國立臺北科技大學

2022 Spring 資工系物件導向程式實習

期末報告

Pacman 吃豆人



第42組

107AB0008 黃詩洳

107AB0042 侯承志

**目錄**

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

「Pacman 吃豆人」是一款經典的大型電玩遊戲，從紅白機到電腦版遊戲，甚至現在也推出手機版遊戲，可以說是所有人的童年回憶。

當初會選擇這款遊戲主要是因為 Pacman 是可以透過矩陣組成的一款復古遊戲，畫面簡單明瞭，開發上相對方便，而且玩法簡單，因此我們決定實作這款經典遊戲巨作。

1. 分工

|  |  |  |
| --- | --- | --- |
| 組員 | 工作內容 | |
| 黃詩洳 | * 選擇遊戲 * 素材建立與調整 * 角色動畫製作 * 角色移動及移動範圍偵測 * 角色碰撞偵測 * 角色得分偵測 * 角色狀態切換 * 顯示地圖物件 * 地圖切換 * 分數、關卡、生命值 * 遊戲狀態切換 * 起始畫面 | * 結束畫面 * About 內容 * 組合密技 * 全螢幕 * 音效 * icon 更換 * 處理 memory leak 問題 * 整合程式碼 * 進度規劃 * 製作書面報告 * 建立 setup 檔 * Demo |
| 侯承志 | * 建立地圖 * Pacman 移動 * 音效 | |

* 1. 遊戲介紹

1. 遊戲說明

* 遊玩方式

透過上下左右方向鍵操控 Pacman 移動方向。

* 遊戲規則

操控 Pacman 吃掉藏在迷宮內所有的豆子，並且不能被鬼魂抓到，吃完所有豆子後進入下一關。Pacman 總共有五個生命值，碰到鬼混生命值減一。

* 特殊功能

迷宮的四個角落有大的閃爍點稱為大力丸，提供 Pacman 一小段時間，可以反過來吃掉鬼魂的力量，鬼魂在這段時間內會變成深藍色，會往反方向逃逸，當深藍色的鬼魂反白閃動時，表示大力丸效力即將消失。

1. 遊戲圖形

* 遊戲角色

|  |  |
| --- | --- |
| 角色 | 圖形 |
| Pacman |  |
| 正常鬼魂 |  |
| 逃避鬼魂 |  |

* 遊戲關卡

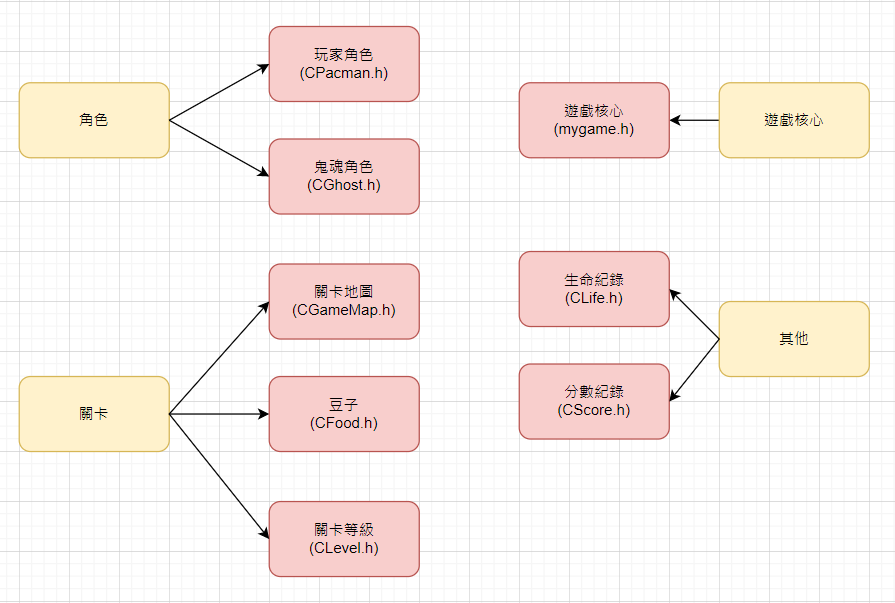
|  |  |
| --- | --- |
| 關卡 | 圖形 |
| 關卡一 |  |
| 關卡二 |  |
| 關卡三 |  |

1. 遊戲音效

|  |  |
| --- | --- |
| 事件 | 音效 |
| 關卡開始 | pacman\_beginning.wav |
| 吃豆子 | pacman\_chomp.wav |
| Pacman 死亡 | pacman\_death.wav |
| 吃大力丸 | pacman\_eatghost.wav |
| Pacman 追到逃避鬼魂 | pacman\_eatghost.wav |

* 1. 程式設計

1. 程式架構



1. 程式類別

|  |  |  |  |
| --- | --- | --- | --- |
| class名稱 | h檔行數 | cpp檔行數 | 用途 |
| mygame | 160 | 771 | * 主程式 * 含有 Init、Run、Over 三個狀態 |
| CPacman | 75 | 404 | * Pacman 角色類別 * 紀錄 Pacman 的資料 * 實作 Pacman 動作 * 實作與鬼魂的互動 |
| CGhost | 63 | 497 | * 鬼魂角色類別 * 紀錄鬼魂的資料 * 實作鬼魂的動作 |
| CFood | 30 | 89 | * 豆子角色類別 * 實作一般豆子及大力丸 * 實作與 Pacman 的互動 |
| CGameMap | 46 | 351 | * 地圖類別 * 顯示、切換地圖 * 顯示一般豆子及大力丸 |
| CLevel | 20 | 69 | * 關卡類別 * 紀錄關卡 |
| CLife | 20 | 62 | * 生命類別 * 記錄生命 |
| CScore | 20 | 65 | * 分數類別 * 紀錄分數 |

1. 程式技術

* 正常狀態鬼魂移動

根據當前座標轉換地圖矩陣位置，取得當前矩陣的上下左右四格的左上角座標，並計算距離 Pacman 左上角座標的直線距離，距離最小值且不會撞牆的方向即為鬼魂下一步的方向。

* 躲避狀態鬼魂移動

根據當前座標轉換地圖矩陣位置，取得當前矩陣的上下左右四格的左上角座標，並計算距離 Pacman 左上角座標的直線距離，距離最大值且不會撞牆的方向即為鬼魂下一步的方向。

* 眼睛狀態鬼魂移動

根據當前座標轉換地圖矩陣位置，取得當前矩陣的上下左右四格的左上角座標，並計算距離初始位置左上角座標的直線距離，距離最小值且不會撞牆的方向即為鬼魂下一步的方向。

* 1. 結語

1. 問題及解決方法

|  |  |
| --- | --- |
| 問題 | 解決方法 |
| include h 檔卻沒辦法正常使用 | 在定義 class 前先呼叫 include 的 class |
| 全螢幕後出現白邊 | 調整全螢幕黑底大小 |

1. 時間表

|  |  |  |  |
| --- | --- | --- | --- |
| 黃詩洳 | | | |
| 週數 | 時間 | 時數 | 工作細目 |
| 1 | 02/21~02/27 | 5小時 | 1. 尋找及研究適合製作的遊戲 2. git 練習及前置作業 |
| 2 | 02/28~03/06 | 3.5小時 | 1. 尋找及研究適合製作的遊戲 2. 研究及熟悉 framework |
| 3 | 03/07~03/13 | 5小時 | 1. 準備素材 2. 顯示素材 3. 顯示暫時的地圖畫面 |
| 4 | 03/14~03/20 | 4小時 | 1. 更換新的 framework 2. 製作 trello 進行進度追蹤 3. 研究角色移動 |
| 5 | 03/21~03/27 | 5小時 | 1. 建立各角色類別 2. 實作 Pacman 操作移動 3. 解決 git 的 conflict 問題 |
| 6 | 03/28~04/03 | 8小時 | 1. 實作 Pacman 撞到牆停止 2. 實作 Pacman 吃豆子 3. 實作鬼魂碰撞 Pacman，Pacman 死亡 4. 實作 Pacman 死亡動畫 5. 完成計分功能 6. git merge Map branch, and debug |
| 7 | 04/03~04/10 | 5.5小時 | 1. 調整 Pacman 移動 2. 調整鬼魂的初始位置 3. 顯示分數及關卡數字 4. 調整分數及關卡的位數 5. Demo(1) |
| 8 | 04/11~04/17 | 3小時 | 1. 新增並顯示大力丸 |
| 9 | 04/18~04/24 | 0小時 | 期中考，暫停一周 |
| 10 | 04/25~05/01 | 5小時 | 1. 當 Pacman 撞到鬼魂，Pacman 回到初始位置，遊戲重新開始 2. 顯示剩餘生命值 icon 及數值 3. 實作生命值為0時，跳到 Over state |
| 11 | 05/02~05/08 | 5小時 | 1. 調整並新增素材 2. 調整 trello 3. 當 Pacman 吃到大力丸，改變鬼魂的狀態 4. 鬼魂狀態依照持續時間切換，正常->逃避->不穩定->正常 5. 實作自動移動演算法，變成眼睛狀態的鬼魂回到基地 |
| 12 | 05/09~05/15 | 5小時 | 1. 實作自動移動演算法，變成眼睛狀態的鬼魂回到基地 2. 實作自動移動演算法，正常鬼魂追逐 Pacman 3. git merge Map branch, and debug 4. 當 Pacman 死亡或切換關卡，鬼魂回到初始位置及狀態 5. Demo(2) |
| 13 | 05/16~05/22 | 2小時 | 1. 當 Pacman 吃完迷宮所有豆子，切換關卡 2. 實作 Pacman 從左右邊界穿越 3. 調整鬼魂的移動 |
| 14 | 05/23~05/29 | 4小時 | 1. 實作自動移動演算法，逃避狀態的鬼魂往離 Pacman 較遠的方向移動 2. 實作鬼魂從左右邊界穿越 3. 調整 Pacman 移動，限制不可進入基地 4. 製作組合密技 |
| 15 | 05/30~06/05 | 1小時 | 1. 完成鬼魂離開基地的移動 2. 製作組合密技 |
| 16 | 06/06~06/12 | 2小時 | 1. 更新 trello 2. 解決 memory leak 問題 3. 調整物件碰撞範圍 4. 當 Pacman 死亡時，鬼魂會停止動作 |
| 17 | 06/13~06/19 | 24小時 | 1. 全螢幕 2. 調整組合密技按鍵 3. 在 Start 狀態中新增 start game、about 按鈕 4. 在 about 頁面新增返回、遊戲內容、操作按鍵、組合密技標籤按鈕 5. 完成遊戲內容、操作按鍵、組合密技頁面內容 6. 當最後一關 Pacman 吃完所有豆子後，顯示完成畫面 7. 當 Pacman 無生命值時，顯示 Game Over 畫面 8. 在 Over 狀態中新增並實作 restart、exit 按鈕 9. 調整開始音效 10. 新增 Pacman 追到逃避鬼的音效 11. 調整地圖素材 12. 更改 Icon 13. 修正各類程式上的 bugs 14. 解決 memory leak 問題 15. git merge 107AB0042 branch，並且整理和刪除多餘程式碼 16. 整理 RES 及所有程式碼 17. 撰寫報告書 |
| 18 | 06/20~06/26 | 0小時 | 1. Demo(3) |
| 總計 | | 87小時 | |

|  |  |  |  |
| --- | --- | --- | --- |
| 侯承志 | | | |
| 週數 | 時間 | 時數 | 工作細目 |
| 1 | 02/21~02/27 | 1小時 | 1. git 練習及前置作業 |
| 2 | 02/28~03/06 | 2小時 | 1. git 練習及前置作業 |
| 3 | 03/07~03/13 | 2小時 | 1. 製作地圖(1) |
| 4 | 03/14~03/20 | 2小時 | 1. 製作地圖(1) |
| 5 | 03/21~03/27 | 2小時 | 1. 製作地圖(1) 2. Pacman 移動邊界偵測 |
| 6 | 03/28~04/03 | 2小時 | 1. 製作地圖(1) 2. Pacman 移動邊界偵測 |
| 7 | 04/03~04/10 | 0小時 |  |
| 8 | 04/11~04/17 | 2小時 | 1. Pacman 移動邊界偵測 |
| 9 | 04/18~04/24 | 0小時 |  |
| 10 | 04/25~05/01 | 2小時 | 1. 製作地圖(2) |
| 11 | 05/02~05/08 | 2小時 | 1. 製作地圖(3) |
| 12 | 05/09~05/15 | 0小時 |  |
| 13 | 05/16~05/22 | 0小時 |  |
| 14 | 05/23~05/29 | 0小時 |  |
| 15 | 05/30~06/05 | 3小時 | 1. 加入音效 |
| 16 | 06/06~06/12 | 8小時 | 1. Pacman 轉彎修正 |
| 17 | 06/13~06/19 | 0小時 |  |
| 18 | 06/20~06/26 | 0小時 |  |
| 總計 | | 28小時 | |

1. 貢獻比例

|  |  |
| --- | --- |
| 黃詩洳 | 侯承志 |
| 76% | 24% |

1. 自我檢核表

|  |  |  |  |
| --- | --- | --- | --- |
| 項目 | 項目 | 完成否 | 無法完成原因 |
| 1 | 解決 Memory leak | □已完成 □未完成 |  |
| 2 | 自訂遊戲 Icon | □已完成 □未完成 |  |
| 3 | 全螢幕啟動 | □已完成 □未完成 |  |
| 4 | 有 About 畫面 | □已完成 □未完成 |  |
| 5 | 初始畫面說明按鍵及滑鼠之用法與密技 | □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、行距、頁碼等格式正確 | □已完成 □未完成 |  |

1. 收穫

|  |  |
| --- | --- |
| 組員 | 收穫 |
| 黃詩洳 | 這次 project 是我第一次製作 2D 遊戲，在開發的過程最有收穫的就是學會使用一個 framework，這是我第一次使用別人寫好的 framework 進行開發，學會看別人寫的 code，並且使用它，我認為是一件非常重要的事情。此外，這次的 framework 有OnInit()、OnMove()、OnShow()等方法，一開始還不熟悉的時候，容易將應該寫在OnMove()的內容寫到OnShow()，雖然不影響結果，但是會造成分工不明確的感覺，因此後期撰寫程式的時候就有特別注意這塊，這也讓我對整個遊戲的運作過程更有感覺，方法的分工對於之後的維護及撰寫真的非常地重要呢！ |
| 侯承志 | 這學期覺得最印象深刻的程式語法是因應遊戲的呈現方式，必須把要執行的物件透過framework提供的方法載入，Onshow的時間點，及其他物件互動的流程跟狀態的改變。 |

1. 心得

|  |  |
| --- | --- |
| 組員 | 心得 |
| 黃詩洳 | 讓我最有成就感的就是完成這項專案，重溫童年的回憶，過程中雖然遇到了不少阻礙，但是也讓我學會進度的安排以及時間的掌握技巧，整個開發過程中，很感謝黃漢軒同學的幫忙，當我在開發上卡關的時候都會來提點我，讓我能順利繼續開發，Pacman 是一份讓我成長許多的 project，很開心能順利完成它。 |
| 侯承志 | 在做這學期的專案時，我覺得物件的新增跟在畫面顯示不是最難的，而是物件該怎麼移動，如何判斷其中的細節是最難的，在做的過程中學到了很多。很感謝隊友幫我解決各式各樣的問題，在我時間不足的情況下幫了很多！ |

附錄

* mygame.h

#include "CGhost.h"

#include "CGameMap.h"

#include "CPacman.h"

#include "CFood.h"

#include "CScore.h"

#include "CLevel.h"

#include "CLife.h"

using namespace std;

namespace game\_framework {

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

// Constants

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

enum AUDIO\_ID

{

// 定義各種音效的編號

AUDIO\_START,// 0

AUDIO\_EAT,// 1

AUDIO\_DEADTH,// 2

AUDIO\_EATGHOST // 3

};

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

// 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 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

protected:

void OnShow();// 顯示這個狀態的遊戲畫面

void OnMove();// 顯示這個狀態的遊戲動畫

private:

CAnimation Pacman\_right;// Pacman 動畫(右)

CAnimation Pacman\_left;// Pacman 動畫(左)

CAnimation ghosts[4];// ghost 動畫

int Pacman\_x;// Pacman 位置

bool isPacmanRight;// Pacman 是否向右

CMovingBitmap logo;// csie的logo

CMovingBitmap startGame;// start game 標籤

CMovingBitmap about;// about 標籤

CMovingBitmap about\_gameContent;// 遊戲內容說明

CMovingBitmap about\_gameOperation;// 遊戲操作說明

CMovingBitmap about\_gameCombination;// 遊戲組合密技說明

CMovingBitmap back;// 回到主頁面

CMovingBitmap label\_gameContent;// 遊戲內容標籤

CMovingBitmap label\_operation;// 操作按鍵標籤

CMovingBitmap label\_combination;// 組合按鍵標籤

CMovingBitmap author;// 作者

bool isAbout;// 是否在about頁面

bool label[3];// 目前點擊標籤

double startGame\_coordinate[4];// Start Game 文字座標

double about\_coordinate[4];// About 文字座標

double back\_coordinate[4];// back 符號座標

double content\_coordinate[4];// 遊戲內容 標籤座標

double operation\_coordinate[4];// 遊戲操作 標籤座標

double combination\_coordinate[4];// 組合密技 標籤座標

};

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

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

// 每個Member function的Implementation都要弄懂

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

class CGameStateRun : public CGameState {

public:

CGameStateRun(CGame \*g);

~CGameStateRun();

void OnBeginState();// 設定每次重玩所需的變數

void OnInit();// 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

protected:

void OnMove();// 移動遊戲元素

void OnShow();// 顯示這個狀態的遊戲畫面

private:

const int NUMMAPS;// 地圖的總數

const int NUMBALLS;// 球的總數

CGameMap gameMap;// 地圖

CMovingBitmap completed;// 完成圖

CMovingBitmap gameover;// 失敗圖

CPacman c\_PacMan;// 黃色小精靈

CGhost ghost[4];// 幽靈們

CScore myScore;// 總分數

CLevel myLevel;// 關卡等級

CLife myLife;// 生命值

int runningTime;// 執行時間

int ghostDelay;// 幽靈出發的delay時間

int stateDelay;// 進入Over狀態前的Delay

int initFoods;// 原始豆子數量

int remainFoods;// 場上剩餘豆子

bool isCompleted;// 通關

bool ctrlDown;// 是否按下Ctrl鍵

void GameRestart();// 重新開始遊戲

};

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

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

void OnInit();

protected:

void OnShow();// 顯示這個狀態的遊戲畫面

private:

CMovingBitmap Restart;

CMovingBitmap Exit;

int counter;// 倒數之計數器

};

}

* mygame.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

#include "cstdlib"

namespace game\_framework {

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

// 這個class為遊戲的遊戲開頭畫面物件

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

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

void CGameStateInit::OnInit()

{

isAbout = false;

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

{

label[i] = false;

}

logo.LoadBitmap(IDB\_STARTLOGO);

startGame.LoadBitmap(IDB\_START\_STARTGAME);

about.LoadBitmap(IDB\_START\_ABOUT);

about\_gameContent.LoadBitmap(IDB\_START\_ABOUT\_GAMECONTENT);

about\_gameOperation.LoadBitmap(IDB\_START\_ABOUT\_GAMEOPERATION);

about\_gameCombination.LoadBitmap(IDB\_START\_ABOUT\_GAMECOMBINATION);

back.LoadBitmap(IDB\_START\_ABOUT\_BACK);

label\_gameContent.LoadBitmap(IDB\_LABEL\_GAMECONTENT);

label\_operation.LoadBitmap(IDB\_LABEL\_OPERATION);

label\_combination.LoadBitmap(IDB\_LABEL\_COMBINATION);

author.LoadBitmap(IDB\_GAMEAUTHOR);

// LEFT

Pacman\_left.AddBitmap(IDB\_PACMAN\_LEFT\_1, RGB(255, 255, 255));//向左\_open

Pacman\_left.AddBitmap(IDB\_PACMAN\_LEFT\_2, RGB(255, 255, 255));//向左\_close

Pacman\_x = (SIZE\_X - startGame.Width()) / 2 + startGame.Width();

Pacman\_left.SetTopLeft(Pacman\_x, SIZE\_Y \* 4 / 8);

// RIGHT

Pacman\_right.AddBitmap(IDB\_PACMAN\_RIGHT\_1, RGB(255, 255, 255));//向右\_open

Pacman\_right.AddBitmap(IDB\_PACMAN\_RIGHT\_2, RGB(255, 255, 255));//向右\_close

Pacman\_x = (SIZE\_X - startGame.Width()) / 2;

Pacman\_right.SetTopLeft(Pacman\_x, SIZE\_Y \* 4 / 8);

isPacmanRight = true;

startGame\_coordinate[0] = (SIZE\_X - startGame.Width()) / 2;// x1

startGame\_coordinate[1] = SIZE\_Y \* 6 / 8;// y1

startGame\_coordinate[2] = (SIZE\_X - startGame.Width()) / 2 + startGame.Width();// x2

startGame\_coordinate[3] = SIZE\_Y \* 6 / 8 + startGame.Height();// y2

about\_coordinate[0] = (SIZE\_X - about.Width()) / 2;// x1

about\_coordinate[1] = SIZE\_Y \* 7 / 8;// y1

about\_coordinate[2] = (SIZE\_X - about.Width()) / 2 + about.Width();// x2

about\_coordinate[3] = SIZE\_Y \* 7 / 8 + about.Height();// y2

back\_coordinate[0] = 0;// x1

back\_coordinate[1] = 0;// y1

back\_coordinate[2] = back.Width();// x2

back\_coordinate[3] = back.Height();// y2

content\_coordinate[0] = 0;// x1

content\_coordinate[1] = back\_coordinate[3];// y1

content\_coordinate[2] = label\_gameContent.Width();// x2

content\_coordinate[3] = back\_coordinate[3] + label\_gameContent.Height();// y2

operation\_coordinate[0] = 0;// x1

operation\_coordinate[1] = content\_coordinate[3];// y1

operation\_coordinate[2] = label\_operation.Width();// x2

operation\_coordinate[3] = content\_coordinate[3] + label\_operation.Height();// y2

combination\_coordinate[0] = 0;// x1

combination\_coordinate[1] = operation\_coordinate[3];// y1

combination\_coordinate[2] = label\_combination.Width();// x2

combination\_coordinate[3] = operation\_coordinate[3] + label\_combination.Height();// y2

}

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::OnLButtonDown(UINT nFlags, CPoint point)

{

double point\_x = point.x;

double point\_y = point.y;

if (!isAbout)

{

if (point\_x >= startGame\_coordinate[0] && point\_x <= startGame\_coordinate[2] && point\_y >= startGame\_coordinate[1] && point\_y <= startGame\_coordinate[3])

{

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

}

else if (point\_x >= about\_coordinate[0] && point\_x <= about\_coordinate[2] && point\_y >= about\_coordinate[1] && point\_y <= about\_coordinate[3])

{

isAbout = true;

label[0] = true;

}

}

else

{

if (point\_x >= back\_coordinate[0] && point\_x <= back\_coordinate[2] && point\_y >= back\_coordinate[1] && point\_y <= back\_coordinate[3])

{

isAbout = false;

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

{

label[i] = false;

}

}

if (point\_x >= content\_coordinate[0] && point\_x <= content\_coordinate[2] && point\_y >= content\_coordinate[1] && point\_y <= content\_coordinate[3])

{

isAbout = true;

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

{

label[i] = false;

}

label[0] = true;

}

if (point\_x >= operation\_coordinate[0] && point\_x <= operation\_coordinate[2] && point\_y >= operation\_coordinate[1] && point\_y <= operation\_coordinate[3])

{

isAbout = true;

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

{

label[i] = false;

}

label[1] = true;

}

if (point\_x >= combination\_coordinate[0] && point\_x <= combination\_coordinate[2] && point\_y >= combination\_coordinate[1] && point\_y <= combination\_coordinate[3])

{

isAbout = true;

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

{

label[i] = false;

}

label[2] = true;

}

}

}

void CGameStateInit::OnMove()

{

if (!isAbout)

{

if (isPacmanRight)

{

Pacman\_right.OnMove();

if (Pacman\_x <= (SIZE\_X - startGame.Width()) / 2 + startGame.Width())

{

Pacman\_x += 2;

Pacman\_right.SetTopLeft(Pacman\_x, SIZE\_Y \* 4 / 8);

}

else

{

isPacmanRight = false;

}

}

else

{

Pacman\_left.OnMove();

if (Pacman\_x >= (SIZE\_X - startGame.Width()) / 2)

{

Pacman\_x -= 2;

Pacman\_left.SetTopLeft(Pacman\_x, SIZE\_Y \* 4 / 8);

}

else

{

isPacmanRight = true;

}

}

}

}

void CGameStateInit::OnShow()

{

if (!isAbout)

{

// 貼上logo

logo.SetTopLeft((SIZE\_X - logo.Width())/2, SIZE\_Y \* 2 / 8);

logo.ShowBitmap();

// 顯示 Pacman

if (isPacmanRight)

{

Pacman\_right.OnShow();

Pacman\_left.SetTopLeft((SIZE\_X - startGame.Width()) / 2 + startGame.Width(), SIZE\_Y \* 4 / 8);

}

else

{

Pacman\_left.OnShow();

Pacman\_right.SetTopLeft((SIZE\_X - startGame.Width()) / 2, SIZE\_Y \* 4 / 8);

}

// 貼上文字

startGame.SetTopLeft((SIZE\_X - startGame.Width()) / 2, SIZE\_Y \* 6 / 8);

startGame.ShowBitmap();

about.SetTopLeft((SIZE\_X - about.Width()) / 2, SIZE\_Y \* 7 / 8);

about.ShowBitmap();

// 顯示作者

author.SetTopLeft((SIZE\_X - author.Width()), SIZE\_Y - author.Height());

author.ShowBitmap();

}

else

{

// 貼上文字

about.SetTopLeft((SIZE\_X - about.Width()) / 2, SIZE\_Y / 10);

about.ShowBitmap();

// 貼上返回鍵

back.SetTopLeft(0, 0);

back.ShowBitmap();

// 貼上標籤

label\_gameContent.SetTopLeft(0, back.Height());

label\_gameContent.ShowBitmap();

label\_operation.SetTopLeft(0, back.Height() + label\_gameContent.Height());

label\_operation.ShowBitmap();

label\_combination.SetTopLeft(0, back.Height() + label\_gameContent.Height() + label\_operation.Height());

label\_combination.ShowBitmap();

// 貼上說明文字

if (label[0])

{

about\_gameContent.SetTopLeft((SIZE\_X - about\_gameContent.Width()) / 2, SIZE\_Y \* 2 / 10);

about\_gameContent.ShowBitmap();

}

else if (label[1])

{

about\_gameOperation.SetTopLeft((SIZE\_X - about\_gameOperation.Width()) / 2, SIZE\_Y \* 2 / 10);

about\_gameOperation.ShowBitmap();

}

else if (label[2])

{

about\_gameCombination.SetTopLeft((SIZE\_X - about\_gameCombination.Width()) / 2, SIZE\_Y \* 2 / 10);

about\_gameCombination.ShowBitmap();

}

}

}

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

// 這個class為遊戲的結束狀態(Game Over)

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

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

void CGameStateOver::OnInit()

{

//

// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人

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

//

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

//

// 開始載入資料

//

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

//

// 最終進度為100%

//

ShowInitProgress(100);

Restart.LoadBitmap(IDB\_OVER\_RESTART);

Exit.LoadBitmap(IDB\_OVER\_EXIT);

Restart.SetTopLeft((SIZE\_X - Restart.Width()) / 2, SIZE\_Y \* 3 / 8);

Exit.SetTopLeft((SIZE\_X - Exit.Width()) / 2, SIZE\_Y \* 4 / 8);

}

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

{

double point\_x = point.x;

double point\_y = point.y;

double restart\_x1 = (SIZE\_X - Restart.Width()) / 2;

double restart\_x2 = restart\_x1 + Restart.Width();

double restart\_y1 = SIZE\_Y \* 3 / 8;

double restart\_y2 = restart\_y1 + Restart.Height();

double exit\_x1 = (SIZE\_X - Exit.Width()) / 2;

double exit\_x2 = exit\_x1 + Exit.Width();

double exit\_y1 = SIZE\_Y \* 4 / 8;

double exit\_y2 = exit\_y1 + Exit.Height();

if (point\_x >= restart\_x1 && point\_x <= restart\_x2 && point\_y >= restart\_y1 && point\_y <= restart\_y2)

{

GotoGameState(GAME\_STATE\_RUN);

}

if (point\_x >= exit\_x1 && point\_x <= exit\_x2 && point\_y >= exit\_y1 && point\_y <= exit\_y2)

{

GotoGameState(GAME\_STATE\_INIT);

}

}

void CGameStateOver::OnShow()

{

Restart.ShowBitmap();

Exit.ShowBitmap();

}

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

// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡

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

CGameStateRun::CGameStateRun(CGame \*g) : CGameState(g), NUMBALLS(28), NUMMAPS(3)

{

ghostDelay = 0;

isCompleted = false;

runningTime = 0;

}

CGameStateRun::~CGameStateRun() {}

void CGameStateRun::OnBeginState()

{

CAudio::Instance()->Play(AUDIO\_START); // 播放 START

}

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

{

runningTime++;

if (runningTime < 120) return;

//

// 判斷Pacman是否碰到food

//

vector<CFood \*>\* allFoods = gameMap.getAllFoods();

for (int i = 0; i < gameMap.getFoodCount(); i++)

{

if ((\*allFoods->at(i)).IsAlive() && (\*allFoods->at(i)).HitPacman(&c\_PacMan))

{

(\*allFoods->at(i)).SetIsAlive(false);

remainFoods--;

myScore.setScore(myScore.getScore() + (\*allFoods->at(i)).GetScore());// 得分

// 如果是碰到大魔豆

if ((\*allFoods->at(i)).GetScore() == 50)

{

// 播放吃到大魔豆的音效

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

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

// ghost轉成躲避鬼模式

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

{

if (ghost[i].GetMode() != 2) ghost[i].changeMode(1);

}

}

else

{

CAudio::Instance()->Play(AUDIO\_EAT); // 播放吃豆子的音效

}

}

}

//

// 判斷Pacman是否碰到ghost

//

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

{

// 若碰到正常鬼，Pacman死亡

if (ghost[i].GetMode() == 0)

{

if (c\_PacMan.IsAlive() && c\_PacMan.HitGhost(&ghost[i]))

{

c\_PacMan.SetIsAlive(false);// Pacman死亡

c\_PacMan.SetLastKey(0);

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

initFoods = remainFoods;

}

}

// 若碰到躲避鬼或不穩定鬼，Pacman得分

else if (ghost[i].GetMode() == 1)

{

if (c\_PacMan.IsAlive() && c\_PacMan.HitGhost(&ghost[i]))

{

// 播放吃到鬼的音效

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

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

ghost[i].changeMode(3);// 鬼變眼睛狀態

myScore.setScore(myScore.getScore() + 10);// 得分

}

}

}

if (!c\_PacMan.IsAlive())

{

if (c\_PacMan.GetDelay() == 28)

{

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

{

ghost[i].restart();// 重新開始

}

}

else {

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

{

ghost[i].changeMode(4);// 靜止

}

}

}

//

// 更新目前生命值

//

myLife.setLife(c\_PacMan.GetLife());

//

// 更新目前關卡

//

if (remainFoods == 0 && myLevel.getLevel() != 3)

{

gameMap.nextMap();

myLevel.levelUp();

myScore.setScore(myScore.getScore());

c\_PacMan.restart();

c\_PacMan.SetMap(gameMap.GetMap());

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

{

ghost[i].restart();

ghost[i].SetMap(gameMap.GetMap());

}

remainFoods = gameMap.getFoodCount();

initFoods = remainFoods;

runningTime = 0;

CAudio::Instance()->Play(AUDIO\_START); // 播放 START

return;

}

else if (remainFoods == 0 && myLevel.getLevel() == 3)

{

isCompleted = true;

}

//

// 移動PacMan

//

if (c\_PacMan.IsAlive())

c\_PacMan.OnMove();

//

// 移動ghosts

//

double eat\_rate = (initFoods - remainFoods - 0.0) / initFoods;

// 馬上出發

ghost[0].OnMove(c\_PacMan.GetX1(), c\_PacMan.GetY1());

// 約吃到10顆豆子時出發

if (eat\_rate >= 0.03) ghost[1].OnMove(c\_PacMan.GetX1(), c\_PacMan.GetY1());

else ghost[1].OnMove(-1, -1);

// 約吃到30顆豆子時出發

if (eat\_rate >= 0.1) ghost[2].OnMove(c\_PacMan.GetX1(), c\_PacMan.GetY1());

else ghost[2].OnMove(-1, -1);

// 約吃到40顆豆子時出發

if (eat\_rate >= 0.13) ghost[3].OnMove(c\_PacMan.GetX1(), c\_PacMan.GetY1());

else ghost[3].OnMove(-1, -1);

}

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

{

// 載入AUDIO

CAudio::Instance()->Load(AUDIO\_START, "sounds\\pacman\_beginning.wav"); // 載入編號0(遊戲開始)的聲音pacman\_beginning.wav

CAudio::Instance()->Load(AUDIO\_EAT, "sounds\\pacman\_chomp.wav"); // 載入編號1(吃豆子)的聲音pacman\_chomp.wav

CAudio::Instance()->Load(AUDIO\_DEADTH, "sounds\\pacman\_death.wav"); // 載入編號2(碰到鬼死亡)的聲音pacman\_death.wav

CAudio::Instance()->Load(AUDIO\_EATGHOST, "sounds\\pacman\_eatghost.wav"); // 載入編號3(碰到大豆子)的聲音pacman\_eatghost.wav

//

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

//

// 載入地圖

gameMap.SetMap(MAP\_1);

gameMap.LoadBitmap();

initFoods = gameMap.getFoodCount();

remainFoods = initFoods;

// 載入PacMan

c\_PacMan.LoadBitmap();

c\_PacMan.SetInitXY(14, 17);

c\_PacMan.SetMap(gameMap.GetMap());

c\_PacMan.SetLastKey(-1);

// 載入豆子數

remainFoods = gameMap.getFoodCount();

// 載入Score文字

myScore.LoadBitmap();

// 載入Level文字

myLevel.LoadBitmap();

// 載入Life圖示

myLife.LoadBitmap();

// 載入 red ghost

int red[4][2] = {

{ IDB\_GHOST\_RED\_DOWN\_1, IDB\_GHOST\_RED\_DOWN\_2 },

{ IDB\_GHOST\_RED\_DOWN\_1, IDB\_GHOST\_RED\_DOWN\_2 },

{ IDB\_GHOST\_RED\_LEFT\_1, IDB\_GHOST\_RED\_LEFT\_2 },

{ IDB\_GHOST\_RED\_RIGHT\_1, IDB\_GHOST\_RED\_RIGHT\_2 } };

ghost[0].LoadBitmap(red);

ghost[0].SetInitXY(MAP\_START + 12 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

ghost[0].SetInitTargetXY(MAP\_START + 14 \* BITMAP\_SIZE, MAP\_START + 11 \* BITMAP\_SIZE);

ghost[0].SetMap(gameMap.GetMap());

// 載入 blue ghost

int blue[4][2] = {

{ IDB\_GHOST\_BLUE\_DOWN\_1, IDB\_GHOST\_BLUE\_DOWN\_2 },

{ IDB\_GHOST\_BLUE\_DOWN\_1, IDB\_GHOST\_BLUE\_DOWN\_2 },

{ IDB\_GHOST\_BLUE\_LEFT\_1, IDB\_GHOST\_BLUE\_LEFT\_2 },

{ IDB\_GHOST\_BLUE\_RIGHT\_1, IDB\_GHOST\_BLUE\_RIGHT\_2 } };

ghost[1].LoadBitmap(blue);

ghost[1].SetInitXY(MAP\_START + 13 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

ghost[1].SetInitTargetXY(MAP\_START + 14 \* BITMAP\_SIZE, MAP\_START + 11 \* BITMAP\_SIZE);

ghost[1].SetMap(gameMap.GetMap());

// 載入 pink ghost

int pink[4][2] = {

{ IDB\_GHOST\_PINK\_DOWN\_1, IDB\_GHOST\_PINK\_DOWN\_2 },

{ IDB\_GHOST\_PINK\_DOWN\_1, IDB\_GHOST\_PINK\_DOWN\_2 },

{ IDB\_GHOST\_PINK\_LEFT\_1, IDB\_GHOST\_PINK\_LEFT\_2 },

{ IDB\_GHOST\_PINK\_RIGHT\_1, IDB\_GHOST\_PINK\_RIGHT\_2 } };

ghost[2].LoadBitmap(pink);

ghost[2].SetInitXY(MAP\_START + 14 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

ghost[2].SetInitTargetXY(MAP\_START + 13 \* BITMAP\_SIZE, MAP\_START + 11 \* BITMAP\_SIZE);

ghost[2].SetMap(gameMap.GetMap());

// 載入 orange ghost

int orange[4][2] = {

{ IDB\_GHOST\_ORANGE\_DOWN\_1, IDB\_GHOST\_ORANGE\_DOWN\_2 },

{ IDB\_GHOST\_ORANGE\_DOWN\_1, IDB\_GHOST\_ORANGE\_DOWN\_2 },

{ IDB\_GHOST\_ORANGE\_LEFT\_1, IDB\_GHOST\_ORANGE\_LEFT\_2 },

{ IDB\_GHOST\_ORANGE\_RIGHT\_1, IDB\_GHOST\_ORANGE\_RIGHT\_2 } };

ghost[3].LoadBitmap(orange);

ghost[3].SetInitXY(MAP\_START + 15 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

ghost[3].SetInitTargetXY(MAP\_START + 13 \* BITMAP\_SIZE, MAP\_START + 11 \* BITMAP\_SIZE);

ghost[3].SetMap(gameMap.GetMap());

// 設置位置

c\_PacMan.SetTopLeft();

myScore.SetTopLeft();

myLevel.SetTopLeft();

myLife.SetTopLeft();

ghost[0].SetTopLeft(MAP\_START + 12 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

ghost[1].SetTopLeft(MAP\_START + 13 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

ghost[2].SetTopLeft(MAP\_START + 14 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

ghost[3].SetTopLeft(MAP\_START + 15 \* BITMAP\_SIZE, MAP\_START + 14 \* BITMAP\_SIZE);

completed.LoadBitmap(IDB\_OVER\_COMPLETED);

gameover.LoadBitmap(IDB\_OVER\_GAMEOVER);

completed.SetTopLeft(MAP\_START + 12 \* BITMAP\_SIZE, MAP\_START + 17 \* BITMAP\_SIZE);

gameover.SetTopLeft(MAP\_START + 12 \* BITMAP\_SIZE, MAP\_START + 17 \* BITMAP\_SIZE);

stateDelay = 0;

}

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

{

if (runningTime < 120) return;

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

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

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

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

const char KEY\_ENTER = 0xD;// keyboard Enter

const char KEY\_CTRL = 0x11;// keyboard Ctrl

const char KEY\_E = 0x45;// keyboard E

const char KEY\_M = 0x4D;// keyboard M

if (nChar == KEY\_UP) c\_PacMan.SetLastKey(0);

if (nChar == KEY\_DOWN) c\_PacMan.SetLastKey(1);

if (nChar == KEY\_LEFT) c\_PacMan.SetLastKey(2);

if (nChar == KEY\_RIGHT) c\_PacMan.SetLastKey(3);

if (nChar == KEY\_CTRL) ctrlDown = true;

else if (nChar != KEY\_E && nChar != KEY\_M && nChar != KEY\_ENTER) ctrlDown = false;

// 切至下一關

if (nChar == KEY\_ENTER)

{

if (myLevel.getLevel() < NUMMAPS)

{

c\_PacMan.restart();

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

{

ghost[i].restart();

}

remainFoods = 0;

if (myLevel.getLevel() == 1) myScore.setScore(3130);

if (myLevel.getLevel() == 2) myScore.setScore(6320);

}

ctrlDown = false;

}

// 組合鍵 - 跳至最後關卡

if (ctrlDown && nChar == KEY\_E)

{

c\_PacMan.restart();

ghost[0].SetTopLeft(MAP\_START + 4 \* BITMAP\_SIZE, MAP\_START + 5 \* BITMAP\_SIZE);

ghost[1].SetTopLeft(MAP\_START + 4 \* BITMAP\_SIZE, MAP\_START + 20 \* BITMAP\_SIZE);

ghost[2].SetTopLeft(MAP\_START + 18 \* BITMAP\_SIZE, MAP\_START + 28 \* BITMAP\_SIZE);

ghost[3].SetTopLeft(MAP\_START + 26 \* BITMAP\_SIZE, MAP\_START + 26 \* BITMAP\_SIZE);

remainFoods = 0;

gameMap.lastMap();

myLevel.setLevel(3);

myScore.setScore(9720);

c\_PacMan.SetLife(0);

myLife.setLife(0);

c\_PacMan.SetMap(gameMap.GetMap());

c\_PacMan.SetLastKey(-1);

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

{

ghost[i].SetMap(gameMap.GetMap());

}

remainFoods = gameMap.getFoodCount();

ctrlDown = false;

}

// 組合鍵 - 轉為逃避鬼模式

if (ctrlDown && nChar == KEY\_M)

{

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

{

ghost[i].changeMode(1);

}

ctrlDown = false;

}

}

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

{

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

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

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

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

}

void CGameStateRun::GameRestart()

{

gameMap.SetMap(MAP\_1);

c\_PacMan.GameRestart();

c\_PacMan.SetMap(gameMap.GetMap());

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

{

ghost[i].GameRestart();

ghost[i].SetMap(gameMap.GetMap());

}

initFoods = gameMap.getFoodCount();

remainFoods = initFoods;

myLevel.setLevel(1);

myScore.setScore(0);

myLife.setLife(4);

ghostDelay = 0;

isCompleted = false;

ctrlDown = false;

runningTime = 0;

stateDelay = 0;

}

void CGameStateRun::OnShow()

{

// 顯示地圖

gameMap.OnShow();

// 轉換狀態

if (c\_PacMan.IsGameover()) {

gameover.ShowBitmap();

stateDelay++;

// 遊戲結束後 delay 一段時間後進入over state

if (stateDelay == 100) {

GameRestart();

GotoGameState(GAME\_STATE\_OVER);

}

}

else if (isCompleted) {

completed.ShowBitmap();

stateDelay++;

// 遊戲結束後 delay 一段時間後進入over state

if (stateDelay == 100) {

GameRestart();

GotoGameState(GAME\_STATE\_OVER);

}

}

else

{

// 顯示Pacman

c\_PacMan.OnShow();

// 顯示Ghost

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

{

ghost[i].OnShow();

}

// 顯示分數

myScore.OnShow();

// 顯示關卡等級

myLevel.OnShow();

// 顯示生命值

myLife.OnShow();

}

}

}

* CPacman.h

namespace game\_framework {

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

// CPacman: Pacman class

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

class CLife;

class CGhost;

class CPacman

{

public:

CPacman();

~CPacman();

bool HitGhost(CGhost \*ghost);// 是否碰到正常的ghost

bool IsAlive();// 是否活著

void restart();// 失去生命或切換關卡時回到起始點

int GetDelay();// 取得目前死亡的delay時間

bool IsGameover();// 遊戲是否結束 (生命值<0時)

void SetIsAlive(bool alive);// 設定是否活著

void LoadBitmap();// 載入圖形

void SetTopLeft();

void OnMove();// 移動

void OnShow();// 將圖形貼到畫面

void SetLife(int life);// 設定生命值

int GetLife();// 目前生命值

void SetInitXY(int x, int y);// 設定初始位置

int GetX1();// 取得x1(左上角)

int GetY1();// 取得y1(左上角)

int GetX2();// 取得x2(右下角)

int GetY2();// 取得y2(右下角)

void GetMapIndex();// 取得左上角及右下角在地圖的座標

void SetMovingUp(bool flag);// 設定是否正在往上移動

void SetMovingDown(bool flag);// 設定是否正在往下移動

void SetMovingLeft(bool flag);// 設定是否正在往左移動

void SetMovingRight(bool flag);// 設定是否正在往右移動

void SetMap(int \*\*map);// 初始化地圖

int FindMapIndex\_X(int x);// 取得座標在矩陣的位置(X軸)

int FindMapIndex\_Y(int y);// 取得座標在矩陣的位置(Y軸)

void SetLastKey(int key);// 設定最後一個按下的KEY

void GameRestart();// 遊戲重新開始

private:

CAnimation animation\_stop\_1;// Pacman的動畫\_靜止狀態(預設向上)

CAnimation animation\_stop\_2;// Pacman的動畫\_靜止狀態(預設向下)

CAnimation animation\_stop\_3;// Pacman的動畫\_靜止狀態(預設向左)

CAnimation animation\_stop\_4;// Pacman的動畫\_靜止狀態(預設向右)

CAnimation animation\_1;// Pacman的動畫\_向上

CAnimation animation\_2;// Pacman的動畫\_向下

CAnimation animation\_3;// Pacman的動畫\_向左

CAnimation animation\_4;// Pacman的動畫\_向右

CAnimation animation\_die;// Pacman的動畫\_死亡

CAnimation \*animation = &animation\_stop\_4;// 當前Pacman動畫方向(預設向右)

int delay = 0;// 等待狀態結束

int myLife = 4;

int init\_X, init\_Y;// Pacman初始位置

int x, y;// Pacman目前位置

int move;// 移動方向

bool isMovingDown;// 是否正在往下移動

bool isMovingLeft;// 是否正在往左移動

bool isMovingRight;// 是否正在往右移動

bool isMovingUp;// 是否正在往上移動

bool isStop = false;// 是否撞牆停止移動

bool HitRectangle(int tx1, int ty1, int tx2, int ty2);// 檢查是否有交集

bool is\_alive;// 生命狀態

bool is\_gameover = false;// 遊戲結束

int map[31][28];// 遊戲地圖

int MapIndex\_X1;// Pacman左上角位於地圖的座標(X軸)

int MapIndex\_Y1;// Pacman左上角位於地圖的座標(Y軸)

int MapIndex\_X2;// Pacman右下角位於地圖的座標(X軸)

int MapIndex\_Y2;// Pacman右下角位於地圖的座標(Y軸)

int wall\_pixel;// 牆壁位置

int lastKey;// 最近一次按的方向

};

}

* CPacman.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CGhost.h"

#include "CLife.h"

#include "CPacman.h"

namespace game\_framework

{

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

// CPacman: Pacman class

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

// Pacman

CPacman::CPacman()

{

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

is\_alive = true;

}

CPacman::~CPacman() {}

void CPacman::GameRestart()

{

restart();

myLife = 4;

delay = 0;

is\_gameover = false;

}

void CPacman::SetInitXY(int x, int y)

{

init\_X = MAP\_START + x \* BITMAP\_SIZE;// 設定Pacman起始的Y座標

init\_Y = MAP\_START + y \* BITMAP\_SIZE;// 設定Pacman起始的X座標

this->x = init\_X;

this->y = init\_Y;

}

void CPacman::LoadBitmap()

{

// STOP

animation\_stop\_1.AddBitmap(IDB\_PACMAN\_UP\_1, RGB(255, 255, 255));//靜止狀態(預設向上)

animation\_stop\_2.AddBitmap(IDB\_PACMAN\_DOWN\_1, RGB(255, 255, 255));//靜止狀態(預設向下)

animation\_stop\_3.AddBitmap(IDB\_PACMAN\_LEFT\_1, RGB(255, 255, 255));//靜止狀態(預設向左)

animation\_stop\_4.AddBitmap(IDB\_PACMAN\_RIGHT\_1, RGB(255, 255, 255));//靜止狀態(預設向右)

// DIE

animation\_die.AddBitmap(IDB\_PACMAN\_UP\_1, RGB(255, 255, 255));//死亡(向上)

animation\_die.AddBitmap(IDB\_PACMAN\_DIE\_1, RGB(255, 255, 255));

animation\_die.AddBitmap(IDB\_PACMAN\_DIE\_2, RGB(255, 255, 255));

animation\_die.AddBitmap(IDB\_PACMAN\_DIE\_3, RGB(255, 255, 255));

animation\_die.AddBitmap(IDB\_PACMAN\_DIE\_4, RGB(255, 255, 255));

animation\_die.AddBitmap(IDB\_PACMAN\_DIE\_BLACK, RGB(255, 255, 255));//全黑結束

// UP

animation\_1.AddBitmap(IDB\_PACMAN\_UP\_1, RGB(255, 255, 255));//向上\_open

animation\_1.AddBitmap(IDB\_PACMAN\_UP\_2, RGB(255, 255, 255));//向上\_close

// DOWN

animation\_2.AddBitmap(IDB\_PACMAN\_DOWN\_1, RGB(255, 255, 255));//向下\_open

animation\_2.AddBitmap(IDB\_PACMAN\_DOWN\_2, RGB(255, 255, 255));//向下\_close

// LEFT

animation\_3.AddBitmap(IDB\_PACMAN\_LEFT\_1, RGB(255, 255, 255));//向左\_open

animation\_3.AddBitmap(IDB\_PACMAN\_LEFT\_2, RGB(255, 255, 255));//向左\_close

// RIGHT

animation\_4.AddBitmap(IDB\_PACMAN\_RIGHT\_1, RGB(255, 255, 255));//向右\_open

animation\_4.AddBitmap(IDB\_PACMAN\_RIGHT\_2, RGB(255, 255, 255));//向右\_close

}

void CPacman::OnMove()

{

if (!is\_alive) return;

const int STEP\_SIZE = 2;

animation->OnMove();

isStop = false;

GetMapIndex();

if (lastKey == 0) SetMovingUp(true);

else if (lastKey == 1) SetMovingDown(true);

else if (lastKey == 2) SetMovingLeft(true);

else if (lastKey == 3) SetMovingRight(true);

if (isMovingUp)

{

if (y - wall\_pixel > STEP\_SIZE) y -= STEP\_SIZE;

else

{

y = wall\_pixel;

isStop = true;

}

}

else if (isMovingDown)

{

if (wall\_pixel - y > STEP\_SIZE) y += STEP\_SIZE;

else

{

y = wall\_pixel;

isStop = true;

}

}

else if (isMovingLeft)

{

if (x < MAP\_START + BITMAP\_SIZE \* (0) + 5)

{

x = MAP\_START + BITMAP\_SIZE \* (27);

SetMovingLeft(true);

return;

}

if (x - wall\_pixel > STEP\_SIZE) x -= STEP\_SIZE;

else

{

x = wall\_pixel;

isStop = true;

}

}

else if (isMovingRight)

{

if (x > MAP\_START + BITMAP\_SIZE \* (27) - 5)

{

x = MAP\_START + BITMAP\_SIZE \* (0);

SetMovingRight(true);

return;

}

if (wall\_pixel - x > STEP\_SIZE) x += STEP\_SIZE;

else

{

x = wall\_pixel;

isStop = true;

}

}

}

// 找座標在map陣列上的位置(X)

int CPacman::FindMapIndex\_X(int x)

{

return (x - MAP\_START) / BITMAP\_SIZE;

}

// 找座標在map陣列上的位置(Y)

int CPacman::FindMapIndex\_Y(int y)

{

return (y - MAP\_START) / BITMAP\_SIZE;

}

// 找Pacman四個點的陣列位置

void CPacman::GetMapIndex()

{

MapIndex\_X1 = FindMapIndex\_X(x);// 左

MapIndex\_X2 = FindMapIndex\_X(x + 22);// 右

MapIndex\_Y1 = FindMapIndex\_Y(y);// 上

MapIndex\_Y2 = FindMapIndex\_Y(y + 22);// 下

}

// 向上移動

void CPacman::SetMovingUp(bool flag)

{

GetMapIndex();

if (map[MapIndex\_Y1 - 1][MapIndex\_X1] != 1 && map[MapIndex\_Y1 - 1][MapIndex\_X2] != 1)

{

animation = &animation\_1;

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

isMovingUp = flag;

// 計算向上終點牆的位置

wall\_pixel = 0;

while (true)

{

if (map[MapIndex\_Y1][MapIndex\_X1] == 1)

{

wall\_pixel = MAP\_START + BITMAP\_SIZE \* (MapIndex\_Y1 + 1);

break;

}

else

MapIndex\_Y1--;

}

}

}

// 向下移動

void CPacman::SetMovingDown(bool flag)

{

GetMapIndex();

if (map[MapIndex\_Y2 + 1][MapIndex\_X1] != 1 && map[MapIndex\_Y2 + 1][MapIndex\_X2] != 1 && map[MapIndex\_Y2 + 1][MapIndex\_X1] != 2 && map[MapIndex\_Y2 + 1][MapIndex\_X2] != 2)

{

animation = &animation\_2;

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

isMovingDown = flag;

// 計算向下終點牆的位置

wall\_pixel = 0;

while (true)

{

if (map[MapIndex\_Y1][MapIndex\_X1] == 1)

{

wall\_pixel = MAP\_START + BITMAP\_SIZE \* (MapIndex\_Y1 - 1);

break;

}

else

MapIndex\_Y1++;

}

}

}

// 向左移動

void CPacman::SetMovingLeft(bool flag)

{

GetMapIndex();

if (map[MapIndex\_Y1][MapIndex\_X1-1] != 1 && map[MapIndex\_Y2][MapIndex\_X1-1] != 1)

{

animation = &animation\_3;

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

isMovingLeft = flag;

// 計算向左終點牆的位置

wall\_pixel = 0;

while (true)

{

if (map[MapIndex\_Y1][MapIndex\_X1] == 1)

{

wall\_pixel = MAP\_START + BITMAP\_SIZE \* (MapIndex\_X1 + 1);

break;

}

else

MapIndex\_X1--;

}

}

}

// 向右移動

void CPacman::SetMovingRight(bool flag)

{

GetMapIndex();

if (map[MapIndex\_Y1][MapIndex\_X2 + 1] != 1 && map[MapIndex\_Y2][MapIndex\_X2 + 1] != 1)

{

animation = &animation\_4;

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

isMovingRight = flag;

// 計算向右終點牆的位置

wall\_pixel = 0;

while (true)

{

if (map[MapIndex\_Y1][MapIndex\_X1] == 1)

{

wall\_pixel = MAP\_START + BITMAP\_SIZE \* (MapIndex\_X1 - 1);

break;

}

else

MapIndex\_X1++;

}

}

}

// 檢查Pacman所構成的矩形是否碰到ghost

bool CPacman::HitGhost(CGhost \*ghost)

{

int adjust = 2;

return HitRectangle(ghost->GetX1() + adjust, ghost->GetY1() + adjust,

ghost->GetX2() - adjust, ghost->GetY2() - adjust);

}

bool CPacman::HitRectangle(int tx1, int ty1, int tx2, int ty2)

{

int adjust = 2;

// 檢查是否有交集

return (tx2 >= x + adjust && tx1 <= (x + animation->Width() - adjust) && ty2 >= y + adjust && ty1 <= (y + animation->Height() - adjust));

}

void CPacman::SetMap(int \*\*m)

{

for (int i = 0;i < 31;i++)

{

for (int j = 0;j < 28;j++)

{

map[i][j] = m[i][j];

}

}

}

void CPacman::SetIsAlive(bool alive)

{

is\_alive = alive;

}

bool CPacman::IsAlive()

{

return is\_alive;

}

int CPacman::GetLife()

{

return myLife;

}

int CPacman::GetX1()

{

return x;

}

int CPacman::GetY1()

{

return y;

}

int CPacman::GetX2()

{

return x + animation->Width();

}

int CPacman::GetY2()

{

return y + animation->Height();

}

void CPacman::restart()

{

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

SetLastKey(-1);

animation = &animation\_stop\_4;

y = init\_Y;

x = init\_X;

is\_alive = true;

}

int CPacman::GetDelay()

{

return delay;

}

void CPacman::SetLife(int life)

{

myLife = life;

}

bool CPacman::IsGameover()

{

if (myLife < 0) return true;

return false;

}

void CPacman::SetTopLeft()

{

animation->SetTopLeft(x, y);

}

void CPacman::OnShow()

{

if (is\_alive)

{

if (isStop)

{

if (isMovingUp)

{

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

animation = &animation\_stop\_1;// 向上停止

}

else if (isMovingDown)

{

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

animation = &animation\_stop\_2;// 向下停止

}

else if (isMovingLeft)

{

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

animation = &animation\_stop\_3;// 向左停止

}

else if (isMovingRight)

{

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

animation = &animation\_stop\_4;// 向右停止

}

}

animation->SetTopLeft(x, y);

animation->OnShow();

}

else {

animation = &animation\_die;

delay++;// 死亡動畫時間

animation->SetTopLeft(x, y);

animation->OnMove();

animation->OnShow();

}

// 死亡動畫結束

if (delay == 29)

{

myLife--;

if (myLife > -1) restart();

delay = 0;

animation->OnMove();

animation->OnShow();

}

}

void CPacman::SetLastKey(int key)

{

//

// 0: 上, 1: 下, 2: 左,3: 右

//

lastKey = key;

}

}

* CGhost.h

namespace game\_framework {

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

// CGolst: Golst class

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

class CGhost

{

public:

CGhost();

~CGhost();

void LoadBitmap(int IDB[4][2]);// 載入圖形

void changeMode(int mode);// 切換模式(正常鬼/躲避鬼)

int GetMode();// 回傳精靈的狀態

int GetX1();// 取得x1(左上角)

int GetY1();// 取得y1(左上角)

int GetX2();// 取得x2(右下角)

int GetY2();// 取得y2(右下角)

void SetInitXY(int x, int y);// 設定初始位置(基地)

void SetInitTargetXY(int x, int y);// 設定初始目標位置

void SetTopLeft(int x, int y);

void GoToDestination(int des\_x, int des\_y);// 移動到指定目的地的路徑

double GetDistance(int x1, int x2, int y1, int y2);// 取得下一步到目的地的直線距離

void SetMap(int \*\*map);// 初始化地圖

int FindMapIndex\_X(int x);// 取得座標在矩陣的位置(X軸)

int FindMapIndex\_Y(int y);// 取得座標在矩陣的位置(Y軸)

void GetMapIndex();// 取得左上角及右下角在地圖的座標

void restart();// Pacman死亡或切換關卡時，回到基地

void GameRestart();// 遊戲重新開始

void OnMove(int pacman\_x, int pacman\_y);// 移動

void OnShow();// 將圖形貼到畫面

private:

CAnimation animation\_1;// 精靈動畫\_上

CAnimation animation\_2;// 精靈動畫\_下

CAnimation animation\_3;// 精靈動畫\_左

CAnimation animation\_4;// 精靈動畫\_右

CAnimation animation\_avoid;// 躲避精靈動畫\_深藍色

CAnimation animation\_change;// 不穩定精靈動畫\_深藍+白色交錯

CAnimation animation\_eyes;// 死亡精靈

CAnimation \*animation;// 目前方向動畫

int init\_x, init\_y;// 初始位置(基地)

int x, y;// 目前位置

bool isGoOut = false;// 鬼是否離開基地

int init\_target\_x, init\_target\_y;// 初始目標位置

int avoidTime = 0;// 躲避狀態持續時間

int direction = 0;// 行走方向

bool stop;// Pacman死亡時暫停動作

double turnUp(int des\_x, int des\_y);// 判斷是否可向上走，並回傳下一步距離目的地的距離

double turnDown(int des\_x, int des\_y);// 判斷是否可向下走，並回傳下一步距離目的地的距離

double turnLeft(int des\_x, int des\_y);// 判斷是否可向左走，並回傳下一步距離目的地的距離

double turnRight(int des\_x, int des\_y);// 判斷是否可向右走，並回傳下一步距離目的地的距離

int map[31][28];// 遊戲地圖

int MapIndex\_X1;// Pacman左上角位於地圖的座標(X軸)

int MapIndex\_Y1;// Pacman左上角位於地圖的座標(Y軸)

int MapIndex\_X2;// Pacman右下角位於地圖的座標(X軸)

int MapIndex\_Y2;// Pacman右下角位於地圖的座標(Y軸)

};

}

* CGhost.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include <cmath>

#include "audio.h"

#include "gamelib.h"

#include "CGhost.h"

namespace game\_framework {

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

// CGolst: Golst class

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

CGhost::CGhost() {

animation = &animation\_4;

stop = false;

}

CGhost::~CGhost() {}

void CGhost::GameRestart()

{

restart();

}

void CGhost::SetInitXY(int x, int y) {

init\_x = x;

init\_y = y;

this->x = init\_x;

this->y = init\_y;

}

void CGhost::SetInitTargetXY(int x, int y) {

init\_target\_x = x;

init\_target\_y = y;

}

void CGhost::SetMap(int \*\*m) {

for (int i = 0;i < 31;i++)

{

for (int j = 0;j < 28;j++) {

map[i][j] = m[i][j];

}

}

}

void CGhost::OnMove(int des\_x, int des\_y) {

animation->OnMove();

if (animation == &animation\_eyes) {

if (x == init\_x && y == init\_y) {

animation = &animation\_1;

isGoOut = false;

return;

}

GoToDestination(init\_x, init\_y);

}

else if (animation == &animation\_avoid || animation == &animation\_change) {

GoToDestination(des\_x, des\_y);

}

else {

if (des\_x == -1 && des\_y == -1) {

return;

}

else if (isGoOut) {

if (!stop) {

GoToDestination(des\_x, des\_y);

}

}

else {

if (x == init\_target\_x && y == init\_target\_y) {

isGoOut = true;

return;

}

GoToDestination(init\_target\_x, init\_target\_y);

}

}

if (x < MAP\_START + BITMAP\_SIZE \* (0) + 5 && direction == 3) {

x = MAP\_START + BITMAP\_SIZE \* (27);

GetMapIndex();

}

if (x > MAP\_START + BITMAP\_SIZE \* (27) - 5 && direction == 4) {

x = MAP\_START + BITMAP\_SIZE \* (0);

GetMapIndex();

}

}

void CGhost::GoToDestination(int des\_x, int des\_y) {

GetMapIndex();// 取得四個點的陣列位置

double distance;// 下一步到目的地的距離

int direction\_min;// 下一步方向 (正常用)

int direction\_max;// 下一步方向 (躲避用)

double minDistance = INT\_MAX;// 在所有方向到目的地的距離中取最短距離 (正常用)

double maxDistance = INT\_MIN;// 在所有方向到目的地的距離中取最長距離 (躲避用)

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// direction 代號表示：

// 0 => 無方向

// 1 => 向上

// 2 => 向下

// 3 => 向左

// 4 => 向右

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// 若前一步為向上或向下，則先判斷上下方向

if (direction == 1 || direction == 2) {

// 向上

distance = turnUp(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 1;// 1 為向上

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 1;// 1 為向上

}

// 向下

distance = turnDown(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 2;// 2 為向下

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 2;// 2 為向下

}

// 向左

distance = turnLeft(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 3;// 3 為向左

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 3;// 3 為向左

}

// 向右

distance = turnRight(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 4;// 4 為向右

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 4;// 4 為向右

}

}

// 若前一步為向左或向右，則先判斷左右方向

else {

// 向左

distance = turnLeft(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 3;// 3 為向左

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 3;// 3 為向左

}

// 向右

distance = turnRight(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 4;// 4 為向右

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 4;// 4 為向右

}

// 向上

distance = turnUp(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 1;// 1 為向上

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 1;// 1 為向上

}

// 向下

distance = turnDown(des\_x, des\_y);

if (distance < minDistance) {

minDistance = distance;

direction\_min = 2;// 2 為向下

}

if (distance > maxDistance && distance != INT\_MAX) {

maxDistance = distance;

direction\_max = 2;// 2 為向下

}

}

if (animation == &animation\_avoid || animation == &animation\_change) {

direction = direction\_max;

}

else {

direction = direction\_min;

}

int STEP\_SIZE;

if (animation == &animation\_avoid || animation == &animation\_change) {

STEP\_SIZE = 1;

}

else {

STEP\_SIZE = 2;

}

switch (direction) {

case 0:

break;

case 1:

y -= STEP\_SIZE;

break;

case 2:

y += STEP\_SIZE;

break;

case 3:

x -= STEP\_SIZE;

break;

case 4:

x += STEP\_SIZE;

break;

}

}

double CGhost::turnUp(int des\_x, int des\_y) {

int next\_x, next\_y;// 下一步的座標位置

double distance = INT\_MAX;

bool canTurn = false;

if (animation == &animation\_eyes || !isGoOut) {

// 狀態為眼睛時，可進入基地

// 或尚未從基地出去的鬼才可待在基地

if (map[MapIndex\_Y2 - 1][MapIndex\_X1] != 1 && map[MapIndex\_Y2 - 1][MapIndex\_X2] != 1) {

canTurn = true;

}

}

else {

// 狀態不為眼睛時，不可進入基地

if (map[MapIndex\_Y2 - 1][MapIndex\_X1] != 1 && map[MapIndex\_Y2 - 1][MapIndex\_X2] != 1 && map[MapIndex\_Y2 - 1][MapIndex\_X1] != 2 && map[MapIndex\_Y2 - 1][MapIndex\_X2] != 2) {

canTurn = true;

}

}

// 向上可以走

if (canTurn) {

if (direction != 2) {

next\_x = MAP\_START + BITMAP\_SIZE \* (MapIndex\_X1);

next\_y = MAP\_START + BITMAP\_SIZE \* (MapIndex\_Y1 - 1);

distance = GetDistance(des\_x, next\_x, des\_y, next\_y);

}

}

return distance;

}

double CGhost::turnDown(int des\_x, int des\_y) {

int next\_x, next\_y;// 下一步的座標位置

double distance = INT\_MAX;

bool canTurn = false;

if (animation == &animation\_eyes || !isGoOut) {

// 狀態為眼睛時，可進入基地

// 或尚未從基地出去的鬼才可待在基地

if (map[MapIndex\_Y1 + 1][MapIndex\_X1] != 1 && map[MapIndex\_Y1 + 1][MapIndex\_X2] != 1) {

canTurn = true;

}

}

else {

// 狀態不為眼睛時，不可進入基地

if (map[MapIndex\_Y1 + 1][MapIndex\_X1] != 1 && map[MapIndex\_Y1 + 1][MapIndex\_X2] != 1 && map[MapIndex\_Y1 + 1][MapIndex\_X1] != 2 && map[MapIndex\_Y1 + 1][MapIndex\_X2] != 2) {

canTurn = true;

}

}

// 向下可以走

if (canTurn) {

if (direction != 1) {

next\_x = MAP\_START + BITMAP\_SIZE \* (MapIndex\_X1);

next\_y = MAP\_START + BITMAP\_SIZE \* (MapIndex\_Y1 + 1);

distance = GetDistance(des\_x, next\_x, des\_y, next\_y);

}

}

return distance;

}

double CGhost::turnLeft(int des\_x, int des\_y) {

int next\_x, next\_y;// 下一步的座標位置

double distance = INT\_MAX;

bool canTurn = false;

if (animation == &animation\_eyes || !isGoOut) {

// 狀態為眼睛時，可進入基地

// 或尚未從基地出去的鬼才可待在基地

if (map[MapIndex\_Y1][MapIndex\_X2 - 1] != 1 && map[MapIndex\_Y2][MapIndex\_X2 - 1] != 1) {

canTurn = true;

}

}

else {

// 狀態不為眼睛時，不可進入基地

if (map[MapIndex\_Y1][MapIndex\_X2 - 1] != 1 && map[MapIndex\_Y2][MapIndex\_X2 - 1] != 1 && map[MapIndex\_Y1][MapIndex\_X2 - 1] != 2 && map[MapIndex\_Y2][MapIndex\_X2 - 1] != 2) {

canTurn = true;

}

}

// 向左可以走

if (canTurn) {

if (direction != 4) {

next\_x = MAP\_START + BITMAP\_SIZE \* (MapIndex\_X1 - 1);

next\_y = MAP\_START + BITMAP\_SIZE \* (MapIndex\_Y1);

distance = GetDistance(des\_x, next\_x, des\_y, next\_y);

}

}

return distance;

}

double CGhost::turnRight(int des\_x, int des\_y) {

int next\_x, next\_y;// 下一步的座標位置

double distance = INT\_MAX;

bool canTurn = false;

if (animation == &animation\_eyes || !isGoOut) {

// 狀態為眼睛時，可進入基地

// 或尚未從基地出去的鬼才可待在基地

if (map[MapIndex\_Y1][MapIndex\_X1 + 1] != 1 && map[MapIndex\_Y2][MapIndex\_X1 + 1] != 1) {

canTurn = true;

}

}

else {

// 狀態不為眼睛時，不可進入基地

if (map[MapIndex\_Y1][MapIndex\_X1 + 1] != 1 && map[MapIndex\_Y2][MapIndex\_X1 + 1] != 1 && map[MapIndex\_Y1][MapIndex\_X1 + 1] != 2 && map[MapIndex\_Y2][MapIndex\_X1 + 1] != 2) {

canTurn = true;

}

}

// 向右可以走

if (canTurn) {

if (direction != 3) {

next\_x = MAP\_START + BITMAP\_SIZE \* (MapIndex\_X1 + 1);

next\_y = MAP\_START + BITMAP\_SIZE \* (MapIndex\_Y1);

distance = GetDistance(des\_x, next\_x, des\_y, next\_y);

}

}

return distance;

}

double CGhost::GetDistance(int x1, int x2, int y1, int y2) {

int distance\_x = x1 - x2;

int distance\_y = y1 - y2;

return sqrt(distance\_x \* distance\_x + distance\_y \* distance\_y);

}

// 找座標在map陣列上的位置(X)

int CGhost::FindMapIndex\_X(int x)

{

return (x - MAP\_START) / BITMAP\_SIZE;

}

// 找座標在map陣列上的位置(Y)

int CGhost::FindMapIndex\_Y(int y)

{

return (y - MAP\_START) / BITMAP\_SIZE;

}

// 找Ghost四個點的陣列位置

void CGhost::GetMapIndex() {

MapIndex\_X1 = FindMapIndex\_X(x);// 左

MapIndex\_X2 = FindMapIndex\_X(x + 22);// 右

MapIndex\_Y1 = FindMapIndex\_Y(y);// 上

MapIndex\_Y2 = FindMapIndex\_Y(y + 22);// 下

}

void CGhost::LoadBitmap(int IDB[4][2]) {

// up

animation\_1.AddBitmap(IDB[0][0], RGB(0, 0, 0));

animation\_1.AddBitmap(IDB[0][1], RGB(0, 0, 0));

// down

animation\_2.AddBitmap(IDB[1][0], RGB(0, 0, 0));

animation\_2.AddBitmap(IDB[1][1], RGB(0, 0, 0));

// left

animation\_3.AddBitmap(IDB[2][0], RGB(0, 0, 0));

animation\_3.AddBitmap(IDB[2][1], RGB(0, 0, 0));

// right

animation\_4.AddBitmap(IDB[3][0], RGB(0, 0, 0));

animation\_4.AddBitmap(IDB[3][1], RGB(0, 0, 0));

// 躲避鬼

animation\_avoid.AddBitmap(IDB\_GHOST\_BLUE\_1, RGB(0, 0, 0));

animation\_avoid.AddBitmap(IDB\_GHOST\_BLUE\_2, RGB(0, 0, 0));

// 不穩定鬼 (躲避鬼變正常鬼間的不穩定狀態)

animation\_change.AddBitmap(IDB\_GHOST\_BLUE\_1, RGB(0, 0, 0));

animation\_change.AddBitmap(IDB\_GHOST\_WHITE\_2, RGB(0, 0, 0));

// 眼睛

animation\_eyes.AddBitmap(IDB\_GHOST\_EYES\_DOWN, RGB(0, 0, 0));

}

//

// 切換鬼的狀態

// 0 => 眼睛 to 正常鬼

// 1 => 正常鬼 to 躲避鬼

// 2 => 躲避鬼 to 不穩定鬼

// 3 => 不穩定鬼 to 眼睛

//

void CGhost::changeMode(int mode) {

if (mode == 0) animation = &animation\_4;// 正常鬼

else if (mode == 1) animation = &animation\_avoid;// 躲避鬼

else if (mode == 2) animation = &animation\_change;// 不穩定鬼

else if (mode == 3) animation = &animation\_eyes;// 眼睛

else if (mode == 4) stop = true;// 靜止狀態

direction = 0;

}

int CGhost::GetMode() {

//

// 0: 正常, 1: 躲避, 2: 眼睛

//

if (animation == &animation\_avoid || animation == &animation\_change) {

return 1;

}

else if (animation == &animation\_eyes) { // 眼睛視為正常狀態

return 2;

}

return 0;

}

int CGhost::GetX1() {

return x;

}

int CGhost::GetY1() {

return y;

}

int CGhost::GetX2() {

return x + animation->Width();

}

int CGhost::GetY2() {

return y + animation->Height();

}

void CGhost::SetTopLeft(int x, int y) {

animation->SetTopLeft(x, y);

this->x = x;

this->y = y;

}

void CGhost::restart()

{

animation = &animation\_1;

y = init\_y;

x = init\_x;

avoidTime = 0;

direction = 0;

isGoOut = false;

stop = false;

}

void CGhost::OnShow() {

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// \* 當狀態為躲避或不穩定時

// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

if (animation == &animation\_avoid || animation == &animation\_change) {

avoidTime++;

}

// 躲避約8秒後，躲避鬼轉為不穩定狀態

if (avoidTime == 180) {

changeMode(2);

}

// 再經過約3秒，從不穩定狀態回到正常狀態

if (avoidTime == 230) {

avoidTime = 0;

changeMode(0);

}

animation->SetTopLeft(x,y);

animation->OnShow();

}

}

* CFood.h

namespace game\_framework {

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

// CFood: Food class

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

class CPacman;

class CFood

{

public:

CFood();

CFood(int score);

bool HitPacman(CPacman \*pacman);// 是否碰到Pacman

bool IsAlive();// 是否活著

void LoadBitmap(int IDB1, int IDB2);// 載入圖形

void SetTopLeft(int x, int y);

void OnMove();// 移動

void OnShow();// 將圖形貼到畫面

int GetScore(); // 回傳food的分數

void SetXY(int nx, int ny);// 設定圓心的座標

void SetIsAlive(bool alive);// 設定是否活著

protected:

CAnimation foods;// food的圖

int x, y;// food的座標

bool is\_alive;// 是否活著

int score; // food的分數

private:

bool HitRectangle(int tx1, int ty1, int tx2, int ty2);// 是否碰到參數範圍的矩形

};

}

* CFood.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CPacman.h"

#include "CFood.h"

namespace game\_framework {

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

// CFood: Food class

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

CFood::CFood()

{

is\_alive = true;

x = y = 0;

}

CFood::CFood(int score)

{

is\_alive = true;

x = y = 0;

this->score = score;

}

bool CFood::HitPacman(CPacman \*pacman)

{

int adjust = 4;

// 檢測Pacman所構成的矩形是否碰到food

return HitRectangle(pacman->GetX1() + adjust, pacman->GetY1() + adjust, pacman->GetX2() - adjust, pacman->GetY2() - adjust);

}

bool CFood::HitRectangle(int tx1, int ty1, int tx2, int ty2)

{

int adjust = 4;

// 檢查是否有交集

return (tx2 >= x + adjust && tx1 <= (x + foods.Width() - adjust) && ty2 >= y + adjust && ty1 <= (y + foods.Height() - adjust));

}

bool CFood::IsAlive()

{

return is\_alive;

}

void CFood::LoadBitmap(int IDB1, int IDB2)

{

foods.AddBitmap(IDB1, RGB(0, 0, 0));

foods.AddBitmap(IDB2, RGB(0, 0, 0));

}

void CFood::OnMove()

{

if (!is\_alive) return;

}

void CFood::SetIsAlive(bool alive)

{

is\_alive = alive;

}

void CFood::SetXY(int nx, int ny)

{

x = nx; y = ny;

}

int CFood::GetScore()

{

return this->score;

}

void CFood::SetTopLeft(int x, int y)

{

foods.SetTopLeft(x, y);

this->x = x;

this->y = y;

}

void CFood::OnShow()

{

if (is\_alive) {

foods.SetTopLeft(x,y);

foods.OnMove();

foods.OnShow();

}

}

}

* CGameMap.h

namespace game\_framework {

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

// CGameMap: Map class

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

class CFood;

class CPacman;

enum MAP\_ID// 定義各種地圖的編號

{

MAP\_1,// 0

MAP\_2,// 1

MAP\_3,// 2

MAP\_END// 3

};

using namespace std;

class CGameMap

{

public:

CGameMap();

~CGameMap();

void LoadBitmap();// 載入地圖

void OnShow();// 顯示地圖

void SetMap(int maptype);// 選擇地圖

int getFoodCount();// 取得小豆子數量

vector<CFood \*>\* getAllFoods();// 取得所有小豆子

int \*\*GetMap();// 取得地圖

void nextMap();// 切換到下一張地圖

void lastMap();// 切換到最後一張地圖

protected:

const int NUMMAPS;// 總地圖數

void setFoods(int map\_info[31][28]);// 配置地圖上豆子的位置

CMovingBitmap \*walls;// 牆壁顏色

CMovingBitmap \*currentWall;// 目前牆壁顏色

vector<CFood \*> allFoods;// 小豆子and大魔豆

int \*\*map;// 構成地圖的陣列

const int X, Y;// 大地圖左上角的座標

const int MW, MH;// 小地圖的長跟寬

int foodCount;// 小豆子數量

int MapType;// 地圖類型

int \*gameMaps;// 地圖順序

int currentLevel = 0;// 目前地圖

};

}

* CGameMap.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CPacman.h"

#include "CFood.h"

#include "CGameMap.h"

namespace game\_framework {

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

// CGameMap: Map class

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

CGameMap::CGameMap() :X(20), Y(20), MW(24), MH(24), NUMMAPS(4)

{

map = new int\*[31];

for (int i = 0; i < 31; i++)

{

map[i] = new int[28];

}

gameMaps = new int[NUMMAPS];

walls = new CMovingBitmap[NUMMAPS];

gameMaps[0] = MAP\_1;

gameMaps[1] = MAP\_2;

gameMaps[2] = MAP\_3;

gameMaps[3] = MAP\_END;

MapType = gameMaps[0];

currentWall = &walls[0];

}

CGameMap::~CGameMap()

{

for (int i = 0; i < (int)allFoods.size(); i++)

{

delete allFoods.at(i);

}

for (int i = 0; i < 31; i++)

{

delete[] map[i];

}

delete[] map;

delete[] walls;

delete[] gameMaps;

}

void CGameMap::setFoods(int map\_info[31][28])

{

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

{

delete allFoods.at(i);

}

allFoods.clear();

foodCount = 0;

if (MapType == MAP\_END)

{

for (int i = 0 ; i < 31; i++)

{

for (int j = 0; j < 28; j++)

{

map[i][j] = map\_info[i][j];

}

}

allFoods.push\_back(new CFood(10));

allFoods.at(0)->LoadBitmap(IDB\_FOOD, IDB\_FOOD);

foodCount++;

return;

}

for (int i = 0; i < 31; i++)

{

for (int j = 0; j < 28; j++)

{

map[i][j] = map\_info[i][j];

if (map\_info[i][j] == 0 || map\_info[i][j] == 4)

{ // 小豆子

allFoods.push\_back(new CFood(10));

allFoods.at(foodCount)->LoadBitmap(IDB\_FOOD, IDB\_FOOD);

foodCount++;

}

else if (map\_info[i][j] == 3)

{ // 大魔豆

allFoods.push\_back(new CFood(50));

allFoods.at(foodCount)->LoadBitmap(IDB\_MAGICFOOD, IDB\_BLACKBMP);

foodCount++;

}

}

}

}

void CGameMap::SetMap(int maptype)

{

MapType = maptype;

if (MapType == MAP\_1)

{

currentLevel = 0;

int map\_info[31][28] = {

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

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

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

{1, 3, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 3, 1},

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

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

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

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

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

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

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

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

{1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 2, 2, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1},

{1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1},

{0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1},

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

{1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1},

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

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

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

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

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

{1, 3, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 3, 1},

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

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

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

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

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

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

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

setFoods(map\_info);

}

else if (MapType == MAP\_2)

{

currentLevel = 1;

int map\_info[31][28] = {

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

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

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

{1, 3, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 3, 1},

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

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

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

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

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

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

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

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

{1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 2, 2, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1},

{1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1},

{1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1},

{1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1},

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

{0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0},

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

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

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

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

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

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

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

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

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

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

{1, 3, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 3, 1},

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

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

};

setFoods(map\_info);

}

else if (MapType == MAP\_3)

{

currentLevel = 2;

int map\_info[31][28] = {

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

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

{1, 3, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 3, 1},

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

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

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

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

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

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

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

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

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

{1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 2, 2, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1},

{1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1},

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

{1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1},

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

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

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

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

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

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

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

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

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

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

{1, 3, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 3, 1},

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

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

};

setFoods(map\_info);

}

else if (MapType == MAP\_END)

{

currentLevel = 2;

int map\_info[31][28] = {

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

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

{1, 3, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 3, 1},

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

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

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

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

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

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

{4, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 4},

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

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

{1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 2, 2, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1},

{1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1},

{1, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1},

{1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 2, 2, 2, 2, 2, 2, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1},

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

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

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

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

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

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

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

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

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

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

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

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

{1, 3, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 3, 1},

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

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

};

setFoods(map\_info);

}

}

void CGameMap::nextMap()

{

currentLevel++;

SetMap(gameMaps[currentLevel]);

MapType = gameMaps[currentLevel];

currentWall = &walls[currentLevel];

}

void CGameMap::lastMap()

{

SetMap(gameMaps[3]);

MapType = gameMaps[3];

currentWall = &walls[3];

}

int CGameMap::getFoodCount()

{

return foodCount;

}

void CGameMap::LoadBitmap()

{

walls[0].LoadBitmap(IDB\_MAP1);

walls[1].LoadBitmap(IDB\_MAP2);

walls[2].LoadBitmap(IDB\_MAP3);

walls[3].LoadBitmap(IDB\_MAP3);

}

vector<CFood \*>\* CGameMap::getAllFoods()

{

return &allFoods;

}

void CGameMap::OnShow()

{

int currentFoodIndex = 0;

switch (MapType)

{

case MAP\_1:

walls[0].SetTopLeft(MAP\_START, MAP\_START);

walls[0].ShowBitmap();

break;

case MAP\_2:

walls[1].SetTopLeft(MAP\_START, MAP\_START);

walls[1].ShowBitmap();

break;

case MAP\_3:

walls[2].SetTopLeft(MAP\_START, MAP\_START);

walls[2].ShowBitmap();

break;

case MAP\_END:

walls[3].SetTopLeft(MAP\_START, MAP\_START);

walls[3].ShowBitmap();

break;

default:

break;

}

if (MapType == MAP\_END)

{

allFoods.at(0)->SetTopLeft(X + (MW \* 16), Y + (MH \* 17));

allFoods.at(0)->OnShow();

}

else {

for (int i = 0; i < 31; i++)

{

for (int j = 0; j < 28; j++)

{

switch (map[i][j])

{

case 0:

allFoods.at(currentFoodIndex)->SetTopLeft(X + (MW\*j), Y + (MH\*i));

allFoods.at(currentFoodIndex)->OnShow();

currentFoodIndex++;

break;

case 1:

break;

case 3:

allFoods.at(currentFoodIndex)->SetTopLeft(X + (MW\*j), Y + (MH\*i));

allFoods.at(currentFoodIndex)->OnShow();

currentFoodIndex++;

break;

default:

break;

}

}

}

}

}

int \*\*CGameMap::GetMap()

{

return map;

}

}

* CLevel.h

namespace game\_framework {

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

// CLevel: Level class

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

class CLevel

{

public:

CLevel();

void LoadBitmap();

void SetTopLeft();

void OnShow();

void setLevel(int LV);// 紀錄關卡等級

void levelUp();// 關卡升級

int getLevel();// 取得關卡等級

private:

CMovingBitmap LEVEL\_text;// LEVEL文字

CInteger myLevel;// 關卡等級

};

}

* CLevel.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CLevel.h"

namespace game\_framework {

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

// CLevel: Level class

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

CLevel::CLevel()

{

// Level預設為1位數

myLevel = CInteger(1);

// 預設為1

myLevel.SetInteger(1);

}

void CLevel::LoadBitmap()

{

LEVEL\_text.LoadBitmap(IDB\_LEVEL\_TEXT, RGB(255, 255, 255));

myLevel.LoadBitmap();

}

void CLevel::setLevel(int LV)

{

myLevel.SetInteger(LV);

// 計算level的位數

int d = 1;

while (true)

{

LV = LV / 10;

if (LV != 0) d++;

else

{

myLevel.SetDigits(d); // 重設level的位數

break;

}

}

}

void CLevel::levelUp()

{

int nextLevel = myLevel.GetInteger() + 1;

setLevel(nextLevel);

}

int CLevel::getLevel()

{

return myLevel.GetInteger();

}

void CLevel::SetTopLeft()

{

LEVEL\_text.SetTopLeft(800, 130);

myLevel.SetTopLeft(820, 155);

}

void CLevel::OnShow()

{

LEVEL\_text.ShowBitmap();

myLevel.ShowBitmap();

}

}

* CLife.h

namespace game\_framework {

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

// CLife: Life class

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

class CLife

{

public:

CLife();

void LoadBitmap();

void SetTopLeft();

void OnShow();

void setLife(int life);// 紀錄生命數量

int getLife();// 取得生命數量

private:

CMovingBitmap Pacman\_icon[4];// Pacman圖片顯示

CMovingBitmap Life\_multi;// Life的 X 號

CInteger myLife;// 生命數量

};

}

* CLife.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CLife.h"

namespace game\_framework {

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

// CLife: Life class

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

CLife::CLife()

{

// Life預設為1位數

myLife = CInteger(1);

// 預設為4

myLife.SetInteger(4);

}

void CLife::LoadBitmap()

{

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

{

Pacman\_icon[i].LoadBitmap(IDB\_PACMAN\_RIGHT\_1, RGB(125, 125, 125));

}

Life\_multi.LoadBitmap(IDB\_LIFEMULTI, RGB(0, 0, 0));

myLife.LoadBitmap();

}

void CLife::OnShow()

{

for (int i = 0; i < myLife.GetInteger(); i++)

{

Pacman\_icon[i].ShowBitmap();

}

Life\_multi.ShowBitmap();

myLife.ShowBitmap();

}

void CLife::SetTopLeft()

{

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

{

Pacman\_icon[i].SetTopLeft(800 + 25\*i, 360);

}

Life\_multi.SetTopLeft(800, 400);

myLife.SetTopLeft(830, 400);

}

void CLife::setLife(int life)

{

myLife.SetDigits(1);

myLife.SetInteger(life);

}

int CLife::getLife()

{

return myLife.GetInteger();

}

}

* CScore.h

namespace game\_framework {

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

// CScore: Score class

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

class CScore

{

public:

CScore();

void LoadBitmap();

void SetTopLeft();

void OnShow();

void setScore(int profit);// 紀錄分數

int getScore();// 取得分數

private:

CMovingBitmap SCORE\_text;// Score的lable

CInteger myScore;// 累計的分數

int totalScore;// 總分(integer)

};

}

* CScore.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "CScore.h"

namespace game\_framework {

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

// CScore: Score class

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

CScore::CScore()

{

myScore.SetInteger(0);

// 預設score只有1位數 (0);

myScore.SetDigits(1);

}

void CScore::LoadBitmap()

{

SCORE\_text.LoadBitmap(IDB\_SCORE\_TEXT, RGB(255, 255, 255));

myScore.LoadBitmap();

}

void CScore::SetTopLeft()

{

SCORE\_text.SetTopLeft(800, 50);

myScore.SetTopLeft(820, 75);

}

void CScore::setScore(int profit)

{

int newScore = profit;

myScore.SetInteger(newScore);

// 計算score的位數

int d = 1;

while (true)

{

newScore = newScore / 10;

if (newScore != 0)

{

d++;

}

else

{

myScore.SetDigits(d); // 重設score的位數

break;

}

}

}

int CScore::getScore()

{

return myScore.GetInteger();

}

void CScore::OnShow()

{

SCORE\_text.ShowBitmap();

myScore.ShowBitmap();

}

}