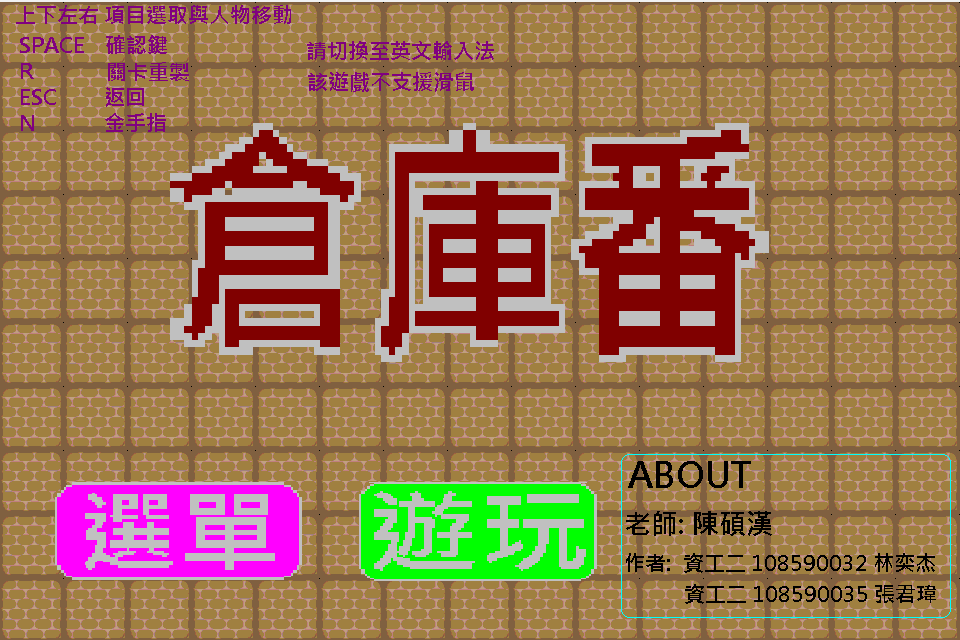
國立臺北科技大學

2020 Spring 資工系物件導向程式實習

期末報告

倉庫番



第12組

108590032 林奕杰

108590035 張君瑋

**目錄**

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

玩法單純、需要思考的益智遊戲—倉庫番，也有些人將它叫做推箱子，無論是在網路上，或是掌上型遊戲機，都很常見到這個遊戲。只需要操作人物移動，將場上的箱子推到指定位置，即可過關。

倉庫番的歷史悠久，可說是個經典遊戲，於是我們決定，利用自己所學，將這款遊戲重現。

1. 分工：

我們會先進行討論，可能是遊戲需要什麼功能，或是各個類別需要什麼方式。討論完畢後，君瑋會提供一些程式想法給我，而我就開始打程式，同時間君瑋會設計遊戲中會用到的圖片檔。分工如下：

張君瑋：提供程式想法、設計遊戲圖片檔案

林奕杰：設計程式、確認程式可行

1. **遊戲介紹**
2. 遊戲說明：

遊戲規則：將場上所有的箱子推到指定地點，即可通關

遊玩方式：

方向鍵上下左右：控制人物移動

特殊功能：

R鍵：重製關卡

密技：

N鍵：直接過關

1. 遊戲圖形：

（圖一：人物）

（圖二：箱子）

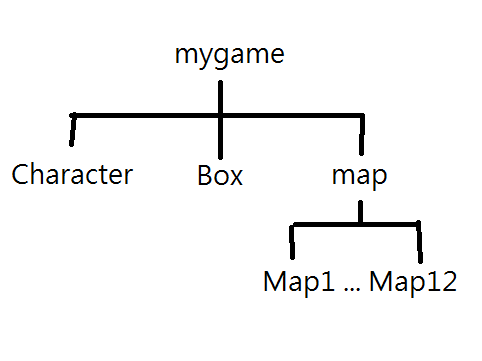
（圖三：終點）

（圖四：牆壁）

1. 遊戲音效：

只使用範例中的ntut.mid音效

1. **程式設計**
2. 程式架構：

mygame類別用到了Character類別、Box類別、繼承自map類別的Map1~Map12類別

1. 程式類別：

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h 檔行數 | .cpp 檔行數 | 說明 |
| Character | 32 | 152 | 人物基本物件 |
| Box | 21 | 72 | 箱子基本物件 |
| Map | 78 | 318 | 各物件位置設計，包含Map1~Map12 |
| 總行數 | 131 | 542 |  |

1. 程式技術：

並沒有特別使用什麼資料結構或演算方式

1. **結語**
2. 問題及解決方法：

因為倉庫番的遊戲方式較為簡單，於是在實作上並沒有遇到太大問題，就算有也可以很快的解決，但其中有道問題困擾我們許久。

在最初的設計，人物的移動很不流暢，可以明顯看出人物是一格一格移動，最後我們是讓人物移動時的距離縮短，並讓它能在碰牆之類的情況下，重設它的座標，讓畫面上看起來較漂亮。

1. 時間表：

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 組員 – 林奕杰(小時) | 組員 – 張君瑋(小時) | 說明 |
| 1 | 10 | 10 | 討論題目 |
| 2 | 0 | 10 | 素材蒐集、設計 |
| 3 | 12 | 12 | 畫面顯示 |
| 4 | 5 | 10 | 素材蒐集、設計 |
| 5 | 12 | 10 | 基本程式設計 |
| 6 | 10 | 12 | 討論程式問題 |
| 7 | 10 | 8 | 基本程式設計 |
| 8 | 10 | 11 | 遊戲錯誤解決 |
| 9 | 5 | 7 | 遊戲debug |
| 10 | 0 | 12 | 素材蒐集、設計 |
| 11 | 0 | 10 | 遊戲debug |
| 12 | 0 | 0 |  |
| 13 | 5 | 5 | 討論 |
| 14 | 10 | 5 | 基本程式設計 |
| 15 | 0 | 0 |  |
| 16 | 15 | 10 | Debug |
| 17 | 10 | 10 | 期末需求項目 |
| 總時數 | 114 | 142 |  |

1. 貢獻比例：

張君瑋：50%

林奕杰：50%

1. 自我檢核表：

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 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之用法與密技 | ■已完成 □未完成 |  |
| 5 | 上傳 setup/apk/source 檔 | ■已完成 □未完成 |  |
| 6 | setup 檔可正確執行 | ■已完成 □未完成 |  |
| 7 | 報告字型、點數、對齊、行距、頁碼等格式正確 | ■已完成 □未完成 |  |
| 8 | 全螢幕啟動–改列加分項目 | □已完成■未完成 | 一開始視窗比例大小不對，但畫面已設計好，因此不設計全螢幕功能 |

1. 收穫：

張君瑋：

這學期如何把陣列0，1轉換成地圖，等等，且程式文法跟上學期的OOP有許多關聯，像是:protected之類的讓我更了解他們的運用方式，protected 修飾符來指定訪問僅限於包含類型。此外，我們可以將它用於從包含類派生的類型。受保護一詞意味著它本身以及派生類都可以訪問或可見。

在此成員或類型的幫助下，只能由在同一類中使用或在派生類中使用的代碼訪問。關鍵字protected 位於private 和public 修飾符之間。它幾乎與私有修飾符相同，但它允許成員訪問派生類。當我們想要讓孩子的父母訪問孩子的屬性時，我們主要使用 protected 關鍵字。所以我們可以在 protected 關鍵字的幫助下重用邏輯。

至於除錯方面，大部分都是我理解跟文法錯誤這些基本錯誤。

林奕杰：

由於OOP的概念上學期就學到了，所以文法什麼的沒有遇到太大的障礙。至於技巧，因為沒有用到特別的資料結構或演算法，不確定在這方面有收穫。這次實作最大的收穫，我想果然還是能正確並好好的使用framework提供的method，以前都是從無到有把程式打出來，但這次不是，而是利用現有的method設計出遊戲，與之前相比，多了一份感動。

1. 心得、感想：

張君瑋：

經過這學期的專題讓我對於於製作遊戲的觀念想法更加了解對於未來製作觀的專案有很大的幫助，想當初我們原本要做音樂判定的遊戲，還好老師當初有阻止我們做傻事哈哈哈哈，否則我們現在肯定累慘，但現在這個倉庫翻也不容易，應為要判定箱子和人物是否有撞到牆壁這個判定用了許久，而且有卡牆的小BUG，所以用了許久，這學期真的學習到許多東西，也非常謝謝我的夥伴奕杰沒有他的話我根本沒辦法順利做出來這個有趣的遊戲。

林奕杰：

幸好我們當初選的遊戲並沒有那麼複雜，否則工作量會更多，不過剛開始真的蠻不好過的，用一個自己並不是那麼熟悉的method，得花不少時間理解。不過熟悉之後，想到不用自己寫最底下的控制類別，慶幸的不得了。也感謝君瑋，有人能討論，寫程式真的方便多了。

1. **附錄**

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

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

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap homepage; // 倉庫番

CMovingBitmap background; // 背景箱子

CMovingBitmap helpText1; // 按鍵用法

CMovingBitmap menu; // 選單1

CMovingBitmap menuInv; // 選單2

CMovingBitmap play; // 遊玩1

CMovingBitmap playInv; // 遊玩2

CMovingBitmap aboutText; // 介紹

CMovingBitmap stage1; // 關卡1

CMovingBitmap stage1Inv;

CMovingBitmap stage2; // 關卡2

CMovingBitmap stage2Inv;

CMovingBitmap stage3; // 關卡3

CMovingBitmap stage3Inv;

CMovingBitmap stage4; // 關卡4

CMovingBitmap stage4Inv;

CMovingBitmap stage5; // 關卡5

CMovingBitmap stage5Inv;

CMovingBitmap stage6; // 關卡6

CMovingBitmap stage6Inv;

CMovingBitmap stage7; // 關卡7

CMovingBitmap stage7Inv;

CMovingBitmap stage8; // 關卡8

CMovingBitmap stage8Inv;

CMovingBitmap stage9; // 關卡9

CMovingBitmap stage9Inv;

CMovingBitmap stage10; // 關卡10

CMovingBitmap stage10Inv;

CMovingBitmap stage11; // 關卡11

CMovingBitmap stage11Inv;

CMovingBitmap stage12; // 關卡12

CMovingBitmap stage12Inv;

int chooseMenu; // 目前是選哪個 菜單選項

int chooseStage; // 目前是選哪個 關卡

bool inChooseStage; // 選擇關卡中

};

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:

void SetObjectPosition(Map gameMap); // 依據不同地圖，設定箱子和人物物件位置

void CheckManHitWall(Map gameMap); // 依據不同地圖，判斷人物有沒有撞牆

void CheckManHitBox(Map gameMap); // 依據不同地圖，判斷人物有沒有撞箱子

bool CheckWin(Map gameMap); // 判斷該地圖箱子有沒有都在終點上

CMovingBitmap background; // 白底背景

Character man; // 人物物件

Box box[8]; // 箱子物件

Map1 gamemap1; // 地圖物件

Map2 gamemap2;

Map3 gamemap3;

Map4 gamemap4;

Map5 gamemap5;

Map6 gamemap6;

Map7 gamemap7;

Map8 gamemap8;

Map9 gamemap9;

Map10 gamemap10;

Map11 gamemap11;

Map12 gamemap12;

int whichMap;

};

class CGameStateOver : public CGameState {

public:

CGameStateOver(CGame \*g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Down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

int chooseMenu; // 選哪個

CMovingBitmap chooseStage; // 回去選關卡

CMovingBitmap chooseStageInv;

CMovingBitmap exit; // 返回主選單

CMovingBitmap exitInv;

CMovingBitmap nextStage; // 去下一關

CMovingBitmap nextStageInv;

CMovingBitmap sign; // 恭喜過關

};

namespace game\_framework {

// 目前關卡

int newStage = 1;

// 是否進入關卡選單

bool initStage = false;

CGameStateInit::CGameStateInit(CGame \*g)

: CGameState(g)

{

chooseMenu = 0;

chooseStage = 0;

inChooseStage = false;

}

void CGameStateInit::OnInit()

{

ShowInitProgress(0);

homepage.LoadBitmap(IDB\_HOMEPAGE, RGB(255, 255, 255));

background.LoadBitmap(IDB\_WALL, RGB(255, 255, 255));

helpText1.LoadBitmap(IDB\_HELPTEXT1, RGB(255, 255, 255));

menu.LoadBitmap(IDB\_MENU, RGB(255, 255, 255));

menuInv.LoadBitmap(IDB\_MENUINV, RGB(255, 255, 255));

play.LoadBitmap(IDB\_PLAY, RGB(255, 255, 255));

playInv.LoadBitmap(IDB\_PLAYINV, RGB(255, 255, 255));

aboutText.LoadBitmap(IDB\_ABOUTTEXT, RGB(255, 255, 255));

stage1.LoadBitmap(IDB\_STAGE1, RGB(255, 255, 255));

stage1Inv.LoadBitmap(IDB\_STAGE1INV, RGB(255, 255, 255));

stage2.LoadBitmap(IDB\_STAGE2, RGB(255, 255, 255));

stage2Inv.LoadBitmap(IDB\_STAGE2INV, RGB(255, 255, 255));

stage3.LoadBitmap(IDB\_STAGE3, RGB(255, 255, 255));

stage3Inv.LoadBitmap(IDB\_STAGE3INV, RGB(255, 255, 255));

stage4.LoadBitmap(IDB\_STAGE4, RGB(255, 255, 255));

stage4Inv.LoadBitmap(IDB\_STAGE4INV, RGB(255, 255, 255));

stage5.LoadBitmap(IDB\_STAGE5, RGB(255, 255, 255));

stage5Inv.LoadBitmap(IDB\_STAGE5INV, RGB(255, 255, 255));

stage6.LoadBitmap(IDB\_STAGE6, RGB(255, 255, 255));

stage6Inv.LoadBitmap(IDB\_STAGE6INV, RGB(255, 255, 255));

stage7.LoadBitmap(IDB\_STAGE7, RGB(255, 255, 255));

stage7Inv.LoadBitmap(IDB\_STAGE7INV, RGB(255, 255, 255));

stage8.LoadBitmap(IDB\_STAGE8, RGB(255, 255, 255));

stage8Inv.LoadBitmap(IDB\_STAGE8INV, RGB(255, 255, 255));

stage9.LoadBitmap(IDB\_STAGE9, RGB(255, 255, 255));

stage9Inv.LoadBitmap(IDB\_STAGE9INV, RGB(255, 255, 255));

stage10.LoadBitmap(IDB\_STAGE10, RGB(255, 255, 255));

stage10Inv.LoadBitmap(IDB\_STAGE10INV, RGB(255, 255, 255));

stage11.LoadBitmap(IDB\_STAGE11, RGB(255, 255, 255));

stage11Inv.LoadBitmap(IDB\_STAGE11INV, RGB(255, 255, 255));

stage12.LoadBitmap(IDB\_STAGE12, RGB(255, 255, 255));

stage12Inv.LoadBitmap(IDB\_STAGE12INV, RGB(255, 255, 255));

}

void CGameStateInit::OnBeginState()

{

inChooseStage = initStage;

}

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

{

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

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

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

if (nChar == KEY\_LEFT && chooseMenu > 1) {

chooseMenu = chooseMenu - 1;

}

if (nChar == KEY\_RIGHT && chooseMenu < 2) {

chooseMenu = chooseMenu + 1;

}

if (nChar == KEY\_LEFT && chooseStage > 1) {

chooseStage = chooseStage - 1;

}

if (nChar == KEY\_RIGHT && chooseStage < 12) {

chooseStage = chooseStage + 1;

}

if (nChar == KEY\_SPACE && inChooseStage) {

newStage = chooseStage;

GotoGameState(GAME\_STATE\_RUN);

}

if (nChar == KEY\_SPACE) {

switch (chooseMenu)

{

case 1:

inChooseStage = true;

break;

case 2:

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

break;

default:

break;

}

}

if (nChar == KEY\_ESC) { // Demo 關閉遊戲的方法

if (inChooseStage) {

chooseMenu = 0;

chooseStage = 0;

inChooseStage = false;

}

else {

PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, 0, 0); // 關閉遊戲

}

}

}

void CGameStateInit::OnShow()

{

for (int i = 0; i < 15; i++) {

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

background.SetTopLeft(i \* 64, j \* 64);

background.ShowBitmap();

}

}

helpText1.SetTopLeft(0, 0);

helpText1.ShowBitmap();

if (!inChooseStage) {

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

menu.SetTopLeft(50, 480);

menuInv.SetTopLeft(50, 480);

play.SetTopLeft(350, 480);

playInv.SetTopLeft(350, 480);

aboutText.SetTopLeft(620, 450);

menu.ShowBitmap();

play.ShowBitmap();

aboutText.ShowBitmap();

homepage.ShowBitmap();

if (chooseMenu == 1) {

menuInv.ShowBitmap();

play.ShowBitmap();

}

else if (chooseMenu == 2) {

menu.ShowBitmap();

playInv.ShowBitmap();

}

else {

menu.ShowBitmap();

play.ShowBitmap();

}

}

else {

switch (chooseStage)

{

case 0:

stage1.SetTopLeft((SIZE\_X - stage1.Width()) / 2, SIZE\_Y / 4);

stage1.ShowBitmap();

stage2.SetTopLeft(448 + (SIZE\_X - stage2.Width()) / 2, SIZE\_Y / 4);

stage2.ShowBitmap();

break;

case 1:

stage1Inv.SetTopLeft((SIZE\_X - stage1Inv.Width()) / 2, SIZE\_Y / 4);

stage1Inv.ShowBitmap();

stage2.SetTopLeft(448 + (SIZE\_X - stage2.Width()) / 2, SIZE\_Y / 4);

stage2.ShowBitmap();

break;

case 2:

stage1.SetTopLeft(-448 + (SIZE\_X - stage1.Width()) / 2, SIZE\_Y / 4);

stage1.ShowBitmap();

stage2Inv.SetTopLeft((SIZE\_X - stage2Inv.Width()) / 2, SIZE\_Y / 4);

stage2Inv.ShowBitmap();

stage3.SetTopLeft(448 + (SIZE\_X - stage3.Width()) / 2, SIZE\_Y / 4);

stage3.ShowBitmap();

break;

case 3:

stage2.SetTopLeft(-448 + (SIZE\_X - stage2.Width()) / 2, SIZE\_Y / 4);

stage2.ShowBitmap();

stage3Inv.SetTopLeft((SIZE\_X - stage3Inv.Width()) / 2, SIZE\_Y / 4);

stage3Inv.ShowBitmap();

stage4.SetTopLeft(448 + (SIZE\_X - stage4.Width()) / 2, SIZE\_Y / 4);

stage4.ShowBitmap();

break;

case 4:

stage3.SetTopLeft(-448 + (SIZE\_X - stage3.Width()) / 2, SIZE\_Y / 4);

stage3.ShowBitmap();

stage4Inv.SetTopLeft((SIZE\_X - stage4Inv.Width()) / 2, SIZE\_Y / 4);

stage4Inv.ShowBitmap();

stage5.SetTopLeft(448 + (SIZE\_X - stage5.Width()) / 2, SIZE\_Y / 4);

stage5.ShowBitmap();

break;

case 5:

stage4.SetTopLeft(-448 + (SIZE\_X - stage4.Width()) / 2, SIZE\_Y / 4);

stage4.ShowBitmap();

stage5Inv.SetTopLeft((SIZE\_X - stage5Inv.Width()) / 2, SIZE\_Y / 4);

stage5Inv.ShowBitmap();

stage6.SetTopLeft(448 + (SIZE\_X - stage6.Width()) / 2, SIZE\_Y / 4);

stage6.ShowBitmap();

break;

case 6:

stage5.SetTopLeft(-448 + (SIZE\_X - stage5.Width()) / 2, SIZE\_Y / 4);

stage5.ShowBitmap();

stage6Inv.SetTopLeft((SIZE\_X - stage6Inv.Width()) / 2, SIZE\_Y / 4);

stage6Inv.ShowBitmap();

stage7.SetTopLeft(448 + (SIZE\_X - stage7.Width()) / 2, SIZE\_Y / 4);

stage7.ShowBitmap();

break;

case 7:

stage6.SetTopLeft(-448 + (SIZE\_X - stage6.Width()) / 2, SIZE\_Y / 4);

stage6.ShowBitmap();

stage7Inv.SetTopLeft((SIZE\_X - stage7Inv.Width()) / 2, SIZE\_Y / 4);

stage7Inv.ShowBitmap();

stage8.SetTopLeft(448 + (SIZE\_X - stage8.Width()) / 2, SIZE\_Y / 4);

stage8.ShowBitmap();

break;

case 8:

stage7.SetTopLeft(-448 + (SIZE\_X - stage7.Width()) / 2, SIZE\_Y / 4);

stage7.ShowBitmap();

stage8Inv.SetTopLeft((SIZE\_X - stage8Inv.Width()) / 2, SIZE\_Y / 4);

stage8Inv.ShowBitmap();

stage9.SetTopLeft(448 + (SIZE\_X - stage9.Width()) / 2, SIZE\_Y / 4);

stage9.ShowBitmap();

break;

case 9:

stage8.SetTopLeft(-448 + (SIZE\_X - stage8.Width()) / 2, SIZE\_Y / 4);

stage8.ShowBitmap();

stage9Inv.SetTopLeft((SIZE\_X - stage9Inv.Width()) / 2, SIZE\_Y / 4);

stage9Inv.ShowBitmap();

stage10.SetTopLeft(448 + (SIZE\_X - stage10.Width()) / 2, SIZE\_Y / 4);

stage10.ShowBitmap();

break;

case 10:

stage9.SetTopLeft(-448 + (SIZE\_X - stage9.Width()) / 2, SIZE\_Y / 4);

stage9.ShowBitmap();

stage10Inv.SetTopLeft((SIZE\_X - stage10Inv.Width()) / 2, SIZE\_Y / 4);

stage10Inv.ShowBitmap();

stage11.SetTopLeft(448 + (SIZE\_X - stage11.Width()) / 2, SIZE\_Y / 4);

stage11.ShowBitmap();

break;

case 11:

stage10.SetTopLeft(-448 + (SIZE\_X - stage10.Width()) / 2, SIZE\_Y / 4);

stage10.ShowBitmap();

stage11Inv.SetTopLeft((SIZE\_X - stage11Inv.Width()) / 2, SIZE\_Y / 4);

stage11Inv.ShowBitmap();

stage12.SetTopLeft(448 + (SIZE\_X - stage12.Width()) / 2, SIZE\_Y / 4);

stage12.ShowBitmap();

break;

case 12:

stage11.SetTopLeft(-448 + (SIZE\_X - stage11.Width()) / 2, SIZE\_Y / 4);

stage11.ShowBitmap();

stage12Inv.SetTopLeft((SIZE\_X - stage12Inv.Width()) / 2, SIZE\_Y / 4);

stage12Inv.ShowBitmap();

break;

}

}

}

CGameStateOver::CGameStateOver(CGame \*g)

: CGameState(g)

{

chooseMenu = 0;

}

void CGameStateOver::OnMove()

{

}

void CGameStateOver::OnBeginState()

{

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

}

void CGameStateOver::OnInit()

{

chooseStage.LoadBitmap(IDB\_CHOOSESTAGE, RGB(255, 255, 255));

chooseStageInv.LoadBitmap(IDB\_CHOOSESTAGEINV, RGB(255, 255, 255));

exit.LoadBitmap(IDB\_EXIT, RGB(255, 255, 255));

exitInv.LoadBitmap(IDB\_EXITINV, RGB(255, 255, 255));

nextStage.LoadBitmap(IDB\_NEXTSTAGE, RGB(255, 255, 255));

nextStageInv.LoadBitmap(IDB\_NEXTSTAGEINV, RGB(255, 255, 255));

sign.LoadBitmap(IDB\_SIGN, RGB(255, 255, 255));

}

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

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

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

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

if (nChar == KEY\_UP && chooseMenu > 1) {

chooseMenu = chooseMenu - 1;

}

if (nChar == KEY\_DOWN && chooseMenu < 3) {

chooseMenu = chooseMenu + 1;

}

if (nChar == KEY\_SPACE) {

switch (chooseMenu)

{

case 1: // 下一關

GotoGameState(GAME\_STATE\_RUN);

break;

case 2: // 選擇關卡

initStage = true;

GotoGameState(GAME\_STATE\_INIT);

break;

case 3: //

initStage = false;

newStage = 1;

GotoGameState(GAME\_STATE\_INIT);

break;

default:

break;

}

}

}

void CGameStateOver::OnShow()

{

nextStage.SetTopLeft(600, 100);

nextStageInv.SetTopLeft(600, 100);

chooseStage.SetTopLeft(600, 300);

chooseStageInv.SetTopLeft(600, 300);

exit.SetTopLeft(600, 500);

exitInv.SetTopLeft(600, 500);

sign.SetTopLeft(75, 75);

switch (chooseMenu)

{

case 1:

if (newStage != 13) {

nextStageInv.ShowBitmap();

}

chooseStage.ShowBitmap();

exit.ShowBitmap();

break;

case 2:

if (newStage != 13) {

nextStage.ShowBitmap();

}

chooseStageInv.ShowBitmap();

exit.ShowBitmap();

break;

case 3:

if (newStage != 13) {

nextStage.ShowBitmap();

}

chooseStage.ShowBitmap();

exitInv.ShowBitmap();

break;

default:

if (newStage != 13) {

nextStage.ShowBitmap();

}

chooseStage.ShowBitmap();

exit.ShowBitmap();

break;

}

sign.ShowBitmap();

}

CGameStateRun::CGameStateRun(CGame \*g) : CGameState(g)

{

whichMap = 1;

}

CGameStateRun::~CGameStateRun()

{

}

void CGameStateRun::OnBeginState()

{

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

whichMap = newStage;

switch (whichMap)

{

case 1:

SetObjectPosition(gamemap1);

break;

case 2:

SetObjectPosition(gamemap2);

break;

case 3:

SetObjectPosition(gamemap3);

break;

case 4:

SetObjectPosition(gamemap4);

break;

case 5:

SetObjectPosition(gamemap5);

break;

case 6:

SetObjectPosition(gamemap6);

break;

case 7:

SetObjectPosition(gamemap7);

break;

case 8:

SetObjectPosition(gamemap8);

break;

case 9:

SetObjectPosition(gamemap9);

break;

case 10:

SetObjectPosition(gamemap10);

break;

case 11:

SetObjectPosition(gamemap11);

break;

case 12:

SetObjectPosition(gamemap12);

break;

default:

break;

}

}

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

{

man.OnMove();

switch (whichMap) {

case 1:

CheckManHitBox(gamemap1);

CheckManHitWall(gamemap1);

if (CheckWin(gamemap1)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 2:

CheckManHitBox(gamemap2);

CheckManHitWall(gamemap2);

if (CheckWin(gamemap2)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 3:

CheckManHitBox(gamemap3);

CheckManHitWall(gamemap3);

if (CheckWin(gamemap3)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 4:

CheckManHitBox(gamemap4);

CheckManHitWall(gamemap4);

if (CheckWin(gamemap4)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 5:

CheckManHitBox(gamemap5);

CheckManHitWall(gamemap5);

if (CheckWin(gamemap5)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 6:

CheckManHitBox(gamemap6);

CheckManHitWall(gamemap6);

if (CheckWin(gamemap6)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 7:

CheckManHitBox(gamemap7);

CheckManHitWall(gamemap7);

if (CheckWin(gamemap7)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 8:

CheckManHitBox(gamemap8);

CheckManHitWall(gamemap8);

if (CheckWin(gamemap8)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 9:

CheckManHitBox(gamemap9);

CheckManHitWall(gamemap9);

if (CheckWin(gamemap9)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 10:

CheckManHitBox(gamemap10);

CheckManHitWall(gamemap10);

if (CheckWin(gamemap10)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 11:

CheckManHitBox(gamemap11);

CheckManHitWall(gamemap11);

if (CheckWin(gamemap11)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

case 12:

CheckManHitBox(gamemap12);

CheckManHitWall(gamemap12);

if (CheckWin(gamemap12)) {

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

break;

default:

break;

}

}

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

{

CAudio::Instance()->Load(AUDIO\_NTUT, "sounds\\ntut.mid");

background.LoadBitmap(IDB\_BACKGROUND2);

man.LoadBitmap();

for (int i = 0; i < 8; i++) {

box[i].LoadBitmap();

}

gamemap1.LoadBitmap();

gamemap2.LoadBitmap();

gamemap3.LoadBitmap();

gamemap4.LoadBitmap();

gamemap5.LoadBitmap();

gamemap6.LoadBitmap();

gamemap7.LoadBitmap();

gamemap8.LoadBitmap();

gamemap9.LoadBitmap();

gamemap10.LoadBitmap();

gamemap11.LoadBitmap();

gamemap12.LoadBitmap();

}

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

{

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

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

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

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

const char KEY\_RESET = 0x52; // R 鍵

const char KEY\_NEXTSTAGE = 0x4E; // N 鍵

if (nChar == KEY\_RESET) {

OnBeginState();

}

if (nChar == KEY\_NEXTSTAGE) {

man.SetMovingUp(false);

man.SetMovingDown(false);

man.SetMovingLeft(false);

man.SetMovingRight(false);

newStage = ++whichMap;

GotoGameState(GAME\_STATE\_OVER);

}

if (nChar == KEY\_LEFT) {

man.SetMovingLeft(true);

man.SetMovingRight(false);

man.SetMovingUp(false);

man.SetMovingDown(false);

}

if (nChar == KEY\_RIGHT) {

man.SetMovingLeft(false);

man.SetMovingRight(true);

man.SetMovingUp(false);

man.SetMovingDown(false);

}

if (nChar == KEY\_UP) {

man.SetMovingLeft(false);

man.SetMovingRight(false);

man.SetMovingUp(true);

man.SetMovingDown(false);

}

if (nChar == KEY\_DOWN) {

man.SetMovingLeft(false);

man.SetMovingRight(false);

man.SetMovingUp(false);

man.SetMovingDown(true);

}

}

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) {

man.SetMovingLeft(false);

}

if (nChar == KEY\_RIGHT) {

man.SetMovingRight(false);

}

if (nChar == KEY\_UP) {

man.SetMovingUp(false);

}

if (nChar == KEY\_DOWN) {

man.SetMovingDown(false);

}

}

void CGameStateRun::OnShow()

{

background.SetTopLeft(0, 0);

background.ShowBitmap();

switch (whichMap){

case 1:

gamemap1.OnShow();

for (int i = 0; i < gamemap1.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 2:

gamemap2.OnShow();

for (int i = 0; i < gamemap2.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 3:

gamemap3.OnShow();

for (int i = 0; i < gamemap3.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 4:

gamemap4.OnShow();

for (int i = 0; i < gamemap4.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 5:

gamemap5.OnShow();

for (int i = 0; i < gamemap5.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 6:

gamemap6.OnShow();

for (int i = 0; i < gamemap6.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 7:

gamemap7.OnShow();

for (int i = 0; i < gamemap7.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 8:

gamemap8.OnShow();

for (int i = 0; i < gamemap8.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 9:

gamemap9.OnShow();

for (int i = 0; i < gamemap9.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 10:

gamemap10.OnShow();

for (int i = 0; i < gamemap10.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 11:

gamemap11.OnShow();

for (int i = 0; i < gamemap11.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

case 12:

gamemap12.OnShow();

for (int i = 0; i < gamemap12.GetChestNumber(); i++) {

box[i].OnShow();

}

man.OnShow();

break;

default:

break;

}

}

void CGameStateRun::SetObjectPosition(Map gameMap) {

int count = 0;

for (int i = 0; i < 15; i++) {

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

switch (gameMap.GetPosition(i \* 64, j \* 64)) {

case 3:

box[count].SetX1Y1(i \* 64, j \* 64);

count++;

break;

case 4:

man.SetX1Y1(i \* 64, j \* 64);

break;

default:

break;

}

}

}

}

void CGameStateRun::CheckManHitWall(Map gameMap) {

// 左上，右上，左下，右下

gameMap.hitWall(man.GetX1() + 5, man.GetY1() + 5, man.GetX2() - 5, man.GetY2() - 5);

bool\* hitDirection = gameMap.checkHitWall;

if (hitDirection[0] && hitDirection[2]) { // 左上、左下 撞 OK

man.SetX1Y1((man.GetX1() / 64 + 1) \* 64, man.GetY1());

}

if (hitDirection[1] && hitDirection[3]) { // 右上、右下 撞 OK

man.SetX2Y2(man.GetX2() / 64 \* 64, man.GetY2());

}

if (hitDirection[0] && hitDirection[1]) { // 左上、右上 撞 OK

man.SetX1Y1(man.GetX1(), (man.GetY1() / 64 + 1) \* 64);

}

if (hitDirection[2] && hitDirection[3]) { // 左下、右下 撞 OK

man.SetX2Y2(man.GetX2(), man.GetY2() / 64 \* 64);

}

}

void CGameStateRun::CheckManHitBox(Map gameMap) {

// (朝對的方向推 有碰到) 且 (箱子碰牆 或 碰箱子) => 人停下且設定位置

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

// 上、下、左、右

man.HitBox(&box[i]);

bool\* manHitBoxDirection = man.checkHitBox;

// 左上，右上，左下，右下

gameMap.hitWall(box[i].GetX1() + 5, box[i].GetY1() + 5, box[i].GetX2() - 5, box[i].GetY2() - 5);

bool\* boxHitWallDirection = gameMap.checkHitWall;

if (manHitBoxDirection[0]) { // 從上撞 OK

for (int j = 0; j < gameMap.GetChestNumber(); j++) {

if (i == j) {

continue;

}

if (boxHitWallDirection[2] || boxHitWallDirection[3] || box[i].HitBox(&box[j])) { // 該箱子碰牆 或 碰箱子

man.SetMovingDown(false);

box[i].SetX2Y2(box[i].GetX2(), box[i].GetY2() / 64 \* 64);

man.SetX2Y2(man.GetX2(), box[i].GetY1());

}

else { // 箱子不碰牆 或 不碰箱子

box[i].SetX1Y1(box[i].GetX1(), man.GetY2());

}

}

}

if (manHitBoxDirection[1]) { // 從下撞 裡面有一句 +15 那個是誤打誤撞 找時間了解

for (int j = 0; j < gameMap.GetChestNumber(); j++) {

if (i == j) {

continue;

}

if (boxHitWallDirection[0] || boxHitWallDirection[1] || box[i].HitBox(&box[j])) {

man.SetMovingUp(false);

box[i].SetX2Y2(box[i].GetX2(), ((box[i].GetY2() + 15) / 64) \* 64);

man.SetX1Y1(man.GetX1(), box[i].GetY2());

}

else { // 箱子不碰牆

box[i].SetX2Y2(box[i].GetX2(), man.GetY1());

}

}

}

if (manHitBoxDirection[2]) { // 從左撞 OK

for (int j = 0; j < gameMap.GetChestNumber(); j++) {

if (i == j) {

continue;

}

if (boxHitWallDirection[1] || boxHitWallDirection[3] || box[i].HitBox(&box[j])) {

man.SetMovingRight(false);

box[i].SetX2Y2(box[i].GetX2() / 64 \* 64, box[i].GetY2());

man.SetX2Y2(box[i].GetX1(), man.GetY2());

}

else { // 箱子不碰牆

box[i].SetX1Y1(man.GetX2(), box[i].GetY1());

}

}

}

if (manHitBoxDirection[3]) { // 從右撞

for (int j = 0; j < gameMap.GetChestNumber(); j++) {

if (i == j) {

continue;

}

if (boxHitWallDirection[0] || boxHitWallDirection[2] || box[i].HitBox(&box[j])) {

man.SetMovingLeft(false);

box[i].SetX1Y1((box[i].GetX1() + 15) / 64 \* 64, box[i].GetY1());

man.SetX1Y1(box[i].GetX2(), man.GetY1());

}

else { // 箱子不碰牆

box[i].SetX2Y2(man.GetX1(), box[i].GetY2());

}

}

}

}

}

bool CGameStateRun::CheckWin(Map gameMap) {

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

if (2 != gameMap.GetPosition(box[i].GetX1() + 32, box[i].GetY1() + 32)) {

return false;

}

}

// 這樣才不會卡鍵

man.SetMovingUp(false);

man.SetMovingDown(false);

man.SetMovingLeft(false);

man.SetMovingRight(false);

return true;

}

}