

國立臺北科技大學
2024 資工系物件導向程式設計實習
期末報告

命令與征服：紅色警戒



第 13 組

盧威任 杜坤翰

目錄

一、 專案簡介	1
1. 遊戲簡介	1
二、 遊戲介紹	2
1. 遊戲規則	2
2. 遊戲畫面	3
三、 程式設計	5
1. 程式架構	5
2. 程式技術	9
四、 結語	12
1. 問題及解決方法.....	12
2. 貢獻比例	12
3. 自我檢核表.....	13
4. 收穫	13
5. 心得、感想	13
6. 對於本課程的建議	14

一、 專案簡介

1. 遊戲簡介

即時戰略遊戲是一款考驗玩家策略能力與戰略思維能力的遊戲，與電腦 AI 或人類玩家對戰，採集資源、建造建築、升級科技、出兵征討，目標將敵人全數殲滅。而其中"命令與征服：紅色警戒"正是其中最有名的，玩家必須要思考在有限的資源情況下對金錢、電力的最有效的配置，戰略上：蓋礦場、電廠、兵營或屯兵；戰術上：守家、殲敵主力部隊、或偷敵方老家。由於此款遊戲的規則之簡單、玩法之自由，每場遊戲遇到的戰場情況均不相同，所以玩家必須快速應變、險中求生、生中求勝，創造出精彩豐富的遊戲體驗。

2. 組別分工

本次專案是盧威任與杜坤翰合作開發，大部份的功能杜坤翰會先開發一個可運作的版本，盧威任會負責完善剩下的功能並根據 OOP 原則重新架構。內容上來說，盧威任負責報告撰寫、規劃程式架構、主要開發：單位尋路與戰鬥、地圖讀取與繪製、UI、完善遊戲機制；杜坤翰主要負責開發：教學關卡、AI 策略、遊戲內 UI 功能實踐、動畫、建築物功能。

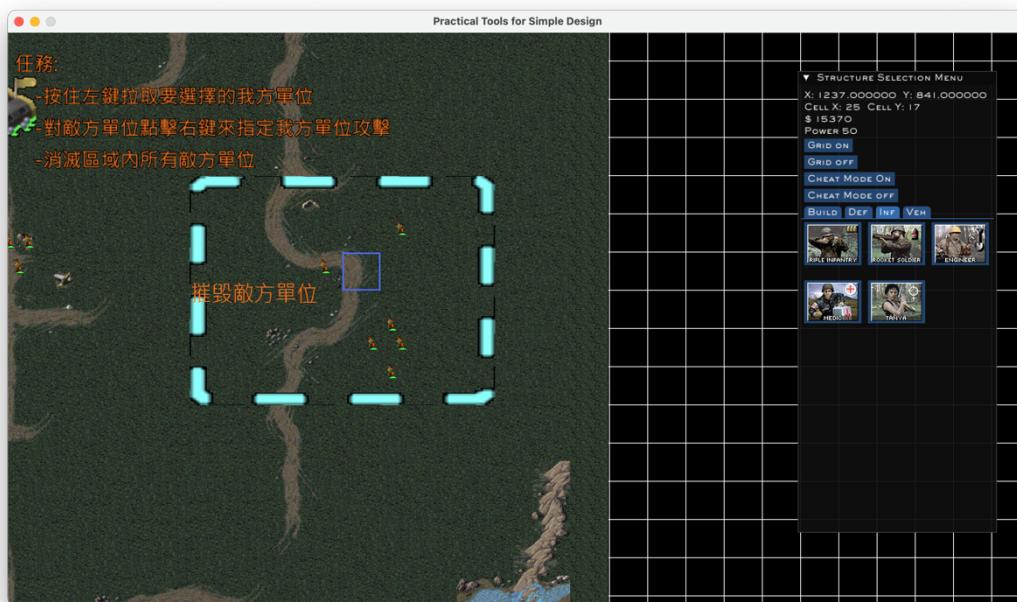
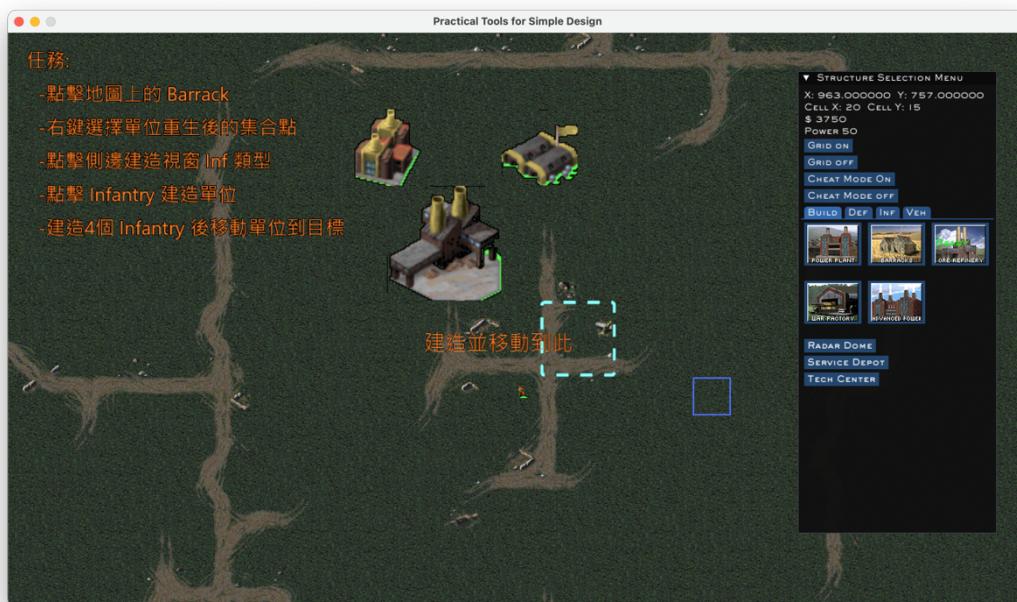
二、 遊戲介紹

1. 遊戲規則

本遊戲的宗旨在於摧毀敵方的所有建築，一旦所有建築被毀則立刻判定戰敗。除此之外，玩家主要有兩項資源：電力(Electric Power)與金錢(Currency)，電力可以透過蓋 Power Plant 或 Advance Power Plant 增加，但建設新的建築（如兵營）則會消耗。礦場(Ore Refinery)會定期增加金錢，建設新的建築、或生產步兵(Infantry)會消耗金錢。步兵可以透過兵營(Barracks)生產，並且他可以去攻擊敵方建築或敵方步兵。

密技的部分，請使用 menu 上的 cheat mode，會增加 50000 金錢、50000 電力、立即建造、無敵士兵等。

2. 遊戲畫面

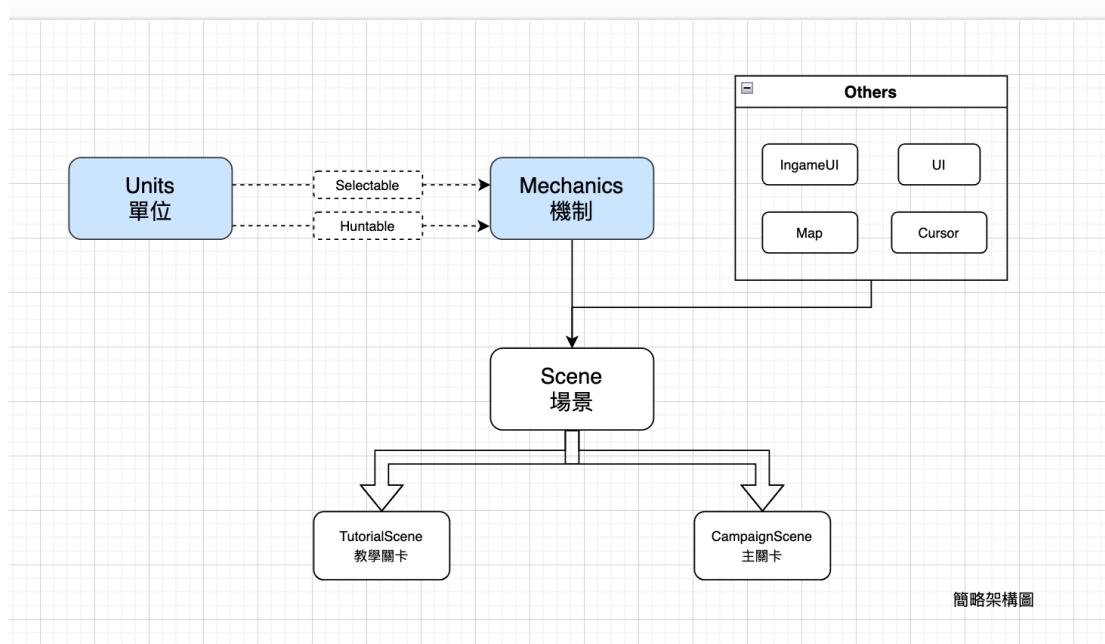




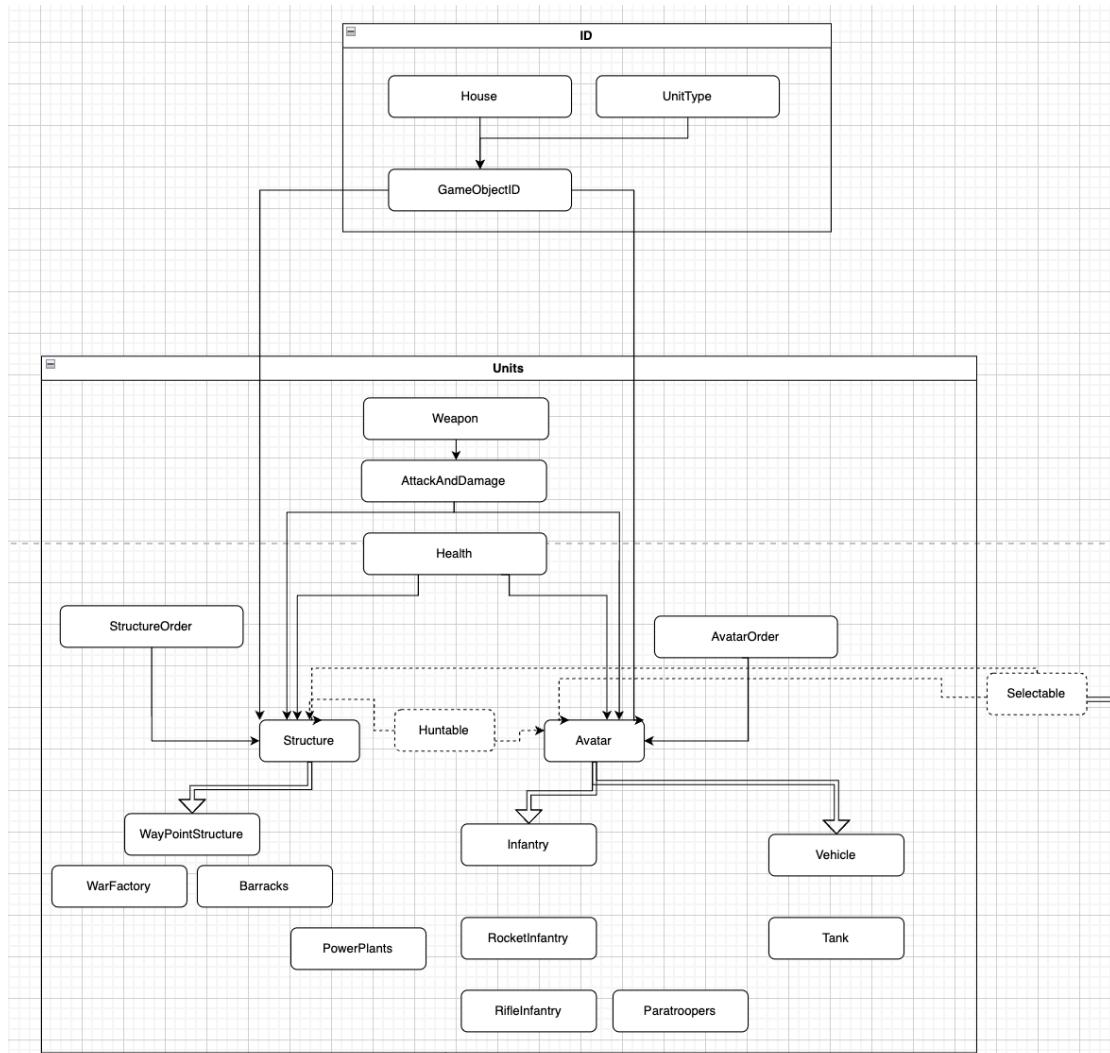
三、 程式設計

1. 程式架構

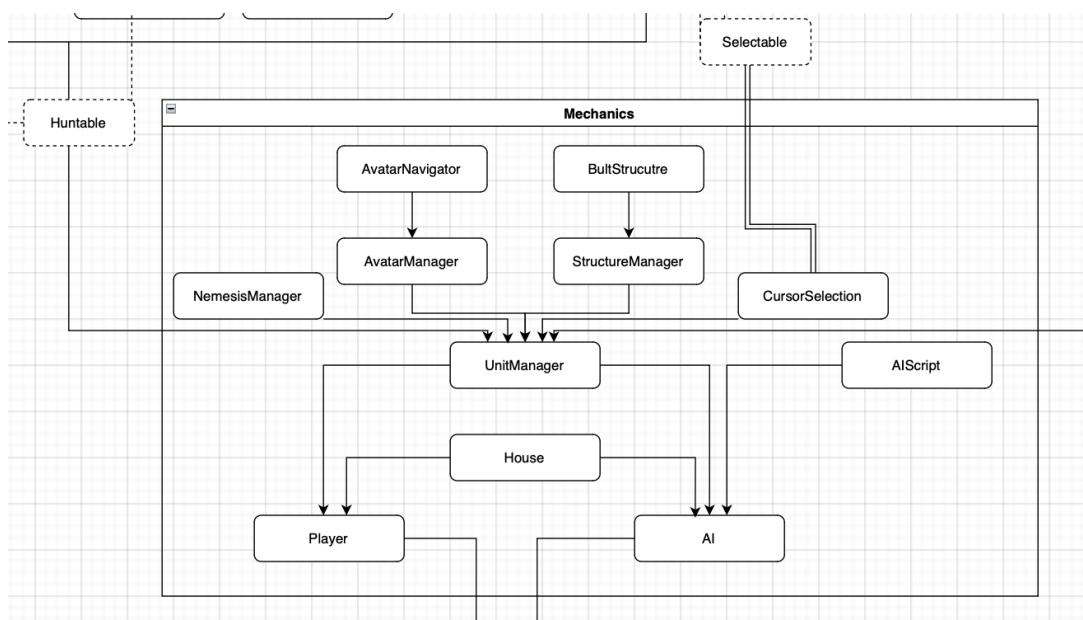
本專案絕大部分的功能均使用組合、多形與介面實作，只有在少數的類別才使用繼承。整個遊戲主要分成三大核心，分別是：機制(mechanics)、單位(units)、場景(Scene)，機制主要負責管理單位(由 UnitManager 負責管理)，發布攻擊命令、移動、改變狀態等，單位則是涵蓋遊戲內步兵(Infantry)與建築物(Structure)的，除此之外場景(Scene)還直接調用、初始化、執行其他的一些輔助功能，例如：UI、遊戲內的 UI(IngameUI)、滑鼠、相機等。

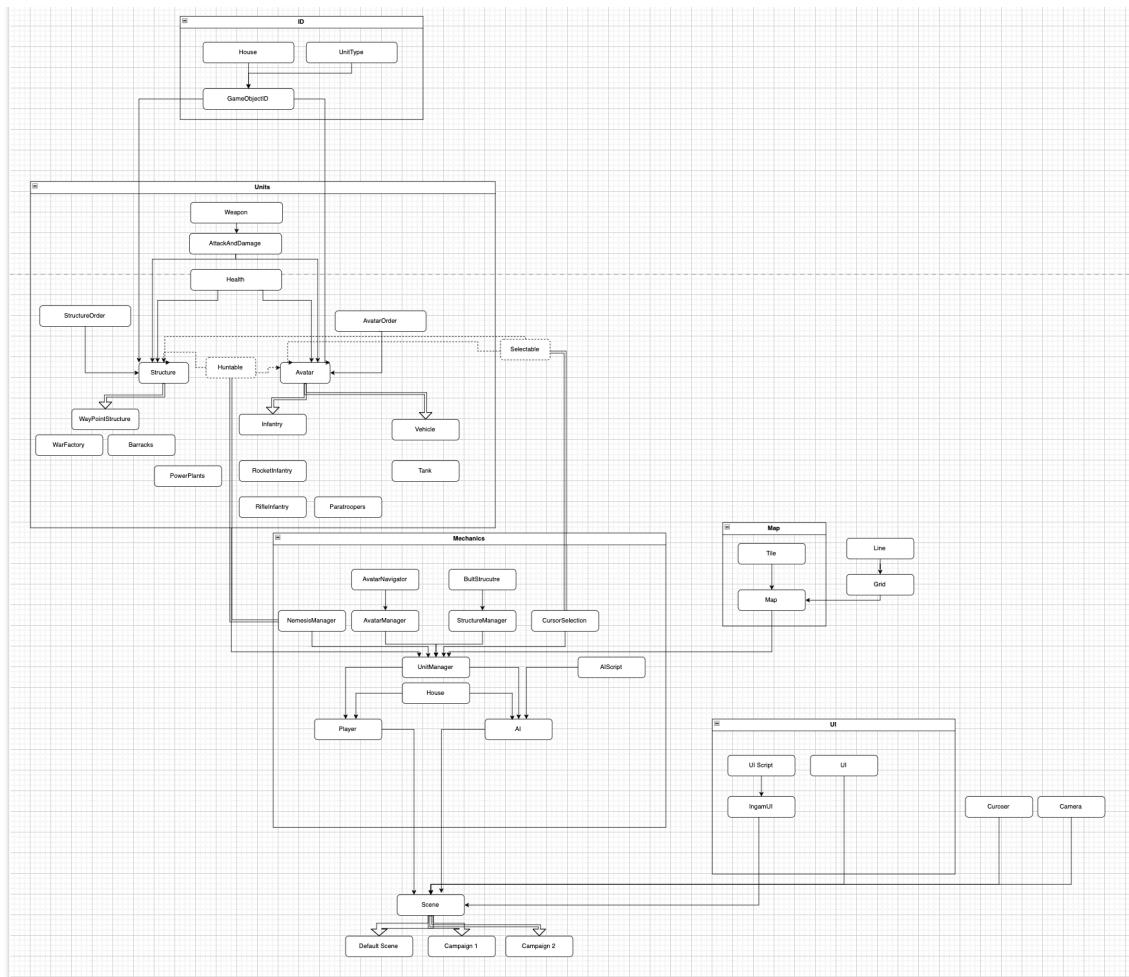


單位(Unit)核心如下，單位是指玩家可操作的單位(Avatar)與建築物(Structure)，另外包含 ID(GameObjectID)、攻擊與受傷系統(AttackAndDamage)與血量系統(Health)。另外由於各單位需要與其他的類別(例如：地圖)以及彼此之間溝通，因此我們設計一個管理者(Manager)來管理所有的單位，並使用 Huntable 與 Selectable 聯絡。



機制(Mechanics)核心如下，主要是有一個單位管理者(Unit Manager)，並包含四個下屬，宿敵管理者(NemesisManager)：負責“記得”單位要攻擊哪個敵人，並處理攻擊的邏輯，可操控單位管理者(AvatarManager)：所有單位的移動與控制，建築管理者(StructureManager)：所有建築物的建造與統計，滑鼠選取(CursorSelection)：處理滑鼠選取單位、並發布命令。最後集合成Player與AI供場景(Scene)與遊戲內UI(IngameUI)調用。





2. 程式技術

以下主要提到幾個我們所學到的技術

A. 尋路演算法

鑑於本遊戲需要處理大量的可操縱單位同時在充斥著障礙物的地圖上移動，因此勢必要採用一種“高效能”的尋路演算法。因此像是在課堂上所學過的 Dijkstra 或 A*演算法，這種遍歷各種可能、找最優解的算法不適合，同時也不合理（某個傢伙要走到某地，要走之前先從該目標地回推走哪條路，太荒謬了）。因此我們開發一種有別於傳統的算法，名為 crash-and-turn，即當撞到障礙物時則沿著障礙物的邊邊走，直到他又重新找到能朝目標直行的路徑，極為相照真人行走的模式，提供 pseudo code 如下：

```
while we have not reached our destination
    if we can advance in a straight line to the destination point, do
        so
    else
        select one direction (left or right) using one of several
        heuristics
        advance in the said direction keeping your left/right hand
        touching the obstacle's wall
        when you can advance in a straight line towards the target
        again, do so
    end if
end while
```

B. 讀取地圖檔

由於遊戲的地圖之大 (90*90)，因此如果手動分配每個地塊(Tile)會非常的耗時，因此我們做了一個可以自動讀取、生成地圖的程式。原版遊戲提供數種主題的地塊集(tile set)，如叢林、沙漠等，會提供每個地塊的圖片以及一個 YAML 檔，包含每個地塊所定應到的地形、圖片、格式如下：

```
Templates:  
  Template@255:  
    Id: 255  
    Images: clear1.tem  
    Size: 1,1  
    PickAny: True  
    Categories: Terrain  
    Tiles:  
      0: Clear
```

另外，每張地圖均有一份 bin 檔，格式如下：一開始會有 2 個值 x 與 y，代表地圖的大小，以及 15 個數值代表地圖的特性（由於用不到，因此跳過），接下來會有 $x * y * 3$ 個數值，從地圖左上角開始，先行再列，直到右下角，每 3 個值分別代表地塊 ID、對應到圖片編號、以及有沒有礦石。

因此我們先讀取 bin 檔，找到 ID 與對應的照片，ID 對應到 YAML 中的地形與照片檔名，生成我們所設計的地塊類別 (Tile)，包含：名稱、可不可走 (isWalkable)、可不可蓋建築(isBuildable)。

C. “特種”命名技術

由於開發中期遇到瓶頸，因此花了不少時間在檢討原先的 code。在過程中，我們花費大量的時間在閱讀、理解原先的程式，在過程中遭受身心的創傷（例如：兵營的集合點稱為 waypoint，步兵移動的目標地也稱為 waypoint，使得理解上出現混亂）。因而，我們發現程式碼的易讀性、易理解性非常重要，在為變數、類別取名時應秉持“一眼就能看出、不需要看上下文(Context)”的原則在命名，盡量用“更精準的英文單詞”為其取名。比方說：NemesisManager 由於負責管理任一單位所被指派的敵人（當選擇己方單位命令他去攻擊時，便是此類別負責處理），因此取名為“宿敵”(nemesis)，又或著遊戲中玩家可操縱單位命名為 Avatar，亦即“虛擬分身”，又或是單位移動的“目的地”稱為 Destination 而非原先的 waypoint。除此之外也儘量遵從 OOP Doc 內的 Naming Rules，例如：變數前綴應為 m_、函式名稱應為 camelCase（除了 Start 與 Update 外）。“特種”技術亦為，查韋伯英文字典，找到英文字確切的意思，此重要程度更甚於特殊、花哨的程式功能，因此特別提及。

四、 結語

1. 問題及解決方法

在期中我們遇到程式開發瓶頸，不但原先的程式碼太醜（上面有提及），我們也發現要開發新功能愈發的困難，原先我自己的習慣是先動手寫，中途邊修改，但期中之後發現要增加一個功能，通常需要在不同的檔案中增添或修改函式，因此會出現動手寫到一半發現構思的內容與現實不符，導致必須要刪掉增添的函式，因此會浪費許多時間。所以與其如此，不如“先思考再做”，先在紙上演練一遍，確認會調用、修改哪些額外的函式、演練實際執行的流程，確認之後再實作功能出來，自從採用這種流程後，開發過程便更順利。

2. 貢獻比例

盧威任 60%，杜坤翰 40%。

3. 自我檢核表

	項目	完成
1	完成協議書上所描述的最小關卡數量。	V
2	完成專案權限改為 public。	V
3	具有 debug mode 的功能。	V
4	解決專案上所有 Memory Leak 的問題。	V
5	報告中沒有任何錯字，以及沒有任何一項遺漏。	V
6	報告至少保持基本的美感，人類可讀	V

4. 收穫

我（盧威任）學到很多，不過最重要的是我現在知道什麼叫做”邊做邊學”，一邊完善功能，同時應用 OOP 的理念，修改程式的框架。

5. 心得、感想

一開始在規劃製作流程時，我預想到期中之後的精力會漸漸耗盡，因此把大部分的功能製作放在期中之前，秉持著“能用就好”的精神進行開發，但程式碼的擴張比我預期還快，導致幾個問題，一個是程式碼的醜陋與難以理解，導致開發進度放緩，二是許多功能沒經過完善的測試（例如尋路）就匆匆上路，導致後來系統更複雜時，出現更多意想不到的 Bug。如果再來一次的話，我想應該要用“小步快跑、走走停停”式的開發，即兩週做新功能、下一週調

整、完善與反思，循環往復。作為 Leader，必須要對專案方向有更精緻的控制，有時候我會擔太多開發的功能在身上，而迷失在細節裡，應該要根據現實情況實時更新專案的涵蓋範圍、調整預期、調整進度。

6. 對於本課程的建議

大體上來說沒問題，但是助教們都蠻用心在解決問題的，不過有時候解法沒有那麼跟專案的實際情況吻合，辛苦了！