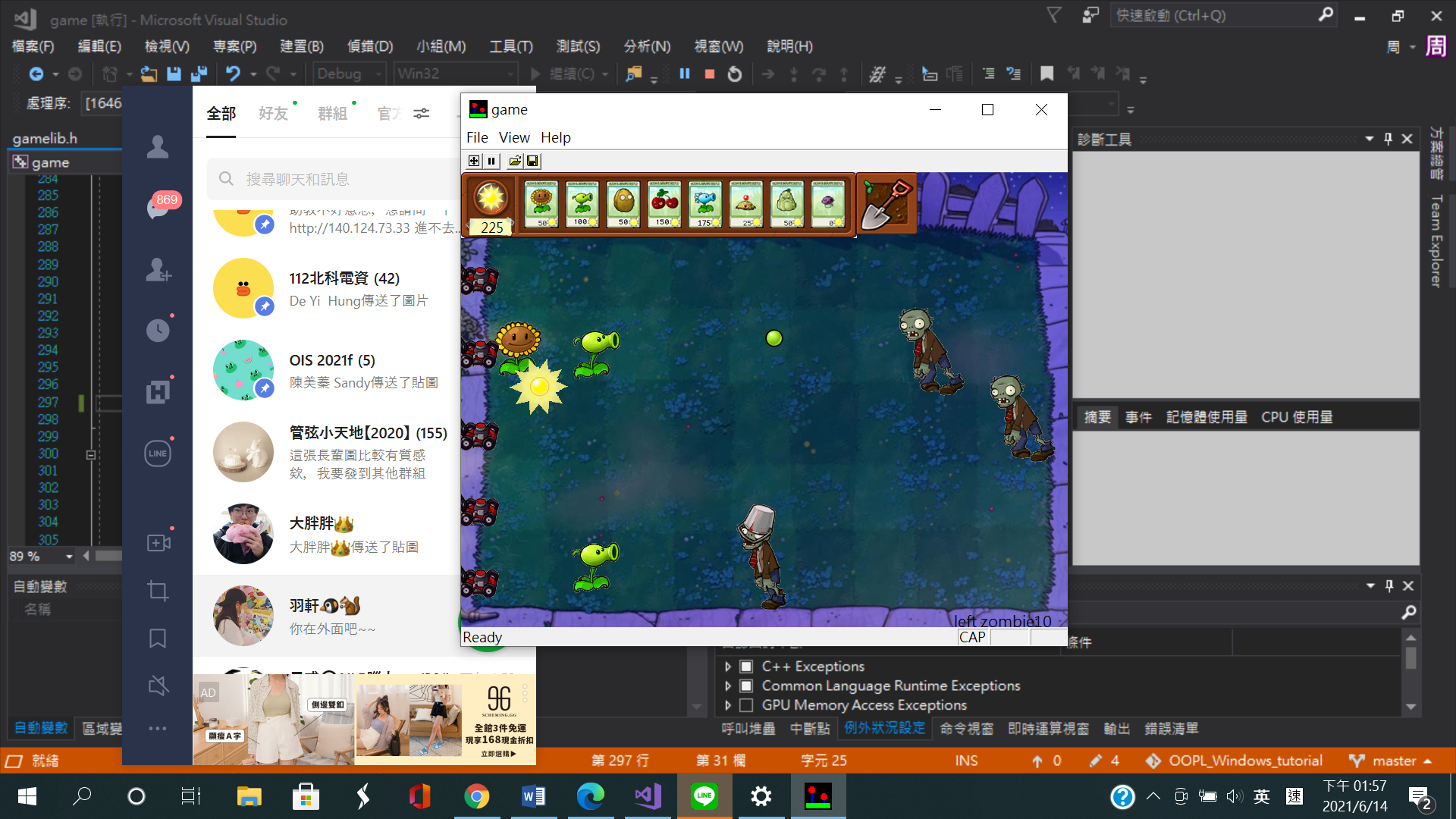
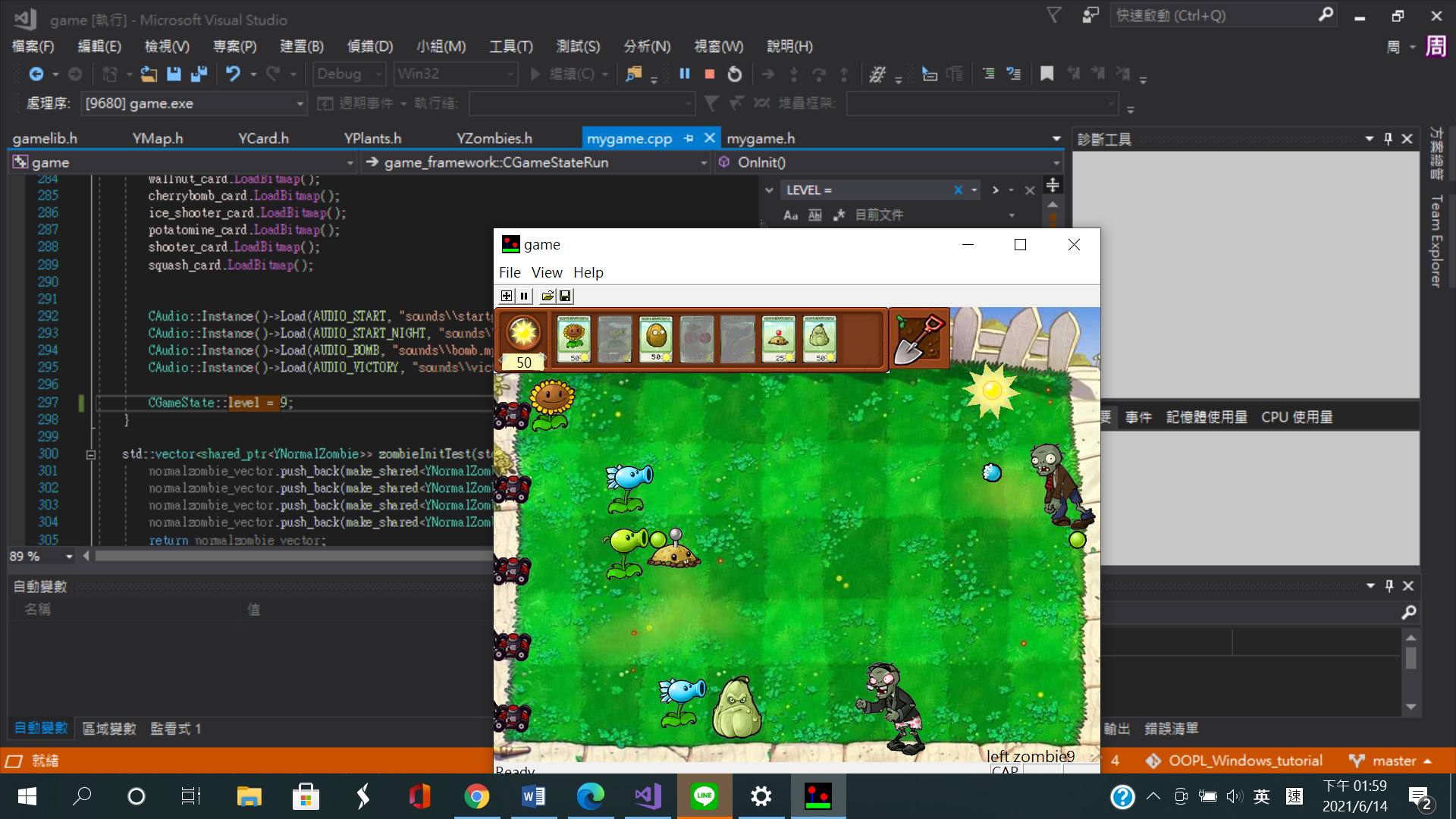
國立臺北科技大學

2020 Spring 資工系物件導向程式實習

期末報告

Plants vs. Zombies



第2組

108820001 羅羽軒

108820015 周雨柔

**目錄**

**一、** **簡介** 1

1. 動機 1

2. 分工 2

**二、** **遊戲介紹** 3

1. 遊戲說明 3

2. 遊戲圖形 4

3. 遊戲音效 8

**三、** **程式設計** 9

1. 程式架構 9

2. 程式類別 11

3. 程式技術 12

**四、** **結語** 13

1. 問題及解決方法 13

2. 時間表 14

3. 貢獻比例 14

4. 自我檢核表 15

5. 收穫 16

6. 心得、感想 16

7. 對於本課程的建議 16

**附錄** 17

1. **簡介**
2. 動機：

植物大戰殭屍雖然是個簡單的遊戲，不過在許多人心中是個經典有趣的回憶，網路上還有一群熱愛者，喜愛嘗試植物大戰殭屍的特殊玩法，甚至統計了各種植物的屬性做成了植物小百科。在玩遊戲時常常會覺得殭屍走太慢，或者太陽不夠用，所以我們藉著這門課，實作這個有趣又可愛的遊戲，並加上一些秘技，增加不同的遊戲體驗。

1. 分工： 由羅羽軒主導，周雨柔協助。
2. **遊戲介紹**
3. 遊戲說明：

本遊戲主要是用滑鼠點擊操控，部分功能與密技會用到鍵盤。遊戲共有10關，需要收集太陽，用太陽去種植各種植物，阻止殭屍進入家園。在殺死該關卡的全部殭屍後，即可進入下一關。以下為遊戲中使用的按鍵與對應事件。

|  |  |  |  |
| --- | --- | --- | --- |
| 密技 | | 功能 | |
| 按鍵 | 對應事件 | 按鍵 | 對應事件 |
| S鍵 | 太陽數值加到500 | esc鍵 | 結束遊戲 |
| Z鍵 | 殭屍加速 | Ctrl鍵+Q鍵 | 暫停遊戲 |
| D鍵 | 全部殭屍死亡 | Ctrl鍵+F鍵 | 全螢幕切換 |
| L鍵 | 跳至下一關 |  |  |

1. 遊戲圖形：

|  |  |  |
| --- | --- | --- |
| bullet | ice bullet | mushroom bullet |
|  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| name | plant | card | name | plant | card |
| shooter |  |  | ice shooter |  |  |
| Sun flower |  |  | mushroom shooter |  |  |
| churry bomb |  |  | squash |  |  |
| nut wall |  |  | potato mine |  |  |

|  |  |
| --- | --- |
| 太陽量顯示與卡片放置版 | |
|  | |
| shovel | trophy |
|  |  |
| Loading畫面 | 選單畫面 |
|  |  |
| 白天場景 | 黑夜場景 |
|  |  |
| 白天失敗畫面 | 黑夜失敗畫面 |
|  |  |

|  |  |  |
| --- | --- | --- |
| name | zombie | attack |
| normal zombie |  |  |
| conehead zombie |  |  |
| bucket zombie |  |  |
| flag zombie |  |  |
| newspaper zombie |  |  |
| no newspaper zombie |  |  |

|  |  |
| --- | --- |
| die (head part) | |
|  |  |
| normal die (body part) | |
|  |  |
| newspaper zombie die (body part) | |
|  |  |
| bomb die | |
|  |  |

1. 遊戲音效：

|  |  |
| --- | --- |
| 事件 | 對應音效 |
| 選單BGM | mainmenu.mp3 |
| 選單進入遊戲 | menutogame.mp3 |
| 白天BGM | startgame.mp3 |
| 夜晚BGM | night\_bgm.mp3 |
| 種植物的聲音 | plants\_sound.mp3 |
| 收集太陽的聲音 | sun\_pick.mp3 |
| 除草機的聲音 | car.mp3 |
| 爆炸的聲音 | bomb.mp3 |
| 完成所有關卡獲得勝利的聲音 | victory.mp3 |

**程式設計**

1. 程式架構：

|  |
| --- |
| CGameState |
|  |
| CGameMap |
|  |
| CDamageObject |
|  |

|  |
| --- |
| Others |
|  |

1. 程式類別：

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| Definitions | 16 |  | 遊戲元件的定義 |
| CBlood | 42 | 77 | 操控美心與爸爸的血條 |
| CBossAI | 58 | 464 | 操控爸爸的AI |
| CDamageContainer | 37 | 125 | 存取所有爸爸攻擊招式及設定攻擊的傷害量 |
| CDamageFish | 30 | 59 | 操控爸爸攻擊招式中的紅龍 |
| CDamageLighter | 30 | 69 | 操控爸爸攻擊招式中的打火機 |
| CDamageObject | 51 | 28 | 爸爸攻擊招式的框架 |
| CDamagePen | 30 | 50 | 操控爸爸攻擊招式中的鋼筆 |
| CDamagePill | 34 | 57 | 操控爸爸攻擊招式中的藥丸 |
| CDamageWinnie | 29 | 42 | 操控爸爸攻擊招式中的符咒 |
| CDaughter | 114 | 305 | 操控美心移動動作等 |
| CDialog | 53 | 124 | 劇情對話框 |
| CFather | 55 | 162 | 操控爸爸移動動作等 |
| CGameData | 43 | 129 | 存取地圖、美心、爸爸位置等資料 |

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| CGameMap | 77 | 43 | 地圖的框架 |
| CGameMap01 | 17 | 37 | 地圖一 |
| CGameMap02 | 17 | 36 | 地圖二 |
| CGameMap03 | 17 | 36 | 地圖三 |
| CGameMap04 | 17 | 36 | 地圖四 |
| CGameMap05 | 17 | 36 | 地圖五 |
| CGameState01 | 38 | 80 | 遊戲章節一 |
| CGameState02 | 39 | 81 | 遊戲章節二 |
| CGameState03 | 39 | 82 | 遊戲章節三 |
| CGameState04 | 43 | 115 | 遊戲章節四 |
| CGameState05 | 48 | 135 | 遊戲章節五 |
| CTransition | 24 | 67 | 遊戲轉場切換 |
| mygame | 96 | 168 | State\_init,state\_over |
| **總行數** | **489** | **952** | **1441** |

1. 程式技術：

最多技術都集中在BossAI的部分上，因為要使他能夠完全自動且執行所有攻擊與移動，所以我用了一個矩陣表示各個狀態間互相轉移的機率，AI會記錄前一個狀態，而透過前一個狀態可以得到轉移至其他狀態的機率，透過隨機的方式決定他下一步要做什麼樣的動作，完成自主運作。

1. **結語**
2. 問題及解決方法：

|  |  |
| --- | --- |
| 問題 | 解決方法 |
| 轉場動畫不知道如何順暢運行 | 丟在OnMove()裡，y就會穩定增加 |
| 不知道Dialog如何換圖 | 參考Aniation |
| 動畫第一張圖會卡頓造成動畫不順 | 在第一張圖加上一張空白圖片吃掉延遲 |

1. 時間表：

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 陳美蓁(小時) | 陳柏瑞(小時) | 說明 |
| 1 | 2 | 2 | 練習 |
| 2 | 2 | 2 | 練習 |
| 3 | 3 | 2 | 練習、開會 |
| 4 | 4 | 4 | 地圖 |
| 5 | 6 | 6 | 主角移動 |
| 6 | 8 | 10 | 螢幕跟主角移動、換地圖 |
| 7 | 7 | 3 | 攻擊與轉場動畫 |
| 8 | 2 | 2 | 攻擊與轉場動畫完成 |
| 9 | 1 | 4 | 地圖4 |
| 10 | 4 | 4 | 攻擊物件、音效 |
| 11 | 5 | 3 | 攻擊物件、血條 |
| 12 | 4 | 4 | 攻擊判定 |
| 13 | 6 | 3 | 攻擊物件、音效與血條完成 |
| 14 | 6 | 4 | 攻擊物件，code clean |
| 15 | 5 | 10 | 剩餘攻擊物件，BOSS實作，劇情對話 |
| 16 | 10 | 20 | BOSS實作、劇情完成 |
| 17 | 20 | 40 | 補音效、劇情、做期末報告 |
| **合計** | **95** | **123** |  |

1. 貢獻比例：

陳美蓁：45%、陳柏瑞：55%

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 | 自訂遊戲icon | ☑已完成 □未完成 |  |
| 3 | 有About畫面 | ☑已完成 □未完成 |  |
| 4 | 初始畫面說明按鍵及滑鼠之用法與密技 | ☑已完成 □未完成 |  |
| 5 | 上傳setup/apk/source檔 | ☑已完成 □未完成 |  |
| 6 | setup檔可正確執行 | ☑已完成 □未完成 |  |
| 7 | 報告字型、點數、對齊、行距、頁碼等格式正確 | ☑已完成 □未完成 |  |
| 8 | 報告封面、側邊格式正確 | ☑已完成 □未完成 |  |
| 9 | 報告附錄程式格式正確 | ☑已完成 □未完成 |  |
| 10 | 全螢幕啟動 | ☑已完成 □未完成 |  |

1. 收穫：

陳美蓁：

可以設中斷點一步一步去找可以通過編譯的錯誤。

陳柏瑞：

如何用最基本的框架，透過自己實現所有物理效果與碰撞，來完成一個遊戲。

1. 心得、感想：

陳美蓁：

在這學期的課程中我學到了如何將一個遊戲從只有框架到一個遊戲完整的產出。其中過程除了程式規劃、程式撰寫外，最耗時間的莫過於蒐集材料，除了需要自己手工做出一張張圖片，還有要找到符合遊戲內的音效。還記得當時在找腳步聲的時候，聽了十幾種聲音才最後定案。很感謝老師開這一堂課，讓我們可以實際運用物件導向程式設計課中的概念，去完成一個自己的作品。在每週回報進度的時候，都會指導我們所遇到的問題該如何解決比較好。感謝同學跟我一起完成了這個遊戲！

陳柏瑞：

滿訝異最後能做出這個遊戲的，本來在設計的時候目標就是想做一個高難度的彈幕遊戲，因為這類遊戲粒子特效跟物件都特別多，也要有很多種不同的攻擊方式，本來是以為做不出來的，最後產出的這個遊戲雖然跟其他的彈幕遊戲還有很大的差距，但我已經十分滿意。

1. 對於本課程的建議：

希望下次期末報告的遊戲圖形可以擴充到5頁。

附錄

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

CBlood.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CBlood.h"

namespace game\_framework {

CBlood::CBlood() : daughterHP(102), fatherHP(450) {}

CBlood::~CBlood(){}

void CBlood::LoadBitmap() {

daughter\_hp.LoadBitmap(IDB\_DAUGHTER\_BLOOD, RGB(255, 255, 255));

daughter\_blood.LoadBitmap(IDB\_BLOOD, RGB(255, 255, 255));

father\_hp.LoadBitmap(IDB\_FATHER\_BLOOD, RGB(255, 255, 255));

father\_blood.LoadBitmap(IDB\_BLOOD, RGB(255, 255, 255));

}

void CBlood::setDaughterHP(int hp) {

daughterHP = hp;

}

void CBlood::setDaughterDamage(int damage,int invincible\_time) {

if (daughter\_invincible < 0) {

daughterHP -= damage;

daughter\_invincible = invincible\_time;

CAudio::Instance()->Play(DAUGHTER\_HURT, false);

}

}

bool CBlood::daughterIsDied() {

if (daughterHP <= -141) {

return true;

}

else {

return false;

}

}

void CBlood::setFatherHP(int hp) {

fatherHP = hp;

}

void CBlood::setFatherDamage(int damage,int invincible\_time) {

if (father\_invincible < 0) {

fatherHP += damage;

father\_invincible = invincible\_time;

CAudio::Instance()->Play(FATHER\_HURT, false);

}

}

bool CBlood::fatherIsDied() {

if (fatherHP >= 698) {

return true;

}

else {

return false;

}

}

void CBlood::OnShow(bool showFatherHP) {

daughter\_hp.SetTopLeft(0, 10);

daughter\_blood.SetTopLeft(daughterHP, 48);

daughter\_blood.ShowBitmap();

daughter\_hp.ShowBitmap();

if (showFatherHP) {

father\_hp.SetTopLeft(449, 10);

father\_blood.SetTopLeft(fatherHP, 48);

father\_blood.ShowBitmap();

father\_hp.ShowBitmap();

}

daughter\_invincible--;

father\_invincible--;

}

}

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

CBlood.h

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

#pragma once

#include "Definitions.h"

#ifndef CBLOOD\_H

#define CBLOOD\_H

namespace game\_framework {

class CBlood {

public:

CBlood();

~CBlood();

void LoadBitmap();

void setDaughterHP(int); //設定美心血量

void setDaughterDamage(int, int = 30); //設定美心被傷害量

int getDaughterHP() const; //回傳美心血量

bool daughterIsDied(); //判斷美心是否死亡

void setFatherHP(int); //設定爸爸血量

void setFatherDamage(int, int = 6); //設定爸爸被傷害量

int getFatherHP() const; //回傳爸爸血量

bool fatherIsDied(); //判斷爸爸是否死亡

void OnShow(bool);

private:

CMovingBitmap daughter\_hp;

CMovingBitmap daughter\_blood;

CMovingBitmap father\_hp;

CMovingBitmap father\_blood;

int daughterHP;

int fatherHP;

int daughter\_invincible;

int father\_invincible;

};

inline int CBlood::getDaughterHP() const{

return daughterHP;

}

inline int CBlood::getFatherHP() const{

return fatherHP;

}

}

#endif

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

CBossAI.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include <stdlib.h>

#include <time.h>

#include "audio.h"

#include "CBossAI.h"

#include "CFather.h"

#include "CGameData.h"

#include "CDamageContainer.h"

namespace game\_framework {

CBossAI::CBossAI() {

int decision\_probiality\_1\_init[AI\_TOTAL\_STATE\_COUNT][AI\_TOTAL\_STATE\_COUNT] = {

{ 0,5,30,0,10,30,0,0,0,30,0 },

{ 5,0,30,0,30,10,0,0,0,30,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0 }

};

int decision\_probiality\_2\_init[AI\_TOTAL\_STATE\_COUNT][AI\_TOTAL\_STATE\_COUNT] = {

{ 0,5,0,30,0,25,30,0,30,30,0 },

{ 5,0,0,30,25,0,30,30,0,30,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 }

};

counter = 0;

gap\_counter = 0;

current\_state = AI\_STATE\_MOVE\_LEFT;

AI\_Lock = false;

decision\_probiality\_1 = new int\*[AI\_TOTAL\_STATE\_COUNT];

for (int i = 0; i < AI\_TOTAL\_STATE\_COUNT; i++) {

decision\_probiality\_1[i] = new int[AI\_TOTAL\_STATE\_COUNT];

for (int j = 0; j < AI\_TOTAL\_STATE\_COUNT; j++) {

decision\_probiality\_1[i][j] = decision\_probiality\_1\_init[i][j];

}

}

}

CBossAI::CBossAI(CFather\* father, CGameData\* game\_data, CDamageContainer\* container,CBlood\* blood) {

int decision\_probiality\_1\_init[AI\_TOTAL\_STATE\_COUNT][AI\_TOTAL\_STATE\_COUNT] = {

{ 0,5,30,0,10,30,0,0,0,30,0 },

{ 5,0,30,0,30,10,0,0,0,30,0 },

{ 30,30,0,0,15,15,0,0,0,0,0 },

{ 30,30,0,0,15,15,0,0,0,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 10,10,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0 }

};

int decision\_probiality\_2\_init[AI\_TOTAL\_STATE\_COUNT][AI\_TOTAL\_STATE\_COUNT] = {

{ 0,5,0,30,0,25,30,0,30,30,30 },

{ 5,0,0,30,25,0,30,30,0,30,30},

{ 30,30,0,0,15,15,0,15,15,0,0 },

{ 30,30,0,0,15,15,0,15,15,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 30,30,0,0,15,15,0,15,15,0,0 },

{ 10,0,0,0,0,0,0,0,0,0,0 },

{ 0,10,0,0,0,0,0,0,0,0,0 },

{ 30,30,0,0,15,15,0,15,15,0,0 },

{ 0,30,0,0,10,0,0,10,0,0,0 }

};

this->father = father;

this->game\_data = game\_data;

this->container = container;

this->blood = blood;

counter = 0;

gap\_counter = 0;

current\_state = AI\_STATE\_MOVE\_LEFT;

AI\_Lock = false;

decision\_probiality\_1 = new int\*[AI\_TOTAL\_STATE\_COUNT];

for (int i = 0; i < AI\_TOTAL\_STATE\_COUNT; i++) {

decision\_probiality\_1[i] = new int[AI\_TOTAL\_STATE\_COUNT];

for (int j = 0; j < AI\_TOTAL\_STATE\_COUNT; j++) {

decision\_probiality\_1[i][j] = decision\_probiality\_1\_init[i][j];

}

}

decision\_probiality\_2 = new int\*[AI\_TOTAL\_STATE\_COUNT];

for (int i = 0; i < AI\_TOTAL\_STATE\_COUNT; i++) {

decision\_probiality\_2[i] = new int[AI\_TOTAL\_STATE\_COUNT];

for (int j = 0; j < AI\_TOTAL\_STATE\_COUNT; j++) {

decision\_probiality\_2[i][j] = decision\_probiality\_2\_init[i][j];

}

}

}

CBossAI::~CBossAI() {

for (int i = 0; i < AI\_TOTAL\_STATE\_COUNT; i++) {

delete[] decision\_probiality\_1[i];

delete[] decision\_probiality\_2[i];

}

delete[] decision\_probiality\_1;

delete[] decision\_probiality\_2;

}

void CBossAI::setAILock(bool l) {

this->AI\_Lock = l;

}

void CBossAI::OnLoop() {

if (AI\_Lock) {

current\_state = AI\_STATE\_IDLE;

previous\_state = AI\_STATE\_MOVE\_LEFT;

}

switch (current\_state) {

case AI\_STATE\_IDLE:

nextStepChoise();

break;

case AI\_STATE\_MOVE\_LEFT:

stateMoveLeft();

break;

case AI\_STATE\_MOVE\_RIGHT:

stateMoveRight();

break;

case AI\_STATE\_ATK\_WINNIE\_1:

stateWinnieATK1();

break;

case AI\_STATE\_ATK\_WINNIE\_2:

stateWinnieATK2();

break;

case AI\_STATE\_ATK\_FISH:

stateFishATK();

break;

case AI\_STATE\_ATK\_PEN\_R:

if (blood->getFatherHP() > 550) {

gap\_counter = 30;

}

else {

gap\_counter = 40;

}

statePenATKR();

break;

case AI\_STATE\_ATK\_PEN\_L:

if (blood->getFatherHP() > 550) {

gap\_counter = 30;

}

else {

gap\_counter = 40;

}

statePenATKL();

break;

case AI\_STATE\_ATK\_PEN\_R\_2:

statePenATKR2();

break;

case AI\_STATE\_ATK\_PEN\_L\_2:

statePenATKL2();

break;

case AI\_STATE\_ATK\_PILL:

if (blood->getFatherHP() > 550) {

gap\_counter = 5;

}

else {

gap\_counter = 10;

}

statePillATK();

break;

case AI\_STATE\_ATK\_LIGHTER:

stateLighterATK();

break;

}

}

void CBossAI::nextStepChoise() {

bool low\_health = blood->getFatherHP() > 550;

srand((unsigned)time(NULL));

int sum = 0;

int rand\_result = 0;

int next\_state = 0;

for (int i = 0; i < AI\_TOTAL\_STATE\_COUNT; i++) {

if (low\_health) {

sum += decision\_probiality\_2[previous\_state][i];

}

else {

sum += decision\_probiality\_1[previous\_state][i];

}

}

rand\_result = (rand() % sum)+1;

for (int i = 0; i < AI\_TOTAL\_STATE\_COUNT; i++) {

if (low\_health) {

rand\_result -= decision\_probiality\_2[previous\_state][i];

}

else {

rand\_result -= decision\_probiality\_1[previous\_state][i];

}

if (rand\_result <= 0) {

next\_state = i;

break;

}

}

if (game\_data->getFatherPos().first > 650) {

next\_state = AI\_STATE\_MOVE\_LEFT;

}

else if (game\_data->getFatherPos().first < 50) {

next\_state = AI\_STATE\_MOVE\_RIGHT;

}

previous\_state = current\_state;

current\_state = next\_state;

counter = 0;

}

bool CBossAI::moveToTargetPos(int x, int y) {

std::pair<int, int> pos = game\_data->getFatherPos();

std::pair<int, int> target\_vector = std::make\_pair<int, int>(x - pos.first, y - pos.second);

if ((target\_vector.first<5 && target\_vector.first>-5) && (target\_vector.second<5 && target\_vector.second>-5)) {

game\_data->setFatherTopLeft(x, y);

}

else {

game\_data->setFatherMove(target\_vector.first / 5, target\_vector.second / 5);

}

pos = game\_data->getFatherPos();

return pos.first == x && pos.second == y;

}

void CBossAI::stateMoveRight() {

father->OnKeyDown(KEY\_RIGHT);

counter++;

if (counter > 45) {

father->OnKeyUP(KEY\_RIGHT);

previous\_state = AI\_STATE\_MOVE\_RIGHT;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

void CBossAI::stateMoveLeft() {

father->OnKeyDown(KEY\_LEFT);

counter++;

if (counter > 45) {

father->OnKeyUP(KEY\_LEFT);

previous\_state = AI\_STATE\_MOVE\_LEFT;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

void CBossAI::stateWinnieATK1() {

father->setOnTakeCtrl(true);

father->setStatus(FATHER\_STATUS\_FACE\_1);

if (moveToTargetPos(350, 10)) {

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter % 20 == 0) {

container->createDamageObject(OBJET\_WINNIE\_1);

}

counter++;

if (counter >= 60) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_WINNIE\_1;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::stateWinnieATK2() {

father->setOnTakeCtrl(true);

father->setStatus(FATHER\_STATUS\_FACE\_1);

if (moveToTargetPos(350, 10)) {

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter % 35 == 0 && counter % 70 != 0) {

container->createDamageObject(OBJET\_WINNIE\_2);

}

else if (counter % 35 == 0 && counter % 70 == 0) {

container->createDamageObject(OBJET\_WINNIE\_3);

}

counter++;

if (counter >= 139) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_WINNIE\_1;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::stateFishATK() {

father->setOnTakeCtrl(true);

father->setStatus(FATHER\_STATUS\_FACE\_1);

if (moveToTargetPos(350, 50)) {

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter == 10) {

container->createDamageObject(OBJET\_FISH);

}

counter++;

if (counter >= 70) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_FISH;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::statePenATKR() {

std::pair<int, int> pos = game\_data->getFatherPos();

father->OnKeyDown(KEY\_RIGHT);

if (pos.first>=700) {

father->OnKeyUP(KEY\_RIGHT);

game\_data->setFatherTopLeft(700, 270);

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter % gap\_counter == 0 && counter % (gap\_counter \* 2) != 0) {

container->createDamageObject(OBJET\_PEN, 0);

container->createDamageObject(OBJET\_PEN, 200);

container->createDamageObject(OBJET\_PEN, 400);

container->createDamageObject(OBJET\_PEN, 600);

}

else if (counter % gap\_counter == 0 && counter % (gap\_counter\*2) == 0) {

container->createDamageObject(OBJET\_PEN, 100);

container->createDamageObject(OBJET\_PEN, 300);

container->createDamageObject(OBJET\_PEN, 500);

}

counter++;

if (counter >= (gap\_counter \* 4)) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_PEN\_R;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::statePenATKL() {

std::pair<int, int> pos = game\_data->getFatherPos();

father->OnKeyDown(KEY\_LEFT);

if (pos.first <= 0) {

father->OnKeyUP(KEY\_LEFT);

game\_data->setFatherTopLeft(0, 270);

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter % gap\_counter == 0 && counter % (gap\_counter \* 2) != 0) {

container->createDamageObject(OBJET\_PEN, 100);

container->createDamageObject(OBJET\_PEN, 300);

container->createDamageObject(OBJET\_PEN, 500);

container->createDamageObject(OBJET\_PEN, 700);

}

else if (counter % gap\_counter == 0 && counter % (gap\_counter \* 2) == 0) {

container->createDamageObject(OBJET\_PEN, 200);

container->createDamageObject(OBJET\_PEN, 400);

container->createDamageObject(OBJET\_PEN, 600);

}

counter++;

if (counter >= (gap\_counter \* 4)) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_PEN\_L;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::statePenATKR2() {

std::pair<int, int> pos = game\_data->getFatherPos();

father->OnKeyDown(KEY\_RIGHT);

if (pos.first >= 700) {

father->OnKeyUP(KEY\_RIGHT);

game\_data->setFatherTopLeft(700, 270);

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter % 5 == 0) {

container->createDamageObject(OBJET\_PEN, 600-((counter/5)\*50));

}

counter++;

if (counter >= 55) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_PEN\_R\_2;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::statePenATKL2() {

std::pair<int, int> pos = game\_data->getFatherPos();

father->OnKeyDown(KEY\_LEFT);

if (pos.first <= 0) {

father->OnKeyUP(KEY\_LEFT);

game\_data->setFatherTopLeft(0, 270);

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter % 5== 0) {

container->createDamageObject(OBJET\_PEN, 100 + ((counter / 5) \* 50));

}

counter++;

if (counter >= 55) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_PEN\_L\_2;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::statePillATK() {

father->setOnTakeCtrl(true);

father->setStatus(FATHER\_STATUS\_FACE\_1);

if (moveToTargetPos(350, 50)) {

father->setStatus(FATHER\_STATUS\_FACE\_2);

if (counter % gap\_counter == 0) {

container->createDamageObject(OBJET\_PILL, 350, 100, game\_data);

}

counter++;

if (counter >= (gap\_counter \* (17-gap\_counter))) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_PILL;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

void CBossAI::stateLighterATK() {

std::pair<int, int> pos = game\_data->getFatherPos();

father->setOnTakeCtrl(true);

father->setStatus(FATHER\_STATUS\_FACE\_1);

if (counter<100) {

if (moveToTargetPos(0, 0)) {

counter = 100;

}

}else if (counter >= 100 && counter<150) {

father->setStatus(FATHER\_STATUS\_STAND);

father->setFaceSide(FACE\_RIGHT);

game\_data->setFatherMove(15, 0);

if (counter % 10 == 0) {

container->createDamageObject(OBJET\_LIGHTER,pos.first);

}

}

else if (counter >= 150) {

father->setStatus(FATHER\_STATUS\_STAND);

father->setFaceSide(FACE\_LEFT);

game\_data->setFatherMove(-15, 0);

if (counter % 15 == 0) {

container->createDamageObject(OBJET\_LIGHTER, game\_data->getFatherPos().first);

}

}

counter++;

if (counter > 200) {

father->setOnTakeCtrl(false);

previous\_state = AI\_STATE\_ATK\_LIGHTER;

current\_state = AI\_STATE\_IDLE;

counter = 0;

}

}

}

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

CBossAI.h

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

#include "Definitions.h"

#ifndef CBOSSAI\_H

#define CBOSSAI\_H

#define AI\_STATE\_IDLE -1

#define AI\_STATE\_MOVE\_LEFT 0

#define AI\_STATE\_MOVE\_RIGHT 1

#define AI\_STATE\_ATK\_WINNIE\_1 2

#define AI\_STATE\_ATK\_WINNIE\_2 3

#define AI\_STATE\_ATK\_PEN\_R 4

#define AI\_STATE\_ATK\_PEN\_L 5

#define AI\_STATE\_ATK\_FISH 6

#define AI\_STATE\_ATK\_PEN\_R\_2 7

#define AI\_STATE\_ATK\_PEN\_L\_2 8

#define AI\_STATE\_ATK\_PILL 9

#define AI\_STATE\_ATK\_LIGHTER 10

#define AI\_TOTAL\_STATE\_COUNT 11

namespace game\_framework {

class CBossAI {

public:

CBossAI();

CBossAI(CFather\*, CGameData\*, CDamageContainer\*, CBlood\*);

~CBossAI();

void OnLoop();

void setAILock(bool);

private:

bool moveToTargetPos(int, int);

void stateMoveRight();

void stateMoveLeft();

void stateWinnieATK1();

void stateWinnieATK2();

void stateFishATK();

void statePenATKR();

void statePenATKL();

void statePenATKR2();

void statePenATKL2();

void statePillATK();

void stateLighterATK();

void nextStepChoise();

int \*\*decision\_probiality\_1;

int \*\*decision\_probiality\_2;

CFather\* father;

CGameData\* game\_data;

CDamageContainer\* container;

CBlood\* blood;

bool AI\_Lock;

int counter;

int gap\_counter;

int current\_state;

int previous\_state;

};

}

#endif

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

CDamageContainer.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CDamageContainer.h"

#include "CDamageObject.h"

#include "CDamagePen.h"

#include "CDamageFish.h"

#include "CDamageWinnie.h"

#include "CDamagePill.h"

#include "CDamageLighter.h"

#include "CBlood.h"

#include "CDaughter.h"

namespace game\_framework {

CDamageContainer::CDamageContainer() {

this->winnie\_loop = 0;

}

CDamageContainer::~CDamageContainer() {

}

void CDamageContainer::OnShow() {

for (auto it: container) {

it->OnShow();

}

this->grabageCollect();

}

void CDamageContainer::damageTake(CBlood \* blood,CDaughter\* d,std::pair<int,int> pos) {

int x\_mid = pos.first + (d->getW() / 2);

int y\_mid = pos.second + (d->getH() / 2);

for (auto it : container) {

pos = it->getPos();

if (x\_mid > pos.first && x\_mid<pos.first + it->getW() && y\_mid>pos.second && y\_mid < pos.second + it->getH()) {

blood->setDaughterDamage(it->getDmg(),it->getInvincibleTime());

break;

}

}

}

void CDamageContainer::OnMove() {

for (auto it : container) {

it->OnMove();

}

objectLoop();

}

void CDamageContainer::grabageCollect() {

vector<CDamageObject\*> temp;

for (auto it : container) {

if (it->isEnd()) {

delete it;

}

else {

temp.push\_back(it);

}

}

container.clear();

container = temp;

}

void CDamageContainer::createDamageObject(int type,int x,int y,CGameData\* game\_data) {

if (type == OBJET\_PEN) {

CDamagePen\* pen=new CDamagePen();

pen->loadBitmap();

pen->setTopLeft(x, -600);

container.push\_back(pen);

}

if (type == OBJET\_FISH) {

CDamageFish\* fish = new CDamageFish();

fish->loadBitmap();

container.push\_back(fish);

}

if (type == OBJET\_WINNIE\_1) {

CDamageWinnie\* paper1= new CDamageWinnie(0);

CDamageWinnie\* paper2 = new CDamageWinnie(1.2);

CDamageWinnie\* paper3 = new CDamageWinnie(-1.2);

CDamageWinnie\* paper4 = new CDamageWinnie(0.6);

CDamageWinnie\* paper5 = new CDamageWinnie(-0.6);

paper1->loadBitmap();

paper2->loadBitmap();

paper3->loadBitmap();

paper4->loadBitmap();

paper5->loadBitmap();

container.push\_back(paper1);

container.push\_back(paper2);

container.push\_back(paper3);

container.push\_back(paper4);

container.push\_back(paper5);

}

if (type == OBJET\_WINNIE\_2) {

winnie\_loop = 10;

}

if (type == OBJET\_WINNIE\_3) {

winnie\_loop = -10;

}

if (type == OBJET\_PILL) {

CDamagePill\* pill = new CDamagePill(game\_data);

pill->loadBitmap();

container.push\_back(pill);

}

if (type == OBJET\_LIGHTER) {

CDamageLighter\* lighter = new CDamageLighter(x);

lighter->loadBitmap();

container.push\_back(lighter);

}

}

void CDamageContainer::objectLoop() {

if (winnie\_loop > 0) {

CDamageWinnie\* paper = new CDamageWinnie((winnie\_loop-5)\*0.5);

paper->loadBitmap();

container.push\_back(paper);

winnie\_loop--;

}else if (winnie\_loop < 0) {

CDamageWinnie\* paper = new CDamageWinnie((winnie\_loop + 5)\*0.5);

paper->loadBitmap();

container.push\_back(paper);

winnie\_loop++;

}

}

}

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

CDamageContainer.h

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

#pragma once

#include "CDamageObject.h"

#include "CBlood.h"

#include "CDaughter.h"

#ifndef CDAMAGECONTAINER\_H

#define CDAMAGECONTAINER\_H

#define OBJET\_IDEL -1

#define OBJET\_PEN 0

#define OBJET\_LIGHTER 1

#define OBJET\_WINNIE\_1 2

#define OBJET\_WINNIE\_2 3

#define OBJET\_WINNIE\_3 4

#define OBJET\_PILL 5

#define OBJET\_FISH 6

namespace game\_framework {

class CDamageContainer {

public:

CDamageContainer();

~CDamageContainer();

void damageTake(CBlood \*,CDaughter \*,std::pair<int,int>);

void createDamageObject(int, int = 0, int = 0, CGameData\* = nullptr);

void OnShow();

void OnMove();

private:

void grabageCollect();

void objectLoop();

vector<CDamageObject\*> container;

int type;

int winnie\_loop;

};

}

#endif

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

CDamageFish.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CDamageFish.h"

namespace game\_framework {

CDamageFish::CDamageFish():

CDamageObject(800, 400, 7, 1){

setTopLeft(0, 600);

this->counter = 0;

this->flash = 0;

}

CDamageFish::~CDamageFish() {

}

void CDamageFish::loadBitmap() {

pic.AddBitmap(IDB\_FISH1, RGB(0, 0, 255));

pic.AddBitmap(IDB\_FISH2, RGB(0, 0, 255));

pic.AddBitmap(IDB\_FISH3, RGB(0, 0, 255));

pic.AddBitmap(IDB\_FISH4, RGB(0, 0, 255));

pic.AddBitmap(IDB\_FISH5, RGB(0, 0, 255));

pic.AddBitmap(IDB\_FISH6, RGB(0, 0, 255));

pic.AddBitmap(IDB\_FISH7, RGB(0, 0, 255));

pic.SetDelayCount(4);

alarm.LoadBitmap(IDB\_ALARM\_LONG, RGB(255, 255, 255));

}

void CDamageFish::OnMove() {

if (counter > 30) {

int p = pic.GetCurrentBitmapNumber();

setTopLeft(x\_range[p], y\_range[p]);

pic.OnMove();

}

if (pic.GetCurrentBitmapNumber() == 1) {

CAudio::Instance()->Play(SEA\_WAVE, false);

}

if (counter % 4 == 0) { flash = (flash + 1) % 2; }

counter++;

}

void CDamageFish::OnShow() {

if (counter > 30) {

pic.SetTopLeft(100, 100);

pic.OnShow();

}

else {

if (flash) {

alarm.SetTopLeft(100, 400);

alarm.ShowBitmap();

}

}

}

}

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

CDamageFish.h

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

#pragma once

#include "CDamageObject.h"

#ifndef CDAMAGEFISH\_H

#define CDAMAGEFISH\_H

namespace game\_framework {

class CDamageFish : public CDamageObject {

public:

CDamageFish();

~CDamageFish();

void OnMove();

void loadBitmap();

void OnShow();

bool isEnd();

protected:

const int x\_range[7] = { 700,560,385,110,110,110,110 };

const int y\_range[7] = { 400,400,240,170,170,170,230 };

CMovingBitmap alarm;

CAnimation pic;

int counter;

int flash;

};

inline bool CDamageFish::isEnd() {

return counter > 64;

}

}

#endif

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

CDamageLighter.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CDamageLighter.h"

namespace game\_framework {

CDamageLighter::CDamageLighter():

CDamageObject(110, 150, 25, 30),speed(40) {

lighter\_pos = std::make\_pair(350, 50);

setTopLeft(328, -100);

counter = 0;

}

CDamageLighter::CDamageLighter(int x):

CDamageObject(110, 150, 20, 30), speed(40) {

lighter\_pos = std::make\_pair(x, 50);

setTopLeft(x-22, -100);

counter = 0;

}

CDamageLighter::~CDamageLighter(){

}

void CDamageLighter::OnMove() {

lighter\_pos.second += speed;

speed += 2;

if (lighter\_pos.second >= 402) {

lighter\_pos.second = 402;

}

if (counter > 30) {

explosion.OnMove();

}

if (explosion.GetCurrentBitmapNumber() == 2) {

pos.second = 350;

CAudio::Instance()->Play(EXPLO\_SOUND, false);

}

counter++;

}

void CDamageLighter::loadBitmap() {

explosion.AddBitmap(IDB\_BOOM\_0, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_1, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_2, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_3, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_4, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_5, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_6, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_7, RGB(0, 0, 0));

explosion.AddBitmap(IDB\_BOOM\_8, RGB(0, 0, 0));

explosion.SetDelayCount(2);

lighter.LoadBitmap(IDB\_LIGHTER, RGB(0, 0, 0));

}

void CDamageLighter::OnShow() {

lighter.SetTopLeft(lighter\_pos.first, lighter\_pos.second);

explosion.SetTopLeft(pos.first, 400);

if (counter <= 45) {

lighter.ShowBitmap();

}

if (counter >= 20) {

explosion.OnShow();

}

}

}

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

CDamageLighter.h

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

#pragma once

#include "CDamageObject.h"

#ifndef CDAMAGELIGHTER\_H

#define CDAMAGELIGHTER\_H

namespace game\_framework {

class CDamageLighter : public CDamageObject {

public:

CDamageLighter();

CDamageLighter(int);

~CDamageLighter();

void OnMove();

void loadBitmap();

void OnShow();

bool isEnd();

private:

std::pair<int, int> lighter\_pos;

CMovingBitmap lighter;

CAnimation explosion;

int speed;

int counter;

};

inline bool CDamageLighter::isEnd() {

return counter > 50;

}

}

#endif

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

CDamageObject.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CDamageObject.h"

#include "CGameData.h"

namespace game\_framework {

CDamageObject::CDamageObject():

W(0),H(0),damage(0), invincible\_time(30){

}

CDamageObject::CDamageObject(int W,int H,int damage,int invincible\_time) {

this->W = W;

this->H = H;

this->damage = damage;

this->invincible\_time = invincible\_time;

}

CDamageObject::~CDamageObject() {

}

void CDamageObject::setTopLeft(int x,int y) {

pos.first=x;

pos.second = y;

}

}

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

CDamageObject.h

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

#pragma once

#ifndef CDAMAGEOBJECT\_H

#define CDAMAGEOBJECT\_H

namespace game\_framework {

class CDamageObject {

public:

CDamageObject();

CDamageObject(int, int, int, int);

virtual ~CDamageObject();

void setTopLeft(int, int);

int getW() const;

int getH() const;

int getDmg() const;

int getInvincibleTime() const;

std::pair<int, int> getPos() const;

virtual void OnMove() = 0;

virtual void OnShow() = 0;

virtual bool isEnd() = 0;

protected:

CMovingBitmap pic;

std::pair<int, int> pos;

int W, H;

int damage;

int invincible\_time;

};

inline int CDamageObject::getW() const {

return W;

}

inline int CDamageObject::getH() const {

return H;

}

inline int CDamageObject::getDmg() const {

return damage;

}

inline int CDamageObject::getInvincibleTime() const {

return invincible\_time;

}

inline std::pair<int, int> CDamageObject::getPos() const {

return pos;

}

}

#endif

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

CDamagePen.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CDamagePen.h"

namespace game\_framework {

CDamagePen::CDamagePen() :

CDamageObject(100, 600, 40, 30),speed(30){

this->counter = 0;

this->flash = 0;

this->play = false;

}

CDamagePen::~CDamagePen() {

}

void CDamagePen::loadBitmap() {

pic.LoadBitmap(IDB\_PEN, RGB(0, 0, 255));

alarm.LoadBitmap(IDB\_ALARMPIC, RGB(255, 255, 255));

}

void CDamagePen::OnMove() {

if (counter > 30) {

pos.second += speed;

speed += 2;

if (pos.second > -100) {

CAudio::Instance()->Play(G\_HIT, false);

pos.second = -100;

}

}

if (counter % 4 == 0) { flash = (flash + 1) % 2; }

counter++;

}

void CDamagePen::OnShow() {

if (counter > 30) {

pic.SetTopLeft(pos.first, pos.second);

pic.ShowBitmap();

}

else {

if (flash) {

alarm.SetTopLeft(pos.first, 400);

alarm.ShowBitmap();

}

}

}

}

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

CDamagePen.h

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

#pragma once

#include "CDamageObject.h"

#ifndef CDAMAGEPEN\_H

#define CDAMAGEPEN\_H

namespace game\_framework {

class CDamagePen : public CDamageObject {

public:

CDamagePen();

~CDamagePen();

void OnMove();

void loadBitmap();

void OnShow();

bool isEnd();

private:

CMovingBitmap alarm;

int speed;

int counter;

int flash;

bool play;

};

inline bool CDamagePen::isEnd() {

return pos.second >= -100;

}

}

#endif

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

CDamagePill.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CDamagePill.h"

#include "CGameData.h"

namespace game\_framework {

CDamagePill::CDamagePill():

CDamageObject(50, 50, 15, 5) {

game\_data = nullptr;

counter = 0;

setTopLeft(350, 100);

}

CDamagePill::CDamagePill(CGameData\* game\_data):

CDamageObject(50, 50, 15, 5) {

this->game\_data = game\_data;

target\_pos = game\_data->getDaughterPos();

target\_pos.second += 50;

counter = 0;

setTopLeft(350, 100);

}

CDamagePill::~CDamagePill() {

}

void CDamagePill::OnMove() {

if (counter < 10) {

target\_pos = game\_data->getDaughterPos();

target\_pos.second += 50;

move\_vector = make\_pair(target\_pos.first-325, target\_pos.second-75);

}

else {

pos.first += (move\_vector.first/15);

pos.second += (move\_vector.second/15);

}

counter++;

}

void CDamagePill::loadBitmap() {

aim.LoadBitmap(IDB\_AIM, RGB(255, 255, 255));

pill.LoadBitmap(IDB\_PILL, RGB(0, 0, 0));

}

void CDamagePill::OnShow() {

aim.SetTopLeft(target\_pos.first, target\_pos.second);

aim.ShowBitmap();

if (counter >= 10) {

pill.SetTopLeft(pos.first, pos.second);

pill.ShowBitmap();

}

}

}

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

CDamagePill.h

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

#pragma once

#include "CDamageObject.h"

#include "CGameData.h"

#ifndef CDAMAGEPILL\_H

#define CDAMAGEPILL\_H

namespace game\_framework {

class CDamagePill : public CDamageObject {

public:

CDamagePill();

CDamagePill(CGameData\*);

~CDamagePill();

void OnMove();

void loadBitmap();

void OnShow();

bool isEnd();

private:

std::pair<int, int> target\_pos;

std::pair<int, int> move\_vector;

CGameData\* game\_data;

CMovingBitmap aim;

CMovingBitmap pill;

int counter;

};

inline bool CDamagePill::isEnd() {

return counter > 30;

}

}

#endif

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

CDamageWinnie.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CDamageWinnie.h"

namespace game\_framework {

CDamageWinnie::CDamageWinnie(double a) :

CDamageObject(70, 100, 20, 5), speed(30) {

this->counter = 0;

this->flash = 0;

this->coef\_a = a;

pos = std::make\_pair(350, 100);

}

CDamageWinnie::~CDamageWinnie() {

}

void CDamageWinnie::loadBitmap() {

pic.LoadBitmap(IDB\_WINNIE,RGB(0,0,0));

}

void CDamageWinnie::OnMove() {

if (counter < 30) {

pos.second -= speed;

pos.first = int((double(pos.second)\*coef\_a) + coef\_b);

speed = (speed-3 < 0 ? 0 : speed-3);

}

else {

pos.second += speed;

pos.first = int((double(pos.second)\*coef\_a) + coef\_b);

speed += 2;

}

counter++;

}

void CDamageWinnie::OnShow() {

pic.SetTopLeft(pos.first, pos.second);

pic.ShowBitmap();

}

}

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

CDamageWinnie.h

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

#pragma once

#include "CDamageObject.h"

#ifndef CDAMAGEWINNIE\_H

#define CDAMAGEWINNIE\_H

namespace game\_framework {

class CDamageWinnie : public CDamageObject {

public:

CDamageWinnie(double a);

~CDamageWinnie();

void OnMove();

void loadBitmap();

void OnShow();

bool isEnd();

private:

const double coef\_b = 350;

int speed;

int counter;

int flash;

double coef\_a;

};

inline bool CDamageWinnie::isEnd() {

return counter > 60;

}

}

#endif

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

CDaughter.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CGameData.h"

#include "CDaughter.h"

namespace game\_framework {

CDaughter::CDaughter():

x\_speed(5),atk\_counter(0),w(DAUGHTER\_STAND\_W),h(DAUGHTER\_STAND\_H){

this->jump = false;

this->move = false;

this->attack = false;

this->slide = false;

this->lock = false;

this->game\_data = nullptr;

}

CDaughter::CDaughter(CGameData \* game\_data):

x\_speed(5),atk\_counter(0),w(DAUGHTER\_STAND\_W),h(DAUGHTER\_STAND\_H){

this->jump = false;

this->move = false;

this->attack = false;

this->slide = false;

this->lock = false;

this->game\_data = game\_data;

}

CDaughter::~CDaughter() {

}

void CDaughter::LoadBitmap() {

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_1, RGB(0, 255, 0));

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_2, RGB(0, 255, 0));

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_3, RGB(0, 255, 0));

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_4, RGB(0, 255, 0));

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_5, RGB(0, 255, 0));

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_6, RGB(0, 255, 0));

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_7, RGB(0, 255, 0));

walk\_R.AddBitmap(IDB\_DAUGHTER\_WALK\_R\_8, RGB(0, 255, 0));

walk\_R.SetDelayCount(3);

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_1, RGB(0, 255, 0));

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_2, RGB(0, 255, 0));

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_3, RGB(0, 255, 0));

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_4, RGB(0, 255, 0));

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_5, RGB(0, 255, 0));

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_6, RGB(0, 255, 0));

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_7, RGB(0, 255, 0));

walk\_L.AddBitmap(IDB\_DAUGHTER\_WALK\_L\_8, RGB(0, 255, 0));

walk\_L.SetDelayCount(3);

stand\_R.LoadBitmap(IDB\_DAUGHTER\_STAND\_R, RGB(0, 255, 0));

stand\_L.LoadBitmap(IDB\_DAUGHTER\_STAND\_L, RGB(0, 255, 0));

jump\_R.LoadBitmap(IDB\_DAUGHTER\_JUMP\_R, RGB(0, 255, 0));

jump\_L.LoadBitmap(IDB\_DAUGHTER\_JUMP\_L, RGB(0, 255, 0));

fall\_R.LoadBitmap(IDB\_DAUGHTER\_FALL\_R, RGB(0, 255, 0));

fall\_L.LoadBitmap(IDB\_DAUGHTER\_FALL\_L, RGB(0, 255, 0));

fly\_R.LoadBitmap(IDB\_DAUGHTER\_FLY\_R, RGB(0, 255, 0));

fly\_L.LoadBitmap(IDB\_DAUGHTER\_FLY\_L, RGB(0, 255, 0));

atk\_R[0].LoadBitmap(IDB\_DAUGHTER\_ATK\_R\_1, RGB(0, 255, 0));

atk\_R[1].LoadBitmap(IDB\_DAUGHTER\_ATK\_R\_2, RGB(0, 255, 0));

atk\_R[2].LoadBitmap(IDB\_DAUGHTER\_ATK\_R\_3, RGB(0, 255, 0));

atk\_L[0].LoadBitmap(IDB\_DAUGHTER\_ATK\_L\_1, RGB(0, 255, 0));

atk\_L[1].LoadBitmap(IDB\_DAUGHTER\_ATK\_L\_2, RGB(0, 255, 0));

atk\_L[2].LoadBitmap(IDB\_DAUGHTER\_ATK\_L\_3, RGB(0, 255, 0));

slide\_L.LoadBitmap(IDB\_DAUGHTER\_SLIDING\_L, RGB(0, 255, 0));

slide\_R.LoadBitmap(IDB\_DAUGHTER\_SLIDING\_R, RGB(0, 255, 0));

front = &stand\_R;

}

void CDaughter::OnShow() {

std::pair<int, int> pos = game\_data->getDaughterPos();

if (status == DAUGHTER\_STATUS\_ATK&&faceSide == FACE\_LEFT) pos.first -= DAUGHTER\_STAND\_W;

front->SetTopLeft(pos.first, pos.second);

front->ShowBitmap();

}

void CDaughter::OnMove() {

if (lock) {

move = false;

}

if (fly) {

this->OnFly();

status = DAUGHTER\_STATUS\_FLY;

}

else if (slide) {

this->OnSlide();

status = DAUGHTER\_STATUS\_SLIDE;

}

else if (jump) {

this->OnJump();

status = (y\_speed > 0 ? DAUGHTER\_STATUS\_JUMP : DAUGHTER\_STATUS\_FALL);

}

else if (attack) {

this->OnAttack();

}

else if (move) {

this->OnWalk();

status = DAUGHTER\_STATUS\_WALK;

}

else {

status = DAUGHTER\_STATUS\_STAND;

}

setFront();

}

void CDaughter::OnWalk() {

switch (faceSide) {

case FACE\_RIGHT:

walk\_R.OnMove();

game\_data->setDaughterMove(x\_speed,0);

break;

case FACE\_LEFT:

walk\_L.OnMove();

game\_data->setDaughterMove(-x\_speed,0);

break;

}

}

void CDaughter::OnFly() {

switch (faceSide) {

case FACE\_RIGHT:

game\_data->setDaughterMove(x\_speed, 0);

break;

case FACE\_LEFT:

game\_data->setDaughterMove(-x\_speed, 0);

break;

}

if (x\_speed > 10) {

x\_speed --;

}

else {

fly = false;

x\_speed = 5;

}

}

void CDaughter::OnJump() {

game\_data->setDaughterMove(0, -y\_speed);

y\_speed--;

if (move) {

this->OnWalk();

}

}

void CDaughter::OnAttack() {

if (atk\_delay\_counter > 0) {

status = DAUGHTER\_STATUS\_ATK;

atk\_delay\_counter--;

}

else {

atk\_counter++;

atk\_counter %= 3;

attack = false;

status = DAUGHTER\_STATUS\_STAND;

}

}

void CDaughter::OnSlide() {

switch (faceSide) {

case FACE\_RIGHT:

game\_data->setDaughterMove(x\_speed, 0);

break;

case FACE\_LEFT:

game\_data->setDaughterMove(-x\_speed, 0);

break;

}

if (x\_speed > 10) {

x\_speed--;

}

else {

slide = false;

x\_speed = 5;

h = DAUGHTER\_STAND\_H;

w = DAUGHTER\_STAND\_W;

game\_data->setDaughterMove(0, -100);

}

}

void CDaughter::OnFloorHit() {

y\_speed = initial\_velocity;

jump = false;

second\_jump = false;

}

void CDaughter::OnFootEmpty() {

if (!jump) {

y\_speed = -10;

jump = true;

second\_jump = true;

}

}

void CDaughter::setFront() {

switch (status){

case DAUGHTER\_STATUS\_STAND:

front = (faceSide == FACE\_RIGHT ? &stand\_R : &stand\_L);

break;

case DAUGHTER\_STATUS\_WALK:

front = (faceSide == FACE\_RIGHT ? walk\_R.GetCurrentBitmap() : walk\_L.GetCurrentBitmap());

break;

case DAUGHTER\_STATUS\_JUMP:

front = (faceSide == FACE\_RIGHT ? &jump\_R : &jump\_L);

break;

case DAUGHTER\_STATUS\_FALL:

front = (faceSide == FACE\_RIGHT ? &fall\_R : &fall\_L);

break;

case DAUGHTER\_STATUS\_FLY:

front = (faceSide == FACE\_RIGHT ? &fly\_R : &fly\_L);

break;

case DAUGHTER\_STATUS\_ATK:

front = (faceSide == FACE\_RIGHT ? &atk\_R[atk\_counter] : &atk\_L[atk\_counter]);

break;

case DAUGHTER\_STATUS\_SLIDE:

front = (faceSide == FACE\_RIGHT ? &slide\_R : &slide\_L);

break;

default:

break;

}

}

void CDaughter::OnKeyDown(UINT nChar) {

if (!outro){

switch (nChar) {

case KEY\_RIGHT:

move = true;

faceSide = FACE\_RIGHT;

break;

case KEY\_LEFT:

move = true;

faceSide = FACE\_LEFT;

break;

case KEY\_Z:

if (jump && y\_speed>0) {

y\_speed = 0;

x\_speed = 30;

fly = true;

}

else {

attack = true;

atk\_delay\_counter = DAUGHTER\_ATK\_DELAY\_COUNT;

}

break;

case KEY\_X:

if (!slide && !jump) {

x\_speed = 20;

game\_data->setDaughterMove(0, 100);

h = DAUGHTER\_SLIDE\_H;

w = DAUGHTER\_SLIDE\_W;

slide = true;

CAudio::Instance()->Play(DAUGHTER\_SLIDE, false);

}

break;

default:

break;

}

if (!jump && move) {

CAudio::Instance()->Play(DAUGHTER\_FOOTSTEP, false);

}

}

}

void CDaughter::OnKeyUP(UINT nChar) {

if (!outro){

switch (nChar) {

case KEY\_RIGHT:

if (faceSide == FACE\_RIGHT) {

move = false;

faceSide = FACE\_RIGHT;

CAudio::Instance()->Stop(DAUGHTER\_FOOTSTEP);

}

break;

case KEY\_LEFT:

if (faceSide == FACE\_LEFT) {

move = false;

faceSide = FACE\_LEFT;

CAudio::Instance()->Stop(DAUGHTER\_FOOTSTEP);

}

break;

case KEY\_SPACE:

if (jump && !second\_jump) {

y\_speed = initial\_velocity;

second\_jump = true;

}

else {

jump = true;

}

default:

break;

}

}

}

}

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

CDaughter.h

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

#pragma once

#include "Definitions.h"

#ifndef CDAUGHTER\_H

#define CDAUGHTER\_H

#define DAUGHTER\_STATUS\_STAND 0

#define DAUGHTER\_STATUS\_WALK 1

#define DAUGHTER\_STATUS\_JUMP 2

#define DAUGHTER\_STATUS\_FALL 3

#define DAUGHTER\_STATUS\_FLY 4

#define DAUGHTER\_STATUS\_ATK 5

#define DAUGHTER\_STATUS\_SLIDE 6

#define DAUGHTER\_ATK\_DELAY\_COUNT 3

#define DAUGHTER\_STAND\_H 200

#define DAUGHTER\_STAND\_W 100

#define DAUGHTER\_SLIDE\_H 100

#define DAUGHTER\_SLIDE\_W 200

namespace game\_framework {

class CDaughter {

public:

CDaughter();

CDaughter(CGameData \*);

~CDaughter();

void LoadBitmap();

void OnMove();

void OnShow();

bool isInATK() const;

int getXSpeed() const;

int getYSpeed() const;

int getStatus() const;

int getFaceSide() const;

int getH() const;

int getW() const;

void OnFloorHit();

void OnFootEmpty();

void setFront();

void OnKeyDown(UINT);

void OnKeyUP(UINT);

bool outro = false;

private:

void OnWalk();

void OnFly();

void OnJump();

void OnAttack();

void OnSlide();

const int initial\_velocity = 15;

CGameData\* game\_data;

CMovingBitmap\* front;

CMovingBitmap stand\_R;

CMovingBitmap stand\_L;

CMovingBitmap jump\_R;

CMovingBitmap jump\_L;

CMovingBitmap fall\_R;

CMovingBitmap fall\_L;

CMovingBitmap fly\_R;

CMovingBitmap fly\_L;

CMovingBitmap atk\_R[3];

CMovingBitmap atk\_L[3];

CMovingBitmap slide\_R;

CMovingBitmap slide\_L;

CAnimation walk\_R;

CAnimation walk\_L;

int x, y;

int w, h;

int faceSide;

int y\_speed;

int x\_speed;

int status;

int atk\_delay\_counter;

int atk\_counter;

bool lock;

bool move;

bool fly;

bool attack;

bool jump;

bool second\_jump;

bool slide;

};

inline bool CDaughter::isInATK() const{

return status == DAUGHTER\_STATUS\_ATK || status == DAUGHTER\_STATUS\_FLY;

}

inline int CDaughter::getXSpeed() const {

return x\_speed;

}

inline int CDaughter::getYSpeed() const {

return y\_speed;

}

inline int CDaughter::getStatus() const {

return status;

}

inline int CDaughter::getFaceSide() const {

return faceSide;

}

inline int CDaughter::getH() const {

return h;

}

inline int CDaughter::getW() const {

return w;

}

}

#endif

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

CDialog.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CDialog.h"

namespace game\_framework {

CDialog::CDialog(){

dlg\_iter = dlg.begin();

}

CDialog::~CDialog(){

dlg.clear();

}

void CDialog::LoadBitmap(int state) {

switch (state) {

case 1:

LoadBitmap\_1();

break;

case 2:

LoadBitmap\_2();

break;

case 3:

LoadBitmap\_3();

break;

case 4:

LoadBitmap\_4();

break;

case 5:

LoadBitmap\_5();

break;

case 6:

LoadBitmap\_Fail();

break;

case 7:

LoadBitmap\_Success();

break;

}

}

void CDialog::LoadBitmap\_1(){

dlg.clear();

daughter\_1\_1.LoadBitmap(IDB\_DLG\_DAUGHTER\_1\_1, RGB(0, 255, 0));

daughter\_1\_2.LoadBitmap(IDB\_DLG\_DAUGHTER\_1\_2, RGB(0, 255, 0));

daughter\_1\_3.LoadBitmap(IDB\_DLG\_DAUGHTER\_1\_3, RGB(0, 255, 0));

daughter\_1\_4.LoadBitmap(IDB\_DLG\_DAUGHTER\_1\_4, RGB(0, 255, 0));

dlg.insert(dlg.end(), daughter\_1\_1);

dlg.insert(dlg.end(), daughter\_1\_2);

dlg.insert(dlg.end(), daughter\_1\_3);

dlg.insert(dlg.end(), daughter\_1\_4);

dlg\_iter = dlg.begin();

}

void CDialog::LoadBitmap\_2() {

dlg.clear();

daughter\_2\_1.LoadBitmap(IDB\_DLG\_DAUGHTER\_2\_1, RGB(0, 255, 0));

daughter\_2\_2.LoadBitmap(IDB\_DLG\_DAUGHTER\_2\_2, RGB(0, 255, 0));

daughter\_2\_3.LoadBitmap(IDB\_DLG\_DAUGHTER\_2\_3, RGB(0, 255, 0));

dlg.insert(dlg.end(), daughter\_2\_1);

dlg.insert(dlg.end(), daughter\_2\_2);

dlg.insert(dlg.end(), daughter\_2\_3);

dlg\_iter = dlg.begin();

}

void CDialog::LoadBitmap\_3() {

dlg.clear();

daughter\_3\_1.LoadBitmap(IDB\_DLG\_DAUGHTER\_3\_1, RGB(0, 255, 0));

daughter\_3\_2.LoadBitmap(IDB\_DLG\_DAUGHTER\_3\_2, RGB(0, 255, 0));

dlg.insert(dlg.end(), daughter\_3\_1);

dlg.insert(dlg.end(), daughter\_3\_2);

dlg\_iter = dlg.begin();

}

void CDialog::LoadBitmap\_4() {

dlg.clear();

father\_4.LoadBitmap(IDB\_DLG\_FATHER\_4, RGB(0, 255, 0));

daughter\_4\_1.LoadBitmap(IDB\_DLG\_DAUGHTER\_4\_1, RGB(0, 255, 0));

daughter\_4\_2.LoadBitmap(IDB\_DLG\_DAUGHTER\_4\_2, RGB(0, 255, 0));

dlg.insert(dlg.end(), father\_4);

dlg.insert(dlg.end(), daughter\_4\_1);

dlg.insert(dlg.end(), daughter\_4\_2);

dlg\_iter = dlg.begin();

}

void CDialog::LoadBitmap\_5() {

dlg.clear();

daughter\_5\_1.LoadBitmap(IDB\_DLG\_DAUGHTER\_5\_1, RGB(0, 255, 0));

father\_5\_2.LoadBitmap(IDB\_DLG\_FATHER\_5\_2, RGB(0, 255, 0));

dlg.insert(dlg.end(), daughter\_5\_1);

dlg.insert(dlg.end(), father\_5\_2);

dlg\_iter = dlg.begin();

}

void CDialog::LoadBitmap\_Fail() {

dlg.clear();

father\_fail.LoadBitmap(IDB\_DLG\_FATHER\_FAIL, RGB(0, 255, 0));

dlg.insert(dlg.end(), father\_fail);

dlg\_iter = dlg.begin();

}

void CDialog::LoadBitmap\_Success() {

dlg.clear();

daughter\_success.LoadBitmap(IDB\_DLG\_DAUGHTER\_SUCCESS, RGB(0, 255, 0));

father\_success.LoadBitmap(IDB\_DLG\_FATHER\_SUCCESS, RGB(0, 255, 0));

dlg.insert(dlg.end(), daughter\_success);

dlg.insert(dlg.end(), father\_success);

dlg\_iter = dlg.begin();

}

void CDialog::OnKeyDown(UINT nChar) {

if (nChar == KEY\_ENTER && dlg\_iter != dlg.end()) {

dlg\_iter++; //按下enter就換一張圖

}

}

void CDialog::OnShow(){

if (dlg\_iter != dlg.end()) {

dlg\_iter->SetTopLeft(0, 400);

dlg\_iter->ShowBitmap();

}

}

}

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

CDialog.h

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

#pragma once

#include <list>

#include "Definitions.h"

#ifndef CDIALOG\_H

#define CDIALOG\_H

namespace game\_framework {

class CDialog {

public:

CDialog();

~CDialog();

void LoadBitmap(int);

void OnKeyDown(UINT);

bool IsFinished() const; //對話結束

void OnShow();

private:

void LoadBitmap\_1();

void LoadBitmap\_2();

void LoadBitmap\_3();

void LoadBitmap\_4();

void LoadBitmap\_5();

void LoadBitmap\_Fail();

void LoadBitmap\_Success();

list<CMovingBitmap> dlg;

list<CMovingBitmap>::iterator dlg\_iter;

CMovingBitmap daughter\_1\_1;

CMovingBitmap daughter\_1\_2;

CMovingBitmap daughter\_1\_3;

CMovingBitmap daughter\_1\_4;

CMovingBitmap daughter\_2\_1;

CMovingBitmap daughter\_2\_2;

CMovingBitmap daughter\_2\_3;

CMovingBitmap daughter\_3\_1;

CMovingBitmap daughter\_3\_2;

CMovingBitmap father\_4;

CMovingBitmap daughter\_4\_1;

CMovingBitmap daughter\_4\_2;

CMovingBitmap daughter\_5\_1;

CMovingBitmap father\_5\_2;

CMovingBitmap father\_fail;

CMovingBitmap daughter\_success;

CMovingBitmap father\_success;

};

inline bool CDialog::IsFinished() const{

return dlg\_iter == dlg.end();

}

}

#endif

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

CFather.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CGameData.h"

#include "CBlood.h"

#include "CFather.h"

namespace game\_framework {

CFather::CFather():

x\_speed(5), y\_speed(0) {

fall = false;

this->game\_data = nullptr;

this->blood = nullptr;

}

CFather::CFather(CGameData \* game\_data,CBlood\* blood):

x\_speed(5), y\_speed(0) {

fall = false;

this->game\_data = game\_data;

this->blood = blood;

}

CFather::~CFather() {

}

void CFather::LoadBitmap() {

walk\_R.AddBitmap(IDB\_FATHER\_WALK\_R\_1, RGB(0, 0, 255));

walk\_R.AddBitmap(IDB\_FATHER\_WALK\_R\_2, RGB(0, 0, 255));

walk\_R.AddBitmap(IDB\_FATHER\_WALK\_R\_3, RGB(0, 0, 255));

walk\_R.AddBitmap(IDB\_FATHER\_WALK\_R\_4, RGB(0, 0, 255));

walk\_R.SetDelayCount(6);

walk\_L.AddBitmap(IDB\_FATHER\_WALK\_L\_1, RGB(0, 0, 255));

walk\_L.AddBitmap(IDB\_FATHER\_WALK\_L\_2, RGB(0, 0, 255));

walk\_L.AddBitmap(IDB\_FATHER\_WALK\_L\_3, RGB(0, 0, 255));

walk\_L.AddBitmap(IDB\_FATHER\_WALK\_L\_4, RGB(0, 0, 255));

walk\_L.SetDelayCount(6);

stop\_L.LoadBitmap(IDB\_FATHER\_WALK\_L\_2, RGB(0, 0, 255));

stop\_R.LoadBitmap(IDB\_FATHER\_WALK\_R\_2, RGB(0, 0, 255));

stand\_1.LoadBitmap(IDB\_FATHER\_STAND\_1, RGB(0, 0, 255));

stand\_2.LoadBitmap(IDB\_FATHER\_STAND\_2, RGB(0, 0, 255));

front = &stop\_L;

}

void CFather::OnShow() {

std::pair<int, int> pos = game\_data->getFatherPos();

front->SetTopLeft(pos.first, pos.second);

front->ShowBitmap();

}

void CFather::OnMove() {

if (game\_data->getFatherPos().second<=265 && !isBeCtrled) {

this->OnFall();

status = FATHER\_STATUS\_STAND;

}

else if (move) {

this->OnWalk();

status = FATHER\_STATUS\_WALK;

}

else{

status = FATHER\_STATUS\_STAND;

}

if (game\_data->getFatherPos().second > 270) {

game\_data->setFatherTopLeft(game\_data->getFatherPos().first, 270);

y\_speed = 0;

}

setFront();

}

void CFather::OnWalk() {

switch (faceSide) {

case FACE\_RIGHT:

walk\_R.OnMove();

game\_data->setFatherMove(x\_speed, 0);

break;

case FACE\_LEFT:

walk\_L.OnMove();

game\_data->setFatherMove(-x\_speed, 0);

break;

}

}

void CFather::DamageTake() {

blood->setFatherDamage(3);

}

void CFather::OnFall(){

game\_data->setFatherMove(0, -y\_speed);

y\_speed--;

if (move) {

this->OnWalk();

}

}

void CFather::setOnTakeCtrl(bool ctrl) {

isBeCtrled = ctrl;

}

void CFather::setStatus(int s) {

status = s;

setFront();

}

void CFather::setFaceSide(int f) {

faceSide = f;

setFront();

}

void CFather::setFront() {

switch (status) {

case FATHER\_STATUS\_STAND:

front = (faceSide == FACE\_RIGHT ? &stop\_R : &stop\_L);

break;

case FATHER\_STATUS\_WALK:

front = (faceSide == FACE\_RIGHT ? walk\_R.GetCurrentBitmap() : walk\_L.GetCurrentBitmap());

break;

case FATHER\_STATUS\_FACE\_1:

front = &stand\_1;

break;

case FATHER\_STATUS\_FACE\_2:

front = &stand\_2;

break;

}

}

void CFather::OnKeyDown(UINT nChar) {

switch (nChar) {

case KEY\_RIGHT:

move = true;

faceSide = FACE\_RIGHT;

break;

case KEY\_LEFT:

move = true;

faceSide = FACE\_LEFT;

break;

}

}

void CFather::OnKeyUP(UINT nChar) {

switch (nChar) {

case KEY\_RIGHT:

if (faceSide == FACE\_RIGHT) {

move = false;

faceSide = FACE\_RIGHT;

}

break;

case KEY\_LEFT:

if (faceSide == FACE\_LEFT) {

move = false;

faceSide = FACE\_LEFT;

}

break;

}

}

}

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

CFather.h

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

#pragma once

#include "Definitions.h"

#ifndef CFATHER\_H

#define CFATHER\_H

#define FATHER\_STATUS\_STAND 0

#define FATHER\_STATUS\_WALK 1

#define FATHER\_STATUS\_FACE\_1 2

#define FATHER\_STATUS\_FACE\_2 3

#define FATHER\_H 230

#define FATHER\_W 100

namespace game\_framework {

class CFather {

public:

CFather();

CFather(CGameData\*,CBlood\*);

~CFather();

void LoadBitmap();

void OnShow();

void OnMove();

void setOnTakeCtrl(bool);

void setStatus(int);

void setFaceSide(int);

void DamageTake();

void OnKeyDown(UINT);

void OnKeyUP(UINT);

private:

void OnWalk();

void setFront();

void OnFall();

CGameData\* game\_data;

CBlood \* blood;

CMovingBitmap\* front;

CAnimation walk\_R;

CAnimation walk\_L;

CMovingBitmap stop\_R;

CMovingBitmap stop\_L;

CMovingBitmap stand\_1;

CMovingBitmap stand\_2;

bool move;

bool fall;

bool isBeCtrled;

int faceSide;

int status;

int x\_speed;

int y\_speed;

};

}

#endif

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

CGameData.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include <utility>

#include "CDaughter.h"

#include "CFather.h"

#include "CGameData.h"

#include "CGameMap.h"

namespace game\_framework {

CGameData::CGameData(){

}

CGameData::~CGameData(){

}

void CGameData::setDaughterTopLeft(int x, int y) {

daughter = std::make\_pair(x, y);

}

void CGameData::setFatherTopLeft(int x, int y) {

father = std::make\_pair(x, y);

}

void CGameData::setMapTopLeft(int x, int y) {

map = std::make\_pair(x, y);

}

void CGameData::setDaughterMove(int x, int y) {

daughter.first += x;

daughter.second += y;

}

void CGameData::setFatherMove(int x, int y) {

father.first += x;

father.second += y;

}

void CGameData::setMapMove(int x, int y) {

map.first += x;

map.second += y;

}

void CGameData::checkMove(CGameMap \*m, CDaughter \*d, CFather \*f){

int x\_right = daughter.first + map.first;

int y\_top = daughter.second + map.second;

int x\_mid = x\_right + (d->getW() / 2);

int y\_mid = y\_top + (d->getH() / 2);

int x\_left = x\_right + d->getW();

int y\_bot = y\_top + d->getH();

if (m->isBlock(x\_mid, y\_bot)) {

d->OnFloorHit();

daughter.second = m->getBlockHeight(y\_bot) - d->getH() - map.second;

}

if (m->isBlock(x\_left, y\_mid)) {

daughter.first = m->getBlockLeft(x\_left) - d->getW() - map.first;

}

if (m->isBlock(x\_right, y\_mid)) {

daughter.first = m->getBlockRight(x\_right) - map.first;

}

if (m->isBlock(x\_mid, y\_top)) {

daughter.second = m->getBlockBot(y\_top) - map.second;

}

if (m->isEmpty(x\_mid, y\_bot)) {

d->OnFootEmpty();

}

if (daughter.first + d->getW() > 700 && map.first + SIZE\_X < m->getMapW()){

daughter.first = 699 - d->getW();

setMapMove(d->getXSpeed(), 0);

}

else if (daughter.first < 100 && map.first >0) {

daughter.first = 101;

setMapMove(-d->getXSpeed(), 0);

}

if (x\_right<0) {

daughter.first = 0;

}

else if (x\_left > m->getMapW()) {

daughter.first = SIZE\_X - d->getW();

}

if (map.first < 0) {

map.first = 0;

}

else if (map.first + SIZE\_X > m->getMapW()) {

map.first = m->getMapW() - SIZE\_X;

}

if (f != nullptr) {

if (d->isInATK()) {

checkDamage(d, f);

}

}

}

void CGameData::checkDamage(CDaughter \*d, CFather \*f){

int d\_x\_left = daughter.first;

int d\_x\_right = d\_x\_left + d->getW();

int d\_x\_left\_border = daughter.first- d->getW();

int d\_x\_right\_border = d\_x\_left + (d->getW()\*2);

int d\_y\_top = daughter.second;

int d\_y\_bot = daughter.second + d->getH();

int f\_x\_mid = ((father.first\*2) + FATHER\_W)/2;

int f\_y\_mid = father.second + (FATHER\_W/2);

if (d->getFaceSide() == FACE\_LEFT) {

if (f\_x\_mid > d\_x\_left\_border && f\_x\_mid<d\_x\_left && f\_y\_mid>d\_y\_top && f\_y\_mid < d\_y\_bot) {

f->DamageTake();

}

}

else {

if (f\_x\_mid < d\_x\_right\_border && f\_x\_mid>d\_x\_right && f\_y\_mid>d\_y\_top && f\_y\_mid < d\_y\_bot) {

f->DamageTake();

}

}

}

}

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

CGameData.h

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

#pragma once

#include "Definitions.h"

#ifndef CGAMEDATA\_H

#define CGAMEDATA\_H

namespace game\_framework {

class CGameData {

public:

CGameData();

~CGameData();

void setDaughterTopLeft(int, int);

void setFatherTopLeft(int, int);

void setMapTopLeft(int, int);

void setDaughterMove(int, int);

void setFatherMove(int, int);

void setMapMove(int, int);

std::pair<int, int> getDaughterPos() const;

std::pair<int, int> getFatherPos() const;

std::pair<int, int> getMapPos() const;

void checkMove(CGameMap \*,CDaughter \*,CFather \*);

void checkDamage(CDaughter \*, CFather \*);

private:

std::pair<int, int> daughter;

std::pair<int, int> map;

std::pair<int, int> father;

};

inline std::pair<int, int> CGameData::getDaughterPos() const{

return daughter;

}

inline std::pair<int, int> CGameData::getFatherPos() const{

return father;

}

inline std::pair<int, int> CGameData::getMapPos() const{

return map;

}

}

#endif

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

CGameMap.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CGameData.h"

#include "CGameMap.h"

namespace game\_framework {

CGameMap::CGameMap():

map\_w(0), map\_h(0), box\_num\_x(0), box\_num\_y(0){

this->map\_data = nullptr;

this->game\_data = nullptr;

}

CGameMap::CGameMap(CGameData \*game\_data, int map\_w,int map\_h){

this->map\_w = map\_w;

this->map\_h = map\_h;

this->box\_num\_x = map\_w / BOX\_W;

this->box\_num\_y = map\_h / BOX\_H;

this->game\_data = game\_data;

}

CGameMap::~CGameMap() {

for (int i = 0; i < box\_num\_y; i++) {

delete[] map\_data[i];

}

delete[] map\_data;

}

void CGameMap::LoadBitmap(int IDB\_BITMAP) {

map.LoadBitmap(IDB\_BITMAP);

}

void CGameMap::OnShow() {

std::pair<int, int> pos = game\_data->getMapPos();

map.SetTopLeft(-pos.first, -pos.second);

map.ShowBitmap();

}

}

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

CGameMap.h

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

#pragma once

#include "Definitions.h"

#ifndef CGAMEMAP\_H

#define CGAMEMAP\_H

#define MAP\_EMPTY 0

#define MAP\_BLOCK 1

namespace game\_framework {

class CGameMap {

public:

CGameMap();

CGameMap(CGameData \*, int, int);

virtual ~CGameMap();

void LoadBitmap(int);

int getBlockHeight(int);

int getBlockBot(int);

int getBlockLeft(int);

int getBlockRight(int);

int getMapH();

int getMapW();

bool isBlock(int, int);

bool isEmpty(int, int);

void OnShow();

protected:

int \*\*map\_data;

int box\_num\_x, box\_num\_y;

private:

CGameData \*game\_data;

CMovingBitmap map;

int map\_w;

int map\_h;

};

inline int CGameMap::getMapH() {

return map\_h;

}

inline int CGameMap::getMapW() {

return map\_w;

}

inline int CGameMap::getBlockHeight(int y) {

int gy = (y / BOX\_H)\*BOX\_H;

return gy;

}

inline int CGameMap::getBlockBot(int y) {

int gy = (y / BOX\_H + 1)\*BOX\_H;

return gy;

}

inline int CGameMap::getBlockLeft(int x) {

int gx = (x / BOX\_W)\*BOX\_W;

return gx;

}

inline int CGameMap::getBlockRight(int x) {

int gx = (x / BOX\_W + 1)\*BOX\_W;

return gx;

}

inline bool CGameMap::isBlock(int x, int y) {

int gx = x / BOX\_W;

int gy = y / BOX\_H;

return map\_data[gy][gx] == MAP\_BLOCK;

}

inline bool CGameMap::isEmpty(int x, int y) {

int gx = x / BOX\_W;

int gy = y / BOX\_H;

return map\_data[gy][gx] == MAP\_EMPTY;

}

}

#endif

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

CGameMap01.cpp

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

#pragma once

#include "stdafx.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CGameMap.h"

#include "CGameData.h"

#include "CGameMap01.h"

namespace game\_framework {

CGameMap01::CGameMap01() :

CGameMap() {

}

CGameMap01::CGameMap01(CGameData \*game\_data, int map\_w, int map\_h) :

CGameMap(game\_data, map\_w, map\_h) {

int map\_init[6][8] = {

{ 1,1,1,1,1,1,1,1 },

{ 1,0,0,0,0,0,0,1 },

{ 1,0,0,0,0,0,0,1 },

{ 0,0,0,0,0,0,0,1 },

{ 0,0,0,0,0,1,1,1 },

{ 1,1,1,1,1,1,1,1 }

};

map\_data = new int\*[box\_num\_y];

for (int i = 0; i < box\_num\_y; i++) {

map\_data[i] = new int[box\_num\_x];

for (int j = 0; j < box\_num\_x; j++) {

map\_data[i][j] = map\_init[i][j];

}

}

}

CGameMap01::~CGameMap01() {

}

}

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

CGameMap01.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#ifndef CGAMEMAP01\_H

#define CGAMEMAP01\_H

namespace game\_framework {

class CGameMap01 : public CGameMap {

public:

CGameMap01();

CGameMap01(CGameData\*, int, int);

~CGameMap01();

};

}

#endif

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

CGameMap02.cpp

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

#include "stdafx.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CGameMap.h"

#include "CGameData.h"

#include "CGameMap02.h"

namespace game\_framework {

CGameMap02::CGameMap02():

CGameMap() {

}

CGameMap02::CGameMap02(CGameData \*game\_data,int map\_w,int map\_h) :

CGameMap(game\_data,map\_w,map\_h) {

int map\_init[6][16] = {

{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1 },

{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 }

};

map\_data = new int\*[box\_num\_y];

for (int i = 0; i < box\_num\_y; i++) {

map\_data[i] = new int[box\_num\_x];

for (int j = 0; j < box\_num\_x; j++) {

map\_data[i][j] = map\_init[i][j];

}

}

}

CGameMap02::~CGameMap02() {

}

}

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

CGameMap02.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#ifndef CGAMEMAP02\_H

#define CGAMEMAP02\_H

namespace game\_framework {

class CGameMap02 : public CGameMap {

public:

CGameMap02();

CGameMap02(CGameData\*, int, int);

~CGameMap02();

};

}

#endif

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

CGameMap03.cpp

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

#include "stdafx.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CGameMap.h"

#include "CGameData.h"

#include "CGameMap03.h"

namespace game\_framework {

CGameMap03::CGameMap03():

CGameMap() {

}

CGameMap03::CGameMap03(CGameData \*game\_data,int map\_w,int map\_h) :

CGameMap(game\_data,map\_w,map\_h) {

int map\_init[6][16] = {

{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 }

};

map\_data = new int\*[box\_num\_y];

for (int i = 0; i < box\_num\_y; i++) {

map\_data[i] = new int[box\_num\_x];

for (int j = 0; j < box\_num\_x; j++) {

map\_data[i][j] = map\_init[i][j];

}

}

}

CGameMap03::~CGameMap03() {

}

}

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

CGameMap03.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#ifndef CGAMEMAP03\_H

#define CGAMEMAP03\_H

namespace game\_framework {

class CGameMap03 : public CGameMap {

public:

CGameMap03();

CGameMap03(CGameData\*, int, int);

~CGameMap03();

};

}

#endif

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

CGameMap04.cpp

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

#include "stdafx.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CGameMap.h"

#include "CGameData.h"

#include "CGameMap04.h"

namespace game\_framework {

CGameMap04::CGameMap04() :

CGameMap() {

}

CGameMap04::CGameMap04(CGameData \*game\_data, int map\_w, int map\_h) :

CGameMap(game\_data, map\_w, map\_h) {

int map\_init[6][16] = {

{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },

{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 }

};

map\_data = new int\*[box\_num\_y];

for (int i = 0; i < box\_num\_y; i++) {

map\_data[i] = new int[box\_num\_x];

for (int j = 0; j < box\_num\_x; j++) {

map\_data[i][j] = map\_init[i][j];

}

}

}

CGameMap04::~CGameMap04() {

}

}

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

CGameMap04.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#ifndef CGAMEMAP04\_H

#define CGAMEMAP04\_H

namespace game\_framework {

class CGameMap04 : public CGameMap {

public:

CGameMap04();

CGameMap04(CGameData\*, int, int);

~CGameMap04();

};

}

#endif

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

CGameMap05.cpp

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

#include "stdafx.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "CGameMap.h"

#include "CGameData.h"

#include "CGameMap05.h"

namespace game\_framework {

CGameMap05::CGameMap05() :

CGameMap() {

}

CGameMap05::CGameMap05(CGameData \*game\_data, int map\_w, int map\_h) :

CGameMap(game\_data, map\_w, map\_h) {

int map\_init[6][8] = {

{ 0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0 },

{ 0,0,0,0,0,0,0,0 },

{ 1,1,1,1,1,1,1,1 }

};

map\_data = new int\*[box\_num\_y];

for (int i = 0; i < box\_num\_y; i++) {

map\_data[i] = new int[box\_num\_x];

for (int j = 0; j < box\_num\_x; j++) {

map\_data[i][j] = map\_init[i][j];

}

}

}

CGameMap05::~CGameMap05() {

}

}

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

CGameMap05.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#ifndef CGAMEMAP05\_H

#define CGAMEMAP05\_H

namespace game\_framework {

class CGameMap05 : public CGameMap {

public:

CGameMap05();

CGameMap05(CGameData\*, int, int);

~CGameMap05();

};

}

#endif

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

CGameState01.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CGameState01.h"

namespace game\_framework {

CGameState01::CGameState01(CGame \*g) :

CGameState(g), game\_data(), daughter(&game\_data), map01(&game\_data, 800, 600), outro(false) {

this->count = 0;

}

CGameState01::~CGameState01() {

}

void CGameState01::OnBeginState(int daughterHP) {

blood.setDaughterHP(daughterHP);

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

}

void CGameState01::OnMove() {

daughter.OnMove();

outro = game\_data.getDaughterPos().first < 100;

if (game\_data.getDaughterPos().first == 100)

CAudio::Instance()->Play(OPEN\_TOILET\_DOOR, false);

if (outro) {

transition.OnOutro(&daughter, FACE\_LEFT);

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

if (transition.transit)

GotoGameState(GAME\_STATE02, blood.getDaughterHP());

} else { game\_data.checkMove(&map01, &daughter, nullptr); }

}

void CGameState01::OnInit() {

ShowInitProgress(0);

map01.LoadBitmap(IDB\_MAP01);

daughter.LoadBitmap();

blood.LoadBitmap();

transition.LoadBitmap();

dialog.LoadBitmap(1);

game\_data.setDaughterTopLeft(400, 300);

game\_data.setMapTopLeft(0, 0);

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

CAudio::Instance()->Load(OPEN\_TOILET\_DOOR, "sounds\\open\_toilet\_door.mp3");

ShowInitProgress(20);

}

void CGameState01::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {

dialog.OnKeyDown(nChar);

if (dialog.IsFinished())

daughter.OnKeyDown(nChar);

if (nChar == KEY\_ESC)

//GotoGameState(GAME\_STATE\_OVER, blood.getDaughterHP());

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

}

void CGameState01::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {

if (dialog.IsFinished())

daughter.OnKeyUP(nChar);

switch (nChar) {

case KEY\_F2:

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

GotoGameState(GAME\_STATE02, blood.getDaughterHP());

}

}

void CGameState01::OnShow() {

map01.OnShow();

daughter.OnShow();

blood.OnShow(false);

transition.OnShow();

if (!outro) {

dialog.OnShow();

}

}

}

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

CGameState01.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#include "CGameMap01.h"

#include "CDaughter.h"

#include "CGameData.h"

#include "CBlood.h"

#include "CTransition.h"

#include "CDialog.h"

#ifndef CGAMESTATE01\_H

#define CGAMESTATE01\_H

namespace game\_framework {

class CGameState01 : public CGameState {

public:

CGameState01(CGame \*g);

~CGameState01();

void OnBeginState(int daughterHP);

void OnInit();

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

protected:

void OnMove();

void OnShow();

private:

CGameMap01 map01;

CDaughter daughter;

CGameData game\_data;

CBlood blood;

CTransition transition;

CDialog dialog;

int count;

bool outro;

};

}

#endif

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

CGameState02.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CGameState02.h"

namespace game\_framework {

CGameState02::CGameState02(CGame \*g) :

CGameState(g), game\_data(), daughter(&game\_data), map02(&game\_data, 1600, 600),

intro(true), outro(false){

}

CGameState02::~CGameState02() {

}

void CGameState02::OnBeginState(int daughterHP) {

blood.setDaughterHP(daughterHP);

}

void CGameState02::OnMove() {

intro = game\_data.getDaughterPos().first < 0;

outro = game\_data.getDaughterPos().first > 640;

daughter.OnMove();

if (game\_data.getDaughterPos().first == -35)

CAudio::Instance()->Play(CLOSE\_DOOR, false);

if (outro) {

transition.OnOutro(&daughter, FACE\_RIGHT);

if (transition.transit)

GotoGameState(GAME\_STATE03, blood.getDaughterHP());

}

else if (!outro && !intro) {

game\_data.checkMove(&map02, &daughter,nullptr);

}

transition.OnIntro(&game\_data, &daughter, FACE\_RIGHT, 0);

}

void CGameState02::OnInit(){

ShowInitProgress(20);

map02.LoadBitmap(IDB\_MAP02);

daughter.LoadBitmap();

blood.LoadBitmap();

transition.LoadBitmap();

dialog.LoadBitmap(2);

game\_data.setDaughterTopLeft(-100, 300);

game\_data.setMapTopLeft(0, 0);

CAudio::Instance()->Load(CLOSE\_DOOR, "sounds\\close\_door.wav");

ShowInitProgress(40);

}

void CGameState02::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {

dialog.OnKeyDown(nChar);

if (dialog.IsFinished())

daughter.OnKeyDown(nChar);

if (nChar == KEY\_ESC)

//GotoGameState(GAME\_STATE\_OVER, blood.getDaughterHP());

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

}

void CGameState02::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {

if (dialog.IsFinished())

daughter.OnKeyUP(nChar);

switch (nChar) {

case KEY\_F2:

GotoGameState(GAME\_STATE03, blood.getDaughterHP());

}

}

void CGameState02::OnShow() {

map02.OnShow();

daughter.OnShow();

blood.OnShow(false);

transition.OnShow();

if (!intro && !outro) {

dialog.OnShow();

}

}

}

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

CGameState02.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#include "CGameMap02.h"

#include "CDaughter.h"

#include "CGameData.h"

#include "CBlood.h"

#include "CTransition.h"

#include "CDialog.h"

#ifndef CGAMESTATE02\_H

#define CGAMESTATE02\_H

namespace game\_framework {

class CGameState02 : public CGameState {

public:

CGameState02(CGame \*g);

~CGameState02();

void OnBeginState(int daughterHP);

void OnInit();

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

protected:

void OnMove();

void OnShow();

private:

CGameMap02 map02;

CDaughter daughter;

CGameData game\_data;

CBlood blood;

CTransition transition;

CDialog dialog;

bool intro;

bool outro;

};

}

#endif

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

CGameState03.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CGameState03.h"

namespace game\_framework {

CGameState03::CGameState03(CGame \*g) :

CGameState(g), game\_data(), daughter(&game\_data), map03(&game\_data, 1600, 600),

intro(true), outro(false) {

}

CGameState03::~CGameState03() {

}

void CGameState03::OnBeginState(int daughterHP){

CAudio::Instance()->Play(WIND\_SOUND, true);

blood.setDaughterHP(daughterHP);

}

void CGameState03::OnMove() {

intro = game\_data.getDaughterPos().first > 700;

outro = game\_data.getDaughterPos().first < 100;

daughter.OnMove();

if (outro) {

transition.OnOutro(&daughter, FACE\_LEFT);

CAudio::Instance()->Play(IRON\_DOOR, false);

CAudio::Instance()->Stop(WIND\_SOUND);

if (transition.transit)

GotoGameState(GAME\_STATE04, blood.getDaughterHP());

}

else if (!outro && !intro) {

game\_data.checkMove(&map03, &daughter,nullptr);

}

transition.OnIntro(&game\_data, &daughter, FACE\_LEFT, 700);

}

void CGameState03::OnInit(){

ShowInitProgress(40);

map03.LoadBitmap(IDB\_MAP03);

daughter.LoadBitmap();

blood.LoadBitmap();

transition.LoadBitmap();

dialog.LoadBitmap(3);

game\_data.setDaughterTopLeft(800, 300);

game\_data.setMapTopLeft(800, 0);

CAudio::Instance()->Load(IRON\_DOOR, "sounds\\iron\_door.mp3");

CAudio::Instance()->Load(WIND\_SOUND, "sounds\\strong\_wind\_sound\_effect.mp3");

ShowInitProgress(60);

}

void CGameState03::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {

dialog.OnKeyDown(nChar);

if (dialog.IsFinished())

daughter.OnKeyDown(nChar);

if (nChar == KEY\_ESC)

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

}

void CGameState03::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {

if (dialog.IsFinished())

daughter.OnKeyUP(nChar);

switch (nChar) {

case KEY\_F2:

GotoGameState(GAME\_STATE04, blood.getDaughterHP());

}

}

void CGameState03::OnShow() {

map03.OnShow();

daughter.OnShow();

blood.OnShow(false);

transition.OnShow();

if (!intro && !outro) {

dialog.OnShow();

}

}

}

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

CGameState03.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#include "CGameMap03.h"

#include "CDaughter.h"

#include "CGameData.h"

#include "CBlood.h"

#include "CTransition.h"

#include "CDialog.h"

#ifndef CGAMESTATE03\_H

#define CGAMESTATE03\_H

namespace game\_framework {

class CGameState03 : public CGameState {

public:

CGameState03(CGame \*g);

~CGameState03();

void OnBeginState(int daughterHP);

void OnInit();

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

protected:

void OnMove();

void OnShow();

private:

CGameMap03 map03;

CDaughter daughter;

CGameData game\_data;

CBlood blood;

CTransition transition;

CDialog dialog;

bool intro;

bool outro;

};

}

#endif

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

CGameState04.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CGameState04.h"

namespace game\_framework {

CGameState04::CGameState04(CGame \*g) :

CGameState(g), game\_data(), daughter(&game\_data), father(&game\_data, &blood), map04(&game\_data, 1600, 600),

intro(true), outro(false) {

}

CGameState04::~CGameState04() {

}

void CGameState04::OnBeginState(int daughterHP) {

CAudio::Instance()->Stop(WIND\_SOUND);

CAudio::Instance()->Play(WHISPER\_SOUND, true);

blood.setDaughterHP(daughterHP);

}

void CGameState04::OnMove() {

intro = game\_data.getDaughterPos().first < 0;

outro = game\_data.getDaughterPos().first > 640;

range = game\_data.getMapPos().first >= 300 && game\_data.getMapPos().first < 800;

daughter.OnMove();

if (outro) {

transition.OnOutro(&daughter, FACE\_RIGHT);

CAudio::Instance()->Play(ROOM\_DOOR, false);

CAudio::Instance()->Stop(WHISPER\_SOUND);

if (transition.transit) {

CAudio::Instance()->Stop(DAUGHTER\_FOOTSTEP);

GotoGameState(GAME\_STATE05, blood.getDaughterHP());

}

}

else if (!outro && !intro) {

game\_data.checkMove(&map04, &daughter,nullptr);

}

if (range) {

game\_data.setMapMove(5, 0);

game\_data.setDaughterMove(-5, 0);

game\_data.setFatherMove(-5, 0);

}

transition.OnIntro(&game\_data, &daughter, FACE\_RIGHT, 0);

}

void CGameState04::OnInit(){

ShowInitProgress(60);

map04.LoadBitmap(IDB\_MAP04);

daughter.LoadBitmap();

father.LoadBitmap();

blood.LoadBitmap();

transition.LoadBitmap();

dialog.LoadBitmap(4);

locked\_door.LoadBitmap(IDB\_DOOR\_LOCK, RGB(0, 0, 255));

locked\_door.SetTopLeft(370, 0);

game\_data.setDaughterTopLeft(-100, 300);

game\_data.setFatherTopLeft(930, 270);

game\_data.setMapTopLeft(0, 0);

CAudio::Instance()->Load(ROOM\_DOOR, "sounds\\room\_door.mp3");

CAudio::Instance()->Load(WHISPER\_SOUND, "sounds\\whisper\_mixdown.mp3");

ShowInitProgress(80);

}

void CGameState04::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {

if(game\_data.getMapPos().first >= 799)

dialog.OnKeyDown(nChar);

if (game\_data.getMapPos().first >= 300 && !dialog.IsFinished()) {

daughter.OnKeyUP(KEY\_RIGHT);

daughter.outro = true;

}

else {

daughter.outro = false;

daughter.OnKeyDown(nChar);

}

if (nChar == KEY\_ESC)

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

}

void CGameState04::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {

if (game\_data.getMapPos().first >= 300 && !dialog.IsFinished()) {

daughter.OnKeyUP(KEY\_RIGHT);

daughter.outro = true;

}

else {

daughter.outro = false;

daughter.OnKeyUP(nChar);

}

switch (nChar) {

case KEY\_F2:

GotoGameState(GAME\_STATE05, blood.getDaughterHP());

}

}

void CGameState04::OnShow() {

map04.OnShow();

daughter.OnShow();

blood.OnShow(false);

transition.OnShow();

if (!dialog.IsFinished()) {

father.OnShow();

}

if (game\_data.getMapPos().first >= 799) {

dialog.OnShow();

if (game\_data.getDaughterPos().first >= 400 && game\_data.getDaughterPos().first <= 500) {

locked\_door.ShowBitmap();

}

}

}

}

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

CGameState04.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#include "CGameMap04.h"

#include "CDaughter.h"

#include "CFather.h"

#include "CGameData.h"

#include "CBlood.h"

#include "CTransition.h"

#include "CDialog.h"

#ifndef CGAMESTATE04\_H

#define CGAMESTATE04\_H

namespace game\_framework {

class CGameState04 : public CGameState {

public:

CGameState04(CGame \*g);

~CGameState04();

void OnBeginState(int daughterHP);

void OnInit();

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

protected:

void OnMove();

void OnShow();

private:

CGameMap04 map04;

CDaughter daughter;

CFather father;

CGameData game\_data;

CBlood blood;

CTransition transition;

CDialog dialog;

CMovingBitmap locked\_door;

bool intro;

bool outro;

bool range;

};

}

#endif

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

CGameState05.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CGameState05.h"

namespace game\_framework {

CGameState05::CGameState05(CGame \*g) :

CGameState(g), game\_data(), blood(), daughter(&game\_data), map05(&game\_data, 800, 600), father(&game\_data,&blood)

,bossAI(&father, &game\_data, &container,&blood){

this->counter = 0;

this->firstStart = false;

}

CGameState05::~CGameState05() {

}

void CGameState05::OnBeginState(int daughterHP) {

CAudio::Instance()->Stop(WHISPER\_SOUND);

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

blood.setDaughterHP(daughterHP);

}

void CGameState05::OnMove() {

if (!dialog.IsFinished() || blood.daughterIsDied() || blood.fatherIsDied()) {

counter = 0;

bossAI.setAILock(true);

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

}

else {

bossAI.setAILock(false);

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

}

if (counter < 25 && blood.getFatherHP()>450 && !blood.fatherIsDied() ) {

blood.setFatherHP(690-(10\*counter));

}

else {

counter = 25;

}

intro = game\_data.getDaughterPos().first < 0;

daughter.OnMove();

container.OnMove();

father.OnMove();

bossAI.OnLoop();

container.damageTake(&blood, &daughter, game\_data.getDaughterPos());

if (!intro)

game\_data.checkMove(&map05, &daughter,&father);

if (dialog\_fail.IsFinished() || dialog\_success.IsFinished()) {

GotoGameState(GAME\_STATE\_OVER, blood.getDaughterHP());

}

transition.OnIntro(&game\_data, &daughter, FACE\_RIGHT, 0);

counter++;

}

void CGameState05::OnInit() {

ShowInitProgress(80);

map05.LoadBitmap(IDB\_MAP05);

daughter.LoadBitmap();

blood.LoadBitmap();

father.LoadBitmap();

transition.LoadBitmap();

dialog.LoadBitmap(5);

dialog\_fail.LoadBitmap(6);

dialog\_success.LoadBitmap(7);

blood.setFatherHP(690);

game\_data.setDaughterTopLeft(-100, 300);

game\_data.setFatherTopLeft(700, 270);

game\_data.setMapTopLeft(0, 0);

CAudio::Instance()->Load(BOSS\_BGM, "sounds\\boss\_bgm.mp3");

ShowInitProgress(100);

}

void CGameState05::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {

if (dialog.IsFinished() && !blood.daughterIsDied() && !blood.fatherIsDied()) {

daughter.OnKeyDown(nChar);

}

dialog.OnKeyDown(nChar);

if (blood.daughterIsDied()) {

dialog\_fail.OnKeyDown(nChar);

}

else if (blood.fatherIsDied()) {

dialog\_success.OnKeyDown(nChar);

}

if (nChar == KEY\_ESC)

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

}

void CGameState05::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {

if (dialog.IsFinished() && !blood.daughterIsDied() && !blood.fatherIsDied()) {

daughter.OnKeyUP(nChar);

}

switch (nChar) {

case KEY\_F3:

blood.setDaughterHP(102);

break;

case KEY\_F4:

blood.setDaughterHP(-141);

break;

case KEY\_F5:

blood.setFatherHP(560);

break;

case KEY\_F6:

blood.setFatherHP(700);

break;

}

}

void CGameState05::OnShow() {

map05.OnShow();

daughter.OnShow();

container.OnShow();

father.OnShow();

if (!dialog.IsFinished()) {

blood.OnShow(false);

}

else {

blood.OnShow(true);

}

transition.OnShow();

if (!intro) {

dialog.OnShow();

}

if (blood.daughterIsDied()) {

dialog\_fail.OnShow();

}

else if(blood.fatherIsDied()) {

dialog\_success.OnShow();

}

}

}

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

CGameState05.h

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

#pragma once

#include "Definitions.h"

#include "CGameMap.h"

#include "CGameMap05.h"

#include "CDaughter.h"

#include "CFather.h"

#include "CBossAI.h"

#include "CGameData.h"

#include "CBlood.h"

#include "CTransition.h"

#include "CDialog.h"

#include "CDamageContainer.h"

#include "CDamagePen.h"

#ifndef CGAMESTATE05\_H

#define CGAMESTATE05\_H

namespace game\_framework {

class CGameState05 : public CGameState {

public:

CGameState05(CGame \*g);

~CGameState05();

void OnBeginState(int daughterHP);

void OnInit();

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

protected:

void OnMove();

void OnShow();

private:

CGameMap05 map05;

CFather father;

CBossAI bossAI;

CDaughter daughter;

CGameData game\_data;

CBlood blood;

CTransition transition;

CDialog dialog;

CDialog dialog\_fail;

CDialog dialog\_success;

CDamageContainer container;

int counter;

bool intro;

bool firstStart;

};

}

#endif

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

CTransition.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "gamelib.h"

#include "audio.h"

#include "CTransition.h"

#include "CGameData.h"

#include "CDaughter.h"

namespace game\_framework {

CTransition::CTransition() {}

CTransition::~CTransition(){}

void CTransition::LoadBitmap() {

outro\_up.LoadBitmap(IDB\_OUTRO\_UP);

outro\_down.LoadBitmap(IDB\_OUTRO\_DOWN);

}

void CTransition::OnIntro(CGameData \*game\_data, CDaughter \*daughter, int faceSide, int stop) {

if (faceSide == FACE\_RIGHT) {

if (game\_data->getDaughterPos().first < stop) {

daughter->OnKeyDown(KEY\_RIGHT); //進場走進來(右)

daughter->outro = true;

}

if (game\_data->getDaughterPos().first == stop) {

daughter->outro = false;

daughter->OnKeyUP(KEY\_RIGHT); //進場停止(右)

}

}

else if (faceSide == FACE\_LEFT) {

if (game\_data->getDaughterPos().first > stop) {

daughter->OnKeyDown(KEY\_LEFT); //進場走進來(左)

daughter->outro = true;

}

if (game\_data->getDaughterPos().first == stop) {

daughter->outro = false;

daughter->OnKeyUP(KEY\_LEFT); //進場停止(左)

}

}

}

void CTransition::OnOutro(CDaughter \*daughter, int faceSide) {

if (y < 100) {

y += 5; //出場橫幅動畫

}

if (y == 100) {

transit = true; //接換遊戲章節

}

if (faceSide == FACE\_RIGHT) {

daughter->OnKeyDown(KEY\_RIGHT); //出場走出去(右)

daughter->outro = true;

}

else if (faceSide == FACE\_LEFT) {

daughter->OnKeyDown(KEY\_LEFT); //出場走出去(左)

daughter->outro = true;

}

}

void CTransition::OnShow() {

outro\_up.SetTopLeft(0, -100 + y);

outro\_down.SetTopLeft(0, 600 - y);

outro\_up.ShowBitmap();

outro\_down.ShowBitmap();

}

}

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

CTransition.h

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

#pragma once

#include "Definitions.h"

#ifndef CTRANSITION\_H

#define CTRANSITION\_H

namespace game\_framework {

class CTransition {

public:

CTransition();

~CTransition();

void LoadBitmap();

void OnIntro(CGameData \*, CDaughter \*, int, int); //進場動畫

void OnOutro(CDaughter \*, int); //出場動畫

void OnShow();

bool transit = false;

private:

int y = 0; //下降幅度

CMovingBitmap outro\_up; //出場橫幅

CMovingBitmap outro\_down; //出場橫幅

};

}

#endif

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

Definitions.h

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

#pragma once

#ifndef DEFINITIONS\_H

#define DEFINITIONS\_H

namespace game\_framework {

class CGameMap;

class CGameData;

class CDaughter;

class CFather;

class CDamageContainer;

class CBlood;

class CTransition;

class CDialog;

}

#endif

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

mygame.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <windows.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

namespace game\_framework {

CGameStateInit::CGameStateInit(CGame \*g)

: CGameState(g){

}

void CGameStateInit::OnInit(){

ShowInitProgress(0);

menu.LoadBitmap(IDB\_MENU);

func.LoadBitmap(IDB\_FUNC);

dot.LoadBitmap(IDB\_DOT);

secret.LoadBitmap(IDB\_SECRET);

secret.SetTopLeft(0, 390);

dot.SetTopLeft(270, 400);

menu.SetTopLeft(0, 0);

func.SetTopLeft(0, 0);

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

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

CAudio::Instance()->Load(DAUGHTER\_FOOTSTEP, "sounds\\footstep.wav");

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

CAudio::Instance()->Load(DAUGHTER\_SLIDE, "sounds\\slide.wav");

CAudio::Instance()->Load(DAUGHTER\_HURT, "sounds\\hurt.mp3");

CAudio::Instance()->Load(FATHER\_HURT, "sounds\\hurt\_father.mp3");

CAudio::Instance()->Load(EXPLO\_SOUND, "sounds\\explosion.mp3");

CAudio::Instance()->Load(SEA\_WAVE, "sounds\\sea\_wave.mp3");

CAudio::Instance()->Load(G\_HIT, "sounds\\ground\_hit.mp3");

}

void CGameStateInit::OnBeginState(){

}

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

nC = nChar;

if (dot.Top() <= 450 && nChar == KEY\_DOWN) {

dot.SetTopLeft(270, dot.Top() + 50);

}

if(dot.Top() >= 450 && nChar == KEY\_UP) {

dot.SetTopLeft(270, dot.Top() - 50);

}

if (dot.Top() == 400 && nChar == KEY\_ENTER) {

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

GotoGameState(GAME\_STATE01, 102);

}

if (dot.Top() == 500 && nChar == KEY\_ENTER) {

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

}

if (nChar == KEY\_ESC) {

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

}

}

void CGameStateInit::OnShow(){

menu.ShowBitmap();

dot.ShowBitmap();

if (dot.Top() == 450 && nC == KEY\_ENTER) {

func.ShowBitmap();

}

if (nC == KEY\_LEFT) {

secret.ShowBitmap();

}

}

void CGameStateInit::OnMove() {

}

CGameStateOver::CGameStateOver(CGame \*g)

: CGameState(g)

{

game = g;

}

void CGameStateOver::OnMove(){

counter--;

if (counter < 0) {

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

}

}

void CGameStateOver::OnBeginState(int daughterHP)

{

counter = 30 \* 3;

}

void CGameStateOver::OnInit()

{

ShowInitProgress(66);

over.LoadBitmap(IDB\_OVER);

over.SetTopLeft(0, 0);

Sleep(300);

ShowInitProgress(100);

}

void CGameStateOver::OnShow()

{

over.ShowBitmap();

}

}

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

mygame.h

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

#include "Definitions.h"

#include "CGameMap.h"

#include "CGameMap02.h"

#include "CGameMap01.h"

#include "CDaughter.h"

#include "CGameData.h"

#include "CGameState01.h"

#include "CGameState02.h"

#include "CGameState03.h"

#include "CGameState04.h"

#include "CGameState05.h"

namespace game\_framework {

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

// 這個class為遊戲的遊戲開頭畫面物件

// 每個Member function的Implementation都要弄懂

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

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

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

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

void OnMove();

private:

UINT nC;

CPoint p;

CMovingBitmap menu;

CMovingBitmap func;

CMovingBitmap dot;

CMovingBitmap secret;

};

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

// 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(int); // 設定每次重玩所需的變數

void OnInit();

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int counter; // 倒數之計數器

CMovingBitmap over;

};

}