

組別:第 8 組

題目:大富翁

組員:簡旭均 學號 106590019

王博偉 學號 106590021

授課老師:陳偉凱

台北科大 資工系 2019年 物件導向程式設計實習報告 第 8 組 題目大富

1. **簡介**
2. 動機.......................................................1
3. 分工.......................................................2
4. **遊戲內容**
5. 遊戲說明...................................................2
6. 遊戲圖形...................................................3
7. 遊戲音效...................................................5
8. **程式設計**
9. 程式架構...................................................6
10. 程式類別...................................................7
11. 程式技術...................................................8
12. **結語**
13. 問題與解決方法.............................................8
14. 時間表.....................................................9
15. 貢獻比例..................................................10
16. 自我檢核表................................................10
17. 收穫......................................................10
18. 心得......................................................11
19. 對於本課程的建議.........................................11
20. **附錄**
21. Mygame.h
22. Mygame.cpp
23. Builging.h
24. Builging.cpp
25. House.h
26. House.cpp
27. Store.h
28. Store.cpp
29. GameOver.h
30. GameOver.cpp
31. Lottery.h
32. Lottery.cpp
33. OthersUI.h
34. OthersUI.cpp
35. Round.h
36. Round.cpp
37. Station.h
38. Station.cpp
39. Dice.h
40. Dice.cpp
41. Role.h
42. Role.cpp
43. BuyUI.h
44. ButUI.cpp
45. Card.h
46. Card.cpp
47. Integer.h. Integer.cpp
48. GameMap.h
49. GameMap.cpp
50. StartMenu.h
51. StartMenu.cpp

**一 簡介**

1.動機

由於蠻多人做格鬥遊戲的，因此想做一個相較不同風格的遊戲，因此選擇了大富翁，不僅可以回味童年的回憶又不失趣味。

2.分工表

A:簡旭均 B:王柏偉

|  |  |  |  |
| --- | --- | --- | --- |
|  | 程式 | 圖片素材 | 整體 |
| mygame | A:70%  B:30% | A:0%  B:0% | A:70%  B:30% |
| GameMap | A:100%  B:0% | A:0%  B:100% | A:80%  B:20% |
| Role | A:100%  B:0% | A:100%  B:0% | A:100%  B:0% |
| Dice | A:100%  B:0% | A:100%  B:0% | A:100%  B:0% |
| Station | A:0%  B:100% | A:0%  B:100% | A:0%  B:100% |
| Building | A:0%  B:100% | A:0%  B:100% | A:0%  B:100% |
| House | A:50%  B:50% | A:0%  B:100% | A:30%  B:70% |
| Store | A:100%  B:0% | A:0%  B:100% | A:80%  B:20% |
| Card | A:100%  B:0% | A:100%  B:0% | A:100%  B:0% |
| BuyUI | A:70%  B:30% | A:70%  B:30% | A:80%  B:20% |
| OthersUI | A:0%  B:100% | A:0%  B:100% | A:0%  B:100% |
| Lottery | A:0%  B:100% | A:0%  B:100% | A:0%  B:100% |
| Round | A:0%  B:100% | A:0%  B:100% | A:0%  B:100% |
| StarMenu | A:100%  B:0% | A:100%  B:0% | A:100%  B:0% |
| Integer | A:100%  B:0% | A:100%  B:0% | A:100%  B:0% |
| GameOver | A:0%  B:100% | A:0%  B:100% | A:0%  B:100% |

**二 遊戲介紹**

1.遊戲說明

遊戲規則：利用土地、卡片、樂透的方式來增加自己財富的機會，每當有角色的財產少於零之後即淘汰，存活到最後者即為贏家。

遊戲方式：

遊戲前：選擇遊戲的人數

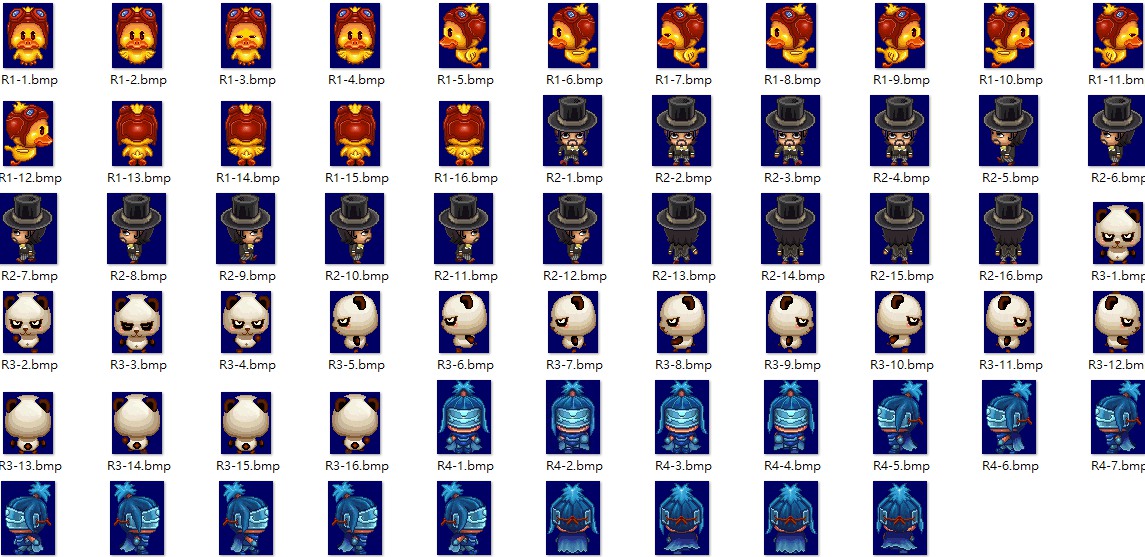
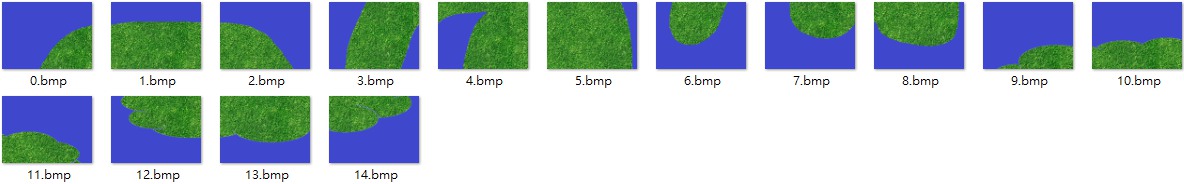
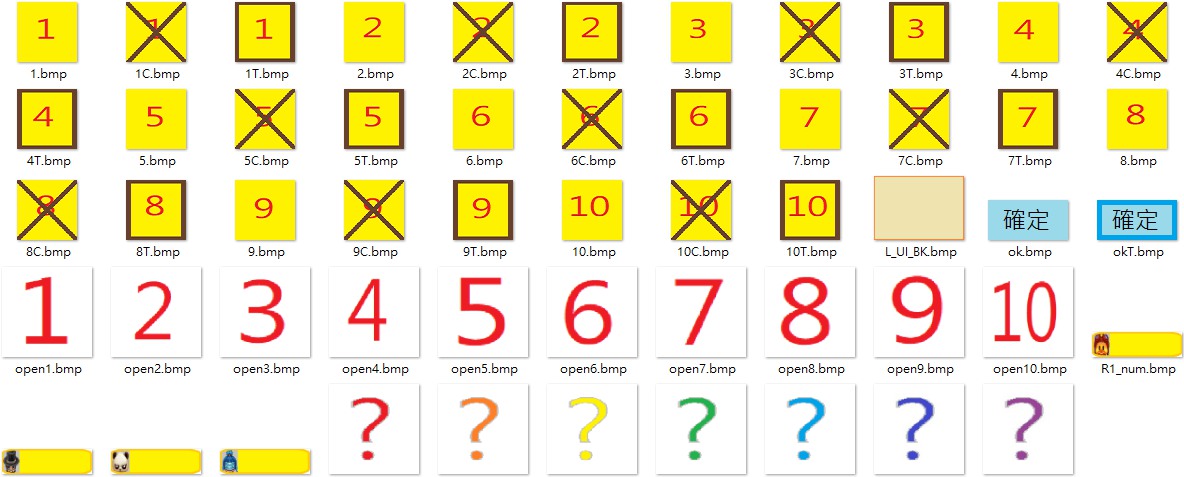
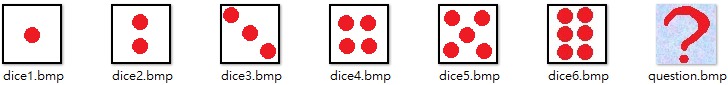
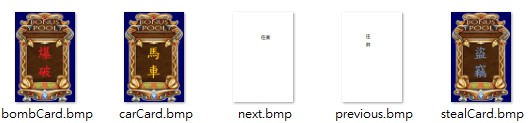
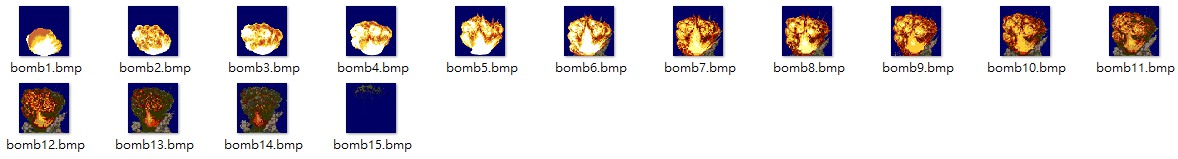
遊戲中：

1. 在擲骰子之前可以按Q開啟個人資料的欄位，其中包含了個人金額、房子的數量、樂透系統
2. 按下空白鍵開始擲骰子進行移動
3. 每個角色一輪只能選一個數字，選一個數字一千元，三個回合會開獎，獎金3000元。
4. 在商店可以花錢買卡片，卡片在當下即發揮其效果

遊戲結束:

當角色金錢餘額為小於等於0時，角色將被淘汰，直到剩 餘一位角色，結束遊戲

遊戲圖形：



遊戲音效：

背景音效:楓之谷鯨魚號

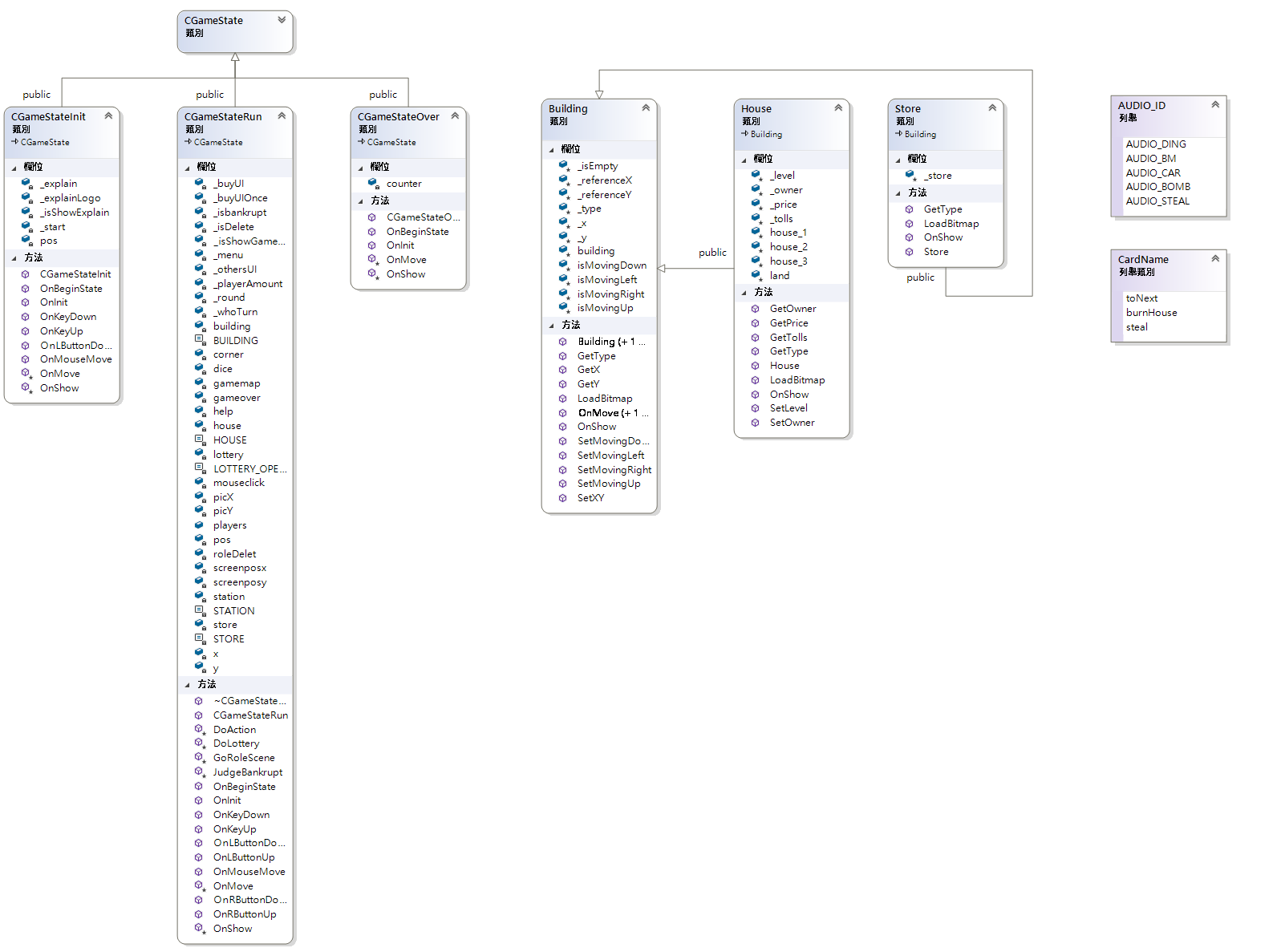
爆炸音效:爆炸聲

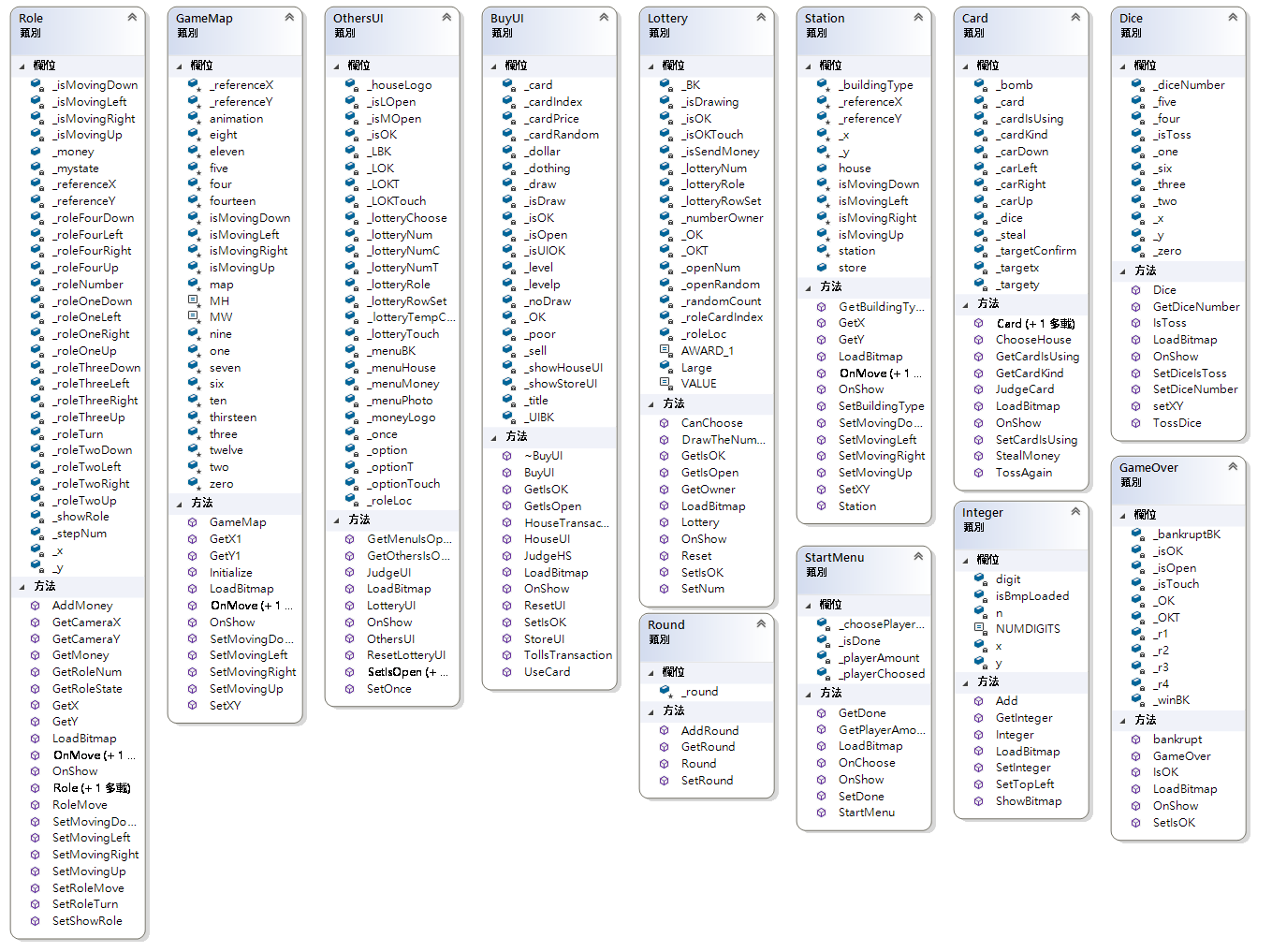
馬車音效:馬蹄聲

盜竊音效:金幣取得音效

**三 程式設計**

1. 程式架構:





1. 程式類別:

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| Mygame | 160 | 770 | 連結幾乎所有物件，做所有物件的媒介  並調配工作。 |
| GameMap | 39 | 205 | 地圖、上下左右移動。 |
| Role | 44 | 434 | 角色、儲存角色資料、並取得點數座標做移動。 |
| Dice | 21 | 88 | 擲出移動點數。 |
| Station | 36 | 128 | 行徑座標。 |
| Building | 33 | 140 | 建築物的base，包括移動及儲存資料。 |
| House | 26 | 102 | 房子、儲存房子資料、房子價格、過路費、房子狀態、擁有者。 |
| Store | 30 | 34 | 商店、儲存卡片價格。 |
| Card | 37 | 289 | 卡片有三種功能  馬車:一次擲3顆骰子。  盜竊:偷取別人金額。  爆破:毀滅他人的房屋。 |
| BuyUI | 34 | 320 | 購買介面、處理房屋及卡片的交易，並做過路費、設定房屋、使用卡片的動作 |
| OthersUI | 36 | 377 | 個人資料的介面，包含簽樂透數字的功能、房地產和財產的資料。 |
| Lottery | 31 | 201 | 公布樂透開獎、存取數字擁有人的資料。 |
| Round | 16 | 21 | 計數回合數。 |
| StarMenu | 20 | 63 | 選擇遊戲人數 2~4。 |
| Integer | 20 | 69 | 遊戲內所有數字的顯示。 |
| GameOver | 17 | 78 | 破產與遊戲結束的畫面。 |

1. 程式技術:

使用了記憶體位址vector:角色

指標陣列:行徑採點、

繼承:建築物

多型:建築物

**四 結語**

1. 問題與解決方法

問題1：在設建築物的時候會因為角色的行徑路線而跨越建築物，房子設在站點的下方會導致行走踏過房子。

解決：針對路線做對房子的位置做個別修改。

問題2:樂透的UI介面，角色要因為破產而消失在介面上，然而版面上會有缺口，因進行向前補齊。

解決:利用陣列進行位置的紀錄，在UI上的角色欄的座標應該要因為有人破產而向前遞補。

問題3:因為mygame是快速的run每個function，所以按下滑鼠時會因為程式的快速而讓許多的function也讀取到down的資料，導致同一個如果有連續兩個UI的同一個位置都有按鈕的話第二個UI的按鈕也會被按到。

解決:使用Sleep的方式。

問題4:對Sleep的瞭解沒有很透徹，自己認知的Sleep沒有發揮自己所想的效用，導致製作破產花太多時間。

解決:重複測試發現Sleep必須是mygame跑過1次之後才會延遲，所以如果有發生問題3時，必須確保mygame已經run一次之後才能保證Sleep是有效果的。

問題5:對於鍵盤的編碼了解有誤，查了資料也並不起效果。

解決:了解到進制的差別，並設中斷點即可知道欲知的編碼。

問題6:第一次使用記憶體位址vector，導致在摸索用法花了很多時間，因為與指標陣列用法有些不一樣

解決:因為網路程式範例很稀少，所以透過問同學及回去翻參考書解決

1. 時間表

|  |  |  |  |
| --- | --- | --- | --- |
| 週次 | 簡旭均(小時) | 王柏偉(小時) | 工作規劃 |
| 0 | 5 | 5 | 建立可移動地圖 |
| 1 | 9 | 5 | 加入人物、完成行徑踩點 |
| 2 | 11 | 7 | 人物移動、建築物設立 |
| 3 | 4 | 8 | 買地、存款管理、製造卡片 |
| 5 | 4 | 4 | 卡片購買介面 |
| 6 | 5 | 3 | 破產刪除角色、個人資料 |
| 7 | 8 | 13 | 個人資料 |
| 8 | 8 | 6 | 樂透開獎 |
| 9 | 2 | 2 | 增加個人資料內容 |
| 11 | 3 | 2 | 增加卡片、製作開頭 |
| 12 | 3 | 4 | 製作結尾與破產畫面 |
| 13 | 4 | 6 | 遊戲優化、報告製作 |

1. 貢獻比例

簡旭均：50%、王柏偉：50%

1. 自我檢核表

|  |  |  |  |
| --- | --- | --- | --- |
|  | 項目 | 完成否 | 無法完成的原因 |
| 1 | 解決memory leak |  |  |
| 2 | 自訂遊戲Icon |  |  |
| 3 | 全螢幕啟動 |  |  |
| 4 | 有About畫面 |  |  |
| 5 | 初始化畫面說明案件及滑鼠支用法與密技 |  |  |
| 6 | 上傳 setup/apk/source 檔 |  |  |
| 7 | Setup檔可正確執行 |  |  |
| 8 | 報告字型、點數、對齊、行 距、頁碼等格式正確 |  |  |
| 9 | 報告封面、側邊格式正確 |  |  |
| 10 | 報告附錄程式格式正確 |  |  |

1. 收穫

簡旭均：

第一次寫遊戲，是一個很好的經驗，除了了解一個遊戲的執行方式外，更了解了遊戲的架構，對於遊戲的運作有更進一步的了解，從這次的實習可以知道與班上某部分的同學有相當大的經驗差異，因此以後需要更積極的督促自己努力，還有第一次與他人合作寫一個遊戲，再次深刻體會溝通上的重要，以及規劃程式的進度及維持進度的重要性

王柏偉：因為程式跑動的關係必須要認真的思考每個function的影響，畢竟只要有稍微設計不良的地方就會影響整個程式，嚴重的話找出bug需要花費不少時間，有真正的對多型與繼承有所體悟，也因為這個框架的關係不好找bug,因此也使用了很多次的中斷點來進行，整體來說算是真正的感覺真正在寫物件導向的感覺。

1. 心得

簡旭均：

這次遊戲挑得有點保守，應該是可以再選擇更困難一點的程式，再選之前因為怕難度過高寫不出來，而選到了大富翁這種靜態的遊戲，因為過於靜態，做了很多也沒有好玩的感覺，很容易失去繼續寫下去的熱情，當然有部份也是這學期的功課很多，總覺得事做不完，做作業的動力消耗的很快，不過這學期還是稍微順利的度過了

王柏偉：比起往年只用namespace std的方式做許多紙上談兵、單純的io輸出，這次真正使用上了所謂的框架來進行OOP的實作，不僅運用到了上學期所學的OOP真正的精髓，也真正體會到了物件導向真正作出來的感覺，自己對於OOP的認知與體驗也是跨了一大步。

1. 對於本課程的建議

希望繼續保持該課程的授課理念，相信也有很多人在這個年紀對於OOP的體驗畫面永遠都還黑底白字，而物件導向程式實習對於學生是很重要的一環。

附錄

Mygame.h

#include "GameMap.h"

#include "Station.h"

#include "Role.h"

#include "Dice.h"

#include "StartMenu.h"

#include "Building.h"

#include "House.h"

#include "Store.h"

#include "Station.h"

#include "OthersUI.h"

#include "BuyUI.h"

#include "Card.h"

#include"Round.h"

#include "GameOver.h"

namespace game\_framework {

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

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

void OnKeyDown(UINT, UINT, UINT);

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

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

protected:

void OnMove();

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap \_start, \_explainLogo, \_explain,\_about,\_aboutLogo;

bool \_isShowExplain;

POINT pos;

};

class CGameStateRun : public CGameState {

public:

CGameStateRun(CGame \*g);

~CGameStateRun();

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit(); // 遊戲的初值及圖形設定

void OnKeyDown(UINT, UINT, UINT);

void OnKeyUp(UINT, UINT, UINT);

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

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

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

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

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

vector <Role\*> players;

protected:

void JudgeBankrupt();

void GoRoleScene();

void DoLottery();

void DoAction();

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

const int STATION, BUILDING, HOUSE,STORE, LOTTERY\_OPEN\_ROUND; // 站點的總數

unsigned int \_whoTurn , \_playerAmount;

CMovingBitmap help; // 說明圖

CMovingBitmap corner; // 角落圖

StartMenu \_menu;

GameMap gamemap;

Station \*station[81];

Building \*building[49];

House \*house[41];

Store \*store[8];

Lottery lottery;

OthersUI \_othersUI;

BuyUI \_buyUI;

Dice dice;

int screenposx, screenposy, roleDelet;

POINT pos;

Round \_round;

bool mouseclick,\_isShowGameOver, \_isbankrupt,\_buyUIOnce,\_isDelete;

int picX, picY ,x=0,y=0;

GameOver gameover;

};

class CGameStateOver : public CGameState {

public:

CGameStateOver(CGame \*g);

void OnBeginState(); // 設定每次重玩所需的變數

void OnInit();

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

};

}

Mygame.cpp

// mygame.cpp: 本檔案儲遊戲本身的class的implementation

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "mygame.h"

namespace game\_framework {

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

{

}

void CGameStateInit::OnInit()

{

ShowInitProgress(0); // 一開始的loading進度為0%

\_start.LoadBitmap(IDB\_INT);

\_explainLogo.LoadBitmap(IDB\_explainIcon);

\_explain.LoadBitmap(IDB\_explain);

\_about.LoadBitmap(IDB\_about);

\_aboutLogo.LoadBitmap(IDB\_aboutIcon);

}

void CGameStateInit::OnBeginState()

{

}

void CGameStateInit::OnMove()

{

}

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

{

}

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

{

const char KEY\_ESC = 27;

const char KEY\_SPACE = 0x20;

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)

{

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

}

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

{

pos = point;

}

void CGameStateInit::OnShow()

{

\_start.SetTopLeft(0, 0);

\_start.ShowBitmap();

\_explainLogo.SetTopLeft(250, 360);

\_explainLogo.ShowBitmap();

\_explain.SetTopLeft(150, 20);

\_about.SetTopLeft(200, 20);

\_aboutLogo.SetTopLeft(350, 360);

\_aboutLogo.ShowBitmap();

if (pos.x > 250 && pos.x < \_explainLogo.Width() + 250 && pos.y>360 && pos.y < \_explainLogo.Height() + 360)

\_explain.ShowBitmap();

if (pos.x > 350 && pos.x < \_aboutLogo.Width() + 350 && pos.y>360 && pos.y < \_aboutLogo.Height() + 360)

\_about.ShowBitmap();

}

CGameStateOver::CGameStateOver(CGame \*g)

: CGameState(g)

{

}

void CGameStateOver::OnMove()

{

counter--;

if (counter < 0) {

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

}

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 5; // 5 seconds

}

void CGameStateOver::OnInit()

{

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

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

ShowInitProgress(100);

}

void CGameStateOver::OnShow()

{

}

CGameStateRun::CGameStateRun(CGame \*g)

: CGameState(g), STATION(81), \_whoTurn(0), \_playerAmount(4), mouseclick(false), BUILDING(49), HOUSE(41), STORE(8), LOTTERY\_OPEN\_ROUND(3), \_isShowGameOver(false), \_isbankrupt(false), \_buyUIOnce(false), \_isDelete(false)

{

Role\* role;

for (unsigned int i = 0; i < \_playerAmount;i++) {

role = new Role(i);

players.push\_back(role);

}

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

station[i] = new Station;

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

house[i] = new House;

}

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

store[i] = new Store;

}

int houseNum=0, storeNum=0;

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

{

if (i == 9 || i == 12 || i == 23 || i == 21 || i == 31 || i == 36 || i == 2 || i == 4)

{

building[i] = store[storeNum++];

}

else

{

building[i] = house[houseNum++];

}

}

}

CGameStateRun::~CGameStateRun()

{

\_buyUI.~BuyUI();

players.clear();

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

delete [] building[i];

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

delete [] station[i];

}

void CGameStateRun::OnBeginState()

{

const int BALL\_GAP = 90;

const int BALL\_XY\_OFFSET = 45;

const int BALL\_PER\_ROW = 7;

const int HITS\_LEFT = 10;

const int HITS\_LEFT\_X = 590;

const int HITS\_LEFT\_Y = 0;

const int BACKGROUND\_X = 60;

const int ANIMATION\_SPEED = 15;

//----------at map[0]-----------------------------------------at map[1] -------------------------------at map[2]

station[0]->SetXY(560, 280);/\*-----------------\*/station[76]->SetXY(1128, 292);/\*-----------------\*/station[58]->SetXY(1592, 376);

building[46]->SetXY(1193, 292);/\*------------\*/building[36]->SetXY(1527, 376);

station[1]->SetXY(496, 332);/\*-----------------\*/station[77]->SetXY(1040, 200);/\*-----------------\*/station[59]->SetXY(1480, 276);

building[0]->SetXY(429, 332);/\*-----------------------------------------------------------------\*/building[37]->SetXY(1490, 241);

station[2]->SetXY(552, 416);/\*-----------------\*/station[78]->SetXY(944, 272);/\*------------------\*/station[60]->SetXY(1388, 228);

building[1]->SetXY(485, 416);

station[80]->SetXY(692, 280);/\*----------------\*/station[61]->SetXY(1428, 412);

building[48]->SetXY(692, 245);/\*---------\*/building[38]->SetXY(1363, 412);

//----------at map[3]-----------------------------------------at map[4] -------------------------------at map[5]

station[3]->SetXY(560, 560);/\*-----------------\*/station[72]->SetXY(1156, 872);/\*-----------------\*/station[55]->SetXY(1740, 864);

building[43]->SetXY(1221, 872); building[34]->SetXY(1805, 864);

station[4]->SetXY(436, 572);/\*-----------------\*/station[73]->SetXY(904, 776);/\*-----------------\*/station[56]->SetXY(1692, 660);

building[2]->SetXY(436, 537);/\*----------------------------------------------------------------\*/building[35]->SetXY(1757, 660);

station[5]->SetXY(344, 552);/\*-----------------\*/station[74]->SetXY(992, 676);/\*-----------------\*/station[57]->SetXY(1692, 660);

building[3]->SetXY(344, 517);/\*----------------\*/building[44]->SetXY(1059, 677);

station[6]->SetXY(284, 620);/\*-----------------\*/station[75]->SetXY(1064, 528);/\*-----------------\*/station[62]->SetXY(1324, 636);

building[4]->SetXY(219, 620);/\*----------------\*/building[45]->SetXY(1130, 528);/\*---------------\*/building[39]->SetXY(1259, 636);

station[7]->SetXY(252, 680);/\*-----------------\*/station[79]->SetXY(852, 500);/\*------------------\*/station[63]->SetXY(1428, 704);

building[5]->SetXY(185, 680);/\*----------------\*/building[47]->SetXY(789, 500);/\*-----------------\*/building[40]->SetXY(1493, 680);

station[8]->SetXY(400, 760);/\*--------------------------------------------------------------------\*/station[64]->SetXY(1312, 768);

building[6]->SetXY(400, 725);/\*---------------------------------------------------------------\*/station[65]->SetXY(1416, 840);

station[9]->SetXY(456, 872);/\*--------------------------------------------------------------------\*/station[66]->SetXY(1512, 908);

//----------at map[6]-----------------------------------------at map[7] -------------------------------at map[8]

station[10]->SetXY(224, 960);/\*-----------------\*/station[68]->SetXY(1064, 1108);/\*-----------------\*/station[50]->SetXY(1684, 1364);

building[41]->SetXY(1064, 1073);

station[11]->SetXY(368, 1000);/\*----------------\*/station[69]->SetXY(940, 1120);/\*-----------------\*/station[51]->SetXY(1516, 1324);

building[42]->SetXY(875, 1120);

station[12]->SetXY(380, 1080);/\*----------------\*/station[70]->SetXY(896, 1020);/\*-----------------\*/station[52]->SetXY(1556, 1220);

station[13]->SetXY(252, 1136);/\*----------------\*/station[71]->SetXY(1040, 980);/\*-------------------\*/building[31]->SetXY(1620, 1220);

/\*-------------------------------------------------------------------------------------------------\*/station[53]->SetXY(1668, 1140);

station[14]->SetXY(212, 1236);/\*---------------------------------------------------------------------\*/building[32]->SetXY(1732, 1140);

/\*-------------------------------------------------------------------------------------------------\*/station[54]->SetXY(1724, 988);

station[15]->SetXY(168, 1352);/\*---------------------------------------------------------------------\*/building[33]->SetXY(1789, 988);

station[67]->SetXY(1312, 1012);

building[34]->SetXY(1312, 977);

//----------at map[9]-----------------------------------------at map[10] -------------------------------at map[11]

station[16]->SetXY(152, 1508);/\*-----------------\*/station[31]->SetXY(744, 1840);/\*-----------------\*/station[42]->SetXY(1664, 1852);

building[18]->SetXY(744, 1805);/\*---------------\*/building[27]->SetXY(1729, 1822);

station[17]->SetXY(172, 1592);/\*-----------------\*/station[32]->SetXY(868, 1788);/\*-----------------\*/station[43]->SetXY(1500, 1852);

building[19]->SetXY(868, 1753);/\*----------------\*/building[28]->SetXY(1500, 1817);

station[18]->SetXY(128, 1696);/\*-----------------\*/station[33]->SetXY(1020, 1784);/\*-----------------\*/station[44]->SetXY(1344, 1784);

building[20]->SetXY(1020, 1749);/\*--------------\*/building[29]->SetXY(1279, 1784);

station[19]->SetXY(128, 1824);/\*-----------------\*/station[34]->SetXY(1156, 1760);/\*-----------------\*/station[45]->SetXY(1436, 1712);

building[21]->SetXY(1210, 1725);/\*--------------\*/building[30]->SetXY(1501, 1712);

station[20]->SetXY(320, 1832);/\*-----------------\*/station[35]->SetXY(1108, 1844);/\*-----------------\*/station[46]->SetXY(1352, 1632);

building[7]->SetXY(320, 1798);/\*----------------\*/building[22]->SetXY(1173, 1844);

station[21]->SetXY(424, 1832);/\*---------------------------------------------------------------------\*/station[47]->SetXY(1460, 1580);

building[8]->SetXY(410, 1798);

station[22]->SetXY(532, 1856);/\*---------------------------------------------------------------------\*/station[48]->SetXY(1576, 1560);

building[9]->SetXY(597, 1856);/\*-----------------------------------------------------------------\*/station[49]->SetXY(1788, 1500);

//----------at map[12]-----------------------------------------at map[13] -------------------------------at map[14]

station[23]->SetXY(504, 1968);/\*-----------------\*/station[28]->SetXY(704, 2112);/\*-----------------\*/station[37]->SetXY(1320, 2016);

building[10]->SetXY(569, 1968);/\*--------------\*/building[15]->SetXY(769, 2112);/\*-----------------\*/building[24]->SetXY(1320, 1981);

station[24]->SetXY(368, 2000);/\*-----------------\*/station[29]->SetXY(704, 2040);/\*-----------------\*/station[38]->SetXY(1412, 2092);

building[11]->SetXY(368, 1965);/\*--------------\*/building[16]->SetXY(769, 2040);

station[25]->SetXY(272, 2072);/\*-----------------\*/station[30]->SetXY(704, 1944);/\*-----------------\*/station[39]->SetXY(1496, 2016);

building[12]->SetXY(207, 2072);/\*-----------------\*/building[17]->SetXY(769, 1944);/\*--------------\*/building[25]->SetXY(1496, 1981);

station[26]->SetXY(428, 2108);/\*-----------------\*/station[36]->SetXY(1176, 1960);/\*----------------\*/station[40]->SetXY(1640, 1964);

building[13]->SetXY(428, 2073);/\*--------------\*/building[23]->SetXY(1111, 1960);

station[27]->SetXY(544, 2108);/\*--------------------------------------------------------------------\*/station[41]->SetXY(1736, 1960);

building[14]->SetXY(544, 2073);/\*-----------------------------------------------------------------\*/building[26]->SetXY(1801, 1960);

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

{

int stationX = station[i]->GetX(), stationY = station[i]->GetY();

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

{

if (stationX + 70 > building[j]->GetX() && stationX - 70 < building[j]->GetX() &&

stationY + 10 > building[j]->GetY() && stationY - 40 < building[j]->GetY() && building[j]->GetType() == "House")

{

station[i]->house = (House \*)building[j];

station[i]->store = nullptr;

station[i]->SetBuildingType("House");

break;

}

else if (stationX + 70 > building[j]->GetX() && stationX - 70 < building[j]->GetX() &&

stationY + 10 > building[j]->GetY() && stationY - 40 < building[j]->GetY() && building[j]->GetType() == "Store")

{

station[i]->house = nullptr;

station[i]->store = (Store \*)building[j];

station[i]->SetBuildingType("Store");

}

}

}

gamemap.Initialize();

help.SetTopLeft(0, SIZE\_Y - help.Height()); // 設定說明圖的起始座標

CAudio::Instance()->Play(AUDIO\_DING, false); // 撥放 WAVE

CAudio::Instance()->Play(AUDIO\_BM, true); // 撥放 MIDI

}

void CGameStateRun::GoRoleScene()

{

int camerax, cameray;

camerax = players[\_whoTurn]->GetCameraX();

cameray = players[\_whoTurn]->GetCameraY();

gamemap.OnMove(camerax, cameray);

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

station[i]->OnMove(camerax, cameray);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->OnMove(camerax, cameray);

}

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

building[i]->OnMove(camerax, cameray);

}

void CGameStateRun::DoAction()

{

if (!\_buyUIOnce)

{

JudgeBankrupt();

\_buyUI.JudgeHS(pos.x, pos.y, mouseclick, players, players[\_whoTurn], station, station[players[\_whoTurn]->GetRoleState()], &gamemap, building, \_isbankrupt);//打開UI 如果沒建築將不做事

if (\_buyUI.GetIsOK()) {

for (vector<Role\*>::iterator i = players.begin(); i != players.end(); i++) { //檢查角色金額是否小於0 刪除 剩一人刪除角色及reset房子

if ((\*i)->GetMoney() <= 0)

{

\_isbankrupt = true;

}

}

if (!\_isbankrupt)

{

lottery.SetIsOK(false);

\_othersUI.SetOnce(false);

\_round.AddRound(\_whoTurn, \_playerAmount);

\_whoTurn++;

dice.SetDiceIsToss(true);

}

if (\_whoTurn == \_playerAmount) {

\_whoTurn = 0;

}

\_buyUI.SetIsOK(false);//重設UI狀態

\_buyUIOnce = true;

GoRoleScene();

}

}

}

void CGameStateRun::DoLottery()

{

if (\_round.GetRound() % LOTTERY\_OPEN\_ROUND == 0 && \_round.GetRound() != 0 && !lottery.GetIsOK())//樂透判斷是否開獎

{

lottery.DrawTheNumber(pos.x, pos.y, mouseclick, players);

if (lottery.GetIsOK())

{

\_othersUI.ResetLotteryUI();

\_round.SetRound(0);

}

}

}

void CGameStateRun::JudgeBankrupt()

{

if (players.size() < 2) {

if (gameover.IsOK())

{

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

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

}

else

gameover.bankrupt(players[0], pos.x, pos.y, mouseclick);

}

else {

for (vector<Role\*>::iterator i = players.begin(); i != players.end(); i++) { //檢查角色金額是否小於0 刪除 剩一人刪除角色及reset房子

if ((\*i)->GetMoney() <= 0)

{

if(!\_isDelete)

\_isbankrupt = true;

roleDelet = (\*i)->GetRoleNum();

gameover.bankrupt(\*i, pos.x, pos.y, mouseclick);

if (gameover.IsOK())

{

\_isDelete = true;

\_isbankrupt = false;

gameover.SetIsOK(false);

\_othersUI.SetOnce(false);

\_othersUI.SetIsOpen(false);

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

if (station[j]->GetBuildingType() == "House") {

if (station[j]->house->GetOwner() == (\*i)->GetRoleNum()) {

station[j]->house->SetLevel(-1);

station[j]->house->SetOwner(4);

}

}

}

if (\_whoTurn == \_playerAmount - 1) {//注意whoturn不能超過playerAmount

\_whoTurn--;

}

i = players.erase(i);

\_playerAmount -= 1;

GoRoleScene();

break;

}

}

}

}

if (\_isDelete){ //角色以刪除

lottery.SetIsOK(false);

\_othersUI.SetOnce(false);

\_round.AddRound(\_whoTurn, \_playerAmount);

\_isDelete = false;

dice.SetDiceIsToss(true);

}

}

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

{

if (!\_menu.GetDone()) {

\_menu.OnChoose(pos.x, pos.y, mouseclick);

}

if (\_menu.GetDone()) {

JudgeBankrupt();

if (!dice.IsToss()) {

if (dice.GetDiceNumber() != 0) {

\_buyUIOnce = false;

GoRoleScene();

}

dice.SetDiceNumber(players[\_whoTurn]->RoleMove(dice.GetDiceNumber(), station));

if (dice.GetDiceNumber() == 0) {//到達最終位置

DoAction();

}

}

else {

DoLottery();

if (\_othersUI.GetMenuIsOpen())

\_othersUI.JudgeUI(pos.x, pos.y, mouseclick, players, players[\_whoTurn], station, lottery);

gamemap.OnMove();

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

station[i]->OnMove();

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->OnMove();

}

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

building[i]->OnMove();

}

}

}

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

{

ShowInitProgress(33);

gamemap.LoadBitmap();

\_menu.LoadBitmap();

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->LoadBitmap();

}

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

station[i]->LoadBitmap();

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

building[i]->LoadBitmap();

dice.LoadBitmap();

\_buyUI.LoadBitmap();

\_othersUI.LoadBitmap();

lottery.LoadBitmap();

gameover.LoadBitmap();

ShowInitProgress(50);

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

help.LoadBitmap(IDB\_HELP, RGB(255, 255, 255)); // 載入說明的圖形

corner.LoadBitmap(IDB\_CORNER); // 載入角落圖形

CAudio::Instance()->Load(AUDIO\_DING, "sounds\\ding.wav"); // 載入編號0的聲音ding.wav

CAudio::Instance()->Load(AUDIO\_BM, "sounds\\backmusic.mp3"); // 載入編號2的聲音ntut.mid

}

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\_SPACE = 0x20;

const char KEY\_Q = 81;

const char KEY1 = 49, KEY2 = 50, KEY3 = 51, KEY4 = 52, KEY5 = 53, KEY6 = 54;

bool tossdice = false;

int diceNum = 0;

if (nChar == KEY1 || nChar == KEY2 || nChar == KEY3 || nChar == KEY4 || nChar == KEY5 || nChar == KEY6 || nChar == KEY\_SPACE) {

tossdice = true;

}

if (nChar == KEY\_LEFT ) {

gamemap.SetMovingLeft(true);

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

station[i]->SetMovingLeft(true);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->SetMovingLeft(true);

}

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

building[i]->SetMovingLeft(true);

}

if (nChar == KEY\_RIGHT) {

gamemap.SetMovingRight(true);

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

station[i]->SetMovingRight(true);

for (unsigned int i = 0 ;i < \_playerAmount; i++) {

players[i]->SetMovingRight(true);

}

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

building[i]->SetMovingRight(true);

}

if (nChar == KEY\_UP ) {

gamemap.SetMovingUp(true);

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

station[i]->SetMovingUp(true);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->SetMovingUp(true);

}

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

building[i]->SetMovingUp(true);

}

if (nChar == KEY\_DOWN) {

gamemap.SetMovingDown(true);

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

station[i]->SetMovingDown(true);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->SetMovingDown(true);

}

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

building[i]->SetMovingDown(true);

}

if (tossdice && dice.IsToss() && \_menu.GetDone() && !\_othersUI.GetMenuIsOpen() && !\_othersUI.GetOthersIsOpen() && !lottery.GetIsOpen())

{

switch (nChar) {

case KEY1:

diceNum = 1;

break;

case KEY2:

diceNum = 2;

break;

case KEY3:

diceNum = 3;

break;

case KEY4:

diceNum = 4;

break;

case KEY5:

diceNum = 5;

break;

case KEY6:

diceNum = 6;

break;

case KEY\_SPACE:

diceNum = 0;

break;

}

dice.TossDice(diceNum);

}

if (nChar == KEY\_Q && \_menu.GetDone() && dice.IsToss() && !\_othersUI.GetOthersIsOpen() && !lottery.GetIsOpen())

{

\_othersUI.SetIsOpen();

}

}

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

{

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

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

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

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

const char KEY\_SPACE = 0x20;

const char KEY1 = 49, KEY2 = 50, KEY3 = 51, KEY4 = 52, KEY5 = 53, KEY6 = 54;

bool tossdice = false;

if (nChar == KEY1 || nChar == KEY2 || nChar == KEY3 || nChar == KEY4 || nChar == KEY5 || nChar == KEY6 || nChar == KEY\_SPACE) {

tossdice = true;

}

Role\* rmrole;

if (nChar == KEY\_LEFT)

{

gamemap.SetMovingLeft(false);

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

station[i]->SetMovingLeft(false);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->SetMovingLeft(false);

}

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

building[i]->SetMovingLeft(false);

}

if (nChar == KEY\_RIGHT)

{

gamemap.SetMovingRight(false);

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

station[i]->SetMovingRight(false);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->SetMovingRight(false);

}

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

building[i]->SetMovingRight(false);

}

if (nChar == KEY\_UP)

{

gamemap.SetMovingUp(false);

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

station[i]->SetMovingUp(false);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->SetMovingUp(false);

}

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

building[i]->SetMovingUp(false);

}

if (nChar == KEY\_DOWN)

{

gamemap.SetMovingDown(false);

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

station[i]->SetMovingDown(false);

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->SetMovingDown(false);

}

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

building[i]->SetMovingDown(false);

}

if (tossdice && !\_menu.GetDone()) {

\_menu.SetDone(true);

\_playerAmount = \_menu.GetPlayerAmount();

for (int i = 0; i != 4 - \_playerAmount; i++) {

rmrole=players.back();

players.pop\_back();

delete rmrole;

}

}

}

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

{

mouseclick = true;

}

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

{

mouseclick = false;

}

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

{

pos = point;

}

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

{

}

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

{

}

void CGameStateRun::OnShow()

{

help.ShowBitmap();

if (!\_menu.GetDone()) {

\_menu.OnShow();

}

if (\_menu.GetDone()) {

gamemap.OnShow();

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

station[i]->OnShow();

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

building[i]->OnShow();

}

for (unsigned int i = 0; i < \_playerAmount; i++) {

players[i]->OnShow(station);

}

dice.OnShow();

\_buyUI.OnShow(players, players[\_whoTurn], station, station[players[\_whoTurn]->GetRoleState()]);

\_othersUI.OnShow(players, players[\_whoTurn], lottery);

lottery.OnShow(players, players[\_whoTurn]);

for (vector<Role \*>::iterator i = players.begin(); i != players.end(); i++) {

if ((\*i)->GetRoleNum() == roleDelet) {

gameover.OnShow((\*i), players.size());

}

else if (players.size() < 2) {

gameover.OnShow(players[0], players.size());

}

}

}

corner.SetTopLeft(0, 0);

corner.ShowBitmap();

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

corner.ShowBitmap();

}

}

Builging.h

#pragma once

namespace game\_framework

{

class Building

{

public:

Building(string);

Building();

int GetX();

int GetY();

void SetXY(int nx, int ny); //設定房子的位置

virtual void LoadBitmap();

virtual void OnShow();

void OnMove();

void OnMove(int camerax, int cameray);

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

virtual string GetType(); // 得取該建築物的類型

protected:

string \_type; // 該建築物的類型

int \_x, \_y;

bool \_isEmpty;

int \_referenceX, \_referenceY; // 座標參考點

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

CMovingBitmap building;

};

}

Builging.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include"Building.h"

namespace game\_framework

{

Building::Building() :\_referenceX(0), \_referenceY(0), \_isEmpty(false)

{

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

\_type = "Building";

}

Building::Building(string type) : \_referenceX(0), \_referenceY(0), \_isEmpty(false)

{

\_type = type;

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

}

void Building::OnShow()

{

if (\_x<639 && \_x>-639 && \_y<479 && \_y>-479)

{

building.SetTopLeft(\_x, \_y);

building.ShowBitmap();

}

}

int Building::GetX()

{

return \_x;

}

int Building::GetY()

{

return \_y;

}

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

{

\_x = nx; \_y = ny;

}

void Building::LoadBitmap()

{

}

string Building::GetType()

{

return \_type;

}

void Building::OnMove()

{

const int STEP\_SIZE = 10;

if (\_referenceX + STEP\_SIZE <= 0) {

if (isMovingLeft)

{

\_x += STEP\_SIZE;

\_referenceX += STEP\_SIZE;

}

}

if (\_referenceX - STEP\_SIZE >= -1280) {

if (isMovingRight)

{

\_x -= STEP\_SIZE;

\_referenceX -= STEP\_SIZE;

}

}

if (\_referenceY + STEP\_SIZE <= 0) {

if (isMovingUp)

{

\_y += STEP\_SIZE;

\_referenceY += STEP\_SIZE;

}

}

if (\_referenceY - STEP\_SIZE >= -1920) {

if (isMovingDown)

{

\_y -= STEP\_SIZE;

\_referenceY -= STEP\_SIZE;

}

}

}

void Building::OnMove(int camerax, int cameray)

{

if (\_referenceX + camerax <= 0 && \_referenceX + camerax >= -1280) {

\_referenceX += camerax;

\_x += camerax;

}

else if (camerax > 0) {

\_x = \_x - \_referenceX;

\_referenceX = 0;

\_x = \_referenceX + \_x;

}

else if (camerax < 0) {

\_x = \_x - \_referenceX;

\_referenceX = -1280;

\_x = \_referenceX + \_x;

}

if (\_referenceY + cameray <= 0 && \_referenceY + cameray >= -1920) {

\_referenceY += cameray;

\_y += cameray;

}

else if (cameray > 0) {

\_y = \_y - \_referenceY;

\_referenceY = 0;

\_y = \_referenceY + \_y;

}

else if (cameray < 0) {

\_y = \_y - \_referenceY;

\_referenceY = -1920;

\_y = \_referenceY + \_y;

}

}

void Building::SetMovingDown(bool flag)

{

isMovingDown = flag;

}

void Building::SetMovingLeft(bool flag)

{

isMovingLeft = flag;

}

void Building::SetMovingRight(bool flag)

{

isMovingRight = flag;

}

void Building::SetMovingUp(bool flag)

{

isMovingUp = flag;

}

}

House.h

#pragma once

#include"Building.h"

namespace game\_framework

{

class House :public Building

{

public:

House();

void SetLevel(int level);

void SetOwner(unsigned int);

unsigned int GetOwner();

int GetPrice(int level);

int GetTolls();

virtual string GetType();

virtual void OnShow();

virtual void LoadBitmap();

protected:

unsigned int \_owner;//--擁有者,owner=4 代表沒人

int \_level;//-----------土地等級 ,0為空地

int \_price[4];

int \_tolls[3];

CMovingBitmap land, house\_1, house\_2, house\_3;

};

}

House.cpp

#pragma once

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include"House.h"

namespace game\_framework

{

House::House() : Building("House"), \_level(0) , \_owner(4) {

\_price[0]= 1000;

\_price[1] = 3000;

\_price[2] = 6000;

\_price[3] = 1000;

\_tolls[0] = 400;

\_tolls[1] = 900;

\_tolls[2] = 1800;

}

unsigned int House::GetOwner()

{

return \_owner;

}

int House::GetPrice(int level) {

if (level == 1){

return \_price[0];

}

else if (level == 2) {

return \_price[1];

}

else if (level == 3) {

return \_price[2];

}

else if (level == -1) {

return \_price[3];

}

return 0;

}

int House::GetTolls() {

if (\_level == 1) {

return \_tolls[0];

}

else if (\_level == 2) {

return \_tolls[1];

}

else if (\_level == 3) {

return \_tolls[2];

}

return 0;

}

void House::SetLevel(int level) {

if (level == -1) {

\_level = 0;

}

else {

\_level = level;

}

}

void House::SetOwner(unsigned int who)

{

\_owner = who;

}

string House::GetType()

{

return \_type;

}

void House::LoadBitmap()

{

land.LoadBitmap(IDB\_LAND, RGB(255, 255, 255));

house\_1.LoadBitmap(IDB\_HOUSE\_1, RGB(255, 255, 255));

house\_2.LoadBitmap(IDB\_HOUSE\_2, RGB(255, 255, 255));

house\_3.LoadBitmap(IDB\_HOUSE\_3, RGB(255, 255, 255));

}

void House::OnShow()

{

land.SetTopLeft(\_x, \_y);

house\_1.SetTopLeft(\_x, \_y - 10);

house\_2.SetTopLeft(\_x, \_y - 20);

house\_3.SetTopLeft(\_x, \_y - 20);

if (\_x<690 && \_x>-50 && \_y<530 && \_y>-50)

{

if (\_level == 0)

{

land.ShowBitmap();

}

else if (\_level == 1)

{

house\_1.ShowBitmap();

}

else if (\_level == 2)

{

house\_2.ShowBitmap();

}

else if (\_level == 3)

{

house\_3.ShowBitmap();

}

}

}

}

Station.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Station.h"

namespace game\_framework

{

Station::Station():\_referenceX(0), \_referenceY(0), \_buildingType("none") { isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false; }

int Station::GetX() {

return \_x;

}

int Station::GetY(){

return \_y;

}

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

{

\_x = nx; \_y = ny;

}

string Station::GetBuildingType()

{

return \_buildingType;

}

void Station::SetBuildingType(string buildingType) {

\_buildingType = buildingType;

}

void Station::OnMove()

{

const int STEP\_SIZE = 10;

if (\_referenceX + STEP\_SIZE <= 0) {

if (isMovingLeft)

{

\_x += STEP\_SIZE;

\_referenceX += STEP\_SIZE;

}

}

if (\_referenceX - STEP\_SIZE >= -1280) {

if (isMovingRight)

{

\_x -= STEP\_SIZE;

\_referenceX -= STEP\_SIZE;

}

}

if (\_referenceY + STEP\_SIZE <= 0) {

if (isMovingUp)

{

\_y += STEP\_SIZE;

\_referenceY += STEP\_SIZE;

}

}

if (\_referenceY - STEP\_SIZE >= -1920) {

if (isMovingDown)

{

\_y -= STEP\_SIZE;

\_referenceY -= STEP\_SIZE;

}

}

}

void Station::OnMove(int camerax, int cameray) {

if (\_referenceX + camerax <= 0 && \_referenceX + camerax >= -1280) {

\_referenceX += camerax;

\_x += camerax;

}

else if (camerax > 0) {

\_x = \_x - \_referenceX;

\_referenceX = 0;

\_x = \_referenceX + \_x;

}

else if (camerax < 0) {

\_x = \_x - \_referenceX;

\_referenceX = -1280;

\_x = \_referenceX + \_x;

}

if (\_referenceY + cameray <= 0 && \_referenceY + cameray >= -1920) {

\_referenceY += cameray;

\_y += cameray;

}

else if (cameray > 0) {

\_y = \_y - \_referenceY;

\_referenceY = 0;

\_y = \_referenceY + \_y;

}

else if (cameray < 0) {

\_y = \_y - \_referenceY;

\_referenceY = -1920;

\_y = \_referenceY + \_y;

}

}

void Station::SetMovingDown(bool flag)

{

isMovingDown = flag;

}

void Station::SetMovingLeft(bool flag)

{

isMovingLeft = flag;

}

void Station::SetMovingRight(bool flag)

{

isMovingRight = flag;

}

void Station::SetMovingUp(bool flag)

{

isMovingUp = flag;

}

void Station::LoadBitmap()

{

station.LoadBitmap(IDB\_dot, RGB(255, 255, 255));

}

void Station::OnShow()

{

if (\_x<690 && \_x>-50 && \_y<530 && \_y>-50)

{

station.SetTopLeft(\_x, \_y);

station.ShowBitmap();

}

}

}

Store.h

#pragma once

#include "Building.h"

namespace game\_framework

{

class Store :public Building

{

public:

Store();

virtual string GetType();

virtual void OnShow();

virtual void LoadBitmap();

protected:

CMovingBitmap \_store;

};

}

Store.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Store.h"

namespace game\_framework

{

Store::Store() : Building("Store") {

}

string Store::GetType()

{

return \_type;

}

void Store::LoadBitmap()

{

\_store.LoadBitmap(IDB\_store, RGB(0, 0, 100));

}

void Store::OnShow()

{

\_store.SetTopLeft(\_x, \_y - 20);

if (\_x<690 && \_x>-50 && \_y<530 && \_y>-50)

{

\_store.ShowBitmap();

}

}

}

Gameover.h

#pragma once

#include"Role.h"

namespace game\_framework

{

class GameOver {

public:

GameOver();

void bankrupt(Role \*role,int x,int y,bool mouseclick); //-破產和遊戲結束的畫面

void LoadBitmap();

void OnShow(Role \*role, int size);

void SetIsOK(bool);//------------------------------------ 設定確定的值

bool IsOK();//------------------------------------------- 是否已按確定

private:

CMovingBitmap \_winBK,\_bankruptBK, \_OK, \_OKT,\_r1, \_r2, \_r3, \_r4 ;

bool \_isOK,\_isOpen,\_isTouch;//----------------------------是否已確認、打開

};

}

Gameover.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "GameOver.h"

namespace game\_framework {

GameOver::GameOver()

{

\_isOpen = \_isOK = \_isTouch = false;

}

bool GameOver::IsOK()

{

return \_isOK;

}

void GameOver::SetIsOK(bool flag)

{

\_isOK = flag;

}

void GameOver::bankrupt(Role \*role, int x, int y,bool mouseclick)

{

\_isOpen = true;

if (x > 300 && x < \_OK.Width() + 300 && y>350 && y < 350 + \_OK.Height())

{

\_isTouch = true;

if (mouseclick)

{

\_isOK = true;

\_isOpen = false;

Sleep(150);

}

}

else

\_isTouch = false;

}

void GameOver::LoadBitmap()

{

\_winBK.LoadBitmap(IDB\_win);

\_bankruptBK.LoadBitmap(IDB\_bankrupt);

\_OK.LoadBitmap(IDB\_L\_OK);

\_OKT.LoadBitmap(IDB\_L\_OK\_T);

\_r1.LoadBitmap(IDB\_R101);

\_r2.LoadBitmap(IDB\_R201);

\_r3.LoadBitmap(IDB\_R301);

\_r4.LoadBitmap(IDB\_R401);

}

void GameOver::OnShow(Role \*role,int size)

{

\_bankruptBK.SetTopLeft(80, 100);

\_winBK.SetTopLeft(80, 100);

\_OK.SetTopLeft(300, 350);

\_OKT.SetTopLeft(300, 350);

\_r1.SetTopLeft(300, 250);

\_r2.SetTopLeft(300, 250);

\_r3.SetTopLeft(300, 250);

\_r4.SetTopLeft(300, 250);

if (\_isOpen)

{

if (size >= 2)//------------------破產的畫面

\_bankruptBK.ShowBitmap();

else//----------------------------勝利者的畫面

\_winBK.ShowBitmap();

if (role->GetRoleNum() == 0)

\_r1.ShowBitmap();

else if (role->GetRoleNum() == 1)

\_r2.ShowBitmap();

else if (role->GetRoleNum() == 2)

\_r3.ShowBitmap();

else if (role->GetRoleNum() == 3)

\_r4.ShowBitmap();

if(\_isTouch)

\_OKT.ShowBitmap();

else

\_OK.ShowBitmap();

}

}

}

Lottery.h

#pragma once

#include"Role.h"

#include"Station.h"

namespace game\_framework

{

class Lottery

{

public:

Lottery();

void SetNum(int, Role \*);//----------------------------------------------------儲存角色簽的數字

void SetIsOK(bool flag);//-----------------------------------------------------設定是否確認

bool GetIsOK();//--------------------------------------------------------------是否已經確認結果

bool GetIsOpen();//------------------------------------------------------------是否UI介面還是打開的

int GetOwner(int);//----------------------------------------------------------得知該是誰選了這個數字

void Reset();//----------------------------------------------------------------開獎後清除所有資料

void LoadBitmap();

void OnShow(vector<Role\*> &roleLList, Role\* role);

bool CanChoose(Role\* player);//------------------------------------------------是否還有資格繼續選數字

void DrawTheNumber(int x, int y, bool mouseclick, vector<Role\*> &roleList); //-開獎

private:

CMovingBitmap \_BK, Large, \_OK, \_OKT, \_lotteryNum[10], \_lotteryRole[4];

CAnimation \_openRandom[10];//--------------------------------------------------10個數字

int \_roleLoc[4][5] = { {50,225,100,160,220} ,//--------------------------------樂透數字的顯示位置

/\*---------------------------\*/{320,225,370,430,490},

/\*---------------------------\*/{50,300,100,160,220},

/\*---------------------------\*/{320,300,370,430,490} };

bool \_isDrawing, \_isOK, \_isOKTouch, \_lotteryRowSet[10]; //---------------------isopen 是否顯示UI、isOK是否做完、\_lotteryRowSet是否已排好每人顯示的數字

int \_numberOwner[10], \_openNum;//----------------------------------------------誰簽的數字、開獎數字

bool \_isSendMoney;//-----------------------------------------------------------是否已發錢

int \_randomCount;//-----------------------------------------------------------公布樂透的閃爍數字

const int AWARD\_1 = 3000, VALUE = 1000;//--------------------------------------獎金、簽一次的價格

};

}

Lottery.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include <random>

#include "Lottery.h"

namespace game\_framework

{

Lottery::Lottery()

{

Reset();

}

bool Lottery::GetIsOK()

{

return \_isOK;

}

void Lottery::SetIsOK(bool flag)

{

\_isOK = flag;

}

int Lottery::GetOwner(int num)

{

return \_numberOwner[num];

}

void Lottery::SetNum(int num, Role \*player)

{

\_numberOwner[num] = player->GetRoleNum();

player->AddMoney(-VALUE);

}

bool Lottery::GetIsOpen() {

return \_isDrawing;

}

void Lottery::Reset()

{

for (int i = 0; i < 10; i++)

{

\_lotteryRowSet[i] = false;

\_numberOwner[i] = -1;

}

\_randomCount = 0;

std::random\_device rd;

std::default\_random\_engine gen = std::default\_random\_engine(rd());

std::uniform\_int\_distribution<int> dis(1, 10);

\_openNum = dis(gen);

\_isSendMoney = false;

\_isOK = true;

\_isDrawing = false;

}

bool Lottery::CanChoose(Role \*role)

{

int times = 0;

for (int i = 0; i < 10; i++)

if (role->GetRoleNum() == \_numberOwner[i])

times++;

if (times >= 3)

return false;

else

return true;

}

void Lottery::DrawTheNumber(int x, int y, bool mouseclick, vector<Role\*> &roleList)

{

if (!\_isOK)

{

\_isDrawing = true;

if (!\_isSendMoney)

{

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

{

if (\_numberOwner[\_openNum - 1] != -1 && roleList[i]->GetRoleNum() == \_numberOwner[\_openNum - 1])//該數字有人抽，且得獎者得存活才發錢

roleList[\_numberOwner[\_openNum - 1]]->AddMoney(AWARD\_1);

else

break;

}

}

\_isSendMoney = true;

if (x > 280 && x < 280 + \_OK.Width() && y>370 && y < 370 + \_OK.Height())

{

\_isOKTouch = true;

if (mouseclick)

{

Reset();

}

}

else

\_isOKTouch = false;

}

}

void Lottery::LoadBitmap()

{

\_BK.LoadBitmap(IDB\_L\_BK);

\_OKT.LoadBitmap(IDB\_L\_OK\_T);

\_OK.LoadBitmap(IDB\_L\_OK);

for (int i = 0; i < 10; i++)

{

\_openRandom[i].AddBitmap(IDB\_random1, RGB(255, 255, 255));

\_openRandom[i].AddBitmap(IDB\_random2, RGB(255, 255, 255));

\_openRandom[i].AddBitmap(IDB\_random3, RGB(255, 255, 255));

\_openRandom[i].AddBitmap(IDB\_random4, RGB(255, 255, 255));

\_openRandom[i].AddBitmap(IDB\_random5, RGB(255, 255, 255));

\_openRandom[i].AddBitmap(IDB\_random6, RGB(255, 255, 255));

\_openRandom[i].AddBitmap(IDB\_random7, RGB(255, 255, 255));

}

\_openRandom[0].AddBitmap(IDB\_open1, RGB(255, 255, 255));

\_openRandom[1].AddBitmap(IDB\_open2, RGB(255, 255, 255));

\_openRandom[2].AddBitmap(IDB\_open3, RGB(255, 255, 255));

\_openRandom[3].AddBitmap(IDB\_open4, RGB(255, 255, 255));

\_openRandom[4].AddBitmap(IDB\_open5, RGB(255, 255, 255));

\_openRandom[5].AddBitmap(IDB\_open6, RGB(255, 255, 255));

\_openRandom[6].AddBitmap(IDB\_open7, RGB(255, 255, 255));

\_openRandom[7].AddBitmap(IDB\_open8, RGB(255, 255, 255));

\_openRandom[8].AddBitmap(IDB\_open9, RGB(255, 255, 255));

\_openRandom[9].AddBitmap(IDB\_open10, RGB(255, 255, 255));

\_lotteryNum[0].LoadBitmap(IDB\_L1);

\_lotteryNum[1].LoadBitmap(IDB\_L2);

\_lotteryNum[2].LoadBitmap(IDB\_L3);

\_lotteryNum[3].LoadBitmap(IDB\_L4);

\_lotteryNum[4].LoadBitmap(IDB\_L5);

\_lotteryNum[5].LoadBitmap(IDB\_L6);

\_lotteryNum[6].LoadBitmap(IDB\_L7);

\_lotteryNum[7].LoadBitmap(IDB\_L8);

\_lotteryNum[8].LoadBitmap(IDB\_L9);

\_lotteryNum[9].LoadBitmap(IDB\_L10);

\_lotteryRole[0].LoadBitmapA(IDB\_L\_R1, RGB(255, 255, 254));

\_lotteryRole[1].LoadBitmapA(IDB\_L\_R2, RGB(255, 255, 254));

\_lotteryRole[2].LoadBitmapA(IDB\_L\_R3, RGB(255, 255, 254));

\_lotteryRole[3].LoadBitmapA(IDB\_L\_R4, RGB(255, 255, 254));

}

void Lottery::OnShow(vector<Role\*> &roleList, Role\* role)

{

if (\_isDrawing)

{

\_BK.SetTopLeft(40, 40);

\_BK.ShowBitmap();

\_openRandom[\_openNum - 1].SetTopLeft(240, 90);

\_openRandom[\_openNum - 1].OnShow();

if (!\_openRandom[\_openNum - 1].IsFinalBitmap())//---------------開獎的七彩數字

{

\_openRandom[\_openNum - 1].OnMove();

}

if (\_randomCount < 7)

{

\_randomCount++;

}

\_OK.SetTopLeft(280, 370);

\_OKT.SetTopLeft(280, 370);

if (\_isOKTouch)

\_OKT.ShowBitmap();

else

\_OK.ShowBitmap();

//--------------------------------------------------------------腳色欄

for (unsigned int i = 0; i < roleList.size(); i++)//------------判斷當前存活的角色

{

\_lotteryRole[roleList[i]->GetRoleNum()].SetTopLeft(\_roleLoc[static\_cast<int> (i)][0], \_roleLoc[static\_cast<int>(i)][1]);

}

for (unsigned int i = 0; i < roleList.size(); i++)//------------Show出所有存活的角色

{

\_lotteryRole[roleList[i]->GetRoleNum()].ShowBitmap();

}

for (unsigned int u\_who = 0; u\_who < roleList.size(); u\_who++)//每個角色所選的樂透數字

{

int index = 2;

int who = static\_cast<int>(u\_who);

for (int i = 0; i < 10; i++)

{

if (\_numberOwner[i] == roleList[who]->GetRoleNum())

{

\_lotteryNum[i].SetTopLeft(\_roleLoc[static\_cast<int> (who)][index], \_roleLoc[static\_cast<int>(who)][1] + 5);

\_lotteryNum[i].ShowBitmap();

index++;

}

}

}

}

}

}

OthersUI.h

#pragma once

#include"Role.h"

#include"Station.h"

#include"Lottery.h"

#include"Integer.h"

namespace game\_framework

{

class OthersUI

{

public:

OthersUI();

void SetIsOpen(bool flag);//---------------------------------------------------------------------------------------------- 設定個人選單關閉

void SetIsOpen();//--------------------------------------------------------------------------------------------------------開關個人選單

bool GetMenuIsOpen();//----------------------------------------------------------------------------------------------------個人選歌是否開啟

void JudgeUI(int x, int y, bool mouseclick, vector<Role\*> &roleList, Role\* role, Station\*\* station, Lottery& lottery);//---判斷當前應該在什麼UI

void LotteryUI(int x, int y, bool mouseclick, vector<Role\*> &roleList, Role\* role, Lottery& lottery);//--------------------進入樂透UI

void ResetLotteryUI();//---------------------------------------------------------------------------------------------------開獎時清除所有人的樂透UI資料

void LoadBitmap();

void SetOnce(bool);//------------------------------------------------------------------------------------------------------該回合已經選過一次數字

bool GetOthersIsOpen();//--------------------------------------------------------------------------------------------------是否有任何UI開啟中

void OnShow(vector<Role\*> &roleList, Role\* role, Lottery &lottery);

private:

////////// menu part /////////

Integer \_menuMoney, \_menuHouse;

CMovingBitmap \_option[2], \_optionT[2], \_menuBK, \_menuPhoto[4], \_moneyLogo, \_houseLogo;

bool \_isMOpen, \_optionTouch[2];

////////// Lottery part /////////

CMovingBitmap \_LBK, \_LOK, \_LOKT, \_lotteryNum[10], \_lotteryNumT[10], \_lotteryNumC[10], \_lotteryRole[4];

bool \_isLOpen, \_isOK;

bool \_lotteryTouch[10], \_lotteryChoose[10], \_lotteryTempChoose[10], \_LOKTouch, \_lotteryRowSet[10], \_once;

int \_roleLoc[4][5] = {{50,225,100,160,220} ,//樂透數字的顯示位置

/\*--------------------------\*/{320,225,370,430,490},

/\*--------------------------\*/{50,300,100,160,220},

/\*--------------------------\*/{320,300,370,430,490}};

};

}

OthersUI.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "OthersUI.h"

namespace game\_framework

{

OthersUI::OthersUI()

{

ResetLotteryUI();

}

void OthersUI::ResetLotteryUI()

{

\_optionTouch[0] = \_isMOpen = \_isLOpen = \_LOKTouch = \_once = false;

for (int i = 0; i < 10; i++)

{

\_lotteryTouch[i] = \_lotteryChoose[i] = \_lotteryRowSet[i] = \_lotteryTempChoose[i] = false;

}

}

void OthersUI::SetIsOpen()

{

if (\_isMOpen)

\_isMOpen = false;

else

\_isMOpen = true;

}

void OthersUI::SetIsOpen(bool flag)

{

\_isMOpen = flag;

}

bool OthersUI::GetMenuIsOpen() {

return \_isMOpen;

}

bool OthersUI::GetOthersIsOpen()

{

if (\_isLOpen)

return true;

return false;

}

void OthersUI::SetOnce(bool flag)

{

\_once = flag;

}

void OthersUI::JudgeUI(int x, int y, bool mouseclick, vector<Role\*> &roleList, Role\* role, Station\*\* station, Lottery& lottery)

{

if (\_isMOpen && !\_isLOpen)//MENU open

{

\_menuMoney.SetInteger(role->GetMoney());

int j = 0;

for (int i = 0; i < 81; i++)

{

if (station[i]->house != nullptr)//避免站點是空的地方

{

if (station[i]->GetBuildingType() == "House"&&station[i]->house->GetOwner() == role->GetRoleNum())

{

j++;

}

}

}

\_menuHouse.SetInteger(j);//設定個人的房地產

for (int i = 0; i < 1; i++)

{

if (x > 100 && x < 100 + \_option[0].Width() && y>60 + i \* (\_option[0].Height() + 20) && y < 60 + i \* (\_option[0].Height() + 20) + \_option[0].Height())

{

\_optionTouch[i] = true;

if (mouseclick)

{

switch (i)

{

case(0):

\_isLOpen = true;

Sleep(150);

break;

}

}

}

else

\_optionTouch[i] = false;

}

}

else if (\_isLOpen)//樂透介面開啟

{

LotteryUI(x, y, mouseclick, roleList, role, lottery);

}

}

void OthersUI::LotteryUI(int x, int y, bool mouseclick, vector<Role\*> &roleList, Role\* role, Lottery& lottery)

{

if (\_isLOpen)

{

if (lottery.CanChoose(role))//沒超過3個號碼

{

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

{

if (x > 50 + 90 \* i&&x < 50 + 90 \* i + \_lotteryNum[i].Width() && y>50 && y < 50 + \_lotteryNum[i].Height() && !\_once)

{

\_lotteryTouch[i] = true;

if (mouseclick && !\_lotteryChoose[i])

{

if (\_lotteryTempChoose[i])

{

\_lotteryTempChoose[i] = false;

Sleep(150);

}

else

{

\_lotteryTempChoose[i] = true;

Sleep(150);

}

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

{

if (i == j)

continue;

\_lotteryTempChoose[j] = false;

}

}

}

else

\_lotteryTouch[i] = false;

if (x > 120 + 90 \* i&&x < 120 + 90 \* i + \_lotteryNum[i + 5].Width() && y>145 && y < 145 + \_lotteryNum[i + 5].Height() && !\_once)

{

\_lotteryTouch[i + 5] = true;

if (mouseclick && !\_lotteryChoose[i + 5] && !\_once)

{

if (\_lotteryTempChoose[i + 5])

{

\_lotteryTempChoose[i + 5] = false;

Sleep(150);

}

else

{

\_lotteryTempChoose[i + 5] = true;

Sleep(150);

}

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

{

if (i + 5 == j)

continue;

\_lotteryTempChoose[j] = false;

}

}

}

else

\_lotteryTouch[i + 5] = false;

}

}

if (x > 280 && x < 280 + \_LOK.Width() && y>370 && y < 370 + \_LOK.Height())//確認鍵

{

\_LOKTouch = true;

if (mouseclick)

{

Sleep(150);

for (int i = 0; i < 10; i++)

{

if (\_lotteryTempChoose[i])

{

lottery.SetNum(i, role);

\_lotteryChoose[i] = true;

\_lotteryTempChoose[i] = false;

\_once = true; //一回合只能簽一次

}

}

\_isLOpen = false;

\_isMOpen = true;

}

}

else

\_LOKTouch = false;

}

}

void OthersUI::LoadBitmap()

{

\_menuBK.LoadBitmapA(IDB\_menu\_BK);

\_menuMoney.LoadBitmap();

\_menuHouse.LoadBitmap();

\_moneyLogo.LoadBitmap(IDB\_logo\_money, RGB(255, 255, 255));

\_houseLogo.LoadBitmap(IDB\_logo\_house, RGB(255, 255, 255));

\_menuPhoto[0].LoadBitmap(IDB\_R101, RGB(0, 0, 100));

\_menuPhoto[1].LoadBitmap(IDB\_R201, RGB(0, 0, 100));

\_menuPhoto[2].LoadBitmap(IDB\_R301, RGB(0, 0, 100));

\_menuPhoto[3].LoadBitmap(IDB\_R401, RGB(0, 0, 100));

\_option[0].LoadBitmap(IDB\_lottery\_option);

\_optionT[0].LoadBitmap(IDB\_lottery\_option\_choosed);

\_option[1].LoadBitmap(IDB\_bag\_option);

\_optionT[1].LoadBitmap(IDB\_bag\_option\_choosed);

\_LBK.LoadBitmap(IDB\_L\_BK);

\_LOK.LoadBitmap(IDB\_L\_OK);

\_LOKT.LoadBitmapA(IDB\_L\_OK\_T);

\_lotteryNum[0].LoadBitmap(IDB\_L1);

\_lotteryNumC[0].LoadBitmap(IDB\_L1C);

\_lotteryNumT[0].LoadBitmap(IDB\_L1T);

\_lotteryNum[1].LoadBitmap(IDB\_L2);

\_lotteryNumC[1].LoadBitmap(IDB\_L2C);

\_lotteryNumT[1].LoadBitmap(IDB\_L2T);

\_lotteryNum[2].LoadBitmap(IDB\_L3);

\_lotteryNumC[2].LoadBitmap(IDB\_L3C);

\_lotteryNumT[2].LoadBitmap(IDB\_L3T);

\_lotteryNum[3].LoadBitmap(IDB\_L4);

\_lotteryNumC[3].LoadBitmap(IDB\_L4C);

\_lotteryNumT[3].LoadBitmap(IDB\_L4T);

\_lotteryNum[4].LoadBitmap(IDB\_L5);

\_lotteryNumC[4].LoadBitmap(IDB\_L5C);

\_lotteryNumT[4].LoadBitmap(IDB\_L5T);

\_lotteryNum[5].LoadBitmap(IDB\_L6);

\_lotteryNumC[5].LoadBitmap(IDB\_L6C);

\_lotteryNumT[5].LoadBitmap(IDB\_L6T);

\_lotteryNum[6].LoadBitmap(IDB\_L7);

\_lotteryNumC[6].LoadBitmap(IDB\_L7C);

\_lotteryNumT[6].LoadBitmap(IDB\_L7T);

\_lotteryNum[7].LoadBitmap(IDB\_L8);

\_lotteryNumC[7].LoadBitmap(IDB\_L8C);

\_lotteryNumT[7].LoadBitmap(IDB\_L8T);

\_lotteryNum[8].LoadBitmap(IDB\_L9);

\_lotteryNumC[8].LoadBitmap(IDB\_L9C);

\_lotteryNumT[8].LoadBitmap(IDB\_L9T);

\_lotteryNum[9].LoadBitmap(IDB\_L10);

\_lotteryNumC[9].LoadBitmap(IDB\_L10C);

\_lotteryNumT[9].LoadBitmap(IDB\_L10T);

\_lotteryRole[0].LoadBitmapA(IDB\_L\_R1, RGB(255, 255, 254));

\_lotteryRole[1].LoadBitmapA(IDB\_L\_R2, RGB(255, 255, 254));

\_lotteryRole[2].LoadBitmapA(IDB\_L\_R3, RGB(255, 255, 254));

\_lotteryRole[3].LoadBitmapA(IDB\_L\_R4, RGB(255, 255, 254));

}

void OthersUI::OnShow(vector<Role\*> &roleList, Role\* role, Lottery &lottery)

{

\_menuBK.SetTopLeft(80, 40);

\_menuMoney.SetTopLeft(470, 250);

\_moneyLogo.SetTopLeft(375, 240);

\_houseLogo.SetTopLeft(375, 275);

for (int i = 0; i < 2; i++)//---------------------------------------設定樂透按鈕位置

{

\_option[i].SetTopLeft(100, 60 + i \* (\_option[0].Height() + 20));

\_optionT[i].SetTopLeft(100, 60 + i \* (\_option[0].Height() + 20));

}

for (int i = 0; i < 4; i++)//---------------------------------------設定角色欄位置

\_menuPhoto[i].SetTopLeft(500, 50);

\_LBK.SetTopLeft(40, 40);

\_LOK.SetTopLeft(280, 370);

\_LOKT.SetTopLeft(280, 370);

for (int i = 0; i < 5; i++)//---------------------------------------樂透的數字

{

if (!\_lotteryChoose[i])//---------------------------------------被選到就不排

\_lotteryNum[i].SetTopLeft(50 + 90 \* i, 50);

\_lotteryNumC[i].SetTopLeft(50 + 90 \* i, 50);

\_lotteryNumT[i].SetTopLeft(50 + 90 \* i, 50);

}

for (int i = 5, j = 0; i < 10; i++, j++)//--------------------------樂透的數字

{

if (!\_lotteryChoose[i])//---------------------------------------被選到就不排

\_lotteryNum[i].SetTopLeft(120 + 90 \* j, 145);

\_lotteryNumC[i].SetTopLeft(120 + 90 \* j, 145);

\_lotteryNumT[i].SetTopLeft(120 + 90 \* j, 145);

}

for (unsigned int i = 0; i < roleList.size(); i++)//----------------判斷當前存活的角色

{

\_lotteryRole[roleList[i]->GetRoleNum()].SetTopLeft(\_roleLoc[static\_cast<int> (i)][0], \_roleLoc[static\_cast<int>(i)][1]);

}

if (\_isMOpen && !\_isLOpen)//----------------------------------------Show 個人選單

{

\_menuBK.ShowBitmap();

if (\_optionTouch[0])

\_optionT[0].ShowBitmap();

else

\_option[0].ShowBitmap();

\_menuPhoto[role->GetRoleNum()].ShowBitmap();

\_menuMoney.ShowBitmap();

\_moneyLogo.ShowBitmap();

\_menuHouse.ShowBitmap();

\_houseLogo.ShowBitmap();

}

else if (\_isLOpen)//------------------------------------------------Show 樂透的UI

{

\_LBK.ShowBitmap();

for (unsigned int i = 0; i < roleList.size(); i++)//------------角色欄

{

\_lotteryRole[roleList[i]->GetRoleNum()].ShowBitmap();

}

if (\_LOKTouch)//------------------------------------------------確認鍵

\_LOKT.ShowBitmap();

else

\_LOK.ShowBitmap();

for (int i = 0; i < 10; i++)//----------------------------------可選的樂透數字

{

if (!\_lotteryTouch[i] && !\_lotteryChoose[i] && !\_lotteryTempChoose[i])

\_lotteryNum[i].ShowBitmap();

else if ((\_lotteryTouch[i] && !\_lotteryChoose[i]) || \_lotteryTempChoose[i])

\_lotteryNumT[i].ShowBitmap();

else if (\_lotteryChoose[i])

\_lotteryNumC[i].ShowBitmap();

}

for (unsigned int u\_who = 0; u\_who < roleList.size(); u\_who++)//角色欄數字

{

int index = 2;

int who = static\_cast<int>(u\_who);

for (int i = 0; i < 10; i++)

{

if (lottery.GetOwner(i) == roleList[who]->GetRoleNum())

{

\_lotteryNum[i].SetTopLeft(\_roleLoc[static\_cast<int> (who)][index], \_roleLoc[static\_cast<int>(who)][1] + 5);

\_lotteryNum[i].ShowBitmap();

index++;

}

}

}

}

}

}

Round.h

#pragma once

namespace game\_framework

{

class Round

{

public:

Round();

void AddRound(int, int);

void SetRound(int i);

int GetRound();

protected:

int \_round = 0;

};

}

Round.cpp

#include "stdafx.h"

#include"Round.h"

namespace game\_framework

{

Round::Round()

{

\_round = 0;

}

void Round::AddRound(int whoturn, int playamount)

{

if (whoturn == playamount - 1)

\_round++;

}

int Round::GetRound()

{

return \_round;

}

void Round::SetRound(int i) {

\_round = i;

}

}

Station.h

#pragma once

#include "House.h"

#include "Store.h"

namespace game\_framework

{

class Station

{

public:

Station();

int GetX();

int GetY();

void SetXY(int nx, int ny);

void LoadBitmap();

void OnShow();

virtual void OnMove();

virtual void OnMove(int camerax, int cameray);

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

House \*house;

Store \*store;

string GetBuildingType();

void SetBuildingType(string buildingType);

protected:

int \_x, \_y;

int \_referenceX, \_referenceY;

string \_buildingType;

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

CMovingBitmap station;

};

}

Dice.h

#pragma once

namespace game\_framework

{

class Dice {

public:

Dice();

int GetDiceNumber();

void setXY(int x, int y);

void TossDice(int diceNum);

void SetDiceNumber(int diceNum);

void SetDiceIsToss(bool flag);

bool IsToss();

void OnShow();

void LoadBitmap();

private:

CMovingBitmap \_one, \_two, \_three, \_four, \_five, \_six, \_zero;

bool \_isToss; //預設為true(可丟) 丟之後會成false 直到腳色到點變回true

int \_diceNumber, \_x, \_y;

};

}

Dice.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include <random>

#include "gamelib.h"

#include "Dice.h"

namespace game\_framework

{

Dice::Dice(): \_diceNumber(0) ,\_isToss(true), \_x(580), \_y(420){}

int Dice::GetDiceNumber() {

return \_diceNumber;

}

void Dice::setXY(int x, int y) {

\_x = x;

\_y = y;

}

void Dice::SetDiceIsToss(bool flag) {

\_isToss = flag;

}

bool Dice::IsToss() {

return \_isToss;

}

void Dice::SetDiceNumber(int diceNum) {

\_diceNumber = diceNum;

}

void Dice::TossDice(int diceNum) {

if (\_isToss) {

if (diceNum == 0) {

std::random\_device rd;

std::default\_random\_engine gen = std::default\_random\_engine(rd());

std::uniform\_int\_distribution<int> dis(1, 6);

\_diceNumber = dis(gen);

}

else {

\_diceNumber = diceNum;

}

\_isToss = false;

}

}

void Dice::LoadBitmap() {

\_zero.LoadBitmap(IDB\_Question);

\_one.LoadBitmap(IDB\_DICEONE);

\_two.LoadBitmap(IDB\_DICETWO);

\_three.LoadBitmap(IDB\_DICETHREE);

\_four.LoadBitmap(IDB\_DICEFOUR);

\_five.LoadBitmap(IDB\_DICEFIVE);

\_six.LoadBitmap(IDB\_DICESIX);

}

void Dice::OnShow() {

switch (\_diceNumber) {

case 0:

\_zero.SetTopLeft(\_x, \_y);

\_zero.ShowBitmap();

break;

case 1:

\_one.SetTopLeft(\_x, \_y);

\_one.ShowBitmap();

break;

case 2:

\_two.SetTopLeft(\_x, \_y);

\_two.ShowBitmap();

break;

case 3:

\_three.SetTopLeft(\_x, \_y);

\_three.ShowBitmap();

break;

case 4:

\_four.SetTopLeft(\_x, \_y);

\_four.ShowBitmap();

break;

case 5:

\_five.SetTopLeft(\_x, \_y);

\_five.ShowBitmap();

break;

case 6:

\_six.SetTopLeft(\_x, \_y);

\_six.ShowBitmap();

break;

}

}

}

Role.h

#pragma once

#include "Integer.h"

#include "Station.h"

namespace game\_framework

{

class Role

{

public:

Role();

Role(int roleNumber);

int GetCameraX();

int GetCameraY();

int GetX();

int GetY();

void SetShowRole(bool flag);

void AddMoney(int price);

int GetMoney();

int GetRoleNum();

int GetRoleState();

void SetRoleTurn(bool flag);

virtual void OnMove();

virtual void OnMove(int camerax, int cameray);//視角切換

bool SetRoleMove(Station \*state,int direction); //外部移動腳色 directoin>0正向 <0反向

int RoleMove(int diceNum, Station \*\*state);

void OnShow(Station \*\*state);

void SetMovingDown(bool flag);

void SetMovingLeft(bool flag);

void SetMovingRight(bool flag);

void SetMovingUp(bool flag);

void LoadBitmap();

Integer \_money;

private:

CAnimation \_roleOneDown, \_roleOneUp, \_roleOneLeft, \_roleOneRight, \_roleTwoDown, \_roleTwoUp, \_roleTwoLeft, \_roleTwoRight, \_roleThreeDown, \_roleThreeUp, \_roleThreeLeft, \_roleThreeRight, \_roleFourDown, \_roleFourUp, \_roleFourLeft, \_roleFourRight;

int \_x, \_y, \_stepNum, \_referenceX, \_referenceY, \_mystate;

unsigned int \_roleNumber;

bool \_roleTurn, \_showRole;

bool \_isMovingDown;

bool \_isMovingLeft;

bool \_isMovingRight;

bool \_isMovingUp;

};

}

Role.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Role.h"

namespace game\_framework

{

Role::Role()

:\_referenceX(0), \_referenceY(0), \_stepNum(0), \_x(560), \_y(280) ,\_mystate(0), \_money(5){

\_roleTurn = \_isMovingLeft = \_isMovingRight = \_isMovingUp = \_isMovingDown = false;

\_showRole = true;

\_money.SetInteger(10000);

}

Role::Role(int roleNumber) : \_referenceX(0), \_referenceY(0), \_stepNum(0), \_money(5),\_roleNumber(roleNumber) {

switch (roleNumber) {

case 0:

\_x = 560;

\_y = 280;

\_mystate = 0;

break;

case 1:

\_x = 224;

\_y = 960;

\_mystate = 10;

break;

case 2:

\_x = 704;

\_y = 1944;

\_mystate = 30;

break;

case 3:

\_x = 1388;

\_y = 228;

\_mystate = 60;

break;

default:

\_x = 560;

\_y = 280;

\_mystate = 0;

}

\_showRole = true;

\_roleTurn = \_isMovingLeft = \_isMovingRight = \_isMovingUp = \_isMovingDown = false;

\_money.SetInteger(10000);

}

void Role::SetShowRole(bool flag) {

\_showRole = flag;

}

int Role::GetCameraX()

{

if (320 != \_x ) {

return 320 - \_x;

}

else {

return 0;

}

}

int Role::GetCameraY()

{

if (240 != \_y) {

return 240 - \_y;

}

else {

return 0;

}

}

int Role::GetX() {

return \_x;

}

int Role::GetY() {

return \_y;

}

void Role::AddMoney(int price) {

\_money.Add(price);

}

int Role::GetMoney() {

return \_money.GetInteger();

}

int Role::GetRoleNum() {

return \_roleNumber;

}

int Role::GetRoleState() {

return \_mystate;

}

bool Role::SetRoleMove(Station\* state,int direction) {

int stepSize = 4;

if (\_x != state->GetX()) {

if (\_x < state->GetX()) {

\_x += stepSize;

}

if (\_x > state->GetX()) {

\_x -= stepSize;

}

}

if (\_y != state->GetY()) {

if (\_y < state->GetY()) {

\_y += stepSize;

}

if (\_y > state->GetY()) {

\_y -= stepSize;

}

}

if (\_x == state->GetX() && \_y == state->GetY()) {

if (direction > 0) {

\_mystate++;

if (\_mystate == 81) {

\_mystate = 0;

}

}

else if(direction<0){

\_mystate--;

if (\_mystate == -1) {

\_mystate = 80;

}

}

return true;

}

else {

return false;

}

}

int Role::RoleMove(int diceNum, Station \*\*state) {

int stepSize = 4;

int nextstate = \_mystate + 1;

if (\_mystate == 80) {

nextstate = 0;

}

if (diceNum > 0) {

if (\_x != state[nextstate]->GetX()) {

if (\_x < state[nextstate]->GetX()) {

\_x += stepSize;

}

if (\_x > state[nextstate]->GetX()) {

\_x -= stepSize;

}

}

if (\_y != state[nextstate]->GetY()) {

if (\_y < state[nextstate]->GetY()) {

\_y += stepSize;

}

if (\_y > state[nextstate]->GetY()) {

\_y -= stepSize;

}

}

if (\_x == state[nextstate]->GetX() && \_y == state[nextstate]->GetY()) {

++\_mystate;

if (\_mystate > 80) {

\_mystate = 0;

}

return diceNum-1;

}

}

if (\_roleNumber == 0) {

\_roleOneDown.OnMove();

\_roleOneUp.OnMove();

\_roleOneLeft.OnMove();

\_roleOneRight.OnMove();

}

else if (\_roleNumber == 1) {

\_roleTwoDown.OnMove();

\_roleTwoUp.OnMove();

\_roleTwoLeft.OnMove();

\_roleTwoRight.OnMove();

}

else if (\_roleNumber == 2) {

\_roleThreeDown.OnMove();

\_roleThreeUp.OnMove();

\_roleThreeLeft.OnMove();

\_roleThreeRight.OnMove();

}

else if (\_roleNumber == 3) {

\_roleFourDown.OnMove();

\_roleFourUp.OnMove();

\_roleFourLeft.OnMove();

\_roleFourRight.OnMove();

}

return diceNum;

}

void Role::SetRoleTurn(bool flag)

{

\_roleTurn = flag;

}

void Role::OnMove()

{

const int STEP\_SIZE = 10;

if (\_referenceX + STEP\_SIZE <= 0 && \_isMovingLeft) {

\_x += STEP\_SIZE;

\_referenceX += STEP\_SIZE;

}

if (\_referenceX - STEP\_SIZE >= -1280 && \_isMovingRight) {

\_x -= STEP\_SIZE;

\_referenceX -= STEP\_SIZE;

}

if (\_referenceY + STEP\_SIZE <= 0 && \_isMovingUp) {

\_y += STEP\_SIZE;

\_referenceY += STEP\_SIZE;

}

if (\_referenceY - STEP\_SIZE >= -1920 && \_isMovingDown) {

\_y -= STEP\_SIZE;

\_referenceY -= STEP\_SIZE;

}

}

void Role::OnMove(int camerax, int cameray) {

if (\_referenceX + camerax <= 0 && \_referenceX + camerax >= -1280) {

\_referenceX += camerax;

\_x += camerax;

}

else if (camerax > 0) {

\_x = \_x - \_referenceX;

\_referenceX = 0;

\_x = \_referenceX + \_x;

}

else if (camerax < 0) {

\_x = \_x - \_referenceX;

\_referenceX = -1280;

\_x = \_referenceX + \_x;

}

if (\_referenceY + cameray <= 0 && \_referenceY + cameray >= -1920) {

\_referenceY += cameray;

\_y += cameray;

}

else if (cameray > 0) {

\_y = \_y - \_referenceY;

\_referenceY = 0;

\_y = \_referenceY + \_y;

}

else if (cameray < 0) {

\_y = \_y - \_referenceY;

\_referenceY = -1920;

\_y = \_referenceY + \_y;

}

}

void Role::LoadBitmap() {

\_money.LoadBitmap();

if (\_roleNumber==0) {

\_roleOneDown.AddBitmap(IDB\_R101, RGB(0, 0, 100));

\_roleOneDown.AddBitmap(IDB\_R102, RGB(0, 0, 100));

\_roleOneDown.AddBitmap(IDB\_R103, RGB(0, 0, 100));

\_roleOneDown.AddBitmap(IDB\_R104, RGB(0, 0, 100));

\_roleOneUp.AddBitmap(IDB\_R113, RGB(0, 0, 100));

\_roleOneUp.AddBitmap(IDB\_R114, RGB(0, 0, 100));

\_roleOneUp.AddBitmap(IDB\_R115, RGB(0, 0, 100));

\_roleOneUp.AddBitmap(IDB\_R116, RGB(0, 0, 100));

\_roleOneLeft.AddBitmap(IDB\_R105, RGB(0, 0, 100));

\_roleOneLeft.AddBitmap(IDB\_R106, RGB(0, 0, 100));

\_roleOneLeft.AddBitmap(IDB\_R107, RGB(0, 0, 100));

\_roleOneLeft.AddBitmap(IDB\_R108, RGB(0, 0, 100));

\_roleOneRight.AddBitmap(IDB\_R109, RGB(0, 0, 100));

\_roleOneRight.AddBitmap(IDB\_R110, RGB(0, 0, 100));

\_roleOneRight.AddBitmap(IDB\_R111, RGB(0, 0, 100));

\_roleOneRight.AddBitmap(IDB\_R112, RGB(0, 0, 100));

}

else if (\_roleNumber == 1) {

\_roleTwoDown.AddBitmap(IDB\_R201, RGB(0, 0, 100));

\_roleTwoDown.AddBitmap(IDB\_R202, RGB(0, 0, 100));

\_roleTwoDown.AddBitmap(IDB\_R203, RGB(0, 0, 100));

\_roleTwoDown.AddBitmap(IDB\_R204, RGB(0, 0, 100));

\_roleTwoUp.AddBitmap(IDB\_R213, RGB(0, 0, 100));

\_roleTwoUp.AddBitmap(IDB\_R214, RGB(0, 0, 100));

\_roleTwoUp.AddBitmap(IDB\_R215, RGB(0, 0, 100));

\_roleTwoUp.AddBitmap(IDB\_R216, RGB(0, 0, 100));

\_roleTwoLeft.AddBitmap(IDB\_R205, RGB(0, 0, 100));

\_roleTwoLeft.AddBitmap(IDB\_R206, RGB(0, 0, 100));

\_roleTwoLeft.AddBitmap(IDB\_R207, RGB(0, 0, 100));

\_roleTwoLeft.AddBitmap(IDB\_R208, RGB(0, 0, 100));

\_roleTwoRight.AddBitmap(IDB\_R209, RGB(0, 0, 100));

\_roleTwoRight.AddBitmap(IDB\_R210, RGB(0, 0, 100));

\_roleTwoRight.AddBitmap(IDB\_R211, RGB(0, 0, 100));

\_roleTwoRight.AddBitmap(IDB\_R212, RGB(0, 0, 100));

}

else if (\_roleNumber == 2) {

\_roleThreeDown.AddBitmap(IDB\_R301, RGB(0, 0, 100));

\_roleThreeDown.AddBitmap(IDB\_R302, RGB(0, 0, 100));

\_roleThreeDown.AddBitmap(IDB\_R303, RGB(0, 0, 100));

\_roleThreeDown.AddBitmap(IDB\_R304, RGB(0, 0, 100));

\_roleThreeUp.AddBitmap(IDB\_R313, RGB(0, 0, 100));

\_roleThreeUp.AddBitmap(IDB\_R314, RGB(0, 0, 100));

\_roleThreeUp.AddBitmap(IDB\_R315, RGB(0, 0, 100));

\_roleThreeUp.AddBitmap(IDB\_R316, RGB(0, 0, 100));

\_roleThreeLeft.AddBitmap(IDB\_R305, RGB(0, 0, 100));

\_roleThreeLeft.AddBitmap(IDB\_R306, RGB(0, 0, 100));

\_roleThreeLeft.AddBitmap(IDB\_R307, RGB(0, 0, 100));

\_roleThreeLeft.AddBitmap(IDB\_R308, RGB(0, 0, 100));

\_roleThreeRight.AddBitmap(IDB\_R309, RGB(0, 0, 100));

\_roleThreeRight.AddBitmap(IDB\_R310, RGB(0, 0, 100));

\_roleThreeRight.AddBitmap(IDB\_R311, RGB(0, 0, 100));

\_roleThreeRight.AddBitmap(IDB\_R312, RGB(0, 0, 100));

}

else if (\_roleNumber == 3) {

\_roleFourDown.AddBitmap(IDB\_R401, RGB(0, 0, 100));

\_roleFourDown.AddBitmap(IDB\_R402, RGB(0, 0, 100));

\_roleFourDown.AddBitmap(IDB\_R403, RGB(0, 0, 100));

\_roleFourDown.AddBitmap(IDB\_R404, RGB(0, 0, 100));

\_roleFourUp.AddBitmap(IDB\_R413, RGB(0, 0, 100));

\_roleFourUp.AddBitmap(IDB\_R414, RGB(0, 0, 100));

\_roleFourUp.AddBitmap(IDB\_R415, RGB(0, 0, 100));

\_roleFourUp.AddBitmap(IDB\_R416, RGB(0, 0, 100));

\_roleFourLeft.AddBitmap(IDB\_R405, RGB(0, 0, 100));

\_roleFourLeft.AddBitmap(IDB\_R406, RGB(0, 0, 100));

\_roleFourLeft.AddBitmap(IDB\_R407, RGB(0, 0, 100));

\_roleFourLeft.AddBitmap(IDB\_R408, RGB(0, 0, 100));

\_roleFourRight.AddBitmap(IDB\_R409, RGB(0, 0, 100));

\_roleFourRight.AddBitmap(IDB\_R410, RGB(0, 0, 100));

\_roleFourRight.AddBitmap(IDB\_R411, RGB(0, 0, 100));

\_roleFourRight.AddBitmap(IDB\_R412, RGB(0, 0, 100));

}

}

void Role::SetMovingDown(bool flag)

{

\_isMovingDown = flag;

}

void Role::SetMovingLeft(bool flag)

{

\_isMovingLeft = flag;

}

void Role::SetMovingRight(bool flag)

{

\_isMovingRight = flag;

}

void Role::SetMovingUp(bool flag)

{

\_isMovingUp = flag;

}

void Role::OnShow(Station \*\*state) {

int nextstate = \_mystate + 1;

if (\_mystate == 80) {

nextstate = 0;

}

if (\_showRole) {

if (\_x <= 690 && \_x >= -50 && \_y >= -50 && \_y <= 530) {

if (\_roleNumber == 0) {

if (\_y < state[nextstate]->GetY()) {

\_roleOneDown.SetTopLeft(\_x + 5, \_y - 45);

\_roleOneDown.OnShow();

}

else if (\_y > state[nextstate]->GetY()) {

\_roleOneUp.SetTopLeft(\_x + 5, \_y - 45);

\_roleOneUp.OnShow();

}

else if (\_x > state[nextstate]->GetX()) {

\_roleOneLeft.SetTopLeft(\_x + 5, \_y - 45);

\_roleOneLeft.OnShow();

}

else if (\_x < state[nextstate]->GetX()) {

\_roleOneRight.SetTopLeft(\_x + 5, \_y - 45);

\_roleOneRight.OnShow();

}

else {

\_roleOneDown.SetTopLeft(\_x + 5, \_y - 45);

\_roleOneDown.OnShow();

}

}

else if (\_roleNumber == 1) {

if (\_y < state[nextstate]->GetY()) {

\_roleTwoDown.SetTopLeft(\_x + 2, \_y - 55);

\_roleTwoDown.OnShow();

}

else if (\_y > state[nextstate]->GetY()) {

\_roleTwoUp.SetTopLeft(\_x + 2, \_y - 55);

\_roleTwoUp.OnShow();

}

else if (\_x > state[nextstate]->GetX()) {

\_roleTwoLeft.SetTopLeft(\_x + 2, \_y - 55);

\_roleTwoLeft.OnShow();

}

else if (\_x < state[nextstate]->GetX()) {

\_roleTwoRight.SetTopLeft(\_x + 2, \_y - 55);

\_roleTwoRight.OnShow();

}

else {

\_roleTwoDown.SetTopLeft(\_x + 2, \_y - 55);

\_roleTwoDown.OnShow();

}

}

else if (\_roleNumber == 2) {

if (\_y < state[nextstate]->GetY()) {

\_roleThreeDown.SetTopLeft(\_x + 5, \_y - 45);

\_roleThreeDown.OnShow();

}

else if (\_y > state[nextstate]->GetY()) {

\_roleThreeUp.SetTopLeft(\_x + 5, \_y - 45);

\_roleThreeUp.OnShow();

}

else if (\_x > state[nextstate]->GetX()) {

\_roleThreeLeft.SetTopLeft(\_x + 5, \_y - 45);

\_roleThreeLeft.OnShow();

}

else if (\_x < state[nextstate]->GetX()) {

\_roleThreeRight.SetTopLeft(\_x + 5, \_y - 45);

\_roleThreeRight.OnShow();

}

else {

\_roleThreeDown.SetTopLeft(\_x + 5, \_y - 45);

\_roleThreeDown.OnShow();

}

}

else if (\_roleNumber == 3) {

if (\_y < state[nextstate]->GetY()) {

\_roleFourDown.SetTopLeft(\_x + 5, \_y - 55);

\_roleFourDown.OnShow();

}

else if (\_y > state[nextstate]->GetY()) {

\_roleFourUp.SetTopLeft(\_x + 5, \_y - 55);

\_roleFourUp.OnShow();

}

else if (\_x > state[nextstate]->GetX()) {

\_roleFourLeft.SetTopLeft(\_x + 5, \_y - 55);

\_roleFourLeft.OnShow();

}

else if (\_x < state[nextstate]->GetX()) {

\_roleFourRight.SetTopLeft(\_x + 5, \_y - 55);

\_roleFourRight.OnShow();

}

else {

\_roleFourDown.SetTopLeft(\_x + 5, \_y - 55);

\_roleFourDown.OnShow();

}

}

}

}

}

}

BuyUI.h

#pragma once

#include "Integer.h"

#include "GameMap.h"

#include "Building.h"

#include "Station.h"

#include "Role.h"

#include "Card.h"

namespace game\_framework

{

class BuyUI {

public:

BuyUI();

~BuyUI();

void SetIsOK(bool flag);

bool GetIsOK();

bool GetIsOpen();

void ResetUI();

void JudgeHS(int x, int y, bool mouseclick, vector<Role\*> &roleLList, Role\* role, Station\*\* station, Station\* rolestation, GameMap \*map, Building\*\* building, bool &isbankrupt);

void HouseUI(int x,int y, bool mouseclick,Role\* role, Station\* rolestation);

void StoreUI(int x, int y, bool mouseclick, Role\* role);

void HouseTransaction(Role\* role, Station\* rolestation);

void TollsTransaction(Role\* owner, Role\* payer, int cash);

void UseCard(vector<Role\*> &roleLList, Role\* role, Station\*\* station, Station\* rolestation, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building);

void LoadBitmap();

void OnShow(vector<Role\*> &roleLList, Role\* role, Station\*\* station, Station\* rolestation);

private:

CMovingBitmap \_UIBK, \_title[3], \_levelp[2][3], \_sell[2], \_poor[4] , \_OK, \_cardRandom, \_draw, \_noDraw;;

bool \_isOpen, \_showHouseUI, \_showStoreUI, \_isUIOK, \_isDraw, \_isOK; //isopen 是否顯示UI isOK是否做完

Integer \_dollar[4]; //房子買賣價格 role存款

string \_dothing; //doHouse, doTolls,doStore,doNothing

int \_level, \_cardIndex, \_cardPrice; // -1=sell 1~3==buy 0= donothing// \_cardIndex -1 =donothing

vector<Card\*> \_card;

};

}

BuyUI.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include <random>

#include "BuyUI.h"

namespace game\_framework

{

BuyUI::BuyUI(): \_isOpen(false), \_showHouseUI(false), \_showStoreUI(false),\_isUIOK(false), \_isOK(false), \_level (0), \_cardIndex(-1), \_dothing("doNothing")

{

\_dollar[0].SetInteger(0);

\_dollar[1].SetInteger(0);

\_dollar[2].SetInteger(0);

\_dollar[3].SetInteger(0);

\_cardPrice = 1000;

\_isDraw = false;

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

CardName cardname = (CardName)i;

Card\* card = new Card(cardname);

\_card.push\_back(card);

}

}

BuyUI::~BuyUI(){

\_card.clear();

}

void BuyUI::SetIsOK(bool flag) {

\_isOK = flag;

}

bool BuyUI::GetIsOK() {

return \_isOK;

}

bool BuyUI::GetIsOpen() {

return \_isOpen;

}

void BuyUI::JudgeHS(int x, int y, bool mouseclick,vector<Role\*> &roleList, Role\* role, Station\*\* station, Station\* rolestation, GameMap \*map, Building\*\* building, bool &isbankrupt) {

if(!\_isUIOK){

if (rolestation->GetBuildingType() == "House") {

\_dollar[0].SetInteger(rolestation->house->GetPrice(1)); //house1價格

\_dollar[1].SetInteger(rolestation->house->GetPrice(2)); //house2價格

\_dollar[2].SetInteger(rolestation->house->GetPrice(3)); //house3價格

\_dollar[3].SetInteger(rolestation->house->GetPrice(-1)); //賣house價格

if (role->GetRoleNum() == rolestation->house->GetOwner() || rolestation->house->GetOwner() == 4) {//土地為role的或無人土地

\_isOpen = true;

\_showHouseUI = true;

\_dothing = "doHouse";

HouseUI(x, y, mouseclick, role, rolestation);

}

else{

for (unsigned int i = 0; i < roleList.size(); i++) {

if (rolestation->house->GetOwner() == roleList[i]->GetRoleNum()) {

TollsTransaction(roleList[i], role, rolestation->house->GetTolls());

\_dothing == "doTolls";

}

}

\_isUIOK = true;

}

}

else if (rolestation->GetBuildingType() == "Store") {

\_isOpen = true;

\_showStoreUI = true;

\_dothing = "doStore";

StoreUI(x, y, mouseclick, role);

}

else {

\_isUIOK = true;

}

}

if (\_isUIOK) {

if (\_dothing == "doHouse") {

HouseTransaction(role, rolestation);

}

else if (\_dothing == "doTolls") {

}

else if (\_dothing == "doStore") {

UseCard(roleList, role ,station, rolestation, mouseclick , x, y, map, building);

}

else if (\_dothing == "doNothing") {

\_isOK = true;

}

if (\_isOK) {

ResetUI();

Sleep(200);

}

}

}

void BuyUI::ResetUI() { //重設選單

\_level = 0;

\_dothing = "doNothing";

\_isUIOK = false;

\_isDraw = false;

\_cardIndex = -1;

}

void BuyUI::HouseUI(int x, int y, bool mouseclick, Role\* role, Station\* rolestation) {

if (mouseclick) {

if (x > 180 && x < 260 && y>180 && y < 220 && \_level != 1 && role->GetMoney() >= rolestation->house->GetPrice(1)) { //選level 如果重複點選=取消

\_level = 1;

}

else if (x > 320 && x < 400 && y>180 && y < 220 && \_level != 2 && role->GetMoney() >= rolestation->house->GetPrice(2)) {

\_level = 2;

}

else if (x > 460 && x < 540 && y>180 && y < 220 && \_level != 3 && role->GetMoney() >= rolestation->house->GetPrice(3)) {

\_level = 3;

}

else if (x > 180 && x < 260 && y>180 && y < 220 && \_level == 1) {

\_level = 0;

}

else if (x > 320 && x < 400 && y>180 && y < 220 && \_level == 2) {

\_level = 0;

}

else if (x > 460 && x < 540 && y>180 && y < 220 && \_level == 3) {

\_level = 0;

}

else if (x > 180 && x < 260 && y>280 && y < 320 && \_level != -1 && role->GetRoleNum()== rolestation->house->GetOwner()) { //sellhouse

\_level = -1;

}

else if (x > 180 && x < 260 && y>280 && y < 320 && \_level != 0) {

\_level = 0;

}

else if (x > 290 && x < 370 && y>360 && y < 400) {

\_isUIOK = true;

\_showHouseUI = false;

\_isOpen = false;

}

Sleep(70);

}

}

void BuyUI::HouseTransaction(Role\* role, Station\* rolestation) {

if (\_level != 0) {

rolestation->house->SetLevel(\_level);

rolestation->house->SetOwner(role->GetRoleNum());

role->AddMoney(-rolestation->house->GetPrice(\_level));

}

\_isOK = true;

}

void BuyUI::TollsTransaction(Role\* owner, Role\* payer, int cash) {

owner->AddMoney(cash);

payer->AddMoney(-cash);

\_isOK = true;

}

void BuyUI::StoreUI(int x, int y, bool mouseclick, Role\* role)

{

if (mouseclick)

{

if (x > 470 && x < 550 && y>90 && y < 130 && role->GetMoney() > \_cardPrice)

{

std::random\_device rd;

std::default\_random\_engine gen = std::default\_random\_engine(rd());

std::uniform\_int\_distribution<int> dis(0, 2);

\_cardIndex = dis(gen);

\_isDraw = true;

role->AddMoney(-\_cardPrice);

Sleep(100);

}

else if (x > 470 && x < 550 && y>350 && y < 390)

{

\_isUIOK = true;

\_showStoreUI = false;

\_isOpen = false;

}

}

}

void BuyUI::UseCard(vector<Role\*> &roleList, Role\* role, Station\*\* station , Station\* rolestation, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building) {

if (\_cardIndex >= 0) { //有抽卡才做

\_card[\_cardIndex]->SetCardIsUsing(true);

\_card[\_cardIndex]->JudgeCard(roleList, role, station, mouseclick, posx, posy, map, building);

if (!\_card[\_cardIndex]->GetCardIsUsing()) { //卡片用完

\_isOK = true;

}

}

else {

\_isOK = true;

}

}

void BuyUI::LoadBitmap() {

\_UIBK.LoadBitmap(IDB\_UI);

\_title[0].LoadBitmap(IDB\_buytitle, RGB(0, 0, 100));

\_title[1].LoadBitmap(IDB\_buy, RGB(0, 0, 100));

\_title[2].LoadBitmap(IDB\_sell, RGB(0, 0, 100));

\_levelp[0][0].LoadBitmap(IDB\_level1, RGB(0, 0, 100));

\_levelp[0][1].LoadBitmap(IDB\_level2, RGB(0, 0, 100));

\_levelp[0][2].LoadBitmap(IDB\_level3, RGB(0, 0, 100));

\_levelp[1][0].LoadBitmap(IDB\_level1c, RGB(0, 0, 100));

\_levelp[1][1].LoadBitmap(IDB\_level2c, RGB(0, 0, 100));

\_levelp[1][2].LoadBitmap(IDB\_level3c, RGB(0, 0, 100));

\_dollar[0].LoadBitmap();

\_dollar[1].LoadBitmap();

\_dollar[2].LoadBitmap();

\_sell[0].LoadBitmap(IDB\_btnsell, RGB(0, 0, 100));

\_sell[1].LoadBitmap(IDB\_btnsellc, RGB(0, 0, 100));

\_dollar[3].LoadBitmap();

\_cardRandom.LoadBitmap(IDB\_cardRandom, RGB(0, 0, 100));

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

\_card[i]->LoadBitmap();

}

\_draw.LoadBitmap(IDB\_draw, RGB(0, 0, 100));

\_noDraw.LoadBitmap(IDB\_poor, RGB(0, 0, 100));

\_OK.LoadBitmap(IDB\_OK, RGB(0, 0, 100));

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

\_poor[i].LoadBitmap(IDB\_poor, RGB(0, 0, 100));

}

void BuyUI::OnShow(vector<Role\*> &roleList, Role\* role, Station\*\* station, Station\* rolestation) {

\_UIBK.SetTopLeft(50, 50);

\_title[0].SetTopLeft(80,80);

\_title[1].SetTopLeft(80,170);

\_title[2].SetTopLeft(80,270);

\_levelp[0][0].SetTopLeft(180, 180);

\_levelp[0][1].SetTopLeft(320, 180);

\_levelp[0][2].SetTopLeft(460, 180);

\_levelp[1][0].SetTopLeft(180, 180);

\_levelp[1][1].SetTopLeft(320, 180);

\_levelp[1][2].SetTopLeft(460, 180);

\_dollar[0].SetTopLeft(170, 240);

\_dollar[1].SetTopLeft(310, 240);

\_dollar[2].SetTopLeft(450, 240);

\_poor[0].SetTopLeft(180,180);

\_poor[1].SetTopLeft(320,180);

\_poor[2].SetTopLeft(460,180);

\_sell[0].SetTopLeft(180,280);

\_sell[1].SetTopLeft(180,280);

\_dollar[3].SetTopLeft(170,340);

\_poor[3].SetTopLeft(180,280);

\_cardRandom.SetTopLeft(90, 90);

\_draw.SetTopLeft(470, 90);

\_noDraw.SetTopLeft(470, 90);

\_OK.SetTopLeft(290, 360);

role->\_money.SetTopLeft(450,360);

if (\_isOpen ) {

if (!\_isUIOK) {

\_UIBK.ShowBitmap();

if (\_showHouseUI) {

\_title[0].ShowBitmap();

\_title[1].ShowBitmap();

\_title[2].ShowBitmap();

switch (\_level) {

case 0:

\_levelp[0][0].ShowBitmap();

\_levelp[0][1].ShowBitmap();

\_levelp[0][2].ShowBitmap();

break;

case 1:

\_levelp[1][0].ShowBitmap();

\_levelp[0][1].ShowBitmap();

\_levelp[0][2].ShowBitmap();

break;

case 2:

\_levelp[0][0].ShowBitmap();

\_levelp[1][1].ShowBitmap();

\_levelp[0][2].ShowBitmap();

break;

case 3:

\_levelp[0][0].ShowBitmap();

\_levelp[0][1].ShowBitmap();

\_levelp[1][2].ShowBitmap();

break;

default:

break;

}

\_dollar[0].ShowBitmap();

\_dollar[1].ShowBitmap();

\_dollar[2].ShowBitmap();

\_dollar[3].ShowBitmap();

role->\_money.ShowBitmap();

if (\_level == 0) {

\_sell[0].ShowBitmap();

}

else if (\_level == -1) {

\_sell[1].ShowBitmap();

}

if (rolestation->GetBuildingType() =="House")

{

if (role->GetMoney() < rolestation->house->GetPrice(3)) {

\_poor[2].ShowBitmap();

}

if (role->GetMoney() < rolestation->house->GetPrice(2)) {

\_poor[1].ShowBitmap();

}

if (role->GetMoney() < rolestation->house->GetPrice(1)) {

\_poor[0].ShowBitmap();

}

if (rolestation->house->GetOwner() != role->GetRoleNum()) {

\_poor[3].ShowBitmap();

}

}

\_OK.ShowBitmap();

}

else if (\_showStoreUI) {

\_OK.SetTopLeft(470, 350);

\_OK.ShowBitmap();

if (!\_isDraw)

{

\_cardRandom.ShowBitmap();

}

else

{

\_card[\_cardIndex]->OnShow(roleList, role, station, rolestation);

}

if (role->GetMoney() <= \_cardPrice)//錢不夠

{

\_noDraw.ShowBitmap();

}

else if (role->GetMoney() > \_cardPrice)//錢夠

{

\_draw.ShowBitmap();

}

}

}

}

if (\_isDraw) {

\_card[\_cardIndex]->OnShow(roleList, role, station, rolestation);

}

}

}

Card.h

#pragma once

#include "Dice.h"

#include "Station.h"

#include "GameMap.h"

#include "Role.h"

namespace game\_framework

{

enum class CardName {

toNext,

burnHouse,

steal

};

class Card

{

public:

Card();

Card(CardName cardKind);

int GetCardKind();

void SetCardIsUsing(bool flag);

bool GetCardIsUsing();

void JudgeCard(vector<Role\*> &roleList, Role\* role, Station\*\* station, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building);

void TossAgain(vector<Role\*> &roleList, Role\* role, Station\*\* station, Station\* rolestation, int direction, GameMap \*map, Building\*\* building); //往前後n站

void ChooseHouse(vector<Role\*> &roleList, Station\*\* station, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building);

void StealMoney(vector<Role\*> &roleList, Station\*\* station, Role\* role, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building);

void LoadBitmap();

void OnShow(vector<Role\*> &roleLList, Role\* role, Station\*\* station, Station\* rolestation);

private:

bool \_cardIsUsing; // true 使用中

bool \_targetConfirm;

int \_targetx, \_targety;

CMovingBitmap \_card ;

CAnimation \_carDown, \_carUp, \_carLeft, \_carRight;

CAnimation \_bomb, \_steal;

Dice \_dice[3];

CardName \_cardKind;

};

}

Card.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Card.h"

namespace game\_framework

{

Card::Card() {}

Card::Card(CardName cardKind): \_cardKind(cardKind), \_cardIsUsing(false), \_targetConfirm(false){}

int Card::GetCardKind() {

switch (\_cardKind) {

case CardName::toNext:

break;

case CardName::burnHouse:

break;

case CardName::steal:

break;

}

return 0;

}

void Card::SetCardIsUsing(bool flag) {

\_cardIsUsing = flag;

}

bool Card::GetCardIsUsing() {

return \_cardIsUsing;

}

void Card::JudgeCard(vector<Role\*> &roleList, Role\* role, Station\*\* station, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building) {

int nextstate=0;

switch (\_cardKind) {

case CardName::toNext:

nextstate = role->GetRoleState() + 1;

if (nextstate == 81) {

nextstate = 0;

}

TossAgain(roleList,role, station,station[nextstate], 1,map,building);

break;

case CardName::burnHouse:

ChooseHouse(roleList,station,mouseclick, posx, posy, map, building);

break;

case CardName::steal:

StealMoney(roleList, station, role, mouseclick, posx, posy, map, building);

break;

}

}

void Card::TossAgain(vector<Role\*> &roleList, Role\* role, Station\*\* station, Station\* rolestation, int direction, GameMap \*map, Building\*\* building){

if (\_dice[2].IsToss() && \_dice[0].IsToss()) {

role->SetShowRole(false);

\_dice[2].setXY(460, 420);

\_dice[2].TossDice(0);

}

if (\_dice[1].IsToss() && \_dice[0].IsToss()) {

\_dice[1].setXY(520, 420);

\_dice[1].TossDice(0);

}

if (\_dice[0].IsToss()) {

\_dice[0].setXY(580, 420);

\_dice[0].TossDice(0);

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

}

int camerax, cameray;

camerax = role->GetCameraX();

cameray = role->GetCameraY();

map->OnMove(camerax, cameray);

for (int i = 0; i < 81; i++)

station[i]->OnMove(camerax, cameray);

for (unsigned int i = 0; i < roleList.size(); i++) {

roleList[i]->OnMove(camerax, cameray);

}

for (int i = 0; i < 49; i++)

building[i]->OnMove(camerax, cameray);

\_carDown.OnMove();

\_carUp.OnMove();

\_carLeft.OnMove();

\_carRight.OnMove();

if (\_dice[2].GetDiceNumber() > 0){

if (role->SetRoleMove(rolestation, direction)) {

\_dice[2].SetDiceNumber(\_dice[2].GetDiceNumber() - 1);

}

}

else if (\_dice[1].GetDiceNumber()>0){

if (role->SetRoleMove(rolestation, direction)) {

\_dice[1].SetDiceNumber(\_dice[1].GetDiceNumber() - 1);

}

}

else if (\_dice[0].GetDiceNumber() > 0) {

if (role->SetRoleMove(rolestation, direction)) {

\_dice[0].SetDiceNumber(\_dice[0].GetDiceNumber() - 1);

}

}

else if (\_dice[0].GetDiceNumber() == 0){

role->SetShowRole(true);

\_dice[0].SetDiceIsToss(true);

\_dice[1].SetDiceIsToss(true);

\_dice[2].SetDiceIsToss(true);

\_cardIsUsing = false;

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

}

}

void Card::ChooseHouse(vector<Role\*> &roleList ,Station\*\* station, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building) {

if (!\_targetConfirm) {

map->OnMove();

for (int i = 0; i < 81; i++) {

station[i]->OnMove();

}

for (int i = 0; i < 49; i++) {

building[i]->OnMove();

}

for (unsigned int i = 0; i < roleList.size(); i++) {

roleList[i]->OnMove();

}

}

if (mouseclick&&!\_targetConfirm) {

for (int i = 0; i < 81; i++) {

if (station[i]->GetBuildingType() == "House") {

if (station[i]->house->GetX() + 50 >= posx && station[i]->house->GetX() <= posx && station[i]->house->GetY() + 50 >= posy && station[i]->house->GetY() <= posy) {

\_targetx = station[i]->house->GetX();

\_targety = station[i]->house->GetY();

station[i]->house->SetLevel(-1);

station[i]->house->SetOwner(4);

\_targetConfirm = true;

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

}

}

}

}

else if (\_targetConfirm) {

if (\_bomb.IsFinalBitmap()) {

\_bomb.Reset();

\_targetConfirm = false;

\_cardIsUsing = false;

}

\_bomb.OnMove();

}

}

void Card::StealMoney(vector<Role\*> &roleList, Station\*\* station, Role\* role, bool mouseclick, int posx, int posy, GameMap \*map, Building\*\* building) {

if (!\_targetConfirm) {

map->OnMove();

for (int i = 0; i < 81; i++) {

station[i]->OnMove();

}

for (int i = 0; i < 49; i++) {

building[i]->OnMove();

}

for (unsigned int i = 0; i < roleList.size(); i++) {

roleList[i]->OnMove();

}

}

if (mouseclick && !\_targetConfirm) {

for (unsigned int i = 0; i < roleList.size(); i++) {

if (roleList[i]->GetX()+50 >= posx && roleList[i]->GetX() <= posx && roleList[i]->GetY() + 10 >= posy && roleList[i]->GetY()-50 <= posy) {

\_targetx = roleList[i]->GetX()-80;

\_targety = roleList[i]->GetY();

roleList[i]->AddMoney(-2000);

role->AddMoney(2000);

\_targetConfirm = true;CAudio::

Instance()->Play(AUDIO\_STEAL, true);

}

}

}

else if (\_targetConfirm) {

if (\_steal.IsFinalBitmap()) {

\_steal.Reset();

\_targetConfirm = false;

\_cardIsUsing = false;

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

}

\_targetx += 5;

\_steal.OnMove();

}

}

void Card::LoadBitmap() {

switch (\_cardKind) {

case CardName::toNext:

\_card.LoadBitmap(IDB\_carCard, RGB(0, 0, 100));

\_dice[0].LoadBitmap();

\_dice[1].LoadBitmap();

\_dice[2].LoadBitmap();

\_carDown.AddBitmap(IDB\_Car\_1, RGB(255, 255, 255));

\_carDown.AddBitmap(IDB\_Car\_2, RGB(255, 255, 255));

\_carDown.AddBitmap(IDB\_Car\_3, RGB(255, 255, 255));

\_carDown.AddBitmap(IDB\_Car\_4, RGB(255, 255, 255));

\_carLeft.AddBitmap(IDB\_Car\_5, RGB(255, 255, 255));

\_carLeft.AddBitmap(IDB\_Car\_6, RGB(255, 255, 255));

\_carLeft.AddBitmap(IDB\_Car\_7, RGB(255, 255, 255));

\_carLeft.AddBitmap(IDB\_Car\_8, RGB(255, 255, 255));

\_carRight.AddBitmap(IDB\_Car\_9, RGB(255, 255, 255));

\_carRight.AddBitmap(IDB\_Car\_10, RGB(255, 255, 255));

\_carRight.AddBitmap(IDB\_Car\_11, RGB(255, 255, 255));

\_carRight.AddBitmap(IDB\_Car\_12, RGB(255, 255, 255));

\_carUp.AddBitmap(IDB\_Car\_13, RGB(255, 255, 255));

\_carUp.AddBitmap(IDB\_Car\_14, RGB(255, 255, 255));

\_carUp.AddBitmap(IDB\_Car\_15, RGB(255, 255, 255));

\_carUp.AddBitmap(IDB\_Car\_16, RGB(255, 255, 255));

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

break;

case CardName::burnHouse:

\_card.LoadBitmap(IDB\_bombCard, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb1, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb2, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb3, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb4, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb5, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb6, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb7, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb8, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb9, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb10, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb11, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb12, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb13, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb14, RGB(0, 0, 100));

\_bomb.AddBitmap(IDB\_bomb15, RGB(0, 0, 100));

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

break;

case CardName::steal:

\_card.LoadBitmap(IDB\_stealCard, RGB(0, 0, 100));

\_steal.AddBitmap(IDB\_steal1, RGB(0, 0, 100));

\_steal.AddBitmap(IDB\_steal2, RGB(0, 0, 100));

\_steal.AddBitmap(IDB\_steal3, RGB(0, 0, 100));

\_steal.AddBitmap(IDB\_steal4, RGB(0, 0, 100));

\_steal.AddBitmap(IDB\_steal5, RGB(0, 0, 100));

\_steal.AddBitmap(IDB\_steal6, RGB(0, 0, 100));

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

break;

}

}

void Card::OnShow(vector<Role\*> &roleLList, Role\* role, Station\*\* station, Station\* rolestation) {

if (\_cardIsUsing) {

if (\_cardKind == CardName::toNext) {

\_dice[0].OnShow();

\_dice[1].OnShow();

\_dice[2].OnShow();

int rolestate=role->GetRoleState() + 1;

if (rolestate == 81) {

rolestate = 0;

}

if (role->GetY() < station[rolestate]->GetY()) {

\_carDown.SetTopLeft(role->GetX(), role->GetY() - 45);

\_carDown.OnShow();

}

else if (role->GetY() > station[rolestate]->GetY()) {

\_carUp.SetTopLeft(role->GetX(), role->GetY() - 45);

\_carUp.OnShow();

}

else if (role->GetX() > station[rolestate]->GetX()) {

\_carLeft.SetTopLeft(role->GetX(), role->GetY() - 45);

\_carLeft.OnShow();

}

else if (role->GetX() < station[rolestate]->GetX()) {

\_carRight.SetTopLeft(role->GetX(), role->GetY() - 45);

\_carRight.OnShow();

}

else {

\_carDown.SetTopLeft(role->GetX(), role->GetY() - 45);

\_carDown.OnShow();

}

}

else if (\_cardKind == CardName::burnHouse) {

if (\_targetConfirm) {

\_bomb.SetTopLeft(\_targetx, \_targety - 30);

\_bomb.OnShow();

}

}

else if (\_cardKind == CardName::steal) {

if (\_targetConfirm) {

\_steal.SetTopLeft(\_targetx, \_targety - 40);

\_steal.OnShow();

}

}

}

else {

\_card.SetTopLeft(90, 90);

\_card.ShowBitmap();

}

}

}

Integer.h

#pragma once

namespace game\_framework

{

class Integer {

public:

Integer(int = 5); // default 5 digits

void Add(int n); // 增加整數值

int GetInteger(); // 回傳整數值

void LoadBitmap(); // 載入0..9及負號之圖形

void SetInteger(int); // 設定整數值

void SetTopLeft(int, int); // 將動畫的左上角座標移至 (x,y)

void ShowBitmap(); // 將動畫貼到螢幕

private:

const int NUMDIGITS; // 共顯示NUMDIGITS個位數

CMovingBitmap digit[11]; // 儲存0..9及負號之圖形(bitmap)

int x, y; // 顯示的座標

int n; // 整數值

bool isBmpLoaded; // 是否已經載入圖形

};

}

Integer.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Integer.h"

namespace game\_framework

{

Integer::Integer(int digits)

: NUMDIGITS(digits)

{

isBmpLoaded = false;

}

void Integer::Add(int x)

{

n += x;

}

int Integer::GetInteger()

{

return n;

}

void Integer::LoadBitmap()

{

int d[11] = { IDB\_000,IDB\_001,IDB\_002,IDB\_003,IDB\_004,IDB\_005,IDB\_006,IDB\_007,IDB\_008,IDB\_009,IDB\_MINUS };

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

digit[i].LoadBitmap(d[i]);

}

void Integer::SetInteger(int i)

{

n = i;

}

void Integer::SetTopLeft(int nx, int ny) // 將動畫的左上角座標移至 (x,y)

{

x = nx; y = ny;

}

void Integer::ShowBitmap()

{

int nx; // 待顯示位數的 x 座標

int MSB; // 最左邊(含符號)的位數的數值

if (n >= 0) {

MSB = n;

nx = x + digit[0].Width()\*(NUMDIGITS - 1);

}

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

int d = MSB % 10;

MSB /= 10;

digit[d].SetTopLeft(nx, y);

digit[d].ShowBitmap();

nx -= digit[d].Width();

}

}

}

GameMap.h

#pragma once

namespace game\_framework {

class GameMap

{

public:

GameMap();

int GetX1();

int GetY1();

void Initialize();

void LoadBitmap();

virtual void OnMove();

virtual void OnMove(int camerax, int cameray);

void OnShow(); // 將擦子圖形貼到畫面

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

void SetXY(int nx, int ny); // 設定擦子左上角座標

protected:

CMovingBitmap zero,one,two,three,four,five,six,seven,eight,nine,ten,eleven,twelve,thirsteen,fourteen;

protected:

CAnimation animation; // 擦子的動畫

int \_referenceX, \_referenceY;// 擦子左上角座標

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

int map[5][3];

const int MW, MH;

};

}

GameMap.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "GameMap.h"

namespace game\_framework {

GameMap::GameMap()

:\_referenceX(0), \_referenceY(0), MW(640), MH(480)

{

Initialize();

}

int GameMap::GetX1()

{

return \_referenceX;

}

int GameMap::GetY1()

{

return \_referenceY;

}

void GameMap::Initialize()

{

int map\_init[5][3] = { {0,1,2},

{3,4,5},

{6,7,8},

{9,10,11},

{12,13,14} };

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

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

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

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;

}

void GameMap::LoadBitmap()

{

zero.LoadBitmap(IDB\_zero);

one.LoadBitmap(IDB\_one);

two.LoadBitmap(IDB\_two);

three.LoadBitmap(IDB\_three);

four.LoadBitmap(IDB\_four);

five.LoadBitmap(IDB\_five);

six.LoadBitmap(IDB\_six);

seven.LoadBitmap(IDB\_seven);

eight.LoadBitmap(IDB\_eight);

nine.LoadBitmap(IDB\_nine);

ten.LoadBitmap(IDB\_ten);

eleven.LoadBitmap(IDB\_eleven);

twelve.LoadBitmap(IDB\_twelve);

thirsteen.LoadBitmapA(IDB\_thirsteen);

fourteen.LoadBitmapA(IDB\_fourteen);

}

void GameMap::OnMove()

{

const int STEP\_SIZE = 10;

if(\_referenceX + STEP\_SIZE <= 0){

if (isMovingLeft)

\_referenceX += STEP\_SIZE;

}

if(\_referenceX - STEP\_SIZE >= -1280){

if (isMovingRight)

\_referenceX -= STEP\_SIZE;

}

if(\_referenceY + STEP\_SIZE <= 0){

if (isMovingUp)

\_referenceY += STEP\_SIZE;

}

if(\_referenceY - STEP\_SIZE >= -1920){

if (isMovingDown)

\_referenceY -= STEP\_SIZE;

}

}

void GameMap::OnMove(int camerax, int cameray) {

if (\_referenceX + camerax <= 0 && \_referenceX + camerax >= -1280) {

\_referenceX += camerax;

}

else if (camerax > 0) {

\_referenceX = 0;

}

else if (camerax < 0) {

\_referenceX = -1280;

}

if (\_referenceY + cameray <= 0 && \_referenceY + cameray >= -1920) {

\_referenceY += cameray;

}

else if (cameray > 0) {

\_referenceY = 0;

}

else if (cameray < 0) {

\_referenceY = -1920;

}

}

void GameMap::SetMovingDown(bool flag)

{

isMovingDown = flag;

}

void GameMap::SetMovingLeft(bool flag)

{

isMovingLeft = flag;

}

void GameMap::SetMovingRight(bool flag)

{

isMovingRight = flag;

}

void GameMap::SetMovingUp(bool flag)

{

isMovingUp = flag;

}

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

{

\_referenceX = nx; \_referenceY = ny;

}

void GameMap::OnShow()

{

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

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

{

if (\_referenceX + (MW\*i) < 641 && \_referenceX + (MW\*i) > -641 && \_referenceY + (MH\*j) < 481 && \_referenceY + (MH\*j) > -481) {

switch (map[j][i])

{

case 0:

zero.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

zero.ShowBitmap();

break;

case 1:

one.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

one.ShowBitmap();

break;

case 2:

two.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));

two.ShowBitmap();

break;

case 3:

three.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

three.ShowBitmap();

break;

case 4:

four.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

four.ShowBitmap();

break;

case 5:

five.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

five.ShowBitmap();

break;

case 6:

six.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

six.ShowBitmap();

break;

case 7:

seven.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

seven.ShowBitmap();

break;

case 8:

eight.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

eight.ShowBitmap();

break;

case 9:

nine.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

nine.ShowBitmap();

break;

case 10:

ten.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

ten.ShowBitmap();

break;

case 11:

eleven.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

eleven.ShowBitmap();

break;

case 12:

twelve.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

twelve.ShowBitmap();

break;

case 13:

thirsteen.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

thirsteen.ShowBitmap();

break;

case 14:

fourteen.SetTopLeft(\_referenceX + (MW\*i), \_referenceY + (MH\*j));//

fourteen.ShowBitmap();

break;

default:

ASSERT(0);

}

}

}

}

}

}

StartManu.h

#pragma once

namespace game\_framework {

class StartMenu {

public:

StartMenu();

bool GetDone();

void SetDone(bool flag);

unsigned int GetPlayerAmount();

void OnChoose(int posx,int posy, bool mouseclick);

void OnShow();

void LoadBitmap();

private:

int \_playerAmount;

CMovingBitmap \_playerChoosed[2][3];

bool \_isDone;//true為已完成

bool \_choosePlayerAmount;

};

}

StartManu.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "StartMenu.h"

namespace game\_framework {

StartMenu::StartMenu() : \_playerAmount(2) , \_isDone(false) {

}

bool StartMenu::GetDone() {

return \_isDone;

}

void StartMenu::SetDone(bool flag) {

\_isDone = flag;

}

unsigned int StartMenu::GetPlayerAmount() {

return \_playerAmount;

}

void StartMenu::OnChoose(int posx, int posy, bool mouseclick) {

if (posx > 85 && posx < 185 && posy > 190 && posy < 290 && mouseclick)

\_playerAmount = 2;

else if (posx > 270 && posx < 370 && posy > 190 && posy < 290 && mouseclick)

\_playerAmount = 3;

else if (posx > 455 && posx < 555 && posy > 190 && posy < 290 && mouseclick)

\_playerAmount = 4;

}

void StartMenu::LoadBitmap() {

\_playerChoosed[0][0].LoadBitmap(IDB\_P2);

\_playerChoosed[0][1].LoadBitmap(IDB\_P3);

\_playerChoosed[0][2].LoadBitmap(IDB\_P4);

\_playerChoosed[1][0].LoadBitmap(IDB\_CP2);

\_playerChoosed[1][1].LoadBitmap(IDB\_CP3);

\_playerChoosed[1][2].LoadBitmap(IDB\_CP4);

}

void StartMenu::OnShow() {

\_playerChoosed[0][0].SetTopLeft(85, 190);

\_playerChoosed[0][1].SetTopLeft(270, 190);

\_playerChoosed[0][2].SetTopLeft(455, 190);

\_playerChoosed[1][0].SetTopLeft(85, 190);

\_playerChoosed[1][1].SetTopLeft(270, 190);

\_playerChoosed[1][2].SetTopLeft(455, 190);

switch (\_playerAmount) {

case 2:

\_playerChoosed[1][0].ShowBitmap();

\_playerChoosed[0][1].ShowBitmap();

\_playerChoosed[0][2].ShowBitmap();

break;

case 3:

\_playerChoosed[0][0].ShowBitmap();

\_playerChoosed[1][1].ShowBitmap();

\_playerChoosed[0][2].ShowBitmap();

break;

case 4:

\_playerChoosed[0][0].ShowBitmap();

\_playerChoosed[0][1].ShowBitmap();

\_playerChoosed[1][2].ShowBitmap();

break;

default:

break;

}

}

}