

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	2
1.	動機	2
2.	分工	2
二、	遊戲介紹	2
1.	遊戲說明	2
2.	遊戲圖形	3
3.	遊戲音效	6
三、	程式設計	7
1.	程式架構	7
2.	程式類別	8
3.	程式技術	9
四、	結語	9
1.	問題及解決方法	9
2.	時間表	10
3.	貢獻比例	11
4.	自我檢核表	11
5.	收穫	12
6.	心得、感想	12
7.	對於本課程的建議	13
	附錄	14

## 一、 簡介

### 1. 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。

### 2. 分工： 由羅羽軒主導，周雨柔協助。




## 二、 遊戲介紹

















### 1. 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		

2. 遊戲圖形：













bullet	ice bullet	mushroom bullet
		









name	plant	card	name	plant	card
shooter			ice shooter		
Sun flower			mushroom shooter		
cherry 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		

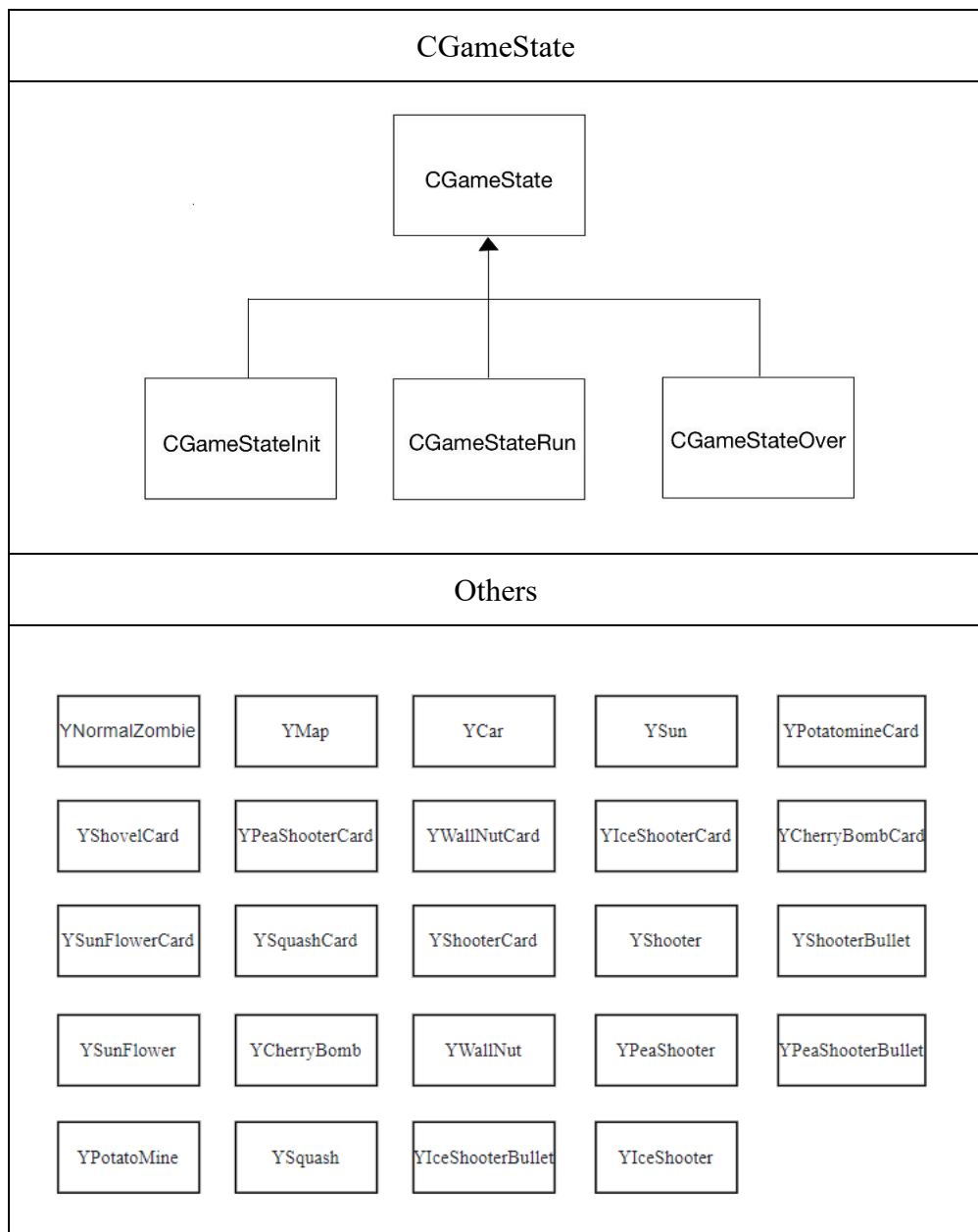
die (head part)	
	
normal die (body part)	
	
newspaper zombie die (body part)	
	
bomb die	
	

### 3. 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：





2. 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	主程式
<b>總行數</b>	<b>4072</b>	<b>156</b>	<b>4228</b>

### 3. 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。

## 三、 結語

### 1. 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

2. 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
<b>合計</b>	<b>170</b>	<b>99</b>	

3. 貢獻比例：

羅羽軒：60%、周雨柔：40%

4. 自我檢核表：

	項目	完成否	無法完成的原因
1	解決 Memory leak	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
2	自訂遊戲 icon	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
3	有 About 畫面	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
4	初始畫面說明按鍵及滑鼠之用法與密技	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
5	上傳 setup/apk/source 檔	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
6	setup 檔可正確執行	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
7	報告字型、點數、對齊、行距、頁碼等格式正確	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
8	報告封面、側邊格式正確	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
9	報告附錄程式格式正確	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
10	全螢幕啟動	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	

## 5. 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。

## 6. 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！

7. 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 <draw.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);          // 切换至 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, "微软正黑體");
    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) {
            potatamine.SetTopLeft(232, 91);
            potatamine.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) {
        // potatamine 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 > potatamine_card.GetX() && point.y > potatamine_card.GetY() &&
point.x < potatamine_card.GetX() + 65 && point.y < potatamine_card.GetY() + 90 && potatamine_card.IsAlive()) {
        potatamine_card.SetIsAlive(false);
        sun_amount -= potatamine_card.GetSunCost();
        generatePotatamineFlag = true;
        potatamine_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 && potatamine_card.GetSunCost() > sun_amount) {
    potatamine_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 && potatamine_card_delay_flag > 0) {
        potatamine_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 < walnut_vector.size(); i++) {
        if (!map.checkmyMap(walnut_vector.at(i)->GetX(), walnut_vector.at(i)->GetY())) {
            walnut_vector.erase(walnut_vector.begin() + i);    //if map is zero, delete the plant
        }
        else {
            walnut_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 < potatamine_vector.size(); i++) {
        if (!map.checkmyMap(potatamine_vector.at(i)->GetX(), potatamine_vector.at(i)->GetY()))
{
            potatamine_vector.erase(potatamine_vector.begin() + i);    //if map is zero, delete
the plant
        }
        else {
            potatamine_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 && !potatamine_vector.empty()) {
            for (size_t j = 0; j < potatamine_vector.size(); j++) {
                if (potatamine_vector.at(j)->Bomb()
&& potatamine_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 < potatamine_vector.size(); j++) {
                if (potatamine_vector.at(j)-
>checkPlantCollideWithZombie(normalzombie_vector.at(i)->GetX() + 70, normalzombie_vector.at(i)->GetY() + 30)) {
                    potatamine_vector.at(j)->SetZombieChecked();
                    potatamine_vector.at(j)->LostBlood(normalzombie_vector.at(i)-
>GetAttackPower());

                    if (potatamine_vector.at(j)->GetBlood() < 1) {
                        potatamine_vector.at(j)->SetIsAlive(false);
                        map.unsetmyMap(potatamine_vector.at(j)->GetX(),
potatamine_vector.at(j)->GetY());

                        potatamine_vector.erase(potatamine_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());

shooter_vector.at(j)->GetBlood() < 1) {
    shooter_vector.at(j)->SetIsAlive(false);
    map.unsetmyMap(shooter_vector.at(j)->GetX(),
    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 < potatamine_vector.size(); i++) {
            if (!potatamine_vector.at(i)->IsAlive()) {
                map.unsetmyMap(potatamine_vector.at(i)->GetX(), potatamine_vector.at(i)-
>GetY());
                potatamine_vector.erase(potatamine_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 >= potatamine_card.GetSunCost() &&
potatamine_card_delay_flag == 0) {
    potatamine_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 < potatamine_vector.size(); i++) {
                potatamine_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) {
            potatamine_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);
CDDraw::ReleaseBackCDC();
}

// zombie amount
if (flag == 2) {

```

// 放掉 font f (千萬不要漏了放掉)  
// 放掉 Back Plain 的 CDC

```

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 generatePotatamineFlag;
    bool generateShooterFlag;
    bool generateSquashFlag;

    bool shovelFlag;

    YSunFlowerCard sun_flower_card;
    YPeaShooterCard pea_shooter_card;
    YWallNutCard walnut_card;
    YCherryBombCard cherrybomb_card;
    YIceShooterCard ice_shooter_card;
    YPotatamineCard potatamine_card;
    YShooterCard shooter_card;
    YSquashCard squash_card;

    int sun_flower_card_delay_flag;
    int peashooter_card_delay_flag;
    int walnut_card_delay_flag;
    int cherrybomb_card_delay_flag;
    int iceshooter_card_delay_flag;
    int potatamine_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>> walnut_vector;
    std::vector<shared_ptr<YCherryBomb>> cherrybomb_vector;
    std::vector<shared_ptr<YIceShooter>> iceshooter_vector;
    std::vector<shared_ptr<YPotatoMine>> potatamine_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();
    void OnInit();
    void OnLButtonDown(UINT nFlags, CPoint point);
protected:
    void OnMove();
    void OnShow();
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  walnut;
    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 <draw.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()
    {
        peasooter_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);
            }
        }
    }
};

```

```

        walnut_animation.OnShow();
    }
    else if (blood > 200) {
        walnut_cracked1_animation.SetTopLeft(x, y);
        walnut_cracked1_animation.OnShow();
    }
    else {
        walnut_cracked2_animation.SetTopLeft(x, y);
        walnut_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 walnut_animation;
    CAnimation walnut_cracked1_animation;
    CAnimation walnut_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 *filenameee[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(filenameee[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(filenameei[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 potatamine_animation;
    CAnimation potatamine_explode_animation;
    CAnimation potatamine_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/
                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 *filenamen[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(filenamen[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 *filamena[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(filamena[i], RGB(255, 255, 255));

    char *filamene[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(filamene[i], RGB(255, 255, 255));

    char *filameno[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(filameno[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 *filenameec[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(filenameec[i], RGB(255, 255, 255));

        char *filenameeca[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(filenameeca[i], RGB(255, 255, 255));
    }

    else if (zombie_style == "flag") {
        char *filenameefa[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(filenameefa[i], RGB(255, 255, 255));

        char *filenameef[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(filenameef[i], RGB(255, 255, 255));
    }

    char *filenameef[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(filenameef[i], RGB(255, 255, 255));

    char *filenameea[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(filenameea[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(filened[i], RGB(255, 255, 255));

    char *filenedh[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(filenedh[i], RGB(255, 255, 255));

    char *filenedb[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(filenedb[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 freeze() {
    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

```