PvZ Duel：植物大戰殭屍雙人對戰版

Development of a Two-Player Battle Mode for Plants vs. Zombies

連哲寬[1], 鄭恆安[2], 張棋凱[3]

111511187, 111511198, 111511180

Department of Eletronics and Eletrical Engineering

National Yang Ming Chiao Tung University

Hsinchu, Taiwan

January 20, 2025

Abstract

本專案「PvZ Duel」是一款基於經典遊戲《植物大戰殭屍》改編的雙人對戰遊戲。我們使用 Pygame 框架開發，實現了經典的單人模式以及創新的雙人對戰模式。該專案不僅豐富了原有遊戲的玩法，也展示了物件導向程式設計的實踐應用。本文詳細描述了遊戲的設計理念、實現方法以及開發過程中的技術要點。

Chapter 1 Introduction

《植物大戰殭屍》作爲一款經典的單人塔防遊戲，在遊戲史上具有重要地位。然而，原版遊戲缺乏玩家之間的互動性，限制了其社交性和競技性。本專案通過加入雙人對戰模式來提升遊戲體驗，同時作爲物件導向程式設計的實踐平台。

本專案Github Repo：`https://github.com/zachlian/aoop-proj-g3`

Chapter 2 Motivation

本專案的主要動機包含以下幾點：

- 擴展經典遊戲的玩法，增加玩家互動性
- 實踐物件導向程式設計概念
- 學習團隊協作開發流程
- 探索遊戲平衡性設計

Chapter 3 Methods

本遊戲包含多個檔案，分別控制遊戲的不同部分，例如：使用者介面、植物、殭屍、陽光、遊戲模式、網格、卡片、發射物等等。以下將簡要介紹各主要檔案的功能和作用。

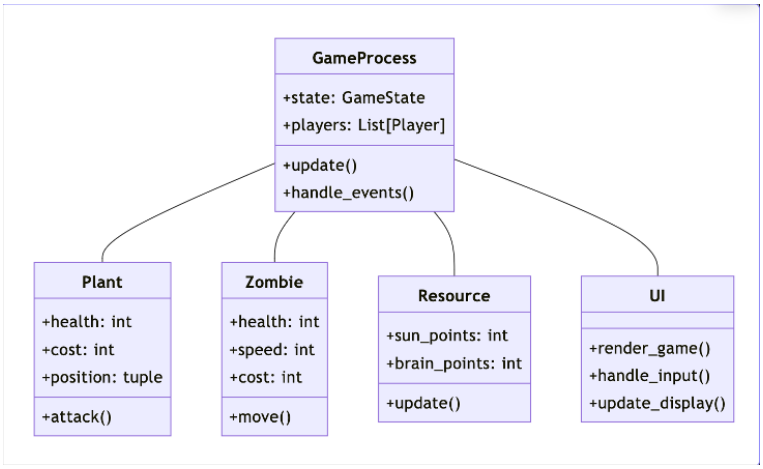

圖 3.1: Class Architecture

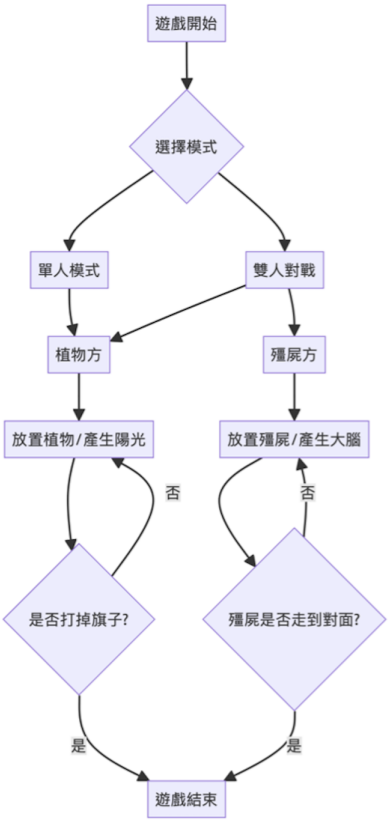

圖 3.2: Flow Chart of the Game

3.1 遊戲主程式 (single_player_game.py, multiplayer_game.py)

‘single_player_game.py‘ 和 ‘multiplayer_game.py‘ 分別爲單人及多人遊戲主程式，繼承自 ‘base_game.py‘，負責遊戲主循環、事件處理、狀態更新和畫面渲染。

- 初始化遊戲物件: ‘_setup_game_objects()‘ 方法初始化遊戲所需的各種物件，包括網格、植物管理器、卡片管理器、陽光管理器、殭屍管理器和特效管理器。
- 遊戲主循環: ‘run()‘ 方法包含遊戲主循環，負責處理事件、更新遊戲狀態、渲染畫面並維持穩定的幀率。
- 事件處理: ‘_handle_events()‘ 和 ‘_process_event()‘ 方法處理各種遊戲事件，例如：滑鼠點擊、鍵盤輸入、自訂事件等。
- 滑鼠點擊處理: ‘_handle_mouse_click()‘ 方法處理滑鼠點擊事件，包括收集陽光、選擇卡片和放置植物。
- 遊戲狀態更新: ‘_update()‘ 方法更新遊戲狀態，包括植物、卡片、陽光、殭屍、發射物和特效的狀態，並檢查遊戲是否結束。
- 畫面渲染: ‘_render()‘ 方法負責渲染遊戲畫面，包括背景、網格、植物、卡片、陽光、殭屍、發射物和特效。
- 遊戲結束處理: ‘_check_game_over()‘ 和 ‘_show_game_over()‘ 方法檢查遊戲是否結束，並顯示遊戲結束畫面。
- 多人模式: ‘multiplayer_game.py‘ 使用 ‘MultiplayerGrid‘、‘BrainManager‘ 和 ‘MultiplayerZombieManager‘，並新增了殭屍卡片管理器 (‘ZombieCardManager‘)，實現多人遊玩功能。

3.2 遊戲基礎類 (base_game.py)

‘base_game.py‘ 檔案定義了 ‘BaseGame‘ 類別，作爲單人及多人遊戲的基礎類別。

- 初始化**:** ‘__init__()‘ 方法初始化 Pygame 並設定遊戲視窗。
- 抽象方法**:** 定義了 ‘_setup_game_objects()‘、‘_handle_events()‘、‘_update()‘ 和 ‘_render()‘ 等抽象方法，由子類別實現具體邏輯。
- 遊戲主循環**:** ‘run()‘ 方法提供基本的遊戲主循環框架。

3.3 植物模型 (plant.py)

‘plant.py‘ 檔案定義了植物相關的類別和資料。

- 植物類型 **(PlantType):** 使用 ‘Enum‘ 定義了不同種類的植物，例如：向日葵 (SUNFLOWER)、豌豆射手 (PEASHOOTER)、堅果牆 (WALLNUT) 和窩瓜 (SQUASH)。
- 植物屬性 **(PlantStats):** 使用 ‘dataclass‘ 定義了植物的屬性，例如：生命值、花費、攻擊力、攻擊速度等。
- 植物屬性配置 **(PLANT_STATS):** 字典，儲存了每種植物的具體屬性值。
- 植物基類 **(Plant):** 定義了植物的通用行爲，例如：更新狀態、受到傷害、繪製自身等。
- 植物子類別**:** ‘Sunflower‘、‘Peashooter‘、‘Wallnut‘ 和 ‘Squash‘ 繼承自 ‘Plant‘，並實現了各自的特殊行爲，例如：產生陽光、發射豌豆、阻擋殭屍等。‘Squash‘更可以偵測並移動攻擊殭屍。

3.4 殭屍模型 (zombie.py)

‘zombie.py‘ 檔案定義了殭屍相關的類別和資料。

- 殭屍類型 **(ZombieType):** 使用 ‘Enum‘ 定義了不同種類的殭屍，例如：普通殭屍 (NORMAL)、路障殭屍 (CONE_HEAD)、水桶殭屍 (BUCKET_HEAD) 和墓碑殭屍 (TOMBSTONE)。

- 殭屍屬性 **(ZombieStats):** 使用 ‘dataclass‘ 定義了殭屍的屬性，例如：名稱、生命值、傷害、速度、攻擊速度等。
- 殭屍屬性配置 **(ZOMBIE_STATS):** 字典，儲存了每種殭屍的具體屬性值。
- 殭屍基類 **(Zombie):** 定義了殭屍的通用行爲，例如：更新狀態、移動、攻擊、受到傷害、繪製自身等。

3.5 陽光模型 (sun.py)

‘sun.py‘ 檔案定義了陽光類別 ‘Sun‘。

- 初始化**:** ‘__init__()‘ 方法初始化陽光的位置、目標高度、陽光值、收集狀態、消失時間等屬性，並載入陽光圖片。
- 更新狀態**:** ‘update()‘ 方法更新陽光的狀態，包括自然掉落、被收集後的移動動畫，以及檢查是否應該消失。
- 收集**:** ‘collect()‘ 方法將陽光標記爲已收集。
- 繪製**:** ‘draw()‘ 方法將陽光繪製到螢幕上。
- 點擊檢測**:** ‘is_clicked()‘ 方法檢測陽光是否被滑鼠點擊。

3.6 發射物模型 (projectiles.py)

‘projectiles.py‘ 檔案定義了豌豆射手發射的豌豆類別 ‘Pea‘。

- 初始化**:** ‘__init__()‘ 方法初始化豌豆的位置、所在行、傷害、速度、半徑和活動狀態等屬性。
- 更新位置**:** ‘update()‘ 方法更新豌豆的位置，並檢測是否超出螢幕範圍或擊中旗幟。

- 繪製**:** ‘draw()‘ 方法將豌豆繪製到螢幕上。
- 獲取碰撞矩形**:** ‘get_rect()‘ 方法獲取豌豆的碰撞矩形。

3.7 網格系統 (grid.py, multiplayer_grid.py)

‘grid.py‘ 和 ‘multiplayer_grid.py‘ 檔案分別定義了單人及多人遊戲的網格系統。

- 初始化**:** ‘__init__()‘ 方法計算網格的起始位置。
- 繪製網格**:** ‘draw()‘ 方法繪製網格線。
- 滑鼠座標轉換**:** ‘get_cell_from_pos()‘ 方法根據滑鼠位置獲取網格座標。
- 區域判斷**:** ‘is_in_plant_zone()‘ 和 ‘is_in_zombie_zone()‘ (僅 ‘multiplayer_grid.py‘) 方法判斷座標是否在植物區域或殭屍區域內。
- 高亮顯示**:** ‘highlight_selected_cell()‘ (僅 ‘multiplayer_grid.py‘) 方法高亮顯示選中的格子。
- 鍵盤事件處理**:** ‘handle_keyboard_event()‘ (僅 ‘multiplayer_grid.py‘) 方法處理鍵盤事件，用於控制殭屍方格子的選擇。

3.8 植物管理器 (plant_manager.py)

‘plant_manager.py‘ 檔案定義了 ‘PlantManager‘ 類別，負責管理植物。

- 添加植物**:** ‘add_plant()‘ 方法在指定位置添加指定類型的植物，並消耗相應的陽光。
- 放置檢查**:** ‘can_place_plant()‘ 方法檢查指定位置是否可以放置植物。
- 更新植物**:** ‘update()‘ 方法更新所有植物的狀態。

- 繪製植物: ‘draw()‘ 方法繪製所有植物。
- 移除植物: ‘remove_plant()‘ 方法移除指定位置的植物。

3.9 殭屍管理器 (zombie_manager.py, multiplayer_zombie_manager.p

‘zombie_manager.py‘ 和 ‘multiplayer_zombie_manager.py‘ 檔案分別定義了單人及多人遊戲的殭屍管理器。

- 生成殭屍: ‘_spawn_zombie()‘ 方法在隨機行生成一個隨機類型的殭屍。
- 更新殭屍: ‘update()‘ 方法更新所有殭屍的狀態，包括移動和攻擊。
- 碰撞檢測: ‘check_collisions()‘ 方法檢查殭屍與植物的碰撞，並觸發攻擊事件。
- 繪製殭屍: ‘draw()‘ 方法繪製所有殭屍。
- 波次控制: ‘start_new_wave()‘ 方法開始新的一波殭屍進攻。
- 多人模式: ‘multiplayer_zombie_manager.py‘ 新增了旗幟生命值控制，並覆寫了部分方法以適應多人遊戲模式。

3.10 陽光管理器 (sun_manager.py)

‘sun_manager.py‘ 檔案定義了 ‘SunManager‘ 類別，負責管理陽光。

- 初始化: ‘__init__()‘ 方法初始化陽光數量、自然生成間隔等屬性，並載入陽光圖示。
- 更新陽光: ‘update()‘ 方法更新陽光系統，包括自然生成陽光和更新所有陽光的狀態。

- 生成陽光: ‘_spawn_sun()‘ 方法自然生成一個陽光；‘add_sun_from_sunflower()‘ 方法從向日葵生成陽光。
- 點擊處理: ‘handle_click()‘ 方法處理陽光的點擊事件，收集被點擊的陽光。
- 消耗陽光: ‘spend_sun()‘ 方法消耗指定數量的陽光。
- 增加陽光 : ‘add_sun()‘ 方法增加指定數量的陽光。
- 繪製陽光: ‘draw()‘ 方法繪製所有陽光和陽光計數器。

3.11 卡片管理器 (card_manager.py)

‘card_manager.py‘ 檔案定義了 ‘CardManager‘ 類別，負責管理植物卡片。

- 初始化卡片: ‘_init_cards()‘ 方法初始化所有植物卡片。
- 點擊處理: ‘handle_click()‘ 方法處理卡片的點擊事件，選擇或取消選擇卡片。
- 使用卡片: ‘use_card()‘ 方法使用選中的卡片，並設置卡片冷卻時間。
- 更新卡片: ‘update()‘ 方法更新所有卡片的冷卻狀態。
- 繪製卡片: ‘draw()‘ 方法繪製所有卡片，並根據陽光數量和冷卻狀態顯示不同的效果。

3.12 殭屍卡片管理器 (zombie_card_manager.py)

‘zombie_card_manager.py‘ 檔案定義了 ‘ZombieCardManager‘ 類別，負責管理殭屍卡片。

- 初始化卡片: ‘_setup_cards()‘ 方法初始化所有殭屍卡片。

- 按鍵處理: ‘handle_key()‘ 方法處理按鍵選擇卡片事件。
- 繪製卡片: ‘draw()‘ 方法繪製所有卡片，並根據花費和冷卻狀態顯示不同的效果。

3.13 殭屍方資源管理器 (brain_manager.py)

‘brain_manager.py‘ 檔案定義了 ‘BrainManager‘ 類別，負責管理殭屍方的大腦資源。

- 更新大腦: ‘update()‘ 方法更新大腦數量。
- 花費檢查: ‘can_afford()‘ 方法檢查是否有足夠的大腦。
- 消耗大腦: ‘spend_brain()‘ 方法消耗指定數量的大腦。
- 繪製: ‘draw()‘ 方法繪製大腦計數器。

3.14 特效管理器 (effect_manager.py)

‘effect_manager.py‘ 檔案定義了 ‘EffectManager‘ 類別，負責管理遊戲中的特效，目前只有顯示傷害數字。

- 添加傷害指示器: ‘add_damage_indicator()‘ 方法添加一個傷害數字特效。
- 更新特效: ‘update()‘ 方法更新所有特效的狀態。
- 繪製特效: ‘draw()‘ 方法繪製所有特效。

3.15 遊戲模式 (game_mode.py)

‘game_mode.py‘ 檔案使用 ‘Enum‘ 定義了遊戲模式：單人模式 (SINGLE_PLAYER) 和多人模式 (MULTI_PLAYER)。

3.16 使用者介面 (base_screen.py, game_menu.py, game_over.py)

- **base_screen.py:** 定義了 'BaseScreen' 類別，作爲所有 UI 畫面的基礎類別，提供繪製按鈕和遮罩的功能。
- **game_menu.py:** 定義了 'GameMenu' 類別，負責顯示遊戲選單，讓玩家選擇單人模式或多人模式。
- **game_over.py:** 定義了 'GameOverScreen' 類別，負責顯示遊戲結束畫面，並顯示勝利者。

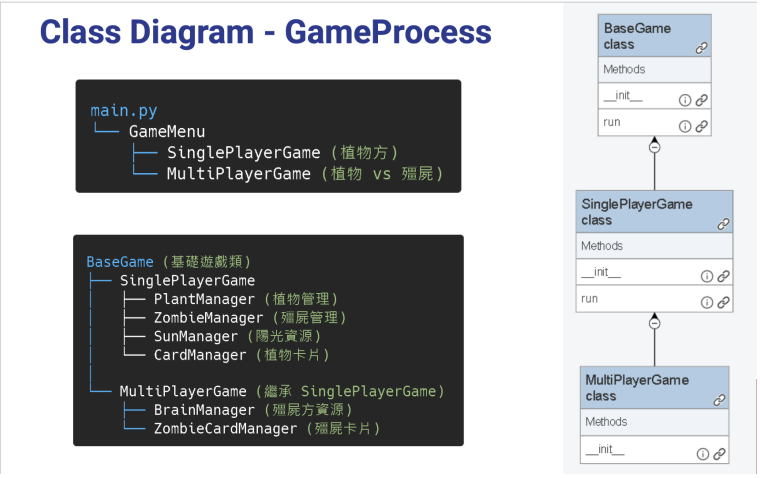

圖 3.3: Class Diagram of GameProcess

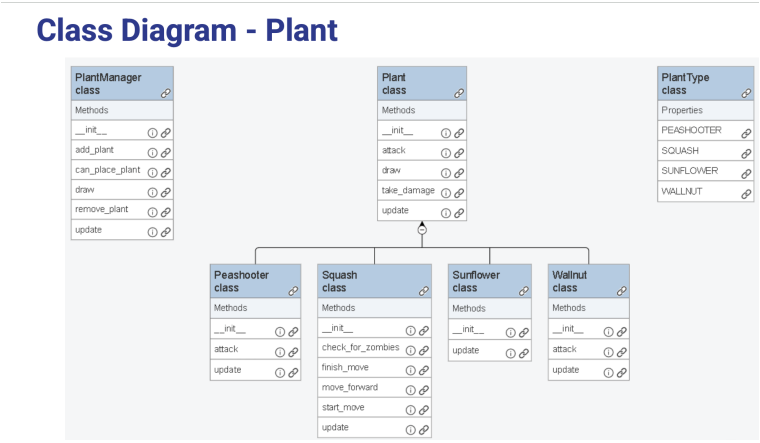

圖 3.4: Class Diagram of Plant

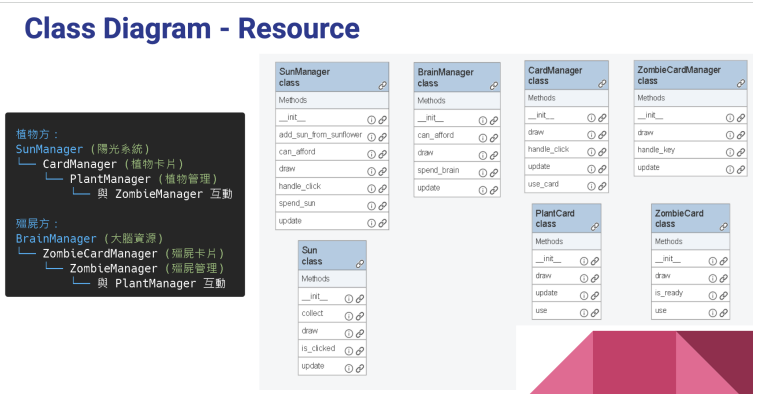

圖 3.5: Class Diagram of Resources

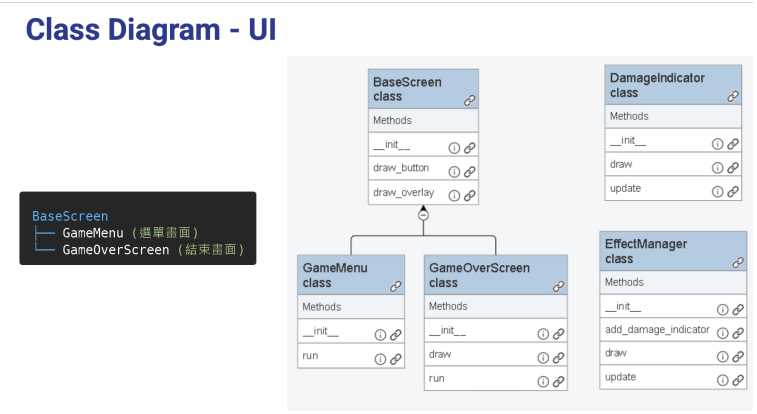

圖 3.6: Class Diagram of UI

Chapter 4 Implementation

4.1 Game Modes

4.1.1 Single Player Mode

單人模式保留了原版遊戲的核心玩法。玩家通過以下操作進行遊戲：

- 使用滑鼠點擊左上角的植物卡片
- 在適當位置放置選中的植物

- 收集畫面中出現的陽光資源
- 策略性地防禦來襲的殭屍

4.1.2 Two-Player Battle Mode

雙人對戰模式是本專案的創新重點，具有以下特色：

- 玩家1控制植物方，沿用原版操作方式
- 玩家2控制殭屍方，使用鍵盤 W/A/S/D 選擇位置
- 殭屍方使用數字鍵 1/2/3/4 放置不同類型的殭屍
- 雙方各自管理獨立的資源系統

Chapter 5 Results and Discussion

5.1 Problem Solutions

5.1.1 Multiplayer Design

成功實現了雙人對戰功能，使玩家能夠選擇控制植物或殭屍陣營，大幅提升了遊戲的互動性與趣味性。

5.1.2 Resource Management

爲植物與殭屍設計了獨立的資源系統（陽光與腦點），並實現了合理的資源生成與消耗機制。

5.1.3 Game Balance

通過反覆測試與調整，實現了植物方與殭屍方的戰力平衡，確保遊戲的公平性和競技性。

5.2 Technical Achievements

在開發過程中，我們掌握並應用了以下技術：

- 物件導向程式設計：運用封裝、繼承等概念構建遊戲架構
- 版本控制：使用 Git 進行團隊協作
- 遊戲開發框架：熟練運用 Pygame 實現遊戲功能
- **AI** 輔助開發：結合 Copilot 提升開發效率

Chapter 6 Future Work

未來的開發計劃包括：

- 遊戲內容擴充：
  - 新增更多植物和殭屍類型
  - 實現特殊技能和道具系統
  - 設計更多互動機制
- 功能改進：
  - 實現網路對戰功能與排行榜功能
  - 開發地圖編輯器
  - 開發 AI 對手系統

Chapter 7 Conclusion

本專案成功將經典的《植物大戰殭屍》改編爲具有雙人對戰功能的新版本，不僅豐富了遊戲玩法，也達成了程式設計學習的目標。通過團隊協作，我們掌握了物件導向程式設計、版本控制等重要技能，爲未來的開發工作奠定了良好基礎。

Bibliography

[1] PopCap Games, "Plants vs. Zombies Official Website," 2009. [Online]. Available: https://www.ea.com/games/plants-vs-zombies

[2] Pygame Community, "Pygame Documentation," 2023. [Online]. Available: https://www.pygame.org/docs/

[3] Z. Lian, H. Zheng, and Q. Zhang, "PvZ Duel Project Repository," 2023. [Online]. Available: https://github.com/zachlian/aoop-proj-g3

[4] Python Software Foundation, "Python Documentation," 2023. [Online]. Available: https://docs.python.org/3/

[5] Open Source Initiative, "The MIT License," [Online]. Available: https://opensource.org/licenses/MIT