CHAPTER 07

角色移動原理 (INPUT)

HORAZON

手遊程式設計

章節目標

- 了解電腦如何讀取鍵盤輸入 (Input)
- 使用程式控制剛體速度 (Rigidbody Velocity)
- 解決角色轉向問題 (Flip)

1. 電腦怎麼知道我按了鍵盤？

Unity 提供了一個簡單的介面 `Input`。

我們最常用的是「軸向 (Axis)」概念：

- **Horizontal (水平)**：對應 A/D 鍵 或 左/右鍵。
- **Vertical (垂直)**：對應 W/S 鍵 或 上/下鍵。

數值範圍：

- 沒有按：**0**
- 按右 (D)：**1**
- 按左 (A)：**-1**

2. 撰寫移動腳本

打開上一章建立的 `PlayerController` (或新建一個)，修改 `Update`：

```
public float moveSpeed = 5f; // 移動速度變數

void Update()
{
    // 1. 偵測玩家輸入 (-1 ~ 1)
    float h = Input.GetAxisRaw("Horizontal");

    // 2. 測試看看 (看完記得註解掉)
    // Debug.Log("輸入值: " + h);
}
```

3. 讓角色動起來 (RIGIDBODY)

我們前面已經幫主角加上了 `Rigidbody 2D` (剛體)。
要讓他移動，最好的方法是直接**修改他的速度** (Velocity)。

```
// 記得先在開頭宣告變數
public Rigidbody2D rb;

void Update()
{
    float h = Input.GetAxisRaw("Horizontal");

    // 修改剛體速度：(X水平速度, Y原本的垂直速度)
    rb.velocity = new Vector2(h * moveSpeed, rb.velocity.y);
}
```

注意：Y 軸速度要維持 `rb.velocity.y`，不然角色會因為重力歸零而浮在空中！

4. 綁定 RIGIDBODY

寫完程式存檔回到 Unity，你會發現 Script 上多了一個 `Rb` 的欄位顯示 "None"。

重要步驟：

1. 選取主角物件。
2. 看 Inspector 的腳本區塊。
3. 將主角自己的 **Rigidbody 2D** Component，拖曳到這個欄位中。

這叫做「**拉線 (Reference)**」，告訴程式碼要控制哪一個剛體。

5. 解決「月球漫步」問題 (轉向)

試玩一下，你會發現：

- 按右，角色往右走 (正常)。
- 按左，角色往左走，**但是臉還是朝右邊 (倒退嚕)**。

解決方案：
當輸入值為負 (往左) 時，把角色的 `Scale X` (縮放) 變成 `-1`。

6. 控制轉向的程式碼

在 Update 的最下方加入：

```
// 如果 輸入 > 0 (往右)，且原本是面向左，就轉正
if (h > 0)
{
    transform.localScale = new Vector3(1, 1, 1);
}
// 如果 輸入 < 0 (往左)，且原本是面向右，就翻轉
else if (h < 0)
{
    transform.localScale = new Vector3(-1, 1, 1);
}
```

總結

現在你的角色應該可以：

1. 透過 A / D 左右移動。
2. 移動時自動**轉向**。
3. 雖然還不能跳躍，但我們已經邁出了一大步！

下一章，我們要來挑戰平台遊戲最難的邏輯：**跳躍與地板偵測**。