**一張含有 室內, 桌, 坐, 建築物 的圖片

自動產生的描述**

**組別 : 第27組**

**題目 : 弱刻仁 X87**

**組員 : 107590053 蔡仲文**

**107590046 彭建豪**

**指導老師 : 陳偉凱老師**









**目錄**

[**一.** **簡介** 2](#_Toc43822435)

[**1.** **動機** 2](#_Toc43822436)

[**2.** **分工** 2](#_Toc43822437)

[**二.** **遊戲介紹** 3](#_Toc43822438)

[**1.** **遊戲說明** 3](#_Toc43822439)

[**2.** **遊戲圖形** 4](#_Toc43822440)

[**3.** **遊戲音效** 8](#_Toc43822441)

[**三.** **程式設計** 9](#_Toc43822442)

[**1.** **程式架構** 9](#_Toc43822443)

[**2.** **程式類別** 10](#_Toc43822444)

[**3.** **程式技術** 10](#_Toc43822445)

[**四.** **結語** 11](#_Toc43822446)

[**1.** **問題及解決方法** 11](#_Toc43822447)

[**2.** **時間表** 13](#_Toc43822448)

[**3.** **貢獻比例** 14](#_Toc43822449)

[**4.** **自我檢核表** 14](#_Toc43822450)

[**5.** **收獲** 15](#_Toc43822451)

[**6.** **心得、感想** 16](#_Toc43822452)

[**附錄** 18](#_Toc43822453)

1. **簡介**
2. **動機**

我們選擇以洛克人為這次遊戲設計樣本的原因是我們對於這款遊戲的喜愛。

洛克人可以說是我們的童年回憶，在遊戲機或是電腦的網頁遊戲裡都能找到洛克人相關的遊戲。我們希望透過自己的能力重現這款經典之作，這是對我們自身的一個考驗，也是對這款遊戲的尊重。

1. **分工**

我們兩人的分工方式以動畫和操作為分工界線，先由一人完成基本的操作功

能，然後再交給另一人完成動畫的呈現。在初期一直維持這樣的分工模式，直到學期中開始我們偶爾會互相交換工作，因為動畫的部份必須要等待基本的運作程式寫出來後才能加工呈現出來，所以有時候其中一人太忙而無法完成自己的部份的時候，另一人就會接手完成，以免拖延進度。整個學期的分工都算順利，除了期中考的兩週裡進度比較慢，其餘時間都非常理想。

1. **遊戲介紹**
2. **遊戲說明**

　　我們這款的遊戲的遊玩方式為橫向捲軸戰鬥。操作方式為上下左右鍵移動，ｚ鍵跳躍，空白鍵衝刺，ｘ鍵攻擊。我們也還完了原作中蓄力炮彈的攻擊，只要長按攻擊鍵，即可按蓄力時長發出不同的炮彈，一共三種炮彈型態。

遊戲規則部分，因為只有半學期的時間，要完整實作出原作的序章、八大關卡、三個最後關卡幾乎是不太可能，所以我們只有將第四代的其中一關的地圖實作出來。遊戲的勝利條件很簡單，只要將地圖中最後面的BOSS擊倒即可獲得勝利，可以不用清除小怪，但是因為地圖上的某種小怪會不斷追擊玩家，甚至到打BOSS時也會持續追擊，故先將地圖上的小怪擊倒，讓玩家在挑戰BOSS時，會相對輕鬆許多。死亡的規則是碰到特定的地形，像是無底洞、岩漿等，或者是持續受到怪物傷害直到生命歸0。

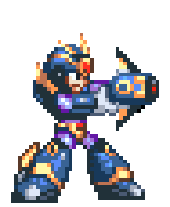
　　在遊戲裡因為一般的跳躍高度有限，無法到達某些地點，所以需要一些特殊技巧才能爬上高處。在洛克人跳躍並持續按著移動鍵貼著牆壁時，會從一般掉落變成慢慢滑落。例如牆壁在洛克人右方，跳躍並貼著牆壁後持續按著右鍵就能做到滑落狀態，想要快速落地只需放開右鍵即可。另外，在貼著牆壁滑落時，再次按z鍵跳躍即可做到踢牆的動作，重複利用這兩種技巧來到達更高的位置。某些按鍵的組合觸發特殊功能，像是按住空白鍵再按住z鍵跳躍會跳躍得更遠，速度也會加快，這種特殊功能在原作常用在閃避怪物的攻擊，是玩洛克人必學的技巧。

　　密技方面我們為新手設計了一個回血並清空全場小怪的功能，第一次玩難免會不習慣，殘血死掉要重來的話的確很麻煩，因此我們做了一個快捷鍵，只需按下去即可把小怪全部瞬間秒殺並回滿血條，以完整的狀態去挑戰BOSS。

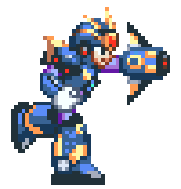
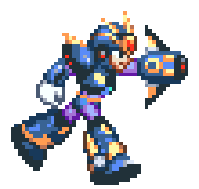
1. **遊戲圖形**

因為圖片過多的關係，所以以下只放部分動畫圖片

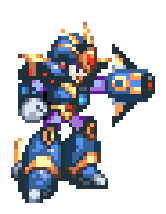
洛克人站立及射擊動畫



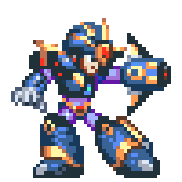
洛克人移動及射擊動畫



洛克人跳躍及射擊動畫



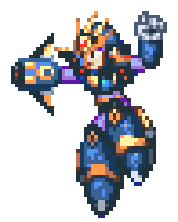
洛克人衝刺及射擊動畫



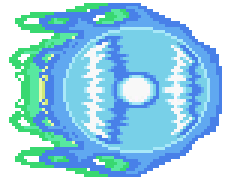
洛克人踢牆及射擊動畫



洛克人黏牆滑落及射擊動畫



洛克人受傷和洛克炮發射及擊中動畫



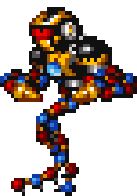
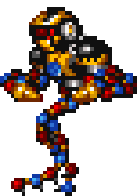
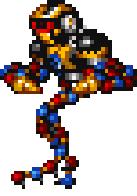
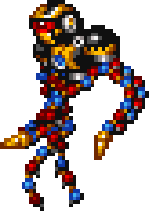
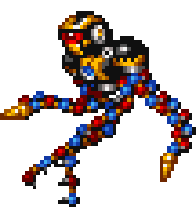
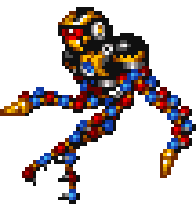
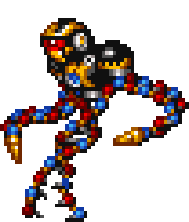
敵人:蝙蝠移動動畫

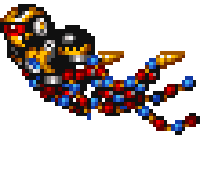
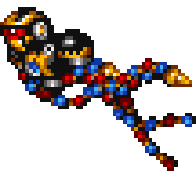


敵人:垃圾怪攻擊、站立、砲彈圖片

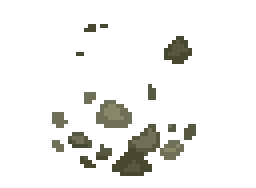


敵人:夢魘站立、攻擊、移動、砲彈圖片





爆炸動畫及生命值圖片

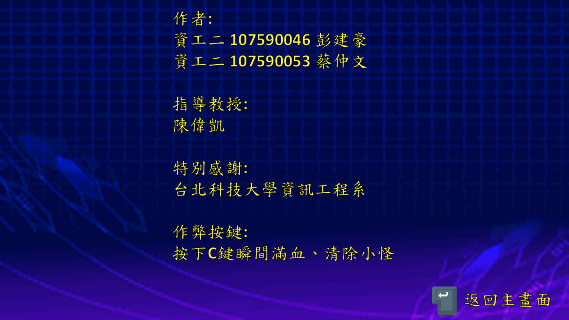
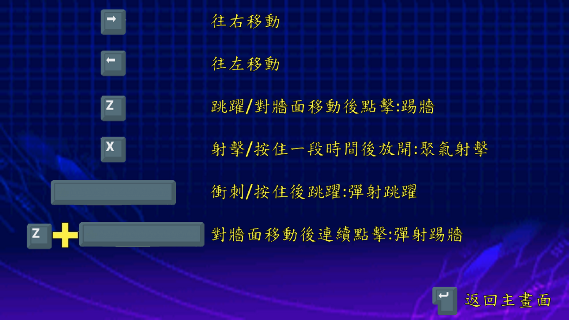


數字圖形





遊戲主畫面、選項、地圖、結束圖片



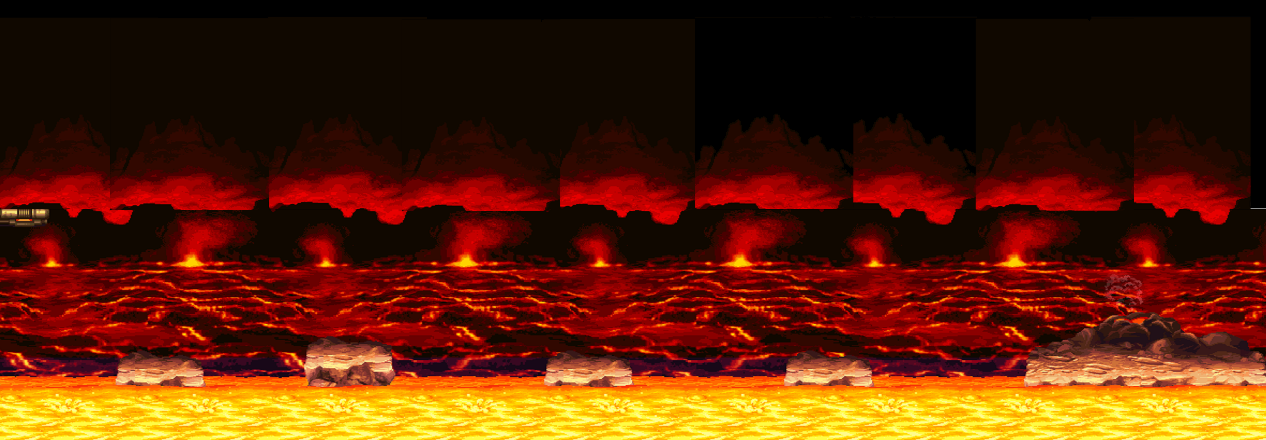


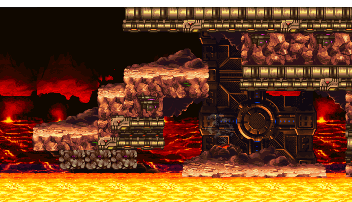




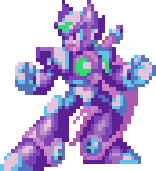
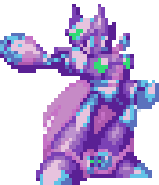


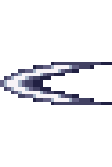
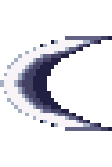
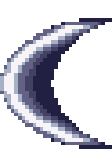
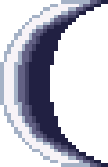
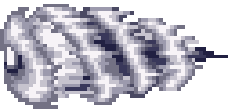
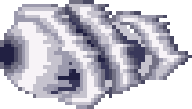
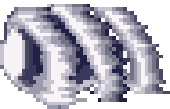


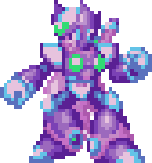
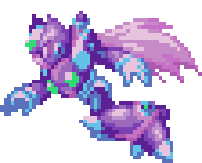


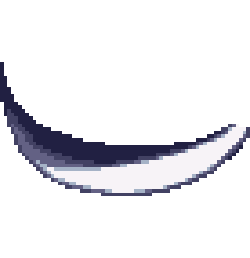
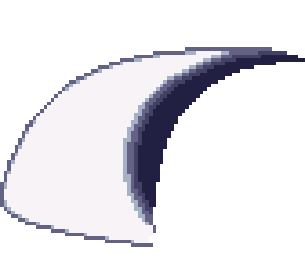


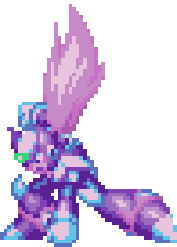
BOSS動畫、砲彈圖片

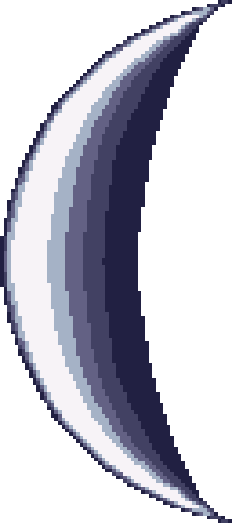
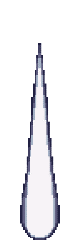
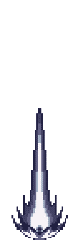
 







1. **遊戲音效**

|  |  |
| --- | --- |
| jump | 洛克人跳躍音效 |
| jump2 | 跳躍時洛克人的語音 |
| sprint | 洛克人衝刺音效 |
| injure | 洛克人受傷音效 |
| cannon1 | 小洛克炮音效 |
| cannon2 | 中洛克炮音效 |
| cannon3 | 大洛克炮音效 |
| cannon3-2 | 發射大洛克炮時的語音 |
| charge | 洛克炮蓄力音效 |
| charge-loop | 洛克炮蓄力到達最大值時的音效 |
| gamestart | 遊戲初始界面的背景音效 |
| Jakob | 挑戰關卡的背景音樂(取自第八代Jakob關卡) |
| boss\_stage | BOSS戰鬥背景音樂(取自第五代X VS 傑洛關卡) |
| enter | 遊戲初始界面按下enter時的音效 |
| optionmove | 遊戲初始界面上下移動選項時的音效 |
| enemyfire | 怪物射擊音效 |
| boom | 爆炸音效 |
| bossSprint | BOSS衝刺音效 |
| sprint\_zero | BOSS衝刺語音 |
| chop | BOSS斬擊音效 |
| dead\_zero | BOSS死亡語音 |
| destroylight | BOSS施展滅閃光(槌地板)音效 |
| destroylight\_zero | BOSS施展滅閃光(槌地板)語音 |
| fire | BOSS砲擊音效 |
| kill | BOSS施展幻夢零(巨大劍氣)音效 |
| kill\_zero | BOSS施展幻夢零(巨大劍氣)語音 |

1. **程式設計**
2. **程式架構**

一張含有 文字 的圖片

自動產生的描述

1. **程式類別**

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| Bat | 38 | 162 | 蝙蝠怪的攻擊與碰撞 |
| Boss | 82 | 779 | Boss的攻擊、移動與功能 |
| BossCannon | 49 | 351 | Boss的炮彈生成與功能 |
| Monster | 41 | 167 | 小怪的攻擊與功能 |
| Trashcannon | 40 | 224 | 小怪的炮彈生成 |
| nightmare | 55 | 296 | 夢魘怪的攻擊動作判定與功能 |
| nightmareCannon | 41 | 232 | 夢魘怪的炮彈生成 |
| Rockcannon | 57 | 437 | 洛克炮的生成 |
| Rockman | 133 | 1182 | 洛克人的移動、攻擊與功能 |
| Terrain | 51 | 1036 | 地圖的生成、碰撞與功能 |
| mygame | 125 | 560 | 遊戲運行主要邏輯 |
| **總行數** | **587** | **5426** |  |

1. **程式技術**

我們在設計遊戲程式過程中，並沒有用到太多演算法或資料結構，真要說演算法的部分，大概就是計算出追蹤砲彈的速度XY分量，程式的原理是先去抓取玩家操控角色的座標，然後和砲彈初始發射的座標做相減得出相差的寬度和高度，再用畢氏定理，也就是寬度和高度做平方相加開根號算出斜邊，再把原先求得的寬度和高度除以斜邊得到單位長，在乘上預設的速度，就可以計算出追蹤砲彈的速度XY分量，此方法就算是單純只有X或Y相差有值也能適用。

1. **結語**
2. **問題及解決方法**

　　這次的實習裡我們遇到不少困難障礙。首先在初期，我們選擇先完成物體碰撞以及跳躍衝刺等基本功能以及判定。但由於素材圖片的觀感與實際大小有一點差別，比如在遊戲看到洛克人沒有跟怪物重疊，但卻受到傷害的狀況，因此我們需要對每一張圖片進行裁剪以及修改判定範圍。

　　另外，在判定碰撞牆壁的時候，一般的走路由於移動速度不快，因此都沒有發現問題，直至我們完成衝刺動作後，發現碰撞判定無法準確的運作。由於我們的衝刺以及跳躍都有做出類似現實中慢慢加速然後再減速的動作，因此在衝刺中判定碰撞牆壁時要立即停下會偶爾出現洛克人半個身體卡進牆壁的怪狀。為了修正這部份的bug，我們花了相當長的時間。我們首先修改判定地形的邏輯運算，以洛克人為中心點作一個四邊形的掃瞄，務求在碰撞的瞬間就能立即偵測到。此外，我們還做了一個防bug的操施，就是每次移動都會更新並保留上一個移動位置，只要出現錯位或卡進牆壁的情況都會立即讀取上一個位置進行修復。

　　另一個遇到的問題出現在我們做出第一隻怪物的戰鬥裡。我們嘗試測試戰鬥時，首先發現了洛克人炮彈動畫的異狀，動畫的顯示有時候並不連續，會出現消失一段距離又再出現的情況。我們發現是因為我們的炮彈只有宣告一個，一旦我們連續按，炮彈的上一個值就會被新的覆蓋掉，因為出現動畫不連續的情況。之後經過教授的建議，我們把炮彈架構修改成一個彈匣的概念，可同時發射多個炮彈並進行回收再使用。

　　洛克人的踢牆動作我們也花費了相當多的時間。我們一開始先把觸發的條件邏輯先寫出來，然後再進行改善動作。條件判斷不太困難，困難的地方在於要怎麼讓洛克人做到踢牆的動作，又不會顯得太僵硬，像是直線飛出去那樣。我們做了很多的測試才找到合適的角度以及平衡點，讓這個操作更接近原作的樣貌。

　　在洛克人移動的過程中炮彈的動畫顯示也有一些問題。由於我們的地圖背景顯示是以洛克人為中心，然後定位坐標來移動鏡頭。但我們沒有考慮到炮彈也有同樣需要，因此出現了洛克人在移動的同時，炮彈也會跟著一起移動，這是非常不合理的，所以我們在之後的炮彈裡都加入了screenX與screenY做為動畫顯示的坐標。

　　由於我們的分工幾乎都不太會影響到對方，所以有時候需要接手對方的進度的時候，會出現要花不少時間來理解程式的內容與架構。這一點對我們來說也是有一點小困擾，但也很好解決。我們每次宋完一部份都會給對方展示，並簡單解釋一下每部份的內容與作用。註解也是非常實用與必需的東西，有助於我們快速了解該功能與定義。

　　最後，遊戲的難度設計也是一個難題，因為每個人對於遊戲的適應力都不同，所以我們也很難找到一個平衡點。我們決定參照原作裡的一些設定來做調整，並自行試玩後再作改善。我們加入回血的密技就是為了能讓大家都有好的遊戲體驗，不必需要不停的挑戰，不停的死亡，來摸清每隻怪物的技能。

1. **時間表**

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 蔡仲文(小時) | 彭建豪(小時) | 說明 |
| 0 | 7 | 7 | 討論遊戲內容與設計方向 |
| 1 | 6 | 8 | 障礙物碰撞判定、敵人生成 |
| 2 | 11 | 13 | 障礙物碰撞判定修正、地圖陣列規劃、角色攻擊、跳躍、衝刺 |
| 3 | 8 | 10 | 地圖視角移動、地圖切片、攻擊與受傷判定 |
| 4 | 10 | 8 | 攻擊傷害判定、踢牆、地圖陣列修正 |
| 5 | 8 | 7 | 怪物攻擊、遊戲UI介面 |
| 6 | 1 | 7 | 碰撞bug修正、怪物攻擊修正 |
| 7 | 8 | 1 | 怪物攻擊、新增其他不同怪物 |
| 8 | 7 | 8 | 怪物攻擊模式、怪物移動 |
| 9 | 10 | 17 | 怪物投射物、連射功能 |
| 10 | 7 | 7 | 生成多隻不同怪物、怪物攻擊動畫、連射bug修正、boss設計討論 |
| 11 | 7 | 10 | 怪物投射物動畫修正、多種怪物設計、遊戲初始介面 |
| 12 | 5 | 4 | Boss戰鬥動畫與素材整理、怪物與洛克人實戰測試 |
| 13 | 6 | 19 | Boss攻擊、Boss實戰 |
| 14 | 13 | 15 | Boss移動、Gameover畫面、遊戲音效、遊戲初始介面優化 |

1. **貢獻比例**

蔡仲文: 50%

彭建豪: 50%

1. **自我檢核表**

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

1. **收獲**

蔡仲文: 在這個學期裡，我學習到怎麼運用c++語言來實作一個遊戲。例如加入bitmap並對其進行操作是最常見的。在程式的架構設計方面，我深深的體會到一個好的架構對於程式的拓展與維護的重要性。良好的程式架構會有助於修改以及增加不同的功能，不能把不同的功能全部堆在同一個function裡面，因為這樣不光是修改很困難，而且還會影響debug時的難度。在除錯方面我也學到了怎麼適當的運用TRACE的功能來快速找出問題原因。剛開始的時候，我還不太熟悉怎麼良好的運用TRACE來debug，比如我會把TRACE放在OnMove裡，結果就是輸出欄跳出超級多的結果，這樣非常難的debug以及快速找到自己想要的數據，因為一次顯示太多，也來不及看。所以我後來學會了把TRACE放在一些重要的條件式裡，有進入條件時才輸出，這樣就相對比較容易發現問題所在，更能快速的debug了。

彭建豪:經過這整學期的遊戲程式撰寫後，我從在設計怪物的攻擊模式，還有洛克人的各種操作中，學習到對物理移動方面程式的撰寫。舉例來說，跳躍通常會是加速一小段時間過後，再慢慢減速至0，之後開始往下掉落。洛克人衝刺的部分也是同理。而在怪物的攻擊模式部分，原作裡的一些怪物的砲彈，會隨著玩家的位置而做到有角度的砲彈移動。在我還沒想到要做這功能時，我一直覺得這部分的程式會很難時做出來，但實際做起來其實並不會太困難，就是需要先去抓取玩家操控角色的座標，然後和怪物自身座標做相減得出相差的寬度和高度，再用畢氏定理算出斜邊把原先求得的寬度和高度除以斜邊得到單位長，在乘上預設的速度，就可以做到有角度的砲彈移動。最後我也從中學習到，如何利用SourceTree去做到更有效率的程式合作撰寫，而不是使用人體操作式的程式結合，也就是把自己組員的code複製貼上到自己的程式上。

1. **心得、感想**

蔡仲文: 我覺得在這次的遊戲程式設計裡獲益良多，對我改善coding方面有非常大的幫助。特別明顯的改進是程式規劃以及物件導向概念的應用。在學習這個課堂以前，我寫程式都是以不同的功能分成不同的副程式，但在OOP裡面，我還必需考慮物件的性質，比如怪物不能擁有地圖、洛克炮不能擁有洛克人這樣的一種概念。我在剛開始寫遊戲的時候程式非常的亂，大部份功能都寫在了mygame.cpp的OnMove裡，後來仔細思考老師教導的內容，以及之前Shop的練習裡的程式架構，發現自己寫的內容完全不合格，所以做了一次整理，把程式的架構重新規劃。這次的遊戲設計實習我感到很滿足，不但收改善了自己的程式規劃，也體驗到了遊戲開發員的辛苦。在第一週要決定做什麼遊戲的時候，我也曾懷疑自己是不是足夠的能力完成，不過我們做出來的成品證明了自己的能力，雖然算不上完美，也沒有完全的還完原作的所有操作，但大部份的細節完成度都非常高。整體而言，我跟組員兩人的合作都算順利，大家遇到bug時都會討論、互相幫忙，氣氛很輕鬆。

彭建豪:我覺得我在本次課程中所獲得的成就感相當得多，因為我們所選的題目是我們小時候就非常喜愛的一款遊戲，其中這款遊戲有一個戰鬥畫面令我印象非常深刻，就是本作中的兩個主角互相對打的BOSS戰鬥場面，其中BOSS的技能特效很吸引我的的目光，今天藉著本課程，成功的將本作中的兩個主角互相對打的BOSS戰鬥場面給還原出來了，能夠靠著自己的能力，將自己的童年回憶再次重現在自己眼前，我其實蠻感動的。在課堂中，除了自己的遊戲程式之外，也還可以玩一下跟自己比較熟的其他組別所做出來的作品，除了可以有不一樣的樂趣之外，有的時候，還可以互相交流一些遊戲上的建議，像是針對作弊的部分，要如何寫才能夠讓角色擁有超級破格的能力，甚至是在他人允許下亂改參數，把一些普通的技能搞到非常強大。總之，這門課雖然要做的事非常多，但也有很多不一樣的樂趣。

**附錄**

**<**mygame.h**>**

#include "BossCannon.h"

#include "Boss.h"

#include "Trashcannon.h"

#include "Monster.h"

#include "nightmareCannon.h"

#include "nightmare.h"

#include "Bat.h"

#include "Terrain.h"

#include "RockCannon.h"

#include "Rockman.h"

namespace game\_framework {

enum AUDIO\_ID { // 定義各種音效的編號

AUDIO\_OPTIONMOVE, // 0

AUDIO\_ENTER, // 1

AUDIO\_START, // 2

AUDIO\_NTUT, // 3

AUDIO\_JUMP, // 4

AUDIO\_JUMP2, // 5

AUDIO\_SPRINT, // 6

AUDIO\_INJURE, // 7

AUDIO\_CANNON1, // 8

AUDIO\_CANNON2, // 9

AUDIO\_CANNON3, // 10

AUDIO\_CANNON3\_2, // 11

AUDIO\_CHARGE, // 12

AUDIO\_CHARGE\_LOOP, // 13

AUDIO\_BOOM, // 14

AUDIO\_ENEMY\_FIRE, // 15

AUDIO\_BOSS\_STAGE, // 16

AUDIO\_BOSS\_SPRINT, // 17

AUDIO\_BOSS\_SPRINT\_2, // 18

AUDIO\_BOSS\_FIRE, // 19

AUDIO\_BOSS\_CHOP, // 20

AUDIO\_BOSS\_DES, // 21

AUDIO\_BOSS\_DES\_2, // 22

AUDIO\_BOSS\_KILL, // 23

AUDIO\_BOSS\_KILL\_2, // 24

AUDIO\_BOSS\_DEAD, // 25

};

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

// 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

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap gameStartBackground; //圖片:遊戲選項主畫面

CMovingBitmap option1; //圖片:遊戲選項1(開始遊戲)

CMovingBitmap option2; //圖片:遊戲選項2(操作指南)

CMovingBitmap option3; //圖片:遊戲選項3(關於)

CMovingBitmap option4; //圖片:遊戲選項4(結束遊戲)

CMovingBitmap option1Select; //圖片:遊戲選項1選擇(開始遊戲)

CMovingBitmap option2Select; //圖片:遊戲選項2選擇(操作指南)

CMovingBitmap option3Select; //圖片:遊戲選項3選擇(關於)

CMovingBitmap option4Select; //圖片:遊戲選項4選擇(結束遊戲)

CMovingBitmap controlGuide; //圖片:操作指南畫面

CMovingBitmap about; //圖片:關於畫面

int option; //當前選項

bool startGuide; //是否開始操作指南

bool startabout; //是否開始關於畫面

bool isplaysound; //是否播放音效

};

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

// 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 PlayRockmanSound();

private:

Rockman x87\_1; //洛克人

Terrain fireDragonMap; //地圖:火龍

CInteger life; //生命值

RockCannon\* \_cannon; //洛克人子彈的陣列

Monster monster[8]; //垃圾怪

Nightmare nightmare[6]; //夢魘

Bat bat[6]; //蝙蝠

Boss zero\_fake; //夢魘傑洛

bool isWin; //是否勝利

int counterEnd; //結束遊戲計數器

bool isplay[2] = { false }; //是否播放音效

bool isplayboom; //是否播放爆炸音效

bool isplayBossStage = false; //是否進入BOSS戰鬥

CMovingBitmap youwin; // 圖片:你贏了

};

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

// 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();

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

CMovingBitmap youdead; // 圖片:你死了

};

}

< mygame.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

namespace game\_framework {

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

// 這個class為遊戲的遊戲開頭畫面物件

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

CGameStateInit::CGameStateInit(CGame\* g)

: CGameState(g)

{

}

void CGameStateInit::OnInit()

{

gameStartBackground.LoadBitmap("RES\\gamestart\\gamestartUI background.bmp");

gameStartBackground.SetTopLeft(0, 0);

option1.LoadBitmap("RES\\gamestart\\option1.bmp", RGB(255, 255, 255));

option2.LoadBitmap("RES\\gamestart\\option2.bmp", RGB(255, 255, 255));

option3.LoadBitmap("RES\\gamestart\\option3.bmp", RGB(255, 255, 255));

option4.LoadBitmap("RES\\gamestart\\option4.bmp", RGB(255, 255, 255));

option1Select.LoadBitmap("RES\\gamestart\\option1select.bmp", RGB(255, 255, 255));

option2Select.LoadBitmap("RES\\gamestart\\option2select.bmp", RGB(255, 255, 255));

option3Select.LoadBitmap("RES\\gamestart\\option3select.bmp", RGB(255, 255, 255));

option4Select.LoadBitmap("RES\\gamestart\\option4select.bmp", RGB(255, 255, 255));

option1.SetTopLeft(771, 450);

option2.SetTopLeft(771, 570);

option3.SetTopLeft(771, 690);

option4.SetTopLeft(771, 810);

option1Select.SetTopLeft(771, 450);

option2Select.SetTopLeft(771, 570);

option3Select.SetTopLeft(771, 690);

option4Select.SetTopLeft(771, 810);

about.LoadBitmap("RES\\gamestart\\about.bmp");

about.SetTopLeft(0, 0);

controlGuide.LoadBitmap("RES\\gamestart\\background.bmp");

controlGuide.SetTopLeft(0, 0);

option = 1;

CAudio::Instance()->Load(AUDIO\_OPTIONMOVE, "sounds\\optionmove.wav"); // 載入編號0的聲音ding.wav

CAudio::Instance()->Load(AUDIO\_ENTER, "sounds\\enter.wav"); // 載入編號0的聲音ding.wav

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

startGuide = false;

isplaysound = true;

// 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

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

}

void CGameStateInit::OnBeginState()

{

}

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

{

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

const char KEY\_ENTER = 0x0D;

if (nChar == KEY\_UP)

{

option--;

if (option <= 0)

option = 4;

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

}

if (nChar == KEY\_DOWN)

{

option++;

if (option > 4)

option = 1;

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

}

if (startGuide == true)

option = 2;

if (startabout == true)

option = 3;

if (nChar == KEY\_ENTER)

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

if (nChar == KEY\_ENTER && startGuide == true)

startGuide = false;

else if (nChar == KEY\_ENTER && startabout == true)

startabout = false;

else if (nChar == KEY\_ENTER && option == 1)

{

isplaysound = true;

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

}

else if (nChar == KEY\_ENTER && option == 2)

startGuide = true;

else if (nChar == KEY\_ENTER && option == 3) // Demo 關閉遊戲的方法

startabout = true; // 關閉遊戲

else if (nChar == KEY\_ENTER && option == 4) // Demo 關閉遊戲的方法

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

}

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

{

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

}

void CGameStateInit::OnShow()

{

if (isplaysound)

{

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

isplaysound = false;

}

gameStartBackground.ShowBitmap();

if (startGuide == true)

controlGuide.ShowBitmap();

else if (startabout == true)

about.ShowBitmap();

else if (option == 1)

{

option1Select.ShowBitmap();

option2.ShowBitmap();

option3.ShowBitmap();

option4.ShowBitmap();

}

else if (option == 2)

{

option1.ShowBitmap();

option2Select.ShowBitmap();

option3.ShowBitmap();

option4.ShowBitmap();

}

else if (option == 3)

{

option1.ShowBitmap();

option2.ShowBitmap();

option3Select.ShowBitmap();

option4.ShowBitmap();

}

else if (option == 4)

{

option1.ShowBitmap();

option2.ShowBitmap();

option3.ShowBitmap();

option4Select.ShowBitmap();

}

}

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

// 這個class為遊戲的結束狀態(Game Over)

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

CGameStateOver::CGameStateOver(CGame\* g)

: CGameState(g)

{

}

void CGameStateOver::OnMove()

{

counter--;

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

CAudio::Instance()->Stop(AUDIO\_BOSS\_STAGE);

CAudio::Instance()->Stop(AUDIO\_CHARGE\_LOOP);

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

if (counter < 0)

GotoGameState(GAME\_STATE\_INIT);

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 2; // 2 seconds

}

void CGameStateOver::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。

youdead.LoadBitmap("RES\\gamestart\\youdead.bmp", RGB(255, 255, 255));

youdead.SetTopLeft(0, 0);

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

// 開始載入資料

// 最終進度為100%

ShowInitProgress(100);

}

void CGameStateOver::OnShow()

{

youdead.ShowBitmap();

}

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

// 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)

{

}

CGameStateRun::~CGameStateRun()

{

}

void CGameStateRun::OnBeginState()

{

x87\_1.Initialize();

fireDragonMap.Initialize();

life.SetInteger(x87\_1.Getlife());

life.SetTopLeft(149, 468);

CAudio::Instance()->Play(AUDIO\_NTUT, true); // 撥放 MIDI

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

isplayboom = false;

isWin = false;

counterEnd = 60;

isplayBossStage = false;

}

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

{

int tmp;//洛克人座標修正值

int fixCannonY = x87\_1.GetY();

int fixCannonX = x87\_1.GetX();

int left, right, top, down;

if (isWin)

{

counterEnd--;

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

CAudio::Instance()->Stop(AUDIO\_BOSS\_STAGE);

CAudio::Instance()->Stop(AUDIO\_CHARGE\_LOOP);

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

if (counterEnd < 0)

GotoGameState(GAME\_STATE\_INIT);

return;

}

\_cannon = x87\_1.getCannon();

for (int i = 0; i < 20; i++)

{

int monster\_x, monster\_y, damage;

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

{

monster[j] = fireDragonMap.getMonster(j);

monster\_x = monster[j].getX();

monster\_y = monster[j].getY();

if (\_cannon[i].GetUsingState())

{

damage = \_cannon[i].collision(monster\_x, monster\_y, monster[j].getAlive(), 1);

if (monster[j].getAlive() == false)

damage = 0;

if (fireDragonMap.getMonsterLife(j, 1) - damage <= 0 && fireDragonMap.getMonsterLife(j, 1) > 0)

isplayboom = true;

fireDragonMap.setMonsterLife(j, damage, 1);

if (damage > 0)

{

\_cannon[i].SetUsingState(0);

break;

}

}

}

for (int j = 0; j < 6; j++)

{

nightmare[j] = fireDragonMap.getNightmare(j);

monster\_x = nightmare[j].getX();

monster\_y = nightmare[j].getY();

if (\_cannon[i].GetUsingState())

{

damage = \_cannon[i].collision(monster\_x, monster\_y, nightmare[j].getAlive(), 2);

if (nightmare[j].getAlive() == false)

damage = 0;

if (fireDragonMap.getMonsterLife(j, 2) - damage <= 0 && fireDragonMap.getMonsterLife(j, 2) > 0)

isplayboom = true;

fireDragonMap.setMonsterLife(j, damage, 2);

if (damage > 0)

{

\_cannon[i].SetUsingState(0);

break;

}

}

}

for (int j = 0; j < 6; j++)

{

bat[j] = fireDragonMap.getBat(j);

monster\_x = bat[j].getX();

monster\_y = bat[j].getY();

if (\_cannon[i].GetUsingState())

{

damage = \_cannon[i].collision(monster\_x, monster\_y, bat[j].getAlive(), 3);

if (bat[j].getAlive() == false)

damage = 0;

if (fireDragonMap.getMonsterLife(j, 3) - damage <= 0 && fireDragonMap.getMonsterLife(j, 3) > 0)

isplayboom = true;

fireDragonMap.setMonsterLife(j, damage, 3);

if (damage > 0)

{

\_cannon[i].SetUsingState(0);

break;

}

}

}

zero\_fake = fireDragonMap.getBoss();

monster\_x = zero\_fake.getX();

monster\_y = zero\_fake.getY();

if (\_cannon[i].GetUsingState())

{

damage = \_cannon[i].collision(monster\_x, monster\_y, zero\_fake.getAlive(), 4);

if (zero\_fake.getAlive() == false)

damage = 0;

fireDragonMap.setMonsterLife(0, damage, 4);

if (damage > 0)

\_cannon[i].SetUsingState(0);

if (fireDragonMap.IsBossDead())

isplayboom = true;

}

\_cannon[i].OnMove();

}

fireDragonMap.GetLastRockmanXY(x87\_1.GetX(), x87\_1.GetY());//取得移動之前座標

int injureMC = fireDragonMap.MosterCannonCollision();

if (fireDragonMap.MonsterCollision())

{

if(fireDragonMap.IsBossStage())

x87\_1.SetInjuredState(true, 4);

else

x87\_1.SetInjuredState(true, 2);

}

else if (injureMC != 0)

{

if(!fireDragonMap.GetIsBossCannon())

isplayboom = true;

x87\_1.SetInjuredState(true, injureMC);

}

x87\_1.OnMove();

fireDragonMap.GetNowRockmanXY(x87\_1.GetX(), x87\_1.GetY());//取得移動之後座標

left = fireDragonMap.crashleft();

right = fireDragonMap.crashright();

top = fireDragonMap.crashtop();

down = fireDragonMap.crashdown();

if (left == -1 || right == -1 || top == -1 || down == -1) //碰到死亡地形，GAME OVER

{

life.Add(-64);

}

if (down != 0)//判斷是否站立於障礙物上

x87\_1.SetCrashState(1);

else

x87\_1.SetCrashState(0);

if (left != 0 && top == 0 && down == 0)

x87\_1.SetCrashStateWall(1);

else if (right != 0 && top == 0 && down == 0)

x87\_1.SetCrashStateWall(2);

else

x87\_1.SetCrashStateWall(0);

if (top != 0 && down == 0)//上方碰撞到障礙物，Y座標還原

x87\_1.SetY(fireDragonMap.GetLastY());

if (down != 0 && top == 0)//下方碰撞到障礙物，Y座標修正至障礙物上方

{

tmp = down - fireDragonMap.GetLastY();

x87\_1.SetY(fireDragonMap.GetLastY() + tmp - 200);

fireDragonMap.GetNowRockmanXY(x87\_1.GetX(), x87\_1.GetY());//刷新洛克人座標

}

//卡牆BUG修正

else if (right != 0 && left != 0 && top != 0 && down != 0)

{

tmp = down - fireDragonMap.GetLastY();

x87\_1.SetY(fireDragonMap.GetLastY() + tmp - 200);

fireDragonMap.GetNowRockmanXY(x87\_1.GetX(), x87\_1.GetY());

}

if (fireDragonMap.crashright() != 0)//右方碰撞到障礙物，X座標還原

{

x87\_1.SetX(fireDragonMap.GetLastX());

}

if (fireDragonMap.crashleft() != 0)//左方碰撞到障礙物，X座標還原

{

x87\_1.SetX(fireDragonMap.GetLastX());

}

fireDragonMap.GetNowRockmanXY(x87\_1.GetX(), x87\_1.GetY());

fireDragonMap.MoveScreen();

fixCannonY -= x87\_1.GetY();

fixCannonX -= x87\_1.GetX();

x87\_1.SetFixCannonScreen(fixCannonY, fixCannonX);

if (life.GetInteger() <= 0)//血量歸0，GAME OVER

GotoGameState(GAME\_STATE\_OVER);

else if(fireDragonMap.IsBossDead())

isWin = true;

life.SetInteger(x87\_1.Getlife());

PlayRockmanSound();

}

void CGameStateRun::PlayRockmanSound()

{

if (fireDragonMap.IsBossDead() && isplayboom)

CAudio::Instance()->Play(25, false);

if (isplayboom)

{

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

isplayboom = false;

}

if (x87\_1.GetJumpDegree() == 1 && x87\_1.GetJumping())

{

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

CAudio::Instance()->Play(AUDIO\_JUMP2, false);

}

if (x87\_1.GetSprintDegree() == 1)

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

if (x87\_1.getInjuredDelay() == 1)

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

if (x87\_1.GetChargeAttack() > 0)

{

CAudio::Instance()->Stop(AUDIO\_CHARGE\_LOOP);

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

}

if (x87\_1.GetCharge() == 6)

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

if (x87\_1.GetCharge() == 50)

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

if (x87\_1.GetChargeAttack() == 0)

{

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

isplay[i] = false;

}

if (x87\_1.GetIsAttacking() && x87\_1.GetDetermineCharge() == 1)

CAudio::Instance()->Play(AUDIO\_CANNON1, false);

if (x87\_1.GetChargeAttack() > 5 && x87\_1.GetChargeAttack() <= 50 && !isplay[0])

{

CAudio::Instance()->Play(AUDIO\_CANNON2, false);

isplay[0] = true;

}

if (x87\_1.GetChargeAttack() > 50 && !isplay[1])

{

CAudio::Instance()->Play(AUDIO\_CANNON3, false);

CAudio::Instance()->Play(AUDIO\_CANNON3\_2, false);

isplay[1] = true;

}

if(fireDragonMap.IsBossStage()&& !isplayBossStage)

{

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

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

isplayBossStage = true;

}

}

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%

// 開始載入資料

// 完成部分Loading動作，提高進度

ShowInitProgress(50);

// 繼續載入其他資料

x87\_1.LoadBitmap();

x87\_1.LoadAttackBitmap();

fireDragonMap.LoadBitMap();

youwin.LoadBitmap("RES\\gamestart\\win.bmp");

youwin.SetTopLeft(0, 0);

life.LoadBitmap();

CAudio::Instance()->Load(AUDIO\_NTUT, "sounds\\Jakob.wav");

CAudio::Instance()->Load(AUDIO\_JUMP, "sounds\\jump.wav");

CAudio::Instance()->Load(AUDIO\_JUMP2, "sounds\\jump2.wav");

CAudio::Instance()->Load(AUDIO\_SPRINT, "sounds\\sprint.wav");

CAudio::Instance()->Load(AUDIO\_INJURE, "sounds\\injure.wav");

CAudio::Instance()->Load(AUDIO\_CANNON1, "sounds\\cannon1.wav");

CAudio::Instance()->Load(AUDIO\_CANNON2, "sounds\\cannon2.wav");

CAudio::Instance()->Load(AUDIO\_CANNON3, "sounds\\cannon3.wav");

CAudio::Instance()->Load(AUDIO\_CANNON3\_2, "sounds\\cannon3-2.wav");

CAudio::Instance()->Load(AUDIO\_CHARGE, "sounds\\charge.wav");

CAudio::Instance()->Load(AUDIO\_CHARGE\_LOOP, "sounds\\charge-loop.wav");

CAudio::Instance()->Load(AUDIO\_BOOM, "sounds\\boom.wav");

CAudio::Instance()->Load(AUDIO\_ENEMY\_FIRE, "sounds\\enemyfire.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_STAGE, "sounds\\boss\_stage.mp3");

CAudio::Instance()->Load(AUDIO\_BOSS\_SPRINT, "sounds\\boss\\bossSprint.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_SPRINT\_2, "sounds\\boss\\sprint\_zero.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_FIRE, "sounds\\boss\\fire.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_CHOP, "sounds\\boss\\chop.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_DES, "sounds\\boss\\destroylight.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_DES\_2, "sounds\\boss\\destroylight\_zero.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_KILL, "sounds\\boss\\kill.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_KILL\_2, "sounds\\boss\\kill\_zero.wav");

CAudio::Instance()->Load(AUDIO\_BOSS\_DEAD, "sounds\\boss\\dead\_zero.wav");

//

// 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)

{

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\_JUMP = 0x5A; // keyboard按鍵Z

const char KEY\_SPRINT = 0x20; // keyboard空白鍵

const char KEY\_ATTACK = 0x58; // keyboard按鍵X

const char KEY\_CHEAT = 0x43; // keyboard按鍵C

const char KEY\_EXIT = 0x1B; // keyboard按鍵ESC

if (nChar == KEY\_LEFT)

x87\_1.SetMovingLeft(true);

if (nChar == KEY\_RIGHT)

x87\_1.SetMovingRight(true);

if ((nChar == KEY\_JUMP) && (x87\_1.GetJumpDegree() == 0))

x87\_1.SetJumping(true);

if ((nChar == KEY\_SPRINT) && (x87\_1.GetSprintDegree() == 0))

x87\_1.SetSprinting(true);

if (nChar == KEY\_SPRINT)

x87\_1.SetKeySprintState(true);

if ((nChar == KEY\_ATTACK) && (x87\_1.GetCharge() == 0))

x87\_1.SetAttacking(true);

else

x87\_1.SetAttacking(false);

if (nChar == KEY\_ATTACK)

x87\_1.SetKeyAttackingState(true);

if (nChar == KEY\_CHEAT)

{

x87\_1.SetLife(64);

fireDragonMap.setLifeToZero();

}

if (nChar == KEY\_EXIT)

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

}

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

{

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

const char KEY\_JUMP = 0x5A; // keyboard按鍵Z

const char KEY\_SPRINT = 0x20; // keyboard空白鍵

const char KEY\_ATTACK = 0x58; // keyboard按鍵X

if (nChar == KEY\_LEFT)

x87\_1.SetMovingLeft(false);

if (nChar == KEY\_RIGHT)

x87\_1.SetMovingRight(false);

if ((nChar == KEY\_JUMP) && (x87\_1.GetJumpDegree() > 0))

x87\_1.SetJumping(false);

if ((nChar == KEY\_SPRINT) && (x87\_1.GetSprintDegree() > 0))

x87\_1.SetSprinting(false);

if (nChar == KEY\_SPRINT)

x87\_1.SetKeySprintState(false);

if (nChar == KEY\_ATTACK)

x87\_1.SetAttacking(false);

if (nChar == KEY\_ATTACK)

x87\_1.SetKeyAttackingState(false);

}

//-------------------------------------------------------------------

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

{

}

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

{

// 沒事。如果需要處理滑鼠移動的話，寫code在這裡

}

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。

//

// 貼上背景圖、撞擊數、球、擦子、彈跳的球

//

if(isWin)

{

youwin.ShowBitmap();

return;

}

fireDragonMap.OnShow();

x87\_1.OnShow();

life.ShowBitmap();

}

}

< Bat.h >

namespace game\_framework

{

class Bat

{

public:

Bat();

void Initialize(); //初始化參數

void LoadBitMap(); //載入圖片

void setXY(int nx, int ny); //設定XY

void setScreenXY(int nx, int ny); //設定螢幕XY

int getX(); //取得蝙蝠X座標

int getY(); //取得蝙蝠Y座標

int getScreenX(); //取得蝙蝠X螢幕座標

int getScreenY(); //取得蝙蝠Y螢幕座標

void setAlive(bool state); //設定存活狀態

bool getAlive(); //取得存活狀態

int getLife(); //取得生命值

void deductLife(int damage); //扣除生命值

void OnShow(); //顯示動畫

void OnShowBoom(); //顯示爆炸動畫

void DeterminAttack(int, int); //決定攻擊模式

bool MonsterCollision(int, int); //與洛克人碰撞判斷

private:

int x, y; //座標

int s\_x, s\_y; //螢幕座標偏移量

int life; //生命

bool isAlive; //存活狀態

CAnimation monsMovingRight; //動畫:往右移動

CAnimation monsMovingLeft; //動畫:往左移動

CAnimation monsBoom; //動畫:爆炸

void LoadMovingRightBitMap(); //載入動畫:往右移動

void LoadMovingLeftBitMap(); //載入動畫:往左移動

void LoadBoomBitmap(); //載入所有動畫

bool AttackDirection; //攻擊方向,0:往左射,1:往右射

int AttackMode; //攻擊模式 0:靜止 1:移動

void moveAttack(int, int); //衝撞攻擊

};

}

< Bat.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Bat.h"

namespace game\_framework

{

Bat::Bat()

{}

void Bat::Initialize()

{

x = y = 0;

s\_x = s\_y = 0;

life = 4;

isAlive = true;

AttackDirection = 0;

AttackMode = 0;

monsBoom.Reset();

}

void Bat::LoadBitMap()

{

LoadMovingRightBitMap();

LoadMovingLeftBitMap();

LoadBoomBitmap();

}

void Bat::LoadMovingRightBitMap()

{

monsMovingRight.AddBitmap("RES\\enemy\\batmoving.bmp", RGB(255, 255, 255));

monsMovingRight.AddBitmap("RES\\enemy\\batmoving2.bmp", RGB(255, 255, 255));

monsMovingRight.AddBitmap("RES\\enemy\\batmoving3.bmp", RGB(255, 255, 255));

monsMovingRight.AddBitmap("RES\\enemy\\batmoving4.bmp", RGB(255, 255, 255));

monsMovingRight.AddBitmap("RES\\enemy\\batmoving5.bmp", RGB(255, 255, 255));

monsMovingRight.AddBitmap("RES\\enemy\\batmoving6.bmp", RGB(255, 255, 255));

monsMovingRight.SetDelayCount(2);

}

void Bat::LoadMovingLeftBitMap()

{

monsMovingLeft.AddBitmap("RES\\enemy\\batmovingleft.bmp", RGB(255, 255, 255));

monsMovingLeft.AddBitmap("RES\\enemy\\batmoving2left.bmp", RGB(255, 255, 255));

monsMovingLeft.AddBitmap("RES\\enemy\\batmoving3left.bmp", RGB(255, 255, 255));

monsMovingLeft.AddBitmap("RES\\enemy\\batmoving4left.bmp", RGB(255, 255, 255));

monsMovingLeft.AddBitmap("RES\\enemy\\batmoving5left.bmp", RGB(255, 255, 255));

monsMovingLeft.AddBitmap("RES\\enemy\\batmoving6left.bmp", RGB(255, 255, 255));

monsMovingLeft.SetDelayCount(2);

}

void Bat::LoadBoomBitmap()

{

monsBoom.AddBitmap("RES\\enemy\\explosion.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion2.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion3.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion4.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion5.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion6.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion7.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion8.bmp", RGB(255, 255, 255));

monsBoom.SetDelayCount(2);

}

void Bat::setXY(int nx, int ny)

{

x = nx;

y = ny;

}

void Bat::setScreenXY(int nx, int ny)

{

s\_x = nx;

s\_y = ny;

}

void Bat::setAlive(bool state)

{

isAlive = state;

}

int Bat::getX()

{

return x;

}

int Bat::getY()

{

return y;

}

int Bat::getScreenX()

{

return s\_x;

}

int Bat::getScreenY()

{

return s\_y;

}

bool Bat::getAlive()

{

return isAlive;

}

int Bat::getLife()

{

return life;

}

void Bat::deductLife(int damage)

{

life -= damage;

if (life <= 0)

isAlive = false;

}

void Bat::moveAttack(int RockX, int RockY)

{

if (RockX <= x)

x -= 30;

else if (RockX > x)

x += 30;

if (RockY <= y)

y -= 20;

else if (RockY > y)

y += 20;

}

void Bat::DeterminAttack(int RockX, int RockY)

{

monsMovingRight.OnMove();

monsMovingLeft.OnMove();

if (((RockX - x) < 1000 && (RockX - x) > 0) || ((x - RockX) < 1000 && (x - RockX) > 0) || AttackMode == 1)

{

AttackMode = 1;

if (isAlive)

moveAttack(RockX, RockY);

}

if (x - RockX > 0)

AttackDirection = 0;

else if (RockX - x > 0)

AttackDirection = 1;

}

bool Bat::MonsterCollision(int RockX, int RockY)

{

if (!isAlive)

return false;

else if (x + 120 > RockX && RockX + 160 > x && y + 160 > RockY && RockY > y)

return true;

else

return false;

}

void Bat::OnShow()

{

if (AttackDirection == 0)

{

monsMovingRight.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsMovingRight.OnShow();

}

else if (AttackDirection == 1)

{

monsMovingLeft.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsMovingLeft.OnShow();

}

}

void Bat::OnShowBoom()

{

if (monsBoom.IsFinalBitmap())

monsBoom.setToSpecifyBitmap(7);

monsBoom.OnMove();

monsBoom.SetTopLeft(x + s\_x, y + s\_y - 1492);

if (isAlive == false && !monsBoom.IsFinalBitmap())

monsBoom.OnShow();

}

}

< Monster.h >

namespace game\_framework

{

class Monster

{

public:

Monster();

void Initialize(); //初始化參數

void LoadBitMap(); //載入圖片

void setXY(int nx, int ny); //設定XY

void setScreenXY(int nx, int ny); //設定螢幕XY

void FixCannonScreenXY(int, int); //修正砲彈螢幕XY

int getX(); //取得蝙蝠X座標

int getY(); //取得蝙蝠Y座標

int getScreenX(); //取得蝙蝠X螢幕座標

int getScreenY(); //取得蝙蝠Y螢幕座標

void setAlive(bool state); //設定存活狀態

bool getAlive(); //取得存活狀態

int getLife(); //取得生命值

void deductLife(int damage); //扣除生命值

void OnShow(); //顯示動畫

void OnShowBoom(); //顯示爆炸動畫

void DeterminAttack(int, int); //決定攻擊模式

bool MonsterCollision(int, int); //與洛克人碰撞判斷

int MonsterCannonCollision(int, int); //砲擊與洛克人碰撞判斷

private:

int x, y; //座標

int s\_x, s\_y; //螢幕座標

int life; //生命

bool isAlive; //存活狀態

CMovingBitmap monsRight; //圖片:往右站立

CMovingBitmap monsLeft; //圖片:往左站立

CMovingBitmap monsRightAttack; //圖片:往右攻擊

CMovingBitmap monsLeftAttack; //圖片:往左攻擊

CAnimation monsBoom; //動畫:爆炸

void LoadBoomBitmap(); //載入動畫:爆炸

TrashCannon cannon; //垃圾砲

bool AttackDirection; //攻擊方向,0:往左射,1:往右射

int AttackDelay; //攻擊延遲

int nowAttackDistance; //當前攻擊距離

};

}

< Monster.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Trashcannon.h"

#include "Monster.h"

namespace game\_framework

{

Monster::Monster()

{}

void Monster::Initialize()

{

x = y = 0;

s\_x = s\_y = 0;

life = 10;

isAlive = true;

AttackDirection = 0;

AttackDelay = 49;

monsBoom.Reset();

}

void Monster::LoadBitMap()

{

monsRight.LoadBitmap("RES\\enemy\\enemy\_X6\_1.bmp", RGB(255, 255, 255));

monsLeft.LoadBitmap("RES\\enemy\\enemy\_X6\_1left.bmp", RGB(255, 255, 255));

monsRightAttack.LoadBitmap("RES\\enemy\\enemy\_X6\_1 attack.bmp", RGB(255, 255, 255));

monsLeftAttack.LoadBitmap("RES\\enemy\\enemy\_X6\_1 attackleft.bmp", RGB(255, 255, 255));

LoadBoomBitmap();

cannon.LoadBitmapA();

}

void Monster::LoadBoomBitmap()

{

monsBoom.AddBitmap("RES\\enemy\\explosion.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion2.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion3.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion4.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion5.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion6.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion7.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion8.bmp", RGB(255, 255, 255));

monsBoom.SetDelayCount(2);

}

void Monster::setXY(int nx, int ny)

{

x = nx;

y = ny;

}

void Monster::setScreenXY(int nx, int ny)

{

s\_x = nx;

s\_y = ny;

}

void Monster::setAlive(bool state)

{

isAlive = state;

}

int Monster::getX()

{

return x;

}

int Monster::getY()

{

return y;

}

int Monster::getScreenX()

{

return s\_x;

}

int Monster::getScreenY()

{

return s\_y;

}

void Monster::FixCannonScreenXY(int fixX, int fixY)

{

cannon.AddScreenX\_fix(-fixX);

cannon.AddScreenY\_fix(-fixY);

}

bool Monster::getAlive()

{

return isAlive;

}

int Monster::getLife()

{

return life;

}

void Monster::deductLife(int damage)

{

life -= damage;

if (life <= 0)

isAlive = false;

}

void Monster::DeterminAttack(int RockX, int RockY)

{

if ((x - RockX > 400 || RockX - x > 400) && AttackDelay < 49)

AttackDelay++;

else if (((x - RockX < 400) && (x - RockX > 0)) || ((RockX - x < 400) && (RockX - x > 0)))

AttackDelay++;

if (x - RockX > 0)

AttackDirection = 0;

else if (RockX - x > 0)

AttackDirection = 1;

if (AttackDelay == 50 && isAlive)

{

AttackDelay = 0;

cannon.SetUsingState(true);

cannon.SetX(x + 121);

cannon.SetY(y + 48);

cannon.SetScreenXY(s\_x + 121, s\_y + 48 - 1492);

cannon.SetLastMovingState(AttackDirection);

cannon.SetCatchAction(1);

}

cannon.OnMove();

}

bool Monster::MonsterCollision(int RockX, int RockY)

{

if (!isAlive)

return false;

else if (x + 254 > RockX && RockX + 160 > x && y + 225 > RockY && RockY > y)

return true;

else

return false;

}

int Monster::MonsterCannonCollision(int RockX, int RockY)

{

if(isAlive)

return cannon.collision(RockX, RockY);

return false;

}

void Monster::OnShow()

{

if (AttackDelay < 5 && AttackDirection == 0)

{

monsRightAttack.SetTopLeft(s\_x, s\_y - 1492);

monsRightAttack.ShowBitmap();

}

else if (AttackDelay < 5 && AttackDirection == 1)

{

monsLeftAttack.SetTopLeft(s\_x, s\_y - 1492);

monsLeftAttack.ShowBitmap();

}

else if (AttackDirection == 0)

{

monsRight.SetTopLeft(s\_x, s\_y - 1492);

monsRight.ShowBitmap();

}

else if (AttackDirection == 1)

{

monsLeft.SetTopLeft(s\_x, s\_y - 1492);

monsLeft.ShowBitmap();

}

if (cannon.GetUsingState())

cannon.OnShow();

else

cannon.OnShowHit();

}

void Monster::OnShowBoom()

{

if (monsBoom.IsFinalBitmap())

monsBoom.setToSpecifyBitmap(7);

monsBoom.OnMove();

monsBoom.SetTopLeft(s\_x, s\_y - 1492);

if (isAlive == false && !monsBoom.IsFinalBitmap())

monsBoom.OnShow();

}

}

< nightmare.h >

namespace game\_framework

{

class Nightmare

{

public:

Nightmare();

void Initialize(); //初始化參數

void LoadBitMap(); //載入圖片

void setXY(int nx, int ny); //設定XY

void setScreenXY(int nx, int ny); //設定螢幕XY

void FixCannonScreenXY(int, int); //修正螢幕座標

int getX(); //取得夢魘X座標

int getY(); //取得夢魘Y座標

int getScreenX(); //取得夢魘X螢幕座標

int getScreenY(); //取得夢魘Y螢幕座標

void setAlive(bool state); //設定存活狀態

bool getAlive(); //取得存活狀態

int getLife(); //取得生命值

void deductLife(int damage); //扣除生命值

void OnShow(); //顯示動畫

void OnShowBoom(); //顯示爆炸動畫

void DeterminAttack(int, int); //決定攻擊模式

bool MonsterCollision(int, int); //與洛克人碰撞判斷

int MonsterCannonCollision(int, int);//砲彈與洛克人碰撞判斷

private:

int x, y; //座標

int s\_x, s\_y; //螢幕座標偏移量

int life; //生命

bool isAlive; //存活狀態

CAnimation monsRight; //動畫:向右站立

CAnimation monsLeft; //動畫:往左站立

CAnimation monsMovingRight; //動畫:向右移動

CAnimation monsMovingLeft; //動畫:往左移動

CAnimation monsRightAttack; //動畫:向右射擊

CAnimation monsLeftAttack; //動畫:往左射擊

CAnimation monsBoom; //動畫:爆炸

void LoadRightBitMap(); //載入動畫:向右站立

void LoadLeftBitMap(); //載入動畫:往左站立

void LoadMovingRightBitMap(); //載入動畫:向右移動

void LoadMovingLeftBitMap(); //載入動畫:往左移動

void LoadattackRightBitMap(); //載入動畫:向右射擊

void LoadattackLeftBitMap(); //載入動畫:往左射擊

void LoadBoomBitmap(); //載入動畫:爆炸

NightmareCannon cannon; //夢魘砲

int AttackDelay; //砲彈攻擊延遲

int moveDelay; //移動攻擊延遲

int determinActionDelay; //決定攻擊模式延遲

bool AttackDirection; //攻擊方向,0:往左射,1:往右射

int AttackMode; //攻擊模式 0:靜止 1:移動 2:開火

void moveAttack(int, int); //攻擊模式:衝撞攻擊

void fire(int, int); //攻擊模式:砲擊

void stay(); //攻擊模式:待機

bool firesound; //是否播放射擊音效

};

}

< nightmare.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "nightmareCannon.h"

#include "nightmare.h"

namespace game\_framework

{

Nightmare::Nightmare()

{}

void Nightmare::Initialize()

{

x = y = 0;

s\_x = s\_y = 0;

life = 10;

isAlive = true;

AttackDelay = 49;

determinActionDelay = 0;

moveDelay = 0;

AttackDirection = 0;

AttackMode = 0;

monsBoom.Reset();

firesound = false;

if (rand() % 2)

moveDelay = 200;

else

AttackDelay = 200;

}

void Nightmare::LoadBitMap()

{

LoadRightBitMap();

LoadLeftBitMap();

LoadMovingRightBitMap();

LoadMovingLeftBitMap();

LoadattackRightBitMap();

LoadattackLeftBitMap();

LoadBoomBitmap();

cannon.LoadBitmapA();

}

void Nightmare::LoadRightBitMap()

{

monsRight.AddBitmap("RES\\enemy\\enemy\_X6\_2.bmp", RGB(255, 255, 255));

monsRight.AddBitmap("RES\\enemy\\enemy\_X6\_2 2.bmp", RGB(255, 255, 255));

monsRight.AddBitmap("RES\\enemy\\enemy\_X6\_2 3.bmp", RGB(255, 255, 255));

monsRight.AddBitmap("RES\\enemy\\enemy\_X6\_2 4.bmp", RGB(255, 255, 255));

monsRight.SetDelayCount(4);

}

void Nightmare::LoadLeftBitMap()

{

monsLeft.AddBitmap("RES\\enemy\\enemy\_X6\_2left.bmp", RGB(255, 255, 255));

monsLeft.AddBitmap("RES\\enemy\\enemy\_X6\_2 2left.bmp", RGB(255, 255, 255));

monsLeft.AddBitmap("RES\\enemy\\enemy\_X6\_2 3left.bmp", RGB(255, 255, 255));

monsLeft.AddBitmap("RES\\enemy\\enemy\_X6\_2 4left.bmp", RGB(255, 255, 255));

monsLeft.SetDelayCount(4);

}

void Nightmare::LoadMovingRightBitMap()

{

monsMovingRight.AddBitmap("RES\\enemy\\enemy\_X6\_2 move.bmp", RGB(255, 255, 255));

monsMovingRight.AddBitmap("RES\\enemy\\enemy\_X6\_2 move2.bmp", RGB(255, 255, 255));

monsMovingRight.AddBitmap("RES\\enemy\\enemy\_X6\_2 move3.bmp", RGB(255, 255, 255));

monsMovingRight.SetDelayCount(4);

}

void Nightmare::LoadMovingLeftBitMap()

{

monsMovingLeft.AddBitmap("RES\\enemy\\enemy\_X6\_2 moveleft.bmp", RGB(255, 255, 255));

monsMovingLeft.AddBitmap("RES\\enemy\\enemy\_X6\_2 move2left.bmp", RGB(255, 255, 255));

monsMovingLeft.AddBitmap("RES\\enemy\\enemy\_X6\_2 move3left.bmp", RGB(255, 255, 255));

monsMovingLeft.SetDelayCount(4);

}

void Nightmare::LoadattackRightBitMap()

{

monsRightAttack.AddBitmap("RES\\enemy\\enemy\_X6\_2 attack.bmp", RGB(255, 255, 255));

monsRightAttack.AddBitmap("RES\\enemy\\enemy\_X6\_2 attack2.bmp", RGB(255, 255, 255));

monsRightAttack.AddBitmap("RES\\enemy\\enemy\_X6\_2 attack3.bmp", RGB(255, 255, 255));

monsRightAttack.SetDelayCount(4);

}

void Nightmare::LoadattackLeftBitMap()

{

monsLeftAttack.AddBitmap("RES\\enemy\\enemy\_X6\_2 attackleft.bmp", RGB(255, 255, 255));

monsLeftAttack.AddBitmap("RES\\enemy\\enemy\_X6\_2 attack2left.bmp", RGB(255, 255, 255));

monsLeftAttack.AddBitmap("RES\\enemy\\enemy\_X6\_2 attack3left.bmp", RGB(255, 255, 255));

monsLeftAttack.SetDelayCount(4);

}

void Nightmare::LoadBoomBitmap()

{

monsBoom.AddBitmap("RES\\enemy\\explosion.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion2.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion3.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion4.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion5.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion6.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion7.bmp", RGB(255, 255, 255));

monsBoom.AddBitmap("RES\\enemy\\explosion8.bmp", RGB(255, 255, 255));

monsBoom.SetDelayCount(2);

}

void Nightmare::setXY(int nx, int ny)

{

x = nx;

y = ny;

}

void Nightmare::setScreenXY(int nx, int ny)

{

s\_x = nx;

s\_y = ny;

}

void Nightmare::setAlive(bool state)

{

isAlive = state;

}

int Nightmare::getX()

{

return x;

}

int Nightmare::getY()

{

return y;

}

int Nightmare::getScreenX()

{

return s\_x;

}

int Nightmare::getScreenY()

{

return s\_y;

}

void Nightmare::FixCannonScreenXY(int fixX, int fixY)

{

cannon.AddScreenX\_fix(-fixX);

cannon.AddScreenY\_fix(-fixY);

}

bool Nightmare::getAlive()

{

return isAlive;

}

int Nightmare::getLife()

{

return life;

}

void Nightmare::deductLife(int damage)

{

life -= damage;

if (life <= 0)

isAlive = false;

}

void Nightmare::moveAttack(int RockX, int RockY)

{

monsMovingRight.OnMove();

monsMovingLeft.OnMove();

moveDelay++;

if (RockX <= x)

x -= 20;

else if (RockX > x)

x += 20;

if (RockY <= y)

y -= 10;

else if (RockY > y)

y += 10;

}

void Nightmare::fire(int RockX, int RockY)

{

monsRightAttack.OnMove();

monsLeftAttack.OnMove();

AttackDelay++;

if (AttackDelay % 35 == 0 && isAlive)

{

cannon.SetUsingState(true);

cannon.SetX(x + 40);

cannon.SetY(y + 36);

cannon.SetScreenXY(x + s\_x + 40, y + s\_y + 36 - 1492);

cannon.SetVelocity(RockX + 80, RockY + 100);

cannon.SetLastMovingState(AttackDirection);

cannon.SetCatchAction(1);

firesound = true;

}

}

void Nightmare::stay()

{

determinActionDelay++;

if (determinActionDelay >= 50)

{

if (AttackDelay == 0 && moveDelay != 0)

{

AttackMode = 2;

moveDelay = 0;

}

else if (moveDelay == 0 && AttackDelay != 0)

{

AttackMode = 1;

AttackDelay = 0;

}

determinActionDelay = 0;

}

}

void Nightmare::DeterminAttack(int RockX, int RockY)

{

if ((AttackDelay == 200 || moveDelay == 200) && AttackMode != 0)

AttackMode = 0;

if (AttackMode == 0 && AttackDelay == 200)

moveDelay = 0;

else if(AttackMode == 0 && moveDelay == 200)

AttackDelay = 0;

if (AttackMode != 2)

{

monsRightAttack.Reset();

monsLeftAttack.Reset();

}

if (((RockX - x) < 1000 && (RockX - x) > 0) || ((x - RockX) < 1000 && (x - RockX) > 0)|| AttackMode != 0)

{

if(isAlive)

{

if (AttackMode == 0)

stay();

else if (AttackMode == 1)

moveAttack(RockX, RockY);

else if (AttackMode == 2)

fire(RockX, RockY);

}

}

monsRight.OnMove();

monsLeft.OnMove();

if (x - RockX > 0)

AttackDirection = 0;

else if (RockX - x > 0)

AttackDirection = 1;

cannon.OnMove();

if (firesound)

{

CAudio::Instance()->Play(15, false);

firesound = false;

}

}

bool Nightmare::MonsterCollision(int RockX, int RockY)

{

if (!isAlive)

return false;

else if (x + 196 > RockX && RockX + 160 > x && y + 90 > RockY && RockY > y)

return true;

else

return false;

}

int Nightmare::MonsterCannonCollision(int RockX, int RockY)

{

if (isAlive)

return cannon.collision(RockX, RockY);

return false;

}

void Nightmare::OnShow()

{

if (AttackMode == 2 && AttackDirection == 0)

{

monsRightAttack.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsRightAttack.OnShow();

}

else if (AttackMode == 2 && AttackDirection == 1)

{

monsLeftAttack.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsLeftAttack.OnShow();

}

else if (AttackMode == 1 && AttackDirection == 0)

{

monsMovingRight.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsMovingRight.OnShow();

}

else if (AttackMode == 1 && AttackDirection == 1)

{

monsMovingLeft.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsMovingLeft.OnShow();

}

else if (AttackDirection == 0)

{

monsRight.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsRight.OnShow();

}

else if (AttackDirection == 1)

{

monsLeft.SetTopLeft(x + s\_x, y + s\_y - 1492);

monsLeft.OnShow();

}

if (cannon.GetUsingState())

cannon.OnShow();

else

cannon.OnShowHit();

}

void Nightmare::OnShowBoom()

{

if (monsBoom.IsFinalBitmap())

monsBoom.setToSpecifyBitmap(7);

monsBoom.OnMove();

monsBoom.SetTopLeft(x + s\_x, y + s\_y - 1492);

if (isAlive == false && !monsBoom.IsFinalBitmap())

monsBoom.OnShow();

}

}

< Boss.h >

namespace game\_framework

{

class Boss

{

public:

Boss();

void Initialize(); //初始化參數

void LoadBitMap(); //載入圖片

void setXY(int nx, int ny); //設定XY

void setScreen\_XY(int nx, int ny); //設定螢幕XY

void FixCannonScreenXY(int, int); //修正砲彈螢幕XY

int getX(); //取得夢魘傑洛X座標

int getY(); //取得夢魘傑洛Y座標

int getScreenX(); //取得夢魘傑洛X螢幕座標

int getScreenY(); //取得夢魘傑洛Y螢幕座標

bool getAlive(); //取得存活狀態

int getLife(); //取得生命值

void deductLife(int damage); //扣除生命值

void OnShow(); //顯示動畫

void OnMove(int RockX, int RockY); //角色移動

void DeterminAttack(int RockX, int RockY); //決定攻擊模式

bool GetStartAttack(); //取得角色是否開始攻擊

bool MonsterCollision(int, int); //與洛克人碰撞判斷

int MonsterCannonCollision(int, int); //砲擊與洛克人碰撞判斷

private:

int x, y; //座標

int screen\_x, screen\_y; //螢幕座標偏移量

int life; //生命

bool isAlive; //存活狀態

CMovingBitmap boss; //圖片:向右站立

CMovingBitmap bossLeft; //圖片:往左站立

CAnimation bossSprint; //動畫:向右衝刺

CAnimation bossSprintLeft; //動畫:往左衝刺

CAnimation bossFire; //動畫:向右射擊

CAnimation bossFireLeft; //動畫:往左射擊

CAnimation bossChop; //動畫:向右斬擊

CAnimation bossChopLeft; //動畫:往左斬擊

CAnimation bossDestoryLight; //動畫:滅閃光(槌地板)

CAnimation bossKill; //動畫:幻夢零(右)(超大劍氣)

CAnimation bossKillLeft; //動畫:幻夢零(左)(超大劍氣)

CAnimation hugeknife; //動畫:幻夢零充能

CAnimation knife; //動畫:光束劍(右)

CAnimation knifeLeft; //動畫:光束劍(左)

CMovingBitmap lifeItem; //圖片:生命條

CMovingBitmap lifeValue; //圖片:生命值

void LoadBossSprintBitMap(); //載入動畫:向右衝刺

void LoadBossSprintLeftBitMap(); //載入動畫:往左衝刺

void LoadBossFireBitMap(); //載入動畫:向右射擊

void LoadBossFireLeftBitMap(); //載入動畫:往左射擊

void LoadBossChopBitMap(); //載入動畫:向右斬擊

void LoadBossChopLeftBitMap(); //載入動畫:往左斬擊

void LoadBossDestoryLightBitMap(); //載入動畫:滅閃光(槌地板)

void LoadBossKillBitMap(); //載入動畫:幻夢零(右)(超大劍氣)

void LoadBossKillLeftBitMap(); //載入動畫:幻夢零(左)(超大劍氣)

void LoadHugeknifeBitMap(); //載入動畫:幻夢零充能

void LoadKnifeBitMap(); //載入動畫:光束劍(右)

void LoadKnifeLeftBitMap(); //載入動畫:光束劍(左)

void LoadLifeObjectBitmap(); //載入圖片:生命物件

void ResetAnimation(); //重置動畫

BossCannon cannon[6]; //夢魘傑洛砲彈

bool startAttack; //是否開始攻擊

bool isfire; //是否已射擊

int attackDelay; //攻擊延遲

int AttackDirection; //攻擊方向,0:往左射,1:往右射

int KillDirection; //幻夢零(超大劍氣)攻擊方向

int AttackCommand[15]; //攻擊指令

int commandProcess; //指令進程

int reverse; //攻擊方向反轉

int skill; //當前攻擊的技能

void Stay(); //攻擊模式:待機

void Sprint(); //攻擊模式:衝刺

void Fire(); //攻擊模式:砲擊

void FireFlySword(int RockX, int RockY); //攻擊模式:追蹤飛刀

void DestroyLight(); //攻擊模式:滅閃光(槌地板)

void Kill(); //攻擊模式:幻夢零(超大劍氣)

bool isInjured; //是否已經受傷

int injureDelay; //受傷無敵時間

int injureShine; //無敵時間閃爍

bool isPlaySound; //是否撥放音效

int destroyLightCount; //滅閃光(槌地板)音效計數器

};

}

< Boss.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "BossCannon.h"

#include "Boss.h"

namespace game\_framework

{

Boss::Boss()

{

}

void Boss::Initialize()

{

x = y = 0;

screen\_x = screen\_y = 0;

life = 48;

isAlive = true;

startAttack = false;

isfire = false;

attackDelay = 0;

AttackDirection = 0;

KillDirection = 0;

AttackCommand[0] = 0;

AttackCommand[1] = 2;

AttackCommand[2] = 2;

AttackCommand[3] = 3;

AttackCommand[4] = 0;

AttackCommand[5] = 2;

AttackCommand[6] = 2;

AttackCommand[7] = 3;

AttackCommand[8] = 1;

AttackCommand[9] = 0;

AttackCommand[10] = 4;

AttackCommand[11] = 0;

AttackCommand[12] = 5;

AttackCommand[13] = 0;

AttackCommand[14] = 1;

commandProcess = 0;

reverse = 0;

skill = 0;

isInjured = false;

injureDelay = 30;

injureShine = 3;

isPlaySound = false;

destroyLightCount = 0;

for (int i = 0; i < 6; i++)

cannon[i].SetUsingState(false);

}

void Boss::LoadHugeknifeBitMap()

{

hugeknife.AddBitmap("RES\\boss\\Kill.bmp", RGB(255, 255, 255));

hugeknife.AddBitmap("RES\\boss\\Kill2.bmp", RGB(255, 255, 255));

hugeknife.AddBitmap("RES\\boss\\Kill3.bmp", RGB(255, 255, 255));

hugeknife.AddBitmap("RES\\boss\\Kill4.bmp", RGB(255, 255, 255));

hugeknife.AddBitmap("RES\\boss\\Kill5.bmp", RGB(255, 255, 255));

hugeknife.SetDelayCount(2);

}

void Boss::LoadBossDestoryLightBitMap()

{

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight2.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight3.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight4.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight5.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight6.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight7.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight8.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight9.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight10.bmp", RGB(255, 255, 255));

bossDestoryLight.AddBitmap("RES\\boss\\bossdestroylight11.bmp", RGB(255, 255, 255));

bossDestoryLight.SetDelayCount(4);

}

void Boss::LoadBossSprintBitMap()

{

bossSprint.AddBitmap("RES\\boss\\bosssprint.bmp", RGB(255, 255, 255));

bossSprint.AddBitmap("RES\\boss\\bosssprint2.bmp", RGB(255, 255, 255));

bossSprint.AddBitmap("RES\\boss\\bosssprint3.bmp", RGB(255, 255, 255));

bossSprint.AddBitmap("RES\\boss\\bosssprint4.bmp", RGB(255, 255, 255));

bossSprint.AddBitmap("RES\\boss\\bosssprint5.bmp", RGB(255, 255, 255));

bossSprint.AddBitmap("RES\\boss\\bosssprint6.bmp", RGB(255, 255, 255));

bossSprint.AddBitmap("RES\\boss\\bosssprint7.bmp", RGB(255, 255, 255));

bossSprint.SetDelayCount(4);

}

void Boss::LoadBossSprintLeftBitMap()

{

bossSprintLeft.AddBitmap("RES\\boss\\bossSprintleft.bmp", RGB(255, 255, 255));

bossSprintLeft.AddBitmap("RES\\boss\\bossSprint2left.bmp", RGB(255, 255, 255));

bossSprintLeft.AddBitmap("RES\\boss\\bossSprint3left.bmp", RGB(255, 255, 255));

bossSprintLeft.AddBitmap("RES\\boss\\bossSprint4left.bmp", RGB(255, 255, 255));

bossSprintLeft.AddBitmap("RES\\boss\\bossSprint5left.bmp", RGB(255, 255, 255));

bossSprintLeft.AddBitmap("RES\\boss\\bossSprint6left.bmp", RGB(255, 255, 255));

bossSprintLeft.AddBitmap("RES\\boss\\bossSprint7left.bmp", RGB(255, 255, 255));

bossSprintLeft.SetDelayCount(4);

}

void Boss::LoadBossFireBitMap()

{

bossFire.AddBitmap("RES\\boss\\bossfire.bmp", RGB(255, 255, 255));

bossFire.AddBitmap("RES\\boss\\bossfire2.bmp", RGB(255, 255, 255));

bossFire.AddBitmap("RES\\boss\\bossfire3.bmp", RGB(255, 255, 255));

bossFire.AddBitmap("RES\\boss\\bossfire4.bmp", RGB(255, 255, 255));

bossFire.AddBitmap("RES\\boss\\bossfire5.bmp", RGB(255, 255, 255));

bossFire.SetDelayCount(4);

}

void Boss::LoadBossFireLeftBitMap()

{

bossFireLeft.AddBitmap("RES\\boss\\bossfireleft.bmp", RGB(255, 255, 255));

bossFireLeft.AddBitmap("RES\\boss\\bossfire2left.bmp", RGB(255, 255, 255));

bossFireLeft.AddBitmap("RES\\boss\\bossfire3left.bmp", RGB(255, 255, 255));

bossFireLeft.AddBitmap("RES\\boss\\bossfire4left.bmp", RGB(255, 255, 255));

bossFireLeft.AddBitmap("RES\\boss\\bossfire5left.bmp", RGB(255, 255, 255));

bossFireLeft.SetDelayCount(4);

}

void Boss::LoadBossChopBitMap()

{

bossChop.AddBitmap("RES\\boss\\bossflysword.bmp", RGB(255, 255, 255));

bossChop.AddBitmap("RES\\boss\\bossflysword2.bmp", RGB(255, 255, 255));

bossChop.AddBitmap("RES\\boss\\bossflysword3.bmp", RGB(255, 255, 255));

bossChop.AddBitmap("RES\\boss\\bossflysword4.bmp", RGB(255, 255, 255));

bossChop.AddBitmap("RES\\boss\\bossflysword5.bmp", RGB(255, 255, 255));

bossChop.AddBitmap("RES\\boss\\bossflysword6.bmp", RGB(255, 255, 255));

bossChop.AddBitmap("RES\\boss\\bossflysword7.bmp", RGB(255, 255, 255));

bossChop.AddBitmap("RES\\boss\\bossflysword8.bmp", RGB(255, 255, 255));

bossChop.SetDelayCount(2);

}

void Boss::LoadBossChopLeftBitMap()

{

bossChopLeft.AddBitmap("RES\\boss\\bossflyswordleft.bmp", RGB(255, 255, 255));

bossChopLeft.AddBitmap("RES\\boss\\bossflysword2left.bmp", RGB(255, 255, 255));

bossChopLeft.AddBitmap("RES\\boss\\bossflysword3left.bmp", RGB(255, 255, 255));

bossChopLeft.AddBitmap("RES\\boss\\bossflysword4left.bmp", RGB(255, 255, 255));

bossChopLeft.AddBitmap("RES\\boss\\bossflysword5left.bmp", RGB(255, 255, 255));

bossChopLeft.AddBitmap("RES\\boss\\bossflysword6left.bmp", RGB(255, 255, 255));

bossChopLeft.AddBitmap("RES\\boss\\bossflysword7left.bmp", RGB(255, 255, 255));

bossChopLeft.AddBitmap("RES\\boss\\bossflysword8left.bmp", RGB(255, 255, 255));

bossChopLeft.SetDelayCount(2);

}

void Boss::LoadBossKillBitMap()

{

bossKill.AddBitmap("RES\\boss\\bossKill.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossKill.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossKill.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossKill.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossKill.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword2.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword3.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword4.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword5.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword6.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword7.bmp", RGB(255, 255, 255));

bossKill.AddBitmap("RES\\boss\\bossflysword8.bmp", RGB(255, 255, 255));

bossKill.SetDelayCount(2);

}

void Boss::LoadBossKillLeftBitMap()

{

bossKillLeft.AddBitmap("RES\\boss\\bossKillleft.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossKillleft.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossKillleft.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossKillleft.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossKillleft.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflyswordleft.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflysword2left.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflysword3left.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflysword4left.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflysword5left.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflysword6left.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflysword7left.bmp", RGB(255, 255, 255));

bossKillLeft.AddBitmap("RES\\boss\\bossflysword8left.bmp", RGB(255, 255, 255));

bossKillLeft.SetDelayCount(2);

}

void Boss::LoadKnifeBitMap()

{

knife.AddBitmap("RES\\boss\\flysword\_sword.bmp", RGB(255, 255, 255));

knife.AddBitmap("RES\\boss\\flysword\_sword2.bmp", RGB(255, 255, 255));

knife.AddBitmap("RES\\boss\\flysword\_sword3.bmp", RGB(255, 255, 255));

knife.AddBitmap("RES\\boss\\flysword\_sword4.bmp", RGB(255, 255, 255));

knife.AddBitmap("RES\\boss\\flysword\_sword5.bmp", RGB(255, 255, 255));

knife.SetDelayCount(2);

}

void Boss::LoadKnifeLeftBitMap()

{

knifeLeft.AddBitmap("RES\\boss\\flysword\_sword1left.bmp", RGB(255, 255, 255));

knifeLeft.AddBitmap("RES\\boss\\flysword\_sword2left.bmp", RGB(255, 255, 255));

knifeLeft.AddBitmap("RES\\boss\\flysword\_sword3left.bmp", RGB(255, 255, 255));

knifeLeft.AddBitmap("RES\\boss\\flysword\_sword4left.bmp", RGB(255, 255, 255));

knifeLeft.AddBitmap("RES\\boss\\flysword\_sword5left.bmp", RGB(255, 255, 255));

knifeLeft.SetDelayCount(2);

}

void Boss::LoadLifeObjectBitmap()

{

lifeItem.LoadBitmap("RES\\life\\life\_item\_boss.bmp", RGB(255, 255, 255));

lifeValue.LoadBitmap("RES\\life\\life\_value.bmp", RGB(255, 255, 255));

}

void Boss::LoadBitMap()

{

boss.LoadBitmap("RES\\boss\\boss.bmp", RGB(255, 255, 255));

bossLeft.LoadBitmap("RES\\boss\\bossleft.bmp", RGB(255, 255, 255));

LoadLifeObjectBitmap();

LoadHugeknifeBitMap();

LoadBossDestoryLightBitMap();

LoadBossSprintBitMap();

LoadBossSprintLeftBitMap();

LoadBossFireBitMap();

LoadBossFireLeftBitMap();

LoadBossChopBitMap();

LoadBossChopLeftBitMap();

LoadBossKillBitMap();

LoadBossKillLeftBitMap();

LoadKnifeBitMap();

LoadKnifeLeftBitMap();

for (int i = 0; i < 6; i++)

cannon[i].LoadBitMap();

}

void Boss::ResetAnimation()

{

if (skill == 5)

{

bossKill.Reset();

bossKillLeft.Reset();

hugeknife.Reset();

knife.Reset();

knifeLeft.Reset();

}

else if (skill == 4)

bossDestoryLight.Reset();

else if (skill == 3)

{

bossChop.Reset();

bossChopLeft.Reset();

knife.Reset();

knifeLeft.Reset();

}

else if (skill == 2)

{

bossFire.Reset();

bossFireLeft.Reset();

}

else if (skill == 1)

{

bossSprint.Reset();

bossSprintLeft.Reset();

}

}

void Boss::OnShow()

{

if (AttackDirection == 0)

hugeknife.SetTopLeft(x + screen\_x - 48, y + screen\_y - 1492 - 523);

else if (AttackDirection == 1)

hugeknife.SetTopLeft(x + screen\_x + 6, y + screen\_y - 1492 - 523);

if (bossChop.GetCurrentBitmapNumber() == 2 || bossChopLeft.GetCurrentBitmapNumber() == 2)

{

knife.SetTopLeft(x + screen\_x - 149, y + screen\_y - 1492 - 66 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 43, y + screen\_y - 1492 - 66 - 45);

}

else if(bossChop.GetCurrentBitmapNumber() == 3 || bossChopLeft.GetCurrentBitmapNumber() == 3)

{

knife.SetTopLeft(x + screen\_x - 192, y + screen\_y - 1492 - 19 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 123, y + screen\_y - 1492 - 19 - 45);

}

else if (bossChop.GetCurrentBitmapNumber() == 4 || bossChopLeft.GetCurrentBitmapNumber() == 4)

{

knife.SetTopLeft(x + screen\_x - 188, y + screen\_y - 1492 + 14 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 122, y + screen\_y - 1492 + 10 - 45);

}

else if (bossChop.GetCurrentBitmapNumber() == 5 || bossChopLeft.GetCurrentBitmapNumber() == 5)

{

knife.SetTopLeft(x + screen\_x - 188, y + screen\_y - 1492 + 46 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 122, y + screen\_y - 1492 + 46 - 45);

}

else if (bossChop.GetCurrentBitmapNumber() == 6 || bossChopLeft.GetCurrentBitmapNumber() == 6)

{

knife.SetTopLeft(x + screen\_x - 80, y + screen\_y - 1492 + 68 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 122, y + screen\_y - 1492 + 68 - 45);

}

if (bossKill.GetCurrentBitmapNumber() == 7 || bossKillLeft.GetCurrentBitmapNumber() == 7)

{

knife.SetTopLeft(x + screen\_x - 149, y + screen\_y - 1492 - 66 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 43, y + screen\_y - 1492 - 66 - 45);

}

else if (bossKill.GetCurrentBitmapNumber() == 8 || bossKillLeft.GetCurrentBitmapNumber() == 8)

{

knife.SetTopLeft(x + screen\_x - 192, y + screen\_y - 1492 - 19 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 123, y + screen\_y - 1492 - 19 - 45);

}

else if (bossKill.GetCurrentBitmapNumber() == 9 || bossKillLeft.GetCurrentBitmapNumber() == 9)

{

knife.SetTopLeft(x + screen\_x - 188, y + screen\_y - 1492 + 14 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 122, y + screen\_y - 1492 + 10 - 45);

}

else if (bossKill.GetCurrentBitmapNumber() == 10 || bossKillLeft.GetCurrentBitmapNumber() == 10)

{

knife.SetTopLeft(x + screen\_x - 188, y + screen\_y - 1492 + 46 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 122, y + screen\_y - 1492 + 46 - 45);

}

else if (bossKill.GetCurrentBitmapNumber() == 11 || bossKillLeft.GetCurrentBitmapNumber() == 11)

{

knife.SetTopLeft(x + screen\_x - 80, y + screen\_y - 1492 + 68 - 45);

knifeLeft.SetTopLeft(x + screen\_x + 122, y + screen\_y - 1492 + 68 - 45);

}

bossKill.SetTopLeft(x + screen\_x, y + screen\_y - 1492-30);

bossKillLeft.SetTopLeft(x + screen\_x, y + screen\_y - 1492-30);

bossDestoryLight.SetTopLeft(x + screen\_x, y + screen\_y - 1492 - 85);

bossChop.SetTopLeft(x + screen\_x, y + screen\_y - 1492 - 45);

bossChopLeft.SetTopLeft(x + screen\_x, y + screen\_y - 1492 - 45);

bossFire.SetTopLeft(x + screen\_x, y + screen\_y - 1492);

bossFireLeft.SetTopLeft(x + screen\_x, y + screen\_y - 1492);

bossSprint.SetTopLeft(x + screen\_x, y + screen\_y - 1492);

bossSprintLeft.SetTopLeft(x + screen\_x, y + screen\_y - 1492);

boss.SetTopLeft(x + screen\_x, y + screen\_y - 1492);

bossLeft.SetTopLeft(x + screen\_x, y + screen\_y - 1492);

if (injureDelay == 30)

injureShine = 3;

if (injureDelay < 30)

injureShine++;

if (injureShine % 2 == 0)

injureShine += 0;

else if (skill == 5 && KillDirection == 0 && life <= 24)

{

bossKill.OnShow();

if (bossKill.GetCurrentBitmapNumber() < 5)

hugeknife.OnShow();

if (!knife.IsFinalBitmap() && bossKill.GetCurrentBitmapNumber() > 6)

knife.OnShow();

}

else if (skill == 5 && KillDirection == 1 && life <= 24)

{

bossKillLeft.OnShow();

if (bossKillLeft.GetCurrentBitmapNumber() < 5)

hugeknife.OnShow();

if (!knifeLeft.IsFinalBitmap() && bossKillLeft.GetCurrentBitmapNumber() > 6)

knifeLeft.OnShow();

}

else if (skill == 4)

bossDestoryLight.OnShow();

else if (skill == 3 && AttackDirection == 0)

{

bossChop.OnShow();

if (!knife.IsFinalBitmap() && bossChop.GetCurrentBitmapNumber() > 1)

knife.OnShow();

}

else if (skill == 3 && AttackDirection == 1)

{

bossChopLeft.OnShow();

if (!knifeLeft.IsFinalBitmap() && bossChopLeft.GetCurrentBitmapNumber() > 1)

knifeLeft.OnShow();

}

else if (skill == 2 && AttackDirection == 0)

bossFire.OnShow();

else if (skill == 2 && AttackDirection == 1)

bossFireLeft.OnShow();

else if (skill == 1 && AttackDirection == 0)

bossSprint.OnShow();

else if (skill == 1 && AttackDirection == 1)

bossSprintLeft.OnShow();

else if (AttackDirection == 0)

boss.ShowBitmap();

else if (AttackDirection == 1)

bossLeft.ShowBitmap();

for (int i = 0; i < 6; i++)

if (cannon[i].GetUsingState())

cannon[i].OnShow();

//--------顯示生命值-------------

lifeItem.SetTopLeft(1700, 143);

if(startAttack)

{

lifeItem.ShowBitmap();

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

{

lifeValue.SetTopLeft(1736, 339 - i \* 4);

lifeValue.ShowBitmap();

}

}

//--------顯示生命值-------------

if (injureShine > 4)

injureShine = 0;

}

int Boss::getX()

{

return x;

}

int Boss::getY()

{

return y;

}

int Boss::getScreenX()

{

return screen\_x;

}

int Boss::getScreenY()

{

return screen\_y;

}

int Boss::getLife()

{

return life;

}

bool Boss::getAlive()

{

return isAlive;

}

void Boss::setXY(int nx, int ny)

{

x = nx;

y = ny;

}

void Boss::setScreen\_XY(int nx, int ny)

{

screen\_x = nx;

screen\_y = ny;

}

void Boss::deductLife(int damage)

{

if (damage > 0)

isInjured = true;

if (damage > 4)

damage = 4;

if (injureDelay == 30 && isInjured == true)

life -= damage;

if (life <= 0)

isAlive = false;

}

void Boss::OnMove(int RockX, int RockY)

{

if (life > 24 && skill == 5)

{

isfire = false;

commandProcess++;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

if (skill == 0)

Stay();

else if (skill == 1)

Sprint();

else if (skill == 2)

Fire();

else if (skill == 3)

FireFlySword(RockX, RockY);

else if (skill == 4)

{

if (bossDestoryLight.GetCurrentBitmapNumber() < 4)

destroyLightCount = 3;

if(bossDestoryLight.GetCurrentBitmapNumber() >= 4)

destroyLightCount++;

if (destroyLightCount % 4 == 0 && destroyLightCount <= 16)

CAudio::Instance()->Play(21, false);

DestroyLight();

}

else if (skill == 5)

Kill();

for (int i = 0; i < 6; i++)

cannon[i].OnMove();

if (isInjured&&injureDelay == 30)

{

injureDelay = 0;

isInjured = false;

}

if (injureDelay < 30)

{

injureDelay++;

isInjured = false;

}

}

void Boss::FixCannonScreenXY(int fixX, int fixY)

{

for (int i = 0; i < 6; i++)

{

cannon[i].AddScreenX\_fix(-fixX);

cannon[i].AddScreenY\_fix(-fixY);

}

}

void Boss::Stay()

{

if (attackDelay < 10 && startAttack)

attackDelay++;

if (attackDelay >= 10)

{

isPlaySound = true;

attackDelay = 0;

commandProcess++;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

}

void Boss::Sprint()

{

if (isPlaySound)

{

CAudio::Instance()->Play(17, false);

CAudio::Instance()->Play(18, false);

isPlaySound = false;

}

if (AttackDirection == 0)

{

x -= 24;

screen\_x -= 24;

bossSprint.OnMove();

}

else if (AttackDirection == 1)

{

x += 24;

screen\_x += 24;

bossSprintLeft.OnMove();

}

if (bossSprint.IsFinalBitmap() || bossSprintLeft.IsFinalBitmap())

{

isPlaySound = true;

commandProcess++;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

}

void Boss::Fire()

{

if (bossFire.GetCurrentBitmapNumber() == 0 || bossFireLeft.GetCurrentBitmapNumber() == 0)

{

if(isPlaySound)

{

CAudio::Instance()->Play(19, false);

isPlaySound = false;

}

for (int i = 0; i < 6; i++)

{

if (isfire)

break;

if(!cannon[i].GetUsingState())

{

cannon[i].setCannon(2);

cannon[i].SetUsingState(true);

if (AttackDirection == 1)

{

cannon[i].SetX(x + 159);

cannon[i].SetY(y - 8);

cannon[i].SetScreenXY(x + screen\_x + 159, y + screen\_y - 8 - 1492);

}

else if (AttackDirection == 0)

{

cannon[i].SetX(x - 170);

cannon[i].SetY(y - 8);

cannon[i].SetScreenXY(x + screen\_x - 170, y + screen\_y - 8 - 1492);

}

cannon[i].SetLastMovingState(AttackDirection);

cannon[i].SetCatchAction(1);

isfire = true;

break;

}

}

}

bossFire.OnMove();

bossFireLeft.OnMove();

if (bossFire.IsFinalBitmap() || bossFireLeft.IsFinalBitmap())

{

isPlaySound = true;

isfire = false;

commandProcess++;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

}

void Boss::FireFlySword(int RockX, int RockY)

{

if (bossChop.GetCurrentBitmapNumber() == 2 || bossChopLeft.GetCurrentBitmapNumber() == 2)

{

if (isPlaySound)

{

CAudio::Instance()->Play(20, false);

isPlaySound = false;

}

for (int i = 0; i < 6; i++)

{

if (isfire)

break;

if (!cannon[i].GetUsingState())

{

cannon[i].setCannon(4);

cannon[i].SetUsingState(true);

if (AttackDirection == 1)

{

cannon[i].SetX(x + 159);

cannon[i].SetY(y - 8);

cannon[i].SetScreenXY(x + screen\_x + 159, y + screen\_y - 8 - 1492);

}

else if (AttackDirection == 0)

{

cannon[i].SetX(x - 170);

cannon[i].SetY(y - 8);

cannon[i].SetScreenXY(x + screen\_x - 170, y + screen\_y - 8 - 1492);

}

cannon[i].SetVelocity(RockX , RockY);

cannon[i].SetLastMovingState(AttackDirection);

cannon[i].SetCatchAction(1);

isfire = true;

break;

}

}

}

bossChop.OnMove();

bossChopLeft.OnMove();

if (bossChop.GetCurrentBitmapNumber() > 1 && bossChopLeft.GetCurrentBitmapNumber() > 1)

{

if (!knife.IsFinalBitmap() && !knifeLeft.IsFinalBitmap())

{

knife.OnMove();

knifeLeft.OnMove();

}

}

if (bossChop.IsFinalBitmap() || bossChopLeft.IsFinalBitmap())

{

isPlaySound = true;

isfire = false;

commandProcess++;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

}

void Boss::DestroyLight()

{

if (isPlaySound)

{

CAudio::Instance()->Play(22, false);

isPlaySound = false;

}

if (bossDestoryLight.GetCurrentBitmapNumber() == 4)

{

for (int i = 0; i < 6; i++)

{

if (i == 5)

{

CAudio::Instance()->Play(21, false);

isPlaySound = false;

}

if (isfire)

break;

if (!cannon[i].GetUsingState())

{

cannon[i].setCannon(3);

cannon[i].SetUsingState(true);

cannon[i].SetX(23310 + 300 \* i);

cannon[i].SetY(1730);

cannon[i].SetScreenXY(23310 + 300 \* i + screen\_x, 1730 + screen\_y - 1492);

cannon[i].SetLastMovingState(AttackDirection);

cannon[i].SetCatchAction(1);

}

}

isfire = true;

}

bossDestoryLight.OnMove();

if (bossDestoryLight.IsFinalBitmap())

{

destroyLightCount = 0;

isPlaySound = true;

isfire = false;

commandProcess++;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

}

void Boss::Kill()

{

if (bossKill.GetCurrentBitmapNumber() == 0 && bossKillLeft.GetCurrentBitmapNumber() == 0)

KillDirection = AttackDirection;

if (isPlaySound)

{

CAudio::Instance()->Play(24, false);

isPlaySound = false;

}

if (bossKill.GetCurrentBitmapNumber() == 6 && bossKillLeft.GetCurrentBitmapNumber() == 6)

{

for (int i = 0; i < 6; i++)

{

if (i==0)

CAudio::Instance()->Play(23, false);

if (isfire)

break;

if (!cannon[i].GetUsingState())

{

cannon[i].setCannon(1);

cannon[i].SetUsingState(true);

if (KillDirection == 1)

{

cannon[i].SetX(x + 159);

cannon[i].SetY(y - 352);

cannon[i].SetScreenXY(x + screen\_x + 159, y + screen\_y - 352 - 1492);

}

else if (KillDirection == 0)

{

cannon[i].SetX(x - 232);

cannon[i].SetY(y - 352);

cannon[i].SetScreenXY(x + screen\_x - 232, y + screen\_y - 352 - 1492);

}

cannon[i].SetLastMovingState(KillDirection);

cannon[i].SetCatchAction(1);

isfire = true;

break;

}

}

}

if (bossKill.GetCurrentBitmapNumber() < 5 && bossKillLeft.GetCurrentBitmapNumber() < 5)

hugeknife.OnMove();

if (bossKill.GetCurrentBitmapNumber() > 6 && bossKillLeft.GetCurrentBitmapNumber() > 6)

{

if (!knife.IsFinalBitmap() && !knifeLeft.IsFinalBitmap())

{

knife.OnMove();

knifeLeft.OnMove();

}

}

bossKill.OnMove();

bossKillLeft.OnMove();

if (bossKill.IsFinalBitmap()|| bossKillLeft.IsFinalBitmap())

{

isPlaySound = true;

isfire = false;

commandProcess++;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

}

void Boss::DeterminAttack(int RockX, int RockY)

{

if (commandProcess >= 15)

{

isfire = false;

if (reverse == 0)

reverse = 1;

else if (reverse == 1)

reverse = 0;

commandProcess = 0;

skill = AttackCommand[commandProcess];

ResetAnimation();

}

if (reverse == 0)

AttackDirection = 0;

else if (reverse == 1)

AttackDirection = 1;

if (x - RockX > 0 && skill != 1)

AttackDirection = 0;

else if (RockX - x > 0 && skill != 1)

AttackDirection = 1;

if ((RockX < 25120) && (RockX > 22880) && (RockY < 2000) && (RockY > 1480) && skill == 0)

{

skill = AttackCommand[commandProcess];

startAttack = true;

isPlaySound = true;

}

OnMove(RockX, RockY);

}

bool Boss::MonsterCollision(int RockX, int RockY)

{

if (!isAlive)

return false;

else if (x + 156 > RockX && RockX + 160 > x && y + 171 > RockY && RockY + 200 > y)

return true;

else

return false;

}

int Boss::MonsterCannonCollision(int RockX, int RockY)

{

if (isAlive)

{

int tmp = 0;

for (int i = 0; i < 6; i++)

{

tmp = cannon[i].collision(RockX, RockY);

if (tmp != 0)

return tmp;

}

}

return false;

}

bool Boss::GetStartAttack()

{

return startAttack;

}

}

< Rockman.h >

namespace game\_framework

{

class Rockman

{

public:

Rockman();

void Initialize(); //初始化參數

void LoadBitmap(); //載入角色動畫

void LoadAttackBitmap(); //載入角色攻擊動畫

void OnMove(); //移動圖形

void OnShow(); //顯示圖形

void SetMovingLeft(bool); //設定是否往左

void SetMovingRight(bool); //設定是否往右

void SetJumping(bool); //設定是否正在跳躍

int GetJumpDegree(); //取得跳躍高度

void JumpDropping(); //跳躍後落下

bool GetJumping(); //回傳跳躍狀態

void Dropping(); //是否正在降落

bool GetDropping(); //回傳落下狀態

void isAlreadyOnGround(); //是否正在地面

void SetSprinting(bool); //設定是否正在衝刺

void SetKeySprintState(bool); //設定衝刺按鍵是否按住

int GetSprintDegree(); //取得衝刺長度

void setkickWall(); //判斷是否正在踢牆

void KickWall(); //踢牆

void SetAttacking(bool); //設定是否正在攻擊

void SetKeyAttackingState(bool);//設定攻擊按鍵是否按住

int GetDetermineCharge(); //取得判斷聚氣

int GetCharge(); //取得聚氣程度

int GetChargeAttack(); //取得衝能攻擊

bool GetIsAttacking(); //取得是否正在攻擊

int GetX(); //取得X;

int GetY(); //取得Y;

void SetX(int); //設定X

void SetY(int); //設定Y

void SetCrashState(int); //設定碰撞狀態

void SetCrashStateWall(int); //設定碰撞牆壁狀態

int getInjuredDelay(); //回傳受傷狀態

void SetInjuredState(bool, int);//設定受傷狀態

void SetLife(int); //設定生命值(作弊)

int Getlife(); //取得生命值

void setCannon(int x, int y, int lastMovingState); //設置子彈

RockCannon\* getCannon(); //return rockcannon[20]

void SetFixCannonScreen(int,int); //修正洛克炮螢幕座標XY

private:

bool isMovingLeft; // 是否正在往左移動

bool isMovingRight; // 是否正在往右移動

bool isJumping; // 是否正在跳躍

bool isDropping; // 是否正在下降

bool lastMovingState; // 上個移動狀態

bool isSprinting; // 是否正在衝刺

bool keepkeydownSprinting; // 一直按住衝刺

bool isAttacking; // 是否正在攻擊

bool keepkeydownAttacking; // 一直按住攻擊

bool isInjured; // 是否受傷

bool showDropping(); // 判斷是否要顯示落下動作

CMovingBitmap lifeItem; // 圖片:生命條

CMovingBitmap lifeValue; // 圖片:生命值1

CAnimation animationRight; //動畫:向右站立

CAnimation animationLeft; //動畫:往左站立

CAnimation animationMovingRight; //動畫:往右移動

CAnimation animationMovingLeft; //動畫:往左移動

CAnimation animation\_jumpRight; //動畫:往右跳躍

CAnimation animation\_jumpLeft; //動畫:往左跳躍

CAnimation animation\_sprintRight; //動畫:往右衝刺

CAnimation animation\_sprintLeft; //動畫:往左衝刺

CAnimation animation\_kickWallLeft; //動畫:往左踢牆

CAnimation animation\_kickWallRight; //動畫:往右踢牆

CAnimation animation\_kickWallSlideLeft; //動畫:往左踢牆滑落

CAnimation animation\_kickWallSlideRight;//動畫:往右踢牆滑落

CAnimation animation\_InjureLeft; //動畫:往左受傷

CAnimation animation\_InjureRight; //動畫:往右受傷

void LoadLifeObjectBitmap(); //載入圖片:生命值、生命條

void LoadRightBitmap(); //載入動畫:向右站立

void LoadLeftBitmap(); //載入動畫:向左站立

void LoadMovingLeftBitmap(); //載入動畫:向左移動

void LoadMovingRightBitmap(); //載入動畫:向右移動

void LoadJumpingLeftBitmap(); //載入動畫:向左跳躍

void LoadJumpingRightBitmap(); //載入動畫:向右跳躍

void LoadSprintingLeftBitmap(); //載入動畫:向左衝刺

void LoadSprintingRightBitmap(); //載入動畫:向右衝刺

void LoadKickWallLeftBitmap(); //載入動畫:向左踢牆

void LoadKickWallRightBitmap(); //載入動畫:向右踢牆

void LoadKickWallSlideLeftBitmap(); //載入動畫:向左踢牆滑落

void LoadKickWallSlideRightBitmap(); //載入動畫:向右踢牆滑落

void LoadInjureLeftBitmap(); //載入動畫:往左受傷

void LoadInjureRightBitmap(); //載入動畫:往右受傷

//------以下為攻擊動畫-----------------------------------------------------------

CAnimation animationRightAttack; //動畫:向右站立攻擊

CAnimation animationLeftAttack; //動畫:往左站立攻擊

CAnimation animationMovingRightAttack; //動畫:往右移動攻擊

CAnimation animationMovingLeftAttack; //動畫:往左移動攻擊

CAnimation animation\_jumpRightAttack; //動畫:往右跳躍攻擊

CAnimation animation\_jumpLeftAttack; //動畫:往左跳躍攻擊

CAnimation animation\_sprintLeftAttack; //動畫:往右衝刺攻擊

CAnimation animation\_sprintRightAttack; //動畫:往左衝刺攻擊

CAnimation animation\_kickWallLeftAttack; //動畫:往左踢牆攻擊

CAnimation animation\_kickWallRightAttack; //動畫:往右踢牆攻擊

CAnimation animation\_kickWallSlideLeftAttack; //動畫:往左踢牆滑落攻擊

CAnimation animation\_kickWallSlideRightAttack; //動畫:往右踢牆滑落攻擊

void LoadRightAttackBitmap(); //載入動畫:向右站立攻擊

void LoadLeftAttackBitmap(); //載入動畫:向左站立攻擊

void LoadMovingLeftAttackBitmap(); //載入動畫:向左移動攻擊

void LoadMovingRightAttackBitmap(); //載入動畫:向右移動攻擊

void LoadJumpingLeftAttackBitmap(); //載入動畫:向左跳躍攻擊

void LoadJumpingRightAttackBitmap(); //載入動畫:向右跳躍攻擊

void LoadSprintingLeftAttackBitmap(); //載入動畫:向左衝刺攻擊

void LoadSprintingRightAttackBitmap(); //載入動畫:向右衝刺攻擊

void LoadKickWallLeftAttackBitmap(); //載入動畫:向左踢牆攻擊

void LoadKickWallRightAttackBitmap(); //載入動畫:向右踢牆攻擊

void LoadKickWallSlideLeftAttackBitmap(); //載入動畫:向左踢牆滑落攻擊

void LoadKickWallSlideRightAttackBitmap(); //載入動畫:向右踢牆滑落攻擊

//-------------------------------------------------------------------------------

int x, y; //腳色當前座標

int jumpDegree, sprintDegree; //跳躍高度、衝刺長度

int velocity\_jump, initial\_velocity\_jump; //當前跳躍速度、初始跳躍速度

int velocity\_sprint, initial\_velocity\_sprint; //當前衝刺速度、初始衝刺速度

int dropDegree, landLevel; //降落位置、降落基準點

int velocity\_drop, initial\_velocity\_drop; //當前降落速度、初始降落速度

int kickWallDegree; //踢牆程度

bool isKickWall; //是否正在踢牆

bool isKickWallSlide; //是否正在踢牆滑落

int charge, chargeAttack; //聚氣程度、聚氣攻擊(三段式)

int determineCharge; //判斷是否正在聚氣

int attackActionDelay; //攻擊動作延遲

RockCannon rockcannon[20]; //物件:洛克炮

int crashState; //碰撞狀態

int crashState\_wall; //是否碰撞牆壁

int life; //洛克人生命值

int injureDelay; //受傷無敵時間

int injureShine; //受傷無敵閃爍變數

};

}

< Rockman.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Rockcannon.h"

#include "Rockman.h"

namespace game\_framework

{

Rockman::Rockman()

{

x = y = 0;

}

void Rockman::Initialize()

{

const int X\_POS = 320;

const int Y\_POS = 1880;

x = X\_POS;

y = Y\_POS;

isMovingLeft = isMovingRight = false;

lastMovingState = 0;

jumpDegree = 0;

isJumping = isSprinting = false;

initial\_velocity\_jump = 20;

velocity\_jump = initial\_velocity\_jump;

sprintDegree = 0;

keepkeydownSprinting = false;

initial\_velocity\_sprint = 14;

velocity\_sprint = initial\_velocity\_sprint;

isAttacking = false;

keepkeydownAttacking = false;

chargeAttack = 0;

charge = 0;

determineCharge = 0;

attackActionDelay = 0;

isInjured = false;

initial\_velocity\_drop = 30;

velocity\_drop = initial\_velocity\_drop;

crashState = 1;

crashState\_wall = 0;

kickWallDegree = 0;

isKickWall = false;

isKickWallSlide = false;

life = 64;

injureDelay = 30;

injureShine = 3;

for (int i = 0; i < 20; i++)

rockcannon[i].SetUsingState(false);

}

void Rockman::JumpDropping()

{

if (jumpDegree < 5)

velocity\_jump -= 10;

else if (jumpDegree >= 5)

velocity\_jump += 10;

y += velocity\_jump;

jumpDegree--;

}

void Rockman::KickWall()

{

if (lastMovingState == 1) //踢牆:左

{

x += 20;

if (keepkeydownSprinting)

x += 40;

kickWallDegree++;

}

if (lastMovingState == 0)//踢牆:右

{

x -= 20;

if (keepkeydownSprinting)

x -= 40;

kickWallDegree++;

}

}

void Rockman::setkickWall()//踢牆

{

if (kickWallDegree == 4)

{

kickWallDegree = 0;

isKickWall = false;

}

if (crashState\_wall > 0 && isJumping)

{

isKickWall = true;

}

}

bool Rockman::showDropping()

{

if (((!isJumping) && (jumpDegree >= 3)) || isDropping)

return true;

return false;

}

void Rockman::OnMove()

{

const int STEP\_SIZE = 20;

animationRight.OnMove();

animationLeft.OnMove();

animationMovingRight.OnMove();

animationMovingLeft.OnMove();

animation\_jumpRight.OnMove();

animation\_jumpLeft.OnMove();

animation\_sprintLeft.OnMove();

animation\_sprintRight.OnMove();

animation\_kickWallLeft.OnMove();

animation\_kickWallRight.OnMove();

animation\_kickWallSlideLeft.OnMove();

animation\_kickWallSlideRight.OnMove();

animationRightAttack.OnMove();

animationLeftAttack.OnMove();

animationMovingRightAttack.OnMove();

animationMovingLeftAttack.OnMove();

animation\_jumpRightAttack.OnMove();

animation\_jumpLeftAttack.OnMove();

animation\_sprintLeftAttack.OnMove();

animation\_sprintRightAttack.OnMove();

animation\_kickWallLeftAttack.OnMove();

animation\_kickWallRightAttack.OnMove();

animation\_kickWallSlideLeftAttack.OnMove();

animation\_kickWallSlideRightAttack.OnMove();

animation\_InjureLeft.OnMove();

animation\_InjureRight.OnMove();

if ((isInjured) && (lastMovingState == 1) && (injureDelay == 30))//碰撞到怪物後，腳色往後退

{

x += STEP\_SIZE \* 5;

}

if ((isInjured) && (lastMovingState == 0) && (injureDelay == 30))//碰撞到怪物後，腳色往後退

{

x -= STEP\_SIZE \* 5;

}

if ((isSprinting) && (sprintDegree < 20) && (jumpDegree == 0) && (crashState\_wall == 0) && (!isKickWall)) //衝刺

{

if (lastMovingState == 0 && (!isInjured || injureDelay < 30))

x += velocity\_sprint;

else if (lastMovingState == 1 && (!isInjured || injureDelay < 30))

x -= velocity\_sprint;

if (sprintDegree < 8)

velocity\_sprint += 7;

else

velocity\_sprint -= 7;

sprintDegree++;

if (sprintDegree == 20)

isSprinting = false;

}

else

{

isSprinting = false;

}

if ((!isSprinting) && (sprintDegree > 0))//衝刺結束後，重置衝刺速度。

{

sprintDegree = 0;

velocity\_sprint = initial\_velocity\_sprint;

}

if ((isMovingLeft) && (x >= 0) && !isKickWall && (!isInjured || injureDelay < 30))//往左移動

{

if ((keepkeydownSprinting) && (jumpDegree > 0))//彈射跳躍

x -= STEP\_SIZE \* 3;

else if (!isSprinting)//往左移動

x -= STEP\_SIZE;

}

if (isMovingRight && !isKickWall && (!isInjured || injureDelay < 30)) //往右移動

{

if ((keepkeydownSprinting) && (jumpDegree > 0))//彈射跳躍

x += STEP\_SIZE \* 3;

else if (!isSprinting)//往右移動

x += STEP\_SIZE;

}

if ((isJumping) && (jumpDegree < 10))//跳躍

{

if((!isInjured || injureDelay < 30))

y -= velocity\_jump;

if (jumpDegree < 4 )

velocity\_jump += 10;

else if (jumpDegree >= 4)

velocity\_jump -= 10;

jumpDegree++;

if (jumpDegree == 10)

isJumping = false;

}

if ((!isJumping) && (jumpDegree > 0))//跳躍落下、黏牆滑落。

{

if (crashState\_wall != 0)

{

jumpDegree = 0;

velocity\_jump = initial\_velocity\_jump;

isKickWallSlide = true;

}

else

{

JumpDropping();

isKickWallSlide = false;

}

}

Dropping();

isAlreadyOnGround();

if (isKickWall)

KickWall();

setkickWall();

if (keepkeydownAttacking)

{

if (determineCharge <= 50)

determineCharge++;

if (determineCharge > 5)

charge = determineCharge;

}

if (isInjured&&injureDelay == 30)

{

injureDelay = 0;

isInjured = false;

}

if (injureDelay < 30)

{

injureDelay++;

isInjured = false;

}

}

void Rockman::SetMovingLeft(bool flag)

{

isMovingLeft = flag;

if (flag == true)

lastMovingState = 1;

}

void Rockman::SetMovingRight(bool flag)

{

isMovingRight = flag;

if (flag == true)

lastMovingState = 0;

}

int Rockman::GetJumpDegree()

{

return jumpDegree;

}

int Rockman::GetSprintDegree()

{

return sprintDegree;

}

int Rockman::GetDetermineCharge()

{

return determineCharge;

}

int Rockman::GetCharge()

{

return charge;

}

int Rockman::GetChargeAttack()

{

return chargeAttack;

}

bool Rockman::GetIsAttacking()

{

return isAttacking;

}

void Rockman::SetJumping(bool flag)

{

animation\_jumpRight.Reset();

animation\_jumpLeft.Reset();

animation\_jumpRightAttack.Reset();

animation\_jumpLeftAttack.Reset();

isJumping = flag;

}

bool Rockman::GetJumping()

{

return isJumping;

}

//------------------------------------------

void Rockman::SetCrashState(int state)

{

crashState = state;

}

void Rockman::SetCrashStateWall(int state)

{

crashState\_wall = state;

if (crashState\_wall != 0)

{

if (jumpDegree == 1)

{

animation\_kickWallLeft.Reset();

animation\_kickWallRight.Reset();

animation\_kickWallLeftAttack.Reset();

animation\_kickWallRightAttack.Reset();

}

if (!isMovingLeft)

{

animation\_kickWallSlideLeft.Reset();

animation\_kickWallSlideLeftAttack.Reset();

}

if (!isMovingRight)

{

animation\_kickWallSlideRight.Reset();

animation\_kickWallSlideRightAttack.Reset();

}

}

}

void Rockman::Dropping()

{

if (crashState == 0)

isDropping = true;

else if (crashState == 1)

isDropping = false;

}

bool Rockman::GetDropping()

{

if (crashState\_wall != 0)

isDropping = false;

return isDropping;

}

void Rockman::isAlreadyOnGround()

{

if (isDropping)

{

if ((crashState == 1) && (jumpDegree == 0))

{

isDropping = false;

dropDegree = 0;

velocity\_drop = initial\_velocity\_drop;

}

else if ((crashState\_wall != 0) && (jumpDegree == 0)&&(!isInjured || injureDelay < 30))//踢牆滑落

{

y += 10;

}

else if ((crashState == 0) && (jumpDegree == 0)&& (!isInjured || injureDelay < 30))

{

y += velocity\_drop;

dropDegree++;

if (velocity\_drop < 60)

velocity\_drop += 10;

}

}

}

//------------------------------------------

void Rockman::SetSprinting(bool flag)

{

animation\_sprintRight.Reset();

animation\_sprintLeft.Reset();

animation\_sprintRightAttack.Reset();

animation\_sprintLeftAttack.Reset();

isSprinting = flag;

}

void Rockman::SetKeySprintState(bool flag)

{

keepkeydownSprinting = flag;

}

void Rockman::SetAttacking(bool flag)

{

isAttacking = flag;

}

void Rockman::SetKeyAttackingState(bool flag)

{

keepkeydownAttacking = flag;

if ((charge == 0) && (chargeAttack > 0))

chargeAttack = 0;

if (flag == false)

{

chargeAttack = charge;

charge = 0;

determineCharge = 0;

if (!((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

if (lastMovingState == 0)

{

if (chargeAttack > 50)

setCannon(x + 170, y, 0);

else

setCannon(x + 170, y + 84, 0);

}

else if (lastMovingState == 1)

{

if (chargeAttack > 50)

setCannon(x - 222, y, 1);

else

setCannon(x, y + 84, 1);

}

}

}

}

void Rockman::SetFixCannonScreen(int fixY,int fixX)

{

for (int i = 0; i < 20; i++)

rockcannon[i].SetFixScreen(fixX,fixY);

}

void Rockman::setCannon(int x, int y, int lastMovingState)

{

for (int i = 0; i < 20; i++)

{

if (rockcannon[i].GetUsingState() == false)

{

rockcannon[i].SetUsingState(true);

if (isKickWallSlide&&crashState\_wall != 0)

{

if (lastMovingState == 0)

rockcannon[i].SetLastMovingState(1);

else if (lastMovingState == 1)

rockcannon[i].SetLastMovingState(0);

}

else

rockcannon[i].SetLastMovingState(lastMovingState);

rockcannon[i].SetX(x);

rockcannon[i].SetY(y);

rockcannon[i].SetCatchAction(1);

rockcannon[i].SetNowCharge(chargeAttack);

break;

}

}

}

void Rockman::SetInjuredState(bool state,int injureValue)

{

isInjured = state;

if (injureDelay == 30)

{

animation\_InjureLeft.Reset();

animation\_InjureRight.Reset();

}

if (injureDelay == 30 && state == true)

life -= injureValue;

}

int Rockman::getInjuredDelay()

{

return injureDelay;

}

void Rockman::SetLife(int value)

{

life = value;

}

RockCannon\* Rockman::getCannon()

{

return rockcannon;

}

int Rockman::Getlife()

{

return life;

}

//------------------------------------------

int Rockman::GetX()

{

return x;

}

int Rockman::GetY()

{

return y;

}

void Rockman::SetX(int X\_BT)

{

x = X\_BT;

}

void Rockman::SetY(int Y\_BT)

{

y = Y\_BT;

}

//------------------------------------------

void Rockman::LoadLifeObjectBitmap()

{

lifeItem.LoadBitmapA("RES\\life\\life\_item.bmp", RGB(255, 255, 255));

lifeValue.LoadBitmapA("RES\\life\\life\_value.bmp", RGB(255, 255, 255));

}

void Rockman::LoadRightBitmap()

{

animationRight.AddBitmap("RES\\super armor.bmp", RGB(255, 255, 255));

animationRight.AddBitmap("RES\\super armor2.bmp", RGB(255, 255, 255));

animationRight.AddBitmap("RES\\super armor3.bmp", RGB(255, 255, 255));

animationRight.AddBitmap("RES\\super armor4.bmp", RGB(255, 255, 255));

animationRight.AddBitmap("RES\\super armor5.bmp", RGB(255, 255, 255));

animationRight.SetDelayCount(4);

}

void Rockman::LoadLeftBitmap()

{

animationLeft.AddBitmap("RES\\super armor left.bmp", RGB(255, 255, 255));

animationLeft.AddBitmap("RES\\super armor left2.bmp", RGB(255, 255, 255));

animationLeft.AddBitmap("RES\\super armor left3.bmp", RGB(255, 255, 255));

animationLeft.AddBitmap("RES\\super armor left4.bmp", RGB(255, 255, 255));

animationLeft.AddBitmap("RES\\super armor left5.bmp", RGB(255, 255, 255));

animationLeft.SetDelayCount(4);

}

void Rockman::LoadMovingLeftBitmap()

{

animationMovingLeft.AddBitmap("RES\\super armor movingleft.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft2.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft3.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft4.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft5.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft6.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft7.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft8.bmp", RGB(255, 255, 255));

animationMovingLeft.AddBitmap("RES\\super armor movingleft9.bmp", RGB(255, 255, 255));

animationMovingLeft.SetDelayCount(2);

}

void Rockman::LoadMovingRightBitmap()

{

animationMovingRight.AddBitmap("RES\\super armor moving.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving2.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving3.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving4.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving5.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving6.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving7.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving8.bmp", RGB(255, 255, 255));

animationMovingRight.AddBitmap("RES\\super armor moving9.bmp", RGB(255, 255, 255));

animationMovingRight.SetDelayCount(2);

}

void Rockman::LoadJumpingLeftBitmap()

{

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft2.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft3.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft4.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft5.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft6.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft7.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft8.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft9.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft10.bmp", RGB(255, 255, 255));

animation\_jumpLeft.AddBitmap("RES\\super armor jumpleft11.bmp", RGB(255, 255, 255));

animation\_jumpLeft.SetDelayCount(2);

}

void Rockman::LoadJumpingRightBitmap()

{

animation\_jumpRight.AddBitmap("RES\\super armor jump.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump2.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump3.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump4.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump5.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump6.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump7.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump8.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump9.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump10.bmp", RGB(255, 255, 255));

animation\_jumpRight.AddBitmap("RES\\super armor jump11.bmp", RGB(255, 255, 255));

animation\_jumpRight.SetDelayCount(2);

}

void Rockman::LoadSprintingLeftBitmap()

{

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft2.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft3.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft5.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft6.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft7.bmp", RGB(255, 255, 255));

animation\_sprintLeft.AddBitmap("RES\\super armor sprintingleft8.bmp", RGB(255, 255, 255));

animation\_sprintLeft.SetDelayCount(2);

}

void Rockman::LoadSprintingRightBitmap()

{

animation\_sprintRight.AddBitmap("RES\\super armor sprinting.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting2.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting3.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting5.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting6.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting7.bmp", RGB(255, 255, 255));

animation\_sprintRight.AddBitmap("RES\\super armor sprinting8.bmp", RGB(255, 255, 255));

animation\_sprintRight.SetDelayCount(2);

}

void Rockman::LoadKickWallLeftBitmap()

{

animation\_kickWallLeft.AddBitmap("RES\\super armor kickwallleft.bmp", RGB(255, 255, 255));

animation\_kickWallLeft.AddBitmap("RES\\super armor kickwall5left.bmp", RGB(255, 255, 255));

animation\_kickWallLeft.AddBitmap("RES\\super armor kickwall6left.bmp", RGB(255, 255, 255));

animation\_kickWallLeft.AddBitmap("RES\\super armor jumpleft4.bmp", RGB(255, 255, 255));

animation\_kickWallLeft.SetDelayCount(2);

}

void Rockman::LoadKickWallRightBitmap()

{

animation\_kickWallRight.AddBitmap("RES\\super armor kickwall.bmp", RGB(255, 255, 255));

animation\_kickWallRight.AddBitmap("RES\\super armor kickwall5.bmp", RGB(255, 255, 255));

animation\_kickWallRight.AddBitmap("RES\\super armor kickwall6.bmp", RGB(255, 255, 255));

animation\_kickWallRight.AddBitmap("RES\\super armor jump4.bmp", RGB(255, 255, 255));

animation\_kickWallRight.SetDelayCount(2);

}

void Rockman::LoadKickWallSlideLeftBitmap()

{

animation\_kickWallSlideLeft.AddBitmap("RES\\super armor kickwallleft.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeft.AddBitmap("RES\\super armor kickwall2left.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeft.AddBitmap("RES\\super armor kickwall3left.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeft.AddBitmap("RES\\super armor kickwall4left.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeft.SetDelayCount(2);

}

void Rockman::LoadKickWallSlideRightBitmap()

{

animation\_kickWallSlideRight.AddBitmap("RES\\super armor kickwall.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRight.AddBitmap("RES\\super armor kickwall2.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRight.AddBitmap("RES\\super armor kickwall3.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRight.AddBitmap("RES\\super armor kickwall4.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRight.SetDelayCount(2);

}

void Rockman::LoadInjureLeftBitmap()

{

animation\_InjureLeft.AddBitmap("RES\\super armor injuredleft.bmp", RGB(255, 255, 255));

animation\_InjureLeft.AddBitmap("RES\\super armor injuredleft2.bmp", RGB(255, 255, 255));

animation\_InjureLeft.AddBitmap("RES\\super armor injuredleft3.bmp", RGB(255, 255, 255));

animation\_InjureLeft.AddBitmap("RES\\super armor injuredleft4.bmp", RGB(255, 255, 255));

animation\_InjureLeft.SetDelayCount(2);

}

void Rockman::LoadInjureRightBitmap()

{

animation\_InjureRight.AddBitmap("RES\\super armor injured.bmp", RGB(255, 255, 255));

animation\_InjureRight.AddBitmap("RES\\super armor injured2.bmp", RGB(255, 255, 255));

animation\_InjureRight.AddBitmap("RES\\super armor injured3.bmp", RGB(255, 255, 255));

animation\_InjureRight.AddBitmap("RES\\super armor injured4.bmp", RGB(255, 255, 255));

animation\_InjureRight.SetDelayCount(2);

}

void Rockman::LoadBitmap()

{

LoadLifeObjectBitmap();

LoadRightBitmap();

LoadLeftBitmap();

LoadMovingLeftBitmap();

LoadMovingRightBitmap();

LoadJumpingLeftBitmap();

LoadJumpingRightBitmap();

LoadSprintingLeftBitmap();

LoadSprintingRightBitmap();

LoadKickWallLeftBitmap();

LoadKickWallRightBitmap();

LoadKickWallSlideLeftBitmap();

LoadKickWallSlideRightBitmap();

LoadInjureLeftBitmap();

LoadInjureRightBitmap();

}

void Rockman::LoadRightAttackBitmap()

{

animationRightAttack.AddBitmap("RES\\attack\\super armor.bmp", RGB(255, 255, 255));

}

void Rockman::LoadLeftAttackBitmap()

{

animationLeftAttack.AddBitmap("RES\\attack\\super armor left.bmp", RGB(255, 255, 255));

}

void Rockman::LoadMovingLeftAttackBitmap()

{

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft2.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft3.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft4.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft5.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft6.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft7.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft8.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.AddBitmap("RES\\attack\\super armor movingleft9.bmp", RGB(255, 255, 255));

animationMovingLeftAttack.SetDelayCount(2);

}

void Rockman::LoadMovingRightAttackBitmap()

{

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving2.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving3.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving4.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving5.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving6.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving7.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving8.bmp", RGB(255, 255, 255));

animationMovingRightAttack.AddBitmap("RES\\attack\\super armor moving9.bmp", RGB(255, 255, 255));

animationMovingRightAttack.SetDelayCount(2);

}

void Rockman::LoadJumpingLeftAttackBitmap()

{

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft2.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft3.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft4.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft5.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft6.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft7.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft8.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft9.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft10.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft11.bmp", RGB(255, 255, 255));

animation\_jumpLeftAttack.SetDelayCount(2);

}

void Rockman::LoadJumpingRightAttackBitmap()

{

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump2.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump3.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump4.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump5.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump6.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump7.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump8.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump9.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump10.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.AddBitmap("RES\\attack\\super armor jump11.bmp", RGB(255, 255, 255));

animation\_jumpRightAttack.SetDelayCount(2);

}

void Rockman::LoadSprintingLeftAttackBitmap()

{

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft2.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft3.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft4.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft5.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft6.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft7.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.AddBitmap("RES\\attack\\super armor sprintingleft8.bmp", RGB(255, 255, 255));

animation\_sprintLeftAttack.SetDelayCount(2);

}

void Rockman::LoadSprintingRightAttackBitmap()

{

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting2.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting3.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting4.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting5.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting6.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting7.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.AddBitmap("RES\\attack\\super armor sprinting8.bmp", RGB(255, 255, 255));

animation\_sprintRightAttack.SetDelayCount(2);

}

void Rockman::LoadKickWallLeftAttackBitmap()

{

animation\_kickWallLeftAttack.AddBitmap("RES\\attack\\super armor kickwallleft.bmp", RGB(255, 255, 255));

animation\_kickWallLeftAttack.AddBitmap("RES\\attack\\super armor kickwall5left.bmp", RGB(255, 255, 255));

animation\_kickWallLeftAttack.AddBitmap("RES\\attack\\super armor kickwall6left.bmp", RGB(255, 255, 255));

animation\_kickWallLeftAttack.AddBitmap("RES\\attack\\super armor jumpleft4.bmp", RGB(255, 255, 255));

animation\_kickWallLeftAttack.SetDelayCount(2);

}

void Rockman::LoadKickWallRightAttackBitmap()

{

animation\_kickWallRightAttack.AddBitmap("RES\\attack\\super armor kickwall.bmp", RGB(255, 255, 255));

animation\_kickWallRightAttack.AddBitmap("RES\\attack\\super armor kickwall5.bmp", RGB(255, 255, 255));

animation\_kickWallRightAttack.AddBitmap("RES\\attack\\super armor kickwall6.bmp", RGB(255, 255, 255));

animation\_kickWallRightAttack.AddBitmap("RES\\attack\\super armor jump4.bmp", RGB(255, 255, 255));

animation\_kickWallRightAttack.SetDelayCount(2);

}

void Rockman::LoadKickWallSlideLeftAttackBitmap()

{

animation\_kickWallSlideLeftAttack.AddBitmap("RES\\attack\\super armor kickwallleft.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeftAttack.AddBitmap("RES\\attack\\super armor kickwall2left.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeftAttack.AddBitmap("RES\\attack\\super armor kickwall3left.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeftAttack.AddBitmap("RES\\attack\\super armor kickwall4left.bmp", RGB(255, 255, 255));

animation\_kickWallSlideLeftAttack.SetDelayCount(2);

}

void Rockman::LoadKickWallSlideRightAttackBitmap()

{

animation\_kickWallSlideRightAttack.AddBitmap("RES\\attack\\super armor kickwall.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRightAttack.AddBitmap("RES\\attack\\super armor kickwall2.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRightAttack.AddBitmap("RES\\attack\\super armor kickwall3.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRightAttack.AddBitmap("RES\\attack\\super armor kickwall4.bmp", RGB(255, 255, 255));

animation\_kickWallSlideRightAttack.SetDelayCount(2);

}

void Rockman::LoadAttackBitmap()

{

LoadRightAttackBitmap();

LoadLeftAttackBitmap();

LoadMovingLeftAttackBitmap();

LoadMovingRightAttackBitmap();

LoadJumpingLeftAttackBitmap();

LoadJumpingRightAttackBitmap();

LoadSprintingLeftAttackBitmap();

LoadSprintingRightAttackBitmap();

LoadKickWallLeftAttackBitmap();

LoadKickWallRightAttackBitmap();

LoadKickWallSlideLeftAttackBitmap();

LoadKickWallSlideRightAttackBitmap();

for (int i = 0; i < 20; i++)

rockcannon[i].LoadBitmap();

}

void Rockman::OnShow()

{

int tmp = x, tmp2 = y;

if (x >= 900)

x = 900;

if (y <= 2700)

y = 380;

animationRight.SetTopLeft(x, y);

animationLeft.SetTopLeft(x, y);

animationMovingRight.SetTopLeft(x, y);

animationMovingLeft.SetTopLeft(x, y);

animation\_jumpRight.SetTopLeft(x, y);

animation\_jumpLeft.SetTopLeft(x, y);

animation\_sprintLeft.SetTopLeft(x, y);

animation\_sprintRight.SetTopLeft(x, y);

animation\_kickWallLeft.SetTopLeft(x, y);

animation\_kickWallRight.SetTopLeft(x, y);

animation\_kickWallSlideLeft.SetTopLeft(x, y);

animation\_kickWallSlideRight.SetTopLeft(x, y);

animationRightAttack.SetTopLeft(x, y);

animationLeftAttack.SetTopLeft(x, y);

animationMovingRightAttack.SetTopLeft(x, y);

animationMovingLeftAttack.SetTopLeft(x, y);

animation\_jumpRightAttack.SetTopLeft(x, y);

animation\_jumpLeftAttack.SetTopLeft(x, y);

animation\_sprintLeftAttack.SetTopLeft(x, y);

animation\_sprintRightAttack.SetTopLeft(x, y);

animation\_kickWallLeftAttack.SetTopLeft(x, y);

animation\_kickWallRightAttack.SetTopLeft(x, y);

animation\_kickWallSlideLeftAttack.SetTopLeft(x, y);

animation\_kickWallSlideRightAttack.SetTopLeft(x, y);

animation\_InjureLeft.SetTopLeft(x, y);

animation\_InjureRight.SetTopLeft(x, y);

if (injureDelay == 30)

injureShine = 3;

if (injureDelay < 30 && (animation\_InjureRight.IsFinalBitmap() || animation\_InjureLeft.IsFinalBitmap()))

injureShine++;

if (injureShine % 2 == 0)

injureShine += 0;

else if (injureDelay < 30 && lastMovingState == 0 && !animation\_InjureRight.IsFinalBitmap())

animation\_InjureRight.OnShow();

else if (injureDelay < 30 && lastMovingState == 1 && !animation\_InjureLeft.IsFinalBitmap())

animation\_InjureLeft.OnShow();

else if (lastMovingState == 1 && isKickWall)//踢牆:左

{

if (animation\_kickWallLeft.IsFinalBitmap() || animation\_kickWallLeftAttack.IsFinalBitmap())

{

animation\_kickWallLeft.setToSpecifyBitmap(3);

animation\_kickWallLeftAttack.setToSpecifyBitmap(3);

}

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_kickWallSlideLeftAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_kickWallLeft.OnShow();

}

else if (lastMovingState == 0 && isKickWall)//踢牆:右

{

if (animation\_kickWallRight.IsFinalBitmap() || animation\_kickWallRightAttack.IsFinalBitmap())

{

animation\_kickWallRight.setToSpecifyBitmap(3);

animation\_kickWallRightAttack.setToSpecifyBitmap(3);

}

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_kickWallRightAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_kickWallRight.OnShow();

}

else if ((jumpDegree != 0) && (lastMovingState == 0))

{

if ((!isJumping) && (jumpDegree > 3))

{

animation\_jumpRight.setToSpecifyBitmap(7);

animation\_jumpRightAttack.setToSpecifyBitmap(7);

}

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_jumpRightAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_jumpRight.OnShow();

}

else if ((jumpDegree != 0) && (lastMovingState == 1))

{

if ((!isJumping) && (jumpDegree > 3))

{

animation\_jumpLeft.setToSpecifyBitmap(7);

animation\_jumpLeftAttack.setToSpecifyBitmap(7);

}

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_jumpLeftAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_jumpLeft.OnShow();

}

else if (crashState\_wall == 1)//踢牆滑落:左

{

if (animation\_kickWallSlideLeft.IsFinalBitmap() || animation\_kickWallSlideLeftAttack.IsFinalBitmap())

{

animation\_kickWallSlideLeft.setToSpecifyBitmap(3);

animation\_kickWallSlideLeftAttack.setToSpecifyBitmap(3);

}

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_kickWallSlideLeftAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_kickWallSlideLeft.OnShow();

}

else if (crashState\_wall == 2)//踢牆滑落:右

{

if (animation\_kickWallSlideRight.IsFinalBitmap() || animation\_kickWallSlideRightAttack.IsFinalBitmap())

{

animation\_kickWallSlideRight.setToSpecifyBitmap(3);

animation\_kickWallSlideRightAttack.setToSpecifyBitmap(3);

}

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_kickWallSlideRightAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_kickWallSlideRight.OnShow();

}

else if (showDropping() && lastMovingState == 0)

{

animation\_jumpRight.setToSpecifyBitmap(7);

animation\_jumpRightAttack.setToSpecifyBitmap(7);

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_jumpRightAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_jumpRight.OnShow();

}

else if (showDropping() && lastMovingState == 1)

{

animation\_jumpLeft.setToSpecifyBitmap(7);

animation\_jumpLeftAttack.setToSpecifyBitmap(7);

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_jumpLeftAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_jumpLeft.OnShow();

}

else if ((sprintDegree != 0) && (lastMovingState == 0))

{

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_sprintRightAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_sprintRight.OnShow();

}

else if ((sprintDegree != 0) && (lastMovingState == 1))

{

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animation\_sprintLeftAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animation\_sprintLeft.OnShow();

}

else if ((isMovingRight == true) && (isMovingLeft == false))

{

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animationMovingRightAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animationMovingRight.OnShow();

}

else if ((isMovingRight == false) && (isMovingLeft == true))

{

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animationMovingLeftAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animationMovingLeft.OnShow();

}

else if (lastMovingState == 0)

{

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animationRightAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animationRight.OnShow();

}

else if (lastMovingState == 1)

{

if (((isAttacking) && (charge == 0)) || (chargeAttack > 5))

{

animationLeftAttack.OnShow();

if (keepkeydownAttacking == false)

attackActionDelay++;

if (attackActionDelay >= 20)

{

attackActionDelay = 0;

chargeAttack = 0;

}

}

else

animationLeft.OnShow();

}

for (int i = 0; i < 20; i++)

{

if (rockcannon[i].GetUsingState())

rockcannon[i].OnShow();

else

rockcannon[i].OnShowHit();

}

x = tmp;

y = tmp2;

//--------顯示生命值-------------

lifeItem.SetTopLeft(100, 143);

lifeItem.ShowBitmap();

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

{

lifeValue.SetTopLeft(140, 403 - i \* 4);

lifeValue.ShowBitmap();

}

//--------顯示生命值-------------

if (injureShine > 4)

injureShine = 0;

}

}

< Trashcannon.h >

namespace game\_framework

{

class TrashCannon

{

public:

TrashCannon();

void LoadBitmap(); //載入砲彈動畫

void SetX(int); //設定X

void SetY(int); //設定Y

void SetScreenXY(int, int); //設定砲擊螢幕座標

void AddScreenX\_fix(int); //修正砲彈螢幕座標X

void AddScreenY\_fix(int); //修正砲彈螢幕座標Y

void OnMove(); //移動圖形

void OnShow(); //顯示圖形

void OnShowHit(); //顯示砲彈擊中動畫

void SetLastMovingState(int); //設定上次移動狀態

void SetUsingState(bool flag); //設定砲彈使用狀態

bool GetUsingState(); //取得砲彈使用狀態

void SetCatchAction(bool); //設定砲彈抓取使用者座標

int collision(int x, int y); //砲彈與洛克人碰撞

private:

int x1, y1; //腳色當前座標

int screenX, screenY; //畫面的座標

int hitX, hitY; //子彈擊中座標

void SetHitXY(); //設定擊中座標

CMovingBitmap trashCannon; //圖片:垃圾炮

CAnimation trashCannonHit; //圖片:垃圾炮擊中

void LoadTrashCannonHitBitmap(); //載入動畫:垃圾炮擊中

void HitAnimationLock(); //擊中動畫鎖定

int velocity\_cannon; //垃圾炮速度

int distance; //當前移動距離

int lastMovingState; //上次的移動方向

bool usingState; //是否被使用中

bool catchAction; //0:抓取動作，1:不抓取動作

int isHitSomething; //是否擊中到東西

bool isCatchHitXY; //是否擷取擊中座標

bool showLock; //顯示鎖定

bool boomsound; //是否播放爆炸音效

};

}

< Trashcannon.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "TrashCannon.h"

namespace game\_framework

{

TrashCannon::TrashCannon()

{

x1 = 0;

y1 = 700;

screenX = 0;

screenY = 0;

hitX = hitY = 0;

velocity\_cannon = 30;

distance = 0;

lastMovingState = 0;

usingState = false;

catchAction = 0;

showLock = 0;

isHitSomething = 0;

isCatchHitXY = false;

boomsound = false;

}

void TrashCannon::LoadBitmap()

{

LoadTrashCannonHitBitmap();

trashCannon.LoadBitmap("RES\\enemy\\trashcannon.bmp", RGB(255, 255, 255));

}

void TrashCannon::SetLastMovingState(int flag)

{

if (catchAction == 0)

lastMovingState = flag;

}

void TrashCannon::SetX(int x)

{

if (catchAction == 0)

x1 = x;

}

void TrashCannon::SetY(int y)

{

if (catchAction == 0)

y1 = y;

}

void TrashCannon::AddScreenX\_fix(int fix)

{

if (usingState == true)

screenX += fix;

if (usingState == false)

hitX += fix;

}

void TrashCannon::AddScreenY\_fix(int fix)

{

if (usingState == true)

screenY += fix;

if (usingState == false)

hitY += fix;

}

void TrashCannon::SetHitXY()

{

if (!isCatchHitXY)

{

hitX = screenX;

hitY = screenY;

isCatchHitXY = true;

}

}

void TrashCannon::SetCatchAction(bool flag)

{

catchAction = flag;

}

bool TrashCannon::GetUsingState()

{

return usingState;

}

void TrashCannon::SetUsingState(bool flag)

{

usingState = flag;

distance = 0;

catchAction = 0;

x1 = 0;

y1 = 0;

if(usingState)

trashCannonHit.Reset();

}

void TrashCannon::SetScreenXY(int x,int y)

{

if (showLock == 0)

{

screenX = x;

screenY = y;

showLock = 1;

}

}

void TrashCannon::HitAnimationLock()

{

if (trashCannonHit.IsFinalBitmap())

{

isCatchHitXY = false;

isHitSomething = 0;

trashCannonHit.setToSpecifyBitmap(7);

}

}

int TrashCannon::collision(int x, int y)

{

if (!usingState)

return 0;

if (lastMovingState == 0)

{

if ((x1 + 32 >= x) && (x1 <= x + 160) && (y1 <= y + 200) && (y1 + 32 >= y))

{

isHitSomething = 2;

SetHitXY();

return 2;

}

}

else if (lastMovingState == 1)

{

if ((x1 <= x + 160) && (x1 >= x) && (y1 <= y + 200) && (y1 + 32 >= y))

{

isHitSomething = 2;

SetHitXY();

return 2;

}

}

return 0;

}

void TrashCannon::OnMove()

{

if (distance >= 800 || isHitSomething != 0)

{

if (usingState)

boomsound = true;

usingState = false;

catchAction = 0;

x1 = 0;

y1 = 0;

}

if (distance >= 800 && trashCannonHit.GetCurrentBitmapNumber() == 0)

{

hitX = screenX;

hitY = screenY;

}

if (catchAction == 0)

showLock = 0;

if (usingState)

{

if (lastMovingState == 0)

{

x1 -= velocity\_cannon;

screenX -= velocity\_cannon;

distance += velocity\_cannon;

}

else if (lastMovingState == 1)

{

x1 += velocity\_cannon;

screenX += velocity\_cannon;

distance += velocity\_cannon;

}

}

else

trashCannonHit.OnMove();

if(boomsound)

{

CAudio::Instance()->Play(14, false);

boomsound = false;

}

}

void TrashCannon::OnShow()

{

int tmp = x1, tmp2 = y1;

if (x1 >= 900)

x1 = screenX;

if (y1 <= 2700)

y1 = screenY;

trashCannon.SetTopLeft(x1, y1);

if (usingState == true)

trashCannon.ShowBitmap();

x1 = tmp;

y1 = tmp2;

}

void TrashCannon::OnShowHit()

{

trashCannonHit.SetTopLeft(hitX, hitY);

HitAnimationLock();

if (usingState == false && !trashCannonHit.IsFinalBitmap())//

{

if ((isHitSomething > 0 && isHitSomething < 8) || distance >= 800)

{

trashCannonHit.OnShow();

}

}

}

void TrashCannon::LoadTrashCannonHitBitmap()

{

trashCannonHit.AddBitmap("RES\\enemy\\explosion.bmp", RGB(255, 255, 255));

trashCannonHit.AddBitmap("RES\\enemy\\explosion2.bmp", RGB(255, 255, 255));

trashCannonHit.AddBitmap("RES\\enemy\\explosion3.bmp", RGB(255, 255, 255));

trashCannonHit.AddBitmap("RES\\enemy\\explosion4.bmp", RGB(255, 255, 255));

trashCannonHit.AddBitmap("RES\\enemy\\explosion5.bmp", RGB(255, 255, 255));

trashCannonHit.AddBitmap("RES\\enemy\\explosion6.bmp", RGB(255, 255, 255));

trashCannonHit.AddBitmap("RES\\enemy\\explosion7.bmp", RGB(255, 255, 255));

trashCannonHit.AddBitmap("RES\\enemy\\explosion8.bmp", RGB(255, 255, 255));

trashCannonHit.SetDelayCount(2);

}

}

< nightmareCannon.h >

namespace game\_framework

{

class NightmareCannon

{

public:

NightmareCannon();

void LoadBitmap(); //載入角色動畫

void SetX(int); //設定X

void SetY(int); //設定Y

void SetScreenXY(int, int); //設定砲擊螢幕座標

void AddScreenX\_fix(int); //修正砲彈螢幕座標X

void AddScreenY\_fix(int); //修正砲彈螢幕座標Y

void OnMove(); //移動圖形

void OnShow(); //顯示圖形

void OnShowHit(); //顯示砲彈擊中動畫

void SetVelocity(int, int); //設定砲彈速度(追蹤砲彈)

void SetLastMovingState(int); //設定上次移動狀態

void SetUsingState(bool flag); //設定砲彈使用狀態

bool GetUsingState(); //取得砲彈使用狀態

void SetCatchAction(bool); //設定砲彈抓取使用者座標

int collision(int x, int y); //砲彈與洛克人碰撞

private:

int x1, y1; //腳色當前座標

int screenX, screenY; //畫面的座標

int hitX, hitY; //子彈擊中座標

void SetHitXY(); //設定擊中座標

CMovingBitmap nightmareCannon; //圖片:夢魘炮

CAnimation nightmareCannonHit; //圖片:夢魘炮擊中

void LoadNightmareCannonHitBitmap(); //載入動畫:夢魘炮擊中

void HitAnimationLock(); //擊中動畫鎖定

int velocity\_cannon\_x,velocity\_cannon\_y;//夢魘炮速度

int distance; //當前移動距離

int lastMovingState; //上次的移動方向

bool usingState; //是否被使用中

bool catchAction; //0:抓取動作，1:不抓取動作

int isHitSomething; //是否擊中到東西

bool isCatchHitXY; //是否擷取擊中座標

bool showLock; //顯示鎖定

bool boomsound; //是否播放爆炸音效

};

}

< nightmareCannon.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "nightmareCannon.h"

#include <cmath>

namespace game\_framework

{

NightmareCannon::NightmareCannon()

{

x1 = 0;

y1 = 700;

screenX = 0;

screenY = 0;

hitX = hitY = 0;

velocity\_cannon\_x = 60;

velocity\_cannon\_y = 0;

distance = 0;

lastMovingState = 0;

usingState = false;

catchAction = 0;

showLock = 0;

isHitSomething = 0;

isCatchHitXY = false;

boomsound = false;

}

void NightmareCannon::LoadBitmap()

{

LoadNightmareCannonHitBitmap();

nightmareCannon.LoadBitmap("RES\\enemy\\blueCannon.bmp", RGB(255, 255, 255));

}

void NightmareCannon::SetLastMovingState(int flag)

{

if (catchAction == 0)

lastMovingState = flag;

}

void NightmareCannon::SetX(int x)

{

if (catchAction == 0)

x1 = x;

}

void NightmareCannon::SetY(int y)

{

if (catchAction == 0)

y1 = y;

}

void NightmareCannon::AddScreenX\_fix(int fix)

{

if (usingState == true)

screenX += fix;

if (usingState == false)

hitX += fix;

}

void NightmareCannon::AddScreenY\_fix(int fix)

{

if (usingState == true)

screenY += fix;

if (usingState == false)

hitY += fix;

}

void NightmareCannon::SetHitXY()

{

if (!isCatchHitXY)

{

hitX = screenX;

hitY = screenY;

isCatchHitXY = true;

}

}

void NightmareCannon::SetCatchAction(bool flag)

{

catchAction = flag;

}

bool NightmareCannon::GetUsingState()

{

return usingState;

}

void NightmareCannon::SetUsingState(bool flag)

{

usingState = flag;

distance = 0;

catchAction = 0;

x1 = 0;

y1 = 0;

screenX = 0;

screenY = 0;

if (usingState)

nightmareCannonHit.Reset();

}

void NightmareCannon::SetScreenXY(int x, int y)

{

if (showLock == 0)

{

screenX = x;

screenY = y;

showLock = 1;

}

}

void NightmareCannon::SetVelocity(int RockX, int RockY)

{

int v\_x, v\_y;

if (RockX > x1)

v\_x = RockX - x1;

else

v\_x = x1 - RockX;

if (RockY > y1)

v\_y = RockY - y1;

else

v\_y = y1 - RockY;

velocity\_cannon\_x = (int)(v\_x \* 60 / sqrt(pow(v\_x, 2) + pow(v\_y, 2)));

velocity\_cannon\_y = (int)(v\_y \* 60 / sqrt(pow(v\_x, 2) + pow(v\_y, 2)));

if (RockY < y1)

velocity\_cannon\_y = velocity\_cannon\_y \* -1;

}

void NightmareCannon::HitAnimationLock()

{

if (nightmareCannonHit.IsFinalBitmap())

{

isCatchHitXY = false;

isHitSomething = 0;

nightmareCannonHit.setToSpecifyBitmap(7);

}

}

int NightmareCannon::collision(int x, int y)

{

if (!usingState)

return 0;

if (lastMovingState == 0)

{

if ((x1 + 58 >= x) && (x1 <= x + 160) && (y1 <= y + 200) && (y1 + 55 >= y))

{

isHitSomething = 2;

SetHitXY();

return 2;

}

}

else if (lastMovingState == 1)

{

if ((x1 <= x + 160) && (x1 >= x) && (y1 <= y + 200) && (y1 + 55 >= y))

{

isHitSomething = 2;

SetHitXY();

return 2;

}

}

return 0;

}

void NightmareCannon::OnMove()

{

CONST int velocity\_cannon = 60;

if (distance >= 1600 || isHitSomething != 0)

{

if(usingState)

boomsound = true;

usingState = false;

catchAction = 0;

x1 = 0;

y1 = 0;

}

if (distance >= 1600 && nightmareCannonHit.GetCurrentBitmapNumber() == 0)

{

hitX = screenX;

hitY = screenY;

}

if (catchAction == 0)

showLock = 0;

if (usingState)

{

if (lastMovingState == 0)

{

x1 -= velocity\_cannon\_x;

screenX -= velocity\_cannon\_x;

distance += velocity\_cannon;

}

else if (lastMovingState == 1)

{

x1 += velocity\_cannon\_x;

screenX += velocity\_cannon\_x;

distance += velocity\_cannon;

}

y1 += velocity\_cannon\_y;

screenY += velocity\_cannon\_y;

}

else

nightmareCannonHit.OnMove();

if (boomsound)

{

CAudio::Instance()->Play(14, false);

boomsound = false;

}

}

void NightmareCannon::OnShow()

{

int tmp = x1, tmp2 = y1;

if (x1 >= 900)

x1 = screenX;

if (y1 <= 2700)

y1 = screenY;

nightmareCannon.SetTopLeft(x1, y1);

if (usingState == true)

nightmareCannon.ShowBitmap();

x1 = tmp;

y1 = tmp2;

}

void NightmareCannon::OnShowHit()

{

nightmareCannonHit.SetTopLeft(hitX, hitY);

HitAnimationLock();

if (usingState == false && !nightmareCannonHit.IsFinalBitmap())//

{

if ((isHitSomething > 0 && isHitSomething < 8) || distance >= 1600)

{

nightmareCannonHit.OnShow();

}

}

}

void NightmareCannon::LoadNightmareCannonHitBitmap()

{

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion.bmp", RGB(255, 255, 255));

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion2.bmp", RGB(255, 255, 255));

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion3.bmp", RGB(255, 255, 255));

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion4.bmp", RGB(255, 255, 255));

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion5.bmp", RGB(255, 255, 255));

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion6.bmp", RGB(255, 255, 255));

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion7.bmp", RGB(255, 255, 255));

nightmareCannonHit.AddBitmap("RES\\enemy\\explosion8.bmp", RGB(255, 255, 255));

nightmareCannonHit.SetDelayCount(2);

}

}

< BossCannon.h >

namespace game\_framework

{

class BossCannon

{

public:

BossCannon();

void LoadBitMap(); //載入砲彈動畫

void SetX(int x); //設定X

void SetY(int y); //設定Y

void SetScreenXY(int x, int y); //設定砲彈螢幕座標XY

void AddScreenX\_fix(int fix); //修正砲彈螢幕座標X

void AddScreenY\_fix(int fix); //修正砲彈螢幕座標Y

void OnMove(); //移動圖形

void OnShow(); //顯示圖形

void setCannon(int t); //設定砲彈型態

void SetVelocity(int RockX, int RockY); //設定砲彈速度(追蹤砲彈)

void SetLastMovingState(int flag); //設定上次移動狀態

void SetUsingState(bool flag); //設定砲彈使用狀態

bool GetUsingState(); //取得砲彈使用狀態

void SetCatchAction(bool flag); //設定砲彈抓取使用者座標

int collision(int x, int y); //砲彈與洛克人碰撞

private:

int x1, y1; //角色當前座標

int screenX, screenY; //畫面座標

int hitX, hitY; //子彈擊中座標

CAnimation cannon; //動畫:螺旋砲(右)

CAnimation cannonLeft; //動畫:螺旋砲(左)

CAnimation flySword; //動畫:追蹤飛刀(右)

CAnimation flySwordLeft; //動畫:追蹤飛刀(左)

CAnimation destroyLight; //動畫:滅閃光砲彈(槌地板砲彈)

CMovingBitmap hugeSword; //圖片:幻夢零劍氣(右)

CMovingBitmap hugeSwordLeft; //圖片:幻夢零劍氣(左)

void LoadCannonBitMap(); //動畫:螺旋砲(右)

void LoadCannonLeftBitMap(); //動畫:螺旋砲(左)

void LoadFlySwordBitMap(); //動畫:追蹤飛刀(右)

void LoadFlySwordLeftBitMap(); //動畫:追蹤飛刀(左)

void LoadDestroyLightBitMap(); //動畫:滅閃光砲彈(槌地板砲彈)

int type; //1 = 幻夢零劍氣 2 = 螺旋砲 3 = 滅閃光砲彈(槌地板砲彈) 4 = 追蹤飛刀

int velocity\_cannon\_x; //夢魘傑洛炮速度

int velocity\_cannon\_y; //夢魘傑洛炮速度

int distance; //當前移動距離

int lastMovingState; //上次的移動方向

bool usingState; //是否被使用中

bool catchAction; //0:抓取動作，1:不抓取動作

int isHitSomething; //是否擊中到東西

bool isCatchHitXY; //是否擷取擊中座標

bool showLock; //顯示鎖定

};

}

< BossCannon.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include "audio.h"

#include "gamelib.h"

#include "BossCannon.h"

namespace game\_framework

{

BossCannon::BossCannon()

{

x1 = 0;

y1 = 700;

screenX = 0;

screenY = 0;

hitX = hitY = 0;

velocity\_cannon\_x = 30;

velocity\_cannon\_y = 30;

distance = 0;

lastMovingState = 0;

usingState = false;

catchAction = 0;

showLock = 0;

isHitSomething = 0;

isCatchHitXY = false;

}

void BossCannon::LoadFlySwordBitMap()

{

flySword.AddBitmap("RES\\boss\\flysword\_shot.bmp", RGB(255, 255, 255));

flySword.AddBitmap("RES\\boss\\flysword\_shot2.bmp", RGB(255, 255, 255));

flySword.AddBitmap("RES\\boss\\flysword\_shot3.bmp", RGB(255, 255, 255));

flySword.AddBitmap("RES\\boss\\flysword\_shot4.bmp", RGB(255, 255, 255));

flySword.AddBitmap("RES\\boss\\flysword\_shot3.bmp", RGB(255, 255, 255));

flySword.AddBitmap("RES\\boss\\flysword\_shot2.bmp", RGB(255, 255, 255));

flySword.SetDelayCount(1);

}

void BossCannon::LoadFlySwordLeftBitMap()

{

flySwordLeft.AddBitmap("RES\\boss\\flysword\_shotleft.bmp", RGB(255, 255, 255));

flySwordLeft.AddBitmap("RES\\boss\\flysword\_shot2left.bmp", RGB(255, 255, 255));

flySwordLeft.AddBitmap("RES\\boss\\flysword\_shot3left.bmp", RGB(255, 255, 255));

flySwordLeft.AddBitmap("RES\\boss\\flysword\_shot4left.bmp", RGB(255, 255, 255));

flySwordLeft.AddBitmap("RES\\boss\\flysword\_shot3left.bmp", RGB(255, 255, 255));

flySwordLeft.AddBitmap("RES\\boss\\flysword\_shot2left.bmp", RGB(255, 255, 255));

flySwordLeft.SetDelayCount(1);

}

void BossCannon::LoadDestroyLightBitMap()

{

destroyLight.AddBitmap("RES\\boss\\destroylight.bmp", RGB(255, 255, 255));

destroyLight.AddBitmap("RES\\boss\\destroylight2.bmp", RGB(255, 255, 255));

destroyLight.AddBitmap("RES\\boss\\destroylight3.bmp", RGB(255, 255, 255));

destroyLight.AddBitmap("RES\\boss\\destroylight4.bmp", RGB(255, 255, 255));

destroyLight.SetDelayCount(1);

}

void BossCannon::LoadCannonBitMap()

{

cannon.AddBitmap("RES\\boss\\cannon.bmp", RGB(255, 255, 255));

cannon.AddBitmap("RES\\boss\\cannon2.bmp", RGB(255, 255, 255));

cannon.AddBitmap("RES\\boss\\cannon3.bmp", RGB(255, 255, 255));

cannon.SetDelayCount(2);

}

void BossCannon::LoadCannonLeftBitMap()

{

cannonLeft.AddBitmap("RES\\boss\\cannonleft.bmp", RGB(255, 255, 255));

cannonLeft.AddBitmap("RES\\boss\\cannon2left.bmp", RGB(255, 255, 255));

cannonLeft.AddBitmap("RES\\boss\\cannon3left.bmp", RGB(255, 255, 255));

cannonLeft.SetDelayCount(2);

}

void BossCannon::LoadBitMap()

{

hugeSword.LoadBitmap("RES\\boss\\Killfire.bmp", RGB(255, 255, 255));

hugeSwordLeft.LoadBitmap("RES\\boss\\Killfireleft.bmp", RGB(255, 255, 255));

LoadFlySwordBitMap();

LoadFlySwordLeftBitMap();

LoadDestroyLightBitMap();

LoadCannonBitMap();

LoadCannonLeftBitMap();

}

void BossCannon::setCannon(int t)

{

type = t;

showLock = 0;

}

void BossCannon::SetVelocity(int RockX, int RockY)

{

int v\_x, v\_y;

if (RockX > x1)

v\_x = RockX - x1;

else

v\_x = x1 - RockX;

if (RockY > y1)

v\_y = RockY - y1;

else

v\_y = y1 - RockY;

velocity\_cannon\_x = (int)(v\_x \* 60 / sqrt(pow(v\_x, 2) + pow(v\_y, 2)));

velocity\_cannon\_y = (int)(v\_y \* 60 / sqrt(pow(v\_x, 2) + pow(v\_y, 2)));

if (RockY < y1)

velocity\_cannon\_y = velocity\_cannon\_y \* -1;

}

void BossCannon::OnMove()

{

if (distance >= 1600)

{

usingState = false;

catchAction = 0;

x1 = 0;

y1 = 0;

distance = 0;

isHitSomething = 0;

}

if (type == 1 || type == 2)

{

velocity\_cannon\_x = 30;

if (type == 1)

velocity\_cannon\_x = 50;

if (catchAction == 0)

showLock = 0;

if (usingState)

{

if (lastMovingState == 0)

{

x1 -= velocity\_cannon\_x;

screenX -= velocity\_cannon\_x;

distance += velocity\_cannon\_x;

}

else if (lastMovingState == 1)

{

x1 += velocity\_cannon\_x;

screenX += velocity\_cannon\_x;

distance += velocity\_cannon\_x;

}

}

cannon.OnMove();

cannonLeft.OnMove();

}

else if (type == 3)

{

velocity\_cannon\_y = 50;

distance += 50;

if (catchAction == 0)

showLock = 0;

if (usingState)

{

y1 -= velocity\_cannon\_y;

screenY -= velocity\_cannon\_y;

distance += velocity\_cannon\_y;

}

destroyLight.OnMove();

}

else if (type == 4)

{

if (catchAction == 0)

showLock = 0;

if (usingState)

{

CONST int velocity\_cannon = 60;

if (lastMovingState == 0)

{

x1 -= velocity\_cannon\_x;

screenX -= velocity\_cannon\_x;

distance += velocity\_cannon;

}

else if (lastMovingState == 1)

{

x1 += velocity\_cannon\_x;

screenX += velocity\_cannon\_x;

distance += velocity\_cannon;

}

y1 += velocity\_cannon\_y;

screenY += velocity\_cannon\_y;

}

flySword.OnMove();

flySwordLeft.OnMove();

}

}

void BossCannon::OnShow()

{

int tmp = x1, tmp2 = y1;

if (x1 >= 900)

x1 = screenX;

if (y1 <= 2700)

y1 = screenY;

if (type == 1 && lastMovingState == 0)

{

hugeSword.SetTopLeft(x1, y1);

hugeSword.ShowBitmap();

}

else if (type == 1 && lastMovingState == 1)

{

hugeSwordLeft.SetTopLeft(x1, y1);

hugeSwordLeft.ShowBitmap();

}

else if (type == 2 && lastMovingState == 0)

{

cannon.SetTopLeft(x1, y1);

cannon.OnShow();

}

else if (type == 2 && lastMovingState == 1)

{

cannonLeft.SetTopLeft(x1, y1);

cannonLeft.OnShow();

}

else if (type == 3)

{

destroyLight.SetTopLeft(x1, y1);

destroyLight.OnShow();

}

else if (type == 4 && lastMovingState == 0)

{

flySword.SetTopLeft(x1, y1);

flySword.OnShow();

}

else if (type == 4 && lastMovingState == 1)

{

flySwordLeft.SetTopLeft(x1, y1);

flySwordLeft.OnShow();

}

x1 = tmp;

y1 = tmp2;

}

void BossCannon::SetLastMovingState(int flag)

{

if (catchAction == 0)

lastMovingState = flag;

}

void BossCannon::SetX(int x)

{

if (catchAction == 0)

x1 = x;

}

void BossCannon::SetY(int y)

{

if (catchAction == 0)

y1 = y;

}

void BossCannon::AddScreenX\_fix(int fix)

{

if (usingState == true)

screenX += fix;

if (usingState == false)

hitX += fix;

}

void BossCannon::AddScreenY\_fix(int fix)

{

if (usingState == true)

screenY += fix;

if (usingState == false)

hitY += fix;

}

void BossCannon::SetScreenXY(int x, int y)

{

if (showLock == 0)

{

screenX = x;

screenY = y;

showLock = 1;

}

}

void BossCannon::SetCatchAction(bool flag)

{

catchAction = flag;

}

void BossCannon::SetUsingState(bool flag)

{

usingState = flag;

distance = 0;

catchAction = 0;

x1 = 0;

y1 = 0;

screenX = 0;

screenY = 0;

}

bool BossCannon::GetUsingState()

{

return usingState;

}

int BossCannon::collision(int x, int y)

{

if (!usingState)

return 0;

if (type == 1)

{

if (lastMovingState == 0)

{

if ((x1 + 232 >= x) && (x1 <= x + 160) && (y1 <= y + 200) && (y1 + 523 >= y))

{

isHitSomething = 1;

return 16;

}

}

else if (lastMovingState == 1)

{

if ((x1 <= x + 160) && (x1 >= x) && (y1 <= y + 200) && (y1 + 523 >= y))

{

isHitSomething = 1;

return 16;

}

}

}

else if (type == 2)

{

if (lastMovingState == 0)

{

if ((x1 + 32 >= x) && (x1 <= x + 160) && (y1 <= y + 200) && (y1 + 109 >= y))

{

isHitSomething = 4;

return 4;

}

}

else if (lastMovingState == 1)

{

if ((x1 + 58 <= x + 160) && (x1 + 58 + 160 >= x) && (y1 <= y + 200) && (y1 + 109 >= y))

{

isHitSomething = 4;

return 4;

}

}

}

else if (type == 3)

{

if ((x1 + 7 + 94 >= x) && (x1 + 7 <= x + 160) && (y1 <= y + 200) && (y1 + 225 >= y))

{

isHitSomething = 4;

return 4;

}

}

else if (type == 4)

{

if (lastMovingState == 0)

{

if ((x1 + 108 >= x) && (x1 <= x + 160) && (y1 <= y + 200) && (y1 + 167 >= y))

{

isHitSomething = 8;

return 8;

}

}

else if (lastMovingState == 1)

{

if ((x1 <= x + 160) && (x1 >= x) && (y1 <= y + 200) && (y1 + 167 >= y))

{

isHitSomething = 8;

return 8;

}

}

}

return 0;

}

}

< Rockcannon.h >

namespace game\_framework

{

class RockCannon

{

public:

RockCannon();

void LoadBitmap(); //載入角色動畫

void SetX(int); //設定X

void SetY(int); //設定Y

void SetFixScreen(int, int); //設定砲擊螢幕座標修正

void OnMove(); //移動圖形

void OnShow(); //顯示圖形

void OnShowHit(); //顯示砲彈擊中動畫

void SetNowCharge(int); //設定現在充能

void SetLastMovingState(int); //設定上次移動狀態

void SetUsingState(bool flag); //設定砲彈使用狀態

bool GetUsingState(); //取得砲彈使用狀態

void SetCatchAction(bool); //設定砲彈抓取使用者座標

int collision(int x, int y,bool MonsterisAlive, int monsterNum); //砲彈與怪物碰撞

private:

int x1, y1; //腳色當前座標

int screenX, screenY; //畫面的座標

int hitX, hitY; //子彈擊中座標

void SetScreen(); //設定砲擊螢幕座標

void SetHitXY(); //設定擊中座標

CAnimation normalCannon; //動畫:一般洛克炮

CAnimation chargeCannon; //動畫:充能洛克炮

CAnimation superChargeCannon; //動畫:超級充能洛克炮

CAnimation normalCannonLeft; //動畫:一般洛克炮(左)

CAnimation chargeCannonLeft; //動畫:充能洛克炮(左)

CAnimation superChargeCannonLeft; //動畫:超級充能洛克炮(左)

CAnimation normalCannonHit; //動畫:一般、充能洛克炮擊中

CAnimation normalCannonHitLeft; //動畫:一般、充能洛克炮擊中(左)

CAnimation superChargeCannonHit; //動畫:超級充能洛克炮擊中

CAnimation superChargeCannonHitLeft; //動畫:超級充能洛克炮擊中(左)

void LoadNormalCannonBitmap(); //載入動畫:一般洛克炮

void LoadChargeCannonBitmap(); //載入動畫:充能洛克炮

void LoadSuperChargeCannonBitmap(); //載入動畫:超級充能洛克炮

void LoadNormalCannonLeftBitmap(); //載入動畫:一般洛克炮(左)

void LoadChargeCannonLeftBitmap(); //載入動畫:充能洛克炮(左)

void LoadSuperChargeCannonLeftBitmap(); //載入動畫:超級充能洛克炮(左)

void LoadNormalCannonHitBitmap(); //載入動畫:一般、充能洛克炮擊中

void LoadNormalCannonHitLeftBitmap(); //載入動畫:一般、充能洛克炮擊中(左)

void LoadSuperChargeCannonHitBitmap(); //載入動畫:超級充能洛克炮擊中

void LoadSuperChargeCannonHitLeftBitmap();//載入動畫:超級充能洛克炮擊中(左)

void HitAnimationLock(); //擊中動畫鎖定

int nowCharge; //現在充能

int velocity\_cannon; //洛克炮速度

int distance; //當前移動距離

int lastMovingState; //上次的移動方向

bool usingState; //是否被使用中

bool catchAction; //0:抓取動作，1:不抓取動作

int isHitSomething; //是否擊中到東西

bool isCatchHitXY; //是否擷取擊中座標

bool showLock; //顯示鎖定

};

}

< Rockcannon.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "RockCannon.h"

namespace game\_framework

{

RockCannon::RockCannon()

{

x1 = 0;

y1 = 700;

screenX = 0;

screenY = 0;

hitX = hitY = 0;

nowCharge = 0;

velocity\_cannon = 35;

distance = 0;

lastMovingState = 0;

usingState = false;

catchAction = 0;

showLock = 0;

isHitSomething = 0;

isCatchHitXY = false;

}

void RockCannon::SetNowCharge(int charge)

{

nowCharge = charge;

}

void RockCannon::SetLastMovingState(int flag)

{

if (catchAction == 0)

lastMovingState = flag;

}

void RockCannon::SetX(int x)

{

if (catchAction == 0)

x1 = x;

}

void RockCannon::SetY(int y)

{

if (catchAction == 0)

y1 = y;

}

void RockCannon::SetFixScreen(int fixX,int fixY)

{

if (usingState == true)

{

screenY += fixY;

screenX += fixX;

}

if (usingState == false)

{

hitY += fixY;

hitX += fixX;

}

}

void RockCannon::SetHitXY()

{

if(!isCatchHitXY)

{

hitX = screenX;

hitY = screenY;

isCatchHitXY = true;

}

}

void RockCannon::SetCatchAction(bool flag)

{

catchAction = flag;

}

bool RockCannon::GetUsingState()

{

return usingState;

}

void RockCannon::SetUsingState(bool flag)

{

usingState = flag;

distance = 0;

catchAction = 0;

x1 = 0;

y1 = 0;

if(usingState)

{

normalCannonHit.Reset();

normalCannonHitLeft.Reset();

superChargeCannonHit.Reset();

superChargeCannonHitLeft.Reset();

}

}

void RockCannon::SetScreen()

{

if (lastMovingState == 0)

{

if (x1 >= 900)

screenX = 900 + 170;

if (y1 <= 2700)

{

if (nowCharge > 50)

screenY = 380;

else

screenY = 380 + 84;

}

}

else if (lastMovingState == 1)

{

if (nowCharge > 50)

{

if (x1 >= 900)

screenX = 900 - 222;

if (y1 <= 2700)

screenY = 380;

}

else

{

if (x1 >= 900)

screenX = 900;

if (y1 <= 2700)

screenY = 380 + 84;

}

}

}

void RockCannon::LoadBitmap()

{

LoadNormalCannonBitmap();

LoadChargeCannonBitmap();

LoadSuperChargeCannonBitmap();

LoadNormalCannonLeftBitmap();

LoadChargeCannonLeftBitmap();

LoadSuperChargeCannonLeftBitmap();

LoadNormalCannonHitBitmap();

LoadNormalCannonHitLeftBitmap();

LoadSuperChargeCannonHitBitmap();

LoadSuperChargeCannonHitLeftBitmap();

}

void RockCannon::HitAnimationLock()

{

if (normalCannonHit.IsFinalBitmap())

{

isCatchHitXY = false;

isHitSomething = 0;

normalCannonHit.setToSpecifyBitmap(3);

}

if (normalCannonHitLeft.IsFinalBitmap())

{

isCatchHitXY = false;

isHitSomething = 0;

normalCannonHitLeft.setToSpecifyBitmap(3);

}

if (superChargeCannonHit.IsFinalBitmap())

{

isCatchHitXY = false;

isHitSomething = 0;

superChargeCannonHit.setToSpecifyBitmap(3);

}

if (superChargeCannonHitLeft.IsFinalBitmap())

{

isCatchHitXY = false;

isHitSomething = 0;

superChargeCannonHitLeft.setToSpecifyBitmap(3);

}

}

int RockCannon::collision(int x, int y,bool MonsterisAlive,int monsterNum)

{

int monsWidth, monsHeight;

if (monsterNum == 1)//垃圾怪體型

{

monsWidth = 254;

monsHeight = 225;

}

else if(monsterNum == 2)//夢魘(章魚怪)體型

{

monsWidth = 196;

monsHeight = 90;

}

else if (monsterNum == 3)//蝙蝠體型

{

monsWidth = 120;

monsHeight = 160;

}

else if (monsterNum == 4)//傑洛體型

{

monsWidth = 156;

monsHeight = 171;

}

if (!MonsterisAlive)

{

isHitSomething = 0;

return 0;

}

if (nowCharge > 50)

{

if (lastMovingState == 0)

{

if ((x1 + 222 >= x) && (x1 <= x + monsWidth) && (y1 <= y + monsHeight) && (y1 + 180 >= y))

{

isHitSomething = 8;

SetHitXY();

return 8;

}

}

else if (lastMovingState == 1)

{

if ((x1 <= x + monsWidth) && (x1 >= x) && (y1 <= y + monsHeight) && (y1 + 180 >= y))

{

isHitSomething = 8;

SetHitXY();

return 8;

}

}

}

else if (nowCharge >= 5 && nowCharge <= 50)

{

if (lastMovingState == 0)

{

if ((x1 + 40 >= x) && (x1 <= x + monsWidth) && (y1 <= y + monsHeight) && (y1 + 27 >= y))

{

isHitSomething = 2;

SetHitXY();

return 2;

}

}

else if (lastMovingState == 1)

{

if ((x1 <= x + monsWidth) && (x1 >= x) && (y1 <= y + monsHeight) && (y1 + 27 >= y))

{

isHitSomething = 2;

SetHitXY();

return 2;

}

}

}

else

{

if (lastMovingState == 0)

{

if ((x1 + 19 >= x) && (x1 <= x + monsWidth) && (y1 <= y + monsHeight) && (y1 + 12 >= y))

{

isHitSomething = 1;

SetHitXY();

return 1;

}

}

else if (lastMovingState == 1)

{

if ((x1 <= x + monsWidth) && (x1 >= x) && (y1 <= y + monsHeight) && (y1 + 12 >= y))

{

isHitSomething = 1;

SetHitXY();

return 1;

}

}

}

return 0;

}

void RockCannon::OnMove()

{

if (distance >= 1920)

{

usingState = false;

distance = 0;

catchAction = 0;

x1 = 0;

y1 = 0;

}

if (catchAction == 0)

showLock = 0;

if (usingState)

{

normalCannon.OnMove();

chargeCannon.OnMove();

superChargeCannon.OnMove();

normalCannonLeft.OnMove();

chargeCannonLeft.OnMove();

superChargeCannonLeft.OnMove();

if (nowCharge > 50)

{

if (lastMovingState == 0)

{

x1 += velocity\_cannon \* 2;

screenX += velocity\_cannon \* 2;

distance += velocity\_cannon \* 2;

}

else if (lastMovingState == 1)

{

x1 -= velocity\_cannon \* 2;

screenX -= velocity\_cannon \* 2;

distance += velocity\_cannon \* 2;

}

}

else if (nowCharge <= 50)

{

if (lastMovingState == 0)

{

x1 += velocity\_cannon;

screenX += velocity\_cannon;

distance += velocity\_cannon;

}

else if (lastMovingState == 1)

{

x1 -= velocity\_cannon;

screenX -= velocity\_cannon;

distance += velocity\_cannon;

}

}

}

else

{

normalCannonHit.OnMove();

normalCannonHitLeft.OnMove();

superChargeCannonHit.OnMove();

superChargeCannonHitLeft.OnMove();

}

}

void RockCannon::OnShow()

{

if (showLock == 0)

SetScreen();

showLock = 1;

int tmp = x1, tmp2 = y1;

if (x1 >= 900)

x1 = screenX;

if (y1 <= 2700)

y1 = screenY;

normalCannon.SetTopLeft(x1, y1);

chargeCannon.SetTopLeft(x1, y1);

superChargeCannon.SetTopLeft(x1, y1);

normalCannonLeft.SetTopLeft(x1, y1);

chargeCannonLeft.SetTopLeft(x1, y1);

superChargeCannonLeft.SetTopLeft(x1, y1);

if (lastMovingState == 0 && usingState == true)

{

if (nowCharge > 50)

superChargeCannon.OnShow();

else if ((nowCharge <= 50) && (nowCharge > 5))

chargeCannon.OnShow();

else

normalCannon.OnShow();

}

else if (lastMovingState == 1 && usingState == true)

{

if (nowCharge > 50)

superChargeCannonLeft.OnShow();

else if ((nowCharge <= 50) && (nowCharge > 5))

chargeCannonLeft.OnShow();

else

normalCannonLeft.OnShow();

}

x1 = tmp;

y1 = tmp2;

}

void RockCannon::OnShowHit()

{

normalCannonHit.SetTopLeft(hitX, hitY);

normalCannonHitLeft.SetTopLeft(hitX, hitY);

superChargeCannonHit.SetTopLeft(hitX, hitY);

superChargeCannonHitLeft.SetTopLeft(hitX, hitY);

HitAnimationLock();

if (lastMovingState == 0 && usingState == false)

{

if (isHitSomething > 0 && isHitSomething < 8)

normalCannonHit.OnShow();

else if (isHitSomething == 8)

superChargeCannonHit.OnShow();

}

else if (lastMovingState == 1 && usingState == false)

{

if (isHitSomething > 0 && isHitSomething < 8)

normalCannonHitLeft.OnShow();

else if (isHitSomething == 8)

superChargeCannonHitLeft.OnShow();

}

}

void RockCannon::LoadNormalCannonBitmap()

{

normalCannon.AddBitmap("RES\\attack\\attack.bmp", RGB(255, 255, 255));

}

void RockCannon::LoadChargeCannonBitmap()

{

chargeCannon.AddBitmap("RES\\attack\\charge1 attack.bmp", RGB(255, 255, 255));

}

void RockCannon::LoadSuperChargeCannonBitmap()

{

superChargeCannon.AddBitmap("RES\\attack\\charge attack.bmp", RGB(255, 255, 255));

superChargeCannon.AddBitmap("RES\\attack\\charge attack2.bmp", RGB(255, 255, 255));

superChargeCannon.AddBitmap("RES\\attack\\charge attack3.bmp", RGB(255, 255, 255));

superChargeCannon.AddBitmap("RES\\attack\\charge attack4.bmp", RGB(255, 255, 255));

}

void RockCannon::LoadNormalCannonLeftBitmap()

{

normalCannonLeft.AddBitmap("RES\\attack\\attack left.bmp", RGB(255, 255, 255));

}

void RockCannon::LoadChargeCannonLeftBitmap()

{

chargeCannonLeft.AddBitmap("RES\\attack\\charge1 attack left.bmp", RGB(255, 255, 255));

}

void RockCannon::LoadSuperChargeCannonLeftBitmap()

{

superChargeCannonLeft.AddBitmap("RES\\attack\\charge attack left.bmp", RGB(255, 255, 255));

superChargeCannonLeft.AddBitmap("RES\\attack\\charge attack left2.bmp", RGB(255, 255, 255));

superChargeCannonLeft.AddBitmap("RES\\attack\\charge attack left3.bmp", RGB(255, 255, 255));

superChargeCannonLeft.AddBitmap("RES\\attack\\charge attack left4.bmp", RGB(255, 255, 255));

}

void RockCannon::LoadNormalCannonHitBitmap()

{

normalCannonHit.AddBitmap("RES\\attack\\attack hit.bmp", RGB(255, 255, 255));

normalCannonHit.AddBitmap("RES\\attack\\attack hit2.bmp", RGB(255, 255, 255));

normalCannonHit.AddBitmap("RES\\attack\\attack hit3.bmp", RGB(255, 255, 255));

normalCannonHit.AddBitmap("RES\\attack\\attack hit4.bmp", RGB(255, 255, 255));

normalCannonHit.SetDelayCount(2);

}

void RockCannon::LoadNormalCannonHitLeftBitmap()

{

normalCannonHitLeft.AddBitmap("RES\\attack\\attack hit.bmp", RGB(255, 255, 255));

normalCannonHitLeft.AddBitmap("RES\\attack\\attack hit2left.bmp", RGB(255, 255, 255));

normalCannonHitLeft.AddBitmap("RES\\attack\\attack hit3left.bmp", RGB(255, 255, 255));

normalCannonHitLeft.AddBitmap("RES\\attack\\attack hit4left.bmp", RGB(255, 255, 255));

normalCannonHitLeft.SetDelayCount(2);

}

void RockCannon::LoadSuperChargeCannonHitBitmap()

{

superChargeCannonHit.AddBitmap("RES\\attack\\charge attack hit.bmp", RGB(255, 255, 255));

superChargeCannonHit.AddBitmap("RES\\attack\\charge attack hit2.bmp", RGB(255, 255, 255));

superChargeCannonHit.AddBitmap("RES\\attack\\charge attack hit3.bmp", RGB(255, 255, 255));

superChargeCannonHit.AddBitmap("RES\\attack\\charge attack hit4.bmp", RGB(255, 255, 255));

superChargeCannonHit.SetDelayCount(2);

}

void RockCannon::LoadSuperChargeCannonHitLeftBitmap()

{

superChargeCannonHitLeft.AddBitmap("RES\\attack\\charge attack hitleft.bmp", RGB(255, 255, 255));

superChargeCannonHitLeft.AddBitmap("RES\\attack\\charge attack hit2left.bmp", RGB(255, 255, 255));

superChargeCannonHitLeft.AddBitmap("RES\\attack\\charge attack hit3left.bmp", RGB(255, 255, 255));

superChargeCannonHitLeft.AddBitmap("RES\\attack\\charge attack hit4left.bmp", RGB(255, 255, 255));

superChargeCannonHitLeft.SetDelayCount(2);

}

}

< Terrain.h >

namespace game\_framework

{

class Terrain //地形(箱子、方塊)

{

public:

Terrain();

void LoadBitMap(); //載入圖片

void Initialize(); //初始化參數

void GetLastRockmanXY(int x, int y);//地圖抓取洛克人移動前之座標

void GetNowRockmanXY(int x, int y); //地圖抓取洛克人移動後之座標

void MoveScreen(); //畫面移動

int GetLastX(); //取得洛克人移動前之座標X

int GetLastY(); //取得洛克人移動前之座標Y

int crashleft(); //左邊碰撞判斷

int crashright(); //右邊碰撞判斷

int crashtop(); //上邊碰撞判斷

int crashdown(); //下邊碰撞判斷

bool MonsterCollision(); //洛克人與所有怪物之碰撞

int MosterCannonCollision(); //洛克人與所有怪物砲彈之碰撞

bool GetIsBossCannon(); //取得是否為BOSS的砲彈

//////////////////////// Monster

Monster getMonster(int index); //取得垃圾怪

Nightmare getNightmare(int index); //取得夢魘

Bat getBat(int index); //取得蝙蝠

Boss getBoss(); //取得夢魘傑洛

void setMonsterLife(int index, int damage, int monsterNum); //設定指定怪物之生命

int getMonsterLife(int index, int monsterNum); //取得指定怪物之生命

bool IsBossStage(); //是否為BOSS戰鬥

bool IsBossDead(); //是否BOSS已死亡

void setLifeToZero(); //所有小怪血量強制歸0(作弊)

void OnShow(); //顯示圖片

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

protected:

CMovingBitmap background; // 背景圖

CMovingBitmap background2;// 背景圖2

CMovingBitmap background3;// 背景圖3

CMovingBitmap background4;// 背景圖4

CMovingBitmap block; // 障礙物

CMovingBitmap deadBlock; // 尖刺、無底洞

Boss boss; // 夢魘傑洛

Monster X6\_1[8]; // 垃圾怪

Nightmare X6\_2[6]; // 夢魘

Bat X4\_1[6]; // 蝙蝠

int picX, picY; // 畫面移動偏移量

int wallX, wallY; // 障礙物畫面移動偏移量

int lastX, lastY; // 洛克人移動之前座標

int nowX, nowY; // 洛克人移動之後座標

int map[31][315]; // 地圖障礙物陣列

bool isBossCannon; // 是否為BOSS砲彈

};

}

< Terrain.cpp >

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "BossCannon.h"

#include "Boss.h"

#include "Trashcannon.h"

#include "Monster.h"

#include "nightmareCannon.h"

#include "nightmare.h"

#include "Bat.h"

#include "Terrain.h"

namespace game\_framework

{

Terrain::Terrain()

{}

void Terrain::LoadBitMap()

{

background.LoadBitmap("Bitmaps\\map.bmp"); // 載入背景的圖形

background2.LoadBitmap("Bitmaps\\map2.bmp");

background3.LoadBitmap("Bitmaps\\map3.bmp");

background4.LoadBitmap("Bitmaps\\map4.bmp");

block.LoadBitmap("Bitmaps\\wall.bmp");

deadBlock.LoadBitmap("Bitmaps\\deadBlock.bmp");

boss.LoadBitMap();

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

X6\_1[i].LoadBitMap();

for (int i = 0; i < 6; i++)

X6\_2[i].LoadBitMap();

for (int i = 0; i < 6; i++)

X4\_1[i].LoadBitMap();

}

void Terrain::OnShow()

{

background.ShowBitmap(); // 貼上背景圖

background2.ShowBitmap();

background3.ShowBitmap();

background4.ShowBitmap();

/\*

for (int i = 0; i < 31; i++)//地形規劃方塊顯示

{

for (int j = 0; j < 315; j++)

{

if (map[i][j] == 1)

{

block.SetTopLeft(wallX + j \* 80, wallY + i \* 80 - 1492);

block.ShowBitmap();

}

else if (map[i][j] == 2)

{

deadBlock.SetTopLeft(wallX + j \* 80, wallY + i \* 80 - 1492);

deadBlock.ShowBitmap();

}

}

}\*/

if (boss.getAlive())

boss.OnShow();

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

if (X6\_1[i].getAlive())

X6\_1[i].OnShow();

else

X6\_1[i].OnShowBoom();

for (int i = 0; i < 6; i++)

if (X6\_2[i].getAlive())

X6\_2[i].OnShow();

else

X6\_2[i].OnShowBoom();

for (int i = 0; i < 6; i++)

if (X4\_1[i].getAlive())

X4\_1[i].OnShow();

else

X4\_1[i].OnShowBoom();

}

void Terrain::Initialize()

{

picX = picY = 0;

wallX = wallY = 0;

lastX = lastY = 0;

/////////////monster/////////////

X6\_1[0].Initialize();

X6\_1[0].setXY(1920, 1295);

X6\_1[0].setScreenXY(1920, 1295);

X6\_1[1].Initialize();

X6\_1[1].setXY(5360, 1370);

X6\_1[1].setScreenXY(5360, 1370);

X6\_1[2].Initialize();

X6\_1[2].setXY(7520, 1500);

X6\_1[2].setScreenXY(7520, 1500);

X6\_1[3].Initialize();

X6\_1[3].setXY(10560, 640);

X6\_1[3].setScreenXY(10560, 640);

X6\_1[4].Initialize();

X6\_1[4].setXY(14640, 1520);

X6\_1[4].setScreenXY(14640, 1520);

X6\_1[5].Initialize();

X6\_1[5].setXY(18240, 1560);

X6\_1[5].setScreenXY(18240, 1560);

X6\_1[6].Initialize();

X6\_1[6].setXY(23120, 530);

X6\_1[6].setScreenXY(23120, 530);

boss.Initialize();

boss.setXY(24560, 1660);

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

srand((unsigned)time(NULL));

X6\_2[0].Initialize();

X6\_2[0].setXY(3440, 1440);

X6\_2[1].Initialize();

X6\_2[1].setXY(7040, 1600);

X6\_2[2].Initialize();

X6\_2[2].setXY(9360, 320);

X6\_2[3].Initialize();

X6\_2[3].setXY(13600, 1440);

X6\_2[4].Initialize();

X6\_2[4].setXY(17440, 1520);

X6\_2[5].Initialize();

X6\_2[5].setXY(21200, 1600);

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

srand((unsigned)time(NULL));

X6\_2[0].Initialize();

X6\_2[0].setXY(3440, 1440);

X6\_2[1].Initialize();

X6\_2[1].setXY(7040, 1600);

X6\_2[2].Initialize();

X6\_2[2].setXY(9360, 320);

X6\_2[3].Initialize();

X6\_2[3].setXY(13600, 1440);

X6\_2[4].Initialize();

X6\_2[4].setXY(17440, 1520);

X6\_2[5].Initialize();

X6\_2[5].setXY(21200, 1600);

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

X4\_1[0].Initialize();

X4\_1[0].setXY(3440, 1440);

X4\_1[1].Initialize();

X4\_1[1].setXY(7040, 1600);

X4\_1[2].Initialize();

X4\_1[2].setXY(9360, 320);

X4\_1[3].Initialize();

X4\_1[3].setXY(13600, 1440);

X4\_1[4].Initialize();

X4\_1[4].setXY(17440, 1520);

X4\_1[5].Initialize();

X4\_1[5].setXY(21200, 1600);

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

for (int i = 0; i < 31; i++)

{

for (int j = 0; j < 315; j++)

map[i][j] = 0;

}

map[5][5] = 1;

map[26][0] = 1;

map[26][1] = 1;

map[26][2] = 1;

map[26][3] = 1;

map[26][4] = 1;

map[27][4] = 1;

map[30][2] = 2;

map[30][3] = 2;

map[30][4] = 2;

map[30][5] = 2;

map[30][6] = 2;

map[30][7] = 2;

map[30][8] = 2;

map[30][9] = 2;

map[30][10] = 2;

map[30][11] = 2;

map[30][12] = 2;

map[30][13] = 2;

map[30][14] = 2;

map[30][15] = 2;

map[30][16] = 2;

map[30][17] = 2;

map[30][18] = 2;

map[30][19] = 2;

map[30][20] = 2;

map[30][21] = 2;

map[30][22] = 2;

map[30][23] = 2;

map[30][24] = 2;

map[30][25] = 2;

map[30][26] = 2;

map[30][27] = 2;

map[30][28] = 2;

map[30][29] = 2;

map[30][30] = 2;

map[30][31] = 2;

map[30][32] = 2;

map[30][33] = 2;

map[30][34] = 2;

map[30][35] = 2;

map[30][36] = 2;

map[30][37] = 2;

map[30][38] = 2;

map[30][39] = 2;

map[30][40] = 2;

map[30][41] = 2;

map[30][42] = 2;

map[30][43] = 2;

map[30][44] = 2;

map[30][45] = 2;

map[30][46] = 2;

map[30][47] = 2;

map[30][48] = 2;

map[30][49] = 2;

map[30][50] = 2;

map[30][51] = 2;

map[30][52] = 2;

map[30][53] = 2;

map[30][54] = 2;

map[30][55] = 2;

map[30][56] = 2;

map[30][57] = 2;

map[30][58] = 2;

map[30][59] = 2;

map[30][60] = 2;

map[30][61] = 2;

map[30][62] = 2;

map[30][63] = 2;

map[30][64] = 2;

map[30][65] = 2;

map[30][66] = 2;

map[30][67] = 2;

map[30][68] = 2;

map[30][69] = 2;

map[30][70] = 2;

map[30][71] = 2;

map[30][72] = 2;

map[30][73] = 2;

map[30][74] = 2;

map[30][75] = 2;

map[30][76] = 2;

map[30][77] = 2;

map[30][78] = 2;

map[30][79] = 2;

map[30][80] = 2;

//---------------------

map[24][9] = 1;

map[24][10] = 1;

map[24][11] = 1;

map[24][12] = 1;

map[25][9] = 1;

map[25][10] = 1;

map[25][11] = 1;

map[25][12] = 1;

map[22][16] = 1;

map[22][17] = 1;

map[22][18] = 1;

map[22][19] = 1;

map[23][16] = 1;

map[23][17] = 1;

map[23][18] = 1;

map[23][19] = 1;

map[19][23] = 1;

map[19][24] = 1;

map[19][25] = 1;

map[19][26] = 1;

map[20][23] = 1;

map[20][24] = 1;

map[20][25] = 1;

map[20][26] = 1;

map[22][31] = 1;

map[22][32] = 1;

map[22][33] = 1;

map[23][31] = 1;

map[23][32] = 1;

map[23][33] = 1;

map[19][36] = 1;

map[19][37] = 1;

map[19][38] = 1;

map[19][39] = 1;

map[20][36] = 1;

map[20][37] = 1;

map[20][38] = 1;

map[20][39] = 1;

map[22][41] = 1;

map[22][42] = 1;

map[22][43] = 1;

map[22][44] = 1;

map[23][41] = 1;

map[23][42] = 1;

map[23][43] = 1;

map[23][44] = 1;

map[20][49] = 1;

map[20][50] = 1;

map[20][51] = 1;

map[21][49] = 1;

map[21][50] = 1;

map[21][51] = 1;

map[22][57] = 1;

map[22][58] = 1;

map[22][59] = 1;

map[23][57] = 1;

map[23][58] = 1;

map[23][59] = 1;

map[20][67] = 1;

map[20][68] = 1;

map[20][69] = 1;

map[21][67] = 1;

map[21][68] = 1;

map[21][69] = 1;

//---------------------------

map[26][77] = 1;

map[27][77] = 1;

map[26][78] = 1;

map[26][79] = 1;

map[26][80] = 1;

map[26][81] = 1;

map[26][82] = 1;

map[26][83] = 1;

map[26][84] = 1;

map[26][85] = 1;

map[26][86] = 1;

map[26][87] = 1;

map[26][88] = 1;

map[26][89] = 1;

map[26][90] = 1;

map[26][91] = 1;

map[26][92] = 1;

map[26][93] = 1;

map[22][94] = 1;

map[23][94] = 1;

map[24][94] = 1;

map[25][94] = 1;

map[26][94] = 1;

map[22][95] = 1;

map[22][96] = 1;

map[22][97] = 1;

map[22][98] = 1;

map[6][88] = 1;

map[7][88] = 1;

map[8][88] = 1;

map[9][88] = 1;

map[10][88] = 1;

map[11][88] = 1;

map[12][88] = 1;

map[13][88] = 1;

map[14][88] = 1;

map[15][88] = 1;

map[0][96] = 1;

map[0][95] = 1;

map[0][94] = 1;

map[0][93] = 1;

map[0][92] = 1;

map[0][91] = 1;

map[0][90] = 1;

map[0][89] = 1;

map[0][88] = 1;

map[0][87] = 1;

map[0][86] = 1;

map[0][85] = 1;

map[0][84] = 1;

map[0][83] = 1;

map[0][82] = 1;

map[1][82] = 1;

map[2][82] = 1;

map[3][82] = 1;

map[4][82] = 1;

map[5][82] = 1;

map[6][82] = 1;

map[6][83] = 1;

map[6][84] = 1;

map[6][85] = 1;

map[6][86] = 1;

map[6][87] = 1;

map[6][88] = 1;

map[6][89] = 1;

map[7][89] = 1;

map[6][90] = 1;

map[7][90] = 1;

map[6][91] = 1;

map[7][91] = 1;

map[6][92] = 1;

map[7][92] = 1;

map[22][98] = 1;

map[21][98] = 1;

map[20][98] = 1;

map[19][98] = 1;

map[18][98] = 1;

map[17][98] = 1;

map[16][98] = 1;

map[15][98] = 1;

map[14][98] = 1;

map[14][98] = 1;

map[14][99] = 1;

map[14][100] = 1;

map[14][101] = 1;

map[14][102] = 1;

map[14][103] = 1;

map[14][104] = 1;

map[14][105] = 1;

map[13][105] = 1;

map[12][105] = 1;

map[11][105] = 1;

map[10][105] = 1;

map[9][105] = 1;

map[8][105] = 1;

map[8][105] = 1;

map[8][106] = 1;

map[8][107] = 1;

map[8][108] = 1;

map[8][109] = 1;

map[8][110] = 1;

map[8][111] = 1;

map[8][112] = 1;

map[8][113] = 1;

map[8][114] = 1;

map[8][115] = 1;

map[8][116] = 1;

map[8][117] = 1;

map[8][118] = 1;

map[8][119] = 1;

map[8][120] = 1;

map[8][121] = 1;

map[8][122] = 1;

map[8][123] = 1;

map[8][124] = 1;

map[8][125] = 1;

map[8][126] = 1;

map[8][127] = 1;

map[8][128] = 1;

map[8][129] = 1;

map[8][129] = 1;

map[9][129] = 1;

map[10][129] = 1;

map[11][129] = 1;

map[11][129] = 1;

map[11][130] = 1;

map[11][131] = 1;

map[11][132] = 1;

map[11][133] = 1;

map[11][134] = 1;

map[11][135] = 1;

map[11][136] = 1;

map[11][137] = 1;

map[11][138] = 1;

map[11][139] = 1;

map[11][140] = 1;

map[11][141] = 1;

map[11][142] = 1;

map[11][143] = 1;

map[11][144] = 1;

map[11][145] = 1;

map[11][146] = 1;

map[11][147] = 1;

map[11][148] = 1;

map[11][148] = 1;

map[12][148] = 1;

map[13][148] = 1;

map[14][148] = 1;

map[14][148] = 1;

map[14][149] = 1;

map[14][150] = 1;

map[14][151] = 1;

map[14][152] = 1;

map[14][153] = 1;

map[14][153] = 1;

map[15][153] = 1;

map[16][153] = 1;

map[17][153] = 1;

map[18][153] = 1;

map[19][153] = 1;

map[20][153] = 1;

map[20][153] = 1;

map[20][154] = 1;

map[20][155] = 1;

map[20][156] = 1;

map[20][157] = 1;

map[20][157] = 1;

map[21][157] = 1;

map[22][157] = 1;

map[23][157] = 1;

map[23][157] = 1;

map[23][158] = 1;

map[23][159] = 1;

map[23][160] = 1;

map[23][161] = 1;

map[23][162] = 1;

map[23][163] = 1;

map[23][164] = 1;

map[23][165] = 1;

map[23][166] = 1;

map[23][167] = 1;

map[13][166] = 1;

map[13][167] = 1;

map[12][166] = 1;

map[12][167] = 1;

map[11][166] = 1;

map[11][167] = 1;

map[11][168] = 1;

map[11][169] = 1;

map[11][170] = 1;

map[11][171] = 1;

map[11][172] = 1;

map[11][173] = 1;

map[10][173] = 2;

map[9][173] = 2;

map[8][173] = 2;

map[7][173] = 2;

map[6][173] = 2;

map[5][173] = 2;

map[4][173] = 2;

map[3][173] = 2;

map[2][173] = 2;

map[1][173] = 2;

map[0][173] = 2;

map[24][168] = 2;

map[24][169] = 2;

map[24][170] = 2;

map[24][171] = 2;

map[24][172] = 2;

map[24][173] = 2;

map[23][174] = 1;

map[23][175] = 1;

map[23][176] = 1;

map[23][177] = 1;

map[24][178] = 2;

map[24][179] = 2;

map[24][180] = 2;

map[24][181] = 2;

map[24][182] = 2;

map[23][183] = 1;

map[23][184] = 1;

map[23][185] = 1;

map[23][186] = 1;

map[22][183] = 1;

map[22][184] = 1;

map[22][185] = 1;

map[22][186] = 1;

map[24][187] = 2;

map[24][188] = 2;

map[24][189] = 2;

map[24][190] = 2;

map[24][191] = 2;

map[24][192] = 2;

map[24][193] = 2;

map[24][194] = 2;

map[23][195] = 1;

map[23][196] = 1;

map[23][197] = 1;

map[23][198] = 1;

map[24][199] = 2;

map[24][200] = 2;

map[24][201] = 2;

map[24][202] = 2;

map[24][203] = 2;

map[24][204] = 2;

map[23][205] = 1;

map[23][206] = 1;

map[23][207] = 1;

map[23][208] = 1;

map[23][209] = 1;

map[24][210] = 2;

map[24][211] = 2;

map[24][212] = 2;

map[24][213] = 2;

map[24][214] = 2;

map[23][215] = 1;

map[23][216] = 1;

map[23][217] = 1;

map[23][218] = 1;

map[23][219] = 1;

map[22][215] = 1;

map[22][216] = 1;

map[22][217] = 1;

map[22][218] = 1;

map[22][219] = 1;

map[24][220] = 2;

map[24][221] = 2;

map[24][222] = 2;

map[24][223] = 2;

map[24][224] = 2;

map[24][225] = 2;

map[24][226] = 2;

map[24][227] = 2;

map[23][228] = 1;

map[23][229] = 1;

map[23][230] = 1;

map[23][231] = 1;

map[24][232] = 2;

map[24][233] = 2;

map[24][234] = 2;

map[24][235] = 2;

map[24][236] = 2;

map[24][237] = 2;

map[24][238] = 2;

map[24][239] = 2;

map[23][240] = 1;

map[23][241] = 1;

map[23][242] = 1;

map[23][243] = 1;

map[23][244] = 1;

map[24][245] = 2;

map[24][246] = 2;

map[24][247] = 2;

map[24][248] = 2;

map[24][249] = 2;

map[24][250] = 2;

map[24][251] = 2;

map[24][252] = 2;

map[23][253] = 1;

map[23][254] = 1;

map[23][255] = 1;

map[23][256] = 1;

map[23][257] = 1;

map[23][258] = 1;

map[23][259] = 1;

map[23][260] = 1;

map[23][261] = 1;

map[23][262] = 1;

map[23][263] = 1;

map[23][264] = 1;

map[23][265] = 1;

map[24][266] = 2;

map[24][267] = 2;

map[24][268] = 2;

map[24][269] = 2;

map[24][270] = 2;

map[24][271] = 2;

map[24][272] = 2;

map[24][273] = 2;

map[24][274] = 2;

map[23][274] = 2;

map[22][274] = 2;

map[21][274] = 2;

map[18][270] = 1;

map[18][271] = 1;

map[18][272] = 1;

map[18][273] = 1;

map[18][274] = 1;

map[18][275] = 1;

map[18][276] = 1;

map[18][277] = 1;

map[18][278] = 1;

map[19][270] = 1;

map[19][271] = 1;

map[19][272] = 1;

map[19][273] = 1;

map[19][274] = 1;

map[20][270] = 1;

map[20][271] = 1;

map[20][272] = 1;

map[20][273] = 1;

map[20][274] = 1;

map[18][279] = 1;

map[17][279] = 1;

map[16][279] = 1;

map[15][279] = 1;

map[14][279] = 1;

map[13][279] = 1;

map[12][279] = 1;

map[12][280] = 1;

map[12][281] = 1;

map[12][282] = 1;

map[12][283] = 1;

map[11][283] = 1;

map[10][283] = 1;

map[10][283] = 1;

map[10][284] = 1;

map[10][285] = 1;

map[10][286] = 1;

map[10][287] = 1;

map[10][288] = 1;

map[10][289] = 1;

map[10][290] = 1;

map[10][291] = 1;

map[10][292] = 1;

map[10][293] = 1;

map[10][293] = 1;

map[11][293] = 1;

map[12][293] = 1;

map[12][292] = 1;

map[12][291] = 1;

map[12][290] = 1;

map[12][289] = 1;

map[12][288] = 1;

map[12][287] = 1;

map[13][287] = 1;

map[14][287] = 1;

map[15][287] = 1;

map[16][287] = 1;

map[17][287] = 1;

map[18][287] = 1;

map[19][287] = 1;

map[20][287] = 1;

map[24][290] = 2;

map[24][289] = 2;

map[24][288] = 2;

map[24][287] = 2;

map[24][286] = 2;

map[24][285] = 2;

map[24][284] = 2;

map[21][283] = 2;

map[22][283] = 2;

map[23][283] = 2;

map[24][283] = 2;

//--------------------boss stage

map[23][309] = 1;

map[23][308] = 1;

map[23][307] = 1;

map[23][306] = 1;

map[23][305] = 1;

map[23][304] = 1;

map[23][303] = 1;

map[23][302] = 1;

map[23][301] = 1;

map[23][300] = 1;

map[23][299] = 1;

map[23][298] = 1;

map[23][297] = 1;

map[23][296] = 1;

map[23][295] = 1;

map[23][294] = 1;

map[23][293] = 1;

map[23][292] = 1;

map[23][291] = 1;

//-------------------------boss stage

map[24][310] = 2;

map[24][311] = 2;

map[24][312] = 2;

map[23][313] = 1;

map[22][313] = 1;

map[21][313] = 1;

map[20][313] = 1;

map[19][313] = 1;

map[18][313] = 1;

map[17][313] = 1;

map[16][313] = 1;

map[15][313] = 1;

map[14][313] = 1;

map[13][313] = 1;

map[12][313] = 1;

map[12][312] = 1;

map[12][311] = 1;

map[12][310] = 1;

map[12][309] = 1;

map[12][308] = 1;

map[12][307] = 1;

map[12][306] = 1;

map[12][305] = 1;

map[12][304] = 1;

map[12][303] = 1;

map[12][302] = 1;

map[12][301] = 1;

map[12][300] = 1;

map[12][299] = 1;

map[11][299] = 1;

map[10][299] = 1;

map[9][299] = 1;

map[8][299] = 1;

map[7][299] = 1;

map[6][299] = 1;

map[5][299] = 1;

map[4][299] = 1;

map[3][299] = 1;

map[3][298] = 1;

map[3][297] = 1;

map[3][296] = 1;

map[3][295] = 1;

map[3][294] = 1;

map[3][293] = 1;

map[3][292] = 1;

map[3][291] = 1;

map[3][290] = 1;

map[3][289] = 1;

map[3][288] = 1;

map[3][287] = 1;

map[3][286] = 1;

map[3][285] = 1;

map[3][284] = 1;

map[3][283] = 1;

map[2][283] = 1;

map[1][283] = 1;

map[0][283] = 1;

map[0][282] = 1;

map[0][281] = 1;

map[0][280] = 1;

map[0][279] = 1;

map[0][278] = 1;

map[0][277] = 1;

map[0][276] = 1;

map[0][275] = 1;

background.SetTopLeft(0, 0 - 1492); // 設定背景的起始座標

background2.SetTopLeft(background.Width(), 0 - 1492);

background3.SetTopLeft(background.Width() + background2.Width(), 0 - 1492);

background4.SetTopLeft(background.Width() + background2.Width() + background3.Width(), 0 - 1492);

}

void Terrain::setLifeToZero()

{

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

X6\_1[i].deductLife(10);

for (int i = 0; i < 6; i++)

X6\_2[i].deductLife(10);

for (int i = 0; i < 6; i++)

X4\_1[i].deductLife(10);

}

Monster Terrain::getMonster(int index)

{

return X6\_1[index];

}

Nightmare Terrain::getNightmare(int index)

{

return X6\_2[index];

}

Bat Terrain::getBat(int index)

{

return X4\_1[index];

}

Boss Terrain::getBoss()

{

return boss;

}

void Terrain::GetLastRockmanXY(int x, int y)

{

lastX = x;

lastY = y;

}

void Terrain::GetNowRockmanXY(int x, int y)

{

nowX = x;

nowY = y;

}

int Terrain::GetLastX()

{

return lastX;

}

int Terrain::GetLastY()

{

return lastY;

}

void Terrain::MoveScreen()

{

int mon\_posX, mon\_posY;

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

{

mon\_posX = X6\_1[i].getScreenX();

mon\_posY = X6\_1[i].getScreenY();

if (nowX > 900)

mon\_posX -= (nowX - lastX);

if (nowY < 2700)

mon\_posY -= (nowY - lastY);

X6\_1[i].setScreenXY(mon\_posX, mon\_posY);

X6\_1[i].DeterminAttack(nowX, nowY);

X6\_1[i].FixCannonScreenXY((nowX - lastX), (nowY - lastY));

if (nowX <= 900)

{

if (i == 0)

X6\_1[0].setScreenXY(1920, mon\_posY);

if (i == 1)

X6\_1[1].setScreenXY(5360, mon\_posY);

if (i == 2)

X6\_1[2].setScreenXY(7520, mon\_posY);

if (i == 3)

X6\_1[3].setScreenXY(10560, mon\_posY);

if (i == 4)

X6\_1[4].setScreenXY(14640, mon\_posY);

if (i == 5)

X6\_1[5].setScreenXY(18240, mon\_posY);

if (i == 6)

X6\_1[6].setScreenXY(23120, mon\_posY);

if (i == 7)

X6\_1[7].setScreenXY(23360, mon\_posY);

}

}

if (nowX > 900)

{

wallX -= (nowX - lastX);

picX -= (nowX - lastX);

}

else

{

wallX = 0;

picX = 0;

}

if (nowY < 2700)

{

wallY -= (nowY - lastY);

picY -= (nowY - lastY);

}

for (int i = 0; i < 6; i++)

{

X6\_2[i].setScreenXY(picX, picY);

X6\_2[i].DeterminAttack(nowX, nowY);

X6\_2[i].FixCannonScreenXY((nowX - lastX), (nowY - lastY));

X4\_1[i].setScreenXY(picX, picY);

X4\_1[i].DeterminAttack(nowX, nowY);

}

boss.setScreen\_XY(picX, picY);

boss.DeterminAttack(nowX, nowY);

boss.FixCannonScreenXY((nowX - lastX), (nowY - lastY));

background.SetTopLeft(picX, picY - 1492);

background2.SetTopLeft(picX + background.Width(), picY - 1492);

background3.SetTopLeft(picX + background.Width() + background2.Width(), picY - 1492);

background4.SetTopLeft(picX + background.Width() + background2.Width() + background3.Width(), picY - 1492);

}

int Terrain::crashleft()

{

int map\_x, map\_y;

map\_x = nowX / 80;

for (int i = 0; i < 200; i++)

{

map\_y = (nowY + i) / 80;

if (map[map\_y][map\_x] == 1)

{

return map\_x \* 80 + 80;

}

if (map[map\_y][map\_x] == 2)

return -1;

}

return 0;

}

int Terrain::crashright()

{

int map\_x, map\_y;

map\_x = (nowX + 160) / 80;

for (int i = 0; i < 200; i++)

{

map\_y = (nowY + i) / 80;

if (map[map\_y][map\_x] == 1)

{

return map\_x \* 80;

}

if (map[map\_y][map\_x] == 2)

return -1;

}

return 0;

}

int Terrain::crashtop()

{

int map\_x, map\_y;

map\_y = nowY / 80;

for (int i = 20; i <= 140; i++)

{

map\_x = (nowX + i) / 80;

if (map[map\_y][map\_x] == 1)

{

return map\_y \* 80 + 80;

}

if (map[map\_y][map\_x] == 2)

return -1;

}

return 0;

}

int Terrain::crashdown()

{

int map\_x, map\_y;

map\_y = (nowY + 200) / 80;

for (int i = 20; i <= 140; i++)

{

map\_x = (nowX + i) / 80;

if (map[map\_y][map\_x] == 1)

{

return map\_y \* 80;

}

if (map[map\_y][map\_x] == 2)

return -1;

}

return 0;

}

bool Terrain::MonsterCollision()

{

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

if (X6\_1[i].MonsterCollision(nowX, nowY))

return true;

for (int i = 0; i < 6; i++)

if (X6\_2[i].MonsterCollision(nowX, nowY))

return true;

for (int i = 0; i < 6; i++)

if (X4\_1[i].MonsterCollision(nowX, nowY))

return true;

if (boss.MonsterCollision(nowX, nowY))

return true;

return false;

}

int Terrain::MosterCannonCollision()

{

int tmp;

isBossCannon = false;

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

{

tmp = X6\_1[i].MonsterCannonCollision(nowX, nowY);

if (tmp != 0)

return tmp;

}

for (int i = 0; i < 6; i++)

{

tmp = X6\_2[i].MonsterCannonCollision(nowX, nowY);

if (tmp != 0)

return tmp;

}

tmp = boss.MonsterCannonCollision(nowX, nowY);

if (tmp != 0)

{

isBossCannon = true;

return tmp;

}

return 0;

}

bool Terrain::IsBossStage()

{

return boss.GetStartAttack();

}

bool Terrain::IsBossDead()

{

return !boss.getAlive();

}

bool Terrain::GetIsBossCannon()

{

return isBossCannon;

}

void Terrain::setMonsterLife(int index, int damage, int monsterNum)

{

if (monsterNum == 1)

X6\_1[index].deductLife(damage);

else if (monsterNum == 2)

X6\_2[index].deductLife(damage);

else if (monsterNum == 3)

X4\_1[index].deductLife(damage);

else if (monsterNum == 4)

boss.deductLife(damage);

}

int Terrain::getMonsterLife(int index, int monsterNum)

{

if (monsterNum == 1)

return X6\_1[index].getLife();

else if (monsterNum == 2)

return X6\_2[index].getLife();

else if (monsterNum == 3)

return X4\_1[index].getLife();

else if (monsterNum == 4)

return boss.getLife();

return 0;

}

}