國立臺北科技大學

2021 Spring 資工系物件導向程式實習

期末報告

MapleStory Old Flash Game

第 19 組

107820014 謝昀義

107820018 陳彥任

目錄

一、 簡介

1. 動機

我們選擇 MapleStory Old Flash Game 的主要原因是楓之谷是我們都玩過的遊戲，而這遊戲難度適中，且這款用 Flash 做的遊戲已無法在網頁上玩到，如果要仿造一款遊戲，那復刻這類懷舊的網頁遊戲應該是再適合不過了。

2. 分工

大致上分工是謝昀義負責程式邏輯及設計，陳彥任負責素材收集和動畫處理。

表 1.2 是較詳細的分工細目，比重只考慮實作程式的部分，討論和提供意見不算在內。

工作	謝昀義	陳彥任
製作圖檔	0%	100%
音效音檔	0%	100%
玩家物件(含動畫)	90%	10%
地形物件	100%	0%
怪物物件(含動畫)	50%	50%
關卡控制(邏輯)	100%	0%
關卡 1 地形生成	100%	0%
關卡 2 地形生成	100%	0%
關卡 3 地形生成	100%	0%
關卡 1 怪物生成	80%	20%
關卡 2 怪物生成	80%	20%
關卡 3 怪物生成	0%	100%
玩家與怪物互動	100%	0%
玩家與地形互動	100%	0%
主選單	0%	100%
遊戲機制	100%	0%
結束畫面	50%	50%

表 1.2 分工細目

二、 遊戲介紹

1. 遊戲說明

我們仿造的 MapleStory Old Flash Game 是非官方的同人創作，所以原本遊戲內容也很單純，遊戲總共有三個關卡，每個關卡的地形和怪物都不太一樣。

遊戲方式：

玩家可透過上下左右、Space、Z 鍵操控遊戲角色進行移動、攀爬、跳躍和攻擊，所有的怪物都不會主動攻擊，但是角色接觸到怪物時會受傷。角色有三條命，每次受傷會損失一條命，三條命耗盡角色會死亡，且沒有回復或補充生命的手段。

遊戲規則：

打倒地圖上所有的怪物就可以進入下一關，玩家要在三條命耗盡之前完成全部的關卡才算過關，若在任一關卡中死亡則算闖關失敗。

特殊功能：

按下 Ctrl + F 鍵可切換成全螢幕，再次按下可取消全螢幕。

密技 1：

遊戲中按下 Shift + Z 鍵可將地圖上剩下的所有怪物全部打倒，以利進入下一關。

密技 2：

遊戲中按 S 鍵可使角色進入無敵狀態，無敵狀態中無視碰撞，角色不會受傷，再次按下取消無敵狀態。

2. 遊戲圖形

圖 2.2.1 第一關畫面

圖 2.2.2 第二關畫面

圖 2.2.3 第三關畫面

圖 2.2.4 過關畫面

圖 2.2.5 失敗畫面

圖 2.2.6 主選單畫面

圖 2.2.7 icon 圖示

3. 遊戲音效

音效	說明
dead.wav	怪物死亡時播放的音效
hit.wav	角色攻擊命中怪物時撥放的音效
hurt.wav	角色受傷時播放的音效

表 2.3 音效

三、 程式設計

1. 程式架構

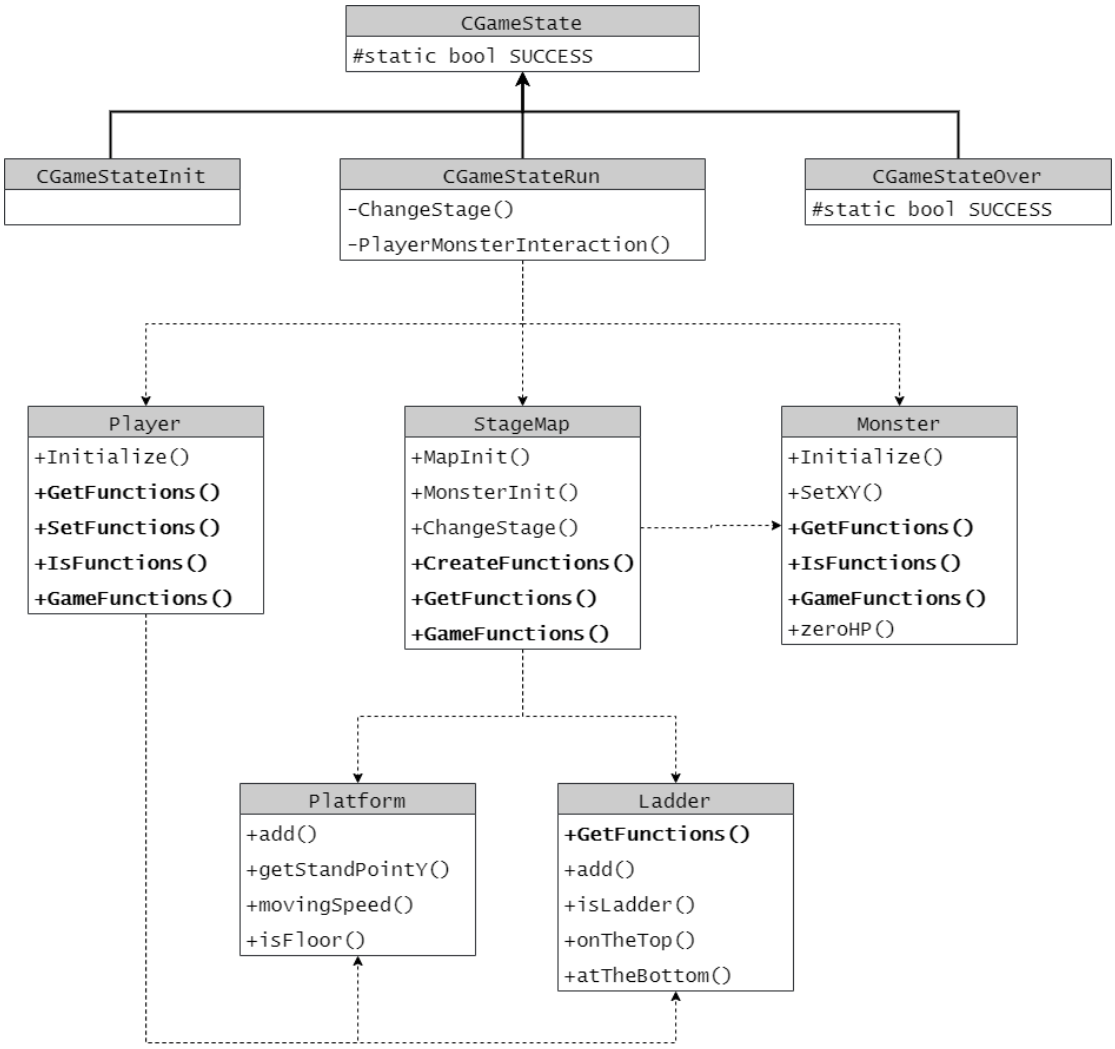

圖 3.1 class diagram

箭頭說明：

粗箭頭是繼承關係(inherit)，虛線箭頭是依賴關係(dependency)。

架構說明：

CGameState 及繼承它的 3 個 class 是教授提供的遊戲框架，我們自己新增的為圖 3.1 下方的 5 個 class，class 之間的關係如圖所示，Platform 和 Ladder 屬於地形物件。StageMap 負責產生每個關卡的地形與怪物，StageMap 會在 CGameStateRun 傳給 Player 當前的地形物件，Player 依賴地形物件才能正常移動。

CGameStateRun 依賴 Player、StageMap、Monster 3 個 class 使遊戲正常運作，Player 和 Monster 的互動是在 CGameStateRun 做判斷，所以 Player 不會直接認識 Monster。關卡的切換是在 CGameStateRun 做判定，再向 StageMap 要下一關的物件。

2. 程式類別

類別名稱	.h 檔行數	.cpp 檔行數	說明
Player	69	451	角色的動作和移動
Platform	21	82	地形障礙物(平台、斜坡)
Ladder	21	63	可攀爬物(繩梯、藤蔓)
Monster	37	317	怪物動作和移動
StageMap	54	237	關卡物件生成和切換
總行數	202	1150	

表 3.2

3. 程式技術

我們沒有使用任何資料結構，遊戲的運作主要都是靠物件狀態去判斷，演算的部分比較值得一提的是地面的判定，我們在地面上劃出一個矩形的區域，角色的足點即將要進入這區域時，就會判定角色在地上，也就是說，除了足點的 XY 座標還加入一個參數 V_Y，V_Y 是用來預測角色下一秒是否進入地面，此做法是為了避免下落速度過快時判斷不到地面，或是避免落地時角色插進地面。

圖 3.3.1 平台實際區域

另外斜坡的設計也花費我們不少時間，最後成功使角色在斜坡上能正常活動有兩個關鍵。一開始很直覺地想讓角色遇到斜坡時斜對角地移動，但這地圖的像素不夠大，無法用向左幾格向上幾格的方式準確移動到另一端，所以改成角色在斜坡上仍是左右移動，而斜坡要根據角色足點的 X 座標推算出斜邊上的 Y 座標並傳給角色，讓角色修正自己的 Y 座標，而計算方式是用平行線截比例線段，此為第一個關鍵。

完成上述功能後，角色上坡就沒有什麼問題，但是下坡時會以在空中的動畫下坡，原因是角色做左右移動，下坡時會先走到空中再落下，因此我們在斜坡上的地面判定放寬幾個像素，才能讓角色正常上下坡，此為第二個關鍵，而且此判定方式同時也解決了斜坡兩端容易踩空的 bug。

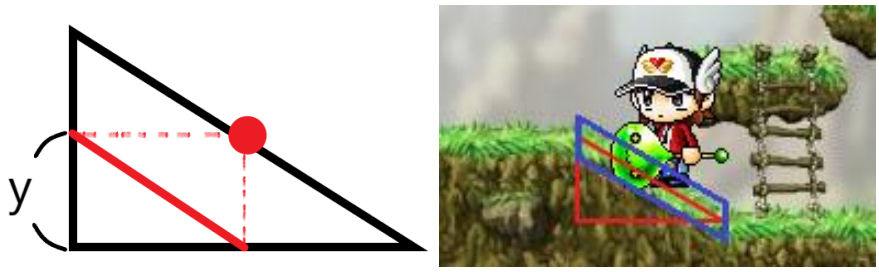

圖 3.3.2 (左)求 Y 座標方法，(右)斜坡實際區域(紅)與判定範圍(藍)

四、 結語

1. 問題及解決方法：

一開始不清楚要如何使角色正常的移動，我們找助教討論，助教提示我們要利用狀態控制角色才知道如何修正，順利開發下去。

有時候從 github 上拉組員更新過的專案會無法執行，我們分析了錯誤訊息發現，圖檔匯入專案裡的時候，要確認圖片來源是相對路徑，不然在不同裝置會找不到檔案。

角色的攻擊動畫原本一直有小 bug，嘗試了各種方法都沒修好，是直到最後幾週在處理過大量的 OnMove 和 OnShow 的運算，回頭來看這 bug，才發現我們在 Player 的 OnMove 中一開始就一直呼叫攻擊動畫的 OnMove，所以每次攻擊的動畫才不是從頭播放。

2. 時間表：

週次	謝昀義	陳彥任	說明
1	0	0	Git tutorial
2	0	0	Git tutorial
3	0	0	Git tutorial
4	3	3	第一關素材+建地形
5	5	5	建地形
6	3	0	建地形
7	11	10	建地形+角色移動
8	3	5	斜坡地形+角色移動
9	6	4	建怪物class+角色功能+互動
10	7	3	refactor
11	7	6	製作關卡
12	6	0	製作關卡+新增怪物
13	2	6	製作關卡
14	4	8	優化角色
15	5	4	切換關卡
16	3	4	製作關卡+切換關卡
17	10	9	主選單+結束畫面
總時數	75	67	

表4.2

3. 貢獻比例：

謝昀羲 60%

陳彥任 40%

4. 自我檢核表：

	項目	完成否	無法完成原因
1	解決 Memory leak	已完成	
2	自訂遊戲 Icon	已完成	
3	有 About 畫面	已完成	
4	初始化面說明按鍵及滑鼠之用法與密技	已完成	
5	上傳 setup 檔	已完成	
6	Setup 檔可正確執行	已完成	
7	報告字型、點數、對齊、行距、頁碼等格式正確	已完成	
8	全螢幕啟動-加分項目	已完成	

表 4.4

5. 收穫：

謝昀羲：

遊戲是個很大的專案，雖然有教授準備好的框架，簡化不少工作，但在設計 class 和 function 的過程中，我確實體會到設計程式的一個原則：SRP。很多時候為求方便或是沒想太多就給一個 class 或一個 function 做太多事，除了程式碼會太長之外，可能過一個禮拜回來看，發現看不懂自己在寫什麼，程式碼的可讀性不高，要維護也不容易。

陳彥任：

慢慢從小部分錯誤去修改 一次想全修正 很容易漏掉或寫錯

較複雜交錯的邏輯部分 還是先畫圖或筆記會比較不容易寫錯

srand(number*(unsigned int)time(NULL));

原本同一關的怪物都一起動 time 隨機是看時間 故想要將怪物個別隨機移動必須乘上一個常數

6. 心得、感想：

謝昀羲：

距離上次寫C++是大一的時候，太久沒碰都忘了程式該怎麼寫，在一點一點的摸索和回憶後，總算是完成這款遊戲，雖然程式有很多地方需要再加強、修改，整個架構也不夠堅固，但就以還原度來看，至少有八九成以上，對此我感到非常欣慰。這學期因為其他課也很花時間，以致於我無法全力投入在這專案，這是我覺得有點遺憾的地方，如果以後有時間，我應該會重新整理這專案。

陳彥任：

剛開始拿到這個框架，就算有大概的範例，還是沒辦法完全了解不同 Function 的詳細功能及其限制，但經過詢問助教以及組員的講解，也漸漸了解及其他操作，雖然實際有寫出的程式碼不多，但也至少嘗試了不少次，也比修課之前更容易看懂這種相較比較多行的程式碼。

7. 對本課程的建議：

無

附錄

mygame.h

```
#include "Player.h"
#include "Monster.h"
#include "StageMap.h"
namespace game_framework {
	/////////////////////////////////////////////////////////////////////////////
	// Constants
	/////////////////////////////////////////////////////////////////////////////
	enum AUDIO_ID {				// 定義各種音效的編號
		AUDIO_HIT,				// 0
		AUDIO_DEAD,				// 1
		AUDIO_HURT				// 2
	};
	/////////////////////////////////////////////////////////////////////////////
	// 這個class為遊戲的遊戲開頭畫面物件
	// 每個Member function的Implementation都要弄懂
	/////////////////////////////////////////////////////////////////////////////
	class CGameStateInit : public CGameState {
	public:
		CGameStateInit(CGame *g);
		void OnInit();  								// 遊戲的初值及圖形設定
		void OnBeginState();							// 設定每次重玩所需的變數
		void OnKeyUp(UINT, UINT, UINT);					// 處理鍵盤Up的動作
		void OnLButtonUp(UINT nFlags, CPoint point);
		void OnLButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
		void OnMouseMove(UINT nFlags, CPoint point);
	protected:
		void OnShow();									// 顯示這個狀態的遊戲畫面
	private:
		CMovingBitmap menu_bg, menu_bg1, menu_bg2, start, start2, about1, about2;
		bool isLButtonUp;
		bool isOnStartButton, isOnAboutButton, isOnBackButton = false;
		bool isAboutOpen = false;
	};
	/////////////////////////////////////////////////////////////////////////////
	// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
	// 每個Member function的Implementation都要弄懂
	/////////////////////////////////////////////////////////////////////////////
	class CGameStateRun : public CGameState {
	public:
		CGameStateRun(CGame *g);
		~CGameStateRun();
		void OnBeginState();							// 設定每次重玩所需的變數
		void OnInit();  								// 遊戲的初值及圖形設定
		void OnKeyDown(UINT, UINT, UINT);
		void OnKeyUp(UINT, UINT, UINT);
		void OnLButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
		void OnLButtonUp(UINT nFlags, CPoint point);	// 處理滑鼠的動作
		void OnMouseMove(UINT nFlags, CPoint point);	// 處理滑鼠的動作
		void OnRButtonDown(UINT nFlags, CPoint point);	// 處理滑鼠的動作
		void OnRButtonUp(UINT nFlags, CPoint point);	// 處理滑鼠的動作
	protected:
		void OnMove();									// 移動遊戲元素
		void OnShow();									// 顯示這個狀態的遊戲畫面
	private:
		void CheckStage();
		void PlayerMonsterInteraction(Player *player, vector<Monster> *monsters);
		Player				player;
		StageMap			map;
		vector<Monster> *monsters;
		bool monsterIsAllDead;
		int delayCounter;
		int trick1;		// shift + Z
		bool trick2;	// S
```

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

mygame.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "mygame.h"
namespace game_framework {
/////////////////////////////////////////////////////////////////////////////
// 這個class為遊戲的遊戲開頭畫面物件
/////////////////////////////////////////////////////////////////////////////
CGameStateInit::CGameStateInit(CGame *g)
: CGameState(g)
{
}
void CGameStateInit::OnInit()
{
	//
	// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人
	//       等的不耐煩，遊戲會出現「Loading ... 」，顯示Loading的進度。
	//
	ShowInitProgress(0);  // 一開始的loading進度為0%
	//
	// 開始載入資料
	//
	menu_bg.LoadBitmap(menu_background);
	menu_bg1.LoadBitmap(menu_background1);
	menu_bg2.LoadBitmap(menu_background2);
	about1.LoadBitmap(about_window1, 0x00FF00FF);
	about2.LoadBitmap(about_window2, 0x00FF00FF);
	start.LoadBitmap(start_button, 0x00FF00FF);
	start2.LoadBitmap(start_button2, 0x00FF00FF);
	ShowInitProgress(60);
	//
	// 此OnInit動作會接到CGameStaterRun::OnInit()，所以進度還沒到100%
	//
}
void CGameStateInit::OnBeginState()
{
}
void CGameStateInit::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	const char KEY_ESC = 27;
	if (nChar == KEY_ESC)                                       // Demo 關閉遊戲的方法
		PostMessage(AfxGetMainWnd()->m_hWnd, WM_CLOSE,0,0);   // 關閉遊戲
}
void CGameStateInit::OnLButtonUp(UINT nFlags, CPoint point)
```

```
{
	isLButtonUp = true;
	if (isOnStartButton)
		GotoGameState(GAME_STATE_RUN);
	if (!isAboutOpen) {
		if(isOnAboutButton)
			isAboutOpen = true;
	}
	else {
		if (isOnBackButton)
			isAboutOpen = false;
	}
}
void CGameStateInit::OnLButtonDown(UINT nFlags, CPoint point)
{
	isLButtonUp = false;
}
void CGameStateInit::OnMouseMove(UINT nFlags, CPoint point)
{
	if (!isAboutOpen) {
		if (point.x > 420 && point.x < 500 && point.y > 415 && point.y < 435)
			isOnStartButton = true;
		else
			isOnStartButton = false;
		if (point.x > 105 && point.x < 160 && point.y > 420 && point.y < 440)
			isOnAboutButton = true;
		else
			isOnAboutButton = false;
	}
	if (isAboutOpen){
		if (point.x > 275 && point.x < 335 && point.y > 255 && point.y < 280)
			isOnBackButton = true;
		else
			isOnBackButton = false;
	}
}
void CGameStateInit::OnShow()
{
	menu_bg1.SetTopLeft(40, 0);
	menu_bg2.SetTopLeft(40, 0);
	about1.SetTopLeft(155, 125);
	about2.SetTopLeft(155, 125);
	start.SetTopLeft(424, 419);
	start2.SetTopLeft(424, 419);
	if (isAboutOpen) {
		menu_bg1.ShowBitmap();
		about1.ShowBitmap();
		if (isOnBackButton) {
			menu_bg1.ShowBitmap();
			about2.ShowBitmap();
		}
	}
	else if(isOnAboutButton) {
			menu_bg2.ShowBitmap();
	}
	else
			menu_bg1.ShowBitmap();
	if (isOnStartButton)
		start.ShowBitmap();
	else
		start2.ShowBitmap();
}
/////////////////////////////////////////////////////////////////////////////
// 這個class為遊戲的結束狀態(Game Over)
/////////////////////////////////////////////////////////////////////////////
CGameStateOver::CGameStateOver(CGame *g)
: CGameState(g)
{
}
```

```
void CGameStateOver::OnMove()
{
	counter--;
	if (counter < 0)
		GotoGameState(GAME_STATE_INIT);
}
void CGameStateOver::OnBeginState()
{
	counter = 30 * 3; // 5 seconds
}
void CGameStateOver::OnInit()
{
	//
	// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人
	//	等的不耐煩，遊戲會出現「Loading ... 」，顯示Loading的進度。
	//
	ShowInitProgress(66);	// 接個前一個狀態的進度，此處進度視為66%
	//
	// 開始載入資料
	success_bg.LoadBitmap(success_background);
	fail_bg.LoadBitmap(gameover_background);
	//
	// 最終進度為100%
	//
	ShowInitProgress(100);
}
void CGameStateOver::OnShow()
{
	if (SUCCESS) {
		success_bg.SetTopLeft(40, 0);
		success_bg.ShowBitmap();
	}
	else {
		fail_bg.SetTopLeft(40, 0);
		fail_bg.ShowBitmap();
	}
}
/////////////////////////////////////////////////////////////////////////////
// 這個class為遊戲的遊戲執行物件，主要的遊戲程式都在這裡
/////////////////////////////////////////////////////////////////////////////
CGameStateRun::CGameStateRun(CGame *g)
: CGameState(g)
{
}
CGameStateRun::~CGameStateRun()
{
}
void CGameStateRun::OnBeginState()
{
	const int BACKGROUND_X = 40;
	const int ANIMATION_SPEED = 15;
	map.ChangeStage(1);
	monsters = map.GetMonsters();
	player.SetMap(map.GetPlatform(), map.GetLadder());
	player.Initialize();
	map.MonsterInit();
	monsterIsAllDead = false;
	delayCounter = 30;
	trick1 = 0;
	trick2 = false;
}
void CGameStateRun::OnMove()					// 移動遊戲元素
{
	CheckStage();
	if (map.GetStage() == 3 && monsterIsAllDead) {
		delayCounter--;
		if (delayCounter < 0) {
			SUCCESS = true;
			GotoGameState(GAME_STATE_OVER);
```

```
			}
		}
		else if (!player.IsAlive()) {
			delayCounter--;
			if (delayCounter < 0) {
				SUCCESS = false;
				GotoGameState(GAME_STATE_OVER);
			}
		}
		map.OnMove();
		player.OnMove();
		for (vector<Monster>::iterator m = monsters->begin(); m != monsters->end(); m++) {
			m->OnMove();
		}
		PlayerMonsterInteraction(&player, monsters);
		if (trick1 == 2) {
			for (size_t i = 0; i < monsters->size(); i++) {
				monsters->at(i).zeroHP();
			}
			CAudio::Instance()->Play(AUDIO_DEAD);
			trick1 = 0;
		}
}
void CGameStateRun::OnInit()  				// 遊戲的初值及圖形設定
{
		//
		// 當圖很多時，OnInit載入所有的圖要花很多時間。為避免玩遊戲的人
		//       等的不耐煩，遊戲會出現「Loading ...」，顯示Loading的進度。
		//
		ShowInitProgress(33);	// 接個前一個狀態的進度，此處進度視為33%
		//
		// 開始載入資料
		//
		player.LoadBitmap();
		map.MapInit();
		map.LoadBitmap();
		map.LoadMonsterBitmap();
		player.SetMap(map.GetPlatform(), map.GetLadder());
		monsters = map.GetMonsters();
		//
		// 完成部分Loading動作，提高進度
		//
		ShowInitProgress(50);
		//
		// 繼續載入其他資料
		//
		CAudio::Instance()->Load(AUDIO_HIT,    "sounds\\hit.wav");
		CAudio::Instance()->Load(AUDIO_DEAD, "sounds\\dead.wav");
		CAudio::Instance()->Load(AUDIO_HURT, "sounds\\hurt.wav");
		//
		// 此OnInit動作會接到CGameStaterOver::OnInit()，所以進度還沒到100%
		//
}
void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)
{
		const char KEY_LEFT  = 0x25; // keyboard左箭頭
		const char KEY_UP    = 0x26; // keyboard上箭頭
		const char KEY_RIGHT = 0x27; // keyboard右箭頭
		const char KEY_DOWN  = 0x28; // keyboard下箭頭
		const char KEY_SPACE = 0x20;
		const char KEY_Z     = 0x5A;
		const char KEY_SHIFT = 0x10;
		const char KEY_S     = 0x53;

		if (nChar == KEY_LEFT) {
			player.SetMovingLeft(true);
		}
		if (nChar == KEY_RIGHT) {
			player.SetMovingRight(true);
```

```
	}
	if (nChar == KEY_UP)
		player.SetMovingUp(true);
	if (nChar == KEY_DOWN)
		player.SetMovingDown(true);
	if (nChar == KEY_SPACE) {
		if(!player.IsInTheAir())
			player.SetJumping(true);
	}
	if (nChar == KEY_Z) {
		player.SetAttackKey(true);
		player.SetAttacking(true);
	}
	if (nChar == KEY_SHIFT) {
		trick1++;
	}
	if (trick1 == 1) {
		if (nChar == KEY_Z) {
			trick1++;
		}
	}
	if (nChar == KEY_S) {
		trick2 = !trick2;
	}
}
void CGameStateRun::OnKeyUp(UINT nChar, UINT nRepCnt, UINT nFlags)
{
	const char KEY_LEFT  = 0x25; // keyboard左箭頭
	const char KEY_UP    = 0x26; // keyboard上箭頭
	const char KEY_RIGHT = 0x27; // keyboard右箭頭
	const char KEY_DOWN  = 0x28; // keyboard下箭頭
	const char KEY_SPACE = 0x20;
	const char KEY_Z     = 0x5A;
	const char KEY_SHIFT = 0x10;
	if (nChar == KEY_LEFT) {
		player.SetMovingLeft(false);
	}
	if (nChar == KEY_RIGHT) {
		player.SetMovingRight(false);
	}
	if (nChar == KEY_UP)
		player.SetMovingUp(false);
	if (nChar == KEY_DOWN)
		player.SetMovingDown(false);
	if (nChar == KEY_SPACE)
		player.SetJumping(false);
	if (nChar == KEY_Z) {
		player.SetAttackKey(false);
		if (trick1 > 0)
			trick1--;
	}
	if (nChar == KEY_SHIFT) {
		if(trick1 > 0)
			trick1--;
	}
}
void CGameStateRun::OnLButtonDown(UINT nFlags, CPoint point)  // 處理滑鼠的動作
{
}
void CGameStateRun::OnLButtonUp(UINT nFlags, CPoint point) // 處理滑鼠的動作
{
}
void CGameStateRun::OnMouseMove(UINT nFlags, CPoint point)        // 處理滑鼠的動作
{
	// 沒事。如果需要處理滑鼠移動的話，寫code在這裡
}
void CGameStateRun::OnRButtonDown(UINT nFlags, CPoint point)  // 處理滑鼠的動作
{
}
```

```
void CGameStateRun::OnRButtonUp(UINT nFlags, CPoint point) // 處理滑鼠的動作
{
}
void CGameStateRun::OnShow()
{
	//
	//  注意：Show裡面千萬不要移動任何物件的座標，移動座標的工作應由Move做才對，
	//        否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語
	//        說，Move負責MVC中的Model，Show負責View，而View不應更動Model。
	//
	map.OnShow();
	player.OnShow();
	for (size_t i = 0; i < monsters->size(); i++) {
		monsters->at(i).OnShow();
	}
}
void CGameStateRun::CheckStage()
{
	for (auto m = monsters->begin(); m != monsters->end(); m++) {
		if (!m->IsDisappear()) {
			monsterIsAllDead = false;
			break;
		}
		else {
			monsterIsAllDead = true;
		}
	}
	if (monsterIsAllDead && map.GetStage() < 3) {
		map.ChangeStage(map.GetStage()+1);
		monsters = map.GetMonsters();
		player.SetMap(map.GetPlatform(), map.GetLadder());
		player.SetXY(504, 380);
		monsterIsAllDead = false;
	}
}
void CGameStateRun::PlayerMonsterInteraction(Player * player, vector<Monster>* monsters)
{
	tuple<int, int, int, int> playerAR = player->GetAttackRange();
	for (auto m = monsters->begin(); m != monsters->end(); m++) {
		if (!player->IsInSuperState() && !trick2) {
			if (m->isCollision(player->GetX1(), player->GetY1(), player->GetX2(), player->GetY2())) {
				player->SetGetHurt(true);
				CAudio::Instance()->Play(AUDIO_HURT);
			}
		}
		if (player->IsAttacking()) {
			if (m->isCollision(get<0>(playerAR), get<1>(playerAR), get<2>(playerAR), get<3>(playerAR))) {
				m->GetHurt(1);
				CAudio::Instance()->Play(AUDIO_HIT);
				if (!m->IsAlive() && !m->IsDisappear()) {
					CAudio::Instance()->Play(AUDIO_DEAD);
				}
			}
		}
	}
}
}
```

Player.h

```
#include "Platform.h"
#include "Ladder.h"
namespace game_framework {
	class Player
	{
	public:
		Player();
		void Initialize();                              // 初始化
		/*************get function************************/
```

```
		int	GetX1();					// 玩家左上角 x 座標
		int	GetY1();					// 玩家左上角 y 座標
		int	GetX2();					// 玩家右下角 x 座標
		int	GetY2();					// 玩家右下角 y 座標
		int	GetMidX();					// 玩家中心 x 座標
		int	GetMidY();					// 玩家中心 y 座標
		tuple<int, int, int, int> GetAttackRange();
		/*************game function************************/
		void LoadBitmap();
		void OnMove();
		void OnShow();
		/*************set function************************/
		void SetMovingDown(bool flag);				// 設定是否正在往下移動
		void SetMovingLeft(bool flag);				// 設定是否正在往左移動
		void SetMovingRight(bool flag);				// 設定是否正在往右移動
		void SetMovingUp(bool flag);				// 設定是否正在往上移動
		void SetFacingLeft(bool flag);
		void SetJumping(bool flag);
		void SetAttackKey(bool flag);
		void SetAttacking(bool flag);
		void SetGetHurt(bool flag);
		void SetXY(int nx, int ny);				// 設定左上角座標
		void SetIsClimbing(bool flag);
		void SetMap(Platform *plat, Ladder *lad);
		/*************is function************************/
		bool IsInTheAir();
		bool IsOnTheGround();
		bool IsAttacking();
		bool IsInSuperState();
		bool IsAlive();
	protected:
		CAnimation idleLeft, idleRight, lieLeft, lieRight, jumpLeft, jumpRight, walkLeft, walkRight, climb, ladderIdle,
attackLeft, attackRight;		// 角色的動畫
		CMovingBitmap dieLeft, dieRight, hp_3, hp_2, hp_1, hp_0;
		int x, y;				// 玩家左上角座標
		int hp;
		int jumpVel;				// 起跳初始速度
		int instantVelX;			// x軸瞬時速度
		int instantVelY;			// y軸瞬時速度
		int g;					// 加速度
		const int superStateCount = 50;
		int superStateCounter;
		Platform *floors;
		Ladder *ladder;
		/*************state variable***************/
		bool isMovingDown;			// 是否正在往下移動
		bool isMovingLeft;			// 是否正在往左移動
		bool isMovingRight;			// 是否正在往右移動
		bool isMovingUp;			// 是否正在往上移動
		bool isFacingLeft;
		bool isJumping;
		bool isClimbing;
		bool isOnTheGround;
		bool isInTheAir;
		bool attackKeyDown;
		bool isAttacking;
		bool isHurt;
		bool superState;
	};
}
```

Player.cpp

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

```
#include "Player.h"
namespace game_framework {
	/////////////////////////////////////////////////////////////////////////////
	// Player: Player class
	/////////////////////////////////////////////////////////////////////////////
	Player::Player()
	{
	}
	int Player::GetX1()
	{
		return x;
	}
	int Player::GetY1()
	{
		return y;
	}
	int Player::GetX2()
	{
		return x + idleLeft.Width();
	}
	int Player::GetY2()
	{
		return y + idleLeft.Height();
	}
	int Player::GetMidX()
	{
		return x + idleLeft.Width()/2;
	}
	int Player::GetMidY()
	{
		return y + idleLeft.Height()/2;
	}
	void Player::Initialize()
	{
		const int X_POS = 504;
		const int Y_POS = 380;
		x = X_POS;
		y = Y_POS;
		g = 3;
		jumpVel = -12;
		instantVelX = 0;
		instantVelY = 0;
		isFacingLeft = isOnTheGround = true;
		isMovingLeft = isMovingRight = isMovingUp = isMovingDown = false;
		isJumping = isClimbing = false;
		attackKeyDown = isAttacking = false;
		isHurt = false;
		superState = false;
		hp = 3;
	}
	void Player::LoadBitmap()
	{
		idleLeft.AddBitmap(character_left_stand, RGB(255, 0, 255));
		idleRight.AddBitmap(character_right_stand, RGB(255, 0, 255));
		lieLeft.AddBitmap(character_left_lie, RGB(255, 0, 255));
		lieRight.AddBitmap(character_right_lie, RGB(255, 0, 255));
		jumpLeft.AddBitmap(character_left_jump, RGB(255, 0, 255));
		jumpRight.AddBitmap(character_right_jump, RGB(255, 0, 255));
		walkLeft.AddBitmap(character_left_stand, RGB(255, 0, 255));
		walkLeft.AddBitmap(character_left_walk1, RGB(255, 0, 255));
		walkLeft.AddBitmap(character_left_stand, RGB(255, 0, 255));
		walkLeft.AddBitmap(character_left_walk1, RGB(255, 0, 255));
		walkLeft.AddBitmap(character_left_stand, RGB(255, 0, 255));
		walkLeft.AddBitmap(character_left_walk1, RGB(255, 0, 255));
		walkLeft.AddBitmap(character_left_walk2, RGB(255, 0, 255));
		walkRight.AddBitmap(character_right_stand, RGB(255, 0, 255));
		walkRight.AddBitmap(character_right_walk1, RGB(255, 0, 255));
		walkRight.AddBitmap(character_right_stand, RGB(255, 0, 255));
		walkRight.AddBitmap(character_right_walk1, RGB(255, 0, 255));
```

```
	walkRight.AddBitmap(character_right_stand, RGB(255, 0, 255));
	walkRight.AddBitmap(character_right_walk1, RGB(255, 0, 255));
	walkRight.AddBitmap(character_right_walk2, RGB(255, 0, 255));
	climb.AddBitmap(character_climb1, RGB(255, 0, 255));
	climb.AddBitmap(character_climb2, RGB(255, 0, 255));
	ladderIdle.AddBitmap(character_climb1, RGB(255, 0, 255));
	attackLeft.AddBitmap(character_left_attack1_1, RGB(255, 0, 255));
	attackLeft.AddBitmap(character_left_attack1_2, RGB(255, 0, 255));
	attackLeft.AddBitmap(character_left_attack1_2, RGB(255, 0, 255));	//判斷用，不會顯示
	attackRight.AddBitmap(character_right_attack1_1, RGB(255, 0, 255));
	attackRight.AddBitmap(character_right_attack1_2, RGB(255, 0, 255));
	attackRight.AddBitmap(character_right_attack1_2, RGB(255, 0, 255));	//判斷用，不會顯示
	dieLeft.LoadBitmap(character_left_die2, RGB(255, 0, 255));
	dieRight.LoadBitmap(character_right_die2, RGB(255, 0, 255));
	hp_3.LoadBitmap(hp3, RGB(255, 0, 255));
	hp_2.LoadBitmap(hp2, RGB(255, 0, 255));
	hp_1.LoadBitmap(hp1, RGB(255, 0, 255));
	hp_0.LoadBitmap(hp0, RGB(255, 0, 255));
}
void Player::OnMove()		//移動
{
	int STEP_SIZE = floors->movingSpeed();
	if (IsAlive()) {
		if (GetMidX() + instantVelX > 40 && GetMidX() + instantVelX < 600) {
			x += instantVelX;
		}
		y += instantVelY;
		if (isMovingLeft) {
			if (!isAttacking) {
				SetFacingLeft(true);
				if (IsInTheAir() && instantVelX >= 0) {
					instantVelX -= 1;
				}
				else if (isOnTheGround) {
					if (isMovingRight) {
						instantVelX = 0;
					}else if (!isHurt) {
						walkLeft.OnMove();
						instantVelX = -STEP_SIZE;
					}
				}
			}
		}
		if (isMovingRight) {
			if (!isAttacking) {
				SetFacingLeft(false);
				if (isInTheAir && instantVelX <= 0) {
					instantVelX += 1;
				}
				else if (isOnTheGround) {
					if (isMovingLeft) {
						instantVelX = 0;
					}else if (!isHurt) {
						walkRight.OnMove();
						instantVelX = STEP_SIZE;
					}
				}
			}
		}
		if (IsInTheAir()) {
			if (instantVelY < 20) {
				instantVelY += g;		// 受阻力影響，掉落速度不超過20左右
			}
		}
		else if (isClimbing) {
			climb.OnMove();
			instantVelX = 0;
			if (isJumping && isMovingLeft) {
				instantVelX = -STEP_SIZE;
```

```
                instantVelY = -10;
                SetIsClimbing(false);
        }else if (isJumping && isMovingRight) {
                instantVelX = STEP_SIZE;
                instantVelY = -10;
                SetIsClimbing(false);
        }
}
if (IsOnTheGround()) {
        instantVelY = 0;
        y = floors->getStandPointY(GetMidX()) - idleLeft.Height();
        if (!isMovingLeft && !isMovingRight) {
                instantVelX = 0;
        }
}
if (isMovingUp) {
        if (ladder->isLadder(GetMidX(), GetMidY())) {
                SetIsClimbing(true);
                x = (ladder->getX1() + ladder->getX2()) / 2 - idleLeft.Width() / 2;
                y -= 4;
                instantVelY = 0;
        }
        else if (isClimbing && ladder->onTheTop(GetMidY())) {
                SetIsClimbing(false);
                y = ladder->getY1() - idleLeft.Height();
        }
}
if (isMovingDown) {
        if (ladder->isLadder(GetMidX(), GetY2())) {
                SetIsClimbing(true);
                x = (ladder->getX1() + ladder->getX2()) / 2 - idleLeft.Width() / 2;
                y += 4;
                instantVelY = 0;
        }
        else if (isClimbing && ladder->atTheBottom(GetY2())) {
                SetIsClimbing(false);
        }
}
if (isJumping) {
        if (IsOnTheGround()) {
                instantVelY = jumpVel;
                y += instantVelY;
        }
}
if (isHurt) {
        superState = true;
        superStateCounter = superStateCount;
        if (!isClimbing) {
                instantVelY = -10;
                if (isFacingLeft) {
                        instantVelX = 7;
                }
                else {
                        instantVelX = -7;
                }
        }
        hp -= 1;
}
if (superState) {
        isHurt = false;
}
if (--superStateCounter <= 0) {
        superState = false;
}
if (isAttacking) {
        if (isFacingLeft) {
                attackLeft.OnMove();
        }
        else {
```

```
                        attackRight.OnMove();
                    }
                    if (isOnTheGround) {
                        instantVelX = 0;
                    }
                }
            }
        }
void Player::SetMovingDown(bool flag)
{
    isMovingDown = flag;
}
void Player::SetMovingLeft(bool flag)
{
    isMovingLeft = flag;
}
void Player::SetMovingRight(bool flag)
{
    isMovingRight = flag;
}
void Player::SetFacingLeft(bool flag)
{
    isFacingLeft = flag;
}
void Player::SetMovingUp(bool flag)
{
    isMovingUp = flag;
}
void Player::SetJumping(bool flag)
{
    isJumping = flag;
}
void Player::SetAttackKey(bool flag)
{
    attackKeyDown = flag;
}
void Player::SetAttacking(bool flag)
{
    isAttacking = flag;
}
void Player::SetGetHurt(bool flag)
{
    isHurt = flag;
}
void Player::SetXY(int nx, int ny)
{
    x = nx; y = ny;
}
void Player::OnShow()
{
    if (!IsAlive()) {
        hp_0.SetTopLeft(50, 455);
        hp_0.ShowBitmap();
        if (isFacingLeft) {
            dieLeft.SetTopLeft(x, y+18);
            dieLeft.ShowBitmap();
        }
        else {
            dieRight.SetTopLeft(x, y+18);
            dieRight.ShowBitmap();
        }
    }
    else {
        if (hp == 3) {
            hp_3.SetTopLeft(50, 455);
            hp_3.ShowBitmap();
        }
        else if (hp == 2) {
            hp_2.SetTopLeft(50, 455);
```

```
		hp_2.ShowBitmap();
	}
	else if (hp == 1) {
		hp_1.SetTopLeft(50, 455);
		hp_1.ShowBitmap();
	}
	if (isInTheAir) {
		if (isFacingLeft) {
			jumpLeft.SetTopLeft(x, y);
			jumpLeft.OnShow();
		}
		else {
			jumpRight.SetTopLeft(x, y);
			jumpRight.OnShow();
		}
	}
	else if (isClimbing) {
		if (isMovingUp || isMovingDown)
		{
			climb.SetTopLeft(x, y);
			climb.OnShow();
		}
		else {
			ladderIdle.SetTopLeft(x, y);
			ladderIdle.OnShow();
		}
	}
	else if(isOnTheGround){
		if (isAttacking) {
			if (isFacingLeft) {
				attackLeft.SetTopLeft(x-40, y+5);
				attackLeft.SetDelayCount(10);
				attackLeft.OnShow();
				if (!attackKeyDown && attackLeft.IsFinalBitmap()) {
					SetAttacking(false);
				}
				if (attackLeft.IsFinalBitmap()) {
					attackLeft.Reset();
				}
			}
			else {
				attackRight.SetTopLeft(x+10, y+5);
				attackRight.SetDelayCount(10);
				attackRight.OnShow();
				if (!attackKeyDown && attackRight.IsFinalBitmap()) {
					SetAttacking(false);
				}
				if (attackRight.IsFinalBitmap()) {
					attackRight.Reset();
				}
			}
		}
		else if (isMovingLeft) {
			walkLeft.SetTopLeft(x, y);
			walkLeft.OnShow();
		}
		else if (isMovingRight) {
			walkRight.SetTopLeft(x, y);
			walkRight.OnShow();
		}
		else if (isMovingDown) {
			if (isFacingLeft) {
				lieLeft.SetTopLeft(x, y + 25 );
				lieLeft.OnShow();
			}
			else {
				lieRight.SetTopLeft(x, y + 25);
				lieRight.OnShow();
			}
```

```
                        }
                        else if (isFacingLeft) {
                                idleLeft.SetTopLeft(x, y);
                                idleLeft.OnShow();
                        }
                        else {
                                idleRight.SetTopLeft(x, y);
                                idleRight.OnShow();
                        }
                }
        }
}
bool Player::IsOnTheGround()
{
        isOnTheGround = floors->isFloor(GetMidX(), GetY2(), instantVelY);
        return isOnTheGround;
}
void Player::SetIsClimbing(bool flag)
{
        isClimbing = flag;
}
bool Player::IsInSuperState()
{
        return superState;
}
bool Player::IsAlive()
{
        return hp > 0;
}
bool Player::IsInTheAir()
{
        isInTheAir = !(IsOnTheGround() || isClimbing);
        return isInTheAir;
}
bool Player::IsAttacking() {
        return isAttacking;
}
tuple<int, int, int, int> Player::GetAttackRange()
{
        if (isAttacking) {
                if (isFacingLeft) {
                        return make_tuple(x-30, y+30, x, y+60);
                }
                else {
                        return make_tuple(GetX2(), y + 30, GetX2()+30, y + 60);
                }
        }
        else {
                return make_tuple(NULL, NULL, NULL, NULL);
        }
}
void Player::SetMap(Platform *plat, Ladder *lad)
{
        floors = plat;
        ladder = lad;
}
}
```

Monster.h

```
namespace game_framework {
        class Monster {
        public:
                Monster(int id, int num, int lb, int rb, int x, int y);
                void initialize();
                int GetX1();
                int GetY1();
                int GetX2();
                int GetY2();
```

```
		void GetHurt(int dmg);
		bool isCollision(int tx1, int ty1, int tx2, int ty2);	//是否與target碰撞
		bool IsAlive();						// 是否活著
		bool IsDisappear();					// 消失
		void SetXY(int nx, int ny);
		void LoadBitmap();
		void OnMove();
		void OnShow();
		/****密技用****/
		void zeroHP();
	private:
		CAnimation moveLeft, moveRight, getHurtLeft, getHurtRight, dyingLeft, dyingRight;
		int monsterID, number;
		int leftBound, rightBound;
		int x, y;
		int HP;
		int step;
		bool isHurt;				// 被攻擊
		bool isAlive;				// 是否活著
		bool isMovingLeft;
		bool isMovingRight;
		bool isFacingRight;			// 面向右為true  面向左為false
		bool isDisappear;
	};
}
```

Monster.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Monster.h"
namespace game_framework {
	Monster::Monster(int id, int num, int lb, int rb, int x, int y)
	{
		monsterID = id;
		number = num;
		leftBound = lb;
		rightBound = rb;
		this->x = x;
		this->y = y;
		HP = 30;
		step = 1;
		isMovingLeft = isMovingRight = false;
		isFacingRight = false;
		isHurt = false;
		isAlive = true;
		isDisappear = false;
	}
	void Monster::initialize()
	{
		HP = 30;
		isAlive = true;
		isDisappear = false;
	}
	int Monster::GetX1()
	{
		return x;
	}
	int Monster::GetY1()
	{
		return y;
	}
	int Monster::GetX2()
	{
		return x + moveRight.Width();
```

```
}
int Monster::GetY2()
{
	return y + moveRight.Height();
}
bool Monster::isCollision(int tx1, int ty1, int tx2, int ty2)
{
	if (!IsAlive())
		return false;
	if (GetX1()+6 <= tx2 && GetY1()+6 <= ty2 && GetX2()-6 >= tx1 && GetY2()-6 >= ty1)
		return true;
	else
		return false;
}
void Monster::GetHurt(int dmg)
{
	HP -= dmg;
	isHurt = true;
}
bool Monster::IsAlive()
{
	return HP > 0;
}
bool Monster::IsDisappear()
{
	return isDisappear;
}
void Monster::SetXY(int nx, int ny)
{
	x = nx;
	y = ny;
}
void Monster::LoadBitmap()
{
	if(monsterID == 1){
		moveLeft.AddBitmap(snail_left_walk1, RGB(255, 0, 255));
		moveLeft.AddBitmap(snail_left_walk2, RGB(255, 0, 255));
		moveLeft.AddBitmap(snail_left_walk3, RGB(255, 0, 255));
		moveRight.AddBitmap(snail_right_walk1, RGB(255, 0, 255));
		moveRight.AddBitmap(snail_right_walk2, RGB(255, 0, 255));
		moveRight.AddBitmap(snail_right_walk3, RGB(255, 0, 255));
		getHurtLeft.AddBitmap(snail_left_hurt, RGB(255, 0, 255));
		getHurtRight.AddBitmap(snail_right_hurt, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snail_left_die1, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snail_left_die2, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snail_left_die3, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snail_left_die3, RGB(255, 0, 255));
		dyingRight.AddBitmap(snail_right_die1, RGB(255, 0, 255));
		dyingRight.AddBitmap(snail_right_die2, RGB(255, 0, 255));
		dyingRight.AddBitmap(snail_right_die3, RGB(255, 0, 255));
		dyingRight.AddBitmap(snail_right_die3, RGB(255, 0, 255));
	}
	else if(monsterID == 2){
		moveLeft.AddBitmap(snake_left_walk1, RGB(255, 0, 255));
		moveLeft.AddBitmap(snake_left_walk2, RGB(255, 0, 255));
		moveLeft.AddBitmap(snake_left_walk3, RGB(255, 0, 255));
		moveRight.AddBitmap(snake_right_walk1, RGB(255, 0, 255));
		moveRight.AddBitmap(snake_right_walk2, RGB(255, 0, 255));
		moveRight.AddBitmap(snake_right_walk3, RGB(255, 0, 255));
		getHurtLeft.AddBitmap(snake_left_hurt, RGB(255, 0, 255));
		getHurtRight.AddBitmap(snake_right_hurt, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snake_left_die1, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snake_left_die2, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snake_left_die3, RGB(255, 0, 255));
		dyingLeft.AddBitmap(snake_left_die3, RGB(255, 0, 255));
		dyingRight.AddBitmap(snake_right_die1, RGB(255, 0, 255));
		dyingRight.AddBitmap(snake_right_die2, RGB(255, 0, 255));
		dyingRight.AddBitmap(snake_right_die3, RGB(255, 0, 255));
		dyingRight.AddBitmap(snake_right_die3, RGB(255, 0, 255));
```

```
}
else if (monsterID == 3) {
	moveLeft.AddBitmap(mushroom_left_walk1, RGB(255, 0, 255));
	moveLeft.AddBitmap(mushroom_left_walk2, RGB(255, 0, 255));
	moveLeft.AddBitmap(mushroom_left_walk3, RGB(255, 0, 255));
	moveRight.AddBitmap(mushroom_right_walk1, RGB(255, 0, 255));
	moveRight.AddBitmap(mushroom_right_walk2, RGB(255, 0, 255));
	moveRight.AddBitmap(mushroom_right_walk3, RGB(255, 0, 255));
	getHurtLeft.AddBitmap(mushroom_left_hurt, RGB(255, 0, 255));
	getHurtRight.AddBitmap(mushroom_right_hurt, RGB(255, 0, 255));
	dyingLeft.AddBitmap(mushroom_left_die1, RGB(255, 0, 255));
	dyingLeft.AddBitmap(mushroom_left_die2, RGB(255, 0, 255));
	dyingLeft.AddBitmap(mushroom_left_die3, RGB(255, 0, 255));
	dyingLeft.AddBitmap(mushroom_left_die3, RGB(255, 0, 255));
	dyingRight.AddBitmap(mushroom_right_die1, RGB(255, 0, 255));
	dyingRight.AddBitmap(mushroom_right_die2, RGB(255, 0, 255));
	dyingRight.AddBitmap(mushroom_right_die3, RGB(255, 0, 255));
	dyingRight.AddBitmap(mushroom_right_die3, RGB(255, 0, 255));
}
else if (monsterID == 4) {
	moveLeft.AddBitmap(green_left_walk1, RGB(255, 0, 255));
	moveLeft.AddBitmap(green_left_walk2, RGB(255, 0, 255));
	moveLeft.AddBitmap(green_left_walk3, RGB(255, 0, 255));
	moveLeft.AddBitmap(green_left_walk4, RGB(255, 0, 255));
	moveLeft.AddBitmap(green_left_walk5, RGB(255, 0, 255));
	moveLeft.AddBitmap(green_left_walk6, RGB(255, 0, 255));
	moveLeft.AddBitmap(green_left_walk7, RGB(255, 0, 255));
	moveRight.AddBitmap(green_right_walk1, RGB(255, 0, 255));
	moveRight.AddBitmap(green_right_walk2, RGB(255, 0, 255));
	moveRight.AddBitmap(green_right_walk3, RGB(255, 0, 255));
	moveRight.AddBitmap(green_right_walk4, RGB(255, 0, 255));
	moveRight.AddBitmap(green_right_walk5, RGB(255, 0, 255));
	moveRight.AddBitmap(green_right_walk6, RGB(255, 0, 255));
	moveRight.AddBitmap(green_right_walk7, RGB(255, 0, 255));
	getHurtLeft.AddBitmap(green_left_hurt, RGB(255, 0, 255));
	getHurtRight.AddBitmap(green_right_hurt, RGB(255, 0, 255));
	dyingLeft.AddBitmap(green_left_die1, RGB(255, 0, 255));
	dyingLeft.AddBitmap(green_left_die2, RGB(255, 0, 255));
	dyingLeft.AddBitmap(green_left_die3, RGB(255, 0, 255));
	dyingLeft.AddBitmap(green_left_die4, RGB(255, 0, 255));
	dyingLeft.AddBitmap(green_left_die4, RGB(255, 0, 255));
	dyingRight.AddBitmap(green_right_die1, RGB(255, 0, 255));
	dyingRight.AddBitmap(green_right_die2, RGB(255, 0, 255));
	dyingRight.AddBitmap(green_right_die3, RGB(255, 0, 255));
	dyingRight.AddBitmap(green_right_die4, RGB(255, 0, 255));
	dyingRight.AddBitmap(green_right_die4, RGB(255, 0, 255));
}
else if (monsterID == 5) {
	moveLeft.AddBitmap(crab_left_walk1, RGB(255, 0, 255));
	moveLeft.AddBitmap(crab_left_walk2, RGB(255, 0, 255));
	moveLeft.AddBitmap(crab_left_walk3, RGB(255, 0, 255));
	moveLeft.AddBitmap(crab_left_walk4, RGB(255, 0, 255));
	moveRight.AddBitmap(crab_right_walk1, RGB(255, 0, 255));
	moveRight.AddBitmap(crab_right_walk2, RGB(255, 0, 255));
	moveRight.AddBitmap(crab_right_walk3, RGB(255, 0, 255));
	moveRight.AddBitmap(crab_right_walk4, RGB(255, 0, 255));
	getHurtLeft.AddBitmap(crab_left_hurt, RGB(255, 0, 255));
	getHurtRight.AddBitmap(crab_right_hurt, RGB(255, 0, 255));
	dyingLeft.AddBitmap(crab_left_die1, RGB(255, 0, 255));
	dyingLeft.AddBitmap(crab_left_die2, RGB(255, 0, 255));
	dyingLeft.AddBitmap(crab_left_die3, RGB(255, 0, 255));
	dyingLeft.AddBitmap(crab_left_die4, RGB(255, 0, 255));
	dyingLeft.AddBitmap(crab_left_die4, RGB(255, 0, 255));
	dyingRight.AddBitmap(crab_right_die1, RGB(255, 0, 255));
	dyingRight.AddBitmap(crab_right_die2, RGB(255, 0, 255));
	dyingRight.AddBitmap(crab_right_die3, RGB(255, 0, 255));
	dyingRight.AddBitmap(crab_right_die4, RGB(255, 0, 255));
	dyingRight.AddBitmap(crab_right_die4, RGB(255, 0, 255));
}
```

```
	else if (monsterID == 6) {
		moveLeft.AddBitmap(star_left_walk1, RGB(255, 0, 255));
		moveLeft.AddBitmap(star_left_walk2, RGB(255, 0, 255));
		moveLeft.AddBitmap(star_left_walk3, RGB(255, 0, 255));
		moveLeft.AddBitmap(star_left_walk4, RGB(255, 0, 255));
		moveLeft.AddBitmap(star_left_walk5, RGB(255, 0, 255));
		moveLeft.AddBitmap(star_left_walk6, RGB(255, 0, 255));
		moveRight.AddBitmap(star_right_walk1, RGB(255, 0, 255));
		moveRight.AddBitmap(star_right_walk2, RGB(255, 0, 255));
		moveRight.AddBitmap(star_right_walk3, RGB(255, 0, 255));
		moveRight.AddBitmap(star_right_walk4, RGB(255, 0, 255));
		moveRight.AddBitmap(star_right_walk5, RGB(255, 0, 255));
		moveRight.AddBitmap(star_right_walk6, RGB(255, 0, 255));
		getHurtLeft.AddBitmap(star_left_hurt, RGB(255, 0, 255));
		getHurtRight.AddBitmap(star_right_hurt, RGB(255, 0, 255));
		dyingLeft.AddBitmap(star_left_die1, RGB(255, 0, 255));
		dyingLeft.AddBitmap(star_left_die2, RGB(255, 0, 255));
		dyingLeft.AddBitmap(star_left_die3, RGB(255, 0, 255));
		dyingLeft.AddBitmap(star_left_die4, RGB(255, 0, 255));
		dyingLeft.AddBitmap(star_left_die5, RGB(255, 0, 255));
		dyingLeft.AddBitmap(star_left_die5, RGB(255, 0, 255));
		dyingRight.AddBitmap(star_right_die1, RGB(255, 0, 255));
		dyingRight.AddBitmap(star_right_die2, RGB(255, 0, 255));
		dyingRight.AddBitmap(star_right_die3, RGB(255, 0, 255));
		dyingRight.AddBitmap(star_right_die4, RGB(255, 0, 255));
		dyingRight.AddBitmap(star_right_die5, RGB(255, 0, 255));
		dyingRight.AddBitmap(star_right_die5, RGB(255, 0, 255));
	}
}
void Monster::OnMove()
{
	moveLeft.OnMove();
	moveRight.OnMove();
	dyingLeft.OnMove();
	dyingRight.OnMove();
	srand(number*(unsigned int)time(NULL));
	int randStatus = rand() % 3;
	if (IsAlive()) {
		if (randStatus == 0 && x > leftBound) {
			isMovingLeft = true;
			isMovingRight = false;
			isFacingRight = false;
		}
		else if (randStatus == 1 && x + moveRight.Width() < rightBound) {
			isMovingLeft = false;
			isMovingRight = true;
			isFacingRight = true;
		}
		else {
			isMovingLeft = false;
			isMovingRight = false;
		}
		if (!isHurt) {
			if (isMovingLeft) {
				x -= step;
			}
			else if (isMovingRight) {
				x += step;
			}
		}
		else {
			isMovingLeft = false;
			isMovingRight = false;
			isHurt = false;
		}
	}
}
void Monster::OnShow()
{
```

```
		if (!isDisappear) {
			if (IsAlive()) {
				if (isFacingRight) {
					if (isHurt) {
						getHurtRight.SetTopLeft(x, y);
						getHurtRight.SetDelayCount(5);
						getHurtRight.OnShow();
					}
					else {
						moveRight.SetTopLeft(x, y);
						moveRight.SetDelayCount(5);
						moveRight.OnShow();
					}
				}
				else {
					if (isHurt) {
						getHurtLeft.SetTopLeft(x, y);
						getHurtLeft.SetDelayCount(5);
						getHurtLeft.OnShow();
					}
					else {
						moveLeft.SetTopLeft(x, y);
						moveLeft.SetDelayCount(5);
						moveLeft.OnShow();
					}
				}
			}
			else {
				if (isFacingRight) {
					dyingRight.SetTopLeft(x, y);
					dyingRight.SetDelayCount(10);
					dyingRight.OnShow();
					if (dyingRight.IsFinalBitmap()) {
						isDisappear = true;
					}
				}
				else {
					dyingLeft.SetTopLeft(x, y);
					dyingLeft.SetDelayCount(10);
					dyingLeft.OnShow();
					if (dyingLeft.IsFinalBitmap()) {
						isDisappear = true;
					}
				}
			}
		}
	}
	void Monster::zeroHP()
	{
		HP = 0;
	}
}
```

Platform.h

```
namespace game_framework {
	//以長方形表示平台，三角形表示斜坡
	class Platform {
	public:
		Platform();
		void add(tuple<string, int, int, int, int> plat);
		int getStandPointY(int tx);
		int movingSpeed();                                    //tell player step
		bool isFloor(int tx, int ty, int vy);      //vy用來預測 = velocity of y
		~Platform();
	protected:
		string getType();
		void assignXY();                //  使用x1 y1等縮短字數
	private:
```

```
		vector<tuple<string, int, int, int, int>> floors;
		tuple<string, int, int, int, int> onThisFloor;
		int x1, y1, x2, y2;		// 左上和右下，若是斜坡則是左端和右端
	};
}
```

Platform.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Platform.h"
namespace game_framework {
	Platform::Platform() {
		onThisFloor = make_tuple("f", 40, 480, 600, 480);;
		floors.reserve(20);
	}
	void Platform::add(tuple<string, int, int, int, int> plat)
	{
		floors.push_back(plat);
	}
	int Platform::getStandPointY(int tx) {
		assignXY();
		if (getType() == "s") {
			return y1 + (y2 - y1)*(tx - x1) / (x2 - x1);
		}
		else {
			return y1;
		}
	}
	int Platform::movingSpeed() {
		//向右走為正
		if (getType() == "s") {
			return 4;
		}
		else {
			return 5;
		}
	}
	bool Platform::isFloor(int tx, int ty, int vy)
	{
		for (int i = 0; i < int(floors.size()); i++) {
			if (get<0>(floors[i]) == "f") {
				if (ty <= get<2>(floors[i])) {
					if (tx >= get<1>(floors[i]) && tx <= get<3>(floors[i]) && ty+vy >= get<2>(floors[i])) {
						onThisFloor = floors[i];
						return true;
					}
				}
			}
			else if (get<0>(floors[i]) == "s") {
				if (tx >= get<1>(floors[i]) && tx <= get<3>(floors[i]) && ty+vy >= -4+(get<2>(floors[i]) +
(get<4>(floors[i]) - get<2>(floors[i]))*(tx - get<1>(floors[i])) / (get<3>(floors[i]) - get<1>(floors[i])))) {
					if (get<4>(floors[i]) > get<2>(floors[i]) && ty <= 4+(get<2>(floors[i]) + (get<4>(floors[i])
- get<2>(floors[i]))*(tx - get<1>(floors[i])) / (get<3>(floors[i]) - get<1>(floors[i])))) {
						onThisFloor = floors[i];
						return true;
					}
					else if (get<2>(floors[i]) > get<4>(floors[i]) && ty <= 4+(get<2>(floors[i]) +
(get<4>(floors[i]) - get<2>(floors[i]))*(tx - get<1>(floors[i])) / (get<3>(floors[i]) - get<1>(floors[i])))) {
						onThisFloor = floors[i];
						return true;
					}
				}
			}
		}
```

```
                onThisFloor = make_tuple("f", 40, 480, 600, 480);
                return false;
        }
        Platform::~Platform()
        {
        }
        string Platform::getType() {
                return get<0>(onThisFloor);
        }
        void Platform::assignXY() {
                x1 = get<1>(onThisFloor);
                y1 = get<2>(onThisFloor);
                x2 = get<3>(onThisFloor);
                y2 = get<4>(onThisFloor);
        }
}
```

Ladder.h

```
namespace game_framework {
        class Ladder {
        public:
                Ladder();
                int getX1();    //左上
                int getY1();
                int getX2();    //右下
                int getY2();
                void add(tuple<int, int, int, int> ladder);
                bool isLadder(int tx, int ty);
                bool onTheTop(int ty);
                bool atTheBottom(int ty);
        protected:
                int x1, y1, x2, y2;
                vector<tuple<int, int, int, int>> ladders;
                tuple<int, int, int, int> onThisLadder;
        };
}
```

Ladder.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "Ladder.h"
namespace game_framework {
        Ladder::Ladder() {
                x1 = y1 = x2 = y2 = 0;
                ladders.reserve(10);
        }
        int Ladder::getX1() {
                return get<0>(onThisLadder);
        }
        int Ladder::getY1() {
                return get<1>(onThisLadder);
        }
        int Ladder::getX2() {
                return get<2>(onThisLadder);
        }
        int Ladder::getY2() {
                return get<3>(onThisLadder);
        }
        void Ladder::add(tuple<int, int, int, int> ladder)
        {
                ladders.push_back(ladder);
        }
        bool Ladder::isLadder(int tx, int ty) {
```

```
		for (int i = 0; i < int(ladders.size()); i++) {
			if (tx >= get<0>(ladders[i]) && tx <= get<2>(ladders[i]) && ty >= get<1>(ladders[i]) && ty <=
get<3>(ladders[i])) {
				onThisLadder = ladders[i];
				return true;
			}
		}
		return false;
	}
	bool Ladder::onTheTop(int ty)
	{
		if (ty <= getY1()) {
			return true;
		}
		else {
			return false;
		}
	}
	bool Ladder::atTheBottom(int ty)
	{
		if (ty >= getY2()) {
			return true;
		}
		else {
			return false;
		}
	}
}.
```

StageMap.h

```
#include "Platform.h"
#include "Ladder.h"
#include "Monster.h"
namespace game_framework {
	class StageMap {
	public:
		StageMap();
		void MapInit();
		void MonsterInit();
		void ChangeStage(int s);
		/*create function*/
		void CreatePlatform();
		void CreateLadder();
		void CreateMonsters();
		/*get function*/
		int GetStage();
		Platform* GetPlatform();
		Ladder* GetLadder();
		vector<Monster>* GetMonsters();
		/*game function*/
		void LoadBitmap();
		void LoadMonsterBitmap();
		void OnMove();
		void OnShow();
	private:
		int stage;
		CMovingBitmap background1;
		CMovingBitmap background2;
		CMovingBitmap background3;
		CMovingBitmap stageTitle1;
		CMovingBitmap stageTitle2;
		CMovingBitmap stageTitle3;
		Platform * ppointer;
		Ladder * lpointer;
		vector<Monster> * mpointer;
		Platform plat1;
		Platform plat2;
		Platform plat3;
```

```
		Ladder ladder1;
		Ladder ladder2;
		Ladder ladder3;
		vector<Monster> monsters1;
		vector<Monster> monsters2;
		vector<Monster> monsters3;
		const int titleCount = 20;
		int titleCounter;
		bool titleShow;
	};
}
```

StageMap.cpp

```
#include "stdafx.h"
#include "Resource.h"
#include <mmsystem.h>
#include <ddraw.h>
#include "audio.h"
#include "gamelib.h"
#include "StageMap.h"
namespace game_framework {

	StageMap::StageMap() {
		monsters1.reserve(10);
		monsters2.reserve(10);
		monsters3.reserve(10);
	}
	void StageMap::MapInit() {
		stage = 1;
		CreatePlatform();
		CreateLadder();
		CreateMonsters();
		titleShow = true;
		titleCounter = titleCount;
	}
	int StageMap::GetStage()
	{
		return stage;
	}
	void StageMap::ChangeStage(int s)
	{
		stage = s;
		titleShow = true;
		titleCounter = titleCount;
	}
	void StageMap::CreatePlatform() {
		//stage 1 map
		plat1.add(make_tuple("f", 380, 460, 600, 480));		//f1
		plat1.add(make_tuple("s", 310, 417, 380, 460));		//s1
		plat1.add(make_tuple("f", 40, 420, 315, 480));		//f2
		plat1.add(make_tuple("f", 355, 387, 430, 397));		//f3
		plat1.add(make_tuple("f", 430, 355, 600, 365));		//f4
		plat1.add(make_tuple("f", 320, 284, 600, 294));		//f5
		plat1.add(make_tuple("f", 40, 284, 290, 294));		//f6
		plat1.add(make_tuple("f", 40, 177, 145, 187));		//f7
		plat1.add(make_tuple("f", 170, 177, 465, 187));		//f8
		plat1.add(make_tuple("f", 495, 177, 565, 187));		//f9
		plat1.add(make_tuple("f", 320, 74, 600, 84));		//f10
		plat1.add(make_tuple("f", 40, 74, 285, 84));		//f11
		//stage 2 map
		plat2.add(make_tuple("f", 40, 460, 600, 480));		//f1
		plat2.add(make_tuple("s", 75, 460, 183, 388));		//s1
		plat2.add(make_tuple("f", 178, 390, 250, 400));		//f2
		plat2.add(make_tuple("f", 280, 390, 365, 400));		//f3
		plat2.add(make_tuple("f", 385, 390, 470, 400));		//f4
		plat2.add(make_tuple("f", 490, 390, 600, 400));		//f5
		plat2.add(make_tuple("f", 420, 285, 600, 295));		//f6
		plat2.add(make_tuple("f", 40, 285, 400, 295));		//f7
```

```
	plat2.add(make_tuple("f", 40, 180, 230, 190));		//f8
	plat2.add(make_tuple("f", 250, 180, 600, 190));		//f9
	plat2.add(make_tuple("f", 490, 75, 600, 85));		//f10
	plat2.add(make_tuple("f", 385, 75, 465, 85));		//f11
	plat2.add(make_tuple("f", 280, 75, 365, 85));		//f12
	plat2.add(make_tuple("f", 175, 75, 260, 85));		//f13
	plat2.add(make_tuple("f", 40, 75, 150, 85));	//f14
	//stage 3 map
	plat3.add(make_tuple("f", 40, 460, 600, 480));		//f1
	plat3.add(make_tuple("f", 40, 360, 150, 370));		//f2
	plat3.add(make_tuple("f", 40, 285, 600, 295));		//f3
	plat3.add(make_tuple("f", 40, 180, 600, 190));		//f4
	plat3.add(make_tuple("f", 40, 75, 600, 85));	//f5
}
void StageMap::CreateLadder()
{
	// stage 1 ladders
	ladder1.add(make_tuple(400, 387, 420, 450));		//f1 to f3
	ladder1.add(make_tuple(539, 284, 559, 345));
	ladder1.add(make_tuple(83, 177, 103, 274));	//f6 to f7
	ladder1.add(make_tuple(432, 74, 452, 167));	//f8 to f10
	// stage 2 ladders
	ladder2.add(make_tuple(537, 285, 557, 370));	//f5 to f6
	ladder2.add(make_tuple(82, 180, 102, 265));	//f7 to f8
	ladder2.add(make_tuple(572, 75, 592, 165));	//f9 to f10
	// stage 3 ladders
	ladder3.add(make_tuple(437, 300, 447, 440));	//
	ladder3.add(make_tuple(402, 300, 412, 440));	//
	ladder3.add(make_tuple(367, 300, 377, 440));	//
	ladder3.add(make_tuple(334, 300, 344, 440));	//
	ladder3.add(make_tuple(299, 300, 309, 440));	//
	ladder3.add(make_tuple(263, 300, 273, 440));	//
	ladder3.add(make_tuple(229, 300, 239, 440));	//
	ladder3.add(make_tuple(195, 300, 205, 440));	//
	ladder3.add(make_tuple(158, 300, 168, 440));	//
	ladder3.add(make_tuple(117, 285, 137, 340));	//f2 to f3
	ladder3.add(make_tuple(468, 180, 488, 263));	//f3 to f4
	ladder3.add(make_tuple(538, 75, 558, 160));	//f4 to f5
}
void StageMap::CreateMonsters()
{
	// stage1 monsters
	monsters1.push_back(Monster(2, 1, 40, 315, 200, 390));		//f2
	monsters1.push_back(Monster(1, 2, 320, 600, 500, 248));		//f5
	monsters1.push_back(Monster(1, 3, 40, 290, 50, 248));		//f6
	monsters1.push_back(Monster(2, 4, 170, 465, 400, 145));		//f8
	monsters1.push_back(Monster(1, 5, 320, 600, 350, 39));		//f10
	monsters1.push_back(Monster(2, 6, 40, 285, 250, 41));		//f11
	// stage2 monsters
	monsters2.push_back(Monster(4, 1, 40, 600, 380, 405));		//f1
	monsters2.push_back(Monster(3, 2, 180, 250, 210, 340));		//f2
	monsters2.push_back(Monster(4, 3, 40, 400, 120, 233));		//f7
	monsters2.push_back(Monster(3, 4, 40, 230, 190, 132));		//f8
	monsters2.push_back(Monster(3, 5, 250, 600, 500, 132));		//f9
	// stage3 monsters
	monsters3.push_back(Monster(5, 1, 40, 600, 380, 414));		//f1
	monsters3.push_back(Monster(6, 2, 40, 600, 200, 211));
	monsters3.push_back(Monster(6, 3, 40, 600, 300, 106));
	monsters3.push_back(Monster(5, 4, 40, 600, 125, 137));
}
void StageMap::MonsterInit()
{
	for (size_t i = 0; i < monsters1.size(); i++) {
		monsters1.at(i).initialize();
	}
	for (size_t i = 0; i < monsters2.size(); i++) {
		monsters2.at(i).initialize();
	}
	for (size_t i = 0; i < monsters3.size(); i++) {
```

```
                monsters3.at(i).initialize();
        }
}
Platform* StageMap::GetPlatform() {
        switch (stage)
        {
        case 1:
                return &plat1;
        case 2:
                return &plat2;
        case 3:
                return &plat3;
        default:
                return NULL;
        }
}
Ladder * StageMap::GetLadder()
{
        switch (stage)
        {
        case 1:
                return &ladder1;
        case 2:
                return &ladder2;
        case 3:
                return &ladder3;
        default:
                return NULL;
        }
}
vector<Monster>* StageMap::GetMonsters()
{
        if (stage == 1)
                return &monsters1;
        else if (stage == 2)
                return &monsters2;
        else if (stage == 3)
                return &monsters3;
        else
                return NULL;
}

void StageMap::LoadBitmap()
{
        background1.LoadBitmap(stage1_background);
        background2.LoadBitmap(stage2_background);
        background3.LoadBitmap(stage3_background);
        stageTitle1.LoadBitmap(stage1_text, RGB(255, 0, 255));
        stageTitle2.LoadBitmap(stage2_text, RGB(255, 0, 255));
        stageTitle3.LoadBitmap(stage3_text, RGB(255, 0, 255));
}
void StageMap::LoadMonsterBitmap()
{
        for (size_t i = 0; i < monsters1.size(); i++) {
                monsters1.at(i).LoadBitmap();
        }
        for (size_t i = 0; i < monsters2.size(); i++) {
                monsters2.at(i).LoadBitmap();
        }
        for (size_t i = 0; i < monsters3.size(); i++) {
                monsters3.at(i).LoadBitmap();
        }
}
void StageMap::OnMove() {
        if (--titleCounter <= 0) {
                titleShow = false;
        }
}
void StageMap::OnShow()
```

```
	{
		switch (stage)
		{
		case 1:
			background1.SetTopLeft(40, 0);
			background1.ShowBitmap();
			if (titleShow) {
				stageTitle1.SetTopLeft(263, 225);
				stageTitle1.ShowBitmap();
			}
			break;
		case 2:
			background2.SetTopLeft(40, 0);
			background2.ShowBitmap();
			if (titleShow) {
				stageTitle2.SetTopLeft(263, 225);
				stageTitle2.ShowBitmap();
			}
			break;
		case 3:
			background3.SetTopLeft(40, 0);
			background3.ShowBitmap();
			if (titleShow) {
				stageTitle3.SetTopLeft(263, 225);
				stageTitle3.ShowBitmap();
			}
			break;
		default:
			break;
		}
	}
}
```