台北科技大學 資訊工程系

物件導向程式設計
書面報告

NTUT CSIE OOP

組　　別：第 5 組

題　　目：Typing Typing

組　　員：104590025 余鎧企

　　104590029 黃省喬

指導老師：陳偉凱

目錄

一、 簡介

1. 動機：

剛學習電腦的時候，打字必為入門基礎。尤其在資工系這塊領域，若打字速度跟不上腦袋思考，必定會錯過許多新穎的想法。而我們在這片學海無涯的道路上，開始了我們的根基。自幼時開始學習英打，接觸過許多網絡上的軟體。有文章式、逐行式、單字類，但因為內容太過於單調與呆版，並不受我們的喜愛與大眾的需求。在一個偶然的機遇下，我們找到了「ZType」！儘管同樣是一個英打練習的網頁遊戲，與眾不同的是，它以「遊戲」的方式來進行，既有傳統的計算速度與正確率，又有新穎的華麗特效，成功吸引了初學者的目光。為了攻略這個遊戲、我們夜以繼日的練習。而後，我們英打獲得突飛猛進的突破。而當我們修完上學期的”物件導向程式設計”後，便充分的利用所學，想要在這學期實現出這個令我們曾經感動的遊戲。藉由我們的實作，帶給其他想學打字的人一條有效的途徑，並提供給玩家前所未有聽覺及視覺的雙重響宴。

2. 分工：

余鎧企：

程式碼撰寫：檔案(File)讀寫處理、紀錄(Record)相關保存、音效(Audio)素材蒐集與處理、魔王(Boss)構思。

黃省喬：

程式碼撰寫：主角(Me)設計、怪物(Enemy)相關技術、特效處理、選單(List)頁面撰寫。

美術介面：選單頁面設計、主角皮膚(Skin)、關卡切換特效、遊戲特效設計。

二、 遊戲介紹

1. 遊戲說明:敵人們正從天而降！趕緊鍵入怪物身上的單字來消滅他們！

（1） 玩法：

在本遊戲中有各種不同外貌的敵人(Enemy)，但是他們都有一個共同點，那就是他們的身上都帶著一組英文單字。玩家必須在敵人接近之前，輸入其身上的單字，才能成功的把的敵人消滅。

（2） 規則：

當玩家鎖定攻擊了其中一隻怪物，正輸入到一半的時候，必須繼續將其單字輸入完畢，改敵人被消滅之後，才可以再鎖定其他敵人攻擊。

（3） 敵人：

本遊戲除了普通的敵人外，另外還有多種強大的 Boss，除了字數較長外，還各自擁有不同的技能。

（4） 普通敵人(Enemy) ：

普通的敵人，會以正常速度從天而降。

（5） Boss A：

技能：在消滅此 Boss 之前，他會每間隔一段時間，召喚出一隻普通敵人。

（6） Boss B：

技能：在消滅此 Boss 之前，他會每間隔一段時間，召喚多隻字數為 1 的敵人，以扇狀的方式朝遠方發射。

（7） 特殊功能：

當身邊有太多敵人即將接近，而且來不及將他們消滅的時候，玩家在每場遊戲，有三次的機會可以使用技能。按下"Enter"釋放出電磁脈衝(EMP)，快速的將身邊的敵人消滅。

（8） 密技：

作弊碼(CheatCode)，在”關於”頁面輸入：

104590025：開啟 Debug 功能

104590029：解鎖全角色

2. 遊戲圖形

我們遊戲圖形是以 8 BITS 復古風作為設計風格，它最大的特色就是在圖形的邊緣，看起來會有明顯的鋸齒。為了達成像素畫(Pixel Art) 的美術風格，大部分的圖形我們都是利用最基本最傳統的“小畫家”來繪製。和一般的繪圖軟體相比，我認為小畫家非常適合來製作點陣圖，因為他可以很精確的

在每一格中填色，並且可以很輕易的輸出 Bmp 格式的圖片，正好符合了我們的需求。

（1） 敵人：

普通敵人： 

1 字小怪： 

（2） BOSS：

（3） 角色：

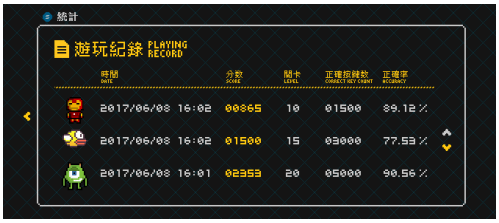
（4） 選單及界面：

- 開頭選單：

- 遊戲說明頁面：

- 角色選擇頁面：

- 統計頁面：

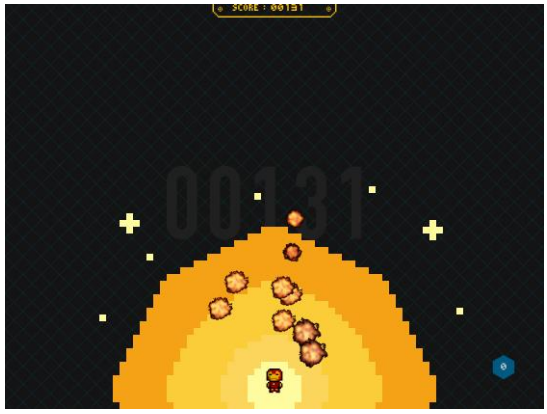


- 關於頁面 / 背景音樂(BGM)及音效(SE)開關 / 清除遊玩紀律：

- 關卡切換的動畫/ Gameover 結算畫面：

- 遊戲暫停：

- 遊戲畫面：



3. **遊戲音效：**

（1） 遊戲背景音樂：

取自於 YouTube 音樂庫中的 The_Coming_Storm，營造出緊張刺激的感覺，提升遊玩上的好感度與刺激性。

（2） 正確按鍵音效：

輸入正確的按鍵，會發出酷似手槍射擊的聲音，讓使用者在遊玩上獲得激勵。

（3） 錯誤按鍵音效：

若輸入錯誤，則會發出「喀喀喀」的音效，提醒使用者請勿盲目地亂敲鍵盤。

（4） EMP(電磁脈衝)音效：

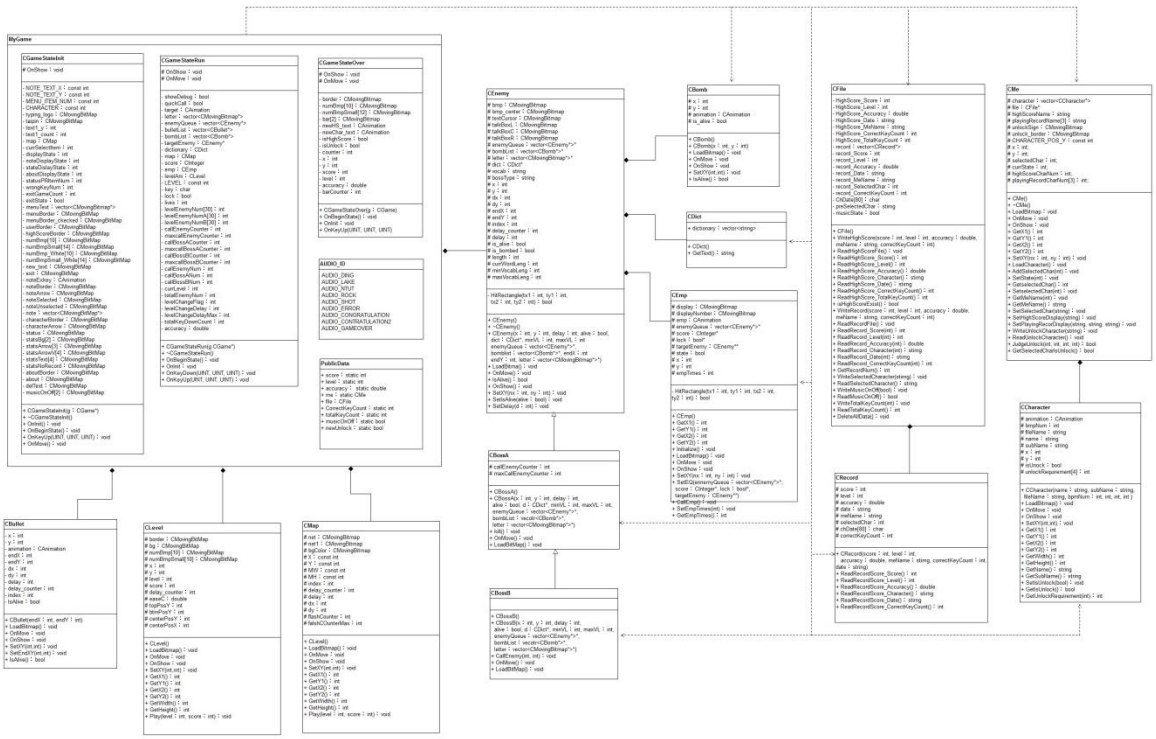
利用「爆炸」的音效，令使用者在狗急跳牆之際得以獲得一絲喘息空間。

（5） GameOver 音效：

除了提醒使用者遊戲結束，利用詼諧逗趣的音樂，令使用者想再重新來過，提升自我實力！

三、 程式設計

1. 程式架構：



2. 程式類別：

類別名稱	.h 檔行數	.cpp 檔行數	說明
CEnemy	65	205	普通敵人，會朝特定方向飛行。
CBossA	15	92	魔王 A，每隔一段時間會召喚一隻 Enemy。
CBossB	12	104	魔王 B，每隔一段時間會以扇狀的方式召喚多隻字數為 1 的 Enemy。
CEmp	39	139	電磁波，使用後會依照動畫清除範圍內的 Enemy
CBullet	25	71	子彈，從主角的位置不偏不倚地射向敵人。
CBomb	23	55	爆炸效果，當敵人死亡時便會顯示此效果。
CMe	52	345	顯示主角的圖樣，在三個遊戲狀態中皆會以不同的形式出現。
CCharacter	31	94	記錄主角的屬性：外觀與解鎖條件。
CFile	67	262	記錄遊戲結果，可以讀寫 TXT 檔，用以儲存最高記錄、每場遊玩記錄及選擇的角色等。
CRecord	33	53	顯示每場遊玩記錄時，用來儲存每筆資料裡面的分數、關卡、按鍵數、正確率及角色。
CMap	24	58	在遊戲畫面顯示持續移動的網狀背景。
CDict	14	46	字典檔，用於讀取 txt 文字檔中的單字，並儲存於 vector。
CLevel	27	115	遊戲內中的換關動畫。
myGame	194	1261	遊戲程式碼本身，概括分為：遊戲開始、遊戲進行中、遊戲結束下去著墨、延伸
總行數	618	2900	

3. 程式技術：

（1） 亂樹種子(Rand) ：

我們利用系統時間產生亂數，令敵人落下時的位置具有隨機性。考量到美術圖外型的關係，所以我們有特別設定敵人生成位置的 X 軸的區間，以免有超出出畫面外的狀況發生。

（2） 三角函數計算：

在敵人 BossB 的設計中，我們設定它每隔一段時間便會以扇狀的方式召喚一群小怪，而在扇狀的計算中， 我們利用了角度與徑度的換算、二維座標定位採取極座標之運算，利用迴圈與變數，讓就算有再多的小怪，也可以精確地計算出位置， 考慮視覺的舒適度與畫面的流暢度，我們最後將小怪定為 7 隻。

（3） 繼承(Inheriance) ：

由於落下的敵人不只是一般的普通小怪，還有難度提升的 BossA 與較難應付的 BossB。在設計的過程中， 我們將 BossA 繼承原有 Enemy，並新增有別與原有 Enemy 的美術圖與新的時間差，再利用下一項提及的「多型」設計新功能函數。 而 BossB 更繼承了 BossA，設計出了極具巧思的特有功能。

（4） 多型(Polymophism) ：

在多型方面，我們利用了虛擬函數(Vitual Function)。我們在普通小怪、BossA、BossB 上做了明確的區分，但因為是繼承關係， 所以在載入圖片、移動(move)函數與死亡設定(kill)函數上運用了虛擬函數做出獨特性。

（5） Vector、迭代(Iterator) ：

我們採用 Vector 的方式來存放場上現有的敵人，而非利用 Array 來儲存，會這麼做的原因有以下考量：使用 Vector 可以快速的在集合尾端插入資料，而且也支援隨機存取，方便我們可以將已死亡的敵人，從

陣列中移除。同時也可以利用.size()函數，很方便的取得當前場上敵人的總數，用來判斷是否可以接到下一關，利用 Iterator 和迴圈可以從頭到尾尋訪所有的敵人，鎖定特定條件的敵人。

（6） 遊玩記錄之保存：

我們利用了大一所學習的檔案讀寫功能，實作出歷史紀錄的保存，同時我們也讀取出最高分的紀錄，來提供玩家一個超越的指標，同時達到振奮效果。除此之外我們也未忽略使用者的遊玩體驗，在小細節上，遊戲也會記錄下玩家上次遊玩所選擇的角色及音效的開關狀態，下次開啟遊戲時，即可保留玩家的使用習性。

（7） 作弊(Cheat)、除錯(Debug)功能：

在「關於」頁面中我們暗藏作弊碼，只要輸入正確即可快速解鎖全角色與開啟 Debug 的功能。Debug 功能中，我們利用快捷鍵即可開啟顯示怪物列表及狀態，與快速清除怪物。在遊戲狀態的操作上，我們實作出自由進出 State，可以加快我們遊戲處理速度。

四、 結語

1. 問題及解決方法：

（１）在利用 vector.size()來跑 For 迴圈時，出現"將警告視為錯誤處理 - 沒有產生 'object' 檔案"之錯誤。

以 vector.size()，它回傳的值是 unsigned int，而平時撰寫程式時，通常習慣直接使用 int 來宣告 i，然而此 framework 對於形態的

規則比較嚴格，因此他將 int 和 unsigned int 視為不同的東西，因此只需要將變數 i 設定成 unsigned int 即可排除此錯誤。

（2）利用 CDC 螢幕字形來顯示怪物身上的單字，當怪物數量變多時，遊戲會嚴重卡頓。

原本遇到這個問題時，還不知道為何遊戲執行到後期，會越來越卡頓。後來看到了老師的 Framework 註解中提到說，盡量不要以此方式顯示文字，而是以 CMovingBitmap 來取而代之比較好。於是我們該採，將 26 個英文單字都製作成一個 Bmp 檔，並且研究比對各種不同字體所呈現出來的效果，於是再次執行後果然效率得到大大的改善，執行更為流暢。然而使用 Bitmap 來顯示文字有個缺點，就是文字的顯示位置必須謹慎拿捏，因為我們是靠迴圈來一個個貼上字母，經過反復的測試才抓到最佳的顯示位置。

（3）Float 與除法運算時，計算結果總不正確。

經過上網查詢資料後，發現 float 運算時有小陷阱，這個陷阱以前我們未曾留意到，是在計算怪物飛行偏移量 dx,dy 時才首次發生的。例如計算 double x= 3/2;，此預期的結果 x 應該要等於 1.5，然而運算結果總等於 1。這是因為算式的 3 和 2 都被視為 int，因此 3/2 整體也被看做是一個 int。後來發現只要預先將數字或變數強制轉型成 double，就可以解決此 Bug。如上面例子所示，可以改寫成 double x= double(3)/double(2);。若是針對常數運算，也可直接在該整數的後方加上小數點，例如 double x= 3.0/2.0;，如此一來 x 也能得到正確的運算結果。

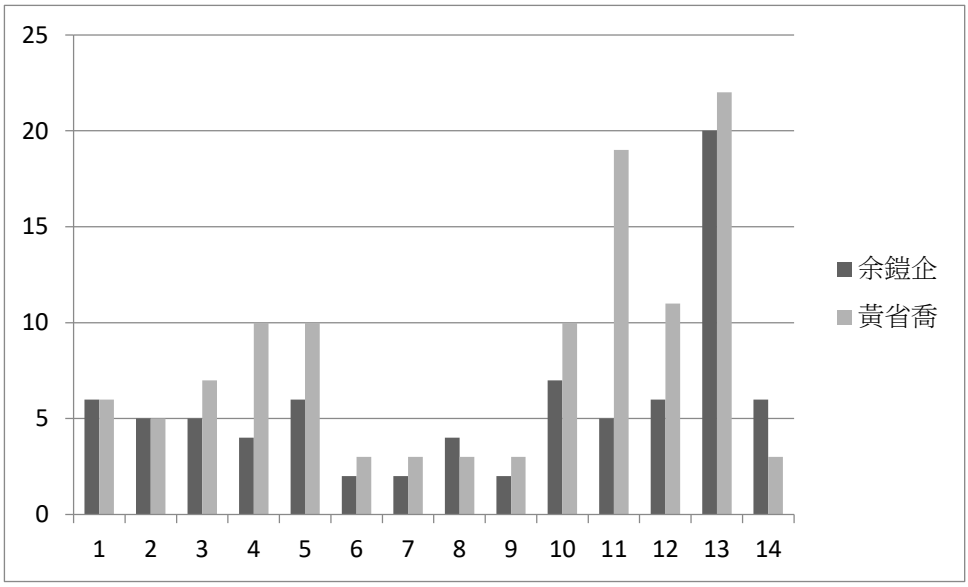
（４）在 KeyDown 事件中，明明使用者只按下一次按鈕，但程式實際卻會多執行數次。

這個 Bug 有點類似於設計電路時的彈跳，雖然表面上看起來只按下了一次，但程式的 Cycle 卻是不斷的在跑，這對於我們這種單字消除遊戲是非常困擾的，例如若出現連續字母的單字(BEE)，那在輸入到第 2 個字母‘E’的時候，會連帶的也把第三個字母‘E’也給消除了。於是我們使用一個字元變數 key，來儲存按下的按鍵。每當 OnKeyDown 被執行時就判斷當前所按下的按鍵是否和 key 相同，否的話則正常執行並將當前按鍵存入 key 中。而當 OnKeyUp 時再將 key 變數設定為 NULL，表示已放開了按鍵。這麼做之後，不僅解決了彈跳的問題，而且若使用者輸入速度過快，同時按下 2 個按鍵時，程式依然可以正確的判定。

（５）在三個不同 State 間，想要共用某些變數。

我們在 MyGame 裡面額外新增了一個 Class，專門儲存共用的變數，ex:檔案(File)的讀寫、角色(Me)的參數、分數(Score)、正確率(Accuracy)等等...，這樣成功的解決在三個 Class 之間變數的進出。

2. 時間表：



3. 貢獻比例：

黃省喬：60%

余鎧企：40%

4. 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	修改 Help->About	☑□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 檔	☑□已完成 □未完成
7	報告字型、點數、對齊、行 距、頁碼等格式正確	☑□已完成 □未完成
8	報告封面、側邊格式正確	☑□已完成 □未完成

5. 收獲：

104590029 黃省喬：

這是我第一次撰寫架構完整的遊戲，對於老師所提供的 Framework 也很陌生，一切都得憑藉著自己的摸索，才得以慢慢的撥雲見日。而在開發遊戲的過程中，我也得到相當多的收穫，學到很多不同的新知。

首先在以 C＋＋來撰寫遊戲時，和以往的程式作業- 僅僅透過 console 視窗來輸入指令進行運算不同。為了使得圖片在畫面中顯示，必須使用 OnShow 函數;為了使得圖片在畫面中移動,則要使用 OnMove 函數。這些對我來說都是非常嶄新的概念，這些函數皆各司其職且缺一不可，他們彼此間相輔相成，才造就起遊戲的穩固根基。

我了解到 OnMove 函數每秒執行約 30 次，所以我們便可以利用此特性來設計出各種動畫，如背景網狀的動畫、切換關卡時從上往下滑的提示文字、瞄準怪物時的準星動畫等，除此之外各種判斷的程式大多也在 OnMove 裡完成，例如搭配計數器(counter)，就可以設計每間隔一段時間便召喚一隻怪物。

由於我們設計的是“單字消除遊戲”，因此 OnKeyDown 的事件就顯得格外重要。在開發時我們就遇到了一個問題：在 OnKeyDown 時，會有類似以前設計電路時按下按鈕時彈跳的窘境。你以為你只按了一下按鍵，但程式的 cycle 卻是不斷在執行的，所以在按下按鍵到放開按鍵這中間，OnKeyDown 可能就被執行了超過 10 次。為了解決這困擾，我們新增了一個變數 key，用於儲存按下的按鍵。每當 OnKeyDown 被執行時就判斷當前所按下的按鍵是否和 key 相同，否的話則正常執行並將當前按鍵存入 key 中。而當 OnKeyUp 時再將 key 變數設定為 NULL，表示已放開了按鍵。

所幸有 Astyle 提供自動排版，讓我們在撰寫程式的過程當中省了不少心，而看到老師提供的 Framework，我才驚覺到原來程式碼和註解的對齊也是一門學問，僅有 Astyle 還不夠，若可以適時善用 TAB 及倒斜線換行來讓程式碼對齊，那麼對於日後閱讀程式碼可有相當大的助益。

為了精簡程式碼並使其更易讀，我也學習到在載入 Bitmap 時，可以善用迴圈及 sprintf 函數來讀取路徑。例如想要隨機顯示敵人皮膚的 Bitmap，我們可以預先將圖檔都編號(faceXX.bmp) ,程式中再搭配 sprintf 及 rand 函數隨機載入 Bitmap。我認為用這種方法，遠比將所有的路徑儲存於字串陣列中，還來的更有效率。

因為以往寫的程式執行的時間不會太長，通常輸入完沒幾行指令程式就結束了，基本上都是簡單的運算及文字處理，所以我一直都忽視了記憶體佔用多寡的重要性。而在設計較大型的遊戲時，妥善的記憶體分配之重要性更是不容小覷。為了減少記憶體的浪費，某些常用且經常重複出現的元素，我選擇將它們宣告在 mygame 的 protected 之中，再利用指標的方式傳入需要的 class 中。例如我們有一個字典 class 叫做 CDict，其用途是自 txt 檔讀取單字字串並存入 vector，並可以回傳隨機一個單字。最初我們是在每一隻敵人(CEnemy)裡都獨立宣告一個 CDict，但在中期時我們發覺，其實沒必要每 new 一隻敵人，同時帶著一份笨重的 CDict。因為 CDict 裡的 vector 儲存了上百筆的字串，隨著敵人的數量增加，想必會對於記憶體造成極大負荷。於是我們採用讓每個敵人都以指標的方式，共用同一個 CDict，結果不出所料，原本生成敵人時會有的卡頓感瞬間煙消雲散。

在 vector 的運用上，我學到要清空 vector 前，必須先將內部的資料都 delete 才行。以下這句金句更是狠狠地烙印我心：“new 了什麼就要記得 delete 什麼。

104590029 余鎧企：

從上學期的火車訂票系統中，學習到了程式架構，便以為自己懂了，直到這學期實作出這個 Typing Typing 的遊戲，才對整個程式有深刻得體悟。剛開始拿到老師示範的 Framework，先理解大方向的三個 GameState，再從 GameStateRun 開始著手了解。得知 BeginState 跟 Init 的差別。前者為每次都要進入的狀態，後者是遊戲一開始時便 Load 的資源與設定的初始值。了解到 KeyUp 與 KeyDown 的不同，KeyDown 為按下便一直執行，KeyUp 為按鍵起來才會執行，這方面卡住我們蠻多時間，主要反映在玩家玩遊戲時的體驗，為了整體的流暢度，我們便設定出一個變數，類似電路防彈跳的功能，再按下時儲存按下按鍵，放開時在清空儲存值，便成功了提高整體順暢度。

OnMove 為遊戲主要在移動的元素，我們從一開始從 Framework 中移動的物體開始延伸，一開始研究紅球落下，到後來加入了數學運算，成功控制了位移的角度，而後許多東西也伴隨著產生：小怪的誕生、射出的子彈、每隻 BOSS 的獨特性。而我們也實作出動畫的效果，利用許多張照片中微妙的差異，置入於 OnMove 函數中，使肉眼產生了動畫的效果，類似 EMP(電子脈衝)、主角中的微妙動畫、看似再移動的地圖背景，這也歸功於 Framework 中會變色的彈跳球(Bouncing Ball)。OnShow 的部分，在後期的程式上，出現了一個突破性的進展，若一直從外部引

用圖片，在最後會造成了程式當機。這是我們壓根都沒有想到的，而我們解決辦法是利用了老師在網站的教學，載入到 Resource File 的方法，解決了程式的負荷。

在音效的部分，我們利用了網路上的開放資源，Youtube 的開放音樂庫與開放合法下載的音效網站，方知有這麼多的資源供我們使用。而在 CAudio 這裡讓我學了一課：我們並不能無條件 Stop 音效，因為並不知道它是否正在播送，唯有確定它在開始狀態才可暫停。而突如其來的暫停會使程式當機！

而這遊戲特別的功能：歷史保存，我們將使用者的紀錄一筆一筆的儲存下來，用的是大一學過的 File 處理，在這 Class 設計上，我們參考了火車訂票系統中的 Class Diagram：TicketOffice 與 Ticket、Station 的關係，CRecord 中單純紀錄單筆資料，在 CFile 中以 Vector 方式儲存 CRecord，再利用 Vector 與 Iterator 的特性迅速的回傳所需的第 X 筆資料。這個用法在 CMe 與 CCharacter 也有使用到，利用了這層關係，使我們在程式架構上更加得清晰，不會在最後彙整的時候產生紊亂。

6. 心得、感想：

104590029 黃省喬：

從小到大我接觸了各式不同種類的遊戲，並且從那個時候就對電腦遊戲感到無比的著迷。對於一個好奇心旺盛的少年來說，電腦遊戲為平時枯燥乏味的求學生活帶來無限的溫暖曙光，探索在無垠無涯的虛擬世界裡，樂趣無窮，我想這就是所謂的青春吧。

當我得知物件導向實習課，是要以寫遊戲來呈現時，其實我是十分

興奮的。在經歷上學期 OOP 課程的洗禮後，我對於物件導向的概念已有初步的認識及瞭解，但若無法將課堂中得到的知識學以致用，那我認為學的再多都是枉然。而這堂 OOP 實習課，便給了我們這千載難逢的機會，讓我們可以將腦中無數創意想法付諸實際，得以活用程式設計的技巧並增加邏輯推理的能力。

而以製作遊戲來當做期末專題，對我來說也是相當耳目一新，畢竟玩過了無數大大小小的遊戲，現在卻要換我們來寫出一個遊戲，無疑是一項艱鉅的挑戰。但漸漸地，我發現我越來越沉浸於撰寫程式和 DEBUG 的過程當中，只要是閒暇之餘就會把程式給打開來寫，甚至到了有些廢寢忘食的地步，這真是奇妙的感覺。

無論是突然的靈光乍現解決了臭蟲，或是編輯素材時，一筆一劃刻畫出的怪物圖片，能憑藉著自己的力量，一步一個腳印的使遊戲壯大，由淺入深的內容變得豐富，看到最終完成的作品，那股成就感是不言而喻的。

對我來說這不僅僅是一門實習課，除了獲得學分之外，我認為在製作專題的過程中學習到的東西才最難能可貴。在遭遇困難時，懂得利用網路資源尋求解決方案，或是和同學請教互相討論，在如此潛移默化下，我在撰寫程式時的思路變得更加清晰有脈絡，在無形中學到的東西更遠遠超過原本我所期待。

104590025 余鎧企：

在國小的時候，參加過大大小小的打字比賽，中英打皆有，過程中藉助了很多市面上的打字練習軟體，所以對於打字練習並不陌生。有天

碰上了 ZType 這個練習網站，便深深的著迷，並不知道原來藉由華麗的遊戲與酷炫的特效，同樣也可達到練習效果，原以為遊戲，就是拿來玩，打發時間用的。

到了大學進入資工系，從 C 開始累積基礎，到二上的 C++開始對於程式有了初步的認知。尤其是強迫一定要新增 Class 的時期，起初覺得過於麻煩，一個小的程式為什麼要新增 Class 呢？直到了「火車訂票系統」，我認為它是對於我寫程式的一個轉捩點。一個 Class Diagram、說明程式所需與限制。我認為我終於像是在寫程式了。

進入下學期的 OOP，知道要從模仿一個遊戲開始時，便十分慌張與猶豫，怕一個輕易決定，都會超過我們目前的能力。最後與夥伴討論出實作出一個「打字練習」為主題的遊戲。起初對於 FrameWork 得進行相當無助，但縮小範圍，一步一腳印地慢慢看，很快便有了起步。我逐漸上手，先從簡單的移動到加入學過的數學式子，便開始步入主題至打字消除，其中經歷了不少的困難，像是記憶體的耗費、美術圖的應用、人性化的控制，都花了我們不少的時間。我們才知那看似簡單的功能，實際上卻非常的消耗腦力，愈是微小的地方，花的精神愈是甚多。

成功消除之後，便開始延伸至 BOSS 的特性，主角的應用。我們在 BOSS 功能與特性上花了多巧思，不然即失去了意義。在主角的設計上，我的夥伴省喬他發揮了精湛美術天賦，利用 8Bit 畫出了我出乎意料的圖形，也讓我對於美工有了更上一層的體悟。

到最後新增了音效與特效，發現好的程式，其中也包含了好的視覺與聽覺享受，就像是 ZType 帶給我的感動一樣。

在這過程中，我們不斷精進自己。培養解決問題的能力與團隊合作、

溝通的能力。比起上學期獨自寫一個程式，團隊合作更顯得可貴，畢竟市面上的程式，都不是一個人可獨自完成的。學會了在沒有老師的奧援下尋求網路得到解答，或是與同學共同討論。學會了清晰的規劃程式架構，不至於在之後的結合產生紊亂。學會了以使用者的角度做設計，深刻理解到程式不只是給自己使用的。最後感謝這堂課帶來給我無窮的受益。

7. 對於本課程的建議：

附錄

1. CBomb.h

```
#pragma once

namespace game_framework {
// CBomb：敵人死亡後會的爆炸動畫
class CBomb {
      public:
            CBomb();
            CBomb(int x, int y);
            void LoadBitmap();          // 載入圖形
            void OnMove();                    // 移動
            void OnShow();                    // 將圖形貼到畫面
            void SetXY(int, int);
            bool IsAlive();
      protected:
            int x, y;                         // 圖形座標
            CAnimation animation;             // 爆炸動畫
            bool is_alive;                    // 是否活著

};
}
```

2. CBomb.cpp

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

namespace game_framework {
// CBomb:  爆炸動畫

CBomb::CBomb() {
}

CBomb::CBomb(int x, int y) {
      animation.SetDelayCount(1);
      this->x = x;
      this->y = y;
      is_alive = true;
}

void CBomb::LoadBitmap() {
      for (int i = 0; i < 16; i++) {
            char str[30];
            sprintf(str, "Bitmaps/bomb/bomb%d.bmp", i + 1);
            animation.AddBitmap(str, RGB(255, 0, 255));
      }
}

void CBomb::OnMove() {
      if (!is_alive) return;

      animation.OnMove();             // 執行一次 animation.OnMove()，animation 才會換圖

      if (animation.GetCurrentBitmapNumber() >= 15) is_alive = false;
}

void CBomb::OnShow() {
      if (!is_alive) return;      // 若 is_alive 為 false 則不執行 OnShow
```

```
	animation.SetTopLeft(x - 20, y - 20);
	animation.OnShow();
}

void CBomb::SetXY(int x, int y) {
	this->x = x;
	this->y = y;
}
bool CBomb::IsAlive() {
	return is_alive;
}

}
```

3. CBossA.h

```
#pragma once
namespace game_framework {

class CBossA : public CEnemy {
	public:
		CBossA();
		CBossA(int x, int y, int delay, bool alive, CDict* d, int minVL, int maxVL, vector<CEnemy*>* enemyQueue,
vector<CBomb*>* bombList, vector<CMovingBitmap*>* letter);
// (X 軸, Y 軸, 移動速度, 生死, 字典檔, 最小長度, 最大長度, 敵人隊伍)
		void CallEnemy(int, int);				// BossA 的技能: 召喚小怪(3~4 字)
		void kill();
		void OnMove();
		void LoadBitmap();
	protected:
		int callEnemyCounter, maxCallEnemyCounter;	// 發動召喚小怪技能 之 計數器
};
}
```

4. CBossA.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <time.h>
#include "audio.h"
#include "gamelib.h"
#include "CDict.h"
#include "CMe.h"
#include "CEnemy.h"
#include "CBossA.h"

namespace game_framework {
CBossA::CBossA() {}
CBossA::CBossA(int x, int y, int delay, bool alive, CDict* d, int minVL, int maxVL, vector<CEnemy*>* enemyQueue,
vector<CBomb*>* bombList, vector<CMovingBitmap*>* letter) {		//	初始值都在此處設定
	is_alive = is_bombed = false;
	dx = dy = index = delay_counter = 0;
	currWordLeng = 0;
	bossType = "bossA";
	////
	SetXY(x, y);
	SetDelay(delay);
	SetIsAlive(alive);
	this->dict = d;
	this->bombList = bombList;
	this->enemyQueue = enemyQueue;
	this->minVocabLeng = minVL;
	this->maxVocabLeng = maxVL;
	this->letter = letter;
	//
	index = (maxVocabLeng == 1 && minVocabLeng == 1) ? 10 : 0;
	callEnemyCounter = maxCallEnemyCounter = 30 * 5;		// 發動召喚小怪技能的間隔
	this->endX = SIZE_X / 2 - 30 + rand() % 60;
	this->endY = SIZE_Y;
	this->letter = letter;
	//
```

```
	SetVocab();
}

void CBossA::CallEnemy(int x, int y) {
	enemyQueue->push_back(new CEnemy(x, y, 2, false, dict, 3, 4, enemyQueue, bombList, SIZE_X / 2 - 50 + rand() % 100,
SIZE_Y, letter));
	enemyQueue->back()->LoadBitmap();
	enemyQueue->back()->SetIsAlive(true);
}

void CBossA::kill() {
	is_alive = false;
	bombList->push_back(new CBomb(GetX() + 10, GetY() + 10));
	bombList->back()->LoadBitmap();
	is_bombed = true;
}

void CBossA::OnMove() {
	const int STEPS = 200;			// 切成幾分 dx

	if (!is_alive) return;

	delay_counter--;
	callEnemyCounter--;

	if (delay_counter < 0) {
		delay_counter = delay;
		index++;

		if (index >= STEPS)
			index = 0;

		double dxTemp = (double(endX) - x) / STEPS * index;
		double dyTemp = (double(endY) - y) / STEPS * index;
		dx = int(dxTemp);			// dx 為 (Enemy<->Me 之 x 總距離) / STEPS * index;
		dy = int(dyTemp);
	}

	if (callEnemyCounter < 0) {		// BossA 技能:召喚小怪
		callEnemyCounter = maxCallEnemyCounter;
		CallEnemy((this->x + dx	/ 2 + rand() % bmp.Width()), (this->y + dy + 5 + bmp.Height() ));
	}
}


void CBossA::LoadBitmap() {
	char str[30];
	const unsigned int bitmapNum = 7;					// 圖檔數量
	sprintf(str, "Bitmaps/face/face_boss%d.bmp", rand() % bitmapNum + 1);	// 隨機挑選 bitmap
	bmp.LoadBitmap(str, RGB(0, 255, 0));				// 載入 怪物 SKIN
	textCursor.LoadBitmap(IDB_TEXT_CURSOR, RGB(0, 255, 0));		// 載入 光標
	talkBoxL.LoadBitmap(IDB_TALK_BOX_LEFT, RGB(0, 255, 0));		// 載入 對話框左
	talkBoxC.LoadBitmap(IDB_TALK_BOX_CENTER, RGB(0, 255, 0));	// 載入 對話框中
	talkBoxR.LoadBitmap(IDB_TALK_BOX_RIGHT, RGB(0, 255, 0));	// 載入 對話框右
}

}
```

5. CBossB.h

```
#pragma once
namespace game_framework {

class CBossB : public CBossA {
	public:
		CBossB();
		CBossB(int x, int y, int delay, bool alive, CDict* d, int minVL, int maxVL, vector<CEnemy*>* enemyQueue,
vector<CBomb*>* bombList, vector<CMovingBitmap*>* letter);// (X 軸, Y 軸, 移動速度, 生死, 字典檔, 最小長度, 最大長
度, 敵人隊伍)
		void CallEnemy(int, int);		// BossA 的技能: 召喚小怪(3~4 字)
		void OnMove();
```

```
        void LoadBitmap();
};
}
```

6. CBossB.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <time.h>
#include "audio.h"
#include "gamelib.h"
#include "CDict.h"
#include "CMe.h"
#include "CEnemy.h"
#include "CBossA.h"
#include "CBossB.h"
#include <math.h>
namespace game_framework {
CBossB::CBossB() {}
CBossB::CBossB(int x, int y, int delay, bool alive, CDict* d, int minVL, int maxVL, vector<CEnemy*>* enemyQueue,
vector<CBomb*>* bombList, vector<CMovingBitmap*>* letter) {          //       初始值都在此處設定
    this->enemyQueue = enemyQueue;
    this->bombList = bombList;
    is_alive = is_bombed = false;
    dx = dy = index = delay_counter = 0;
    currWordLeng = 0;
    bossType = "bossB";
    ////
    SetXY(x, y);
    SetDelay(delay);
    SetIsAlive(alive);
    dict = d;
    this->bombList = bombList;
    minVocabLeng = minVL;
    maxVocabLeng = maxVL;
    callEnemyCounter = maxCallEnemyCounter = 30 * 7;                  // 發動召喚小怪技能的間隔
    endX = SIZE_X / 2    - 30 + rand() % 60;
    endY = SIZE_Y;
    this->letter = letter;
    //
    SetVocab();
}

void CBossB::CallEnemy(int x, int y) {
    const double PI = 3.141592653;                                    // 定義圓周率
    const int ONE_WORD_ENEMY_NUM = 7;                                 // 共要生成幾隻小怪，範圍:3,5,7,9...
    const double ONE_WORD_ENEMY_RADIOUS = (PI / 180.0) * 7.0 ;        // 每一隻小怪的角度偏移量，double 內填寫
角度
                                                                      // 兩側的小怪

    for (int i = 1; i <= (ONE_WORD_ENEMY_NUM - 1) / 2; i++) {
        double r = ONE_WORD_ENEMY_RADIOUS * i;
        enemyQueue->push_back(new CEnemy(x + (bmp.Width() / 2) - 5, y + 40, 1, false, dict, 1, 1, enemyQueue, bombList, \
                                         int(double(x) + double(800) * cos(PI / 2 - r)), int(double(y) + double(600) *
sin(PI / 2 - r)), letter));
        enemyQueue->back()->LoadBitmap();
        enemyQueue->back()->SetIsAlive(true);
        enemyQueue->push_back(new CEnemy(x + (bmp.Width() / 2) - 5, y + 40, 1, false, dict, 1, 1, enemyQueue, bombList, \
                                         int(double(x) + double(800) * cos(PI / 2 + r)), int(double(y) + double(600) *
sin(PI / 2 + r)), letter));
        enemyQueue->back()->LoadBitmap();
        enemyQueue->back()->SetIsAlive(true);
    }

    //////////
    // 中間的那隻小怪
    enemyQueue->push_back(new CEnemy(x + (bmp.Width() / 2) - 5, y + 40, 1, false, dict, 1, 1, enemyQueue, bombList, \
```

```
                                        int(double(x) + double(800) * cos(PI / 2)), int(double(y) + double(600) * sin(PI /
2)), letter));
    enemyQueue->back()->LoadBitmap();
    enemyQueue->back()->SetIsAlive(true);
}
void CBossB::OnMove() {
    const int STEPS = 200;  // 切成幾分 dx

    if (!is_alive) return;

    delay_counter--;
    callEnemyCounter--;

    if (delay_counter < 0) {
        delay_counter = delay;
        index++;

        if (index >= STEPS)
            index = 0;

        double dxTemp = (double(endX) - x) / STEPS * index;
        double dyTemp = (double(endY) - y) / STEPS * index;
        dx = int(dxTemp);                   // dx 為  (Enemy<->Me 之 x 總距離) / STEPS * index;
        dy = int(dyTemp);
    }

    if (callEnemyCounter < 0) {             // BossA 技能:召喚小怪
        callEnemyCounter = maxCallEnemyCounter;
        CallEnemy((this->x + dx ), (this->y + dy));
    }
}
void CBossB::LoadBitmap() {
    char str[30];
    const unsigned int bitmapNum = 7;               // 圖檔數量
    sprintf(str, "Bitmaps/face/face_boss%d.bmp", rand() % bitmapNum + 1);
    bmp.LoadBitmap(str, RGB(0, 255, 0));                    // 載入 怪物 SKIN
    //
    textCursor.LoadBitmap(IDB_TEXT_CURSOR, RGB(0, 255, 0));            // 載入 光標
    talkBoxL.LoadBitmap(IDB_TALK_BOX_LEFT, RGB(0, 255, 0));            // 載入 對話框左
    talkBoxC.LoadBitmap(IDB_TALK_BOX_CENTER, RGB(0, 255, 0));          // 載入 對話框中
    talkBoxR.LoadBitmap(IDB_TALK_BOX_RIGHT, RGB(0, 255, 0));           // 載入 對話框右
    /////
}

}
```

7. CBullet.h

```
#pragma once
namespace game_framework {
// CBullet:     主角攻擊時發射的子彈

class CBullet {
    public:
        CBullet(int endX, int endY);
        void LoadBitmap();                      // 載入圖形
        void OnMove();                          // 移動
        void OnShow();                          // 將圖形貼到畫面
        void SetXY(int, int);
        void SetEndXY(int, int);
        bool IsAlive();
    private:
        int x, y;                               // 圖形座標   where ZHUJIAO is
        CAnimation animation;                   // 利用動畫作圖形
        int endX, endY;                         // where X & Y should end, it's the position of the enemy.
        int dx, dy;                             // 位移量
        int delay, delay_counter, index;        //delay_counter=delay; ;delay--
        bool is_alive;
};
}
```

8. CBullet.cpp

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

namespace game_framework {
// CBullet:  子彈
CBullet::CBullet(int endX, int endY) {
	/////
	delay = delay_counter = 0;
	dx = dy = index = 0;
	is_alive = 1;
	this->endX = endX;
	this->endY = endY;
	/////
	x = SIZE_X / 2;                         // 暫時設為中間底部
	y = SIZE_Y - 10;
	LoadBitmap();
}

void CBullet::LoadBitmap() {
	char* filename[2] = { ".\\bitmaps\\bullet1.bmp", ".\\bitmaps\\bullet2.bmp"};

	for (int i = 0; i < 2; i++)
		animation.AddBitmap(filename[i], RGB(0, 255, 0));
}

void CBullet::OnMove() {
	delay_counter--;
	animation.OnMove();                     // 執行一次 animation.OnMove()，animation 才會換圖
	const int STEPS = 10;

	if (delay_counter <= 0 && is_alive) {
		index++;

		if (index    >= STEPS ) {
			is_alive = 0;
		}

		dx = -(x - endX) / STEPS * index;
		dy = -(y - endY) / STEPS * index;
		delay_counter = delay;              // 計數器歸回原位
	}
}

void CBullet::OnShow() {
	if (is_alive) {
		animation.SetTopLeft(x + dx, y + dy);
		animation.OnShow();
	}
}

void CBullet::SetXY(int x, int y) {
	this->x = x;
	this->y = y;
}

void CBullet::SetEndXY(int x, int y) {
	this->endX = x;
	this->endY = y;
}
bool CBullet::IsAlive() {
	return is_alive;
}
```

```
}
```

9. CCharacter.h

```
#pragma once

namespace game_framework {

class CCharacter {
    public:
        CCharacter(string name, string subName, string fileName, int bmpNum, int, int, int, int);
        // 名稱, 圖片路徑陣列, 圖片數量,  UNLOCK 條件 1~4
        // 解鎖條件(依序)： 1.累計總正確按鍵數 2.單場分數 3.單場正確率 4.單場達到最高關卡
        void LoadBitmap();
        void OnMove();
        void OnShow();
        void SetXY(int, int);
        int   GetX1(), GetY1(), GetX2(), GetY2();
        int   GetWidth(), GetHeight();
        string GetName(), GetSubName();
        void SetIsUnlock(bool);
        bool GetIsUnlock();
        int GetUnlockRequirement(int);   // 回傳第 X 個解鎖條件, X=0~3
    protected:
        CAnimation              animation;                          // 主角動畫
        int                     bmpNum;                             // 圖檔數量
        string                  fileName;                           // 檔名
        string                  name, subName;                      // 名稱
        int                     x, y;
        bool                    isUnlock;                           // 是否解鎖
        int                     unlockRequirement[4];               // 解鎖條件
};
}
```

10. CCharacter.cpp

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

namespace game_framework {
// CCharacter:  主角

CCharacter::CCharacter(string name, string subName, string fn, int bmpNum, int ur0, int ur1, int ur2, int ur3) {
    this->name = name;
    this->subName = subName;
    this->bmpNum = bmpNum;
    this->x = 0;
    this->y = 0;
    this->fileName = fn;
    this->isUnlock = false;
    unlockRequirement[0] = ur0;
    unlockRequirement[1] = ur1;
    unlockRequirement[2] = ur2;
    unlockRequirement[3] = ur3;
    LoadBitmap();
}

void CCharacter::LoadBitmap() {
    char str[50];

    if (bmpNum == 1) {
        sprintf(str, "Bitmaps/me/%s.bmp", fileName.c_str());
        animation.AddBitmap(str, RGB(0, 255, 0));
    }
    else {
```

```
            for (int i = 0; i < bmpNum; i++) {            // 載入動畫
                sprintf(str, "Bitmaps/me/%s%d.bmp", fileName.c_str(), i + 1);
                animation.AddBitmap(str, RGB(0, 255, 0));
            }
        }

        if (name == "Tsai Ing-wen")animation.SetDelayCount(5);

        if (name == "Kirby")animation.SetDelayCount(3);
}

void CCharacter::OnMove() {
        animation.OnMove();
}

void CCharacter::OnShow() {
        animation.SetTopLeft(x, y);
        animation.OnShow();
}
int CCharacter::GetUnlockRequirement(int num) {
        return num < 4 ? unlockRequirement[num] : 0;
}

void CCharacter::SetIsUnlock(bool isUnlock) {
        this->isUnlock = isUnlock;
}
bool CCharacter::GetIsUnlock() {
        return isUnlock;
}
void CCharacter::SetXY(int x, int y) {
        this->x = x;
        this->y = y;
}
int CCharacter::GetX1() {
        return x;
}
int CCharacter::GetY1() {
        return y;
}
int CCharacter::GetX2() {
        return x + animation.Width();
}
int CCharacter::GetY2() {
        return y + animation.Height();
}
int CCharacter::GetWidth() {
        return animation.Width();
}
int CCharacter::GetHeight() {
        return animation.Height();
}
string CCharacter::GetName() {
        return name;
}
string CCharacter::GetSubName() {
        return subName;
}
}
```

11. CDict.h

```
#pragma once
namespace game_framework {
// CDict:       管理字典檔
class CDict {
        public:
                CDict();
                string GetText();
        private:
                vector<string> dictionary;
};
```

```
}
```

12. CDict.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CDict.h"
#include <fstream>
#include <string>
#include <vector>
#include <cctype>
#include <ctime>

namespace game_framework {
// CDict: 字典

CDict::CDict() {
    fstream file;
    bool repeated;                                          // 重複
    file.open("dict/text.txt", ios::in);                    // 讀取字典檔
    string temp;

    if (file) {
        while (file >> temp) {                              // 判斷讀入的單字 在字典裡是否已重複
            repeated = false;

            for (unsigned int i = 0; i < dictionary.size(); i++) {
                if (temp == dictionary[i])
                    repeated = true;
            }

            if (!repeated)
                dictionary.push_back(temp);                 // 將字典檔讀入
        }
    }

    file.close();
}
string CDict::GetText() {                                   // 每次 GetText()時 都隨機 return 一個單字
    unsigned int rnd = rand()    % dictionary.size();       // rnd 從 0~字典字數 隨機抓數字
    return dictionary[rnd];
}

}
```

13. CEmp.h

```
#pragma once

namespace game_framework {

enum AUDIO_EMP_ID {
    AUDIO_EMP
};

class CEmp {
    public:
        CEmp();
        int    GetX1();                                     // 左上角 x 座標
        int    GetY1();                                     // 左上角 y 座標
        int    GetX2();                                     // 右下角 x 座標
        int    GetY2();                                     // 右下角 y 座標
        void Initialize();                                  // 設定初始值
        void LoadBitmap();                                  // 載入圖形
        void OnMove();
        void OnShow();
        void SetXY(int nx, int ny);                         // 設定左上角座標
        void SetEQ(vector<CEnemy*>* enemyQueue, CInteger* score, bool* lock, CEnemy** targetEnemy);
        void CallEmp(bool);
        void SetEmpTimes(int);                              // 設定電磁波可使用的次數
        int GetEmpTimes();
    protected:
        CMovingBitmap           displayBG, displayNumber[10];
        CAnimation              emp;                        // 衝擊波
        vector<CEnemy*>*        enemyQueue;                 // [指標] 儲存所有敵人的 Vector
```

```
        CInteger*               score;              //[指標] 分數
        bool*                   lock;               //[指標] 是否已鎖定敵人
        CEnemy**                targetEnemy;        //[指標] 被鎖定的敵人
        bool                    state;              // 是否處於 技能播放中
        int                     x, y;               // 左上角座標
        int                     empTimes;           // 剩餘可召喚的 EMP 次數
    private:
        bool HitRectangle(int tx1, int ty1, int tx2, int ty2);      // 是否碰到參數範圍的矩形

};
}
```

14. CEmp.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "CEnemy.h"
#include "CEmp.h"


namespace game_framework {
// CEmp: EMP(電磁波)

CEmp::CEmp() {
    Initialize();
}

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

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

int CEmp::GetX2() {
    return x + emp.Width();
}

int CEmp::GetY2() {
    return y + emp.Height();
}

void CEmp::Initialize() {
    const int X_POS = ( SIZE_X - 640) / 2;
    const int Y_POS = SIZE_Y - 350;
    x = X_POS;
    y = Y_POS;
    state = false;
    emp.SetDelayCount(2);
    empTimes = 3;
}

void CEmp::LoadBitmap() {
    for (int i = 0; i < 7; i++) {               // 載入 電磁波動畫圖
        char str[40];
        sprintf(str, "Bitmaps/big_wave/big_wave%d.bmp", i);
        emp.AddBitmap(str, RGB(0, 255, 0));
    }

    for (int i = 0; i < 10; i++) {              // 載入 數字圖
        char str[40];
        sprintf(str, "Bitmaps/emp_text/%d.bmp", i);
        displayNumber[i].LoadBitmap(str, RGB(0, 255, 0));
        displayNumber[i].SetTopLeft(SIZE_X - 80 + 16, SIZE_Y - 80 + 12);
    }

    displayBG.LoadBitmap("Bitmaps/emp_text/bg.bmp", RGB(0, 255, 0));
```

```
	displayBG.SetTopLeft(SIZE_X - 80, SIZE_Y - 80);
	CAudio::Instance()->Load(AUDIO_EMP, "sounds\\EMP.wav");
}

void CEmp::OnMove() {
	if (state) {
		emp.OnMove();

		for (unsigned int i = 0; i < enemyQueue->size(); i++) {
			if	(HitRectangle(enemyQueue->at(i)->GetX(),	enemyQueue->at(i)->GetY(),	enemyQueue->at(i)->GetX2(),
enemyQueue->at(i)->GetY2())) {
				if (*lock && enemyQueue->at(i) == *targetEnemy) {
					*lock = 0;
				}

				score->Add(enemyQueue->at(i)->GetVocabLeng());
				enemyQueue->at(i)->kill();
			}
		}

		if (emp.IsFinalBitmap()) {
			state = false;
			emp.Reset();
		}
	}
}


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

void CEmp::OnShow() {
	emp.SetTopLeft((SIZE_X - 640) / 2, (SIZE_Y - 350));
	emp.OnShow();
	displayBG.ShowBitmap();                                          // 顯示 背景
	displayNumber[empTimes].ShowBitmap();                            // 顯示 剩餘次數數字
}

void CEmp::SetEQ(vector<CEnemy*>* enemyQueue, CInteger* score, bool* lock, CEnemy** targetEnemy) {
	this->enemyQueue = enemyQueue;
	this->score = score;
	this->lock = lock;
	this->targetEnemy = targetEnemy;
}

void CEmp::CallEmp(bool music) {
	if (!state && empTimes != 0) {
		empTimes--;
		state = true;

		if (music)CAudio::Instance()->Play(AUDIO_EMP, false);                    // 撥放 射擊音效
	}
}

bool CEmp::HitRectangle(int tx1, int ty1, int tx2, int ty2) {// 80 160 320 480 480
	if (emp.GetCurrentBitmapNumber() != 0 && !emp.IsFinalBitmap()) {
		int arr[10] = { 1, 2, 4, 6, 6, 6, 6, 6, 6 };
		int x1, y1, x2, y2;
		int	i = emp.GetCurrentBitmapNumber() - 1;
		x1 = x + (320 - (80 * arr[i]) / 2);
		y1 = y + (320 - (80 * arr[i]) / 2);
		x2 = x + (320 + (80 * arr[i]) / 2);
		y2 = y + (320 + (80 * arr[i]) / 2);
		// 檢測怪物 face 的矩形與參數矩形是否有交集
		return (tx2 >= x1 && tx1 <= x2 && ty2 >= y1 && ty1 <= y2);
	}
	else return 0;
```

```
}
void CEmp::SetEmpTimes(int num) {
	this->empTimes =	num;
}

int CEmp::GetEmpTimes() {
	return empTimes;
}

}
```

15. CEnemy.h

```
#pragma once
#include "CBomb.h"
#include "CDict.h"
#include "CMe.h"
namespace game_framework {

class CEnemy {
	public:
		CEnemy();
		~CEnemy();
		CEnemy(int x, int y, int delay, bool alive, CDict* dict, int minVL, int maxVL, \
			vector<CEnemy*>* enemyQueue, vector<CBomb*>* bombList, int endX, int endY, \
			vector<CMovingBitmap*>* letter);
		virtual void LoadBitmap();                          // 載入圖形
		virtual void OnMove();                              // 移動
		bool	IsAlive();                                  // 是否活著
		void	OnShow();                                   // 將圖形貼到畫面
		void	SetXY(int nx, int ny);                      // 設定坐標
		void	SetIsAlive(bool alive);                         // 設定是否活著
		void	SetDelay(int d);                            // 設定掉落速度

		//////////////

		void	SetVocab();                                 // 隨機從 dict 中抓取一個單字到 vocab 裡面
		string		GetVocab();                             // 回傳整組單字(ex: "apple")
		char	GetFirstWord();                             // 以 char 回傳一個字 (ex: 'a')
		void	AddCurrWordLeng();                          // CurrWord++
		int			GetCurrWordLeng();                      // 回傳 int 型態的 CurrWord
		int			GetVocabLeng();                         // 回傳單字總長度
		int			GetX(), GetY();                             // 取得 X 軸(x+dx), Y 軸(y+dy)
		int			GetX2(), GetY2();
                                                            // 取得 X2 (x+dx+bmp.width), Y2(y+dy+bmp.height)
		void	MinusIndex(int num);                        // 擊退怪物 num 為擊退多少 index
		bool	HitMe(CMe* me);                             // 是否碰到主角
		bool	IsBombed();                                 // 是否爆炸過了
		virtual void	kill();
		string			GetBossType();

	protected:
		CMovingBitmap bmp;                                  // 球的圖
		CMovingBitmap bmp_center;                           // 圓心的圖
		CMovingBitmap textCursor;                           // 文字光標 圖
		CMovingBitmap talkBoxL, talkBoxC, talkBoxR;         // 優化過的對話框 圖
		vector<CEnemy*>* enemyQueue;
		vector<CBomb*>* bombList;
		vector<CMovingBitmap*>* letter;                     // 文字圖檔
		CDict* dict;                                        // 字典檔指標
		string vocab;                                       // 儲存單字
		string bossType;                                    // 怪物的類型
		int x, y;                                           // 圓心的座標
		int dx, dy;                                         // 球距離圓心的位移量
		int endX, endY;                                     // 飛行的終點坐標
		int index;                                          // 將距離切成 index 等分
		int delay_counter;                                  // 掉落速度的計數器
		int delay;                                          // 掉落的速度
		bool is_alive;                                      // 是否活著
		bool is_bombed;                                     // 是否爆炸過了
		int length;                                         // 單字總長度
```

```
        int currWordLeng;                                   // 當前輸入文字的字元
        int minVocabLeng, maxVocabLeng;
    // 規定怪物生成單字長度的區間( minVocabLeng ~ maxVocabLeng)

    private:
        bool HitRectangle(int tx1, int ty1, int tx2, int ty2);          // 是否碰到參數範圍的矩形

};
}
```

16. CEnemy.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <time.h>
#include "audio.h"
#include "gamelib.h"
#include "CDict.h"
#include "CMe.h"
#include "CEnemy.h"
#include <math.h>
#include <stdlib.h>

namespace game_framework {
/////////////////////////////////////////////////////////////////////////////
// CEnemy: Enemy
/////////////////////////////////////////////////////////////////////////////
CEnemy::CEnemy() {
}
CEnemy::~CEnemy() {
}
CEnemy::CEnemy(int x, int y, int delay, bool alive, CDict* d, int minVL, int maxVL, vector<CEnemy*>* enemyQueue,
vector<CBomb*>* bombList, int endX, int endY, vector<CMovingBitmap*>* letter) {    //      初始值都在此處設定
    is_alive = is_bombed = false;
    dx = dy = index = delay_counter = 0;
    currWordLeng = 0;
    bossType = "enemy";
    ////
    SetXY(x, y);
    SetDelay(delay);
    SetIsAlive(alive);
    this->dict = d;
    this->bombList = bombList;
    this->enemyQueue = enemyQueue;
    this->minVocabLeng = minVL;
    this->maxVocabLeng = maxVL;
    this->endX = endX;
    this->endY = endY;
    this->letter = letter;
    SetVocab();
    index = (maxVocabLeng == 1 && minVocabLeng == 1) ? 10 : 0;
}

bool CEnemy::HitRectangle(int tx1, int ty1, int tx2, int ty2) {
    int x1 = x + dx;                                    // 怪物 face 的左上角 x 座標
    int y1 = y + dy;                                    // 怪物 face 的左上角 y 座標
    int x2 = x1 + bmp.Width();                          // 怪物 face 的右下角 x 座標
    int y2 = y1 + bmp.Height();                         // 怪物 face 的右下角 y 座標
    return (tx2 >= x1 && tx1 <= x2 && ty2 >= y1 && ty1 <= y2);      // 檢測怪物 face 的矩形與參數矩形是否有交集
}

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

void CEnemy::LoadBitmap() {
    char str[30];
    const unsigned int bitmapNum = 7;                                   // 圖檔總數量
```

```
    if (maxVocabLeng == 1 && minVocabLeng == 1)                    // 隨機挑選 bitmap
        sprintf(str, "Bitmaps/face/face_min%d.bmp", rand() % (4) + 1);   // 1 字小怪的 bmp
    else {
        sprintf(str, "Bitmaps/face/face%d.bmp", rand() % bitmapNum + 1);   // 一般小怪的 bmp
    }

    bmp.LoadBitmap(str, RGB(0, 255, 0));                           // 載入 怪物 SKIN
    textCursor.LoadBitmap(IDB_TEXT_CURSOR, RGB(0, 255, 0));        // 載入 光標
    talkBoxL.LoadBitmap(IDB_TALK_BOX_LEFT, RGB(0, 255, 0));        // 載入 對話框左
    talkBoxC.LoadBitmap(IDB_TALK_BOX_CENTER, RGB(0, 255, 0));      // 載入 對話框中
    talkBoxR.LoadBitmap(IDB_TALK_BOX_RIGHT, RGB(0, 255, 0));       // 載入 對話框右
}

void CEnemy::OnMove() {
    const int STEPS = 180;                                         // 切成幾分 dx

    if (!is_alive) return;

    delay_counter--;

    if (delay_counter < 0) {
        delay_counter = delay;
        index++;
        double dxTemp = (double(endX) - x) / STEPS * index;
        double dyTemp = (double(endY) - y) / STEPS * index;
        dx    = int(dxTemp);                                        // dx 為 (Enemy<->Me 之 x 總距離) / STEPS *
index;
        dy    = int(dyTemp);
    }
}

void CEnemy::SetDelay(int d) {
    delay = d;
}
void CEnemy::SetIsAlive(bool alive) {
    is_alive = alive;
}
void CEnemy::SetXY(int nx, int ny) {
    x = nx;
    y = ny;
}
void CEnemy::OnShow() {
    if (is_alive) {
        bmp.SetTopLeft(x + dx, y + dy);
        bmp.ShowBitmap();

        if (currWordLeng < length) {                               // 若當前輸入到的字 < 總長度 則顯示對話框
            talkBoxL.SetTopLeft(x + dx + bmp.Width(), y + dy);
            talkBoxL.ShowBitmap();                                 // 顯示 對話框左

            for (int i = 0; i < length; i++) {                     // 顯示 數個對話框中(根據單字長度)
                talkBoxC.SetTopLeft(x + dx + bmp.Width() + talkBoxL.Width() + i * talkBoxC.Width(), y + dy);
                talkBoxC.ShowBitmap();
            }

            talkBoxR.SetTopLeft(x + dx + bmp.Width() + talkBoxL.Width() + length * talkBoxC.Width(), y + dy);
            talkBoxR.ShowBitmap();                                 // 顯示 對話框右
            textCursor.SetTopLeft(x + dx + bmp.Width() + talkBoxL.Width() + ((currWordLeng ) * 10) - 1, \
                                  y + dy);
            (currWordLeng != 0) ? textCursor.ShowBitmap() : 0;     // 若當前輸入到的字!=0 則顯示光標

            for (int i = 0; i < length; i++) {                     // 顯示單字 bmp
                letter->at(vocab[i] - 97)->SetTopLeft(x + dx + bmp.Width() + talkBoxL.Width() + letter->at(0)->Width() * i,
y + dy + 4);
                letter->at(vocab[i] - 97)->ShowBitmap();
            }

            for (int i = 0; i < currWordLeng; i++) {               // 再次顯示 talkBoxC, 讓打過的單字 被蓋掉消失
```

```
不見
                    talkBoxC.SetTopLeft(x + dx + bmp.Width() + talkBoxL.Width() + i * talkBoxC.Width(), y + dy);
                    talkBoxC.ShowBitmap();
                }
            }
        }
}
void CEnemy::kill() {
    is_alive = false;
    bombList->push_back(new CBomb(GetX(), GetY()));          // new 爆炸特效進 vecor
    bombList->back()->LoadBitmap();
    is_bombed = true;
}


void   CEnemy::SetVocab() {                                  // 隨機從 dict 中抓取一個單字到 vocab 裡面
    while (1) {
        if (maxVocabLeng == 1 && minVocabLeng == 1) {        // 若為 1 字小怪
            vocab = "a";                                     // 先隨意給予 1 字 string
            vocab[0] = 97 + rand() % 26;                     // 隨機挑選單字
            length = 1;

        else {                                                   // 一般長度怪物
            vocab = dict->GetText();                             // 從字典中隨機挑選單字給 vocab
            length = vocab.length();                             // 長度= 單字長度
        }

        if (length >= minVocabLeng && length <= maxVocabLeng) {     // 長度符合 ,使用 break 跳出迴圈 確定生成此單
字
            bool firstWordBounceFlag = 0;                            // 有撞到第一個單字的 flag

            for (int i = enemyQueue->size() - 1; i >= 0; i--) {              // 與場上所有怪物比較, 確定是否首字單字重複
                if (vocab[0] == enemyQueue->at(i)->GetFirstWord() && enemyQueue->at(i)->IsAlive())
                    firstWordBounceFlag = 1;
            }

            if (!(firstWordBounceFlag && enemyQueue->size() < 25))
                break;                                           // 若沒撞到且 場上怪物數量小於 26
        }
    }
}

string CEnemy::GetVocab() {                                  // 回傳整組單字(ex: "apple")
    return vocab;
}
char CEnemy::GetFirstWord() {                                // 以 char 回傳一個字 (ex: 'a')
    return vocab[0];
}
void CEnemy::AddCurrWordLeng() {
    currWordLeng++;
}
int CEnemy::GetCurrWordLeng() {
    return currWordLeng;
}
int CEnemy::GetVocabLeng() {
    return length;
}
int CEnemy::GetX() {
    return x + dx;
}
int CEnemy::GetY() {
    return y + dy;
}
int CEnemy::GetX2() {
    return x + dx + bmp.Width();
}
int CEnemy::GetY2() {
    return y + dy + bmp.Height();
}
```

```
void CEnemy::MinusIndex(int num) {
	index = index - num;
}
bool CEnemy::HitMe(CMe* me) {
	return HitRectangle(me->GetX1(), me->GetY1(),
				me->GetX2(), me->GetY2());
}
bool CEnemy::IsBombed() {
	return is_bombed;
}
string CEnemy::GetBossType() {
	return bossType;
}
}
```

17. CFile.h

```
#pragma once
#include "CRecord.h"
namespace game_framework {

class CFile {
	public:
		CFile();
		void WriteHighScore(int score, int level, double accuracy, string meName, int correctKeyCount);
		void ReadHighScoreFile();
		int ReadHighScore_Score();                                  //回傳最高分裡面的分
		int ReadHighScore_Level();                                  //回傳最高分裡面的關
		double ReadHighScore_Accuracy();                            //回傳最高分裡面的正確率
		string	ReadHighScore_Character();                          //回傳最高分裡面的角色名
		string	ReadHighScore_Date();                               //回傳最高分裡面的日期
		int		ReadHighScore_CorrectKeyCount();                    //回傳最高分裡面的正確按鍵數量 （單場）
		int		ReadHighScore_TotalKeyCount();                      //回傳總按鍵數（會累積）
		bool isHighScoreExist();                                    //回傳是否存在最高紀錄
		//================================
		void WriteRecord(int score, int level, double accuracy, string meName, int correctKeyCount);
//寫入:分數 關卡 正確率 主角名稱
		void ReadRecordFile();
		int ReadRecord_Score(int);                                      //回傳第 x 筆裡面的分
		int ReadRecord_Level(int);                                      //回傳第 x 筆裡面的關
		double ReadRecord_Accuracy(int);                                //回傳第 x 筆裡面的正確率
		string	ReadRecord_Character(int);                          //回傳第 x 筆裡面的角色名
		string	ReadRecord_Date(int);                               //回傳第 x 筆裡面的日期
		int		ReadRecord_CorrectKeyCount(int);                    //回傳第 x 筆裡面的正確按鍵數量
		int		GetRecordNum();                                     //回傳 record 大小
		//================================
		void WriteSelectedCharacter(string);                        //寫入上一次所選角色
		string	ReadSelectedCharacter();                            //讀取上一次所選角色
		//================================
		void WriteMusicOnOff(bool);
		bool ReadMusicOnOff();

		//-----------
		void WriteTotalKeyCount(int);
		int ReadTotalKeyCount();
		//================================

		void DeleteAllData();                                       //清除所有記錄

	private:
		int HighScore_Score;                                        //最高紀錄的得分
		int HighScore_Level;                                        //最高紀錄的關卡
		double HighScore_Accuracy;                                  //最高紀錄的正確率
		string HighScore_Date;                                      //最高紀錄的日期
		string HighScore_MeName;                                    //最高紀錄主角名稱(英文)
		int HighScore_CorrectKeyCount;                              //最高紀錄的正確按鍵數量
		int HighScore_TotalKeyCount;                                //總按鍵數（會累積的）
		//==================
		vector<CRecord*> record;
		int record_Score;                                           //得分
		int record_Level;                                           //關卡
```

```
        double record_Accuracy;                         //正確率
        string record_Date;                             //日期
        string record_MeName;                           //主角名稱(英文)
        int    record_SelectedChar;                     //主角編號
        int    record_CorrectKeyCount;                  //正確按鍵數
        char ChDate[80];                                //char 型態的日期
        //=====================
        string      preSelectedChar;                    //上一隻角色
        //================================
        bool    musicState;

};
}
```

18. CFile.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include <string>
#include <sstream>
#include <fstream>
#include <ctime>
#include "CFile.h"


namespace game_framework {


CFile::CFile() {
}
void CFile::WriteHighScore(int score, int level, double accuracy, string meName, int correctKeyCount) {
    struct tm* T = NULL;
    time_t t;
    time(&t);
    T = localtime(&t);
    fstream    fp, test;
    fp.open("user/bestRecord.txt", ios::out);
    sprintf(ChDate,    "%d%02d%02d%02d%02d",    int(T->tm_year    +    1900),    int(T->tm_mon    +    1),    int(T->tm_mday),
int(T->tm_hour), int(T->tm_min));
    this->HighScore_Date.assign(ChDate);
    fp << "character:" << meName
        << ",score:" << score
        << ",level:" << level
        << ",accuracy:" << accuracy
        << ",date:" << HighScore_Date
        << ",correctKeyCount:" << correctKeyCount << endl;
    fp.close();
}
void CFile::WriteTotalKeyCount(int num) {
    fstream    fp;
    fp.open("user/tkc.txt", ios::out);
    fp << num;
    fp.close();
}
int CFile::ReadTotalKeyCount() {
    fstream    fp;
    fp.open("user/tkc.txt", ios::in);
    string temp;

    if (fp) {
        while (fp >> temp) {}
    }

    return stoi(temp);
    fp.close();
}
```

```
void CFile::ReadHighScoreFile() {
	string slideOne, slideTwo[14];
	char temp[200];
	int i = 0;
	fstream	fp;
	fp.open("user/bestRecord.txt", ios::in);

	while (fp.getline(temp, sizeof(temp), ',')) {
		slideOne = temp;
		stringstream ss(slideOne);

		while (getline(ss, slideTwo[i], ':')) {
			if (i == 1) this->HighScore_MeName = slideTwo[1];
			else if (i == 3) this->HighScore_Score = stoi(slideTwo[3], nullptr, 10);
			else if (i == 5) this->HighScore_Level = stoi(slideTwo[5], nullptr, 10);
			else if (i == 7) this->HighScore_Accuracy = stod(slideTwo[7], nullptr);
			else if (i == 9) this->HighScore_Date = slideTwo[9];
			else if (i == 11) this->HighScore_CorrectKeyCount = stoi(slideTwo[11], nullptr, 10);

			i++;
		}
	}

	if (!fp.is_open())HighScore_Score = 0;

	fp.close();
	TRACE("%d, %d\n", HighScore_Score, HighScore_Level);
}
int CFile::ReadHighScore_Score() {
	return this->HighScore_Score;
}
int CFile::ReadHighScore_Level() {
	return this->HighScore_Level;
}
double CFile::ReadHighScore_Accuracy() {
	return this->HighScore_Accuracy;
}
string CFile::ReadHighScore_Character() {
	return this->HighScore_MeName;
}
string CFile::ReadHighScore_Date() {
	return this->HighScore_Date;
}
int	CFile::ReadHighScore_CorrectKeyCount() {
	return this->HighScore_CorrectKeyCount;
}
bool CFile::isHighScoreExist() {
	if (!(HighScore_Score == 0))return 1;
	else return 0;
}
//=====================================
void CFile::WriteRecord(int score, int level, double accuracy, string meName, int correctKeyCount) {
	struct tm* T = NULL;
	time_t t;
	time(&t);
	T = localtime(&t);
	fstream	fp;
	fp.open("user/record.txt", ios::out | ios::app);
	sprintf(ChDate,	"%d%02d%02d%02d%02d",	int(T->tm_year	+	1900),	int(T->tm_mon	+	1),	int(T->tm_mday),
int(T->tm_hour), int(T->tm_min));
	this->record_Date.assign(ChDate);
	fp << "character:" << meName
	   << ",score:" << score
	   << ",level:" << level
	   << ",accuracy:" << accuracy
	   << ",date:" << record_Date
	   << ",correctKeyCount:" << correctKeyCount << endl;
	fp.close();
```

```
}
void CFile::ReadRecordFile() {
	fstream	fp;
	fp.open("user/record.txt", ios::in);
	int i = 0;
	char line[200];
	string SlideOne, SlideTwo, SlideThree[100];

	for (CRecord* cr : record) delete cr;

	record.clear();

	while (fp.getline(line, sizeof(line), '\n')) {
		SlideOne = line;
		stringstream ss(SlideOne);

		while (getline(ss, SlideTwo, ',')) {
			stringstream ss2(SlideTwo);

			while (getline(ss2, SlideThree[i], ':')) {
				if (i == 1) this->record_MeName = SlideThree[1];
				else if (i == 3) this->record_Score = stoi(SlideThree[3], nullptr, 10);
				else if (i == 5) this->record_Level = stoi(SlideThree[5], nullptr, 10);
				else if (i == 7) this->record_Accuracy = stod(SlideThree[7], nullptr);
				else if (i == 9) this->record_Date = SlideThree[9];
				else if (i == 11) this->record_CorrectKeyCount = stoi(SlideThree[11], nullptr, 10);

				i++;

				if (i == 12) i = 0;
			}
		}

		if (record_MeName != "")
			record.insert(record.begin(), new CRecord(record_Score, record_Level, record_Accuracy, record_MeName,
record_CorrectKeyCount, record_Date));		//從頭插進去
	}

	TRACE("size[%d]\n", record.size());
}
int CFile::ReadRecord_Score(int num) {
	return record.at(num)->ReadRecordScore_Score();
}
int CFile::ReadRecord_Level(int num) {
	return record.at(num)->ReadRecordScore_Level();
}
double CFile::ReadRecord_Accuracy(int num) {
	return record.at(num)->ReadRecordScore_Accuracy();
}
string CFile::ReadRecord_Character(int num) {
	return record.at(num)->ReadRecordScore_Character();
}
string CFile::ReadRecord_Date(int num) {
	return record.at(num)->ReadRecordScore_Date();
}
int	CFile::ReadRecord_CorrectKeyCount(int num) {
	return record.at(num)->ReadRecordScore_CorrectKeyCount();
}
int CFile::GetRecordNum() {
	return int(record.size());
}
void CFile::DeleteAllData() {
	fstream	fp, test;
	fp.open("user/bestRecord.txt", ios::out);
	fp << "";
	fp.close();
	fp.open("user/preSelectedChar.txt", ios::out);
	fp << "";
	fp.close();
```

```
	fp.open("user/record.txt", ios::out);
	fp << "";
	fp.close();
	fp.open("user/unlock.txt", ios::out);
	fp << "Iron Man\n";
	fp.close();
	fp.open("user/tkc.txt", ios::out);
	fp << "0";
	fp.close();
	DeleteFile("user/bestRecord.txt");
}
int	CFile::ReadHighScore_TotalKeyCount() {
	return this->HighScore_TotalKeyCount;
}
string CFile::ReadSelectedCharacter() {
	fstream	fp;
	fp.open("user/preSelectedChar.txt", ios::in);
	int i = 0;
	char line[200];
	string SlideOne;

	while (fp.getline(line, sizeof(line), ':')) {
		SlideOne = line;

		if (i == 1) this->preSelectedChar = SlideOne;

		i++;
	}

	fp.close();
	return this->preSelectedChar;
}
void CFile::WriteSelectedCharacter(string preSelectedChar) {
	fstream	fp;
	fp.open("user/preSelectedChar.txt", ios::out);
	fp << "preSelectedChar:" << preSelectedChar;
	fp.close();
}
void CFile::WriteMusicOnOff(bool musicState) {
	fstream fp;
	fp.open("user/musicState.txt", ios::out);
	fp << "musicState:" << musicState;
	fp.close();
}
bool CFile::ReadMusicOnOff() {
	fstream	fp;
	fp.open("user/musicState.txt", ios::in);
	int i = 0;
	char line[200];
	string SlideOne;

	while (fp.getline(line, sizeof(line), ':')) {
		SlideOne = line;
		stringstream ss(SlideOne);

		if (i == 1) ss >> musicState;

		i++;
	}

	fp.close();
	return this->musicState;
}
}
```

19. CLevel.h

```
#pragma once

namespace game_framework {
```

```
class CLevel {
    public:
        CLevel();
        void LoadBitmap();
        void OnMove();
        void OnShow();
        //
        void SetXY(int, int);
        int   GetX1(), GetY1(), GetX2(), GetY2();
        int   GetWidth(), GetHeight();
        void Play(int level, int score);                                    // 播放關卡切換動畫

    protected:
        CMovingBitmap       border, bg;                                     // 框線和背景
        CMovingBitmap       numBmp[10], numBmpSmall[10];                    // 數字圖檔
        int                 x, y;
        int                 level, score;
        int                 delay_counter;
        double              easeC;
        int                 topPosY, btmPosY;
        int                 centerPosY, centerPosX;
};
}
```

20. CLevel.cpp

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

namespace game_framework {
// CLevel: 關卡切換動畫

CLevel::CLevel() {
    delay_counter = 20000;                  // 設定此數字讓動畫停止播放
    easeC = 0;
    level = 0;
    score = 87;
}
void CLevel::Play(int level, int score) {
    delay_counter = 0;
    easeC = 0;
    y = topPosY;
    this->level = level;
    this->score = score;
}
void CLevel::LoadBitmap() {
    char str[50];
    border.LoadBitmap("Bitmaps/level/level_border.bmp", RGB(0, 255, 0));
    bg.LoadBitmap("Bitmaps/level/level_bg.bmp", RGB(0, 255, 0));

    for (int i = 0; i < 10; i++) {                  // 載入數字圖
        sprintf(str, "Bitmaps/level/num/%d.bmp", i);
        numBmp[i].LoadBitmap(str, RGB(0, 255, 0));
        sprintf(str, "Bitmaps/level/num_s/%d.bmp", i);
        numBmpSmall[i].LoadBitmap(str, RGB(0, 255, 0));
    }

    topPosY = 0 - border.Height();
    btmPosY = SIZE_Y;
    centerPosX = (SIZE_X - border.Width()) / 2;
    centerPosY = (SIZE_Y - border.Height()) / 2 - 15;
    x = centerPosX;
    y = topPosY;
}
```

```
void CLevel::OnMove() {
    if (delay_counter < 30 * 10) delay_counter++; // 動畫播放期間 delay_counter 會持續增加

    if (delay_counter < 30 * 3) {                 // 滑入期間
        if (y < centerPosY) {
            easeC += .60;
            y += 20 - int(easeC)   ;
        }
        else    easeC = 0;
    }
    else if (delay_counter < 30 * 5) {                          // 滑出期間
        if (y < btmPosY) {
            easeC += .7;
            y += 0 + int(easeC);
        }
        else    easeC = 0;
    }
    else if (delay_counter < 30 * 10) {                         // 停止播放
        //delay_counter = 0;
        easeC = 0;
        y = topPosY;
    }
}

void CLevel::OnShow() {
    bg.SetTopLeft(x, y);
    bg.ShowBitmap();
    border.SetTopLeft(x, y);
    border.ShowBitmap();
    int tempScore = score, tempLevel = level;

    for (int i = 0; i < 5; i++) {                               // 顯示分數數字 bmp
        numBmpSmall[tempScore % 10].SetTopLeft(x + 127 - 10 * i, y + 76);
        numBmpSmall[tempScore % 10].ShowBitmap();
        tempScore /= 10;
    }

    for (int i = 0; i < 2; i++) {                               // 顯示關卡數字 bmp
        numBmp[tempLevel % 10].SetTopLeft(x + 138 - 20 * i, y + 10 );
        numBmp[tempLevel % 10].ShowBitmap();
        tempLevel /=    10;
    }
}

void CLevel::SetXY(int x, int y) {
    this->x = x;
    this->y = y;
}
int CLevel::GetX1() {
    return x;
}
int CLevel::GetY1() {
    return y;
}
int CLevel::GetX2() {
    return x + border.Width();
}
int CLevel::GetY2() {
    return y + border.Height();
}
int CLevel::GetWidth() {
    return border.Width();
}
int CLevel::GetHeight() {
    return border.Height();
}

}
```

21. CMap.h

```
#pragma once

namespace game_framework {

class CMap {
	public:
		CMap();
		void LoadBitmap();
		void OnMove();
		void OnShow();
		void PlayFlash();

	protected:
		CMovingBitmap net, net1, bgColor;
		const int X, Y;
		const int MW, MH;
		int index;
		int delay_counter;
		int delay;
		int dx, dy;
		int flashCounter, flashCounterMax;
};

}
```

22. CMap.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <time.h>
#include "gamelib.h"
#include "CMap.h"

namespace game_framework {
// CMap: 網狀背景

CMap::CMap(): X(0), Y(0), MW(27), MH(27) {
	delay = delay_counter = index = 0;
	dx = dy = 0;
	flashCounter = -1;
	flashCounterMax = 3;
}
void CMap::LoadBitmap() {
	bgColor.LoadBitmap("Bitmaps/background_color.bmp", RGB(0, 255, 0));
	net.LoadBitmap("Bitmaps/background_net.bmp", RGB(0, 255, 0));
	net1.LoadBitmap("Bitmaps/background_net1.bmp", RGB(0, 255, 0));
}
void CMap::PlayFlash() {
	flashCounter = flashCounterMax;
}
void CMap::OnMove() {
	delay_counter--;

	if (flashCounter >= 0)flashCounter--;

	if (delay_counter < 0) {
		delay_counter = delay;
		const int STEPS = 27;
		index += 1;

		if (index >= STEPS)index = 0;

		dy = index;
	}
}
void CMap::OnShow() {
	for (int i = 0; i < (SIZE_X / MW) + 1; i++) {
		for (int j = 0; j < SIZE_Y / MH + 27; j++) {
			if (flashCounter >= 0) {
```

```
                net1.SetTopLeft(X + i * MW, Y + j * MH + dy - 27);
                net1.ShowBitmap();
            }
            else {
                net.SetTopLeft(X + i * MW, Y + j * MH + dy - 27);
                net.ShowBitmap();
            }
        }
    }
}

}
```

23. CMe.h

```
#pragma once
#include "CCharacter.h"


namespace game_framework {
// 控制所有的主角 CHARACTER


class CMe {
    public:
        CMe();
        ~CMe();
        void LoadBitmap();                                  // 載入圖形
        void OnMove();                                      // 移動圖形
        void OnShow();                                      // 將圖形圖形貼到畫面
        int  GetX1();                                       // ME 左上角 x 座標
        int  GetY1();                                       // ME 左上角 y 座標
        int  GetX2();                                       // ME 右下角 x 座標
        int  GetY2();                                       // ME 右下角 y 座標
        void SetXY(int nx, int ny);                         // 設定左上角座標
        void LoadCharacter();                               // 載入角色
        void AddSelectedChar(int);                              // 選擇角色時移動游標
        void SetState(int);                                 // 設定當前顯示的狀態

        int GetselectedChar();                              // 取得選中的角色編號
        void SetselectedChar(int);                          // 設定選中的角色編號
        string GetMeName();                                 // 取得選中的角色名
        void SetSelectedChar(string);                       // 設定遊玩中選擇的角色
        //
        void SetHighScoreDisplay(string);                   // 設定 【最高記錄】頁面的角色顯示
        void SetPlayingRecordDisplay(string, string, string);   // 設定 【遊玩記錄】頁面的角色顯示×3
        //
        void WriteUnlockCharacter(string);                  // 寫入已解鎖的角色
        void ReadUnlockCharacter();                         // 讀已解鎖的角色
        bool JudgeUnlock(int, int, int, int);
      // 帶入本次遊玩的成績，查看是否符合解鎖條件，並回傳一個 bool
        bool GetSelectedCharIsUnlock();                     // 回傳選定的角色是否已解鎖

    protected:
        vector<CCharacter*> character;
        CFile*                          file;
        string highScoreName, playingRecordName[3];
        CMovingBitmap           unlockSign, unlock_border;

        const int CHARACTER_POS_Y;
        int x, y;                                           // 圖形左上角座標
        int  selectedChar;                                  // 選中的角色編號
        int  currState;
      // 0: 遊戲中, 1: 選擇角色畫面, 2: GAMEOVER 畫面, 3: 最高記錄畫面, 4: 遊玩記錄畫面
        int highScoreCharNum, playingRecordCharNum[3];
};
}
```

24. CMe.cpp

```
#include "stdafx.h"
#include "Resource.h"
```

```
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include <sstream>
#include <fstream>
#include "CEnemy.h"
#include "CMe.h"
#include "CRecord.h"


namespace game_framework {
// CMe: 控制顯示主角的中樞


CMe::CMe(): CHARACTER_POS_Y(320) {
	x = 100;
	y = SIZE_Y - 60;
	currState = 1;
	selectedChar = 0;
	highScoreName = "";
	highScoreCharNum = 0;
}
CMe::~CMe() {
	for (CCharacter* cc : character) delete cc;
}


void CMe::LoadBitmap() {
	unlockSign.LoadBitmap("Bitmaps/me/me_unlock.bmp", RGB(0, 255, 0));
	unlock_border.LoadBitmap("Bitmaps/menu/character/character_unlock_border.bmp", RGB(0, 255, 0));
	LoadCharacter();
}

void CMe::WriteUnlockCharacter(string name) {
	fstream	fp;
	fp.open("user/unlock.txt", ios::out | ios::app);
	fp << name << endl;
	fp.close();
}
void CMe::ReadUnlockCharacter() {
	for (CCharacter* cc : character)
		cc->SetIsUnlock(0);

	//
	fstream	fp;
	fp.open("user/unlock.txt", ios::in);
	char temp[100];

	if (fp) {
		while (!fp.eof()) {
			fp.getline(temp, sizeof(temp));

			for (CCharacter* cc : character) {
				if (cc->GetName() == temp)
					cc->SetIsUnlock(true);
			}
		}
	}

	for (CCharacter* cc : character) TRACE("%d\n", cc->GetIsUnlock());

	fp.close();
}
bool CMe::JudgeUnlock(int ur0, int ur1, int ur2, int ur3) {
	bool atLeastOneUnlockFlag = false;			// 在本次判斷中 至少解鎖了一個新的角色
	int data[4] = { ur0, ur1, ur2, ur3 };		// 將欲判斷的資料 存入陣列

	for (CCharacter* cc : character) {			// 迴圈跑所有角色 vector
```

```
            bool allGoodToUnlock = true;                    // 先設定“全達解鎖條件”為 true

            for (int i = 0; i < 4; i++) {                   // 迴圈跑 4 項解鎖條件
                if (!(data[i] >= cc->GetUnlockRequirement(i)))
                    allGoodToUnlock = false;                // 若其中一項不符合“全達解鎖條件”便設定為 false
            }

            if (allGoodToUnlock && !cc->GetIsUnlock()) {    // 若“全達解鎖條件”且 尚未被解鎖過 則解鎖該角色
                WriteUnlockCharacter(cc->GetName());        // 解鎖該角色手段為 寫入 txt
                atLeastOneUnlockFlag = true;                // 設定“至少解鎖一角色為 true”
            }
        }

        ReadUnlockCharacter();
        return atLeastOneUnlockFlag;
    }
    bool CMe::GetSelectedCharIsUnlock() {
        return character[selectedChar]->GetIsUnlock();
    }

    void CMe::LoadCharacter() {
        // 解鎖條件(依序)： 1.累計總正確按鍵數 2.單場分數 3.單場正確率 4.單場達到最高關卡
        character.push_back(new CCharacter("Iron Man","鋼鐵人",   "me_ironman", 2, 0, 0, 0, 0));
        character.push_back(new CCharacter("Captain American",    "美利堅隊長","me_captain_american", 1,0, 300, 92, 0));
        character.push_back(new CCharacter("Hulk",  "浩克","me_hulk", 1,   0, 600, 87, 0));
        character.push_back(new CCharacter("Creeper","苦力怕","me_creeper", 1, 0, 1000, 80, 0));
        character.push_back(new CCharacter("Cow","牛","me_cow", 1,500, 0, 0, 0));
        character.push_back(new CCharacter("Minion","小小兵","me_minion", 1,0, 1500, 80, 10));
        character.push_back(new CCharacter("Doge",  "狗狗","me_doge",2,0, 2000, 78, 0));
        character.push_back(new CCharacter("Tsai Ing-wen",  "蔡英文","me_Ing_wen", 10,0, 3000, 75, 0));
        character.push_back(new CCharacter("Donald Trump","川普","me_trump", 1,0, 5000, 70, 0));
        character.push_back(new CCharacter("Bouncing Ball","跳動的球","me_ball", 4,0, 0, 97, 5));
        character.push_back(new CCharacter("Eraser","擦子","me_eraser", 4,0, 0, 92, 15));
        character.push_back(new CCharacter("Zombie Brain","殭屍腦","me_zombie", 1,1000, 0, 0, 0));
        character.push_back(new CCharacter("Pikachu","皮卡撐","me_pikachu", 2,1500, 0, 0, 0));
        character.push_back(new CCharacter("Mike Wazowski","反霸凌博士","me_mike", 2,2000, 0, 0, 0));
        character.push_back(new CCharacter("Mushroom","馬力歐裡面的菇", "me_mushroom", 1,2500, 0, 0, 0));
        character.push_back(new CCharacter("Kirby","卡比","me_kirby",6,3000, 0, 0, 0));
        character.push_back(new CCharacter("Finn","阿寶","me_finn", 1,4000, 0, 0, 0));
        character.push_back(new CCharacter("Flappy Bird","像素鳥","me_flappy_bird", 2,5000, 0, 0, 0));
        character.push_back(new CCharacter("ROC","中華民國","me_roc", 1,10000, 0, 0, 0));
        character.push_back(new CCharacter("PRC","中華人民共和國", "me_prc", 1,34260, 0, 0, 0));
        character.push_back(new CCharacter("Japan", "日本","me_japan", 1,50000, 0, 0, 0));
    }

    void CMe::OnMove() {
        if (currState == 0) character[selectedChar]->OnMove();
        else if (currState == 1) character[selectedChar]->OnMove();
        else if (currState == 3) {
            character[highScoreCharNum]->OnMove();
        }
        else if (currState == 4) {
            for (int i = 0; i < 3; i++) {
                if (playingRecordCharNum[i] != 999)
                    character[playingRecordCharNum[i]]->OnMove();
            }
        }
    }

    void CMe::OnShow() {
        if (currState == 0) {                                   // 遊戲中顯示
            x = (SIZE_X - character[selectedChar]->GetWidth()) / 2;
            y = SIZE_Y - 60;
            character[selectedChar]->SetXY(x, y);
            character[selectedChar]->OnShow();
        }
        else if (currState == 1) {                              //選擇角色畫面
            x = (SIZE_X - character[selectedChar]->GetWidth()) / 2;
            y = CHARACTER_POS_Y + 70;
```

```
        if (character[selectedChar]->GetIsUnlock()) {               // 中間
            character[selectedChar]->SetXY(x, y);
            character[selectedChar]->OnShow();
        }
        else {
            unlockSign.SetTopLeft((SIZE_X - unlockSign.Width()) / 2, y);
            unlockSign.ShowBitmap();
        }

        for (unsigned int i = 1; i <= 4; i++) {
            const int PADDING_CENTER = 40, PADDING_EACH = 80;
            int position = selectedChar + i;

            if (position >= 0 && position < int(character.size()) ) {       // 右邊
                if (character[selectedChar + i]->GetIsUnlock()) {       // 若已被解鎖
                    character[selectedChar + i]->SetXY(x + PADDING_CENTER + PADDING_EACH * i, y);
                    character[selectedChar + i]->OnShow();
                }
                else {                                                  // 尚未被解鎖
                    unlockSign.SetTopLeft((SIZE_X - unlockSign.Width()) / 2 + PADDING_CENTER +
PADDING_EACH * i, y);
                    unlockSign.ShowBitmap();
                }
            }

            position = selectedChar - i;

            if (position >= 0 && position < int(character.size()))          {       // 左邊
                if (character[selectedChar - i]->GetIsUnlock()) {
                    character[selectedChar - i]->SetXY(x - PADDING_CENTER - PADDING_EACH * i, y);
                    character[selectedChar - i]->OnShow();
                }
                else {
                    unlockSign.SetTopLeft((SIZE_X - unlockSign.Width()) / 2 - PADDING_CENTER - PADDING_EACH
* i, y);
                    unlockSign.ShowBitmap();
                }
            }
        }

        ////
        if (character[selectedChar]->GetIsUnlock()) {
            CDC* pDC = CDDraw::GetBackCDC();
            CFont f, *fp;
            f.CreatePointFont(100, "新細明體");
            fp = pDC->SelectObject(&f);
            pDC->SetBkColor(RGB(0, 90, 130));
            pDC->SetBkMode(TRANSPARENT);
            char temp[20];
            pDC->SetTextColor(RGB(200, 200, 200));
            string name;
            name = character[selectedChar]->GetIsUnlock() ? character[selectedChar]->GetName() : "? ? ?";
            sprintf(temp, "%s", name.c_str());
            pDC->TextOut(350, CHARACTER_POS_Y + 60 + 60, temp);
            pDC->SetTextColor(RGB(255, 200, 15));
            name = character[selectedChar]->GetIsUnlock() ? character[selectedChar]->GetSubName() : "? ? ?";
            sprintf(temp, "%s", name.c_str());
            pDC->TextOut(350, CHARACTER_POS_Y + 60 + 74, temp);
            pDC->SelectObject(fp);                                      // 放掉 font f (千萬不要漏了放掉)
            CDDraw::ReleaseBackCDC();                                       // 放掉 Back Plain 的 CDC
            ////
        }
        else {                                                          // 顯示解鎖條件
            const int UNLOCK_BORDER_X = 470, UNLOCK_BORDER_Y = 280;
            unlock_border.SetTopLeft(UNLOCK_BORDER_X, UNLOCK_BORDER_Y);
            unlock_border.ShowBitmap();
            //
            CDC* pDC = CDDraw::GetBackCDC();
```

```
                CFont f, *fp;
                f.CreatePointFont(80, "新細明體");
                fp = pDC->SelectObject(&f);
                pDC->SetBkMode(TRANSPARENT);
                char temp[100];
                pDC->SetTextColor(RGB(200, 200, 200));
                int ulrPrintNum = 0;
                // 解鎖條件(依序)： 1.累計總正確按鍵數 2.單場分數 3.單場正確率 4.單場達到最高關卡

                for (int i = 0; i < 4; i++) {
                    int num = character[selectedChar]->GetUnlockRequirement(i);

                    if (i == 0)    sprintf(temp, "%s%d%s", "累計總按鍵數達 ", num, " 次");
                    else if (i == 1) sprintf(temp, "%s%d%s", "單場分數達 ", num, " 分");
                    else if (i == 2) sprintf(temp, "%s%d%s", "單場正確率高於 ", num, " %");
                    else if (i == 3) sprintf(temp, "%s%d%s", "單場達到第 ", num, " 關卡");

                    if (num != 0) {
                        if (i == 0) pDC->SetTextColor(RGB(255, 200, 15));
                        else        pDC->SetTextColor(RGB(200, 200, 200));

                        pDC->TextOut(UNLOCK_BORDER_X + 20, UNLOCK_BORDER_Y + 25 + ulrPrintNum * 14,
temp);
                        ulrPrintNum++;
                    }
                }

                //
                pDC->SelectObject(fp);                                  // 放掉 font f (千萬不要漏了放掉)
                CDDraw::ReleaseBackCDC();                                       // 放掉 Back Plain 的 CDC
            }
        }

        if (currState == 2) {                                           // 遊戲結束顯示
            x = (SIZE_X - character[selectedChar]->GetWidth()) / 2 + 200;
            y =   400;
            character[selectedChar]->SetXY(x, y);
            character[selectedChar]->OnShow();
        }

        if (currState == 3) {                                           // 最高紀錄顯示
            x = 294;
            y = 420;
            character[highScoreCharNum]->SetXY(x - character[highScoreCharNum]->GetWidth() / 2, y );
            character[highScoreCharNum]->OnShow();
        }

        if (currState == 4) {                                           // 遊玩紀錄顯示
            x = 160;
            y = 400;
            const int LINE_MARGIN = 44;                                 // 角色間的 Y 軸距離

            for (int i = 0; i < 3; i++) {
                if (playingRecordCharNum[i] != 999) {
                    character[ playingRecordCharNum[i] ]->SetXY(x - character[playingRecordCharNum[i]]->GetWidth() / 2, \
                            y + LINE_MARGIN * i);
                    character[ playingRecordCharNum[i] ]->OnShow();
                }
            }
        }

        if (currState == 5) {}
}

void CMe::AddSelectedChar(int num) {
        int result = selectedChar + num;

        if (result >= 0 && result < int(character.size()) ) selectedChar = result;
}
```

```
void CMe::SetSelectedChar(string characterName) {
	selectedChar = 0;

	for (unsigned int i = 0; i < character.size(); i++) {
		if (characterName == character[i]->GetName()) {
			selectedChar = i;
			break;
		}
	}
}
void CMe::SetXY(int nx, int ny) {
	x = nx;
	y = ny;
}
void CMe::SetHighScoreDisplay( string characterName) {
	this->highScoreName = characterName;

	for (unsigned int i = 0; i < character.size(); i++) {
		if (highScoreName == character[i]->GetName()) {
			highScoreCharNum = i;
			break;
		}
	}
}
void CMe::SetPlayingRecordDisplay(string s0, string s1, string s2) {
	playingRecordName[0] = s0;
	playingRecordName[1] = s1;
	playingRecordName[2] = s2;

	for (int j = 0; j < 3; j++) {
		for (unsigned int i = 0; i < character.size(); i++) {
			if (playingRecordName[j] == character[i]->GetName()) {   // 若輸入的角色名和 Vector 內匹配
				playingRecordCharNum[j] = i;
				break;
			}
			else if (playingRecordName[j] == "") {                    // 不顯示主角 Bitmap
				playingRecordCharNum[j] = 999;                        // 999 表示不顯示主角 Bitmap
				break;
			}
		}
	}
}
int CMe::GetX1() {
	return character[selectedChar]->GetX1();
}
int CMe::GetY1() {
	return character[selectedChar]->GetY1();
}
int CMe::GetX2() {
	return character[selectedChar]->GetX2();
}
int CMe::GetY2() {
	return character[selectedChar]->GetY2();
}
void CMe::SetState(int state) {
	currState = state;
}
int CMe::GetselectedChar() {
	return selectedChar;
}
string CMe:: GetMeName() {
	return character[selectedChar]->GetName();
}
void CMe::SetselectedChar(int selectedChar) {
	this->selectedChar = selectedChar;
}
}
```

25. CRecord.h

```
#pragma once
#include "CMe.h"
namespace game_framework {

class CRecord {
	public:
		CRecord(int score, int level, double accuracy, string meName, int correctKeyCount, string date);
		int ReadRecordScore_Score();                                    //回傳第 x 筆裡面的分
		int ReadRecordScore_Level();                                    //回傳第 x 筆裡面的關
		double ReadRecordScore_Accuracy();                              //回傳第 x 筆裡面的正確率
		string	ReadRecordScore_Character();                            //回傳第 x 筆裡面的角色名
		string	ReadRecordScore_Date();                                 //回傳第 x 筆裡面的日期
		int		ReadRecordScore_selectedChar();                         //回傳第 x 筆裡面的主角編號
		int		ReadRecordScore_CorrectKeyCount();                       //回傳第 x 筆裡面的正確按鍵數量
	private:
		int score;                                  //得分
		int level;                                  //關卡
		double accuracy;                            //正確率
		string date;                                //日期
		string meName;                              //主角名稱(英文)
		int	selectedChar;                           //主角編號
		char chDate[80];                            //char 型態的日期
		int correctKeyCount;                         //正確按鍵數量

};
}
```

26. CRecord.cpp

```
#pragma warning( disable : 4996 )
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "CRecord.h"
#include <iomanip>
#include <fstream>
#include <ctime>
#include <string>
#include <sstream>

namespace game_framework {

CRecord::CRecord(int score, int level, double accuracy, string meName, int correctKeyCount, string date) {
	this->score = score;
	this->level = level;
	this->accuracy = accuracy;
	this->meName = meName;
	this->correctKeyCount = correctKeyCount;
	this->date = date;
}

int CRecord::ReadRecordScore_Score() {
	return score;
}
int CRecord::ReadRecordScore_Level() {
	return level;
}
double CRecord::ReadRecordScore_Accuracy() {
	return accuracy;
}
string CRecord::ReadRecordScore_Character() {
	return meName;
}
string CRecord::ReadRecordScore_Date() {
	return date;
}
int CRecord::ReadRecordScore_selectedChar() {
```

```
	return selectedChar;
}
int CRecord::ReadRecordScore_CorrectKeyCount() {
	return correctKeyCount;
}

}
```

27. myGame.h

```
#pragma once
#include "CCharacter.h"
#include "CMe.h"
#include "CDict.h"
#include "CBomb.h"
#include "CEnemy.h"
#include "CBullet.h"
#include "CMap.h"
#include "CBossA.h"
#include "CBossB.h"
#include "CEmp.h"
#include "CLevel.h"
#include "CRecord.h"
#include "CFile.h"

namespace game_framework {
// Constants
enum AUDIO_ID {				// 定義各種音效的編號
	AUDIO_DING,				// 0
	AUDIO_LAKE,				// 1
	AUDIO_NTUT,				// 2
	//=========================================
	AUDIO_ROCK,				// 3
	AUDIO_SHOT,				// 4
	AUDIO_ERROR,				// 5
	AUDIO_CONGRATULATION,		// 6
	AUDIO_CONGRATULATION2,		// 6
	AUDIO_GAMEOVER				// 7

};

// 這個 class 為遊戲的遊戲開頭畫面物件
// 每個 Member function 的 Implementation 都要弄懂
class PublicData {
	public:
		static int			score;				// 共用變數：儲存分數
		static int			level;				// 共用變數：儲存關卡
		static double		accuracy;			// 共用變數：正確率
		static CMe			me;					// 共用變數：主角參數
		static CFile		file;

		static int			CorrectKeyCount;	// 共用變數：正確按鍵數
		static int			totalKeyCount;		// 共用變數：正確按鍵數(會累積的)
		static bool			musicOnOff;			// 共用變數：音樂開關
		static bool			newUnlock;			// 共用變數：是否有新的解鎖角色

		static bool			debugMode;
};

class CGameStateInit : public CGameState {
	public:
		CGameStateInit(CGame* g);
		~CGameStateInit();
		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 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 OnMouseMove(UINT nFlags, CPoint point);	// 處理滑鼠的動作
		void OnMove();
```

```
    protected:
        void OnShow();                                          // 顯示這個狀態的遊戲畫面
    private:
        const int NOTE_TEXT_X, NOTE_TEXT_Y;                        // 定義 遊戲說明 擺放的位置
        const int MENU_Y;                                           // 定義 MENU 的 Y 軸
        const int MENU_ITEM_NUM;                                // 定義 MENU 項目的數量
        const int CHARACTER_Y;                                      // 定義 角色選擇 之 角色 Y 軸
        CMovingBitmap typing_logo, taipin;                      // typing typing 精美的 LOGO
        CMovingBitmap text1;                                    // 說明文字
        int text1_y, text1_count;                               // 說明文字移出效果
        CMap map;                                               // 背景
        int currSelectItem, displayState;                       // 當前選擇的 MENU, 當前顯示的狀態
        int noteDisplayState, statsDisplayState, aboutDisplayState;     // 當前顯示的說明狀態
        int statsPRItemNum;                                     // 遊玩記錄的項目數字
        int wrongKeyNum;                                        // 錯誤按鍵數
        int exitGameCount;                                      // 關閉遊戲計數
        bool exitState = false;
                                                                    //主選單元素
        vector<CMovingBitmap*>menuText;                             // 選單文字 VECTOR
        CMovingBitmap       menuBorder, menuBorder_ckecked;     // 頁面指示燈
        CMovingBitmap       userBorder, highScoreBorder ;       // 最高分的框
        CMovingBitmap       numBmp[10], numBmpSmall[14];    // 數字圖檔
        CMovingBitmap       numBmp_White[10], numBmpSmall_White[14];    // 數字圖檔（白色）
        CMovingBitmap       new_text;
        CMovingBitmap       exit;
        // 遊戲說明 元素
        CAnimation          noteExkey;                              // 遊戲說明裡面的 打字動畫
        CMovingBitmap       noteBorder, noteArrow;                  // 框線, 箭頭
        CMovingBitmap       noteSelected, noteUnselected;           // 指示燈
        vector<CMovingBitmap*>note;                                 // 多頁的說明文字
        // 角色選擇 元素
        CMovingBitmap       characterBorder, characterArrow;        // 角色選擇框 箭頭
        // 統計        元素
        CMovingBitmap       statsBorder, statsBg[2];            // 統計頁面框, 統計頁面左右頁
        CMovingBitmap       statsArrow[3], statsArrowV[4];      // 左右箭頭, 上下四種狀態箭頭
        CMovingBitmap       statsText [4];                      // 左頁項目文字
        CMovingBitmap       statsNoRecord;                      // 無記錄 文字
        // 介紹頁面 元素
        string                  cheatCode;                      // 儲存作弊碼
        CMovingBitmap       aboutBorder, about;                 // 關於框, 關於文字
        CMovingBitmap       delText;                            // 確認刪除文字
        CMovingBitmap       musicOnOff[2];

};

// 這個 class 為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
// 每個 Member function 的 Implementation 都要弄懂

class CGameStateRun : public CGameState {
    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
    protected:
        void OnMove();                                          // 移動遊戲元素
        void OnShow();                                          // 顯示這個狀態的遊戲畫面
    private:
        bool            showDebug = false;                      // 是否顯示 debug 資訊
        bool            quickCall = false;                      // 是否開啟快速召喚
        CMovingBitmap   pauseText;                              // 暫停視窗文字
        CMovingBitmap   debugText;                              // debug 提示文字
```

```
        CAnimation          target;                                     // 鎖定的動畫
        vector<CMovingBitmap*>letter;
        vector<CEnemy*> enemyQueue;                                     // 儲存所有敵人的 Vector
        vector<CBullet*>bulletList;                                         // 儲存飛行中的子彈的 Vector
        vector<CBomb*>   bombList;                                      // 儲存爆炸效果的 vector
        //
        CEnemy*             targetEnemy;                                // 指標 用於指向瞄準的敵人
        CDict               dictionary;                                 // 所有怪物共用的字典
        CMap                map;                                        // 背景圖
        CInteger            score;                                      // 分數顯示器
        CEmp                emp;                                        // 電磁波
        CLevel              levelAni;                                   // 切換關卡時的動畫
        //
        const int           LEVEL;                              // 關卡總數
        char                key;                                // 記錄所按下的按鍵 用於防止彈跳
        bool                lock;                               // 判斷是否鎖住第一個字母了
        bool                pause;
        int                 lives;                              // 生命值
        int                 levelEnemyNum[30] = { 4,  5,  5,  6,  7,  7,  7,  7,  8,  9, 10, 10, 10, 11, 11, 11, 12, 12, 12,
13, 13, 13, 14, 14, 14, 14, 15, 15, 15, 17 };                       // 該關卡最大的敵人數
        int                 levelBossANum[30] = { 0,  0,  1,  1,  1,  2,  1,  1,  1,  2,  1,  2,  2,  2,  3,  3,  3,
3,  3,  3,  4,  4,  4,  5,  5,  5,  5,  5,  5,  7 };
        int                 levelBossBNum[30] = { 0,  0,  0,  0,  0,  0,  1,  2,  2,  1,  2,  1,  2,  2,  2,  2,
2,  2,  2,  3,  3,  3,  3,  3,  3,  3,  4,  4,  4,  4 };
        int                 callEnemyCounter, maxCallEnemyCounter;
// 召喚怪物間隔計數器, 召喚怪物間隔; maxCallEnemyCounter 決定怪物生成速度 越小速度越快
        int                 callBossACounter, maxCallBossACounter;
        int                 callBossBCounter, maxCallBossBCounter;
        int                 currEnemyNum;                       // 當前該關卡 已召喚的敵人數量
        int                 currBossANum, currBossBNum;         // 當前該關卡 已召喚的 BossA & BossB 數量
        int                 currLevel;                          // 當前關卡
        int                 totalEnemyNum;                      // 總召喚的敵人數量
        int                 levelChangeFlag, levelChangeDelay, levelChangeDelayMax; // 關卡和關卡間的 delay
        int                 totalKeyDownCount;                  // 總按鍵數, 總正確按鍵數
        double              accuracy;                           // 正確率

};

//////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的結束狀態(Game Over)
// 每個 Member function 的 Implementation 都要弄懂
//////////////////////////////////////////////////////////////////////////

class CGameStateOver : public CGameState {
    public:
        CGameStateOver(CGame* g);
        void OnBeginState();                                        // 設定每次重玩所需的變數
        void OnInit();
        void OnKeyUp(UINT, UINT, UINT);
    protected:
        void OnMove();                                              // 移動遊戲元素
        void OnShow();                                              // 顯示這個狀態的遊戲畫面
    private:
        CMovingBitmap       border;                             // 框線
        CMovingBitmap       numBmp[10], numBmpSmall[12];            // 數字圖檔
        CMovingBitmap       bar[2];                                 // 進度條
        CAnimation          newHS_text;                             // 破紀錄文字
        CAnimation          newChar_text;
        bool                isHighScore;                            // 本次遊玩的是否破紀錄
        bool                isUnlock;                               // 本次遊玩的是否解鎖新角色
        int                     counter;                            // 倒數之計數器
        int                     x, y;                               // 圖檔顯示位置
        int                     score, level;                       // 分數, 關卡
        double                  accuracy;                           // 正確率
        int                     barCounter;                         // 進度條的計數器

};

}
```

28. myGame.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <time.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"

namespace game_framework {
/////////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的遊戲開頭畫面物件
/////////////////////////////////////////////////////////////////////////////

int PublicData::score = 0;
int PublicData::level = 0;
int PublicData::CorrectKeyCount = 0;
int PublicData::totalKeyCount = 0;
double PublicData::accuracy = 0.0;
bool PublicData::musicOnOff = 1;
bool PublicData::newUnlock = false;
bool PublicData::debugMode = false;
CFile PublicData::file;
CMe    PublicData::me;

CGameStateInit::CGameStateInit(CGame* g)
    : CGameState(g), NOTE_TEXT_X(60), NOTE_TEXT_Y(280), MENU_Y(320),
      MENU_ITEM_NUM(5), CHARACTER_Y(320) {
}

CGameStateInit::~CGameStateInit() {
    for (CMovingBitmap* mt : menuText) delete mt;

    for (CMovingBitmap* nt : note) delete nt;
}
void CGameStateInit::OnInit() {
    ShowInitProgress(0);                                        // 一開始的 loading 進度為 0%
    currSelectItem = displayState = 0;                          // 初始化選單選取項目
    noteDisplayState = statsDisplayState = aboutDisplayState = 0;          // 初始化"遊戲說明"及"統計"選取頁面項目
    statsPRItemNum = 0;                                         // 初始化統計頁面 最高記錄的選取項目
    wrongKeyNum = 0;
    exitGameCount = 0;
    cheatCode = "";

    PublicData::me.ReadUnlockCharacter();
    PublicData::me.LoadBitmap();                                          // 主角
    PublicData::me.SetSelectedChar(PublicData::file.ReadSelectedCharacter());     // 讀取上次選取的角色
    PublicData::musicOnOff = PublicData::file.ReadMusicOnOff();                 // 讀取上次音樂的狀態
    map.LoadBitmap();                                                     // 背景網狀動畫
    typing_logo.LoadBitmap("Bitmaps/start_logo1.bmp", RGB(0, 255, 0));    // logo
    taipin.LoadBitmap("Bitmaps/taipin.bmp", RGB(0, 255, 0));
    text1.LoadBitmap("Bitmaps/text1_start.bmp", RGB(0, 255, 0));          // 操作方式提示 bitmap
    highScoreBorder.LoadBitmap("Bitmaps/menu/highscore_border.bmp", RGB(0, 255, 0));
    new_text.LoadBitmap("Bitmaps/menu/menu_new_text.bmp", RGB(0, 255, 0));
    /////
    //
    // 載入選單元素
    ShowInitProgress(5);

    for (int i = 0; i < 5; i++) {                 // 5 個選單
        char str[50];
        sprintf(str, "Bitmaps/menu/menu_t_%de.bmp", i + 1);
        menuText.push_back(new CMovingBitmap);
        menuText.back()->LoadBitmap(str, RGB(0, 255, 0));
    }

    ShowInitProgress(10);
```

```
menuText.push_back(new CMovingBitmap);
menuText.back()->LoadBitmap("Bitmaps/menu/menu_t_be.bmp", RGB(0, 255, 0)); // 返回按鈕
menuBorder.LoadBitmap("Bitmaps/menu/menu_border.bmp", RGB(0, 255, 0));
menuBorder_ckecked.LoadBitmap("Bitmaps/menu/menu_border_checked.bmp", RGB(0, 255, 0));
                                                                    // 載入遊戲說明元素
noteBorder.LoadBitmap("Bitmaps/menu/note/note_text_border.bmp", RGB(0, 255, 0));      // 說明框線
noteArrow.LoadBitmap ("Bitmaps/menu/note/note_text_arraw.bmp", RGB(0, 255, 0));       // 說明箭頭
noteUnselected.LoadBitmap("Bitmaps/menu/note/note_unselected.bmp", RGB(0, 255, 0));
noteSelected.LoadBitmap("Bitmaps/menu/note/note_selected.bmp", RGB(0, 255, 0));
exit.LoadBitmap("Bitmaps/menu/exit.bmp", RGB(0, 255, 0));
ShowInitProgress(15);

for (int i = 0; i < 6; i++) {                                       // 說明框裡面的按鍵動畫
	char str[50];
	sprintf(str, "Bitmaps/menu/note/note1_exkey_%d.bmp", i + 1);
	noteExkey.AddBitmap(str, RGB(0, 255, 0));
}

noteExkey.SetDelayCount(10);

for (int i = 0; i < 9; i++) {                                       // 多頁的說明文字
	char str[50];
	sprintf(str, "Bitmaps/menu/note/note_text_p%d.bmp", i + 1);
	note.push_back(new CMovingBitmap);
	note.back()->LoadBitmap(str, RGB(0, 255, 0));
}

for (int i = 0; i < 10; i++) {                                      // 載入數字圖
	char str[50];
	sprintf(str, "Bitmaps/level/num/%d.bmp", i);
	numBmp[i].LoadBitmap(str, RGB(0, 255, 0));
	sprintf(str, "Bitmaps/level/num_s/%d.bmp", i);
	numBmpSmall[i].LoadBitmap(str, RGB(0, 255, 0));
	sprintf(str, "Bitmaps/level/num_white/%d.bmp", i);
	numBmp_White[i].LoadBitmap(str, RGB(0, 255, 0));
	sprintf(str, "Bitmaps/level/num_s_white/%d.bmp", i);
	numBmpSmall_White[i].LoadBitmap(str, RGB(0, 255, 0));
}

numBmpSmall[10].LoadBitmap("Bitmaps/level/num_s/per.bmp", RGB(0, 255, 0));
numBmpSmall[11].LoadBitmap("Bitmaps/level/num_s/dot.bmp", RGB(0, 255, 0));
numBmpSmall[12].LoadBitmap("Bitmaps/level/num_s/slash.bmp", RGB(0, 255, 0));
numBmpSmall_White[10].LoadBitmap("Bitmaps/level/num_s_white/per.bmp", RGB(0, 255, 0));
numBmpSmall_White[11].LoadBitmap("Bitmaps/level/num_s_white/dot.bmp", RGB(0, 255, 0));
numBmpSmall_White[12].LoadBitmap("Bitmaps/level/num_s_white/slash.bmp", RGB(0, 255, 0));
numBmpSmall_White[13].LoadBitmap("Bitmaps/level/num_s_white/colon.bmp", RGB(0, 255, 0));
ShowInitProgress(20);
// 載入角色選擇 元素
characterBorder.LoadBitmap("Bitmaps/menu/character/character_border.bmp", RGB(0, 255, 0));
characterArrow.LoadBitmap("Bitmaps/menu/character/character_arrow.bmp", RGB(0, 255, 0));
// 載入統計元素
statsBorder.LoadBitmap("Bitmaps/menu/stats/stats_border.bmp", RGB(0, 255, 0));
statsBg[0].LoadBitmap("Bitmaps/menu/stats/stats_bg1.bmp", RGB(0, 255, 0));
statsBg[1].LoadBitmap("Bitmaps/menu/stats/stats_bg2.bmp", RGB(0, 255, 0));
statsArrow[0].LoadBitmap("Bitmaps/menu/stats/stats_arrow.bmp", RGB(0, 255, 0));
statsArrow[1].LoadBitmap("Bitmaps/menu/stats/stats_arrow_right.bmp", RGB(0, 255, 0));
statsArrow[2].LoadBitmap("Bitmaps/menu/stats/stats_arrow_left.bmp", RGB(0, 255, 0));
statsArrowV[0].LoadBitmap("Bitmaps/menu/stats/stats_arrow_v.bmp", RGB(0, 255, 0));
statsArrowV[1].LoadBitmap("Bitmaps/menu/stats/stats_arrow_v_up.bmp", RGB(0, 255, 0));
statsArrowV[2].LoadBitmap("Bitmaps/menu/stats/stats_arrow_v_down.bmp", RGB(0, 255, 0));
statsArrowV[3].LoadBitmap("Bitmaps/menu/stats/stats_arrow_v_none.bmp", RGB(0, 255, 0));

for (int i = 0; i < 4; i++)
	statsArrowV[i].SetTopLeft((SIZE_X - statsArrow[0].Width()) / 2 + 570, NOTE_TEXT_Y + 163);

statsText[0].LoadBitmap("Bitmaps/menu/stats/stats_text_tkc.bmp", RGB(0, 255, 0));
statsText[1].LoadBitmap("Bitmaps/menu/stats/stats_text_hl.bmp", RGB(0, 255, 0));
statsText[2].LoadBitmap("Bitmaps/menu/stats/stats_text_ckc.bmp", RGB(0, 255, 0));
statsText[3].LoadBitmap("Bitmaps/menu/stats/stats_text_acc.bmp", RGB(0, 255, 0));
```

```
	statsNoRecord.LoadBitmap("Bitmaps/menu/stats/stats_no_record.bmp", RGB(0, 255, 0));
	// 載入關於元素
	aboutBorder.LoadBitmap("Bitmaps/menu/about/about_border.bmp", RGB(0, 255, 0));		// 介紹框線
	about.LoadBitmap("Bitmaps/menu/about/about_text_p2.bmp", RGB(0, 255, 0));		// 介紹文字
	delText.LoadBitmap("Bitmaps/menu/about/about_del_text.bmp", RGB(0, 255, 0));		// 確認刪除視窗
	musicOnOff[0].LoadBitmap("Bitmaps/menu/about/about_music_on.bmp", RGB(0, 255, 0));
	musicOnOff[1].LoadBitmap("Bitmaps/menu/about/about_music_off.bmp", RGB(0, 255, 0));
}

void CGameStateInit::OnBeginState() {
	text1_y = 550;
	text1_count = 0;
	PublicData::file.ReadHighScoreFile();
	PublicData::file.ReadRecordFile();
	statsPRItemNum = 0;					// 遊玩記錄的項目數字歸零（回到第一筆資料）
	PublicData::totalKeyCount = PublicData::file.ReadTotalKeyCount();
}

void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
	PublicData::me.ReadUnlockCharacter();
	const char KEY_ESC = 27;
	const char KEY_SPACE = ' ';
	const char KEY_ENTER = 0xD;
	const char KEY_LEFT  = 0x25;				// keyboard 左箭頭
	const char KEY_UP    = 0x26;				// keyboard 上箭頭
	const char KEY_RIGHT = 0x27`				// keyboard 右箭頭
	const char KEY_DOWN  = 0x28;				// keyboard 下箭頭

	if (!(nChar == KEY_ESC || nChar == KEY_LEFT || nChar == KEY_UP || nChar == KEY_RIGHT || nChar == KEY_DOWN || nChar == KEY_ENTER || nChar == 'D' || nChar == 'Y' || nChar == 'N' || (nChar <= '5' && nChar >= '1'))) {
		//wrongKeyNum++;
	}

	if ((nChar <= '5' && nChar >= '1') && displayState == 0) {		// 提供以數字鍵 1～5 來操縱選單
		if (nChar == '1')  GotoGameState(GAME_STATE_RUN);

		currSelectItem = displayState = nChar - '1';
		noteDisplayState = statsDisplayState = aboutDisplayState = 0;
		cheatCode = "";
	}

	if (nChar == KEY_ESC) {							// ESC 鍵...
		if (!(displayState == 0) && !(displayState == 2 && !PublicData::me.GetSelectedCharIsUnlock()))
			displayState = 0;					// 返回主選單
		else if (!(displayState == 2) && !exitState) {
			exitState = true;
			return;
		}
	}

	if (exitState) {
		if (nChar == 'N' || nChar == KEY_ESC)exitState = false;
		else if (nChar == 'Y')		PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}

	if (displayState == 0 ) {						// 在主選單...
		if (nChar == KEY_UP || nChar == KEY_DOWN) {			// 移動光標
			if (nChar == KEY_UP) currSelectItem--;
			else if (nChar == KEY_DOWN) currSelectItem++;

			if (currSelectItem < 0)currSelectItem = MENU_ITEM_NUM - 1;
			else if (currSelectItem > MENU_ITEM_NUM - 1)currSelectItem = 0;
		}
		else if (nChar == KEY_ENTER) {				// 按下 ENTER 鍵選取...
			if (currSelectItem == 0) {
				GotoGameState(GAME_STATE_RUN);			// 開始遊戲
			}
			else {
```

```
                displayState = currSelectItem;                          // 前往所選取的 state
                noteDisplayState = statsDisplayState = aboutDisplayState = 0;
    // 初始化說明文字的選取項目 及 遊玩記錄的項目數字
                cheatCode = "";

                if (currSelectItem == 2) PublicData::newUnlock = false;
            }
        }
    }
    else if (displayState == 1 ) {                                      // [遊戲說明]
        if (nChar == KEY_ENTER) displayState = 0;                       //  ->返回主選單
        else if (nChar == KEY_LEFT || nChar == KEY_RIGHT) {             // [遊戲說明] 左右翻頁遊戲說明
            if      (nChar == KEY_LEFT)     noteDisplayState--;
            else if (nChar == KEY_RIGHT) noteDisplayState++;

            if (noteDisplayState < 0) noteDisplayState = int(note.size()) - 1;
            else if (noteDisplayState >    int(note.size()) - 1) noteDisplayState = 0;
        }
    }
    else if (displayState == 2) {                                       // [角色選擇]
        if (nChar == KEY_ENTER) {
            if (PublicData::me.GetSelectedCharIsUnlock()) {
                PublicData::file.WriteSelectedCharacter(PublicData::me.GetMeName());
                displayState = 0;                                       // ->返回主選單
            }
        }
        else if (nChar == KEY_LEFT || nChar == KEY_RIGHT) {
            if      (nChar == KEY_LEFT)     PublicData::me.AddSelectedChar(-1);
            else if (nChar == KEY_RIGHT)    PublicData::me.AddSelectedChar(1);
        }
    }
    else if (displayState == 3) { // [統計]
        if (nChar == KEY_ENTER) displayState = 0;                       // ->返回主選單
        else if (nChar == KEY_LEFT || nChar == KEY_RIGHT) {             // [統計] 左頁:最高記錄; 右頁:遊玩
記錄
            if      (nChar == KEY_LEFT)         statsDisplayState = 0;
            else if (nChar == KEY_RIGHT) statsDisplayState = 1;
        }

        if (statsDisplayState == 1) {                                   // 若為 遊玩記錄狀態
            if      (nChar == KEY_UP) {                                 // 向上查找記錄
                if (statsPRItemNum > 0)statsPRItemNum--;
            }
            else if (nChar == KEY_DOWN) {                               // 向下查找記錄
                if (statsPRItemNum < PublicData::file.GetRecordNum() - 3)statsPRItemNum++;
            }
        }
    }
    else if (displayState == 4 ) {                                      // [關於]
        if (aboutDisplayState == 0 && nChar == KEY_ENTER) displayState = 0;      // ->返回主選單

        if (aboutDisplayState == 0) {
            cheatCode = cheatCode + char(nChar);

            if (cheatCode == "104590029") {                             // 作弊碼:解鎖所有角色
                PublicData::me.JudgeUnlock(99999, 99999, 100, 30);
                displayState = 0;
                cheatCode = "";
            }
            else if (cheatCode == "104590025") {                        // 作弊碼:允許 debug 模式
                PublicData::debugMode = true;
                displayState = 0;
                cheatCode = "";
            }
        }

        if (aboutDisplayState == 0 && nChar == 'M') {                   // 開關音樂
            PublicData::musicOnOff = PublicData::musicOnOff ? false : true;
            PublicData::file.WriteMusicOnOff(PublicData::musicOnOff);
```

```
        }

        if (aboutDisplayState == 0 && nChar == 'D' ) {
            aboutDisplayState = 1;                                  // 顯示清除遊玩紀錄視窗
        }

        if (aboutDisplayState == 1) {
            if (nChar == 'Y') {                                     // 選 YES 確認刪除遊戲紀錄
                PublicData::file.DeleteAllData();                       // 清空 txt 檔
                PublicData::file.ReadHighScoreFile();               // 重新載入遊戲紀錄
                PublicData::file.ReadRecordFile();
                PublicData::me.SetSelectedChar("Iron Man");         // 重置選取的角色為 IronMan
                displayState = 0;                                   // 返回主選單
            }
            else if (nChar == 'N') {                                    // 選 NO 取消
                aboutDisplayState = 0;
            }
        }
    }
}

void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point) {
    wrongKeyNum++;
}
void CGameStateInit::OnMouseMove(UINT nFlags, CPoint point) {}
void CGameStateInit::OnMove() {
    map.OnMove();
    noteExkey.OnMove();
    PublicData::me.OnMove();

    if (exitGameCount != 0 && exitGameCount < 15)
        exitGameCount++;
    else
        exitGameCount = 0;

    if (text1_count < 400) text1_count++;

    if (text1_count > 5 * 30)    text1_y += int((text1_count - 5 * 30) * 1.1);

    if (wrongKeyNum > 3) {                                  // 若多次按下不相關按鈕，則友善提醒重新播放說明文字
        text1_y = 550;
        text1_count = wrongKeyNum = 0;
    }

    if (0) GotoGameState(GAME_STATE_OVER);
}
void CGameStateInit::OnShow() {
    map.OnShow();
    typing_logo.SetTopLeft((SIZE_X - typing_logo.Width()) / 2, 100);
    typing_logo.ShowBitmap();
    taipin.SetTopLeft((SIZE_X + typing_logo.Width()) / 2 - 60, 100 + 31);
    taipin.ShowBitmap();

    if (displayState == 0) {                                // 顯示主選單
        if (!exitState) {
            const int MENU_MARGIN_BTM = 40;
            menuBorder_ckecked.SetTopLeft((SIZE_X - menuBorder.Width()) / 2, MENU_Y + MENU_MARGIN_BTM *
currSelectItem);
            menuBorder_ckecked.ShowBitmap();

            for (int i = 0; i < MENU_ITEM_NUM; i++) {
                menuBorder.SetTopLeft((SIZE_X - menuBorder.Width()) / 2, MENU_Y + MENU_MARGIN_BTM * i);
                menuBorder.ShowBitmap();
                menuText[i]->SetTopLeft((SIZE_X - menuText[i]->Width()) / 2, MENU_Y + 7 + MENU_MARGIN_BTM
* i);
                menuText[i]->ShowBitmap();
            }

            int HIGHSCORE_POS_X = (SIZE_X + menuBorder.Width()) / 2 + 8;
```

```
            int HIGHSCORE_POS_Y = MENU_Y + 10;

            if (PublicData::file.isHighScoreExist()) {                    // 顯示最高分(bitmap)
                int tempScore = PublicData::file.ReadHighScore_Score();
                highScoreBorder.SetTopLeft(HIGHSCORE_POS_X, HIGHSCORE_POS_Y);
                highScoreBorder.ShowBitmap();

                for (int i = 0; i < 5; i++) {                                 // 顯示分數數字 bmp
                    numBmpSmall[tempScore % 10].SetTopLeft( HIGHSCORE_POS_X - 10 * i + 130,
HIGHSCORE_POS_Y + 5);
                    numBmpSmall[tempScore % 10].ShowBitmap();
                    tempScore /= 10;
                }
            }

            if (PublicData::newUnlock) {
                new_text.SetTopLeft((SIZE_X - menuBorder.Width()) / 2 + 130, MENU_Y + MENU_MARGIN_BTM * 2);
                new_text.ShowBitmap();
            }
        }
        else {
            exit.SetTopLeft((SIZE_X - exit.Width()) / 2, (NOTE_TEXT_Y + exit.Height() / 2));
            exit.ShowBitmap();
        }
    }
    else if (displayState == 1) {                                             // 顯示說明文字
        // 說明框線
        noteBorder.SetTopLeft((SIZE_X - noteBorder.Width()) / 2, NOTE_TEXT_Y);
        noteBorder.ShowBitmap();
        // 說明箭頭
        noteArrow.SetTopLeft((SIZE_X - noteArrow.Width()) / 2, NOTE_TEXT_Y + (noteBorder.Height() -
noteArrow.Height()) / 2 + 11);
        noteArrow.ShowBitmap();
        // 說明文字
        note[noteDisplayState]->SetTopLeft((SIZE_X - noteBorder.Width()) / 2, NOTE_TEXT_Y + (noteBorder.Height() -
note[noteDisplayState]->Height() ) / 2 + 11);
        note[noteDisplayState]->ShowBitmap();

        // 說明文字 指示燈
        for (unsigned int i = 0; i < note.size(); i++) {
            noteUnselected.SetTopLeft((SIZE_X - noteBorder.Width()) / 2 + noteBorder.Width() - 25 - 8 * i,
NOTE_TEXT_Y + noteBorder.Height() - 20);
            noteUnselected.ShowBitmap();
        }

        noteSelected.SetTopLeft((SIZE_X - noteBorder.Width()) / 2 + noteBorder.Width() - 25 - 8 * (note.size() -
noteDisplayState - 1), NOTE_TEXT_Y + noteBorder.Height() - 20);
        noteSelected.ShowBitmap();

        // 說明打字動畫
        if (noteDisplayState == 0) {
            noteExkey.SetTopLeft((SIZE_X - noteExkey.Width()) / 2, NOTE_TEXT_Y + 160);
            noteExkey.OnShow();
        }
    }
    else if (displayState == 2) {                                             // 顯示 選擇角色 頁面
        characterBorder.SetTopLeft((SIZE_X - characterBorder.Width()) / 2, CHARACTER_Y);
        characterBorder.ShowBitmap();
        characterArrow.SetTopLeft((SIZE_X - characterArrow.Width()) / 2, CHARACTER_Y + characterBorder.Height() / 2);
        characterArrow.ShowBitmap();
        PublicData::me.SetState(1);
        PublicData::me.OnShow();
    }
    else if (displayState == 3) {                                             // 顯示 統計 頁面
        const int STATS_POS_X = (SIZE_X - statsBorder.Width()) / 2;  // 統計框之位置
        const int STATS_TEXT_POS_Y = 110, STATS_TEXT_POS_X = 310;  // 統計頁文字之位置
        const int STATS_TEXT_MARGIN = 30;                                     // 文字之 行距
        statsBorder.SetTopLeft(STATS_POS_X, NOTE_TEXT_Y);
        statsBorder.ShowBitmap();                                             // 顯示統計頁之框
```

```
        PublicData::me.SetState(statsDisplayState + 3);

        if (statsDisplayState == 0) {        // 左頁 最高記錄
            statsBg[0].SetTopLeft(STATS_POS_X, NOTE_TEXT_Y);
            statsBg[0].ShowBitmap();
            statsArrow[1].SetTopLeft((SIZE_X - statsArrow[0].Width()) / 2, NOTE_TEXT_Y + (statsBorder.Height() -
statsArrow[0].Height()) / 2 + 4);
            statsArrow[1].ShowBitmap();

            //
            //從 CFile 中取得最高分內容並顯示
            if (!PublicData::file.isHighScoreExist()) {                    // 若最高分不存在
                PublicData::me.SetState(5);
                statsNoRecord.SetTopLeft((SIZE_X - statsNoRecord.Width()) / 2 + 115, NOTE_TEXT_Y + 150);
                statsNoRecord.ShowBitmap();
            }
            else {
                int tempScore = PublicData::file.ReadHighScore_Score(),
                    tempTotalKeyCount = PublicData::totalKeyCount,
                    tempLevel = PublicData::file.ReadHighScore_Level(),
                    tempCorrectKeyCount = PublicData::file.ReadHighScore_CorrectKeyCount(),
                    tempAccuracy = int(PublicData::file.ReadHighScore_Accuracy() * 100.0);
                PublicData::me.SetHighScoreDisplay(PublicData::file.ReadHighScore_Character());

                //
                for (int i = 0; i < 5; i++) {                                    // 顯示分數數字 bmp
                    numBmp[tempScore % 10].SetTopLeft(STATS_POS_X + 227 - 20 * i, NOTE_TEXT_Y + 196 );
                    numBmp[tempScore % 10].ShowBitmap();
                    tempScore /= 10;
                }

                for (int j = 0; j < 4; j++) {                                    // 顯示項目文字（高關、按鍵、正確）及數
字
                    statsText[j].SetTopLeft(STATS_POS_X + STATS_TEXT_POS_X, \
                                            NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN
* j);
                    statsText[j].ShowBitmap();                    // 顯示項目文字
                    const int   STATS_TEXT_MARGIN_R = 170;   // 項目文字 與 數字 之距離

                    if (j == 0) {                                    // 0 為顯示 累計總按鍵數
                        for (int i = 0; i < 7; i++) {                // 顯示總按鍵數數字 bmp
                            numBmpSmall[tempTotalKeyCount        %        10].SetTopLeft(STATS_POS_X        +
STATS_TEXT_POS_X + STATS_TEXT_MARGIN_R - 10 * i, \
                                           NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                            numBmpSmall[tempTotalKeyCount % 10].ShowBitmap();
                            tempTotalKeyCount /= 10;

                            if (tempTotalKeyCount == 0)break;
                        }
                    }

                    if (j == 1) {                                    // 1 為顯示 關卡數字
                        for (int i = 0; i < 2; i++) {                // 顯示關卡數字 bmp
                            numBmpSmall_White[tempLevel        %        10].SetTopLeft(STATS_POS_X        +
STATS_TEXT_POS_X + STATS_TEXT_MARGIN_R - 10 * i, \
                                           NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                            numBmpSmall_White[tempLevel % 10].ShowBitmap();
                            tempLevel /= 10;
                        }
                    }
                    else if (j == 2) {                                    // 2 為顯示 總按鍵數
                        for (int i = 0; i < 5; i++) {                // 顯示總按鍵數數字 bmp
                            numBmpSmall_White[tempCorrectKeyCount    %    10].SetTopLeft(STATS_POS_X    +
STATS_TEXT_POS_X + STATS_TEXT_MARGIN_R - 10 * i, \
                                           NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                            numBmpSmall_White[tempCorrectKeyCount % 10].ShowBitmap();
                            tempCorrectKeyCount /= 10;

                            if (tempCorrectKeyCount == 0)break;
```

```
                    }
                }
                else if (j == 3) {                                  // 3 為顯示 正確率
                    if (tempAccuracy != 10000) {                    //若正確率非 100%
                        for (int i = 0, dotPos = 0; i < 4; i++) {   // 顯示正確率 bmp
                            numBmpSmall_White[tempAccuracy % 10].SetTopLeft \
                            (STATS_POS_X + STATS_TEXT_POS_X + STATS_TEXT_MARGIN_R - 10 * i -
dotPos, \
                             NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                            numBmpSmall_White[tempAccuracy % 10].ShowBitmap();
                            tempAccuracy /= 10;

                            if (i == 1) {                           // 顯示小數點
                                dotPos = 5;
                                numBmpSmall_White[11].SetTopLeft\
                                (STATS_POS_X + STATS_TEXT_POS_X + STATS_TEXT_MARGIN_R - 10 * i -
dotPos, \
                                 NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                                numBmpSmall_White[11].ShowBitmap();
                            }

                            tempLevel /= 10;
                        }
                    }
                    else {                                                  // 針對 100%顯示
                        numBmpSmall_White[1].SetTopLeft(STATS_POS_X     +     STATS_TEXT_POS_X     +
STATS_TEXT_MARGIN_R - 10 * 2,NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                        numBmpSmall_White[1].ShowBitmap();
                        numBmpSmall_White[0].SetTopLeft(STATS_POS_X     +     STATS_TEXT_POS_X     +
STATS_TEXT_MARGIN_R – 10,NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                        numBmpSmall_White[0].ShowBitmap();
                        numBmpSmall_White[0].SetTopLeft(STATS_POS_X     +     STATS_TEXT_POS_X     +
STATS_TEXT_MARGIN_R,NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                        numBmpSmall_White[0].ShowBitmap();
                    }

                    numBmpSmall_White[10].SetTopLeft(STATS_POS_X     +     STATS_TEXT_POS_X     +
STATS_TEXT_MARGIN_R + 14, NOTE_TEXT_Y + STATS_TEXT_POS_Y + STATS_TEXT_MARGIN * j + 1);
                    numBmpSmall_White[10].ShowBitmap();                 // 顯示百分比符號
                }
            }
        }
    }
    else if (statsDisplayState == 1) {                                  // 右頁 遊玩記錄
        statsBg[1].SetTopLeft(STATS_POS_X, NOTE_TEXT_Y);
        statsBg[1].ShowBitmap();
        statsArrow[2].SetTopLeft((SIZE_X - statsArrow[0].Width()) / 2, NOTE_TEXT_Y + (statsBorder.Height() -
statsArrow[0].Height()) / 2 + 4);
        statsArrow[2].ShowBitmap();

                                                                        // 設定顯示的箭頭樣式
        if (PublicData::file.GetRecordNum() <= 3 && PublicData::file.GetRecordNum() >= 0) {
            statsArrowV[3].ShowBitmap();                                // 箭頭樣式：全暗
        }
        else {
            if (statsPRItemNum == 0)
                statsArrowV[2].ShowBitmap();                            // 箭頭樣式：下亮
            else if (statsPRItemNum == PublicData::file.GetRecordNum() - 3)
                statsArrowV[1].ShowBitmap();                            // 箭頭樣式：下亮
            else
                statsArrowV[0].ShowBitmap();                            // 箭頭樣式：全亮
        }

        if (PublicData::file.GetRecordNum() == 0) {                     // 當查無遊戲記錄時
            PublicData::me.SetState(5);
            statsNoRecord.SetTopLeft((SIZE_X - statsNoRecord.Width()) / 2, NOTE_TEXT_Y + 163);
            statsNoRecord.ShowBitmap();
        }
```

```
                PublicData::me.SetPlayingRecordDisplay                    // 顯示該筆紀錄的 ME(若不存在則不顯示)
                (statsPRItemNum >= PublicData::file.GetRecordNum() ? "" :
                 PublicData::file.ReadRecord_Character(statsPRItemNum),
                 statsPRItemNum + 1 >= PublicData::file.GetRecordNum() ? "" :
                 PublicData::file.ReadRecord_Character(statsPRItemNum + 1),
                 statsPRItemNum + 2 >= PublicData::file.GetRecordNum() ? "" :
                 PublicData::file.ReadRecord_Character(statsPRItemNum + 2) );

                for (int j = 0; j < 3; j++) {
                    if (statsPRItemNum + j >= PublicData::file.GetRecordNum())
                        break;

                    int tempScore = PublicData::file.ReadRecord_Score(statsPRItemNum + j),
                        tempLevel = PublicData::file.ReadRecord_Level(statsPRItemNum + j),
                        tempCorrectKeyCount = PublicData::file.ReadRecord_CorrectKeyCount(statsPRItemNum + j),
                        tempAccuracy = int(PublicData::file.ReadRecord_Accuracy(statsPRItemNum + j) * 100.0);
                    string tempDate = PublicData::file.ReadRecord_Date(statsPRItemNum + j);
                    const int STATS_PR_NUM_POS_X = 135, STATS_PR_NUM_PER_POS_X = 380;
                    const int LINE_MARGIN = 44;

                    for (int i = 0, signNum = 0; i < 12; i++) {                    // 顯示日期時間
                        numBmpSmall_White[tempDate[i] - '0'].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X - 40
+ 10 * i + signNum * 8, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                        numBmpSmall_White[tempDate[i] - '0'].ShowBitmap();

                        if (i == 3) {
                            numBmpSmall_White[12].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X - 40 + 10 * i
+ 10 + signNum * 8, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                            numBmpSmall_White[12].ShowBitmap();
                            signNum++;
                        }

                        if (i == 5) {
                            numBmpSmall_White[12].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X - 40 + 10 * i
+ 10 + signNum * 8, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                            numBmpSmall_White[12].ShowBitmap();
                            signNum++;
                        }

                        if (i == 7)    signNum++;

                        if (i == 9) {                                    // 冒號
                            numBmpSmall_White[13].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X - 40 + 10 * i
+ 12 + signNum * 8, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                            numBmpSmall_White[13].ShowBitmap();
                            signNum++;
                        }
                    }

                    for (int i = 0; i < 5; i++) {                                // 顯示分數數字 bmp
                        numBmpSmall[tempScore % 10].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X + 170 - 10
* i, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                        numBmpSmall[tempScore % 10].ShowBitmap();
                        tempScore /= 10;
                    }

                    for (int i = 0; i < 2; i++) {                                // 顯示關卡數字 bmp
                        numBmpSmall_White[tempLevel  %  10].SetTopLeft(STATS_POS_X  +  STATS_PR_NUM_POS_X  +
222 - 10 * i, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                        numBmpSmall_White[tempLevel % 10].ShowBitmap();
                        tempLevel /= 10;
                    }

                    for (int i = 0; i < 5; i++) {                                // 顯示總按鍵數 bmp
                        numBmpSmall_White[tempCorrectKeyCount          %          10].SetTopLeft(STATS_POS_X          +
STATS_PR_NUM_POS_X + 306 - 10 * i, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                        numBmpSmall_White[tempCorrectKeyCount % 10].ShowBitmap();
                        tempCorrectKeyCount /= 10;
                    }
```

```
                if (tempAccuracy != 10000) {                    //若正確率非 100%
                    for (int i = 0, dotPos = 0; i < 4; i++) {       // 顯示正確率 bmp
                        numBmpSmall_White[tempAccuracy % 10].SetTopLeft \
                        (STATS_POS_X + STATS_PR_NUM_POS_X + STATS_PR_NUM_PER_POS_X - 10 * i - dotPos,
\
                          NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                        numBmpSmall_White[tempAccuracy % 10].ShowBitmap();
                        tempAccuracy /= 10;

                        if (i == 1) {                           // 顯示小數點
                            dotPos = 5;
                            numBmpSmall_White[11].SetTopLeft\
                            (STATS_POS_X + STATS_PR_NUM_POS_X + STATS_PR_NUM_PER_POS_X - 10 * i -
dotPos, \
                              NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                            numBmpSmall_White[11].ShowBitmap();
                        }

                        tempLevel /= 10;
                    }
                }
                else {                                                  // 針對 100%顯示
                    numBmpSmall_White[1].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X +
STATS_PR_NUM_PER_POS_X - 10 * 2, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                    numBmpSmall_White[1].ShowBitmap();
                    numBmpSmall_White[0].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X +
STATS_PR_NUM_PER_POS_X - 10, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                    numBmpSmall_White[0].ShowBitmap();
                    numBmpSmall_White[0].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X +
STATS_PR_NUM_PER_POS_X, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                    numBmpSmall_White[0].ShowBitmap();
                }

                numBmpSmall_White[10].SetTopLeft(STATS_POS_X + STATS_PR_NUM_POS_X +
STATS_PR_NUM_PER_POS_X + 14, NOTE_TEXT_Y + 130 + j * LINE_MARGIN);
                numBmpSmall_White[10].ShowBitmap();                 // 顯示百分比符號
            }
        }

        PublicData::me.OnShow();                                    // 顯示統計頁之角色（最高分和遊玩記錄）
    }
    else if (displayState == 4) {                                   // 顯示關於頁面
        if (aboutDisplayState == 0) {
            aboutBorder.SetTopLeft((SIZE_X - aboutBorder.Width()) / 2, NOTE_TEXT_Y);
            aboutBorder.ShowBitmap();                               // 顯示關於框
            about.SetTopLeft((SIZE_X - aboutBorder.Width()) / 2, NOTE_TEXT_Y + 11);
            about.ShowBitmap();                                     // 顯示關於文字

            if (PublicData::musicOnOff) {
                musicOnOff[0].SetTopLeft((SIZE_X - aboutBorder.Width()) / 2 + 458, NOTE_TEXT_Y + 180);
                musicOnOff[0].ShowBitmap();
            }
            else {
                musicOnOff[1].SetTopLeft((SIZE_X - aboutBorder.Width()) / 2 + 458, NOTE_TEXT_Y + 180);
                musicOnOff[1].ShowBitmap();
            }
        }
        else if (aboutDisplayState == 1) {                          // 刪除遊戲紀錄頁
            delText.SetTopLeft((SIZE_X - delText.Width()) / 2, (NOTE_TEXT_Y + delText.Height() / 2) );
            delText.ShowBitmap();
        }
    }

    text1.SetTopLeft((SIZE_X - text1.Width()) / 2, text1_y);
    text1.ShowBitmap();

    if (0) {                                                        // 展示顯示數字及字體
        for (int i = 0; i < 13; i++) {
```

```
                numBmpSmall[i].SetTopLeft(10 + i * 10, 300);
                numBmpSmall[i].ShowBitmap();
                numBmpSmall_White[i].SetTopLeft(10 + i * 10, 310);
                numBmpSmall_White[i].ShowBitmap();

                if (i < 10) {
                    numBmp[i].SetTopLeft(10 + i * 20, 320);
                    numBmp[i].ShowBitmap();
                    numBmp_White[i].SetTopLeft(10 + i * 20, 340);
                    numBmp_White[i].ShowBitmap();
                }
            }
        }
}
/////////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的結束狀態(Game Over)
/////////////////////////////////////////////////////////////////////////////
CGameStateOver::CGameStateOver(CGame* g)
    : CGameState(g) {
}
void CGameStateOver::OnMove() {
    if (isHighScore)  newHS_text.OnMove();                              // 移動破紀錄動畫

    //if (isUnlock)newChar_text.OnMove();

    if (counter < 0)    GotoGameState(GAME_STATE_INIT);

    if (barCounter < 200 && barCounter < accuracy) {
        barCounter +=    2;
    }
}
void CGameStateOver::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
    const char KEY_ENTER = 0xD;
    (nChar == KEY_ENTER) ? GotoGameState(GAME_STATE_INIT) : 0;      // 按下 Enter 鍵返回開頭頁面
}
void CGameStateOver::OnBeginState() {
    counter = 1000 * 30;
    barCounter = 0;
    isHighScore = isUnlock = false;
    //
    score = PublicData::score;
    level = PublicData::level;
    accuracy = PublicData::accuracy;
    //
    PublicData::me.SetState(2);
    PublicData::file.ReadHighScoreFile();

    if (PublicData::file.GetRecordNum() == 0)                           // 若不曾有遊玩記錄
        PublicData::totalKeyCount = PublicData::CorrectKeyCount;
    else {
        PublicData::totalKeyCount = PublicData::file.ReadTotalKeyCount();
        PublicData::totalKeyCount += PublicData::CorrectKeyCount;
    }

    if (score > PublicData::file.ReadHighScore_Score()) {               // 若本次分數大於 最高分則寫入
        PublicData::file.WriteHighScore(score,          level,          accuracy,          PublicData::me.GetMeName(),
PublicData::CorrectKeyCount);
        isHighScore = true;
    }

    PublicData::file.WriteRecord(score, level, accuracy, PublicData::me.GetMeName(), PublicData::CorrectKeyCount);
      // 寫入 單筆遊玩記錄
    PublicData::file.WriteTotalKeyCount(PublicData::totalKeyCount);     // 寫入 總按鍵數
    PublicData::file.ReadRecordFile();                                  // 讀 遊玩記錄

    //
    if (PublicData::me.JudgeUnlock(PublicData::totalKeyCount, score, int(accuracy), level)) {// 判斷是否達成解鎖要素
        isUnlock = PublicData::newUnlock = true;
    }
```

```
    if (PublicData::musicOnOff) {
        CAudio::Instance()->Stop(AUDIO_ROCK);                    // 暫停 背景音效
        CAudio::Instance()->Play(AUDIO_GAMEOVER, false);         // 播放 GAMEOVER 音效
    }
}
void CGameStateOver::OnInit() {
    ShowInitProgress(66);                                        // 接個前一個狀態的進度，此處進度視為 66%
    border.LoadBitmap("Bitmaps/gameover/gameover_border.bmp", RGB(0, 255, 0));
    newHS_text.AddBitmap("Bitmaps/gameover/gameover_new_hs1.bmp", RGB(0, 255, 0));
    newHS_text.AddBitmap("Bitmaps/gameover/gameover_new_hs2.bmp", RGB(0, 255, 0));
    newHS_text.SetDelayCount(20);
    newChar_text.AddBitmap("Bitmaps/gameover/gameover_new_char1.bmp", RGB(0, 255, 0));
    newChar_text.AddBitmap("Bitmaps/gameover/gameover_new_char2.bmp", RGB(0, 255, 0));
    newChar_text.SetDelayCount(20);
    ShowInitProgress(80);
    isHighScore = isUnlock = 0;

    for (int i = 0; i < 10; i++) {                               // 載入數字圖
        char str[80];
        sprintf(str, "Bitmaps/level/num/%d.bmp", i);
        numBmp[i].LoadBitmap(str, RGB(0, 255, 0));
        sprintf(str, "Bitmaps/level/num_s/%d.bmp", i);
        numBmpSmall[i].LoadBitmap(str, RGB(0, 255, 0));
    }

    bar[0].LoadBitmap("Bitmaps/gameover/bar_0.bmp", RGB(0, 255, 0));
    bar[1].LoadBitmap("Bitmaps/gameover/bar_1.bmp", RGB(0, 255, 0));
    numBmpSmall[10].LoadBitmap("Bitmaps/level/num_s/per.bmp", RGB(0, 255, 0));  // 載入百分比圖片
    numBmpSmall[11].LoadBitmap("Bitmaps/level/num_s/dot.bmp", RGB(0, 255, 0));  // 載入小數點圖片
    x = (SIZE_X - border.Width()) / 2;
    y = (SIZE_Y - border.Height()) / 2;
    ShowInitProgress(100);

    if (PublicData::musicOnOff)
        CAudio::Instance()->Load(AUDIO_GAMEOVER, "sounds\\gameover.mp3");
}
void CGameStateOver::OnShow() {                                  // GAMEOVER 畫面顯示
    border.SetTopLeft(x, y);
    border.ShowBitmap();

    if (isHighScore) {                                           // 顯示破紀錄動畫
        newHS_text.SetTopLeft((SIZE_X - newHS_text.Width()) / 2, y + border.Height() + 20);
        newHS_text.OnShow();
    }

    if (isUnlock) {
        newChar_text.SetTopLeft((SIZE_X - newHS_text.Width()) / 2, y + border.Height() + 20 + 35 * isHighScore);
        newChar_text.OnShow();
    }

    int tempScore = score, tempLevel = level, tempAccuracy = int (accuracy * 100.0);
    int dotPos = 0;
    int scorePosX = 230, levelPosX = 130, accPosX = 155,         // 定義各數字位置偏移量
        scorePosY =  45, levelPosY =  90, accPosY = 108,
        barPosX   = 120, barPosY   = 122;

    for (int i = 0; i < 5; i++) {                                // 顯示分數數字 bmp
        numBmp[tempScore % 10].SetTopLeft(x + scorePosX - 20 * i, y + scorePosY);
        numBmp[tempScore % 10].ShowBitmap();
        tempScore /= 10;
    }

    for (int i = 0; i < 2; i++) {                                // 顯示關卡數字 bmp
        numBmpSmall[tempLevel % 10].SetTopLeft(x + levelPosX - 10 * i, y + levelPosY);
        numBmpSmall[tempLevel % 10].ShowBitmap();
        tempLevel /= 10;
    }
```

```
    if (accuracy != 100) {                                          //若正確率非 100%
        for (int i = 0, dotPos = 0; i < 4; i++) {                   // 顯示正確率 bmp
            numBmpSmall[tempAccuracy % 10].SetTopLeft(x + accPosX - 10 * i - dotPos, y + accPosY);
            numBmpSmall[tempAccuracy % 10].ShowBitmap();
            tempAccuracy /= 10;

            if (i == 1) {                                           // 顯示小數點
                dotPos = 5;
                numBmpSmall[11].SetTopLeft(x + accPosX - 10 * i - dotPos, y + accPosY);
                numBmpSmall[11].ShowBitmap();
            }

            tempLevel /= 10;
        }

        numBmpSmall[10].SetTopLeft(x + accPosX + 14, y + accPosY);
        numBmpSmall[10].ShowBitmap();                               // 顯示百分比符號
    }
    else {                                                          // 針對 100%顯示
        numBmpSmall[1].SetTopLeft(x + accPosX - 35, y + accPosY);
        numBmpSmall[1].ShowBitmap();
        numBmpSmall[0].SetTopLeft(x + accPosX - 35 + 10, y + accPosY);
        numBmpSmall[0].ShowBitmap();
        numBmpSmall[0].SetTopLeft(x + accPosX - 35 + 10 * 2, y + accPosY);
        numBmpSmall[0].ShowBitmap();
        numBmpSmall[10].SetTopLeft(x + accPosX - 35 + 10 * 4, y + accPosY);
        numBmpSmall[10].ShowBitmap();                               // 顯示百分比符號
    }

    for (int i = 0; i < 100; i++) {                                 // 顯示百分比的進度條
        bar[0].SetTopLeft(x + barPosX + i, y + barPosY);
        bar[0].ShowBitmap();

        if (i < barCounter) {
            bar[1].SetTopLeft(x + barPosX + i, y + barPosY);
            bar[1].ShowBitmap();
        }
    }

    PublicData::me.OnShow();
}
////////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
////////////////////////////////////////////////////////////////////////////
CGameStateRun::CGameStateRun(CGame* g)
    : CGameState(g), LEVEL(30) {
    srand((unsigned)time(NULL));  // 亂數種子
    callEnemyCounter = maxCallEnemyCounter = 20;                    // maxCallEnemyCounter 決定怪物生成速度
    callBossACounter = maxCallBossACounter = 65;
    callBossBCounter = maxCallBossBCounter = 75;
}
CGameStateRun::~CGameStateRun() {
    for (CEnemy* eq : enemyQueue) delete eq;

    for (CBullet* bl : bulletList) delete bl;

    for (CBomb* cb : bombList) delete cb;
}
void CGameStateRun::OnBeginState() {
    const int SCORE_X = 240, SCORE_Y = 240;

    if (PublicData::musicOnOff)
        CAudio::Instance()->Play(AUDIO_ROCK, true);                 // 撥放 MIDI

    score.SetInteger(0);                                            // 設定 SCORE 為 0
    score.SetTopLeft(SCORE_X, SCORE_Y);
    currEnemyNum = currBossANum = currBossBNum = 0;                 // 初始化各關敵人已召喚數量
    lock = false;                                                   // 取消鎖定
    targetEnemy = NULL;
```

```
    currLevel = 0;                                                  // 初始化關卡
    enemyQueue.clear();                                             // 清空 EQ
    lives = 3;
    totalKeyDownCount = PublicData::CorrectKeyCount = 0;
    accuracy = 0;
    emp.SetEQ(&enemyQueue, &score, &lock, &targetEnemy);
    emp.SetEmpTimes(3);
    PublicData::me.SetState(0);
    totalEnemyNum = 0;
    levelChangeFlag = 0;
    levelChangeDelay = -1;
    levelChangeDelayMax = int( 3.5 * 30 );                                  // 設定關卡間 delay 3 秒
    pause = false;
}
void CGameStateRun::OnInit() {                                      // 遊戲的初值及圖形設定
    srand((unsigned)time(NULL));
    ShowInitProgress(33);
    //
    // 繼續載入其他資料
    //
    score.LoadBitmap();                             // 載入分數顯示器
    map.LoadBitmap();                               // 載入背景
    emp.LoadBitmap();                               // 載入 EMP
    levelAni.LoadBitmap();                          // 載入切換關卡過場動畫
    pauseText.LoadBitmap("Bitmaps/menu/pause.bmp", RGB(0, 255, 0));
    debugText.LoadBitmap("Bitmaps/debug_text.bmp", RGB(0, 255, 0));

    if (1) {
        CAudio::Instance()->Load(AUDIO_ROCK, "sounds\\The_Coming_Storm.mp3");
        CAudio::Instance()->Load(AUDIO_SHOT, "sounds\\shot.mp3");
        CAudio::Instance()->Load(AUDIO_ERROR, "sounds\\error.mp3");
        CAudio::Instance()->Load(AUDIO_CONGRATULATION, "sounds\\congratulation.mp3");
        CAudio::Instance()->Load(AUDIO_CONGRATULATION2, "sounds\\congratulation2.mp3");
    }

    ShowInitProgress(40);

    for (int i = 0; i < 26; i++) {                          // 載入字型圖片
        char str[50];
        sprintf(str, "Bitmaps/char4/%c.bmp", i + 97);
        letter.push_back(new CMovingBitmap);
        letter.back()->LoadBitmap(str, RGB(255, 255, 255));
    }

    for (int i = 0; i < 22; i++) {                          // 載入鎖定敵人瞄準動畫
        char str[50];
        sprintf(str, "Bitmaps/target/target_s%d.bmp", i + 1);
        target.AddBitmap(str, RGB(0, 255, 0));
    }

    target.SetDelayCount(2);
    ShowInitProgress(50);
}
void CGameStateRun::OnMove() {                  // 移動遊戲元素
    if (pause)return;

    if (levelChangeDelay >= 0)      levelChangeDelay--;         // 關卡與關卡間延遲的計數器

    accuracy = (totalKeyDownCount != 0) ? \
                100 * double(PublicData::CorrectKeyCount) / double(totalKeyDownCount) : \
                100;        // 計算正確率
    PublicData::score = score.GetInteger();
    PublicData::level = currLevel;
    PublicData::accuracy = accuracy;
    callEnemyCounter--;                                         // 每隻怪物 生成間隔 之 計數器
    callBossACounter--;
    callBossBCounter--;

    if (quickCall) {                                // 【DEBUG 區】 將第 0 關設定生成 200 只怪物，且召喚 delay 為 0 秒
```

```
            //levelEnemyNum[0] = 200;
            levelChangeDelayMax = 0;
            maxCallEnemyCounter = maxCallBossACounter    = maxCallBossBCounter = 0;
        }
        else {
            levelChangeDelayMax = int(3.5 * 30);
            maxCallEnemyCounter = 25;
            maxCallBossACounter = 75;
            maxCallBossBCounter = 90;
        }

        //==小怪==================================
        if (callEnemyCounter < 0 && currEnemyNum < levelEnemyNum[currLevel]) {      // counter 數到 0 後就開始召喚新怪
            callEnemyCounter = maxCallEnemyCounter;                                 // 把 counter 調回 max 繼續數
            int randX = (rand() % (SIZE_X - 100)) ;                                 // SIZE_X - 100 為了不讓怪物的單字超出螢幕太
多
            enemyQueue.push_back(new CEnemy(randX, 0, 3, true, &dictionary, 2, 5, &enemyQueue, &bombList, \
                                            PublicData::me.GetX1()  -  30  +  rand()  %  60,  PublicData::me.GetY1(),
&letter) ); // 將召喚的新怪放入 vecotr 內
            enemyQueue.back()->LoadBitmap();                                        // 載入召喚的新怪
            currEnemyNum++;                                                         // 在本關卡已召喚的怪物計數器
            totalEnemyNum++;                                                        // 總以召喚的怪物數量
        }

        //==BossA=================================
        if (callBossACounter < 0 && currBossANum < levelBossANum[currLevel]) {
            callBossACounter = maxCallBossACounter;
            int randX = (rand() % (SIZE_X - 350) + 200);
            enemyQueue.push_back(new CBossA(randX, 0, 5, true, &dictionary, 6, 12, &enemyQueue, &bombList, &letter));
            enemyQueue.back()->LoadBitmap();
            currBossANum++;
            totalEnemyNum++;
        }

        //==BossB=================================
        if (callBossBCounter < 0 && currBossBNum < levelBossBNum[currLevel]) {
            callBossBCounter = maxCallBossBCounter;
            int randX = (rand() % (SIZE_X - 350) + 200);
            enemyQueue.push_back(new CBossB(randX, 0, 5, true, &dictionary, 6, 12, &enemyQueue, &bombList, &letter));
            enemyQueue.back()->LoadBitmap();
            currBossBNum++;
            totalEnemyNum++;
        }

        //===判斷 Me 是否碰到 Enemy===

        for (CEnemy* eq : enemyQueue) {
            if (eq->IsAlive() && eq->HitMe(&PublicData::me)) {
                lives--;
                (lives <= 0) ? GotoGameState(GAME_STATE_OVER) : 0; // 若生命值為 0 則 GOTO 遊戲結束的 STATE
            }

            if (eq->GetX() > SIZE_X + 40 || eq->GetX() < -40 || eq->GetY() > SIZE_Y + 40) {
                eq->kill();                                                         // 當 enemy 飛出畫面外時殺掉怪物
            }
        }

        // ===Enemy===
        bool enemyAllDead = true;

        for (unsigned int i = 0; i < enemyQueue.size(); i++) {                      // 移動 VECTOR 內的所有怪物
            enemyQueue[i]->OnMove();
            enemyQueue[i]->IsAlive() ? enemyAllDead = false : 0;
        }

        for (unsigned int i = 0; i < enemyQueue.size(); i++) {
            //若 Enemy IsAlive=0, 則從 vector 中移除
            if (!enemyQueue[i]->IsAlive() && enemyQueue[i]->IsBombed()) {
                vector<CEnemy*>::iterator iterenemyQueue = enemyQueue.begin();
```

```
            delete enemyQueue[i];
            enemyQueue[i] = NULL;
            enemyQueue.erase(iterenemyQueue + i);
            i = 0;
        }
    }

    ////===bullet===
    bool bulletAllDead = true;

    for (CBullet* bl : bulletList) {
        bl->OnMove();                                   // 移動 BULLET

        if (bl->IsAlive()) bulletAllDead = false;
    }

    if (bulletAllDead) {                        // 若 VECTOR 內的子彈皆播放完畢，釋放記憶體並清空 VECOTR
        for (CBullet* bl : bulletList) delete bl;

        bulletList.clear();
    }

    ////===bomb===
    bool bombAllDead = true;

    for (CBomb* cb : bombList) {
        cb->OnMove();     // 移動 BOMB

        if (cb->IsAlive()) bombAllDead = false;
    }

    if (bombAllDead) {
        for (CBomb* cb : bombList) {            // 若 VECTOR 內的爆炸皆播放完畢，釋放記憶體並清空 VECOTR
            delete cb;
            cb = NULL;
        }

        bombList.clear();
    }

    ////===Change Level===
    if (currEnemyNum >= levelEnemyNum[currLevel] && currBossANum >= levelBossANum[currLevel] \
            && currBossBNum >= levelBossBNum[currLevel] && enemyQueue.size() == 0) {
        // 換 關卡
        if (!levelChangeFlag) {                                 // 等待 delay 算完
            levelAni.Play(currLevel, score.GetInteger());           // 播放切換關卡動畫
            levelChangeFlag = true;
            levelChangeDelay = levelChangeDelayMax;             // 將計數器調回

            if (PublicData::musicOnOff) {
                //CAudio::Instance()->Stop(AUDIO_ROCK);             // 暫停 背景音效
                //CAudio::Instance()->Play(AUDIO_CONGRATULATION, false);        // 撥放 過關音效
                //CAudio::Instance()->Play(AUDIO_CONGRATULATION2, false); // 撥放 過關音效 2
            }
        }

        if (levelChangeDelay < 0 && levelChangeFlag) {                  // 當 delay 算完後 再實際切
換關卡
            if (PublicData::musicOnOff) {
                //CAudio::Instance()->Stop(AUDIO_CONGRATULATION);       //暫停 過關音效
                //CAudio::Instance()->Stop(AUDIO_CONGRATULATION2);          //暫停 過關音效 2
                //CAudio::Instance()->Play(AUDIO_ROCK, true);           // 撥放 背景音效
            }

            currLevel++;

            if (currLevel > LEVEL)GotoGameState(GAME_STATE_OVER);
// 若當前關卡大於最大關卡則 GOTO 遊戲結束的 STATE
```

```
            currEnemyNum = currBossANum = currBossBNum = 0;                 // 重置該關卡已召喚的怪物數量
            callEnemyCounter = maxCallEnemyCounter;                             // 調回召喚怪物的計數器
            callBossACounter = maxCallBossACounter;
            callBossBCounter = maxCallBossBCounter;
            levelChangeFlag = false;
        }
    }

    //
    map.OnMove();                   // 移動背景
    emp.OnMove();                   // 移動 EMP
    PublicData::me.OnMove()     ;   // 移動主角
    levelAni.OnMove();              // 移動關卡切換動畫

    if (lock && targetEnemy != NULL) {
        target.OnMove();            // 移動鎖定目標動畫
    }
}
void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
    const char KEY_LEFT = 0x25;     // keyboard 左箭頭
    const char KEY_UP = 0x26;       // keyboard 上箭頭
    const char KEY_RIGHT = 0x27;        // keyboard 右箭頭
    const char KEY_DOWN = 0x28;         // keyboard 下箭頭

    if (pause) return;

    if (key != nChar) {
        key = nChar;                                                    // 變數 key 來儲存按下的按鍵，防止按鍵彈
跳

        for (int unsigned i = 0; i < enemyQueue.size(); i++) {          // 跑目前關卡怪物的數量
            if (enemyQueue[i]->IsAlive()) {                             // 回傳當前怪物是否存在
                if (!lock) {                                                    // 尚未鎖定
                    if (nChar + 32 == enemyQueue[i]->GetFirstWord()) {  // 若等於第一個字母:鎖住 and 目前字元位
置+1
                        PublicData::CorrectKeyCount++;                  // 正確按鍵數+1
                        map.PlayFlash();

                        if (PublicData::musicOnOff)
                            CAudio::Instance()->Play(AUDIO_SHOT, false);    // 撥放 射擊音效

                        if (enemyQueue[i]->GetVocabLeng() == 1) {           // A. 針對 1 字小怪攻擊
                            targetEnemy = enemyQueue[i];                // targetEnemy 為指標->正在攻擊的敵人
                            bulletList.push_back(new CBullet(targetEnemy->GetX() + 10, targetEnemy->GetY() + 10));

                                                                        // 射子彈
                            targetEnemy->kill();                        // 成功殺害怪物
                            score.Add(int(targetEnemy->GetVocabLeng() * (1 + accuracy / 100)));
                                                                        // 分數+= 怪物長度
                            targetEnemy = NULL;
                            break;
                        }
                        else {                                          // B. 針對一般的怪物攻擊
                            lock = true;                                // 已鎖定某只怪物
                            targetEnemy = enemyQueue[i];                // targetEnemy 為指標->正在攻擊的敵人
                            targetEnemy->AddCurrWordLeng();             // 已輸入到的單子數++
                            bulletList.push_back(new CBullet(targetEnemy->GetX() + 10, targetEnemy->GetY() + 10));
                                                                        // 射子彈
                            targetEnemy->MinusIndex(2);                 // 擊退怪物
                            break;
                        }
                    }
                    else {
                        if (nChar >= 65 && nChar <= 90)
                            if (PublicData::musicOnOff)
                                CAudio::Instance()->Play(AUDIO_ERROR, false);   // 撥放 射擊音效
                    }
                }
                else {                                                  // 若已鎖定
```

```
                    if (nChar + 32 == targetEnemy->GetVocab()[targetEnemy->GetCurrWordLeng()]) {
                                                                    // 若等於當前字母
                        targetEnemy->AddCurrWordLeng();
                        bulletList.push_back(new CBullet(targetEnemy->GetX(), targetEnemy->GetY()));
                        targetEnemy->MinusIndex(rand() % 2 + 1);        // 擊退怪物
                        PublicData::CorrectKeyCount++;                  // 正確按鍵數+1
                        map.PlayFlash();

                        if (PublicData::musicOnOff)
                            CAudio::Instance()->Play(AUDIO_SHOT, false);      // 撥放 射擊音效

                        if (targetEnemy->GetCurrWordLeng() == targetEnemy->GetVocabLeng()) {
                                                                    // 若當前長度 等於 字母的長度
                            targetEnemy->kill();                            // 成功殺害怪物
                            score.Add(int(targetEnemy->GetVocabLeng() * (1 + accuracy / 100)));
                                                                    // 分數+= 怪物長度
                            targetEnemy = NULL;                 // 因怪物已被殺害,將 targetEnemy 指標指向 NULL
                            lock = false;                       // 取消鎖定怪物
                        }

                        break;
                    }
                    else {
                        if (nChar >= 65 && nChar <= 90)
                            if (PublicData::musicOnOff)
                                CAudio::Instance()->Play(AUDIO_ERROR, false);      // 撥放 射擊音效
                    }
                }
            }
        }

        if (nChar >= 65 && nChar <= 90) totalKeyDownCount++;                // 總按鍵數++

        if (nChar == 13) {                                          // 若按下 ENTER 則發動 EMP 攻擊.
            emp.CallEmp(PublicData::musicOnOff);
        }
    }
}
void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
    const char KEY_LEFT = 0x25;                                     // keyboard 左箭頭
    const char KEY_UP = 0x26;                                   // keyboard 上箭頭
    const char KEY_RIGHT = 0x27;                                    // keyboard 右箭頭
    const char KEY_DOWN = 0x28;                                     // keyboard 下箭頭
    const char KEY_ESC = 27;
    key = NULL;

    if (nChar == KEY_ESC) pause = pause ? false : true;

    if (nChar == 'B' && pause) {
        CAudio::Instance()->Pause();
        GotoGameState(GAME_STATE_INIT);
    }

    if (PublicData::debugMode) {                                    // 允許使用 DEBUG 按鍵
        if (nChar == '1')showDebug = showDebug ? false : true;      // 按 1 開關 debug

        if (nChar == '2' && enemyQueue.size() > 0) {                // 按 2 清除 EQ 最後一隻敵人
            enemyQueue.back()->kill();
            lock = 0;
        }

        if (nChar == '3' && enemyQueue.size() > 0) {                // 按 3 清除 EQ 中所有敵人
            for (CEnemy* ce : enemyQueue) ce->kill();

            lock = 0;
        }

        if (nChar == '4')GotoGameState(GAME_STATE_INIT);            // 按 4 GOTO 起始畫面
```

```
        if (nChar == '5')GotoGameState(GAME_STATE_OVER);                    // 按 5 GOTO 遊戲結束畫面

        if (nChar == '6')quickCall = quickCall ? false : true       ;       // 按 6 開關快速召喚

        if (nChar == '7')score.Add(100);                                    // 按 7 加百分
    }
}
void CGameStateRun::OnLButtonDown(UINT nFlags, CPoint point) {}
void CGameStateRun::OnLButtonUp(UINT nFlags, CPoint point) {}
void CGameStateRun::OnMouseMove(UINT nFlags, CPoint point) {}
void CGameStateRun::OnRButtonDown(UINT nFlags, CPoint point) {}
void CGameStateRun::OnRButtonUp(UINT nFlags, CPoint point) {}
void CGameStateRun::OnShow() {
    map.OnShow();                                   // 顯示背景
    score.ShowBitmap();                             // 顯示分數
    emp.OnShow();                                   // 顯示 EMP（電磁波）
    PublicData::me.OnShow();                        // 顯示主角
    levelAni.OnShow();                              // 顯示關卡切換動畫

    for (CBomb* cb : bombList)     cb->OnShow();            //顯示 VECTOR 中所有的 爆炸

    for (CBullet* bl : bulletList)     bl->OnShow();                //顯示 VECTOR 中所有的 子彈

    for (CEnemy* eq : enemyQueue)       eq->OnShow();       //顯示 VECTOR 中所有的 怪物

    if (lock && targetEnemy != NULL) {
        targetEnemy->OnShow();                      // 加上這一行 讓被鎖定的怪物再次顯示，以防被其他怪物蓋住

        if (targetEnemy->GetBossType() == "enemy")
            target.SetTopLeft(targetEnemy->GetX() - 2, targetEnemy->GetY() - 2);    // 設定普通怪物 瞄準動畫的位置
        else
            target.SetTopLeft(targetEnemy->GetX() + 8, targetEnemy->GetY() + 8);    // 設定 BOSS 瞄準動畫的位置

        target.OnShow();                                                            // 顯示瞄準動畫
    }

    if (PublicData::debugMode) {                                                    // 顯示 debug 操作說明
        debugText.SetTopLeft(SIZE_X - debugText.Width() - 20, 20);
        debugText.ShowBitmap();
    }

    if (pause) {
            // 顯示暫停視窗
        pauseText.SetTopLeft((SIZE_X - pauseText.Width()) / 2, (SIZE_Y - pauseText.Height() ) / 2);
        pauseText.ShowBitmap();
    }

    if (showDebug) {                                                                // 顯示 debug 資訊
        CDC* pDC = CDDraw::GetBackCDC();
        CFont f, *fp;
        f.CreatePointFont(100, "Fixedsys");
        fp = pDC->SelectObject(&f);
        pDC->SetBkColor(RGB(0, 0, 0));
        //
        char temp[100];
        sprintf(temp, "TotalKeyNum: %d TotalCorrectKeyNum: %d , Accuracy: %.2lf%%", \
                totalKeyDownCount, PublicData::CorrectKeyCount, accuracy);
        pDC->SetTextColor(RGB(50, 200, 200));
        pDC->TextOut(20, 20, temp);
        //
        char temp1[100];
        sprintf(temp1, "EQ size: %d,    Live: %d,    Level: %d,    LOCK: %d,    TotalEnemyNum: %d", \
                enemyQueue.size(), lives, currLevel, bool(lock), totalEnemyNum);
        pDC->SetTextColor(RGB(255, 255, 255));
        pDC->TextOut(20, 40, temp1);

        //

        if (1) {
```

```
                for (unsigned int i = 0; i < enemyQueue.size(); i++) {                  // 顯示場上怪物之 單字,curr/length
                    char temp[50];
                    sprintf(temp,           "%s          %d/%d          (x:%d,y:%d)",          enemyQueue[i]->GetVocab().c_str(),
enemyQueue[i]->GetCurrWordLeng(), enemyQueue[i]->GetVocabLeng(), enemyQueue[i]->GetX(), enemyQueue[i]->GetY());
                    pDC->SetTextColor(RGB(180 + i, 180 + i, 180 + i));
                    pDC->TextOut(20, i * 14 + 60, temp);
                }
            }

            ////
            pDC->SelectObject(fp);                                          // 放掉 font f (千萬不要漏了放掉)
            CDDraw::ReleaseBackCDC();                                       // 放掉 Back Plain 的 CDC
        }
    }
    }
```

台北科大　資工系　2017　物件導向程式設計實習　第5組　題目 Typing Typing