**目錄**

[壹、 簡介 1](#_Toc11285950)

[一、 動機 1](#_Toc11285951)

[二、 分工 1](#_Toc11285952)

[貳、 遊戲介紹 2](#_Toc11285953)

[一、 遊戲說明 2](#_Toc11285954)

[二、 遊戲圖形 2](#_Toc11285955)

[三、 遊戲音效 2](#_Toc11285956)

[參、 程式設計 3](#_Toc11285957)

[一、 程式架構 3](#_Toc11285958)

[二、 程式類別 3](#_Toc11285959)

[三、 程式技術 11](#_Toc11285960)

[肆、 結語 15](#_Toc11285961)

[一、 問題與解決方法： 15](#_Toc11285962)

[二、 時間表 26](#_Toc11285963)

[三、 貢獻比例 35](#_Toc11285964)

[四、 自我檢核表 35](#_Toc11285965)

[五、 收穫 35](#_Toc11285966)

[六、 心得、感想 36](#_Toc11285967)

[伍、 附錄 39](#_Toc11285968)

2. 簡介
   1. 動機

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

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

* 1. 分工

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

1. 遊戲介紹
   1. 遊戲說明
   2. 遊戲圖形
   3. 遊戲音效
2. 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 |
| CEraser | 66 | 192 | CRole、CMonster、CNPC的父類別，提供基礎函式與屬性。 |
| CRole | 113 | 441 | 遊戲中玩家操控的角色，具有移動、跳躍、攻擊(丟蔥)、碰撞等操作。 |
| CNPC | 35 | 101 | 遊戲中玩家能與之對話的NPC，具有虛擬函式，為NPC1、NPC3之父類別。 |
| CNPC1 | 13 | 19 | 遊戲中玩家能與之對話的其中一種NPC，負責一般的對話。 |
| CNPC3 | 19 | 50 | 遊戲中玩家能與之對話的其中一種NPC，與之對話時會播放音樂(簡單來說就是音樂播放器) 。 |
| CMonster | 30 | 200 | 遊戲中玩家能以攻擊消滅的小怪物，可以移動以及跟障礙物碰撞，消滅時會玩家獲得分數，具有虛擬函式，為CMonsterType1以及CMonsterType2的父類別。 |
| CMonsterType1 | 7 | 13 | 遊戲中玩家能以攻擊消滅的小怪物，沒有動畫。 |
| CMonsterType2 | 8 | 45 | 遊戲中玩家能以攻擊消滅的小怪物，具有動畫。 |
| CBall | 23 | 51 | CSacllion的父類別，提供基礎函式與屬性。 |
| CSacllion | 29 | 128 | 玩家或Boss能丟出的子彈，能設定以拋物線軌道或直線軌道飛行。 |
| CBlackHole | 19 | 80 | 女老師可以丟的特殊子彈，繼承自CSacllion，當黑洞(此類別)與玩家碰觸時，會將玩家吸向黑洞的中心。 |
| CRay | 24 | 32 | 直線射線，具有動畫，只有動畫的最後一格能造成傷害。 |
| CBoss | 73 | 168 | Boss的基礎類別，具有虛擬函式，提供Boss一些基礎函式與屬性。 |
| CXingting | 46 | 299 | 繼承自CBoss，會以圓周運動丟子彈(遠東單字本)攻擊玩家，並且會丟黑洞，該Boss無法被玩家攻擊，玩家須閃躲攻擊40秒後，此Boss會自爆死亡。 |
| CFacaiSeed | 39 | 289 | 繼承自CBoss，會丟出金幣、釋放射線攻擊玩家，玩家與他接觸時會降低魔力，當玩家的魔力降為0的時候，玩家會持續扣血。該Boss可被玩家攻擊，其血量歸零，Boss死亡。 |
| CSimpleMapObj | 25 | 0  (函式直接定義在.h) | 地圖上使用的簡單物件，提供一些基礎函式與屬性。 |
| CBlock | 9 | 0  (函式直接定義在.h) | 繼承CSimpleMapObj的簡單物件，為擺在地圖上的障礙物，角色能站在障礙物上，碰到障礙物時無法移動。 |
| CDoor | 20 | 0  (函式直接定義在.h) | 繼承CSimpleMapObj的簡單物件，為傳送門，角色能透過門傳送到上方地圖/下方地圖。 |
| CBlockMap | 64 | 168 | 遊戲中使用的地圖，一個CBlockMap即儲存一張地圖的資訊，可以透過txt讀取資訊，也可以將資訊寫出來。 |
| myLibrary  (namespace) | 11 | 149 | 自定義library，有許多函式，像是將string轉成char\*，取得資料夾內的所有檔名、取得資料夾內的檔案多寡，將字串轉成數字，切割字串等等。 |
| BitmapPath | 23 | 0(函式直接定義在.h) | 儲存圖片、動畫的路徑、檔名、數量、透明色等。 |
| IsPointInRect  (function) | 1 | 7 | 自定義函式，判斷點是否在目標矩形內。 |
| IsRectCollision (function) | 1 | 4 | 自定義函式，判斷矩形是否與目標矩形碰撞。 |
| CTimer | 22 | 61 | 提供計時功能。 |
| CAnimate | 45 | 239 | 改寫自framework內類別CAnimation，提供動畫功能。 |
| CInteger | 31 | 77 | 改寫自framework內類別CInteger，提供數字圖像化功能。 |
| CLayer | 12 | 24 | 可以設定/取得圖層。 |
| CDialog | 29 | 95 | 儲存對話所需要的資訊，包含文本、名稱、能否重複觸發等。 |
| CCamera | 23 | 68 | 使螢幕的顯示範圍限縮在Camera照到的區域，換句話說，使地圖具有卷軸功能。 |
| CButton | 45 | 172 | 提供有圖片，可點擊的按扭。 |
| CAction | 54 | 211 | 提供動作功能(各種不同動作的動畫)。 |
| CEnd | 22 | 67 | 儲存結局的資訊，包含圖片、文本、是否已獲得該結局等。 |
| CToumeiImage | 24 | 62 | 提供帶透明度的圖片(需配合alphaBlend) |
| CStatusBoard | 26 | 85 | 提供玩家的UI介面，包含血量、魔力、頭像等。 |
| CBossBoard | 37 | 127 | 提供Boss的UI介面，包含血量、頭像等。 |
| CWindow | 37 | 115 | 提供彈出式視窗，具有背景與關閉按扭。 |
| CScrollWindow | 27 | 177 | 繼承自CWindow，為一可滑動視窗，可透過滑鼠滾輪調整顯示範圍。 |
| CSwitchWindow | 25 | 158 | 繼承自CWindow，為一可切頁視窗(透過螢幕上的箭頭)，同時滑鼠點擊左/右鍵可以換螢幕上的圖(下/上一張)。 |
| CPanel | 22 | 63 | 繼承自CWindow，為一彈出式視窗，用於遊戲中的暫停頁面，可使玩家選擇繼續遊戲、重開遊戲、回到選單、離開遊戲四種操作。 |
| CLayerManager | 32 | 120 | 圖層管理器，將所有圖片、動畫、動作、圖像數字蒐集起來，在此依照圖層分類並輸出。 |
| CMapManager | 63 | 337 | 地圖管理器，創建、管理地圖資訊(CBlockMap)，包含圖片、位置、地圖上的物件等，能切換地圖，以及使Monster重生。 |
| CMonsterManager | 19 | 79 | 小怪物管理器，負責創建、刪除小怪物。 |
| CBossManager | 18 | 77 | Boss管理器，負責儲存Boss，並且提供選取Boss功能。 |
| CDialogManager | 53 | 382 | 對話管理器，負責創建、管理對話(CDialog)，並提供CDialog開始對話、結束對話、顯示對話等功能。 |
| CNPCManager | 19 | 81 | NPC管理器，負責創建、管理NPC，能在切換地圖的時候顯示不同的NPC。 |
| CEventManager | 16 | 72 | 事件管理器，負責觸發主程式的一些事件，以位置事件為主。 |
| CEndManager | 37 | 179 | 結局管理器，負責創建、管理結局(CEnd)，並提供CEnd開始結局、結束結局等功能，並能記錄獲得哪些結局。 |
| CButtonManager | 32 | 107 | 按鈕管理器，負責管理按扭(CButton)。 |
| UIManager | 23 | 36 | UI管理器，控制角色、Boss的血條，以及玩家獲得的分數。 |
| ImageInfo | 15 | 0(函式直接定義在.h) | 繼承CSimpleMapObj，僅在地圖編輯器中使用，為地圖編輯器上的物件。 |
| Arrow | 50 | 0(函式直接定義在.h) | 僅在地圖編輯器中使用，設置地圖模式中出現的可點擊切換地圖的箭頭。 |
| CMapEditer | 148 | 453 | 地圖編輯器，以工具列提供簡單編輯地圖功能，可以新增/載入/儲存地圖，可以新增障礙物或門在地圖上，且障礙物或門可以進行拖曳。其他還可以設置連結地圖。 |
| CGameStaeInit | 26 | 207 | 遊戲的開頭畫面，具有三個按扭：進入遊戲、觀看擁有結局，教學頁面 |
| CGameStateRun | 48 | 677 | 遊戲的遊戲畫面，為程式的主要部分。 |
| CGameStateOver | 21 | 106 | 遊戲的結束頁面，在這裡播放結局。 |
| CGameState  MapEditer | 23 | 153 | 遊戲的地圖編輯頁面，在這裡使用地圖編輯器 |
| 總計 | 1822 | 7303 |  |

* 1. 程式技術

|  |  |
| --- | --- |
| 技術 | 說明 |
| 檔案讀寫 | 以fstream讀取、寫入檔案。 |
| 分割字串 | 透過stringstream，以空白將字串切割，並能得到切割後的子字串。 |
| 字串轉換數字 | 透過stringstream讀取字串，並將其輸出給整數型態，從而得到數字。 |
| 點與矩形碰撞 | 透過點的座標與矩形的座標，判斷點是否在矩形中。  將此函數擴展，可以得到矩形與矩形碰撞(矩形的四個點，任一點在目標矩形中，即與目標矩形碰撞)。 |
| 圖層 | 將圖片/動畫/動作等賦予圖層，並交由圖層管理器(layerManager)統一顯示。  其概念為：將圖片/動畫/動作放入對應圖層(index)的陣列裡面的vector，而圖層管理器在輸出的時候，從陣列index為0開始輸出圖片，直到陣列index為最大圖層數時停止，如此一來，圖層叫小的圖片/動畫/動作會先被呼叫顯示，而圖層大的則後呼叫顯示。  為了方便將圖片/動畫/動作加入圖層管理器，圖層管理器設置為static，令其能在任何地方被呼叫。 |
| 對話 | 對話能讀取文本中的字串，並由對話管理器(CDialogManager)統一管理所有的對話。  其概念為：對話主體(CDialog)能夠以fstream讀寫檔案，配合字串分析，取得對話文本中「誰說了什麼話」，而對話管理器CDialogManager統一管理所有的對話，並提供介面使管理的對話能執行開始對話(Start)、結束對話(Stop)、下一句(Next)、顯示對話(showText)。  其中開始對話以指標取得指定的CDialog，並進行初始化設定，結束對話則將指標還為nullptr。顯示對話為主要部份，會取得指定的CDialog的字串，並將其以最大字數(bytes數，ASCII體系，中文2bytes，英文1bytes)切割為個別的子字串，再將切割完的子字串交由CDC繪出。  對話同時也具有逐字顯示的效果，是利用計時器，每隔0.1秒改變指定字串的長度來達成。  為了方便讓對話能在任何地方被觸發，因此對話管理器設成static。 |
| 結局 | 結局利用圖片+對話的方式呈現，同樣由結局管理器(CEndManager)統一管理所有結局。  其概念為：一個結局具有多組圖+多組對話，一張圖片會搭配一個對話，當圖片播放完時代表結局結束。結局管理器(CEndManager)提供介面使管理的結局能執行開始結局(Start)、結束結局(Stop)，同時能透過fstream讀取「擁有哪些結局」並記錄之，使得擁有的結局不會因為關閉程式而遺失。  結局中使用的圖片帶有淡入淡出的透明度效果，其透明度以MFC自帶的CBitmap以及CDC內的alphaBlend函式達成。 |
| 透明圖片 | 使用MFC自帶的CBitmap以及CDC內的alphaBlend函式達成，CBitmap雖然本身並不帶透明度，但可以將其傳給alphaBlend，並在alphaBlend額外賦予透明度，將CBitmap以帶透明的效果顯示。 |
| 資料夾操作 | 能夠開啟指定的資料夾，並讀取資料夾內所有的檔名，或是取得資料夾內的檔案總數目。  資料夾操作藉由dirent.h達成，dirent.h為unix系統下，C/C++的library，但很不幸的Windows並不支援，因此我們使用他人改寫能相容於Windows系統的dirent.h。  程式中透過while迴圈以及readdir()操作，讀取指定資料夾內所有的檔名，利用此方法，我們成功自動化讀取音效音樂、對話文本、對話頭像、結局等等需要大量Load指令的檔案，使用上更為人性。  Github：<https://github.com/tronkko/dirent> |
| 滑鼠滾輪事件 | afxwin中已經有定義滑鼠滾輪事件onMouseWheel()了，因此只需在CGame中新增virtual function：OnMouseWheel，即可在CGameStateInit、CGameStateRun、CGameStateOver中使用滑鼠滾輪事件，另外，滾輪事件中傳遞的short參數zDelta為判斷滾輪是上滾或是下滾，zDelta > 0代表滾輪上滾，zDelta < 0代表滾輪下滾。 |
| 複製檔案 | MFC使用原生函數CopyFile  CopyFile(原檔案路徑, 目的地路徑, 是否不覆蓋檔案) |
| 刪除檔案 | MFC使用原生函數DeleteFile  DeleteFile(檔案路徑) |
| 工具列 | 工具列的使用必須先在game.rc的MENU中新增按鍵，並設置按鍵的ID，然後在gameView.cpp中設置並撰寫對應按鍵ID的函式。  在程式中我們希望能利用工具列執行主程式上不同的操作，因此利用fstream檔案讀寫技術，當點擊工具列特定按扭時，以txt形式將指令寫出，同時在主程式讀取該txt並分析其內容，再根據內容做出不同的操作，藉此達到目的。 |
| 地圖編輯器 | 地圖編輯器提供簡單的圖形化編輯地圖功能，能以工具列新增/開啟地圖、新增物件(障礙物或門)。  新增出來的物件可以藉由滑鼠拖曳改變其位置，其概念為：當滑鼠點擊時，選取被點擊的物件，並在滑鼠持續按住左鍵的狀態中，讓選取的物件隨著滑鼠座標移動。同樣概念，被選取的物件也可以透過delete刪除。  編輯好的地圖可以進行存檔操作，在存檔操作中會把地圖編輯器上的物件轉換為基礎地圖資訊(CBlockMap)，並透過CBlockMap自帶的函式將地圖資訊寫出。  在地圖編輯器編輯完地圖後，不需重啟程式來重新載入地圖。我們在編輯的過程的同時紀錄「哪些地圖被編輯過」，在重新回到遊戲的時候，將被編輯過的地圖重新創建，替換掉原本的地圖，達到線上重新載入的功能。 |

1. 結語
   1. 問題與解決方法：

問題一：

學期初執行程式時，程式偶爾會跳出存取被拒，導致不能執行。

解決方法：

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

問題二：

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

解決方法：

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

問題三：

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

解決方法：

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

問題四：

地圖捲動的時候，地圖上的物件沒有跟著捲動。

解決方法：

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

問題五：

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

解決方法：

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

問題六：

有時候需要計時，時間到執行某個動作。

解決方法：

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

問題七：

每次碰撞都要取出點座標進行判斷，不僅麻煩也容易出錯。

解決方法：

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

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

問題八：

有時候執行程式時，會馬上彈出CDDraw的某個東西為nullptr。

解決方法：

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

問題九：

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

解決方法：

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

問題十：

製作對話的時候，發現如果將所有字串都硬設在程式裡，要修改時相當不容易。

解決方法：

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

問題十一：

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

解決方法：

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

問題十二：

CDialog對話所讀取的字串如果太長，在顯示的時候會超過遊戲畫面。

解決方式：

限制每一行最大字數(bytes數)，超過該最大字數就換行，注意這裡的換行並不是字串加上\n，而是「換一個char\*」給CDC輸出，如此才能確保位置正確。

問題十三：

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

解決方法：

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

問題十四：

CDialog對話在進行的時候，該如何讓文字逐字顯示？

解決方法：

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

問題十五：

地圖上的小怪物如何重生？

解決方法：

當子彈與小怪物碰撞時，將刪除小怪物並將其從vector中移除，與此同時在vector中new一個新的小怪物，並賦予新的小怪物一亂數時間，當時間到的時候才讓小怪物顯示在地圖上，如此就有一種重生的效果。

問題十六：

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

解決方法：

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

問題十七：

CAudio每次多一條音效時都要多寫一行載入程式來載入音效，挺麻煩的。

解決方式：

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

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

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

問題十八：

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

解決方法：

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

問題十九：

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

解決方法：

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

問題二十：

CMovingBitmap只能讀bmp，但又想要讓結局的圖片有透明效果

解決方法：

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

問題二十一：

想要在關掉程式後依然能夠存取已經擁有的結局。

解決方法：

在離開遊戲的時候將「目前擁有的結局」透過fstream寫入txt檔儲存。

問題二十二：

能在擁有結局的頁面中透過點擊的方式看結局的劇情。

解決方法：

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

問題二十三：

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

解決方法：

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

問題二十四：

string如何以空白切割字串？

解決方法：

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

問題二十五：

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

解決方法：

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

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

問題二十六：

地圖編輯器中新增的物件想要進行拖曳以及刪除操作。

解決方式：

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

問題二十七：

新增/編輯好的地圖要存檔。

解決方式：

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

問題二十八：

編輯好地圖後，希望能直接測試，不用關掉程式讓地圖重新載入。

解決方法：

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

問題二十九：

人物在2D卷軸上，想把上(W)鍵，做成跳躍

解決方式：

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

問題三十：

人物跳躍後落在地圖物件或地上，要停止

解決方式：

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

問題三十一：

人物跳在地圖物件上後，離開不會下降

解決方式：

在主程式中加入isOnFloor，來判斷是否在地面或地圖物件上，並在人物中加入GetDrop，來獲取目前是否處於下降狀態，當isOnFloor = false，並沒有在跳躍時，將人物強制設成在下降狀態，並無法跳躍，直到回到地板，或地圖物件上。

問題三十二：

人物之攻擊方式，想以拋物線，而不是直線進行攻擊

解決方式：

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

問題三十二：

在人物攻擊後，出現大量memory leak問題

解決方式：

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

問題三十三：

人物的動畫只能循環跑圖

解決方式：

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

問題三十四：

人物的動畫，在左右方向切換時，有閃爍現象

解決方式：

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

問題三十五：

想在某些地方顯示類似視窗，可額外提供資訊之面板

解決方式：

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

問題三十六：

在面板中想實現滾輪捲動頁面

解決方式：

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

* 1. 時間表

說明部分，黑色為郭宗育所做，紅色為徐紹崴所做

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 時數  郭宗育(小時) | 時數  徐紹崴(小時) | 說明 |
| 1 | 3 | 3 | 練習Game Framework Tutorial |
| 2 | 5 | 4 | 確定分組、題目，完成Git練習 |
| 3 | 6 | 2 | 放入地圖、角色  讓角色能左右行走、添加角色動畫  當角色跨越地圖時，可以切換地圖  (如果沒有地圖可以跨越，角色會停止在屏幕邊界)  實作函式使String轉換成char\* |
| 4 | 10 | 4 | 新增圖層概念，實作layerManager管理圖層  新增小怪物，並能顯示在地圖上  如果地圖的寬度大於640，地圖具有卷軸效果  讓角色能夠跳躍 |
| 5 | 12 | 6 | 使角色能朝向滑鼠方向丟子彈(蔥)  實作點與矩形碰撞、矩形與矩形碰撞  當小怪物被子彈砸到時，可以獲得分數，並且小怪死亡，經過一段時間後重生  新增類別CTimer，負責計時  函數化CDC的繪出文字 |
| 6 | 6 | 4 | 添加更多音效  製作icon  更新layerManager，令其使用更方便  修正CTimer，令其能以毫秒為單位計時  解決轉換函式(string轉成char\*)產生的memory leaks |
| 7 | 8 | 2 | 新增類別BitmapPath，方便讀取動畫  實作Boss以及BossManager、試放Boss  實作Dialog負責儲存對話，  CDialogManager，負責進行對話  能夠進行簡單的對話  製作Dialog文本 |
| 8 | 10 | 10 | 新增Camera，解決有時候卷軸發生異常問題  能讓地圖上其他物件隨卷軸移動  以txt文本讀取對話，並可以逐字顯示在螢幕上  解決對話因為中、英文大小不依產生的亂碼問題  一段對話結束時，有提示  提醒使用者可以繼續下一句對話  更新對話背景素材 |
| 9 | 14 | 8 | 改寫CAudio(將音效定義在CAudio內)  新增Boss-女老師，女老師可以丟子彈(遠東單字本)  她可以丟出圓周運動的子彈  她丟出的子彈碰到角色，角色會扣血  更新碰撞函式，解決大矩形碰到小矩形時碰撞失敗的問題  實作Npc以及NpcManager  可以與Npc進行對話  可以與另一型的Npc進行對話，並且聽指定的音樂(該Npc為一音樂播放器)  整理RES資料夾  將以CDC繪出的數字圖像化  新增Button，方便時做兩種狀態之按扭 |
| 10 | 14 | 12 | 新增3個NPC  完成Boss-女老師，新增攻擊：丟黑洞，當角色碰到黑洞的時候，會被黑洞吸走，當黑洞炸裂時，會釋放12個子彈  女老師不能被攻擊，但時間到的時候女老師會自爆  實作End及EndManager負責控制結局，結局具有圖片+對話框  使結局的圖片能以透明效果淡入淡出  增加數字的圖片資源  實作CAction，增加角色的動作(待機、移動、跳躍)  ，使腳色跳躍、待機時能有不同方向的動畫 |
| 11 | 6 | 12 | 更新結局，令結局能以多組圖片+對話顯示  角色新增判定點，Boss攻擊到角色的判定點，角色才會扣血  修正女老師不能打第二次的錯誤  新增結局：鹹魚(什麼條件都沒達成)  修正Camera的錯誤  新增1個NPC  新增2結局：打贏女老師、打輸女老師  時做視窗，使畫面可以多一個可關閉的面板，顯示戰時需要的資訊。  實作滑動視窗，滑動視窗能以滑鼠滾輪控制視窗的顯示範圍 |
| 12 | 6 | 6 | 解決虛擬函式產生的memory leaks  使程式能讀取資料夾內的所有檔案，自動化載入音效  實作地圖編輯器MapEditer，能以工具列新增地圖、在地圖上放障礙物，儲存新增的地圖  新增Ending視窗，以滑動視窗顯示目前擁有的結局  ButtonManager，將Button集中管理、將Mouse坐標傳  入其中，將原本Button的更新狀態整理至其內。 |
| 13 | 12 | 3 | 能以滑鼠拖曳地圖編輯器上的物件  將地圖的資訊以txt的方式存起來  遊戲能以txt的方式讀取地圖  地圖編輯器能以txt的方式載入地圖，並進行新增物件、儲存地圖操作  解決buttonManager帶來的memory leaks  新增血條，將角色血量圖像化 |
| 14 | 12 | 4 | 在遊戲中新增障礙物  角色能與障礙物進行碰撞  角色能跳到障礙物上  地圖編輯器中而使用delete鍵刪除選取的物件  地圖素材整理、更新 |
| 15 | 12 | 3 | 在地圖編輯器新增/編輯完地圖時可以直接在遊戲中重新載入，不用重啟程式  修正地圖編輯器不能換背景的錯誤  地圖編輯器中可以對地圖設置上下左右方向的連結地圖  角色從障礙物上離開可以掉落  修正角色的碰撞範圍  角色跳起來時，如果撞到障礙物，不會穿過障礙物，而會直接進行降落  對話文字可以變色  更新地圖 |
| 16 | 18 | 20 | 新增傳送門，能讓角色傳送到上方的地圖或下方的地圖(如果有的話)  擁有結局的選單中，點選已經擁有的結局，可以觀看結局劇情  程式會記錄擁有那些結局，不會因為關閉程式而重置  讀取資料夾內的檔案，自動化載入對話所需要的文字以及圖片  新增6個Npc  新增1個小怪物  修正Npc，讓Npc可以看向角色的方向  小怪可以與障礙物進行碰撞  小怪物從障礙物讓離開時可以掉落  增加UIManager，控制人物以及Boss的血條，以及Boss對應之頭像，時間、分數的控制也在內部做更新  地圖大翻新，更換障礙物圖片 |
| 17 | 18 | 16 | 新增Boss-發財種子  發財種子會朝向角色移動，當碰到發財種子時，角色的魔力會逐漸下降，當沒有魔力時，角色會逐漸扣血  發財種子會從背上發射出大量金幣，金幣碰到角色時會使角色扣血  發財種子每隔一段時間會發射宇宙射線，宇宙射線碰到角色時會扣血  角色打贏發財種子的時候，會開啟1張新地圖  新增1個小怪物  新增可換頁視窗，用於操作說明  美化初始畫面  遊戲中新增暫停視窗，玩家在暫停視窗中可以選擇繼續玩、重新開始、回到畫面、結束，四種選項  完成操作說明指示 |
| 合計 | 172 | 120 |  |

* 1. 貢獻比例

徐紹崴 50%

郭宗育 50%

* 1. 自我檢核表

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

* 1. 收穫

|  |  |
| --- | --- |
| 郭宗育 | 徐紹崴 |
| Visual Studio 中斷點除錯功能 |  |
| C++ 物件使用概念 |  |
| C++ 物件繼承概念 |  |
| C++ fstream檔案讀寫 |  |
| C++ stringstrream 字串符流 |  |
| C++ STL map、vector |  |
| C++ static概念 |  |
| MFC CopyFile、DeleteFile |  |
| MFC FileDialog |  |
| Dirent 資料夾操作 |  |
| 自我學習 |  |

* 1. 心得、感想

郭宗育：

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

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

第一次demo的結果老實講，真的很令我訝異，我本來想說做到這樣頂多B+吧，沒想到拿到了A-的成績，讓我很是驚喜，於是在第一次demo後，我帶著愉悅的心情，做了整個程式我最滿意的東西──對話與對話管理器，此外，為了合理使用對話，我製作了NPC，一種NPC是一般的看劇情的NPC，而另一種NPC，則是可以播放音樂的音樂播放器。同時，我也進行Boss的製作，並做出結局，讓玩家打贏Boss、被Boss打死、或是什麼事情都不做的時候，都能進入相應的結局。本來我以為做了這些東西，第二次demo應該起碼能苟個A吧？然而事情卻不像我想的一樣順利──第二次demo一樣只拿了A-，老實講，在當時我不太能接受這件事情，當我看到成績的時候，氣到直接去打麻將。

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

1. UI做的不夠好，像是角色的血量不是以血條方式呈現
2. 自認做的最好的對話，不被認為是以fstream讀檔，而是在程式中硬設字串，或是讀取圖片
3. 我報告講太爛

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

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

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

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

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

徐紹崴：

1. 附錄
2. mygame.h

/\*

\* mygame.h: 本檔案儲遊戲本身的class的interface

\* Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>

\*

\* This file is part of game, a free game development framework for windows.

\*

\* game is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation; either version 2 of the License, or

\* (at your option) any later version.

\*

\* game is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

\*

\* 2004-03-02 V4.0

\* 1. Add CGameStateInit, CGameStateRun, and CGameStateOver to

\* demonstrate the use of states.

\* 2005-09-13

\* Rewrite the codes for CBall and CEraser.

\* 2005-09-20 V4.2Beta1.

\* 2005-09-29 V4.2Beta2.

\* 2006-02-08 V4.2

\* 1. Rename OnInitialUpdate() -> OnInit().

\* 2. Replace AUDIO\_CANYON as AUDIO\_NTUT.

\* 3. Add help bitmap to CGameStateRun.

\* 2006-09-09 V4.3

\* 1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they are

\* event driven.

\* 2008-02-15 V4.4

\* 1. Add namespace game\_framework.

\* 2. Replace the demonstration of animation as a new bouncing ball.

\* 3. Use ShowInitProgress(percent) to display loading progress.

\*/

#pragma once

#include "CEraser.h"

#include "CBall.h"

#include "CManager.h"

#include "CBoss.h"

#include "Refactor.h"

namespace game\_framework {

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame\* g);

~CGameStateInit();

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit(); // 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT); // 處理鍵盤Down的動作

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Up的動作

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnMouseWheel(UINT nFlags, short zDelta, CPoint point);

protected:

void OnMove();

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CButtonManager buttonManager;

CPoint mouse;

CMovingBitmap background; // background

CRole miku;

CScrollWindow windowsEnding;

CSwitchWindow windowsHandbook;

};

class CGameStateRun : public CGameState {

friend class CEventManager;

public:

CGameStateRun(CGame\* g);

~CGameStateRun();

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit(); // 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnMouseWheel(UINT nFlags, short zDelta, CPoint point);

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CRole role;

CInteger time\_left; // 剩下的撞擊數

CTimer timer; //stateRun狀態下的timer (其實好像在什麼狀態下都無所謂) (gameRun timer)

CTimer\* nowUsedTimer; //現在被使用的計時器 (BOSS戰鬥的時候 原本的遊戲計時器不會倒數)

vector<CScallion\*>\* scallions;

vector<CMonster\*>\* passerbys;

CMapManager mapManager;

CBossManager bossManager;

CNPCManager npcManager;

CEventManager eventManager;

CStatusBoard roleStatus;

CBossBoard bossStatus;

UIManager uiManager;

CPanel panel;

bool isWinXingting;

bool isWinFacaiSeed;

#pragma region - 自定義函數 -

void PositionTrigger();

void ChangeMap(string);

void ChangeMap(int);

void SwitchState(int);

void SwitchTimer(CTimer\*);

void GoToEnd();

#pragma endregion

};

class CGameStateOver : public CGameState {

public:

CGameStateOver(CGame\* g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

void OnKeyDown(UINT, UINT, UINT);

void OnLButtonDown(UINT nFlags, CPoint point);

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int counter; // 倒數之計數器

int alpha;

bool canSwitchState = false;

bool canDrawGameOverImage = false;

CTimer timer\_exit; //結局結束後要多久時間離開GameOver

CMovingBitmap overBitmap;

CToumeiImage gameOverImage;

CPanel panel;

};

class CGameStateMapEditer : public CGameState

{

public:

CGameStateMapEditer(CGame\* g);

~CGameStateMapEditer();

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

void OnLButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnLButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonDown(UINT nFlags, CPoint point); // 處理滑鼠的動作

void OnRButtonUp(UINT nFlags, CPoint point); // 處理滑鼠的動作

protected:

void OnMove();

void OnShow();

private:

CMapEditer mapEditer;

CPoint mousePoint;

};

1. mygame.cpp

/\*

\* mygame.cpp: 本檔案儲遊戲本身的class的implementation

\* Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>

\*

\* This file is part of game, a free game development framework for windows.

\*

\* game is free software; you can redistribute it and/or modify

\* it under the terms of the GNU General Public License as published by

\* the Free Software Foundation; either version 2 of the License, or

\* (at your option) any later version.

\*

\* game is distributed in the hope that it will be useful,

\* but WITHOUT ANY WARRANTY; without even the implied warranty of

\* MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the

\* GNU General Public License for more details.

\*

\* You should have received a copy of the GNU General Public License

\* along with this program; if not, write to the Free Software

\* Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

\*

\* History:

\* 2002-03-04 V3.1

\* Add codes to demostrate the use of CMovingBitmap::ShowBitmap(CMovingBitmap &).

\* 2004-03-02 V4.0

\* 1. Add CGameStateInit, CGameStateRun, and CGameStateOver to

\* demonstrate the use of states.

\* 2. Demo the use of CInteger in CGameStateRun.

\* 2005-09-13

\* Rewrite the codes for CBall and Crole.

\* 2005-09-20 V4.2Beta1.

\* 2005-09-29 V4.2Beta2.

\* 1. Add codes to display IDC\_GAMECURSOR in GameStateRun.

\* 2006-02-08 V4.2

\* 1. Revise sample screens to display in English only.

\* 2. Add code in CGameStateInit to demo the use of PostQuitMessage().

\* 3. Rename OnInitialUpdate() -> OnInit().

\* 4. Fix the bug that OnBeginState() of GameStateInit is not called.

\* 5. Replace AUDIO\_CANYON as AUDIO\_NTUT.

\* 6. Add help bitmap to CGameStateRun.

\* 2006-09-09 V4.3

\* 1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they are

\* event driven.

\* 2006-12-30

\* 1. Bug fix: fix a memory leak problem by replacing PostQuitMessage(0) as

\* PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE,0,0).

\* 2008-02-15 V4.4

\* 1. Add namespace game\_framework.

\* 2. Replace the demonstration of animation as a new bouncing ball.

\* 3. Use ShowInitProgress(percent) to display loading progress.

\* 2010-03-23 V4.6

\* 1. Demo MP3 support: use lake.mp3 to replace lake.wav.

\*/

#pragma once

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

#include "CManager.h"

#include "CBall.h"

#include "CLibrary.h"

#include <string>

#include <fstream>

#include <vector>

using namespace std;

using namespace myLibrary;

namespace game\_framework {

#pragma region - CGame Init -

CGameStateInit::CGameStateInit(CGame\* g) : CGameState(g)

{

#pragma region - Button Create -

buttonManager.CreateButton(BitmapPath("RES\\Button", "music", 2, RGB(214, 214, 214)), CPoint(250, 420), true, false);

buttonManager.CreateButton(BitmapPath("RES\\Button", "sound", 2, RGB(214, 214, 214)), CPoint(450, 420), true, false);

buttonManager.CreateButton(BitmapPath("RES\\Button", "play", 2, RGB(214, 214, 214)), CPoint(230, 200), false, true);

buttonManager.CreateButton(BitmapPath("RES\\Button", "ending", 2, RGB(214, 214, 214)), CPoint(230, 270), false, true);

buttonManager.CreateButton(BitmapPath("RES\\Button", "tutorial", 2, RGB(214, 214, 214)), CPoint(230, 340), false, true);

#pragma endregion

#pragma region - reset -

DeleteFile("RES\\Map\\FileName.txt");

DeleteFile("RES\\Map\\ReloadMapInformation.txt");

#pragma endregion

}

CGameStateInit::~CGameStateInit()

{

buttonManager.Clear();

}

void CGameStateInit::OnBeginState()

{

static bool isLoaded = false;

CLayerManager::Instance()->Initialize();

CEndManager::Instance()->Initialize();

CAudio::Instance()->Initialize();

CAudio::Instance()->Play("SLoWMoTIoN\_Menu", true);

if (!isLoaded)

{

buttonManager.Load();

windowsEnding.LoadResource();

windowsHandbook.LoadResource("RES\\Handbook\\introduction\\");

miku.LoadasMascot();

}

buttonManager.Initialize();

buttonManager.SetValid(true);

windowsEnding.Initialize(CPoint(0, 0));

windowsHandbook.Initialize(CPoint(0, 0));

miku.Initialize();

miku.SetXY(50, 320);

miku.GetAction()->SetFaceTo("\_R");

isLoaded = true;

}

void CGameStateInit::OnInit()

{

ShowInitProgress(0); // 一開始的loading進度為0%

background.LoadBitmap(".\\RES\\Map\\Menu\_3.bmp");

}

void CGameStateInit::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

if (nChar == 'N')

{

GotoGameState(GAME\_STATE\_MAPEDITER);

}

}

void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_ESC = 27;

if (nChar == KEY\_SPACE)

{

GotoGameState(GAME\_STATE\_RUN); // 切換至GAME\_STATE\_RUN

}

else if (nChar == KEY\_ESC)

{

if (windowsEnding.IsOpen())

{

windowsEnding.Close();

}

else if (windowsHandbook.IsOpen())

{

windowsHandbook.Close();

}

else

{

PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, 0, 0); // 關閉遊戲

}

}

}

void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point)

{

if (windowsEnding.IsCollisionClose(point))

{

windowsEnding.Close();

}

else if (windowsHandbook.IsCollisionClose(point))

{

windowsHandbook.Close();

}

else if (windowsEnding.IsOpen())

{

string endName = windowsEnding.GetCollisionButtonName(point);

if (endName != "NoButtonClick") //有點到結局

{

CAudio::Instance()->Stop("SLoWMoTIoN\_Menu");

CEndManager::Instance()->Start(endName);

GotoGameState(GAME\_STATE\_OVER);

}

}

else if (windowsHandbook.IsOpen())

{

CAudio::Instance()->Play("page\_next");

windowsHandbook.ClickWindows(point, "left");

}

else if (buttonManager.GetCollisionButtonName() != "NoButtonClick")

{

string btnName = buttonManager.GetCollisionButtonName();

CAudio::Instance()->Play("btn\_click\_3");

if (btnName == "music")

{

buttonManager.ClickButton("music");

CAudio::Instance()->SetIsPlayMusic(buttonManager.GetState("music"));

}

else if (btnName == "sound")

{

buttonManager.ClickButton("sound");

CAudio::Instance()->SetIsPlaySound(buttonManager.GetState("sound"));

}

else if (btnName == "ending")

{

windowsEnding.Open();

}

else if (btnName == "play")

{

GotoGameState(GAME\_STATE\_RUN); // 切換至GAME\_STATE\_RUN

}

else if (btnName == "tutorial")

{

windowsHandbook.Open();

}

}

}

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

{

}

void CGameStateInit::OnRButtonDown(UINT nFlags, CPoint point)

{

if (windowsHandbook.IsOpen())

{

CAudio::Instance()->Play("page\_next");

windowsHandbook.ClickWindows(point, "right");

}

}

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

{

if (windowsEnding.IsOpen())

{

windowsEnding.CollisionClose(point);

return;

}

else if (windowsHandbook.IsOpen())

{

windowsHandbook.CollisionClose(point);

return;

}

buttonManager.UpdateState(point);

}

void CGameStateInit::OnMouseWheel(UINT nFlags, short zDelta, CPoint point)

{

//往使用者方向滾 -> 往下捲動 (圖片 y -= dy)

//遠離 使用者方向滾 -> 往上捲動 (圖片 y += dy)

windowsEnding.OnScrolling(zDelta);

}

void CGameStateInit::OnMove()

{

SetCursor(AfxGetApp()->LoadCursor(IDC\_GAMECURSOR));

#pragma region mute music with menu\_bgm

if (!buttonManager.GetState("music")) //music mute

{

CAudio::Instance()->Pause();

}

else

{

CAudio::Instance()->Resume();

}

#pragma endregion

background.SetTopLeft(0, 0);

buttonManager.OnCycle();

buttonManager.SetValid(!(windowsEnding.IsOpen() || windowsHandbook.IsOpen()));

windowsEnding.OnCycle();

windowsHandbook.OnCycle();

miku.GetAction()->OnMove("run");

}

void CGameStateInit::OnShow()

{

background.ShowBitmap();

CLayerManager::Instance()->ShowLayer();

windowsEnding.OnShow();

}

#pragma endregion

#pragma region - CGame Run -

CGameStateRun::CGameStateRun(CGame\* g)

: CGameState(g)

{

#pragma region - Set eventManager -

eventManager.SetGameStateRun(this);

#pragma endregion

panel.CreatButton();

}

CGameStateRun::~CGameStateRun()

{

npcManager.Clear();

bossManager.Clear();

panel.Clear();

}

void CGameStateRun::OnBeginState()

{

CLayerManager::Instance()->Initialize();

CDialogManager::Instance()->Initialize();

CCamera::Instance()->Initialize();

mapManager.Initialize();

bossManager.Initialize();

npcManager.Initialize(mapManager.GetNowMap()); //npcManager初始化所有NPC的同時，顯示nowMap上的NPC

eventManager.Initialize();

role.Initialize();

uiManager.Initialize(&role, &bossManager);

panel.Initialize(CPoint(200, 50));

timer = CTimer(GAME\_TIME);

nowUsedTimer = &timer;

CAudio::Instance()->Stop("SLoWMoTIoN\_Menu");

CAudio::Instance()->Play("SLoWMoTIoN\_Game");

isWinXingting = false;

isWinFacaiSeed = false;

}

void CGameStateRun::OnInit() // 遊戲的初值及圖形設定

{

ShowInitProgress(33); // 接個前一個狀態的進度，此處進度視為33%

ShowInitProgress(50); // 載入圖形

#pragma region - Initialize - MapManager -

mapManager.LoadMapBitmap();

#pragma endregion

#pragma region 載入角色

role.Load();

#pragma endregion

uiManager.Load();

panel.LoadResource();

}

void CGameStateRun::OnMove() // 移動遊戲元素

{

SetCursor(AfxGetApp()->LoadCursor(IDC\_GAMECURSOR));

#pragma region - on panel -

panel.OnCycle();

if (panel.IsOpen())

{

return;

}

#pragma endregion

#pragma region - pause game state in dialoging -

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->OnCycle();

return;

}

#pragma endregion

#pragma region - timer count down -

nowUsedTimer->CountDown();

if (timer.IsTimeOut())

{

CAudio::Instance()->Stop("SLoWMoTIoN\_Game");

GoToEnd();

SwitchState(GAME\_STATE\_OVER);

}

#pragma endregion

#pragma region - run boss endprocess -

if (bossManager.targetBoss != NULL && bossManager.targetBoss->InEndProcess())

{

bossManager.targetBoss->EndProcess();

return;

}

#pragma endregion

bool roleCanMoving = true;

#pragma region - role - dead -

if (role.IsDead())

{

if (bossManager.targetBoss != NULL)

{

if (bossManager.targetBoss->GetBossId() == "Xingting")

{

CEndManager::Instance()->Start(END\_NAME\_LOSEXINGTING);

role.SetXY(0, 0);

SwitchState(GAME\_STATE\_OVER);

return;

}

if (bossManager.targetBoss->GetBossId() == BOSS\_FACAISEED)

{

CEndManager::Instance()->Start(END\_NAME\_LOSE\_FACAISEED);

role.SetXY(640, 0);

SwitchState(GAME\_STATE\_OVER);

return;

}

}

else

{

CEndManager::Instance()->Start(END\_NAME\_SALTEDFISH);

role.SetXY(0, 0);

SwitchState(GAME\_STATE\_OVER);

return;

}

}

#pragma endregion

#pragma region - boos - dead -

if (bossManager.targetBoss != NULL && bossManager.targetBoss->IsDead())

{

if (bossManager.targetBoss->GetBossId() == "Xingting")

{

isWinXingting = true;

}

if (bossManager.targetBoss->GetBossId() == BOSS\_FACAISEED)

{

isWinFacaiSeed = true;

}

bossManager.BossDead();

SwitchTimer(&timer);

}

#pragma endregion

#pragma region - end battle -

//boss 沒死 但達成其他終結battle條件

if (bossManager.targetBoss != NULL && bossManager.targetBoss->IsEnd()) //有Boss 且 Boss終結

{

bossManager.BossDead();

SwitchTimer(&timer);

}

#pragma endregion

#pragma region - Position Trigger -

PositionTrigger();

#pragma endregion

#pragma region - Moving -

int screenPosX = ScreenX(mapManager.GetX1(), role.GetX3());

#pragma region SetCamera

int mapR = mapManager.GetSplitRight();

int mapL = mapManager.GetSplitLeft();

int CameraBoundary\_R = (mapManager.GetBitmapWidth() / 2 - mapL) + (mapR - mapManager.GetBitmapWidth() / 2);//鏡頭右邊界

int CameraBoundary\_L = -CameraBoundary\_R; //鏡頭左邊界

CCamera::Instance()->SetCanMoving(true);

#pragma endregion

#pragma region --- role - collision with boss ---

if (bossManager.targetBoss != NULL)

{

if (bossManager.targetBoss != NULL)

{

if (role.IsCollisionBoss(bossManager.targetBoss)) //collision with boss

{

if (bossManager.targetBoss->GetID() != "Xingting" && bossManager.targetBoss->GetID() != "FacaiSeed")

{

CCamera::Instance()->SetCanMoving(false);

roleCanMoving = false;

}

}

}

}

#pragma endregion

#pragma region - role - collision with block -

vector<CBlock>\* block = mapManager.GetBlockVector();

bool isOnFloor = false;

int floorY = SIZE\_Y;

for (vector<CBlock>::iterator bkiter = block->begin(); bkiter != block->end(); bkiter++)

{

if (role.IsCollisionBlock(&(\*bkiter))) //collision with block

{

CCamera::Instance()->SetCanMoving(false);

roleCanMoving = false;

}

if (role.IsRoleOnBlock(&(\*bkiter)) && role.GetDrop())

{

floorY = bkiter->y;

isOnFloor = true;

}

if (role.IsCollisionBlockOnJumping(&(\*bkiter)) && !role.GetDrop())

{

role.SetDrop();

}

}

if (role.GetY2() >= SIZE\_Y - 1)

{

floorY = SIZE\_Y - 1;

isOnFloor = true;

}

#pragma endregion

#pragma region -- Moving Left --

if (role.GetMovingLeft())

{

if (role.GetX3() <= SIZE\_X / 2)

{

if (screenPosX > mapL)

{

CCamera::Instance()->AddX(-MOVE\_DISTANCE);

roleCanMoving = false;

}

if (bossManager.targetBoss != NULL)

{

bossManager.targetBoss->OnMove();

}

}

}

#pragma endregion

#pragma region -- Moving Right --

if (role.GetMovingRight())

{

if (role.GetX3() >= SIZE\_X / 2)

{

if (screenPosX < mapR)

{

CCamera::Instance()->AddX(MOVE\_DISTANCE);

//role.SetCanMoving(false);

roleCanMoving = false;

}

}

}

#pragma endregion

mapManager.OnMove();

#pragma region -- 左右界地圖檢查 - 卡邊界or換地圖 --

#pragma region --- 右邊有地圖且人物往右邊行走 ---

if (role.GetX2() >= SIZE\_X && mapManager.GetRightMap() >= 0 && role.GetMovingRight())

{

#pragma region ---- 超過邊界 - 換地圖 ----

if (role.GetX1() >= SIZE\_X)

{

ChangeMap("right");

}

#pragma endregion

}

#pragma endregion

#pragma region --- 右邊沒有地圖且人物往右邊行走 ---

if (role.GetX2() >= SIZE\_X && role.GetMovingRight() && (mapManager.GetRightMap() < 0 || bossManager.IsBattle()))

{

roleCanMoving = false;

}

#pragma endregion

#pragma region --- 左邊有地圖且人物往左邊行走 ---

if (role.GetX1() <= 0 && mapManager.GetLeftMap() >= 0 && role.GetMovingLeft())

{

#pragma region ---- 超過邊界 - 換地圖 ----

if (role.GetX2() <= 0) //超過邊界，換地圖

{

ChangeMap("left");

}

#pragma endregion

}

#pragma endregion

#pragma region --- 左邊沒有地圖且人物往左邊行走 ---

if (role.GetX1() <= 0 && role.GetMovingLeft() && (mapManager.GetLeftMap() < 0 || bossManager.IsBattle()))

{

roleCanMoving = false;

}

#pragma endregion

#pragma endregion

#pragma region -- role Moving --

role.SetCanMoving(roleCanMoving);

role.OnMove();

#pragma endregion

#pragma endregion

#pragma region - Jump -

bool isDrop = false;

if (role.GetMovingJump())

{

if ((role.GetY2() - role.GetVelocity() >= SIZE\_Y || isOnFloor) && role.GetDrop()) //on Floor

{

int ddy = 2;

role.SetMovingJump(false);

role.SetCanJumping(true);

role.GetAction()->SetAction("idle");

role.SetXY(role.GetX1(), floorY - role.Height() + ddy);

}

else //Jumping

{

role.SetMovingJump(true);

role.SetCanJumping(false);

}

}

else

{

if (!isOnFloor)

{

role.SetDrop();

}

}

#pragma endregion

#pragma region - boss cycle (move and attack) -

if (bossManager.targetBoss != NULL && bossManager.IsBattle())

{

bossManager.targetBoss->OnCycle(&role);

}

#pragma endregion

#pragma region - Collision -

scallions = role.GetScallion(); //取出蔥的指標做碰撞

passerbys = mapManager.GetPasserby(); //取出passerby指標碰撞

for (vector<CScallion\*>::iterator scallionk = scallions->begin(); scallionk != scallions->end(); )

{

for (vector<CMonster\*>::iterator passerbyj = passerbys->begin(); passerbyj != passerbys->end(); )

{

if ((\*scallionk)->IsCollision(\*passerbyj) && (\*passerbyj)->GetValid() && (\*scallionk)->IsAlive())

{

CAudio::Instance()->Play("hit");

role.AddScore((\*passerbyj)->GetScore());

(\*scallionk)->SetIsAlive(false);

mapManager.DeletePasserby(passerbyj); //從mapManager的passerbyManager中移除passerby

mapManager.AddPasserby();

break;

}

else

{

passerbyj++;

}

}

if (!(\*scallionk)->IsAlive())

{

delete \*scallionk;

\*scallionk = NULL;

scallionk = scallions->erase(scallionk);

}

else

{

scallionk++;

}

}

#pragma endregion

#pragma region - Passerby Moving -

for (vector<CMonster\*>::iterator passerbyj = passerbys->begin(); passerbyj != passerbys->end(); passerbyj++)

{

(\*passerbyj)->SetMoving();

(\*passerbyj)->OnMove();

}

#pragma endregion

#pragma region - NPC Moving -

npcManager.OnCycle(mapManager.GetNowMap(), CPoint(screenPosX, role.GetY1()));

#pragma endregion

uiManager.OnCycle(nowUsedTimer->GetTime(1));

panel.OnCycle();

}

void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

if (nChar == KEY\_A)

role.SetMovingLeft(true);

if (nChar == KEY\_D)

role.SetMovingRight(true);

if (nChar == KEY\_W)

{

bool canRoleJump = true;

for (vector<CBlock>::iterator bkiter = mapManager.GetBlockVector()->begin(); bkiter != mapManager.GetBlockVector()->end(); bkiter++)

{

if (role.GetCanJumping())

{

if (role.IsCollisionBlockOnJumping(&(\*bkiter)))

{

canRoleJump = false;

}

}

}

if (canRoleJump)

role.SetMovingUp(true);

}

if (nChar == 'U')

{

nowUsedTimer->ResetTime(5.0);

}

if (nChar == 'T')

{

for (int i = 0; i < 30; i++)

{

role.SubHp();

}

}

if (nChar == 'Y')

{

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->Stop();

}

}

if (nChar == 27) //ESC

{

if (panel.IsOpen())

panel.Close();

else

{

//Audio::Instance()->Play("open\_window");

panel.Open();

}

}

if (nChar == KEY\_Z) //dialog with npc

{

vector<CNPC\*>\* npc = npcManager.GetNpc(mapManager.GetNowMap());

for (vector<CNPC\*>::iterator npciter = npc->begin(); npciter != npc->end(); npciter++)

{

if (role.IsCollisionNPC(\*npciter))

{

(\*npciter)->RoleCollision();

(\*npciter)->SetTalked(true);

}

}

}

if (nChar == ' ') //open door

{

#pragma region - role - collision with door -

vector<CDoor>\* door = mapManager.GetDoorVector();

for (vector<CDoor>::iterator bkiter = door->begin(); bkiter != door->end(); bkiter++)

{

if (role.IsCollisionDoor(&(\*bkiter))) //collision with door

{

CAudio::Instance()->Play("door");

ChangeMap(bkiter->GetSwitchMapIndex());

}

}

#pragma endregion

}

if (nChar == 'K')

{

role.SetRoleNoSubHp();

}

}

void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

if (nChar == KEY\_A)

role.SetMovingLeft(false);

if (nChar == KEY\_D)

role.SetMovingRight(false);

if (nChar == KEY\_W)

role.SetMovingUp(false);

}

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

{

if (panel.IsOpen() && panel.IsCollisionClose(point))

{

panel.Close();

return;

}

role.SetIsFire(true & !CDialogManager::Instance()->GetDialogState());

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->Next();

}

}

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

{

role.SetIsFire(false);

if (!panel.IsOpen())

{

return;

}

if (panel.GetCollisionButtonName() != "NoButtonClick")

{

string btnName = panel.GetCollisionButtonName();

CAudio::Instance()->Play("btn\_click\_3");

if (btnName == "resume")

{

panel.Close();

}

else if (btnName == "restart")

{

panel.Close();

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->Stop();

}

GotoGameState(GAME\_STATE\_RUN);

}

else if (btnName == "menu")

{

panel.Close();

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->Stop();

}

GotoGameState(GAME\_STATE\_INIT); // 切換至GAME\_STATE\_RUN

}

else if (btnName == "exit")

{

panel.Clear();

PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, 0, 0);

}

}

}

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

{

role.SetMouseXY(point.x, point.y);

if (panel.IsOpen())

{

panel.CollisionClose(point);

panel.UpdateMouse(point);

}

}

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

{

}

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

{

}

void CGameStateRun::OnMouseWheel(UINT nFlags, short zDelta, CPoint point)

{

if (zDelta > 0)

{

role.AddScore(5);

}

else if (zDelta < 0)

{

role.AddScore(-5);

}

}

void CGameStateRun::PositionTrigger()

{

eventManager.trigger();

}

void CGameStateRun::ChangeMap(string dir)

{

int nowMap = mapManager.GetNowMap(), nextMap = nowMap;

if (dir == "left")

{

nextMap = mapManager.GetLeftMap();

mapManager.ChangeMap(mapManager.GetLeftMap(), "left");

role.SetXY(SIZE\_X - (role.GetX3() - role.GetX1()), role.GetY1() - 1);

}

else if (dir == "right")

{

nextMap = mapManager.GetRightMap();

mapManager.ChangeMap(mapManager.GetRightMap(), "right");

role.SetXY(0 - (role.GetX3() - role.GetX1()), role.GetY1() - 1);

}

bossManager.TargetBoss(mapManager.GetNowMap());

npcManager.ChangeMap(nowMap, nextMap);

}

void CGameStateRun::ChangeMap(int nextMap)

{

int nowMap = mapManager.GetNowMap();

if (nextMap != -1 && nextMap < mapManager.GetBlockMapSize()) //nextMap in blockMap

{

mapManager.ChangeMap(nextMap, "right");

bossManager.TargetBoss(mapManager.GetNowMap());

npcManager.ChangeMap(nowMap, nextMap);

role.SetXY(0, SIZE\_Y - role.Height());

}

}

void CGameStateRun::SwitchState(int state)

{

if (bossManager.targetBoss != NULL)

{

bossManager.targetBoss->Clear();

bossManager.targetBoss = NULL;

}

this->GotoGameState(state);

}

void CGameStateRun::SwitchTimer(CTimer\* swtimer)

{

nowUsedTimer = swtimer;

}

void CGameStateRun::GoToEnd()

{

#pragma region - is the role talking to all npc -

bool IsTalkedAllNpc = true;

vector<CNPC\*>\* npc = npcManager.GetNpc(mapManager.GetNowMap());

for (vector<CNPC\*>::iterator npciter = npc->begin(); npciter != npc->end(); npciter++)

{

if (!(\*npciter)->IsTalked())

{

IsTalkedAllNpc = false;

}

}

#pragma endregion

if (isWinFacaiSeed)

{

CEndManager::Instance()->Start(END\_NAME\_WIN\_FACAISEED);

}

else if (isWinXingting)

{

CEndManager::Instance()->Start(END\_NAME\_WINXINGTING);

}

else if (IsTalkedAllNpc)

{

CEndManager::Instance()->Start(END\_NAME\_SALTEDFISH);

}

else if (role.GetScore() > 999)

{

CEndManager::Instance()->Start(END\_NAME\_SALTEDFISH);

}

else

{

CEndManager::Instance()->Start(END\_NAME\_SALTEDFISH);

}

}

void CGameStateRun::OnShow()

{

panel.OnShow();

#pragma region - paint layer -

CLayerManager::Instance()->ShowLayer();

#pragma endregion

}

#pragma endregion

#pragma region - CGame Over -

CGameStateOver::CGameStateOver(CGame\* g) : CGameState(g)

{

}

void CGameStateOver::OnMove()

{

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->OnCycle();

return;

}

#pragma region - CountDown for drawGameOverImage -

if (!CEndManager::Instance()->GetEndingState())

{

timer\_exit.CountDown();

if (timer\_exit.IsTimeOut())

{

canDrawGameOverImage = true;

}

}

#pragma endregion

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 5; // 5 seconds

alpha = 0;

timer\_exit.ResetTime(1.5);

CAudio::Instance()->Play("SLoWMoTIoN\_Gameover");

CAudio::Instance()->Stop("SLoWMoTIoN\_Game");

canSwitchState = false;

canDrawGameOverImage = false;

#pragma region - init -

CLayerManager::Instance()->Initialize();

CDialogManager::Instance()->Initialize();

#pragma endregion

}

void CGameStateOver::OnInit()

{

overBitmap.LoadBitmap(".\\RES\\Map\\Gameover.bmp");

gameOverImage.SetBmp("RES\\Map\\Gameover.bmp");

gameOverImage.SetFadeInOut(60, -40);

ShowInitProgress(66); // 接個前一個狀態的進度，此處進度視為66%

ShowInitProgress(100);

}

void CGameStateOver::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

if (nChar == 'Y')

{

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->Stop();

}

}

}

void CGameStateOver::OnLButtonDown(UINT nFlags, CPoint point)

{

if (CDialogManager::Instance()->GetDialogState())

{

CDialogManager::Instance()->Next();

}

else

{

if (canSwitchState)

{

GotoGameState(GAME\_STATE\_INIT);

}

}

}

void CGameStateOver::OnShow()

{

if (CEndManager::Instance()->GetEndingState())

{

CEndManager::Instance()->OnCycle();

}

#pragma region - Draw Dialog -

CLayerManager::Instance()->ShowLayer(); //顯示layerManager

#pragma endregion

#pragma region - DrawGameOverImage when Ending end -

if (canDrawGameOverImage)

{

gameOverImage.DrawImage();

gameOverImage.FadeIn();

if (gameOverImage.GetAlpha() >= 255)

{

canSwitchState = true;

}

}

#pragma endregion

}

#pragma endregion

#pragma region - MapEditer -

CGameStateMapEditer::CGameStateMapEditer(CGame\* g) : CGameState(g)

{

}

CGameStateMapEditer::~CGameStateMapEditer()

{

}

void CGameStateMapEditer::OnInit()

{

}

void CGameStateMapEditer::OnBeginState()

{

mapEditer.Initialize();

}

void CGameStateMapEditer::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

if (nChar == 'S')

{

mapEditer.OnSave();

}

if (nChar == 'A') // map move left

{

mapEditer.SetMapMoveDir("left");

}

if (nChar == 'D') //map move right

{

mapEditer.SetMapMoveDir("right");

}

if (nChar == 9) //Key Tab

{

mapEditer.isPrintNowMap = !mapEditer.isPrintNowMap;

}

if (nChar == 46) //Key Delete

{

if (mapEditer.IsInSelectMapMode())

{

mapEditer.SetSelectMapZero();

}

else

{

mapEditer.DeleteBlock();

}

}

#pragma region - Set upMap / downMap / leftMap / rightMap -

if (!mapEditer.IsInSelectMapMode()) //不在設定地圖模式中

{

string dir[4] = { "up", "down", "left", "right" };

//Key 1, 2, 3, 4

if (nChar >= '1' && nChar <= '4')

{

mapEditer.SetSelectMapMode(dir[nChar - 48 - 1]);

}

}

#pragma endregion

}

void CGameStateMapEditer::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)

{

if (nChar == 27) //Key ESC, return to stateInit

{

if (!mapEditer.IsInSelectMapMode())

{

mapEditer.StoreMapInformaion();

mapEditer.CreateReloadMapInformation();

GotoGameState(GAME\_STATE\_INIT);

}

else

{

mapEditer.SetSelectMapMode("none");

}

}

if (nChar == 'A')

{

mapEditer.SetMapMoveDir("cleft");

}

if (nChar == 'D')

{

mapEditer.SetMapMoveDir("cright");

}

}

void CGameStateMapEditer::OnLButtonDown(UINT nFlags, CPoint point)

{

if (mapEditer.IsInSelectMapMode()) //在選擇地圖的模式中

{

mapEditer.ClickArrow(point);

}

else

{

mapEditer.SetMouseState(true);

mapEditer.SelectBlock(point);

}

}

void CGameStateMapEditer::OnLButtonUp(UINT nFlags, CPoint point)

{

if (!mapEditer.IsInSelectMapMode())

{

mapEditer.SetMouseState(false);

}

}

void CGameStateMapEditer::OnMouseMove(UINT nFlags, CPoint point)

{

mousePoint = point;

}

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

{

}

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

{

}

void CGameStateMapEditer::OnMove()

{

if (mapEditer.IsInSelectMapMode()) //選擇地圖中 - 下面功能不動作

{

return;

}

#pragma region - 判斷檔案是否存在 -

bool isDataExist = false; //先假設檔案不存在

fstream dataFileName;

dataFileName.open("RES\\Map\\FileName.txt");

#pragma region -- 檔案存在 --

if (dataFileName.is\_open()) //有檔案就load他

{

vector<string> fileContext;

fileContext.clear();

string fileName;

while (dataFileName >> fileName)

{

fileContext.push\_back(fileName);

}

mapEditer.AddImage(fileContext);

isDataExist = true; //檔案存在

}

#pragma endregion

dataFileName.close(); //關檔

#pragma region -- 檔案真的存在 - 刪除檔案^^ --

if (isDataExist)

{

DeleteFile("RES\\Map\\FileName.txt");

}

#pragma endregion

#pragma endregion

#pragma region - 拖曳block -

if (mapEditer.GetMouseState()) //滑鼠按住中

{

mapEditer.DragBlock(mousePoint);

}

#pragma endregion

mapEditer.OnMove();

}

void CGameStateMapEditer::OnShow()

{

mapEditer.OnShow();

PaintText("MapEditer", 0, 445, "微軟正黑體", 20, RGB(0, 0, 0), RGB(255, 255, 255)); //Text ,位置, 文字字形(sp), 文字大小, 文字顏色, 背景顏色

if (mapEditer.isPrintNowMap)

{

char\* address = ConvertCharPointToString(mapEditer.GetNowMap());

PaintText(address, 320, 0, "微軟正黑體", 20, RGB(0, 0, 0), RGB(255, 255, 255)); //Text ,位置, 文字字形(sp), 文字大小, 文字顏色, 背景顏色

delete address;

}

if (mapEditer.IsInSelectMapMode())

{

char\* address = ConvertCharPointToString(mapEditer.GetSelectMapMode());

PaintText(address, 90, 0, "微軟正黑體", 20, RGB(0, 0, 0), RGB(255, 255, 255)); //Text ,位置, 文字字形(sp), 文字大小, 文字顏色, 背景顏色

delete address;

}

}

#pragma endregion

}//End

1. CEraser.h

#pragma once

#include "CBoss.h"

#include "CLibrary.h"

namespace game\_framework {

#pragma region - CEraser -

class CBlock;

class CEraser

{

public:

CEraser();

virtual ~CEraser();

int GetX1(); // 左上角 x 座標

int GetY1(); // 左上角 y 座標

int GetX3(); // 中心點 X 座標

int GetX2(); // 右下角 x 座標

int GetY2(); // 右下角 y 座標

int GetY3(); // 中心點 Y 座標

int GetScore();

int Height();

int Width();

bool GetMovingLeft(); // 方方是否在往左走

bool GetMovingRight(); // 方方是否再往右走

void Initialize(); // 初始化

// 由路徑載入圖形(資料夾, name, 張數)

void LoadBitmap(string, string, int, COLORREF);

void LoadBitmap(BitmapPath);

void OnMove(); // 移動擦子

void OnShow(); // 將擦子圖形貼到畫面

CAnimate\* GetAnimate();

void SetMovingDown(bool flag); // 設定是否正在往下移動

void SetMovingLeft(bool flag); // 設定是否正在往左移動

void SetMovingRight(bool flag); // 設定是否正在往右移動

void SetMovingUp(bool flag); // 設定是否正在往上移動

void SetXY(int nx, int ny); // 設定擦子左上角座標

void SetCanMoving(bool);

bool GetCanMoving();

CLayer layer;

bool GetValid();

void SetValid(bool);

#pragma region - collision -

virtual void ResetCollisionRect();

bool IsCollisionBlock(CBlock\*); //一般左右 跟block碰撞

bool IsRoleOnBlock(CBlock\*); //站在方塊上

#pragma endregion

protected:

CAnimate animation; // 擦子的動畫

int height, width; // 擦子的高、寬

int x, y; // 擦子左上角座標

int score;

bool isMovingDown; // 是否正在往下移動

bool isMovingLeft; // 是否正在往左移動

bool isMovingRight; // 是否正在往右移動

bool isMovingUp; // 是否正在往上移動

bool canMoving; // 是否可以移動

CRect collisionRect;

CRect collisionDownRect;

private:

int move\_distance = MOVE\_DISTANCE;

};

#pragma endregion

#pragma region - CNPC -

class CNPC : public CEraser

{

public:

CNPC();

CNPC(CPoint, BitmapPath, string, double);

virtual ~CNPC();

void SetCurrentXY(int, int);

void SetXY();

void LookRole(CPoint);

virtual void OnCycle(CPoint);

virtual void OnMove();

virtual void Initialize();

virtual void RoleCollision() {}; //碰撞到後要做的事情

virtual void SetValid(bool);

void SetTalked(bool f) {

isTalked = f;

};

bool IsTalked() {

return isTalked;

};

protected:

#pragma region - init information -

CPoint initPoint;

BitmapPath initLoadPath;

double initResetTime;

#pragma endregion

CAnimate leftAnimate;

CAnimate rightAnimate;

int currentX, currentY;

string id;

string faceTo;

bool isTalked;

};

#pragma endregion

#pragma region - CRole -

class CBlackHole;

class CDoor;

class CRole : public CEraser

{

friend class CScallion;

public:

CRole();

~CRole();

int GetX3(); // 中新點 X 座標

int GetX2(); // 擦子右下角 x 座標

int GetY2(); // 擦子右下角 y 座標

int GetY3(); // 中心點 Y 座標

int Height();

int Width();

void SetXY(int \_x, int \_y); // 設定擦子左上角座標

bool GetValid();

void SetValid(bool);

void Load();

void LoadasMascot();

void LoadAction(string, BitmapPath); //載入動應動作

void OnMove();

bool GetMovingJump();

void SetMovingJump(bool);

bool GetCanJumping();

void SetCanJumping(bool);

void SetMouseXY(int, int);

void AddScore(int);

void SubHp();

int GetHp() {

return hp;

};

void SubEq();

int GetEq() {

return eq;

};

void Initialize();

bool GetIsFire();

void SetIsFire(bool);

void Fire(int, int); //傳入滑鼠座標

void SetDrop();

bool GetDrop();

int GetVelocity();

void SetVelocity(int v) {

velocity = v;

};

bool IsMoving();

bool IsDead() {

return isDead;

};

bool IsZZ() {

return isZZ;

};

void SetRoleNoSubHp() {

isRoleNoSubHp = !isRoleNoSubHp;

};

CAction\* GetAction() {

return &action;

};

#pragma region - Collision -

bool IsCollisionBoss(CBoss\*);

#pragma region -- Collision - Block --

bool IsCollisionBlockOnJumping(CBlock\*); //跳起來撞到方塊

#pragma endregion

bool IsCollisionDoor(CDoor\*);

bool IsCollisionLevel4(CScallion\*);

bool IsCollisionNPC(CNPC\*);

bool IsCollisionBlackHole(CBlackHole\*);

bool IsCollisionBlackHoleCenter(CBlackHole\*);

bool IsCollisionRay(CRay\*);

#pragma endregion

vector<CScallion\*>\* GetScallion();

void SetCatched(bool flag) {

isCatched = flag;

};

bool IsCatched() {

return isCatched;

};

protected:

vector<CScallion\*> scallion;

bool isJumping; // 是否正在跳躍

bool canJumping; // 是否可以跳躍

bool isFire; // 是否正在射擊

int init\_velocity; // 往上的初速度

int velocity; // 速度

int gravity; // 重力

char last\_right\_left;

int mouse\_x, mouse\_y;

int move\_distance = MOVE\_DISTANCE; //每次移動的距離

CTimer shoot\_cd;

CMovingBitmap decisionPoint;

CRect collisionTopRect;

CRect collisionDoorRect;

void ResetCollisionRect();

private:

CAction action;

#pragma region - init information -

int inithp = 20;

int initEq = 20;

bool isLoaded;

string now\_action;

#pragma endregion

int hp, eq;

bool isCatched;

bool isDead;

bool isZZ;

bool isRoleNoSubHp;

};

#pragma endregion

#pragma region - CMonster -

class CMonster : public CEraser

{

friend class CScallion;

public:

CMonster();

CMonster(int, int);

void SetBlock(vector<CBlock>\* bkvector) {

blockVector = bkvector;

}

void SetXY(int, int);

void SetScore(int);

void SetMoving();

virtual void OnMove();

virtual ~CMonster();

protected:

int initX;

CTimer moveTimer;

CTimer stopTimer;

CTimer recreateTimer;

int move\_distance = 2;

CAnimate leftAnimate;

CAnimate rightAnimate;

string faceto;

bool onFloor;

bool moveleft, moveright;

vector<CBlock>\* blockVector;

void ResetCollisionRect();

};

#pragma region - CMonster1 - LUKA -

class CMonsterType1 : public CMonster

{

public:

CMonsterType1();

CMonsterType1(int, int, BitmapPath, int);

~CMonsterType1();

};

#pragma endregion

#pragma region - CMonster2 - mushroom -

class CMonsterType2 : public CMonster

{

public:

CMonsterType2();

CMonsterType2(int, int, BitmapPath, int);

void OnMove();

~CMonsterType2();

};

#pragma endregion

#pragma endregion

#pragma region - CNPC - No.1 -

//CNPC1 > 普通的NPC (對話NPC)

class CNPC1 : public CNPC

{

public:

CNPC1();

CNPC1(CPoint, BitmapPath, string, string, double resetTime = 0.05); //座標 路徑 id 對話文本 一個動畫的影格的時間

~CNPC1();

void RoleCollision();

private:

string txt;

};

#pragma endregion

#pragma region - CNPC - No.3 -

//CNPC3 > 音樂播放NPC

class CNPC3 : public CNPC

{

public:

CNPC3();

CNPC3(CPoint, BitmapPath, string, string, string); //座標 路徑 ID 音樂 對話文本

~CNPC3();

void RoleCollision();

void Initialize();

void SetValid(bool);

void OnMove();

private:

string openTxt;

string openMusic;

CMovingBitmap headphoneIcon;

bool isLoadHeadPhoneIcon;

};

#pragma endregion

}

1. CEraser.cpp

#pragma once

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CEraser.h"

#include "CBall.h"

#include <sstream>

#include "CBoss.h"

#include "CManager.h"

#include "CLibrary.h"

using namespace myLibrary;

namespace game\_framework {

#pragma region - CEraser -

CEraser::CEraser()

{

Initialize();

}

CEraser::~CEraser()

{

}

void CEraser::Initialize()

{

const int X\_POS = 0;

const int Y\_POS = -160;

x = X\_POS;

y = Y\_POS;

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

canMoving = true;

layer.SetLayer(5);

score = 0;

}

int CEraser::GetX1()

{

return x;

}

int CEraser::GetY1()

{

return y;

}

int CEraser::GetX2()

{

return x + animation.Width();

}

int CEraser::GetY2()

{

return y + animation.Height();

}

int CEraser::GetY3()

{

return y + animation.Height() / 2;

}

int CEraser::GetScore()

{

return score;

}

int CEraser::GetX3()

{

return x + animation.Width() / 2;

}

int CEraser::Height()

{

return height;

}

int CEraser::Width()

{

return width;

}

bool CEraser::GetMovingLeft()

{

return isMovingLeft;

}

bool CEraser::GetMovingRight()

{

return isMovingRight;

}

void CEraser::LoadBitmap(string ziliaojia, string name, int number, COLORREF color)

{

animation.LoadBitmap(ziliaojia, name, number, color);

animation.SetTopLeft(x, y);

height = animation.Height();

width = animation.Width();

}

void CEraser::LoadBitmap(BitmapPath loadpath)

{

animation.LoadBitmap(loadpath.ziliaojia, loadpath.name, loadpath.number, loadpath.color);

}

void CEraser::OnMove()

{

const int STEP\_SIZE = move\_distance;

int dir = -1;

if (isMovingUp)

{

dir = 0;

}

if (isMovingRight)

{

dir = 1;

if (canMoving)

x += STEP\_SIZE;

}

if (isMovingDown)

{

dir = 2;

if (canMoving)

y += STEP\_SIZE;

}

if (isMovingLeft)

{

dir = 3;

if (canMoving)

x -= STEP\_SIZE;

}

animation.SetTopLeft(x, y);

animation.OnMove(dir);

}

void CEraser::OnShow()

{

}

void CEraser::SetMovingDown(bool flag)

{

isMovingDown = flag;

canMoving = flag | isMovingLeft | isMovingRight | isMovingUp;

}

void CEraser::SetMovingLeft(bool flag)

{

isMovingLeft = flag;

canMoving = flag | isMovingDown | isMovingRight | isMovingUp;

}

void CEraser::SetMovingRight(bool flag)

{

isMovingRight = flag;

canMoving = flag | isMovingDown | isMovingLeft | isMovingUp;

}

void CEraser::SetMovingUp(bool flag)

{

isMovingUp = flag;

canMoving = flag | isMovingLeft | isMovingRight | isMovingDown;

}

void CEraser::SetCanMoving(bool \_canMoving)

{

canMoving = \_canMoving;

}

bool CEraser::GetCanMoving()

{

return canMoving;

}

bool CEraser::GetValid()

{

return animation.GetValid();

}

void CEraser::SetValid(bool flag)

{

animation.SetValid(flag);

}

void CEraser::SetXY(int nx, int ny)

{

x = nx;

y = ny;

}

CAnimate\* CEraser::GetAnimate()

{

animation.SetTopLeft(x, y);

return &animation;

}

void CEraser::ResetCollisionRect()

{

if (!animation.IsNull())

{

collisionRect = animation.GetRect();

collisionDownRect = animation.GetRect();

}

}

#pragma endregion

#pragma region - CRole -

CRole::CRole() : CEraser()

{

isLoaded = false;

}

CRole::~CRole()

{

for (vector<CScallion\*>::iterator it = scallion.begin(); it != scallion.end(); it++)

{

delete (\*it);

(\*it) = NULL;

}

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

}

int CRole::GetX2()

{

return x + action.Width();

}

int CRole::GetY2()

{

return y + action.Height();

}

int CRole::GetY3()

{

return y + action.Height() / 2;

}

int CRole::GetX3()

{

return x + action.Width() / 2;

}

int CRole::Height()

{

return action.Height();

}

int CRole::Width()

{

return action.Width();

}

void CRole::SetXY(int \_x, int \_y)

{

x = \_x;

y = \_y;

action.SetTopLeft(\_x, \_y);

}

bool CRole::GetValid()

{

return action.GetValid();

}

void CRole::SetValid(bool \_flag)

{

action.SetValid(\_flag);

}

void CRole::Load()

{

LoadAction("idle", BitmapPath("RES\\Role\\miku\\idle", "idle", 19, RGB(150, 200, 250)));

LoadAction("run", BitmapPath("RES\\Role\\miku\\run", "run", 7, RGB(150, 200, 250)));

LoadAction("jump", BitmapPath("RES\\Role\\miku\\jump", "jump", 5, RGB(150, 200, 250)));

decisionPoint.LoadBitmap("RES\\Role\\miku\\cursor.bmp", RGB(214, 214, 214));

}

void CRole::LoadasMascot()

{

LoadAction("idle", BitmapPath("RES\\Role\\miku\\idle", "idle", 19, RGB(150, 200, 250)));

LoadAction("run", BitmapPath("RES\\Role\\miku\\run", "run", 7, RGB(150, 200, 250)));

}

void CRole::LoadAction(string \_action, BitmapPath \_loadpath)

{

action.LoadAction(\_action, \_loadpath);

}

void CRole::OnMove()

{

const int STEP\_SIZE = move\_distance;

int dir = 0;

if (isMovingUp)

{

dir = 1;

if (canJumping)

{

CAudio::Instance()->Play("jump");

isJumping = true;

}

}

if (isMovingRight)

{

dir = 2;

if (canMoving)

x += STEP\_SIZE;

action.SetFaceTo("\_R");

}

if (isMovingDown)

{

dir = 3;

if (canMoving)

y += STEP\_SIZE;

}

if (isMovingLeft)

{

dir = 4;

if (canMoving)

x -= STEP\_SIZE;

action.SetFaceTo("\_L");

}

if (isJumping)

{

if (canJumping)

{

velocity = init\_velocity;

canJumping = false;

}

else

{

//jump

if (velocity > 0)

{

velocity -= gravity;

y -= velocity;

}

else

{

velocity -= gravity;

y -= velocity;

}

}

}

if (isFire)

{

Fire(mouse\_x, mouse\_y);

}

shoot\_cd.CountDown(); //設置射擊的 CD時間

if (isJumping)

{

decisionPoint.SetTopLeft(GetX3() - 5, GetY3() + 16);

}

else

{

decisionPoint.SetTopLeft(GetX3() - 5, GetY3());

}

action.SetTopLeft(x, y);

if (isJumping)

{

action.OnMove("jump");

action.SetTopLeft(x, y - (action.GetActionHeight("jump") - action.GetActionHeight("idle")));

}

else if (isMovingLeft || isMovingRight)

{

action.OnMove("run");

}

else

{

action.OnMove("idle");

}

for (unsigned int i = 0; i < scallion.size(); i++)

{

scallion[i]->OnMove();

}

}

bool CRole::GetMovingJump()

{

return isJumping;

}

void CRole::SetMovingJump(bool flag)

{

isJumping = flag;

}

bool CRole::GetCanJumping()

{

return canJumping;

}

void CRole::SetCanJumping(bool flag)

{

canJumping = flag;

}

void CRole::SetMouseXY(int \_x, int \_y)

{

mouse\_x = \_x;

mouse\_y = \_y;

}

bool CRole::GetIsFire()

{

return isFire;

}

void CRole::SetIsFire(bool flag)

{

isFire = flag;

}

void CRole::Fire(int mouseX, int mouseY)

{

if (shoot\_cd.IsTimeOut())

{

CAudio::Instance()->Play("throw");

CScallion\* newCScallion = new CScallion(BitmapPath("RES\\Object\\Scallions", "scallion", 2), CPoint(GetX3(), GetY1()), CPoint(mouseX, mouseY)); //先創建一個蔥的物件

scallion.push\_back(newCScallion); //將蔥放進vector

shoot\_cd.ResetTime(0.33);

}

}

void CRole::SetDrop()

{

SetMovingJump(true);

SetCanJumping(false);

SetVelocity(0);

}

bool CRole::GetDrop()

{

return velocity < 0;

}

int CRole::GetVelocity()

{

return velocity;

}

bool CRole::IsMoving()

{

return (isMovingRight || isMovingLeft);

}

#pragma region -- Collision --

bool CRole::IsCollisionBoss(CBoss\* boss)

{

ResetCollisionRect();

return IsRectCollision(collisionRect, boss->GetAnimate()->GetRect());

}

bool CEraser::IsCollisionBlock(CBlock\* block) //get iterator in mygame.cpp

{

ResetCollisionRect();

return IsRectCollision(collisionRect, block->bmp.GetRect());

}

bool CEraser::IsRoleOnBlock(CBlock\* block)

{

ResetCollisionRect();

return IsRectCollision(collisionDownRect, block->bmp.GetRect());

}

bool CRole::IsCollisionBlockOnJumping(CBlock\* block)

{

ResetCollisionRect();

return IsRectCollision(collisionTopRect, block->bmp.GetRect());

}

bool CRole::IsCollisionDoor(CDoor\* door)

{

ResetCollisionRect();

return IsRectCollision(collisionDoorRect, door->bmp.GetRect());

}

bool CRole::IsCollisionLevel4(CScallion\* level4)

{

return IsRectCollision(decisionPoint.GetRect(), level4->GetAnimate()->GetRect());

}

bool CRole::IsCollisionNPC(CNPC\* npc)

{

return IsRectCollision(action.GetRect(), npc->GetAnimate()->GetRect());

}

bool CRole::IsCollisionBlackHole(CBlackHole\* blackHole)

{

return IsRectCollision(decisionPoint.GetRect(), blackHole->GetCenterRect());

}

bool CRole::IsCollisionBlackHoleCenter(CBlackHole\* blackHole)

{

return IsRectCollision(decisionPoint.GetRect(), blackHole->GetCenterRect());

}

bool CRole::IsCollisionRay(CRay\* ray)

{

return IsRectCollision(decisionPoint.GetRect(), ray->GetAnimate()->GetRect());

}

#pragma endregion

void CRole::AddScore(int \_score)

{

score += \_score;

}

void CRole::SubHp()

{

if (!isRoleNoSubHp)

{

hp--;

}

if (hp <= 0)

{

isDead = true;

}

}

void CRole::SubEq()

{

eq--;

if (eq <= 0)

{

isZZ = true;

}

}

void CRole::Initialize()

{

action.Initialize();

#pragma region Normal Setting

CEraser::Initialize();

isJumping = true;

canJumping = false;

isFire = false;

hp = inithp;

eq = initEq;

score = 0;

const int INIT\_VELOCITY = 23; //設定初速度

const int GRAVITY = 1; //設定重力

init\_velocity = velocity = INIT\_VELOCITY;

gravity = GRAVITY;

mouse\_x, mouse\_y = 0;

x = 0;

y = -Height();

shoot\_cd = CTimer(0); //初始化射擊冷卻時間

isCatched = false;

isDead = false;

isZZ = false;

isRoleNoSubHp = false;

SetValid(true);

#pragma endregion

#pragma region Load

if (!isLoaded)

{

isLoaded = true;

}

#pragma endregion

collisionRect = CRect(CPoint(x - move\_distance, y - move\_distance), CPoint(x + Width() + move\_distance, y + Height() + move\_distance));

CLayerManager::Instance()->AddObject(&decisionPoint, layer.GetLayer() + 1);

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

}

vector<CScallion\*>\* CRole::GetScallion()

{

return &scallion;

}

void CRole::ResetCollisionRect()

{

#pragma region - collision Rect - Right and Left -

collisionRect = action.GetRect();

int dx = move\_distance;

collisionRect.bottom -= 10;

if (isMovingRight)

{

//Test地方調整往右/往左的判斷寬度 (原本人物寬度 + dx，改成從中心點算起的寬度 + dx)

collisionRect.left = GetX3(); //Test

collisionRect.right += dx;

}

else if (isMovingLeft)

{

collisionRect.left -= dx;

collisionRect.right = GetX3(); //Test

}

#pragma endregion

#pragma region - collision Rect - Down -

collisionDownRect = action.GetRect();

//調整 下方判斷方塊的高度，以及縮減左/右的寬度

collisionDownRect.top = GetY2() - 5; //調整判斷的高度

collisionDownRect.right -= dx \* 3;

collisionDownRect.left += dx \* 3;

int dy = move\_distance ;

if (GetDrop())

{

collisionDownRect.bottom += dy;

}

#pragma endregion

#pragma region - collision Rect - Top -

collisionTopRect = action.GetRect();

//縮減左/右的寬度

collisionTopRect.right -= dx \* 3;

collisionTopRect.left += dx \* 3;

collisionTopRect.bottom = GetY3(); //調整判斷方塊高度

if (!GetMovingJump()) //沒跳躍

{

collisionTopRect.top -= dy \* 3;

}

else if (GetMovingJump() && !GetDrop()) //跳躍中

{

collisionTopRect.top += dy \* 3;

}

#pragma endregion

#pragma region - collsion Rect - Door -

collisionDoorRect = decisionPoint.GetRect();

collisionDoorRect.top -= 20;

collisionDoorRect.bottom += 20;

collisionDoorRect.left -= 20;

collisionDoorRect.right += 20;

#pragma endregion

}

#pragma endregion

#pragma region - CMonster -

CMonster::CMonster() : CEraser()

{

initX = 0;

x = initX;

y = 300;

layer.SetLayer(4);

score = 0;

moveTimer = CTimer(0.5);

stopTimer = CTimer(1.5);

SetValid(false);

recreateTimer = CTimer((double)GetRandom(1, 4) / 2.0);

}

CMonster::CMonster(int \_x, int \_y) : CEraser()

{

initX = \_x;

x = initX;

y = \_y;

layer.SetLayer(4);

CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());

score = 0;

moveTimer = CTimer(0.5);

stopTimer = CTimer(1.5);

SetValid(false);

recreateTimer = CTimer((double)GetRandom(1, 2) / 2.0);

onFloor = false;

blockVector = NULL;

}

void CMonster::SetXY(int \_x, int \_y)

{

initX = \_x;

if (\_y + height >= SIZE\_Y - 2)

{

y = SIZE\_Y - height - 2;

}

else

{

y = \_y;

}

int dx = CCamera::Instance()->GetX();

x = initX - dx;

animation.SetTopLeft(x, y);

}

void CMonster::SetScore(int \_score)

{

score = \_score;

}

void CMonster::SetMoving()

{

if (GetValid() == false)

{

recreateTimer.CountDown();

if (!recreateTimer.IsTimeOut())

{

return;

}

}

if ( !moveTimer.IsTimeOut())

{

moveTimer.CountDown();

}

else

{

SetCanMoving(false);

stopTimer.CountDown();

if (stopTimer.IsTimeOut())

{

moveleft = moveright = false;

int randomNumber = GetRandom(1, 10);

if (randomNumber % 2 == 0)

{

moveleft = true;

moveright = false;

}

else

{

moveleft = false;

moveright = true;

}

double resetStopTime = GetRandom(1, 10) / 10;

SetCanMoving(true);

moveTimer.ResetTime(0.1);

stopTimer.ResetTime(resetStopTime);

}

}

}

void CMonster::OnMove()

{

if (!recreateTimer.IsTimeOut())

return;

const int STEP\_SIZE = move\_distance;

bool nowOnFloor = false;

if (blockVector != NULL)

{

for (vector<CBlock>::iterator bkiter = blockVector->begin(); bkiter != blockVector->end(); bkiter++)

{

if (IsPointInRect(GetY2(), bkiter->bmp.GetRect()))

{

y += (GetY2() - bkiter->y) + 2;

}

if (IsCollisionBlock(&(\*bkiter)))

{

moveright = false;

moveleft = false;

SetCanMoving(false);

}

if ((IsRoleOnBlock(&(\*bkiter))))

{

nowOnFloor = true;

}

}

}

if (GetY2() >= SIZE\_Y - 2)

{

nowOnFloor = true;

}

if (nowOnFloor)

{

SetValid(true);

SetMovingDown(false);

onFloor = true;

}

else

{

moveleft = false;

moveright = false;

SetMovingDown(true);

}

SetMovingLeft(moveleft);

SetMovingRight(moveright);

if (isMovingRight)

{

if (canMoving)

initX += STEP\_SIZE;

}

if (isMovingDown)

{

if (canMoving)

{

int dy = STEP\_SIZE \* 3;

y += dy;

}

}

if (isMovingLeft)

{

if (canMoving)

initX -= STEP\_SIZE;

}

SetXY(initX, y);

animation.OnMove();

}

CMonster::~CMonster()

{

}

void CMonster::ResetCollisionRect()

{

//reset passerby's collision rect

#pragma region - collision Rect - Right and Left -

collisionRect = animation.GetRect();

int dx = 4;

collisionRect.bottom -= 2;

if (isMovingRight)

{

//Test地方調整往右/往左的判斷寬度 (原本人物寬度 + dx，改成從中心點算起的寬度 + dx)

collisionRect.left = GetX3(); //Test

collisionRect.right += dx;

}

else if (isMovingLeft)

{

collisionRect.left -= dx;

collisionRect.right = GetX3(); //Test

}

#pragma endregion

#pragma region - collision Rect - Down -

collisionDownRect = animation.GetRect();

//調整 下方判斷方塊的高度，以及縮減左/右的寬度

collisionDownRect.top = GetY2() - 5; //調整判斷的高度

int dy = 4;

collisionDownRect.bottom += dy;

#pragma endregion

}

#pragma region - CMonsterType1 - Luka -

CMonsterType1::CMonsterType1() : CMonster()

{

}

CMonsterType1::CMonsterType1(int \_x, int \_y, BitmapPath \_loadPath, int \_score) : CMonster(\_x, \_y)

{

SetScore(\_score);

LoadBitmap(\_loadPath.ziliaojia, \_loadPath.name, \_loadPath.number, \_loadPath.color);

}

CMonsterType1::~CMonsterType1()

{

}

#pragma endregion

#pragma region - CMonsterType2 - mushroom -

CMonsterType2::CMonsterType2() : CMonster()

{

}

CMonsterType2::CMonsterType2(int \_x, int \_y, BitmapPath \_loadPath, int \_score) : CMonster(\_x, \_y)

{

SetScore(\_score);

BitmapPath leftPath = \_loadPath;

BitmapPath rightPath = \_loadPath;

leftPath.ziliaojia += "\\L";

rightPath.ziliaojia += "\\R";

LoadBitmap(leftPath.ziliaojia, leftPath.name, leftPath.number, leftPath.color);

leftAnimate = animation;

rightAnimate.LoadBitmap(rightPath.ziliaojia, rightPath.name, rightPath.number, rightPath.color);

leftAnimate.ResetDelayTime(0.1);

rightAnimate.ResetDelayTime(0.1);

leftAnimate.CopyAnimateInformation(&animation);

rightAnimate.CopyAnimateInformation(&animation);

animation = leftAnimate;

faceto = "left";

}

CMonsterType2::~CMonsterType2()

{

}

void CMonsterType2::OnMove()

{

if (GetMovingLeft() && faceto != "left")

{

leftAnimate.CopyAnimateInformation(&animation);

animation = leftAnimate;

faceto = "left";

}

else if (GetMovingRight() && faceto != "right")

{

rightAnimate.CopyAnimateInformation(&animation);

animation = rightAnimate;

faceto = "right";

}

CMonster::OnMove();

}

#pragma endregion

#pragma endregion

#pragma region - CNPC -

CNPC::CNPC() : CEraser()

{

}

CNPC::CNPC(CPoint \_point, BitmapPath \_loadPath, string \_id, double resetTime)

{

initPoint = \_point;

initLoadPath = \_loadPath;

id = \_id;

initResetTime = resetTime;

}

void CNPC::Initialize()

{

#pragma region - load animation (only once) -

if (animation.IsNull())

{

BitmapPath leftPath = initLoadPath;

leftPath.ziliaojia += "\\L\\";

BitmapPath rightPath = initLoadPath;

rightPath.ziliaojia += "\\R\\";

LoadBitmap(leftPath);

leftAnimate = animation;

rightAnimate.LoadBitmap(rightPath);

leftAnimate.ResetDelayTime(initResetTime);

rightAnimate.ResetDelayTime(initResetTime);

leftAnimate.CopyAnimateInformation(&animation);

rightAnimate.CopyAnimateInformation(&animation);

animation = leftAnimate;

faceTo = "left";

}

#pragma endregion

SetCurrentXY(initPoint.x, initPoint.y);

layer.SetLayer(NPC\_LAYER);

animation.SetValid(false);

CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());

isTalked = false;

}

void CNPC::SetValid(bool flag)

{

animation.SetValid(flag);

}

void CNPC::SetCurrentXY(int \_x, int \_y)

{

currentX = \_x;

currentY = \_y;

SetXY();

}

void CNPC::SetXY()

{

x = currentX;

y = currentY;

int dx = CCamera::Instance()->GetX();

x = currentX - dx;

animation.SetTopLeft(x, y);

}

void CNPC::LookRole(CPoint rolePoint)

{

if (currentX < rolePoint.x && faceTo != "right") //look right

{

rightAnimate.CopyAnimateInformation(&animation);

animation = rightAnimate;

faceTo = "right";

}

else if (currentX >= rolePoint.x && faceTo != "left")

{

leftAnimate.CopyAnimateInformation(&animation);

animation = leftAnimate;

faceTo = "left";

}

}

void CNPC::OnCycle(CPoint rolePoint)

{

LookRole(rolePoint);

OnMove();

}

void CNPC::OnMove()

{

int dx = CCamera::Instance()->GetX();

SetCurrentXY(currentX, currentY);

animation.OnMove();

}

CNPC::~CNPC()

{

}

#pragma endregion

#pragma region - CNPC1 - No.1 -

CNPC1::CNPC1()

{

}

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

{

txt = \_txt;

}

CNPC1::~CNPC1()

{

}

void CNPC1::RoleCollision()

{

CDialogManager::Instance()->Start(txt);

}

#pragma endregion

#pragma region - CNPC3 - No.3 -

CNPC3::CNPC3()

{

}

CNPC3::CNPC3(CPoint \_point, BitmapPath \_loadPath, string \_id, string \_music, string \_txt) : CNPC(\_point, \_loadPath, \_id, 0.1)

{

openMusic = \_music;

openTxt = \_txt;

isLoadHeadPhoneIcon = false;

}

CNPC3::~CNPC3()

{

}

void CNPC3::RoleCollision()

{

CAudio::Instance()->Stop("SLoWMoTIoN\_Game");

CAudio::Instance()->Play(openMusic, true);

CDialogManager::Instance()->OpenMusicPlayer(openMusic);

CDialogManager::Instance()->Start(openTxt);

}

void CNPC3::Initialize()

{

CNPC::Initialize();

if (!isLoadHeadPhoneIcon)

{

headphoneIcon.LoadBitmap("RES\\NPC\\headphone3.bmp", RGB(255, 255, 255));

isLoadHeadPhoneIcon = true;

}

headphoneIcon.SetTopLeft(GetX3() - headphoneIcon.Width() / 2, GetY1() - headphoneIcon.Height() - 10);

headphoneIcon.SetValid(false);

CLayerManager::Instance()->AddObject(&headphoneIcon, layer.GetLayer());

}

void CNPC3::SetValid(bool flag)

{

animation.SetValid(flag);

headphoneIcon.SetValid(flag);

}

void CNPC3::OnMove()

{

CNPC::OnMove();

headphoneIcon.SetTopLeft(GetX3() - headphoneIcon.Width() / 2, GetY1() - headphoneIcon.Height() - 10);

}

#pragma endregion

}

1. CBall.h

#pragma once

#include "CLibrary.h"

#include "CEraser.h"

#include "CBoss.h"

namespace game\_framework {

#pragma region - ball -

class CBall

{

public:

CBall();

bool IsAlive(); // 是否活著

void SetCurrentXY(double, double);

void SetXY(int nx, int ny); // 設定圓心的座標

void SetIsAlive(bool alive); // 設定是否活著

CPoint GetCenterPoint() {

return CPoint(x + animation.Width() / 2, y + animation.Height() / 2);

};

void LoadBitmap(string, string, int);

void LoadBitmap(BitmapPath);

CAnimate\* GetAnimate();

CLayer layer;

protected:

CAnimate animation;

double currentX, currentY; //當前的座標

int x, y; // 圓心的座標

bool is\_alive; // 是否活著

};

#pragma endregion

#pragma region - CScallion -

class CScallion : public CBall

{

public:

CScallion();

CScallion(BitmapPath, CPoint, CPoint, int = 2); //給兩CPoint + int 為參數時，第一個為目前座標，第二個為滑鼠座標，最後為重力參數

CScallion(BitmapPath, CPoint, int, int); //若不給重力參數，第一個CPoint不變，但後面變為「x初速, y初速」

void OnMove();

void OnShow();

void SetInitVelocity(int, int, int, int, int = 5);

void SetInitVelocity(int, int);

bool IsCollision(CMonster);

bool IsCollision(CMonster\*);

bool IsCollision(CBoss\*);

bool IsCollision(CRect);

void Clear();

void Initialize(CPoint);

int GetAtk() {

return atk;

};

void SetAtk(int k) {

atk = k;

};

protected:

int atk = 1;

int velocity\_x;

int velocity\_y;

int gravity;

int direction; //丟出時面對的方向

};

#pragma endregion

#pragma region - CBlackHole -

class CBlackHole : public CScallion

{

public:

CBlackHole();

CBlackHole(BitmapPath, CPoint, CPoint, int = 4); //給兩CPoint + int 為參數時，第一個為目前座標，第二個為滑鼠座標，最後為重力參數

CBlackHole(BitmapPath, CPoint, int, int); //若不給重力參數，第一個CPoint不變，但後面變為「x初速, y初速」

CRect GetCenterRect() {

return centerRect;

};

void OnMove();

void SetRole(CRole\* \_r) {

role = \_r;

};

void Initialize();

private:

CTimer selfBang; //自爆timer

CRect centerRect;

CRect centerRectSize;

CRole\* role;

int csize = 100;

};

#pragma endregion

#pragma region - CRay -

class CRay

{

public:

CRay();

CRay(BitmapPath, CPoint, double = 0.1); //動畫路徑, 起始位置, 每個影格間隔時間

~CRay();

void OnMove();

bool GetValid() {

return animation.GetValid();

};

void SetValid(bool f) {

animation.SetValid(f);

};

CAnimate\* GetAnimate() {

return &animation;

};

void SetXY(int \_x, int \_y) {

x = \_x;

y = \_y;

animation.SetTopLeft(x, y);

}

bool GetAttackValid() {

return AttackValid;

};

private:

CAnimate animation;

int x, y;

bool AttackValid; //攻擊有效

};

#pragma endregion

}

1. CBall.cpp

#pragma once

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CBall.h"

#include "CEraser.h"

#include "CManager.h"

using namespace myLibrary;

namespace game\_framework {

#pragma region - ball -

CBall::CBall()

{

is\_alive = true;

}

bool CBall::IsAlive()

{

return is\_alive;

}

void CBall::SetIsAlive(bool alive)

{

is\_alive = alive;

}

void CBall::SetXY(int nx, int ny)

{

x = nx;

y = ny;

if (!animation.IsNull())

animation.SetTopLeft(x, y);

}

void CBall::SetCurrentXY(double cx, double cy)

{

currentX = cx;

currentY = cy;

int dx = CCamera::Instance()->GetX();

int nx = (int)(currentX) - dx;

int ny = (int)(currentY);

SetXY(nx, ny);

}

void CBall::LoadBitmap(string ziliaojia, string name, int number)

{

animation.LoadBitmap(ziliaojia, name, number);

}

void CBall::LoadBitmap(BitmapPath loadpath)

{

animation.LoadBitmap(loadpath.ziliaojia, loadpath.name, loadpath.number, loadpath.color);

}

CAnimate\* CBall::GetAnimate()

{

return &animation;

}

#pragma endregion

#pragma region - CScallion -

CScallion::CScallion()

{

const int INIT\_X = 0, INIT\_Y = 0;

const int GRAVITY = 4;

currentX = INIT\_X;

currentY = INIT\_Y;

x = INIT\_X;

y = INIT\_Y;

gravity = GRAVITY;

layer.SetLayer(BULLET\_LAYER);

animation.SetTopLeft(INIT\_X, INIT\_Y);

is\_alive = true;

LoadBitmap("Role", "scallion", 4);

CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());

}

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

{

gravity = \_gravity;

LoadBitmap(loadpath);

Initialize(point);

SetInitVelocity(point.x, point.y, finalPoint.x, finalPoint.y);

}

CScallion::CScallion(BitmapPath loadpath, CPoint point, int initV\_x, int initV\_y)

{

gravity = 0;

LoadBitmap(loadpath);

Initialize(point);

velocity\_x = initV\_x;

velocity\_y = initV\_y;

}

void CScallion::Clear()

{

this->~CScallion();

}

void CScallion::Initialize(CPoint initPos)

{

const int INIT\_X = initPos.x;

const int INIT\_Y = initPos.y;

currentX = x = INIT\_X;

currentY = y = INIT\_Y;

is\_alive = true;

animation.SetTopLeft(x, y);

animation.ResetDelayTime(0.1);

layer.SetLayer(BULLET\_LAYER);

CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());

}

void CScallion::OnMove()

{

if (!IsAlive())

return;

if (x > SIZE\_X || x < 0 || y > SIZE\_Y || y < -300) //超出螢幕

{

SetIsAlive(false);

animation.SetValid(false);

return;

}

#pragma region - 重力計算 -

if (velocity\_y > 0)

{

velocity\_y -= gravity;

y -= velocity\_y;

}

else

{

velocity\_y -= gravity;

y -= velocity\_y;

}

#pragma endregion

animation.OnMove();

SetXY(x + velocity\_x, y);

}

void CScallion::OnShow()

{

if (!IsAlive())

return;

animation.SetTopLeft(x, y);

animation.OnShow();

}

void CScallion::SetInitVelocity(int vx, int vy)

{

velocity\_x = vx;

velocity\_y = vy;

}

void CScallion::SetInitVelocity(int b\_x, int b\_y, int f\_x, int f\_y, int fix\_velocity)

{

int dx = (f\_x - b\_x) / fix\_velocity / 2;

int dy = -(f\_y - b\_y) / fix\_velocity / 2;

velocity\_x = dx;

velocity\_y = dy;

}

bool CScallion::IsCollision(CMonster passerby)

{

return false;

return IsPointInRect(CPoint(x + animation.Width(), y), passerby.animation.GetRect());

}

bool CScallion::IsCollision(CMonster\* passerby)

{

return IsRectCollision(animation.GetRect(), passerby->animation.GetRect());

}

bool CScallion::IsCollision(CBoss\* boss)

{

return IsRectCollision(animation.GetRect(), boss->GetAnimate()->GetRect());

}

bool CScallion::IsCollision(CRect rect)

{

return IsRectCollision(animation.GetRect(), rect);

}

#pragma endregion

#pragma region - CBlackHole -

CBlackHole::CBlackHole() : CScallion()

{

}

CBlackHole::CBlackHole(BitmapPath loadpath, CPoint point, CPoint finalPoint, int \_gravity) : CScallion(loadpath, point, finalPoint, \_gravity)

{

SetInitVelocity(point.x, point.y, finalPoint.x, finalPoint.y, 25);

Initialize();

}

CBlackHole::CBlackHole(BitmapPath loadpath, CPoint point, int initV\_x, int initV\_y) : CScallion(loadpath, point, initV\_x, initV\_y)

{

Initialize();

}

void CBlackHole::Initialize()

{

centerRectSize = CRect(0, 0, csize, csize);

role = NULL;

selfBang = CTimer(1.8);

SetIsAlive(true);

}

void CBlackHole::OnMove()

{

selfBang.CountDown();

if (selfBang.IsTimeOut())

{

SetIsAlive(false);

role = NULL;

return;

}

if (y + animation.Height() < SIZE\_Y)

{

if (!IsAlive())

return;

if (x > SIZE\_X || x + animation.Width() + 87 < 0 || y > SIZE\_Y || y < -300) //超出螢幕

{

SetIsAlive(false);

animation.SetValid(false);

return;

}

#pragma region - 重力計算 -

if (velocity\_y > 0)

{

velocity\_y -= gravity;

currentY -= velocity\_y;

}

else

{

velocity\_y -= gravity;

currentY -= velocity\_y;

}

#pragma endregion

animation.OnMove();

SetCurrentXY(currentX + velocity\_x, currentY);

}

animation.OnMove();

int centerRectWidth = centerRectSize.right;

int centerRectHeight = centerRectSize.bottom;

int centerPointX = x + animation.Width() / 2;

int centerPointY = y + animation.Height() / 2;

centerRect = CRect(centerPointX - centerRectWidth / 2, //left

centerPointY - centerRectHeight / 2, //top

centerPointX + centerRectWidth / 2, //right

centerPointY + centerRectHeight / 2); //bottom

if (role != NULL)

{

int ddrsx = (role->GetX3() - centerPointX) / 8;

role->SetXY(role->GetX1() - ddrsx, role->GetY1());

}

}

#pragma endregion

#pragma region - CRay -

CRay::CRay()

{

}

CRay::CRay(BitmapPath loadPath, CPoint Fpoint, double resetTime)

{

animation.LoadBitmap(loadPath.ziliaojia, loadPath.name, loadPath.number, loadPath.color);

SetValid(false);

SetXY(Fpoint.x, Fpoint.y);

AttackValid = false;

animation.ResetDelayTime(resetTime);

CLayerManager::Instance()->AddObject(&animation, 6);

}

CRay::~CRay()

{

}

void CRay::OnMove()

{

animation.SetValid(true);

if (animation.GetIndex() < animation.GetIndexSize() - 1)

{

animation.OnMove();

}

else

{

AttackValid = true;

}

}

#pragma endregion

}

1. CBoss.h

#pragma once

#include "CLibrary.h"

#include "Refactor.h"

#include <vector>

using namespace myLibrary;

namespace game\_framework

{

#pragma region - CBoss -

class CScallion;

class CRole;

class CBlackHole;

class CBoss

{

public:

CBoss();

CBoss(int, int, int, string, BitmapPath); //x, y, hp, (路徑, 顏色)

virtual ~CBoss();

CPoint GetLeftTopPoint() {

return CPoint(currentX, currentY);

};

CPoint GetCenterPoint() {

return CPoint(currentX + width / 2, currentY + height / 2);

};

CPoint GetRightBottomPoint() {

return CPoint(currentX + width, currentY + height);

};

void LoadBitmap();

void virtual Initialize();

void InitializeDirAnimate(string, double = 0.1);

void SetFaceTo(CPoint);

void SetXY(int, int);

void SetCurrentXY(int, int);

void SetHp(int);

void SetIsAlive(bool);

bool GetAlive() {

return IsAlive;

};

string GetID() {

return id;

};

int GetHp() {

return hp;

};

int GetInitHp() {

return initHp;

};

void OnMove();

void MoveWithMap(string);

void CollisionScallion(CRole\*);

virtual void OnCycle(CRole\*) {};

virtual void Attack(CRole\*) {};

virtual void Clear() {};

virtual bool InEndProcess() {

return false;

};

virtual void EndProcess() {};

bool IsEnd() {

return isEnd;

};

virtual vector<CScallion\*>\* GetBullet() {

return nullptr;

};

bool IsDead() {

return !IsAlive;

};

CTimer\* GetAliveTimer() {

return &AliveTime;

};

CLayer layer;

CAnimate\* GetAnimate();

string GetBossId() {

return id;

};

protected:

#pragma region Init boss Information

int initx, inity, initHp;

BitmapPath loadPath;

COLORREF transparentColor;

#pragma endregion

CAnimate animation;

CAnimate leftAnimate;

CAnimate rightAnimate;

string faceTo;

int hp;

int currentX, currentY;

int x, y;

int width, height;

bool IsAlive = true;

string id;

CTimer AliveTime;

CTimer ChangeFaceTimer;

const double PI = 4 \* atan(1.0);

bool isEnd;

};

#pragma endregion

#pragma region - CXingting -

class CXingting : public CBoss

{

public:

CXingting();

CXingting(int, int, int, string, BitmapPath);

~CXingting();

void Initialize();

void OnCycle(CRole\*);

void Attack(CRole\*);

void OnMove();

void Clear();

void ClearBullet();

bool IsDead();

private:

vector<CScallion\*> level4;

vector<CBlackHole\*> blackhole;

CBlackHole\* targetBlackhole;

void Level4Collision(CRole\*);

CTimer shootLevel4\_cd;

CTimer shoot\_atk2\_cd;

CTimer shoot\_atk3\_cd;

CTimer moveToGoal;

CTimer mode\_Attack2\_timer;

CTimer mode\_Attack4\_CreateBlackHole;

int goal\_x, goal\_y;

int angle\_atk2;

double mode4\_AttackTime;

int mode4\_RoleMoveDir; // 1 = 左 2 = 右 3 = 不能動

void Attack1();

void Attack2();

void Attack3();

void Attack4(CRole\*);

bool moveToGoalPoint;

bool mode\_Attack1;

bool mode\_Attack2;

bool mode\_Attack3;

bool mode\_Attack4;

int atkCounter;

};

#pragma endregion

#pragma region - CFaicaiSeed -

class CRay;

class CFacaiSeed : public CBoss

{

public:

CFacaiSeed();

CFacaiSeed(int, int, int, string, BitmapPath);

~CFacaiSeed();

void Initialize();

void OnCycle(CRole\*);

void Attack(CRole\*);

void OnMove();

void Clear();

bool InEndProcess() {

return fly;

};

void EndProcess();

private:

void Attack1(CRole\*);

void Attack2();

void Attack3(); //Ray attack

void ClearBullet();

void Collision(CRole\*);

CPoint GetCreateCoinPoint(); //射出金幣的起始點

CTimer movingTime;

CTimer shootCoinTimer;

CTimer attackRoleTimer; //role 硬值時間

CTimer subRoleHp\_NoEQ;

bool fly;

int flyY = 1;

vector<CScallion\*> coinVector;

int coinAngle[4] = { 45, 75, 105, 135 };

#pragma region - ray -

CTimer rayStartTime;

CTimer rayStayTime;

CRay\* ray = NULL;

#pragma endregion

};

#pragma endregion

}

1. CBoss.cpp

#pragma once

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CEraser.h"

#include "CBall.h"

#include <sstream>

#include "CManager.h"

#include "CLibrary.h"

#include "CBoss.h"

#include <vector>

using namespace myLibrary;

namespace game\_framework

{

#pragma region - CBoss -

CBoss::CBoss()

{

initHp = 0;

initx = 0;

inity = 0;

}

CBoss::CBoss(int \_x, int \_y, int \_hp, string \_id, BitmapPath \_loadPath)

{

initx = \_x;

inity = \_y;

initHp = \_hp;

loadPath = \_loadPath;

id = \_id;

}

CBoss::~CBoss()

{

}

void CBoss::LoadBitmap()

{

#pragma region - Load animateion and left, right animation -

BitmapPath leftPath = loadPath;

BitmapPath rightPath = loadPath;

leftPath.ziliaojia += "\\L";

rightPath.ziliaojia += "\\R";

animation.LoadBitmap(leftPath.ziliaojia, leftPath.name, leftPath.number, leftPath.color);

rightAnimate.LoadBitmap(rightPath.ziliaojia, rightPath.name, rightPath.number, rightPath.color);

#pragma endregion

animation.SetTopLeft(initx, inity);

height = animation.Height();

width = animation.Width();

leftAnimate = animation;

}

void CBoss::Initialize()

{

#pragma region - 僅一次的載入圖片 -

if (animation.IsNull())

{

LoadBitmap();

}

#pragma endregion

inity = SIZE\_Y - animation.Height();

SetCurrentXY(initx, inity);

SetXY(initx, inity);

hp = initHp;

layer.SetLayer(BOSS\_LAYER);

IsAlive = true;

isEnd = false;

animation.SetValid(false);

CLayerManager::Instance()->AddObject(&animation, layer.GetLayer());

ChangeFaceTimer = CTimer(1.0);

AliveTime = CTimer(99.0);

}

void CBoss::InitializeDirAnimate(string dir, double resetTime)

{

animation.ResetDelayTime(resetTime);

animation.SetTopLeft(initx, inity);

leftAnimate.ResetDelayTime(resetTime);

rightAnimate.ResetDelayTime(resetTime);

leftAnimate.CopyAnimateInformation(&animation);

rightAnimate.CopyAnimateInformation(&animation);

faceTo = dir;

if (faceTo == "left")

animation = leftAnimate;

else if (faceTo == "right")

animation = rightAnimate;

}

void CBoss::SetFaceTo(CPoint point)

{

if (point.x >= GetCenterPoint().x) //right

{

rightAnimate.CopyAnimateInformation(&animation);

animation = rightAnimate;

faceTo = "right";

}

else

{

leftAnimate.CopyAnimateInformation(&animation);

animation = leftAnimate;

faceTo = "left";

}

}

void CBoss::SetXY(int \_x, int \_y)

{

x = \_x;

y = \_y;

animation.SetTopLeft(x, y);

}

void CBoss::SetCurrentXY(int \_x, int \_y)

{

currentX = \_x;

currentY = \_y;

}

void CBoss::SetHp(int \_hp)

{

hp = \_hp;

}

void CBoss::SetIsAlive(bool flag)

{

IsAlive = flag;

}

void CBoss::MoveWithMap(string dir)

{

if (dir == "left")

x += MOVE\_DISTANCE;

if (dir == "right")

x -= MOVE\_DISTANCE;

SetXY(x, y);

}

void CBoss::CollisionScallion(CRole\* role)

{

#pragma region - scallion collision this -

vector<CScallion\*>\* scallion = role->GetScallion();

for (vector<CScallion\*>::iterator scallionk = scallion->begin(); scallionk != scallion->end();)

{

if ((\*scallionk)->IsAlive() && (\*scallionk)->IsCollision(this))

{

if (GetID() != "Xingting")

CAudio::Instance()->Play("hit");

(\*scallionk)->SetIsAlive(false);

hp -= (\*scallionk)->GetAtk();

}

if (!(\*scallionk)->IsAlive())

{

delete \*scallionk;

\*scallionk = NULL;

scallionk = scallion->erase(scallionk);

}

else

{

scallionk++;

}

}

#pragma endregion

}

void CBoss::OnMove()

{

int dx = CCamera::Instance()->GetX();

SetXY(currentX - dx, currentY);

animation.OnMove();

}

CAnimate\* CBoss::GetAnimate()

{

return &animation;

}

#pragma endregion

#pragma region - CXingting -

CXingting::CXingting()

{

}

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

{

}

CXingting::~CXingting()

{

ClearBullet();

}

void CXingting::Initialize()

{

CBoss::Initialize();

ClearBullet();

targetBlackhole = NULL;

goal\_x = 320 - width / 2;

goal\_y = 0;

shootLevel4\_cd = CTimer(0.75);

shoot\_atk2\_cd = CTimer(0.033);

shoot\_atk3\_cd = CTimer(2.5);

moveToGoal = CTimer(0.04);

mode\_Attack2\_timer = CTimer(8.0);

mode\_Attack4\_CreateBlackHole = CTimer(0.5);

mode4\_AttackTime = 1.8; //mode4時 每1.8發射一顆黑洞

angle\_atk2 = 0;

mode4\_RoleMoveDir = 0;

moveToGoalPoint = true;

mode\_Attack1 = true;

mode\_Attack2 = false;

mode\_Attack3 = false;

mode\_Attack4 = false;

atkCounter = 0;

AliveTime = CTimer(40.0);

}

void CXingting::OnCycle(CRole\* role)

{

if (AliveTime.IsTimeOut()) //時間到 杏庭自爆

{

if (IsAlive)

{

CDialogManager::Instance()->Start(DIALOG\_DATA\_VSXingting3);

IsAlive = false;

isEnd = true;

}

}

if (!IsDead())

{

OnMove();

Attack(role);

}

}

void CXingting::OnMove()

{

if (moveToGoalPoint)

{

int move\_dx = (currentX > goal\_x) ? (initx - goal\_x) / 50 : 0;

int move\_dy = (currentY > goal\_y) ? (inity - goal\_y) / 50 : 0;

moveToGoal.CountDown();

if (moveToGoal.IsTimeOut())

{

SetCurrentXY(currentX - move\_dx, currentY - move\_dy);

moveToGoal.ResetTime();

}

if (currentX <= goal\_x && currentY <= goal\_y)

{

moveToGoalPoint = false;

mode\_Attack1 = false;

mode\_Attack2 = true;

mode\_Attack3 = true;

atkCounter = 0;

CDialogManager::Instance()->Start(DIALOG\_DATA\_VSXingting2);

}

}

int dx = CCamera::Instance()->GetX();

SetXY(currentX - dx, currentY);

animation.OnMove();

}

void CXingting::Attack(CRole\* role)

{

if (mode\_Attack1)

{

Attack1();

}

if (mode\_Attack2)

{

Attack2();

mode\_Attack2\_timer.CountDown();

if (mode\_Attack2\_timer.IsTimeOut())

{

mode\_Attack2 = false;

mode\_Attack3 = false;

mode\_Attack4 = true;

}

}

if (mode\_Attack3)

{

Attack3();

}

if (mode\_Attack4)

{

Attack4(role);

}

Level4Collision(role);

}

void CXingting::Attack1()

{

shootLevel4\_cd.CountDown();

if (shootLevel4\_cd.IsTimeOut())

{

for (int i = 0; i < 12; i++)

{

double speed = 15.0 / 2;

int angle = (-150 + atkCounter \* 15) + i \* 30;

double mx = sin(angle \* (PI / 180.0)) \* speed;

double my = cos(angle \* (PI / 180.0)) \* speed;

int drs = 3; //給一個倍數讓books不從中心點出現

CPoint center = GetCenterPoint() + CPoint((int)mx \* drs, -(int)my \* drs);

CScallion\* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4), center, CPoint(0, 0), 0); //先創建一個蔥的物件

newlevel4->SetInitVelocity((int)mx, (int)my);

level4.push\_back(newlevel4); //將蔥放進vector

shootLevel4\_cd.ResetTime();

}

}

}

void CXingting::Attack2()

{

shoot\_atk2\_cd.CountDown();

if (shoot\_atk2\_cd.IsTimeOut())

{

#pragma region - bullet parameter -

double speed = 15.0 / 2;

int angle = angle\_atk2;

double mx = sin(angle \* (PI / 180.0)) \* speed;

double my = cos(angle \* (PI / 180.0)) \* speed;

int drs = 3; //用意在給一個倍數讓books不從中心點出現

CPoint center = GetCenterPoint() + CPoint((int)mx \* drs, -(int)my \* drs);

#pragma endregion

CAudio::Instance()->Play("blackhole");

CScallion\* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4), center, CPoint(0, 0), 0); //先創建一個蔥的物件

newlevel4->SetInitVelocity((int)mx, (int)my);

level4.push\_back(newlevel4); //將蔥放進vector

shoot\_atk2\_cd.ResetTime();

angle\_atk2 += 31;

}

}

void CXingting::Attack3()

{

shoot\_atk3\_cd.CountDown();

if (shoot\_atk3\_cd.IsTimeOut())

{

for (int i = 0; i < 18; i++)

{

#pragma region - bullet parameter -

double speed = 10.0 / 2;

double angle = i \* (360 / 18);

double mx = sin(angle \* (PI / 180.0)) \* speed;

double my = cos(angle \* (PI / 180.0)) \* speed;

int drs = 3; //用意在給一個倍數讓books不從中心點出現

CPoint center = GetCenterPoint() + CPoint((int)my \* drs, -(int)mx \* drs);

#pragma endregion

CScallion\* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4), center, CPoint(0, 0), 0); //先創建一個蔥的物件

newlevel4->SetInitVelocity((int)my, (int)mx);

level4.push\_back(newlevel4); //將蔥放進vector

shoot\_atk3\_cd.ResetTime();

}

}

}

void CXingting::Attack4(CRole\* role)

{

mode\_Attack4\_CreateBlackHole.CountDown();

if (mode\_Attack4\_CreateBlackHole.IsTimeOut())

{

CAudio::Instance()->Play("blackhole");

CPoint center = GetCenterPoint();

CBlackHole\* newBh = new CBlackHole(BitmapPath("RES\\Object\\blackhole", "blackhole", 1, RGB(0, 0, 0)), center + CPoint(-75, -68), CPoint(role->GetX1(), role->GetY1()), 0); //先創建一個蔥的物件

blackhole.push\_back(newBh);

mode\_Attack4\_CreateBlackHole.ResetTime(mode4\_AttackTime); //reset

}

}

void CXingting::Clear()

{

ClearBullet();

animation.SetValid(false);

}

void CXingting::ClearBullet()

{

for (vector<CBlackHole\*>::iterator bkiter = blackhole.begin(); bkiter != blackhole.end(); bkiter++)

{

delete \*bkiter;

\*bkiter = NULL;

}

for (vector<CScallion\*>::iterator level4iter = level4.begin(); level4iter != level4.end(); level4iter++)

{

delete \*level4iter;

\*level4iter = NULL;

}

level4.clear();

blackhole.clear();

}

bool CXingting::IsDead()

{

return AliveTime.IsTimeOut();

}

void CXingting::Level4Collision(CRole\* role)

{

for (vector<CBlackHole\*>::iterator bkiter = blackhole.begin(); bkiter != blackhole.end();)

{

(\*bkiter)->OnMove();

if ((\*bkiter)->IsAlive() && !role->IsCatched() && role->IsCollisionBlackHole((\*bkiter)))

{

targetBlackhole = \*bkiter;

role->SetCatched(true);

(\*bkiter)->SetRole(role);

}

if (!(\*bkiter)->IsAlive())

{

#pragma region - Create Bullet -

int bulletNumber = 12;

for (int i = 0; i < bulletNumber; i++)

{

#pragma region - bullet parameter -

double speed = 15.0 / 2;

double angle = i \* (360 / bulletNumber);

double mx = sin(angle \* (PI / 180.0)) \* speed;

double my = cos(angle \* (PI / 180.0)) \* speed;

int drs = 3; //亂取的名字 用意在給一個倍數讓books不從中心點出現

CPoint center = GetCenterPoint() + CPoint((int)my \* drs, -(int)mx \* drs);

#pragma endregion

CScallion\* newlevel4 = new CScallion(BitmapPath("RES\\Object\\books", "book", 4), (\*bkiter)->GetCenterPoint(), CPoint(0, 0), 0); //先創建一個蔥的物件

newlevel4->SetInitVelocity((int)mx, (int)my);

level4.push\_back(newlevel4); //將蔥放進vector

}

#pragma endregion

targetBlackhole = NULL;

role->SetCatched(false);

}

if (!(\*bkiter)->IsAlive())

{

delete \*bkiter;

\*bkiter = NULL;

bkiter = blackhole.erase(bkiter);

}

else

{

bkiter++;

}

}

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

{

(\*level4iter)->OnMove();

if ((\*level4iter)->IsAlive() && role->IsCollisionLevel4(\*level4iter))

{

(\*level4iter)->SetIsAlive(false);

CAudio::Instance()->Play("role\_hitted");

role->SubHp();

}

if (!(\*level4iter)->IsAlive())

{

delete \*level4iter;

\*level4iter = NULL;

level4iter = level4.erase(level4iter);

}

else

{

level4iter++;

}

}

}

#pragma endregion

#pragma region - CFaicaiSeed -

CFacaiSeed::CFacaiSeed()

{

ray = NULL;

}

CFacaiSeed::CFacaiSeed(int \_x, int \_y, int \_hp, string \_id, BitmapPath \_loadPath) : CBoss(\_x, \_y, \_hp, \_id, \_loadPath)

{

ray = NULL;

}

CFacaiSeed::~CFacaiSeed()

{

Clear();

}

void CFacaiSeed::Initialize()

{

CBoss::Initialize();

InitializeDirAnimate("right", 0.05);

#pragma region - Init ray -

ray = NULL;

rayStartTime.ResetTime(2.5); //預設k秒後發射

rayStayTime.ResetTime(2.0); //預設持續兩秒

#pragma endregion

Clear();

fly = false;

flyY = 1;

movingTime = CTimer(0.2);

shootCoinTimer = CTimer(0.2); //每0.2s發射1波金幣

attackRoleTimer = CTimer(0.4); //硬值時間0.4s

subRoleHp\_NoEQ = CTimer(0.6); //每0.6s掉一次血

AliveTime = CTimer(99.0);

}

void CFacaiSeed::OnCycle(CRole\* role)

{

#pragma region - boss dead -

if (hp <= 0) //boss dead

{

CDialogManager::Instance()->Start("roleWinFacaiSeed");

SetIsAlive(false);

}

#pragma endregion

#pragma region - Boss Alive -

if (!IsDead())

{

attackRoleTimer.CountDown();

if (ray == NULL) //射線中不轉換

{

ChangeFaceTimer.CountDown();

if (ChangeFaceTimer.IsTimeOut())

{

SetFaceTo(CPoint(role->GetX3(), role->GetY3()));

ChangeFaceTimer.ResetTime();

}

}

Attack(role);

OnMove();

Collision(role);

}

#pragma endregion

#pragma region - role dead -

if (role->IsDead() || AliveTime.IsTimeOut())

{

CDialogManager::Instance()->Start("roleLoseFacaiSeed");

ClearBullet();

fly = true;

}

#pragma endregion

}

void CFacaiSeed::Attack(CRole\* role)

{

Attack1(role);

bool isTalking = CDialogManager::Instance()->GetDialogState();

if (ray == NULL && !isTalking)

{

CAudio::Instance()->Play("ray");

Attack2();

}

Attack3();

}

void CFacaiSeed::OnMove()

{

#pragma region - rays animation -

if (ray != NULL)

{

ray->OnMove();

}

#pragma endregion

#pragma region - dont have ray - move -

else

{

int dx = 1;

if (faceTo == "right")

{

SetCurrentXY(currentX + dx, currentY);

}

else if (faceTo == "left")

{

SetCurrentXY(currentX - dx, currentY);

}

}

#pragma endregion

CBoss::OnMove();

}

void CFacaiSeed::Clear()

{

ClearBullet();

animation.SetValid(false);

}

void CFacaiSeed::EndProcess()

{

#pragma region - role dead and this fly -

if (fly)

{

CAudio::Instance()->Play("faacai\_fly");

SetCurrentXY(currentX, currentY - flyY);

flyY++;

CBoss::OnMove();

if (GetRightBottomPoint().y <= -10)

{

fly = false;

isEnd = true;

Clear();

}

}

#pragma endregion

}

void CFacaiSeed::Attack1(CRole\* role)

{

if (role->IsZZ())

{

subRoleHp\_NoEQ.CountDown();

if (subRoleHp\_NoEQ.IsTimeOut())

{

CAudio::Instance()->Play("role\_hitted");

role->SubHp();

subRoleHp\_NoEQ.ResetTime();

}

}

}

void CFacaiSeed::Attack2()

{

shootCoinTimer.CountDown();

if (shootCoinTimer.IsTimeOut())

{

CAudio::Instance()->Play("door");

for (int i = 0; i < 4; i++)

{

double speed = 27.0;

int angle = coinAngle[i];

double mx = sin(angle \* (PI / 180.0)) \* speed;

double my = cos(angle \* (PI / 180.0)) \* speed;

double drs = 2.5;

CPoint center = GetCreateCoinPoint() + CPoint((int)(my \* drs), -(int)(mx \* drs));

CScallion\* newcoin = new CScallion(BitmapPath("RES\\Object\\coin", "coin", 4, RGB(255, 255, 255)), center, CPoint(0, 0), 1); //先創建一個蔥的物件

newcoin->SetInitVelocity((int)my, (int)mx);

coinVector.push\_back(newcoin);

shootCoinTimer.ResetTime();

}

}

}

void CFacaiSeed::Attack3()

{

rayStartTime.CountDown();

if (rayStartTime.IsTimeOut())

{

if (ray == NULL)

{

if (faceTo == "right")

{

ray = new CRay(BitmapPath("RES\\Object\\Ray", "ray", 5, RGB(214, 214, 214)), CPoint(GetRightBottomPoint().x, GetLeftTopPoint().y + 40));

}

else

{

ray = new CRay(BitmapPath("RES\\Object\\Ray", "ray", 5, RGB(214, 214, 214)), CPoint(GetLeftTopPoint().x - 531, GetLeftTopPoint().y + 40));

}

}

rayStayTime.CountDown(); //開始計算持續時間

if (rayStayTime.IsTimeOut()) //delete ray

{

if (ray != NULL)

{

delete ray;

ray = NULL;

}

rayStayTime.ResetTime();

rayStartTime.ResetTime();

}

}

}

void CFacaiSeed::ClearBullet()

{

#pragma region - clear coin -

for (vector<CScallion\*>::iterator coiniter = coinVector.begin(); coiniter != coinVector.end(); coiniter++)

{

delete \*coiniter;

\*coiniter = NULL;

}

coinVector.clear();

#pragma endregion

#pragma region - clear ray -

if (ray != NULL)

{

delete ray;

ray = NULL;

}

#pragma endregion

}

void CFacaiSeed::Collision(CRole\* role)

{

#pragma region - coin collision role -

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

{

(\*cointiter)->OnMove();

if ((\*cointiter)->IsAlive() && role->IsCollisionLevel4(\*cointiter))

{

CAudio::Instance()->Play("role\_hitted");

(\*cointiter)->SetIsAlive(false);

role->SubHp();

}

if (!(\*cointiter)->IsAlive())

{

delete \*cointiter;

\*cointiter = NULL;

cointiter = coinVector.erase(cointiter);

}

else

{

cointiter++;

}

}

#pragma endregion

#pragma region - ray collision role -

if (ray != NULL)

{

if (ray->GetAttackValid() && role->IsCollisionRay(ray))

{

if (attackRoleTimer.IsTimeOut())

{

CAudio::Instance()->Play("role\_hitted");

role->SubHp();

attackRoleTimer.ResetTime();

}

}

}

#pragma endregion

#pragma region - this collision role -

if (IsRectCollision(role->GetAction()->GetRect(), animation.GetRect()))

{

if (attackRoleTimer.IsTimeOut())

{

role->SubEq();

attackRoleTimer.ResetTime();

}

}

#pragma endregion

#pragma region - this collsion scallion -

CollisionScallion(role);

#pragma endregion

}

CPoint CFacaiSeed::GetCreateCoinPoint()

{

CPoint center = GetCenterPoint();

int dx = 45;

if (faceTo == "left")

{

center += CPoint(dx, 0);

}

else

{

center -= CPoint(dx, 0);

}

return center;

}

#pragma endregion

}

1. CBlockMap.h

#pragma once

#include <vector>

#include "CEraser.h"

using namespace std;

namespace game\_framework

{

#pragma region - CSimpleMapObj -

class CSimpleMapObj

{

public:

CSimpleMapObj() {

path = "";

x = y = 0;

};

CSimpleMapObj(string \_path, int \_x, int \_y) {

path = \_path;

x = \_x;

y = \_y;

};

~CSimpleMapObj() {};

void SetXY(int \_x, int \_y, int camera = 0) {

x = \_x;

y = \_y;

bmp.SetTopLeft(x - camera, y);

};

void LoadImg() {

char\* address = ConvertCharPointToString(path);

bmp.LoadBitmap(address, RGB(214, 214, 214));

delete address;

};

CMovingBitmap bmp;

string path;

int x, y; //實際位置上的x y

};

#pragma endregion

#pragma region - CBlock -

class CBlock : public CSimpleMapObj

{

public:

CBlock() : CSimpleMapObj() {};

CBlock(string \_path, int \_x, int \_y) : CSimpleMapObj(\_path, \_x, \_y) {};

~CBlock() {};

};

#pragma endregion

#pragma region - CDoor -

class CBlockMap;

class CDoor : public CSimpleMapObj

{

public:

CDoor() : CSimpleMapObj() {};

CDoor(string \_path, int \_x, int \_y, string \_type, int \_deliveMapIndex) : CSimpleMapObj(\_path, \_x, \_y) {

type = \_type;

deliverMapIndex = \_deliveMapIndex;

};

~CDoor() {};

int GetSwitchMapIndex() {

return deliverMapIndex;

}

string GetType() {

return type;

};

private:

string type; //上 or 下

int deliverMapIndex = -1; //要傳送的地圖

};

#pragma endregion

#pragma region - CBlockMap -

class CBlockMap

{

public:

CBlockMap(); //default constructor

CBlockMap(int);

~CBlockMap();

void LoadImg();

void LoadInformation(int);

void LoadInformation(string);

void CreateInformation();

void CreateInformation(string);

void Initialize(); //有用到的東西都歸0

bool isLoad = false;

int nowMap; //目前地圖的編號

int leftMap, rightMap, upMap, downMap; //目前地圖，其上下左右地圖的編號 ( < 0代表不存在)

int loadMap; //載入的地圖資訊

CMovingBitmap backgroundBitmap; //背景圖片 (用LoadBitmap讀取loadMap)

int passerbyMaxSize;

vector<int> passerbyID = {0, 1};

string ziliaojia, name;

int number;

string loadPath;

vector<CBlock> block;

vector<CDoor> door;

private:

void LoadMap(string); //讀取string的txt

void WriteMap(string); //寫下string的txt

};

#pragma endregion

}

1. CBlockMap.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <vector>

#include "audio.h"

#include "gamelib.h"

#include <fstream>

#include <sstream>

#include "CBlockMap.h"

#include "CManager.h"

#include "CEraser.h"

using namespace std;

namespace game\_framework

{

#pragma region - CBlockMap -

CBlockMap::CBlockMap()

{

nowMap = 0;

upMap = -1;

downMap = -1;

leftMap = -1;

rightMap = -1;

loadMap = IDB\_MAP0;

passerbyMaxSize = 0;

loadPath = "RES\\Map\\IDB\_Map\_0.bmp";

}

CBlockMap::CBlockMap(int \_nowMap)

{

nowMap = \_nowMap;

LoadInformation(nowMap);

}

CBlockMap::~CBlockMap()

{

}

void CBlockMap::LoadImg()

{

char\* address = ConvertCharPointToString(loadPath);

backgroundBitmap.LoadBitmap(address);

delete address;

for (vector<CBlock>::iterator blockiter = block.begin(); blockiter != block.end(); blockiter++)

{

blockiter->LoadImg();

}

for (vector<CDoor>::iterator dooriter = door.begin(); dooriter != door.end(); dooriter++)

{

dooriter->LoadImg();

}

isLoad = true;

}

void CBlockMap::LoadInformation(int mapIndex)

{

string mapFilePath = "RES\\Map\\Information\\map" + std::to\_string(mapIndex) + ".txt";

LoadMap(mapFilePath);

}

void CBlockMap::LoadInformation(string mapFileName)

{

string mapFilePath = "RES\\Map\\Information\\" + mapFileName; //含附檔名

LoadMap(mapFilePath);

}

void CBlockMap::LoadMap(string mapFilePath)

{

fstream mapData;

mapData.open(mapFilePath);

string lineData;

while (getline(mapData, lineData))

{

vector<string> tempString = SplitString(lineData);

#pragma region - load map information -

if (tempString[0] == "background")

{

loadPath = tempString[1];

}

else if (tempString[0] == "block")

{

block.push\_back(CBlock(tempString[1], ConvertStringToInt(tempString[2]), ConvertStringToInt(tempString[3])));

}

else if (tempString[0] == "upDoor" || tempString[0] == "downDoor")

{

door.push\_back(CDoor(tempString[1], ConvertStringToInt(tempString[2]), ConvertStringToInt(tempString[3]), tempString[0], ConvertStringToInt(tempString[4])));

}

else if (tempString[0] == "up")

{

upMap = ConvertStringToInt(tempString[1]);

}

else if (tempString[0] == "down")

{

downMap = ConvertStringToInt(tempString[1]);

}

else if (tempString[0] == "left")

{

leftMap = ConvertStringToInt(tempString[1]);

}

else if (tempString[0] == "right")

{

rightMap = ConvertStringToInt(tempString[1]);

}

else if (tempString[0] == "passerbyNumber")

{

passerbyMaxSize = ConvertStringToInt(tempString[1]);

}

else if (tempString[0] == "passerbyID")

{

passerbyID.clear();

for (unsigned int i = 1; i < tempString.size(); i++)

{

passerbyID.push\_back(ConvertStringToInt(tempString[i]));

}

}

#pragma endregion

}

mapData.close();

}

void CBlockMap::CreateInformation()

{

string fileName = "RES\\Map\\Information\\map" + std::to\_string(nowMap) + ".txt";

WriteMap(fileName);

}

void CBlockMap::CreateInformation(string mapName) //含附檔名

{

string fileName = "RES\\Map\\Information\\" + mapName;

WriteMap(fileName);

}

void CBlockMap::Initialize()

{

isLoad = false;

nowMap = 0;

leftMap = rightMap = upMap = downMap = -1;

block.clear();

door.clear();

passerbyMaxSize = 0;

loadPath = "";

}

void CBlockMap::WriteMap(string fileName)

{

fstream data;

data.open(fileName, ios::out);

vector<string> r;

r.push\_back("background " + loadPath + " 0 0\n");

#pragma region - write block -

for (unsigned int i = 0; i < block.size(); i++)

{

r.push\_back("block " + block[i].path + " " + std::to\_string(block[i].x) + " " + std::to\_string(block[i].y) + "\n");

}

#pragma endregion

#pragma region - write door -

for (unsigned int i = 0; i < door.size(); i++)

{

r.push\_back(door[i].GetType() + " " + door[i].path + " " + std::to\_string(door[i].x) + " " + std::to\_string(door[i].y) + " " + std::to\_string(door[i].GetSwitchMapIndex()) + "\n");

}

#pragma endregion

r.push\_back("up " + std::to\_string(upMap) + "\n");

r.push\_back("down " + std::to\_string(downMap) + "\n");

r.push\_back("left " + std::to\_string(leftMap) + "\n");

r.push\_back("right " + std::to\_string(rightMap) + "\n");

r.push\_back("passerbyNumber " + std::to\_string(passerbyMaxSize) + "\n");

string passID = "";

for (unsigned int i = 0; i < passerbyID.size(); i++)

{

string tempid = " " + std::to\_string(passerbyID[i]);

passID += tempid;

}

r.push\_back("passerbyID" + passID + "\n");

for (unsigned int i = 0; i < r.size(); i++)

{

data << r[i];

}

data.close();

}

#pragma endregion

}

1. CLibrary.h

#pragma once

#include "Refactor.h"

namespace myLibrary

{

char\* ConvertCharPointToString(string, string, int = -1);

char\* ConvertCharPointToString(string);

COLORREF ConvertStringToColor(string);

bool ConvertStringToBoolen(string);

void getFolderFile(string, vector<string>\*);

int getFolerFileNumber(string);

string getFileName(string); //得到file的檔名(不含副檔名)

vector<string> SplitString(string);

void DeleteCharPoint(vector<char\*>&);

int GetPostive(int);

int GetRandom(int, int); //得到Random數字，min <= k <= max

int ConvertStringToInt(string);

class BitmapPath

{

public:

BitmapPath() {};

BitmapPath(string \_fold, string \_name, int \_number) {

ziliaojia = \_fold;

name = \_name;

number = \_number;

color = RGB(255, 255, 255);

};

BitmapPath(string \_fold, string \_name, int \_number, COLORREF \_color) {

ziliaojia = \_fold;

name = \_name;

number = \_number;

color = \_color;

};

BitmapPath(vector<string> \_path, COLORREF \_color) {

path = \_path;

color = \_color;

};

~BitmapPath() {};

string ziliaojia;

string name;

int number;

COLORREF color;

vector<string> path;

};

}

using namespace myLibrary;

namespace game\_framework

{

int ScreenX(int, int);

bool IsPointInRect(CPoint, CRect);

bool IsRectCollision(CRect, CRect);

#pragma region - timer -

class CTimer

{

public:

CTimer();

CTimer(int);

CTimer(double);

~CTimer();

void CountDown();

double GetTime();

int GetTime(int);

bool IsTimeOut();

void ResetTime(double);

void ResetTime();

void operator=(CTimer); //運算子多載，方便在GameStateRun::OnBeginState中重構Timer

private:

const double reflash = (1000 / GAME\_CYCLE\_TIME); //一秒刷新幾次

double time;

double initTime;

};

#pragma endregion

#pragma region - CAnimate -

class CAnimate

{

public:

CAnimate();

~CAnimate();

CMovingBitmap\* AddBitmap(int, COLORREF = CLR\_INVALID); // 增加一張圖形至動畫(圖的編號及透明色)

CMovingBitmap\* AddBitmap(char\*, COLORREF = CLR\_INVALID); // 增加一張圖形至動畫(圖的編號及透明色)

void LoadBitmap(vector<string>, COLORREF = CLR\_INVALID); //以字串陣列載入圖片

void LoadBitmap(string, string, int, COLORREF = CLR\_INVALID);

void LoadBitmap(BitmapPath);

int Height(); // 取得動畫的高度

int Left(); // 取得動畫的左上角的 x 座標

void OnMove(int); // 依照方向更換bitmap

void OnMove();

void OnShow(); // 將動畫貼到螢幕

void Reset(); // 重設播放順序回到第一張圖形

void SetTopLeft(int, int); // 將動畫的左上角座標移至 (x,y)

int Top(); // 取得動畫的左上角的 y 座標

int Width(); // 取得動畫的寬度

bool IsNull();

void ReleaseAnimate();

void SetValid(bool);

bool GetValid();

void SetIndex(int);

int GetIndex();

int GetIndexSize() {

return bmp.size();

};

void CopyAnimateInformation(CAnimate\*);

void ResetDelayTime(double);

CRect GetRect();

private:

vector<CMovingBitmap> bmp;

int bmp\_amount;

int x, y; // 動畫的座標

int bmp\_index;

bool isValid;

CTimer delayTimer;

double delayTime;

};

#pragma endregion

#pragma region - CLayer -

class CLayer

{

public:

CLayer();

~CLayer();

void SetLayer(int);

int GetLayer();

private:

int layer;

};

#pragma endregion

#pragma region - CDialog -

class CDialog

{

public:

CDialog();

CDialog(string, string);

~CDialog();

bool GetTriggered();

void SetTriggered();

void Initialize();

int GetTxtSize();

string GetAvatar(unsigned int);

string GetDialogTxt(unsigned int);

COLORREF GetDialogColor(unsigned int);

string GetMode();

void LoadTxt();

private:

string path;

string mode;

vector<string> txt;

vector<string> avatar;

vector<COLORREF> color;

bool IsTriggered;

bool CanReTrigger;

};

#pragma endregion

#pragma region - Camera -

class CCamera

{

public:

CCamera();

~CCamera();

int GetX();

int GetY();

void SetXY(int, int);

void AddX(int);

void AddY(int);

void SetCanMoving(bool);

void Initialize();

void Reset();

void SetCameraBoundary(int, int);

static CCamera\* Instance();

private:

int x, y;

int max\_left, max\_right;

bool canMoving;

static CCamera camera;

};

#pragma endregion

#pragma region - CButton -

class CButton

{

friend class CButtonManager;

public:

CButton();

CButton(const CButton&);

CButton(BitmapPath, CPoint, bool, bool); //路徑、初始點、初始狀態, 是否需要與滑鼠碰撞

int GetX();

int GetY();

void SetXY(int, int);

void SetState(bool);

bool GetState();

int Width();

int Height();

void SetName(string \_name) {

name = \_name;

};

string GetName() {

return name;

};

void LoadBitmap();

void LoadBitmap(BitmapPath);

void OnMove(); //更新Button狀態，有需要時也可更改顯示位置

void OnShow();

void Initialize();

void Initialize(CPoint, bool);

void SetValid(bool);

bool GetValid();

void CollisonMouse(CPoint);

void ClickButton();

bool IsCollisionMouse(CPoint);

CAnimate\* GetAnimate();

void operator=(const CButton&);

private:

CAnimate animation;

BitmapPath loadpath;

string name;

int x, y;

bool needCollision;

bool state;

bool valid;

};

#pragma endregion

#pragma region - CInteger -

class CInteger

{

public:

CInteger(int = 5); // default 5 digits

void Initialize(CPoint, int, int = 2);

void Add(int n); // 增加整數值

int GetInteger(); // 回傳整數值

void LoadBitmap(string, string);

void SetInteger(int); // 設定整數值

void SetTopLeft(int, int); // 將動畫的左上角座標移至 (x,y)

void ShowBitmap(); // 將動畫貼到螢幕

int NUMDIGITS;

bool IsNull() {

return !isBmpLoaded;

};

void SetValid(bool \_flag) {

isValid = \_flag;

};

bool GetValid() {

return isValid;

};

CMovingBitmap digit[11]; // 儲存0..9及負號之圖形(bitmap)

CMovingBitmap number[4];

CLayer layer;

private:

int x, y; // 顯示的座標

int n; // 整數值

bool isBmpLoaded; // 是否已經載入圖形

bool isValid;

};

#pragma endregion

#pragma region - CAction -

class CAction

{

public:

CAction();

void OnMove(string);

void OnShow();

void Initialize();

void LoadAction(string, BitmapPath); //action ,BitmapPath

int Height(); // 取得動畫的高度

int Width(); // 取得動畫的寬度

int Left(); // 取得動畫的左上角的 x 座標

int Top(); // 取得動畫的左上角的 y 座標

void SetTopLeft(int, int); // 將動畫的左上角座標移至 (x,y)

bool IsNull();

void SetAction(string);

string GetAction();

void SetValid(bool);

bool GetValid();

void SetIndex(int);

int GetIndex();

void SetFaceTo(string);

string GetFaceTo();

CLayer layer;

CMovingBitmap\* GetNowBitmap();

CRect GetRect();

int GetActionHeight(string);

private:

CMovingBitmap\* nowBitmap;

vector<CMovingBitmap>\* nowAction;

map<string, vector<CMovingBitmap> > paser;

CTimer delayTimer;

map <string, double> delayAdapter;

double delay\_run = 0.08;

double delay\_idle = 0.12;

double delay\_jump = 0.16;

bool isBmpLoaded; // 是否已經載入圖形

bool isValid;

int x, y; // 顯示的座標

int action\_index;

string action;

string faceTo; // 目前面對方向

};

#pragma endregion

#pragma region - CEnd -

class CEnd

{

public:

CEnd();

CEnd(string);

~CEnd();

string GetBmpPath(int);

string GetTxt(int);

void SetGetEnd(bool f = true) {

isGet = f;

}

bool IsGetEnd() {

return isGet;

};

private:

string endName;

vector<string> bmpPath;

vector<string> txt;

void LoadEnd();

void LoadBmpTxt(string);

bool isGet;

};

#pragma endregion

#pragma region - CToumeiImage -

class CToumeiImage

{

public:

CToumeiImage();

CToumeiImage(int, int);

CToumeiImage(string, int, int);

~CToumeiImage();

void SetAlpha(int \_a);

int GetAlpha() {

return alpha;

};

void SetBmp(string \_path);

void SetFadeInOut(int, int);

void FadeIn(); //淡入 每次alpha+3

void FadeOut(); //淡出 每次alpha-4

void DrawImage();

bool temp = false;

private:

CBitmap bmp;

string bmpLoadPath;

int alpha = 0;

int dFadeInValue = 3;

int dFadeOutValue = -4;

};

#pragma endregion

#pragma region - CWindows -

#pragma region - CWindow -

class CWindow

{

friend class CScrollWindow;

public:

CWindow();

CWindow(CPoint);

~CWindow();

void LoadResource();

void Initialize(CPoint);

void Clear();

void Open();

void Close();

void UpdateMouse(CPoint \_p) {

mousePos = \_p;

};

bool IsCollisionClose(CPoint);

bool IsOpen();

void SetCloseButton(CPoint);

void SetXY(CPoint);

void CollisionClose(CPoint);

void OnCycle();

void OnShow();

protected:

int x, y;

CButton\* closeButton = NULL;

CMovingBitmap background;

CPoint mousePos;

private:

bool isOpen;

bool isLoaded;

};

#pragma endregion

#pragma region - CScrollWindow -

class CScrollWindow : public CWindow

{

public:

CScrollWindow();

~CScrollWindow();

void LoadResource();

void Initialize(CPoint);

void Close();

void OnCycle();

void OnShow();

void OnScrolling(short \_s);

string GetCollisionButtonName(CPoint);

private:

CMovingBitmap cover, cover\_bottom;

vector< vector <CButton> > endingVector;

vector<string> endName;

int colNum, rowNum;

int img\_x, img\_y;

int img\_height, img\_width;

int limit\_top, limit\_buttom;

};

#pragma endregion

#pragma region - CSwitchWindow -

class CSwitchWindow : public CWindow

{

public:

CSwitchWindow();

~CSwitchWindow();

void LoadResource(string);

void Initialize(CPoint);

void Open();

void Close();

bool CollisionArrow(CPoint);

void ClickWindows(CPoint, string);

void OnCycle();

private:

int index = 0;

int step = 0;

vector< vector<CMovingBitmap> > bmp;

CMovingBitmap arrow\_left;

CMovingBitmap arrow\_right;

void SetIndex(int);

void Switchwindow(string);

};

#pragma endregion

#pragma region - CPanel -

class CButtonManager;

class CPanel : public CWindow

{

public:

CPanel();

~CPanel() {

Clear();

};

void CreatButton();

void LoadResource();

void Initialize(CPoint);

void Clear();

void UpdateMouse(CPoint \_p) {

mousePos = \_p;

};

void OnCycle();

string GetCollisionButtonName();

private:

CButtonManager\* btnManager = NULL;

CPoint mousePos;

int img\_x, img\_y;

};

#pragma endregion

#pragma endregion

#pragma region - CStatusBoard -

class CStatusBoard

{

public:

CStatusBoard();

void Load();

void Initialize(CPoint, int, int); // 初始點, 血量, EQ

void UpdateBar(int, int);

void SetXY(CPoint);

void SetDeltaBar(int, int);

int GetHP() {

return hp;

};

int GetEQ() {

return eq;

};

void OnCycle(int, int);

protected:

CMovingBitmap HP\_frame, EQ\_frame;

CMovingBitmap HP\_bar, EQ\_bar;

CMovingBitmap avatar, avatar\_frame;

private:

int dHP, dEQ; // 一次扣的寬度

int hp, eq;

};

#pragma endregion

#pragma region - CBossBoard -

class CBossManager;

class CBossBoard

{

friend class CBossManager;

public:

CBossBoard();

void Load();

void Initialize(CPoint); // 初始點

void UpdateBar(int);

void SetXY(CPoint);

void SetDeltaBar(int);

void SetShow(bool);

void SetHP(int \_hp);

int GetHP() {

return hp;

};

bool IsShow() {

return isShow;

};

void OnCycle(CBossManager\*);

protected:

map<string, CMovingBitmap> bossAvatar;

CMovingBitmap NULLAvatar;

CMovingBitmap HP\_frame;

CMovingBitmap HP\_bar;

CMovingBitmap avatar, avatar\_frame;

private:

CPoint initPos;

double dHP; // 一次扣的寬度

int hp;

bool isShow;

};

#pragma endregion

}