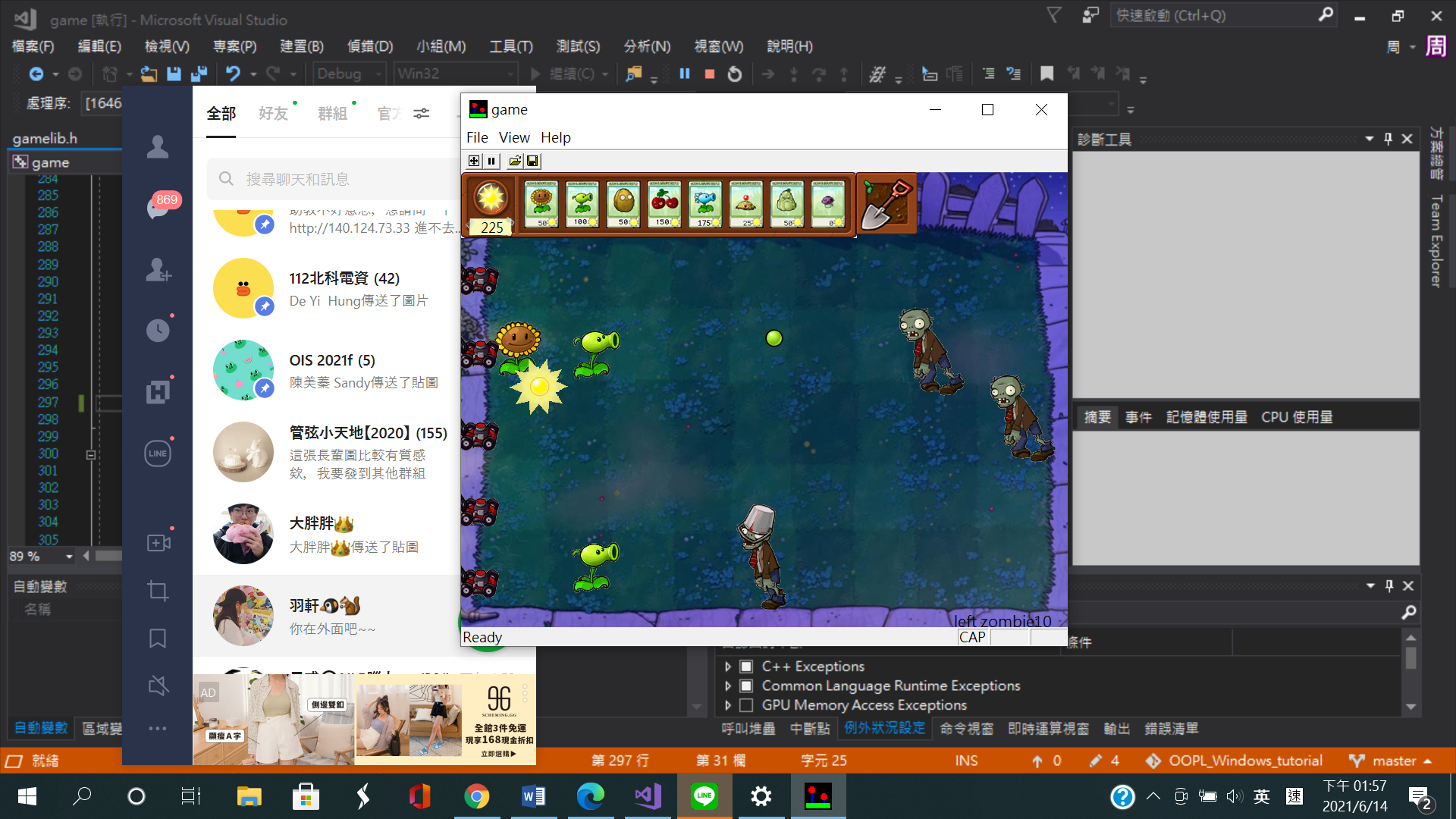
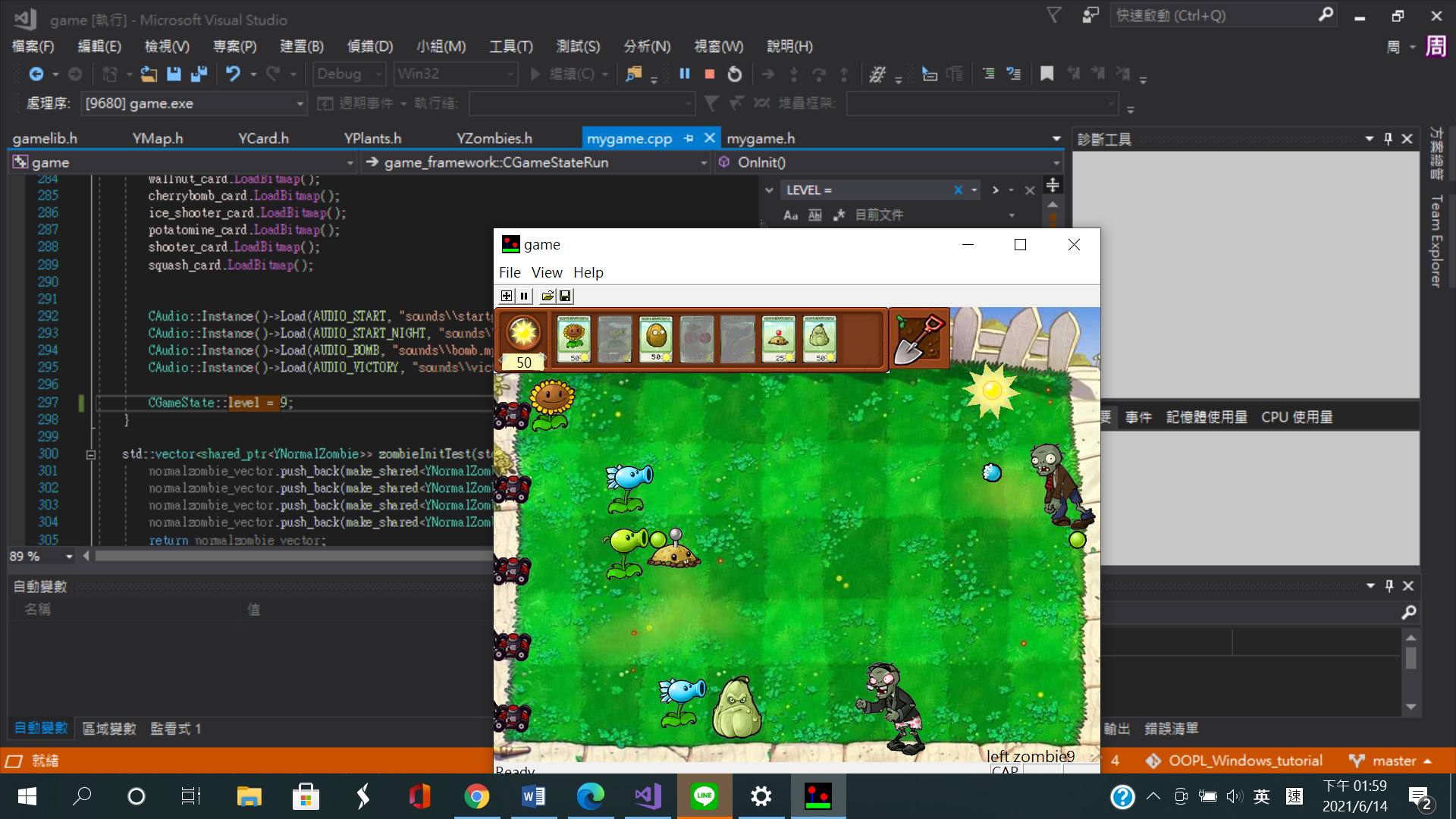
國立臺北科技大學

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 |
| C:\Users\zoezo\OneDrive\文件\OOPL.jpg |
| Others |
|  |

1. 程式類別：

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| YNormalZombie | 510 | 0 | 各式殭屍 |
| YMap | 651 | 0 | 地圖 |
| YCar | 69 | 0 | 車子 |
| YSunFlowerCard | 50 | 0 | 種植植物卡片 |
| YPeaShooterCard | 48 | 0 | 種植植物卡片 |
| YWallNutCard | 49 | 0 | 種植植物卡片 |
| YCherryBombCard | 48 | 0 | 種植植物卡片 |
| YIceShooterCard | 48 | 0 | 種植植物卡片 |
| YPotatomineCard | 48 | 0 | 種植植物卡片 |
| YSquashCard | 48 | 0 | 種植植物卡片 |
| YShooterCard | 48 | 0 | 種植植物卡片 |
| YShovelCard | 48 | 0 | 移除植物卡片 |
| YSun | 36 | 126 | 太陽 |
| YSunFlower | 101 | 0 | 向日葵 |
| YPeaShooterBullet | 51 | 0 | 豌豆射手子彈 |
| YPeaShooter | 121 | 0 | 豌豆射手 |
| YWallNut | 105 | 0 | 牆果 |
| YCherryBomb | 85 | 0 | 櫻桃炸彈 |
| YIceShooterBullet | 51 | 0 | 雪花豌豆子彈 |
| YIceShooter | 116 | 0 | 雪花豌豆 |
| YPotatoMine | 137 | 0 | 土豆地雷 |
| YSquash | 121 | 0 | 窩瓜 |
| YShooterBullet | 51 | 0 | 噴射蘑菇子彈 |
| YShooter | 132 | 0 | 噴射蘑菇 |
| mygame | 1300 | 30 | State\_init,state\_over |
| **總行數** | **4072** | **156** | **4228** |

1. 程式技術：

大多的技術都集中在plants裡面，每個植物都有自己各自的vector，在每次點擊卡片的同時我們才會創建新的物件，使用vector的存放物件和清除都是非常好用的，再來就是搭配shared\_ptr的使用讓物件的生命週期更加穩定，讓程式幫忙做記憶體得配置與回收才不容易讓程式出現不可預期的錯誤。

再來是Map我們使用矩陣存地圖，因為地圖的格式是不變的，所以我們使用最簡單的方法做定位，但也非常方便。

最後是static變數在各個state 之間傳遞，static變數的宣告方法與生命週期也是程式裡面很重要的部分。其他物件導向很重要的物件之間的互動也是在這堂課中學到的東西。

1. **結語**
2. 問題及解決方法：

|  |  |
| --- | --- |
| 問題 | 解決方法 |
| 動畫vector的錯誤 | Static的使用方法錯誤，改為使用區域變數紀錄 |
| 動畫播放問題 | 改用shared\_ptr在vector解決 |
| 全螢幕無法實行 | 將遊戲畫面x座標平移，以符合比例要求。 |
| 動畫load太多玩到後面整個程式卡死 | 優化程式，最低限度減少動畫 |

1. 時間表：

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 羅羽軒(小時) | 周雨柔(小時) | 說明 |
| 1 | 2 | 2 | 選題、分組、練習 |
| 2 | 3 | 2.5 | git練習 |
| 3 | 10 | 6 | tutorial練習、開會 |
| 4 | 12 | 6 | 素材、init畫面、主畫面、音效、場景移動 |
| 5 | 12 | 6.5 | Sun class，落下、點擊，chooser |
| 6 | 11 | 6 | Sunflower class 種植、卡片 |
| 7 | 12 | 4 | YSunflower動畫、YPeashooter class、冷卻時間、map、zombie |
| 8 | 12 | 5 | YMap class、殭屍動畫、攻擊 |
| 9 | 9 | 3 | 殭屍模式、peashooter bullet |
| 10 | 12 | 6 | 子彈攻擊、car、殭屍死亡 |
| 11 | 11 | 5 | 子彈、車攻擊殭屍、鏟子、改用shared\_ptr |
| 12 | 12 | 5 | 殭屍攻擊植物、修正動畫錯誤、SunFlower產生Sun、car、殭屍換關動畫、失敗動畫 |
| 13 | 9 | 7 | Level 2 3、Wallnut、cherrybomb  、flag zombie、conehead zombie |
| 14 | 8 | 5 | Level 4 5、Night mode、加速、太陽密技、clean code |
| 15 | 9 | 7 | Level 6 7 8、Iceshooter、potatomine、bucket zombie |
| 16 | 11 | 11 | Level 9 10、音效、shroom、squash換關背景更新、換關錯誤更改、newspaper zombie、 |
| 17 | 15 | 12 | squash、全螢幕、clear code、zombie all die、level++密技、night, bomb , victory sound、從頭到尾試玩、動畫錯誤修改 |
| **合計** | **170** | **99** |  |

1. 貢獻比例：

羅羽軒：60%、周雨柔：40%

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：

羅羽軒：

遇到錯誤時可以設中斷點一步一步去找可以通過編譯的錯誤，遇到會讓整個遊戲停下的錯誤可藉由stack看最後叫了那些函式。物件互動、遊戲框架、shared\_ptr使用、實踐遊戲流程，try and error……。

周雨柔：

觀察範例遊戲的框架，理解框架中各函式的用法，學習整理與編輯遊戲素材，讀取多張圖片到程式中製造動畫效果，練習看報錯訊息，在多個檔案間查看相關函式定義與推測出錯的地方。

1. 心得、感想：

羅羽軒：

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，還要收集素材、錄製音效等等，到最後寫出整個遊戲時還是會想再增進UI、讓操作更人性化畫增加不同音效來豐富遊戲，一步步地完成這個遊戲並持續美化讓我很有成就感。

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，在遇到困難的時候，都會指導我們所遇到的問題該如何解決比較好，也感謝老師在我們遇到一個讓我們程式有奇怪動畫錯誤出現的時候幫助我們非常多，那一次static的錯誤也讓我了解了非常多底層的概念。

周雨柔：

本課程中讓我複習了以前物件導向課程的概念來製作一個遊戲，也培養我耐心去觀察原始遊戲的詳細規則。雖然這堂課難度跟時間成本相較其他課高，但獲得的經驗也真的很不一樣，最後有一個這樣的成品蠻有成就感的。感謝學校安排這堂實習課，讓我們可以實際運用以前課程中的概念，去完成一個自己的作品。感謝老師和助教在每週回報進度的時候，耐心替我們講解，從何處下手調整遊戲有誤的地方。最後也感謝同學，常常替我解答疑惑，帶著我完成了這個可愛的遊戲！

1. 對於本課程的建議：

希望期末報告能早一點給範本，一周周做並記錄才不會最後才和所有事情撞在一起、框架需要修改、全螢幕的要求及條件可以提早說明。

附錄

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

mygame.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <stdlib.h> /\* 亂數相關函數 \*/

#include <time.h> /\* 時間相關函數 \*/

#include <string> // to\_string

#include <sstream>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

#include "YMap.h"

namespace game\_framework {

int CGameState::victoryflag = 0;

int CGameState::level = 1;

bool CGameState::night\_mode = false;

bool CGameState::all\_victory\_flag = false;

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

// 這個class為遊戲的遊戲開頭畫面物件

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

CGameStateInit::CGameStateInit(CGame \*g)

: CGameState(g)

{

}

void CGameStateInit::OnInit()

{

ShowInitProgress(0);

mainmenu.LoadBitmap("Bitmaps/MainMenu.bmp");

adventure0.LoadBitmap("Bitmaps/Adventure0.bmp", RGB(255, 255, 255));

all\_level\_done.LoadBitmap("Bitmaps/vic\_done.bmp", RGB(0, 0, 0));

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

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

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

CAudio::Instance()->Load(AUDIO\_SUNPICK, "sounds\\sun\_pick.mp3");

CAudio::Instance()->Load(AUDIO\_PLANTS, "sounds\\plants\_sound.mp3");

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

}

void CGameStateInit::OnBeginState()

{

}

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

{

const char KEY\_ESC = 27;

const char KEY\_L = 76;

//const char KEY\_SPACE = ' ';

//if (nChar == KEY\_SPACE)

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

if (nChar == KEY\_ESC)

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

if (nChar == KEY\_L) {

level += 1;

if (level == 11) {

level = 1;

}

}

}

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

{

if (point.x > 418 && point.y > 100 && point.x < 702 && point.y < 260) {

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

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

}

}

void CGameStateInit::OnShow()

{

mainmenu.SetTopLeft(0, 0);

mainmenu.ShowBitmap();

adventure0.SetTopLeft(418, 100);

adventure0.ShowBitmap();

if (all\_victory\_flag) {

all\_level\_done.SetTopLeft(45, 150);

all\_level\_done.ShowBitmap();

}

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

CFont f, \*fp;

f.CreatePointFont(130, "微軟正黑體");

fp = pDC->SelectObject(&f);

pDC->SetBkMode(TRANSPARENT);

pDC->SetTextColor(RGB(174, 225, 174));

if (level == 1)

pDC->TextOut(710, 10, "level: 1");

else if (level == 2)

pDC->TextOut(710, 10, "level: 2");

else if (level == 3)

pDC->TextOut(710, 10, "level: 3");

else if (level == 4)

pDC->TextOut(710, 10, "level: 4");

else if (level == 5)

pDC->TextOut(710, 10, "level: 5");

else if (level == 6)

pDC->TextOut(710, 10, "level: 6");

else if (level == 7)

pDC->TextOut(710, 10, "level: 7");

else if (level == 8)

pDC->TextOut(710, 10, "level: 8");

else if (level == 9)

pDC->TextOut(710, 10, "level: 9");

else if (level == 10)

pDC->TextOut(710, 10, "level: 10");

pDC->SetTextColor(RGB(204, 255, 204));

pDC->TextOut(440, 260, "點擊 \"冒險模式\" 開始 !");

pDC->SetTextColor(RGB(197, 229, 210));

pDC->TextOut(470, 295, "L: level += 1");

pDC->TextOut(470, 325, "S: sun = 500");

pDC->TextOut(465, 355, "Z: zombie fast");

pDC->TextOut(460, 385, "D: zombie all die");

pDC->SetTextColor(RGB(185, 229, 255));

pDC->TextOut(5, 516, "Use mouse to plant your plants and beat all zombies!");

pDC->SetTextColor(RGB(255, 255, 0));

pDC->TextOut(5, 543, "Ctrl-Q: pause the Game. Alt-F4 or ESC: Quit.");

pDC->TextOut(5, 570, "Ctrl-F: switch between windows / full screen mode.");

pDC->SelectObject(fp);

CDDraw::ReleaseBackCDC();

}

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

// 這個class為遊戲的結束狀態(Game Over)

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

CGameStateOver::CGameStateOver(CGame \*g)

: CGameState(g)

{

}

void CGameStateOver::OnMove()

{

counter--;

if (!victoryflag && counter < 0)

GotoGameState(GAME\_STATE\_INIT);

}

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

{

if (victoryflag && point.x > 363 && point.y > 501 && point.x < 538 && point.y < 545) {

if (CGameState::level == 11) {

CGameState::level = 1;

all\_victory\_flag = true;

}

GotoGameState(GAME\_STATE\_INIT);

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

}

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 5; // 5 seconds

}

void CGameStateOver::OnInit()

{

//

// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人

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

//

loose.LoadBitmap("Bitmaps/GameLoose.bmp");

loose\_night.LoadBitmap("Bitmaps/GameLoose\_night.bmp");

newplant.LoadBitmap("Bitmaps/almanac/newplant.bmp");

bucket.LoadBitmap("Bitmaps/almanac/bucket.bmp");

cherrybomb.LoadBitmap("Bitmaps/almanac/cherrybomb.bmp");

conehead.LoadBitmap("Bitmaps/almanac/conehead.bmp");

flag.LoadBitmap("Bitmaps/almanac/flag.bmp");

potatomine.LoadBitmap("Bitmaps/almanac/potatomine.bmp");

puff\_shroom.LoadBitmap("Bitmaps/almanac/puff\_shroom.bmp");

snowpea.LoadBitmap("Bitmaps/almanac/snowpea.bmp");

wallnut.LoadBitmap("Bitmaps/almanac/wallnut.bmp");

squash.LoadBitmap("Bitmaps/almanac/squash.bmp");

newspaper.LoadBitmap("Bitmaps/almanac/newspaper.bmp");

victory.LoadBitmap("Bitmaps/almanac/victory.bmp");

ShowInitProgress(66);

ShowInitProgress(85);

ShowInitProgress(100);

}

void CGameStateOver::OnShow()

{

if (victoryflag) {

newplant.SetTopLeft(0, 0);

newplant.ShowBitmap();

if (CGameState::level == 2) {

wallnut.SetTopLeft(232, 91);

wallnut.ShowBitmap();

}

else if (CGameState::level == 3) {

cherrybomb.SetTopLeft(232, 91);

cherrybomb.ShowBitmap();

}

else if (CGameState::level == 4) {

conehead.SetTopLeft(232, 91);

conehead.ShowBitmap();

}

else if (CGameState::level == 5) {

snowpea.SetTopLeft(232, 91);

snowpea.ShowBitmap();

}

else if (CGameState::level == 6) {

bucket.SetTopLeft(232, 91);

bucket.ShowBitmap();

}

else if (CGameState::level == 7) {

potatomine.SetTopLeft(232, 91);

potatomine.ShowBitmap();

}

else if (CGameState::level == 8) {

newspaper.SetTopLeft(232, 91);

newspaper.ShowBitmap();

}

else if (CGameState::level == 9) {

squash.SetTopLeft(0, 0);

squash.ShowBitmap();

}

else if (CGameState::level == 10) {

puff\_shroom.SetTopLeft(232, 91);

puff\_shroom.ShowBitmap();

}

else if (CGameState::level == 11) {

victory.SetTopLeft(232, 91);

victory.ShowBitmap();

}

}

else {

if (night\_mode) {

loose\_night.SetTopLeft(0, 0);

loose\_night.ShowBitmap();

}

else {

loose.SetTopLeft(0, 0);

loose.ShowBitmap();

}

}

}

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

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

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

CGameStateRun::CGameStateRun(CGame \*g)

: CGameState(g), NUMBALLS(28)

{

picX = picY = 0;

}

CGameStateRun::~CGameStateRun()

{

}

void CGameStateRun::OnInit()

{

ShowInitProgress(33);

chooser.LoadBitmap("Bitmaps/ChooserBackground.bmp");

background.LoadBitmap("Bitmaps/Background.bmp");

background\_night.LoadBitmap("Bitmaps/Background\_1.bmp");

shovel\_card.LoadBitmap();

ShowInitProgress(50);

sun.LoadBitmap();

sun\_flower\_card.LoadBitmap();

pea\_shooter\_card.LoadBitmap();

wallnut\_card.LoadBitmap();

cherrybomb\_card.LoadBitmap();

ice\_shooter\_card.LoadBitmap();

potatomine\_card.LoadBitmap();

shooter\_card.LoadBitmap();

squash\_card.LoadBitmap();

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

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

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

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

//CGameState::level = 1;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitTest(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(150, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(260, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(500, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 1));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel1(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1250, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1560, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1850, 2, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1900, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2000, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2050, 4));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel2(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// wallnut morning

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1270, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1360, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1550, 1, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1600, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1650, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2190, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2270, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2320, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2390, 1));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel3(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// cherrybomb morning

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1150, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1360, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1550, 4, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1650, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1860, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2270, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2320, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2350, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2420, 0));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel4(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// cone zombie night

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1150, 1, "conehead"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1360, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1500, 4, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1650, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1900, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2350, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2350, 1, "conehead"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2400, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2550, 0));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel5(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// ice shooter morning

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1000, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1270, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1360, 2, "conehead"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1950, 4, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2100, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2350, 3, "conehead"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2850, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2900, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2900, 4, "conehead"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(3050, 1));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel6(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// bucket night

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1000, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1270, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1460, 2, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1950, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2100, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2370, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2430, 0, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2850, 2, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2950, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(3200, 3));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel7(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// potatomine morning

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1270, 0, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1520, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1750, 1, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1800, 3, "conehead"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1870, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2150, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2230, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2250, 0, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2370, 4));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel8(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// newspaper night

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1270, 1, "newspaper"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1460, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1850, 2, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1950, 0, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2100, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2250, 3));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2850, 4, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2920, 3, "newspaper"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(3000, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(3120, 2));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel9(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// squash morning

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1050, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1270, 4, "newspaper"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1460, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1680, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1750, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1750, 4, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1870, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2450, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2480, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2550, 3));

return normalzombie\_vector;

}

std::vector<shared\_ptr<YNormalZombie>> zombieInitLevel10(std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector) {

// shooter night

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1270, 4, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1480, 1));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(1600, 2));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2250, 0, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2300, 3, "bucket"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2470, 0));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(2650, 1, "newspaper"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(3150, 3, "flag"));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(3170, 4));

normalzombie\_vector.push\_back(make\_shared<YNormalZombie>(3320, 2));

return normalzombie\_vector;

}

void CGameStateRun::OnBeginState()

{

const int BALL\_GAP = 90;

const int BALL\_XY\_OFFSET = 45;

const int BALL\_PER\_ROW = 7;

const int HITS\_LEFT = 10;

const int HITS\_LEFT\_X = 590;

const int HITS\_LEFT\_Y = 0;

const int BACKGROUND\_X = 0;

const int ANIMATION\_SPEED = 15;

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

flag = 0;

sun\_amount = 50;

generateSunFlowerFlag = false;

generatePeaShooterFlag = false;

generateWallNutFlag = false;

generateCherryBombFlag = false;

generateIceShooterFlag = false;

generatePotatomineFlag = false;

generateShooterFlag = false;

generateSquashFlag = false;

sun\_flower\_card.SetIsAlive(false);

pea\_shooter\_card.SetIsAlive(false);

wallnut\_card.SetIsAlive(false);

cherrybomb\_card.SetIsAlive(false);

ice\_shooter\_card.SetIsAlive(false);

potatomine\_card.SetIsAlive(false);

shooter\_card.SetIsAlive(false);

squash\_card.SetIsAlive(false);

shovelFlag = false;

sun\_flower\_card\_delay\_flag = 0;

peashooter\_card\_delay\_flag = 0;

wallnut\_card\_delay\_flag = 0;

cherrybomb\_card\_delay\_flag = 0;

iceshooter\_card\_delay\_flag = 0;

potatomine\_card\_delay\_flag = 0;

shooter\_card\_delay\_flag = 0;

squash\_card\_delay\_flag = 0;

car0 = YCar(0);

car1 = YCar(1);

car2 = YCar(2);

car3 = YCar(3);

car4 = YCar(4);

car0.LoadBitmap();

car1.LoadBitmap();

car2.LoadBitmap();

car3.LoadBitmap();

car4.LoadBitmap();

car0\_flag = true;

car1\_flag = true;

car2\_flag = true;

car3\_flag = true;

car4\_flag = true;

car0\_sound\_flag = true;

car1\_sound\_flag = true;

car2\_sound\_flag = true;

car3\_sound\_flag = true;

car4\_sound\_flag = true;

zombie\_home\_flag = true;

normalzombie\_vector.clear();

sunflower\_vector.clear();

peashooter\_vector.clear();

wallnut\_vector.clear();

iceshooter\_vector.clear();

shooter\_vector.clear();

squash\_vector.clear();

map.clear();

zombie\_fast\_mode = false;

sun.SetY(-200);

if (CGameState::level == 0) {

night\_mode = false;

normalzombie\_vector = zombieInitTest(normalzombie\_vector);

}

else if (CGameState::level == 1) {

night\_mode = false;

normalzombie\_vector = zombieInitLevel1(normalzombie\_vector);

}

else if (CGameState::level == 2) {

night\_mode = false;

normalzombie\_vector = zombieInitLevel2(normalzombie\_vector);

}

else if (CGameState::level == 3) {

night\_mode = false;

normalzombie\_vector = zombieInitLevel3(normalzombie\_vector);

}

else if (CGameState::level == 4) {

night\_mode = true;

normalzombie\_vector = zombieInitLevel4(normalzombie\_vector);

}

else if (CGameState::level == 5) {

night\_mode = false;

normalzombie\_vector = zombieInitLevel5(normalzombie\_vector);

}

else if (CGameState::level == 6) {

night\_mode = true;

normalzombie\_vector = zombieInitLevel6(normalzombie\_vector);

}

else if (CGameState::level == 7) {

night\_mode = false;

normalzombie\_vector = zombieInitLevel7(normalzombie\_vector);

}

else if (CGameState::level == 8) {

night\_mode = true;

normalzombie\_vector = zombieInitLevel8(normalzombie\_vector);

}

else if (CGameState::level == 9) {

night\_mode = false;

normalzombie\_vector = zombieInitLevel9(normalzombie\_vector);

}

else if (CGameState::level == 10) {

night\_mode = true;

normalzombie\_vector = zombieInitLevel10(normalzombie\_vector);

}

if (night\_mode) {

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

sun.SetIsAlive(false);

}

else {

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

sun.SetIsAlive(true);

}

for (auto normalzombie\_sp : normalzombie\_vector) {

normalzombie\_sp->LoadBitmap();

}

}

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

{

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

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

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

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

const char KEY\_Z = 90;

const char KEY\_S = 83;

const char KEY\_D = 68;

if (nChar == KEY\_Z) {

zombie\_fast\_mode = true;

}

if (nChar == KEY\_S) {

sun\_amount = 500;

}

if (nChar == KEY\_D) {

for (auto z : normalzombie\_vector) {

z->SetIsAlive(false);

}

}

}

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

{

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

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

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

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

const char KEY\_Z = 90;

if (nChar == KEY\_Z) {

zombie\_fast\_mode = false;

}

}

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

{

if (generateSunFlowerFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

int tx = map.getXmyMapLocation(point.x, point.y);

int ty = map.getYmyMapLocation(point.x, point.y);

map.setmyMap(point.x, point.y);

auto sp = make\_shared<YSunFlower>(tx, ty);

sp->LoadBitmap();

sunflower\_vector.push\_back(sp);

generateSunFlowerFlag = false;

}

else if (generatePeaShooterFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

int tx = map.getXmyMapLocation(point.x, point.y);

int ty = map.getYmyMapLocation(point.x, point.y);

map.setmyMap(point.x, point.y);

auto sp = make\_shared<YPeaShooter>(tx, ty);

sp->LoadBitmap();

peashooter\_vector.push\_back(sp);

generatePeaShooterFlag = false;

}

else if (CGameState::level > 1 && generateWallNutFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

int tx = map.getXmyMapLocation(point.x, point.y);

int ty = map.getYmyMapLocation(point.x, point.y);

map.setmyMap(point.x, point.y);

auto sp = make\_shared<YWallNut>(tx, ty);

sp->LoadBitmap();

wallnut\_vector.push\_back(sp);

generateWallNutFlag = false;

}

else if (CGameState::level > 2 && generateCherryBombFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

auto sp = make\_shared<YCherryBomb>(point.x, point.y);

sp->LoadBitmap();

cherrybomb\_vector.push\_back(sp);

generateCherryBombFlag = false;

}

else if (CGameState::level > 4 && generateIceShooterFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

int tx = map.getXmyMapLocation(point.x, point.y);

int ty = map.getYmyMapLocation(point.x, point.y);

map.setmyMap(point.x, point.y);

auto sp = make\_shared<YIceShooter>(tx, ty);

sp->LoadBitmap();

iceshooter\_vector.push\_back(sp);

generateIceShooterFlag = false;

}

else if (CGameState::level > 6 && generatePotatomineFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

int tx = map.getXmyMapLocation(point.x, point.y);

int ty = map.getYmyMapLocation(point.x, point.y);

map.setmyMap(point.x, point.y);

auto sp = make\_shared<YPotatoMine>(tx, ty);

sp->LoadBitmap();

potatomine\_vector.push\_back(sp);

generatePotatomineFlag = false;

}

else if (CGameState::level > 8 && generateSquashFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

int tx = map.getXmyMapLocation(point.x, point.y);

int ty = map.getYmyMapLocation(point.x, point.y);

map.setmyMap(point.x, point.y);

auto sp = make\_shared<YSquash>(tx, ty);

sp->LoadBitmap();

squash\_vector.push\_back(sp);

generateSquashFlag = false;

}

else if (CGameState::level > 9 && generateShooterFlag && !map.checkmyMap(point.x, point.y) && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

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

int tx = map.getXmyMapLocation(point.x, point.y);

int ty = map.getYmyMapLocation(point.x, point.y);

map.setmyMap(point.x, point.y);

auto sp = make\_shared<YShooter>(tx, ty);

sp->LoadBitmap();

shooter\_vector.push\_back(sp);

generateShooterFlag = false;

}

else if (shovelFlag && point.x > 30 && point.x < 770 && point.y>78 && point.y < 571) {

map.unsetmyMap(point.x, point.y);

shovelFlag = false;

}

// sun

if (point.x > sun.GetX() && point.y > sun.GetY() &&

point.x < sun.GetX() + 75 && point.y < sun.GetY() + 75 && sun.IsAlive()) {

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

sun.SetIsAlive(false);

sun\_amount += 25;

sun.SetY(-400);

}

for (size\_t i = 0; i < sunflower\_vector.size(); i++) {

if (sunflower\_vector.at(i)->GetSunIsAlive() &&

point.x > sunflower\_vector.at(i)->GetSunX() &&

point.x < sunflower\_vector.at(i)->GetSunX() + 75 &&

point.y > sunflower\_vector.at(i)->GetSunY() &&

point.y < sunflower\_vector.at(i)->GetSunY() + 75

) {

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

sun\_amount += 25;

sunflower\_vector.at(i)->initSun();

}

}

// card

if (point.x > sun\_flower\_card.GetX() && point.y > sun\_flower\_card.GetY() && point.x < sun\_flower\_card.GetX() + 65 && point.y < sun\_flower\_card.GetY() + 90 && sun\_flower\_card.IsAlive()) {

sun\_flower\_card.SetIsAlive(false);

sun\_amount -= sun\_flower\_card.GetSunCost();

generateSunFlowerFlag = true;

sun\_flower\_card\_delay\_flag = 150;

}

else if (point.x > pea\_shooter\_card.GetX() && point.y > pea\_shooter\_card.GetY() && point.x < pea\_shooter\_card.GetX() + 65 && point.y < pea\_shooter\_card.GetY() + 90 && pea\_shooter\_card.IsAlive()) {

pea\_shooter\_card.SetIsAlive(false);

sun\_amount -= pea\_shooter\_card.GetSunCost();

generatePeaShooterFlag = true;

peashooter\_card\_delay\_flag = 150;

}

else if (CGameState::level > 1 && point.x > wallnut\_card.GetX() && point.y > wallnut\_card.GetY() && point.x < wallnut\_card.GetX() + 65 && point.y < wallnut\_card.GetY() + 90 && wallnut\_card.IsAlive()) {

wallnut\_card.SetIsAlive(false);

sun\_amount -= wallnut\_card.GetSunCost();

generateWallNutFlag = true;

wallnut\_card\_delay\_flag = 150;

}

else if (CGameState::level > 2 && point.x > cherrybomb\_card.GetX() && point.y > cherrybomb\_card.GetY() && point.x < cherrybomb\_card.GetX() + 65 && point.y < cherrybomb\_card.GetY() + 90 && cherrybomb\_card.IsAlive()) {

cherrybomb\_card.SetIsAlive(false);

sun\_amount -= cherrybomb\_card.GetSunCost();

generateCherryBombFlag = true;

cherrybomb\_card\_delay\_flag = 150;

}

else if (CGameState::level > 4 && point.x > ice\_shooter\_card.GetX() && point.y > ice\_shooter\_card.GetY() && point.x < ice\_shooter\_card.GetX() + 65 && point.y < ice\_shooter\_card.GetY() + 90 && ice\_shooter\_card.IsAlive()) {

ice\_shooter\_card.SetIsAlive(false);

sun\_amount -= ice\_shooter\_card.GetSunCost();

generateIceShooterFlag = true;

iceshooter\_card\_delay\_flag = 150;

}

else if (CGameState::level > 6 && point.x > potatomine\_card.GetX() && point.y > potatomine\_card.GetY() && point.x < potatomine\_card.GetX() + 65 && point.y < potatomine\_card.GetY() + 90 && potatomine\_card.IsAlive()) {

potatomine\_card.SetIsAlive(false);

sun\_amount -= potatomine\_card.GetSunCost();

generatePotatomineFlag = true;

potatomine\_card\_delay\_flag = 150;

}

else if (CGameState::level > 8 && point.x > squash\_card.GetX() && point.y > squash\_card.GetY() && point.x < squash\_card.GetX() + 65 && point.y < squash\_card.GetY() + 90 && squash\_card.IsAlive()) {

squash\_card.SetIsAlive(false);

sun\_amount -= squash\_card.GetSunCost();

generateSquashFlag = true;

squash\_card\_delay\_flag = 150;

}

else if (CGameState::level > 9 && point.x > shooter\_card.GetX() && point.y > shooter\_card.GetY() && point.x < shooter\_card.GetX() + 65 && point.y < shooter\_card.GetY() + 90 && shooter\_card.IsAlive()) {

shooter\_card.SetIsAlive(false);

sun\_amount -= shooter\_card.GetSunCost();

generateShooterFlag = true;

shooter\_card\_delay\_flag = 150;

}

else if (point.x > shovel\_card.GetX() && point.y > shovel\_card.GetY() && point.x < shovel\_card.GetX() + 82 && point.y < shovel\_card.GetY() + 82) {

shovelFlag = true;

shovel\_card.SetIsAlive(false);

}

if (sun\_flower\_card.GetSunCost() > sun\_amount) {

sun\_flower\_card.SetIsAlive(false);

}

if (pea\_shooter\_card.GetSunCost() > sun\_amount) {

pea\_shooter\_card.SetIsAlive(false);

}

if (CGameState::level > 1 && wallnut\_card.GetSunCost() > sun\_amount) {

wallnut\_card.SetIsAlive(false);

}

if (CGameState::level > 2 && cherrybomb\_card.GetSunCost() > sun\_amount) {

cherrybomb\_card.SetIsAlive(false);

}

if (CGameState::level > 4 && ice\_shooter\_card.GetSunCost() > sun\_amount) {

ice\_shooter\_card.SetIsAlive(false);

}

if (CGameState::level > 6 && potatomine\_card.GetSunCost() > sun\_amount) {

potatomine\_card.SetIsAlive(false);

}

if (CGameState::level > 8 && squash\_card.GetSunCost() > sun\_amount) {

squash\_card.SetIsAlive(false);

}

if (CGameState::level > 9 && shooter\_card.GetSunCost() > sun\_amount) {

shooter\_card.SetIsAlive(false);

}

}

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

{

}

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

{

}

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::OnMove()

{

chooser.SetTopLeft(0, 0);

if (picX > -500 && flag == 0) {

picX -= 8;

}

else if (flag == 0) {

Sleep(1000);

flag = 1;

}

else if (picX < -220) {

picX += 8;

}

else if (flag == 4 && picX == 0) {

picX = 0;

}

else if (flag == 4) {

picX += 2;

}

else {

flag = 2;

picX = -220;

}

if (night\_mode) {

background\_night.SetTopLeft(picX, picY);

}

else {

background.SetTopLeft(picX, picY);

}

if (flag == 2) {

if (!night\_mode) {

sun.OnMove();

}

// card

if (sun\_flower\_card\_delay\_flag > 0) {

sun\_flower\_card\_delay\_flag--;

}

if (peashooter\_card\_delay\_flag > 0) {

peashooter\_card\_delay\_flag--;

}

if (CGameState::level > 1 && wallnut\_card\_delay\_flag > 0) {

wallnut\_card\_delay\_flag--;

}

if (CGameState::level > 2 && cherrybomb\_card\_delay\_flag > 0) {

cherrybomb\_card\_delay\_flag--;

}

if (CGameState::level > 4 && iceshooter\_card\_delay\_flag > 0) {

iceshooter\_card\_delay\_flag--;

}

if (CGameState::level > 6 && potatomine\_card\_delay\_flag > 0) {

potatomine\_card\_delay\_flag--;

}

if (CGameState::level > 8 && squash\_card\_delay\_flag > 0) {

squash\_card\_delay\_flag--;

}

if (CGameState::level > 9 && shooter\_card\_delay\_flag > 0) {

shooter\_card\_delay\_flag--;

}

// shovel the plant

for (size\_t i = 0; i < sunflower\_vector.size(); i++) {

if (!map.checkmyMap(sunflower\_vector.at(i)->GetX(), sunflower\_vector.at(i)->GetY())) {

sunflower\_vector.erase(sunflower\_vector.begin() + i); //if map is zero, delete the plant

}

else {

sunflower\_vector.at(i)->OnMove();

}

}

for (size\_t i = 0; i < peashooter\_vector.size(); i++) {

if (!map.checkmyMap(peashooter\_vector.at(i)->GetX(), peashooter\_vector.at(i)->GetY())) {

peashooter\_vector.erase(peashooter\_vector.begin() + i); //if map is zero, delete the plant

}

else {

peashooter\_vector.at(i)->OnMove();

}

}

if (CGameState::level > 1) {

for (size\_t i = 0; i < wallnut\_vector.size(); i++) {

if (!map.checkmyMap(wallnut\_vector.at(i)->GetX(), wallnut\_vector.at(i)->GetY())) {

wallnut\_vector.erase(wallnut\_vector.begin() + i); //if map is zero, delete the plant

}

else {

wallnut\_vector.at(i)->OnMove();

}

}

}

if (CGameState::level > 2) {

for (size\_t i = 0; i < cherrybomb\_vector.size(); i++) {

cherrybomb\_vector.at(i)->OnMove();

}

}

if (CGameState::level > 4) {

for (size\_t i = 0; i < iceshooter\_vector.size(); i++) {

if (!map.checkmyMap(iceshooter\_vector.at(i)->GetX(), iceshooter\_vector.at(i)->GetY())) {

iceshooter\_vector.erase(iceshooter\_vector.begin() + i); //if map is zero, delete the plant

}

else {

iceshooter\_vector.at(i)->OnMove();

}

}

}

if (CGameState::level > 6) {

for (size\_t i = 0; i < potatomine\_vector.size(); i++) {

if (!map.checkmyMap(potatomine\_vector.at(i)->GetX(), potatomine\_vector.at(i)->GetY())) {

potatomine\_vector.erase(potatomine\_vector.begin() + i); //if map is zero, delete the plant

}

else {

potatomine\_vector.at(i)->OnMove();

}

}

}

if (CGameState::level > 8) {

for (size\_t i = 0; i < squash\_vector.size(); i++) {

if (!map.checkmyMap(squash\_vector.at(i)->GetX(), squash\_vector.at(i)->GetY())) {

squash\_vector.erase(squash\_vector.begin() + i); //if map is zero, delete the plant

}

else {

squash\_vector.at(i)->OnMove();

}

}

}

if (CGameState::level > 9) {

for (size\_t i = 0; i < shooter\_vector.size(); i++) {

if (!map.checkmyMap(shooter\_vector.at(i)->GetX(), shooter\_vector.at(i)->GetY())) {

shooter\_vector.erase(shooter\_vector.begin() + i); //if map is zero, delete the plant

}

else {

shooter\_vector.at(i)->OnMove();

}

}

}

// zombie

for (size\_t i = 0; i < normalzombie\_vector.size(); i++) {

// zombie eat plant

for (size\_t j = 0; j < sunflower\_vector.size(); j++) {

if (sunflower\_vector.at(j)->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

sunflower\_vector.at(j)->LostBlood(normalzombie\_vector.at(i)->GetAttackPower());

if (sunflower\_vector.at(j)->GetBlood() < 1) {

sunflower\_vector.at(j)->SetIsAlive(false);

map.unsetmyMap(sunflower\_vector.at(j)->GetX(), sunflower\_vector.at(j)->GetY());

sunflower\_vector.erase(sunflower\_vector.begin() + j);

}

}

}

for (size\_t j = 0; j < peashooter\_vector.size(); j++) {

if (peashooter\_vector.at(j)->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

peashooter\_vector.at(j)->LostBlood(normalzombie\_vector.at(i)->GetAttackPower());

if (peashooter\_vector.at(j)->GetBlood() < 1) {

peashooter\_vector.at(j)->SetIsAlive(false);

map.unsetmyMap(peashooter\_vector.at(j)->GetX(), peashooter\_vector.at(j)->GetY());

peashooter\_vector.erase(peashooter\_vector.begin() + j);

}

}

}

if (CGameState::level > 1) {

for (size\_t j = 0; j < wallnut\_vector.size(); j++) {

if (wallnut\_vector.at(j)->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

wallnut\_vector.at(j)->LostBlood(normalzombie\_vector.at(i)->GetAttackPower());

if (wallnut\_vector.at(j)->GetBlood() < 1) {

wallnut\_vector.at(j)->SetIsAlive(false);

map.unsetmyMap(wallnut\_vector.at(j)->GetX(), wallnut\_vector.at(j)->GetY());

wallnut\_vector.erase(wallnut\_vector.begin() + j);

}

}

}

}

if (CGameState::level > 2) {

if (!cherrybomb\_vector.empty() && cherrybomb\_vector.at(0)->Bomb()

&& cherrybomb\_vector.at(0)->checkNearbyZombies(normalzombie\_vector.at(i)->GetX(), normalzombie\_vector.at(i)->GetY())) {

normalzombie\_vector.at(i)->SetBombFlag();

normalzombie\_vector.at(i)->OnMove(std::string("bomb"));

}

if (CGameState::level > 6 && !potatomine\_vector.empty()) {

for (size\_t j = 0; j < potatomine\_vector.size(); j++) {

if (potatomine\_vector.at(j)->Bomb()

&& potatomine\_vector.at(j)->checkNearbyZombies(normalzombie\_vector.at(i)->GetX(), normalzombie\_vector.at(i)->GetY())) {

normalzombie\_vector.at(i)->SetBombFlag();

normalzombie\_vector.at(i)->OnMove(std::string("bomb"));

}

}

}

if (map.checkmyMap(normalzombie\_vector.at(i)->GetX() + 80, normalzombie\_vector.at(i)->GetY() + 35)

&& !map.checkmyMap(normalzombie\_vector.at(i)->GetX() + 90, normalzombie\_vector.at(i)->GetY() + 35)

&& normalzombie\_vector.at(i)->IsAlive()) {

normalzombie\_vector.at(i)->OnMove(std::string("attack"));

}

else if (!normalzombie\_vector.at(i)->IsAlive()) {

normalzombie\_vector.at(i)->OnMove(std::string("die"));

}

else {

if (zombie\_fast\_mode)

normalzombie\_vector.at(i)->OnMove(std::string("walk"), true);

else

normalzombie\_vector.at(i)->OnMove(std::string("walk"));

}

if (CGameState::level > 4) {

for (size\_t j = 0; j < iceshooter\_vector.size(); j++) {

if (iceshooter\_vector.at(j)->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

iceshooter\_vector.at(j)->LostBlood(normalzombie\_vector.at(i)->GetAttackPower());

if (iceshooter\_vector.at(j)->GetBlood() < 1) {

iceshooter\_vector.at(j)->SetIsAlive(false);

map.unsetmyMap(iceshooter\_vector.at(j)->GetX(), iceshooter\_vector.at(j)->GetY());

iceshooter\_vector.erase(iceshooter\_vector.begin() + j);

}

}

}

}

if (CGameState::level > 6) {

for (size\_t j = 0; j < potatomine\_vector.size(); j++) {

if (potatomine\_vector.at(j)->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

potatomine\_vector.at(j)->SetZombieChecked();

potatomine\_vector.at(j)->LostBlood(normalzombie\_vector.at(i)->GetAttackPower());

if (potatomine\_vector.at(j)->GetBlood() < 1) {

potatomine\_vector.at(j)->SetIsAlive(false);

map.unsetmyMap(potatomine\_vector.at(j)->GetX(), potatomine\_vector.at(j)->GetY());

potatomine\_vector.erase(potatomine\_vector.begin() + j);

}

}

}

}

if (CGameState::level > 8) {

for (size\_t j = 0; j < squash\_vector.size(); j++) {

if (squash\_vector.at(j)->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

squash\_vector.at(j)->SetZombieChecked();

normalzombie\_vector.at(i)->SetIsAlive(false);

if (!squash\_vector.at(j)->IsAlive()) {

map.unsetmyMap(squash\_vector.at(j)->GetX(), squash\_vector.at(j)->GetY());

squash\_vector.erase(squash\_vector.begin() + j);

}

}

}

}

if (CGameState::level > 9) {

for (size\_t j = 0; j < shooter\_vector.size(); j++) {

if (normalzombie\_vector.at(i)->IsAlive() && shooter\_vector.at(j)->isClose(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

shooter\_vector.at(j)->SetClose(true);

}

else if (!normalzombie\_vector.at(i)->IsAlive()) {

shooter\_vector.at(j)->SetClose(false);

}

if (shooter\_vector.at(j)->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX() + 70, normalzombie\_vector.at(i)->GetY() + 30)) {

shooter\_vector.at(j)->LostBlood(normalzombie\_vector.at(i)->GetAttackPower());

if (shooter\_vector.at(j)->GetBlood() < 1) {

shooter\_vector.at(j)->SetIsAlive(false);

map.unsetmyMap(shooter\_vector.at(j)->GetX(), shooter\_vector.at(j)->GetY());

shooter\_vector.erase(shooter\_vector.begin() + j);

}

}

}

}

}

else {

if (map.checkmyMap(normalzombie\_vector.at(i)->GetX() + 80, normalzombie\_vector.at(i)->GetY() + 35)

&& !map.checkmyMap(normalzombie\_vector.at(i)->GetX() + 90, normalzombie\_vector.at(i)->GetY() + 35)

&& normalzombie\_vector.at(i)->IsAlive()) {

normalzombie\_vector.at(i)->OnMove(std::string("attack"));

}

else if (!normalzombie\_vector.at(i)->IsAlive()) {

normalzombie\_vector.at(i)->OnMove(std::string("die"));

}

else {

if (zombie\_fast\_mode)

normalzombie\_vector.at(i)->OnMove(std::string("walk"), true);

else

normalzombie\_vector.at(i)->OnMove(std::string("walk"));

}

}

if (CGameState::level > 2) {

for (size\_t i = 0; i < cherrybomb\_vector.size(); i++) {

if (!cherrybomb\_vector.at(i)->IsAlive()) {

cherrybomb\_vector.erase(cherrybomb\_vector.begin() + i);

}

}

}

if (CGameState::level > 6) {

for (size\_t i = 0; i < potatomine\_vector.size(); i++) {

if (!potatomine\_vector.at(i)->IsAlive()) {

map.unsetmyMap(potatomine\_vector.at(i)->GetX(), potatomine\_vector.at(i)->GetY());

potatomine\_vector.erase(potatomine\_vector.begin() + i);

}

}

}

// zombie walk to car -> car move

if (car0.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 48 && normalzombie\_vector.at(i)->GetX() < car0.GetX() + 10) || !car0\_flag) {

if (car0\_sound\_flag) {

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

car0\_sound\_flag = false;

}

car0\_flag = false;

car0.OnMove();

}

if (car1.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 152 && normalzombie\_vector.at(i)->GetX() < car1.GetX() - 10) || !car1\_flag) {

if (car1\_sound\_flag) {

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

car1\_sound\_flag = false;

}

car1\_flag = false;

car1.OnMove();

}

if (car2.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 240 && normalzombie\_vector.at(i)->GetX() < car2.GetX() - 10) || !car2\_flag) {

if (car2\_sound\_flag) {

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

car2\_sound\_flag = false;

}

car2\_flag = false;

car2.OnMove();

}

if (car3.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 338 && normalzombie\_vector.at(i)->GetX() < car3.GetX() - 10) || !car3\_flag) {

if (car3\_sound\_flag) {

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

car3\_sound\_flag = false;

}

car3\_flag = false;

car3.OnMove();

}

if (car4.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 434 && normalzombie\_vector.at(i)->GetX() < car4.GetX() - 10) || !car4\_flag) {

if (car4\_sound\_flag) {

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

car4\_sound\_flag = false;

}

car4\_flag = false;

car4.OnMove();

}

// car hits zombies

if (car0.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 48 && normalzombie\_vector.at(i)->GetX() < car0.GetX() - 10)) {

normalzombie\_vector.at(i)->SetIsAlive(false);

}

if (car1.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 152 && normalzombie\_vector.at(i)->GetX() < car1.GetX() - 10)) {

normalzombie\_vector.at(i)->SetIsAlive(false);

}

if (car2.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 240 && normalzombie\_vector.at(i)->GetX() < car2.GetX() - 10)) {

normalzombie\_vector.at(i)->SetIsAlive(false);

}

if (car3.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 338 && normalzombie\_vector.at(i)->GetX() < car3.GetX() - 10)) {

normalzombie\_vector.at(i)->SetIsAlive(false);

}

if (car4.IsAlive() && (normalzombie\_vector.at(i)->GetY() == 434 && normalzombie\_vector.at(i)->GetX() < car4.GetX() - 10)) {

normalzombie\_vector.at(i)->SetIsAlive(false);

}

for (auto p : peashooter\_vector) {

int temp\_y = map.getYmyMapLocation(normalzombie\_vector.at(i)->GetX(), normalzombie\_vector.at(i)->GetY() + 30);

if (p->checkBulletCollideWithZombie(normalzombie\_vector.at(i)->GetX(), temp\_y)) {

normalzombie\_vector.at(i)->LostBlood(1);

}

}

if (CGameState::level > 4) {

for (auto p : iceshooter\_vector) {

int temp\_y = map.getYmyMapLocation(normalzombie\_vector.at(i)->GetX(), normalzombie\_vector.at(i)->GetY() + 30);

if (p->checkBulletCollideWithZombie(normalzombie\_vector.at(i)->GetX(), temp\_y)) {

normalzombie\_vector.at(i)->LostBlood(2);

normalzombie\_vector.at(i)->freezen();

}

}

}

if (CGameState::level > 8) {

for (auto p : squash\_vector) {

int temp\_y = map.getYmyMapLocation(normalzombie\_vector.at(i)->GetX(), normalzombie\_vector.at(i)->GetY() + 30);

if (p->checkPlantCollideWithZombie(normalzombie\_vector.at(i)->GetX(), temp\_y)) {

normalzombie\_vector.at(i)->LostBlood(10000);

}

}

}

if (CGameState::level > 9) {

for (auto p : shooter\_vector) {

int temp\_y = map.getYmyMapLocation(normalzombie\_vector.at(i)->GetX(), normalzombie\_vector.at(i)->GetY() + 30);

if (p->checkBulletCollideWithZombie(normalzombie\_vector.at(i)->GetX(), temp\_y)) {

normalzombie\_vector.at(i)->LostBlood(1);

}

}

}

if (normalzombie\_vector.at(i)->GetX() > 900 && !normalzombie\_vector.at(i)->IsAlive()) {

normalzombie\_vector.erase(normalzombie\_vector.begin() + i);

}

}

for (size\_t i = 0; i < normalzombie\_vector.size(); i++) {

if (normalzombie\_vector.at(i)->GetX() < -70 && normalzombie\_vector.at(i)->IsAlive()) {

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

flag = 4;

}

}

// card

if (sun\_amount >= sun\_flower\_card.GetSunCost() && sun\_flower\_card\_delay\_flag == 0) {

sun\_flower\_card.SetIsAlive(true);

}

if (sun\_amount >= pea\_shooter\_card.GetSunCost() && peashooter\_card\_delay\_flag == 0) {

pea\_shooter\_card.SetIsAlive(true);

}

if (CGameState::level > 1 && sun\_amount >= wallnut\_card.GetSunCost() && wallnut\_card\_delay\_flag == 0) {

wallnut\_card.SetIsAlive(true);

}

if (CGameState::level > 2 && sun\_amount >= cherrybomb\_card.GetSunCost() && cherrybomb\_card\_delay\_flag == 0) {

cherrybomb\_card.SetIsAlive(true);

}

if (CGameState::level > 4 && sun\_amount >= ice\_shooter\_card.GetSunCost() && iceshooter\_card\_delay\_flag == 0) {

ice\_shooter\_card.SetIsAlive(true);

}

if (CGameState::level > 6 && sun\_amount >= potatomine\_card.GetSunCost() && potatomine\_card\_delay\_flag == 0) {

potatomine\_card.SetIsAlive(true);

}

if (CGameState::level > 8 && sun\_amount >= squash\_card.GetSunCost() && squash\_card\_delay\_flag == 0) {

squash\_card.SetIsAlive(true);

}

if (CGameState::level > 9 && sun\_amount >= shooter\_card.GetSunCost() && shooter\_card\_delay\_flag == 0) {

shooter\_card.SetIsAlive(true);

}

if (!shovelFlag) {

shovel\_card.SetIsAlive(true);

}

// go to game state over

if (normalzombie\_vector.empty()) {

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

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

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

victoryflag = 1;

CGameState::level += 1;

GotoGameState(GAME\_STATE\_OVER);

}

}

if (flag == 4) {

if (zombie\_home\_flag) {

normalzombie\_vector.at(0)->SetX(10);

normalzombie\_vector.at(0)->SetY(300);

zombie\_home\_flag = false;

}

normalzombie\_vector.at(0)->OnMove(std::string("walk"));

if (normalzombie\_vector.at(0)->GetX() < -35) {

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

victoryflag = 0;

GotoGameState(GAME\_STATE\_OVER);

}

}

// 修改cursor樣式

//if (generateSunFlowerFlag) {

// SetCursor(AfxGetApp()->LoadCursor(".\\bitmaps\\SunFlower\\SunFlower\_0.bmp"));

//}

}

void CGameStateRun::OnShow()

{

if (night\_mode) {

background\_night.ShowBitmap();

}

else {

background.ShowBitmap();

}

if (flag == 4) {

normalzombie\_vector.at(0)->OnShow(std::string("walk"));

}

if (flag == 2) {

for (size\_t i = 0; i < sunflower\_vector.size(); i++) {

sunflower\_vector.at(i)->OnShow();

}

if (CGameState::level > 1) {

for (size\_t i = 0; i < wallnut\_vector.size(); i++) {

wallnut\_vector.at(i)->OnShow();

}

}

if (CGameState::level > 2) {

for (size\_t i = 0; i < cherrybomb\_vector.size(); i++) {

cherrybomb\_vector.at(i)->OnShow();

}

}

if (CGameState::level > 4) {

for (size\_t i = 0; i < iceshooter\_vector.size(); i++) {

iceshooter\_vector.at(i)->OnShow();

}

}

if (CGameState::level > 6) {

for (size\_t i = 0; i < potatomine\_vector.size(); i++) {

potatomine\_vector.at(i)->OnShow();

}

}

if (CGameState::level > 8) {

for (size\_t i = 0; i < squash\_vector.size(); i++) {

squash\_vector.at(i)->OnShow();

}

}

if (CGameState::level > 9) {

for (size\_t i = 0; i < shooter\_vector.size(); i++) {

shooter\_vector.at(i)->OnShow();

}

}

for (size\_t i = 0; i < peashooter\_vector.size(); i++) {

peashooter\_vector.at(i)->OnShow();

}

for (auto normalzombie : normalzombie\_vector) {

if (normalzombie->GetX() < 950) {

if (map.checkmyMap(normalzombie->GetX() + 80, normalzombie->GetY() + 35)

&& !map.checkmyMap(normalzombie->GetX() + 90, normalzombie->GetY() + 35)

&& normalzombie->IsAlive())

{

normalzombie->OnShow(std::string("attack"));

}

else if (!normalzombie->IsAlive()) {

normalzombie->OnShow(std::string("die"));

}

else {

normalzombie->OnShow(std::string("walk"));

}

}

}

sun.OnShow();

chooser.ShowBitmap();

shovel\_card.OnShow();

sun\_flower\_card.OnShow();

pea\_shooter\_card.OnShow();

if (CGameState::level > 1) {

wallnut\_card.OnShow();

}

if (CGameState::level > 2) {

cherrybomb\_card.OnShow();

}

if (CGameState::level > 4) {

ice\_shooter\_card.OnShow();

}

if (CGameState::level > 6) {

potatomine\_card.OnShow();

}

if (CGameState::level > 8) {

squash\_card.OnShow();

}

if (CGameState::level > 9) {

shooter\_card.OnShow();

}

car0.OnShow();

car1.OnShow();

car2.OnShow();

car3.OnShow();

car4.OnShow();

}

// sun amount

if (flag == 2) {

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

CFont f, \*fp;

f.CreatePointFont(90, "微軟正黑體"); // 產生 font f; 160表示16 point的字

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

pDC->SetBkMode(TRANSPARENT);

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

if (sun\_amount == 50)

pDC->TextOut(30, 62, "50");

else if (sun\_amount == 0)

pDC->TextOut(34, 62, "0");

else if (sun\_amount == 25)

pDC->TextOut(30, 62, "25");

else if (sun\_amount == 75)

pDC->TextOut(30, 62, "75");

else if (sun\_amount == 100)

pDC->TextOut(26, 62, "100");

else if (sun\_amount == 125)

pDC->TextOut(26, 62, "125");

else if (sun\_amount == 150)

pDC->TextOut(26, 62, "150");

else if (sun\_amount == 175)

pDC->TextOut(26, 62, "175");

else if (sun\_amount == 200)

pDC->TextOut(26, 62, "200");

else if (sun\_amount == 225)

pDC->TextOut(26, 62, "225");

else if (sun\_amount == 250)

pDC->TextOut(26, 62, "250");

else if (sun\_amount == 275)

pDC->TextOut(26, 62, "275");

else if (sun\_amount == 300)

pDC->TextOut(26, 62, "300");

else if (sun\_amount == 325)

pDC->TextOut(26, 62, "325");

else if (sun\_amount == 350)

pDC->TextOut(26, 62, "350");

else if (sun\_amount == 375)

pDC->TextOut(26, 62, "375");

else if (sun\_amount == 400)

pDC->TextOut(26, 62, "400");

else if (sun\_amount == 425)

pDC->TextOut(26, 62, "425");

else if (sun\_amount == 450)

pDC->TextOut(26, 62, "450");

else if (sun\_amount == 475)

pDC->TextOut(26, 62, "475");

else if (sun\_amount == 500)

pDC->TextOut(26, 62, "500");

else if (sun\_amount == 525)

pDC->TextOut(26, 62, "525");

else if (sun\_amount == 550)

pDC->TextOut(26, 62, "550");

else if (sun\_amount == 575)

pDC->TextOut(26, 62, "575");

else if (sun\_amount == 600)

pDC->TextOut(26, 62, "600");

else if (sun\_amount == 625)

pDC->TextOut(26, 62, "625");

else if (sun\_amount == 650)

pDC->TextOut(26, 62, "650");

else if (sun\_amount == 675)

pDC->TextOut(26, 62, "675");

else if (sun\_amount == 700)

pDC->TextOut(26, 62, "700");

else {

pDC->TextOut(26, 62, "###");

}

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

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

}

// zombie amount

if (flag == 2) {

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

CFont f, \*fp;

f.CreatePointFont(100, "微軟正黑體"); // 產生 font f; 160表示16 point的字

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

pDC->SetBkMode(TRANSPARENT);

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

pDC->TextOut(650, 580, "left zombie: ");

int zombie\_amount = normalzombie\_vector.size();

if (zombie\_amount == 0)

pDC->TextOut(755, 580, "0");

else if (zombie\_amount == 1)

pDC->TextOut(755, 580, "1");

else if (zombie\_amount == 2)

pDC->TextOut(755, 580, "2");

else if (zombie\_amount == 3)

pDC->TextOut(755, 580, "3");

else if (zombie\_amount == 4)

pDC->TextOut(755, 580, "4");

else if (zombie\_amount == 5)

pDC->TextOut(755, 580, "5");

else if (zombie\_amount == 6)

pDC->TextOut(755, 580, "6");

else if (zombie\_amount == 7)

pDC->TextOut(755, 580, "7");

else if (zombie\_amount == 8)

pDC->TextOut(755, 580, "8");

else if (zombie\_amount == 9)

pDC->TextOut(755, 580, "9");

else if (zombie\_amount == 10)

pDC->TextOut(755, 580, "10");

else if (zombie\_amount == 11)

pDC->TextOut(755, 580, "11");

else if (zombie\_amount == 12)

pDC->TextOut(755, 580, "12");

else if (zombie\_amount == 13)

pDC->TextOut(755, 580, "13");

else if (zombie\_amount == 14)

pDC->TextOut(755, 580, "14");

else if (zombie\_amount == 15)

pDC->TextOut(755, 580, "15");

else if (zombie\_amount == 16)

pDC->TextOut(755, 580, "16");

else if (zombie\_amount == 17)

pDC->TextOut(755, 580, "17");

else if (zombie\_amount == 18)

pDC->TextOut(755, 580, "18");

else if (zombie\_amount == 19)

pDC->TextOut(755, 580, "19");

else if (zombie\_amount == 20)

pDC->TextOut(755, 580, "20");

else if (zombie\_amount == 21)

pDC->TextOut(755, 580, "21");

else if (zombie\_amount == 22)

pDC->TextOut(755, 580, "22");

else if (zombie\_amount == 23)

pDC->TextOut(755, 580, "23");

else {

pDC->TextOut(755, 580, "###");

}

pDC->SelectObject(fp);

CDDraw::ReleaseBackCDC();

}

}

}

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

mygame.h

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

#ifndef MYGAME\_H

#define MYGAME\_H

#include <memory>

#include <vector>

#include "YSun.h"

#include "YCard.h"

#include "YPlants.h"

#include "YMap.h"

#include "YZombies.h"

#include "YCar.h"

namespace game\_framework {

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

// Constants

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

enum AUDIO\_ID { // 定義各種音效的編號

AUDIO\_START,

AUDIO\_MAIN,

AUDIO\_MENUTOGAME,

AUDIO\_SUNPICK,

AUDIO\_PLANTS,

AUDIO\_CAR,

AUDIO\_START\_NIGHT,

AUDIO\_BOMB,

AUDIO\_VICTORY

};

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

// 這個class為遊戲的遊戲開頭畫面物件

// 每個Member function的Implementation都要弄懂

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

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

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

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

//void OnMouseMove(UINT nFlags, CPoint point);

//void OnMouseHover(UINT nFlags, CPoint point);

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap mainmenu; // mainmenu picture

CMovingBitmap loadtext; // Loading... picture

CMovingBitmap adventure0;

CMovingBitmap all\_level\_done;

};

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

// 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:

friend class YZombies;

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:

const int NUMBALLS; // 球的總數

CMovingBitmap background; // 背景圖

CMovingBitmap background\_night; // 背景圖

CMovingBitmap chooser;

YShovelCard shovel\_card;

int picX, picY;

int flag;

YSun sun;

int sun\_amount;

bool generateSunFlowerFlag;

bool generatePeaShooterFlag;

bool generateWallNutFlag;

bool generateCherryBombFlag;

bool generateIceShooterFlag;

bool generatePotatomineFlag;

bool generateShooterFlag;

bool generateSquashFlag;

bool shovelFlag;

YSunFlowerCard sun\_flower\_card;

YPeaShooterCard pea\_shooter\_card;

YWallNutCard wallnut\_card;

YCherryBombCard cherrybomb\_card;

YIceShooterCard ice\_shooter\_card;

YPotatomineCard potatomine\_card;

YShooterCard shooter\_card;

YSquashCard squash\_card;

int sun\_flower\_card\_delay\_flag;

int peashooter\_card\_delay\_flag;

int wallnut\_card\_delay\_flag;

int cherrybomb\_card\_delay\_flag;

int iceshooter\_card\_delay\_flag;

int potatomine\_card\_delay\_flag;

int shooter\_card\_delay\_flag;

int squash\_card\_delay\_flag;

std::vector<shared\_ptr<YSunFlower>> sunflower\_vector;

std::vector<shared\_ptr<YPeaShooter>> peashooter\_vector;

std::vector<shared\_ptr<YWallNut>> wallnut\_vector;

std::vector<shared\_ptr<YCherryBomb>> cherrybomb\_vector;

std::vector<shared\_ptr<YIceShooter>> iceshooter\_vector;

std::vector<shared\_ptr<YPotatoMine>> potatomine\_vector;

std::vector<shared\_ptr<YShooter>> shooter\_vector;

std::vector<shared\_ptr<YSquash>> squash\_vector;

std::vector<shared\_ptr<YNormalZombie>> normalzombie\_vector;

YMap map;

bool animation\_flag;

YCar car0;

YCar car1;

YCar car2;

YCar car3;

YCar car4;

bool car0\_flag;

bool car1\_flag;

bool car2\_flag;

bool car3\_flag;

bool car4\_flag;

bool car0\_sound\_flag;

bool car1\_sound\_flag;

bool car2\_sound\_flag;

bool car3\_sound\_flag;

bool car4\_sound\_flag;

bool zombie\_home\_flag;

bool zombie\_fast\_mode;

};

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

// 這個class為遊戲的結束狀態(Game Over)

// 每個Member function的Implementation都要弄懂

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

class CGameStateOver : public CGameState {

public:

CGameStateOver(CGame \*g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

void OnLButtonDown(UINT nFlags, CPoint point);

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 loose;

CMovingBitmap loose\_night;

CMovingBitmap newplant;

CMovingBitmap bucket;

CMovingBitmap cherrybomb;

CMovingBitmap conehead;

CMovingBitmap flag;

CMovingBitmap newspaper;

CMovingBitmap potatomine;

CMovingBitmap puff\_shroom;

CMovingBitmap snowpea;

CMovingBitmap wallnut;

CMovingBitmap squash;

CMovingBitmap victory;

};

}

#endif

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

YCar.h

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

#ifndef YCAR\_H

#define YCAR\_H

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

//#include <ddraw.h>

//#include "audio.h"

#include "gamelib.h"

namespace game\_framework {

class YCar {

public:

YCar() {

x = -20;

is\_alive = true;

}

YCar(int i) {

x = -20;

int a[5] = { 85, 182, 290, 390, 484 };

y = a[i]; // i: 0~4

is\_alive = true;

}

void LoadBitmap() {

car.LoadBitmap("Bitmaps/car.bmp", RGB(255,255,255));

}

void OnMove() {

if (x < 800)

x = x + 4;

else

is\_alive = false;

}

void OnShow() {

if (is\_alive) {

car.SetTopLeft(x, y + 20);

car.ShowBitmap();

}

}

bool IsAlive() {

return is\_alive;

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

int SetY(int i) {

int a[5] = { 85, 182, 290, 390, 484 };

this->y = a[i];

}

private:

int x;

int y;

bool is\_alive;

CMovingBitmap car;

};

}

#endif

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

YCard.h

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

#ifndef YCARD\_H

#define YCARD\_H

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

//#include <ddraw.h>

//#include "audio.h"

#include "gamelib.h"

namespace game\_framework {

class YSunFlowerCard

{

public:

YSunFlowerCard() {

x = 83;

y = 11;

is\_alive = false;

cost = 50;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_sunflower\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_sunflower\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(83, 11);

card\_die.SetTopLeft(83, 11);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YPeaShooterCard

{

public:

YPeaShooterCard() {

x = 137;

y = 11;

is\_alive = false;

cost = 100;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_peashooter\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_peashooter\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(137, 11);

card\_die.SetTopLeft(137, 11);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YWallNutCard

{

public:

YWallNutCard() {

x = 191;

y = 11;

is\_alive = false;

cost = 50;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_wallnut\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_wallnut\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(x, y);

card\_die.SetTopLeft(x, y);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YCherryBombCard

{

public:

YCherryBombCard() {

x = 191 + 54;

y = 11;

is\_alive = false;

cost = 150;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_cherrybomb\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_cherrybomb\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(x, y);

card\_die.SetTopLeft(x, y);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YIceShooterCard

{

public:

YIceShooterCard() {

x = 299;

y = 11;

is\_alive = false;

cost = 175;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_snowpea\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_snowpea\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(x, y);

card\_die.SetTopLeft(x, y);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YPotatomineCard

{

public:

YPotatomineCard() {

x = 299 + 54;

y = 11;

is\_alive = false;

cost = 25;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_potatomine\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_potatomine\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(x, y);

card\_die.SetTopLeft(x, y);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YSquashCard

{

public:

YSquashCard() {

x = 407;

y = 11;

is\_alive = false;

cost = 50;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_squash\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_squash\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(x, y);

card\_die.SetTopLeft(x, y);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YShooterCard

{

public:

YShooterCard() {

x = 407+54;

y = 11;

is\_alive = false;

cost = 0;

}

bool IsAlive() {

return is\_alive;

}

void LoadBitmap() {

card\_alive.LoadBitmap("Bitmaps/cards/card\_puffshroom\_alive.bmp");

card\_die.LoadBitmap("Bitmaps/cards/card\_puffshroom\_die.bmp");

}

void OnMove() {

}

void OnShow() {

card\_alive.SetTopLeft(x, y);

card\_die.SetTopLeft(x, y);

if (is\_alive == true) {

card\_alive.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetSunCost() {

return cost;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card\_alive;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

int cost;

};

class YShovelCard

{

public:

YShovelCard() {

x = 520;

y = 0;

is\_alive = true;

//cost = 0;

}

void LoadBitmap() {

card.LoadBitmap("Bitmaps/Shovel.bmp");

card\_die.LoadBitmap("Bitmaps/Shovel1.bmp");

}

void OnMove() {

}

void OnShow() {

card.SetTopLeft(520, 0);

card\_die.SetTopLeft(520, 0);

if (is\_alive == true) {

card.ShowBitmap();

}

else {

card\_die.ShowBitmap();

}

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

/\*int GetSunCost() {

return cost;

}\*/

int GetX() {

return x;

}

int GetY() {

return y;

}

private:

CMovingBitmap card;

CMovingBitmap card\_die;

int x, y;

bool is\_alive;

//int cost;

};

}

#endif

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

YMap.h

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

#ifndef YMAP\_H

#define YMAP\_H

namespace game\_framework {

class YMap {

private:

int mymap[9][5];

public:

YMap() :mymap{ 0 } {

}

void clear() {

for (int i = 0; i < 9; i++) {

for (int j = 0; j < 5; j++) {

mymap[i][j] = 0;

}

}

}

void setmyMap(int x, int y) {

if (x >= 100-70 && x < 190-70) {

if (y > 78 && y < 182) {

mymap[0][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[0][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[0][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[0][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[0][4] = 1;

}

}

else if (x >= 190-70 && x < 260-70) {

if (y > 78 && y < 182) {

mymap[1][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[1][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[1][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[1][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[1][4] = 1;

}

}

else if (x >= 260-70 && x < 340-70) {

if (y > 78 && y < 182) {

mymap[2][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[2][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[2][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[2][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[2][4] = 1;

}

}

else if (x >= 340-70 && x < 425-70) {

if (y > 78 && y < 182) {

mymap[3][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[3][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[3][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[3][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[3][4] = 1;

}

}

else if (x >= 425-70 && x < 505-70) {

if (y > 78 && y < 182) {

mymap[4][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[4][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[4][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[4][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[4][4] = 1;

}

}

else if (x >= 505-70 && x < 585-70) {

if (y > 78 && y < 182) {

mymap[5][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[5][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[5][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[5][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[5][4] = 1;

}

}

else if (x >= 585-70 && x < 660-70) {

if (y > 78 && y < 182) {

mymap[6][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[6][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[6][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[6][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[6][4] = 1;

}

}

else if (x >= 660-70 && x < 747-70) {

if (y > 78 && y < 182) {

mymap[7][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[7][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[7][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[7][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[7][4] = 1;

}

}

else if (x >= 747-70 && x < 840-70) {

if (y > 78 && y < 182) {

mymap[8][0] = 1;

}

else if (y >= 182 && y < 270) {

mymap[8][1] = 1;

}

else if (y >= 270 && y < 368) {

mymap[8][2] = 1;

}

else if (y >= 368 && y < 464) {

mymap[8][3] = 1;

}

else if (y >= 464 && y < 571) {

mymap[8][4] = 1;

}

}

}

void unsetmyMap(int x, int y) {

if (x >= 100-70 && x < 190-70) {

if (y > 78 && y < 182) {

mymap[0][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[0][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[0][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[0][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[0][4] = 0;

}

}

else if (x >= 190-70 && x < 260-70) {

if (y > 78 && y < 182) {

mymap[1][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[1][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[1][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[1][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[1][4] = 0;

}

}

else if (x >= 260-70 && x < 340-70) {

if (y > 78 && y < 182) {

mymap[2][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[2][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[2][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[2][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[2][4] = 0;

}

}

else if (x >= 340-70 && x < 425-70) {

if (y > 78 && y < 182) {

mymap[3][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[3][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[3][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[3][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[3][4] = 0;

}

}

else if (x >= 425-70 && x < 505-70) {

if (y > 78 && y < 182) {

mymap[4][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[4][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[4][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[4][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[4][4] = 0;

}

}

else if (x >= 505-70 && x < 585-70) {

if (y > 78 && y < 182) {

mymap[5][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[5][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[5][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[5][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[5][4] = 0;

}

}

else if (x >= 585-70 && x < 660-70) {

if (y > 78 && y < 182) {

mymap[6][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[6][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[6][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[6][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[6][4] = 0;

}

}

else if (x >= 660-70 && x < 747-70) {

if (y > 78 && y < 182) {

mymap[7][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[7][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[7][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[7][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[7][4] = 0;

}

}

else if (x >= 747-70 && x < 840-70) {

if (y > 78 && y < 182) {

mymap[8][0] = 0;

}

else if (y >= 182 && y < 270) {

mymap[8][1] = 0;

}

else if (y >= 270 && y < 368) {

mymap[8][2] = 0;

}

else if (y >= 368 && y < 464) {

mymap[8][3] = 0;

}

else if (y >= 464 && y < 571) {

mymap[8][4] = 0;

}

}

}

bool checkmyMap(const int x, const int y) {

if (x >= 100-70 && x < 190-70) {

if (y > 78 && y < 182) {

if (mymap[0][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[0][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[0][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[0][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[0][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 190-70 && x < 260-70) {

if (y > 78 && y < 182) {

if (mymap[1][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[1][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[1][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[1][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[1][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 260-70 && x < 340-70) {

if (y > 78 && y < 182) {

if (mymap[2][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[2][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[2][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[2][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[2][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 340-70 && x < 425-70) {

if (y > 78 && y < 182) {

if (mymap[3][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[3][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[3][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[3][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[3][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 425-70 && x < 505-70) {

if (y > 78 && y < 182) {

if (mymap[4][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[4][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[4][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[4][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[4][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 505-70 && x < 585-70) {

if (y > 78 && y < 182) {

if (mymap[5][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[5][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[5][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[5][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[5][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 585-70 && x < 660-70) {

if (y > 78 && y < 182) {

if (mymap[6][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[6][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[6][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[6][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[6][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 660-70 && x < 747-70) {

if (y > 78 && y < 182) {

if (mymap[7][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[7][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[7][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[7][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[7][4] == 1)

return true;

return false;

}

return false;

}

else if (x >= 747-70 && x < 840-70) {

if (y > 78 && y < 182) {

if (mymap[8][0] == 1)

return true;

return false;

}

else if (y >= 182 && y < 270) {

if (mymap[8][1] == 1)

return true;

return false;

}

else if (y >= 270 && y < 368) {

if (mymap[8][2] == 1)

return true;

return false;

}

else if (y >= 368 && y < 464) {

if (mymap[8][3] == 1)

return true;

return false;

}

else if (y >= 464 && y < 571) {

if (mymap[8][4] == 1)

return true;

return false;

}

return false;

}

return false;

}

int getXmyMapLocation(int x, int y) {

if (x >= 100-70 && x < 190-70) {

return 100-70;

}

else if (x >= 190-70 && x < 260-70) {

return 190-70;

}

else if (x >= 260-70 && x < 340-70) {

return 260-70;

}

else if (x >= 340-70 && x < 425-70) {

return 340-70;

}

else if (x >= 425-70 && x < 505-70) {

return 425-70;

}

else if (x >= 505-70 && x < 585-70) {

return 505-70;

}

else if (x >= 585-70 && x < 660-70) {

return 585-70;

}

else if (x >= 660-70 && x < 747-70) {

return 660-70;

}

else if (x >= 747-70 && x < 840-70) {

return 747-70;

}

return 0;

}

int getYmyMapLocation(int x, int y) {

if (y >= 78 && y < 182)

{

return 78;

}

else if (y >= 182 && y < 270)

{

return 182;

}

else if (y >= 270 && y < 368)

{

return 270;

}

else if (y >= 368 && y < 464)

{

return 368;

}

else if (y >= 464 && y < 571)

{

return 464;

}

return 0;

}

};

}

#endif

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

YPlant.h

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

#ifndef YPLANTS\_H

#define YPLANTS\_H

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include "gamelib.h"

#include "YSun.h"

#include "YZombies.h"

namespace game\_framework {

class YSunFlower

{

public:

YSunFlower(int x, int y) {

this->x = x + 8;

this->y = y + 13;

blood = 300;

sun\_make\_time = 60 \* 5;

is\_alive = true;

YSun temp(this->x, this->y);

sun = temp;

sun.LoadBitmap();

}

~YSunFlower() {

}

void LoadBitmap() {

char \*filename[5] = { ".\\bitmaps\\SunFlower\\SunFlower\_0.bmp"

,".\\bitmaps\\SunFlower\\SunFlower\_1.bmp"

, ".\\bitmaps\\SunFlower\\SunFlower\_2.bmp"

, ".\\bitmaps\\SunFlower\\SunFlower\_4.bmp"

,".\\bitmaps\\SunFlower\\SunFlower\_7.bmp"

};

for (int i = 0; i < 5; i++)

sun\_flower\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

}

void OnMove() {

if (sun\_make\_time > 0) {

sun\_make\_time--;

}

else {

sun.OnMove();

sun.SetIsAlive(true);

}

sun\_flower\_animation.OnMove();

}

void OnShow() {

if (IsAlive()) {

sun\_flower\_animation.SetTopLeft(x, y);

sun\_flower\_animation.OnShow();

}

if (sun\_make\_time > 0) {

sun\_make\_time--;

}

else {

sun.OnShow();

}

}

bool checkPlantCollideWithZombie(int zx, int zy) {

if (zy == y - 13) {

if (x > zx - 72 && x < zx - 48) {

return true;

}

}

return false;

}

bool IsAlive() {

return is\_alive;

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

int GetX() {

return x;

}

int GetY() {

return y;

}

bool GetSunIsAlive() {

return sun.IsAlive();

}

int GetSunX() {

return sun.GetSunFlowerSunX();

}

int GetSunY() {

return sun.GetSunFlowerSunY();

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

int GetBlood() {

return blood;

}

void initSun() {

sun.SetIsAlive(false);

sun\_make\_time = 60 \* 7;

YSun temp(this->x, this->y);

sun = temp;

sun.LoadBitmap();

}

private:

int x, y;

bool is\_alive;

int blood;

int sun\_make\_time;

CAnimation sun\_flower\_animation;

YSun sun;

};

class YPeaShooterBullet {

public:

YPeaShooterBullet(int x, int y) {

this->x = x + 30;

this->y = y;

is\_alive = true;

}

void LoadBitmap()

{

peashooter\_bullet.LoadBitmap(".\\bitmaps\\PeaShooter\\PeaNormal\_0.bmp", RGB(255, 255, 255));

}

void OnMove()

{

x += 7;

if (x > 800) {

is\_alive = false;

};

}

void OnShow()

{

if (is\_alive)

{

peashooter\_bullet.SetTopLeft(x, y);

peashooter\_bullet.ShowBitmap();

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

private:

int x, y;

bool is\_alive;

CMovingBitmap peashooter\_bullet;

};

class YPeaShooter

{

public:

YPeaShooter(int x, int y)

{

delay = 30;

this->x = x + 20;

this->y = y + 20;

blood = 300;

is\_alive = true;

}

~YPeaShooter()

{

}

void LoadBitmap()

{

char \*filename[5] = { ".\\bitmaps\\PeaShooter\\PeaShooter\_1.bmp", ".\\bitmaps\\PeaShooter\\PeaShooter\_2.bmp", ".\\bitmaps\\PeaShooter\\PeaShooter\_3.bmp", ".\\bitmaps\\PeaShooter\\PeaShooter\_4.bmp", ".\\bitmaps\\PeaShooter\\PeaShooter\_5.bmp" };

for (int i = 0; i < 5; i++)

peashooter\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

}

void OnMove()

{

peashooter\_animation.OnMove();

if (delay == 0) {

fireBullet();

delay = 80;

}

delay--;

for (size\_t i = 0; i < bullets\_vector.size(); i++) {

if (bullets\_vector.at(i)->IsAlive())

bullets\_vector.at(i)->OnMove();

else {

bullets\_vector.erase(bullets\_vector.begin() + i);

}

}

}

void OnShow()

{

if (is\_alive)

{

peashooter\_animation.SetTopLeft(x, y);

peashooter\_animation.OnShow();

}

for (size\_t i = 0; i < bullets\_vector.size(); i++) {

if (bullets\_vector.at(i)->IsAlive())

bullets\_vector.at(i)->OnShow();

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

void SetBlood(int attack\_blood)

{

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

int GetBlood()

{

return blood;

}

void fireBullet() {

auto sp = make\_shared<YPeaShooterBullet>(x, y);

sp->LoadBitmap();

bullets\_vector.push\_back(sp);

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

bool checkBulletCollideWithZombie(int zx, int mapy) {

int by;

if (!bullets\_vector.empty()) {

by = bullets\_vector.at(0)->GetY() - 20;

}

if (!bullets\_vector.empty() && bullets\_vector.at(0)->GetY() - 20 == mapy) {

int t = bullets\_vector.at(0)->GetX();

if (bullets\_vector.at(0)->GetX() > zx + 20 && bullets\_vector.at(0)->GetX() < zx + 80) {

bullets\_vector.at(0)->SetIsAlive(false);

return true;

}

}

return false;

}

bool checkPlantCollideWithZombie(int zx, int zy) {

if (zy == y - 20) {

if (x > zx - 60 && x < zx - 35) {

return true;

}

}

return false;

}

private:

int x, y;

bool is\_alive;

int blood;

std::vector<shared\_ptr<YPeaShooterBullet>> bullets\_vector;

CAnimation peashooter\_animation;

int delay;

};

class YWallNut

{

public:

YWallNut(int x, int y)

{

this->x = x + 20;

this->y = y + 20;

blood = 600;

is\_alive = true;

}

~YWallNut()

{

}

void LoadBitmap()

{

char \*filename[4] = { ".\\bitmaps\\WallNut\\WallNut\\WallNut\_0.bmp", ".\\bitmaps\\WallNut\\WallNut\\WallNut\_2.bmp", ".\\bitmaps\\WallNut\\WallNut\\WallNut\_3.bmp",

".\\bitmaps\\WallNut\\WallNut\\WallNut\_4.bmp" };

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

wallnut\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

char \*filename1[4] = { ".\\bitmaps\\WallNut\\WallNut\_cracked1\\WallNut\_cracked1\_0.bmp", ".\\bitmaps\\WallNut\\WallNut\_cracked1\\WallNut\_cracked1\_2.bmp", ".\\bitmaps\\WallNut\\WallNut\_cracked1\\WallNut\_cracked1\_3.bmp",

".\\bitmaps\\WallNut\\WallNut\_cracked1\\WallNut\_cracked1\_4.bmp"};

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

wallnut\_cracked1\_animation.AddBitmap(filename1[i], RGB(255, 255, 255));

char \*filename2[4] = { ".\\bitmaps\\WallNut\\WallNut\_cracked2\\WallNut\_cracked2\_0.bmp", ".\\bitmaps\\WallNut\\WallNut\_cracked2\\WallNut\_cracked2\_2.bmp", ".\\bitmaps\\WallNut\\WallNut\_cracked2\\WallNut\_cracked2\_3.bmp",

".\\bitmaps\\WallNut\\WallNut\_cracked2\\WallNut\_cracked2\_4.bmp" };

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

wallnut\_cracked2\_animation.AddBitmap(filename2[i], RGB(255, 255, 255));

}

void OnMove()

{

if (blood > 400) {

wallnut\_animation.OnMove();

}

else if (blood > 200) {

wallnut\_cracked1\_animation.OnMove();

}

else {

wallnut\_cracked2\_animation.OnMove();

}

}

void OnShow()

{

if (is\_alive)

{

if (blood > 400) {

wallnut\_animation.SetTopLeft(x, y);

wallnut\_animation.OnShow();

}

else if (blood > 200) {

wallnut\_cracked1\_animation.SetTopLeft(x, y);

wallnut\_cracked1\_animation.OnShow();

}

else {

wallnut\_cracked2\_animation.SetTopLeft(x, y);

wallnut\_cracked2\_animation.OnShow();

}

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

int GetBlood()

{

return blood;

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

bool checkPlantCollideWithZombie(int zx, int zy) {

if (zy == y - 20) {

if (x > zx - 60 && x < zx - 35) {

return true;

}

}

return false;

}

private:

int x, y;

bool is\_alive;

int blood;

CAnimation wallnut\_animation;

CAnimation wallnut\_cracked1\_animation;

CAnimation wallnut\_cracked2\_animation;

};

class YCherryBomb

{

public:

YCherryBomb(int x, int y)

{

this->x = x;

this->y = y;

is\_alive = true;

bomb = false;

counter = 65;

}

~YCherryBomb()

{

}

void LoadBitmap()

{

char \*filename[7] = { ".\\bitmaps\\CherryBomb\\CherryBomb\_0.bmp", ".\\bitmaps\\CherryBomb\\CherryBomb\_1.bmp", ".\\bitmaps\\CherryBomb\\CherryBomb\_2.bmp",

".\\bitmaps\\CherryBomb\\CherryBomb\_3.bmp", ".\\bitmaps\\CherryBomb\\CherryBomb\_4.bmp",

".\\bitmaps\\CherryBomb\\Boom.bmp", ".\\bitmaps\\CherryBomb\\Boom.bmp" };

for (int i = 0; i < 7; i++)

cherrybomb\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

}

void OnMove()

{

if (counter == 22) {

bomb = true;

}

else if (counter < 0) {

is\_alive = false;

}

cherrybomb\_animation.OnMove();

counter -= 1;

}

void OnShow()

{

if (!bomb)

{

cherrybomb\_animation.SetTopLeft(x - 55, y - 40);

cherrybomb\_animation.OnShow();

}

else {

cherrybomb\_animation.SetTopLeft(x - 85, y - 64);

cherrybomb\_animation.OnShow();

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

bool Bomb() {

return bomb;

}

bool checkNearbyZombies(int zx, int zy) {

if (y - 160 < zy && zy < y + 80) {

if (zx > x - 160 && zx < x + 120) {

return true;

}

}

return false;

}

private:

int x, y;

bool is\_alive;

CAnimation cherrybomb\_animation;

int counter;

int bomb;

};

class YIceShooterBullet {

public:

YIceShooterBullet(int x, int y) {

this->x = x + 30;

this->y = y;

is\_alive = true;

}

void LoadBitmap()

{

iceshooter\_bullet.LoadBitmap(".\\bitmaps\\IceShooter\\PeaIce\_0.bmp", RGB(255, 255, 255));

}

void OnMove()

{

x += 7;

if (x > 800) {

is\_alive = false;

};

}

void OnShow()

{

if (is\_alive)

{

iceshooter\_bullet.SetTopLeft(x, y);

iceshooter\_bullet.ShowBitmap();

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

private:

int x, y;

bool is\_alive;

CMovingBitmap iceshooter\_bullet;

};

class YIceShooter

{

public:

YIceShooter(int x, int y)

{

delay = 30;

this->x = x + 20;

this->y = y + 20;

blood = 450;

is\_alive = true;

}

~YIceShooter()

{

}

void LoadBitmap()

{

char \*filename[5] = { ".\\bitmaps\\IceShooter\\SnowPea\_1.bmp", ".\\bitmaps\\IceShooter\\SnowPea\_2.bmp", ".\\bitmaps\\IceShooter\\SnowPea\_3.bmp", ".\\bitmaps\\IceShooter\\SnowPea\_4.bmp", ".\\bitmaps\\IceShooter\\SnowPea\_5.bmp" };

for (int i = 0; i < 5; i++)

iceshooter\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

}

void OnMove()

{

iceshooter\_animation.OnMove();

if (delay == 0) {

fireBullet();

delay = 80;

}

delay--;

for (size\_t i = 0; i < bullets\_vector.size(); i++) {

if (bullets\_vector.at(i)->IsAlive())

bullets\_vector.at(i)->OnMove();

else {

bullets\_vector.erase(bullets\_vector.begin() + i);

}

}

}

void OnShow()

{

if (is\_alive)

{

iceshooter\_animation.SetTopLeft(x, y);

iceshooter\_animation.OnShow();

}

for (size\_t i = 0; i < bullets\_vector.size(); i++) {

if (bullets\_vector.at(i)->IsAlive())

bullets\_vector.at(i)->OnShow();

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

void SetBlood(int attack\_blood)

{

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

int GetBlood()

{

return blood;

}

void fireBullet() {

auto sp = make\_shared<YIceShooterBullet>(x, y);

sp->LoadBitmap();

bullets\_vector.push\_back(sp);

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

bool checkBulletCollideWithZombie(int zx, int mapy) {

if (!bullets\_vector.empty() && bullets\_vector.at(0)->GetY() - 20 == mapy) {

int t = bullets\_vector.at(0)->GetX();

if (bullets\_vector.at(0)->GetX() > zx + 20 && bullets\_vector.at(0)->GetX() < zx + 80) {

bullets\_vector.at(0)->SetIsAlive(false);

return true;

}

}

return false;

}

bool checkPlantCollideWithZombie(int zx, int zy) {

if (zy == y - 20) {

if (x > zx - 60 && x < zx - 35) {

return true;

}

}

return false;

}

private:

int x, y;

bool is\_alive;

int blood;

std::vector<shared\_ptr<YIceShooterBullet>> bullets\_vector;

CAnimation iceshooter\_animation;

int delay;

};

class YPotatoMine

{

public:

YPotatoMine(int x, int y)

{

this->x = x + 10;

this->y = y + 20;

is\_alive = true;

bomb = 0;

counter = 0;

blood = 300;

zombie\_checked = false;

}

~YPotatoMine()

{

}

void LoadBitmap()

{

char \*filename[8] = { ".\\bitmaps\\PotatoMine\\PotatoMine\_0.bmp",

".\\bitmaps\\PotatoMine\\PotatoMine\_1.bmp", ".\\bitmaps\\PotatoMine\\PotatoMine\_2.bmp",

".\\bitmaps\\PotatoMine\\PotatoMine\_3.bmp", ".\\bitmaps\\PotatoMine\\PotatoMine\_4.bmp",

".\\bitmaps\\PotatoMine\\PotatoMine\_5.bmp",".\\bitmaps\\PotatoMine\\PotatoMine\_6.bmp",

".\\bitmaps\\PotatoMine\\PotatoMine\_7.bmp" };

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

potatomine\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

char \*filenamee[5] = { ".\\bitmaps\\PotatoMine\\PotatoMineExplode\_0.bmp",

".\\bitmaps\\PotatoMine\\PotatoMineExplode\_0.bmp",".\\bitmaps\\PotatoMine\\PotatoMineExplode\_0.bmp",

".\\bitmaps\\PotatoMine\\PotatoMineExplode\_0.bmp" ,".\\bitmaps\\PotatoMine\\PotatoMineExplode\_0.bmp" };

for (int i = 0; i < 5; i++)

potatomine\_explode\_animation.AddBitmap(filenamee[i], RGB(255, 255, 255));

char \*filenamei[2] = { ".\\bitmaps\\PotatoMine\\PotatoMineInit\_0.bmp",

".\\bitmaps\\PotatoMine\\PotatoMineInit\_0.bmp" };

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

potatomine\_init\_animation.AddBitmap(filenamei[i], RGB(255, 255, 255));

}

void OnMove()

{

if (counter < 700) {

counter += 1;

potatomine\_init\_animation.OnMove();

}

else if (bomb > 10) {

is\_alive = false;

}

else if (counter > 705 && zombie\_checked) {

counter += 1;

bomb += 1;

potatomine\_explode\_animation.OnMove();

}

else {

counter += 1;

potatomine\_animation.OnMove();

}

}

void SetZombieChecked() {

zombie\_checked = true;

}

void OnShow()

{

if (counter < 700) {

potatomine\_init\_animation.SetTopLeft(x, y);

potatomine\_init\_animation.OnShow();

}

else if (bomb) {

potatomine\_explode\_animation.SetTopLeft(x, y);

potatomine\_explode\_animation.OnShow();

}

else {

potatomine\_animation.SetTopLeft(x, y);

potatomine\_animation.OnShow();

}

}

int GetBlood()

{

return blood;

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

bool checkPlantCollideWithZombie(int zx, int zy) {

if (zy == y - 20) {

if (x > zx - 75 && x < zx - 30) {

return true;

}

}

return false;

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

bool Bomb() {

return bomb;

}

bool checkNearbyZombies(int zx, int zy) {

if (y - 160 < zy && zy < y + 80) {

if (x - 160 < zx && zx < x + 140) {

return true;

}

}

return false;

}

private:

int x, y;

bool is\_alive;

CAnimation potatomine\_animation;

CAnimation potatomine\_explode\_animation;

CAnimation potatomine\_init\_animation;

int blood;

int counter;

int bomb;

bool zombie\_checked;

};

class YSquash

{

public:

YSquash(int x, int y)

{

this->x = x + 10;

this->y = y + 20;

is\_alive = true;

bomb = 0;

counter = 0;

blood = 300;

zombie\_checked = false;

}

~YSquash()

{

}

void LoadBitmap()

{

char \*filename[8] = { ".\\bitmaps\\Squash\\Squash\_0.bmp",

".\\bitmaps\\Squash\\Squash\_1.bmp", ".\\bitmaps\\Squash\\Squash\_2.bmp",

".\\bitmaps\\Squash\\Squash\_3.bmp", ".\\bitmaps\\Squash\\Squash\_5.bmp",

".\\bitmaps\\Squash\\Squash\_7.bmp", ".\\bitmaps\\Squash\\SquashAim\_0.bmp",

".\\bitmaps\\Squash\\SquashAim\_0.bmp" };

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

squash\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

char \*filenamee[5] = { ".\\bitmaps\\Squash\\SquashAttack\_0.bmp",

".\\bitmaps\\Squash\\SquashAttack\_1.bmp",".\\bitmaps\\Squash\\SquashAttack\_2.bmp",

".\\bitmaps\\Squash\\SquashAttack\_3.bmp" ,".\\bitmaps\\Squash\\SquashAttack\_3.bmp" };

for (int i = 0; i < 5; i++)

squash\_attack\_animation.AddBitmap(filenamee[i], RGB(255, 255, 255));

}

void OnMove()

{

if (zombie\_checked) {

counter += 1;

bomb += 1;

squash\_attack\_animation.OnMove();

if (counter == 35) {

is\_alive = false;

}

}

else {

squash\_animation.OnMove();

}

}

void SetZombieChecked() {

zombie\_checked = true;

}

void OnShow()

{

if (zombie\_checked) {

squash\_attack\_animation.SetTopLeft(x - 10, y - 140);

squash\_attack\_animation.OnShow();

}

else {

squash\_animation.SetTopLeft(x - 10, y - 140);

squash\_animation.OnShow();

}

}

int GetBlood()

{

return blood;

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

bool checkPlantCollideWithZombie(int zx, int zy) {

if (zy == y - 20) {

if (x > zx - 75 && x < zx - 30) {

return true;

}

}

return false;

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

bool Bomb() {

return bomb;

}

bool checkNearbyZombies(int zx, int zy) {

if (y - 160 < zy && zy < y + 80) {

if (x - 160 < zx && zx < x + 140) {

return true;

}

}

return false;

}

private:

int x, y;

bool is\_alive;

CAnimation squash\_animation;

CAnimation squash\_attack\_animation;

int blood;

int counter;

int bomb;

bool zombie\_checked;

};

class YShooterBullet {

public:

YShooterBullet(int x, int y) {

this->x = x + 30;

this->y = y;

is\_alive = true;

}

void LoadBitmap()

{

shooter\_bullet.LoadBitmap(".\\bitmaps\\Shroom\\BulletMushRoom\_0.bmp", RGB(255, 255, 255));

}

void OnMove()

{

x += 6;

if (x > 800) {

is\_alive = false;

};

}

void OnShow()

{

if (is\_alive)

{

shooter\_bullet.SetTopLeft(x, y + 25);

shooter\_bullet.ShowBitmap();

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

private:

int x, y;

bool is\_alive;

CMovingBitmap shooter\_bullet;

};

class YShooter

{

public:

YShooter(int x, int y)

{

delay = 30;

this->x = x + 20;

this->y = y + 20;

blood = 450;

is\_alive = true;

close = false;

}

~YShooter()

{

}

void LoadBitmap()

{

char \*filename[4] = { ".\\bitmaps\\Shroom\\PuffShroom\_0.bmp", ".\\bitmaps\\Shroom\\PuffShroom\_2.bmp", ".\\bitmaps\\Shroom\\PuffShroom\_3.bmp", ".\\bitmaps\\Shroom\\PuffShroom\_4.bmp" };

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

shooter\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

}

void OnMove()

{

shooter\_animation.OnMove();

if (delay == 0) {

fireBullet();

delay = 80;

}

delay--;

for (size\_t i = 0; i < bullets\_vector.size(); i++) {

if (bullets\_vector.at(i)->IsAlive())

bullets\_vector.at(i)->OnMove();

else {

bullets\_vector.erase(bullets\_vector.begin() + i);

}

}

}

void OnShow()

{

if (is\_alive)

{

shooter\_animation.SetTopLeft(x, y + 10);

shooter\_animation.OnShow();

}

for (size\_t i = 0; i < bullets\_vector.size(); i++) {

if (bullets\_vector.at(i)->IsAlive())

bullets\_vector.at(i)->OnShow();

}

}

bool IsAlive()

{

return is\_alive;

}

void SetIsAlive(bool alive)

{

is\_alive = alive;

}

int GetX()

{

return x;

}

int GetY()

{

return y;

}

void SetBlood(int attack\_blood)

{

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

int GetBlood()

{

return blood;

}

void fireBullet() {

if (close) {

auto sp = make\_shared<YShooterBullet>(x, y);

sp->LoadBitmap();

bullets\_vector.push\_back(sp);

}

}

bool isClose(int zx, int zy) {

if (zy == y - 20) {

if (x + 300 > zx && zx > x - 60) {

return true;

}

}

return false;

}

void SetClose(bool is\_close)

{

close = is\_close;

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

bool checkBulletCollideWithZombie(int zx, int mapy) {

if (!bullets\_vector.empty() && bullets\_vector.at(0)->GetY() - 20 == mapy) {

int t = bullets\_vector.at(0)->GetX();

if (bullets\_vector.at(0)->GetX() > zx + 20 && bullets\_vector.at(0)->GetX() < zx + 80) {

bullets\_vector.at(0)->SetIsAlive(false);

return true;

}

}

return false;

}

bool checkPlantCollideWithZombie(int zx, int zy) {

if (zy == y - 20) {

if (x > zx - 60 && x < zx - 35) { // zx - 35 > x > zx - 60

return true;

}

}

return false;

}

private:

int x, y;

bool is\_alive;

int blood;

std::vector<shared\_ptr<YShooterBullet>> bullets\_vector;

CAnimation shooter\_animation;

int delay;

bool close;

};

}

#endif

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

YSun.cpp

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

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

//#include "audio.h"

#include "gamelib.h"

#include "YSun.h"

namespace game\_framework {

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

// YSun: Sun class

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

YSun::YSun() {

srand((int)time(NULL));

x = (rand() % 550) + 100; // 100~650

y = -150;

floor = (rand() % 250) + 250; //250~500

is\_alive = true;

sunflowerflag = false;

}

YSun::YSun(int x, int y) {

this->x = x;

this->y = y;

floor = y + 45;

is\_alive = false;

sunflowerflag = true;

rising = true;

velocity = 10;

}

YSun & YSun::operator= (const YSun & other) {

if (this != &other) {

this->x = other.x;

this->y = other.y;

this->floor = other.floor;

this->is\_alive = other.is\_alive;

this->sunflowerflag = other.sunflowerflag;

this->velocity = other.velocity;

this->rising = other.rising;

}

return \*this;

}

bool YSun::IsAlive() {

return is\_alive;

}

void YSun::LoadBitmap() {

char \*filename[5] = { ".\\bitmaps\\Sun\\sun0.bmp",".\\bitmaps\\Sun\\sun1.bmp",".\\bitmaps\\Sun\\sun2.bmp",".\\bitmaps\\Sun\\sun3.bmp"

, ".\\bitmaps\\Sun\\sun4.bmp" };

for (int i = 0; i < 5; i++)

sunanimation.AddBitmap(filename[i], RGB(255, 255, 255));

}

void YSun::OnMove() {

if (sunflowerflag) {

if (rising) { // 上升狀態

if (velocity > 0) {

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

velocity--; // 受重力影響，下次的上升速度降低

x += 1;

}

else {

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

velocity = 1; // 下降的初速(velocity)為1

x += 1;

}

}

else { // 下降狀態

if (y < floor - 1) { // 當y座標還沒碰到地板

y += velocity; // y軸下降(移動velocity個點，velocity的單位為 點/次)

velocity++; // 受重力影響，下次的下降速度增加

x += 1;

}

}

}

if (!sunflowerflag && GetY() > 0) {

SetIsAlive(true);

}

if (!sunflowerflag && GetY() < GetFloor())

y = y + 2;

sunanimation.OnMove();

}

void YSun::OnShow() {

if (IsAlive()) {

sunanimation.SetTopLeft(x, y);

sunanimation.OnShow();

}

}

void YSun::SetIsAlive(bool alive) {

is\_alive = alive;

}

int YSun::GetSunFlowerSunX() {

return x;

}

int YSun::GetSunFlowerSunY() {

return y;

}

void YSun::SetY(int y) {

srand((int)time(NULL));

x = (rand() % 550) + 200; // 200~750

this->y = y;

floor = (rand() % 250) + 250; //250~500

}

int YSun::GetX() {

return x;

}

int YSun::GetY() {

return y;

}

int YSun::GetFloor() {

return floor;

}

}

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

YSun.h

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

#ifndef YSUN\_H

#define YSUN\_H

namespace game\_framework {

class YSun

{

public:

YSun();

YSun(int x, int y);

YSun & operator= (const YSun & other);

bool IsAlive(); // 是否活著

void LoadBitmap(); // 載入圖形

void OnMove(); // 移動

void OnShow(); // 將圖形貼到畫面

void SetIsAlive(bool alive); // 設定是否活著

int GetSunFlowerSunX();

int GetSunFlowerSunY();

void SetY(int y);

int GetX();

int GetY();

int GetFloor();

private:

//CMovingBitmap sun;

int x, y;

bool is\_alive;

CAnimation sunanimation; // 利用動畫作圖形

int floor; // where sun stop

bool sunflowerflag;

int velocity;

bool rising;

};

}

#endif

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

YZombie.h

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

#ifndef YZOMBIES\_H

#define YZOMBIES\_H

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include "gamelib.h"

#include "YPlants.h"

#include "YMap.h"

namespace game\_framework {

class YNormalZombie {

public:

YNormalZombie() {

srand((int)time(NULL));

int i = rand() % 5; // 0~4

Map\_Y\_Location = i; // 0~4

int a[5] = { 78, 182, 270, 368, 464 };

y = a[i] - 30;

x = 900;

blood = 10;

is\_alive = true;

zombie\_die\_delay\_time = 115;

}

YNormalZombie(int x, int my) {

Map\_Y\_Location = my; // 0~4

int a[5] = { 78, 182, 270, 368, 464 };

y = a[my] - 30;

this->x = x;

blood = 10;

is\_alive = true;

bomb\_flag = false;

zombie\_die\_delay\_time = 115;

zombie\_bomb\_die\_delay\_time = 160;

}

YNormalZombie(int x, int my, std::string style) {

Map\_Y\_Location = my; // 0~4

int a[5] = { 78, 182, 270, 368, 464 };

y = a[my] - 30;

this->x = x;

blood = 10;

is\_alive = true;

bomb\_flag = false;

zombie\_die\_delay\_time = 115;

zombie\_bomb\_die\_delay\_time = 160;

zombie\_style = style;

if (zombie\_style == "conehead") {

blood = 20;

}

if (zombie\_style == "bucket") {

blood = 30;

}

if (zombie\_style == "newspaper") {

blood = 20;

}

}

~YNormalZombie() {

}

void LoadBitmap() {

if (zombie\_style == "newspaper") {

char \*filenameni[6] = { ".\\bitmaps\\NormalZombie\\NewspaperZombie\\NewspaperZombie\_0.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombie\\NewspaperZombie\_1.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombie\\NewspaperZombie\_3.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombie\\NewspaperZombie\_5.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombie\\NewspaperZombie\_7.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombie\\NewspaperZombie\_9.bmp" };

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

zombie\_news\_animation.AddBitmap(filenameni[i], RGB(255, 255, 255));

char \*filenamen[5] = { ".\\bitmaps\\NormalZombie\\NewspaperZombieAttack\\NewspaperZombieAttack\_0.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieAttack\\NewspaperZombieAttack\_1.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieAttack\\NewspaperZombieAttack\_3.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieAttack\\NewspaperZombieAttack\_5.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieAttack\\NewspaperZombieAttack\_7.bmp" };

for (int i = 0; i < 5; i++)

zombie\_news\_attack\_animation.AddBitmap(filenamen[i], RGB(255, 255, 255));

char \*filenamena[11] = { ".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_0.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_1.bmp", ".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_2.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_3.bmp", ".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_4.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_5.bmp", ".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_6.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_7.bmp", ".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_8.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_9.bmp", ".\\bitmaps\\NormalZombie\\NewspaperZombieDie\\NewspaperZombieDie\_10.bmp" };

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

zombie\_news\_die\_animation.AddBitmap(filenamena[i], RGB(255, 255, 255));

char \*filenamene[6] = { ".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaper\\NewspaperZombieNoPaper\_0.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaper\\NewspaperZombieNoPaper\_1.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaper\\NewspaperZombieNoPaper\_3.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaper\\NewspaperZombieNoPaper\_5.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaper\\NewspaperZombieNoPaper\_7.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaper\\NewspaperZombieNoPaper\_9.bmp" };

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

zombie\_nonews\_animation.AddBitmap(filenamene[i], RGB(255, 255, 255));

char \*filenameno[6] = { ".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaperAttack\\NewspaperZombieNoPaperAttack\_0.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaperAttack\\NewspaperZombieNoPaperAttack\_1.bmp", ".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaperAttack\\NewspaperZombieNoPaperAttack\_2.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaperAttack\\NewspaperZombieNoPaperAttack\_3.bmp", ".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaperAttack\\NewspaperZombieNoPaperAttack\_4.bmp",

".\\bitmaps\\NormalZombie\\NewspaperZombieNoPaperAttack\\NewspaperZombieNoPaperAttack\_5.bmp" };

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

zombie\_nonews\_attack\_animation.AddBitmap(filenameno[i], RGB(255, 255, 255));

}

else if (zombie\_style == "bucket") {

char \*filenameb[6] = { ".\\bitmaps\\NormalZombie\\BucketheadZombie\\BucketheadZombie\_0.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombie\\BucketheadZombie\_1.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombie\\BucketheadZombie\_3.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombie\\BucketheadZombie\_5.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombie\\BucketheadZombie\_7.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombie\\BucketheadZombie\_9.bmp" };

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

zombie\_bucket\_animation.AddBitmap(filenameb[i], RGB(255, 255, 255));

char \*filenameba[5] = {

".\\bitmaps\\NormalZombie\\BucketheadZombieAttack\\BucketheadZombieAttack\_1.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombieAttack\\BucketheadZombieAttack\_3.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombieAttack\\BucketheadZombieAttack\_5.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombieAttack\\BucketheadZombieAttack\_7.bmp",

".\\bitmaps\\NormalZombie\\BucketheadZombieAttack\\BucketheadZombieAttack\_9.bmp" };

for (int i = 0; i < 5; i++)

zombie\_bucket\_attack\_animation.AddBitmap(filenameba[i], RGB(255, 255, 255));

}

else if (zombie\_style == "conehead") {

char \*filenamec[5] = {

".\\bitmaps\\NormalZombie\\ConeheadZombie\\ConeheadZombie\_1.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombie\\ConeheadZombie\_3.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombie\\ConeheadZombie\_5.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombie\\ConeheadZombie\_7.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombie\\ConeheadZombie\_9.bmp" };

for (int i = 0; i < 5; i++)

zombie\_conehead\_animation.AddBitmap(filenamec[i], RGB(255, 255, 255));

char \*filenameca[5] = {

".\\bitmaps\\NormalZombie\\ConeheadZombieAttack\\ConeheadZombieAttack\_1.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombieAttack\\ConeheadZombieAttack\_3.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombieAttack\\ConeheadZombieAttack\_5.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombieAttack\\ConeheadZombieAttack\_7.bmp",

".\\bitmaps\\NormalZombie\\ConeheadZombieAttack\\ConeheadZombieAttack\_9.bmp" };

for (int i = 0; i < 5; i++)

zombie\_conehead\_attack\_animation.AddBitmap(filenameca[i], RGB(255, 255, 255));

}

else if (zombie\_style == "flag") {

char \*filenamefa[5] = {

".\\bitmaps\\NormalZombie\\FlagZombie\\FlagZombie\_1.bmp",

".\\bitmaps\\NormalZombie\\FlagZombie\\FlagZombie\_3.bmp",

".\\bitmaps\\NormalZombie\\FlagZombie\\FlagZombie\_5.bmp",

".\\bitmaps\\NormalZombie\\FlagZombie\\FlagZombie\_7.bmp",

".\\bitmaps\\NormalZombie\\FlagZombie\\FlagZombie\_9.bmp" };

for (int i = 0; i < 5; i++)

zombie\_flag\_animation.AddBitmap(filenamefa[i], RGB(255, 255, 255));

char \*filename[5] = {

".\\bitmaps\\NormalZombie\\FlagZombieAttack\\FlagZombieAttack\_1.bmp",

".\\bitmaps\\NormalZombie\\FlagZombieAttack\\FlagZombieAttack\_3.bmp",

".\\bitmaps\\NormalZombie\\FlagZombieAttack\\FlagZombieAttack\_5.bmp",

".\\bitmaps\\NormalZombie\\FlagZombieAttack\\FlagZombieAttack\_7.bmp",

".\\bitmaps\\NormalZombie\\FlagZombieAttack\\FlagZombieAttack\_9.bmp" };

for (int i = 0; i < 5; i++)

zombie\_flag\_attack\_animation.AddBitmap(filename[i], RGB(255, 255, 255));

}

char \*filenamef[5] = { ".\\bitmaps\\NormalZombie\\Zombie\\Zombie\_0.bmp",

".\\bitmaps\\NormalZombie\\Zombie\\Zombie\_1.bmp", ".\\bitmaps\\NormalZombie\\Zombie\\Zombie\_2.bmp",

".\\bitmaps\\NormalZombie\\Zombie\\Zombie\_3.bmp", ".\\bitmaps\\NormalZombie\\Zombie\\Zombie\_4.bmp"

};

for (int i = 0; i < 5; i++)

zombie\_animation.AddBitmap(filenamef[i], RGB(255, 255, 255));

char \*filenamea[5] = { ".\\bitmaps\\NormalZombie\\ZombieAttack\\ZombieAttack\_0.bmp", ".\\bitmaps\\NormalZombie\\ZombieAttack\\ZombieAttack\_1.bmp", ".\\bitmaps\\NormalZombie\\ZombieAttack\\ZombieAttack\_3.bmp", ".\\bitmaps\\NormalZombie\\ZombieAttack\\ZombieAttack\_4.bmp", ".\\bitmaps\\NormalZombie\\ZombieAttack\\ZombieAttack\_5.bmp"};

for (int i = 0; i < 5; i++)

zombie\_attack\_animation.AddBitmap(filenamea[i], RGB(255, 255, 255));

char \*filenamed[12] = { ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_0.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_1.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_2.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_3.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_4.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_5.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_6.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_7.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_8.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_9.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_10.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieDie\_11.bmp" };

for (int i = 0; i < 12; i++)

zombie\_die\_animation.AddBitmap(filenamed[i], RGB(255, 255, 255));

char \*filenamedh[12] = { ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_0.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_1.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_2.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_3.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_4.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_5.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_6.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_7.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_8.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_9.bmp", ".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_10.bmp",

".\\bitmaps\\NormalZombie\\ZombieDie\\ZombieHead\_11.bmp" };

for (int i = 0; i < 12; i++)

zombie\_die\_animation\_head.AddBitmap(filenamedh[i], RGB(255, 255, 255));

char \*filenamedb[19] = { ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_0.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_1.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_2.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_3.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_4.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_5.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_6.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_7.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_8.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_9.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_10.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_11.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_12.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_14.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_15.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_16.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_17.bmp", ".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_18.bmp",

".\\bitmaps\\NormalZombie\\BoomDie\\BoomDie\_19.bmp" };

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

zombie\_die\_animation\_bomb.AddBitmap(filenamedb[i], RGB(255, 255, 255));

}

void OnMove(std::string mode) {

if (bomb\_flag) {

mode = "bomb";

}

if (mode == "attack") {

if (zombie\_style == "flag") {

zombie\_flag\_attack\_animation.OnMove();

}

else if (zombie\_style == "conehead" && blood > 10) {

zombie\_conehead\_attack\_animation.OnMove();

}

else if (zombie\_style == "bucket" && blood > 10) {

zombie\_bucket\_attack\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood > 10) {

zombie\_news\_attack\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood <= 10) {

zombie\_nonews\_attack\_animation.OnMove();

}

else {

zombie\_attack\_animation.OnMove();

}

}

else if (mode == "walk") {

if (delay % 2 == 0) {

x -= 1;

if (zombie\_style == "flag") {

zombie\_flag\_animation.OnMove();

}

else if (zombie\_style == "conehead" && blood > 10) {

zombie\_conehead\_animation.OnMove();

}

else if (zombie\_style == "bucket" && blood > 10) {

zombie\_bucket\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood > 10) {

zombie\_news\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood <= 10) {

zombie\_nonews\_animation.OnMove();

x -= 3;

}

else {

zombie\_animation.OnMove();

}

delay = 0;

}

}

else if (mode == "die") {

if (zombie\_style == "newspaper") {

zombie\_news\_die\_animation.OnMove();

zombie\_die\_animation\_head.OnMove();

}

else {

zombie\_die\_animation.OnMove();

zombie\_die\_animation\_head.OnMove();

}

}

else if (mode == "bomb") {

zombie\_die\_animation\_bomb.OnMove();

}

delay++;

}

void OnMove(std::string mode, bool fast\_mode) {

if (bomb\_flag) {

mode = "bomb";

}

if (mode == "attack") {

if (zombie\_style == "flag") {

zombie\_flag\_attack\_animation.OnMove();

}

else if (zombie\_style == "conehead" && blood > 10) {

zombie\_conehead\_attack\_animation.OnMove();

}

else if (zombie\_style == "bucket" && blood > 10) {

zombie\_bucket\_attack\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood > 10) {

zombie\_news\_attack\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood <= 10) {

zombie\_nonews\_attack\_animation.OnMove();

}

else {

zombie\_attack\_animation.OnMove();

}

}

else if (mode == "walk") {

if (delay % 1 == 0) {

x -= 5;

if (zombie\_style == "flag") {

zombie\_flag\_animation.OnMove();

}

else if (zombie\_style == "conehead" && blood > 10) {

zombie\_conehead\_animation.OnMove();

}

else if (zombie\_style == "bucket" && blood > 10) {

zombie\_bucket\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood > 10) {

zombie\_news\_animation.OnMove();

}

else if (zombie\_style == "newspaper" && blood <= 10) {

zombie\_nonews\_animation.OnMove();

x -= 1;

}

else {

zombie\_animation.OnMove();

}

}

}

else if (mode == "die") {

if (zombie\_style == "newspaper") {

zombie\_news\_die\_animation.OnMove();

zombie\_die\_animation\_head.OnMove();

}

else {

zombie\_die\_animation.OnMove();

zombie\_die\_animation\_head.OnMove();

}

}

else if (mode == "bomb") {

zombie\_die\_animation\_bomb.OnMove();

}

delay++;

}

void OnShow(std::string mode) {

if (bomb\_flag) {

mode = "bomb";

}

if (mode == "attack") {

if (zombie\_style == "flag") {

zombie\_flag\_attack\_animation.SetTopLeft(x, y);

zombie\_flag\_attack\_animation.OnShow();

}

else if (zombie\_style == "conehead" && blood > 10) {

zombie\_conehead\_attack\_animation.SetTopLeft(x, y);

zombie\_conehead\_attack\_animation.OnShow();

}

else if (zombie\_style == "bucket" && blood > 10) {

zombie\_bucket\_attack\_animation.SetTopLeft(x, y);

zombie\_bucket\_attack\_animation.OnShow();

}

else if (zombie\_style == "newspaper" && blood > 10) {

zombie\_news\_attack\_animation.SetTopLeft(x, y);

zombie\_news\_attack\_animation.OnShow();

}

else if (zombie\_style == "newspaper" && blood <= 10) {

zombie\_nonews\_attack\_animation.SetTopLeft(x, y);

zombie\_nonews\_attack\_animation.OnShow();

}

else {

zombie\_attack\_animation.SetTopLeft(x, y);

zombie\_attack\_animation.OnShow();

}

}

else if (mode == "walk") {

if (zombie\_style == "flag") {

zombie\_flag\_animation.SetTopLeft(x, y);

zombie\_flag\_animation.OnShow();

}

else if (zombie\_style == "conehead" && blood > 10) {

zombie\_conehead\_animation.SetTopLeft(x, y);

zombie\_conehead\_animation.OnShow();

}

else if (zombie\_style == "bucket" && blood > 10) {

zombie\_bucket\_animation.SetTopLeft(x, y);

zombie\_bucket\_animation.OnShow();

}

else if (zombie\_style == "newspaper" && blood > 10) {

zombie\_news\_animation.SetTopLeft(x, y);

zombie\_news\_animation.OnShow();

}

else if (zombie\_style == "newspaper" && blood <= 10) {

zombie\_nonews\_animation.SetTopLeft(x, y);

zombie\_nonews\_animation.OnShow();

}

else {

zombie\_animation.SetTopLeft(x, y);

zombie\_animation.OnShow();

}

}

else if (mode == "die") {

if (zombie\_die\_delay\_time > 0) {

if (zombie\_style == "newspaper") {

zombie\_news\_die\_animation.SetTopLeft(x, y);

zombie\_news\_die\_animation.OnShow();

zombie\_die\_animation\_head.SetTopLeft(x + 50, y);

zombie\_die\_animation\_head.OnShow();

zombie\_die\_delay\_time--;

}

else {

zombie\_die\_animation.SetTopLeft(x, y);

zombie\_die\_animation.OnShow();

zombie\_die\_animation\_head.SetTopLeft(x + 50, y);

zombie\_die\_animation\_head.OnShow();

zombie\_die\_delay\_time--;

}

}

else {

x = 1000;

}

}

else if (mode == "bomb") {

if (zombie\_bomb\_die\_delay\_time > 0) {

zombie\_die\_animation\_bomb.SetTopLeft(x, y);

zombie\_die\_animation\_bomb.OnShow();

zombie\_bomb\_die\_delay\_time--;

}

else {

is\_alive = false;

x = 1000;

}

}

}

void SetBombFlag() {

bomb\_flag = true;

}

bool IsAlive() {

return is\_alive;

}

void SetIsAlive(bool alive) {

is\_alive = alive;

}

void freezen() {

x += 2;

}

void SetX(int x) {

this->x = x;

}

void SetY(int y) {

this->y = y;

}

int GetX() {

return int(x);

}

int GetY() {

return int(y);

}

void LostBlood(int attack\_blood) {

blood = blood - attack\_blood;

if (blood == 0) {

is\_alive = false;

}

}

int GetBlood() {

return blood;

}

int GetMapYLocation() {

return Map\_Y\_Location;

}

int GetAttackPower() {

return attack\_power;

}

private:

int x, y;

bool is\_alive;

int blood;

int Map\_Y\_Location;

CAnimation zombie\_animation;

CAnimation zombie\_attack\_animation;

CAnimation zombie\_die\_animation;

CAnimation zombie\_die\_animation\_head;

CAnimation zombie\_die\_animation\_bomb;

CAnimation zombie\_flag\_animation;

CAnimation zombie\_flag\_attack\_animation;

CAnimation zombie\_conehead\_animation;

CAnimation zombie\_conehead\_attack\_animation;

CAnimation zombie\_bucket\_animation;

CAnimation zombie\_bucket\_attack\_animation;

CAnimation zombie\_news\_animation;

CAnimation zombie\_news\_attack\_animation;

CAnimation zombie\_news\_die\_animation;

CAnimation zombie\_nonews\_animation;

CAnimation zombie\_nonews\_attack\_animation;

int zombie\_bomb\_die\_delay\_time;

int zombie\_die\_delay\_time;

int attack\_power = 1;

int delay = 0;

bool bomb\_flag;

std::string zombie\_style;

};

}

#endif