封面

目錄

目錄

[一、 簡介 1](#_Toc10192963)

[1. 動機: 1](#_Toc10192964)

[2. 分工: 1](#_Toc10192965)

[二、 遊戲介紹 2](#_Toc10192966)

[1、 遊戲說明 2](#_Toc10192967)

[2、 遊戲圖形 2](#_Toc10192968)

[3、 遊戲音效 3](#_Toc10192969)

[三、 程式設計 4](#_Toc10192970)

[1、 程式架構 4](#_Toc10192971)

[2、 程式類別 4](#_Toc10192972)

[3、 程式技術 6](#_Toc10192973)

[四、 結語 10](#_Toc10192974)

[1、 問題及解決方法 10](#_Toc10192975)

[2、 時間表 11](#_Toc10192976)

[3、 貢獻比例 12](#_Toc10192977)

[4、 自我檢核表 12](#_Toc10192978)

[5、 收穫 13](#_Toc10192979)

[6、 心得、感想 13](#_Toc10192980)

[7、 對於本課程的建議 14](#_Toc10192981)

[附錄 15](#_Toc10192982)

1. 簡介
2. 動機:

我們選的遊戲是Dead Cell「死亡細胞」，會選這個遊戲的原因是因為他是最近一年相當紅的2D橫向卷軸遊戲，但它獨特的地方是它具有最近很火紅的「魂系列」元素，是以高難度和輪迴的為特色的一種遊戲方式，流暢的角色操作和多元的裝備組合受到玩家的好評，我們認為是最近具代表的遊戲之一。

1. 分工:

廖彥澤:

怪物(惡魔)、裝備(弓、劍)、特效、地圖設計、UI、傳送門、終點、門、寶箱。

郭家佑:

主角、Boss、遊戲系統、地圖系統、顯示文字、按鈕、整合、障礙排除。

1. 遊戲介紹
   1. 遊戲說明
      1. 遊戲目標

在第一關打敗怪物獲取武器後，進入第二關打敗BOSS。

* + 1. 操作說明:

方向鍵:移動

Z:翻滾

X:攻擊

C:互動(開寶箱、撿東西、使用傳送門等)

空白鍵:跳躍

下+空白鍵:下跳

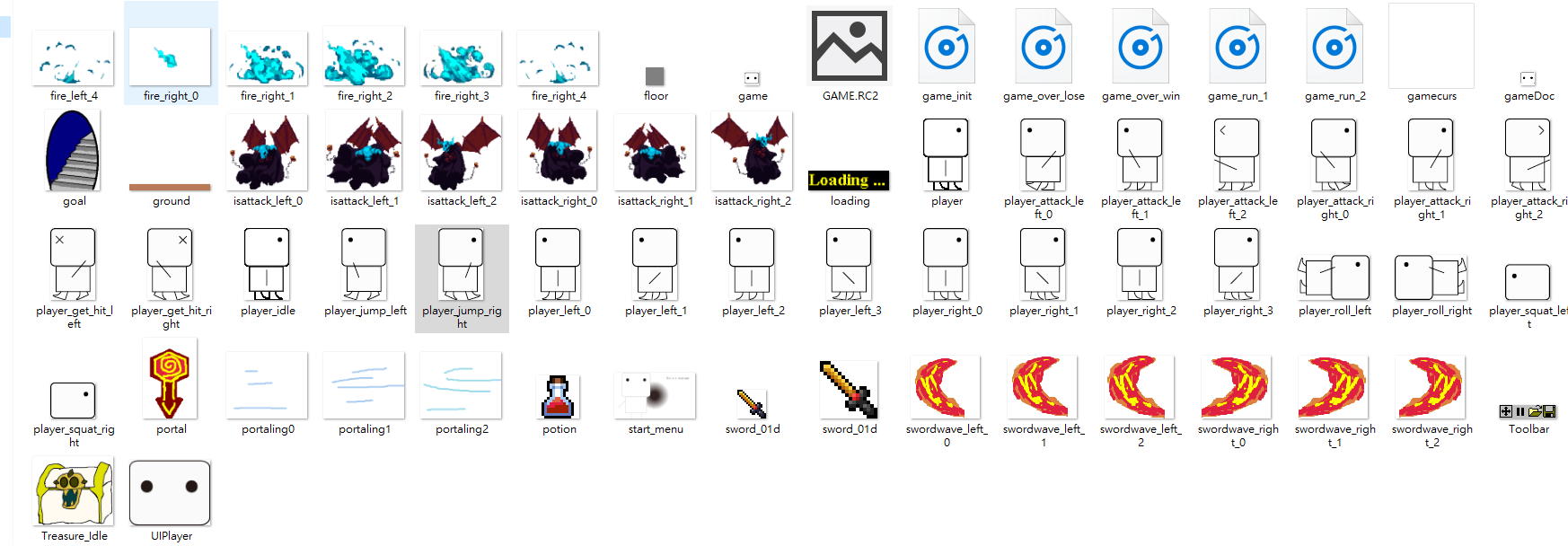
1:切換到武器1

2:切換到武器2

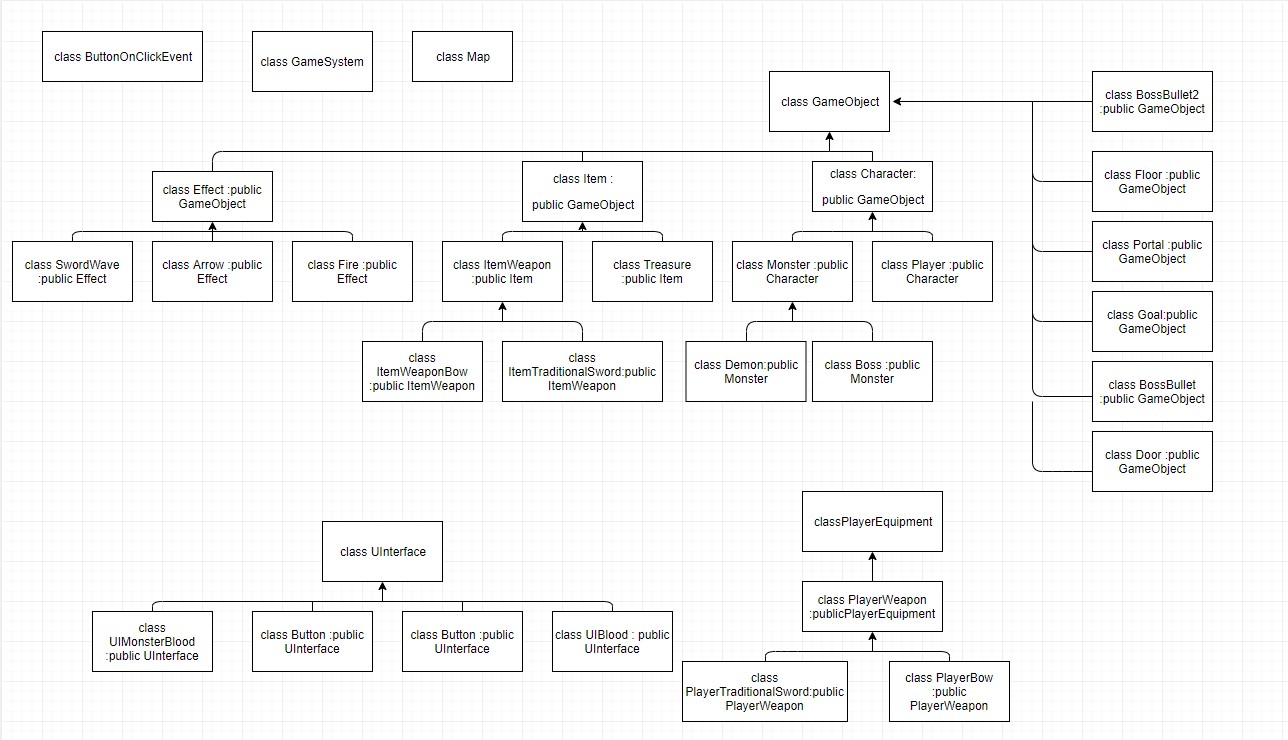
* + 1. 密技:

F:回血100

G:Boss扣血1000

* 1. 遊戲圖形
  2. 遊戲音效

|  |  |
| --- | --- |
| 代號 | 用處 |
| AUDIO\_GAME\_INIT | 開始畫面的音樂 |
| AUDIO\_GAME\_RUN\_1 | 第一關的音樂 |
| AUDIO\_GAME\_RUN\_2 | 第二關的音樂 |
| AUDIO\_GAME\_OVER\_LOSE | 遊戲失敗的音樂 |
| AUDIO\_GAME\_OVER\_WIN | 遊戲勝利的音樂 |

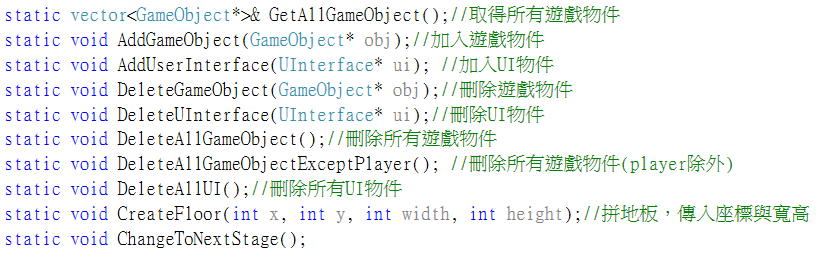
1. 程式設計
   1. 程式架構
   2. 程式類別

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| Arrow | 23 | 75 | 繼承Effect，弓箭射出的箭 |
| Boss | 88 | 377 | 繼承Monster，控制地圖上的Boss |
| BossBullet | 36 | 121 | 繼承GameObject，Boss發射的大子彈 |
| BossBullet2 | 31 | 80 | 繼承GameObject，Boss發射的小子彈 |
| Button | 25 | 49 | 繼承 UInterface，起始畫面的按鈕 |
| ButtonOnClickEvent | 33 | 64 | 當Button被點擊時，所觸發的事件 |
| Character | 77 | 298 | 繼承GameObject，具有角色基本的變數 |
| Demon | 24 | 221 | 繼承Monster，地圖上的小怪 |
| Door | 33 | 93 | 繼承GameObject，玩家擊破門時，怪物暈眩 |
| Effect | 38 | 154 | 繼承GameObject，武器、怪物的攻擊特效 |
| EquipedSlot | 18 | 49 | 繼承UInterface，顯示玩家目前的裝備 |
| Fire | 26 | 70 | 繼承Effcet，Demo攻擊得特效 |
| Floor | 19 | 41 | 繼承GameOject，地圖的地板和牆壁 |
| GameObject | 55 | 127 | 為地圖上所有物件的Base |
| GameSystem | 221 | 463 | 負責統一物件運行物件的動作(ShowAni..) |
| Goal | 11 | 21 | 繼承GameObject，到下一個地圖的門 |
| Item | 42 | 133 | 繼承GameObject，掉落物的Base |
| ItemTraditionalSword | 24 | 72 | 繼承ItemWeapon，只有圖片沒有武器功能(劍) |
| ItemWeapon | 27 | 30 | 繼承Item，具有武器素質的變數 |
| ItemWeaponBow | 14 | 67 | 繼承ItemWeapon，只有圖片沒有武器功能(弓) |
| Map | 39 | 110 | 紀錄地圖地形和大小，還有轉換座標的轉換 |
| Monster | 78 | 182 | 繼承Character，具有怪物的基本變數 |
| Player | 190 | 863 | 繼承Character，控制玩家 |
| PlayerBow | 21 | 48 | 繼承PlayerWeapon，有實質武器的功能(弓) |
| PlayerEquipment | 52 | 127 | 紀錄玩家身上的裝備 |
| PlayerTraditionalSword | 23 | 43 | 繼承PlayerWeapon，有實質武器的功能(劍) |
| PlayerWeapon | 18 | 15 | 繼承 PlayerEquipment，具有功能的Weapon |
| Portal | 22 | 56 | 繼承GameObject，地圖上兩點的傳送門 |
| Potion | 24 | 34 | 繼承Item，回血道具 |
| SwordWave | 19 | 65 | 繼承Effect，劍的攻擊特效 |
| Treasure | 17 | 57 | 繼承Item，地圖上的寶箱 |
| UIBlood | 19 | 59 | 繼承UIterface，顯示玩家血量 |
| UIMonsterBlood | 22 | 31 | 繼承UIterface，顯示怪物血量 |
| UInterface | 44 | 94 | UI的Base class |
| 總行數 | 1453 | 4261 |  |

* 1. 程式技術
     1. Static

遊戲中有系統與地圖兩個class，但整個遊戲中只會有一個系統與一個地圖，因此利用static來確保遊戲中使用到的是同一個系統與地圖。

* + - 1. GameSystem紀錄了所有遊戲中可供所有類別使用的函式或變數。例如:存放所有物件的gameObjectList、取得特定類別物件的函式GetGameObjectsWithType()等。
      2. Map設定了地圖大小，用以確保場景上的物件不會超出地圖邊界。還紀錄了所有地板的位置，可用來進行物件的碰撞判定。以及螢幕左上角的座標，用來換算螢幕上要看到的是地圖中的哪一部份。



▲使用static確保遊戲中使用到的都是同一個GameSystem

* + 1. Inheritance與polymorphism

我們寫了一個base class叫GameObject，裡面有x,y,width,height,layer等所有物件都會有的成員變數，以及SetX()、SetY()、ShowBitMap()等成員函式。場景中每個物件都會繼承GameObject，接著在mygame的OnShow()中，用for迴圈執行每個GameObject的ShowBitMap()，就可以用短短幾行來顯示全部的物件。

* + 1. Virtual destructor與memory leaks

有不少class有new pointer，沒有適當地delete將會造成memory leaks。由於我們每個物件都繼承了GameObject，因此可以透過virtual destructor來delete每個class中new出來的pointer。

* + 1. Layer

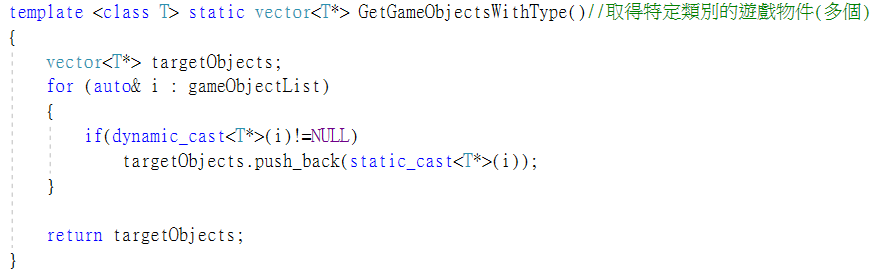
2D遊戲有圖層的問題，哪張圖要在哪張圖的前方是需要管理的。GameSystem中有一個vector叫gameObjectList，用來存放所有GameObject的指標。在gameObjectList插入新物件時，會根據每個GameObject的layer來決定該插入到哪個位置。圖層越高的物件會插入在越後面，這樣在OnShow()用for迴圈執行ShowBitMap()時就會比較晚顯示，就能顯示在越前方。

* + 1. Template與Dynamic\_cast
       1. Template

同一個函式可以利用template丟入不同型態的參數，省去重複撰寫的麻煩。GameSystem中的GetGameObjectsWithType，就是利用了template的特性，可以取得特定類別的所有物件。例如要取得所有的「怪物」，可以這樣寫:GetGameObjectsWithType<Monster>()，即可取得Monster、Demon、Boss等繼承了Monster的子類別物件。

* + - 1. Dynamic\_cast

在GetGameObjectsWithType中，利用了dynamic\_cast轉型失敗會回傳NullPtr的特性，可以找到特定類別的物件，以及繼承了這個類別的子類別的物件。用iterator對gameObjectList做迭代，接著做動態轉型dynamic\_cast<T\*>(iter)，若這個物件不是T的子類別，則不會回傳，反之則回傳。



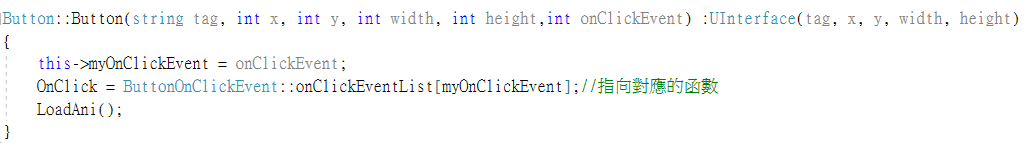
▲使用template與dynamic\_cast來取得特定類別的遊戲物件

* + 1. Enum

每個物件可能有多種動畫，例如左走、右走、上跳、攻擊等，因此將每個動畫存入vector，並透過enum來記錄每個動畫的編號，來決定該播放哪個動畫。

* + 1. Button、ButtonOnClickEvent與Function Pointer

每個Button都有x,y,width,height等基本的成員變數，但每種按鈕都有各自的OnClickEvent，所以除了Button外，還寫了一個class:ButtonOnClickEvent。ButtonOnClickEvent定義了每種按鈕的按下時的事件，而Button中定義了一個函式指標OnClick，會指向ButtonOnClickEvent中對應的事件。這樣子，Button的class就會很簡單，只需要x,y,width,height,以及OnClick，就能做出不同功能的按鈕。



▲Button只要在constructor指向對應的OnClick事件即可做出不同的效果。

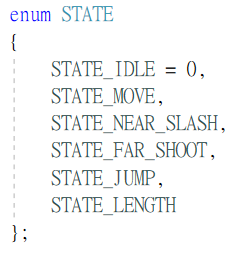
* + 1. Random

遊戲中有許多地方用到亂數，例如怪物掉落的裝備數值、每項道具的掉落機率等。亂數用到了C++的<Random>與<Functional>函式庫，利用這些函式庫就不用自己設定seed，減少了出錯的可能，而且產生的亂數真的很「亂」。

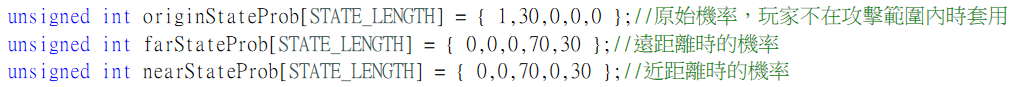
* + 1. Probability

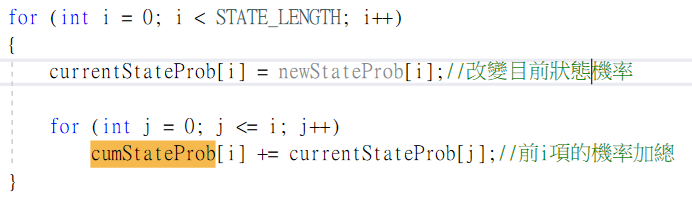
Boss的各項行動是隨機的，且每個動作的機率是不同的，這些機率又會因為與玩家之間的距離而產生變化。

首先定義Boss有幾種動作，再來用一個陣列紀錄每種動作各別的機率。接著再用一個陣列，將前i項的機率加總。最後使用Random產生0~機率總和之間的亂數，看這個亂數落在哪個區間，則執行哪個動作。

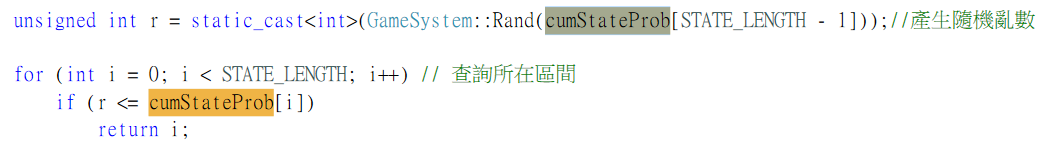


▲定義每種狀態

 ▲為每種狀態設定不同的機率，數字越大機率越高



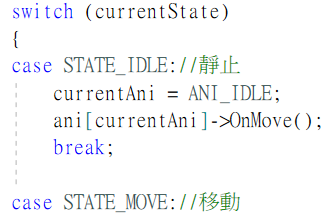
▲將前i項機率加總



▲查詢所在區間

* + 1. State Machine

狀態機能使設計變得簡單，Boss的各項動作便是使用狀態機。首先用enum定義各項狀態，接著再用另一個int變數currentState來記錄目前在哪個狀態，最後使用switch(currentState)來決定要執行哪個狀態。



▲透過switch(currentState)來執行各種狀態

1. 結語
   1. 問題及解決方法
      1. Template在.cpp會有無法解析的外部符號

將template寫到.h即可。

* + 1. 實體轉型會造成錯誤

將變數型態改為指標，即可轉型成功。

* + 1. 一個物件有很多種動畫，該怎麼儲存?

用Vector存動畫，用enum當編號。

* + 1. 碰撞偵測怎麼做?

用for迴圈判斷每一個x,y座標，有沒有在另一個物件的範圍內。

* + 1. 撿起裝備的方法?

裝備撿起後，裝備應該要「屬於」玩家，而不是「一件裝備」與「一個玩家」顯示在同一個座標。因此對地上的裝備道具按下互動鍵後，利用operator=，在玩家身上新增一個數值一模一樣的玩家裝備，並將地上的裝備道具刪除，即可讓裝備屬於玩家。

* + 1. 該怎麼設計按鈕的OnClick事件?

最初的按鈕的OnClick是設計成Virtual Function，想要有不同的OnClick就新增一個class繼承按鈕。但後來覺得，每個Button只有OnClick會不一樣，要因此而多寫一個class顯得有些多餘。最後決定將OnClick改為函式指標，然後再另外設計一個class ButtonOnClickEvent，讓函式指標指向其中的一個Function，即可只用兩個class做出不同的OnClick事件。

* 1. 時間表

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 郭家佑 | 廖彥澤 | 說明 |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 8 |  |  |  |
| 9 |  |  |  |
| 10 |  |  |  |
| 11 |  |  |  |
| 12 |  |  |  |
| 13 |  |  |  |
| 14 |  |  |  |
| 15 |  |  |  |
| 16 |  |  |  |
| 17 |  |  |  |
| 合計 | XXX | XXX |  |

* 1. 貢獻比例

郭家佑50%

廖彥澤50%

* 1. 自我檢核表

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

* 1. 收穫
     1. 106590025 資工二 郭家佑

學到了template、dynamic\_cast、static、function pointer、random等，以前比較沒機會用到的語法。也更加熟悉polymorphism、inheritance等物件導向的精髓功能。

106590018 資工二　廖彥澤

學到如何與別人一起完成一個project，還有物件導向的概念，例如：

繼承、多型……等等，必須將一個物件包裝好，讓隊友能簡單使

用，才能保持project的架構和效率。

，

* 1. 心得、感想

106590025 資工二 郭家佑

透過這堂課，學會了多型與繼承的用法，實際地應用了物件導向的觀念與語法。也明白了如何解決memory leak，new完後要記得delete，若是在class的member variable做了new，那要記得在destructor進行delete。

由於是分組作業，因此還學會了github的用法，如何push、pull與merge。

也明白了與組員溝通協調的重要。如果自己負責的部分與組員衝突到，該怎麼辦?base class有功能可以新增，該由誰來更新?誰要負責寫哪個部分?不論如何，最重要的還是要互相溝通，不然最後兩個人的程式大雜燴，會使得程式不易維護。

106590018 資工二　廖彥澤

經過了這堂課的學習，在專案讓用到了許多繼承、多形……等等物件導向的觀念，除了在技術上的使用，在寫程式了邏輯也變得非常注重，

若程式不夠簡潔，就會寫出許多的垃圾code，在易讀性會大幅降低，

在維護上也會更加困難；在與隊友的合作、討論也相當重要，在大家初次練習的情況下，在遊戲初期，我們採取分開分工的方式，各自負責較不相關的class，讓遊戲快速有個架構，也比較不會混亂，在後期的中後期的時候再開始討論class之間該如何運作，整體來說算是相當順利。

* 1. 對於本課程的建議

Framework Bug很多，又很舊，請好好維護與更新。

附錄

Arrow.h

#pragma once

#include "Effect.h"

class Arrow :public Effect

{

public:

Arrow(string tag, int x, int y, int width, int height);

Arrow::Arrow(string tag, int x, int y, int width, int height, int attackRange, int direction);

~Arrow();

void ShowBitMap()override; //顯示動畫

private:

void LoadAni()override;

void Fall(); //讓箭有落下的效果

int initX, initY; //箭初始的位置

int dX, dY; //每次下降多往前多少

int direction; //方向

enum ANI {

ANI\_ARROW\_IDLE = 0,

ANI\_ARROW\_LEFT,

ANI\_ARROW\_RIGHT

};

};

Arrow.cpp

#include"StdAfx.h"

#include"Arrow.h"

Arrow::Arrow(string tag, int x, int y, int width, int height) :Effect(tag, x, y, width, height) {

tag = "Arrow";

LoadAni();

SetBitMapPosition();

currentAni = ANI\_ARROW\_IDLE;

this->direction = ANI\_ATTACK\_LEFT;

LoadBitMap(".\\res\\arrow.bmp");

initX = x + 300;

initY = y + 20;

dX = 25;

dY = 0;

}

Arrow::Arrow(string tag, int x, int y, int width, int height, int attackRange, int direction) :Effect(tag, x, y, width, height) {

tag = "Arrow";

LoadAni();

SetBitMapPosition();

currentAni = ANI\_ARROW\_IDLE;

LoadBitMap(".\\res\\arrow.bmp");

this->initX = x;

this->initY = y;

this->direction = direction;

this->dX = 25;

this->dY = 0;

}

Arrow::~Arrow()

{

}

void Arrow::ShowBitMap() {

Fall();

if (direction == ANI\_ATTACK\_LEFT&& CanMoveLeft(dX)&& CanMoveDown(dY) && isHit == false)

{

currentAni = ANI\_ARROW\_LEFT;

x = x - dX;

y = y + dY;

SetBitMapPosition();

ani[currentAni]->OnShow();

EffectAttackMonster(player->GetAttackDamage());

}

else if (direction == ANI\_ATTACK\_RIGHT && CanMoveRight(dX) && CanMoveDown(dY) && isHit == false)

{

currentAni = ANI\_ARROW\_RIGHT;

x = x + dX;

y = y + dY;

SetBitMapPosition();

ani[currentAni]->OnShow();

EffectAttackMonster(player->GetAttackDamage());

}

else if((CanMoveRight(dX) || CanMoveDown(dY))){

Dead();

}

}

void Arrow::LoadAni() {

char\* aniArrow\_idle[1] = { ".\\res\\arrow.bmp" };

AddAniBitMaps(aniArrow\_idle, ANI\_ARROW\_IDLE, 1);

char\* aniArrow\_left[3] = { ".\\res\\arrow\_left.bmp", ".\\res\\arrow\_left.bmp", ".\\res\\arrow\_left.bmp" };

AddAniBitMaps(aniArrow\_left, ANI\_ARROW\_LEFT, 3);

char\* aniArrow\_right[3] = { ".\\res\\arrow\_right.bmp", ".\\res\\arrow\_right.bmp",".\\res\\arrow\_right.bmp" };

AddAniBitMaps(aniArrow\_right, ANI\_ARROW\_RIGHT, 3);

}

void Arrow :: Fall()

{

if (x - initX >= 250) //如果箭已經飛了一段距離(250)就開始下降

dY = dY + 1;

}

Boss.h

#pragma once

#include "Character.h"

#include "Monster.h"

//Boss

//王

class Boss :public Monster

{

public:

Boss();

~Boss();

Boss(string tag, int x, int y, int width, int height);

void Act();//行動

int GetPhase();//目前階段

private:

void Dead()override;//死亡

void Jump();//跳躍

void Fall(int perDisplacement);//掉落

void Move(int dx, int dy)override;

bool isShoot = false;//是否發射了子彈

int originJumpDisplacement = 30, jumpDisplacement = originJumpDisplacement;//跳躍位移量

void ShowBitMap()override;

void NearSlash();//砍

void JumpBack();//往後跳

void JumpFront();//往前跳

const int JUMP\_FRONT = 0, JUMP\_BACK = 1;

int jumpMode;//跳躍模式

void FarShoot();//遠程攻擊

void InstantDeath();//即死技能

int RandomState();//決定隨機狀態

int currentState = 0;//目前狀態

int nearAttackRange = width;//近攻擊距離

int farAttackRange = SIZE\_X \* 2 / 3;//遠攻擊距離

int playerDistanceX, playerDistanceY;//玩家跟BOSS的距離

enum STATE

{

STATE\_IDLE = 0,

STATE\_MOVE,

STATE\_NEAR\_SLASH,

STATE\_FAR\_SHOOT,

STATE\_JUMP,

STATE\_LENGTH

};

unsigned int currentStateProb[STATE\_LENGTH];//目前各種狀態的機率，會根據玩家的距離、BOSS的血量等因素而變化

unsigned int cumStateProb[STATE\_LENGTH] = { 0 };//各項狀態的累計機率，用來判別亂數用的

void ChangeStateProb(unsigned int newStateProb[]);//改變各項狀態機率

//各種狀態的機率，數字越大機率越高

unsigned int originStateProb[STATE\_LENGTH] = { 1,30,0,0,0 };//原始機率，玩家不在攻擊範圍內時套用

unsigned int farStateProb[STATE\_LENGTH] = { 0,0,0,70,30 };//遠距離時的機率

unsigned int nearStateProb[STATE\_LENGTH] = { 0,0,70,0,30 };//近距離時的機率

bool InNear();//在近距攻擊範圍內

bool InFar();//在遠距攻擊範圍內

const bool FACE\_LEFT = true;//面向左邊

const bool FACE\_RIGHT = false;//面向右邊

bool faceLR = FACE\_LEFT;//面向左邊或面向右邊

int currentAni = 0;//目前動畫

enum ANI

{

ANI\_IDLE = 0,

ANI\_LEFT,

ANI\_RIGHT,

ANI\_NEAR\_SLASH\_LEFT,

ANI\_NEAR\_SLASH\_RIGHT,

ANI\_JUMP,

ANI\_FAR\_SHOOT\_LEFT,

ANI\_FAR\_SHOOT\_RIGHT,

ANI\_LENGTH

};

int phase = 1;//階段

void LoadAni()override;

};

Boss.cpp

#include "StdAfx.h"

#include "Boss.h"

#include "BossBullet.h"

#include "Potion.h"

#include <math.h>

Boss::Boss()

{

}

Boss::~Boss()

{

}

Boss::Boss(string tag, int x, int y, int width, int height) :Monster(tag, x, y, width, height)

{

tag = "Monster";

maxHP = 10000;

HP = maxHP;

aniDelay = 10;

moveSpeed = 3;

attackRange = 100;

attackDamage = 50;

isShoot = false;

LoadAni();

for (int i = 0; i < STATE\_LENGTH; i++)//建立各項機率表

{

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

cumStateProb[i] += originStateProb[j];

}

phase = 1;//第一階段

currentState = STATE\_IDLE;//預設動作

}

void Boss::Act()

{

if (phase <= 1 && static\_cast<float>(HP) / maxHP <= 0.5)//血量小於50 %，第二階段

{

phase = 2;

GameSystem::AddGameObject(new Potion("Potion", x, y, 48, 48));//生成一個藥水

for (int i = 0; i < ANI\_LENGTH; i++)//動畫加速

ani[i]->SetDelayCount(aniDelay \* 2 / 3);

}

else if (phase <= 2 && static\_cast<float>(HP) / maxHP <= 0.2)//血量小於20%，第三階段

{

phase = 3;

GameSystem::AddGameObject(new Potion("Potion", x, y, 48, 48));//生成一個藥水

for (int i = 0; i < ANI\_LENGTH; i++)//動畫加速

ani[i]->SetDelayCount(aniDelay / 2);

}

//currentState = STATE\_FAR\_SHOOT;

switch (currentState)//根據狀態做不同動作

{

case STATE\_IDLE://靜止

if (player->GetX() + player->GetWidth() / 2 - (x + width / 2) < 0)//玩家在左邊

faceLR = FACE\_LEFT;//面向左邊

else

faceLR = FACE\_RIGHT;//面相右邊

//faceLR = static\_cast<int>(GameSystem::Rand(2));

//if (ani[currentAni]->IsEnd())//播完動畫後

{

currentState = RandomState();//隨機改變狀態

if (currentState == STATE\_JUMP)//下一個狀態是跳躍時，紀錄玩家跟BOSS的距離

{

playerDistanceX = player->GetX() - x;

playerDistanceY = player->GetY() - y;

}

}

break;

case STATE\_MOVE://移動

if (faceLR == FACE\_LEFT)

{

Move(-moveSpeed, 0);

}

else

{

Move(moveSpeed, 0);

}

if (ani[currentAni]->IsEnd())//移動動畫播完

{

currentState = STATE\_IDLE;//就回到靜止狀態

}

break;

case STATE\_NEAR\_SLASH://攻擊

NearSlash();

break;

case STATE\_FAR\_SHOOT://射擊

FarShoot();

break;

case STATE\_JUMP:

Jump();

break;

}

Fall(fallDisplacement);

}

int Boss::GetPhase()

{

return phase;

}

void Boss::Fall(int perDisplacement)

{

if (CanMoveDown(fallDisplacement))//如果腳下沒東西

{

fallDisplacement++;

Move(0, fallDisplacement);

}

else

{

while (CanMoveDown(1))//再繼續用下降位移量下降，將會卡進地板，所以一次向下位移1進行微調0

Move(0, 1);

fallDisplacement = 0;

}

}

void Boss::NearSlash()

{

if (ani[currentAni]->IsEnd())//攻擊判定只會出現一次，因此攻擊動畫播完才攻擊

{

int leftEdge = 0, rightEdge = 0;

if (faceLR == FACE\_LEFT)//設定向左邊界

{

leftEdge = this->x - attackRange;

rightEdge = this->x + this->width / 2;

}

else//設定向右邊界

{

leftEdge = this->x + this->width / 2;

rightEdge = this->x + this->width + attackRange;

}

if (player->GetX() + player->GetWidth() / 2 > leftEdge

&& player->GetX() + player->GetWidth() / 2 < rightEdge

&& player->GetY() + player->GetHeight() > this->y

&& player->GetY() < this->y + this->height)//玩家在範圍內

{

player->DecreaseHP(attackDamage);//攻擊玩家

}

currentState = STATE\_IDLE;//回到靜止狀態

}

}

void Boss::FarShoot()

{

if (ani[currentAni]->IsEnd())

{

isShoot = false;

currentState = STATE\_IDLE;//回到靜止狀態

}

else if (ani[currentAni]->GetCurrentBitmapNumber() == 3)//第三張圖片時發射子彈

{

if (isShoot == false)

{

isShoot = true;

GameSystem::AddGameObject(new BossBullet("BossBullet", x, y, 50, 50));

}

}

}

void Boss::Dead()

{

GameSystem::SetGameOver(GameSystem::OVER\_WIN);//遊戲結束

GameSystem::DeleteGameObject(this);

}

void Boss::Jump()

{

if (jumpDisplacement-- > 0)//跳躍位移量隨時間遞減

{

if (jumpMode == JUMP\_BACK)//在近處

JumpBack();//向後跳

else if (jumpMode == JUMP\_FRONT)//在遠方

JumpFront();//向前跳

}

else

{

jumpDisplacement = originJumpDisplacement;//跳躍位移量還原

if (player->GetX() + player->GetWidth() > this->x - attackRange

&& player->GetX() < this->x + this->width + attackRange

&& player->GetY() + player->GetHeight() > this->y + this->height / 2

&& player->GetY() < this->y + this->height)//玩家在範圍內

{

if (player->GetIsJump() == false)//玩家沒有在跳躍

player->DecreaseHP(attackDamage);//攻擊玩家

}

currentState = STATE\_IDLE;//回到靜止狀態

}

ani[currentAni]->OnMove();

}

void Boss::Move(int dx, int dy)

{

if (x + dx >= 0 && x + dx < Map::WORLD\_SIZE\_X)//在地圖範圍內

this->x += dx;

if (y + dy >= 0 && y + dy < Map::WORLD\_SIZE\_Y)//在地圖範圍內

this->y += dy;

}

void Boss::JumpBack()

{

int dx = 0;

if(playerDistanceX != 0)//確保分母不為0

dx = static\_cast<int>(-5 \* moveSpeed \* playerDistanceX / (abs(playerDistanceX)));//向玩家反方向跳躍

Move(dx, -jumpDisplacement);//向上位移，並向玩家方向跳躍

}

void Boss::JumpFront()

{

int dx = static\_cast<int>(5 \* moveSpeed \* playerDistanceX / sqrt(pow(playerDistanceX, 2) + pow(playerDistanceY, 2)));//向玩家跳躍

Move(dx, -jumpDisplacement);//向上位移，並向玩家方向跳躍

}

void Boss::InstantDeath()

{

}

int Boss::RandomState()

{

if (InNear())//近距離

{

ChangeStateProb(nearStateProb);//變為近距攻擊狀態

jumpMode = JUMP\_BACK;//向後跳

}

else if (InFar())//遠距離

{

ChangeStateProb(farStateProb);//變為遠距攻擊狀態

jumpMode = JUMP\_FRONT;

}

else//超過遠距離攻擊範圍，BOSS不再攻擊玩家

ChangeStateProb(originStateProb);//原始狀態

unsigned int r = static\_cast<int>(GameSystem::Rand(cumStateProb[STATE\_LENGTH - 1]));//產生隨機亂數

for (int i = 0; i < STATE\_LENGTH; i++) // 查詢所在區間

if (r <= cumStateProb[i])

return i;

return 0;

}

void Boss::ChangeStateProb(unsigned int newStateProb[])//改變各項狀態機率

{

for (int i = 0; i < STATE\_LENGTH; i++)//重置累計機率為0

cumStateProb[i] = 0;

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

{

currentStateProb[i] = newStateProb[i];//改變目前狀態機率

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

cumStateProb[i] += currentStateProb[j];//前i項的機率加總

}

}

bool Boss::InNear()

{

if (sqrt(pow(player->GetX() + player->GetWidth() / 2 - (x + width / 2), 2) + pow(player->GetY() + player->GetHeight() / 2 - (y + height / 2), 2)) < nearAttackRange)//近距離

return true;

else

return false;

}

bool Boss::InFar()

{

if (sqrt(pow(player->GetX() + player->GetWidth() / 2 - (x + width / 2), 2) + pow(player->GetY() + player->GetHeight() / 2 - (y + height / 2), 2)) < farAttackRange)//遠距離

return true;

else

return false;

}

void Boss::ShowBitMap()

{

switch (currentState)

{

case STATE\_IDLE://靜止

currentAni = ANI\_IDLE;

ani[currentAni]->OnMove();

break;

case STATE\_MOVE://移動

if (faceLR == FACE\_LEFT)

{

currentAni = ANI\_LEFT;

}

else

{

currentAni = ANI\_RIGHT;

}

ani[currentAni]->OnMove();

break;

case STATE\_NEAR\_SLASH://攻擊

if (faceLR == FACE\_LEFT)

{

currentAni = ANI\_ATTACK\_LEFT;

}

else

{

currentAni = ANI\_ATTACK\_RIGHT;

}

ani[currentAni]->OnMove();

break;

case STATE\_FAR\_SHOOT:

if (faceLR == FACE\_LEFT)

{

currentAni = ANI\_FAR\_SHOOT\_LEFT;

}

else

{

currentAni = ANI\_FAR\_SHOOT\_RIGHT;

}

ani[currentAni]->OnMove();

break;

case STATE\_JUMP://跳躍

currentAni = ANI\_JUMP;

break;

}

GameSystem::ShowText(to\_string(GetHP()), x - Map::GetSX(), y - Map::GetSY() - 30, x + width - Map::GetSX(), y + height - Map::GetSY(), GameSystem::ALIGN\_CENTER, GameSystem::ALIGN\_TOP, 16, RGB(0, 0, 0));

ani[currentAni]->OnShow();

}

void Boss::LoadAni()

{

char\* aniIdle[4] = { ".\\res\\boss\_idle\_0.bmp", ".\\res\\boss\_idle\_1.bmp", ".\\res\\boss\_idle\_2.bmp", ".\\res\\boss\_idle\_3.bmp" };

AddAniBitMaps(aniIdle, ANI::ANI\_IDLE, 4, aniDelay);

char\* aniLeft[4] = { ".\\res\\boss\_left\_0.bmp", ".\\res\\boss\_left\_1.bmp", ".\\res\\boss\_left\_2.bmp", ".\\res\\boss\_left\_3.bmp" };

AddAniBitMaps(aniLeft, ANI\_LEFT, 4, aniDelay);

char\* aniRight[4] = { ".\\res\\boss\_right\_0.bmp", ".\\res\\boss\_right\_1.bmp", ".\\res\\boss\_right\_2.bmp", ".\\res\\boss\_right\_3.bmp" };

AddAniBitMaps(aniRight, ANI\_RIGHT, 4, aniDelay);

char\* aniNearSlashLeft[3] = { ".\\res\\boss\_near\_slash\_left\_0.bmp", ".\\res\\boss\_near\_slash\_left\_1.bmp", ".\\res\\boss\_near\_slash\_left\_2.bmp" };

AddAniBitMaps(aniNearSlashLeft, ANI\_NEAR\_SLASH\_LEFT, 3, aniDelay);

char\* aniNearSlashRight[3] = { ".\\res\\boss\_near\_slash\_right\_0.bmp", ".\\res\\boss\_near\_slash\_right\_1.bmp", ".\\res\\boss\_near\_slash\_right\_2.bmp" };

AddAniBitMaps(aniNearSlashRight, ANI\_NEAR\_SLASH\_RIGHT, 3, aniDelay);

char\* aniJump[3] = { ".\\res\\boss\_jump\_0.bmp", ".\\res\\boss\_jump\_1.bmp", ".\\res\\boss\_jump\_2.bmp" };

AddAniBitMaps(aniJump, ANI\_JUMP, 3, aniDelay);

char\* aniFarShootLeft = ".\\res\\boss\_far\_shoot\_left";

AddAniBitMaps(aniFarShootLeft, ANI\_FAR\_SHOOT\_LEFT, 7, aniDelay);

char\* aniFarShootRight = ".\\res\\boss\_far\_shoot\_right";

AddAniBitMaps(aniFarShootRight, ANI\_FAR\_SHOOT\_RIGHT, 7, aniDelay);

}

BossBullet.h

#pragma once

#include "GameObject.h"

#include "Player.h"

#include "Boss.h"

//BossBullet

//Boss發射的子彈

class BossBullet :public GameObject

{

public:

BossBullet();

BossBullet(string tag, int x, int y, int width, int height);

~BossBullet();

void Dead()override;

void Act()override;

void SetBitMapPosition()override;

void ShowBitMap()override;

private:

Player \* player = GameSystem::GetGameObjectWithType<Player>();//玩家

int originX, originY;//原始座標

int targetX, targetY;//目標座標

int distanceX, distanceY;//與目標的距離

int dx, dy;//x,y位移量

int moveSpeed;//移動速度

int attackDamage;//攻擊傷害

Boss\* boss = GameSystem::GetGameObjectWithType<Boss>();//boss

CAnimation \* ani;

void LoadAni();

};

BossBullet.cpp

#include "StdAfx.h"

#include "BossBullet.h"

#include "GameSystem.h"

#include "BossBullet2.h"

#include <math.h>

BossBullet::BossBullet()

{

}

BossBullet::BossBullet(string tag, int x, int y, int width, int height) :GameObject(tag, x, y, width, height)

{

layer = GameSystem::LAYER\_MONSTER;

ani = new CAnimation(5);

LoadAni();

moveSpeed = 10;

attackDamage = 50;

Player\* player = GameSystem::GetGameObjectWithType<Player>();

originX = x;

originY = y;

targetX = player->GetX() + player->GetWidth() / 2;

targetY = player->GetY() + player->GetHeight() / 2;

distanceX = (targetX - (x + width / 2));

distanceY = (targetY - (y + height / 2));

int r = static\_cast<int>(sqrt(pow(distanceX, 2) + pow(distanceY, 2)));//x,y畫成直角三角形後的斜邊

dx = static\_cast<int>(static\_cast<float>(distanceX) / r \* moveSpeed);

dy = static\_cast<int>(static\_cast<float>(distanceY) / r \* moveSpeed);

}

BossBullet::~BossBullet()

{

delete ani;

}

void BossBullet::Dead()

{

player->DecreaseHP(attackDamage);

GameObject::Dead();

}

void BossBullet::Act()

{

x += dx;//移動

y += dy;//移動

if (x + width < 0 || x >= Map::WORLD\_SIZE\_X || y + height < 0 || y >= Map::WORLD\_SIZE\_Y)//飛出邊界

GameObject::Dead();//直接刪除

else if (x + width > player->GetX() && x < player->GetX() + player->GetWidth()

&& y + height > player->GetY() && y < player->GetY() + player->GetHeight())//打中玩家

{

Dead();//先攻擊玩家再刪除

}

else if (boss->GetPhase() == 2)//Boss目前在第二階段，有散射功能

{

if (sqrt(pow(x - originX, 2) + pow(y - originY, 2)) >= sqrt(pow(SIZE\_X, 2) + pow(SIZE\_Y, 2)) / 2)

//或者移動的距離超過螢幕斜邊的1/2，就散射子彈

{

//製造八顆子彈

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

{

GameSystem::AddGameObject(new BossBullet2("BossBullet", x, y, width, height, 30 + 45 \* i));

}

GameObject::Dead();

}

}

else if (boss->GetPhase() == 3)//Boss目前在第三階段，散射所需距離變短

{

if (sqrt(pow(x - originX, 2) + pow(y - originY, 2)) >= sqrt(pow(SIZE\_X, 2) + pow(SIZE\_Y, 2)) \* 5 / 12)

//或者移動的距離超過螢幕斜邊的5/12，就散射子彈

{

//製造八顆子彈

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

{

GameSystem::AddGameObject(new BossBullet2("BossBullet", x, y, width, height, 30 + 45 \* i));

}

GameObject::Dead();

}

}

}

void BossBullet::SetBitMapPosition()

{

ani->SetTopLeft(x - Map::GetSX(), y - Map::GetSY());

}

void BossBullet::ShowBitMap()

{

ani->OnMove();

ani->OnShow();

}

void BossBullet::LoadAni()

{

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

{

//將檔名串接底線+編號+副檔名

string sPic = "";

if (boss->GetPhase() == 1)//第一階段的圖

sPic = ".\\res\\boss\_bullet";

else if (boss->GetPhase() == 2)//第二階段的圖

sPic = ".\\res\\boss\_bullet2";

else if (boss->GetPhase() == 3)//第三階段的圖(跟二階一樣)

sPic = ".\\res\\boss\_bullet2";

sPic += "\_" + to\_string(i) + ".bmp";

//轉成c-string

char\* cPic = new char[sPic.length() + 1];

strcpy(cPic, sPic.c\_str());

ani->AddBitmap(cPic, RGB(255, 255, 255));//加入圖片

delete cPic;//delete new

}

}

BossBullet2.h

#pragma once

#include "GameObject.h"

#include "Player.h"

//BossBullet2

//Boss的子彈散射出來的子彈

class BossBullet2 :public GameObject

{

public:

BossBullet2();

BossBullet2(string tag, int x, int y, int width, int height, double angle);//飛向的角度

~BossBullet2();

void Dead()override;

void Act()override;

void SetBitMapPosition()override;

void ShowBitMap()override;

private:

Player \* player = GameSystem::GetGameObjectWithType<Player>();//玩家

double normalX, normalY;//x,y單位位移量

int moveSpeed;//移動速度

int attackDamage;//攻擊傷害

CAnimation \* ani;

void LoadAni();

};

BossBullet2.cpp

#include "StdAfx.h"

#include "BossBullet2.h"

#include "GameSystem.h"

#include <math.h>

#define M\_PI 3.14159265358979323846264338327950288

BossBullet2::BossBullet2()

{

}

BossBullet2::BossBullet2(string tag, int x, int y, int width, int height,double angle) :GameObject(tag, x, y, width, height)

{

layer = GameSystem::LAYER\_MONSTER;

ani = new CAnimation(5);

LoadAni();

moveSpeed = 10;

attackDamage = 10;

Player\* player = GameSystem::GetGameObjectWithType<Player>();

normalX = cos(angle \* M\_PI / 180.0);

normalY = sin(angle \* M\_PI / 180.0);

}

BossBullet2::~BossBullet2()

{

delete ani;

}

void BossBullet2::Dead()

{

player->DecreaseHP(attackDamage);

GameObject::Dead();

}

void BossBullet2::Act()

{

x += static\_cast<int>(normalX \* moveSpeed);

y += static\_cast<int>(normalY \* moveSpeed);

if (x + width < 0 || x >= Map::WORLD\_SIZE\_X || y + height < 0 || y >= Map::WORLD\_SIZE\_Y)//飛出邊界

GameObject::Dead();//直接刪除

else if (x + width > player->GetX() && x < player->GetX() + player->GetWidth()

&& y + height > player->GetY() && y < player->GetY() + player->GetHeight())//打中玩家

{

Dead();//先攻擊玩家再刪除

}

}

void BossBullet2::SetBitMapPosition()

{

ani->SetTopLeft(x - Map::GetSX(), y - Map::GetSY());

}

void BossBullet2::ShowBitMap()

{

ani->OnMove();

ani->OnShow();

}

void BossBullet2::LoadAni()

{

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

{

//將檔名串接底線+編號+副檔名

string sPic = ".\\res\\boss\_bullet";

sPic += "\_" + to\_string(i) + ".bmp";

//轉成c-string

char\* cPic = new char[sPic.length() + 1];

strcpy(cPic, sPic.c\_str());

ani->AddBitmap(cPic, RGB(255, 255, 255));//加入圖片

delete cPic;//delete new

}

}

Button.h

#pragma once

#include "UInterface.h"

#include "ButtonOnClickEvent.h"

//Button

//按鈕

class Button :public UInterface

{

public:

Button();

Button(string tag, int x, int y, int width, int height, int onClickEvent);

~Button();

void Dead();

bool (\*OnClick)(CGame\* c);//按下的事件

void SetBitMapPosition();

void ShowBitMap();

private:

virtual void LoadAni();

int myOnClickEvent;//這顆按鈕的OnClick事件是哪一個

};

Button.cpp

#include "StdAfx.h"

#include "Button.h"

Button::Button()

{

}

Button::Button(string tag, int x, int y, int width, int height,int onClickEvent) :UInterface(tag, x, y, width, height)

{

this->myOnClickEvent = onClickEvent;

OnClick = ButtonOnClickEvent::onClickEventList[myOnClickEvent];//指向對應的函數

LoadAni();

}

Button::~Button()

{

}

void Button::Dead()

{

}

void Button::SetBitMapPosition()

{

UIpicture.SetTopLeft(x,y);

}

void Button::ShowBitMap()

{

UIpicture.ShowBitmap();

}

void Button::LoadAni()

{

switch (myOnClickEvent)

{

case ButtonOnClickEvent::ON\_CLICK\_START:

LoadBitMap(".\\res\\button\_start.bmp");

break;

case ButtonOnClickEvent::ON\_CLICK\_EXIT:

LoadBitMap(".\\res\\button\_exit.bmp");

break;

case ButtonOnClickEvent::ON\_CLICK\_OPTION:

LoadBitMap(".\\res\\button\_option.bmp");

break;

}

}

ButtonOnClickEvent.h

#pragma once

#include <Map>

#include "UInterface.h"

//ButtonOnClickEvent

//按鈕事件

class ButtonOnClickEvent

{

public:

ButtonOnClickEvent();

~ButtonOnClickEvent();

static bool OnClickStart(CGame\* c);//開始按鈕

static bool OnClickExit(CGame\* c);//結束按鈕

static bool OnClickOption(CGame\* c);//選項按鈕

static void LoadOptionAni();//載入動畫

enum OnClickEvent

{

ON\_CLICK\_START = 0,

ON\_CLICK\_EXIT,

ON\_CLICK\_OPTION,

ON\_CLICK\_LENGTH

};

static map<int, bool(\*)(CGame\*)> onClickEventList;//儲存按鈕事件的函數指標

private:

static CAnimation ani;

static bool isAniLoaded;

};

ButtonOnClickEvent.cpp

#include "StdAfx.h"

#include "ButtonOnClickEvent.h"

#include "GameSystem.h"

CAnimation ButtonOnClickEvent::ani(5);

bool ButtonOnClickEvent::isAniLoaded = false;

map<int, bool(\*)(CGame\*)> ButtonOnClickEvent::onClickEventList =

{

{ON\_CLICK\_START,&OnClickStart},

{ON\_CLICK\_EXIT,&OnClickExit},

{ON\_CLICK\_OPTION,&OnClickOption}

};

ButtonOnClickEvent::ButtonOnClickEvent()

{

}

ButtonOnClickEvent::~ButtonOnClickEvent()

{

}

bool ButtonOnClickEvent::OnClickStart(CGame\* c)

{

c->Instance()->SetGameState(GAME\_STATE\_RUN);//按下開始，切換到GAME\_STATE\_RUN

return true;

}

bool ButtonOnClickEvent::OnClickExit(CGame\* c)

{

//exit(0);

::PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, NULL, NULL);//賴神寫的

return true;

}

bool ButtonOnClickEvent::OnClickOption(CGame \* c)

{

GameSystem::DrawRectangle(0, 0, SIZE\_X, SIZE\_Y, RGB(240, 240, 240));

GameSystem::ShowText("遊戲說明:\n在第一關打敗怪物獲取武器後，進入第二關打敗BOSS吧!\n\n操作說明:\n方向鍵:移動\nZ:翻滾\nX:攻擊\nC:互動(開寶箱、撿東西、使用傳送門等)\n空白鍵:跳躍(最多三段跳)\n下+空白鍵:下跳\n1:切換到武器1\n2:切換到武器2\n\n密技:\nF:回血100\nG:Boss扣血1000\n\n(再次按下滑鼠左鍵回主選單)"

, SIZE\_X / 10, 0, SIZE\_X, SIZE\_Y, GameSystem::ALIGN\_LEFT, GameSystem::ALIGN\_CENTER

, 12);

ani.SetTopLeft(SIZE\_X \* 3 / 5, 200);

ani.OnMove();

ani.OnShow();

return true;

}

void ButtonOnClickEvent::LoadOptionAni()

{

if (isAniLoaded == false)

{

ani.AddBitmap(".\\res\\boss\_boss\_0.bmp", RGB(255, 255, 255));

ani.AddBitmap(".\\res\\boss\_boss\_1.bmp", RGB(255, 255, 255));

ani.AddBitmap(".\\res\\boss\_boss\_2.bmp", RGB(255, 255, 255));

ani.AddBitmap(".\\res\\boss\_boss\_3.bmp", RGB(255, 255, 255));

isAniLoaded = true;

}

}

Character.h

#pragma once

//Character

//角色

//繼承GameObject

#include <vector>

#include "GameObject.h"

class Character :public GameObject

{

public:

Character();

virtual~Character();

Character::Character(string tag, int x, int y, int width, int height, int pictureID);

Character::Character(string tag, int x, int y, int width, int height); //No PictureID 的 Contructor

void SetMaxHP(int maxHp);//設定最大血量

void SetHP(int hp);//設定目前血量

void SetOriginMoveSpeed(int originMoveSpeed);//設定原始移動速度

void SetMoveSpeed(int moveSpeed);//設定目前移動速度

void SetOriginAttackDamage(int originAttack);//設定原始攻擊力

void SetAttackDamage(int attack);//設定目前攻擊力

void SetOriginAttackSpeed(int originAttackSpeed);//設定原始攻擊速度

void SetAttackSpeed(int attackSpeed);//設定目前攻擊速度

void SetOriginAttackRange(int originAttackRange);//設定原始攻擊範圍

void SetAttackRange(int attackRange);//設定目前攻擊範圍

void SetDefense(int defense);

void SetIsDead(bool isDead);//設定是否死亡

int GetMaxHP();//取得最大血量

int GetHP();//取得目前血量

int GetOriginMoveSpeed();//取得原始移動速度

int GetMoveSpeed();//取得目前移動速度

int GetOriginAttackDamage();//取得原始攻擊力

int GetAttackDamage();//取得目前攻擊力

int GetOriginAttackSpeed();//取得原始攻擊速度

int GetAttackSpeed();//取得攻擊速度

int GetOriginAttackRange();//取得原始攻擊範圍

int GetAttackRange();//取得攻擊範圍

int GetDefense();

bool IsDead();//確認是否死亡

void SetBitMapPosition()override;//設定動畫位置

void DecreaseHP(int dhp);//扣血

void IncreaseHP(int ihp);//加血

virtual void Attack() = 0;//攻擊

virtual void Move(int dx, int dy) = 0;//移動

virtual void MoveTo(int x, int y);//移動至

protected:

int maxHP, HP;//最大血量、目前血量

int originMoveSpeed, moveSpeed;//原始移動速度、目前移動速度

int originAttackDamage, attackDamage;//原始攻擊力、目前攻擊力

int originAttackSpeed, attackSpeed;//原始攻擊速度、目前攻擊速度

int originAttackRange, attackRange;//原始攻擊範圍、目前攻擊範圍

int originDefense, defense;//原始防禦力、目前防禦力

bool isGetHit;//被擊

bool isDead;//是否死亡

void Dead()override;//死亡時呼叫

vector<CAnimation\*> ani;//動畫

virtual void LoadAni(); //載入動畫

int aniDelay = 10;//動畫播放延遲

void AddAniBitMaps(string pic, int aniType, int picCount, int aniDelay = 10);//增加多張動畫圖片，傳入的pic為檔名，這個副程式會自動為每張圖加上編號

void AddAniBitMaps(char\* pic[], int aniType, int picCount, int aniDelay = 10);//增加多張動畫圖片

void AddAniBitMap(char\* pic, int aniType, int aniDelay = 10);//增加動畫圖片

bool CanMoveLeft(int perDisplacement);//可以向左移動，perDisplacement:每次的位移量是多少

bool CanMoveRight(int perDisplacement);//可以向右移動

bool CanMoveUp(int perDisplacement);//可以向上移動

bool CanMoveDown(int perDisplacement);//可以向下移動

bool IsInFloor();//是否卡在地板中

};

Character.cpp

#include "StdAfx.h"

#include "Character.h"

#include "GameSystem.h"

#include "Item.h"

Character::Character() :GameObject()

{

tag = "Character";

}

Character::~Character()

{

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

{

delete ani[i];

}

ani.clear();

}

Character::Character(string tag, int x, int y, int width, int height, int pictureID) : GameObject(tag, x, y, width, height, pictureID)

{

tag = "Character";

isGetHit = false;

}

Character::Character(string tag, int x, int y, int width, int height) : GameObject(tag, x, y, width, height) //No PictureID 的 Contructor

{

tag = "Character";

isGetHit = false;

}

void Character::SetMaxHP(int maxHp)

{

this->maxHP = maxHp;

}

void Character::SetHP(int hp)

{

this->HP = hp;

}

void Character::SetOriginMoveSpeed(int originMoveSpeed)

{

this->originMoveSpeed = originMoveSpeed;

}

void Character::SetMoveSpeed(int moveSpeed)

{

this->moveSpeed = moveSpeed;

}

void Character::SetOriginAttackDamage(int originAttack)

{

this->originAttackDamage = originAttack;

}

void Character::SetAttackDamage(int attack)

{

this->attackDamage = attack;

}

void Character::SetOriginAttackSpeed(int originAttackSpeed)

{

this->originAttackSpeed = originAttackSpeed;

}

void Character::SetAttackSpeed(int attackSpeed)

{

this->attackSpeed = attackSpeed;

}

void Character::SetOriginAttackRange(int originAttackRange)

{

this->originAttackRange = originAttackRange;

}

void Character::SetAttackRange(int attackRange)

{

this->attackRange = attackRange;

}

void Character::SetDefense(int defense) {

this->defense = defense;

}

void Character::SetIsDead(bool isDead)

{

this->isDead = isDead;

}

int Character::GetMaxHP()

{

return this->maxHP;

}

int Character::GetHP()

{

return this->HP;

}

int Character::GetOriginMoveSpeed()

{

return this->originMoveSpeed;

}

int Character::GetMoveSpeed()

{

return this->moveSpeed;

}

int Character::GetOriginAttackDamage()

{

return this->originAttackDamage;

}

int Character::GetAttackDamage()

{

return this->attackDamage;

}

int Character::GetOriginAttackSpeed()

{

return originAttackSpeed;

}

int Character::GetAttackSpeed()

{

return attackSpeed;

}

int Character::GetOriginAttackRange()

{

return this->originAttackRange;

}

int Character::GetAttackRange()

{

return this->attackRange;

}

int Character::GetDefense()

{

return this->defense;

}

bool Character::IsDead()

{

return this->isDead;

}

void Character::SetBitMapPosition()

{

for (auto& i : ani)

{

i->SetTopLeft(this->x - Map::GetSX(), this->y - Map::GetSY());

}

}

void Character::DecreaseHP(int dhp)

{

isGetHit = true;

this->HP -= dhp;

if (this->HP <= 0)//死亡

{

HP = 0;

isDead = true;

Dead();

}

}

void Character::IncreaseHP(int ihp)

{

this->HP += ihp;

if (this->HP > this->maxHP)//超過最大血量

this->HP = this->maxHP;

}

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

{

this->x = x;

this->y = y;

}

void Character::Dead()

{

GameSystem::DeleteGameObject(this);//刪除此物件

}

void Character::LoadAni()

{

}

void Character::AddAniBitMaps(string pic, int aniType, int picCount, int aniDelay)

{

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

{

//將檔名串接底線+編號+副檔名

string sPic;

sPic += pic + "\_" + to\_string(i) + ".bmp";

//轉成c-string

char\* cPic = new char[sPic.length() + 1];

strcpy(cPic, sPic.c\_str());

AddAniBitMap(cPic, aniType, aniDelay);//加入圖片

delete cPic;//delete new

}

}

void Character::AddAniBitMaps(char \* pic[], int aniType, int picCount, int aniDelay)

{

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

{

AddAniBitMap(pic[i], aniType, aniDelay);

}

}

void Character::AddAniBitMap(char\* pic, int aniType, int aniDelay)

{

while ((int)ani.size() <= aniType)//vector的大小不夠大

ani.push\_back(new CAnimation(aniDelay));//增加大小

ani[aniType]->SetDelayCount(aniDelay);//設定動畫速度

ani[aniType]->AddBitmap(pic, RGB(255, 255, 255));

}

bool Character::CanMoveLeft(int perDisplacement)

{

bool canMoveLeft = true;

for (int i = y; i < y + height; i++)

{

if (Map::HasObject(this->x - perDisplacement, i))//左半邊有東西

{

canMoveLeft = false;

return canMoveLeft;

}

}

return canMoveLeft;

}

bool Character::CanMoveRight(int perDisplacement)

{

bool canMoveRight = true;

for (int i = y; i < y + height; i++)

{

if (Map::HasObject(this->x + this->width + perDisplacement, i))//右半邊有東西

{

canMoveRight = false;

return canMoveRight;

}

}

return canMoveRight;

}

bool Character::CanMoveUp(int perDisplacement)

{

bool canMoveUp = true;

for (int i = x; i < x + width; i++)

{

if (Map::HasObject(i, y - perDisplacement) || y < 0)//上面有東西

{

canMoveUp = false;

return canMoveUp;

}

}

return canMoveUp;

}

bool Character::CanMoveDown(int perDisplacement)

{

bool canMoveDown = true;

for (int i = x; i < x + width; i++)

{

if (Map::HasObject(i, y + height + perDisplacement))//下面有東西

{

canMoveDown = false;

return canMoveDown;

}

}

return canMoveDown;

}

bool Character::IsInFloor()

{

for (int i = x; i < x + width; i++)

{

for (int j = y; j < y + height; j++)

{

if (Map::HasObject(i, j + 1))//+1是因為要判斷腳下

return true;

}

}

return false;

}

Demon.h

#pragma once

#include "Monster.h"

#include "Fire.h"

#include "Player.h"

//DefenseRange

class Demon:public Monster {

public:

Demon(); //BaseConstructor

~Demon();

Demon::Demon(string tag, int x, int y, int width, int height); //Constructor

void Attack()override; //攻擊

void IsAttack();

void AutoMove()override; //移動

void ShowBitMap()override; //顯示動畫

void Act()override; //組合各種動作(ISATTACK, AUTOMOVE, ATTACK)

private:

void LoadAni()override; //載入所有動畫

Fire\* fire; //自帶火焰特效

int currentBitMapNumber;

bool aniFix=false;

bool isBreak; //攻擊是否被中斷

};

Demon.cpp

#include "StdAfx.h"

#include "Demon.h"

Demon::Demon() {

tag = "Monster";

}

Demon::~Demon()

{

delete fire;

}

Demon::Demon(string tag, int x, int y, int width, int height) :Monster(tag, x, y, width, height) {

tag = "Monster";

SetDefenseRange(300);

SetAttackField(125);

SetdX(); //左邊

SetdY(); //右邊

SetOriginAttackRange(10);

maxHP = 50;

HP = maxHP;

attackDamage = 1;

SetAttackRange(GetOriginAttackRange());

originMoveSpeed = 3;

SetMoveSpeed(GetOriginMoveSpeed());

SetDefense(0);

status = STANDBY;

aniDelay = 5;

LoadAni();

LoadBitMap(".\\res\\demon\_idle.bmp");

fire = new Fire("Fire", x, y, 154, 105);

}

void Demon::AutoMove() {

if (GetRL() == LEFT) { //如果GetLR == LEFT

if (CanMoveLeft(moveSpeed)) { //如果往左沒有卡住

this->x = x - moveSpeed; //X 往左移動

if (CanMoveDown(moveSpeed)) { //往左會掉下去

this->x = x + moveSpeed;

SetRL(RIGHT);//換往右

}

currentAni = ANI\_LEFT; //設定現在動畫為LEFT

}

else //如果往左遇到障礙物、地形卡住

{

SetRL(RIGHT); //換往右

status = RIGHT;

}

}

else if (GetRL() == RIGHT) { //如果GetLR == R ,

if (CanMoveRight(moveSpeed)) { //如果往右沒有卡住

this->x = x + moveSpeed; //X 往右移動

if (CanMoveDown(moveSpeed)) { //往右會掉下去

this->x = x - moveSpeed;

SetRL(LEFT); //換往左

}

currentAni = ANI\_RIGHT; //設定現在動畫為RIGHT

}

else //如果往右遇到障礙物、地形卡住

{

SetRL(LEFT); //換往左

status = LEFT;

}

}

if (x <= GetdX(0) && GetRL() != 0) //GetdX[0]左邊警戒範圍，如果超過左邊 改右邊移動

SetRL(RIGHT);

else if (x >= GetdX(1) && GetRL() != 0) //GetdX[1]右邊警戒範圍，如果超過右邊 改左邊移動

SetRL(LEFT);

}

void Demon::Attack() {

const int ATTACK\_SPEED = 4;

int placeRelativePlayer = PlaceRelativePlayer(player);

if (ani[ANI\_ATTACK\_RIGHT]->GetCurrentBitmapNumber() != 0) {

status = ATTACK;

currentAni = ANI\_ATTACK\_RIGHT;

}

else if (ani[ANI\_ATTACK\_LEFT]->GetCurrentBitmapNumber() != 0) {

status = ATTACK;

currentAni = ANI\_ATTACK\_LEFT;

}

else if (IsInAttackField(player->GetX(), player->GetY(), 100, -20, 0, 0)) { //Player在怪物攻擊領域內 跟隨 #要增加跟隨感應距離改AttackField

status = ATTACK;

moveSpeed = ATTACK\_SPEED;

if (IsPlayerInRange(player, 0, 0, 25, 5) == false) { //如果怪物還沒撞到腳色

if (placeRelativePlayer == RIGHT) //如果怪物在人的右邊

{

if (!CanMoveLeft(moveSpeed)) //可以往左追

{

Monster::Up(); //怪物往上移(走道地形上)

}

x -= moveSpeed;

currentAni = ANI\_LEFT;

}

else if (placeRelativePlayer == LEFT) //如果怪物在人的左邊

{

if (!CanMoveRight(moveSpeed)) //可以往右追

{

Monster::Up(); //怪物往上移(走道地形上)

}

x += moveSpeed;

currentAni = ANI\_RIGHT; //設定往右的動畫

}

}

else {

if (placeRelativePlayer == RIGHT) { //如果怪物在人的右邊

currentAni = ANI\_ATTACK\_LEFT; //設定往左攻擊的動畫

}

else if (placeRelativePlayer == LEFT) { //如果怪物在人的左邊

currentAni = ANI\_ATTACK\_RIGHT; //設定往右攻擊的動畫

}

}

}

else { //脫離警戒領域 回復來回走動

if (placeRelativePlayer == RIGHT)

currentAni = ANI\_RIGHT;

else if (placeRelativePlayer == LEFT)

currentAni = ANI\_LEFT;

status = STANDBY;

moveSpeed = originMoveSpeed;

}

}

void Demon::IsAttack()

{

int placeRelativePlayer = PlaceRelativePlayer(player);

if (currentAni != ANI\_ISATTACK\_LEFT && currentAni != ANI\_ATTACK\_RIGHT)

{

if (placeRelativePlayer == RIGHT) //決定被攻擊的動畫方向

{

status = ISATTACK;

currentAni = ANI\_ISATTACK\_LEFT;

}

else if (placeRelativePlayer == LEFT) //決定被攻擊的動畫方向

{

status = ISATTACK;

currentAni = ANI\_ISATTACK\_RIGHT;

}

}

if (!ani[currentAni]->IsEnd()) { //reset state;(GetCurrentBitmapNumber()要在ShowBitMap才會動)

status = ISATTACK;

}

else

{

ani[ANI\_ATTACK\_RIGHT]->Reset();

ani[ANI\_ATTACK\_LEFT]->Reset();

status = STANDBY;

}

}

void Demon::Act() //組合各種動作(ISATTACK, AUTOMOVE, ATTACK)

{

Fall(moveSpeed);

if (status == ISATTACK) {

IsAttack();

}

if (status != ISATTACK) {

Attack();

}

if (status == STANDBY || status == LEFT || status == RIGHT) {

AutoMove();

}

ani[currentAni]->OnMove(); //顯示動畫

}

void Demon::ShowBitMap() {

currentBitMapNumber = ani[currentAni]->GetCurrentBitmapNumber();

fire->ShowBitMap(x, y, currentAni, ani[currentAni]->GetCurrentBitmapNumber());

ani[currentAni]->OnShow();

}

void Demon::LoadAni()

{

char\* aniIdle[1] = { ".\\res\\demon\_idle.bmp" };

AddAniBitMaps(aniIdle, ANI\_IDLE, 1, aniDelay);

char\* aniLeft[6] = { ".\\res\\demon\_left\_0.bmp",".\\res\\demon\_left\_1.bmp" ,".\\res\\demon\_left\_2.bmp"

,".\\res\\demon\_left\_3.bmp" ,".\\res\\demon\_left\_4.bmp" ,".\\res\\demon\_left\_5.bmp" };

AddAniBitMaps(aniLeft, ANI\_LEFT, 6, aniDelay);

char\* aniRight[6] = { ".\\res\\demon\_right\_0.bmp", ".\\res\\demon\_right\_1.bmp" ,".\\res\\demon\_right\_2.bmp" ,

".\\res\\demon\_right\_3.bmp" ,".\\res\\demon\_right\_4.bmp" ,".\\res\\demon\_right\_5.bmp" };

AddAniBitMaps(aniRight, ANI\_RIGHT, 6, aniDelay);

char\* aniAttack\_left[11] = { ".\\res\\demon\_attackleft\_0.bmp", ".\\res\\demon\_attackleft\_1.bmp", ".\\res\\demon\_attackleft\_2.bmp",

".\\res\\demon\_attackleft\_3.bmp", ".\\res\\demon\_attackleft\_4.bmp", ".\\res\\demon\_attackleft\_5.bmp",

".\\res\\demon\_attackleft\_6.bmp", ".\\res\\demon\_attackleft\_7.bmp", ".\\res\\demon\_attackleft\_7.bmp",

".\\res\\demon\_attackleft\_7.bmp", ".\\res\\demon\_attackleft\_7.bmp" };

AddAniBitMaps(aniAttack\_left, ANI\_ATTACK\_LEFT, 11, aniDelay);

char\* aniAttack\_right[11] = { ".\\res\\demon\_attackright\_0.bmp", ".\\res\\demon\_attackright\_1.bmp", ".\\res\\demon\_attackright\_2.bmp",

".\\res\\demon\_attackright\_3.bmp", ".\\res\\demon\_attackright\_4.bmp", ".\\res\\demon\_attackright\_5.bmp",

".\\res\\demon\_attackright\_6.bmp", ".\\res\\demon\_attackright\_7.bmp", ".\\res\\demon\_attackright\_7.bmp",

".\\res\\demon\_attackright\_7.bmp", ".\\res\\demon\_attackright\_7.bmp" };

AddAniBitMaps(aniAttack\_right, ANI\_ATTACK\_RIGHT, 11, aniDelay);

char\* aniIsAttack\_right[7] = { ".\\res\\isattack\_right\_0.bmp", ".\\res\\isattack\_right\_1.bmp", ".\\res\\isattack\_right\_2.bmp",".\\res\\isattack\_right\_2.bmp",

".\\res\\isattack\_right\_2.bmp",".\\res\\isattack\_right\_2.bmp",".\\res\\isattack\_right\_2.bmp" };

AddAniBitMaps(aniIsAttack\_right, ANI\_ISATTACK\_RIGHT, 7, aniDelay);

char\* aniIsAttack\_left[7] = { ".\\res\\isattack\_left\_0.bmp", ".\\res\\isattack\_left\_1.bmp", ".\\res\\isattack\_left\_2.bmp",

".\\res\\isattack\_left\_2.bmp", ".\\res\\isattack\_left\_2.bmp",".\\res\\isattack\_left\_2.bmp",".\\res\\isattack\_left\_2.bmp "};

AddAniBitMaps(aniIsAttack\_left, ANI\_ISATTACK\_LEFT, 7, aniDelay);

}

Door.h

#pragma once

//Door

//門

//繼承GameObject

#include "GameObject.h"

#include "GameSystem.h"

#include "Monster.h"

#include "Player.h"

class Door :public GameObject

{

public:

Door();

~Door();

Door(string tag, int x, int y, int width, int height);

void Kicked();//被踢

bool GetRuin();

void SetBitMapPosition()override;

private:

const int LEFT = 1, RIGHT = 2;

void Dead()override;

void DazeMonster();

void ShowBitMap()override;

void LoadAni();//載入圖片

bool isRuin;//是否被破壞

int lr; //被破壞時門飛出去的方向

CAnimation\* ani;

int stunRangeX, stunRangeY;

};

Door.cpp

#include "StdAfx.h"

#include "Door.h"

Door::Door() :GameObject()

{

tag = "Door";

}

Door::~Door()

{

delete ani;

}

Door::Door(string tag, int x, int y, int width, int height) : GameObject(tag, x, y, width, height)

{

isRuin = false;

ani = new CAnimation();

layer = GameSystem::LAYER::LAYER\_DOOR;

LoadAni();

stunRangeX = 100;

stunRangeY = 100;

}

void Door::Kicked()

{

Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player");

isRuin = true;

if (player->GetX()+50 > this->x) //被破壞時決定往哪飛

lr = LEFT;

else

lr = RIGHT;

}

bool Door::GetRuin()

{

return this->isRuin;

}

void Door::SetBitMapPosition()

{

ani->SetTopLeft(x - Map::GetSX(), y - Map::GetSY());

}

void Door::Dead()

{

//攻擊怪物

DazeMonster();

GameSystem::DeleteGameObject(this);//刪除物件

}

void Door::DazeMonster()

{

vector <Monster\*> allMonster = GameSystem::GetGameObjectsWithTag<Monster>("Monster");

for (auto & i : allMonster)

{

bool x = i->GetX() < this->x + this->width + stunRangeX && i->GetX() > this->x - stunRangeX;

bool y = i->GetY() < this->y + stunRangeY && i->GetY() > this->y - stunRangeY/2;

if (x && y)

{

i->PlayerAttack(0);

}

}

}

void Door::ShowBitMap()

{

if (isRuin)

{

if (lr == RIGHT)

x += 1;

else

x -= 1;

ani->OnMove();//播放破壞動畫

}//被破壞的話

ani->OnShow();

if (ani->IsEnd())//播完破壞動畫後

Dead();//刪除物件

}

void Door::LoadAni()

{

char\* pic0 = ".\\res\\door0.bmp";

char\* pic1 = ".\\res\\door1.bmp";

char\* pic2 = ".\\res\\door2.bmp";

char\* pic3 = ".\\res\\door3.bmp";

ani->AddBitmap(pic0);

ani->AddBitmap(pic1);

ani->AddBitmap(pic2);

ani->AddBitmap(pic3);

ani->SetDelayCount(3);

}

Effect.h

#pragma once

#include "GameObject.h"

#include "Player.h"

#include "GameSystem.h"

class Effect :public GameObject{

public:

Effect(string tag, int x, int y, int width, int height);

~Effect();

void SetHit(int hit); //設定是否有打中(每一組動畫只能造成一次傷害)

virtual void SetXY(int hostX, int hostY, int hostCurrentAni); //隨著擁有者移動 設定XY

virtual void ShowBitMap(int hostX, int hostY, int hostCurrentAni, int attackAniNumber); //顯示動畫特效

protected:

void EffectAttackMonster(int attackDamage); //特效對怪獸造成傷害(每個特效只能造成一次傷害(範圍))

//-------------------------Picture And Animation------------------------------//

void AddAniBitMaps(char\* pic[], int aniType, int picCount); //增加多張動畫圖片

void AddAniBitMap(char\* pic, int aniType); //增加動畫圖片

void SetBitMapPosition()override; //設定動畫左上角

virtual void LoadAni(); //載入動畫

//-----------------------------jedgment------------------------------------------------//

bool IsObjectInRange(GameObject\* obj, int right\_fix, int left\_fix, int up\_fix, int down\_fix); //判斷動畫與物件是否重疊

bool CanMoveLeft(int perDisplacement);//可以向左移動，perDisplacement:每次的位移量是多少

bool CanMoveRight(int perDisplacement);//可以向右移動

bool CanMoveUp(int perDisplacement);//可以向上移動

bool CanMoveDown(int perDisplacement);//可以向下移動

Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player");

int currentAni = 0; //現在執行哪個動畫 (左邊:0, 右邊:1)

bool isHit = false; //每一次的火焰只能造成一次傷害

const int ANI\_ATTACK\_LEFT = 3, ANI\_ATTACK\_RIGHT = 4; //E攻擊動畫

vector<CAnimation\*> ani; //動畫集

};

Effect.cpp

#include "StdAfx.h"

#include "Effect.h"

#include "Monster.h"

//---------------------------------Constructor------------------------------//

Effect::Effect(string tag, int x, int y, int width, int height):GameObject(tag, x, y, width, height){

isHit = 0;

currentAni = 0;

}

Effect::~Effect()

{

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

{

delete ani[i];

}

ani.clear();

}

//-----------------------------------SetHit----------------------------------//

void Effect::SetHit(int hit) {

this->isHit = hit;

}

void Effect::SetBitMapPosition()

{

for (auto& i : ani)

{

i->SetTopLeft(this->x - Map::GetSX(), this->y - Map::GetSY());

}

}

//-------------------------------other-----------------------------------------//

void Effect::EffectAttackMonster(int attackDamage)

{

vector<Monster\*> allMonsters = GameSystem::GetGameObjectsWithTag<Monster>("Monster");

for (auto& i : allMonsters) {

if (IsObjectInRange(i, 0, 0, 15, 0) == 1) {

i->PlayerAttack(attackDamage);

isHit = true;

}

}

}

void Effect::SetXY(int hostX, int hostY, int hostCurrentAni) {}

//---------------------------Picture And Animation-----------------------------------------//

void Effect::AddAniBitMaps(char \* pic[], int aniType, int picCount)

{

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

{

AddAniBitMap(pic[i], aniType);

}

}

void Effect::AddAniBitMap(char\* pic, int aniType)

{

while ((int)ani.size() <= aniType)//vector的大小不夠大

ani.push\_back(new CAnimation);//增加大小

ani[aniType]->AddBitmap(pic, RGB(255, 255, 255));

}

void Effect::LoadAni() {}

void Effect::ShowBitMap(int hostX, int hostY, int hostCurrentAni, int attackAniNumber) {}

//---------------------------------judgment-------------------------------------//

bool Effect::IsObjectInRange(GameObject\* obj, int right\_fix, int left\_fix, int up\_fix, int down\_fix) {

int RIGHT\_EDGE = x + width + right\_fix, LEFT\_EDGE = x - left\_fix,

UP\_EDGE = y - up\_fix, DOWN\_EDGE = y + height + down\_fix;

int OB\_X = obj->GetX(), OB\_Y = obj->GetY(), OB\_WIDTH = obj->GetWidth(), OB\_HEIGHT = obj->GetHeight();

int OB\_RIGHT\_EDGE = OB\_X + OB\_WIDTH, OB\_LEFT\_EDGE = OB\_X,

OB\_UP\_EDGE = OB\_Y, OB\_DOWN\_EDGE = OB\_Y + OB\_HEIGHT;

if (OB\_DOWN\_EDGE < UP\_EDGE)//物件整個再特效上面

return false;

else if (OB\_UP\_EDGE > DOWN\_EDGE)//物件整個再特效下面

return false;

else if (OB\_RIGHT\_EDGE >= LEFT\_EDGE && OB\_RIGHT\_EDGE <= RIGHT\_EDGE) //人在左, 火焰在右

return true;

else if (OB\_LEFT\_EDGE <= RIGHT\_EDGE && OB\_LEFT\_EDGE >= LEFT\_EDGE) //人在右, 火焰在左

return true;

return false;

}

bool Effect::CanMoveLeft(int perDisplacement)

{

bool canMoveLeft = true;

for (int i = y; i < y + height; i++)

{

if (Map::HasObject(this->x - perDisplacement, i))//左半邊有東西

{

canMoveLeft = false;

return canMoveLeft;

}

}

return canMoveLeft;

}

bool Effect::CanMoveRight(int perDisplacement)

{

bool canMoveRight = true;

for (int i = y; i < y + height; i++)

{

if (Map::HasObject(this->x + this->width + perDisplacement, i))//右半邊有東西

{

canMoveRight = false;

return canMoveRight;

}

}

return canMoveRight;

}

bool Effect::CanMoveUp(int perDisplacement)

{

bool canMoveUp = true;

for (int i = x; i < x + width; i++)

{

if (Map::HasObject(i, y - perDisplacement) || y < 0)//上面有東西

{

canMoveUp = false;

return canMoveUp;

}

}

return canMoveUp;

}

bool Effect::CanMoveDown(int perDisplacement)

{

bool canMoveDown = true;

for (int i = x; i < x + width; i++)

{

if (Map::HasObject(i, y + height + perDisplacement))//下面有東西

{

canMoveDown = false;

return canMoveDown;

}

}

return canMoveDown;

}

EquipedSlot.h

#pragma once

#include "UInterface.h"

class EquipedSlot :public UInterface {

public:

EquipedSlot(string tag, int x, int y, int width, int height,int ID);

~EquipedSlot();

int GetID();

bool GetIsEquipmentPicSet();

void SetEquipmentPicture(char\* pictureAddress, int x, int y); //當裝備picked 設定quipment圖片

void ShowBitMap()override; //顯示現有圖片(可能只有UI)

private:

CMovingBitmap\* equipmentPicture;

bool isEquipmentPicSet = false;

int ID; //洞的編號

};

EquipedSlot.cpp

#include "StdAfx.h"

#include "EquipedSlot.h"

EquipedSlot::EquipedSlot(string tag, int x, int y, int width, int height, int ID):UInterface(tag, x, y, width, height)

{

this->tag = "EquipedSlot";

LoadBitMap(".\\res\\equiped\_slot.bmp");

equipmentPicture = new CMovingBitmap();

UIpicture.SetTopLeft(x, y);

isEquipmentPicSet = false;

this->ID = ID;

}

EquipedSlot::~EquipedSlot()

{

delete equipmentPicture;

}

int EquipedSlot::GetID()

{

return this->ID;

}

bool EquipedSlot::GetIsEquipmentPicSet()

{

return this->isEquipmentPicSet;

}

void EquipedSlot::SetEquipmentPicture(char\* pictureAddress, int x, int y) {

if (isEquipmentPicSet == false) {

equipmentPicture->LoadBitmap(pictureAddress);

equipmentPicture->SetTopLeft(x, y);

isEquipmentPicSet = true;

}

else {

delete equipmentPicture;

equipmentPicture = new CMovingBitmap();

equipmentPicture->LoadBitmap(pictureAddress);

equipmentPicture->SetTopLeft(x, y);

isEquipmentPicSet = true;

}

}

void EquipedSlot::ShowBitMap() {

UIpicture.ShowBitmap();

if(isEquipmentPicSet==true)

equipmentPicture->ShowBitmap();

}

Fire.h

#pragma once

#include"Effect.h"

#include"GameSystem.h"

#include"Player.h"

class Fire :public Effect {

public:

Fire(string tag, int x, int y, int width, int height); //Constructor

~Fire();

void ShowBitMap(int hostX, int hostY, int hostCurrentAni, int attackAniNumber)override; //顯示動畫(有條件)

//bool IsPlayerInRange(int right\_fix, int left\_fix, int up\_fix, int down\_fix); //兩者的圖形是否有觸碰到

private:

void LoadAni()override;

void SetXY(int hostX, int hostY, int hostCurrentAni)override; //隨著Demon更新火焰位置

//void SetBitMapPosition()override; //設定動畫左上角

//vector<CAnimation\*> ani; //動畫集

//Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player");

///int currentAni=0; //現在執行哪個動畫 (左邊火焰:0, 右邊火焰版:1)

//int hit = 0; //每一次的火焰只能造成一次傷害

enum ANI {

ANI\_FIRE\_LEFT = 0,

ANI\_FIRE\_RIGHT

};

};

Fire.cpp

#include "StdAfx.h"

#include "Fire.h"

#include "Monster.h"

#include "GameSystem.h"

#include "Player.h"

#include <vector>

Fire::Fire(string tag, int x, int y, int width, int height):Effect(tag, x, y, width, height) {

tag = "Fire";

LoadAni();

}

Fire::~Fire()

{

}

void Fire::SetXY(int hostX, int hostY, int hostCurrentAni) {

if (hostCurrentAni == ANI\_ATTACK\_RIGHT) { //3是左攻擊狀態

this->x = hostX + 80;

this->y = hostY + 85;

}

else if (hostCurrentAni == ANI\_ATTACK\_LEFT) {

this->x = hostX - 95;

this->y = hostY + 85;

}

}

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

void Fire::ShowBitMap(int hostX, int hostY, int hostCurrentAni, int attackAniNumber) {

SetXY(hostX, hostY, hostCurrentAni);

if (attackAniNumber >= 6 && hostCurrentAni == ANI\_ATTACK\_LEFT) { //3是左攻擊狀態

currentAni = ANI\_FIRE\_LEFT;

SetBitMapPosition();

ani[currentAni]->OnMove();

ani[currentAni]->OnShow();

if (IsObjectInRange(player, 0, 0, 15, 25) && ani[currentAni]->GetCurrentBitmapNumber()>=2 && isHit ==0){

player->DecreaseHP(10);

isHit = 1;

}

}

else if (attackAniNumber >= 6 && hostCurrentAni == ANI\_ATTACK\_RIGHT) {//4是右攻擊狀態

currentAni = ANI\_FIRE\_RIGHT;

SetBitMapPosition();

ani[currentAni]->OnMove();

ani[currentAni]->OnShow();

if (IsObjectInRange(player,0,0, 15, 25) && ani[currentAni]->GetCurrentBitmapNumber() >= 2 && isHit ==0){

player->DecreaseHP(10);

isHit = 1;

}

}

else isHit = 0;

}

void Fire::LoadAni() {

char\* aniFire\_left[5] = { ".\\res\\fire\_left\_0.bmp", ".\\res\\fire\_left\_1.bmp",

".\\res\\fire\_left\_2.bmp" , ".\\res\\fire\_left\_3.bmp" , ".\\res\\fire\_left\_4.bmp" };

AddAniBitMaps(aniFire\_left, ANI\_FIRE\_LEFT, 5);

char\* aniFire\_right[5] = { ".\\res\\fire\_right\_0.bmp", ".\\res\\fire\_right\_1.bmp",

".\\res\\fire\_right\_2.bmp" , ".\\res\\fire\_right\_3.bmp" , ".\\res\\fire\_right\_4.bmp" };

AddAniBitMaps(aniFire\_right, ANI\_FIRE\_RIGHT, 5);

}

Floor.h

#pragma once

//Floor

//地板

//繼承GameObject

#include "GameObject.h"

class Floor :public GameObject

{

public:

Floor();

~Floor();

Floor(string tag, int x, int y, int width, int height, int pictureID);

Floor(string tag, int x, int y, int width, int height);

void ShowBitMap()override;

private:

};

Floor.cpp

#include "StdAfx.h"

#include "Floor.h"

#include "GameSystem.h"

Floor::Floor()

{

tag = "Floor";

}

Floor::~Floor()

{

}

Floor::Floor(string tag, int x, int y, int width, int height, int pictureID) :GameObject(tag, x, y, width, height, pictureID)

{

layer = GameSystem::LAYER::LAYER\_FLOOR;

}

Floor::Floor(string tag, int x, int y, int width, int height) : GameObject(tag, x, y, width, height)

{

layer = GameSystem::LAYER::LAYER\_FLOOR;

LoadBitMap(".\\res\\floor.bmp");

}

void Floor::ShowBitMap()

{

//objectPic.ShowBitmap();

CDC \*pDC = CDDraw::GetBackCDC();

CPen \*pp, p(PS\_SOLID, 1, RGB(128, 128, 128));

pp = pDC->SelectObject(&p);

CBrush \*pg, g(RGB(128, 128, 128));

pg = pDC->SelectObject(&g);

pDC->Rectangle(x - Map::GetSX(), y - Map::GetSY(), x + width - Map::GetSX(), y + height - Map::GetSY());

pDC->SelectObject(pp);

pDC->SelectObject(pg);

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

}

GameObject.h

#pragma once

//GameObject

//遊戲物件

//場上所有物件的Base Class

#include "StdAfx.h"

#include <string>

using namespace std;

#include "gamelib.h"

using namespace game\_framework;

#include "resource.h"

class GameObject

{

public:

GameObject();

virtual ~GameObject();

GameObject(string tag, int x, int y, int width, int height, int pictureID);

GameObject(string tag, int x, int y, int width, int height);

void SetTag(string tag);

void SetX(int x);

void SetY(int y);

void SetWidth(int width);

void SetHeight(int height);

string GetTag();

int GetX();

int GetY();

int GetWidth();

int GetHeight();

int GetLayer();//取得圖層

virtual void Act();

void LoadBitMap(char\* pictureAddress);//用路徑載入圖片

void LoadBitMap(int pictureID);//用編號載入圖片

virtual void SetBitMapPosition();//設定圖片位置

virtual void ShowBitMap();//顯示圖片

//bool IsAnthorObjectInRange(GameObject\* object); //object是否與此物件有重疊

protected:

string tag;//物件標籤

int x, y;//x,y座標

int width;//寬度

int height;//高度

virtual void Dead();//消滅時呼叫

CMovingBitmap objectPic;//物件圖片

int pictureID;//圖片編號

int layer;//圖層

};

GameObject.cpp

#include "StdAfx.h"

#include "GameObject.h"

#include "Map.h"

#include "Monster.h"

GameObject::GameObject()

{

tag = "GameObject";

x = 0;

y = 0;

width = 10;

height = 10;

layer = 0;

}

GameObject::~GameObject()

{

}

GameObject::GameObject(string tag, int x, int y, int width, int height, int pictureID)

{

this->tag = tag;

this->x = x;

this->y = y;

this->width = width;

this->height = height;

this->pictureID = pictureID;

this->LoadBitMap(pictureID);//載入圖片

}

GameObject::GameObject(string tag, int x, int y, int width, int height) //沒有pictureID的Contructor

{

this->tag = tag;

this->x = x;

this->y = y;

this->width = width;

this->height = height;

}

string GameObject::GetTag()

{

return tag;

}

int GameObject::GetX()

{

return this->x;

}

int GameObject::GetY()

{

return this->y;

}

void GameObject::SetTag(string tag)

{

this->tag = tag;

}

void GameObject::SetX(int x)

{

this->x = x;

}

void GameObject::SetY(int y)

{

this->y = y;

}

void GameObject::SetWidth(int width)

{

this->width = width;

}

void GameObject::SetHeight(int height)

{

this->height = height;

}

int GameObject::GetWidth()

{

return width;

}

int GameObject::GetHeight()

{

return height;

}

int GameObject::GetLayer()

{

return layer;

}

void GameObject::Act()

{

}

void GameObject::LoadBitMap(char\* pictureAddress)

{

objectPic.LoadBitmap(pictureAddress, RGB(255, 255, 255));

}

void GameObject::LoadBitMap(int pictureID)

{

objectPic.LoadBitmap(pictureID, RGB(255, 255, 255));

}

void GameObject::SetBitMapPosition()

{

objectPic.SetTopLeft(x - Map::GetSX(), y - Map::GetSY());

}

void GameObject::ShowBitMap()

{

objectPic.ShowBitmap();

}

void GameObject::Dead()

{

GameSystem::DeleteGameObject(this);//刪除此物件

}

GameSystem.h

#pragma once

//GameSystem

//遊戲系統

//紀錄場上所有物件資訊

//提供各種共用函式

#include "StdAfx.h"

#include <vector>

using namespace std;

#include "gamelib.h"

using namespace game\_framework;

#include "Map.h"

#include "audio.h"

#include "GameObject.h"

#include "UInterface.h"

class GameSystem

{

public:

GameSystem();

static void Load();//載入

static void Init();//初始化

static void Exit();//離開遊戲

enum OVER\_MODE

{

OVER\_WIN,

OVER\_LOSE

};

static int GetOverMode();//取得哪種結束模式

static bool IsGameOver();//是否遊戲結束

static void SetGameOver(int setOverMode);//設定遊戲結束

static bool IsNextStage();

static void SetIsNextStage(bool nextStage);

static void StagePlusOne();

static int GetNowStage();

static vector<GameObject\*>& GetAllGameObject();//取得所有遊戲物件

static void AddGameObject(GameObject\* obj);//加入遊戲物件

static void AddUserInterface(UInterface\* ui); //加入UI物件

static void DeleteGameObject(GameObject\* obj);//刪除遊戲物件

static void DeleteUInterface(UInterface\* ui);//刪除UI物件

static void DeleteAllGameObject();//刪除所有遊戲物件

static void DeleteAllGameObjectExceptPlayer(); //刪除所有遊戲物件(player除外)

static void DeleteAllUI();//刪除所有UI物件

static void CreateFloor(int x, int y, int width, int height);//拼地板，傳入座標與寬高

static void ChangeToNextStage();

static double Rand(double maxValue);//亂數

static double Rand(double minValue, double maxValue);//亂數

template <class T> static T\* GetGameObjectWithTag(string tag)//取得特定標籤的遊戲物件(單一個)

{

for (auto& i : gameObjectList)

{

if (i->GetTag() == tag)

{

return static\_cast<T\*>(i);

}

}

return NULL;

}

template <class T> static vector<T\*> GetGameObjectsWithTag(string tag)//取得特定標籤的遊戲物件(多個)

{

vector<T\*> targetObjects;

for (auto& i : gameObjectList)

{

if (typeid(T) == typeid(GameObject))

{

}

if (i->GetTag() == tag)

{

targetObjects.push\_back(static\_cast<T\*>(i));

}

}

return targetObjects;

}

template <class T> static vector<T\*> GetGameObjectsWithType()//取得特定類別的遊戲物件(多個)

{

vector<T\*> targetObjects;

for (auto& i : gameObjectList)

{

if(dynamic\_cast<T\*>(i)!=NULL)

targetObjects.push\_back(static\_cast<T\*>(i));

}

return targetObjects;

}

template <class T> static T\* GetGameObjectWithType()//取得特定類別的遊戲物件(單一個)

{

for (auto& i : gameObjectList)

{

if (dynamic\_cast<T\*>(i) != NULL)

return dynamic\_cast<T\*>(i);

}

return NULL;

}

template <class T>static T\* GetUInterfaceWithType() //取得特定類別的UI(單一個)

{

for (auto& i : gameUIList)

{

if (dynamic\_cast<T\*>(i) != NULL)

return dynamic\_cast<T\*>(i);

}

return NULL;

}

template <class T>static T\* GetUInterfaceWithTag(string tag) //取得特定標籤的遊戲物件(單一個)

{

for (auto& i : gameUIList) {

int x = i->GetX();

if (i->GetTag() == tag)

{

return static\_cast<T\*>(i);

}

}

return NULL;

}

template <class T>static vector<T\*> GetUInterfacesWithTag(string tag) //取得特定標籤的UI物件(多個)

{

vector<T\*> targetUIs;

for (auto& i : gameUIList)

{

if (typeid(T) == typeid(UInterface))

{

}

if (i->GetTag() == tag)

{

targetUIs.push\_back(static\_cast<T\*>(i));

}

}

return targetUIs;

}

static void AllObjectAct();//所有物件動作

static void SetAllObjectBitMapPosition();//設定所有物件的圖片位置

static void ShowAllObject();//顯示所有物件

static void ShowAllUI();//顯示所有UI物件

//------------顯示文字-------------//

enum ALIGN

{

ALIGN\_TOP = 0,

ALIGN\_BOTTOM,

ALIGN\_LEFT,

ALIGN\_RIGHT,

ALIGN\_CENTER

};

static void ShowText(string text

, int left = 0, int top = 0, int right = SIZE\_X, int bottom = SIZE\_Y//矩形的四個點座標

, int alignHor = ALIGN\_LEFT, int alignVer = ALIGN\_TOP//顯示文字,對齊方式

, int fontSize = 16, COLORREF textColor = RGB(0, 0, 0)//字體大小,字體顏色

, int bkMode = 1, COLORREF bkColor = RGB(255, 255, 255));//背景模式(OPAQUE:不透明, TRANSPARENT:透明),背景顏色

static void DrawRectangle(int x,int y,int width,int height,COLORREF color);//畫矩形

//------------音訊相關-------------//

enum AUDIO

{

AUDIO\_GAME\_INIT = 0,

AUDIO\_GAME\_RUN\_1,

AUDIO\_GAME\_RUN\_2,

AUDIO\_GAME\_OVER\_LOSE,

AUDIO\_GAME\_OVER\_WIN

};

static void PlayAudio(AUDIO id);//播放音樂

static void StopAudio(AUDIO id);//停止音樂

static void CreatStage1Object();

//------------圖層相關------------//

enum LAYER//圖層編號

{

LAYER\_DOOR = 0,

LAYER\_FLOOR,

LAYER\_ITEM,

LAYER\_MONSTER,

LAYER\_PLAYER

};

//------------裝備ID------------//

enum EQUIPMENT\_ID

{

EQUIPMENT\_WEAPON,

EQUIPMENT\_TRADITIONAL\_SWORD

};

private:

static void CreatStage2Object(); //創建第二關(第一關以後要保留UI和Player)

static vector<GameObject\*> gameObjectList;//存放所有遊戲物件的列表

static vector<UInterface\*> gameUIList;//存放所有遊戲UI物件列表

const static int FLOOR\_WIDTH = 20, FLOOR\_HEIGHT = 20;

static int overMode;//以哪種模式結束(玩家死掉?打贏BOSS?)

static bool isGameOver;//遊戲是否結束

static bool isNextStage;

static int nowStage;

static const int stage1FloorCount = 35;//第一關地板數量

static int stage1Floor[stage1FloorCount][4];//第一關的地板，x,y,width,height四個數字

static bool isLoaded;//是否已載入

};

GameSystem.cpp

#include"Stdafx.h"

#include <functional>

#include <random>

#include <gdiplus.h>

#include "GameSystem.h"

#include "Monster.h"

#include "Player.h"

#include "GameObject.h"

#include "UInterface.h"

#include "gamelib.h"

#include "Map.h"

#include "GameObject.h"

#include "Character.h"

#include "Player.h"

#include "Floor.h"

#include "Item.h"

#include "Fire.h"

#include "Demon.h"

#include "EquipedSlot.h"

#include "UIBlood.h"

#include "UIMonsterBlood.h"

#include "Door.h"

#include "Goal.h"

#include "Boss.h"

#include "Portal.h"

#include "Treasure.h"

vector<GameObject\*> GameSystem::gameObjectList;//初始化物件列表

vector<UInterface\*> GameSystem::gameUIList;//初始化物件列表

int GameSystem::overMode = GameSystem::OVER\_LOSE;//預設遊戲結束模式

bool GameSystem::isGameOver = false;

bool GameSystem::isNextStage = false;

int GameSystem::nowStage = 0;

bool GameSystem::isLoaded = false;

int GameSystem::stage1Floor[stage1FloorCount][4] = //第一關的地板

{

{0,600,710,120},{ 0, 720, 710, 640 },{ 330, 400, 410, 80 },{ 890, 170, 200, 45 },{ 1090, 600, 850, 120 }

,{ 1090, 0, 65, 490 },{ 0, 1360, 310, 205 },{ 650, 1360, 60, 150 },{ 710, 910, 220, 45 },{ 825, 1150, 180, 35 }

,{ 710, 1325, 120, 35 },{ 920, 710, 170, 35 },{ 1090, 720, 850, 230 },{ 1090, 1200, 850, 170 },{ 1530, 950, 410,250}

,{ 1570, 1370, 100, 195 },{ 1115, 380, 455, 35 },{ 1315, 170, 170, 95 },{ 1425, 170, 275, 50 },{ 1700, 120, 80, 480 }

,{ 2160, 600, 1680, 120 },{ 1890, 445, 610, 35 },{ 2160, 720, 310, 650 },{ 2230, 1370, 140, 75 },{ 2700, 720, 520, 110 }

,{ 3080, 830, 140, 130 },{ 2700, 960, 520, 320 },{ 2140, 0, 1700, 30 },{ 2140, 210, 930, 30 },{ 2630, 30, 80, 125 }

,{ 3260, 30, 580, 360 },{ 3260, 390, 100, 210 },{ 2830, 410, 80, 190 },{ 2630, 460, 200, 50 },{ 2910, 410, 160, 40 }

};

GameSystem::GameSystem()

{

}

int GameSystem::GetOverMode()

{

return overMode;

}

bool GameSystem::IsGameOver()

{

return isGameOver;

}

void GameSystem::SetGameOver(int setOverMode)

{

isGameOver = true;

overMode = setOverMode;

}

bool GameSystem::IsNextStage()

{

return isNextStage;

}

void GameSystem::SetIsNextStage(bool nextStage)

{

isNextStage = nextStage;

}

void GameSystem::StagePlusOne()

{

nowStage = nowStage + 1;

}

int GameSystem::GetNowStage()

{

return nowStage;

}

vector<GameObject\*>& GameSystem::GetAllGameObject()//取得所有物件

{

return gameObjectList;

}

void GameSystem::AddGameObject(GameObject\* obj)//增加遊戲物件

{

for (unsigned int i = 0; i < gameObjectList.size(); i++)//用圖層進行排序後插入

{

if (gameObjectList[i]->GetLayer() >= obj->GetLayer())//相同圖層 或是 更高的圖層

{

gameObjectList.insert(gameObjectList.begin() + i, obj);//在此位置加入物件

return;

}

}

gameObjectList.push\_back(obj);//還沒有與此物件相同圖層的物件，將此物件加到最尾端

}

void GameSystem::AddUserInterface(UInterface\* ui) {

gameUIList.push\_back(ui);

}

void GameSystem::DeleteGameObject(GameObject \* obj)

{

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

{

if (gameObjectList[i] == obj)

{

delete gameObjectList[i];

gameObjectList.erase(gameObjectList.begin() + i);

return;

}

else

{

i++;

}

}

}

void GameSystem::DeleteUInterface(UInterface\* ui)

{

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

{

if (gameUIList[i] == ui)

{

delete gameUIList[i];

gameUIList.erase(gameUIList.begin() + i);

return;

}

else

{

i++;

}

}

}

void GameSystem::DeleteAllUI()

{

while (!gameUIList.empty())

{

DeleteUInterface(gameUIList[0]);

}

}

void GameSystem::DeleteAllGameObject()

{

while (!gameObjectList.empty())

{

DeleteGameObject(gameObjectList[0]);

}

}

void GameSystem::DeleteAllGameObjectExceptPlayer()

{

int index = 0;

while (gameObjectList.size() != 1)

{

if (gameObjectList[index]->GetTag() == "Player")

{

index++;

}

DeleteGameObject(gameObjectList[index]);

}

}

void GameSystem::CreateFloor(int x, int y, int width, int height)

{

for (int currentY = y; currentY < y + height; currentY += FLOOR\_HEIGHT)//從y座標開始，持續往下增加地板

{

for (int currentX = x; currentX < x + width; currentX += FLOOR\_WIDTH)//從x座標開始，持續往右增加地板

{

AddGameObject(new Floor("Floor", currentX, currentY, FLOOR\_WIDTH, FLOOR\_HEIGHT));

}

}

}

void GameSystem::AllObjectAct()

{

for (auto& i : gameObjectList)

{

if(GameSystem::GetGameObjectWithType<Boss>())

GameSystem::GetGameObjectWithType<Boss>()->GetPhase();

i->Act();

}

//GetGameObjectWithType<Player>()->Act();

//MonstersAct();

}

void GameSystem::SetAllObjectBitMapPosition()//設定所有物件圖片位置

{

for (auto& i : gameObjectList)

{

i->SetBitMapPosition();

}

}

void GameSystem::ShowAllObject()//顯示所有物件

{

for (auto& i : gameObjectList)

{

string k = i->GetTag();

bool i1 = i->GetX() <= Map::GetSX() + SIZE\_X;

bool i2 = i->GetX() + i->GetWidth() >= Map::GetSX();

bool i3 = i->GetY() <= Map::GetSY() + SIZE\_Y;

int iy = i->GetY();

bool i4 = i->GetY() + i->GetHeight() >= Map::GetSY();

if (i->GetX() <= Map::GetSX() + SIZE\_X

&& i->GetX() + i->GetWidth() >= Map::GetSX()

&& i->GetY() <= Map::GetSY() + SIZE\_Y

&& i->GetY() + i->GetHeight() >= Map::GetSY())//只顯示在畫面中的物件

i->ShowBitMap();

}

}

void GameSystem::ShowAllUI()

{

for (auto& i : gameUIList)

{

i->ShowBitMap();

}

}

void GameSystem::ShowText(string text, int left, int top, int right, int bottom, int alignHor, int alignVer, int fontSize, COLORREF textColor, int bkMode, COLORREF bkColor)

{

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

CFont f, \*fp;

f.CreatePointFont(fontSize \* 10, "Times New Roman"); // 產生 font f; 160表示16 point的字

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

pDC->SetBkMode(bkMode);

pDC->SetBkColor(bkColor);

pDC->SetTextColor(textColor);

char str[1000]; // Demo 數字對字串的轉換

sprintf(str, text.c\_str());

CRect rect = { left,top,right,bottom };//設定矩形左、上、右、下的座標

if (alignVer == ALIGN\_TOP)//上

{

if (alignHor == ALIGN\_LEFT)

{

pDC->DrawText(str, rect, DT\_LEFT | DT\_WORDBREAK);//靠左對齊，可換行

}

else if (alignHor == ALIGN\_CENTER)

{

pDC->DrawText(str, rect, DT\_CENTER | DT\_WORDBREAK);//靠中對齊，可換行

}

else if (alignHor == ALIGN\_RIGHT)

{

pDC->DrawText(str, rect, DT\_RIGHT | DT\_WORDBREAK);//靠右對齊，可換行

}

}

else if (alignVer == ALIGN\_CENTER)

{

//這裡在設定垂直置中

CRect temp = rect;

int height = pDC->DrawText(str, temp, DT\_CENTER | DT\_WORDBREAK | DT\_CALCRECT);

rect.top += (rect.Height() - height) / 2;

//rect.DeflateRect(0, (rect.Height() - height) / 2);

if (alignHor == ALIGN\_LEFT)

{

pDC->DrawText(str, rect, DT\_LEFT | DT\_WORDBREAK);//靠左對齊，可換行

}

else if (alignHor == ALIGN\_CENTER)

{

pDC->DrawText(str, rect, DT\_CENTER | DT\_WORDBREAK);

}

else if (alignHor == ALIGN\_RIGHT)

{

pDC->DrawText(str, rect, DT\_RIGHT | DT\_WORDBREAK);//靠右對齊，可換行

}

}

else if (alignVer == ALIGN\_BOTTOM)

{

//這裡在設定垂直置下

CRect temp = rect;

int height = pDC->DrawText(str, temp, DT\_WORDBREAK | DT\_CALCRECT);

rect.top += rect.Height() - height;

//rect.DeflateRect(0, (rect.Height() - height));

if (alignHor == ALIGN\_LEFT)

{

pDC->DrawText(str, rect, DT\_LEFT | DT\_WORDBREAK);//靠左對齊，可換行

}

else if (alignHor == ALIGN\_CENTER)

{

pDC->DrawText(str, rect, DT\_CENTER | DT\_WORDBREAK);

}

else if (alignHor == ALIGN\_RIGHT)

{

pDC->DrawText(str, rect, DT\_RIGHT | DT\_WORDBREAK);//靠右對齊，可換行

}

}

//pDC->DrawText(str, rect, DT\_CENTER | DT\_WORDBREAK);

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

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

}

void GameSystem::DrawRectangle(int x, int y, int width, int height, COLORREF color)

{

CDC \*pDC = CDDraw::GetBackCDC();

CPen \*pp, p(PS\_NULL, 0, RGB(0, 0, 0));

pp = pDC->SelectObject(&p);

CBrush \*pr, r(color);

pr = pDC->SelectObject(&r);

pDC->Rectangle(x, y, x + width, y + height);

pDC->SelectObject(pp);

pDC->SelectObject(pr);

CDDraw::ReleaseBackCDC();

}

void GameSystem::Load()

{

if (isLoaded == false)//還沒載入

{

CAudio::Instance()->Load(AUDIO::AUDIO\_GAME\_INIT, ".\\res\\game\_init.mp3");

CAudio::Instance()->Load(AUDIO::AUDIO\_GAME\_RUN\_1, ".\\res\\game\_run\_1.mp3");

CAudio::Instance()->Load(AUDIO::AUDIO\_GAME\_RUN\_2, ".\\res\\game\_run\_2.mp3");

CAudio::Instance()->Load(AUDIO::AUDIO\_GAME\_OVER\_LOSE, ".\\res\\game\_over\_lose.mp3");

CAudio::Instance()->Load(AUDIO::AUDIO\_GAME\_OVER\_WIN, ".\\res\\game\_over\_win.mp3");

isLoaded = true;

}

}

void GameSystem::Init()

{

isGameOver = false;//重置遊戲狀態

isNextStage = false;

nowStage = 1;

Map::SetSX(0);

Map::SetSY(0);

DeleteAllGameObject();//刪除所有物件

isGameOver = false;

DeleteAllGameObject();

DeleteAllUI();

}

void GameSystem::Exit()

{

DeleteAllGameObject();

DeleteAllUI();

gameObjectList.clear();

gameUIList.clear();

}

void GameSystem::PlayAudio(AUDIO id)

{

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

}

void GameSystem::StopAudio(AUDIO id)

{

CAudio::Instance()->Stop(id);

}

void GameSystem::ChangeToNextStage()

{

DeleteAllGameObjectExceptPlayer();

CreatStage2Object();

SetIsNextStage(false);

}

double GameSystem::Rand(double maxValue)

{

std::random\_device rd;

std::default\_random\_engine gen = std::default\_random\_engine(rd());

std::uniform\_real\_distribution<double> dis(0, maxValue);

auto randFun = std::bind(dis, gen);

return randFun();

}

double GameSystem::Rand(double minValue, double maxValue)

{

std::random\_device rd;

std::default\_random\_engine gen = std::default\_random\_engine(rd());

std::uniform\_real\_distribution<double> dis(minValue, maxValue);

auto randFun = std::bind(dis, gen);

return randFun();

}

void GameSystem::CreatStage1Object()

{

Init();

AddGameObject(new Player("Player", 80, 80, 50, 80));

Player\* player = GetGameObjectWithTag<Player>("Player");

Map::SetSX(player->GetX() - SIZE\_X / 2);

Map::SetSY(player->GetY() - SIZE\_Y / 2);

AddGameObject(new Floor("Floor", 0, Map::WORLD\_SIZE\_Y - 355, Map::WORLD\_SIZE\_X, 355, IDB\_GROUND));

for (int i = 0; i < stage1FloorCount; i++)

{

//CreateFloor(stage1Floor[i][0], stage1Floor[i][1], stage1Floor[i][2], stage1Floor[i][3]);

GameSystem::AddGameObject(new Floor("Floor", stage1Floor[i][0], stage1Floor[i][1], stage1Floor[i][2], stage1Floor[i][3]));

}

AddGameObject(new Treasure("Treasure", 400, 1388, 250, 187, 10,100, 0, 10));

AddGameObject(new Treasure("Treasure", 3580, 420, 250, 187, 10, 100, 0, 10));

AddGameObject(new Door("Door", 530, 480, 12, 120));//門

AddGameObject(new Door("Door", 1315, 260, 12, 120));//門

AddGameObject(new Door("Door", 2330, 1445, 12, 120));//門

AddGameObject(new Goal("Goal", 3510, 1353, 143, 212));

AddGameObject(new Portal("Portal", 1750, 1445, 80, 120, 1));

AddGameObject(new Portal("Portal", 3400, 480, 80, 120, 99));

AddGameObject(new Demon("Monster", 400, 200, 110, 158));

AddGameObject(new Demon("Monster", 900, 1380, 110, 158));

AddGameObject(new Demon("Monster", 800, 1380, 110, 158));

AddGameObject(new Demon("Monster", 1400, 1010, 110, 158));

AddGameObject(new Demon("Monster", 1200, 415, 110, 158));

AddGameObject(new Demon("Monster", 1425, 10, 110, 158));

AddGameObject(new Demon("Monster", 1920, 200, 110, 158));

AddGameObject(new Demon("Monster", 2150, 1370, 110, 158));

AddGameObject(new Demon("Monster", 2700, 1290, 110, 158));

AddGameObject(new Demon("Monster", 2950, 1290, 110, 158));

AddGameObject(new Demon("Monster", 2200, 40, 110, 158));

AddUserInterface(new EquipedSlot("EquipedSlot", SIZE\_X / 10 - 100, SIZE\_Y \* 9 / 10, 64, 60, 0));

AddUserInterface(new EquipedSlot("EquipedSlot", SIZE\_X / 10 - 10, SIZE\_Y \* 9 / 10, 64, 60, 1));

AddUserInterface(new UIBlood("UIBlood", 0, SIZE\_Y - 30, 400, 30));

AddUserInterface(new UIMonsterBlood("UIMonster", 0, 0, 0, 0));

Map::SetStaticObject();

}

void GameSystem::CreatStage2Object()

{

AddGameObject(new Boss("Monster", SIZE\_X - 200, Map::WORLD\_SIZE\_Y - 100 - 280, 180, 280));

AddGameObject(new Floor("Floor", 0, Map::WORLD\_SIZE\_Y - 100, Map::WORLD\_SIZE\_X, 100, IDB\_GROUND));//地圖最下方的地板

AddGameObject(new Goal("Goal", 150, Map::WORLD\_SIZE\_Y - 312, 143, 212));

Map::Init();

Map::SetStaticObject();

}

Goal.h

#pragma once

#include "GameObject.h"

#include "GameSystem.h"

class Goal:public GameObject

{

public:

Goal(string tag, int x, int y, int width, int height);

~Goal();

void Picked();

private:

};

Goal.cpp

#include "StdAfx.h"

#include "Goal.h"

Goal::Goal(string tag, int x, int y, int width, int height):GameObject(tag, x, y, width, height)

{

LoadBitMap(".\\res\\goal.bmp");

layer = GameSystem::LAYER::LAYER\_FLOOR;

SetBitMapPosition();

}

Goal::~Goal()

{

}

void Goal::Picked()

{

if (GameSystem::GetNowStage() <= 1)//只有第一關的終點有作用

{

GameSystem::SetIsNextStage(true);

GameSystem::StagePlusOne();

}

}

Item.h

#pragma once

//Item

//道具

//繼承GameObject

#include "GameSystem.h"

#include "GameObject.h"

class Item :public GameObject

{

public:

Item();

virtual ~Item();

Item(string tag, int x, int y, int width, int height, int pictureID);

Item(string tag, int x, int y, int width, int height); //沒有pictureID 的 Constructor

virtual void Picked(); //被撿起

virtual void ShowBitMap(); //顯示動畫

virtual void Act();

void Fall(int perDisplacement); //噴裝備動畫

//virtual void SetXY(int hostX, int hostY, int playerCurrentAni, int playerAniNumber); //隨著擁有者移動 設定XY

protected:

//bool IsPlayerInRange(GameObject\* obj, int right\_fix, int left\_fix, int up\_fix, int down\_fix);//判斷動畫是否即擊中

void AddAniBitMaps(char\* pic[], int aniType, int picCount); //增加多張動畫圖片

void AddAniBitMap(char\* pic, int aniType); //增加動畫圖片

bool CanMoveDown(int perDisplacement); //物件是否可往下一

virtual void LoadAni();

virtual void Dead()override;

void SetBitMapPosition()override;

int currentAni = ANI\_IDLE; //IDLE 動畫

int perDisplacement; //掉落速度;

vector<CAnimation\*> ani;//動畫

enum ANI {

ANI\_IDLE = 0,

};

};

Item.cpp

#include "StdAfx.h"

#include "Item.h"

Item::Item()

{

tag = "Item";

}

Item::~Item()

{

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

{

delete ani[i];

}

ani.clear();

}

Item::Item(string tag, int x, int y, int width, int height, int pictureID) :GameObject(tag, x, y, width, height, pictureID)

{

tag = "Item";

layer = GameSystem::LAYER::LAYER\_ITEM;

while (Map::HasObject(this->x, this->y + height))//若道具位在地板中

this->y--;//將道具往上移至地板外

while (!Map::HasObject(this->x, this->y + height + 1))//若道具在空中

this->y++;//將道具往下移至地板外

perDisplacement =1;

}

Item::Item(string tag, int x, int y, int width, int height) :GameObject(tag, x, y, width, height)

{

tag = "Item";

layer = GameSystem::LAYER::LAYER\_ITEM;

while (Map::HasObject(this->x, this->y + height))//若道具位在地板中

this->y--;//將道具往上移至地板外

//while (!Map::HasObject(this->x, this->y + height + 1))//若道具在空中

//this->y++;//將道具往下移至地板外

ani.push\_back(new CAnimation());//加入一個動畫

perDisplacement = -4;

}

void Item::AddAniBitMaps(char \* pic[], int aniType, int picCount)

{

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

{

AddAniBitMap(pic[i], aniType);

}

}

void Item::AddAniBitMap(char\* pic, int aniType)

{

while ((int)ani.size() <= aniType)//vector的大小不夠大

ani.push\_back(new CAnimation);//增加大小

ani[aniType]->AddBitmap(pic, RGB(255, 255, 255));

}

void Item::LoadAni(){}

void Item::ShowBitMap() {

ani[currentAni]->OnShow();

}

void Item::Fall(int perDisplacement)

{

int k = GetY();

if (CanMoveDown(perDisplacement))//如果腳下沒東西

{

SetY(GetY() +perDisplacement);

this->perDisplacement++;

}

else

{

while (CanMoveDown(1))//再繼續用下降位移量下降，將會卡進地板，所以一次向下位移1進行微調0

SetY(GetY()+1);

}

}

bool Item::CanMoveDown(int perDisplacement)

{

bool canMoveDown = true;

for (int i = x; i < x + width; i++)

{

if (Map::HasObject(i, y + height + perDisplacement))//下面有東西

{

canMoveDown = false;

return canMoveDown;

}

}

return canMoveDown;

}

void Item::SetBitMapPosition()

{

for (auto& i : ani)

{

i->SetTopLeft(this->x - Map::GetSX(), this->y - Map::GetSY());

}

}

void Item::Act()

{

Fall(this->perDisplacement);

}

void Item::Picked()

{

Dead();

}

void Item::Dead()

{

GameSystem::DeleteGameObject(this);//刪除此物件

}

ItemTraditionalSword.h

#pragma once

#include "StdAfx.h"

#include "ItemWeapon.h"

#include "SwordWave.h"

#include "Player.h"

#include "EquipedSlot.h"

class ItemTraditionalSword:public ItemWeapon {

public:

ItemTraditionalSword();

~ItemTraditionalSword();

ItemTraditionalSword(string tag, int x, int y, int width, int height);

ItemTraditionalSword(string tag, int x, int y, int width, int height,int damage, int defense);

private:

void LoadAni()override; //載入動畫

void Picked()override; //出發效果

};

ItemTraditionalSword.cpp

#include"StdAfx.h"

#include "ItemTraditionalSword.h"

#include "PlayerTradationalSword.h"

ItemTraditionalSword::ItemTraditionalSword() {

tag = "ItemWeapon";

}

ItemTraditionalSword::~ItemTraditionalSword()

{

}

ItemTraditionalSword::ItemTraditionalSword(string tag, int x, int y, int width, int height) :ItemWeapon(tag, x, y, width, height) {

tag = "ItemWeapon";

attackDamage = 100;

attackRange = 100;

defense = 2;

LoadAni();

LoadBitMap(".\\res\\sword\_01d.bmp");

}

ItemTraditionalSword::ItemTraditionalSword(string tag, int x, int y, int width, int height,int damage, int defense) :ItemWeapon(tag, x, y, width, height) {

tag = "ItemWeapon";

this->attackDamage = damage;

attackRange = 100;

this->defense = defense;

LoadAni();

LoadBitMap(".\\res\\sword\_01d.bmp");

}

void ItemTraditionalSword::LoadAni() {

char\* aniIdle[1] = { ".\\res\\sword\_01d.bmp" };

AddAniBitMaps(aniIdle, ANI\_IDLE, 1);

}

void ItemTraditionalSword::Picked() {

Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player");

//家佑測試的

player->AddEquipment<PlayerTraditionalSword>(new PlayerTraditionalSword(this));

//--------------UI設定--------------------------//

vector<EquipedSlot\*> equipedSlots = GameSystem::GetUInterfacesWithTag<EquipedSlot>("EquipedSlot");

for (auto& i : equipedSlots)

{

if (i->GetIsEquipmentPicSet() == false)

{

i->SetEquipmentPicture(".\\res\\sword\_01d.bmp", i->GetX() + 15, i->GetY() + 8);

break;

}

else if (i->GetIsEquipmentPicSet() == true && player->GetPickCount() > 2

&&

((player->CurrentWeapon() == 1 && i->GetID() == 0)

||

(player->CurrentWeapon() == 2 && i->GetID() == 1))

)

{

i->SetEquipmentPicture(".\\res\\sword\_01d.bmp", i->GetX() + 15, i->GetY() + 8);

break;

}

}

Dead();

}

ItemWeapon.h

#pragma once

//Weapon

//場上的武器道具

//繼承Item

//目前virtual funtion(LoadAni、Picked、ShowBitMap、SetXY)

#include "Item.h"

class ItemWeapon :public Item

{

friend class PlayerEquipment;

public:

ItemWeapon();

~ItemWeapon();

ItemWeapon(string tag, int x, int y, int width, int height);

int GetAttackDamage();

int GetAttackRange();

int GetDefense();

protected:

int attackDamage; //攻擊力

int attackRange; //攻擊範圍

int defense; //防禦力

int haveEffect; //是否 有特效;

};

ItemWeapon.cpp

#include "StdAfx.h"

#include "ItemWeapon.h"

ItemWeapon::ItemWeapon()

{

tag = "ItemWeapon";

}

ItemWeapon::~ItemWeapon()

{

}

ItemWeapon::ItemWeapon(string tag, int x, int y, int width, int height):Item(tag, x, y, width, height){

tag = "ItemWeapon";

}

int ItemWeapon::GetAttackDamage()

{

return attackDamage;

}

int ItemWeapon::GetAttackRange()

{

return attackRange;

}

int ItemWeapon::GetDefense()

{

return defense;

}

ItemWeaponBow.h

#pragma once

#include"ItemWeapon.h"

#include "EquipedSlot.h"

class ItemWeaponBow :public ItemWeapon

{

public:

ItemWeaponBow(string tag, int x, int y, int width, int hegiht);

ItemWeaponBow(string tag, int x, int y, int width, int hegiht,int damage, int defense);

~ItemWeaponBow();

private:

void Picked()override; //出發效果

void LoadAni()override; //載入動畫

};

ItemWeaponBow.cpp

#include "StdAfx.h"

#include "ItemWeaponBow.h"

#include "PlayerBow.h"

ItemWeaponBow::ItemWeaponBow(string tag, int x, int y, int width, int height) :ItemWeapon(tag, x, y, width, height)

{

tag = "ItemWeapon";

attackRange = 100;

this->attackDamage = 50;

this->defense = 0;

LoadAni();

LoadBitMap(".\\res\\bow.bmp");

}

ItemWeaponBow::ItemWeaponBow(string tag, int x, int y, int width, int height, int damage, int defense) :ItemWeapon(tag, x, y, width, height)

{

tag = "ItemWeapon";

attackRange = 100;

this->attackDamage = damage;

this->defense = defense;

LoadAni();

LoadBitMap(".\\res\\bow.bmp");

}

ItemWeaponBow::~ItemWeaponBow()

{

}

void ItemWeaponBow::Picked() {

Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player");

//家佑測試的

player->AddEquipment<PlayerBow>(new PlayerBow(this));

//--------------UI設定--------------------------//

vector<EquipedSlot\*> equipedSlots = GameSystem::GetUInterfacesWithTag<EquipedSlot>("EquipedSlot");

for (auto& i : equipedSlots)

{

if (i->GetIsEquipmentPicSet() == false)

{

i->SetEquipmentPicture(".\\res\\bow.bmp", i->GetX() + 15, i->GetY() + 8);

break;

}

else if (i->GetIsEquipmentPicSet() == true && player->GetPickCount() > 2

&&

((player->CurrentWeapon() == 1 && i->GetID() == 0)

||

(player->CurrentWeapon() == 2 && i->GetID() == 1))

)

{

i->SetEquipmentPicture(".\\res\\bow.bmp", i->GetX() + 15, i->GetY() + 8);

break;

}

}

Dead();

}

void ItemWeaponBow::LoadAni()

{

char\* aniIdle[1] = { ".\\res\\bow.bmp" };

AddAniBitMaps(aniIdle, ANI\_IDLE, 1);

}

Map.h

#pragma once

//Map

//地圖

//座標系統管理

#include "gamelib.h"

using namespace game\_framework;

class Map

{

public:

Map();

~Map();

static void Init();//初始化

static void Load();//載入地圖

static void ShowStartMenu();//顯示開始選單

static void ShowBackgroundPic();//顯示背景圖片

static int GetSX();//取得螢幕左上角X座標

static int GetSY();//取得螢幕左上角Y座標

static void SetSX(int x);//設定螢幕左上角X座標

static void SetSY(int y);//設定螢幕左上角Y座標

static void MoveScreenTopLeft(int dx, int dy);//移動螢幕左上角位置

static void SetStaticObject();//設定不會移動的物件座標，只會在遊戲一開始呼叫

static bool HasObject(int x, int y);//該座標是否有物件

static const int WORLD\_SIZE\_X = 6 \* 640, WORLD\_SIZE\_Y = 4 \* 480;//整個遊戲地圖的長寬

private:

static bool coordinate[WORLD\_SIZE\_X][WORLD\_SIZE\_Y];//紀錄哪個座標有物件

static int sx, sy;//螢幕左上角的座標在地圖座標的哪裡

static CMovingBitmap startMenu;//開始選單

static CMovingBitmap backgroundPic;//背景圖片

static bool isLoaded;//是否已載入

};

Map.cpp

#include "StdAfx.h"

#include "gamelib.h"

using namespace game\_framework;

#include "GameSystem.h"

#include "Map.h"

CMovingBitmap Map::startMenu;//開始選單

CMovingBitmap Map::backgroundPic;//背景圖片

bool Map::coordinate[WORLD\_SIZE\_X][WORLD\_SIZE\_Y] = { false };//紀錄哪個座標有物件

int Map::sx = 0;//預設螢幕左上角在地圖座標的0,0

int Map::sy = 0;//預設螢幕左上角在地圖座標的0,0

bool Map::isLoaded = false;//是否已載入

Map::~Map()

{

}

void Map::Init()

{

for (int i = 0; i < WORLD\_SIZE\_X; i++)

{

for (int j = 0; j < WORLD\_SIZE\_Y; j++)

{

coordinate[i][j] = false;

}

}

}

void Map::Load()

{

if (isLoaded == false)

{

startMenu.LoadBitmap(".\\res\\start\_menu.bmp");

backgroundPic.LoadBitmap(IDB\_BACKGROUND);

isLoaded = true;

}

}

void Map::ShowStartMenu()

{

startMenu.SetTopLeft(0, 0);

startMenu.ShowBitmap();

}

void Map::ShowBackgroundPic()

{

backgroundPic.SetTopLeft(-sx, -sy);

backgroundPic.ShowBitmap();

}

void Map::SetSX(int x)

{

if (x < 0)//確保螢幕在遊戲畫面中

x = 0;

if (x + SIZE\_X >= WORLD\_SIZE\_X)//確保螢幕在遊戲畫面中

x = WORLD\_SIZE\_X - SIZE\_X - 1;

sx = x;

}

void Map::SetSY(int y)

{

if (y < 0)//確保螢幕在遊戲畫面中

y = 0;

if (y + SIZE\_Y >= WORLD\_SIZE\_Y)//確保螢幕在遊戲畫面中

y = WORLD\_SIZE\_Y - SIZE\_Y - 1;

sy = y;

}

int Map::GetSX()

{

return sx;

}

int Map::GetSY()

{

return sy;

}

void Map::MoveScreenTopLeft(int dx, int dy)

{

Map::SetSX(Map::GetSX() + dx);

Map::SetSY(Map::GetSY() + dy);

}

void Map::SetStaticObject()//設定不會移動的物件座標，只會在遊戲一開始呼叫

{

for (auto& i : GameSystem::GetAllGameObject())

{

if (i->GetTag() == "Floor")//不記錄會動的物件

{

for (int x = i->GetX(); x < i->GetX() + i->GetWidth(); x++)

{

for (int y = i->GetY(); y < i->GetY() + i->GetHeight(); y++)

{

if (x >= 0 && x < Map::WORLD\_SIZE\_X && y >= 0 && y < Map::WORLD\_SIZE\_Y)//在地圖範圍內才紀錄

coordinate[x][y] = true;//物件的長寬範圍以內的座標都設為true

}

}

}

}

}

bool Map::HasObject(int x, int y)//該座標是否有物件

{

return coordinate[x][y];//回傳該座標是否有物件

}

Monster.h

#pragma once

//Monster

//怪物

//繼承Character

#include "Character.h"

#include "GameSystem.h"

#include "Player.h"

class Monster :public Character

{

public:

Monster();

~Monster();

Monster(string tag, int x, int y, int width, int height);

void SetDefenseRange(int defenseRange); //設定來回走動距離

void SetRL(int rl); //設定往左往右

void SetdX(); //設定水平防守邊界

void SetdY(); //設定垂直防守邊界

void SetAttackField(int attackField); //設定觸發跟隨的距離

void SetAttackRange(int attackRange); //設定攻擊距離;

int GetDefenseRange(); //取得來回走動距離(從中間到兩邊)

int GetRL(); //取得現在是往右移動還是往左移動

int GetdX(int point); //GetdX(0)取得左邊防守邊線, GetdX(1)取得右邊防守邊線

int GetStatus(); //取得怪物現在狀態 STANDBY:0, LEFT:1, RIGTH:2, ATTACK = 3

//----------------移動相關---------------------//

virtual void AutoMove(); //在防守區域內來回走動

void Move(int dx, int dy)override;

void Fall(int perDisplacement); //落下

void Up(); //怪物走上地形

//----------------攻擊相關---------------------//

bool IsInAttackField(int playerX, int playerY, int right\_fix, int left\_fix, int up\_fix, int down\_fix); //Player是否在AttackField內, 是的話，衝向主角

bool IsPlayerInRange(Player\* player, int right\_fix, int left\_fix, int up\_fix, int down\_fix); //兩者的圖形是否有觸碰到

int PlaceRelativePlayer(Player\* player); //怪物相對於腳色的位置

virtual void Attack();//當Player進入防守區，攻擊他

void PlayerAttack(int damage);

//-----------------Animation------------------///

void ShowBitMap()override;

int GetAttackAniNumber();

void Act()override; //組合各種動作(ISATTACK, AUTOMOVE, ATTACK)

protected:

static const int LEFT = 1, RIGHT = 2, STANDBY = 0, ATTACK = 3, FALL = 4, ISATTACK=5; //怪物的狀態

int status ; //現在狀態

Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player"); //常用到就先記錄下來

int fallDisplacement = 0;

int currentAni = 0; //現在執行得動畫

enum ANI

{

ANI\_IDLE = 0, //原地

ANI\_LEFT, //左移動動畫

ANI\_RIGHT, //右移動動畫

ANI\_ATTACK\_LEFT, //左邊攻擊動畫

ANI\_ATTACK\_RIGHT, //右邊攻擊動畫

ANI\_ISATTACK\_RIGHT, //被攻擊右邊動畫

ANI\_ISATTACK\_LEFT //被攻擊左邊動畫

};

private:

int dX[2], dY[2]; //0左1右、0上1下

int rl; //現在往哪邊走

int defenseRange, attackField; //defenseRange 來回走動的範圍, attackfield衝過去攻擊主角的範圍

void Dead()override; //死亡

};

Monster.cpp

#pragma once

#include "StdAfx.h"

#include <ctime>

#include "Monster.h"

#include "Player.h"

#include "Item.h"

#include "Potion.h"

#include "ItemTraditionalSword.h"

#include "ItemWeaponBow.h"

Monster::Monster()

{

tag = "Monster";

}

Monster::~Monster()

{

}

Monster::Monster(string tag, int x, int y, int width, int height) :Character(tag, x, y, width, height)

{

tag = "Monster";

rl = LEFT;

layer = GameSystem::LAYER::LAYER\_MONSTER;

}

void Monster::SetDefenseRange(int defenseRange) {

this->defenseRange = defenseRange;

}

void Monster::SetRL(int rl) {

this->rl = rl;

}

void Monster::SetdX() {

dX[0] = x - defenseRange; //左邊

dX[1] = x + defenseRange; //右邊

}

void Monster::SetdY() {

this->dY[0] = y - defenseRange; //下

this->dY[1] = y + defenseRange; //上

}

void Monster::SetAttackField(int attackField){

this->attackField = attackField;

}

void Monster::SetAttackRange(int attackRange) {

this->attackRange = attackRange;

}

//-----------------------------------GET------------------------------------------------//

int Monster::GetDefenseRange() {

return defenseRange;

}

int Monster::GetRL() {

return rl;

}

int Monster::GetdX(int point) {

if (point == 0)

return dX[0];

else

return dX[1];

}

int Monster::GetStatus() {

return status;

}

//--------------------------OTHER FUNCTION-----------------------------------//

void Monster::AutoMove() {}

void Monster::Move(int dx, int dy){

this->x += dx;

this->y += dy;

}

void Monster::Fall(int perDisplacement)

{

if (CanMoveDown(fallDisplacement))//如果腳下沒東西

{

fallDisplacement++;

Move(0, fallDisplacement);

}

else

{

while (CanMoveDown(1))//再繼續用下降位移量下降，將會卡進地板，所以一次向下位移1進行微調0

Move(0, 1);

fallDisplacement = 0;

dX[0] = x - defenseRange;

dX[1] = x + defenseRange;

}

}

void Monster::Up()

{

while (Map::HasObject(x + width, y + height)) {

y-=1;

}

}

bool Monster::IsInAttackField(int playerX, int playerY, int right\_fix, int left\_fix, int up\_fix, int down\_fix) {

int attackXField\_Left = x - attackField - left\_fix, attackXField\_Right = x + attackField + right\_fix;

int attackYField\_down = y + attackField + down\_fix, attackYField\_up = y - attackField - up\_fix;

if (playerX >= attackXField\_Left && playerX <= attackXField\_Right &&

playerY >= attackYField\_up && playerY <= attackYField\_down)

return true;

return false;

}

bool Monster::IsPlayerInRange(Player\* player, int right\_fix, int left\_fix, int up\_fix, int down\_fix) {

int RIGHT\_EDGE = x + width + right\_fix, LEFT\_EDGE = x - left\_fix,

UP\_EDGE = y - up\_fix, DOWN\_EDGE = y + height + down\_fix;

int OB\_X = player->GetX(), OB\_Y = player->GetY(), OB\_WIDTH = player->GetWidth(), OB\_HEIGHT = player->GetHeight();

int OB\_RIGHT\_EDGE = OB\_X + OB\_WIDTH , OB\_LEFT\_EDGE = OB\_X,

OB\_UP\_EDGE = OB\_Y, OB\_DOWN\_EDGE = OB\_Y + OB\_HEIGHT;

if (OB\_RIGHT\_EDGE >= LEFT\_EDGE - attackRange && PlaceRelativePlayer(player) == RIGHT && OB\_UP\_EDGE> UP\_EDGE && OB\_UP\_EDGE<DOWN\_EDGE) //人在左, 怪物在右

return true;

else if (OB\_LEFT\_EDGE <= RIGHT\_EDGE +attackRange && PlaceRelativePlayer(player) == LEFT && OB\_UP\_EDGE> UP\_EDGE && OB\_UP\_EDGE<DOWN\_EDGE) //人在右, 怪物在左

return true;

return false;

}

int Monster::PlaceRelativePlayer(Player\* player) {

if (x >= player->GetX())

return RIGHT;

else

return LEFT;

}

void Monster::PlayerAttack(int damage)

{

DecreaseHP(damage);

status = ISATTACK;

}

void Monster::ShowBitMap()

{

ani[currentAni]->OnMove();

ani[currentAni]->OnShow();

}

void Monster::Dead()

{

int i = static\_cast<int>(GameSystem::Rand(10));

int damage = static\_cast<int>(GameSystem::Rand(30,60));

int defense = static\_cast<int>(GameSystem::Rand(5, 15));

if(i<=1)

GameSystem::AddGameObject(new Potion("Potion", x , y, 48, 48));//產生一個細胞道具

else if(i <= 5)

GameSystem::AddGameObject(new ItemTraditionalSword("ItemWeapon", x + width / 2, y, 32, 32, damage, defense));

else if (i <= 8)

GameSystem::AddGameObject(new ItemWeaponBow("ItemWeapomBow", x + width / 2, y, 32, 32, damage, defense));

GameSystem::DeleteGameObject(this);

}

int Monster::GetAttackAniNumber() {

if(currentAni == ANI::ANI\_ATTACK\_LEFT || currentAni ==ANI::ANI\_ATTACK\_RIGHT)

return ani[currentAni]->GetCurrentBitmapNumber();

return 0;

}

void Monster::Attack() {}

void Monster::Act(){}

mygame.h

#include "Button.h"

namespace game\_framework {

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

// Constants

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

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

// 這個class為遊戲的遊戲開頭畫面物件

// 每個Member function的Implementation都要弄懂

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

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

~CGameStateInit();

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

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

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

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

void Load();//載入

void Dead();//離開遊戲時用到

bool isLoaded = false;//是否已載入

vector<Button\*> buttonList;//按鈕列表

bool isShowOption = false;//是否顯示選單

};

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

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

// 每個Member function的Implementation都要弄懂

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

class CGameStateRun : public CGameState {

public:

CGameStateRun(CGame \*g);

~CGameStateRun();

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit(); // 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

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

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

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

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

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

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

};

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

// 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 originCountDown = 3 \* 30;

int countDown = 3\*30;

};

}

mygame.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "mygame.h"

#include "Map.h"

#include "GameSystem.h"

#include "GameObject.h"

#include "Character.h"

#include "Player.h"

#include "Monster.h"

#include "Floor.h"

#include "Item.h"

#include "Fire.h"

#include "Demon.h"

#include "EquipedSlot.h"

#include "UIBlood.h"

#include "UIMonsterBlood.h"

#include "Door.h"

#include "Goal.h"

#include "Boss.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)

{

}

CGameStateInit::~CGameStateInit()

{

Dead();

}

void CGameStateInit::OnInit()

{

//不要在這用

}

void CGameStateInit::OnBeginState()

{

Load();//載入

GameSystem::CreatStage1Object();

GameSystem::StopAudio(GameSystem::AUDIO\_GAME\_OVER\_WIN);//停止遊戲結束的音樂

GameSystem::StopAudio(GameSystem::AUDIO\_GAME\_OVER\_LOSE);//停止遊戲結束的音樂

GameSystem::PlayAudio(GameSystem::AUDIO\_GAME\_INIT);//播放遊戲結束的音樂

}

void CGameStateInit::Load()

{

if (isLoaded == false)

{

//OnInit

GameSystem::Load();

Map::Load();

//

buttonList.push\_back(new Button("ButtonStart", SIZE\_X \* 3 / 4, SIZE\_Y / 2 - 50, 100, 50, ButtonOnClickEvent::ON\_CLICK\_START));

buttonList.push\_back(new Button("ButtonOption", SIZE\_X \* 3 / 4, SIZE\_Y / 2 + 50, 100, 50, ButtonOnClickEvent::ON\_CLICK\_OPTION));

buttonList.push\_back(new Button("ButtonExit", SIZE\_X \* 3 / 4, SIZE\_Y / 2 + 150, 200, 50, ButtonOnClickEvent::ON\_CLICK\_EXIT));

isLoaded = true;

}

}

void CGameStateInit::Dead()

{

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

{

delete buttonList[i];

i++;

}

}

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

{

//GotoGameState(GAME\_STATE\_RUN);

}

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

{

if (isShowOption)//開著選單時，按下滑鼠左鍵可關閉

{

isShowOption = false;

return;

}

for (auto& i : buttonList)

{

if (point.x > i->GetX() && point.x < i->GetX() + i->GetWidth()

&& point.y > i->GetY() && point.y < i->GetY() + i->GetHeight())//滑鼠在按鈕上

{

if (i->GetTag() == "ButtonOption")//按下選項

{

ButtonOnClickEvent::LoadOptionAni();

isShowOption = true;

}

i->OnClick(this->game);//執行按鈕事件

}

}

}

void CGameStateInit::OnShow()

{

Map::ShowStartMenu();

for (auto& i : buttonList)

{

i->SetBitMapPosition();

i->ShowBitMap();

}

if (isShowOption)//顯示選項

{

ButtonOnClickEvent::OnClickOption(this->game);

}

}

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

// 這個class為遊戲的結束狀態(Game Over)

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

CGameStateOver::CGameStateOver(CGame \*g)

: CGameState(g)

{

}

void CGameStateOver::OnMove()

{

countDown--;

if (countDown <= 0)

{

GotoGameState(GAME\_STATE\_INIT);

}

}

void CGameStateOver::OnBeginState()

{

countDown = 3 \* 30;

//GameSystem::StopAudio(GameSystem::AUDIO::AUDIO\_GAME\_OVER);//停止遊戲結束的音樂

//GameSystem::PlayAudio(GameSystem::AUDIO::AUDIO\_GAME\_INIT);//播放遊戲開始的音樂

GameSystem::StopAudio(GameSystem::AUDIO::AUDIO\_GAME\_RUN\_1);//停止遊戲中的音樂

GameSystem::StopAudio(GameSystem::AUDIO::AUDIO\_GAME\_RUN\_2);//停止遊戲中的音樂

if (GameSystem::GetOverMode() == GameSystem::OVER\_LOSE)

GameSystem::PlayAudio(GameSystem::AUDIO::AUDIO\_GAME\_OVER\_LOSE);//播放失敗的音樂

else

GameSystem::PlayAudio(GameSystem::AUDIO::AUDIO\_GAME\_OVER\_WIN);//播放勝利的音樂

}

void CGameStateOver::OnInit()

{

}

void CGameStateOver::OnShow()

{

string text;

if (GameSystem::GetOverMode() == GameSystem::OVER\_LOSE)

{

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

{

text += "G";

}

GameSystem::ShowText(text, 0, 0, SIZE\_X, SIZE\_Y, GameSystem::ALIGN\_CENTER, GameSystem::ALIGN\_CENTER, 16

, RGB(255 \* (originCountDown - countDown) / originCountDown, 255 \* (originCountDown - countDown) / originCountDown, 255 \* (originCountDown - countDown) / originCountDown));

}

else if (GameSystem::GetOverMode() == GameSystem::OVER\_WIN)

{

text += "太神啦";

int r, g, b;

r = static\_cast<int>(GameSystem::Rand(210, 250));

g = static\_cast<int>(GameSystem::Rand(210, 250));

b = static\_cast<int>(GameSystem::Rand(210, 250));

GameSystem::DrawRectangle(0, 0, SIZE\_X, SIZE\_Y, RGB(r, g, b));

GameSystem::ShowText(text, 0, 0, SIZE\_X, SIZE\_Y, GameSystem::ALIGN\_CENTER, GameSystem::ALIGN\_CENTER, 16, RGB(255 - r, 255 - g, 255 - b));

}

if (countDown <= 0)

countDown = originCountDown;

}

CGameStateRun::CGameStateRun(CGame \*g)

: CGameState(g)

{

}

CGameStateRun::~CGameStateRun()

{

GameSystem::Exit();//離開遊戲

}

void CGameStateRun::OnBeginState()

{

Player& player = \*(GameSystem::GetGameObjectWithTag<Player>("Player"));

player.AdjustPositionOnBegin();

//GameSystem::SetIsNextStage(true);

if (GameSystem::IsNextStage()) {

ShowInitProgress(30); //換關卡的Loading

Sleep(300); //Delay

GameSystem::ChangeToNextStage();

ShowInitProgress(50);

Sleep(300);

player.MoveTo(GameSystem::GetGameObjectWithTag<Goal>("Goal")->GetX()

, GameSystem::GetGameObjectWithTag<Goal>("Goal")->GetY());//移動至傳送門的位置

GameSystem::StopAudio(GameSystem::AUDIO::AUDIO\_GAME\_RUN\_1);//停止第一關的音樂

GameSystem::PlayAudio(GameSystem::AUDIO::AUDIO\_GAME\_RUN\_2);//播放第二關的音樂

}

else

{

GameSystem::StopAudio(GameSystem::AUDIO::AUDIO\_GAME\_RUN\_2);//停止第二關的音樂

GameSystem::PlayAudio(GameSystem::AUDIO::AUDIO\_GAME\_RUN\_1);//播放第一關的音樂

}

GameSystem::StopAudio(GameSystem::AUDIO::AUDIO\_GAME\_INIT);//停止結束的音樂

ShowInitProgress(100);

}

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

{

if (GameSystem::IsGameOver())//遊戲結束

GotoGameState(GAME\_STATE\_OVER);//跳至遊戲結束狀態

if (GameSystem::IsNextStage())

GotoGameState(GAME\_STATE\_RUN);

GameSystem::AllObjectAct();

GameSystem::SetAllObjectBitMapPosition();//設定所有物件圖片位置

}

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

{

}

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\_SPACE = 0x20;//空白鍵

const char KEY\_ATTACK = 0x58;//攻擊鍵X鍵

const char KEY\_INTERACT = 0x43;//互動鍵c鍵

const char KEY\_ROLL = 0x5a;//翻滾鍵Z鍵

const char KEY\_F = 0x46;//F鍵

const char KEY\_G = 0x47;//G鍵

const char KEY\_1 = 0x31;//1鍵

const char KEY\_2 = 0x32;//2鍵

Player& player = \*(GameSystem::GetGameObjectWithTag<Player>("Player"));//宣告一個玩家，避免每次都要打一長串GetGameObject...

if (nChar == KEY\_F)//測試用，按下時玩家加血

{

player.SetHP(player.GetHP() + 100);

}

if (nChar == KEY\_G)//測試用，按下時BOSS扣血

{

Boss\* boss = GameSystem::GetGameObjectWithType<Boss>();

if (boss)//有boss才扣血

boss->DecreaseHP(1000);

}

if (nChar == KEY\_LEFT)

{

player.SetIsMoveLeft(true);

}

if (nChar == KEY\_RIGHT)

{

player.SetIsMoveRight(true);

}

if (nChar == KEY\_UP)

{

//player.SetIsJump(true);

}

if (nChar == KEY\_DOWN)

{

player.SetIsSquat(true);

}

if (nChar == KEY\_SPACE)

{

player.SetIsJump(true);

//player.SetIsDownJump(true);

}

if (nChar == KEY\_ATTACK)

{

player.SetIsAttack(true);

}

if (nChar == KEY\_INTERACT)

{

player.Interact();

}

if (nChar == KEY\_ROLL)

{

player.SetIsRoll(true);

}

if (nChar == KEY\_1)//換武器

{

player.ChangeWeapon(1);

}

if (nChar == KEY\_2)//換武器

{

player.ChangeWeapon(2);

}

}

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

Player& player = \*(GameSystem::GetGameObjectWithTag<Player>("Player"));//宣告一個玩家，避免每次都要打一長串GetGameObject...

if (nChar == KEY\_LEFT)

{

player.SetIsMoveLeft(false);

}

if (nChar == KEY\_RIGHT)

{

player.SetIsMoveRight(false);

}

if (nChar == KEY\_UP)

{

player.SetIsJump(false);

}

if (nChar == KEY\_DOWN)

{

player.SetIsSquat(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

{

}

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()

{

Map::ShowBackgroundPic();

GameSystem::ShowAllObject();

GameSystem::ShowAllUI();

Player& player = \*(GameSystem::GetGameObjectWithTag<Player>("Player"));

}

}

Player.h

#pragma once

//Player

//玩家

//繼承Character

#include "StdAfx.h"

#include "Character.h"

#include "Item.h"

#include "ItemWeapon.h"

#include "PlayerEquipment.h"

#include "GameSystem.h"

class Player :public Character

{

public:

Player();

~Player();

Player(string tag, int x, int y, int width, int height);

void AdjustPositionOnBegin();//載入遊戲時微調玩家位置，避免玩家跟地板重疊

void DecreaseHP(int dhp);

//----------------按鍵相關---------------------//

void SetIsMoveLeft(bool isMoveLeft);//設定是否向左移動

void SetIsMoveRight(bool isMoveRight);//設定是否向右移動

void SetIsGrounded(bool isGrounded);//設定是否在地上

void SetIsJump(bool isJump);//設定是否按下跳躍

void SetIsDownJump(bool isDownJump);//設定是否下跳

void SetIsAttack(bool isAttack);//設定是否按下攻擊

void SetIsRoll(bool isRoll);//設定是否按下翻滾

void SetIsSquat(bool isSquat);//設定是否蹲下

void SetIsPortaling(bool isPortaling, int destX, int destY); //設定是否在傳送 和 傳送位置

void SetIsUnconquered(bool isUnconquered);//設定是否無敵

bool GetIsJump();//是否在跳躍

int GetPickCount(); //撿了幾次武器

bool HasWeapon();//是否有武器

void ChangeWeapon(int weaponNum);//切換武器

PlayerEquipment\* GetWeapon();

PlayerEquipment\* GetWeapon1();

PlayerEquipment\* GetWeapon2();

int CurrentWeapon();//目前用的武器是哪一把

//----------------動作相關---------------------//

void Act();//按下按鍵行動

void Fall();//下降

void Jump();//跳躍

void DownJump();//下跳

void Interact();//互動

void Move(int dx, int dy)override;//移動特定距離

void MoveTo(int x, int y)override;//移動到特定座標

void Attack()override;//攻擊

void Portaling();//傳送

//----------------動畫相關---------------------//

void ShowBitMap()override;//顯示動畫

//-------------顯示玩家資訊------------------//

void ShowInformation();//顯示玩家資訊

//-------------系統相關------------------//

template <class T> void AddEquipment(T\* equipment)//增加裝備

{

this->equipments.push\_back(equipment);

if (dynamic\_cast<PlayerWeapon\*>(equipment))//撿起的裝備是武器

{

pickCount++; //計算撿裝備的次數

if (weaponCount < MAX\_WEAPON\_COUNT)//目前武器數量小於最大武器數量

{

if (!weapon1)//還沒有武器

{

this->weapon1 = equipments[equipments.size() - 1];

this->weapon = weapon1;

}

else if (!weapon2)//還沒有第二把武器

{

this->weapon2 = equipments[equipments.size() - 1];

this->weapon = weapon2;

}

hasWeapon = true;//有武器了

weaponCount++;//持有武器數量+1

}

else

{

RemoveWeapon();//移除武器

if (!weapon1)//武器1已清除

{

this->weapon1 = equipments[equipments.size() - 1];

this->weapon = weapon1;

}

else if (!weapon2)//武器2已清除

{

this->weapon2 = equipments[equipments.size() - 1];

this->weapon = weapon2;

}

}

}

CalculateAbility(equipment);//計算能力值

}

void RemoveWeapon();//移除武器

private:

bool isMoveLeft;//是否向左移動

bool isMoveRight;//是否向右移動

bool isJumpKeyDown;//是否按下跳躍

bool isJump;//是否正在跳躍

bool isDownJump;//是否正在下跳

bool isFall;//是否正在下降

bool isAttack;//是否正在攻擊

bool isRoll;//是否正在翻滾

bool isRollKeyDown;//是否按下翻滾

bool isSquatKeyDown;//是否按下蹲下

bool isSquat;//是否蹲下

bool isUnconquered;//是否無敵

bool isGetHit;//是否被打

bool isPortaling;//是否正在傳送途中

//---------------跳躍相關---------------//

int fallDisplacement;//下降位移量(移動到沒有地板的位置會用到)

int originJumpDisplacement, jumpDisplacement;//跳躍位移量

bool isGrounded;//是否在地上

int jumpCount = 0;//跳躍段數計數

const int MAX\_JUMP\_COUNT = 3;//最多能幾段跳

//---------------蹲下相關---------------//

void ChangeHeight(int height);//讓寬高變一樣

bool CanStand();//目前空間高度，能不能站起來

//---------------下跳相關---------------//

//bool IsFloorOnGround();//地板跟最底層地板剛好貼合，代表不能下跳

bool HasSpaceToDownJump();//下方有足夠的高度可以容納玩家

void CalculateAbility(PlayerEquipment\* equipment);//計算能力值

void Dead()override;//死亡時呼叫

//---------------翻滾相關---------------//

void Roll();//翻滾

int originRollDisplacement, rollDisplacement;//翻滾位移量

int originWidth, originHeight;//翻滾時角色是倒下的，寬高會互換，用origin變數儲存原本的寬高

void FlipWidthHeight();//對調寬高

void AdjustY(int dy);//調整y座標

//---------------傳送相關---------------//

int destinationX, destinationY;

//---------------動畫相關---------------//

void ShowWeapon();//顯示武器

bool faceLR;//面向左邊或面向右邊

const bool FACE\_LEFT = true;//面向左邊

const bool FACE\_RIGHT = false;//面向右邊

void LoadAni()override;//載入動畫

int currentAni = 0;//目前動畫

enum ANI

{

ANI\_IDLE = 0,

ANI\_LEFT,

ANI\_RIGHT,

ANI\_ATTACK\_LEFT,

ANI\_ATTACK\_RIGHT,

ANI\_JUMP\_LEFT,

ANI\_JUMP\_RIGHT,

ANI\_GET\_HIT\_LEFT,

ANI\_GET\_HIT\_RIGHT,

ANI\_ROLL\_LEFT,

ANI\_ROLL\_RIGHT,

ANI\_SQUAT\_LEFT,

ANI\_SQUAT\_RIGHT,

ANI\_PORTALING

};

vector<PlayerEquipment\*> equipments;//裝備

PlayerEquipment\* weapon = NULL;//目前武器

PlayerEquipment\* weapon1 = NULL,\*weapon2 = NULL;//身上的兩把武器

const int MAX\_WEAPON\_COUNT = 2;//最大武器數量

int weaponCount = 0;//武器數量

bool hasWeapon = false;

int pickCount = 0; //算撿起的數量

};

Player.cpp

#include "StdAfx.h"

#include "Player.h"

#include "Monster.h"

#include "Map.h"

#include "Item.h"

#include "PlayerTradationalSword.h"

#include "Floor.h"

#include "Door.h"

#include "Goal.h"

#include "Portal.h"

Player::Player()

{

tag = "Player";

}

Player::~Player()

{

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

{

delete equipments[i];

}

equipments.clear();

}

Player::Player(string tag, int x, int y, int width, int height) :Character(tag, x, y, width, height)

{

tag = "Player";

maxHP = 1000;

HP = maxHP;

originWidth = width;

originHeight = height;

originMoveSpeed = 5;

moveSpeed = originMoveSpeed;

fallDisplacement = 0;

originJumpDisplacement = 15;

jumpDisplacement = originJumpDisplacement;

originRollDisplacement = 15;

rollDisplacement = originRollDisplacement;

attackRange = 10;

attackDamage = 10;

attackSpeed = 5;

defense = 0;

destinationX = 0;

destinationY = 0;

pickCount = 0;

isMoveLeft = false;

isMoveRight = false;

isJumpKeyDown = false;

isJump = false;

isFall = false;

isGrounded = false;

isRollKeyDown = false;

isRoll = false;

isSquatKeyDown = false;

isSquat = false;

isUnconquered = false;

isDownJump = false;

isPortaling = false;

isAttack = false;

isGetHit = false;

faceLR = FACE\_RIGHT;

currentAni = ANI::ANI\_IDLE;

layer = GameSystem::LAYER::LAYER\_PLAYER;

LoadAni();

}

void Player::AdjustPositionOnBegin()

{

while (Map::HasObject(x, y + this->height + 1))

{

this->y--;

}

}

void Player::DecreaseHP(int dhp)

{

//if (isUnconquered == false)

{

isGetHit = true;

int finalDHp = dhp - defense;//最終的扣血量

if (finalDHp < 0)

finalDHp = 0;

HP -= finalDHp;

}

if (HP <= 0)

{

HP = 0;

isDead = true;

Dead();

}

}

void Player::Move(int dx, int dy)

{

if ((dx > 0 && CanMoveRight(moveSpeed)) || (dx < 0 && CanMoveLeft(moveSpeed)))//x位移量往右而且右邊沒東西 或者 x位移量往左而且左邊沒東西

{

this->x += dx;//玩家x移動

}

this->y += dy;//玩家y移動

if (dx > 0 && this->x + this->width / 2 >= Map::GetSX() + SIZE\_X / 2)

Map::MoveScreenTopLeft(dx, 0);//螢幕移動

else if (dx < 0 && this->x + this->width / 2 < Map::GetSX() + SIZE\_X / 2)

Map::MoveScreenTopLeft(dx, 0);//螢幕移動

if (dy > 0 && this->y + this->height / 2 >= Map::GetSY() + SIZE\_Y \* 3 / 4)

Map::MoveScreenTopLeft(0, dy);//螢幕移動

else if (dy < 0 && this->y + this->height / 2 < Map::GetSY() + SIZE\_Y / 4)

Map::MoveScreenTopLeft(0, dy);//螢幕移動

}

void Player::MoveTo(int x, int y)

{

Character::MoveTo(x, y);

Map::SetSX(x - SIZE\_X / 2);

Map::SetSY(y - SIZE\_Y / 2);

}

void Player::SetIsMoveLeft(bool isMoveLeft)

{

this->isMoveLeft = isMoveLeft;

}

void Player::SetIsMoveRight(bool isMoveRight)

{

this->isMoveRight = isMoveRight;

}

void Player::SetIsJump(bool isJump)

{

this->isJumpKeyDown = isJump;

}

void Player::SetIsDownJump(bool isDownJump)

{

this->isDownJump = isDownJump;

}

void Player::SetIsAttack(bool isAttack)

{

this->isAttack = isAttack;

}

void Player::SetIsRoll(bool isRoll)

{

this->isRollKeyDown = isRoll;

}

void Player::SetIsSquat(bool isSquat)

{

this->isSquatKeyDown = isSquat;

}

void Player::SetIsUnconquered(bool isUnconquered)

{

this->isUnconquered = isUnconquered;

}

bool Player::GetIsJump()

{

return isJump;

}

int Player::GetPickCount()

{

return pickCount;

}

void Player::SetIsPortaling(bool isPortaling, int destX, int destY)

{

this->isPortaling = isPortaling;

this->destinationX = destX;

this->destinationY = destY;

}

bool Player::HasWeapon()

{

return hasWeapon;

}

void Player::ChangeWeapon(int weaponNum)

{

switch (weaponNum)

{

case 1:

if(weapon1 != NULL)//如果有武器1

weapon = weapon1;

break;

case 2:

if(weapon2 != NULL)//如果有武器2

weapon = weapon2;

break;

}

if(weapon != NULL)//有武器

CalculateAbility(weapon);//重新計算能力值

}

PlayerEquipment \* Player::GetWeapon()

{

return weapon;

}

PlayerEquipment \* Player::GetWeapon1()

{

return weapon1;

}

PlayerEquipment \* Player::GetWeapon2()

{

return weapon2;

}

int Player::CurrentWeapon()

{

if (weapon == weapon1)

return 1;

else if (weapon == weapon2)

return 2;

else

return 0;

}

void Player::SetIsGrounded(bool isGrounded)

{

this->isGrounded = isGrounded;

}

void Player::Act()//行動

{

if (isRoll == true || isSquat)//翻滾中或蹲下中不能攻擊

isAttack = false;

if (isAttack == false)//沒在攻擊

{

if (isRollKeyDown)//按下翻滾

{

isRollKeyDown = false;

if (isGrounded && isRoll == false)//在地上才能開始翻，而且目前還沒翻滾(翻轉寬高只需翻轉一次)

{

isUnconquered = true;//翻滾時無敵

isRoll = true;

FlipWidthHeight();//翻轉寬高

}

}

else if (isRoll == false)

{

if (this->isSquatKeyDown && isGetHit == false)//蹲下

{

//isSquatKeyDown = false;

if (isGrounded)

{

isSquat = true;//改變蹲下狀態

//if(height == originHeight)//高度跟原本一樣

ChangeHeight(originHeight / 2);//將高度變為一半

SetMoveSpeed(originMoveSpeed / 2);//速度變為一半

//else

//ChangeHeight(originHeight);//將高度還原

}

}

else

{

if (CanStand())

{

isSquat = false;

ChangeHeight(originHeight);//將高度還原

SetMoveSpeed(originMoveSpeed);//將速度還原

}

}

if (this->isMoveLeft)

{

if (this->x > 0)

{

faceLR = FACE\_LEFT;//面向左邊

Move(-moveSpeed, 0);

}

}

if (this->isMoveRight)

{

if (this->x + this->width < Map::WORLD\_SIZE\_X)

{

faceLR = FACE\_RIGHT;//面向右邊

Move(moveSpeed, 0);

}

}

}

if (this->isJumpKeyDown)//如果按下跳躍

{

isJumpKeyDown = false;//跳躍鍵不能持續按住

if (isRoll == false)//翻滾中不能跳躍

{

if (isGrounded)//如果在地上

{

if (isSquat)//蹲下且跳躍，進行下跳

{

SetIsDownJump(true);//下跳

isSquat = false;//下跳時不再蹲下

ChangeHeight(originHeight);//將高度還原

SetMoveSpeed(originMoveSpeed);//將速度還原

}

else

{

isJump = true;//正在跳躍

}

isGrounded = false;//沒在地上

}

if (!isDownJump)//沒有下跳才能多段跳

{

if (jumpCount < MAX\_JUMP\_COUNT)//小於最大跳躍段數

{

jumpCount++;//計數目前是幾段跳

jumpDisplacement = originJumpDisplacement;//重置跳躍位移量，呈現二段跳的效果

//以下是:若在下降中按下二段跳

isFall = false;//如果原本在下降，則不再下降

fallDisplacement = 0;//下降位移量重置

isJump = true;//重新往上跳跳躍

}

}

}

}

}

if (isRoll)

Roll();

if (isPortaling)

Portaling();

else if (isDownJump)

{

DownJump();

}

else

{

if (isJump)

Jump();//跳躍

else

Fall();//下降

}

}

void Player::Fall()

{

if (CanMoveDown(fallDisplacement))//如果腳下沒東西

{

isFall = true;//正在下降

isGrounded = false;//不在地上

fallDisplacement++;

Move(0, fallDisplacement);

}

else

{

while (CanMoveDown(1))//再繼續用下降位移量下降，將會卡進地板，所以一次向下位移1進行微調

Move(0, 1);

fallDisplacement = 0;

isGrounded = true;//在地上

isFall = false;//沒在下降

jumpCount = 0;//跳躍段數重置

if (currentAni == ANI::ANI\_JUMP\_LEFT)//將跳躍動畫還原

currentAni = ANI::ANI\_LEFT;

else if (currentAni == ANI::ANI\_JUMP\_RIGHT)

currentAni = ANI::ANI\_RIGHT;

}

}

void Player::Jump()

{

if (isGrounded == false && isFall == false)//如果沒在地上且沒在下降

{

jumpDisplacement--;//位移量隨著時間改變

if (jumpDisplacement >= 0)//往上升

{

if (CanMoveUp(jumpDisplacement))//可向上移動

Move(0, -jumpDisplacement);

else

jumpDisplacement = 0;//開始往下掉

}

else//往下降

{

isJump = false;

isFall = true;

jumpDisplacement = originJumpDisplacement;//跳躍位移量還原

}

}

}

void Player::DownJump()

{

Move(0, fallDisplacement);//向下移動

fallDisplacement++;

if (IsInFloor() == false//不在地板中

//|| y + height + 1 >= GameSystem::GetGameObjectWithTag<Floor>("Ground")->GetY()//已在最底層地板

//|| IsFloorOnGround()//地板跟最底層地板剛好貼合，代表不能下跳

|| !HasSpaceToDownJump())//下方沒有足夠的空間可以下跳

{

isDownJump = false;//沒有下跳

}

}

bool Player::HasSpaceToDownJump()

{

int returnValue = true;//回傳值

for (int i = x; i < x + width; i++)

{

int cy = y + height + 1;//玩家腳底的位置

while (Map::HasObject(i, cy))//此處有地板

{

cy++;//cy往下移動直到沒有地板

if (cy >= Map::WORLD\_SIZE\_Y)//cy已超出地圖範圍

{

return false;//沒有空間可以下跳

}

}

int h = 0;//確認高度是否足夠讓玩家進入

while (!Map::HasObject(i, cy))//此處沒有地板

{

h++;//高度增加

cy++;//cy往下移動

}

if (h < height)//沒有足夠的高度

{

return false;//不能下跳

}

}

return true;

}

void Player::ChangeHeight(int height)

{

AdjustY(this->height - height);

this->height = height;

}

bool Player::CanStand()

{

for (int i = x; i < x + width; i++)

{

int h = 0;//計算高度

int cy = y + h;//從玩家腳底開始

while (!Map::HasObject(i, cy) && h < height)//往上計算高度有多少

{

cy--;

h++;

}

if (h < height)//高度不夠

return false;

}

return true;//每個x座標上方的高度都夠

}

void Player::Interact()

{

//若有多個能夠互動的物件重疊，則先對道具互動，再對門互動

vector<Item\*> itemList;//道具

vector<Goal\*> goalList;//終點的門

vector<Portal\*> portalList;//傳送門

for (auto&i : GameSystem::GetAllGameObject())//將對應的物件加入對應的列表中

{

if (dynamic\_cast<Item\*>(i) != NULL)//道具

itemList.push\_back(dynamic\_cast<Item\*>(i));

else if (dynamic\_cast<Goal\*>(i) != NULL)//終點

goalList.push\_back(dynamic\_cast<Goal\*>(i));

else if (dynamic\_cast<Portal\*>(i) != NULL)//傳送門

portalList.push\_back(dynamic\_cast<Portal\*>(i));

}

for (auto& i : itemList)//對道具互動

{

if (i->GetX() + i->GetWidth() > this->x && i->GetX() < this->x + this->width

&& i->GetY() + i->GetHeight() > this->y && i->GetY() < this->y + this->height) {

static\_cast<Item\*>(i)->Picked();

return;

}

}

for (auto& i : goalList)//對終點互動

{

if (i->GetX() + i->GetWidth() > this->x && i->GetX() < this->x + this->width

&& i->GetY() + i->GetHeight() > this->y && i->GetY() < this->y + this->height) {

static\_cast<Goal\*>(i)->Picked();

return;

}

}

for (auto& i : portalList)//對傳送門互動

{

if (i->GetX() + i->GetWidth() > this->x && i->GetX() < this->x + this->width

&& i->GetY() + i->GetHeight() > this->y && i->GetY() < this->y + this->height) {

static\_cast<Portal\*>(i)->Used();

return;

}

}

}

void Player::Attack()

{

vector<Monster\*>monsters = GameSystem::GetGameObjectsWithType<Monster>();

//vector<Monster\*> monsters = GameSystem::GetGameObjectsWithTag<Monster>("Monster");

for (auto& i : monsters)//對怪物攻擊

{

if (i->GetX() + i->GetWidth() > this->x - attackRange && i->GetX() < this->x + this->width + attackRange

&& i->GetY() + i->GetHeight() > this->y && i->GetY() < this->y + this->height)//怪物在攻擊範圍內

{

i->PlayerAttack(attackDamage);

GameSystem::ShowText(to\_string(GetAttackDamage()), i->GetX() - Map::GetSX() + 10, i->GetY() - Map::GetSY() - 30, i->GetX() + i->GetWidth() - Map::GetSX(), i->GetY() + i->GetHeight() - Map::GetSY(), GameSystem::ALIGN\_CENTER, GameSystem::ALIGN\_TOP, 20, RGB(255, 0, 0));

}

}

isAttack = false;//攻擊結束

}

void Player::Portaling() //傳送

{

int dX = (destinationX - x) / 4;//四張動畫 每張動畫得位移量

int dY = (destinationY - y) / 4;

if (x != destinationX && dX > 0)

{

x += dX;

Map::MoveScreenTopLeft(dX, 0);

}

else if (x != destinationX && dX < 0)

{

x += dX;

Map::MoveScreenTopLeft(dX, 0);

}

if (y != destinationY && dY < 0)

{

y += dY;

Map::MoveScreenTopLeft(0, dY);

}

else if (y != destinationY && dY > 0)

{

y += dY;

Map::MoveScreenTopLeft(0, dY);

}

if (abs(x - destinationX) <= 30 && abs(y - destinationY) <= 30)

isPortaling = false;

}

void Player::ShowBitMap()

{

if (isGetHit)//被攻擊

{

if (faceLR == FACE\_LEFT)

{

currentAni = ANI\_GET\_HIT\_LEFT;

}

else

{

currentAni = ANI\_GET\_HIT\_RIGHT;

}

ani[currentAni]->OnMove();

if (ani[currentAni]->IsEnd())

isGetHit = false;

}

else if (isAttack)

{

if (faceLR == FACE\_LEFT)

{

currentAni = ANI::ANI\_ATTACK\_LEFT;

ani[ANI::ANI\_ATTACK\_LEFT]->OnMove();

}

else

{

currentAni = ANI::ANI\_ATTACK\_RIGHT;

ani[ANI::ANI\_ATTACK\_RIGHT]->OnMove();

}

if (ani[currentAni]->IsEnd())

{

Attack();

}

}

else if (isRoll)//翻滾動畫

{

if (faceLR == FACE\_LEFT)

{

currentAni = ANI::ANI\_ROLL\_LEFT;

}

else

{

currentAni = ANI::ANI\_ROLL\_RIGHT;

}

}

else if (isSquat)

{

if (faceLR == FACE\_LEFT)

{

currentAni = ANI::ANI\_SQUAT\_LEFT;

}

else

{

currentAni = ANI::ANI\_SQUAT\_RIGHT;

}

}

else if (isJump || isFall || isDownJump)//跳躍動畫

{

if (faceLR == FACE\_LEFT)

{

currentAni = ANI::ANI\_JUMP\_LEFT;

}

else

{

currentAni = ANI::ANI\_JUMP\_RIGHT;

}

}

else if (isPortaling)

{

currentAni = ANI::ANI\_PORTALING;

ani[ANI::ANI\_PORTALING]->OnMove();

}

else if (isMoveLeft)//左移動畫

{

currentAni = ANI::ANI\_LEFT;

ani[ANI::ANI\_LEFT]->OnMove();

}

else if (isMoveRight)//右移動畫

{

currentAni = ANI::ANI\_RIGHT;

ani[ANI::ANI\_RIGHT]->OnMove();

}

else

{

for (auto& i : ani)

{

i->Reset();//重置所有動畫

}

if (faceLR == FACE\_LEFT)

currentAni = ANI::ANI\_LEFT;

else

currentAni = ANI::ANI\_RIGHT;

}

ani[currentAni]->OnShow();

if (hasWeapon)//有武器就顯示武器

{

ShowWeapon();

}

}

void Player::RemoveWeapon()

{

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

{

if (equipments[i] == weapon)

{

if (weapon == weapon1)

{

weapon1 = NULL;//清除武器

}

else if (weapon == weapon2)

{

weapon2 = NULL;//清除武器

}

delete equipments[i];

equipments.erase(equipments.begin() + i);

break;

}

}

}

void Player::CalculateAbility(PlayerEquipment\* equipment)

{

//攻擊力與防禦累加上去

this->attackDamage = equipment->GetAttackDamage();

this->defense = equipment->GetDefense();

//攻擊距離使用武器的攻擊距離

this->attackRange = equipment->GetAttackRange();

}

void Player::Dead()

{

GameSystem::SetGameOver(GameSystem::OVER\_LOSE);//遊戲結束

//GameSystem::DeleteGameObject(this);

}

void Player::Roll()

{

if (faceLR == FACE\_LEFT)

{

Move(-rollDisplacement, 0);

}

else

{

Move(rollDisplacement, 0);

}

for (auto& i : GameSystem::GetGameObjectsWithTag<Door>("Door"))//翻滾時踢門

{

if (i->GetX() < this->x + this->width && i->GetX() + i->GetWidth() > this->x

&& i->GetY() < this->y + this->height && i->GetY() + i->GetHeight() > this->y && !i->GetRuin())//門在玩家範圍內

{

i->Kicked();//踢門

break;

}

}

if (rollDisplacement-- <= 0)

{

isUnconquered = false;//無敵結束

isRoll = false;//翻滾結束

rollDisplacement = originRollDisplacement;

FlipWidthHeight();//還原寬高

if (CanStand() == false)//翻滾完之後的位置無法站起

{

isSquat = true;//變為蹲下狀態

ChangeHeight(originHeight / 2);//將高度變為一半

SetMoveSpeed(originMoveSpeed / 2);//速度變為一半

}

}

}

void Player::FlipWidthHeight()

{

int temp = width;

width = height;

height = temp;

AdjustY(width - height);

}

void Player::AdjustY(int dy)

{

y += dy;

}

void Player::ShowWeapon()

{

if (faceLR == FACE\_LEFT)

{

weapon->SetXY(x, y, ANI\_ATTACK\_LEFT, ani[ANI::ANI\_ATTACK\_LEFT]->GetCurrentBitmapNumber());

weapon->ShowBitMap();

}

else

{

weapon->SetXY(x, y, ANI\_ATTACK\_RIGHT, ani[ANI::ANI\_ATTACK\_RIGHT]->GetCurrentBitmapNumber());

weapon->ShowBitMap();

}

}

void Player::LoadAni()

{

objectPic.LoadBitmap(".\\res\\player.bmp");

//---------------靜止

char\* aniIdle[1] = { ".\\res\\player\_idle.bmp" };

AddAniBitMaps(aniIdle, ANI::ANI\_IDLE, 1);

//---------------左走

char\* aniLeft[4] = { ".\\res\\player\_left\_0.bmp", ".\\res\\player\_left\_1.bmp", ".\\res\\player\_left\_2.bmp", ".\\res\\player\_left\_3.bmp" };

AddAniBitMaps(aniLeft, ANI::ANI\_LEFT, 4);

//---------------右走

char\* aniRight[4] = { ".\\res\\player\_right\_0.bmp", ".\\res\\player\_right\_1.bmp", ".\\res\\player\_right\_2.bmp", ".\\res\\player\_right\_3.bmp" };

AddAniBitMaps(aniRight, ANI::ANI\_RIGHT, 4);

//---------------左跳

char\* aniJumpLeft = ".\\res\\player\_jump\_left.bmp";

AddAniBitMap(aniJumpLeft, ANI::ANI\_JUMP\_LEFT);

//---------------右跳

char\* aniJumpRight = ".\\res\\player\_jump\_right.bmp";

AddAniBitMap(aniJumpRight, ANI::ANI\_JUMP\_RIGHT);

//---------------左攻

char\* aniAttackLeft[3] = { ".\\res\\player\_attack\_left\_0.bmp", ".\\res\\player\_attack\_left\_1.bmp", ".\\res\\player\_attack\_left\_2.bmp" };

AddAniBitMaps(aniAttackLeft, ANI::ANI\_ATTACK\_LEFT, 3, 5);

//---------------右攻

char\* aniAttackRight[3] = { ".\\res\\player\_attack\_right\_0.bmp", ".\\res\\player\_attack\_right\_1.bmp", ".\\res\\player\_attack\_right\_2.bmp" };

AddAniBitMaps(aniAttackRight, ANI::ANI\_ATTACK\_RIGHT, 3, 5);

//---------------左被擊

char\* aniGetHitLeft = ".\\res\\player\_get\_hit\_left.bmp";

AddAniBitMap(aniGetHitLeft, ANI::ANI\_GET\_HIT\_LEFT);

//---------------右被擊

char\* aniGetHitRight = ".\\res\\player\_get\_hit\_right.bmp";

AddAniBitMap(aniGetHitRight, ANI::ANI\_GET\_HIT\_RIGHT);

//---------------左翻滾

char\* aniRollLeft = ".\\res\\player\_roll\_left.bmp";

AddAniBitMap(aniRollLeft, ANI::ANI\_ROLL\_LEFT);

//---------------右翻滾

char\* aniRollRight = ".\\res\\player\_roll\_right.bmp";

AddAniBitMap(aniRollRight, ANI::ANI\_ROLL\_RIGHT);

//左蹲下

char\* aniSquatLeft = ".\\res\\player\_squat\_left.bmp";

AddAniBitMap(aniSquatLeft, ANI::ANI\_SQUAT\_LEFT);

//右蹲下

char\* aniSquatRight = ".\\res\\player\_squat\_right.bmp";

AddAniBitMap(aniSquatRight, ANI::ANI\_SQUAT\_RIGHT);

//傳送

char\* aniPortaling[4] = { ".\\res\\portaling0.bmp", ".\\res\\portaling1.bmp" , ".\\res\\portaling2.bmp" ,".\\res\\portaling0.bmp" };

AddAniBitMaps(aniPortaling, ANI::ANI\_PORTALING, 4, 2);

}

PlayerBow.h

#pragma once

#include "PlayerWeapon.h"

#include "Arrow.h"

#include "Player.h"

class PlayerBow :public PlayerWeapon

{

public:

~PlayerBow();

PlayerBow(ItemWeapon\* rhs);

void ShowBitMap()override;

void SetXY(int hostX, int hostY, int playerCurrentAni, int playerAniNumber)override;

private:

void LoadAni()override;

bool isShoot; //每次攻擊射一發

enum ANI {

ANI\_IDLE = 0,

ANI\_LEFT,

ANI\_RIGHT

};

};

PlayerBow.cpp

#include "StdAfx.h"

#include"PlayerBow.h"

PlayerBow::~PlayerBow()

{

}

PlayerBow::PlayerBow(ItemWeapon\* rhs) :PlayerWeapon(rhs)

{

SetX(player->GetX());

SetY(player->GetY());

LoadAni();

haveEffect = true;

isShoot = false;

}

void PlayerBow::SetXY(int hostX, int hostY, int playerCurrentAni, int playerAniNumber)

{

SetX(hostX);

SetY(hostY);

this->playerCurrentAni = playerCurrentAni;

this->playerAniNumber = playerAniNumber;

}

void PlayerBow::ShowBitMap()

{

if((playerCurrentAni == ANI\_ATTACK\_LEFT || playerCurrentAni == ANI\_ATTACK\_RIGHT) && playerAniNumber==2 && isShoot == false){ //增加arrow物件

isShoot = true;

GameSystem::AddGameObject(new Arrow("Arrow", x, y, 48, 48, 300, playerCurrentAni));

}

else if (playerAniNumber == 0 && isShoot == true)

{

isShoot = false;

}

SetXY(player->GetX(), player->GetY(), playerCurrentAni, playerAniNumber);

SetBitMapPosition();

ani[currentAni]->OnShow();

}

void PlayerBow::LoadAni()

{

char\* aniIdle[1] = { ".\\res\\bow.bmp" };

AddAniBitMaps(aniIdle, ANI\_IDLE, 1);

}

PlayerEquipment.h

#pragma once

//PlayerEquipment

//主角穿著的裝備#include "GameSystem.h"

#include "ItemWeapon.h"

class Player;

class PlayerEquipment

{

public:

PlayerEquipment();

virtual ~PlayerEquipment();

PlayerEquipment(ItemWeapon\* rhs);

PlayerEquipment\* operator=(ItemWeapon\* rhs);//增加裝備時可直接轉換

void SetX(int x);

void SetY(int y);

string GetTag();

int GetAttackDamage();

int GetAttackRange();

int GetDefense();

virtual void SetXY(int hostX, int hostY, int playerCurrentAni, int palyerAniNumber); //

virtual void ShowBitMap();//顯示圖片

protected:

bool IsObjectInRange(GameObject\* obj, int right\_fix, int left\_fix, int up\_fix, int down\_fix);//判斷動畫是否即擊中

void AddAniBitMaps(char\* pic[], int aniType, int picCount); //增加多張動畫圖片

void AddAniBitMap(char\* pic, int aniType); //增加動畫圖片

void LoadBitMap(char\* picAddress); //載入圖片(CMovingBitamp pic)

void SetBitMapPosition(); //設定動畫位置

virtual void LoadAni(); //載入動畫(CAnimation ani)

int currentAni = ANI\_IDLE; //IDLE 動畫

vector<CAnimation\*> ani;//動畫

enum ANI {

ANI\_IDLE = 0,

};

string tag; //

int x, y;

int attackDamage; //攻擊力

int attackRange; //攻擊範圍

int defense; //防禦力

CMovingBitmap pic;

Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player");

};

PlayerEquipment.cpp

#include "StdAfx.h"

#include "PlayerEquipment.h"

#include "Player.h"

PlayerEquipment::PlayerEquipment()

{

tag = "PlayerEquipment";

}

PlayerEquipment::~PlayerEquipment()

{

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

{

delete ani[i];

}

ani.clear();

}

PlayerEquipment::PlayerEquipment(ItemWeapon\* rhs):PlayerEquipment()

{

player = GameSystem::GetGameObjectWithTag<Player>("Player"); //常用到就先記錄下來

tag = "PlayerEquipment";

operator=(rhs);

}

PlayerEquipment\* PlayerEquipment::operator=(ItemWeapon\* rhs)

{

attackDamage = rhs->GetAttackDamage();

attackRange = rhs->GetAttackRange();

defense = rhs->GetDefense();

pic = rhs->objectPic;

return this;

}

void PlayerEquipment::SetX(int x)

{

this->x = x;

}

void PlayerEquipment::SetY(int y)

{

this->y = y;

}

string PlayerEquipment::GetTag()

{

return tag;

}

int PlayerEquipment::GetAttackDamage()

{

return attackDamage;

}

int PlayerEquipment::GetAttackRange()

{

return attackRange;

}

int PlayerEquipment::GetDefense()

{

return defense;

}

void PlayerEquipment::SetXY(int hostX, int hostY, int playerCurrentAni, int playerAniNumber) {

this->x = hostX;

this->y = hostY;

}

void PlayerEquipment::ShowBitMap()

{

pic.SetTopLeft(x - Map::GetSX(), y - Map::GetSY());

pic.ShowBitmap();

}

//-----------------------------PROTECTED-----------------------------------//

bool PlayerEquipment::IsObjectInRange(GameObject\* obj, int right\_fix, int left\_fix, int up\_fix, int down\_fix) {

int width = player->GetWidth(), height = player->GetHeight();

int RIGHT\_EDGE = x + player->GetWidth() + right\_fix, LEFT\_EDGE = x - left\_fix,

UP\_EDGE = y - up\_fix, DOWN\_EDGE = y + height + down\_fix;

int OB\_X = obj->GetX(), OB\_Y = obj->GetY(), OB\_WIDTH = obj->GetWidth(), OB\_HEIGHT = obj->GetHeight();

int OB\_RIGHT\_EDGE = OB\_X + OB\_WIDTH, OB\_LEFT\_EDGE = OB\_X,

OB\_UP\_EDGE = OB\_Y, OB\_DOWN\_EDGE = OB\_Y + OB\_HEIGHT;

if (OB\_RIGHT\_EDGE >= LEFT\_EDGE && OB\_RIGHT\_EDGE <= RIGHT\_EDGE) //人在左, 火焰在右

return true;

else if (OB\_LEFT\_EDGE <= RIGHT\_EDGE && OB\_LEFT\_EDGE >= LEFT\_EDGE) //人在右, 火焰在左

return true;

return false;

}

void PlayerEquipment::AddAniBitMaps(char \* pic[], int aniType, int picCount)

{

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

{

AddAniBitMap(pic[i], aniType);

}

}

void PlayerEquipment::AddAniBitMap(char\* pic, int aniType)

{

while ((int)ani.size() <= aniType)//vector的大小不夠大

ani.push\_back(new CAnimation);//增加大小

ani[aniType]->AddBitmap(pic, RGB(255, 255, 255));

}

void PlayerEquipment::LoadBitMap(char\* picAddress)

{

pic.LoadBitmap(picAddress);

}

void PlayerEquipment::SetBitMapPosition()

{

for (auto& i : ani)

{

i->SetTopLeft(this->x - Map::GetSX(), this->y - Map::GetSY());

}

}

void PlayerEquipment::LoadAni() {}

PlayerTraditionalSword.h

#pragma once

#include "PlayerWeapon.h"

#include "SwordWave.h"

class PlayerTraditionalSword:public PlayerWeapon

{

public:

PlayerTraditionalSword();

~PlayerTraditionalSword();

PlayerTraditionalSword(ItemWeapon\* rhs);

void SetXY(int hostX, int hostY, int playerCurrentAni, int playerAniNumber)override;

void ShowBitMap()override;

private:

void LoadAni()override;

SwordWave\* wave;//劍波

enum ANI {

ANI\_IDLE = 0,

ANI\_LEFT,

ANI\_RIGHT

};

};

PlayerTraditionalSword.cpp

#include "StdAfx.h"

#include "PlayerTradationalSword.h"

PlayerTraditionalSword::PlayerTraditionalSword()

{

}

PlayerTraditionalSword::~PlayerTraditionalSword()

{

delete wave;

}

PlayerTraditionalSword::PlayerTraditionalSword(ItemWeapon\* rhs):PlayerWeapon(rhs)

{

SetX(player->GetX());

SetY(player->GetY());

LoadAni();

wave = new SwordWave("SwordWave", x, y, 77, 70);

haveEffect = true;

}

void PlayerTraditionalSword::SetXY(int hostX, int hostY, int playerCurrentAni, int playerAniNumber)

{

SetX(hostX);

SetY(hostY);

this->playerCurrentAni = playerCurrentAni;

this->playerAniNumber = playerAniNumber;

}

void PlayerTraditionalSword::ShowBitMap()

{

wave->ShowBitMap(x, y, playerCurrentAni, playerAniNumber); //int hostX, int hostY, int hostCurrentAni, int attackAniNumber

//SetXY(player->GetX(), player->GetY(), playerCurrentAni, playerAniNumber);

SetBitMapPosition();

ani[currentAni]->OnShow();

}

void PlayerTraditionalSword::LoadAni()

{

char\* aniIdle[1] = { ".\\res\\sword\_01d.bmp" };

AddAniBitMaps(aniIdle, ANI\_IDLE, 1);

}

PlayerWeapon.h

#pragma once

#include "PlayerEquipment.h"

#include "Player.h"

class PlayerWeapon :public PlayerEquipment

{

public:

PlayerWeapon();

virtual ~PlayerWeapon();

PlayerWeapon(ItemWeapon\* rhs);

protected:

bool haveEffect = false; //是否 有特效;

int playerCurrentAni = 0, playerAniNumber = 0; //用來判斷玩家攻擊動畫進行到哪個階段，進而顯示Effect

const int ANI\_ATTACK\_LEFT = 3, ANI\_ATTACK\_RIGHT = 4; //攻擊動編碼

};

PlayerWeapon.cpp

#include "StdAfx.h"

#include "PlayerWeapon.h"

PlayerWeapon::PlayerWeapon() {

tag = "PlayerWeapon";

}

PlayerWeapon::~PlayerWeapon()

{

}

PlayerWeapon::PlayerWeapon(ItemWeapon\* rhs) :PlayerEquipment(rhs) {

tag = "PlayerWeapon";

playerCurrentAni = 0;

playerAniNumber = 0;

}

Portal.h

#pragma once

#include "GameObject.h"

#include "GameSystem.h"

#include "Monster.h"

#include "Player.h"

class Portal :public GameObject

{

public:

Portal();

~Portal();

Portal(string tag, int x, int y, int width, int height, int number);

int GetConnectedNumber();

void Used(); //玩家使用傳送門

void SetBitMapPosition()override;

void ShowBitMap()override;

void LoadAni();

private:

CAnimation\* ani;

Player\* player;

int connectedNumber; //傳送門編號

Portal\* Connected(); //回傳另一個傳送點(如果找不到 就會回傳自己)

};

Portal.cpp

#include "StdAfx.h"

#include "Portal.h"

Portal::Portal() {};

Portal::Portal(string tag, int x, int y, int width, int height, int number) :GameObject(tag, x, y, width, height)

{

ani = new CAnimation();

LoadAni();

player = GameSystem::GetGameObjectWithTag<Player>("Player");

layer = GameSystem::LAYER::LAYER\_FLOOR;

connectedNumber = number;

}

Portal::~Portal()

{

delete ani;

}

int Portal::GetConnectedNumber()

{

return this->connectedNumber;

}

void Portal::Used()

{

Portal\* anotherPortal = Connected();

player->SetIsPortaling(true, Connected()->GetX(), Connected()->GetY()); //讓玩家進入傳送模式

}

void Portal::SetBitMapPosition()

{

ani->SetTopLeft(x - Map::GetSX(), y - Map::GetSY());

}

void Portal::ShowBitMap()

{

ani->OnShow();

}

void Portal::LoadAni()

{

char\* picture0 = ".\\res\\portal.bmp";

ani->AddBitmap(picture0,RGB(255,255,255));

};

Portal\* Portal::Connected()

{

vector<Portal\*> portals = GameSystem::GetGameObjectsWithTag<Portal>("Portal");

for (auto& i : portals)

{

if (i->GetConnectedNumber() + this->connectedNumber == 100 && i)

{

return i;

}

}

return this;

}

Potion.h

#pragma once

//Potion

//場上的補品道具

//繼承Item

#include "GameSystem.h"

#include "Item.h"

#include "Player.h"

class Potion :public Item

{

public:

Potion();

~Potion();

Potion(string tag, int x, int y, int width, int height);

void Picked()override;//被撿取

private:

virtual void Dead()override;

int recoverHP;//HP回復量

};

Potion.cpp

#include "StdAfx.h"

#include "Potion.h"

Potion::Potion()

{

tag = "Potion";

}

Potion::~Potion()

{

}

Potion::Potion(string tag, int x, int y, int width, int height) :Item(tag, x, y, width, height)

{

recoverHP = 100;

ani[0]->AddBitmap(".\\res\\potion.bmp",RGB(255,255,255));

//LoadBitMap(".\\res\\cell\_green.bmp");

}

void Potion::Picked()

{

Player& player = \*(GameSystem::GetGameObjectWithTag<Player>("Player"));

player.IncreaseHP(recoverHP);//玩家回血

Dead();

}

void Potion::Dead()

{

GameSystem::DeleteGameObject(this);//刪除此物件

}

SwordWave.h

#pragma once

#include"StdAfx.h"

#include"Effect.h"

class SwordWave :public Effect {

public:

SwordWave(string tag, int x, int y, int width, int height);

~SwordWave();

void ShowBitMap(int hostX, int hostY, int hostCurrentAni, int attackAniNumber)override; //顯示動畫

private:

void LoadAni()override;

void SetXY(int hostX, int hostY, int hostCurrentAni)override; //跟著擁有者移動

enum ANI{

ANI\_SWORDWAVE\_IDLE =0,

ANI\_SWORDWAVE\_LEFT,

ANI\_SWORDWAVE\_RIGHT

};

};

SwordWave.cpp

#include"StdAfx.h"

#include"SwordWave.h"

SwordWave::SwordWave(string tag, int x, int y, int width, int height) :Effect(tag, x, y, width, height) {

tag = "SwordWave";

LoadAni();

SetBitMapPosition();

currentAni = ANI\_SWORDWAVE\_IDLE;

LoadBitMap(".\\res\\swordwave\_left\_0.bmp");

}

SwordWave::~SwordWave()

{

}

void SwordWave::ShowBitMap(int hostX, int hostY, int hostCurrentAni, int attackAniNumber) {

if(attackAniNumber == 0&&(hostCurrentAni== ANI\_ATTACK\_LEFT || hostCurrentAni == ANI\_ATTACK\_RIGHT)){

SetXY(hostX, hostY, hostCurrentAni);

}

else if (attackAniNumber >= 1 && currentAni == ANI\_SWORDWAVE\_LEFT && CanMoveLeft(5) && isHit == 0) { //3是左攻擊狀態

x -= 25; //讓Wave往左飛

SetBitMapPosition();

ani[currentAni]->OnMove();

ani[currentAni]->OnShow();

EffectAttackMonster(player->GetAttackDamage());

}

else if (attackAniNumber >= 1 && currentAni == ANI\_SWORDWAVE\_RIGHT && CanMoveRight(5) && isHit == 0) { //4是右攻擊狀態

x += 25; //讓Wave往右飛

SetBitMapPosition();

ani[currentAni]->OnMove();

ani[currentAni]->OnShow();

EffectAttackMonster(player->GetAttackDamage());

}

else {

isHit = 0;

currentAni = ANI\_SWORDWAVE\_IDLE;

}

}

void SwordWave::SetXY(int hostX, int hostY, int hostCurrentAni) {

if (hostCurrentAni == ANI\_ATTACK\_LEFT) {//為了確保腳色攻擊變換方向時 特效不會換方向

currentAni = ANI\_SWORDWAVE\_LEFT;

this->x = hostX - 45;

this->y = hostY;

}

else if (hostCurrentAni == ANI\_ATTACK\_RIGHT) { //為了確保腳色攻擊變換方向時 特效不會換方向

currentAni = ANI\_SWORDWAVE\_RIGHT;

this->x = hostX + 45;

this->y = hostY ;

}

}

void SwordWave::LoadAni() {

char\* aniSwordWave\_idle[1] = { ".\\res\\swordwave\_left\_0.bmp" };

AddAniBitMaps(aniSwordWave\_idle, ANI\_SWORDWAVE\_IDLE, 1);

char\* aniSwordWave\_left[3] = { ".\\res\\swordwave\_left\_0.bmp", ".\\res\\swordwave\_left\_1.bmp", ".\\res\\swordwave\_left\_2.bmp" };

AddAniBitMaps(aniSwordWave\_left, ANI\_SWORDWAVE\_LEFT, 3);

char\* aniSwordWave\_right[3] = { ".\\res\\swordwave\_right\_0.bmp", ".\\res\\swordwave\_right\_1.bmp",".\\res\\swordwave\_right\_2.bmp" };

AddAniBitMaps(aniSwordWave\_right, ANI\_SWORDWAVE\_RIGHT, 3);

}

Treasure.h

#pragma once

#include "Item.h"

#include "GameSystem.h"

#include "Potion.h"

#include "ItemTraditionalSword.h"

#include "ItemWeaponBow.h"

class Treasure :public Item

{

public:

Treasure(string tag, int x, int y, int width, int height, int damageMin, int damageMax, int defenseMin, int defenseMax);

void Picked()override;

private:

void LoadAni()override;

void GenWeapon(int damageMin, int damageMax, int defenseMin, int denfenseMax); //產生裝備

int damageMin, damageMax, defenseMin, defenseMax;//用來設定這箱寶箱裝備的品質

};

Treasure.cpp

#include"StdAfx.h"

#include "Treasure.h"

Treasure::Treasure(string tag, int x, int y, int width, int height, int damageMin, int damageMax, int defenseMin, int defenseMax) :Item(tag, x, y, width, height)

{

tag = "Treasure";

//SetY(y); //不加這行Y軸會在某個地方被改到

LoadAni();

this->damageMin = damageMin;

this->damageMax = damageMax;

this->defenseMin = defenseMin;

this->defenseMax = defenseMax;

//LoadBitMap(".\\res\\Treasure\_Idle.bmp");

}

void Treasure::Picked()

{

GenWeapon(damageMin, damageMax, defenseMin, defenseMax);

//GameSystem::AddGameObject(new ItemTraditionalSword("ItemWeapon", x, y, 32, 32, 0, 0));

Dead();

}

void Treasure::LoadAni()

{

char\* aniIdle[1] = { ".\\res\\Treasure\_Idle.bmp" };

AddAniBitMaps(aniIdle, ANI\_IDLE, 1);

};

void Treasure::GenWeapon(int damageMin, int damageMax, int defenseMin, int defenseMax)

{

int amount = 0;

amount = static\_cast<int>( GameSystem::Rand(4));

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

{

int type = static\_cast<int>(GameSystem::Rand(10));

if (type <= 2)

{

GameSystem::AddGameObject(new Potion("Potion", x + 10 \* i, y, 48, 48));

}

else if (type <= 5)

{

int damage = static\_cast<int>(GameSystem::Rand(damageMin, damageMax));

int defense = static\_cast<int>(GameSystem::Rand(defenseMin, defenseMax));

GameSystem::AddGameObject(new ItemTraditionalSword("ItemWeapon", x + 16\* i, y, 32, 32, damage, defense));

}

else if (type <= 10)

{

int damage = static\_cast<int>(GameSystem::Rand(damageMin, damageMax));

int defense = static\_cast<int>(GameSystem::Rand(defenseMin, defenseMax));

GameSystem::AddGameObject(new ItemWeaponBow("ItemWeapon", x + 10 \* i, y, 48, 48, damage, defense));

}

}

}

UIBlood.h

#pragma once

#include "UInterface.h"

#include "Player.h"

#include "GameSystem.h"

class UIBlood : public UInterface

{

public:

UIBlood(string tag, int x, int y, int width, int height);

~UIBlood();

void ShowBitMap();

private:

void RefreshTargetWidth();//更新目標寬度

void OnShow();

Player\* player = GameSystem::GetGameObjectWithTag<Player>("Player");

int targetWidth;//目標寬度

int maxWidth;//最大寬度

};

UIBlood.cpp

#include"StdAfx.h"

#include"UIBlood.h"

UIBlood::UIBlood(string tag, int x, int y, int width, int height) :UInterface(tag, x, y, width, height)

{

this->x = 110;

this->y = 10;

this->width = maxWidth \* player->GetHP() / player->GetMaxHP();

this->maxWidth = 100;//最大寬度

this->width = maxWidth;//最開始的寬度設為最大寬度

targetWidth = this->width;//最開始的寬度設為最大寬度

}

UIBlood::~UIBlood()

{

}

void UIBlood::ShowBitMap()

{

RefreshTargetWidth();

OnShow();

}

void UIBlood::RefreshTargetWidth()

{

this->targetWidth = maxWidth \* player->GetHP() / player->GetMaxHP();

//this->targetWidth = player->GetHP() \* 2;

}

void UIBlood::OnShow() {

GameSystem::DrawRectangle(0, 0, 300, 100, RGB(210, 210, 210));

if (targetWidth < width && width <= maxWidth)

{

width -= 1;

}

else if (targetWidth > width && width < maxWidth)

{

width += 1;

}

GameSystem::DrawRectangle(x, y, maxWidth, height, RGB(0, 0, 0));//畫血條底部

GameSystem::DrawRectangle(x, y, width, height, RGB(255, 0, 0));//畫血條長度

//顯示UI文字

string text = "HP:" + to\_string(player->GetHP()) + "\n攻擊力:" + to\_string(player->GetAttackDamage())

+ "\n防禦力:" + to\_string(player->GetDefense());

GameSystem::ShowText(text, x + maxWidth + 10, 10, SIZE\_X, SIZE\_Y, GameSystem::ALIGN\_LEFT, GameSystem::ALIGN\_TOP, 8);

//顯示UI圖片

CMovingBitmap uiPlayer;

uiPlayer.LoadBitmap(".\\res\\UIPlayer.bmp", RGB(255, 255, 255));

uiPlayer.SetTopLeft(0, 0);

uiPlayer.ShowBitmap();

}

UIMonsterBlood.h

#pragma once

#include "UInterface.h"

#include "Monster.h"

#include "Map.h"

class UIMonsterBlood :public UInterface

{

public:

UIMonsterBlood(string tag, int x, int y, int width, int height);

~UIMonsterBlood();

void ShowBitMap()

{

vector<Monster\*> allMonster = GameSystem::GetGameObjectsWithTag<Monster>("Monster");

for (auto& i : allMonster)//全部怪物顯示血條

{

if (i->GetStatus() == ISATTACK && i->GetHP() > 0)//怪物被打中才顯示血條

OnShow(i->GetX(), i->GetY(),i->GetWidth(),i->GetHeight(), i->GetHP(), i->GetMaxHP());

}

}

private:

const int ISATTACK = 5;

void OnShow(int x, int y, int width, int height, int hp, int maxHP);

};

UIMonsterBlood.cpp

#include "StdAfx.h"

#include "UIMonsterBlood.h"

UIMonsterBlood::UIMonsterBlood(string tag, int x, int y, int width, int height) :UInterface(tag, x, y, width, height)

{

}

UIMonsterBlood::~UIMonsterBlood()

{

}

void UIMonsterBlood::OnShow(int x, int y, int width, int height, int hp, int maxHP)

{

int bloodHeight = 5;

int bloodWidth = width \* hp / maxHP;

CDC \*pDC = CDDraw::GetBackCDC();

CPen \*pp, p(PS\_NULL, 0, RGB(0, 0, 0));

pp = pDC->SelectObject(&p);

CBrush \*pg, g(RGB(255, 0, 0));

pg = pDC->SelectObject(&g);

pDC->Rectangle(x - Map::GetSX(), y - Map::GetSY() - 10, x + bloodWidth - Map::GetSX(), y + bloodHeight - Map::GetSY());

/\*釋放pen\*/

pDC->SelectObject(pp);

pDC->SelectObject(pg);

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

}

UInterface.h

#pragma once

#include "StdAfx.h"

#include <string>

using namespace std;

#include "gamelib.h"

using namespace game\_framework;

#include "resource.h"

class UInterface{

public:

UInterface();

virtual ~UInterface();

UInterface(string tag, int x, int y, int width, int height);

void SetX(int x); //設定UI的X座標

void SetY(int y); //設定UI的Y座標

void SetWidth(int width); //設定UI的width

void SetHeight(int height); //設定UI的Height

string GetTag(); //回傳Tag

int GetX(); //回傳UI的X座標

int GetY(); //回傳UI的Y座標

int GetWidth(); //回傳UI的width

int GetHeight(); //回傳UI的height

virtual void ShowBitMap(); //顯示UI圖片

protected:

CMovingBitmap UIpicture; //儲存UI圖片

vector<string> UItexts; //儲存UI文字

string tag; //分類tag

int x, y; //座標

int width, height; //寬、高

vector<CAnimation\*> ani; //儲存動畫(有必要的話)

void AddAniBitMaps(char\* pic[], int aniType, int picCount); //增加多張動畫圖片(有必要的話)

void AddAniBitMap(char\* pic, int aniType); //增加動畫圖片(有必要的話)

void LoadBitMap(int pictureID); //加入UI圖片-方法1

void LoadBitMap(char\* picAddress); //加入UI圖片-方法

virtual void LoadAni(); //載入動畫

};

UInterface.cpp

#include "StdAfx.h"

#include"UInterface.h"

UInterface::UInterface() {

this->tag = "UInterface";

x = 0;

y = 0;

width = 32;

height = 32;

}

UInterface::~UInterface()

{

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

{

delete ani[i];

}

ani.clear();

}

UInterface::UInterface(string tag, int x, int y, int width, int height){

//this->tag = "UInterface";

this->tag = tag;

this->x = x;

this->y = y;

this->width=width;

this->height=height;

}

//-----------------------SET-----------------------------//

void UInterface::SetX(int x) {

this->x = x;

}

void UInterface::SetY(int y) {

this->y = y;

}

void UInterface::SetWidth(int width) {

this->width = width;

}

void UInterface::SetHeight(int height) {

this->height = height;

}

//-----------------GET----------------------------------//

string UInterface::GetTag() {

return this->tag;

}

int UInterface::GetX() {

return this->x;

}

int UInterface::GetY() {

return this->y;

}

int UInterface::GetWidth() {

return this->width;

}

//-----------------other-------------------------------//

int UInterface::GetHeight() {

return this->height;

}

void UInterface::ShowBitMap() {

}

//-----------------PROTECTED-----------------------------//

void UInterface::AddAniBitMaps(char \* pic[], int aniType, int picCount)

{

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

{

AddAniBitMap(pic[i], aniType);

}

}

void UInterface::AddAniBitMap(char\* pic, int aniType)

{

while ((int)ani.size() <= aniType)//vector的大小不夠大

ani.push\_back(new CAnimation);//增加大小

ani[aniType]->AddBitmap(pic, RGB(255, 255, 255));

}

void UInterface::LoadAni(){}

void UInterface::LoadBitMap(int pictureID) {

UIpicture.LoadBitmap(pictureID, RGB(255, 255, 255));

}

void UInterface::LoadBitMap(char\* pictureAddress) {

UIpicture.LoadBitmap(pictureAddress, RGB(255, 255, 255));

}