目錄

壹、 簡介 1
一、 動機 1
二、 分工 1
貳、 遊戲介紹 2
一、 遊戲說明 2
遊戲方式 2
遊戲規則 2
特殊功能 3
二、 遊戲圖形 6
三、 遊戲音效 10
參、 程式設計 11
一、 遊戲架構 11
二、 程式類別 13
三、 程式技術 21
肆、 結語 26
一、 問題與解決方法 26
二、 時間表 37
三、 貢獻比例 46
四、 自我檢核表 46
五、 收穫 47
六、 心得、感想 47
伍、 附錄 51
一、 mygame.h 51
二、 mygame.cpp 53
三、 CEraser.h 70
四、 CEraser.cpp 74
五、 CBall.h 89
六、 CBall.cpp 91
七、 CBoss.h 96
八、 CBoss.cpp 98
九、 CBlockMap.h 110
十、 CBlockMap.cpp 112
十一、 CLibrary.h 115
十二、 CLibrary.cpp 122
十三、 CManager.h 152
十四、 CManager.cpp 160
十五、 Refactor.h 189

一、 簡介

1. 動機

起初我們對決定遊戲方向很茫然，深怕做太簡單的遊戲沒挑戰性，又怕做太困難的遊戲無法負擔，於是不知不覺就把念頭動到了 RPG 遊戲上，多結局、開放的世界，使得遊戲的擴充性高，Boss、NPC、小怪等做起來也不簡單；重點是每個物件的重複利用性都會較高，我們不需要花太多時間，去做各種只用一次的物件，還無法被繼承的物件，更是與物件導向的觀念契合。

由於時間被限制在一個學期內，我們不可能無限的擴充地圖、怪物、角色，於是我們決定限制時間，以限制玩家能活動的範圍。而這時正好看到遊戲實況主在試完一款叫「隕石 60 秒」的遊戲，玩家要在 60 秒內與地圖上的各種人物互動、而隨著互動方式、以及接觸的人不同，遊戲會導向不同的結局。與我們的想法不謀而合，於是就決定往 2D 卷軸，有 NPC、小怪，BOSS，且有多結局的小品卷軸 RPG 了。

2. 分工

郭宗育	徐紹崴
主程式	角色
地圖管理、地圖編輯器	UI 介面
Boss、小怪、NPC	地圖製作
優化讀寫輸入	大部分素材蒐集、處理

二、 遊戲介紹

1. 遊戲說明

a. 遊戲方式

A、D 為左右移動，W 為跳躍，滑鼠左鍵為射擊

b. 遊戲規則

主角有血量(左上)、魔力(左上下面)兩種狀態，當生命值歸 0，即進入結局，而魔力歸 0，則會開始持續扣生命值。而時間(中上)結束，也會強制進入結局，另外狀態欄下方的是積分，可以透過即殺小怪來獲得。

c. 特殊功能

i. 傳送門：

↑ 在門上按空白鍵使用傳送門，直接移動至其他地圖

ii. 對話框

↑ 遇到 NPC 時按 Z 鍵可與其對話

↑ 滑鼠左鍵按任意處可繼續對話，按 Y 鍵可跳過對話

iii. 暫停面板

↑提供繼續遊戲、重新開始、回主選單、離開、繼續

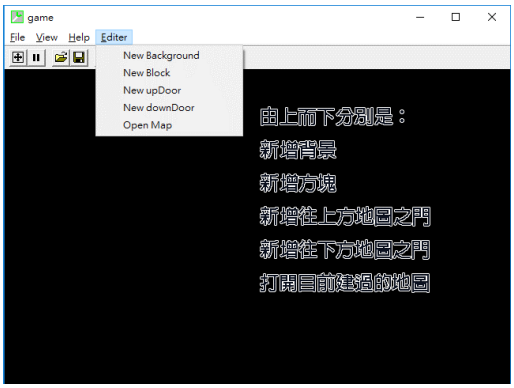
iv. 回顧結局

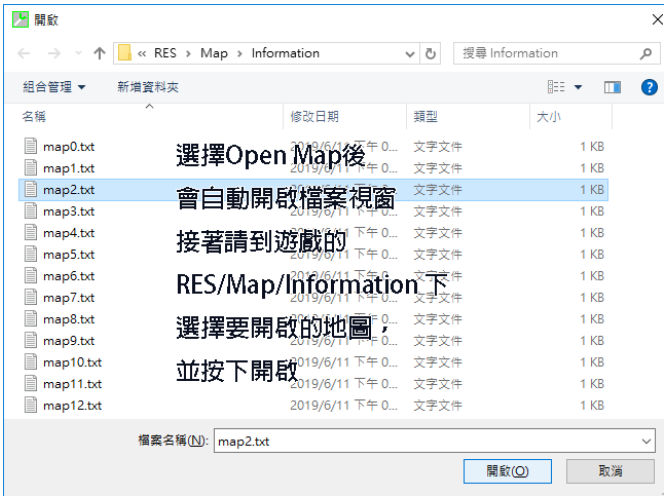
↑ 可透過滑鼠滾輪捲動頁面，瀏覽更多結局，已破過的結局可以點選，回顧該結局

v. 地圖編輯器

提供快速的修改、新增地圖介面，

在主程式中按下 N 鍵後進入





↑ 以 Open Map 存在的地圖，地圖資訊 txt 檔在遊戲底下的 RES\\Map\\Information

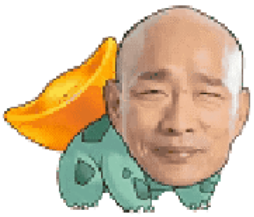
鍵盤按下 A、D 鍵：左右捲動背景(背景圖長度需大於 640)
鍵盤按下 Tab 鍵：查看目前地圖的編號
鍵盤按下 S 鍵：儲存目前地圖
(新地圖需儲存在 RES\\Map\\Information)
滑鼠拖曳畫面上的障礙物或門：拖曳該物件
滑鼠點擊畫面上的障礙物或門，並按下 delete 鍵，
刪除選取的物件。

再開啟遊戲時，地圖就會編輯後的狀態

2. 遊戲圖形

角色、NPC

Boss

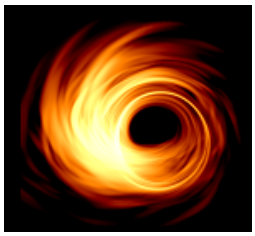



怪物：

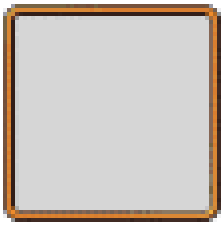
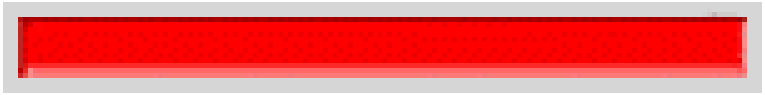
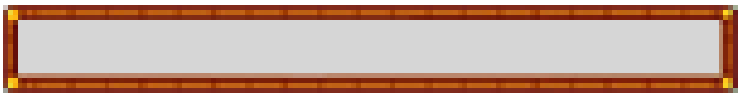
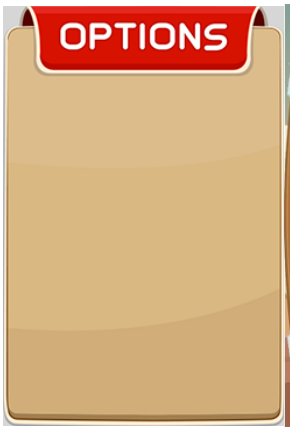


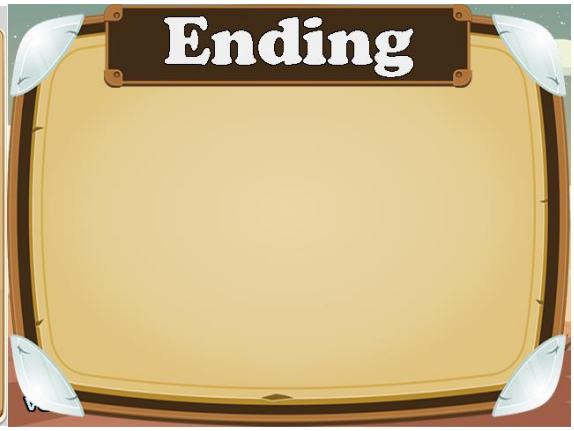

投擲物、攻擊元素：



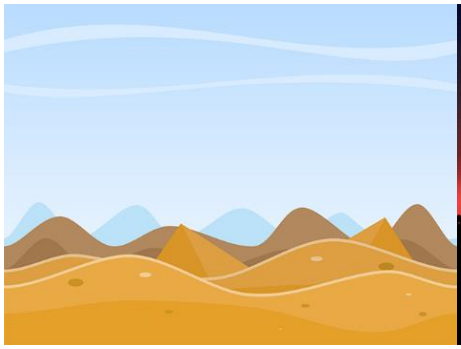




UI



地圖




すろおもおしょん
SLoWMoTIoN
ver.1.19.0

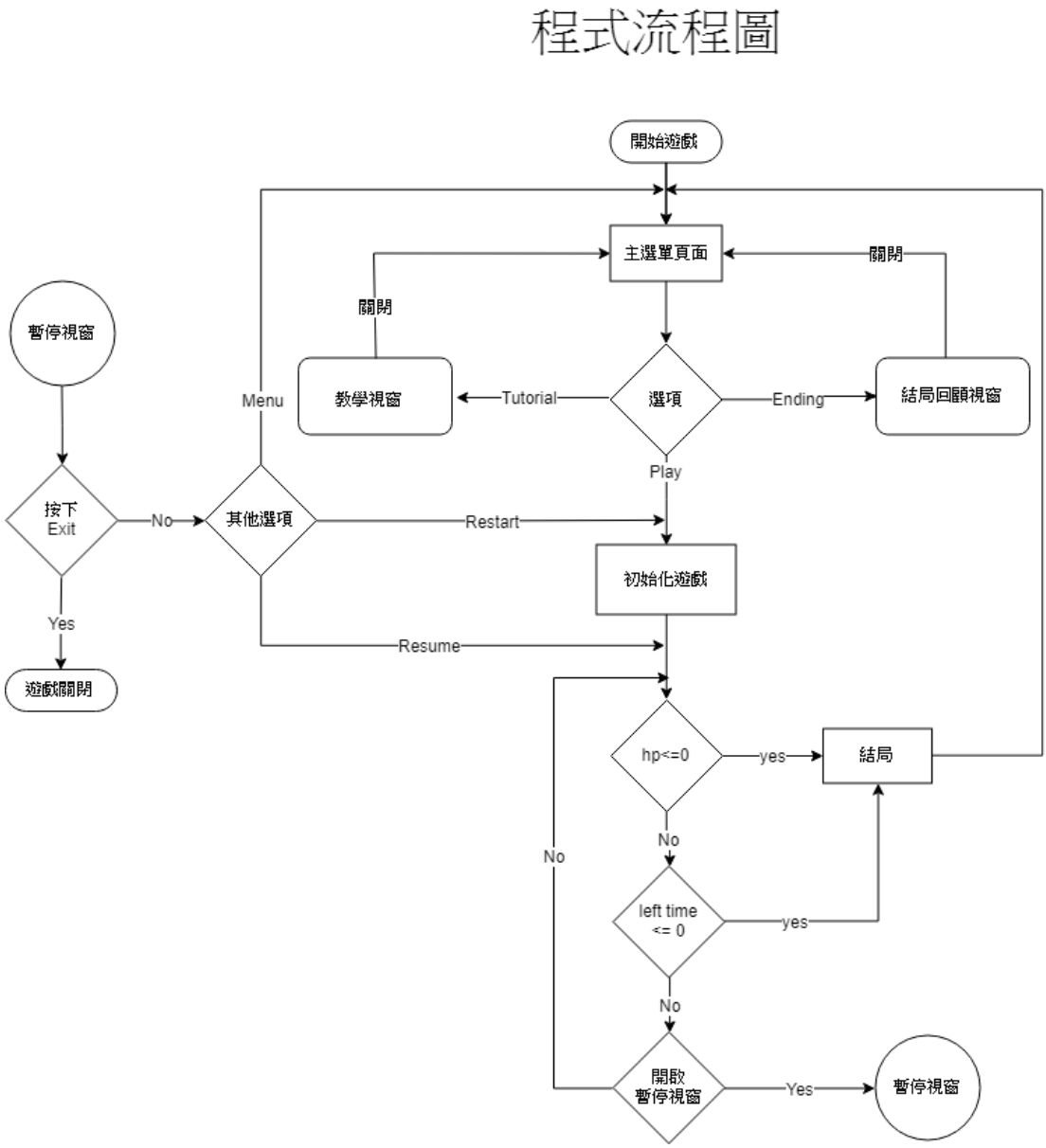

Game OVER

3. 遊戲音效

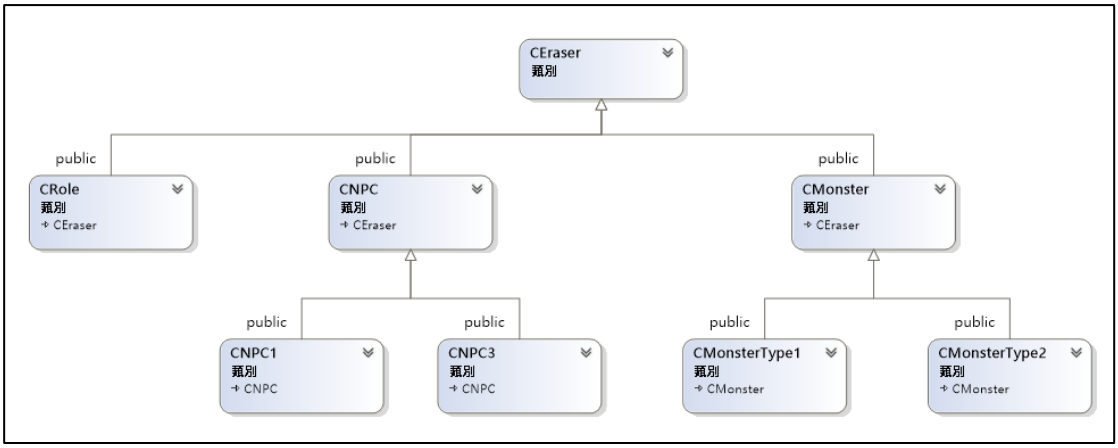
類型	名稱	說明
音樂	SLoWMoTIoN_Menu	於主選單時之背景音樂
	SLoWMoTIoN_Game	於遊戲中之背景音樂
	SLoWMoTIoN_Gameover	進入結局前之音樂
	FloweringNight	與特定 NPC 對話時撥放
	MyVoiceIsDead	
	NativeFaith	
	Septette	
	UN_OwenWasHer	
音效	blackhole	Boss Xingting 黑洞攻擊音效
	book_shoot	Boss Xingting 書本攻擊之音效
	facai_fly	發財種子飛走時之音效
	faqai	發財種子 NPC 之聲音
	ray	發財種子，光束攻擊之音效
	door	使用門傳送時之音效
	hit	命中小怪時之音效
	jump	角色跳躍之音效
	role_hitted	角色被打到時之音效
	throw	角色攻擊之音效
	not_clear	Ending 視窗中，按下未解鎖結局之音效
	open_window	打開視窗類時之音效
	page_close	關閉視窗類時之音效
	btn_click_3	按鈕按下時之音效
	btn_collision_3	滑鼠狀到按鈕時之音效

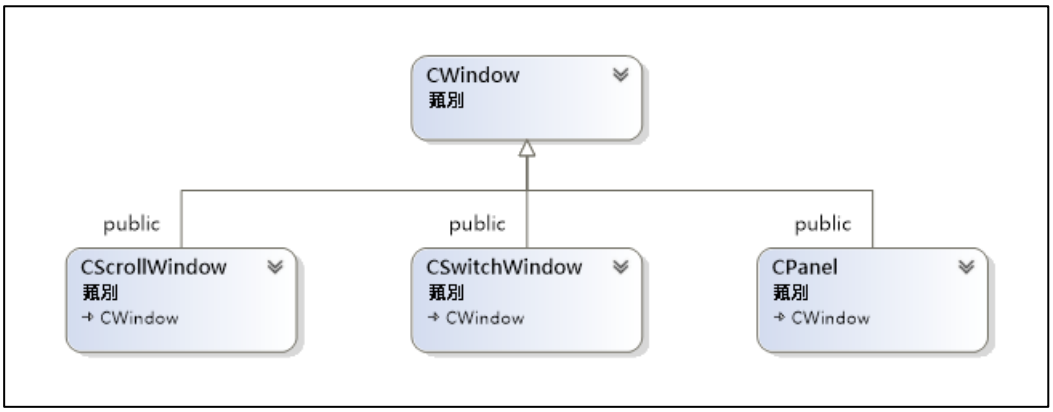
三、 程式設計

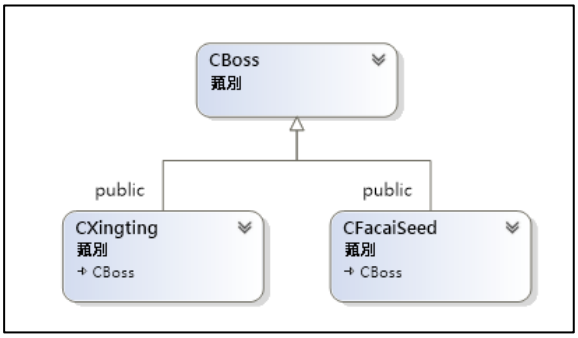
1. 遊戲架構

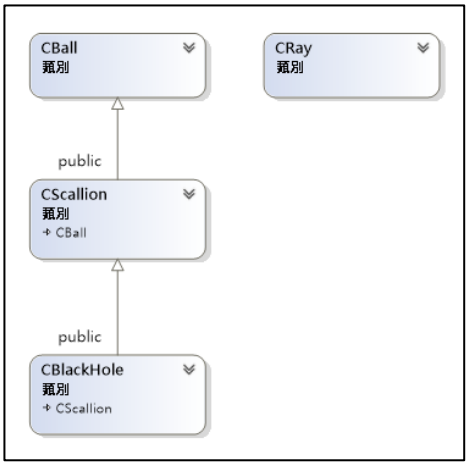

程式流程圖
開始遊戲
主選單頁面
關閉
關閉
暫停視窗
教學視窗
Tutorial
選項
Ending
結局回顧視窗
Menu
Play
按下
Exit
No
其他選項
Restart
初始化遊戲
Yes
Resume
遊戲關閉
hp<=0
yes
結局
No
No
left time
<= 0
yes
No
開啟
暫停視窗
Yes
暫停視窗

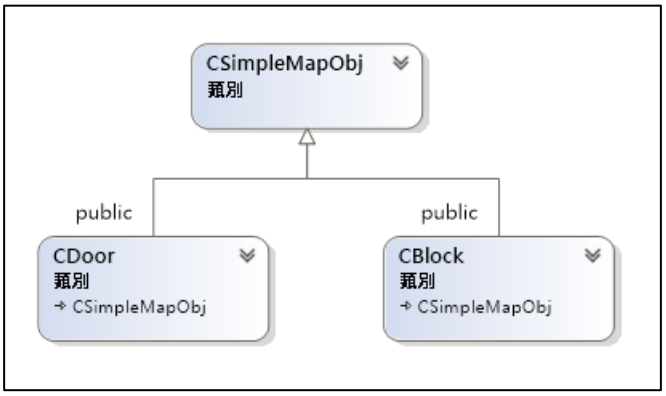
Class Diagram











2. 程式類別

類別名稱	.h 檔行數	.cpp 檔行數	說明
CEraser	66	192	CRole、CMonster、CNPC 的父類別，提供基礎函式與屬性。
CRole	113	441	遊戲中玩家操控的角色，具有移動、跳躍、攻擊(丟蔥)、碰撞等操作。
CNPC	35	101	遊戲中玩家能與之對話的 NPC，具有虛擬函式，為 NPC1、NPC3 之父類別。
CNPC1	13	19	遊戲中玩家能與之對話的其中一種 NPC，負責一般的對話。
CNPC3	19	50	遊戲中玩家能與之對話

			的其中一種 NPC，與之對話時會播放音樂(簡單來說就是音樂播放器) 。
CMonster	30	200	遊戲中玩家能以攻擊消滅的小怪物，可以移動以及跟障礙物碰撞，消滅時會玩家獲得分數，具有虛擬函式，為 CMonsterType1 以及 CMonsterType2 的父類別。
CMonsterType1	7	13	遊戲中玩家能以攻擊消滅的小怪物，沒有動畫。
CMonsterType2	8	45	遊戲中玩家能以攻擊消滅的小怪物，具有動畫。
CBall	23	51	CSacllion 的父類別，提供基礎函式與屬性。
CSacllion	29	128	玩家或 Boss 能丟出的子彈，能設定以拋物線軌道或直線軌道飛行。
CBlackHole	19	80	女老師可以丟的特殊子彈，繼承自 CSacllion，當黑洞(此類別)與玩家碰觸時，會將玩家吸向黑洞的中心。

CRay	24	32	直線射線，具有動畫，只有動畫的最後一格能造成傷害。
CBoss	73	168	Boss 的基礎類別，具有虛擬函式，提供 Boss 一些基礎函式與屬性。
CXingting	46	299	繼承自 CBoss，會以圓周運動丟子彈(遠東單字本)攻擊玩家，並且會丟黑洞，該 Boss 無法被玩家攻擊，玩家須閃躲攻擊 40 秒後，此 Boss 會自爆死亡。
CFacaiSeed	39	289	繼承自 CBoss，會丟出金幣、釋放射線攻擊玩家，玩家與他接觸時會降低魔力，當玩家的魔力降為 0 的時候，玩家會持續扣血。該 Boss 可被玩家攻擊，其血量歸零，Boss 死亡。
CSimpleMapObj	25	0 (函式直接定義在.h)	地圖上使用的簡單物件，提供一些基礎函式與屬性。
CBlock	9	0	繼承 CSimpleMapObj 的

		(函式直接定義在.h)	簡單物件，為擺在地圖上的障礙物，角色能站在障礙物上，碰到障礙物時無法移動。
CDoor	20	0 (函式直接定義在.h)	繼承 CSimpleMapObj 的簡單物件，為傳送門，角色能透過門傳送到上方地圖/下方地圖。
CBlockMap	64	168	遊戲中使用的地圖，一個 CBlockMap 即儲存一張地圖的資訊，可以透過 txt 讀取資訊，也可以將資訊寫出來。
myLibrary (namespace)	11	149	自定義 library，有許多函式，像是將 string 轉成 char*，取得資料夾內的所有檔名、取得資料夾內的檔案多寡，將字串轉成數字，切割字串等等。
BitmapPath	23	0(函式直接定義在.h)	儲存圖片、動畫的路徑、檔名、數量、透明色等。
IsPointInRect (function)	1	7	自定義函式，判斷點是否在目標矩形內。
IsRectCollision (function)	1	4	自定義函式，判斷矩形是否與目標矩形碰撞。

CTimer	22	61	提供計時功能。
CAnimate	45	239	改寫自 framework 內類別 CAnimation，提供動畫功能。
CInteger	31	77	改寫自 framework 內類別 CInteger，提供數字圖像化功能。
CLayer	12	24	可以設定/取得圖層。
CDialog	29	95	儲存對話所需要的資訊，包含文本、名稱、能否重複觸發等。
CCamera	23	68	使螢幕的顯示範圍限縮在 Camera 照到的區域，換句話說，使地圖具有卷軸功能。
CButton	45	172	提供有圖片，可點擊的按扭。
CAction	54	211	提供動作功能(各種不同動作的動畫)。
CEnd	22	67	儲存結局的資訊，包含圖片、文本、是否已獲得該結局等。
CToumeiImage	24	62	提供帶透明度的圖片(需配合 alphaBlend)
CStatusBoard	26	85	提供玩家的 UI 介面，包

			含血量、魔力、頭像等。
CBossBoard	37	127	提供 Boss 的 UI 介面，包含血量、頭像等。
CWindow	37	115	提供彈出式視窗，具有背景與關閉按扭。
CScrollWindow	27	177	繼承自 CWindow，為一可滑動視窗，可透過滑鼠滾輪調整顯示範圍。
CSwitchWindow	25	158	繼承自 CWindow，為一可切頁視窗(透過螢幕上的箭頭)，同時滑鼠點擊左/右鍵可以換螢幕上的圖(下/上一張)。
CPanel	22	63	繼承自 CWindow，為一彈出式視窗，用於遊戲中的暫停頁面，可使玩家選擇繼續遊戲、重開遊戲、回到選單、離開遊戲四種操作。
CLayerManager	32	120	圖層管理器，將所有圖片、動畫、動作、圖像數字蒐集起來，在此依照圖層分類並輸出。
CMapManager	63	337	地圖管理器，創建、管理地圖資訊(CBlockMap)，

			包含圖片、位置、地圖上的物件等，能切換地圖，以及使 Monster 重生。
CMonsterManager	19	79	小怪物管理器，負責創建、刪除小怪物。
CBossManager	18	77	Boss 管理器，負責儲存 Boss，並且提供選取 Boss 功能。
CDialogManager	53	382	對話管理器，負責創建、管理對話(CDialog)，並提供 CDialog 開始對話、結束對話、顯示對話等功能。
CNPCManager	19	81	NPC 管理器，負責創建、管理 NPC，能在切換地圖的時候顯示不同的 NPC。
CEventManager	16	72	事件管理器，負責觸發主程式的一些事件，以位置事件為主。
CEndManager	37	179	結局管理器，負責創建、管理結局(CEnd)，並提供 CEnd 開始結局、結束結局等功能，並能記錄獲得哪些結局。

CButtonManager	32	107	按鈕管理器，負責管理按扭(CButton)。
UIManager	23	36	UI 管理器，控制角色、Boss 的血條，以及玩家獲得的分數。
ImageInfo	15	0(函式直接定義在.h)	繼承 CSimpleMapObj，僅在地圖編輯器中使用，為地圖編輯器上的物件。
Arrow	50	0(函式直接定義在.h)	僅在地圖編輯器中使用，設置地圖模式中出現的可點擊切換地圖的箭頭。
CMapEditer	148	453	地圖編輯器，以工具列提供簡單編輯地圖功能，可以新增/載入/儲存地圖，可以新增障礙物或門在地圖上，且障礙物或門可以進行拖曳。其他還可以設置連結地圖。
CGameStaeInit	26	207	遊戲的開頭畫面，具有三個按扭：進入遊戲、觀看擁有結局，教學頁面
CGameStateRun	48	677	遊戲的遊戲畫面，為程式的主要部分。

CGameStateOver	21	106	遊戲的結束頁面,在這裡播放結局。
CGameState MapEditer	23	153	遊戲的地圖編輯頁面,在這裡使用地圖編輯器
總計	1822	7303	

3. 程式技術

技術	說明
檔案讀寫	以 fstream 讀取、寫入檔案。
分割字串	透過 stringstream，以空白將字串切割，並能得到切割後的子字串。
字串轉換數字	透過 stringstream 讀取字串，並將其輸出給整數型態，從而得到數字。
點與矩形碰撞	透過點的座標與矩形的座標，判斷點是否在矩形中。 將此函數擴展,可以得到矩形與矩形碰撞(矩形的四個點，任一點在目標矩形中，即與目標矩形碰撞)。
圖層	將圖片/動畫/動作等賦予圖層，並交由圖層管理器(layerManager)統一顯示。 其概念為：將圖片/動畫/動作放入對應圖層(index)的陣列裡面的 vector，而圖層管理器在輸出的時候，從陣列 index 為 0 開始輸出圖片，直到陣列 index 為最大圖層數時停止，如此一來，圖層叫小的圖片/動畫/動作會先被呼叫顯示，而圖層大的則後呼叫顯示。 為了方便將圖片/動畫/動作加入圖層管理器，圖層管理器設置為 static，令其能在任何地方被呼叫。

對話	對話能讀取文本中的字串，並由對話管理器(CDialogManager)統一管理所有的對話。 其概念為：對話主體(CDialog)能夠以 fstream 讀寫檔案，配合字串分析，取得對話文本中「誰說了什麼話」，而對話管理器 CDialogManager 統一管理所有的對話，並提供介面使管理的對話能執行開始對話(Start)、結束對話(Stop)、下一句(Next)、顯示對話(showText)。 其中開始對話以指標取得指定的 CDialog，並進行初始化設定，結束對話則將指標還為 nullptr。顯示對話為主要部份，會取得指定的 CDialog 的字串，並將其以最大字數(bytes 數，ASCII 體系，中文 2bytes，英文 1bytes)切割為個別的子字串，再將切割完的子字串交由 CDC 繪出。 對話同時也具有逐字顯示的效果，是利用計時器，每隔 0.1 秒改變指定字串的長度來達成。 為了方便讓對話能在任何地方被觸發，因此對話管理器設成 static。
結局	結局利用圖片+對話的方式呈現，同樣由結局管理器(CEndManager)統一管理所有結局。 其概念為：一個結局具有多組圖+多組對話，一張圖片會搭配一個對話，當圖片播放完時代表結局結束。 結局管理器(CEndManager)提供介面使管理的結局能執行開始結局(Start)、結束結局(Stop)，同時能透過 fstream 讀取「擁有哪些結局」並記錄之，使得擁有的

	結局不會因為關閉程式而遺失。 結局中使用的圖片帶有淡入淡出的透明度效果，其透明度以 MFC 自帶的 CBitmap 以及 CDC 內的 alphaBlend 函式達成。
透明圖片	使用 MFC 自帶的 CBitmap 以及 CDC 內的 alphaBlend 函式達成，CBitmap 雖然本身並不帶透明度，但可以將其傳給 alphaBlend，並在 alphaBlend 額外賦予透明度，將 CBitmap 以帶透明的效果顯示。
資料夾操作	能夠開啟指定的資料夾，並讀取資料夾內所有的檔名，或是取得資料夾內的檔案總數目。 資料夾操作藉由 dirent.h 達成，dirent.h 為 unix 系統下，C/C++的 library，但很不幸的 Windows 並不支援，因此我們使用他人改寫能相容於 Windows 系統的 dirent.h。 程式中透過 while 迴圈以及 readdir()操作，讀取指定資料夾內所有的檔名，利用此方法，我們成功自動化讀取音效音樂、對話文本、對話頭像、結局等等需要大量 Load 指令的檔案，使用上更為人性。 Github：https://github.com/tronkko/dirent
滑鼠滾輪事件	afxwin 中已經有定義滑鼠滾輪事件 onMouseWheel()了，因此只需在 CGame 中新增 virtual function：OnMouseWheel，即可在 CGameStateInit、CGameStateRun、CGameStateOver 中使用滑鼠滾輪事件，另外，滾輪事件中傳遞的 short 參數 zDelta 為判斷滾輪是上滾或是下滾，zDelta > 0 代表滾輪上滾，

	zDelta < 0 代表滾輪下滾。
複製檔案	MFC 使用原生函數 CopyFile CopyFile(原檔案路徑, 目的地路徑, 是否不覆蓋檔案)
刪除檔案	MFC 使用原生函數 DeleteFile DeleteFile(檔案路徑)
工具列	工具列的使用必須先在 game.rc 的 MENU 中新增按鍵，並設置按鍵的 ID，然後在 gameView.cpp 中設置並撰寫對應按鍵 ID 的函式。 在程式中我們希望能利用工具列執行主程式上不同的操作，因此利用 fstream 檔案讀寫技術，當點擊工具列特定按扭時，以 txt 形式將指令寫出，同時在主程式讀取該 txt 並分析其內容，再根據內容做出不同的操作，藉此達到目的。
地圖編輯器	地圖編輯器提供簡單的圖形化編輯地圖功能，能以工具列新增/開啟地圖、新增物件(障礙物或門)。 新增出來的物件可以藉由滑鼠拖曳改變其位置，其概念為：當滑鼠點擊時，選取被點擊的物件，並在滑鼠持續按住左鍵的狀態中，讓選取的物件隨著滑鼠座標移動。同樣概念，被選取的物件也可以透過 delete 刪除。 編輯好的地圖可以進行存檔操作，在存檔操作中會把地圖編輯器上的物件轉換為基礎地圖資訊(CBlockMap)，並透過 CBlockMap 自帶的函式將地圖資訊寫出。 在地圖編輯器編輯完地圖後，不需重啟程式來重新載

	入地圖。我們在編輯的過程的同時紀錄「哪些地圖被編輯過」，在重新回到遊戲的時候，將被編輯過的地圖重新創建，替換掉原本的地圖，達到線上重新載入的功能。

四、 結語

1. 問題與解決方法

問題一：

學期初執行程式時，程式偶爾會跳出存取被拒，導致不能執行。

解決方法：

經過反覆測試後發現問題出在我的電腦有一個遊戲有防外掛系統，因此我刪除該遊戲永絕後患，問題就解決了。

問題二：

MFC 中很多原生函式都只接受 char*型態，但寫 C++程式時常以 string 為主。

解決方法：

自行做一函式將 string 轉為 char*：先用 new 關鍵字配置一些空間給 char*，在將 string 的內容複製給 char*並回傳，要注意的是，因為有 new，所以新創建的 char*在使用完後必須 delete，否則會 memory leaks。

問題三：

在做地圖的卷軸效果的時候，原本預期達到的效果是人物走→走到一半換成地圖捲動→地圖捲到底時，在換人物走，但有時候會發生人物走到一半地圖卻沒有捲動。

解決方法：

原本的判斷是計算人物的座標是否達到開始捲動的值(達到時地圖捲動)，以及判斷地圖是否捲到底(捲到底則地圖停止捲動，人物開始行走)，後來新增 Camera，在人物行走的時候修正 Camera 的值，並透過 Camera 的值判斷地圖是否該開始捲動。

問題四：

地圖捲動的時候，地圖上的物件沒有跟著捲動。

解決方法：

將物件的座標分成兩種：實際上的座標、物件在螢幕上的座標，物件在自主移動的時候更改自己實際上的座標，而物件在螢幕上的座標，則透過實際座標減去 Camera 的值求得。

問題五：

每次要更改物件顯示的優先權很麻煩(例如物件 A 要顯示在物件 B 上面，就要在 OnShow()將物件 A 的程式碼放在物件 B 下面) 。

解決方法：

製作 layerManager 負責管理圖層，該 Manager 為 vector<圖片>陣列，只需將個別物件的圖片放入對應陣列裡的 vector，再將此 vector 陣列由小到大輸出即可，舉個例子：設定物件 A 的圖層為 1，物件 B 為 2，如此將物件 A 放入 vector<圖片>[1]中，物件 B 放入 vector<圖片>[2]中，而輸出的時候從 vector[0]輸出到 vector[9]，這樣一來，圖層小的圖片自然會被圖層大的圖片蓋住。

問題六：

有時候需要計時，時間到執行某個動作。

解決方法：

將 gameframe_work 原本的時間計算統合整理成一類別：CTimer，CTimer 會計算出「要跑過多少個執行緒」，使用時透過其函式 CountDown()將計數-1，同時使用 IsTimeOut()判斷是否已經計時完成，若計時完成，則執行自定

義的程式，同時透過 ResetTime()回復 Timer 的計數。

問題七：

每次碰撞都要取出點座標進行判斷，不僅麻煩也容易出錯。

解決方法：

在製作的過程中，我們發現 MFC 自帶 CRect，用來表示矩形的位置，因此我們製作函式 IsPointInRect(CPoint, CRect)，傳入點座標與矩形，用來判斷「該點是否在目標矩形內」。然後擴展成另一個函式 IsRectCollision(CRect,CRect)，判斷「矩形是否與矩形碰撞」，其概念為：將矩形 A 的四個座標取出來，透過 IsPointInRect 判斷，如果四個點中有一個在目標矩形中，那麼兩矩形就形成碰撞。

該函式的好處是，由於 CMovingBitmap 內本來就有表示位置的矩形，因此只要將矩形取出，就可以簡單的判斷兩個圖片(或物件)是否碰撞。如果要更改碰撞範圍，也只要修正矩形的範圍就好。

問題八：

有時候執行程式時，會馬上彈出 CDDraw 的某個東西為 nullptr。

解決方法：

發生該問題時，大多數原因是出自在 default constructor 中 Load 圖片，解決方式很簡單，將 Load 圖片從 default constructor 移出，另外做個 LoadImg 的函式，並在遊戲的 OnInit()呼叫 LoadImg 來載入圖片。

問題九：

CGameStateInit 的 OnInit 跟 OnBeginState()是反的，因此在 OnInit()載入圖片，在 OnBeginState()進行初始化的時候，會因為「圖片還沒被載入」而

使程式崩潰。

解決方法：

架空 OnInit()，將所有程式碼移到 OnBeginState()，並使用布林變數判斷「哪些部分只須要做一次」(像是載入圖片、音效，只需做一次)。

問題十：

製作對話的時候，發現如果將所有字串都硬設在程式裡，要修改時相當不容易。

解決方法：

透過 fstream 開啟指定的 txt 檔，並將其內容讀給自定義的類別 CDialog，同時透過 CDialogManager 管理所有的 CDialog，要開啟對話的時候透過 CDialogManager 開啟指定的 CDialog 即可。

問題十一：

很多情況下都會觸發到對話，如果每個要觸發對話的物件都必須擁有 CDialogManager 的指標來控制，在寫程式上會有些不方便(因為要把指標傳來傳去，常常傳到最後不知道自己在幹嘛) 。

解決方法：

參考前人所作之 CAudio，將 CDialogManager，static 化，令其能在所有地方都能被呼叫。

問題十二：

CDialog 對話所讀取的字串如果太長，在顯示的時候會超過遊戲畫面。

解決方式：

限制每一行最大字數(bytes 數)，超過該最大字數就換行，注意這裡的換

行並不是字串加上\n，而是「換一個 char*」給 CDC 輸出，如此才能確保位置正確。

問題十三：

承問題十二，CDialog 對話在換行時，由於中文和英文所使用空間不同(ASCII 下中文 2bytes，英文 1byte)，在切割字串時若正好切到中文，則輸出時會顯示亂碼。

解決方法：

該問題原因來自於，因為中文要兩個 bytes 在一起才能正常顯示，如果正好將中文切成 2bytes，第一行的末尾拿到第一個 byte，第二行開頭拿到第二個 byte，則該中文的編碼則沒有連續，因此產生亂碼。解決方法是判斷「如果拿了這個中文字，會不會導致這行的總 bytes 數」，如果會超標，則把該中文字放到下一行顯示。

問題十四：

CDialog 對話在進行的時候，該如何讓文字逐字顯示？

解決方法：

將 CDialog 讀取到的字串切成長度為 k 的子字串，而 k 的值隨著時間逐漸增加，在輸出文字的時候選擇子字串輸出。

問題十五：

地圖上的小怪物如何重生？

解決方法：

當子彈與小怪物碰撞時，將刪除小怪物並將其從 vector 中移除，與此同時在 vector 中 new 一個新的小怪物，並賦予新的小怪物一亂數時間，當時

間到的時候才讓小怪物顯示在地圖上，如此就有一種重生的效果。

問題十六：

CAudio 對使用何種音效的定義寫在 mygame.cpp，若要在 mygame.cpp 以外的地方使用要把該定義傳來傳去很不方便。

解決方法：

改寫 CAudio，將使用到的音效以 map 的方式儲存起來，並利用多型新增 Play(string)，令 CAuido 可以使用字串當 map 的 key 來使用音效。

問題十七：

CAudio 每次多一條音效時都要多寫一行載入程式來載入音效，挺麻煩的。

解決方式：

使用 dirent.h 取得指定資料夾中的所有檔名，如此一來就可以一次載入所有音效，每當新增一條音效的時候只要將音效放進指定的資料夾即可自動載入。

dirent.h 為 unix 系統下，C/C++自帶的 library，能夠操作資料夾，但 windows 的 library 並不支援，因此本程式使用他人改寫自可相容 windows 的 dirent.h。

github：https://github.com/tronkko/dirent

問題十八：

CDialog 對話每次新增新的文本(txt)或是人物頭像時都要多寫一行程式來載入文本或圖片，很不方便。

解決方法：

呈問題十七之解，透過資料夾操作取得指定資料夾的所有檔名，一次性載入所有文本、人物頭像，如此一來只要將檔案放進指定的資料夾就可以使用。

問題十九：

製作完 Boss 以及 NPC 之後，已經確定所有 new 出來的空間都有正確 delete 掉，但還是莫名其妙多出很多 memory leaks。

解決方法：

因為 Boss 以及 NPC 具有 virtual function，底下有個別的子類別繼承他們倆，而製作的時候忘記具有 virtual function 的 class，其 deconstructor 也要變成 virtual function，否則只會解構父類別，而不會解構到子類別。

問題二十：

CMovingBitmap 只能讀 bmp，但又想要讓結局的圖片有透明效果

解決方法：

CMovingBitmap 在讀取的那一瞬間就已經決定好圖片的透明色、底色，即使套用 CDC 的 alphaBlend，也只會得到背景是黑色的透明色，因此在做有透明效果的圖片時採用 MFC 原本的 CBitmap，並搭配 CDC 的 alphaBlend 轉換透明度，缺點是硬體設備不夠好的電腦執行速度可能會有點慢。

問題二十一：

想要在關掉程式後依然能夠存取已經擁有的結局。

解決方法：

在離開遊戲的時候將「目前擁有的結局」透過 fstream 寫入 txt 檔儲存。

問題二十二：

能在擁有結局的頁面中透過點擊的方式看結局的劇情。

解決方法：

擁有結局的頁面中放入自行製作的 CButton，令其具有點擊事件，並在點擊的時候判斷是否擁有該結局，如果有，則透過 CEndManager 播放對應的結局劇情。

問題二十三：

兩個.h 互相引入對方，例如想在 role 所在的 CEraser 中，引入投擲物 CBall，又想在 CBall 中取用 role 目前的位置，就需要交互引入兩個標頭檔。

解決方法：

根據上述例子，交互引入對方後，在宣告 class CBall 之前使用先前置宣告 class CEraser 即可。

問題二十四：

string 如何以空白切割字串？

解決方法：

透過 C++的 stringstream 字串符流，將字串以<<運算子添加到 stringstream，全部添加完後，再用>>運算子添加回 vector<string>，最後的 vector<string>即是分割完後的字串 vector。

問題二十五：

做地圖編輯器的時候，想利用工具列做出新增物件/載入地圖等等功能，但工具列實作在 gameView.cpp 中，跟 mygame.cpp 似乎沒有交集，因此在 mygame.cpp 取得不到透過工具列得到的檔名。

解決方法：

以新增物件為例，當透過工具列新增物件的時候，將新增的物件複製到指定的資料夾中，同時在該資料夾以 fstream 寫下 txt，其內容為「新增的物件種類」以及「該物件的檔名」，兩字串以空白分割。

新增完 txt 後，在 mygame.cpp 中檢查「是否有新增的 txt」，如果有，則分析該 txt 的內容，並將其新增的物件添加到地圖編輯器中，同時刪掉新增的 txt。

問題二十六：

地圖編輯器中新增的物件想要進行拖曳以及刪除操作。

解決方式：

在地圖編輯器中，當滑鼠點擊螢幕時，抓取「點擊到的物件」，如果按住左鍵並移動，則讓抓取的物件隨著滑鼠座標移動，如果按下 delete 鍵，則將物件從刪除。需要注意的是，抓取的物件移動時設定的位置不能是滑鼠的位置，應該要設成滑鼠位置減去「點擊瞬間，滑鼠位置與選擇物件的座標差」。

問題二十七：

新增/編輯好的地圖要存檔。

解決方式：

將地圖編輯器中新增/編輯好的地圖，其地圖上的物件轉換成遊戲中使用的地圖 CBlockMap，並使用 CBlockMap 內的函式新增/覆寫地圖資訊。

問題二十八：

編輯好地圖後，希望能直接測試，不用關掉程式讓地圖重新載入。

解決方法：

此問題來自於原本載入地圖發生在開啟程式的瞬間，因此如果在地圖編輯器修改完地圖，重新進入遊戲時是不會重新載入地圖的。而如果改成每次進入遊戲都重新載入地圖，又感覺略沒效率，因此我們採用：在存檔地圖的同時，紀錄「被修改過的地圖編號」，在進入遊戲的時候，只重新創建被修改過的地圖，替換掉原本的地圖。

問題二十九：

人物在 2D 卷軸上，想把上(W)鍵，做成跳躍

解決方式：

在人物中給予 y 方向的初速度，以及重力，判斷在跳躍後，每次都將初速減掉重力來做為目前 y 軸之速，即完成跳躍。

問題三十：

人物跳躍後落在地圖物件或地上，要停止

解決方式：

在無額外地圖物件的情況下，當人物的 y + height(也就是底部)>=480 時，便修正其 y 坐標，並設置人物是否在跳躍中的布林變數為 false，使人物重新「站在地板上」；而在方塊上則是與 480 的判斷，換成與地圖物件頂部坐標的判斷，其餘是同樣的方式。

問題三十一：

人物跳在地圖物件上後，離開不會下降

解決方式：

在主程式中加入 isOnFloor，來判斷是否在地面或地圖物件上，並在人

物中加入 GetDrop，來獲取目前是否處於下降狀態，當 isOnFloor = false，並沒有在跳躍時，將人物強制設成在下降狀態，並無法跳躍，直到回到地板，或地圖物件上。

問題三十二：

人物之攻擊方式，想以拋物線，而不是直線進行攻擊

解決方式：

將滑鼠坐標，與人物之坐標為基礎，計算出一初速度，並賦予重力使其會以拋物線方式運動。

問題三十二：

在人物攻擊後，出現大量 memory leak 問題

解決方式：

判斷丟出物件是否該消失(碰撞到 BOSS、小怪，或超出地圖後)，delete 該物件，並使用 erase 將 vector 其位置清除。

問題三十三：

人物的動畫只能循環跑圖

解決方式：

新增類別 CAction，以 map 將動作，對應至該動作之 vector，並再主程式中判斷應使用何動作，以及目前面向哪邊，在 onMove 時，傳入 action 中，以動作_方向(如向左的閒置動畫：idle_L)作為 key 值，循環執行該動作之 vector。

問題三十四：

人物的動畫，在左右方向切換時，有閃爍現象

解決方式：

在 CAction 的 onMove 中，判斷目前動作是否有切換，並以 index 紀錄目前跑到的張數，當人物左右方向切換時，繼續另一方向的 index，使動畫連貫，而不閃爍。

問題三十五：

想在某些地方顯示類似視窗，可額外提供資訊之面板

解決方式：

實作 CWindow，並在加入 Button，使其可以被關閉，並有 isOpen 方便在面板上顯示之物件，可以和 Window 一起開啟時出現、關閉實消失。

問題三十六：

在面板中想實現滾輪捲動頁面

解決方式：

新增 CSrolling 繼承 CWindows，並將 afxwin 中的 onMouseWheel，接回 GameView 中新增狀態 OnMouseWheel，呼叫 CGame 的 static instance -> 新增的滑鼠捲動狀態，最後於 CGame 中將新增狀態 OnMouseWhell，並於 CGameState 中以 virtul 宣告，於需要的遊戲狀態，如 GameStateInit 中新增即可，並以其傳入的 short 參數，來判斷要往上還是往下捲動，以及要捲動多少。

2. 時間表

說明部分，黑色為郭宗育所做，紅色為徐紹崴所做

週次	時數 郭宗育(小時)	時數 徐紹崴(小時)	說明

1	3	3	練習 Game Framework Tutorial
2	5	4	確定分組、題目，完成 Git 練習
3	6	2	放入地圖、角色 讓角色能左右行走、添加角色動畫 當角色跨越地圖時，可以切換地圖 (如果沒有地圖可以跨越，角色會停止在屏幕邊界) 實作函式使 String 轉換成 char*
4	10	4	新增圖層概念，實作 layerManager 管理圖層 新增小怪物，並能顯示在地圖上 如果地圖的寬度大於 640，地圖具有卷軸效果 讓角色能夠跳躍
5	12	6	使角色能朝向滑鼠方向丟子彈(蔥) 實作點與矩形碰撞、矩形與矩形碰撞 當小怪物被子彈砸到時，可以獲得分數，並且小怪死亡，經

			過一段時間後重生 新增類別 CTimer，負責計時 函數化 CDC 的繪出文字
6	6	4	添加更多音效 製作 icon 更新 layerManager，令其使用更方便 修正 CTimer，令其能以毫秒為單位計時 解決轉換函式(string 轉成 char*)產生的 memory leaks
7	8	2	新增類別 BitmapPath，方便讀取動畫 實作 Boss 以及 BossManager、試放 Boss 實作 Dialog 負責儲存對話，CDialogManager，負責進行對話 能夠進行簡單的對話 製作 Dialog 文本
8	10	10	新增 Camera，解決有時候卷軸發生異常問題 能讓地圖上其他物件隨卷軸移動 以 txt 文本讀取對話，並可以

			逐字顯示在螢幕上 解決對話因為中、英文大小不依產生的亂碼問題 一段對話結束時，有提示 提醒使用者可以繼續下一句對話 更新對話背景素材
9	14	8	改寫 CAudio(將音效定義在 CAudio 內) 新增 Boss-女老師，女老師可以丟子彈(遠東單字本) 她可以丟出圓周運動的子彈 她丟出的子彈碰到角色，角色會扣血 更新碰撞函式，解決大矩形碰到小矩形時碰撞失敗的問題 實作 Npc 以及 NpcManager 可以與 Npc 進行對話 可以與另一型的 Npc 進行對話，並且聽指定的音樂(該 Npc 為一音樂播放器) 整理 RES 資料夾 將以 CDC 繪出的數字圖像化 新增 Button，方便時做兩種狀態之按扭

10	14	12	新增 3 個 NPC 完成 Boss-女老師，新增攻擊：丟黑洞，當角色碰到黑洞的時候，會被黑洞吸走，當黑洞炸裂時，會釋放 12 個子彈 女老師不能被攻擊，但時間到的時候女老師會自爆 實作 End 及 EndManager 負責控制結局，結局具有圖片+對話框 使結局的圖片能以透明效果淡入淡出 增加數字的圖片資源 實作 CAction，增加角色的動作(待機、移動、跳躍) ，使腳色跳躍、待機時能有不同方向的動畫
11	6	12	更新結局，令結局能以多組圖片+對話顯示 角色新增判定點，Boss 攻擊到角色的判定點，角色才會扣血 修正女老師不能打第二次的錯誤 新增結局：鹹魚(什麼條件都沒達成)

			修正 Camera 的錯誤 新增 1 個 NPC 新增 2 結局：打贏女老師、打輸女老師 時做視窗，使畫面可以多一個可關閉的面板，顯示戰時需要的資訊。 實作滑動視窗，滑動視窗能以滑鼠滾輪控制視窗的顯示範圍
12	6	6	解決虛擬函式產生的 memory leaks 使程式能讀取資料夾內的所有檔案，自動化載入音效 實作地圖編輯器 MapEditer，能以工具列新增地圖、在地圖上放障礙物，儲存新增的地圖 新增 Ending 視窗，以滑動視窗顯示目前擁有的結局 ButtonManager，將 Button 集中管理、將 Mouse 坐標傳入其中，將原本 Button 的更新狀態整理至其內。
13	12	3	能以滑鼠拖曳地圖編輯器上的物件

			將地圖的資訊以 txt 的方式存起來 遊戲能以 txt 的方式讀取地圖 地圖編輯器能以 txt 的方式載入地圖，並進行新增物件、儲存地圖操作 解決 buttonManager 帶來的 memory leaks 新增血條，將角色血量圖像化
14	12	4	在遊戲中新增障礙物 角色能與障礙物進行碰撞 角色能跳到障礙物上 地圖編輯器中而使用 delete 鍵刪除選取的物件 地圖素材整理、更新
15	12	3	在地圖編輯器新增/編輯完地圖時可以直接在遊戲中重新載入，不用重啟程式 修正地圖編輯器不能換背景的錯誤 地圖編輯器中可以對地圖設置上下左右方向的連結地圖 角色從障礙物上離開可以掉落 修正角色的碰撞範圍

			角色跳起來時，如果撞到障礙物，不會穿過障礙物，而會直接進行降落 對話文字可以變色 更新地圖
16	18	20	新增傳送門，能讓角色傳送到上方的地圖或下方的地圖(如果有的話) 擁有結局的選單中，點選已經擁有的結局，可以觀看結局劇情 程式會記錄擁有那些結局，不會因為關閉程式而重置 讀取資料夾內的檔案，自動化載入對話所需要的文字以及圖片 新增 6 個 Npc 新增 1 個小怪物 修正 Npc，讓 Npc 可以看向角色的方向 小怪可以與障礙物進行碰撞 小怪物從障礙物讓離開時可以掉落 增加 UIManager，控制人物以及 Boss 的血條，以及 Boss 對

			應之頭像，時間、分數的控制 也在內部做更新 地圖大翻新，更換障礙物圖片
17	48	46	新增 Boss-發財種子 發財種子會朝向角色移動，當碰到發財種子時，角色的魔力會逐漸下降，當沒有魔力時，角色會逐漸扣血 發財種子會從背上發射出大量金幣，金幣碰到角色時會使角色扣血 發財種子每隔一段時間會發射宇宙射線，宇宙射線碰到角色時會扣血 角色打贏發財種子的時候，會開啟 1 張新地圖 新增 1 個小怪物 新增可換頁視窗，用於操作說明 做報告 美化初始畫面 遊戲中新增暫停視窗，玩家在暫停視窗中可以選擇繼續玩、重新開始、回到畫面、結束，四種選項

			完成操作說明指示 做報告
合計	202	150	

3. 貢獻比例

徐紹崴 50%

郭宗育 50%

4. 自我檢核表

	項目	完成否	無法完成的原因
1	解決 Memory leak	■已完成 □未完成	
2	自定遊戲 Icon	■已完成 □未完成	
3	全螢幕啟動	■已完成 □未完成	
4	有 About 畫面	■已完成 □未完成	
5	初始畫面說明按鍵及滑鼠之用法與密技	■已完成 □未完成	
6	上傳 setup/apk/source 檔	■已完成 □未完成	
7	Setup 檔可正確執行	■已完成 □未完成	
8	報告字型、點數、對齊、行距、頁碼等格式正確	■已完成 □未完成	
9	報告封面、側邊格式正確	■已完成 □未完成	
10	報告附錄程式格式正確	■已完成 □未完成	

5. 收穫

郭宗育	徐紹崴
Visual Studio 中斷點除錯功能	以呼叫堆疊、中斷點除錯
C++ 物件使用概念	C++ 物件多型、繼承、封裝
C++ 物件繼承概念	C++ 虛擬函式
C++ fstream 檔案讀寫	C++ friend class
C++ stringstrream 字串符流	C++ namespace 之應用
C++ STL map、vector	C++ STL map、vector
C++ static 概念	C++ Static 物件
MFC CopyFile、DeleteFile	C++ Manger 管理物件
MFC FileDialog	C++ Copy Constructor
Dirent 資料夾操作	MFC afxwin api
自我學習	時間規劃

6. 心得、感想

郭宗育：

不知不覺間，已經到學期末了，憶起剛開學時知道要以遊戲來練習物件導向，那場景是歷歷在目，恍如昨日，在這十七週中，走過找一堆 bug 的日子，走過跟框架大眼瞪小眼的日子，走過爬 MFC 參考文件的日子，走過做各種神奇管理器的日子，走過用 fstream 檔案讀寫存一堆 txt，寫一堆 txt 的日子，走過搞圖像化編輯器的日子，我想就算是我，這樣一路走來，多少也有進步一些吧？

記得剛開學時，陰錯陽差之下決定了以 RPG 為主軸，做出一個能在限制時間內遊玩，並根據遊玩內容獲取不同結局的遊戲，剛開始也沒多想，就想說先讓角色能移動、跳躍、丟子彈吧，之後又搞出了小怪物，讓角色丟的子彈可以消滅小怪物，而且小怪物可以重生，另外，也為了方便顯示圖片方便，導入圖層概念，並做出了圖層管理器——之後，第一次 demo 到了。

第一次 demo 的結果老實講，真的很令我訝異，我本來想說做到這樣頂多 B+吧，沒想到拿到了 A-的成績，讓我很是驚喜，於是在第一次 demo 後，我帶著愉悅的心情，做了整個程式我最滿意的東西——對話

與對話管理器，此外，為了合理使用對話，我製作了 NPC，一種 NPC 是一般的看劇情的 NPC，而另一種 NPC，則是可以播放音樂的音樂播放器。同時，我也進行 Boss 的製作，並做出結局，讓玩家打贏 Boss、被 Boss 打死、或是什麼事情都不做的時候，都能進入相應的結局。本來我以為做了這些東西，第二次 demo 應該起碼能苟個 A 吧？然而事情卻不像我想的一樣順利——第二次 demo 一樣只拿了 A-，老實講，在當時我不太能接受這件事情，當我看到成績的時候，氣到直接去打麻將。

但總是有個原因使我認為有 A 的遊戲變成 A-，如果不找出這個原因，那期末 demo 一樣不會有好成績，在我經過多方詢問下，我歸納出幾個可能導致被判定 A-的原因：

一、UI 做的不夠好，像是角色的血量不是以血條方式呈現

二、自認做的最好的對話，不被認為是以 fstream 讀檔，而是在程式中硬設字串，或是讀取圖片

三、我報告講太爛

沒想到我認為報告不必講技術層面這種觀念會給我帶來如此大的影響，倒也好解決，在期末 demo 的時候預計由組員出馬，由他來 cue 我講技術層面的東西，至於第一個問題也由我組員做 UI 介面解決，而我也決定，前兩次 demo 我做的都是看不到的東西(管理器)，第三次，我要做看的到的東西。

因此，地圖編輯器誕生了，他的出發點來自於開學的時候偉凱老師曾經問的一句話「你們的遊戲會有障礙物嗎？」當時我回答了「可以做阿」，我才想到這遊戲是不是該放障礙物上去了，但如果改變障礙物的位置的時候都要去程式裡改就不是好的方法，因此我希望能提供一個可以讀取圖片做為障礙物，並有滑鼠拖曳操作改變障礙物座標，還能存檔成地圖的格式，而這，就是地圖編輯器。雖然還想幫地圖編輯器擴充其

他功能，但無奈時間不太夠。

有了障礙物之後，就要與障礙物碰撞，能站在障礙物上，也能從上面掉下來，所幸高職的畢業專題有做過類似的東西，還算過得去。

這樣回首過來，我們好像做了很多新功能，但這些東西似乎又不符合本堂課程的方向了，令我有些尷尬，而這又是另外一個故事了。

總而言之，課程進展到現在，再過幾天就要期末 demo 了，我想我也只能盡人事、聽天命，而在這邊，我一定要感謝我的組員好朋友，鼎力相助各種素材、提供各種想法、肩負除錯還要做新 UI 的任務，還是個創建地圖大師，我想沒有他，我現在還在用超醜的圖片當人物動畫。

徐紹崴：

時間一晃也已到了期末，憶起當初前兩個禮拜從徬徨的尋找主題，到終於找到方向；又毫無頭緒地開始建置人物，到在啥都沒有的地圖上開始讓主角動起來；又從只能走動的人物，一步步升級至能跳躍、丟投置物攻擊的主角，單單那麼簡單的幾件事，卻在一開始帶來給我繼續做這個遊戲下去的動力；即使現在回首，無論素材、程式都以是不堪回首的過去，但想想就這短短的十幾周，我想自己還是有些長進的吧。
我們在沒有周延的考量下，選擇了小品的 RPG 作為遊戲，起初的想法是，既然我們一學期的時間，那不如限制遊戲的時間，但以不同的條件來觸發不同的結局，在有效限制完加活動範圍的同時，仍不失遊戲的耐完性，而後也新增了卷軸地圖，以及小怪可以擊殺賺取積分，就這麼到了第一次 demo，原以為這樣的遊戲會部會還是太單薄，沒想到還是拿到 A-的成績，比我們預期的略高一點，但坦講白就是比上不足，比下有餘。

而後我們便開始著手讓遊戲變更完整，從可以對話的 NPC，到 Boss，乃至結局，而 NPC 對話中的字更是使用讀取文檔的方式，且能逐字顯

示，於是抱著既期待又害派傷害的心情，我們結束了第二次 Demo，而生硬的事實卻唐突得打亂我們的步調，我們自認為完整的遊戲架構，卻仍只拿到了 A-的成績，我們不服氣的在事後詢問助教，才得知我們失意之因，首先是缺少圖像化的介面，雖然有量化的血顯示在一旁，但確實因為當時無暇注意，而忽略了他其實挺不顯眼的事實，這確實是我們要檢討的地方；其二，則是報告時沒有準備搞，導致我們前一天討論的重點，並沒有全部都在報告上說出來，也造成了可能因報告太快，也沒特別提及，使大家無從得知對話是使用讀檔，且逐字顯示的功能，這也確實是我們的疏失，不該怪罪任何人，所以我們也決議，少做點看不到管理器，多做點看的到的東西，這也是這次新增 UI 管理器來方便管理 UI 的原因。

而之後，我們由於偉凱老師的提醒，終於開始做起了地圖，這裡必須感謝隊友的強大讀寫檔能力，以及自行在 MFC 架構上架出地圖編輯器，使我們之後對地圖的修改，著實方便、有效率續許多，也新增了更多「看的見」的東西，如地圖障礙物、血量條、暫停介面…等。

總歸來說，這學期的收穫確實不少，要再次感謝強大隊友的幫助，從地圖控管，到小怪、Boss 基本上都是他一手包辦，模組化的管理也使我學習到非常多觀念，我想這就是團隊作業的好處吧。

五、 附錄

1. mygame.h

```
/*
 * mygame.h: 本檔案儲遊戲本身的 class 的 interface
 * Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>
 *
 * This file is part of game, a free game development framework for windows.
 *
 * game is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * game is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 *
 *      2004-03-02 V4.0
 *        1. Add CGameStateInit, CGameStateRun, and CGameStateOver to
 *           demonstrate the use of states.
 *    2005-09-13
 *        Rewrite the codes for CBall and CEraser.
 *    2005-09-20 V4.2Beta1.
 *    2005-09-29 V4.2Beta2.
 *    2006-02-08 V4.2
 *        1. Rename OnInitialUpdate() -> OnInit().
 *        2. Replace AUDIO_CANYON as AUDIO_NTUT.
 *        3. Add help bitmap to CGameStateRun.
 *    2006-09-09 V4.3
 *        1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they are
 *           event driven.
 *    2008-02-15 V4.4
 *        1. Add namespace game_framework.
 *        2. Replace the demonstration of animation as a new bouncing ball.
 *        3. Use ShowInitProgress(percent) to display loading progress.
*/
#pragma once
#include "CEraser.h"
#include "CBall.h"
#include "CManager.h"
#include "CBoss.h"
#include "Refactor.h"
namespace game_framework {
  class CGameStateInit : public CGameState {
    public:
      CGameStateInit(CGame* g);
      ~CGameStateInit();
      void OnBeginState(); // 設定每次重玩所需的變數
      void OnInit(); // 遊戲的初值及圖形設定
      void OnKeyDown(UINT, UINT, UINT); // 處理鍵盤 Down 的動作
      void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤 Up 的動作
      void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作
      void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作
      void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作
      void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作
      void OnMouseWheel(UINT nFlags, short zDelta, CPoint point);
    protected:
      void OnMove();
      void OnShow(); // 顯示這個狀態的遊戲畫面
    private:
```

```
    CButtonManager buttonManager;
    CPoint mouse;
    CMovingBitmap background; // background
    CRole miku;
    CScrollWindow windowsEnding;
    CSwitchWindow windowsHandbook;
};

class CGameStateRun : public CGameState {
    friend class CEventManager;
  public:
    CGameStateRun(CGame* g);
    ~CGameStateRun();
    void OnBeginState(); // 設定每次重玩所需的變數
    void OnInit(); // 遊戲的初值及圖形設定
    void OnKeyDown(UINT, UINT, UINT);
    void OnKeyUp(UINT, UINT, UINT);
    void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnRButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnMouseWheel(UINT nFlags, short zDelta, CPoint point);
  protected:
    void OnMove(); // 移動遊戲元素
    void OnShow(); // 顯示這個狀態的遊戲畫面
  private:
    CRole role;
    CInteger time_left;     // 剩下的撞擊數
    CTimer timer; //stateRun 狀態下的 timer (其實好像在什麼狀態下都無所謂) (gameRun timer)
    CTimer* nowUsedTimer; //現在被使用的計時器 (BOSS 戰鬥的時候 原本的遊戲計時器不會倒數)
    vector<CScallion*>* scallions;
    vector<CMonster*>* passerbys;
    CMapManager mapManager;
    CBossManager bossManager;
    CNPCManager npcManager;
    CEventManager eventManager;
    CStatusBoard roleStatus;
    CBossBoard bossStatus;
    UIManager;
    CPanel panel;
    bool isWinXingting;
    bool isWinFacaiSeed;
    #pragma region - 自定義函數 -
    void PositionTrigger();
    void ChangeMap(string);
    void ChangeMap(int);
    void SwitchState(int);
    void SwitchTimer(CTimer*);
    void GoToEnd();
    #pragma endregion
};

class CGameStateOver : public CGameState {
  public:
    CGameStateOver(CGame* g);
    void OnBeginState(); // 設定每次重玩所需的變數
    void OnInit();
    void OnKeyDown(UINT, UINT, UINT);
    void OnLButtonDown(UINT nFlags, CPoint point);
  protected:
    void OnMove(); // 移動遊戲元素
    void OnShow(); // 顯示這個狀態的遊戲畫面
  private:
    int counter; // 倒數之計數器
    int alpha;
    bool canSwitchState = false;
    bool canDrawGameOverImage = false;
    CTimer timer_exit; //結局結束後要多久時間離開 GameOver
```

```
    CMovingBitmap overBitmap;
    CToumeiImage gameOverImage;
    CPanel panel;
};

class CGameStateMapEditer : public CGameState
{
  public:
    CGameStateMapEditer(CGame* g);
    ~CGameStateMapEditer();
    void OnInit(); // 遊戲的初值及圖形設定
    void OnBeginState(); // 設定每次重玩所需的變數
    void OnKeyDown(UINT, UINT, UINT);
    void OnKeyUp(UINT, UINT, UINT);
    void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作
    void OnRButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作
  protected:
    void OnMove();
    void OnShow();
  private:
    CMapEditer mapEditer;
    CPoint mousePoint;
};
```

2. mygame.cpp

```
/*
* mygame.cpp: 本檔案儲遊戲本身的 class 的 implementation
* Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>
*
* This file is part of game, a free game development framework for windows.
*
* game is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License as published by
* the Free Software Foundation; either version 2 of the License, or
* (at your option) any later version.
*
* game is distributed in the hope that it will be useful,
* but WITHOUT ANY WARRANTY; without even the implied warranty of
* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
* GNU General Public License for more details.
*
* You should have received a copy of the GNU General Public License
* along with this program; if not, write to the Free Software
* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
*
* History:
*   2002-03-04 V3.1
*           Add codes to demostrate the use of CMovingBitmap::ShowBitmap(CMovingBitmap &).
*   2004-03-02 V4.0
*     1. Add CGameStateInit, CGameStateRun, and CGameStateOver to
*        demonstrate the use of states.
*     2. Demo the use of CInteger in CGameStateRun.
*   2005-09-13
*     Rewrite the codes for CBall and Crole.
*   2005-09-20 V4.2Beta1.
*   2005-09-29 V4.2Beta2.
*     1. Add codes to display IDC_GAMECURSOR in GameStateRun.
*   2006-02-08 V4.2
*     1. Revise sample screens to display in English only.
*     2. Add code in CGameStateInit to demo the use of PostQuitMessage().
*     3. Rename OnInitialUpdate() -> OnInit().
*     4. Fix the bug that OnBeginState() of GameStateInit is not called.
*     5. Replace AUDIO_CANYON as AUDIO_NTUT.
*     6. Add help bitmap to CGameStateRun.
*   2006-09-09 V4.3
```

```
*           1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they are
*              event driven.
*     2006-12-30
*           1. Bug fix: fix a memory leak problem by replacing PostQuitMessage(0) as
*              PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE,0,0).
*     2008-02-15 V4.4
*           1. Add namespace game_framework.
*           2. Replace the demonstration of animation as a new bouncing ball.
*           3. Use ShowInitProgress(percent) to display loading progress.
*     2010-03-23 V4.6
*           1. Demo MP3 support: use lake.mp3 to replace lake.wav.
*/
#pragma once
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"
#include "CManager.h"
#include "CBall.h"
#include "CLibrary.h"
#include <string>
#include <fstream>
#include <vector>
using namespace std;
using namespace myLibrary;
namespace game_framework {
   #pragma region - CGame Init -
   CGameStateInit::CGameStateInit(CGame* g) : CGameState(g)
   {
      #pragma region - Button Create -
      buttonManager.CreateButton(BitmapPath("RES\\Button", "music", 2, RGB(214, 214, 214)), CPoint(250, 420), true,
false);
      buttonManager.CreateButton(BitmapPath("RES\\Button", "sound", 2, RGB(214, 214, 214)), CPoint(450, 420), true,
false);
      buttonManager.CreateButton(BitmapPath("RES\\Button", "play", 2, RGB(214, 214, 214)), CPoint(230, 200), false,
true);
      buttonManager.CreateButton(BitmapPath("RES\\Button", "ending", 2, RGB(214, 214, 214)), CPoint(230, 270), false,
true);
      buttonManager.CreateButton(BitmapPath("RES\\Button", "tutorial", 2, RGB(214, 214, 214)), CPoint(230, 340),
false, true);
      #pragma endregion
      #pragma region - reset -
      DeleteFile("RES\\Map\\FileName.txt");
      DeleteFile("RES\\Map\\ReloadMapInformation.txt");
      #pragma endregion
   }

   CGameStateInit::~CGameStateInit()
   {
      buttonManager.Clear();
   }

   void CGameStateInit::OnBeginState()
   {
      static bool isLoaded = false;
      CLayerManager::Instance()->Initialize();
      CEndManager::Instance()->Initialize();
      CAudio::Instance()->Initialize();
      CAudio::Instance()->Play("SLoWMoTIoN_Menu", true);
      if (!isLoaded)
      {
         buttonManager.Load();
         windowsEnding.LoadResource();
         windowsHandbook.LoadResource("RES\\Handbook\\introduction\\");
         miku.LoadasMascot();
      }
```

```
    buttonManager.Initialize();
    buttonManager.SetValid(true);
    windowsEnding.Initialize(CPoint(0, 0));
    windowsHandbook.Initialize(CPoint(0, 0));
    miku.Initialize();
    miku.SetXY(50, 320);
    miku.GetAction()->SetFaceTo("_R");
    isLoaded = true;
}

void CGameStateInit::OnInit()
{
    ShowInitProgress(0); // 一開始的 loading 進度為 0%
    background.LoadBitmap(".\\RES\\Map\\Menu_3.bmp");
}

void CGameStateInit::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    if (nChar == 'N')
    {
        GotoGameState(GAME_STATE_MAPEDITER);
    }
}

void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    const char KEY_ESC = 27;
    if (nChar == KEY_SPACE)
    {
        GotoGameState(GAME_STATE_RUN); // 切換至 GAME_STATE_RUN
    }
    else if (nChar == KEY_ESC)
    {
        if (windowsEnding.IsOpen())
        {
            windowsEnding.Close();
        }
        else if (windowsHandbook.IsOpen())
        {
            windowsHandbook.Close();
        }
        else
        {
            PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0); // 關閉遊戲
        }
    }
}

void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point)
{
    if (windowsEnding.IsCollisionClose(point))
    {
        windowsEnding.Close();
    }
    else if (windowsHandbook.IsCollisionClose(point))
    {
        windowsHandbook.Close();
    }
    else if (windowsEnding.IsOpen())
    {
        string endName = windowsEnding.GetCollisionButtonName(point);
        if (endName != "NoButtonClick") //有點到結局
        {
            CAudio::Instance()->Stop("SLoWMoTIoN_Menu");
            CEndManager::Instance()->Start(endName);
            GotoGameState(GAME_STATE_OVER);
        }
    }
    else if (windowsHandbook.IsOpen())
```

```
    {
        CAudio::Instance()->Play("page_next");
        windowsHandbook.ClickWindows(point, "left");
    }
    else if (buttonManager.GetCollisionButtonName() != "NoButtonClick")
    {
        string btnName = buttonManager.GetCollisionButtonName();
        CAudio::Instance()->Play("btn_click_3");
        if (btnName == "music")
        {
            buttonManager.ClickButton("music");
            CAudio::Instance()->SetIsPlayMusic(buttonManager.GetState("music"));
        }
        else if (btnName == "sound")
        {
            buttonManager.ClickButton("sound");
            CAudio::Instance()->SetIsPlaySound(buttonManager.GetState("sound"));
        }
        else if (btnName == "ending")
        {
            windowsEnding.Open();
        }
        else if (btnName == "play")
        {
            GotoGameState(GAME_STATE_RUN);  // 切換至 GAME_STATE_RUN
        }
        else if (btnName == "tutorial")
        {
            windowsHandbook.Open();
        }
    }
}

void CGameStateInit::OnLButtonUp(UINT nFlags, CPoint point)        // 處理滑鼠的動作
{
}

void CGameStateInit::OnRButtonDown(UINT nFlags, CPoint point)
{
    if (windowsHandbook.IsOpen())
    {
        CAudio::Instance()->Play("page_next");
        windowsHandbook.ClickWindows(point, "right");
    }
}

void CGameStateInit::OnMouseMove(UINT nFlags, CPoint point)        // 處理滑鼠的動作
{
    if (windowsEnding.IsOpen())
    {
        windowsEnding.CollisionClose(point);
        return;
    }
    else if (windowsHandbook.IsOpen())
    {
        windowsHandbook.CollisionClose(point);
        return;
    }
    buttonManager.UpdateState(point);
}

void CGameStateInit::OnMouseWheel(UINT nFlags, short zDelta, CPoint point)
{
    //往使用者方向滾 -> 往下捲動 (圖片 y -= dy)
    //遠離 使用者方向滾 -> 往上捲動 (圖片 y += dy)
    windowsEnding.OnScrolling(zDelta);
}

void CGameStateInit::OnMove()
```

```
{
    SetCursor(AfxGetApp()->LoadCursor(IDC_GAMECURSOR));
    #pragma region mute music with menu_bgm
    if (!buttonManager.GetState("music"))  //music mute
    {
        CAudio::Instance()->Pause();
    }
    else
    {
        CAudio::Instance()->Resume();
    }
    #pragma endregion
    background.SetTopLeft(0, 0);
    buttonManager.OnCycle();
    buttonManager.SetValid(!(windowsEnding.IsOpen() || windowsHandbook.IsOpen()));
    windowsEnding.OnCycle();
    windowsHandbook.OnCycle();
    miku.GetAction()->OnMove("run");
}

void CGameStateInit::OnShow()
{
    background.ShowBitmap();
    CLayerManager::Instance()->ShowLayer();
    windowsEnding.OnShow();
}

#pragma endregion
#pragma region - CGame Run -
CGameStateRun::CGameStateRun(CGame* g)
    : CGameState(g)
{
    #pragma region - Set eventManager -
    eventManager.SetGameStateRun(this);
    #pragma endregion
    panel.CreatButton();
}

CGameStateRun::~CGameStateRun()
{
    npcManager.Clear();
    bossManager.Clear();
    panel.Clear();
}

void CGameStateRun::OnBeginState()
{
    CLayerManager::Instance()->Initialize();
    CDialogManager::Instance()->Initialize();
    CCamera::Instance()->Initialize();
    mapManager.Initialize();
    bossManager.Initialize();
    npcManager.Initialize(mapManager.GetNowMap()); //npcManager 初始化所有 NPC 的同時，顯示 nowMap 上的
NPC
    eventManager.Initialize();
    role.Initialize();
    uiManager.Initialize(&role, &bossManager);
    panel.Initialize(CPoint(200, 50));
    timer = CTimer(GAME_TIME);
    nowUsedTimer = &timer;
    CAudio::Instance()->Stop("SLoWMoTIoN_Menu");
    CAudio::Instance()->Play("SLoWMoTIoN_Game");
    isWinXingting = false;
    isWinFacaiSeed = false;
}

void CGameStateRun::OnInit() //  遊戲的初值及圖形設定
{
    ShowInitProgress(33);   // 接個前一個狀態的進度，此處進度視為 33%
```

```
    ShowInitProgress(50); // 載入圖形
    #pragma region - Initialize - MapManager -
    mapManager.LoadMapBitmap();
    #pragma endregion
    #pragma region 載入角色
    role.Load();
    #pragma endregion
    uiManager.Load();
    panel.LoadResource();
}

void CGameStateRun::OnMove() // 移動遊戲元素
{
    SetCursor(AfxGetApp()->LoadCursor(IDC_GAMECURSOR));
    #pragma region - on panel -
    panel.OnCycle();
    if (panel.IsOpen())
    {
        return;
    }
    #pragma endregion
    #pragma region - pause game state in dialoging -
    if (CDialogManager::Instance()->GetDialogState())
    {
        CDialogManager::Instance()->OnCycle();
        return;
    }
    #pragma endregion
    #pragma region - timer count down -
    nowUsedTimer->CountDown();
    if (timer.IsTimeOut())
    {
        CAudio::Instance()->Stop("SLoWMoTIoN_Game");
        GoToEnd();
        SwitchState(GAME_STATE_OVER);
    }
    #pragma endregion
    #pragma region - run boss endprocess -
    if (bossManager.targetBoss != NULL && bossManager.targetBoss->InEndProcess())
    {
        bossManager.targetBoss->EndProcess();
        return;
    }
    #pragma endregion
    bool roleCanMoving = true;
    #pragma region - role - dead -
    if (role.IsDead())
    {
        if (bossManager.targetBoss != NULL)
        {
            if (bossManager.targetBoss->GetBossId() == "Xingting")
            {
                CEndManager::Instance()->Start(END_NAME_LOSEXINGTING);
                role.SetXY(0, 0);
                SwitchState(GAME_STATE_OVER);
                return;
            }
            if (bossManager.targetBoss->GetBossId() == BOSS_FACAISEED)
            {
                CEndManager::Instance()->Start(END_NAME_LOSE_FACAISEED);
                role.SetXY(640, 0);
                SwitchState(GAME_STATE_OVER);
                return;
            }
        }
        else
        {
            CEndManager::Instance()->Start(END_NAME_SALTEDFISH);
            role.SetXY(0, 0);
```

```
            SwitchState(GAME_STATE_OVER);
            return;
        }
    }
    #pragma endregion
    #pragma region - boos - dead -
    if (bossManager.targetBoss != NULL && bossManager.targetBoss->IsDead())
    {
        if (bossManager.targetBoss->GetBossId() == "Xingting")
        {
            isWinXingting = true;
        }
        if (bossManager.targetBoss->GetBossId() == BOSS_FACAISEED)
        {
            isWinFacaiSeed = true;
        }
        bossManager.BossDead();
        SwitchTimer(&timer);
    }
    #pragma endregion
    #pragma region - end battle -
    //boss 沒死 但達成其他終結 battle 條件
    if (bossManager.targetBoss != NULL && bossManager.targetBoss->IsEnd()) //有 Boss 且 Boss 終結
    {
        bossManager.BossDead();
        SwitchTimer(&timer);
    }
    #pragma endregion
    #pragma region - Position Trigger -
    PositionTrigger();
    #pragma endregion
    #pragma region - Moving -
    int screenPosX = ScreenX(mapManager.GetX1(), role.GetX3());
    #pragma region SetCamera
    int mapR = mapManager.GetSplitRight();
    int mapL = mapManager.GetSplitLeft();
    int CameraBoundary_R = (mapManager.GetBitmapWidth() / 2 - mapL) + (mapR - mapManager.GetBitmapWidth() /
2);//鏡頭右邊界
    int CameraBoundary_L = -CameraBoundary_R; //鏡頭左邊界
    CCamera::Instance()->SetCanMoving(true);
    #pragma endregion
    #pragma region --- role - collision with boss ---
    if (bossManager.targetBoss != NULL)
    {
        if (bossManager.targetBoss != NULL)
        {
            if (role.IsCollisionBoss(bossManager.targetBoss)) //collision with boss
            {
                if (bossManager.targetBoss->GetID() != "Xingting" && bossManager.targetBoss->GetID() != "FacaiSeed")
                {
                    CCamera::Instance()->SetCanMoving(false);
                    roleCanMoving = false;
                }
            }
        }
    }
    #pragma endregion
    #pragma region - role - collision with block -
    vector<CBlock>* block = mapManager.GetBlockVector();
    bool isOnFloor = false;
    int floorY = SIZE_Y;
    for (vector<CBlock>::iterator bkiter = block->begin(); bkiter != block->end(); bkiter++)
    {
        if (role.IsCollisionBlock(&(*bkiter))) //collision with block
        {
            CCamera::Instance()->SetCanMoving(false);
            roleCanMoving = false;
        }
        if (role.IsRoleOnBlock(&(*bkiter)) && role.GetDrop())
```

```
            {
                floorY = bkiter->y;
                isOnFloor = true;
            }
            if (role.IsCollisionBlockOnJumping(&(*bkiter)) && !role.GetDrop())
            {
                role.SetDrop();
            }
        }
        if (role.GetY2() >= SIZE_Y - 1)
        {
            floorY = SIZE_Y - 1;
            isOnFloor = true;
        }
        #pragma endregion
        #pragma region -- Moving Left --
        if (role.GetMovingLeft())
        {
            if (role.GetX3() <= SIZE_X / 2)
            {
                if (screenPosX > mapL)
                {
                    CCamera::Instance()->AddX(-MOVE_DISTANCE);
                    roleCanMoving = false;
                }
                if (bossManager.targetBoss != NULL)
                {
                    bossManager.targetBoss->OnMove();
                }
            }
        }
        #pragma endregion
        #pragma region -- Moving Right --
        if (role.GetMovingRight())
        {
            if (role.GetX3() >= SIZE_X / 2)
            {
                if (screenPosX < mapR)
                {
                    CCamera::Instance()->AddX(MOVE_DISTANCE);
                    //role.SetCanMoving(false);
                    roleCanMoving = false;
                }
            }
        }
        #pragma endregion
        mapManager.OnMove();
        #pragma region -- 左右界地圖檢查 - 卡邊界 or 換地圖 --
        #pragma region --- 右邊有地圖且人物往右邊行走 ---
        if (role.GetX2() >= SIZE_X && mapManager.GetRightMap() >= 0 && role.GetMovingRight())
        {
            #pragma region ---- 超過邊界 - 換地圖 ----
            if (role.GetX1() >= SIZE_X)
            {
                ChangeMap("right");
            }
            #pragma endregion
        }
        #pragma endregion
        #pragma region --- 右邊沒有地圖且人物往右邊行走 ---
        if (role.GetX2() >= SIZE_X && role.GetMovingRight() && (mapManager.GetRightMap() < 0 ||
bossManager.IsBattle()))
        {
            roleCanMoving = false;
        }
        #pragma endregion
        #pragma region --- 左邊有地圖且人物往左邊行走 ---
        if (role.GetX1() <= 0 && mapManager.GetLeftMap() >= 0 && role.GetMovingLeft())
        {
```

```
    #pragma region ---- 超過邊界 - 換地圖 ----
    if (role.GetX2() <= 0) //超過邊界，換地圖
    {
        ChangeMap("left");
    }
    #pragma endregion
}
#pragma endregion
#pragma region --- 左邊沒有地圖且人物往左邊行走 ---
if (role.GetX1() <= 0 && role.GetMovingLeft() && (mapManager.GetLeftMap() < 0 || bossManager.IsBattle()))
{
    roleCanMoving = false;
}
#pragma endregion
#pragma endregion
#pragma region -- role Moving --
role.SetCanMoving(roleCanMoving);
role.OnMove();
#pragma endregion
#pragma endregion
#pragma region - Jump -
bool isDrop = false;
if (role.GetMovingJump())
{
    if ((role.GetY2() - role.GetVelocity() >= SIZE_Y || isOnFloor) && role.GetDrop()) //on Floor
    {
        int ddy = 2;
        role.SetMovingJump(false);
        role.SetCanJumping(true);
        role.GetAction()->SetAction("idle");
        role.SetXY(role.GetX1(), floorY - role.Height() + ddy);
    }
    else //Jumping
    {
        role.SetMovingJump(true);
        role.SetCanJumping(false);
    }
}
else
{
    if (!isOnFloor)
    {
        role.SetDrop();
    }
}
#pragma endregion
#pragma region - boss cycle (move and attack) -
if (bossManager.targetBoss != NULL && bossManager.IsBattle())
{
    bossManager.targetBoss->OnCycle(&role);
}
#pragma endregion
#pragma region - Collision -
scallions = role.GetScallion();    //取出蔥的指標做碰撞
passerbys = mapManager.GetPasserby(); //取出 passerby 指標碰撞
for (vector<CScallion*>::iterator scallionk = scallions->begin(); scallionk != scallions->end(); )
{
    for (vector<CMonster*>::iterator passerbyj = passerbys->begin(); passerbyj != passerbys->end(); )
    {
        if ((*scallionk)->IsCollision(*passerbyj) && (*passerbyj)->GetValid() && (*scallionk)->IsAlive())
        {
            CAudio::Instance()->Play("hit");
            role.AddScore((*passerbyj)->GetScore());
            (*scallionk)->SetIsAlive(false);
            mapManager.DeletePasserby(passerbyj); //從 mapManager 的 passerbyManager 中移除 passerby
            mapManager.AddPasserby();
            break;
        }
        else
```

```
            {
                passerbyj++;
            }
        }
        if (!(*scallionk)->IsAlive())
        {
            delete *scallionk;
            *scallionk = NULL;
            scallionk = scallions->erase(scallionk);
        }
        else
        {
            scallionk++;
        }
    }
    #pragma endregion
    #pragma region - Passerby Moving -
    for (vector<CMonster*>::iterator passerbyj = passerbys->begin(); passerbyj != passerbys->end(); passerbyj++)
    {
        (*passerbyj)->SetMoving();
        (*passerbyj)->OnMove();
    }
    #pragma endregion
    #pragma region - NPC Moving -
    npcManager.OnCycle(mapManager.GetNowMap(), CPoint(screenPosX, role.GetY1()));
    #pragma endregion
    uiManager.OnCycle(nowUsedTimer->GetTime(1));
    panel.OnCycle();
}

void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    if (nChar == KEY_A)
        role.SetMovingLeft(true);
    if (nChar == KEY_D)
        role.SetMovingRight(true);
    if (nChar == KEY_W)
    {
        bool canRoleJump = true;
        for (vector<CBlock>::iterator bkiter = mapManager.GetBlockVector()->begin(); bkiter !=
mapManager.GetBlockVector()->end(); bkiter++)
        {
            if (role.GetCanJumping())
            {
                if (role.IsCollisionBlockOnJumping(&(*bkiter)))
                {
                    canRoleJump = false;
                }
            }
        }
        if (canRoleJump)
            role.SetMovingUp(true);
    }
    if (nChar == 'U')
    {
        nowUsedTimer->ResetTime(5.0);
    }
    if (nChar == 'T')
    {
        for (int i = 0; i < 30; i++)
        {
            role.SubHp();
        }
    }
    if (nChar == 'Y')
    {
        if (CDialogManager::Instance()->GetDialogState())
        {
            CDialogManager::Instance()->Stop();
```

```
        }
    }
    if (nChar == 27)  //ESC
    {
        if (panel.IsOpen())
            panel.Close();
        else
        {
            //Audio::Instance()->Play("open_window");
            panel.Open();
        }
    }
    if (nChar == KEY_Z) //dialog with npc
    {
        vector<CNPC*>* npc = npcManager.GetNpc(mapManager.GetNowMap());
        for (vector<CNPC*>::iterator npciter = npc->begin(); npciter != npc->end(); npciter++)
        {
            if (role.IsCollisionNPC(*npciter))
            {
                (*npciter)->RoleCollision();
                (*npciter)->SetTalked(true);
            }
        }
    }
    if (nChar == ' ') //open door
    {
        #pragma region - role - collision with door -
        vector<CDoor>* door = mapManager.GetDoorVector();
        for (vector<CDoor>::iterator bkiter = door->begin(); bkiter != door->end(); bkiter++)
        {
            if (role.IsCollisionDoor(&(*bkiter))) //collision with door
            {
                CAudio::Instance()->Play("door");
                ChangeMap(bkiter->GetSwitchMapIndex());
            }
        }
        #pragma endregion
    }
    if (nChar == 'K')
    {
        role.SetRoleNoSubHp();
    }
}

void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    if (nChar == KEY_A)
        role.SetMovingLeft(false);

    if (nChar == KEY_D)
        role.SetMovingRight(false);

    if (nChar == KEY_W)
        role.SetMovingUp(false);
}

void CGameStateRun::OnLButtonDown(UINT nFlags, CPoint point)    // 處理滑鼠的動作
{
    if (panel.IsOpen() && panel.IsCollisionClose(point))
    {
        panel.Close();
        return;
    }
    role.SetIsFire(true & !CDialogManager::Instance()->GetDialogState());

    if (CDialogManager::Instance()->GetDialogState())
    {
        CDialogManager::Instance()->Next();
    }
```

```
}

void CGameStateRun::OnLButtonUp(UINT nFlags, CPoint point)        // 處理滑鼠的動作
{
  role.SetIsFire(false);
  if (!panel.IsOpen())
  {
    return;
  }
  if (panel.GetCollisionButtonName() != "NoButtonClick")
  {
    string btnName = panel.GetCollisionButtonName();
    CAudio::Instance()->Play("btn_click_3");
    if (btnName == "resume")
    {
      panel.Close();
    }
    else if (btnName == "restart")
    {
      panel.Close();
      if (CDialogManager::Instance()->GetDialogState())
      {
        CDialogManager::Instance()->Stop();
      }
      GotoGameState(GAME_STATE_RUN);
    }
    else if (btnName == "menu")
    {
      panel.Close();

      if (CDialogManager::Instance()->GetDialogState())
      {
        CDialogManager::Instance()->Stop();
      }
      GotoGameState(GAME_STATE_INIT);   // 切換至 GAME_STATE_RUN
    }
    else if (btnName == "exit")
    {
      panel.Clear();
      PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
    }
  }
}

void CGameStateRun::OnMouseMove(UINT nFlags, CPoint point)       // 處理滑鼠的動作
{
  role.SetMouseXY(point.x, point.y);
  if (panel.IsOpen())
  {
    panel.CollisionClose(point);
    panel.UpdateMouse(point);
  }
}

void CGameStateRun::OnRButtonDown(UINT nFlags, CPoint point)   // 處理滑鼠的動作
{
}

void CGameStateRun::OnRButtonUp(UINT nFlags, CPoint point)        // 處理滑鼠的動作
{
}

void CGameStateRun::OnMouseWheel(UINT nFlags, short zDelta, CPoint point)
{
  if (zDelta > 0)
  {
    role.AddScore(5);
  }
  else if (zDelta < 0)
```

```
    {
        role.AddScore(-5);
    }
}

void CGameStateRun::PositionTrigger()
{
    eventManager.trigger();
}

void CGameStateRun::ChangeMap(string dir)
{
    int nowMap = mapManager.GetNowMap(), nextMap = nowMap;
    if (dir == "left")
    {
        nextMap = mapManager.GetLeftMap();
        mapManager.ChangeMap(mapManager.GetLeftMap(), "left");
        role.SetXY(SIZE_X - (role.GetX3() - role.GetX1()), role.GetY1() - 1);
    }
    else if (dir == "right")
    {
        nextMap = mapManager.GetRightMap();
        mapManager.ChangeMap(mapManager.GetRightMap(), "right");
        role.SetXY(0 - (role.GetX3() - role.GetX1()), role.GetY1() - 1);
    }
    bossManager.TargetBoss(mapManager.GetNowMap());
    npcManager.ChangeMap(nowMap, nextMap);
}

void CGameStateRun::ChangeMap(int nextMap)
{
    int nowMap = mapManager.GetNowMap();
    if (nextMap != -1 && nextMap < mapManager.GetBlockMapSize()) //nextMap in blockMap
    {
        mapManager.ChangeMap(nextMap, "right");
        bossManager.TargetBoss(mapManager.GetNowMap());
        npcManager.ChangeMap(nowMap, nextMap);
        role.SetXY(0, SIZE_Y - role.Height());
    }
}

void CGameStateRun::SwitchState(int state)
{
    if (bossManager.targetBoss != NULL)
    {
        bossManager.targetBoss->Clear();
        bossManager.targetBoss = NULL;
    }
    this->GotoGameState(state);
}

void CGameStateRun::SwitchTimer(CTimer* swtimer)
{
    nowUsedTimer = swtimer;
}

void CGameStateRun::GoToEnd()
{
    #pragma region - is the role talking to all npc -
    bool IsTalkedAllNpc = true;
    vector<CNPC*>* npc = npcManager.GetNpc(mapManager.GetNowMap());
    for (vector<CNPC*>::iterator npciter = npc->begin(); npciter != npc->end(); npciter++)
    {
        if (!(*npciter)->IsTalked())
        {
            IsTalkedAllNpc = false;
        }
    }
    #pragma endregion
```

```
    if (isWinFacaiSeed)
    {
        CEndManager::Instance()->Start(END_NAME_WIN_FACAISEED);
    }
    else if (isWinXingting)
    {
        CEndManager::Instance()->Start(END_NAME_WINXINGTING);
    }
    else if (IsTalkedAllNpc)
    {
        CEndManager::Instance()->Start(END_NAME_SALTEDFISH);
    }
    else if (role.GetScore() > 999)
    {
        CEndManager::Instance()->Start(END_NAME_SALTEDFISH);
    }
    else
    {
        CEndManager::Instance()->Start(END_NAME_SALTEDFISH);
    }
}

void CGameStateRun::OnShow()
{
    panel.OnShow();
    #pragma region - paint layer -
    CLayerManager::Instance()->ShowLayer();
    #pragma endregion
}

#pragma endregion
#pragma region - CGame Over -
CGameStateOver::CGameStateOver(CGame* g) : CGameState(g)
{
}

void CGameStateOver::OnMove()
{
    if (CDialogManager::Instance()->GetDialogState())
    {
        CDialogManager::Instance()->OnCycle();
        return;
    }
    #pragma region - CountDown for drawGameOverImage -
    if (!CEndManager::Instance()->GetEndingState())
    {
        timer_exit.CountDown();
        if (timer_exit.IsTimeOut())
        {
            canDrawGameOverImage = true;
        }
    }
    #pragma endregion
}

void CGameStateOver::OnBeginState()
{
    counter = 30 * 5; // 5 seconds
    alpha = 0;
    timer_exit.ResetTime(1.5);
    CAudio::Instance()->Play("SLoWMoTIoN_Gameover");
    CAudio::Instance()->Stop("SLoWMoTIoN_Game");
    canSwitchState = false;
    canDrawGameOverImage = false;
    #pragma region - init -
    CLayerManager::Instance()->Initialize();
    CDialogManager::Instance()->Initialize();
    #pragma endregion
}
```

```
void CGameStateOver::OnInit()
{
   overBitmap.LoadBitmap(".\\RES\\Map\\Gameover.bmp");
   gameOverImage.SetBmp("RES\\Map\\Gameover.bmp");
   gameOverImage.SetFadeInOut(60, -40);
   ShowInitProgress(66);   // 接個前一個狀態的進度，此處進度視為 66%
   ShowInitProgress(100);
}

void CGameStateOver::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
   if (nChar == 'Y')
   {
      if (CDialogManager::Instance()->GetDialogState())
      {
         CDialogManager::Instance()->Stop();
      }
   }
}

void CGameStateOver::OnLButtonDown(UINT nFlags, CPoint point)
{
   if (CDialogManager::Instance()->GetDialogState())
   {
      CDialogManager::Instance()->Next();
   }
   else
   {
      if (canSwitchState)
      {
         GotoGameState(GAME_STATE_INIT);
      }
   }
}

void CGameStateOver::OnShow()
{
   if (CEndManager::Instance()->GetEndingState())
   {
      CEndManager::Instance()->OnCycle();
   }
   #pragma region - Draw Dialog -
   CLayerManager::Instance()->ShowLayer(); //顯示 layerManager
   #pragma endregion
   #pragma region - DrawGameOverImage when Ending end -
   if (canDrawGameOverImage)
   {
      gameOverImage.DrawImage();
      gameOverImage.FadeIn();

      if (gameOverImage.GetAlpha() >= 255)
      {
         canSwitchState = true;
      }
   }
   #pragma endregion
}
#pragma endregion
#pragma region - MapEditer -
CGameStateMapEditer::CGameStateMapEditer(CGame* g) : CGameState(g)
{
}
CGameStateMapEditer::~CGameStateMapEditer()
{
}

void CGameStateMapEditer::OnInit()
{
```

```
}

void CGameStateMapEditer::OnBeginState()
{
   mapEditer.Initialize();
}

void CGameStateMapEditer::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
   if (nChar == 'S')
   {
      mapEditer.OnSave();
   }
   if (nChar == 'A') // map move left
   {
      mapEditer.SetMapMoveDir("left");
   }
   if (nChar == 'D') //map move right
   {
      mapEditer.SetMapMoveDir("right");
   }
   if (nChar == 9) //Key Tab
   {
      mapEditer.isPrintNowMap = !mapEditer.isPrintNowMap;
   }
   if (nChar == 46) //Key Delete
   {
      if (mapEditer.IsInSelectMapMode())
      {
         mapEditer.SetSelectMapZero();
      }
      else
      {
         mapEditer.DeleteBlock();
      }
   }
   #pragma region - Set upMap / downMap / leftMap / rightMap -
   if (!mapEditer.IsInSelectMapMode()) //不在設定地圖模式中
   {
      string dir[4] = { "up", "down", "left", "right" };

      //Key 1, 2, 3, 4
      if (nChar >= '1' && nChar <= '4')
      {
         mapEditer.SetSelectMapMode(dir[nChar - 48 - 1]);
      }
   }
   #pragma endregion
}

void CGameStateMapEditer::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
   if (nChar == 27) //Key ESC, return to stateInit
   {
      if (!mapEditer.IsInSelectMapMode())
      {
         mapEditer.StoreMapInformaion();
         mapEditer.CreateReloadMapInformation();
         GotoGameState(GAME_STATE_INIT);
      }
      else
      {
         mapEditer.SetSelectMapMode("none");
      }
   }
   if (nChar == 'A')
   {
      mapEditer.SetMapMoveDir("cleft");
   }
```

```
    if (nChar == 'D')
    {
        mapEditer.SetMapMoveDir("cright");
    }
}

void CGameStateMapEditer::OnLButtonDown(UINT nFlags, CPoint point)
{
    if (mapEditer.IsInSelectMapMode()) //在選擇地圖的模式中
    {
        mapEditer.ClickArrow(point);
    }
    else
    {
        mapEditer.SetMouseState(true);
        mapEditer.SelectBlock(point);
    }
}

void CGameStateMapEditer::OnLButtonUp(UINT nFlags, CPoint point)
{
    if (!mapEditer.IsInSelectMapMode())
    {
        mapEditer.SetMouseState(false);
    }
}

void CGameStateMapEditer::OnMouseMove(UINT nFlags, CPoint point)
{
    mousePoint = point;
}

void CGameStateMapEditer::OnRButtonDown(UINT nFlags, CPoint point)
{
}

void CGameStateMapEditer::OnRButtonUp(UINT nFlags, CPoint point)
{
}

void CGameStateMapEditer::OnMove()
{
    if (mapEditer.IsInSelectMapMode()) //選擇地圖中 - 下面功能不動作
    {
        return;
    }
    #pragma region - 判斷檔案是否存在 -
    bool isDataExist = false; //先假設檔案不存在
    fstream dataFileName;
    dataFileName.open("RES\\Map\\FileName.txt");
    #pragma region -- 檔案存在 --
    if (dataFileName.is_open()) //有檔案就 load 他
    {
        vector<string> fileContext;
        fileContext.clear();
        string fileName;
        while (dataFileName >> fileName)
        {
            fileContext.push_back(fileName);
        }
        mapEditer.AddImage(fileContext);
        isDataExist = true; //檔案存在
    }
    #pragma endregion
    dataFileName.close(); //關檔
    #pragma region -- 檔案真的存在 - 刪除檔案^^ --
    if (isDataExist)
    {
        DeleteFile("RES\\Map\\FileName.txt");
```

```
    }
    #pragma endregion
    #pragma endregion
    #pragma region - 拖曳 block -
    if (mapEditer.GetMouseState()) //滑鼠按住中
    {
      mapEditer.DragBlock(mousePoint);
    }
    #pragma endregion
    mapEditer.OnMove();
  }

  void CGameStateMapEditer::OnShow()
  {
    mapEditer.OnShow();
    PaintText("MapEditer", 0, 445, "微軟正黑體", 20, RGB(0, 0, 0), RGB(255, 255, 255)); //Text ,位置, 文字字形(sp),
文字大小, 文字顏色, 背景顏色
    if (mapEditer.isPrintNowMap)
    {
      char* address = ConvertCharPointToString(mapEditer.GetNowMap());
      PaintText(address, 320, 0, "微軟正黑體", 20, RGB(0, 0, 0), RGB(255, 255, 255)); //Text ,位置, 文字字形(sp),
文字大小, 文字顏色, 背景顏色
      delete address;
    }
    if (mapEditer.IsInSelectMapMode())
    {
      char* address = ConvertCharPointToString(mapEditer.GetSelectMapMode());
      PaintText(address, 90, 0, "微軟正黑體", 20, RGB(0, 0, 0), RGB(255, 255, 255)); //Text ,位置, 文字字形(sp), 文
字大小, 文字顏色, 背景顏色
      delete address;
    }
  }
  #pragma endregion
}//End
```

3. CEraser.h

```
#pragma once
#include "CBoss.h"
#include "CLibrary.h"
namespace game_framework {
  #pragma region - CEraser -
  class CBlock;
  class CEraser
  {
    public:
      CEraser();
      virtual ~CEraser();
      int    GetX1(); // 左上角 x 座標
      int    GetY1(); // 左上角 y 座標
      int    GetX3(); // 中心點 X 座標
      int    GetX2(); // 右下角 x 座標
      int    GetY2(); // 右下角 y 座標
      int    GetY3(); // 中心點 Y 座標
      int    GetScore();
      int    Height();
      int    Width();
      bool GetMovingLeft(); // 方方是否在往左走
      bool GetMovingRight(); // 方方是否再往右走
      void Initialize(); // 初始化
      // 由路徑載入圖形(資料夾, name, 張數)
      void LoadBitmap(string, string, int, COLORREF);
      void LoadBitmap(BitmapPath);
      void OnMove(); // 移動擦子
      void OnShow(); // 將擦子圖形貼到畫面
      CAnimate* GetAnimate();
      void SetMovingDown(bool flag); // 設定是否正在往下移動
      void SetMovingLeft(bool flag); // 設定是否正在往左移動
      void SetMovingRight(bool flag); // 設定是否正在往右移動
```

```
        void SetMovingUp(bool flag); // 設定是否正在往上移動
        void SetXY(int nx, int ny); // 設定擦子左上角座標
        void SetCanMoving(bool);
        bool GetCanMoving();
        CLayer layer;
        bool GetValid();
        void SetValid(bool);
        #pragma region - collision -
        virtual void ResetCollisionRect();
        bool IsCollisionBlock(CBlock*); //一般左右 跟 block 碰撞
        bool IsRoleOnBlock(CBlock*); //站在方塊上
        #pragma endregion
    protected:
        CAnimate animation; // 擦子的動畫
        int height, width; // 擦子的高、寬
        int x, y; // 擦子左上角座標
        int score;
        bool isMovingDown; // 是否正在往下移動
        bool isMovingLeft; // 是否正在往左移動
        bool isMovingRight; // 是否正在往右移動
        bool isMovingUp; // 是否正在往上移動
        bool canMoving; // 是否可以移動
        CRect collisionRect;
        CRect collisionDownRect;
    private:
        int move_distance = MOVE_DISTANCE;
};
#pragma endregion
#pragma region - CNPC -
class CNPC : public CEraser
{
    public:
        CNPC();
        CNPC(CPoint, BitmapPath, string, double);
        virtual ~CNPC();
        void SetCurrentXY(int, int);
        void SetXY();
        void LookRole(CPoint);
        virtual void OnCycle(CPoint);
        virtual void OnMove();
        virtual void Initialize();
        virtual void RoleCollision() {}; //碰撞到後要做的事情
        virtual void SetValid(bool);
        void SetTalked(bool f) {
            isTalked = f;
        };
        bool IsTalked() {
            return isTalked;
        };
    protected:
        #pragma region - init information -
        CPoint initPoint;
        BitmapPath initLoadPath;
        double initResetTime;
        #pragma endregion
        CAnimate leftAnimate;
        CAnimate rightAnimate;
        int currentX, currentY;
        string id;
        string faceTo;
        bool isTalked;
};
#pragma endregion
#pragma region - CRole -
class CBlackHole;
class CDoor;
class CRole : public CEraser
{
        friend class CScallion;
```

```
public:
    CRole();
    ~CRole();
    int    GetX3(); // 中新點 X 座標
    int    GetX2(); // 擦子右下角 x 座標
    int    GetY2(); // 擦子右下角 y 座標
    int    GetY3(); // 中心點 Y 座標
    int    Height();
    int    Width();
    void SetXY(int _x, int _y); // 設定擦子左上角座標
    bool GetValid();
    void SetValid(bool);
    void Load();
    void LoadasMascot();
    void LoadAction(string, BitmapPath); //載入動應動作
    void OnMove();
    bool GetMovingJump();
    void SetMovingJump(bool);
    bool GetCanJumping();
    void SetCanJumping(bool);
    void SetMouseXY(int, int);
    void AddScore(int);
    void SubHp();
    int GetHp() {
        return hp;
    };
    void SubEq();
    int GetEq() {
        return eq;
    };
    void Initialize();
    bool GetIsFire();
    void SetIsFire(bool);
    void Fire(int, int); //傳入滑鼠座標
    void SetDrop();
    bool GetDrop();
    int GetVelocity();
    void SetVelocity(int v) {
        velocity = v;
    };
    bool IsMoving();
    bool IsDead() {
        return isDead;
    };
    bool IsZZ() {
        return isZZ;
    };
    void SetRoleNoSubHp() {
        isRoleNoSubHp = !isRoleNoSubHp;
    };
    CAction* GetAction() {
        return &action;
    };
    #pragma region - Collision -
    bool IsCollisionBoss(CBoss*);
    #pragma region -- Collision - Block --
    bool IsCollisionBlockOnJumping(CBlock*); //跳起來撞到方塊
    #pragma endregion
    bool IsCollisionDoor(CDoor*);
    bool IsCollisionLevel4(CScallion*);
    bool IsCollisionNPC(CNPC*);
    bool IsCollisionBlackHole(CBlackHole*);
    bool IsCollisionBlackHoleCenter(CBlackHole*);
    bool IsCollisionRay(CRay*);
    #pragma endregion
    vector<CScallion*>* GetScallion();
    void SetCatched(bool flag) {
        isCatched = flag;
    };
```

```
        bool IsCatched() {
            return isCatched;
        };
    protected:
        vector<CScallion*>    scallion;
        bool isJumping; // 是否正在跳躍
        bool canJumping; // 是否可以跳躍
        bool isFire; // 是否正在射擊
        int init_velocity; // 往上的初速度
        int velocity; // 速度
        int gravity; // 重力
        char last_right_left;
        int mouse_x, mouse_y;
        int move_distance = MOVE_DISTANCE; //每次移動的距離
        CTimer shoot_cd;
        CMovingBitmap decisionPoint;
        CRect collisionTopRect;
        CRect collisionDoorRect;
        void ResetCollisionRect();
    private:
        CAction    action;
        #pragma region - init information -
        int inithp = 20;
        int initEq = 20;
        bool isLoaded;
        string now_action;
        #pragma endregion
        int hp, eq;
        bool isCatched;
        bool isDead;
        bool isZZ;
        bool isRoleNoSubHp;
};
#pragma endregion
#pragma region - CMonster -
class CMonster : public CEraser
{
        friend class CScallion;
    public:
        CMonster();
        CMonster(int, int);
        void SetBlock(vector<CBlock>* bkvector) {
            blockVector = bkvector;
        }
        void SetXY(int, int);
        void SetScore(int);
        void SetMoving();
        virtual void OnMove();
        virtual ~CMonster();
    protected:
        int initX;
        CTimer moveTimer;
        CTimer stopTimer;
        CTimer recreateTimer;
        int move_distance = 2;
        CAnimate leftAnimate;
        CAnimate rightAnimate;
        string faceto;
        bool onFloor;
        bool moveleft, moveright;
        vector<CBlock>* blockVector;
        void ResetCollisionRect();
};
#pragma region - CMonster1 - LUKA -
class CMonsterType1 : public CMonster
{
    public:
        CMonsterType1();
        CMonsterType1(int, int, BitmapPath, int);
```

```
        ~CMonsterType1();
    };
    #pragma endregion
    #pragma region - CMonster2 - mushroom -
    class CMonsterType2 : public CMonster
    {
      public:
        CMonsterType2();
        CMonsterType2(int, int, BitmapPath, int);
        void OnMove();
        ~CMonsterType2();
    };
    #pragma endregion
    #pragma endregion
    #pragma region - CNPC - No.1 -
    //CNPC1 > 普通的 NPC (對話 NPC)
    class CNPC1 : public CNPC
    {
      public:
        CNPC1();
        CNPC1(CPoint, BitmapPath, string, string, double resetTime = 0.05); //座標 路徑 id 對話文本 一個動畫的影
格的時間
        ~CNPC1();
        void RoleCollision();
      private:
        string txt;
    };
    #pragma endregion
    #pragma region - CNPC - No.3 -
    //CNPC3 > 音樂播放 NPC
    class CNPC3 : public CNPC
    {
      public:
        CNPC3();
        CNPC3(CPoint, BitmapPath, string, string, string); //座標 路徑 ID 音樂 對話文本
        ~CNPC3();
        void RoleCollision();
        void Initialize();
        void SetValid(bool);
        void OnMove();
      private:
        string openTxt;
        string openMusic;
        CMovingBitmap headphoneIcon;
        bool isLoadHeadPhoneIcon;
    };
    #pragma endregion
}
```

4. CEraser.cpp

```
#pragma once
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "CEraser.h"
#include "CBall.h"
#include <sstream>
#include "CBoss.h"
#include "CManager.h"
#include "CLibrary.h"
using namespace myLibrary;
namespace game_framework {
    #pragma region - CEraser -
    CEraser::CEraser()
    {
```

```
    Initialize();
}

CEraser::~CEraser()
{
}

void CEraser::Initialize()
{
    const int X_POS = 0;
    const int Y_POS = -160;
    x = X_POS;
    y = Y_POS;
    isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;
    canMoving = true;
    layer.SetLayer(5);
    score = 0;
}

int CEraser::GetX1()
{
    return x;
}

int CEraser::GetY1()
{
    return y;
}

int CEraser::GetX2()
{
    return x + animation.Width();
}

int CEraser::GetY2()
{
    return y + animation.Height();
}

int CEraser::GetY3()
{
    return y + animation.Height() / 2;
}

int CEraser::GetScore()
{
    return score;
}

int CEraser::GetX3()
{
    return x + animation.Width() / 2;
}

int CEraser::Height()
{
    return height;
}

int CEraser::Width()
{
    return width;
}

bool CEraser::GetMovingLeft()
{
    return isMovingLeft;
}
```

```
bool CEraser::GetMovingRight()
{
   return isMovingRight;
}

void CEraser::LoadBitmap(string ziliaojia, string name, int number, COLORREF color)
{
   animation.LoadBitmap(ziliaojia, name, number, color);
   animation.SetTopLeft(x, y);
   height = animation.Height();
   width = animation.Width();
}

void CEraser::LoadBitmap(BitmapPath loadpath)
{
   animation.LoadBitmap(loadpath.ziliaojia, loadpath.name, loadpath.number, loadpath.color);
}

void CEraser::OnMove()
{
   const int STEP_SIZE = move_distance;
   int dir = -1;
   if (isMovingUp)
   {
      dir = 0;
   }
   if (isMovingRight)
   {
      dir = 1;

      if (canMoving)
         x += STEP_SIZE;
   }
   if (isMovingDown)
   {
      dir = 2;

      if (canMoving)
         y += STEP_SIZE;
   }
   if (isMovingLeft)
   {
      dir = 3;

      if (canMoving)
         x -= STEP_SIZE;
   }
   animation.SetTopLeft(x, y);
   animation.OnMove(dir);
}

void CEraser::OnShow()
{
}

void CEraser::SetMovingDown(bool flag)
{
   isMovingDown = flag;
   canMoving = flag | isMovingLeft | isMovingRight | isMovingUp;
}

void CEraser::SetMovingLeft(bool flag)
{
   isMovingLeft = flag;
   canMoving = flag | isMovingDown | isMovingRight | isMovingUp;
}

void CEraser::SetMovingRight(bool flag)
{
```

```
    isMovingRight = flag;
    canMoving = flag | isMovingDown | isMovingLeft | isMovingUp;
}

void CEraser::SetMovingUp(bool flag)
{
    isMovingUp = flag;
    canMoving = flag | isMovingLeft | isMovingRight | isMovingDown;
}

void CEraser::SetCanMoving(bool _canMoving)
{
    canMoving = _canMoving;
}

bool CEraser::GetCanMoving()
{
    return canMoving;
}

bool CEraser::GetValid()
{
    return animation.GetValid();
}

void CEraser::SetValid(bool flag)
{
    animation.SetValid(flag);
}

void CEraser::SetXY(int nx, int ny)
{
    x = nx;
    y = ny;
}

CAnimate* CEraser::GetAnimate()
{
    animation.SetTopLeft(x, y);
    return &animation;
}

void CEraser::ResetCollisionRect()
{
    if (!animation.IsNull())
    {
        collisionRect = animation.GetRect();
        collisionDownRect = animation.GetRect();
    }
}
#pragma endregion

#pragma region - CRole -
CRole::CRole() : CEraser()
{
    isLoaded = false;
}
CRole::~CRole()
{
    for (vector<CScallion*>::iterator it = scallion.begin(); it != scallion.end(); it++)
    {
        delete (*it);
        (*it) = NULL;
    }

    vector<CScallion*>().swap(scallion);
}

int CRole::GetX2()
```

```
{
    return x + action.Width();
}

int CRole::GetY2()
{
    return y + action.Height();
}

int CRole::GetY3()
{
    return y + action.Height() / 2;
}

int CRole::GetX3()
{
    return x + action.Width() / 2;
}

int CRole::Height()
{
    return action.Height();
}

int CRole::Width()
{
    return action.Width();
}

void CRole::SetXY(int _x, int _y)
{
    x = _x;
    y = _y;
    action.SetTopLeft(_x, _y);
}

bool CRole::GetValid()
{
    return action.GetValid();
}

void CRole::SetValid(bool _flag)
{
    action.SetValid(_flag);
}

void CRole::Load()
{
    LoadAction("idle", BitmapPath("RES\\Role\\miku\\idle", "idle", 19, RGB(150, 200, 250)));
    LoadAction("run", BitmapPath("RES\\Role\\miku\\run", "run", 7, RGB(150, 200, 250)));
    LoadAction("jump", BitmapPath("RES\\Role\\miku\\jump", "jump", 5, RGB(150, 200, 250)));
    decisionPoint.LoadBitmap("RES\\Role\\miku\\cursor.bmp", RGB(214, 214, 214));
}

void CRole::LoadasMascot()
{
    LoadAction("idle", BitmapPath("RES\\Role\\miku\\idle", "idle", 19, RGB(150, 200, 250)));
    LoadAction("run", BitmapPath("RES\\Role\\miku\\run", "run", 7, RGB(150, 200, 250)));
}

void CRole::LoadAction(string _action, BitmapPath _loadpath)
{
    action.LoadAction(_action, _loadpath);
}

void CRole::OnMove()
{
    const int STEP_SIZE = move_distance;
    int dir = 0;
```

```
if (isMovingUp)
{
   dir = 1;

   if (canJumping)
   {
      CAudio::Instance()->Play("jump");
      isJumping = true;
   }
}
if (isMovingRight)
{
   dir = 2;

   if (canMoving)
      x += STEP_SIZE;

   action.SetFaceTo("_R");
}
if (isMovingDown)
{
   dir = 3;

   if (canMoving)
      y += STEP_SIZE;
}
if (isMovingLeft)
{
   dir = 4;

   if (canMoving)
      x -= STEP_SIZE;

   action.SetFaceTo("_L");
}
if (isJumping)
{
   if (canJumping)
   {
      velocity = init_velocity;
      canJumping = false;
   }
   else
   {
      //jump
      if (velocity > 0)
      {
         velocity -= gravity;
         y -= velocity;
      }
      else
      {
         velocity -= gravity;
         y -= velocity;
      }
   }
}
if (isFire)
{
   Fire(mouse_x, mouse_y);
}
shoot_cd.CountDown(); //設置射擊的 CD 時間
if (isJumping)
{
   decisionPoint.SetTopLeft(GetX3() - 5, GetY3() + 16);
}
else
{
   decisionPoint.SetTopLeft(GetX3() - 5, GetY3());
```

```
    }
    action.SetTopLeft(x, y);
    if (isJumping)
    {
        action.OnMove("jump");
        action.SetTopLeft(x, y - (action.GetActionHeight("jump") - action.GetActionHeight("idle")));
    }
    else if (isMovingLeft || isMovingRight)
    {
        action.OnMove("run");
    }
    else
    {
        action.OnMove("idle");
    }
    for (unsigned int i = 0; i < scallion.size(); i++)
    {
        scallion[i]->OnMove();
    }
}

bool CRole::GetMovingJump()
{
    return isJumping;
}

void CRole::SetMovingJump(bool flag)
{
    isJumping = flag;
}

bool CRole::GetCanJumping()
{
    return canJumping;
}

void CRole::SetCanJumping(bool flag)
{
    canJumping = flag;
}

void CRole::SetMouseXY(int _x, int _y)
{
    mouse_x = _x;
    mouse_y = _y;
}

bool CRole::GetIsFire()
{
    return isFire;
}

void CRole::SetIsFire(bool flag)
{
    isFire = flag;
}

void CRole::Fire(int mouseX, int mouseY)
{
    if (shoot_cd.IsTimeOut())
    {
        CAudio::Instance()->Play("throw");
        CScallion* newCScallion = new CScallion(BitmapPath("RES\\Object\\Scallions", "scallion", 2), CPoint(GetX3(),
GetY1()), CPoint(mouseX, mouseY)); //先創建一個蔥的物件
        scallion.push_back(newCScallion); //將蔥放進 vector
        shoot_cd.ResetTime(0.33);
    }
}
```

```
void CRole::SetDrop()
{
   SetMovingJump(true);
   SetCanJumping(false);
   SetVelocity(0);
}

bool CRole::GetDrop()
{
   return velocity < 0;
}

int CRole::GetVelocity()
{
   return velocity;
}

bool CRole::IsMoving()
{
   return (isMovingRight || isMovingLeft);
}

#pragma region -- Collision --
bool CRole::IsCollisionBoss(CBoss* boss)
{
   ResetCollisionRect();
   return IsRectCollision(collisionRect, boss->GetAnimate()->GetRect());
}

bool CEraser::IsCollisionBlock(CBlock* block) //get iterator in mygame.cpp
{
   ResetCollisionRect();
   return IsRectCollision(collisionRect, block->bmp.GetRect());
}

bool CEraser::IsRoleOnBlock(CBlock* block)
{
   ResetCollisionRect();
   return IsRectCollision(collisionDownRect, block->bmp.GetRect());
}

bool CRole::IsCollisionBlockOnJumping(CBlock* block)
{
   ResetCollisionRect();
   return IsRectCollision(collisionTopRect, block->bmp.GetRect());
}

bool CRole::IsCollisionDoor(CDoor* door)
{
   ResetCollisionRect();
   return IsRectCollision(collisionDoorRect, door->bmp.GetRect());
}

bool CRole::IsCollisionLevel4(CScallion* level4)
{
   return IsRectCollision(decisionPoint.GetRect(), level4->GetAnimate()->GetRect());
}

bool CRole::IsCollisionNPC(CNPC* npc)
{
   return IsRectCollision(action.GetRect(), npc->GetAnimate()->GetRect());
}

bool CRole::IsCollisionBlackHole(CBlackHole* blackHole)
{
   return IsRectCollision(decisionPoint.GetRect(), blackHole->GetCenterRect());
}

bool CRole::IsCollisionBlackHoleCenter(CBlackHole* blackHole)
```

```
{
    return IsRectCollision(decisionPoint.GetRect(), blackHole->GetCenterRect());
}
bool CRole::IsCollisionRay(CRay* ray)
{
    return IsRectCollision(decisionPoint.GetRect(), ray->GetAnimate()->GetRect());
}
#pragma endregion

void CRole::AddScore(int _score)
{
    score += _score;
}

void CRole::SubHp()
{
    if (!isRoleNoSubHp)
    {
        hp--;
    }
    if (hp <= 0)
    {
        isDead = true;
    }
}

void CRole::SubEq()
{
    eq--;
    if (eq <= 0)
    {
        isZZ = true;
    }
}

void CRole::Initialize()
{
    action.Initialize();
    #pragma region    Normal Setting
    CEraser::Initialize();
    isJumping = true;
    canJumping = false;
    isFire = false;
    hp = inithp;
    eq = initEq;
    score = 0;
    const int INIT_VELOCITY = 23; //設定初速度
    const int GRAVITY = 1; //設定重力
    init_velocity = velocity = INIT_VELOCITY;
    gravity = GRAVITY;
    mouse_x, mouse_y = 0;
    x = 0;
    y = -Height();
    shoot_cd = CTimer(0); //初始化射擊冷卻時間
    isCatched = false;
    isDead = false;
    isZZ = false;
    isRoleNoSubHp = false;
    SetValid(true);
    #pragma endregion
    #pragma region Load
    if (!isLoaded)
    {
        isLoaded = true;
    }
    #pragma endregion
    collisionRect = CRect(CPoint(x - move_distance, y - move_distance), CPoint(x + Width() + move_distance, y +
Height() + move_distance));
    CLayerManager::Instance()->AddObject(&decisionPoint, layer.GetLayer() + 1);
```

```
    CLayerManager::Instance()->AddObject(&action, layer.GetLayer());
}

vector<CScallion*>* CRole::GetScallion()
{
    return &scallion;
}

void CRole::ResetCollisionRect()
{
    #pragma region - collision Rect - Right and Left -
    collisionRect = action.GetRect();
    int dx = move_distance;
    collisionRect.bottom -= 10;
    if (isMovingRight)
    {
        //Test 地方調整往右/往左的判斷寬度 (原本人物寬度 + dx，改成從中心點算起的寬度 + dx)
        collisionRect.left = GetX3(); //Test
        collisionRect.right += dx;
    }
    else if (isMovingLeft)
    {
        collisionRect.left -= dx;
        collisionRect.right = GetX3(); //Test
    }
    #pragma endregion
    #pragma region - collision Rect - Down -
    collisionDownRect = action.GetRect();
    //調整 下方判斷方塊的高度，以及縮減左/右的寬度
    collisionDownRect.top = GetY2() - 5; //調整判斷的高度
    collisionDownRect.right -= dx * 3;
    collisionDownRect.left += dx * 3;
    int dy = move_distance ;
    if (GetDrop())
    {
        collisionDownRect.bottom += dy;
    }
    #pragma endregion
    #pragma region - collision Rect - Top -
    collisionTopRect = action.GetRect();
    //縮減左/右的寬度
    collisionTopRect.right -= dx * 3;
    collisionTopRect.left += dx * 3;
    collisionTopRect.bottom = GetY3(); //調整判斷方塊高度
    if (!GetMovingJump()) //沒跳躍
    {
        collisionTopRect.top -= dy * 3;
    }
    else if (GetMovingJump() && !GetDrop()) //跳躍中
    {
        collisionTopRect.top += dy * 3;
    }
    #pragma endregion
    #pragma region - collsion Rect - Door -
    collisionDoorRect = decisionPoint.GetRect();
    collisionDoorRect.top -= 20;
    collisionDoorRect.bottom += 20;
    collisionDoorRect.left -= 20;
    collisionDoorRect.right += 20;
    #pragma endregion
}
#pragma endregion

#pragma region - CMonster -
CMonster::CMonster() : CEraser()
{
    initX = 0;
    x = initX;
    y = 300;
```

```
    layer.SetLayer(4);
    score = 0;
    moveTimer = CTimer(0.5);
    stopTimer = CTimer(1.5);
    SetValid(false);
    recreateTimer = CTimer((double)GetRandom(1, 4) / 2.0);
}

CMonster::CMonster(int _x, int _y) : CEraser()
{
    initX = _x;
    x = initX;
    y = _y;
    layer.SetLayer(4);
    CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());
    score = 0;
    moveTimer = CTimer(0.5);
    stopTimer = CTimer(1.5);
    SetValid(false);
    recreateTimer = CTimer((double)GetRandom(1, 2) / 2.0);
    onFloor = false;
    blockVector = NULL;
}

void CMonster::SetXY(int _x, int _y)
{
    initX = _x;
    if (_y + height >= SIZE_Y - 2)
    {
        y = SIZE_Y - height - 2;
    }
    else
    {
        y = _y;
    }
    int dx = CCamera::Instance()->GetX();
    x = initX - dx;
    animation.SetTopLeft(x, y);
}

void CMonster::SetScore(int _score)
{
    score = _score;
}

void CMonster::SetMoving()
{
    if (GetValid() == false)
    {
        recreateTimer.CountDown();
        if (!recreateTimer.IsTimeOut())
        {
            return;
        }
    }
    if ( !moveTimer.IsTimeOut())
    {
        moveTimer.CountDown();
    }
    else
    {
        SetCanMoving(false);
        stopTimer.CountDown();
        if (stopTimer.IsTimeOut())
        {
            moveleft = moveright = false;
            int randomNumber = GetRandom(1, 10);
            if (randomNumber % 2 == 0)
            {
```

```
            moveleft = true;
            moveright = false;
        }
        else
        {
            moveleft = false;
            moveright = true;
        }
        double resetStopTime = GetRandom(1, 10) / 10;
        SetCanMoving(true);
        moveTimer.ResetTime(0.1);
        stopTimer.ResetTime(resetStopTime);
    }
  }
}

void CMonster::OnMove()
{
  if (!recreateTimer.IsTimeOut())
    return;
  const int STEP_SIZE = move_distance;
  bool nowOnFloor = false;
  if (blockVector != NULL)
  {
    for (vector<CBlock>::iterator bkiter = blockVector->begin(); bkiter != blockVector->end(); bkiter++)
    {
      if (IsPointInRect(GetY2(), bkiter->bmp.GetRect()))
      {
        y += (GetY2() - bkiter->y) + 2;
      }
      if (IsCollisionBlock(&(*bkiter)))
      {
        moveright = false;
        moveleft = false;
        SetCanMoving(false);
      }
      if ((IsRoleOnBlock(&(*bkiter))))
      {
        nowOnFloor = true;
      }
    }
  }
  if (GetY2() >= SIZE_Y - 2)
  {
    nowOnFloor = true;
  }
  if (nowOnFloor)
  {
    SetValid(true);
    SetMovingDown(false);
    onFloor = true;
  }
  else
  {
    moveleft = false;
    moveright = false;
    SetMovingDown(true);
  }
  SetMovingLeft(moveleft);
  SetMovingRight(moveright);
  if (isMovingRight)
  {
    if (canMoving)
      initX += STEP_SIZE;
  }
  if (isMovingDown)
  {
    if (canMoving)
    {
```

```
            int dy = STEP_SIZE * 3;
            y += dy;
        }
    }
    if (isMovingLeft)
    {
        if (canMoving)
            initX -= STEP_SIZE;
    }
    SetXY(initX, y);
    animation.OnMove();
}

CMonster::~CMonster()
{
}

void CMonster::ResetCollisionRect()
{
    //reset passerby's collision rect
    #pragma region - collision Rect - Right and Left -
    collisionRect = animation.GetRect();
    int dx = 4;
    collisionRect.bottom -= 2;
    if (isMovingRight)
    {
        //Test 地方調整往右/往左的判斷寬度 (原本人物寬度 + dx，改成從中心點算起的寬度 + dx)
        collisionRect.left = GetX3(); //Test
        collisionRect.right += dx;
    }
    else if (isMovingLeft)
    {
        collisionRect.left -= dx;
        collisionRect.right = GetX3(); //Test
    }
    #pragma endregion
    #pragma region - collision Rect - Down -
    collisionDownRect = animation.GetRect();
    //調整 下方判斷方塊的高度，以及縮減左/右的寬度
    collisionDownRect.top = GetY2() - 5; //調整判斷的高度
    int dy = 4;
    collisionDownRect.bottom += dy;
    #pragma endregion
}

#pragma region - CMonsterType1 - Luka -
CMonsterType1::CMonsterType1() : CMonster()
{
}
CMonsterType1::CMonsterType1(int _x, int _y, BitmapPath _loadPath, int _score) : CMonster(_x, _y)
{
    SetScore(_score);
    LoadBitmap(_loadPath.ziliaojia, _loadPath.name, _loadPath.number, _loadPath.color);
}
CMonsterType1::~CMonsterType1()
{
}
#pragma endregion

#pragma region - CMonsterType2 - mushroom -
CMonsterType2::CMonsterType2() : CMonster()
{
}

CMonsterType2::CMonsterType2(int _x, int _y, BitmapPath _loadPath, int _score) : CMonster(_x, _y)
{
    SetScore(_score);
    BitmapPath leftPath = _loadPath;
    BitmapPath rightPath = _loadPath;
```

```
    leftPath.ziliaojia += "\\L";
    rightPath.ziliaojia += "\\R";
    LoadBitmap(leftPath.ziliaojia, leftPath.name, leftPath.number, leftPath.color);
    leftAnimate = animation;
    rightAnimate.LoadBitmap(rightPath.ziliaojia, rightPath.name, rightPath.number, rightPath.color);
    leftAnimate.ResetDelayTime(0.1);
    rightAnimate.ResetDelayTime(0.1);
    leftAnimate.CopyAnimateInformation(&animation);
    rightAnimate.CopyAnimateInformation(&animation);
    animation = leftAnimate;
    faceto = "left";
}

CMonsterType2::~CMonsterType2()
{
}

void CMonsterType2::OnMove()
{
    if (GetMovingLeft() && faceto != "left")
    {
        leftAnimate.CopyAnimateInformation(&animation);
        animation = leftAnimate;
        faceto = "left";
    }
    else if (GetMovingRight() && faceto != "right")
    {
        rightAnimate.CopyAnimateInformation(&animation);
        animation = rightAnimate;
        faceto = "right";
    }
    CMonster::OnMove();
}
#pragma endregion

#pragma endregion

#pragma region - CNPC -
CNPC::CNPC() : CEraser()
{
}

CNPC::CNPC(CPoint _point, BitmapPath _loadPath, string _id, double resetTime)
{
    initPoint = _point;
    initLoadPath = _loadPath;
    id = _id;
    initResetTime = resetTime;
}

void CNPC::Initialize()
{
    #pragma region - load animation (only once) -
    if (animation.IsNull())
    {
        BitmapPath leftPath = initLoadPath;
        leftPath.ziliaojia += "\\L\\";
        BitmapPath rightPath = initLoadPath;
        rightPath.ziliaojia += "\\R\\";
        LoadBitmap(leftPath);
        leftAnimate = animation;
        rightAnimate.LoadBitmap(rightPath);
        leftAnimate.ResetDelayTime(initResetTime);
        rightAnimate.ResetDelayTime(initResetTime);
        leftAnimate.CopyAnimateInformation(&animation);
        rightAnimate.CopyAnimateInformation(&animation);
        animation = leftAnimate;
        faceTo = "left";
    }
```

```
    #pragma endregion
    SetCurrentXY(initPoint.x, initPoint.y);
    layer.SetLayer(NPC_LAYER);
    animation.SetValid(false);
    CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());
    isTalked = false;
}

void CNPC::SetValid(bool flag)
{
    animation.SetValid(flag);
}

void CNPC::SetCurrentXY(int _x, int _y)
{
    currentX = _x;
    currentY = _y;
    SetXY();
}

void CNPC::SetXY()
{
    x = currentX;
    y = currentY;
    int dx = CCamera::Instance()->GetX();
    x = currentX - dx;
    animation.SetTopLeft(x, y);
}

void CNPC::LookRole(CPoint rolePoint)
{
    if (currentX < rolePoint.x && faceTo != "right") //look right
    {
        rightAnimate.CopyAnimateInformation(&animation);
        animation = rightAnimate;
        faceTo = "right";
    }
    else if (currentX >= rolePoint.x && faceTo != "left")
    {
        leftAnimate.CopyAnimateInformation(&animation);
        animation = leftAnimate;
        faceTo = "left";
    }
}

void CNPC::OnCycle(CPoint rolePoint)
{
    LookRole(rolePoint);
    OnMove();
}

void CNPC::OnMove()
{
    int dx = CCamera::Instance()->GetX();
    SetCurrentXY(currentX, currentY);
    animation.OnMove();
}

CNPC::~CNPC()
{
}
#pragma endregion

#pragma region - CNPC1 - No.1 -
CNPC1::CNPC1()
{
}

CNPC1::CNPC1(CPoint _point, BitmapPath _loadPath, string _id, string _txt, double resetTime) : CNPC(_point,
```

```
_loadPath, _id, resetTime)
    {
        txt = _txt;
    }

    CNPC1::~CNPC1()
    {
    }

    void CNPC1::RoleCollision()
    {
        CDialogManager::Instance()->Start(txt);
    }
    #pragma endregion

    #pragma region - CNPC3 - No.3 -
    CNPC3::CNPC3()
    {
    }

    CNPC3::CNPC3(CPoint _point, BitmapPath _loadPath, string _id, string _music, string _txt) : CNPC(_point,
_loadPath, _id, 0.1)
    {
        openMusic = _music;
        openTxt = _txt;
        isLoadHeadPhoneIcon = false;
    }

    CNPC3::~CNPC3()
    {
    }

    void CNPC3::RoleCollision()
    {
        CAudio::Instance()->Stop("SLoWMoTIoN_Game");
        CAudio::Instance()->Play(openMusic, true);
        CDialogManager::Instance()->OpenMusicPlayer(openMusic);
        CDialogManager::Instance()->Start(openTxt);
    }

    void CNPC3::Initialize()
    {
        CNPC::Initialize();
        if (!isLoadHeadPhoneIcon)
        {
            headphoneIcon.LoadBitmap("RES\\NPC\\headphone3.bmp", RGB(255, 255, 255));
            isLoadHeadPhoneIcon = true;
        }
        headphoneIcon.SetTopLeft(GetX3() - headphoneIcon.Width() / 2, GetY1() - headphoneIcon.Height() - 10);
        headphoneIcon.SetValid(false);
        CLayerManager::Instance()->AddObject(&headphoneIcon, layer.GetLayer());
    }

    void CNPC3::SetValid(bool flag)
    {
        animation.SetValid(flag);
        headphoneIcon.SetValid(flag);
    }

    void CNPC3::OnMove()
    {
        CNPC::OnMove();
        headphoneIcon.SetTopLeft(GetX3() - headphoneIcon.Width() / 2, GetY1() - headphoneIcon.Height() - 10);
    }
    #pragma endregion
}
```

5. CBall.h

```
#pragma once
#include "CLibrary.h"
#include "CEraser.h"
#include "CBoss.h"
namespace game_framework {
  #pragma region - ball -
  class CBall
  {
    public:
      CBall();
      bool IsAlive(); // 是否活著
      void SetCurrentXY(double, double);
      void SetXY(int nx, int ny); // 設定圓心的座標
      void SetIsAlive(bool alive); // 設定是否活著
      CPoint GetCenterPoint() {
        return CPoint(x + animation.Width() / 2, y + animation.Height() / 2);
      };
      void LoadBitmap(string, string, int);
      void LoadBitmap(BitmapPath);
      CAnimate* GetAnimate();
      CLayer layer;
    protected:
      CAnimate animation;
      double currentX, currentY;   //當前的座標
      int x, y; // 圓心的座標
      bool is_alive;  // 是否活著
  };
  #pragma endregion
  #pragma region - CScallion -
  class CScallion : public CBall
  {
    public:
      CScallion();
      CScallion(BitmapPath, CPoint, CPoint, int = 2);      //給兩 CPoint + int 為參數時，第一個為目前座標，第二個為滑鼠座標，最後為重力參數
      CScallion(BitmapPath, CPoint, int, int); //若不給重力參數，第一個 CPoint 不變，但後面變為「x 初速, y 初速」
      void OnMove();
      void OnShow();
      void SetInitVelocity(int, int, int, int, int = 5);
      void SetInitVelocity(int, int);
      bool IsCollision(CMonster);
      bool IsCollision(CMonster*);
      bool IsCollision(CBoss*);
      bool IsCollision(CRect);
      void Clear();
      void Initialize(CPoint);
      int    GetAtk() {
        return atk;
      };
      void SetAtk(int k) {
        atk = k;
      };
    protected:
      int atk = 1;
      int velocity_x;
      int velocity_y;
      int gravity;
      int direction; //丟出時面對的方向
  };
  #pragma endregion
  #pragma region - CBlackHole -
  class CBlackHole : public CScallion
  {
    public:
      CBlackHole();
      CBlackHole(BitmapPath, CPoint, CPoint, int = 4); //給兩 CPoint + int 為參數時，第一個為目前座標，第二個為滑鼠座標，最後為重力參數
      CBlackHole(BitmapPath, CPoint, int, int); //若不給重力參數，第一個 CPoint 不變，但後面變為「x 初速, y 初速」
```

```
        CRect GetCenterRect() {
            return centerRect;
        };
        void OnMove();
        void SetRole(CRole* _r) {
            role = _r;
        };
        void Initialize();
    private:
        CTimer selfBang; //自爆 timer
        CRect centerRect;
        CRect centerRectSize;
        CRole* role;
        int csize = 100;
};
#pragma endregion
#pragma region - CRay -
class CRay
{
    public:
        CRay();
        CRay(BitmapPath, CPoint, double = 0.1); //動畫路徑, 起始位置, 每個影格間隔時間
        ~CRay();
        void OnMove();
        bool GetValid() {
            return animation.GetValid();
        };
        void SetValid(bool f) {
            animation.SetValid(f);
        };
        CAnimate* GetAnimate() {
            return &animation;
        };
        void SetXY(int _x, int _y) {
            x = _x;
            y = _y;
            animation.SetTopLeft(x, y);
        }
        bool GetAttackValid() {
            return AttackValid;
        };
    private:
        CAnimate animation;
        int x, y;
        bool AttackValid; //攻擊有效
};
#pragma endregion
}
```

6. CBall.cpp

```
#pragma once
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "CBall.h"
#include "CEraser.h"
#include "CManager.h"

using namespace myLibrary;

namespace game_framework {

    #pragma region - ball -

    CBall::CBall()
```

```
{
   is_alive = true;
}

bool CBall::IsAlive()
{
   return is_alive;
}

void CBall::SetIsAlive(bool alive)
{
   is_alive = alive;
}

void CBall::SetXY(int nx, int ny)
{
   x = nx;
   y = ny;

   if (!animation.IsNull())
      animation.SetTopLeft(x, y);
}

void CBall::SetCurrentXY(double cx, double cy)
{
   currentX = cx;
   currentY = cy;
   int dx = CCamera::Instance()->GetX();
   int nx = (int)(currentX) - dx;
   int ny = (int)(currentY);
   SetXY(nx, ny);
}

void CBall::LoadBitmap(string ziliaojia, string name, int number)
{
   animation.LoadBitmap(ziliaojia, name, number);
}

void CBall::LoadBitmap(BitmapPath loadpath)
{
   animation.LoadBitmap(loadpath.ziliaojia, loadpath.name, loadpath.number, loadpath.color);
}

CAnimate* CBall::GetAnimate()
{
   return &animation;
}

#pragma endregion

#pragma region - CScallion -
CScallion::CScallion()
{
   const int INIT_X = 0, INIT_Y = 0;
   const int GRAVITY = 4;
   currentX = INIT_X;
   currentY = INIT_Y;
   x = INIT_X;
   y = INIT_Y;
   gravity = GRAVITY;
   layer.SetLayer(BULLET_LAYER);
   animation.SetTopLeft(INIT_X, INIT_Y);
   is_alive = true;
   LoadBitmap("Role", "scallion", 4);
   CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());
}

CScallion::CScallion(BitmapPath loadpath, CPoint point, CPoint finalPoint, int _gravity) //gravity 預設 4 可不指定
{
```

```
    gravity = _gravity;
    LoadBitmap(loadpath);
    Initialize(point);
    SetInitVelocity(point.x, point.y, finalPoint.x, finalPoint.y);
}

CScallion::CScallion(BitmapPath loadpath, CPoint point, int initV_x, int initV_y)
{
    gravity = 0;
    LoadBitmap(loadpath);
    Initialize(point);
    velocity_x = initV_x;
    velocity_y = initV_y;
}

void CScallion::Clear()
{
    this->~CScallion();
}

void CScallion::Initialize(CPoint initPos)
{
    const int INIT_X = initPos.x;
    const int INIT_Y = initPos.y;
    currentX = x = INIT_X;
    currentY = y = INIT_Y;
    is_alive = true;
    animation.SetTopLeft(x, y);
    animation.ResetDelayTime(0.1);
    layer.SetLayer(BULLET_LAYER);
    CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());
}

void CScallion::OnMove()
{
    if (!IsAlive())
        return;

    if (x > SIZE_X || x < 0 || y > SIZE_Y || y < -300) //超出螢幕
    {
        SetIsAlive(false);
        animation.SetValid(false);
        return;
    }

    #pragma region - 重力計算 -

    if (velocity_y > 0)
    {
        velocity_y -= gravity;
        y -= velocity_y;
    }
    else
    {
        velocity_y -= gravity;
        y -= velocity_y;
    }

    #pragma endregion
    animation.OnMove();
    SetXY(x + velocity_x, y);
}

void CScallion::OnShow()
{
    if (!IsAlive())
        return;

    animation.SetTopLeft(x, y);
```

```
    animation.OnShow();
}

void CScallion::SetInitVelocity(int vx, int vy)
{
    velocity_x = vx;
    velocity_y = vy;
}

void CScallion::SetInitVelocity(int b_x, int b_y, int f_x, int f_y, int fix_velocity)
{
    int dx = (f_x - b_x) / fix_velocity / 2;
    int dy = -(f_y - b_y) / fix_velocity / 2;
    velocity_x = dx;
    velocity_y = dy;
}

bool CScallion::IsCollision(CMonster passerby)
{
    return false;
    return IsPointInRect(CPoint(x + animation.Width(), y), passerby.animation.GetRect());
}

bool CScallion::IsCollision(CMonster* passerby)
{
    return IsRectCollision(animation.GetRect(), passerby->animation.GetRect());
}
bool CScallion::IsCollision(CBoss* boss)
{
    return IsRectCollision(animation.GetRect(), boss->GetAnimate()->GetRect());
}
bool CScallion::IsCollision(CRect rect)
{
    return IsRectCollision(animation.GetRect(), rect);
}
#pragma endregion

#pragma region - CBlackHole -
CBlackHole::CBlackHole() : CScallion()
{
}

CBlackHole::CBlackHole(BitmapPath loadpath, CPoint point, CPoint finalPoint, int _gravity) : CScallion(loadpath,
point, finalPoint, _gravity)
{
    SetInitVelocity(point.x, point.y, finalPoint.x, finalPoint.y, 25);
    Initialize();
}

CBlackHole::CBlackHole(BitmapPath loadpath, CPoint point, int initV_x, int initV_y) : CScallion(loadpath, point,
initV_x, initV_y)
{
    Initialize();
}

void CBlackHole::Initialize()
{
    centerRectSize = CRect(0, 0, csize, csize);
    role = NULL;
    selfBang = CTimer(1.8);
    SetIsAlive(true);
}

void CBlackHole::OnMove()
{
    selfBang.CountDown();

    if (selfBang.IsTimeOut())
    {
```

```
        SetIsAlive(false);
        role = NULL;
        return;
    }

    if (y + animation.Height() < SIZE_Y)
    {
        if (!IsAlive())
            return;

        if (x > SIZE_X || x + animation.Width() + 87 < 0 || y > SIZE_Y || y < -300)//超出螢幕
        {
            SetIsAlive(false);
            animation.SetValid(false);
            return;
        }

        #pragma region -  重力計算  -

        if (velocity_y > 0)
        {
            velocity_y -= gravity;
            currentY -= velocity_y;
        }
        else
        {
            velocity_y -= gravity;
            currentY -= velocity_y;
        }

        #pragma endregion
        animation.OnMove();
        SetCurrentXY(currentX + velocity_x, currentY);
    }

    animation.OnMove();
    int centerRectWidth = centerRectSize.right;
    int centerRectHeight = centerRectSize.bottom;
    int centerPointX = x + animation.Width() / 2;
    int centerPointY = y + animation.Height() / 2;
    centerRect = CRect(centerPointX - centerRectWidth / 2,      //left
                       centerPointY - centerRectHeight / 2,    //top
                       centerPointX + centerRectWidth / 2,      //right
                       centerPointY + centerRectHeight / 2); //bottom

    if (role != NULL)
    {
        int ddrsx = (role->GetX3() - centerPointX) / 8;
        role->SetXY(role->GetX1() - ddrsx, role->GetY1());
    }
}
#pragma endregion

#pragma region - CRay -
CRay::CRay()
{
}

CRay::CRay(BitmapPath loadPath, CPoint Fpoint, double resetTime)
{
    animation.LoadBitmap(loadPath.ziliaojia, loadPath.name, loadPath.number, loadPath.color);
    SetValid(false);
    SetXY(Fpoint.x, Fpoint.y);
    AttackValid = false;
    animation.ResetDelayTime(resetTime);
    CLayerManager::Instance()->AddObject(&animation, 6);
}

CRay::~CRay()
```

```
    {
    }

    void CRay::OnMove()
    {
        animation.SetValid(true);

        if (animation.GetIndex() < animation.GetIndexSize() - 1)
        {
            animation.OnMove();
        }
        else
        {
            AttackValid = true;
        }
    }
    #pragma endregion
}
```

7. CBoss.h

```
#pragma once
#include "CLibrary.h"
#include "Refactor.h"
#include <vector>
using namespace myLibrary;
namespace game_framework
{
    #pragma region - CBoss -
    class CScallion;
    class CRole;
    class CBlackHole;
    class CBoss
    {
        public:
            CBoss();
            CBoss(int, int, int, string, BitmapPath); //x, y, hp, (路徑, 顏色)
            virtual ~CBoss();
            CPoint GetLeftTopPoint() {
                return CPoint(currentX, currentY);
            };
            CPoint GetCenterPoint() {
                return CPoint(currentX + width / 2, currentY + height / 2);
            };
            CPoint GetRightBottomPoint() {
                return CPoint(currentX + width, currentY + height);
            };
            void LoadBitmap();
            void virtual Initialize();
            void InitializeDirAnimate(string, double = 0.1);
            void SetFaceTo(CPoint);
            void SetXY(int, int);
            void SetCurrentXY(int, int);
            void SetHp(int);
            void SetIsAlive(bool);
            bool GetAlive() {
                return IsAlive;
            };
            string GetID() {
                return id;
            };
            int GetHp() {
                return hp;
            };
            int GetInitHp() {
                return initHp;
            };
            void OnMove();
            void MoveWithMap(string);
```

```
        void CollisionScallion(CRole*);
        virtual void OnCycle(CRole*) {};
        virtual void Attack(CRole*) {};
        virtual void Clear() {};
        virtual bool InEndProcess() {
            return false;
        };
        virtual void EndProcess() {};
        bool IsEnd() {
            return isEnd;
        };
        virtual vector<CScallion*>* GetBullet() {
            return nullptr;
        };
        bool IsDead() {
            return !IsAlive;
        };
        CTimer* GetAliveTimer() {
            return &AliveTime;
        };
        CLayer layer;
        CAnimate* GetAnimate();
        string GetBossId() {
            return id;
        };
    protected:
        #pragma region Init boss Information
        int initx, inity, initHp;
        BitmapPath loadPath;
        COLORREF transparentColor;
        #pragma endregion
        CAnimate animation;
        CAnimate leftAnimate;
        CAnimate rightAnimate;
        string faceTo;
        int hp;
        int currentX, currentY;
        int x, y;
        int width, height;
        bool IsAlive = true;
        string id;
        CTimer AliveTime;
        CTimer ChangeFaceTimer;
        const double PI = 4 * atan(1.0);
        bool isEnd;
};
#pragma endregion
#pragma region - CXingting -
class CXingting : public CBoss
{
    public:
        CXingting();
        CXingting(int, int, int, string, BitmapPath);
        ~CXingting();
        void Initialize();
        void OnCycle(CRole*);
        void Attack(CRole*);
        void OnMove();
        void Clear();
        void ClearBullet();
        bool IsDead();
    private:
        vector<CScallion*>    level4;
        vector<CBlackHole*> blackhole;
        CBlackHole* targetBlackhole;
        void Level4Collision(CRole*);
        CTimer shootLevel4_cd;
        CTimer shoot_atk2_cd;
        CTimer shoot_atk3_cd;
```

```
        CTimer moveToGoal;
        CTimer mode_Attack2_timer;
        CTimer mode_Attack4_CreateBlackHole;
        int goal_x, goal_y;
        int angle_atk2;
        double mode4_AttackTime;
        int mode4_RoleMoveDir; // 1 = 左 2 = 右 3 = 不能動
        void Attack1();
        void Attack2();
        void Attack3();
        void Attack4(CRole*);
        bool moveToGoalPoint;
        bool mode_Attack1;
        bool mode_Attack2;
        bool mode_Attack3;
        bool mode_Attack4;
        int atkCounter;
    };
    #pragma endregion
    #pragma region - CFaicaiSeed -
    class CRay;
    class CFacaiSeed : public CBoss
    {
      public:
        CFacaiSeed();
        CFacaiSeed(int, int, int, string, BitmapPath);
        ~CFacaiSeed();
        void Initialize();
        void OnCycle(CRole*);
        void Attack(CRole*);
        void OnMove();
        void Clear();
        bool InEndProcess() {
          return fly;
        };
        void EndProcess();
      private:
        void Attack1(CRole*);
        void Attack2();
        void Attack3(); //Ray attack
        void ClearBullet();
        void Collision(CRole*);
        CPoint GetCreateCoinPoint(); //射出金幣的起始點
        CTimer movingTime;
        CTimer shootCoinTimer;
        CTimer attackRoleTimer; //role 硬值時間
        CTimer subRoleHp_NoEQ;
        bool fly;
        int flyY = 1;
        vector<CScallion*>   coinVector;
        int coinAngle[4] = { 45, 75, 105, 135 };
        #pragma region - ray -
        CTimer rayStartTime;
        CTimer rayStayTime;
        CRay* ray = NULL;
        #pragma endregion
    };
    #pragma endregion
}
```

8. CBoss.cpp

```
#pragma once
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
```

```
#include "CEraser.h"
#include "CBall.h"
#include <sstream>
#include "CManager.h"
#include "CLibrary.h"
#include "CBoss.h"
#include <vector>
using namespace myLibrary;

namespace game_framework
{
  #pragma region - CBoss -
  CBoss::CBoss()
  {
    initHp = 0;
    initx = 0;
    inity = 0;
  }

  CBoss::CBoss(int _x, int _y, int _hp, string _id, BitmapPath _loadPath)
  {
    initx = _x;
    inity = _y;
    initHp = _hp;
    loadPath = _loadPath;
    id = _id;
  }

  CBoss::~CBoss()
  {
  }

  void CBoss::LoadBitmap()
  {
    #pragma region - Load animateion and left, right animation -
    BitmapPath leftPath = loadPath;
    BitmapPath rightPath = loadPath;
    leftPath.ziliaojia += "\\L";
    rightPath.ziliaojia += "\\R";
    animation.LoadBitmap(leftPath.ziliaojia, leftPath.name, leftPath.number, leftPath.color);
    rightAnimate.LoadBitmap(rightPath.ziliaojia, rightPath.name, rightPath.number, rightPath.color);
    #pragma endregion
    animation.SetTopLeft(initx, inity);
    height = animation.Height();
    width = animation.Width();
    leftAnimate = animation;
  }

  void CBoss::Initialize()
  {
    #pragma region - 僅一次的載入圖片 -

    if (animation.IsNull())
    {
      LoadBitmap();
    }

    #pragma endregion
    inity = SIZE_Y - animation.Height();
    SetCurrentXY(initx, inity);
    SetXY(initx, inity);
    hp = initHp;
    layer.SetLayer(BOSS_LAYER);
    IsAlive = true;
    isEnd = false;
    animation.SetValid(false);
    CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());
    ChangeFaceTimer = CTimer(1.0);
    AliveTime = CTimer(99.0);
```

```
}

void CBoss::InitializeDirAnimate(string dir, double resetTime)
{
   animation.ResetDelayTime(resetTime);
   animation.SetTopLeft(initx, inity);
   leftAnimate.ResetDelayTime(resetTime);
   rightAnimate.ResetDelayTime(resetTime);
   leftAnimate.CopyAnimateInformation(&animation);
   rightAnimate.CopyAnimateInformation(&animation);
   faceTo = dir;

   if (faceTo == "left")
      animation = leftAnimate;
   else if (faceTo == "right")
      animation = rightAnimate;
}

void CBoss::SetFaceTo(CPoint point)
{
   if (point.x >= GetCenterPoint().x) //right
   {
      rightAnimate.CopyAnimateInformation(&animation);
      animation = rightAnimate;
      faceTo = "right";
   }
   else
   {
      leftAnimate.CopyAnimateInformation(&animation);
      animation = leftAnimate;
      faceTo = "left";
   }
}

void CBoss::SetXY(int _x, int _y)
{
   x = _x;
   y = _y;
   animation.SetTopLeft(x, y);
}

void CBoss::SetCurrentXY(int _x, int _y)
{
   currentX = _x;
   currentY = _y;
}

void CBoss::SetHp(int _hp)
{
   hp = _hp;
}

void CBoss::SetIsAlive(bool flag)
{
   IsAlive = flag;
}

void CBoss::MoveWithMap(string dir)
{
   if (dir == "left")
      x += MOVE_DISTANCE;

   if (dir == "right")
      x -= MOVE_DISTANCE;

   SetXY(x, y);
}

void CBoss::CollisionScallion(CRole* role)
```

```
{
    #pragma region - scallion collision this -
    vector<CScallion*>* scallion = role->GetScallion();

    for (vector<CScallion*>::iterator scallionk = scallion->begin(); scallionk != scallion->end();)
    {
        if ((*scallionk)->IsAlive() && (*scallionk)->IsCollision(this))
        {
            if (GetID() != "Xingting")
                CAudio::Instance()->Play("hit");

            (*scallionk)->SetIsAlive(false);
            hp -= (*scallionk)->GetAtk();
        }

        if (!(*scallionk)->IsAlive())
        {
            delete *scallionk;
            *scallionk = NULL;
            scallionk = scallion->erase(scallionk);
        }
        else
        {
            scallionk++;
        }
    }

    #pragma endregion
}

void CBoss::OnMove()
{
    int dx = CCamera::Instance()->GetX();
    SetXY(currentX - dx, currentY);
    animation.OnMove();
}

CAnimate* CBoss::GetAnimate()
{
    return &animation;
}
#pragma endregion

#pragma region - CXingting -
CXingting::CXingting()
{
}

CXingting::CXingting(int _x, int _y, int _hp, string _id, BitmapPath _loadPath) : CBoss(_x, _y, _hp, _id,    _loadPath)
{
}

CXingting::~CXingting()
{
    ClearBullet();
}

void CXingting::Initialize()
{
    CBoss::Initialize();
    ClearBullet();
    targetBlackhole = NULL;
    goal_x = 320 - width / 2;
    goal_y = 0;
    shootLevel4_cd = CTimer(0.75);
    shoot_atk2_cd = CTimer(0.033);
    shoot_atk3_cd = CTimer(2.5);
    moveToGoal = CTimer(0.04);
    mode_Attack2_timer = CTimer(8.0);
```

```
    mode_Attack4_CreateBlackHole = CTimer(0.5);
    mode4_AttackTime = 1.8; //mode4 時 每 1.8 發射一顆黑洞
    angle_atk2 = 0;
    mode4_RoleMoveDir = 0;
    moveToGoalPoint = true;
    mode_Attack1 = true;
    mode_Attack2 = false;
    mode_Attack3 = false;
    mode_Attack4 = false;
    atkCounter = 0;
    AliveTime = CTimer(40.0);
}

void CXingting::OnCycle(CRole* role)
{
    if (AliveTime.IsTimeOut()) //時間到 杏庭自爆
    {
        if (IsAlive)
        {
            CDialogManager::Instance()->Start(DIALOG_DATA_VSXingting3);
            IsAlive = false;
            isEnd = true;
        }
    }

    if (!IsDead())
    {
        OnMove();
        Attack(role);
    }
}

void CXingting::OnMove()
{
    if (moveToGoalPoint)
    {
        int move_dx = (currentX > goal_x) ? (initx - goal_x) / 50 : 0;
        int move_dy = (currentY > goal_y) ? (inity - goal_y) / 50 : 0;
        moveToGoal.CountDown();

        if (moveToGoal.IsTimeOut())
        {
            SetCurrentXY(currentX - move_dx, currentY - move_dy);
            moveToGoal.ResetTime();
        }

        if (currentX <= goal_x && currentY <= goal_y)
        {
            moveToGoalPoint = false;
            mode_Attack1 = false;
            mode_Attack2 = true;
            mode_Attack3 = true;
            atkCounter = 0;
            CDialogManager::Instance()->Start(DIALOG_DATA_VSXingting2);
        }
    }

    int dx = CCamera::Instance()->GetX();
    SetXY(currentX - dx, currentY);
    animation.OnMove();
}

void CXingting::Attack(CRole* role)
{
    if (mode_Attack1)
    {
        Attack1();
    }
```

```
    if (mode_Attack2)
    {
        Attack2();
        mode_Attack2_timer.CountDown();

        if (mode_Attack2_timer.IsTimeOut())
        {
            mode_Attack2 = false;
            mode_Attack3 = false;
            mode_Attack4 = true;
        }
    }

    if (mode_Attack3)
    {
        Attack3();
    }

    if (mode_Attack4)
    {
        Attack4(role);
    }

    Level4Collision(role);
}

void CXingting::Attack1()
{
    shootLevel4_cd.CountDown();

    if (shootLevel4_cd.IsTimeOut())
    {
        for (int i = 0; i < 12; i++)
        {
            double speed = 15.0 / 2;
            int angle = (-150 + atkCounter * 15) + i * 30;
            double mx = sin(angle * (PI / 180.0)) * speed;
            double my = cos(angle * (PI / 180.0)) * speed;
            int drs = 3; //給一個倍數讓 books 不從中心點出現
            CPoint center = GetCenterPoint() + CPoint((int)mx * drs, -(int)my * drs);
            CScallion* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4), center, CPoint(0, 0), 0);
//先創建一個蔥的物件
            newlevel4->SetInitVelocity((int)mx, (int)my);
            level4.push_back(newlevel4); //將蔥放進 vector
            shootLevel4_cd.ResetTime();
        }
    }
}

void CXingting::Attack2()
{
    shoot_atk2_cd.CountDown();

    if (shoot_atk2_cd.IsTimeOut())
    {
        #pragma region - bullet parameter -
        double speed = 15.0 / 2;
        int angle = angle_atk2;
        double mx = sin(angle * (PI / 180.0)) * speed;
        double my = cos(angle * (PI / 180.0)) * speed;
        int drs = 3; //用意在給一個倍數讓 books 不從中心點出現
        CPoint center = GetCenterPoint() + CPoint((int)mx * drs, -(int)my * drs);
        #pragma endregion
        CAudio::Instance()->Play("blackhole");
        CScallion* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4), center, CPoint(0, 0), 0); //
先創建一個蔥的物件
        newlevel4->SetInitVelocity((int)mx, (int)my);
        level4.push_back(newlevel4); //將蔥放進 vector
        shoot_atk2_cd.ResetTime();
```

```
            angle_atk2 += 31;
        }
    }

    void CXingting::Attack3()
    {
        shoot_atk3_cd.CountDown();

        if (shoot_atk3_cd.IsTimeOut())
        {
            for (int i = 0; i < 18; i++)
            {
                #pragma region - bullet parameter -
                double speed = 10.0 / 2;
                double angle = i * (360 / 18);
                double mx = sin(angle * (PI / 180.0)) * speed;
                double my = cos(angle * (PI / 180.0)) * speed;
                int drs = 3; //用意在給一個倍數讓 books 不從中心點出現
                CPoint center = GetCenterPoint() + CPoint((int)my * drs, -(int)mx * drs);
                #pragma endregion
                CScallion* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4), center, CPoint(0, 0), 0);
//先創建一個蔥的物件
                newlevel4->SetInitVelocity((int)my, (int)mx);
                level4.push_back(newlevel4); //將蔥放進 vector
                shoot_atk3_cd.ResetTime();
            }
        }
    }

    void CXingting::Attack4(CRole* role)
    {
        mode_Attack4_CreateBlackHole.CountDown();

        if (mode_Attack4_CreateBlackHole.IsTimeOut())
        {
            CAudio::Instance()->Play("blackhole");
            CPoint center = GetCenterPoint();
            CBlackHole* newBh = new CBlackHole(BitmapPath("RES\\Object\\blackhole", "blackhole", 1, RGB(0, 0, 0)),
center + CPoint(-75, -68), CPoint(role->GetX1(), role->GetY1()), 0); //先創建一個蔥的物件
            blackhole.push_back(newBh);
            mode_Attack4_CreateBlackHole.ResetTime(mode4_AttackTime); //reset
        }
    }

    void CXingting::Clear()
    {
        ClearBullet();
        animation.SetValid(false);
    }

    void CXingting::ClearBullet()
    {
        for (vector<CBlackHole*>::iterator bkiter = blackhole.begin(); bkiter != blackhole.end(); bkiter++)
        {
            delete *bkiter;
            *bkiter = NULL;
        }

        for (vector<CScallion*>::iterator level4iter = level4.begin(); level4iter != level4.end(); level4iter++)
        {
            delete *level4iter;
            *level4iter = NULL;
        }

        level4.clear();
        blackhole.clear();
    }

    bool CXingting::IsDead()
```

```
{
    return AliveTime.IsTimeOut();
}

void CXingting::Level4Collision(CRole* role)
{
    for (vector<CBlackHole*>::iterator bkiter = blackhole.begin(); bkiter != blackhole.end();)
    {
        (*bkiter)->OnMove();

        if ((*bkiter)->IsAlive() && !role->IsCatched() && role->IsCollisionBlackHole((*bkiter)))
        {
            targetBlackhole = *bkiter;
            role->SetCatched(true);
            (*bkiter)->SetRole(role);
        }

        if (!(*bkiter)->IsAlive())
        {
            #pragma region - Create Bullet -
            int bulletNumber = 12;

            for (int i = 0; i < bulletNumber; i++)
            {
                #pragma region - bullet parameter -
                double speed = 15.0 / 2;
                double angle = i * (360 / bulletNumber);
                double mx = sin(angle * (PI / 180.0)) * speed;
                double my = cos(angle * (PI / 180.0)) * speed;
                int drs = 3; //亂取的名字 用意在給一個倍數讓 books 不從中心點出現
                CPoint center = GetCenterPoint() + CPoint((int)my * drs, -(int)mx * drs);
                #pragma endregion
                CScallion* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4),
(*bkiter)->GetCenterPoint(), CPoint(0, 0), 0); //先創建一個蔥的物件
                newlevel4->SetInitVelocity((int)mx, (int)my);
                level4.push_back(newlevel4); //將蔥放進 vector
            }

            #pragma endregion
            targetBlackhole = NULL;
            role->SetCatched(false);
        }

        if (!(*bkiter)->IsAlive())
        {
            delete *bkiter;
            *bkiter = NULL;
            bkiter = blackhole.erase(bkiter);
        }
        else
        {
            bkiter++;
        }
    }

    for (vector<CScallion*>::iterator level4iter = level4.begin(); level4iter != level4.end();)
    {
        (*level4iter)->OnMove();

        if ((*level4iter)->IsAlive() && role->IsCollisionLevel4(*level4iter))
        {
            (*level4iter)->SetIsAlive(false);
            CAudio::Instance()->Play("role_hitted");
            role->SubHp();
        }

        if (!(*level4iter)->IsAlive())
        {
            delete *level4iter;
```

```
            *level4iter = NULL;
            level4iter = level4.erase(level4iter);
        }
        else
        {
            level4iter++;
        }
    }
}

#pragma endregion

#pragma region - CFaicaiSeed -
CFacaiSeed::CFacaiSeed()
{
    ray = NULL;
}

CFacaiSeed::CFacaiSeed(int _x, int _y, int _hp, string _id, BitmapPath _loadPath) : CBoss(_x, _y, _hp, _id, _loadPath)
{
    ray = NULL;
}

CFacaiSeed::~CFacaiSeed()
{
    Clear();
}

void CFacaiSeed::Initialize()
{
    CBoss::Initialize();
    InitializeDirAnimate("right", 0.05);
    #pragma region - Init ray -
    ray = NULL;
    rayStartTime.ResetTime(2.5); //預設 k 秒後發射
    rayStayTime.ResetTime(2.0); //預設持續兩秒
    #pragma endregion
    Clear();
    fly = false;
    flyY = 1;
    movingTime = CTimer(0.2);
    shootCoinTimer = CTimer(0.2); //每 0.2s 發射 1 波金幣
    attackRoleTimer = CTimer(0.4); //硬值時間 0.4s
    subRoleHp_NoEQ = CTimer(0.6); //每 0.6s 掉一次血
    AliveTime = CTimer(99.0);
}

void CFacaiSeed::OnCycle(CRole* role)
{
    #pragma region - boss dead -

    if (hp <= 0) //boss dead
    {
        CDialogManager::Instance()->Start("roleWinFacaiSeed");
        SetIsAlive(false);
    }

    #pragma endregion
    #pragma region - Boss Alive -

    if (!IsDead())
    {
        attackRoleTimer.CountDown();

        if (ray == NULL) //射線中不轉換
        {
            ChangeFaceTimer.CountDown();

            if (ChangeFaceTimer.IsTimeOut())
```

```
            {
                SetFaceTo(CPoint(role->GetX3(), role->GetY3()));
                ChangeFaceTimer.ResetTime();
            }
        }

        Attack(role);
        OnMove();
        Collision(role);
    }

    #pragma endregion
    #pragma region - role dead -

    if (role->IsDead() || AliveTime.IsTimeOut())
    {
        CDialogManager::Instance()->Start("roleLoseFacaiSeed");
        ClearBullet();
        fly = true;
    }

    #pragma endregion
}

void CFacaiSeed::Attack(CRole* role)
{
    Attack1(role);
    bool isTalking = CDialogManager::Instance()->GetDialogState();

    if (ray == NULL && !isTalking)
    {
        CAudio::Instance()->Play("ray");
        Attack2();
    }

    Attack3();
}

void CFacaiSeed::OnMove()
{
    #pragma region - rays animation -

    if (ray != NULL)
    {
        ray->OnMove();
    }

    #pragma endregion
    #pragma region - dont have ray - move -
    else
    {
        int dx = 1;

        if (faceTo == "right")
        {
            SetCurrentXY(currentX + dx, currentY);
        }
        else if (faceTo == "left")
        {
            SetCurrentXY(currentX - dx, currentY);
        }
    }

    #pragma endregion
    CBoss::OnMove();
}

void CFacaiSeed::Clear()
{
```

```
    ClearBullet();
    animation.SetValid(false);
}

void CFacaiSeed::EndProcess()
{
    #pragma region - role dead and this fly -

    if (fly)
    {
        CAudio::Instance()->Play("faacai_fly");
        SetCurrentXY(currentX, currentY - flyY);
        flyY++;
        CBoss::OnMove();

        if (GetRightBottomPoint().y <= -10)
        {
            fly = false;
            isEnd = true;
            Clear();
        }
    }

    #pragma endregion
}

void CFacaiSeed::Attack1(CRole* role)
{
    if (role->IsZZ())
    {
        subRoleHp_NoEQ.CountDown();

        if (subRoleHp_NoEQ.IsTimeOut())
        {
            CAudio::Instance()->Play("role_hitted");
            role->SubHp();
            subRoleHp_NoEQ.ResetTime();
        }
    }
}

void CFacaiSeed::Attack2()
{
    shootCoinTimer.CountDown();

    if (shootCoinTimer.IsTimeOut())
    {
        CAudio::Instance()->Play("door");

        for (int i = 0; i < 4; i++)
        {
            double speed = 27.0;
            int angle = coinAngle[i];
            double mx = sin(angle * (PI / 180.0)) * speed;
            double my = cos(angle * (PI / 180.0)) * speed;
            double drs = 2.5;
            CPoint center = GetCreateCoinPoint() + CPoint((int)(my * drs), -(int)(mx * drs));
            CScallion* newcoin = new CScallion(BitmapPath("RES\\Object\\coin", "coin", 4, RGB(255, 255, 255)), center,
CPoint(0, 0), 1); //先創建一個蔥的物件
            newcoin->SetInitVelocity((int)my, (int)mx);
            coinVector.push_back(newcoin);
            shootCoinTimer.ResetTime();
        }
    }
}

void CFacaiSeed::Attack3()
{
    rayStartTime.CountDown();
```

```
        if (rayStartTime.IsTimeOut())
        {
            if (ray == NULL)
            {
                if (faceTo == "right")
                {
                    ray = new CRay(BitmapPath("RES\\Object\\Ray", "ray", 5, RGB(214, 214, 214)),
CPoint(GetRightBottomPoint().x, GetLeftTopPoint().y + 40));
                }
                else
                {
                    ray = new CRay(BitmapPath("RES\\Object\\Ray", "ray", 5, RGB(214, 214, 214)),
CPoint(GetLeftTopPoint().x - 531, GetLeftTopPoint().y + 40));
                }
            }

            rayStayTime.CountDown(); //開始計算持續時間

            if (rayStayTime.IsTimeOut()) //delete ray
            {
                if (ray != NULL)
                {
                    delete ray;
                    ray = NULL;
                }

                rayStayTime.ResetTime();
                rayStartTime.ResetTime();
            }
        }
    }

    void CFacaiSeed::ClearBullet()
    {
        #pragma region - clear coin -

        for (vector<CScallion*>::iterator coiniter = coinVector.begin(); coiniter != coinVector.end(); coiniter++)
        {
            delete *coiniter;
            *coiniter = NULL;
        }

        coinVector.clear();
        #pragma endregion
        #pragma region - clear ray -

        if (ray != NULL)
        {
            delete ray;
            ray = NULL;
        }

        #pragma endregion
    }

    void CFacaiSeed::Collision(CRole* role)
    {
        #pragma region - coin collision role -

        for (vector<CScallion*>::iterator cointiter = coinVector.begin(); cointiter != coinVector.end();)
        {
            (*cointiter)->OnMove();

            if ((*cointiter)->IsAlive() && role->IsCollisionLevel4(*cointiter))
            {
                CAudio::Instance()->Play("role_hitted");
                (*cointiter)->SetIsAlive(false);
                role->SubHp();
```

```
            }

            if (!(*cointiter)->IsAlive())
            {
                delete *cointiter;
                *cointiter = NULL;
                cointiter = coinVector.erase(cointiter);
            }
            else
            {
                cointiter++;
            }
        }

        #pragma endregion
        #pragma region - ray collision role -

        if (ray != NULL)
        {
            if (ray->GetAttackValid() && role->IsCollisionRay(ray))
            {
                if (attackRoleTimer.IsTimeOut())
                {
                    CAudio::Instance()->Play("role_hitted");
                    role->SubHp();
                    attackRoleTimer.ResetTime();
                }
            }
        }

        #pragma endregion
        #pragma region - this collision role -

        if (IsRectCollision(role->GetAction()->GetRect(), animation.GetRect()))
        {
            if (attackRoleTimer.IsTimeOut())
            {
                role->SubEq();
                attackRoleTimer.ResetTime();
            }
        }

        #pragma endregion
        #pragma region - this collsion scallion -
        CollisionScallion(role);
        #pragma endregion
    }

    CPoint CFacaiSeed::GetCreateCoinPoint()
    {
        CPoint center = GetCenterPoint();
        int dx = 45;

        if (faceTo == "left")
        {
            center += CPoint(dx, 0);
        }
        else
        {
            center -= CPoint(dx, 0);
        }

        return center;
    }
    #pragma endregion
}
```

9. CBlockMap.h

```
#pragma once
#include <vector>
#include "CEraser.h"
using namespace std;
namespace game_framework
{
  #pragma region - CSimpleMapObj -
  class CSimpleMapObj
  {
    public:
      CSimpleMapObj() {
        path = "";
        x = y = 0;
      };
      CSimpleMapObj(string _path, int _x, int _y) {
        path = _path;
        x = _x;
        y = _y;
      };
      ~CSimpleMapObj() {};
      void SetXY(int _x, int _y, int camera = 0) {
        x = _x;
        y = _y;
        bmp.SetTopLeft(x - camera, y);
      };
      void LoadImg() {
        char* address = ConvertCharPointToString(path);
        bmp.LoadBitmap(address, RGB(214, 214, 214));
        delete address;
      };
      CMovingBitmap bmp;
      string path;
      int x, y; //實際位置上的 x y
  };
  #pragma endregion
  #pragma region - CBlock -
  class CBlock : public CSimpleMapObj
  {
    public:
      CBlock() : CSimpleMapObj() {};
      CBlock(string _path, int _x, int _y) : CSimpleMapObj(_path, _x, _y) {};
      ~CBlock() {};
  };
  #pragma endregion
  #pragma region - CDoor -
  class CBlockMap;
  class CDoor : public CSimpleMapObj
  {
    public:
      CDoor() : CSimpleMapObj() {};
      CDoor(string _path, int _x, int _y, string _type, int _deliveMapIndex) : CSimpleMapObj(_path, _x, _y) {
        type = _type;
        deliverMapIndex = _deliveMapIndex;
      };
      ~CDoor() {};
      int GetSwitchMapIndex() {
        return deliverMapIndex;
      }
      string GetType() {
        return type;
      };
    private:
      string type; //上 or 下
      int deliverMapIndex = -1; //要傳送的地圖
  };
  #pragma endregion
  #pragma region - CBlockMap -
  class CBlockMap
  {
```

```
    public:
        CBlockMap(); //default constructor
        CBlockMap(int);
        ~CBlockMap();
        void LoadImg();
        void LoadInformation(int);
        void LoadInformation(string);
        void CreateInformation();
        void CreateInformation(string);
        void Initialize(); //有用到的東西都歸 0
        bool isLoad = false;
        int nowMap; //目前地圖的編號
        int leftMap, rightMap, upMap, downMap; //目前地圖，其上下左右地圖的編號 (< 0 代表不存在)
        int loadMap; //載入的地圖資訊
        CMovingBitmap backgroundBitmap; //背景圖片 (用 LoadBitmap 讀取 loadMap)
        int passerbyMaxSize;
        vector<int> passerbyID = {0, 1};
        string ziliaojia, name;
        int number;
        string loadPath;
        vector<CBlock> block;
        vector<CDoor> door;
    private:
        void LoadMap(string); //讀取 string 的 txt
        void WriteMap(string); //寫下 string 的 txt
};
#pragma endregion
}
```

10. CBlockMap.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <vector>
#include "audio.h"
#include "gamelib.h"
#include <fstream>
#include <sstream>
#include "CBlockMap.h"
#include "CManager.h"
#include "CEraser.h"
using namespace std;
namespace game_framework
{
#pragma region - CBlockMap -
CBlockMap::CBlockMap()
{
    nowMap = 0;
    upMap = -1;
    downMap = -1;
    leftMap = -1;
    rightMap = -1;
    loadMap = IDB_MAP0;
    passerbyMaxSize = 0;
    loadPath = "RES\\Map\\IDB_Map_0.bmp";
}

CBlockMap::CBlockMap(int _nowMap)
{
    nowMap = _nowMap;
    LoadInformation(nowMap);
}

CBlockMap::~CBlockMap()
{
}
```

```
void CBlockMap::LoadImg()
{
    char* address = ConvertCharPointToString(loadPath);
    backgroundBitmap.LoadBitmap(address);
    delete address;
    for (vector<CBlock>::iterator blockiter = block.begin(); blockiter != block.end(); blockiter++)
    {
        blockiter->LoadImg();
    }
    for (vector<CDoor>::iterator dooriter = door.begin(); dooriter != door.end(); dooriter++)
    {
        dooriter->LoadImg();
    }
    isLoad = true;
}

void CBlockMap::LoadInformation(int mapIndex)
{
    string mapFilePath = "RES\\Map\\Information\\map" + std::to_string(mapIndex) + ".txt";
    LoadMap(mapFilePath);
}

void CBlockMap::LoadInformation(string mapFileName)
{
    string mapFilePath = "RES\\Map\\Information\\" + mapFileName; //含附檔名
    LoadMap(mapFilePath);
}

void CBlockMap::LoadMap(string mapFilePath)
{
    fstream mapData;
    mapData.open(mapFilePath);
    string lineData;
    while (getline(mapData, lineData))
    {
        vector<string> tempString = SplitString(lineData);
        #pragma region - load map information -
        if (tempString[0] == "background")
        {
            loadPath = tempString[1];
        }
        else if (tempString[0] == "block")
        {
            block.push_back(CBlock(tempString[1], ConvertStringToInt(tempString[2]),
ConvertStringToInt(tempString[3])));
        }
        else if (tempString[0] == "upDoor" || tempString[0] == "downDoor")
        {
            door.push_back(CDoor(tempString[1], ConvertStringToInt(tempString[2]), ConvertStringToInt(tempString[3]),
tempString[0], ConvertStringToInt(tempString[4])));
        }
        else if (tempString[0] == "up")
        {
            upMap = ConvertStringToInt(tempString[1]);
        }
        else if (tempString[0] == "down")
        {
            downMap = ConvertStringToInt(tempString[1]);
        }
        else if (tempString[0] == "left")
        {
            leftMap = ConvertStringToInt(tempString[1]);
        }
        else if (tempString[0] == "right")
        {
            rightMap = ConvertStringToInt(tempString[1]);
        }
        else if (tempString[0] == "passerbyNumber")
        {
```

```
            passerbyMaxSize = ConvertStringToInt(tempString[1]);
        }
        else if (tempString[0] == "passerbyID")
        {
            passerbyID.clear();
            for (unsigned int i = 1; i < tempString.size(); i++)
            {
                passerbyID.push_back(ConvertStringToInt(tempString[i]));
            }
        }
        #pragma endregion
    }
    mapData.close();
}

void CBlockMap::CreateInformation()
{
    string fileName = "RES\\Map\\Information\\map" + std::to_string(nowMap) + ".txt";
    WriteMap(fileName);
}

void CBlockMap::CreateInformation(string mapName) //含附檔名
{
    string fileName = "RES\\Map\\Information\\" + mapName;
    WriteMap(fileName);
}

void CBlockMap::Initialize()
{
    isLoad = false;
    nowMap = 0;
    leftMap = rightMap = upMap = downMap = -1;
    block.clear();
    door.clear();
    passerbyMaxSize = 0;
    loadPath = "";
}

void CBlockMap::WriteMap(string fileName)
{
    fstream data;
    data.open(fileName, ios::out);
    vector<string> r;
    r.push_back("background " + loadPath + " 0 0\n");
    #pragma region - write block -
    for (unsigned int i = 0; i < block.size(); i++)
    {
        r.push_back("block " + block[i].path + " " + std::to_string(block[i].x) + " " + std::to_string(block[i].y) + "\n");
    }
    #pragma endregion
    #pragma region - write door -
    for (unsigned int i = 0; i < door.size(); i++)
    {
        r.push_back(door[i].GetType() + " " + door[i].path + " " + std::to_string(door[i].x) + " " + std::to_string(door[i].y)
+ " " + std::to_string(door[i].GetSwitchMapIndex()) + "\n");
    }
    #pragma endregion
    r.push_back("up " + std::to_string(upMap) + "\n");
    r.push_back("down " + std::to_string(downMap) + "\n");
    r.push_back("left " + std::to_string(leftMap) + "\n");
    r.push_back("right " + std::to_string(rightMap) + "\n");
    r.push_back("passerbyNumber " + std::to_string(passerbyMaxSize) + "\n");
    string passID = "";
    for (unsigned int i = 0; i < passerbyID.size(); i++)
    {
        string tempid = " " + std::to_string(passerbyID[i]);
        passID += tempid;
    }
    r.push_back("passerbyID" + passID + "\n");
```

```
        for (unsigned int i = 0; i < r.size(); i++)
        {
            data << r[i];
        }
        data.close();
    }
    #pragma endregion
}
```

11. CLibrary.h

```
#pragma once
#include "Refactor.h"
namespace myLibrary
{
    char* ConvertCharPointToString(string, string, int = -1);
    char* ConvertCharPointToString(string);
    COLORREF ConvertStringToColor(string);
    bool ConvertStringToBoolen(string);
    void getFolderFile(string, vector<string>*);
    int getFolerFileNumber(string);
    string getFileName(string); //得到 file 的檔名(不含副檔名)
    vector<string> SplitString(string);
    void DeleteCharPoint(vector<char*>&);
    int GetPostive(int);
    int GetRandom(int, int); //得到 Random 數字，min <= k <= max
    int ConvertStringToInt(string);
    class BitmapPath
    {
        public:
            BitmapPath() {};
            BitmapPath(string _fold, string _name, int _number) {
                ziliaojia = _fold;
                name = _name;
                number = _number;
                color = RGB(255, 255, 255);
            };
            BitmapPath(string _fold, string _name, int _number, COLORREF _color) {
                ziliaojia = _fold;
                name = _name;
                number = _number;
                color = _color;
            };
            BitmapPath(vector<string> _path, COLORREF _color) {
                path = _path;
                color = _color;
            };
            ~BitmapPath() {};
            string ziliaojia;
            string name;
            int number;
            COLORREF color;
            vector<string> path;
    };
}
using namespace myLibrary;
namespace game_framework
{
    int ScreenX(int, int);
    bool IsPointInRect(CPoint, CRect);
    bool IsRectCollision(CRect, CRect);
    #pragma region - timer -
    class CTimer
    {
        public:
            CTimer();
            CTimer(int);
            CTimer(double);
            ~CTimer();
```

```
        void CountDown();
        double GetTime();
        int GetTime(int);
        bool IsTimeOut();
        void ResetTime(double);
        void ResetTime();
        void operator=(CTimer); //運算子多載，方便在 GameStateRun::OnBeginState 中重構 Timer
    private:
        const double reflash = (1000 / GAME_CYCLE_TIME); //一秒刷新幾次
        double time;
        double initTime;
};
#pragma endregion
#pragma region - CAnimate -
class CAnimate
{
    public:
        CAnimate();
        ~CAnimate();
        CMovingBitmap* AddBitmap(int, COLORREF = CLR_INVALID); // 增加一張圖形至動畫(圖的編號及透明色)
        CMovingBitmap* AddBitmap(char*, COLORREF = CLR_INVALID);    // 增加一張圖形至動畫(圖的編號及透明色)
        void LoadBitmap(vector<string>, COLORREF = CLR_INVALID);        //以字串陣列載入圖片
        void LoadBitmap(string, string, int, COLORREF = CLR_INVALID);
        void LoadBitmap(BitmapPath);
        int Height(); // 取得動畫的高度
        int Left(); // 取得動畫的左上角的 x 座標
        void OnMove(int); // 依照方向更換 bitmap
        void OnMove();
        void OnShow(); // 將動畫貼到螢幕
        void Reset(); // 重設播放順序回到第一張圖形
        void SetTopLeft(int, int);      // 將動畫的左上角座標移至 (x,y)
        int Top(); // 取得動畫的左上角的 y 座標
        int Width(); // 取得動畫的寬度
        bool IsNull();
        void ReleaseAnimate();
        void SetValid(bool);
        bool GetValid();
        void SetIndex(int);
        int GetIndex();
        int GetIndexSize() {
            return bmp.size();
        };
        void CopyAnimateInformation(CAnimate*);
        void ResetDelayTime(double);
        CRect GetRect();
    private:
        vector<CMovingBitmap> bmp;
        int bmp_amount;
        int x, y; // 動畫的座標
        int bmp_index;
        bool isValid;
        CTimer delayTimer;
        double delayTime;
};
#pragma endregion
#pragma region - CLayer -
class CLayer
{
    public:
        CLayer();
        ~CLayer();
        void SetLayer(int);
        int GetLayer();
    private:
        int layer;
};
#pragma endregion
```

```
#pragma region - CDialog -
class CDialog
{
   public:
      CDialog();
      CDialog(string, string);
      ~CDialog();
      bool GetTriggered();
      void SetTriggered();
      void Initialize();
      int GetTxtSize();
      string GetAvatar(unsigned int);
      string GetDialogTxt(unsigned int);
      COLORREF GetDialogColor(unsigned int);
      string GetMode();
      void LoadTxt();
   private:
      string path;
      string mode;
      vector<string> txt;
      vector<string> avatar;
      vector<COLORREF> color;
      bool IsTriggered;
      bool CanReTrigger;
};
#pragma endregion
#pragma region - Camera -
class CCamera
{
   public:
      CCamera();
      ~CCamera();
      int GetX();
      int GetY();
      void SetXY(int, int);
      void AddX(int);
      void AddY(int);
      void SetCanMoving(bool);
      void Initialize();
      void Reset();
      void SetCameraBoundary(int, int);
      static CCamera* Instance();
   private:
      int x, y;
      int max_left, max_right;
      bool canMoving;
      static CCamera camera;
};
#pragma endregion
#pragma region - CButton -
class CButton
{
      friend class CButtonManager;
   public:
      CButton();
      CButton(const CButton&);
      CButton(BitmapPath, CPoint, bool, bool); //路徑、初始點、初始狀態，是否需要與滑鼠碰撞
      int GetX();
      int GetY();
      void SetXY(int, int);
      void SetState(bool);
      bool GetState();
      int Width();
      int Height();
      void SetName(string _name) {
         name = _name;
      };
      string   GetName() {
         return name;
```

```
        };
        void LoadBitmap();
        void LoadBitmap(BitmapPath);
        void OnMove(); //更新 Button 狀態，有需要時也可更改顯示位置
        void OnShow();
        void Initialize();
        void Initialize(CPoint, bool);
        void SetValid(bool);
        bool GetValid();
        void CollisonMouse(CPoint);
        void ClickButton();
        bool IsCollisionMouse(CPoint);
CAnimate* GetAnimate();
        void operator=(const CButton&);
    private:
        CAnimate animation;
        BitmapPath loadpath;
        string name;
        int x, y;
        bool needCollision;
        bool state;
        bool valid;
};
#pragma endregion
#pragma region - CInteger -
class CInteger
{
    public:
        CInteger(int = 5); // default 5 digits
        void Initialize(CPoint, int, int = 2);
        void Add(int n); // 增加整數值
        int   GetInteger(); // 回傳整數值
        void LoadBitmap(string, string);
        void SetInteger(int); // 設定整數值
        void SetTopLeft(int, int);      // 將動畫的左上角座標移至 (x,y)
        void ShowBitmap(); // 將動畫貼到螢幕
        int NUMDIGITS;
        bool IsNull() {
            return !isBmpLoaded;
        };
        void SetValid(bool _flag) {
            isValid = _flag;
        };
        bool GetValid() {
            return isValid;
        };
        CMovingBitmap digit[11]; // 儲存 0..9 及負號之圖形(bitmap)
        CMovingBitmap number[4];
        CLayer layer;
    private:
        int   x, y; // 顯示的座標
        int   n; // 整數值
        bool isBmpLoaded; // 是否已經載入圖形
        bool isValid;
};
#pragma endregion
#pragma region - CAction -
class CAction
{
    public:
        CAction();
        void OnMove(string);
        void OnShow();
        void Initialize();
        void LoadAction(string, BitmapPath);          //action ,BitmapPath
        int Height(); // 取得動畫的高度
        int Width(); // 取得動畫的寬度
        int Left(); // 取得動畫的左上角的 x 座標
        int Top(); // 取得動畫的左上角的 y 座標
```

```
        void SetTopLeft(int, int);   // 將動畫的左上角座標移至 (x,y)
        bool IsNull();
        void SetAction(string);
        string GetAction();
        void SetValid(bool);
        bool GetValid();
        void SetIndex(int);
        int GetIndex();
        void SetFaceTo(string);
        string GetFaceTo();
        CLayer layer;
        CMovingBitmap* GetNowBitmap();
        CRect GetRect();
        int GetActionHeight(string);
    private:
        CMovingBitmap* nowBitmap;
        vector<CMovingBitmap>*    nowAction;
        map<string, vector<CMovingBitmap> >    paser;
        CTimer delayTimer;
        map <string, double> delayAdapter;
        double delay_run = 0.08;
        double delay_idle = 0.12;
        double delay_jump = 0.16;
        bool isBmpLoaded; // 是否已經載入圖形
        bool isValid;
        int x, y; // 顯示的座標
        int action_index;
        string action;
        string faceTo; // 目前面對方向
};
#pragma endregion
#pragma region - CEnd -
class CEnd
{
    public:
        CEnd();
        CEnd(string);
        ~CEnd();
        string GetBmpPath(int);
        string GetTxt(int);
        void SetGetEnd(bool f = true) {
            isGet = f;
        }
        bool IsGetEnd() {
            return isGet;
        };
    private:
        string endName;
        vector<string> bmpPath;
        vector<string> txt;
        void LoadEnd();
        void LoadBmpTxt(string);
        bool isGet;
};
#pragma endregion
#pragma region - CToumeiImage -
class CToumeiImage
{
    public:
        CToumeiImage();
        CToumeiImage(int, int);
        CToumeiImage(string, int, int);
        ~CToumeiImage();
        void SetAlpha(int _a);
        int GetAlpha() {
            return alpha;
        };
        void SetBmp(string _path);
        void SetFadeInOut(int, int);
```

```
        void FadeIn(); //淡入 每次 alpha+3
        void FadeOut(); //淡出 每次 alpha-4
        void DrawImage();
        bool temp = false;
    private:
        CBitmap bmp;
        string bmpLoadPath;
        int alpha = 0;
        int dFadeInValue = 3;
        int dFadeOutValue = -4;
};
#pragma endregion
#pragma region - CWindows -
#pragma region - CWindow -
class CWindow
{
        friend class CScrollWindow;
    public:
        CWindow();
        CWindow(CPoint);
        ~CWindow();
        void LoadResource();
        void Initialize(CPoint);
        void Clear();
        void Open();
        void Close();
        void UpdateMouse(CPoint _p) {
            mousePos = _p;
        };
        bool IsCollisionClose(CPoint);
        bool IsOpen();
        void SetCloseButton(CPoint);
        void SetXY(CPoint);
        void CollisionClose(CPoint);
        void OnCycle();
        void OnShow();
    protected:
        int x, y;
        CButton* closeButton = NULL;
        CMovingBitmap background;
        CPoint mousePos;
    private:
        bool isOpen;
        bool isLoaded;
};
#pragma endregion
#pragma region - CScrollWindow -
class CScrollWindow : public CWindow
{
    public:
        CScrollWindow();
        ~CScrollWindow();
        void LoadResource();
        void Initialize(CPoint);
        void Close();
        void OnCycle();
        void OnShow();
        void OnScrolling(short _s);
        string GetCollisionButtonName(CPoint);
    private:
        CMovingBitmap cover, cover_bottom;
        vector< vector <CButton> > endingVector;
        vector<string> endName;
        int colNum, rowNum;
        int img_x, img_y;
        int img_height, img_width;
        int limit_top, limit_buttom;
};
#pragma endregion
```

```
#pragma region - CSwitchWindow -
class CSwitchWindow : public CWindow
{
   public:
      CSwitchWindow();
      ~CSwitchWindow();
      void LoadResource(string);
      void Initialize(CPoint);
      void Open();
      void Close();
      bool CollisionArrow(CPoint);
      void ClickWindows(CPoint, string);
      void OnCycle();
   private:
      int index = 0;
      int step = 0;
      vector< vector<CMovingBitmap> > bmp;
      CMovingBitmap arrow_left;
      CMovingBitmap arrow_right;
      void SetIndex(int);
      void Switchwindow(string);
};
#pragma endregion
#pragma region - CPanel -
class CButtonManager;
class CPanel : public CWindow
{
   public:
      CPanel();
      ~CPanel() {
         Clear();
      };
      void CreatButton();
      void LoadResource();
      void Initialize(CPoint);
      void Clear();
      void UpdateMouse(CPoint _p) {
         mousePos = _p;
      };
      void OnCycle();
      string GetCollisionButtonName();
   private:
      CButtonManager* btnManager = NULL;
      CPoint mousePos;
      int img_x, img_y;
};
#pragma endregion
#pragma endregion
#pragma region - CStatusBoard -
class CStatusBoard
{
   public:
      CStatusBoard();
      void Load();
      void Initialize(CPoint, int, int); // 初始點, 血量, EQ
      void UpdateBar(int, int);
      void SetXY(CPoint);
      void SetDeltaBar(int, int);
      int GetHP() {
         return hp;
      };
      int GetEQ() {
         return eq;
      };
      void OnCycle(int, int);
   protected:
      CMovingBitmap HP_frame, EQ_frame;
      CMovingBitmap HP_bar, EQ_bar;
      CMovingBitmap avatar, avatar_frame;
```

```
    private:
        int dHP, dEQ; //  一次扣的寬度
        int hp, eq;
};
#pragma endregion
#pragma region - CBossBoard -
class CBossManager;
class CBossBoard
{
    friend class CBossManager;
    public:
        CBossBoard();
        void Load();
        void Initialize(CPoint); //  初始點
        void UpdateBar(int);
        void SetXY(CPoint);
        void SetDeltaBar(int);
        void SetShow(bool);
        void SetHP(int _hp);
        int GetHP() {
            return hp;
        };
        bool IsShow() {
            return isShow;
        };
        void OnCycle(CBossManager*);
    protected:
        map<string, CMovingBitmap> bossAvatar;
        CMovingBitmap NULLAvatar;
        CMovingBitmap HP_frame;
        CMovingBitmap HP_bar;
        CMovingBitmap avatar, avatar_frame;
    private:
        CPoint initPos;
        double dHP; //  一次扣的寬度
        int hp;
        bool isShow;
};
#pragma endregion
}
```

12. CLibrary.cpp

```
#pragma once
#include "stdafx.h"
#include "Resource.h"
#include "Refactor.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "CLibrary.h"
#include "CManager.h"
#include "dirent.h"
#include "game.h"
#include "MainFrm.h"
#include <string.h>
#include "mygame.h"
#include <time.h>
#include <fstream>
#include <sstream>

namespace myLibrary
{
    char* ConvertCharPointToString(string ziliaojia, string name, int number)
    {
        char* address;
        std::stringstream ss;
        ss << number;
```

```
    string file = ".\\" + ziliaojia + "\\" + name;

    if (number == -1)
    {
        file += ".bmp";
    }
    else
    {
        file += "_" + ss.str() + ".bmp";
    }

    address = new char[file.length() + 1];
    strcpy(address, file.c_str());
    return address;
}

char* ConvertCharPointToString(string str)
{
    char* address;
    string file = str;
    address = new char[file.length() + 1];
    strcpy(address, file.c_str());
    return address;
}

COLORREF ConvertStringToColor(string _color)
{
    if (_color == "red")
        return RGB(255, 0, 0);
    else if (_color == "green")
        return RGB(0, 255, 0);
    else if (_color == "blue")
        return RGB(0, 0, 255);
    else if (_color == "white")
        return RGB(255, 255, 255);
    else if (_color == "black")
        return RGB(0, 0, 0);
    else if (_color == "txtColor")
        return DIALOG_TEXT_COLOR;

    return RGB(255, 255, 255); //white
}

bool ConvertStringToBoolen(string str)
{
    return str == "true";
}

void getFolderFile(string folderPath, vector<string>* file) //得到資料夾下的所有檔名
{
    DIR* fp; // create folder point
    fp = opendir(folderPath.c_str());
    struct dirent* folderp;
    int k = 0;

    while ((folderp = readdir(fp)) != NULL)
    {
        if (k >= 2)
        {
            string newPath = string(folderp->d_name);
            file->push_back(newPath);
        }

        k++;
    }

    closedir(fp);
}
```

```
int getFolerFileNumber(string folderPath)
{
   DIR* fp; // create folder point
   fp = opendir(folderPath.c_str());
   struct dirent* folderp;
   int k = 0;

   while ((folderp = readdir(fp)) != NULL)
   {
      k++;
   }

   closedir(fp);
   return k - 2;
}

string getFileName(string file)
{
   return file.substr(0, file.length() - 4);
}

vector<string> SplitString(string str)
{
   stringstream ss;
   #pragma region - init stringstream -
   ss.str("");
   ss.clear();
   #pragma endregion
   ss << str;
   vector<string> lineInfo;
   #pragma region - split string -

   while (ss)
   {
      string templine;
      ss >> templine;

      if (templine != "")
      {
         lineInfo.push_back(templine);
      }
   }

   return lineInfo;
}

void DeleteCharPoint(vector<char*> &addresses)
{
   for (vector<char*>::iterator it = addresses.begin(); it != addresses.end(); it++)
   {
      if (NULL != *it)
      {
         delete *it;
         *it = NULL;
      }
   }

   addresses.clear();
}

int GetPostive(int k)
{
   return k > 0 ? k : -k;
}

int GetRandom(int minNumber, int maxNumber)
{
   int random = rand() % (maxNumber - minNumber + 1) + minNumber;
   return random;
```

```
    }

    int ConvertStringToInt(string str)
    {
        stringstream ss;
        int cs;
        ss << str;
        ss >> cs;
        return cs;
    }
}

using namespace myLibrary;
namespace game_framework
{
    #pragma region - Function -
    int ScreenX(int mapx, int rolex)
    {
        return GetPostive(mapx) + rolex;
    }

    bool IsPointInRect(CPoint point, CRect rect)
    {
        return ((rect.left <= point.x) &&
                (point.x <= rect.right) &&
                (rect.top <= point.y) &&
                (point.y <= rect.bottom));
    }

    //a 碰到 b
    bool rectCollision(CRect rect1, CRect rect2)
    {
        CPoint leftTop = CPoint(rect1.left, rect1.top);
        CPoint rightTop = CPoint(rect1.right, rect1.top);
        CPoint leftBottom = CPoint(rect1.left, rect1.bottom);
        CPoint rightBottom = CPoint(rect1.right, rect1.bottom);
        return (IsPointInRect(leftTop, rect2) || IsPointInRect(rightTop, rect2) ||
                IsPointInRect(leftBottom, rect2) || IsPointInRect(rightBottom, rect2));
    }

    //a 碰到 b 或 b 碰到 a
    bool IsRectCollision(CRect rect1, CRect rect2)
    {
        return rectCollision(rect1, rect2) || rectCollision(rect2, rect1);
    }

    bool IsRectInRect(CRect rect1, CRect rect2) //矩形碰撞
    {
        CPoint leftTop = CPoint(rect1.left, rect1.top);
        CPoint rightTop = CPoint(rect1.right, rect1.top);
        CPoint leftBottom = CPoint(rect1.left, rect1.bottom);
        CPoint rightBottom = CPoint(rect1.right, rect1.bottom);
        return (IsPointInRect(leftTop, rect2) && IsPointInRect(rightTop, rect2) &&
                IsPointInRect(leftBottom, rect2) && IsPointInRect(rightBottom, rect2));
    }
    #pragma endregion

    #pragma region - CAnimate -
    CAnimate::CAnimate()
    {
        x = y = bmp_index = bmp_amount = 0;
        SetValid(true);
        double wiatTime = (double)GetRandom(2, 4) / 5.0;
        delayTimer = CTimer(wiatTime);
        bmp.clear();
    }

    CAnimate::~CAnimate()
    {
```

```
    ReleaseAnimate();
}

void CAnimate::OnMove()
{
    //GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded first.");
    if (bmp.size() == 0)
        return;

    if (delayTimer.IsTimeOut())
    {
        delayTimer.ResetTime();

        if (bmp_index < (int)bmp.size() - 1)
        {
            bmp_index++;
        }
        else
        {
            bmp_index = 0;
        }
    }
    else
    {
        delayTimer.CountDown();
    }

    bmp[bmp_index].SetTopLeft(x, y);
}

void CAnimate::OnMove(int dir)
{
    //GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded first.");
    if (bmp.size() == 0)
        return;

    int upperLimit = (bmp_amount - 1) / 4;

    if (dir != 0)          //Move
    {
        if (bmp_index >= (dir - 1) * upperLimit + 1 && bmp_index < upperLimit * dir)
        {
            bmp_index++;
        }
        else
            bmp_index = (dir - 1) * upperLimit + 1;
    }
    else //static
    {
        bmp_index = 0;
    }

    bmp[bmp_index].SetTopLeft(x, y);
}

void CAnimate::LoadBitmap(vector<string> bmps, COLORREF colorkey)
{
    for (unsigned int i = 0; i < bmps.size(); i++)
    {
        char* address = ConvertCharPointToString(bmps[i]);
        AddBitmap(address, colorkey);
        delete address;
    }
}

void CAnimate::LoadBitmap(string ziliaojia, string name, int number, COLORREF transparentColor)
{
    for (int i = 0; i < number; i++)
    {
```

```
        char* address = ConvertCharPointToString(ziliaojia, name, i);
        AddBitmap(address, transparentColor);
        delete address;
    }
}

void CAnimate::LoadBitmap(BitmapPath loadpath)
{
    for (int i = 0; i < loadpath.number; i++)
    {
        char* address = ConvertCharPointToString(loadpath.ziliaojia, loadpath.name, i);
        AddBitmap(address, loadpath.color);
        delete address;
    }
}

CMovingBitmap* CAnimate::AddBitmap(int IDB_BITMAP, COLORREF colorkey)
{
    CMovingBitmap add_bmp;
    add_bmp.LoadBitmap(IDB_BITMAP, colorkey);
    bmp.push_back(add_bmp);
    bmp_amount++;
    return &(bmp[bmp.size() - 1]);
}

CMovingBitmap* CAnimate::AddBitmap(char* filename, COLORREF colorkey)
{
    CMovingBitmap add_bmp;
    add_bmp.LoadBitmap(filename, colorkey);
    bmp.push_back(add_bmp);
    bmp_amount++;
    return &(bmp[bmp.size() - 1]);
}

void CAnimate::Reset()
{
    GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded first.");
}

void CAnimate::SetTopLeft(int nx, int ny)
{
    x = nx, y = ny;
    bmp[bmp_index].SetTopLeft(x, y);
}

void CAnimate::OnShow()
{
    //GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded before they are shown.");
    if (bmp_index < (int)bmp.size())
    {
        bmp[bmp_index].ShowBitmap();
    }
}

int CAnimate::Top()
{
    //GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded first.");
    if (bmp.size() != 0)
    {
        return y;
    }
    else
    {
        return 0;
    }
}

int CAnimate::Left()
{
```

```
    //GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded first.");
    if (bmp.size() != 0)
    {
        return x;
    }
    else
    {
        return 0;
    }
}

int CAnimate::Height()
{
    //GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded first.");
    if (bmp.size() != 0)
    {
        return bmp[bmp_index].Height();
    }
    else
    {
        return 0;
    }
}

int CAnimate::Width()
{
    //GAME_ASSERT(bmp.size() != 0, "CAnimation: Bitmaps must be loaded first.");
    if (bmp.size() != 0)
    {
        return bmp[bmp_index].Width();
    }
    else
    {
        return 0;
    }
}

bool CAnimate::IsNull()
{
    return bmp.size() == 0;
}

void CAnimate::ReleaseAnimate()
{
    bmp.clear();
    vector<CMovingBitmap>().swap(bmp);
}

void CAnimate::SetValid(bool flag)
{
    isValid = flag;
}

bool CAnimate::GetValid()
{
    return isValid;
}

void CAnimate::SetIndex(int _index)
{
    if (_index < (int)bmp.size())
    {
        bmp_index = _index;
    }
    else
    {
        bmp_index = 0;
    }
}
```

```
int CAnimate::GetIndex()
{
   return bmp_index;
}

void CAnimate::CopyAnimateInformation(CAnimate* copyAnimate)
{
   SetValid(copyAnimate->GetValid());
   SetTopLeft(copyAnimate->Left(), copyAnimate->Top());
   SetIndex(copyAnimate->GetIndex());
}

void CAnimate::ResetDelayTime(double _time)
{
   delayTimer.ResetTime(_time);
}

CRect CAnimate::GetRect()
{
   return bmp[bmp_index].GetRect();
}
#pragma endregion

#pragma region - CLayer -
CLayer::CLayer()
{
   layer = 0;
}

CLayer::~CLayer()
{
}

int CLayer::GetLayer()
{
   return layer;
}

void CLayer::SetLayer(int _layer)
{
   if (_layer >= MAX_LAYER_NUMBER)
      _layer = MAX_LAYER_NUMBER - 1;
   else if (_layer < 0)
      _layer = 0;

   layer = _layer;
}
#pragma endregion

#pragma region - timer -

CTimer::CTimer() //default constructor
{
   ResetTime(99.0);
}

CTimer::CTimer(int _time)
{
   ResetTime(_time);
   time = (int)time;
}

CTimer::CTimer(double _time) //給予初始時間
{
   ResetTime(_time);
}

CTimer::~CTimer()
```

```
{
}

void CTimer::CountDown()
{
   time--;
}

int CTimer::GetTime(int k)
{
   return (int)time / (int)reflash;
}

double CTimer::GetTime()
{
   return time / reflash;
}

bool CTimer::IsTimeOut()
{
   return time <= 0.00;
}

void CTimer::ResetTime(double _resetTime)
{
   time = initTime = _resetTime * reflash;
}

void CTimer::ResetTime()
{
   time = initTime;
}

#pragma region -- operator overload --
void CTimer::operator=(CTimer _timer)
{
   time = _timer.time;
   initTime = _timer.time;
}
#pragma endregion

#pragma endregion

#pragma region - CDialog -
CDialog::CDialog()
{
}

CDialog::CDialog(string txtPath, string _txt)
{
   path = txtPath;
   mode = _txt;
   IsTriggered = false;
   LoadTxt(); // load txt
}

CDialog::~CDialog()
{
}

bool CDialog::GetTriggered()
{
   return IsTriggered;
}

void CDialog::SetTriggered()
{
   IsTriggered = !CanReTrigger;
}
```

```
void CDialog::Initialize()
{
   IsTriggered = false;
}

string CDialog::GetAvatar(unsigned int step)
{
   if (step < avatar.size())
      return avatar[step];
   else
      return "RREF";
}

string CDialog::GetDialogTxt(unsigned int step)
{
   if (step < txt.size())
      return txt[step];
   else
      return "RREF";
}

COLORREF CDialog::GetDialogColor(unsigned int step)
{
   if (step < color.size())
      return color[step];

   return ConvertStringToColor("txtColor");
}

int CDialog::GetTxtSize()
{
   return txt.size();
}

string CDialog::GetMode()
{
   return mode;
}

void CDialog::LoadTxt()
{
   fstream dialogTxt; //一個 txt 的文本資料
   dialogTxt.open(path); //open txt
   int index = 0;
   string dialogData; //文本資料的一行
   getline(dialogTxt, dialogData); //先讀取一行 "初始設定"
   CanReTrigger = ConvertStringToBoolen(dialogData); //設定對話可否重複觸發

   while (getline(dialogTxt, dialogData)) //get line
   {
      #pragma region - get Split stirng -
      vector<string> lineInfo = SplitString(dialogData);
      #pragma endregion
      #pragma region - save txt information -
      COLORREF txtColor = ConvertStringToColor(lineInfo.size() >= 3 ? lineInfo[2] : "txtColor"); //get print txt color
      avatar.push_back(lineInfo[0]);
      txt.push_back(lineInfo[1]);
      color.push_back(txtColor);
      #pragma endregion
   }

   dialogTxt.close();
}

#pragma endregion

#pragma region - Camera -
CCamera CCamera::camera;
```

```
CCamera::CCamera()
{
   Initialize();
}

CCamera::~CCamera()
{
}

void CCamera::Initialize()
{
   x = y = 0;
   canMoving = true;
}

void CCamera::Reset()
{
   x = y = 0;
   canMoving = true;
}

void CCamera::SetCameraBoundary(int _left, int _right)
{
   max_left = _left;
   max_right = _right;
}

CCamera* CCamera::Instance()
{
   return &camera;
}

int CCamera::GetX()
{
   return x;
}

int CCamera::GetY()
{
   return y;
}

void CCamera::SetXY(int _x, int _y)
{
   x = _x;
   y = _y;
}

void CCamera::AddX(int _dx)
{
   if (canMoving)
      x += _dx;
}

void CCamera::AddY(int _dy)
{
   if (canMoving)
      y += _dy;
}

void CCamera::SetCanMoving(bool flag)
{
   canMoving = flag;
}
#pragma endregion

#pragma region - Button -
CButton::CButton()
```

```
{
    x = y = 0;
    name = "none";
    state = true;
}

CButton::CButton(const CButton & button)
{
    *this = button;
    name = button.name;
}

CButton::CButton(BitmapPath _loadpath, CPoint initPos, bool initState, bool _needCollision)
{
    loadpath = _loadpath;
    name = _loadpath.name;
    needCollision = _needCollision;
    x = initPos.x;
    y = initPos.y;
    SetState(initState);
}

int CButton::GetX()
{
    return x;
}

int CButton::GetY()
{
    return y;
}

void CButton::SetXY(int _x, int _y)
{
    x = _x;
    y = _y;
    animation.SetTopLeft(x, y);
}

void CButton::SetState(bool flag)
{
    state = flag;
}

bool CButton::GetState()
{
    return state;
}

int CButton::Width()
{
    return animation.Width();
}

int CButton::Height()
{
    return animation.Height();
}

void CButton::LoadBitmap()
{
    if (animation.IsNull())
    {
        if (loadpath.path.empty())
            animation.LoadBitmap(loadpath.ziliaojia, loadpath.name, loadpath.number, loadpath.color);
        else
            animation.LoadBitmap(loadpath.path, loadpath.color);
    }
}
```

```
void CButton::LoadBitmap(BitmapPath _loadpath)
{
   if (loadpath.path.empty())
      animation.LoadBitmap(_loadpath.ziliaojia, _loadpath.name, _loadpath.number, _loadpath.color);
   else
      animation.LoadBitmap(_loadpath.path, _loadpath.color);
}

void CButton::OnMove()
{
   if (state)   //ON
   {
      animation.SetIndex(1);
   }
   else    //OFF
   {
      animation.SetIndex(0);
   }

   animation.SetTopLeft(x, y);
}

void CButton::OnShow()
{
   animation.OnShow();
}

void CButton::Initialize()
{
   if (animation.IsNull())
      LoadBitmap(loadpath);

   SetValid(true);
   SetState(state);
   SetXY(x, y);
   animation.SetTopLeft(x, y);
   CLayerManager::Instance()->AddObject(&animation, INTERFACE_LAYER);
}

void CButton::Initialize(CPoint pos, bool flag)
{
   if (animation.IsNull())
      LoadBitmap(loadpath);

   SetValid(true);
   SetState(flag);
   SetXY(pos.x, pos.y);
   animation.SetTopLeft(x, y);
}

void CButton::SetValid(bool _flag)
{
   animation.SetValid(_flag);
}

bool CButton::GetValid()
{
   return animation.GetValid();
}

CAnimate* CButton::GetAnimate()
{
   return &animation;
}

void CButton::CollisonMouse(CPoint _m)
{
   if (!needCollision)
```

```
        return;

    if (IsCollisionMouse(_m))
    {
        if (!GetState()) //只有第一次進入 Button 有音效
            CAudio::Instance()->Play("btn_collision_3");

        SetState(true);
    }
    else
        SetState(false);
}

void CButton::ClickButton()
{
    SetState(!GetState());
    CAudio::Instance()->Play("btn_click_3");
}

bool CButton::IsCollisionMouse(CPoint _m)
{
    return IsPointInRect(_m, animation.GetRect());
}

void CButton::operator=(const CButton& button)
{
    loadpath = button.loadpath;
    name = button.loadpath.name;
    needCollision = button.needCollision;
    Initialize(CPoint(button.x, button.y), button.state);
}
#pragma endregion

#pragma region - CInteger -
CInteger::CInteger(int digits)
    : NUMDIGITS(digits)
{
    isBmpLoaded = false;
}

void CInteger::Initialize(CPoint init_pos, int init_num, int init_digitNum)
{
    SetInteger(init_num);
    SetTopLeft(init_pos.x, init_pos.y);
    NUMDIGITS = init_digitNum;
    SetValid(true);
    layer.SetLayer(INTERFACE_LAYER);
}

void CInteger::Add(int x)
{
    n += x;
}

int CInteger::GetInteger()
{
    return n;
}

void CInteger::LoadBitmap(string ziliaojia, string name)
{
    if (!isBmpLoaded)
    {
        for (int i = 0; i < 11; i++)
        {
            char* address = ConvertCharPointToString(ziliaojia, name, i);
            digit[i].LoadBitmap(address, RGB(255, 255, 255));
            delete address;
        }
```

```
        isBmpLoaded = true;
    }
}

void CInteger::SetInteger(int i)
{
    n = i;
}

void CInteger::SetTopLeft(int nx, int ny) // 將動畫的左上角座標移至 (x,y)
{
    x = nx;
    y = ny;
}

void CInteger::ShowBitmap()
{
    GAME_ASSERT(isBmpLoaded, "CInteger: 請先執行 LoadBitmap，然後才能 ShowBitmap");
    int nx;   // 待顯示位數的 x 座標
    int MSB; // 最左邊(含符號)的位數的數值

    if (n >= 0) {
        MSB = n;
        nx = x + digit[0].Width() * (NUMDIGITS - 1);
    }
    else {
        MSB = -n;
        nx = x + digit[0].Width() * NUMDIGITS;
    }

    for (int i = 0; i < NUMDIGITS; i++) {
        int d = MSB % 10;
        MSB /= 10;
        digit[d].SetTopLeft(nx, y);
        digit[d].ShowBitmap();
        nx -= digit[d].Width();
    }

    if (n < 0) { // 如果小於 0，則顯示負號
        digit[10].SetTopLeft(nx, y);
        digit[10].ShowBitmap();
    }
}
#pragma endregion

#pragma region - CAction -
CAction::CAction()
{
    x = y = 0;
    SetValid(true);
    delayAdapter["idle"] = delay_idle;
    delayAdapter["run"] = delay_run;
    delayAdapter["jump"] = delay_jump;
    nowAction = NULL;
    nowBitmap = NULL;
    double waitTime = delay_idle;
    delayTimer = CTimer(waitTime);
}

void CAction::OnMove(string _nowAction)
{
    delayTimer.CountDown();

    if (action != _nowAction)          //切換動作
    {
        delayTimer.ResetTime(delayAdapter[_nowAction]);
        SetAction(_nowAction);
    }
```

```
    if (!delayTimer.IsTimeOut())     //延遲
    {
        return;
    }
    else
    {
        delayTimer.ResetTime();
    }

    //現在動作 + 方向，從 map 取出對應 Action Vector
    nowAction = &(paser[action + faceTo]);

    if (action_index < (int)(nowAction->size() - 1))
    {
        action_index++;
    }
    else
    {
        action_index = 0;
    }

    nowBitmap = &((paser[_nowAction + faceTo])[action_index]);
    nowBitmap->SetTopLeft(x, y);
}

void CAction::OnShow()
{
    nowBitmap->ShowBitmap();
}

void CAction::SetAction(string _action)
{
    action = _action;
    action_index = 0;
    action = _action;
    nowAction = &(paser[action + faceTo]);
    nowBitmap = &((paser[_action + faceTo])[action_index]);
    nowBitmap->SetTopLeft(x, y);
}

string CAction::GetAction()
{
    return action;
}

void CAction::Initialize()
{
    nowAction = NULL;
    nowBitmap = NULL;
    action = "idle";
    faceTo = "_R";
    action_index = 0;

    if (!paser.empty())
    {
        nowAction = &(paser[action + faceTo]);
        nowBitmap = &((paser[action + faceTo])[0]);
    }
}

//ex : LoadAction("idle", BitmapPath("..\\RES\\miku\\idle", "idle", 19));
void CAction::LoadAction(string _action, BitmapPath loadpath)
{
    #pragma region Load action
    vector<CMovingBitmap> vector_R(loadpath.number);
    vector<CMovingBitmap> vector_L(loadpath.number);

    for (int i = 0; i < loadpath.number; i++)
```

```
    {
        char* address_R = ConvertCharPointToString(loadpath.ziliaojia + "\\R", loadpath.name, i);
        char* address_L = ConvertCharPointToString(loadpath.ziliaojia + "\\L", loadpath.name + "_L", i);
        vector_R[i].LoadBitmap(address_R, loadpath.color);
        vector_L[i].LoadBitmap(address_L, loadpath.color);
        delete address_R;
        delete address_L;
    }

    paser[_action + "_R"] = vector_R;
    paser[_action + "_L"] = vector_L;
    #pragma endregion
}

int CAction::Height()
{
    if (nowBitmap == NULL)
    {
        return 0;
    }

    return nowBitmap->Height();
}

int CAction::Width()
{
    if (nowBitmap == NULL)
    {
        return 0;
    }

    return nowBitmap->Width();
}

int CAction::Left()
{
    if (nowBitmap == NULL)
    {
        return 0;
    }

    return x;
}

int CAction::Top()
{
    if (nowBitmap == NULL)
    {
        return 0;
    }

    return y;
}

void CAction::SetTopLeft(int _x, int _y)
{
    x = _x;
    y = _y;
    nowBitmap->SetTopLeft(x, y);
}

bool CAction::IsNull()
{
    return (nowAction == NULL || nowBitmap == NULL || nowAction->size() <= 0);
}

void CAction::SetValid(bool _valid)
{
    isValid = _valid;
```

```
}

bool CAction::GetValid()
{
    return isValid;
}

void CAction::SetIndex(int _index)
{
    action_index = _index;
}

int CAction::GetIndex()
{
    return action_index;
}

void CAction::SetFaceTo(string _faceTo)
{
    faceTo = _faceTo;
}

string CAction::GetFaceTo()
{
    return faceTo;
}

CMovingBitmap* CAction::GetNowBitmap()
{
    return nowBitmap;
}

CRect CAction::GetRect()
{
    return nowBitmap->GetRect();
}

int CAction::GetActionHeight(string _action)
{
    if (paser.count(_action) && !paser[_action].empty())
    {
        return paser[_action].begin()->Height();
    }

    return 108;
}
#pragma endregion

#pragma region - CEnd -
CEnd::CEnd()
{
    isGet = false;
    endName = "";
}

CEnd::CEnd(string _endName)
{
    endName = _endName;
    isGet = false;
    LoadEnd();
}

CEnd::~CEnd()
{
}

string CEnd::GetBmpPath(int index)
{
    if (index < 0 || index >= (int)bmpPath.size())
```

```
    {
        return END_EOF;
    }
    else
    {
        return bmpPath[index];
    }
}

string CEnd::GetTxt(int index)
{
    if (index < 0 || index >= (int)txt.size())
    {
        return END_EOF;
    }
    else
    {
        return txt[index];
    }
}

void CEnd::LoadEnd()
{
    LoadBmpTxt(endName);
}

void CEnd::LoadBmpTxt(string _endName)
{
    string loadbmpFolderPath = "RES\\End\\" + _endName + "\\bmp\\";
    string loadtxtFolderPath = "RES\\End\\" + _endName + "\\txt\\";
    getFolderFile(loadbmpFolderPath, &bmpPath);
    getFolderFile(loadtxtFolderPath, &txt);

    for (unsigned int i = 0; i < bmpPath.size(); i++)
    {
        bmpPath[i] = loadbmpFolderPath + bmpPath[i];
    }

    for (unsigned int i = 0; i < txt.size(); i++)
    {
        CDialogManager::Instance()->LoadDialog(txt[i], loadtxtFolderPath + txt[i]);
    }
}
#pragma endregion

#pragma region - CToumeiImage -
CToumeiImage::CToumeiImage()
{
    SetAlpha(0);
    dFadeInValue = 3;
    dFadeOutValue = -4;
}

CToumeiImage::CToumeiImage(int _dFadeInValue, int _dFadeOutValue)
{
    SetAlpha(0);
    SetFadeInOut(_dFadeInValue, _dFadeOutValue);
}

CToumeiImage::CToumeiImage(string _loadbmpPath, int _dFadeInValue, int _dFadeOutValue)
{
    CToumeiImage(_dFadeInValue, _dFadeOutValue);
    bmpLoadPath = _loadbmpPath;
    SetBmp(_loadbmpPath);
}

CToumeiImage::~CToumeiImage()
{
}
```

```
void CToumeiImage::SetAlpha(int _a)
{
   alpha = _a;

   if (alpha <= 0)
      alpha = 0;
   else if (alpha >= 255)
      alpha = 255;
}

void CToumeiImage::SetBmp(string _path)
{
   bmp.DeleteObject();
   bmp.m_hObject = LoadImage(NULL, _path.c_str(), IMAGE_BITMAP, 0, 0, LR_LOADFROMFILE);
}

void CToumeiImage::SetFadeInOut(int dFIV, int dFOV)
{
   dFadeInValue = dFIV;
   dFadeOutValue = dFOV;
}

void CToumeiImage::FadeIn()
{
   SetAlpha(alpha + dFadeInValue);
}

void CToumeiImage::FadeOut()
{
   SetAlpha(alpha + dFadeOutValue);
}

void CToumeiImage::DrawImage()
{
   DrawBitmap(&bmp, alpha);
}
#pragma endregion

#pragma region - Windows -

#pragma region - CWindow -
CWindow::CWindow()
{
   x = y = 0;
   isLoaded = false;
   isOpen = false;
   closeButton = new CButton(BitmapPath("RES\\Button", "close", 2, RGB(214, 214, 214)), CPoint(0, 0), false, true);
}

CWindow::CWindow(CPoint _p)
{
   SetXY(_p);
   isLoaded = false;
   isOpen = false;
}

CWindow::~CWindow()
{
   Clear();
}

void CWindow::LoadResource()
{
   closeButton->LoadBitmap();
   background.LoadBitmap("RES\\Windows", "EndWindow_2", RGB(214, 214, 214));
}

void CWindow::Initialize(CPoint _initP)
```

```
{
   SetXY(_initP);
   const int CLSBTN_INIT_X = x + background.Width() - closeButton->Width() - 10;
   const int CLSBTN_INIT_Y = y + 10;
   closeButton->Initialize(CPoint(CLSBTN_INIT_X, CLSBTN_INIT_Y), false);
   background.SetTopLeft(x, y);
   background.SetValid(false);
   closeButton->SetValid(false);
   CLayerManager::Instance()->AddObject(&background, INTERFACE_LAYER);
   CLayerManager::Instance()->AddObject(closeButton->GetAnimate(), INTERFACE_LAYER);
   isOpen = false;
}

void CWindow::Clear()
{
   if (closeButton != NULL)
   {
      delete closeButton;
      closeButton = NULL;
   }
}

void CWindow::Open()
{
   if (!isOpen)
   {
      isOpen = true;
      CAudio::Instance()->Play("open_window");
   }
}

void CWindow::Close()
{
   if (isOpen)
   {
      isOpen = false;
      CAudio::Instance()->Play("page_close");
   }
}

bool CWindow::IsCollisionClose(CPoint _m)
{
   return closeButton->IsCollisionMouse(_m) && isOpen;
}

bool CWindow::IsOpen()
{
   return isOpen;
}

void CWindow::SetCloseButton(CPoint _p)
{
   closeButton->SetXY(_p.x, _p.y);
}

void CWindow::SetXY(CPoint _p)
{
   x = _p.x;
   y = _p.y;
}

void CWindow::CollisionClose(CPoint _p)
{
   closeButton->CollisonMouse(_p);
}

void CWindow::OnCycle()
{
   background.SetValid(isOpen);
```

```
    closeButton->SetValid(isOpen);

    if (!IsOpen())
    {
        return;
    }

    closeButton->OnMove();
    background.SetTopLeft(x, y);
}

void CWindow::OnShow()
{
    if (!IsOpen())
    {
        return;
    }

    background.ShowBitmap();
    closeButton->OnShow();
}
#pragma endregion

#pragma region - Scroll windows -

CScrollWindow::CScrollWindow(): CWindow()
{
    rowNum = 2;
    colNum = 4;
    #pragma region - Create endName -
    fstream haveEnd;
    haveEnd.open("RES\\End\\HaveEnd.txt", ios::in);
    string lineData;

    while (getline(haveEnd, lineData))
    {
        vector<string> lineInfo = SplitString(lineData);
        endName.push_back(lineInfo[0]);
    }

    haveEnd.close();
    #pragma endregion
}

CScrollWindow::~CScrollWindow()
{
}

void CScrollWindow::OnScrolling(short _s)
{
    int move = -(int)_s;
    move /= 12;
    const int NOW_BOTTOM = (endingVector[rowNum - 1][colNum - 1]).GetAnimate()->GetRect().bottom;
    const int LAST_TOP = (endingVector[rowNum - 1][colNum - 1]).GetAnimate()->Top();

    if (LAST_TOP - move < cover.GetRect().bottom || NOW_BOTTOM - move > limit_buttom)
    {
        return;
    }

    for (int r = 0; r < rowNum; r++)
    {
        for (int c = 0; c < colNum; c++)
        {
            int orgin_x = endingVector[r][c].GetAnimate()->Left();
            int orgin_y = endingVector[r][c].GetAnimate()->Top();
            endingVector[r][c].SetXY(orgin_x, orgin_y - move);
        }
    }
```

```
}

string CScrollWindow::GetCollisionButtonName(CPoint mPoint)
{
   for (unsigned r = 0; r < endingVector.size(); r++)
   {
      for (unsigned c = 0; c < endingVector[r].size(); c++)
      {
         if (IsPointInRect(mPoint, endingVector[r][c].GetAnimate()->GetRect())) // 有碰撞
         {
            if (!endingVector[r][c].GetState())                    // 未解鎖
            {
               CAudio::Instance()->Play("not_clear");
               return "NoButtonClick";
            }

            return endingVector[r][c].GetName();   // 有碰撞 + 已解鎖
         }
      }
   }

   return "NoButtonClick";
}

void CScrollWindow::LoadResource()
{
   CWindow::LoadResource();
   cover.LoadBitmap("RES\\Windows", "cover_top", RGB(214, 214, 214));
   cover_bottom.LoadBitmap("RES\\Windows", "cover_bottom", RGB(214, 214, 214));

   for (int r = 0; r < rowNum; r++)
   {
      vector<CButton> columnBitmaps(colNum);

      for (int c = 0; c < colNum; c++)
      {
         int endID = r + c;

         if (c != 0)
            endID++;

         #pragma region - Create loadString -
         vector<string> loadString;
         loadString.push_back("RES\\End\\EndImg\\end_lock.bmp");
         loadString.push_back("RES\\End\\EndImg\\end_" + to_string(endID) + "_0.bmp");
         #pragma endregion
         columnBitmaps[c] = CButton(BitmapPath(loadString, RGB(214, 214, 214)), CPoint(x, y), false, true);
         columnBitmaps[c].SetName(endID < (int)endName.size() ? endName[endID] : "lock");
         columnBitmaps[c].SetValid(false);
      }

      endingVector.push_back(columnBitmaps);
      columnBitmaps.clear();
   }

   img_height = (endingVector[0][0]).Height();
   img_width =    (endingVector[0][0]).Width();
}

void CScrollWindow::Initialize(CPoint _p)
{
   CWindow::Initialize(_p);
   cover.SetTopLeft(_p.x, _p.y);
   cover_bottom.SetTopLeft(_p.x, SIZE_Y - cover_bottom.Height());
   // set image initial position
   img_x = x + 150;
   img_y = cover.GetRect().bottom + 10;
   // set padding
   const int PADDING_X = 10;
```

```
    const int PADDING_Y = 10;
    //set limit of scrolling
    limit_top = img_y;
    limit_buttom = img_y + colNum * (img_height + PADDING_Y);

    for (int r = 0; r < rowNum; r++)
    {
      for (int c = 0; c < colNum; c++)
      {
        endingVector[r][c].SetXY(img_x + r * (img_width + PADDING_X), img_y + c * (img_height +
PADDING_Y));
        #pragma region - set get end -
        bool flag = CEndManager::Instance()->IsPassEnd(endingVector[r][c].GetName());
        endingVector[r][c].SetState(CEndManager::Instance()->IsPassEnd(endingVector[r][c].GetName()));
        #pragma endregion
      }
    }
  }

  void CScrollWindow::Close()
  {
    CWindow::Close();
    isOpen = false;
    Initialize(CPoint(x, y));
  }

  void CScrollWindow::OnCycle()
  {
    CWindow::OnCycle();

    if (!IsOpen())
    {
      return;
    }

    cover.SetTopLeft(x, y);

    for (int r = 0; r < rowNum; r++)
    {
      for (int c = 0; c < colNum; c++)
      {
        endingVector[r][c].OnMove();
        //endingVector[r][c].CollisonMouse(mousePos);
      }
    }
  }

  void CScrollWindow::OnShow()
  {
    if (!IsOpen())
    {
      return;
    }

    background.ShowBitmap();

    for (int r = 0; r < rowNum; r++)
    {
      for (int c = 0; c < colNum; c++)
      {
        if (endingVector[r][c].GetAnimate()->GetRect().bottom <= background.GetRect().bottom &&
endingVector[r][c].GetAnimate()->Top() >= y)
          endingVector[r][c].OnShow();
      }
    }

    cover.ShowBitmap();
    cover_bottom.ShowBitmap();
    closeButton->OnShow();
```

```
}
#pragma endregion

#pragma region - CSwitchWindow -
CSwitchWindow::CSwitchWindow() : CWindow()
{
   bmp.clear();
   index = 0;
   step = 0;
   arrow_left.SetValid(false);
   arrow_right.SetValid(false);
}

CSwitchWindow::~CSwitchWindow()
{
}

void CSwitchWindow::LoadResource(string folder)
{
   closeButton->LoadBitmap();
   vector<string> bmpPath;
   int folderNumber = getFolerFileNumber(folder);

   for (int k = 0; k < folderNumber; k++)
   {
      vector<CMovingBitmap> tempVectorBmp;
      string child_folder = folder + std::to_string(k) + "\\";
      bmpPath.clear();
      getFolderFile(child_folder, &bmpPath);

      for (unsigned int i = 0; i < bmpPath.size(); i++)
      {
         #pragma region - add bmp -
         CMovingBitmap tempBmp;
         string path = child_folder + bmpPath[i];
         char* address = ConvertCharPointToString(child_folder + bmpPath[i]);
         tempBmp.LoadBitmap(address);
         delete address;
         tempBmp.SetValid(false);
         tempBmp.SetTopLeft(0, 0);
         #pragma endregion
         tempVectorBmp.push_back(tempBmp);
      }

      bmp.push_back(tempVectorBmp);
   }

   background = bmp[0][0];
   #pragma region - set arrow -
   arrow_left.LoadBitmap("RES\\Handbook\\arrow_left.bmp", RGB(214, 214, 214));
   arrow_right.LoadBitmap("RES\\Handbook\\arrow_right.bmp", RGB(214, 214, 214));
   #pragma endregion
}

void CSwitchWindow::Initialize(CPoint _point)
{
   CWindow::Initialize(_point);
   arrow_left.SetTopLeft(0, 180);
   arrow_right.SetTopLeft(560, 180);
   arrow_left.SetValid(false);
   arrow_right.SetValid(false);
   CLayerManager::Instance()->AddObject(&arrow_left, INTERFACE_LAYER + 1);
   CLayerManager::Instance()->AddObject(&arrow_right, INTERFACE_LAYER + 1);
}

void CSwitchWindow::Open()
{
   CWindow::Open();
```

```
    if (index < (int)bmp.size())
    {
        background = bmp[index][step];
        arrow_left.SetValid(false);
        arrow_right.SetValid(true);
    }
}

void CSwitchWindow::Close()
{
    CWindow::Close();
    bmp[index][step].SetValid(false);
    index = 0;
    step = 0;
    arrow_left.SetValid(false);
    arrow_right.SetValid(false);
}

void CSwitchWindow::Switchwindow(string dir)
{
    bmp[index][step].SetValid(false);

    if (dir == "left")
    {
        SetIndex(index - 1);
        step = 0;
    }
    else if (dir == "right")
    {
        SetIndex(index + 1);
        step = 0;
    }

    bmp[index][step].SetValid(true);
    background = bmp[index][step];
}

void CSwitchWindow::SetIndex(int _index)
{
    if (_index <= 0)
    {
        index = 0;
    }
    else if (_index >= (int)bmp.size())
    {
        index = (int)bmp.size() - 1;
    }
    else
    {
        index = _index;
    }
}

bool CSwitchWindow::CollisionArrow(CPoint mPoint)
{
    if (arrow_left.GetValid() && IsPointInRect(mPoint, arrow_left.GetRect()))
    {
        Switchwindow("left");
        return true;
    }
    else if (arrow_right.GetValid() && IsPointInRect(mPoint, arrow_right.GetRect()))
    {
        Switchwindow("right");
        return true;
    }

    return false;
}
```

```
void CSwitchWindow::ClickWindows(CPoint point, string mouseKey)
{
    if (!CollisionArrow(point))
    {
        if (mouseKey == "left" && step < (int)bmp[index].size() - 1)
        {
            step++;
        }
        else if (mouseKey == "right" && step > 0)
        {
            step--;
        }

        Switchwindow("this");
    }
}

void CSwitchWindow::OnCycle()
{
    CWindow::OnCycle();

    if (!IsOpen())
    {
        return;
    }

    arrow_left.SetValid(IsOpen() && index != 0);
    arrow_right.SetValid(IsOpen() && index < (int)bmp.size() - 1);
}
#pragma endregion

#pragma region - CPanel -
CPanel::CPanel()
{
    btnManager = new CButtonManager();
}

void CPanel::CreatButton()
{
    const int BTN_X = 240;
    const int F_BTN_Y = 120;
    btnManager->CreateButton(BitmapPath("RES\\Button", "resume", 2, RGB(214, 214, 214)), CPoint(BTN_X,
F_BTN_Y), false, true);
    btnManager->CreateButton(BitmapPath("RES\\Button", "restart", 2, RGB(214, 214, 214)), CPoint(BTN_X,
F_BTN_Y + 60), false, true);
    btnManager->CreateButton(BitmapPath("RES\\Button", "menu", 2, RGB(214, 214, 214)), CPoint(BTN_X,
F_BTN_Y + 120), false, true);
    btnManager->CreateButton(BitmapPath("RES\\Button", "exit", 2, RGB(214, 214, 214)), CPoint(BTN_X, F_BTN_Y
+ 180), false, true);
}

void CPanel::LoadResource()
{
    CWindow::LoadResource();
    btnManager->Load();
    CMovingBitmap panelBK;
    panelBK.LoadBitmap("RES\\UI\\panel", "panel", RGB(214, 214, 214));
    background = panelBK;
}

void CPanel::Initialize(CPoint _p)
{
    CWindow::Initialize(_p);
    btnManager->Initialize();
    btnManager->SetValid(false);
    const int BTN_X = _p.x + background.Width() - 50;
    const int BTN_Y = _p.y - 30;
    closeButton->SetXY(BTN_X, BTN_Y);
}
```

```
void CPanel::Clear()
{
   CWindow::Clear();

   if (btnManager != NULL)
   {
      btnManager->Clear();
      delete btnManager;
      btnManager = NULL;
   }
}

void CPanel::OnCycle()
{
   CWindow::OnCycle();
   btnManager->SetValid(IsOpen());
   btnManager->UpdateState(mousePos);
   CollisionClose(mousePos);
   btnManager->OnCycle();
}

string CPanel::GetCollisionButtonName()
{
   return btnManager->GetCollisionButtonName();
}
#pragma endregion

#pragma endregion

#pragma region - Status Board -

#pragma region - CStatusBoard -
CStatusBoard::CStatusBoard()
{
}

void CStatusBoard::Load()
{
   HP_bar.LoadBitmap("RES\\UI\\status\\blood.bmp", RGB(214, 214, 214));
   HP_frame.LoadBitmap("RES\\UI\\status\\bar_frame.bmp", RGB(214, 214, 214));
   EQ_bar.LoadBitmap("RES\\UI\\status\\EQ.bmp", RGB(214, 214, 214));
   EQ_frame.LoadBitmap("RES\\UI\\status\\bar_frame.bmp", RGB(214, 214, 214));
   avatar.LoadBitmap("RES\\UI\\status\\avatar.bmp");
   avatar_frame.LoadBitmap("RES\\UI\\status\\avatar_frame.bmp", RGB(214, 214, 214));
}

void CStatusBoard::Initialize(CPoint _p, int _HP, int _EQ)
{
   SetXY(_p);
   SetDeltaBar(_HP, _EQ);
   hp = _HP;
   eq = _EQ;
   CLayerManager::Instance()->AddObject(&HP_bar, INTERFACE_LAYER - 1);
   CLayerManager::Instance()->AddObject(&EQ_bar, INTERFACE_LAYER - 1);
   CLayerManager::Instance()->AddObject(&avatar, INTERFACE_LAYER - 1);
   CLayerManager::Instance()->AddObject(&HP_frame, INTERFACE_LAYER);
   CLayerManager::Instance()->AddObject(&EQ_frame, INTERFACE_LAYER);
   CLayerManager::Instance()->AddObject(&avatar_frame, INTERFACE_LAYER);
}

void CStatusBoard::UpdateBar(int _HP, int _EQ)
{
   if (_HP <= 0)
   {
      HP_bar.SetTopLeft(-HP_bar.Width(), HP_bar.Top());
   }
   else
   {
```

```
        int subHP = hp - _HP;
        hp = _HP;
        HP_bar.SetTopLeft(HP_bar.Left() - dHP * subHP, HP_bar.Top());
    }

    if (_EQ <= 0)
        EQ_bar.SetTopLeft(-EQ_bar.Width(), EQ_bar.Top());

    {
        int subEQ = eq - _EQ;
        eq = _EQ;
        EQ_bar.SetTopLeft(EQ_bar.Left() - dEQ * subEQ, EQ_bar.Top());
    }
}

void CStatusBoard::SetXY(CPoint _p)
{
    avatar.SetTopLeft(_p.x, _p.y);
    avatar_frame.SetTopLeft(_p.x, _p.y);
    const int HP_X = _p.x + avatar_frame.Width();
    const int HP_Y = _p.y;
    const int PADDING_Y = 15;
    HP_frame.SetTopLeft(HP_X, HP_Y + PADDING_Y);
    HP_bar.SetTopLeft(HP_X, HP_Y + PADDING_Y);
    const int EQ_X = _p.x + avatar_frame.Width();
    const int EQ_Y = HP_bar.GetRect().bottom;
    EQ_frame.SetTopLeft(HP_X, EQ_Y);
    EQ_bar.SetTopLeft(HP_X, EQ_Y);
}

void CStatusBoard::SetDeltaBar(int _blood, int _EQ)
{
    dHP = HP_bar.Width() / _blood;
    dEQ = EQ_bar.Width() / _EQ;
}

void CStatusBoard::OnCycle(int _HP, int _EQ)
{
    UpdateBar(_HP, _EQ);
}
#pragma endregion

#pragma region - BossBoard -
CBossBoard::CBossBoard()
{
}

void CBossBoard::Load()
{
    HP_bar.LoadBitmap("RES\\UI\\status\\blood.bmp", RGB(214, 214, 214));
    HP_frame.LoadBitmap("RES\\UI\\status\\bar_frame.bmp", RGB(214, 214, 214));
    avatar.LoadBitmap("RES\\Boss\\Avatar\\Xingting.bmp");       // 要擋住血條
    NULLAvatar.LoadBitmap("RES\\Boss\\Avatar\\NULL.bmp");
    avatar_frame.LoadBitmap("RES\\UI\\status\\avatar_frame.bmp", RGB(214, 214, 214));
    #pragma region - Load boss avatar -
    vector<string> avatarName;
    string avatarFolderPath = "RES\\Boss\\Avatar\\";
    getFolderFile(avatarFolderPath, &avatarName);

    for (unsigned i = 0; i < avatarName.size(); i++)
    {
        int strLength = avatarName[i].length();
        string ext = avatarName[i].substr(strLength - 4, strLength);

        if (ext == ".bmp")
        {
            string sAvatar = getFileName(avatarName[i]);
            bossAvatar[sAvatar] = CMovingBitmap();
            char* address = ConvertCharPointToString((avatarFolderPath + avatarName[i]));
```

```
            bossAvatar[sAvatar].LoadBitmap(address);
            delete address;
        }
    }

    #pragma endregion
}

void CBossBoard::Initialize(CPoint _p)
{
    SetXY(_p);
    initPos = _p;
    CLayerManager::Instance()->AddObject(&HP_bar, INTERFACE_LAYER - 1);
    CLayerManager::Instance()->AddObject(&HP_frame, INTERFACE_LAYER);
    CLayerManager::Instance()->AddObject(&avatar_frame, INTERFACE_LAYER);
    CLayerManager::Instance()->AddObject(&avatar, INTERFACE_LAYER - 1);
    SetShow(false);
}

void CBossBoard::UpdateBar(int _HP)
{
    if (_HP <= 0)      //強制補正
    {
        HP_bar.SetTopLeft(avatar_frame.Left(), HP_bar.Top());
    }

    int subHP = hp - _HP;
    hp = _HP;
    HP_bar.SetTopLeft(HP_bar.Left() + (int)(dHP * subHP), HP_bar.Top());
}

void CBossBoard::SetXY(CPoint _p)
{
    const int AVATAR_X = SIZE_X - avatar_frame.Width();
    const int AVATAR_Y = _p.y;
    avatar.SetTopLeft(AVATAR_X, AVATAR_Y);
    avatar_frame.SetTopLeft(AVATAR_X, AVATAR_Y);
    const int HP_X = AVATAR_X - HP_frame.Width();
    const int HP_Y = avatar_frame.GetRect().bottom - HP_frame.Height();
    const int PADDING_Y = 15;
    HP_frame.SetTopLeft(HP_X, HP_Y);
    HP_bar.SetTopLeft(HP_X, HP_Y);
}

void CBossBoard::SetDeltaBar(int _HP)
{
    dHP = (double)(HP_bar.Width() / _HP);
}

void CBossBoard::SetShow(bool _isValid)
{
    isShow = _isValid;
    HP_bar.SetValid(_isValid);
    HP_frame.SetValid(_isValid);
    avatar.SetValid(_isValid);
    avatar_frame.SetValid(_isValid);
}

void CBossBoard::SetHP(int _hp)
{
    UpdateBar(_hp);
}

void CBossBoard::OnCycle(CBossManager* bManager)
{
    if (bManager->targetBoss != NULL && bManager->IsBattle())
    {
        if (!IsShow()) // 只需換一次
        {
```

```
            string bossID = (bManager->targetBoss)->GetID();

            if (bossAvatar.count(bossID) && !bossAvatar[bossID].IsNull())
            {
               avatar = bossAvatar[bossID];
               hp = (bManager->targetBoss)->GetHp();
               SetDeltaBar(hp);
            }
            else
            {
               avatar = NULLAvatar;
            }

            SetXY(initPos);
            SetShow(true);
         }

         SetHP(bManager->targetBoss->GetHp());
      }
      else
      {
         SetShow(false);
      }
   }
   #pragma endregion

   #pragma endregion
}
```

13. CManager.h

```
#pragma once
#include "CBlockMap.h"
#include "Refactor.h"
#include "CEraser.h"
#include "CBoss.h"
#include "CLibrary.h"

namespace game_framework
{
   #pragma region - monsterManager -
   class CMonsterManager
   {
         friend class CMapManager;
      public:
         CMonsterManager();
         ~CMonsterManager();
         void Clear();
         CMonster* GetMonsterType(int);
         CMonster* AddMonster(vector<int>, int );
         void CreateMonster(int, vector<int>, int, vector<CBlock>*);
         void DeleteMonster(vector<CMonster*>::iterator);
         vector<CMonster*> GetMonster();
         CLayer layer;

      private:
         vector<CMonster*> monsterManager;
   };
   #pragma endregion

   #pragma region - MapManager -
   class CMapManager
   {
      public:
         CMapManager();
         ~CMapManager();
         int GetNowMap();
         int GetUpMap();
         int GetDownMap();
```

```
        int GetLeftMap();
        int GetRightMap();
        int GetLoadMap();
        int GetSplitLeft();
        int GetSplitRight();
        int GetX1();
        int GetX2();
        int GetBitmapWidth();
        int GetNpcNumber();
        int GetNpcLayer(int);
        bool GetNpcValid(int);

        int GetBlockMapSize() {
           return (int)blockMap.size();
        };
        vector<CBlock>* GetBlockVector();
        vector<CDoor>* GetDoorVector();

        void SetMovingLeft(bool);
        void SetMovingRight(bool);
        void OnMove();
        void SetXY(int, int);

        bool IsMapEnd();

        void ChangeMap(int, string);
        void LoadMapBitmap();
        void OnShow();
        CMovingBitmap* GetBitmap();
        CAnimate* GetNpc(int);
        CLayer layer;

        vector<CMonster*>* GetPasserby();
        void DeletePasserby(vector<CMonster*>::iterator); //temp
        void AddPasserby();
        void Initialize();
        CMonsterManager passerbyManager; //temp

        void ReloadBlockMap(); //如果有更新 map，線上更新
    private:
        const int directionX = MOVE_DISTANCE;
        int nowMap;
        int loadMap;
        string loadMapPath;
        vector<CBlockMap> blockMap;
        CMovingBitmap background;
        int x, init_x;
        bool isMovingLeft;
        bool isMovingRight;
        int npcNumber;

        void InitializeCBlockMap();

        int max_map_number;
};
#pragma endregion

#pragma region - layerManager -
#pragma region - layerManager - How to use -
//How to use layerManager
//Step1 : include "CManager.h"
//Step2 : confirm your animate or cmovingbitmap is correctly Load
//Step3 : write "CLayerManager::Instance()->AddObject( (animate/cmovingbitmap), layer )"
//Maybe the statement can write anywhere if Step 2 is true!
//by the way, Notice your Step3 can't Add same animate or cmovingbitmap repeatly
//                    Notice If your animate or comvingbitmap is move, you must reset there point(use SetTopLeft(x, y))
#pragma endregion
class CLayerManager
{
```

```
    public:
        CLayerManager();
        ~CLayerManager();
        void Clear();
        void AddObject(CMovingBitmap*, int);
        void AddObject(CAnimate*, int);
        void AddObject(CAction*, int);
        void AddObject(CInteger*, int);

        void ShowLayer();
        static CLayerManager* Instance();
        void Initialize();
    private:
        vector <CMovingBitmap*> layerBitmap[MAX_LAYER_NUMBER];
        vector <CAnimate*> layerAnimate[MAX_LAYER_NUMBER];
        vector <CAction*> layerAction[MAX_LAYER_NUMBER];
        vector <CInteger*> layerInteger[MAX_LAYER_NUMBER];
        static CLayerManager layerManager;
};
#pragma endregion

#pragma region - DialogManager -
class CDialogManager
{
    public:
        CDialogManager();
        ~CDialogManager();
        void OnCycle(); //要重複執行的事情
        void Initialize();
        static CDialogManager* Instance();
        void Start(string);
        void Next();
        void Stop();
        void ShowText();
        bool GetDialogState();
        void LoadDialog(string, string);

        void OpenMusicPlayer(string flag) {
            musicFromNPC = flag;
        };
    private:
        void Dialog();
        map<string, vector<string> > txt;
        map<string, CDialog> dialogmap;
        map<string, CMovingBitmap> dialogAvatar;

        CMovingBitmap dialog_background;
        CMovingBitmap avatar_null; //隨便 load 一張圖，給予他 valid = false 屬性，對話結束的時候將 avatar 指向這
像伙，使 LayerManager 不顯示 avatar
        CMovingBitmap avatar;
        CAnimate textNext;

        CLayer backgroundLayer;
        CLayer avatarLayer;

        static CDialogManager dialogManager;

        int step;
        string showtext;
        int nowShowTextSize;
        bool IsBitmapLoaded = false;
        bool IsDialoging = false;
        bool IsDialogLoad = false;
        bool IsPrintTips = false; //用於一個對話完畢，要到下一個對話的提示文字 (右下角的 next)

        string musicFromNPC = "";
        CDialog* nowDialog;
        string nowTxtName; //目前使用的文本名稱
        CTimer AddShowTextTimer;
```

```
        void Load_Image();
        void LoadDialog();
        void ShowText_Next();
        void DialogWithSound(); //例外處理 - 配合對話播出語音 (因為不是每句都有 所以用利外處理)
        bool DebugMode = DEBUG_MODE;
};
#pragma endregion

#pragma region - BossManager -
class CBossManager
{
    public:
        CBossManager();
        ~CBossManager();
        void Initialize();
        void Clear();
        void TargetBoss(int);
        void SetBattle(bool flag) {
            isBattle = flag;
        };
        bool IsBattle() {
            return isBattle;
        };
        void BossDead();
        CBoss* targetBoss;
    private:
        map<string, CBoss*> bossInformation;
        bool isBattle = false;
};
#pragma endregion

#pragma region - NPCManager -
class CNPCManager
{
    public:
        CNPCManager();
        ~CNPCManager();
        void Clear();
        void Initialize(int = NOW_MAP);
        void ChangeMap(int, int);
        void SetNPCValid(int, bool);
        void OnCycle(int, CPoint);
        vector<CNPC*>* GetNpc(int nowMap) {
            return &npc[nowMap];
        };
    private:
        vector<CNPC*> npc[99];

        void LoadNPC();
        int map_max_Number;
};
#pragma endregion

#pragma region - EventManager -
class CEventManager
{
    public:
        CEventManager();
        ~CEventManager();
        void Initialize();
        void SetGameStateRun(CGameStateRun*);
        void trigger();
    private:
        CGameStateRun* gameState = nullptr;
        bool tips = true;
        bool dialogWithXingting = false;
        bool dialogWithFacaiSeed = false;
};
```

```
#pragma endregion

#pragma region - CEndManager -
class CEndManager
{
   public:
      CEndManager();
      ~CEndManager();
      void Initialize();
      void Start(string);
      void Stop();
      bool GetEndingState() {
         return isEnding;
      };
      void OnCycle();
      bool IsPassEnd(string);
      static CEndManager* Instance() {
         return &endManager;
      };

      vector<string> GetAllEndName();
   private:
      map< string, CEnd > endmap;
      CEnd* nowEnd;
      CToumeiImage endBmp;

      CTimer time_remaining;

      int step = 0;
      int alpha = 0;

      bool isEnding = false;
      bool isLoadEnd = false;
      bool isFadeIn = true;
      bool isFadeOut = false;
      bool isOpenDialog = false;

      void LoadEnd();
      void Play();

      static CEndManager endManager;
};
#pragma endregion

#pragma region - ButtonManager -
class CButtonManager
{
   public:
      CButtonManager();
      ~CButtonManager();

      void Load();
      void Initialize();
      void Clear();

      void AddButton(CButton*);
      void CreateButton(BitmapPath, CPoint, bool, bool); //loadpath, init location, init state
      void CreateButton(CButton*);

      void ClickButton(string);

      bool IsCollisionMouse(string);
      bool GetState(string);
      string GetCollisionButtonName();

      void SetValid(bool _v) {
         isValid = _v;
      };
      void UpdateState(CPoint);
```

```
        void ShowButton();
        void OnCycle();

    private:
        map <string, CButton*> buttons;
        CPoint mouse;
        bool isValid = false;
};
#pragma endregion

#pragma region - CMapEditer -
class CMapEditer
{
    public:
        CMapEditer();
        ~CMapEditer();
        void Initialize();
        void AddImage(vector<string>);
        void SetImageXY(CPoint point) { };
        void SetMapMoveDir(string);
        void OnSave();
        void OnMove();
        void OnShow();

        void SelectBlock(CPoint);
        void DragBlock(CPoint); //拖曳 Block
        void DeleteBlock();

        bool GetMouseState() {
            return isMouseDown;
        };    //取得滑鼠狀態(是否按住)
        void SetMouseState(bool f) {
            isMouseDown = f;
        }; //設置滑鼠狀態

        void SetDPoint_MouseToTopLeft(CPoint mouse) {
            if (selectObj != NULL) dpoint_mouseToTopleft = CPoint(mouse.x - selectObj->bmp.Left() - cameraX,
mouse.y - selectObj->bmp.Top());
        };
        void SetDPoint_MouseToTopLeft() {
            dpoint_mouseToTopleft = CPoint(0, 0);
        };

        bool isPrintNowMap = false;
        string GetNowMap();

        void CreateReloadMapInformation(); //按 ESC 離開 mapEditer 時，寫下 reload 的資訊
        void LoadReloadMapInformation();    //進入 mapEditer 時，如果上次有留下來的 reload 資訊就留著

        void StoreMapInformaion(); //儲存所有 blockMap 的資料

        bool IsInSelectMapMode() { //是否在選擇地圖的模式中
            return (selectMapMode == "up" || selectMapMode == "down" || selectMapMode == "left" || selectMapMode ==
"right");
        }

        void SetSelectMapZero(); //上/下/左/右的連結地圖 -> 歸零
        void SetSelectMapMode(string _mode);
        string GetSelectMapMode() {
            return "Set" + selectMapMode + "Map";
        };

        void ClickArrow(CPoint);

        void SwitchMap(string);
        void GotoMap(int);

    private:
        #pragma region - image info -
```

```
class ImageInfo : public CSimpleMapObj
{
   public:
      ImageInfo() : CSimpleMapObj() { };
      ImageInfo(string _path, string _id) : CSimpleMapObj(_path, 0, 0)
      {
         id = _id;
         LoadImg();
         SetXY(x, y);
      };
      ~ImageInfo() {};
      string id;
};
#pragma endregion

ImageInfo background;
vector<ImageInfo> mapObj; //地圖上一些物件
ImageInfo* selectObj;

bool haveBG = false;
bool isMouseDown;

string saveTxtName;
CPoint dpoint_mouseToTopleft; //儲存滑鼠座標與 block 座標的 x, y 座標差

CBlockMap bkmap;

bool isMapRight, isMapLeft;
int cameraX;

bool isSaved; //曾經有儲存過地圖

void CreateObjToBkmap();
void LoadBlockMap(string);

int* printNowMap = NULL;
int nowMap;
int nowMapNumber; //reload 前的地圖數
vector<bool> reloadMap;

vector<CBlockMap> blockMap; //所有地圖
string selectMapMode;
int selectNowMap;

#pragma region - Arrow -
class Arrow
{
   public:
      Arrow() {
         dir = "";
         isLoad = false;
      };
      Arrow(string _dir) {
         dir = _dir;
      };
      ~Arrow() {};
      void Initialize() {
         LoadImg();
         canShow = true;
      }
      void LoadImg() {
         if (!isLoad)
         {
            #pragma region - Load Image -

            if (dir == "up")
            {
               bmp.LoadBitmap("RES\\Object\\Arrow\\up_arrow.bmp", RGB(255, 255, 255));
               bmp.SetTopLeft(320 - bmp.Width() / 2, 0);
```

```
                }
                else if (dir == "down")
                {
                    bmp.LoadBitmap("RES\\Object\\Arrow\\down_arrow.bmp", RGB(255, 255, 255));
                    bmp.SetTopLeft(320 - bmp.Width() / 2, 480 - bmp.Height());
                }
                else if (dir == "left")
                {
                    bmp.LoadBitmap("RES\\Object\\Arrow\\left_arrow.bmp", RGB(255, 255, 255));
                    bmp.SetTopLeft(0, 240 - bmp.Height() / 2);
                }
                else if (dir == "right")
                {
                    bmp.LoadBitmap("RES\\Object\\Arrow\\right_arrow.bmp", RGB(255, 255, 255));
                    bmp.SetTopLeft(620 - bmp.Width(), 240 - bmp.Height() / 2);
                }

                #pragma endregion
                isLoad = true;
            }
        }
        bool CanShow() {
            return canShow;
        };
        void SetCanShow(bool f) {
            canShow = f;
        };
        CMovingBitmap* GetBmp() {
            return &bmp;
        };
        string GetDir() {
            return dir;
        };
    private:
        string dir;
        CMovingBitmap bmp;
        bool isLoad = false;
        bool canShow;
};
#pragma endregion

Arrow arrow[4];
void SetArrowCanShow();

void NewMapInit();
};
#pragma endregion

#pragma region - UIManager -
class CRole;
class UIManager
{
  public:
    UIManager();

    void Load();
    void Initialize(CRole*, CBossManager*);
    void OnCycle(int);

  private:
    int left_time;   // Game left time
    int point; // Usless kill passerby point

    CInteger uiTime, uiScore;
    CStatusBoard roleStatus;
    CBossBoard bossStatus;

    CRole* role;
    CBossManager* bManager;
```

```
    };
    #pragma endregion
}
```

14. CManager.cpp

```
#pragma once
#include "stdafx.h"
#include "Resource.h"
#include "Refactor.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include <map>
#include <sstream>
#include <fstream>
#include "mygame.h"
#include "CManager.h"
using namespace myLibrary;

namespace game_framework
{
    #pragma region - mapManager -
    CMapManager::CMapManager()
    {
        max_map_number = getFolerFileNumber("RES\\Map\\Information\\");
        InitializeCBlockMap();
    }

    CMapManager::~CMapManager()
    {
    }

    vector<CMonster*>* CMapManager::GetPasserby()
    {
        return &passerbyManager.monsterManager;
    }

    void CMapManager::DeletePasserby(vector<CMonster*>::iterator passerbyj)
    {
        passerbyManager.DeleteMonster(passerbyj);
    }

    void CMapManager::AddPasserby()
    {
        CMonster* newPasserby = passerbyManager.AddMonster(blockMap[nowMap].passerbyID, background.Width());
        newPasserby->SetBlock(GetBlockVector());
        passerbyManager.monsterManager.push_back(newPasserby);
    }

    void CMapManager::Initialize()
    {
        ReloadBlockMap(); //每次 init 都檢查是否要重新載入 blockmap
        nowMap = NOW_MAP;
        loadMap = blockMap[nowMap].loadMap;
        x = 0;
        layer.SetLayer(0);
        background = blockMap[nowMap].backgroundBitmap;
        background.SetTopLeft(0, 0);
        CLayerManager::Instance()->AddObject(&background, layer.GetLayer());
        #pragma region - 設置 block -

        for (int i = 0 ; i < max_map_number; i++)
        {
            for (vector<CBlock>::iterator bkiter = blockMap[i].block.begin(); bkiter != blockMap[i].block.end(); bkiter++)
            {
                bkiter->bmp.SetValid(false | (i == nowMap));
                bkiter->SetXY(bkiter->x, bkiter->y);
```

```
            CLayerManager::Instance()->AddObject(&(bkiter->bmp), BLOCK_LAYER);
        }
    }

    #pragma endregion
    #pragma region - 設置 Door -

    for (int i = 0; i < max_map_number; i++)
    {
        for (vector<CDoor>::iterator bkiter = blockMap[i].door.begin(); bkiter != blockMap[i].door.end(); bkiter++)
        {
            bkiter->bmp.SetValid(false | (i == nowMap));
            bkiter->SetXY(bkiter->x, bkiter->y);
            CLayerManager::Instance()->AddObject(&(bkiter->bmp), BLOCK_LAYER);
        }
    }

    #pragma endregion
    passerbyManager.CreateMonster(blockMap[nowMap].passerbyMaxSize, blockMap[nowMap].passerbyID,
blockMap[nowMap].backgroundBitmap.Width(), GetBlockVector());
}

void CMapManager::ReloadBlockMap()
{
    bool isreloadDataExist = false; //先假設檔案不存在
    fstream reloadData;
    reloadData.open("RES\\Map\\ReloadMapInformation.txt");
    #pragma region - 檔案存在 - reload -

    if (reloadData.is_open()) //檔案存在
    {
        isreloadDataExist = true;
        string line; //temp string
        #pragma region -- 檢查是否有新增的地圖 --
        reloadData >> line; // load first string in reloadData
        int nowMapNumber = ConvertStringToInt(line); //現在 Map - Information 有的地圖總數
        int nowBlockSize = blockMap.size();

        if (nowBlockSize < nowMapNumber) //blockMap 的 size 比 目前的地圖總數小 > 新增地圖
        {
            max_map_number = nowMapNumber;

            for (int mapIndex = nowBlockSize; mapIndex < nowMapNumber; mapIndex++)
            {
                blockMap.push_back(CBlockMap(mapIndex));
            }
        }

        #pragma endregion
        #pragma region -- 重新載入編輯過的地圖 --

        while (reloadData >> line)
        {
            if (line != "")
            {
                int reloadMapIndex = ConvertStringToInt(line); //取得要重新載入的地圖編號

                if (reloadMapIndex < nowBlockSize) //small small 防呆
                {
                    blockMap[reloadMapIndex] = CBlockMap(reloadMapIndex);
                }
            }
        }

        #pragma endregion
        #pragma region -- 新增&重新載入完成 - 載入新的圖片 --
        LoadMapBitmap();
        #pragma endregion
    }
```

```
    #pragma endregion
    reloadData.close();
    #pragma region - 檔案存在 - 刪檔 -

    if (isreloadDataExist)
    {
        DeleteFile("RES\\Map\\ReloadMapInformation.txt");
    }

    #pragma endregion
}

#pragma region - GetMap -

int CMapManager::GetNowMap()
{
    return nowMap;
}

int CMapManager::GetUpMap()
{
    return blockMap[nowMap].upMap;
}

int CMapManager::GetDownMap()
{
    return blockMap[nowMap].downMap;
}

int CMapManager::GetLeftMap()
{
    return blockMap[nowMap].leftMap;
}

int CMapManager::GetRightMap()
{
    return blockMap[nowMap].rightMap;
}

int CMapManager::GetLoadMap()
{
    return loadMap;
}

int CMapManager::GetNpcNumber()
{
    return passerbyManager.monsterManager.size();
}

int CMapManager::GetNpcLayer(int npcIndex)
{
    return passerbyManager.monsterManager[npcIndex]->layer.GetLayer();
}

bool CMapManager::GetNpcValid(int npcIndex)
{
    return passerbyManager.monsterManager[npcIndex]->GetValid();
}

vector<CBlock>* CMapManager::GetBlockVector()
{
    return &blockMap[nowMap].block;
}

vector<CDoor>* CMapManager::GetDoorVector()
{
    return &blockMap[nowMap].door;
}
```

```
CMovingBitmap* CMapManager::GetBitmap()
{
    return &background;
}

CAnimate* CMapManager::GetNpc(int npcIndex)
{
    return (passerbyManager.monsterManager[npcIndex]->GetAnimate());
}

int CMapManager::GetSplitLeft()
{
    return 0 + (SIZE_X / 2);
}

int CMapManager::GetSplitRight()
{
    return background.Width() - (SIZE_X / 2);
}

int CMapManager::GetX1()
{
    return x;
}

int CMapManager::GetX2()
{
    return x + (SIZE_X);
}

int CMapManager::GetBitmapWidth()
{
    return background.Width();
}
#pragma endregion

void CMapManager::ChangeMap(int changeMap, string nextMap)
{
    #pragma region - 設置 block -

    //關掉原本的
    for (vector<CBlock>::iterator bkiter = blockMap[nowMap].block.begin(); bkiter != blockMap[nowMap].block.end();
bkiter++)
    {
        bkiter->bmp.SetValid(false);
    }

    //打開後來的
    for (vector<CBlock>::iterator bkiter = blockMap[changeMap].block.begin(); bkiter !=
blockMap[changeMap].block.end(); bkiter++)
    {
        bkiter->bmp.SetValid(true);
    }

    #pragma endregion
    #pragma region - 設置 Door -

    //關掉原本的
    for (vector<CDoor>::iterator bkiter = blockMap[nowMap].door.begin(); bkiter != blockMap[nowMap].door.end();
bkiter++)
    {
        bkiter->bmp.SetValid(false);
    }

    //打開後來的
    for (vector<CDoor>::iterator bkiter = blockMap[changeMap].door.begin(); bkiter !=
blockMap[changeMap].door.end(); bkiter++)
    {
```

```
            bkiter->bmp.SetValid(true);
        }

        #pragma endregion
        nowMap = changeMap;
        loadMap = blockMap[nowMap].loadMap;
        background = blockMap[nowMap].backgroundBitmap;
        CCamera::Instance()->Reset();

        if (nextMap == "left") //下一張地圖(要換的地圖)，在原本地圖的左邊 (移動到左邊的地圖)
        {
            CCamera::Instance()->SetXY((GetBitmapWidth() - SIZE_X), 0);
        }
        else if (nextMap == "right")  //下一張地圖(要換的地圖)，在原本地圖的右邊 (移動到右邊的地圖)
        {
            CCamera::Instance()->SetXY(0, 0);
        }

        background.SetTopLeft(x, 0);
        passerbyManager.Clear();
        passerbyManager.CreateMonster(blockMap[nowMap].passerbyMaxSize, blockMap[nowMap].passerbyID,
blockMap[nowMap].backgroundBitmap.Width(), GetBlockVector());
        x = CCamera::Instance()->GetX();
    }

    void CMapManager::SetMovingLeft(bool _flag)
    {
        isMovingLeft = _flag;
    }

    void CMapManager::SetMovingRight(bool _flag)
    {
        isMovingRight = _flag;
    }

    void CMapManager::OnMove()
    {
        #pragma region - 螢幕上的 background 要移動 -
        int dx = CCamera::Instance()->GetX();
        x = -dx;
        SetXY(x, 0);
        #pragma endregion
        #pragma region - 螢幕上的 block 也要移動 -

        for (vector<CBlock>::iterator bkiter = blockMap[nowMap].block.begin(); bkiter != blockMap[nowMap].block.end();
bkiter++)
        {
            bkiter->SetXY(bkiter->x, bkiter->y, dx);
        }

        #pragma endregion
        #pragma region - 螢幕上的 door 也要移動 -

        for (vector<CDoor>::iterator bkiter = blockMap[nowMap].door.begin(); bkiter != blockMap[nowMap].door.end();
bkiter++)
        {
            bkiter->SetXY(bkiter->x, bkiter->y, dx);
        }

        #pragma endregion
    }

    void CMapManager::SetXY(int _x, int _y)
    {
        x = _x;

        if (x <= -((background.Width()) - 640))
            x = -((background.Width()) - 640);
```

```
    if (x >= 0)
        x = 0;

    background.SetTopLeft(x, _y);
}

bool CMapManager::IsMapEnd()
{
    return x < 0 || x > ((background.Width()) - 640);
}

void CMapManager::OnShow() //顯示對應到的 blockMap 圖片 (nowMap = 1, 顯示 blockMap[1]的 background, 類推)
{
    background.SetTopLeft(0, 0);
    background.ShowBitmap();
}

void CMapManager::InitializeCBlockMap()
{
    for (int mapIndex = 0; mapIndex < max_map_number; mapIndex++) //初始化 blockMap 的上下左右地圖資訊，增加可讀性使用 switch 敘述
    {
        blockMap.push_back(CBlockMap(mapIndex));
    }
}

void CMapManager::LoadMapBitmap() //如字面意思，LoadMapBitmap，在 GameStateRun:OnInit 運行，一次性 load blockMap 的所有圖片
{
    for (int mapIndex = 0; mapIndex < max_map_number; mapIndex++)
    {
        if (!blockMap[mapIndex].isLoad)
        {
            blockMap[mapIndex].LoadImg();
            blockMap[mapIndex].isLoad = true;
        }
    }
}
#pragma endregion

class CDialogManager;
#pragma region - layerManager -
CLayerManager CLayerManager::layerManager;
CLayerManager::CLayerManager()
{
    Clear();
}

CLayerManager::~CLayerManager()
{
}

void CLayerManager::Clear()
{
    for (int i = 0; i < MAX_LAYER_NUMBER; i++)
    {
        layerBitmap[i].clear();
        layerAnimate[i].clear();
        layerAction[i].clear();
        layerInteger[i].clear();
    }
}

void CLayerManager::AddObject(CMovingBitmap* object, int targetLayer)
{
    layerBitmap[targetLayer].push_back(object);
}
```

```
void CLayerManager::AddObject(CAnimate* object, int targetLayer)
{
   layerAnimate[targetLayer].push_back(object);
}

void CLayerManager::AddObject(CAction* object, int targetLayer)
{
   layerAction[targetLayer].push_back(object);
}

void CLayerManager::AddObject(CInteger* object, int targetLayer)
{
   layerInteger[targetLayer].push_back(object);
}

void CLayerManager::ShowLayer()
{
   for (int i = 0; i < MAX_LAYER_NUMBER; i++)
   {
      for (vector<CMovingBitmap*>::iterator k = layerBitmap[i].begin(); k != layerBitmap[i].end(); k++)
      {
         if ((*k)->GetValid())
         {
            (*k)->ShowBitmap();
         }
      }

      for (vector<CAnimate*>::iterator k = layerAnimate[i].begin(); k != layerAnimate[i].end();)
      {
         if ((*k)->IsNull())
         {
            k = layerAnimate[i].erase(k);
         }
         else
         {
            if ((*k)->GetValid())       //if valid then show
            {
               (*k)->OnShow();
            }

            k++;
         }
      }

      for (vector<CAction*>::iterator k = layerAction[i].begin(); k != layerAction[i].end();)
      {
         if ((*k)->IsNull())
         {
            k = layerAction[i].erase(k);
         }
         else
         {
            if ((*k)->GetValid())       //if valid then show
            {
               (*k)->OnShow();
            }

            k++;
         }
      }

      for (vector<CInteger*>::iterator k = layerInteger[i].begin(); k != layerInteger[i].end();)
      {
         if ((*k)->IsNull())
         {
            k = layerInteger[i].erase(k);
         }
         else
         {
```

```
            if ((*k)->GetValid())       //if valid then show
            {
               (*k)->ShowBitmap();
            }

            k++;
         }
      }

      if (CDialogManager::Instance()->GetDialogState() && i == 8)
      {
         CDialogManager::Instance()->ShowText();
      }
   }
}

CLayerManager* CLayerManager::Instance()
{
   return &layerManager;
}

void CLayerManager::Initialize()
{
   Clear();
}
#pragma endregion

#pragma region - passerbyManager -
CMonsterManager::CMonsterManager()
{
   layer.SetLayer(4);
   monsterManager.clear();
}

CMonsterManager::~CMonsterManager()
{
   Clear();
}

void CMonsterManager::Clear()
{
   for (vector<CMonster*>::iterator it = monsterManager.begin(); it != monsterManager.end(); it++)
   {
      delete *it;
      (*it) = NULL;
   }

   vector<CMonster*> del;
   monsterManager.swap(del);
   monsterManager.clear();
}

CMonster* CMonsterManager::GetMonsterType(int randomID)
{
   if (randomID == 0)
   {
      return new CMonsterType1(0, 0, BitmapPath("RES\\Role\\NPC\\LUKA", "LUKA", 2, RGB(214, 214, 214)), 10);
//先創建一個 default passerby
   }
   else if (randomID == 1)
   {
      return new CMonsterType1(0, 0, BitmapPath("RES\\Role\\NPC\\RIN", "RIN", 2, RGB(214, 214, 214)), 15); //先
創建一個 default passerby
   }
   else if (randomID == 2)
   {
      return new CMonsterType2(0, 0, BitmapPath("RES\\Role\\NPC\\mushroom", "mushroom", 5, RGB(214, 214,
214)), 20); //先創建一個 default passerby
   }
```

```
    else if (randomID == 3)
    {
      return new CMonsterType2(0, 0, BitmapPath("RES\\Role\\NPC\\faqai", "faqai", 5, RGB(255, 255, 255)), 87); //先創建一個 default passerby
    }

    return new CMonsterType1(0, 0, BitmapPath("RES\\Role\\NPC\\LUKA", "LUKA", 2, RGB(214, 214, 214)), 10); //先創建一個 default passerby
  }

  CMonster* CMonsterManager::AddMonster(vector<int> id, int mapWidth)
  {
    #pragma region Create a Passerby
    int randomID = GetRandom(0, id.size() - 1); //random 決定 passerby 種類 (1 號 or2 號)
    CMonster* newPasserby = GetMonsterType(id[randomID]);
    int randomX = GetRandom(0, mapWidth - newPasserby->Width()); //random 決定 passerby 的出現位置
    newPasserby->SetXY(randomX, -newPasserby->Height()); //set passerby x, y
    return newPasserby;
    #pragma endregion
  }

  void CMonsterManager::CreateMonster(int createNumber, vector<int> id, int mapWidth, vector<CBlock>* bkvector)
  {
    for (int i = 0; i < createNumber; i++)
    {
      CMonster* newPasserby = AddMonster(id, mapWidth);
      newPasserby->SetBlock(bkvector);
      newPasserby->SetValid(false);
      monsterManager.push_back(newPasserby);
    }
  }

  void CMonsterManager::DeleteMonster(vector<CMonster*>::iterator passerbyj)
  {
    delete *passerbyj;
    *passerbyj = NULL;
    passerbyj = monsterManager.erase(passerbyj);
  }

  vector<CMonster*> CMonsterManager::GetMonster()
  {
    return monsterManager;
  }
  #pragma endregion

  #pragma region - CDialogManager -
  CDialogManager CDialogManager::dialogManager;

  CDialogManager::CDialogManager()
  {
    IsBitmapLoaded = false;
    IsDialoging = false;
    IsPrintTips = false;
    nowDialog = NULL;
    showtext = "";
    txt.clear();
    nowShowTextSize = 0;
    AddShowTextTimer = CTimer(DIALOG_ADDTEXT_TIME);
  }

  CDialogManager::~CDialogManager()
  {
    dialogmap.clear();
    nowDialog = NULL;
  }

  void CDialogManager::OnCycle()
  {
    #pragma region - add show text -
```

```
    if (showtext != "" && nowShowTextSize < (int)showtext.size())
    {
        AddShowTextTimer.CountDown();

        if (AddShowTextTimer.IsTimeOut()) //add show text
        {
            if (showtext[nowShowTextSize] < 0) //非 ASCII 體系中的字
            {
                nowShowTextSize += 2;
            }
            else
            {
                nowShowTextSize++;
            }

            AddShowTextTimer.ResetTime();
            ShowText();
        }
    }

    #pragma endregion
    #pragma region - print finish tips (next) -

    if (showtext != "" && nowShowTextSize >= (int)showtext.size())
    {
        IsPrintTips = true;
    }

    #pragma endregion
}

void CDialogManager::Load_Image()
{
    #pragma region - Load avatar -
    vector<string> avatarName;
    string avatarFolderPath = "RES\\Dialog\\Avatar\\";
    getFolderFile(avatarFolderPath, &avatarName);

    for (unsigned i = 0; i < avatarName.size(); i++)
    {
        int strLength = avatarName[i].length();
        string ext = avatarName[i].substr(strLength - 4, strLength);

        if (ext == ".bmp")
        {
            string avatar = getFileName(avatarName[i]);
            dialogAvatar[avatar] = CMovingBitmap();
            char* address = ConvertCharPointToString((avatarFolderPath + avatarName[i]));
            dialogAvatar[avatar].LoadBitmap(address, RGB(214, 214, 214));
            delete address;
        }
    }

    #pragma endregion
    #pragma region - load image -
    #pragma region - load dialog background -
    dialog_background.LoadBitmap("RES\\Dialog", "ground", RGB(255, 255, 255));
    #pragma endregion
    #pragma region - load null -
    avatar_null.LoadBitmap("RES\\Dialog\\Avatar", "null");
    #pragma endregion
    #pragma region - load next animation -
    textNext.LoadBitmap("RES\\Dialog\\next", "next", 4, DIALOG_BACKGROUND_COLOR);
    textNext.SetTopLeft(500, 450);
    #pragma endregion
    #pragma endregion
    IsBitmapLoaded = true;
}
```

```
void CDialogManager::LoadDialog()
{
    vector<string> dialogTxt;
    string dialogFolderPath = "RES\\Dialog\\Txt\\";
    getFolderFile(dialogFolderPath, &dialogTxt);

    for (unsigned int i = 0; i < dialogTxt.size(); i++)
    {
        string dialogFileName = getFileName(dialogTxt[i]);
        dialogmap[dialogFileName] = CDialog(dialogFolderPath + dialogTxt[i], dialogFileName);
    }

    IsDialogLoad = true;
}


void CDialogManager::LoadDialog(string dialogName, string loadPath)
{
    dialogmap[dialogName] = CDialog(loadPath, dialogName);
}

void CDialogManager::ShowText_Next()
{
    if (IsPrintTips)
    {
        textNext.SetValid(true);
        textNext.OnMove();
    }
    else
    {
        textNext.SetValid(false);
    }
}

void CDialogManager::DialogWithSound()
{
    if (nowTxtName == DIALOG_DATA_FAQAI)
    {
        if (step == 3 || step == 5)
        {
            CAudio::Instance()->Play("faqai");
        }
    }
    else if (nowTxtName == "roleWinFacaiSeed")
    {
        if (step == 1)
        {
            CAudio::Instance()->Play("faqai");
        }
    }
}

void CDialogManager::Initialize()
{
    if (IsBitmapLoaded == false)
    {
        Load_Image();
        #pragma region - Init - Image Point -
        dialog_background.SetTopLeft(0, SIZE_Y - dialog_background.Height()); //reset dbg's xy

        for (map<string, CMovingBitmap>::iterator dbmp = dialogAvatar.begin(); dbmp != dialogAvatar.end(); dbmp++)
        {
            dbmp->second.SetTopLeft(dialog_background.Left() + MARGIN_DIALOG_AVATAR,
dialog_background.Top() + MARGIN_DIALOG_AVATAR);
        }

        #pragma endregion
    }
```

```
    #pragma region - reload - dialog -

    if (IsDialogLoad == false)
    {
        LoadDialog();
        IsDialogLoad = true;
    }
    else
    {
        for (map<string, CDialog>::iterator dialogiter = dialogmap.begin(); dialogiter != dialogmap.end(); dialogiter++)
        {
            dialogiter->second.Initialize();
        }
    }

    #pragma endregion
    avatar_null.SetValid(false);
    dialog_background.SetValid(false);
    textNext.SetValid(false);
    avatar = avatar_null;
    backgroundLayer.SetLayer(8);
    avatarLayer.SetLayer(backgroundLayer.GetLayer() + 1);
    CLayerManager::Instance()->AddObject(&dialog_background, backgroundLayer.GetLayer());
    CLayerManager::Instance()->AddObject(&avatar, avatarLayer.GetLayer());
    CLayerManager::Instance()->AddObject(&textNext, avatarLayer.GetLayer());
    nowDialog = NULL;
    step = 0;
    nowShowTextSize = 0;
    IsPrintTips = false;
    musicFromNPC = "";
    AddShowTextTimer.ResetTime(DIALOG_ADDTEXT_TIME);
}

CDialogManager* CDialogManager::Instance()
{
    return &dialogManager;
}

void CDialogManager::Start(string mode)
{
    if (DebugMode)
        return;

    nowDialog = &dialogmap[mode];

    if (nowDialog->GetTriggered()) //被觸發過 回去
    {
        nowDialog = NULL;
        return;
    }

    nowTxtName = mode;
    IsDialoging = true;
    IsPrintTips = false;
    dialog_background.SetValid(true);
    Dialog();
}

void CDialogManager::Next()
{
    if (IsDialoging)
    {
        if (nowShowTextSize < (int)showtext.size()) //當字還沒顯示完時，就已經點擊畫面 > 直接顯示完全部文字
        {
            nowShowTextSize = (int)showtext.size();
        }
        else //字已經顯示完了，可以切換
        {
```

```
            step++;
            IsPrintTips = false;
            nowShowTextSize = 0;
            Dialog();
        }
    }
}

void CDialogManager::Stop()
{
    if (nowDialog != NULL)
    {
        avatar = avatar_null;
        dialog_background.SetValid(false);
        textNext.SetValid(false);
        IsDialoging = false;
        IsPrintTips = false;
        nowDialog->SetTriggered();
        nowDialog = NULL;
        nowShowTextSize = 0;
        step = 0;
        nowTxtName = "";

        if (musicFromNPC != "")
        {
            CAudio::Instance()->Stop(musicFromNPC);
            CAudio::Instance()->Play("SLoWMoTIoN_Game");
            musicFromNPC = "";
        }
    }
}

void CDialogManager::ShowText()
{
    if (showtext != "")
    {
        #pragma region - split showtext -
        vector<char*> split_showtext;
        int charindex = 0;
        int vectorindex = 0;
        char tempk[DIALOG_MAX_TEXT + 5];
        memset(tempk, '\0', sizeof(tempk));

        for (int i = 0; i < nowShowTextSize;)
        {
            if (charindex < DIALOG_MAX_TEXT)
            {
                //將字元加入 tempk, 當字元超過一定量的時候給 split-showtext(換行)
                if (showtext[i] == '_') //例外處理，將底線換成空白 (文本不能打空白)
                {
                    tempk[charindex++] = ' ';
                    i++;
                }
                else    if (showtext[i] == '@') //將 at 換成換行
                {
                    charindex += 999;
                    i++;
                }
                else
                {
                    if (showtext[i] > 0)
                    {
                        tempk[charindex++] = showtext[i++];
                    }
                    else
                    {
                        if (charindex + 1 < DIALOG_MAX_TEXT) //add a chinese
                        {
                            tempk[charindex++] = showtext[i++];
```

```
                    tempk[charindex++] = showtext[i++];
                }
                else
                {
                    charindex += 999;
                }
            }
        }

        #pragma region -- 針對倒數第二個字元是英文，但最後一個字是中文的額外設定 --
        #pragma region --- 條件 & 說明 ---
        /*
        1. 目前文字是英文
        2. 下一個字是中文
        3. 下一個字在 29
        說明：問題發生的原因是因為英文的 size 是 1，但中文是 2，而我設定當 size 到 30 就會換行
        因此如果 size=28 的時候裝一個英文字(裝一個英文字後 size=29)
        但下一個字是中文時，照理講要裝兩個 size 才會正確，但因為只能裝一個，另一個會到下一行才裝
        此時中文字不會正常顯示，會變成亂碼
        解決方式：符合上述條件就直接先換行
        */
        #pragma endregion

        if (showtext[i] > 0 && showtext[i + 1] < 0 && charindex + 1 >= DIALOG_MAX_TEXT)
        {
            charindex += 999;
        }

        #pragma endregion
    }

    if (charindex >= DIALOG_MAX_TEXT)
    {
        char* tteemmppkkk = new char[DIALOG_MAX_TEXT + 5];
        strcpy(tteemmppkkk, tempk);
        split_showtext.push_back(tteemmppkkk);
        charindex = 0;
        memset(tempk, '\0', sizeof(tempk));
    }
}

char* tteemmppkkk = new char[DIALOG_MAX_TEXT + 5];
strcpy(tteemmppkkk, tempk);
split_showtext.push_back(tteemmppkkk);
#pragma endregion
#pragma region - Set Print Text Color -
COLORREF txtColor = DIALOG_TEXT_COLOR;

if (nowDialog != NULL)
{
    txtColor = nowDialog->GetDialogColor(step);
}

#pragma endregion
#pragma region - draw text -

for (unsigned int i = 0; i < split_showtext.size(); i++)
{
    PaintText(split_showtext[i], avatar.Left() + avatar.Width() + MARGIN_DIALOG_TEXT_X, avatar.Top() +
MARGIN_DIALOG_TEXT_Y + i * 35, "微軟正黑體", DIALOG_TEXT_SIZE, txtColor,
DIALOG_BACKGROUND_COLOR);
    delete split_showtext[i];
}

#pragma endregion
#pragma region - draw finish tips text -
ShowText_Next();
#pragma endregion
}
```

```
}

bool CDialogManager::GetDialogState()
{
    return IsDialoging;
}

void CDialogManager::Dialog()
{
    if (step < nowDialog->GetTxtSize())
    {
        avatar = dialogAvatar[nowDialog->GetAvatar(step)];
        showtext = nowDialog->GetDialogTxt(step);
        DialogWithSound();
    }
    else
    {
        step = 0;
        showtext = "";
        Stop();
    }
}
#pragma endregion

#pragma region - BossManager -
CBossManager::CBossManager()
{
    #pragma region -- Create Boss --
    bossInformation[BOSS_XINGTING] = new CXingting(450, 250, 8787, "Xingting",
BitmapPath("RES\\Boss\\xingting", "xingting", 2, RGB(214, 214, 214)));
    bossInformation[BOSS_FACAISEED] = new CFacaiSeed(100, 270, 35, BOSS_FACAISEED,
BitmapPath("RES\\Boss\\FacaiSeed", "faqai", 19, RGB(255, 255, 255)));
    #pragma endregion
    targetBoss = NULL;
    isBattle = false;
}

CBossManager::~CBossManager()
{
    if (targetBoss != NULL)
    {
        targetBoss->Clear();
    }

    Clear();
}

void CBossManager::Initialize() //CGameRun::OnBeginState 時 initialize
{
    //(second(value)為指標，所以 second 後面用-> )
    for (map<string, CBoss*>::iterator bossiter = bossInformation.begin(); bossiter != bossInformation.end();
bossiter++)
    {
        bossiter->second->Initialize(); //初始化 boss
    }

    targetBoss = NULL;
    isBattle = false;
}

void CBossManager::Clear()
{
    //解構
    for (map<string, CBoss*>::iterator bossiter = bossInformation.begin(); bossiter != bossInformation.end();
bossiter++)
    {
        delete bossiter->second;
        bossiter->second = NULL;
    }
```

```
    bossInformation.clear();
    targetBoss = NULL;
}

void CBossManager::TargetBoss(int nowMap)
{
    if (nowMap == BOSS_MAP_XINGTING && bossInformation[BOSS_XINGTING]->GetAlive())
    {
        targetBoss = bossInformation[BOSS_XINGTING];
        targetBoss->Initialize();
        targetBoss->GetAnimate()->SetValid(true);
    }
    else if (nowMap == BOSS_MAP_FACAISEED && (bossInformation[BOSS_FACAISEED]->GetAlive()
&& !bossInformation[BOSS_FACAISEED]->IsEnd()))
    {
        targetBoss = bossInformation[BOSS_FACAISEED];
        targetBoss->Initialize();
        targetBoss->GetAnimate()->SetValid(true);
    }
    else
    {
        if (targetBoss != NULL)
        {
            targetBoss->GetAnimate()->SetValid(false);
            targetBoss->Clear();
            isBattle = false;
            targetBoss = NULL;
        }
    }
}

void CBossManager::BossDead()
{
    targetBoss->Clear();
    targetBoss = NULL;
    SetBattle(false);
}
#pragma endregion

#pragma region - CNPCManager -
CNPCManager::CNPCManager()
{
    Clear();
    LoadNPC();
}

CNPCManager::~CNPCManager()
{
    Clear();
}

void CNPCManager::Clear()
{
    for (int i = 0; i < MAX_MAP_NUMBER; i++)
    {
        for (vector<CNPC*>::iterator npciter = npc[i].begin(); npciter != npc[i].end(); ++npciter)
        {
            delete (*npciter);
            (*npciter) = NULL;
        }

        npc[i].clear();
    }
}

void CNPCManager::Initialize(int nowMap)
{
    map_max_Number = getFolerFileNumber("RES\\Map\\Information\\");
```

```
    for (int i = 0; i < MAX_MAP_NUMBER; i++)
    {
        for (vector<CNPC*>::iterator npciter = npc[i].begin(); npciter != npc[i].end(); npciter++)
        {
            (*npciter)->Initialize();
        }
    }

    SetNPCValid(nowMap, true);
}

void CNPCManager::ChangeMap(int nowMap, int nextMap)
{
    SetNPCValid(nowMap, false);
    SetNPCValid(nextMap, true);
}

void CNPCManager::SetNPCValid(int thisMap, bool flag)
{
    for (vector<CNPC*>::iterator npciter = npc[thisMap].begin(); npciter != npc[thisMap].end(); npciter++)
    {
        (*npciter)->SetValid(flag);
    }
}

void CNPCManager::OnCycle(int thisMap, CPoint rolePoint)
{
    for (vector<CNPC*>::iterator npciter = npc[thisMap].begin(); npciter != npc[thisMap].end(); npciter++)
    {
        (*npciter)->OnCycle(rolePoint);
    }
}

void CNPCManager::LoadNPC()
{
    npc[4].push_back(new CNPC1(CPoint(50, 388), BitmapPath("RES\\NPC\\test", "test", 1, RGB(255, 255, 255)),
"frog", FROG));
    npc[4].push_back(new CNPC3(CPoint(450, 388), BitmapPath("RES\\NPC\\test2", "test2", 1, RGB(255, 255, 255)),
"deadlock", "MyVoiceIsDead", MyVoiceIsDead));
    npc[3].push_back(new CNPC1(CPoint(50, 350), BitmapPath("RES\\NPC\\studentB", "studentB", 1, RGB(255, 255,
255)), DIALOG_AVATAR_NAME_STUDENTB, DIALOG_DATA_MEETSTB));
    npc[3].push_back(new CNPC1(CPoint(450, 350), BitmapPath("RES\\NPC\\studentG", "studentG", 1, RGB(255,
255, 255)), DIALOG_AVATAR_NAME_STUDENTG, DIALOG_DATA_STGHAVEBREAKFAST));
    npc[3].push_back(new CNPC1(CPoint(560, 90), BitmapPath("RES\\NPC\\faqai", "faqai", 20, RGB(255, 255, 255)),
"faqai", DIALOG_DATA_FAQAI));
    npc[7].push_back(new CNPC1(CPoint(400, 360), BitmapPath("RES\\NPC\\an", "an", 6, RGB(214, 214, 214)), "an",
"EasyAndroidHomework"));
    npc[8].push_back(new CNPC3(CPoint(680, 360), BitmapPath("RES\\NPC\\remilia_music", "remilia", 4, RGB(254,
254, 254)), "remilia", "Septette", "Music_Septette"));
    npc[8].push_back(new CNPC3(CPoint(900, 360), BitmapPath("RES\\NPC\\flandre_music", "flandre", 4, RGB(214,
214, 214)), "flandre", "UN_OwenWasHer", "Music_UN"));
    npc[8].push_back(new CNPC3(CPoint(340, 340), BitmapPath("RES\\NPC\\sakuya_music", "sakuya", 6, RGB(214,
214, 214)), "sakuya", "FloweringNight", "Music_FloweringNight"));
    npc[8].push_back(new CNPC3(CPoint(120, 360), BitmapPath("RES\\NPC\\suwako_music", "suwako", 2, RGB(214,
214, 214)), "suwako", "NativeFaith", "Music_NativeFaith"));
    npc[10].push_back(new CNPC1(CPoint(100, 360), BitmapPath("RES\\NPC\\mushroom", "mushroom", 4, RGB(214,
214, 214)), "mushroom", "mushroom"));
}
#pragma endregion

#pragma region - CEventManager -
CEventManager::CEventManager()
{
    gameState = NULL;
}

CEventManager::~CEventManager()
{
```

```
}

void CEventManager::Initialize()
{
    if (gameState != NULL)
    {
        dialogWithXingting = false;
        dialogWithFacaiSeed = false;
    }
}

void CEventManager::SetGameStateRun(CGameStateRun* state)
{
    if (gameState == NULL)
    {
        gameState = state;
    }
}

void CEventManager::trigger()
{
    if (gameState != NULL)
    {
        #pragma region - local variable -
        int nowMap = gameState->mapManager.GetNowMap();
        int rolePosition = ScreenX(gameState->mapManager.GetX1(), gameState->role.GetX3());
        #pragma endregion
        #pragma region - dialog - tips -

        if (tips)
        {
            CDialogManager::Instance()->Start(Tips);
            tips = false;
        }

        #pragma endregion
        #pragma region - dialog - vs xingting -

        if (!dialogWithXingting && nowMap == BOSS_MAP_XINGTING && gameState->role.IsDead() == false &&
gameState->bossManager.IsBattle() == false)
        {
            if (rolePosition >= 100)
            {
                CDialogManager::Instance()->Start(DIALOG_DATA_VSXingting1);
                dialogWithXingting = true;
                gameState->bossManager.SetBattle(true);
                gameState->SwitchTimer(gameState->bossManager.targetBoss->GetAliveTimer());
            }
        }

        #pragma endregion
        #pragma region - dialog - with FacaiSeed -

        if (!dialogWithFacaiSeed && nowMap == BOSS_MAP_FACAISEED && gameState->role.IsDead() == false
&& gameState->bossManager.IsBattle() == false)
        {
            if (rolePosition <= 540)
            {
                CDialogManager::Instance()->Start("roleVsFacaiSeed1");
                dialogWithFacaiSeed = true;
                gameState->bossManager.SetBattle(true);
                gameState->SwitchTimer(gameState->bossManager.targetBoss->GetAliveTimer());
            }
        }

        #pragma endregion
    }
}
#pragma endregion
```

```
#pragma region    - CEndManager -
CEndManager CEndManager::endManager;
CEndManager::CEndManager()
{
   endBmp.SetFadeInOut(30, -40);
   Initialize();
}

CEndManager::~CEndManager()
{
   //防止 map 更改 HaveEnd 的順序 > 先讀取 在寫入
   #pragma region - load -
   fstream haveEnd_open;
   string tempName, nonstr;
   vector<string> endName;
   haveEnd_open.open("RES\\End\\HaveEnd.txt", ios::in);

   while (haveEnd_open >> tempName >> nonstr)
   {
      endName.push_back(tempName);
   }

   haveEnd_open.close();
   #pragma endregion
   #pragma region - write -
   fstream haveEnd_write;
   haveEnd_write.open("RES\\End\\HaveEnd.txt", ios::out);

   for (unsigned int i = 0; i < endName.size(); i++)
   {
      string writeLine = endName[i] + " " + (endmap[endName[i]].IsGetEnd() ? "true" : "false");
      haveEnd_write << writeLine << "\n";
   }

   haveEnd_write.close();
   #pragma endregion
}

void CEndManager::Initialize()
{
   if (isLoadEnd == false)
   {
      LoadEnd();
      isLoadEnd = true;
   }

   nowEnd = nullptr;
   isEnding = false;
   step = 0;
   alpha = 0;
   time_remaining = CTimer(1.0);
   isFadeIn = true;
   isFadeOut = false;
   isOpenDialog = false;
}

void CEndManager::LoadEnd()
{
   endmap[END_NAME_WINXINGTING] = CEnd(END_NAME_WINXINGTING);
   endmap[END_NAME_LOSEXINGTING] = CEnd(END_NAME_LOSEXINGTING);
   endmap[END_NAME_SALTEDFISH] = CEnd(END_NAME_SALTEDFISH);
   endmap[END_NAME_WIN_FACAISEED] = CEnd(END_NAME_WIN_FACAISEED);
   endmap[END_NAME_LOSE_FACAISEED] = CEnd(END_NAME_LOSE_FACAISEED);
   #pragma region - open HaveEnd.txt to Load The End is Get -
   fstream haveEnd;
   haveEnd.open("RES\\End\\HaveEnd.txt", ios::in);
   string lineData;
   map<string, CEnd>::iterator iter;
```

```
    while (getline(haveEnd, lineData))
    {
        //lineInfo[0] = name, [1] = have end (true or false)
        vector<string> lineInfo = SplitString(lineData);
        #pragma region - find lineInfo[0] is the key of map -
        iter = endmap.find(lineInfo[0]);

        if (iter != endmap.end()) //map have this key
        {
            iter->second.SetGetEnd(ConvertStringToBoolen(lineInfo[1]));
        }

        #pragma endregion
    }

    haveEnd.close();
    #pragma endregion
}

void CEndManager::OnCycle()
{
    Play();
}

bool CEndManager::IsPassEnd(string endName)
{
    if (!endmap.count(endName))
        return false;

    return endmap[endName].IsGetEnd();
}

vector<string> CEndManager::GetAllEndName()
{
    vector<string> endName;

    for (map<string, CEnd>::iterator iter = endmap.begin(); iter != endmap.end(); iter++)
    {
        endName.push_back(iter->first);
    }

    return endName;
}

void CEndManager::Start(string endName)
{
    nowEnd = &endmap[endName];
    isEnding = true;
    endBmp.SetBmp(nowEnd->GetBmpPath(0));
    step = 0;
    isFadeIn = true;
    isFadeOut = false;
}

void CEndManager::Play()
{
    if (nowEnd->GetBmpPath(step) != END_EOF) //取得的路徑位置還還不是空路徑
    {
        if (isFadeIn)
        {
            endBmp.FadeIn();

            if (endBmp.GetAlpha() >= 255)
            {
                if (!CDialogManager::Instance()->GetDialogState())
                {
                    if (nowEnd->GetTxt(step) != END_EOF && isOpenDialog == false) //正常結局+對話 (中的對話)
                    {
```

```
                CDialogManager::Instance()->Start(nowEnd->GetTxt(step));
                isOpenDialog = true;
            }
            else if (nowEnd->GetTxt(step) == END_EOF && isOpenDialog == false) //僅有結局，沒有對話 (重設
計時器)
            {
                time_remaining.ResetTime(1.5);
                isOpenDialog = true; //! 開啟 true 後 下一個 thread 會進入到下方的 else 並計時
            }
            else
            {
                time_remaining.CountDown(); //對話結束後 持續一段時間 才切換下一張

                if (time_remaining.IsTimeOut()) //1.5s 時間到 換狀態
                {
                    isFadeIn = false;
                    isFadeOut = true;
                    isOpenDialog = false;
                    time_remaining.ResetTime(1.0);
                }
            }
        }
    }
    else if (isFadeOut)
    {
        endBmp.FadeOut();

        if (endBmp.GetAlpha() <= 0) //圖片完全透明 換狀態 換圖片
        {
            step++;

            if (nowEnd->GetBmpPath(step) != END_EOF) //更改 step 在做一次防呆
            {
                endBmp.SetBmp(nowEnd->GetBmpPath(step));
                isFadeIn = true;
                isFadeOut = false;
            }
        }
    }

    endBmp.DrawImage();
  }
  else
  {
    nowEnd->SetGetEnd();
    Stop();
  }
}

void CEndManager::Stop()
{
  isEnding = false;
}

#pragma endregion

#pragma region - Button Manager -
CButtonManager::CButtonManager()
{
  Initialize();
}

CButtonManager::~CButtonManager()
{
  Clear();
}

void CButtonManager::Load()
```

```
{
   for (map<string, CButton*>::iterator btniter = buttons.begin(); btniter != buttons.end(); btniter++)
   {
      (btniter->second)->LoadBitmap();
   }
}

void CButtonManager::Initialize()
{
   isValid = false;

   for (map<string, CButton*>::iterator btniter = buttons.begin(); btniter != buttons.end(); btniter++)
   {
      (btniter->second)->Initialize();
   }
}

void CButtonManager::Clear()
{
   for (map<string, CButton*>::iterator btniter = buttons.begin(); btniter != buttons.end(); btniter++)
   {
      delete btniter->second;
      btniter->second = NULL;
   }

   buttons.clear();
}

void CButtonManager::AddButton(CButton* _button)
{
   if (_button->name != "")
      buttons[_button->name] = _button;
}

void CButtonManager::CreateButton(BitmapPath _loadpath, CPoint point, bool _state, bool _needCollision)
{
   AddButton(new CButton(_loadpath, point, _state, _needCollision));
}

void CButtonManager::CreateButton(CButton* button)
{
   AddButton(button);
}

void CButtonManager::ClickButton(string _btnName)
{
   (buttons[_btnName])->ClickButton();
}

bool CButtonManager::IsCollisionMouse(string _btnName)
{
   return (buttons[_btnName])->IsCollisionMouse(mouse);
}

bool CButtonManager::GetState(string _btnName)
{
   return (buttons[_btnName])->state;
}

string CButtonManager::GetCollisionButtonName()
{
   for (map<string, CButton*>::iterator btniter = buttons.begin(); btniter != buttons.end(); btniter++)
   {
      if (IsPointInRect(mouse, btniter->second->GetAnimate()->GetRect())) // 有碰撞
      {
         return btniter->first;
      }
   }
```

```
    return "NoButtonClick";
}

void CButtonManager::UpdateState(CPoint _m)
{
    mouse = _m;

    for (map<string, CButton*>::iterator btniter = buttons.begin(); btniter != buttons.end(); btniter++)
    {
        (btniter->second)->CollisonMouse(_m);
    }
}

void CButtonManager::ShowButton()
{
    for (map<string, CButton*>::iterator btniter = buttons.begin(); btniter != buttons.end(); btniter++)
    {
        (btniter->second)->OnShow();
    }
}

void CButtonManager::OnCycle()
{
    for (map<string, CButton*>::iterator btniter = buttons.begin(); btniter != buttons.end(); btniter++)
    {
        (btniter->second)->OnMove();
        (btniter->second)->SetValid(isValid);
    }
}
#pragma endregion

#pragma region - CMapEditer -
CMapEditer::CMapEditer()
{
    arrow[0] = Arrow("up");
    arrow[1] = Arrow("down");
    arrow[2] = Arrow("left");
    arrow[3] = Arrow("right");
}

CMapEditer::~CMapEditer()
{
    mapObj.clear();

    if (!blockMap.empty())
    {
        StoreMapInformaion();
    }
}

void CMapEditer::Initialize()
{
    nowMap = getFolerFileNumber("RES\\Map\\Information\\");
    #pragma region - blockMap - ReLoad -
    blockMap.clear();

    for (int i = 0; i < nowMap; i++)
    {
        blockMap.push_back(CBlockMap(i));
        blockMap[i].LoadImg();
    }

    selectNowMap = 0;
    selectMapMode = "none";
    #pragma endregion
    #pragma region - Load arrow -

    for (int i = 0; i < 4; i++)
    {
```

```
        arrow[i].Initialize();
    }

    SetArrowCanShow();
    #pragma endregion
    LoadReloadMapInformation();
    NewMapInit();
}

void CMapEditer::NewMapInit()
{
    mapObj.clear(); //ImageInfo - block
    bkmap.Initialize();
    printNowMap = &nowMap;
    isPrintNowMap = false;
    saveTxtName = EDITER_PRESET_SAVETXTNAME;
    isMouseDown = false;
    isSaved = false;
    haveBG = false;
    isMapRight = isMapLeft = false;
    selectObj = NULL;
    cameraX = 0;
}

void CMapEditer::AddImage(vector<string> path)
{
    //類型  load (load map)
    //類型(背景 / block), 圖片路徑
    if (path[0] == "background")
    {
        haveBG = true;
        ImageInfo tempcv;
        tempcv = ImageInfo(path[1], "background");
        background = tempcv;
        background.SetXY(0, 0, 0);
    }
    else if (path[0] == "block")
    {
        ImageInfo tempObj = ImageInfo(path[1], "block");
        tempObj.SetXY(cameraX, tempObj.y);
        mapObj.push_back(tempObj);
    }
    else if (path[0] == "load")
    {
        NewMapInit();
        LoadBlockMap(path[1]);
        saveTxtName = path[1];
        background.SetXY(0, 0, 0);
    }
    else if (path[0] == "upDoor" || path[0] == "downDoor")
    {
        ImageInfo tempObj = ImageInfo(path[1], path[0]);
        tempObj.SetXY(cameraX, tempObj.y);
        mapObj.push_back(tempObj);
    }
}

void CMapEditer::SetMapMoveDir(string dir)
{
    if (dir == "right")
    {
        isMapRight = true;
    }
    else if (dir == "left")
    {
        isMapLeft = true;
    }
    else if (dir == "cleft")
    {
```

```
            isMapLeft = false;
        }
        else if (dir == "cright")
        {
            isMapRight = false;
        }
    }

    void CMapEditer::OnSave()
    {
        if (IsInSelectMapMode())
        {
            if (selectMapMode == "up")
            {
                bkmap.upMap = selectNowMap;
                blockMap[selectNowMap].downMap = nowMap;
            }
            else if (selectMapMode == "down")
            {
                bkmap.downMap = selectNowMap;
                blockMap[selectNowMap].upMap = nowMap;
            }
            else if (selectMapMode == "left")
            {
                bkmap.leftMap = selectNowMap;
                blockMap[selectNowMap].rightMap = nowMap;
            }
            else if (selectMapMode == "right")
            {
                bkmap.rightMap = selectNowMap;
                blockMap[selectNowMap].leftMap = nowMap;
            }

            reloadMap[selectNowMap] = true;
            SetSelectMapMode("none");
        }
        else
        {
            bkmap.nowMap = nowMap;
            CreateObjToBkmap();

            if (!isSaved) //沒有儲存過地圖
            {
                CString saveDir = ".txt";
                string tempSaveName = "map" + std::to_string(nowMap) + ".txt"; //緩衝
                CString saveName = tempSaveName.c_str();
                CString saveExt = "txt (*.txt)|*.txt||";
                CFileDialog saveDlg(false, saveDir, saveName, OFN_HIDEREADONLY | OFN_OVERWRITEPROMPT,
saveExt);
                int result = saveDlg.DoModal();

                if (result == IDOK)
                {
                    saveTxtName = saveDlg.GetFileName();
                    isSaved = true;
                    nowMapNumber++;
                    bkmap.CreateInformation(saveTxtName);
                }

                if (result == IDCANCEL)
                {
                    return;
                }
            }

            if (nowMap < (int)reloadMap.size())
            {
                reloadMap[nowMap] = true;
            }
```

```
        #pragma region - bkmap write to blockmap -

        if (nowMap < (int)blockMap.size()) //修改一張地圖
        {
            blockMap[nowMap] = bkmap;
        }
        else //要新增一張地圖
        {
            blockMap.push_back(bkmap);
        }

        #pragma endregion
    }
}

void CMapEditer::CreateObjToBkmap()
{
    bkmap.block.clear();
    bkmap.door.clear();
    bkmap.loadPath = background.path;
    #pragma region - Load Map Simple Object -

    for (vector<ImageInfo>::iterator mbiter = mapObj.begin(); mbiter != mapObj.end(); mbiter++)
    {
        if (mbiter->id == "block")
        {
            bkmap.block.push_back(CBlock(mbiter->path, mbiter->x, mbiter->y));
        }
        else if (mbiter->id == "upDoor")
        {
            bkmap.door.push_back(CDoor(mbiter->path, mbiter->x, mbiter->y, mbiter->id, bkmap.upMap));
        }
        else if (mbiter->id == "downDoor")
        {
            bkmap.door.push_back(CDoor(mbiter->path, mbiter->x, mbiter->y, mbiter->id, bkmap.downMap));
        }
    }

    #pragma endregion
}

void CMapEditer::LoadBlockMap(string mapName)
{
    sscanf(mapName.c_str(), "map%d.txt", &nowMap);
    bkmap = blockMap[nowMap];
    isSaved = true;
    background = ImageInfo(bkmap.loadPath, "background"); //load background
    haveBG = true;
    #pragma region - Load block -

    for (vector<CBlock>::iterator mbiter = bkmap.block.begin(); mbiter != bkmap.block.end(); mbiter++)
    {
        ImageInfo tempk = ImageInfo(mbiter->path, "block");
        tempk.SetXY(mbiter->x, mbiter->y, cameraX);
        mapObj.push_back(tempk);
    }

    #pragma endregion
    #pragma region - Load Door -

    for (vector<CDoor>::iterator mbiter = bkmap.door.begin(); mbiter != bkmap.door.end(); mbiter++)
    {
        ImageInfo tempk = ImageInfo(mbiter->path, mbiter->GetType());
        tempk.SetXY(mbiter->x, mbiter->y, cameraX);
        mapObj.push_back(tempk);
    }

    #pragma endregion
```

```
}

void CMapEditer::StoreMapInformaion()
{
   for (unsigned int i = 0; i < blockMap.size(); i++)
   {
      blockMap[i].CreateInformation();
   }

   blockMap.clear();
}

void CMapEditer::OnMove()
{
   if (isMapRight)
   {
      if (cameraX >= 0 && (cameraX + SIZE_X) < background.bmp.Width())
      {
         cameraX += MOVE_DISTANCE;
      }
      else
      {
         cameraX = background.bmp.Width() - SIZE_X;
      }
   }

   if (isMapLeft)
   {
      if (cameraX > 0 && cameraX + SIZE_X <= background.bmp.Width())
      {
         cameraX -= MOVE_DISTANCE;
      }
      else
      {
         cameraX = 0;
      }
   }

   if (haveBG)
   {
      background.bmp.SetTopLeft(background.x - cameraX, background.y);
   }

   for (vector<ImageInfo>::iterator mbiter = mapObj.begin(); mbiter != mapObj.end(); mbiter++)
   {
      mbiter->bmp.SetTopLeft(mbiter->x - cameraX, mbiter->y);
   }
}

void CMapEditer::OnShow()
{
   if (!IsInSelectMapMode()) //不在選擇地圖的模式中
   {
      if (haveBG)
      {
         background.bmp.ShowBitmap();
      }

      for (vector<ImageInfo>::iterator mbiter = mapObj.begin(); mbiter != mapObj.end(); mbiter++)
      {
         mbiter->bmp.ShowBitmap();
      }
   }
   else //選擇地圖中
   {
      if (selectNowMap < (int)blockMap.size())
      {
         blockMap[selectNowMap].backgroundBitmap.ShowBitmap();
      }
```

```
    for (int i = 0; i < 4; i++)
    {
      if (arrow[i].CanShow())
      {
        arrow[i].GetBmp()->ShowBitmap();
      }
    }
  }
}

void CMapEditer::SelectBlock(CPoint mouse)
{
  for (vector<ImageInfo>::reverse_iterator mbiter = mapObj.rbegin(); mbiter != mapObj.rend(); mbiter++)
  {
    if (IsPointInRect(mouse, mbiter->bmp.GetRect()))
    {
      selectObj = &(*mbiter);
      SetDPoint_MouseToTopLeft(mouse);
      return;
    }
  }

  selectObj = NULL;
  SetDPoint_MouseToTopLeft();
}

void CMapEditer::DragBlock(CPoint mouse)
{
  if (selectObj == NULL)
    return;

  selectObj->SetXY(mouse.x - dpoint_mouseToTopleft.x, mouse.y - dpoint_mouseToTopleft.y, cameraX);
}

void CMapEditer::DeleteBlock()
{
  if (selectObj != NULL)
  {
    for (vector<ImageInfo>::iterator mbiter = mapObj.begin(); mbiter != mapObj.end(); mbiter++)
    {
      if (selectObj == &(*mbiter))
      {
        mbiter = mapObj.erase(mbiter);
        selectObj = NULL;
        return;
      }
    }
  }
}

string CMapEditer::GetNowMap()
{
  return "now map: " + std::to_string(*printNowMap);
}

void CMapEditer::CreateReloadMapInformation()
{
  fstream reloadData;
  reloadData.open("RES\\Map\\ReloadMapInformation.txt", ios::out);
  reloadData << std::to_string(nowMapNumber) + "\n"; //write 目前有多少張地圖勿

  for (unsigned int i = 0; i < reloadMap.size(); i++)
  {
    if (reloadMap[i])
    {
      reloadData << std::to_string(i) + "\n"; //write  那些地圖要 reload
    }
  }
```

```
    reloadData.close();
}

void CMapEditer::LoadReloadMapInformation()
{
    fstream reloadData;
    reloadData.open("RES\\Map\\ReloadMapInformation.txt");

    if (!reloadData.is_open()) //開啟失敗
    {
        nowMapNumber = nowMap;
        reloadMap.clear();
        reloadMap.resize(nowMapNumber, false);
    }

    reloadData.close();
}

void CMapEditer::SetSelectMapZero()
{
    selectNowMap = -1;
    OnSave();
}

void CMapEditer::SetSelectMapMode(string _mode)
{
    if (_mode == "up" || _mode == "down" || _mode == "left" || _mode == "right")
        printNowMap = &selectNowMap;
    else
    {
        selectNowMap = 0;
        printNowMap = &nowMap;
    }

    selectMapMode = _mode;
}

void CMapEditer::ClickArrow(CPoint _mouse)
{
    for (int i = 0; i < 4; i++)
    {
        if (arrow[i].CanShow() && IsPointInRect(_mouse, arrow[i].GetBmp()->GetRect()))
        {
            SwitchMap(arrow[i].GetDir());
        }
    }
}

void CMapEditer::SwitchMap(string _dir)
{
    if (_dir == "up")
    {
        GotoMap(blockMap[selectNowMap].upMap);
    }
    else if (_dir == "down")
    {
        GotoMap(blockMap[selectNowMap].downMap);
    }
    else if (_dir == "left")
    {
        GotoMap(blockMap[selectNowMap].leftMap);
    }
    else if (_dir == "right")
    {
        GotoMap(blockMap[selectNowMap].rightMap);
    }
}
```

```
    void CMapEditer::GotoMap(int mapIndex)
    {
        if (mapIndex != -1)
        {
            selectNowMap = mapIndex;
            SetArrowCanShow();
        }
    }

    void CMapEditer::SetArrowCanShow()
    {
        arrow[0].SetCanShow(blockMap[selectNowMap].upMap >= 0 && blockMap[selectNowMap].upMap <
(int)blockMap.size());
        arrow[1].SetCanShow(blockMap[selectNowMap].downMap >= 0 && blockMap[selectNowMap].downMap <
(int)blockMap.size());
        arrow[2].SetCanShow(blockMap[selectNowMap].leftMap >= 0 && blockMap[selectNowMap].leftMap <
(int)blockMap.size());
        arrow[3].SetCanShow(blockMap[selectNowMap].rightMap >= 0 && blockMap[selectNowMap].rightMap <
(int)blockMap.size());
    }
    #pragma endregion

    #pragma region - UIManager -
    UIManager::UIManager()
    {
    }

    void UIManager::Load()
    {
        roleStatus.Load();
        bossStatus.Load();
        uiTime.LoadBitmap(".\\RES\\Number\\cookiezi", "default");
        uiScore.LoadBitmap(".\\RES\\Number\\cookiezi2", "default");
    }

    void UIManager::Initialize(CRole* _role, CBossManager* _bManager)
    {
        role = _role;
        bManager = _bManager;
        roleStatus.Initialize(CPoint(0, 0), role->GetHp(), role->GetEq());
        bossStatus.Initialize(CPoint(0, 0));
        uiTime.Initialize(CPoint(250, 0), GAME_TIME, 2);
        uiScore.Initialize(CPoint(150, 60), 0, 3);
        CLayerManager::Instance()->AddObject(&uiTime, INTERFACE_LAYER);
        CLayerManager::Instance()->AddObject(&uiScore, INTERFACE_LAYER);
    }

    void UIManager::OnCycle(int _time)
    {
        uiTime.SetInteger(_time);
        roleStatus.OnCycle(role->GetHp(), role->GetEq());
        bossStatus.OnCycle(bManager);
        uiTime.SetInteger(_time);
        uiScore.SetInteger(role->GetScore());
    }
    #pragma endregion
}
```

15. Refactor.h

```
//////////////////////////
//定義常數
//////////////////////////
#pragma once
#define MAX_LAYER_NUMBER 11
#define MOVE_DISTANCE 8
#define MAX_MAP_NUMBER 99
#define NOW_MAP 0
#define KEY_SPACE 32
```

```
#define KEY_LEFT 37 // keyboard 左箭頭
#define KEY_UP 38 // keyboard 上箭頭
#define KEY_RIGHT 39 // keyboard 右箭頭
#define KEY_DOWN 40 // keyboard 下箭頭
#define KEY_W 87
#define KEY_A 65
#define KEY_S 83
#define KEY_D 68
#define KEY_Q 81
#define KEY_Z 90
#define KEY_C 67
#define KEY_B 66
#pragma region - Layer
#define INTERFACE_LAYER 9
#define BULLET_LAYER 6
#define BOSS_LAYER 6
#define NPC_LAYER 4
#define BLOCK_LAYER 5
#pragma endregion

#pragma region - dialog - margin -
#define MARGIN_DIALOG_AVATAR 16
// 16 + avatar.width()
#define MARGIN_DIALOG_TEXT_X 16
#define MARGIN_DIALOG_TEXT_Y 8
#pragma endregion

#pragma region - dialog - setting -
#define DIALOG_TEXT_SIZE 16
#define DIALOG_TEXT_COLOR RGB(254, 254, 254)
//MAX TEXT - ENGLISH SIZE (1 CHINESE SIZE = 2 ENGLISH SIZE)
#define DIALOG_MAX_TEXT 30
#define DIALOG_BACKGROUND_COLOR RGB(66, 66, 66)
#pragma endregion

#pragma region - dialog - avatar name -
#define DIALOG_AVATAR_NAME_ROLE "role"
#define DIALOG_AVATAR_NAME_XINGTING "xingting"
#define DIALOG_AVATAR_NAME_LOCK "lock"
#define DIALOG_AVATAR_NAME_QUESTION "question"
#define DIALOG_AVATAR_NAME_STUDENTB "studendB"
#define DIALOG_AVATAR_NAME_STUDENTG "studentG"
#define DIALOG_AVATAR_NAME_FAQAISEED "faqaiSeed"
#pragma endregion

#pragma region - dialog - txt data -
#define DIALOG_DATA_VSXingting1 "roleVsXingting1"
#define DIALOG_DATA_VSXingting2 "roleVsXingting2"
#define DIALOG_DATA_VSXingting3 "roleVsXingting3"
#define Tips "InitTip"
#define FROG "frog"
#define MyVoiceIsDead "MyVoiceIsDead"
#define DIALOG_DATA_STGHAVEBREAKFAST "StudentGHaveBreakfast"
#define DIALOG_DATA_MEETSTB "StudentSmoke"
#define DIALOG_DATA_STGSUBMIT "StudentSubmit"
#define DIALOG_DATA_FAQAI "Faqai"
#pragma endregion

#pragma region - dialog - add a text on cycle -
#define DIALOG_ADDTEXT_TIME 0.1
#pragma endregion

#pragma region - boss -
#define BOSS_DEPRATMENT "boss"
#define BOSS_MAP_XINGTING 5
#define BOSS_MAP_FACAISEED 11
#pragma endregion

#pragma region - boss name -
```

```
#define BOSS_XINGTING "xingting"
#define BOSS_FACAISEED "FacaiSeed"
#pragma endregion

#pragma region - block -
#define BLOCK_DEPRATMENT "block"
#pragma endregion

#pragma region - End - Name -
#define END_NAME_WINXINGTING "WinXingting"
#define END_NAME_LOSEXINGTING "LoseXingting"
#define END_NAME_SALTEDFISH "SaltedFish"
#define END_NAME_WIN_FACAISEED "WinFacaiSeed"
#define END_NAME_LOSE_FACAISEED "LoseFacaiSeed"
#pragma endregion

#define END_EOF "nonenone"
#define EDITER_PRESET_SAVETXTNAME "mapTest.txt"

#define GAME_TIME 60 //限制時間
#define DEBUG_MODE false
```