

國立台北科技大學

National Taipei University of Technology

99 學年度

物件導向程式設計實習

Oriented-Object Programming Labs



指導教授：陳偉凱

組別：g20

組員：98590316 張竣翔、98590342 李旻憲

中華民國 100 年 6 月 17 日

目錄

一、簡介.....	2
1.動機	2
2.分工	2
二、遊戲說明	3
1.簡介：	3
2.遊戲圖形：	3
3.遊戲音效	7
三、程式設計	8
1.程式架構	8
2.程式類別	12
3.程式技術	24
四、結語.....	25
1.問題與解決方法	25
2.時間表(Time Log).....	27
3.貢獻比例	30
4.檢核表	30
5.收穫	31
6.心得	32
7.對於本課程的建議	33

一、簡介

1.動機

我們會選擇這個主題的靈感來至手機遊戲有個寶物盜賊的遊戲(Treasure Rub)，這個遊戲的主角是到每一個主人的家裡偷取寶物，因為覺得這款遊戲刺激有趣，故想試著做出類似此款遊戲，所以我們才選定寶物獵人當作我們的主題。

2.分工

在這次的遊戲專題設計上我們並沒有很詳細的規劃出我們兩個人所必須完成的事情取而代之的是互相協助與幫忙，每周互相將自己未完成以及完成的部分提報給對方知道並且了解運作原理以及相互討論來完成此次的期末專題。

品項名稱	負責人	品項名稱	負責人
怪物自動尋找路徑	張竣翔	建立遊戲整體規劃與所需物件列表	李旻憲
怪物可自動開關門	張竣翔	地圖設計(全)	李旻憲
顯示地圖迷宮撰寫	張竣翔	尋找與設計遊戲圖檔(全)	李旻憲
腳色移動視窗位置	張竣翔	尋找與設計遊戲音效(全)	李旻憲
怪物追蹤主角撰寫	張竣翔	計分機制	李旻憲
加入 KEY 及對應門撰寫	張竣翔	計時機制	李旻憲
加入寶物及取得撰寫	張竣翔	修改 CInteger 使其能提供新方法	李旻憲
		儲存遊戲紀錄之檔案 I/O(全)	李旻憲
		遊戲音效/圖檔程式碼撰寫(全)	李旻憲
		閃爍及遊戲說明設計撰寫	李旻憲
		報告撰寫	李旻憲
		開始/過關/結束/失敗狀態及方法(全)	李旻憲

二、遊戲說明

1.簡介：

這款遊戲是讓玩家操縱一名寶物獵人，去迷宮中將所有的寶物通通偷出來，迷宮中有數位怪物在巡邏，遊戲的宗旨就是不能讓怪物抓到並且偷得所有的寶物後找到出口離開，並且有時限制，需要在時限內完成，不然也是任務失敗。本遊戲總共有兩關，完成一關後會自動跳至下一關，在主選單玩家可以直接選取想開始的關卡，進行遊戲。

2.遊戲圖形：



↑ 遊戲選單



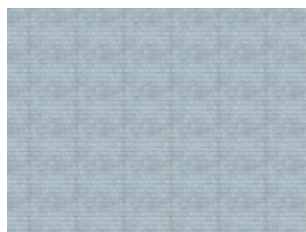
↑ 選關畫面



↑ 關於遊戲畫面



↑ 遊戲說明



↑ 遊戲的地板



↑ 遊戲結束畫面



↑ 過關畫面



↑ 破台畫面



↑ 紅屏



↑ 黃色的鑰匙



↑ 綠色鑰匙



↑ 紫色鑰匙



↑ 紅色鑰匙



























↑ 黃色上鎖橫門



↑ 黃色上鎖直門



↑ 綠色上鎖橫門

			
↑ 綠色上鎖直門	↑ 紫色上鎖橫門	↑ 紫色上鎖直門	↑ 紅色上鎖橫門
			
↑ 紅色上鎖直門	↑ 無上鎖橫門	↑ 無上鎖直門	↑ 寶物
			
↑ 分數用數字 0	↑ 分數用數字 1	↑ 分數用數字 2	↑ 分數用數字 3
			
↑ 分數用數字 4	↑ 分數用數字 5	↑ 分數用數字 6	↑ 分數用數字 7
			
↑ 分數用數字 8	↑ 分數用數字 9	↑ 過關剩餘時間數字 0	↑ 過關剩餘時間數字 1
			
↑ 過關剩餘時間數字 2	↑ 過關剩餘時間數字 3	↑ 過關剩餘時間數字 4	↑ 過關剩餘時間數字 5

6

↑ 過關剩餘時間數字
6

0

↑ 倒數時間數字 0

7

↑ 過關剩餘時間數字
7

1

↑ 倒數時間數字 1

8

↑ 過關剩餘時間數字
8

2

↑ 倒數時間數字 2

9

↑ 過關剩餘時間數字
9

3

↑ 倒數時間數字 3

4

↑ 倒數時間數字 4

5

↑ 倒數時間數字 5

6

↑ 倒數時間數字 6

7

↑ 倒數時間數字 7

8

↑ 倒數時間數字 8

9

↑ 倒數時間數字 9



↑ 倒數時間數字 10



↑ 牆壁



↑ ICON

LOADING

↑ 讀取中



↑ 作弊模式開啟圖片

遊戲暫停中...

↑ 遊戲暫停圖片



↑ 怪物向下走圖片 1



↑ 怪物向下走圖片 2



↑ 怪物向下走圖片 3



↑ 怪物向下走圖片 4



↑ 怪物向左走圖片 1



↑ 怪物向左走圖片 2



↑ 怪物向左走圖片 3



↑ 怪物向左走圖片 4



↑ 怪物向右走圖片 1



↑ 怪物向右走圖片 2



↑ 怪物向右走圖片 3



↑ 怪物向右走圖片 4



↑ 怪物向上走圖片 1



↑ 怪物向上走圖片 2



↑ 怪物向上走圖片 3



↑ 怪物向上走圖片 4



↑ 小偷向下走圖片 1



↑ 小偷向下走圖片 2



↑ 小偷向下走圖片 3



↑ 小偷向下走圖片 4



↑ 小偷向左走圖片 1



↑ 小偷向左走圖片 2



↑ 小偷向左走圖片 3



↑ 小偷向左走圖片 4



↑ 小偷向右走圖片 1



↑ 小偷向右走圖片 2



↑ 小偷向右走圖片 3



↑ 小偷向右走圖片 4



↑ 小偷向上走圖片 1



↑ 小偷向上走圖片 2



↑ 小偷向上走圖片 3



↑ 小偷向上走圖片 4

3.遊戲音效

使用時機	音效檔案	擷取來源	備註
遊戲開頭	STGBbgs.mp3	祖馬遊戲	
開始選單選擇	SELECT.wav	仙劍奇俠傳 4	
開門時候	DOOR.wav	仙劍奇俠傳 4	
撿到 KEY	KEY.mp3	仙劍奇俠傳 4	
撿到寶物	PICK.wav	仙劍奇俠傳 4	
遊戲過關	SCUESS.wav	仙劍奇俠傳 4	
遊戲失敗	FAILED.mp3	仙劍奇俠傳 4	
第一關背景音樂	Map1.mp3	仙劍奇俠傳 4	
第二關背景音樂	Map2.mp3	軒轅劍- 雲之遙	

三、程式設計

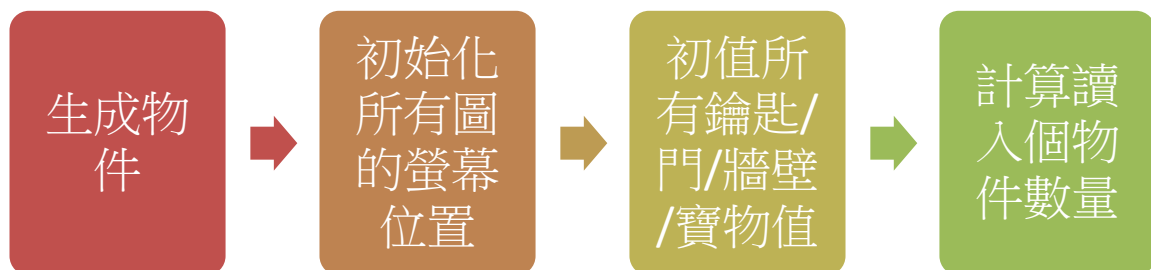
1. 程式架構

我們的遊戲是利用老師的 GameFrameWork 所撰寫成，原先老師提供的只有三個 State，我們將每個關卡及過關/失敗/破關/選關頁面都寫成一個 State，所以每完成一關或者失敗，皆會跳至相對應的 State。

* 遊戲關卡物件一覽

牆壁	寶物	鑰匙	主角	怪物
地圖	鑰匙	移動點	門	閃爍紅屏
剩餘時間	幫助頁面	第一關	第二關	過關
破關	選關卡	開始頁面	失敗	

* 地圖物件流程



在地圖這個物件中，只要地圖一被宣告，就會直接先跑第一次的初始化，以利於後面參數的使用，初始化依序將螢幕顯示的最右上角位置設為地圖座標的(0,0)，並且將鑰匙數、牆壁數、寶物數、門數，歸為 0 後讀入的圖檔後，將其上述的物件數量，計算出來。而我們的地圖檔是使用 TXT 檔的方式去完成。

*txt 檔設定代號一覽

空白	一般可走路徑	h	普通橫向門
o	牆壁	v	普通直向門
t	寶物	Y	黃色直向上鎖門
e	出口	Z	黃色橫向上鎖門
y	黃色鑰匙	R	紅色上直向鎖門
r	紅色鑰匙	S	紅色橫向上鎖門
g	綠色鑰匙	Q	紫色橫向上鎖門
p	紫色鑰匙	G	綠色直向上鎖門
P	紫色直向上鎖門	H	綠色橫向上鎖門

此函式還有提供，回傳地圖上的物品類別、回傳門的類別、設定為空白、設定門、計算與左上角 x 差距函式、計算與左上角 y 差距還式，回傳參數，與設定左上角做邊的功能。

在牆壁這個物件中，提供讀取牆壁圖檔、顯示障礙、設定座標三個涵式。

而在寶物物件中，提供讀取圖檔、設定座標、設定大小、設定狀態、辨別是否撞擊到主角、回傳狀態、取得高度、取得寬度、取得座標等涵式，我們的寶物設定是佔 2x2 格，並且其位置是紀錄在地圖檔中。

在鑰匙方面，與寶物非常的類似，但他多了回傳其鑰匙的顏色。門的物件也非常的類似，在碰撞的部分，只要怪物或者人物與門有重疊，就會判定碰撞，反之就是沒有，門有分上鎖的門與普通門，怪物可開所有的門，主角要拿到鑰匙才能開非棕色的門，而判定能不能開，不是門的類別所提供。

小偷類別，和前者一樣提供讀圖的函式，還有顯示圖片、設定是否往上、下、左、右移、回傳座標、設定座標、重設步伐等涵式，不公開的函式有控制腳步(依其腳步換圖)、判別是否可行走兩個函式，人物沒拿到相對應顏色鑰匙，是不能通過也是在這邊所判別。

怪物類別，提供重設步伐，回傳座標，設定座標，設定一般移動路徑，取得是否為追蹤模式，不公開的函式為找尋路徑、控制腳步(依其腳步換圖)、判別是否可行走、判定小偷是否在視線範圍內。

閃爍紅屏類別中，裡面自己有個 COUNTER，當人物出現在小偷的視線範圍內，會將閃爍紅屏設為 isFlash 設為 True，透過 CLASS 中的計數器，控制紅屏的顯示與否。當離開時，會將 isFlash 歸回 False 並且將計數器歸零。

說明頁面物件，控制說明頁面的顯示與否。

在開始執行遊戲時，地圖在被宣告的同時，也完成第一次的初始化後，開始讀取圖片，依照門、寶物、鑰匙、怪物、牆壁、數量，要其相當的記憶體後，並且將門、寶物、鑰匙、牆壁的座標設定好後，將地圖中記錄門、寶物、鑰匙的地方設為空白，表可以行走。

將所有都 OnInit 跑完後，程式進入 CGameStateInit 表單畫面，若使用者點選重新遊戲，將跳至 CGameStateRun 若玩家點選選擇關卡，進入 CGameStateSelect，若點選關於程式，則會顯示關於程式的圖片後，當使用在再次點選空白鍵，則圖片消失，回到原畫面，點離開則程式結束。

當遊戲一進入 CGamaStateRun 則會先在跑一次初始化，將所有參數設定好，怪物的路徑也是在此輸入，初始化完成後即開始遊戲，一開始會顯示說明畫面，點任意建就會跳離，說明畫面顯示時，因為我們加入說明畫面是顯示時，及馬上 RETURN 不繼續執行，故在看說明時間是不會倒數怪物也不會移動。怪物的移動當在非追蹤模式時，一樣是使用路徑追蹤去找路徑，而移動點，是照其陣列，移動到一定點後，在往下一個點移動。當人物進入到怪物的視線範圍內時，怪物會進入追蹤模式，追蹤模式是先取得主角位置後，先移動到其位置後，再次取得主角位置值，在移動到那個點，並且持續記錄移動過後的位置，直到主角離開怪物上下左右 1 5 格外，或者被抓到跳至 CGameStateOver。當主角離開上述範圍後，一照剛剛的移動紀錄，回到最原先尚未啟動追蹤的位置，若尚未回到啟動追蹤的點時，主角又觸發追蹤條件，則一樣會啟動追蹤，並且一樣會加在紀錄的路徑中，不會發生回不到原本地的狀況。在怪物的視線上，若作弊模式開啟，怪物會形同瞎子，看不到主角，若是在追蹤狀態下，則會馬上放棄追蹤。若時間到，也是會進入 CGameStateOver，若主角取得寶物後，並且抵達逃脫點，先會進入

CGameStatePass 的過關畫面，當使用者按下空白鍵後，則會跳至 CGameStateRun2，進行第二關。第二關與第一關相同，只是地圖檔與怪物路徑不同，其運作模式架構同第一關，一樣任務失敗會進入 CGamerStateOver，過關一樣進入 CGameStatePass，因我們只有設計兩關，此時進入 CGameStatePass 後，按下空白鍵，會跳進 CGameStateWin，顯示出破台畫面，此時在按下空白鍵後，則會跳回 CGameStateInit，回到選單的畫面。在進入 CGamerStateOver，一樣會顯示 GAMEOVER 的畫面，並且會倒數，自動進入 CGameStateInit，或者按下空白鍵後，直接進入 CGameStateInit。若玩家選擇選擇關卡進入 CGameStateSelect 後，按入相對應的關卡，會進入相對的 GameState，遊戲結束或過關都如同上面所敘，並無只玩單關模式。

2.程式類別

說明：本頁為自訂類別一欄表與說明，下一頁 cpp 檔中可看到所有詳細方法用法。

所在檔案 mygame.h (總行數：543)

類別名稱	功用	說明	總行數
CMap	地圖	地圖	29
Obstacle	障礙物	建立記憶體中障礙物位置	10
MovingNode	移動點	A* 專用的座標點與一般預設座標點	16
CintTime	剩餘時間	CInteger 並轉換時間為分秒制後顯示	14
Door	門	設定門的定義及碰撞偵測/開關/位置定義	21
Key	鑰匙	設定鑰匙位置/顏色/碰撞/大小	19
Thief	盜賊	定義盜賊位置與行走/鍵盤偵測	35
Monster	怪物	怪物主體/自動移動/追蹤/返回	38
Treasure	寶物	寶物位置/大小/圖檔/碰撞	21
redscreen	閃爍紅屏	可設定是否閃爍紅屏表被追殺	14
help	遊戲說明	於第一關時顯示玩法	13
CGameStateSelect	選擇關卡	可以直接選關不必從頭開始玩	12
CGameStatePASS	過關頁面	秀出此關的剩餘時間與得分	34
CGameStateWIN	破關畫面	全部破關後顯示遊戲贏家	13

說明：本頁為 cpp 檔內所有自訂類別方法介紹。

所在檔案 mygame.cpp (總行數：2968)

類別名稱	包含方法	總行數
CMap	<pre> public: CMap(const char* fn); void Initialize(const char* fn); void LoadMap(); void OnShow(); void MoveMap(char direction); int Isempy(int x,int y); int DetialDoor(int x,int y); int computsx(int x); int computsy(int y); int Getcoutobstacle(); int Getcounttreasure(); int Getcountdoors(); int Getcountkeys(); char KeyColor(int x,int y); void Setsxsy(int x,int y); void toempty(int x,int y); void setdoor(int x,int y,char type); private: CMovingBitmap background; char map[160][120]; int sx,sy; </pre>	150

	<pre> int countobstacle; int counttreasures; int countdoors; int countkeys; void readmapfromfile(const char* fn); </pre>	
Obstacle	<pre> public: Obstacle(); void OnShow(CMap* gamemap); void Load(); void setxy(int setx, int sety); private: int x,y; CMovingBitmap bmp; </pre>	16
MovingNode	<pre> public: MovingNode(int sx, int sy); int Getx(); int Gety(); int Getg(); int Getf(); MovingNode* GetParent(); void caculateparameter(const int sx, const int sy,const int tx,const int ty); void setparent(MovingNode* settarget); private: int x,y; </pre>	28

	<pre> int f,g,h; MovingNode* parent; bool isInList; </pre>	
CintTime	<pre> public: CintTime(); void Initialize(); void SetTime(int); int GetTime(); void LoadBitmap(); void OnShow(); private: void calTime(); int min,sec; CInteger Cint; </pre>	43
Door	<pre> public: Door(); void Initialize(CMap* gamemap); void LoadDoor(); void setClose(bool closeflag,CMap* gamemap); bool HitObject(int sx,int sy); void setDoor(int sx,int sy,char type,char color); bool GetisClose(); void OnShow(CMap* gamemap); char Getdoortype(); </pre>	166

	<pre> char Getdoorcolor(); private: bool isClose; bool isVertical; int x,y; static const int doorsize = 8; CMovingBitmap doorpicture; char doortype; char doorcolor; </pre>	
Key	<pre> public: Key(); void Initialize(); void Onshow(CMap* gamemap); void setKey(int sx,int sy,char color); void LoadKey(); char getColor(); bool hitThief(int sx,int sy); bool getisOnMap(); void setOnMap(bool flag); private: int x,y; static const int width =2; static const int high =2; bool isOnMap; char keycolor; </pre>	51

	CMovingBitmap keypicture;	
Thief	<pre> public: Thief(); void Initialize(); void LoadBitmap(); void OnMove(CMap* gamemap,Door* door,Key* key); void OnShow(CMap* gamemap); void SetMovingDown(bool flag); void SetMovingLeft(bool flag); void SetMovingRight(bool flag); void SetMovingUp(bool flag); void RandSetXY(CMap* gamemap); void ChangeCounter(int counter); void reststep(); int Getx(); int Gety(); protected: CMovingBitmap rogue[4][4]; CMovingBitmap *play; int step; int x,y; bool isMovingDown; bool isMovingLeft; bool isMovingRight; </pre>	229

	<pre> bool isMovingUp; void contralstep(int direction); bool checkempty(char direction,CMap* Gamemap,Key *key,Door *door); CMovingBitmap Tcounter[10]; bool isGetallTreasure; vector <Key*> Getkeys; </pre>	
Monster	<pre> public: Monster(); void Initialize(); void LoadBitmap(); void OnMove(CMap* gamemap,Thief* thief,Door* door,bool cheatMode); void OnShow(CMap* gamemap); void reststep(); int Getx(); int Gety(); void SetXY(int x, int y); void Setcommenpath(vector<MovingNode*> &scriptpath); bool GetisTrace(); protected: CMovingBitmap guide[4][4]; CMovingBitmap *monster_1; int mstep; </pre>	757

	<pre> int gx,gy; static const int sightrange = 10; unsigned int nowstep; char nowdirection; vector<MovingNode*> path; vector<MovingNode*> returnpath; vector<MovingNode*> commenpath; void findpath(const int tx, const int ty,CMap* gamemap); bool isMoving; bool isTrace; bool isBack; bool isComeback; void contralstep(int direction); bool checkempty(int x, int y,char direction,CMap* Gamemap); bool looksight(Thief* thief,CMap* gamemap,bool cheatmode); int mcounter; unsigned int commenwalkstep; bool opendoor; Door *thedoor; bool wait; int waitcounter; </pre>	
Treasure	public:	55

	<pre> Treasure(); void Initialize(); void LoadBitmap(); void OnShow(CMap* gamemap); void SetXY(int sx,int sy); void SetSize(int h, int w); void Setlife(bool flag); bool HitThief(Thief* thief); bool Isalive(); int Getheight(); int Getwidth(); int Getx(); int Gety(); private: int x,y; int height,width; CMovingBitmap treasure_bitmap; bool islife; </pre>	
redscreen	<pre> public: redscreen(); void Initialize(); void LoadBitmap(); void OnShow(); void SetisFlash(bool flasflag); bool GetisFlash(); </pre>	28

	<pre>private: int counter; bool isFlash; CMovingBitmap redscreen_bitmap;</pre>	
help	<pre>public: help(); void Initialize(); void LoadBitmap(); void OnShow(); void SetOk(bool flag); bool GetState(); private: bool flag; CMovingBitmap help_bitmap;</pre>	22
CGameStateSelect	<pre>public: CGameStateSelect(CGame *g); void OnInit(); void OnBeginState(); void OnKeyUp(UINT, UINT, UINT); protected: void OnShow(private: CMovingBitmap Selec; CMovingBitmap ExMouse;</pre>	8
CGameStatePASS	<pre>public:</pre>	205

	<pre> CGameStatePASS(CGame *g); void OnInit(); protected: void OnBeginState(); void CalST(); void OnKeyUp(UINT, UINT, UINT); void OnShow(); void readfromfile(); private: int counter; int min,sec; int time; int score; int min1,min2; int sec1,sec2; int stage;//關卡位置 CMovingBitmap PASS; CMovingBitmap t1; CMovingBitmap t2; CMovingBitmap t3; CMovingBitmap t4; CMovingBitmap t5; CMovingBitmap t6; CMovingBitmap t7; CMovingBitmap t8; CMovingBitmap t9; CMovingBitmap t0; </pre>	
--	--	--

	CIntegerST CiST; bool mu;	
CGameStateWIN	public: CGameStateWIN(CGame *g); void OnBeginState(); void OnInit(); protected: void OnKeyUp(UINT, UINT, UINT); void OnMove(); void OnShow(); private: int counter; CMovingBitmap WIN;	31

3.程式技術

怪物自動路徑偵測

路徑偵測我們是使用 A^* 的方式來做路徑找尋的動作，在 A^* 中，移動點有 F、G、H 三種值，F 為評價分數，G 為啟始點到目前節點的距離，H 為預測目前節點到結束點的距離，並且有兩運用兩個 List，一個是 Open List 放尚未探詢過的點，另一個是 Close List，放以探詢過的點，一開始，會先將起始點放入 Open List 中，然後開啟周圍四個點，計算這四個點的參數後，將這四個點的父點設定為自己後，將這四個點放入 Open List 中後將起始點從 Open List 移除，推入 Close List 中，之後反覆地在 Open List 找出評價分數最低的，設為現在探索點，開啟周圍四個點，如果已在 Close List 中，則不再開啟，之後把開啟的點推入 Open List 並計算其參數與設定父點為現在探索點後，再將目前的探索點從 Open List 移除後再放入 Close List 中，直到目前探索點的座標與目標相同，依序從目前探索點開始從父值往回追蹤並記錄，此就為最短路徑。若探索到最後 OPENLIST 是空的，表示找不到路徑。

四、結語

1.問題與解決方法

*地圖系統的設計

在剛開始修這門時，我們所遇到的最大問題就是地圖系統，當時對於這個問題是莫大的頭痛，後來有去問其他同學與老師後發現，原來老師有提供參考的文件，於是我們看過問件後，首先就解決了我們一開始人會動，背景不會動的情形，但是後來發現，事實上人物在移動時應該是背景的地圖上未移動到底時，人物是維持在螢幕中間的，而我們原本的移動模式是採像素，所之後改為以座標格的方式移動，解決的方式經過多次的推敲，找到判別方式後就解決了。

後來就因為我們有用到地圖，需要讀檔，而剛開始讀檔經常發生問題，後來就上網查讀檔的資料後，再使用 DEVC++測試過後，就放入主程式後就可順利執行了。

*怪物自動路徑偵測 AI

接下來是 AI 的問題，因為我們的 AI 必須追小偷，所以必須撰寫自動找尋路徑方法，原先我有以以前郭老師教的遞迴撰寫，但是因為一直有 Bug 所以就上網找到 A*演算法，將其方式學起來後，套用到我們的程式當中，但也發現了一個很大的問題，當兩點距離過遠時其找路徑要花費非常多的時間，所以後來我們定點移動時，都取約直線距離 20 左右來移動。而我們在這個部分卡了非常的久，花了不少時間再完成 AI 的項目。

當完成 AI 時，原本我們的追蹤方式是採在範圍內，就會去追，但是後來發現，此舉動會拖垮效能，因理當有障礙物時，AI 是不應該要追的，所以就開始撰寫視線的涵式，在這期間也花了不少的時間，就一直反覆推，直到推出一個較好的方法後，套用上去，就解決了這個問題。

*紅屏螢幕閃爍

在程式開發末期，我們發現被怪物追蹤時閃爍紅屏會有持續紅螢幕無法消失的問題，接著我們改寫時間控制有物件控制，之後就沒有發現有此錯誤，但是最近在運行時發現，還是有機

率還是會出現此錯誤，因為發生的次數非常的少，我們依然無法找不到此 Bug，也因是最後一個禮拜，也無法再問人，所以這個是我們剩餘的無法解決的問題。

2.時間表(Time Log)

週期	內容細項		
02/23 ~ 03/02 本周總時數 6 小時 0 分	第一次上課-學習遊戲框架	張竣翔：3 小時 0 分	
		李旻憲：3 小時 0 分	
03/02 ~ 03/09 本周總時數 9 小時 10 分	論該如何分工、遊戲模式架構為何	張竣翔：3 小時 0 分	
		李旻憲：3 小時 0 分	
	練習遊戲框架	張竣翔：2 小時 0 分	
		李旻憲：1 小時 30 分	
03/09 ~ 03/16 本周總時數 7 小時 30 分	與助教討論	張竣翔：0 小時 15 分	
		李旻憲：0 小時 15 分	
	與同組同學討論	張竣翔：0 小時 30 分	
		李旻憲：0 小時 30 分	
	完成人物移動模型	張竣翔：2 小時 0 分	
	撰寫地圖初始架構	張竣翔：4 小時 0 分	
	完成地圖設計	李旻憲：0 小時 30 分	
03/16 ~ 03/23 本周總時數 17 小時 45 分	繪製遊戲初始頁面底圖及選單	李旻憲：1 小時 15 分	
	完成讀檔讀取地圖結構並人物在地圖中可正確移動	張竣翔：12 小時 0 分	
	確認討論項目及了解同伴到目前所撰寫的程式碼內容	李旻憲：1 小時 30 分	
	修正移動錯誤及地圖大小/位置變更之修正與測試	李旻憲：3 小時 0 分	
03/23 ~ 03/30	討論遊戲寶物加入方式,計分規則,過	李旻憲：0 小時 55 分	

本周總時數 7 小時 45 分	關方式/發現人物移動變慢問題	張竣翔：0 小時 55 分	
	加入第一關遊戲背景及音樂,測試程式要如何貼圖檔/修改地圖大小	李旻憲：1 小時 55 分	
	討論地圖演算法架構及其改進方式	李旻憲：2 小時 0 分	
		張竣翔：2 小時 0 分	
03/30 ~ 04/06 本周總時數 8 小時 25 分	繪製遊戲開始/破關/過關頁面	李旻憲：1 小時 50 分	
	加入第一關遊戲背景及音樂,測試程式要如何貼圖檔/修改地圖大小	李旻憲：1 小時 55 分	
	討論地圖演算法架構及其改進方式	李旻憲：2 小時 20 分	
		張竣翔：2 小時 20 分	
04/06 ~ 04/13 本周總時數 27 小時 30 分	看組員撰寫程式碼/加入怪物可移動	李旻憲：2 小時 0 分	
	撰寫守衛自動移動	張竣翔：10 小時 0 分	
	測試程式(自動移動)	李旻憲：2 小時 30 分	
	測試程式(守衛及盜賊)	李旻憲：0 小時 45 分	
	與同組討論思考如何實現 A*演算法在此遊戲上	李旻憲：3 小時 30 分	
		張竣翔：5 小時 20 分	
	思考(設計演算法或類別)	李旻憲：3 小時 15 分	
04/13 ~ 04/20 本周總時數 11 小時 45 分	與助教討論(NPC 架構討論)	李旻憲：0 小時 30 分	
	與助教討論(NPC 架構討論)	張竣翔：0 小時 30 分	
	修改 NPC 會穿牆問題	張竣翔：6 小時 0 分	
	NPC 追蹤角色演算法改進	張竣翔：1 小時 15 分	
	看組員撰寫程式碼/加入鑰匙	李旻憲：1 小時 15 分	
	遊戲跳關/計數器/路徑改進	李旻憲：2 小時 15 分	
04/20 ~ 04/27 本周總時數	主角隨機產生/寶物顯示於地圖	李旻憲：5 小時 0 分	
	持續改良守衛移動演算法	張竣翔：8 小時 45 分	

13 小時 45 分			
04/27 ~ 05/04 本周總時數 14 小時 40 分	撰寫怪物 AI 視線	張竣翔：8 小時 30 分	
	撰寫怪物返回路徑	張竣翔：1 小時 30 分	
	增加一名守衛在地圖上	張竣翔：4 小時 0 分	
	測試視線程式/撰寫檔案 I/O	李旻憲：0 小時 40 分	
05/04 ~ 05/11 本周總時數 11 小時 0 分	改寫怪物路徑 Link List	張竣翔：3 小時 0 分	
	撰寫遊戲過關頁面及成績/秒數	李旻憲：4 小時 0 分	
	思考(設計演算法或類別)	張竣翔：4 小時 0 分	
05/11 ~ 05/18 本周總時數 16 小時 0 分	將第一關卡所有功能設計完畢	張竣翔：15 小時 0 分	
	測試程式(試玩遊戲)	李旻憲：0 小時 30 分	
	測試程式(試玩遊戲)	張竣翔：0 小時 30 分	
05/18 ~ 05/25 本周總時數 3 小時 30 分	與同組同學討論	張竣翔：1 小時 0 分	
		李旻憲：1 小時 0 分	
	思考(設計演算法或類別)	李旻憲：1 小時 30 分	
05/25 ~ 06/01 本周總時數 8 小時 30 分	增加各種顏色的門及鑰匙	張竣翔：5 小時 0 分	
	尋找鑰匙/寶物圖檔,地圖/選單/過關/ 取得寶物及鑰匙/失敗/開門音效	李旻憲：3 小時 30 分	
06/01 ~ 06/08 本周總時數 8 小時 30 分	修改 DOOR 讓他有兩種型態	張竣翔：2 小時 0 分	
	撰寫程式碼/撰寫報告	李旻憲：6 小時 30 分	
06/08 ~ 06/15 本周總時數 6 小時 30 分	修正遊戲數個剩餘小問題	李旻憲：2 小時 30 分	
	修正所有遊戲圖檔(稍微美化)	李旻憲：2 小時 0 分	
	遊戲除錯	張竣翔：2 小時 0 分	
06/15 ~ 06/22 本周總時數	加入第二關/修改遊戲符合規定	李旻憲：3 小時 0 分	
	遊戲報告撰寫(架構/演算法)	張竣翔：2 小時 0 分	

5 小時 50 分	遊戲除錯	李旻憲：0 小時 50 分	
-----------	------	---------------	--

3.貢獻比例

張竣翔	60%	李旻憲	40%
-----	-----	-----	-----

4.檢核表

	項目	完成否	無法完成的原因
1.	自訂遊戲 ICON	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
2.	全螢幕啟動	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
3.	修改 Help->About	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
4.	初始畫面說明案件及滑鼠用法與密技	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	
5.	上傳 Setup 檔	<input checked="" type="checkbox"/> 已完成 <input type="checkbox"/> 未完成	

5.收穫

98590316 張竣翔

在這次的過程中，因為會感覺到壓力，我覺得比較有收穫的就是時間規劃與抗壓性，在其他課業都還蠻繁重的同時，要顧及每項作業都得完成，並且每周都要有進度交代，與在企業沒什麼兩樣，也讓我們體會到在業界的感覺，這次也因為是使用老師的框架，也是學習到該怎麼看懂別人的程式，如果一開始我們沒有看懂老師的程式，那麼我們這學期的遊戲我看連一點皮毛都寫不出來，並且這次的遊戲並不是一個人所能完成的，所以也時常的要與另外一位夥伴溝通，互相追問彼此的進度，也在這次的專題中，了解到互相合作的重要性，雖然在過程中摩擦非常的多，但是我們還是走了過來，並且一同完成了這個專題，這也是非常難得可貴的，並且在這次專題製作中，也學習到該怎麼解決問題，該怎麼使用週邊的資源，總而言之，這個專題製作讓我收益良多。

98590342 李旻憲

在這次的課程當中，說沒有壓力真的是騙人的，因為我們要從一個完全沒用過的框架來接手開發任務，可能我們寫了一學期都還沒完全搞懂怎麼最有效的運用這個遊戲框架，但是我卻學到了自己在開發程式的過程中要如何與同伴做溝通，如何將自己所製作的程式碼迅速有效地給同伴讀懂且能利用，還有在設計遇到困難或者不會使用時可以透過微軟的 MSDN 來找答案或找到文章來解決困難，當然上課時更可以請助教或老師幫忙，有問題就一定要問，事情才能獲得解決。而這次的實習課也讓我感覺到其實我自己的能力還是明顯不足，需要多多練習程式還有基礎題，這樣才能更進步以及對於開發這部分更有想法。

6.心得

98590316 張竣翔

在我上大學前，我是完全沒有程式基礎的人，到了大學後，才接觸到程式，在大一、大二上學期，都是在打基礎，到了下學期接觸到這門課後，才真正了解到，原來撰寫程式是那麼的不容易，在以前我們所交的作業，真的就只是一個小小的一份作業，那時候成績還不錯，感覺程式好像沒有那麼難。

俗話說的好，萬事起頭難，在一開始的時候，我和我的夥伴對於要製作什麼遊戲完全沒有想法，那時題目是一個換一個，一直不知道該做什麼，當決定好題目後，因為也沒有經驗，也不知道該怎麼做，在前四個禮拜，寫了又整個改掉，寫了又整個刪掉，差不多過了4個禮拜左右，才真正比較熟悉老師的架構，而我也把整的遊戲的雛型給弄了出來，當然之後所遭遇的事情都沒有那麼簡單，但是到了現在結束了，雖然我們所寫的遊戲有BUG會讓他當機，而我們也嘗試過要解決他，但因他出現的機率真的很不一定，上次我連玩了好幾次都沒有遇到問題，當遇到了也不知道該如何解決，因為次數太少無法知道其正確原因為何，這個也讓我們感到很不安，很怕因此而被當掉，但是看到自己所做得遊戲，能夠跑起來，真的很有成就感，也在這之中學到了很多，在這次也有體會到當科技人熬夜趕工的感覺，也讓我知道我對程式到底熟不熟？也從這之中去思考我是否真的適合這種做工作嗎？等等的問題，或許實務專題真的很累人，但是從做中學才會讓人有收穫，再累看到自己的成果，就覺得很欣慰了，如果成績還是不如人意，而我們也已經很努力去做了，我相信未來還會有更多的進步空間的。

98590342 李旻憲

以前，我從來不懂會寫程式後為什麼可以做出像企業樣很大型的程式，我只了解他們透過很多人的分工去完成，實際到底怎麼跑出來的，遊戲公司又如何做出來的根本完全沒有概念，即使我們在一、二年級的課程中已經學習到了C還有C++但這不過就只是讓我們能夠學到基礎的概念還有遵守基本的開發原則，在程式設計上更有效率，我覺得在每一次的程式設計課程上，總是會有許多新的概念衝擊我，也讓我對於程式設計師有新的想法與認知，而在這堂課程上我們要學著如何分工，還有要從老師已經開發好的框架來接手，這不就是跟進到一間公司後

立即要面對的問題嗎？而且還得共同完成一個專案，這說大不大說小不小，但沒有妥善的溝通我想我們根本不必修這堂課了，而老師在上課也說過，這堂課就是驗證我們的功力，以前底子打得好，現在開發遊戲起來當然輕鬆很多，底子不好，搞不好要生出一個物件就很困難，更別提要遵守什麼低凝聚性高內聚力的 OOP 原則了。終於到了學期末了，也是讓我們展現自己的作品時候，我們的遊戲稱不上是最棒的，但我想這是我們用心做出來的，每一個環節都有我們每天盯著電腦的痕跡，在這一次的實習課程中，我有了不一樣的體驗，真的非常的棒！而且讓我印象非常深刻，我覺得做中學是最有效的實務解決辦法，就跟你去外面打工一樣，做一次，你就會了，學到了更多。

7. 對於本課程的建議

在這次的課程中，有發現的我們所使用的課程網頁有問題，我是第 20 組的成員，但是在檢視報表中所顯示的是第 4 組，完全是無法看到我們這組的報表，這個是需要修正的，希望課程大綱能在寒假前就公布完畢後，讓學弟們能夠很早的就能著手開始寫，這樣完成的東西會更多。

附錄

MyGame.h

```

/*
 * mygame.h: 本檔案儲遊戲本身的 class 的 interface
 * Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>
 *
 * This file is part of game, a free game development framework for windows.
 *
 * game is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * game is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 *
 * 2004-03-02 V4.0
 * 1. Add CGameStateInit, CGameStateRun, and CGameStateOver to
 * demonstrate the use of states.
 * 2005-09-13
 * Rewrite the codes for CBall and CEraser.
 * 2005-09-20 V4.2Beta1.
 * 2005-09-29 V4.2Beta2.
 * 2006-02-08 V4.2
 * 1. Rename OnInitialUpdate() -> OnInit().
 * 2. Replace AUDIO_CANYON as AUDIO_NTUT.
 * 3. Add help bitmap to CGameStateRun.
 * 2006-09-09 V4.3
 * 1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they
 * are event driven.
 * 2008-02-15 V4.4
 * 1. Add namespace game_framework.
 * 2. Replace the demonstration of animation as a new bouncing ball.
 * 3. Use ShowInitProgress(percent) to display loading progress.
 */
// Constants
// 定義各種音效的編號
enum AUDIO_ID {
    AUDIO_STBGS, // 0 - 遊戲開始畫面背景音樂
    AUDIO_SELECT, // 1 - 選單音效
    AUDIO_SCUESS, // 2 - 成功過關音效
    AUDIO_FAILED, // 3 - 失敗(GAME OVER)頁面音樂
    AUDIO_PICK, // 4 - 撿起寶物聲音
    AUDIO_DOOR, // 5 - 開門聲音
    AUDIO_MAP1, // 6 - 地圖 1 音樂
    AUDIO_MAP2, // 7 - 地圖 2 音樂
    AUDIO_MAP3, // 8 - 地圖 3 音樂
    AUDIO_MAP4, // 9 - 地圖 4 音樂
    AUDIO_MAP5, // 10 - 地圖 5 音樂
    AUDIO_KEY // 11 - 撿到 KEY 音樂
};
namespace game_framework {
//地圖設計
class CMap{
public:

```

```

CMap(const char* fn);
void Initialize(const char* fn);           //初始化
void LoadMap();                          //載入背景圖片
void OnShow();
void MoveMap(char direction);             //移動地圖
int Iempty(int x,int y);                  //回傳座標(X,Y)狀態
int DetialDoor(int x,int y);              //回傳座標(X,Y)門的型態
int computsx(int x);                      //回傳 X 與螢幕左上角 SX 的距離
int computsy(int y);                      //回傳 Y 與螢幕左上角 SY 的距離
int Getcoutobstacle();                   //回傳總牆壁數
int Getcounttreasure();                  //回傳總寶物數
int Getcountdoors();                     //回傳總門數
int Getcountkeys();                      //回傳總鑰匙數
char KeyColor(int x,int y);              //回傳鑰匙顏色
void Setsxsy(int x,int y);               //設定螢幕左上角座標
void toempty(int x,int y);               //設定指定座標為空白
void setdoor(int x,int y,char type);      //設定指定座標為門
private:
    CMovingBitmap background;             //背景圖
    char map[160][120];                  //地圖陣列
    int sx,sy;                           //座標
    int countobstacle;                   //總牆壁數
    int counttreasures;                  //總寶物數
    int countdoors;                      //總門數
    int countkeys;                       //總鑰匙數
    void readmapfromfile(const char* fn); //讀檔函式
};
///////////////////////////////////////////////////////////////////
//障礙物物件
///////////////////////////////////////////////////////////////////
class Obstacle{
public:
    Obstacle();
    void OnShow(CMap* gamemap);
    void Load();                         //載入圖片
    void setxy(int setx, int sety);       //設定牆壁座標
private:
    int x,y;                             //座標
    CMovingBitmap bmp;                   //圖片
};
///////////////////////////////////////////////////////////////////
//移動點物件
///////////////////////////////////////////////////////////////////
class MovingNode{
public:
    MovingNode(int sx, int sy);           //建構元
    int Getx();                           //取得座標 x
    int Gety();                           //取得座標 y
    int Getg();                           //取得與起始點的距離
    int Getf();                           //取得評價分數
    MovingNode* GetParent();              //取得父點
    // ↓ 計算參數
    void caculateparameter(const int sx, const int sy,const int tx,const int ty);
    void setparent(MovingNode* settarget); //設定父點
private:
    int x,y;                             //座標
    int f,g,h;                           //f 評價分數 g 與起始點的距離 h 與目標點的距離
    MovingNode* parent;                  //父點
};
///////////////////////////////////////////////////////////////////
//時間物件
///////////////////////////////////////////////////////////////////
//將 CInteger 轉為時分秒制,用於關卡時間計算
class CintTime{
public:
    CintTime();
    void Initialize();                    //初始化
    void SetTime(int);                   //設定時間,為秒
    int GetTime();                       //取得目前時間,為秒
};

```



```

    void LoadBitmap(bool CTFirst);           //讀取圖片
    void OnShow();                           //顯示圖形
private:
    void calTime();                          //將秒數轉換為時分秒制
    int min,sec;                             //分、秒
    CInteger Cint;                           //數字轉圖片顯示
};
//Door
class Door{
public:
    Door();
    void Initialize(CMap* gamemap);          //初始化
    void LoadDoor();                        //載入圖片
    void setClose(bool closeflag,CMap* gamemap); //設定開關狀態
    bool HitObject(int sx,int sy);          //是否有撞到人
    void setDoor(int sx,int sy,char type,char color); //設定門
    bool GetisClose();                      //回傳開關狀態
    void OnShow(CMap* gamemap);            //顯示門
    char Getdoortype();                     //取得門的類型
    char Getdoorcolor();                   //取得門的顏色
private:
    bool isClose;                          //開關狀態
    bool isVertical;                        //True 直向 False 橫向
    int x,y;                               //座標
    static const int doorsize = 8;         //門的大小
    CMovingBitmap doorpicture;             //門的圖檔
    char doortype;                          //門的型態
    char doorcolor;                        //門的顏色
};
//Key
class Key{
public:
    Key();
    void Initialize();                      //初始化
    void Onshow(CMap* gamemap);            //顯示鑰匙
    void setKey(int sx,int sy,char color); //設定鑰匙
    void LoadKey();                        //載入圖片
    char getColor();                        //取得鑰匙顏色
    bool hitThief(int sx,int sy);           //是否撞到小偷
    bool getisOnMap();                      //回傳是否在地圖上
    void setOnMap(bool flag);               //設定是否在地圖上
private:
    int x,y;                               //座標
    static const int width =2;              //鑰匙的寬
    static const int high =2;              //鑰匙的高
    bool isOnMap;                           //是否在地圖上
    char keycolor;                           //鑰匙顏色
    CMovingBitmap keypicture;               //鑰匙圖片
};
//盜賊物件
class Thief{
public:
    Thief();
    void Initialize();                      // 設定小偷為初始值
    void LoadBitmap();                     // 載入圖形
    void OnMove(CMap* gamemap,Door* door,Key* key); // 移動小偷
    void OnShow(CMap* gamemap);            // 將小偷圖形貼到畫面
    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 sx,int sy);             // 設定小偷貼圖左上角座標
    void reststep();                        // 重設腳步(step 歸零)
};

```

```

    int Getx();                // 回傳座標 X
    int Gety();                // 回傳座標 Y
protected:
    CMovingBitmap rogue[4][4]; // 小偷的圖片陣列
    CMovingBitmap *play;       // 目前顯示的圖片
    int step;                   // 行走狀態
    int x,y;                    // 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
    void contralstep(int direction); // 控制腳步(圖片)
    // ↓判斷是否可以朝指定方向移動
    bool checkempty(char direction,CMap* Gamemap,Key *key,Door *door);
};
////////////////////////////////////
//怪獸物件
////////////////////////////////////
class Monster{
public:
    Monster();
    void Initialize();          // 設定怪物為初始值
    void LoadBitmap();          // 載入圖形
    void OnMove(CMap* gamemap,Thief* thief,Door* door,bool cheatMode); // 移動怪物
    void OnShow(CMap* gamemap); // 將怪物圖形貼到畫面
    void reststep();            // 重設腳步(mstep)
    int Getx();                  // 回傳 X
    int Gety();                  // 回傳 Y
    void SetXY(int x, int y);    // 設定座標
    void Setcommenpath( vector<MovingNode*> &scriptpath); //設定一般路線
    bool GetisTrace();           // 取得追中狀態
protected:
    CMovingBitmap guide[4][4];  // 怪物的動畫
    CMovingBitmap *monster_1;    // 正在顯示的圖片
    int mstep;                   // 行走狀態
    int gx,gy;                   // 警衛所在座標
    static const int sightrange = 10; // 緊備範圍
    unsigned int nowstep;        // 目前的步數(用於移動路徑)
    char nowdirection;           // 目前方向
    vector<MovingNode*> path;     // 移動路徑
    vector<MovingNode*> returnpath; // 追蹤啟動後所存的移動路徑(便於返回)
    vector<MovingNode*> commenpath; // 既定路徑(儲存移動點)
    void findpath(const int tx, const int ty,CMap* gamemap); //找尋路徑
    bool isMoving;               // 是否正在移動
    bool isTrace;                // 是否正在追蹤
    bool isBack;                 // 是否在反回原既定路徑中
    bool isComeback;             // 是否已走到既定路徑的終點
    void contralstep(int direction); // 控制 mstep(圖片)
    bool checkempty(int x, int y,char direction,CMap* Gamemap); //確認指定方向是否可以移動
    bool looksight(Thief* thief,CMap* gamemap,bool cheatmode); //判定小偷是否在視線範圍內
    int mcounter;                // 計數器(用以計數用來減緩移動速度)
    unsigned int commenwalkstep; // 紀錄目前移動到哪個既定點的位置
    bool opendoor;               // 是否有碰撞到門
    Door *thedoor;               // 紀錄剛才所碰撞的門
    bool wait;                   // 是否進入延遲
    int waitcounter;             // 延遲計數器
};
////////////////////////////////////
//寶物
////////////////////////////////////
class Treasure{
public:
    Treasure();
    void Initialize();           // 初始化
    void LoadBitmap();           // 載入圖片
    void OnShow(CMap* gamemap);  // 顯示圖片
    void SetXY(int sx,int sy);    // 設定座標
    void SetSize(int h, int w);  // 設定圖片大小
    void Setlife(bool flag);      // 設定是否還存在

```

```

    bool HitThief(Thief* thief);           //回傳是否撞擊到小偷
    bool Isalive();                       //回傳是否還存在
    int Getheight();                      //回傳寶物高度
    int Getwidth();                      //回傳寶物寬度
    int Getx();                          //取得 X
    int Gety();                          //取得 Y
private:
    int x,y;                             //座標
    int height,width;                   //長、寬
    CMovingBitmap treasure_bitmap;      //圖形
    bool islife;                        //是否存在
};
//紅螢幕物件
class redscreen{
public:
    redscreen();
    void Initialize();                  //初始化
    void LoadBitmap();                 //載入圖片
    void OnShow();                     //顯示圖片
    void SetisFlash(bool flasflag);    //設定是否進入閃爍
    bool GetisFlash();                 //取得閃爍狀態
private:
    int counter;                       //計數器(透過計數器控制閃爍)
    bool isFlash;                      //是否為閃爍狀態
    CMovingBitmap redscreen_bitmap;    //紅色圖
};
//說明頁面
class help{
public:
    help();
    void Initialize();                 //初始化
    void LoadBitmap();                //讀取圖片
    void OnShow();                    //顯示圖片
    void SetOk(bool flag);             //設定是否看過說明
    bool GetState();                  //取得狀態
private:
    bool flag;                        //是否看過說明狀態
    CMovingBitmap help_bitmap;        //說明圖
};
// 這個 class 為遊戲的遊戲開頭畫面物件
// 每個 Member function 的 Implementation 都要弄懂
class CGameStateInit : public CGameState {
public:
    CGameStateInit(CGame *g);
    void OnInit();                    // 遊戲的初值及圖形設定
    void OnBeginState();              // 設定每次重玩所需的變數
    void OnKeyUp(UINT, UINT, UINT);  // 處理鍵盤 Up 的動作
protected:
    void OnShow();                    // 顯示這個狀態的遊戲畫面
private:
    CMovingBitmap stbg;               //遊戲開始開始頁面背景圖
    CMovingBitmap About;              //關於製作底圖
    CMovingBitmap ExMouse;           //箭頭圖案
    bool mu,About_bool;              //是否在 MENU 與 是否在 ABOUT 頁面
};
// 這個 class 為遊戲的選擇關卡頁面
class CGameStateSelect : public CGameState {
public:
    CGameStateSelect(CGame *g);
    void OnInit();                    // 遊戲的初值及圖形設定
    void OnBeginState();              // 設定每次重玩所需的變數

```

```

    void OnKeyUp(UINT, UINT, UINT);          // 處理鍵盤 Up 的動作
protected:
    void OnShow();                          // 顯示這個狀態的遊戲畫面
private:
    CMovingBitmap Select;                   // 遊戲開始開始頁面背景圖
    CMovingBitmap ExMouse;                  // 箭頭圖案
};
// 這個 class 為遊戲的遊戲執行物件(第一關)，主要的遊戲程式都在這裡
// 每個 Member function 的 Implementation 都要弄懂
// 這個 class 為遊戲的遊戲執行物件(第二關)，主要的遊戲程式都在這裡
// 每個 Member function 的 Implementation 都要弄懂
class CGameStateRun : public CGameState {
public:
    CGameStateRun(CGame *g);
    ~CGameStateRun();
    void OnBeginState();                    // 設定每次重玩所需的變數
    void OnInit();                          // 遊戲的初值及圖形設定
    void OnKeyDown(UINT, UINT, UINT);       // 處理鍵盤 Up 的動作
    void OnKeyUp(UINT, UINT, UINT);         // 處理鍵盤 Down 的動作
protected:
    void OnMove();                          // 移動遊戲元素
    void OnShow();                          // 顯示這個狀態的遊戲畫面
private:
    const char *fn;                         // 讀檔位置
    CMap gamemap;                           // 地圖
    Thief thief;                            // 小偷
    Monster *monster;                       // 怪物
    Obstacle *obstacles;                   // 牆壁
    Treasure *treasure;                     // 寶物
    Door *door;                             // 門
    Key *key;                               // 鑰匙
    redscreen redscreen;                    // 閃爍紅屏
    int treasure_remind;                     // 剩餘寶物數量
    int counter;                             // 倒數之計數器
    static const int limatetime = 306;      // 此關卡遊戲時間
    static const int stage=1;               // 關卡編號
    bool cheatmode;                         // 作弊模式狀態
    bool dkeybordisdown;                    // D 鍵是否在被按下狀態
    bool wkeybordisdown;                    // W 鍵是否再被按下狀態
    int score,time;                         // 成績紀錄
    help help;                              // 說明
    CMovingBitmap cheat;                    // 作弊模式圖
    bool togetalltreasure;                  // 取得所有寶物模式旗標
    CintTime Tcc;                           // 時間計數器
};
// 這個 class 為遊戲的遊戲執行物件(第二關)，主要的遊戲程式都在這裡
// 每個 Member function 的 Implementation 都要弄懂
// 這個 class 為遊戲的遊戲執行物件(第二關)，主要的遊戲程式都在這裡
// 每個 Member function 的 Implementation 都要弄懂
class CGameStateRun2 : public CGameState {
public:
    CGameStateRun2(CGame *g);
    ~CGameStateRun2();
    void OnBeginState();                    // 設定每次重玩所需的變數
    void OnInit();                          // 遊戲的初值及圖形設定
    void OnKeyDown(UINT, UINT, UINT);
    void OnKeyUp(UINT, UINT, UINT);
protected:
    void OnMove();                          // 移動遊戲元素
    void OnShow();                          // 顯示這個狀態的遊戲畫面
private:
    const char *fn;                         // 讀檔位置
    CMap gamemap;                           // 地圖
    Thief thief;                            // 小偷
    Monster *monster;                       // 怪物
    Obstacle *obstacles;                   // 牆壁
    Treasure *treasure;                     // 寶物
    Door *door;                             // 門
    Key *key;                               // 鑰匙

```

```

redscreen redscreen;
int treasure_remind;
int counter;           // 倒數之計數器
static const int limatetime = 533; // 此關卡遊戲時間
static const int stage=2;
bool cheatmode;        // 作弊模式狀態
int score,time;        //成績紀錄
help help;
CMovingBitmap cheat;
CintTime Tcc;
bool togetalltreasure;
bool dkeybordisdown;
bool wkeybordisdown;
};
/////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的遊戲執行物件(過關畫面)
/////////////////////////////////////////////////////////////////
class CGameStatePASS : public CGameState {
public:
    CGameStatePASS(CGame *g);
    void OnInit();           // 遊戲的初值及圖形設定
protected:
    void OnBeginState();
    void CalST();           //計算時間與分數
    void OnKeyUp(UINT, UINT, UINT);
    void OnShow();         // 顯示這個狀態的遊戲畫面
    void readfromfile();   // 讀檔
private:
    int counter;           // 倒數之計數器
    //儲存過關頁面要顯示的資訊
    int min,sec;           //分秒
    int time;             //儲存轉成 int 的總秒數
    int score;            //分數
    int min1,min2;
    int sec1,sec2;
    int stage;            //關卡位置
    CMovingBitmap PASS;   //遊戲開始頁面背景圖
    //以下為數字圖片
    CMovingBitmap t1;
    CMovingBitmap t2;
    CMovingBitmap t3;
    CMovingBitmap t4;
    CMovingBitmap t5;
    CMovingBitmap t6;
    CMovingBitmap t7;
    CMovingBitmap t8;
    CMovingBitmap t9;
    CMovingBitmap t0;
    CIntegerST CiST;       //數字顯示
    bool mu;
};
/////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的破關狀態(Game WIN)
/////////////////////////////////////////////////////////////////
class CGameStateWIN : public CGameState {
public:
    CGameStateWIN(CGame *g);
    void OnBeginState();
    void OnInit();           // 設定每次重玩所需的變數
protected:
    void OnKeyUp(UINT, UINT, UINT);
    void OnMove();           // 移動遊戲元素
    void OnShow();         // 顯示這個狀態的遊戲畫面
private:
    int counter;           // 倒數之計數器
    CMovingBitmap WIN;     // 破台圖片
};

```

```

////////////////////////////////////
// 這個 class 為遊戲的結束狀態(Game Over)
// 每個 Member function 的 Implementation 都要弄懂
////////////////////////////////////
class CGameStateOver : public CGameState {
public:
    CGameStateOver(CGame *g);
    void OnBeginState();           // 設定每次重玩所需的變數
    void OnInit();
protected:
    void OnKeyUp(UINT, UINT, UINT);
    void OnMove();                 // 移動遊戲元素
    void OnShow();                 // 顯示這個狀態的遊戲畫面
private:
    int counter;                  // 倒數之計數器
    CMovingBitmap Over;           // 遊戲結束圖片
};
}

```

MyGame.cpp

```

/*
 * mygame.cpp: 本檔案儲遊戲本身的 class 的 implementation
 * Copyright (C) 2002-2008 Woei-Kae Chen <wkc@csie.ntut.edu.tw>
 *
 * This file is part of game, a free game development framework for windows
 *
 * game is free software; you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation; either version 2 of the License, or
 * (at your option) any later version.
 *
 * game is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program; if not, write to the Free Software
 * Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
 *
 * History:
 * 2002-03-04 V3.1
 *     Add codes to demonstrate the use of CMovingBitmap::ShowBitmap(CMovingBitmap &).
 * 2004-03-02 V4.0
 *     1. Add CGameStateInit, CGameStateRun, and CGameStateOver to
 *        demonstrate the use of states.
 *     2. Demo the use of CInteger in CGameStateRun.
 * 2005-09-13
 *     Rewrite the codes for CBall and CEraser.
 * 2005-09-20 V4.2Beta1.
 * 2005-09-29 V4.2Beta2.
 *     1. Add codes to display IDC_GAMECURSOR in GameStateRun.
 * 2006-02-08 V4.2
 *     1. Revise sample screens to display in English only.
 *     2. Add code in CGameStateInit to demo the use of PostQuitMessage().
 *     3. Rename OnInitialUpdate() -> OnInit().
 *     4. Fix the bug that OnBeginState() of GameStateInit is not called.
 *     5. Replace AUDIO_CANYON as AUDIO_NTUT.
 *     6. Add help bitmap to CGameStateRun.
 * 2006-09-09 V4.3
 *     1. Rename Move() and Show() as OnMove and OnShow() to emphasize that they are
 *        event driven.
 * 2006-12-30
 *     1. Bug fix: fix a memory leak problem by replacing PostQuitMessage(0) as
 *        PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE,0,0).
 * 2008-02-15 V4.4
 *     1. Add namespace game_framework.
 *     2. Replace the demonstration of animation as a new bouncing ball.
 *     3. Use ShowInitProgress(percent) to display loading progress.
 */

```

```

* 2010-03-23 V4.6
* 1. Demo MP3 support: use lake.mp3 to replace lake.wav.
*/

#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "mygame.h"
#include <time.h>
#include <fstream>
#include <math.h>
#include <vector>
#include <list>
#include <string>
namespace game_framework {
//////////
//紅螢幕物件
//////////
redscreen::redscreen(){}
}
void redscreen::Initialize(){} //初始化
    isFlash = false;
    counter =0;
};
void redscreen::LoadBitmap(){} //載入圖片
    redscreen_bitmap.LoadBitmap("Bitmaps/red.bmp");
};
void redscreen::OnShow(){} //顯示圖片
    if(isFlash){ //若是閃爍旗標為 TRUE
        if(counter == 0){ //則每執行 25 次才顯示 1 次紅屏
            redscreen_bitmap.ShowBitmap();
            counter++;
        }
        else if(counter==25){
            counter=0;
        }
        else{
            counter++;
        }
    }
};
void redscreen::SetisFlash(bool flag){ //設定閃爍旗標
    isFlash = flag;
    if(!flag){
        counter = 0; //若更改設定為 FLASE 則讓 COUNTER 歸 0
    }
}
//////////
//說明頁面
//////////
help::help(){}
}
void help::Initialize(){}
    flag = false;
};
void help::LoadBitmap(){}
    help_bitmap.LoadBitmap("Bitmaps/help.bmp",RGB(255,255,255));
};
void help::OnShow(){}
    if(!flag){
        help_bitmap.SetTopLeft(0,0);
        help_bitmap.ShowBitmap();
    }else{
    }
};
void help::SetOk(bool flag1){

```

```

    flag = flag1;
}

bool help::GetState(){
    return flag;
}

////////////////////////////////////
//寶物 CLASS
////////////////////////////////////
Treasure::Treasure(){
}
bool Treasure::Isalive(){
    return islife;
}
void Treasure::Initialize(){
    height=2;
    width=2;
    islife=true;
}
void Treasure::Setlife(bool flag){
    islife = flag;
}

void Treasure::OnShow(CMap* gamemap){
    if(islife){
        treasure_bitmap.SetTopLeft(gamemap->computsx(x)*20,gamemap->computsy(y)*20); //依照其與螢幕的相對位置貼圖
        treasure_bitmap.ShowBitmap();
    }
}

bool Treasure::HitThief(Thief* thief){
    int tx1 = thief->Getx();          //人物左邊界
    int tx2 = thief->Getx()+3;        //人物右邊界
    int ty1 = thief->Gety();          //人物上邊界
    int ty2 = thief->Gety()+4;        //人物下邊界
    //判定是否碰撞
    if(((x>=tx1&& x<=tx2)|| (x+1>=tx1&& x+1<=tx2))&& ((y>=ty1&& y<=ty2)|| (y+1>=ty1&& y+1<=ty2))){
        return true;
    }
    else{
        return false;
    }
}

void Treasure::SetXY(int sx,int sy){
    x = sx;
    y = sy;
}

void Treasure::SetSize(int h,int w){
    height = h;
    width = w;
}

void Treasure::LoadBitmap(){
    treasure_bitmap.LoadBitmap("Bitmaps/treasure.bmp",RGB(0,0,0));
}

int Treasure::Getx(){
    return x;
}

int Treasure::Gety(){
    return y;
}

int Treasure::Getheight(){
    return height;
}

int Treasure::Getwidth(){
    return width;
}

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

```



```

// 障礙物 CLASS
///////////////////////////////////////////////////
Obstacle::Obstacle():x(0),y(0){
}
//載入當障礙物的藍色圖片
void Obstacle::Load(){
    bmp.LoadBitmap(IDB_BLUE);
}
void Obstacle::OnShow(CMap* gamemap){
    if(gamemap->computsx(x)>=0&&gamemap->computsy(y)>=0){
        //設定障礙物顯示位置
        bmp.SetTopLeft(gamemap->computsx(x)*20,gamemap->computsy(y)*20);
        //顯示障礙物
        bmp.ShowBitmap();
    }
}
void Obstacle::setxy(int setx,int sety){
    x = setx;
    y = sety;
}
///////////////////////////////////////////////////
// 移動點 CLASS
///////////////////////////////////////////////////
MovingNode::MovingNode(int sx, int sy):x(sx),y(sy){
    parent = NULL;
}
int MovingNode::GetX(){
    return x;
}
int MovingNode::GetY(){
    return y;
}
int MovingNode::Getg(){
    return g;
}
int MovingNode::Getf(){
    return f;
}
MovingNode* MovingNode::GetParent(){
    return parent;
}
void MovingNode::setparent(MovingNode* settarget){
    parent = settarget;
}
void MovingNode::caculateparameter(const int sx, const int sy,const int tx,const int ty){
    h = abs(tx-x) + abs(ty-y);          //計算與終點的位置距離
    if(parent==NULL)                    //若無父點表示是第一個點 故與起點的距離為 0
        g=0;
    else
        g = abs(x-sx)+abs(y-sy);        //計算與起點的距離
    f = g+h;                             //計算評價分數
}
///////////////////////////////////////////////////
// 地圖 CLASS
///////////////////////////////////////////////////
CMap::CMap(const char* fn){
    Initialize(fn);
}
int CMap::DetialDoor(int x,int y){      //依照其門的型別 回傳對應數字
    if(map[x][y]=='h')
        return 1;
    else if(map[x][y]=='v')
        return 2;
    else if(map[x][y]=='Y')
        return 3;
    else if(map[x][y]=='Z')
        return 4;
    else if(map[x][y]=='R')
        return 5;
}

```

```

    else if(map[x][y]=='S')
        return 6;
    else if(map[x][y]=='G')
        return 7;
    else if(map[x][y]=='H')
        return 8;
    else if(map[x][y]=='P')
        return 9;
    else if(map[x][y]=='Q')
        return 10;
    else
        return -1;
}
char CMap::KeyColor(int x, int y){    //回傳鑰匙顏色(撰寫在地圖上)
    return map[x][y];
}
void CMap::Initialize(const char* fn){
    sx = 0;
    sy = 0;
    countkeys=0;
    countobstacle=0;
    counttreasures=0;
    countdoors= 0;
    readmapfromfile(fn);            //讀取地圖檔案
    for(int i=0;i<160;i++){
        for(int j=0 ; j<120;j++){    //計算各物件數
            if(Isempy(i,j)==0)
                countobstacle++;
            else if(Isempy(i,j)==2){
                counttreasures++;
            }
            else if(Isempy(i,j)==4)
                countdoors++;
            else if(Isempy(i,j)==5){
                countkeys++;
            }
        }
    }
}
//載入遊戲所需地圖檔案
void CMap::LoadMap(){
    background.LoadBitmap("Bitmaps/floor.bmp");
}
//由讀進來的地圖檔判斷障礙物位置及可走路徑
int CMap::Isempy(int x, int y){
    if(map[x][y]==' ')                //可走路徑
        return 1;
    else if(map[x][y]=='o')            //障礙物
        return 0;
    else if(map[x][y]=='t')            //寶物
        return 2;
    else if(map[x][y]=='e')            //出口
        return 999;
    else if(map[x][y]=='h' || map[x][y]=='v') //4 的皆為門
        return 4;
    else if(map[x][y]=='Y' || map[x][y]=='Z')
        return 4;
    else if(map[x][y]=='R' || map[x][y]=='S')
        return 4;
    else if(map[x][y]=='G' || map[x][y]=='H')
        return 4;
    else if(map[x][y]=='P' || map[x][y]=='Q')
        return 4;
    else if(map[x][y]=='y' || map[x][y]=='r' || map[x][y]=='g' || map[x][y]=='p') //鑰匙
        return 5;
    else
        return -1;
}
void CMap::OnShow(){

```

```

    int x = 0 - sx*20;          //因 SX 與 SY 是螢幕左上方的位置 故在貼背景圖時 要依照其相對應位置貼圖
    int y = 0 - sy*20;
    background.SetTopLeft(x,y);
    background.ShowBitmap();
}
void CMap::MoveMap(char direction){ //移動背景圖
    if(direction == 'r' && sx<128)
        sx++;
    if(direction == 'l' && sx>0)
        sx--;
    if(direction == 'u' && sy>0)
        sy--;
    if(direction == 'd' && sy<96)
        sy++;
}
void CMap::toempty(int x , int y){
    map[x][y] = ' ';
}
void CMap::setdoor(int x, int y,char type){
    map[x][y]=type;
}
//取得人物目前位置 x 與地圖的左上角格子相差多少
int CMap::computsx(int x){
    return (x-sx);
}
//取得人物目前位置 y 與地圖的左上角格子相差多少
int CMap::computsy(int y){
    return (y-sy);
}
//回傳障礙物個數總數
int CMap::Getcoutobstacle(){
    return countobstacle;
}
int CMap::Getcounttreasure(){
    return counttreasures;
}
int CMap::Getcountdoors(){
    return countdoors;
}
int CMap::Getcountkeys(){
    return countkeys;
}
void CMap::readmapfromfile(const char* fn){          //地圖檔讀取地方
    ifstream inf;
    char temp[120][161];
    inf.open(fn);
    GAME_ASSERT(inf,"地圖檔讀取失敗.");
    for(int i=0;i<120;i++)
        inf.read(temp[i],161);                      //161 的位置是分段符號
    for(int i=0;i<120;i++){                          //寫入陣列中
        for(int j=0;j<160;j++){
            map[j][i]=temp[i][j];
        }
    }
    inf.close();
}
void CMap::Setsxsy(int x,int y){
    if(x<0)
        x=0;
    if(x>159)
        x=159;
    if(y<0)
        y=0;
    if(y>119)
        x=119;
    sx=x;
    sy=y;
}
////////////////////////////////////

```

```

//Key
////////////////////////////////////
Key::Key(){
}
void Key::Initialize(){
    isOnMap = true;
}
char Key::getColor(){
    return keycolor;
}
void Key::LoadKey(){
    if(keycolor == 'y'){
        keypicture.LoadBitmap("Bitmaps/yellow-key.bmp",RGB(255,255,255));
    }
    else if(keycolor == 'r'){
        keypicture.LoadBitmap("Bitmaps/red-key.bmp",RGB(255,255,255));
    }
    else if(keycolor == 'g'){
        keypicture.LoadBitmap("Bitmaps/green-key.bmp",RGB(255,255,255));
    }
    else if(keycolor=='p'){
        keypicture.LoadBitmap("Bitmaps/purple-key.bmp",RGB(255,255,255));
    }
}
void Key::setKey(int sx,int sy, char color){
    x = sx;
    y = sy;
    keycolor = color;
}
void Key::Onshow(CMap* gamemap){
    if(isOnMap){ //有在地圖上才顯示
        keypicture.SetTopLeft(gamemap->computsx(x)*20,gamemap->computsy(y)*20);
        keypicture.ShowBitmap();
    }
}
bool Key::getisOnMap(){
    return isOnMap;
}
bool Key::hitThief(int sx,int sy){
    int tx1 = sx; //人物左邊界
    int tx2 = sx+3; //人物右邊界
    int ty1 = sy; //人物上邊界
    int ty2 = sy+4; //人物下邊界
    //判定是否碰撞
    if(((x>tx1&& x<tx2)|| (x+width>tx1&& x+width<tx2))&& ((y>ty1&& y<ty2)|| (y+high>ty1&& y+high<ty2))){
        return true;
    }
    else{
        return false;
    }
}
void Key::setOnMap(bool flag){
    isOnMap = flag;
}

////////////////////////////////////
// Door
////////////////////////////////////
Door::Door(){
}
void Door::Initialize(CMap* gamemap){
    isClose = false; //預設門是開的後
    setClose(true,gamemap); //將所有門關上(將門的位置都填上其代號);
}
void Door::LoadDoor(){
    if(!isVertical){
        if(doortype=='n') //N 為普通門
            doorpicture.LoadBitmap("Bitmaps/hdoor.bmp");
        else if(doortype=='k'){ //K 為鑰匙門

```

```

        if(doorcolor == 'y')
            doorpicture.LoadBitmap("Bitmaps/yellow-door-h.bmp");
        else if(doorcolor == 'g')
            doorpicture.LoadBitmap("Bitmaps/green-door-h.bmp");
        else if(doorcolor == 'r')
            doorpicture.LoadBitmap("Bitmaps/red-door-h.bmp");
        else if(doorcolor == 'p')
            doorpicture.LoadBitmap("Bitmaps/purple-door-h.bmp");
    }
}
else{
    if(doortype=='n')
        doorpicture.LoadBitmap("Bitmaps/vdoor.bmp");
    else if(doortype=='k'){
        if(doorcolor == 'y')
            doorpicture.LoadBitmap("Bitmaps/yellow-door-v.bmp");
        else if(doorcolor == 'g')
            doorpicture.LoadBitmap("Bitmaps/green-door-v.bmp");
        else if(doorcolor == 'r')
            doorpicture.LoadBitmap("Bitmaps/red-door-v.bmp");
        else if(doorcolor == 'p')
            doorpicture.LoadBitmap("Bitmaps/purple-door-v.bmp");
    }
}
}

void Door::setClose(bool flag,CMap* gamemap){
    if(flag&&isClose){          //若門的狀態與設定之狀態相同 就直接不執行
        return;
    }
    else if(flag&&!isClose){      //若不同 設定關門 則依照其類型 填回地圖
        isClose = flag;
        if(doortype=='n'&&isVertical){
            for(int i=y;i<y+doorsize;i++){
                gamemap->setdoor(x,i,'v');
            }
        }
        else if(doortype=='n'&&!isVertical){
            for(int i=x;i<x+doorsize;i++){
                gamemap->setdoor(i,y,'h');
            }
        }
        else if(doortype == 'k'){
            if(doorcolor == 'y'){
                if(isVertical){
                    for(int i=y;i<x+doorsize;i++){
                        gamemap->setdoor(x,i,'Y');
                    }
                }
                else{
                    for(int i=x;i<x+doorsize;i++){
                        gamemap->setdoor(i,y,'Z');
                    }
                }
            }
        }
        else if(doorcolor == 'r'){
            if(isVertical){
                for(int i=y;i<x+doorsize;i++){
                    gamemap->setdoor(x,i,'R');
                }
            }
            else{
                for(int i=x;i<x+doorsize;i++){
                    gamemap->setdoor(i,y,'S');
                }
            }
        }
        else if(doorcolor == 'g'){
            if(isVertical){
                for(int i=y;i<x+doorsize;i++){

```

```

        gamemap->setdoor(x,i,'G');
    }
    else{
        for(int i=x;i<x+doorsize;i++){
            gamemap->setdoor(i,y,'H');
        }
    }
}
else if(doorcolor == 'p'){
    if(isVertical){
        for(int i=y;i<x+doorsize;i++){
            gamemap->setdoor(x,i,'P');
        }
    }
    else{
        for(int i=x;i<x+doorsize;i++){
            gamemap->setdoor(i,y,'Q');
        }
    }
}
}
}
else if(!flag&&isClose){          //若設定為開 就將原本門的地方設為空白
    isClose = flag;
    if(isVertical){
        for(int i=y;i<y+doorsize;i++){
            gamemap->toempty(x,i);
        }
    }
    else{
        for(int i=x;i<x+doorsize;i++){
            gamemap->toempty(i,x);
        }
    }
}
else{
    return;
}
}
}
bool Door::HitObject(int sx,int sy){
    int tx1= sx;          //人物左邊界
    int tx2= sx+3;         //人物右邊界
    int ty1= sy;          //人物上邊界
    int ty2= sy+4;         //人物下邊界
    //判定碰撞
    if(isVertical){
        if((x>=tx1&&x<=tx2)&&((ty1>=y&&ty1<=y+doorsize-1)|| (ty2>=y&&ty2<=y+doorsize-1))){
            return true;
        }
        else{
            return false;
        }
    }
    else{
        if(((tx1>=x&&tx1<=x+doorsize-1)|| (tx2>=x&&tx2<=x+doorsize-1))&&(y>=ty1&&y<=ty2)){
            return true;
        }
        else{
            return false;
        }
    }
}
}
bool Door::GetisClose(){
    return isClose;
}
}
void Door::OnShow(CMap* gamemap){
    if(isClose){          //若門是關上的才顯示
        doorpicture.SetTopLeft(gamemap->computsx(x)*20,gamemap->computsy(y)*20);
    }
}

```

```

        doorpicture.ShowBitmap();
    }
}
void Door::setDoor(int sx, int sy, char type,char color){
    x=sx;
    y=sy;
    doorcolor = color;
    if(type == 'h'){
        isVertical = false;
        doortype = 'n';
    }
    else if(type == 'v'){
        isVertical = true;
        doortype = 'n';
    }
    else if(type == 'Y'){
        isVertical = true;
        doortype = 'k';
    }
    else if(type == 'Z'){
        isVertical = false;
        doortype = 'k';
    }
    else if(type == 'G'){
        isVertical = true;
        doortype = 'k';
    }
    else if(type == 'H'){
        isVertical = false;
        doortype = 'k';
    }
    else if(type == 'R'){
        isVertical = true;
        doortype = 'k';
    }
    else if(type == 'S'){
        isVertical = false;
        doortype = 'k';
    }
    else if(type == 'P'){
        isVertical = true;
        doortype = 'k';
    }
    else if(type == 'Q'){
        isVertical = false;
        doortype = 'k';
    }
}

}
char Door::Getdoorcolor(){
    return doorcolor;
}
char Door::Getdoortype(){
    return doortype;
}
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// 這個 class 為專屬於顯示計時器的所有方法
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
CintTime::CintTime(){
    Initialize();
}

void CintTime::Initialize()          //初始化
{
    min=sec=0;
}

void CintTime::SetTime(int number){  //設定時間,為秒
    if(number>=60){

```

```

        min=number/60;
        sec=number%60;
    }
    else{
        sec=number;
        min=0;
    }
}

int CintTime::GetTime(){                //取得目前時間,為秒
    return min*60+sec;
}

void CintTime::LoadBitmap(bool CTFirst){//載入圖形
    if(!CTFirst){
        Cint.LoadBitmap();
    }
    Cint.SetIsLoadBitmap(true);
}

void CintTime::OnShow(){                //顯示圖形

    //分
    Cint.SetInteger(min);
    Cint.SetTopLeft(270,10);
    Cint.ShowBitmap();

    //秒
    Cint.SetInteger(sec);
    Cint.SetTopLeft(350,10);
    Cint.ShowBitmap();
}

/////////////////////////////////////////////////////////////////
// 這個 class 為盜賊(Thief)的所有方法
/////////////////////////////////////////////////////////////////
Thief::Thief(){
    Initialize();
}

void Thief::Initialize()
{
    //主角位置預設為 1,1;
    x=1,y=1;
    //主角預設往下第一張圖片
    play = &rogue[0][0];
    step = 0;
    //預設移動狀態都是否
    isMovingLeft = isMovingRight= isMovingUp = isMovingDown = false;
}

void Thief::SetXY(int sx,int sy){
    x=sx;
    y=sy;
}

void Thief::LoadBitmap()
{
    rogue[0][0].LoadBitmap(IDB_SHARP_D1,RGB(255,255,255));
    rogue[0][1].LoadBitmap(IDB_SHARP_D2,RGB(255,255,255));
    rogue[0][2].LoadBitmap(IDB_SHARP_D3,RGB(255,255,255));
    rogue[0][3].LoadBitmap(IDB_SHARP_D4,RGB(255,255,255));
    rogue[1][0].LoadBitmap(IDB_SHARP_U1,RGB(255,255,255));
    rogue[1][1].LoadBitmap(IDB_SHARP_U2,RGB(255,255,255));
    rogue[1][2].LoadBitmap(IDB_SHARP_U3,RGB(255,255,255));
    rogue[1][3].LoadBitmap(IDB_SHARP_U4,RGB(255,255,255));
    rogue[2][0].LoadBitmap(IDB_SHARP_R1,RGB(255,255,255));
    rogue[2][1].LoadBitmap(IDB_SHARP_R2,RGB(255,255,255));
    rogue[2][2].LoadBitmap(IDB_SHARP_R3,RGB(255,255,255));
    rogue[2][3].LoadBitmap(IDB_SHARP_R4,RGB(255,255,255));
    rogue[3][0].LoadBitmap(IDB_SHARP_L1,RGB(255,255,255));
    rogue[3][1].LoadBitmap(IDB_SHARP_L2,RGB(255,255,255));
}

```



```

    rogue[3][2].LoadBitmap(IDB_SHARP_L3,RGB(255,255,255));
    rogue[3][3].LoadBitmap(IDB_SHARP_L4,RGB(255,255,255));
}
//用來重設圖片走路的方法(每次走路都不一樣的圖片,重複顯示)
void Thief::contralstep(int direction){
    step++;
    if(step>3)
        step=0;
    play=&rogue[direction][step];
}
void Thief::reststep(){
    step=0;
}
void Thief::OnMove(CMap* gamemap,Door* door,Key* key)
{
    //判斷盜賊是否開門
    for(int i=0;i<gamemap->Getcountdoors();i++){
        if(door[i].GetisClose()&&(door[i].HitObject(x,y))){
            door[i].setClose(false,gamemap);
            //開門音效
            CAudio::Instance()->Play(AUDIO_DOOR, false);
            break;
        }
    }
    if (isMovingLeft){
        if(checkempty('l',gamemap,key,door)&&x>0){
            x--;
            if(gamemap->computsx(x)<16)//移動視窗用(移動地圖讓地圖跟著人跑)
                gamemap->MoveMap('l');
        }
        contralstep(3);
    }
    else if (isMovingRight){
        if(checkempty('r',gamemap,key,door)&&x<156){
            x++;
            if(gamemap->computsx(x)>16)//移動視窗用(移動地圖讓地圖跟著人跑)
                gamemap->MoveMap('r');
        }
        contralstep(2);
    }
    else if (isMovingUp){
        if(checkempty('u',gamemap,key,door)&&y>0){
            y--;
            if(gamemap->computsy(y)<12)//移動視窗用(移動地圖讓地圖跟著人跑)
                gamemap->MoveMap('u');
        }
        contralstep(1);
    }
    else if (isMovingDown){
        if(checkempty('d',gamemap,key,door)&&y<116){
            y++;
            if(gamemap->computsy(y)>12)//移動視窗用(移動地圖讓地圖跟著人跑)
                gamemap->MoveMap('d');
        }
        contralstep(0);
    }
    else{}
}

//判斷按鍵的方向傳入布林值 T 為按下去,F 為否
void Thief::SetMovingDown(bool flag)
{
    isMovingDown = flag;
}
void Thief::SetMovingLeft(bool flag)
{
    isMovingLeft = flag;
}

```

```

void Thief::SetMovingRight(bool flag)
{
    isMovingRight = flag;
}
void Thief::SetMovingUp(bool flag)
{
    isMovingUp = flag;
}
//設定盜賊在螢幕上顯示的位置並顯示
void Thief::OnShow(CMap* gamemap)
{
    play->SetTopLeft(gamemap->computex(x)*20,gamemap->computey(y)*20);
    play->ShowBitmap();
}
//判定是否指定方向可以走
bool Thief::checkempty(char direction,CMap* gamemap,Key *key , Door *door){
    if(direction=='r'){ //判別往右走是否可走
        for(int i=y;i<=y+4;i++){
            if(gamemap->Isempy(x+4,i)==0)
                return false;
        }
        //判定右方的門是否可開啟
        for(int i=0;i< gamemap->Getcountdoors();i++){
            if(door[i].Getdoortype()=='k'&&door[i].HitObject(x+1,y)&&door[i].GetisClose()){
                if(gamemap->Getcountkeys() == 0){
                    return false;
                }
                for(int j=0;j<gamemap->Getcountkeys();j++){
                    if(door[i].Getdoorcolor()==key[j].getColor()){
                        if(key[j].getisOnMap()){
                            return false;
                        }
                    }
                    else{
                        return true;
                    }
                }
            }
        }
        return false;
    }
    return true;
}
else if(direction=='l'){ //判別往左走是否可走
    for(int i=y;i<=y+4;i++){
        if(gamemap->Isempy(x-1,i)==0)
            return false;
    }
    //判定左方的門是否可開啟
    for(int i=0;i< gamemap->Getcountdoors();i++){
        if(door[i].Getdoortype() == 'k' &&door[i].HitObject(x-1,y)&&door[i].GetisClose()){
            if(gamemap->Getcountkeys() ==0){
                return false;
            }
            for(int j=0;j<gamemap->Getcountkeys();j++){
                if(door[i].Getdoorcolor()==key[j].getColor()){
                    if(key[j].getisOnMap()){
                        return false;
                    }
                }
                else{
                    return true;
                }
            }
        }
    }
    return false;
}
return true;
}
}

```

```

else if(direction=='u'){    //判定上方的門是否可走
    for(int i=x;i<=x+3;i++){
        if(gamemap->Isempy(i,y-1)==0)
            return false;
    }
    //判斷上方的們是否可開啟
    for(int i=0;i< gamemap->Getcountdoors();i++){
        if(door[i].Getdoortype()=='k'&&door[i].HitObject(x,y-1)&&door[i].GetisClose()){
            if(gamemap->Getcountkeys() == 0){
                return false;
            }
            for(int j=0;j<gamemap->Getcountkeys();j++){
                if(door[i].Getdoorcolor()==key[j].getColor()){
                    if(key[j].getisOnMap()){
                        return false;
                    }
                    else{
                        return true;
                    }
                }
            }
            return false;
        }
    }
    return true;
}
else if(direction=='d'){    //判定下方的是否可走
    for(int i=x;i<=x+3;i++){
        if(gamemap->Isempy(i,y+5)==0)
            return false;
    }
    //判斷下方的們是否可開啟
    for(int i=0;i< gamemap->Getcountdoors();i++){
        if(door[i].Getdoortype()=='k'&&door[i].HitObject(x,y+1)&&door[i].GetisClose()){
            if(gamemap->Getcountkeys() == 0){
                return false;
            }
            for(int j=0;j<gamemap->Getcountkeys();j++){
                if(door[i].Getdoorcolor()==key[j].getColor()){
                    if(key[j].getisOnMap()){
                        return false;
                    }
                    else{
                        return true;
                    }
                }
            }
            return false;
        }
    }
    return true;
}
else
    return false;
}

int Thief::Getx(){
    return x;
}
int Thief::Gety(){
    return y;
}

////////////////////////////////////
// 這個 class 為怪物(Monster)的所有方法
////////////////////////////////////
//初始化
Monster::Monster()
{

```

```

    Initialize();
}
void Monster::Initialize()
{
    //怪物位置預設為 42 10;
    gx=12,gy=86;
    //怪物預設往下第一張圖片
    monster_1 = &guide[0][0];
    mstep = 0;
    //預設移動狀態都是否
    isMoving=false;
    isTrace=false;
    isBack=false;
    nowstep = 0;
    mcounter=0;
    path.clear();
    returnpath.clear();
    nowdirection='d';
    comenwalkstep = 0;
    isComeback = false;
    thedoor=NULL;
    opendoor = false;
    waitcounter = 0;
}
void Monster::LoadBitmap()
{
    guide[0][0].LoadBitmap(IDB_GUIDED1,RGB(255,255,255));
    guide[0][1].LoadBitmap(IDB_GUIDED2,RGB(255,255,255));
    guide[0][2].LoadBitmap(IDB_GUIDED3,RGB(255,255,255));
    guide[0][3].LoadBitmap(IDB_GUIDED4,RGB(255,255,255));
    guide[1][0].LoadBitmap(IDB_GUIDE1,RGB(255,255,255));
    guide[1][1].LoadBitmap(IDB_GUIDE2,RGB(255,255,255));
    guide[1][2].LoadBitmap(IDB_GUIDE3,RGB(255,255,255));
    guide[1][3].LoadBitmap(IDB_GUIDE4,RGB(255,255,255));
    guide[2][0].LoadBitmap(IDB_GUIDER1,RGB(255,255,255));
    guide[2][1].LoadBitmap(IDB_GUIDER2,RGB(255,255,255));
    guide[2][2].LoadBitmap(IDB_GUIDER3,RGB(255,255,255));
    guide[2][3].LoadBitmap(IDB_GUIDER4,RGB(255,255,255));
    guide[3][0].LoadBitmap(IDB_GUIDEL1,RGB(255,255,255));
    guide[3][1].LoadBitmap(IDB_GUIDEL2,RGB(255,255,255));
    guide[3][2].LoadBitmap(IDB_GUIDEL3,RGB(255,255,255));
    guide[3][3].LoadBitmap(IDB_GUIDEL4,RGB(255,255,255));
}
void Monster::contralstep(int direction){
    mstep++;
    if(mstep>3)
        mstep=0;
    if(direction==0 && nowdirection != 'd')
        mstep=0;
    else if(direction==1 && nowdirection != 'u')
        mstep=0;
    else if(direction==2 && nowdirection != 'r')
        mstep=0;
    else if(direction==3 && nowdirection != 'l')
        mstep=0;
    else{}
    monster_1=&guide[direction][mstep];
}
void Monster::reststep(){
    mstep=0;
}
void Monster::Setcomenpath(vector<MovingNode*> &scriptpath){
    comenpath = scriptpath;
}
void Monster::findpath(const int tx, const int ty, CMap* gamemap){
    if(tx==gx&&ty==gy){
        return; //若一開始就到達終點就不執行
    }
    list<MovingNode*> openlist; //開啟清單

```

```

list<MovingNode*> closelist; //以探索清單
MovingNode* expendpoint = new MovingNode(gx,gy); //目前拓展點
expendpoint->setparent(NULL); //一開始將拓展點的父點設為 NULL 後
expendpoint->caculateparameter(gx,gy,tx,ty); //計算參數
openlist.push_back(expendpoint); //推入開啟清單
list<MovingNode*>::iterator it; //宣告 list 的指標
MovingNode* targetpoint = NULL; //最終點
while(!openlist.empty()){ //若開啟清單不是空的就繼續指找尋
    for(it =openlist.begin(), expendpoint=*it ;it != openlist.end();it++){//找出評價分數最低的點
        MovingNode* now = *it;
        if(now->Getf()<expendpoint->Getf()){
            expendpoint=now;
        }
        else{
            if(now->Getf()==expendpoint->Getf()){
                if(now->Getg()<expendpoint->Getg()){
                    expendpoint=now;
                }
                else{
                }
            }
        }
    }
    if(expendpoint->Getx()==tx&&expendpoint->Gety()==ty){ //若達到終點 則離迴圈
        targetpoint = expendpoint;
        break;
    }
    //若右邊是可走的並且不再 CloseList 或 Openlist 中的 計算參數後 並將父點設為現在探索點 並計算參數後 放入 Openlist 中
    if(checkempty(expendpoint->Getx(),expendpoint->Gety(),'r',gamemap)){
        bool inlist=false;
        for(it = openlist.begin();it!=openlist.end();it++){
            MovingNode* now=*it;
            if(now->Getx() == (expendpoint->Getx()+1) && now->Gety() == expendpoint->Gety()){
                inlist=true;
                break;
            }
        }
        if(!inlist){
            for(it = closelist.begin();it!=closelist.end();it++){
                MovingNode* now=*it;
                if(now->Getx() == (expendpoint->Getx()+1) && now->Gety() == expendpoint->Gety()){
                    inlist=true;
                    break;
                }
            }
        }
        if(!inlist){
            MovingNode* temp= new MovingNode(expendpoint->Getx()+1,expendpoint->Gety());
            temp->setparent(expendpoint);
            temp->caculateparameter(gx,gy,tx,ty);
            openlist.push_back(temp);
        }
    }
    //若左邊是可走的並且不再 CloseList 或 Openlist 中的 計算參數後 並將父點設為現在探索點 並計算參數後 放入 Openlist 中
    if(checkempty(expendpoint->Getx(),expendpoint->Gety(),'l',gamemap)){
        bool inlist=false;
        for(it = openlist.begin();it!=openlist.end();it++){
            MovingNode* now = *it;
            if(now->Getx()==(expendpoint->Getx()-1)&& now->Gety() == expendpoint->Gety()){
                inlist=true;
                break;
            }
        }
        if(!inlist){
            for(it=closelist.begin();it!=closelist.end();it++){
                MovingNode* now = *it;
                if(now->Getx()==(expendpoint->Getx()-1)&& now->Gety() == expendpoint->Gety()){
                    inlist=true;
                    break;
                }
            }
        }
    }
}

```

```

    }
}
}
if(!inlist){
    MovingNode* temp= new MovingNode(expendpoint->Getx()-1,expendpoint->Gety());
    temp->setparent(expendpoint);
    temp->caculateparameter(gx,gy,tx,ty);
    openlist.push_back(temp);
}
}
//若下面是可走的並且不再 CloseList 或 Openlist 中的 計算參數後 並將父點設為現在探索點 並計算參數後 放入 Openlist
if(checkempty(expendpoint->Getx(),expendpoint->Gety(),'d',gamemap)){
    bool inlist=false;
    for(it=openlist.begin();it!=openlist.end();it++){
        MovingNode* now = *it;
        if(now->Getx()==expendpoint->Getx() && now->Gety() == (expendpoint->Gety()+1)){
            inlist=true;
            break;
        }
    }
    if(!inlist){
        for(it=closetlist.begin();it!=closetlist.end();it++){
            MovingNode* now=*it;
            if(now->Getx()==expendpoint->Getx()&& now->Gety() == (expendpoint->Gety()+1)){
                inlist=true;
                break;
            }
        }
    }
    if(!inlist){
        MovingNode* temp = new MovingNode(expendpoint->Getx(),expendpoint->Gety()+1);
        temp->setparent(expendpoint);
        temp->caculateparameter(gx,gy,tx,ty);
        openlist.push_back(temp);
    }
}
//若上方是可走的並且不再 CloseList 或 Openlist 中的 計算參數後 並將父點設為現在探索點 並計算參數後 放入 Openlist 中
if(checkempty(expendpoint->Getx(),expendpoint->Gety(),'u',gamemap)){
    bool inlist=false;
    for(it=openlist.begin();it!=openlist.end();it++){
        MovingNode* now = *it;
        if(now->Getx()==expendpoint->Getx()&&now->Gety() == (expendpoint->Gety()-1)){
            inlist=true;
            break;
        }
    }
    if(!inlist){
        for(it=closetlist.begin();it!=closetlist.end();it++){
            MovingNode* now = *it;
            if(now->Getx()==expendpoint->Getx() && now->Gety() == (expendpoint->Gety() -1)){
                inlist=true;
                break;
            }
        }
    }
    if(!inlist){
        MovingNode * temp = new MovingNode(expendpoint->Getx(),expendpoint->Gety()-1);
        temp->setparent(expendpoint);
        temp->caculateparameter(gx,gy,tx,ty);
        openlist.push_back(temp);
    }
}
closetlist.push_back(expendpoint); //將目前的點從 Openlist 中移除
openlist.remove(expendpoint); //將目前的探索點放入 CloseList 中
}
for(MovingNode* now = targetpoint; now->GetParent()!=NULL;now=now->GetParent()){ //從終點開始往回到起點 將路徑紀錄
    path.push_back(new MovingNode(now->Getx(),now->Gety()));
}

```

中

下來

```

    }
    for(it=openlist.begin();it!=openlist.end();it++){           //釋放所有在 OpenList 中的所有點
        MovingNode* now=*it;
        if(now!=NULL)
            delete now;
    }
    openlist.clear(); //清空 Openlist
    for(it=closetlist.begin();it!=closetlist.end();it++){       //釋放所有在 CloseList 中所有的點
        MovingNode* now=*it;
        if(now!=NULL)
            delete now;
    }
    closetlist.clear(); //清空 CloseList
}

bool Monster::checkempty(int x,int y,char direction,CMap* gamemap){
    if(direction=='r'){
        for(int i=y;i<=y+4;i++){
            if(gamemap->Iseempty(x+4,i)!=1&&gamemap->Iseempty(x+4,i)!=4)
                return false;
        }
        return true;
    }
    else if(direction=='l'){
        for(int i=y;i<=y+4;i++){
            if(gamemap->Iseempty(x-1,i)!=1&&gamemap->Iseempty(x-1,i)!=4)
                return false;
        }
        return true;
    }
    else if(direction=='u'){
        for(int i=x;i<=x+3;i++){
            if(gamemap->Iseempty(i,y-1)!=1&&gamemap->Iseempty(i,y-1)!=4)
                return false;
        }
        return true;
    }
    else if(direction=='d'){
        for(int i=x;i<=x+3;i++){
            if(gamemap->Iseempty(i,y+5)!=1&&gamemap->Iseempty(i,y+5)!=4)
                return false;
        }
        return true;
    }
    else
        return false;
}

void Monster::SetXY(int x,int y){
    gx = x;
    gy = y;
}

bool Monster::looksight(Thief* thief ,CMap* gamemap,bool cheatmode){
    if(cheatmode){           //若在作弊模式下一律回傳 false
        return false;
    }
    int mx1 = gx-sightrange;           //設定最左方邊界    最小值 0
    if(mx1 < 0){
        mx1 =0;
    }
    int my1 = gy-sightrange;           //設定最上方邊界    最小值 0
    if(my1 < 0){
        my1 =0;
    }
    int mx2 = gx+sightrange+3;         //設定最右方邊界    最大值 160
    if(mx2>160){
        mx2 = 160;
    }
}

```

```

int my2 = gy+sightrange+4;          //設定最下方邊界    最大值 160
if( my2 >120){
    my2 = 120;
}
int tx1 = thief->GetX();              //小偷左邊界
int ty1 = thief->GetY();              //小偷上邊界
int tx2 = thief->GetX()+3;           //小偷右邊界
int ty2 = thief->GetY()+4;           //小偷下邊界
//先判斷是否出現在 範圍內 若在範圍內 在判定是否能看得到
if(((tx1>=mx1&&tx1<=mx2)|| (tx2>=mx1&&ty2<=mx2))&&((ty1>=my1&&ty1<=my2)|| (ty2>=my1&&ty2<=my2))){
    if(nowdirection == 'r'){
        for(int i=gx;i<=mx2;i++){
            if(i>=gx+4&&(gamemap->Iseempty(i,gy-1)==0||gamemap->Iseempty(i,gy-1)==4)){
                break;
            }
            for(int j=i,k=gy-1;j<=mx2&&k>=my1;j++,k--){
                if(gamemap->Iseempty(j,k)==0||gamemap->Iseempty(j,k)==4){
                    break;
                }
                else{
                    if(j>=tx1&&j<=tx2&&k>=ty1&&k<=ty2){
                        return true;
                    }
                }
            }
        }
    }
    for(int i=gx;i<=mx2;i++){
        if(i>=gx+4&&(gamemap->Iseempty(i,gy+5)==0||gamemap->Iseempty(i,gy+5)==4)){
            break;
        }
        for(int j=i,k=gy+5;j<=mx2&&k<=my2;j++,k++){
            if(gamemap->Iseempty(j,k)==0||gamemap->Iseempty(j,k)==4){
                break;
            }
            else{
                if(j>=tx1&&j<=tx2&&k>=ty1&&k<=ty2){
                    return true;
                }
            }
        }
    }
    for(int i = gy;i<=gy+4;i++){
        for(int j=gx+4;i<=mx2;j++){
            if(gamemap->Iseempty(j,i)==0||gamemap->Iseempty(j,i)==4){
                break;
            }
            else{
                if(i>=ty1&&i<=ty2&&j>=tx1&&j<=tx2){
                    return true;
                }
            }
        }
    }
    return false;
}
else if(nowdirection == 'l'){
    for(int i=gx+3;i>=mx1;i--){
        if(i<=gx-1&&(gamemap->Iseempty(i,gy-1)==0||gamemap->Iseempty(i,gy-1)==4)){
            break;
        }
        for(int j=i,k=gy-1;j>=mx1&&k>=my1;j--,k--){
            if(gamemap->Iseempty(j,k)==0||gamemap->Iseempty(j,k)==4){
                break;
            }
            else{
                if(j>=tx1&&j<=tx2&&k>=ty1&&k<=ty2){
                    return true;
                }
            }
        }
    }
}

```



```

    }
}
for(int i=gx+3;i>=mx1;i--){
    if(i<=gx-1&&(gamemap->Isempy(i,gy+5)==0||gamemap->Isempy(i,gy+5)==4)){
        break;
    }
    for(int j=i,k=gy+5;j>=mx1&&k<=my2;j--,k++){
        if(gamemap->Isempy(j,k)==0||gamemap->Isempy(j,k)==4){
            break;
        }
        else{
            if(j>=tx1&&j<=tx2&&k>=ty1&&k<=ty2){
                return true;
            }
        }
    }
}
for(int i=gy;i>=gy+4;i++){
    for(int j=gx-1;i>=mx1;j--){
        if(gamemap->Isempy(j,i)==0||gamemap->Isempy(j,i)==4){
            break;
        }
        else{
            if(i>=ty1&&i<=ty2&&j>=tx1&&j<=tx2){
                return true;
            }
        }
    }
}
return false;
}
else if(nowdirection == 'u'){
    for(int i = gy+4;i>=my1;i--){
        if(i<=gy-1&&(gamemap->Isempy(gx+4,i)==0||gamemap->Isempy(gx+4,i)==4)){
            break;
        }
        for(int j=gx+4,k=i;j<=mx2&&k>=my1;j++,k--){
            if(gamemap->Isempy(j,k)==0||gamemap->Isempy(j,k)==4){
                break;
            }
            else{
                if(j>=tx1&&j<=tx2&&k>=ty1&&k<=ty2){
                    return true;
                }
            }
        }
    }
}
for(int i = gy+4;i>=my1;i--){
    if(i<=gy-1&&(gamemap->Isempy(gx-1,i)==0||gamemap->Isempy(gx-1,i)==4)){
        break;
    }
    for(int j=gx-1,k=i;j>=mx1&&k>=my1;j--,k--){
        if(gamemap->Isempy(j,k)==0||gamemap->Isempy(j,k)==4){
            break;
        }
        else{
            if(j>=tx1&&j<=tx2&&k>=ty1&&k<=ty2){
                return true;
            }
        }
    }
}
}
for(int i=gx;i<=gx+3;i++){
    for(int j=gy-1;i>=my1;j--){
        if(gamemap->Isempy(i,j)==0||gamemap->Isempy(i,j)==4){
            break;
        }
        else{
            if(i>=tx1&&i<=tx2&&j>=ty1&&j<=ty2){

```

```

        return true;
    }
}
}
return false;
}
else if(nowdirection == 'd'){
    for(int i=gy; i<=my2; i++){
        if(i>=gy+5 && (gamemap->Iseempty(gx+4,i)==0 || gamemap->Iseempty(gx+4,i)==4)){
            break;
        }
        for(int j=gx+4, k=i; j<=mx2 && k<=my2; j++, k++){
            if(gamemap->Iseempty(j,k)==0 || gamemap->Iseempty(j,k)==0){
                break;
            }
            else{
                if(j>=tx1 && j<=tx2 && k>=ty1 && k<=ty2){
                    return true;
                }
            }
        }
    }
}
for(int i=gy; i<=my2; i++){
    if(i>=gy+5 && (gamemap->Iseempty(gx-1,i)==0 || gamemap->Iseempty(gx-1,i)==4)){
        break;
    }
    for(int j=gx-1, k=i; j>=mx1 && k<=my2; j--, k++){
        if(gamemap->Iseempty(j,k)==0 || gamemap->Iseempty(j,k)==4){
            break;
        }
        else{
            if(j>=tx1 && j<=tx2 && k>=ty1 && k<=ty2){
                return true;
            }
        }
    }
}
}
for(int i=gx; i<=gx+3; i++){
    for(int j=gy+5; j<=my2; j++){
        if(gamemap->Iseempty(i,j)==0 || gamemap->Iseempty(i,j)==4){
            break;
        }
        else{
            if(i>=tx1 && i<=tx2 && j>=ty1 && j<=ty2){
                return true;
            }
        }
    }
}
}
return false;
}
return false;
}
else{
    return false;
}
}
}

void Monster::OnMove(CMap* gamemap, Thief* thief, Door* door, bool cheatmode)
{
    for(int i=0; i<gamemap->Getcountdoors(); i++){ //判斷人物有沒有開門
        if((door[i].HitObject(gx,gy))){
            if(door[i].GetisClose()){
                door[i].setClose(false, gamemap);
                opendoor = true; //有開門就將開門標記為 true
                thedoor = &door[i]; //紀錄剛才的那扇門
                break;
            }
        }
    }
}

```

```

}
if((!isTracellisBack)&&opendoor&&thedoor!=NULL){ //若不是在追蹤模式下 則開過門要關門
    if(thedoor->HitObject(gx,gy)==false){ //已經沒撞到門 設定門為關閉 然後將人物設為面相另一個方向
        thedoor->setClose(true,gamemap);
        if(nowdirection == 'r'){
            monster_l = &guide[3][0];
        }
        else if(nowdirection == 'l'){
            monster_l = &guide[2][0];
        }
        else if(nowdirection == 'd'){
            monster_l = &guide[1][0];
        }
        else{
            monster_l = &guide[0][0];
        }
        opendoor = false;
        wait = true; //啟動關門延遲
    }
}
if(wait){
    if(waitcounter<10){ //當延遲計數不到十 都是跳離 讓怪物不動
        waitcounter++;
        return;
    }
    else{
        waitcounter=0; //當延遲時間已屬到 10 表示延遲結束
        wait = false;
    }
}
}
if(!isMoving&&!isTrace&&!isBack){ //一般模式下 未移動
    if(looksight(thief,gamemap,cheatmode)){ //判斷是否小偷出現在視野放圍內 拐果是會進入追蹤模式
        isMoving=false;
        isTrace=true;
        nowstep=0;
    }
    else{ //不是則會去找尋下一個既定點移動過去
        findpath(commenpath[commenwalkstep]->GetX(),commenpath[commenwalkstep]->GetY(),gamemap);
        nowstep=0;
        isMoving = true;
        if(commenwalkstep+1 >= commenpath.size()&&!isComeback){ //若以達到既定點最後一點 會開始往回走
            isComeback = true;
        }
        else if(commenwalkstep==0 && isComeback){ //若是到達原點 則又開始會再往 另一邊走
            isComeback = false;
        }
        if(isComeback){
            commenwalkstep--;
        }
        else if(!isComeback){
            commenwalkstep++;
        }
    }
}
}
else if(isMoving && !isTrace&&!isBack){ //若是在一般模式移動
    if(nowstep<path.size()&&!path.empty()){
        if(mcounter<1){ //速度控制
            mcounter++;
            return;
        }
        else if(mcounter>=1){ //速度控制 執行一次 休息不執行一次
            mcounter=0;
        }
        //控制方向與移動
        if(path[path.size()-nowstep-1]->GetX() - gx ==1 && path[path.size()-nowstep-1]->GetY() == gy){
            contralstep(2);
            nowdirection = 'r';
        }
    }
}

```

```

else if(path[path.size()-nowstep-1]->GetX() - gx ==-1 && path[path.size()-nowstep-1]->GetY() == gy){
    contralstep(3);
    nowdirection = 'l';
}
else if(path[path.size()-nowstep-1]->GetX() == gx && path[path.size()-nowstep-1]->GetY()- gy ==1){
    contralstep(0);
    nowdirection = 'd';
}
else if(path[path.size()-nowstep-1]->GetX() == gx && path[path.size()-nowstep-1]->GetY()- gy ==-1){
    contralstep(1);
    nowdirection = 'u';
}
gx = path[path.size()-nowstep-1]->GetX();
gy = path[path.size()-nowstep-1]->GetY();
nowstep++;
if(looksight(thief,gamemap,cheatmode)){           //若小偷出現在視野範圍內 啟動追蹤模式
    isMoving=false;
    isTrace = true;
    for(unsigned int i=0;i<path.size();i++){       //釋放掉路徑
        if(path[i]!=NULL)
            delete path[i];
    }
    nowstep=0;
    path.clear();
}
if(nowstep>=path.size()){                           //若以移動到指定點 釋放掉路徑 切回非移動模式
    isMoving=false;
    for(unsigned int i=0;i<path.size();i++){
        if(path[i]!=NULL)
            delete path[i];
    }
    nowstep=0;
    path.clear();
}
}
else{                                                //若路徑是空的，清空路徑後 切回非移動模式
    if(looksight(thief,gamemap,cheatmode)){
        isTrace = true;
    }
    isMoving=false;
    nowstep=0;
    path.clear();
}
}
else if(!isMoving&&isTrace&&!isBack){              //若在追蹤模式下 尚未移動中
    findpath(thief->GetX(),thief->GetY(),gamemap); //找尋目前小偷位置 切換到移動中
    nowstep=0;
    isMoving=true;
}
else if(!isBack&&isTrace&&isMoving){               //若在追蹤模式下 移動中
    if(cheatmode){                                  //若開啟作弊模式 則清空路徑 切到返回模式
        isBack=true;
        isMoving=false;
        isTrace=false;
        for(unsigned int i=0;i<path.size();i++){
            if(path[i]!=NULL)
                delete path[i];
        }
        nowstep=0;
        path.clear();
    }
    if(nowstep<path.size()&&!path.empty()){
        int mxl = gx-sightrange;    //警戒左邊界
        if(mxl < 0){
            mxl = 0;
        }
        int myl = gy-sightrange;    //警戒上邊界
        if(myl < 0){
            myl = 0;
        }
    }
}

```

```

    }
    int mx2 = gx+sightrange;    //警戒右邊界
    if(mx2>160){
        mx2 = 160;
    }
    int my2 = gy+sightrange;    //警戒下邊界
    if( my2 >120){
        my2 = 120;
    }
    int tx1 = thief->Getx();    //小偷左邊界
    int ty1 = thief->Gety();    //小偷上邊界
    int tx2 = thief->Getx()+3;  //小偷右邊界
    int ty2 = thief->Gety()+4;  //小偷左邊界
    if(mcounter<10){           //TRACE SPEED
        mcounter++;
    }
    else if(mcounter>=10){//TRACE SPEED
        mcounter=0;
        return;
    }
    if(mstep%2==0){
        returnpath.push_back(new
MovingNode(path[path.size()-nowstep-1]->Getx(),path[path.size()-nowstep-1]->Gety())); //每走兩步紀錄移動位置
    }
    //下面為控制方向與移動
    if(path[path.size()-nowstep-1]->Getx() - gx ==1 && path[path.size()-nowstep-1]->Gety() == gy){
        contralstep(2);
        nowdirection = 'r';
    }
    else if(path[path.size()-nowstep-1]->Getx() - gx ==-1 && path[path.size()-nowstep-1]->Gety() == gy){
        contralstep(3);
        nowdirection = 'l';
    }
    else if(path[path.size()-nowstep-1]->Getx() == gx && path[path.size()-nowstep-1]->Gety()- gy ==1){
        contralstep(0);
        nowdirection = 'd';
    }
    else if(path[path.size()-nowstep-1]->Getx() == gx && path[path.size()-nowstep-1]->Gety()- gy ==-1){
        contralstep(1);
        nowdirection = 'u';
    }
    gx = path[path.size()-nowstep-1]->Getx();
    gy = path[path.size()-nowstep-1]->Gety();
    nowstep++;
    if(!(((tx1>=mx1&&tx1<=mx2)||((tx2>=mx1&&ty2<=mx2))&&((ty1>=my1&&ty1<=my2)||((ty2>=my1&&ty2<=my2))))){ //若小偷
離開警戒範圍內 進入 返回 未移動狀態
        isBack=true;
        isMoving=false;
        isTrace=false;
        for(unsigned int i=0;i<path.size();i++){ //清空探詢路徑
            if(path[i]!=NULL)
                delete path[i];
        }
        nowstep=0;
        path.clear();
    }
}
if(nowstep>=path.size()){ //若是走到定點 釋放路徑 回到 追蹤未移動 狀態
    isMoving=false;
    for(unsigned int i=0;i<path.size();i++){
        if(path[i]!=NULL)
            delete path[i];
    }
    nowstep=0;
    path.clear();
}
}
else if(isBack&&!isMoving&&!isTrace){ //返回模式 非移動中
    if(returnpath.size(>0){ //若返回路徑不是空的

```

```

path.clear(); //清除探詢路徑
findpath(returnpath[returnpath.size()-1]->GetX(),returnpath[returnpath.size()-1]->GetY(),gamemap); //尋找 往回
的下一點
    nowstep=0;
    isMoving=true;
}
else { //若返回路徑是空的 表示已到達原點 就切入 一般非移動模式
    nowstep=0;
    isBack=false;
    isMoving=false;
}
}
else if(!isTrace &&isBack&&isMoving){ //返回模式 移動中
    if(nowstep<path.size()&&!path.empty()){
        if(mcounter<4){ //速度控制
            mcounter++;
        }
        else if(mcounter>=4){
            mcounter=0;
            return;
        }
        //移動與控制方向
        if(path[path.size()-nowstep-1]->GetX() - gx ==1 && path[path.size()-nowstep-1]->GetY() == gy){
            contralstep(2);
            nowdirection = 'r';
        }
        else if(path[path.size()-nowstep-1]->GetX() - gx ==-1 && path[path.size()-nowstep-1]->GetY() == gy){
            contralstep(3);
            nowdirection = 'l';
        }
        else if(path[path.size()-nowstep-1]->GetX() == gx && path[path.size()-nowstep-1]->GetY()- gy ==1){
            contralstep(0);
            nowdirection = 'd';
        }
        else if(path[path.size()-nowstep-1]->GetX() == gx && path[path.size()-nowstep-1]->GetY()- gy ==-1){
            contralstep(1);
            nowdirection = 'u';
        }
        gx = path[path.size()-nowstep-1]->GetX();
        gy = path[path.size()-nowstep-1]->GetY();
        nowstep++;
        if(looksight(thief,gamemap,cheatmode)){ //若小偷再次出現於視線範圍內 則在切換是 追蹤 非移動 狀態 並
釋放探詢路徑(返回路徑不是放)
            isBack=false;
            isMoving=false;
            isTrace=true;
            for(unsigned int i=0;i<path.size();i++){
                if(path[i]!=NULL)
                    delete path[i];
            }
            nowstep=0;
            path.clear();
        }
    }
}
if(nowstep>=path.size()){ //若是走到探詢路徑的最後個點 切回 返回模式 非移動狀態
    isMoving=false;
    for(unsigned int i=0;i<path.size();i++){ //釋放路徑
        if(path[i]!=NULL)
            delete path[i];
    }
    delete returnpath[returnpath.size()-1]; //釋放已到達的返回點
    returnpath.pop_back(); //從路徑中釋放出來
    nowstep=0;
    path.clear();
}
}
}
void Monster::OnShow(CMap* gamemap)
{

```

```

monster_1->SetTopLeft(gamemap->computex(gx)*20,gamemap->computsy(gy)*20);
monster_1->ShowBitmap();
}
int Monster::Getx(){
    return gx;
}
int Monster::Gety(){
    return gy;
}
bool Monster::GetisTrace(){
    return isTrace;
}

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

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

void CGameStateInit::OnInit(){
    //
    // 當圖很多時，OnInit 載入所有的圖要花很多時間。為避免玩遊戲的人
    // 等的不耐煩，遊戲會出現「Loading ...」，顯示 Loading 的進度。
    //
    ShowInitProgress(0);    // 一開始的 loading 進度為 0%
    //
    // 開始載入資料
    //
    stbg.LoadBitmap(IDB_STBG);
    About.LoadBitmap("Bitmaps/ABOUT.bmp");           //載入關於製作
    ExMouse.LoadBitmap("Bitmaps/SELECT_ICON.bmp",RGB(255,255,255)); //箭頭圖案
    ExMouse.SetTopLeft(250,250);                      //先設定座標
    // 此 OnInit 動作會接到 CGameStaterRun::OnInit()，所以進度還沒到 100%
    //
    //載入所有音效檔案，僅在第一次執行
    CAudio::Instance()->Load(AUDIO_STBGS, "sounds\\STARTbgs.mp3"); // 載入 0 - 遊戲開始畫面背景音樂
    CAudio::Instance()->Load(AUDIO_SELECT, "sounds\\SELECT.wav"); // 載入 1 - 選單音效
    CAudio::Instance()->Load(AUDIO_SCUESS, "sounds\\SCUESS.wav"); // 載入 2 - 成功過關音效
    CAudio::Instance()->Load(AUDIO_FAILED, "sounds\\FAILED.wav"); // 載入 3 - 失敗(GAME OVER)頁面音樂
    CAudio::Instance()->Load(AUDIO_PICK, "sounds\\PICK.wav"); // 載入 4 - 撿起道具聲音
    CAudio::Instance()->Load(AUDIO_DOOR, "sounds\\DOOR.wav"); // 載入 5 - 開門聲音
    CAudio::Instance()->Load(AUDIO_MAP1, "sounds\\map1.mp3"); // 載入 6 - 地圖 1 音樂
    CAudio::Instance()->Load(AUDIO_MAP2, "sounds\\map2.mp3"); // 載入 7 - 地圖 2 音樂
    CAudio::Instance()->Load(AUDIO_MAP3, "sounds\\map3.mp3"); // 載入 8 - 地圖 3 音樂
    CAudio::Instance()->Load(AUDIO_MAP4, "sounds\\map4.mp3"); // 載入 9 - 地圖 4 音樂
    CAudio::Instance()->Load(AUDIO_MAP5, "sounds\\map5.mp3"); // 載入 10- 地圖 5 音樂
    CAudio::Instance()->Load(AUDIO_KEY, "sounds\\KEY.mp3"); // 載入 10- 地圖 5 音樂
    mu=true; //判斷是否是重新開始
    ShowInitProgress(20);    //
}

void CGameStateInit::OnBeginState(){
    if(mu==true){//重新開始才執行撥放音樂
        CAudio::Instance()->Play(AUDIO_STBGS, false);
    }
    About_bool = false;
}

void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags){
    const char KEY_UP = 0x26; // keyboard 上箭頭
    const char KEY_DOWN = 0x28; // keyboard 下箭頭
    const char KEY_SPACE = ' '; // keyboard 空白

    if(((nChar == KEY_UP)&&(ExMouse.Top()!=250))||((nChar == KEY_DOWN)&&(ExMouse.Top()!=379))){
        CAudio::Instance()->Play(AUDIO_SELECT, false);
    }
    //執行選單動作

```

```

if (nChar == KEY_SPACE){
    if(ExMouse.Top()==250){
        GotoGameState(GAME_STATE_RUN); // 切換至 GAME_STATE_RUN(開始第一關遊戲)
    }
    else if(ExMouse.Top()==293){          //選擇關卡
        GotoGameState(GAME_STATE_Select); // 切換至 GAME_STATE_Select
    }
    else if(ExMouse.Top()==336){          //關於製作
        if(About_bool){
            About_bool=false;
        }else{
            About_bool=true;
        }
    }
    else if(ExMouse.Top()==379){          //離開遊戲
        PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE,0,0);
    }
}
else if((nChar == KEY_UP)&&(ExMouse.Top()>250)){ //往上移動失效
    ExMouse.SetTopLeft(250,ExMouse.Top()-43);
}
else if(nChar == KEY_DOWN&&(ExMouse.Top()<379)){ //往下移動失效
    ExMouse.SetTopLeft(250,ExMouse.Top()+43);
}
}

void CGameStateInit::OnShow(){

    //貼背景圖
    stbg.SetTopLeft(0,0);
    stbg.ShowBitmap();
    //貼選單
    ExMouse.ShowBitmap(); //箭頭圖案
    if(About_bool==true){
        About.SetTopLeft(0,0);
        About.ShowBitmap();
    }else{

    }
}

////////////////////////////////////
// 這個 class 為遊戲的遊戲選擇關卡畫面
////////////////////////////////////

CGameStateSelect::CGameStateSelect(CGame *g):CGameState(g){
}

void CGameStateSelect::OnInit(){
    Select.LoadBitmap("Bitmaps/Select_bg.bmp");
    ExMouse.LoadBitmap("Bitmaps/SELECT_ICON.bmp",RGB(255,255,255)); //箭頭圖案
    ExMouse.SetTopLeft(250,293); //先設定座標
    ShowInitProgress(50); // 一開始的 loading 進度為 0%
}

void CGameStateSelect::OnBeginState(){
}

void CGameStateSelect::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags){
    const char KEY_UP = 0x26; // keyboard 上箭頭
    const char KEY_DOWN = 0x28; // keyboard 下箭頭
    const char KEY_SPACE = ' '; // keyboard 空白
    const char KEY_Q = 0x51; // keyboard

    if(((nChar == KEY_UP)&&(ExMouse.Top()!=293))||((nChar == KEY_DOWN)&&(ExMouse.Top()!=336))){
        CAudio::Instance()->Play(AUDIO_SELECT, false);
    }
}

```



```

//執行選單動作
if (nChar == KEY_SPACE){
    if(ExMouse.Top()==293){
        GotoGameState(GAME_STATE_RUN); // 第1關
    }
    else if(ExMouse.Top()==336){ // 第2關
        GotoGameState(GAME_STATE_RUN2); // 第1關
    }
}
}
else if((nChar == KEY_UP)&&(ExMouse.Top()>293)){ //往上移動失效
    ExMouse.SetTopLeft(250,ExMouse.Top()-43);
}
else if(nChar == KEY_DOWN&&(ExMouse.Top()<336)){ //往下移動失效
    ExMouse.SetTopLeft(250,ExMouse.Top()+43);
}
else if(nChar == KEY_Q){
    GotoGameState(GAME_STATE_INIT);
}
}
}

void CGameStateSelect::OnShow(){

    //貼背景圖
    Select.SetTopLeft(0,0);
    Select.ShowBitmap();
    //貼選單
    ExMouse.ShowBitmap();//箭頭圖案
}

////////////////////////////////////
// 這個 class 為遊戲的破關狀態(Game WIN)
////////////////////////////////////

CGameStateWIN::CGameStateWIN(CGame *g): CGameState(g){
}

void CGameStateWIN::OnMove()
{
}

void CGameStateWIN::OnBeginState()
{
    CAudio::Instance()->Stop(6);//map1 audio
    CAudio::Instance()->Stop(7);//map2 audio
    CAudio::Instance()->Play(AUDIO_SCUESS, false);
}

void CGameStateWIN::OnInit(){
    WIN.LoadBitmap("Bitmaps/WIN.bmp");
    ShowInitProgress(60); // 一開始的 loading 進度為 0%
}

void CGameStateWIN::OnShow()
{
    WIN.SetTopLeft(0,0);
    WIN.ShowBitmap();
}

void CGameStateWIN::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags){
    const char KEY_SPACE = ' ';
    if (nChar == KEY_SPACE){
        GotoGameState(GAME_STATE_INIT); // 切換至 GAME_STATE_RUN
    }
}

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

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

void CGameStateOver::OnMove()

```

```

{
    counter--;
    if (counter < 0){        //若 counter 歸 0 自動跳到選單
        CAudio::Instance()->Stop(3);
        GotoGameState(GAME_STATE_INIT);
    }
}

void CGameStateOver::OnBeginState()
{
    counter = 30 * 9;    //設定倒數秒數 9 秒
    CAudio::Instance()->Stop(6);
    CAudio::Instance()->Play(AUDIO_FAILED, true);
}

void CGameStateOver::OnInit()
{
    //
    // 開始載入資料
    ShowInitProgress(100);
    CAudio::Instance()->Play(AUDIO_STBGS, true);        // 撥放[開始遊戲]背景音樂
    Over.LoadBitmap(IDB_OVER);                        //結束頁面底圖
}

void CGameStateOver::OnShow()
{
    Over.SetTopLeft(0,0);
    Over.ShowBitmap();
    CDC *pDC = CDDraw::GetBackCDC();                // 取得 Back Plain 的 CDC
    CFont f,*fp;
    f.CreatePointFont(160,"微軟正黑體");            // 產生 font f; 160 表示 16 point 的字
    fp=pDC->SelectObject(&f);                        // 選用 font f
    pDC->SetBkColor(RGB(0,0,0));
    pDC->SetTextColor(RGB(255,255,255));
    char str[80];                                    // Demo 數字對字串的轉換
    sprintf(str, "%d", counter / 30);
    pDC->TextOut(623,455,str);
    pDC->SelectObject(fp);                            // 放掉 font f (千萬不要漏了放掉)
    CDDraw::ReleaseBackCDC();                        // 放掉 Back Plain 的 CDC
}

void CGameStateOver::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags){
    const char KEY_SPACE = ' ';
    if (nChar == KEY_SPACE){
        CAudio::Instance()->Stop(3);
        GotoGameState(GAME_STATE_INIT);        // 切換至 GAME_STATE_RUN
    }
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的遊戲執行物件(過關畫面)
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
CGameStatePASS::CGameStatePASS(CGame *g):CGameState(g){
}

void CGameStatePASS::OnInit(){
    PASS.LoadBitmap(IDB_PASS);
    //剩餘時間的數字圖
    t0.LoadBitmap("Bitmaps/t0.bmp",RGB(255,255,255));
    t1.LoadBitmap("Bitmaps/t1.bmp",RGB(255,255,255));
    t2.LoadBitmap("Bitmaps/t2.bmp",RGB(255,255,255));
    t3.LoadBitmap("Bitmaps/t3.bmp",RGB(255,255,255));
    t4.LoadBitmap("Bitmaps/t4.bmp",RGB(255,255,255));
    t5.LoadBitmap("Bitmaps/t5.bmp",RGB(255,255,255));
    t6.LoadBitmap("Bitmaps/t6.bmp",RGB(255,255,255));
    t7.LoadBitmap("Bitmaps/t7.bmp",RGB(255,255,255));
    t8.LoadBitmap("Bitmaps/t8.bmp",RGB(255,255,255));
    t9.LoadBitmap("Bitmaps/t9.bmp",RGB(255,255,255));
}

```

```

//載入成績顯示圖形
CiST.LoadBitmap();
mu=true;//判斷是否已經執行到這裡
ShowInitProgress(40); //一開始的 loading 進度為 0%
}

void CGameStatePASS::OnBeginState(){
    CAudio::Instance()->Stop(6);//map1 audio
    CAudio::Instance()->Stop(7);//map2 audio
    if(mu==true) //第二次重玩才會執行
        CAudio::Instance()->Play(AUDIO_SCUESS, false);
}

void CGameStatePASS::CalST(){
    //讀取檔案時間與分數
    readfromfile();
    //將時間轉回成分秒制
    if(time>=60){
        min=time/60;
        sec=time%60;
    }
    else{
        sec=time;
        min=0;
    }
    min1=min/10;
    min2=min%10;
    sec1=sec/10;
    sec2=sec%10;
    //載入成績成圖片
    CiST.SetInteger(score);
}

void CGameStatePASS::OnShow(){
    //設定背景圖顯示與位置
    PASS.ShowBitmap();
    PASS.SetTopLeft(0,0);
    /////成績顯示
    CiST.SetTopLeft(227,320);
    CiST.ShowBitmap();

    //呼叫計算成績與時間
    CalST();
    //分的第一位數顯示
    if(min1==1){
        t1.SetTopLeft(305,178);
        t1.ShowBitmap();
    }else if(min1==2){
        t2.SetTopLeft(303,178);
        t2.ShowBitmap();
    }else if(min1==3){
        t3.SetTopLeft(303,178);
        t3.ShowBitmap();
    }else if(min1==4){
        t4.SetTopLeft(303,178);
        t4.ShowBitmap();
    }else if(min1==5){
        t5.SetTopLeft(303,178);
        t5.ShowBitmap();
    }else if(min1==6){
        t6.SetTopLeft(303,178);
        t6.ShowBitmap();
    }else if(min1==7){
        t7.SetTopLeft(303,178);
        t7.ShowBitmap();
    }else if(min1==8){
        t8.SetTopLeft(303,178);
        t8.ShowBitmap();
    }else if(min1==9){

```

```

        t9.SetTopLeft(303,178);
        t9.ShowBitmap();
    }
    //分的第二位數顯示
    if(min2<1){
        t0.SetTopLeft(323,178);
        t0.ShowBitmap();
    }else if(min2==1){
        t1.SetTopLeft(323,178);
        t1.ShowBitmap();
    }else if(min2==2){
        t2.SetTopLeft(323,178);
        t2.ShowBitmap();
    }else if(min2==3){
        t3.SetTopLeft(323,178);
        t3.ShowBitmap();
    }else if(min2==4){
        t4.SetTopLeft(323,178);
        t4.ShowBitmap();
    }else if(min2==5){
        t5.SetTopLeft(323,178);
        t5.ShowBitmap();
    }else if(min2==6){
        t6.SetTopLeft(323,178);
        t6.ShowBitmap();
    }else if(min2==7){
        t7.SetTopLeft(323,178);
        t7.ShowBitmap();
    }else if(min2==8){
        t8.SetTopLeft(323,178);
        t8.ShowBitmap();
    }else if(min2==9){
        t9.SetTopLeft(323,178);
        t9.ShowBitmap();
    }
    //秒的第一位數顯示
    if(sec1==0){
        t0.SetTopLeft(388,178);
        t0.ShowBitmap();
    }else if(sec1==1){
        t1.SetTopLeft(388,178);
        t1.ShowBitmap();
    }else if(sec1==2){
        t2.SetTopLeft(388,178);
        t2.ShowBitmap();
    }else if(sec1==3){
        t3.SetTopLeft(388,178);
        t3.ShowBitmap();
    }else if(sec1==4){
        t4.SetTopLeft(388,178);
        t4.ShowBitmap();
    }else if(sec1==5){
        t5.SetTopLeft(388,178);
        t5.ShowBitmap();
    }else if(sec1==6){
        t6.SetTopLeft(388,178);
        t6.ShowBitmap();
    }else if(sec1==7){
        t7.SetTopLeft(388,178);
        t7.ShowBitmap();
    }else if(sec1==8){
        t8.SetTopLeft(388,178);
        t8.ShowBitmap();
    }else if(sec1==9){
        t9.SetTopLeft(388,178);
        t9.ShowBitmap();
    }
    //秒的第二位數顯示
    if(sec2==0){

```

```

        t0.SetTopLeft(408,178);
        t0.ShowBitmap();
    }else if(sec2==1){
        t1.SetTopLeft(408,178);
        t1.ShowBitmap();
    }else if(sec2==2){
        t1.SetTopLeft(408,178);
        t2.ShowBitmap();
    }else if(sec2==3){
        t3.SetTopLeft(408,178);
        t3.ShowBitmap();
    }else if(sec2==4){
        t4.SetTopLeft(408,178);
        t4.ShowBitmap();
    }else if(sec2==5){
        t5.SetTopLeft(408,178);
        t5.ShowBitmap();
    }else if(sec2==6){
        t6.SetTopLeft(408,178);
        t6.ShowBitmap();
    }else if(sec2==7){
        t7.SetTopLeft(408,178);
        t7.ShowBitmap();
    }else if(sec2==8){
        t8.SetTopLeft(408,178);
        t8.ShowBitmap();
    }else if(sec2==9){
        t9.SetTopLeft(408,178);
        t9.ShowBitmap();
    }
}

void CGameStatePASS::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags){
    const char KEY_SPACE = ' ';
    if ((nChar == KEY_SPACE)){
        if(stage==1)
            GotoGameState(GAME_STATE_RUN2);    // 切換至 GAME_STATE_RUN2
        else if(stage==2)
            GotoGameState(GAME_STATE_WIN);    // 切換至破關畫面
    }
}

void CGameStatePASS::readfromfile(){          //讀取分數檔案
    ifstream inf;
    inf.open("map/ST.bin",ios::binary);
    GAME_ASSERT(inf,"儲存遊戲時間/得分失敗!\n(不影響遊戲進行，可進到下一關)");
    inf.read((char*)&score,sizeof(int));
    inf.read((char*)&time,sizeof(int));
    inf.read((char*)&stage,sizeof(int));
    inf.close();
}

////////////////////////////////////
// 這個 class 為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
////////////////////////////////////

CGameStateRun::CGameStateRun(CGame *g)
: CGameState(g),fn("map/map1.txt"),gamemap(fn)    //設定該關卡的地圖檔案
{
    obstacles = new Obstacle[gamemap.Getcountobstacle()]; //產生障礙物件
    monster = new Monster[4];                               //產生怪物數量為 4
    treasure = new Treasure[gamemap.Getcounttreasure()]; //產生寶物物件
    door = new Door[gamemap.Getcountdoors()];             //產生門物件
    if(gamemap.Getcountkeys()!=0){
        key = new Key[gamemap.Getcountkeys()];
    }
    else{
        key = NULL;
    }
}

```

```

    }
}

CGameStateRun::~CGameStateRun()
{
    delete [] obstacles;
    delete [] monster;
    delete [] treasure;
    delete [] door;
    delete [] key;
}

void CGameStateRun::OnBeginState()
{
    score=0; //該關關卡成績計算初始為 0
    gamemap.Initialize(fn); //初始化遊戲地圖
    thief.Initialize(); //初始化盜賊
    redscreen.Initialize(); //初始化紅幕閃爍
    help.Initialize(); //初始化遊戲說明
    for(int i=0; i<4; i++){
        monster[i].Initialize();
    }
    for(int i=0; i<gamemap.Getcounttreasure(); i++){
        treasure[i].Initialize();
    }
    for(int i=0; i<gamemap.Getcountdoors(); i++){
        door[i].Initialize(&gamemap);
    }
    for(int i=0; i<gamemap.Getcountkeys(); i++){
        key[i].Initialize();
    }
    //設定怪物行走路徑
    int path0[9][2]={57,8},{77,8},{97,8},{123,8},{123,20},{103,20},{83,20},{63,20},{57,20}};
    int path1[8][2]={147,27},{127,27},{107,27},{87,27},{67,27},{47,27},{23,27},{23,32}};
    int path2[11][2]={1,38},{1,50},{21,50},{41,50},{61,50},{81,50},{101,50},{112,50},{112,38},{132,38},{154,38}};
    int path3[11][2]={1,59},{22,59},{22,83},{44,83},{64,83},{64,94},{84,94},{104,94},{124,94},{144,94},{152,94}};
    //生設定怪物行走路徑
    monster[0].SetXY(path0[0][0],path0[0][1]);
    monster[1].SetXY(path1[0][0],path1[0][1]);
    monster[2].SetXY(path2[0][0],path2[0][1]);
    monster[3].SetXY(path3[0][0],path3[0][1]);
    //產生怪物行走路徑 Link
    vector<MovingNode*> scriptpath0;
    vector<MovingNode*> scriptpath1;
    vector<MovingNode*> scriptpath2;
    vector<MovingNode*> scriptpath3;
    //依序`放入 Link
    for(int i=0; i<9; i++){
        scriptpath0.push_back(new MovingNode(path0[i][0],path0[i][1]));
    }
    for(int i=0; i<8; i++){
        scriptpath1.push_back(new MovingNode(path1[i][0],path1[i][1]));
    }
    for(int i=0; i<11; i++){
        scriptpath2.push_back(new MovingNode(path2[i][0],path2[i][1]));
    }
    for(int i=0; i<11; i++){
        scriptpath3.push_back(new MovingNode(path3[i][0],path3[i][1]));
    }

    //設定入既定入路徑
    monster[0].Setcommenpath(scriptpath0);
    monster[1].Setcommenpath(scriptpath1);
    monster[2].Setcommenpath(scriptpath2);
    monster[3].Setcommenpath(scriptpath3);
    //關閉開始畫面音樂,播放第一關音樂
    CAudio::Instance()->Stop(0);
    CAudio::Instance()->Play(AUDIO_MAP1, true);
    gamemap.Setxsy(thief.Getx()-16,thief.Gety()-12); //產生盜賊的位置
    Tcc.SetTime(limatetime); //設定關卡遊戲時間
}

```

```

counter = 30 * Tcc.GetTime(); //設定遊戲關卡可以玩的時間,時間到進入遊戲失敗頁
treasure_remind = gamemap.Getcounttreasure(); //取得寶物總數量
cheatmode = false; //作弊模式預設為不啟動
dkeybordisdown = false;
wkeybordisdown = false;
togetalltreasure = false;
}

void CGameStateRun::OnMove() // 移動遊戲元素
{
    if(!help.GetState()){
        return;
    }
    if(togetalltreasure&&treasure_remind!=0){
        treasure_remind = 0;
        for(int i=0;i<gamemap.Getcounttreasure();i++){
            treasure[i].Setlife(false);
        }
    }
    counter--; //時間計算依照每秒 30 運行次數倒數
    thief.OnMove(&gamemap,door,key);
    for(int i=0;i<4;i++){
        monster[i].OnMove(&gamemap,&thief,door,cheatmode);
    }
    for(int i=0;i<gamemap.Getcounttreasure();i++){
        if(treasure[i].Isalive()&&treasure[i].HitThief(&thief)){//寶物存活且被盜賊擊中
            CAudio::Instance()->Play(AUDIO_PICK, false);
            treasure_remind--;
            treasure[i].Setlife(false);//將寶物設定已取得
            score+=1000; //增加成績分數(寶物)
        }
    }
    for(int i=0;i<gamemap.Getcountkeys();i++){
        if(key[i].hitThief(thief.Getx(),thief.Gety())&&key[i].getisOnMap()){
            CAudio::Instance()->Play(AUDIO_KEY, false); //播放 KEY 音效
            key[i].setOnMap(false);
            score+=15; //增加成績分數(KEY)
        }
    }
    if(cheatmode == false){
        for(int i=0;i<4;i++){
            if(thief.Getx()-monster[i].Getx())>=-4&&thief.Getx()-monster[i].Getx()<=4&&thief.Gety()-monster[i].Gety()<=5&&thief.Gety()-monster[i].Gety()>=-5){
                CAudio::Instance()->Play(AUDIO_MAP1, false);
                redscreen.SetisFlash(false);
                GotoGameState(GAME_STATE_OVER);
            }
        }
    }
    for(int i=0;i<4;i++){
        if(monster[i].GetisTrace()){//偵測到被怪物追蹤
            redscreen.SetisFlash(true);//設定閃爍螢幕
            break;
        }
        else if(i==3&&!monster[i].GetisTrace()){//偵測到不被怪物追蹤
            redscreen.SetisFlash(false);//將閃爍解除
        }
    }
    if(treasure_remind==0){//已經取得所有寶物了,可以進到過關條件
        if(gamemap.IsEmpty(thief.Getx(),thief.Gety())==999){
            //寫入成績/時間
            int savetime = Tcc.GetTime();
            ofstream ouf;
            ouf.open("map/ST.bin",ios::out|ios::binary);
            GAME_ASSERT(ouf,"寫入成績/時間失敗.");
            ouf.write ((char*)&score,sizeof(int));
            ouf.write ((char*)&savetime,sizeof(int));
            ouf.write ((char*)&stage,sizeof(int));
            ouf.close();
        }
    }
}

```

```

        CAudio::Instance()->Play(AUDIO_MAP1, false);
        GotoGameState(GAME_STATE_PASS);
    }
    else if(gamemap.Isempty(thief.Getx()+3,thief.Gety())==999){

        //寫入成績/時間
        int savetime = Tcc.GetTime();
        ofstream ouf;
        ouf.open("map/ST.bin",ios::out|ios::binary);
        GAME_ASSERT(ouf,"寫入成績/時間失敗.");
        ouf.write ((char*)&score,sizeof(int));
        ouf.write ((char*)&savetime,sizeof(int));
        ouf.write ((char*)&stage,sizeof(int));
        ouf.close();
        CAudio::Instance()->Play(AUDIO_MAP1, false);
        GotoGameState(GAME_STATE_PASS);
    }
    else if(gamemap.Isempty(thief.Getx(),thief.Gety()+4)==999){

        //寫入成績/時間
        int savetime = Tcc.GetTime();
        ofstream ouf;
        ouf.open("map/ST.bin",ios::out|ios::binary);
        GAME_ASSERT(ouf,"寫入成績/時間失敗.");
        ouf.write ((char*)&score,sizeof(int));
        ouf.write ((char*)&savetime,sizeof(int));
        ouf.write ((char*)&stage,sizeof(int));
        ouf.close();
        CAudio::Instance()->Play(AUDIO_MAP1, false);
        GotoGameState(GAME_STATE_PASS);
    }
    else if(gamemap.Isempty(thief.Getx()+3,thief.Gety()+4)==999){

        //寫入成績/時間
        int savetime = Tcc.GetTime();
        ofstream ouf;
        ouf.open("map/ST.bin",ios::out|ios::binary);
        GAME_ASSERT(ouf,"寫入成績/時間失敗.");
        ouf.write ((char*)&score,sizeof(int));
        ouf.write ((char*)&savetime,sizeof(int));
        ouf.write ((char*)&stage,sizeof(int));
        ouf.close();
        CAudio::Instance()->Play(AUDIO_MAP1, false);
        GotoGameState(GAME_STATE_PASS);
    }
}
//如果計數器剛好被整除,代表執行到剛好秒的位子
if((counter%30)==0){
    Tcc.SetTime(Tcc.GetTime()-1); //剛好要到換時間的地方就呼叫改變時間
}else if(counter < 0){ //時間到,進入失敗畫面
    CAudio::Instance()->Play(AUDIO_MAP1, false);
    redscreen.SetisFlash(false);
    GotoGameState(GAME_STATE_OVER);
}else{
}
}

void CGameStateRun::OnInit(){ //開始載入資料,遊戲的初值及圖形設定
    ShowInitProgress(50); // 接個前一個狀態的進度,此處進度視為 33%
    cheat.LoadBitmap(IDB_CHEAT,RGB(255,255,255));
    help.LoadBitmap(); //載入說明圖檔
    thief.LoadBitmap(); //載入盜賊圖檔
    redscreen.LoadBitmap(); //載入紅螢幕圖檔
    for(int i=0;i<4;i++){ //載入四個怪物圖檔
        monster[i].LoadBitmap();
    }
    Tcc.LoadBitmap(false); //載入時間倒數圖檔
    gamemap.LoadMap(); //載入地圖圖檔
    for(int i=0;i<gamemap.Getcounttreasure();i++) //初始化各個物件

```



```

    treasure[i].LoadBitmap();
for(int i=0;i<gamemap.Getcountobstacle();i++)
    obstacles[i].Load();
int k=0;
for(int i=0;i<160;i++){
    for(int j=0;j<120;j++){
        if(gamemap.Isempty(i,j)==0){
            obstacles[k].setxy(i,j);
            k++;
        }
    }
}
k=0;
for(int i=0;i<160;i++){
    for(int j=0;j<120;j++){
        if(gamemap.Isempty(i,j)==2){
            treasure[k].SetXY(i,j);
            gamemap.toempty(i,j);
            k++;
        }
    }
}
k=0;
for(int i=0;i<160;i++){
    for(int j=0;j<120;j++){
        if(gamemap.DetialDoor(i,j)==1){
            door[k].setDoor(i,j,'h','n');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==2){
            door[k].setDoor(i,j,'v','n');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==3){
            door[k].setDoor(i,j,'Y','y');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==4){
            door[k].setDoor(i,j,'Z','y');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==5){
            door[k].setDoor(i,j,'R','r');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==6){
            door[k].setDoor(i,j,'S','r');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==7){
            door[k].setDoor(i,j,'G','g');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==8){
            door[k].setDoor(i,j,'H','g');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==9){
            door[k].setDoor(i,j,'P','p');
            k++;
        }
        else if(gamemap.DetialDoor(i,j)==10){
            door[k].setDoor(i,j,'Q','p');
            k++;
        }
    }
}
}
GAME_ASSERT(gamemap.Getcountdoors()==k,"取得門數量錯誤!");
k=0;

```

```

for(int i=0;i<160;i++){
    for(int j=0;j<120;j++){
        if(gamemap.Isempty(i,j)==5){
            key[k].setKey(i,j,gamemap.KeyColor(i,j));
            gamemap.toempty(i,j);
            k++;
        }
    }
}
for(int i=0;i<gamemap.Getcountdoors();i++)
    door[i].LoadDoor();
for(int i=0;i<gamemap.Getcountkeys();i++)
    key[i].LoadKey();

ShowInitProgress(80);    // 完成部分 Loading 動作，提高進度
// 繼續載入其他資料
// 此 OnInit 動作會接到 CGameStateOver::OnInit()，所以進度還沒到 100%
}

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_D      = 'D';
    const char KEY_SPACE  = ' ';
    const char KEY_W      = 'W';

    if((nChar == KEY_SPACE)&&(help.GetState()==false)){//一開始出現說明頁,玩家要按空白鍵繼續遊戲
        help.SetOk(true);//設定解除遊戲說明頁
        Tcc.SetTime(1imatetime);//重設時間為一開始的時間,也就是不列入看說明的時間
    }
    //盜賊上下左右移動鍵盤判定
    if (nChar == KEY_LEFT){
        thief.SetMovingLeft(true);
    }
    if (nChar == KEY_RIGHT){
        thief.SetMovingRight(true);
    }
    if (nChar == KEY_UP){
        thief.SetMovingUp(true);
    }
    if (nChar == KEY_DOWN){
        thief.SetMovingDown(true);
    }
    if (nChar == KEY_D){
        dkeybordisdown = true;
    }
    if(nChar == KEY_W){
        wkeybordisdown = true;
    }
}

void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
    const char KEY_LEFT  = 0x25; // keyboard 左箭頭
    const char KEY_UP    = 0x26; // keyboard 上箭頭
    const char KEY_RIGHT = 0x27; // keyboard 右箭頭
    const char KEY_DOWN  = 0x28; // keyboard 下箭頭
    const char KEY_D      = 'D';
    const char KEY_SPACE  = ' ';
    const char KEY_W      = 'W';

    if((nChar == KEY_SPACE)&&(help.GetState()==false)){//一開始出現說明頁,玩家要按空白鍵繼續遊戲
        help.SetOk(true);//設定解除遊戲說明頁
        Tcc.SetTime(1imatetime);//重設時間為一開始的時間,也就是不列入看說明的時間
    }
    //盜賊上下左右移動鍵盤判定
    if (nChar == KEY_LEFT){

```

```

        thief.SetMovingLeft(false);
        thief.reststep();
    }
    if (nChar == KEY_RIGHT){
        thief.SetMovingRight(false);
        thief.reststep();
    }
    if (nChar == KEY_UP){
        thief.SetMovingUp(false);
        thief.reststep();
    }
    if (nChar == KEY_DOWN){
        thief.SetMovingDown(false);
        thief.reststep();
    }
    //作弊模式
    if(nChar == KEY_D){
        if(dkeybordisdown){
            if(cheatmode){
                cheatmode = false;
            }
            else{
                cheatmode = true;
            }
            dkeybordisdown = false;
        }
    }
    if(nChar == KEY_W){
        if(wkeybordisdown){
            togetalltreasure=true;
        }
    }
}

void CGameStateRun::OnShow()
{
    // 注意：Show 裡面千萬不要移動任何物件的座標，移動座標的工作應由 Move 做才對，
    // 否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語
    // 說，Move 負責 MVC 中的 Model，Show 負責 View，而 View 不應更動 Model。
    gamemap.OnShow();//地圖背景(地板)顯示
    thief.OnShow(&gamemap); //盜賊顯示
    for(int i=0;i<gamemap.Getcountkeys();i++)
        key[i].Onshow(&gamemap);
    for(int i=0;i<4;i++)
        monster[i].OnShow(&gamemap);//怪獸顯示
    for(int i=0 ;i<gamemap.Getcoutobstacle();i++)//不斷重新產生障礙地圖
        obstacles[i].OnShow(&gamemap);
    for(int i=0;i<gamemap.Getcounttreasure();i++)
        treasure[i].OnShow(&gamemap);
    for(int i=0;i<gamemap.Getcountdoors();i++)
        door[i].OnShow(&gamemap);
    Tcc.OnShow();//專屬於每關卡計時器的物件
    redscreen.OnShow();//顯示紅螢幕物件
    help.OnShow();//顯示說明物件
    if(cheatmode==true){
        cheat.SetTopLeft(-1,421);
        cheat.ShowBitmap();
    }else{

    }
}

////////////////////////////////////
// 這個 class 為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
////////////////////////////////////

CGameStateRun2::CGameStateRun2(CGame *g)
: CGameState(g),fn("map/map2.txt"),gamemap(fn)//設定該關卡的地圖檔案
{

```

```

obstacles = new Obstacle[gamemap.Getcountobstacle()]; //產生障礙物件
monster = new Monster[4]; //產生怪物數量為 4
treasure = new Treasure[gamemap.Getcounttreasure()]; //產生寶物物件
door = new Door[gamemap.Getcountdoors()]; //產生門物件
if(gamemap.Getcountkeys()!=0){
    key = new Key[gamemap.Getcountkeys()];
}
else{
    key = NULL;
}
}

CGameStateRun2::~CGameStateRun2()
{
    delete [] obstacles;
    delete [] monster;
    delete [] treasure;
    delete [] door;
    delete [] key;
}

void CGameStateRun2::OnBeginState()
{
    score=0; //該關關卡成績計算初始為 0
    gamemap.Initialize(fn); //初始化遊戲地圖
    thief.Initialize(); //初始化盜賊
    redscreen.Initialize(); //初始化紅幕閃爍
    help.Initialize(); //初始化遊戲說明
    togetalltreasure = false;
    for(int i=0;i<4;i++){
        monster[i].Initialize();
    }
    for(int i=0;i<gamemap.Getcounttreasure();i++){
        treasure[i].Initialize();
    }
    for(int i=0;i<gamemap.Getcountdoors();i++){
        door[i].Initialize(&gamemap);
    }
    for(int i=0;i<gamemap.Getcountkeys();i++){
        key[i].Initialize();
    }
    //設定怪物行走路徑
    int
    path0[13][2]={101,40},{89,40},{89,44},{106,44},{106,53},{86,53},{66,53},{59,53},{59,62},{39,62},{19,62},{8,62},{8,81}};
    int
    path1[12][2]={69,111},{49,111},{45,111},{45,114},{25,114},{4,114},{4,94},{4,90},{23,90},{23,95},{39,95},{39,105}};
    int path2[8][2]={128,114},{154,114},{154,94},{134,94},{129,94},{129,80},{109,80},{94,80}};
    int path3[6][2]={139,61},{119,61},{99,61},{79,61},{75,61},{75,80}};
    //生設定怪物行走路徑
    monster[0].SetXY(path0[0][0],path0[0][1]);
    monster[1].SetXY(path1[0][0],path1[0][1]);
    monster[2].SetXY(path2[0][0],path2[0][1]);
    monster[3].SetXY(path3[0][0],path3[0][1]);
    //產生怪物行走路徑 Link
    vector<MovingNode*> scriptpath0;
    vector<MovingNode*> scriptpath1;
    vector<MovingNode*> scriptpath2;
    vector<MovingNode*> scriptpath3;
    for(int i=0;i<13;i++){
        scriptpath0.push_back(new MovingNode(path0[i][0],path0[i][1]));
    }
    for(int i=0;i<12;i++){
        scriptpath1.push_back(new MovingNode(path1[i][0],path1[i][1]));
    }
    for(int i=0;i<8;i++){
        scriptpath2.push_back(new MovingNode(path2[i][0],path2[i][1]));
    }
    for(int i=0;i<6;i++){
        scriptpath3.push_back(new MovingNode(path3[i][0],path3[i][1]));
    }
}

```

```

}
monster[0].Setcomenpath(scriptpath0);
monster[1].Setcomenpath(scriptpath1);
monster[2].Setcomenpath(scriptpath2);
monster[3].Setcomenpath(scriptpath3);
//關閉開始畫面音樂,播放第一關音樂
CAudio::Instance()->Stop(0);
CAudio::Instance()->Stop(6);
CAudio::Instance()->Play(AUDIO_MAP2, true);
gamemap.Setxsxy(thief.Getx()-16,thief.Gety()-12); //產生盜賊的位置
Tcc.SetTime(limatetime); //設定關卡遊戲時間
counter = 30 * Tcc.GetTime(); //設定遊戲關卡可以玩的時間,時間到進入遊戲失敗頁
treasure_remind = gamemap.Getcounttreasure();//取得寶物總數量
cheatmode = false; //作弊模式預設為不啟動
wkeybordisdown = false;
dkeybordisdown = false;
}

void CGameStateRun2::OnMove() // 移動遊戲元素
{
    //如果希望修改 cursor 的樣式,則將下面程式的 comment 取消即可
    //SetCursor(AfxGetApp()->LoadCursor(IDC_GAMECURSOR));
    if(!help.GetState()){
        return;
    }
    if(togetalltreasure&&treasure_remind!=0){
        treasure_remind = 0;
        for(int i=0;i<gamemap.Getcounttreasure();i++){
            treasure[i].Setlife(false);
        }
    }
    counter--; //時間計算依照每秒 30 運行次數倒數
    thief.OnMove(&gamemap,door,key);
    for(int i=0;i<4;i++){
        monster[i].OnMove(&gamemap,&thief,door,cheatmode);
    }
    for(int i=0;i<gamemap.Getcounttreasure();i++){
        if(treasure[i].Isalive()&&treasure[i].HitThief(&thief)){//寶物存活且被盜賊擊中
            CAudio::Instance()->Play(AUDIO_PICK, false);
            treasure_remind--;
            treasure[i].Setlife(false);//將寶物設定已取得
            score+=1000; //增加成績分數(寶物)
        }
    }
    for(int i=0;i<gamemap.Getcountkeys();i++){
        if(key[i].hitThief(thief.Getx(),thief.Gety())&&key[i].getisOnMap()){
            CAudio::Instance()->Play(AUDIO_KEY, false); //播放 KEY 音效
            key[i].setOnMap(false);
            score+=15; //增加成績分數(KEY)
        }
    }
    if(cheatmode == false){
        for(int i=0;i<4;i++){
            if(thief.Getx()-monster[i].Getx())>=-4&&thief.Getx()-monster[i].Getx()<=4&&thief.Gety()-monster[i].Gety()<=5&&thief.Gety()-monster[i].Gety()>=-5){
                CAudio::Instance()->Play(AUDIO_MAP2, false);
                redscreen.SetisFlash(false);
                GotoGameState(GAME_STATE_OVER);
            }
        }
    }
    for(int i=0;i<4;i++){
        if(monster[i].GetisTrace()){//偵測到被怪物追蹤
            redscreen.SetisFlash(true);//設定閃爍螢幕
            break;
        }
        else if(i==3&& !monster[i].GetisTrace()){//偵測到不被怪物追蹤
            redscreen.SetisFlash(false);//將閃爍解除
        }
    }
}

```

```

    }
}
if(treasure_remind==0){//已經取得所有寶物了,可以進到過關條件
    if(gamemap.Isempty(thief.Getx(),thief.Gety())==999){
        //寫入成績/時間
        int savetime = Tcc.GetTime();
        ofstream ouf;
        ouf.open("map/ST.bin",ios::out|ios::binary);
        GAME_ASSERT(ouf,"寫入成績/時間失敗.");
        ouf.write ((char*)&score,sizeof(int));
        ouf.write ((char*)&savetime,sizeof(int));
        ouf.write ((char*)&stage,sizeof(int));
        ouf.close();
        CAudio::Instance()->Play(AUDIO_MAP2, false);
        GotoGameState(GAME_STATE_PASS);
    }
    else if(gamemap.Isempty(thief.Getx()+3,thief.Gety())==999){

        //寫入成績/時間
        int savetime = Tcc.GetTime();
        ofstream ouf;
        ouf.open("map/ST.bin",ios::out|ios::binary);
        GAME_ASSERT(ouf,"寫入成績/時間失敗.");
        ouf.write ((char*)&score,sizeof(int));
        ouf.write ((char*)&savetime,sizeof(int));
        ouf.write ((char*)&stage,sizeof(int));
        ouf.close();
        CAudio::Instance()->Play(AUDIO_MAP2, false);
        GotoGameState(GAME_STATE_PASS);
    }
    else if(gamemap.Isempty(thief.Getx(),thief.Gety()+4)==999){

        //寫入成績/時間
        int savetime = Tcc.GetTime();
        ofstream ouf;
        ouf.open("map/ST.bin",ios::out|ios::binary);
        GAME_ASSERT(ouf,"寫入成績/時間失敗.");
        ouf.write ((char*)&score,sizeof(int));
        ouf.write ((char*)&savetime,sizeof(int));
        ouf.write ((char*)&stage,sizeof(int));
        ouf.close();
        CAudio::Instance()->Play(AUDIO_MAP2, false);
        GotoGameState(GAME_STATE_PASS);
    }
    else if(gamemap.Isempty(thief.Getx()+3,thief.Gety()+4)==999){

        //寫入成績/時間
        int savetime = Tcc.GetTime();
        ofstream ouf;
        ouf.open("map/ST.bin",ios::out|ios::binary);
        GAME_ASSERT(ouf,"寫入成績/時間失敗.");
        ouf.write ((char*)&score,sizeof(int));
        ouf.write ((char*)&savetime,sizeof(int));
        ouf.write ((char*)&stage,sizeof(int));
        ouf.close();
        CAudio::Instance()->Play(AUDIO_MAP2, false);
        GotoGameState(GAME_STATE_PASS);
    }
}
//如果計數器剛好被整除,代表執行到剛好秒的位子
if((counter%30)==0){
    Tcc.SetTime(Tcc.GetTime()-1);        //剛好要到換時間的地方就呼叫改變時間
}else if(counter < 0){//時間到,進入失敗畫面
    CAudio::Instance()->Play(AUDIO_MAP2, false);
    redscreen.SetisFlash(false);
    GotoGameState(GAME_STATE_OVER);
}else{
}
}
}

```

```

void CGameStateRun2::OnInit(){
    //開始載入資料,遊戲的初值及圖形設定
    //ShowInitProgress(50); // 接個前一個狀態的進度,此處進度視為 33%
    Tcc.LoadBitmap(true);
    cheat.LoadBitmap(IDB_CHEAT,RGB(255,255,255));
    help.LoadBitmap(); //載入說明圖檔
    thief.LoadBitmap();//載入盜賊圖檔
    redscreen.LoadBitmap();//載入紅螢幕圖檔
    for(int i=0;i<4;i++){//載入四個怪物圖檔
        monster[i].LoadBitmap();
    }
    gamemap.LoadMap();//載入地圖圖檔
    for(int i=0;i<gamemap.Getcounttreasure();i++)
        treasure[i].LoadBitmap();
    for(int i=0;i<gamemap.Getcoutobstacle();i++)
        obstacles[i].Load();
    int k=0;
    for(int i=0;i<160;i++){
        for(int j=0;j<120;j++){
            if(gamemap.Isempty(i,j)==0){
                obstacles[k].setxy(i,j);
                k++;
            }
        }
    }
    k=0;
    for(int i=0;i<160;i++){
        for(int j=0;j<120;j++){
            if(gamemap.Isempty(i,j)==2){
                treasure[k].SetXY(i,j);
                gamemap.toempty(i,j);
                k++;
            }
        }
    }
    k=0;
    for(int i=0;i<160;i++){
        for(int j=0;j<120;j++){
            if(gamemap.DetialDoor(i,j)==1){
                door[k].setDoor(i,j,'h','n');
                k++;
            }
            else if(gamemap.DetialDoor(i,j)==2){
                door[k].setDoor(i,j,'v','n');
                k++;
            }
            else if(gamemap.DetialDoor(i,j)==3){
                door[k].setDoor(i,j,'Y','y');
                k++;
            }
            else if(gamemap.DetialDoor(i,j)==4){
                door[k].setDoor(i,j,'Z','y');
                k++;
            }
            else if(gamemap.DetialDoor(i,j)==5){
                door[k].setDoor(i,j,'R','r');
                k++;
            }
            else if(gamemap.DetialDoor(i,j)==6){
                door[k].setDoor(i,j,'S','r');
                k++;
            }
            else if(gamemap.DetialDoor(i,j)==7){
                door[k].setDoor(i,j,'G','g');
                k++;
            }
            else if(gamemap.DetialDoor(i,j)==8){
                door[k].setDoor(i,j,'H','g');
                k++;
            }
        }
    }
}

```

```

    }
    else if(gamemap.DetialDoor(i,j)==9){
        door[k].setDoor(i,j,'P','p');
        k++;
    }
    else if(gamemap.DetialDoor(i,j)==10){
        door[k].setDoor(i,j,'Q','p');
        k++;
    }
}
}
GAME_ASSERT(gamemap.Getcountdoors()==k,"取得門數量錯誤!");
k=0;
for(int i=0;i<160;i++){
    for(int j=0;j<120;j++){
        if(gamemap.Isempty(i,j)==5){
            key[k].setKey(i,j,gamemap.KeyColor(i,j));
            gamemap.toempty(i,j);
            k++;
        }
    }
}
for(int i=0;i<gamemap.Getcountdoors();i++)
    door[i].LoadDoor();
for(int i=0;i<gamemap.Getcountkeys();i++)
    key[i].LoadKey();

// 繼續載入其他資料
// 此 OnInit 動作會接到 CGameStateOver::OnInit(), 所以進度還沒到 100%
}

void CGameStateRun2::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_D = 'D';
    const char KEY_SPACE = ' ';
    const char KEY_W = 'W';

    if((nChar == KEY_SPACE)&&(help.GetState()==false)){//一開始出現說明頁,玩家要按空白鍵繼續遊戲
        help.SetOk(true);//設定解除遊戲說明頁
        Tcc.SetTime(1imatetime);//重設時間為一開始的時間,也就是不列入看說明的時間
    }
    //盜賊上下左右移動鍵盤判定
    if (nChar == KEY_LEFT){
        thief.SetMovingLeft(true);
    }
    if (nChar == KEY_RIGHT){
        thief.SetMovingRight(true);
    }
    if (nChar == KEY_UP){
        thief.SetMovingUp(true);
    }
    if (nChar == KEY_DOWN){
        thief.SetMovingDown(true);
    }
    if (nChar == KEY_D){
        dkeybordisdown = true;
    }
    if(nChar == KEY_W){
        wkeybordisdown = true;
    }
}

void CGameStateRun2::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_D = 'D';
const char KEY_SPACE = ' ';
const char KEY_W = 'W';

if((nChar == KEY_SPACE)&&(help.GetState()==false)){//一開始出現說明頁,玩家要按空白鍵繼續遊戲
    help.SetOk(true);//設定解除遊戲說明頁
    Tcc.SetTime(limatetime);//重設時間為一開始的時間,也就是不列入看說明的時間
}
//盜賊上下左右移動鍵盤判定
if (nChar == KEY_LEFT){
    thief.SetMovingLeft(false);
    thief.reststep();
}
if (nChar == KEY_RIGHT){
    thief.SetMovingRight(false);
    thief.reststep();
}
if (nChar == KEY_UP){
    thief.SetMovingUp(false);
    thief.reststep();
}
if (nChar == KEY_DOWN){
    thief.SetMovingDown(false);
    thief.reststep();
}
//作弊模式
if(nChar == KEY_D){
    if(dkeybordisdown){
        if(cheatmode){
            cheatmode = false;
        }
        else{
            cheatmode = true;
        }
        dkeybordisdown = false;
    }
}
if(nChar == KEY_W){
    if(wkeybordisdown){
        togetalltreasure=true;
    }
}
}

void CGameStateRun2::OnShow()
{
    // 注意：Show 裡面千萬不要移動任何物件的座標，移動座標的工作應由 Move 做才對，
    // 否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語
    // 說，Move 負責 MVC 中的 Model，Show 負責 View，而 View 不應更動 Model。
    gamemap.OnShow();//地圖背景(地板)顯示
    thief.OnShow(&gamemap); //盜賊顯示
    for(int i=0;i<gamemap.Getcountkeys();i++)
        key[i].Onshow(&gamemap);
    for(int i=0;i<4;i++)
        monster[i].OnShow(&gamemap);//怪獸顯示
    for(int i=0 ;i<gamemap.Getcoutobstacle();i++)//不斷重新產生障礙地圖
        obstacles[i].OnShow(&gamemap);
    for(int i=0;i<gamemap.Getcounttreasure();i++)
        treasure[i].OnShow(&gamemap);
    for(int i=0;i<gamemap.Getcountdoors();i++)
        door[i].OnShow(&gamemap);

    Tcc.OnShow();//專屬於每關卡計時器的物件
    redscreen.OnShow();//顯示紅螢幕物件
    help.OnShow();//顯示說明物件
    if(cheatmode==true){
        cheat.SetTopLeft(-1,421);
    }
}

```

```
        cheat.ShowBitmap();  
    }  
    else{  
    }  
}  
}
```

全文完