國立臺北科技大學

2020 Spring 資工系物件導向程式實習

期末報告

Boom!!

第 14 組

108590001 李文至

108590018 顏維鴻

目錄

一、 簡介

1. 動機

一開始在討論題目的時候，因為我們小時候常常玩電腦遊戲，其中最有印象的就是爆爆王等等的遊戲，於是我們上網尋找有沒有跟爆爆王類似的小遊戲可以參考，最後就找到「Boom!!」這款遊戲來當做我們的主題。

2. 分工

李文至：

(一) 各選單程式撰寫、部分遊戲內程式撰寫(例如金幣、得分、生命值)、暫停時的資料保存與載回遊戲畫面、素材製作。

(二) 報告。

顏維鴻：

(一) 關卡內大部份遊戲設計:
腳色(移動、放置炸彈)、炸彈&障礙物互動、敵人(移動、攻擊、死亡)、敵人子彈(移動、物體碰觸判斷)、跨關卡資料保存&讀取、全部音效。

(二) 報告。

二、 遊戲介紹

1. 遊戲說明

(一) 此遊戲是要拿炸彈攻擊敵人。

(二) 操控方式：

- 上下左右：移動腳色。
- 空白鍵：放置炸彈。
- ESC / P：暫停遊戲。
- Ctrl + F：切換/取消為全螢幕。

(三) 遊戲規則：

- 成功爆破粉色石頭、撿拾金幣、殺死敵人等，即可增加分數。
- 若被敵人攻擊、碰到炸彈爆炸的火花等，都會被扣血。

(四) 分數計算：

- 爆破粉色石頭：獲得10分。
- 撿拾金幣：獲得150分。
- 殺死敵人：獲得100分。

(五) 生命計算：

生命共有3條，每條8顆心，當生命歸零的時候，玩家就會死亡。

- 被敵人攻擊：扣0.5顆心。
- 碰到爆炸的火花：扣0.5顆心。

(六) 通關方式：

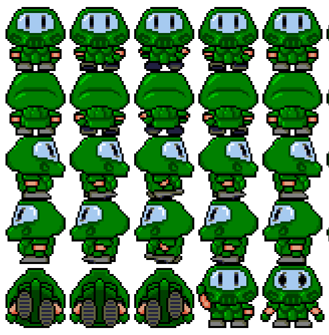
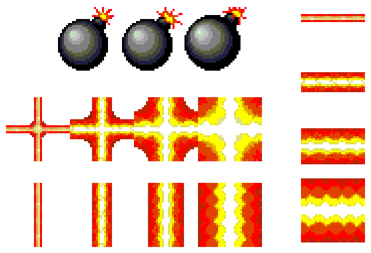
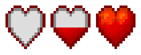
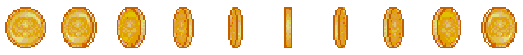
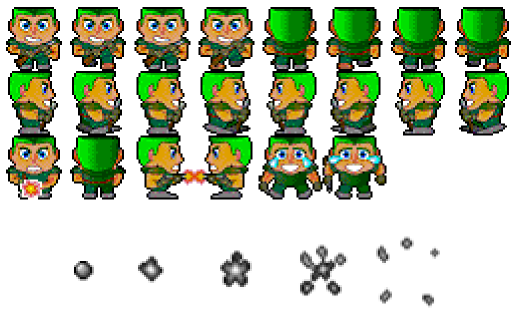
必須要將所有敵人殺死後才算通關。

(七) 密技：

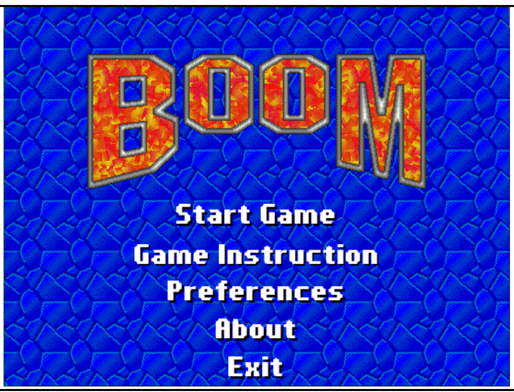
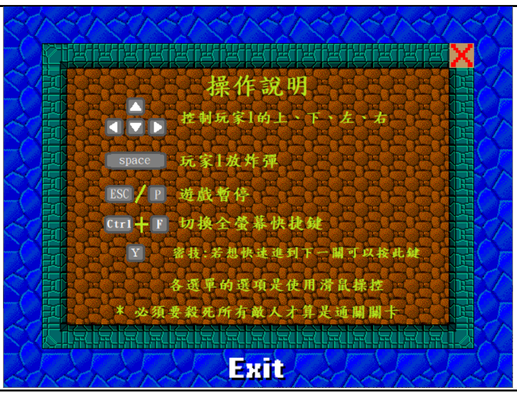
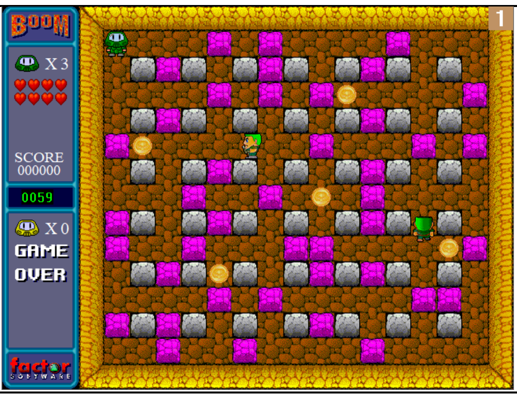
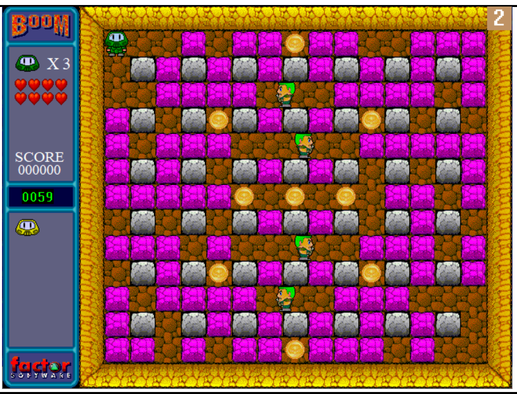
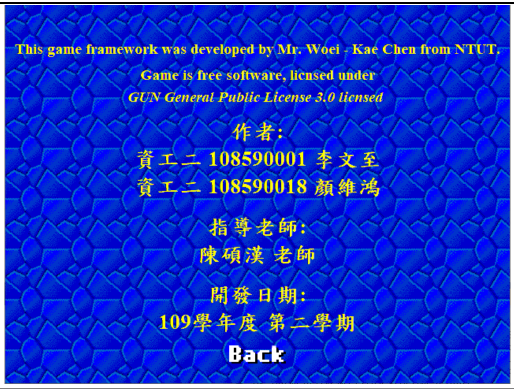
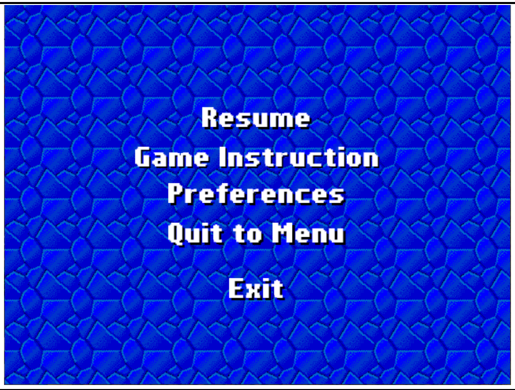
若不想要慢慢爆破石頭殺死所有敵人的話，可以按下「Y」鍵，此時即可快速進到下一關。

2. 遊戲圖形

遊戲圖形:

角色	炸彈 & 爆炸火花
	
玩家生命	金幣
	
敵人 & 子彈	選單背景、關卡地板 關卡：石頭障礙 & 被炸得石頭
	

遊戲畫面:

開始畫面	操作說明
 BOOM Start Game Game Instruction Preferences About Exit	操作說明 控制玩家1的上、下、左、右 space 玩家1放炸彈 ESC / P 遊戲暫停 Ctrl + F 切換全螢幕快捷鍵 Y 密技:若想快速進到下一關可以按此鍵 各選單的選項是使用滑鼠操控 * 必須要殺死所有敵人才算是通關關卡
第一關	第二關
 BOOM X 3 SCORE 000000 0059 X 0 GAME OVER 1 factor SOFTWARE	 BOOM X 3 SCORE 000000 0059 2 factor SOFTWARE
「關於」畫面	「暫停」畫面
This game framework was developed by Mr. Woei - Kae Chen from NTUT. Game is free software, licnsed under *GUN General Public License 3.0 licnsed* 作者: 資工二 108590001 李文至 資工二 108590018 顏維鴻 指導老師: 陳碩漢 老師 開發日期: 109學年度 第二學期 	 Resume Game Instruction Preferences Quit to Menu Exit

3. 遊戲音效

(一) 炸彈爆炸音效：Sounds/POWER.wav
來源: https://youtu.be/Hq2SlCja3zo?t=307

(二) 主畫面音效：Sounds/meum.mp3
來源: https://youtu.be/Zzo6L3wsf8c

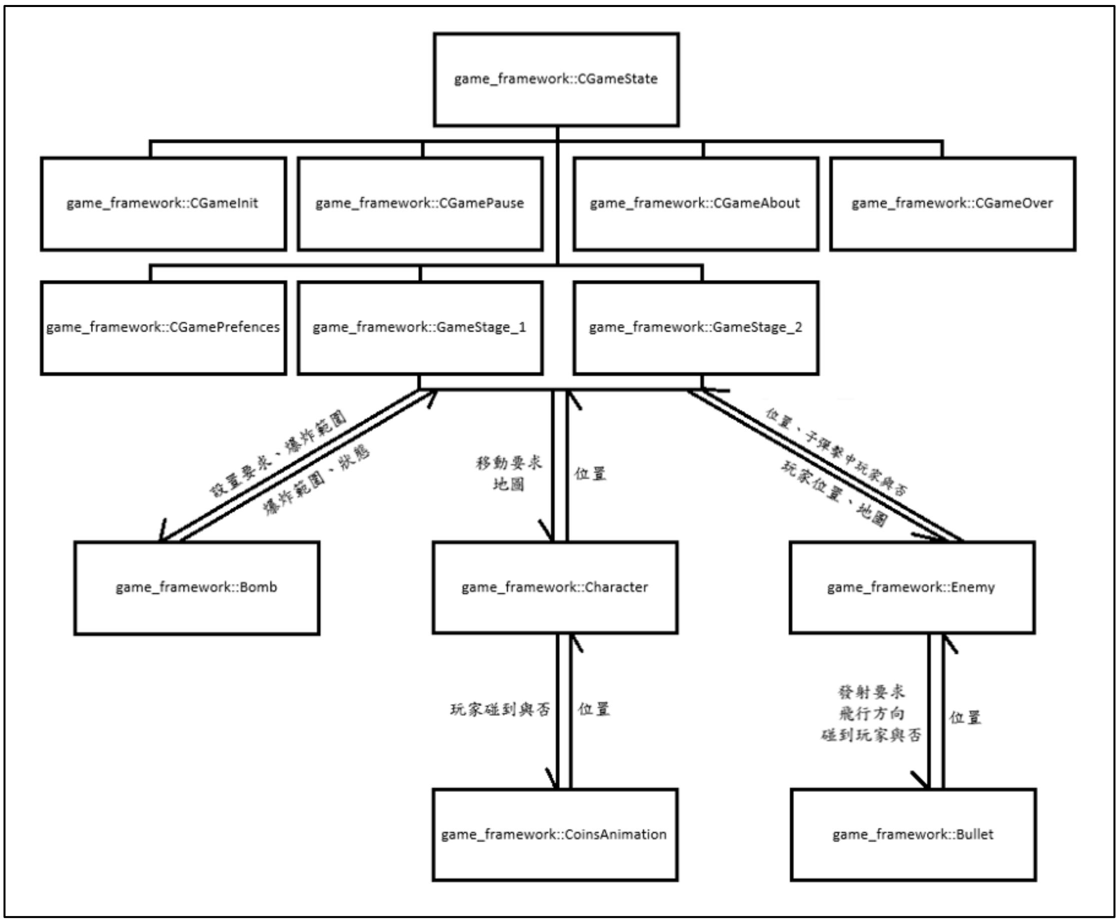
(三) 關卡一背景音樂: Sounds/stage1BGM.wav
來源: https://youtu.be/nKH_1GUmIhc?t=47

(四) 關卡二背景音樂：Sounds/stage2BGM.mp3
來源: https://youtu.be/Zz1bfhtKsHM

(五) 玩家受傷：Sounds/player1_hurt. mp3
來源:原作者

三、 程式設計

1. 程式架構


game_framework::CGameState
game_framework::CGameInit
game_framework::CGamePause
game_framework::CGameAbout
game_framework::CGameOver
game_framework::CGamePrefences
game_framework::GameStage_1
game_framework::GameStage_2
設置要求、爆炸範圍
爆炸範圍、狀態
移動要求
地圖
位置
位置、子彈擊中玩家與否
玩家位置、地圖
game_framework::Bomb
game_framework::Character
game_framework::Enemy
玩家碰到與否
位置
發射要求
飛行方向
碰到玩家與否
位置
game_framework::CoinsAnimation
game_framework::Bullet

2. 程式類別

類別名稱	.h 檔行數	.cpp 檔行數	說明
Bomb	44	165	炸彈Base Class
Bullet	25	90	子彈Base Class
Character	53	206	角色Base Class
CoinsAnimation	20	69	金幣動畫
Enemy	45	243	敵人Base Class
Healths	23	49	玩家生命
Obstacle	20	62	粉色牆壁Base Class
GameStage_2	61	583	第二關的關卡內容
GameStage_1	48	582	第一關的關卡內容
總行數	339	2049	

3. 程式技術

(一) 玩家:

玩家移動是由關卡傳送訊號，腳色會一直執行移動的函式，直到關卡傳送停止訊號，判斷移動所需的地圖會在一開始跟地圖變動的時候由關卡傳給玩家class。

炸彈實際上不是由玩家class來控制，而是關卡。

(二) 炸彈:

關卡在按下空白鍵之後會給可用的炸彈中的第一個傳送訊號，並改變地圖對應上的標記(由 0 改成 4)，位置的判斷依據是玩家的中心點，玩家的中心點在 13x15 的地圖上的哪個點就把炸彈設在哪裡，接到訊號的炸彈變為待爆狀態，class 內的計數器開始計時，等待兩秒之後由待爆變為爆炸狀態，在關卡接收到爆炸訊號之後，會向該炸彈索取各個方向的爆炸範圍，有了中心點跟爆炸範圍，接著在地圖陣列上改變數值，等爆炸動畫結束就將地圖恢復。

(三) 敵人:

敵人分兩部分:移動、攻擊，移動方面我想盡可能減少 AI 一直重複上下或左右移動的機率，我想出的解決辦法是讓可以走的長度越長的方向往那邊移動的機率越大，為了方便計算我讓 AI 只在走完完整一格才做一次判斷,算式簡化會是: 亂數 /(上+下+左+右)，用取出來的餘數。

攻擊的部分,我一開始是用現成的數值(做移動判斷時留下的上下左右可移動布數),原因是我認為敵人的視野範圍跟這個數值是一樣的，但可移動布數只在走完完整一格才更新，如果讓敵人一直執行攻擊判斷會發生沒看到玩家卻進行攻擊的情況，所以我的解決方法是讓 AI 在走完完整一格的時候才進行攻擊判斷。

(四) 子彈:

子彈在接收到 AI 發射訊號時，會一併收到起始座標跟飛行方向，子彈只負責朝指定的方向飛行，接著回傳數值給該 AI,子彈碰到物體的判斷跟動畫位置都由 AI 來處理。

四、 結語

1. 問題及解決方法

Q. 創建新的.h檔和.cpp檔沒辦法順利的正常使用。
A. 搞錯引用.h檔的順序，所以重新調整就可以正常使用了。

Q. 時間在遊戲結束後，重新開始時，不會歸零重新計時。
A. 重新調整暫存時間的方式，並增加載入時間後就清空暫存的內容。

Q. 在撿拾金幣的時候，金幣不會有動畫。
A. OnMove、OnShow、判斷金幣是否還存在的function等位置擺放有問題，重新調整之後就可以了。

Q. 碰到爆炸的火花會反覆讀取，導致血量不到一秒就被扣光。
A. 設定一個變數用來計時，如果達到一定的時間時，再進行扣血的動作。

Q. 殺死敵人以後分數會一直反覆增加，不會停止。
A. 跟上一個火花的部分類似，只要給予一個輔助判斷的變數之後就可以在殺死一個人只加一次分，不會反覆加了。

2. 時間表

週次	組員 – 李文至	組員 – 顏維鴻	說明
1	0	0	
2	0	0	研究Framework架構
3	6	3	研究Framework架構 尋找素材
4	8	5	研究Framework架構 製作起始畫面 起始畫面開發 關卡內背景圖
5	11	9	起始畫面(含按鈕變化) 暫停畫面素材 暫停畫面 關卡內背景圖完成 開始人物移動設計
6	12	7	暫停畫面(含按鈕變化) 倒數計時開發中 移除功能列 程式暫停 完成人物移動 開始炸彈設計
7	10	4	倒數計時完成 Prefences 畫面 研究關卡進入暫停畫面的暫存 炸彈架構完成，發現重複設置問題
8	10	2	關卡進入暫停畫面的時間暫存 About畫面 研究如何顯示全螢幕 炸彈大致完成 開始設計障礙物
9	10	2	拾取金幣(含動畫)開發50% 研究如何顯示全螢幕 障礙物大致完成 開始設計障礙物與炸彈的互動
10	12	0	顯示全螢幕開發35% 生命值開發45%

11	10	0	生命值開發65% 遊戲得分30% 炸彈剩下一些判斷錯誤，等之後再處理 開始設計敵人
12	11	0	角色死亡動畫開發50% 研究角色碰到火花反覆扣血問題 完成敵人移動動畫，發現隨機移動方式問題
13	12	0	角色死亡動畫開發90% 研究角色碰到火花反覆扣血問題 敵人隨機移動完成，開始敵人子彈設計
14	13	0	Prefences 畫面 暫停功能開發50% 研究攻擊敵人反覆加分問題
15	12	0	Prefences 畫面(含按鈕變化) 遊戲得分60% 敵人子彈完成
16	13	0	研究角色死亡不會動作問題 暫停功能開發100% 遊戲得分90% 生命、得分傳到下一關開發50% 複數敵人顯示測試、第二關製作
17	9	0	遊戲得分100% 重新製作About頁面 製作、開發操作說明視窗 關卡資料傳到下一關開發90% 第二關製作完成，缺乏換關條件
18	3	10	加入音效 完成換關 細項bug解決 封裝 報告
合計	162	42	

3. 貢獻比例

李文至：40%
顏維鴻：60%

4. 自我檢核表

週次	項目	完成否	無法完成原因
1	解決 Memory leak	■已完成 □未完成	
2	自定遊戲 Icon	■已完成 □未完成	
3	有 About 畫面	■已完成 □未完成	
4	初始畫面說明按鍵及滑之用法與密技	■已完成 □未完成	
5	上傳 setup/apk/source 檔	■已完成 □未完成	
6	setup 檔可正確執行	■已完成 □未完成	
7	報告字型、點數、對齊、行距、頁碼等格式正確	■已完成 □未完成	
8	全螢幕啟動–改列加分項目	□已完成 ■未完成	時間不足
9	報告封面、側邊格式正確	■已完成 □未完成	
10	報告附錄程式格式正確	■已完成 □未完成	

5. 收獲

李文至：

上學期的時候，就有上過物件導向了，這學期是透過寫遊戲來進行實際操作，也因為這堂課我才了解原來市面上的遊戲小到普通的遊戲，大到主流遊戲都有使用C語言進行開發。在開發的過程中，我更加瞭解了class的繼承方法，也瞭解到為什麼一個軟體當中要把東西個別分開來開發再組合，因為這樣才能夠有效的精簡化程式，讓程式碼看起來不會太雜亂。在製作素材時，讓我對於圖片的處理與製作更佳的上手，雖然有些地方不知道要怎麼去調整，但還好有其他方式可以替代，這學期的課程讓我學到了不少東西。

顏維鴻：

這學期我學到的是物件之間的互動，在我負責的部分有玩家跟敵人，兩個部分都有需要控制的物件，一開始設計炸彈時有一些數值我不知道怎麼去傳給其他物件，所以直接把炸彈的物件掛在mygame下，之後我在想敵人子彈時候就套用了上次失敗的經驗來做改善，結果上來說是成功的，只是後來我看到gamelib裡的CGame指標操作讓我有全新的想法，因為時間問題所以我沒改

6. 心得、感想

李文至：

一開始聽說要開發一個遊戲，在撰寫的時候只要把每個東西分別拆開再合併應該是不會太難，可以順利的寫出很多內容，結果沒想到實際撰寫的時候卻一點都不簡單，我跟組員挑了一個看起來好像很簡單的遊戲，可是寫起來卻比想像中的複雜，例如遊戲中的動畫要怎麼呈現才可以符合我們要的樣子、生命值得顯示要怎麼樣才能符合期待等等，本來以為有了想法，可是寫出來卻無法達到自己的想法，因為這樣，讓我更加了解到自己的程式能力有許多不足，謝謝組員在這學期的課程中，幫我解決了不少我不知道該怎麼解決的問題，我知道自己的程式有許多不足，所以也很感謝組員這次願意跟我一組，讓我學習到不少我本來還不太會的語法。也謝謝老師開設的這堂課，讓我深深的感受到原來開發是很深的一門學問，還有像遊戲這種大型專案只有一個人，是很難順利開發完成的。雖然最後做出來的東西不像學期初預期的那樣完美，但是至少已經努力過了，這堂課的收穫一定會在我未來寫程式時，成為我的助力。

顏維鴻：

我覺得在開發過程發現、思考以及解決問題才是最重要的，我會在很多時候去思索這些，不過大部分是在睡覺的時候想出來的(´・ω・`)

我覺得我們這組的遊戲內容除了附加功能以外還算簡單，因為我們的地圖系統比起其他組要簡單很多，只是我沒想到我的隊友在處理其他功能要花的時間實在太長了，不只沒辦法支援我，我還得去支援他，導致最終成果簡化了很多，不然我原本打算加一隻boss作為整個遊戲的最後一關

7. 對於本課程的建議

顏維鴻：

不要讓學弟或是明年的我選擇有其他附加功能的遊戲，因為這些功能所需圖像的處理很耗時間且跟物件導向沒有太大關係

附錄

mygame.h

```
#ifndef _MYGAME_
#define _MYGAME_
#include "Character.h"
#include "Bomb.h"
#include "Obstacle.h"
#include "Enemy.h"
#include "CoinsAnimation.h"
#include "Healths.h"
#include "GameStage_2.h"
namespace game_framework {
	/////////////////////////////////////////////////////////////////////////////
	// Constants
	/////////////////////////////////////////////////////////////////////////////
	static int form_state = 0;			// 1 為起始畫面  2 為暫停畫面   3 為 Preference
	static int form_ori = 0;			// 讀取上一個離開的畫面是哪一個
	static int show = 1;				// 是否顯示遊戲說明對話框(show:1 unshow:0)
	static int FS_state = 0;
	static int SF_state = 0;
	static int Vsync_state = 1;
	/////////////////////////////////////////////////////////////////////////////
	// 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 OnLButtonDown(UINT nFlags, CPoint point);		// 處理滑鼠的動作
	protected:
		void OnMove();
		void OnShow();						// 顯示這個狀態的遊戲畫面
	private:
		CMovingBitmap logo;					// Boom 的 logo
		CMovingBitmap scr_start;
		CMovingBitmap scr_gie;
		CMovingBitmap scr_preferences;
		CMovingBitmap scr_about;
		CMovingBitmap scr_exit;
		CMovingBitmap gameInstruction;
		CMovingBitmap close;
		POINT p;
	};
	/////////////////////////////////////////////////////////////////////////////
	// 這個 class 為遊戲的第一關執行物件
	/////////////////////////////////////////////////////////////////////////////
	class GameStage_1 : public CGameState {
	public:
		GameStage_1(CGame* g);
		~GameStage_1();
		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);			// 鍵盤動作
		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
		void setBomb(int);
		void mapChange(int,int,int);				// 地圖變動&通知 character
		void BombState();
		void setBombRange(int,int,int,int);			// 爆炸時設置範圍
		void GetCoins();					// 偵測碰撞金幣&動畫
		void HealthState();
	private:
        CMovingBitmap level;
        int bg[13][15];                          // 0 地板 1 石塊 2 粉色石 4 未爆彈 5 爆炸中
        int coins_pos[5][2];                     // 硬幣位置
        CMovingBitmap block_0;
        CMovingBitmap block_1;
        Obstacle*       block_2;
        int             block_2_pos[42][2];
        CMovingBitmap panel;
        CMovingBitmap border;
        Character       character_1;             // Range undone
        CMovingBitmap character_2;               // 類別之後改
        int Enemy1_num;                          // 敵人 1 的數量
        int Enemy2_num;                          // 敵人 2 的數量
        Enemy             *AI;
        int coins_num;                           //金幣總數
        CoinsAnimation* coin_Ani;
        Bomb*             Bomb_ch1;
        CMovingBitmap playerhead_1;
        CMovingBitmap playerhead_2;
        int life;
        int heart_num[8];
        int blood_ori, blood_vol;
        Healths* heart;
        bool taking_Damage;
        int k = 0;
        CInteger count_down;
        int timer;
        int score;
};
/////////////////////////////////////////////////////////////////////////////
//遊戲過程中的暫停畫面
/////////////////////////////////////////////////////////////////////////////
class CGamestatePause : public CGameState {
public:
        CGamestatePause(CGame* g);
        void OnInit();
        void OnBeginState();                                   // 設定每次重玩所需的變數
        void OnLButtonDown(UINT nFlags, CPoint point);         // 處理滑鼠的動作
        void OnMouseMove(UINT nFlags, CPoint point);
protected:
        void OnShow();
private:
        CMovingBitmap bg;
        CMovingBitmap scr_resume;
        CMovingBitmap scr_gie;
        CMovingBitmap scr_preferences;
        CMovingBitmap scr_quitToMenu;
        CMovingBitmap scr_exit;
        CMovingBitmap gameInstruction;
        CMovingBitmap close;
        POINT p;
};
/////////////////////////////////////////////////////////////////////////////
// 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 OnMove();                                              // 移動遊戲元素
        void OnShow();                                              // 顯示這個狀態的遊戲畫面
private:
        CMovingBitmap over;
        int counter;                                                // 倒數之計數器
        int score;
};
```

```
        /////////////////////////////////////////////////////////////////////////////
        //Prefences 畫面
        /////////////////////////////////////////////////////////////////////////////
        class GamePrefences : public CGameState {
        public:
                GamePrefences(CGame* g);
                void OnBeginState();
                void OnInit();
                void OnLButtonDown(UINT nFlags, CPoint point);  // 處理滑鼠的動作
                void OnMouseMove(UINT nFlags, CPoint point);
        protected:
                void OnShow();
        private:
                CMovingBitmap bg;
                CMovingBitmap scr;
                CMovingBitmap scr_ok;
                CMovingBitmap scr_cancel;
                CMovingBitmap scr_FX_down;          // FX 音量減小
                CMovingBitmap scr_FX_up;            // FX 音量放大
                CMovingBitmap scr_FS_yes;           // Fullscreen YES
                CMovingBitmap scr_FS_no;            // Fullscreen NO
                CMovingBitmap scr_FR;               // Fllscr. Res.
                CMovingBitmap scr_SF_yes;           // Show FPS YES
                CMovingBitmap scr_SF_no;            // Show FPS NO
                CMovingBitmap scr_Vsync_yes;        // Vsync YES
                CMovingBitmap scr_Vsync_no;         // Vsync NO
                int FS_ori_state;                   // Fullscreen 原狀態
                int SF_ori_state;                   // Show FPS 原狀態
                int Vsync_ori_state;                // Vsync 原狀態
                POINT p;
        };
        /////////////////////////////////////////////////////////////////////////////
        //About 畫面
        /////////////////////////////////////////////////////////////////////////////
        class GameAbout : public CGameState {
        public:
                GameAbout(CGame* g);
                void OnBeginState();
                void OnInit();
                void OnLButtonDown(UINT nFlags, CPoint point);  // 處理滑鼠的動作
                void OnMouseMove(UINT nFlags, CPoint point);
        protected:
                void OnShow();
        private:
                CMovingBitmap bg;
                CMovingBitmap aboutForm;
                CMovingBitmap scr_back;
                POINT p;
        };
}
#endif
```

mygame.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"
#include <iostream>
namespace game_framework {
/////////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的遊戲開頭畫面物件
/////////////////////////////////////////////////////////////////////////////
CGameStateInit::CGameStateInit(CGame *g)
: CGameState(g)
{
}
void CGameStateInit::OnInit()
{
	//
	// 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
	//
	logo.LoadBitmap(IDB_LOGO, RGB(255,255, 255));
	scr_start.LoadBitmap(IDB_SCREEN_START, RGB(0, 0, 255));
	scr_gie.LoadBitmap(IDB_SCREEN_GI, RGB(0,0, 255));
	scr_preferences.LoadBitmap(IDB_SCREEN_PREFERENCES, RGB(0,0, 255));
	scr_about.LoadBitmap(IDB_SCREEN_ABOUT, RGB(0, 0, 255));
	scr_exit.LoadBitmapA(IDB_SCREEN_EXIT, RGB(0,0, 255));
	//
	// 此 OnInit 動作會接到 CGameStaterRun::OnInit()，所以進度還沒到 100%
	//
}
void CGameStateInit::OnBeginState()
{
	form_state = 1;
	if (!CAudio::Instance()->loadCheck(AUDIO_MEUM))
		CAudio::Instance()->Load(AUDIO_MEUM, "sounds\\meum.mp3");
	CAudio::Instance()->Play(AUDIO_MEUM, true);
}
void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point)
{
	if (show == 1) {
		if (p.x > close.Left() && p.x < close.Left() + close.Width() &&
			p.y > close.Top() && p.y < close.Top() + close.Height()) {
			show = 0;
		}
	} else {
		if (p.x > scr_start.Left() && p.x < scr_start.Left() + scr_start.Width() &&
			p.y > scr_start.Top() && p.y < scr_start.Top() + scr_start.Height()) {
			CAudio::Instance()->Stop(AUDIO_MEUM);
			GotoGameState(GAME_STAGE_1);                    // 切換至第一關
		}
		else if (p.x > scr_gie.Left() && p.x < scr_gie.Left() + scr_gie.Width() &&
			p.y > scr_gie.Top() && p.y < scr_gie.Top() + scr_gie.Height()) {
			show = 1;                                       // 顯示遊戲說明
		}
		else if (p.x > scr_preferences.Left() && p.x < scr_preferences.Left() + scr_preferences.Width() &&
			p.y > scr_preferences.Top() && p.y < scr_preferences.Top() + scr_preferences.Height()) {
			GotoGameState(GAME_PREFENCES);                  // 切換至 Prefences 畫面
		}
		else if (p.x > scr_about.Left() && p.x < scr_about.Left() + scr_about.Width() &&
			p.y > scr_about.Top() && p.y < scr_about.Top() + scr_about.Height()) {
			GotoGameState(GAME_ABOUT);                      // 切換至 About 畫面
		}
		else if (p.x > scr_exit.Left() && p.x < scr_exit.Left() + scr_exit.Width() &&
			p.y > scr_exit.Top() && p.y < scr_exit.Top() + scr_exit.Height()) {
			PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);		// 關閉遊戲
		}
	}
}
void CGameStateInit::OnMove()
{
	GetCursorPos(&p);
	ScreenToClient(AfxGetMainWnd()->m_hWnd, &p);		// 把螢幕座標轉換為視窗座標，並讀取出來
	//TRACE("mouse:: x: %d, y: %d\n", p.x, p.y);
}
void CGameStateInit::OnShow()
{
	// 放背景
	CMovingBitmap bg;
	bg.LoadBitmap(IDB_SCREENBG1, RGB(0,0,0));
	for (int x = 0; x < SIZE_X; x += bg.Width()) {
		for (int y = 0; y < SIZE_Y; y += bg.Height()) {
			bg.SetTopLeft(x,y);
			bg.ShowBitmap();
		}
	}
	//
	// 貼上 logo
	//
	logo.SetTopLeft((SIZE_X - logo.Width())/2, SIZE_Y/8);
	logo.ShowBitmap();
	// Screen_Start
	if (p.x > scr_start.Left() && p.x < scr_start.Left() + scr_start.Width() &&
		p.y > scr_start.Top() && p.y < scr_start.Top() + scr_start.Height()) {
		CMovingBitmap scr_start_red;
		scr_start_red.LoadBitmap(IDB_SCREEN_START_RED, RGB(0, 0, 255));
		scr_start_red.SetTopLeft((SIZE_X - scr_start_red.Width()) / 2, SIZE_Y / 2);
		scr_start_red.ShowBitmap();
	} else {
		scr_start.SetTopLeft((SIZE_X - scr_start.Width()) / 2, SIZE_Y / 2);
		scr_start.ShowBitmap();
	}
	// Screen_Game_Instructions
	if (p.x > scr_gie.Left() && p.x < scr_gie.Left() + scr_gie.Width() &&
		p.y > scr_gie.Top() && p.y < scr_gie.Top() + scr_gie.Height()) {
		CMovingBitmap scr_gie_red;
		scr_gie_red.LoadBitmap(IDB_SCREEN_GI_RED, RGB(0, 0, 255));
		scr_gie_red.SetTopLeft((SIZE_X - scr_gie_red.Width()) / 2, SIZE_Y / 2 + scr_start.Height());
		scr_gie_red.ShowBitmap();
	}
	else {
		scr_gie.SetTopLeft((SIZE_X - scr_gie.Width()) / 2, SIZE_Y / 2 + scr_gie.Height());
		scr_gie.ShowBitmap();
	}
	// Screen_Preferences
	if (p.x > scr_preferences.Left() && p.x < scr_preferences.Left() + scr_preferences.Width() &&
		p.y > scr_preferences.Top() && p.y < scr_preferences.Top() + scr_preferences.Height()) {
		CMovingBitmap scr_preferences_red;
		scr_preferences_red.LoadBitmap(IDB_SCREEN_PREFERENCES_RED, RGB(0, 0, 255));
		scr_preferences_red.SetTopLeft((SIZE_X - scr_preferences_red.Width()) / 2, SIZE_Y / 2 + scr_gie.Height() +
scr_preferences.Height());
		scr_preferences_red.ShowBitmap();
	}
	else {
		scr_preferences.SetTopLeft((SIZE_X - scr_preferences.Width()) / 2, SIZE_Y / 2 + scr_gie.Height() +
scr_preferences.Height());
		scr_preferences.ShowBitmap();
	}
	// Screen_about
	if (p.x > scr_about.Left() && p.x < scr_about.Left() + scr_about.Width() &&
		p.y > scr_about.Top() && p.y < scr_about.Top() + scr_about.Height()) {
		CMovingBitmap scr_about_red;
```

```
			scr_about_red.LoadBitmap(IDB_SCREEN_ABOUT_RED, RGB(0, 0, 255));
			scr_about_red.SetTopLeft((SIZE_X - scr_about_red.Width()) / 2, SIZE_Y / 2 + scr_gie.Height() +
scr_preferences.Height() + scr_about.Height());
			scr_about_red.ShowBitmap();
		}
		else {
			scr_about.SetTopLeft((SIZE_X - scr_about.Width()) / 2, SIZE_Y / 2 + scr_gie.Height() +
scr_preferences.Height() + scr_about.Height());
			scr_about.ShowBitmap();
		}
		// Screen_Exit
		if (p.x > scr_exit.Left() && p.x < scr_exit.Left() + scr_exit.Width() &&
			p.y > scr_exit.Top() && p.y < scr_exit.Top() + scr_exit.Height()) {
			CMovingBitmap scr_exit_red;
			scr_exit_red.LoadBitmap(IDB_SCREEN_EXIT_RED, RGB(0, 0, 255));
			scr_exit_red.SetTopLeft((SIZE_X - scr_exit_red.Width()) / 2, SIZE_Y / 2 + scr_gie.Height() +
scr_preferences.Height() + scr_about.Height() + scr_exit.Height());
			scr_exit_red.ShowBitmap();
		}
		else {
			scr_exit.SetTopLeft((SIZE_X - scr_exit.Width()) / 2, SIZE_Y / 2 + scr_gie.Height() + scr_preferences.Height()
+ scr_about.Height() + scr_exit.Height());
			scr_exit.ShowBitmap();
		}
		// 顯示遊戲說明
		if (show == 1) {
			gameInstruction.LoadBitmap(IDB_SCR_GAMEINFO);
			close.LoadBitmap(IDB_CLOSE);
			gameInstruction.SetTopLeft((SIZE_X - gameInstruction.Width()) / 2, (SIZE_Y - gameInstruction.Height()) / 2);
			close.SetTopLeft((SIZE_X - gameInstruction.Width()) / 2 + (gameInstruction.Width() - close.Width()),
(SIZE_Y - gameInstruction.Height()) / 2);
			gameInstruction.ShowBitmap();
			close.ShowBitmap();
		}
}
/////////////////////////////////////////////////////////////////////////////
//game pause
/////////////////////////////////////////////////////////////////////////////
CGamestatePause::CGamestatePause(CGame* g)
: CGameState(g)
{
}
void CGamestatePause::OnInit()
{
		scr_resume.LoadBitmap(IDB_SCREEN_RESUME, RGB(0, 0, 255));
		scr_gie.LoadBitmapA(IDB_SCREEN_GI, RGB(0, 0, 255));
		scr_preferences.LoadBitmapA(IDB_SCREEN_PREFERENCES, RGB(0, 0, 255));
		scr_quitToMenu.LoadBitmap(IDB_SCREEN_QUIT_TO_MENU, RGB(0, 0, 255));
		scr_exit.LoadBitmapA(IDB_SCREEN_EXIT, RGB(0, 0, 255));
}
void CGamestatePause::OnBeginState()
{
		form_state = 2;
}
void CGamestatePause::OnLButtonDown(UINT nFlags, CPoint point)
{
		if (show == 1) {
			if (p.x > close.Left() && p.x < close.Left() + close.Width() &&
				p.y > close.Top() && p.y < close.Top() + close.Height()) {
				show = 0;
			}
		}
		else {
			if (p.x > scr_resume.Left() && p.x < scr_resume.Left() + scr_resume.Width() &&
				p.y > scr_resume.Top() && p.y < scr_resume.Top() + scr_resume.Height()) {
				//Resume
				game_framework::CGame::Instance()->ContiuneState(game_framework::CGame::Instance()-
>getState());
				game_framework::CGame::Instance()->SaveState(nullptr);		// clean savestate
```

```
                }
                else if (p.x > scr_gie.Left() && p.x < scr_gie.Left() + scr_gie.Width() &&
                        p.y > scr_gie.Top() && p.y < scr_gie.Top() + scr_gie.Height()) {
                        show = 1;
                }
                else if (p.x > scr_preferences.Left() && p.x < scr_preferences.Left() + scr_preferences.Width() &&
                        p.y > scr_preferences.Top() && p.y < scr_preferences.Top() + scr_preferences.Height()) {
                        GotoGameState(GAME_PREFENCES);
                }
                else if (p.x > scr_quitToMenu.Left() && p.x < scr_quitToMenu.Left() + scr_quitToMenu.Width() &&
                        p.y > scr_quitToMenu.Top() && p.y < scr_quitToMenu.Top() + scr_quitToMenu.Height()) {
                        GotoGameState(GAME_STATE_INIT);
                }
                else if (p.x > scr_exit.Left() && p.x < scr_exit.Left() + scr_exit.Width() &&
                        p.y > scr_exit.Top() && p.y < scr_exit.Top() + scr_exit.Height()) {
                        PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);          // 關閉遊戲
                }
        }
}
void CGamestatePause::OnMouseMove(UINT nFlags, CPoint point)
{
        GetCursorPos(&p);
        ScreenToClient(AfxGetMainWnd()->m_hWnd, &p);         // 把螢幕座標轉換為視窗座標，並讀取出來
        //TRACE("mouse:: x: %d, y: %d\n", p.x, p.y);
}
void CGamestatePause::OnShow()
{
        // 放背景
        bg.LoadBitmap(IDB_SCREENBG1, RGB(0, 0, 0));
        for (int x = 0; x < SIZE_X; x += bg.Width()) {
                for (int y = 0; y < SIZE_Y; y += bg.Height()) {
                        bg.SetTopLeft(x, y);
                        bg.ShowBitmap();
                }
        }
        // Resume
        if (p.x > scr_resume.Left() && p.x < scr_resume.Left() + scr_resume.Width() &&
                p.y > scr_resume.Top() && p.y < scr_resume.Top() + scr_resume.Height()) {
                CMovingBitmap scr_resume_red;
                scr_resume_red.LoadBitmap(IDB_SCREEN_RESUME_RED, RGB(0, 0, 255));
                scr_resume_red.SetTopLeft((SIZE_X - scr_resume_red.Width()) / 2, SIZE_Y * 25 / 100);
                scr_resume_red.ShowBitmap();
        } else {
                scr_resume.SetTopLeft((SIZE_X - scr_resume.Width()) / 2, SIZE_Y * 25 / 100);
                scr_resume.ShowBitmap();
        }
        // Game Instruction
        if (p.x > scr_gie.Left() && p.x < scr_gie.Left() + scr_gie.Width() &&
                p.y > scr_gie.Top() && p.y < scr_gie.Top() + scr_gie.Height()) {
                CMovingBitmap scr_gie_red;
                scr_gie_red.LoadBitmap(IDB_SCREEN_GI_RED, RGB(0, 0, 255));
                scr_gie_red.SetTopLeft((SIZE_X - scr_gie_red.Width()) / 2, SIZE_Y * 35 / 100);
                scr_gie_red.ShowBitmap();
        } else {
                scr_gie.SetTopLeft((SIZE_X - scr_gie.Width()) / 2, SIZE_Y * 35 / 100);
                scr_gie.ShowBitmap();
        }
        // Preferences
        if (p.x > scr_preferences.Left() && p.x < scr_preferences.Left() + scr_preferences.Width() &&
                p.y > scr_preferences.Top() && p.y < scr_preferences.Top() + scr_preferences.Height()) {
                CMovingBitmap scr_preferences_red;
                scr_preferences_red.LoadBitmap(IDB_SCREEN_PREFERENCES_RED, RGB(0, 0, 255));
                scr_preferences_red.SetTopLeft((SIZE_X - scr_preferences_red.Width()) / 2, SIZE_Y * 45 / 100);
                scr_preferences_red.ShowBitmap();
        } else {
                scr_preferences.SetTopLeft((SIZE_X - scr_preferences.Width()) / 2, SIZE_Y * 45 / 100);
                scr_preferences.ShowBitmap();
        }
        // Quit to Menu
```

```
	if (p.x > scr_quitToMenu.Left() && p.x < scr_quitToMenu.Left() + scr_quitToMenu.Width() &&
		p.y > scr_quitToMenu.Top() && p.y < scr_quitToMenu.Top() + scr_quitToMenu.Height()) {
		CMovingBitmap scr_quitToMenu_red;
		scr_quitToMenu_red.LoadBitmap(IDB_SCREEN_QUIT_TO_MENU_RED, RGB(0, 0, 255));
		scr_quitToMenu_red.SetTopLeft((SIZE_X - scr_quitToMenu_red.Width()) / 2, SIZE_Y * 55 / 100);
		scr_quitToMenu_red.ShowBitmap();
	} else {
		scr_quitToMenu.SetTopLeft((SIZE_X - scr_quitToMenu.Width()) / 2, SIZE_Y * 55 / 100);
		scr_quitToMenu.ShowBitmap();
	}
	// Screen_Exit
	if (p.x > scr_exit.Left() && p.x < scr_exit.Left() + scr_exit.Width() &&
		p.y > scr_exit.Top() && p.y < scr_exit.Top() + scr_exit.Height()) {
		CMovingBitmap scr_exit_red;
		scr_exit_red.LoadBitmap(IDB_SCREEN_EXIT_RED, RGB(0, 0, 255));
		scr_exit_red.SetTopLeft((SIZE_X - scr_exit_red.Width()) / 2, SIZE_Y * 70 / 100);
		scr_exit_red.ShowBitmap();
	} else {
		scr_exit.SetTopLeft((SIZE_X - scr_exit.Width()) / 2, SIZE_Y * 70 / 100);
		scr_exit.ShowBitmap();
	}
	// 顯示遊戲說明
	if (show == 1) {
		gameInstruction.LoadBitmap(IDB_SCR_GAMEINFO);
		close.LoadBitmap(IDB_CLOSE);
		gameInstruction.SetTopLeft((SIZE_X - gameInstruction.Width()) / 2, (SIZE_Y - gameInstruction.Height()) / 2);
		close.SetTopLeft((SIZE_X - gameInstruction.Width()) / 2 + (gameInstruction.Width() - close.Width()),
(SIZE_Y - gameInstruction.Height()) / 2);
		gameInstruction.ShowBitmap();
		close.ShowBitmap();
	}
}
/////////////////////////////////////////////////////////////////////////////
// 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)
		GotoGameState(GAME_STATE_INIT);
}
void CGameStateOver::OnBeginState()
{
	counter = 30 * 5; // 5 seconds
	over.LoadBitmap(IDB_SCREEN_GAMEOVER, RGB(255, 255, 255));
	int data[1] = {0};
	game->loadData(data, 1);
	score = data[0];
}
void CGameStateOver::OnInit()
{
	//
	// 當圖很多時，OnInit 載入所有的圖要花很多時間。為避免玩遊戲的人
	// 等的不耐煩，遊戲會出現「Loading ...」，顯示 Loading 的進度。
	//
	ShowInitProgress(66);	// 接個前一個狀態的進度，此處進度視為 66%
	//
	// 開始載入資料
	//
	// 最終進度為 100%
	//
	ShowInitProgress(100);
}
void CGameStateOver::OnShow()
{
        over.SetTopLeft((SIZE_X - over.Width()) / 2, SIZE_Y * 25 / 100);
        over.ShowBitmap();
        CDC* pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
        CFont f, * fp;
        f.CreatePointFont(200, "Times New Roman");          // 產生 font f; 160 表示 16point 的字
        fp = pDC->SelectObject(&f);                         // 選用 font f
        pDC->SetBkColor(RGB(0, 0, 0));
        pDC->SetTextColor(RGB(255, 255, 0));
        char str[80];                                               // Demo 數字對字串的轉換
        sprintf(str, "You got %d points in this game!", score);
        pDC->TextOut(140, SIZE_Y * 40 / 100, str);
        pDC->SelectObject(fp);                                      // 放掉 font f (千萬不要漏了放掉)
        CDDraw::ReleaseBackCDC();                                   // 放掉 Back Plain 的 CDC
        CDC* pDC1 = CDDraw::GetBackCDC();                           // 取得 Back Plain 的 CDC
        CFont f1, * fp1;
        f1.CreatePointFont(140, "Times New Roman");                 // 產生 font f; 160 表示 16point 的字
        fp1 = pDC1->SelectObject(&f1);                              // 選用 font f
        pDC1->SetBkColor(RGB(0, 0, 0));
        pDC1->SetTextColor(RGB(255, 255, 255));
        char str1[80];                                              // Demo 數字對字串的轉換
        sprintf(str1, "Wait %d second back to Menu !", counter / 30);
        pDC1->TextOut(220, SIZE_Y * 50 / 100, str1);
        pDC1->SelectObject(fp);                                     // 放掉 font f (千萬不要漏了放掉)
        CDDraw::ReleaseBackCDC();                                   // 放掉 Back Plain 的 CDC
}
/////////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的第一關
/////////////////////////////////////////////////////////////////////////////
GameStage_1::GameStage_1(CGame* g) : CGameState(g)
{
        Bomb_ch1 = new Bomb [7];
        block_2 = new Obstacle[42];
        coin_Ani = new CoinsAnimation[5];
        heart = new Healths[8];
        AI = new Enemy[2];
}
GameStage_1::~GameStage_1() {
        delete [] Bomb_ch1;
        delete[] block_2;
        delete[] coin_Ani;
        delete[] heart;
        delete[] AI;
}
void GameStage_1::OnBeginState() {
        for (int i = 0; i < 7; i++) {
                Bomb_ch1[i].Initialize();
        }
        int bg_reset[13][15] = {            //0 地板 1 石塊 2 粉色石
                        {0,0,0,0,2,0,2,0,0,0,0,2,0,0,0},
                        {0,1,2,1,0,1,2,1,0,1,2,1,0,1,0},
                        {0,0,0,0,2,0,2,0,2,0,0,0,0,0,2},
                        {0,1,2,1,0,1,0,1,0,1,2,1,0,1,0},
                        {2,0,0,0,0,0,0,0,2,0,0,0,2,0,2},
                        {0,1,0,1,2,1,0,1,0,1,2,1,0,1,0},
                        {0,0,0,2,0,0,2,0,0,0,2,0,0,0,0},
                        {2,1,0,1,2,1,0,1,0,1,2,1,0,1,0},
                        {2,0,0,2,0,0,0,2,2,0,0,0,0,0,2},
                        {0,1,2,1,0,1,0,1,2,1,2,1,0,1,0},
                        {0,0,0,0,2,0,2,0,0,0,0,0,2,2,0},
                        {2,1,2,1,0,1,0,1,2,1,0,1,0,1,0},
                        {0,0,2,0,0,2,0,0,0,0,2,0,0,0,0}
        };
        int obstacle_reset[42][2] = {
                {0, 4},{0, 6},{0, 11},{1, 2},{1, 6},{1, 10},{2, 4},{2, 6},{2, 8},{2, 14},{3, 2},{3,10},
                {4,0},{4,8},{4,12},{4,14},{5,4},{5,10},{6,3},{6,6},{6,10},{7,0},{7,4},{7,10},{8,0},{8,3},
                {8,7},{8,8},{8,14},{9,2},{9,8},{9,10},{10,4},{10,6},{10,12},{10,13},{11,0},{11,2},{11,8},
                {12,2},{12,5},{12,10}
        };
        int coins_reset[5][2] = {
```

```
			{2,9},{4,1},{6,8},{8,13},{9,4}
		};
		for (int i = 0; i < 13; i++) {
			for (int j = 0; j < 15; j++) {
				bg[i][j] = bg_reset[i][j];
			}
		}
		for (int i = 0; i < 42; i++) {
			block_2_pos[i][0] = obstacle_reset[i][0];
			block_2_pos[i][1] = obstacle_reset[i][1];
			block_2[i].Initialize(block_2_pos[i][1] * 32 + 128, block_2_pos[i][0] * 32 + 32);
		}
		for (int i = 0; i < 5; i++) {
			coins_pos[i][0] = coins_reset[i][0];
			coins_pos[i][1] = coins_reset[i][1];
			coin_Ani[i].Initialize(coins_pos[i][1] * 32 + 128, coins_pos[i][0] * 32 + 32);
		}
		life = 3;
		int health_reset[8] = { 2, 2, 2, 2, 2, 2, 2, 2};
		for (int i = 0; i < 8; i++) {
			heart_num[i] = health_reset[i];
		}
		blood_ori = blood_vol = 16;		// 預設血量總值為 16
		// 腳色數值重置
		character_1.Initialize(128, 32);
		character_1.LoadMap(bg);
		coins_num = 5;			// 該關卡共有 5 個金幣
		Enemy1_num = 1;			// 敵人 1 的數量
		Enemy2_num = 1;			// 敵人 2 的數量
		AI[0].Initialize(6 * 32 + 128, 4 * 32 + 32);
		AI[1].Initialize(12 * 32 + 128,   8 * 32 + 32);
		for (int i = 0; i < 2; i++) {
			AI[i].LoadMap(bg);
		}
		score = 0;
		timer = 0;
		CAudio::Instance()->Play(AUDIO_BGM1, true);
}
void GameStage_1::OnInit() {
		timer = 250;
		level.LoadBitmap(IDB_LEVEL_1);
		block_0.LoadBitmap(IDB_Bg_1, RGB(255, 255, 255));
		block_1.LoadBitmap(IDB_Blocks, RGB(255, 255, 255));
		for (int i = 0; i < 42; i++) {
			block_2[i].LoadBitmap();
		}
		border.LoadBitmap(IDB_BORDER_0, RGB(255, 255, 255));
		panel.LoadBitmap(IDB_Panel, RGB(255, 255, 255));
		character_1.LoadBitmap();
		for (int i = 0; i < 7; i++) {
			Bomb_ch1[i].LoadBitmap();
		}
		for (int i = 0; i < 5; i++) {
			coin_Ani[i].LoadBitmap();
		}
		for (int i = 0; i < 2; i++) {
			AI[i].LoadBitmap();
		}
		playerhead_1.LoadBitmap(IDB_PLAYERHEAD1, RGB(255, 255, 255));
		playerhead_2.LoadBitmap(IDB_PLAYERHEAD2, RGB(255, 255, 255));
		for (int i = 0; i < 8; i++) {
			heart[i].LoadBitmap();
		}
		count_down.SetInteger(60);
		/*
		for (int i = 0; i < 7; i++) {
			CAudio::Instance()->Load(AUDIO_BOMB + i, "sounds\\POWER.wav");
			TRACE("%d\n", AUDIO_BOMB + i);
		}
```

```
	*/
	CAudio::Instance()->Load(AUDIO_BOMB, "sounds\\POWER.wav");
	CAudio::Instance()->Load(AUDIO_BGM1, "sounds\\stage1BGM.wav");
	CAudio::Instance()->Load(AUDIO_OOF, "sounds\\player1_hurt.mp3");
}
void GameStage_1::OnMove() {
	timer++;
	int second = timer / 30;
	// int min = second / 60;
	second %= 60;
	// TRACE("second %d\n", second);
	// TRACE("min %d\n", min);
	if (!(timer % 30))
		count_down.Add(-1);
	bool nextState = false;
	for (int i = 0; i < 2; i++) {
		nextState = nextState | AI[i].Alive();
	}
	if (!nextState) {
		CAudio::Instance()->Stop(AUDIO_BGM2);
		int data[3] = {score, life, blood_vol };
		game->saveData(data, 3);
		GotoGameState(GAME_STAGE_2);
	}
	for (int i = 0; i < 42; i++) {
		block_2[i].OnMove();
		if (block_2[i].getActive() && !block_2[i].getExp()) {
			mapChange(block_2_pos[i][1], block_2_pos[i][0], 0);
			block_2[i].setActive(false);
			block_2[i].setExp(true);
		}
	}
	BombState();
	character_1.OnMove();
	for (int i = 0; i < 2; i++) {
		AI[i].OnMove(character_1.GetX1(), character_1.GetY1(), 0, 0);
	}
	GetCoins();
	HealthState();
	if (blood_vol > 0) {
		character_1.SetDead(false);
	}
	if (blood_vol == 0 && life != 1) {
		character_1.SetDead(true);
		life--;
		int health_reset[8] = { 2, 2, 2, 2, 2, 2, 2, 2};
		for (int i = 0; i < 8; i++) {
			heart_num[i] = health_reset[i];
		}
		blood_ori = blood_vol = 16;		// 預設血量總值為 16
	}
	else if (blood_vol == 0 && life == 1) {
		life--;
		CAudio::Instance()->Stop(AUDIO_BGM1);
		int data[1] = { score };
		game->saveData(data, 1);
		GotoGameState(GAME_STATE_OVER);
	}
	// 判斷敵人被殺死後給予得分
	if (!(AI[0].Alive()) && Enemy1_num > 0) {
		score += 100;
		Enemy1_num--;
	}
	if (!(AI[1].Alive()) && Enemy2_num > 0) {
		score += 100;
		Enemy2_num--;
	}
}
void GameStage_1::OnShow() {				// 越後放的顯示會越上層
```

```
panel.SetTopLeft(0, 0);
panel.ShowBitmap();
border.SetTopLeft(96, 0);
border.ShowBitmap();
level.SetTopLeft(609, 0);
level.ShowBitmap();
for (int i = 0; i < 13; i++) {                    // 方塊顯示  j 是 X 軸 i 是 Y 軸
        for (int j = 0; j < 15; j++) {
                if (bg[i][j] == 1) {
                        block_1.SetTopLeft(128 + 32 * j, 32 * (i + 1));
                        block_1.ShowBitmap();
                }
                else {
                        block_0.SetTopLeft(128 + 32 * j, 32 * (i + 1));
                        block_0.ShowBitmap();
                }
        }
}
for (int i = 0; i < 42; i++) {
        block_2[i].OnShow();
}
for (int i = 0; i < coins_num; i++) {
        coin_Ani[i].setTopLeft(128 + coins_pos[i][1] * 32, 32 * (coins_pos[i][0] + 1));
        coin_Ani[i].OnShow();
}
count_down.SetTopLeft(panel.Width() * 25 / 100, panel.Height() * 48 / 100);
count_down.LoadBitmap();
count_down.ShowBitmap();
character_1.OnShow();
for (int i = 0; i < 7; i++) {
        Bomb_ch1[i].OnShow();
}
for (int i = 0; i < 2; i++) {
        AI[i].OnShow();
}
playerhead_1.SetTopLeft((panel.Width() * 16 / 100), panel.Height() * 13 / 100);
playerhead_1.ShowBitmap();
CDC* pDC = CDDraw::GetBackCDC();                  // 取得 Back Plain 的 CDC
CFont f, * fp;
f.CreatePointFont(160, "Times New Roman");        // 產生 font f; 160 表示 16point 的字
fp = pDC->SelectObject(&f);                       // 選用 font f
pDC->SetBkMode(TRANSPARENT);
pDC->SetBkColor(RGB(0, 0, 255));
pDC->SetTextColor(RGB(255, 255, 255));
char str[80];                                     // Demo 數字對字串的轉換
sprintf(str, "X %d", life);
pDC->TextOut((panel.Width() * 57 / 100), panel.Height() * 13 / 100, str);
pDC->SelectObject(fp);                            // 放掉 font f (千萬不要漏了放掉)
CDDraw::ReleaseBackCDC();                         // 放掉 Back Plain 的 CDC
CDC* pDC1 = CDDraw::GetBackCDC();                 // 取得 Back Plain 的 CDC
CFont f1, * fp1;
f1.CreatePointFont(140, "Times New Roman");       // 產生 font f; 160 表示 16point 的字
fp1 = pDC1->SelectObject(&f1);                    // 選用 font f
pDC1->SetBkMode(TRANSPARENT);
pDC1->SetBkColor(RGB(0, 0, 255));
pDC1->SetTextColor(RGB(255, 255, 255));
pDC1->TextOut((panel.Width() * 16 / 100), panel.Height() * 375 / 1000, "SCORE");
char str1[80];                                    // Demo 數字對字串的轉換
sprintf(str1, "%06d", score);
pDC1->TextOut((panel.Width() * 20 / 100), panel.Height() * 41 / 100, str1);
pDC1->SelectObject(fp1);                          // 放掉 font f (千萬不要漏了放掉)
CDDraw::ReleaseBackCDC();                         // 放掉 Back Plain 的 CDC
playerhead_2.SetTopLeft((panel.Width() * 16 / 100), panel.Height() * 56 / 100);
playerhead_2.ShowBitmap();
CDC* pDC2 = CDDraw::GetBackCDC();                 // 取得 Back Plain 的 CDC
CFont f2, * fp2;
f2.CreatePointFont(160, "Times New Roman");       // 產生 font f; 160 表示 16point 的字
fp2 = pDC2->SelectObject(&f2);                    // 選用 font f
pDC2->SetBkMode(TRANSPARENT);
```

```
        pDC2->SetBkColor(RGB(0, 0, 255));
        pDC2->SetTextColor(RGB(255, 255, 255));
        char str2[80];                                      // Demo 數字對字串的轉換
        sprintf(str2, "X %d", 0);
        pDC2->TextOut((panel.Width() * 57 / 100), panel.Height() * 56 / 100, str2);
        pDC2->SelectObject(fp2);                            // 放掉 font f (千萬不要漏了放掉)
        CDDraw::ReleaseBackCDC();                           // 放掉 Back Plain 的 CDC
        CMovingBitmap playerover;
        playerover.LoadBitmap(IDB_PLAYER_GAMEOVER, RGB(255, 255, 0));
        playerover.SetTopLeft(panel.Width() * 10 / 100, panel.Height() * 61 / 100);
        playerover.ShowBitmap();
        for (int i = 0; i < 2; i++) {
                for (int j = 0; j < 8; j++) {
                        if (i == 0 && j < 4) {
                                heart[j].setTopLeft((panel.Width() * 15 / 100) + 17 * j, panel.Height() * 19 / 100);
                                heart[j].OnShow();
                        }
                        else if (i == 1 && j > 3 &&j < 8) {
                                heart[j].setTopLeft((panel.Width() * 15 / 100) + 17 * (j - 4), panel.Height() * 225 / 1000);
                                heart[j].OnShow();
                        }
                }
        }
}
void GameStage_1::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_ESC = 0x1B;
        const char KEY_P = 0x50;
        const char KEY_SPACE = 0x20;
        const char KET_Y = 0x59;
        if (nChar == KEY_ESC || nChar == KEY_P) {
                // game_framework::CGame::Instance()->OnFilePause();
                CAudio::Instance()->Pause();
                game_framework::CGame::Instance()->SaveState(this);
                GotoGameState(GAME_STATE_PAUSE);
        }
        if (nChar == KEY_LEFT) {
                character_1.SetMovingLeft(true);
        }
        if (nChar == KEY_RIGHT) {
                character_1.SetMovingRight(true);
        }
        if (nChar == KEY_UP) {
                character_1.SetMovingUp(true);
        }
        if (nChar == KEY_DOWN) {
                character_1.SetMovingDown(true);
        }
        if (nChar == KEY_SPACE) {
                setBomb(1);
        }
        if (nChar == KET_Y) {
                CAudio::Instance()->Stop(AUDIO_BGM1);
                int data[3] = { score, life, blood_vol };
                game->saveData(data, 3);
                GotoGameState(GAME_STAGE_2);
        }
}
void GameStage_1::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        const char KEY_LEFT = 0x25;               // keyboard 左箭頭
        const char KEY_UP = 0x26;                 // keyboard 上箭頭
        const char KEY_RIGHT = 0x27;              // keyboard 右箭頭
        const char KEY_DOWN = 0x28;               // keyboard 下箭頭
        if (nChar == KEY_LEFT)
```

```
			character_1.SetMovingLeft(false);
		if (nChar == KEY_RIGHT)
			character_1.SetMovingRight(false);
		if (nChar == KEY_UP)
			character_1.SetMovingUp(false);
		if (nChar == KEY_DOWN)
			character_1.SetMovingDown(false);
}
void GameStage_1::setBomb(int id) {
		if (id == 1) {								// [y][x]
			int x = (character_1.GetX1() + character_1.GetX2()) / 2;		// 腳色中心點
			int y = (character_1.GetY1() + character_1.GetY2()) / 2;		// 腳色中心點
			x = (x - 128) / 32;						// 轉換成 13*15 地圖模式
			y = (y - 32) / 32;
			if (bg[y][x] == 0) {
				for (int i = 0; i < 7; i++) {
					if (!Bomb_ch1[i].getActive()) {
						Bomb_ch1[i].setTopleft(x * 32 + 128, (y + 1) * 32);
						Bomb_ch1[i].setActive(true);
						mapChange(x, y, 4);
						break;
					}
				}
			}
			else {
			}
		}
		else if (id == 2) {
			// player2's operating
		}
}
void GameStage_1::mapChange(int x, int y, int value) {
		bg[y][x] = value;
		character_1.LoadMap(bg);
		for (int i = 0; i < 2; i++) {
			AI[i].LoadMap(bg);
		}
}
void GameStage_1::BombState() {
		for (int i = 0; i < 7; i++) {
			Bomb_ch1[i].OnMove();
			if (Bomb_ch1[i].getActive() && !Bomb_ch1[i].getExp()) {
				int nx = (Bomb_ch1[i].getTop_Bomb() - 128) / 32;
				int ny = (Bomb_ch1[i].getLeft_Bomb() - 32) / 32;
				if(bg[ny][nx] != 4)mapChange(nx, ny, 4);
			}
			if (Bomb_ch1[i].getActive() && Bomb_ch1[i].getExp()) {	// 爆炸中的炸彈位置重設成可行走
				int nx = (Bomb_ch1[i].getTop_Bomb() - 128) / 32;
				int ny = (Bomb_ch1[i].getLeft_Bomb() - 32) / 32;
				mapChange(nx, ny, 5);
				if(!Bomb_ch1[i].getObs())setBombRange(1, i, nx, ny);
				if(Bomb_ch1[i].getAud())CAudio::Instance()->Play(AUDIO_BOMB, false);
				for (int j = 1; j <= Bomb_ch1[i].getUp(); j++)mapChange(nx, ny - j, 5);
				for (int j = 1; j <= Bomb_ch1[i].getDown(); j++)mapChange(nx, ny + j, 5);
				for (int j = 1; j <= Bomb_ch1[i].getRight(); j++)mapChange(nx + j, ny, 5);
				for (int j = 1; j <= Bomb_ch1[i].getLeft(); j++)mapChange(nx - j, ny, 5);
			}
			if (!Bomb_ch1[i].getActive() && Bomb_ch1[i].getExp()) {
				int nx = (Bomb_ch1[i].getTop_Bomb() - 128) / 32;
				int ny = (Bomb_ch1[i].getLeft_Bomb() - 32) / 32;
				mapChange(nx, ny, 0);
				for (int j = 1; j <= Bomb_ch1[i].getUp(); j++)mapChange(nx, ny - j, 0);
				for (int j = 1; j <= Bomb_ch1[i].getDown(); j++)mapChange(nx, ny + j, 0);
				for (int j = 1; j <= Bomb_ch1[i].getRight(); j++)mapChange(nx + j, ny, 0);
				for (int j = 1; j <= Bomb_ch1[i].getLeft(); j++)mapChange(nx - j, ny, 0);
				Bomb_ch1[i].Initialize();
			}
		}
}
```

```
void GameStage_1::setBombRange(int id, int i,int x,int y) {
    if (id == 1) {
        int j;
        int range = character_1.GetRange();
        for (j = 1; j <= range; j++) {
            if (y - j < 0 || bg[y - j][x] == 1) {
                break;
            }
            else if (bg[y - j][x] == 2) {
                for (int k = 0; k < 42; k++) {
                    if (block_2_pos[k][0] == y - j && block_2_pos[k][1] == x) {
                        block_2[k].setActive();
                        score += 10;
                        break;
                    }
                }
                break;
            }
        }
        Bomb_ch1[i].setUp(--j);
        for (j = 1; j <= range; j++) {
            if (y + j > 12 || bg[y + j][x] == 1) {
                break;
            }
            else if (bg[y + j][x] == 2) {
                for (int k = 0; k < 42; k++) {
                    if (block_2_pos[k][0] == y + j && block_2_pos[k][1] == x) {
                        block_2[k].setActive();
                        score += 10;
                        break;
                    }
                }
                break;
            }
        }
        Bomb_ch1[i].setDown(--j);
        for (j = 1; j <= range; j++) {
            if (x + j > 14 || bg[y][x+j] == 1) {
                break;
            }
            else if (bg[y][x+j] == 2) {
                for (int k = 0; k < 42; k++) {
                    if (block_2_pos[k][0] == y && block_2_pos[k][1] == x + j) {
                        block_2[k].setActive();
                        score += 10;
                        break;
                    }
                }
                break;
            }
        }
        Bomb_ch1[i].setRight(–j);
        for (j = 1; j <= range; j++) {
            if (x - j < 0 || bg[y][x - j] == 1) {
                break;
            }
            else if (bg[y][x - j] == 2) {
                for (int k = 0; k < 42; k++) {
                    if (block_2_pos[k][0] == y && block_2_pos[k][1] == x - j) {
                        block_2[k].setActive();
                        score += 10;
                        break;
                    }
                }
                break;
            }
        }
        Bomb_ch1[i].setLeft(–j);
        Bomb_ch1[i].setObs(true);
        }
}
void GameStage_1::GetCoins() {
        //找出腳色所在位置(x,y)
        int x = (character_1.GetX1() + character_1.GetX2()) / 2;              //腳色中心點
        int y = (character_1.GetY1() + character_1.GetY2()) / 2;              //腳色中心點
        x = (x - 128) / 32;                                                   //轉換成 13*15 地圖模式
        y = (y - 32) / 32;
        for (int i = 0; i < coins_num; i++) {
                coin_Ani[i].OnMove();
                if (x == coins_pos[i][1] && y == coins_pos[i][0] && !coin_Ani[i].getActive() && !coin_Ani[i].getExp()) {
                        coin_Ani[i].setActive();
                        score += 150;
                }
        }
}
void GameStage_1::HealthState() {
        for (int i = 0; i < 8; i++) {
                heart[i].OnMove();
                if (heart_num[i] == 2) {
                        heart[i].SetDescision(2);
                }
                else if (heart_num[i] == 1) {
                        heart[i].SetDescision(1);
                }
                else if (heart_num[i] == 0) {
                        heart[i].SetDescision(0);
                }
        }
        int x = (character_1.GetX1() + character_1.GetX2()) / 2;              // 腳色中心點
        int y = (character_1.GetY1() + character_1.GetY2()) / 2;              // 腳色中心點
        x = (x - 128) / 32;                                                   // 轉換成 13*15 地圖模式
        y = (y - 32) / 32;
        // TRACE("%d\n", character_1.GetDead());
        if (!character_1.GetDead()) {
                if (taking_Damage) {
                        // wait two seconds
                        k++;
                        if (k == 60) {
                                taking_Damage = false;
                                k = 0;
                        }
                } else {
                        if (bg[y][x] == 5) {
                                CAudio::Instance()->Play(AUDIO_OOF, false);
                                blood_vol = blood_vol - 1;
                                taking_Damage = true;
                                TRACE("%d %d %d\n", life, blood_vol, blood_ori);
                        }
                        for (int i = 0; i < 2; i++) {
                                int x1 = (AI[i].GetX1() + AI[i].GetX2()) / 2;         // 敵人中心點
                                int y1 = (AI[i].GetY1() + AI[i].GetY2()) / 2;         // 敵人中心點
                                x1 = (x1 - 128) / 32;                                 // 轉換成 13*15 地圖模式
                                y1 = (y1 - 32) / 32;
                                if (x == x1 && y == y1 && AI[i].Alive()) {
                                        CAudio::Instance()->Play(AUDIO_OOF, false);
                                        blood_vol = blood_vol - 1;
                                        taking_Damage = true;
                                        TRACE("%d %d %d\n", life, blood_vol, blood_ori);
                                }
                                if (AI[i].BulletHitPlayer() && AI[i].Alive()) {
                                        CAudio::Instance()->Play(AUDIO_OOF, false);
                                        blood_vol = blood_vol - 1;
                                        taking_Damage = true;
                                        TRACE("%d %d %d\n", life, blood_vol, blood_ori);
                                }
                        }
                }
        }
```

```
		double value = std::fmod(blood_ori, blood_vol);
		for (int i = 7; i >= 0; i--) {
			if (heart_num[i] != 0) {
				if (heart_num[i] - value < 0) {
					value -= heart_num[i];
					heart_num[i] = 0;
					blood_ori = blood_vol;
				}
				else if (heart_num[i] - value == 1) {
					heart_num[i] = 1;
					blood_ori = blood_vol;
					value = 0;
				}
				else if (heart_num[i] - value == 0) {
					heart_num[i] = 0;
					blood_ori = blood_vol;
					value = 0;
				}
			}
		}
}
/////////////////////////////////////////////////////////////////////////////
//Prefences
/////////////////////////////////////////////////////////////////////////////
GamePrefences::GamePrefences(CGame* g) : CGameState(g)
{
}
void GamePrefences::OnInit()
{
		scr.LoadBitmap(IDB_SCREEN_PREFENCES, RGB(0, 0, 255));
		scr_ok.LoadBitmap(IDB_SCREEN_OK, RGB(0, 0, 255));
		scr_cancel.LoadBitmap(IDB_SCREEN_CANCEL, RGB(0, 0, 255));
		scr_FX_down.LoadBitmap(IDB_SCREEN_MINUX, RGB(0, 0, 255));
		scr_FX_up.LoadBitmap(IDB_SCREEN_PLUS, RGB(0, 0, 255));
		scr_FS_yes.LoadBitmap(IDB_SCREEN_YES, RGB(0, 0, 255));
		scr_FS_no.LoadBitmap(IDB_SCREEN_NO, RGB(0, 0, 255));
		scr_FR.LoadBitmap(IDB_SCREEN_MINUX, RGB(0, 0, 255));
		scr_SF_yes.LoadBitmap(IDB_SCREEN_YES, RGB(0, 0, 255));
		scr_SF_no.LoadBitmap(IDB_SCREEN_NO, RGB(0, 0, 255));
		scr_Vsync_yes.LoadBitmap(IDB_SCREEN_YES, RGB(0, 0, 255));
		scr_Vsync_no.LoadBitmap(IDB_SCREEN_NO, RGB(0, 0, 255));
}
void GamePrefences::OnBeginState()
{
		form_ori = form_state;
		form_state = 3;
		FS_ori_state = FS_state;
		SF_ori_state = SF_state;
		Vsync_ori_state = Vsync_state;
}
void GamePrefences::OnLButtonDown(UINT nFlags, CPoint point)
{
		if (p.x > scr_ok.Left() && p.x < scr_ok.Left() + scr_ok.Width() &&
			p.y > scr_ok.Top() && p.y < scr_ok.Top() + scr_ok.Height()) {
			// OK
			if (form_ori == 1) {
				GotoGameState(GAME_STATE_INIT);
			} else if (form_ori == 2) {
				GotoGameState(GAME_STATE_PAUSE);
			}
		}
		else if (p.x > scr_cancel.Left() && p.x < scr_cancel.Left() + scr_cancel.Width() &&
			p.y > scr_cancel.Top() && p.y < scr_cancel.Top() + scr_cancel.Height()) {
			// Cancel
			FS_state = FS_ori_state;
			SF_state = SF_ori_state;
			Vsync_state = Vsync_ori_state;
			if (form_ori == 1) {
				GotoGameState(GAME_STATE_INIT);
```

```
                } else if (form_ori == 2) {
                        GotoGameState(GAME_STATE_PAUSE);
                }
        }
        if (p.x > scr_FS_no.Left() && p.x < scr_FS_no.Left() + scr_FS_no.Width() &&
                p.y > scr_FS_no.Top() && p.y < scr_FS_no.Top() + scr_FS_no.Height() && FS_state == 0) {
                FS_state = 1;
        } else if (p.x > scr_FS_yes.Left() && p.x < scr_FS_yes.Left() + scr_FS_yes.Width() &&
                p.y > scr_FS_yes.Top() && p.y < scr_FS_yes.Top() + scr_FS_yes.Height() && FS_state == 1) {
                FS_state = 0;
        }
        if (p.x > scr_SF_no.Left() && p.x < scr_SF_no.Left() + scr_SF_no.Width() &&
                p.y > scr_SF_no.Top() && p.y < scr_SF_no.Top() + scr_SF_no.Height() && SF_state == 0) {
                SF_state = 1;
        } else if (p.x > scr_SF_yes.Left() && p.x < scr_SF_yes.Left() + scr_SF_yes.Width() &&
                p.y > scr_SF_yes.Top() && p.y < scr_SF_yes.Top() + scr_SF_yes.Height() && SF_state == 1) {
                SF_state = 0;
        }
        if (p.x > scr_Vsync_no.Left() && p.x < scr_Vsync_no.Left() + scr_Vsync_no.Width() &&
                p.y > scr_Vsync_no.Top() && p.y < scr_Vsync_no.Top() + scr_Vsync_no.Height() && Vsync_state == 0) {
                Vsync_state = 1;
        } else if (p.x > scr_Vsync_yes.Left() && p.x < scr_Vsync_yes.Left() + scr_Vsync_yes.Width() &&
                p.y > scr_Vsync_yes.Top() && p.y < scr_Vsync_yes.Top() + scr_Vsync_yes.Height() && Vsync_state == 1) {
                Vsync_state = 0;
        }
}
void GamePrefences::OnMouseMove(UINT nFlags, CPoint point)
{
        GetCursorPos(&p);
        ScreenToClient(AfxGetMainWnd()->m_hWnd, &p);        // 把螢幕座標轉換為視窗座標，並讀取出來
        // TRACE("mouse:: x: %d, y: %d\n", p.x, p.y);
}
void GamePrefences::OnShow()
{
        bg.LoadBitmap(IDB_SCREENBG1, RGB(0, 0, 0));
        for (int x = 0; x < SIZE_X; x += bg.Width()) {
                for (int y = 0; y < SIZE_Y; y += bg.Height()) {
                        bg.SetTopLeft(x, y);
                        bg.ShowBitmap();
                }
        }
        scr.SetTopLeft(SIZE_X * 4 / 1000, SIZE_Y * 4 / 1000);
        scr.ShowBitmap();
        // OK
        if (p.x > scr_ok.Left() && p.x < scr_ok.Left() + scr_ok.Width() &&
                p.y > scr_ok.Top() && p.y < scr_ok.Top() + scr_ok.Height()) {
                CMovingBitmap scr_ok_red;
                scr_ok_red.LoadBitmap(IDB_SCREEN_OK_RED, RGB(0, 0, 255));
                scr_ok_red.SetTopLeft((SIZE_X - scr_ok_red.Width()) * 34 / 100, SIZE_Y * 85 / 100);
                scr_ok_red.ShowBitmap();
        } else {
                scr_ok.SetTopLeft((SIZE_X - scr_ok.Width()) * 34 / 100, SIZE_Y * 85 / 100);
                scr_ok.ShowBitmap();
        }
        // Cancel
        if (p.x > scr_cancel.Left() && p.x < scr_cancel.Left() + scr_cancel.Width() &&
                p.y > scr_cancel.Top() && p.y < scr_cancel.Top() + scr_cancel.Height()) {
                CMovingBitmap scr_cancel_red;
                scr_cancel_red.LoadBitmap(IDB_SCREEN_CANCEL_RED, RGB(0, 0, 255));
                scr_cancel_red.SetTopLeft((SIZE_X - scr_cancel_red.Width()) * 68 / 100, SIZE_Y * 85 / 100);
                scr_cancel_red.ShowBitmap();
        } else {
                scr_cancel.SetTopLeft((SIZE_X - scr_cancel.Width()) * 68 / 100, SIZE_Y * 85 / 100);
                scr_cancel.ShowBitmap();
        }
        // FX_down
        if (p.x > scr_FX_down.Left() && p.x < scr_FX_down.Left() + scr_FX_down.Width() &&
                p.y > scr_FX_down.Top() && p.y < scr_FX_down.Top() + scr_FX_down.Height()) {
                CMovingBitmap scr_FX_down_red;
```

```
            scr_FX_down_red.LoadBitmap(IDB_SCREEN_MINUX_RED, RGB(0, 0, 255));
            scr_FX_down_red.SetTopLeft((SIZE_X - scr_FX_down_red.Width()) * 36 / 100, SIZE_Y * 11 / 100);
            scr_FX_down_red.ShowBitmap(1.6);
        } else {
            scr_FX_down.SetTopLeft((SIZE_X - scr_FX_down.Width()) * 36 / 100, SIZE_Y * 11 / 100);
            scr_FX_down.ShowBitmap(1.6);
        }
        // FX_up
        if (p.x > scr_FX_up.Left() && p.x < scr_FX_up.Left() + scr_FX_up.Width() &&
            p.y > scr_FX_up.Top() && p.y < scr_FX_up.Top() + scr_FX_up.Height()) {
            CMovingBitmap scr_FX_up_red;
            scr_FX_up_red.LoadBitmap(IDB_SCREEN_PLUS_RED, RGB(0, 0, 255));
            scr_FX_up_red.SetTopLeft((SIZE_X - scr_FX_up_red.Width()) * 71 / 100, SIZE_Y * 115 / 1000);
            scr_FX_up_red.ShowBitmap(1.6);
        } else {
            scr_FX_up.SetTopLeft((SIZE_X - scr_FX_up.Width()) * 71 / 100, SIZE_Y * 115 / 1000);
            scr_FX_up.ShowBitmap();
        }
        // Fullscreen
        if (FS_state == 1) {
            //scr_FS_yes
            if (p.x > scr_FS_yes.Left() && p.x < scr_FS_yes.Left() + scr_FS_yes.Width() &&
                p.y > scr_FS_yes.Top() && p.y < scr_FS_yes.Top() + scr_FS_yes.Height()) {
                CMovingBitmap scr_FS_yes_red;
                scr_FS_yes_red.LoadBitmap(IDB_SCREEN_YES_RED, RGB(0, 0, 255));
                scr_FS_yes_red.SetTopLeft((SIZE_X - scr_FS_yes_red.Width()) * 39 / 100, SIZE_Y * 22 / 100);
                scr_FS_yes_red.ShowBitmap();
            } else {
                scr_FS_yes.SetTopLeft((SIZE_X - scr_FS_yes.Width()) * 39 / 100, SIZE_Y * 22 / 100);
                scr_FS_yes.ShowBitmap();
            }
        } else if (FS_state == 0) {
            //scr_FS_no
            if (p.x > scr_FS_no.Left() && p.x < scr_FS_no.Left() + scr_FS_no.Width() &&
                p.y > scr_FS_no.Top() && p.y < scr_FS_no.Top() + scr_FS_no.Height()) {
                CMovingBitmap scr_FS_no_red;
                scr_FS_no_red.LoadBitmap(IDB_SCREEN_NO_RED, RGB(0, 0, 255));
                scr_FS_no_red.SetTopLeft((SIZE_X - scr_FS_no_red.Width()) * 38 / 100, SIZE_Y * 22 / 100);
                scr_FS_no_red.ShowBitmap();
            } else {
                scr_FS_no.SetTopLeft((SIZE_X - scr_FS_no.Width()) * 38 / 100, SIZE_Y * 22 / 100);
                scr_FS_no.ShowBitmap();
            }
        }
        // scr_FR: Fllscr.Res
        if (p.x > scr_FR.Left() && p.x < scr_FR.Left() + scr_FR.Width() &&
            p.y > scr_FR.Top() && p.y < scr_FR.Top() + scr_FR.Height()) {
            CMovingBitmap scr_FR_red;
            scr_FR_red.LoadBitmap(IDB_SCREEN_MINUX_RED, RGB(0, 0, 255));
            scr_FR_red.SetTopLeft((SIZE_X - scr_FR_red.Width()) * 38 / 100, SIZE_Y * 31 / 100);
            scr_FR_red.ShowBitmap();
        } else {
            scr_FR.SetTopLeft((SIZE_X - scr_FR.Width()) * 38 / 100, SIZE_Y * 31 / 100);
            scr_FR.ShowBitmap();
        }
        // Show FPS
        if (SF_state == 1) {
            // scr_SF_yes
            if (p.x > scr_SF_yes.Left() && p.x < scr_SF_yes.Left() + scr_SF_yes.Width() &&
                p.y > scr_SF_yes.Top() && p.y < scr_SF_yes.Top() + scr_SF_yes.Height()) {
                CMovingBitmap scr_SF_yes_red;
                scr_SF_yes_red.LoadBitmap(IDB_SCREEN_YES_RED, RGB(0, 0, 255));
                scr_SF_yes_red.SetTopLeft((SIZE_X - scr_SF_yes_red.Width()) * 39 / 100, SIZE_Y * 35 / 100);
                scr_SF_yes_red.ShowBitmap();
            } else {
                scr_SF_yes.SetTopLeft((SIZE_X - scr_SF_yes.Width()) * 39 / 100, SIZE_Y * 35 / 100);
                scr_SF_yes.ShowBitmap();
            }
        } else if (SF_state == 0) {
```

```
			// scr_SF_no
			if (p.x > scr_SF_no.Left() && p.x < scr_SF_no.Left() + scr_SF_no.Width() &&
				p.y > scr_SF_no.Top() && p.y < scr_SF_no.Top() + scr_SF_no.Height()) {
				CMovingBitmap scr_SF_no_red;
				scr_SF_no_red.LoadBitmap(IDB_SCREEN_NO_RED, RGB(0, 0, 255));
				scr_SF_no_red.SetTopLeft((SIZE_X - scr_SF_no_red.Width()) * 38 / 100, SIZE_Y * 35 / 100);
				scr_SF_no_red.ShowBitmap();
			} else {
				scr_SF_no.SetTopLeft((SIZE_X - scr_SF_no.Width()) * 38 / 100, SIZE_Y * 35 / 100);
				scr_SF_no.ShowBitmap();
			}
		}
		// Vsync
		if (Vsync_state == 1) {
			// scr_Vsync_yes
			if (p.x > scr_Vsync_yes.Left() && p.x < scr_Vsync_yes.Left() + scr_Vsync_yes.Width() &&
				p.y > scr_Vsync_yes.Top() && p.y < scr_Vsync_yes.Top() + scr_Vsync_yes.Height()) {
				CMovingBitmap scr_Vsync_yes_red;
				scr_Vsync_yes_red.LoadBitmap(IDB_SCREEN_YES_RED, RGB(0, 0, 255));
				scr_Vsync_yes_red.SetTopLeft((SIZE_X - scr_Vsync_yes_red.Width()) * 39 / 100, SIZE_Y * 43 / 100);
				scr_Vsync_yes_red.ShowBitmap();
			} else {
				scr_Vsync_yes.SetTopLeft((SIZE_X - scr_Vsync_yes.Width()) * 39 / 100, SIZE_Y * 43 / 100);
				scr_Vsync_yes.ShowBitmap();
			}
		} else if (Vsync_state == 0) {
			// scr_Vsync_no
			if (p.x > scr_Vsync_no.Left() && p.x < scr_Vsync_no.Left() + scr_Vsync_no.Width() &&
				p.y > scr_Vsync_no.Top() && p.y < scr_Vsync_no.Top() + scr_Vsync_no.Height()) {
				CMovingBitmap scr_Vsync_no_red;
				scr_Vsync_no_red.LoadBitmap(IDB_SCREEN_NO_RED, RGB(0, 0, 255));
				scr_Vsync_no_red.SetTopLeft((SIZE_X - scr_Vsync_no_red.Width()) * 38 / 100, SIZE_Y * 43 / 100);
				scr_Vsync_no_red.ShowBitmap();
			} else {
				scr_Vsync_no.SetTopLeft((SIZE_X - scr_Vsync_no.Width()) * 38 / 100, SIZE_Y * 43 / 100);
				scr_Vsync_no.ShowBitmap();
			}
		}
}
/////////////////////////////////////////////////////////////////////////////
//About
/////////////////////////////////////////////////////////////////////////////
GameAbout::GameAbout(CGame* g) : CGameState(g)
{
}
void GameAbout::OnInit()
{
	aboutForm.LoadBitmap(IDB_SCR_ABOUT, RGB(0, 0, 255));
	scr_back.LoadBitmap(IDB_SCREEN_BACK, RGB(0, 0, 255));
}
void GameAbout::OnBeginState()
{
}
void GameAbout::OnLButtonDown(UINT nFlags, CPoint point)
{
	if (p.x > scr_back.Left() && p.x < scr_back.Left() + scr_back.Width() &&
		p.y > scr_back.Top() && p.y < scr_back.Top() + scr_back.Height()) {
		GotoGameState(GAME_STATE_INIT);
	}
}
void GameAbout::OnMouseMove(UINT nFlags, CPoint point)
{
	GetCursorPos(&p);
	ScreenToClient(AfxGetMainWnd()->m_hWnd, &p);		// 把螢幕座標轉換為視窗座標，並讀取出來
	// TRACE("mouse:: x: %d, y: %d\n", p.x, p.y);
}
void GameAbout::OnShow()
{
	bg.LoadBitmap(IDB_SCREENBG1, RGB(0, 0, 0));
```

```
        for (int x = 0; x < SIZE_X; x += bg.Width()) {
            for (int y = 0; y < SIZE_Y; y += bg.Height()) {
                bg.SetTopLeft(x, y);
                bg.ShowBitmap();
            }
        }
        aboutForm.SetTopLeft(0, 0);
        aboutForm.ShowBitmap();
        // Back
        if (p.x > scr_back.Left() && p.x < scr_back.Left() + scr_back.Width() &&
            p.y > scr_back.Top() && p.y < scr_back.Top() + scr_back.Height()) {
            CMovingBitmap scr_back_red;
            scr_back_red.LoadBitmap(IDB_SCREEN_BACK_RED, RGB(0, 0, 255));
            scr_back_red.SetTopLeft((SIZE_X - scr_back_red.Width()) / 2, SIZE_Y * 88 / 100);
            scr_back_red.ShowBitmap();
        } else {
            scr_back.SetTopLeft((SIZE_X - scr_back.Width()) / 2, SIZE_Y * 88 / 100);
            scr_back.ShowBitmap();
        }
}
}
```

Character.h

```
#define STEP    4
namespace game_framework {
        class Character {
        public:
                Character();
                int   GetX1();                        // 腳色左上角 x 座標
                int   GetY1();                        // 腳色左上角 y 座標
                int   GetX2();                        // 腳色右下角 x 座標
                int   GetY2();                        // 腳色右下角 y 座標
                int   GetStep();                      // 腳色步數
                int   GetRange();                     // 爆炸距離
                bool GetDead();                       // 腳色是否死亡
                void Initialize(int nx, int ny);// 設定腳色為初始值 對不同腳色設定初始位置
                void LoadBitmap();                    // 載入圖形
                void OnMove();                        // 移動腳色
                void OnShow();                        // 將腳色圖形貼到畫面
                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 SetDead(bool flag);
                void SetXY(int nx, int ny);           // 設定腳色左上角座標
                void SetRange(int);                   // 設定爆炸距離
                void LoadMap(int maps[13][15]);
                int   GetPosition(int, int);
        private:
                CAnimation Character_down;            // 腳色向下的動畫
                CAnimation Character_up;
                CAnimation Character_left;
                CAnimation Character_right;
                CAnimation Character_dead;            // 腳色死亡
                int   Animate_State;                  // 腳色移動狀態 1 為下 2 為上 3 為左 4 為右
                int   x, y;                           // 腳色左上角座標
                int   move_step = STEP;               // 腳色步數
                int   Explosion_range;                // 最大爆炸距離
                //int   Bomb_count;                   // 可用炸彈數
                bool isMovingDown;                    // 是否正在往下移動
                bool isMovingLeft;                    // 是否正在往左移動
                bool isMovingRight;                   // 是否正在往右移動
                bool isMovingUp;                      // 是否正在往上移動
                bool isDead;                          // 是否死亡
                int   map_simple[13][15];
                int   map[416][480];
                bool isMoveable(int, int);
        };
}
/*
* 註一:炸彈在短時間內重複要求放置會有機率在一個位置放置兩個以上 推測是因為在地圖更新之前又要求設置
*/
```

Character.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Character.h"
namespace game_framework {
Character::Character() {
        Initialize(0,0);
}
void Character::Initialize(const int nx, const int ny) {
        x = nx;
        y = ny;
        Animate_State= 1;
        isMovingDown = isMovingLeft = isMovingRight = isMovingUp = false;
        Explosion_range= 5;
        isDead = false;
}
int Character::GetX1() {
        return x;
}
int Character::GetY1() {
        return y;
}
int Character::GetX2() {
        return x + Character_down.Width();
}
int Character::GetY2() {
        return y + Character_down.Width();
}
int Character::GetStep() {
        return move_step;
}
int Character::GetRange() {
        return Explosion_range;
}
bool Character::GetDead() {
        return isDead;
}
void Character::LoadBitmap() {
        Character_down.SetDelayCount(5);
        Character_down.AddBitmap(IDB_PLAYER1_DW_1, RGB(255,255, 255));
        Character_down.AddBitmap(IDB_PLAYER1_DW_2, RGB(255,255, 255));
        Character_up.SetDelayCount(5);
        Character_up.AddBitmap(IDB_PLAYER1_UP_1,RGB(255,255, 255));
        Character_up.AddBitmap(IDB_PLAYER1_UP_2,RGB(255,255, 255));
        Character_left.SetDelayCount(5);
        Character_left.AddBitmap(IDB_PLAYER1_LE_1,RGB(255,255, 255));
        Character_left.AddBitmap(IDB_PLAYER1_LE_2,RGB(255,255, 255));
        Character_right.SetDelayCount(5);
        Character_right.AddBitmap(IDB_PLAYER1_RE_1, RGB(255, 255, 255));
        Character_right.AddBitmap(IDB_PLAYER1_RE_2, RGB(255, 255, 255));
        Character_dead.SetDelayCount(5);
        Character_dead.AddBitmap(IDB_PLAYER1_DEAD_1, RGB(255, 255, 255));
        Character_dead.AddBitmap(IDB_PLAYER1_DEAD_2, RGB(255, 255, 255));
        Character_dead.AddBitmap(IDB_PLAYER1_DEAD_3, RGB(255, 255, 255));
        Character_dead.AddBitmap(IDB_PLAYER1_DEAD_4, RGB(255, 255, 255));
}
void Character::OnMove() {
        if (isMovingDown) {
                Animate_State= 1;
                Character_down.OnMove();
                int x1 = x - 128;
                int x2 = x + Character_down.Width() - 128 - 1;
                int y2 = y + Character_down.Height() - 32 - 1;
                if (isMoveable(x1,y2 + move_step) && isMoveable(x2, y2 + move_step)) {       // 邊界判定
                        y += move_step;
```

```
			}
			else if (isMoveable(x1, y2 + move_step - 2) && isMoveable(x2, y2 + move_step - 2)) {
				y += 2;
			}
			else if (GetPosition(x1,y2) == 4 && GetPosition(x1, y2 + move_step) == 4 && y2 / 32 == (y2 + move_step) /
32) {			// 在剛設下的 32*32 炸彈空間內可以正常移動 但離開後不能重新回來
						y += move_step;
			}
			else if (GetPosition(x1, y2) == 4 && GetPosition(x1,y2 + move_step - 2) == 4 && y2 / 32 == (y2 + move_step
- 2) / 32) {
				y += 2;
			}
		}
		if (isMovingUp) {
			Animate_State = 2;
			Character_up.OnMove();
			int x1 = x - 128;
			int x2 = x + Character_up.Width() - 128 - 1;
			int y1 = y - 32;
			if (isMoveable(x1,y1 - move_step) && isMoveable(x2, y1 - move_step))
				y -= move_step;
			else if (isMoveable(x1, y1 - move_step + 2) && isMoveable(x2, y1 - move_step + 2))		// 邊界判定
				y -= 2;
			else if (GetPosition(x1, y1) == 4 && GetPosition(x1, y1 - move_step) == 4 && y1 / 32 == (y1 - move_step) /
32)
				y -= move_step;
			else if (GetPosition(x1, y1) == 4 && GetPosition(x1,y1 - move_step + 2) == 4 && y1 / 32 == (y1 - move_step
+ 2) / 32)
				y -= 2;
		}
		if (isMovingLeft) {
			Animate_State = 3;
			Character_left.OnMove();
			int x1 = x - 128;
			int y1 = y - 32;
			int y2 = y + Character_left.Height() - 32 - 1;
			if (isMoveable(x1 - move_step, y1) && isMoveable(x1 - move_step, y2))
				x -= move_step;
			else if (isMoveable(x1 - move_step + 2, y1) && isMoveable(x1 - move_step + 2, y2))
				x -= 2;
			else if (GetPosition(x1, y1) == 4 && GetPosition(x1 - move_step, y1) == 4 && x1 / 32 == (x1 - move_step) /
32)
				x -= move_step;
			else if (GetPosition(x1, y1) == 4 && GetPosition(x1 - move_step + 2, y1) == 4 && x1 / 32 == (x1 - move_step
+ 2) / 32)
				x -= 2;
		}
		if (isMovingRight) {
			Animate_State = 4;
			Character_right.OnMove();
			int x2 = x + Character_right.Width() - 128 - 1;
			int y1 = y - 32;
			int y2 = y + Character_right.Height() - 32 - 1;
			if (isMoveable(x2 + move_step, y1) && isMoveable(x2 + move_step, y2))
				x += move_step;
			else if (isMoveable(x2 + move_step - 2, y1) && isMoveable(x2 + move_step - 2, y2))
				x += 2;
			else if (GetPosition(x2, y1) == 4 && GetPosition(x2 + move_step, y1) == 4 && x2 / 32 == (x2 + move_step) /
32)
				x += move_step;
			else if (GetPosition(x2, y1) == 4 && GetPosition(x2 + move_step - 2, y1) == 4 && x2 / 32 == (x2 + move_step
- 2) / 32)
				x += 2;
		}
		if (isDead) {
			Animate_State = 5;
			Character_dead.OnMove();
		}
}
```

```
void Character::SetMovingDown(bool flag)
{
        isMovingDown = flag;
}
void Character::SetMovingLeft(bool flag)
{
        isMovingLeft = flag;
}
void Character::SetMovingRight(bool flag)
{
        isMovingRight = flag;
}
void Character::SetMovingUp(bool flag)
{
        isMovingUp = flag;
}
void Character::SetDead(bool flag)
{
        isDead = flag;
}
void Character::SetXY(int nx, int ny)
{
        x = nx; y = ny;
}
void Character::SetRange(int i) {
        Explosion_range = i;
}
void Character::OnShow() {
        Character_down.SetTopLeft(x, y);
        Character_up.SetTopLeft(x, y);
        Character_left.SetTopLeft(x, y);
        Character_right.SetTopLeft(x, y);
        Character_dead.SetTopLeft(x, y);
        if (Animate_State == 1)        Character_down.OnShow();
        else if (Animate_State == 2)Character_up.OnShow();
        else if (Animate_State == 3)Character_left.OnShow();
        else if (Animate_State == 4)Character_right.OnShow();
        else if (Animate_State == 5)Character_dead.OnShow();
}
void Character::LoadMap(int maps[13][15]) {
        for (int i = 0; i < 416; i++) {
                for (int j = 0; j < 480; j++) {
                        map[i][j] = maps[i / 32][j / 32];
                }
        }
        for (int i = 0; i < 13; i++) {
                for (int j = 0; j < 15; j++) {
                        map_simple[i][j] = maps[i][j];
                }
        }
}
int Character::GetPosition(int px, int py) {
        if (px < 0 || px > 480 || py < 0 || py > 416)
                return 2;
        return map[py][px];
}
bool Character::isMoveable(int x, int y) {
        if (x < 0 || y < 0 || x > 480 || y > 416)return false;
        if (map[y][x] == 1)return false;
        if (map[y][x] == 2)return false;
        if (map[y][x] == 4)return false;
        return true;
}
}
```

Enemy.h

```
#ifndef ENEMY
#define ENEMY
#include "Bullet.h"
namespace game_framework {
        class Enemy {
        private:
                CAnimation Character_down;                              // 腳色向下的動畫
                CAnimation Character_up;
                CAnimation Character_left;
                CAnimation Character_right;
                CAnimation Character_death;
                int    Animate_State;                                   // 腳色移動狀態 1 為下 2 為上 3 為左 4 為右
                int    x, y;                                            // 腳色左上角座標
                int    move_step = 2;                                   // 腳色步數
                int    upRange, downRange, leftRange, rightRange;       // 各方向可移動步數
                int    descision;                                       // 1 上 2 下 3 左 4 右
                bool isAlive;                                           // 存活與否 預設一滴血
                int    time;                                            // FOR random
                int    bg[13][15];
                int    GetPath();
                Bullet b;
                bool BulletHit;                                         // 子彈打到玩家
                int    DeathAnimateCount;
        public:
                Enemy();
                int    GetX1();                         // 腳色左上角 x 座標
                int    GetY1();                         // 腳色左上角 y 座標
                int    GetX2();                         // 腳色右下角 x 座標
                int    GetY2();                         // 腳色右下角 y 座標
                void Initialize(int, int);              // 設定腳色為初始值 對不同腳色設定初始位置
                void LoadBitmap();                      // 載入圖形
                void OnMove(int, int, int, int);        // 移動腳色
                void OnShow();                          // 將腳色圖形貼到畫面
                void LoadMap(int maps[13][15]);
                int    GetPosition(int, int);
                void Attack(int nx, int ny);
                int    BulletPosX();
                int    BulletPosY();
                void BulletTouch(int, int);
                bool BulletHitPlayer();                 // 攻擊判斷 碰到玩家返回 true 到 mygame
                bool Alive();
        };
}
#endif
```

Enemy.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include <cstdlib>              // 亂數相關函數
#include "Enemy.h"
#include "mygame.h"
namespace game_framework {
Enemy::Enemy() {
        Initialize(0,0);
}
void Enemy::Initialize(int nx, int ny) {
        x = nx;
        y = ny;
        time = 0;
        Animate_State = 1;
        descision = 0;
        upRange = downRange = leftRange = rightRange = 0;
        DeathAnimateCount = 0;
        isAlive = true;
        BulletHit = false;
}
void Enemy::LoadBitmap() {
        Character_down.SetDelayCount(5);
        Character_down.AddBitmap(IDB_EM_DW_1, RGB(255, 255, 255));
        Character_down.AddBitmap(IDB_EM_DW_2, RGB(255, 255, 255));
        Character_down.AddBitmap(IDB_EM_DW_3, RGB(255, 255, 255));
        Character_down.AddBitmap(IDB_EM_DW_4, RGB(255, 255, 255));
        Character_up.SetDelayCount(5);
        Character_up.AddBitmap(IDB_EM_UP_1, RGB(255,255, 255));
        Character_up.AddBitmap(IDB_EM_UP_2, RGB(255,255, 255));
        Character_up.AddBitmap(IDB_EM_UP_3, RGB(255,255, 255));
        Character_up.AddBitmap(IDB_EM_UP_4, RGB(255,255, 255));
        Character_left.SetDelayCount(5);
        Character_left.AddBitmap(IDB_EM_LE_1,RGB(255,255, 255));
        Character_left.AddBitmap(IDB_EM_LE_2,RGB(255,255, 255));
        Character_left.AddBitmap(IDB_EM_LE_3,RGB(255,255, 255));
        Character_left.AddBitmap(IDB_EM_LE_4,RGB(255,255, 255));
        Character_right.SetDelayCount(5);
        Character_right.AddBitmap(IDB_EM_RE_1, RGB(255, 255,255));
        Character_right.AddBitmap(IDB_EM_RE_2, RGB(255, 255,255));
        Character_right.AddBitmap(IDB_EM_RE_3, RGB(255, 255,255));
        Character_right.AddBitmap(IDB_EM_RE_4, RGB(255, 255,255));
        Character_death.SetDelayCount(1);
        Character_death.AddBitmap(IDB_EM_DE_1, RGB(255,255, 255));
        Character_death.AddBitmap(IDB_EM_DE_2, RGB(255,255, 255));
        b.LoadBitmap();
}
void Enemy::OnMove(int ax, int ay, int bx, int by) {
        if (isAlive) {
                time++;
                if (bg[(y - 32) / 32][(x - 128) / 32] == 5) {
                        isAlive = false;
                        time = -1;
                }
                if ((x - 128) % 32 == 0 && (y - 32) % 32 == 0) {
                        descision = GetPath();
                        if (!b.getActive())Attack(ax, ay);
                }
                if (descision == 1) {
                        y -= move_step;
                        Character_up.OnMove();
                }
                else if (descision == 2) {
                        y += move_step;
                        Character_down.OnMove();
```

```
			}
			else if (descision == 3) {
				x -= move_step;
				Character_left.OnMove();
			}
			else if (descision == 4) {
				x += move_step;
				Character_right.OnMove();
			}
			if (b.getActive())BulletTouch(ax, ay);
			b.OnMove();
		}
		else if (time == -1) {
			Character_death.OnMove();
		}
}
void Enemy::OnShow() {
		if (isAlive) {
			Character_down.SetTopLeft(x, y);
			Character_up.SetTopLeft(x, y);
			Character_left.SetTopLeft(x, y);
			Character_right.SetTopLeft(x, y);
			if (descision == 1)Character_up.OnShow();
			else if (descision == 4)Character_right.OnShow();
			else if (descision == 3)Character_left.OnShow();
			else Character_down.OnShow();
			b.OnShow();
		}
		else if (time == -1) {
			Character_death.SetTopLeft(x, y);
			Character_death.OnShow();
			if (Character_death.IsFinalBitmap()) {
				DeathAnimateCount++;
			}
			if (DeathAnimateCount == 3) {
				time = -2;
			}
		}
}
void Enemy::LoadMap(int maps[13][15]) {
		for (int i = 0; i < 13; i++) {
			for (int j = 0; j < 15; j++) {
				bg[i][j] = maps[i][j];
			}
		}
}
int	Enemy::GetPosition(int nx, int ny) {
		return bg[ny][nx];
}
int	Enemy::GetPath() {
		int nx = (x - 128) / 32;
		int ny = (y - 32) / 32;
		for (int i = 1;; i++) {
			if (ny - i < 0 || bg[ny - i][nx] == 1 || bg[ny - i][nx] == 2 || bg[ny - i][nx] == 4) {
				upRange = --i;
				break;
			}
		}
		for (int i = 1;; i++) {
			if (ny + i >= 12 || bg[ny + i][nx] == 1 || bg[ny + i][nx] == 2 || bg[ny + i][nx] == 4) {
				downRange = --i;
				break;
			}
		}
		for (int i = 1;; i++) {
			if (nx - i < 0 || bg[ny][nx - i] == 1 || bg[ny][nx - i] == 2 || bg[ny][nx - i] == 4) {
				leftRange = --i;
				break;
			}
		}
		for (int i = 1;; i++) {
			if (nx + i >= 14 || bg[ny][nx + i] == 1 || bg[ny][nx + i] == 2 || bg[ny][nx + i] == 4) {
				rightRange = --i;
				break;
			}
		}
		srand(time);
		int total = upRange + downRange + leftRange + rightRange;
		if (total == 0)return 0;
		int Rand = rand() % total;
		if (Rand < upRange && upRange != 0) {		//rand 剛好整除且 upRange 又為 0 AI 不能往上 *只有向上才會
有這種情況
			return 1;				// 向上
		}
		else if (Rand < upRange + downRange) {
			return 2;				// 向下
		}
		else if (Rand < upRange + downRange + leftRange) {
			return 3;				// 向左
		}
		else return 4;				// 向右
}
int Enemy::GetX1() {
		return x;
}
int Enemy::GetY1() {
		return y;
}
int Enemy::GetX2() {
		return x + Character_down.Width();
}
int Enemy::GetY2() {
		return y + Character_down.Width();
}
void Enemy::Attack(int nx, int ny) {
		nx = nx + 16;							// 腳色中心點
		ny = ny + 16;
		if (nx >= x && nx <= x + 32) {					// 上下判斷 先判斷 x 軸是否相同
			if (ny <= y && ny >= y - upRange * 32) {		// 用 ENEMY 的左上做判斷
				b.setPath(x + 16, y, 1);
			}
			else if (ny >= y + 32 && ny <= y + downRange * 32 + 32) {	// 用 ENEMY 的右下做判斷，不然腳色在
最下那格會判斷不到
				b.setPath(x + 16, y + 32, 2);
			}
		}
		else if (ny >= y && ny <= y + 32) {				// 左右判斷
			if (nx <= x && nx >= x - leftRange * 32) {
				b.setPath(x, y + 16, 3);
			}
			else if (nx >= x + 32 && nx <= x + rightRange * 32 + 32) {	// 用 ENEMY 的右下做判斷，不然腳色在最
右那格會判斷不到
				b.setPath(x + 32, y + 16, 4);
			}
		}
}
void Enemy::BulletTouch(int cx, int cy) {
		int bx = b.getX();
		int by = b.getY();
		int dir = b.getDir();
		if (!b.isTouched()) {
			if (bx >= cx && bx <= cx + 32 && by >= cy && by <= cy + 32) {	// 子彈打到玩家
				BulletHit = true;
				if (dir == 1) b.isTouched(bx, cy + 32);
				else if (dir == 2) b.isTouched(bx, cy);
				else if (dir == 3) b.isTouched(cx + 32, by);
				else if (dir == 4) b.isTouched(cx, by);
			}
		else if (bx <= 128 || bx >= 128 + 32 * 15) {
			int nx = 128 + 480 * (dir - 3);							// 向左(dir = 3)超出邊界在 x =
128 爆裂 向右(dir=4)設在 x = 128+480
			b.isTouched(nx, by);
		}
		else if (by <= 32 || by >= 32 + 32 * 13) {
			int ny = 32 + 416 * (dir - 1);							//向左(dir = 3)超出邊界在 x = 128
爆裂 向右(dir = 4)設在 x = 128+480
			b.isTouched(bx, ny);
		}
		else if (bg[(by - 32) / 32][(bx - 128) / 32] == 1 || bg[(by - 32) / 32][(bx - 128) / 32] == 2) {  // 子彈打到牆壁
或障礙
			int nx = (bx - 128) / 32;
			int ny = (by - 32) / 32;
			if (dir == 1) b.isTouched(bx, (ny + 1) * 32 + 31);				// 設定爆炸動畫位置
			else if (dir == 2) b.isTouched(bx, ny * 32 + 32);
			else if (dir == 3) b.isTouched((nx + 1) * 32 + 127, by);
			else if (dir == 4) b.isTouched(nx * 32 + 128, by);
		}
	}
	else {
		BulletHit = false;
	}
}
bool Enemy::BulletHitPlayer() {
	return BulletHit;
}
bool Enemy::Alive() {
	return isAlive;
}
}
```

Bomb.h

```
namespace game_framework {
	class Bomb {
	public:
		Bomb();
		void Initialize();
		void LoadBitmap();
		void OnMove();
		void OnShow();
		void setActive(bool);
		void setUp(int);
		void setDown(int);
		void setRight(int);
		void setLeft(int);
		void setRange(int, int, int, int);
		void setTopleft(int, int);
		void setObs(bool);
		int   getUp();
		int   getDown();
		int   getRight();
		int   getLeft();
		int   getTop_Bomb();
		int   getLeft_Bomb();
		bool getActive();
		bool getExp();
		bool getObs();
		bool getAud();
	private:
		CAnimation waiting;
		CAnimation Explosion;
		CAnimation Exp_V;
		CAnimation Exp_H;
		bool		active;			// 00 等待 10 未爆 11 爆炸中 01 結束
		bool		isExp;
		bool		Obs_break;		// 是否範圍設置
		bool		audio_played;		// 音效是否播放 以防重複執行撥放
		int		x, y;
		int		range_up;
		int		range_down;
		int		range_left;
		int		range_right;
		int		timer;			// 爆炸倒數
	};
}
```

Bomb.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Bomb.h"
namespace game_framework {
Bomb::Bomb() {
        x = 0;
        y = 0;
        active = false;
        isExp    = false;
}
void Bomb::Initialize() {
        x = 0;
        y = 0;
        active = false;
        isExp    = false;
        Obs_break = false;
        audio_played = false;
        timer = 0;
        Obs_break=range_up = range_down = range_left = range_right = 0;
        Explosion.Reset();
        Exp_H.Reset();
        Exp_V.Reset();
}
void Bomb::LoadBitmap() {
        waiting.AddBitmap(IDB_BOMB_1, RGB(255,255, 255));
        waiting.AddBitmap(IDB_BOMB_2, RGB(255,255, 255));
        Explosion.SetDelayCount(5);
        Explosion.AddBitmap(IDB_expC_4, RGB(255, 255, 255));
        Explosion.AddBitmap(IDB_expC_3, RGB(255, 255, 255));
        Explosion.AddBitmap(IDB_expC_2, RGB(255, 255, 255));
        Explosion.AddBitmap(IDB_expC_1, RGB(255, 255, 255));
        Exp_H.SetDelayCount(5);
        Exp_H.AddBitmap(IDB_expH_4, RGB(255,255, 255));
        Exp_H.AddBitmap(IDB_expH_3, RGB(255,255, 255));
        Exp_H.AddBitmap(IDB_expH_2, RGB(255,255, 255));
        Exp_H.AddBitmap(IDB_expH_1, RGB(255,255, 255));
        Exp_V.SetDelayCount(5);
        Exp_V.AddBitmap(IDB_expV_4, RGB(255, 255,255));
        Exp_V.AddBitmap(IDB_expV_3, RGB(255, 255,255));
        Exp_V.AddBitmap(IDB_expV_2, RGB(255, 255,255));
        Exp_V.AddBitmap(IDB_expV_1, RGB(255, 255,255));
}
void Bomb::OnMove() {
        if (active && !isExp) {
                waiting.OnMove();
        }
        else if (active && isExp) {
                Explosion.OnMove();
                Exp_H.OnMove();
                Exp_V.OnMove();
        }
}
void Bomb::OnShow() {
        if (active && !isExp) {
                timer++;
                waiting.SetTopLeft(x, y);
                waiting.OnShow();
                if (timer == 30 * 2) {                    // 30FPS * 2 秒
                        timer = 0;
                        isExp    = true;
                        audio_played = true;
                }
        }
        else if (active && isExp && Obs_break) {
```

```
                        Explosion.SetTopLeft(x, y);
                        Explosion.OnShow();
                        for (int i = 1; i <= range_left; i++) {
                                Exp_H.SetTopLeft(x - 32 * i, y);
                                Exp_H.OnShow();
                        }
                        for (int i = 1; i <= range_right; i++) {
                                Exp_H.SetTopLeft(x + 32 * i, y);
                                Exp_H.OnShow();
                        }
                        for (int i = 1; i <= range_up; i++) {
                                Exp_V.SetTopLeft(x, y - 32 * i);
                                Exp_V.OnShow();
                        }
                        for (int i = 1; i <= range_down; i++) {
                                Exp_V.SetTopLeft(x, y + 32 * i);
                                Exp_V.OnShow();
                        }
                        if (Explosion.IsFinalBitmap()) {
                                active = false;
                        }
                        audio_played = false;
                }
}
void Bomb::setActive(bool act) {
        if (act)isExp = 0;
        active = act;
}
void Bomb::setUp(int up) {
        range_up = up;
        if (range_up < 0)range_up = 0;
}
void Bomb::setDown(int dw) {
        range_down = dw;
        if (range_down < 0)range_down = 0;
}
        void Bomb::setRight(int rt) {
        range_right = rt;
        if (range_right < 0)range_right = 0;
}
void Bomb::setLeft(int le) {
        range_left = le;
        if (range_left < 0)range_left = 0;
}
void Bomb::setRange(int up, int dw, int le, int rt) {
        range_up = up;
        range_down = dw;
        range_left = le;
        range_right = rt;
}
void Bomb::setObs(bool o) {
        Obs_break = o;
}
int     Bomb::getUp() {
        return range_up;
}
int     Bomb::getDown() {
        return range_down;
}
int     Bomb::getRight() {
        return range_right;
}
int     Bomb::getLeft() {
        return range_left;
}
bool    Bomb::getActive() {
        return active;
}
bool    Bomb::getExp() {
        return isExp;
}
int   Bomb::getTop_Bomb() {
        return x;
}
int   Bomb::getLeft_Bomb() {
        return y;
}
void Bomb::setTopleft(int nx, int ny) {
        x = nx;
        y = ny;
}
bool Bomb::getObs() {
        return Obs_break;
}
bool Bomb::getAud() {
        return audio_played;
}
}
```

Healths.h

```
namespace game_framework {
	class Healths {
	public:
		void Initialize();
		void Initialize(int, int);
		void LoadBitmap();
		void OnMove();
		void OnShow();
		void SetDescision(int);
		void setActive();
		void setTopLeft(int, int);
		bool getActive();
		bool getExp();
	private:
		CAnimation heart2;
		CAnimation heart1;
		CAnimation heart0;
		bool Active, exp;		// 00 等待 11 旋轉 10 結束
		int x, y;
		int descision;			// 0 沒血 1 半血 2 滿血
	};
}
```

Healths.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Healths.h"
namespace game_framework {
void Healths::Initialize() {
	Initialize(0, 0);
}
void Healths::Initialize(int nx, int ny) {
	Active = exp = false;
	x = nx;
	y = ny;
	descision = 3;
}
void Healths::LoadBitmap() {
	heart2.SetDelayCount(5);
	heart2.AddBitmap(IDB_HEALTH_2, RGB(255, 255, 255));
	heart1.SetDelayCount(5);
	heart1.AddBitmap(IDB_HEALTH_1, RGB(255, 255, 255));
	heart0.SetDelayCount(5);
	heart0.AddBitmap(IDB_HEALTH_0, RGB(255, 255, 255));
}
void Healths::OnMove() {
	heart2.OnMove();
	heart1.OnMove();
	heart0.OnMove();
}
void Healths::OnShow() {
	heart2.SetTopLeft(x, y);
	heart1.SetTopLeft(x, y);
	heart0.SetTopLeft(x, y);
	if (descision == 2) heart2.OnShow();
	else if (descision == 1) heart1.OnShow();
	else if (descision == 0) heart0.OnShow();
}
void Healths::SetDescision(int des) {
	descision = des;
}
void Healths::setTopLeft(int nx, int ny) {
	x = nx;
	y = ny;
}
}
```

CoinsAnimation.h

```
namespace game_framework {
	class CoinsAnimation {
	public:
		void Initialize();
		void Initialize(int, int);
		void LoadBitmap();
		void OnMove();
		void OnShow();
		void setActive();
		void setTopLeft(int, int);
		bool getActive();
		bool getExp();
	private:
		CAnimation glod;
		bool Active, exp;		// 00 等待 11 旋轉 10 結束
		int x, y;
	};
}
```

CoinsAnimation.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "CoinsAnimation.h"
namespace game_framework {
void CoinsAnimation::Initialize() {
	glod.Reset();
	Active = exp = false;
	x = y = 0;
}
void CoinsAnimation::Initialize(int nx, int ny) {
	glod.Reset();
	Active = exp = false;
	x = nx;
	y = ny;
}
void CoinsAnimation::LoadBitmap() {
	glod.SetDelayCount(5);
	glod.AddBitmap(IDB_COIN_0, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_1, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_2, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_3, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_4, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_5, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_6, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_7, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_8, RGB(255, 255, 255));
	glod.AddBitmap(IDB_COIN_9, RGB(255, 255, 255));
}
void CoinsAnimation::OnMove() {
	if (Active && exp) {
		glod.OnMove();
		if (glod.IsFinalBitmap()) {
			exp = false;
			// Active = false;
		}
	}
}
void CoinsAnimation::OnShow() {
	if ((Active && exp) || (!Active && !exp)) {
		glod.SetTopLeft(x, y);
		glod.OnShow();
	}
}
void CoinsAnimation::setActive() {
	Active = true;
	exp = true;
}
void CoinsAnimation::setTopLeft(int nx, int ny) {
	x = nx;
	y = ny;
}
bool CoinsAnimation::getActive() {
	return Active;
}
bool CoinsAnimation::getExp() {
	return exp;
}
}
```

Bullet.h

```
namespace game_framework {
	class Bullet {
	private:
		CAnimation bullet;
		int	x, y;
		int	direction;			// 1 上 2 下 3 左 4 右
		int	speed = 8;
		bool active;
		bool touched;
	public:
		Bullet();
		void setPath(int, int, int);
		void setActive(bool);
		bool getActive();			// 20210516 - None used at this time
		int	getX();				//  中心點
		int	getY();
		int	getDir();
		void isTouched(int, int);
		bool isTouched();
		void Initialize(int, int);		// 設定腳色為初始值 對不同腳色設定初始位置
		void LoadBitmap();			// 載入圖形
		void OnMove();			// 移動腳色
		void OnShow();			// 將腳色圖形貼到畫面
	};
}
```

Bullet.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Bullet.h"
namespace game_framework {
Bullet::Bullet() {
	Initialize(0,0);
}
void Bullet::Initialize(int nx, int ny) {
	x = nx;
	y = ny;
	direction = 0;
	active = touched =false;
}
void Bullet::LoadBitmap() {
	bullet.SetDelayCount(5);
	bullet.AddBitmap(IDB_BULLET_1,RGB(255,255,255));
	bullet.AddBitmap(IDB_BULLET_2,RGB(255,255,255));
	bullet.AddBitmap(IDB_BULLET_3,RGB(255,255,255));
	bullet.AddBitmap(IDB_BULLET_4,RGB(255,255,255));
	bullet.AddBitmap(IDB_BULLET_5,RGB(255,255,255));
}
void Bullet::OnMove() {
	if (!touched && direction == 1) {
		y -= speed;
	}
	else if (!touched && direction == 2) {
		y += speed;
	}
	else if (!touched && direction == 3) {
		x -= speed;
	}
	else if (!touched && direction == 4) {
		x += speed;
	}
	else if (touched) {
		bullet.OnMove();
		if (bullet.IsFinalBitmap()) {
			active = false;
			Initialize(0,0);
			bullet.Reset();
		}
	}
}
void Bullet::OnShow() {
	if (active) {
		int bitmapNumber = bullet.GetCurrentBitmapNumber();
		if (bitmapNumber == 0)bullet.SetTopLeft(x - 3, y - 3);		// 中心點轉換成左上座標
		else if (bitmapNumber == 1)bullet.SetTopLeft(x - 4, y - 4);
		else if (bitmapNumber == 2)bullet.SetTopLeft(x - 5, y - 6);
		else if (bitmapNumber == 3)bullet.SetTopLeft(x - 8, y - 8);
		else if (bitmapNumber == 4)bullet.SetTopLeft(x - 11, y - 12);
		bullet.OnShow();
	}
}
void Bullet::setPath(int nx, int ny, int dir) {
	x = nx;
	y = ny;
	direction = dir;
	active = true;
}
void Bullet::setActive(bool a) {
	active = a;
}
bool Bullet::getActive() {
        return active;
}
int    Bullet::getX() {
        return x + 3;
}
int    Bullet::getY() {
        return y + 3;
}
int    Bullet::getDir() {
        return direction;
}
void Bullet::isTouched(int nx, int ny) {
        x = nx;
        y = ny;
        touched = true;
}
bool Bullet::isTouched() {
        return touched;
}
}
```

Obstacle.h

```
namespace game_framework {
	class Obstacle {
	public:
		void Initialize();
		void Initialize(int, int);
		void LoadBitmap();
		void OnMove();
		void OnShow();
		void setActive();
		void setActive(bool);
		void setExp(bool);
		void setTopLeft(int, int);
		bool getActive();
		bool getExp();
	private:
		CAnimation block;
		bool Active, exp;		// 00 等待 11 爆裂動畫 10 結束
		int x, y;
	};
}
```

Obstacle.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Obstacle.h"
namespace game_framework {
void Obstacle::Initialize() {
	block.Reset();
	Active = exp = false;
	x = y = 0;
}
void Obstacle::Initialize(int nx, int ny) {
	block.Reset();
	Active = exp = false;
	x = nx;
	y = ny;
}
void Obstacle::LoadBitmap() {
	block.SetDelayCount(5);
	block.AddBitmap(IDB_BREAK_0, RGB(255, 255, 255));
	block.AddBitmap(IDB_BREAK_1, RGB(255, 255, 255));
	block.AddBitmap(IDB_BREAK_2, RGB(255, 255, 255));
	block.AddBitmap(IDB_BREAK_3, RGB(255, 255, 255));
}
void Obstacle::OnMove() {
	if (Active && exp) {
		block.OnMove();
		if (block.IsFinalBitmap()) {
			exp = false;
		}
	}
}
void Obstacle::OnShow() {
	if ((Active && exp) || (!Active && !exp)) {
		block.SetTopLeft(x, y);
		block.OnShow();
	}
}
void Obstacle::setActive() {
	Active = true;
	exp = true;
}
void Obstacle::setActive(bool a) {
	Active = a;
}
void Obstacle::setExp(bool e) {
	exp = e;
}
void Obstacle::setTopLeft(int nx, int ny) {
	x = nx;
	y = ny;
}
bool Obstacle::getActive() {
	return Active;
}
bool Obstacle::getExp() {
	return exp;
}
}
```

GameStage_2.h

```
#ifndef _STAGE2_
#define _STAGE2_
namespace game_framework {
	class GameStage_2 : public CGameState {
	public:
		GameStage_2(CGame* g);
		~GameStage_2();
		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);             // 鍵盤動作
		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
		void setBomb(int);
		void mapChange(int, int, int);                // 地圖變動&通知 character
		void BombState();
		void setBombRange(int, int, int, int);        // 爆炸時設置範圍
		void GetCoins();                              // 偵測碰撞金幣
		void HealthState();
	private:
		CMovingBitmap level;
		int bg[13][15];                               // 0 地板 1 石塊 2 粉色石 4 未爆彈 5 爆炸中
		int coins_pos[9][2];                          // 硬幣位置
		CMovingBitmap block_0;
		CMovingBitmap block_1;
		Obstacle*       block_2;
		int             block_2_pos[88][2],block2_num;
		CMovingBitmap panel;
		CMovingBitmap border;
		Character       character_1;                  // Range undone
		CMovingBitmap character_2;                    // 類別之後改
		int Enemy1_num;                               // 敵人 1 的數量
		int Enemy2_num;                               // 敵人 2 的數量
		int Enemy3_num;                               // 敵人 3 的數量
		int Enemy4_num;                               // 敵人 4 的數量
		Enemy* AI;
		int coins_num;                                // 金幣總數
		int sc;                                       // 紀錄吃到幾個金幣
		CoinsAnimation* coin_Ani;
		Bomb* Bomb_ch1;
		CMovingBitmap playerhead_1;
		CMovingBitmap playerhead_2;
		int life;
		int heart_num[8];
		int blood_ori, blood_vol;
		Healths* heart;
		bool taking_Damage;
		int k = 0;
		CInteger count_down;
		int timer;
		int score;
	};
}
#endif
```

GameStage_2.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Character.h"
#include "Bomb.h"
#include "Obstacle.h"
#include "Enemy.h"
#include "CoinsAnimation.h"
#include "Healths.h"
#include <iostream>
#include "GameStage_2.h"
namespace game_framework {
GameStage_2::GameStage_2(CGame* g) : CGameState(g)
{
	block2_num = 88;
	coins_num = 9;
	sc = 0;
	Bomb_ch1 = new Bomb[7];
	block_2 = new Obstacle[88];
	coin_Ani = new CoinsAnimation[9];
	heart = new Healths[8];
	AI = new Enemy[4];
}
GameStage_2::~GameStage_2() {
	delete[] Bomb_ch1;
	delete[] block_2;
	delete[] coin_Ani;
	delete[] heart;
	delete[] AI;
}
void GameStage_2::OnBeginState() {
	for (int i = 0; i < 7; i++) {
		Bomb_ch1[i].Initialize();
	}
	int obstacle_reset[88][2] = {
		{0,3},{0,5},{0,6},{0,8},{0,9},{0,12},{0,13},{0,14},
		{1,2},{1,4},{1,6},{1,8},{1,10},{1,12},{1,14},
		{2,2},{2,3},{2,4},{2,5},{2,9},{2,10},{2,11},{2,12},{2,14},
		{3,0},{3,6},{3,8},
		{4,0},{4,2},{4,3},{4,4},{4,10},{4,11},{4,12},{4,13},{4,14},
		{5,0},{5,2},{5,6},{5,8},{5,14},
		{6,0},{6,1},{6,2},{6,3},{6,4},{6,11},{6,12},{6,14},
		{7,6},{7,8},{7,12},
		{8,0},{8,1},{8,2},{8,4},{8,10},{8,11},{8,12},{8,14},
		{9,2},{9,6},{9,8},{9,12},{9,14},
		{10,0},{10,2},{10,3},{10,4},{10,5},{10,9},{10,10},{10,11},
		{11,0},{11,4},{11,6},{11,8},{11,10},{11,12},
		{12,0},{12,1},{12,3},{12,5},{12,6},{12,8},{12,9},{12,10},{12,12}
	};
	for (int i = 0; i < 13; i++) {
		for (int j = 0; j < 15; j++) {
			if (i % 2 == 1 && j % 2 == 1)bg[i][j] = 1;
			else bg[i][j] = 0;
		}
	}
	for (int i = 0; i < block2_num; i++) {
		block_2_pos[i][0] = obstacle_reset[i][0];
		block_2_pos[i][1] = obstacle_reset[i][1];
		bg[block_2_pos[i][0]][block_2_pos[i][1]] = 2;
		block_2[i].Initialize(block_2_pos[i][1] * 32 + 128, block_2_pos[i][0] * 32 + 32);
	}
	coins_num = 9;
	int coins_reset[9][2] = {
		{0,7},{3,4},{3,10},{6,5},{6,7},{6,9},{9,4},{9,10},{12,7}
	};
```

```
        for (int i = 0; i < coins_num; i++) {
                coins_pos[i][0] = coins_reset[i][0];
                coins_pos[i][1] = coins_reset[i][1];
                coin_Ani[i].Initialize(coins_pos[i][1] * 32 + 128, coins_pos[i][0] * 32 + 32);
        }
        int data[3];
        game->loadData(data, 3);
        score = data[0];
        life = data[1];
        int blood_reset = blood_ori = blood_vol = data[2];            // 預設血量總值為第一關傳遞過來的數值
        for (int i = 0; i < 8; i++) {
                if (blood_reset >= 2) {
                        heart_num[i] = 2;
                        blood_reset -= 2;
                }
                else if (blood_reset == 1) {
                        heart_num[i] = 1;
                        blood_reset = 0;
                }
                else heart_num[i] = 0;
        }
        // 腳色數值重置
        character_1.Initialize(128, 32);
        character_1.LoadMap(bg);
        AI[0].Initialize(7 * 32 + 128, 2 * 32 + 32);
        AI[1].Initialize(7 * 32 + 128, 4 * 32 + 32);
        AI[2].Initialize(7 * 32 + 128, 8 * 32 + 32);
        AI[3].Initialize(7 * 32 + 128, 10 * 32 + 32);
        for (int i = 0; i < 4; i++) {
                AI[i].LoadMap(bg);
        }
        timer = 0;
        CAudio::Instance()->Play(AUDIO_BGM2, true);
}
void GameStage_2::OnInit() {
        timer = 250;
        level.LoadBitmap(IDB_LEVEL_2);
        block_0.LoadBitmap(IDB_Bg_1, RGB(255, 255, 255));
        block_1.LoadBitmap(IDB_Blocks, RGB(255, 255, 255));
        for (int i = 0; i < block2_num; i++) {
                block_2[i].LoadBitmap();
        }
        border.LoadBitmap(IDB_BORDER_0, RGB(255, 255, 255));
        panel.LoadBitmap(IDB_Panel, RGB(255, 255, 255));
        character_1.LoadBitmap();
        for (int i = 0; i < 7; i++) {
                Bomb_ch1[i].LoadBitmap();
        }
        for (int i = 0; i < coins_num; i++) {
                coin_Ani[i].LoadBitmap();
        }
        for (int i = 0; i < 4; i++) {
                AI[i].LoadBitmap();
        }
        playerhead_1.LoadBitmap(IDB_PLAYERHEAD1, RGB(255, 255, 255));
        playerhead_2.LoadBitmap(IDB_PLAYERHEAD2, RGB(255, 255, 255));
        for (int i = 0; i < 8; i++) {
                heart[i].LoadBitmap();
        }
        count_down.SetInteger(60);
        CAudio::Instance()->Load(AUDIO_BGM2, "sounds\\stage2BGM.mp3");
}
void GameStage_2::OnMove() {
        timer++;
        int second = timer / 30;
        // int min = second / 60;
        second %= 60;
        if (!(timer % 30))
                count_down.Add(-1);
        bool nextState=false;                    // 下一關 為 0 就進下一關
        for (int i = 0; i < 4; i++) {
                nextState = nextState | AI[i].Alive();
        }
        if (!nextState) {
                CAudio::Instance()->Stop(AUDIO_BGM2);
                int data[1] = {score};
                game->saveData(data, 1);
                GotoGameState(GAME_STATE_OVER);
        }
        for (int i = 0; i < block2_num; i++) {
                block_2[i].OnMove();
                if (block_2[i].getActive() && !block_2[i].getExp()) {
                        mapChange(block_2_pos[i][1], block_2_pos[i][0], 0);
                        block_2[i].setActive(false);
                        block_2[i].setExp(true);
                }
        }
        BombState();
        character_1.OnMove();
        for (int i = 0; i < 4; i++) {
                AI[i].OnMove(character_1.GetX1(), character_1.GetY1(), 0, 0);
        }
        GetCoins();
        HealthState();
        if (blood_vol > 0) {
                character_1.SetDead(false);
        }
        if (blood_vol == 0 && life != 1) {
                character_1.SetDead(true);
                life--;
                int health_reset[8] = { 2, 2, 2, 2, 2, 2, 2, 2 };
                for (int i = 0; i < 8; i++) {
                        heart_num[i] = health_reset[i];
                }
                blood_ori= blood_vol = 16;           // 預設血量總值為 16
        }
        else if (blood_vol == 0 && life == 1) {
                life--;
                CAudio::Instance()->Stop(AUDIO_BGM2);
                int data[1] = { score };
                game->saveData(data, 1);
                GotoGameState(GAME_STATE_OVER);
        }
        // 判斷敵人被殺死後給予得分
        if (!(AI[0].Alive()) && Enemy1_num > 0) {
                score += 100;
                Enemy1_num--;
        }
        if (!(AI[1].Alive()) && Enemy2_num > 0) {
                score += 100;
                Enemy2_num--;
        }
}
void GameStage_2::OnShow() {                              // 越後放的顯示會越上層
        panel.SetTopLeft(0, 0);
        panel.ShowBitmap();
        border.SetTopLeft(96, 0);
        border.ShowBitmap();
        level.SetTopLeft(609, 0);
        level.ShowBitmap();
        for (int i = 0; i < 13; i++) {                    // 方塊顯示    j 是 X 軸 i 是 Y 軸
                for (int j = 0; j < 15; j++) {
                        if (bg[i][j] == 1) {
                                block_1.SetTopLeft(128 + 32 * j, 32 * (i + 1));
                                block_1.ShowBitmap();
                        }
                        else {
                                block_0.SetTopLeft(128 + 32 * j, 32 * (i + 1));
```

```
                              block_0.ShowBitmap();
                        }
                  }
            }
            for (int i = 0; i < block2_num; i++) {
                  block_2[i].OnShow();
            }
            for (int i = 0; i < coins_num; i++) {
                  coin_Ani[i].setTopLeft(128 + coins_pos[i][1] * 32, 32 * (coins_pos[i][0] + 1));
                  coin_Ani[i].OnShow();
            }
            count_down.SetTopLeft(panel.Width() * 25 / 100, panel.Height() * 48 / 100);
            count_down.LoadBitmap();
            count_down.ShowBitmap();
            character_1.OnShow();
            for (int i = 0; i < 7; i++) {
                  Bomb_ch1[i].OnShow();
            }
            for (int i = 0; i < 4; i++) {
                  AI[i].OnShow();
            }
            playerhead_1.SetTopLeft((panel.Width() * 16 / 100), panel.Height() * 13 / 100);
            playerhead_1.ShowBitmap();
            CDC* pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
            CFont f, * fp;
            f.CreatePointFont(160, "Times New Roman");          // 產生 font f; 160 表示 16point 的字
            fp = pDC->SelectObject(&f);                         // 選用 font f
            pDC->SetBkMode(TRANSPARENT);
            pDC->SetBkColor(RGB(0, 0, 255));
            pDC->SetTextColor(RGB(255, 255, 255));
            char str[80];                                       // Demo 數字對字串的轉換
            sprintf(str, "X %d", life);
            pDC->TextOut((panel.Width() * 59 / 100), panel.Height() * 13 / 100, str);
            pDC->SelectObject(fp);                              // 放掉 font f (千萬不要漏了放掉)
            CDDraw::ReleaseBackCDC();                           // 放掉 Back Plain 的 CDC
            CDC* pDC1 = CDDraw::GetBackCDC();                   // 取得 Back Plain 的 CDC
            CFont f1, * fp1;
            f1.CreatePointFont(140, "Times New Roman");         // 產生 font f; 160 表示 16point 的字
            fp1 = pDC1->SelectObject(&f1);                      // 選用 font f
            pDC1->SetBkMode(TRANSPARENT);
            pDC1->SetBkColor(RGB(0, 0, 255));
            pDC1->SetTextColor(RGB(255, 255, 255));
            pDC1->TextOut((panel.Width() * 16 / 100), panel.Height() * 375 / 1000, "SCORE");
            char str1[80];                                      // Demo 數字對字串的轉換
            sprintf(str1, "%06d", score);
            pDC1->TextOut((panel.Width() * 20 / 100), panel.Height() * 41 / 100, str1);
            pDC1->SelectObject(fp1);                            // 放掉 font f (千萬不要漏了放掉)
            CDDraw::ReleaseBackCDC();                           // 放掉 Back Plain 的 CDC
            playerhead_2.SetTopLeft((panel.Width() * 16 / 100), panel.Height() * 56 / 100);
            playerhead_2.ShowBitmap();
            for (int i = 0; i < 2; i++) {
                  for (int j = 0; j < 8; j++) {
                        if (i == 0 && j < 4) {
                              heart[j].setTopLeft((panel.Width() * 15 / 100) + 17 * j, panel.Height() * 19 / 100);
                              heart[j].OnShow();
                        }
                        else if (i == 1 && j > 3 && j < 8) {
                              heart[j].setTopLeft((panel.Width() * 15 / 100) + 17 * (j - 4), panel.Height() * 225 / 1000);
                              heart[j].OnShow();
                        }
                  }
            }
}
void GameStage_2::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_ESC = 0x1B;
        const char KEY_P = 0x50;
        const char KEY_SPACE = 0x20;
        const char KET_Y = 0x59;
        if (nChar == KEY_ESC || nChar == KEY_P) {
                game_framework::CGame::Instance()->OnFilePause();
                game_framework::CGame::Instance()->SaveState(this);
                GotoGameState(GAME_STATE_PAUSE);
        }
        if (nChar == KEY_LEFT) {
                character_1.SetMovingLeft(true);
        }
        if (nChar == KEY_RIGHT) {
                character_1.SetMovingRight(true);
        }
        if (nChar == KEY_UP) {
                character_1.SetMovingUp(true);
        }
        if (nChar == KEY_DOWN) {
                character_1.SetMovingDown(true);
        }
        if (nChar == KEY_SPACE) {
                setBomb(1);
        }
        if (nChar == KET_Y) {
                CAudio::Instance()->Stop(AUDIO_BGM2);
                int data[2] = { score};
                game->saveData(data, 1);
                GotoGameState(GAME_STATE_OVER);
        }
}
void GameStage_2::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        const char KEY_LEFT = 0x25;     // keyboard 左箭頭
        const char KEY_UP = 0x26;       // keyboard 上箭頭
        const char KEY_RIGHT = 0x27;    // keyboard 右箭頭
        const char KEY_DOWN = 0x28;     // keyboard 下箭頭
        if (nChar == KEY_LEFT)
                character_1.SetMovingLeft(false);
        if (nChar == KEY_RIGHT)
                character_1.SetMovingRight(false);
        if (nChar == KEY_UP)
                character_1.SetMovingUp(false);
        if (nChar == KEY_DOWN)
                character_1.SetMovingDown(false);
}
void GameStage_2::setBomb(int id) {
        if (id == 1) {                                                          //[y][x]
                int x = (character_1.GetX1() + character_1.GetX2()) / 2;        //腳色中心點
                int y = (character_1.GetY1() + character_1.GetY2()) / 2;        //腳色中心點
                x = (x - 128) / 32;                                             //轉換成 13*15 地圖模式
                y = (y - 32) / 32;
                if (bg[y][x] == 0) {
                        for (int i = 0; i < 7; i++) {
                                if (!Bomb_ch1[i].getActive()) {
                                        Bomb_ch1[i].setTopleft(x * 32 + 128, (y + 1) * 32);
                                        Bomb_ch1[i].setActive(true);
                                        mapChange(x, y, 4);
                                        break;
                                }
                        }
                }
                else {
                }
        }
        else if (id == 2) {
                // player2's operating
        }
}
```

```
void GameStage_2::mapChange(int x, int y, int value) {
	bg[y][x] = value;
	character_1.LoadMap(bg);
	for (int i = 0; i < 4; i++) {
		AI[i].LoadMap(bg);
	}
}
void GameStage_2::BombState() {
	for (int i = 0; i < 7; i++) {
		Bomb_ch1[i].OnMove();
		if (Bomb_ch1[i].getActive() && !Bomb_ch1[i].getExp()) {
			int nx = (Bomb_ch1[i].getTop_Bomb() - 128) / 32;
			int ny = (Bomb_ch1[i].getLeft_Bomb() - 32) / 32;
			if (bg[ny][nx] != 4)mapChange(nx, ny, 4);
		}
		if (Bomb_ch1[i].getActive() && Bomb_ch1[i].getExp()) {  // 爆炸中的炸彈位置重設成可行走
			int nx = (Bomb_ch1[i].getTop_Bomb() - 128) / 32;
			int ny = (Bomb_ch1[i].getLeft_Bomb() - 32) / 32;
			mapChange(nx, ny, 5);
			if (!Bomb_ch1[i].getObs())setBombRange(1, i, nx, ny);
			if (Bomb_ch1[i].getAud())CAudio::Instance()->Play(AUDIO_BOMB, false);
			for (int j = 1; j <= Bomb_ch1[i].getUp(); j++)mapChange(nx, ny - j, 5);
			for (int j = 1; j <= Bomb_ch1[i].getDown(); j++)mapChange(nx, ny + j, 5);
			for (int j = 1; j <= Bomb_ch1[i].getRight(); j++)mapChange(nx + j, ny, 5);
			for (int j = 1; j <= Bomb_ch1[i].getLeft(); j++)mapChange(nx - j, ny, 5);
		}
		if (!Bomb_ch1[i].getActive() && Bomb_ch1[i].getExp()) {
			int nx = (Bomb_ch1[i].getTop_Bomb() - 128) / 32;
			int ny = (Bomb_ch1[i].getLeft_Bomb() - 32) / 32;
			mapChange(nx, ny, 0);
			for (int j = 1; j <= Bomb_ch1[i].getUp(); j++)mapChange(nx, ny - j, 0);
			for (int j = 1; j <= Bomb_ch1[i].getDown(); j++)mapChange(nx, ny + j, 0);
			for (int j = 1; j <= Bomb_ch1[i].getRight(); j++)mapChange(nx + j, ny, 0);
			for (int j = 1; j <= Bomb_ch1[i].getLeft(); j++)mapChange(nx - j, ny, 0);
			Bomb_ch1[i].Initialize();
		}
	}
}
void GameStage_2::setBombRange(int id, int i, int x, int y) {
	if (id == 1) {
		int j;
		int range = character_1.GetRange();
		for (j = 1; j <= range; j++) {
			if (y - j < 0 || bg[y - j][x] == 1) {
				break;
			}
			else if (bg[y - j][x] == 2) {
				for (int k = 0; k < block2_num; k++) {
					if (block_2_pos[k][0] == y - j && block_2_pos[k][1] == x) {
						block_2[k].setActive();
						score += 10;
						break;
					}
				}
				break;
			}
		}
		Bomb_ch1[i].setUp(--j);
		for (j = 1; j <= range; j++) {
			if (y + j > 12 || bg[y + j][x] == 1) {
				break;
			}
			else if (bg[y + j][x] == 2) {
				for (int k = 0; k < block2_num; k++) {
				if (block_2_pos[k][0] == y + j && block_2_pos[k][1] == x) {
						block_2[k].setActive();
						score += 10;
						break;
					}
```

```
                        }
                        break;
                    }
                }
                Bomb_ch1[i].setDown(--j);
                for (j = 1; j <= range; j++) {
                    if (x + j > 14 || bg[y][x + j] == 1) {
                        break;
                    }
                    else if (bg[y][x + j] == 2) {
                        for (int k = 0; k < block2_num; k++) {
                            if (block_2_pos[k][0] == y && block_2_pos[k][1] == x + j) {
                                block_2[k].setActive();
                                score += 10;
                                break;
                            }
                        }
                        break;
                    }
                }
                Bomb_ch1[i].setRight(--j);
                for (j = 1; j <= range; j++) {
                    if (x - j < 0 || bg[y][x - j] == 1) {
                        break;
                    }
                    else if (bg[y][x - j] == 2) {
                        for (int k = 0; k < block2_num; k++) {
                            if (block_2_pos[k][0] == y && block_2_pos[k][1] == x - j) {
                                block_2[k].setActive();
                                score += 10;
                                break;
                            }
                        }
                        break;
                    }
                }
                Bomb_ch1[i].setLeft(--j);
                Bomb_ch1[i].setObs(true);
        }
}
void GameStage_2::GetCoins() {
        //找出腳色所在位置(x,y)
        int x = (character_1.GetX1() + character_1.GetX2()) / 2;        // 腳色中心點
        int y = (character_1.GetY1() + character_1.GetY2()) / 2;        // 腳色中心點
        x = (x - 128) / 32;                                              // 轉換成 13*15 地圖模式
        y = (y - 32) / 32;
        for (int i = 0; i < coins_num; i++) {
                coin_Ani[i].OnMove();
                if (x == coins_pos[i][1] && y == coins_pos[i][0] && !coin_Ani[i].getActive() && !coin_Ani[i].getExp()) {
                        coin_Ani[i].setActive();
                        /*吃掉的硬幣+1*/
                        sc++;
                }
        }
}
void GameStage_2::HealthState() {
        for (int i = 0; i < 8; i++) {
                heart[i].OnMove();
                if (heart_num[i] == 2) {
                        heart[i].SetDescision(2);
                }
                else if (heart_num[i] == 1) {
                        heart[i].SetDescision(1);
                }
                else if (heart_num[i] == 0) {
                        heart[i].SetDescision(0);
                }
        }
        int x = (character_1.GetX1() + character_1.GetX2()) / 2;        // 腳色中心點
        int y = (character_1.GetY1() + character_1.GetY2()) / 2;            // 腳色中心點
        x = (x - 128) / 32;                                                  // 轉換成 13*15 地圖模式
        y = (y - 32) / 32;
        // TRACE(" %d\n", character_1.GetDead());
        if (!character_1.GetDead()) {
                if (taking_Damage) {
                        // wait two seconds
                        k++;
                        if (k == 60) {
                                taking_Damage = false;
                                k = 0;
                        }
                }
                else {
                        if (bg[y][x] == 5) {
                                CAudio::Instance()->Play(AUDIO_OOF, false);
                                blood_vol = blood_vol - 1;
                                taking_Damage = true;
                                TRACE("%d %d %d\n", life, blood_vol, blood_ori);
                        }
                        for (int i = 0; i < 4; i++) {
                                int x1 = (AI[i].GetX1() + AI[i].GetX2()) / 2;        // 敵人中心點
                                int y1 = (AI[i].GetY1() + AI[i].GetY2()) / 2;        // 敵人中心點
                                x1 = (x1 - 128) / 32;                                // 轉換成 13*15 地圖模式
                                y1 = (y1 - 32) / 32;
                                if (x == x1 && y == y1 && AI[i].Alive()) {
                                        CAudio::Instance()->Play(AUDIO_OOF, false);
                                        blood_vol = blood_vol - 1;
                                        taking_Damage = true;
                                        TRACE("%d %d %d\n", life, blood_vol, blood_ori);
                                }
                                if (AI[i].BulletHitPlayer() && AI[i].Alive()) {
                                        CAudio::Instance()->Play(AUDIO_OOF, false);
                                        blood_vol = blood_vol - 1;
                                        taking_Damage = true;
                                        TRACE("%d %d %d\n", life, blood_vol, blood_ori);
                                }
                        }
                }
        }
        double value = std::fmod(blood_ori, blood_vol);
        for (int i = 7; i >= 0; i--) {
                if (heart_num[i] != 0) {
                        if (heart_num[i] - value < 0) {
                                value -= heart_num[i];
                                heart_num[i] = 0;
                                blood_ori = blood_vol;
                        }
                        else if (heart_num[i] - value == 1) {
                                heart_num[i] = 1;
                                blood_ori = blood_vol;
                                value = 0;
                        }
                        else if (heart_num[i] - value == 0) {
                                heart_num[i] = 0;
                                blood_ori = blood_vol;
                                value = 0;
                        }
                }
        }
}
}
```