CHAPTER 08

# 跳躍與地面檢測 (PHYSICS LOGIC)

HORAZON

手遊程式設計

# 章節目標

- 實作角色 **跳躍 (Jump)** 功能
- 解決 **無限跳躍 (Infinite Jump)** BUG
- 使用 **Raycast (射線)** 偵測地板

# 1. 跳躍的原理

在物理學中，跳躍就是「瞬間給予一個向上的力」。

Unity 指令：`rb.AddForce()`

```
public float jumpForce = 300f; // 跳躍力道

void Update()
{
    // ... (原本的移動程式碼) ...

    // 如果按下 "Jump" 鍵 (預設是空白鍵 Space)
    if (Input.GetButtonDown("Jump"))
    {
        // 給予一個向上的力 (Vector2.up)
        rb.AddForce(Vector2.up * jumpForce);
    }
}
```

# 2. 嚴重的 BUG：無限連跳

試玩看看，狂按空白鍵。
你會發現主角可以**踩著空氣**一直往上飛，變成「憤怒鳥」了。

**原因**：我們沒有限制「只有在地上」才能跳。
電腦很笨，你叫它跳它就跳，不管你在哪裡。

# 3. 解決方案：地面檢測 (GROUND CHECK)

我們需要一雙「眼睛」，隨時檢查「腳下有沒有地板」。

**Raycast (射線投射)：**
就像從腳底發出一道雷射光，如果有照到東西 (地板)，就回傳 True。

# 4. 實作 RAYCAST

修改程式碼，加入地面檢測：

```
public LayerMask groundLayer; // 設定什麼是地板層
public float rayLength = 0.6f; // 射線長度（腿長）

bool IsGrounded() // 自定義的功能：檢查是否在地板
{
    // 從主角位置，向下發射，長度 rayLength，只偵測 groundLayer
    RaycastHit2D hit = Physics2D.Raycast(transform.position, Vector2.down, rayLength, groundLayer);

    // 如果 hit.collider 不是空的，代表有照到東西
    return hit.collider != null;
}
```

# 5. 修改跳躍條件

回到 `Update()`，把跳躍條件加上限制：

```
// 如果 按下跳躍 並且(AND) 在地板上
if (Input.GetButtonDown("Jump") && IsGrounded())
{
    rb.AddForce(Vector2.up * jumpForce);
}
```

還有，為了看到這條隱形的雷射光，我們可以畫出來除錯：

```
void OnDrawGizmos()
{
    Gizmos.color = Color.red;
    Gizmos.DrawLine(transform.position, transform.position + Vector3.down * rayLength);
}
```

# 6. UNITY 設定步驟 (超重要)

程式寫好了，但還沒設定 Layer，雷射光會穿透地板。

1. 右上角 **Layers** -> **Edit Layers**。
2. 在 User Layer 3 (或其他空格) 輸入 `Ground`。
3. 選取所有地板物件 (Tilemap)，將 Layer 改為 **Ground**。
4. 回到主角腳本，**Ground Layer** 選單中，勾選 **Ground**。

# 總結

現在你的角色跳躍功能應該很完美了：

- 按空白鍵跳躍。
- 空中按空白鍵無效。
- 透過 Raycast 進行科學的物理偵測。
- 理解 LayerMask 的過濾功能。

下一章，我們要讓生硬移動的方塊人，變成有動作的動畫角色！