目錄

一、　簡介

1. 動機：

當時正好我們兩個人分別玩、看「還願」遊戲實況，對於原作結局感到很惆悵，於是遊戲背景就訂在若美心在被浸泡蛇酒之後並沒有死亡，尋找爸爸並與之對戰。而遊戲名稱「淨化 Decontamination」是以美心為了要將爸爸的癲狂淨化而取，英文部分則是呼應原遊戲創作公司赤燭公司所出的兩個遊戲英文名皆由 D 開頭命名。遊戲內容則是參閱「拉比哩比」這款遊戲，裡面王關就是彈幕的遊戲。

2. 分工：

陳美蓁	陳柏瑞
CBlood CDialog CTransition 音效 對話框 血條 初始頁面 地圖 5	CBossAI CDamageContainer CDamageFish CDamageLighter CDamageObject CDamagePen CDamagePill CDamageWinnie CDaughter CFather CGameData CGameMap CGameMap01~05 CGameState01~05 Definitions 地圖 1~4 攻擊物件 音效

二、 遊戲介紹

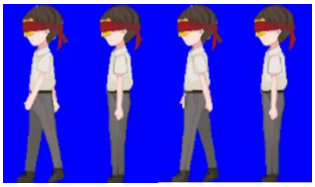
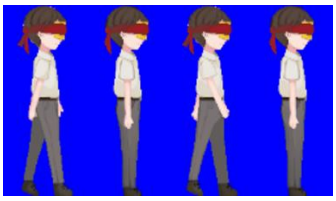
1. 遊戲說明：

本遊戲是用鍵盤操控。遊戲中地圖 1~4 主要為推進劇情，而最後一張地圖是王關，需要躲避 Boss 的攻擊招式，並將 Boss 殺死。以下為遊戲中使用的按鍵與對應事件。

移動事件			
按鍵	對應事件	按鍵	對應事件
左鍵	向左走	右鍵	向右走
空白鍵	跳躍	空白鍵按兩下	二段跳躍
Z 鍵	攻擊	空白鍵+Z 鍵	飛行
X 鍵	滑行	暫停遊戲	Ctrl 鍵+Q 鍵
結束遊戲	esc 鍵		
密技			
按鍵	對應事件	按鍵	對應事件
F2	切換至下一章節	F3	美心滿血
F4	美心死亡	F5	爸爸半血
F6	爸爸死亡		

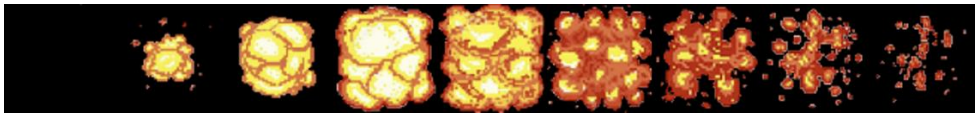
2. 遊戲圖形：

美心站立	美心滑行（左、右）
	
美心向左走	
	
美心向右走	
	
美心飛行（左、右）	美心跳躍（左、右）
	
美心攻擊（左、右）	
	

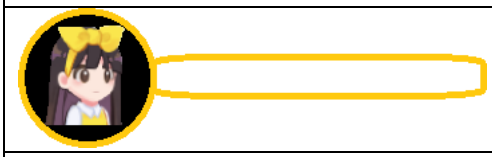
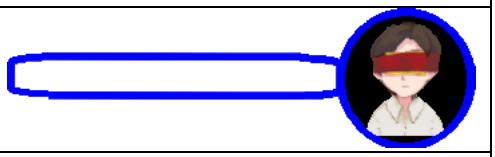
爸爸向左走	爸爸向右走
	

爸爸站立	鋼筆	警示標誌
		

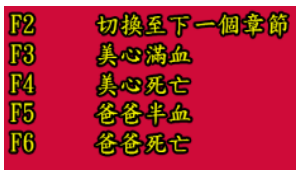
藥丸	藥丸準心	符咒	打火機
			

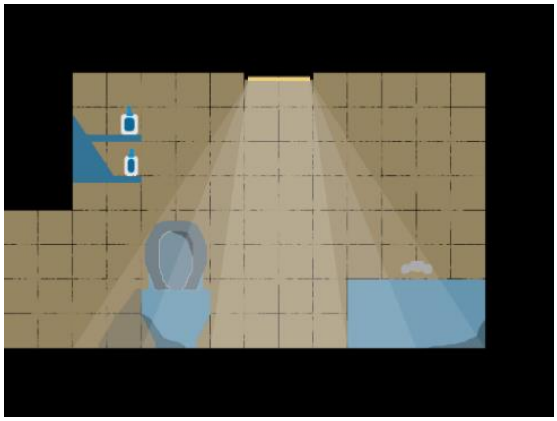
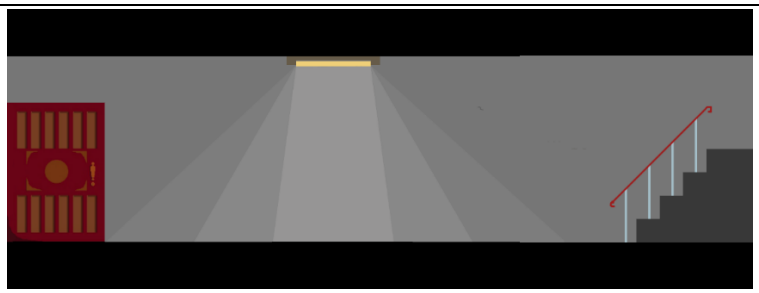
打火機爆炸特效


紅龍魚躍


美心血條	爸爸血條
	

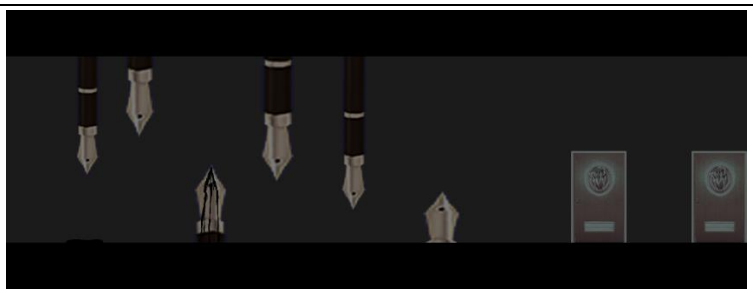
血條	出場上下橫幅
	

密技	此門不通	遊戲 icon
		

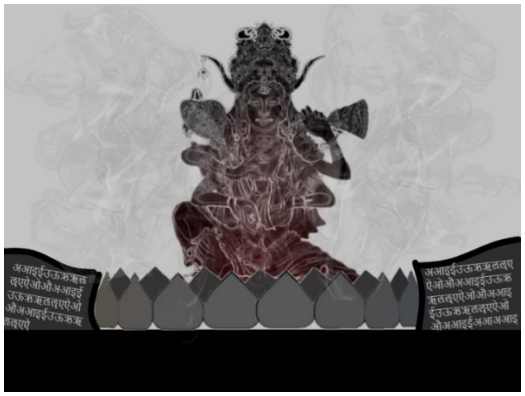
初始選單	功能介紹
	
暫停遊戲	遊戲結束
	
地圖一	
	
地圖二	
	

地圖三

地圖四



地圖五



對話框

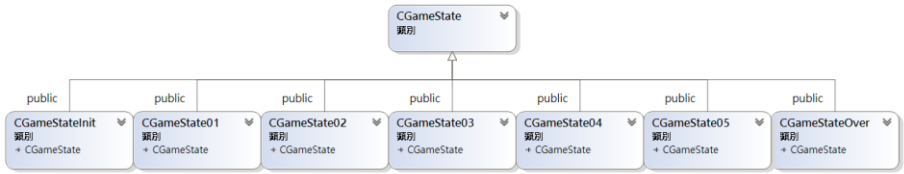
3. 遊戲音效：

事件	對應音效
選單 BGM	devotion.mp3
Map01 BGM	drops.mp3
Map01 出場	open_toilet_door.mp3
Map02 進場	close_door.wav
Map03 BGM	strong_wind_sound_effect.mp3
Map03 出場	iron_door.mp3
Map04 BGM	whisper_mixdown.mp3
Map04 出場	room_door.mp3
王關 BGM	boss_bgm.mp3
美心腳步聲	footstep.wav
美心滑行聲	slide.wav
美心攻擊聲	attack.mp3
美心被攻擊聲	hurt.mp3
爸爸被攻擊聲	hurt_father.mp3
打火機爆炸聲	explosion.mp3
紅龍魚躍聲	sea_wave.mp3
鋼筆落地聲	ground_hit.mp3

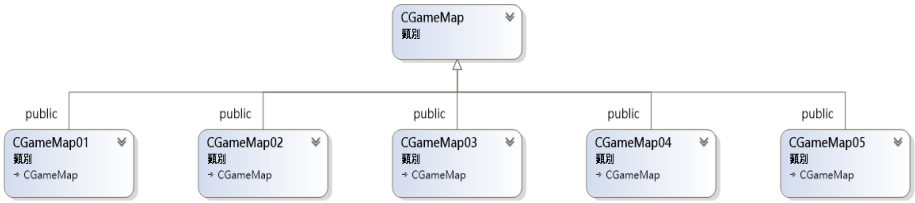
三、 程式設計

1. 程式架構：

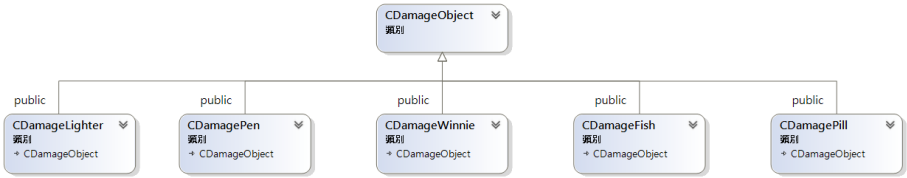
CGameState



CGameMap



CDamageObject



Others


CAboutDlg
類別
CDialog
CAnimation
類別
CAudio
類別
CBlood
類別
CBossAI
類別
CDamageConta...
類別
CDaughter
類別
CDDraw
類別
CDialog
類別
CFather
類別
CGame
類別
CGameApp
類別
CWinApp
CGameData
類別
CGameDoc
類別
CDocument
CGameView
類別
CView
CInteger
類別
CMainFrame
類別
CFrameWnd
CMovingBitmap
類別
CSpecialEffect
類別
CTransition
類別
AUDIO_ID
列舉
GAME_STATES
列舉

2. 程式類別：

類別名稱	.h 檔行數	.cpp 檔行數	說明
Definitions	16		遊戲元件的定義
CBlood	42	77	操控美心與爸爸的血條
CBossAI	58	464	操控爸爸的 AI
CDamageContainer	37	125	存取所有爸爸攻擊招式及設定攻擊的傷害量
CDamageFish	30	59	操控爸爸攻擊招式中的紅龍
CDamageLighter	30	69	操控爸爸攻擊招式中的打火機
CDamageObject	51	28	爸爸攻擊招式的框架
CDamagePen	30	50	操控爸爸攻擊招式中的鋼筆
CDamagePill	34	57	操控爸爸攻擊招式中的藥丸
CDamageWinnie	29	42	操控爸爸攻擊招式中的符咒
CDaughter	114	305	操控美心移動動作等
CDialog	53	124	劇情對話框
CFather	55	162	操控爸爸移動動作等
CGameData	43	129	存取地圖、美心、爸爸位置等資料

類別名稱	.h 檔行數	.cpp 檔行數	說明
CGameMap	77	43	地圖的框架
CGameMap01	17	37	地圖一
CGameMap02	17	36	地圖二
CGameMap03	17	36	地圖三
CGameMap04	17	36	地圖四
CGameMap05	17	36	地圖五
CGameState01	38	80	遊戲章節一
CGameState02	39	81	遊戲章節二
CGameState03	39	82	遊戲章節三
CGameState04	43	115	遊戲章節四
CGameState05	48	135	遊戲章節五
CTransition	24	67	遊戲轉場切換
mygame	96	168	State_init,state_over
總行數	**489**	**952**	**1441**

3. 程式技術：

最多技術都集中在 BossAI 的部分上，因為要使他能夠完全自動且執行所有攻擊與移動，所以我用了一個矩陣表示各個狀態間互相轉移的機率，AI 會記錄前一個狀態，而透過前一個狀態可以得到轉移至其他狀態的機率，透過隨機的方式決定他下一步要做什麼樣的動作，完成自主運作。

四、 結語

1. 問題及解決方法：

問題	解決方法
轉場動畫不知道如何順暢運行	丟在 OnMove()裡，y 就會穩定增加
不知道 Dialog 如何換圖	參考 Aniation
動畫第一張圖會卡頓造成動畫不順	在第一張圖加上一張空白圖片吃掉延遲

2. 時間表：

週次	陳美蓁(小時)	陳柏瑞(小時)	說明
1	2	2	練習
2	2	2	練習
3	3	2	練習、開會
4	4	4	地圖
5	6	6	主角移動
6	8	10	螢幕跟主角移動、換地圖
7	7	3	攻擊與轉場動畫
8	2	2	攻擊與轉場動畫完成
9	1	4	地圖 4
10	4	4	攻擊物件、音效
11	5	3	攻擊物件、血條
12	4	4	攻擊判定
13	6	3	攻擊物件、音效與血條完成
14	6	4	攻擊物件，code clean
15	5	10	剩餘攻擊物件，BOSS 實作，劇情對話
16	10	20	BOSS 實作、劇情完成
17	20	40	補音效、劇情、做期末報告
合計	**95**	**123**	

3. 貢獻比例：

陳美蓁：45%、陳柏瑞：55%

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	自訂遊戲 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：

陳美蓁：

可以設中斷點一步一步去找可以通過編譯的錯誤。

陳柏瑞：

如何用最基本的框架，透過自己實現所有物理效果與碰撞，來完成一個遊戲。

6. 心得、感想：

陳美蓁：

在這學期的課程中我學到了如何將一個遊戲從只有框架到一個遊戲完整的產出。其中過程除了程式規劃、程式撰寫外，最耗時間的莫過於蒐集材料，除了需要自己手工做出一張張圖片，還有要找到符合遊戲內的音效。還記得當時在找腳步聲的時候，聽了十幾種聲音才最後定案。很感謝老師開這一堂課,讓我們可以實際運用物件導向程式設計課中的概念,去完成一個自己的作品。在每週回報進度的時候，都會指導我們所遇到的問題該如何解決比較好。感謝同學跟我一起完成了這個遊戲！

陳柏瑞：

滿訝異最後能做出這個遊戲的，本來在設計的時候目標就是想做一個高難度的彈幕遊戲，因為這類遊戲粒子特效跟物件都特別多，也要有很多種不同的攻擊方式，本來是以為做不出來的，最後產出的這個遊戲雖然跟其他的彈幕遊戲還有很大的差距，但我已經十分滿意。

7. 對於本課程的建議：

希望下次期末報告的遊戲圖形可以擴充到 5 頁。

附錄

```
==============================
CBlood.cpp
==============================
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CBlood.h"
namespace game_framework {
	CBlood::CBlood() : daughterHP(102), fatherHP(450) {}
	CBlood::~CBlood(){}
	void CBlood::LoadBitmap() {
		daughter_hp.LoadBitmap(IDB_DAUGHTER_BLOOD, RGB(255, 255, 255));
		daughter_blood.LoadBitmap(IDB_BLOOD, RGB(255, 255, 255));
		father_hp.LoadBitmap(IDB_FATHER_BLOOD, RGB(255, 255, 255));
		father_blood.LoadBitmap(IDB_BLOOD, RGB(255, 255, 255));
	}
	void CBlood::setDaughterHP(int hp) {
		daughterHP = hp;
	}
	void CBlood::setDaughterDamage(int damage,int invincible_time) {
		if (daughter_invincible < 0) {
			daughterHP -= damage;
			daughter_invincible = invincible_time;
			CAudio::Instance()->Play(DAUGHTER_HURT, false);
		}
	}
	bool CBlood::daughterIsDied() {
		if (daughterHP <= -141) {
			return true;
		}
		else {
			return false;
		}
	}
	void CBlood::setFatherHP(int hp) {
		fatherHP = hp;
	}
	void CBlood::setFatherDamage(int damage,int invincible_time) {
		if (father_invincible < 0) {
			fatherHP += damage;
			father_invincible = invincible_time;
			CAudio::Instance()->Play(FATHER_HURT, false);
		}
	}
	bool CBlood::fatherIsDied() {
		if (fatherHP >= 698) {
			return true;
		}
		else {
			return false;
		}
	}
	void CBlood::OnShow(bool showFatherHP) {
		daughter_hp.SetTopLeft(0, 10);
		daughter_blood.SetTopLeft(daughterHP, 48);
		daughter_blood.ShowBitmap();
		daughter_hp.ShowBitmap();
		if (showFatherHP) {
			father_hp.SetTopLeft(449, 10);
			father_blood.SetTopLeft(fatherHP, 48);
			father_blood.ShowBitmap();
			father_hp.ShowBitmap();
```

```
		}
		daughter_invincible--;
		father_invincible--;
	}
}
```

==========

CBlood.h

==========

```
#pragma once
#include "Definitions.h"
#ifndef CBLOOD_H
#define CBLOOD_H
namespace game_framework {
	class CBlood {
	public:
		CBlood();
		~CBlood();
		void LoadBitmap();
		void setDaughterHP(int);		//設定美心血量
		void setDaughterDamage(int, int = 30);		//設定美心被傷害量
		int getDaughterHP() const;		//回傳美心血量
		bool daughterIsDied();		//判斷美心是否死亡
		void setFatherHP(int);		//設定爸爸血量
		void setFatherDamage(int, int = 6);		//設定爸爸被傷害量
		int getFatherHP() const;		//回傳爸爸血量
		bool fatherIsDied();		//判斷爸爸是否死亡
		void OnShow(bool);
	private:
		CMovingBitmap daughter_hp;
		CMovingBitmap daughter_blood;
		CMovingBitmap father_hp;
		CMovingBitmap father_blood;
		int daughterHP;
		int fatherHP;
		int daughter_invincible;
		int father_invincible;
	};
	inline int CBlood::getDaughterHP() const{
		return daughterHP;
	}
	inline int CBlood::getFatherHP() const{
		return fatherHP;
	}
}
#endif
```

==========

CBossAI.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include <stdlib.h>
#include <time.h>
#include "audio.h"
#include "CBossAI.h"
#include "CFather.h"
#include "CGameData.h"
#include "CDamageContainer.h"
namespace game_framework {
	CBossAI::CBossAI() {
		int decision_probiality_1_init[AI_TOTAL_STATE_COUNT][AI_TOTAL_STATE_COUNT] = {
			{ 0,5,30,0,10,30,0,0,0,30,0 },
			{ 5,0,30,0,30,10,0,0,0,30,0 },
			{ 10,10,0,0,0,0,0,0,0,0,0 },
			{ 10,10,0,0,0,0,0,0,0,0,0 },
			{ 10,0,0,0,0,0,0,0,0,0,0 },
			{ 0,10,0,0,0,0,0,0,0,0,0 },
```

```
		{ 10,10,0,0,0,0,0,0,0,0,0 },
		{ 10,0,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 }
	};
	int decision_probiality_2_init[AI_TOTAL_STATE_COUNT][AI_TOTAL_STATE_COUNT] = {
		{ 0,5,0,30,0,25,30,0,30,30,0 },
		{ 5,0,0,30,25,0,30,30,0,30,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 },
		{ 10,0,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 },
		{ 10,0,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 },
	    { 10,10,0,0,0,0,0,0,0,0,0 }
	};
	counter = 0;
	gap_counter = 0;
	current_state = AI_STATE_MOVE_LEFT;
	AI_Lock = false;
	decision_probiality_1 = new int*[AI_TOTAL_STATE_COUNT];
	for (int i = 0; i < AI_TOTAL_STATE_COUNT; i++) {
		decision_probiality_1[i] = new int[AI_TOTAL_STATE_COUNT];
		for (int j = 0; j < AI_TOTAL_STATE_COUNT; j++) {
			decision_probiality_1[i][j] = decision_probiality_1_init[i][j];
		}
	}
}
CBossAI::CBossAI(CFather* father, CGameData* game_data, CDamageContainer* container,CBlood* blood) {
	int decision_probiality_1_init[AI_TOTAL_STATE_COUNT][AI_TOTAL_STATE_COUNT] = {
		{ 0,5,30,0,10,30,0,0,0,30,0 },
		{ 5,0,30,0,30,10,0,0,0,30,0 },
		{ 30,30,0,0,15,15,0,0,0,0,0 },
		{ 30,30,0,0,15,15,0,0,0,0,0 },
		{ 10,0,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 },
		{ 10,0,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 },
		{ 10,10,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 }
	};
	int decision_probiality_2_init[AI_TOTAL_STATE_COUNT][AI_TOTAL_STATE_COUNT] = {
		{ 0,5,0,30,0,25,30,0,30,30,30 },
		{ 5,0,0,30,25,0,30,30,0,30,30},
		{ 30,30,0,0,15,15,0,15,15,0,0 },
		{ 30,30,0,0,15,15,0,15,15,0,0 },
		{ 10,0,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 },
		{ 30,30,0,0,15,15,0,15,15,0,0 },
		{ 10,0,0,0,0,0,0,0,0,0,0 },
		{ 0,10,0,0,0,0,0,0,0,0,0 },
		{ 30,30,0,0,15,15,0,15,15,0,0 },
		{ 0,30,0,0,10,0,0,10,0,0,0 }
	};
	this->father = father;
	this->game_data = game_data;
	this->container = container;
	this->blood = blood;
	counter = 0;
	gap_counter = 0;
	current_state = AI_STATE_MOVE_LEFT;
	AI_Lock = false;
	decision_probiality_1 = new int*[AI_TOTAL_STATE_COUNT];
	for (int i = 0; i < AI_TOTAL_STATE_COUNT; i++) {
		decision_probiality_1[i] = new int[AI_TOTAL_STATE_COUNT];
```

```
		for (int j = 0; j < AI_TOTAL_STATE_COUNT; j++) {
			decision_probiality_1[i][j] = decision_probiality_1_init[i][j];
		}
	}
	decision_probiality_2 = new int*[AI_TOTAL_STATE_COUNT];
	for (int i = 0; i < AI_TOTAL_STATE_COUNT; i++) {
		decision_probiality_2[i] = new int[AI_TOTAL_STATE_COUNT];
		for (int j = 0; j < AI_TOTAL_STATE_COUNT; j++) {
			decision_probiality_2[i][j] = decision_probiality_2_init[i][j];
		}
	}
}
CBossAI::~CBossAI() {
	for (int i = 0; i < AI_TOTAL_STATE_COUNT; i++) {
		delete[] decision_probiality_1[i];
		delete[] decision_probiality_2[i];
	}
	delete[] decision_probiality_1;
	delete[] decision_probiality_2;
}
void CBossAI::setAILock(bool l) {
	this->AI_Lock = l;
}
void CBossAI::OnLoop() {
	if (AI_Lock) {
		current_state = AI_STATE_IDLE;
		previous_state = AI_STATE_MOVE_LEFT;
	}
	switch (current_state) {
	case AI_STATE_IDLE:
		nextStepChoise();
		break;
	case AI_STATE_MOVE_LEFT:
		stateMoveLeft();
		break;
	case AI_STATE_MOVE_RIGHT:
		stateMoveRight();
		break;
	case AI_STATE_ATK_WINNIE_1:
		stateWinnieATK1();
		break;
	case AI_STATE_ATK_WINNIE_2:
		stateWinnieATK2();
		break;
	case AI_STATE_ATK_FISH:
		stateFishATK();
		break;
	case AI_STATE_ATK_PEN_R:
		if (blood->getFatherHP() > 550) {
			gap_counter = 30;
		}
		else {
			gap_counter = 40;
		}
		statePenATKR();
		break;
	case AI_STATE_ATK_PEN_L:
		if (blood->getFatherHP() > 550) {
			gap_counter = 30;
		}
		else {
			gap_counter = 40;
		}
		statePenATKL();
		break;
	case AI_STATE_ATK_PEN_R_2:
		statePenATKR2();
		break;
	case AI_STATE_ATK_PEN_L_2:
```

```
		statePenATKL2();
		break;
	case AI_STATE_ATK_PILL:
		if (blood->getFatherHP() > 550) {
			gap_counter = 5;
		}
		else {
			gap_counter = 10;
		}
		statePillATK();
		break;
	case AI_STATE_ATK_LIGHTER:
		stateLighterATK();
		break;
	}
}
void CBossAI::nextStepChoise() {
	bool low_health = blood->getFatherHP() > 550;
	srand((unsigned)time(NULL));
	int sum = 0;
	int rand_result = 0;
	int next_state = 0;
	for (int i = 0; i < AI_TOTAL_STATE_COUNT; i++) {
		if (low_health) {
			sum += decision_probiality_2[previous_state][i];
		}
		else {
			sum += decision_probiality_1[previous_state][i];
		}
	}
	rand_result = (rand() % sum)+1;
	for (int i = 0; i < AI_TOTAL_STATE_COUNT; i++) {
		if (low_health) {
			rand_result -= decision_probiality_2[previous_state][i];
		}
		else {
			rand_result -= decision_probiality_1[previous_state][i];
		}
		if (rand_result <= 0) {
			next_state = i;
			break;
		}
	}
	if (game_data->getFatherPos().first > 650) {
		next_state = AI_STATE_MOVE_LEFT;
	}
	else if (game_data->getFatherPos().first < 50) {
		next_state = AI_STATE_MOVE_RIGHT;
	}
	previous_state = current_state;
	current_state = next_state;
	counter = 0;
}
bool CBossAI::moveToTargetPos(int x, int y) {
	std::pair<int, int> pos = game_data->getFatherPos();
	std::pair<int, int> target_vector = std::make_pair<int, int>(x - pos.first, y - pos.second);
	if ((target_vector.first<5 && target_vector.first>-5) && (target_vector.second<5 && target_vector.second>-5)) {
		game_data->setFatherTopLeft(x, y);
	}
	else {
		game_data->setFatherMove(target_vector.first / 5, target_vector.second / 5);
	}
	pos = game_data->getFatherPos();
	return pos.first == x && pos.second == y;
}
void CBossAI::stateMoveRight() {
	father->OnKeyDown(KEY_RIGHT);
	counter++;
	if (counter > 45) {
```

```
		father->OnKeyUP(KEY_RIGHT);
		previous_state = AI_STATE_MOVE_RIGHT;
		current_state = AI_STATE_IDLE;
		counter = 0;
	}
}
void CBossAI::stateMoveLeft() {
	father->OnKeyDown(KEY_LEFT);
	counter++;
	if (counter > 45) {
		father->OnKeyUP(KEY_LEFT);
		previous_state = AI_STATE_MOVE_LEFT;
		current_state = AI_STATE_IDLE;
		counter = 0;
	}
}
void CBossAI::stateWinnieATK1() {
	father->setOnTakeCtrl(true);
	father->setStatus(FATHER_STATUS_FACE_1);
	if (moveToTargetPos(350, 10)) {
		father->setStatus(FATHER_STATUS_FACE_2);
		if (counter % 20 == 0) {
			container->createDamageObject(OBJET_WINNIE_1);
		}
		counter++;
		if (counter >= 60) {
			father->setOnTakeCtrl(false);
			previous_state = AI_STATE_ATK_WINNIE_1;
			current_state = AI_STATE_IDLE;
			counter = 0;
		}
	}
}
void CBossAI::stateWinnieATK2() {
	father->setOnTakeCtrl(true);
	father->setStatus(FATHER_STATUS_FACE_1);
	if (moveToTargetPos(350, 10)) {
		father->setStatus(FATHER_STATUS_FACE_2);
		if (counter % 35 == 0 && counter % 70 != 0) {
			container->createDamageObject(OBJET_WINNIE_2);
		}
		else if (counter % 35 == 0 && counter % 70 == 0) {
			container->createDamageObject(OBJET_WINNIE_3);
		}
		counter++;
		if (counter >= 139) {
			father->setOnTakeCtrl(false);
			previous_state = AI_STATE_ATK_WINNIE_1;
			current_state = AI_STATE_IDLE;
			counter = 0;
		}
	}
}
void CBossAI::stateFishATK() {
	father->setOnTakeCtrl(true);
	father->setStatus(FATHER_STATUS_FACE_1);
	if (moveToTargetPos(350, 50)) {
		father->setStatus(FATHER_STATUS_FACE_2);
		if (counter == 10) {
			container->createDamageObject(OBJET_FISH);
		}
		counter++;
		if (counter >= 70) {
			father->setOnTakeCtrl(false);
			previous_state = AI_STATE_ATK_FISH;
			current_state = AI_STATE_IDLE;
			counter = 0;
		}
	}
```

```
}
void CBossAI::statePenATKR() {
	std::pair<int, int> pos = game_data->getFatherPos();
	father->OnKeyDown(KEY_RIGHT);
	if (pos.first>=700) {
		father->OnKeyUP(KEY_RIGHT);
		game_data->setFatherTopLeft(700, 270);
		father->setStatus(FATHER_STATUS_FACE_2);
		if (counter % gap_counter == 0 && counter % (gap_counter * 2) != 0) {
			container->createDamageObject(OBJET_PEN, 0);
			container->createDamageObject(OBJET_PEN, 200);
			container->createDamageObject(OBJET_PEN, 400);
			container->createDamageObject(OBJET_PEN, 600);
		}
		else if (counter % gap_counter == 0 && counter % (gap_counter*2) == 0) {
			container->createDamageObject(OBJET_PEN, 100);
			container->createDamageObject(OBJET_PEN, 300);
			container->createDamageObject(OBJET_PEN, 500);
		}
		counter++;
		if (counter >= (gap_counter * 4)) {
			father->setOnTakeCtrl(false);
			previous_state = AI_STATE_ATK_PEN_R;
			current_state = AI_STATE_IDLE;
			counter = 0;
		}
	}
}
void CBossAI::statePenATKL() {
	std::pair<int, int> pos = game_data->getFatherPos();
	father->OnKeyDown(KEY_LEFT);
	if (pos.first <= 0) {
		father->OnKeyUP(KEY_LEFT);
		game_data->setFatherTopLeft(0, 270);
		father->setStatus(FATHER_STATUS_FACE_2);
		if (counter % gap_counter == 0 && counter % (gap_counter * 2) != 0) {
			container->createDamageObject(OBJET_PEN, 100);
			container->createDamageObject(OBJET_PEN, 300);
			container->createDamageObject(OBJET_PEN, 500);
			container->createDamageObject(OBJET_PEN, 700);
		}
		else if (counter % gap_counter == 0 && counter % (gap_counter * 2) == 0) {
			container->createDamageObject(OBJET_PEN, 200);
			container->createDamageObject(OBJET_PEN, 400);
			container->createDamageObject(OBJET_PEN, 600);
		}
		counter++;
		if (counter >= (gap_counter * 4)) {
			father->setOnTakeCtrl(false);
			previous_state = AI_STATE_ATK_PEN_L;
			current_state = AI_STATE_IDLE;
			counter = 0;
		}
	}
}
void CBossAI::statePenATKR2() {
	std::pair<int, int> pos = game_data->getFatherPos();
	father->OnKeyDown(KEY_RIGHT);
	if (pos.first >= 700) {
		father->OnKeyUP(KEY_RIGHT);
		game_data->setFatherTopLeft(700, 270);
		father->setStatus(FATHER_STATUS_FACE_2);
		if (counter % 5 == 0) {
			container->createDamageObject(OBJET_PEN, 600-((counter/5)*50));
		}
		counter++;
		if (counter >= 55) {
			father->setOnTakeCtrl(false);
			previous_state = AI_STATE_ATK_PEN_R_2;
```

```
                current_state = AI_STATE_IDLE;
                counter = 0;
            }
        }
}
void CBossAI::statePenATKL2() {
        std::pair<int, int> pos = game_data->getFatherPos();
        father->OnKeyDown(KEY_LEFT);
        if (pos.first <= 0) {
            father->OnKeyUP(KEY_LEFT);
            game_data->setFatherTopLeft(0, 270);
            father->setStatus(FATHER_STATUS_FACE_2);
            if (counter % 5== 0) {
                container->createDamageObject(OBJET_PEN, 100 + ((counter / 5) * 50));
            }
            counter++;
            if (counter >= 55) {
                father->setOnTakeCtrl(false);
                previous_state = AI_STATE_ATK_PEN_L_2;
                current_state = AI_STATE_IDLE;
                counter = 0;
            }
        }
}
void CBossAI::statePillATK() {
        father->setOnTakeCtrl(true);
        father->setStatus(FATHER_STATUS_FACE_1);
        if (moveToTargetPos(350, 50)) {
            father->setStatus(FATHER_STATUS_FACE_2);
            if (counter % gap_counter == 0) {
                container->createDamageObject(OBJET_PILL, 350, 100, game_data);
            }
            counter++;
            if (counter >= (gap_counter * (17-gap_counter))) {
                father->setOnTakeCtrl(false);
                previous_state = AI_STATE_ATK_PILL;
                current_state = AI_STATE_IDLE;
                counter = 0;
            }
        }
}
void CBossAI::stateLighterATK() {
        std::pair<int, int> pos = game_data->getFatherPos();
        father->setOnTakeCtrl(true);
        father->setStatus(FATHER_STATUS_FACE_1);
        if (counter<100) {
            if (moveToTargetPos(0, 0)) {
                counter = 100;
            }
        }else if (counter >= 100 && counter<150) {
            father->setStatus(FATHER_STATUS_STAND);
            father->setFaceSide(FACE_RIGHT);
            game_data->setFatherMove(15, 0);
            if (counter % 10 == 0) {
                container->createDamageObject(OBJET_LIGHTER,pos.first);
            }
        }
        else if (counter >= 150) {
            father->setStatus(FATHER_STATUS_STAND);
            father->setFaceSide(FACE_LEFT);
            game_data->setFatherMove(-15, 0);
            if (counter % 15 == 0) {
                container->createDamageObject(OBJET_LIGHTER, game_data->getFatherPos().first);
            }
        }
        counter++;
        if (counter > 200) {
            father->setOnTakeCtrl(false);
            previous_state = AI_STATE_ATK_LIGHTER;
```

```
                        current_state = AI_STATE_IDLE;
                        counter = 0;
                }
        }
}
```

==========

CBossAI.h

==========

```
#include "Definitions.h"
#ifndef CBOSSAI_H
#define CBOSSAI_H
#define AI_STATE_IDLE -1
#define AI_STATE_MOVE_LEFT 0
#define AI_STATE_MOVE_RIGHT 1
#define AI_STATE_ATK_WINNIE_1 2
#define AI_STATE_ATK_WINNIE_2 3
#define AI_STATE_ATK_PEN_R 4
#define AI_STATE_ATK_PEN_L 5
#define AI_STATE_ATK_FISH 6
#define AI_STATE_ATK_PEN_R_2 7
#define AI_STATE_ATK_PEN_L_2 8
#define AI_STATE_ATK_PILL 9
#define AI_STATE_ATK_LIGHTER 10
#define AI_TOTAL_STATE_COUNT 11
namespace game_framework {
        class CBossAI {
        public:
                CBossAI();
                CBossAI(CFather*, CGameData*, CDamageContainer*, CBlood*);
                ~CBossAI();
                void OnLoop();
                void setAILock(bool);
        private:
                bool moveToTargetPos(int, int);
                void stateMoveRight();
                void stateMoveLeft();
                void stateWinnieATK1();
                void stateWinnieATK2();
                void stateFishATK();
                void statePenATKR();
                void statePenATKL();
                void statePenATKR2();
                void statePenATKL2();
                void statePillATK();
                void stateLighterATK();
                void nextStepChoise();
                int **decision_probiality_1;
                int **decision_probiality_2;
                CFather* father;
                CGameData* game_data;
                CDamageContainer* container;
                CBlood* blood;
                bool AI_Lock;
                int counter;
                int gap_counter;
                int current_state;
                int previous_state;
        };
}
#endif
```

==========

CDamageContainer.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
```

```
#include "CDamageContainer.h"
#include "CDamageObject.h"
#include "CDamagePen.h"
#include "CDamageFish.h"
#include "CDamageWinnie.h"
#include "CDamagePill.h"
#include "CDamageLighter.h"
#include "CBlood.h"
#include "CDaughter.h"
namespace game_framework {
	CDamageContainer::CDamageContainer() {
		this->winnie_loop = 0;
	}
	CDamageContainer::~CDamageContainer() {
	}
	void CDamageContainer::OnShow() {
		for (auto it: container) {
			it->OnShow();
		}
		this->grabageCollect();
	}
	void CDamageContainer::damageTake(CBlood * blood,CDaughter* d,std::pair<int,int> pos) {
		int x_mid = pos.first + (d->getW() / 2);
		int y_mid = pos.second + (d->getH() / 2);
		for (auto it : container) {
			pos = it->getPos();
			if (x_mid > pos.first && x_mid<pos.first + it->getW() && y_mid>pos.second && y_mid < pos.second +
it->getH()) {
				blood->setDaughterDamage(it->getDmg(),it->getInvincibleTime());
				break;
			}
		}
	}
	void CDamageContainer::OnMove() {
		for (auto it : container) {
			it->OnMove();
		}
		objectLoop();
	}
	void CDamageContainer::grabageCollect() {
		vector<CDamageObject*> temp;
		for (auto it : container) {
			if (it->isEnd()) {
				delete it;
			}
			else {
				temp.push_back(it);
			}
		}
		container.clear();
		container = temp;
	}
	void CDamageContainer::createDamageObject(int type,int x,int y,CGameData* game_data) {
		if (type == OBJET_PEN) {
			CDamagePen* pen=new CDamagePen();
			pen->loadBitmap();
			pen->setTopLeft(x, -600);
			container.push_back(pen);
		}
		if (type == OBJET_FISH) {
			CDamageFish* fish = new CDamageFish();
			fish->loadBitmap();
			container.push_back(fish);
		}
		if (type == OBJET_WINNIE_1) {
			CDamageWinnie* paper1= new CDamageWinnie(0);
			CDamageWinnie* paper2 = new CDamageWinnie(1.2);
			CDamageWinnie* paper3 = new CDamageWinnie(-1.2);
			CDamageWinnie* paper4 = new CDamageWinnie(0.6);
```

```
		CDamageWinnie* paper5 = new CDamageWinnie(-0.6);
		paper1->loadBitmap();
		paper2->loadBitmap();
		paper3->loadBitmap();
		paper4->loadBitmap();
		paper5->loadBitmap();
		container.push_back(paper1);
		container.push_back(paper2);
		container.push_back(paper3);
		container.push_back(paper4);
		container.push_back(paper5);
	}
	if (type == OBJET_WINNIE_2) {
		winnie_loop = 10;
	}
	if (type == OBJET_WINNIE_3) {
		winnie_loop = -10;
	}
	if (type == OBJET_PILL) {
		CDamagePill* pill = new CDamagePill(game_data);
		pill->loadBitmap();
		container.push_back(pill);
	}
	if (type == OBJET_LIGHTER) {
		CDamageLighter* lighter = new CDamageLighter(x);
		lighter->loadBitmap();
		container.push_back(lighter);
	}
}
void CDamageContainer::objectLoop() {
	if (winnie_loop > 0) {
		CDamageWinnie* paper = new CDamageWinnie((winnie_loop-5)*0.5);
		paper->loadBitmap();
		container.push_back(paper);
		winnie_loop--;
	}else if (winnie_loop < 0) {
		CDamageWinnie* paper = new CDamageWinnie((winnie_loop + 5)*0.5);
		paper->loadBitmap();
		container.push_back(paper);
		winnie_loop++;
	}
}
}
==============================
CDamageContainer.h
==============================
#pragma once
#include "CDamageObject.h"
#include "CBlood.h"
#include "CDaughter.h"
#ifndef CDAMAGECONTAINER_H
#define CDAMAGECONTAINER_H
#define OBJET_IDEL -1
#define OBJET_PEN 0
#define OBJET_LIGHTER 1
#define OBJET_WINNIE_1 2
#define OBJET_WINNIE_2 3
#define OBJET_WINNIE_3 4
#define OBJET_PILL 5
#define OBJET_FISH 6
namespace game_framework {
	class CDamageContainer {
	public:
		CDamageContainer();
		~CDamageContainer();
		void damageTake(CBlood *,CDaughter *,std::pair<int,int>);
		void createDamageObject(int, int = 0, int = 0, CGameData* = nullptr);
		void OnShow();
		void OnMove();
```

```
	private:
		void grabageCollect();
		void objectLoop();
		vector<CDamageObject*> container;
		int type;
		int winnie_loop;
	};
}
#endif
```

==========

CDamageFish.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CDamageFish.h"
namespace game_framework {
	CDamageFish::CDamageFish():
		CDamageObject(800, 400, 7, 1){
		setTopLeft(0, 600);
		this->counter = 0;
		this->flash = 0;
	}
	CDamageFish::~CDamageFish() {
	}
	void CDamageFish::loadBitmap() {
		pic.AddBitmap(IDB_FISH1, RGB(0, 0, 255));
		pic.AddBitmap(IDB_FISH2, RGB(0, 0, 255));
		pic.AddBitmap(IDB_FISH3, RGB(0, 0, 255));
		pic.AddBitmap(IDB_FISH4, RGB(0, 0, 255));
		pic.AddBitmap(IDB_FISH5, RGB(0, 0, 255));
		pic.AddBitmap(IDB_FISH6, RGB(0, 0, 255));
		pic.AddBitmap(IDB_FISH7, RGB(0, 0, 255));
		pic.SetDelayCount(4);
		alarm.LoadBitmap(IDB_ALARM_LONG, RGB(255, 255, 255));
	}
	void CDamageFish::OnMove() {
		if (counter > 30) {
			int p = pic.GetCurrentBitmapNumber();
			setTopLeft(x_range[p], y_range[p]);
			pic.OnMove();
		}
		if (pic.GetCurrentBitmapNumber() == 1) {
			CAudio::Instance()->Play(SEA_WAVE, false);
		}
		if (counter % 4 == 0) { flash = (flash + 1) % 2; }
		counter++;
	}
	void CDamageFish::OnShow() {
		if (counter > 30) {
			pic.SetTopLeft(100, 100);
			pic.OnShow();
		}
		else {
			if (flash) {
				alarm.SetTopLeft(100, 400);
				alarm.ShowBitmap();
			}
		}
	}
}
```

==========

CDamageFish.h

==========

```
#pragma once
#include "CDamageObject.h"
```

```
#ifndef CDAMAGEFISH_H
#define CDAMAGEFISH_H
namespace game_framework {
	class CDamageFish : public CDamageObject {
	public:
		CDamageFish();
		~CDamageFish();
		void OnMove();
		void loadBitmap();
		void OnShow();
		bool isEnd();
	protected:
		const int x_range[7] = { 700,560,385,110,110,110,110 };
		const int y_range[7] = { 400,400,240,170,170,170,230 };
		CMovingBitmap alarm;
		CAnimation pic;
		int counter;
		int flash;
	};
	inline bool CDamageFish::isEnd() {
		return counter > 64;
	}
}
#endif
==============================
CDamageLighter.cpp
==============================
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CDamageLighter.h"
namespace game_framework {
	CDamageLighter::CDamageLighter():
		CDamageObject(110, 150, 25, 30),speed(40) {
		lighter_pos = std::make_pair(350, 50);
		setTopLeft(328, -100);
		counter = 0;
	}
	CDamageLighter::CDamageLighter(int x):
		CDamageObject(110, 150, 20, 30), speed(40) {
		lighter_pos = std::make_pair(x, 50);
		setTopLeft(x-22, -100);
		counter = 0;
	}
	CDamageLighter::~CDamageLighter(){
	}
	void CDamageLighter::OnMove() {
		lighter_pos.second += speed;
		speed += 2;
		if (lighter_pos.second >= 402) {
			lighter_pos.second = 402;
		}
		if (counter > 30) {
			explosion.OnMove();
		}
		if (explosion.GetCurrentBitmapNumber() == 2) {
			pos.second = 350;
			CAudio::Instance()->Play(EXPLO_SOUND, false);
		}
		counter++;
	}
	void CDamageLighter::loadBitmap() {
		explosion.AddBitmap(IDB_BOOM_0, RGB(0, 0, 0));
		explosion.AddBitmap(IDB_BOOM_1, RGB(0, 0, 0));
		explosion.AddBitmap(IDB_BOOM_2, RGB(0, 0, 0));
		explosion.AddBitmap(IDB_BOOM_3, RGB(0, 0, 0));
```

```
		explosion.AddBitmap(IDB_BOOM_4, RGB(0, 0, 0));
		explosion.AddBitmap(IDB_BOOM_5, RGB(0, 0, 0));
		explosion.AddBitmap(IDB_BOOM_6, RGB(0, 0, 0));
		explosion.AddBitmap(IDB_BOOM_7, RGB(0, 0, 0));
		explosion.AddBitmap(IDB_BOOM_8, RGB(0, 0, 0));
		explosion.SetDelayCount(2);
		lighter.LoadBitmap(IDB_LIGHTER, RGB(0, 0, 0));
	}
	void CDamageLighter::OnShow() {
		lighter.SetTopLeft(lighter_pos.first, lighter_pos.second);
		explosion.SetTopLeft(pos.first, 400);
		if (counter <= 45) {
			lighter.ShowBitmap();
		}
		if (counter >= 20) {
			explosion.OnShow();
		}
	}
}
==============================
CDamageLighter.h
==============================
#pragma once
#include "CDamageObject.h"
#ifndef CDAMAGELIGHTER_H
#define CDAMAGELIGHTER_H
namespace game_framework {
	class CDamageLighter : public CDamageObject {
	public:
		CDamageLighter();
		CDamageLighter(int);
		~CDamageLighter();
		void OnMove();
		void loadBitmap();
		void OnShow();
		bool isEnd();
	private:
		std::pair<int, int> lighter_pos;
		CMovingBitmap lighter;
		CAnimation explosion;
		int speed;
		int counter;
	};
	inline bool CDamageLighter::isEnd() {
		return counter > 50;
	}
}
#endif
==============================
CDamageObject.cpp
==============================
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CDamageObject.h"
#include "CGameData.h"
namespace game_framework {
	CDamageObject::CDamageObject():
		W(0),H(0),damage(0), invincible_time(30){
	}
	CDamageObject::CDamageObject(int W,int H,int damage,int invincible_time) {
		this->W = W;
		this->H = H;
		this->damage = damage;
		this->invincible_time = invincible_time;
	}
	CDamageObject::~CDamageObject() {
```

```
	}
	void CDamageObject::setTopLeft(int x,int y) {
		pos.first=x;
		pos.second = y;
	}
}
```

==========

CDamageObject.h

==========

```
#pragma once
#ifndef CDAMAGEOBJECT_H
#define CDAMAGEOBJECT_H
namespace game_framework {
	class CDamageObject {
	public:
		CDamageObject();
		CDamageObject(int, int, int, int);
		virtual ~CDamageObject();
		void setTopLeft(int, int);
		int getW() const;
		int getH() const;
		int getDmg() const;
		int getInvincibleTime() const;
		std::pair<int, int> getPos() const;
		virtual void OnMove() = 0;
		virtual void OnShow() = 0;
		virtual bool isEnd() = 0;
	protected:
		CMovingBitmap pic;
		std::pair<int, int> pos;
		int W, H;
		int damage;
		int invincible_time;
	};
	inline int CDamageObject::getW() const {
		return W;
	}
	inline int CDamageObject::getH() const {
		return H;
	}
	inline int CDamageObject::getDmg() const {
		return damage;
	}
	inline int CDamageObject::getInvincibleTime() const {
		return invincible_time;
	}
	inline std::pair<int, int> CDamageObject::getPos() const {
		return pos;
	}
}
#endif
```

==========

CDamagePen.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "CDamagePen.h"
namespace game_framework {
	CDamagePen::CDamagePen() :
		CDamageObject(100, 600, 40, 30),speed(30){
		this->counter = 0;
		this->flash = 0;
		this->play = false;
	}
	CDamagePen::~CDamagePen() {
```

```
	}
	void CDamagePen::loadBitmap() {
		pic.LoadBitmap(IDB_PEN, RGB(0, 0, 255));
		alarm.LoadBitmap(IDB_ALARMPIC, RGB(255, 255, 255));
	}
	void CDamagePen::OnMove() {
		if (counter > 30) {
			pos.second += speed;
			speed += 2;
			if (pos.second > -100) {
				CAudio::Instance()->Play(G_HIT, false);
				pos.second = -100;
			}
		}
		if (counter % 4 == 0) { flash = (flash + 1) % 2; }
		counter++;
	}
	void CDamagePen::OnShow() {
		if (counter > 30) {
			pic.SetTopLeft(pos.first, pos.second);
			pic.ShowBitmap();
		}
		else {
			if (flash) {
				alarm.SetTopLeft(pos.first, 400);
				alarm.ShowBitmap();
			}
		}
	}
}
```

==========

CDamagePen.h

==========

```
#pragma once
#include "CDamageObject.h"
#ifndef CDAMAGEPEN_H
#define CDAMAGEPEN_H
namespace game_framework {
	class CDamagePen : public CDamageObject {
	public:
		CDamagePen();
		~CDamagePen();
		void OnMove();
		void loadBitmap();
		void OnShow();
		bool isEnd();
	private:
		CMovingBitmap alarm;
		int speed;
		int counter;
		int flash;
		bool play;
	};
	inline bool CDamagePen::isEnd() {
		return pos.second >= -100;
	}
}
#endif
```

==========

CDamagePill.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CDamagePill.h"
#include "CGameData.h"
```

```
namespace game_framework {
	CDamagePill::CDamagePill():
		CDamageObject(50, 50, 15, 5) {
		game_data = nullptr;
		counter = 0;
		setTopLeft(350, 100);
	}
	CDamagePill::CDamagePill(CGameData* game_data):
		CDamageObject(50, 50, 15, 5) {
		this->game_data = game_data;
		target_pos = game_data->getDaughterPos();
		target_pos.second += 50;
		counter = 0;
		setTopLeft(350, 100);
	}
	CDamagePill::~CDamagePill() {
	}
	void CDamagePill::OnMove() {
		if (counter < 10) {
			target_pos = game_data->getDaughterPos();
			target_pos.second += 50;
			move_vector = make_pair(target_pos.first-325, target_pos.second-75);
		}
		else {
			pos.first += (move_vector.first/15);
			pos.second += (move_vector.second/15);
		}
		counter++;
	}
	void CDamagePill::loadBitmap() {
		aim.LoadBitmap(IDB_AIM, RGB(255, 255, 255));
		pill.LoadBitmap(IDB_PILL, RGB(0, 0, 0));
	}
	void CDamagePill::OnShow() {
		aim.SetTopLeft(target_pos.first, target_pos.second);
		aim.ShowBitmap();
		if (counter >= 10) {
			pill.SetTopLeft(pos.first, pos.second);
			pill.ShowBitmap();
		}
	}
}
==============================
CDamagePill.h
==============================
#pragma once
#include "CDamageObject.h"
#include "CGameData.h"
#ifndef CDAMAGEPILL_H
#define CDAMAGEPILL_H
namespace game_framework {
	class CDamagePill : public CDamageObject {
	public:
		CDamagePill();
		CDamagePill(CGameData*);
		~CDamagePill();
		void OnMove();
		void loadBitmap();
		void OnShow();
		bool isEnd();
	private:
		std::pair<int, int> target_pos;
		std::pair<int, int> move_vector;
		CGameData* game_data;
		CMovingBitmap aim;
		CMovingBitmap pill;
		int counter;
	};
	inline bool CDamagePill::isEnd() {
```

```
		return counter > 30;
	}
}
#endif
```

==========

CDamageWinnie.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CDamageWinnie.h"
namespace game_framework {
	CDamageWinnie::CDamageWinnie(double a) :
		CDamageObject(70, 100, 20, 5), speed(30) {
		this->counter = 0;
		this->flash = 0;
		this->coef_a = a;
		pos = std::make_pair(350, 100);
	}
	CDamageWinnie::~CDamageWinnie() {
	}
	void CDamageWinnie::loadBitmap() {
		pic.LoadBitmap(IDB_WINNIE,RGB(0,0,0));
	}
	void CDamageWinnie::OnMove() {
		if (counter < 30) {
			pos.second -= speed;
			pos.first = int((double(pos.second)*coef_a) + coef_b);
			speed = (speed-3 < 0 ? 0 : speed-3);
		}
		else {
			pos.second += speed;
			pos.first = int((double(pos.second)*coef_a) + coef_b);
			speed += 2;
		}
		counter++;
	}
	void CDamageWinnie::OnShow() {
		pic.SetTopLeft(pos.first, pos.second);
		pic.ShowBitmap();
	}
}
```

==========

CDamageWinnie.h

==========

```
#pragma once
#include "CDamageObject.h"
#ifndef CDAMAGEWINNIE_H
#define CDAMAGEWINNIE_H
namespace game_framework {
	class CDamageWinnie : public CDamageObject {
	public:
		CDamageWinnie(double a);
		~CDamageWinnie();
		void OnMove();
		void loadBitmap();
		void OnShow();
		bool isEnd();
	private:
		const double coef_b = 350;
		int speed;
		int counter;
		int flash;
		double coef_a;
	};
	inline bool CDamageWinnie::isEnd() {
		return counter > 60;
```

```
	}
}
#endif
==============================
CDaughter.cpp
==============================
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CGameData.h"
#include "CDaughter.h"
namespace game_framework {
	CDaughter::CDaughter():
		x_speed(5),atk_counter(0),w(DAUGHTER_STAND_W),h(DAUGHTER_STAND_H){
		this->jump = false;
		this->move = false;
		this->attack = false;
		this->slide = false;
		this->lock = false;
		this->game_data = nullptr;
	}
	CDaughter::CDaughter(CGameData * game_data):
		x_speed(5),atk_counter(0),w(DAUGHTER_STAND_W),h(DAUGHTER_STAND_H){
		this->jump = false;
		this->move = false;
		this->attack = false;
		this->slide = false;
		this->lock = false;
		this->game_data = game_data;
	}
	CDaughter::~CDaughter() {
	}
	void CDaughter::LoadBitmap() {
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_1, RGB(0, 255, 0));
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_2, RGB(0, 255, 0));
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_3, RGB(0, 255, 0));
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_4, RGB(0, 255, 0));
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_5, RGB(0, 255, 0));
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_6, RGB(0, 255, 0));
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_7, RGB(0, 255, 0));
		walk_R.AddBitmap(IDB_DAUGHTER_WALK_R_8, RGB(0, 255, 0));
		walk_R.SetDelayCount(3);
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_1, RGB(0, 255, 0));
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_2, RGB(0, 255, 0));
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_3, RGB(0, 255, 0));
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_4, RGB(0, 255, 0));
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_5, RGB(0, 255, 0));
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_6, RGB(0, 255, 0));
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_7, RGB(0, 255, 0));
		walk_L.AddBitmap(IDB_DAUGHTER_WALK_L_8, RGB(0, 255, 0));
		walk_L.SetDelayCount(3);
		stand_R.LoadBitmap(IDB_DAUGHTER_STAND_R, RGB(0, 255, 0));
		stand_L.LoadBitmap(IDB_DAUGHTER_STAND_L, RGB(0, 255, 0));
		jump_R.LoadBitmap(IDB_DAUGHTER_JUMP_R, RGB(0, 255, 0));
		jump_L.LoadBitmap(IDB_DAUGHTER_JUMP_L, RGB(0, 255, 0));
		fall_R.LoadBitmap(IDB_DAUGHTER_FALL_R, RGB(0, 255, 0));
		fall_L.LoadBitmap(IDB_DAUGHTER_FALL_L, RGB(0, 255, 0));
		fly_R.LoadBitmap(IDB_DAUGHTER_FLY_R, RGB(0, 255, 0));
		fly_L.LoadBitmap(IDB_DAUGHTER_FLY_L, RGB(0, 255, 0));
		atk_R[0].LoadBitmap(IDB_DAUGHTER_ATK_R_1, RGB(0, 255, 0));
		atk_R[1].LoadBitmap(IDB_DAUGHTER_ATK_R_2, RGB(0, 255, 0));
		atk_R[2].LoadBitmap(IDB_DAUGHTER_ATK_R_3, RGB(0, 255, 0));
		atk_L[0].LoadBitmap(IDB_DAUGHTER_ATK_L_1, RGB(0, 255, 0));
		atk_L[1].LoadBitmap(IDB_DAUGHTER_ATK_L_2, RGB(0, 255, 0));
		atk_L[2].LoadBitmap(IDB_DAUGHTER_ATK_L_3, RGB(0, 255, 0));
		slide_L.LoadBitmap(IDB_DAUGHTER_SLIDING_L, RGB(0, 255, 0));
```

```
	slide_R.LoadBitmap(IDB_DAUGHTER_SLIDING_R, RGB(0, 255, 0));
	front = &stand_R;
}
void CDaughter::OnShow() {
	std::pair<int, int> pos = game_data->getDaughterPos();
	if (status == DAUGHTER_STATUS_ATK&&faceSide == FACE_LEFT) pos.first -= DAUGHTER_STAND_W;
	front->SetTopLeft(pos.first, pos.second);
	front->ShowBitmap();
}
void CDaughter::OnMove() {
	if (lock) {
		move = false;
	}
	if (fly) {
		this->OnFly();
		status = DAUGHTER_STATUS_FLY;
	}
	else if (slide) {
		this->OnSlide();
		status = DAUGHTER_STATUS_SLIDE;
	}
	else if (jump) {
		this->OnJump();
		status = (y_speed > 0 ? DAUGHTER_STATUS_JUMP : DAUGHTER_STATUS_FALL);
	}
	else if (attack) {
		this->OnAttack();
	}
	else if (move) {
		this->OnWalk();
		status = DAUGHTER_STATUS_WALK;
	}
	else {
		status = DAUGHTER_STATUS_STAND;
	}
	setFront();
}
void CDaughter::OnWalk() {
	switch (faceSide) {
	case FACE_RIGHT:
		walk_R.OnMove();
		game_data->setDaughterMove(x_speed,0);
		break;
	case FACE_LEFT:
		walk_L.OnMove();
		game_data->setDaughterMove(-x_speed,0);
		break;
	}
}
void CDaughter::OnFly() {
	switch (faceSide) {
	case FACE_RIGHT:
		game_data->setDaughterMove(x_speed, 0);
		break;
	case FACE_LEFT:
		game_data->setDaughterMove(-x_speed, 0);
		break;
	}
	if (x_speed > 10) {
		x_speed --;
	}
	else {
		fly = false;
		x_speed = 5;
	}
}
void CDaughter::OnJump() {
	game_data->setDaughterMove(0, -y_speed);
	y_speed--;
```

```
	if (move) {
		this->OnWalk();
	}
}
void CDaughter::OnAttack() {
	if (atk_delay_counter > 0) {
		status = DAUGHTER_STATUS_ATK;
		atk_delay_counter--;
	}
	else {
		atk_counter++;
		atk_counter %= 3;
		attack = false;
		status = DAUGHTER_STATUS_STAND;
	}
}
void CDaughter::OnSlide() {
	switch (faceSide) {
	case FACE_RIGHT:
		game_data->setDaughterMove(x_speed, 0);
		break;
	case FACE_LEFT:
		game_data->setDaughterMove(-x_speed, 0);
		break;
	}
	if (x_speed > 10) {
		x_speed--;
	}
	else {
		slide = false;
		x_speed = 5;
		h = DAUGHTER_STAND_H;
		w = DAUGHTER_STAND_W;
		game_data->setDaughterMove(0, -100);
	}
}
void CDaughter::OnFloorHit() {
	y_speed = initial_velocity;
	jump = false;
	second_jump = false;
}
void CDaughter::OnFootEmpty() {
	if (!jump) {
		y_speed = -10;
		jump = true;
		second_jump = true;
	}
}
void CDaughter::setFront() {
	switch (status){
	case DAUGHTER_STATUS_STAND:
		front = (faceSide == FACE_RIGHT ? &stand_R : &stand_L);
		break;
	case DAUGHTER_STATUS_WALK:
		front = (faceSide == FACE_RIGHT ? walk_R.GetCurrentBitmap() : walk_L.GetCurrentBitmap());
		break;
	case DAUGHTER_STATUS_JUMP:
		front = (faceSide == FACE_RIGHT ? &jump_R : &jump_L);
		break;
	case DAUGHTER_STATUS_FALL:
		front = (faceSide == FACE_RIGHT ? &fall_R : &fall_L);
		break;
	case DAUGHTER_STATUS_FLY:
		front = (faceSide == FACE_RIGHT ? &fly_R : &fly_L);
		break;
	case DAUGHTER_STATUS_ATK:
		front = (faceSide == FACE_RIGHT ? &atk_R[atk_counter] : &atk_L[atk_counter]);
		break;
	case DAUGHTER_STATUS_SLIDE:
```

```
		front = (faceSide == FACE_RIGHT ? &slide_R : &slide_L);
		break;
	default:
		break;
	}
}
void CDaughter::OnKeyDown(UINT nChar) {
	if (!outro){
		switch (nChar) {
		case KEY_RIGHT:
			move = true;
			faceSide = FACE_RIGHT;
			break;
		case KEY_LEFT:
			move = true;
			faceSide = FACE_LEFT;
			break;
		case KEY_Z:
			if (jump && y_speed>0) {
				y_speed = 0;
				x_speed = 30;
				fly = true;
			}
			else {
				attack = true;
				atk_delay_counter = DAUGHTER_ATK_DELAY_COUNT;
			}
			break;
		case KEY_X:
			if (!slide && !jump) {
				x_speed = 20;
				game_data->setDaughterMove(0, 100);
				h = DAUGHTER_SLIDE_H;
				w = DAUGHTER_SLIDE_W;
				slide = true;
				CAudio::Instance()->Play(DAUGHTER_SLIDE, false);
			}
			break;
		default:
			break;
		}
		if (!jump && move) {
			CAudio::Instance()->Play(DAUGHTER_FOOTSTEP, false);
		}
	}
}
void CDaughter::OnKeyUP(UINT nChar) {
	if (!outro){
		switch (nChar) {
		case KEY_RIGHT:
			if (faceSide == FACE_RIGHT) {
				move = false;
				faceSide = FACE_RIGHT;
				CAudio::Instance()->Stop(DAUGHTER_FOOTSTEP);
			}
			break;
		case KEY_LEFT:
			if (faceSide == FACE_LEFT) {
				move = false;
				faceSide = FACE_LEFT;
				CAudio::Instance()->Stop(DAUGHTER_FOOTSTEP);
			}
			break;
		case KEY_SPACE:
			if (jump && !second_jump) {
				y_speed = initial_velocity;
				second_jump = true;
			}
			else {
```

```
                                        jump = true;
                                }
                        default:
                                break;
                        }
                }
        }
}
```

==========

CDaughter.h

==========

```
#pragma once
#include "Definitions.h"
#ifndef CDAUGHTER_H
#define CDAUGHTER_H
#define DAUGHTER_STATUS_STAND 0
#define DAUGHTER_STATUS_WALK    1
#define DAUGHTER_STATUS_JUMP    2
#define DAUGHTER_STATUS_FALL    3
#define DAUGHTER_STATUS_FLY     4
#define DAUGHTER_STATUS_ATK     5
#define DAUGHTER_STATUS_SLIDE 6
#define DAUGHTER_ATK_DELAY_COUNT 3
#define DAUGHTER_STAND_H 200
#define DAUGHTER_STAND_W 100
#define DAUGHTER_SLIDE_H 100
#define DAUGHTER_SLIDE_W 200
namespace game_framework {
        class CDaughter {
        public:
                CDaughter();
                CDaughter(CGameData *);
                ~CDaughter();
                void LoadBitmap();
                void OnMove();
                void OnShow();
                bool isInATK() const;
                int    getXSpeed() const;
                int    getYSpeed() const;
                int        getStatus() const;
                int    getFaceSide() const;
                int    getH() const;
                int    getW() const;
                void OnFloorHit();
                void OnFootEmpty();
                void setFront();
                void OnKeyDown(UINT);
                void OnKeyUP(UINT);
                bool outro = false;
        private:
                void OnWalk();
                void OnFly();
                void OnJump();
                void OnAttack();
                void OnSlide();
                const int initial_velocity = 15;
                CGameData* game_data;
                CMovingBitmap* front;
                CMovingBitmap stand_R;
                CMovingBitmap stand_L;
                CMovingBitmap jump_R;
                CMovingBitmap jump_L;
                CMovingBitmap fall_R;
                CMovingBitmap fall_L;
                CMovingBitmap fly_R;
                CMovingBitmap fly_L;
                CMovingBitmap atk_R[3];
                CMovingBitmap atk_L[3];
                CMovingBitmap slide_R;
```

```
		CMovingBitmap slide_L;
		CAnimation walk_R;
		CAnimation walk_L;
		int x, y;
		int w, h;
		int faceSide;
		int y_speed;
		int x_speed;
		int status;
		int atk_delay_counter;
		int atk_counter;
		bool lock;
		bool move;
		bool fly;
		bool attack;
		bool jump;
		bool second_jump;
		bool slide;
	};
	inline bool CDaughter::isInATK() const{
		return status == DAUGHTER_STATUS_ATK || status == DAUGHTER_STATUS_FLY;
	}
	inline int CDaughter::getXSpeed() const {
		return x_speed;
	}
	inline int CDaughter::getYSpeed() const {
		return y_speed;
	}
	inline int CDaughter::getStatus() const {
		return status;
	}
	inline int CDaughter::getFaceSide() const {
		return faceSide;
	}
	inline int CDaughter::getH() const {
		return h;
	}
	inline int CDaughter::getW() const {
		return w;
	}
}
#endif
==============================
CDialog.cpp
==============================
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CDialog.h"
namespace game_framework {
	CDialog::CDialog(){
		dlg_iter = dlg.begin();
	}
	CDialog::~CDialog(){
		dlg.clear();
	}
	void CDialog::LoadBitmap(int state) {
		switch (state) {
		case 1:
			LoadBitmap_1();
			break;
		case 2:
			LoadBitmap_2();
			break;
		case 3:
			LoadBitmap_3();
			break;
```

```
		case 4:
			LoadBitmap_4();
			break;
		case 5:
			LoadBitmap_5();
			break;
		case 6:
			LoadBitmap_Fail();
			break;
		case 7:
			LoadBitmap_Success();
			break;
		}
}
void CDialog::LoadBitmap_1(){
		dlg.clear();
		daughter_1_1.LoadBitmap(IDB_DLG_DAUGHTER_1_1, RGB(0, 255, 0));
		daughter_1_2.LoadBitmap(IDB_DLG_DAUGHTER_1_2, RGB(0, 255, 0));
		daughter_1_3.LoadBitmap(IDB_DLG_DAUGHTER_1_3, RGB(0, 255, 0));
		daughter_1_4.LoadBitmap(IDB_DLG_DAUGHTER_1_4, RGB(0, 255, 0));
		dlg.insert(dlg.end(), daughter_1_1);
		dlg.insert(dlg.end(), daughter_1_2);
		dlg.insert(dlg.end(), daughter_1_3);
		dlg.insert(dlg.end(), daughter_1_4);
		dlg_iter = dlg.begin();
}
void CDialog::LoadBitmap_2() {
		dlg.clear();
		daughter_2_1.LoadBitmap(IDB_DLG_DAUGHTER_2_1, RGB(0, 255, 0));
		daughter_2_2.LoadBitmap(IDB_DLG_DAUGHTER_2_2, RGB(0, 255, 0));
		daughter_2_3.LoadBitmap(IDB_DLG_DAUGHTER_2_3, RGB(0, 255, 0));
		dlg.insert(dlg.end(), daughter_2_1);
		dlg.insert(dlg.end(), daughter_2_2);
		dlg.insert(dlg.end(), daughter_2_3);
		dlg_iter = dlg.begin();
}
void CDialog::LoadBitmap_3() {
		dlg.clear();
		daughter_3_1.LoadBitmap(IDB_DLG_DAUGHTER_3_1, RGB(0, 255, 0));
		daughter_3_2.LoadBitmap(IDB_DLG_DAUGHTER_3_2, RGB(0, 255, 0));
		dlg.insert(dlg.end(), daughter_3_1);
		dlg.insert(dlg.end(), daughter_3_2);
		dlg_iter = dlg.begin();
}
void CDialog::LoadBitmap_4() {
		dlg.clear();
		father_4.LoadBitmap(IDB_DLG_FATHER_4, RGB(0, 255, 0));
		daughter_4_1.LoadBitmap(IDB_DLG_DAUGHTER_4_1, RGB(0, 255, 0));
		daughter_4_2.LoadBitmap(IDB_DLG_DAUGHTER_4_2, RGB(0, 255, 0));
		dlg.insert(dlg.end(), father_4);
		dlg.insert(dlg.end(), daughter_4_1);
		dlg.insert(dlg.end(), daughter_4_2);
		dlg_iter = dlg.begin();
}
void CDialog::LoadBitmap_5() {
		dlg.clear();
		daughter_5_1.LoadBitmap(IDB_DLG_DAUGHTER_5_1, RGB(0, 255, 0));
		father_5_2.LoadBitmap(IDB_DLG_FATHER_5_2, RGB(0, 255, 0));
		dlg.insert(dlg.end(), daughter_5_1);
		dlg.insert(dlg.end(), father_5_2);
		dlg_iter = dlg.begin();
}
void CDialog::LoadBitmap_Fail() {
		dlg.clear();
		father_fail.LoadBitmap(IDB_DLG_FATHER_FAIL, RGB(0, 255, 0));
		dlg.insert(dlg.end(), father_fail);
		dlg_iter = dlg.begin();
}
void CDialog::LoadBitmap_Success() {
```

```
		dlg.clear();
		daughter_success.LoadBitmap(IDB_DLG_DAUGHTER_SUCCESS, RGB(0, 255, 0));
		father_success.LoadBitmap(IDB_DLG_FATHER_SUCCESS, RGB(0, 255, 0));
		dlg.insert(dlg.end(), daughter_success);
		dlg.insert(dlg.end(), father_success);
		dlg_iter = dlg.begin();
	}
	void CDialog::OnKeyDown(UINT nChar) {
		if (nChar == KEY_ENTER && dlg_iter != dlg.end()) {
			dlg_iter++;				//按下 enter 就換一張圖
		}
	}
	void CDialog::OnShow(){
		if (dlg_iter != dlg.end()) {
			dlg_iter->SetTopLeft(0, 400);
			dlg_iter->ShowBitmap();
		}
	}
}
==============================
CDialog.h
==============================
#pragma once
#include <list>
#include "Definitions.h"
#ifndef CDIALOG_H
#define CDIALOG_H
namespace game_framework {
	class CDialog {
	public:
		CDialog();
		~CDialog();
		void LoadBitmap(int);
		void OnKeyDown(UINT);
		bool IsFinished() const;	//對話結束
		void OnShow();
	private:
		void LoadBitmap_1();
		void LoadBitmap_2();
		void LoadBitmap_3();
		void LoadBitmap_4();
		void LoadBitmap_5();
		void LoadBitmap_Fail();
		void LoadBitmap_Success();
		list<CMovingBitmap> dlg;
		list<CMovingBitmap>::iterator	dlg_iter;
		CMovingBitmap daughter_1_1;
		CMovingBitmap daughter_1_2;
		CMovingBitmap daughter_1_3;
		CMovingBitmap daughter_1_4;
		CMovingBitmap daughter_2_1;
		CMovingBitmap daughter_2_2;
		CMovingBitmap daughter_2_3;
		CMovingBitmap daughter_3_1;
		CMovingBitmap daughter_3_2;
		CMovingBitmap father_4;
		CMovingBitmap daughter_4_1;
		CMovingBitmap daughter_4_2;
		CMovingBitmap daughter_5_1;
		CMovingBitmap father_5_2;
		CMovingBitmap father_fail;
		CMovingBitmap daughter_success;
		CMovingBitmap father_success;
	};
	inline bool CDialog::IsFinished() const{
		return dlg_iter == dlg.end();
	}
}
#endif
```

==========
CFather.cpp
==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CGameData.h"
#include "CBlood.h"
#include "CFather.h"
namespace game_framework {
	CFather::CFather():
		x_speed(5), y_speed(0) {
		fall = false;
		this->game_data = nullptr;
		this->blood = nullptr;
	}
	CFather::CFather(CGameData * game_data,CBlood* blood):
		x_speed(5), y_speed(0) {
		fall = false;
		this->game_data = game_data;
		this->blood = blood;
	}
	CFather::~CFather() {
	}
	void CFather::LoadBitmap() {
		walk_R.AddBitmap(IDB_FATHER_WALK_R_1, RGB(0, 0, 255));
		walk_R.AddBitmap(IDB_FATHER_WALK_R_2, RGB(0, 0, 255));
		walk_R.AddBitmap(IDB_FATHER_WALK_R_3, RGB(0, 0, 255));
		walk_R.AddBitmap(IDB_FATHER_WALK_R_4, RGB(0, 0, 255));
		walk_R.SetDelayCount(6);
		walk_L.AddBitmap(IDB_FATHER_WALK_L_1, RGB(0, 0, 255));
		walk_L.AddBitmap(IDB_FATHER_WALK_L_2, RGB(0, 0, 255));
		walk_L.AddBitmap(IDB_FATHER_WALK_L_3, RGB(0, 0, 255));
		walk_L.AddBitmap(IDB_FATHER_WALK_L_4, RGB(0, 0, 255));
		walk_L.SetDelayCount(6);
		stop_L.LoadBitmap(IDB_FATHER_WALK_L_2, RGB(0, 0, 255));
		stop_R.LoadBitmap(IDB_FATHER_WALK_R_2, RGB(0, 0, 255));
		stand_1.LoadBitmap(IDB_FATHER_STAND_1, RGB(0, 0, 255));
		stand_2.LoadBitmap(IDB_FATHER_STAND_2, RGB(0, 0, 255));
		front = &stop_L;
	}
	void CFather::OnShow() {
		std::pair<int, int> pos = game_data->getFatherPos();
		front->SetTopLeft(pos.first, pos.second);
		front->ShowBitmap();
	}
	void CFather::OnMove() {
		if (game_data->getFatherPos().second<=265 && !isBeCtrled) {
			this->OnFall();
			status = FATHER_STATUS_STAND;
		}
		else if (move) {
			this->OnWalk();
			status = FATHER_STATUS_WALK;
		}
		else{
			status = FATHER_STATUS_STAND;
		}
		if (game_data->getFatherPos().second > 270) {
			game_data->setFatherTopLeft(game_data->getFatherPos().first, 270);
			y_speed = 0;
		}
		setFront();
	}
	void CFather::OnWalk() {
		switch (faceSide) {
```

```
		case FACE_RIGHT:
			walk_R.OnMove();
			game_data->setFatherMove(x_speed, 0);
			break;
		case FACE_LEFT:
			walk_L.OnMove();
			game_data->setFatherMove(-x_speed, 0);
			break;
		}
}
void CFather::DamageTake() {
		blood->setFatherDamage(3);
}
void CFather::OnFall(){
		game_data->setFatherMove(0, -y_speed);
		y_speed--;
		if (move) {
			this->OnWalk();
		}
}
void CFather::setOnTakeCtrl(bool ctrl) {
		isBeCtrled = ctrl;
}
void CFather::setStatus(int s) {
		status = s;
		setFront();
}
void CFather::setFaceSide(int f) {
		faceSide = f;
		setFront();
}
void CFather::setFront() {
		switch (status) {
		case FATHER_STATUS_STAND:
			front = (faceSide == FACE_RIGHT ? &stop_R : &stop_L);
			break;
		case FATHER_STATUS_WALK:
			front = (faceSide == FACE_RIGHT ? walk_R.GetCurrentBitmap() : walk_L.GetCurrentBitmap());
			break;
		case FATHER_STATUS_FACE_1:
			front = &stand_1;
			break;
		case FATHER_STATUS_FACE_2:
			front = &stand_2;
			break;
		}
}
void CFather::OnKeyDown(UINT nChar) {
		switch (nChar) {
		case KEY_RIGHT:
			move = true;
			faceSide = FACE_RIGHT;
			break;
		case KEY_LEFT:
			move = true;
			faceSide = FACE_LEFT;
			break;
		}
}
void CFather::OnKeyUP(UINT nChar) {
		switch (nChar) {
		case KEY_RIGHT:
			if (faceSide == FACE_RIGHT) {
				move = false;
				faceSide = FACE_RIGHT;
			}
			break;
		case KEY_LEFT:
			if (faceSide == FACE_LEFT) {
```

```
                        move = false;
                        faceSide = FACE_LEFT;
                    }
                    break;
                }
        }
}
```

==========

CFather.h

==========

```
#pragma once
#include "Definitions.h"
#ifndef CFATHER_H
#define CFATHER_H
#define FATHER_STATUS_STAND 0
#define FATHER_STATUS_WALK    1
#define FATHER_STATUS_FACE_1    2
#define FATHER_STATUS_FACE_2    3
#define FATHER_H 230
#define FATHER_W 100
namespace game_framework {
        class CFather {
        public:
                CFather();
                CFather(CGameData*,CBlood*);
                ~CFather();
                void LoadBitmap();
                void OnShow();
                void OnMove();
                void setOnTakeCtrl(bool);
                void setStatus(int);
                void setFaceSide(int);
                void DamageTake();
                void OnKeyDown(UINT);
                void OnKeyUP(UINT);
        private:
                void OnWalk();
                void setFront();
                void OnFall();
                CGameData* game_data;
                CBlood * blood;
                CMovingBitmap* front;
                CAnimation walk_R;
                CAnimation walk_L;
                CMovingBitmap stop_R;
                CMovingBitmap stop_L;
                CMovingBitmap stand_1;
                CMovingBitmap stand_2;
                bool move;
                bool fall;
                bool isBeCtrled;
                int faceSide;
                int status;
                int x_speed;
                int y_speed;
        };
}
#endif
```

==========

CGameData.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include <utility>
#include "CDaughter.h"
#include "CFather.h"
```

```
#include "CGameData.h"
#include "CGameMap.h"
namespace game_framework {
	CGameData::CGameData(){
	}
	CGameData::~CGameData(){
	}
	void CGameData::setDaughterTopLeft(int x, int y) {
		daughter = std::make_pair(x, y);
	}
	void CGameData::setFatherTopLeft(int x, int y) {
		father = std::make_pair(x, y);
	}
	void CGameData::setMapTopLeft(int x, int y) {
		map = std::make_pair(x, y);
	}
	void CGameData::setDaughterMove(int x, int y) {
		daughter.first += x;
		daughter.second += y;
	}
	void CGameData::setFatherMove(int x, int y) {
		father.first += x;
		father.second += y;
	}
	void CGameData::setMapMove(int x, int y) {
		map.first += x;
		map.second += y;
	}
	void CGameData::checkMove(CGameMap *m, CDaughter *d, CFather *f){
		int x_right = daughter.first + map.first;
		int	y_top = daughter.second + map.second;
		int x_mid = x_right + (d->getW() / 2);
		int y_mid = y_top + (d->getH() / 2);
		int x_left = x_right + d->getW();
		int y_bot = y_top + d->getH();
		if (m->isBlock(x_mid, y_bot)) {
			d->OnFloorHit();
			daughter.second = m->getBlockHeight(y_bot) - d->getH() - map.second;
		}
		if (m->isBlock(x_left, y_mid)) {
			daughter.first = m->getBlockLeft(x_left) - d->getW() - map.first;
		}
		if (m->isBlock(x_right, y_mid)) {
			daughter.first = m->getBlockRight(x_right) - map.first;
		}
		if (m->isBlock(x_mid, y_top)) {
			daughter.second = m->getBlockBot(y_top) - map.second;
		}
		if (m->isEmpty(x_mid, y_bot)) {
			d->OnFootEmpty();
		}
		if (daughter.first + d->getW() > 700 && map.first + SIZE_X < m->getMapW()){
			daughter.first = 699 - d->getW();
			setMapMove(d->getXSpeed(), 0);
		}
		else if (daughter.first < 100 && map.first >0) {
			daughter.first = 101;
			setMapMove(-d->getXSpeed(), 0);
		}
		if (x_right<0) {
			daughter.first = 0;
		}
		else if (x_left > m->getMapW()) {
			daughter.first = SIZE_X - d->getW();
		}
		if (map.first < 0) {
			map.first = 0;
		}
		else if (map.first + SIZE_X > m->getMapW()) {
```

```
		map.first = m->getMapW() - SIZE_X;
	}
	if (f != nullptr) {
		if (d->isInATK()) {
			checkDamage(d, f);
		}
	}
}
void CGameData::checkDamage(CDaughter *d, CFather *f){
	int d_x_left = daughter.first;
	int d_x_right = d_x_left + d->getW();
	int d_x_left_border = daughter.first- d->getW();
	int d_x_right_border = d_x_left + (d->getW()*2);
	int d_y_top = daughter.second;
	int d_y_bot = daughter.second + d->getH();
	int f_x_mid = ((father.first*2) + FATHER_W)/2;
	int f_y_mid = father.second + (FATHER_W/2);
	if (d->getFaceSide() == FACE_LEFT) {
		if (f_x_mid > d_x_left_border && f_x_mid<d_x_left && f_y_mid>d_y_top && f_y_mid < d_y_bot) {
			f->DamageTake();
		}
	}
	else {
		if (f_x_mid < d_x_right_border && f_x_mid>d_x_right && f_y_mid>d_y_top && f_y_mid < d_y_bot) {
			f->DamageTake();
		}
	}
}
}
==============================
CGameData.h
==============================
#pragma once
#include "Definitions.h"
#ifndef CGAMEDATA_H
#define CGAMEDATA_H
namespace game_framework {
	class CGameData {
	public:
		CGameData();
		~CGameData();
		void setDaughterTopLeft(int, int);
		void setFatherTopLeft(int, int);
		void setMapTopLeft(int, int);
		void setDaughterMove(int, int);
		void setFatherMove(int, int);
		void setMapMove(int, int);
		std::pair<int, int> getDaughterPos() const;
		std::pair<int, int> getFatherPos() const;
		std::pair<int, int> getMapPos() const;
		void checkMove(CGameMap *,CDaughter *,CFather *);
		void checkDamage(CDaughter *, CFather *);
	private:
		std::pair<int, int> daughter;
		std::pair<int, int> map;
		std::pair<int, int> father;
	};
	inline std::pair<int, int> CGameData::getDaughterPos() const{
		return daughter;
	}
	inline std::pair<int, int> CGameData::getFatherPos() const{
		return father;
	}
	inline std::pair<int, int> CGameData::getMapPos() const{
		return map;
	}
}
#endif
================================
```

CGameMap.cpp

```
==============================
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CGameData.h"
#include "CGameMap.h"
namespace game_framework {
	CGameMap::CGameMap():
		map_w(0), map_h(0), box_num_x(0), box_num_y(0){
		this->map_data = nullptr;
		this->game_data = nullptr;
	}
	CGameMap::CGameMap(CGameData *game_data, int map_w,int map_h){
		this->map_w = map_w;
		this->map_h = map_h;
		this->box_num_x = map_w / BOX_W;
		this->box_num_y = map_h / BOX_H;
		this->game_data = game_data;
	}
	CGameMap::~CGameMap() {
		for (int i = 0; i < box_num_y; i++) {
			delete[] map_data[i];
		}
		delete[] map_data;
	}
	void CGameMap::LoadBitmap(int IDB_BITMAP) {
		map.LoadBitmap(IDB_BITMAP);
	}
	void CGameMap::OnShow() {
		std::pair<int, int> pos = game_data->getMapPos();
		map.SetTopLeft(-pos.first, -pos.second);
		map.ShowBitmap();
	}
}
==============================
```

CGameMap.h

```
==============================
#pragma once
#include "Definitions.h"
#ifndef CGAMEMAP_H
#define CGAMEMAP_H
#define MAP_EMPTY 0
#define MAP_BLOCK 1
namespace game_framework {
	class CGameMap {
	public:
		CGameMap();
		CGameMap(CGameData *, int, int);
		virtual ~CGameMap();
		void LoadBitmap(int);
		int getBlockHeight(int);
		int getBlockBot(int);
		int getBlockLeft(int);
		int getBlockRight(int);
		int getMapH();
		int getMapW();
		bool isBlock(int, int);
		bool isEmpty(int, int);
		void OnShow();
	protected:
		int **map_data;
		int box_num_x, box_num_y;
	private:
		CGameData *game_data;
		CMovingBitmap map;
		int map_w;
```

```
		int	map_h;
	};
	inline int CGameMap::getMapH() {
		return map_h;
	}
	inline int CGameMap::getMapW() {
		return map_w;
	}
	inline int CGameMap::getBlockHeight(int y) {
		int gy = (y / BOX_H)*BOX_H;
		return gy;
	}
	inline int CGameMap::getBlockBot(int y) {
		int gy = (y / BOX_H + 1)*BOX_H;
		return gy;
	}
	inline int CGameMap::getBlockLeft(int x) {
		int gx = (x / BOX_W)*BOX_W;
		return gx;
	}
	inline int CGameMap::getBlockRight(int x) {
		int gx = (x / BOX_W + 1)*BOX_W;
		return gx;
	}
	inline bool CGameMap::isBlock(int x, int y) {
		int gx = x / BOX_W;
		int gy = y / BOX_H;
		return map_data[gy][gx] == MAP_BLOCK;
	}
	inline bool CGameMap::isEmpty(int x, int y) {
		int gx = x / BOX_W;
		int gy = y / BOX_H;
		return map_data[gy][gx] == MAP_EMPTY;
	}
}
#endif
```

==========

CGameMap01.cpp

==========

```
#pragma once
#include "stdafx.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CGameMap.h"
#include "CGameData.h"
#include "CGameMap01.h"
namespace game_framework {
	CGameMap01::CGameMap01() :
		CGameMap() {
	}
	CGameMap01::CGameMap01(CGameData *game_data, int map_w, int map_h) :
		CGameMap(game_data, map_w, map_h) {
		int map_init[6][8] = {
			{ 1,1,1,1,1,1,1,1 },
			{ 1,0,0,0,0,0,0,1 },
			{ 1,0,0,0,0,0,0,1 },
			{ 0,0,0,0,0,0,0,1 },
			{ 0,0,0,0,0,1,1,1 },
			{ 1,1,1,1,1,1,1,1 }
		};
		map_data = new int*[box_num_y];
		for (int i = 0; i < box_num_y; i++) {
			map_data[i] = new int[box_num_x];
			for (int j = 0; j < box_num_x; j++) {
				map_data[i][j] = map_init[i][j];
			}
		}
	}
```

```
	CGameMap01::~CGameMap01() {
	}
}
```

==========

CGameMap01.h

==========

```
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#ifndef CGAMEMAP01_H
#define CGAMEMAP01_H
namespace game_framework {
	class CGameMap01 : public CGameMap {
	public:
		CGameMap01();
		CGameMap01(CGameData*, int, int);
		~CGameMap01();
	};
}
#endif
```

==========

CGameMap02.cpp

==========

```
#include "stdafx.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CGameMap.h"
#include "CGameData.h"
#include "CGameMap02.h"
namespace game_framework {
	CGameMap02::CGameMap02():
		CGameMap() {
	}
	CGameMap02::CGameMap02(CGameData *game_data,int map_w,int map_h) :
		CGameMap(game_data,map_w,map_h) {
		int map_init[6][16] = {
			{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,1 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1 },
			{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 }
		};
		map_data = new int*[box_num_y];
		for (int i = 0; i < box_num_y; i++) {
			map_data[i] = new int[box_num_x];
			for (int j = 0; j < box_num_x; j++) {
				map_data[i][j] = map_init[i][j];
			}
		}
	}
	CGameMap02::~CGameMap02() {
	}
}
```

==========

CGameMap02.h

==========

```
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#ifndef CGAMEMAP02_H
#define CGAMEMAP02_H
namespace game_framework {
	class CGameMap02 : public CGameMap {
	public:
		CGameMap02();
		CGameMap02(CGameData*, int, int);
		~CGameMap02();
```

```
	};
}
#endif
```
==========
CGameMap03.cpp
==========
```
#include "stdafx.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CGameMap.h"
#include "CGameData.h"
#include "CGameMap03.h"
namespace game_framework {
	CGameMap03::CGameMap03():
		CGameMap() {
	}
	CGameMap03::CGameMap03(CGameData *game_data,int map_w,int map_h) :
		CGameMap(game_data,map_w,map_h) {
		int map_init[6][16] = {
			{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 }
		};
		map_data = new int*[box_num_y];
		for (int i = 0; i < box_num_y; i++) {
			map_data[i] = new int[box_num_x];
			for (int j = 0; j < box_num_x; j++) {
				map_data[i][j] = map_init[i][j];
			}
		}
	}
	CGameMap03::~CGameMap03() {
	}
}
```
==========
CGameMap03.h
==========
```
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#ifndef CGAMEMAP03_H
#define CGAMEMAP03_H
namespace game_framework {
	class CGameMap03 : public CGameMap {
	public:
		CGameMap03();
		CGameMap03(CGameData*, int, int);
		~CGameMap03();
	};
}
#endif
```
==========
CGameMap04.cpp
==========
```
#include "stdafx.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CGameMap.h"
#include "CGameData.h"
#include "CGameMap04.h"
namespace game_framework {
	CGameMap04::CGameMap04() :
		CGameMap() {
	}
```

```
	CGameMap04::CGameMap04(CGameData *game_data, int map_w, int map_h) :
		CGameMap(game_data, map_w, map_h) {
		int map_init[6][16] = {
			{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 },
			{ 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 }
		};
		map_data = new int*[box_num_y];
		for (int i = 0; i < box_num_y; i++) {
			map_data[i] = new int[box_num_x];
			for (int j = 0; j < box_num_x; j++) {
				map_data[i][j] = map_init[i][j];
			}
		}
	}
	CGameMap04::~CGameMap04() {
	}
}
==============================
CGameMap04.h
==============================
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#ifndef CGAMEMAP04_H
#define CGAMEMAP04_H
namespace game_framework {
	class CGameMap04 : public CGameMap {
	public:
		CGameMap04();
		CGameMap04(CGameData*, int, int);
		~CGameMap04();
	};
}
#endif
==============================
CGameMap05.cpp
==============================
#include "stdafx.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "CGameMap.h"
#include "CGameData.h"
#include "CGameMap05.h"
namespace game_framework {
	CGameMap05::CGameMap05() :
		CGameMap() {
	}
	CGameMap05::CGameMap05(CGameData *game_data, int map_w, int map_h) :
		CGameMap(game_data, map_w, map_h) {
		int map_init[6][8] = {
			{ 0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0 },
			{ 0,0,0,0,0,0,0,0 },
			{ 1,1,1,1,1,1,1,1 }
		};
		map_data = new int*[box_num_y];
		for (int i = 0; i < box_num_y; i++) {
			map_data[i] = new int[box_num_x];
			for (int j = 0; j < box_num_x; j++) {
				map_data[i][j] = map_init[i][j];
			}
		}
```

```
	}
	CGameMap05::~CGameMap05() {
	}
}
```

==========

CGameMap05.h

==========

```
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#ifndef CGAMEMAP05_H
#define CGAMEMAP05_H
namespace game_framework {
	class CGameMap05 : public CGameMap {
	public:
		CGameMap05();
		CGameMap05(CGameData*, int, int);
		~CGameMap05();
	};
}
#endif
```

==========

CGameState01.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CGameState01.h"
namespace game_framework {
	CGameState01::CGameState01(CGame *g) :
		CGameState(g), game_data(), daughter(&game_data), map01(&game_data, 800, 600), outro(false) {
		this->count = 0;
	}
	CGameState01::~CGameState01() {
	}
	void CGameState01::OnBeginState(int daughterHP) {
		blood.setDaughterHP(daughterHP);
		CAudio::Instance()->Play(AUDIO_DROPS, true);
	}
	void CGameState01::OnMove() {
		daughter.OnMove();
		outro = game_data.getDaughterPos().first < 100;
		if (game_data.getDaughterPos().first == 100)
			CAudio::Instance()->Play(OPEN_TOILET_DOOR, false);
		if (outro) {
			transition.OnOutro(&daughter, FACE_LEFT);
			CAudio::Instance()->Stop(AUDIO_DROPS);
			if (transition.transit)
				GotoGameState(GAME_STATE02, blood.getDaughterHP());
		} else { game_data.checkMove(&map01, &daughter, nullptr); }
	}
	void CGameState01::OnInit() {
		ShowInitProgress(0);
		map01.LoadBitmap(IDB_MAP01);
		daughter.LoadBitmap();
		blood.LoadBitmap();
		transition.LoadBitmap();
		dialog.LoadBitmap(1);
		game_data.setDaughterTopLeft(400, 300);
		game_data.setMapTopLeft(0, 0);
		CAudio::Instance()->Load(AUDIO_DROPS, "sounds\\drops.mp3");
		CAudio::Instance()->Load(OPEN_TOILET_DOOR, "sounds\\open_toilet_door.mp3");
		ShowInitProgress(20);
	}
	void CGameState01::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
```

```
		dialog.OnKeyDown(nChar);
		if (dialog.IsFinished())
			daughter.OnKeyDown(nChar);
		if (nChar == KEY_ESC)
			//GotoGameState(GAME_STATE_OVER, blood.getDaughterHP());
			PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}
	void CGameState01::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
		if (dialog.IsFinished())
			daughter.OnKeyUP(nChar);
		switch (nChar) {
			case KEY_F2:
				CAudio::Instance()->Stop(AUDIO_DROPS);
				GotoGameState(GAME_STATE02, blood.getDaughterHP());
		}
	}
	void CGameState01::OnShow() {
		map01.OnShow();
		daughter.OnShow();
		blood.OnShow(false);
		transition.OnShow();
		if (!outro) {
			dialog.OnShow();
		}
	}
}
==============================
CGameState01.h
==============================
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#include "CGameMap01.h"
#include "CDaughter.h"
#include "CGameData.h"
#include "CBlood.h"
#include "CTransition.h"
#include "CDialog.h"
#ifndef CGAMESTATE01_H
#define CGAMESTATE01_H
namespace game_framework {
	class CGameState01 : public CGameState {
	public:
		CGameState01(CGame *g);
		~CGameState01();
		void OnBeginState(int daughterHP);
		void OnInit();
		void OnKeyDown(UINT, UINT, UINT);
		void OnKeyUp(UINT, UINT, UINT);
	protected:
		void OnMove();
		void OnShow();
	private:
		CGameMap01 map01;
		CDaughter daughter;
		CGameData game_data;
		CBlood blood;
		CTransition transition;
		CDialog dialog;
		int count;
		bool outro;
	};
}
#endif
==============================
CGameState02.cpp
==============================
#include "stdafx.h"
#include "Resource.h"
```

```
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CGameState02.h"
namespace game_framework {
	CGameState02::CGameState02(CGame *g) :
		CGameState(g), game_data(), daughter(&game_data), map02(&game_data, 1600, 600),
		intro(true), outro(false){
	}
	CGameState02::~CGameState02() {
	}
	void CGameState02::OnBeginState(int daughterHP) {
		blood.setDaughterHP(daughterHP);
	}
	void CGameState02::OnMove() {
		intro = game_data.getDaughterPos().first < 0;
		outro = game_data.getDaughterPos().first > 640;
		daughter.OnMove();
		if (game_data.getDaughterPos().first == -35)
			CAudio::Instance()->Play(CLOSE_DOOR, false);
		if (outro) {
			transition.OnOutro(&daughter, FACE_RIGHT);
			if (transition.transit)
				GotoGameState(GAME_STATE03, blood.getDaughterHP());
		}
		else if (!outro && !intro) {
			game_data.checkMove(&map02, &daughter,nullptr);
		}
		transition.OnIntro(&game_data, &daughter, FACE_RIGHT, 0);
	}
	void CGameState02::OnInit(){
		ShowInitProgress(20);
		map02.LoadBitmap(IDB_MAP02);
		daughter.LoadBitmap();
		blood.LoadBitmap();
		transition.LoadBitmap();
		dialog.LoadBitmap(2);
		game_data.setDaughterTopLeft(-100, 300);
		game_data.setMapTopLeft(0, 0);
		CAudio::Instance()->Load(CLOSE_DOOR, "sounds\\close_door.wav");
		ShowInitProgress(40);
	}
	void CGameState02::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
		dialog.OnKeyDown(nChar);
		if (dialog.IsFinished())
			daughter.OnKeyDown(nChar);
		if (nChar == KEY_ESC)
			//GotoGameState(GAME_STATE_OVER, blood.getDaughterHP());
			PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}
	void CGameState02::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
		if (dialog.IsFinished())
			daughter.OnKeyUP(nChar);
		switch (nChar) {
		case KEY_F2:
			GotoGameState(GAME_STATE03, blood.getDaughterHP());
		}
	}
	void CGameState02::OnShow() {
		map02.OnShow();
		daughter.OnShow();
		blood.OnShow(false);
		transition.OnShow();
		if (!intro && !outro) {
			dialog.OnShow();
		}
	}
}
```

==========
CGameState02.h
==========

```
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#include "CGameMap02.h"
#include "CDaughter.h"
#include "CGameData.h"
#include "CBlood.h"
#include "CTransition.h"
#include "CDialog.h"
#ifndef CGAMESTATE02_H
#define CGAMESTATE02_H
namespace game_framework {
	class CGameState02 : public CGameState {
	public:
		CGameState02(CGame *g);
		~CGameState02();
		void OnBeginState(int daughterHP);
		void OnInit();
		void OnKeyDown(UINT, UINT, UINT);
		void OnKeyUp(UINT, UINT, UINT);
	protected:
		void OnMove();
		void OnShow();
	private:
		CGameMap02 map02;
		CDaughter daughter;
		CGameData game_data;
		CBlood blood;
		CTransition transition;
		CDialog dialog;
		bool intro;
		bool outro;
	};
}
#endif
```

==========
CGameState03.cpp
==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CGameState03.h"
namespace game_framework {
	CGameState03::CGameState03(CGame *g) :
		CGameState(g), game_data(), daughter(&game_data), map03(&game_data, 1600, 600),
		intro(true), outro(false) {
	}
	CGameState03::~CGameState03() {
	}
	void CGameState03::OnBeginState(int daughterHP){
		CAudio::Instance()->Play(WIND_SOUND, true);
		blood.setDaughterHP(daughterHP);
	}
	void CGameState03::OnMove() {
		intro = game_data.getDaughterPos().first > 700;
		outro = game_data.getDaughterPos().first < 100;
		daughter.OnMove();
		if (outro) {
			transition.OnOutro(&daughter, FACE_LEFT);
			CAudio::Instance()->Play(IRON_DOOR, false);
			CAudio::Instance()->Stop(WIND_SOUND);
			if (transition.transit)
				GotoGameState(GAME_STATE04, blood.getDaughterHP());
```

```
		}
		else if (!outro && !intro) {
			game_data.checkMove(&map03, &daughter,nullptr);
		}
		transition.OnIntro(&game_data, &daughter, FACE_LEFT, 700);
	}
	void CGameState03::OnInit(){
		ShowInitProgress(40);
		map03.LoadBitmap(IDB_MAP03);
		daughter.LoadBitmap();
		blood.LoadBitmap();
		transition.LoadBitmap();
		dialog.LoadBitmap(3);
		game_data.setDaughterTopLeft(800, 300);
		game_data.setMapTopLeft(800, 0);
		CAudio::Instance()->Load(IRON_DOOR, "sounds\\iron_door.mp3");
		CAudio::Instance()->Load(WIND_SOUND, "sounds\\strong_wind_sound_effect.mp3");
		ShowInitProgress(60);
	}
	void CGameState03::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
		dialog.OnKeyDown(nChar);
		if (dialog.IsFinished())
			daughter.OnKeyDown(nChar);
		if (nChar == KEY_ESC)
			PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}
	void CGameState03::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
		if (dialog.IsFinished())
			daughter.OnKeyUP(nChar);
		switch (nChar) {
		case KEY_F2:
			GotoGameState(GAME_STATE04, blood.getDaughterHP());
		}
	}
	void CGameState03::OnShow() {
		map03.OnShow();
		daughter.OnShow();
		blood.OnShow(false);
		transition.OnShow();
		if (!intro && !outro) {
			dialog.OnShow();
		}
	}
}
==============================
CGameState03.h
==============================
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#include "CGameMap03.h"
#include "CDaughter.h"
#include "CGameData.h"
#include "CBlood.h"
#include "CTransition.h"
#include "CDialog.h"
#ifndef CGAMESTATE03_H
#define CGAMESTATE03_H
namespace game_framework {
	class CGameState03 : public CGameState {
	public:
		CGameState03(CGame *g);
		~CGameState03();
		void OnBeginState(int daughterHP);
		void OnInit();
		void OnKeyDown(UINT, UINT, UINT);
		void OnKeyUp(UINT, UINT, UINT);
	protected:
		void OnMove();
```

```
		void OnShow();
	private:
		CGameMap03 map03;
		CDaughter daughter;
		CGameData game_data;
		CBlood blood;
		CTransition transition;
		CDialog dialog;
		bool intro;
		bool outro;
	};
}
#endif
==============================
CGameState04.cpp
==============================
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CGameState04.h"
namespace game_framework {
	CGameState04::CGameState04(CGame *g) :
		CGameState(g), game_data(), daughter(&game_data), father(&game_data, &blood), map04(&game_data, 1600,
600),
		intro(true), outro(false) {
	}
	CGameState04::~CGameState04() {
	}
	void CGameState04::OnBeginState(int daughterHP) {
		CAudio::Instance()->Stop(WIND_SOUND);
		CAudio::Instance()->Play(WHISPER_SOUND, true);
		blood.setDaughterHP(daughterHP);
	}
	void CGameState04::OnMove() {
		intro = game_data.getDaughterPos().first < 0;
		outro = game_data.getDaughterPos().first > 640;
		range = game_data.getMapPos().first >= 300 && game_data.getMapPos().first < 800;
		daughter.OnMove();
		if (outro) {
			transition.OnOutro(&daughter, FACE_RIGHT);
			CAudio::Instance()->Play(ROOM_DOOR, false);
			CAudio::Instance()->Stop(WHISPER_SOUND);
			if (transition.transit) {
				CAudio::Instance()->Stop(DAUGHTER_FOOTSTEP);
				GotoGameState(GAME_STATE05, blood.getDaughterHP());
			}
		}
		else if (!outro && !intro) {
			game_data.checkMove(&map04, &daughter,nullptr);
		}
		if (range) {
			game_data.setMapMove(5, 0);
			game_data.setDaughterMove(-5, 0);
			game_data.setFatherMove(-5, 0);
		}
		transition.OnIntro(&game_data, &daughter, FACE_RIGHT, 0);
	}
	void CGameState04::OnInit(){
		ShowInitProgress(60);
		map04.LoadBitmap(IDB_MAP04);
		daughter.LoadBitmap();
		father.LoadBitmap();
		blood.LoadBitmap();
		transition.LoadBitmap();
		dialog.LoadBitmap(4);
		locked_door.LoadBitmap(IDB_DOOR_LOCK, RGB(0, 0, 255));
```

```
		locked_door.SetTopLeft(370, 0);
		game_data.setDaughterTopLeft(-100, 300);
		game_data.setFatherTopLeft(930, 270);
		game_data.setMapTopLeft(0, 0);
		CAudio::Instance()->Load(ROOM_DOOR, "sounds\\room_door.mp3");
		CAudio::Instance()->Load(WHISPER_SOUND, "sounds\\whisper_mixdown.mp3");
		ShowInitProgress(80);
	}
	void CGameState04::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
		if(game_data.getMapPos().first >= 799)
			dialog.OnKeyDown(nChar);
		if (game_data.getMapPos().first >= 300 && !dialog.IsFinished()) {
			daughter.OnKeyUP(KEY_RIGHT);
			daughter.outro = true;
		}
		else {
			daughter.outro = false;
			daughter.OnKeyDown(nChar);
		}
		if (nChar == KEY_ESC)
			PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}
	void CGameState04::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
		if (game_data.getMapPos().first >= 300 && !dialog.IsFinished()) {
			daughter.OnKeyUP(KEY_RIGHT);
			daughter.outro = true;
		}
		else {
			daughter.outro = false;
			daughter.OnKeyUP(nChar);
		}
		switch (nChar) {
		case KEY_F2:
			GotoGameState(GAME_STATE05, blood.getDaughterHP());
		}
	}
	void CGameState04::OnShow() {
		map04.OnShow();
		daughter.OnShow();
		blood.OnShow(false);
		transition.OnShow();
		if (!dialog.IsFinished()) {
			father.OnShow();
		}
		if (game_data.getMapPos().first >= 799) {
			dialog.OnShow();
			if (game_data.getDaughterPos().first >= 400 && game_data.getDaughterPos().first <= 500) {
				locked_door.ShowBitmap();
			}
		}
	}
}
==============================
CGameState04.h
==============================
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#include "CGameMap04.h"
#include "CDaughter.h"
#include "CFather.h"
#include "CGameData.h"
#include "CBlood.h"
#include "CTransition.h"
#include "CDialog.h"
#ifndef CGAMESTATE04_H
#define CGAMESTATE04_H
namespace game_framework {
	class CGameState04 : public CGameState {
```

```
	public:
		CGameState04(CGame *g);
		~CGameState04();
		void OnBeginState(int daughterHP);
		void OnInit();
		void OnKeyDown(UINT, UINT, UINT);
		void OnKeyUp(UINT, UINT, UINT);
	protected:
		void OnMove();
		void OnShow();
	private:
		CGameMap04 map04;
		CDaughter daughter;
		CFather father;
		CGameData game_data;
		CBlood blood;
		CTransition transition;
		CDialog dialog;
		CMovingBitmap locked_door;
		bool intro;
		bool outro;
		bool range;
	};
}
#endif
```

==========

CGameState05.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CGameState05.h"
namespace game_framework {
	CGameState05::CGameState05(CGame *g) :
		CGameState(g), game_data(), blood(), daughter(&game_data), map05(&game_data, 800, 600),
father(&game_data,&blood)
		,bossAI(&father, &game_data, &container,&blood){
		this->counter = 0;
		this->firstStart = false;
	}
	CGameState05::~CGameState05() {
	}
	void CGameState05::OnBeginState(int daughterHP) {
		CAudio::Instance()->Stop(WHISPER_SOUND);
		CAudio::Instance()->Play(BOSS_BGM, true);
		blood.setDaughterHP(daughterHP);
	}
	void CGameState05::OnMove() {
		if (!dialog.IsFinished() || blood.daughterIsDied() || blood.fatherIsDied()) {
			counter = 0;
			bossAI.setAILock(true);
			CAudio::Instance()->Pause();
		}
		else {
			bossAI.setAILock(false);
			CAudio::Instance()->Resume();
		}
		if (counter < 25 && blood.getFatherHP()>450 && !blood.fatherIsDied() ) {
			blood.setFatherHP(690-(10*counter));
		}
		else {
			counter = 25;
		}
		intro = game_data.getDaughterPos().first < 0;
		daughter.OnMove();
		container.OnMove();
```

```
	father.OnMove();
	bossAI.OnLoop();
	container.damageTake(&blood, &daughter, game_data.getDaughterPos());
	if (!intro)
		game_data.checkMove(&map05, &daughter,&father);
	if (dialog_fail.IsFinished() || dialog_success.IsFinished()) {
		GotoGameState(GAME_STATE_OVER, blood.getDaughterHP());
	}
	transition.OnIntro(&game_data, &daughter, FACE_RIGHT, 0);
	counter++;
}
void CGameState05::OnInit() {
	ShowInitProgress(80);
	map05.LoadBitmap(IDB_MAP05);
	daughter.LoadBitmap();
	blood.LoadBitmap();
	father.LoadBitmap();
	transition.LoadBitmap();
	dialog.LoadBitmap(5);
	dialog_fail.LoadBitmap(6);
	dialog_success.LoadBitmap(7);
	blood.setFatherHP(690);
	game_data.setDaughterTopLeft(-100, 300);
	game_data.setFatherTopLeft(700, 270);
	game_data.setMapTopLeft(0, 0);
	CAudio::Instance()->Load(BOSS_BGM, "sounds\\boss_bgm.mp3");
	ShowInitProgress(100);
}
void CGameState05::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags) {
	if (dialog.IsFinished() && !blood.daughterIsDied() && !blood.fatherIsDied()) {
		daughter.OnKeyDown(nChar);
	}
	dialog.OnKeyDown(nChar);
	if (blood.daughterIsDied()) {
		dialog_fail.OnKeyDown(nChar);
	}
	else if (blood.fatherIsDied()) {
		dialog_success.OnKeyDown(nChar);
	}
	if (nChar == KEY_ESC)
		PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
}
void CGameState05::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags) {
	if (dialog.IsFinished() && !blood.daughterIsDied() && !blood.fatherIsDied()) {
		daughter.OnKeyUP(nChar);
	}
	switch (nChar) {
	case KEY_F3:
		blood.setDaughterHP(102);
		break;
	case KEY_F4:
		blood.setDaughterHP(-141);
		break;
	case KEY_F5:
		blood.setFatherHP(560);
		break;
	case KEY_F6:
		blood.setFatherHP(700);
		break;
	}
}
void CGameState05::OnShow() {
	map05.OnShow();
	daughter.OnShow();
	container.OnShow();
	father.OnShow();
	if (!dialog.IsFinished()) {
		blood.OnShow(false);
	}
```

```
		else {
			blood.OnShow(true);
		}
		transition.OnShow();
		if (!intro) {
			dialog.OnShow();
		}
		if (blood.daughterIsDied()) {
			dialog_fail.OnShow();
		}
		else if(blood.fatherIsDied()) {
			dialog_success.OnShow();
		}
	}
}
```

==========

CGameState05.h

==========

```
#pragma once
#include "Definitions.h"
#include "CGameMap.h"
#include "CGameMap05.h"
#include "CDaughter.h"
#include "CFather.h"
#include "CBossAI.h"
#include "CGameData.h"
#include "CBlood.h"
#include "CTransition.h"
#include "CDialog.h"
#include "CDamageContainer.h"
#include "CDamagePen.h"
#ifndef CGAMESTATE05_H
#define CGAMESTATE05_H
namespace game_framework {
	class CGameState05 : public CGameState {
	public:
		CGameState05(CGame *g);
		~CGameState05();
		void OnBeginState(int daughterHP);
		void OnInit();
		void OnKeyDown(UINT, UINT, UINT);
		void OnKeyUp(UINT, UINT, UINT);
	protected:
		void OnMove();
		void OnShow();
	private:
		CGameMap05 map05;
		CFather father;
		CBossAI bossAI;
		CDaughter daughter;
		CGameData game_data;
		CBlood blood;
		CTransition transition;
		CDialog dialog;
		CDialog dialog_fail;
		CDialog dialog_success;
		CDamageContainer container;
		int counter;
		bool intro;
		bool firstStart;
	};
}
#endif
```

==========

CTransition.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
```

```
#include <ddraw.h>
#include "gamelib.h"
#include "audio.h"
#include "CTransition.h"
#include "CGameData.h"
#include "CDaughter.h"
namespace game_framework {
	CTransition::CTransition() {}
	CTransition::~CTransition(){}
	void CTransition::LoadBitmap() {
		outro_up.LoadBitmap(IDB_OUTRO_UP);
		outro_down.LoadBitmap(IDB_OUTRO_DOWN);
	}
	void CTransition::OnIntro(CGameData *game_data, CDaughter *daughter, int faceSide, int stop) {
		if (faceSide == FACE_RIGHT) {
			if (game_data->getDaughterPos().first < stop) {
				daughter->OnKeyDown(KEY_RIGHT);		//進場走進來(右)
				daughter->outro = true;
			}
			if (game_data->getDaughterPos().first == stop) {
				daughter->outro = false;
				daughter->OnKeyUP(KEY_RIGHT);		//進場停止(右)
			}
		}
		else if (faceSide == FACE_LEFT) {
			if (game_data->getDaughterPos().first > stop) {
				daughter->OnKeyDown(KEY_LEFT);		//進場走進來(左)
				daughter->outro = true;
			}
			if (game_data->getDaughterPos().first == stop) {
				daughter->outro = false;
				daughter->OnKeyUP(KEY_LEFT);		//進場停止(左)
			}
		}
	}
	void CTransition::OnOutro(CDaughter *daughter, int faceSide) {
		if (y < 100) {
			y += 5;				//出場橫幅動畫
		}
		if (y == 100) {
			transit = true;		//接換遊戲章節
		}
		if (faceSide == FACE_RIGHT) {
			daughter->OnKeyDown(KEY_RIGHT);		//出場走出去(右)
			daughter->outro = true;
		}
		else if (faceSide == FACE_LEFT) {
			daughter->OnKeyDown(KEY_LEFT);		//出場走出去(左)
			daughter->outro = true;
		}
	}
	void CTransition::OnShow() {
		outro_up.SetTopLeft(0, -100 + y);
		outro_down.SetTopLeft(0, 600 - y);
		outro_up.ShowBitmap();
		outro_down.ShowBitmap();
	}
}
==============================
CTransition.h
==============================
#pragma once
#include "Definitions.h"
#ifndef CTRANSITION_H
#define CTRANSITION_H
namespace game_framework {
	class CTransition {
	public:
		CTransition();
```

```
		~CTransition();
		void LoadBitmap();
		void OnIntro(CGameData *, CDaughter *, int, int);		//進場動畫
		void OnOutro(CDaughter *, int);						//出場動畫
		void OnShow();
		bool transit = false;
	private:
		int y = 0;								//下降幅度
		CMovingBitmap outro_up;                                 //出場橫幅
		CMovingBitmap outro_down;                               //出場橫幅
	};
}
#endif
```

==========

Definitions.h

==========

```
#pragma once
#ifndef DEFINITIONS_H
#define DEFINITIONS_H
namespace game_framework {
	class CGameMap;
	class CGameData;
	class CDaughter;
	class CFather;
	class CDamageContainer;
	class CBlood;
	class CTransition;
	class CDialog;
}
#endif
```

==========

mygame.cpp

==========

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include <windows.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"
namespace game_framework {
CGameStateInit::CGameStateInit(CGame *g)
: CGameState(g){
}
void CGameStateInit::OnInit(){
	ShowInitProgress(0);
	menu.LoadBitmap(IDB_MENU);
	func.LoadBitmap(IDB_FUNC);
	dot.LoadBitmap(IDB_DOT);
	secret.LoadBitmap(IDB_SECRET);
	secret.SetTopLeft(0, 390);
	dot.SetTopLeft(270, 400);
	menu.SetTopLeft(0, 0);
	func.SetTopLeft(0, 0);
	CAudio::Instance()->Load(AUDIO_DEVOTION, "sounds\\devotion.mp3");
	CAudio::Instance()->Play(AUDIO_DEVOTION, true);
	CAudio::Instance()->Load(DAUGHTER_FOOTSTEP, "sounds\\footstep.wav");
	CAudio::Instance()->Load(DAUGHTER_ATTACK, "sounds\\attack.mp3");
	CAudio::Instance()->Load(DAUGHTER_SLIDE, "sounds\\slide.wav");
	CAudio::Instance()->Load(DAUGHTER_HURT, "sounds\\hurt.mp3");
	CAudio::Instance()->Load(FATHER_HURT, "sounds\\hurt_father.mp3");
	CAudio::Instance()->Load(EXPLO_SOUND, "sounds\\explosion.mp3");
	CAudio::Instance()->Load(SEA_WAVE, "sounds\\sea_wave.mp3");
	CAudio::Instance()->Load(G_HIT, "sounds\\ground_hit.mp3");
}
void CGameStateInit::OnBeginState(){
}
void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags){
```

```
	nC = nChar;
	if (dot.Top() <= 450 && nChar == KEY_DOWN) {
		dot.SetTopLeft(270, dot.Top() + 50);
	}
	if(dot.Top() >= 450 && nChar == KEY_UP) {
		dot.SetTopLeft(270, dot.Top() - 50);
	}
	if (dot.Top() == 400 && nChar == KEY_ENTER) {
		CAudio::Instance()->Stop(AUDIO_DEVOTION);
		GotoGameState(GAME_STATE01, 102);
	}
	if (dot.Top() == 500 && nChar == KEY_ENTER) {
		PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}
	if (nChar == KEY_ESC) {
		PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}
}
void CGameStateInit::OnShow(){
	menu.ShowBitmap();
	dot.ShowBitmap();
	if (dot.Top() == 450 && nC == KEY_ENTER) {
		func.ShowBitmap();
	}
	if (nC == KEY_LEFT) {
		secret.ShowBitmap();
	}
}
void CGameStateInit::OnMove() {
}
CGameStateOver::CGameStateOver(CGame *g)
: CGameState(g)
{
	game = g;
}
void CGameStateOver::OnMove(){
	counter--;
	if (counter < 0) {
		PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE, 0, 0);
	}
}
void CGameStateOver::OnBeginState(int daughterHP)
{
	counter = 30 * 3;
}
void CGameStateOver::OnInit()
{
	ShowInitProgress(66);
	over.LoadBitmap(IDB_OVER);
	over.SetTopLeft(0, 0);
	Sleep(300);
	ShowInitProgress(100);
}
void CGameStateOver::OnShow()
{
	over.ShowBitmap();
}
}
==============================
mygame.h
==============================
#include "Definitions.h"
#include "CGameMap.h"
#include "CGameMap02.h"
#include "CGameMap01.h"
#include "CDaughter.h"
#include "CGameData.h"
#include "CGameState01.h"
#include "CGameState02.h"
```

```
#include "CGameState03.h"
#include "CGameState04.h"
#include "CGameState05.h"
namespace game_framework {
	/////////////////////////////////////////////////////////////////////////////
	// 這個 class 為遊戲的遊戲開頭畫面物件
	// 每個 Member function 的 Implementation 都要弄懂
	/////////////////////////////////////////////////////////////////////////////
	class CGameStateInit : public CGameState {
	public:
		CGameStateInit(CGame *g);
		void OnInit();  // 遊戲的初值及圖形設定
		void OnBeginState();  // 設定每次重玩所需的變數
		void OnKeyUp(UINT, UINT, UINT);  // 處理鍵盤 Up 的動作
	protected:
		void OnShow();  // 顯示這個狀態的遊戲畫面
		void OnMove();
	private:
		UINT nC;
		CPoint p;
		CMovingBitmap menu;
		CMovingBitmap func;
		CMovingBitmap dot;
		CMovingBitmap secret;
	};
	/////////////////////////////////////////////////////////////////////////////
	// 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(int);  // 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
		CMovingBitmap over;
	};
}
```