當我們移動角色時，背景都是不會動的。這是因爲我的 Camera 一直都保持不動的位置。因此，當你超出了它的視線，我們會看不到我們操控的物件。一般的 FPS 遊戲，場景會隨著角色的視線移動而改變。要做到這點，我們可以讓 Camera follow 我們的角色。要做到這種情況很簡單，我們只需要把 SmoothFollow 這個 script 連結到 Camera 上，並且指定好要跟蹤的對象(target)即可。SmoothFollow 在安裝時就有了，可以使用 Search，打入 smooth 就可以看到。找到後，把它拖曳給 Main Camera，然後設定 Target 就完成了。



3. 3D Game 製作-簡單射擊

3.1. 配置場景

建立一個 script，然後 attach(連結) 到 Camera。

需要使用到：

a. pointed light

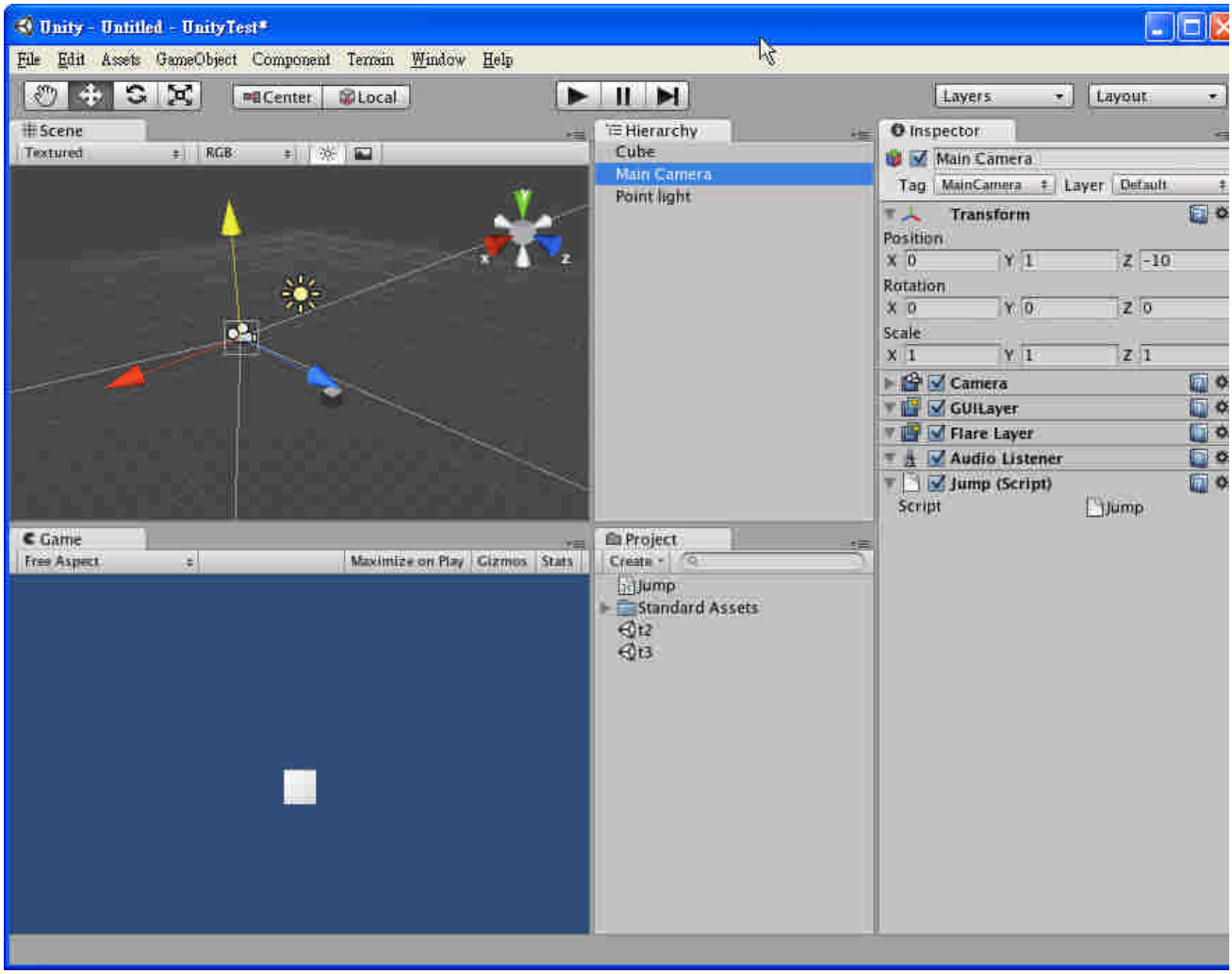
b. cube

c. 下面的 Script: Jump

```
function Update () {
    if(Input.GetButtonDown("Jump"))
    {
        transform.position.z += 1.0; //沿著 Camera 的前方(z-軸)移動
    }
}
```

學習：

- 如何 attach script 到 Game Object



(Jump 對應到空白鍵)

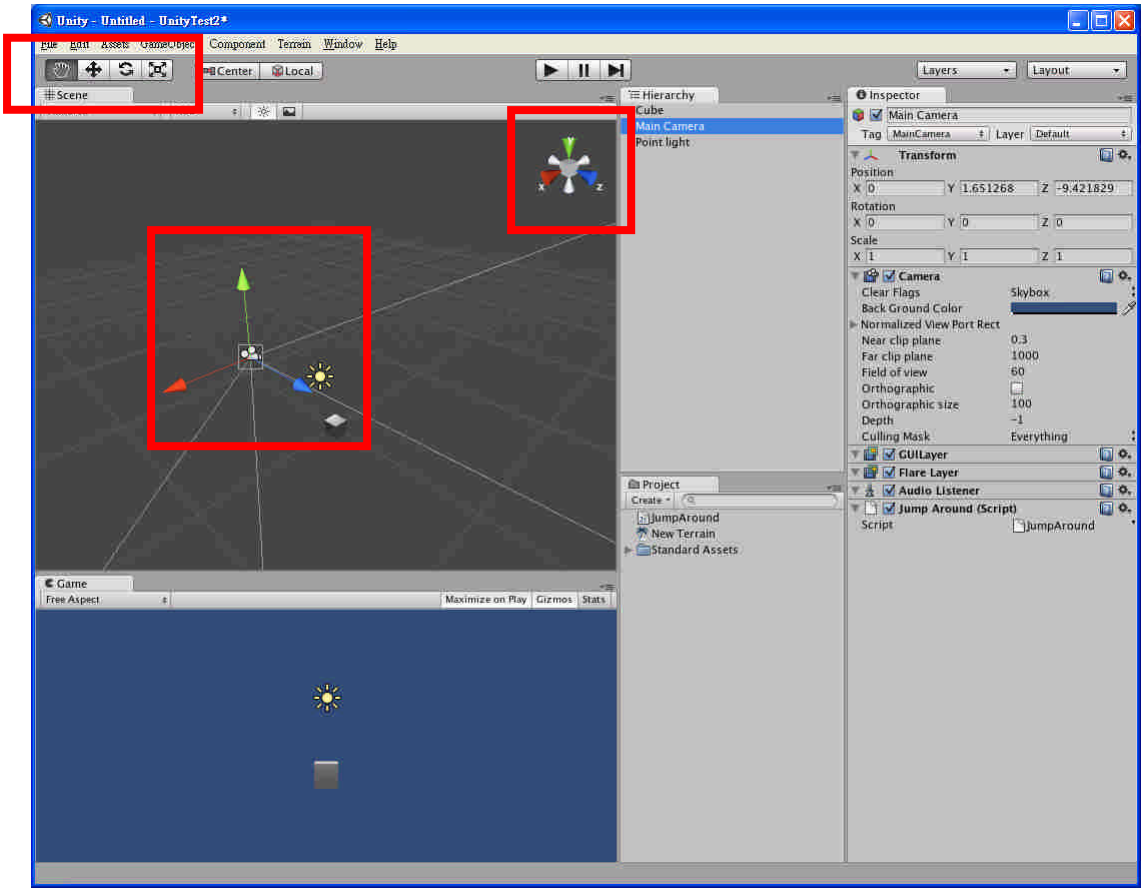
3.2. 練習：配置鍵盤操作

Edit -> Project Settings -> Input
新的 script：MoveAround

```
var speed = 2.0;
var turnSpeed = 50.0;
function Update () {
	if(Input.GetButton("Forward"))
	{
		transform.position += transform.forward * speed * Time.deltaTime;
	}
	if(Input.GetButton("Backward"))
	{
		transform.position += -transform.forward * speed * Time.deltaTime;
	}
	if(Input.GetButton("Left"))
	{
		transform.eulerAngles.y  += -turnSpeed* Time.deltaTime;
	}
		if(Input.GetButton("Right"))
	{
		transform.eulerAngles.y  += turnSpeed* Time.deltaTime;
	}
}
```

把剛才的 Jump 從 Camera 中刪除，然後 attach 這個新的操控 script。執行一下程式。

我們可以把 Camera 當作是 Actor 的眼睛，因此，整個遊戲就會感覺起來好像是 Actor 在場景中走來走去。



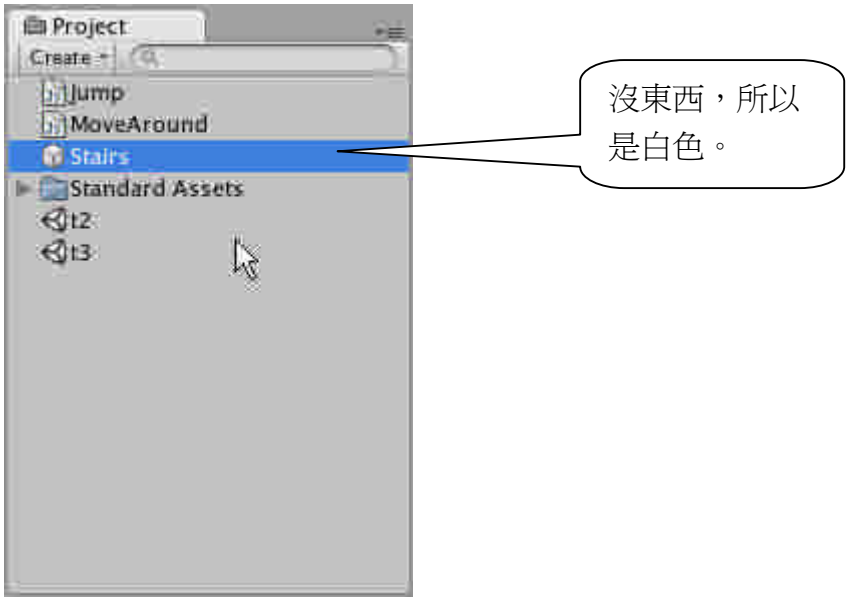
學習如何複製 Game Object
點選要複製的 GO，然後，使用 Ctrl-D 複製。再接著只按住 Ctrl 鍵，可以依據格子點來移動新增的 GO。利用這個作法，做一個階梯出來。

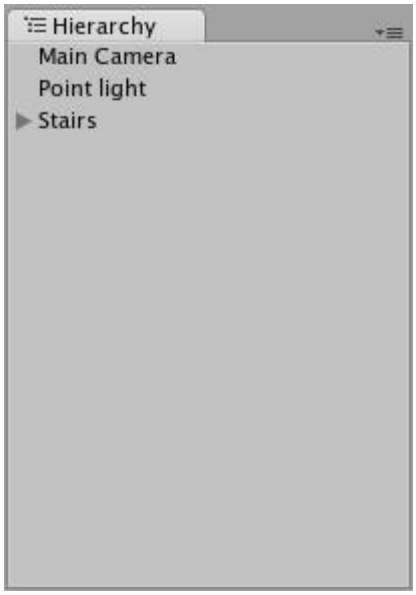
3.3. 學習如何 群組 GameObject 以及 prefab

建立五個 cube (或者是接續上面的範例)，建立一個空的 GameObject，命名爲：Stairs
選取那五個 cube, 拖曳至 Stairs 內。
練習一下複製，變形等操作，然後，刪除調新增的 stairs，只留一個 Stairs 在場景。

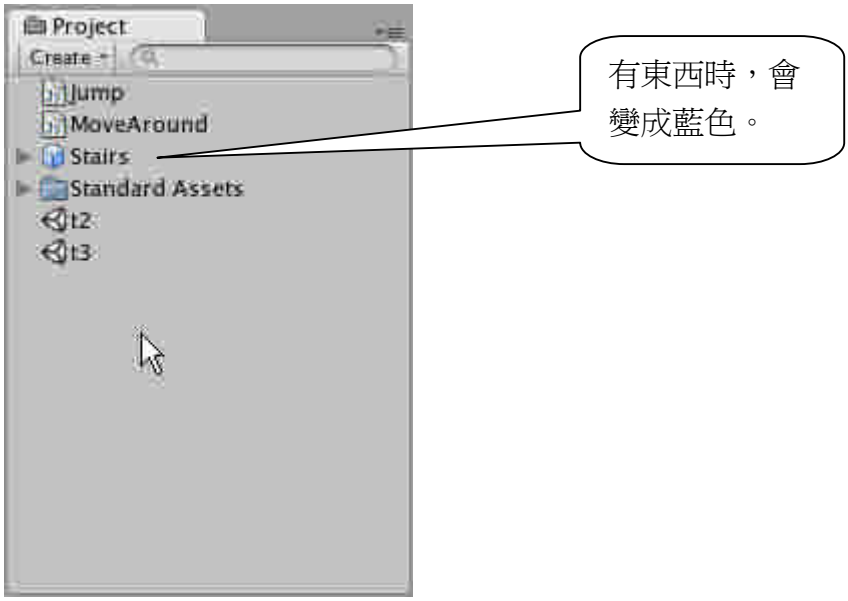
Prefabs：
需要的時候才出場，可以被重複使用，可以在 script 中建構的東西。

在「Project」面版內，點滑鼠右鍵，Create -> Prefab

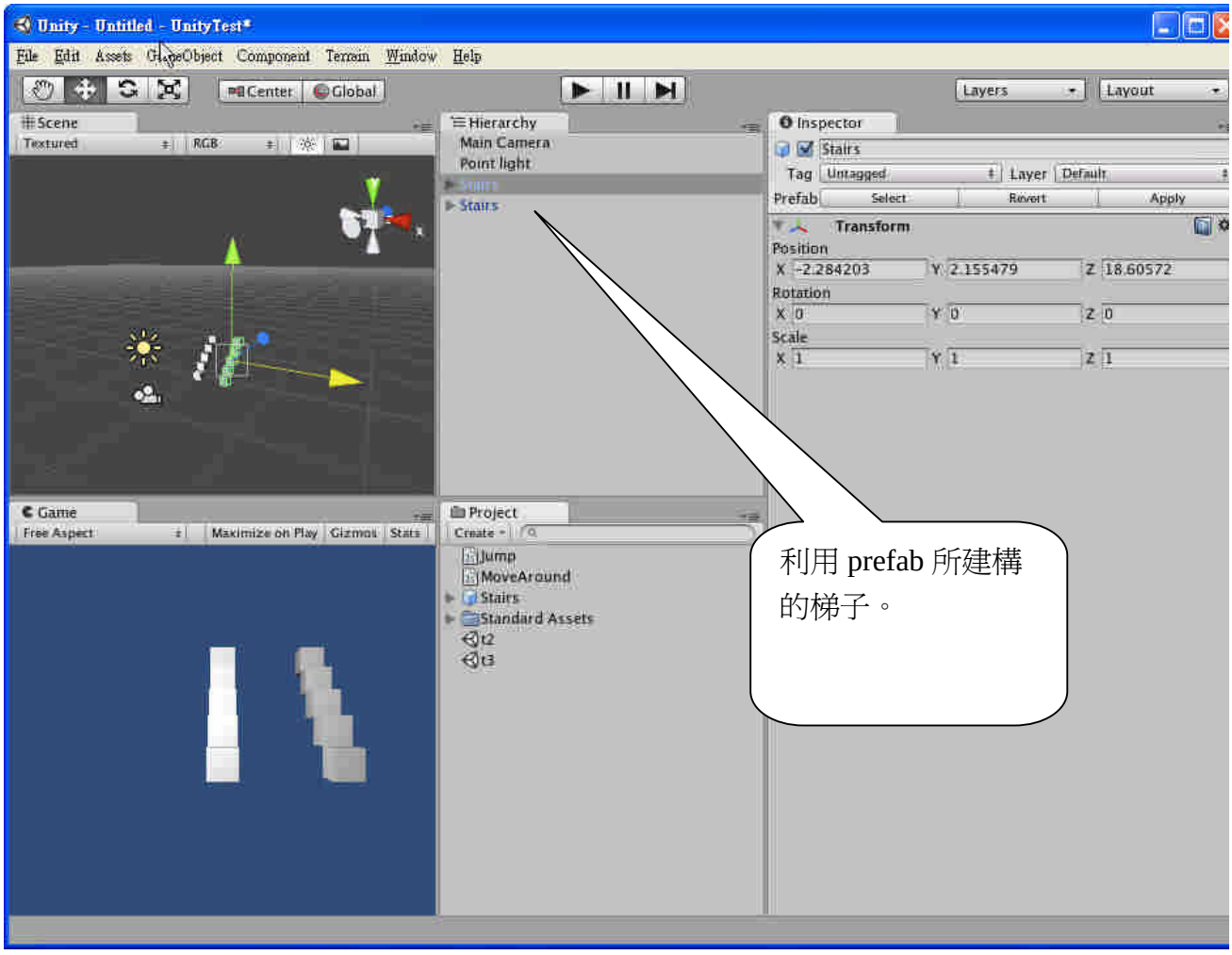




建好之後，把在Hierarchy面版中的Stairs(這是Game Object) 拉到 Project 面版中的 Stairs (這是 prefab，在 Project，將來可以拉很多個到場景中)



這時候，你可以把 Hierarchy 面版中的 Stairs 刪除，將來有需要 Stairs，從 Project 中拉進去場景即可。



練習：建立一個簡單的射擊場景：Bullet prefabs
在一個射擊遊戲中，子彈可以一直發射。因此子彈就可以把它建立成 prefab。

建立一個簡單的射擊場景：Bullet prefabs
目標：使用者按空白鍵時，會發射子彈
步驟(活動圖)：

1. 建立一個 Cube，x, y, z 設成 0，命名爲「Bullet」，然後，記得 Attach 「Rigidbody」。(有了 Ridigbody 屬性，它才會有物理的行爲)
2. 在「Project」面版中，建立一個 prefab，叫 Bullet。
3. 把 GameObject Bullet 拖到 prefab Bullet 上。然後可以把 GameObject Bullet 刪除掉。
4. 建立一個新的 script:Shoot

Script: Shoot

```
var prefabBullet:Transform;

function Update () {
    if(Input.GetButtonDown("Jump"))
    {
        var instanceBullet = Instantiate(prefabBullet,
transform.position, Quaternion.identity);
    }
}
```

變數 prefabBullet，用來連結到我們的 prefab Bullet
var instanceBullet = Instantiate(prefabBullet, transform.position, Quaternion.identity);
上面的程式碼是用來建構我們的子彈
prefabBullet ：要建構的 prefab 變數
transform.position：目前這個 game object 的 transform 的位置。(因爲等一下我們會把這個 script attach 到 Main Camera, 所以，transform 指的是 Main Camera 的 transform 屬性。
Quaternion.identity：第三個引數跟旋轉有關，目前使用預設的旋轉

5. 上面的那句程式碼一旦執行時，我們的 prefab 就會出現在 Viewport 中(也就是照相機的視窗中)
6. 寫完後，記得把 script attach 到 Main Camera
7. 測試一下。(一直按空白鍵)

這時候，可能會發現，有一大堆問題：

(1) 不要忘了 Shoot 到 Main Camera

(2) 不要忘了連結 Bullet prefab 到 Shoot 中

然後，再測試。還是怪怪的。原來是做 Bullet prefab 時，忘了指定 Ridigbody 給 GameObject。所以，重新製作做一個 prefab。(如果你有照著每一步驟做，應該不會有這個問題。) 再測試一下。(一直按空白鍵)

應該有看到一大堆 Cube 一直跑出來，掉下去。這就對了。會有這種現象是因爲我們還沒有賦予子彈一個向前的推力 (Force)。打開 Shoot script，新增紅色字的部分。

Script: Shoot

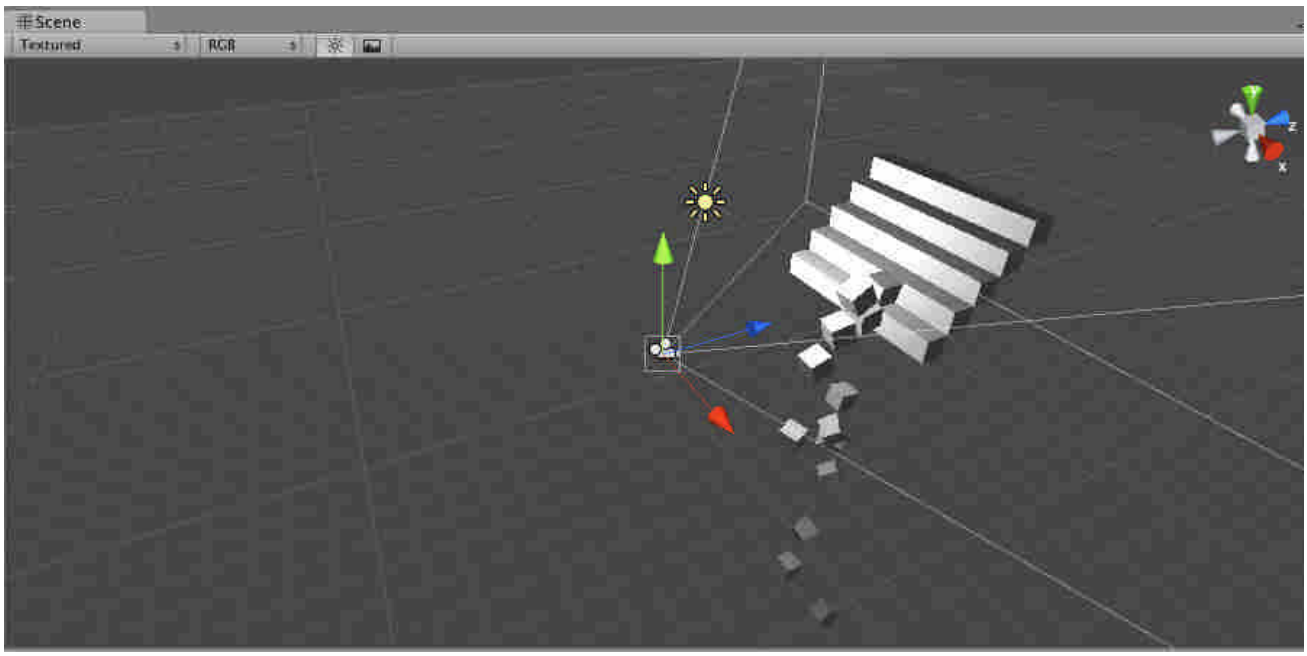
```
var prefabBullet:Transform;
var shootForce:float;

function Update () {
    if(Input.GetButtonDown("Jump"))
    {
        var instanceBullet = Instantiate(prefabBullet,
transform.position, Quaternion.identity);
        instanceBullet.rigidbody.AddForce(transform.forward *
shootForce);
    }
}
```

增加 Force

instanceBullet.rigidbody.AddForce(transform.forward * shootForce);

這句話的意思是：對於所創建的 instanceBullet，在它的屬性 rigidbody 下，新增一個推力(AddForce)，引數中，我們需給定一個推力的大小與方向 (它是一個向量)。

shootForce 應該設定在 1000 左右。

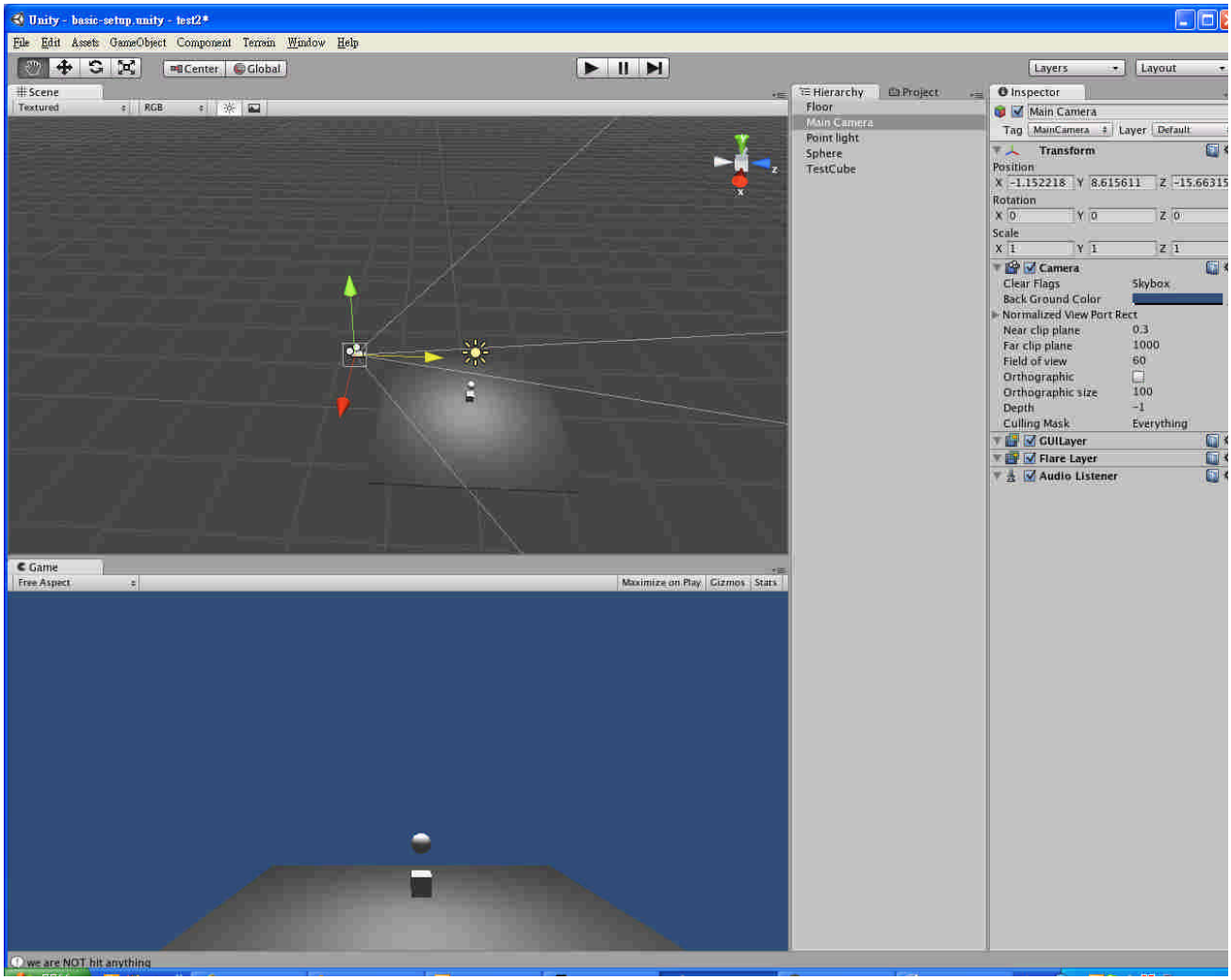


可以放一塊板子在下方。子彈不會一直掉下去。

4. 3D Game 製作-偵測滑鼠點擊

4.1. 配置場景

首先，先建好預設的場景



4.2. 撰寫 Script

```
function Update () {
    var DoubleClickRotateAmount = new Vector3(0, 15, 0);

    if(Input.GetMouseButtonDown(0))
    {
        var mainCamera = FindCamera();
        // We need to actually hit an object
        var hit : RaycastHit;
       if (Physics.Raycast(mainCamera.ScreenPointToRay(Input.mousePosition),
```

```
hit, 1000)) {
                if(hit.collider.gameObject.name == "TestCube"){
                    hit.transform.position.y   = transform.position.y    + .1 *
speed;
                    hit.transform.Rotate(DoubleClickRotateAmount);
                }
            }else{
                print("we are NOT hit anything");
            }
    }
}
function FindCamera ()
{
    if (camera)
        return camera;
    else
        return Camera.main;
}
```

說明：

- FindCamera() 這個函示是用來找到在 Scene 中的鏡頭物件。
- 這相關物件請參閱網路之 API 手冊。

```
if(Input.GetMouseButtonDown(0))   // 0:滑鼠左鍵。偵測是否按下。
{
    var mainCamera = FindCamera(); // 找出鏡頭
    var hit : RaycastHit; // 這是一個資料結構，用來儲存 hit 到的物件的資訊
   if (Physics.Raycast(mainCamera.ScreenPointToRay(Input.mousePosition),
hit, 1000)) {
        if(hit.collider.gameObject.name == "TestCube"){
            //執行相關的動作
            hit.transform.position.y   = transform.position.y    + .1 * speed;
            hit.transform.Rotate(DoubleClickRotateAmount);
        }
    }else{
        print("we are NOT hit anything");
    }
}
```

// 假如有 Hit 到，回傳 true。資料放在 hit 變數中。1000 是指距離。
Physics.Raycast(mainCamera.ScreenPointToRay(Input.mousePosition), hit, 1000)

// 用這個方式取出被 hit 的物件名稱，比對用的。所以，Game Object 都要有賦予一個適當的名稱。
hit.collider.gameObject.name

2. 把 script attach 到 TestCube 這個 Game Object。
3. 測試。

其他知識：

Use **OnMouseOver** with an object that has a collider (this uses **raycasting**, but you don't have to program it explicitly):

```
function OnMouseOver () {
    if (Input.GetMouseButtonDown(1)) {
        // Do right-click stuff here
    }
}
```

相關參考：

1. **Getting the object's name from RaycastHit**, http://answers.unity3d.com/questions/3007/getting-the-objects-name-from-raycasthit

5. 3D Game 製作-燈的開關

5.1. 配置場景

先在 Unity 3D 的場景中，建立一個 Cube 當作 switch。名稱爲 **ComputerRoomSwitch**。

5.2. 撰寫 Script

```
var linkedLight : Light;

function Update () {
	if(Input.GetMouseButtonDown(0))
	{
		var mainCamera = FindCamera();

		var hit : RaycastHit;
		var v3 : Vector3 = Vector3(Screen.width/2,Screen.height/2); // 螢幕的中央座標

		if (Physics.Raycast(mainCamera.ScreenPointToRay(v3),  hit, 100)) {
			if(hit.collider.gameObject.name == "ComputerRoomSwitch"){
				linkedLight.enabled = !linkedLight.enabled;
			}
		}else{
		}
	}

}

function FindCamera () {
	if (camera)
		return camera;
	else
		return  Camera.main;

}

function Start () {
linkedLight.enabled = false;
```

```
}    else
          return   Camera.main;
}
```

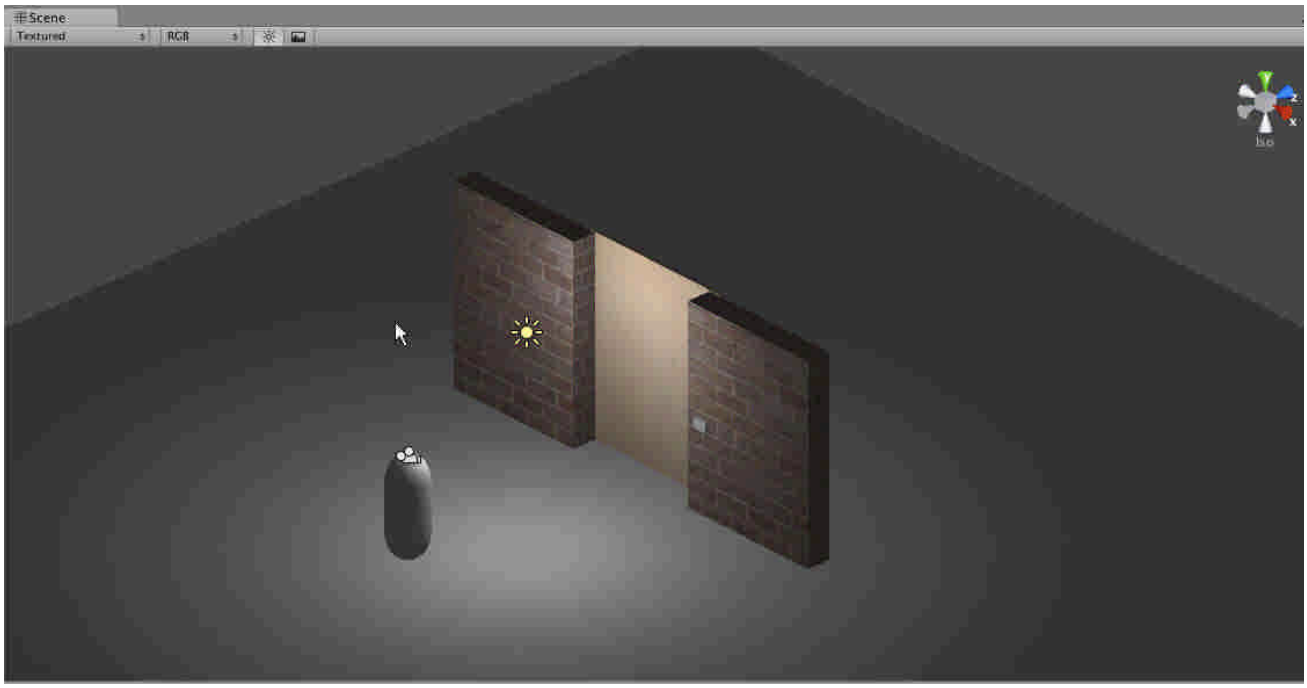
說明：

- FindCamera() 這個函示是用來找到在 Scene 中的鏡頭物件。
- 這相關物件請參閱網路之 API 手冊。
- 把 Script 連結到名稱爲 **ComputerRoomSwitch** 的 Game Object 上。

相關參考：

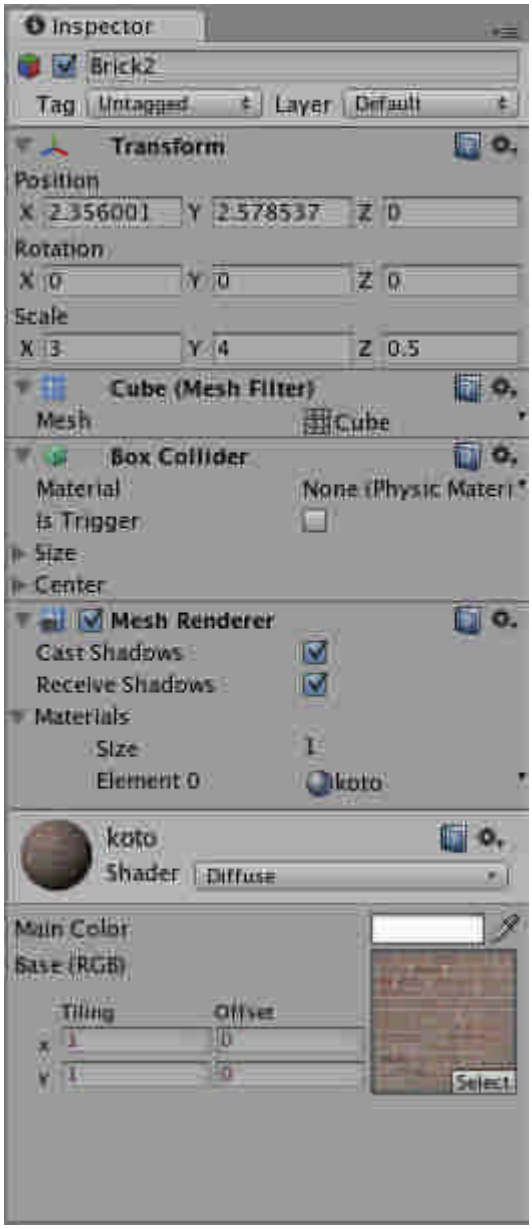
6. 3D Game 製作-開門的作法(左右)

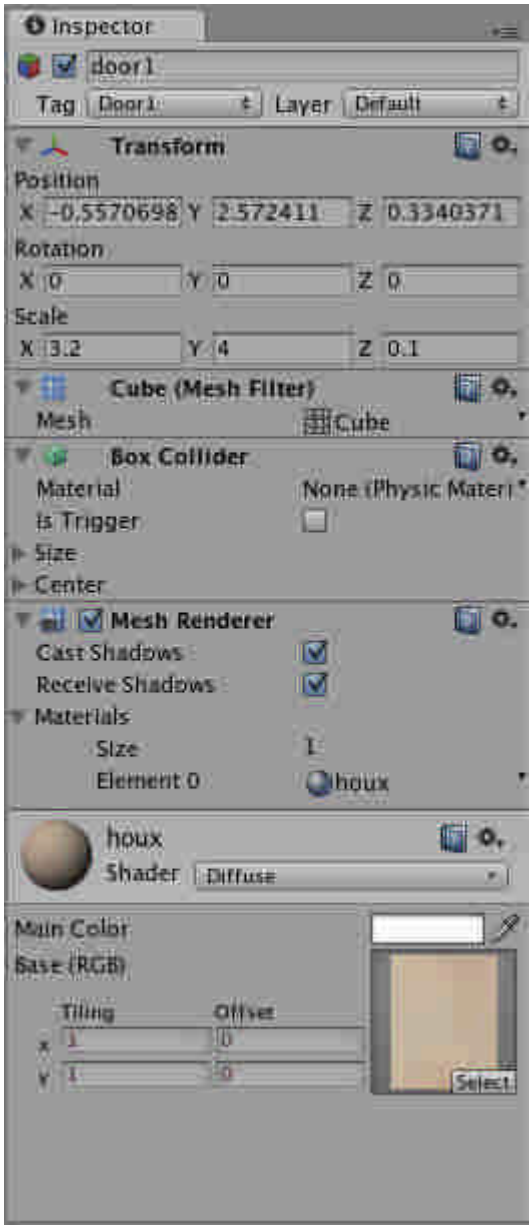
6.1. 配置場景



6.2. 實驗

走向門，按一下門右方的按鈕。門會由左向右打開。再按一次，門會關起來。


Inspector
Brick2
Tag Untagged
Layer Default
Transform
Position
X 2.356001 Y 2.578537 Z 0
Rotation
X 0 Y 0 Z 0
Scale
X 3 Y 4 Z 0.5
Cube (Mesh Filter)
Mesh Cube
Box Collider
Material None (Physic Mater)
Is Trigger
Size
Center
Mesh Renderer
Cast Shadows
Receive Shadows
Materials
Size 1
Element 0 koto
koto
Shader Diffuse
Main Color
Base (RGB)
Tiling
Offset
x 1 0
y 1 0
Select


Inspector
door1
Tag Door1
Layer Default
Transform
Position
X -0.5570698 Y 2.572411 Z 0.3340371
Rotation
X 0 Y 0 Z 0
Scale
X 3.2 Y 4 Z 0.1
Cube (Mesh Filter)
Mesh Cube
Box Collider
Material None (Physic Mater)
Is Trigger
Size
Center
Mesh Renderer
Cast Shadows
Receive Shadows
Materials
Size 1
Element 0 houx
houx
Shader Diffuse
Main Color
Base (RGB)
Tiling
Offset
Select

6.3. 撰寫 Script

```
var door1opened : boolean = false;
var door1closed : boolean = false;
var door1opening : boolean = false;
var door1closing:boolean = false;
var timer : float = 0.0;
var shouldstop:boolean = false;

function Update () {
    if(Input.GetMouseButtonDown(0))  {
        var mainCamera = FindCamera();

        var hit : RaycastHit;
        var v3 : Vector3 = Vector3(Screen.width/2,Screen.height/2);

        if (Physics.Raycast(mainCamera.ScreenPointToRay(v3),  hit, 100)) {
            if((hit.collider.gameObject.name == "Switch1") && (door1opened
== false)){
                door1opening = true; // 這個的作用像棋標，用來控制門的移動
            }

            if((hit.collider.gameObject.name == "Switch1") && (door1opened
== true)){
                door1closing = true; // 這個的作用像棋標，用來控制門的移動
            }
        }else{
        }
    }

    // 開門的程式碼
    if((door1opening) && (door1opened == false)){
        var door1 = GameObject.FindWithTag("Door1");
        door1.transform.Translate(1 * Time.deltaTime, 0, 0);

        timer += Time.deltaTime;
        if((timer >= .5) && (timer < 1.5))          {
            door1.transform.Translate(1.5 * Time.deltaTime, 0, 0);
        }
```

```
            if(timer >=1.5) {
                door1opening = false;
                door1opened = true;
                door1closed = false;
                door1closing = false;
                timer = 0.0;
            }
        }

        // 關門的程式碼
        if((door1closing) && (door1opened == true)){
            door1 = GameObject.FindWithTag("Door1");
            door1.transform.Translate(-1 * Time.deltaTime, 0, 0);

            timer += Time.deltaTime;
            if((timer >= .5) && (timer < 1.5))          {
                door1.transform.Translate(-1.5 * Time.deltaTime, 0, 0);
            }
            if(timer >=1.5) {
                door1closing = false;
                door1closed = true;
                door1opening = false;
                door1opened = false;
                timer = 0.0;
            }
        }
}

function FindCamera () {
        if (camera){
            return camera;
        }
        else
            return   Camera.main;
}
```

說明：

- FindCamera() 這個函示是用來找到在 Scene 中的鏡頭物件。

- 這相關物件請參閱網路之 API 手冊。
- 把 Script 連結到 First Person Controller。

相關參考：

1. **Simple collision script not working, what am I doing wrong?**
http://answers.unity3d.com/questions/939/simple-collision-script-not-working-what-am-i-doing-wrong

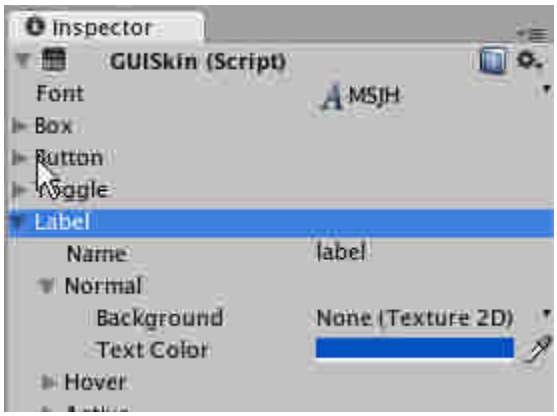
7. 3D Game 製作-安裝中文字體

7.1. GUI 介面

[Unity 3D]
在 Unity 3D 中，GUI 的介面顯示是由 GUI Style 及 GUI Skin 來控制。因此，首先，先建立一個 GUI Skin。

7.2. 建立 GUISkin

在 Project 面版中，點滑鼠右鍵，Create -> GUI Skin。將此 Skin 命名爲 **ChineseFontSkin**。然後，點擊新建的 ChineseFontSkin，在右側的屬性面版中，展開 Label 並且修改 Text Color 屬性。這裡我們只測試 Label。其他的可以類推。



7.3. 建立使用 skin 的 script

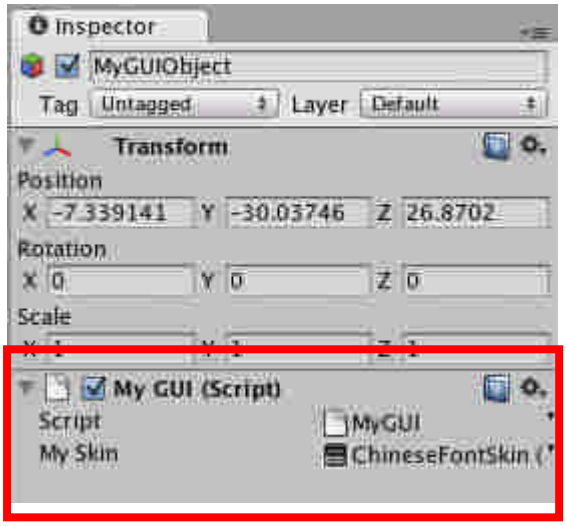
接著，在 Project 中建立一個 Javascript，命名爲 **MyGUI**，我們將在此 script 中建立 GUI 物件。滑鼠右鍵點選 MyGUI，然後選擇「show in explorer」，利用 Editplus (或任何可以儲存成 UTF-8 的文字編輯器)把文件打開，填入：

```
var mySkin:GUISkin;
function OnGUI () {
    if(mySkin){
        GUI.skin = mySkin;
    }

    GUI.Label(Rect(10, 10, 400, 300), "98 年度 專題製作 電腦教室 3D 虛擬環境");
    //GUI.Button(new Rect(420, 580, 100,30), "繼續");
}
```

從程式碼中我們可以看到這個 script 接受一個 GUISkin 類型的物件。
注意：script 檔案一定要存成 UTF-8 的編碼形式，不然，等一下中文跑不出來。

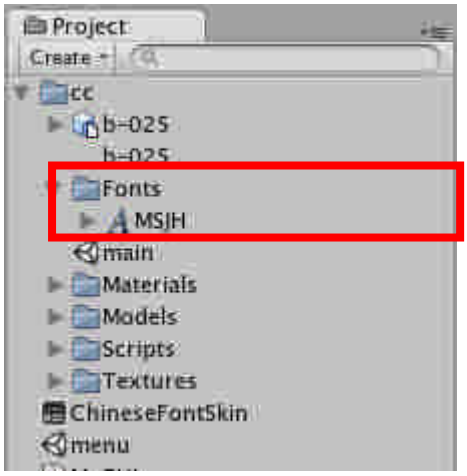
7.4. 建立一個測試用 GameObject

接著我們建立一個測試用 GameObject，命名爲 MyGUIObject。**然後把 MyGUI 拖放到 MyGUIObject 上。並且把 ChineseFontSkin 拖放到 My Skin 中**。

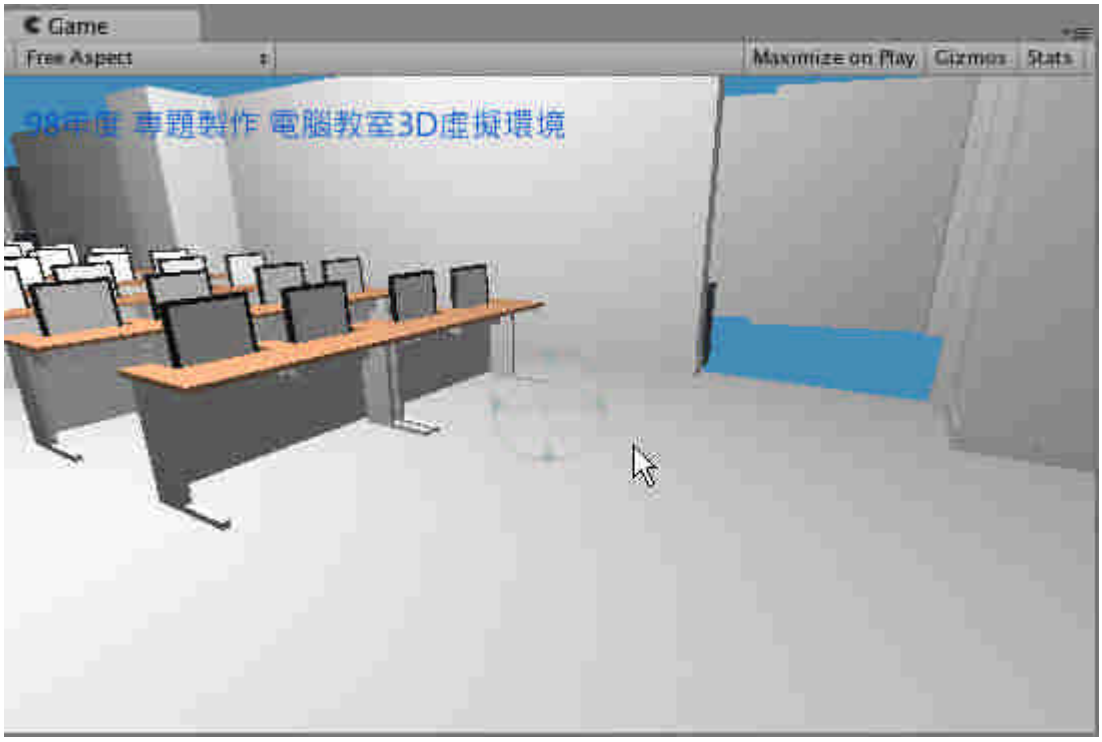


7.5. 安裝中文字形

本範例以微軟正黑體爲例，(註：並不是每種字體都可能安裝的成功。)
先在計畫 cc 下建構一個目錄夾 Fonts。然後，從 C:\WINDOWS\Fonts 下找到微軟正黑體。把檔案 copy 至 ..\cc\Fonts 目錄夾中。這時候，切換回 Unity 3D，螢幕會沒有反應，等兩三分鐘等字形轉換完成。完成後，會在 Unity 中看到。



選擇 ChinesFontSkin，然後，把 Font 屬性改用微軟正黑體。
執行一下遊戲，應該可以看到中文字了。



Reference:

1. http://web3d.5d6d.com/viewthread.php?tid=1806

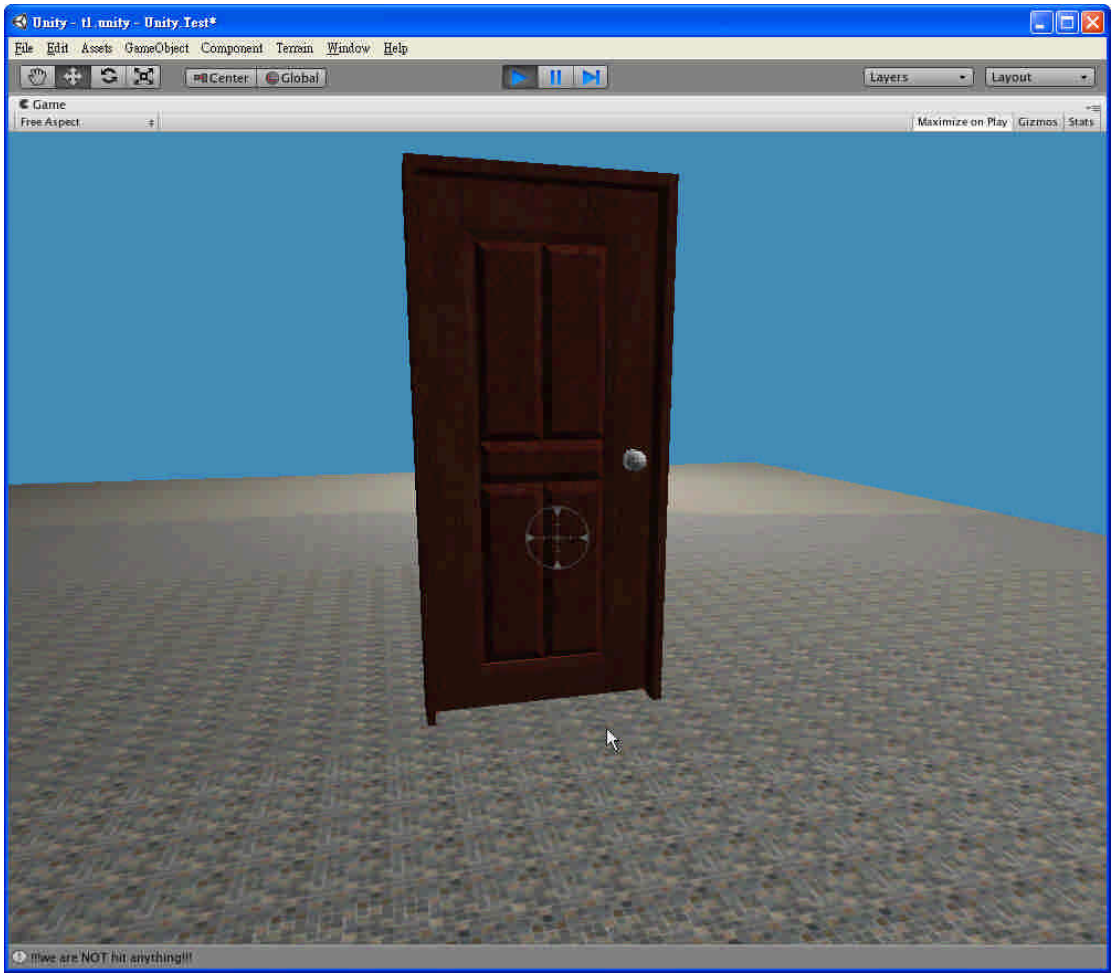
8. 3D Game 製作-偵測點擊物件

8.1. 配置場景

[Unity 3D]

如何在 3D 環境之中，偵測點擊的物件，然後做相關對應的動作是一個很基本的技巧。

首先，下圖是我們設定的場景：



這裡我們使用到了之前的 corsshair.js

corsshair.js

```
var crosshairTexture : Texture2D;
var position : Rect;
function Start()
{
    position = Rect( ( Screen.width - crosshairTexture.width ) / 2,
( Screen.height - crosshairTexture.height ) / 2,
```

```
crosshairTexture.width, crosshairTexture.height );

    //Screen.showCursor = false;

}
function OnGUI()
{
    GUI.DrawTexture( position, crosshairTexture );
}
```

註：記得要把 scrip 連結到一個 Game Object 上。設定了中心點之後，我們就可以做 3D 空間的射線偵測(Ray Casting)。

從 Blender 3D 中找一個門的模型，並且利用 UV map 來貼上材質。讓場景比較逼真一點。

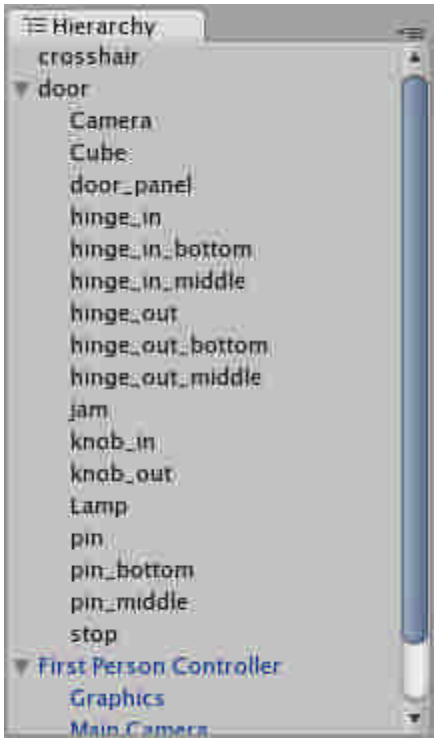
open_nobe.js

```
function Update () {

    if(Input.GetMouseButtonDown(0))
    {
        var mainCamera = FindCamera();

        var hit : RaycastHit;
        var v3 : Vector3 = Vector3(Screen.width/2,Screen.height/2);


        if
(Physics.Raycast(mainCamera.ScreenPointToRay(v3),  hit,
100)) {
            if((hit.collider.gameObject.name == "door_panel")){
                print("HIT hit door_panel");
            }else{
                print("NOT hit door_panel");
            }
        }else{
            print("!!!we are NOT hit anything!!!");
        }
```

```
    }
}


function FindCamera ()
{
    if (camera){
        return camera;
    }
    else
        return   Camera.main;
}
```

要瞭解這個程式碼，首先要清楚這個 door 物件模型的 mesh 結構。

我們可以從右圖中看到，這個 door 模型是由許多的物件組成。Door 下方有一個 door_panel，就是門的面版。我們希望做的偵測就是當角色的視線(螢幕中心點)對著門板，並且敲下滑鼠左鍵時，我們輸出訊息：「HIT hit door_panel」，如果沒有 hit 到面板時，我們輸出訊息：「NOT hit door_panel」。如果沒有點擊到任何場景中的任何物件時，我們輸出訊息：「!!!we are NOT hit anything!!!」



基本上，這就是程式碼中的邏輯說明。

因此，這些都很容易理解。最重要的一行程式碼是：

Physics.Raycast(mainCamera.ScreenPointToRay(v3), hit, 100

mainCamera 就是主要相機

v3 是螢幕中心點座標，Vector3(Screen.width/2,Screen.height/2)

因此，在執行射線偵測(Raycast)時，就是從相機(剛好就位於螢幕中心點)射出一條直線到 v3 螢幕中心點 (要用三維的觀念去想)。在這條線中，任何被射擊

到的物件會被儲存在 hit 這個變數中。

Hit 的型態是 RaycastHit，而不是 Game Object。這點要注意。因此，要取出 game object 出來，要用 hit.collider.gameObject。

參考資料：

1. http://unity3d.com/support/documentation/ScriptReference/RaycastHit.html

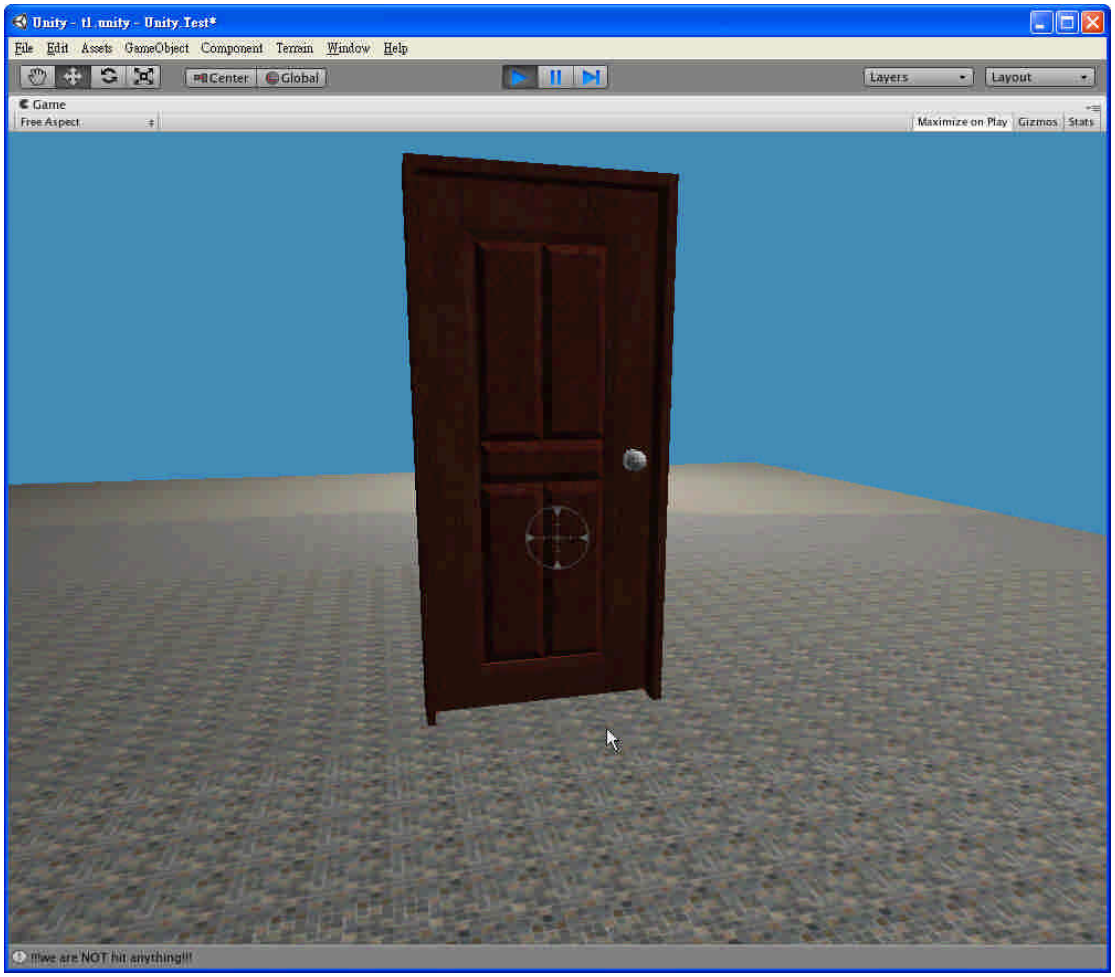
9. 3D Game 製作-改變鼠標

9.1. 配置場景

[Unity 3D]

對於場景中一些可以互動的物件，如果我們希望在滑鼠移動至上方時將鼠標改變，則對於使用者來說是一個很有用的提示方式。當使用者看到鼠標有變化時，只要點擊一下，我們就可以做一些變化，例如：開門，開燈，變換場景等效果了。

首先，下圖是我們設定的場景：



9.2. Script 程式碼

這裡我們使用到了之前的 corsshair.js

corsshair.js

```
var crosshairTexture : Texture2D;
var position : Rect;
function Start()
{
```

```
    position = Rect( ( Screen.width - crosshairTexture.width ) / 2,
( Screen.height - crosshairTexture.height ) / 2,
crosshairTexture.width, crosshairTexture.height );

    //Screen.showCursor = false;

}
function OnGUI()
{
    GUI.DrawTexture( position, crosshairTexture );
}
```

註：記得要把 scrip 連結到一個 Game Object 上。設定了中心點之後，我們就可以做 3D 空間的射線偵測(Ray Casting)。

從 Blender 3D 中找一個門的模型，並且利用 UV map 來貼上材質。讓場景比較逼真一點。

change_cursor.js

```
var cursor : Texture2D;
var link : Texture2D;
var normal : Texture2D;

/*On mouse over hide the cursor and show the "link" cursor.*/

function OnMouseOver () {
    Screen.showCursor = false;
    cursor = link;
    }

/*On mouse exit reset to the "normal" cursor. You can leave the
"normal" variable blank or populate with any texture.*/

function OnMouseExit () {
    Screen.showCursor = true;
    cursor = normal;
    }
```

```
/*Make the "link" texture follow the cursor.*/

function OnGUI () {
    GUI.Label(Rect(Input.mousePosition.x-12, Screen.height -
Input.mousePosition.y-10, 100, 100), cursor);
}
```

首先找一個圖樣，需要透明背景的圖。這裡我們用了一個 cursor-hand。

這個圖要拖曳連結到 change_cursor.js script 上面中的 **link** 變數上，其他兩個變數可以不用設定。

在 script 中，我們有兩個方法：OnMouseOver() 以及 OnMouseExit()。因此，當 mouse 在物件上面時我們就變換鼠標，一旦離開了物件，我們就把鼠標在換回原來的。最後，需要用一個 OnGUI()把 cursor 畫出來。這裡，使用到了 GUI.Label 物件來執行這個工作。

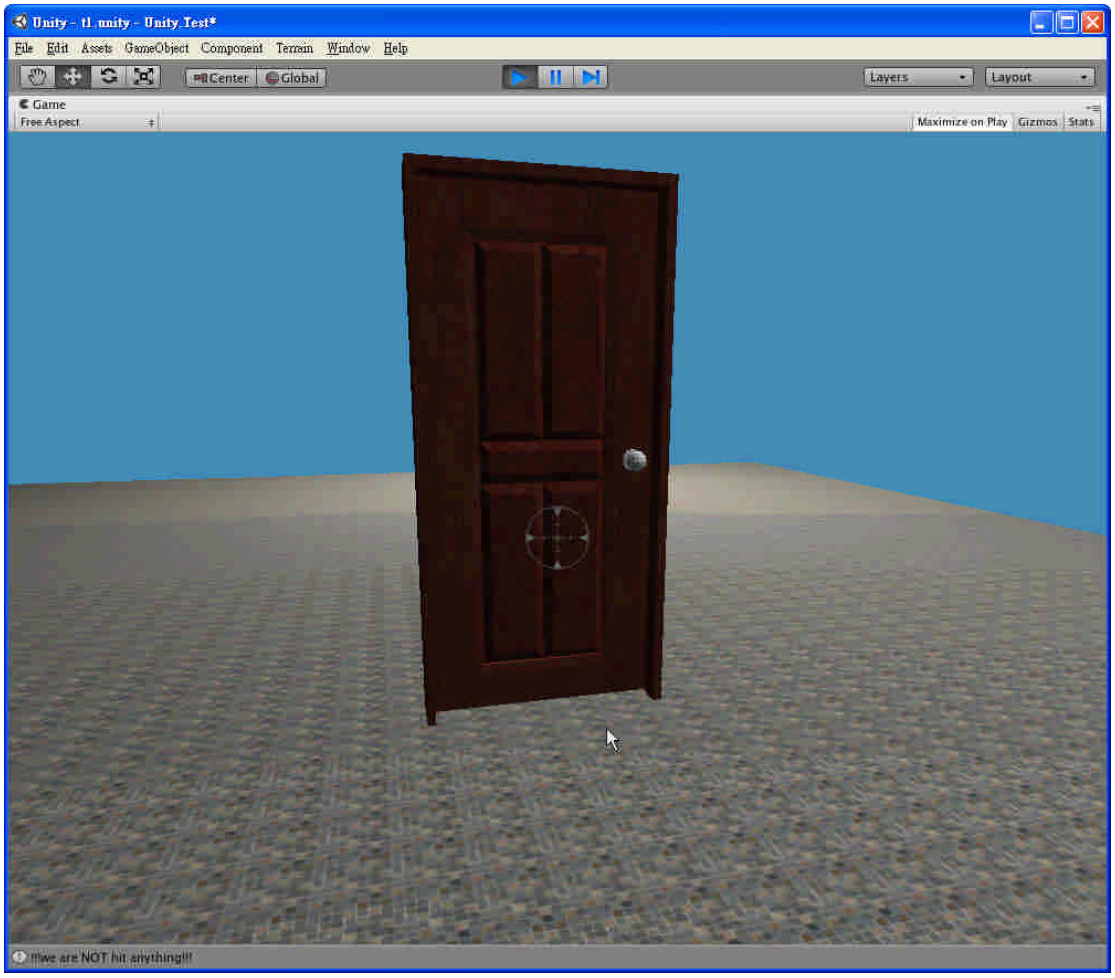
10. 3D Game 製作-開門的作法(旋轉)

10.1. 配置場景

[Unity 3D]

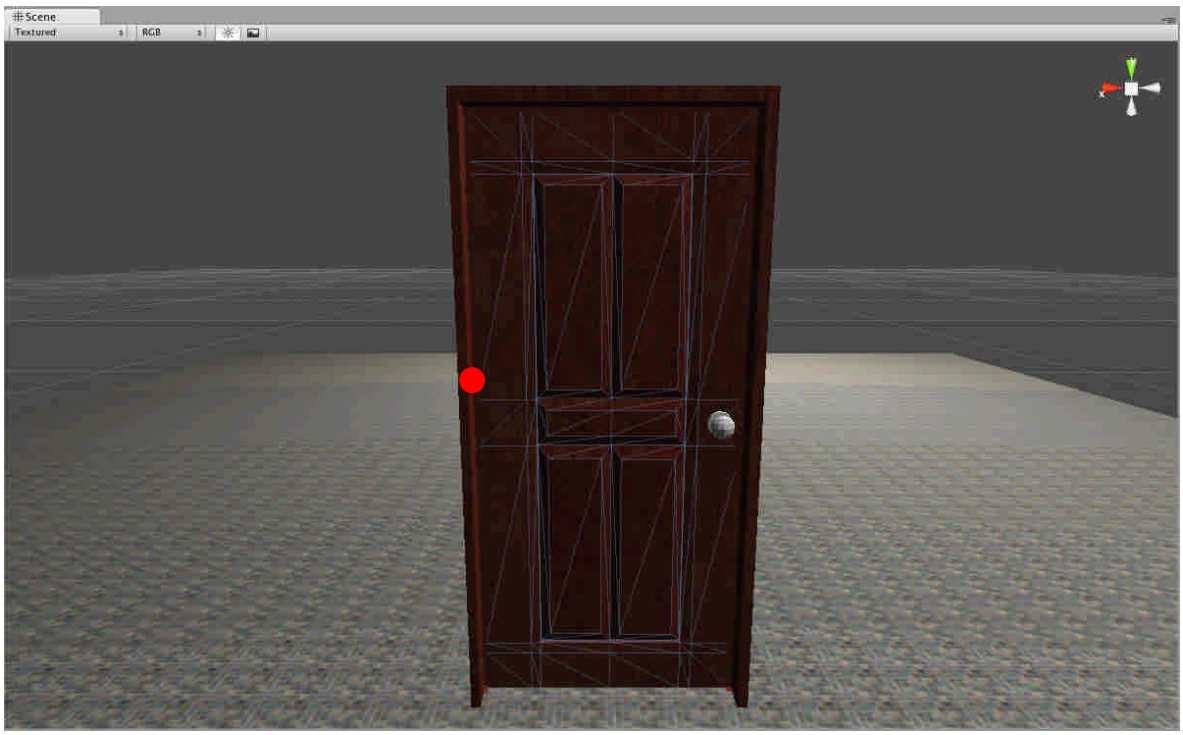
因爲開門的動作可以想像成是一個動畫(animation)的過程，因此，使用動畫是最簡單的方法。在 Unity 3D 中，製作動畫的過程很類似於使用 Flash。

下面是我們的場景。

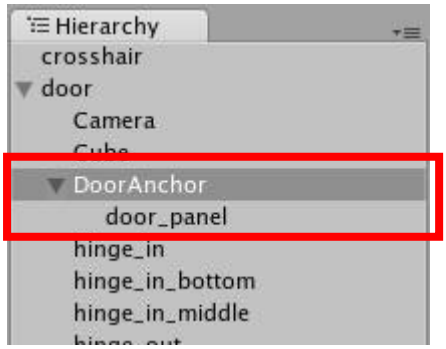


在做動畫時，我們可以操控 mesh 的 x, y, z 以及 x, y, z 的旋轉角度。不過，如果只是讓門對著 y 軸(往上方向)旋轉，則門會以 mesh 的中心點做轉動。這並不是我們希望的行爲。

解決這個問題的方法是建立一個空的 game object 當作旋轉的參考物件, 然後把門拖曳到這個空的 game object 下面當成子元件即可。下圖顯示參考物件的位置：這個位置也是一般 hinge joint 所在的位置。



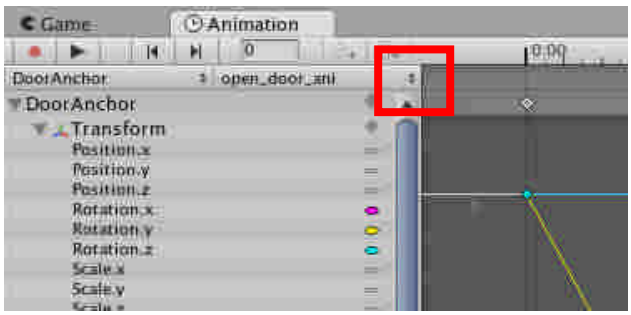
下圖顯示門板與參考物件的結構關係：



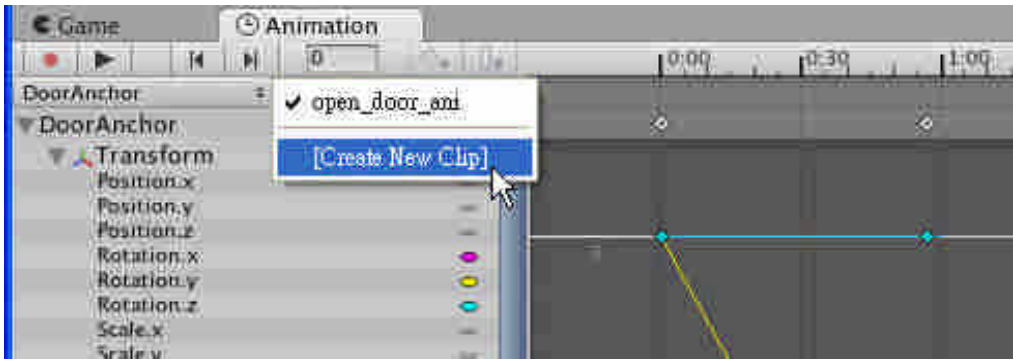
參考物件的名稱爲 DoorAnchor，door_panel 是門板。

接下來就是錄製門開起來到停止的動畫。

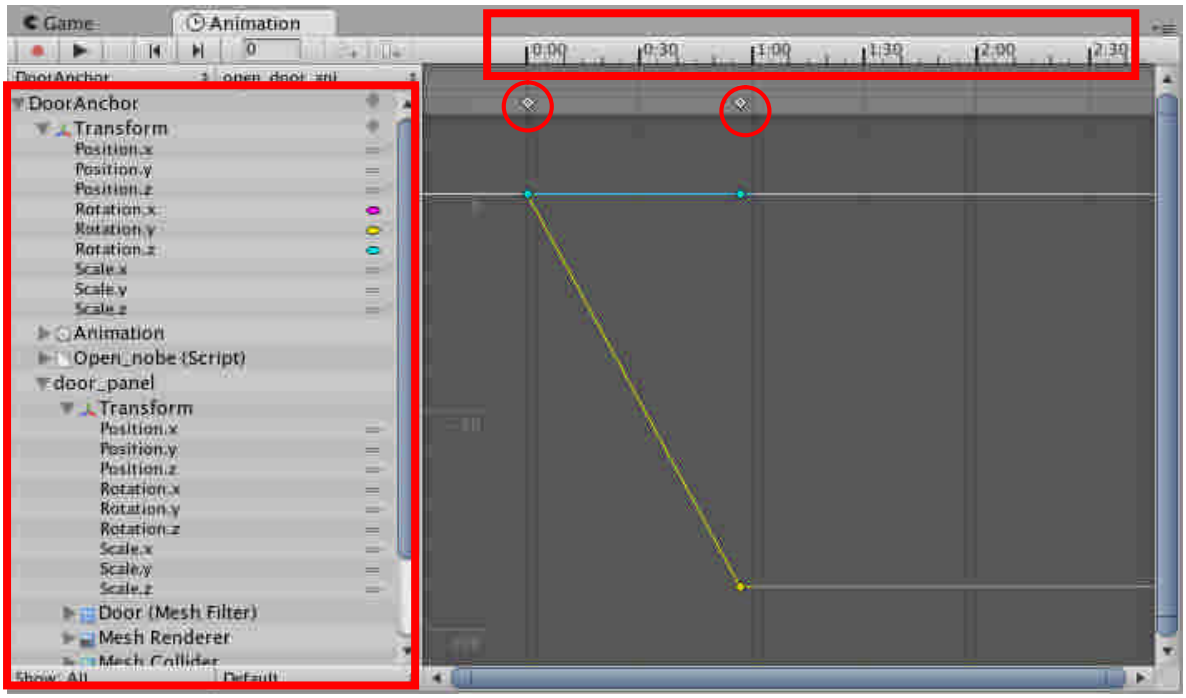
首先，點選 DoorAnchor 物件，然後選 「**Window -> Animation**」來開啓動畫面板。首先點選下圖中的一個小三角形，選擇建立一個新的影片[Create New Clip]。



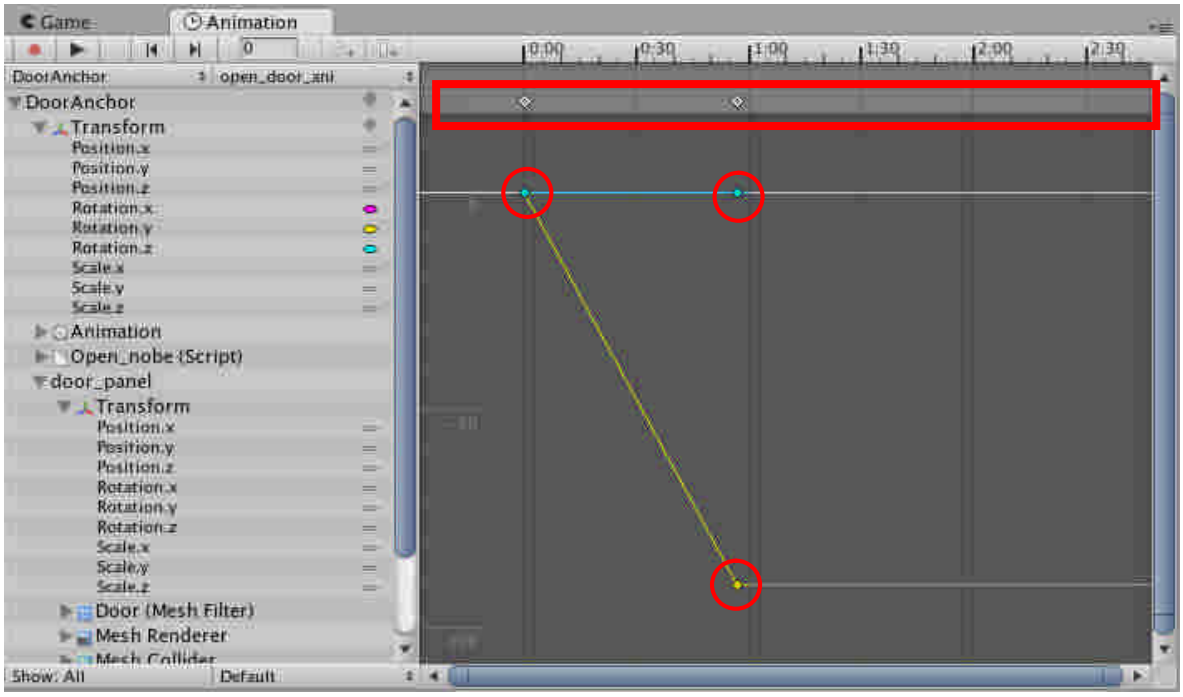
輸入影片名稱 open_door_ani。



在 Animation 中上方的數字是時間軸，單位爲分鐘。左邊面板則顯示了目前物件可以改變的 transform 屬性。對於不同的屬性，Unity 3D 使用了不同的顏色來代替。



圖中有一個小圓圈的代表 keyframe(關鍵影格)。



我們在上方顯示的區域使用滑鼠右鍵，呼叫出彈跳視窗，用以新增關鍵影格等等功能。我們也可以直接拖曳圖中的圓圈部分，改變曲線的方式。

開始錄製動畫時，請先點選左上方的紅色鈕。
然後，請在左側面板找到「Rotation.y」，按一下它右方的鈕，然後選擇「Add Curve」。在 timeline 中會跑出一條紅線。在 timeline 上，直接拖曳紅線到 1:00 處，然後，把門旋轉 90 度。好了。這個時候，如果你用滑鼠移動 timeline 中的紅線，3D 場景的物件應該有動作了。這個很類似 Flash 中的補間動畫。這樣門的 animation 就做好了。

做好了 animation，我們一定要記得把 animation 拖曳到某個 Game Object 上。在這裡，我們就把「open_door_ani」animation 連結到 DoorAnchor 上。

10.2. Script 程式碼

接下來是 open_door.js 程式碼。

open_door.js

```
var opened = false;
var opening = false;
function Update () {
    if(Input.GetMouseButtonDown(0))
    {
```

```
        var mainCamera = FindCamera();
        var hit : RaycastHit;
        var v3 : Vector3 = Vector3(Screen.width/2,Screen.height/2);

        if (Physics.Raycast(mainCamera.ScreenPointToRay(v3),   hit,
100)) {
            if((hit.collider.gameObject.name == "door_panel")){
                if(opened == false){
                    var gooo = GameObject.Find("DoorAnchor");
                    var a = gooo.animation.Play("open_door_ani");
                }
            }else{
                print("NOT hit door_panel");
            }
        }else{
            print("!!!we are NOT hit anything!!!");
        }
    }
}

function FindCamera ()
{
    if (camera){
        return camera;
    }
    else
        return   Camera.main;
}
```

基本上跟之前的程式碼都很像。我們用 ray cast 來偵測使用者是不是有 hit 到門板，如果有的話，我們就找出 DoorAnchor 物件，然後呼叫 animation「open_door_ani」。

```
var gooo = GameObject.Find("DoorAnchor");
var a = gooo.animation.Play("open_door_ani");
```

請注意，因爲參考點物件只是一個空的 game object，因此，hit 偵測不可以用它，只能使用面板(也可以改成把手)，但是旋轉的動畫是掛在 DoorAnchor

上，不是門板。因此，我們要先找出 DoorAnchor，然後才執行動畫。