國立臺北科技大學

2022 Spring 資工系物件導向程式實習

期末報告

元氣騎士 (**SoulKnight)**



第26組

資工二 109590041 范遠皓

資工二 109590043 柯瑞霖

**目錄**

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

**動機**

一開始不太清楚遊戲框架，也不確定能做到甚麼程度，雖然在當時一切都是未知數，但我們依舊選擇了一個我們認為整體偏難的遊戲，算是給自己的一個挑戰。元氣騎士不只符合了教授列出的條件，也符合了我們想要的難度，因此當下就決定是它了。

**分工**

一開始開發因為兩人同時都傳圖片，或是Coding了同一部份，造成在 git push 、 pull 、 merge時出現很多問題，所以之後改由兩人一起討論主要一人撰寫程式，或是主要一人處理圖片。

柯瑞霖：主要處理UI介面和圖片，另外參與主程式討論。

范遠皓：主要撰寫主程式，Debug，細節調整。

**遊戲說明**

簡介:

元氣騎士 (SoulKnight) 是一款俯視射擊遊戲，遊戲內關卡、敵人、道具為隨機生成，操作角色擊殺怪物進行闖關。

角色、按鍵操作:

使用WASD或 鍵盤方向鍵 操控角色移動。

使用Z或 空格鍵 來互動或使用武器 (貼近敵人可進戰攻擊)。

使用X或 直接點擊武器圖示 來切換武器。

使用C或 直接點擊技能圖示 使用技能。

遊戲關卡:

遊戲每小關 (1-1、1-2 …) 裡面有多個小房間，進入傳送門可到下一關，每五關會有 BOSS (1-5、2-5、3-5) 打完 3-5 即通關進入結束畫面。

密技:

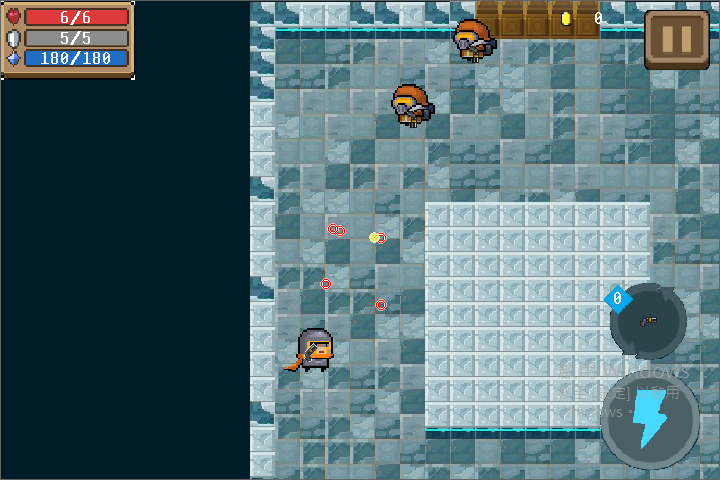
使用 N 可直接通一小關進入過場動畫，在按一次可跳過動畫。

**遊戲圖形**

選單畫面

****

進入遊戲



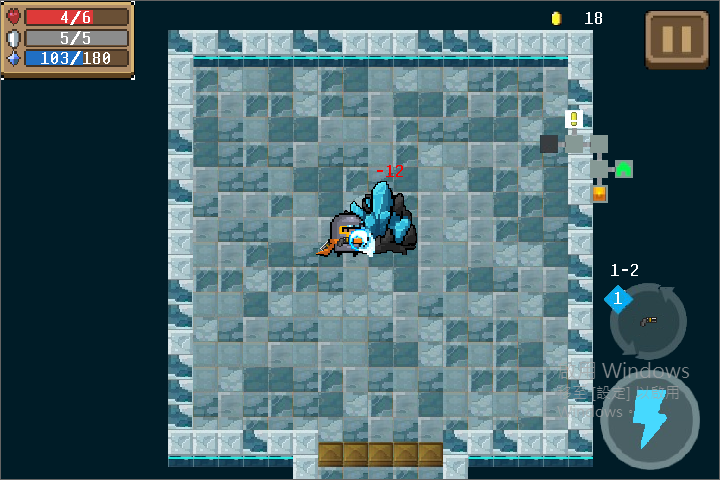
進入房間開始戰鬥

****

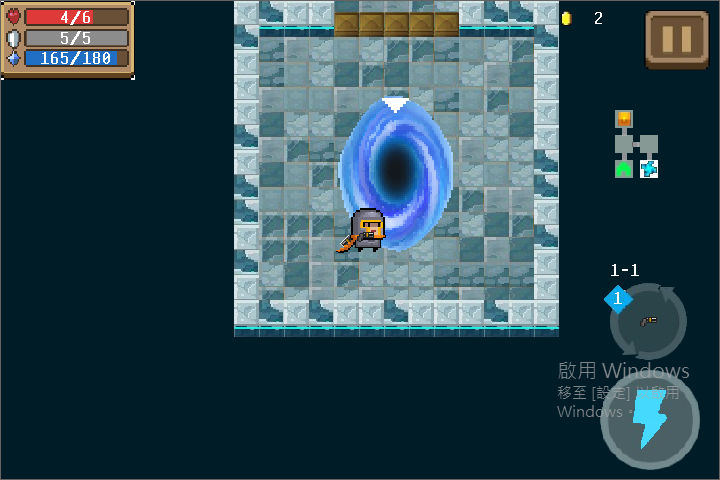
使用技能武器會變成2把

****

寶箱房間可開出武器



特殊房間有水晶礦或金礦



傳送房間

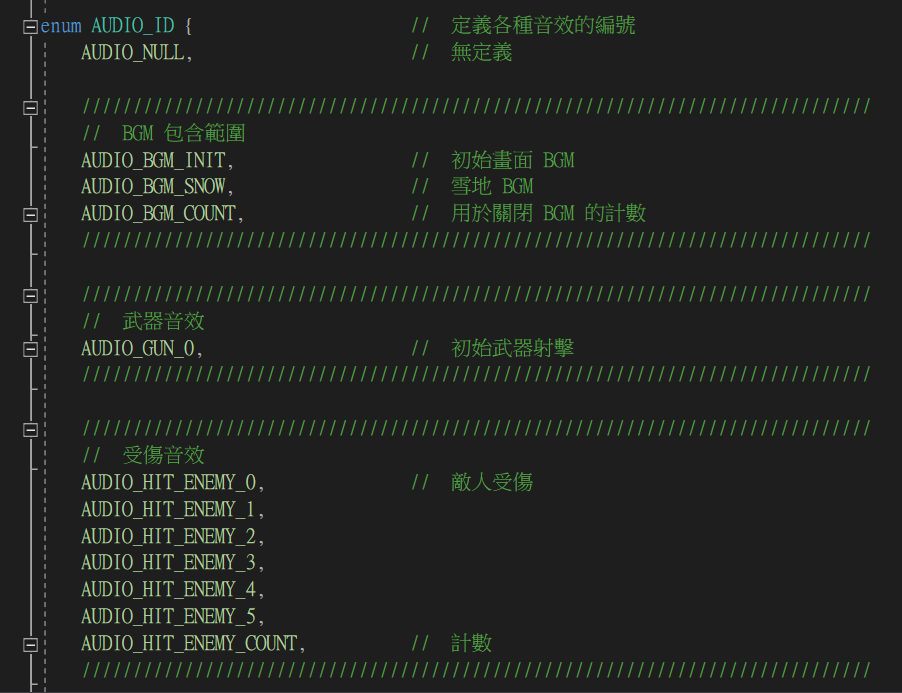


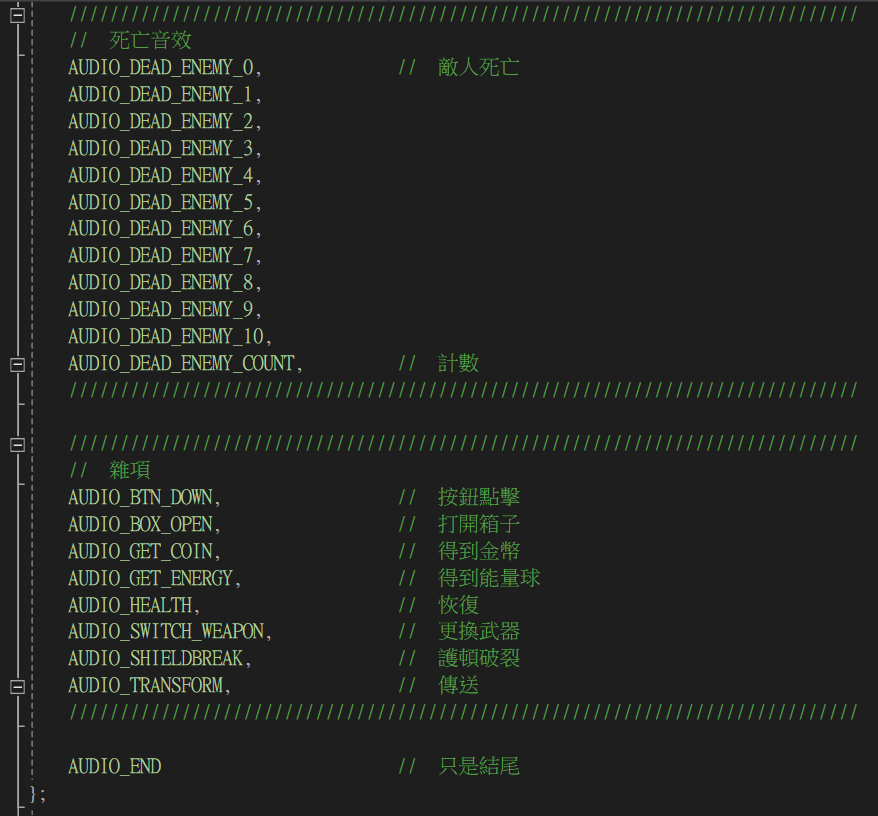
Boss房間



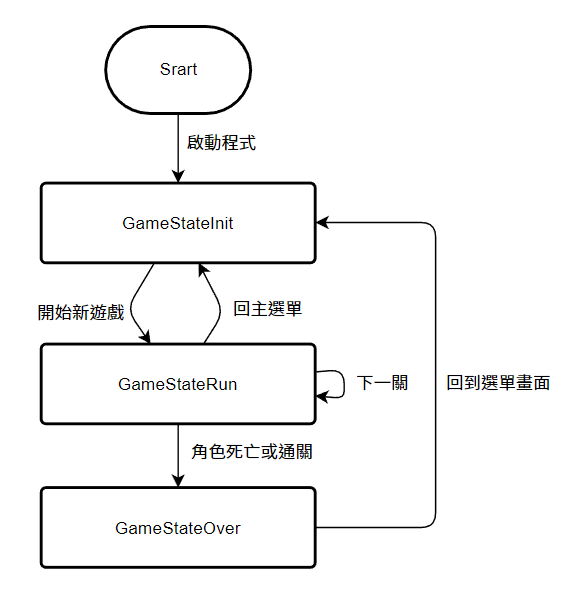
結束畫面(通到第幾關、時間、金幣、殺敵數)

**遊戲音效**

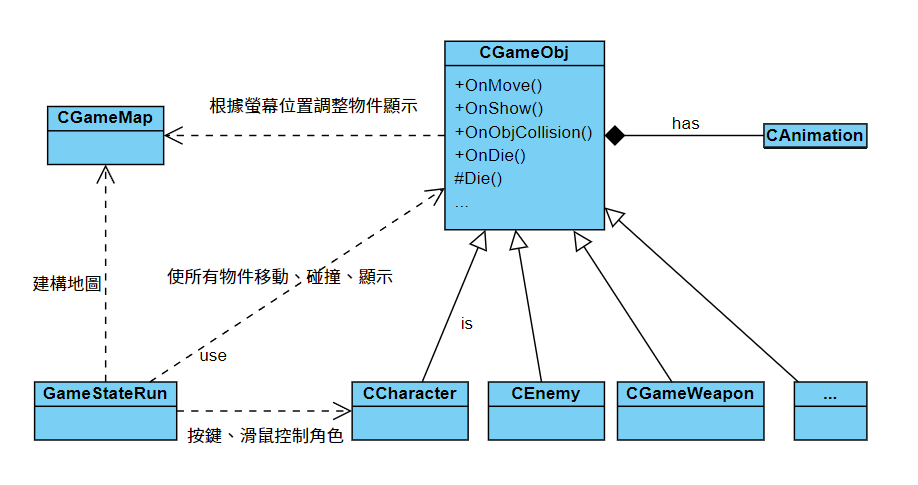
****

****

**程式架構**

****

簡單遊戲狀態變化

****

CGameMap 、 CCharacter 都是單例模式，並在GameStateRun狀態進行使用。大部分物件繼承 CGameObj (角色、敵人、武器、子彈、道具…)，物件顯示時將使用到儲存在 CGameMap 的螢幕位置在正確顯示。把建構出來的CGameObj物件加入 CGameObj 類別內的靜態 vector，並在 GameStateRun 狀態進行移動、碰撞、顯示、刪除，透過override OnMove()、OnObjCollision() 等函式來達成不同物件的互動效果。

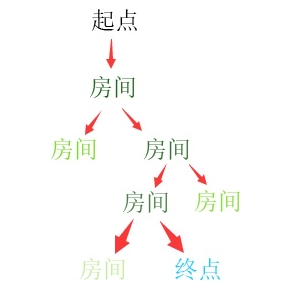
**程式類別**

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h 檔行數 | .cpp 檔行數 | 說明 |
| CGameMap | 143 | 834 | 遊戲地圖，用於建構地圖與房間資料 |
| CGameObj | 158 | 354 | 大部分遊戲內物件的 base class  用於處理在地圖上移動、碰撞、顯示 |
| CGameRoom | 537 | 600 | 管理房間的物件、控制房間內物件(怪物、寶箱、傳送門)的生成 |
| CGameFactorys | 89 | 48 | 工廠物件，將物件類別註冊到工廠後，可透過工廠建立物件 |
| CGameTool | 12 | 143 | 放一些工具函式 |
| CGameTimer | 27 | 57 | 用來計算遊戲時間的物件 |
| CCharacter | 93 | 995 | 角色的動作與功能 |
| CEnemy | 58 | 288 | 怪物、boss的 base class定義怪物屬性及功能 |
| CGameEnemys | 183 | 638 | 放所有設計的敵人類別 |
| CGameWeapon | 63 | 283 | 武器的 base class 定義武器屬性及功能 |
| CGameWeapons | 109 | 311 | 放所有設計的武器類別 |
| CGameBullet | 33 | 100 | 子彈的 base class 定義子彈屬性及功能 |
| CGameBullets | 93 | 180 | 放所有設計的子彈類別 |
| CGameArrow | 33 | 15 | 箭頭動畫類別，方便建構 |
| CGameDmgInteger | 21 | 50 | 角色及敵人受傷顯示的數字動畫 |
| CGameInteractOnceObj | 55 | 129 | 定義可與玩家互動的物件類別 |
| CGameTrackObj | 47 | 127 | 追蹤玩家的物件(金幣、能量球) |
| CUIBossHpBar | 37 | 49 | Boss 血條UI動畫 |
| CUIButton | 19 | 12 | 的按鈕UI動畫及點擊判斷 |
| CUIMinMap | 30 | 217 | 小地圖UI的動作與顯示 |
| CUISkill | 36 | 52 | 使用技能圖示UI動畫 |
| CUIWeapon | 37 | 73 | 顯示武器數值的UI動畫 |
| CUIWeaponSwitch | 34 | 40 | 切換武器圖示的UI動畫 |
| 總行數 | 1947 | 5595 |  |

**程式技術**

地圖建構:

原本的遊戲在地圖建構好像是使用圖結構，但不太了解具體怎麼生成，最後選擇使用簡單樹結構。



其他:

只有使用到基本排序、向量計算等等，並沒有用到其他厲害的演算法。

**問題及解決方法**

遊戲地圖:

在遊戲地圖的部分遇到很多問題，一開始實作時的想法是讓角色保持在螢幕中央，移動地圖來達成移動的效果，但在參考遊戲地圖與座標系統概論後，更改了實作的方式，並才可在後來時做出螢幕移動的效果。然後是地圖的建構，因想盡量符合遊戲的生成方式，更改了好幾次的生成方式，但技術不足最後選擇使用簡單的樹結構來生成，雖然不完全一樣但已經很相似了(應該吧)。

旋轉圖片:

原本遊戲的武器是可以 360 度旋轉，我們想到的方式只有用 360 張圖片，不知如何實作，但覺得使用 360 張圖片太費時，所以最後選擇只做 8 個角度。

Memory leak:

一開始使用指標的複製問題沒有處理好，以及在房間物件持有物件而沒有刪除或是double free 而造成 Memory leak 再檢查時花了很多時間，最後重新檢查撰寫 copy 以及正確 delete 才解決。

音效:

同個音效被多個物件使用會中斷播放，沒有想到該怎麼解決。

**時間表**

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 范遠皓 | 柯瑞霖 | 說明 |
| 1 | 3 | 2 | 範例練習 |
| 2 | 2 | 2 | 遊戲選擇 |
| 3 | 5 | 5 | Git 練習 |
| 4 | 3 | 2 | 簡單地圖 |
| 5 | 5 | 4 | 動畫、base class |
| 6 | 20 | 15 | 碰撞、射擊 |
| 7 | 13 | 11 | 房間建構、UI |
| 8 | 8 | 6 | 地圖調整 |
| 9 | 10 | 5 | 解決 Memory leak、傳送門 |
| 10 | 15 | 17 | 通關寶箱、細節調整 |
| 11 | 6 | 10 | 小地圖建立 |
| 12 | 30 | 20 | 架構調整、金幣、能量球、藥水、螢幕移動、地圖生成、bug修正 |
| 13 | 14 | 12 | 遊戲選單介面、暫停按鈕、音效 |
| 14 | 14 | 12 | 武器轉向、工廠建立 |
| 15 | 22 | 10 | 技能、武器切換、近戰攻擊、UI |
| 16 | 15 | 10 | 新增武器、敵人、BOSS、UI |
| 17 | 15 | 5 | About按鈕、安裝設定、最後調整、報告 |
| 總時數 | 200 | 148 |  |

**貢獻比例**

范遠皓 50%

柯瑞霖 50%

**自我檢核表**

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 項目 | 完成否 | 無法完成原因 |
| 1 | 解決 Memory leak | 已完成 |  |
| 2 | 自訂遊戲 Icon | 已完成 |  |
| 3 | 全螢幕啟動 | 已完成 |  |
| 4 | 有 About 畫面 | 已完成 |  |
| 5 | 初始畫面說明按鍵及滑鼠之密技與用法 | 已完成 |  |
| 6 | 上傳 setup/apk/source 檔 | 已完成 |  |
| 7 | Setup 檔可正常執行 | 已完成 |  |
| 8 | 報告字型、點數、對齊、行距、頁碼格式正確 | 已完成 |  |

**收獲**

范遠皓:

經過這學期的課程，有學習到單例模式、一些工廠設計模式，例如: 簡單工廠、工廠方式模式和抽象工廠模式等等，還有遊戲動畫的設計，以及更了解 .cpp .h實作與介面的分別與意義，最後就是 debug 工具的使用與測試的重要性，這次專案覺得麻煩所以沒有寫測試，但在開發過程中出 bug 或 memory leak 會很難排除，現在的專案還很小，未來如果要繼續開發程式，在撰寫測試方面也應該要加強。

柯瑞霖:

經過這次的專題，我在實作中了解到了遊戲的架構，和一些遊戲動畫上的設計，對於圖片處理的技術也更上一層樓了，然後也是因為這次專題，才去學怎麼對APK拆包，並用python寫了個分類器，對拆出來的圖片做分類。收穫最多的是我對指標和物件導向的概念又更加熟悉了。

**心得、感想**

范遠皓:

經過這次專案，發現自己還有很多的不足，程式很多地方都不完善，也收穫很多經驗，有一段時間想法很多所以一直在寫，每完成一個功能都會很開心，比較可惜地方的是，原本的計畫，是要做的跟原本遊戲一樣的可以選角色、有初始大廳等等，但因為我不熟悉設計方式、寫程式較慢，過程也遇到很多問題，所以遊戲地圖、怪物、武器、道具數量也都很少，沒辦法原本遊戲相符，最後就更改計畫大廳改成選單、只有一張地圖，希望經過這次專案的經驗，下次再開發專案時，能更好規劃計畫，並精進自己的程式能力，最後能交出自己覺得滿分的作品。

柯瑞霖:

在實作這次的專題中，雖然過程磕磕絆絆，但從中學到了非常多東西，對我而言也是十分的有趣。一開始的時候對於遊戲功能和架構的設計還沒那麼有雛型，所以前面都還在理解，尤其是到了製作地圖的部分，我們卡了一段時間，當時真的心很累，一度想要放棄。但經過我們不懈的努力，地圖的問題最終被我們成功解決了，自從這一刻，我們突然信心大增，進度可以說是突飛猛進，也因當時有了基本的架構和前面的經驗，我們後面遇到的問題越來越少。雖然中間多少會遇到挫折，但我們愈挫愈勇，隨著解決的問題和進度越多，我們從中獲得的成就感也就越多，最後也算是把原本遊戲的還原度做的挺高了(自己覺得)。最後想說，這次的專題，我真的從中學到了非常多，也希望我可以藉由這些經驗，將來可以自己設計並製作一款遊戲。然後處裡圖片的時候，真的很枯燥...。

**附錄**

**mygame.h**

#include "CGameObj.h

#include "CGameMap.h"

#include "CCharacter.h"

#include "CEnemy.h"

#include "CGameRoom.h"

#include "CGameTool.h"

#include "CGameFactorys.h"

#include "CGameWeapons.h"

#include "CGameEnemys.h"

#include "CGameTimer.h"

#include "CUIMinMap.h"

#include "CUIButton.h"

#include "CUIWeapon.h"

#include "CUISkill.h"

#include "CUIWeaponSwitch.h"

#include "CUIBossHpBar.h"

namespace game\_framework {

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

// Constants

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

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

// 這個class為遊戲的遊戲開頭畫面物件

// 每個Member function的Implementation都要弄懂

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

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

void 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

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

void OnMove();

private:

// 遊戲內狀態

enum class STATE {

RUN, // 正常執行

ABOUT // 轉換 about 畫面

};

STATE gameInitState;

CMovingBitmap background, title, aboutBackGround; // 背景圖

CAnimation noteboard; // 遊戲說明

CAnimation start; // 點擊開始動畫

CUIButton btn\_newgame, btn\_gamenote, btn\_about, btn\_close; // 按鈕

int btn\_posy, btn\_movey, board\_posy, board\_movey; // 按鈕位移

bool isLoad; // 不知為何 OnBeginState() 會比 OnInit() 先執行 用來判斷是否已下載資料

};

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

// 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:

// 遊戲內狀態

enum class STATE{

RUN, // 正常執行

PAUSE, // 暫停

GOTONEXT, // 正常執行

PLAYERDIE // 玩家死亡

};

STATE gameRunState;

// UI

CInteger HP, MP, SP, MAXHP, MAXMP, MAXSP, GAMELEVEL, GOLDINTGER;

CMovingBitmap HPBACKGROUND, SLASH, HPBAR, SPBAR, MPBAR, MINUS, GOLD, pause\_UI, setup\_UI;

CUIButton btn\_pause, btn\_goBack, btn\_continue, btn\_setup, btn\_BGM, btn\_soundEffect;

CUIMinMap minMap;

int dMinMap; // 小地圖位移

int btn\_posy, UI\_posy, setup\_posy; // 暫停介面位移

// GAMERUN

CGameMap\* gameMap; // 地圖

CCharacter\* character; // 角色

CGameRoom\* Rooms[MYMAXNOFROOM][MYMAXNOFROOM]; // 房間物件

// GOTONEXT

CAnimation cutscenes;

int cutscenesCounter; // 過場動畫計數

// debug

CInteger debugx, debugy;

// function

void GameEnd(); // 遊戲結束重製

};

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

// 這個class為遊戲的結束狀態(Game Over)

// 每個Member function的Implementation都要弄懂

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

class CGameStateOver : public CGameState {

public:

CGameStateOver(CGame \*g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

void 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

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CUIButton btn\_statectl;

CInteger counterDown, enemyDie, coin, SpendTime, gameLevel;

CMovingBitmap background, colon, minus;

CAnimation player;

enum class STATE {start, runAnima, gotoInit };

STATE state; // 進行狀態

int counter; // 倒數之計數器

int spendSecond; // 儲存花費遊戲時間

int animaRate; // 動畫完成進度

const int animaFinishTime; // 多久要完成動畫

int endCounter;

};

}

**mygame.cpp**

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

namespace game\_framework {

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

// 這個class為遊戲的遊戲開頭畫面物件

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

CGameStateInit::CGameStateInit(CGame \*g)

: CGameState(g)

{

isLoad = false;

}

void CGameStateInit::OnInit()

{

if (isLoad)

return;

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

// Audio

// BGM

CAudio::Instance()->Load(AUDIO\_BGM\_INIT, "sounds\\BGM\\bgm\_1Low.wav");

CAudio::Instance()->Load(AUDIO\_BGM\_SNOW, "sounds\\BGM\\bgm\_5Low.wav");

// BTN

CAudio::Instance()->Load(AUDIO\_BTN\_DOWN, "sounds\\Btn\\fx\_btn1.wav");

// BOX

CAudio::Instance()->Load(AUDIO\_BOX\_OPEN, "sounds\\Box\\fx\_chest\_open.wav");

// COIN, ENERGY

CAudio::Instance()->Load(AUDIO\_GET\_COIN, "sounds\\Coin\\fx\_coin.wav");

CAudio::Instance()->Load(AUDIO\_GET\_ENERGY, "sounds\\Coin\\fx\_energy.wav");

// HEALTH

CAudio::Instance()->Load(AUDIO\_HEALTH, "sounds\\Health\\fx\_healthpot.wav");

CAudio::Instance()->Load(AUDIO\_SHIELDBREAK, "sounds\\Health\\shield\_break.wav");

// AUDIO\_SWITCH\_WEAPON

CAudio::Instance()->Load(AUDIO\_SWITCH\_WEAPON, "sounds\\Player\\fx\_switch.wav");

// AUDIO\_TRANSFORM

CAudio::Instance()->Load(AUDIO\_TRANSFORM, "sounds\\Player\\fx\_transform.wav");

// WEAPON

CAudio::Instance()->Load(AUDIO\_GUN\_0, "sounds\\Shoot\\fx\_gun\_1.wav");

// HIT

CAudio::Instance()->Load(AUDIO\_HIT\_ENEMY\_0, "sounds\\Hit\\fx\_hit.wav");

CAudio::Instance()->Load(AUDIO\_HIT\_ENEMY\_1, "sounds\\Hit\\fx\_hit\_p1.wav");

CAudio::Instance()->Load(AUDIO\_HIT\_ENEMY\_2, "sounds\\Hit\\fx\_hit\_p2.wav");

CAudio::Instance()->Load(AUDIO\_HIT\_ENEMY\_3, "sounds\\Hit\\fx\_hit\_p3.wav");

CAudio::Instance()->Load(AUDIO\_HIT\_ENEMY\_4, "sounds\\Hit\\fx\_hit\_p4.wav");

CAudio::Instance()->Load(AUDIO\_HIT\_ENEMY\_5, "sounds\\Hit\\fx\_hit\_p5.wav");

// DEAD

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_0, "sounds\\Dead\\fx\_dead.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_1, "sounds\\Dead\\fx\_dead\_cell\_absorb.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_2, "sounds\\Dead\\fx\_dead2.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_3, "sounds\\Dead\\fx\_dead3.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_4, "sounds\\Dead\\fx\_dead4.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_5, "sounds\\Dead\\fx\_dead5.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_6, "sounds\\Dead\\fx\_dead6.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_7, "sounds\\Dead\\fx\_dead7.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_8, "sounds\\Dead\\fx\_dead8.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_9, "sounds\\Dead\\fx\_dead\_cell\_absorb.wav");

CAudio::Instance()->Load(AUDIO\_DEAD\_ENEMY\_10, "sounds\\Dead\\fx\_dead\_note.wav");

// 物件註冊

Registrar::Registrars();

// 遊戲物件初始化

CGameObj::Init();

CGameRoom::Init();

// UI

background.LoadBitmap(IDB\_Homepage);

title.LoadBitmap(IDB\_Homepage\_title, RGB(0, 0, 0));

aboutBackGround.LoadBitmap(IDB\_aboutBackGround);

start.AddBitmap(IDB\_start0, RGB(0, 0, 0));

start.AddBitmap(IDB\_start1, RGB(0, 0, 0));

start.AddBitmap(IDB\_start2, RGB(0, 0, 0));

start.AddBitmap(IDB\_start3, RGB(0, 0, 0));

start.AddBitmap(IDB\_start4, RGB(0, 0, 0));

start.AddBitmap(IDB\_start5, RGB(0, 0, 0));

start.AddBitmap(IDB\_start6, RGB(0, 0, 0));

start.AddBitmap(IDB\_start7, RGB(0, 0, 0));

start.AddBitmap(IDB\_start6, RGB(0, 0, 0));

start.AddBitmap(IDB\_start5, RGB(0, 0, 0));

start.AddBitmap(IDB\_start4, RGB(0, 0, 0));

start.AddBitmap(IDB\_start3, RGB(0, 0, 0));

start.AddBitmap(IDB\_start2, RGB(0, 0, 0));

start.AddBitmap(IDB\_start1, RGB(0, 0, 0));

start.SetDelayCount(3);

btn\_newgame.AddBitmap(IDB\_newgame0, RGB(0, 0, 0));

btn\_newgame.AddBitmap(IDB\_newgame1, RGB(0, 0, 0));

btn\_gamenote.AddBitmap(IDB\_gamenote0, RGB(0, 0, 0));

btn\_gamenote.AddBitmap(IDB\_gamenote1, RGB(0, 0, 0));

noteboard.AddBitmap(IDB\_note\_test, RGB(0, 0, 0));

btn\_about.AddBitmap(IDB\_BTN\_about\_0, RGB(0, 0, 0));

btn\_about.AddBitmap(IDB\_BTN\_about\_1, RGB(0, 0, 0));

btn\_close.AddBitmap(IDB\_BTN\_close\_0, RGB(0, 0, 0));

btn\_close.AddBitmap(IDB\_BTN\_close\_1, RGB(0, 0, 0));

isLoad = true;

}

void CGameStateInit::OnBeginState()

{

// 按鈕位置、位移設定

board\_posy = -330;

board\_movey = -20;

btn\_posy = SIZE\_Y;

btn\_movey = 10;

gameInitState = STATE::RUN;

// 避免未先執行過 OnInit

this->OnInit();

// 播放 BGM

CAudio::Instance()->Play(AUDIO\_BGM\_INIT);

// 初始化殺敵數

CEnemy::ResetDieAmount();

// 初始化地圖物件關卡數

CGameMap::Instance()->AddGameLevel(-CGameMap::Instance()->GetGameLevel());

// 玩家物件初始化

CCharacter::Instance()->Init();

// boss 血條 UI初始化

CUIBossHpBar::Instance().SetEnable(false);

// 圖片、動畫、按鈕位置設定

background.SetTopLeft(0, 0);

title.SetTopLeft(20, 10);

start.SetTopLeft((SIZE\_X - start.Width()) / 2 - 10, SIZE\_Y - start.Height() - 50);

aboutBackGround.SetTopLeft(0, SIZE\_Y);

btn\_about.SetTopLeft(10, SIZE\_Y - btn\_about.Height() - 20);

}

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

{

switch (gameInitState)

{

case game\_framework::CGameStateInit::STATE::RUN:

{

if (btn\_newgame.PointIn(point.x, point.y)) {

if (!btn\_newgame.IsFinalBitmap()) {

btn\_newgame.OnMove();

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

}

else if (btn\_gamenote.PointIn(point.x, point.y)) {

if (!btn\_gamenote.IsFinalBitmap()) {

btn\_gamenote.OnMove();

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

}

else if (btn\_about.PointIn(point.x, point.y)) {

if (!btn\_about.IsFinalBitmap()) {

btn\_about.OnMove();

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

}

else if (board\_movey < 0) {

btn\_movey \*= -1;

}

break;

}

case game\_framework::CGameStateInit::STATE::ABOUT:

{

if (btn\_close.PointIn(point.x, point.y)) {

if (!btn\_close.IsFinalBitmap()) {

btn\_close.OnMove();

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

}

break;

}

default:

break;

}

}

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

{

switch (gameInitState)

{

case game\_framework::CGameStateInit::STATE::RUN:

{

if (btn\_newgame.PointIn(point.x, point.y)) {

Sleep(150); // 延遲一下再進入遊戲

// 停止 BGM

CAudio::Instance()->Stop(AUDIO\_BGM\_INIT);

// 開始計時

CGameTimer::Instance().SetStartPoint(-1); // 小於 0 起點會被設為當前時間

GotoGameState(GAME\_STATE\_RUN);

}

else if (btn\_gamenote.PointIn(point.x, point.y)) {

board\_movey \*= -1;

}

else if (btn\_about.PointIn(point.x, point.y)) {

gameInitState = STATE::ABOUT;

}

else if (board\_movey > 0) {

board\_movey \*= -1;

}

btn\_newgame.Reset();

btn\_gamenote.Reset();

btn\_about.Reset();

break;

}

case game\_framework::CGameStateInit::STATE::ABOUT:

{

if (btn\_close.PointIn(point.x, point.y))

{

gameInitState = STATE::RUN;

}

btn\_close.Reset();

break;

}

default:

break;

}

}

void CGameStateInit::OnMove()

{

start.OnMove();

btn\_posy += btn\_movey;

if (btn\_posy > SIZE\_Y) {

btn\_posy = SIZE\_Y;

}

else if (btn\_posy < (SIZE\_Y - btn\_newgame.Height() - 20)) {

btn\_posy = SIZE\_Y - btn\_newgame.Height() - 20;

}

board\_posy += board\_movey;

if (board\_posy < -noteboard.Height()) {

board\_posy = -noteboard.Height();

}

else if (board\_posy > (30)) {

board\_posy = 30;

}

switch (gameInitState)

{

case game\_framework::CGameStateInit::STATE::RUN: // 一般畫面

{

if (aboutBackGround.Top() < SIZE\_Y)

{

aboutBackGround.SetTopLeft(aboutBackGround.Left(), aboutBackGround.Top() + 40);

}

else

{

aboutBackGround.SetTopLeft(aboutBackGround.Left(), SIZE\_Y);

}

break;

}

case game\_framework::CGameStateInit::STATE::ABOUT: // 進入 about 畫面

{

if (aboutBackGround.Top() > 0)

{

aboutBackGround.SetTopLeft(aboutBackGround.Left(), aboutBackGround.Top() - 40);

}

else

{

aboutBackGround.SetTopLeft(aboutBackGround.Left(), 0);

}

break;

}

default:

break;

}

}

void CGameStateInit::OnShow()

{

// 背景、動畫

background.ShowBitmap();

title.ShowBitmap();

if (btn\_posy == SIZE\_Y) {

start.OnShow();

}

noteboard.SetTopLeft(110, board\_posy);

noteboard.OnShow();

// 按鈕

btn\_newgame.SetTopLeft(90, btn\_posy);

btn\_newgame.OnShow();

btn\_gamenote.SetTopLeft(120 + btn\_newgame.Width(), btn\_posy);

btn\_gamenote.OnShow();

btn\_about.OnShow();

// about 畫面

aboutBackGround.ShowBitmap();

btn\_close.SetTopLeft(aboutBackGround.Left() + 472, aboutBackGround.Top() + 120);

btn\_close.OnShow();

}

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

// 這個class為遊戲的結束狀態(Game Over)

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

CGameStateOver::CGameStateOver(CGame \*g)

: CGameState(g), animaFinishTime(2 \* GAME\_ONE\_SECONED)

{

}

void CGameStateOver::OnMove()

{

counter--;

if (counter < 0)

{

state = STATE::gotoInit;

}

switch (state)

{

case game\_framework::CGameStateOver::STATE::start:

if (counter < 6 \* GAME\_ONE\_SECONED) // 剩 6 秒開始跑動畫

{

state = STATE::runAnima;

}

break;

case game\_framework::CGameStateOver::STATE::runAnima:

{

if(animaRate < animaFinishTime)

animaRate++;

else if(!player.IsFinalBitmap() && CGameMap::Instance()->GetGameLevel() != 15)

{

player.OnMove();

}

break;

}

case game\_framework::CGameStateOver::STATE::gotoInit:

{

animaRate = animaFinishTime;

if (!player.IsFinalBitmap() && CGameMap::Instance()->GetGameLevel() != 15)

{

player.OnMove();

}

endCounter--;

if (counter < 0 || endCounter < 0)

{

GotoGameState(GAME\_STATE\_INIT);

}

break;

}

default:

break;

}

}

void CGameStateOver::OnBeginState()

{

// 設定倒數秒數

counter = 10 \* GAME\_ONE\_SECONED; // 10 seconds

// 初始化狀態

state = STATE::start;

// 重置動畫進度

animaRate = 0;

// 重置動畫

player.Reset();

// 設定按兩下的延遲以便看到動畫完成

endCounter = GAME\_ONE\_SECONED;

// 取得計時時間

spendSecond = CGameTimer::ConvertToSeconds(CGameTimer::Instance().GetTimingTime());

}

void CGameStateOver::OnInit()

{

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

// 背景

background.LoadBitmap(IDB\_gameover\_background);

background.SetTopLeft(0, 0);

// 按鈕

btn\_statectl.AddBitmap(IDB\_BTN\_continue\_0, RGB(255, 255, 255));

btn\_statectl.AddBitmap(IDB\_BTN\_continue\_1, RGB(255, 255, 255));

btn\_statectl.SetTopLeft((SIZE\_X - btn\_statectl.Width()) / 2, SIZE\_Y - btn\_statectl.Height() - 10);

// 到計時

counterDown.SetTopLeft(btn\_statectl.Left() + (btn\_statectl.Width() / 2) + 20, btn\_statectl.Top() + ((btn\_statectl.Height() - counterDown.GetHeight()) / 2));

// 殺敵數

enemyDie.SetTopLeft(390, 245);

// 取得金幣

coin.SetColor(CInteger::Color::YELLOW);

coin.SetTopLeft(240, 245);

// 冒號

colon.LoadBitmap(IDB\_COLON, RGB(0, 0, 0));

// 減號

minus.LoadBitmap(IDB\_MINUS, RGB(0, 0, 0));

// 腳色

player.AddBitmap(IDB\_gameover\_player\_1, RGB(0, 0, 0));

player.AddBitmap(IDB\_gameover\_player\_0, RGB(0, 0, 0));

ShowInitProgress(100);

}

void CGameStateOver::OnKeyDown(UINT, UINT, UINT)

{

if (!btn\_statectl.IsFinalBitmap()) {

btn\_statectl.OnMove();

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

}

void CGameStateOver::OnKeyUp(UINT, UINT, UINT)

{

switch (state)

{

case game\_framework::CGameStateOver::STATE::start:

state = STATE::runAnima;

break;

case game\_framework::CGameStateOver::STATE::runAnima:

state = STATE::gotoInit;

break;

default:

break;

}

btn\_statectl.Reset();

}

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

{

if (btn\_statectl.PointIn(point.x, point.y)) {

if (!btn\_statectl.IsFinalBitmap()) {

btn\_statectl.OnMove();

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

}

}

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

{

if (btn\_statectl.PointIn(point.x, point.y)) {

switch (state)

{

case game\_framework::CGameStateOver::STATE::start:

state = STATE::runAnima;

break;

case game\_framework::CGameStateOver::STATE::runAnima:

state = STATE::gotoInit;

break;

default:

break;

}

}

btn\_statectl.Reset();

}

void CGameStateOver::OnShow()

{

// 背景

background.ShowBitmap();

// 按鈕

btn\_statectl.OnShow();

// 倒數數字

counterDown.SetInteger(counter / GAME\_ONE\_SECONED);

counterDown.ShowBitmap(false);

// 殺敵數

enemyDie.SetInteger(CEnemy::GetDieAmount() \* animaRate / animaFinishTime);

enemyDie.ShowBitmap(false);

// 取得金幣數

coin.SetInteger(CCharacter::Instance()->GetGold() \* animaRate / animaFinishTime);

coin.ShowBitmap(false);

// 花費遊戲時間

int temp;

temp = spendSecond \* animaRate / animaFinishTime;

SpendTime.SetTopLeft(80, 245);

SpendTime.SetInteger(temp / 3600); // 小時

SpendTime.ShowBitmap(false);

colon.SetTopLeft(SpendTime.GetLeft() + SpendTime.GetWidth() \* SpendTime.GetLen(), 245);

colon.ShowBitmap();

SpendTime.SetTopLeft(colon.Left() + colon.Width(), 245);

SpendTime.SetInteger((temp / 60) % 60); // 分鐘

SpendTime.ShowBitmap(false);

colon.SetTopLeft(SpendTime.GetLeft() + SpendTime.GetWidth() \* SpendTime.GetLen(), 245);

colon.ShowBitmap();

SpendTime.SetTopLeft(colon.Left() + colon.Width(), 245);

SpendTime.SetInteger(temp % 60); // 秒

SpendTime.ShowBitmap(false);

// 關卡數

temp = (1 + CGameMap::Instance()->GetGameLevel()) \* animaRate / animaFinishTime;

if (temp == 16)

temp--;

minus.SetTopLeft(15 + 45 \* temp, 185);

minus.ShowBitmap();

gameLevel.SetTopLeft(minus.Left() - gameLevel.GetWidth(), 185);

gameLevel.SetInteger(temp < 1 ? 0 : 1 + (temp - 1) / 5);

gameLevel.ShowBitmap(false);

gameLevel.SetTopLeft(minus.Left() + minus.Width(), 185);

gameLevel.SetInteger(temp < 1 ? 0 : 1 + (temp - 1) % 5);

gameLevel.ShowBitmap(false);

// 角色

player.SetTopLeft(15 - (player.Width() / 2) + 45 \* temp, 100);

player.OnShow();

}

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

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

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

CGameStateRun::CGameStateRun(CGame \*g)

: CGameState(g)//, NUMBALLS(28)

{

}

CGameStateRun::~CGameStateRun()

{

//delete [] ball;

CGameObj::FreeAllObj();

}

void CGameStateRun::OnBeginState()

{

// 初始化狀態

gameRunState = STATE::RUN;

cutscenesCounter = 2 \* GAME\_ONE\_SECONED;

// UI

dMinMap = 0;

btn\_setup.Reset();

// 暫停介面

btn\_posy = -300;

UI\_posy = 0;

setup\_posy = SIZE\_Y;

// UI 設定位置

CUISkill::Instance().SetTopLeft(SIZE\_X - CUISkill::Instance().Width() - 20, SIZE\_Y - CUISkill::Instance().Height() - 10);

CUIWeaponSwitch::Instance().SetTopLeft(CUISkill::Instance().Left() + ((CUISkill::Instance().Width() - CUIWeaponSwitch::Instance().Width()) >> 1) - 5, CUISkill::Instance().Top() - CUIWeaponSwitch::Instance().Height() - 10);

// Game

// 清空地圖物件

CGameObj::FreeAllObj();

// 生成地圖

gameMap->GenerateMap();

// 將房間資訊傳入小地圖

minMap.SetRoom(gameMap->GetRooms());

// 重設角色屬性

character->Reset();

character->SetXY(MYMAPWIDTH \* gameMap->GetRoom(MYORGROOM, MYORGROOM)->CenterX(),

MYMAPHIGH \* gameMap->GetRoom(MYORGROOM, MYORGROOM)->CenterY()); //設定初始位置

CGameObj::AddObj(character);

// 房間建構

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

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

{

CGameRoom\* room = new CGameRoom(gameMap->GetRoom(i, j));

room->Initialization(gameMap);

CGameObj::AddObj(room);

Rooms[i][j] = room;

}

// Audio

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

}

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

{

// UI button

if (btn\_setup.IsFinalBitmap())

{

if (setup\_posy > SIZE\_Y - setup\_UI.Height())

setup\_posy -= 15;

}

else

{

if (setup\_posy < SIZE\_Y)

setup\_posy += 15;

}

switch (gameRunState)

{

case game\_framework::CGameStateRun::STATE::PAUSE: // 暫停處理

{

if (UI\_posy > -HPBACKGROUND.Height() - 50)

UI\_posy -= 10;

if (btn\_posy < 100)

btn\_posy += 50;

// 暫停時間不列入計算

CGameTimer::Instance().SetStartPoint(CGameTimer::Instance().GetStartPoint() + 1);

break;

}

case game\_framework::CGameStateRun::STATE::GOTONEXT: // 進下一關

{

cutscenesCounter--;

this->GameEnd();

cutscenes.OnMove();

if (cutscenesCounter < 0)

{

GotoGameState(GAME\_STATE\_RUN);

}

break;

}

case game\_framework::CGameStateRun::STATE::PLAYERDIE: // 玩家死亡

{

this->GameEnd();

GotoGameState(GAME\_STATE\_OVER);

break;

}

case game\_framework::CGameStateRun::STATE::RUN: // 正常執行

{

if (btn\_posy > -300)

btn\_posy -= 50;

// 武器 UI 調動

CUIWeapon::Instance().MoveUI();

// GAME

gameMap->OnMove(character->CenterX(), character->CenterY());

CGameObj::UpdateObjs();

// 移動、新增物件

for (int i = 0; i < (int)CGameObj::\_allObj.size(); i++)

{

CGameObj\* obj = CGameObj::\_allObj.at(i);

if (obj->IsEnable())

{

obj->OnMove(gameMap);

}

else if (obj->IsDie())

{

obj->OnDie(gameMap);

}

}

// 處理碰撞

for (int i = 0; i < (int)CGameObj::\_allObj.size(); i++)

{

if (!CGameObj::\_allObj.at(i)->IsCollision())

continue;

for (int j = i + 1; j < (int)CGameObj::\_allObj.size(); j++)

if (CGameObj::\_allObj.at(j)->IsCollision() && CGameObj::\_allObj.at(i)->Collision(CGameObj::\_allObj.at(j)))

{

CGameObj::\_allObj.at(i)->OnObjCollision(gameMap, CGameObj::\_allObj.at(j));

CGameObj::\_allObj.at(j)->OnObjCollision(gameMap, CGameObj::\_allObj.at(i));

}

}

// 角色死亡

if (!character->IsEnable() && !character->IsDie())

{

gameRunState = STATE::PLAYERDIE;

}

// UI小地圖

minMap.OnMove();

int px = minMap.GetPlayerIn(0), py = minMap.GetPlayerIn(1);

if (Rooms[px][py]->IsStrat())// 進一般、Boss房間小地圖收起

{

if (dMinMap < 180)

dMinMap += 5;

}

else

{

if (dMinMap > 0)

dMinMap -= 5;

}

break;

}

default:

break;

}

}

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

{

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

// GAME

gameMap = CGameMap::Instance();

character = CCharacter::Instance();

ShowInitProgress(50);

// UI

CInteger::LoadBitmap(); // 數字圖片

HPBACKGROUND.LoadBitmap(IDB\_UI\_HP); // bar 背景

HPBAR.LoadBitmap(IDB\_UI\_HPBar); // bar 圖片

SPBAR.LoadBitmap(IDB\_UI\_SPBar);

MPBAR.LoadBitmap(IDB\_UI\_MPBar);

SLASH.LoadBitmap(IDB\_Slash, RGB(0, 0, 0)); // 斜線圖片

MINUS.LoadBitmap(IDB\_MINUS, RGB(0, 0, 0)); // 減號圖片

GOLD.LoadBitmap(IDB\_gold\_0, RGB(255, 255, 255)); // 金幣圖片

MAXHP.SetTopLeft(82, 7);

MAXSP.SetTopLeft(82, 28);

MAXMP.SetTopLeft(82, 48);

HPBACKGROUND.SetTopLeft(0, 0);

// UI小地圖

minMap.LoadBitmap();

// 暫停UI

pause\_UI.LoadBitmap(IDB\_pause\_UI, RGB(255, 255, 255));

setup\_UI.LoadBitmap(IDB\_setup\_UI, RGB(255, 255, 255));

// 按鈕

btn\_goBack.AddBitmap(IDB\_BTN\_goback\_0, RGB(255, 255, 255));

btn\_goBack.AddBitmap(IDB\_BTN\_goback\_1, RGB(255, 255, 255));

btn\_continue.AddBitmap(IDB\_BTN\_continue\_0, RGB(255, 255, 255));

btn\_continue.AddBitmap(IDB\_BTN\_continue\_1, RGB(255, 255, 255));

btn\_pause.AddBitmap(IDB\_BTN\_pause\_0, RGB(255, 255, 255));

btn\_pause.AddBitmap(IDB\_BTN\_pause\_1, RGB(255, 255, 255));

btn\_setup.AddBitmap(IDB\_BTN\_setup\_0, RGB(255, 255, 255));

btn\_setup.AddBitmap(IDB\_BTN\_setup\_1, RGB(255, 255, 255));

btn\_BGM.AddBitmap(IDB\_BTN\_BGM\_0, RGB(255, 255, 255));

btn\_BGM.AddBitmap(IDB\_BTN\_BGM\_1, RGB(255, 255, 255));

btn\_soundEffect.AddBitmap(IDB\_BTN\_soundEffect\_0, RGB(255, 255, 255));

btn\_soundEffect.AddBitmap(IDB\_BTN\_soundEffect\_1, RGB(255, 255, 255));

// 過場動畫

cutscenes.SetDelayCount(1);

cutscenes.AddBitmap(IDB\_TransferGate\_0, RGB(255, 255, 255));

cutscenes.AddBitmap(IDB\_TransferGate\_1, RGB(255, 255, 255));

cutscenes.AddBitmap(IDB\_TransferGate\_2, RGB(255, 255, 255));

cutscenes.AddBitmap(IDB\_TransferGate\_3, RGB(255, 255, 255));

cutscenes.AddBitmap(IDB\_TransferGate\_4, RGB(255, 255, 255));

cutscenes.AddBitmap(IDB\_TransferGate\_5, RGB(255, 255, 255));

cutscenes.AddBitmap(IDB\_TransferGate\_6, RGB(255, 255, 255));

cutscenes.AddBitmap(IDB\_TransferGate\_7, RGB(255, 255, 255));

cutscenes.SetTopLeft((SIZE\_X - cutscenes.Width()) / 2, 100);

}

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

{

switch (gameRunState)

{

case game\_framework::CGameStateRun::STATE::RUN:

{

if (nChar == 0x47) // 按 G 扣血

character->TakeDmg(9999);

else if (nChar == 78) // 按 N 進入下一關卡

{

//test

gameMap->AddGameLevel(1);

if (gameMap->GetGameLevel() == 15) // 打完 3-5 通關

{

gameRunState = STATE::PLAYERDIE;

}

else

{

CAudio::Instance()->Play(AUDIO\_TRANSFORM);

gameRunState = STATE::GOTONEXT;

}

}

character->OnKeyDown(nChar);

// 碰觸傳送門進下一關

CGameTransferGate\* TransferGate = CGameTransferGate::Instance();

if (character->IsDoingSomeThing() && character->Collision(TransferGate))

{

TransferGate->SetDie(false);

// 關卡數加一

gameMap->AddGameLevel(1);

if (gameMap->GetGameLevel() == 15) // 打完 3-5 通關

{

GotoGameState(GAME\_STATE\_OVER);

}

else

{

CAudio::Instance()->Play(AUDIO\_TRANSFORM);

gameRunState = STATE::GOTONEXT;

}

}

break;

}

case game\_framework::CGameStateRun::STATE::PAUSE:

break;

case game\_framework::CGameStateRun::STATE::GOTONEXT:

{

if (nChar == 78) // 按 N 跳過動畫

{

GotoGameState(GAME\_STATE\_RUN);

}

break;

}

default:

break;

}

}

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

{

switch (gameRunState)

{

case game\_framework::CGameStateRun::STATE::RUN:

character->OnKeyUp(nChar);

break;

case game\_framework::CGameStateRun::STATE::PAUSE:

break;

case game\_framework::CGameStateRun::STATE::GOTONEXT:

break;

case game\_framework::CGameStateRun::STATE::PLAYERDIE:

break;

default:

break;

}

}

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

{

switch (gameRunState)

{

case game\_framework::CGameStateRun::STATE::RUN:

{

if (btn\_pause.PointIn(point.x, point.y)) {

// 暫停按鈕動畫變換

if (!btn\_pause.IsFinalBitmap()) {

btn\_pause.OnMove();

}

}

else if (CUISkill::Instance().PointIn(point.x, point.y))

{

character->UseSkill();

}

else if (CUIWeaponSwitch::Instance().PointIn(point.x, point.y))

{

character->SwitchWeapon();

}

break;

}

case game\_framework::CGameStateRun::STATE::PAUSE:

{

if (btn\_goBack.PointIn(point.x, point.y))

{

if (!btn\_goBack.IsFinalBitmap()) {

btn\_goBack.OnMove();

}

}

else if (btn\_continue.PointIn(point.x, point.y))

{

if (!btn\_continue.IsFinalBitmap()) {

btn\_continue.OnMove();

}

}

break;

}

case game\_framework::CGameStateRun::STATE::GOTONEXT:

break;

case game\_framework::CGameStateRun::STATE::PLAYERDIE:

break;

default:

break;

}

}

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

{

switch (gameRunState)

{

case game\_framework::CGameStateRun::STATE::RUN:

{

if (btn\_pause.PointIn(point.x, point.y)) {

gameRunState = STATE::PAUSE;

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

break;

}

case game\_framework::CGameStateRun::STATE::PAUSE:

{

if (btn\_goBack.PointIn(point.x, point.y))

{

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

GameEnd();

GotoGameState(GAME\_STATE\_INIT);

}

else if (btn\_continue.PointIn(point.x, point.y))

{

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

gameRunState = STATE::RUN;

UI\_posy = 0;

btn\_setup.Reset();

}

else if (btn\_setup.PointIn(point.x, point.y))

{

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

btn\_setup.OnMove();

}

else if (btn\_BGM.PointIn(point.x, point.y))

{

btn\_BGM.OnMove();

if (!btn\_soundEffect.IsFinalBitmap())

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

if (btn\_BGM.IsFinalBitmap())

{

// 暫停所有 BGM

for (unsigned int id = AUDIO\_ID::AUDIO\_BGM\_INIT; id < AUDIO\_ID::AUDIO\_BGM\_COUNT; id++)

{

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

}

}

else

{

// 恢復所有 BGM

for (unsigned int id = AUDIO\_ID::AUDIO\_BGM\_INIT; id < AUDIO\_ID::AUDIO\_BGM\_COUNT; id++)

{

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

}

}

}

else if (btn\_soundEffect.PointIn(point.x, point.y))

{

btn\_soundEffect.OnMove();

if (btn\_soundEffect.IsFinalBitmap())

{

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

}

else

{

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

CAudio::Instance()->Play(AUDIO\_BTN\_DOWN);

}

if (btn\_BGM.IsFinalBitmap())

{

// 暫停所有 BGM

for (unsigned int id = AUDIO\_ID::AUDIO\_BGM\_INIT; id < AUDIO\_ID::AUDIO\_BGM\_COUNT; id++)

{

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

}

}

else

{

// 恢復所有 BGM

for (unsigned int id = AUDIO\_ID::AUDIO\_BGM\_INIT; id < AUDIO\_ID::AUDIO\_BGM\_COUNT; id++)

{

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

}

}

}

break;

}

case game\_framework::CGameStateRun::STATE::GOTONEXT:

break;

case game\_framework::CGameStateRun::STATE::PLAYERDIE:

break;

default:

break;

}

btn\_pause.Reset();

btn\_goBack.Reset();

btn\_continue.Reset();

}

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

{

// 沒事。如果需要處理滑鼠移動的話，寫code在這裡

}

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

{

}

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

{

}

void CGameStateRun::OnShow()

{

//

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

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

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

//

switch (gameRunState)

{

case game\_framework::CGameStateRun::STATE::PAUSE:

case game\_framework::CGameStateRun::STATE::PLAYERDIE:

case game\_framework::CGameStateRun::STATE::RUN:

{

// GAME

gameMap->OnShow(false);

for (CGameObj\* obj : CGameObj::\_allObj)

{

if (gameMap->InScreen(obj->GetX1(), obj->GetY1(), obj->GetX2(), obj->GetY2()))

obj->OnShow(gameMap);

}

gameMap->OnShow(true);

// UI

HPBACKGROUND.ShowBitmap();

MAXHP.SetTopLeft(82, 7 + UI\_posy);

MAXSP.SetTopLeft(82, 28 + UI\_posy);

MAXMP.SetTopLeft(82, 48 + UI\_posy);

HPBACKGROUND.SetTopLeft(0, UI\_posy);

// 血量、魔量、護頓 動畫

int percent = character->GetHP() \* 100 / character->GetMAXHP();

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

{

HPBAR.SetTopLeft(27 + i, 10 + UI\_posy);

HPBAR.ShowBitmap();

}

percent = character->GetShield() \* 100 / character->GetMAXShield();

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

{

SPBAR.SetTopLeft(27 + i, 31 + UI\_posy);

SPBAR.ShowBitmap();

}

percent = character->GetMP() \* 100 / character->GetMAXMP();

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

{

MPBAR.SetTopLeft(26 + i, 51 + UI\_posy);

MPBAR.ShowBitmap();

}

// 血量、魔量、護頓數字

HP.SetInteger(character->GetHP());

SP.SetInteger(character->GetShield());

MP.SetInteger(character->GetMP());

MAXHP.SetInteger(character->GetMAXHP());

MAXSP.SetInteger(character->GetMAXShield());

MAXMP.SetInteger(character->GetMAXMP());

HP.SetTopLeft(70 - HP.GetLen() \* HP.GetWidth(), 7 + UI\_posy);

SP.SetTopLeft(70 - SP.GetLen() \* SP.GetWidth(), 28 + UI\_posy);

MP.SetTopLeft(70 - MP.GetLen() \* MP.GetWidth(), 48 + UI\_posy);

HP.ShowBitmap(false);

MP.ShowBitmap(false);

SP.ShowBitmap(false);

MAXHP.ShowBitmap(false);

MAXMP.ShowBitmap(false);

MAXSP.ShowBitmap(false);

// 斜線

SLASH.SetTopLeft(68, 8 + UI\_posy);

SLASH.ShowBitmap();

SLASH.SetTopLeft(68, 28 + UI\_posy);

SLASH.ShowBitmap();

SLASH.SetTopLeft(68, 48 + UI\_posy);

SLASH.ShowBitmap();

// 金幣

GOLDINTGER.SetInteger(character->GetGold());

GOLDINTGER.SetTopLeft((SIZE\_X - btn\_pause.Width() - 50) - GOLDINTGER.GetLen() \* GOLDINTGER.GetWidth(), 8 + UI\_posy);

GOLDINTGER.ShowBitmap(false);

GOLD.SetTopLeft((SIZE\_X - btn\_pause.Width() - 70) - (GOLDINTGER.GetLen() \* GOLDINTGER.GetWidth() + GOLD.Width()), 10 + UI\_posy);

GOLD.ShowBitmap();

// 關卡數

GAMELEVEL.SetInteger(1 + gameMap->GetGameLevel() / 5);

GAMELEVEL.SetTopLeft(SIZE\_X - 110 + dMinMap, 200 + btn\_pause.Height());

GAMELEVEL.ShowBitmap(false);

GAMELEVEL.SetInteger(1 + gameMap->GetGameLevel() % 5);

GAMELEVEL.SetTopLeft(SIZE\_X - 90 + dMinMap, 200 + btn\_pause.Height());

GAMELEVEL.ShowBitmap(false);

MINUS.SetTopLeft(SIZE\_X - 100 + dMinMap, 200 + btn\_pause.Height());

MINUS.ShowBitmap();

// UI小地圖

minMap.SetXY(SIZE\_X - 180 + dMinMap, 25 + btn\_pause.Height());

minMap.OnShow();

// UI BossHpBar

CUIBossHpBar::Instance().ShowHpBar();

// 技能 BOUTTON UI

CUISkill::Instance().ShowUI();

// 更換武器 BOUTTON UI

CUIWeaponSwitch::Instance().SetWeapon(character->GetNowWeapon());

CUIWeaponSwitch::Instance().ShowUI();

// 武器 UI 調動

CUIWeapon::Instance().ShowUI();

// 暫停介面

pause\_UI.SetTopLeft(((SIZE\_X - pause\_UI.Width()) >> 1), btn\_posy);

pause\_UI.ShowBitmap();

setup\_UI.SetTopLeft((SIZE\_X >> 1) - (setup\_UI.Width() >> 1), setup\_posy);

setup\_UI.ShowBitmap();

// 按鈕

btn\_pause.SetTopLeft(SIZE\_X - btn\_pause.Width() - 10, 10 + UI\_posy);

btn\_pause.OnShow();

btn\_goBack.SetTopLeft(12 + pause\_UI.Left(), btn\_posy + 181);

btn\_goBack.OnShow();

btn\_continue.SetTopLeft(8 + btn\_goBack.Left() + btn\_goBack.Width(), btn\_posy + 181);

btn\_continue.OnShow();

btn\_setup.SetTopLeft(8 + btn\_continue.Left() + btn\_continue.Width(), btn\_posy + 181);

btn\_setup.OnShow();

btn\_BGM.SetTopLeft(11 + setup\_UI.Left(), 11 + setup\_posy);

btn\_BGM.OnShow();

btn\_soundEffect.SetTopLeft(5 + btn\_BGM.Left() + btn\_BGM.Width(), 11 + setup\_posy);

btn\_soundEffect.OnShow();

break;

}

case game\_framework::CGameStateRun::STATE::GOTONEXT:

{

cutscenes.OnShow();

break;

}

default:

break;

}

//debug

/\*debugx.SetTopLeft(0, SIZE\_Y - 20);

debugy.SetTopLeft(80, SIZE\_Y - 20);

debugx.SetInteger(character.GetX1());

debugy.SetInteger(character.GetY1());

debugx.ShowBitmap(false);

debugy.ShowBitmap(false);\*/

/\*debugx.SetTopLeft(160, 460);

debugy.SetTopLeft(240, 460);

debugx.SetInteger(TransferGate.GetX1());

debugy.SetInteger(TransferGate.GetY1());

debugx.ShowBitmap(false);

debugy.ShowBitmap(false);\*/

/\*debugx.SetTopLeft(320, 460);

debugy.SetTopLeft(400, 460);

debugx.SetInteger(TransferGate.GetX2());

debugy.SetInteger(TransferGate.GetY2());

debugx.ShowBitmap(false);

debugy.ShowBitmap(false);\*/

}

void CGameStateRun::GameEnd()

{

CAudio::Instance()->Stop(AUDIO\_BGM\_SNOW);

}

}

**Audio.h有修改部分**

#pragma once

#include <list>

#include <vector>

#include <map>

using namespace std;

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

// 這個class會建立DirectX Audio物件，以提供音效

// 每個Public Interface的用法都要懂，Implementation可以不懂

// 注意事項：音效檔案可以用WAVE檔(.wav)或MIDI檔(.mid)，數個音效檔案可以

// 混合(同時)撥放，但是其中只能混合一個MIDI檔，兩個MIDI檔案不

// 能同時撥放，如果同時撥放兩個MIDI檔的話，前一個MIDI檔會被停

// 掉。

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

namespace game\_framework {

class CAudio {

public:

~CAudio();

void Close(); // 關閉Direct Sound介面

static CAudio\* Instance(); // 取得CAudio的Instance

bool Load(unsigned, char \*); // 載入編號i的聲音，指定MIDI檔案

bool Open(); // 開啟Direct Sound介面

void Pause(); // 暫停播放所有音效

void Pause(unsigned id); // 暫停某個 id 音效(自己加的不知會不會出錯)

void Play(unsigned, bool=false);// 開始撥放編號i的聲音，指定是否重覆撥放

void Resume(); // 復原暫停播放的音效

void Resume(unsigned id); // 復原暫停播放的某個 id 音效(自己加的不知會不會出錯)

void SetPowerResume(); // 電源恢復

void Stop(unsigned); // 停止撥放編號i的聲音

private:

class Info {

public:

Info() {

repeat = isGood = false;

}

bool repeat, isGood;

string fileName;

};

static void MCIThread(HANDLE); // MCI thread method

static void ExecuteMciCommand(char \*);

void SendMciCommand(char \*);

CAudio(); // private constructor

static CAudio audio; // 自動destruct

map<int, Info> info;

bool isOpened;

HANDLE hThread; // MCI command thread

HANDLE hWriteEnd; // Pipe write handle for thread

const static int MAX\_MCI\_COMMAND\_SIZE = 400;

};

enum AUDIO\_ID { // 定義各種音效的編號

AUDIO\_NULL, // 無定義

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

// BGM 包含範圍

AUDIO\_BGM\_INIT, // 初始畫面 BGM

AUDIO\_BGM\_SNOW, // 雪地 BGM

AUDIO\_BGM\_COUNT, // 用於關閉 BGM 的計數

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

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

// 武器音效

AUDIO\_GUN\_0, // 初始武器射擊

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

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

// 受傷音效

AUDIO\_HIT\_ENEMY\_0, // 敵人受傷

AUDIO\_HIT\_ENEMY\_1,

AUDIO\_HIT\_ENEMY\_2,

AUDIO\_HIT\_ENEMY\_3,

AUDIO\_HIT\_ENEMY\_4,

AUDIO\_HIT\_ENEMY\_5,

AUDIO\_HIT\_ENEMY\_COUNT, // 計數

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

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

// 死亡音效

AUDIO\_DEAD\_ENEMY\_0, // 敵人死亡

AUDIO\_DEAD\_ENEMY\_1,

AUDIO\_DEAD\_ENEMY\_2,

AUDIO\_DEAD\_ENEMY\_3,

AUDIO\_DEAD\_ENEMY\_4,

AUDIO\_DEAD\_ENEMY\_5,

AUDIO\_DEAD\_ENEMY\_6,

AUDIO\_DEAD\_ENEMY\_7,

AUDIO\_DEAD\_ENEMY\_8,

AUDIO\_DEAD\_ENEMY\_9,

AUDIO\_DEAD\_ENEMY\_10,

AUDIO\_DEAD\_ENEMY\_COUNT, // 計數

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

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

// 雜項

AUDIO\_BTN\_DOWN, // 按鈕點擊

AUDIO\_BOX\_OPEN, // 打開箱子

AUDIO\_GET\_COIN, // 得到金幣

AUDIO\_GET\_ENERGY, // 得到能量球

AUDIO\_HEALTH, // 恢復

AUDIO\_SWITCH\_WEAPON, // 更換武器

AUDIO\_SHIELDBREAK, // 護頓破裂

AUDIO\_TRANSFORM, // 傳送

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

AUDIO\_END // 只是結尾

};

}

**Audio.cpp有修改部分**

void CAudio::Pause()

{

if (!isOpened)

return;

map<int, Info>::iterator i;

for (i = info.begin(); i != info.end(); i++) {

if (i->second.isGood) {

char command[MAX\_MCI\_COMMAND\_SIZE];

sprintf(command, "pause device%d wait", i->first);

SendMciCommand(command);

i->second.isGood = false; // Pause 後要 Resume 才能 Play

}

}

}

void CAudio::Pause(unsigned id)

{

if (!isOpened)

return;

if (info[id].isGood) {

char command[MAX\_MCI\_COMMAND\_SIZE];

sprintf(command, "pause device%d wait", id);

SendMciCommand(command);

info[id].isGood = false; // Pause 後要 Resume 才能 Play

}

}

void CAudio::Play(unsigned id, bool repeat\_flag)

{

if (!isOpened)

return;

GAME\_ASSERT(info.find(id) != info.end(), "Can not play back audio: incorrect Audio ID!");

//

// Do not play, if the file is no good

//

//Pause();

if (!info[id].isGood)

return;

char command[MAX\_MCI\_COMMAND\_SIZE];

if (repeat\_flag)

sprintf(command, "play device%d from 0 repeat", id);

else

sprintf(command, "play device%d from 0", id);

SendMciCommand(command);

}

void CAudio::Resume()

{

if (!isOpened)

return;

map<int, Info>::iterator i;

for (i = info.begin(); i != info.end(); i++) {

char command[MAX\_MCI\_COMMAND\_SIZE];

sprintf(command, "resume device%d", i->first);

SendMciCommand(command);

i->second.isGood = true; // Pause 後要 Resume 才能 Play

}

}

void CAudio::Resume(unsigned id)

{

if (!isOpened)

return;

char command[MAX\_MCI\_COMMAND\_SIZE];

sprintf(command, "resume device%d", id);

SendMciCommand(command);

info[id].isGood = true; // Pause 後要 Resume 才能 Play

}

**gamelib.h有修改部分**

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

// 這個class提供可以移動的動畫

// 每個Public Interface的用法都要懂，Implementation可以不懂

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

class CAnimation {

public:

CAnimation(int=10); // Constructor (預設動畫播放頻率每1/3秒換一張圖)

CAnimation(const CAnimation&); // Copy Construct

CAnimation& operator=(const CAnimation& other); // Copy Assigment

void AddBitmap(int,COLORREF=CLR\_INVALID);

// 增加一張圖形至動畫(圖的編號及透明色)

void AddBitmap(char \*,COLORREF=CLR\_INVALID);

// 增加一張圖形至動畫(圖的編號及透明色)

int GetCurrentBitmapNumber(); // 取得正在撥放的bitmap是第幾個bitmap

int Height(); // 取得動畫的高度

bool IsFinalBitmap(); // 回傳正在撥放的bitmap是否為最後一個bitmap

int Left(); // 取得動畫的左上角的 x 座標

void OnMove(); // 依頻率更換bitmap

void OnShow(); // 將動畫貼到螢幕

void Reset(); // 重設播放順序回到第一張圖形

void SetDelayCount(int); // 設定動畫播放速度的常數(越大越慢)

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

int Top(); // 取得動畫的左上角的 y 座標

int Width(); // 取得動畫的寬度

int BmpSize(); // 取得圖片數量

private:

list<CMovingBitmap> bmp; // list of CMovingBitmap

list<CMovingBitmap>::iterator bmp\_iter; // list iterator

int bmp\_counter; // 儲存bmp\_iter為第n個bmp

int delay\_counter; // 延緩動畫播放速度的計數器

int delay\_count; // 動畫播放速度的常數

int x, y; // 動畫的座標

void copy(const CAnimation&);

};

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

// 這個class提供顯示整數圖形的能力

// 每個Public Interface的用法都要懂，Implementation可以不懂

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

class CInteger {

public:

enum class Color {WHITE, RED, YELLOW, BLUE, COLORCOUNT};

CInteger(); // default 5 digits

void Add(int n); // 增加整數值

int GetInteger(); // 回傳整數值

int GetFinal(); // 回傳最後圖片在螢幕的位置

int GetWidth(); // 取得圖片寬度

int GetHeight(); // 取得圖片高度

int GetLeft(); // 取得 x 座標

int GetTop(); // 取得 y 座標

int GetLen(); // 取得數字位數

void SetInteger(int); // 設定整數值

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

void ShowBitmap(); // 將動畫貼到螢幕

void ShowBitmap(bool leadingZero); // 有無前導0

void SetColor(Color); // 設定數字顏色

static void CInteger::LoadBitmap() // 載入0..9及負號之圖形

{

//

// digit[i]為class varibale，所以必須避免重複LoadBitmap

//

if (!isBmpLoaded) {

int d[(int)Color::COLORCOUNT][11] =

{

{ IDB\_0,IDB\_1,IDB\_2,IDB\_3,IDB\_4,IDB\_5,IDB\_6,IDB\_7,IDB\_8,IDB\_9,IDB\_MINUS },

{ IDB\_red\_0,IDB\_red\_1,IDB\_red\_2,IDB\_red\_3,IDB\_red\_4,IDB\_red\_5,

IDB\_red\_6,IDB\_red\_7,IDB\_red\_8,IDB\_red\_9,IDB\_MINUS\_red },

{ IDB\_yello\_0,IDB\_yello\_1,IDB\_yello\_2,IDB\_yello\_3,IDB\_yello\_4,IDB\_yello\_5,

IDB\_yello\_6,IDB\_yello\_7,IDB\_yello\_8,IDB\_yello\_9,IDB\_MINUS\_yellow },

{ IDB\_blue\_0,IDB\_blue\_1,IDB\_blue\_2,IDB\_blue\_3,IDB\_blue\_4,IDB\_blue\_5,IDB\_blue\_6,

IDB\_blue\_7,IDB\_blue\_8,IDB\_blue\_9,IDB\_MINUS\_blue }

};

for (int c = 0; c < (int)Color::COLORCOUNT; c++)

{

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

digit[c][i].LoadBitmap(d[c][i], RGB(0, 0, 0));

}

isBmpLoaded = true;

}

}

private:

int NUMDIGITS; // 共顯示NUMDIGITS個位數

static CMovingBitmap digit[(int)Color::COLORCOUNT][11]; // 儲存0..9及負號之圖形(bitmap)

int x, y; // 顯示的座標

int n; // 整數值

static bool isBmpLoaded; // 是否已經載入圖形

Color color; // 數字顏色

int Intlength(int);

};

**gamelib.cpp有修改部分**

CAnimation::CAnimation(int count)

{

delay\_count = count;

delay\_counter = delay\_count;

x = y = bmp\_counter = 0;

}

CAnimation::CAnimation(const CAnimation& other)

{

this->bmp = other.bmp;

this->bmp\_iter = bmp.begin();

this->bmp\_counter = other.bmp\_counter;

this->delay\_counter = other.delay\_counter;

this->delay\_count = other.delay\_count;

this->x = other.x;

this->y = other.y;

}

CAnimation& CAnimation::operator=(const CAnimation& other)

{

if(this != &other)

copy(other);

return \*this;

}

void CAnimation::copy(const CAnimation& other)

{

this->bmp = other.bmp;

this->bmp\_iter = bmp.begin();

this->bmp\_counter = other.bmp\_counter;

this->delay\_counter = other.delay\_counter;

this->delay\_count = other.delay\_count;

this->x = other.x;

this->y = other.y;

}

}

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

// CInteger: 這個class提供顯示整數圖形的能力

// 1. 要懂得怎麼呼叫(運用)其各種能力，但是可以不懂下列的程式是什麼意思

// 2. 自己寫到運用CMovingBitmap的程式時，可以參考下列程式的寫法

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

CMovingBitmap CInteger::digit[(int)Color::COLORCOUNT][11];

bool CInteger::isBmpLoaded = false;

CInteger::CInteger()

{

NUMDIGITS = 5;

color = Color::WHITE;

}

void CInteger::SetColor(Color c)

{

color = c;

}

void CInteger::Add(int x)

{

n += x;

}

int CInteger::GetInteger()

{

return n;

}

int CInteger::GetFinal()

{

return x + digit[(int)color][0].Width() \* NUMDIGITS;

}

int CInteger::GetLeft()

{

return x;

}

int CInteger::GetTop()

{

return y;

}

int CInteger::GetWidth()

{

return digit[(int)color][0].Width();

}

int CInteger::GetHeight()

{

return digit[(int)color][0].Height();

}

int CInteger::GetLen()

{

int len = Intlength(n);

if (len == 0)

{

len = 1;

}

return len;

}

void CInteger::SetInteger(int i)

{

n = i;

}

void CInteger::SetTopLeft(int nx, int ny) // 將動畫的左上角座標移至 (x,y)

{

x = nx; y = ny;

}

void CInteger::ShowBitmap(bool leadingZero) {

if (leadingZero == false) {

NUMDIGITS = GetLen();

}

ShowBitmap();

}

void CInteger::ShowBitmap()

{

GAME\_ASSERT(isBmpLoaded, "CInteger: 請先執行LoadBitmap，然後才能ShowBitmap");

int nx; // 待顯示位數的 x 座標

int MSB; // 最左邊(含符號)的位數的數值

if (n >= 0) {

MSB = n;

nx = x+digit[(int)color][0].Width()\*(NUMDIGITS-1);

} else {

MSB = -n;

nx = x+digit[(int)color][0].Width()\*NUMDIGITS;

}

for (int i=0; i < NUMDIGITS; i++) {

int d = MSB % 10;

MSB /= 10;

digit[(int)color][d].SetTopLeft(nx, y);

digit[(int)color][d].ShowBitmap();

nx -= digit[(int)color][d].Width();

}

if (n < 0) { // 如果小於0，則顯示負號

digit[(int)color][10].SetTopLeft(nx, y);

digit[(int)color][10].ShowBitmap();

}

}

int CInteger::Intlength(int n)

{

int ans = 0;

if (n < 0)

n = ~n + 1;

while (n > 0)

{

n /= 10;

ans++;

}

return ans;

}

全文完