國立臺北科技大學

2022 Spring 資工系物件導向程式實習

期末報告

Dungeon Munchies 餐癮地城



第９組

108820021 李以謙

108820031 簡上博

**目錄**

1. **簡介**
2. 動機 …………………………………………………………. 2
3. 分工 …………………………………………………………. 2
4. **遊戲介紹**
5. 遊戲說明 ....…………………………………………………. 2
6. 遊戲圖形 .....…………………………………………………. 5
7. 遊戲音效 .....…………………………………………………. 8
8. **程式設計**
9. 程式架構 .....…………………………………………………. 9
10. 程式類別 .....…………………………………………………. 10
11. 程式技術 .....…………………………………………………. 12
12. **結語**
13. 問題及解決方法 ….…………………………………………. 15
14. 時間表 ………….….………………………………………… 18
15. 貢獻比例 ….………….……………………………………… 20
16. 自我檢核表 ….………………………………………………. 20
17. 收獲 ….………………………………………………………. 20
18. 心得、感想 ….………………………………………………. 22
19. 對於本課程的建議 ….………………………………………. 23
20. 附錄 ….………………………………………………………. 23
21. **簡介**
    1. 動機

根據老師的規定，遊戲需要有關卡，且不要塔防、射擊、炸彈人、打磚塊等等。許多經典遊戲都不建議，所以我們想到的就是傳統打怪過關的卷軸動作RPG遊戲。朝這個方向找了許久，有些遊戲不符合規定，有些又覺得太簡單。後來，才在steam上面發現了這款「Dungeon Munchies 餐癮地城」，評價非常好，還是臺灣團隊製作，而且難易度也適中，是我們認為在一學期可以做得較為完整且較好展現的遊戲，因此決定定微我們的主題。

* 1. 分工

利用github做版本管理，一起安排進度，再分配工作。

分開作業時，積極溝通目前進度及遇到的問題，兩人共同解決。

大致分工內容：

李以謙：角色動作、物件碰撞、地圖圖片及相關設定。

簡上博：處理圖片、各種系統(如開始結束畫面、角色狀態面板、素材及道具、鍛造台、對話等)、怪物。

1. **遊戲介紹**
   1. 遊戲說明
      1. 操作說明(遊玩方式)
2. 開始選單
   * + 1. 點選：滑鼠左鍵。
       2. 左下角圖示可連結至官方網站、頻道。
       3. 遊玩說明內有操作說明，欲查看作弊說明(密技)，請點選遊玩說明之標題。
3. 角色動作
4. 面對方向:滑鼠位置。
5. 角色向左邊移動：A。
6. 角色向右邊移動：D。
7. 角色向下面移動：S。
8. 角色跳躍：W、SPACE。
9. 角色翻滾：CTRL (翻滾可穿越怪物) 。
10. 角色攻擊：滑鼠左鍵。
11. 系統操作
12. 角色與地圖物件互動(傳送門、開/關對話、鍛造台)：E。
13. 對話訊息切換：滑鼠左鍵。
14. 查看角色狀態：TAB。
15. 暫停：CTRL + Q。
16. 全螢幕切換：CTRL + F。
17. 關閉程式：ESC。
18. 鍛造台操作
19. 點選：滑鼠左鍵。
20. 若素材及燈泡足夠，可點選畫面右下之烹調。  
    烹調後，素材及燈泡會依道具受到消耗，效果會直接附加。  
    左下部分會顯示使用中道具之圖示。

若想取消道具使用，可點選左下部分之圖示，燈泡及素材都會歸還。

1. 關閉鍛造台畫面：E、點選右上角之X。
2. 作弊說明(密技)
3. 直接通關：H。
4. 直接死亡：G。
5. 角色拿起 / 放下綠劍：Q。
6. 回復角色50血量：R。
7. 增加角色10攻擊：T。
8. 設定燈泡量為1000：Y。
9. 設定各素材量為100：U。
10. 直接跳到指定關卡：數字鍵1~7。
11. 直接取得/丟棄所有道具：P
12. 勝利畫面
13. 點選：滑鼠左鍵。
14. 點選GAME COMPLETED會結束遊戲。
15. 點選PLAY AGAIN會回到開始選單。
16. 死亡畫面
17. 點選：滑鼠左鍵。
18. 點選GAME OVER會結束遊戲。
19. 點選PLAY AGAIN會回到開始選單。
    * 1. 遊戲規則

本遊戲一共有7關，通過傳送門連接關卡，您可以打倒沿途的魔物取得素材，並利用鍛造台打造道具增強自己，或是透過翻滾躲避遇到的魔物，抵達最後一關。

請擊倒魔物王取得獲勝吧！

死亡條件：血量歸零。

獲勝條件：打倒第七關的王。

p.s.第五關存在隱藏通道可直接傳送至第七關與魔物王對戰，然而，這種貿然的接觸會讓魔物王更加狂暴！

* + 1. 特殊功能

1. 可以在空中跳躍 (道具—鹽酥大蚊)。
2. 增加20最大生命 (道具—鹽烤蝦)。
3. 增加移動速度40% (道具—快樂草手卷)。
4. 近戰武器擊中敵人時，附帶額外3點傷害 (道具—炸香蕉)。
5. 不攻擊一段時間後，下次近戰傷害增加15點 (道具—炭烤蝦鉗)。
6. 每十秒恢復生命3點 (道具—芭樂汁)。
7. 綠劍 (由變種之仙人掌魔物掉落)。
8. 直接通關（密技按鍵：H）。
9. 直接死亡（密技按鍵：G）。
10. 角色拿起 / 放下綠劍（密技按鍵：Q）。
11. 回復角色50血量（密技按鍵：R）。
12. 增加角色10攻擊（密技按鍵：T）。
13. 設定燈泡量為1000（密技按鍵：Y）。
14. 設定各素材量為100（密技按鍵：U）。
15. 直接跳到指定關卡（密技按鍵：數字鍵1~7）。
16. 直接取得/丟棄所有道具（密技按鍵：Ｐ）
    1. 遊戲圖形

角色(未持綠劍):

|  |  |  |
| --- | --- | --- |
| 站立 | 攻擊 | 翻滾 |
|  | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\s_left_attack02.bmp | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\left_roll04.bmp |

角色(綠劍):

|  |  |
| --- | --- |
| 站立 | 攻擊 |
| C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\g_left_stand.bmp | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\g_attack02_left.bmp |

魔物

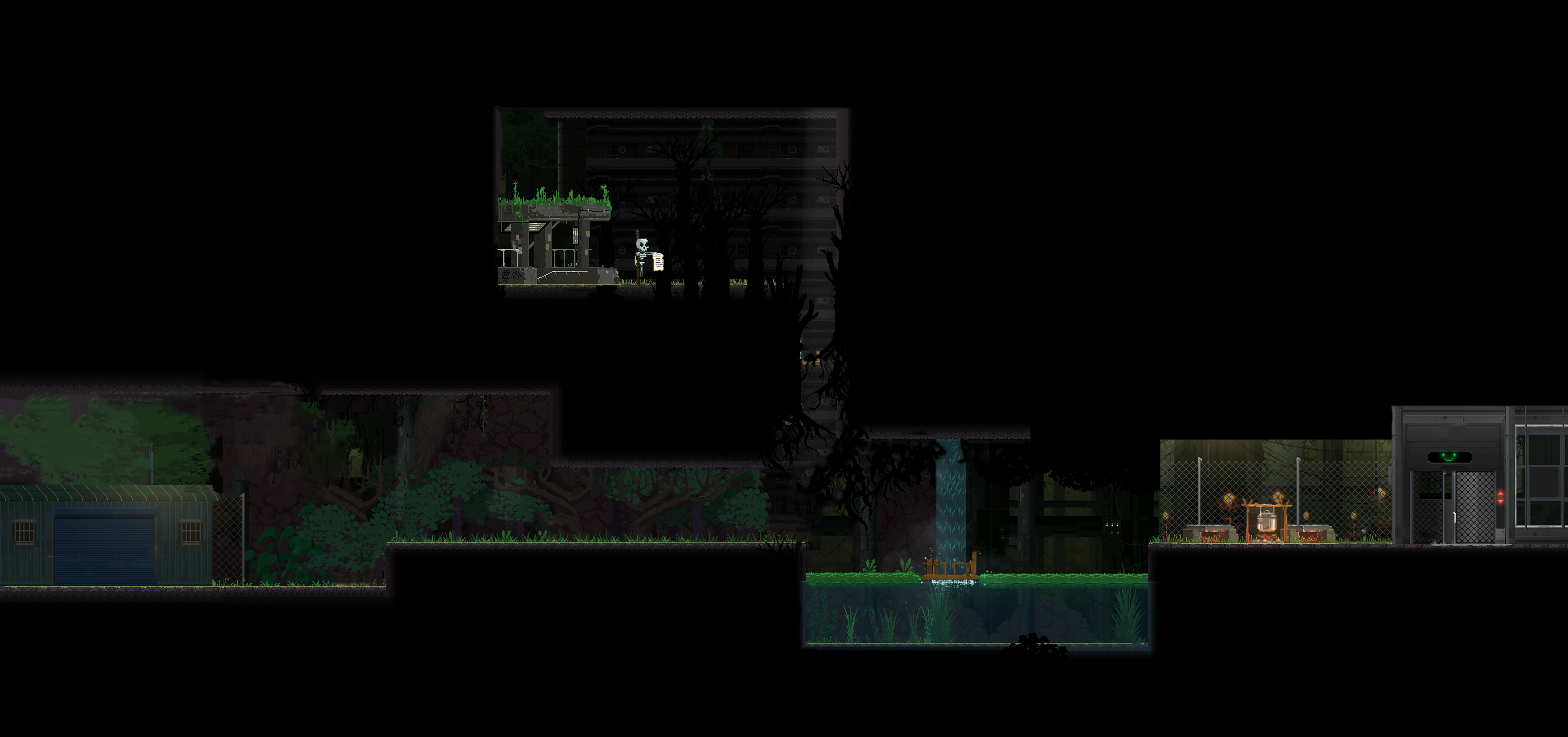
|  |  |  |
| --- | --- | --- |
| 香蕉 | 樹 | 仙人掌 |
| C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\monster_banana.bmp | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\monster_tree_attack05_left.bmp |  |
| 蝦子 | | 蚊子 |
| C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\monster_shrimp_attack01_left.bmpC:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\monster_shrimp_attack03_left.bmp | | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\monster_mosquito01_left.bmp |
| 王 | | |
|  |  |  |

素材與道具道具

|  |  |  |  |
| --- | --- | --- | --- |
| 炸香蕉 | | 快樂草手卷 | |
| C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\source_banana_attack.bmp |  | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\source_grass_fast.bmp |  |
| 芭樂汁 | | 炭烤蝦鉗 | |
| C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\source_guava_juice_blood.bmp |  | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\source_shrimp_attack.bmp |  |
| 鹽烤蝦 | | 鹽酥大蚊 | |
| C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\source_shrimp_blood.bmp | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\prop_shrimp_blood_icon.bmp | C:\Users\Ｕｓｅｒ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\source_mosquito_jump.bmp |  |

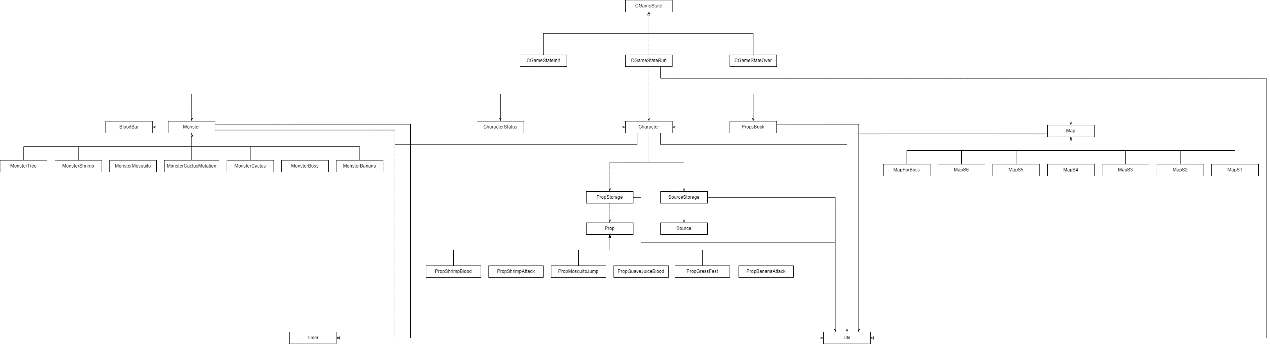
遊戲畫面：





* 1. 遊戲音效
     1. start\_menu\_audio.mp3：開始選單之背景音樂，由原遊戲中取材。
     2. music01.mp3：第一關之背景音樂，由原遊戲中取材。
     3. music02.mp3：第二關之背景音樂，由原遊戲中取材。
     4. music03.mp3：第三關之背景音樂，由原遊戲中取材。
     5. music04.mp3：第四關之背景音樂，由原遊戲中取材。
     6. music05.mp3：第五關之背景音樂，由原遊戲中取材。
     7. music06.mp3：第六關之背景音樂，由原遊戲中取材。
     8. music07.mp3：第七關(王關)之背景音樂，取材於<https://reurl.cc/RrQK66>。
     9. victory.mp3：勝利畫面之背景音樂，取材於<https://reurl.cc/GxlzXy>。
     10. lose.mp3：死亡畫面之背景音樂，取材於<https://reurl.cc/7DWZRb>。
     11. choose.mp3：鼠標滑上開始選單之選項、X等的音效，由原遊戲中取材。
     12. pot.mp3：經過鍛造台之音效，取材於<https://reurl.cc/QLO16Z>。
     13. cook.mp3：使用鍛造台烹煮之音效，由原遊戲中取材。
     14. attack\_hu.mp3：角色空手攻擊之音效，由原遊戲中取材。
     15. attack\_ta.mp3：角色使用綠劍攻擊之音效，由原遊戲中取材。
     16. recycle\_can.mp3：經過回收桶之音效，由原遊戲中取材。
     17. roll2.mp3：翻滾時音效，由原遊戲中取材。

1. **程式設計**
   1. 程式架構



(上圖之drawio檔案連結：https://reurl.cc/7DLeE1)

3個遊戲狀態(CGameStateInit、CGameStateRun、CGameStateOver)皆繼承CGameState。

7張地圖(MapS1、MapS2、MapS3、MapS4、MapS5、MapS6、MapForBoss)皆繼承Map。

6個道具(PropBananaAttack、PropGrassFast、PropGuavaJuiceBlood、PropMosquitoJump、PropShrimpAttack、PropShrimpBlood)皆繼承Prop。

7種魔物(MonsterTree、MonsterShrimp、MonsterMosquito、MonsterCactusMutation、、MonsterCactus、MonsterBoss、MonsterBanana)皆繼承Monster。

CGameStateRun主要由Character、CharacterStatus、Monster、Map構成。

Chracter擁有SourceStorage及PropStorage，分別儲存角色擁有之素材及道具。另外，角色擁有的燈泡數量亦記錄於Character。

PropStorage內含6種道具之資料，SourceStorage內含6種素材之資料。

Util.h為紀錄資料(如：const、enum之值)的標頭檔。

Timer為計算時間之工具。

* 1. 程式類別

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| mygame | 196 | 1472 | 主程式 |
| Character | 219 | 1548 | 角色 |
| CharacterStatus | 42 | 250 | 角色狀態 |
| Map | 121 | 264 | 地圖 |
| MapS1 | 39 | 242 | 關卡1地圖 |
| MapS2 | 40 | 320 | 關卡2地圖 |
| MapS3 | 47 | 314 | 關卡3地圖 |
| MapS4 | 48 | 370 | 關卡4地圖 |
| MapS5 | 45 | 439 | 關卡5地圖 |
| MapS6 | 50 | 341 | 關卡6地圖 |
| MapForBoss | 29 | 132 | 王關地圖 |
| Monster | 138 | 454 | 魔物 |
| MonsterBanana | 29 | 142 | 魔物香蕉 |
| MonsterCactus | 29 | 148 | 魔物仙人掌 |
| MonsterCactusMutation | 29 | 141 | 魔物變種仙人掌 |
| MonsterTree | 39 | 368 | 魔物樹 |
| MonsterShrimp | 37 | 320 | 魔物蝦子 |
| MonsterMosquito | 38 | 231 | 魔物蚊子 |
| MonsterBoss | 90 | 802 | 魔物王 |
| BloodBar | 25 | 108 | 怪物血條 |
| Source | 27 | 65 | 素材 |
| SourceStorage | 36 | 90 | 素材儲存空間 |
| Prop | 24 | 38 | 道具 |
| PropStorage | 35 | 67 | 道具儲存空間 |
| PropsBook | 84 | 531 | 鍛造台 |
| PropBananaAttack | 19 | 33 | 道具炸香蕉 |
| PropGrassFast | 19 | 33 | 道具快樂草手卷 |
| PropGuavaJuiceBlood | 19 | 33 | 道具芭樂汁 |
| PropShrimpAttack | 19 | 33 | 道具炭烤蝦鉗 |
| PropShrimpBlood | 19 | 33 | 道具鹽烤蝦 |
| PropMosquitoJump | 19 | 33 | 道具鹽酥大蚊 |
| Timer | 28 | 74 | 計時器 |
| Util | 118 | 0 | 資料紀錄 |
| 總行數 | 1796 | 9469 | 11265 |

* 1. 程式技術
     1. vector：  
        在CGameStateRun中，以vector<Monster\*>建立每關的魔物。  
        在PropsBook中，setIconPosition函式用vector<CMovingBitmap>方便做自動化處理。
     2. pointer：  
        在許多情況都需要保持資料的同步，比起用pass by value的方式抓取資料處理再回傳，使用pass by reference真的方便許多，因為改同一份資料自然資料會保持一致，也較不容易出現不可預期的錯誤。  
        例如：在PropsBook(鍛造台畫面)進行操作時，它就是直接利用pointer的方式抓取character擁有的素材及道具的資料。CharacterStatus亦是如此。在Timer中，也有傳入boolean之pointer來控制外部行為。
     3. 角色翻滾及跳躍

由於一開始翻滾的程式碼寫得不好，因此在將翻滾的動作與動畫連接上出現了極大的困難，最後只能靠不斷的微調時間和執行次數，才能讓動畫和動作比較配合得上。

跳躍則是需要物理公式以及地面的判定，當起跳的時候給角色一個初速度，並隨時判定角色是上升或是下降，速度會隨著時間改變，當下一次判定落到地面或以下時再將角色設在地圖的地板上。

* + 1. Timer  
       抓取當下時間進行相減算出時間間距。  
       有startTime及finishTime之變數分別記錄開始及結束時間，當動作開始時，記錄下startTime之時間，而finishTime則是持續紀錄，當finishTime – startTime >= 設定值時，代表時間到。  
       衍伸：將控制行為之boolean指標傳入，當時間到時修改boolean值，方便的函式更助於程式的撰寫。
    2. 各式判定  
       每樣物體都設置了LeftX、RightX、TopY、BottonY，這些數值代表該物體之左、右、上、下(絕對位置)，並運用其範圍進行判定。  
       應用舉例：兩物體(A、B)重疊之判定分成了左右是否重疊和上下是否重疊，當下述條件達成即為重疊(越右、下數值越大)：  
       左右判定：  
       A之LeftX或RightX位於B之LeftX和RightX之間，或是  
       A之LeftX比B之LeftX小且A之RightX比B之RightX大(A面積比B大，A橫跨B之情況)，或是  
       A之LeftX比B之LeftX大且A之RightX比B之RightX小(A面積比B小，A位在B之中的情況)  
       上下判定如左右判定之方式判斷是否重疊，若左右重疊且上下重疊即表示兩物體重疊。
    3. 地圖移動  
       當角色移動，地圖會隨之變化。在角色抵達畫面X軸中心前為角色移動(地圖不動)，抵達後改為背景地圖移動(角色顯示維持在中間)。地圖上下移動則是另外設定條件，以此調整到最適合的畫面。且地圖移動需連動物件位置隨之改變(物件顯示時抓地圖移動)，讓物件顯示在地圖正確的位置上。
    4. 角色受到傷害  
       角色受到傷害後，會損失血量，根據受攻擊的方位被擊退，並進入短暫的無敵時間。
    5. 怪物血條顯示  
       怪物血量分為10等分，依照怪物當下的血量/最大血量顯示不同的血條。顯示位置根據怪物之LeftX及TopX搭配地圖移動。
    6. 受傷動畫  
       使用一張全黑的圖(利用CMovingBitmap中去除特定顏色產生透明的效果)，若角色或魔物受到傷害，以及短的時間換成黑色的圖，再換回原動作，就達成了閃爍的效果。  
       利用了兩個Timer，一個計算單次閃爍、回到原圖的時間，一個計算受傷動畫會維持多久。
    7. 魔物蚊子飛行  
       把飛行模式分成了4種，分別為左上、右上、左下、右下之飛行方向，再來計算隨機時間((rand() % 200) / 100)使其變換飛行模式，達成隨機飛行的效果。  
       計算隨機時間時，想要控制每次飛行模式最多2秒，並非直接使用取2之餘數，而是先取200之餘數再除以100是為了增加飛行之隨機性。若直接取2之餘數，結果只有0、1、2，但先取200之餘數就能有更多不同的可能性。  
       變換飛行模式時，依照左上、左下、右下、右上的順序，其目的是為了讓魔物不偏離原設定的地方過多，大致以繞圈的方式飛行。
    8. 魔物樹、蝦子動作  
       計算魔物與角色之距離，以此距離判定是休息、走路、攻擊狀態。  
       若角色距離過遠會停在原地休息，角色進入活動範圍則會走向角色，進入攻擊範圍則會攻擊。攻擊時魔物不會移動。攻擊判定採上述之重疊判定方式，若角色位於攻擊範圍內即為攻擊成功。  
       血量歸零即死亡，會在地板上顯示遺體和素材圖示。
    9. 魔物王、勝利畫面  
       王有5種行為，分別是走路、捶地攻擊、衝撞攻擊、召刺攻擊、死亡。行為模式主要依照與角色的距離，近距離會施展捶地攻擊，中距離會走路，遠距離攻擊則是依據血量是否大於50%施展衝撞攻擊或召刺攻擊。

魔物王血量歸零即死亡，死亡會撥放死亡動畫，當動畫完畢進入勝利畫面。

* + 1. 鍛造台  
       用pointer抓取character sourceStorage和propStorage的資料，檢查道具是否使用中或素材是否足夠，若素材足夠即可合成道具，道具使用中會在對應位置上顯示彩色的道具圖示，且無法再合成該道具，右下角鍋子之圖示會隨之改變。  
       道具圖示的顯示根據圖片資訊(每格的X、Y)寫成公式計算，自動判斷為何種case，撰寫函式回傳每種case對應的結果，就不用每次都自己設定，更利於擴充程式，也使程式更整潔。

1. **結語**
   1. 問題及解決方法
      1. 檔案建置位置  
         一開始想新增class時一直出錯，程式編譯錯誤，但又覺得自己的程式碼沒有出錯，研究了一下才發現是建置檔案時位子沒有建對。然而，這不單單是刪掉重建這麼簡單，因為執行資訊會被記錄在vcxproj檔等地方，需要修改其中內容，程式才能正常運作。可以看git紀錄來做對應的修改，或直接rollback至上一次完好的commit直接重做。vcxproj檔若不知如何編輯可以用Notepad++開啟，並搭配搜尋找出建錯的檔案資訊。
      2. 函式作用方式(資料處理無效等問題)  
         函式可以選擇pass by reference或是pass by value，但使用pass by value要記得回傳處理好的資料，再從外面接起來，若是想直接修改原資料還是建議使用pass by reference。
      3. 物件生命週期  
         有時物件的生命週期會因種種原因出錯，但根據個人經驗，變數改為pointer或是函式是pass by reference，會有效解決這類問題。
      4. 物件的全面性  
         若是寫物件時只把當下要做的功能完成，像宣告變數，該變數可能會被其它的class修改，卻因為當前只要做物件內部操作而沒有寫set和get函式，之後處理其他物件要使用到該物件資料才回去補，需要做很多雜事容易使撰寫者思緒錯亂，導致錯誤率升高。尤其是工具類的class，要保持為別人服務的精神，思考該物件怎麼使用最為方便，盡可能的完善它，相信不僅僅團隊開發更效率更高，對未來自己使用該物件也很有幫助。  
         因此，在撰寫程式碼時，需要思考整體程式的流程，不單單只把該物件寫好，還需考量到與其他物件的互動，且若是發現問題需盡早處理，否則後面程式碼擴充較為困難外，要回頭去修之前的bugs也更加麻煩，因為物件間互動會用到物件本身資料，導致其它行為也可能會受到影響。
      5. 程式碼質量問題  
         程式碼的整潔非常重要，若是雜亂不堪，括號沒有對齊、格式不統一，容易使撰寫者出錯、協作者更是難以修改程式，因此每次寫完都要整理格式，最快的方式就是使用編譯器格式化文件的功能。  
         除此之外，有時寫物件的行為判斷會有許多條件，而擴充程式時可能又會要新增條件，導致判斷式冗長，難以理解。像是攻擊行為要判斷距離外，還需判斷當下是否為可以攻擊的狀態，擴充功能可能又要再附加新的條件(如同樣攻擊無法連續使用等)。因此建議行為或判斷盡可能地分出函式，如此除了更好理解程式碼，也可以重複使用，而且Debug時，因為程式碼十分乾淨，職責分配清楚，也較容易找出問題。
      6. Class編譯順序  
         當class編譯前要先編譯過其它的class，或是說物件之間有上下層關係(該class內有宣告某class之變數)時，需在該class的標頭檔，撰寫該class之前打上要先執行的class確保程式編譯順序。  
         例如：要有角色狀態需要先有角色、或是角色擁有道具儲存空間，就需先編譯過道具儲存空間的class。
      7. include檔案置放位子  
         由於範例將include檔案放在cpp檔，一開始我們也都照本宣科，但第一次自己寫了一個class要給其它class使用時，發生錯誤了。  
         除了上述提及的編譯順序問題之外，還有include檔案放置位子也是個關鍵。若class在cpp檔被include，那麼宣告該class的變數只能是pointer，但若不想要變數是指標，其實把include的檔案放到標頭檔(.h檔)就可以了。我個人建議使用工具(如計時器等)時，把class放置於標頭檔內直接宣告一個變數即可。
      8. 工作時間、規劃問題  
         原本為了讓進度一直有進展，約定好每周三晚上一起作業。然而，雙方都有各自的安排，即使後來改至星期一也無法解決約好的時間無法開發的問題。因此，最後我們建置了進度表單，不再固定時間共同開發，只需自己找時間在期限內完成事務即可。若是遇到問題或覺得需要討論就再約討論時間。但這種方式也需要雙方的積極配合，按照表單的進度外，也要積極與對方溝通，減少資訊不對等，或方向不對導致後來才發現要大改的情形。
   2. 時間表

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 李以謙(小時) | 簡上博(小時) | 說明 |
| 1 | 2 | 2 | 選擇題目、練習 |
| 2 | 4 | 6 | 熟悉框架、建造Map和character |
| 3 | 10.5 | 15 | 角色、魔物等圖片、角色行走及跳躍動作、地圖 |
| 4 | 3.5 | 7 | 道具圖片、道具、鍛造台、角色翻滾 |
| 5 | 3 | 4.5 | 鍛造台、素材、角色基本體質 |
| 6 | 5 | 4.25 | 處理地圖、素材和道具儲存空間 |
| 7 | 3.5 | 4.25 | Timer、開始選單、鍛造台、怪物血條 |
| 8 | 5.5 | 8.5 | 角色血量、角色道具能力(攻擊以外)、魔物仙人掌、處理圖片 |
| 9 | 4 | 7.25 | 關卡切換、魔物圖片、怪物和角色的互動、角色無敵時間 |
| 10 | 3.2 | 5 | 地圖左右相對移動、地圖一、魔物蝦子 |
| 11 | 10.5 | 13.5 | 角色與怪物碰撞、怪物與地圖碰撞、怪物相對移動、魔物樹、開始和結束頁面、角色攻擊、道具(bananaAttack和shrimpAttack) |
| 12 | 4.5 | 3.7 | 地圖處理、角色狀態面板、gamePause之調整、角色動作、魔物樹休眠動作 |
| 13 | 10.5 | 17 | 地圖上下相對移動、魔物王、勝利頁面、怪物扣血顯示、掉落物功能、魔物香蕉、燈泡功能 |
| 14 | 2.5 | 10.3 | 地圖處理、開始選單的超連結、受攻擊後的閃爍效果、背景音樂、道具取消功能 |
| 15 | 4 | 6.25 | 地圖處理、回收桶、放大鏡圖示、遊戲操作說明畫面、作弊按鍵、對談功能、按鈕音效、魔物蚊子 |
| 16 | 10.5 | 25.3 | 地圖處理、魔物蚊子、綠劍角色、魔物變種仙人掌、修改各種說明頁面、增加遊戲中的提示、撰寫報告 |
| 17 | 23.5 | 15 | 全部關卡設定完成、撰寫報告、遊戲最終調整 |
| 總共 | 110.2 | 154.8 |  |

* 1. 貢獻比例  
     李以謙：35%；簡上博：65%
  2. 自我檢核表

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 項目 | 完成否 | 無法完成的原因 |
| 1 | 解決Memory leak | ☑已完成 □未完成 |  |
| 2 | 自訂遊戲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 |  |

* 1. 收獲

簡上博：

在程式語法方面個人覺得熟悉了不少，其中最讓我有深刻體悟的就是指標的使用，若是不用指標容易發生種種難以預料之錯誤，像物件的生命週期難以掌握，在被使用過後就直接受到刪除，導致程式出錯等情況，改為指標就能解決。

對函式使用pass by value和pass by reference也有更好的領悟，用pass by value的函式執行過後要記得把回傳的資料接起來才會是經過處理過的資料，不然可以使用pass by reference的方式直接對原資料做處理，也十分方便。

在程式撰寫前，要先盡可能全面的思考程式整體的運作方式，做好規劃才能避免之後又要做許多修改。像是角色移動牽涉地圖的移動，之後每個物件再去抓地圖移動，而非角色移動一起移動所有物件；或是，資料傳遞方式及型態，是否要使用指標等；亦或繼承關係、物件之間的互動等等，思考哪種方式最好擴充功能，如果能先設想好對開發會十分有幫助。

出錯時，若是編譯器報錯可以查看錯誤訊息，若看不懂可以把錯誤訊息放到網路上，查看別人的建議對解開自己的問題非常有幫助。若是邏輯錯誤要先冷靜思考整個程式的流程，鎖定可能出問題的部分設置中斷點，利用Debugger一行一行檢查，這麼做有助於找出錯誤原因，也能對程式有更好的了解。除此之外，可以利用git紀錄跟前一個版本做比較，這也是每做完一件事最好都commit一次的原因，保有最新沒有錯誤的程式比較不會後來全部混在一起。

而在push前要重新建置確定整個程式真的完全沒問題。之前曾經有雖然程式碼相同，但有時能成功編譯，有時又會報錯的情況。後來我猜想是編譯器執行時會幫我緩存檔案來提升下次執行的速度，而當我發現編譯出錯進行修改後重新編譯，因沒有重新建置，即便當下程式沒抓到正確的檔案，但該檔案已於上次執行時緩存，因此程式也能正常運行。而這種問題可能和物件的生命週期有關，可以試試使用指標，或是注意include檔案之位子。

include的檔案可以置放在標頭檔內，這能解決一些宣告變數的問題，若是放在cpp內，則想宣告該class之變數就只能使用指標，若是放在標頭檔中，就可以直接宣告變數，無需使用指標。像Timer和Util等工具類的class，就建議include在標頭檔內。

若在class A編譯前要先編譯過class B也要注意，在class A前面打上class B;，讓程式知道編譯先後順序。

協做方面，先討論好整體架構、統一用法和名稱十分有助於開發。畢竟分工很難完全切割，大部分都會使用到對方撰寫的物件，若是大致知道對方會怎麼做，且將自己需要的功能告訴對方一起創建出來也更為方便，也會減少之後需要大幅修改的情況。而且為他人著想的心十分重要，像如果保持好習慣，盡可能把函式分出去降低程式閱讀的難度，雖然一開始可能會多花一點心思，但絕對有助於開發，凌亂的程式碼真的會讓撰寫者出錯率升高，看程式很亂心情也會不好，進而影響到工作品質。打好程式記得把程式格式化，保持格式統一也十分重要。且完成什麼要積極和對方溝通，讓資訊對等，彼此對目前的進度有所掌握，也好做下一步規劃。

李以謙：

從這堂課我學習到了很多開發程式上會遇到的問題，以及學會如何去解決。之前的課程並不會親自寫到萬行的程式碼，因此每次都對自己的程式架構沒有到非常重視，而透過這堂課，我清楚了解到程式的架構對於整個專案的重要性。越到後面，我越後悔一開始的並沒有仔細思考怎樣的架構及程式碼才是更好、延展性更高的，因為原本的程式碼沒辦法輕易修改了，所以只能再接下提醒自己不要犯相同的錯誤。

* 1. 心得、感想

簡上博：

非常感謝有這個契機自己做了一個遊戲，有這次做遊戲的經驗，真的更理解開發者的心思了。需要花許多時間蒐集、製作素材，藉著這次的機會我也大致學會了使用Photoshop，真的是一個功能強大的圖片編輯軟體。

程式只要複雜一點就容易出現bug，遊玩者可能覺得只是一件小事為何不修，但對於開發者而言需要考慮到各式各樣的情況，可能單純改了這邊，另一部分又會出問題，又可能有更重要的事情因此先放著比較無關緊要的bug，真是辛苦。

同時很感謝老師提供這個契機，也在我們問問題時都不厭其煩的回答，有了這次的經驗我對C++和物件導向的概念及重要性又有更深層的了解，對於打程式時該注意什麼相信也在不知不覺中刻進了腦海裡，真的是收穫滿滿難以言喻。

李以謙：

這堂課花費了我許多的時間，但也是這學期學到最多的一堂課，儘管靠的都是以前教過的東西來進行開發，但神奇的事，再沒有學到新知識的情況下，居然能夠發現自己過去所學可以運用到哪裡，從而讓知識發揮到最大化，而不只是停留在腦海中。

這堂課同時也大大的鍛鍊我的邏輯及數學運算，在開發過程中，偶爾會在一些小地方卡住抑或是要添加一個新功能導致短時間開發無法有進展，通常這時候我會去吃點東西休息一下或是洗澡，在這段期間我比較能更清晰的分析該如何去架構或是邏輯的判斷該如何去設定，想到了再回來coding我認為會比一直在電腦前面沒有清晰仔細的去思考來的更有效率。

另外就是在團隊開發上，儘管我自認為在這個專案花費了不少心力，但相較於組員仍是不足許多，非常感謝他的督促並提供給我許多的建議，讓我在開發過程更加順利，也才能讓我也能參與在我自認為蠻滿意的作品之中。

* 1. 對於本課程的建議

希望能在學期開始時就提供報告模板，根據進度每週記錄。  
如果可以的話希望能一開始就提醒因為全螢幕之要求要注意畫面解析度，這樣就能從一開始用正確的比例製作圖片，不用後來再做調整。

附錄

===============================

mygame.h

===============================

#include "Character.h"

#include "Map.h"

#include "MapForBoss.h"

#include "MapS1.h"

#include "MapS2.h"

#include "MapS3.h"

#include "MapS4.h"

#include "MapS5.h"

#include "MapS6.h"

#include "Util.h"

#include "PropsBook.h"

#include "CharacterStatus.h"

#include "Monster.h"

#include "MonsterCactus.h"

#include "MonsterCactusMutation.h"

#include "MonsterBanana.h"

#include "MonsterShrimp.h"

#include "MonsterTree.h"

#include "MonsterMosquito.h"

#include "MonsterBoss.h"

namespace game\_framework

{

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

// Constants

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

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

// 這個class為遊戲的遊戲開頭畫面物件

// 每個Member function的Implementation都要弄懂

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

class CGameStateInit : public CGameState

{

public:

CGameStateInit(CGame\* g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

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

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

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

protected:

void isChoiceChange();

void OnMove();

void OnShow(); // 顯示這個狀態的遊戲畫面

void instructionsPageShow();

private:

bool isMouseOnArrowR(CPoint point);

bool isMouseOnArrowL(CPoint point);

BeginningStage stage;

InstructionsPage instructionsPage;

int record; //1代表instructionsPage第一頁，2代表第二頁

CMovingBitmap startMenu;

CMovingBitmap startMenuChoice;

CMovingBitmap staff;

CMovingBitmap instructionsPage01;

CMovingBitmap instructionsPage02;

CMovingBitmap instructionsPage03;

CMovingBitmap instructionsPageC;

CMovingBitmap whiteX;

CMovingBitmap arrowL;

CMovingBitmap arrowR;

int choice;

int lastChoice;

int yChoice;

bool onChoice;

bool isMouseOnX;

bool isArrowLShow;

bool isArrowRShow;

};

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

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

// 每個Member function的Implementation都要弄懂

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

class CGameStateRun : public CGameState

{

public:

CGameStateRun(CGame\* g);

~CGameStateRun();

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit(); // 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

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

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

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

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

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

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

void LightBulbOnShow();

void monsterInitialize();

private:

void gameCompleteNoteShow();

void bgmPlayer();

void stopAllBgm();

bool isPlayingStage();

Map\* GetCurrentMap(); //要是playing stage才能抓地圖

Character character;

MapForBoss bossMap;

MapS1 mapS1;

MapS2 mapS2;

MapS3 mapS3;

MapS4 mapS4;

MapS5 mapS5;

MapS6 mapS6;

PropsBook propsBook;

CharacterStatus characterStatus;

Stage currentStage;

Stage lastStage; //for propsBook

bool characterStatusCall;

bool haveCalledCharacterStatus;

bool gamePause;

vector<Monster\*> monsterS1;

vector<Monster\*> monsterS2;

vector<Monster\*> monsterS3;

vector<Monster\*> monsterS4;

vector<Monster\*> monsterS5;

vector<Monster\*> monsterS6;

vector<Monster\*> monsterS7;

bool messageShowFlag;

CPoint mousePosition;

CMovingBitmap lightBulb;

CMovingBitmap gameComplete;

bool gameCompleteFlag;

bool isStageChanged; //for bmgPlayer

bool isCheatPropsOn;

};

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

// 這個class為遊戲的結束狀態(Game Over)

// 每個Member function的Implementation都要弄懂

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

class CGameStateOver : public CGameState

{

public:

CGameStateOver(CGame\* g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

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

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int counter; // 倒數之計數器

bool isBgmPlayed;

CMovingBitmap gameOver;

};

}

===============================

mygame.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ctime>

#include <ddraw.h>

#include <stdlib.h>

#include <stdio.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

#include "Util.h"

namespace game\_framework

{

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

// 這個class為遊戲的遊戲開頭畫面物件

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

CGameStateInit::CGameStateInit(CGame\* g)

: CGameState(g)

{

}

void CGameStateInit::OnInit()

{

//

// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人

// 等的不耐煩，遊戲會出現「Loading ...」，顯示Loading的進度。

//

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

//

// 開始載入資料

//

startMenu.LoadBitmap(IDB\_STARTMENU);

startMenuChoice.LoadBitmap(".\\res\\start\_menu\_choice.bmp", RGB(0, 0, 0));

staff.LoadBitmap(".\\res\\staff.bmp");

instructionsPage01.LoadBitmap(".\\res\\instructions01.bmp");

instructionsPage02.LoadBitmap(".\\res\\instructions02.bmp");

instructionsPage03.LoadBitmap(".\\res\\instructions03.bmp");

instructionsPageC.LoadBitmap(".\\res\\instructions\_c01.bmp");

whiteX.LoadBitmap(".\\res\\white\_x.bmp", RGB(0, 0, 0));

arrowL.LoadBitmap(".\\res\\arrow\_left.bmp", RGB(0, 0, 0));

arrowR.LoadBitmap(".\\res\\arrow\_right.bmp", RGB(0, 0, 0));

CAudio::Instance()->Load(AUDIO\_STARTMENU, "sounds\\start\_menu\_audio.mp3");

CAudio::Instance()->Load(AUDIO\_CHOOSE, "sounds\\choose.mp3");

CAudio::Instance()->Play(AUDIO\_STARTMENU, true);

}

void CGameStateInit::OnBeginState()

{

onChoice = false;

yChoice = 380;

choice = 0;

lastChoice = 0;

stage = stage\_start\_menu;

instructionsPage = instructions\_page01;

record = 1;

isArrowLShow = isArrowRShow = false;

isMouseOnX = false;

}

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

{

if (nChar == KEY\_SPACE)

{

CAudio::Instance()->Stop(AUDIO\_STARTMENU);

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

}

else if (nChar == KEY\_ESC) // 關閉遊戲

{

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

}

}

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

{

if (stage == stage\_start\_menu)

{

if (choice == 1)

{

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

CAudio::Instance()->Stop(AUDIO\_STARTMENU);

}

else if (choice == 2)

{

GotoGameState(GAME\_STATE\_OVER);

}

else if (choice == 3)

{

stage = stage\_staff;

}

else if (choice == 4)

{

stage = stage\_instructions;

}

if (point.x >= 0 && point.x <= 31 && point.y >= 735 && point.y <= 768)

{

ShellExecute(0, NULL, \_T("https://www.facebook.com/dungeonmunchies/"), NULL, NULL, SW\_NORMAL);

}

if (point.x > 31 && point.x <= 73 && point.y >= 735 && point.y <= 768)

{

ShellExecute(0, NULL, \_T("https://twitter.com/dungeonmunchies"), NULL, NULL, SW\_NORMAL);

}

if (point.x > 73 && point.x <= 98 && point.y >= 735 && point.y <= 768)

{

ShellExecute(0, NULL, \_T("https://discord.com/invite/8NyCnxN"), NULL, NULL, SW\_NORMAL);

}

}

else if (stage == stage\_staff)

{

if (point.x >= 1306 && point.y <= 60)

{

stage = stage\_start\_menu;

}

}

else if (stage == stage\_instructions)

{

if (instructionsPage == instructions\_page01 && isMouseOnArrowR(point))

{

instructionsPage = instructions\_page02;

record = 2;

}

else if (instructionsPage == instructions\_page02 && isMouseOnArrowL(point))

{

instructionsPage = instructions\_page01;

record = 1;

}

else if (instructionsPage == instructions\_page02 && isMouseOnArrowR(point))

{

instructionsPage = instructions\_page03;

record = 3;

}

else if (instructionsPage == instructions\_page03 && isMouseOnArrowL(point))

{

instructionsPage = instructions\_page02;

record = 2;

}

if (!(instructionsPage == instructions\_page\_cheat) && point.x >= 380 && point.x <= 980 && point.y >= 50 && point.y <= 120)

{

instructionsPage = instructions\_page\_cheat;

}

else if (instructionsPage == instructions\_page\_cheat && point.x >= 555 && point.x <= 815 && point.y >= 50 && point.y <= 120)

{

if (record == 1)

{

instructionsPage = instructions\_page01;

}

else if (record == 2)

{

instructionsPage = instructions\_page02;

}

else

{

instructionsPage = instructions\_page02;

}

}

if (point.x >= 1306 && point.y <= 60)

{

stage = stage\_start\_menu;

instructionsPage = instructions\_page01;

record = 1;

}

}

}

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

{

if (stage == stage\_start\_menu)

{

if (point.x > 603 && point.x < 770)

{

if (point.y > 380 && point.y < 446)

{

yChoice = 380;

choice = 1;

onChoice = true;

}

else if (point.y > 465 && point.y < 531)

{

yChoice = 465;

choice = 2;

onChoice = true;

}

else if (point.y > 550 && point.y < 616)

{

yChoice = 550;

choice = 3;

onChoice = true;

}

else if (point.y > 635 && point.y < 701)

{

yChoice = 635;

choice = 4;

onChoice = true;

}

else

{

choice = 0;

}

}

else

{

choice = 0;

}

}

else if (stage == stage\_staff)

{

if (point.x >= 1306 && point.y <= 60)

{

if (!isMouseOnX)

{

CAudio::Instance()->Play(AUDIO\_CHOOSE, false);

}

isMouseOnX = true;

}

else

{

isMouseOnX = false;

}

}

else if (stage == stage\_instructions)

{

if (instructionsPage == instructions\_page01 && isMouseOnArrowR(point))

{

if (!isArrowRShow)

{

CAudio::Instance()->Play(AUDIO\_CHOOSE, false);

}

isArrowRShow = true;

}

else if (instructionsPage == instructions\_page02 && isMouseOnArrowL(point))

{

if (!isArrowLShow)

{

CAudio::Instance()->Play(AUDIO\_CHOOSE, false);

}

isArrowLShow = true;

}

else if (instructionsPage == instructions\_page02 && isMouseOnArrowR(point))

{

if (!isArrowRShow)

{

CAudio::Instance()->Play(AUDIO\_CHOOSE, false);

}

isArrowRShow = true;

}

else if (instructionsPage == instructions\_page03 && isMouseOnArrowL(point))

{

if (!isArrowLShow)

{

CAudio::Instance()->Play(AUDIO\_CHOOSE, false);

}

isArrowLShow = true;

}

else

{

isArrowLShow = false;

isArrowRShow = false;

}

if (point.x >= 1306 && point.y <= 60)

{

if (!isMouseOnX)

{

CAudio::Instance()->Play(AUDIO\_CHOOSE, false);

}

isMouseOnX = true;

}

else

{

isMouseOnX = false;

}

}

}

void CGameStateInit::isChoiceChange()

{

if (lastChoice != choice && choice != 0)

{

CAudio::Instance()->Play(AUDIO\_CHOOSE, false);

lastChoice = choice;

}

}

void CGameStateInit::OnMove()

{

isChoiceChange();

}

void CGameStateInit::OnShow()

{

if (stage == stage\_start\_menu)

{

startMenu.SetTopLeft(0, 0);

startMenu.ShowBitmap();

if (onChoice)

{

startMenuChoice.SetTopLeft(603, yChoice);

startMenuChoice.ShowBitmap();

}

}

else if (stage == stage\_staff)

{

staff.SetTopLeft(0, 0);

staff.ShowBitmap();

if (isMouseOnX)

{

whiteX.SetTopLeft(1306, 0);

whiteX.ShowBitmap();

}

}

else if (stage == stage\_instructions)

{

instructionsPageShow();

if (isArrowLShow)

{

arrowL.SetTopLeft(30, 350);

arrowL.ShowBitmap();

}

if (isArrowRShow)

{

arrowR.SetTopLeft(1278, 350);

arrowR.ShowBitmap();

}

if (isMouseOnX)

{

whiteX.SetTopLeft(1306, 0);

whiteX.ShowBitmap();

}

}

}

void CGameStateInit::instructionsPageShow()

{

if (instructionsPage == instructions\_page01)

{

instructionsPage01.SetTopLeft(0, 0);

instructionsPage01.ShowBitmap();

}

else if (instructionsPage == instructions\_page02)

{

instructionsPage02.SetTopLeft(0, 0);

instructionsPage02.ShowBitmap();

}

else if (instructionsPage == instructions\_page03)

{

instructionsPage03.SetTopLeft(0, 0);

instructionsPage03.ShowBitmap();

}

else if (instructionsPage == instructions\_page\_cheat)

{

instructionsPageC.SetTopLeft(0, 0);

instructionsPageC.ShowBitmap();

}

}

bool CGameStateInit::isMouseOnArrowR(CPoint point)

{

if (point.x >= 1278 && point.x <= 1336 && point.y >= 350 && point.y <= 418)

{

return true;

}

return false;

}

bool CGameStateInit::isMouseOnArrowL(CPoint point)

{

if (point.x >= 30 && point.x <= 88 && point.y >= 350 && point.y <= 418)

{

return true;

}

return false;

}

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

// 這個class為遊戲的結束狀態(Game Over)

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

CGameStateOver::CGameStateOver(CGame\* g)

: CGameState(g)

{

}

void CGameStateOver::OnMove()

{

counter--;

if (counter < 0)

{

CAudio::Instance()->Stop(AUDIO\_LOSE);

CAudio::Instance()->Play(AUDIO\_STARTMENU, true);

GotoGameState(GAME\_STATE\_INIT);

}

if (!isBgmPlayed)

{

CAudio::Instance()->Play(AUDIO\_LOSE, true);

isBgmPlayed = true;

}

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 5; // 5 seconds

isBgmPlayed = false;

}

void CGameStateOver::OnInit()

{

gameOver.LoadBitmap(".\\res\\game\_over.bmp");

CAudio::Instance()->Load(AUDIO\_LOSE, "sounds\\lose.mp3");

ShowInitProgress(100);

}

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

{

if (point.x > 340 && point.x < 1085 && point.y > 275 && point.y < 410)

{

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

}

if (point.x > 440 && point.x < 975 && point.y > 410 && point.y < 500)

{

CAudio::Instance()->Stop(AUDIO\_LOSE);

CAudio::Instance()->Play(AUDIO\_STARTMENU, true);

GotoGameState(GAME\_STATE\_INIT);

}

}

void CGameStateOver::OnShow()

{

gameOver.SetTopLeft(0, 0);

gameOver.ShowBitmap();

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 255));

char str[80]; // Demo 數字對字串的轉換

sprintf(str, "Please click \"GAME OVER\" to close the game, or \"PLAY AGAIN\" to restart! (%d)", counter / 30);

pDC->TextOut(680, 730, str);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

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

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

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

CGameStateRun::CGameStateRun(CGame\* g)

: CGameState(g)

{

monsterS2.push\_back(new MonsterCactus(1070, 325, &character));

monsterS2.push\_back(new MonsterCactus(1330, 325, &character));

monsterS2.push\_back(new MonsterBanana(2650, 400, &character));

monsterS3.push\_back(new MonsterMosquito(550, 170, &character));

monsterS3.push\_back(new MonsterMosquito(610, 210, &character));

monsterS3.push\_back(new MonsterMosquito(660, 150, &character));

monsterS3.push\_back(new MonsterMosquito(730, 200, &character));

monsterS3.push\_back(new MonsterMosquito(800, 200, &character));

monsterS3.push\_back(new MonsterMosquito(900, 180, &character));

monsterS3.push\_back(new MonsterMosquito(980, 230, &character));

monsterS3.push\_back(new MonsterMosquito(1020, 190, &character));

monsterS4.push\_back(new MonsterTree(600, 400, &character));

monsterS4.push\_back(new MonsterTree(350, 350, &character));

monsterS4.push\_back(new MonsterCactus(2530, 400, &character));

monsterS4.push\_back(new MonsterCactus(2610, 400, &character));

monsterS4.push\_back(new MonsterCactusMutation(2690, 400, &character));

monsterS4.push\_back(new MonsterCactus(2770, 400, &character));

monsterS4.push\_back(new MonsterCactus(2850, 400, &character));

monsterS4.push\_back(new MonsterCactus(2930, 400, &character));

monsterS5.push\_back(new MonsterTree(1270, 300, &character));

monsterS5.push\_back(new MonsterTree(1400, 300, &character));

monsterS5.push\_back(new MonsterTree(1520, 300, &character));

monsterS5.push\_back(new MonsterShrimp(1930, 400, &character));

monsterS5.push\_back(new MonsterShrimp(1980, 400, &character));

monsterS5.push\_back(new MonsterShrimp(2360, 400, &character));

monsterS5.push\_back(new MonsterShrimp(2450, 400, &character));

monsterS6.push\_back(new MonsterMosquito(900, 400, &character));

monsterS6.push\_back(new MonsterMosquito(1200, 350, &character));

monsterS6.push\_back(new MonsterMosquito(1500, 460, &character));

monsterS6.push\_back(new MonsterShrimp(1000, 400, &character));

monsterS6.push\_back(new MonsterShrimp(1400, 400, &character));

monsterS6.push\_back(new MonsterShrimp(1550, 400, &character));

monsterS6.push\_back(new MonsterShrimp(1860, 400, &character));

monsterS6.push\_back(new MonsterTree(2300, 400, &character));

monsterS6.push\_back(new MonsterTree(2450, 400, &character));

monsterS6.push\_back(new MonsterTree(2600, 400, &character));

monsterS7.push\_back(new MonsterBoss(900, 280, &character));

}

CGameStateRun::~CGameStateRun()

{

for (vector<Monster\*>::iterator it\_i = monsterS1.begin(); it\_i != monsterS1.end(); ++it\_i)

{

delete\* it\_i;

}

for (vector<Monster\*>::iterator it\_i = monsterS2.begin(); it\_i != monsterS2.end(); ++it\_i)

{

delete\* it\_i;

}

for (vector<Monster\*>::iterator it\_i = monsterS3.begin(); it\_i != monsterS3.end(); ++it\_i)

{

delete\* it\_i;

}

for (vector<Monster\*>::iterator it\_i = monsterS4.begin(); it\_i != monsterS4.end(); ++it\_i)

{

delete\* it\_i;

}

for (vector<Monster\*>::iterator it\_i = monsterS5.begin(); it\_i != monsterS5.end(); ++it\_i)

{

delete\* it\_i;

}

for (vector<Monster\*>::iterator it\_i = monsterS6.begin(); it\_i != monsterS6.end(); ++it\_i)

{

delete\* it\_i;

}

for (vector<Monster\*>::iterator it\_i = monsterS7.begin(); it\_i != monsterS7.end(); ++it\_i)

{

delete\* it\_i;

}

}

void CGameStateRun::OnBeginState()

{

character.Initialize();

mapS1.Initialize();

mapS2.Initialize();

mapS3.Initialize();

mapS4.Initialize();

mapS5.Initialize();

mapS6.Initialize();

bossMap.Initialize();

propsBook.Initialize(&character);

characterStatus.Initialize(&character);

characterStatusCall = false;

haveCalledCharacterStatus = false;

gamePause = false;

gameCompleteFlag = false;

messageShowFlag = false;

currentStage = stage\_1;

lastStage = currentStage;

monsterInitialize();

isStageChanged = true;

isCheatPropsOn = false;

}

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

{

//

// 如果希望修改cursor的樣式，則將下面程式的commment取消即可

//

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

//

// 移動背景圖的座標

//

//if (background.Top() > SIZE\_Y)

// background.SetTopLeft(60 ,-background.Height());

//background.SetTopLeft(background.Left(),background.Top()+1);

if (character.GetCurrentHp() <= 0)

{

stopAllBgm();

GotoGameState(GAME\_STATE\_OVER);

}

if (currentStage != stage\_game\_complete)

{

for (int i = 0; i < (signed)monsterS7.size(); i++)

{

if (monsterS7[i]->GetBossDead())

{

gameCompleteFlag = true;

isStageChanged = true;

currentStage = stage\_game\_complete;

}

}

}

if (currentStage != stage\_game\_complete && gamePause == false)

{

switch (currentStage)

{

case stage\_1:

for (unsigned i = 0; i < monsterS1.size(); i++)

{

monsterS1[i]->OnMove(&mapS1);

}

character.OnMove(&mapS1, &monsterS1);

break;

case stage\_2:

for (unsigned i = 0; i < monsterS2.size(); i++)

{

monsterS2[i]->OnMove(&mapS2);

}

character.OnMove(&mapS2, &monsterS2);

break;

case stage\_3:

for (unsigned i = 0; i < monsterS3.size(); i++)

{

monsterS3[i]->OnMove(&mapS3);

}

character.OnMove(&mapS3, &monsterS3);

break;

case stage\_4:

for (unsigned i = 0; i < monsterS4.size(); i++)

{

monsterS4[i]->OnMove(&mapS4);

}

character.OnMove(&mapS4, &monsterS4);

break;

case stage\_5:

for (unsigned i = 0; i < monsterS5.size(); i++)

{

monsterS5[i]->OnMove(&mapS5);

}

character.OnMove(&mapS5, &monsterS5);

break;

case stage\_6:

for (unsigned i = 0; i < monsterS6.size(); i++)

{

monsterS6[i]->OnMove(&mapS6);

}

character.OnMove(&mapS6, &monsterS6);

break;

case stage\_boss:

for (unsigned i = 0; i < monsterS7.size(); i++)

{

monsterS7[i]->OnMove(&bossMap);

}

character.OnMove(&bossMap, &monsterS7);

break;

default:

break;

}

}

if (isStageChanged)

{

bgmPlayer();

characterStatus.setStage(currentStage);

isStageChanged = false;

}

}

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

{

//

// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人

// 等的不耐煩，遊戲會出現「Loading ...」，顯示Loading的進度。

//

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

//

// 完成部分Loading動作，提高進度

//

ShowInitProgress(50);

// 繼續載入其他資料

character.LoadBitmap();

mapS1.LoadBitmap();

mapS2.LoadBitmap();

mapS3.LoadBitmap();

mapS4.LoadBitmap();

mapS5.LoadBitmap();

mapS6.LoadBitmap();

bossMap.LoadBitmap();

propsBook.LoadBitmap();

characterStatus.LoadBitmap();

lightBulb.LoadBitmap(".\\res\\light\_bulb.bmp", RGB(0, 0, 0));

gameComplete.LoadBitmap(".\\res\\game\_complete.bmp");

for (unsigned i = 0; i < monsterS1.size(); i++)

{

monsterS1[i]->LoadBitmap();

}

for (unsigned i = 0; i < monsterS2.size(); i++)

{

monsterS2[i]->LoadBitmap();

}

for (unsigned i = 0; i < monsterS3.size(); i++)

{

monsterS3[i]->LoadBitmap();

}

for (unsigned i = 0; i < monsterS4.size(); i++)

{

monsterS4[i]->LoadBitmap();

}

for (unsigned i = 0; i < monsterS5.size(); i++)

{

monsterS5[i]->LoadBitmap();

}

for (unsigned i = 0; i < monsterS6.size(); i++)

{

monsterS6[i]->LoadBitmap();

}

for (unsigned i = 0; i < monsterS7.size(); i++)

{

monsterS7[i]->LoadBitmap();

}

CAudio::Instance()->Load(AUDIO\_COOK, "sounds\\cook.mp3");

CAudio::Instance()->Load(AUDIO\_ATTACK\_HU, "sounds\\attack\_hu.mp3");

CAudio::Instance()->Load(AUDIO\_MUSIC\_01, "sounds\\music01.mp3");

CAudio::Instance()->Load(AUDIO\_MUSIC\_02, "sounds\\music02.mp3");

CAudio::Instance()->Load(AUDIO\_MUSIC\_03, "sounds\\music03.mp3");

CAudio::Instance()->Load(AUDIO\_MUSIC\_04, "sounds\\music04.mp3");

CAudio::Instance()->Load(AUDIO\_MUSIC\_05, "sounds\\music05.mp3");

CAudio::Instance()->Load(AUDIO\_MUSIC\_06, "sounds\\music06.mp3");

CAudio::Instance()->Load(AUDIO\_MUSIC\_07, "sounds\\music07.mp3");

CAudio::Instance()->Load(AUDIO\_VICTORY, "sounds\\victory.mp3");

CAudio::Instance()->Load(AUDIO\_RECYCLE\_CAN, "sounds\\recycle\_can.mp3");

CAudio::Instance()->Load(AUDIO\_POT, "sounds\\pot.mp3");

CAudio::Instance()->Load(AUDIO\_ROLL, "sounds\\roll2.mp3");

// 此OnInit動作會接到CGameStaterOver::OnInit()，所以進度還沒到100%

}

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

{

if (!gameCompleteFlag)

{

if (!gamePause)

{

if (nChar == KEY\_A)

character.SetMovingLeft(true);

else if (nChar == KEY\_D)

character.SetMovingRight(true);

else if (nChar == KEY\_W || nChar == KEY\_SPACE)

character.SetMovingUp(true);

else if (nChar == KEY\_S)

character.SetMovingDown();

else if (nChar == KEY\_CTRL)

character.SetRolling(true);

}

if (nChar == KEY\_E)

{

if (character.GetCanGoToNextMap())

{

monsterInitialize();

switch (currentStage)

{

case stage\_1:

mapS2.Initialize();

character.SetXY(mapS2.getStartPosition(), 100);

currentStage = stage\_2;

for (int i = 0; i < (signed)monsterS2.size(); i++)

{

monsterS2[i]->SetIsViolent(true);

}

break;

case stage\_2:

if (character.GetLeftX() < 1500)

{

mapS1.setXY(-2190, -320);

character.SetXY(mapS1.getFinalPosition(), 100);

currentStage = stage\_1;

}

else

{

mapS3.Initialize();

character.SetXY(mapS3.getStartPosition(), 100);

currentStage = stage\_3;

}

break;

case stage\_3:

if (character.GetLeftX() < 500)

{

mapS2.setXY(-2820, -550);

character.SetXY(mapS2.getFinalPosition(), 100);

currentStage = stage\_2;

for (int i = 0; i < (signed)monsterS2.size(); i++)

{

monsterS2[i]->SetIsViolent(true);

}

}

else

{

mapS4.Initialize();

character.SetXY(mapS4.getStartPosition(), 100);

currentStage = stage\_4;

}

break;

case stage\_4:

if (character.GetLeftX() < 500)

{

mapS3.setXY(-3680, -260);

character.SetXY(mapS3.getFinalPosition(), 100);

currentStage = stage\_3;

}

else

{

mapS5.Initialize();

character.SetXY(mapS5.getStartPosition(), 100);

currentStage = stage\_5;

monsterS5[0]->SetIsViolent(true);

monsterS5[1]->SetIsViolent(true);

monsterS5[2]->SetIsViolent(true);

}

break;

case stage\_5:

if (character.GetLeftX() < 700)

{

mapS4.setXY(-2580, -440);

character.SetXY(mapS4.getFinalPosition(), 100);

currentStage = stage\_4;

}

else if (character.GetLeftX() < 2250)

{

bossMap.Initialize();

character.SetXY(bossMap.getStartPosition(), 100);

currentStage = stage\_boss;

monsterS7[0]->SetIsViolent(true);

}

else

{

mapS6.Initialize();

character.SetXY(mapS6.getStartPosition(), 100);

currentStage = stage\_6;

for (int i = 0; i < (signed)monsterS6.size(); i++)

{

monsterS6[i]->SetIsViolent(true);

}

}

break;

case stage\_6:

if (character.GetLeftX() < 700)

{

mapS5.setXY(-2515, -800);

character.SetXY(mapS5.getFinalPosition(), 100);

currentStage = stage\_5;

monsterS5[0]->SetIsViolent(true);

monsterS5[1]->SetIsViolent(true);

monsterS5[2]->SetIsViolent(true);

}

else

{

bossMap.Initialize();

character.SetXY(bossMap.getStartPosition(), 100);

currentStage = stage\_boss;

}

break;

default:

break;

}

isStageChanged = true;

character.SetCanGoToNextMap(false);

}

else if (currentStage == stage\_props)

{

currentStage = lastStage;

if (!haveCalledCharacterStatus)

{

gamePause = false;

}

else

{

characterStatusCall = true;

}

}

else if (character.GetCanCraft())

{

lastStage = currentStage;

currentStage = stage\_props;

haveCalledCharacterStatus = characterStatusCall;

characterStatusCall = false;

gamePause = true;

}

else if (isPlayingStage() && GetCurrentMap()->getCanShowMessage() && !messageShowFlag)

{

gamePause = true;

messageShowFlag = true;

}

else if (messageShowFlag)

{

messageShowFlag = false;

gamePause = false;

GetCurrentMap()->InitializeMessage();

if (GetCurrentMap()->getMapName() == "MapS6")

{

mapS6.setHavePlayedAutoMessage(true);

mapS6.setBossAssistantExist(false);

}

}

}

}

if (nChar == KEY\_TAB)

{

if (characterStatusCall == true)

{

characterStatusCall = false;

haveCalledCharacterStatus = false;

gamePause = false;

}

else

{

if (currentStage != stage\_props)

{

characterStatusCall = true;

haveCalledCharacterStatus = true;

gamePause = true;

}

}

}

if (nChar == KEY\_Q)

{

if (character.GetCharacterStage() == 0)

character.characterHasSword(true);

else if (character.GetCharacterStage() == 1)

character.characterHasSword(false);

}

if (nChar == KEY\_R)

character.restoreCurrentHp(50);

if (nChar == KEY\_T)

character.addAttackDamage(10);

if (nChar == KEY\_Y)

character.SetLightBulbNum(1000);

if (nChar == KEY\_U)

character.SetAllSourceNumToHundred();

if (nChar == KEY\_H)

{

currentStage = stage\_game\_complete;

isStageChanged = true;

}

if (nChar == KEY\_P)

{

if (!isCheatPropsOn)

{

propsBook.getAllProps();

isCheatPropsOn = true;

}

else

{

propsBook.loseAllProps();

isCheatPropsOn = false;

}

}

if (nChar == KEY\_G)

{

character.SetCurrentHp(0);

stopAllBgm();

}

if (nChar == KEY\_1)

{

monsterInitialize();

gamePause = false;

characterStatusCall = false;

mapS1.Initialize();

character.SetXY(mapS1.getStartPosition(), 100);

currentStage = stage\_1;

isStageChanged = true;

}

if (nChar == KEY\_2)

{

monsterInitialize();

gamePause = false;

characterStatusCall = false;

mapS2.Initialize();

character.SetXY(mapS2.getStartPosition(), 100);

currentStage = stage\_2;

isStageChanged = true;

for (int i = 0; i < (signed)monsterS2.size(); i++)

{

monsterS2[i]->SetIsViolent(true);

}

}

if (nChar == KEY\_3)

{

monsterInitialize();

gamePause = false;

characterStatusCall = false;

mapS3.Initialize();

character.SetXY(mapS3.getStartPosition(), 100);

currentStage = stage\_3;

isStageChanged = true;

}

if (nChar == KEY\_4)

{

monsterInitialize();

gamePause = false;

characterStatusCall = false;

mapS4.Initialize();

character.SetXY(mapS4.getStartPosition(), 100);

currentStage = stage\_4;

isStageChanged = true;

}

if (nChar == KEY\_5)

{

monsterInitialize();

gamePause = false;

characterStatusCall = false;

mapS5.Initialize();

character.SetXY(mapS5.getStartPosition(), 100);

currentStage = stage\_5;

isStageChanged = true;

monsterS5[0]->SetIsViolent(true);

monsterS5[1]->SetIsViolent(true);

monsterS5[2]->SetIsViolent(true);

}

if (nChar == KEY\_6)

{

monsterInitialize();

gamePause = false;

characterStatusCall = false;

mapS6.Initialize();

character.SetXY(mapS6.getStartPosition(), 100);

currentStage = stage\_6;

isStageChanged = true;

for (int i = 0; i < (signed)monsterS6.size(); i++)

{

monsterS6[i]->SetIsViolent(true);

}

}

if (nChar == KEY\_7)

{

monsterInitialize();

gamePause = false;

characterStatusCall = false;

bossMap.Initialize();

character.SetXY(bossMap.getStartPosition(), 100);

currentStage = stage\_boss;

isStageChanged = true;

}

if (nChar == KEY\_ESC)

{

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

}

}

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

{

if (nChar == KEY\_A)

character.SetMovingLeft(false);

if (nChar == KEY\_D)

character.SetMovingRight(false);

if (nChar == KEY\_W || nChar == KEY\_SPACE)

character.SetMovingUp(false);

}

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

{

if (currentStage == stage\_props)

{

if (point.x > 1253 && point.y > 57 && point.x < 1327 && point.y < 117)

{

currentStage = lastStage;

if (!haveCalledCharacterStatus)

{

gamePause = false;

}

else

{

characterStatusCall = true;

}

}

else

{

propsBook.setCase(point);

}

}

else if (currentStage == stage\_game\_complete)

{

if (point.x > 80 && point.y > 170 && point.x < 700 && point.y < 610)

{

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

}

if (point.x > 840 && point.y > 215 && point.x < 1210 && point.y < 585)

{

CAudio::Instance()->Stop(AUDIO\_VICTORY);

CAudio::Instance()->Play(AUDIO\_STARTMENU, true);

GotoGameState(GAME\_STATE\_INIT);

}

}

else

{

if (gamePause == false && !character.GetIsAttacking() && !character.GetIsRolling())

{

if (currentStage == stage\_1)

{

character.attack(&monsterS1);

}

else if (currentStage == stage\_2)

{

character.attack(&monsterS2);

}

else if (currentStage == stage\_3)

{

character.attack(&monsterS3);

}

else if (currentStage == stage\_4)

{

character.attack(&monsterS4);

}

else if (currentStage == stage\_5)

{

character.attack(&monsterS5);

}

else if (currentStage == stage\_6)

{

character.attack(&monsterS6);

}

else if (currentStage == stage\_boss)

{

character.attack(&monsterS7);

}

}

}

if (isPlayingStage() && messageShowFlag && !GetCurrentMap()->getMessageEndFlag())

{

GetCurrentMap()->setMessageCounterToNext();

}

else if (isPlayingStage() && messageShowFlag && GetCurrentMap()->getMessageEndFlag())

{

messageShowFlag = false;

gamePause = false;

GetCurrentMap()->InitializeMessage();

if (GetCurrentMap()->getMapName() == "MapS6")

{

mapS6.setHavePlayedAutoMessage(true);

mapS6.setBossAssistantExist(false);

}

}

if (characterStatusCall == true && gameCompleteFlag == false)

{

if (point.x > 1030 && point.y > 133 && point.x < 1082 && point.y < 189)

{

characterStatusCall = false;

gamePause = false;

}

}

}

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

{

}

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

{

if (!gamePause)

{

character.SetFacingDirection(point.x); // 滑鼠位置改變角色方向

mousePosition = 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::OnShow()

{

//

// 注意：Show裡面千萬不要移動任何物件的座標，移動座標的工作應由Move做才對，

// 否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語

// 說，Move負責MVC中的Model，Show負責View，而View不應更動Model。

//

if (isPlayingStage() && GetCurrentMap()->getAutoShowMessage() && !messageShowFlag)

{

gamePause = true;

messageShowFlag = true;

}

switch (currentStage)

{

case stage\_1:

mapS1.onShow();

character.OnShow(&mapS1);

for (unsigned i = 0; i < monsterS1.size(); i++)

{

monsterS1[i]->OnShow(&mapS1);

}

LightBulbOnShow();

if (messageShowFlag)

{

mapS1.messageOnShow();

}

break;

case stage\_2:

mapS2.onShow();

character.OnShow(&mapS2);

for (unsigned i = 0; i < monsterS2.size(); i++)

{

monsterS2[i]->OnShow(&mapS2);

}

LightBulbOnShow();

break;

case stage\_3:

mapS3.onShow();

character.OnShow(&mapS3);

for (unsigned i = 0; i < monsterS3.size(); i++)

{

monsterS3[i]->OnShow(&mapS3);

}

if (messageShowFlag)

{

mapS3.messageOnShow();

}

LightBulbOnShow();

break;

case stage\_4:

mapS4.onShow();

character.OnShow(&mapS4);

for (unsigned i = 0; i < monsterS4.size(); i++)

{

monsterS4[i]->OnShow(&mapS4);

}

LightBulbOnShow();

break;

case stage\_5:

mapS5.onShow();

character.OnShow(&mapS5);

for (unsigned i = 0; i < monsterS5.size(); i++)

{

monsterS5[i]->OnShow(&mapS5);

}

if (messageShowFlag)

{

mapS5.messageOnShow();

}

LightBulbOnShow();

break;

case stage\_6:

mapS6.onShow();

character.OnShow(&mapS6);

for (unsigned i = 0; i < monsterS6.size(); i++)

{

monsterS6[i]->OnShow(&mapS6);

}

if (messageShowFlag)

{

mapS6.messageOnShow();

}

LightBulbOnShow();

break;

case stage\_boss:

bossMap.onShow();

character.OnShow(&bossMap);

for (unsigned i = 0; i < monsterS7.size(); i++)

{

monsterS7[i]->OnShow(&bossMap);

}

LightBulbOnShow();

break;

case stage\_props:

propsBook.onShow();

break;

case stage\_game\_complete:

gameComplete.ShowBitmap();

gameCompleteNoteShow();

break;

default:

break;

}

if (characterStatusCall)

{

characterStatus.onShow();

}

}

void CGameStateRun::LightBulbOnShow()

{

lightBulb.SetTopLeft(18, 80);

lightBulb.ShowBitmap();

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(200, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkMode(TRANSPARENT);

pDC->SetTextColor(RGB(255, 255, 255));

char position[100]; // Demo 數字對字串的轉換

sprintf(position, "x %d", character.GetLightBulbNum());

pDC->TextOut(57, 87, position);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

void CGameStateRun::gameCompleteNoteShow()

{

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(160, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 255));

char str[80]; // Demo 數字對字串的轉換

sprintf(str, "Please click \"GAME COMPLETE\" to close the game, or \"PLAY AGAIN\" to restart!");

pDC->TextOut(665, 730, str);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

void CGameStateRun::bgmPlayer()

{

stopAllBgm();

if (currentStage == stage\_1)

{

CAudio::Instance()->Play(AUDIO\_MUSIC\_01, true);

}

else if (currentStage == stage\_2)

{

CAudio::Instance()->Play(AUDIO\_MUSIC\_02, true);

}

else if (currentStage == stage\_3)

{

CAudio::Instance()->Play(AUDIO\_MUSIC\_03, true);

}

else if (currentStage == stage\_4)

{

CAudio::Instance()->Play(AUDIO\_MUSIC\_04, true);

}

else if (currentStage == stage\_5)

{

CAudio::Instance()->Play(AUDIO\_MUSIC\_05, true);

}

else if (currentStage == stage\_6)

{

CAudio::Instance()->Play(AUDIO\_MUSIC\_06, true);

}

else if (currentStage == stage\_boss)

{

CAudio::Instance()->Play(AUDIO\_MUSIC\_07, true);

}

else if (currentStage == stage\_game\_complete)

{

CAudio::Instance()->Play(AUDIO\_VICTORY, true);

}

}

void CGameStateRun::stopAllBgm()

{

CAudio::Instance()->Stop(AUDIO\_MUSIC\_01);

CAudio::Instance()->Stop(AUDIO\_MUSIC\_02);

CAudio::Instance()->Stop(AUDIO\_MUSIC\_03);

CAudio::Instance()->Stop(AUDIO\_MUSIC\_04);

CAudio::Instance()->Stop(AUDIO\_MUSIC\_05);

CAudio::Instance()->Stop(AUDIO\_MUSIC\_06);

CAudio::Instance()->Stop(AUDIO\_MUSIC\_07);

CAudio::Instance()->Stop(AUDIO\_VICTORY);

}

bool CGameStateRun::isPlayingStage()

{

if (currentStage != stage\_props && currentStage != stage\_game\_complete)

{

return true;

}

return false;

}

Map\* CGameStateRun::GetCurrentMap()

{

if (currentStage == stage\_1)

{

return &mapS1;

}

else if (currentStage == stage\_2)

{

return &mapS2;

}

else if (currentStage == stage\_3)

{

return &mapS3;

}

else if (currentStage == stage\_4)

{

return &mapS4;

}

else if (currentStage == stage\_5)

{

return &mapS5;

}

else if (currentStage == stage\_6)

{

return &mapS6;

}

else if (currentStage == stage\_boss)

{

return &bossMap;

}

else

{

throw("GetCurrentMap() didn't have return");

}

}

void CGameStateRun::monsterInitialize()

{

for (unsigned i = 0; i < monsterS1.size(); i++)

{

monsterS1[i]->Initialize();

}

for (unsigned i = 0; i < monsterS2.size(); i++)

{

monsterS2[i]->Initialize();

}

for (unsigned i = 0; i < monsterS3.size(); i++)

{

monsterS3[i]->Initialize();

}

for (unsigned i = 0; i < monsterS4.size(); i++)

{

monsterS4[i]->Initialize();

}

for (unsigned i = 0; i < monsterS5.size(); i++)

{

monsterS5[i]->Initialize();

}

for (unsigned i = 0; i < monsterS6.size(); i++)

{

monsterS6[i]->Initialize();

}

for (unsigned i = 0; i < monsterS7.size(); i++)

{

monsterS7[i]->Initialize();

}

}

}

===============================

Character.h

===============================

#ifndef \_\_CHARACTER\_\_

#define \_\_CHARACTER\_\_

#include "Map.h"

#include "Timer.h"

#include "Util.h"

namespace game\_framework

{

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

// 這個class為遊戲的角色人物物件

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

class Source;

class SourceStorage;

class Prop;

class PropStorage;

class Monster;

class Character

{

public:

Character();

~Character();

Map\* GetMap(); // 地圖

void Initialize(); // 設定初始值

void LoadBitmap(); // 載入圖形

void OnMove(Map\* m, vector<Monster\*>\* monsters); // 移動

void OnShow(Map\* m); // 將圖形貼到畫面

void BloodShow();

void ScreenCXY(Map\* m);

/\*map\*/

void shouldShowMagnifier(Map\* m);

void shouldShowMessageIconFlag(Map\* m);

void SetMapCanShowMessage(Map\* m);

void SetMapAutoShowMessage(Map\* m);

int GetLeftX(); // 左上角 x 座標

int GetTopY(); // 左上角 y 座標

int GetRightX(); // 右下角 x 座標

int GetBottomY(); // 右下角 y 座標

int GetCharacterStage();

bool GetIsMovingLeft(); // 回傳是否正在往左移動

bool GetIsMovingRight(); // 回傳是否正在往右移動

bool GetIsMovingUp(); // 回傳是否正在往上移動

bool GetIsOnTheFloor(); // 回傳是否正在地面

bool GetIsRising(); // 回傳是否正在上升

bool GetCanGoToNextMap(); // 回傳是否可以前往下一關

bool GetCanCraft(); // 回傳是否能合成

int GetMaxHp(); // 取得最大血量

int GetCurrentHp(); // 取得目前血量

bool GetIsInvincible(); //回傳是否為無敵狀態

void SetMap(Map\* m);

void SetMovingDown(); // 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 SetFacingDirection(int mouseX); // 設定面對方向

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

void SetXY(int x, int y); // 設定左上角座標

void SetCanGoToNextMap(bool flag); // 設定是否能前往下一關

void SetCanCraft(bool flag); // 設定是否能合成

void SetCurrentHp(int x); // 設定目前血量

void SetIsInvincible(bool flag); //設定是否為無敵狀態

void restoreCurrentHp(int n); // 恢復目前血量

void lossCurrentHp(int n); // 損血

void hpLimit();

bool isIntersect(int lX, int rX, int tY, int bY); //看物體是否和角色重疊

void characterHasSword(bool flag);

/\*行為\*/

//判定是否可以左右移動

bool CanMovingLeft(Map\* m, vector<Monster\*>\* monsters);

bool CanMovingRight(Map\* m, vector<Monster\*>\* monsters);

//速度

void SetSpeed(int x);

int GetSpeed();

//翻滾

void SetRolling(bool flag); // 設定是否翻滾

bool GetIsRolling(); // 回傳是否正在翻滾

void Rolling(Map\* m, bool flag, vector<Monster\*>\* monsters); // 翻滾動作

//攻擊

void SetAttacking(bool flag); // 設定是否攻擊

bool GetIsAttacking(); // 回傳是否正在攻擊

void SetAttackDamage(int x); // 設定攻擊力

int GetAttackDamage(); // 取得攻擊力

void addAttackDamage(int x); // 提升攻擊力

void SetAttackRange(int x);

int GetAttackRange();

void attack(vector<Monster\*>\* monsters);

void attackOnMove();

//受到攻擊

void SetIsAttackedFromRight(bool flag); // 設定攻擊是否來自右方

bool GetIsAttackedFromRight(); // 回傳攻擊是否來自右方

void SetIsAttackedFromLeft(bool flag); // 設定攻擊是否來自左方

bool GetIsAttackedFromLeft(); // 回傳攻擊是否來自左方

void SetIsAttackedFromBottom(bool flag); // 設定攻擊是否來自下方

bool GetIsAttackedFromBottom(); // 回傳攻擊是否來自下方

void isAttackedEffectCaculation();

void isAttackedEffectOnShow();

/\*餐點能力\*/

SourceStorage\* GetSourceStorage(); // 回傳素材儲存空間

PropStorage\* GetPropStorage(); // 回傳道具儲存空間

void EatMosquitoJump(bool flag); // 吃下二段跳能力

bool CanDoubleJump(); // 回傳是否可以二段跳

void SetDoubleJump(bool flag); // 設定是否可以二段跳

void EatGrassFast(bool flag); // 吃下移動速度提升能力

void ChangeSpeed(double m); // 提升移動速度(倍率)

void EatShrimpBlood(bool flag); // 吃下血量提升能力

void addMaxHp(int blood); // 提升血量上限

void EatGuavaJuiceBlood(bool flag); // 吃下每十秒回三滴血能力

void healBloodEveryTenSeconds(); // 每十秒回三滴血

void EatBananaAttack(bool flag);

void EatShrimpAttack(bool flag);

void SetIsMosquitoJump(bool flag);

bool GetIsMosquitoJump();

void SetIsGrassFast(bool flag);

bool GetIsGrassFast();

void SetIsShrimpBlood(bool flag);

bool GetIsShrimpBlood();

void SetIsGuavaJuiceBlood(bool flag);

bool GetIsGuavaJuiceBlood();

void SetIsBananaAttack(bool flag);

bool GetIsBananaAttack();

void SetIsShrimpAttack(bool flag);

bool GetIsShrimpAttack();

void SetLightBulbNum(int num);

int GetLightBulbNum();

void AddLightBulb(int num);

void ConsumeLightBulb(int num);

void SetAllSourceNumToHundred();

protected:

bool isAttackSuccessfullyL(int range, Monster\* monster);

bool isAttackSuccessfullyR(int range, Monster\* monster);

CMovingBitmap bloodFrame; // 角色血量框

CMovingBitmap characterBlood[10]; // 角色血量

CAnimation leftRolling; // 左滾動畫

CAnimation rightRolling; // 右滾動畫

CMovingBitmap standLeft[2];

CMovingBitmap standRight[2];

CAnimation walkingLeft[2];

CAnimation walkingRight[2];

CAnimation leftJump[2];

CAnimation rightJump[2];

CAnimation leftAttacking[2];

CAnimation rightAttacking[2];

Map\* currentMap;

int characterX, characterY;

int screenCX, screenCY;

int characterW, characterH;

bool mapScreenCanMoving;

bool canGoToNextMap;

bool canCraft;

int STEP\_SIZE; // 移動速度

int BORDER;

int monsterTop;

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 isRolling; // 是否正在翻滾

Timer rollingTimer;

bool facingLR; // 面向 左:0, 右:1

bool isOnTheFloor; // 是否位於地面

bool isRising; // 是否正在上升

bool isAttacking; // 是否正在攻擊

int attackRange;

bool isAttackedFromRight;

bool isAttackedFromLeft;

bool isAttackedFromBottom;

Timer isAttackedTimer;

bool isAttacked; // 是否受到攻擊

Timer sparkleEffectTimer;

bool isSparkleEffectTimerStart;

bool isSparkleEffectShow;

CMovingBitmap black;

int velocity;

int rolling\_time;

Action action;

Timer healBloodTimer;

bool healBlood;

Timer invincibleTimer;

bool isInvincible;

int currentHp;

int maxHp;

int attackDamage;

bool doubleJump;

bool DJtemp;

/\*餐點\*/

bool isMosquitoJump;

bool isGrassFast;

bool isShrimpBlood;

bool isGuavaJuiceBlood;

bool isBananaAttack;

Timer ShrimpAttackTimer;

bool shrimpAttack;

bool isShrimpAttack;

int characterStage;

int lightBulbNum;

SourceStorage\* sourceStorage;

PropStorage\* propStorage;

};

}

#endif

===============================

Character.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Map.h"

#include "Source.h"

#include "SourceStorage.h"

#include "Prop.h"

#include "PropStorage.h"

#include "Monster.h"

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

// 這個class為遊戲的角色人物物件

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

namespace game\_framework

{

Character::Character()

{

sourceStorage = new SourceStorage();

propStorage = new PropStorage();

}

Character::~Character()

{

delete sourceStorage;

delete propStorage;

}

void Character::LoadBitmap()

{

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

bloodFrame.LoadBitmap(IDB\_CHARACTERBLOODFRAME, RGB(0, 0, 0));

characterBlood[0].LoadBitmap(IDB\_CHARACTERBLOOD01, RGB(255, 255, 255));

characterBlood[1].LoadBitmap(IDB\_CHARACTERBLOOD02, RGB(255, 255, 255));

characterBlood[2].LoadBitmap(IDB\_CHARACTERBLOOD03, RGB(255, 255, 255));

characterBlood[3].LoadBitmap(IDB\_CHARACTERBLOOD04, RGB(255, 255, 255));

characterBlood[4].LoadBitmap(IDB\_CHARACTERBLOOD05, RGB(255, 255, 255));

characterBlood[5].LoadBitmap(IDB\_CHARACTERBLOOD06, RGB(255, 255, 255));

characterBlood[6].LoadBitmap(IDB\_CHARACTERBLOOD07, RGB(255, 255, 255));

characterBlood[7].LoadBitmap(IDB\_CHARACTERBLOOD08, RGB(255, 255, 255));

characterBlood[8].LoadBitmap(IDB\_CHARACTERBLOOD09, RGB(255, 255, 255));

characterBlood[9].LoadBitmap(IDB\_CHARACTERBLOOD10, RGB(255, 255, 255));

standLeft[0].LoadBitmap(IDB\_HEROLEFTSTAND\_S, RGB(0, 0, 0)); //向左站

standRight[0].LoadBitmap(IDB\_HERORIGHTSTAND\_S, RGB(0, 0, 0)); //向右站

//向左走動畫

vector<int> walkingLeftAnimation\_S = { IDB\_HEROLEFTWALK1\_S, IDB\_HEROLEFTWALK2\_S, IDB\_HEROLEFTWALK3\_S };

for (int i = 0; i < 3; i++) // 載入動畫

walkingLeft[0].AddBitmap(walkingLeftAnimation\_S[i], RGB(0, 0, 0));

//向右走動畫

vector<int> walkingRightAnimation\_S = { IDB\_HERORIGHTWALK1\_S, IDB\_HERORIGHTWALK2\_S, IDB\_HERORIGHTWALK3\_S };

for (int i = 0; i < 3; i++) // 載入動畫

walkingRight[0].AddBitmap(walkingRightAnimation\_S[i], RGB(0, 0, 0));

leftJump[0].AddBitmap(IDB\_HEROLEFTJUMP\_S, RGB(0, 0, 0));

for (int i = 0; i < 3; i++)

leftJump[0].AddBitmap(IDB\_HEROLEFTSTAND\_S, RGB(0, 0, 0));

rightJump[0].AddBitmap(IDB\_HERORIGHTJUMP\_S, RGB(0, 0, 0));

for (int i = 0; i < 3; i++)

rightJump[0].AddBitmap(IDB\_HERORIGHTSTAND\_S, RGB(0, 0, 0));

vector<int> rollingLeftAnimation = { IDB\_HEROLEFTROLL1, IDB\_HEROLEFTROLL2, IDB\_HEROLEFTROLL3, IDB\_HEROLEFTROLL4, IDB\_HEROLEFTROLL5, IDB\_HEROLEFTROLL6, IDB\_HEROLEFTROLL7, IDB\_HEROLEFTROLL8 };

for (int i = 0; i < 8; i++)

leftRolling.AddBitmap(rollingLeftAnimation[i], RGB(0, 0, 0));

vector<int> rollingRightAnimation = { IDB\_HERORIGHTROLL1, IDB\_HERORIGHTROLL2, IDB\_HERORIGHTROLL3, IDB\_HERORIGHTROLL4, IDB\_HERORIGHTROLL5, IDB\_HERORIGHTROLL6, IDB\_HERORIGHTROLL7, IDB\_HERORIGHTROLL8 };

for (int i = 0; i < 8; i++)

rightRolling.AddBitmap(rollingRightAnimation[i], RGB(0, 0, 0));

vector<int> attackingLeftAnimation\_S = { IDB\_HEROLEFTATTACK1\_S, IDB\_HEROLEFTATTACK2\_S, IDB\_HEROLEFTATTACK3\_S, IDB\_HEROLEFTATTACK4\_S };

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

leftAttacking[0].AddBitmap(attackingLeftAnimation\_S[i], RGB(0, 0, 0));

vector<int> attackingRightAnimation\_S = { IDB\_HERORIGHTATTACK1\_S, IDB\_HERORIGHTATTACK2\_S, IDB\_HERORIGHTATTACK3\_S, IDB\_HERORIGHTATTACK4\_S };

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

rightAttacking[0].AddBitmap(attackingRightAnimation\_S[i], RGB(0, 0, 0));

//has sword

standLeft[1].LoadBitmap(IDB\_HEROLEFTSTAND\_G, RGB(0, 0, 0)); //向左站

standRight[1].LoadBitmap(IDB\_HERORIGHTSTAND\_G, RGB(0, 0, 0)); //向右站

//向左走動畫

vector<int> walkingLeftAnimation\_G = { IDB\_HEROLEFTWALK1\_G, IDB\_HEROLEFTWALK2\_G, IDB\_HEROLEFTWALK3\_G };

for (int i = 0; i < 3; i++) // 載入動畫

walkingLeft[1].AddBitmap(walkingLeftAnimation\_G[i], RGB(0, 0, 0));

//向右走動畫

vector<int> walkingRightAnimation\_G = { IDB\_HERORIGHTWALK1\_G, IDB\_HERORIGHTWALK2\_G, IDB\_HERORIGHTWALK3\_G };

for (int i = 0; i < 3; i++) // 載入動畫

walkingRight[1].AddBitmap(walkingRightAnimation\_G[i], RGB(0, 0, 0));

leftJump[1].AddBitmap(IDB\_HEROLEFTJUMP\_G, RGB(0, 0, 0));

for (int i = 0; i < 3; i++)

leftJump[1].AddBitmap(IDB\_HEROLEFTSTAND\_G, RGB(0, 0, 0));

rightJump[1].AddBitmap(IDB\_HERORIGHTJUMP\_G, RGB(0, 0, 0));

for (int i = 0; i < 3; i++)

rightJump[1].AddBitmap(IDB\_HERORIGHTSTAND\_G, RGB(0, 0, 0));

leftAttacking[1].AddBitmap(".\\res\\g\_attack01\_left.bmp", RGB(0, 0, 0));

leftAttacking[1].AddBitmap(".\\res\\g\_attack02\_left.bmp", RGB(0, 0, 0));

leftAttacking[1].AddBitmap(".\\res\\g\_attack03\_left.bmp", RGB(0, 0, 0));

leftAttacking[1].AddBitmap(".\\res\\g\_attack04\_left.bmp", RGB(0, 0, 0));

leftAttacking[1].AddBitmap(".\\res\\g\_attack05\_left.bmp", RGB(0, 0, 0));

leftAttacking[1].AddBitmap(".\\res\\g\_attack06\_left.bmp", RGB(0, 0, 0));

rightAttacking[1].AddBitmap(".\\res\\g\_attack01\_right.bmp", RGB(0, 0, 0));

rightAttacking[1].AddBitmap(".\\res\\g\_attack02\_right.bmp", RGB(0, 0, 0));

rightAttacking[1].AddBitmap(".\\res\\g\_attack03\_right.bmp", RGB(0, 0, 0));

rightAttacking[1].AddBitmap(".\\res\\g\_attack04\_right.bmp", RGB(0, 0, 0));

rightAttacking[1].AddBitmap(".\\res\\g\_attack05\_right.bmp", RGB(0, 0, 0));

rightAttacking[1].AddBitmap(".\\res\\g\_attack06\_right.bmp", RGB(0, 0, 0));

}

void Character::Initialize()

{

characterH = 120;

characterW = 80;

const int X\_POS = 335; //角色起始X軸

const int Y\_POS = 80; //角色起始Y軸

characterX = X\_POS;

characterY = Y\_POS;

characterStage = 0;

facingLR = 1;

STEP\_SIZE = 10;

BORDER = 5;

isMovingLeft = isMovingRight = isMovingUp = isRising = isRolling = false;

canGoToNextMap = canCraft = false;

maxHp = 80;

currentHp = 80;

rolling\_time = 0;

isAttacking = false;

attackDamage = 10;

attackRange = 60;

isAttackedFromRight = false;

isAttackedFromLeft = false;

isAttackedFromBottom = false;

isAttacked = false;

isSparkleEffectTimerStart = false;

isSparkleEffectShow = false;

isInvincible = false;

doubleJump = false;

DJtemp = doubleJump;

healBlood = false;

isShrimpAttack = false;

isMosquitoJump = isGrassFast = isShrimpBlood = isGuavaJuiceBlood = isBananaAttack = false;

shrimpAttack = false;

currentMap = NULL;

action = walk\_a;

lightBulbNum = 0;

}

void Character::OnMove(Map\* m, vector<Monster\*>\* monsters)

{

int monsterBorder;

bool isOnMonster = false;

if (currentMap == NULL || m->getMapName() != currentMap->getMapName())

{

SetMap(m);

velocity = 30;

}

shouldShowMagnifier(m);

shouldShowMessageIconFlag(m);

SetMapCanShowMessage(m);

SetMapAutoShowMessage(m);

if (m->getMapName() != "MapS1" && m->getMapName() != "MapForBoss")

{

m->setCharacterX((GetLeftX() + GetRightX()) / 2);

m->setCharacterY(characterY);

m->characterFloorAndCeiling();

}

if (m->isPortal(GetLeftX() - BORDER, GetTopY()))

{

if (!canGoToNextMap)

{

m->setPortalOpen(true);

SetCanGoToNextMap(true);

}

}

else

{

m->setPortalOpen(false);

SetCanGoToNextMap(false);

}

if (m->isCraftTable(GetLeftX() - BORDER, GetTopY()))

{

if (!canCraft)

{

CAudio::Instance()->Play(AUDIO\_POT, false);

m->setCraftTableOpen(true);

SetCanCraft(true);

}

}

else

{

m->setCraftTableOpen(false);

SetCanCraft(false);

}

if (m->isTrashCan(GetLeftX() - BORDER, GetTopY()) && !m->getTrashCanOpen())

{

if (!m->getTrashCanOpen())

{

CAudio::Instance()->Play(AUDIO\_RECYCLE\_CAN, false);

m->setTrashCanOpen(true);

currentHp = maxHp;

}

}

if (m->isGetHurtPlace(GetLeftX() - BORDER, GetBottomY()))

{

if (!isInvincible)

{

lossCurrentHp(5);

if (rand() % 2)

{

isAttackedFromLeft = true;

isAttackedFromBottom = true;

}

else

{

isAttackedFromRight = true;

isAttackedFromBottom = true;

}

}

}

if (GetIsRolling())

{

rollingTimer.Start();

if (isMovingLeft)

Rolling(m, 0, monsters);

else if (isMovingRight)

Rolling(m, 1, monsters);

else if (facingLR)

Rolling(m, 1, monsters);

else if (facingLR == 0)

Rolling(m, 0, monsters);

}

else

{

if (CanMovingLeft(m, monsters))

{

if (characterX <= 670 || GetMap()->mapScreenMoving() == false)

{

characterX -= STEP\_SIZE;

}

else

{

if (characterX - STEP\_SIZE < 670)

{

m->addSX(characterX - 670);

characterX = 670;

}

else

{

characterX -= STEP\_SIZE;

m->addSX(STEP\_SIZE);

} //視角移動(王關不用)

}

}

if (CanMovingRight(m, monsters))

{

if (characterX < 670)

{

if (characterX + STEP\_SIZE > 670 && GetMap()->mapScreenMoving() == true)

{

characterX = 670;

}

else

{

characterX += STEP\_SIZE;

}

}

else

{

characterX += STEP\_SIZE;

if (GetMap()->mapScreenMoving() == true)

{

m->addSX(-STEP\_SIZE);

}

}

}

if (GetIsMovingUp() && ((GetBottomY() >= m->getFloor() && velocity == 0) || GetBottomY() == monsterTop))

{

isRising = true;

velocity = 13;

}

if (GetIsMovingUp() && (velocity < 8 || !GetIsRising()) && velocity > 0 && CanDoubleJump())

{

velocity = 10;

isRising = true;

SetDoubleJump(false);

}

if (GetIsRising()) // 上升狀態

{

if (velocity > 0)

{

if (characterY - velocity \* 2 < m->getCeiling())

{

characterY = m->getCeiling();

velocity = 0;

}

else

{

characterY -= velocity \* 2; // 當速度 > 0時，y軸上升(移動velocity個點，velocity的單位為 點/次

velocity--; // 受重力影響，下次的上升速度降低

}

}

else

{

isRising = false; // 當速度 <= 0，上升終止，下次改為下降

velocity = 1; // 下降的初速(velocity)為1

}

}

else // 下降狀態

{

if (GetBottomY() < m->getFloor()) // 當y座標還沒碰到地板

{

if (characterY + velocity \* 3 < m->getFloor() - 120)

{

for (unsigned int i = 0; i < monsters->size(); i++)

{

monsterBorder = monsters->at(i)->GetBorder();

if (GetRightX() > monsters->at(i)->GetLeftX() + monsterBorder && GetLeftX() < monsters->at(i)->GetRightX() - monsterBorder && GetBottomY() + velocity \* 3 >= monsters->at(i)->GetTopY() + monsterBorder && monsters->at(i)->isAlive())

{

if (characterY <= monsters->at(i)->GetTopY() + monsterBorder - 100 && monsters->at(i)->GetCanStandOn())

{

characterY = monsters->at(i)->GetTopY() + monsterBorder - 120;

monsterTop = monsters->at(i)->GetTopY() + monsterBorder;

isOnMonster = true;

}

break;

}

}

if (!isOnMonster)

characterY += velocity \* 3;

}

else

characterY = m->getFloor() - 119;

if (velocity < 6)

velocity++;

}

else

{

monsterTop = 0;

characterY = m->getFloor() - 120; // 當y座標低於地板，更正為地板上

velocity = 0;

SetDoubleJump(DJtemp);

}

}

if (isAttackedFromRight) //還要判定是否能移動

{

for (int i = 0; i < 50; i++)

{

if (m->isEmpty(GetLeftX() - 1 - BORDER, GetTopY()) && m->isEmpty(GetLeftX() - 1 - BORDER, GetBottomY() - BORDER))

{

if (characterX > 670 && m->mapScreenMoving())

{

m->addSX(1);

}

characterX -= 1;

}

}

isAttackedFromRight = false;

}

if (isAttackedFromLeft)

{

for (int i = 0; i < 50; i++)

{

if (m->isEmpty(GetRightX() + 1 + BORDER, GetTopY()) && m->isEmpty(GetRightX() + 1 + BORDER, GetBottomY() - BORDER))

{

if (characterX > 670 && m->mapScreenMoving())

{

m->addSX(-1);

}

characterX += 1;

}

}

isAttackedFromLeft = false;

}

if (isAttackedFromBottom)

{

for (int i = 0; i < 50; i++)

{

characterY -= 1;

}

isAttackedFromBottom = false;

}

}

if (isInvincible)

{

invincibleTimer.CaculateTimeForFalse(&isInvincible, 1);

}

if (isShrimpAttack && !shrimpAttack)

{

ShrimpAttackTimer.CaculateTimeForTrue(&shrimpAttack, 3);

}

if (healBlood)

{

healBloodEveryTenSeconds();

}

isAttackedEffectCaculation();

for (int i = 0; i < 2; i++)

{

walkingLeft[i].OnMove();

walkingRight[i].OnMove();

leftJump[i].OnMove();

rightJump[i].OnMove();

}

leftRolling.OnMove();

rightRolling.OnMove();

if (isAttacking)

{

attackOnMove();

}

}

void Character::OnShow(Map\* m)

{

hpLimit();

BloodShow();

ScreenCXY(m);

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow();

}

else

{

if (facingLR == 0)

{

if (isRolling)//action == roll\_a)

{

CAudio::Instance()->Play(AUDIO\_ROLL, false);

leftRolling.SetTopLeft(screenCX - 5, characterY + 10 + m->getYMovement());

leftRolling.OnShow();

leftRolling.SetDelayCount(1);

if (leftRolling.IsFinalBitmap())

action = walk\_a;

}

else if (isAttacking)//action == attack\_a) //attack

{

leftAttacking[characterStage].SetTopLeft(screenCX - 30, screenCY + m->getYMovement());

leftAttacking[characterStage].SetDelayCount(1);

leftAttacking[characterStage].OnShow();

if (leftAttacking[characterStage].IsFinalBitmap())

{

isAttacking = false;

action = walk\_a;

leftAttacking[characterStage].Reset();

}

}

else if (GetIsRising() == true)

{

leftJump[characterStage].SetTopLeft(screenCX, characterY + m->getYMovement());

leftJump[characterStage].OnShow();

leftJump[characterStage].SetDelayCount(3);

}

else if ((GetIsMovingLeft() == true || GetIsMovingRight() == true) && GetIsOnTheFloor() == true)

{

walkingLeft[characterStage].SetTopLeft(screenCX, characterY + m->getYMovement());

walkingLeft[characterStage].OnShow();

}

else

{

standLeft[characterStage].SetTopLeft(screenCX, characterY + m->getYMovement());

standLeft[characterStage].ShowBitmap();

}

}

else

{

if (isRolling)//action == roll\_a)

{

CAudio::Instance()->Play(AUDIO\_ROLL, false);

rightRolling.SetTopLeft(screenCX - 5, characterY + 10 + m->getYMovement());

rightRolling.OnShow();

rightRolling.SetDelayCount(1);

if (rightRolling.IsFinalBitmap())

action = walk\_a;

}

else if (isAttacking)//action == attack\_a)

{

rightAttacking[characterStage].SetTopLeft(screenCX + 30, screenCY + m->getYMovement());

rightAttacking[characterStage].SetDelayCount(1);

rightAttacking[characterStage].OnShow();

if (rightAttacking[characterStage].IsFinalBitmap())

{

isAttacking = false;

action = walk\_a;

rightAttacking[characterStage].Reset();

}

}

else if (GetIsRising() == true)

{

rightJump[characterStage].SetTopLeft(screenCX, characterY + m->getYMovement());

rightJump[characterStage].OnShow();

rightJump[characterStage].SetDelayCount(3);

}

else if ((GetIsMovingLeft() == true || GetIsMovingRight() == true) && GetIsOnTheFloor() == true)

{

walkingRight[characterStage].SetTopLeft(screenCX, characterY + m->getYMovement());

walkingRight[characterStage].OnShow();

}

else

{

standRight[characterStage].SetTopLeft(screenCX, characterY + m->getYMovement());

standRight[characterStage].ShowBitmap();

}

}

}

}

void Character::shouldShowMagnifier(Map\* m)

{

if (m->getMapName() == "MapS1" && characterX > 1135 && characterX < 1300)

{

m->setShowMaginifierFlag(true);

}

else if (m->getMapName() == "MapS5" && characterX > 1450 && characterX < 1620 && GetBottomY() <= 640)

{

m->setShowMaginifierFlag(true);

}

else

{

m->setShowMaginifierFlag(false);

}

}

void Character::shouldShowMessageIconFlag(Map\* m)

{

if (m->getMapName() == "MapS3" && characterX > 3280 && characterX < 3500)

{

m->setShowMessageIconFlag(true);

}

else

{

m->setShowMessageIconFlag(false);

}

}

void Character::SetMapCanShowMessage(Map\* m)

{

if (m->getMapName() == "MapS1" && characterX > 1135 && characterX < 1300)

{

m->setCanShowMessage(true, 0);

}

else if (m->getMapName() == "MapS3" && characterX > 3280 && characterX < 3500)

{

m->setCanShowMessage(true, 0);

}

else if (m->getMapName() == "MapS5" && characterX > 1450 && characterX < 1620 && GetBottomY() <= 640)

{

m->setCanShowMessage(true, 0);

}

else

{

m->setCanShowMessage(false, 0);

}

}

void Character::SetMapAutoShowMessage(Map\* m)

{

if (m->getMapName() == "MapS6" && !m->getHavePlayedAutoMessage() && characterX > 3500 && characterX < 3600)

{

m->setAutoShowMessage(true, 0);

}

else

{

m->setAutoShowMessage(false, 0);

}

}

/\*Getter\*/

int Character::GetLeftX()

{

return characterX;

}

int Character::GetTopY()

{

return characterY;

}

int Character::GetRightX()

{

return characterX + characterW;

}

int Character::GetBottomY()

{

return characterY + characterH; // + animation.Height();

}

int Character::GetCharacterStage()

{

return characterStage;

}

Map\* Character::GetMap()

{

return currentMap;

}

bool Character::GetIsMovingLeft()

{

return isMovingLeft;

}

bool Character::GetIsMovingRight()

{

return isMovingRight;

}

bool Character::GetIsMovingUp()

{

return isMovingUp;

}

int Character::GetCurrentHp()

{

return currentHp;

}

bool Character::GetIsInvincible()

{

return isInvincible;

}

int Character::GetMaxHp()

{

return maxHp;

}

bool Character::GetIsAttackedFromRight()

{

return isAttackedFromRight;

}

bool Character::GetIsAttackedFromLeft()

{

return isAttackedFromLeft;

}

bool Character::GetIsAttackedFromBottom()

{

return isAttackedFromBottom;

}

void Character::isAttackedEffectCaculation()

{

if (isAttacked)

{

isAttackedTimer.CaculateTimeForFalse(&isAttacked, 0.5);

}

if (isSparkleEffectShow)

{

sparkleEffectTimer.CaculateTimeForFalse(&isSparkleEffectShow, 0.1);

if (!isSparkleEffectShow)

{

isSparkleEffectTimerStart = false;

}

}

else

{

if (!isSparkleEffectTimerStart)

{

sparkleEffectTimer.Start();

isSparkleEffectTimerStart = true;

}

sparkleEffectTimer.CaculateTimeForTrue(&isSparkleEffectShow, 0.15);

}

}

void Character::isAttackedEffectOnShow()

{

black.SetTopLeft(screenCX, characterY - GetMap()->getYMovement());

black.ShowBitmap();

}

bool Character::GetIsOnTheFloor()

{

if (GetBottomY() == GetMap()->getFloor())

{

isOnTheFloor = true;

}

else

{

isOnTheFloor = false;

}

return isOnTheFloor;

}

bool Character::GetIsRising()

{

return isRising;

}

bool Character::GetCanGoToNextMap()

{

return canGoToNextMap;

}

bool Character::GetCanCraft()

{

return canCraft;

}

/\*Setter\*/

void Character::SetMap(Map\* m)

{

currentMap = m;

}

void Character::SetMovingDown()

{

if (GetMap()->isBridge(characterX - BORDER, GetBottomY()))

characterY += 10;

}

void Character::SetMovingLeft(bool flag)

{

isMovingLeft = flag;

}

void Character::SetMovingRight(bool flag)

{

isMovingRight = flag;

}

void Character::SetMovingUp(bool flag)

{

isMovingUp = flag;

}

void Character::SetFacingDirection(int mouseX)

{

if (characterX < 670 || GetMap()->mapScreenMoving() == false)

{

if (mouseX < characterX)

facingLR = 0;

else if (mouseX > GetRightX())

facingLR = 1;

}

else

{

if (mouseX > 670)

facingLR = 1;

else if (mouseX < 670)

facingLR = 0;

}

}

void Character::SetIsAttackedFromRight(bool flag)

{

isAttackedFromRight = flag;

}

void Character::SetIsAttackedFromLeft(bool flag)

{

isAttackedFromLeft = flag;

}

void Character::SetIsAttackedFromBottom(bool flag)

{

isAttackedFromBottom = flag;

}

void Character::SetCurrentHp(int x)

{

currentHp = x;

}

void Character::SetIsInvincible(bool flag)

{

isInvincible = flag;

}

/\*判斷角色是否能向左移動\*/

bool Character::CanMovingLeft(Map\* m, vector<Monster\*>\* monsters)

{

int monsterBorder;

if (GetIsMovingLeft())

{

if (m->isEmpty(GetLeftX() - STEP\_SIZE - BORDER, GetTopY()) && m->isEmpty(GetLeftX() - STEP\_SIZE - BORDER, GetBottomY() - BORDER))

{

if (monsters == NULL)

{

return true;

}

else

{

for (unsigned int i = 0; i < monsters->size(); i++)

{

monsterBorder = monsters->at(i)->GetBorder();

if (GetLeftX() - STEP\_SIZE - BORDER <= monsters->at(i)->GetLeftX() + monsterBorder || GetLeftX() - STEP\_SIZE - BORDER >= monsters->at(i)->GetRightX() - monsterBorder)

{

}

else if (GetBottomY() - BORDER < monsters->at(i)->GetTopY() + monsterBorder || GetTopY() + BORDER > monsters->at(i)->GetBottomY() - monsterBorder)

{

}

else

{

if (monsters->at(i)->isAlive())

return false;

}

}

return true;

}

}

}

return false;

}

/\*判斷角色是否能向右移動\*/

bool Character::CanMovingRight(Map\* m, vector<Monster\*>\* monsters)

{

int monsterBorder;

if (GetIsMovingRight())

{

if (m->isEmpty(GetRightX() + STEP\_SIZE + BORDER, GetTopY()) && m->isEmpty(GetRightX() + STEP\_SIZE + BORDER, GetBottomY() - BORDER))

{

if (monsters == NULL)

{

return true;

}

else

{

for (unsigned int i = 0; i < monsters->size(); i++)

{

monsterBorder = monsters->at(i)->GetBorder();

if (GetRightX() + STEP\_SIZE + BORDER >= monsters->at(i)->GetRightX() - monsterBorder || GetRightX() + STEP\_SIZE + BORDER <= monsters->at(i)->GetLeftX() + monsterBorder)

{

}

else if (GetBottomY() - BORDER < monsters->at(i)->GetTopY() + monsterBorder || GetTopY() + BORDER > monsters->at(i)->GetBottomY() - monsterBorder)

{

}

else

{

if (monsters->at(i)->isAlive())

return false;

}

}

return true;

}

}

}

return false;

}

void Character::SetSpeed(int x)

{

STEP\_SIZE = x;

}

int Character::GetSpeed()

{

return STEP\_SIZE;

}

void Character::SetRolling(bool flag)

{

rollingTimer.Finish();

if (GetIsOnTheFloor() && rollingTimer.GetTime() > 100)

{

rolling\_time = 5;

isRolling = true;

}

else

{

isRolling = false;

}

}

bool Character::GetIsRolling()

{

return isRolling;

}

void Character::Rolling(Map\* m, bool flag, vector<Monster\*>\* monsters) //左:0 右:1

{

action = roll\_a;

isInvincible = true;

invincibleTimer.Start();

const int ROLLING\_SIZE = 2; //角色翻滾距離

const int ROLLING\_LENGTH = 12; //角色翻滾距離

const int BORDER = 5;

if (flag)

{

if (rolling\_time >= 0)

{

if (characterX <= 670)

{

for (int i = 0; i < ROLLING\_LENGTH; i++)

{

if (m->isEmpty(GetRightX() + ROLLING\_SIZE, GetTopY()) && m->isEmpty(GetRightX() + ROLLING\_SIZE, GetBottomY() - BORDER))

{

if (characterX + ROLLING\_SIZE > 670 && characterX + ROLLING\_SIZE < 670 + ROLLING\_SIZE && GetMap()->mapScreenMoving() == true)

{

characterX = 670;

}

else if (characterX >= 670)

{

characterX += ROLLING\_SIZE;

if (GetMap()->mapScreenMoving() == true)

{

m->addSX(-ROLLING\_SIZE);

}

}

else

{

characterX += ROLLING\_SIZE;

}

}

else

break;

}

}

else

{

for (int i = 0; i < ROLLING\_LENGTH; i++)

{

if (m->isEmpty(GetRightX() + ROLLING\_SIZE, GetTopY()) && m->isEmpty(GetRightX() + ROLLING\_SIZE, GetBottomY() - BORDER))

{

characterX += ROLLING\_SIZE;

if (GetMap()->mapScreenMoving() == true)

{

m->addSX(-ROLLING\_SIZE);

}

}

else

break;

}

}

rolling\_time--;

}

else

{

rolling\_time = 0;

isRolling = false;

}

}

else

{

if (rolling\_time >= 0)

{

if (characterX <= 670 || GetMap()->mapScreenMoving() == false)

{

for (int i = 0; i < ROLLING\_LENGTH; i++)

{

if (m->isEmpty(GetLeftX() - ROLLING\_SIZE, GetTopY()) && m->isEmpty(GetLeftX() - ROLLING\_SIZE, GetBottomY() - BORDER))

characterX -= ROLLING\_SIZE;

else

break;

}

}

else

{

for (int i = 0; i < ROLLING\_LENGTH; i++)

{

if (m->isEmpty(GetLeftX() - ROLLING\_SIZE, GetTopY()) && m->isEmpty(GetLeftX() - ROLLING\_SIZE, GetBottomY() - BORDER))

{

if (characterX - ROLLING\_SIZE < 670)

{

m->addSX(characterX - 670);

characterX = 670;

break;

}

else

{

characterX -= ROLLING\_SIZE;

m->addSX(ROLLING\_SIZE);

}

}

else

break;

}

}

rolling\_time--;

}

else

{

rolling\_time = 0;

isRolling = false;

}

}

}

/\*攻擊\*/

void Character::SetAttacking(bool flag)

{

isAttacking = flag;

}

bool Character::GetIsAttacking()

{

return isAttacking;

}

void Character::SetAttackDamage(int x)

{

attackDamage = x;

}

int Character::GetAttackDamage()

{

return attackDamage;

}

void Character::addAttackDamage(int x)

{

attackDamage += x;

}

void Character::SetAttackRange(int x)

{

attackRange = x;

}

int Character::GetAttackRange()

{

return attackRange;

}

void Character::attack(vector<Monster\*>\* monsters)

{

CAudio::Instance()->Play(AUDIO\_ATTACK\_HU, false);

action = attack\_a;

isAttacking = true;

for (unsigned int i = 0; i < monsters->size(); i++)

{

if (facingLR == 0)

{

if (isAttackSuccessfullyL(attackRange, monsters->at(i)))

{

if (monsters->at(i)->isAlive())

{

//monsters[i].SetIsAttackedFromRight(true);

if (shrimpAttack)

{

monsters->at(i)->lossCurrentHp(attackDamage + 15);

shrimpAttack = false;

ShrimpAttackTimer.Start();

}

else

{

monsters->at(i)->lossCurrentHp(attackDamage);

}

}

}

}

else

{

if (isAttackSuccessfullyR(attackRange, monsters->at(i)))

{

if (monsters->at(i)->isAlive())

{

//monsters[i].SetIsAttackedFromLeft(true);

if (shrimpAttack)

{

monsters->at(i)->lossCurrentHp(attackDamage + 15);

shrimpAttack = false;

ShrimpAttackTimer.Start();

}

else

{

monsters->at(i)->lossCurrentHp(attackDamage);

}

}

}

}

}

}

void Character::attackOnMove()

{

if (facingLR == 0)

{

leftAttacking[0].OnMove();

leftAttacking[1].OnMove();

}

else

{

rightAttacking[0].OnMove();

rightAttacking[1].OnMove();

}

}

bool Character::isAttackSuccessfullyL(int range, Monster\* monster)

{

if (((monster->GetRightX() >= GetLeftX() - range && monster->GetRightX() <= GetLeftX()) ||

(monster->GetLeftX() >= GetLeftX() - range && monster->GetLeftX() <= GetLeftX()) ||

(monster->GetLeftX() <= GetLeftX() - range && monster->GetRightX() >= GetLeftX()))

&& ((monster->GetBottomY() >= GetTopY() && monster->GetBottomY() <= GetBottomY()) ||

(monster->GetTopY() >= GetTopY() && monster->GetTopY() <= GetBottomY()) ||

(monster->GetTopY() <= GetTopY() && monster->GetBottomY() >= GetBottomY())))

{

return true;

}

else

{

return false;

}

}

bool Character::isAttackSuccessfullyR(int range, Monster\* monster)

{

if (((monster->GetLeftX() <= GetRightX() + range && monster->GetLeftX() >= GetRightX()) ||

(monster->GetRightX() <= GetRightX() + range && monster->GetRightX() >= GetRightX()) ||

(monster->GetLeftX() <= GetRightX() && monster->GetRightX() >= GetRightX() + range))

&& ((monster->GetBottomY() >= GetTopY() && monster->GetBottomY() <= GetBottomY()) ||

(monster->GetTopY() >= GetTopY() && monster->GetTopY() <= GetBottomY()) ||

(monster->GetBottomY() >= GetBottomY() && monster->GetTopY() <= GetTopY())))

{

return true;

}

else

{

return false;

}

}

/\*餐點能力\*/

SourceStorage\* Character::GetSourceStorage()

{

return sourceStorage;

}

PropStorage\* Character::GetPropStorage()

{

return propStorage;

}

void Character::EatMosquitoJump(bool flag)

{

isMosquitoJump = flag;

SetDoubleJump(flag);

DJtemp = flag;

}

bool Character::CanDoubleJump()

{

return doubleJump;

}

void Character::SetDoubleJump(bool flag)

{

doubleJump = flag;

}

void Character::EatGrassFast(bool flag)

{

if (flag && !isGrassFast)

{

ChangeSpeed(1.4);

}

else if(!flag && isGrassFast)

{

ChangeSpeed(0.72);

}

isGrassFast = flag;

}

void Character::ChangeSpeed(double m)

{

STEP\_SIZE = int(STEP\_SIZE \* m);

}

void Character::EatShrimpBlood(bool flag)

{

if (flag && !isShrimpBlood)

{

maxHp += 20;

}

else if(!flag && isShrimpBlood)

{

maxHp -= 20;

}

isShrimpBlood = flag;

}

void Character::addMaxHp(int blood)

{

maxHp += blood;

}

void Character::EatGuavaJuiceBlood(bool flag)

{

isGuavaJuiceBlood = flag;

healBlood = flag;

}

void Character::healBloodEveryTenSeconds()

{

if (healBloodTimer.GetTime() == 0)

{

healBloodTimer.Start();

}

else

{

healBloodTimer.Finish();

if (healBloodTimer.GetTime() >= 10 \* CLOCKS\_PER\_SEC)

{

restoreCurrentHp(3);

healBloodTimer.Start();

}

}

}

void Character::EatBananaAttack(bool flag)

{

if (flag && !isBananaAttack)

{

attackDamage += 3;

}

else if(!flag && isBananaAttack)

{

attackDamage -= 3;

}

isBananaAttack = flag;

}

void Character::EatShrimpAttack(bool flag)

{

if (flag && !isShrimpAttack)

{

isShrimpAttack = true;

ShrimpAttackTimer.Start();

}

else if (!flag && isShrimpAttack)

{

isShrimpAttack = false;

}

}

void Character::SetIsMosquitoJump(bool flag)

{

isMosquitoJump = flag;

}

bool Character::GetIsMosquitoJump()

{

return isMosquitoJump;

}

void Character::SetIsGrassFast(bool flag)

{

isGrassFast = flag;

}

bool Character::GetIsGrassFast()

{

return isGrassFast;

}

void Character::SetIsShrimpBlood(bool flag)

{

isShrimpBlood = flag;

}

bool Character::GetIsShrimpBlood()

{

return isShrimpBlood;

}

void Character::SetIsGuavaJuiceBlood(bool flag)

{

isGuavaJuiceBlood = flag;

}

bool Character::GetIsGuavaJuiceBlood()

{

return isGuavaJuiceBlood;

}

void Character::SetIsBananaAttack(bool flag)

{

isBananaAttack = flag;

}

bool Character::GetIsBananaAttack()

{

return isBananaAttack;

}

void Character::SetIsShrimpAttack(bool flag)

{

isShrimpAttack = flag;

}

bool Character::GetIsShrimpAttack()

{

return isShrimpAttack;

}

void Character::SetLightBulbNum(int num)

{

lightBulbNum = num;

}

int Character::GetLightBulbNum()

{

return lightBulbNum;

}

void Character::AddLightBulb(int num)

{

lightBulbNum += num;

}

void Character::ConsumeLightBulb(int num)

{

lightBulbNum -= num;

}

void Character::SetAllSourceNumToHundred()

{

sourceStorage->getSource(0)->setNum(100);

sourceStorage->getSource(1)->setNum(100);

sourceStorage->getSource(2)->setNum(100);

sourceStorage->getSource(3)->setNum(100);

sourceStorage->getSource(4)->setNum(100);

sourceStorage->getSource(5)->setNum(100);

}

void Character::restoreCurrentHp(int n)

{

(GetCurrentHp() + n) >= GetMaxHp() ? SetCurrentHp(GetMaxHp()) : currentHp += n;

}

void Character::lossCurrentHp(int n)

{

currentHp -= n;

isInvincible = true;

invincibleTimer.Start();

isAttacked = true;

isAttackedTimer.Start();

isSparkleEffectShow = true;

sparkleEffectTimer.Start();

isSparkleEffectTimerStart = true;

}

void Character::hpLimit()

{

if (currentHp > maxHp) {

currentHp = maxHp;

}

}

bool Character::isIntersect(int lX, int rX, int tY, int bY)

{

if ((GetRightX() >= lX && GetRightX() <= rX ||

GetLeftX() <= rX && GetLeftX() >= lX ||

GetLeftX() <= lX && GetRightX() >= rX || //角色比東西寬

GetRightX() <= rX && GetLeftX() >= lX) && //東西比角色寬

((GetBottomY() >= tY && GetBottomY() <= bY) ||

(GetTopY() >= tY && GetTopY() <= bY) ||

(GetTopY() >= tY && GetBottomY() <= bY) ||

(GetTopY() <= tY && GetBottomY() >= bY)))

{

return true;

}

return false;

}

void Character::characterHasSword(bool flag)

{

if (flag)

{

attackDamage = 25;

attackRange = 170;

characterStage = 1;

}

else

{

attackDamage = 10;

attackRange = 60;

characterStage = 0;

}

}

void Character::SetXY(int x, int y)

{

characterX = x;

characterY = y;

}

void Character::SetCanGoToNextMap(bool flag)

{

canGoToNextMap = flag;

}

void Character::SetCanCraft(bool flag)

{

canCraft = flag;

}

void Character::BloodShow()

{

int frameNum;

int fullHeartNum;

int chagingHeart;

frameNum = GetMaxHp() / 20;

fullHeartNum = GetCurrentHp() / 20;

chagingHeart = GetCurrentHp() - fullHeartNum \* 20;

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

{

bloodFrame.SetTopLeft(i \* 70, 0);

bloodFrame.ShowBitmap();

}

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

{

characterBlood[0].SetTopLeft(i \* 70, 0);

characterBlood[0].ShowBitmap();

}

if (chagingHeart == 0)

{

}

else if (chagingHeart <= 2)

{

characterBlood[9].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[9].ShowBitmap();

}

else if (chagingHeart <= 4)

{

characterBlood[8].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[8].ShowBitmap();

}

else if (chagingHeart <= 6)

{

characterBlood[7].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[7].ShowBitmap();

}

else if (chagingHeart <= 8)

{

characterBlood[6].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[6].ShowBitmap();

}

else if (chagingHeart <= 10)

{

characterBlood[5].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[5].ShowBitmap();

}

else if (chagingHeart <= 12)

{

characterBlood[4].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[4].ShowBitmap();

}

else if (chagingHeart <= 14)

{

characterBlood[3].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[3].ShowBitmap();

}

else if (chagingHeart <= 16)

{

characterBlood[2].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[2].ShowBitmap();

}

else if (chagingHeart <= 18)

{

characterBlood[1].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[1].ShowBitmap();

}

else if (chagingHeart <= 20)

{

characterBlood[0].SetTopLeft(fullHeartNum \* 70, 0);

characterBlood[0].ShowBitmap();

}

}

void Character::ScreenCXY(Map\* m)

{

int bmpBorderX = 0;

int bmpBorderY = 0;

if (characterStage == 1)

{

if (facingLR)

{

if (isAttacking)

{

bmpBorderX = 105;

bmpBorderY = 105;

}

else

bmpBorderX = 25;

}

else

{

if (isAttacking)

{

bmpBorderX = 155;

bmpBorderY = 105;

}

else

bmpBorderX = 35;

}

}

screenCY = characterY - bmpBorderY;

if (characterX < 670 || m == NULL || m->mapScreenMoving() == false)

{

screenCX = characterX - bmpBorderX;

}

else

{

screenCX = 670 - bmpBorderX;

}

}

}

===============================

CharacterStatus.h

===============================

#ifndef \_\_CHARACTERSTATUS\_\_

#define \_\_CHARACTERSTATUS\_\_

namespace game\_framework

{

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

// 這個class為角色狀態

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

class Character;

class CharacterStatus

{

public:

CharacterStatus();

~CharacterStatus();

void LoadBitmap();

void Initialize(Character\* character);

void onShow();

void showhp();

void showAttackDamage();

void showAttackRange();

void showSpeed();

void showEffect();

void effectCheck();

void showData(int x, int y, int data);

void showData(int x, int y, char\* data);

void showData(int x, int y, int data1, int data2);

void setStage(Stage stage);

Stage getStage();

void showStage();

private:

CMovingBitmap characterStatus;

Character\* \_character;

Stage \_stage;

char\* cstage;

int row;

int col[3];

bool haveFirstLine;

int x, y;

int mjLen, gfLen, sbLen;

};

}

#endif

===============================

CharacterStatus.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include <vector>

#include "Character.h"

#include "CharacterStatus.h"

namespace game\_framework

{

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

// 這個class為道具書模式

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

CharacterStatus::CharacterStatus()

{

}

CharacterStatus::~CharacterStatus()

{

}

void CharacterStatus::LoadBitmap()

{

characterStatus.LoadBitmap(".\\res\\character\_status.bmp");

}

void CharacterStatus::Initialize(Character\* character)

{

\_character = character;

\_stage = stage\_1;

row = 0;

col[3] = { 0 };

haveFirstLine = false;

x = 508;

y = 434;

mjLen = 120;

gfLen = 150;

sbLen = 150;

}

void CharacterStatus::onShow()

{

characterStatus.SetTopLeft(282, 113);

characterStatus.ShowBitmap();

showhp();

showAttackDamage();

showAttackRange();

showSpeed();

showEffect();

showStage();

}

void CharacterStatus::showhp()

{

showData(450, 222, \_character->GetCurrentHp(), \_character->GetMaxHp());

}

void CharacterStatus::showAttackDamage()

{

showData(474, 276, \_character->GetAttackDamage());

}

void CharacterStatus::showAttackRange()

{

showData(503, 329, \_character->GetAttackRange());

}

void CharacterStatus::showSpeed()

{

showData(447, 381, \_character->GetSpeed());

}

void CharacterStatus::showEffect()

{

x = 508;

y = 434;

effectCheck();

if (haveFirstLine)

{

if (col[0] == 1)

{

showData(x, y, "二段跳");

x += mjLen;

}

if (col[1] == 1)

{

showData(x, y, "速度+40%");

x += gfLen;

}

if (col[2] == 1)

{

showData(x, y, "最大生命+20");

x += sbLen;

}

y += 45;

x = 508;

}

if (\_character->GetIsGuavaJuiceBlood())

{

showData(x, y, "每十秒恢復3點血量");

y += 45;

}

if (\_character->GetIsBananaAttack())

{

showData(x, y, "近戰攻擊+3");

y += 45;

}

if (\_character->GetIsShrimpAttack())

{

showData(x, y, "不攻擊一段時間，下次攻擊+15");

y += 45;

}

}

void CharacterStatus::effectCheck()

{

row = 0;

col[3] = { 0 };

if (\_character->GetIsMosquitoJump() || \_character->GetIsGrassFast() || \_character->GetIsShrimpBlood())

{

haveFirstLine = true;

row += 1;

if (\_character->GetIsMosquitoJump())

{

col[0] = 1;

}

else

{

col[0] = 0;

}

if (\_character->GetIsGrassFast())

{

col[1] = 1;

}

else

{

col[1] = 0;

}

if (\_character->GetIsShrimpBlood())

{

col[2] = 1;

}

else

{

col[2] = 0;

}

}

}

void CharacterStatus::showData(int x, int y, int data)

{

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(200, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255, 255, 255));

char position[500]; // Demo 數字對字串的轉換

sprintf(position, "%d", data);

pDC->TextOut(x, y, position);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

void CharacterStatus::showData(int x, int y, char\* data)

{

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(200, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkMode(TRANSPARENT);

pDC->SetTextColor(RGB(255, 255, 255));

char position[500]; // Demo 數字對字串的轉換

sprintf(position, "%s", data);

pDC->TextOut(x, y, position);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

void CharacterStatus::showData(int x, int y, int data1, int data2)

{

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(200, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkMode(TRANSPARENT);

pDC->SetTextColor(RGB(255, 255, 255));

char position[500]; // Demo 數字對字串的轉換

sprintf(position, "%d / %d", data1, data2);

pDC->TextOut(x, y, position);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

void CharacterStatus::setStage(Stage stage)

{

\_stage = stage;

}

Stage CharacterStatus::getStage()

{

return \_stage;

}

void CharacterStatus::showStage()

{

if (\_stage == stage\_1)

{

cstage = "第一關";

}

else if (\_stage == stage\_2)

{

cstage = "第二關";

}

else if (\_stage == stage\_3)

{

cstage = "第三關";

}

else if (\_stage == stage\_4)

{

cstage = "第四關";

}

else if (\_stage == stage\_5)

{

cstage = "第五關";

}

else if (\_stage == stage\_6)

{

cstage = "第六關";

}

else if (\_stage == stage\_boss)

{

cstage = "王關";

}

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(240, "標楷體"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkMode(TRANSPARENT);

pDC->SetTextColor(RGB(255, 255, 255));

char position[500]; // Demo 數字對字串的轉換

sprintf(position, "%s", cstage);

pDC->TextOut(478, 150, position);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

}

===============================

Map.h

===============================

#ifndef \_\_MAPS\_H\_\_

#define \_\_MAPS\_H\_\_

#include "Util.h"

namespace game\_framework {

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

// 這個class提供地圖構成

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

class Map

{

public:

Map(int x, int y);

string getMapName();

void setMapName(string name);

int getStartPosition();

int getFinalPosition();

void setSX(int x);

void setSY(int y);

int getSX();

int getSY();

bool mapScreenMoving();

virtual void addSX(int n);

virtual void addSY(int n);

virtual int screenX(int x) { return 0; };

virtual int screenY(int y) { return 0; };

void setCharacterX(int x);

void setCharacterY(int y);

void setScreenMoving(bool flag);

virtual void setPos(int x, int y, int n) = 0;

void setXY(int x, int y);

void setInitialX(int x);

int getInitialX();

void setInitialY(int y);

int getInitialY();

void setInitialXY(int x, int y);

int getXMovement();

int getYMovement();

//地圖方塊設定

virtual bool isEmpty(int x, int y) const = 0;

virtual bool isPortal(int x, int y) const = 0;

virtual bool isBridge(int x, int y) const = 0;

virtual bool isCraftTable(int x, int y) const = 0;

virtual bool isTrashCan(int x, int y) const = 0;

virtual bool isGetHurtPlace(int x, int y) const = 0;

//角色及怪物地板、天花板

virtual void monsterFloorChanging(int x) {};

int getMonsterFloor(); //怪物地板

void setMonsterFloor(int y);

virtual void characterFloorAndCeiling() {};

int getFloor(); //角色地板

void setFloor(int y);

int getCeiling(); //角色天花板

void setCeiling(int y);

//鍛造台、傳送門、垃圾桶

bool getCraftTableOpen();

void setCraftTableOpen(bool flag);

bool getPortalOpen();

void setPortalOpen(bool flag);

bool getTrashCanOpen();

void setTrashCanOpen(bool flag);

/\*訊息相關函式\*/

//放大鏡

void loadMagnifierBitmap();

void loadMagnifierLBitmap();

void showMaginifier(int x, int y);

void showMaginifierL(int x, int y);

void setShowMaginifierFlag(bool flag);

bool getShowMaginifierFlag();

//訊息圖示

void loadMessageIconBitmap();

void showMessageIcon(int x, int y);

void setShowMessageIconFlag(bool flag);

bool getShowMessageIconFlag();

//訊息

void setCanShowMessage(bool flag, int code);

bool getCanShowMessage();

void setAutoShowMessage(bool flag, int code);

bool getAutoShowMessage();

void setHavePlayedAutoMessage(bool flag);

bool getHavePlayedAutoMessage();

void setMessageCounter(int n);

int getMessageCounter();

void setMessageCounterToNext();

void setMessageEndFlag(bool flag);

bool getMessageEndFlag();

void InitializeMessage();

virtual void messageOnShow();

protected:

CMovingBitmap exitBitmap;

CMovingBitmap pressEBitmap;

CMovingBitmap trashCanOpenBitmap, trashCanClosedBitmap;

CMovingBitmap craftingBitmap;

int sx, sy; //screen 的座標

int characterX, characterY;

int initialX, initialY; //初始X、Y

int startX; //開始位置

int finalX; //結束位置

string mapName;

int floor;

int monsterFloor;

int ceiling;

bool screenMoving;

bool portalOpen;

bool craftTableOpen;

bool trashCanOpen;

CMovingBitmap magnifier;

CMovingBitmap magnifierL;

bool showMaginifierFlag;

CMovingBitmap messageIcon;

bool showMessageIconFlag;

int messageCode;

bool canShowMessage;

bool autoShowMessage;

bool havePlayedAutoMessage;

int messageCounter;

int messageSize;

bool messageEndFlag;

};

}

#endif

===============================

Map.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include <vector>

namespace game\_framework

{

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

// 這個class提供地圖構成

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

Map::Map(int x, int y)

{

sx = x;

sy = y;

portalOpen = false;

}

string Map::getMapName()

{

return mapName;

}

int Map::getStartPosition()

{

return startX;

}

int Map::getFinalPosition()

{

return finalX;

}

void Map::setSX(int x)

{

sx = x;

}

void Map::setSY(int y)

{

sy = y;

}

int Map::getSX()

{

return sx;

}

int Map::getSY()

{

return sy;

}

bool Map::mapScreenMoving()

{

return screenMoving;

}

int Map::getFloor()

{

return floor;

}

int Map::getCeiling()

{

return ceiling;

}

bool Map::getCraftTableOpen()

{

return craftTableOpen;

}

void Map::setCraftTableOpen(bool flag)

{

craftTableOpen = flag;

}

bool Map::getPortalOpen()

{

return portalOpen;

}

void Map::setMapName(string name)

{

mapName = name;

}

int Map::getMonsterFloor()

{

return monsterFloor;

}

void Map::setMonsterFloor(int y)

{

monsterFloor = y;

}

void Map::addSX(int n) //sx+=n 表示地圖向左移動n

{

sx += n;

}

void Map::addSY(int n) //sx+=n 表示地圖上移動n

{

sy += n;

}

void Map::setCharacterX(int x)

{

characterX = x;

}

void Map::setCharacterY(int y)

{

characterY = y;

}

void Map::setXY(int x, int y)

{

sx = x;

sy = y;

}

void Map::setInitialX(int x)

{

initialX = x;

}

int Map::getInitialX()

{

return initialX;

}

void Map::setInitialY(int y)

{

initialY = y;

}

int Map::getInitialY()

{

return initialY;

}

void Map::setInitialXY(int x, int y)

{

initialX = x;

initialY = y;

}

int Map::getXMovement()

{

return sx - initialX;

}

int Map::getYMovement()

{

return sy - initialY;

}

void Map::loadMagnifierBitmap()

{

magnifier.LoadBitmap(".\\res\\magnifier.bmp", RGB(0, 0, 0));

}

void Map::loadMagnifierLBitmap()

{

magnifierL.LoadBitmap(".\\res\\magnifierL.bmp", RGB(0, 0, 0));

}

void Map::showMaginifier(int x, int y)

{

magnifier.SetTopLeft(x + getSX(), y + getSY());

magnifier.ShowBitmap();

}

void Map::showMaginifierL(int x, int y)

{

magnifierL.SetTopLeft(x + getSX(), y + getSY());

magnifierL.ShowBitmap();

}

void Map::setShowMaginifierFlag(bool flag)

{

showMaginifierFlag = flag;

}

bool Map::getShowMaginifierFlag()

{

return showMaginifierFlag;

}

void Map::loadMessageIconBitmap()

{

messageIcon.LoadBitmap(".\\res\\message\_icon.bmp", RGB(0, 0, 0));

}

void Map::showMessageIcon(int x, int y)

{

messageIcon.SetTopLeft(x + getSX(), y + getSY());

messageIcon.ShowBitmap();

}

void Map::setShowMessageIconFlag(bool flag)

{

showMessageIconFlag = flag;

}

bool Map::getShowMessageIconFlag()

{

return showMessageIconFlag;

}

void Map::setCanShowMessage(bool flag, int code)

{

canShowMessage = flag;

messageCode = code;

}

bool Map::getCanShowMessage()

{

return canShowMessage;

}

void Map::setAutoShowMessage(bool flag, int code)

{

autoShowMessage = flag;

messageCode = code;

}

bool Map::getAutoShowMessage()

{

return autoShowMessage;

}

void Map::setHavePlayedAutoMessage(bool flag)

{

havePlayedAutoMessage = flag;

}

bool Map::getHavePlayedAutoMessage()

{

return havePlayedAutoMessage;

}

void Map::setFloor(int y)

{

floor = y;

}

void Map::setCeiling(int y)

{

ceiling = y;

}

void Map::setScreenMoving(bool flag)

{

screenMoving = flag;

}

void Map::setPortalOpen(bool flag)

{

portalOpen = flag;

}

bool Map::getTrashCanOpen()

{

return trashCanOpen;

}

void Map::setTrashCanOpen(bool flag)

{

trashCanOpen = flag;

}

void Map::setMessageCounter(int n)

{

messageCounter = n;

}

int Map::getMessageCounter()

{

return messageCounter;

}

void Map::setMessageCounterToNext()

{

if (messageCounter != messageSize)

{

messageCounter += 1;

}

else

{

messageCounter = 1;

}

}

void Map::setMessageEndFlag(bool flag)

{

messageEndFlag = flag;

}

bool Map::getMessageEndFlag()

{

return messageEndFlag;

}

void Map::InitializeMessage()

{

messageCounter = 1;

messageEndFlag = false;

}

void Map::messageOnShow()

{

}

}

===============================

MapS1.h

===============================

#ifndef \_\_MAPS1\_\_

#define \_\_MAPS1\_\_

namespace game\_framework

{

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

// 這個class提供地圖構成

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

class MapS1 : public Map

{

public:

MapS1();

~MapS1();

void LoadBitmap();

void Initialize();

void setPos(int x, int y, int n); //將地圖座標的陣列設為傳入的數字

bool isEmpty(int x, int y) const;

bool isPortal(int x, int y) const;

bool isBridge(int x, int y) const;

bool isCraftTable(int x, int y) const;

bool isTrashCan(int x, int y) const;

bool isGetHurtPlace(int x, int y) const;

void onShow();

void monsterFloorChanging(int x) override;

int screenX(int x) override;

int screenY(int y) override;

void messageOnShow() override;

protected:

CMovingBitmap map;

int X, Y;

int mapGrid[100][18];

CMovingBitmap white, blue;

int gridW, gridH;

CMovingBitmap message01;

CMovingBitmap message02;

CMovingBitmap message03;

CMovingBitmap message04;

};

}

#endif

===============================

MapS1.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "MapS1.h"

#include <vector>

namespace game\_framework

{

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

// 這個class提供地圖構成

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

MapS1::MapS1() : Map(0, 0) //地圖設置：0為不能走、1為可以走、2為傳送門、3為橋、4為鍛造台、5為垃圾桶、6為受傷區域

{

X = 40;

Y = 40;

gridW = 40;

gridH = 40;

startX = 335;

finalX = 2450;

setFloor(680);

setMonsterFloor(680);

int mapGrid\_init[100][18];

for (int i = 0; i < 100; i++)

{

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

{

mapGrid\_init[i][j] = 0;

}

}

//給予地圖左上角座標及每張小圖寬度

for (int i = 2; i < 45; i++)

{

for (int j = 1; j < 17; j++)

{

mapGrid\_init[i][j] = 1;

}

}

for (int i = 45; i < 60; i++)

{

for (int j = 1; j < 16; j++)

{

mapGrid\_init[i][j] = 1;

}

}

for (int i = 60; i < 71; i++)

{

for (int j = 1; j < 12; j++)

{

mapGrid\_init[i][j] = 1;

}

}

for (int j = 1; j < 10; j++)

{

mapGrid\_init[52][j] = 0;

}

for (int i = 60; i < 65; i++)

{

for (int j = 5; j < 12; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 0; i < 100; i++)

{

for (int j = 0; j < 19; j++)

{

mapGrid[i][j] = mapGrid\_init[i][j]; //依序填入mapGrid

}

}

}

MapS1::~MapS1()

{

}

void MapS1::LoadBitmap()

{

map.LoadBitmap(".\\res\\map01.bmp");

exitBitmap.LoadBitmap(IDB\_EXIT, RGB(0, 0, 0));

pressEBitmap.LoadBitmap(IDB\_PRESSE, RGB(0, 0, 0));

message01.LoadBitmap(".\\res\\message0101.bmp");

message02.LoadBitmap(".\\res\\message0102.bmp");

message03.LoadBitmap(".\\res\\message0103.bmp");

message04.LoadBitmap(".\\res\\message0104.bmp");

loadMagnifierBitmap();

loadMessageIconBitmap();

}

void MapS1::Initialize()

{

setXY(-410, -320);

setInitialXY(-410, -320);

setScreenMoving(true);

setMapName("MapS1");

messageCode = 0;

canShowMessage = false;

autoShowMessage = false;

havePlayedAutoMessage = false;

showMaginifierFlag = false;

showMessageIconFlag = false;

messageCounter = 1;

messageSize = 4;

messageEndFlag = false;

}

void MapS1::setPos(int x, int y, int n)

{

int gridX = x / 40;

int gridY = y / 40;

mapGrid[gridX][gridY] = n;

}

bool MapS1::isEmpty(int x, int y) const

{

int gridX = x / 40;

int gridY = y / 40;

if (mapGrid[gridX][gridY] != 0)

{

return true;

}

return false;

}

bool MapS1::isPortal(int x, int y) const

{

int gridX = x / 40;

int gridY = y / 40;

if (mapGrid[gridX][gridY] == 2)

{

return true;

}

return false;

}

bool MapS1::isBridge(int x, int y) const

{

int gridX = x / 40;

int gridY = y / 40;

if (mapGrid[gridX][gridY] == 3)

{

return true;

}

return false;

}

bool MapS1::isCraftTable(int x, int y) const

{

return false;

}

bool MapS1::isTrashCan(int x, int y) const

{

return false;

}

bool MapS1::isGetHurtPlace(int x, int y) const

{

return false;

}

void MapS1::onShow()

{

map.SetTopLeft(getSX(), getSY());

map.ShowBitmap();

if (getPortalOpen())

{

exitBitmap.SetTopLeft(2895 + getSX(), 180);

exitBitmap.ShowBitmap();

pressEBitmap.SetTopLeft(600, 810 + getSY());

pressEBitmap.ShowBitmap();

}

if (getShowMaginifierFlag())

{

showMaginifier(1615, 845);

pressEBitmap.SetTopLeft(600, 1020 + getSY());

pressEBitmap.ShowBitmap();

}

if (getSX() > -1502)

{

setFloor(680);

}

else if (getSX() > -2081)

{

setFloor(600);

}

else

{

setFloor(460);

}

if (getSX() > -425)

{

setCeiling(0);

}

else if (getSX() > -1865)

{

setCeiling(420);

}

else

{

setCeiling(0);

}

}

void MapS1::monsterFloorChanging(int x)

{

if (x < 1750)

{

setMonsterFloor(680);

}

else if (x < 2500)

{

setMonsterFloor(600);

}

else

{

setMonsterFloor(500);

}

}

int MapS1::screenX(int x)

{

return x + getSX();

}

int MapS1::screenY(int y)

{

return y + getSY();

}

void MapS1::messageOnShow()

{

if (messageCounter == 1)

{

message01.SetTopLeft(173, 535);

message01.ShowBitmap();

}

else if (messageCounter == 2)

{

message02.SetTopLeft(173, 535);

message02.ShowBitmap();

}

else if (messageCounter == 3)

{

message03.SetTopLeft(173, 535);

message03.ShowBitmap();

}

else if (messageCounter == 4)

{

message04.SetTopLeft(173, 535);

message04.ShowBitmap();

messageEndFlag = true;

}

}

}

===============================

MapS2.h

===============================

#ifndef \_\_MAPS2\_\_

#define \_\_MAPS2\_\_

namespace game\_framework

{

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

// 這個class提供地圖構成

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

class MapS2 : public Map

{

public:

MapS2();

~MapS2();

void LoadBitmap();

void Initialize();

void setPos(int x, int y, int n); //將地圖座標的陣列設為傳入的數字

bool isEmpty(int x, int y) const;

bool isPortal(int x, int y) const;

bool isBridge(int x, int y) const;

bool isCraftTable(int x, int y) const;

bool isTrashCan(int x, int y) const;

bool isGetHurtPlace(int x, int y) const;

void onShow();

void syMoving(int y);

void monsterFloorChanging(int x) override;

void characterFloorAndCeiling() override;

int screenX(int x) override;

int screenY(int y) override;

protected:

CMovingBitmap map;

int X, Y;

int mapGrid[400][70];

CMovingBitmap white, blue;

int gridW, gridH;

int previousFloor;

int targetSY;

bool screenUp;

bool screenDown;

};

}

#endif

===============================

MapS2.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "MapS2.h"

#include <vector>

namespace game\_framework

{

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

// 這個class提供地圖構成

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

MapS2::MapS2() : Map(0, 0) //地圖設置：0為不能走、1為可以走、2為傳送門、3為橋、4為鍛造台、5為垃圾桶、6為受傷區域

{

X = 20;

Y = 20;

gridW = 20;

gridH = 20;

startX = 100;

finalX = 3390;

setFloor(540);

setCeiling(0);

setMonsterFloor(540);

int mapGrid\_init[400][70];

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 1;

}

}

//給予地圖左上角座標及每張小圖寬度

for (int i = 0; i < 11; i++)

{

for (int j = 27; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 10; i++)

{

for (int j = 0; j < 27; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 11; i < 26; i++)

{

mapGrid\_init[i][34] = 6;

}

for (int i = 42; i < 77; i++)

{

for (int j = 0; j < 13; j++)

{

mapGrid\_init[i][j] = 0;

}

}

mapGrid\_init[42][14] = 0;

for (int i = 27; i < 77; i++)

{

for (int j = 22; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 95; i < 114; i++)

{

for (int j = 12; j < 37; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 77; i < 94; i++)

{

mapGrid\_init[i][29] = 3;

mapGrid\_init[i][36] = 3;

}

for (int i = 186; i < 190; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 166; i < 186; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid[i][j] = mapGrid\_init[i][j]; //依序填入mapGrid

}

}

}

MapS2::~MapS2()

{

}

void MapS2::LoadBitmap()

{

map.LoadBitmap(".\\res\\map02.bmp");

exitBitmap.LoadBitmap(IDB\_EXIT, RGB(0, 0, 0));

pressEBitmap.LoadBitmap(IDB\_PRESSE, RGB(0, 0, 0));

loadMagnifierBitmap();

loadMessageIconBitmap();

}

void MapS2::Initialize()

{

setXY(-100, -260);

setInitialXY(-100, -260);

setScreenMoving(true);

setMapName("MapS2");

setCeiling(0);

previousFloor = 0;

targetSY = 0;

screenUp = screenDown = false;

messageCode = 0;

canShowMessage = false;

autoShowMessage = false;

havePlayedAutoMessage = false;

showMaginifierFlag = false;

showMessageIconFlag = false;

messageCounter = 1;

messageSize = 0;

messageEndFlag = false;

}

void MapS2::setPos(int x, int y, int n)

{

int gridX = x / 20;

int gridY = y / 20;

mapGrid[gridX][gridY] = n;

}

bool MapS2::isEmpty(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] != 0)

{

return true;

}

return false;

}

bool MapS2::isPortal(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] == 2)

{

return true;

}

return false;

}

bool MapS2::isBridge(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] == 3)

{

return true;

}

return false;

}

bool MapS2::isCraftTable(int x, int y) const

{

return false;

}

bool MapS2::isTrashCan(int x, int y) const

{

return false;

}

bool MapS2::isGetHurtPlace(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] == 6)

{

return true;

}

return false;

}

void MapS2::onShow()

{

map.SetTopLeft(getSX(), getSY());

map.ShowBitmap();

syMoving(targetSY);

if (getPortalOpen())

{

if (characterX < 1000)

exitBitmap.SetTopLeft(150 + getSX(), 245);

else

exitBitmap.SetTopLeft(3470 + getSX(), 300);

pressEBitmap.SetTopLeft(600, 615);

exitBitmap.ShowBitmap();

pressEBitmap.ShowBitmap();

}

}

void MapS2::syMoving(int y)

{

if (screenUp)

{

if (getSY()-20 > y)

{

addSY(-20);

}

else

{

screenUp = false;

addSY(getSY()-y);

}

}

if (screenUp)

{

if (getSY()-20 > y)

{

addSY(-20);

}

else

{

screenUp = false;

addSY(y - getSY());

}

}

else if (screenDown)

{

if (getSY() + 20 < y)

{

addSY(20);

}

else

{

screenDown = false;

addSY(y - getSY());

}

}

}

void MapS2::monsterFloorChanging(int x)

{

if (x < 1750)

setMonsterFloor(435);

else if (x < 2500)

setMonsterFloor(560);

else

setMonsterFloor(880);

}

void MapS2::characterFloorAndCeiling()

{

if (characterX < 258)

setFloor(540);

else if (characterX < 440)

setFloor(680);

else if (characterX < 520)

{

if (characterY > 382)

setFloor(680);

else

setFloor(500);

}

else if (characterX < 1578)

{

setFloor(440);

if (characterX > 779)

setCeiling(265);

else

setCeiling(0);

}

else

{

if (characterY > 612 || characterX > 2280)

{

setFloor(880);

if (characterX > 1850 && characterX < 2285)

setCeiling(740);

else

setCeiling(470);

if (getSY() != -550)

{

screenUp = true;

targetSY = -550;

}

previousFloor = 880;

}

else if (characterY > 462)

{

setFloor(730);

setCeiling(265);

if (characterY > 490 && previousFloor == 580 && getSY() != -550)

{

screenUp = true;

targetSY = -550;

}

}

else

{

setFloor(580);

if (getSY() != -260)

{

screenDown = true;

targetSY = -260;

}

previousFloor = 580;

}

}

}

int MapS2::screenX(int x)

{

return x + getSX();

}

int MapS2::screenY(int y)

{

return y + getSY();

}

}

===============================

MapS3.h

===============================

#ifndef \_\_MAPS3\_\_

#define \_\_MAPS3\_\_

namespace game\_framework

{

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

// 這個class提供地圖構成

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

class MapS3 : public Map

{

public:

MapS3();

~MapS3();

void LoadBitmap();

void Initialize();

void setPos(int x, int y, int n); //將地圖座標的陣列設為傳入的數字

bool isEmpty(int x, int y) const;

bool isPortal(int x, int y) const;

bool isBridge(int x, int y) const;

bool isCraftTable(int x, int y) const;

bool isTrashCan(int x, int y) const;

bool isGetHurtPlace(int x, int y) const;

int blockProperty(int x, int y) const;

void onShow();

void monsterFloorChanging(int x) override;

void characterFloorAndCeiling() override;

int screenX(int x) override;

int screenY(int y) override;

void messageOnShow() override;

protected:

CMovingBitmap map;

int X, Y;

int mapGrid[400][70];

CMovingBitmap white, blue;

int gridW, gridH;

int previousFloor;

int targetSY;

bool screenUp;

bool screenDown;

CMovingBitmap message01;

CMovingBitmap message02;

CMovingBitmap message03;

CMovingBitmap message04;

CMovingBitmap message05;

CMovingBitmap message06;

};

}

#endif

===============================

MapS3.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "MapS3.h"

#include <vector>

namespace game\_framework

{

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

// 這個class提供地圖構成

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

MapS3::MapS3() : Map(0, 0) //地圖設置：0為不能走、1為可以走、2為傳送門、3為橋、4為鍛造台、5為垃圾桶、6為受傷區域

{

X = 20;

Y = 20;

gridW = 20;

gridH = 20;

startX = 200;

finalX = 4350;

setFloor(610);

setCeiling(0);

setMonsterFloor(610);

int mapGrid\_init[400][70];

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 1;

}

}

//給予地圖左上角座標及每張小圖寬度

for (int i = 0; i < 19; i++)

{

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

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 3; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 91; i++)

{

for (int j = 31; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 2; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 2; i < 12; i++)

{

for (int j = 18; j < 31; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 27; i < 91; i++)

{

for (int j = 25; j < 31; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 63; i < 400; i++)

{

for (int j = 0; j < 11; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 91; i < 400; i++)

{

for (int j = 33; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 152; i < 163; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 5;

}

}

for (int i = 190; i < 203; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 4;

}

}

for (int i = 210; i < 230; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 230; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid[i][j] = mapGrid\_init[i][j]; //依序填入mapGrid

}

}

}

MapS3::~MapS3()

{

}

void MapS3::LoadBitmap()

{

map.LoadBitmap(".\\res\\map03.bmp");

exitBitmap.LoadBitmap(IDB\_EXIT, RGB(0, 0, 0));

pressEBitmap.LoadBitmap(IDB\_PRESSE, RGB(0, 0, 0));

trashCanClosedBitmap.LoadBitmap(IDB\_TRASHCANCLOSED, RGB(0, 0, 0));

trashCanOpenBitmap.LoadBitmap(IDB\_TRASHCANOPEN, RGB(0, 0, 0));

craftingBitmap.LoadBitmap(IDB\_CRAFTING, RGB(0, 0, 0));

loadMagnifierBitmap();

loadMessageIconBitmap();

message01.LoadBitmap(".\\res\\message0301.bmp", RGB(0, 0, 0));

message02.LoadBitmap(".\\res\\message0302.bmp", RGB(0, 0, 0));

message03.LoadBitmap(".\\res\\message0303.bmp", RGB(0, 0, 0));

message04.LoadBitmap(".\\res\\message0304.bmp", RGB(0, 0, 0));

message05.LoadBitmap(".\\res\\message0305.bmp", RGB(0, 0, 0));

message06.LoadBitmap(".\\res\\message0306.bmp", RGB(0, 0, 0));

}

void MapS3::Initialize()

{

setXY(0, -260);

setInitialXY(0, -260);

setScreenMoving(true);

setMapName("MapS3");

setCeiling(0);

previousFloor = 0;

trashCanOpen = false;

screenUp = screenDown = false;

messageCode = 0;

canShowMessage = false;

autoShowMessage = false;

havePlayedAutoMessage = false;

showMaginifierFlag = false;

showMessageIconFlag = false;

messageCounter = 1;

messageSize = 6;

messageEndFlag = false;

}

void MapS3::setPos(int x, int y, int n)

{

int gridX = x / 20;

int gridY = y / 20;

mapGrid[gridX][gridY] = n;

}

bool MapS3::isEmpty(int x, int y) const

{

return (blockProperty(x, y) != 0);

}

bool MapS3::isPortal(int x, int y) const

{

return (blockProperty(x, y) == 2);

}

bool MapS3::isBridge(int x, int y) const

{

return (blockProperty(x, y) == 3);

}

bool MapS3::isCraftTable(int x, int y) const

{

return (blockProperty(x, y) == 4);

}

bool MapS3::isTrashCan(int x, int y) const

{

return (blockProperty(x, y) == 5);

}

bool MapS3::isGetHurtPlace(int x, int y) const

{

return false;

}

int MapS3::blockProperty(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

return mapGrid[gridX][gridY];

}

void MapS3::onShow()

{

map.SetTopLeft(getSX(), getSY());

map.ShowBitmap();

if (!getTrashCanOpen())

{

trashCanClosedBitmap.SetTopLeft(3025 + getSX(), getFloor() - 132);

trashCanClosedBitmap.ShowBitmap();

}

else

{

trashCanOpenBitmap.SetTopLeft(3025 + getSX(), getFloor() - 180);

trashCanOpenBitmap.ShowBitmap();

}

if (getShowMessageIconFlag())

{

showMessageIcon(3410, 705);

pressEBitmap.SetTopLeft(600, 940 + getSY());

pressEBitmap.ShowBitmap();

}

if (getPortalOpen())

{

if (characterX < 500)

exitBitmap.SetTopLeft(100 + getSX(), 370);

else

exitBitmap.SetTopLeft(4345 + getSX(), 400);

pressEBitmap.SetTopLeft(600, 940 + getSY());

exitBitmap.ShowBitmap();

pressEBitmap.ShowBitmap();

}

if (getCraftTableOpen())

{

craftingBitmap.SetTopLeft(3880 + getSX(), 280);

craftingBitmap.ShowBitmap();

pressEBitmap.SetTopLeft(600, 940 + getSY());

pressEBitmap.ShowBitmap();

}

}

void MapS3::monsterFloorChanging(int x)

{

if (x < 502)

setMonsterFloor(610);

else if (x < 1840)

setMonsterFloor(485);

else

setMonsterFloor(655);

}

void MapS3::characterFloorAndCeiling()

{

if (characterX < 502)

setFloor(610);

else if (characterX < 1838)

setFloor(485);

else

setFloor(655);

if (characterX < 370)

setCeiling(360);

else if (characterX < 1220)

setCeiling(60);

else if (characterX < 1930)

setCeiling(230);

else

setCeiling(255);

}

int MapS3::screenX(int x)

{

return x + getSX();

}

int MapS3::screenY(int y)

{

return y + getSY();

}

void MapS3::messageOnShow()

{

int mX = 136, mY = 201;

if (messageCounter == 1)

{

message01.SetTopLeft(mX, mY);

message01.ShowBitmap();

}

else if (messageCounter == 2)

{

message02.SetTopLeft(mX, mY);

message02.ShowBitmap();

}

else if (messageCounter == 3)

{

message03.SetTopLeft(mX, mY);

message03.ShowBitmap();

}

else if (messageCounter == 4)

{

message04.SetTopLeft(mX, mY);

message04.ShowBitmap();

}

else if (messageCounter == 5)

{

message05.SetTopLeft(mX, mY);

message05.ShowBitmap();

}

else if (messageCounter == 6)

{

message06.SetTopLeft(mX, mY);

message06.ShowBitmap();

messageEndFlag = true;

}

}

}

===============================

MapS4.h

===============================

#ifndef \_\_MapS4\_\_

#define \_\_MAPS4\_\_

namespace game\_framework

{

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

// 這個class提供地圖構成

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

class MapS4 : public Map

{

public:

MapS4();

~MapS4();

void LoadBitmap();

void Initialize();

void setPos(int x, int y, int n); //將地圖座標的陣列設為傳入的數字

bool isEmpty(int x, int y) const;

bool isPortal(int x, int y) const;

bool isBridge(int x, int y) const;

bool isCraftTable(int x, int y) const;

bool isTrashCan(int x, int y) const;

bool isGetHurtPlace(int x, int y) const;

int blockProperty(int x, int y) const;

void onShow();

void syMoving(int y);

void monsterFloorChanging(int x) override;

void characterFloorAndCeiling() override;

int screenX(int x) override;

int screenY(int y) override;

void messageOnShow() override;

protected:

CMovingBitmap map;

int X, Y;

int mapGrid[400][70];

CMovingBitmap white, blue;

int gridW, gridH;

int previousFloor;

int targetSY;

bool screenUp;

bool screenDown;

CMovingBitmap message01;

CMovingBitmap message02;

CMovingBitmap message03;

CMovingBitmap message04;

CMovingBitmap message05;

CMovingBitmap message06;

};

}

#endif

===============================

MapS4.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "MapS4.h"

#include <vector>

namespace game\_framework

{

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

// 這個class提供地圖構成

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

MapS4::MapS4() : Map(0, 0) //地圖設置：0為不能走、1為可以走、2為傳送門、3為橋、4為鍛造台、5為垃圾桶、6為受傷區域

{

X = 20;

Y = 20;

gridW = 20;

gridH = 20;

startX = 200;

finalX = 3250;

setFloor(540);

setCeiling(0);

setMonsterFloor(540);

int mapGrid\_init[400][70];

for (int i = 1; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 1;

}

}

//給予地圖左上角座標及每張小圖寬度

for (int i = 0; i < 400; i++)

{

for (int j = 37; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 5; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 14; i++)

{

for (int j = 5; j < 37; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 35; i < 400; i++)

{

for (int j = 5; j < 14; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 38; i++)

{

for (int j = 27; j < 32; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 72; i++)

{

for (int j = 32; j < 37; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 37; i < 54; i++)

{

mapGrid\_init[i][30] = 6;

mapGrid\_init[i][31] = 6;

}

for (int i = 56; i < 65; i++)

{

for (int j = 26; j < 32; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 79; i < 88; i++)

{

for (int j = 14; j < 28; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 92; i < 97; i++)

{

mapGrid\_init[i][33] = 3;

}

for (int i = 98; i < 113; i++)

{

for (int j = 31; j < 37; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 97; i < 113; i++)

{

mapGrid\_init[i][29] = 6;

mapGrid\_init[i][30] = 6;

}

for (int i = 101; i < 107; i++)

{

mapGrid\_init[i][27] = 3;

}

for (int i = 117; i < 400; i++)

{

for (int j = 14; j < 21; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 155; i < 176; i++)

{

for (int j = 21; j < 37; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 176; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid[i][j] = mapGrid\_init[i][j]; //依序填入mapGrid

}

}

}

MapS4::~MapS4()

{

}

void MapS4::LoadBitmap()

{

map.LoadBitmap(".\\res\\map04.bmp");

exitBitmap.LoadBitmap(IDB\_EXIT, RGB(0, 0, 0));

pressEBitmap.LoadBitmap(IDB\_PRESSE, RGB(0, 0, 0));

trashCanClosedBitmap.LoadBitmap(IDB\_TRASHCANCLOSED, RGB(0, 0, 0));

trashCanOpenBitmap.LoadBitmap(IDB\_TRASHCANOPEN, RGB(0, 0, 0));

craftingBitmap.LoadBitmap(IDB\_CRAFTING, RGB(0, 0, 0));

loadMagnifierBitmap();

}

void MapS4::Initialize()

{

setXY(0, -270);

setInitialXY(0, -270);

setScreenMoving(true);

setMapName("MapS4");

setCeiling(0);

previousFloor = 0;

targetSY = 0;

trashCanOpen = false;

screenUp = screenDown = false;

messageCode = 0;

canShowMessage = false;

autoShowMessage = false;

havePlayedAutoMessage = false;

showMaginifierFlag = false;

showMessageIconFlag = false;

messageCounter = 1;

messageSize = 6;

messageEndFlag = false;

}

void MapS4::setPos(int x, int y, int n)

{

int gridX = x / 20;

int gridY = y / 20;

mapGrid[gridX][gridY] = n;

}

bool MapS4::isEmpty(int x, int y) const

{

return (blockProperty(x, y) != 0);

}

bool MapS4::isPortal(int x, int y) const

{

return (blockProperty(x, y) == 2);

}

bool MapS4::isBridge(int x, int y) const

{

return (blockProperty(x, y) == 3);

}

bool MapS4::isCraftTable(int x, int y) const

{

return (blockProperty(x, y) == 4);

}

bool MapS4::isTrashCan(int x, int y) const

{

return (blockProperty(x, y) == 5);

}

bool MapS4::isGetHurtPlace(int x, int y) const

{

return (blockProperty(x, y) == 6);

}

int MapS4::blockProperty(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

return mapGrid[gridX][gridY];

}

void MapS4::onShow()

{

map.SetTopLeft(getSX(), getSY());

map.ShowBitmap();

syMoving(targetSY);

if (getPortalOpen())

{

if (characterX < 500)

exitBitmap.SetTopLeft(80 + getSX(), 230);

else

exitBitmap.SetTopLeft(3240 + getSX(), 250);

pressEBitmap.SetTopLeft(600, 610);

exitBitmap.ShowBitmap();

pressEBitmap.ShowBitmap();

}

}

void MapS4::syMoving(int y)

{

if (screenUp)

{

if (getSY() - 15 > y)

{

addSY(-15);

}

else

{

addSY(y - getSY());

screenUp = false;

}

}

else if (screenDown)

{

if (getSY() + 15 < y)

{

addSY(15);

}

else

{

addSY(y - getSY());

screenDown = false;

}

}

}

void MapS4::monsterFloorChanging(int x)

{

if (x < 750)

setMonsterFloor(540);

else if (x < 1120)

setMonsterFloor(630);

else if (x < 1315)

setMonsterFloor(485);

else if (x < 1447)

setMonsterFloor(630);

else if (x < 1925)

setMonsterFloor(725);

else if (x < 2265)

setMonsterFloor(605);

else

setMonsterFloor(725);

}

void MapS4::characterFloorAndCeiling()

{

if (characterX < 750)

setFloor(540);

else if (characterX < 1090)

setFloor(630);

else if (characterX < 1315)

{

setFloor(485);

if (getSY() != -270)

{

screenDown = true;

targetSY = -270;

}

}

else if (characterX < 1447)

setFloor(630);

else if (characterX < 1925)

{

if (getSY() != -440)

{

screenUp = true;

targetSY = -440;

}

if (characterX > 1850 && characterY+120 < 674)

setFloor(665);

else

setFloor(725);

}

else if (characterX < 2265)

{

if (characterX > 2035 && characterX < 2115 && characterY + 120 < 554)

setFloor(545);

else

setFloor(605);

}

else

setFloor(725);

if (characterX < 650)

setCeiling(100);

else if (characterX < 1535)

setCeiling(290);

else if (characterX < 1780)

setCeiling(570);

else if (characterX < 2335)

setCeiling(340);

else

setCeiling(405);

}

int MapS4::screenX(int x)

{

return x + getSX();

}

int MapS4::screenY(int y)

{

return y + getSY();

}

void MapS4::messageOnShow()

{

int mX = 136, mY = 201;

if (messageCounter == 1)

{

message01.SetTopLeft(mX, mY);

message01.ShowBitmap();

}

else if (messageCounter == 2)

{

message02.SetTopLeft(mX, mY);

message02.ShowBitmap();

}

else if (messageCounter == 3)

{

message03.SetTopLeft(mX, mY);

message03.ShowBitmap();

}

else if (messageCounter == 4)

{

message04.SetTopLeft(mX, mY);

message04.ShowBitmap();

}

else if (messageCounter == 5)

{

message05.SetTopLeft(mX, mY);

message05.ShowBitmap();

}

else if (messageCounter == 6)

{

message06.SetTopLeft(mX, mY);

message06.ShowBitmap();

messageEndFlag = true;

}

}

}

===============================

MapS5.h

===============================

#ifndef \_\_MapS5\_\_

#define \_\_MapS5\_\_

namespace game\_framework

{

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

// 這個class提供地圖構成

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

class MapS5 : public Map

{

public:

MapS5();

~MapS5();

void LoadBitmap();

void Initialize();

void setPos(int x, int y, int n); //將地圖座標的陣列設為傳入的數字

bool isEmpty(int x, int y) const;

bool isPortal(int x, int y) const;

bool isBridge(int x, int y) const;

bool isCraftTable(int x, int y) const;

bool isTrashCan(int x, int y) const;

bool isGetHurtPlace(int x, int y) const;

void onShow();

void syMoving(int y);

void monsterFloorChanging(int x) override;

void characterFloorAndCeiling() override;

int screenX(int x) override;

int screenY(int y) override;

void messageOnShow() override;

protected:

CMovingBitmap map;

int X, Y;

int mapGrid[400][70];

CMovingBitmap white, blue;

int gridW, gridH;

int previousFloor;

int targetSY;

bool screenUp;

bool screenDown;

CMovingBitmap message01;

CMovingBitmap message02;

CMovingBitmap message03;

CMovingBitmap message04;

};

}

#endif

===============================

MapS5.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "MapS5.h"

#include <vector>

namespace game\_framework

{

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

// 這個class提供地圖構成

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

MapS5::MapS5() : Map(0, 0) //地圖設置：0為不能走、1為可以走、2為傳送門、3為橋、4為鍛造台、5為垃圾桶、6為受傷區域

{

X = 20;

Y = 20;

gridW = 20;

gridH = 20;

startX = 300;

finalX = 3250;

setFloor(517);

setCeiling(0);

setMonsterFloor(0);

int mapGrid\_init[400][70];

for (int i = 2; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 1;

}

}

//給予地圖左上角座標及每張小圖寬度

for (int i = 0; i < 400; i++)

{

for (int j = 66; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 43; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 3; i < 28; i++)

{

for (int j = 43; j < 64; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 47; i < 92; i++)

{

for (int j = 61; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 48; i < 59; i++)

{

for (int j = 43; j < 61; j++)

{

mapGrid\_init[i][j] = 5;

}

}

for (int i = 66; i < 89; i++)

{

for (int j = 43; j < 52; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 58; i < 90; i++)

{

for (int j = 13; j < 32; j++)

{

mapGrid\_init[i][j] = 1;

}

}

for (int i = 58; i < 70; i++)

{

for (int j = 13; j < 32; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 90; i < 99; i++)

{

for (int j = 13; j < 43; j++)

{

mapGrid\_init[i][j] = 1;

}

}

for (int i = 90; i < 97; i++)

{

mapGrid\_init[i][40] = 3;

mapGrid\_init[i][47] = 3;

}

for (int i = 99; i < 400; i++)

{

for (int j = 43; j < 48; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 132; i < 400; i++)

{

for (int j = 61; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 137; i < 148; i++)

{

for (int j = 49; j < 61; j++)

{

mapGrid\_init[i][j] = 4;

}

}

for (int i = 157; i < 179; i++)

{

for (int j = 49; j < 61; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 179; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid[i][j] = mapGrid\_init[i][j]; //依序填入mapGrid

}

}

}

MapS5::~MapS5()

{

}

void MapS5::LoadBitmap()

{

map.LoadBitmap(".\\res\\map05.bmp");

exitBitmap.LoadBitmap(IDB\_EXIT, RGB(0, 0, 0));

pressEBitmap.LoadBitmap(IDB\_PRESSE, RGB(0, 0, 0));

craftingBitmap.LoadBitmap(IDB\_CRAFTING, RGB(0, 0, 0));

trashCanClosedBitmap.LoadBitmap(IDB\_TRASHCANCLOSED, RGB(0, 0, 0));

trashCanOpenBitmap.LoadBitmap(IDB\_TRASHCANOPEN, RGB(0, 0, 0));

loadMagnifierBitmap();

loadMagnifierLBitmap();

loadMessageIconBitmap();

message01.LoadBitmap(".\\res\\message0501.bmp");

message02.LoadBitmap(".\\res\\message0502.bmp");

message03.LoadBitmap(".\\res\\message0503.bmp");

message04.LoadBitmap(".\\res\\message0504.bmp");

}

void MapS5::Initialize()

{

setXY(65, -800);

setInitialXY(65, 0);

setScreenMoving(true);

setMapName("MapS5");

setCeiling(0);

trashCanOpen = false;

previousFloor = 0;

targetSY = 0;

screenUp = screenDown = false;

messageCode = 0;

canShowMessage = false;

autoShowMessage = false;

havePlayedAutoMessage = false;

showMaginifierFlag = false;

showMessageIconFlag = false;

messageCounter = 1;

messageSize = 4;

messageEndFlag = false;

}

void MapS5::setPos(int x, int y, int n)

{

int gridX = x / 20;

int gridY = y / 20;

mapGrid[gridX][gridY] = n;

}

bool MapS5::isEmpty(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] != 0)

{

return true;

}

return false;

}

bool MapS5::isPortal(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] == 2)

{

return true;

}

return false;

}

bool MapS5::isBridge(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] == 3)

{

return true;

}

return false;

}

bool MapS5::isCraftTable(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] == 4)

{

return true;

}

return false;

}

bool MapS5::isTrashCan(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

if (mapGrid[gridX][gridY] == 5)

{

return true;

}

return false;

}

bool MapS5::isGetHurtPlace(int x, int y) const

{

return false;

}

void MapS5::onShow()

{

map.SetTopLeft(getSX(), getSY());

map.ShowBitmap();

syMoving(targetSY);

if (getShowMaginifierFlag())

{

showMaginifierL(1458, 494);

pressEBitmap.SetTopLeft(600, 670 + getSY());

pressEBitmap.ShowBitmap();

}

if (getPortalOpen())

{

if (characterX > 1580)

{

exitBitmap.SetTopLeft(3170 + getSX(), 120);

exitBitmap.ShowBitmap();

}

else if (characterX > 1000)

{

exitBitmap.SetTopLeft(1155 + getSX(), 250);

exitBitmap.ShowBitmap();

}

else

{

exitBitmap.SetTopLeft(230, 250);

exitBitmap.ShowBitmap();

}

pressEBitmap.SetTopLeft(600, 585);

pressEBitmap.ShowBitmap();

}

if (getCraftTableOpen())

{

craftingBitmap.SetTopLeft(2760 + getSX(), 175);

craftingBitmap.ShowBitmap();

pressEBitmap.SetTopLeft(600, 585);

pressEBitmap.ShowBitmap();

}

if (!getTrashCanOpen())

{

trashCanClosedBitmap.SetTopLeft(860 + getSX(), 1217 - 132 + getSY());

trashCanClosedBitmap.ShowBitmap();

}

else

{

trashCanOpenBitmap.SetTopLeft(860 + getSX(), 1217 - 180 + getSY());

trashCanOpenBitmap.ShowBitmap();

}

}

void MapS5::syMoving(int y)

{

if (screenUp)

{

if (getSY() - 20 > y)

{

addSY(-20);

}

else

{

screenUp = false;

addSY(y - getSY());

}

}

else if (screenDown)

{

if (getSY() + 20 < y)

{

addSY(20);

}

else

{

screenDown = false;

addSY(y - getSY());

}

}

}

void MapS5::monsterFloorChanging(int x)

{

if (x < 920)

setMonsterFloor(1317);

else if (x < 1875)

setMonsterFloor(1217);

else if (x < 2625)

setMonsterFloor(1290);

else

setMonsterFloor(1217);

}

void MapS5::characterFloorAndCeiling()

{

if (characterX > 1000 && characterX < 1840 && characterY + 120 < 649)

setFloor(640);

else if (characterX < 920)

setFloor(1317);

else if (characterX < 1875)

{

if (characterX > 1810 && characterY + 120 < 822)

{

setFloor(813);

if (getSY() != -100)

{

screenDown = true;

targetSY = -100;

}

previousFloor = 813;

}

else if (characterX > 1810 && characterY + 120 < 954)

{

setFloor(945);

if (getSY() != -450)

{

if (previousFloor == 1217)

screenDown = true;

else

screenUp = true;

targetSY = -450;

}

}

else

{

setFloor(1217);

if (getSY() != -800)

{

screenUp = true;

targetSY = -800;

}

previousFloor = 1217;

}

}

else if (characterX < 2625)

{

if (characterX < 1910 && characterY + 120 < 822)

{

setFloor(813);

if (getSY() != -100)

{

screenDown = true;

targetSY = -100;

}

previousFloor = 945;

}

else

{

setFloor(1287);

if (getSY() != -800)

{

screenUp = true;

targetSY = -800;

}

}

}

else

setFloor(1217);

if (characterX < 1290 && characterY > 700)

setCeiling(875);

else if (characterX < 1820 && characterY > 700)

setCeiling(1040);

else if (characterX < 1980)

setCeiling(260);

else

setCeiling(975);

}

int MapS5::screenX(int x)

{

return x + getSX();

}

int MapS5::screenY(int y)

{

return y + getSY();

}

void MapS5::messageOnShow()

{

if (messageCounter == 1)

{

message01.SetTopLeft(173, 535);

message01.ShowBitmap();

}

else if (messageCounter == 2)

{

message02.SetTopLeft(173, 535);

message02.ShowBitmap();

}

else if (messageCounter == 3)

{

message03.SetTopLeft(173, 535);

message03.ShowBitmap();

}

else if (messageCounter == 4)

{

message04.SetTopLeft(173, 535);

message04.ShowBitmap();

messageEndFlag = true;

}

}

}

===============================

MapS6.h

===============================

#ifndef \_\_MapS6\_\_

#define \_\_MapS6\_\_

namespace game\_framework

{

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

// 這個class提供地圖構成

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

class MapS6 : public Map

{

public:

MapS6();

~MapS6();

void LoadBitmap();

void Initialize();

void setPos(int x, int y, int n); //將地圖座標的陣列設為傳入的數字

bool isEmpty(int x, int y) const;

bool isPortal(int x, int y) const;

bool isBridge(int x, int y) const;

bool isCraftTable(int x, int y) const;

bool isTrashCan(int x, int y) const;

bool isGetHurtPlace(int x, int y) const;

int blockProperty(int x, int y) const;

void onShow();

void syMoving(int y);

void monsterFloorChanging(int x) override;

void characterFloorAndCeiling() override;

int screenX(int x) override;

int screenY(int y) override;

void messageOnShow() override;

void setBossAssistantExist(bool flag);

bool getBossAssistantExist();

protected:

CMovingBitmap map;

int X, Y;

int mapGrid[400][70];

CMovingBitmap white, blue;

int gridW, gridH;

int previousFloor;

int targetSY;

bool screenUp;

bool screenDown;

CMovingBitmap bossAssistant;

bool bossAssistantExist;

CMovingBitmap message01;

CMovingBitmap message02;

CMovingBitmap message03;

CMovingBitmap message04;

};

}

#endif

===============================

MapS6.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "MapS6.h"

#include <vector>

namespace game\_framework

{

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

// 這個class提供地圖構成

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

MapS6::MapS6() : Map(0, 0) //地圖設置：0為不能走、1為可以走、2為傳送門、3為橋、4為鍛造台、5為垃圾桶、6為受傷區域

{

X = 20;

Y = 20;

gridW = 20;

gridH = 20;

startX = 300;

setFloor(600);

setCeiling(0);

setMonsterFloor(600);

int mapGrid\_init[400][70];

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 1;

}

}

//給予地圖左上角座標及每張小圖寬度

for (int i = 0; i < 6; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 7; i < 23; i++)

{

for (int j = 21; j < 38; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 38; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 6; i < 22; i++)

{

for (int j = 16; j < 21; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

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

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 82; i++)

{

for (int j = 4; j < 16; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 105; i < 400; i++)

{

for (int j = 29; j < 38; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 145; i < 153; i++)

{

for (int j = 4; j < 29; j++)

{

mapGrid\_init[i][j] = 5;

}

}

for (int i = 160; i < 170; i++)

{

for (int j = 4; j < 29; j++)

{

mapGrid\_init[i][j] = 4;

}

}

for (int i = 180; i < 200; i++)

{

for (int j = 4; j < 29; j++)

{

mapGrid\_init[i][j] = 2;

}

}

for (int i = 200; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid\_init[i][j] = 0;

}

}

for (int i = 0; i < 400; i++)

{

for (int j = 0; j < 70; j++)

{

mapGrid[i][j] = mapGrid\_init[i][j]; //依序填入mapGrid

}

}

}

MapS6::~MapS6()

{

}

void MapS6::LoadBitmap()

{

map.LoadBitmap(".\\res\\map06.bmp");

exitBitmap.LoadBitmap(IDB\_EXIT, RGB(0, 0, 0));

pressEBitmap.LoadBitmap(IDB\_PRESSE, RGB(0, 0, 0));

trashCanClosedBitmap.LoadBitmap(IDB\_TRASHCANCLOSED, RGB(0, 0, 0));

trashCanOpenBitmap.LoadBitmap(IDB\_TRASHCANOPEN, RGB(0, 0, 0));

craftingBitmap.LoadBitmap(IDB\_CRAFTING, RGB(0, 0, 0));

bossAssistant.LoadBitmap(".\\res\\boss\_assistant.bmp", RGB(0, 0, 0));

message01.LoadBitmap(".\\res\\message0601.bmp", RGB(0, 0, 0));

message02.LoadBitmap(".\\res\\message0602.bmp", RGB(0, 0, 0));

message03.LoadBitmap(".\\res\\message0603.bmp", RGB(0, 0, 0));

message04.LoadBitmap(".\\res\\message0604.bmp", RGB(0, 0, 0));

}

void MapS6::Initialize()

{

setXY(100, -260);

setInitialXY(100, -100);

setScreenMoving(true);

setMapName("MapS6");

setCeiling(0);

previousFloor = 0;

targetSY = 0;

trashCanOpen = false;

screenUp = screenDown = false;

messageCode = 0;

canShowMessage = false;

autoShowMessage = false;

havePlayedAutoMessage = false;

showMaginifierFlag = false;

showMessageIconFlag = false;

messageCounter = 1;

messageSize = 4;

messageEndFlag = false;

}

void MapS6::setPos(int x, int y, int n)

{

int gridX = x / 20;

int gridY = y / 20;

mapGrid[gridX][gridY] = n;

}

bool MapS6::isEmpty(int x, int y) const

{

return (blockProperty(x, y) != 0);

}

bool MapS6::isPortal(int x, int y) const

{

return (blockProperty(x, y) == 2);

}

bool MapS6::isBridge(int x, int y) const

{

return (blockProperty(x, y) == 3);

}

bool MapS6::isCraftTable(int x, int y) const

{

return (blockProperty(x, y) == 4);

}

bool MapS6::isTrashCan(int x, int y) const

{

return (blockProperty(x, y) == 5);

}

bool MapS6::isGetHurtPlace(int x, int y) const

{

return false;

}

int MapS6::blockProperty(int x, int y) const

{

int gridX = x / 20;

int gridY = y / 20;

return mapGrid[gridX][gridY];

}

void MapS6::onShow()

{

map.SetTopLeft(getSX(), getSY());

map.ShowBitmap();

if (bossAssistantExist)

{

bossAssistant.SetTopLeft(3616 + getSX(), 458 + getSY());

bossAssistant.ShowBitmap();

}

syMoving(targetSY);

if (getPortalOpen())

{

if (characterX < 500)

{

exitBitmap.SetTopLeft(143 + getSX(), 280);

pressEBitmap.SetTopLeft(600, 655);

}

else

{

exitBitmap.SetTopLeft(3614 + getSX(), 255);

pressEBitmap.SetTopLeft(600, 625);

}

exitBitmap.ShowBitmap();

pressEBitmap.ShowBitmap();

}

if (getCraftTableOpen())

{

craftingBitmap.SetTopLeft(3165 + getSX(), 280);

craftingBitmap.ShowBitmap();

pressEBitmap.SetTopLeft(600, 625);

pressEBitmap.ShowBitmap();

}

if (!getTrashCanOpen())

{

trashCanClosedBitmap.SetTopLeft(2770 + getSX(), 664 - 132 + getSY());

trashCanClosedBitmap.ShowBitmap();

}

else

{

trashCanOpenBitmap.SetTopLeft(2770 + getSX(), 664 - 180 + getSY());

trashCanOpenBitmap.ShowBitmap();

}

}

void MapS6::syMoving(int y)

{

if (screenUp)

{

if (getSY() - 15 > y)

{

addSY(-15);

}

else

{

addSY(y - getSY());

screenUp = false;

}

}

else if (screenDown)

{

if (getSY() + 15 < y)

{

addSY(15);

}

else

{

screenDown = false;

addSY(y - getSY());

}

}

}

void MapS6::monsterFloorChanging(int x)

{

if (x < 895)

setMonsterFloor(760);

else if (x < 2076)

setMonsterFloor(744);

else

setMonsterFloor(566);

}

void MapS6::characterFloorAndCeiling()

{

if (characterX < 895)

setFloor(760);

else if (characterX < 2076)

{

setFloor(744);

if (getSY() != -260)

{

screenUp = true;

targetSY = -260;

}

}

else

{

setFloor(566);

if (getSY() != -100)

{

screenDown = true;

targetSY = -100;

}

}

if (characterX < 460)

setCeiling(420);

else if (characterX < 1660)

setCeiling(335);

else

setCeiling(100);

}

int MapS6::screenX(int x)

{

return x + getSX();

}

int MapS6::screenY(int y)

{

return y + getSY();

}

void MapS6::messageOnShow()

{

if (messageCounter == 1)

{

message01.SetTopLeft(173, 120);

message01.ShowBitmap();

}

else if (messageCounter == 2)

{

message02.SetTopLeft(173, 120);

message02.ShowBitmap();

}

else if (messageCounter == 3)

{

message03.SetTopLeft(173, 120);

message03.ShowBitmap();

}

else if (messageCounter == 4)

{

message04.SetTopLeft(173, 120);

message04.ShowBitmap();

messageEndFlag = true;

}

}

void MapS6::setBossAssistantExist(bool flag)

{

bossAssistantExist = flag;

}

bool MapS6::getBossAssistantExist()

{

return bossAssistantExist;

}

}

===============================

MapForBoss.h

===============================

namespace game\_framework {

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

// 這個class提供地圖構成

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

class MapForBoss : public Map {

public:

MapForBoss();

~MapForBoss();

void LoadBitmap();

void Initialize();

void setPos(int x, int y, int n); //將地圖座標的陣列設為傳入的數字

bool isEmpty(int x, int y) const;

bool isPortal(int x, int y) const;

bool isBridge(int x, int y) const;

bool isCraftTable(int x, int y) const;

bool isTrashCan(int x, int y) const;

bool isGetHurtPlace(int x, int y) const;

void onShow();

void monsterFloorChanging(int x);

int screenX(int x) override;

int screenY(int y) override;

protected:

CMovingBitmap map;

int X, Y;

int mapGrid[40][18];

CMovingBitmap white, blue;

int gridW, gridH;

};

}

===============================

MapForBoss.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "MapForBoss.h"

#include <vector>

namespace game\_framework {

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

// 這個class提供地圖構成

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

MapForBoss::MapForBoss() : Map(0, 0) //地圖設置：0為不能走、1為可以走、2為傳送門、3為橋、4為鍛造台、5為垃圾桶、6為受傷區域

{

X = 0;

Y = 0;

gridW = 40;

gridH = 40;

setFloor(600);

setMonsterFloor(560);

int mapGrid\_init[40][18];

for (int i = 0; i < 40; i++)

{

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

{

mapGrid\_init[i][j] = 0;

}

}

//給予地圖左上角座標及每張小圖寬度

for (int i = 2; i < 30; i++)

{

for (int j = 5; j < 16; j++)

{

mapGrid\_init[i][j] = 1;

}

}

for (int i = 0; i < 40; i++) {

for (int j = 0; j < 18; j++) {

mapGrid[i][j] = mapGrid\_init[i][j]; //依序填入mapGrid

}

}

}

MapForBoss::~MapForBoss()

{

}

void MapForBoss::LoadBitmap()

{

map.LoadBitmap(".\\res\\boss\_map.bmp");

}

void MapForBoss::Initialize()

{

setXY(0, 0);

setInitialXY(0, 0);

setScreenMoving(false);

setMapName("MapForBoss");

startX = 100;

messageCode = 0;

canShowMessage = false;

autoShowMessage = false;

havePlayedAutoMessage = false;

showMaginifierFlag = false;

showMessageIconFlag = false;

messageCounter = 1;

messageSize = 0;

messageEndFlag = false;

}

void MapForBoss::setPos(int x, int y, int n)

{

int gridX = x / 40;

int gridY = y / 40;

mapGrid[gridX][gridY] = n;

}

bool MapForBoss::isEmpty(int x, int y) const

{

int gridX = x / 40;

int gridY = y / 40;

if (mapGrid[gridX][gridY] != 0)

{

return true;

}

return false;

}

bool MapForBoss::isPortal(int x, int y) const

{

int gridX = x / 40;

int gridY = y / 40;

if (mapGrid[gridX][gridY] == 2)

{

return true;

}

return false;

}

bool MapForBoss::isBridge(int x, int y) const

{

int gridX = x / 40;

int gridY = y / 40;

if (mapGrid[gridX][gridY] == 3)

{

return true;

}

return false;

}

bool MapForBoss::isCraftTable(int x, int y) const

{

return false;

}

bool MapForBoss::isTrashCan(int x, int y) const

{

return false;

}

bool MapForBoss::isGetHurtPlace(int x, int y) const

{

return false;

}

void MapForBoss::onShow() {

map.SetTopLeft(getSX(), getSY() + 20);

map.ShowBitmap();

}

void MapForBoss::monsterFloorChanging(int x)

{

setMonsterFloor(600);

}

int MapForBoss::screenX(int x)

{

return x + getSX();

}

int MapForBoss::screenY(int y)

{

return y + getSY();

}

}

============================================================

Monster.h

============================================================

#ifndef \_\_MONSTER\_\_

#define \_\_MONSTER\_\_

#include "BloodBar.h"

#include "Timer.h"

namespace game\_framework

{

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

// 這個class是魔物的介面 //

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

class Character;

class Map;

class BloodBar;

class SourceStorage;

class Monster

{

public:

Monster();

Monster::Monster(int x, int y, int enemyHp, int attackDamage, Character\* c);

virtual ~Monster();

virtual void LoadBitmap() = 0;//加載敵人Bitmap

virtual void Initialize() = 0; //初始化

virtual void OnShow(Map\* m) = 0;//顯示敵人於畫面上

virtual void OnMove(Map\* m) = 0;//敵人移動

virtual int GetLeftX() = 0; //左上角 x 座標

virtual int GetTopY() = 0; //左上角 y 座標

virtual int GetRightX() = 0;//右下角 x 座標

virtual int GetBottomY() = 0;//右下角 y 座標

//英雄是否和此敵人碰觸

virtual void intersect();

void SetIsIntersect(bool flag);

bool GetIsIntersect();

double hpProportion();

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

// 設定Monster的座標的函式們 //

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

void SetXY(int x, int y); // x, y座標

int GetBorder();

bool GetCanStandOn();

void SetMovingDown(bool b); //下

bool GetIsMovingDown(); // 回傳是否正在往上移動

void SetMovingUp(bool b); //上

bool GetIsMovingUp(); // 回傳是否正在往上移動

void SetMovingLeft(bool b); //左

bool GetIsMovingLeft(); // 回傳是否正在往左移動

void SetMovingRight(bool b); //右

bool GetIsMovingRight(); // 回傳是否正在往右移動

void SetFacingLR(bool b); //右

bool GetFacingLR(); // 回傳是否正在往右移動

int distanceToCharacter();

void SetCharacterDirection();//設定角色相對怪物的位置(0:角色在怪物左邊或下面)

bool GetCharacterDirectionLR();//回傳角色相對怪物的位置

bool GetCharacterDirectionTD();//回傳角色相對怪物的位置

bool isAlive();

bool isAttack();

void knockBack();

void SetIsOnTheFloor(bool b);

bool GetIsOnTheFloor();//回傳是否正在地面

//血量

void SetCurrentHp(int x);//設定目前血量

int GetCurrentHp();//取得目前血量

void lossCurrentHp(int x);

void lossHpShow(Map\* m);

//攻擊

void SetIsAttacking(bool flag); //設定是否攻擊

bool GetIsAttacking();//回傳是否正在攻擊

void SetAttackDamage(int x);

int GetAttackDamage();

virtual void attackStart();

virtual void attackJudge(int attackRange);

void SetIsAttacked(bool flag);//設定是否受到攻擊

bool GetIsAttacked();//回傳是否受到攻擊

void isAttackedEffectCaculation();

void isAttackedEffectOnShow(Map\* m);

//狂暴化

void SetIsViolent(bool flag); //設定是否受到攻擊

bool GetIsViolent();//回傳是否受到攻擊

//結束

void SetBossDead(bool flag);

bool GetBossDead();

protected:

bool isAttackSuccessfullyL(int range);

bool isAttackSuccessfullyR(int range);

void touchSource(Map\* m, int sourceCase);

Character\* character;

int \_x, \_y;

int init\_x, init\_y;

int currentFloor;

int hp;

BloodBar bloodBar;

int attackDamage;

bool isIntersect;

bool hasGottenSource; //看素材是否已被取得

int lightBulbInside;

bool hasGottenLightBulb;

bool canStandOn;

Action action;

int STEP\_SIZE; //移動速度

int walkLength;

int BORDER;

bool isMovingLeft; //是否正在往左移動

bool isMovingRight;//是否正在往右移動

bool isMovingDown;

bool isMovingUp;

bool facingLR;//面向左:0,右:1

bool isOnTheFloor; //是否位於地面

bool isAttacking; //是否正在攻擊

Timer attackCDTime;

bool attackCD;

int attackDelayCount;

Timer isAttackedTimer;

bool isAttacked; //是否受到攻擊

Timer sparkleEffectTimer;

bool isSparkleEffectTimerStart;

bool isSparkleEffectShow;

CMovingBitmap black;

int lossHp;

bool lossHpShowFlag;

Timer lossHpTimer;

bool characterDirectionLR;

bool characterDirectionTD;

bool isViolent;

bool haveSetViolent;

bool bossDead;

};

}

#endif

============================================================

Monster.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Util.h"

#include "Monster.h"

#include "Character.h"

#include "SourceStorage.h"

#include "Map.h"

namespace game\_framework

{

Monster::Monster()

{

\_x = 400;

\_y = 400;

hp = 1200;

attackDamage = 0;

isAttacking = false;

isIntersect = false;

}

Monster::Monster(int x, int y, int enemyHp, int ATK, Character\* c)

{

\_x = x;

\_y = y;

init\_x = x;

init\_y = y;

hp = enemyHp;

bloodBar.setFullHP(hp);

attackDamage = ATK;

isAttacking = false;

isIntersect = false;

character = c;

characterDirectionLR = 0;

characterDirectionTD = 0;

bossDead = false;

hasGottenSource = false;

lossHp = 0;

lossHpShowFlag = 0;

lightBulbInside = 0;

hasGottenLightBulb = false;

isAttacked = false;

isSparkleEffectTimerStart = false;

isSparkleEffectShow = false;

}

Monster::~Monster()

{

}

void Monster::intersect()

{

if (isAlive())

{

if (character->GetRightX() >= GetLeftX() && character->GetRightX() <= GetRightX()

&& character->GetBottomY() >= GetTopY() && character->GetBottomY() <= GetBottomY())

{ //角色右方碰到怪物

isIntersect = true;

if (!character->GetIsInvincible())

{

character->SetIsAttackedFromRight(true);

}

}

if (character->GetLeftX() <= GetRightX() && character->GetLeftX() >= GetLeftX()

&& character->GetBottomY() >= GetTopY() && character->GetBottomY() <= GetBottomY())

{ //角色左方碰到怪物

isIntersect = true;

if (!character->GetIsInvincible())

{

character->SetIsAttackedFromLeft(true);

}

}

if ((character->GetRightX() >= GetLeftX() && character->GetRightX() <= GetRightX() ||

character->GetLeftX() <= GetRightX() && character->GetLeftX() >= GetLeftX() ||

character->GetLeftX() <= GetLeftX() && character->GetRightX() >= GetRightX() || //角色比怪物寬

character->GetRightX() <= GetRightX() && character->GetLeftX() >= GetLeftX()) && //怪物比角色寬

((character->GetBottomY() >= GetTopY() && character->GetBottomY() <= GetBottomY()) || //角色下方碰到怪物

(character->GetTopY() >= GetTopY() && character->GetTopY() <= GetBottomY())))

{

isIntersect = true;

if (!character->GetIsInvincible())

{

character->SetIsAttackedFromBottom(true);

}

}

if (isIntersect && !character->GetIsInvincible())

{

character->lossCurrentHp(attackDamage);

}

}

isIntersect = false;

}

void Monster::SetIsIntersect(bool flag)

{

isIntersect = flag;

}

bool Monster::GetIsIntersect()

{

return isIntersect;

}

int Monster::distanceToCharacter() //與角色的距離 若太遠則不用動作

{

int x\_distance = (GetLeftX() + GetRightX()) / 2 - (character->GetLeftX() + character->GetRightX()) / 2;

int y\_distance = \_y - character->GetTopY();

return (int)(sqrt(pow(x\_distance, 2) + pow(y\_distance, 2)));

}

void Monster::SetCharacterDirection()

{

int cX, cY, mX, mY;

cX = (character->GetLeftX() + character->GetRightX()) / 2;

cY = (character->GetBottomY() + character->GetTopY()) / 2;

mX = (\_x + GetRightX()) / 2;

mY = (\_y + GetBottomY()) / 2;

if (cX < mX)

{

characterDirectionLR = 0; //角色在怪物左邊

}

else

{

characterDirectionLR = 1;

}

if (cY < mY)

{

characterDirectionTD = 0;

}

else

{

characterDirectionTD = 1;

}

}

bool Monster::GetCharacterDirectionLR()

{

return characterDirectionLR;

}

bool Monster::GetCharacterDirectionTD()

{

return characterDirectionTD;

}

double Monster::hpProportion()

{

return (double)(hp) / bloodBar.getFullHP();

}

void Monster::SetXY(int x, int y)

{

\_x = x;

\_y = y;

}

int Monster::GetBorder()

{

return BORDER;

}

bool Monster::GetCanStandOn()

{

return canStandOn;

}

void Monster::SetMovingDown(bool b)

{

isMovingDown = b;

}

bool Monster::GetIsMovingDown()

{

return isMovingDown;

}

void Monster::SetMovingUp(bool b)

{

isMovingUp = b;

}

bool Monster::GetIsMovingUp()

{

return isMovingUp;

}

void Monster::SetMovingLeft(bool b)

{

isMovingLeft = b;

}

bool Monster::GetIsMovingLeft()

{

return isMovingLeft;

}

void Monster::SetMovingRight(bool b)

{

isMovingRight = b;

}

bool Monster::GetIsMovingRight()

{

return isMovingRight;

}

void Monster::SetFacingLR(bool b)

{

facingLR = b;

}

bool Monster::GetFacingLR()

{

return facingLR;

}

bool Monster::isAlive()

{

if (hp <= 0)

{

return false;

}

return true;

}

bool Monster::isAttack()

{

return isAttacking;

}

void Monster::knockBack()

{

if (facingLR == 0)

{

\_x += 1;

}

if (facingLR == 1)

{

\_x -= 1;

}

}

void Monster::SetIsOnTheFloor(bool b)

{

isOnTheFloor = b;

}

bool Monster::GetIsOnTheFloor()

{

return isOnTheFloor;

}

void Monster::SetCurrentHp(int x)

{

hp = x;

}

int Monster::GetCurrentHp()

{

return hp;

}

void Monster::lossCurrentHp(int x)

{

hp -= x;

lossHp = x;

lossHpShowFlag = true;

lossHpTimer.Start();

isAttacked = true;

isAttackedTimer.Start();

isSparkleEffectShow = true;

sparkleEffectTimer.Start();

isSparkleEffectTimerStart = true;

}

void Monster::lossHpShow(Map\* m)

{

CDC\* pDC = CDDraw::GetBackCDC();

CFont f, \* fp;

f.CreatePointFont(120, "Times New Roman");

fp = pDC->SelectObject(&f);

/\*pDC->SetBkColor(RGB(230, 220, 200));\*/

pDC->SetBkMode(TRANSPARENT);

pDC->SetTextColor(RGB(255, 0, 0));

char position[500];

sprintf(position, "%d", lossHp);

pDC->TextOut(GetRightX() + m->getXMovement() + 10, GetTopY() + m->getYMovement() + 10, position);

pDC->SelectObject(fp);

CDDraw::ReleaseBackCDC();

}

void Monster::SetIsAttacking(bool flag)

{

isAttacking = flag;

}

bool Monster::GetIsAttacking()

{

return isAttacking;

}

void Monster::SetAttackDamage(int x)

{

attackDamage = x;

}

int Monster::GetAttackDamage()

{

return attackDamage;

}

void Monster::attackStart()

{

action = attack\_a;

attackCDTime.Start();

attackCD = true;

}

void Monster::attackJudge(int attackRange)

{

if (!character->GetIsInvincible())

{

if (facingLR == 0)

{

if (isAttackSuccessfullyL(attackRange))

{

character->SetIsAttackedFromRight(true);

character->lossCurrentHp(attackDamage);

}

}

else

{

if (isAttackSuccessfullyR(attackRange))

{

character->SetIsAttackedFromLeft(true);

character->lossCurrentHp(attackDamage);

}

}

}

}

void Monster::SetIsAttacked(bool flag)

{

isAttacked = flag;

}

bool Monster::GetIsAttacked()

{

return isAttacked;

}

void Monster::isAttackedEffectCaculation()

{

if (isAttacked)

{

isAttackedTimer.CaculateTimeForFalse(&isAttacked, 0.5);

}

if (isSparkleEffectShow)

{

sparkleEffectTimer.CaculateTimeForFalse(&isSparkleEffectShow, 0.1);

if (!isSparkleEffectShow)

{

isSparkleEffectTimerStart = false;

}

}

else

{

if (!isSparkleEffectTimerStart)

{

sparkleEffectTimer.Start();

isSparkleEffectTimerStart = true;

}

sparkleEffectTimer.CaculateTimeForTrue(&isSparkleEffectShow, 0.15);

}

}

void Monster::isAttackedEffectOnShow(Map\* m)

{

black.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

black.ShowBitmap();

}

void Monster::SetIsViolent(bool flag)

{

isViolent = flag;

}

bool Monster::GetIsViolent()

{

return isViolent;

}

void Monster::SetBossDead(bool flag)

{

bossDead = flag;

}

bool Monster::GetBossDead()

{

return bossDead;

}

bool Monster::isAttackSuccessfullyL(int range)

{

if (((character->GetRightX() >= GetLeftX() - range && character->GetRightX() <= GetLeftX()) ||

(character->GetLeftX() >= GetLeftX() - range && character->GetLeftX() <= GetLeftX()) ||

(character->GetLeftX() <= GetLeftX() - range && character->GetRightX() >= GetRightX()))

&& ((character->GetBottomY() >= GetTopY() && character->GetBottomY() <= GetBottomY()) ||

(character->GetTopY() >= GetTopY() && character->GetTopY() <= GetBottomY()) ||

(character->GetTopY() <= GetTopY() && character->GetBottomY() >= GetBottomY())))

{

return true;

}

else

{

return false;

}

}

bool Monster::isAttackSuccessfullyR(int range)

{

if (((character->GetLeftX() <= GetRightX() + range && character->GetLeftX() >= GetRightX()) ||

(character->GetRightX() <= GetRightX() + range && character->GetRightX() >= GetRightX()) ||

(character->GetLeftX() <= GetRightX() && character->GetRightX() >= GetRightX() + range))

&& ((character->GetBottomY() >= GetTopY() && character->GetBottomY() <= GetBottomY()) ||

(character->GetTopY() >= GetTopY() && character->GetTopY() <= GetBottomY()) ||

(character->GetBottomY() >= GetBottomY() && character->GetTopY() <= GetTopY())))

{

return true;

}

else

{

return false;

}

}

void Monster::touchSource(Map\* m, int sourceCase)

{

if (character->isIntersect((GetLeftX() + GetRightX()) / 2, (GetLeftX() + GetRightX()) / 2 + 64, currentFloor - 64, currentFloor))

{

hasGottenSource = true;

if (sourceCase == green\_sword\_s)

{

character->characterHasSword(true);

}

else

{

character->GetSourceStorage()->add(sourceCase);

}

}

}

}

============================================================

MonsterTree.h

============================================================

#ifndef \_\_MONSTERTREE\_\_

#define \_\_MONSTERTREE\_\_

namespace game\_framework

{

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

//這個class是怪物Tree的物件 //

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

class MonsterTree :public Monster

{

public:

MonsterTree();

MonsterTree(int x, int y, Character\* c);

~MonsterTree();

void LoadBitmap();

void Initialize() override;

void OnShow(Map\* m) override;

void OnMove(Map\* m) override;

int GetLeftX();// 左上角 x 座標

int GetTopY();// 左上角 y 座標

int GetRightX();// 右下角 x 座標

int GetBottomY();// 右下角 y 座標

Action actionController();

private:

void walkOnMove();

void attackOnMove();

int velocity;

CMovingBitmap sleepLeft;

CMovingBitmap sleepRight;

CAnimation walkLeft;

CAnimation walkRight;

CAnimation attackLeft;

CAnimation attackRight;

CMovingBitmap deadLeft;

CMovingBitmap deadRight;

CMovingBitmap sourceGuavaJuiceBlood;

};

}

#endif

============================================================

MonsterTree.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include <ctgmath>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "Character.h"

#include "BloodBar.h"

#include "Monster.h"

#include "MonsterTree.h"

namespace game\_framework

{

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

//這個class是怪物Tree的物件 //

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

MonsterTree::MonsterTree()

{

\_x = 400;

\_y = 400;

hp = 10;

attackDamage = 5;

facingLR = 0;

action = sleep\_a;

STEP\_SIZE = 1;

attackCD = false;

}

MonsterTree::MonsterTree(int x, int y, Character\* c) : Monster(x, y, 50, 5, c)

{

hp = 50;

attackDamage = 5;

facingLR = 0;

action = sleep\_a;

STEP\_SIZE = 1;

attackCD = false;

}

MonsterTree::~MonsterTree()

{

}

void MonsterTree::LoadBitmap()

{

bloodBar.LoadBitmap();

sleepLeft.LoadBitmap(".\\res\\monster\_tree\_sleep\_left.bmp", RGB(0, 0, 0));

sleepRight.LoadBitmap(".\\res\\monster\_tree\_sleep\_right.bmp", RGB(0, 0, 0));

walkLeft.AddBitmap(".\\res\\monster\_tree\_walk01\_left.bmp", RGB(0, 0, 0));

walkLeft.AddBitmap(".\\res\\monster\_tree\_walk02\_left.bmp", RGB(0, 0, 0));

walkLeft.AddBitmap(".\\res\\monster\_tree\_walk03\_left.bmp", RGB(0, 0, 0));

walkLeft.AddBitmap(".\\res\\monster\_tree\_walk04\_left.bmp", RGB(0, 0, 0));

walkRight.AddBitmap(".\\res\\monster\_tree\_walk01\_right.bmp", RGB(0, 0, 0));

walkRight.AddBitmap(".\\res\\monster\_tree\_walk02\_right.bmp", RGB(0, 0, 0));

walkRight.AddBitmap(".\\res\\monster\_tree\_walk03\_right.bmp", RGB(0, 0, 0));

walkRight.AddBitmap(".\\res\\monster\_tree\_walk04\_right.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_tree\_attack01\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_tree\_attack02\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_tree\_attack03\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_tree\_attack04\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_tree\_attack05\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_tree\_attack06\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_tree\_attack07\_left.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_tree\_attack01\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_tree\_attack02\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_tree\_attack03\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_tree\_attack04\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_tree\_attack05\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_tree\_attack06\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_tree\_attack07\_right.bmp", RGB(0, 0, 0));

deadLeft.LoadBitmap(".\\res\\monster\_tree\_dead\_left.bmp", RGB(0, 0, 0));

deadRight.LoadBitmap(".\\res\\monster\_tree\_dead\_right.bmp", RGB(0, 0, 0));

sourceGuavaJuiceBlood.LoadBitmap(".\\res\\source\_guava\_juice\_blood.bmp", RGB(0, 0, 0));

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

}

void MonsterTree::Initialize()

{

\_x = init\_x;

\_y = init\_y;

currentFloor = 0;

BORDER = 5;

hp = 50;

attackDamage = 8;

attackDelayCount = 4;

facingLR = 0;

action = sleep\_a;

bloodBar.setFullHP(hp);

STEP\_SIZE = 2;

walkLength = 2;

velocity = 0;

lightBulbInside = 15;

hasGottenLightBulb = false;

hasGottenSource = false;

canStandOn = true;

isViolent = false;

haveSetViolent = false;

}

void MonsterTree::OnShow(Map\* m)

{

if (isAlive())

{

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow(m);

}

else

{

if (facingLR == 0)

{

if (action == sleep\_a)

{

sleepLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

sleepLeft.ShowBitmap();

}

else if (action == walk\_a)

{

walkLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement()); //讓圖片中怪物顯示靠向左

//walkLeft.SetDelayCount(3);

walkLeft.OnShow();

}

else

{

attackLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

attackLeft.SetDelayCount(attackDelayCount);

attackLeft.OnShow();

if (attackLeft.GetCurrentBitmapNumber() == 4)

{

attackJudge(90);

}

if (attackLeft.IsFinalBitmap())

{

action = actionController();

attackLeft.Reset();

}

}

}

else

{

if (action == sleep\_a)

{

sleepRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

sleepRight.ShowBitmap();

}

else if (action == walk\_a)

{

walkRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

walkRight.OnShow();

}

else

{

attackRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

attackRight.SetDelayCount(attackDelayCount);

attackRight.OnShow();

if (attackRight.GetCurrentBitmapNumber() == 4)

{

attackJudge(90);

}

if (attackRight.IsFinalBitmap())

{

action = actionController();

attackRight.Reset();

}

}

}

}

bloodBar.setXY(GetLeftX() + m->getXMovement(), GetTopY() + m->getYMovement() - 16);

bloodBar.showBloodBar(m, hp);

if (lossHpShowFlag)

{

lossHpShow(m);

}

lossHpTimer.CaculateTimeForFalse(&lossHpShowFlag, 0.5);

}

else

{

if (facingLR == 0)

{

deadLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

deadLeft.ShowBitmap();

}

else

{

deadRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

deadRight.ShowBitmap();

}

if (!hasGottenSource)

{

sourceGuavaJuiceBlood.SetTopLeft((GetLeftX() + GetRightX()) / 2 + m->getXMovement(), GetBottomY() - 64 + m->getYMovement());

sourceGuavaJuiceBlood.ShowBitmap();

}

}

}

void MonsterTree::OnMove(Map\* m)

{

if (isViolent && !haveSetViolent)

{

hp = 220;

attackDamage = 15;

attackDelayCount = 3;

walkLength = 4;

haveSetViolent = true;

}

if (!m == NULL)

{

m->monsterFloorChanging(GetLeftX());

if (m->getMonsterFloor() > currentFloor)

{

if (\_y < m->getMonsterFloor() - 210)

{

\_y += velocity \* 2;

if (velocity < 6)

velocity++;

}

else

{

currentFloor = m->getMonsterFloor();

\_y = currentFloor - 210; // 當y座標低於地板，更正為地板上

velocity = 0;

}

}

}

if (isAlive())

{

SetCharacterDirection();

if (distanceToCharacter() < 280 && action == sleep\_a)

{

action = walk\_a;

}

if (distanceToCharacter() >= 280 && action == walk\_a)

{

action = sleep\_a;

}

if (action == walk\_a)

{

facingLR = characterDirectionLR;

}

if (distanceToCharacter() < 130 && attackCD == false && action != attack\_a)

{

attackStart();

}

else if (distanceToCharacter() < 280 && action == walk\_a)

{

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

{

if (characterDirectionLR == 0 && (GetLeftX() - STEP\_SIZE + BORDER) >= character->GetRightX() && m->isEmpty(GetLeftX() - STEP\_SIZE - BORDER, GetBottomY()))

{

\_x -= STEP\_SIZE;

}

else if (characterDirectionLR == 1 && (GetRightX() + STEP\_SIZE - BORDER - 5) <= character->GetLeftX() && m->isEmpty(GetRightX() + STEP\_SIZE + BORDER, GetBottomY()))

{

\_x += STEP\_SIZE;

}

}

}

attackCDTime.CaculateTimeForFalse(&attackCD, 3);

if (action == walk\_a)

{

walkOnMove();

}

if (action == attack\_a)

{

attackOnMove();

}

isAttackedEffectCaculation();

//intersect();

}

else

{

if (!hasGottenLightBulb)

{

character->AddLightBulb(lightBulbInside);

hasGottenLightBulb = true;

}

if (!hasGottenSource)

{

touchSource(m, guava\_juice\_blood\_p);

}

}

}

int MonsterTree::GetLeftX() //以物件本體為主(攻擊範圍不要算在裡面)

{

if (facingLR == 0) //left

{

return \_x + 87;

}

else

{

return \_x + 60;

}

}

int MonsterTree::GetTopY() //需調整以對應顯示的圖(\_y + (圖片高度-物體高度))

{

if (action == walk\_a) //left

{

return \_y + 72;

}

else

{

return \_y + 50;

}

}

int MonsterTree::GetRightX() //加上物體本身的長度

{

if (facingLR == 0) //left

{

return \_x + 175;

}

else

{

return \_x + 145;

}

}

int MonsterTree::GetBottomY()

{

return \_y + walkLeft.Height();

}

Action MonsterTree::actionController()

{

if (distanceToCharacter() < 280)

{

return walk\_a;

}

else

{

return sleep\_a;

}

}

void MonsterTree::walkOnMove()

{

if (facingLR == 0)

{

walkLeft.OnMove();

}

else if (facingLR == 1)

{

walkRight.OnMove();

}

}

void MonsterTree::attackOnMove()

{

if (facingLR == 0)

{

attackLeft.OnMove();

}

else if (facingLR == 1)

{

attackRight.OnMove();

}

}

}

============================================================

MonsterShrimp.h

============================================================

#ifndef \_\_MONSTERSHRIMP\_\_

#define \_\_MONSTERSHRIMP\_\_

namespace game\_framework

{

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

//這個class是怪物Shrimp的物件 //

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

class MonsterShrimp :public Monster

{

public:

MonsterShrimp();

MonsterShrimp(int x, int y, Character\* c);

~MonsterShrimp();

void LoadBitmap();

void Initialize() override;

void OnShow(Map\* m) override;

void OnMove(Map\* m) override;

int GetLeftX();//左上角 x 座標

int GetTopY();//左上角 y 座標

int GetRightX();//右下角 x 座標

int GetBottomY();//右下角 y 座標

private:

void attackOnMove();

int velocity;

CAnimation walkLeft;

CAnimation walkRight;

CAnimation attackLeft;

CAnimation attackRight;

CMovingBitmap deadLeft;

CMovingBitmap deadRight;

CMovingBitmap sourceShrimpAttack;

CMovingBitmap sourceShrimpBlood;

int randN;

};

}

#endif

============================================================

MonsterShrimp.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include <ctgmath>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "Character.h"

#include "BloodBar.h"

#include "Monster.h"

#include "MonsterShrimp.h"

namespace game\_framework

{

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

//這個class是怪物Shrimp的物件 //

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

MonsterShrimp::MonsterShrimp()

{

\_x = 400;

\_y = 400;

hp = 10;

attackDamage = 5;

facingLR = 0;

action = walk\_a;

STEP\_SIZE = 1;

attackCD = false;

}

MonsterShrimp::MonsterShrimp(int x, int y, Character\* c) : Monster(x, y, 12, 50, c)

{

hp = 50;

attackDamage = 5;

facingLR = 0;

action = walk\_a;

STEP\_SIZE = 1;

attackCD = false;

}

MonsterShrimp::~MonsterShrimp()

{

}

void MonsterShrimp::LoadBitmap()

{

bloodBar.LoadBitmap();

walkLeft.AddBitmap(".\\res\\monster\_shrimp\_walk01\_left.bmp", RGB(0, 0, 0));

walkLeft.AddBitmap(".\\res\\monster\_shrimp\_walk02\_left.bmp", RGB(0, 0, 0));

walkRight.AddBitmap(".\\res\\monster\_shrimp\_walk01\_right.bmp", RGB(0, 0, 0));

walkRight.AddBitmap(".\\res\\monster\_shrimp\_walk02\_right.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_shrimp\_attack01\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_shrimp\_attack02\_left.bmp", RGB(0, 0, 0));

attackLeft.AddBitmap(".\\res\\monster\_shrimp\_attack03\_left.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_shrimp\_attack01\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_shrimp\_attack02\_right.bmp", RGB(0, 0, 0));

attackRight.AddBitmap(".\\res\\monster\_shrimp\_attack03\_right.bmp", RGB(0, 0, 0));

deadLeft.LoadBitmap(".\\res\\monster\_shrimp\_dead\_left.bmp", RGB(0, 0, 0));

deadRight.LoadBitmap(".\\res\\monster\_shrimp\_dead\_right.bmp", RGB(0, 0, 0));

sourceShrimpAttack.LoadBitmap(".\\res\\source\_shrimp\_attack.bmp", RGB(0, 0, 0));

sourceShrimpBlood.LoadBitmap(".\\res\\source\_shrimp\_blood.bmp", RGB(0, 0, 0));

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

}

void MonsterShrimp::Initialize()

{

\_x = init\_x;

\_y = init\_y;

currentFloor = 0;

BORDER = 5;

hp = 150;

bloodBar.setFullHP(hp);

attackDamage = 10;

attackDelayCount = 4;

facingLR = 0;

action = walk\_a;

STEP\_SIZE = 2;

walkLength = 2;

velocity = 0;

hasGottenSource = false;

lightBulbInside = 15;

hasGottenLightBulb = false;

srand((unsigned int)time(NULL));

canStandOn = true;

isViolent = false;

haveSetViolent = false;

}

void MonsterShrimp::OnShow(Map\* m)

{

if (isAlive())

{

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow(m);

}

else

{

if (facingLR == 0)

{

if (action == walk\_a)

{

walkLeft.SetTopLeft(\_x - 140 + m->getXMovement(), \_y + m->getYMovement()); //讓圖片中怪物顯示靠向左

//walkLeft.SetDelayCount(3);

walkLeft.OnShow();

}

else

{

attackLeft.SetTopLeft(\_x + m->getXMovement() - 110, \_y + m->getYMovement());

attackLeft.SetDelayCount(attackDelayCount);

attackLeft.OnShow();

if (attackLeft.GetCurrentBitmapNumber() == 2)

{

attackJudge(100);

}

if (attackLeft.IsFinalBitmap())

{

action = walk\_a;

attackLeft.Reset();

}

}

}

else

{

if (action == walk\_a)

{

walkRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

walkRight.OnShow();

}

else

{

attackRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

attackRight.SetDelayCount(attackDelayCount);

attackRight.OnShow();

if (attackRight.GetCurrentBitmapNumber() == 2)

{

attackJudge(100);

}

if (attackRight.IsFinalBitmap())

{

action = walk\_a;

attackRight.Reset();

}

}

}

}

bloodBar.setXY(GetLeftX() + m->getXMovement(), GetTopY() + m->getYMovement() - 16);

bloodBar.showBloodBar(m, hp);

if (lossHpShowFlag)

{

lossHpShow(m);

}

lossHpTimer.CaculateTimeForFalse(&lossHpShowFlag, 0.5);

}

else

{

if (facingLR == 0)

{

deadLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

deadLeft.ShowBitmap();

}

else

{

deadRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

deadRight.ShowBitmap();

}

if (!hasGottenSource)

{

if (randN == 0)

{

sourceShrimpAttack.SetTopLeft((GetLeftX() + GetRightX()) / 2 + m->getXMovement(), GetBottomY() - 64 + m->getYMovement());

sourceShrimpAttack.ShowBitmap();

}

else

{

sourceShrimpBlood.SetTopLeft((GetLeftX() + GetRightX()) / 2 + m->getXMovement(), GetBottomY() - 64 + m->getYMovement());

sourceShrimpBlood.ShowBitmap();

}

}

}

}

void MonsterShrimp::OnMove(Map\* m)

{

if (isViolent && !haveSetViolent)

{

hp = 350;

attackDamage = 15;

attackDelayCount = 3;

walkLength = 4;

haveSetViolent = true;

}

if (!m == NULL)

{

m->monsterFloorChanging(GetLeftX());

if (m->getMonsterFloor() > currentFloor)

{

if (\_y < m->getMonsterFloor() - 178)

{

\_y += velocity \* 2;

if (velocity < 6)

velocity++;

}

else

{

currentFloor = m->getMonsterFloor();

\_y = currentFloor - 178; // 當y座標低於地板，更正為地板上

velocity = 0;

}

}

}

if (isAlive())

{

SetCharacterDirection();

if (action == walk\_a)

{

facingLR = characterDirectionLR;

}

if (distanceToCharacter() < 190 && attackCD == false)

{

attackStart();

}

else if (distanceToCharacter() < 280 && action == walk\_a)

{

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

{

if(characterDirectionLR==0&&(GetLeftX()-STEP\_SIZE+BORDER)>=character->GetRightX()&&m->isEmpty(GetLeftX()-STEP\_SIZE-BORDER,GetBottomY()-BORDER))

{

\_x-=STEP\_SIZE;

}

else if (characterDirectionLR == 1 && (GetRightX() + STEP\_SIZE - BORDER) <= character->GetLeftX() && m->isEmpty(GetRightX() + STEP\_SIZE + BORDER, GetBottomY() - BORDER))

{

\_x += STEP\_SIZE;

}

}

randN = rand() % 2;

}

attackCDTime.CaculateTimeForFalse(&attackCD, 2);

walkLeft.OnMove();

walkRight.OnMove();

if (action == attack\_a)

{

attackOnMove();

}

isAttackedEffectCaculation();

//intersect();

}

else

{

if (!hasGottenLightBulb)

{

character->AddLightBulb(lightBulbInside);

hasGottenLightBulb = true;

}

if (hasGottenSource == false)

{

if (randN == 0)

{

touchSource(m, shrimp\_attack\_p);

}

else

{

touchSource(m, shrimp\_blood\_p);

}

}

}

}

int MonsterShrimp::GetLeftX() //顯示的圖會往左邊靠(onShow調整的)

{

return \_x;

}

int MonsterShrimp::GetTopY() //需調整以對應顯示的圖(\_y + (圖片高度-物體高度))

{

if (action == walk\_a) //walk

{

return \_y + 95;

}

else

{

return \_y + 75;

}

}

int MonsterShrimp::GetRightX() //加上物體本身的長度

{

if (action == walk\_a) //walk

{

return \_x + 125;

}

else

{

return \_x + 140;

}

}

int MonsterShrimp::GetBottomY()

{

return \_y + walkLeft.Height();

}

void MonsterShrimp::attackOnMove()

{

if (facingLR == 0)

{

attackLeft.OnMove();

}

else if (facingLR == 1)

{

attackRight.OnMove();

}

}

}

============================================================

MonsterCactus.h

============================================================

#ifndef \_\_MONSTERCACTUS\_\_

#define \_\_MONSTERCACTUS\_\_

namespace game\_framework

{

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

//這個class是怪物Cactus的物件 //

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

class MonsterCactus :public Monster

{

public:

MonsterCactus();

MonsterCactus(int x, int y, Character\* c);

~MonsterCactus();

void LoadBitmap();

void Initialize() override;

void OnShow(Map\* m) override;

int GetLeftX(); // 左上角 x 座標

int GetTopY(); // 左上角 y 座標

int GetRightX(); // 右下角 x 座標

int GetBottomY(); // 右下角 y 座標

void OnMove(Map\* m) override;

private:

CMovingBitmap cactusAlive;

CMovingBitmap cactusDead;

CMovingBitmap sourceGrassFast;

};

}

#endif

============================================================

MonsterCactus.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include <ctgmath>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "Character.h"

#include "BloodBar.h"

#include "Monster.h"

#include "MonsterCactus.h"

namespace game\_framework

{

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

//這個class是怪物Cactus的物件 //

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

MonsterCactus::MonsterCactus()

{

\_x = 400;

\_y = 400;

hp = 50;

attackDamage = 5;

BORDER = 20;

}

MonsterCactus::MonsterCactus(int x, int y, Character\* c) : Monster(x, y, 50, 5, c)

{

BORDER = 20;

}

MonsterCactus::~MonsterCactus()

{

}

void MonsterCactus::LoadBitmap()

{

bloodBar.LoadBitmap();

cactusAlive.LoadBitmap(IDB\_MONSTERCACTUSALIVE, RGB(0, 0, 0));

cactusDead.LoadBitmap(IDB\_MONSTERCACTUSDEAD, RGB(0, 0, 0));

sourceGrassFast.LoadBitmap(".\\res\\source\_grass\_fast.bmp", RGB(0, 0, 0));

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

}

void MonsterCactus::Initialize()

{

\_x = init\_x;

\_y = init\_y;

currentFloor = 0;

hp = 50;

attackDamage = 5;

bloodBar.setFullHP(hp);

hasGottenSource = false;

lightBulbInside = 10;

hasGottenLightBulb = false;

canStandOn = true;

isViolent = false;

haveSetViolent = false;

}

void MonsterCactus::OnShow(Map\* m)

{

if (isAlive())

{

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow(m);

}

else

{

cactusAlive.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

cactusAlive.ShowBitmap();

}

bloodBar.setXY(GetLeftX() + m->getXMovement(), GetTopY() + m->getYMovement() - 16);

bloodBar.showBloodBar(m, hp);

if (lossHpShowFlag)

{

lossHpShow(m);

}

lossHpTimer.CaculateTimeForFalse(&lossHpShowFlag, 0.5);

}

else

{

cactusDead.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement() + (cactusAlive.Height() - cactusDead.Height()));

cactusDead.ShowBitmap();

if (!hasGottenSource)

{

sourceGrassFast.SetTopLeft((GetLeftX() + GetRightX()) / 2 + m->getXMovement(), GetBottomY() - 64 + m->getYMovement());

sourceGrassFast.ShowBitmap();

}

}

}

int MonsterCactus::GetLeftX()

{

return \_x;

}

int MonsterCactus::GetTopY()

{

return \_y;

}

int MonsterCactus::GetRightX()

{

return \_x + cactusAlive.Width();

}

int MonsterCactus::GetBottomY()

{

return \_y + cactusAlive.Height();

}

void MonsterCactus::OnMove(Map\* m)

{

if (isViolent && !haveSetViolent)

{

hp = 1000;

attackDamage = 1;

haveSetViolent = true;

}

if (!m == NULL)

{

m->monsterFloorChanging(GetLeftX());

\_y = m->getMonsterFloor() - 120;

currentFloor = m->getMonsterFloor();

}

if (isAlive())

{

intersect();

isAttackedEffectCaculation();

}

else

{

if (!hasGottenLightBulb)

{

character->AddLightBulb(lightBulbInside);

hasGottenLightBulb = true;

}

if (!hasGottenSource)

{

touchSource(m, grass\_fast\_p);

}

}

}

}

============================================================

MonsterCactusMutation.h

============================================================

#ifndef \_\_MonsterCactusMUTATION\_\_

#define \_\_MonsterCactusMUTATION\_\_

namespace game\_framework

{

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

//這個class是怪物CactusMutation的物件 //

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

class MonsterCactusMutation :public Monster

{

public:

MonsterCactusMutation();

MonsterCactusMutation(int x, int y, Character\* c);

~MonsterCactusMutation();

void LoadBitmap();

void Initialize() override;

void OnShow(Map\* m) override;

int GetLeftX();//左上角 x 座標

int GetTopY();//左上角 y 座標

int GetRightX();//右下角 x 座標

int GetBottomY();//右下角 y 座標

void OnMove(Map\* m) override;

private:

CMovingBitmap cactusAlive;

CMovingBitmap cactusDead;

CMovingBitmap sourceGreenSword;

};

}

#endif

============================================================

MonsterCactusMutation.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include <ctgmath>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "Character.h"

#include "BloodBar.h"

#include "Monster.h"

#include "MonsterCactusMutation.h"

namespace game\_framework

{

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

//這個class是怪物CactusMutation的物件 //

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

MonsterCactusMutation::MonsterCactusMutation()

{

\_x = 400;

\_y = 400;

hp = 50;

attackDamage = 5;

BORDER = 20;

}

MonsterCactusMutation::MonsterCactusMutation(int x, int y, Character\* c) : Monster(x, y, 50, 5, c)

{

BORDER = 20;

}

MonsterCactusMutation::~MonsterCactusMutation()

{

}

void MonsterCactusMutation::LoadBitmap()

{

bloodBar.LoadBitmap();

cactusAlive.LoadBitmap(".\\res\\cactus\_g\_alive.bmp", RGB(0, 0, 0));

cactusDead.LoadBitmap(".\\res\\cactus\_g\_dead.bmp", RGB(0, 0, 0));

sourceGreenSword.LoadBitmap(".\\res\\source\_green\_sword.bmp", RGB(0, 0, 0));

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

}

void MonsterCactusMutation::Initialize()

{

\_x = init\_x;

\_y = init\_y;

currentFloor = 0;

hp = 100;

attackDamage = 5;

bloodBar.setFullHP(hp);

hasGottenSource = false;

lightBulbInside = 10;

hasGottenLightBulb = false;

isViolent = false;

haveSetViolent = false;

}

void MonsterCactusMutation::OnShow(Map\* m)

{

if (isAlive())

{

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow(m);

}

else

{

cactusAlive.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

cactusAlive.ShowBitmap();

}

bloodBar.setXY(GetLeftX() + m->getXMovement(), GetTopY() + m->getYMovement() - 16);

bloodBar.showBloodBar(m, hp);

if (lossHpShowFlag)

{

lossHpShow(m);

}

lossHpTimer.CaculateTimeForFalse(&lossHpShowFlag, 0.5);

}

else

{

cactusDead.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement() + (cactusAlive.Height() - cactusDead.Height()));

cactusDead.ShowBitmap();

if (!hasGottenSource)

{

sourceGreenSword.SetTopLeft((GetLeftX() + GetRightX()) / 2 + m->getXMovement(), GetBottomY() - 64 + m->getYMovement());

sourceGreenSword.ShowBitmap();

}

}

}

int MonsterCactusMutation::GetLeftX()

{

return \_x;

}

int MonsterCactusMutation::GetTopY()

{

return \_y;

}

int MonsterCactusMutation::GetRightX()

{

return \_x + cactusAlive.Width();

}

int MonsterCactusMutation::GetBottomY()

{

return \_y + cactusAlive.Height();

}

void MonsterCactusMutation::OnMove(Map\* m)

{

if (!m == NULL)

{

m->monsterFloorChanging(GetLeftX());

\_y = m->getMonsterFloor() - 120;

currentFloor = m->getMonsterFloor();

}

if (isAlive())

{

intersect();

isAttackedEffectCaculation();

}

else

{

if (!hasGottenLightBulb)

{

character->AddLightBulb(lightBulbInside);

hasGottenLightBulb = true;

}

if (!hasGottenSource)

{

touchSource(m, green\_sword\_s);

}

}

}

}

============================================================

MonsterMosquito.h

============================================================

#ifndef \_\_MONSTERMOSQUITO\_\_

#define \_\_MONSTERMOSQUITO\_\_

namespace game\_framework

{

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

//這個class是怪物Mosquito的物件

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

class MonsterMosquito :public Monster

{

public:

MonsterMosquito();

MonsterMosquito(int x, int y, Character\* c);

~MonsterMosquito();

void LoadBitmap();

void Initialize() override;

void OnMove(Map\* m) override;

void OnShow(Map\* m) override;

int GetLeftX();

int GetTopY();

int GetRightX();

int GetBottomY();

private:

void moveUp(Map\* m);

void moveDown(Map\* m);

void moveLeft(Map\* m);

void moveRight(Map\* m);

void flyCaseChanger();

int velocity;

int flyCase;

bool shouldFlyCaseChange;

Timer flyTimer;

CAnimation faceLeft;

CAnimation faceRight;

CMovingBitmap sourceMosquitoJump;

};

}

#endif

============================================================

MonsterMosquito.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include <ctgmath>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "Character.h"

#include "BloodBar.h"

#include "Monster.h"

#include "MonsterMosquito.h"

namespace game\_framework

{

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

//這個class是怪物Mosquito的物件

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

MonsterMosquito::MonsterMosquito()

{

\_x = 400;

\_y = 400;

hp = 50;

attackDamage = 5;

BORDER = 20;

}

MonsterMosquito::MonsterMosquito(int x, int y, Character\* c) : Monster(x, y, 50, 5, c)

{

BORDER = 20;

}

MonsterMosquito::~MonsterMosquito()

{

}

void MonsterMosquito::LoadBitmap()

{

bloodBar.LoadBitmap();

faceLeft.AddBitmap(".\\res\\monster\_mosquito01\_left.bmp", RGB(0, 0, 0));

faceLeft.AddBitmap(".\\res\\monster\_mosquito02\_left.bmp", RGB(0, 0, 0));

faceRight.AddBitmap(".\\res\\monster\_mosquito01\_right.bmp", RGB(0, 0, 0));

faceRight.AddBitmap(".\\res\\monster\_mosquito02\_right.bmp", RGB(0, 0, 0));

sourceMosquitoJump.LoadBitmap(".\\res\\source\_mosquito\_jump.bmp", RGB(0, 0, 0));

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

}

void MonsterMosquito::Initialize()

{

\_x = init\_x;

\_y = init\_y;

currentFloor = 0;

BORDER = 5;

hp = 20;

attackDamage = 3;

facingLR = 0;

bloodBar.setFullHP(hp);

STEP\_SIZE = 5;

velocity = 0;

lightBulbInside = 2;

hasGottenLightBulb = false;

hasGottenSource = false;

flyCase = 2;

shouldFlyCaseChange = false;

canStandOn = false;

isViolent = false;

haveSetViolent = false;

}

void MonsterMosquito::OnMove(Map\* m)

{

if (isViolent && !haveSetViolent)

{

hp = 200;

attackDamage = 5;

haveSetViolent = true;

}

if (!m == NULL)

{

m->monsterFloorChanging((GetLeftX() + GetRightX()) / 2);

currentFloor = m->getMonsterFloor();

}

if (isAlive())

{

if (shouldFlyCaseChange)

{

flyTimer.Start();

flyCaseChanger();

shouldFlyCaseChange = false;

}

if (flyCase == 1) // 左上

{

moveUp(m);

moveLeft(m);

}

else if (flyCase == 2) // 左下

{

moveDown(m);

moveLeft(m);

}

else if (flyCase == 3) // 右下

{

moveDown(m);

moveRight(m);

}

else if (flyCase == 4) // 右上

{

moveUp(m);

moveRight(m);

}

flyTimer.CaculateTimeForTrue(&shouldFlyCaseChange, (rand() % 200) / 100);

intersect();

isAttackedEffectCaculation();

faceLeft.OnMove();

faceRight.OnMove();

}

else

{

if (!hasGottenLightBulb)

{

character->AddLightBulb(lightBulbInside);

hasGottenLightBulb = true;

}

if (!hasGottenSource)

{

touchSource(m, mosquito\_jump\_p);

}

}

}

void MonsterMosquito::OnShow(Map\* m)

{

if (isAlive())

{

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow(m);

}

else

{

if (facingLR == 0)

{

faceLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

faceLeft.SetDelayCount(1);

faceLeft.OnShow();

}

else

{

faceRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

faceRight.SetDelayCount(1);

faceRight.OnShow();

}

}

bloodBar.setXY(GetLeftX() + m->getXMovement(), GetTopY() + m->getYMovement() - 25);

bloodBar.showBloodBar(m, hp);

if (lossHpShowFlag)

{

lossHpShow(m);

}

lossHpTimer.CaculateTimeForFalse(&lossHpShowFlag, 0.5);

}

else

{

if (!hasGottenSource)

{

sourceMosquitoJump.SetTopLeft((GetLeftX() + GetRightX()) / 2 + m->getXMovement(), currentFloor - 64 + m->getYMovement());

sourceMosquitoJump.ShowBitmap();

}

}

}

int MonsterMosquito::GetLeftX()

{

return \_x + 18;

}

int MonsterMosquito::GetTopY()

{

return \_y + 28;

}

int MonsterMosquito::GetRightX()

{

return \_x + 80;

}

int MonsterMosquito::GetBottomY()

{

return \_y + 68;

}

void MonsterMosquito::moveUp(Map\* m)

{

if (m->isEmpty(GetLeftX(), GetTopY() - STEP\_SIZE) && m->isEmpty(GetRightX(), GetTopY() - STEP\_SIZE)) //向上移動

{

\_y -= STEP\_SIZE;

}

}

void MonsterMosquito::moveDown(Map\* m)

{

if (m->isEmpty(GetLeftX(), GetBottomY() + STEP\_SIZE)) //向下移動

{

\_y += STEP\_SIZE;

}

}

void MonsterMosquito::moveLeft(Map\* m)

{

if (m->isEmpty(GetLeftX() - STEP\_SIZE, GetTopY()) && m->isEmpty(GetLeftX() - STEP\_SIZE, GetBottomY())) //向左移動

{

\_x -= STEP\_SIZE;

}

}

void MonsterMosquito::moveRight(Map\* m)

{

if (m->isEmpty(GetRightX() + STEP\_SIZE, GetTopY()) && m->isEmpty(GetRightX() + STEP\_SIZE, GetBottomY())) //向右移動

{

\_x += STEP\_SIZE;

}

}

void MonsterMosquito::flyCaseChanger()

{

if (flyCase < 4)

{

flyCase++;

}

else

{

flyCase = 1;

}

}

}

============================================================

MonsterBanana.h

============================================================

#ifndef \_\_MONSTERBANANA\_\_

#define \_\_MONSTERBANANA\_\_

namespace game\_framework

{

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

//這個class是怪物Banana的物件 //

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

class MonsterBanana :public Monster

{

public:

MonsterBanana();

MonsterBanana(int x, int y, Character\* c);

~MonsterBanana();

void LoadBitmap();

void Initialize() override;

void OnShow(Map\* m) override;

int GetLeftX();

int GetTopY();

int GetRightX();

int GetBottomY();

void OnMove(Map\* m) override;

private:

CMovingBitmap bananaAlive;

CMovingBitmap bananaDead;

CMovingBitmap sourceBananaAttack;

};

}

#endif

============================================================

MonsterBanana.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include <ctgmath>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "Character.h"

#include "BloodBar.h"

#include "Monster.h"

#include "MonsterBanana.h"

namespace game\_framework

{

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

//這個class是怪物Banana的物件 //

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

MonsterBanana::MonsterBanana()

{

\_x = 400;

\_y = 400;

hp = 50;

attackDamage = 0;

BORDER = 20;

}

MonsterBanana::MonsterBanana(int x, int y, Character\* c) : Monster(x, y, 20, 5, c)

{

BORDER = 20;

}

MonsterBanana::~MonsterBanana()

{

}

void MonsterBanana::LoadBitmap()

{

bloodBar.LoadBitmap();

bananaAlive.LoadBitmap(".\\res\\monster\_banana.bmp", RGB(0, 0, 0));

bananaDead.LoadBitmap(".\\res\\monster\_banana\_dead.bmp", RGB(0, 0, 0));

sourceBananaAttack.LoadBitmap(".\\res\\source\_banana\_attack.bmp", RGB(0, 0, 0));

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

}

void MonsterBanana::Initialize()

{

\_x = init\_x;

\_y = init\_y;

currentFloor = 0;

hp = 1;

attackDamage = 5;

bloodBar.setFullHP(hp);

hasGottenSource = false;

lightBulbInside = 60;

hasGottenLightBulb = false;

canStandOn = true;

isViolent = false;

haveSetViolent = false;

}

void MonsterBanana::OnShow(Map\* m)

{

if (isAlive())

{

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow(m);

}

else

{

bananaAlive.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

bananaAlive.ShowBitmap();

}

bloodBar.setXY(GetLeftX() + m->getXMovement(), GetTopY() - 16 + m->getYMovement());

bloodBar.showBloodBar(m, hp);

if (lossHpShowFlag)

{

lossHpShow(m);

}

lossHpTimer.CaculateTimeForFalse(&lossHpShowFlag, 0.5);

}

if (!isAlive())

{

bananaDead.SetTopLeft(\_x + m->getXMovement(), \_y + bananaAlive.Height() - bananaDead.Height() + m->getYMovement());

bananaDead.ShowBitmap();

if (!hasGottenSource)

{

sourceBananaAttack.SetTopLeft((GetLeftX() + GetRightX()) / 2 + m->getXMovement(), GetBottomY() - 64 + m->getYMovement());

sourceBananaAttack.ShowBitmap();

}

}

}

int MonsterBanana::GetLeftX()

{

return \_x;

}

int MonsterBanana::GetTopY()

{

return \_y;

}

int MonsterBanana::GetRightX()

{

return \_x + bananaAlive.Width();

}

int MonsterBanana::GetBottomY()

{

return \_y + bananaAlive.Height();

}

void MonsterBanana::OnMove(Map\* m)

{

if (!m == NULL)

{

m->monsterFloorChanging(GetLeftX());

\_y = m->getMonsterFloor() - bananaAlive.Height() - 6;

currentFloor = m->getMonsterFloor();

}

if (isAlive())

{

//intersect();

isAttackedEffectCaculation();

}

else

{

if (!hasGottenLightBulb)

{

character->AddLightBulb(lightBulbInside);

hasGottenLightBulb = true;

}

if (!hasGottenSource)

{

touchSource(m, banana\_attack\_p);

}

}

}

}

============================================================

MonsterBoss.h

============================================================

#ifndef \_\_MONSTERBOSS\_\_

#define \_\_MONSTERBOSS\_\_

namespace game\_framework

{

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

//這個class是怪物Boss的物件 //

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

class MonsterBoss :public Monster

{

public:

MonsterBoss();

MonsterBoss(int x, int y, Character\* c);

~MonsterBoss();

void LoadBitmap();

void Initialize() override;

void OnMove(Map\* m) override;

void OnShow(Map\* m) override;

int GetLeftX();

int GetTopY();

int GetRightX();

int GetBottomY();

private:

/\*行為\*/

/\*走路\*/

void walk(Map\* m);

void walkOnMove();

void walkOnShow(Map\* m);

bool CanWalkL(Map\* m);

bool CanWalkR(Map\* m);

/\*攻擊\*/

void ResetAtkCounter();

void atkCount(int i);

//hit

void hitStart();

void hitOnMove();

void hitOnShow(Map\* m);

void hitJudge();

//collide

void collideStart();

void collideOnMove();

void collideOnShow(Map\* m);

void collideJudge();

void rushMove(Map\* m);

//thron

void thronStart();

void thronBossOnMove();

void thronBossOnShow(Map\* m);

void thronOnShow(Map\* m);

void thronJudge();

bool thronCountFlag;

void SetCMidX();

//dead

void deadOnMove();

void deadOnShow(Map\* m);

CAnimation walkingRight;//向右行走動畫

CAnimation walkingLeft;//向左行走動畫

CAnimation collideRight;//向右衝撞動畫

CAnimation collideLeft;//向左衝撞動畫

CAnimation hitRight;//向右捶動畫

CAnimation hitLeft;//向左捶動畫

CAnimation hitRightEffect;//向右捶動畫

CAnimation hitLeftEffect;//向左捶動畫

CAnimation thronRight;//向右刺動畫

CAnimation thronLeft;//向左刺動畫

CAnimation thron;//刺動畫

CAnimation deadRight;

CAnimation deadLeft;

/\*攻擊\*/

int atkCounter[3]; //計算攻擊已連續使用幾次 (0:hit ; 1:collide ; 2:thronBoss)

//hit

int hitDamage;

int hitDelayCount;

bool hitCD;

Timer hitCDTimer;

//collide

int rushDistance;

int rushStepSize;

//thron

int thronDamage;

int thronCount;

int cMidX, floor;

bool thronExist;

int thronDelayCount;

};

}

#endif

============================================================

MonsterBoss.cpp

============================================================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include <ctgmath>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include "Character.h"

#include "BloodBar.h"

#include "Monster.h"

#include "MonsterBoss.h"

namespace game\_framework

{

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

//這個class是怪物Boss的物件

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

MonsterBoss::MonsterBoss()

{

\_x = 400;

\_y = 400;

hp = 100;

hitDamage = 5;

action = walk\_a;

BORDER = 5;

}

MonsterBoss::MonsterBoss(int x, int y, Character\* c) : Monster(x, y, 100, 5, c)

{

}

MonsterBoss::~MonsterBoss()

{

}

void MonsterBoss::LoadBitmap()

{

bloodBar.LoadBitmap();

//向右走動畫

walkingRight.AddBitmap(".\\res\\boss\_right\_walk01.bmp", RGB(0, 0, 0));

walkingRight.AddBitmap(".\\res\\boss\_right\_walk02.bmp", RGB(0, 0, 0));

walkingRight.AddBitmap(".\\res\\boss\_right\_walk03.bmp", RGB(0, 0, 0));

walkingRight.AddBitmap(".\\res\\boss\_right\_walk04.bmp", RGB(0, 0, 0));

//向左走動畫

walkingLeft.AddBitmap(".\\res\\boss\_left\_walk01.bmp", RGB(0, 0, 0));

walkingLeft.AddBitmap(".\\res\\boss\_left\_walk02.bmp", RGB(0, 0, 0));

walkingLeft.AddBitmap(".\\res\\boss\_left\_walk03.bmp", RGB(0, 0, 0));

walkingLeft.AddBitmap(".\\res\\boss\_left\_walk04.bmp", RGB(0, 0, 0));

//向右衝撞動畫

collideRight.AddBitmap(".\\res\\boss\_right\_collide01.bmp", RGB(0, 0, 0));

collideRight.AddBitmap(".\\res\\boss\_right\_collide02.bmp", RGB(0, 0, 0));

collideRight.AddBitmap(".\\res\\boss\_right\_collide03.bmp", RGB(0, 0, 0));

collideRight.AddBitmap(".\\res\\boss\_right\_collide03.bmp", RGB(0, 0, 0));

collideRight.AddBitmap(".\\res\\boss\_right\_collide04.bmp", RGB(0, 0, 0));

//向左衝撞動畫

collideLeft.AddBitmap(".\\res\\boss\_left\_collide01.bmp", RGB(0, 0, 0));

collideLeft.AddBitmap(".\\res\\boss\_left\_collide02.bmp", RGB(0, 0, 0));

collideLeft.AddBitmap(".\\res\\boss\_left\_collide03.bmp", RGB(0, 0, 0));

collideLeft.AddBitmap(".\\res\\boss\_left\_collide03.bmp", RGB(0, 0, 0));

collideLeft.AddBitmap(".\\res\\boss\_left\_collide04.bmp", RGB(0, 0, 0));

//向右捶

hitRight.AddBitmap(".\\res\\boss\_right\_hit01.bmp", RGB(0, 0, 0));

hitRight.AddBitmap(".\\res\\boss\_right\_hit02.bmp", RGB(0, 0, 0));

hitRight.AddBitmap(".\\res\\boss\_right\_hit03.bmp", RGB(0, 0, 0));

hitRight.AddBitmap(".\\res\\boss\_right\_hit04.bmp", RGB(0, 0, 0));

hitRight.AddBitmap(".\\res\\boss\_right\_hit05.bmp", RGB(0, 0, 0));

hitRight.AddBitmap(".\\res\\boss\_right\_hit06.bmp", RGB(0, 0, 0));

//向右捶效果

hitRightEffect.AddBitmap(".\\res\\boss\_right\_hit\_effect05.bmp", RGB(0, 0, 0));

hitRightEffect.AddBitmap(".\\res\\boss\_right\_hit\_effect04.bmp", RGB(0, 0, 0));

//向左捶

hitLeft.AddBitmap(".\\res\\boss\_left\_hit01.bmp", RGB(0, 0, 0));

hitLeft.AddBitmap(".\\res\\boss\_left\_hit02.bmp", RGB(0, 0, 0));

hitLeft.AddBitmap(".\\res\\boss\_left\_hit03.bmp", RGB(0, 0, 0));

hitLeft.AddBitmap(".\\res\\boss\_left\_hit04.bmp", RGB(0, 0, 0));

hitLeft.AddBitmap(".\\res\\boss\_left\_hit05.bmp", RGB(0, 0, 0));

hitLeft.AddBitmap(".\\res\\boss\_left\_hit06.bmp", RGB(0, 0, 0));

//向左捶效果

hitLeftEffect.AddBitmap(".\\res\\boss\_left\_hit\_effect05.bmp", RGB(0, 0, 0));

hitLeftEffect.AddBitmap(".\\res\\boss\_left\_hit\_effect04.bmp", RGB(0, 0, 0));

//向右刺

thronRight.AddBitmap(".\\res\\boss\_right\_thorn01.bmp", RGB(0, 0, 0));

thronRight.AddBitmap(".\\res\\boss\_right\_thorn02.bmp", RGB(0, 0, 0));

thronRight.AddBitmap(".\\res\\boss\_right\_thorn03.bmp", RGB(0, 0, 0));

//向左刺

thronLeft.AddBitmap(".\\res\\boss\_left\_thorn01.bmp", RGB(0, 0, 0));

thronLeft.AddBitmap(".\\res\\boss\_left\_thorn02.bmp", RGB(0, 0, 0));

thronLeft.AddBitmap(".\\res\\boss\_left\_thorn03.bmp", RGB(0, 0, 0));

//刺

thron.AddBitmap(".\\res\\thron\_grow01.bmp", RGB(0, 0, 0));

thron.AddBitmap(".\\res\\thron\_grow02.bmp", RGB(0, 0, 0));

thron.AddBitmap(".\\res\\thron\_red.bmp", RGB(0, 0, 0));

//死亡動畫

deadRight.AddBitmap(".\\res\\boss\_right\_dead01.bmp", RGB(0, 0, 0));

deadRight.AddBitmap(".\\res\\boss\_right\_dead02.bmp", RGB(0, 0, 0));

deadRight.AddBitmap(".\\res\\boss\_right\_dead03.bmp", RGB(0, 0, 0));

deadRight.AddBitmap(".\\res\\boss\_right\_dead04.bmp", RGB(0, 0, 0));

deadRight.AddBitmap(".\\res\\boss\_right\_dead05.bmp", RGB(0, 0, 0));

deadRight.AddBitmap(".\\res\\boss\_right\_dead06.bmp", RGB(0, 0, 0));

deadLeft.AddBitmap(".\\res\\boss\_left\_dead01.bmp", RGB(0, 0, 0));

deadLeft.AddBitmap(".\\res\\boss\_left\_dead02.bmp", RGB(0, 0, 0));

deadLeft.AddBitmap(".\\res\\boss\_left\_dead03.bmp", RGB(0, 0, 0));

deadLeft.AddBitmap(".\\res\\boss\_left\_dead04.bmp", RGB(0, 0, 0));

deadLeft.AddBitmap(".\\res\\boss\_left\_dead05.bmp", RGB(0, 0, 0));

deadLeft.AddBitmap(".\\res\\boss\_left\_dead06.bmp", RGB(0, 0, 0));

black.LoadBitmap(".\\res\\black.bmp", RGB(0, 0, 0));

}

void MonsterBoss::Initialize()

{

\_x = init\_x;

\_y = init\_y;

hp = 1000;

bloodBar.setFullHP(hp);

action = walk\_a;

BORDER = 5;

facingLR = 1;

STEP\_SIZE = 5;

attackDamage = 10;

ResetAtkCounter();

hitDamage = 5;

hitDelayCount = 8;

hitCD = false;

rushDistance = 10;

rushStepSize = 3;

thronDamage = 10;

thronCount = 0;

thronDelayCount = 4;

cMidX = 0;

floor = 595;

thronExist = false;

thronCountFlag = false;

canStandOn = true;

bossDead = false;

isViolent = false;

haveSetViolent = false;

}

void MonsterBoss::OnMove(Map\* m)

{

if (isAlive())

{

if (isViolent && !haveSetViolent)

{

hitDamage = 20;

hitDelayCount = 5;

attackDamage = 20;

thronDamage = 30;

thronDelayCount = 3;

haveSetViolent = true;

}

SetCharacterDirection();

if (action == walk\_a)

{

facingLR = characterDirectionLR;

}

if (distanceToCharacter() < 300 && hitCD == false && action == walk\_a && atkCounter[0] < 3)

{

hitStart();

atkCount(0);

}

else if ((distanceToCharacter()>550||atkCounter[0]>=3)&&action==walk\_a&&hpProportion() > 0.5)

{

collideStart();

atkCount(1);

}

else if ((distanceToCharacter()>550||atkCounter[0]>=3)&&action==walk\_a&&hpProportion()<= 0.5)

{

thronStart();

atkCount(2);

}

if (action == walk\_a)

{

walk(m);

walkOnMove();

}

else if (action == hit\_a)

{

hitOnMove();

}

else if (action == collide\_a)

{

collideOnMove();

}

else if (action == thron\_a)

{

thronBossOnMove();

}

hitCDTimer.CaculateTimeForFalse(&hitCD, 4);

isAttackedEffectCaculation();

}

else

{

if (!bossDead)

{

deadOnMove();

}

}

}

void MonsterBoss::OnShow(Map\* m)

{

if (isAlive())

{

if (isAttacked && isSparkleEffectShow)

{

isAttackedEffectOnShow(m);

}

else

{

if (action == walk\_a)

{

walkOnShow(m);

}

else if (action == hit\_a)

{

hitOnShow(m);

}

else if (action == collide\_a)

{

collideOnShow(m);

}

else if (action == thron\_a)

{

thronBossOnShow(m);

}

}

bloodBar.setXY(\_x + m->getXMovement(), \_y + m->getYMovement() - 16);

bloodBar.showBloodBar(m, hp);

if (lossHpShowFlag)

{

lossHpShow(m);

}

lossHpTimer.CaculateTimeForFalse(&lossHpShowFlag, 0.5);

}

else

{

if (!bossDead)

{

deadOnShow(m);

}

}

}

int MonsterBoss::GetLeftX()

{

if (facingLR == 0)

{

if (action == walk\_a)

{

return \_x + 90;

}

else if (action == hit\_a)

{

return \_x + 130;

}

else// if (action == collide\_a) //第三張圖 衝撞ing not蓄力

{

return \_x + 51;

}

}

else

{

if (action == walk\_a)

{

return \_x + 72;

}

else if (action == hit\_a)

{

return \_x + 90;

}

else// if (action == collide\_a)

{

return \_x + 177;

}

}

}

int MonsterBoss::GetTopY()

{

if (action == walk\_a || action == hit\_a)

{

return \_y + 90;

}

else// if (action == collide\_a)

{

return \_y + 100;

}

}

int MonsterBoss::GetRightX()

{

if (facingLR == 0)

{

if (action == walk\_a)

{

return \_x + 220;

}

else if (action == hit\_a)

{

return \_x + 216;

}

else// if (action == collide\_a)

{

return \_x + 117;

}

}

else

{

if (action == walk\_a)

{

return \_x + 200;

}

else if (action == hit\_a)

{

return \_x + 160;

}

else// if (action == collide\_a)

{

return \_x + 237;

}

}

}

int MonsterBoss::GetBottomY()

{

return \_y + walkingLeft.Height();

}

void MonsterBoss::walk(Map\* m)

{

if (facingLR == 0)

{

if (CanWalkL(m))

{

\_x -= STEP\_SIZE;

}

}

else

{

if (CanWalkR(m))

{

\_x += STEP\_SIZE;

}

}

}

void MonsterBoss::walkOnMove()

{

walkingLeft.OnMove();

walkingRight.OnMove();

}

void MonsterBoss::walkOnShow(Map\* m)

{

if (facingLR == 0)

{

walkingLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

walkingLeft.OnShow();

}

else

{

walkingRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

walkingRight.OnShow();

}

}

bool MonsterBoss::CanWalkL(Map\* m)

{

if (m->isEmpty(\_x - STEP\_SIZE, GetTopY()) && !character->isIntersect(GetLeftX() - STEP\_SIZE, GetRightX() - STEP\_SIZE, GetTopY(), GetBottomY()))

{

return true;

}

return false;

}

bool MonsterBoss::CanWalkR(Map\* m)

{

if (m->isEmpty(\_x - STEP\_SIZE, GetTopY()) && !character->isIntersect(GetLeftX() + STEP\_SIZE, GetRightX() + STEP\_SIZE, GetTopY(), GetBottomY()))

{

return true;

}

return false;

}

void MonsterBoss::ResetAtkCounter()

{

for (int i = 0; i < 3; i++)

{

atkCounter[i] = 0;

}

}

void MonsterBoss::atkCount(int i)

{

int temp = atkCounter[i];

ResetAtkCounter();

atkCounter[i] = temp + 1;

}

void MonsterBoss::hitStart()

{

action = hit\_a;

hitCDTimer.Start();

hitCD = true;

}

void MonsterBoss::hitOnMove()

{

if (facingLR == 0)

{

hitLeft.OnMove();

hitLeftEffect.OnMove();

}

else if (facingLR == 1)

{

hitRight.OnMove();

hitRightEffect.OnMove();

}

}

void MonsterBoss::hitOnShow(Map\* m)

{

if (facingLR == 0)

{

if (hitLeft.GetCurrentBitmapNumber() == 2 || hitLeft.GetCurrentBitmapNumber() == 3 || hitLeft.GetCurrentBitmapNumber() == 4)

{

hitLeft.SetTopLeft(\_x + m->getXMovement() - 60, \_y + m->getYMovement());

}

else

{

hitLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

}

hitLeft.SetDelayCount(hitDelayCount);

hitLeft.OnShow();

if (hitLeft.GetCurrentBitmapNumber() == 3 || hitLeft.GetCurrentBitmapNumber() == 4)

{

hitLeftEffect.SetTopLeft(\_x + m->getXMovement() - 60 - 290, \_y + m->getYMovement());

hitLeftEffect.SetDelayCount(hitDelayCount);

hitLeftEffect.OnShow();

hitJudge();

}

if (hitLeft.GetCurrentBitmapNumber() == 3)

{

hitJudge();

}

if (hitLeft.IsFinalBitmap())

{

action = walk\_a;

hitLeft.Reset();

}

}

else

{

if (hitRight.GetCurrentBitmapNumber() == 2 || hitRight.GetCurrentBitmapNumber() == 3 || hitRight.GetCurrentBitmapNumber() == 4)

{

hitRight.SetTopLeft(\_x + m->getXMovement() + 60, \_y + m->getYMovement());

}

else

{

hitRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

}

hitRight.SetDelayCount(hitDelayCount);

hitRight.OnShow();

if (hitRight.GetCurrentBitmapNumber() == 3 || hitRight.GetCurrentBitmapNumber() == 4)

{

hitRightEffect.SetTopLeft(\_x + m->getXMovement() + 60 - 130, \_y + m->getYMovement());

hitRightEffect.SetDelayCount(hitDelayCount);

hitRightEffect.OnShow();

hitJudge();

}

if (hitRight.GetCurrentBitmapNumber() == 3)

{

hitJudge();

}

if (hitRight.IsFinalBitmap())

{

action = walk\_a;

hitRight.Reset();

}

}

}

void MonsterBoss::hitJudge()

{

if (!character->GetIsInvincible())

{

if (facingLR == 0)

{

if (isAttackSuccessfullyL(435))

{

character->SetIsAttackedFromRight(true);

character->SetIsAttackedFromBottom(true);

character->lossCurrentHp(hitDamage);

}

if (isAttackSuccessfullyR(135))

{

character->SetIsAttackedFromLeft(true);

character->SetIsAttackedFromBottom(true);

character->lossCurrentHp(hitDamage);

}

}

else

{

if (isAttackSuccessfullyR(435))

{

character->SetIsAttackedFromLeft(true);

character->SetIsAttackedFromBottom(true);

character->lossCurrentHp(hitDamage);

}

if (isAttackSuccessfullyL(135))

{

character->SetIsAttackedFromRight(true);

character->SetIsAttackedFromBottom(true);

character->lossCurrentHp(hitDamage);

}

}

}

}

void MonsterBoss::collideStart()

{

action = collide\_a;

}

void MonsterBoss::collideOnMove()

{

if (facingLR == 0)

{

collideLeft.OnMove();

}

else if (facingLR == 1)

{

collideRight.OnMove();

}

}

void MonsterBoss::collideOnShow(Map\* m)

{

if (facingLR == 0)

{

if (collideLeft.GetCurrentBitmapNumber() == 1)

{

collideLeft.SetTopLeft(\_x + m->getXMovement() - 20, \_y + m->getYMovement());

}

else

{

collideLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

}

if (collideLeft.GetCurrentBitmapNumber() == 2 || collideLeft.GetCurrentBitmapNumber() == 3)

{

collideJudge();

rushMove(m);

}

collideLeft.OnShow();

if (collideLeft.IsFinalBitmap())

{

action = walk\_a;

collideLeft.Reset();

}

}

else

{

if (collideRight.GetCurrentBitmapNumber() == 1)

{

collideRight.SetTopLeft(\_x + m->getXMovement() + 20, \_y + m->getYMovement());

}

else

{

collideRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

}

if (collideRight.GetCurrentBitmapNumber() == 2 || collideRight.GetCurrentBitmapNumber() == 3)

{

collideJudge();

rushMove(m);

}

collideRight.OnShow();

if (collideRight.IsFinalBitmap())

{

action = walk\_a;

collideRight.Reset();

}

}

}

void MonsterBoss::collideJudge()

{

intersect();

}

void MonsterBoss::rushMove(Map\* m)

{

if (facingLR == 0)

{

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

{

if (m->isEmpty(\_x - rushStepSize, GetTopY())/\* && m->isEmpty(\_x - STEP\_SIZE, GetBottomY())\*/)

{

\_x -= rushStepSize;

}

else

{

break;

}

}

}

else

{

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

{

if (m->isEmpty(\_x + rushStepSize, GetTopY())/\* && m->isEmpty(\_x + STEP\_SIZE, GetBottomY())\*/)

{

\_x += rushStepSize;

}

else

{

break;

}

}

}

}

void MonsterBoss::thronStart()

{

action = thron\_a;

}

void MonsterBoss::thronBossOnMove()

{

if (facingLR == 0)

{

if (thronLeft.GetCurrentBitmapNumber() != 2)

{

thronLeft.OnMove();

}

else

{

thron.OnMove();

}

}

else if (facingLR == 1)

{

if (thronRight.GetCurrentBitmapNumber() != 2)

{

thronRight.OnMove();

}

else

{

thron.OnMove();

}

}

}

void MonsterBoss::thronBossOnShow(Map\* m)

{

if (facingLR == 0)

{

thronLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

//thronLeft.SetDelayCount(thronDelayCount);

thronLeft.OnShow();

if (thronLeft.GetCurrentBitmapNumber() == 2)

{

thronOnShow(m);

}

if (thronLeft.IsFinalBitmap() && thronCount >= 3)

{

thronCount = 0;

action = walk\_a;

thronLeft.Reset();

thron.Reset();

}

}

else

{

thronRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

//thronRight.SetDelayCount(thronDelayCount);

thronRight.OnShow();

if (thronRight.GetCurrentBitmapNumber() == 2)

{

thronOnShow(m);

}

if (thronRight.IsFinalBitmap() && thronCount >= 3)

{

thronCount = 0;

action = walk\_a;

thronRight.Reset();

thron.Reset();

}

}

}

void MonsterBoss::thronOnShow(Map\* m)

{

thron.SetDelayCount(thronDelayCount);

if (thron.GetCurrentBitmapNumber() == 0)

{

thronCountFlag = false;

if (!thronExist)

{

SetCMidX();

thronExist = true;

}

thron.SetTopLeft(cMidX + m->getXMovement() - 7, floor + m->getYMovement() - 50);

}

else

{

thron.SetTopLeft(cMidX + m->getXMovement() - 18, floor + m->getYMovement() - 199);

}

if (thron.GetCurrentBitmapNumber() == 2)

{

thronJudge();

if (!thronCountFlag)

{

thronCount += 1;

thronCountFlag = true;

}

thronExist = false;

}

thron.OnShow();

}

void MonsterBoss::thronJudge()

{

if (((character->GetRightX()>=cMidX - 18 && character->GetRightX() <= cMidX + 20) || //角色右邊碰到刺

(character->GetLeftX()<=cMidX+ 20 && character->GetLeftX() >= cMidX - 18) || //角色左邊碰到刺

(character->GetLeftX()<=cMidX-18&&character->GetRightX()>=cMidX+20))&&//角色左右橫跨刺

((character->GetBottomY() >= floor - 200 && character->GetBottomY() <= floor) ||

(character->GetTopY() >= floor - 200 && character->GetTopY() <= floor)))

{

if (!character->GetIsInvincible())

{

character->SetIsAttackedFromBottom(true);

character->lossCurrentHp(thronDamage);

}

}

}

void MonsterBoss::SetCMidX()

{

cMidX = (character->GetLeftX() + character->GetRightX()) / 2;

}

void MonsterBoss::deadOnMove()

{

if (facingLR == 0)

{

deadLeft.OnMove();

}

else if (facingLR == 1)

{

deadRight.OnMove();

}

}

void MonsterBoss::deadOnShow(Map\* m)

{

if (facingLR == 0)

{

deadLeft.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

deadLeft.OnShow();

if (deadLeft.IsFinalBitmap())

{

bossDead = true;

deadLeft.Reset();

}

}

else

{

deadRight.SetTopLeft(\_x + m->getXMovement(), \_y + m->getYMovement());

deadRight.OnShow();

if (deadRight.IsFinalBitmap())

{

bossDead = true;

deadRight.Reset();

}

}

}

}

===============================

BloodBar.h

===============================

#ifndef \_\_BLOODBAR\_\_

#define \_\_BLOODBAR\_\_

namespace game\_framework

{

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

// 這個class提供血條 //

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

class Map;

class BloodBar

{

public:

BloodBar();

void LoadBitmap();

void showBloodBar(Map\* m, int hp);

void setFullHP(int n); //設定滿血的血量

void setXY(int x, int y); //設定血條的x y 座標

int getFullHP(); //回傳滿血的血量

private:

CMovingBitmap bloodBar[10]; //不同長度的血條

double maxHp;

int \_x, \_y;

};

}

#endif

===============================

BloodBar.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Map.h"

#include "BloodBar.h"

namespace game\_framework

{

BloodBar::BloodBar()

{

\_x = 0;

\_y = 0;

maxHp = 0;

}

void BloodBar::setFullHP(int n)

{

maxHp = n;

}

void BloodBar::setXY(int x, int y)

{

\_x = x;

\_y = y;

}

int BloodBar::getFullHP()

{

return (int)maxHp;

}

void BloodBar::LoadBitmap()

{

bloodBar[0].LoadBitmap(IDB\_BLOOD1, RGB(0, 0, 0));

bloodBar[1].LoadBitmap(IDB\_BLOOD2, RGB(0, 0, 0));

bloodBar[2].LoadBitmap(IDB\_BLOOD3, RGB(0, 0, 0));

bloodBar[3].LoadBitmap(IDB\_BLOOD4, RGB(0, 0, 0));

bloodBar[4].LoadBitmap(IDB\_BLOOD5, RGB(0, 0, 0));

bloodBar[5].LoadBitmap(IDB\_BLOOD6, RGB(0, 0, 0));

bloodBar[6].LoadBitmap(IDB\_BLOOD7, RGB(0, 0, 0));

bloodBar[7].LoadBitmap(IDB\_BLOOD8, RGB(0, 0, 0));

bloodBar[8].LoadBitmap(IDB\_BLOOD9, RGB(0, 0, 0));

bloodBar[9].LoadBitmap(IDB\_BLOOD10, RGB(0, 0, 0));

}

void BloodBar::showBloodBar(Map\* m, int hp)

{ // 現在血量/滿血血量 的比例 決定血條多長 顯示哪個血條bitmap

if (hp >= maxHp)

{

setFullHP(hp);

}

double proportion = hp / maxHp;

if (proportion <= 1.0 / 10)

{

bloodBar[0].SetTopLeft(\_x, \_y - 20);

bloodBar[0].ShowBitmap();

}

else if (proportion <= 2.0 / 10)

{

bloodBar[1].SetTopLeft(\_x, \_y - 20);

bloodBar[1].ShowBitmap();

}

else if (proportion <= 3.0 / 10)

{

bloodBar[2].SetTopLeft(\_x, \_y - 20);

bloodBar[2].ShowBitmap();

}

else if (proportion <= 4.0 / 10)

{

bloodBar[3].SetTopLeft(\_x, \_y - 20);

bloodBar[3].ShowBitmap();

}

else if (proportion <= 5.0 / 10)

{

bloodBar[4].SetTopLeft(\_x, \_y - 20);

bloodBar[4].ShowBitmap();

}

else if (proportion <= 6.0 / 10)

{

bloodBar[5].SetTopLeft(\_x, \_y - 20);

bloodBar[5].ShowBitmap();

}

else if (proportion <= 7.0 / 10)

{

bloodBar[6].SetTopLeft(\_x, \_y - 20);

bloodBar[6].ShowBitmap();

}

else if (proportion <= 8.0 / 10)

{

bloodBar[7].SetTopLeft(\_x, \_y - 20);

bloodBar[7].ShowBitmap();

}

else if (proportion <= 9.0 / 10)

{

bloodBar[8].SetTopLeft(\_x, \_y - 20);

bloodBar[8].ShowBitmap();

}

else

{

bloodBar[9].SetTopLeft(\_x, \_y-20);

bloodBar[9].ShowBitmap();

}

}

}

===============================

Source.h

===============================

#ifndef \_\_SOURCE\_\_

#define \_\_SOURCE\_\_

namespace game\_framework

{

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

// 這個class為素材

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

class Source

{

public:

Source();

void LoadBitmap(int code);

void Initialize();

void setNum(int n);

int getNum();

void consume(int n);

void add(int n);

void add();

void setXY(int x, int y);

void onShow();

private:

int num;

int \_x, \_y;

CMovingBitmap sourceBmp;

};

}

#endif

===============================

Source.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include <vector>

#include "Source.h"

namespace game\_framework

{

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

// 這個class為素材

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

Source::Source()

{

num = \_x = \_y = 0;

}

void Source::LoadBitmap(int code)

{

sourceBmp.LoadBitmap(code);

}

void Source::Initialize()

{

num = \_x = \_y = 0;

}

void Source::setNum(int n)

{

num = n;

}

int Source::getNum()

{

return num;

}

void Source::consume(int n)

{

num = num - n;

}

void Source::add(int n)

{

num = num + n;

}

void Source::add()

{

num++;

}

void Source::setXY(int x, int y)

{

\_x = x;

\_y = y;

}

void Source::onShow()

{

sourceBmp.SetTopLeft(\_x, \_y);

sourceBmp.ShowBitmap();

}

}

===============================

SourceStorage.h

===============================

#ifndef \_\_SOURCESTROAGE\_\_

#define \_\_SOURCESTROAGE\_\_

#include "Util.h"

namespace game\_framework {

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

// 這個class為素材儲存空間

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

class Source;

class SourceStorage

{

public:

SourceStorage();

~SourceStorage();

//void LoadBitmap(int code);

void Initialize();

Source\* getSource(int sourceCase);

void setNum(int sourceCase, int num);

int getNum(int sourceCase);

void add(int sourceCase, int num);

void add(int sourceCase);

void consume(int sourceCase, int num);

void setXY(int sourceCase, int x, int y);

private:

SourceCase sourceCase;

Source\* MosquitoJumpS;

Source\* ShrimpBloodS;

Source\* GrassFastS;

Source\* BananaAttackS;

Source\* ShrimpAttackS;

Source\* GuavaJuiceBloodS;

};

}

#endif

===============================

SourceStoorage.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include <vector>

#include "Source.h"

#include "SourceStorage.h"

namespace game\_framework {

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

// 這個class為素材儲存空間

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

SourceStorage::SourceStorage() {

MosquitoJumpS = new Source();

ShrimpBloodS = new Source();

GrassFastS = new Source();

BananaAttackS = new Source();

ShrimpAttackS = new Source();

GuavaJuiceBloodS = new Source();

}

SourceStorage::~SourceStorage()

{

delete MosquitoJumpS;

delete ShrimpBloodS;

delete GrassFastS;

delete BananaAttackS;

delete ShrimpAttackS;

delete GuavaJuiceBloodS;

}

//void SourceStorage::LoadBitmap(int code)

//{

//}

void SourceStorage::Initialize() {

MosquitoJumpS->Initialize();

ShrimpBloodS->Initialize();

GrassFastS->Initialize();

BananaAttackS->Initialize();

ShrimpAttackS->Initialize();

GuavaJuiceBloodS->Initialize();

}

Source\* SourceStorage::getSource(int sourceCase) {

switch (sourceCase)

{

case mosquito\_jump\_s:

return MosquitoJumpS;

case shrimp\_blood\_s:

return ShrimpBloodS;

case grass\_fast\_s:

return GrassFastS;

case banana\_attack\_s:

return BananaAttackS;

case shrimp\_attack\_s:

return ShrimpAttackS;

case guava\_juice\_blood\_s:

return GuavaJuiceBloodS;

default:

throw "sourceCase does not exist!";

}

}

void SourceStorage::setNum(int sourceCase, int num) {

getSource(sourceCase)->setNum(num);

}

int SourceStorage::getNum(int sourceCase) {

return getSource(sourceCase)->getNum();

}

void SourceStorage::add(int sourceCase, int num) {

getSource(sourceCase)->add(num);

}

void SourceStorage::add(int sourceCase) {

getSource(sourceCase)->add();

}

void SourceStorage::consume(int sourceCase, int num) {

getSource(sourceCase)->consume(num);

}

void SourceStorage::setXY(int sourceCase, int x, int y) {

getSource(sourceCase)->setXY(x, y);

}

}

===============================

Prop.h

===============================

#ifndef \_\_PROP\_\_

#define \_\_PROP\_\_

namespace game\_framework {

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

// 這個class為道具

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

class Character;

class Prop

{

public:

Prop();

void LoadBitmap(int code);

virtual void Initialize();

void setPropFlag(bool flag);

bool getPropFlag();

virtual void effect(Character\* character, bool flag) = 0;

void onShow();

private:

bool propFlag;

CMovingBitmap propDetails;

};

}

#endif

===============================

Prop.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include <vector>

#include "Character.h"

#include "Prop.h"

namespace game\_framework {

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

// 這個class為道具

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

Prop::Prop() {

propFlag = false;

}

void Prop::LoadBitmap(int code) {

propDetails.LoadBitmap(code);

}

void Prop::Initialize() {

propFlag = false;

}

void Prop::setPropFlag(bool flag) {

propFlag = flag;

}

bool Prop::getPropFlag() {

return propFlag;

}

void Prop::onShow() {

propDetails.SetTopLeft(474, 27);

propDetails.ShowBitmap();

}

}

===============================

PropStorage.h

===============================

#ifndef \_\_PROPSTROAGE\_\_

#define \_\_PROPSTROAGE\_\_

#include "Util.h"

namespace game\_framework {

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

// 這個class為道具儲存空間

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

class Prop;

class PropMosquitoJump;

class PropShrimpBlood;

class PropGrassFast;

class PropBananaAttack;

class PropShrimpAttack;

class PropGuavaJuiceBlood;

class PropStorage

{

public:

PropStorage();

~PropStorage();

void Initialize();

Prop\* getProp(int Case);

private:

PropCase Case;

PropMosquitoJump\* MosquitoJumpP;

PropShrimpBlood\* ShrimpBloodP;

PropGrassFast\* GrassFastP;

PropBananaAttack\* BananaAttackP;

PropShrimpAttack\* ShrimpAttackP;

PropGuavaJuiceBlood\* GuavaJuiceBloodP;

};

}

#endif

===============================

PropStorage.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include <vector>

#include "Prop.h"

#include "PropMosquitoJump.h"

#include "PropShrimpBlood.h"

#include "PropGrassFast.h"

#include "PropBananaAttack.h"

#include "PropShrimpAttack.h"

#include "PropGuavaJuiceBlood.h"

#include "PropStorage.h"

namespace game\_framework {

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

// 這個class為道具儲存空間

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

PropStorage::PropStorage() {

MosquitoJumpP = new PropMosquitoJump();

ShrimpBloodP = new PropShrimpBlood();

GrassFastP = new PropGrassFast();

BananaAttackP = new PropBananaAttack();

ShrimpAttackP = new PropShrimpAttack();

GuavaJuiceBloodP = new PropGuavaJuiceBlood();

}

PropStorage::~PropStorage()

{

delete MosquitoJumpP;

delete ShrimpBloodP;

delete GrassFastP;

delete BananaAttackP;

delete ShrimpAttackP;

delete GuavaJuiceBloodP;

}

void PropStorage::Initialize() {

MosquitoJumpP->Initialize();

ShrimpBloodP->Initialize();

GrassFastP->Initialize();

BananaAttackP->Initialize();

ShrimpAttackP->Initialize();

GuavaJuiceBloodP->Initialize();

}

Prop\* PropStorage::getProp(int Case) {

switch (Case)

{

case mosquito\_jump\_p:

return MosquitoJumpP;

case shrimp\_blood\_p:

return ShrimpBloodP;

case grass\_fast\_p:

return GrassFastP;

case banana\_attack\_p:

return BananaAttackP;

case shrimp\_attack\_p:

return ShrimpAttackP;

case guava\_juice\_blood\_p:

return GuavaJuiceBloodP;

default:

throw "Case does not exist!";

}

}

}

===============================

PropsBook.h

===============================

#ifndef \_\_PropsBook\_\_

#define \_\_PropsBook\_\_

namespace game\_framework {

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

// 這個class為鍛造台模式

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

class Prop;

class PropMosquitoJump;

class PropShrimpBlood;

class PropGrassFast;

class PropBananaAttack;

class PropShrimpAttack;

class PropGuavaJuiceBlood;

class PropStorage;

class Character;

class Source;

class SourceStorage;

class PropsBook

{

public:

PropsBook();

~PropsBook();

void LoadBitmap();

void Initialize(Character\* character);

void setXY(int x, int y);

void setXY(CPoint mousePosition);

int getMX();

int getMY();

void setCase();

void setCase(CPoint mousePosition);

void setIconList();

vector<CMovingBitmap> setIconPosition(vector<CMovingBitmap> iconList);

void cook();

void sourceNumShow(int sourceCase);

void propListOnShow();

void propCancellation();

CMovingBitmap\* propBitmap(PropCase pc); //回傳該bitmap是何prop

void onShow();

void LightBulbOnShow();

//cheat

void getAllProps();

void loseAllProps();

private:

int mx, my;

int choicex, choicey;

PropCase Case;

CMovingBitmap propsBook;

CMovingBitmap propsBookChoice;

CMovingBitmap propHave;

CMovingBitmap propsPot1;

CMovingBitmap MosquitoJumpIcon;

CMovingBitmap ShrimpBloodIcon;

CMovingBitmap GrassFastIcon;

CMovingBitmap BananaAttackIcon;

CMovingBitmap ShrimpAttackIcon;

CMovingBitmap GuavaJuiceBloodIcon;

vector<CMovingBitmap> iconList;

CMovingBitmap MosquitoJumpIconS;

CMovingBitmap ShrimpBloodIconS;

CMovingBitmap GrassFastIconS;

CMovingBitmap BananaAttackIconS;

CMovingBitmap ShrimpAttackIconS;

CMovingBitmap GuavaJuiceBloodIconS;

vector<PropCase> propList;

int px;

CMovingBitmap MosquitoJumpDetails;

CMovingBitmap ShrimpBloodDetails;

CMovingBitmap GrassFastDetails;

CMovingBitmap BananaAttackDetails;

CMovingBitmap ShrimpAttackDetails;

CMovingBitmap GuavaJuiceBloodDetails;

Character\* \_character;

PropStorage\* propStorage;

SourceStorage\* sourceStorage;

};

}

#endif

===============================

PropsBook.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include <vector>

#include "Util.h"

#include "SourceStorage.h"

#include "Prop.h"

#include "PropMosquitoJump.h"

#include "PropShrimpBlood.h"

#include "PropGrassFast.h"

#include "PropBananaAttack.h"

#include "PropShrimpAttack.h"

#include "PropGuavaJuiceBlood.h"

#include "PropsBook.h"

#include "PropStorage.h"

#include "Character.h"

#include "Source.h"

#include "SourceStorage.h"

namespace game\_framework

{

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

// 這個class為鍛造台模式

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

PropsBook::PropsBook()

{

Case = mosquito\_jump\_p;

}

PropsBook::~PropsBook()

{

}

void PropsBook::LoadBitmap()

{

propsBook.LoadBitmap(IDB\_PROPSBOOK);

propsBookChoice.LoadBitmap(IDB\_PROPSBOOKCHOICE, RGB(0, 0, 0));

propHave.LoadBitmap(IDB\_PROPHAVE);

propsPot1.LoadBitmap(IDB\_PROPSPOT1);

MosquitoJumpIcon.LoadBitmap(IDB\_PROPMOSQUITOJUMPICON);

ShrimpBloodIcon.LoadBitmap(IDB\_PROPSHRIMPBLOODICON);

GrassFastIcon.LoadBitmap(IDB\_PROPGRASSFASTICON);

BananaAttackIcon.LoadBitmap(IDB\_PROPBANANAATTACKICON);

ShrimpAttackIcon.LoadBitmap(IDB\_PROPSHRIMPATTACKICON);

GuavaJuiceBloodIcon.LoadBitmap(IDB\_PROPGUAVAJUICEBLOODICON);

MosquitoJumpIconS.LoadBitmap(".\\res\\prop\_mosquito\_jump\_icon\_s.bmp");

ShrimpBloodIconS.LoadBitmap(".\\res\\prop\_shrimp\_blood\_icon\_s.bmp");

GrassFastIconS.LoadBitmap(".\\res\\prop\_grass\_fast\_icon\_s.bmp");

BananaAttackIconS.LoadBitmap(".\\res\\prop\_banana\_attack\_icon\_s.bmp");

ShrimpAttackIconS.LoadBitmap(".\\res\\prop\_shrimp\_attack\_icon\_s.bmp");

GuavaJuiceBloodIconS.LoadBitmap(".\\res\\prop\_guava\_juice\_blood\_icon\_s.bmp");

MosquitoJumpDetails.LoadBitmap(IDB\_PROPMOSQUITOJUMP);

ShrimpBloodDetails.LoadBitmap(IDB\_PROPSHRIMPBLOOD);

GrassFastDetails.LoadBitmap(IDB\_PROPGRASSFAST);

BananaAttackDetails.LoadBitmap(IDB\_PROPBANANAATTACK);

ShrimpAttackDetails.LoadBitmap(IDB\_PROPSHRIMPATTACK);

GuavaJuiceBloodDetails.LoadBitmap(IDB\_PROPGUAVAJUICEBLOOD);

}

void PropsBook::Initialize(Character\* character)

{

Case = mosquito\_jump\_p;

choicex = 177;

choicey = 127;

setIconList();

propStorage = character->GetPropStorage();

sourceStorage = character->GetSourceStorage();

\_character = character;

px = 219;

}

void PropsBook::setXY(CPoint mousePosition)

{

mx = mousePosition.x;

my = mousePosition.y;

}

void PropsBook::setXY(int x, int y)

{

mx = x;

my = y;

}

int PropsBook::getMX()

{

return mx;

}

int PropsBook::getMY()

{

return my;

}

void PropsBook::setCase()

{

int column = (mx - 177) / 62;

int row = (my - 127) / 62;

if (row == 0 && column == 0)

{

Case = mosquito\_jump\_p;

}

else if (row == 0 && column == 1)

{

Case = shrimp\_blood\_p;

}

else if (row == 0 && column == 2)

{

Case = grass\_fast\_p;

}

else if (row == 0 && column == 3)

{

Case = banana\_attack\_p;

}

else if (row == 0 && column == 4)

{

Case = shrimp\_attack\_p;

}

else if (row == 0 && column == 5)

{

Case = guava\_juice\_blood\_p;

}

choicex = column \* 62 + 177;

choicey = row \* 62 + 128;

}

void PropsBook::setCase(CPoint mousePosition)

{

setXY(mousePosition);

if (mx > 177 && my > 127 && mx < 543 && my < 185)

{

setCase();

}

else if (mx > 900 && my > 580 && mx < 1023 && my < 623)

{

cook();

}

else if (mx > 219 && my > 612 && mx < 541 && my < 657)

{

propCancellation();

}

}

void PropsBook::setIconList()

{

iconList.push\_back(MosquitoJumpIcon);

iconList.push\_back(ShrimpBloodIcon);

iconList.push\_back(GrassFastIcon);

iconList.push\_back(BananaAttackIcon);

iconList.push\_back(ShrimpAttackIcon);

iconList.push\_back(GuavaJuiceBloodIcon);

iconList = setIconPosition(iconList);

}

vector<CMovingBitmap> PropsBook::setIconPosition(vector<CMovingBitmap> iconList)

{

int iconx, icony;

int counter = 0;

for (int row = 0; row < 7; row++)

{

icony = row \* 62 + 127;

for (int column = 0; column < 7; column++)

{

iconx = column \* 62 + 177;

iconList[counter].SetTopLeft(iconx, icony);

if (counter == (iconList).size() - 1)

{

return iconList;

}

counter++;

}

}

return iconList;

}

void PropsBook::cook()

{ //用過不能再用

switch (Case)

{

case mosquito\_jump\_p:

if (!(propStorage->getProp(mosquito\_jump\_p)->getPropFlag()) && sourceStorage->getSource(mosquito\_jump\_s)->getNum() >= 5 && \_character->GetLightBulbNum() >= 0)

{

CAudio::Instance()->Play(AUDIO\_COOK, false);

sourceStorage->getSource(mosquito\_jump\_s)->consume(5);

propStorage->getProp(mosquito\_jump\_p)->setPropFlag(true);

propStorage->getProp(mosquito\_jump\_p)->effect(\_character, true);

propList.push\_back(mosquito\_jump\_p);

}

break;

case shrimp\_blood\_p:

if (!(propStorage->getProp(shrimp\_blood\_p)->getPropFlag()) && sourceStorage->getSource(shrimp\_blood\_s)->getNum() >= 2 && \_character->GetLightBulbNum() >= 50)

{

CAudio::Instance()->Play(AUDIO\_COOK, false);

sourceStorage->getSource(shrimp\_blood\_s)->consume(2);

propStorage->getProp(shrimp\_blood\_p)->setPropFlag(true);

propStorage->getProp(shrimp\_blood\_p)->effect(\_character, true);

\_character->ConsumeLightBulb(50);

propList.push\_back(shrimp\_blood\_p);

}

break;

case grass\_fast\_p:

if (!(propStorage->getProp(grass\_fast\_p)->getPropFlag()) && sourceStorage->getSource(grass\_fast\_s)->getNum() >= 4 && \_character->GetLightBulbNum() >= 40)

{

CAudio::Instance()->Play(AUDIO\_COOK, false);

sourceStorage->getSource(grass\_fast\_s)->consume(4);

propStorage->getProp(grass\_fast\_p)->setPropFlag(true);

propStorage->getProp(grass\_fast\_p)->effect(\_character, true);

\_character->ConsumeLightBulb(40);

propList.push\_back(grass\_fast\_p);

}

break;

case banana\_attack\_p:

if (!(propStorage->getProp(banana\_attack\_p)->getPropFlag()) && sourceStorage->getSource(banana\_attack\_s)->getNum() >= 1 && \_character->GetLightBulbNum() >= 60)

{

CAudio::Instance()->Play(AUDIO\_COOK, false);

sourceStorage->getSource(banana\_attack\_s)->consume(1);

propStorage->getProp(banana\_attack\_p)->setPropFlag(true);

propStorage->getProp(banana\_attack\_p)->effect(\_character, true);

\_character->ConsumeLightBulb(60);

propList.push\_back(banana\_attack\_p);

}

break;

case shrimp\_attack\_p:

if (!(propStorage->getProp(shrimp\_attack\_p)->getPropFlag()) && sourceStorage->getSource(shrimp\_attack\_s)->getNum() >= 2 && \_character->GetLightBulbNum() >= 60)

{

CAudio::Instance()->Play(AUDIO\_COOK, false);

sourceStorage->getSource(shrimp\_attack\_s)->consume(2);

propStorage->getProp(shrimp\_attack\_p)->setPropFlag(true);

propStorage->getProp(shrimp\_attack\_p)->effect(\_character, true);

\_character->ConsumeLightBulb(60);

propList.push\_back(shrimp\_attack\_p);

}

break;

case guava\_juice\_blood\_p:

if (!(propStorage->getProp(guava\_juice\_blood\_p)->getPropFlag()) && sourceStorage->getSource(guava\_juice\_blood\_s)->getNum() >= 4 && \_character->GetLightBulbNum() >= 50)

{

CAudio::Instance()->Play(AUDIO\_COOK, false);

sourceStorage->getSource(guava\_juice\_blood\_s)->consume(4);

propStorage->getProp(guava\_juice\_blood\_p)->setPropFlag(true);

propStorage->getProp(guava\_juice\_blood\_p)->effect(\_character, true);

\_character->ConsumeLightBulb(50);

propList.push\_back(guava\_juice\_blood\_p);

}

break;

default:

break;

}

}

void PropsBook::propListOnShow()

{

px = 219;

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

{

propBitmap(propList.at(i))->SetTopLeft(px, 612);

propBitmap(propList.at(i))->ShowBitmap();

px += 46;

}

}

void PropsBook::LightBulbOnShow()

{

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(235, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkMode(TRANSPARENT);

pDC->SetTextColor(RGB(145, 137, 125));

char position[100]; // Demo 數字對字串的轉換

sprintf(position, "%d", \_character->GetLightBulbNum());

pDC->TextOut(228, 73, position);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

void PropsBook::getAllProps()

{

if (!propStorage->getProp(mosquito\_jump\_p)->getPropFlag())

{

propStorage->getProp(mosquito\_jump\_p)->setPropFlag(true);

propStorage->getProp(mosquito\_jump\_p)->effect(\_character, true);

propList.push\_back(mosquito\_jump\_p);

}

if (!propStorage->getProp(shrimp\_blood\_p)->getPropFlag())

{

propStorage->getProp(shrimp\_blood\_p)->setPropFlag(true);

propStorage->getProp(shrimp\_blood\_p)->effect(\_character, true);

propList.push\_back(shrimp\_blood\_p);

}

if (!propStorage->getProp(grass\_fast\_p)->getPropFlag())

{

propStorage->getProp(grass\_fast\_p)->setPropFlag(true);

propStorage->getProp(grass\_fast\_p)->effect(\_character, true);

propList.push\_back(grass\_fast\_p);

}

if (!propStorage->getProp(banana\_attack\_p)->getPropFlag())

{

propStorage->getProp(banana\_attack\_p)->setPropFlag(true);

propStorage->getProp(banana\_attack\_p)->effect(\_character, true);

propList.push\_back(banana\_attack\_p);

}

if (!propStorage->getProp(shrimp\_attack\_p)->getPropFlag())

{

propStorage->getProp(shrimp\_attack\_p)->setPropFlag(true);

propStorage->getProp(shrimp\_attack\_p)->effect(\_character, true);

propList.push\_back(shrimp\_attack\_p);

}

if (!propStorage->getProp(guava\_juice\_blood\_p)->getPropFlag())

{

propStorage->getProp(guava\_juice\_blood\_p)->setPropFlag(true);

propStorage->getProp(guava\_juice\_blood\_p)->effect(\_character, true);

propList.push\_back(guava\_juice\_blood\_p);

}

}

void PropsBook::loseAllProps()

{

if (propStorage->getProp(mosquito\_jump\_p)->getPropFlag())

{

propStorage->getProp(mosquito\_jump\_p)->setPropFlag(false);

propStorage->getProp(mosquito\_jump\_p)->effect(\_character, false);

}

if (propStorage->getProp(shrimp\_blood\_p)->getPropFlag())

{

propStorage->getProp(shrimp\_blood\_p)->setPropFlag(false);

propStorage->getProp(shrimp\_blood\_p)->effect(\_character, false);

}

if (propStorage->getProp(grass\_fast\_p)->getPropFlag())

{

propStorage->getProp(grass\_fast\_p)->setPropFlag(false);

propStorage->getProp(grass\_fast\_p)->effect(\_character, false);

}

if (propStorage->getProp(banana\_attack\_p)->getPropFlag())

{

propStorage->getProp(banana\_attack\_p)->setPropFlag(false);

propStorage->getProp(banana\_attack\_p)->effect(\_character, false);

}

if (propStorage->getProp(shrimp\_attack\_p)->getPropFlag())

{

propStorage->getProp(shrimp\_attack\_p)->setPropFlag(false);

propStorage->getProp(shrimp\_attack\_p)->effect(\_character, false);

}

if (propStorage->getProp(guava\_juice\_blood\_p)->getPropFlag())

{

propStorage->getProp(guava\_juice\_blood\_p)->setPropFlag(false);

propStorage->getProp(guava\_juice\_blood\_p)->effect(\_character, false);

}

propList.erase(propList.begin(), propList.end());

}

CMovingBitmap\* PropsBook::propBitmap(PropCase pc)

{

switch (pc)

{

case mosquito\_jump\_p:

return &MosquitoJumpIconS;

case shrimp\_blood\_p:

return &ShrimpBloodIconS;

case grass\_fast\_p:

return &GrassFastIconS;

case banana\_attack\_p:

return &BananaAttackIconS;

case shrimp\_attack\_p:

return &ShrimpAttackIconS;

case guava\_juice\_blood\_p:

return &GuavaJuiceBloodIconS;

}

throw;

}

void PropsBook::propCancellation()

{

int cancellationCase = (mx - 219) / 46;

vector <PropCase>::iterator it;

it = propList.begin();

advance(it, cancellationCase);

switch (propList.at(cancellationCase))

{

case mosquito\_jump\_p:

propStorage->getProp(mosquito\_jump\_p)->setPropFlag(false);

sourceStorage->getSource(mosquito\_jump\_s)->add(5);

propStorage->getProp(mosquito\_jump\_p)->effect(\_character, false);

propList.erase(it);

break;

case shrimp\_blood\_p:

propStorage->getProp(shrimp\_blood\_p)->setPropFlag(false);

sourceStorage->getSource(shrimp\_blood\_s)->add(1);

\_character->AddLightBulb(50);

propStorage->getProp(shrimp\_blood\_p)->effect(\_character, false);

propList.erase(it);

break;

case grass\_fast\_p:

propStorage->getProp(grass\_fast\_p)->setPropFlag(false);

sourceStorage->getSource(grass\_fast\_p)->add(5);

\_character->AddLightBulb(40);

propStorage->getProp(grass\_fast\_p)->effect(\_character, false);

propList.erase(it);

break;

case banana\_attack\_p:

propStorage->getProp(banana\_attack\_p)->setPropFlag(false);

sourceStorage->getSource(banana\_attack\_p)->add(1);

\_character->AddLightBulb(60);

propStorage->getProp(banana\_attack\_p)->effect(\_character, false);

propList.erase(it);

break;

case shrimp\_attack\_p:

propStorage->getProp(shrimp\_attack\_p)->setPropFlag(false);

sourceStorage->getSource(shrimp\_attack\_p)->add(1);

\_character->AddLightBulb(60);

propStorage->getProp(shrimp\_attack\_p)->effect(\_character, false);

propList.erase(it);

break;

case guava\_juice\_blood\_p:

propStorage->getProp(guava\_juice\_blood\_p)->setPropFlag(false);

sourceStorage->getSource(guava\_juice\_blood\_p)->add(1);

\_character->AddLightBulb(50);

propStorage->getProp(guava\_juice\_blood\_p)->effect(\_character, false);

propList.erase(it);

break;

}

}

void PropsBook::sourceNumShow(int sourceCase)

{

int sourceNum = sourceStorage->getSource(sourceCase)->getNum();

CDC\* pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f, \* fp;

f.CreatePointFont(100, "Times New Roman"); // 產生 font f; 160表示16 point的字

fp = pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(230, 220, 200));

pDC->SetTextColor(RGB(0, 0, 0));

CString cstr(to\_string(sourceNum).c\_str());

pDC->TextOut(890, 395, cstr);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

void PropsBook::onShow()

{

propsBook.SetTopLeft(0, 0);

propsBook.ShowBitmap();

propHave.SetTopLeft(675, 350);

propsPot1.SetTopLeft(806, 530);

MosquitoJumpDetails.SetTopLeft(673, 38);

ShrimpBloodDetails.SetTopLeft(673, 38);

GrassFastDetails.SetTopLeft(673, 38);

BananaAttackDetails.SetTopLeft(673, 38);

ShrimpAttackDetails.SetTopLeft(673, 38);

GuavaJuiceBloodDetails.SetTopLeft(673, 38);

propListOnShow();

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

{

if (propStorage->getProp(i)->getPropFlag())

{

iconList[i].ShowBitmap();

}

}

switch (Case)

{

case mosquito\_jump\_p:

MosquitoJumpDetails.ShowBitmap();

sourceNumShow(mosquito\_jump\_p);

if (propStorage->getProp(mosquito\_jump\_p)->getPropFlag())

{

propHave.ShowBitmap();

propsPot1.ShowBitmap();

}

break;

case shrimp\_blood\_p:

ShrimpBloodDetails.ShowBitmap();

sourceNumShow(shrimp\_blood\_p);

if (propStorage->getProp(shrimp\_blood\_p)->getPropFlag())

{

propHave.ShowBitmap();

propsPot1.ShowBitmap();

}

break;

case grass\_fast\_p:

GrassFastDetails.ShowBitmap();

sourceNumShow(grass\_fast\_p);

if (propStorage->getProp(grass\_fast\_p)->getPropFlag())

{

propHave.ShowBitmap();

propsPot1.ShowBitmap();

}

break;

case banana\_attack\_p:

BananaAttackDetails.ShowBitmap();

sourceNumShow(banana\_attack\_p);

if (propStorage->getProp(banana\_attack\_p)->getPropFlag())

{

propHave.ShowBitmap();

propsPot1.ShowBitmap();

}

break;

case shrimp\_attack\_p:

ShrimpAttackDetails.ShowBitmap();

sourceNumShow(shrimp\_attack\_p);

if (propStorage->getProp(shrimp\_attack\_p)->getPropFlag())

{

propHave.ShowBitmap();

propsPot1.ShowBitmap();

}

break;

case guava\_juice\_blood\_p:

GuavaJuiceBloodDetails.ShowBitmap();

sourceNumShow(guava\_juice\_blood\_p);

if (propStorage->getProp(guava\_juice\_blood\_p)->getPropFlag())

{

propHave.ShowBitmap();

propsPot1.ShowBitmap();

}

break;

default:

break;

}

propsBookChoice.SetTopLeft(choicex, choicey);

propsBookChoice.ShowBitmap();

LightBulbOnShow();

}

}

===============================

PropBananaAttack.h

===============================

#ifndef \_\_PROPBANANAATTACK\_\_

#define \_\_PROPBANANAATTACK\_\_

namespace game\_framework {

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

// 這個class為道具BananaAttack

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

class Character;

class PropBananaAttack : public Prop {

public:

PropBananaAttack();

~PropBananaAttack();

void Initialize();

void effect(Character\* character, bool flag) override;

private:

};

}

#endif

===============================

PropBananaAttack.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Prop.h"

#include "PropBananaAttack.h"

#include <vector>

namespace game\_framework {

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

// 這個class為道具BananaAttack

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

PropBananaAttack::PropBananaAttack() : Prop()

{

}

PropBananaAttack::~PropBananaAttack()

{

}

void PropBananaAttack::Initialize()

{

}

void PropBananaAttack::effect(Character\* character, bool flag)

{

character->EatBananaAttack(flag);

}

}

===============================

PropGrassFast.h

===============================

#ifndef \_\_PROPGRASSFAST\_\_

#define \_\_PROPGRASSFAST\_\_

namespace game\_framework {

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

// 這個class為道具GrassFast

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

class Character;

class PropGrassFast : public Prop {

public:

PropGrassFast();

~PropGrassFast();

void Initialize();

void effect(Character\* character, bool flag) override;

private:

};

}

#endif

===============================

PropGrassFast.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Prop.h"

#include "PropGrassFast.h"

#include <vector>

namespace game\_framework {

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

// 這個class為道具GrassFast

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

PropGrassFast::PropGrassFast() : Prop()

{

}

PropGrassFast::~PropGrassFast()

{

}

void PropGrassFast::Initialize()

{

}

void PropGrassFast::effect(Character\* character, bool flag)

{

character->EatGrassFast(flag);

}

}

===============================

PropGuavaJuiceBlood.h

===============================

#ifndef \_\_PROPGUAVAJUICEBLOOD\_\_

#define \_\_PROPGUAVAJUICEBLOOD\_\_

namespace game\_framework {

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

// 這個class為道具GuavaJuiceBlood

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

class Character;

class PropGuavaJuiceBlood : public Prop {

public:

PropGuavaJuiceBlood();

~PropGuavaJuiceBlood();

void Initialize();

void effect(Character\* character, bool flag) override;

private:

};

}

#endif

===============================

PropGuavaJuiceBlood.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Prop.h"

#include "PropGuavaJuiceBlood.h"

#include <vector>

namespace game\_framework {

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

// 這個class道具GuavaJuiceBlood

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

PropGuavaJuiceBlood::PropGuavaJuiceBlood() : Prop()

{

}

PropGuavaJuiceBlood::~PropGuavaJuiceBlood()

{

}

void PropGuavaJuiceBlood::Initialize()

{

}

void PropGuavaJuiceBlood::effect(Character\* character, bool flag)

{

character->EatGuavaJuiceBlood(flag);

}

}

===============================

PropShrimpAttack.h

===============================

#ifndef \_\_PROPSHRIMPATTACK\_\_

#define \_\_PROPSHRIMPATTACK\_\_

namespace game\_framework {

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

// 這個class為道具ShrimpAttack

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

class Character;

class PropShrimpAttack : public Prop {

public:

PropShrimpAttack();

~PropShrimpAttack();

void Initialize();

void effect(Character\* character, bool flag) override;

private:

};

}

#endif

===============================

PropShrimpAttack.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Prop.h"

#include "PropShrimpAttack.h"

#include <vector>

namespace game\_framework {

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

// 這個class為道具ShrimpAttack

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

PropShrimpAttack::PropShrimpAttack() : Prop()

{

}

PropShrimpAttack::~PropShrimpAttack()

{

}

void PropShrimpAttack::Initialize()

{

}

void PropShrimpAttack::effect(Character\* character, bool flag)

{

character->EatShrimpAttack(flag);

}

}

===============================

PropShrimpBlood.h

===============================

#ifndef \_\_PROPSHRIMPBLOOD\_\_

#define \_\_PROPSHRIMPBLOOD\_\_

namespace game\_framework {

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

// 這個class為道具ShrimpBlood

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

class Character;

class PropShrimpBlood : public Prop {

public:

PropShrimpBlood();

~PropShrimpBlood();

void Initialize();

void effect(Character\* character, bool flag) override;

private:

};

}

#endif

===============================

PropShrimpBlood.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Prop.h"

#include "PropShrimpBlood.h"

#include <vector>

namespace game\_framework {

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

// 這個class道具ShrimpBlood

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

PropShrimpBlood::PropShrimpBlood() : Prop()

{

}

PropShrimpBlood::~PropShrimpBlood()

{

}

void PropShrimpBlood::Initialize()

{

}

void PropShrimpBlood::effect(Character\* character, bool flag)

{

character->EatShrimpBlood(flag);

}

}

===============================

PropMosquitoJump.h

===============================

#ifndef \_\_PROPMOSQUITOJUMP\_\_

#define \_\_PROPMOSQUITOJUMP\_\_

namespace game\_framework {

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

// 這個class為道具MosquitoJump

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

class Character;

class PropMosquitoJump : public Prop {

public:

PropMosquitoJump();

~PropMosquitoJump();

void Initialize();

void effect(Character\* character, bool flag) override;

private:

};

}

#endif

===============================

PropMosquitoJump.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Character.h"

#include "Prop.h"

#include "PropMosquitoJump.h"

#include <vector>

namespace game\_framework {

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

// 這個class為道具MosquitoJump

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

PropMosquitoJump::PropMosquitoJump() : Prop()

{

}

PropMosquitoJump::~PropMosquitoJump()

{

}

void PropMosquitoJump::Initialize()

{

}

void PropMosquitoJump::effect(Character\* character, bool flag)

{

character->EatMosquitoJump(flag);

}

}

===============================

Timer.h

===============================

#ifndef \_\_TIMER\_\_

#define \_\_TIMER\_\_

namespace game\_framework

{

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

// 這個class提供計時功能 //

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

class Timer

{

public:

Timer();

~Timer();

void SetStartTime(clock\_t time);

void SetFinishTime(clock\_t time);

clock\_t GetStartTime();

clock\_t GetFinishTime();

void Start();

void Finish();

clock\_t GetTime();

void CaculateTimeForFalse(bool\* b, double time);

void CaculateTimeForTrue(bool\* b, double time);

private:

clock\_t startTime;

clock\_t finishTime;

};

}

#endif

===============================

Timer.cpp

===============================

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include <ctime>

#include "Timer.h"

namespace game\_framework

{

Timer::Timer()

{

startTime = 0;

finishTime = 0;

}

Timer::~Timer()

{

}

void Timer::SetStartTime(clock\_t time)

{

startTime = time;

}

void Timer::SetFinishTime(clock\_t time)

{

finishTime = time;

}

clock\_t Timer::GetStartTime()

{

return startTime;

}

clock\_t Timer::GetFinishTime()

{

return finishTime;

}

void Timer::Start()

{

startTime = clock();

}

void Timer::Finish()

{

finishTime = clock();

}

clock\_t Timer::GetTime()

{

return finishTime - startTime;

}

void Timer::CaculateTimeForFalse(bool\* b, double time)

{

Finish();

if (GetTime() >= time \* CLOCKS\_PER\_SEC)

{

\*b = false;

}

}

void Timer::CaculateTimeForTrue(bool\* b, double time)

{

Finish();

if (GetTime() >= time \* CLOCKS\_PER\_SEC)

{

\*b = true;

}

}

}

===============================

Util.h

===============================

#ifndef \_\_UTIL\_H\_\_

#define \_\_UTIL\_H\_\_

///以下為各鍵值

const char KEY\_ESC = 27;

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

const char KEY\_1 = 49;

const char KEY\_2 = 50;

const char KEY\_3 = 51;

const char KEY\_4 = 52;

const char KEY\_5 = 53;

const char KEY\_6 = 54;

const char KEY\_7 = 55;

const char KEY\_W = 0x57;

const char KEY\_A = 0x41;

const char KEY\_S = 0x53;

const char KEY\_D = 0x44;

const char KEY\_E = 69;

const char KEY\_Q = 81;

const char KEY\_R = 82;

const char KEY\_T = 84;

const char KEY\_Y = 89;

const char KEY\_U = 85;

const char KEY\_H = 72;

const char KEY\_G = 71;

const char KEY\_P = 80;

const char KEY\_SPACE = 0x20;

const char KEY\_CTRL = 0x11;

const char KEY\_TAB = 9;

// 定義各種音效的編號

enum AUDIO\_ID

{

AUDIO\_STARTMENU,

AUDIO\_CHOOSE,

AUDIO\_COOK,

AUDIO\_ATTACK\_HU,

AUDIO\_MUSIC\_01,

AUDIO\_MUSIC\_02,

AUDIO\_MUSIC\_03,

AUDIO\_MUSIC\_04,

AUDIO\_MUSIC\_05,

AUDIO\_MUSIC\_06,

AUDIO\_MUSIC\_07,

AUDIO\_VICTORY,

AUDIO\_LOSE,

AUDIO\_RECYCLE\_CAN,

AUDIO\_POT,

AUDIO\_ROLL

};

// 定義各關卡的編號

enum BeginningStage

{

stage\_start\_menu,

stage\_staff,

stage\_instructions

};

enum InstructionsPage

{

instructions\_page01,

instructions\_page02,

instructions\_page03,

instructions\_page\_cheat

};

enum Stage

{

stage\_1,

stage\_2,

stage\_3,

stage\_4,

stage\_5,

stage\_6,

stage\_boss,

stage\_props,

stage\_game\_complete

};

enum Action

{

sleep\_a,

stand\_a,

walk\_a,

roll\_a,

jump\_a,

attack\_a,

hit\_a,

collide\_a,

thron\_a

};

enum PropCase

{

mosquito\_jump\_p,

shrimp\_blood\_p,

grass\_fast\_p,

banana\_attack\_p,

shrimp\_attack\_p,

guava\_juice\_blood\_p

};

enum SourceCase

{

mosquito\_jump\_s,

shrimp\_blood\_s,

grass\_fast\_s,

banana\_attack\_s,

shrimp\_attack\_s,

guava\_juice\_blood\_s,

green\_sword\_s

};

#endif