CHAPTER 3

UNITY MCP

HORAZON

ANTIGRAVITY

什麼是 MCP?

Model Context Protocol (MCP)

- 這是一個標準協定，讓 AI 模型能與外部工具、資源做溝通。
- 以前 AI 只能透過「文字」寫程式，無法「看到」或「操作」編輯器。
- 透過 MCP，AI 就像有了 **手** 和 **眼**。

MCP-UNITY 伺服器

這是一個專門為 Unity 開發的 MCP 伺服器，賦予 AI 以下能力：

1. **讀取場景 (Inspect)**：
    - 獲取 Hierarchy 結構、GameObject 屬性、Component 資訊。
    - 不用再手動報座標給 AI，它自己看得到！
2. **操作物件 (Manipulate)**：
    - 移動、旋轉、縮放物件。
    - 生成 (Instantiate) 或 刪除 (Delete) 物件。
    - 修改 Inspector 上的數值與材質。
3. **執行測試 (Test)**：
    - 觸發 Unity Test Runner，並讀取測試結果。

1. 設置開發環境 (WORKSPACE)

為了讓 AI 能順利同時處理「程式」與「Unity 場景」，建議的 Workspace 結構如下：

- **建議方式**：在 VS Code 中開啟一個**空資料夾**作為根目錄。
- **目錄結構**：
  - `UnityProject/` (你的完整 Unity 專案)
  - `Notes/` 或 `Docs/` (存放開發文件的資料夾)

> **優點**：AI 可以同時存取專案外的筆記與專案內的程式碼，且不會因為直接開啟 Unity 根目錄而掃描過多暫存檔。

2. 安裝與連接 UNITY MCP

要讓 Antigravity 獲得操作能力，必須在 Unity 中建立「通訊門」：

安裝方式 (二選一)

1. **手動安裝**：將 `mcp-unity` 套件匯入 Unity 專案中。
2. **AI 自動安裝**：直接對 Antigravity 下指令：

   > *"請幫我在目前這個 Unity 專案中安裝 mcp-unity 伺服器，並開啟連接。"*

3. 確認連接狀態

完成安裝並開啟 Unity 後，請依照以下步驟確認：

1. **Unity 端**：確認畫面上顯示 `MCP Server Running on Port XXXX`。
    - 若沒看到面板，請點擊上方選單 **Window > MCP Server**。
    - 點擊面板中的 **Configure Antigravity** 按鈕進行同步。
2. **Antigravity 端**：
    - 在對話框輸入 `/mcp` 或檢查工具列。
    - 若狀態沒更新，請嘗試 **重新啟動 Antigravity** 或重新輸入指令。
    - 若看到 `Unity` 狀態為 **Connected (綠燈)**，代表連線成功！

實戰演練：AI 協助場景建置

你可以直接對 Antigravity 下達連續指令，觀察它如何操作 Unity：

任務 A：生成與佈置

> *"在場景中建立 3 個紅色的方塊，讓它們沿著 X 軸分開排列，並且調整攝影機的位置，確保 3 個方塊都能被拍到。"*

實戰演練：AI 協助程式開發

AI 不僅能寫 C#，還能直接實裝到場景測試效果：

任務 B：動態效果與互動

> *"請讓中間那個方塊會自動上下移動。另外幫我寫一個腳本，讓它在運行時每隔一秒隨機變換一次顏色。寫完直接幫我掛載上去。"*

AI 的極限：並非無所不能

雖然 MCP 賦予了 AI 操作 Unity 的能力，但它仍有其限制：

- **受限於工具集 (MCP Tools)**：
  - AI 只能執行 MCP 伺服器有提供的功能 (如移動、建立物件等)。
  - 對於某些極其細微的編輯器操作或冷門功能，AI 可能無法直接控制。
- **需要手動介入**：
  - 當場景結構極度複雜，或 AI 無法理解特定的專案邏輯時，仍需**手動操作 Unity 編輯器**進行調整或修復。
  - **不要依賴 AI 完成 100% 的工作**，應將其視為「協作夥伴」。

大型專案協作建議 (FULL GAME)

當要從零開始製作一個功能完整的遊戲時，建議採取以下進階模式：

1. **切換計畫模式 (Plan Mode)**：
    - 面對複雜任務，務必使用 **Plan Mode**。
    - 搭配 **Gemini 3 Pro** 等高階模型，以獲得最嚴謹的邏輯推理。
2. **仔細審核「實作計畫書」**：
    - AI 會在動手前生成 `Implementation_Plan.md`。
    - **不要直接點確認**！請務必仔細檢查：
        - 遊戲架構是否合理？物件層級是否正確？
        - 是否有遺漏的組件 (Components) 或資產？
    - 若有問題，請在對話視窗中指正，讓 AI 更新計畫後再執行。

> **心法**：計畫做得越精確，AI 執行任務的錯誤率就越低。

實作驗證：測試與持續溝通

確保 Unity 場景與程式碼同步的關鍵：

1. **即時驗證 (Unity Inspection)：**
   - AI 執行任務後，務必在 Unity 編輯器中點擊 **Play** 測試。
   - 利用 MCP 的 Inspect 能力，讓 AI 再次掃描場景，確認物件狀態。
2. **明確溝通錯誤 (Error Reporting)：**
   - 若 AI 生成的物件位置偏移，直接反饋：「左側方塊過大，請縮小一倍」。
   - 若 C# 報錯，AI 會主動讀取 Console，但也需你的確認。
3. **小步快跑，多次溝通：**
   - 不要一次要求 AI 做「整個轉化專案」。
   - 先做「場景配置」，再做「核心邏輯」，最後做「UI 連結」。

未來展望：自然語言開發

隨著 MCP 與 Antigravity 的進化，遊戲開發將變得更直覺：

- **Natural Language to Game**：用說的做遊戲。
- **Automated QA**：AI 自動玩遊戲並找 Bug。
- **Intelligent Assistant**：隨時在旁邊幫你查 API、修 Code、調參數的超級助手。

結語：與 AI 共舞

雖然 AI 能極大地提升開發效率，但我們**仍然需要學習遊戲引擎與程式相關邏輯**，因為：

- **決策權在你**：AI 只是執行者，你需要足夠的知識來審核它的行為。
- **基礎定生死**：當 AI 遇到瓶頸或無法解決的 Bug 時，你的專業知識是最後的防線。
- **創意與調優**：遊戲的靈魂與細節手感，依然來自人類的感性與經驗。

AI 不是要取代你，而是讓我們能專注在更有創意的事上。