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 45 組

108590452 林峻霆

107590451 曾政翔

目錄

一、 簡介

1.動機：這個遊戲是我們的童年，因爲一說到做遊戲就想起了這個，所以我們就選擇這個遊戲，並模仿和執行這個計畫

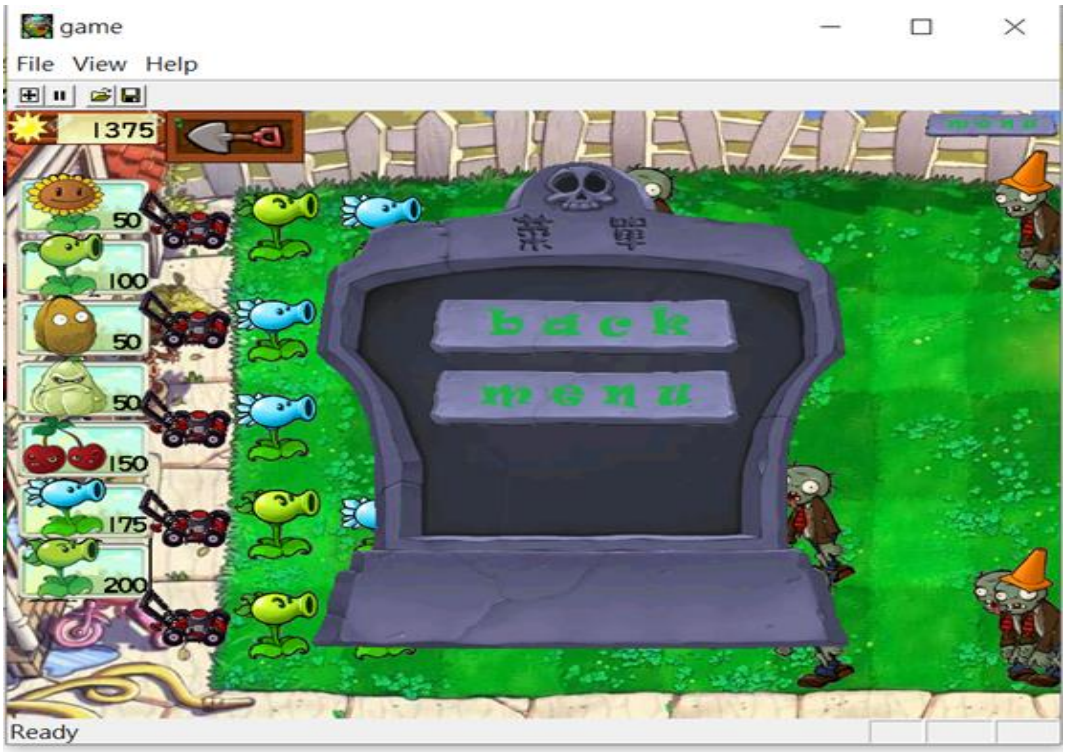
2.分工：基本上我們都是一起寫的，一起想的，搜尋參考資料並一起討論。

二、 遊戲介紹

1. 遊戲說明：可怕的殭屍(Zombies) 將要入侵你甜蜜的家，你必須栽種 7 種的植物(Plants)，巧妙運用他們的特性和佈陣以阻擋他們的進攻。遊戲當中植物會協助你驅趕殭屍，而必須不斷賺取太陽能源以種植新的植物，每個關皆以阻擋定數量的殭屍群為過關目的，每一關過完都可以賺到一個新的植物，並能在下一關選擇之一起抵禦敵人，遊戲規則簡單但變化性很大。非常適合打發時間時候趣味玩之。

2. 遊戲圖形

(遊戲進行畫面，各式殭屍行走以及各種攻擊手的攻擊畫面)



(遊戲菜單介面，可以暫停遊戲，提供回主畫面的互動功能)

(遊戲勝利畫面，提供下一關及回主畫面的互動功能)

(遊戲失敗畫面，提供下一關及回主畫面的互動功能)

(使用櫻桃炸彈攻擊畫面，可以把周圍殭屍一次性消滅)

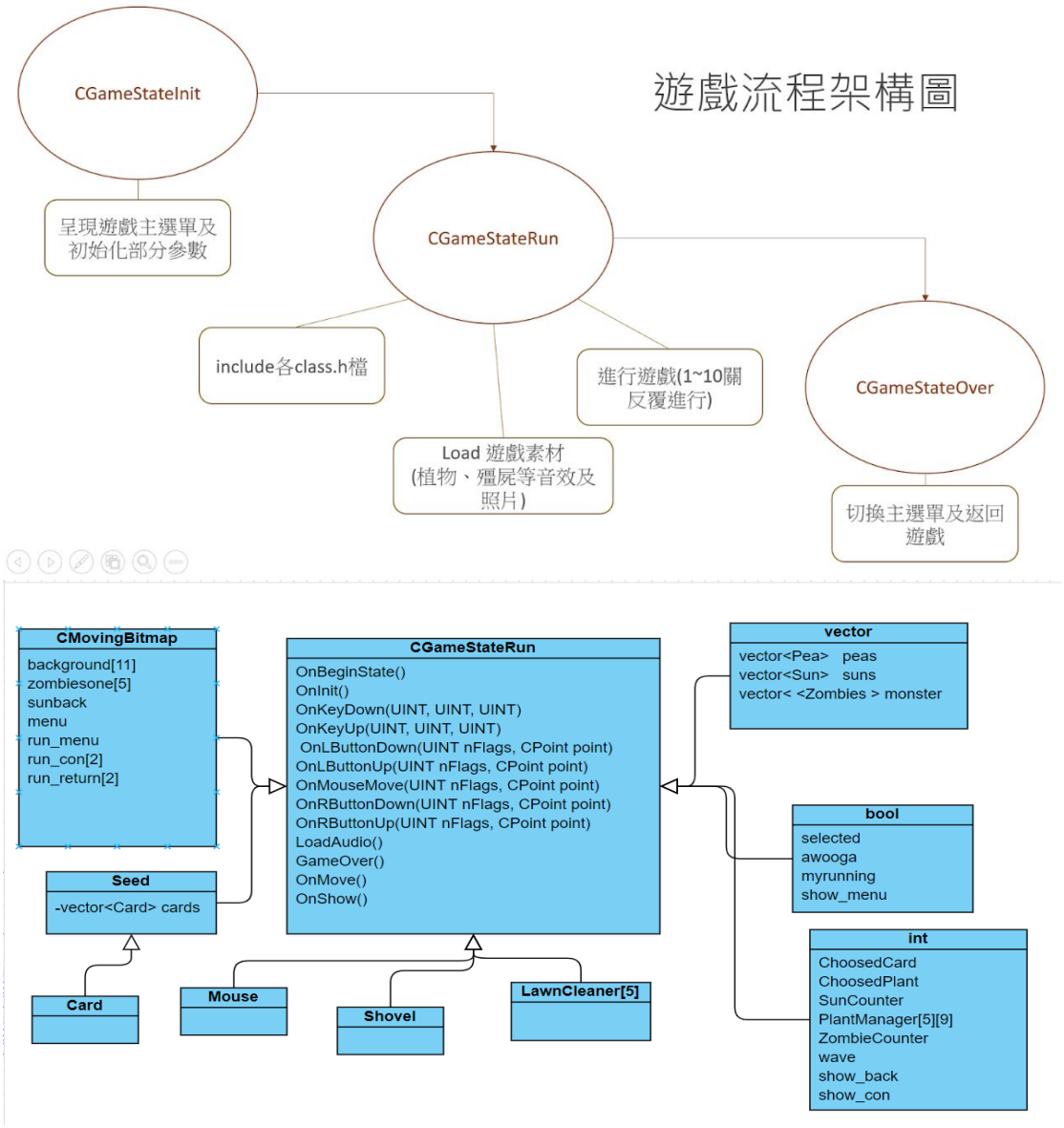
3. 遊戲音效

項次	AUDIO_ID	使用情境
1	AUDIO_AWOOGA,	殭屍產生的吼叫聲
2	AUDIO_CHERRY_BOMB,	櫻桃炸彈的音效
3	AUDIO_CHOMP_1,	殭屍咬擊植物聲音
4	AUDIO_CHOMP_2,	殭屍咬擊植物聲音
5	AUDIO_CHOMP_3,	殭屍咬擊植物聲音
6	AUDIO_EVIL_LAUGH,	遊戲開場的聲音
7	AUDIO_FINALWAVE,	遊戲開場的聲音
8	AUDIO_GROAN_1,	殭屍吼叫聲
9	AUDIO_GROAN_2,	殭屍吼叫聲
10	AUDIO_GROAN_3,	殭屍吼叫聲
11	AUDIO_GROAN_4,	殭屍吼叫聲
12	AUDIO_GROAN_5,	殭屍吼叫聲
13	AUDIO_GROAN_6,	殭屍吼叫聲
14	AUDIO_LAWN_MOWER,	割草機引擎聲
15	AUDIO_LOSE_MUSIC,	遊戲失敗的音效
16	AUDIO_MAIN_MUSIC,	遊戲進行中的背景音樂
17	AUDIO_MENU,	主選頁的背景音樂
18	AUDIO_PLANT,	植物放置的音效
19	AUDIO_COLLECT_SUN,	點擊太陽的音效
20	AUDIO_CHOOSE_CARD,	選擇卡片音效
21	AUDIO_HIT_BUCKET_1	水桶殭屍被攻擊音效

22	AUDIO_HIT_BUCKET_2	水桶殭屍被攻擊音效
23	AUDIO_SHOVEL,	點擊鏟子的音效
24	AUDIO_SPLAT_1,	豆子擊中殭屍音效
25	AUDIO_SPLAT_2,	豆子擊中殭屍音效
26	AUDIO_SPLAT_3,	豆子擊中殭屍音效
27	AUDIO_SHOOT,	噴射豆子音效
28	AUDIO_WIN_MUSIC	勝利音效
共 28 項		

三、 程式設計

1.程式架構



(class diagram)

2 程式類別

類別名稱	.h 行數	.cpp 行數	說明
card	37	144	提供植物選擇給玩家
LawnCleaner	19	45	除草機(殺殭屍用)
maps	14	26	處理所有地圖(沒用到)
mouse	24	64	處理所有滑鼠功能
pea	24	73	豌豆功能
Plants	68	323	7 種不同植物，和功能
Seed	25	114	處理所擁有之太陽
Shovel	17	36	處理鏟子功能
Sun	29	98	處理太陽動畫和功能
Zombies	73	329	處理殭屍功能
總行數	330	1252	

3.程式技術

1. 背景：用一個 array[11]存取 11 張圖，0-10 存白天的圖，0:只有一個 row,三個 row 2-9:五個 row 10:夜晚圖片我們設置一個變數為 gamelevel 紀錄當下是第幾關，以此變數決定當下背景，每當是最後一波殭屍進攻時，無條件進入夜晚模式。
2. 祕技：上鍵：讓殭屍消失，下鍵：太陽(money 這個變數)數量增加 1000，右鍵：增加殭屍走路速度(velocity)
3. 殭屍：分成三種殭屍 ：在程式碼 zombies.cpp 有用註解說明
4. 植物：分成 7 種不同植物，且有不同功能， in mygame.cpp 中用兩種 5*9 的 array 存取，PlantManger 處理紀錄當下所有的植物 id，PlantsClasst 處理當下所有植物的功能
5. 太陽 ：在程式碼 sun.h 有用註解說明
6. 鏟子 ：移除植物的時候將 PlantsManger 中該格設成 0，PlantsClass 該格植物做清除
7. 種子 ：在 mygame.cpp 中用 money 紀錄執行購買植物的功能

8. 卡片 ：顯示左列植物選單，並處理圖片功能
9. 豌豆 ：和植物(ID 2 and 7)相關，當此兩種植物發動功能時出現，並攻擊殭屍
10. 除草機 ：第三關後出現，當殭屍碰到後發動移動功能，清除當排所有殭屍
11. 滑鼠：在程式碼 mouse.h 有用註解說明
12. 暫停：在 mygame.cpp 中我們用 myrunning 來當遊戲執行的參數

四、 結語

1.問題及解決方法

vector 異常:

在 window framework 中使用 vector 去裝遊戲裡的植物 class 物件時，在呼叫 onshow()的時候會引發底層 framework 的 error，經詢問助教之後，發現可能是 vector 內的物件排列不是順序排列組合，觸發 onshow()內的 error，後來我們改用 array 去裝 class 才解決底層 error 的問題，但也不是所有 class 都不能採用 vector 去包，例如遊戲內使用的太陽就沒有任何問題，和殭屍所使用的 class，引用 vector function clear()也會觸發底層程式碼的錯誤，改用了 shared_ptr 才解決此問題。

全螢幕問題:

在擴大全螢幕後，發現因為比例不同，導致圖片無法放大，經過助教幫助下才知道，每個照片的比例要符合 framework 內設計的比例大小，才能切換全螢幕，因此我們又多花了一周的時間把過去所使用到的圖片素材都裁切成 800:600 的比例大小，因為有許多的 if 條件判斷都是從畫面的位置去加以控制的，所以花了不少的時間處理有關比例調整造成問題的程式碼。

2. 時間表

周次	曾政翔	林峻霆	說明
1	10	10	學習 framework,git,和決定遊戲
2	10	10	實作 framework,git 和收集素材
3	10	10	實作主畫面選單 和遊戲背景移動

4	10	10	製作 sun class
5	10	10	製作 seed class
6	10	10	製作 card class
7	12	12	製作 plants class (sun flower)
8	12	12	製作 plants and pea class (pea shooter)
9	10	10	製作 mouse class
10	10	10	製作 shovel class and lawncleaner
11	15	15	製作 zombies class
12	10	10	增加植物種類和祕技
13	10	10	增加殭屍種類 和關卡
14	10	10	增加音效和處理關卡畫面
15	10	10	增加關卡連接畫面
16	10	10	處理 meau 之操作和按鍵操作
17	10	10	解決全螢幕和安裝檔之問題
總時數	179	179	

3.貢獻比例

組員 A:曾政翔 50%	組員 B:林峻霆 50%

4.自我檢核表

週次	項目	完成否	無法完成原因
1	解決 Memory leak	☑已完成 □未完成	
2	自定遊戲 Icon	☑已完成 □未完成	
3	有 About 畫面	☑已完成 □未完成	
4	初始畫面說明按鍵及滑鼠	☑已完成 □未完成	
5	之用法與密技	☑已完成 □未完成	
6	上傳 setup/apk/source 檔	☑已完成 □未完成	
7	setup 檔可正確執行	☑已完成 □未完成	
8	報告字型、點數、對齊、行	☑已完成 □未完成	
9	距、頁碼等格式正確	☑已完成 □未完成	
10	全螢幕啟動–改列加分項目	☑已完成 □未完成	

5.收穫

林峻霆：

開始的前幾週遇到的難題是不知道 Game Framework 要怎麼使用，也沒辦法找到相關資訊，只好自己埋頭研究，看老師提供的範例來推敲要怎麼使用，可能寫了好幾個小時才完成一個小功能，甚至不知不覺就寫到天亮了，然而過了適應其後就越來越得心應手，可以猜測到程式碼的邏輯大概是怎麼樣就不用花這麼多時間在研究程式碼。另外 Game Framework 有很多我們沒發現的功能或是沒有的功能，所以我們必須自己增加或改寫成我們想要的。

經過了這學期的課程，我變得更了解 C++程式設計、設計遊戲程式的思維、和他人共同使用 git、可以自己試著理解參考資料不多的程式碼、能跟其他人在較大的 project 中溝通合作。

曾政翔：

剛開始在修這門課的時候，以為可以跟其他程式語言的課程一樣，把所會的技術運用起來就可以了，但是想不到的是，這門實習課考驗的不只是個人的程式實力，講求的是團隊合作的成果，所以不是靠單幹，而如何跟隊友做好溝通，讓 1+1 大於 2 就是這們課背後裡所要訓練我們的核心價值，其中溝通上面非常重要，兩個人若不明白彼此的想法，打出來的程式肯定無法相容，所以開發工具就額外重要許多，這學期打程式前老師先帶著我們學 git 工具，學了 git 就發現團隊合作下非常好用，相信未來在公司工作的時候，git 技術也是必備的條件之一。

6. 心得、感想

林峻霆：

這個物件導向程式設計實習課是第一次在學校被指派要完成一個較大型的程式，因為我開始寫程式到現在的時間也不長（一年半），所以也是目前寫過最大的程式了。我覺得老師用這種方式來讓我們開始習慣以後類似以後去工作模式，不是出社會之後才沒做好準備直接碰壁，是一件很棒的事。在寫大型的程式時要做好很多規劃，在我的認知中實習業界的開發者要不斷的開會統整資訊，還要對自己負責的部分加以說明，來讓同事或未來維護的人員可以更快進入況狀。因為時間上的壓力，坦白說我們的程式碼沒有規劃得非常好看容易閱讀維護，也有很多進步的空間，有些記憶體上面的設計可以更佳優化，但是經過這學期的課程的經驗，相信在未來做大 project 時可以做得更好且更得心應手。

曾政翔：

經過一學期的實習後，學會了很多經驗，例如團隊合作的經驗，從收穫裡我也有提到，除了程式上所學會的技術外，最大的收穫是團隊合作的部分，也了解了原來要打出一套完整的遊戲，需要非常多的時間與精力，也明白那些上市的遊戲出現 bug 後，遊戲工程師在修這些 bug 的時候的心情，真是感觸良多，同時銜接上學期的物件導向所學的架構，class 的繼承與使用，運用在 framework 上，遇到問題也有助教及老師可以提問，幾乎都是有問必答，學習起來非常順利，相信未來因為這門課的基礎可以勝任其他的任務。

7.對於本課程的建議：

關於這堂課我覺得這門課安排的時間不夠多，應該有更多的時間能夠後老師或者助教討論一些問題，另一個建議是，我們認為這堂課的人數過多，老師無法顧及到所有學生的權益。

五、 附錄

card.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "card.h"
#include "Seed.h"
namespace game_framework
{
    Card::Card() {
    }
    // 卡片的初始化
    Card::Card(int a) {
        ID = a;
```

```
    availible = true;                          // 設定卡片的等待時間和價錢

    switch (ID) {

    case 1:delay = 33 * 5;  price = 50;  break; // SunFlower

    case 2:delay = 33 * 5;  price = 100; break; // PeaShoot

    case 3:delay = 33 * 10; price = 50;  break; // WallNut

    case 4:delay = 33 * 30; price = 50;  break; // Squash

    case 5:delay = 33 * 30; price = 150; break; // Cherry boom

    case 6:delay = 33 * 5;  price = 175; break; // Snow

    case 7:delay = 33 * 5;  price = 200; break; // Repeater

    }

    counting = false;

    counter = delay;

}

//每次重新開始遊戲的重設卡片初始狀態

void Card::Reset() {

    availible = true;

    counter = delay;

    counting = false;

}
```

```
// 讀取卡片所需的所有圖片

void Card::LoadBitmap() {

    LoadPlant();

    LoadPrice();

}

void Card::LoadPrice() {

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

        for (int j = 0; j < 10; j++) {

            char FILENAME[100];

            sprintf(FILENAME, ".\\BMP_RES\\image\\interface\\%d.bmp", j);

            number[i][j].LoadBitmap(FILENAME, RGB(255, 255, 255));

        }

    }



}

void Card::LoadPlant() {

    switch (ID) {
```

```
        case 1:
plant.LoadBitmap(".\\BMP_RES\\image\\interface\\menu\\handbook\\Card\\plants
\\SunFlower.bmp", RGB(0, 0, 0));   break;
        case 2:
plant.LoadBitmap(".\\BMP_RES\\image\\interface\\menu\\handbook\\Card\\plants
\\Peashooter.bmp", RGB(0, 0, 0));  break;
        case 3:
plant.LoadBitmap(".\\BMP_RES\\image\\interface\\menu\\handbook\\Card\\plants
\\WallNut.bmp", RGB(0, 0, 0));     break;
        case 4:
plant.LoadBitmap(".\\BMP_RES\\image\\interface\\menu\\handbook\\Card\\plants
\\Squash.bmp", RGB(0, 0, 0));      break;
        case 5:
plant.LoadBitmap(".\\BMP_RES\\image\\interface\\menu\\handbook\\Card\\plants
\\CherryBomb.bmp", RGB(0, 0, 0));  break;
        case 6:
plant.LoadBitmap(".\\BMP_RES\\image\\interface\\menu\\handbook\\Card\\plants
\\SnowPea.bmp", RGB(0, 0, 0));     break;
```

```
        case 7:
plant.LoadBitmap(".\\BMP_RES\\image\\interface\\menu\\handbook\\Card\\plants
\\Repeater.bmp", RGB(0, 0, 0));    break;
        }
    }
    int Card::GetPrice() {
        return price;
    }
    // int Card::GetWidth() {
    //  return bmp.Width();
    // }
    // int Card::GetHeight() {
    //  return bmp.Height();
    // }
    int Card::GetX() {
        return x;
    }
    int Card::GetY() {
        return y;
```

```
}

int Card::GetID() {

    return ID;

}

void Card::SetXY(int num) {

    x = 0;

    y = 50 + num * 60;

}

// 設定卡片是可被選取

void Card::SetAvailible(bool a) {

    availible = a;

}

// 回傳卡片是否可被選取

bool Card::isAvailible() {

    return availible && CounterFinished();

}

// 計時卡片的冷卻時間

void Card::DelayCounter() {

    if (!CounterFinished()) {
```

```
        counting = true;

        counter++;

    }

    else if (CounterFinished()) {

        counting = false;

    }

}

void Card::ResetCounter() {

    counter = 0;

}

bool Card::CounterFinished() {

    return counter == delay;

}

void Card::OnMove() {

    DelayCounter();

    y2 = double(y) - 70 * counter / delay;

}

void Card::OnShow() {

    int modx = 0, mody = 0;
```

```
if (!CounterFinished()) {

    plant.SetTopLeft(x + 8 + modx, y + 12 + mody);

    plant.ShowBitmap(1);

    int y3 = int(y2);

}

else if (!availible) {

    plant.SetTopLeft(x + 8 + modx, y + 12 + mody);

    plant.ShowBitmap(1);

}

else if (availible) {



    plant.SetTopLeft(x + 8 + modx, y + 12 + mody);

    plant.ShowBitmap(1);

}

if (GetPrice() > 0 && GetPrice() < 100) {

    for (int i = 0, num = GetPrice(); i < 2; i++, num /= 10) {

        number[i][num % 10].SetTopLeft(x + 90 - i * 10, y + 47);

        number[i][num % 10].ShowBitmap();
```

```
            }

        }

        else if (GetPrice() >= 100 && GetPrice() < 1000) {

            for (int i = 0, num = GetPrice(); i < 3; i++, num /= 10) {

                number[i][num % 10].SetTopLeft(x + 95 - i * 11, y + 47);

                number[i][num % 10].ShowBitmap();

            }

        }

    }

}
```

card.h

```
#pragma once

namespace game_framework {

    class Card {

    public:

        Card();

        Card(int);
```

```
    void OnMove();              //處理卡片的動作

    void OnShow();              //顯示卡片

    void LoadBitmap();          //讀取所需的圖檔

    void LoadPlant();           //讀取植物的圖片

    void LoadPrice();           //讀取價錢的圖片

    void SetXY(int);            //設定卡片的位置

    int GetPrice();

    int GetWidth();

    int GetHeight();

    int GetX();

    int GetY();

    int GetID();

    void SetAvailible(bool);  //設定卡片是否可被選取

    bool isAvailible();       //回傳卡片是否可被選取

    void DelayCounter();

    void ResetCounter();

    bool CounterFinished();

    bool counting;

    void Reset();             //重設卡片的初始狀態
```

```
  private:

      int x, y;

      double y2;                //卡片等待時間黑幕的位置

      int ID;                   //卡片代表的植物種類

      int price;                //價錢

      int delay;                //等待所需時間

      int counter;              //計時器

      bool availible;

      CMovingBitmap   plant;

      CMovingBitmap   number[3][10]; // price picture

  };

}
```

LawnClear.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"
```

```cpp
#include "gamelib.h"

#include "LawnCleaner.h"

namespace game_framework {

    LawnCleaner::LawnCleaner() {

        x = 100;

        move = false;

    }

    //重設除草機的位置

    void LawnCleaner::Reset() {

        x = 100;

        move = false;

    }

    //設定除草機的位置

    void LawnCleaner::SetY(int a) {

        y = a * 98 + 80;

    }

    //取得除草機的位置

    int LawnCleaner::GetX() {

        return x;
```

```
    }

    //讀取除草機的圖片

    void LawnCleaner::LoadBitmap() {

        bmp_LawnCleaner.LoadBitmap(".\\BMP_RES\\image\\interface\\LawnCleaner
.bmp", RGB(0, 0, 0));

    }

    //移動除草機

    void LawnCleaner::OnMove() {

        if (move == true && x < 1000) {

            x += 10;

        }

    }

    //顯示除草機

    void LawnCleaner::OnShow() {

        bmp_LawnCleaner.SetTopLeft(x, y);

        bmp_LawnCleaner.ShowBitmap();

    }

    //讓除草機動起來

    void LawnCleaner::StartMove() {
```

```
        move = true;

    }

}
```

LawnCleaner.h

```
#pragma once

#pragma once

namespace game_framework {

    class LawnCleaner {

    public:

        LawnCleaner();

        void SetY(int);       //設定除草機的位置

        int GetX();           //取得除草機的位置

        void OnMove();        //處理除草機的移動

        void OnShow();        //顯示除草機

        void LoadBitmap();    //讀取除草機的圖片

        void StartMove();     //讓除草機開始移動

        void Reset();         //讓除草機回復初始狀態

    private:
```

```
        int x, y;

        bool move;            //儲存除草機目前的狀態

        CMovingBitmap       bmp_LawnCleaner;

    };

}
```

map.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "maps.h"

namespace game_framework {

    Maps::Maps() {

        x = 0;

        y = 0;

    }
```

```
    //讀取地圖

    void Maps::LoadBitmap() {

        bmp.LoadBitmap("BMP_RES/image/interface/background1unsodded_1.bmp");

    }

    //顯示地圖

    void Maps::OnShow(int x = 0, int y = 0) {

        bmp.SetTopLeft(x, y);

        bmp.ShowBitmap();

    }



    //int Maps::Left() {

        //return x;

    //}

}
```

map.h

```
#pragma once



namespace game_framework {
```

```
    class Maps {

    public:

        Maps();

        void LoadBitmap();                      // 載入圖形

        void LoadBitmap(std::string path);      // 載入圖形

        void OnShow(int, int);                  // 將圖形貼到畫面

    private:

        int x, y;                               // 圖形座標

        CMovingBitmap bmp;

    };

}
```

mouse.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"
```

```
#include "gamelib.h"

#include "mouse.h"

namespace game_framework {

   Mouse::Mouse() {

       which = 0;

   }

    //讀取所需圖片

   void Mouse::LoadBitmap() {

       Sunflower.LoadBitmap(".\\BMP_RES\\image\\plants\\SunFlower\\SunFlower

_0.bmp",RGB(0,0,0));

       peashooter.LoadBitmap(".\\BMP_RES\\image\\plants\\PeaShooter\\PeaShoo

ter_0.bmp", RGB(0, 0, 0));

       Wallnut.LoadBitmap(".\\BMP_RES\\image\\plants\\WallNut\\WallNut_0.bmp

", RGB(0, 0, 0));

       squash.LoadBitmap(".\\BMP_RES\\image\\plants\\Squash\\Squash_0.bmp",

RGB(0, 0, 0));

       cherrybomb.LoadBitmap(".\\BMP_RES\\image\\plants\\CherryBomb\\CherryB

omb_0.bmp", RGB(0, 0, 0));
```

```
        snowpea.LoadBitmap(".\\BMP_RES\\image\\plants\\SnowPea\\SnowPea_0.bmp
", RGB(0, 0, 0));
        repeater.LoadBitmap(".\\BMP_RES\\image\\plants\\Repeater\\Repeater_0.
bmp", RGB(0, 0, 0));
        shovel.LoadBitmap(".\\BMP_RES\\image\\interface\\Shovel.bmp", RGB(0,
0, 255));
    }
    void Mouse::OnMove() {
    }
     //設定游標目前要顯示的圖片
    void Mouse::SetWhich(int a) {
        which = a;
    }
     //讓圖片跟著游標移動
    void Mouse::SetXY(int xx, int yy) {
        x = xx;
        y = yy;
        if (which == 4 || which == 5) {
            x -= 10;
```

```
    }

    // if (which == 4) {

    //  y -= 150;

    // }

}

 //顯示圖片

void Mouse::OnShow() {

    if (which != 0) {

        CMovingBitmap*  pointer;

        switch (which) {

        case 1: pointer = &Sunflower;   break;

        case 2: pointer = &peashooter;  break;

        case 3: pointer = &Wallnut;     break;

        case 4: pointer = &squash;      break;

        case 5: pointer = &cherrybomb;  break;

        case 6: pointer = &snowpea;     break;

        case 7: pointer = &repeater;    break;

        case 8: pointer = &shovel;      break;

        }
```

```
            if (which != 8) {

                pointer->SetTopLeft(x-30, y-30);

                pointer->ShowBitmap();

            }

            else {

                pointer->SetTopLeft(x - 30, y - 15);

                pointer->ShowBitmap();

            }

        }

    }

}
```

mouse.h

```
#pragma once

//class CMovingBitmap;

namespace game_framework {

    class Mouse {

    public:

        Mouse();
```

```
        void LoadBitmap();

        void OnMove();

        void SetWhich(int);

        void OnShow();

        void SetXY(int, int);

    private:

        int x, y;

        int which;

        CMovingBitmap   Sunflower;

        CMovingBitmap   peashooter;

        CMovingBitmap   Wallnut;

        CMovingBitmap   squash;

        CMovingBitmap   cherrybomb;

        CMovingBitmap   snowpea;

        CMovingBitmap   repeater;

        CMovingBitmap   shovel;

    };

}
```

mygame.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

#include <memory>









namespace game_framework {

    bool YouWin = false;

    int gamelevel=1;

    bool isGameOver;

    //bool myrunning=true;

    bool frist_load = true;
```

```
//////////////////////////////////////////////////////////////////////////
////
// 這個 class 為遊戲的遊戲開頭畫面物件
//////////////////////////////////////////////////////////////////////////
////

CGameStateInit::CGameStateInit(CGame *g)
    : CGameState(g)
{

}

//讀取音效檔
void CGameStateInit::LoadAudio() {
    CAudio::Instance()->Load(AUDIO_MENU,
".\\Sounds\\ZombiesOnYourLawn.wav");
```

```
	CAudio::Instance()->Load(AUDIO_EVIL_LAUGH,
".\\Sounds\\evillaugh.wav");
   }


   // is finished
   void CGameStateInit::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。
	//
	ShowInitProgress(0); // 一開始的 loading 進度為 0%
	//loading_picture.ShowBitmap();
	//
	// 開始載入資料
	//

	conditionA = false;
	conditionB = false;
```

```
        LoadAudio();

        logo.LoadBitmap(Background);

        adventure_block.LoadBitmap(Adventure, RGB(255, 255, 255));

        adventure_block2.LoadBitmap(".\\BMP_RES\\image\\interface\\adventure2
.bmp", RGB(255, 255, 255));

        //Sleep(300);    // 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep

        //

        // 此 OnInit 動作會接到 CGameStaterRun::OnInit()，所以進度還沒到 100%

        //

    }



    void CGameStateInit::OnBeginState()

    {

        play_Audio = false;



    }



    // is finsihed

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

```
{

    const char KEY_ESC = 27;

    const char KEY_SPACE = ' ';

    if (nChar == KEY_SPACE)

        GotoGameState(GAME_STATE_RUN);      // 切換至 GAME_STATE_RUN

    else if (nChar == KEY_ESC)        // Demo 關閉遊戲的方法

        PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0); // 關閉遊戲

}




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

{

    // 處理滑鼠和選單的互動

    bool conditionA1 = (point.y - 125 <= 35 * (point.x - 350) / 250);

    bool conditionA2 = (point.y - 68 >= -57 * (point.x - 351));

    bool conditionA3 = (point.y - 68 >= 17 * (point.x - 351) / 261);

    bool conditionA4 = (point.y - 160 <= -75 * (point.x - 600) / 12);

    bool conditionB1 = (point.y - 70 >= -20 * (point.x - 400) / 25);
```

```
	bool conditionB2 = (point.y >= 50);

	bool conditionB3 = (point.y - 50 >= 31 * (point.x - 520) / 26);

	bool conditionB4 = (point.y - 70 <= 11 * (point.x - 400) / 146);



	if (point.y < 240 && point.y > 100 && point.x < 680 && point.x > 390)
{



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

		CAudio::Instance()->Play(AUDIO_EVIL_LAUGH, false);

	}



	conditionA = (point.y < 245 && point.y > 100 && point.x <680 &&
point.x > 390);

	conditionB = (conditionB1 && conditionB2 && conditionB3 &&
conditionB4);

	//menu.SetHighLight(conditionA || conditionB);

   }



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

```
{

    if (conditionA || conditionB) {

        //menu.Shine();

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

        CAudio::Instance()->Play(AUDIO_EVIL_LAUGH, false);

        GotoGameState(GAME_STATE_RUN);  // 切換至 GAME_STATE_RUN

    }




}



// is finish

void CGameStateInit::OnShow()

{



    Sleep(300);    // 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep

    logo.SetTopLeft(-100, 0);

    logo.ShowBitmap();

    if (conditionA) {
```

```
            adventure_block2.SetTopLeft(380, 100);

            adventure_block2.ShowBitmap();

        }else {

            adventure_block.SetTopLeft(380, 100);

            adventure_block.ShowBitmap();

        }




        if (play_Audio == false) {

            play_Audio = true;

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

        }

        //   //

        //   // Demo 螢幕字型的使用，不過開發時請盡量避免直接使用字型，改用
CMovingBitmap 比較好

        //   //

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

        //   CFont f, *fp;
```

```
	//	f.CreatePointFont(160,"Times New Roman"); // 產生 font f; 160 表示
16 point 的字
	//	fp = pDC->SelectObject(&f);    // 選用 font f
	//	pDC->SetBkColor(RGB(0, 0, 0));
	//	pDC->SetTextColor(RGB(255, 255, 0));
	//	pDC->TextOut(120, 220, "Please click mouse or press SPACE to
begin.");
	//	pDC->TextOut(5, 395, "Press Ctrl-F to switch in between window
mode and full screen mode.");
	//	if (ENABLE_GAME_PAUSE)
	//	 pDC->TextOut(5, 425, "Press Ctrl-Q to pause the Game.");
	//	pDC->TextOut(5, 455, "Press Alt-F4 or ESC to Quit.");
	//	pDC->SelectObject(fp);     // 放掉 font f (千萬不要漏了放掉)
	//	CDDraw::ReleaseBackCDC();    // 放掉 Back Plain 的 CDC
   }
```

```
    ///////////////////////////////////////////////////////////////////////////
////
    // 這個 class 為遊戲的結束狀態(Game Over)
    ///////////////////////////////////////////////////////////////////////////
////


    CGameStateOver::CGameStateOver(CGame *g)
        : CGameState(g)
    {


    }
    //讀取結果的圖檔
    void CGameStateOver::LoadBitmap() {
        ZombiesWon.LoadBitmap(".\\BMP_RES\\image\\interface\\ZombiesWon0.bmp"
);


        ZombiesWon.SetTopLeft(0, 0);
        ZombieLose.LoadBitmap(".\\BMP_RES\\image\\interface\\win0.bmp");
```

```
    ZombieLose.SetTopLeft(0, 0);

    FinalWin.LoadBitmap(".\\BMP_RES\\image\\interface\\finalwin.bmp");

    FinalWin.SetTopLeft(0, 0);

}



void CGameStateOver::OnMove()

{

    if (gamelevel == 11) {

        //Sleep(3000);



        //GotoGameState(GAME_STATE_INIT);

    }

    // counter--;

    // if (counter < 0)

        // GotoGameState(GAME_STATE_RUN);

}



void CGameStateOver:: 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


    if (nChar == KEY_LEFT) {
        GotoGameState(GAME_STATE_INIT);
    }
    if (nChar == KEY_RIGHT) {
        GotoGameState(GAME_STATE_RUN);
    }



}

void CGameStateOver::OnBeginState()
{

    counter = 30 * 5; // 5 seconds
    if(YouWin==true){
```

```
            gamelevel++;

            CAudio::Instance()->Play(AUDIO_WIN_MUSIC, false);

        }

        else

            CAudio::Instance()->Play(AUDIO_LOSE_MUSIC, false);

    }

    // is finished

    void CGameStateOver::LoadAudio()

    {

        CAudio::Instance()->Load(AUDIO_LOSE_MUSIC,
".\\Sounds\\losemusic.wav");

        CAudio::Instance()->Load(AUDIO_WIN_MUSIC, ".\\Sounds\\winmusic.wav");

    }

    void CGameStateOver::OnInit()

    {

        LoadAudio();

        ShowInitProgress(66);

        //Sleep(300);

        ShowInitProgress(100);
```

```
    LoadBitmap();

}



void CGameStateOver::OnShow()

{

    if(isGameOver)

        if (YouWin == false) {

            ZombiesWon.ShowBitmap();

        }

        else {

            if (gamelevel == 11) {

                FinalWin.ShowBitmap();



                //GotoGameState(GAME_STATE_INIT);

                //gamelevel = 1;

            }

            else {

                ZombieLose.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)
    {
        srand((unsigned)time(NULL));
    }

    CGameStateRun::~CGameStateRun()
    {
    }
```

```
void CGameStateRun::OnBeginState()
{
    if (gamelevel == 11) {
        //Sleep(3000);
        gamelevel = 1;
        //GotoGameState(GAME_STATE_INIT);
    }

    show_menu = false;
    background[gamelevel-1].SetTopLeft(-500, 0);                    // 設定
背景的起始座標
    //help.SetTopLeft(0, SIZE_Y - help.Height());   // 設定說明圖的起始座標
    CAudio::Instance()->Play(AUDIO_MAIN_MUSIC, true); //撥放遊戲背景音樂
    sunback.SetTopLeft(-400, 10);
    zombiesone[0].SetTopLeft(640, 150);
    zombiesone[1].SetTopLeft(660, 200);
    zombiesone[2].SetTopLeft(640, 250);
    zombiesone[3].SetTopLeft(660, 300);
```

```
	zombiesone[4].SetTopLeft(640, 350);



	run_menu.SetTopLeft(230, 50);

	run_con[0].SetTopLeft(315, 180);

	run_con[1].SetTopLeft(315, 180);

	run_return[0].SetTopLeft(315, 250);

	run_return[1].SetTopLeft(315, 250);

	//設定和滑鼠相關的變數

	SunCounter = 0;                                    //從空中掉落太陽的計時器

	ZombieCounter = 0;

	selected = false;

	ChoosedCard = -1;

	ChoosedPlant = -1;

	isGameOver=false;

	YouWin=false;

	show_back = 0;

	show_con = 0;

	//確保所有的 vector 清空
```

```
monster.clear();

// plants.clear();

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

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

        PlantManager[i][j] = 0;

        PlantClass[i][j] = Plants(0, j, i);

    }

}

peas.clear();

suns.clear();

seed.Reset();



//重設除草機

if(gamelevel>=3){

    for(int i=0;i<5;i++)

    LawnCleaner[i].Reset();

}



wave = 0;
```

```
    // CAudio::Instance()->Play(AUDIO_LAKE, true);   // 撥放 WAVE
    // CAudio::Instance()->Play(AUDIO_DING, false);  // 撥放 WAVE
    // CAudio::Instance()->Play(AUDIO_NTUT, true);   // 撥放 MIDI
}

void CGameStateRun::OnInit()          // 遊戲的初值及圖形設定
{
    //TODO:
    LoadAudio();
     ShowInitProgress(33);
     Sleep(500); // 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep
     ShowInitProgress(50);


    myrunning = true;
    if(gamelevel==1 && frist_load == true){
        background[0].LoadBitmap(Background1row);     // 載入背景的圖形
```

```
            background[1].LoadBitmap("BMP_RES\\image\\interface\\background1u
nsodded2.bmp");
            background[2].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
            background[3].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
            background[4].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
            background[5].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
            background[6].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
            background[7].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
            background[8].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
            background[9].LoadBitmap("BMP_RES\\image\\interface\\background1.
bmp");
```

```
            background[10].LoadBitmap("BMP_RES\\image\\interface\\night1.bmp"
);
            menu.LoadBitmap("BMP_RES\\image\\interface\\menu\\mainmenu\\menu0
.bmp", RGB(0, 0, 0));
            run_menu.LoadBitmap("BMP_RES\\image\\interface\\choicemenu\\Optio
nsMenuback8.bmp", RGB(0, 0, 0));
            for (int i = 0; i < 2; i++) {
                char FILENAME[100];
                sprintf(FILENAME,
"BMP_RES\\image\\interface\\menu\\mainmenu\\back%d.bmp", i);
                run_con[i].LoadBitmap(FILENAME, RGB(0, 0, 0));      //繼續
                sprintf(FILENAME,
"BMP_RES\\image\\interface\\menu\\mainmenu\\menu%d.bmp", i);
                run_return[i].LoadBitmap(FILENAME, RGB(0, 0, 0)); // 回選單
            }
            int temp[] = {1,2,3,4,5,6,7};
```

```
            seed.Load(7, temp);



            for (int i = 0; i < 5; i++) {                    //載入殭屍

                zombiesone[i].LoadBitmap(".\\BMP_RES\\image\\zombies\\Normal

Zombie\\Zombie_0.bmp", RGB(0, 0, 0));

                LawnCleaner[i].LoadBitmap();

            }



            mouse.LoadBitmap();

            sunback.LoadBitmap("BMP_RES/image/interface/SunBack.bmp", RGB(0,

0, 0));

            shovel.LoadBitmap();

            frist_load = false;

        }

    }



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

    {
```

```
        if(myrunning)

        {

            //  開始的移動畫面

            if (background[gamelevel-1].Left() < -80){

                background[gamelevel-1].SetTopLeft(background[gamelevel-

1].Left() + 10, 0);

                // TODO:

                //for



            }

            if(gamelevel>=3){

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

                    LawnCleaner[i].SetY(i);

                    LawnCleaner[i].OnMove();

                }

            }
```

```
            if (sunback.Left() < 100)
                sunback.SetTopLeft(sunback.Left() + 10, 10);


            // 放置殭屍 (right)
            for (int i = 0; i < 5; i++) {
                if (zombiesone[i].Left() < 1040)
                    zombiesone[i].SetTopLeft(zombiesone[i].Left() + 10, 150 +
i * 50);
            }


            //每隔 7 秒從空中產生一個太陽
            SunCounter++;
            if (SunCounter == 300) {
                SunCounter = 0;
                suns.push_back(Sun(rand() % 400 + 100, rand() % 300 + 100,
false));
            }
```

```
            ZombieCounter++;

            if (wave < gamelevel*1 && wave >= 0) {

                ///每隔 10 秒產生 1~3 隻隨機種類的殭屍

                if (ZombieCounter == 200-gamelevel*5) {

                    wave++;

                    if (awooga == false) {

                        CAudio::Instance()->Play(AUDIO_AWOOGA, false);

                        awooga = true;

                    }

                    ZombieCounter = 0;

                    int groan = rand() % 6;

                    CAudio::Instance()->Play(AUDIO_GROAN_1 + groan, false);



                    if(gamelevel==1){

                            monster.push_back(make_shared<Zombies>(1, 3,
800));

                    }
```

```
                if(gamelevel==2){ // 3row

                    for (int i = 0; i < 2; i++) {

                        monster.push_back(make_shared<Zombies>(rand()%3+1
,rand()%3+2, 800));// for demo

                    }

                }

                if(gamelevel>=3){ // 5 row

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

                        monster.push_back(make_shared<Zombies>(rand()%3+1
,rand()%5+1, 800));// for demo

                    }

                }



            }

        }

        // TODO:

        if(wave == gamelevel*1){

            int checkwin=1;

            for(auto &itz:monster){
```

```
                    if(itz->GetX()!=1000){

                        checkwin = 0;

                        break;

                    }

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

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

                            if (PlantClass[i][j].isFinished() == false &&
PlantClass[i][j].GetID()==4 && PlantClass[i][j].WhichAction() == 2) {

                                checkwin = 0;

                                break;

                            }

                        }

                    }

                }

                if(checkwin==1){

                    YouWin = true;

                    isGameOver = true;

                }

            }
```

```
            for (auto &itz : monster) {
                //處理所有殭屍的動作
                itz->OnMove();
                if (itz->GetX() < 50) {
                    YouWin = false;
                    isGameOver = true;                                    //
如果殭屍跑進家裡，遊戲結束
                }
                //尋找殭屍可攻擊的第一隻植物
                int closest = 10;
                for (int i = (itz->GetX() - 80) / 75; i >= 0; i--) {
                    if (PlantManager[itz->GetRow()][i] > 0) {
                        closest = i;
                        break;
                    }
                }
```

```
                if (itz->isAlive() == true) {

                    //如果殭屍被除草機撞到就馬上死亡

                    if(gamelevel>=3)

                        if (LawnCleaner[itz->GetRow()].GetX() > itz->GetX() +
30 && LawnCleaner[itz->GetRow()].GetX() < itz->GetX() + 100) {

                            LawnCleaner[itz->GetRow()].StartMove();

                            CAudio::Instance()->Play(AUDIO_LAWN_MOWER,
false);

                            itz->GoToDie();

                        }



                    bool found = false;

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

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

                            if (PlantClass[i][j].GetRow() == itz->GetRow() &&
PlantClass[i][j].GetColumn() == closest) {

                                if (PlantClass[i][j].GetX() <= itz->GetX()+80
&& PlantClass[i][j].GetX() >= itz->GetX() + 30) {
```

```
                                    itz-
>SetStatus(2);                                  //如果離殭屍最近的植物進入攻擊範圍就
進入攻擊狀態
                                    found = true;
                                }
                                else {
                                    itz->SetStatus(1);
                                }
                                //如果殭屍正在攻擊狀態且攻擊冷卻時間結束，植物就受到攻
擊
                                if (itz->GetStatus() == 2 && itz->Attack() ==
true) {
                                    int chomp = rand() % 3;
                                    CAudio::Instance()->Play(AUDIO_CHOMP_1 +
chomp, false);
                                    PlantClass[i][j].BeingAttacked();
                                    if (PlantClass[i][j].isAlive() == false)
{
                                        PlantClass[i][j] = Plants(0, i,j);
```

```
                                        PlantManager[i][j] = 0;

                                        itz-
>SetStatus(1);                                  //如果植物被殭屍吃掉了，殭屍馬上恢復普
通狀態

                                    }

                                }



                            }

                        }

                    }



                    if (found == false) {

                        itz->SetStatus(1);

                    }

                }

                //殭屍死亡

                // TODO:

                if (itz->isFinished() == true) {
```

```
                itz->GoToDie();

            }









        }

        //處理所有植物的動作

        bool ErasePlant = false;

        if(isGameOver==true)

            GameOver();



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

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

                if (PlantClass[i][j].isAlive() == false)

{                           //如果植物的生命為零，設定植物的死亡

                    ErasePlant = true;

                    PlantManager[i][j] = 0;

                    continue;

                }

                PlantClass[i][j].OnMove();
```

```
                    //處理向日葵的動作
                    if (PlantClass[i][j].GetID() == 1) {
                        PlantClass[i][j].SetCounterOn(true);
                        if (PlantClass[i][j].isAction() == true)
{                           //時間到了就產生一個太陽
                            suns.push_back(Sun(PlantClass[i][j].GetX(),
PlantClass[i][j].GetY(), true));
                        }
                        continue;
                    }
                    //處理一般豌豆的動作
                    if (PlantClass[i][j].GetID() == 2) {
                        bool FoundZombie = false;
                        for (auto &itz : monster) {
                            if (PlantClass[i][j].GetRow() ==
PlantClass[i][j].GetRow() && PlantClass[i][j].GetX() + 50 >=
PlantClass[i][j].GetX()) {
                                FoundZombie = true;
```

```
                                if (PlantClass[i][j].GetX() + 50 >
PlantClass[i][j].GetX()) {
                                    PlantClass[i][j].SetCounterOn(true);
                    //如果找到可攻擊的殭屍就進入攻擊狀態
                                }
                                else {
                                    PlantClass[i][j].SetCounterOn(false);
                                }
                            }

                            if (PlantClass[i][j].isAction() == true) {
                                CAudio::Instance()->Play(AUDIO_SHOOT, false);
                                peas.push_back(Pea(PlantClass[i][j].GetX() +
50, PlantClass[i][j].GetRow(), 0)); //如果攻擊冷卻時間到了就射出一顆豆子
                            }
                            if (FoundZombie == false) {
                                PlantClass[i][j].SetCounterOn(false);
                            }
                        }
```

```
                    }



                    //處理 ID 4 的動作



                    if (PlantClass[i][j].GetID() == 4) {

                        for (auto &itz : monster) {

                            if (itz->GetRow() == PlantClass[i][j].GetRow() &&
(itz->GetX()-80)/75 == j+1 || (itz->GetX()-80)/75 == j+2){

                                PlantClass[i][j].SetTargetX(10);

                                if ((itz->GetX() )  <
PlantClass[i][j].GetX()+10) {

                                    itz->SetSnowCounter();

                                }



                                if (PlantClass[i][j].WhichAction() == 2) {

                                    itz->GoToDie();

                                }

                            }

                        }
```

```
                    if (PlantClass[i][j].WhichAction() == 3){

                        PlantClass[i][j]= Plants(0,i,j);

                        PlantManager[i][j]= 0;

                    }



                }

                //處理櫻桃的動作

                if (PlantClass[i][j].GetID() == 5) {

                    if (PlantClass[i][j].WhichAction() == 2) {

                        CAudio::Instance()->Play(AUDIO_CHERRY_BOMB,
false);

                        for (auto &itz : monster) {

                            if (abs(itz->GetRow() -
PlantClass[i][j].GetRow()) <= 1 && itz->GetX() + 50 <
PlantClass[i][j].GetX() + 100 && itz->GetX() + 50 > PlantClass[i][j].GetX()
- 100) {

                                itz-
>BoomToDie();                                //如果殭屍在櫻桃的爆炸範圍內就會瞬間
被炸死
```

```
                                }

                            }

                        }

                        if (PlantClass[i][j].WhichAction() == 3){

                            PlantClass[i][j]= Plants(0,i,j);

                            PlantManager[i][j]= 0;

                        }



                    }

                    //處理冷凍豌豆的動作

                    if (PlantClass[i][j].GetID() == 6) {

                        bool FoundZombie = false;

                        for (auto &itz : monster) {

                            if (itz->GetRow() == PlantClass[i][j].GetRow() &&
itz->GetX() + 50 >= PlantClass[i][j].GetX()) {

                                FoundZombie = true;

                                if (itz->GetX() + 50 >
PlantClass[i][j].GetX()) {
```

```
                                PlantClass[i][j].SetCounterOn(true);

                    //如果有殭屍在攻擊範圍內就進入攻擊模式

                            }

                            else {

                                PlantClass[i][j].SetCounterOn(false);

                            }

                        }

                    }

                    if (PlantClass[i][j].isAction() == true) {

                        CAudio::Instance()->Play(AUDIO_SHOOT, false);

                        peas.push_back(Pea(PlantClass[i][j].GetX() + 50,
PlantClass[i][j].GetRow(), 1));//如果攻擊冷卻時間到了就射出一顆豆子

                    }

                    if (FoundZombie == false) {

                        PlantClass[i][j].SetCounterOn(false);

                    }

                }

                //處理連射豌豆的動作

                if (PlantClass[i][j].GetID() == 7) {
```

```
                        bool FoundZombie = false;

                        for (auto &itz :monster) {

                            if (itz->GetRow() == PlantClass[i][j].GetRow() &&
itz->GetX() + 50 >= PlantClass[i][j].GetX()) {

                                FoundZombie = true;

                                if (itz->GetX() + 50 >
PlantClass[i][j].GetX()) {

                                    PlantClass[i][j].SetCounterOn(true);

                    //如果有殭屍在攻擊範圍內就進入攻擊模式

                                }

                                else {

                                    PlantClass[i][j].SetCounterOn(false);

                                }

                            }

                        }

                        if (PlantClass[i][j].isAction() == true) {

                            CAudio::Instance()->Play(AUDIO_SHOOT, false);

                            peas.push_back(Pea(PlantClass[i][j].GetX() + 50,
PlantClass[i][j].GetRow(), 0));//如果攻擊冷卻時間到了就射出一顆豆子
```

```
                        }

                        if (FoundZombie == false) {

                            PlantClass[i][j].SetCounterOn(false);

                        }

                    }

                }

                // 處理所有豆子

                for (vector<Pea>::iterator itpea = peas.begin(); itpea !=
peas.end(); itpea++) {

                    itpea->OnMove();

                    bool HitZombie = false;

                    int mi = 1000;

                    for (auto &itz : monster) {



                        if (itpea->GetRow() == itz->GetRow() && itpea->GetX()
> itz->GetX() + 75 && itpea->GetX() < itz->GetX() + 110) {

                            if (mi > itz->GetX()) {

                                mi = itz->GetX();

                                //itzz = itz;
```

```
                                HitZombie = true;

                                itpea->SetHitZombie(true);

                                //處理擊中殭屍後的動作

                                if (itz->GetID() == 3 && itz->GetLife() > 10)
{

                                    int sound = rand() % 2;

                                    CAudio::Instance()-
>Play(AUDIO_HIT_BUCKET_1 + sound, false);

                                }

                                else {

                                    int sound = rand() % 3;

                                    CAudio::Instance()->Play(AUDIO_SPLAT_1 +
sound, false);

                                }

                                itz->Hitted(itpea->MyType());

                            }

                        }

                        if (itpea->GetX() > 880) {

                            itpea->SetHitZombie(true);
```

```
                        }

                    }

                }



                //處理所有太陽的動作

                vector<Sun>::iterator itss;

                bool EraseSun = false;

                for (vector<Sun>::iterator its = suns.begin(); its !=
suns.end(); its++) {

                    if (its == suns.begin()) {

                        its->MoveAnime();

                    }

                    its->OnMove();

                    if (its->isFinished()) {

                        itss = its;

                        EraseSun = true;

                    }

                }

                if (EraseSun == true) {
```

```
                    suns.erase(itss);

                    EraseSun = false;

                }

                seed.OnMove();

            }

        }

    }



    //讀取所需的音效檔

    void CGameStateRun::LoadAudio() {

        CAudio::Instance()->Load(AUDIO_MAIN_MUSIC,
".\\Sounds\\mainmusic.wav");

        CAudio::Instance()->Load(AUDIO_AWOOGA, ".\\Sounds\\awooga.wav");

        for (int i = 1; i <= 6; i++) {

            char FILENAME[100];

            sprintf(FILENAME, ".\\Sounds\\groan%d.wav", i);

            CAudio::Instance()->Load(AUDIO_GROAN_1 + i - 1, FILENAME);

        }

        for (int i = 1; i <= 3; i++) {
```

```
            char FILENAME[100];

            sprintf(FILENAME, ".\\Sounds\\chomp%d.wav", i);

            CAudio::Instance()->Load(AUDIO_CHOMP_1 + i - 1, FILENAME);

        }

        CAudio::Instance()->Load(AUDIO_CHERRY_BOMB,
".\\Sounds\\cherrybomb.wav");

        CAudio::Instance()->Load(AUDIO_LAWN_MOWER,
".\\Sounds\\lawnmower.wav");

        CAudio::Instance()->Load(AUDIO_PLANT, ".\\Sounds\\plant.wav");

        CAudio::Instance()->Load(AUDIO_COLLECT_SUN, ".\\Sounds\\points.wav");

        CAudio::Instance()->Load(AUDIO_CHOOSE_CARD,
".\\Sounds\\seedlift.wav");

        CAudio::Instance()->Load(AUDIO_HIT_BUCKET_1,
".\\Sounds\\shieldhit1.wav");

        CAudio::Instance()->Load(AUDIO_HIT_BUCKET_2,
".\\Sounds\\shieldhit2.wav");

        CAudio::Instance()->Load(AUDIO_SHOVEL, ".\\Sounds\\shovel.wav");

        CAudio::Instance()->Load(AUDIO_SPLAT_1, ".\\Sounds\\splat.wav");

        CAudio::Instance()->Load(AUDIO_SPLAT_2, ".\\Sounds\\splat2.wav");
```

```
        CAudio::Instance()->Load(AUDIO_SPLAT_3, ".\\Sounds\\splat3.wav");

        CAudio::Instance()->Load(AUDIO_SHOOT, ".\\Sounds\\puff.wav");

        CAudio::Instance()->Load(AUDIO_FINALWAVE,
".\\Sounds\\finalwave.wav");
    }




    //進入 GameStateOver，並將 vector 清空，否則第二次完會產生錯誤
    void CGameStateRun::GameOver()
    {
        monster.clear();
        suns.swap(vector<Sun>());
        peas.swap(vector<Pea>());
        CAudio::Instance()->Stop(AUDIO_MAIN_MUSIC);
        GotoGameState(GAME_STATE_OVER);
    }


    //密技
```

```
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



    if (nChar == KEY_UP) {

        for(auto &it:monster)

            it->GoToDie();

    }

    else if (nChar == KEY_DOWN) {

        for (int i = 0; i < 40; i++) {

            seed.GotSun();

        }

    }

    else if (nChar == KEY_LEFT) {

        seed.ResetCD();

    }
```

```
    else if (nChar == KEY_RIGHT) {

        for(auto &it:monster)

            it->Faster();

        //myrunning=!myrunning;

    }

}



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



    // if (nChar == KEY_LEFT)

    //  eraser.SetMovingLeft(false);

    // if (nChar == KEY_RIGHT)

    //  eraser.SetMovingRight(false);

    // if (nChar == KEY_UP)
```

```
        //  eraser.SetMovingUp(false);

        // if (nChar == KEY_DOWN)

        //  eraser.SetMovingDown(false);



        // gamemap.OnKeyDown(nChar);

    }



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

    {

        if (!selected) {

            //處理 menu 的動作

            if (point.x >= 695 && point.x <= 800 && point.y >= 0 && point.y
<= 40 ) {

                myrunning = false;

                show_menu = true;

                selected = true;

            }

            //處理點擊太陽的動作
```

```
            bool GotSun = false;

            for (vector<Sun>::iterator its = suns.begin(); its != suns.end();
its++) {

                if (point.x >= its->GetX() && point.x <= its->GetX() + its-
>GetWidth() && point.y >= its->GetY() && point.y <= its->GetY() + its-
>GetHeight()) {

                    its->PickUp();

                    CAudio::Instance()->Play(AUDIO_COLLECT_SUN, false);

                    seed.GotSun();

                    GotSun = true;

                    break;

                }

            }

            //處理選擇卡片的動作

            if (point.x >= 0 && point.x <= 95 && point.y >= 50 && point.y <=
470 && GotSun == false) {

                ChoosedCard = (point.y - 50) / 60;

                if(ChoosedCard < gamelevel+1){

                    ChoosedPlant = seed.GetCardID(ChoosedCard);
```

```
                    if (seed.isCardAvailible(ChoosedCard)) {

                        CAudio::Instance()->Play(AUDIO_CHOOSE_CARD, false);

                        selected = true;

                        mouse.SetXY(point.x, point.y);

                        mouse.SetWhich(ChoosedPlant); //讓游標的樣子變成準備種植的
植物



                    }

                }



            }

            //處理點擊產子的動作

            if (point.x >= 135 && point.x <= 211 && point.y >= 10 && point.y
<= 44 && GotSun == false) {

                if (shovel.isChoosed() == false) {

                    CAudio::Instance()->Play(AUDIO_SHOVEL, false);

                    shovel.SetChoosed(true);

                    selected = true;

                    mouse.SetXY(point.x, point.y);
```

```
                    mouse.SetWhich(8); //讓游標的樣子變成鏟子

                }

            }









        }

        else if (selected == true &&  myrunning == false) {

            //實現 back to game 的功能

            if (point.x >= 315 && point.x <= 550 && point.y >= 180 && point.y
<= 230 && show_menu == true && myrunning == false) {

                myrunning = true;

                selected = false;

                show_menu = false;

            }

            //實現 back to menu 的功能

            if (point.x >= 315 && point.x <= 550 && point.y >= 250 && point.y
<= 300 && show_menu == true && myrunning == false) {

                monster.clear();

                suns.swap(vector<Sun>());
```

```
                peas.swap(vector<Pea>());

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

                myrunning = !myrunning;

                selected = false;

                show_menu = false;

                GotoGameState(GAME_STATE_INIT);



            }



        }

        else if (selected) {

            if (point.x >= 172 && point.x <= 800 && point.y >= 80 && point.y
<= 580) {

                //實現用鏟子移除植物的功能



                if (shovel.isChoosed() == true) {

                    PlantManager[(point.y - 80) / 98][(point.x - 172) / 80] =
0;

                    bool ErasePlant = false;
```

```
                    PlantClass[(point.y - 80) / 98][(point.x - 172) / 80] =
Plants(0, (point.x - 172) / 80, (point.y - 80) / 98);



                }



                else if (shovel.isChoosed() == false) {

                    //植物功能

                    if (PlantManager[(point.y - 80) / 98][(point.x - 172) /
80] == 0) {

                        PlantManager[(point.y - 80) / 98][(point.x - 172) /
80] = ChoosedPlant;

                        PlantClass[(point.y - 80) / 98][(point.x - 172) / 80]
= Plants(ChoosedPlant, (point.x- 172) / 80, (point.y - 80) / 98);

                        CAudio::Instance()->Play(AUDIO_PLANT, false);

                        // TODO:

                        seed.ResetCardCounter(ChoosedCard);//冷卻

                        seed.Buy(ChoosedCard);



                    }
```

```
            }

        }

        mouse.SetWhich(0);

        selected = false;

        ChoosedCard = -1;

        shovel.SetChoosed(false);

    }




    //  touch meau then myrunning ==false

    // 然後判斷點擊位置 回到主選單或繼續



}



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
{
    if (selected) {
        mouse.SetXY(point.x, point.y);
    }
    if (show_menu) {
        if (point.x >= 315 && point.x <= 550 && point.y >= 180 && point.y
<= 230) {
            show_con = 1;
        }
        else {
            show_con = 0;
        }
        //實現 back to menu 的功能
        if (point.x >= 315 && point.x <= 550 && point.y >= 250 && point.y
<= 300) {
            show_back = 1;
        }
        else {
```

```
                show_back = 0;

            }

        }

    }



    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

    {



        for (auto &itz : monster) {

            itz->Faster();

        }

    }
```

```
void CGameStateRun::OnShow()

{

    if (wave >= gamelevel * 1 && gamelevel >= 3) {// 第三關後才有黑夜

        background[10].SetTopLeft(background[gamelevel - 1].Left(),0);

        background[10].ShowBitmap();

    }

    else {

        background[gamelevel - 1].ShowBitmap();  // 貼上背景圖

    }







    seed.OnShow(gamelevel+1);



    for (int  i = 0; i < 5; i++)

    {

        if(background[gamelevel-1].Left()>=-80 && gamelevel>=3)

            LawnCleaner[i].OnShow();

    }

    if (background[gamelevel - 1].Left() >= -80) {
```

```
            menu.SetTopLeft(695,0);

            menu.ShowBitmap(0.3);

        }









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

            // Show plants

            if(isGameOver==false){

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

                    bool ErasePlant = false;

                    if (PlantClass[i][j].GetRow() == i){

                        PlantClass[i][j].OnShow();

                        if (PlantClass[i][j].isFinished() == true)
{               //讓植物在死亡後或是動作結束後(葫蘆和櫻桃)被解構

                                ErasePlant = true;

                                PlantManager[PlantClass[i][j].GetRow()][Plant
Class[i][j].GetColumn()] = 0;

                                continue;
```

```
                }

                if (ErasePlant == true) {

                    ErasePlant = false;

                    }

                }

            }

        }

    }

    // Show zombies

    for (auto &itz : monster) {

        itz->OnShow();

    }

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

        vector<Pea>::iterator itpea;

        bool ErasePea = false;
```

```
            for (vector<Pea>::iterator it = peas.begin(); it != peas.end();
it++) {
                if (it->isFinished() == true) {            //讓豆子在擊中殭屍的動
畫結束後才會被解構
                    itpea = it;
                    ErasePea = true;
                    continue;
                }
                it->OnShow();
            }
            if (ErasePea == true) {
                peas.erase(itpea);
                ErasePea = false;
            }
        }

        for (int i = 0; i < 5; i++)
            zombiesone[i].ShowBitmap();
        shovel.OnShow();
```

```
        for (vector<Sun>::iterator its = suns.begin(); its != suns.end();
its++) {
            its->OnShow();
        }
        if (selected) {

            mouse.OnShow();
        }
        //show menu
        if (show_menu) {
            run_menu.ShowBitmap();
            run_con[show_con].ShowBitmap(0.7);
            run_return[show_back].ShowBitmap(0.7);

        }

    }
```

```
}
```

mygame.h

```
#include "maps.h"

#include "Plants.h"

#include "Sun.h"

#include "zombies.h"

#include "Seed.h"

#include "Pea.h"

#include "mouse.h"

#include "Shovel.h"

#include "LawnCleaner.h"

#include <vector>

#include <memory>

// #include "Selector.h"



namespace game_framework {


```

```
    ////////////////////////////////////////////////////////////////////////////
////

    // Constants

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


    enum AUDIO_ID {             // 定義各種音效的編號

        AUDIO_AWOOGA,

        AUDIO_CHERRY_BOMB,

        AUDIO_CHOMP_1,

        AUDIO_CHOMP_2,

        AUDIO_CHOMP_3,

        AUDIO_EVIL_LAUGH,

        AUDIO_FINALWAVE,

        AUDIO_GROAN_1,

        AUDIO_GROAN_2,

        AUDIO_GROAN_3,

        AUDIO_GROAN_4,

        AUDIO_GROAN_5,
```

```
    AUDIO_GROAN_6,

    AUDIO_LAWN_MOWER,

    AUDIO_LOSE_MUSIC,

    AUDIO_MAIN_MUSIC,

    AUDIO_MENU,

    AUDIO_PLANT,

    AUDIO_COLLECT_SUN,

    AUDIO_CHOOSE_CARD,

    AUDIO_HIT_BUCKET_1,

    AUDIO_HIT_BUCKET_2,

    AUDIO_SHOVEL,

    AUDIO_SPLAT_1,

    AUDIO_SPLAT_2,

    AUDIO_SPLAT_3,

    AUDIO_SHOOT,

    AUDIO_WIN_MUSIC

};
```

```
    ///////////////////////////////////////////////////////////////////////////
////
    // 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
        //TODO:
```

```
        void OnMouseMove(UINT nFlags, CPoint point);                    // 處理
滑鼠的動作
        void LoadAudio();                                               // 讀取所需
的音效檔



    protected:
        void OnShow();                                  // 顯示這個狀態的遊戲畫面
    private:
        CMovingBitmap   logo;                               // 冒險模式畫面
        CMovingBitmap   adventure_block;                // 冒險模式
        CMovingBitmap   adventure_block2;
        //CMovingBitmap   loading_picture;              // 冒險模式


        // TODO:
        // Selector menu;
         bool conditionA;                               // 檢查游標是否有碰到開始遊戲
的按鈕
```

```
        bool conditionB;                                // 檢查游標是否有碰到開始遊戲
的按鈕
        bool play_Audio;                                // 儲存是否已經開始撥放背景音
樂的 flag
        bool load_Audio;                                // 儲存是否已經讀取音效檔的
flag
       // CMovingBitmap   help;                            // help 的圖檔
   };

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



   typedef shared_ptr<Zombies> ZombiesPtr;
```

```
class CGameStateRun : public CGameState {

public:

    CGameStateRun(CGame *g);

    ~CGameStateRun();

    void OnBeginState();                            // 設定每次重玩所需的變數

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

    void OnKeyDown(UINT, UINT, UINT);

    void OnKeyUp(UINT, UINT, UINT);

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

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

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

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

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

    void LoadAudio();                           // 讀取所需的音效檔

    void GameOver();

protected:

    void OnMove();                                  // 移動遊戲元素

    void OnShow();                                  // 顯示這個狀態的遊戲畫面

private:
```

```
	// finish
	CMovingBitmap background[11];	// 背景圖
	CMovingBitmap zombiesone[5];	// 殭屍 右圖
	CMovingBitmap sunback;
	CMovingBitmap menu;
	CMovingBitmap run_menu;
	CMovingBitmap run_con[2];	//繼續
	CMovingBitmap run_return[2];	// 回選單
	//CMovingBitmap	loading_picture;
	// TODO:
	Seed		seed;				// 視窗上方管理卡片的物件
	Mouse		mouse;				// 用來顯示目前選取的東西
	Shovel		shovel;				// 鏟子
	LawnCleaner	LawnCleaner[5];			// 除草機
	std::vector< shared_ptr<Zombies> >  monster;	// 儲存所有殭屍的 vector
```

```
        Plants PlantClass[5][9] ;
        vector<Pea>         peas;                                   // 儲存所有
豆子的 vector
        vector<Sun>         suns;                                   // 儲存所有
太陽的 vector
        bool                selected;                               // 判斷目前
是否有選取東西
        int                 ChoosedCard;                            // 目前選取
的卡片
        int                 ChoosedPlant;                           // 目前選取
的植物
        int                 SunCounter;                             // 從空中掉
落太陽的計時器
        int                 PlantManager[5][9] = { 0 };              // 儲存場
上植物的位置
        int flow;
        int                 ZombieCounter;                          // 產生殭屍
的計時器
```

```
        int                 wave;                                       // 目前的波
數
        bool                awooga;                                   // 第一批殭屍出
現時撥放音樂的 flag
        bool                myrunning;
        bool                show_menu;
        int show_back;
        int show_con;
    };

    ////////////////////////////////////////////////////////////////////////////
////
    // 這個 class 為遊戲的結束狀態(Game Over)
    // 每個 Member function 的 Implementation 都要弄懂
    ////////////////////////////////////////////////////////////////////////////
////

    class CGameStateOver : public CGameState {
    public:
```

```
        CGameStateOver(CGame *g);



        void OnBeginState();                        // 設定每次重玩所需的變數

        void OnInit();

        // TODO:

        void LoadBitmap();

        void LoadAudio();                           // 讀取所需的音效檔

        void OnKeyDown(UINT, UINT, UINT);  // 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:

        int counter;                                // 倒數之計數器

        CMovingBitmap ZombiesWon;                   // 遊戲失敗的畫面

        CMovingBitmap ZombieLose;                   // 遊戲勝利的畫面

        CMovingBitmap FinalWin;                 // 最後遊戲勝利的畫面

    };

}
```

Pea.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Plants.h"

#include "zombies.h"

#include "Pea.h"

bool LoadPeaAlready = false;

namespace game_framework {

   CMovingBitmap   anime;

   CMovingBitmap   snow;

   CMovingBitmap   hit;

   Pea::Pea() {

   }

   Pea::Pea(int xx,int rows,int id){

       finished = false;
```

```
        x = xx;

        y = 80 + 100 * rows;

        row = rows;

        type = id;

        finished = false;

        HitZombie = false;

        if (LoadPeaAlready == false) {

            LoadBitmap();

            LoadPeaAlready = true;

        }

    }

    void Pea::LoadBitmap() {

        anime.LoadBitmap(".\\BMP_RES\\image\\plants\\PeaBullet.bmp",RGB(0,0,0
));

        snow.LoadBitmap(".\\BMP_RES\\image\\plants\\SnowPeaBullet.bmp",
RGB(0, 0, 0));

        hit.LoadBitmap(".\\BMP_RES\\image\\plants\\PeaBulletHit.bmp",RGB(0,0,
0));

    }
```

```
void Pea::OnMove() {

    x+=4;

}

void Pea::OnShow() {

    if (HitZombie == true) {

        hit.SetTopLeft(x, y);

        hit.ShowBitmap();

        finished = true;

    }

    else {

        if (type == 0) {

            anime.SetTopLeft(x, y);

            anime.ShowBitmap();

        }

        else if (type == 1) {

            snow.SetTopLeft(x, y);

            snow.ShowBitmap();

        }

    }
```

```
}

int Pea::GetRow() {

    return row;

}

int Pea::GetX() {

    return x;

}

void Pea::SetHitZombie(bool a) {

    HitZombie = a;

}

bool Pea::isHitZombie() {

    return HitZombie;

}

bool Pea::isFinished() {

    return finished;

}

int Pea::MyType() {

    return type;

}
```

```
}
```

Pea.h

```
#pragma once

namespace game_framework {

    class Pea {

    public:

        Pea();

        Pea(int,int,int);                    // 豆子的初始化

        void LoadBitmap();                  // 讀取豆子的相關圖片

        void OnShow();

        void OnMove();

        int GetRow();                          // 回傳豆子所在的排數

        int GetX();                            // 回傳豆子目前的 x 位置

        void SetHitZombie(bool);         // 設定豆子目前是否擊中殭屍

        bool isHitZombie();                 // 回傳豆子的狀態

        bool isFinished();                  // 回傳豆子的動畫是否已經結束

        int MyType();                          // 回傳豆子的型態

    private:
```

```
        int x, y;

        int row;

        int type;                                   // 豆子的型態(0 是普通豆子，1 是冷凍豆
子)

        bool finished;                          // 豆子的動畫是否已經顯示結束

        bool HitZombie;                         // 豆子是否有擊中殭屍

    };

}
```

Plants.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Plants.h"

//#include "zombies.h"

#include <sstream>
```

```
namespace game_framework {

   Plants::Plants() {

   }

   Plants::Plants(int id, int col, int roww) { //0-8 0-4

       ID = id;

       x = 172 + col* 80;

       y = 80 + roww * 98;

       velocity = 25;

       row = roww;

       column = col;

       SetLife();

       LoadBitmap();

       anime.SetDelayCount(3);

       anime2.SetDelayCount(3);

       anime3.SetDelayCount(3);

       Boom.SetDelayCount(1);

       Action = false;

       CounterOn = false;

       TargetX = 1000;
```

```
    MovingCounter = 0;

    status = 1;

    switch (ID) {

    case 1: ActionCounter = 30 * 10;  break;

    case 2: ActionCounter = 30 * 1;   break;

    case 6: ActionCounter = 30 * 1;   break;

    case 7: ActionCounter = 30 * 1;   break;

    default:ActionCounter = -1;        break;

    }

    Counter = ActionCounter - 1;

    if (ID == 1 || ID == 4) {

        Counter = 0;

    }

    anime2Counter = 0;

}

void Plants::SetID(int a) {

    ID = a;

}

int Plants::GetID() {
```

```
    return ID;

}

void Plants::ResetCounter() {

    Counter = 0;

}

void Plants::SetLife() {

    switch (ID) {

    case 1:life = 5;  break;

    case 2:life = 5;  break;

    case 3:life = 30; break;

    case 4:life = 5;  break;

    case 5:life = 5;  break;

    case 6:life = 5;  break;

    case 7:life = 5;  break;

    }

}

int Plants::GetLife() {

    return life;

}
```

```
bool Plants::isAlive() {

    if (GetLife() > 0)  return true;

    return false;

}

void Plants::BeingAttacked() {

    life--;

}

void Plants::SetFrames()

{

    switch (ID) {

    case 1: frames = 17;    break;

    case 2: frames = 12;    break;

    case 3: frames = 15;    break;

    case 31:frames = 10;    break;

    case 32:frames = 14;    break;

    case 4: frames = 16;    break;

    case 41:frames = 3;     break;

    case 5: frames = 6;     break;

    case 6: frames = 14;    break;
```

```
    case 7: frames = 14;    break;

    case 0: frames = 18;    break;

    default:    break;

    }

}

int Plants::GetRow()

{

    return row;

}

int Plants::GetColumn() {

    return column;

}

int Plants::GetX()

{

    return x;

}

int Plants::GetY() {

    return y;

}
```

```
int Plants::GetWidth()

{

    return anime.Width();

}

void Plants::SetX(int a) {

    x = a;

}

void Plants::SetY(int a) {

    y = a;

}

void Plants::LoadBitmap() {

    SetFrames();

    if (ID == 0) {

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

            char FILENAME[100];

            sprintf(FILENAME, "%s.bmp", GetPath().c_str());



            anime.AddBitmap(FILENAME, RGB(0, 0, 0));

        }
```

```
    }

    else {

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

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp", GetPath().c_str(),i);



            anime.AddBitmap(FILENAME, RGB(0, 0, 0));

        }

    }



    if (ID == 3) {

        SetID(31);

        SetFrames();

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

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp", GetPath().c_str(),i);

            anime2.AddBitmap(FILENAME, RGB(0, 0, 0));

        }

        SetID(32);
```

```
        SetFrames();

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

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp", GetPath().c_str(),i);

            anime3.AddBitmap(FILENAME, RGB(0, 0, 0));

        }

        SetID(3);

    }

    else if (ID == 4) {

        SetID(41);

        SetFrames();

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

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp", GetPath().c_str(),i);

            anime2.AddBitmap(FILENAME, RGB(0, 0, 0));

        }

        for (int i = 0; i <= 2; i++) {

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp", GetPath().c_str(), 3);
```

```
            anime2.AddBitmap(FILENAME, RGB(0, 0, 0));

        }

        SetID(4);

    }

    else if (ID == 5) {

        for (int i = 0; i <= 12; i++) {

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp",
".\\BMP_RES\\image\\plants\\CherryBomb\\Boom",i);

            Boom.AddBitmap(FILENAME, RGB(0, 0, 0));

        }

    }



}

void Plants::SetCounterOn(bool a) {

    CounterOn = a;

    if (a == false) {

        Counter = 1;

    }
```

```
}

bool Plants::isCounterOn() {

    return CounterOn;

}

int Plants::CounterLeft() {

    return ActionCounter - (Counter%ActionCounter);

}

bool Plants::isAction() {

    return Action && CounterOn;

}

void Plants::StartAction() {

    status = 2;

}

int Plants::WhichAction() {

    return status;

}

void Plants::SetTargetX(int a) {

    TargetX = a;

    velocity = (TargetX - x) / 10;
```

```
}

bool Plants::isFinished() {

    if (ID == 4 && anime2.IsFinalBitmap() == true) {

        return true;

    }

    else if (ID == 5 && anime2Counter == 5) {

        return true;

    }

    return false;

}

void Plants::OnMove() {

    if (status == 1) {

        anime.OnMove();

    }

    else if (ID == 4 && status == 2) {

        anime2.SetDelayCount(6);

        anime2.OnMove();

    }

    else if (ID == 5 && status == 2) {
```

```
        anime2Counter++;

    }

    // if (ID == 4 && anime.IsFinalBitmap() == true) {

    //  StartAction();

    // }

    if (ID == 4 && anime2.IsFinalBitmap() == true) {

        status=3;

    }

     if (ID == 4 && TargetX != 1000) {

        if (MovingCounter != 10) {

            x = x + 6;

            y -= 12;

            MovingCounter++;

        }

        else {

            //y += 160;

            StartAction();

            TargetX = 1000;

        }
```

```
 }

if (ID == 5 && anime.IsFinalBitmap() == true) {

    StartAction();

}

if (ID == 5 && Boom.IsFinalBitmap() == true) {

    status=3;

}



if (CounterOn == true) {

    Counter++;

    if (Counter % ActionCounter == 0) {

        Action = true;

    }

    else {

        if (ID == 4 && Counter > ActionCounter) {

            Action = true;

        }

        if (ID == 7 && Counter % 30 == 5) {

            Action = true;
```

```
            }

            else {

                Action = false;

            }

        }

    }

}

void Plants::OnShow() {

    if (ID == 4) {

        if (WhichAction() == 1) {

            anime.SetTopLeft(x, y);

            anime.OnMove();

            anime.OnShow();

        }

        else if (WhichAction() == 2) {

            anime2.SetTopLeft(x+20, y);

            anime2.OnMove();

            anime2.OnShow();

        }
```

```
    }

    else if (ID == 3) {

        if (GetLife() > 20) {

            anime.SetTopLeft(x, y);

            anime.OnShow();

        }

        else if (GetLife() > 10) {

            anime2.OnMove();

            anime2.SetTopLeft(x, y);

            anime2.OnShow();

        }

        else if (GetLife() > 0) {

            anime3.OnMove();

            anime3.SetTopLeft(x, y);

            anime3.OnShow();

        }

    }

    else if (ID == 5 && anime2Counter != 5 && status == 2) {

        Boom.SetTopLeft(x-60, y-80);
```

```
            Boom.OnMove();

            Boom.OnShow();



        }

        else {

            anime.SetTopLeft(x, y);

            anime.OnShow();

        }

    }

    string Plants::GetPath() {

        stringstream ss;

        switch (ID) {

        case 0: ss <<

".\\BMP_RES\\image\\plants\\void";                    break;

        case 1: ss <<

".\\BMP_RES\\image\\plants\\SunFlower\\SunFlower_";      break;

        case 2: ss <<

".\\BMP_RES\\image\\plants\\PeaShooter\\PeaShooter_";    break;
```

```
        case 3: ss <<
".\\BMP_RES\\image\\plants\\WallNut\\WallNut_";          break;
        case 31:ss <<
".\\BMP_RES\\image\\plants\\WallNut\\WallNut_cracked1_";  break;
        case 32:ss <<
".\\BMP_RES\\image\\plants\\WallNut\\WallNut_cracked2_";  break;


        case 4: ss <<
".\\BMP_RES\\image\\plants\\Squash\\Squash_";            break;
        case 41:ss <<
".\\BMP_RES\\image\\plants\\Squash\\SquashAttack_";       break;
        case 5: ss <<
".\\BMP_RES\\image\\plants\\CherryBomb\\CherryBomb_";     break;
        case 6: ss <<
".\\BMP_RES\\image\\plants\\SnowPea\\SnowPea_";                  break;
        case 7: ss <<
".\\BMP_RES\\image\\plants\\Repeater\\Repeater_";                break;
        default:ss <<
"";                                                 break;
```

```
        }

        return ss.str();

    }

}
```

Plants.h

```
#pragma once

namespace game_framework {

    class Plants {

    public:

        Plants();

        Plants(int,int,int);

        void SetID(int);

        void SetLife();

        int GetLife();

        bool isAlive();

        void LoadBitmap();

        void OnMove();

        void OnShow();
```

```
string GetPath();

void SetFrames();

int GetRow();

int GetColumn();

int GetX();

int GetY();

int GetWidth();

void SetX(int);

void SetY(int);

void BeingAttacked();

int GetID();

void ResetCounter();

void SetCounterOn(bool);

bool isCounterOn();

bool isAction();

int CounterLeft();

void StartAction();

int WhichAction();

void SetTargetX(int);
```

```
    bool isFinished();

private:

    int x, y;

    int Counter;

    int ActionCounter;

    int MovingCounter;

    int anime2Counter;

    int velocity;

    bool CounterOn;

    bool Action;

    int status;

    int TargetX;

    int row, column;

    int life;

    int ID;



    /*

        0 : empty

        1 : SunFlower
```

```
            2 : PeaShooter

            3 : WallNut

            31: WallNut_cracked1

            32: WallNut_cracked2

            4 : Squash

            41: SquashAttack

            5 : CherryBomb

            6 : SnowPea

            7 : Repeater

        */

        int frames;

        CAnimation anime;

        CAnimation anime2;

        CAnimation anime3;

        CAnimation Boom;

    };

}
```

Seed.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Seed.h"

#include "card.h"

namespace game_framework {

   Seed::Seed() {

       money =100;

   }

   void Seed::Reset() {                                  // 重設初始狀態

       money = 100;

       for (vector<Card>::iterator it = cards.begin(); it != cards.end();

it++) {

           it->Reset();

       }

   }
```

```
void Seed::ResetCD() {                              // 強制冷卻時間歸零(密技用)
    for (vector<Card>::iterator it = cards.begin(); it != cards.end();
it++) {
        it->Reset();
    }
}
int Seed::GetMoney() {                              // 回傳目前太陽數量
    return money;
}
void Seed::Buy(int a) {                             // 購買植物
    money -= cards[a].GetPrice();
}
void Seed::GotSun() {                                 // 增加太陽
    money += 25;
    if (money > 9999) {                               // 上限 9999
        money = 9999;
    }
}
void Seed::Load(int sum, int CardIDs[]) {           //讀取所需的資源(包含卡片)
```

```
        LoadBitmap();

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

            //cards.push_back(Card(CardIDs[i]));

            cards.push_back(Card(i+1));

        }

        for (vector<Card>::iterator it = cards.begin(); it != cards.end();
it++) {

            it->LoadBitmap();

        }

    }

    void Seed::LoadBitmap() {                                    //讀取所需的圖檔

        // sum money

        bmp.LoadBitmap("./BMP_RES/image/interface/SunBack.bmp", RGB(0, 0,
0));

        for (int i = 0; i < 4; i++) {

            for (int j = 0; j < 10; j++) {

                char FILENAME[100];

                sprintf(FILENAME,  ".\\BMP_RES\\image\\interface\\%d.bmp", j);

                sun[i][j].LoadBitmap(FILENAME, RGB(255, 255, 255));
```

```
            }

        }

    }

    int Seed::GetCardID(int a) {                                    // 回傳特定卡片所代
表的植物種類

        return cards[a].GetID();

    }

    void Seed::ResetCardCounter(int a) {                            // 重設特定卡片的
冷卻時間

        cards[a].ResetCounter();

    }

    bool Seed::isCardAvailible(int a) {                             // 回傳特定卡片是
否可被選取

        return cards[a].isAvailible();

    }

    void Seed::OnMove()                                             // 控制商店的動作

    {

        for (vector<Card>::iterator it = cards.begin(); it != cards.end();
it++) {
```

```
            it->OnMove();

            if (it->GetPrice() > GetMoney()) {

                it->SetAvailible(false);

            }

            else

                it->SetAvailible(true);

        }

    }

    void Seed::OnShow(int num)                                          // 顯示商
店

    {

        bmp.SetTopLeft(0, 0);

        bmp.ShowBitmap();



        for (vector<Card>::iterator it = cards.begin(); it != cards.end();
it++) {

                it->SetXY(it-cards.begin());

                it->OnShow();

                num--;
```

```
        if(num==0)
            break;


    }
    // 顯示太陽數量
    if (GetMoney() == 0) {
        sun[0][0].SetTopLeft(100, 10);
        sun[0][0].ShowBitmap();
    }
    else if (GetMoney() > 0 && GetMoney() < 100) {
        for (int i = 0, num = GetMoney(); i < 2; i++, num /= 10) {
            sun[i][num % 10].SetTopLeft(100 - 13 * i,8);
            sun[i][num % 10].ShowBitmap();
        }
    }
    else if (GetMoney() >= 100 && GetMoney() < 1000) {
        for (int i = 0, num = GetMoney(); i < 3; i++, num /= 10) {
            sun[i][num % 10].SetTopLeft(100 - 13 * i,8);
            sun[i][num % 10].ShowBitmap();
```

```
            }

        }

        else {

            for (int i = 0, num = GetMoney(); i < 4; i++, num /= 10) {

                sun[i][num % 10].SetTopLeft(100 - 13 * i,8);

                sun[i][num % 10].ShowBitmap();

            }

        }

    }

}
```

Seed.h

```
#pragma once

#include "card.h"

namespace game_framework {

    class Seed {

    public:

        Seed();                                 // 初始化

        int GetMoney();                         // 回傳目前擁有的太陽
```

```
        void OnMove();                          // 處理動作
        void OnShow(int);                       // 顯示
        void Load(int, int[]);          // 讀取所需的資源(包含卡片)
        void LoadBitmap();                  // 讀取所需的圖檔
        void Buy(int);                      // 購買植物
        void GotSun();                      // 增加太陽
        int GetCardID(int);                 // 回傳特定卡片代表的植物種類
        void ResetCardCounter(int);     // 重設特定卡片的冷卻時間
        bool isCardAvailible(int);      // 回傳特定卡片是否可被選取
        void Reset();                           // 重設為初始狀態
        void ResetCD();                     // 重設所有卡片的冷卻時間(密技用)
    private:
        unsigned int money;                 // 目前擁有的太陽
        vector<Card> cards;                 // 儲存擁有的卡片
        CMovingBitmap bmp;
        CMovingBitmap   sun[4][10];
    };
}
```

Shovel.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Shovel.h"

namespace game_framework {

    //  鏟子的初始化

    Shovel::Shovel() {

        xb = 120;

        yb = 0;

        x = 135;

        y = 10;

        Choosed = false;

    }

    //  讀取鏟子所需的圖檔

    void Shovel::LoadBitmap() {
```

```
        bmp.LoadBitmap(".\\BMP_RES\\image\\interface\\Shovel.bmp", RGB(0, 0,
255));
        back.LoadBitmap(".\\BMP_RES\\image\\interface\\ShovelBack.bmp",
RGB(0, 0, 0));
    }
    void Shovel::OnShow() {
        back.SetTopLeft(xb, yb);
        back.ShowBitmap(1.5);
        if (isChoosed() == false) {
            bmp.SetTopLeft(x, y);
            bmp.ShowBitmap();
        }
    }
    bool Shovel::isChoosed() {
        return Choosed;
    }
    void Shovel::SetChoosed(bool a) {
        Choosed = a;
    }
```

```
}
```

Shovel.h

```
#pragma once

namespace game_framework {

    class Shovel {

    public:

        Shovel();                           // 鏟子的初始化

        void OnShow();                  // 顯示在畫面上

        void LoadBitmap();            // 讀取所需的圖檔

        bool isChoosed();               // 檢查鏟子是否被選取

        void SetChoosed(bool);      // 設定鏟子是否已被選取

    private:

        int x, y;                           // 鏟子的座標

        int xb, yb;                        // 放置鏟子的框框的位置

        bool Choosed;                   // 儲存鏟子的狀態

        CMovingBitmap bmp;

        CMovingBitmap back;

    };
```

```
}
```

Sun.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Sun.h"

#include <ctime>

#include <cstdlib>

bool LoadSunAlready = false;                    // 太陽圖檔是否已經讀取

namespace game_framework {

    CAnimation  anime;

    Sun::Sun() {

    }

    Sun::Sun(int a, int b, bool c) {          // a 是 X 座標, b 是 Y 座標 ,c 代表怎麼
被產生的
```

```
        x = a;
        y = (c ? b : -100);
        desy = b;
        picked = false;
        MoveCounter = 0;
        finished = false;
        CallFromSunFlower = c;
        if (c) {                                        //如果是從向日葵產生的則以
拋物線的方式出現
            srand((unsigned)time(NULL));
            vy = -8;
            vx = (rand() % 5) - 2;
        }
        if (LoadSunAlready == false) {          //  如果已經讀取過圖檔則不用讀取
            LoadBitmap();
            LoadSunAlready = true;
        }
    }
    void Sun::LoadBitmap() {                        // 讀取圖檔
```

```
    for (int i = 0; i <= 21; i++) {
        char FILENAME[100];
        sprintf(FILENAME,"%s%d.bmp", "BMP_RES/image/Sun/Sun_",i);
        anime.AddBitmap(FILENAME, RGB(0, 0, 0));
    }
    anime.SetDelayCount(2);
}
int Sun::GetX() {                                   // 回傳 X 座標
    return x;
}
int Sun::GetY() {                                   // 回傳 Y 座標
    return y;
}
int Sun::GetHeight() {                              // 回傳圖的高度
    return anime.Height();
}
int Sun::GetWidth() {                               // 回傳圖的寬度
    return anime.Width();
}
```

```
void Sun::PickUp() {                                    // 設定太陽被撿起後的移動
    picked = true;
    stepX = (x - 25) / 20;
    stepY = (y - 25) / 20;

}
bool Sun::isFinished() {                                // 檢查移動是否結束
    return finished;
}
void Sun::drop() {                                      // 讓太陽從空中落下
    if (y <= desy) {
        y += 2;
    }
}
void Sun::fly() {                                       // 讓太陽以拋物線飛出來
    if (y <= desy) {
        y += vy;
        vy++;
```

```
        x += vx;

    }

}

void Sun::MoveAnime() {                          // 處理太陽的動畫

    anime.OnMove();

}

void Sun::OnMove() {                             // 處理太陽的移動

    if (picked == false && CallFromSunFlower == false) {

        drop();

    }

    else if (picked == false && CallFromSunFlower == true) {

        fly();

    }

    else if (picked == true) {

        x -= stepX;

        y -= stepY;

        MoveCounter++;

    }

    if (MoveCounter == 20) {
```

```
            finished = true;

        }

    }

    void Sun::OnShow() {                                //    讓太陽顯示在畫面中

        anime.SetTopLeft(x, y);

        anime.OnShow();

    }

}
```

Sun.h

```
#pragma once

namespace game_framework {

    class Sun {

    public:

        Sun();

        Sun(int, int, bool);            // 太陽的初始化

        void LoadBitmap();              // 讀取所需圖檔

        void OnMove();                  // 處理太陽的移動

        void MoveAnime();               // 處理太陽的動畫
```

```
    void OnShow();                  // 顯示在畫面中
    void drop();                    // 讓太陽從空中掉下來
    void fly();                     // 讓太陽以拋物線的方式移動
    int GetX();                     // 回傳 X 座標
    int GetY();                     // 回傳 Y 座標
    int GetHeight();                // 回傳圖的高度
    int GetWidth();                 // 回傳圖的寬度
    void PickUp();                  // 讓太陽被撿起
    bool isFinished();              // 檢查移動是否已結束
private:
    int x, y;                       // X, Y 座標
    int desy;                       // 太陽最終的 Y 座標
    int vx, vy;                     // X 方向的速度和 Y 方向的速度
    bool picked;                    // 是否已被選取
    int MoveCounter;                // 移動的計時器
    int stepX, stepY;               // 移動過程中每一次移動的距離
    bool finished;                  // 移動是否已結束
    bool CallFromSunFlower;         // 是否是從向日葵中產生
};
```

```
}
```

zombies.cpp

```
#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "zombies.h"

#include <cstring>

#include <sstream>

#include <stdio.h>

namespace game_framework {

    Zombies::Zombies() {

    }

    // 殭屍的初始化

    Zombies::Zombies(int a, int b, int c) :status(1) {

        ID = a;
```

```
x = c;

y =  35 + (b-1) * 98;

row = b-1;

LastMove = 0;

AttackCounter = 0;

AttackClock = 30;

SnowCounter = 0;

HeadFinished = false;

DieFinished = false;

BoomFinished = false;

Boom = false;

velocity = 1;

NormalWalking1.SetDelayCount(7);

NormalAttacking1.SetDelayCount(3);

NormalWalking2.SetDelayCount(7);

NormalAttacking2.SetDelayCount(3);

NormalWalking3.SetDelayCount(7);

NormalAttacking3.SetDelayCount(3);

AnimeDie.SetDelayCount(4);
```

```
    Head.SetDelayCount(4);

    BoomDie.SetDelayCount(4);

    GiveLife();

    LoadBitmap();

}

//  設定殭屍的種類

void Zombies::SetID(int a) {

    ID = a;

}

//  回傳殭屍的種類

int Zombies::GetID() {

    return ID;

}

//  設定殭屍的生命

void Zombies::GiveLife() {

    switch (ID) {

    case 1:life = 50; break;

    case 2:life = 60; break;

    case 3:life = 70; break;
```

```
    default:          break;

    }

}

//  回傳目前剩下的生命

int Zombies::GetLife() {

    return life;

}

//  回傳是否已經死亡

bool Zombies::isAlive() {

    if (GetLife() > 0)  return true;

    return false;

}

//  讓殭屍瞬間死亡

void Zombies::GoToDie() {

    life = 0;

    x=1000;

}

void Zombies::SetX(int newx) {

    x = newx;
```

```
}

//  讓殭屍被炸死

void Zombies::BoomToDie() {

    Boom = true;

    life = 0;

}

// 設定殭屍的狀態

void Zombies::SetStatus(int now) {

    status = now;

}

//  回傳目前的狀態

int Zombies::GetStatus() {

    return status;

}

//  根據殭屍的種類決定圖片的數量

// void Zombies::SetFrames()

// {

//  switch (ID) {

//  case 1: WalkingFrames1 = 17;    AttackingFrames1 = 20; break;
```

```
   //  case 2: WalkingFrames2 = 20;    AttackingFrames2 = 10; break;

   //  case 3: WalkingFrames3 = 14;    AttackingFrames3 = 10; break;

   //  default:                                           break;

   //  }

   // }

   //  讀取所需的圖檔

   void Zombies::LoadBitmap() {

       // SetStatus(1);

       // SetFrames();

       // SetStatus(2);

       for (int i = 0; i <= 17; i++) {

           char FILENAME[100];

           sprintf(FILENAME,".\\BMP_RES\\image\\zombies\\Normal
Zombie\\Zombie_%d.bmp",i);

           NormalWalking1.AddBitmap(FILENAME, RGB(0, 0, 0));

       }

       for (int i = 0; i <= 20; i++) {

           char FILENAME[100];
```

```
            sprintf(FILENAME,".\\BMP_RES\\image\\zombies\\Normal
Zombie\\ZombieAttack_%d.bmp",i);
            NormalAttacking1.AddBitmap(FILENAME, RGB(0, 0, 0));
        }

        for (int i = 0; i <= 20; i++) {
            char FILENAME[100];

            sprintf(FILENAME,".\\BMP_RES\\image\\zombies\\ConeHeadZombie\\Zom
bie_%d.bmp",i);
            NormalWalking2.AddBitmap(FILENAME, RGB(0, 0, 0));
        }
        for (int i = 0; i <= 10; i++) {
            char FILENAME[100];

            sprintf(FILENAME,".\\BMP_RES\\image\\zombies\\ConeHeadZombie\\Zom
bieAttack_%d.bmp",i);
            NormalAttacking2.AddBitmap(FILENAME, RGB(0, 0, 0));
        }
```

```
        for (int i = 0; i <= 14; i++) {
            char FILENAME[100];
            sprintf(FILENAME,".\\BMP_RES\\image\\zombies\\BucketHeadZombie\\Z
ombie_%d.bmp",i);
            NormalWalking3.AddBitmap(FILENAME, RGB(0, 0, 0));
        }
        for (int i = 0; i <= 10; i++) {
            char FILENAME[100];
            sprintf(FILENAME,".\\BMP_RES\\image\\zombies\\BucketHeadZombie\\Z
ombieAttack_%d.bmp",i);
            NormalAttacking3.AddBitmap(FILENAME, RGB(0, 0, 0));
        }

        // SetStatus(1);
        for (int i = 0; i <= 9; i++) {
            char FILENAME[100];
            sprintf(FILENAME,"%s%d.bmp",".\\BMP_RES\\image\\zombies\\Normal
Zombie\\ZombieDie_",i);
```

```
            AnimeDie.AddBitmap(FILENAME, RGB(0, 0, 0));

        }



        // 可能沒有頭繼續走

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

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp" ,".\\BMP_RES\\image\\zombies\\Normal
Zombie\\Head_",i);

            Head.AddBitmap(FILENAME, RGB(0, 0, 0));

        }



        for (int i = 0; i <= 19; i++) {

            char FILENAME[100];

            sprintf(FILENAME, "%s%d.bmp",
".\\BMP_RES\\image\\zombies\\BoomDie\\BoomDie_",i);

            BoomDie.AddBitmap(FILENAME, RGB(0, 0, 255));

        }



    }
```

```
    //  產生圖檔路徑
// string Zombies::GetPath() {
//  stringstream ss;
//  switch (ID) {
//  case 1: ss << ".\\BMP_RES\\image\\zombies\\Normal Zombie\\"; break;
//  case 2: ss << ".\\BMP_RES\\image\\zombies\\ConeHeadZombie\\"; break;
//  case 3: ss << ".\\BMP_RES\\image\\zombies\\BucketHeadZombie\\"; break;
//  default:ss << "";  break;
//  }
//  return ss.str();
// }
//  回傳殭屍所在的列數
int Zombies::GetRow()
{
    return row;
}
//  回傳 X 座標
int Zombies::GetX() {
    return x;
```

```
}

//  檢查殭屍是否要攻擊

bool Zombies::Attack() {

    if (AttackCounter == AttackClock) {

        AttackCounter = 0;

        return true;

    }

    return false;

}

//  讓殭屍加速(密技用)

void Zombies::Faster() {

    velocity+=5;

}

// 讓殭屍向前移動

void Zombies::MoveX() {

    if (SnowCounter == 0) {

         x-= velocity;

    }

    else if (SnowCounter != 0) {
```

```
        //  如果殭屍處於冷凍狀態則移動速度減半
        SnowCounter--;
        if (LastMove == 0) {
            x -= 1;
            LastMove = 1;
        }
        else if (LastMove == 1) {
            LastMove = 0;
        }
    }
}
//  處理殭屍的動作
void Zombies::OnMove() {
    if (SnowCounter == 1) {
        //AttackClock = 30;
    }
    if (isAlive() == false && Boom == false) {
        AnimeDie.OnMove();
        Head.OnMove();
```

```
    }

    else if (isAlive() == false && Boom == true) {

        BoomDie.OnMove();

    }

    else if (GetStatus() == 1) {

        NormalWalking1.OnMove();

        NormalWalking2.OnMove();

        NormalWalking3.OnMove();



        MoveX();

    }

    else if (GetStatus() == 2) {

        AttackCounter++;

        NormalAttacking1.OnMove();

        NormalAttacking2.OnMove();

        NormalAttacking3.OnMove();

    }



}
```

```
// 處理殭屍的動畫

void Zombies::OnShow() {



    if (isAlive() == false) {

        if (Boom == false) {

            if (DieFinished == false) {

                AnimeDie.SetTopLeft(x, y);

                AnimeDie.OnShow();

                if (AnimeDie.IsFinalBitmap() == true) {

                    DieFinished = true;

                }

            }



            if (HeadFinished == false) {

                Head.SetTopLeft(x, y);

                Head.OnShow();

                if (Head.IsFinalBitmap() == true) {

                    HeadFinished = true;

                }
```

```
            }



        }

        else if (Boom == true) {

            if (BoomFinished == false) {

                BoomDie.SetTopLeft(x, y);

                BoomDie.OnShow();

                if (BoomDie.IsFinalBitmap() == true) {

                    BoomFinished = true;

                }

            }

        }

    }

    else if (GetStatus() == 1) {

        if(life<=50){

            NormalWalking1.SetTopLeft(x, y);

            NormalWalking1.OnShow();

        }

        if(ID==2 && life>50){
```

```
            NormalWalking2.SetTopLeft(x, y);

            NormalWalking2.OnShow();

        }

        if(ID==3 && life>50){

            NormalWalking3.SetTopLeft(x, y);

            NormalWalking3.OnShow();

        }

    }

    else if (GetStatus() == 2) {

        if(life<=50){

            NormalAttacking1.SetTopLeft(x, y);

            NormalAttacking1.OnShow();

        }

        if(ID==2 && life>50){

            NormalAttacking2.SetTopLeft(x, y);

            NormalAttacking2.OnShow();

        }

        if(ID==3 && life>50){

            NormalAttacking3.SetTopLeft(x, y);
```

```
            NormalAttacking3.OnShow();

        }

    }

}

//  產生圖檔路徑



// string Zombies::GetPathWithStatus() {

//  if (status == 2)    return "Attack_";

//  return "_";

// }



//  設定殭屍進入冷凍狀態

void Zombies::SetSnowCounter() {

    SnowCounter = 150;

}

//  殭屍被擊中

void Zombies::Hitted(int type)

{

    life--;
```

```
        if (type == 1) {

            SetSnowCounter();

            AttackClock = 60;

            NormalAttacking1.SetDelayCount(5);

            NormalWalking1.SetDelayCount(10);

            NormalAttacking2.SetDelayCount(5);

            NormalWalking2.SetDelayCount(10);

            NormalAttacking3.SetDelayCount(5);

            NormalWalking3.SetDelayCount(10);

        }

    }

    //  檢查死亡的動畫是否已經結束

    bool Zombies::isFinished() {

        if (DieFinished == true && HeadFinished == true || BoomFinished ==
true) {

            return true;

        }

        return false;

    }
```

```
}
```

zombies.h

```
#pragma once

namespace game_framework {

    class Zombies

    {

    public:

        Zombies();

        Zombies(int, int, int);

        void LoadBitmap();

        void OnMove();

        void OnShow();

        int GetLife();

        bool isAlive();

        void GiveLife();

        string GetPath();

        string GetPathWithStatus();

        void SetStatus(int);
```

```
    int GetStatus();

    void SetFrames();

    void Hitted(int);

    int GetRow();

    int GetX();

    void SetID(int);

    void SetX(int);

    int GetID();

    bool Attack();

    bool isFinished();

    void GoToDie();

    void BoomToDie();

    void MoveX();

    void SetSnowCounter();

    void Faster();

private:

    int x, y;

    int AttackCounter;

    int AttackClock;
```

```
int SnowCounter;

int row;

unsigned int life;

int ID;

/*

    0:none

    1:zombie

    2:cone head zombie

    3:bucket head zombie

*/

int status;

int LastMove;

int velocity;

int WalkingFrames1=17;

int AttackingFrames1=20;

int WalkingFrames2=20;

int AttackingFrames2=10;

int WalkingFrames3=14;

int AttackingFrames3=10;
```

```
        CAnimation NormalWalking1;// for id 1
        CAnimation NormalAttacking1;
        CAnimation NormalWalking2;// for id 2
        CAnimation NormalAttacking2;
        CAnimation NormalWalking3;// for id 3
        CAnimation NormalAttacking3;




        CAnimation AnimeDie;
        CAnimation Head;
        CAnimation BoomDie;
        bool Boom;
        bool HeadFinished;
        bool DieFinished;
        bool BoomFinished;
    };
}
```