補充教材

遊戲存檔系統 (SAVE & LOAD)

HORAZON

手機遊戲開發

本章目標

1. 理解 **PlayerPrefs** 的運作原理
2. 實作 **儲存 (Save)**：分數、音量設定
3. 實作 **讀取 (Load)**：恢復上次遊玩的狀態
4. 實作：最高分 (High Score) 系統

為什麼需要存檔？

試著想像一款 RPG 遊戲，每次關掉重開都要從 Level 1 開始打...
玩家會崩潰！

手機遊戲常見的存檔需求：

- **進度**：關卡解鎖狀態 (Level 1, Level 2...)
- **貨幣**：金幣數量、鑽石
- **設定**：音樂大小聲、語言
- **成就**：最高分數

UNITY 的神器：PLAYERPREFS

Unity 內建了一個非常簡單的存檔工具，叫做 `PlayerPrefs` (Player Preferences)。
它適合儲存**簡單的資料**。

支援的三種型別

- `SetInt(key, value)`：存整數 (分數、關卡)
- `SetFloat(key, value)`：存浮點數 (音量、靈敏度)
- `SetString(key, value)`：存文字 (玩家名稱)

儲存資料 (SAVE)

把它想像成一個**置物櫃**，每個格子都有一個**鑰匙 (Key)**。

```
// 存分數（整數）
PlayerPrefs.SetInt("Score", 100);

// 存音量（小數）
PlayerPrefs.SetFloat("MusicVol", 0.8f);

// 存名字（字串）
PlayerPrefs.SetString("PlayerName", "Horazon");

// 重要！存完記得呼叫 Save，確保寫入硬碟
PlayerPrefs.Save();
```

讀取資料 (LOAD)

讀取時，要告訴它如果**找不到**該怎麼辦 (預設值)。

```
// 讀取分數，如果沒存過，預設給 0
int currentScore = PlayerPrefs.GetInt("Score", 0);

// 讀取音量，預設 1.0 (最大聲)
float vol = PlayerPrefs.GetFloat("MusicVol", 1.0f);

// 讀取名字
string name = PlayerPrefs.GetString("PlayerName", "Unknown Hero");
```

實作：最高分系統 (HIGH SCORE)

我們希望遊戲結束時，如果分數創新高，就存起來。

```
public class GameManager : MonoBehaviour
{
    public int currentScore;

    public void GameOver()
    {
        // 1. 讀取舊的最高分
        int highest = PlayerPrefs.GetInt("HighScore", 0);

        // 2. 比較
        if (currentScore > highest)
        {
            // 3. 創新高了！存起來！
            PlayerPrefs.SetInt("HighScore", currentScore);
            PlayerPrefs.Save();
            Debug.Log("New High Score Saved!");
        }
    }
}
```

實作：刪除資料 (RESET)

開發時常需要重置進度測試。

```
// 刪除單一資料
PlayerPrefs.DeleteKey("Score");

// 刪除全部資料（慎用！）
PlayerPrefs.DeleteAll();
```

> Tip: 也可以在 Unity 上方選單 `Edit` -> `Clear All PlayerPrefs` 來手動清除。

PLAYERPREFS 的缺點

雖然簡單，但它有缺點：

1. **不安全**：資料是明碼存的，玩家很容易修改 (作弊)。
2. **型別少**：只能存 int, float, string。
3. **效能**：資料量大時 (例如存幾百個道具) 會變慢。

進階解法：
使用 `JsonUtility` 將複雜物件轉成文字 (JSON)，再寫入檔案或加密儲存。

進階：儲存複雜資料 (JSON)

如果想存背包裡的所有道具 (List)，PlayerPrefs 做不到。

我們需要把物件轉成 **JSON 字串**。

```
[System.Serializable]
public class SaveData
{
    public int gold;
    public List<string> items;
}

// 轉成文字
SaveData data = new SaveData();
string json = JsonUtility.ToJson(data);
PlayerPrefs.SetString("SaveData", json);

// 讀回來
SaveData loaded = JsonUtility.FromJson<SaveData>(json);
```

這就是商業遊戲常用的存檔方式！

總結

1. 使用 `PlayerPrefs` 快速實作存檔功能。
2. 記得 `Set` 完要呼叫 `Save()`。
3. `Get` 時要設定好**預設值**。
4. 適合存**設定**與**簡單數據**，不適合存複雜的大型資料。

未來展望：從 DEMO 到上架

做完這個遊戲後，下一步呢？

1. 上架商店 (STORE RELEASE)

- **Google Play**: 開發者帳號 (25 美金/一次性)。
- **App Store**: 開發者帳號 (99 美金/每年)。
- 這是檢驗自己游戲最快的方式，面對真實玩家的評價！

2. 商業化 (MONETIZATION)

- **廣告 (Ads)**: AdMob, Unity Ads。
- **內購 (IAP)**: 買鑽石、去廣告。

3. 持續優化 (OPTIMIZATION)

- 減少過場讀取時間
- 降低耗電量與發熱

下一章：學期總結與專案發布