物件導向程式設計
OBJECT ORIENTED PROGRAMMING
書面報告

組　　別：第 16 組

題　　目：卡牌遊戲

組　　員：107590034　　姓名鄧鎧晨

107590035　　姓名許哲維

指導老師：陳偉凱老師

章節

簡介

1.動機

 現在的遊戲有上百種，像是 RPG、RTS、FPS……等，而其中包含了卡牌遊戲，我們認為卡牌遊戲在整體的時間上與其他遊戲比起相對的較短，由於現在常常會有很多瑣碎事情,因此這種短時間的遊戲是我們比較想要製作的。

而製作遊戲，也是我們團隊一直以來想要做的事情，但是由於我們資質並不高，無法做出很完美的遊戲，因此也選擇了較不困難的遊戲來製作，卡牌遊戲也是我們私底下較常玩的遊戲,由於可以在遊玩過程中做些其他簡單的事情，因此深受我們的喜愛。

2.分工

遊戲美工、遊戲流程規劃、目錄、基本效果、程式碼:鄧鎧晨

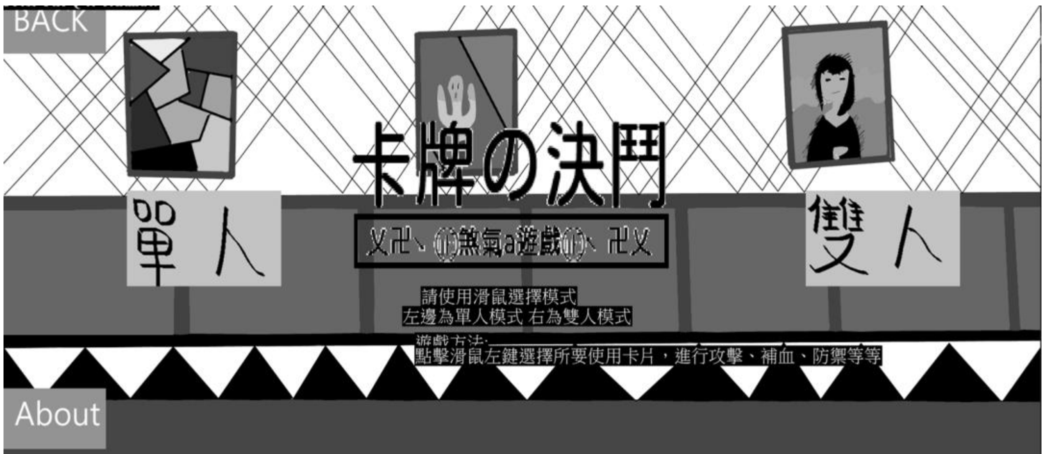
遊戲音效、遊戲架構規劃、動畫、進階效果、程式碼:許哲維

遊戲介紹

1.說明

遊戲首先會進到目錄，在目錄上可以看見有單人及雙人兩種模式可以選擇。在進入單人模式後，即可選擇四種地圖(四種難度，排序為左上<右上<左下<右下)，在選擇後即可開始進行遊戲，我方為左邊角色，反之右方為敵人，左下方可見 3 張鈔票即為本遊戲卡牌點數(一張五百為一點)，而在卡牌左上方即可看見該牌所需點數，而卡牌中央下方會顯示出該牌效果，點選該牌後，該牌就無法再次使用，點選下方結束案紐，其餘卡牌將會重新替換掉，而敵方也會自動做出攻擊，而左上方會顯示出一些卡牌的效果。而若選擇雙人模式，則兩方人各自出牌，並部會自動戰鬥。

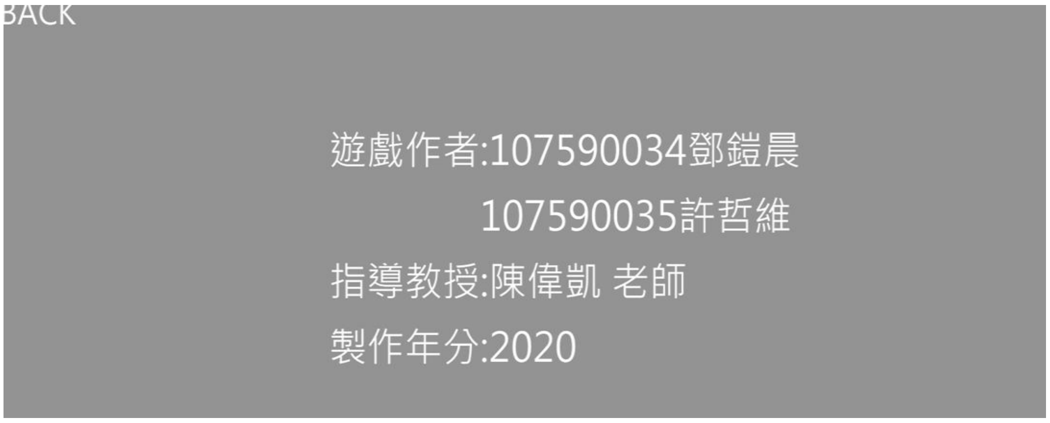
(1)遊戲主畫面



請使用滑鼠選擇模式
左邊為單人模式 右為雙人模式
遊戲方法:
點擊滑鼠左鍵選擇所要使用卡片，進行攻擊、補血、防禦等等

此為進入遊戲後會看到的主畫面，最左方為 Back 及 About 部分，Back 即為退出遊戲，而 About 部分可以看到作者簡介，此畫面即可選擇單人或雙人。

(2)About


BACK

遊戲作者:107590034鄧鎧晨

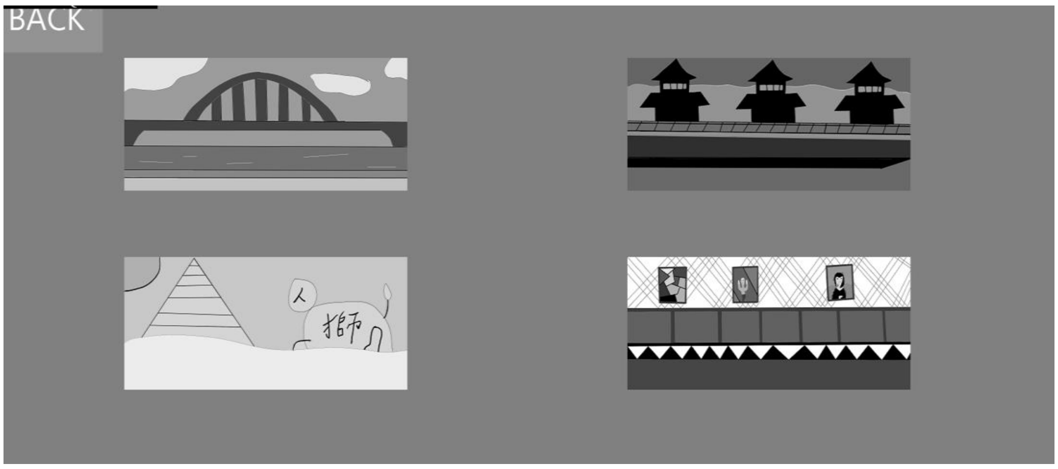
107590035許哲維

指導教授:陳偉凱 老師

製作年分:2020

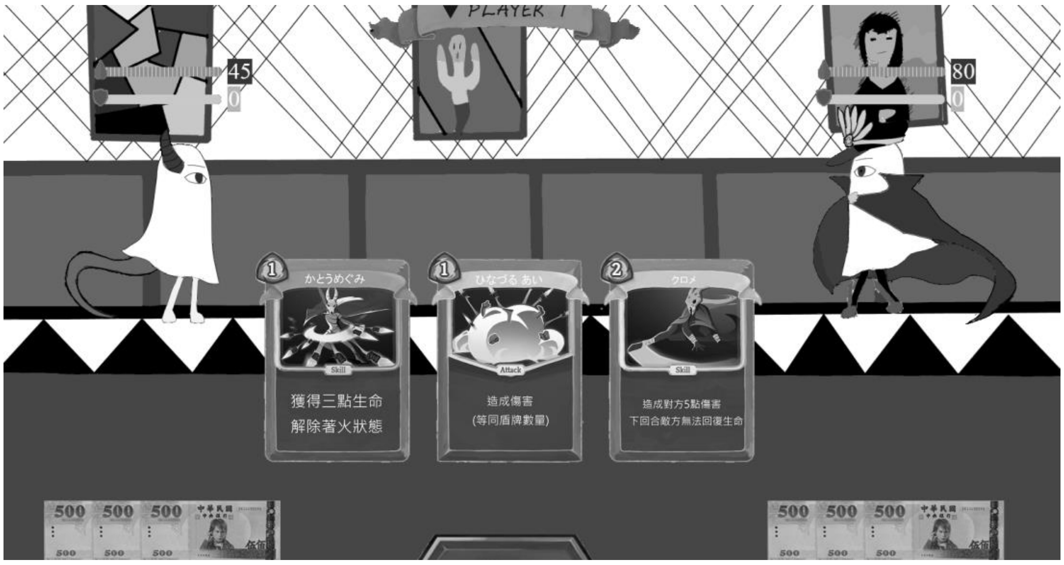
左上方可以回到上一頁。

(3)地圖選擇



在雙人模式下存為背景的選擇，而在單人模式下即為難度選擇。

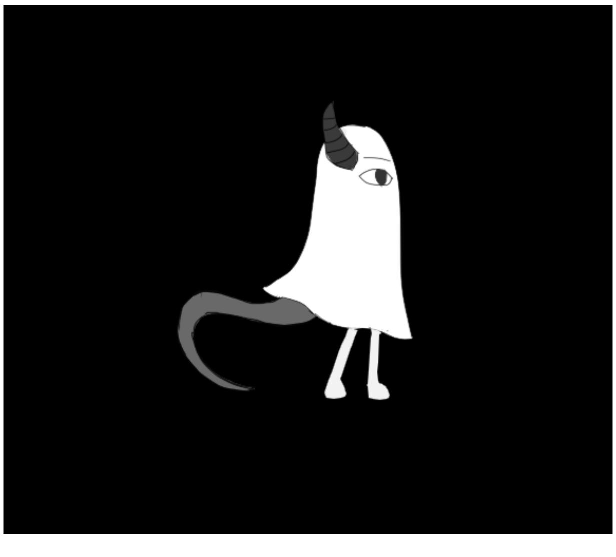
(4)遊戲畫面



下方為卡牌，上面有說明，點選後即會觸發，腳色上方可以看見血量、防禦值，左上方右上方則會顯示當前有的負面效果，下面鈔票即為點數

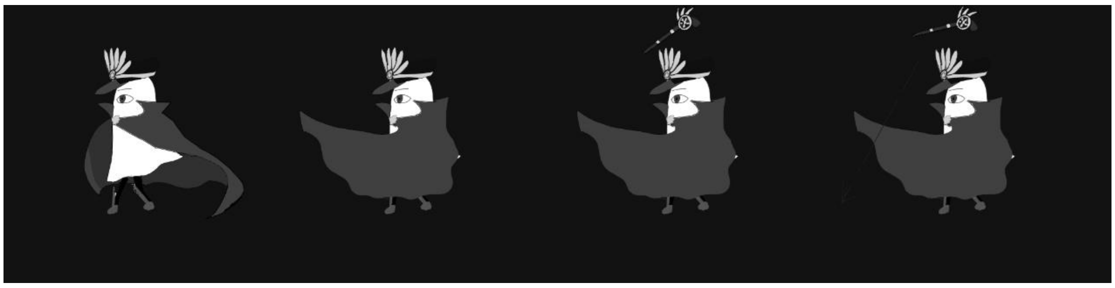
遊戲圖形:

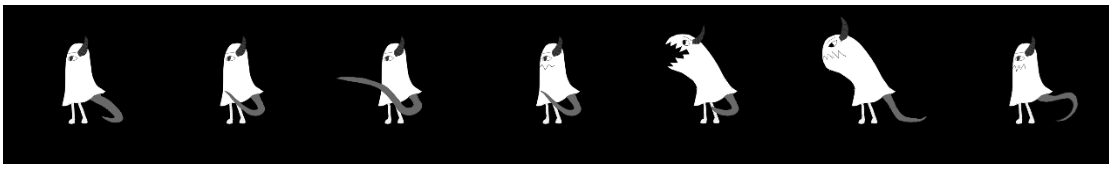
1. 角色設定圖



腳色取材來自埃及神話中的一位「不知名的神」，有人稱他為梅傑德，為一隻類似套著白布的神，而我們加上自己的想法，左邊長角的為「伊莉莎白」右邊則為「織田信長」。

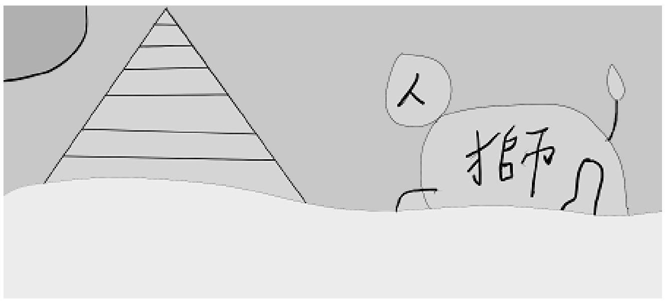
2. 角色分解圖



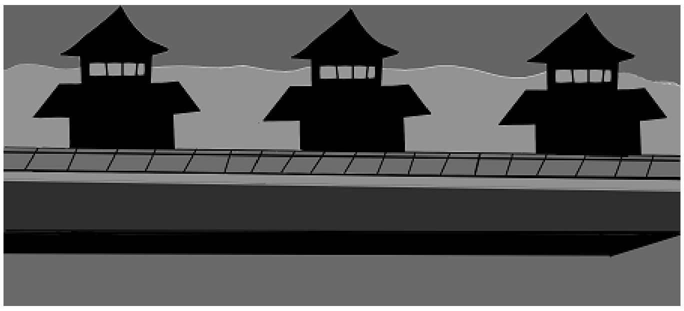


此為用來製作動畫的分解圖，全部都為手工繪製。

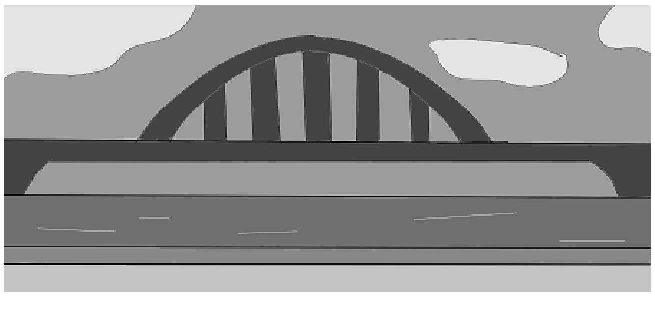
3. 地圖



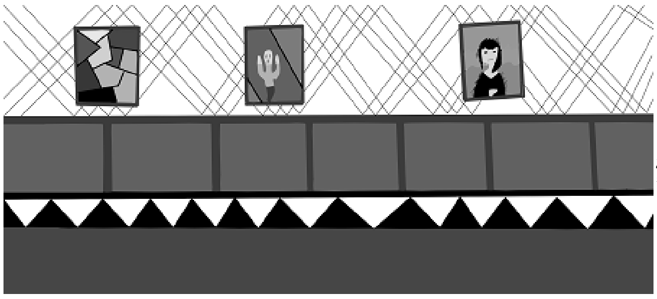
此圖取材來自埃及人面獅身。



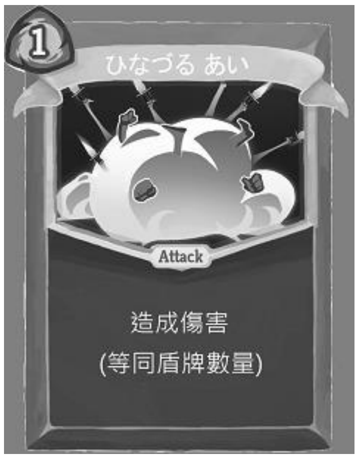
此圖取材來自日本忍者村。

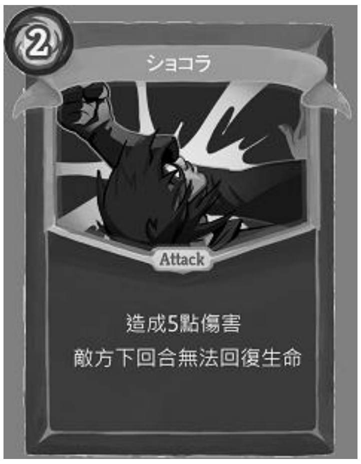


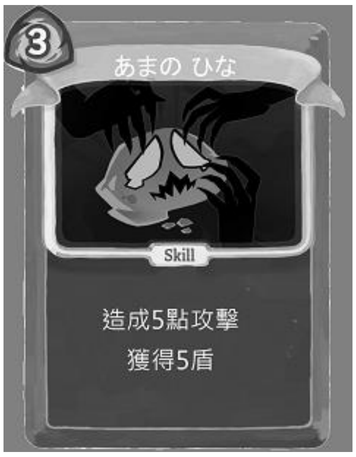
此圖取自台灣關渡大橋。


此圖取自美術館，右兩幅畫為吶喊、蒙娜麗莎。

4. 卡牌



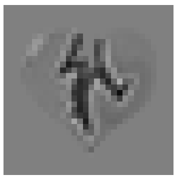




此三張為範例，各別為 1 費、2 費及 3 費。效果總共有 10 種，主要有攻擊、破防、燃燒(中毒)、破防、回血、頓擊(造成對方我的防禦值)等。

5.配件

此為血條，在低血量時會顯示出不同效果。



此為負面效果顯示圖。

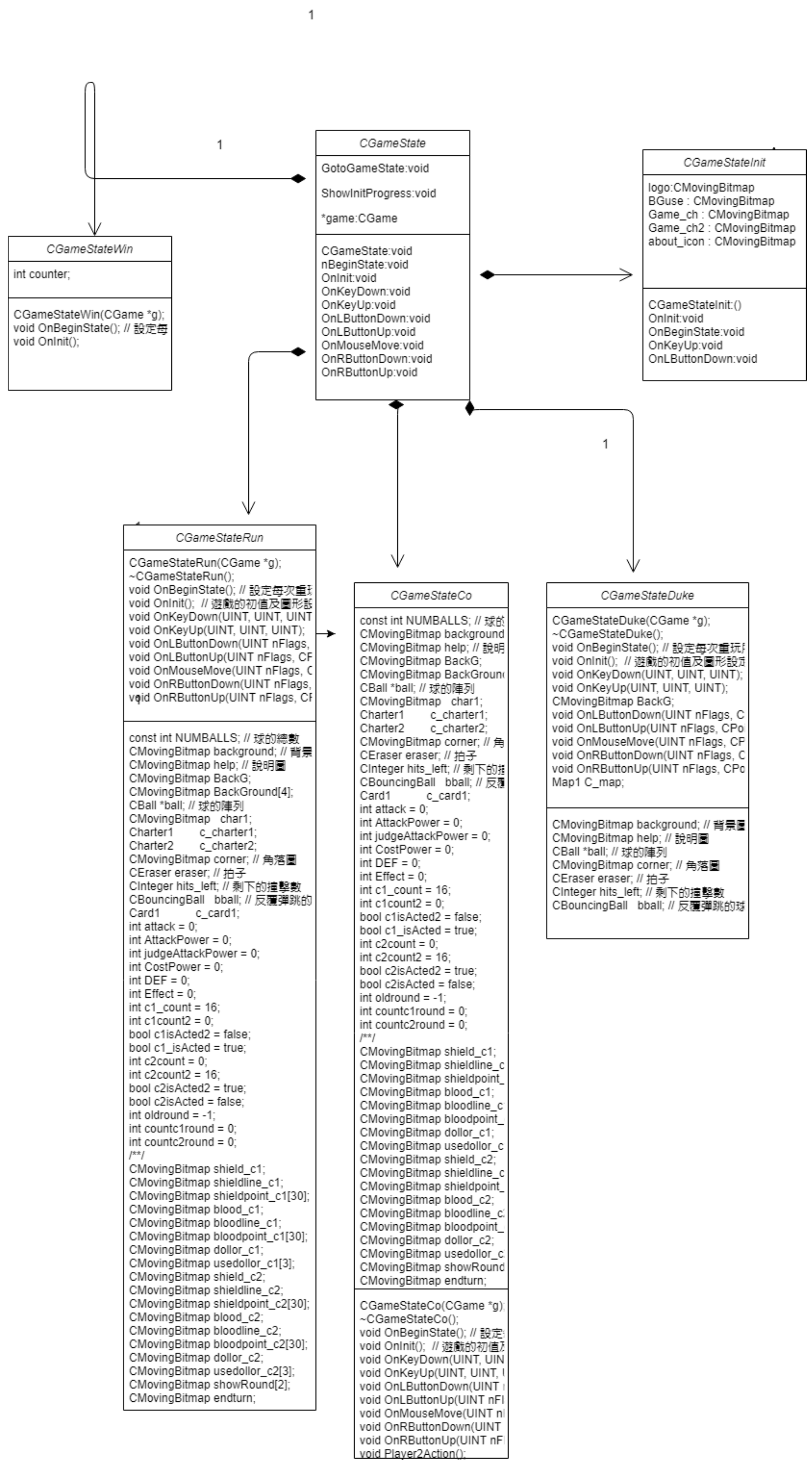
3.遊戲音效

編號	檔名	來源	說明
1	ntut.mid	老師	聲音測試
2	ding.wav	老師	聲音測試
3	lake.mp3	老師	聲音測試
4	BGM.mp3	https://youtu.be/3bmddRsZFkU	對戰音樂

程式設計

1. 程式架構

(一) Class



```
CGameState
GotoGameState:void
ShowInitProgress:void
*game:CGame
CGameState:void
nBeginState:void
OnInit:void
OnKeyDown:void
OnKeyUp:void
OnLButtonDown:void
OnLButtonUp:void
OnMouseMove:void
OnRButtonDown:void
OnRButtonUp:void
```

```
CGameStateWin
int counter;
CGameStateWin(CGame *g);
void OnBeginState(); // 設定每
void OnInit();
```

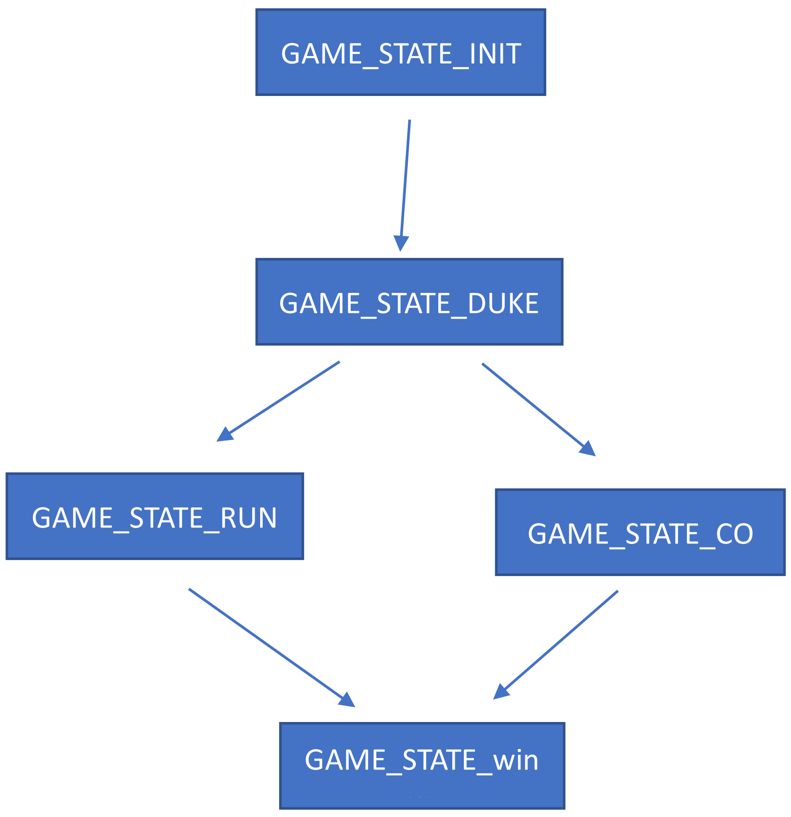
```
CGameStateInit
logo:CMovingBitmap
BGuse : CMovingBitmap
Game_ch : CMovingBitmap
Game_ch2 : CMovingBitmap
about_icon : CMovingBitmap
CGameStateInit:()
OnInit:void
OnBeginState:void
OnKeyUp:void
OnLButtonDown:void
```

```
CGameStateRun
CGameStateRun(CGame *g);
~CGameStateRun();
void OnBeginState(); // 設定每次重
void OnInit();  // 遊戲的初值及圖形設
void OnKeyDown(UINT, UINT, UINT
void OnKeyUp(UINT, UINT, UINT);
void OnLButtonDown(UINT nFlags,
void OnLButtonUp(UINT nFlags, CP
void OnMouseMove(UINT nFlags, C
void OnRButtonDown(UINT nFlags,
void OnRButtonUp(UINT nFlags, CP
const int NUMBALLS; // 球的總數
CMovingBitmap background; // 背景
CMovingBitmap help; // 說明圖
CMovingBitmap BackG;
CMovingBitmap BackGround[4];
CBall *ball; // 球的陣列
CMovingBitmap   char1;
Charter1        c_charter1;
Charter2        c_charter2;
CMovingBitmap corner; // 角落圖
CEraser eraser; // 拍子
CInteger hits_left; // 剩下的撞擊數
CBouncingBall   bball; // 反覆彈跳的
Card1           c_card1;
int attack = 0;
int AttackPower = 0;
int judgeAttackPower = 0;
int CostPower = 0;
int DEF = 0;
int Effect = 0;
int c1_count = 16;
int c1count2 = 0;
bool c1isActed2 = false;
bool c1_isActed = true;
int c2count = 0;
int c2count2 = 16;
bool c2isActed2 = true;
bool c2isActed = false;
int oldround = -1;
int countc1round = 0;
int countc2round = 0;
/**/
CMovingBitmap shield_c1;
CMovingBitmap shieldline_c1;
CMovingBitmap shieldpoint_c1[30];
CMovingBitmap blood_c1;
CMovingBitmap bloodline_c1;
CMovingBitmap bloodpoint_c1[30];
CMovingBitmap dollor_c1;
CMovingBitmap usedollor_c1[3];
CMovingBitmap shield_c2;
CMovingBitmap shieldline_c2;
CMovingBitmap shieldpoint_c2[30];
CMovingBitmap blood_c2;
CMovingBitmap bloodline_c2;
CMovingBitmap bloodpoint_c2[30];
CMovingBitmap dollor_c2;
CMovingBitmap usedollor_c2[3];
CMovingBitmap showRound[2];
CMovingBitmap endturn;
```

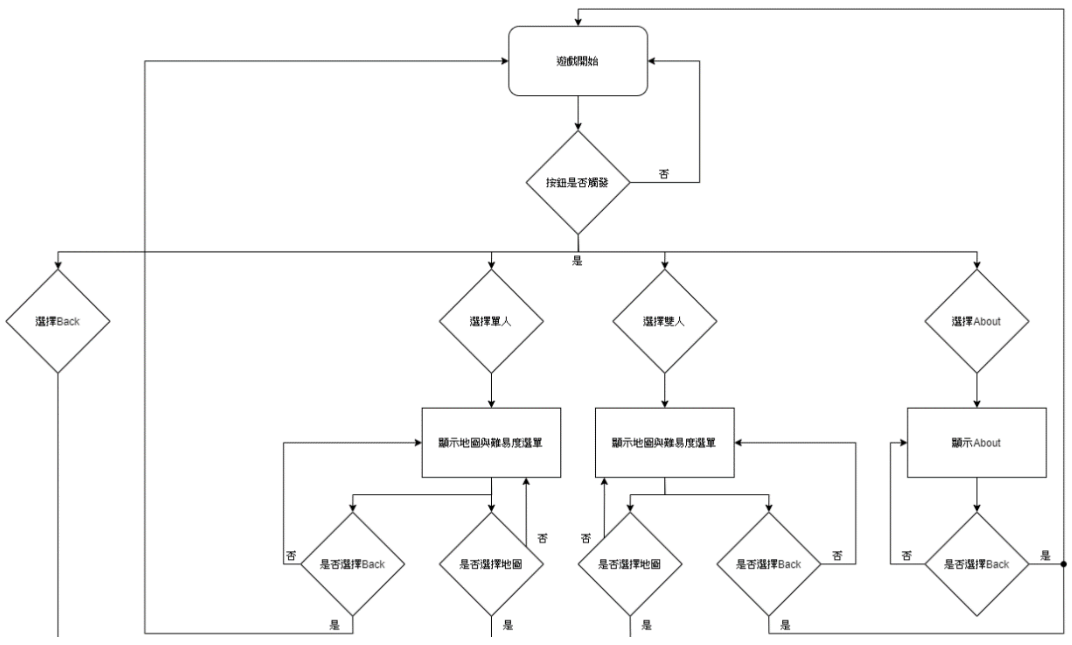
```
CGameStateCo
const int NUMBALLS; // 球的
CMovingBitmap background
CMovingBitmap help; // 說明
CMovingBitmap BackG;
CMovingBitmap BackGroun
CBall *ball; // 球的陣列
CMovingBitmap   char1;
Charter1        c_charter1;
Charter2        c_charter2;
CMovingBitmap corner; // 角
CEraser eraser; // 拍子
CInteger hits_left; // 剩下的
CBouncingBall   bball; // 反
Card1           c_card1;
int attack = 0;
int AttackPower = 0;
int judgeAttackPower = 0;
int CostPower = 0;
int DEF = 0;
int Effect = 0;
int c1_count = 16;
int c1count2 = 0;
bool c1isActed2 = false;
bool c1_isActed = true;
int c2count = 0;
int c2count2 = 16;
bool c2isActed2 = true;
bool c2isActed = false;
int oldround = -1;
int countc1round = 0;
int countc2round = 0;
/**/
CMovingBitmap shield_c1;
CMovingBitmap shieldline_c
CMovingBitmap shieldpoint_
CMovingBitmap blood_c1;
CMovingBitmap bloodline_c
CMovingBitmap bloodpoint_
CMovingBitmap dollor_c1;
CMovingBitmap usedollor_c
CMovingBitmap shield_c2;
CMovingBitmap shieldline_c
CMovingBitmap shieldpoint_
CMovingBitmap blood_c2;
CMovingBitmap bloodline_c
CMovingBitmap bloodpoint_
CMovingBitmap dollor_c2;
CMovingBitmap usedollor_c
CMovingBitmap showRound
CMovingBitmap endturn;
CGameStateCo(CGame *g)
~CGameStateCo();
void OnBeginState(); // 設定
void OnInit();  // 遊戲的初值
void OnKeyDown(UINT, UIN
void OnKeyUp(UINT, UINT,
void OnLButtonDown(UINT
void OnLButtonUp(UINT nFl
void OnMouseMove(UINT n
void OnRButtonDown(UINT
void OnRButtonUp(UINT nF
void Player2Action();
```

```
CGameStateDuke
CGameStateDuke(CGame *g);
~CGameStateDuke();
void OnBeginState(); // 設定每次重玩
void OnInit();  // 遊戲的初值及圖形設
void OnKeyDown(UINT, UINT, UINT);
void OnKeyUp(UINT, UINT, UINT);
CMovingBitmap BackG;
void OnLButtonDown(UINT nFlags, C
void OnLButtonUp(UINT nFlags, CPo
void OnMouseMove(UINT nFlags, CP
void OnRButtonDown(UINT nFlags, C
void OnRButtonUp(UINT nFlags, CPo
Map1 C_map;
CMovingBitmap background; // 背景圖
CMovingBitmap help; // 說明圖
CBall *ball; // 球的陣列
CMovingBitmap corner; // 角落圖
CEraser eraser; // 拍子
CInteger hits_left; // 剩下的撞擊數
CBouncingBall   bball; // 反覆彈跳的球
```

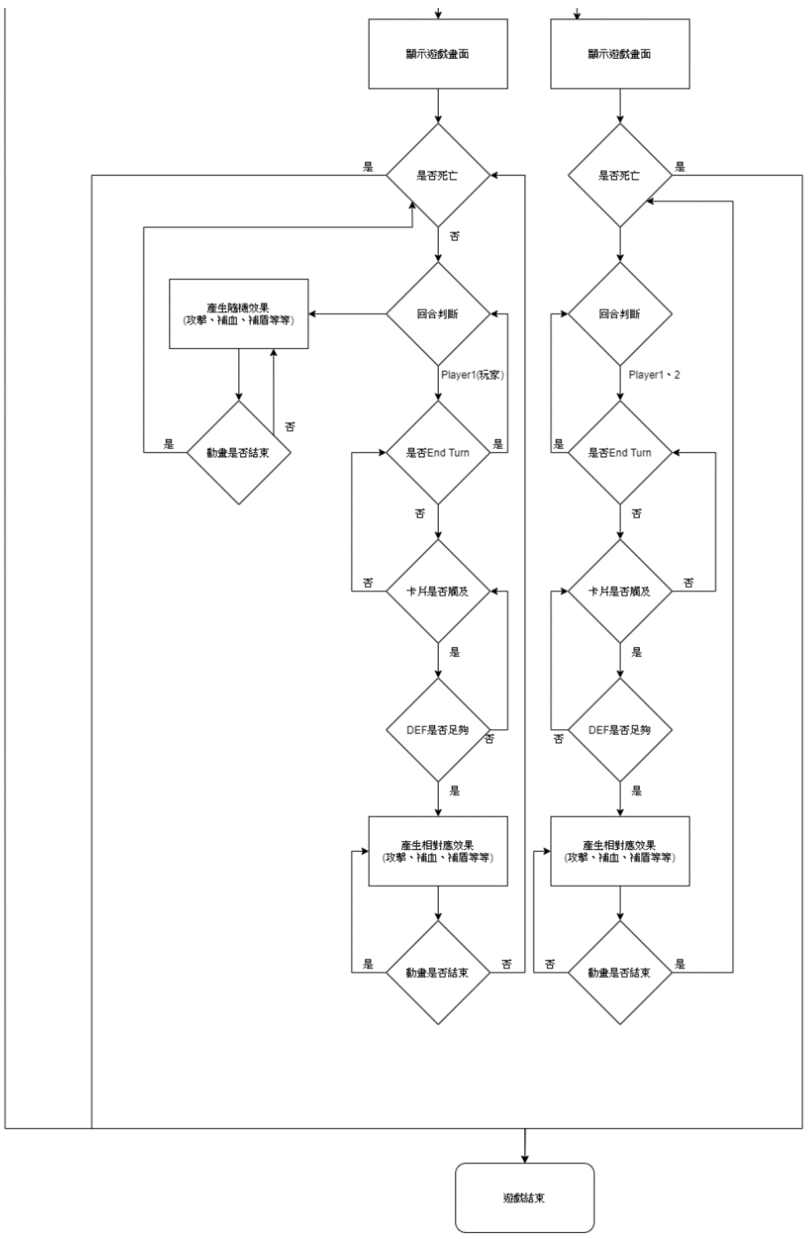
(二) GAME_STATE 運作



(三)完整主程式流程圖



接續下一頁


顯示遊戲畫面
顯示遊戲畫面
是否死亡
是
否
是否死亡
是
產生隨機效果
(攻擊、補血、補盾等等)
回合判斷
回合判斷
Player1(玩家)
Player1、2
否
是
動畫是否結束
是否End Turn
是
否
是否End Turn
否
否
卡片是否觸及
卡片是否觸及
否
是
是
DEF是否足夠
否
否
DEF是否足夠
是
是
產生相對應效果
(攻擊、補血、補盾等等)
產生相對應效果
(攻擊、補血、補盾等等)
是
動畫是否結束
否
否
動畫是否結束
是
遊戲結束

2. 程式分類

類別名稱	.h 檔行數	.cpp 檔行數	說明
CGameStateInit	11	103	初始化(目錄)，選擇介面
CGameStateWin	12	98	判斷輸贏
CGameStateRun	82	1368	執行雙人模式(傷害判定,卡牌效果)
CGameStateDuke	40	168	選擇地圖(儲存地圖資訊)
CGameStateCo	80	1443	執行單人模式(傷害判定,卡牌效果)
Charter1	19	114	儲存角色 1 資訊
Charter2	19	114	儲存角色 2 資訊
Card_info	9	18	儲存卡片基礎值
Card1	15	417	儲存卡片效果、產生卡片
Map1	14	46	產生地圖
總行數	301	3889	

結語

1.問題及解決方法

(1)團隊溝通問題:

在剛開始時,因為還不太清楚目標,所以兩個人都在做一樣的事情,並沒有分工,導致一樣的效果產生了兩個一樣的程式碼。

Sol:之後花了點時間討論遊戲整體的規劃，並且分工各自寫哪些部分。

(2)動畫無法呈現:

在剛開始時，動畫不是不會動，就是動了之後會一直不斷的重複。

Sol:增加判斷其狀態是否以觸發、是否已做完來解決此問題。

(3)最大化會超出介面:

在第二次投影到螢幕上時，最大化會超出介面

Sol:筆電解析度在投影在螢幕上時產生設定上的問題，但在替換電腦後已解決此問題。

(4)卡片無法在下次遊戲重新開始

Sol:多寫了一個參數判定是否要重新抓取新資訊。

(5)遊戲過於單調:

Sol:多增加了不同模式，並且增加難度選擇。

(6)不知道怎麼增加切換頁面

Sol:在詢問老師和同學後得到了幫助。

(7)效果一開始很多都會無法使用，或是隨機跑出來

Sol:把最初始的版本改掉了，之後多了很多的判斷確保不會出錯，並且確定可以運行。

2.時間表

週數	姓名	時數	說明
1	鄧鎧晨	6	練習 framework
	許哲維	12	練習 framework、遊戲構思
2	鄧鎧晨	6	遊戲構想、找素材
	許哲維	11	角色呈現、背景設定
3	鄧鎧晨	8	製作角色、背景
	許哲維	11	卡片設定
4	鄧鎧晨	8	製作卡片效果
	許哲維	9	基礎動畫呈現
5	鄧鎧晨	8	製作回合
	許哲維	8	動畫修正
6	鄧鎧晨	6	修正效果
	許哲維	7	卡片效果除錯
7	鄧鎧晨	10	增加卡片費用
	許哲維	7	卡片效果除錯
8	鄧鎧晨	11	補充地圖、做出地圖選擇
	許哲維	4	卡片效果改善
9	鄧鎧晨	6	遊戲畫面美化
	許哲維	11	畫面美化、動畫修正

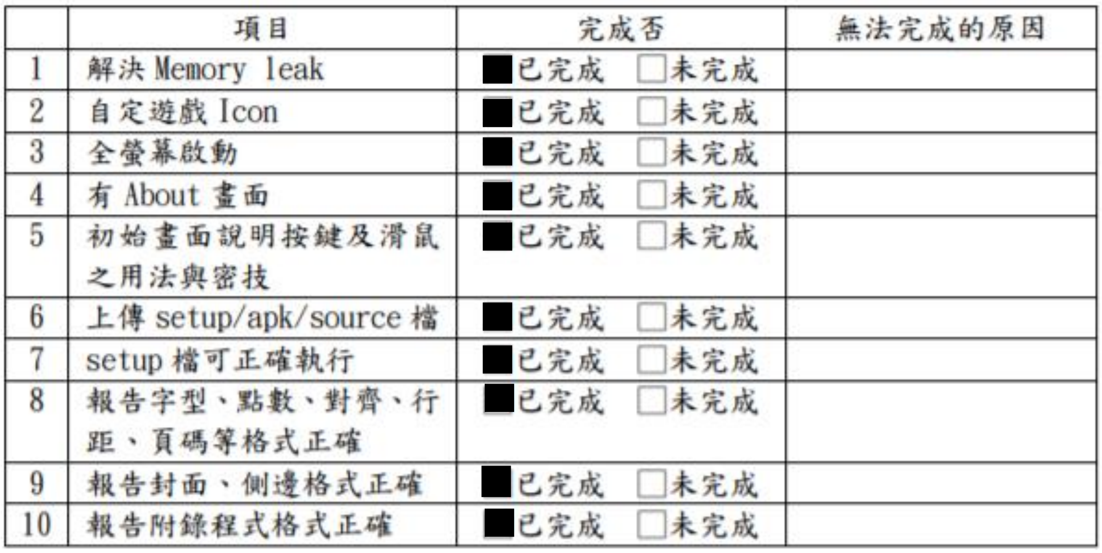
10	鄧鎧晨	8	討論單人模式
	許哲維	18	數值顯示、畫面美化
11	鄧鎧晨	8	製作電腦自動攻擊
	許哲維	4	增加遊戲結束畫面
12	鄧鎧晨	6	設計難度
	許哲維	5	自動攻擊設定
13	鄧鎧晨	10	做出不同難度
	許哲維	5	自動攻擊修正
14	鄧鎧晨	8	修正遊戲難度 BUG
	許哲維	8	動畫除錯
15	鄧鎧晨	6	補足剩下不達標準處(簡介等...)
	許哲維	20	狀態顯示改善

3.貢獻比例

鄧鎧晨----50%

許哲維----50%

4.自我檢核表



	項目	完成否	無法完成的原因
1	解決 Memory leak	■已完成 □未完成	
2	自定遊戲 Icon	■已完成 □未完成	
3	全螢幕啟動	■已完成 □未完成	
4	有 About 畫面	■已完成 □未完成	
5	初始畫面說明按鍵及滑鼠之用法與密技	■已完成 □未完成	
6	上傳 setup/apk/source 檔	■已完成 □未完成	
7	setup 檔可正確執行	■已完成 □未完成	
8	報告字型、點數、對齊、行距、頁碼等格式正確	■已完成 □未完成	
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.收穫

鄧鎧晨

這次的實作中學到了 Github 的運用，利用 Github 可以很方便的與隊友進行合作。而在程式碼中，學會了如何使用一些事件的觸發，像是點擊滑鼠，一開始很擔心不知道如何使用，但很快的看了一下，就懂了像是 x.point 就可以很快的做出滑鼠的觸發動作，也學到了怎麼 show 出一個圖片出來，並且了解了 GAME_STATE 的運作，怎麼在不同的階段間進行切換的動作，也重新的複習了不同的 class 之間要如何有效的互相運作。

許哲維

在這學習的實習中學到了如何運用 GitHub 來管理版本，以及如何透過 GitHub 的邀請來方便與隊友合作進行作業。在撰寫程式碼中，學到了如何以多張圖片來顯示動畫，也透過閱讀程式碼，了解了滑鼠點擊事件的處理與運作模式，而在創造不同 GAME_STATE 時，也學到了如何進行畫面的切換，重新複習到了 Class 的切換與資源共享、傳送等等，在這學期的實習中讓我學到許多、收穫滿滿。

6.心得

鄧鎧晨

經過了上學期物件導向的學習之後，這學期要用自己的想法來做出一個遊戲出來，最主要的收穫就是讓我學會所謂的團隊合作，很多時候不是一個人想做什麼就固執的堅持著自己的想法，有時候聽聽隊友的意見，問題都可以比自己一個人想還要來得容易解決，也學習到了怎麼實際運用 Github，以前雖然看過 Github 但是卻不懂是在幹嘛的東西，現在了解到了他的方便，變得超喜歡使用，也在這個每個禮拜進度壓力下，逼出了我一直以來覺得我無法打出的程式碼，最後可以如期完成，也要好好感謝我的隊友一路上的支持與幫助。

而製作遊戲也是我一直以來想做的事情，所以也進入了資工系，在製作的過程雖然相當的辛苦，但是遇到一些 BUG 時有時也會覺得怎麼那麼好笑，感覺到了自己的無能，但是在最後的那種成就感真的很難形容，就好像這個遊戲是自己的孩子，從什麼都沒有的樣子，到最後呈現出了很多一開始從未想過會有的樣子，真的很感動，但是也很明顯的看出了我的實力遠遠不足其他的同學甚至是我的隊友，很多時候我都必須四處詢問才知道要怎麼解決，而看到成績時，心中也有點傷心，感覺拖累了我的隊友，很常覺得是不是做錯選擇了，是不是我自己沒有這個天分在程式上，好險我的隊友相當的包容我，所以我才能撐到最後的 DEMO，真的相當感謝他的包容。

許哲維

經過這一整個學期的物件導向實習課程，在課程中學習到如何自己製作出一款遊戲，而在這一次的實習，讓我深深了解到團隊合作的重要，常常在撰寫程式碼時遇到問題時，透過隊友的解釋與協助，讓我能有效率的解決問題、並改善程式，也透過這次實習，讓我第一次操作到了 GitHub，也透過實作了解到 GitHub 的便利性、以及在版本管理上更是清楚明瞭、淺顯易懂，也因每個禮拜的進度壓力下，讓我從中體會到撰寫程式的辛苦，也因為這個進度壓力，讓我能夠不拖延時間趕緊完成程式的製作業，也讓我們能夠順利的完成遊戲的製作，最後，透過隊友將遊戲整體架構先撰寫、規劃出來，讓我們的程式能夠順利地運作，以減少修正錯誤的時間，感謝他一路上的種種協助與付出。

而在製作遊戲時，我常常因為自己的不嚴謹，讓程式產生許多不應該發生的錯誤，造成時間上的浪費，而在每次 Demo 成績公布時，常常因為成績不理想而感到灰心，但藉由助教與老師的建議下，讓我們有遊戲製作的方向，而在我越接近期末時，常常被其他作業壓得喘不過氣，為了其他作業，導致程式的撰寫大多在三更半夜，在撰寫時常常一邊思考自己是不是不適合走這條路，但在透過隊友的協助與包容下，讓我能順利撐過最後的 Demo，感謝他這一路以來對我的協助與包容。

附錄

Mygame.cpp

```
//第一關正常
//第二關血量 2 倍
//第三關攻擊力+5
//第四關血量 2 倍  攻擊力+5
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"
#include <stdlib.h>
#include <time.h>
#include <sstream>

namespace game_framework {
	/////////////////////////////////////////////////////////////////////////////
	// 這個 class 為遊戲的遊戲開頭畫面物件
	/////////////////////////////////////////////////////////////////////////////
	int map_choose = 0,game_choose=0,is_attack=0;
	int Player2isEnd = 0;
	int get_win = 0;
	int init_pass = 0;

	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
		//
		//BGuse.LoadBitmap("Bitmaps/ESHL.bmp");
		BGuse.LoadBitmap("Bitmaps/ESHLbig.bmp");
		Game_ch.LoadBitmap("Bitmaps/O.bmp");
		Game_ch2.LoadBitmap("Bitmaps/CO.bmp");
		about_icon.LoadBitmap("Bitmaps/about_icon.bmp");
		logo.LoadBitmap("RES/Logon.bmp", RGB(255, 255, 255));
		back.LoadBitmap("Bitmaps\\back.bmp");
		//logo.LoadBitmap(IDB_BACKGROUND);
		// 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep
//
// 此 OnInit 動作會接到 CGameStaterRun::OnInit()，所以進度還沒到 100%
//
	}

	void CGameStateInit::OnBeginState()
	{
	}

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

	}

	void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point)
	{
        game_choose = 0;
        if (point.x > 0 && point.x < 166 && point.y>0 && point.y < 96)
        {
                exit(0);
        }
        if (point.x > 200 && point.x < 450 && point.y>300 && point.y < 445)
        {
                game_choose = 1;
        }
        if (point.x > 1300 && point.x < 1550 && point.y>300 && point.y < 445)
        {
                game_choose = 2;
        }
        if (point.x > 0 && point.x < 166 && point.y>600 && point.y < 692)
        {
                game_choose = 0;
                GotoGameState(GAME_STATE_OVER);
        }
        if (game_choose != 0) {
                GotoGameState(GAME_STATE_DUKE);             // 切換至 GAME_STATE_RUN
        }
}

void CGameStateInit::OnShow()
{
        //
        // 貼上 logo
        //
        BGuse.SetTopLeft(0, 0);
        BGuse.ShowBitmap();
        Game_ch.SetTopLeft(200, 300);
        Game_ch.ShowBitmap();
        Game_ch2.SetTopLeft(1300, 300);
        Game_ch2.ShowBitmap();
        about_icon.SetTopLeft(0, 600);
        about_icon.ShowBitmap();
        back.SetTopLeft(0, 0);
        back.ShowBitmap();
        //
        //
        logo.SetTopLeft((SIZE_X - logo.Width()) / 2, SIZE_Y / 8);
        logo.ShowBitmap();
        //
        // Demo 螢幕字型的使用，不過開發時請盡量避免直接使用字型，改用 CMovingBitmap 比較好
        //
        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 16 point 的字
        fp = pDC->SelectObject(&f);                         // 選用 font f
        pDC->SetBkColor(RGB(0, 0, 0));
        pDC->SetTextColor(RGB(255, 255, 0));
        pDC->TextOut(577+100, 395 + 50, "請使用滑鼠選擇模式");
        pDC->TextOut(677-30, 395+80, "左邊為單人模式 右為雙人模式");
        if (ENABLE_GAME_PAUSE)
                pDC->TextOut(637+30, 395 + 120, "遊戲方法:");
        pDC->TextOut(667, 395 + 140, "點擊滑鼠左鍵選擇所要使用卡片，進行攻擊、補血、防禦等等");
        pDC->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 為遊戲的結束狀態(Game Over)
/////////////////////////////////////////////////////////////////////////////

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

```
	void CGameStateOver::OnLButtonUp(UINT nFlags, CPoint point)
	{

		if (point.x > 0 && point.x < 166 && point.y>0 && point.y < 96)
		{
			GotoGameState(GAME_STATE_INIT);
		}

	}

	void CGameStateOver::OnBeginState()
	{

	}

	void CGameStateOver::OnInit()
	{
		back.LoadBitmap("Bitmaps\\back.bmp");
		about.LoadBitmap("Bitmaps\\about.bmp");
	}

	void CGameStateOver::OnShow()
	{
		about.SetTopLeft(0, 0);
		about.ShowBitmap();
		back.SetTopLeft(0, 0);
		back.ShowBitmap();

	}

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




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


	//地圖選擇

	Map1::Map1()
	{
		this->x = 200;
		this->y = 100;
	}
	void Map1::LoadBit()
	{
		LoadMap[0].LoadBitmap("Bitmaps\\DM_mini.bmp");
		LoadMap[1].LoadBitmap("Bitmaps\\BNS_mini.bmp");
		LoadMap[2].LoadBitmap("Bitmaps\\IG_mini.bmp");
		LoadMap[3].LoadBitmap("Bitmaps\\ESHL_mini.bmp");

	}
	void Map1::OnLButtonUp(UINT nFlags, CPoint point)
	{
		map_choose = map_choose;
		if (point.x > 200 && point.x < 650 && point.y>100 && point.y < 300)
		{
			map_choose = 1;
		}
		if (point.x > 1000 && point.x < 1450 && point.y>100 && point.y < 300)
		{
			map_choose = 2;
		}
		if (point.x > 200 && point.x < 650 && point.y>400 && point.y < 600)
```

```
				{
					map_choose = 3;
				}
				if (point.x >1000 && point.x < 1450 && point.y>400 && point.y < 600)
				{
					map_choose = 4;
				}

		}
		void Map1::onShow()
		{
			LoadMap[0].SetTopLeft(x, y);
			LoadMap[1].SetTopLeft(x+800, y);
			LoadMap[2].SetTopLeft(x, y+300);
			LoadMap[3].SetTopLeft(x+800, y+300);
			LoadMap[0].ShowBitmap();
			LoadMap[1].ShowBitmap();
			LoadMap[2].ShowBitmap();
			LoadMap[3].ShowBitmap();
		}

		//

CGameStateDuke::CGameStateDuke(CGame *g)
		: CGameState(g)
{
		//
}

CGameStateDuke::~CGameStateDuke()
{
		//
}

void CGameStateDuke::OnBeginState()
{
		const int BALL_GAP = 90;
		const int BALL_XY_OFFSET = 45;
		const int BALL_PER_ROW = 7;
		const int HITS_LEFT = 10;
		const int HITS_LEFT_X = 590;
		const int HITS_LEFT_Y = 0;
		const int BACKGROUND_X = 60;
		const int ANIMATION_SPEED = 15;
		background.SetTopLeft(BACKGROUND_X, 0);				// 設定背景的起始座標
		BackG.SetTopLeft(0, 0);
		help.SetTopLeft(0, SIZE_Y - help.Height());			// 設定說明圖的起始座標

}

void CGameStateDuke::OnMove()							// 移動遊戲元素
{
		//
}

void CGameStateDuke::OnInit()								// 遊戲的初值及圖形設定
{
		//
		// 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(33);		// 接個前一個狀態的進度，此處進度視為 33%
		//
		// 開始載入資料
		//
		background.LoadBitmap("Bitmaps\\BG.bmp");				// 載入背景的圖形
		BackG.LoadBitmap("Bitmaps\\BG.bmp");
		back.LoadBitmap("Bitmaps\\back.bmp");
		//C_map.LoadMap[0].
```

```
        C_map.LoadBit();
        //
        // 完成部分 Loading 動作，提高進度
        //
        ShowInitProgress(50);
        Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep
        //
        // 繼續載入其他資料
        //
        help.LoadBitmap(IDB_HELP, RGB(255, 255, 255));                    // 載入說明的圖形
        corner.LoadBitmap(IDB_CORNER);                                    // 載入角落圖形
        corner.ShowBitmap(background);                                    // 將 corner 貼到 background

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

void CGameStateDuke::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
        //
}

void CGameStateDuke::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
        const char KEY_LEFT = 0x25; // keyboard 左箭頭
        const char KEY_UP = 0x26; // keyboard 上箭頭
        const char KEY_RIGHT = 0x27; // keyboard 右箭頭
        const char KEY_DOWN = 0x28; // keyboard 下箭頭
        //
}

void CGameStateDuke::OnLButtonDown(UINT nFlags, CPoint point)   // 處理滑鼠的動作
{
        C_map.OnLButtonUp(nFlags, point);
        if (point.x > 0 && point.x < 166 && point.y>0 && point.y < 96)
        {
                game_choose = 0;
                GotoGameState(GAME_STATE_INIT);
        }
        if (map_choose > 0) {
                if (game_choose == 1)
                {
                        GotoGameState(GAME_STATE_CO);
                }
                if (game_choose == 2)
                {
                        GotoGameState(GAME_STATE_RUN);
                }
        }
}

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

void CGameStateDuke::OnMouseMove(UINT nFlags, CPoint point)      // 處理滑鼠的動作
{
        // 沒事。如果需要處理滑鼠移動的話，寫 code 在這裡
}

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

```
}

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

void CGameStateDuke::OnShow()
{
        //
        //  注意：Show 裡面千萬不要移動任何物件的座標，移動座標的工作應由 Move 做才對，
        //          否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語
        //          說，Move 負責 MVC 中的 Model，Show 負責 View，而 View 不應更動 Model。
        //
        //
        //  貼上背景圖、撞擊數、球、擦子、彈跳的球
        //

        //background.ShowBitmap();                        // 貼上背景圖
        BackG.ShowBitmap();
        C_map.onShow();
        back.SetTopLeft(0, 0);
        back.ShowBitmap();
        //help.ShowBitmap();                              // 貼上說明圖
        //hits_left.ShowBitmap();
        //for (int i = 0; i < NUMBALLS; i++)
        //        ball[i].OnShow();                       // 貼上第 i 號球
        //bball.OnShow();                                         // 貼上彈跳的球
        //eraser.OnShow();                                // 貼上擦子
        //
        //  貼上左上及右下角落的圖
        //
        //corner.SetTopLeft(0, 0);
        //corner.ShowBitmap();
        //corner.SetTopLeft(SIZE_X - corner.Width(), SIZE_Y - corner.Height());
        //corner.ShowBitmap();
}



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


CGameStateCo::CGameStateCo(CGame *g)
        : CGameState(g), NUMBALLS(28)
{
        //ball = new CBall [NUMBALLS];
}

CGameStateCo::~CGameStateCo()
{
        //delete [] ball;

}

void CGameStateCo::OnBeginState()
{
        CAudio::Instance()->Play(AUDIO_BGM, true);
        /*const int BALL_GAP = 90;
        const int BALL_XY_OFFSET = 45;
        const int BALL_PER_ROW = 7;
        const int HITS_LEFT = 10;
        const int HITS_LEFT_X = 590;
        const int HITS_LEFT_Y = 0;
        const int BACKGROUND_X = 60;
        const int ANIMATION_SPEED = 15;
        for (int i = 0; i < NUMBALLS; i++) {                      // 設定球的起始座標
```

```
			int x_pos = i % BALL_PER_ROW;
			int y_pos = i / BALL_PER_ROW;
			ball[i].SetXY(x_pos * BALL_GAP + BALL_XY_OFFSET, y_pos * BALL_GAP + BALL_XY_OFFSET);
			ball[i].SetDelay(x_pos);
			ball[i].SetIsAlive(true);
		}
		eraser.Initialize();
		background.SetTopLeft(BACKGROUND_X,0);				// 設定背景的起始座標
		help.SetTopLeft(0, SIZE_Y - help.Height());			// 設定說明圖的起始座標
		hits_left.SetInteger(HITS_LEFT);					// 指定剩下的撞擊數
		hits_left.SetTopLeft(HITS_LEFT_X,HITS_LEFT_Y);		// 指定剩下撞擊數的座標
		CAudio::Instance()->Play(AUDIO_LAKE, true);			// 撥放 WAVE
		CAudio::Instance()->Play(AUDIO_DING, false);		// 撥放 WAVE
		CAudio::Instance()->Play(AUDIO_NTUT, true);			// 撥放 MIDI*/
}

void CGameStateCo::OnMove()							// 移動遊戲元素
{
		//
		// 如果希望修改 cursor 的樣式，則將下面程式的 commment 取消即可
		//
		// SetCursor(AfxGetApp()->LoadCursor(IDC_GAMECURSOR));
		//
		// 移動背景圖的座標
		//
		/*BackGround.SetTopLeft(0, 0);
		if (background.Top() > SIZE_Y)
			background.SetTopLeft(60 ,-background.Height());
		background.SetTopLeft(background.Left(),background.Top()+1);*/
		//
		// 移動球
		//

		//int i;
		/*for (i=0; i < NUMBALLS; i++)
			ball[i].OnMove();*/
			//
			// 移動擦子
			//
			//eraser.OnMove();
			//
			// 判斷擦子是否碰到球
			//
			/*for (i=0; i < NUMBALLS; i++)
				if (ball[i].IsAlive() && ball[i].HitEraser(&eraser)) {
					ball[i].SetIsAlive(false);
					CAudio::Instance()->Play(AUDIO_DING);
					hits_left.Add(-1);
					//
					// 若剩餘碰撞次數為 0，則跳到 Game Over 狀態
					//
					if (hits_left.GetInteger() <= 0) {
						CAudio::Instance()->Stop(AUDIO_LAKE);	// 停止 WAVE
						CAudio::Instance()->Stop(AUDIO_NTUT);	// 停止 MIDI
						GotoGameState(GAME_STATE_OVER);
					}
				}*/
				//
				// 移動彈跳的球
				//
				//bball.OnMove();
}

void CGameStateCo::OnInit()								// 遊戲的初值及圖形設定
{
		//
		// 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(33);           // 接個前一個狀態的進度，此處進度視為 33%
//
// 開始載入資料
//
//BackGround.LoadBitmap("Bitmaps/ESHL.bmp");

BackGround[0].LoadBitmap("Bitmaps\\DMbig.bmp");

BackGround[1].LoadBitmap("Bitmaps\\BNSbig.bmp");

BackGround[2].LoadBitmap("Bitmaps\\IGbig.bmp");

BackGround[3].LoadBitmap("Bitmaps\\ESHLbig.bmp");


char1.LoadBitmapA("Bitmaps\\Forward.bmp", RGB(0, 0, 0));
shield_c1.LoadBitmapA("Bitmaps\\showshield.bmp", RGB(50, 150, 150));
shieldline_c1.LoadBitmapA("Bitmaps\\shieldline.bmp", RGB(50, 150, 150));
for (int i = 0; i < 30; i++)
{
        if (i == 0) { shieldpoint_c1[0].LoadBitmapA("Bitmaps\\shield_h.bmp", RGB(50, 150, 150)); }
        else if (i == 29) { shieldpoint_c1[29].LoadBitmapA("Bitmaps\\shield_t.bmp", RGB(50, 150, 150)); }
        else { shieldpoint_c1[i].LoadBitmapA("Bitmaps\\shield_m.bmp", RGB(50, 150, 150)); }
}
blood_c1.LoadBitmapA("Bitmaps\\showblood.bmp", RGB(50, 150, 150));
bloodline_c1.LoadBitmapA("Bitmaps\\bloodline.bmp", RGB(50, 150, 150));
for (int i = 0; i < 30; i++)
{
        if (i == 0) { bloodpoint_c1[0].LoadBitmapA("Bitmaps\\blood_h.bmp", RGB(50, 150, 150)); }
        else if (i == 29) { bloodpoint_c1[29].LoadBitmapA("Bitmaps\\blood_t.bmp", RGB(50, 150, 150)); }
        else { bloodpoint_c1[i].LoadBitmapA("Bitmaps\\blood_m.bmp", RGB(50, 150, 150)); }
}
for (int i = 0; i < 3; i++) { usedollor_c1[i].LoadBitmapA("Bitmaps\\dollor.bmp", RGB(50, 150, 150)); }
shield_c2.LoadBitmapA("Bitmaps\\showshield.bmp", RGB(50, 150, 150));
shieldline_c2.LoadBitmapA("Bitmaps\\shieldline.bmp", RGB(50, 150, 150));
for (int i = 0; i < 30; i++)
{
        if (i == 0) { shieldpoint_c2[0].LoadBitmapA("Bitmaps\\shield_h.bmp", RGB(50, 150, 150)); }
        else if (i == 29) { shieldpoint_c2[29].LoadBitmapA("Bitmaps\\shield_t.bmp", RGB(50, 150, 150)); }
        else { shieldpoint_c2[i].LoadBitmapA("Bitmaps\\shield_m.bmp", RGB(50, 150, 150)); }
}
blood_c2.LoadBitmapA("Bitmaps\\showblood.bmp", RGB(50, 150, 150));
bloodline_c2.LoadBitmapA("Bitmaps\\bloodline.bmp", RGB(50, 150, 150));
for (int i = 0; i < 30; i++)
{
        if (i == 0) { bloodpoint_c2[0].LoadBitmapA("Bitmaps\\blood_h.bmp", RGB(50, 150, 150)); }
        else if (i == 29) { bloodpoint_c2[29].LoadBitmapA("Bitmaps\\blood_t.bmp", RGB(50, 150, 150)); }
        else { bloodpoint_c2[i].LoadBitmapA("Bitmaps\\blood_m.bmp", RGB(50, 150, 150)); }
}
for (int i = 0; i < 3; i++) { usedollor_c2[i].LoadBitmapA("Bitmaps\\dollor.bmp", RGB(50, 150, 150)); }
showRound[0].LoadBitmapA("Bitmaps\\charter1n.bmp", RGB(50, 150, 150));
showRound[1].LoadBitmapA("Bitmaps\\charter2n.bmp", RGB(50, 150, 150));
endturn.LoadBitmapA("Bitmaps\\endturn.bmp", RGB(50, 150, 150));
c_charter1.LoadBit();
c_charter2.LoadBit();
c_card1.LoadBit();

/*int i;
for (i = 0; i < NUMBALLS; i++)
        ball[i].LoadBitmap();                                          // 載入第 i 個球的圖形*/
        //eraser.LoadBitmap();
        //background.LoadBitmap(IDB_BACKGROUND);                              // 載入背景的圖形
        //
        // 完成部分 Loading 動作，提高進度
        //
ShowInitProgress(50);
Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep
//
// 繼續載入其他資料
```

```
	//
	help.LoadBitmap(IDB_HELP, RGB(255, 255, 255));				// 載入說明的圖形
	/*corner.LoadBitmap(IDB_CORNER);						// 載入角落圖形
	corner.ShowBitmap(background);						// 將 corner 貼到 background
	bball.LoadBitmap();							// 載入圖形
	hits_left.LoadBitmap();			*/

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

}

void CGameStateCo::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
	if (nChar == KEY_LEFT)
		eraser.SetMovingLeft(true);
	if (nChar == KEY_RIGHT)
		eraser.SetMovingRight(true);
	if (nChar == KEY_UP)
		eraser.SetMovingUp(true);
	if (nChar == KEY_DOWN)
		eraser.SetMovingDown(true);*/
}

void CGameStateCo::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)
		eraser.SetMovingLeft(false);
	if (nChar == KEY_RIGHT)
		eraser.SetMovingRight(false);
	if (nChar == KEY_UP)
		eraser.SetMovingUp(false);
	if (nChar == KEY_DOWN)
		eraser.SetMovingDown(false);*/
}

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





}

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

	if (init_pass==0)
	{
		if (map_choose == 1)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 45;
		}
		if (map_choose == 2)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 80;
		}
		if (map_choose == 3)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 45;
		}
		if (map_choose == 4)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 80;
		}
		init_pass = 1;
	}
	//eraser.SetMovingLeft(false);
	int *getreturn;
	/*int AttackPower = 0;
	int CostPower = 0;
	int DEF = 0;
	int Effect = 0;*/

	if (c_charter1.x == 0 && ((c1_isActed == true && c2isActed2 == true && c1isActed2 == false && c2isActed == false) ||
(c1_isActed == false && c2isActed2 == false && c1isActed2 == true && c2isActed == true)))
		//if (c_charter1.x == 0)
	{
		if (c_card1.round % 2 == 0)
		{
			getreturn = c_card1.OnLButtonUp(nFlags, point, c_charter1.pow);
			AttackPower = getreturn[0]; //血
			judgeAttackPower = AttackPower;
			CostPower = getreturn[1];	//花費
			DEF = getreturn[2];		//盾
			judgeShield = DEF;
			Effect = getreturn[3];		//影響
		}
		/*else
		{
			//getreturn = c_card1.OnLButtonUp(nFlags, point, c_charter2.pow);
			AttackPower = 0; //血
			judgeAttackPower = 0;
			CostPower = 0;	//花費
			DEF = 0;		//盾
			Effect = 0;		//影響
		}*/

		/*AttackPower = getreturn[0]; //血
		judgeAttackPower = AttackPower;
		CostPower = getreturn[1];	//花費
		DEF = getreturn[2];		//盾
		Effect = getreturn[3];		//影響*/
		/**/

		if (oldround != c_card1.round)
		{
			if (countc1round > 0) { countc1round--; }
			else { c_charter1.getlife = 0; }
			if (countc2round > 0) { countc2round--; }
			else { c_charter2.getlife = 0; }
			if (c_card1.round % 2 == 0)
			{
				if (c_charter2.fire >= 0)
				{
					c_charter2.blood -= c_charter2.fire;
					c_charter2.fire *= 2;
				}
				/*if (c_charter1.getlife != 0)
				{
					c_charter1.getlife = 0;
				}*/
				oldround = c_card1.round;
			}
			else if (c_card1.round % 2 == 1)
			{
				if (c_charter1.fire > 0)
				{
					c_charter1.blood -= c_charter1.fire;
					c_charter1.fire *= 2;
				}
				/*if (c_charter2.getlife != 0)
				{
					c_charter2.getlife = 0;
				}*/
				oldround = c_card1.round;
			}
			else
			{
				oldround = c_card1.round;
			}
		}

		if (CostPower == 777)
		{
			c_charter1.pow = 3;
			c_charter2.pow = 3;
		}
		/**/
		else if ((c_card1.round % 2 == 0 && c_charter1.pow >= CostPower) || (c_card1.round % 2 == 1 && c_charter2.pow
>= CostPower))
		{
			if (c_card1.round % 2 == 0)
			{
				c_charter1.pow -= CostPower;
				/**/
				if (Effect == 1)
				{
					AttackPower += c_charter1.shield;
					judgeAttackPower = AttackPower;
				}

				if (AttackPower > 0) //攻擊
				{
					c1isActed2 = false;
					c2isActed = false;
					c1count2 = 0;
					c2count = 0;
					if (Effect == 6)
					{
						c_charter2.blood -= AttackPower;
					}
					else if (c_charter2.shield > 0)
					{
						if (c_charter2.shield > AttackPower)
						{
							c_charter2.shield -= AttackPower;
						}
						else
						{
							AttackPower -= c_charter2.shield;
							c_charter2.shield = 0;
							c_charter2.blood -= AttackPower;
						}
					}
					else if (c_charter2.shield <= 0)
					{
						c_charter2.blood -= AttackPower;
					}
				}
				else if (AttackPower < 0) //補血
				{
```

```
                    if (c_charter1.getlife == 0)
                    {
                        c_charter1.blood -= AttackPower;
                    }

                }

                if (DEF > 0)
                {
                    c_charter1.shield += DEF;
                }

                if (Effect == 2)
                {
                    c_charter1.fire = 0;
                }
                else if (Effect == 3)
                {
                    if (c_charter2.fire <= 0)
                    {
                        c_charter2.fire++;
                    }
                }
                else if (Effect == 4)
                {
                    c_charter1.getlife = 1;
                    countc1round += 2;
                }
                else if (Effect == 5)
                {
                    c_charter2.getlife = 1;
                    countc2round += 1;
                }
            }
        }
        c_card1.onShow();
    }


}

void CGameStateCo::OnMouseMove(UINT nFlags, CPoint point)  // 處理滑鼠的動作
{
    // 沒事。如果需要處理滑鼠移動的話，寫 code 在這裡
}

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

void CGameStateCo::OnRButtonUp(UINT nFlags, CPoint point)  // 處理滑鼠的動作
{
    //
}
void CGameStateCo::Player2Action()
{
    if ((c1_isActed == true && c2isActed == false) || Player2isEnd == 1)
    {
        AttackPower = 0; //血
        judgeAttackPower = AttackPower;
        CostPower = 0;     //花費
        DEF = 0;                //盾
        Effect = 0;          //影響
        Player2isEnd = 1;
        c_card1.round += 1;
        c_card1.c_rand = (rand() % 9);
        c_card1.c_rand2 = (rand() % 9);
        c_card1.c_rand3 = (rand() % 9);
```

```
		c_card1.onShow();
		return;
	}
	if (c1_count != 0)
	{
		return;
	}
	unsigned seed = (unsigned)time(NULL); //random 種子宣告(以時間)
	srand(seed);
	int gete = rand() % 100;
	if (gete % 10 < 6)
	{
		if (gete % 10 < 2)
		{
			Effect = 3;
		}
		if (map_choose == 1 || map_choose == 2)
		{
			AttackPower = rand() % 15 + 1;
		}
		if (map_choose == 3 || map_choose == 4)
		{
			AttackPower = (rand() % 15 + 1)+5;
		}

	}
	else if (gete % 10 > 6)
	{
		if (gete % 10 > 8)
		{
			Effect = 2;
		}
		if (map_choose == 1 || map_choose == 2)
		{
			AttackPower = -1 * (rand() % 10) - 1;
		}
		if (map_choose == 1 || map_choose == 2)
		{
			AttackPower = -1 * (rand() % 10) - 6;
		}

	}
	else
	{
		if (map_choose == 1 || map_choose == 2)
		{
			c_charter2.shield += 5;
		}
		if (map_choose == 1 || map_choose == 2)
		{
			c_charter2.shield += 8;
		}

	}
	judgeAttackPower = AttackPower;

	c_charter2.pow = 0;
	//c_charter2.pow -= CostPower;
	/**/
	if (Effect == 1)
	{
		AttackPower += c_charter2.shield;
		judgeAttackPower = AttackPower;
	}

	if (AttackPower > 0) //攻擊
	{
		c1_isActed = false;
		c2isActed2 = false;
```

```
			c1_count = 0;
			c2count2 = 0;
			if (Effect == 6)
			{
				c_charter1.blood -= AttackPower;
			}
			else if (c_charter1.shield > 0)
			{
				if (c_charter1.shield > AttackPower)
				{
					c_charter1.shield -= AttackPower;
				}
				else
				{
					AttackPower -= c_charter1.shield;
					c_charter1.shield = 0;
					c_charter1.blood -= AttackPower;
				}
			}
			else if (c_charter1.shield <= 0)
			{
				c_charter1.blood -= AttackPower;
			}
		}
		else if (AttackPower < 0) //補血
		{
			if (c_charter2.getlife == 0)
			{
				c_charter2.blood -= AttackPower;
			}

		}

		if (DEF > 0)
		{
			judgeShield = DEF;
			c_charter2.shield += DEF;
		}

		if (Effect == 2)
		{
			c_charter2.fire = 0;
		}
		else if (Effect == 3)
		{
			if (c_charter1.fire <= 0)
			{
				c_charter1.fire++;
			}
		}
		else if (Effect == 4)
		{
			c_charter2.getlife = 1;
			countc2round += 2;
		}
		else if (Effect == 5)
		{
			c_charter1.getlife = 1;
			countc1round += 1;
		}
		c_card1.onShow();
		return;
}
void CGameStateCo::OnShow()
{
		//
			//  注意：Show 裡面千萬不要移動任何物件的座標，移動座標的工作應由 Move 做才對，
			//        否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語
			//        說，Move 負責 MVC 中的 Model，Show 負責 View，而 View 不應更動 Model。
```

```
        //
        //
        //  貼上背景圖、撞擊數、球、擦子、彈跳的球
        //
        //CMovingBitmap       isfire, arefire;
if (map_choose == 1) { BackGround[0].ShowBitmap(); }
else if (map_choose == 2) { BackGround[1].ShowBitmap(); }
else if (map_choose == 3) { BackGround[2].ShowBitmap(); }
else if (map_choose == 4) { BackGround[3].ShowBitmap(); }
else { BackGround[2].ShowBitmap(); }
CMovingBitmap       isfire, arefire;
CMovingBitmap       isban, areban;
CMovingBitmap shieldR, shieldR1, shieldR2;
if (c_charter1.blood < 0) { c_charter1.blood = 0; }
if (c_charter2.blood < 0) { c_charter2.blood = 0; }
if (c_charter1.x >= 0)
{
        bloodline_c1.SetTopLeft(201, 160);
        bloodline_c1.ShowBitmap();
        {
                CDC *pDC = CDDraw::GetBackCDC();                        // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32];
                stringstream ss;
                ss << c_charter1.blood;
                ss >> blood_char;
                f.CreatePointFont(200, "Times New Roman");       // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                              // 選用 font f
                pDC->SetTextColor(RGB(255, 255, 255));
                pDC->SetBkColor(RGB(255, 0, 0));
                pDC->TextOut(400, 160, blood_char);
                pDC->SelectObject(fp);                                           // 放掉 font f (千萬不要漏了放掉)
                CDDraw::ReleaseBackCDC();                                        // 放掉 Back Plain 的 CDC
        }
        int bx = 201;
        for (int i = 0; i < c_charter1.blood; i++)
        {
                if (i == 28)
                {
                        bloodpoint_c1[29].SetTopLeft(201, 160);
                        bloodpoint_c1[29].ShowBitmap();
                }
                else if (i == 0)
                {
                        bloodpoint_c1[0].SetTopLeft(201, 160);
                        bloodpoint_c1[0].ShowBitmap();
                }
                else if (i < 28)
                {
                        bloodpoint_c1[i].SetTopLeft(bx, 160);
                        bloodpoint_c1[i].ShowBitmap();
                        bx += 6;
                }
        }
        blood_c1.SetTopLeft(200, 160);
        blood_c1.ShowBitmap();
        shieldline_c1.SetTopLeft(200, 200);
        shieldline_c1.ShowBitmap();
        bx = 200;
        {
                CDC *pDC = CDDraw::GetBackCDC();                        // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char shield_char[32];
                stringstream ss;
                ss << c_charter1.shield;
                ss >> shield_char;
                f.CreatePointFont(200, "Times New Roman");       // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                              // 選用 font f
                pDC->SetTextColor(RGB(255, 255, 255));
        pDC->SetBkColor(RGB(0, 255, 0));
        pDC->TextOut(400, 200, shield_char);
        pDC->SelectObject(fp);                                  // 放掉 font f (千萬不要漏了放掉)
        CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
}
for (int i = 0; i < c_charter1.shield; i++)
{
        if (i == 28)
        {
                shieldpoint_c1[29].SetTopLeft(200, 200);
                shieldpoint_c1[29].ShowBitmap();
        }
        else if (i == 0)
        {
                shieldpoint_c1[0].SetTopLeft(200, 200);
                shieldpoint_c1[0].ShowBitmap();
        }
        else if (i < 28)
        {
                shieldpoint_c1[i].SetTopLeft(bx, 200);
                shieldpoint_c1[i].ShowBitmap();
                bx += 6;
        }
}
shield_c1.SetTopLeft(200, 200);
shield_c1.ShowBitmap();
bx = 100;
for (int i = 0; i < c_charter1.pow; i++)
{
        usedollor_c1[i].SetTopLeft(bx, 800);
        usedollor_c1[i].ShowBitmap();
        bx += 70;
}
bloodline_c2.SetTopLeft(1251, 160);
bloodline_c2.ShowBitmap();
bx = 1251;
{
        CDC *pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
        CFont f, *fp;
        char blood_char[32];
        stringstream ss;
        ss << c_charter2.blood;
        ss >> blood_char;
        f.CreatePointFont(200, "Times New Roman");       // 產生 font f; 160 表示 16 point 的字
        fp = pDC->SelectObject(&f);                         // 選用 font f
        pDC->SetTextColor(RGB(255, 255, 255));
        pDC->SetBkColor(RGB(255, 0, 0));
        pDC->TextOut(1450, 160, blood_char);
        pDC->SelectObject(fp);                                  // 放掉 font f (千萬不要漏了放掉)
        CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
}
for (int i = 0; i < c_charter2.blood; i++)
{
        if (i == 28)
        {
                bloodpoint_c2[29].SetTopLeft(1251, 160);
                bloodpoint_c2[29].ShowBitmap();
        }
        else if (i == 0)
        {
                bloodpoint_c2[0].SetTopLeft(1251, 160);
                bloodpoint_c2[0].ShowBitmap();
        }
        else if (i < 28)
        {
                bloodpoint_c2[i].SetTopLeft(bx, 160);
                bloodpoint_c2[i].ShowBitmap();
                bx += 6;
        }
        }
        blood_c2.SetTopLeft(1250, 160);
        blood_c2.ShowBitmap();
        shieldline_c1.SetTopLeft(1250, 200);
        shieldline_c1.ShowBitmap();
        bx = 1250;
        {
            CDC *pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
            CFont f, *fp;
            char shield_char[32];
            stringstream ss;
            ss << c_charter2.shield;
            ss >> shield_char;
            f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
            fp = pDC->SelectObject(&f);                         // 選用 font f
            pDC->SetTextColor(RGB(255, 255, 255));
            pDC->SetBkColor(RGB(0, 255, 0));
            pDC->TextOut(1450, 200, shield_char);
            pDC->SelectObject(fp);                                  // 放掉 font f (千萬不要漏了放掉)
            CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
        }
        for (int i = 0; i < c_charter2.shield; i++)
        {
            if (i == 28)
            {
                shieldpoint_c2[29].SetTopLeft(1250, 200);
                shieldpoint_c2[29].ShowBitmap();
            }
            else if (i == 0)
            {
                shieldpoint_c2[0].SetTopLeft(1250, 200);
                shieldpoint_c2[0].ShowBitmap();
            }
            else if (i < 28)
            {
                shieldpoint_c2[i].SetTopLeft(bx, 200);
                shieldpoint_c2[i].ShowBitmap();
                bx += 6;
            }
        }
        shield_c2.SetTopLeft(1250, 200);
        shield_c2.ShowBitmap();
        bx = 1150;
        for (int i = 0; i < c_charter2.pow; i++)
        {
            usedollor_c2[i].SetTopLeft(bx, 800);
            usedollor_c2[i].ShowBitmap();
            bx += 70;
        }
    }
    c_charter2.OnMove();
    c_charter1.OnMove();
    if (c_card1.round % 2 == 1)
    {
        Player2Action();
        if (c1_count == 0) c1_count++; //c1_isActed = false;
        //if (judgeAttackPower > 0)
        if (judgeAttackPower > 0 && c1_isActed == false && c2count2 >= 16)
        {
            c1count2 = 0;
            c1isActed2 = false;
            if (c1_count == 1)c_charter1.pic = c_charter1.attackedpic[0];
            if (c1_count == 4)c_charter1.pic = c_charter1.attackedpic[1];
            if (c1_count == 8)c_charter1.pic = c_charter1.attackedpic[2];
            if (c1_count == 12)c_charter1.pic = c_charter1.attackedpic[3];
            if (c1_count == 16) { c_charter1.pic = c_charter1.attackedpic[4]; c1_isActed = true; Player2isEnd = 1; }
            c1_count++;
            //if (c1_count >= 16) { c1_count = 0; }
        }
```

```
            else if (judgeAttackPower <= 0 && c1_isActed == false)
            {
                        c1_isActed = true;
                        c1count2 = 0;
                        c1isActed2 = false;
                        Player2isEnd = 1;
            }
            if (c2count2 == 0) c2count2++; //c2isActed2 = false;
            //if (judgeAttackPower > 0)
            if (judgeAttackPower > 0 && c2isActed2 == false)
            {
                        c2count = 0;
                        c2isActed = false;
                        if (c2count2 == 1)c_charter2.pic = c_charter2.attackpic[0];
                        if (c2count2 == 4)c_charter2.pic = c_charter2.attackpic[1];
                        if (c2count2 == 8)c_charter2.pic = c_charter2.attackpic[2];
                        if (c2count2 == 12)c_charter2.pic = c_charter2.attackpic[3];
                        if (c2count2 == 16) { c_charter2.pic = c_charter2.attackpic[4]; c2isActed2 = true; }
                        c2count2++;
                        //if (c2count2 >= 16) { c2count2 = 0; }
            }
            else if (judgeAttackPower <= 0 && c2isActed2 == false)
            {
                        c2isActed2 = true;
                        c2count = 0;
                        c2isActed = false;
                        Player2isEnd = 1;
            }
}
if (c_card1.round % 2 == 0)
{
            if (c1count2 == 0) c1count2++; //c1isActed2 = false;
            //if (judgeAttackPower > 0)
            if (judgeAttackPower > 0 && c1isActed2 == false)
            {
                        c1_count = 0;
                        c1_isActed = false;
                        if (c1count2 == 1) { c_charter1.pic = c_charter1.attackpic[0]; }
                        if (c1count2 == 4) { c_charter1.pic = c_charter1.attackpic[1]; }
                        if (c1count2 == 8) { c_charter1.pic = c_charter1.attackpic[2]; }
                        if (c1count2 == 12) { c_charter1.pic = c_charter1.attackpic[3]; }
                        if (c1count2 == 16) { c_charter1.pic = c_charter1.attackpic[4]; }
                        if (c1count2 == 20) { c_charter1.pic = c_charter1.attackpic[5]; }
                        if (c1count2 == 24) { c_charter1.pic = c_charter1.attackpic[6]; c1isActed2 = true; }
                        c1count2++;
                        //if (c1count2 >= 24) { c1count2 = 0; }
            }
            else if (judgeAttackPower <= 0 && c1isActed2 == false)
            {
                        c1_count = 0;
                        c1_isActed = false;
                        c1isActed2 = true;
            }
            if (c2count == 0) c2count++; //c2isActed = false;
            //if (judgeAttackPower > 0)
            if (judgeAttackPower > 0 && c2isActed == false && c1count2 >= 24)
            {
                        c2count2 = 0;
                        c2isActed2 = false;
                        if (c2count == 1)c_charter2.pic = c_charter2.attackedpic[0];
                        if (c2count == 4)c_charter2.pic = c_charter2.attackedpic[1];
                        if (c2count == 8)c_charter2.pic = c_charter2.attackedpic[2];
                        if (c2count == 12)c_charter2.pic = c_charter2.attackedpic[3];
                        if (c2count == 16) { c_charter2.pic = c_charter2.attackedpic[4]; c2isActed = true; }
                        c2count++;
                        //if (c2count >= 16) { c2count = 0; }
            }
            else if (judgeAttackPower <= 0 && c2isActed == false)
            {
```

```
                        c2count2 = 0;
                        c2isActed2 = false;
                        c2isActed = true;
                }
        }
        /**/
        if ((c_card1.round % 2 == 1 && judgeAttackPower > 0 && c1_isActed == false && c2count2 >= 16) || (c_card1.round %
2 == 1 && judgeAttackPower < 0)) {
                {
                        CDC *pDC = CDDraw::GetBackCDC();                  // 取得 Back Plain 的 CDC
                        CFont f, *fp;
                        char blood_char[32];
                        int myattack = 0;
                        stringstream ss;
                        myattack = -1 * judgeAttackPower;
                        if (myattack > 0)
                        {
                                ss << "+";
                        }
                        ss << myattack;
                        ss >> blood_char;
                        f.CreatePointFont(200, "Times New Roman");       // 產生 font f; 160 表示 16 point 的字
                        fp = pDC->SelectObject(&f);                       // 選用 font f
                        pDC->SetTextColor(RGB(255, 255, 255));
                        pDC->SetBkColor(RGB(255, 0, 0));

                        if (judgeAttackPower < 0)
                        {
                                for (int i = 0; i < 6000; i++) {
                                        pDC->TextOut(1000, 250, blood_char);

                                }
                                judgeAttackPower = 0;
                        }
                        else {
                                pDC->TextOut(500, 250, blood_char);
                        }
                        pDC->SelectObject(fp);                                    // 放掉 font f (千萬不要漏了放掉)
                        CDDraw::ReleaseBackCDC();                                 // 放掉 Back Plain 的 CDC
                }
        }
        if (c_card1.round % 2 == 1 && judgeShield > 0) {
                {
                        CDC *pDC = CDDraw::GetBackCDC();                  // 取得 Back Plain 的 CDC
                        CFont f, *fp;
                        char blood_char[32];
                        stringstream ss;
                        ss << "+";
                        ss << judgeShield;
                        ss >> blood_char;
                        f.CreatePointFont(200, "Times New Roman");       // 產生 font f; 160 表示 16 point 的字
                        fp = pDC->SelectObject(&f);                       // 選用 font f
                        pDC->SetTextColor(RGB(255, 255, 255));
                        pDC->SetBkColor(RGB(0, 255, 0));
                        for (int i = 0; i < 6000; i++) {
                                pDC->TextOut(1000, 250, blood_char);
                        }
                        judgeShield = 0;
                        pDC->SelectObject(fp);                                    // 放掉 font f (千萬不要漏了放掉)
                        CDDraw::ReleaseBackCDC();                                 // 放掉 Back Plain 的 CDC
                }
        }
        if ((c_card1.round % 2 == 0 && judgeAttackPower > 0 && c2isActed == false && c1count2 >= 24) || (c_card1.round %
2 == 0 && judgeAttackPower < 0)) {
                {
                        CDC *pDC = CDDraw::GetBackCDC();                  // 取得 Back Plain 的 CDC
                        CFont f, *fp;
                        char blood_char[32];
                        int myattack = 0;
```

```
				stringstream ss;
				myattack = -1 * judgeAttackPower;
				if (myattack > 0)
				{
					ss << "+";
				}
				ss << myattack;
				ss >> blood_char;
				f.CreatePointFont(200, "Times New Roman");		// 產生 font f; 160 表示 16 point 的字
				fp = pDC->SelectObject(&f);					// 選用 font f
				pDC->SetTextColor(RGB(255, 255, 255));
				pDC->SetBkColor(RGB(255, 0, 0));

				if (judgeAttackPower < 0)
				{
					for (int i = 0; i < 6000; i++) {
						pDC->TextOut(500, 250, blood_char);
					}
					judgeAttackPower = 0;
				}
				else {

					pDC->TextOut(1000, 250, blood_char);
				}
				pDC->SelectObject(fp);						// 放掉 font f (千萬不要漏了放掉)
				CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC
			}
		}
		if (c_card1.round % 2 == 0 && judgeShield > 0) {
			{
				CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
				CFont f, *fp;
				char blood_char[32];
				stringstream ss;
				ss << "+";
				ss << judgeShield;
				ss >> blood_char;
				f.CreatePointFont(200, "Times New Roman");		// 產生 font f; 160 表示 16 point 的字
				fp = pDC->SelectObject(&f);					// 選用 font f
				pDC->SetTextColor(RGB(255, 255, 255));
				pDC->SetBkColor(RGB(0, 255, 0));
				for (int i = 0; i < 6000; i++) {
					pDC->TextOut(500, 250, blood_char);
				}
				judgeShield = 0;
				pDC->SelectObject(fp);						// 放掉 font f (千萬不要漏了放掉)
				CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC
			}
		}
		/**/
		c_charter1.onShow();
		c_charter2.onShow();
		endturn.SetTopLeft(600, 850);
		endturn.ShowBitmap();
		/**/
		if (c_charter1.fire > 0)
		{
			CDC *pDC1 = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
			CFont f1, *fp1;
			stringstream ss1;
			ss1 << c_charter1.fire*-1;
			char fire1[32] = "{ CHARTER 1 }";
			ss1 >> fire1;
			f1.CreatePointFont(200, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
			fp1 = pDC1->SelectObject(&f1);				// 選用 font f
			pDC1->SetTextColor(RGB(255, 0, 0));
			pDC1->SetBkColor(RGB(255, 255, 255));
			pDC1->TextOut(290, 30, fire1);
			pDC1->SelectObject(fp1);					// 放掉 font f (千萬不要漏了放掉)
```

```
        CDDraw::ReleaseBackCDC();                              // 放掉 Back Plain 的 CDC
        {
                CDC *pDC = CDDraw::GetBackCDC();               // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32] = "     燙傷: 下回血量";
                int myattack = 0;
                f.CreatePointFont(200, "Times New Roman");     // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                    // 選用 font f
                pDC->SetTextColor(RGB(255, 0, 0));
                pDC->SetBkColor(RGB(255, 255, 255));
                pDC->TextOut(50, 30, blood_char);
                pDC->SelectObject(fp);                                  // 放掉 font f(千萬不要漏了放掉)
                CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
        }
        isfire.LoadBitmapA("Bitmaps\\FireMini.bmp", RGB(50, 150, 150));
        isfire.SetTopLeft(12, 12);
        isfire.ShowBitmap();
}
if (c_charter2.fire > 0)
{
        {
                CDC *pDC = CDDraw::GetBackCDC();               // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32] = "             .";
                int myattack = 0;
                f.CreatePointFont(200, "Times New Roman");     // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                    // 選用 font f
                pDC->SetTextColor(RGB(255, 0, 0));
                pDC->SetBkColor(RGB(255, 255, 255));
                pDC->TextOut(1550, 30, blood_char);
                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;
        stringstream ss1;
        ss1 << c_charter2.fire*-1 << "    ";
        char fire1[32] = "{ CHARTER 1 }";
        ss1 >> fire1;
        f1.CreatePointFont(200, "Times New Roman");    // 產生 font f; 160 表示 16 point 的字
        fp1 = pDC1->SelectObject(&f1);                         // 選用 font f
        pDC1->SetTextColor(RGB(255, 0, 0));
        pDC1->SetBkColor(RGB(255, 255, 255));
        pDC1->TextOut(1560, 30, fire1);
        pDC1->SelectObject(fp1);                               // 放掉 font f(千萬不要漏了放掉)
        CDDraw::ReleaseBackCDC();                              // 放掉 Back Plain 的 CDC
        {
                CDC *pDC = CDDraw::GetBackCDC();               // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32] = "燙傷: 下回血量";
                int myattack = 0;
                f.CreatePointFont(200, "Times New Roman");     // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                    // 選用 font f
                pDC->SetTextColor(RGB(255, 0, 0));
                pDC->SetBkColor(RGB(255, 255, 255));
                pDC->TextOut(1340, 30, blood_char);
                pDC->SelectObject(fp);                                  // 放掉 font f(千萬不要漏了放掉)
                CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
        }
        arefire.LoadBitmapA("Bitmaps\\FireMini.bmp", RGB(50, 150, 150));
        arefire.SetTopLeft(1600, 12);
        arefire.ShowBitmap();
}
if (c_charter1.getlife != 0)
{
        {
                CDC *pDC = CDDraw::GetBackCDC();               // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32] = "     禁止補血";
```

```
                    int myattack = 0;
                    f.CreatePointFont(200, "Times New Roman");        // 產生 font f; 160 表示 16 point 的字
                    fp = pDC->SelectObject(&f);                               // 選用 font f
                    pDC->SetTextColor(RGB(255, 0, 0));
                    pDC->SetBkColor(RGB(255, 255, 255));
                    pDC->TextOut(50, 118, blood_char);
                    pDC->SelectObject(fp);                                           // 放掉 font f (千萬不要漏了放掉)
                    CDDraw::ReleaseBackCDC();                                    // 放掉 Back Plain 的 CDC
          }
          isban.LoadBitmapA("Bitmaps\\banaddbloodmini.bmp", RGB(50, 150, 150));
          isban.SetTopLeft(12, 100);
          isban.ShowBitmap();
}
if (c_charter2.getlife != 0)
{
          {
                    CDC *pDC = CDDraw::GetBackCDC();                      // 取得 Back Plain 的 CDC
                    CFont f, *fp;
                    char blood_char[32] = "禁止補血   ";
                    int myattack = 0;
                    f.CreatePointFont(200, "Times New Roman");        // 產生 font f; 160 表示 16 point 的字
                    fp = pDC->SelectObject(&f);                               // 選用 font f
                    pDC->SetTextColor(RGB(255, 0, 0));
                    pDC->SetBkColor(RGB(255, 255, 255));
                    pDC->TextOut(1465, 118, blood_char);
                    pDC->SelectObject(fp);                                           // 放掉 font f (千萬不要漏了放掉)
                    CDDraw::ReleaseBackCDC();                                    // 放掉 Back Plain 的 CDC
          }
          areban.LoadBitmapA("Bitmaps\\banaddbloodmini.bmp", RGB(50, 150, 150));
          areban.SetTopLeft(1600, 100);
          areban.ShowBitmap();
}

/**/
c_card1.onShow();
/*if (c_card1.round%2==1)
{
          Player2Action();
}*/
/*BackGround.ShowBitmap();
c_charter2.OnMove();
c_charter1.OnMove();
if (c_card1.round % 2 == 1)
{
          if (AttackPower > 0 && c1_isActed == false)
          {
                    if (c1_count == 0) { c_charter1.pic = c_charter1.attackedpic[0];}
                    if (c1_count == 4) { c_charter1.pic = c_charter1.attackedpic[1]; }
                    if (c1_count == 8) { c_charter1.pic = c_charter1.attackedpic[2]; }
                    if (c1_count == 12) { c_charter1.pic = c_charter1.attackedpic[3]; }
                    if (c1_count == 16) { c_charter1.pic = c_charter1.attackedpic[4]; c1_isActed = true; }
                    c1_count++;
          }
}
if (c_card1.round % 2 == 0)
{
          c1_count = 0;
          c1_isActed = false;
}
c_charter1.onShow();
c_charter2.onShow();
c_card1.onShow();*/
//  貼上左上及右下角落的圖
//
/*顯示回合*/
if (c_card1.round % 2 == 0)
{
          showRound[0].SetTopLeft(600, 50);
          showRound[0].ShowBitmap();
```

```
			/*CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
			CFont f, *fp;
			char me[32] = "{ CHARTER 1 }";
			f.CreatePointFont(360, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
			fp = pDC->SelectObject(&f);					// 選用 font f
			pDC->SetTextColor(RGB(255, 255, 255));
			pDC->SetBkColor(RGB(255, 0, 0));
			pDC->TextOut(600, 850, me);
			pDC->SelectObject(fp);						// 放掉 font f (千萬不要漏了放掉)
			CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC*/
		}
		else
		{
			showRound[1].SetTopLeft(600, 50);
			showRound[1].ShowBitmap();
			/*
			CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
			CFont f, *fp;
			char me[32] = "{ CHARTER 2 }";
			f.CreatePointFont(360, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
			fp = pDC->SelectObject(&f);					// 選用 font f
			pDC->SetTextColor(RGB(255, 255, 255));
			pDC->SetBkColor(RGB(0, 255, 0));
			pDC->TextOut(600, 850, me);
			pDC->SelectObject(fp);						// 放掉 font f (千萬不要漏了放掉)
			CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC*/
		}
		/**/
		/*if (c_charter2.pow <= 0 && ((c1_isActed == true && c2isActed2 == true && c1isActed2 == false && c2isActed ==
	false) || (c1_isActed == false && c2isActed2 == false && c1isActed2 == true && c2isActed == true)))
		{
		}*/

		if ((c_charter2.blood <= 0 || c_charter1.blood <= 0) && ((c1_isActed == true && c2isActed2 == true && c1isActed2 ==
	false && c2isActed == false) || (c1_isActed == false && c2isActed2 == false && c1isActed2 == true && c2isActed == true)))
		{
			if (c_charter1.blood <= 0)
			{
				get_win = 2;
			}
			else if (c_charter2.blood <= 0)
			{
				get_win = 1;
			}
			else
			{
				get_win = 0;
			}
			c_charter1.blood = 30;
			c_charter2.blood = 30;
			c_charter1.shield = 0;
			c_charter2.shield = 0;
			c_charter1.getlife = 0;
			c_charter2.getlife = 0;
			c_charter1.fire = 0;
			c_charter2.fire = 0;
			c_card1.round = 0;
			c_charter1.x = -580;
			c_charter2.x = 1680;
			AttackPower = 0;
			judgeAttackPower = 0;
			CostPower = 0;
			DEF = 0;
			Effect = 0;
			c1_count = 16;
			c1count2 = 0;
			c1isActed2 = false;
			c1_isActed = true;
			c2count = 0;
```

```
            c2count2 = 16;
            c2isActed2 = true;
            c2isActed = false;
            oldround = -1;
            countc1round = 0;
            countc2round = 0;
            c_card1.onShow();
            unsigned seed = (unsigned)time(NULL); //random 種子宣告(以時間)
            srand(seed); //用時間取亂數
            c_card1.c_rand = (rand() % 9);
            c_card1.c_rand2 = (rand() % 9);
            c_card1.c_rand3 = (rand() % 9);
            GotoGameState(GAME_STATE_WIN);
        }
}

    //角色 1
    Charter1::Charter1()
    {
            x = -580;
            y = 170;
            this->blood = 45;
            this->shield = 0;
            this->fire = 0;
            this->getlife = 0;
            this->pow = 3;
    }
    void Charter1::OnMove()
    {
            if (x < 0) {
                    x += 5;
            }
            else {
                    x = 0;
            }
            Sleep(1);
    }
    void Charter1::LoadBit()
    {
            pic.LoadBitmap("Bitmaps\\Forward.bmp", RGB(0, 0, 0));
            attackedpic[0].LoadBitmap("Bitmaps\\attacked1.bmp", RGB(0, 0, 0));
            attackedpic[1].LoadBitmap("Bitmaps\\attacked2.bmp", RGB(0, 0, 0));
            attackedpic[2].LoadBitmap("Bitmaps\\attacked3.bmp", RGB(0, 0, 0));
            attackedpic[3].LoadBitmap("Bitmaps\\attacked4.bmp", RGB(0, 0, 0));
            attackedpic[4].LoadBitmap("Bitmaps\\Forward.bmp", RGB(0, 0, 0));
            attackpic[0].LoadBitmap("Bitmaps\\attack1.bmp", RGB(0, 0, 0));
            attackpic[1].LoadBitmap("Bitmaps\\attack2.bmp", RGB(0, 0, 0));
            attackpic[2].LoadBitmap("Bitmaps\\attack3.bmp", RGB(0, 0, 0));
            attackpic[3].LoadBitmap("Bitmaps\\attack4.bmp", RGB(0, 0, 0));
            attackpic[4].LoadBitmap("Bitmaps\\attack5.bmp", RGB(0, 0, 0));
            attackpic[5].LoadBitmap("Bitmaps\\attack6.bmp", RGB(0, 0, 0));
            attackpic[6].LoadBitmap("Bitmaps\\Forward.bmp", RGB(0, 0, 0));
            this->blood = 45;
            this->shield = 0;
    }
    void Charter1::AttackedMove()
    {
            if (x > -100) {
                    x -= 5;
                    Sleep(1);
            }
            else {
                    x = 0;
            }

    }
    void Charter1::onShow()
    {
            /*CDC *pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
```

```
			CFont f, *fp;
			char blood_char[32] = "                    ";
			char shield_char[32] = "                    ";
			char pow_char[32] = "                    ";
			stringstream ss;
			ss << this->blood;
			ss >> blood_char;
			f.CreatePointFont(360, "Times New Roman");		// 產生 font f; 160 表示 16 point 的字
			fp = pDC->SelectObject(&f);					// 選用 font f
			//pDC->SetBkColor(RGB(255, 255, 255));
			//pDC->SetTextColor(RGB(0, 0, 0));
			//
			if (blood <= 5)
			{
				pDC->SetTextColor(RGB(255, 255, 255));
				pDC->SetBkColor(RGB(255, 0, 0));
			}
			else
			{
				pDC->SetTextColor(RGB(255, 0, 0));
				pDC->SetBkColor(RGB(255, 255, 255));
			}
			//
			pDC->TextOut(750, 40, blood_char);
			ss.clear();
			ss << this->shield;
			ss >> shield_char;
			if (shield <= 5)
			{
				pDC->SetTextColor(RGB(255, 255, 255));
				pDC->SetBkColor(RGB(0, 255, 0));
			}
			else
			{
				pDC->SetTextColor(RGB(0, 255, 0));
				pDC->SetBkColor(RGB(255, 255, 255));
			}
			pDC->TextOut(650, 40, shield_char);
			ss.clear();
			ss << this->pow;
			ss >> pow_char;
			pDC->SetTextColor(RGB(0, 255, 0));
			pDC->SetBkColor(RGB(255, 255, 255));
			pDC->TextOut(200, 200, pow_char);
			pDC->SelectObject(fp);						// 放掉 font f (千萬不要漏了放掉)
			CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC*/
			pic.SetTopLeft(x, y);
			pic.ShowBitmap();
		}
		/**/
	//
	//角色 2
		Charter2::Charter2()
		{
			x = 1680;
			y = 150;
			this->blood = 45;
			this->shield = 0;
			this->fire = 0;
			this->getlife = 0;
			this->pow = 3;
		}
		void Charter2::OnMove()
		{
			if (x > 1100) {
				x -= 5;
			}
			else {
				x = 1100;
		}
		Sleep(1);
}
void Charter2::AttackedMove()
{
		if (x < 1300) {
			x += 5;
			Sleep(1);
		}
		else {
			x = 1100;
		}

}
void Charter2::LoadBit()
{
		this->blood = 45;
		this->shield = 0;
		//pic.LoadBitmap("Bitmaps\\forward3.bmp",RGB(0,0,255));
		pic.LoadBitmap("Bitmaps\\newnew.bmp", RGB(0, 0, 255));
		attackedpic[0].LoadBitmap("Bitmaps\\attacked21.bmp", RGB(0, 0, 255));
		attackedpic[1].LoadBitmap("Bitmaps\\attacked22.bmp", RGB(0, 0, 255));
		attackedpic[2].LoadBitmap("Bitmaps\\attacked23.bmp", RGB(0, 0, 255));
		attackedpic[3].LoadBitmap("Bitmaps\\attacked24.bmp", RGB(0, 0, 255));
		attackedpic[4].LoadBitmap("Bitmaps\\newnew.bmp", RGB(0, 0, 255));
		attackpic[0].LoadBitmap("Bitmaps\\attack21.bmp", RGB(0, 0, 255));
		attackpic[1].LoadBitmap("Bitmaps\\attack22.bmp", RGB(0, 0, 255));
		attackpic[2].LoadBitmap("Bitmaps\\attack23.bmp", RGB(0, 0, 255));
		attackpic[3].LoadBitmap("Bitmaps\\attack24.bmp", RGB(0, 0, 255));
		attackpic[4].LoadBitmap("Bitmaps\\newnew.bmp", RGB(0, 0, 255));

}
void Charter2::onShow()
{

		/*CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
		CFont f, *fp;
		char blood_char[32] = "                              ";
		char shield_char[32] = "                              ";
		char pow_char[32] = "                              ";
		//char blood_line[32] = "                              ";
		stringstream ss;
		ss << this->blood;
		ss >> blood_char;
		f.CreatePointFont(360, "Times New Roman");		// 產生 font f; 160 表示 16 point 的字
		fp = pDC->SelectObject(&f);				// 選用 font f
		//pDC->SetBkColor(RGB(255, 255, 255));
		/*blood_line[0] = '(';
		blood_line[blood + 1] = ')';
		for (int i = 1; i <= blood; i++)
		{
			blood_line[i] = 'X';
		}
		if (blood>0)
		{
			pDC->SetTextColor(RGB(255, 0, 0));
		}
		else
		{
			pDC->SetTextColor(RGB(0, 0, 0));
		}
		//
		if (blood <= 5)
		{
			pDC->SetTextColor(RGB(255, 255, 255));
			pDC->SetBkColor(RGB(255, 0, 0));
		}
		else
		{
			pDC->SetTextColor(RGB(255, 0, 0));
			pDC->SetBkColor(RGB(255, 255, 255));
		}
		//
		pDC->TextOut(854, 40, blood_char);
		//pDC->TextOut(950, 15, blood_line);
		ss.clear();
		ss << this->shield;
		ss >> shield_char;
		if (shield <= 5)
		{
			pDC->SetTextColor(RGB(255, 255, 255));
			pDC->SetBkColor(RGB(0, 255, 0));
		}
		else
		{
			pDC->SetTextColor(RGB(0, 255, 0));
			pDC->SetBkColor(RGB(255, 255, 255));
		}
		pDC->TextOut(954, 40, shield_char);
		ss.clear();
		ss << this->pow;
		ss >> pow_char;
		pDC->SetTextColor(RGB(0, 255, 0));
		pDC->SetBkColor(RGB(255, 255, 255));
		pDC->TextOut(1100, 200, pow_char);
		pDC->SelectObject(fp);                                // 放掉 font f(千萬不要漏了放掉)
		CDDraw::ReleaseBackCDC();                             // 放掉 Back Plain 的 CDC*/
		pic.SetTopLeft(x, y);
		pic.ShowBitmap();
}
//
/*卡牌資訊*/
Card_info::Card_info()
{
		name = "";
		AttackPower = 0;
}
/**/
//
//卡牌
Card1::Card1()
{
		//srand(5);//我不知道怎麼用現在時間當亂數
		unsigned seed = (unsigned)time(NULL); //random 種子宣告(以時間)
		srand(seed); //用時間取亂數
		x = 440;
		y = 440;
		c_rand = (rand() % 9);
		c_rand2 = (rand() % 9);
		c_rand3 = (rand() % 9);


}

void Card1::LoadBit()
{
		/*卡牌資料*/
		theCard[10].name = "1R_1";
		theCard[10].CostPower = 1;
		theCard[10].DEF = 0;
		theCard[10].AttackPower = 5;
		theCard[10].Effect = 0;
		theCard[10].todo.LoadBitmapA("Bitmaps\\1R_1.bmp", RGB(50, 150, 150));
		theCard[11].name = "1R_2";
		theCard[11].CostPower = 1;
		theCard[11].DEF = 0;
		theCard[11].AttackPower = 0;
		theCard[11].Effect = 1;//attack = def
```

```
theCard[11].todo.LoadBitmapA("Bitmaps\\1R_2.bmp", RGB(50, 150, 150));
theCard[12].name = "1R_3";
theCard[12].CostPower = 1;
theCard[12].DEF = 0;
theCard[12].AttackPower = -3;
theCard[12].Effect = 2;//outfire
theCard[12].todo.LoadBitmapA("Bitmaps\\1R_3.bmp", RGB(50, 150, 150));
theCard[13].name = "2R_1";
theCard[13].CostPower = 2;
theCard[13].DEF = 0;
theCard[13].AttackPower = 0;
theCard[13].Effect = 3;//fire
theCard[13].todo.LoadBitmapA("Bitmaps\\2R_1.bmp", RGB(50, 150, 150));
theCard[14].name = "2R_2";
theCard[14].CostPower = 2;
theCard[14].DEF = 0;
theCard[14].AttackPower = 10;
theCard[14].Effect = 4;//cant get life me
theCard[14].todo.LoadBitmapA("Bitmaps\\2R_2.bmp", RGB(50, 150, 150));
theCard[15].name = "2R_3";
theCard[15].CostPower = 2;
theCard[15].DEF = 0;
theCard[15].AttackPower = 5;
theCard[15].Effect = 5;//cant get life enmy
theCard[15].todo.LoadBitmapA("Bitmaps\\2R_3.bmp", RGB(50, 150, 150));
theCard[16].name = "2R_4";
theCard[16].CostPower = 2;
theCard[16].DEF = 10;
theCard[16].AttackPower = 0;
theCard[16].Effect = 0;
theCard[16].todo.LoadBitmapA("Bitmaps\\2R_4.bmp", RGB(50, 150, 150));
theCard[17].name = "2R_5";
theCard[17].CostPower = 2;
theCard[17].DEF = 0;
theCard[17].AttackPower = 5;
theCard[17].Effect = 6;//undef
theCard[17].todo.LoadBitmapA("Bitmaps\\2R_5.bmp", RGB(50, 150, 150));
theCard[18].name = "3R_1";
theCard[18].CostPower = 3;
theCard[18].DEF = 5;
theCard[18].AttackPower = 5;
theCard[18].Effect = 0;
theCard[18].todo.LoadBitmapA("Bitmaps\\3R_1.bmp", RGB(50, 150, 150));
theCard[19].name = "3R_2";
theCard[19].CostPower = 3;
theCard[19].DEF = 0;
theCard[19].AttackPower = -12;
theCard[19].Effect = 0;
theCard[19].todo.LoadBitmapA("Bitmaps\\3R_2.bmp", RGB(50, 150, 150));
theCard[0].name = "1G_1";
theCard[0].CostPower = 1;
theCard[0].DEF = 0;
theCard[0].AttackPower = 0;
theCard[0].Effect = 1;//attack = def
theCard[0].todo.LoadBitmapA("Bitmaps\\1G_1.bmp", RGB(50, 150, 150));
theCard[1].name = "1G_2";
theCard[1].CostPower = 1;
theCard[1].DEF = 0;
theCard[1].AttackPower = -3;
theCard[1].Effect = 2;//outfire
theCard[1].todo.LoadBitmapA("Bitmaps\\1G_2.bmp", RGB(50, 150, 150));
theCard[2].name = "1G_3";
theCard[2].CostPower = 1;
theCard[2].DEF = 0;
theCard[2].AttackPower = 5;
theCard[2].Effect = 0;
theCard[2].todo.LoadBitmapA("Bitmaps\\1G_3.bmp", RGB(50, 150, 150));
theCard[3].name = "2G_1";
theCard[3].CostPower = 2;
```

```
		theCard[3].DEF = 0;
		theCard[3].AttackPower = 0;
		theCard[3].Effect = 3;//fire
		theCard[3].todo.LoadBitmapA("Bitmaps\\2G_1.bmp", RGB(50, 150, 150));
		theCard[4].name = "2G_2";
		theCard[4].CostPower = 2;
		theCard[4].DEF = 0;
		theCard[4].AttackPower = 5;
		theCard[4].Effect = 6;//undef
		theCard[4].todo.LoadBitmapA("Bitmaps\\2G_2.bmp", RGB(50, 150, 150));
		theCard[5].name = "2G_3";
		theCard[5].CostPower = 2;
		theCard[5].DEF = 10;
		theCard[5].AttackPower = 0;
		theCard[5].Effect = 0;
		theCard[5].todo.LoadBitmapA("Bitmaps\\2G_3.bmp", RGB(50, 150, 150));
		theCard[6].name = "2G_4";
		theCard[6].DEF = 0;
		theCard[6].AttackPower = 5;
		theCard[6].CostPower = 2;
		theCard[6].Effect = 5;//cant get life enmy
		theCard[6].todo.LoadBitmapA("Bitmaps\\2G_4.bmp", RGB(50, 150, 150));
		theCard[7].name = "2G_5";
		theCard[7].CostPower = 2;
		theCard[7].DEF = 0;
		theCard[7].AttackPower = 10;
		theCard[7].Effect = 4;//cant get life me
		theCard[7].todo.LoadBitmapA("Bitmaps\\2G_5.bmp", RGB(50, 150, 150));
		theCard[8].name = "3G_1";
		theCard[8].CostPower = 3;
		theCard[8].DEF = 5;
		theCard[8].AttackPower = 5;
		theCard[8].Effect = 0;
		theCard[8].todo.LoadBitmapA("Bitmaps\\3G_1.bmp", RGB(50, 150, 150));
		theCard[9].name = "3G_2";
		theCard[9].CostPower = 3;
		theCard[9].DEF = 0;
		theCard[9].AttackPower = -12;
		theCard[9].Effect = 0;
		theCard[9].todo.LoadBitmapA("Bitmaps\\3G_2.bmp", RGB(50, 150, 150));
		/**/
		theCard[20].name = "G_BK";
		theCard[20].CostPower = 0;
		theCard[20].DEF = 0;
		theCard[20].AttackPower = 0;
		theCard[20].Effect = 0;
		theCard[20].todo.LoadBitmapA("Bitmaps\\G_BK.bmp", RGB(50, 150, 150));
		theCard[21].name = "R_BK";
		theCard[21].CostPower = 0;
		theCard[21].DEF = 0;
		theCard[21].AttackPower = 0;
		theCard[21].Effect = 0;
		theCard[21].todo.LoadBitmapA("Bitmaps\\R_BK.bmp", RGB(50, 150, 150));

}
void Card1::OnMove()
{
		if (y <= SIZE_Y) {
			x += 3;
			y += 3;
		}
		else {
			x = y = 0;
		}
}
void Card1::onShow()
{
		theCard[c_rand].todo.SetTopLeft(x, y);
		theCard[c_rand].todo.ShowBitmap();
```

```
	theCard[c_rand2].todo.SetTopLeft(x + 250, y);
	theCard[c_rand2].todo.ShowBitmap();
	theCard[c_rand3].todo.SetTopLeft(x + 500, y);
	theCard[c_rand3].todo.ShowBitmap();
	/*card[c_rand].SetTopLeft(x, y);
	card[c_rand].ShowBitmap();
	card[c_rand2].SetTopLeft(x+100, y);
	card[c_rand2].ShowBitmap();
	card[c_rand3].SetTopLeft(x+200, y);
	card[c_rand3].ShowBitmap();*/
}

int* Card1::OnLButtonUp(UINT nFlags, CPoint point, int cost)
{
	int *num = new int[4];
	if (game_choose == 2)
	{
		/**/
		if (game_choose == 1)
		{
			if (point.x > 650 && point.x < 950 && point.y>850 && point.y < 950)
			{
				round += 2;
				num[0] = 0;
				num[1] = 777;
				num[2] = 0;
				num[3] = 0;
				is_attack = 1;
				if (round % 2 == 0)
				{

					c_rand = (rand() % 9);
					c_rand2 = (rand() % 9);
					c_rand3 = (rand() % 9);
				}
				if (round % 2 == 1)
				{
					c_rand = (rand() % 9 + 10);
					c_rand2 = (rand() % 9 + 10);
					c_rand3 = (rand() % 9 + 10);
				}
			}
			/**/
			else if (point.x > 440 && point.x < 678 && point.y>440 && point.y < 746)
			{
				num[0] = theCard[c_rand].AttackPower;
				num[1] = theCard[c_rand].CostPower;
				num[2] = theCard[c_rand].DEF;
				num[3] = theCard[c_rand].Effect;
				if (cost - num[1] >= 0)
				{
					if (round % 2 == 0)
					{
						c_rand = 20;
					}
					else
					{
						c_rand = 21;
					}
				}

			}
			else if (point.x > 690 && point.x < 928 && point.y > 440 && point.y < 746)
			{
				num[0] = theCard[c_rand2].AttackPower;
				num[1] = theCard[c_rand2].CostPower;
				num[2] = theCard[c_rand2].DEF;
				num[3] = theCard[c_rand2].Effect;
				if (cost - num[1] >= 0)
		{
			if (round % 2 == 0)
			{
				c_rand2 = 20;
			}
			else
			{
				c_rand2 = 21;
			}
		}

	}
	else if (point.x > 940 && point.x < 1178 && point.y > 440 && point.y < 746)
	{
		num[0] = theCard[c_rand3].AttackPower;
		num[1] = theCard[c_rand3].CostPower;
		num[2] = theCard[c_rand3].DEF;
		num[3] = theCard[c_rand3].Effect;
		if (cost - num[1] >= 0)
		{
			if (round % 2 == 0)
			{
				c_rand3 = 20;
			}
			else
			{
				c_rand3 = 21;
			}
		}

	}
	/*else if (OnRButtonUp(nFlags, point)==true)
	{
		round += 1;
		if (round % 2 == 0)
		{
			c_rand = (rand() % 9);
			c_rand2 = (rand() % 9);
			c_rand3 = (rand() % 9);
		}
		if (round % 2 == 1)
		{
			c_rand = (rand() % 9 + 10);
			c_rand2 = (rand() % 9 + 10);
			c_rand3 = (rand() % 9 + 10);
		}
		num[0] = 0;
		num[1] = 777;
		num[2] = 0;
		num[3] = 0;
	}*/
	/*else if (point.x > 700 && point.x < 800 && point.y>750 && point.y < 900)
	{
		round += 1;
		num[0] = 0;
		num[1] = 777;
		num[2] = 0;
		num[3] = 0;
		if (round % 2 == 0)
		{
			c_rand = (rand() % 9);
			c_rand2 = (rand() % 9);
			c_rand3 = (rand() % 9);
		}
		if (round % 2 == 1)
		{
			c_rand = (rand() % 9 + 10);
			c_rand2 = (rand() % 9 + 10);
			c_rand3 = (rand() % 9 + 10);
```

```
				}
			}*/
			else
			{
				num[0] = 0;
				num[1] = 0;
				num[2] = 0;
				num[3] = 0;
			}
		}
		else if (game_choose == 2)
		{
			if (point.x > 650 && point.x < 950 && point.y>850 && point.y < 950)
			{
				round += 1;
				num[0] = 0;
				num[1] = 777;
				num[2] = 0;
				num[3] = 0;
				if (round % 2 == 0)
				{
					c_rand = (rand() % 9);
					c_rand2 = (rand() % 9);
					c_rand3 = (rand() % 9);
				}
				if (round % 2 == 1)
				{
					c_rand = (rand() % 9 + 10);
					c_rand2 = (rand() % 9 + 10);
					c_rand3 = (rand() % 9 + 10);
				}
			}
			/**/
			else if (point.x > 440 && point.x < 678 && point.y>440 && point.y < 746)
			{
				num[0] = theCard[c_rand].AttackPower;
				num[1] = theCard[c_rand].CostPower;
				num[2] = theCard[c_rand].DEF;
				num[3] = theCard[c_rand].Effect;
				if (cost - num[1] >= 0)
				{
					if (round % 2 == 0)
					{
						c_rand = 20;
					}
					else
					{
						c_rand = 21;
					}
				}

			}
			else if (point.x > 690 && point.x < 928 && point.y > 440 && point.y < 746)
			{
				num[0] = theCard[c_rand2].AttackPower;
				num[1] = theCard[c_rand2].CostPower;
				num[2] = theCard[c_rand2].DEF;
				num[3] = theCard[c_rand2].Effect;
				if (cost - num[1] >= 0)
				{
					if (round % 2 == 0)
					{
						c_rand2 = 20;
					}
					else
					{
						c_rand2 = 21;
					}
				}
				}
				else if (point.x > 940 && point.x < 1178 && point.y > 440 && point.y < 746)
				{
					num[0] = theCard[c_rand3].AttackPower;
					num[1] = theCard[c_rand3].CostPower;
					num[2] = theCard[c_rand3].DEF;
					num[3] = theCard[c_rand3].Effect;
					if (cost - num[1] >= 0)
					{
						if (round % 2 == 0)
						{
							c_rand3 = 20;
						}
						else
						{
							c_rand3 = 21;
						}
					}

				}

				else
				{
					num[0] = 0;
					num[1] = 0;
					num[2] = 0;
					num[3] = 0;
				}
			}

		}
		if (game_choose == 1)
		{
			if (point.x > 650 && point.x < 950 && point.y>850 && point.y < 950)
			{
				round += 1;
				num[0] = 0;
				num[1] = 777;
				num[2] = 0;
				num[3] = 0;
				if (round % 2 == 0)
				{
					c_rand = (rand() % 9);
					c_rand2 = (rand() % 9);
					c_rand3 = (rand() % 9);
				}
				if (round % 2 == 1)
				{
					Player2isEnd = 0;
					/*c_rand = (rand() % 9 + 10);
					c_rand2 = (rand() % 9 + 10);
					c_rand3 = (rand() % 9 + 10);*/
					c_rand = 21;
					c_rand2 = 21;
					c_rand3 = 21;
				}
			}
			/**/
			else if (point.x > 440 && point.x < 678 && point.y>440 && point.y < 746)
			{
				num[0] = theCard[c_rand].AttackPower;
				num[1] = theCard[c_rand].CostPower;
				num[2] = theCard[c_rand].DEF;
				num[3] = theCard[c_rand].Effect;
				if (cost - num[1] >= 0)
				{
					if (round % 2 == 0)
					{
```

```
                c_rand = 20;
            }
            else
            {
                c_rand = 21;
            }
        }

}
else if (point.x > 690 && point.x < 928 && point.y > 440 && point.y < 746)
{
        num[0] = theCard[c_rand2].AttackPower;
        num[1] = theCard[c_rand2].CostPower;
        num[2] = theCard[c_rand2].DEF;
        num[3] = theCard[c_rand2].Effect;
        if (cost - num[1] >= 0)
        {
            if (round % 2 == 0)
            {
                c_rand2 = 20;
            }
            else
            {
                c_rand2 = 21;
            }
        }

}
else if (point.x > 940 && point.x < 1178 && point.y > 440 && point.y < 746)
{
        num[0] = theCard[c_rand3].AttackPower;
        num[1] = theCard[c_rand3].CostPower;
        num[2] = theCard[c_rand3].DEF;
        num[3] = theCard[c_rand3].Effect;
        if (cost - num[1] >= 0)
        {
            if (round % 2 == 0)
            {
                c_rand3 = 20;
            }
            else
            {
                c_rand3 = 21;
            }
        }

}
/*else if (OnRButtonUp(nFlags, point)==true)
{
        round += 1;
        if (round % 2 == 0)
        {
            c_rand = (rand() % 9);
            c_rand2 = (rand() % 9);
            c_rand3 = (rand() % 9);
        }
        if (round % 2 == 1)
        {
            c_rand = (rand() % 9 + 10);
            c_rand2 = (rand() % 9 + 10);
            c_rand3 = (rand() % 9 + 10);
        }
        num[0] = 0;
        num[1] = 777;
        num[2] = 0;
        num[3] = 0;
}*/
/*else if (point.x > 700 && point.x < 800 && point.y>750 && point.y < 900)
{
```

```
                    round += 1;
                    num[0] = 0;
                    num[1] = 777;
                    num[2] = 0;
                    num[3] = 0;
                    if (round % 2 == 0)
                    {
                            c_rand = (rand() % 9);
                            c_rand2 = (rand() % 9);
                            c_rand3 = (rand() % 9);
                    }
                    if (round % 2 == 1)
                    {
                            c_rand = (rand() % 9 + 10);
                            c_rand2 = (rand() % 9 + 10);
                            c_rand3 = (rand() % 9 + 10);
                    }
            }*/
            else
            {
                    num[0] = 0;
                    num[1] = 0;
                    num[2] = 0;
                    num[3] = 0;
            }
    }
    return num;

}

/*bool Card1::OnRButtonUp(UINT nFlags, CPoint point)
{
    return true;
}*/
//
CGameStateRun::CGameStateRun(CGame *g)
    : CGameState(g), NUMBALLS(28)
{
    //ball = new CBall [NUMBALLS];
}

CGameStateRun::~CGameStateRun()
{
    //delete [] ball;

}

void CGameStateRun::OnBeginState()
{
    CAudio::Instance()->Play(AUDIO_BGM, true);
    /*const int BALL_GAP = 90;
    const int BALL_XY_OFFSET = 45;
    const int BALL_PER_ROW = 7;
    const int HITS_LEFT = 10;
    const int HITS_LEFT_X = 590;
    const int HITS_LEFT_Y = 0;
    const int BACKGROUND_X = 60;
    const int ANIMATION_SPEED = 15;
    for (int i = 0; i < NUMBALLS; i++) {                    // 設定球的起始座標
            int x_pos = i % BALL_PER_ROW;
            int y_pos = i / BALL_PER_ROW;
            ball[i].SetXY(x_pos * BALL_GAP + BALL_XY_OFFSET, y_pos * BALL_GAP + BALL_XY_OFFSET);
            ball[i].SetDelay(x_pos);
            ball[i].SetIsAlive(true);
    }
    eraser.Initialize();
    background.SetTopLeft(BACKGROUND_X,0);                  // 設定背景的起始座標
    help.SetTopLeft(0, SIZE_Y - help.Height());             // 設定說明圖的起始座標
    hits_left.SetInteger(HITS_LEFT);                        // 指定剩下的撞擊數
```

```
        hits_left.SetTopLeft(HITS_LEFT_X,HITS_LEFT_Y);                  // 指定剩下撞擊數的座標
        CAudio::Instance()->Play(AUDIO_LAKE, true);                     // 撥放 WAVE
        CAudio::Instance()->Play(AUDIO_DING, false);             // 撥放 WAVE
        CAudio::Instance()->Play(AUDIO_NTUT, true);                     // 撥放 MIDI*/
}

void CGameStateRun::OnMove()                                                    // 移動遊戲元素
{
        //
        // 如果希望修改 cursor 的樣式，則將下面程式的 commment 取消即可
        //
        // SetCursor(AfxGetApp()->LoadCursor(IDC_GAMECURSOR));
        //
        // 移動背景圖的座標
        //
        /*BackGround.SetTopLeft(0, 0);
        if (background.Top() > SIZE_Y)
                background.SetTopLeft(60 ,-background.Height());
        background.SetTopLeft(background.Left(),background.Top()+1);*/
        //
        // 移動球
        //

        //int i;
        /*for (i=0; i < NUMBALLS; i++)
                ball[i].OnMove();*/
                //
                // 移動擦子
                //
                //eraser.OnMove();
                //
                // 判斷擦子是否碰到球
                //
                /*for (i=0; i < NUMBALLS; i++)
                        if (ball[i].IsAlive() && ball[i].HitEraser(&eraser)) {
                                ball[i].SetIsAlive(false);
                                CAudio::Instance()->Play(AUDIO_DING);
                                hits_left.Add(-1);
                                //
                                // 若剩餘碰撞次數為 0，則跳到 Game Over 狀態
                                //
                                if (hits_left.GetInteger() <= 0) {
                                        CAudio::Instance()->Stop(AUDIO_LAKE); // 停止 WAVE
                                        CAudio::Instance()->Stop(AUDIO_NTUT); // 停止 MIDI
                                        GotoGameState(GAME_STATE_OVER);
                                }
                        }*/
                        //
                        // 移動彈跳的球
                        //
                        //bball.OnMove();
}

void CGameStateRun::OnInit()                                                            // 遊戲的初值及圖形設定
{
        //
        // 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(33);           // 接個前一個狀態的進度，此處進度視為 33%
        //
        // 開始載入資料
        //
        //BackGround.LoadBitmap("Bitmaps/ESHL.bmp");

                BackGround[0].LoadBitmap("Bitmaps\\DMbig.bmp");

                BackGround[1].LoadBitmap("Bitmaps\\BNSbig.bmp");
                BackGround[2].LoadBitmap("Bitmaps\\IGbig.bmp");

                BackGround[3].LoadBitmap("Bitmaps\\ESHLbig.bmp");

        char1.LoadBitmapA("Bitmaps\\Forward.bmp", RGB(0, 0, 0));
        shield_c1.LoadBitmapA("Bitmaps\\showshield.bmp", RGB(50, 150, 150));
        shieldline_c1.LoadBitmapA("Bitmaps\\shieldline.bmp", RGB(50, 150, 150));
        for (int i = 0; i < 30; i++)
        {
                if (i == 0) { shieldpoint_c1[0].LoadBitmapA("Bitmaps\\shield_h.bmp", RGB(50, 150, 150)); }
                else if (i == 29) { shieldpoint_c1[29].LoadBitmapA("Bitmaps\\shield_t.bmp", RGB(50, 150, 150)); }
                else { shieldpoint_c1[i].LoadBitmapA("Bitmaps\\shield_m.bmp", RGB(50, 150, 150)); }
        }
        blood_c1.LoadBitmapA("Bitmaps\\showblood.bmp", RGB(50, 150, 150));
        bloodline_c1.LoadBitmapA("Bitmaps\\bloodline.bmp", RGB(50, 150, 150));
        for (int i = 0; i < 30; i++)
        {
                if (i == 0) { bloodpoint_c1[0].LoadBitmapA("Bitmaps\\blood_h.bmp", RGB(50, 150, 150)); }
                else if (i == 29) { bloodpoint_c1[29].LoadBitmapA("Bitmaps\\blood_t.bmp", RGB(50, 150, 150)); }
                else { bloodpoint_c1[i].LoadBitmapA("Bitmaps\\blood_m.bmp", RGB(50, 150, 150)); }
        }
        for (int i = 0; i < 3; i++) { usedollor_c1[i].LoadBitmapA("Bitmaps\\dollor.bmp", RGB(50, 150, 150)); }
        shield_c2.LoadBitmapA("Bitmaps\\showshield.bmp", RGB(50, 150, 150));
        shieldline_c2.LoadBitmapA("Bitmaps\\shieldline.bmp", RGB(50, 150, 150));
        for (int i = 0; i < 30; i++)
        {
                if (i == 0) { shieldpoint_c2[0].LoadBitmapA("Bitmaps\\shield_h.bmp", RGB(50, 150, 150)); }
                else if (i == 29) { shieldpoint_c2[29].LoadBitmapA("Bitmaps\\shield_t.bmp", RGB(50, 150, 150)); }
                else { shieldpoint_c2[i].LoadBitmapA("Bitmaps\\shield_m.bmp", RGB(50, 150, 150)); }
        }
        blood_c2.LoadBitmapA("Bitmaps\\showblood.bmp", RGB(50, 150, 150));
        bloodline_c2.LoadBitmapA("Bitmaps\\bloodline.bmp", RGB(50, 150, 150));
        for (int i = 0; i < 30; i++)
        {
                if (i == 0) { bloodpoint_c2[0].LoadBitmapA("Bitmaps\\blood_h.bmp", RGB(50, 150, 150)); }
                else if (i == 29) { bloodpoint_c2[29].LoadBitmapA("Bitmaps\\blood_t.bmp", RGB(50, 150, 150)); }
                else { bloodpoint_c2[i].LoadBitmapA("Bitmaps\\blood_m.bmp", RGB(50, 150, 150)); }
        }
        for (int i = 0; i < 3; i++) { usedollor_c2[i].LoadBitmapA("Bitmaps\\dollor.bmp", RGB(50, 150, 150)); }
        showRound[0].LoadBitmapA("Bitmaps\\charter1n.bmp", RGB(50, 150, 150));
        showRound[1].LoadBitmapA("Bitmaps\\charter2n.bmp", RGB(50, 150, 150));
        endturn.LoadBitmapA("Bitmaps\\endturn.bmp", RGB(50, 150, 150));
        c_charter1.LoadBit();
        c_charter2.LoadBit();
        c_card1.LoadBit();

        /*int i;
        for (i = 0; i < NUMBALLS; i++)
                ball[i].LoadBitmap();                                           // 載入第 i 個球的圖形*/
                //eraser.LoadBitmap();
                //background.LoadBitmap(IDB_BACKGROUND);                              // 載入背景的圖形
                //
                // 完成部分 Loading 動作，提高進度
                //
        ShowInitProgress(50);
        Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep
        //
        // 繼續載入其他資料
        //
        help.LoadBitmap(IDB_HELP, RGB(255, 255, 255));                          // 載入說明的圖形
        /*corner.LoadBitmap(IDB_CORNER);                                            // 載入角落圖形
        corner.ShowBitmap(background);                                          // 將 corner 貼到 background
        bball.LoadBitmap();                                                         // 載入圖形
        hits_left.LoadBitmap();                 */
        CAudio::Instance()->Load(AUDIO_DING, "sounds\\ding.wav");   // 載入編號 0 的聲音 ding.wav
        CAudio::Instance()->Load(AUDIO_LAKE, "sounds\\lake.mp3");   // 載入編號 1 的聲音 lake.mp3
        CAudio::Instance()->Load(AUDIO_NTUT, "sounds\\ntut.mid");   // 載入編號 2 的聲音 ntut.mid
        CAudio::Instance()->Load(AUDIO_BGM, "sounds\\BGM.mp3");
        //
```

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

}

void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	/*const char KEY_LEFT    = 0x25; // keyboard 左箭頭
	const char KEY_UP      = 0x26; // keyboard 上箭頭
	const char KEY_RIGHT = 0x27; // keyboard 右箭頭
	const char KEY_DOWN    = 0x28; // keyboard 下箭頭
	if (nChar == KEY_LEFT)
		eraser.SetMovingLeft(true);
	if (nChar == KEY_RIGHT)
		eraser.SetMovingRight(true);
	if (nChar == KEY_UP)
		eraser.SetMovingUp(true);
	if (nChar == KEY_DOWN)
		eraser.SetMovingDown(true);*/
}

void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	/*const char KEY_LEFT    = 0x25; // keyboard 左箭頭
	const char KEY_UP      = 0x26; // keyboard 上箭頭
	const char KEY_RIGHT = 0x27; // keyboard 右箭頭
	const char KEY_DOWN    = 0x28; // keyboard 下箭頭
	if (nChar == KEY_LEFT)
		eraser.SetMovingLeft(false);
	if (nChar == KEY_RIGHT)
		eraser.SetMovingRight(false);
	if (nChar == KEY_UP)
		eraser.SetMovingUp(false);
	if (nChar == KEY_DOWN)
		eraser.SetMovingDown(false);*/
}

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

}

void CGameStateRun::OnLButtonUp(UINT nFlags, CPoint point) // 處理滑鼠的動作
{
	if (init_pass == 0)
	{
		if (map_choose == 1)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 45;
		}
		if (map_choose == 2)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 45;
		}
		if (map_choose == 3)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 45;
		}
		if (map_choose == 4)
		{
			c_charter1.blood = 45;
			c_charter2.blood = 45;
		}
		init_pass = 1;
	}
```

```
		//eraser.SetMovingLeft(false);
		int *getreturn;
		/*int AttackPower = 0;
		int CostPower = 0;
		int DEF = 0;
		int Effect = 0;*/

		if (c_charter1.x == 0 && ((c1_isActed == true && c2isActed2 == true && c1isActed2 == false && c2isActed ==
false) || (c1_isActed == false && c2isActed2 == false && c1isActed2 == true && c2isActed == true)))
			//if (c_charter1.x == 0)
		{
			if (c_card1.round % 2 == 0)
			{
				getreturn = c_card1.OnLButtonUp(nFlags, point, c_charter1.pow);
			}
			else
			{
				getreturn = c_card1.OnLButtonUp(nFlags, point, c_charter2.pow);
			}

			AttackPower = getreturn[0]; //血
			judgeAttackPower = AttackPower;
			CostPower = getreturn[1];	//花費
			DEF = getreturn[2];		//盾
			judgeShield = DEF;
			Effect = getreturn[3];		//影響
			/**/

			if (oldround != c_card1.round)
			{
				if (countc1round > 0){countc1round--;}
				else{c_charter1.getlife = 0;}
				if (countc2round > 0){countc2round--;}
				else{c_charter2.getlife = 0;}
				if (c_card1.round % 2 == 0)
				{
					if (c_charter2.fire >= 0)
					{
						c_charter2.blood -= c_charter2.fire;
						c_charter2.fire *= 2;
					}
					/*if (c_charter1.getlife != 0)
					{
						c_charter1.getlife = 0;
					}*/
					oldround = c_card1.round;
				}
				else if (c_card1.round % 2 == 1)
				{
					if (c_charter1.fire > 0)
					{
						c_charter1.blood -= c_charter1.fire;
						c_charter1.fire *= 2;
					}
					/*if (c_charter2.getlife != 0)
					{
						c_charter2.getlife = 0;
					}*/
					oldround = c_card1.round;
				}
				else
				{
					oldround = c_card1.round;
				}
			}

			if (CostPower == 777)
			{
				c_charter1.pow = 3;
```

```
                c_charter2.pow = 3;
            }
            /**/
            else if ((c_card1.round % 2 == 0 && c_charter1.pow >= CostPower) || (c_card1.round % 2 == 1 &&
c_charter2.pow >= CostPower))
            {
                if (c_card1.round % 2 == 0)
                {
                    c_charter1.pow -= CostPower;
                    /**/
                    if (Effect == 1)
                    {
                        AttackPower += c_charter1.shield;
                        judgeAttackPower = AttackPower;
                    }

                    if (AttackPower > 0) //攻擊
                    {
                        c1isActed2 = false;
                        c2isActed = false;
                        c1count2 = 0;
                        c2count = 0;
                        if (Effect == 6)
                        {
                            c_charter2.blood -= AttackPower;
                        }
                        else if (c_charter2.shield > 0)
                        {
                            if (c_charter2.shield > AttackPower)
                            {
                                c_charter2.shield -= AttackPower;
                            }
                            else
                            {
                                AttackPower -= c_charter2.shield;
                                c_charter2.shield = 0;
                                c_charter2.blood -= AttackPower;
                            }
                        }
                        else if (c_charter2.shield <= 0)
                        {
                            c_charter2.blood -= AttackPower;
                        }
                    }
                    else if (AttackPower < 0) //補血
                    {
                        if (c_charter1.getlife == 0)
                        {
                            c_charter1.blood -= AttackPower;
                        }

                    }

                    if (DEF > 0)
                    {
                        c_charter1.shield += DEF;
                    }

                    if (Effect == 2)
                    {
                        c_charter1.fire = 0;
                    }
                    else if (Effect == 3)
                    {
                        if (c_charter2.fire <= 0)
                        {
                            c_charter2.fire++;
                        }
                    }
```

```
          else if (Effect == 4)
          {
                    c_charter1.getlife = 1;
                    countc1round += 2;
          }
          else if (Effect == 5)
          {
                    c_charter2.getlife = 1;
                    countc2round += 1;
          }
}
else if (c_card1.round % 2 == 1)
{
          c_charter2.pow -= CostPower;
          /**/
          if (Effect == 1)
          {
                    AttackPower += c_charter2.shield;
                    judgeAttackPower = AttackPower;
          }

          if (AttackPower > 0) //攻擊
          {
                    c1_isActed = false;
                    c2isActed2 = false;
                    c1_count = 0;
                    c2count2 = 0;
                    if (Effect == 6)
                    {
                              c_charter1.blood -= AttackPower;
                    }
                    else if (c_charter1.shield > 0)
                    {
                              if (c_charter1.shield > AttackPower)
                              {
                                        c_charter1.shield -= AttackPower;
                              }
                              else
                              {
                                        AttackPower -= c_charter1.shield;
                                        c_charter1.shield = 0;
                                        c_charter1.blood -= AttackPower;
                              }
                    }
                    else if (c_charter1.shield <= 0)
                    {
                              c_charter1.blood -= AttackPower;
                    }
          }
          else if (AttackPower < 0) //補血
          {
                    if (c_charter2.getlife == 0)
                    {
                              c_charter2.blood -= AttackPower;
                    }

          }

          if (DEF > 0)
          {
                    c_charter2.shield += DEF;
          }

          if (Effect == 2)
          {
                    c_charter2.fire = 0;
          }
          else if (Effect == 3)
          {
                if (c_charter1.fire <= 0)
                {
                        c_charter1.fire++;
                }
        }
        else if (Effect == 4)
        {
                c_charter2.getlife = 1;
                countc2round += 2;
        }
        else if (Effect == 5)
        {
                c_charter1.getlife = 1;
                countc1round += 1;
        }
}
/*if (Effect >= 0)
{
        if (AttackPower > 0 || Effect == 1 || Effect == 3 || c_charter1.fire >= 1 || c_charter2.fire >= 1)
        {
                if (c_card1.round % 2 == 0)
                {
                        if (Effect == 1)
                        {
                                AttackPower = c_charter1.shield;
                        }
                        if (c_charter2.fire >= 1)
                        {
                                AttackPower += c_charter2.fire;
                        }
                        if (c_charter2.shield > 0 && Effect != 6)
                        {
                                if (c_charter2.shield > AttackPower)
                                {
                                        c_charter2.shield -= AttackPower;
                                }
                                else
                                {
                                        AttackPower -= c_charter2.shield;
                                        c_charter2.shield = 0;
                                        c_charter2.blood -= AttackPower;
                                }
                        }
                        else
                        {
                                c_charter2.blood -= AttackPower;
                        }
                        if (Effect == 3)
                        {
                                c_charter2.fire = 1;
                                if (c_charter2.fire >= 1)
                                {
                                        c_charter2.fire = c_charter2.fire * 2;
                                }
                        }
                        if (Effect == 4)
                        {
                                c_charter1.getlife = 1;
                        }
                        if (Effect == 5)
                        {
                                c_charter2.getlife = 1;
                        }
                        c_charter2.onShow();
                }
                if (c_card1.round % 2 == 1)
                {
                        if (Effect == 1)
                        {
```

```
					AttackPower = c_charter2.shield;
					judgeAttackPower = AttackPower;
				}
				if (c_charter1.fire >= 1)
				{
					AttackPower += c_charter1.fire;
					c_charter1.fire = c_charter1.fire * 2;
				}
				if (c_charter1.shield > 0 && Effect != 6)
				{
					if (c_charter1.shield > AttackPower)
					{
						c_charter1.shield -= AttackPower;
					}
					else
					{
						AttackPower -= c_charter1.shield;
						c_charter1.shield = 0;
						c_charter1.blood -= AttackPower;
					}
				}
				else
				{
					c_charter1.blood -= AttackPower;
				}
				if (Effect == 3)
				{
					c_charter1.fire = 1;
					if (c_charter1.fire >= 1)
					{
						c_charter1.fire = c_charter1.fire * 2;
					}
				}
				if (Effect == 4)
				{
					c_charter2.getlife = 1;
				}
				if (Effect == 5)
				{
					c_charter1.getlife = 1;
				}
				//c_charter1.onShow();
			}

		}
		else if (AttackPower < 0 || Effect == 2)
		{
			if (c_card1.round % 2 == 0)
			{
				if (Effect == 2)
				{
					c_charter1.fire = 0;
				}
				if (c_charter1.getlife == 0)
				{
					c_charter1.blood -= AttackPower;
				}
				//c_charter1.onShow();
			}
			if (c_card1.round % 2 == 1)
			{
				if (Effect == 2)
				{
					c_charter1.fire = 0;
				}
				if (c_charter2.getlife == 0)
				{
					c_charter2.blood -= AttackPower;
				}
```

```
                                        c_charter2.onShow();
                                }

                        }

                        if (DEF > 0)
                        {
                                if (c_card1.round % 2 == 0)
                                {
                                        c_charter1.shield += DEF;
                                        //c_charter1.onShow();
                                }
                                if (c_card1.round % 2 == 1)
                                {
                                        c_charter2.shield += DEF;
                                        c_charter2.onShow();
                                }

                        }

                        if (c_charter2.blood <= 0 || c_charter1.blood <= 0)
                        {
                                c_charter1.blood = 30;
                                c_charter2.blood = 30;
                                c_charter1.shield = 0;
                                c_charter2.shield = 0;
                                GotoGameState(GAME_STATE_OVER);
                        }
                        c_card1.onShow();
                }*/
        }
        /*if ((c_charter2.blood <= 0 || c_charter1.blood <= 0) && ((c1_isActed == true && c2isActed2 == true &&
c1isActed2 == false && c2isActed == false) || (c1_isActed == false && c2isActed2 == false && c1isActed2 == true && c2isActed
== true)))
        {
                if (c_charter1.blood <= 0)
                {
                        get_win = 2;
                }
                else if (c_charter2.blood <= 0)
                {
                        get_win = 1;
                }
                else
                {
                        get_win = 0;
                }
                c_charter1.blood = 30;
                c_charter2.blood = 30;
                c_charter1.shield = 0;
                c_charter2.shield = 0;
                c_charter1.getlife = 0;
                c_charter2.getlife = 0;
                c_charter1.fire = 0;
                c_charter2.fire = 0;
                c_card1.round = 0;
                c_charter1.x = -580;
                c_charter2.x = 1680;
                AttackPower = 0;
                judgeAttackPower = 0;
                judgeShield = 0;
                CostPower = 0;
                DEF = 0;
                Effect = 0;
                c1_count = 16;
                c1count2 = 0;
                c1isActed2 = false;
                c1_isActed = true;
                c2count = 0;
                c2count2 = 16;
                c2isActed2 = true;
                c2isActed = false;
                oldround = -1;
                countc1round = 0;
                countc2round = 0;
                c_card1.onShow();
                unsigned seed = (unsigned)time(NULL); //random 種子宣告(以時間)
                srand(seed); //用時間取亂數
                c_card1.c_rand = (rand() % 9);
                c_card1.c_rand2 = (rand() % 9);
                c_card1.c_rand3 = (rand() % 9);
                GotoGameState(GAME_STATE_WIN);
            }
            */
            /*else if (CostPower == 777)
            {
                c_charter1.pow = 3;
                c_charter2.pow = 3;
            }*/
        }
}

void CGameStateRun::OnMouseMove(UINT nFlags, CPoint point)         // 處理滑鼠的動作
{
        // 沒事。如果需要處理滑鼠移動的話，寫 code 在這裡
}

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

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

void CGameStateRun::OnShow()
{
        //
        //  注意：Show 裡面千萬不要移動任何物件的座標，移動座標的工作應由 Move 做才對，
        //          否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語
        //          說，Move 負責 MVC 中的 Model，Show 負責 View，而 View 不應更動 Model。
        //
        //
        //  貼上背景圖、撞擊數、球、擦子、彈跳的球
        //
        //CMovingBitmap     isfire, arefire;
        if (map_choose == 1)
        {
                BackGround[0].ShowBitmap();
        }
        else if (map_choose == 2)
        {
                BackGround[1].ShowBitmap();
        }
        else if (map_choose == 3)
        {
                BackGround[2].ShowBitmap();
        }
        else if (map_choose == 4)
        {
                BackGround[3].ShowBitmap();
        }
        else
        {
                BackGround[2].ShowBitmap();
        }
```

```
CMovingBitmap       isfire, arefire;
CMovingBitmap       isban, areban;
CMovingBitmap shieldR, shieldR1, shieldR2;
if (c_charter1.blood < 0) { c_charter1.blood = 0; }
if (c_charter2.blood < 0) { c_charter2.blood = 0; }
if (c_charter1.x >= 0)
{
        bloodline_c1.SetTopLeft(201, 160);
        bloodline_c1.ShowBitmap();
        {
                CDC *pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32];
                stringstream ss;
                ss << c_charter1.blood;
                ss >> blood_char;
                f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                           // 選用 font f
                pDC->SetTextColor(RGB(255, 255, 255));
                pDC->SetBkColor(RGB(255, 0, 0));
                pDC->TextOut(400, 160, blood_char);
                pDC->SelectObject(fp);                                          // 放掉 font f (千萬不要漏了
放掉)
                CDDraw::ReleaseBackCDC();                                       // 放掉 Back Plain 的 CDC
        }
        int bx = 201;
        for (int i = 0; i < c_charter1.blood; i++)
        {
                if (i == 28)
                {
                        bloodpoint_c1[29].SetTopLeft(201, 160);
                        bloodpoint_c1[29].ShowBitmap();
                }
                else if (i == 0)
                {
                        bloodpoint_c1[0].SetTopLeft(201, 160);
                        bloodpoint_c1[0].ShowBitmap();
                }
                else if (i < 28)
                {
                        bloodpoint_c1[i].SetTopLeft(bx, 160);
                        bloodpoint_c1[i].ShowBitmap();
                        bx += 6;
                }
        }
        blood_c1.SetTopLeft(200, 160);
        blood_c1.ShowBitmap();
        shieldline_c1.SetTopLeft(200, 200);
        shieldline_c1.ShowBitmap();
        bx = 200;
        {
                CDC *pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char shield_char[32];
                stringstream ss;
                ss << c_charter1.shield;
                ss >> shield_char;
                f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                           // 選用 font f
                pDC->SetTextColor(RGB(255, 255, 255));
                pDC->SetBkColor(RGB(0, 255, 0));
                pDC->TextOut(400, 200, shield_char);
                pDC->SelectObject(fp);                                          // 放掉 font f (千萬不要漏了
放掉)
                CDDraw::ReleaseBackCDC();                                       // 放掉 Back Plain 的 CDC
        }
        for (int i = 0; i < c_charter1.shield; i++)
        {
                if (i == 28)
```

```
          {
                    shieldpoint_c1[29].SetTopLeft(200, 200);
                    shieldpoint_c1[29].ShowBitmap();
          }
          else if (i == 0)
          {
                    shieldpoint_c1[0].SetTopLeft(200, 200);
                    shieldpoint_c1[0].ShowBitmap();
          }
          else if (i < 28)
          {
                    shieldpoint_c1[i].SetTopLeft(bx, 200);
                    shieldpoint_c1[i].ShowBitmap();
                    bx += 6;
          }
}
shield_c1.SetTopLeft(200, 200);
shield_c1.ShowBitmap();
bx = 100;
for (int i = 0; i < c_charter1.pow; i++)
{
          usedollor_c1[i].SetTopLeft(bx, 800);
          usedollor_c1[i].ShowBitmap();
          bx += 70;
}
bloodline_c2.SetTopLeft(1251, 160);
bloodline_c2.ShowBitmap();
bx = 1251;
{
          CDC *pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
          CFont f, *fp;
          char blood_char[32];
          stringstream ss;
          ss << c_charter2.blood;
          ss >> blood_char;
          f.CreatePointFont(200, "Times New Roman");    // 產生 font f; 160 表示 16 point 的字
          fp = pDC->SelectObject(&f);                         // 選用 font f
          pDC->SetTextColor(RGB(255, 255, 255));
          pDC->SetBkColor(RGB(255, 0, 0));
          pDC->TextOut(1450, 160, blood_char);
          pDC->SelectObject(fp);                                        // 放掉 font f (千萬不要漏了
放掉)
          CDDraw::ReleaseBackCDC();                                     // 放掉 Back Plain 的 CDC
}
for (int i = 0; i < c_charter2.blood; i++)
{
          if (i == 28)
          {
                    bloodpoint_c2[29].SetTopLeft(1251, 160);
                    bloodpoint_c2[29].ShowBitmap();
          }
          else if (i == 0)
          {
                    bloodpoint_c2[0].SetTopLeft(1251, 160);
                    bloodpoint_c2[0].ShowBitmap();
          }
          else if (i < 28)
          {
                    bloodpoint_c2[i].SetTopLeft(bx, 160);
                    bloodpoint_c2[i].ShowBitmap();
                    bx += 6;
          }
}
blood_c2.SetTopLeft(1250, 160);
blood_c2.ShowBitmap();
shieldline_c1.SetTopLeft(1250, 200);
shieldline_c1.ShowBitmap();
bx = 1250;
{
```

```
            CDC *pDC = CDDraw::GetBackCDC();                    // 取得 Back Plain 的 CDC
            CFont f, *fp;
            char shield_char[32];
            stringstream ss;
            ss << c_charter2.shield;
            ss >> shield_char;
            f.CreatePointFont(200, "Times New Roman");       // 產生 font f; 160 表示 16 point 的字
            fp = pDC->SelectObject(&f);                          // 選用 font f
            pDC->SetTextColor(RGB(255, 255, 255));
            pDC->SetBkColor(RGB(0, 255, 0));
            pDC->TextOut(1450, 200, shield_char);
            pDC->SelectObject(fp);                                       // 放掉 font f (千萬不要漏了
放掉)
            CDDraw::ReleaseBackCDC();                                    // 放掉 Back Plain 的 CDC
        }
        for (int i = 0; i < c_charter2.shield; i++)
        {
            if (i == 28)
            {
                shieldpoint_c2[29].SetTopLeft(1250, 200);
                shieldpoint_c2[29].ShowBitmap();
            }
            else if (i == 0)
            {
                shieldpoint_c2[0].SetTopLeft(1250, 200);
                shieldpoint_c2[0].ShowBitmap();
            }
            else if (i < 28)
            {
                shieldpoint_c2[i].SetTopLeft(bx, 200);
                shieldpoint_c2[i].ShowBitmap();
                bx += 6;
            }
        }
        shield_c2.SetTopLeft(1250, 200);
        shield_c2.ShowBitmap();
        bx = 1150;
        for (int i = 0; i < c_charter2.pow; i++)
        {
            usedollor_c2[i].SetTopLeft(bx, 800);
            usedollor_c2[i].ShowBitmap();
            bx += 70;
        }
    }
    c_charter2.OnMove();
    c_charter1.OnMove();
    if (c_card1.round % 2 == 1)
    {
        //if (judgeAttackPower > 0)
        if (judgeAttackPower > 0 && c1_isActed == false && c2count2 >= 16)
        {
            c1count2 = 0;
            c1isActed2 = false;
            if (c1_count == 0)c_charter1.pic = c_charter1.attackedpic[0];
            if (c1_count == 4)c_charter1.pic = c_charter1.attackedpic[1];
            if (c1_count == 8)c_charter1.pic = c_charter1.attackedpic[2];
            if (c1_count == 12)c_charter1.pic = c_charter1.attackedpic[3];
            if (c1_count == 16) { c_charter1.pic = c_charter1.attackedpic[4]; c1_isActed = true; }
            c1_count++;
            //if (c1_count >= 16) { c1_count = 0; }
        }
        else if (judgeAttackPower < 0 && c1_isActed == false)
        {
            c1_isActed = true;
            c1count2 = 0;
            c1isActed2 = false;
        }
        //if (judgeAttackPower > 0)
        if (judgeAttackPower > 0 && c2isActed2 == false)
				{
					c2count = 0;
					c2isActed = false;
					if (c2count2 == 0)c_charter2.pic = c_charter2.attackpic[0];
					if (c2count2 == 4)c_charter2.pic = c_charter2.attackpic[1];
					if (c2count2 == 8)c_charter2.pic = c_charter2.attackpic[2];
					if (c2count2 == 12)c_charter2.pic = c_charter2.attackpic[3];
					if (c2count2 == 16) { c_charter2.pic = c_charter2.attackpic[4]; c2isActed2 = true; }
					c2count2++;
					//if (c2count2 >= 16) { c2count2 = 0; }
				}
				else if (judgeAttackPower < 0 && c2isActed2 == false)
				{
					c2isActed2 = true;
					c2count = 0;
					c2isActed = false;
				}
			}
			if (c_card1.round % 2 == 0)
			{
				//if (judgeAttackPower > 0)
				if (judgeAttackPower > 0 && c1isActed2 == false)
				{
					c1_count = 0;
					c1_isActed = false;
					if (c1count2 == 0) { c_charter1.pic = c_charter1.attackpic[0]; }
					if (c1count2 == 4) { c_charter1.pic = c_charter1.attackpic[1]; }
					if (c1count2 == 8) { c_charter1.pic = c_charter1.attackpic[2]; }
					if (c1count2 == 12) { c_charter1.pic = c_charter1.attackpic[3]; }
					if (c1count2 == 16) { c_charter1.pic = c_charter1.attackpic[4]; }
					if (c1count2 == 20) { c_charter1.pic = c_charter1.attackpic[5]; }
					if (c1count2 == 24) { c_charter1.pic = c_charter1.attackpic[6]; c1isActed2 = true; }
					c1count2++;
					//if (c1count2 >= 24) { c1count2 = 0; }
				}
				else if (judgeAttackPower < 0 && c1isActed2 == false)
				{
					c1_count = 0;
					c1_isActed = false;
					c1isActed2 = true;
				}
				//if (judgeAttackPower > 0)
				if (judgeAttackPower > 0 && c2isActed == false && c1count2 >= 24)
				{
					c2count2 = 0;
					c2isActed2 = false;
					if (c2count == 0)c_charter2.pic = c_charter2.attackedpic[0];
					if (c2count == 4)c_charter2.pic = c_charter2.attackedpic[1];
					if (c2count == 8)c_charter2.pic = c_charter2.attackedpic[2];
					if (c2count == 12)c_charter2.pic = c_charter2.attackedpic[3];
					if (c2count == 16) { c_charter2.pic = c_charter2.attackedpic[4]; c2isActed = true; }
					c2count++;
					//if (c2count >= 16) { c2count = 0; }
				}
				else if (judgeAttackPower < 0 && c2isActed == false)
				{
					c2count2 = 0;
					c2isActed2 = false;
					c2isActed = true;
				}
			}
			c_charter1.onShow();
			c_charter2.onShow();
			endturn.SetTopLeft(600, 850);
			endturn.ShowBitmap();
			/**/
			if ((c_card1.round % 2 == 1 && judgeAttackPower>0 && c1_isActed == false && c2count2 >= 16) ||
(c_card1.round % 2 == 1 && judgeAttackPower < 0)) {
				{
```

```
				CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
				CFont f, *fp;
				char blood_char[32];
				int myattack = 0;
				stringstream ss;
				myattack = -1 * judgeAttackPower;
				if (myattack>0)
				{
					ss << "+";
				}
				ss << myattack;
				ss >> blood_char;
				f.CreatePointFont(200, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
				fp = pDC->SelectObject(&f);				// 選用 font f
				pDC->SetTextColor(RGB(255, 255, 255));
				pDC->SetBkColor(RGB(255, 0, 0));

				if (judgeAttackPower<0)
				{
					for (int i = 0; i < 6000; i++) {
						pDC->TextOut(1000, 250, blood_char);

					}
					judgeAttackPower = 0;
				}
				else {
					pDC->TextOut(500, 250, blood_char);
				}
				pDC->SelectObject(fp);					// 放掉 font f (千萬不要漏了
放掉)
				CDDraw::ReleaseBackCDC();				// 放掉 Back Plain 的 CDC
			}
		}
		if (c_card1.round % 2 == 1 && judgeShield > 0) {
			{
				CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
				CFont f, *fp;
				char blood_char[32];
				stringstream ss;
				ss << "+";
				ss << judgeShield;
				ss >> blood_char;
				f.CreatePointFont(200, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
				fp = pDC->SelectObject(&f);				// 選用 font f
				pDC->SetTextColor(RGB(255, 255, 255));
				pDC->SetBkColor(RGB(0, 255, 0));
				for (int i = 0; i < 6000; i++) {
					pDC->TextOut(1000, 250, blood_char);
				}
				judgeShield = 0;
				pDC->SelectObject(fp);					// 放掉 font f (千萬不要漏了
放掉)
				CDDraw::ReleaseBackCDC();				// 放掉 Back Plain 的 CDC
			}
		}
		if ((c_card1.round % 2 == 0 && judgeAttackPower > 0 && c2isActed == false && c1count2 >= 24) ||
(c_card1.round % 2 == 0 && judgeAttackPower < 0)) {
			{
				CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
				CFont f, *fp;
				char blood_char[32];
				int myattack = 0;
				stringstream ss;
				myattack = -1 * judgeAttackPower;
				if (myattack > 0)
				{
					ss << "+";
				}
				ss << myattack;
```

```
                    ss >> blood_char;
                    f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                    fp = pDC->SelectObject(&f);                          // 選用 font f
                    pDC->SetTextColor(RGB(255, 255, 255));
                    pDC->SetBkColor(RGB(255, 0, 0));

                    if (judgeAttackPower < 0)
                    {
                         for (int i = 0; i < 6000; i++) {
                              pDC->TextOut(500, 250, blood_char);
                         }
                         judgeAttackPower = 0;
                    }
                    else {

                         pDC->TextOut(1000, 250, blood_char);
                    }
                    pDC->SelectObject(fp);                                    // 放掉 font f (千萬不要漏了
放掉)
                    CDDraw::ReleaseBackCDC();                                 // 放掉 Back Plain 的 CDC
               }
          }
          if (c_card1.round % 2 == 0 && judgeShield > 0) {
               {
                    CDC *pDC = CDDraw::GetBackCDC();                 // 取得 Back Plain 的 CDC
                    CFont f, *fp;
                    char blood_char[32];
                    stringstream ss;
                    ss << "+";
                    ss << judgeShield;
                    ss >> blood_char;
                    f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                    fp = pDC->SelectObject(&f);                          // 選用 font f
                    pDC->SetTextColor(RGB(255, 255, 255));
                    pDC->SetBkColor(RGB(0, 255, 0));
                    for (int i = 0; i < 6000; i++) {
                         pDC->TextOut(500, 250, blood_char);
                    }
                    judgeShield = 0;
                    pDC->SelectObject(fp);                                    // 放掉 font f (千萬不要漏了
放掉)
                    CDDraw::ReleaseBackCDC();                                 // 放掉 Back Plain 的 CDC
               }
          }
          /**/
          //c_charter1.fire = c_charter1.getlife = 1;
          //c_charter2.fire = c_charter2.getlife = 10;
          if (c_charter1.fire > 0)
          {
               CDC *pDC1 = CDDraw::GetBackCDC();                 // 取得 Back Plain 的 CDC
               CFont f1, *fp1;
               stringstream ss1;
               ss1 << c_charter1.fire*-1;
               char fire1[32] = "{ CHARTER 1 }";
               ss1 >> fire1;
               f1.CreatePointFont(200, "Times New Roman");     // 產生 font f; 160 表示 16 point 的字
               fp1 = pDC1->SelectObject(&f1);                          // 選用 font f
               pDC1->SetTextColor(RGB(255, 0, 0));
               pDC1->SetBkColor(RGB(255, 255, 255));
               pDC1->TextOut(290, 30, fire1);
               pDC1->SelectObject(fp1);                                  // 放掉 font f(千萬不要漏了放掉)
               CDDraw::ReleaseBackCDC();                                 // 放掉 Back Plain 的 CDC
               {
                    CDC *pDC = CDDraw::GetBackCDC();                 // 取得 Back Plain 的 CDC
                    CFont f, *fp;
                    char blood_char[32]="     燙傷: 下回血量";
                    int myattack = 0;
                    f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                    fp = pDC->SelectObject(&f);                          // 選用 font f
```

```
                pDC->SetTextColor(RGB(255, 0, 0));
                pDC->SetBkColor(RGB(255, 255, 255));
                pDC->TextOut(50, 30, blood_char);
                pDC->SelectObject(fp);                          // 放掉 font f (千萬不要漏了
放掉)
                CDDraw::ReleaseBackCDC();                       // 放掉 Back Plain 的 CDC
            }
            isfire.LoadBitmapA("Bitmaps\\FireMini.bmp", RGB(50, 150, 150));
            isfire.SetTopLeft(12, 12);
            isfire.ShowBitmap();
        }
        if (c_charter2.fire > 0)
        {
            {
                CDC *pDC = CDDraw::GetBackCDC();            // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32] = "            .";
                int myattack = 0;
                f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                 // 選用 font f
                pDC->SetTextColor(RGB(255, 0, 0));
                pDC->SetBkColor(RGB(255, 255, 255));
                pDC->TextOut(1550, 30, blood_char);
                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;
            stringstream ss1;
            ss1 << c_charter2.fire*-1<<"    ";
            char fire1[32] = "{ CHARTER 1 }";
            ss1 >> fire1;
            f1.CreatePointFont(200, "Times New Roman");     // 產生 font f; 160 表示 16 point 的字
            fp1 = pDC1->SelectObject(&f1);                  // 選用 font f
            pDC1->SetTextColor(RGB(255, 0, 0));
            pDC1->SetBkColor(RGB(255, 255, 255));
            pDC1->TextOut(1560, 30, fire1);
            pDC1->SelectObject(fp1);                        // 放掉 font f(千萬不要漏了放掉)
            CDDraw::ReleaseBackCDC();                       // 放掉 Back Plain 的 CDC
            {
                CDC *pDC = CDDraw::GetBackCDC();            // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32] = "燙傷: 下回血量";
                int myattack = 0;
                f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                 // 選用 font f
                pDC->SetTextColor(RGB(255, 0, 0));
                pDC->SetBkColor(RGB(255, 255, 255));
                pDC->TextOut(1340, 30, blood_char);
                pDC->SelectObject(fp);                          // 放掉 font f (千萬不要漏了
放掉)
                CDDraw::ReleaseBackCDC();                       // 放掉 Back Plain 的 CDC
            }
            arefire.LoadBitmapA("Bitmaps\\FireMini.bmp", RGB(50, 150, 150));
            arefire.SetTopLeft(1600, 12);
            arefire.ShowBitmap();
        }
        if (c_charter1.getlife != 0)
        {
            {
                CDC *pDC = CDDraw::GetBackCDC();            // 取得 Back Plain 的 CDC
                CFont f, *fp;
                char blood_char[32] = "    禁止補血";
                int myattack = 0;
                f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                fp = pDC->SelectObject(&f);                 // 選用 font f
                pDC->SetTextColor(RGB(255, 0, 0));
                pDC->SetBkColor(RGB(255, 255, 255));
```

```
                                pDC->TextOut(50, 118, blood_char);
                                pDC->SelectObject(fp);                                  // 放掉 font f (千萬不要漏了
放掉)
                                CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
                        }
                        isban.LoadBitmapA("Bitmaps\\banaddbloodmini.bmp", RGB(50, 150, 150));
                        isban.SetTopLeft(12, 100);
                        isban.ShowBitmap();
                }
                if (c_charter2.getlife != 0)
                {
                        {
                                CDC *pDC = CDDraw::GetBackCDC();                        // 取得 Back Plain 的 CDC
                                CFont f, *fp;
                                char blood_char[32] = "禁止補血   ";
                                int myattack = 0;
                                f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                                fp = pDC->SelectObject(&f);                             // 選用 font f
                                pDC->SetTextColor(RGB(255, 0, 0));
                                pDC->SetBkColor(RGB(255, 255, 255));
                                pDC->TextOut(1465, 118, blood_char);
                                pDC->SelectObject(fp);                                  // 放掉 font f (千萬不要漏了
放掉)
                                CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
                        }
                        areban.LoadBitmapA("Bitmaps\\banaddbloodmini.bmp", RGB(50, 150, 150));
                        areban.SetTopLeft(1600, 100);
                        areban.ShowBitmap();
                }
                /**/
                c_card1.onShow();
                if ((c_card1.round % 2 == 1 && judgeAttackPower > 0 && c1_isActed == false && c2count2 >= 16) ||
(c_card1.round % 2 == 1 && judgeAttackPower < 0)) {
                        {
                                CDC *pDC = CDDraw::GetBackCDC();                        // 取得 Back Plain 的 CDC
                                CFont f, *fp;
                                char blood_char[32];
                                int myattack = 0;
                                stringstream ss;
                                myattack = -1 * judgeAttackPower;
                                if (myattack > 0)
                                {
                                        ss << "+";
                                }
                                ss << myattack;
                                ss >> blood_char;
                                f.CreatePointFont(200, "Times New Roman");      // 產生 font f; 160 表示 16 point 的字
                                fp = pDC->SelectObject(&f);                             // 選用 font f
                                pDC->SetTextColor(RGB(255, 255, 255));
                                pDC->SetBkColor(RGB(255, 0, 0));

                                if (judgeAttackPower < 0)
                                {
                                        for (int i = 0; i < 6000; i++) {
                                                pDC->TextOut(1000, 250, blood_char);

                                        }
                                        judgeAttackPower = 0;
                                }
                                else {
                                        pDC->TextOut(500, 250, blood_char);
                                }
                                pDC->SelectObject(fp);                                  // 放掉 font f (千萬不要漏了
放掉)
                                CDDraw::ReleaseBackCDC();                               // 放掉 Back Plain 的 CDC
                        }
                }
                if (c_card1.round % 2 == 1 && judgeShield > 0) {
                        {
```

```
					CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
					CFont f, *fp;
					char blood_char[32];
					stringstream ss;
					ss << "+";
					ss << judgeShield;
					ss >> blood_char;
					f.CreatePointFont(200, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
					fp = pDC->SelectObject(&f);				// 選用 font f
					pDC->SetTextColor(RGB(255, 255, 255));
					pDC->SetBkColor(RGB(0, 255, 0));
					for (int i = 0; i < 6000; i++) {
						pDC->TextOut(1000, 250, blood_char);
					}
					judgeShield = 0;
					pDC->SelectObject(fp);					// 放掉 font f (千萬不要漏了
放掉)
					CDDraw::ReleaseBackCDC();				// 放掉 Back Plain 的 CDC
				}
			}
			if ((c_card1.round % 2 == 0 && judgeAttackPower > 0 && c2isActed == false && c1count2 >= 24) ||
(c_card1.round % 2 == 0 && judgeAttackPower < 0)) {
				{
					CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
					CFont f, *fp;
					char blood_char[32];
					int myattack = 0;
					stringstream ss;
					myattack = -1 * judgeAttackPower;
					if (myattack > 0)
					{
						ss << "+";
					}
					ss << myattack;
					ss >> blood_char;
					f.CreatePointFont(200, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
					fp = pDC->SelectObject(&f);				// 選用 font f
					pDC->SetTextColor(RGB(255, 255, 255));
					pDC->SetBkColor(RGB(255, 0, 0));

					if (judgeAttackPower < 0)
					{
						for (int i = 0; i < 6000; i++) {
							pDC->TextOut(500, 250, blood_char);
						}
						judgeAttackPower = 0;
					}
					else {

						pDC->TextOut(1000, 250, blood_char);
					}
					pDC->SelectObject(fp);					// 放掉 font f (千萬不要漏了
放掉)
					CDDraw::ReleaseBackCDC();				// 放掉 Back Plain 的 CDC
				}
			}
			if (c_card1.round % 2 == 0 && judgeShield > 0) {
				{
					CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
					CFont f, *fp;
					char blood_char[32];
					stringstream ss;
					ss << "+";
					ss << judgeShield;
					ss >> blood_char;
					f.CreatePointFont(200, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
					fp = pDC->SelectObject(&f);				// 選用 font f
					pDC->SetTextColor(RGB(255, 255, 255));
					pDC->SetBkColor(RGB(0, 255, 0));
					for (int i = 0; i < 6000; i++) {
						pDC->TextOut(500, 250, blood_char);
					}
					judgeShield = 0;
					pDC->SelectObject(fp);					// 放掉 font f (千萬不要漏了
放掉)
					CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC
				}
			}
			/**/

			if ((c_charter2.blood <= 0 || c_charter1.blood <= 0) && ((c1_isActed == true && c2isActed2 == true &&
c1isActed2 == false && c2isActed == false) || (c1_isActed == false && c2isActed2 == false && c1isActed2 == true && c2isActed
== true)))
			{
				if (c_charter1.blood <= 0)
				{
					get_win = 2;
				}
				else if (c_charter2.blood <= 0)
				{
					get_win = 1;
				}
				else
				{
					get_win = 0;
				}
				c_charter1.blood = 30;
				c_charter2.blood = 30;
				c_charter1.shield = 0;
				c_charter2.shield = 0;
				c_charter1.getlife = 0;
				c_charter2.getlife = 0;
				c_charter1.fire = 0;
				c_charter2.fire = 0;
				c_card1.round = 0;
				c_charter1.x = -580;
				c_charter2.x = 1680;
				AttackPower = 0;
				judgeAttackPower = 0;
				CostPower = 0;
				DEF = 0;
				Effect = 0;
				c1_count = 16;
				c1count2 = 0;
				c1isActed2 = false;
				c1_isActed = true;
				c2count = 0;
				c2count2 = 16;
				c2isActed2 = true;
				c2isActed = false;
				oldround = -1;
				countc1round = 0;
				countc2round = 0;
				c_card1.onShow();
				unsigned seed = (unsigned)time(NULL); //random 種子宣告(以時間)
				srand(seed); //用時間取亂數
				c_card1.c_rand = (rand() % 9);
				c_card1.c_rand2 = (rand() % 9);
				c_card1.c_rand3 = (rand() % 9);
				GotoGameState(GAME_STATE_WIN);
			}
			/*BackGround.ShowBitmap();
			c_charter2.OnMove();
			c_charter1.OnMove();
			if (c_card1.round % 2 == 1)
			{
				if (AttackPower > 0 && c1_isActed == false)
				{
```

```
				if (c1_count == 0) { c_charter1.pic = c_charter1.attackedpic[0];}
				if (c1_count == 4) { c_charter1.pic = c_charter1.attackedpic[1]; }
				if (c1_count == 8) { c_charter1.pic = c_charter1.attackedpic[2]; }
				if (c1_count == 12) { c_charter1.pic = c_charter1.attackedpic[3]; }
				if (c1_count == 16) { c_charter1.pic = c_charter1.attackedpic[4]; c1_isActed = true; }
				c1_count++;
			}
		}
		if (c_card1.round % 2 == 0)
		{
			c1_count = 0;
			c1_isActed = false;
		}
		c_charter1.onShow();
		c_charter2.onShow();
		c_card1.onShow();*/
		//  貼上左上及右下角落的圖
		//
		/*顯示回合*/
		if (c_card1.round % 2 == 0)
		{
			showRound[0].SetTopLeft(600, 50);
			showRound[0].ShowBitmap();
			/*CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
			CFont f, *fp;
			char me[32] = "{ CHARTER 1 }";
			f.CreatePointFont(360, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
			fp = pDC->SelectObject(&f);				// 選用 font f
			pDC->SetTextColor(RGB(255, 255, 255));
			pDC->SetBkColor(RGB(255, 0, 0));
			pDC->TextOut(600, 850, me);
			pDC->SelectObject(fp);					// 放掉 font f (千萬不要漏了放掉)
			CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC*/
		}
		else
		{
			showRound[1].SetTopLeft(600, 50);
			showRound[1].ShowBitmap();
			/*
			CDC *pDC = CDDraw::GetBackCDC();			// 取得 Back Plain 的 CDC
			CFont f, *fp;
			char me[32] = "{ CHARTER 2 }";
			f.CreatePointFont(360, "Times New Roman");	// 產生 font f; 160 表示 16 point 的字
			fp = pDC->SelectObject(&f);				// 選用 font f
			pDC->SetTextColor(RGB(255, 255, 255));
			pDC->SetBkColor(RGB(0, 255, 0));
			pDC->TextOut(600, 850, me);
			pDC->SelectObject(fp);					// 放掉 font f (千萬不要漏了放掉)
			CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC*/
		}
		/**/
}
CGameStateWin::CGameStateWin(CGame *g)
	: CGameState(g)
{
}

void CGameStateWin::OnMove()
{
	counter--;
	if (counter < 0)
		GotoGameState(GAME_STATE_INIT);
}

void CGameStateWin::OnBeginState()
{
	CAudio::Instance()->Stop(AUDIO_BGM);
	counter = 30 * 5; // 5 seconds
}
```

```
	void CGameStateWin::OnInit()
	{
		//
		// 當圖很多時，OnInit 載入所有的圖要花很多時間。為避免玩遊戲的人
		//     等的不耐煩，遊戲會出現「Loading ...」，顯示 Loading 的進度。
		//
		ShowInitProgress(66);	// 接個前一個狀態的進度，此處進度視為 66%
		//
		// 開始載入資料
		//

		Sleep(300);				// 放慢，以便看清楚進度，實際遊戲請刪除此 Sleep
		//
		// 最終進度為 100%
		//
		ShowInitProgress(100);
	}

	void CGameStateWin::OnShow()
	{
		init_pass = 0;
		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 16 point 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
		if (get_win == 1)
		{
			if(game_choose==1)
			{
				sprintf(str, "You Win ! (%d)", counter / 30);
			}
			else
			{
				sprintf(str, "Player 1 Win ! (%d)", counter / 30);
			}

		}
		else if (get_win == 2)
		{
			if (game_choose == 1)
			{
				sprintf(str, "You Lose ! (%d)", counter / 30);
			}
			else
			{
				sprintf(str, "Player 2 Win ! (%d)", counter / 30);
			}
		}
		else
		{
			sprintf(str, "平手 ! (%d)", counter / 30);
		}
		//sprintf(str, "Game Over ! (%d)", counter / 30);
		pDC->TextOut(240, 210, str);
		pDC->SelectObject(fp);						// 放掉 font f(千萬不要漏了放掉)
		CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC
	}
}
```

Mygame.h

```
#include "CEraser.h"
#include "CBall.h"
#include "CBouncingBall.h"
#include <stdlib.h>
```

```
namespace game_framework {
	/////////////////////////////////////////////////////////////////////////////
	// Constants
	/////////////////////////////////////////////////////////////////////////////

	enum AUDIO_ID {				// 定義各種音效的編號
		AUDIO_DING,				// 0
		AUDIO_LAKE,				// 1
		AUDIO_NTUT,				// 2
		AUDIO_BGM
	};

	/////////////////////////////////////////////////////////////////////////////
	// 這個 class 為遊戲的遊戲開頭畫面物件
	// 每個 Member function 的 Implementation 都要弄懂
	/////////////////////////////////////////////////////////////////////////////

	class CGameStateInit : public CGameState {
	public:
		CGameStateInit(CGame *g);
		void OnInit();								// 遊戲的初值及圖形設定
		void OnBeginState();							// 設定每次重玩所需的變數
		void OnKeyUp(UINT, UINT, UINT);					// 處理鍵盤 Up 的動作
		void OnLButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	protected:
		void OnShow();									// 顯示這個狀態的遊戲畫面
	private:
		CMovingBitmap logo, BGuse,Game_ch, Game_ch2,about_icon,back;
		// csie 的 logo
	};

	/////////////////////////////////////////////////////////////////////////////
	// 這個 class 為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
	// 每個 Member function 的 Implementation 都要弄懂
	/////////////////////////////////////////////////////////////////////////////



	//角色 1
	class Charter1 {
	public:
		Charter1();
		void LoadBit();
		void OnMove();
		void AttackedMove();
		void onShow();
		int blood;
		int shield;
		int fire, getlife;
		int pow;
		int x, y;
		CMovingBitmap pic;
		CMovingBitmap attackedpic[5];
		CMovingBitmap attackpic[7];
	private:
		//CMovingBitmap pic;


	};
	//
	//角色 2
	class Charter2 {
	public:
		Charter2();
		void LoadBit();
		void OnMove();
		void AttackedMove();
		void onShow();
	int blood;
	int shield;
	int fire, getlife;
	int pow;
	int x, y;
	CMovingBitmap pic;
	CMovingBitmap attackedpic[5];
	CMovingBitmap attackpic[5];
private:
	//CMovingBitmap pic;


};
//
/*卡片資訊*/
class Card_info {
public:
	Card_info();
	CMovingBitmap todo;
	string name;
	int AttackPower,CostPower,DEF,Effect;
};
/**/
//
//卡片
class Card1 {
public:
	Card1();
	void LoadBit();
	void OnMove();
	void onShow();
	int* OnLButtonUp(UINT nFlags, CPoint point, int cost);
	Card_info theCard[22];
	int round = 0;
	int c_rand, c_rand2, c_rand3;
private:
	//CMovingBitmap card[10];
	int x, y;


};
//
class Map1 {
public:
	Map1();
	void LoadBit();
	void onShow();
	void OnLButtonUp(UINT nFlags, CPoint point);
	CMovingBitmap LoadMap[4];
private:
	int x, y;


};
class CGameStateWin : public CGameState {
public:
	CGameStateWin(CGame *g);
	void OnBeginState();					// 設定每次重玩所需的變數
	void OnInit();
protected:
	void OnMove();						// 移動遊戲元素
	void OnShow();						// 顯示這個狀態的遊戲畫面
private:
	int counter;	// 倒數之計數器
};
//
class CGameStateRun : public CGameState {
public:
	CGameStateRun(CGame *g);
```

```
        ~CGameStateRun();
        void OnBeginState();                                    // 設定每次重玩所需的變數
        void OnInit();                                              // 遊戲的初值及圖形設定
        void OnKeyDown(UINT, UINT, UINT);
        void OnKeyUp(UINT, UINT, UINT);
        void OnLButtonDown(UINT nFlags, CPoint point);  // 處理滑鼠的動作
        void OnLButtonUp(UINT nFlags, CPoint point);    // 處理滑鼠的動作
        void OnMouseMove(UINT nFlags, CPoint point);    // 處理滑鼠的動作
        void OnRButtonDown(UINT nFlags, CPoint point);  // 處理滑鼠的動作
        void OnRButtonUp(UINT nFlags, CPoint point);    // 處理滑鼠的動作
protected:
        void OnMove();                                                          // 移動遊戲元素
        void OnShow();                                                          // 顯示這個狀態的遊戲畫面
private:
        const int               NUMBALLS;  // 球的總數
        CMovingBitmap       background;  // 背景圖
        CMovingBitmap       help;            // 說明圖
        CMovingBitmap       BackG;
        CMovingBitmap       BackGround[4];
        CBall                   *ball;           // 球的陣列
        CMovingBitmap     char1;
        Charter1            c_charter1;
        Charter2            c_charter2;
        CMovingBitmap       corner;         // 角落圖
        CEraser                         eraser;         // 拍子
        CInteger                hits_left;      // 剩下的撞擊數
        CBouncingBall     bball;            // 反覆彈跳的球
        Card1               c_card1;
        int attack = 0;
        int AttackPower = 0;
        int judgeAttackPower = 0;
        int judgeShield = 0;
        int CostPower = 0;
        int DEF = 0;
        int Effect = 0;
        int c1_count = 16;
        int c1count2 = 0;
        bool c1isActed2 = false;
        bool c1_isActed = true;
        int c2count = 0;
        int c2count2 = 16;
        bool c2isActed2 = true;
        bool c2isActed = false;
        int oldround = -1;
        int countc1round = 0;
        int countc2round = 0;
        /**/
        CMovingBitmap shield_c1;
        CMovingBitmap shieldline_c1;
        CMovingBitmap shieldpoint_c1[30];
        CMovingBitmap blood_c1;
        CMovingBitmap bloodline_c1;
        CMovingBitmap bloodpoint_c1[30];
        CMovingBitmap dollor_c1;
        CMovingBitmap usedollor_c1[3];
        CMovingBitmap shield_c2;
        CMovingBitmap shieldline_c2;
        CMovingBitmap shieldpoint_c2[30];
        CMovingBitmap blood_c2;
        CMovingBitmap bloodline_c2;
        CMovingBitmap bloodpoint_c2[30];
        CMovingBitmap dollor_c2;
        CMovingBitmap usedollor_c2[3];
        CMovingBitmap showRound[2];
        CMovingBitmap endturn;
};

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

	void OnShow();									// 顯示這個狀態的遊戲畫面
private:
	int counter;	// 倒數之計數器
	CMovingBitmap about,back;
};


class CGameStateDuke : public CGameState {
public:
	CGameStateDuke(CGame *g);
	~CGameStateDuke();
	void OnBeginState();							// 設定每次重玩所需的變數
	void OnInit();									// 遊戲的初值及圖形設定
	void OnKeyDown(UINT, UINT, UINT);
	void OnKeyUp(UINT, UINT, UINT);
	CMovingBitmap BackG;
	void OnLButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnLButtonUp(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnMouseMove(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnRButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnRButtonUp(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	Map1 C_map;
protected:
	void OnMove();									// 移動遊戲元素
	void OnShow();									// 顯示這個狀態的遊戲畫面
private:
	//const int		NUMBALLS;	// 球的總數
	CMovingBitmap	background;	// 背景圖
	CMovingBitmap	help;		// 說明圖
	CBall			*ball;		// 球的陣列
	CMovingBitmap	corner;		// 角落圖
	CEraser				eraser;		// 拍子
	CInteger		hits_left;	// 剩下的撞擊數
	CBouncingBall	bball;		// 反覆彈跳的球
	CMovingBitmap back;
};

class CGameStateCo : public CGameState {
public:
	CGameStateCo(CGame *g);
	~CGameStateCo();
	void OnBeginState();							// 設定每次重玩所需的變數
	void OnInit();									// 遊戲的初值及圖形設定
	void OnKeyDown(UINT, UINT, UINT);
	void OnKeyUp(UINT, UINT, UINT);
	void OnLButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnLButtonUp(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnMouseMove(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnRButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void OnRButtonUp(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	void Player2Action();
protected:
	void OnMove();									// 移動遊戲元素
	void OnShow();									// 顯示這個狀態的遊戲畫面
private:
	const int		NUMBALLS;	// 球的總數
	CMovingBitmap	background;	// 背景圖
	CMovingBitmap	help;		// 說明圖
```

```
		CMovingBitmap		BackG;
		CMovingBitmap		BackGround[4];
		CBall			*ball;		// 球的陣列
		CMovingBitmap	char1;
		Charter1		c_charter1;
		Charter2		c_charter2;
		CMovingBitmap		corner;		// 角落圖
		CEraser				eraser;		// 拍子
		CInteger		hits_left;	// 剩下的撞擊數
		CBouncingBall	bball;		// 反覆彈跳的球
		Card1			c_card1;
		int attack = 0;
		int AttackPower = 0;
		int judgeAttackPower = 0;
		int judgeShield = 0;
		int CostPower = 0;
		int DEF = 0;
		int Effect = 0;
		int c1_count = 16;
		int c1count2 = 0;
		bool c1isActed2 = false;
		bool c1_isActed = true;
		int c2count = 0;
		int c2count2 = 16;
		bool c2isActed2 = true;
		bool c2isActed = false;
		int oldround = -1;
		int countc1round = 0;
		int countc2round = 0;
		/**/
		CMovingBitmap shield_c1;
		CMovingBitmap shieldline_c1;
		CMovingBitmap shieldpoint_c1[30];
		CMovingBitmap blood_c1;
		CMovingBitmap bloodline_c1;
		CMovingBitmap bloodpoint_c1[30];
		CMovingBitmap dollor_c1;
		CMovingBitmap usedollor_c1[3];
		CMovingBitmap shield_c2;
		CMovingBitmap shieldline_c2;
		CMovingBitmap shieldpoint_c2[30];
		CMovingBitmap blood_c2;
		CMovingBitmap bloodline_c2;
		CMovingBitmap bloodpoint_c2[30];
		CMovingBitmap dollor_c2;
		CMovingBitmap usedollor_c2[3];
		CMovingBitmap showRound[2];
		CMovingBitmap endturn;
	};
}
```

3.gamelib.cpp

```
/*
//#define           INITGUID
#include "stdafx.h"
#include "game.h"
#include "MainFrm.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <direct.h>
#include <string.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"

namespace game_framework {

CAnimation::CAnimation(int count)
{
        delay_count = count;
        delay_counter = delay_count;
        x = y = bmp_counter = 0;
}

void CAnimation::AddBitmap(int IDB_BITMAP, COLORREF colorkey)
{
        CMovingBitmap add_bmp;
        add_bmp.LoadBitmap(IDB_BITMAP, colorkey);
        bmp.insert(bmp.end(), add_bmp);
        Reset();
}

void CAnimation::AddBitmap(char *filename, COLORREF colorkey)
{
        CMovingBitmap add_bmp;
        add_bmp.LoadBitmap(filename, colorkey);
        bmp.insert(bmp.end(), add_bmp);
        Reset();
}

int CAnimation::GetCurrentBitmapNumber()
{
        return bmp_counter;
}

int CAnimation::Height()
{
        GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded first.");
        return bmp_iter->Height();
}

bool CAnimation::IsFinalBitmap()
{
        GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded first.");
        return bmp_counter == (bmp.size()-1);
}

int CAnimation::Left()
{
        GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded first.");
        return x;
}

void CAnimation::OnMove()
{
        GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded first.");
        if (--delay_counter <= 0)   {
                delay_counter = delay_count;
                bmp_iter++;
```

```
			bmp_counter++;
			if (bmp_iter == bmp.end()) {
				bmp_iter = bmp.begin();
				bmp_counter = 0;
			}
		}
}

void CAnimation::Reset()
{
		GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded first.");
		delay_counter = delay_count;
		bmp_iter = bmp.begin();
		bmp_counter = 0;
}

void CAnimation::SetDelayCount(int dc)
{
		GAME_ASSERT(dc > 0, "CAnimation: Delay count must be >= 1");
		delay_count = dc;
}

void CAnimation::SetTopLeft(int nx, int ny)
{
		x = nx, y = ny;
		bmp_iter->SetTopLeft(x, y);
}

void CAnimation::OnShow()
{
		GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded before they are shown.");
		bmp_iter->SetTopLeft(x,y);
		bmp_iter->ShowBitmap();
}

int CAnimation::Top()
{
		GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded first.");
		return y;
}

int CAnimation::Width()
{
		GAME_ASSERT(bmp.size() != 0,"CAnimation: Bitmaps must be loaded first.");
		return bmp_iter->Width();
}

/////////////////////////////////////////////////////////////////////////////
// CInteger:  這個 class 提供顯示整數圖形的能力
// 1.  要懂得怎麼呼叫(運用)其各種能力，但是可以不懂下列的程式是什麼意思
// 2.  自己寫到運用 CMovingBitmap 的程式時，可以參考下列程式的寫法
/////////////////////////////////////////////////////////////////////////////

CMovingBitmap CInteger::digit[11];

CInteger::CInteger(int digits)
: NUMDIGITS(digits)
{
		isBmpLoaded = false;
}

void CInteger::Add(int x)
{
		n += x;
}

int CInteger::GetInteger()
{
		return n;
```

```
}

void CInteger::LoadBitmap()
{
	//
	// digit[i]為 class varibale，所以必須避免重複 LoadBitmap
	//
	if (!isBmpLoaded) {
		int d[11]={IDB_0,IDB_1,IDB_2,IDB_3,IDB_4,IDB_5,IDB_6,IDB_7,IDB_8,IDB_9,IDB_MINUS};
		for (int i=0; i < 11; i++)
			digit[i].LoadBitmap(d[i],RGB(255,255,255));
		isBmpLoaded = true;
	}
}

void CInteger::SetInteger(int i)
{
	n = i;
}

void CInteger::SetTopLeft(int nx, int ny)		// 將動畫的左上角座標移至 (x,y)
{
	x = nx; y = ny;
}

void CInteger::ShowBitmap()
{
	GAME_ASSERT(isBmpLoaded, "CInteger: 請先執行 LoadBitmap，然後才能 ShowBitmap");
	int nx;		// 待顯示位數的 x 座標
	int MSB;	// 最左邊(含符號)的位數的數值
	if (n >= 0) {
		MSB = n;
		nx = x+digit[0].Width()*(NUMDIGITS-1);
	} else {
		MSB = -n;
		nx = x+digit[0].Width()*NUMDIGITS;
	}
	for (int i=0; i < NUMDIGITS; i++) {
		int d = MSB % 10;
		MSB /= 10;
		digit[d].SetTopLeft(nx, y);
		digit[d].ShowBitmap();
		nx -= digit[d].Width();
	}
	if (n < 0) { // 如果小於 0，則顯示負號
		digit[10].SetTopLeft(nx, y);
		digit[10].ShowBitmap();
	}
}

/////////////////////////////////////////////////////////////////////////////
// CMovingBitmap: Moving Bitmap class
// 這個 class 提供可以移動的圖形
// 要懂得怎麼呼叫(運用)其各種能力，但是可以不懂下列的程式是什麼意思
/////////////////////////////////////////////////////////////////////////////

CMovingBitmap::CMovingBitmap()
{
	isBitmapLoaded = false;
}

int CMovingBitmap::Height()
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before Height() is called !!!");
	return location.bottom - location.top;
}

int CMovingBitmap::Left()
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before Left() is called !!!");
	return location.left;
}

void CMovingBitmap::LoadBitmap(int IDB_BITMAP, COLORREF color)
{
	const int nx = 0;
	const int ny = 0;
	GAME_ASSERT(!isBitmapLoaded,"A bitmap has been loaded. You can not load another bitmap !!!");
	CBitmap bitmap;
	BOOL rval = bitmap.LoadBitmap(IDB_BITMAP);
	GAME_ASSERT(rval,"Load bitmap failed !!! Please check bitmap ID (IDB_XXX).");
	BITMAP bitmapSize;
	bitmap.GetBitmap(&bitmapSize);
	location.left = nx; location.top = ny;
	location.right = nx+bitmapSize.bmWidth;
	location.bottom = ny+bitmapSize.bmHeight;
	SurfaceID = CDDraw::RegisterBitmap(IDB_BITMAP, color);
	isBitmapLoaded = true;
}

void CMovingBitmap::LoadBitmap(char *filename, COLORREF color)
{
	const int nx = 0;
	const int ny = 0;
	GAME_ASSERT(!isBitmapLoaded,"A bitmap has been loaded. You can not load another bitmap !!!");
	HBITMAP hbitmap = (HBITMAP)LoadImage(NULL,filename,IMAGE_BITMAP,0,0,LR_LOADFROMFILE);
	if (hbitmap == NULL) {
		char error_msg[300];
		sprintf(error_msg, "Loading bitmap from file \"%s\" failed !!!", filename);
		GAME_ASSERT(false, error_msg);
	}
	CBitmap *bmp = CBitmap::FromHandle(hbitmap ); // memory will be deleted automatically
	BITMAP bitmapSize;
	bmp->GetBitmap(&bitmapSize);
	location.left = nx; location.top = ny;
	location.right = nx+bitmapSize.bmWidth;
	location.bottom = ny+bitmapSize.bmHeight;
	SurfaceID = CDDraw::RegisterBitmap(filename, color);
	isBitmapLoaded = true;
}

void CMovingBitmap::SetTopLeft(int x, int y)
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before SetTopLeft() is called !!!");
	int dx = location.left - x;
	int dy = location.top - y;
	location.left = x;
	location.top = y;
	location.right -= dx;
	location.bottom -= dy;
}

void CMovingBitmap::ShowBitmap()
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before ShowBitmap() is called !!!");
	CDDraw::BltBitmapToBack(SurfaceID,location.left,location.top);
}

void CMovingBitmap::ShowBitmap(double factor)
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before ShowBitmap() is called !!!");
	CDDraw::BltBitmapToBack(SurfaceID,location.left,location.top,factor);
}

void CMovingBitmap::ShowBitmap(CMovingBitmap &bm)
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before ShowBitmap() is called !!!");
	GAME_ASSERT(bm.isBitmapLoaded,"A bitmap must be loaded before ShowBitmap() is called !!!");
	CDDraw::BltBitmapToBitmap(SurfaceID, bm.SurfaceID, location.left,location.top);
}

int CMovingBitmap::Top()
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before Top() is called !!!");
	return location.top;
}

int CMovingBitmap::Width()
{
	GAME_ASSERT(isBitmapLoaded,"A bitmap must be loaded before Width() is called !!!");
	return location.right - location.left;
}

/////////////////////////////////////////////////////////////////////////////
// 這個 class 為遊戲的各種狀態之 Base class(是一個 abstract class)
/////////////////////////////////////////////////////////////////////////////

CGameState::CGameState(CGame *g)
{
	game = g;	// 設定 game 的 pointer
}

void CGameState::GotoGameState(int state)
{
	game->SetGameState(state);
}

void CGameState::ShowInitProgress(int percent)
{
	if (!SHOW_LOAD_PROGRESS)
		return;
	const int bar_width = (SIZE_X+100) * 2 / 3;
	const int bar_height = SIZE_Y / 20;
	const int x1 = (SIZE_X+100 - bar_width) / 2;
	const int x2 = x1 + bar_width;
	const int y1 = (SIZE_Y - bar_height) / 2;
	const int y2 = y1 + bar_height;
	const int pen_width = bar_height / 8;
	const int progress_x1 = x1 + pen_width;
	const int progress_x2 = progress_x1 + percent * (bar_width-2*pen_width) / 100;
	const int progress_x2_end = x2 - pen_width;
	const int progress_y1 = y1 + pen_width;
	const int progress_y2 = y2 - pen_width;

	CDDraw::BltBackColor(DEFAULT_BG_COLOR);		// 將 Back Plain 塗上預設的顏色
	CMovingBitmap loading;				// 貼上 loading 圖示
	loading.LoadBitmap(IDB_LOADING, RGB(0,0,0));
	loading.SetTopLeft((SIZE_X - loading.Width())/2, y1 - 2 * loading.Height());
	loading.ShowBitmap();
	//
	// 以下為 CDC 的用法
	//
	CDC *pDC = CDDraw::GetBackCDC();		// 取得 Back Plain 的 CDC
	CPen *pp, p(PS_NULL, 0, RGB(0,0,0));		// 清除 pen
	pp = pDC->SelectObject(&p);

	CBrush *pb, b(RGB(0,255,0));			// 畫綠色 progress 框
	pb = pDC->SelectObject(&b);
	pDC->Rectangle(x1,y1,x2,y2);

	CBrush b1(DEFAULT_BG_COLOR);			// 畫黑色 progrss 中心
	pDC->SelectObject(&b1);
	pDC->Rectangle(progress_x1,progress_y1,progress_x2_end,progress_y2);

	CBrush b2(RGB(255,255,0));			// 畫黃色 progrss 進度
	pDC->SelectObject(&b2);
	pDC->Rectangle(progress_x1,progress_y1,progress_x2,progress_y2);
	pDC->SelectObject(pp);						// 釋放 pen
	pDC->SelectObject(pb);						// 釋放 brush
	CDDraw::ReleaseBackCDC();					// 放掉 Back Plain 的 CDC
	//
	// 如果是別的地方用到 CDC 的話，不要抄以下這行，否則螢幕會閃爍
	//
	CDDraw::BltBackToPrimary();					// 將 Back Plain 貼到螢幕
}

void CGameState::OnDraw() // Template Method
{
	OnShow();
}

void CGameState::OnCycle() // Template Method
{
	OnMove();
	OnShow();
}

/////////////////////////////////////////////////////////////////////////////
// CGame: Game Class
// 這個 class 是遊戲的 facade，是 MFC 與各個遊戲狀態的橋樑，如果不增加或減少
// 遊戲狀態的話，可以不用管這個 class 的介面與實作。
/////////////////////////////////////////////////////////////////////////////

CGame CGame::instance;

CGame::CGame()
: NUM_GAME_STATES(6)
{
	running = true;
	suspended = false;
	gameStateTable[GAME_STATE_INIT] = new CGameStateInit(this);
	gameStateTable[GAME_STATE_RUN]    = new CGameStateRun(this);
	gameStateTable[GAME_STATE_OVER] = new CGameStateOver(this);
	gameStateTable[GAME_STATE_DUKE] = new CGameStateDuke(this);
	gameStateTable[GAME_STATE_CO] = new CGameStateCo(this);
	gameStateTable[GAME_STATE_WIN]=new CGameStateWin(this);
	gameState = NULL;
}

CGame::~CGame()
{
	for (int i = 0; i < NUM_GAME_STATES; i++)
		delete gameStateTable[i];
}

CGame *CGame::Instance()
{
	return &instance;
}

bool CGame::IsRunning()
{
	return running;
}

void CGame::OnDraw()
{
	CDDraw::BltBackColor(DEFAULT_BG_COLOR); // 將 Back Plain 塗黑
	gameState->OnDraw();					// 顯示遊戲中的每個元素
	if (!running) {
		//
		// 如果在暫停狀態，則顯示 Ctrl-Q...
		//
		CMovingBitmap bmp;
		bmp.LoadBitmap(IDB_CONTINUE);
```

```
			bmp.SetTopLeft(0,0);
			bmp.ShowBitmap();
		}
		CDDraw::BltBackToPrimary();				// 將 Back Plain 貼到螢幕
}

void	CGame::OnFilePause()
{
		if (ENABLE_GAME_PAUSE) {
			if (running)
				CAudio::Instance()->Pause();
			else
				CAudio::Instance()->Resume();
			running = !running;
		} else {
			CAudio::Instance()->Resume();
			running = true;
		}
}

bool CGame::OnIdle()	// 修改功能不要修改 OnIdle()，而應修改 OnMove()及 OnShow()
{
		if (suspended) {
			running = false;
			suspended = false;
		}
		//
		// 控制遊戲是否暫停
		//
		if (!running)
			return false;
		//
		// 以下是遊戲的主迴圈
		//
		CDDraw::BltBackColor(DEFAULT_BG_COLOR); // 將 Back Plain 塗上預設的顏色
		gameState->OnCycle();
		CDDraw::BltBackToPrimary();				// 將 Back Plain 貼到螢幕
		//
		// 以下的程式控制遊戲進行的速度，注意事項：
		// 1. 用 Debug mode 可以檢視每一次迴圈花掉的時間，令此時間為 t。
		// 2. 從上次離開 OnIdle()至此，時間定為 33ms，不可刪除，其時間不可低於 t。
		//
		if (SHOW_GAME_CYCLE_TIME)
			TRACE("Ellipse		time		for		the		%d		th		cycle=%d		\n",
CSpecialEffect::GetCurrentTimeCount(),CSpecialEffect::GetEllipseTime());
		CSpecialEffect::DelayFromSetCurrentTime(GAME_CYCLE_TIME);
		CSpecialEffect::SetCurrentTime();	// 設定離開 OnIdle()的時間
		return true;
}

void CGame::OnInit()// OnInit() 只在程式一開始時執行一次
{
		//
		// 啟動亂數
		//
		srand((unsigned)time(NULL));
		//
		// 開啟 DirectX 繪圖介面
		//
		CDDraw::Init(SIZE_X, SIZE_Y);						// 設定遊戲解析度
		//
		// 開啟 DirectX 音效介面
		//
		if (!CAudio::Instance()->Open())				// 開啟音效介面
			AfxMessageBox("Audio Interface Failed (muted)");// 無音效介面
		//
		// Switch to the first state
		//
		gameState = gameStateTable[GAME_STATE_INIT];
		gameState->OnBeginState();
		CSpecialEffect::SetCurrentTime();
		running = true;
}

void CGame::OnInitStates()
{
		//
		// 呼叫每個狀態的 OnInitialUpdate
		//
		for (int i = 0; i < NUM_GAME_STATES; i++)
				gameStateTable[i]->OnInit();
}

void CGame::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
		if (running)
				if ((nFlags & 0x4000) == 0) // 去除 auto repeat
						gameState->OnKeyDown(nChar, nRepCnt, nFlags);
#ifdef _UNITTEST								// invike unit test if _UNITTEST is defined
		void runTest();
		if (nChar == 'T')
				runTest();
#endif
}

void CGame::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
		if (running)
				gameState->OnKeyUp(nChar, nRepCnt, nFlags);
}

void CGame::OnKillFocus()
{
		CAudio::Instance()->Pause();
		if (ENABLE_GAME_PAUSE)
				running = false;
		else if (CDDraw::IsFullScreen())
				running = false;
}

void CGame::OnLButtonDown(UINT nFlags, CPoint point)
{
		if (running)
				gameState->OnLButtonDown(nFlags, point);
}

void CGame::OnRButtonDown(UINT nFlags, CPoint point)
{
		if (running)
				gameState->OnRButtonDown(nFlags, point);
}

void CGame::OnLButtonUp(UINT nFlags, CPoint point)
{
		if (running)
				gameState->OnLButtonUp(nFlags, point);
}

void CGame::OnMouseMove(UINT nFlags, CPoint point)
{
		if (running)
				gameState->OnMouseMove(nFlags, point);
}

void CGame::OnRButtonUp(UINT nFlags, CPoint point)
{
		if (running)
				gameState->OnRButtonUp(nFlags, point);
```

```
}

void CGame::OnResume()
{
	//
	// Note: the resume message is not synchronized with the other messages
	//
}

void CGame::OnSetFocus()
{
	if (!ENABLE_GAME_PAUSE) {
		CAudio::Instance()->Resume();
		running = true;
	}
}

void CGame::OnSuspend()
{
	//
	// Note: the suspend message is not synchronized with the other messages
	//
	suspended = true;
	CAudio::Instance()->SetPowerResume();
}

void CGame::SetGameState(int state)
{
	ASSERT(state >=0 && state < NUM_GAME_STATES);
	gameState = gameStateTable[state];
	gameState->OnBeginState();
	OnDraw();
	CSpecialEffect::SetCurrentTime();
	running = true;
}

/////////////////////////////////////////////////////////////////////////////
// CSpecialEffect: Specail Effect functions
// 一般的遊戲並不需直接操作這個物件，因此可以全部略過不看
/////////////////////////////////////////////////////////////////////////////

DWORD CSpecialEffect::ctime=0;
int	CSpecialEffect::ctimeCount=0;

void CSpecialEffect::Delay(DWORD ms)
{
	DWORD ctime = timeGetTime();
	int waitms;
	waitms = ms - (timeGetTime() - ctime);
	if (waitms > 0)
		Sleep(waitms);
}

void CSpecialEffect::DelayFromSetCurrentTime(DWORD ms)
{
	int waitms;
	waitms = ms - (timeGetTime() - ctime);
	if (waitms > 0)
		Sleep(waitms);
}

void CSpecialEffect::SetCurrentTime()
{
	ctime = timeGetTime();
	ctimeCount++;
}

DWORD CSpecialEffect::GetEllipseTime()
{
	return timeGetTime()-ctime;
}

int CSpecialEffect::GetCurrentTimeCount()
{
	return ctimeCount;
}

/////////////////////////////////////////////////////////////////////////////
// CDDraw: Direct Draw Object
// 這個 class 會建立 DirectDraw 物件，以提供其他 class 使用
// 這個 class 的全部程式都是低階的繪圖介面，可以全部略過不看
/////////////////////////////////////////////////////////////////////////////

HDC					CDDraw::hdc;
CDC					CDDraw::cdc;
CView				*CDDraw::pCView;
LPDIRECTDRAW2		CDDraw::lpDD;
LPDIRECTDRAWCLIPPER	CDDraw::lpClipperPrimary;
LPDIRECTDRAWCLIPPER	CDDraw::lpClipperBack;
LPDIRECTDRAWSURFACE		CDDraw::lpDDSPrimary;
LPDIRECTDRAWSURFACE		CDDraw::lpDDSBack;
HRESULT				CDDraw::ddrval;
bool				CDDraw::fullscreen;
int					CDDraw::size_x, CDDraw::size_y;
CDDraw				CDDraw::ddraw;
vector<int>			CDDraw::BitmapID;
vector<string>		CDDraw::BitmapName;
vector<CRect>			CDDraw::BitmapRect;
vector<COLORREF>		CDDraw::BitmapColorKey;
vector<LPDIRECTDRAWSURFACE>	CDDraw::lpDDS;

CDDraw::CDDraw()
{
	pCView=NULL;
	lpClipperPrimary=lpClipperBack=NULL;
	lpDDSPrimary=lpDDSBack=NULL;
	fullscreen = OPEN_AS_FULLSCREEN;
	lpDD=NULL;
}

CDDraw::~CDDraw()
{
	ReleaseSurface();
	if (lpDD)
		lpDD->Release();
	pCView=NULL; lpDD=NULL;
	lpClipperPrimary=lpClipperBack=NULL;
	lpDDSPrimary=lpDDSBack=NULL;
	TRACE("~CDDraw()\n");
}

void CDDraw::BltBackColor(DWORD color)
{
	if (lpDDSBack) {
		if (lpDDSBack->IsLost())
			RestoreSurface();
		DDBLTFX ddbltfx;
		ddbltfx.dwSize = sizeof(ddbltfx);
		ddbltfx.dwFillColor = MatchColorKey(lpDDSBack,color);
		ddrval = lpDDSBack->Blt(NULL,NULL,NULL,DDBLT_COLORFILL|DDBLT_WAIT,&ddbltfx);
		CheckDDFail("BltBackColor: blitting failed");
	}
}

void CDDraw::BltBackToPrimary()
{
	if (!pCView || !lpDDSPrimary || !lpDDSBack)
		return;
	if (fullscreen) {
		CRect ClientRect;
		GetClientRect(ClientRect);
		if (lpDDSBack->IsLost())
			RestoreSurface();
		if (lpDDSPrimary->IsLost())
			RestoreSurface();
		ddrval = lpDDSPrimary->Blt(ClientRect, lpDDSBack, ClientRect, DDBLTFAST_WAIT, NULL);
		CheckDDFail("Blt Back to primary failed");
	} else {
		if (lpDDSBack->IsLost())
			RestoreSurface();
		if (lpDDSPrimary->IsLost())
			RestoreSurface();
		CRect ClientRect;
		GetClientRect(ClientRect);
		CRect WindowRect;
		pCView->GetWindowRect(&WindowRect);
		WindowRect.right = WindowRect.left + size_x;
		WindowRect.bottom = WindowRect.top + size_y;
		ddrval = lpDDSPrimary->Blt(WindowRect, lpDDSBack, ClientRect, DDBLTFAST_WAIT, NULL);
		CheckDDFail("Blt Back to primary failed");
	}
}

void CDDraw::BltBitmapToBack(unsigned SurfaceID, int x, int y)
{
	GAME_ASSERT(lpDDSBack && (SurfaceID < lpDDS.size()) && lpDDS[SurfaceID], "Internal Error: Incorrect
SurfaceID in BltBitmapToBack");
	CRect TargetRect;
	TargetRect.left = x;
	TargetRect.top = y;
	TargetRect.right = x + BitmapRect[SurfaceID].right-BitmapRect[SurfaceID].left;
	TargetRect.bottom = y + BitmapRect[SurfaceID].bottom-BitmapRect[SurfaceID].top;
	int blt_flag;
	if (BitmapColorKey[SurfaceID] != CLR_INVALID)
		blt_flag = DDBLT_WAIT | DDBLT_KEYSRC;
	else
		blt_flag = DDBLT_WAIT;
	if (lpDDSBack->IsLost())
		RestoreSurface();
	if (lpDDS[SurfaceID]->IsLost())
		RestoreSurface();
	ddrval = lpDDSBack->Blt(TargetRect, lpDDS[SurfaceID],NULL, blt_flag, NULL);
	CheckDDFail("Blt Bitmap to Back Failed");
}

void CDDraw::BltBitmapToBack(unsigned SurfaceID, int x, int y, double factor)
{
	GAME_ASSERT(lpDDSBack && (SurfaceID < lpDDS.size()) && lpDDS[SurfaceID], "Internal Error: Incorrect
SurfaceID in BltBitmapToBack");
	CRect TargetRect;
	TargetRect.left = x;
	TargetRect.top = y;
	TargetRect.right = x + (int) ((BitmapRect[SurfaceID].right-BitmapRect[SurfaceID].left)*factor);
	TargetRect.bottom = y + (int) ((BitmapRect[SurfaceID].bottom-BitmapRect[SurfaceID].top)*factor);
	int blt_flag;
	if (BitmapColorKey[SurfaceID] != CLR_INVALID)
		blt_flag = DDBLT_WAIT | DDBLT_KEYSRC;
	else
		blt_flag = DDBLT_WAIT;
	if (lpDDSBack->IsLost())
		RestoreSurface();
	if (lpDDS[SurfaceID]->IsLost())
		RestoreSurface();
	ddrval = lpDDSBack->Blt(TargetRect, lpDDS[SurfaceID],NULL, blt_flag, NULL);
	CheckDDFail("Blt Bitmap to Back Failed");
}
```

```
void CDDraw::BltBitmapToBitmap(unsigned SourceID, unsigned TargetID, int x, int y)
{
	GAME_ASSERT((SourceID < lpDDS.size()) && (TargetID < lpDDS.size()) && (SourceID != TargetID), "Internal Error: 
Incorrect SourceID in BltBitmapToBitmap");
	int blt_flag;
	if (BitmapColorKey[SourceID] != CLR_INVALID)
		blt_flag = DDBLTFAST_WAIT | DDBLTFAST_SRCCOLORKEY;
	else
		blt_flag = DDBLTFAST_WAIT;
	if (lpDDS[SourceID]->IsLost())
		RestoreSurface();
	if (lpDDS[TargetID]->IsLost())
		RestoreSurface();
	ddrval = lpDDS[TargetID]->BltFast(x, y, lpDDS[SourceID], NULL, blt_flag );
	CheckDDFail("Blt Bitmap to Back Failed");
}

bool CDDraw::CreateSurface()
{
	//
	// Since all surfaces are created and loaded, this operation can be very slow.
	//
	SetCursor(AfxGetApp()->LoadStandardCursor(IDC_WAIT));
	ReleaseSurface();
	if (fullscreen) {
		if (!CreateSurfaceFullScreen())
			return false;
	} else {
		if (!CreateSurfaceWindowed())
			return false;
	}
	for (unsigned i = 0; i < lpDDS.size(); i++) {
		if (BitmapID[i] != -1) // from resource
			LoadBitmap(i, BitmapID[i]);
		else
			LoadBitmap(i, (char *) BitmapName[i].c_str()); // from file
		SetColorKey(i, BitmapColorKey[i]);
	}
	return true;
}

bool CDDraw::CreateSurfaceFullScreen()
{
    ddrval = lpDD->SetCooperativeLevel(AfxGetMainWnd()->m_hWnd, DDSCL_EXCLUSIVE | DDSCL_FULLSCREEN);
    CheckDDFail("Can not SetCooperativeLevel Exclusive");
    ddrval = lpDD->SetDisplayMode(size_x, size_y, 32, 0, 0);
	if (ddrval != DD_OK) {
	      ddrval = lpDD->SetCooperativeLevel(AfxGetMainWnd()->m_hWnd, DDSCL_NORMAL);
		CheckDDFail("Can not SetCooperativeLevel Normal");
		return false;
	}
	CheckDDFail("SetDisplayMode FAILED");

	DDSURFACEDESC ddsd;
	ZeroMemory(&ddsd,sizeof(ddsd));
    ddsd.dwSize = sizeof(ddsd);
    ddsd.dwFlags = DDSD_CAPS;
    ddsd.ddsCaps.dwCaps = DDSCAPS_PRIMARYSURFACE;
    ddrval = lpDD->CreateSurface( &ddsd, &lpDDSPrimary, NULL );
	CheckDDFail("Create Primary Surface failed");

	// Create clippers for the primary and back surfaces
    // ddrval = lpDD->CreateClipper(0, &lpClipperPrimary, NULL);
	// CheckDDFail("Create Primay Surface Clipper FAILED");
    ddrval = lpDD->CreateClipper(0, &lpClipperBack, NULL);
	CheckDDFail("Create Back Surface Clipper FAILED");

	typedef struct {
		RGNDATAHEADER hdr;
```

```
                RECT rgndata[4];
        } CLIPLIST, *LPCLIPLIST;
            CLIPLIST        ClipList;
            RECT rc;
            SetRect(&rc, 0, 0, size_x, size_y);
        ClipList.hdr.dwSize = sizeof(RGNDATAHEADER);
        ClipList.hdr.iType = RDH_RECTANGLES;
            ClipList.hdr.nCount = 1;
        ClipList.hdr.nRgnSize = 0;
        memcpy(&ClipList.hdr.rcBound, &rc, sizeof(RECT));
        memcpy(&ClipList.rgndata, &rc, sizeof(RECT));
            ddrval = lpClipperBack->SetClipList((LPRGNDATA)&ClipList,0);
            CheckDDFail("SetHWnd FAILED");

            // Create Back (Secondary) Surface
            ddsd.dwFlags = DDSD_CAPS | DDSD_HEIGHT | DDSD_WIDTH;
        ddsd.ddsCaps.dwCaps = DDSCAPS_OFFSCREENPLAIN;
            ddsd.dwHeight = size_y; ddsd.dwWidth = size_x;
        ddrval = lpDD->CreateSurface( &ddsd, &lpDDSBack, NULL );
            CheckDDFail("Can not create back plain");

            // In fullscreen mode, the clipper for primary surafce is fixed to screen size
            ddrval = lpDDSPrimary->SetClipper(lpClipperBack);
            CheckDDFail("SetClipper FAILED");
        ddrval = lpDDSBack->SetClipper(lpClipperBack);
            CheckDDFail("SetClipper FAILED");
            lpClipperBack->Release();

            BltBackColor(RGB(0,0,0));
            BltBackToPrimary();
            return true;
}

bool CDDraw::CreateSurfaceWindowed()
{
        ddrval = lpDD->SetCooperativeLevel(AfxGetMainWnd()->m_hWnd, DDSCL_NORMAL);
        CheckDDFail("Can not SetCooperativeLevel ");

            DDSURFACEDESC ddsd;
            ZeroMemory(&ddsd,sizeof(ddsd));
        ddsd.dwSize = sizeof(ddsd);
        ddsd.dwFlags = DDSD_CAPS;
        ddsd.ddsCaps.dwCaps = DDSCAPS_PRIMARYSURFACE;
        ddrval = lpDD->CreateSurface( &ddsd, &lpDDSPrimary, NULL );
            CheckDDFail("Create Primary Surface failed");

            // Create clippers for the primary and back surfaces
        ddrval = lpDD->CreateClipper(0, &lpClipperPrimary, NULL);
            CheckDDFail("Create Primay Surface Clipper FAILED");
        ddrval = lpDD->CreateClipper(0, &lpClipperBack, NULL);
            CheckDDFail("Create Back Surface Clipper FAILED");

        ddrval = lpClipperPrimary->SetHWnd(0, AfxGetMainWnd()->m_hWnd);
            CheckDDFail("Primary Surface SetHWnd FAILED");

            typedef struct {
                RGNDATAHEADER hdr;
                RECT rgndata[4];
        } CLIPLIST, *LPCLIPLIST;
            CLIPLIST        ClipList;
            RECT rc;
            SetRect(&rc, 0, 0, size_x, size_y);
        ClipList.hdr.dwSize = sizeof(RGNDATAHEADER);
        ClipList.hdr.iType = RDH_RECTANGLES;
            ClipList.hdr.nCount = 1;
        ClipList.hdr.nRgnSize = 0;
        memcpy(&ClipList.hdr.rcBound, &rc, sizeof(RECT));
        memcpy(&ClipList.rgndata, &rc, sizeof(RECT));
            ddrval = lpClipperBack->SetClipList((LPRGNDATA)&ClipList,0);
```

```
	CheckDDFail("SetHWnd FAILED");

	// Create Back (Secondary) Surface
	ddsd.dwFlags = DDSD_CAPS | DDSD_HEIGHT | DDSD_WIDTH;
    ddsd.ddsCaps.dwCaps = DDSCAPS_OFFSCREENPLAIN;
	ddsd.dwHeight = size_y; ddsd.dwWidth = size_x;
    ddrval = lpDD->CreateSurface( &ddsd, &lpDDSBack, NULL );
	CheckDDFail("Can not create back plain");

	ddrval = lpDDSPrimary->SetClipper(lpClipperPrimary);
	CheckDDFail("SetClipper FAILED");
	lpClipperPrimary->Release();
    ddrval = lpDDSBack->SetClipper(lpClipperBack);
	CheckDDFail("SetClipper FAILED");
	lpClipperBack->Release();

	BltBackColor(RGB(0,0,0));
	return true;
}

void CDDraw::GetClientRect(CRect &r)
{
	r = CRect(0,0,size_x,size_y);
}

void CDDraw::Init(int sx, int sy)
{
	// set target screen size
	size_x = sx, size_y = sy;
	// init lpDD
	LPDIRECTDRAW lpDD0;
    ddrval = DirectDrawCreate(NULL, &lpDD0, NULL);
	CheckDDFail("DDraw create failed");
	ddrval = lpDD0->QueryInterface(IID_IDirectDraw2, (void **) &lpDD);
	CheckDDFail("DDraw surface 2 create failed");
	// init pCView
	POSITION pos = AfxGetApp()->GetFirstDocTemplatePosition();
	CDocTemplate *doc_t = AfxGetApp()->GetNextDocTemplate(pos);
	pos = doc_t->GetFirstDocPosition();
	CDocument *doc = doc_t->GetNextDoc(pos);
	pos = doc->GetFirstViewPosition();
	pCView = doc->GetNextView(pos);
	// init surfaces
	SetFullScreen(fullscreen);
}

bool CDDraw::IsFullScreen()
{
	return fullscreen;
}

void CDDraw::LoadBitmap(int i, int IDB_BITMAP)
{
	CBitmap bitmap;
	bitmap.LoadBitmap(IDB_BITMAP);
	CDC mDC;
	mDC.CreateCompatibleDC(NULL);
	CBitmap* pOldBitmap = mDC.SelectObject(&bitmap);
	BITMAP bitmapSize;
	bitmap.GetBitmap(&bitmapSize);
	DDSURFACEDESC ddsd;
	ZeroMemory(&ddsd,sizeof(ddsd));
	ddsd.dwSize = sizeof( ddsd );
	ddsd.dwFlags = DDSD_CAPS | DDSD_HEIGHT | DDSD_WIDTH;
	ddsd.ddsCaps.dwCaps = DDSCAPS_OFFSCREENPLAIN;
	BitmapRect[i].bottom = ddsd.dwHeight = bitmapSize.bmHeight;
	BitmapRect[i].right = ddsd.dwWidth = bitmapSize.bmWidth;
	ddrval = lpDD->CreateSurface(&ddsd, &lpDDS[i], NULL);
	CheckDDFail("Create Bitmap Surface Failed");
```

```
        HDC hdc;
    ddrval= lpDDS[i]->GetDC(&hdc);
        CheckDDFail("Get surface HDC failed");
        CDC cdc;
        cdc.Attach(hdc);
        cdc.BitBlt(0,0,bitmapSize.bmWidth,bitmapSize.bmHeight, &mDC,0,0,SRCCOPY);
        cdc.Detach();
        lpDDS[i]->ReleaseDC(hdc);
        // avoid memory leak
        // According to spec, mDC should delete itself automatically.    However,
        // it appears that we have to do it explictly.
        mDC.SelectObject(&pOldBitmap);
        mDC.DeleteDC();
        bitmap.DeleteObject();
}

void CDDraw::LoadBitmap(int i, char *filename)
{

        HBITMAP hbitmap = (HBITMAP)LoadImage(NULL,filename,IMAGE_BITMAP,0,0,LR_LOADFROMFILE);
        GAME_ASSERT(hbitmap != NULL,"Load bitmap failed !!! Please check bitmap ID (IDB_XXX).");
        CBitmap *bmp = CBitmap::FromHandle(hbitmap ); // will be deleted automatically
        CDC mDC;
        mDC.CreateCompatibleDC(NULL);
        CBitmap* pOldBitmap = mDC.SelectObject(bmp);
        BITMAP bitmapSize;
        bmp->GetBitmap(&bitmapSize);
        DDSURFACEDESC ddsd;
        ZeroMemory(&ddsd,sizeof(ddsd));
        ddsd.dwSize = sizeof( ddsd );
        ddsd.dwFlags = DDSD_CAPS | DDSD_HEIGHT | DDSD_WIDTH;
        ddsd.ddsCaps.dwCaps = DDSCAPS_OFFSCREENPLAIN;
        BitmapRect[i].bottom = ddsd.dwHeight = bitmapSize.bmHeight;
        BitmapRect[i].right = ddsd.dwWidth = bitmapSize.bmWidth;
        ddrval = lpDD->CreateSurface(&ddsd, &lpDDS[i], NULL);
        CheckDDFail("Create Bitmap Surface Failed");
        HDC hdc;
    ddrval= lpDDS[i]->GetDC(&hdc);
        CheckDDFail("Get surface HDC failed");
        CDC cdc;
        cdc.Attach(hdc);
        cdc.BitBlt(0,0,bitmapSize.bmWidth,bitmapSize.bmHeight, &mDC,0,0,SRCCOPY);
        cdc.Detach();
        lpDDS[i]->ReleaseDC(hdc);
        // avoid memory leak
        // According to spec, mDC should delete itself automatically.    However,
        // it appears that we have to do it explictly.
        mDC.SelectObject(&pOldBitmap);
        mDC.DeleteDC();
        bmp->DeleteObject();
}

DWORD CDDraw::MatchColorKey(LPDIRECTDRAWSURFACE lpDDSurface, COLORREF color)
{
    DDSURFACEDESC ddsd;
        HDC hdc;
        HRESULT hres;
        COLORREF rgbT= CLR_INVALID;
        DWORD dw=CLR_INVALID,mask=(DWORD)~0;
        if (lpDDSurface && color != CLR_INVALID) {
                if (lpDDSurface->GetDC(&hdc) == DD_OK) {
                        rgbT = GetPixel(hdc, 0, 0);                 // save (0,0) pixel value
                        SetPixel(hdc, 0, 0, color);                 // set our value
                        lpDDSurface->ReleaseDC(hdc);
                }
                ddsd.dwSize = sizeof(ddsd);
                while ((hres = lpDDSurface->Lock(NULL, &ddsd, 0, NULL)) == DDERR_WASSTILLDRAWING);
                if (hres == DD_OK) {
                        dw = *(DWORD *)ddsd.lpSurface;              // get (0,0) data
				if (ddsd.ddpfPixelFormat.dwRGBBitCount < 32)
					mask = (1 << ddsd.ddpfPixelFormat.dwRGBBitCount)-1;
				dw &= mask;								// mask it to bpp
				lpDDSurface->Unlock(NULL);
			}
			if (lpDDSurface->GetDC(&hdc) == DD_OK)		{
				SetPixel(hdc, 0, 0, rgbT);					// restore (0,0) pixel value
				lpDDSurface->ReleaseDC(hdc);
			}
		}
		return dw;
}

CDC* CDDraw::GetBackCDC()
{
		if (lpDDSBack->IsLost())
			RestoreSurface();
	   ddrval= lpDDSBack->GetDC(&hdc);
		CheckDDFail("Get back surface HDC failed");
		cdc.Attach(hdc);
		return &cdc;
}

int CDDraw::RegisterBitmap(int IDB_BITMAP, COLORREF ColorKey)
{
		unsigned i;
		for (i = 0; i < lpDDS.size(); i++)
			if (BitmapID[i] == IDB_BITMAP)
				return i;
		//
		// Enlarge the size of vectors
		//
		BitmapID.push_back(IDB_BITMAP);
		BitmapName.push_back("");
		BitmapColorKey.push_back(ColorKey);
		BitmapRect.push_back(CRect(0,0,0,0));
		lpDDS.push_back(NULL);
		LoadBitmap(i, IDB_BITMAP);
		SetColorKey(i, ColorKey);
		return i;
}

int CDDraw::RegisterBitmap(char *filename, COLORREF ColorKey)
{
		unsigned i;
		for (i = 0; i < lpDDS.size(); i++)
			if (BitmapName[i].compare(filename) == 0)
				return i;
		//
		// Enlarge the size of vectors
		//
		BitmapID.push_back(-1);
		BitmapName.push_back(filename);
		BitmapColorKey.push_back(ColorKey);
		BitmapRect.push_back(CRect(0,0,0,0));
		lpDDS.push_back(NULL);
		LoadBitmap(i, filename);
		SetColorKey(i, ColorKey);
		return i;
}

void CDDraw::ReleaseBackCDC()
{
		cdc.Detach();
		ddrval = lpDDSBack->ReleaseDC(hdc);
		CheckDDFail("Release back HDC failed");
}

void CDDraw::ReleaseSurface()
```

```
{
    if (lpDD)
    {
            for (unsigned i = 0; i < lpDDS.size(); i++)
                  if (lpDDS[i]) {
                        lpDDS[i]->Release();
                        lpDDS[i] = NULL;
                  }
        if (lpDDSBack)
        {
            lpDDSBack->Release();
            lpDDSBack = NULL;
        }
        if (lpDDSPrimary)
        {
            lpDDSPrimary->Release();
            lpDDSPrimary = NULL;
        }
      }
}

void CDDraw::RestoreSurface()
{
      //
      // Since all surfaces are restored and reloaded, this operation can be very slow.
      //
      SetCursor(AfxGetApp()->LoadStandardCursor(IDC_WAIT));
    if (lpDD != NULL)
    {
            CreateSurface();
            while (lpDDSBack->IsLost() || lpDDSPrimary->IsLost()) {
                  Sleep(100);
                  CreateSurface();
            }
    }
}

void CDDraw::SetColorKey(unsigned SurfaceID, COLORREF color)
{
      if (color != CLR_INVALID) {
            DDCOLORKEY ddck;
            ddck.dwColorSpaceLowValue   = MatchColorKey(lpDDS[SurfaceID], color);
            ddck.dwColorSpaceHighValue = ddck.dwColorSpaceLowValue;
            ddrval = lpDDS[SurfaceID]->SetColorKey(DDCKEY_SRCBLT, &ddck);
            CheckDDFail("Can not Set Color Key");
      }
}

bool CDDraw::SetFullScreen(bool isFullScreen)
{
      fullscreen = isFullScreen;
      return CreateSurface();
}

void CDDraw::CheckDDFail(char *s)
{
      if (ddrval != DD_OK) {
            TRACE("Error Code: %d (%s)\n", ddrval, s);
            //
            // For some unknown reason, in Win98, ddrval can be changed
            //      after AfxMessageBox() is called, resulting an unknown
            //      error message.
            //
            AfxMessageBox(s);
            static int ErrorCode[] = {

      DDERR_ALREADYINITIALIZED             ,DDERR_BLTFASTCANTCLIP                   ,DDERR_CANNOTAT
TACHSURFACE              ,DDERR_CANNOTDETACHSURFACE             ,
```

DDERR_CANTCREATEDC ,DDERR_CANTDUPLICATE ,DDERR_CANTLOC KSURFACE ,DDERR_CANTPAGELOCK ,

DDERR_CANTPAGEUNLOCK ,DDERR_CLIPPERISUSINGHWND ,DDERR_COLORK EYNOTSET ,DDERR_CURRENTLYNOTAVAIL ,

DDERR_DCALREADYCREATED ,DDERR_DIRECTDRAWALREADYCREATED ,DDERR_EXC EPTION ,DDERR_EXCLUSIVEMODEALREADYSET ,

DDERR_GENERIC ,DDERR_HEIGHTALIGN ,DDERR_HWNDALREA DYSET ,DDERR_HWNDSUBCLASSED ,

DDERR_IMPLICITLYCREATED ,DDERR_INCOMPATIBLEPRIMARY ,DDERR_INVALIDC APS ,DDERR_INVALIDCLIPLIST ,

DDERR_INVALIDDIRECTDRAWGUID ,DDERR_INVALIDMODE ,DDERR_INVALID OBJECT ,DDERR_INVALIDPARAMS ,

DDERR_INVALIDPIXELFORMAT ,DDERR_INVALIDPOSITION ,DDERR_INVALIDRE CT ,DDERR_INVALIDSURFACETYPE ,

DDERR_LOCKEDSURFACES ,DDERR_NO3D ,DDERR_NOALPHAH W ,DDERR_NOBLTHW ,

DDERR_NOCLIPLIST ,DDERR_NOCLIPPERATTACHED ,DDERR_NOCOLORC ONVHW ,DDERR_NOCOLORKEY ,

DDERR_NOCOLORKEYHW ,DDERR_NOCOOPERATIVELEVELSET ,DDERR_NODC ,DDERR_NODDROPSHW ,

DDERR_NODIRECTDRAWHW ,DDERR_NODIRECTDRAWSUPPORT ,DDERR_NOEM ULATION ,DDERR_NOEXCLUSIVEMODE ,

DDERR_NOFLIPHW ,DDERR_NOGDI ,DDERR_NOHWND ,DDERR_NOMIPMAPHW ,

DDERR_NOMIRRORHW ,DDERR_NOOVERLAYDEST ,DDERR_NOOVER LAYHW ,DDERR_NOPALETTEATTACHED ,

DDERR_NOPALETTEHW ,DDERR_NORASTEROPHW ,DDERR_NOROTAT IONHW ,DDERR_NOSTRETCHHW ,

DDERR_NOT4BITCOLOR ,DDERR_NOT4BITCOLORINDEX ,DDERR_NOT8BITC OLOR ,DDERR_NOTAOVERLAYSURFACE ,

DDERR_NOTEXTUREHW ,DDERR_NOTFLIPPABLE ,DDERR_NOTFOUN D ,DDERR_NOTINITIALIZED ,

DDERR_NOTLOCKED ,DDERR_NOTPAGELOCKED ,DDERR_NOTPALE TTIZED ,DDERR_NOVSYNCHW ,

DDERR_NOZBUFFERHW ,DDERR_NOZOVERLAYHW ,DDERR_OUTOFC APS ,DDERR_OUTOFMEMORY ,

DDERR_OUTOFVIDEOMEMORY ,DDERR_OVERLAYCANTCLIP ,DDERR_OVERL AYCOLORKEYONLYONEACTIVE,DDERR_OVERLAYNOTVISIBLE ,

DDERR_PALETTEBUSY ,DDERR_PRIMARYSURFACEALREADYEXISTS ,DDERR_REGION TOOSMALL ,DDERR_SURFACEALREADYATTACHED ,

DDERR_SURFACEALREADYDEPENDENT ,DDERR_SURFACEBUSY ,DDERR_SURFA CEISOBSCURED ,DDERR_SURFACELOST ,

DDERR_SURFACENOTATTACHED ,DDERR_TOOBIGHEIGHT ,DDERR_TOOBIGSI ZE ,DDERR_TOOBIGWIDTH ,

DDERR_UNSUPPORTED ,DDERR_UNSUPPORTEDFORMAT ,DDERR_UNSUPP ORTEDMASK ,DDERR_UNSUPPORTEDMODE ,

```
	DDERR_VERTICALBLANKINPROGRESS	,DDERR_WASSTILLDRAWING	,DDERR_WRON
GMODE	,DDERR_XALIGN
		};
		static char *ErrorMsg[] = {
			"DDERR_ALREADYINITIALIZED	","DDERR_BLTFASTCANTCLIP
","DDERR_CANNOTATTACHSURFACE	","DDERR_CANNOTDETACHSURFACE	",
			"DDERR_CANTCREATEDC	","DDERR_CANTDUPLICATE
","DDERR_CANTLOCKSURFACE	","DDERR_CANTPAGELOCK	",
			"DDERR_CANTPAGEUNLOCK	","DDERR_CLIPPERISUSINGHWND
","DDERR_COLORKEYNOTSET	","DDERR_CURRENTLYNOTAVAIL	",
			"DDERR_DCALREADYCREATED	","DDERR_DIRECTDRAWALREADYCREATED
","DDERR_EXCEPTION	","DDERR_EXCLUSIVEMODEALREADYSET	",
			"DDERR_GENERIC	","DDERR_HEIGHTALIGN
","DDERR_HWNDALREADYSET	","DDERR_HWNDSUBCLASSED	",
			"DDERR_IMPLICITLYCREATED	","DDERR_INCOMPATIBLEPRIMARY
","DDERR_INVALIDCAPS	","DDERR_INVALIDCLIPLIST	",
			"DDERR_INVALIDDIRECTDRAWGUID	","DDERR_INVALIDMODE
","DDERR_INVALIDOBJECT	","DDERR_INVALIDPARAMS	",
			"DDERR_INVALIDPIXELFORMAT	","DDERR_INVALIDPOSITION
","DDERR_INVALIDRECT	","DDERR_INVALIDSURFACETYPE	",
			"DDERR_LOCKEDSURFACES	","DDERR_NO3D
","DDERR_NOALPHAHW	","DDERR_NOBLTHW	",
			"DDERR_NOCLIPLIST	","DDERR_NOCLIPPERATTACHED
","DDERR_NOCOLORCONVHW	","DDERR_NOCOLORKEY	",
			"DDERR_NOCOLORKEYHW	","DDERR_NOCOOPERATIVELEVELSET
","DDERR_NODC	","DDERR_NODDROPSHW	",
			"DDERR_NODIRECTDRAWHW	","DDERR_NODIRECTDRAWSUPPORT
","DDERR_NOEMULATION	","DDERR_NOEXCLUSIVEMODE	",
			"DDERR_NOFLIPHW	","DDERR_NOGDI
","DDERR_NOHWND	","DDERR_NOMIPMAPHW	",
			"DDERR_NOMIRRORHW	","DDERR_NOOVERLAYDEST
","DDERR_NOOVERLAYHW	","DDERR_NOPALETTEATTACHED	",
			"DDERR_NOPALETTEHW	","DDERR_NORASTEROPHW
","DDERR_NOROTATIONHW	","DDERR_NOSTRETCHHW	",
			"DDERR_NOT4BITCOLOR	","DDERR_NOT4BITCOLORINDEX
","DDERR_NOT8BITCOLOR	","DDERR_NOTAOVERLAYSURFACE	",
			"DDERR_NOTEXTUREHW	","DDERR_NOTFLIPPABLE
","DDERR_NOTFOUND	","DDERR_NOTINITIALIZED	",
			"DDERR_NOTLOCKED	","DDERR_NOTPAGELOCKED
","DDERR_NOTPALETTIZED	","DDERR_NOVSYNCHW	",
			"DDERR_NOZBUFFERHW	","DDERR_NOZOVERLAYHW
","DDERR_OUTOFCAPS	","DDERR_OUTOFMEMORY	",
			"DDERR_OUTOFVIDEOMEMORY	","DDERR_OVERLAYCANTCLIP
","DDERR_OVERLAYCOLORKEYONLYONEACTIVE","DDERR_OVERLAYNOTVISIBLE	",
			"DDERR_PALETTEBUSY	","DDERR_PRIMARYSURFACEALREADYEXISTS
","DDERR_REGIONTOOSMALL	","DDERR_SURFACEALREADYATTACHED	",
			"DDERR_SURFACEALREADYDEPENDENT	","DDERR_SURFACEBUSY
","DDERR_SURFACEISOBSCURED	","DDERR_SURFACELOST	",
			"DDERR_SURFACENOTATTACHED	","DDERR_TOOBIGHEIGHT
","DDERR_TOOBIGSIZE	","DDERR_TOOBIGWIDTH	",
			"DDERR_UNSUPPORTED	","DDERR_UNSUPPORTEDFORMAT
","DDERR_UNSUPPORTEDMASK	","DDERR_UNSUPPORTEDMODE	",
			"DDERR_VERTICALBLANKINPROGRESS	","DDERR_WASSTILLDRAWING
","DDERR_WRONGMODE	","DDERR_XALIGN	"
		};
		for (int i = 0; i < sizeof(ErrorCode)/sizeof(int); i++)
			if (ddrval == ErrorCode[i])
				GAME_ASSERT(0, ErrorMsg[i]);
		GAME_ASSERT(0, "Direct Draw Failed due to unknown error code !!!");
	}
}

}
```

4.gamelib.h

```
#define SIZE_X				1680		// 設定遊戲畫面的解析度為 640x480
#define SIZE_Y				1050		// 註：若不使用標準的解析度，則不能切換到全螢幕
```

```
#define OPEN_AS_FULLSCREEN     false            // 是否以全螢幕方式開啟遊戲
#define SHOW_LOAD_PROGRESS      true                   // 是否顯示 loading(OnInit)的進度
#define DEFAULT_BG_COLOR        RGB(0,0,0)  // 遊戲畫面預設的背景顏色(黑色)
#define GAME_CYCLE_TIME                 33                   // 每 33ms 跑一次 Move 及 Show(每秒 30 次)
#define SHOW_GAME_CYCLE_TIME false                   // 是否在 debug mode 顯示 cycle time
#define ENABLE_GAME_PAUSE       true             // 是否允許以  Ctrl-Q  暫停遊戲
#define ENABLE_AUDIO                true             // 啟動音效介面

enum GAME_STATES {
        GAME_STATE_INIT,
        GAME_STATE_DUKE,
        GAME_STATE_CO,
        GAME_STATE_RUN,
        GAME_STATE_WIN,
        GAME_STATE_OVER
};

#include <list>
#include <vector>
#include <map>
using namespace std;

#define GAME_ASSERT(boolexp,str)                                                    \
                if (!(boolexp)) {                                                       \
                        int id;
        \
                        char s[300]="";
        \
                        sprintf(s,"Game fatal error:\n\n%s\n\nFile: %s\n\nLine: %d"          \
                                "\n\n(Press Retry to debug the application, "          \
                                "if it is executed in debug mode.)"                    \
                                "\n(Press Cancel otherwise.)",                         \
                                  str , __FILE__,__LINE__);                            \
                        id = AfxMessageBox(s, MB_RETRYCANCEL);                        \
                        if (id == IDCANCEL)                                             \
                                exit(1);                                                \
                        AfxDebugBreak();                                                \
                }

namespace game_framework {


class CSpecialEffect {
public:
        static void    SetCurrentTime();                                  // 儲存目前的時間至 ctime
        static DWORD GetEllipseTime();                                    // 讀取目前的時間  - ctime
        static int      GetCurrentTimeCount();                        // 讀取儲存 ctime 的次數
        static void    Delay(DWORD ms);                                   // 延遲  x ms
        static void    DelayFromSetCurrentTime(DWORD ms);      // 自  ctime 起算，延遲  x ms
private:
        static DWORD ctime;
        static int          ctimeCount;
};


class CDDraw {
        friend class CMovingBitmap;
public:
        ~CDDraw();
        static void    BltBackColor(DWORD);               // 將 Back plain 全部著上指定的顏色
        static void    BltBackToPrimary();             // 將 Back plain 貼至 Primary plain
        static CDC*    GetBackCDC();                               // 取得 Back Plain 的 DC (device context)
        static void    GetClientRect(CRect &r);        // 取得設定的解析度
        static void    Init(int, int);                 // Initialize direct draw
        static void    ReleaseBackCDC();                  // 放掉 Back Plain 的 DC (device context)
        static bool    SetFullScreen(bool);            // 設定為全螢幕模式/視窗模式
        static bool    IsFullScreen();                 // 回答是否為全螢幕模式/視窗模式
private:
        CDDraw();                                                 // private constructor
        static void    BltBitmapToBack(unsigned SurfaceID, int x, int y);
        static void    BltBitmapToBack(unsigned SurfaceID, int x, int y, double factor);
        static void    BltBitmapToBitmap(unsigned SourceID, unsigned TargetID, int x, int y);
        static void          CheckDDFail(char *s);
        static bool    CreateSurface();
        static bool    CreateSurfaceFullScreen();
        static bool    CreateSurfaceWindowed();
        static void    LoadBitmap(int i, int IDB_BITMAP);
        static void    LoadBitmap(int i, char *filename);
        static DWORD MatchColorKey(LPDIRECTDRAWSURFACE lpDDSurface, COLORREF color);
        static int      RegisterBitmap(int IDB_BITMAP, COLORREF ColorKey);
        static int      RegisterBitmap(char *filename, COLORREF ColorKey);
        static void    ReleaseSurface();
        static void    RestoreSurface();
        static void    SetColorKey(unsigned SurfaceID, COLORREF color);
      static HDC                                 hdc;
        static CDC                               cdc;
        static CView                     *pCView;
      static LPDIRECTDRAW2               lpDD;
        static LPDIRECTDRAWCLIPPER   lpClipperPrimary;
        static LPDIRECTDRAWCLIPPER   lpClipperBack;
        static LPDIRECTDRAWSURFACE  lpDDSPrimary;
        static LPDIRECTDRAWSURFACE  lpDDSBack;
        static vector<LPDIRECTDRAWSURFACE>        lpDDS;
      static HRESULT                     ddrval;
        static vector<int>               BitmapID;
        static vector<string>        BitmapName;
        static vector<CRect>         BitmapRect;
        static vector<COLORREF>          BitmapColorKey;
        static bool                              fullscreen;
        static CDDraw                    ddraw;
        static int                               size_x, size_y;
};


class CMovingBitmap {
public:
        CMovingBitmap();
        int     Height();                                   // 取得圖形的高度
        int     Left();                                 // 取得圖形的左上角的 x 座標
        void    LoadBitmap(int,COLORREF=CLR_INVALID);               // 載入圖，指定圖的編號(resource)及透明色
        void    LoadBitmap(char *,COLORREF=CLR_INVALID); // 載入圖，指定圖的檔名及透明色
        void    SetTopLeft(int,int);                // 將圖的左上角座標移至 (x,y)
        void    ShowBitmap();                           // 將圖貼到螢幕
        void    ShowBitmap(double factor);  // 將圖貼到螢幕 factor < 1 時縮小，>1 時放大。注意：需要 VGA 卡硬體的支
援，否則會很慢
        void    ShowBitmap(CMovingBitmap &);    // 將圖貼到到另一張圖上 (僅供特殊用途)
        int     Top();                                  // 取得圖形的左上角的 y 座標
        int     Width();                                // 取得圖形的寬度
protected:
        CRect       location;                   // location of the bitmap
        bool        isBitmapLoaded;   // whether a bitmap has been loaded
        unsigned SurfaceID;                     // the surface id of this bitmap
};


class CAnimation {
public:
        CAnimation(int=10);                     // Constructor (預設動畫播放頻率每 1/3 秒換一張圖)
        void    AddBitmap(int,COLORREF=CLR_INVALID);
                                                    // 增加一張圖形至動畫(圖的編號及透明色)
        void    AddBitmap(char *,COLORREF=CLR_INVALID);
                                                    // 增加一張圖形至動畫(圖的編號及透明色)
        int     GetCurrentBitmapNumber();  // 取得正在撥放的 bitmap 是第幾個 bitmap
        int     Height();                               // 取得動畫的高度
        bool    IsFinalBitmap();                    // 回傳正在撥放的 bitmap 是否為最後一個 bitmap
        int     Left();                             // 取得動畫的左上角的 x 座標
        void    OnMove();                               // 依頻率更換 bitmap
        void    OnShow();                               // 將動畫貼到螢幕
```

```
	void	Reset();				// 重設播放順序回到第一張圖形
	void	SetDelayCount(int);		// 設定動畫播放速度的常數(越大越慢)
	void	SetTopLeft(int,int);		// 將動畫的左上角座標移至 (x,y)
	int	Top();				// 取得動畫的左上角的 y 座標
	int	Width();				// 取得動畫的寬度
private:
	list<CMovingBitmap>			bmp;			// list of CMovingBitmap
	list<CMovingBitmap>::iterator	bmp_iter;		// list iterator
	int						bmp_counter;	// 儲存 bmp_iter 為第 n 個 bmp
	int						delay_counter;// 延緩動畫播放速度的計數器
	int						delay_count;	// 動畫播放速度的常數
	int						x, y;			// 動畫的座標
};


class CInteger {
public:
	CInteger(int=5);			// default 5 digits
	void Add(int n);			// 增加整數值
	int	GetInteger();			// 回傳整數值
	void LoadBitmap();			// 載入 0..9 及負號之圖形
	void SetInteger(int);		// 設定整數值
	void SetTopLeft(int,int);	// 將動畫的左上角座標移至 (x,y)
	void ShowBitmap();			// 將動畫貼到螢幕
private:
	const int NUMDIGITS;			// 共顯示 NUMDIGITS 個位數
	static CMovingBitmap digit[11]; // 儲存 0..9 及負號之圖形(bitmap)
	int x, y;					// 顯示的座標
	int n;					// 整數值
	bool isBmpLoaded;			// 是否已經載入圖形
};


class CGame;
class CGameStateInit;
class CGameStateRun;
class CGameStateOver;
class CGameStateDuke;
class CGameStateWin;
class CGameStateCo;

class CGameState {
public:
	CGameState(CGame *g);
	void OnDraw();			// Template Method
	void OnCycle();			// Template Method
	//
	// virtual functions, 由繼承者提供 implementation
	//
	virtual ~CGameState() {}						// virtual destructor
	virtual void OnBeginState() {}					// 設定每次進入這個狀態時所需的
初值
	virtual void OnInit() {}						// 狀態的初值及圖形設定
	virtual void OnKeyDown(UINT, UINT, UINT) {}			// 處理鍵盤 Down 的動作
	virtual void OnKeyUp(UINT, UINT, UINT) {}			// 處理鍵盤 Up 的動作
	virtual void OnLButtonDown(UINT nFlags, CPoint point) {}// 處理滑鼠的動作
	virtual void OnLButtonUp(UINT nFlags, CPoint point) {}  // 處理滑鼠的動作
	virtual void OnMouseMove(UINT nFlags, CPoint point) {}   // 處理滑鼠的動作
	virtual void OnRButtonDown(UINT nFlags, CPoint point) {}// 處理滑鼠的動作
	virtual void OnRButtonUp(UINT nFlags, CPoint point) {}  // 處理滑鼠的動作
protected:
	void GotoGameState(int state);					// 跳躍至指定的 state
	void ShowInitProgress(int percent);				// 顯示初始化的進度
	//
	// virtual functions, 由繼承者提供 implementation
	//
	virtual void OnMove() {}						// 移動這個狀態的遊戲元素
	virtual void OnShow() = 0;					// 顯示這個狀態的遊戲畫面
	CGame *game;
```

```
};

class CGame {
public:
        CGame();                                                            // Constructor
        ~CGame();                                                           // Destructor
        bool IsRunning();                                          // 讀取遊戲是否正在進行中
        void OnDraw();                                                      // 對應 CGameView 的 OnDraw()
        void OnFilePause();                                        // 遊戲暫停
        void OnInit();                                             // 遊戲繪圖及音效的初始化
        void OnInitStates();                              // 遊戲各狀態的初值及圖形設定
        bool OnIdle();                                             // 遊戲的主迴圈
        void OnKeyDown(UINT, UINT, UINT);                 // 處理鍵盤 Down 的動作
        void OnKeyUp(UINT, UINT, UINT);                            // 處理鍵盤 Up 的動作
        void OnKillFocus();                                        // 遊戲被迫暫停
        void OnLButtonDown(UINT nFlags, CPoint point);// 處理滑鼠的動作
        void OnLButtonUp(UINT nFlags, CPoint point);    // 處理滑鼠的動作
        void OnMouseMove(UINT nFlags, CPoint point);      // 處理滑鼠的動作
        void OnRButtonDown(UINT nFlags, CPoint point);// 處理滑鼠的動作
        void OnRButtonUp(UINT nFlags, CPoint point);    // 處理滑鼠的動作
        void OnResume();                                           // 處理自「待命」還原的動作
        void OnSetFocus();                                         // 處理 Focus
        void OnSuspend();                                          // 處理「待命」的動作
        void SetGameState(int);
        static CGame *Instance();
private:
        bool            running;                // 遊戲是否正在進行中(未被 Pause)
        bool          suspended;                // 遊戲是否被 suspended
        const int       NUM_GAME_STATES;        // 遊戲的狀態數(3 個狀態)
        CGameState      *gameState;             // pointer 指向目前的遊戲狀態
        CGameState      *gameStateTable[6];  // 遊戲狀態物件的 pointer
        static CGame  instance;              // 遊戲唯一的 instance
};
}
//state pattern
```

全文完