

OOP FINAL PROJECT

GROUP 進擊的宅男

資工四 B98505005 蔣佳航

資工四 B97502052 林哲民

資工三 B99902041 丘紹庭

Our game development's walkthrough



V0.3 增加 GUI 地圖系統



V0.4 增加系統對話框



V0.5 增加對話系統



V0.6 創建起始點，開始加入劇情

	
<p>V0.62 建立系統>狀態對話框</p>	<p>V0.65 建立道具與裝備系統</p>
	
<p>V0.8 新增戰鬥系統</p>	<p>V0.82 完成最終戰鬥系統, DEMO</p>

How you divide the responsibilities of the team members

首先我們一起討論了一些通用的設定，class 之間要如何聯絡，要有那些 class，要有那些系統等。之後分工時，由於組員中有之前做過半即時戰鬥系統的和之前就刻過許多次 GUI 的人，因此採用以下分工方式：

蔣佳航：決定架構、處理 GUI 相關的 class

林哲民：負責處理系統相關的 class

丘紹庭：負責處理遊戲內容、劇本、圖片、音樂等遊戲素材

然而一旦有新的想法，或者對於一個系統的設置方式有疑問，我們就會開一個三人小組會議確定彼此 class 之間的聯絡方式和確定最後要使用的方式為何，以避免之後處理出現問題。

Relations between the classes that we design

我們的 project 主要分成 GUI 和系統兩個部分，GUI 包含各個系統的呈現、

系統間的互動及資料的 I/O 等，而系統的部分就是如道具、戰鬥、地圖、對話等各個系統的實現。

GUI

Package of gui

我們的遊戲從 **MainFrame** 開始，首先將各個 **Loader** 初始化，再加入 **OptionPanel**、**DialogPanel**、**BattlePanel**，方便之後使用。之後遊戲就交給 **Game** 來控制，裡面主要由兩個 **method**: **tick** 和 **run** 來更新畫面的資訊及重刷地圖。其中更新畫面的部分會再交給 **Screen** 處理，**Game** 主要負責決定何時重刷地圖(目前採用的是每次都重刷的方式)。

在遊戲當中的每一個跳出視窗都是一個 **Panel**。如上列所述，**OptionPanel** 負責處理選單，**DialogPanel** 負責處理對話跳出視窗，**BattlePanel** 負責處理戰鬥跳出視窗，而 **Game** 則是處理地圖本體。其中，每次更新畫面時 **Screen** 都會調整現在的狀況變數，並以此變數設定各 **Panel** 的 **visible** 來調整他們的呈現與否。

另外，**ActorKeyListener** 也在本 **package** 內，這個 **class** 會擔任這個遊戲中唯一一個與玩家的鍵盤聯繫的 **object**，負責 **handle** 所有的鍵盤輸入，也是遊戲流程中的最重要控制單位。

Package of gui.io

包含 **BGMFileLoader**、**CharacterLoader**、**DialogLoader**、**ImageLoader**、**MapFileLoader** 及 **SpriteSheetLoader**，用來讀取我們在遊戲中需要用到的素材資料，像是背景音樂、角色圖片、對話和地圖資料等等。這裡面的 **Object** 全部都是 **static** 屬性，原因在於對於一個 **Game** 只需要一個 **Loader** 就可以取得該範疇需要的資訊。透過分類不同的 **Loader**，我們就可以在地圖中、角色中不同的地方，不需要去知道 **Loader** 的 **reference** 即可呼叫使用。

Package of gui.music

BGMPlayer 是用來播放 **BGM**，在每次切換地圖之後會呼叫其中的 **playFile()** 來播放指定的音樂。同樣作為一個 **static object**，不過由於其與音訊有相關，因此特別開一個 **package** 來放置。

Package of gui.option

OptionPanel 用來呈現主選單及各個細選項中的細節。內部有大量的 **JComponent**，並且有大量的 **boolean** 變數來設定他們的呈現與否。當使用者在遊戲中按下 **Esc** 按鈕時，此選單即會出現。

SYSTEM

Package of system

AttributeManage

將角色中的五項屬性 HP,MP,STR,MSTR,AGI 包裝在其中，透過 API 方便道具裝備、技能系統與角色系統之間傳遞屬性值與修改。

DialogHandler & DialogDetail

處理腳本的 class，將存好的角色對話由 DialogHandler 讀出並 parse，再包裝成 DialogDetail 的 type 去存放於角色中。

Package of system.actor

角色的基本 class (BaseActor) 放置在此 package，定義了角色擁有的數值、技能及屬性，同時包含其位在地圖上的座標與移動狀態等。可以藉由繼承的方式創造出各個不同的角色，並擁有各自的屬性、技能和裝備欄等，另 NPC、怪獸皆是 a BaseActor。在與 GUI 的聯絡中，每一個 BaseActor 也會擁有屬於自己的專屬對話、專屬圖片等。

Package of system.arena & system.arena.skill

BaseArena

戰鬥場地的 class，儲存對戰角色，對戰怪獸，場地效果。由於戰鬥系統是採用半即時制(依據場地中每個角色的屬性值決定其攻擊的頻率，而非每個角色在相同時間內都有相同次數的攻擊權)，故用 clock time 與 priority queue 去實現。每個 clock time 去 queue 裡面找看時間點有無事件發生，有則取出事件，並使其發動效果(執行其 act())。這樣定義的好處是在戰鬥系統龐大、技能效果變多，或是判定規則複雜化時，皆可經由定義其 time 與 priority 來達到控制其發生先後順序的效果，故可較簡單的實現"未來攻擊/防禦"或處理某些優先度高技能的相互衝突問題。

Event

定義一個事件的基本型態，包含其 name, message, priority, 效果等等。戰鬥場地內的場地效果、觸發事件、怪獸技能皆是一種 Event，將欲發生的 Event 設定好其 time、priority 和 act() 再加入 queue 中，則時間到達時會依照以定義好的 Event 順序去判定與發動。

AttackSkill

繼承自 Event 的一個普通優先度的純攻擊技能 class，額外記錄其 target、value 與 multiplier 等值，可經由 override 其 calculateValue() 去重新定義傷害的計算方式，或重新 override act() 去修改其發動效果。system.arena.skill 中即是存放以定義好傷害公式與技能效果的 class。

Package of system.item

BackPack

BackPack 是角色的背包，也就是共通的道具欄，攜帶身上擁有的道具及尚未裝備上的裝備清單。

BaselItem

BaselItem 定義了幾個道具最基本的屬性，包含名稱以及敘述等。之後細分成 Item 和 Equipment 兩大類，分別代表消耗品類的道具以及角色的裝備，再藉由繼承這兩個類別就可以創造出各種不同的道具以及裝備。

定義一個基本的物品包含 name, type 與 description。一個物品可由其 type 去得知其是否為消耗品、特殊道具或是會出現在道具欄的哪個分類中等等。此 class 另有 act()、stop() 方便其繼承者去定義其使用效果與拔除裝備效果等。

BaselItemManage

包含所有已定義的物品的 class，可經由物品的 name 或是其 id (位於此 class 內 array 的 index) 去取得其 instance 的 reference。所以在連續使用道具的時候我們不會一直開 class，而是去取得那個 item 的 reference，並使用他的效果。因此每一個道具在遊戲中的反應子都是唯一的。同樣屬於 static object 的一員。

Item & Equipment

兩者皆 is a BaselItem，Item 為消耗品且有其自己唯一的 type。Equipment 為裝備，有多個 type 去區分該裝備可裝備之部位。Item 僅 override act() 去更動其對象角色的現況能力值 (如:目前血量)；Equipment 由於是可重複裝備與拆卸的物品，override act() 去同時增加對象角色的最大和現況之能力值，同樣地，override stop() 去變更對象因裝備而增加的能力及血量。也因為這樣定義出一個 field，我們可以在裝備更動的時候快速的計算更換的位置，而不需要每個位置去一一比對。

Package of system.map

BaseMap & MapChangeDetail & MapMovable

地圖的部分，我們利用了 RPGMaker 中的素材(參考右上圖)來製作。不過我們並不是在 RPGMaker 中畫好地圖然後透過截圖，將存好的圖做為 image 呈現。



取而代之的是，我們將地圖切成一格格 32x32 像素的的方格，而地圖的資料就是分成三層記錄每一格對應的 tile，然後依序填進每一格所對應的 tile 來完成整個地圖，之後再將該 tile 從 tile set 中取出並一格一格貼圖。此外，BaseMap 記錄了每張地圖擁有的基本屬性如：長、寬及地圖上的 NPC、與其他地圖的連結點和 Flag 等。MapChangeDetail 是用來記錄地圖間的關係，在哪個位置可以 teleport 到另一張地圖的某個位置，輔助我們切換地圖。另外一個 MapMovable 是來記錄素材表中每一個素材在遊戲內部呈現的優先順位(共分為：人物下、與人物同級、高於人物三個層級)。因此在遊戲中可以呈現出陰影的效果，也可以呈現出角色在畫面上被其他物件擋住的感受。

MapManage

我們對於地圖物件的創建，在 Game 的 Constructor 之前即會執行完成，因此我們需要一個管理的 class 來負責管理，這就是 MapMange。我們創建 Map 的方式為讀取一個 String 陣列，並利用 Class.forName()的方式 construct。將所有 Map 創建完後，若有一個移動地圖的事件發生，我們只要聯絡身為 static instance 的 MapManage 即可取得每一個 Map 的 reference。如此一來，我們只要在每一個設定好要連到的 Map 的名字為何，就可以實現地圖轉移，而不用讓每一個 Map has a Map。

Advantages of our design

- (1) 方便擴充，增加新的素材。利用 BaseMap、Item、Equipment、BaseActor 等就可以很容易的新增新的內容。
- (2) GUI 跟系統分開，只要設定好兩者間溝通的介面，就可以同時進行開發，又較不容易發生衝突。以本次為例，一人 GUI、一人系統、一人負責遊戲內容的方式在使用我們的規畫時就可以運作得很順利。
- (3) 由於大量利用的 static instance 來擔任不同物件的管理，因此不會有 Obj has a Obj 的情況出現，避免掉多餘的麻煩。並且在未來，我們有機會完成所有的 map 都利用 txt 來定義，而我們的程式只要讀取這些 txt 檔即可自行製造所有 map class 並呈現遊戲。這樣一來就可以讓使用者在不更動 code 的情況下擴充遊戲。
- (4) Panel 和 Panel 之間的概念很直觀，彼此之間的關聯容易理解。由於使用了大量地 class，將一個大 project 拆成無數細項的緣故，我們的 code 的 readability 應該相當的不錯。

Disadvantages of our design

- (1) 每一張地圖都需要花費很多時間來完成，必須一格一格填入需要 tile，最

後才能拼成我們所需要的地圖。這點未來可能要實作出一個 GUI 介面的 **Map Editor** 來補足。

- (2) 由於從頭到尾都沒有使用過任何外部 **library**，因此 **Code** 的效能可能不夠優秀。所有 **code** 都從頭到尾自行創建或許也是我們的特色之一，不過這點可能就是我們的 **Disadvantage**，讓我們在建立系統上花了很多時間