OOP Group Project - 地下城RPG設計與實作

Group 20 :　B123245011 楊鎧榤　B123245007 許珈瑜　B123040015 陳進發

一、物件導向設計與專案概觀

本專題實作了一款三關制的 2D 地下城 RPG，玩家將帶領奇異鳥主角 Kiwi 與隊友，目標是擊敗位於第三層的噴火龍 Boss。專案設計的核心並非單純堆疊遊戲內容，而是透過建構一個符合 Gym 介面的自訂環境，搭配多種 Agent 與敵人 AI，來實踐物件導向程式設計的原則，避免程式淪為冗長的 if-else 判斷式。

在 OOP 概念的應用上，本專案重點如下：

- ➔ **Encapsulation**：
  將角色的生命值管理扣血、補血、死亡判定與行動邏輯嚴格封裝於 Character 類別中；戰鬥流程與關卡狀態則封裝於 DungeonBattleEnv 的 reset()、step() 與 render() 方法內。外部程式像是主迴圈或介面層，僅透過公開介面進行互動，確保內部狀態的安全性與一致性。
- ➔ **Inheritance & Polymorphism**：
  以 Character 為基底類別，衍生出 PlayerCharacter、EnemyCharacter 以及具有特殊行為狂暴、範圍噴火的 FireDragon 子類。技能系統亦採同樣邏輯，Skill 基類定義了介面，單體、範圍、治癒等具體行為則在子類中實作。透過 apply() 的統一呼叫，環境無須知道具體技能類型即可執行對應效果，具體展現多型優勢。
- ➔ 設計模式與結構化思維：
  專案採用 Strategy Pattern 實作敵人 AI 與 Agent 決策，便於動態切換不同邏輯；技能執行流程則應用了 Template Method 概念。類別間大量使用 組合 Composition，例如 DungeonBattleEnv 組合了 TurnManager 來管理回合順序，降低模組間的耦合度Coupling。此外，針對視覺表現，我們引入了物理模擬概念，透過粒子系統與加速度運算優化動畫流暢度，提升整體遊戲質感。

二、實作重點與系統設計

系統架構主要劃分為三個層次：戰鬥核心模型（Model）、操作控制（Control/Agent）以及介面呈現。

1. 戰鬥核心與三層地下城

遊戲世界由地城環境與角色系統構成。地城設計包含教學用的第一層、強度提升的第二層，以及 Boss 駐守的第三層。當呼叫 DungeonBattleEnv.step() 時，TurnManager 會依據角色速度屬性排序出手順序。系統依序處理行動時，若為自動模式則由 Agent 決策，若為敵人則由綁定的 EnemyAIStrategy 產生行動。此架構延續了強化學習中「Environment+ Agent」的互動思維，便於未來擴充訓練模型。

2. 操作模式與 Agent 切換

為了展示架構彈性，我們設計了手動與自動兩種模式，並實作三種繼承自 BaseAgent 的策略：

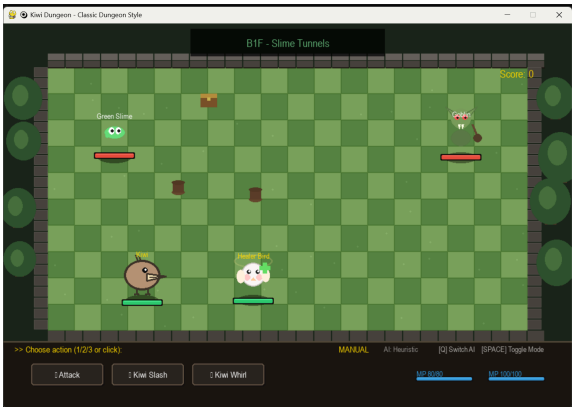
- ➔ **Random Agent**：隨機選取合法行動，作為效能比較的基線。
- ➔ **Heuristic Agent**：規則型 AI，具備隊友殘血優先治療、Boss 關卡集火首領、一般關卡優先擊殺殘血敵人等...戰術邏輯。
- ➔ **Q-Learning Agent**：將樓層、HP 分段、敵人存活數等特徵離散化為狀態，並簡化 Q-table 進行決策，嘗試在規則之外尋找最佳解。
  由於所有 Agent 皆繼承自同一基底類別，主程式僅需替換物件實例，即可無縫切換遊戲的操作風格。

3. Pygame 地下城介面與視覺優化

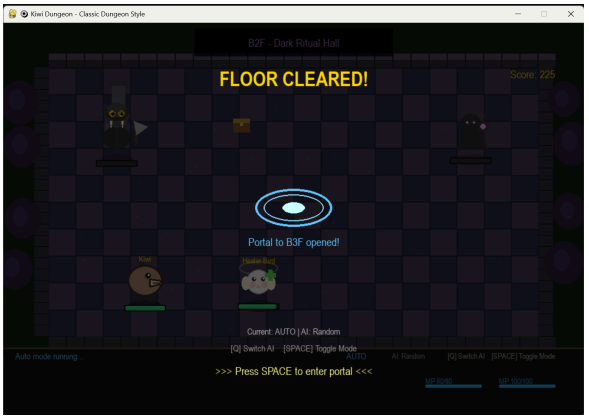
視覺層面使用 Pygame 實作。開發初期我們面臨 2D 介面過於僵硬、特效實作困難的問題，為了突破此瓶頸，我們引入了 AI 輔助開發模式進行大幅優化：

- ➔ 介面與美術重構：初始版本僅使用簡單幾何圖形(圓球)代表角色，表現力嚴重不足。後續我們導入了角色立繪，並重寫了 UI 邏輯，設計出包含滑鼠互動與按鈕回饋的動態介面，以及依據樓層變化的多層次背景與圖塊渲染的地圖元素，大幅提升了視覺豐富度。
- ➔ **AI** 驅動的特效程式碼生成：為了在有限時間內實現高品質的戰鬥特效，我們運用了**Antigravity** 來協助重構 Pygame 的渲染層。透過 AI 的協助，我們快速生成了複雜的effect 類別繼承體系，並成功將粒子系統、加速度運算與平滑插值等數學概念應用於程式碼中。這讓我們能夠以極高的效率產出斬擊、旋風、火焰噴射等具有動態軌跡與淡出動畫的技能特效。這不僅解決了手寫特效邏輯過於繁瑣的問題，更顯著提升了整體戰鬥的遊戲感與打擊回饋。

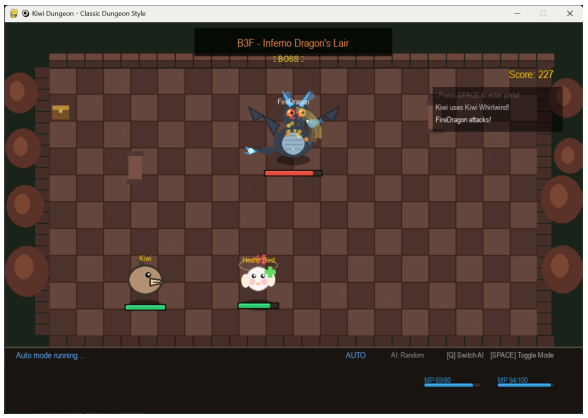
❖ 遊戲畫面展示



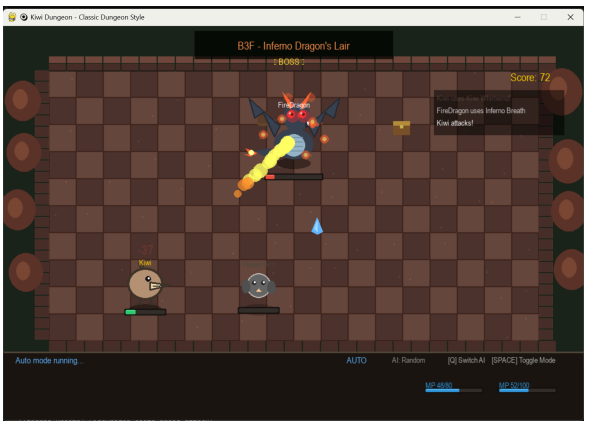
-> B1F



->B2F (clear and time gate)



-> B3F(boss level1)



->B3F(boss level2 crazyyy mode)

三、學習收穫與反思

1.物件導向設計對專案結構的影響

在實作過程中，先設計架構與邊寫邊改的差異體現得淋漓盡致。若非初期即定義好 BaseAgent、Skill 與 EnemyAIStrategy 等抽象層，後期在擴充 Q-Learning 或 Boss 特殊行為時，勢必面臨牽一髮動全身的維護災難。良好的封裝與多型設計，讓我們能專注於單一功能的擴充，而不必擔心破壞既有系統。

2.跨領域技術的整合

本次專案不僅是 OOP 的實踐，更是將抽象概念轉化為可執行程式的過程。我們成功將課堂上的 UML 概念落地，並整合了不同課程的知識像是運用 antigravity 協助我們於遊戲特效的改進與優化。這種將演算法、軟體架構與圖學數學結合的經驗，讓我們更深刻理解如何構建一個完整且具備良好使用者體驗的軟體系統。

3.AI 工具的協作心得

在撰寫程式碼的過程中，我們將 AI 工具整合至工作流，應用於架構腦力激盪(ex. part3工作分配)、template程式碼生成以及演算法優化。AI 的介入顯著加速了我們收斂實作方向，讓我們能將時間集中在核心邏輯的設計上，尤其在最後應用antigravity，大幅度地優化我們的遊戲介面，還有遊戲角色的協助繪製，上述的種種AI協作令我們最終得到更佳的遊戲體驗。值得反思的是，AI 工具的使用實際上是對我們 OOP 架構的一種壓力測試，AI 生成的程式碼將難以整合；簡言之，正因為我們確立了清晰的chrarcter繼承與Env 介面，AI 產出的特效與邏輯才能順利地整合進入系統中。