**目錄**

[一、 簡介 - 1 -](#_Toc517278713)

[1. 動機 - 1 -](#_Toc517278714)

[2. 分工 - 1 -](#_Toc517278715)

[二、 遊戲介紹 - 2 -](#_Toc517278716)

[1. 遊戲說明 - 2 -](#_Toc517278717)

[a. 遊戲內容 - 2 -](#_Toc517278718)

[b. 遊戲操作 - 2 -](#_Toc517278719)

[c. 遊戲劇情 - 3 -](#_Toc517278720)

[d. 角色介紹 - 3 -](#_Toc517278721)

[2. 遊戲圖形 - 5 -](#_Toc517278722)

[3. 遊戲音效 - 9 -](#_Toc517278723)

[三、 程式設計 - 10 -](#_Toc517278724)

[1. 程式架構 - 10 -](#_Toc517278725)

[2. 程式類別 - 11 -](#_Toc517278726)

[3. 程式技術 - 12 -](#_Toc517278727)

[四、 結語 - 13 -](#_Toc517278728)

[1. 問題及解決方法 - 13 -](#_Toc517278729)

[2. 時間表 - 15 -](#_Toc517278730)

[3. 貢獻比例 - 16 -](#_Toc517278731)

[4. 檢核表 - 16 -](#_Toc517278732)

[5. 收穫 - 17 -](#_Toc517278733)

[6. 心得、感想 - 18 -](#_Toc517278734)

[7. 對於本課程的建議 - 19 -](#_Toc517278735)

[五、 附錄 - 21 -](#_Toc517278736)

[Mygame.h - 21 -](#_Toc517278737)

[Mygame.cpp - 24 -](#_Toc517278738)

[Kirby.h - 41 -](#_Toc517278739)

[Kirby.cpp - 44 -](#_Toc517278740)

[Enemy.h - 79 -](#_Toc517278741)

[Enemy.cpp - 80 -](#_Toc517278742)

[Map.h - 84 -](#_Toc517278743)

[Map.cpp - 85 -](#_Toc517278744)

[Door.h - 90 -](#_Toc517278745)

[Door.cpp - 91 -](#_Toc517278746)

# 簡介

## 動機

經過期初的討論，從眾多的遊戲中，我們想到卡比之星這部由任天堂出品的經典作品，其中鏡之大迷宮這個系列陪伴我們度過不少的童年時光，對這部作品很熟悉也很有興趣，起初是抱著嘗試看看的想法，沒想到這個主題非常適合老師的framework，便以此做為物件導向實習的主題。

## 分工

每個部份我們都有分配負責人，但非完全由個人完成，是經過兩人多次討論、不斷修正的結果。

以下列出兩人負責的項目：

周宇天：地圖系統、閃電怪、火焰怪、樹王、鏡子碎片、開始畫面。

林彥廷：卡比、閃電卡比、火焰卡比、飛機王、音效、操作說明。

# 遊戲介紹

## 遊戲說明

### 遊戲內容

此遊戲是模仿星之卡比鏡之大迷宮，以練習物件導向程式設計為目的，利用一學期的時間撰寫，達到實作與理論並重的課程練習，我們使用陳偉凱老師提供的framework為架構，加上兩人合力完成的Kirby、Map等等物件，經過多次的討論與修正，最終完成整個遊戲。

### 遊戲操作

↑：吸氣飛翔、進入傳送門

↓：蹲下、變身

←：左移

→：右移

Ｚ：吐氣、攻擊、吸怪

Ｘ：跳躍

Ｓ：血量補滿

↓　+　Z：踢擊

→　+　C：向右跑步

←　+　C：向左跑步

### 遊戲劇情

此遊戲包含一張主地圖、兩張普通地圖與兩張魔王地圖，主地圖用來顯示鏡子碎片的收集狀況與連通各個地圖的傳送門；普通地圖包含各種小怪，提供玩家練習操作與變身，兩張普通地圖有各自的主題，一張是洞窟，另一張是凹凸草原，讓玩家能體驗不同的遊戲風景；最後是魔王地圖，這是全遊戲的關鍵，需要打敗兩隻魔王奪回鏡子碎片才能過關。

### 角色介紹

遊戲裡包含兩隻魔王，分別是塔王與飛機王，塔王的地圖中會隨機落下石塊攻擊卡比，但卡比也能利用吸入落下的石塊，吐出星星攻擊塔王，算是比較容易攻破的魔王；飛機王是相對比較困難的，它會在地圖中上下飛行，讓玩家不好攻擊，撞到卡比也會讓卡比受到傷害，更恐怖的是它會發射飛彈，不僅僅是碰到飛彈會受傷，更要小心它的爆炸範圍，建議玩家多利用空氣砲來攻擊，即便是變身過後，飛機王也不是個好對付的對手。

小怪部分有普通的小怪、閃電怪和火焰怪，普通的小怪會在地圖中來回走動，用踢擊可以近距離將它踢死，玩家可以利用它來練習操作，吃掉它是不能變身的喔！閃電怪會追蹤卡比，並且發出電場攻擊卡比，值得注意的是它會跳躍，不是跳得高高的就能高枕無憂唷，如此強大的對手與其殺掉它不如把它吃了，變身成閃電卡比吧！火焰怪會噴出火焰，讓玩家很難靠近它，但它也不能永無止盡的噴下去，把握它的攻擊間隔時間打敗它吧！當然你也能選擇吃掉它變成華麗的火焰卡比，享受火烤怪物的樂趣，火焰怪也會追蹤卡比，沒有錯我們的怪獸都是有智慧的，為了考驗玩家們的智商，太無腦是沒辦法過關的喔！。

卡比總共有三種型態，普通卡比、閃電卡比、火焰卡比，攻擊部分，普通卡比可以將怪物吸進嘴裡，吐出星星攻擊或是吞下去變身，也可以把飛行吸入的氣體吐出，作為攻擊的手段喔；閃電卡比是發出強大電場，電擊範圍內的怪物，火焰卡比則是吐出熊熊烈火，火烤就是美味。各種型態的卡比都是能使用踢擊的，玩家可以每一種都是嘗試看看，找出屬於自己的卡比吧！

## 遊戲圖形

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 開始畫面 | | | 說明 | | |
| C:\Users\宇天\AppData\Local\Microsoft\Windows\INetCache\Content.Word\開始畫面.png | | | C:\Users\宇天\AppData\Local\Microsoft\Windows\INetCache\Content.Word\說明.png | | |
| 卡比 | | | | | |
| D:\kirby\game4.10\RES\Kirby\KB_R_0.bmp | | | | | |
| 行走動畫 | | | | | |
| D:\kirby\game4.10\RES\Kirby\KB_R_9.bmpD:\kirby\game4.10\RES\Kirby\KB_R_8.bmpD:\kirby\game4.10\RES\Kirby\KB_R_7.bmpD:\kirby\game4.10\RES\Kirby\KB_R_6.bmpD:\kirby\game4.10\RES\Kirby\KB_R_5.bmpD:\kirby\game4.10\RES\Kirby\KB_R_4.bmpD:\kirby\game4.10\RES\Kirby\KB_R_3.bmpD:\kirby\game4.10\RES\Kirby\KB_R_2.bmpD:\kirby\game4.10\RES\Kirby\KB_R_1.bmpD:\kirby\game4.10\RES\Kirby\KB_R_10.bmp | | | | | |
| 跑步動畫 | | | | | |
| D:\kirby\game4.10\RES\Kirby\RUN_R_6.bmpD:\kirby\game4.10\RES\Kirby\RUN_R_5.bmpD:\kirby\game4.10\RES\Kirby\RUN_R_4.bmpD:\kirby\game4.10\RES\Kirby\RUN_R_3.bmpD:\kirby\game4.10\RES\Kirby\RUN_R_2.bmpD:\kirby\game4.10\RES\Kirby\RUN_R_1.bmpD:\kirby\game4.10\RES\Kirby\RUN_R_0.bmpD:\kirby\game4.10\RES\Kirby\RUN_R_7.bmp | | | | | |
| 翻滾動畫 | | | | | |
| D:\kirby\game4.10\RES\Kirby\KB_Hurted_R_3.bmpD:\kirby\game4.10\RES\Kirby\KB_Hurted_R_2.bmpD:\kirby\game4.10\RES\Kirby\KB_Hurted_R_1.bmpD:\kirby\game4.10\RES\Kirby\KB_Hurted_R_4.bmp | | | | | |
| 飛行動畫 | | | | | |
| D:\kirby\game4.10\RES\Kirby\KB_U_R_10.bmpD:\kirby\game4.10\RES\Kirby\KB_U_R_9.bmpD:\kirby\game4.10\RES\Kirby\KB_U_R_8.bmpD:\kirby\game4.10\RES\Kirby\KB_U_R_7.bmpD:\kirby\game4.10\RES\Kirby\KB_U_R_6.bmpD:\kirby\game4.10\RES\Kirby\KB_U_R_11.bmp | | | | | |
| 跳躍 | 落下 | 吐氣&星星 | | 踢擊 | 壓扁 |
| D:\kirby\game4.10\RES\Kirby\KB_Jump_R.bmp | D:\kirby\game4.10\RES\Kirby\KB_Landing_R.bmp | D:\kirby\game4.10\RES\Kirby\KB_Exhale_R.bmp | | D:\kirby\game4.10\RES\Kirby\KB_DownAttack_R.bmp | D:\kirby\game4.10\RES\Kirby\KB_Down_R.bmp |

|  |
| --- |
| 吸氣&吸怪動畫 |
| D:\kirby\game4.10\RES\Kirby\KB_Suck_R_3.bmpD:\kirby\game4.10\RES\Kirby\KB_Suck_R_2.bmpD:\kirby\game4.10\RES\Kirby\KB_Suck_R_1.bmpD:\kirby\game4.10\RES\Kirby\KB_Suck_R_4.bmp |
| 吸怪後的行走動畫 |
| D:\kirby\game4.10\RES\Kirby\BKB_R_7.bmpD:\kirby\game4.10\RES\Kirby\BKB_R_6.bmpD:\kirby\game4.10\RES\Kirby\BKB_R_5.bmpD:\kirby\game4.10\RES\Kirby\BKB_R_4.bmpD:\kirby\game4.10\RES\Kirby\BKB_R_3.bmpD:\kirby\game4.10\RES\Kirby\BKB_R_2.bmpD:\kirby\game4.10\RES\Kirby\BKB_R_8.bmp |
| 火焰卡比 |
| D:\kirby\game4.10\RES\FireKirby\FireKirby_Stand_R_3.bmp |
| 火焰卡比攻擊動畫 |
| D:\kirby\game4.10\RES\FireKirby\FireKirby_Attack_R_2.bmpD:\kirby\game4.10\RES\FireKirby\FireKirby_Attack_R_1.bmpD:\kirby\game4.10\RES\FireKirby\FireKirby_Attack_R_0.bmpD:\kirby\game4.10\RES\FireKirby\FireKirby_Attack_R_3.bmpD:\kirby\game4.10\RES\FireKirby\attack1_1.bmpD:\kirby\game4.10\RES\FireKirby\attack2_4.bmpD:\kirby\game4.10\RES\FireKirby\attack3_5.bmp |
| 閃電卡比 |
| D:\kirby\game4.10\RES\SparkKirby\Spark_stand_R_3.bmp |
| 閃電卡比攻擊動畫 |
| D:\kirby\game4.10\RES\SparkKirby\Spark_Attack_R_5.bmpD:\kirby\game4.10\RES\SparkKirby\Spark_Attack_R_4.bmpD:\kirby\game4.10\RES\SparkKirby\Spark_Attack_R_6.bmp |

|  |
| --- |
| 普通小怪行走動畫 |
| D:\kirby\game4.10\RES\NormalMonster\NormalMonster_R_6.bmpD:\kirby\game4.10\RES\NormalMonster\NormalMonster_R_5.bmpD:\kirby\game4.10\RES\NormalMonster\NormalMonster_R_4.bmpD:\kirby\game4.10\RES\NormalMonster\NormalMonster_R_3.bmpD:\kirby\game4.10\RES\NormalMonster\NormalMonster_R_2.bmpD:\kirby\game4.10\RES\NormalMonster\NormalMonster_R_1.bmpD:\kirby\game4.10\RES\NormalMonster\NormalMonster_R_7.bmp |
| 普通小怪被吸 |
| D:\kirby\game4.10\RES\NormalMonster\NormalMonster_Sucked_R.bmp |
| 火焰怪行走動畫 |
| D:\kirby\game4.10\RES\Fire\Fire_walk_R_4.bmpD:\kirby\game4.10\RES\Fire\Fire_walk_R_3.bmpD:\kirby\game4.10\RES\Fire\Fire_walk_R_2.bmpD:\kirby\game4.10\RES\Fire\Fire_walk_R_1.bmpD:\kirby\game4.10\RES\Fire\Fire_walk_R_0.bmpD:\kirby\game4.10\RES\Fire\Fire_walk_R_5.bmp |
| 火焰怪攻擊動畫 |
| D:\kirby\game4.10\RES\Fire\Fire_Attack_R_0.bmpD:\kirby\game4.10\RES\Fire\Fire_Attack_R_1.bmpD:\kirby\game4.10\RES\Fire\Fire_Attack_R_2.bmp |
| 火焰怪被吸動畫 |
| D:\kirby\game4.10\RES\Fire\Fire_sucked_L_0.bmpD:\kirby\game4.10\RES\Fire\Fire_sucked_L_1.bmp |
| 閃電怪跳躍動畫 |
| D:\kirby\game4.10\RES\Spirky\Spirky_jump_R_0.bmpD:\kirby\game4.10\RES\Spirky\Spirky_jump_R_1.bmpD:\kirby\game4.10\RES\Spirky\Spirky_jump_R_2.bmpD:\kirby\game4.10\RES\Spirky\Spirky_jump_R_3.bmpD:\kirby\game4.10\RES\Spirky\Spirky_jump_R_4.bmpD:\kirby\game4.10\RES\Spirky\Spirky_jump_R_5.bmpD:\kirby\game4.10\RES\Spirky\Spirky_jump_R_6.bmpD:\kirby\game4.10\RES\Spirky\Spirky_jump_R_7.bmp |
| 閃電怪攻擊動畫 |
| D:\kirby\game4.10\RES\Spirky\Spirky_Attack_8.bmpD:\kirby\game4.10\RES\Spirky\Spirky_Attack_6.bmpD:\kirby\game4.10\RES\Spirky\Spirky_Attack_14.bmpD:\kirby\game4.10\RES\Spirky\Spirky_Attack_17.bmp |
| 閃電怪被吸動畫 |
| D:\kirby\game4.10\RES\Spirky\Spirky_Sucked_R_0.pngD:\kirby\game4.10\RES\Spirky\Spirky_Sucked_R_1.png |

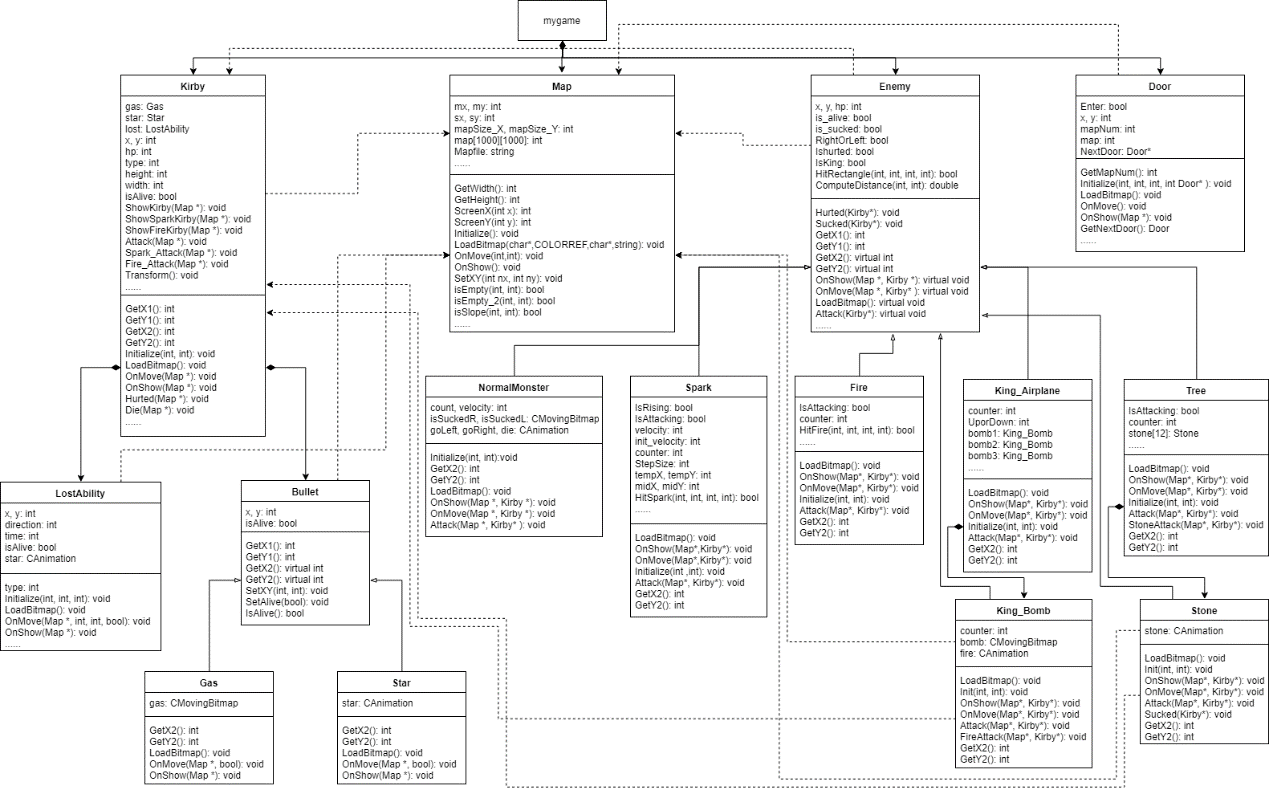
|  |  |
| --- | --- |
| 飛機王飛行動畫 | |
| D:\kirby\game4.10\RES\Bombar\Bombar_goR_3.bmpD:\kirby\game4.10\RES\Bombar\Bombar_goR_4.bmpD:\kirby\game4.10\RES\Bombar\Bombar_goR_6.bmpD:\kirby\game4.10\RES\Bombar\Bombar_goR_7.bmpD:\kirby\game4.10\RES\Bombar\Bombar_goR_9.bmpD:\kirby\game4.10\RES\Bombar\Bombar_goR_11.bmp | |
| 塔王 | |
| SD:\kirby\截圖\未命名.png | |
| 初始地圖 | 破關後的地圖 |
|  |  |
| 過關動畫 | |
|  | |

## 遊戲音效

|  |  |
| --- | --- |
| 音效 | 說明 |
| die.wav | 卡比死亡音效 |
| ending.mp3 | 結束動畫背景音樂 |
| enemyDie.wav | 怪物死亡音效 |
| fire.wav | 火焰攻擊音效 |
| fly.wav | 飛行音效 |
| gas.wav | 吐氣音效 |
| hurted.wav | 受到傷害的音效 |
| jump.wav | 跳躍音效 |
| kick.wav | 踢擊音效 |
| Kirby\_background.mp3 | 遊戲時的背景音樂 |
| landing.wav | 落地音效 |
| lostAbility.wav | 失去能力時的音效 |
| run.wav | 跑步音效 |
| spark.wav | 閃電攻擊音效 |
| star.wav | 星星撞擊時的音效 |
| start.mp3 | 開始畫面的背景音樂 |
| suck.wav | 吸怪時的音效 |
| swallow.wav | 吞怪時的音效 |

# 程式設計

## 程式架構



## 程式類別

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| mygame | 106 | 658 | 整個遊戲流程 |
| Bullet | 19 | 49 | 發射物的基礎內容(父類別) |
| Door | 43 | 118 | 傳送門 |
| Enemy | 54 | 137 | 怪物的基本內容(父類別) |
| Fire | 28 | 235 | 火焰怪，繼承Enemy |
| Gas | 18 | 48 | 空氣砲，繼承Bullet |
| King\_Airplane | 26 | 175 | 飛機王，繼承Enemy |
| King\_Bomb | 24 | 87 | 飛機王的炸彈，繼承Enemy |
| King\_Stone | 23 | 100 | 塔王的石頭，繼承Enemy |
| Kirby | 133 | 1726 | 卡比的所有行為(變身、攻擊…) |
| LostAbility | 46 | 113 | 卡比受傷時失去的變身能力 |
| Map | 39 | 197 | 地圖 |
| NormalMonster | 24 | 163 | 普通小怪，繼承Enemy |
| Spark | 35 | 261 | 閃電怪，繼承Enemy |
| Star | 17 | 54 | 卡比吐出的星星，繼承Bullet |
| Tree | 32 | 128 | 塔王，繼承Enemy |

## 程式技術

我們的遊戲由Kirby、Enemy、Map這三個主要的物件交織而成，運用不少繼承與多型來擴充程式功能，並針對需求增加新物件供主要的物件使用，各功能主要由大量的判斷式與迴圈來完成，對於每種情況都進行嚴格的檢測。

Kirby部分，原先的計畫是將各種卡比切割成多個物件，但考慮到建構、解構與資源佔用的問題，決定將卡比結合在一個物件，並新增LostAbility、Bullet兩個物件，還有Gas與Star，兩者繼承Bullet並運用多型形成不同的攻擊方式與傷害。

Enemy是每隻怪物的基礎，包含兩隻王與它們丟出的石頭或飛彈，同樣運用繼承與多型，每隻怪物在基礎的function中再延伸出各自的功能，並且可以透過一個Enemy指標陣列進行統一控管。

Map是貫穿整個遊戲的物件，與地圖或是位子相關的判斷都透過Map來完成，運用1000\*1000的陣列儲存地圖的地形資訊，可以將地形資訊設定到幾乎吻合我們的地圖，此外我們有設計一個設定地形資訊的function，用滑鼠點擊遊戲畫面就能設定，並且將地形資訊儲存到各自的文件檔，切換地圖時就能馬上載入新的地形資訊，每張地圖透過Door物件連通，Door物件將地圖做轉換，並實作轉場動畫讓地圖轉換更流暢，不會顯得突兀。

# 結語

## 問題及解決方法

(1)斜坡的判斷式不夠精準

解決方法：由於地圖系統是多個正方形組成的關係，沒辦法很精準地完成斜坡功能，起初試了許多種判斷式，都沒有辦法克服，最終只好重新規劃地形資料，以達到幾乎精準的斜坡判定。

(2)不知道是否該將不同型態的卡比切割成多個物件

解決方法：經過仔細思考之後，我們決定不切割卡比，一方面避免建構與解構時發生問題，另一方面也可以減少不必要的程式複雜度，因為卡比本來就是單一的物件，即便他變身了還是卡比。

(3)動態配置的結構不穩定

解決方案：改用靜態的陣列處理動態配置的物件，雖然耗費效能，但能有效解決動態配置不穩定的問題。

(4)音效重複讀取或讀取太慢導致播放時跳例外

解決方案：充分了解framework功能後，修改CAudio中的部分程式碼，以符合我們自己程式的需求。

(5)企劃不夠周詳

解決方案：針對負責的項目與完成先後順序做更詳細的討論，避免彼此衝突或是資訊不流通的狀況。

(6)程式撰寫規劃不足

解決方案：在開始設計程式前先進行功能規劃，以免更改程式時造成程式邏輯的前後矛盾。

(7)大量檔案的引入

解決辦法：一開始對圖片的引入是手動在resource裡建立新的點陣圖，但在後期處理大量圖檔時太過耗時且無法直接對圖檔進行更動，後來改以圖檔的相對位置開啟。

(8)素材來源格式不一

解決辦法：一開始的人物與地圖的比例非常奇怪，後來篩選圖片的來源並對圖檔統一放大。

## 時間表

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 週 | 每週工作時間(小時) | | 總和 | 工作規劃 |
| 周宇天 | 林彥廷 |
| 0 | 3 | 3 | 6 | Tutorial |
| 1 | 4 | 14 | 18 | 完成地圖 |
| 2 | 5.5 | 14 | 19.5 | 角色在地圖上移動 |
| 3 | 4.5 | 2 | 6.5 | 怪物與卡比的互動 |
| 4 | 6.5 | 11 | 17.5 | 攻擊怪物與被攻擊 |
| 5 | 5 | 2 | 7 | 完成傳送門 |
| 6 | 3.5 | 4 | 7.5 | 吃怪、吐星星、吐氣…. |
| 7 | 7.5 | 7 | 14.5 | 新增怪物與卡比變身 |
| 8 | 5 | 7 | 12 | 第二關卡完成 |
| 9 | 8.5 | 9 | 17.5 | 變身火焰卡比與魔王關卡完成 |
| 10 | 2 | 3 | 5 | 新增音效 |
| 11 | 3 | 5 | 8 | 增加關卡、關卡難度與複雜度 |
| 12 | 4 | 6.5 | 10.5 | 製作過關動畫&通關方式 |
| 13 | 3 | 5 | 8 | 製作報告 |
| 14 | 10 | 5 | 15 | 製作報告 |
| 總和 | **75** | **97.5** | **172.5** |  |

## 貢獻比例

林彥廷：５０％

周宇天：５０％

## 檢核表

|  |  |  |
| --- | --- | --- |
|  | 項目 | 完成與否 |
| 1 | 解決Memory leak | ☑已完成 ☐未完成 |
| 2 | 自定遊戲Icon | ☑已完成 ☐未完成 |
| 3 | 全螢幕啟動 | ☑已完成 ☐未完成 |
| 4 | 修改Help->About | ☑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 檔 | ☑已完成 ☐未完成 |
| 7 | 報告字型、點數、對齊、行距、頁碼等格式正確 | ☑已完成 ☐未完成 |
| 8 | 報告封面、側邊格式正確 | ☑已完成 ☐未完成 |

## 收穫

林彥廷：

經過這一個學期的歷練，讓我對物件導向的概念有更多體悟，也有更多的程式撰寫經歷，對於程式的架構也有更深入的了解，由於這是第一次寫這種較大型的專案，有了這次經驗以後，相信再面對這類型的專案時，我就不會太畏懼，也能更有自信的估計耗費的時間或資源，此外這也是我第一次運用這麼多繼承與多型，經過多次失敗與搜尋，讓我更清楚如何正確地使用繼承與多型，來簡化複雜的物件關係或豐富程式功能，本次課程讓我學到最多的是團隊合作，有過各式各樣的合作經驗，合作完成一個較大型的專案倒是第一次，了解到合作完成一個程式的所有過程，也更知道如何和隊友清楚表達自己的程式或需求，表達正是我所欠缺的能力，這是個珍貴的練習經驗。

周宇天：

這學期的實作再次加深的我對物件導向的觀念與能力，專案裡大量使用了繼承與多型，不再像以前是以練習語法為目的的作業，這次我們扎扎實實的練習了編寫大型專案的感覺與技巧。不僅如此，透過大量閱讀別人的code了解程式的脈絡與結構，讓我有能力對程式進行優化與改寫，以別人的程式為基礎做出自己需要的功能。最重要的是在這個專案學會了與夥伴溝通合作，單憑一個人是很難做出一個完善的作品，不單單是因為每個人擅長的方面不同，憑一己之力是很難做出像樣的作品，我們都站在巨人的肩膀上，使用了別人的心血結晶，使用了老師的framework為基礎才能完成。

## 心得、感想

林彥廷：

實作的過程中經歷了許多挫折與失敗，一次又一次的修改，一遍又一遍的尋找資料，正所謂失敗為成功之母，每次挫敗都讓我學到更多，像是繼承與多型的部分，就讓我收穫很多，但有些部分直到最後還是沒有辦法完成，像是動態生成一些非常駐的物件，起初空氣砲與星星的部分我是打算用動態完成，但卡在一些觀念不完整，且時間也不足夠，最後只好用比較不理想的方法完成這些功能，這是讓我比較遺憾的，但也讓我更清楚自己有哪部分需要加強，整體來說，這個專案我很滿意，雖然成品還是有些小缺失，但整個過程讓我成長許多，程式邏輯的思緒也更清晰，表達能力也有加強，感謝辛苦的老師與助教，帶我們完成這個專案，更感謝我的夥伴，讓我有這麼棒的合作經驗。

周宇天：

以前作業大部分都是自己一個人完成，但這次的實作讓我有機會能與夥伴合作一起完成專案，讓我初步認識該怎麼與人合作一同撰寫程式。在與其他人合作時我覺得最困難的地方在於溝通與閱讀，我們時常會無法理解對方需要的程式功能就是因為溝通不良所導致；而在編寫與維護程式時時常需要繼承或改寫別人的程式碼，如果註解與說明不夠清楚與明確將會造成很大的麻煩，這部分我應要改進，不要因為懶惰而造成夥伴的困擾。這學期我們從無到有做出了令自己滿意的作品，體驗了北科誠樸精勤實作精神。

感謝這學期教授與助教的用心教導

感謝夥伴的包容與忍耐

## 對於本課程的建議

林彥廷：

希望在Demo的時候可以將畫面廣播到台下的電腦上，讓台下的同學能看得更清楚，如果Android不行，至少Windows、HTML的可以。

周宇天：

希望能讓Demo的流程與效率增加，以及讓git更穩定不然網站上的專案進度、編輯次數等功能就這麼白費了。

# 附錄

## Mygame.h

#include "Kirby.h"

#include "Map.h"

#include "NormalMonster.h"

#include "Door.h"

#include "Gas.h"

#include "Star.h"

#include "LostAbility.h"

#include "Spark.h"

#include "Fire.h"

#include "Tree.h"

#include "King\_Airplane.h"

namespace game\_framework {

enum AUDIO\_ID { // 定義各種音效的編號

AUDIO\_BACKGROUND, // 0

jump, // 1

landing, // 2

kick, // 3

die, // 4

fly, // 5

hurted, // 6

run, // 7

suck, // 8

gasSound, // 9

starSound, // 10

swallow, // 11

spark, // 12

fire, // 13

stone, // 14

start, // 15

enemyDie, // 16

lostAbility, 　// 17

ending // 18

};

class CGameStateInit : public CGameState {

public:

CGameStateInit(CGame \*g);

void OnInit(); // 遊戲的初值及圖形設定

void OnBeginState(); // 設定每次重玩所需的變數

void OnKeyUp(UINT, UINT, UINT); // 處理鍵盤Up的動作

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

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int page;

CAnimation StartAnimation;

CMovingBitmap background, pressStart;

};

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 OnMouseMove(UINT nFlags, CPoint point); // 處理滑鼠的動作

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 OnRButtonDown(UINT nFlags, CPoint point);// 處理滑鼠的動作

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

void ResetMonster();

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面 　　　private:

Kirby kirby; // 卡比

Map map[5]; // 地圖

Map \*index;

Enemy \*monster[10]; // 怪物

Enemy \*Boss;

NormalMonster normalMonster1[5],normalMonster4[1];

Spark spark1[4],spark4[1];

Fire fire1[2], fire4[1];

Tree tree;

King\_Airplane airplane;

Door door[10];

Door door1[2];

Door door2;

Door door3;

Door door4[2];

Door \*gate;

int mapNum; //設定現在為第幾號地圖

int Mirror\_L\_Y,Mirror\_R\_Y;

int end\_Y;

CAnimation Transition;

CAnimation Mirror\_L, Mirror\_R;

CMovingBitmap end;

bool Istransiting = false;

bool MovingMirror = false;

bool Show\_Mirror\_L = false;

bool Show\_Mirror\_R = false;

bool isEnd = false;

};

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:

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 {

CGameStateInit::CGameStateInit(CGame \*g): CGameState(g)

{

}

void CGameStateInit::OnInit()

{

StartAnimation.AddBitmap(".\\RES\\Start\\title\_0.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_1.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_2.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_3.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_4.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_5.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_6.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_7.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_8.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_9.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_10.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_11.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_12.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_13.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_14.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_15.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_16.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_17.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_18.bmp", RGB(255, 255, 255));

StartAnimation.AddBitmap(".\\RES\\Start\\title\_19.bmp", RGB(255, 255, 255));

background.LoadBitmap(".\\RES\\Start\\background.bmp");

pressStart.LoadBitmap(".\\RES\\Start\\Press\_start.bmp", RGB(248, 248, 248));

}

void CGameStateInit::OnBeginState()

{

page = 0;

if(!CAudio::Instance()->IsLoaded(start))

CAudio::Instance()->Load(start, "Sounds\\start.mp3");

CAudio::Instance()->Play(start, true);

}

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

{

const char KEY\_ESC = 27;

const char KEY\_SPACE = ' ';

if (nChar == KEY\_SPACE)

{

if (page == 0)

page++;

else

{

CAudio::Instance()->Stop(start);

GotoGameState(GAME\_STATE\_RUN); // 切換至GAME\_STATE\_RUN

}

}

else 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::OnLButtonDown(UINT nFlags, CPoint point)

{

if (page == 0)

page++;

else

{

CAudio::Instance()->Stop(start);

GotoGameState(GAME\_STATE\_RUN); // 切換至GAME\_STATE\_RUN

}

}

void CGameStateInit::OnShow()

{

if (page == 0)

{

background.SetTopLeft(0, 0);

StartAnimation.SetTopLeft(0, 0);

pressStart.SetTopLeft(60, 380);

background.ShowBitmap();

StartAnimation.OnMove();

StartAnimation.OnShow();

pressStart.ShowBitmap();

}

else

{

// /\*Demo螢幕字型的使用，不過開發時請盡量避免直接使用字型，改用CMovingBitmap比較好\*/

CDC \*pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f,\*fp;

f.CreatePointFont(160,"Times New Roman"); // 產生 font f; 160表示16 point的字

fp=pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0, 0, 0));

pDC->SetTextColor(RGB(255 , 255, 0));

pDC->TextOut(35, 50, "操作 : ");

pDC->TextOut(65, 80, "↑ : 吸氣飛翔、進入傳送門 ↓ : 蹲下、變身");

pDC->TextOut(65, 110, "← : 左移 → : 右移");

pDC->TextOut(65, 140, "Z : 攻擊、吸怪 ↓ + Z : 踢擊");

pDC->TextOut(65, 170, "X : 跳躍");

pDC->TextOut(65, 200, "C : 跑步(按住加左右移)");

pDC->TextOut(65, 230, "Esc : 關閉遊戲");

pDC->TextOut(35, 260, "密技 : ");

pDC->TextOut(65, 290, "S : 血量補滿");

pDC->TextOut(35, 350, "打敗兩隻魔王，拿回碎片拯救世界吧 ! 卡比");

pDC->TextOut(220, 380, "(按下空白鍵開始)");

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC\*/

}

}

CGameStateOver::CGameStateOver(CGame \*g): CGameState(g)

{

}

void CGameStateOver::OnMove()

{

if(counter==0)

GotoGameState(GAME\_STATE\_INIT);

}

void CGameStateOver::OnBeginState()

{

counter = 300;

}

void CGameStateOver::OnInit()

{

}

void CGameStateOver::OnShow()

{

counter--;

CDC \*pDC = CDDraw::GetBackCDC(); // 取得 Back Plain 的 CDC

CFont f,\*fp;

f.CreatePointFont(160,"Times New Roman"); 　　　// 產生 font f; 160表示16 point的字

fp=pDC->SelectObject(&f); // 選用 font f

pDC->SetBkColor(RGB(0,0,0));

pDC->SetTextColor(RGB(255,255,0));

char str[80]; // Demo 數字對字串的轉換

sprintf(str, "Game Over ! (%d)",counter/30);

pDC->TextOut(240,210,str);

pDC->SelectObject(fp); // 放掉 font f (千萬不要漏了放掉)

CDDraw::ReleaseBackCDC(); // 放掉 Back Plain 的 CDC

}

CGameStateRun::CGameStateRun(CGame \*g): CGameState(g)

{

}

CGameStateRun::~CGameStateRun()

{

}

void CGameStateRun::OnBeginState()

{

map[0].Initialize();

map[1].Initialize();

map[2].Initialize();

map[3].Initialize();

map[4].Initialize();

Mirror\_L.SetTopLeft(608, -100);

Mirror\_R.SetTopLeft(608, -100);

kirby.Initialize(640,400);

door[0].Initialize(123, 37, 1, 0, &door1[0]);

door[1].Initialize(621, 37, 2, 0, &door2);

door[2].Initialize(1118, 37, 4, 0, &door4[0]);

door[3].Initialize(83, 328, 4, 0, &door4[1]);

door[4].Initialize(248, 369, 4, 0, &door4[0]);

door[5].Initialize(993, 367, 1, 0, &door1[0]);

door[6].Initialize(1159, 326, 1, 0, &door1[1]);

door[7].Initialize(207, 575, 1, 0, &door1[1]);

door[8].Initialize(620, 614, 3, 0, &door3);

door[9].Initialize(1036, 574, 4, 0, &door4[1]);

door1[0].Initialize(30, 425, 0,1, &door[5]);

door1[1].Initialize(4450, 350, 0, 1, &door[6]);

door2.Initialize(320, 240, 0, 2, &door[1]);

door3.Initialize(150, 350, 0, 3, &door[8]);

door4[0].Initialize(48, 435, 0, 4, &door[4]);

door4[1].Initialize(3140, 240 , 0, 4, &door[3]);

ResetMonster();

tree.Initialize(450, 100);

airplane.Initialize(450, 60);

Mirror\_L\_Y = -10;

Mirror\_R\_Y = -10;

end\_Y = 0;

mapNum = 0;

index = &map[mapNum];

gate = &door[5];

CAudio::Instance()->Play(AUDIO\_BACKGROUND, true);

}

void CGameStateRun::OnMove() // 移動遊戲元素

{

index->OnMove(kirby.GetX1(), kirby.GetY1());

Mirror\_R.OnMove();

Mirror\_L.OnMove();

if (Istransiting) {

Transition.OnMove();

Transition.SetDelayCount(4);

}

else if (MovingMirror) {

map[0].SetXY(320, 160);

if (Show\_Mirror\_R && Mirror\_R\_Y != 327) {

Mirror\_R\_Y += 1;

Mirror\_R.SetTopLeft(map[0].ScreenX(608), map[0].ScreenY(Mirror\_R\_Y));

}

else if (Show\_Mirror\_L && Mirror\_L\_Y != 327 ) {

Mirror\_L\_Y += 1;

Mirror\_L.SetTopLeft(map[0].ScreenX(608), map[0].ScreenY(Mirror\_L\_Y));

}

else {

MovingMirror = false;

}

}

else if (isEnd)

{

end\_Y--;

if (end\_Y == -1)

{

CAudio::Instance()->Stop(AUDIO\_BACKGROUND);

CAudio::Instance()->Play(ending);

}

else if (end\_Y == -1050)

{

CAudio::Instance()->Stop(ending);

CAudio::Instance()->Play(AUDIO\_BACKGROUND);

isEnd = false;

end\_Y = 1;

}

}

else {

if (kirby.IsAlive())

kirby.OnMove(index);

if (mapNum == 0)

{

for (int i = 0; i < 10; i++) {

door[i].OnMove();

if (door[i].IsEnter(&kirby)) {

door[i].SetEnter(false);

Istransiting = true;

Transition.Reset();

gate = &door[i];

}

}

}

if (mapNum == 1) {

for (int m = 0; m < 10; m++)

monster[m]->OnMove(index, &kirby);

for (int i = 0; i < 2; i++) {

door1[i].OnMove();

if (door1[i].IsEnter(&kirby)) {

door1[i].SetEnter(false);

Istransiting = true;

Transition.Reset();

gate = &door1[i];

}

}

}

if (mapNum == 2) {

airplane.OnMove(index,&kirby);

if (!airplane.IsAlive())

{

if (!Show\_Mirror\_R) {

MovingMirror = true;

Show\_Mirror\_R = true;

Istransiting = true;

Transition.Reset();

Mirror\_L.Reset();

Mirror\_R.Reset();

}

else {

door2.OnMove();

if (door2.IsEnter(&kirby)) {

door2.SetEnter(false);

Istransiting = true;

Transition.Reset();

gate = &door2;

}

}

}

}

if (mapNum == 3) {

tree.OnMove(index,&kirby);

if (!tree.IsAlive()) {

if (!Show\_Mirror\_L) {

MovingMirror = true;

Show\_Mirror\_L = true;

Istransiting = true;

Transition.Reset();

}

else {

door3.OnMove();

if (door3.IsEnter(&kirby)) {

door3.SetEnter(false);

Istransiting = true;

Transition.Reset();

gate = &door3;

}

}

}

}

if (mapNum == 4) {

door4[0].OnMove();

door4[1].OnMove();

for (int m = 0; m < 3; m++)

monster[m]->OnMove(index, &kirby);

for (int i = 0; i < 2; i++) {

if (door4[i].IsEnter(&kirby)) {

door4[i].SetEnter(false);

Istransiting = true;

Transition.Reset();

gate = &door4[i];

}

}

}

}

if (Transition.IsFinalBitmap()) {

Istransiting = false;

}

if (Transition.GetCurrentBitmapNumber() == 7 && !MovingMirror) {

mapNum = gate->GetMapNum();

index = &map[mapNum];

ResetMonster();

kirby.SetXY(gate->GetNextDoor().GetX() - 50, gate->GetNextDoor().GetY2()-kirby.GetHeight());

}

}

void CGameStateRun::OnInit() // 遊戲的初值及圖形設定

{

Transition.AddBitmap(".//Map//Transition\_7.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_6.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_5.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_4.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_3.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_2.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_1.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_0.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_1.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_2.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_3.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_4.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_5.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_6.bmp", RGB(0, 0, 0));

Transition.AddBitmap(".//Map//Transition\_7.bmp", RGB(0, 0, 0));

Mirror\_L.AddBitmap(".//RES//Door//mirror\_L\_0.bmp", RGB(255, 255, 255));

Mirror\_L.AddBitmap(".//RES//Door//mirror\_L\_1.bmp", RGB(255, 255, 255));

Mirror\_L.AddBitmap(".//RES//Door//mirror\_L\_2.bmp", RGB(255, 255, 255));

Mirror\_L.AddBitmap(".//RES//Door//mirror\_L\_3.bmp", RGB(255, 255, 255));

Mirror\_R.AddBitmap(".//RES//Door//mirror\_R\_0.bmp", RGB(255, 255, 255));

Mirror\_R.AddBitmap(".//RES//Door//mirror\_R\_1.bmp", RGB(255, 255, 255));

Mirror\_R.AddBitmap(".//RES//Door//mirror\_R\_2.bmp", RGB(255, 255, 255));

Mirror\_R.AddBitmap(".//RES//Door//mirror\_R\_3.bmp", RGB(255, 255, 255));

end.LoadBitmap(".//RES//end.bmp");

map[0].LoadBitmap(".//Map//foreground.bmp", RGB(255, 255, 255), ".//Map//background.bmp", ".//Map//map.txt");

map[1].LoadBitmap(".//Map//map1.bmp", RGB(255, 255, 255), ".//Map//background\_1.bmp", ".//Map//map1.txt");

map[2].LoadBitmap(".//Map//Boss\_map.bmp", RGB(255, 255,255),".//Map//background\_2.bmp", ".//Map//map2.txt");

map[3].LoadBitmap(".//Map//King\_foreground.bmp",RGB(255,255,255),".//Map//King\_background.bmp",".//Map//map3.txt");

map[4].LoadBitmap(".//Map//foreground\_4.bmp",RGB(255,255,255),".//Map//background\_4.bmp",".//Map//map4.txt");

kirby.LoadBitmap();

monster[0] = &fire1[0];

monster[1] = &fire1[1];

monster[2] = &normalMonster1[0];

monster[3] = &normalMonster1[1];

monster[4] = &normalMonster1[2];

monster[5] = &normalMonster1[3];

monster[6] = &spark1[0];

monster[7] = &spark1[1];

monster[8] = &spark1[2];

monster[9] = &spark1[3];

fire4[0].LoadBitmap();

spark4[0].LoadBitmap();

normalMonster4[0].LoadBitmap();

for (int m = 0; m < 10; m++)

monster[m]->LoadBitmap();

for (int i = 0; i < 10;i++)

door[i].LoadBitmap();

door1[0].LoadBitmap();

door1[1].LoadBitmap();

door2.LoadBitmap();

door3.LoadBitmap();

for (int i = 0; i < 2; i++)

door4[i].LoadBitmap();

tree.LoadBitmap();

airplane.LoadBitmap();

CAudio::Instance()->Load(AUDIO\_BACKGROUND, "Sounds\\Kirby\_background.mp3"); //背景音樂

CAudio::Instance()->Load(jump, "Sounds\\jump.wav");

CAudio::Instance()->Load(landing, "Sounds\\landing.wav");

CAudio::Instance()->Load(kick, "Sounds\\kick.wav");

CAudio::Instance()->Load(die, "Sounds\\die.wav");

CAudio::Instance()->Load(fly, "Sounds\\fly.wav");

CAudio::Instance()->Load(hurted, "Sounds\\hurted.wav");

CAudio::Instance()->Load(run, "Sounds\\run.wav");

CAudio::Instance()->Load(suck, "Sounds\\suck.wav");

CAudio::Instance()->Load(gasSound, "Sounds\\gas.wav");

CAudio::Instance()->Load(starSound, "Sounds\\star.wav");

CAudio::Instance()->Load(swallow, "Sounds\\swallow.wav");

CAudio::Instance()->Load(spark, "Sounds\\spark.wav");

CAudio::Instance()->Load(fire, "Sounds\\fire.wav");

CAudio::Instance()->Load(enemyDie, "Sounds\\enemyDie.wav");

CAudio::Instance()->Load(lostAbility, "Sounds\\lostAbility.wav");

CAudio::Instance()->Load(ending, "Sounds\\ending.mp3");

}

void CGameStateRun::OnKeyDown(UINT nChar, UINT nRepCnt, UINT nFlags)

{

const char KEY\_ESC = 27; // keyboard Esc

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\_Jump = 0x58; // keyboard X鍵

const char KEY\_Attack = 0x5A; 　// keyboard Z鍵

const char KEY\_Run = 0x43; // keyboard C鍵

const char KEY\_Restore = 0x53; // keyboard S鍵

if (nChar == KEY\_LEFT)

kirby.SetMovingLeft(true);

if (nChar == KEY\_RIGHT)

kirby.SetMovingRight(true);

if (nChar == KEY\_UP) {

kirby.SetMovingUp(true);

if(mapNum==0)

for (int i = 0; i < 10;i++)

door[i].SetEnter(true);

if (mapNum == 1)

for (int i = 0; i < 2;i++)

door1[i].SetEnter(true);

if (mapNum == 2)

door2.SetEnter(true);

if (mapNum == 3)

door3.SetEnter(true);

if (mapNum == 4)

for (int i = 0; i < 2; i++)

door4[i].SetEnter(true);

}

if (nChar == KEY\_DOWN)

{

kirby.SetMovingDown(true);

if (kirby.IsBig())

CAudio::Instance()->Play(swallow);

}

if (nChar == KEY\_Attack)

{

kirby.SetAttack(true);

if (!MovingMirror)

{

if (kirby.IsFly())

CAudio::Instance()->Play(gasSound);

else if (kirby.IsDown())

CAudio::Instance()->Play(kick);

else if (kirby.GetType() == 0 && !kirby.IsBig())

CAudio::Instance()->Play(suck);

else if (kirby.GetType() == 0 && kirby.IsBig())

CAudio::Instance()->Play(starSound);

else if (kirby.GetType() == 1)

CAudio::Instance()->Play(spark);

else if (kirby.GetType() == 2)

CAudio::Instance()->Play(fire);

}

}

if (nChar == KEY\_Run)

{

kirby.SetRun(true);

if (!kirby.IsFly()&&!index->isEmpty((kirby.GetX1()+kirby.GetX2())/2, kirby.GetY2() + 1) && kirby.IsMove())

CAudio::Instance()->Play(run);

}

if (nChar == KEY\_Jump && !kirby.IsFly() && (!index->isEmpty(kirby.GetX1(), kirby.GetY2() + 1) || !index->isEmpty(kirby.GetX2(), kirby.GetY2() + 1))) //按下X,卡比不是在飛行且落地才可跳躍

kirby.SetJump(true);

if (nChar == KEY\_ESC)

PostMessage(AfxGetMainWnd()->m\_hWnd, WM\_CLOSE, 0, 0); // 關閉遊戲

if (nChar == KEY\_Restore)

kirby.Restore();

}

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\_Attack = 0x5A; 　 // keyboard Z鍵

const char KEY\_Run = 0x43; // keyboard C鍵

if (nChar == KEY\_LEFT)

kirby.SetMovingLeft(false);

if (nChar == KEY\_RIGHT)

kirby.SetMovingRight(false);

if (nChar == KEY\_UP) {

kirby.SetMovingUp(false);

if(mapNum==0)

for (int i = 0; i < 10;i++)

door[i].SetEnter(false);

if (mapNum == 1)

for (int i = 0; i < 2; i++)

door1[i].SetEnter(false);

if (mapNum == 2)

door2.SetEnter(false);

if (mapNum==3)

door3.SetEnter(false);

if (mapNum == 4)

for (int i = 0; i < 2; i++)

door4[i].SetEnter(false);

}

if (nChar == KEY\_DOWN)

kirby.SetMovingDown(false);

if (nChar == KEY\_Attack)

{

kirby.SetAttack(false);

kirby.SetSuck(false);

CAudio::Instance()->Stop(suck);

CAudio::Instance()->Stop(spark);

CAudio::Instance()->Stop(fire);

}

if (nChar == KEY\_Run)

kirby.SetRun(false);

}

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

index->SetMouse(point.x, point.y);

}

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

{

index->IsLclick(true);

}

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

{

index->IsLclick(false);

}

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

{

index->IsRclick(true);

}

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

{

index->IsRclick(false);

}

void CGameStateRun::ResetMonster()

{

if (mapNum == 1) {

normalMonster1[0].Initialize(947, 467);

normalMonster1[1].Initialize(2459, 517);

normalMonster1[2].Initialize(2903, 517);

normalMonster1[3].Initialize(3495, 59);

spark1[0].Initialize(675, 393);

spark1[1].Initialize(1673, 163);

spark1[2].Initialize(3247, 269);

spark1[3].Initialize(4045, 477);

fire1[0].Initialize(417, 467);

fire1[1].Initialize(3745, 477);

monster[0] = &fire1[0];

monster[1] = &fire1[1];

monster[2] = &normalMonster1[0];

monster[3] = &normalMonster1[1];

monster[4] = &normalMonster1[2];

monster[5] = &normalMonster1[3];

monster[6] = &spark1[0];

monster[7] = &spark1[1];

monster[8] = &spark1[2];

monster[9] = &spark1[3];

}

if (mapNum == 4) {

fire4[0].Initialize(410, 400);

spark4[0].Initialize(1125, 525);

normalMonster4[0].Initialize(1600, 525);

monster[0] = &fire4[0];

monster[1] = &spark4[0];

monster[2] = &normalMonster4[0];

}

}

void CGameStateRun::OnShow()

{

end.SetTopLeft(0, end\_Y);

index->OnShow();

if (mapNum == 0) {

for (int i = 0; i < 10;i++)

door[i].OnShow(index);

if (Show\_Mirror\_L) {

Mirror\_L.SetTopLeft(map[0].ScreenX(608), map[0].ScreenY(Mirror\_L\_Y));

Mirror\_L.OnShow();

}

if (Show\_Mirror\_R) {

Mirror\_R.SetTopLeft(map[0].ScreenX(608), map[0].ScreenY(Mirror\_R\_Y));

Mirror\_R.OnShow();

}

}

else if (mapNum == 1) {

for (int i = 0; i < 2;i++)

door1[i].OnShow(index);

for (int m = 0; m < 10; m++)

monster[m]->OnShow(index, &kirby);

}

else if (mapNum == 2) {

airplane.OnShow(index,&kirby);

if(!airplane.IsAlive())

door2.OnShow(index);

}

else if (mapNum == 3) {

tree.OnShow(index,&kirby);

if (!tree.IsAlive()) {

door3.OnShow(index);

}

}

else if(mapNum == 4) {

for (int i = 0; i < 2; i++)

door4[i].OnShow(index);

for (int m = 0; m < 3; m++)

monster[m]->OnShow(index, &kirby);

}

if (kirby.IsAlive()) {

kirby.OnShow(index);

}

else if (kirby.GetY1() <= 1)

{

// 卡比死掉就不用播消失能力的音效

CAudio::Instance()->Stop(lostAbility);

// 鏡子重置

Show\_Mirror\_L = false;

Show\_Mirror\_R = false;

Mirror\_L\_Y = -10;

Mirror\_R\_Y = -10;

GotoGameState(GAME\_STATE\_OVER);

}

else

kirby.Die(index);

if (MovingMirror && Transition.GetCurrentBitmapNumber()>6) {

map[0].SetXY(320, 160);

map[0].OnShow();

if (Show\_Mirror\_L) {

Mirror\_L.SetTopLeft(map[0].ScreenX(608), map[0].ScreenY(Mirror\_L\_Y));

Mirror\_L.OnShow();

}

if (Show\_Mirror\_R) {

Mirror\_R.SetTopLeft(map[0].ScreenX(608), map[0].ScreenY(Mirror\_R\_Y));

Mirror\_R.OnShow();

}

}

if (Istransiting) {

Transition.OnShow();

}

if (Show\_Mirror\_L && Show\_Mirror\_R && Mirror\_L\_Y == 327 && Mirror\_R\_Y == 327 && end\_Y == 0)

isEnd = true;

if (isEnd)

end.ShowBitmap();

}

}

## Kirby.h

#pragma once

#include "Map.h"

#include "Gas.h"

#include "Star.h"

#include "LostAbility.h"

namespace game\_framework {

class Kirby

{

public:

Kirby();

int GetX1(); // 左上角 x 座標

int GetY1(); // 左上角 y 座標

int GetX2(); // 右下角 x 座標

int GetY2(); // 右下角 y 座標

int GetType(); // 取得當前型態

int GetHeight(); // 取得身高

int GetWidth(); // 取得寬度

void Initialize(int ,int); // 設定為初始值

bool IsAlive(); 　　　 // 是否活著

bool IsFly(); // 是否在飛

bool IsKick(); // 是否踢擊

bool IsSuck(); // 是否吸怪

bool IsAttack(); // 是否按Z鍵

bool IsRight(); // 面對方向

bool IsDown(); // 是否縮小

bool IsBig(); // 是否是大隻的

bool IsMove(); // 是否在移動

void LoadBitmap(); // 載入圖形

void OnMove(Map \*m); // 移動

void OnShow(Map \*m); // 將圖形貼到畫面

void Hurted(Map \*m); // 受傷

void Die(Map \*m); // 死去

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 SetJump(bool flag); // 設定是否按下X鍵

void SetAttack(bool flag); // 設定是否按下Z鍵

void SetSuck(bool flag); // 設定是否吸怪

void SetBig(bool flag); // 設定是否含滷蛋

void SetRun(bool); // 設定是否按下C

void SetXY(int nx, int ny); // 設定左上角的座標

void SetEat(int t); // 設定吃了什麼怪物

void Restore(); // 密技:血回到滿

Gas\* GetGas();

Star\* GetStar();

protected:

CMovingBitmap blood6, blood5, blood4, blood3, blood2, blood1, blood0;

// 普通卡比

CMovingBitmap originR, originL, exhaleR, exhaleL, jumpR, jumpL, downR, downL, landingR, landingL, downAttackR, downAttackL, GG;

CAnimation goL, goR, flyR, prepareFlyR, flyL, prepareFlyL, hurtedL, hurtedR, runL, runR, suckR, suckL;

// 含東西卡比

CMovingBitmap bigOriginR, bigOriginL, bigJumpR, bigJumpL, bigLandingR, bigLandingL, threwR, threwL;

CAnimation bigGoL, bigGoR, swallowR, swallowL;

// 雷電卡比

CMovingBitmap Spark\_exhaleR, Spark\_exhaleL;

CAnimation Spark\_originR, Spark\_originL, Spark\_downR, Spark\_downL, Spark\_goR, Spark\_goL, Spark\_jumpR, Spark\_jumpL, Spark\_landingR, Spark\_landingL, Spark\_downAttackR, Spark\_downAttackL, Spark\_flyR, Spark\_prepareFlyR, Spark\_flyL, Spark\_prepareFlyL, Spark\_runR, Spark\_runL, Spark\_attackR, Spark\_attackL;

// 火焰卡比

CMovingBitmap fire\_exhaleR, fire\_exhaleL;

CAnimation fire\_originR, fire\_originL, fire\_downR, fire\_downL, fire\_goR, fire\_goL, fire\_jumpR, fire\_jumpL, fire\_landingR, fire\_landingL, fire\_downAttackR, fire\_downAttackL, fire\_flyR, fire\_prepareFlyR, fire\_flyL, fire\_prepareFlyL, fire\_runR, fire\_runL, fire\_attackR, fire\_attackL, attack1, attack2, attack3;

Gas gas;

Star star;

LostAbility lost;

int x, y; // 左上角座標

bool isMovingDown; // 是否正在往下移動

bool isMovingLeft; // 是否正在往左移動

bool isMovingRight; // 是否正在往右移動

bool isMovingUp; // 是否正在往上移動

bool isExhale; // 是否吐氣

bool isJump; // 是否按下X鍵

bool isAttack; // 是否按下Z鍵

bool isKick; // 是否使用踢擊

bool isSuck; // 是否使用吸怪

bool isHurted; // 是否被攻擊

bool isAlive; // 是否活著

bool isFly; // 是否在飛

bool isRunning; // 是否在跑

bool isBig; // 是否含滷蛋

bool isSwallow; // 是否在吞怪

bool isLanding; // 是否降落中

bool rightOrLeft; // 判斷左右

private:

void ShowKirby(Map \*m); // 顯示普通卡比

void ShowSparkKirby(Map \*m); // 顯示雷電卡比

void ShowFireKirby(Map \*m); // 顯示火焰卡比

int exhaleDelay; // 吐氣的時間

int jumpDistance; // 跳躍的距離

int kickDistance; // 踢擊的距離

int gasDistance; // 氣體飛行距離

int starDistance; // 星星飛行距離

bool bulletDirection; // 吐出物體方向

bool isInvincible; // 是否無敵

int InvincibleTime; // 無敵時間

void Attack(Map \*m); // 普通卡比攻擊

void Spark\_Attack(Map \*m); // 雷電卡比攻擊

void Fire\_Attack(Map \*m); // 火焰卡比攻擊

void Transform(); // 變身

int hp; // 血量

int type; // 型態

int eat; // 吃了什麼

int velocity; // 重力加速度

int count; // 計數是否一秒

int height; // 記錄當下的身高

int width; // 記錄當下的寬度

};

}

## Kirby.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Kirby.h"

namespace game\_framework {

Kirby::Kirby()

{

}

int Kirby::GetX1()

{

return x;

}

int Kirby::GetY1()

{

return y;

}

int Kirby::GetX2()

{

return x + width;

}

int Kirby::GetY2()

{

return y + height;

}

int Kirby::GetType()

{

return type;

}

int Kirby::GetHeight()

{

return height;

}

int Kirby::GetWidth()

{

return width;

}

void Kirby::Initialize(int px, int py)

{

const int X\_POS = px;

const int Y\_POS = py;

x = X\_POS;

y = Y\_POS;

hp = 6;

jumpDistance = 120;

kickDistance = 100;

exhaleDelay = 10;

gasDistance = starDistance = 0;

InvincibleTime = 0;

type = 0;

eat = -1;

velocity = 2;

count = 0;

isMovingLeft = isMovingRight = isMovingUp = isMovingDown = isExhale = isJump = isAttack = false;

isKick = isFly = isHurted = isSuck = isBig = isSwallow = isRunning = isInvincible = isLanding = false;

isAlive = rightOrLeft = true;

blood0.SetTopLeft(SIZE\_X / 2 - blood0.Width() / 2, SIZE\_Y - blood0.Height());

blood1.SetTopLeft(SIZE\_X / 2 - blood1.Width() / 2, SIZE\_Y - blood1.Height());

blood2.SetTopLeft(SIZE\_X / 2 - blood2.Width() / 2, SIZE\_Y - blood2.Height());

blood3.SetTopLeft(SIZE\_X / 2 - blood3.Width() / 2, SIZE\_Y - blood3.Height());

blood4.SetTopLeft(SIZE\_X / 2 - blood4.Width() / 2, SIZE\_Y - blood4.Height());

blood5.SetTopLeft(SIZE\_X / 2 - blood5.Width() / 2, SIZE\_Y - blood5.Height());

blood6.SetTopLeft(SIZE\_X / 2 - blood6.Width() / 2, SIZE\_Y - blood6.Height());

}

bool Kirby::IsAlive()

{

return isAlive;

}

bool Kirby::IsFly()

{

return isFly;

}

bool Kirby::IsKick()

{

return isKick;

}

bool Kirby::IsSuck()

{

return isSuck;

}

bool Kirby::IsAttack()

{

return isAttack;

}

bool Kirby::IsRight()

{

return rightOrLeft;

}

bool Kirby::IsDown()

{

return isMovingDown;

}

bool Kirby::IsBig()

{

return isBig;

}

bool Kirby::IsMove()

{

return isMovingLeft || isMovingRight;

}

void Kirby::LoadBitmap()　{　．．．．．　} //所有圖形的addbitmap

void Kirby::OnMove(Map \*m)

{

//被攻擊且不是在空中也沒超出邊界時

if (isHurted)

{

isFly = false;

if (m->isEmpty(GetX2() - width / 2, GetY2() + 1))

y++;

if (rightOrLeft && m->isEmpty(GetX1() - STEP\_SIZE, GetY1() + height / 2)

&& x - STEP\_SIZE >= 0)

x -= STEP\_SIZE;

else if (!rightOrLeft && m->isEmpty(GetX2() + STEP\_SIZE, GetY2() - height / 2) && x + STEP\_SIZE <= m->GetWidth() - width)

x += STEP\_SIZE;

}

else

{

if (isMovingLeft && !isSuck && !isSwallow && !isAttack)

{

rightOrLeft = false; //設定面向左邊

// 上坡

if (m->isSlope(GetX1() - 1, GetY2()) && !isMovingDown && (isFly || !isMovingUp)) {

if (isRunning && !isFly)

{

x -= STEP\_SIZE \* 2;

y -= STEP\_SIZE \* 2;

}

else

{

x -= STEP\_SIZE;

y -= STEP\_SIZE;

}

}

// 下坡，前腳(左腳)離開斜坡(下方不是空的)就不要在下移了，不然會衝進土裡

else if (m->isSlope(GetX2(), GetY2() + 1) && m->isEmpty(GetX1(), GetY2() + 1) && !isMovingDown && (isFly || !isMovingUp)) {

if (isRunning && !isFly)

{

x -= STEP\_SIZE \* 2;

y += STEP\_SIZE \* 2;

}

else

{

x -= STEP\_SIZE;

y += STEP\_SIZE;

}

}

//先判斷左邊是否可走且沒有按Down，狀態要是向左飛行中或正常向左走

//為了防止變身後卡比變高導致牆壁失效，將判斷撞牆的點設在Y2-10

else if (m->isEmpty(GetX1() - STEP\_SIZE, GetY2() - 10) && !isMovingDown && (isFly || !isMovingUp))

{

if (x <= 0) //邊界

x = 0;

else if (isRunning && !isFly && !m->isEmpty(GetX1() + width / 2, GetY2() + 1))

x -= STEP\_SIZE \* 2;

else

x -= STEP\_SIZE;

}

}

else if (isMovingRight && !isSuck && !isSwallow && !isAttack)

{

rightOrLeft = true; //設定面向右邊

// 上坡

if (m->isSlope(GetX2() + 1, GetY2()) && !isMovingDown && (isFly || !isMovingUp)) {

if (isRunning && !isFly)

{

x += STEP\_SIZE \* 2;

y -= STEP\_SIZE \* 2;

}

else

{

x += STEP\_SIZE;

y -= STEP\_SIZE;

}

}

// 下坡，前腳(右腳)離開斜坡(下方不是空的)就不要在下移了，不然會衝進土裡

else if (m->isSlope(GetX1(), GetY2() + 1) && m->isEmpty(GetX2(), GetY2() + 1) && !isMovingDown && (isFly || !isMovingUp)) {

if (isRunning && !isFly)

{

x += STEP\_SIZE \* 2;

y += STEP\_SIZE \* 2;

}

else

{

x += STEP\_SIZE;

y += STEP\_SIZE;

}

}

//先判斷右邊是否可走且沒有按Down，狀態要是向右飛行中或正常向右走

//為了防止變身後卡比變高導致牆壁失效，將判斷撞牆的點設在Y2-10

else if (m->isEmpty(GetX2() + STEP\_SIZE, GetY2() - 10) && !isMovingDown

&& (isFly || !isMovingUp))

{

if (x >= m->GetWidth() - width) //邊界

x = m->GetWidth() - width;

else if (isRunning && !isFly && !m->isEmpty(GetX2() - width / 2, GetY2() + 1))

x += STEP\_SIZE \* 2;

else

x += STEP\_SIZE;

}

}

if (isMovingUp && !isBig)

{

if (isFly && m->isEmpty\_2(GetX1() + flyR.Width() / 2, GetY1() - STEP\_SIZE))

{

if (y <= 0) //邊界

y = 0;

else

y -= STEP\_SIZE;

}

}

if (isJump && !isSuck)

{

if (jumpDistance == 120)

CAudio::Instance()->Play(jump);

jumpDistance -= 5;

if (m->isEmpty\_2(GetX1() + width / 2, GetY1() - 5) && y - 5 > 0) //會不會撞到頭

y -= 5;

if (jumpDistance == 0 || y - 5 <= 0)

{

jumpDistance = 120;

isJump = false;

}

}

}

//地吸引力

if (!(isMovingUp || isJump) && m->isEmpty(GetX2() , GetY2() + 1) && m->isEmpty(GetX1(), GetY2() + 1))

{

count++;

isLanding = true;

if (isFly)

y += 1;

else

{

y += velocity; // y軸下降(移動velocity個點，velocity的單位為 點/次)

if (velocity < 5 && count == 30)

{

velocity++; // 受重力影響，下次的下降速度增加

count = 0;

}

}

}

else if (!isFly && m->isEmpty(x, y + height + 1) && m->isEmpty(x + width, y + height + 1))

y += 1;

else

{

if (isLanding == true && !isFly)

{

CAudio::Instance()->Play(landing);

isLanding = false;

}

velocity = 2; // 重設重力加速度

count = 0;

}

}

void Kirby::OnShow(Map \*m)

{

if (type == 0)

{

if (isBig)

{

height = bigOriginR.Height();

width = bigOriginR.Width();

}

else

{

height = originR.Height();

width = originR.Width();

}

}

else if (type == 1)

{

height = Spark\_originR.Height();

width = Spark\_originR.Width();

}

else if (type == 2)

{

height = fire\_originR.Height();

width = fire\_originR.Width();

}

if (hp == 6) blood6.ShowBitmap();

else if (hp == 5) blood5.ShowBitmap();

else if (hp == 4) blood4.ShowBitmap();

else if (hp == 3) blood3.ShowBitmap();

else if (hp == 2) blood2.ShowBitmap();

else if (hp == 1) blood1.ShowBitmap();

else blood0.ShowBitmap();

if (InvincibleTime != 0) InvincibleTime--;

else isInvincible = false;

if (type == 0) ShowKirby(m);

else if (type == 1) ShowSparkKirby(m);

else if (type == 2) ShowFireKirby(m);

if (lost.IsAlive())

{

lost.OnMove(m, x, y, isSuck);

lost.OnShow(m);

}

}

void Kirby::Hurted(Map \*m)

{

if (type != 0)

{

lost.Initialize(type, x, y);

type = 0;

}

if (!isInvincible)

{

hp--;

isInvincible = true;

InvincibleTime = 60;

if(hp>0)

CAudio::Instance()->Play(hurted);

}

if (hp > 0)

isHurted = true;

else

{

CAudio::Instance()->Play(die);

CAudio::Instance()->Stop(AUDIO\_BACKGROUND);

isAlive = false;

}

}

void Kirby::Die(Map \*m)

{

GG.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

GG.ShowBitmap();

y -= 3;

}

void Kirby::SetMovingDown(bool flag)

{

isMovingDown = flag;

}

void Kirby::SetMovingLeft(bool flag)

{

isMovingLeft = flag;

}

void Kirby::SetMovingRight(bool flag)

{

isMovingRight = flag;

}

void Kirby::SetMovingUp(bool flag)

{

if (isBig)

isMovingUp = false;

else

isMovingUp = flag;

}

void Kirby::SetJump(bool flag)

{

isJump = flag;

}

void Kirby::SetAttack(bool flag)

{

isAttack = flag;

}

void Kirby::SetSuck(bool flag)

{

isSuck = flag;

}

void Kirby::SetBig(bool flag)

{

isBig = flag;

}

void Kirby::SetRun(bool flag) {

isRunning = flag;

}

void Kirby::SetXY(int nx, int ny)

{

x = nx; y = ny;

}

void Kirby::Restore()

{

hp = 6;

}

void Kirby::SetEat(int t)

{

eat = t;

}

Gas\* Kirby::GetGas()

{

return &gas;

}

Star\* Kirby::GetStar()

{

return &star;

}

void Kirby::Attack(Map \*m)

{

//------空氣彈------

if (isFly && isAttack)

{

gas.SetXY(x, y);

gas.SetAlive(true);

gasDistance = 176;

bulletDirection = rightOrLeft;

}

if (gasDistance != 0)

{

gasDistance -= STEP\_SIZE \* 2;

gas.OnMove(m, bulletDirection);

gas.OnShow(m);

}

else

gas.SetAlive(false);

//------星星------

if (isBig && isAttack && !isSuck && !star.IsAlive())

{

star.SetXY(x, y);

star.SetAlive(true);

starDistance = 376;

bulletDirection = rightOrLeft;

}

if (starDistance != 0)

{

starDistance -= STEP\_SIZE \* 2;

star.OnMove(m, bulletDirection);

star.OnShow(m);

}

else

star.SetAlive(false);

//------踢擊------

if (!isFly && !isBig && (isMovingDown || isKick) && (isAttack || isKick) && !m->isEmpty(GetX2() - width / 2, GetY2() + 1)) //在地面上蹲下按攻擊

{

isKick = true;

isAttack = false;

kickDistance -= 5;

if (rightOrLeft && m->isEmpty(GetX2() + 5, GetY2() - 5) && x + 5 <= m->GetWidth() - downAttackR.Width()) //右邊會不會踢牆(y-5是補償卡比大小)

x += 5;

else if (!rightOrLeft && m->isEmpty(GetX1() - 5, GetY2() - 5) && x - 5 >= 0) //左邊會不會踢牆(y-5是補償卡比大小)

x -= 5;

//踢完.踢到邊界.腳沒踏到地都不能使用踢擊

if (kickDistance == 0 || x <= 0 || x >= m->GetWidth() || m->isEmpty(GetX2() - width / 2, GetY2() + 1))

{

kickDistance = 100;

isKick = false;

}

}

}

void Kirby::Spark\_Attack(Map \*m)

{

//放電在怪物那邊偵測

//------空氣彈------

if (isFly && isAttack)

{

gas.SetXY(x, y + 40);

gas.SetAlive(true);

gasDistance = 176;

bulletDirection = rightOrLeft;

}

if (gasDistance != 0)

{

gasDistance -= STEP\_SIZE \* 2;

gas.OnMove(m, bulletDirection);

gas.OnShow(m);

}

else

gas.SetAlive(false);

//------踢擊------

if (!isFly && !isBig && (isMovingDown || isKick) && (isAttack || isKick) && !m->isEmpty(GetX2() - width / 2, GetY2() + 1)) //在地面上蹲下按攻擊

{

isKick = true;

isAttack = false;

kickDistance -= 5;

if (rightOrLeft && m->isEmpty(GetX2() + 5, GetY2() - 5) && x + 5 <= m->GetWidth() - Spark\_downAttackR.Width() + 40) //右邊會不會踢牆(y-5是補償卡比大小)(+40是消除動畫補償)

x += 5;

else if (!rightOrLeft && m->isEmpty(GetX1() - 5, GetY2() - 5) && x - 5 >= 0) //左邊會不會踢牆(y-5是補償卡比大小)

x -= 5;

//踢完.踢到邊界.腳沒踏到地都不能使用踢擊

if (kickDistance == 0 || x <= 0 || x >= m->GetWidth() || m->isEmpty(GetX2() - width / 2, GetY2() + 1))

{

kickDistance = 100;

isKick = false;

}

}

}

void Kirby::Fire\_Attack(Map \*m)

{

//放火在怪物那邊偵測

//------空氣彈------

if (isFly && isAttack)

{

gas.SetXY(x, y + 40);

gas.SetAlive(true);

gasDistance = 176;

bulletDirection = rightOrLeft;

}

if (gasDistance != 0)

{

gasDistance -= STEP\_SIZE \* 2;

gas.OnMove(m, bulletDirection);

gas.OnShow(m);

}

else

gas.SetAlive(false);

//------踢擊------

if (!isFly && !isBig && (isMovingDown || isKick) && (isAttack || isKick) && !m->isEmpty(GetX2() - width / 2, GetY2() + 1)) //在地面上蹲下按攻擊

{

isKick = true;

isAttack = false;

kickDistance -= 5;

if (rightOrLeft && m->isEmpty(GetX2() + 5, GetY2() - 5) && x + 5 <= m->GetWidth() - fire\_downAttackR.Width() + 30) //右邊會不會踢牆(y-5是補償卡比大小)(+40是消除動畫補償)

x += 5;

else if (!rightOrLeft && m->isEmpty(GetX1() - 5, GetY2() - 5) && x - 5 >= 0) //左邊會不會踢牆(y-5是補償卡比大小)

x -= 5;

//踢完.踢到邊界.腳沒踏到地都不能使用踢擊

if (kickDistance == 0 || x <= 0 || x >= m->GetWidth() || m->isEmpty(GetX2() - width / 2, GetY2() + 1))

{

kickDistance = 100;

isKick = false;

}

}

}

void Kirby::Transform()

{

if (eat == 1)

{

type = 1;

y -= 40;

}

else if (eat == 2)

{

type = 2;

y -= 40;

}

}

void Kirby::ShowKirby(Map \*m)

{

originR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

originL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

goL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

goR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

prepareFlyR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

prepareFlyL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

flyR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

flyL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

exhaleR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

exhaleL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

jumpR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

jumpL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

downR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

downL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

landingL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

landingR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

downAttackL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

downAttackR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

hurtedL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

hurtedR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

runR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

runL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

suckL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

suckR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigOriginR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigOriginL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigJumpR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigJumpL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigLandingR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigLandingL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

threwR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

threwL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigGoL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

bigGoR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

swallowR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

swallowL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Attack(m);

if (!isMovingUp && !isFly) //沒吸氣也沒飛行就要reset吸氣動畫

{

prepareFlyR.Reset();

prepareFlyL.Reset();

}

if (isHurted) // 被攻擊

{

isBig = false;

if (rightOrLeft)

{

hurtedR.OnMove();

hurtedR.OnShow();

}

else

{

hurtedL.OnMove();

hurtedL.OnShow();

}

if (hurtedL.IsFinalBitmap() || hurtedR.IsFinalBitmap())

{

isHurted = false;

hurtedL.Reset();

hurtedR.Reset();

}

}

else if (isBig)

{

if(isSuck)

isAttack = false;

if (isJump) //跳躍

{

if (rightOrLeft)

bigJumpR.ShowBitmap();

else

bigJumpL.ShowBitmap();

}

// || (starDistance < 376 && starDistance > 346)是為了讓吐怪的圖能顯示久一點到大概1秒(每秒跑30次OnShow迴圈)

else if (isAttack || (starDistance < 376 && starDistance > 346)) //吐怪

{

// isAttack要設為false，避免吐怪完馬上吸怪的bug

isAttack = false;

if (rightOrLeft)

threwR.ShowBitmap();

else

threwL.ShowBitmap();

// 在吐氣的圖顯示結束前將卡比變回原形

if (starDistance <= 350)

isBig = false;

}

else if (isSwallow || (isMovingDown && (!m->isEmpty(GetX1(), GetY2() + 1) || !m->isEmpty(GetX2(), GetY2() + 1)))) //吞食

{

isSwallow = true;

if (rightOrLeft)

{

swallowR.OnMove();

swallowR.OnShow();

}

else

{

swallowL.OnMove();

swallowL.OnShow();

}

if (swallowR.IsFinalBitmap() || swallowL.IsFinalBitmap())

{

isSwallow = false;

isBig = false;

Transform();

swallowR.Reset();

swallowL.Reset();

}

}

//自由落體

//兩腳地都必須空，且不是走在斜坡，才會顯示降落的圖

else if (!isFly && m->isEmpty(GetX1(), GetY2() + 1) && m->isEmpty(GetX2(), GetY2() + 1) && !m->isSlope(GetX1(), GetY2() + 11) && !m->isSlope(GetX2(), GetY2() + 11))

{

if (rightOrLeft)

bigLandingR.ShowBitmap();

else

bigLandingL.ShowBitmap();

}

else if (isMovingLeft) //向左走

{

if (isRunning)

bigGoL.SetDelayCount(2);

else

bigGoL.SetDelayCount(10);

bigGoL.OnShow();

bigGoL.OnMove();

}

else if (isMovingRight) //向右走

{

if (isRunning)

bigGoR.SetDelayCount(2);

else

bigGoR.SetDelayCount(10);

bigGoR.OnShow();

bigGoR.OnMove();

}

else if (rightOrLeft) //面相右

bigOriginR.ShowBitmap();

else if (!rightOrLeft) //面相左

bigOriginL.ShowBitmap();

}

else

{

if (isAttack && !isFly && !isMovingDown) //吸怪

{

isSuck = true;

if (rightOrLeft)

{

suckR.OnMove();

suckR.OnShow();

}

else

{

suckL.OnMove();

suckL.OnShow();

}

if (lost.IsAlive() && (lost.GetX() - x >= -2) && (lost.GetX() - x <= 2) && (lost.GetY() - y >= -2) && (lost.GetY() - y <= 2))

{

lost.SetAlive(false);

SetBig(true);

SetEat(lost.type);

}

}

else if (isJump) //跳躍

{

if (rightOrLeft)

jumpR.ShowBitmap();

else

jumpL.ShowBitmap();

}

else if ((isAttack || isExhale) && isFly) //吐氣

{

// isAttack要設為false，避免吐氣完馬上吸怪的bug

isAttack = false;

isExhale = true;

exhaleDelay--;

if (rightOrLeft)

exhaleR.ShowBitmap();

else

exhaleL.ShowBitmap();

if (exhaleDelay == 0)

{

exhaleDelay = 10;

prepareFlyR.Reset();

prepareFlyL.Reset();

isFly = false;

isExhale = false;

}

SetEat(-1);

}

else if (isMovingUp)

{

if (prepareFlyR.IsFinalBitmap() || prepareFlyL.IsFinalBitmap()) //飛行前吸氣

{

isFly = true;

if (rightOrLeft)

{

if (flyR.GetCurrentBitmapNumber() == 1)

CAudio::Instance()->Play(fly);

flyR.OnShow();

flyR.OnMove();

}

else

{

if (flyL.GetCurrentBitmapNumber() == 1)

CAudio::Instance()->Play(fly);

flyL.OnShow();

flyL.OnMove();

}

}

else

{

if (rightOrLeft)

{

prepareFlyR.OnShow();

prepareFlyR.OnMove();

}

else

{

prepareFlyL.OnShow();

prepareFlyL.OnMove();

}

}

}

else if (isFly) //飛行

{

if (rightOrLeft)

{

if (flyR.GetCurrentBitmapNumber() == 1)

CAudio::Instance()->Play(fly);

flyR.OnShow();

flyR.OnMove();

}

else

{

if (flyL.GetCurrentBitmapNumber() == 1)

CAudio::Instance()->Play(fly);

flyL.OnShow();

flyL.OnMove();

}

}

//自由落體

//兩腳地都必須空，且不是走在斜坡，才會顯示降落的圖

else if (!isFly && m->isEmpty(GetX1(), GetY2() + 1) && m->isEmpty(GetX2(), GetY2() + 1) && !m->isSlope(GetX1(), GetY2() + 11) && !m->isSlope(GetX2(), GetY2() + 11))

{

if (rightOrLeft)

landingR.ShowBitmap();

else

landingL.ShowBitmap();

}

else if (isKick && !isFly) //踢擊

{

if (rightOrLeft)

downAttackR.ShowBitmap();

else

downAttackL.ShowBitmap();

}

else if (isMovingDown && !m->isEmpty(GetX1() + width / 2, GetY2() + 1)) //縮小

{

if (rightOrLeft)

downR.ShowBitmap();

else

downL.ShowBitmap();

}

else if (!isFly && isRunning && (isMovingLeft || isMovingRight)) //跑

{

if (!rightOrLeft)

{

runL.OnShow();

runL.OnMove();

}

else

{

runR.OnShow();

runR.OnMove();

}

}

else if (!isFly && isMovingLeft) //一般向左走

{

goL.OnShow();

goL.OnMove();

}

else if (!isFly && isMovingRight) //一般向右走

{

goR.OnShow();

goR.OnMove();

}

else if (rightOrLeft) //面相右

originR.ShowBitmap();

else if (!rightOrLeft) //面相左

originL.ShowBitmap();

}

}

void Kirby::ShowSparkKirby(Map \*m)

{

// 雷電卡比圖片.動畫定位及大小補償

Spark\_exhaleR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_exhaleL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_downR.SetTopLeft(m->ScreenX(x), m->ScreenY(y+20));

Spark\_downL.SetTopLeft(m->ScreenX(x), m->ScreenY(y+20));

Spark\_landingR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_landingL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_originR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_originL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_goR.SetTopLeft(m->ScreenX(x), m->ScreenY(y-5));

Spark\_goL.SetTopLeft(m->ScreenX(x), m->ScreenY(y-5));

Spark\_jumpR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_jumpL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_downAttackR.SetTopLeft(m->ScreenX(x-40), m->ScreenY(y+30));

Spark\_downAttackL.SetTopLeft(m->ScreenX(x), m->ScreenY(y+30));

Spark\_flyR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_prepareFlyR.SetTopLeft(m->ScreenX(x), m->ScreenY(y+5));

Spark\_flyL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_prepareFlyL.SetTopLeft(m->ScreenX(x), m->ScreenY(y+5));

Spark\_runR.SetTopLeft(m->ScreenX(x-20), m->ScreenY(y+10));

Spark\_runL.SetTopLeft(m->ScreenX(x+10), m->ScreenY(y+10));

Spark\_attackR.SetTopLeft(m->ScreenX(x-40), m->ScreenY(y-20));

Spark\_attackL.SetTopLeft(m->ScreenX(x-40), m->ScreenY(y-20));

hurtedL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

hurtedR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Spark\_Attack(m);

if (!isMovingUp && !isFly) //沒吸氣也沒飛行就要reset吸氣動畫

{

Spark\_prepareFlyR.Reset();

Spark\_prepareFlyL.Reset();

}

if (isHurted) // 被攻擊

{

y += 40; //還原補償的高度

if (rightOrLeft)

{

hurtedR.OnMove();

hurtedR.OnShow();

}

else

{

hurtedL.OnMove();

hurtedL.OnShow();

}

if (hurtedL.IsFinalBitmap() || hurtedR.IsFinalBitmap())

{

isHurted = false;

hurtedL.Reset();

hurtedR.Reset();

}

}

else

{

if (isAttack && !isFly && !isMovingDown) //放電

{

if (rightOrLeft)

{

Spark\_attackR.SetDelayCount(5);

Spark\_attackR.OnMove();

Spark\_attackR.OnShow();

}

else

{

Spark\_attackL.SetDelayCount(5);

Spark\_attackL.OnMove();

Spark\_attackL.OnShow();

}

}

else if (isJump) //跳躍

{

if (rightOrLeft)

{

Spark\_jumpR.OnMove();

Spark\_jumpR.OnShow();

}

else

{

Spark\_jumpL.OnMove();

Spark\_jumpL.OnShow();

}

}

else if ((isAttack || isExhale) && isFly) //吐氣

{

// isAttack要設為false，避免吐氣完馬上吸怪的bug

isAttack = false;

isExhale = true;

exhaleDelay--;

if (rightOrLeft)

Spark\_exhaleR.ShowBitmap();

else

Spark\_exhaleL.ShowBitmap();

if (exhaleDelay == 0)

{

exhaleDelay = 10;

Spark\_prepareFlyR.Reset();

Spark\_prepareFlyL.Reset();

isFly = false;

isExhale = false;

}

}

else if (isMovingUp)

{

if (Spark\_prepareFlyR.IsFinalBitmap() || Spark\_prepareFlyL.IsFinalBitmap()) //飛行前吸氣

{

isFly = true;

if (rightOrLeft)

{

if (Spark\_flyR.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

Spark\_flyR.OnShow();

Spark\_flyR.OnMove();

}

else

{

if (Spark\_flyL.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

Spark\_flyL.OnShow();

Spark\_flyL.OnMove();

}

}

else

{

if (rightOrLeft)

{

Spark\_prepareFlyR.OnShow();

Spark\_prepareFlyR.OnMove();

}

else

{

Spark\_prepareFlyL.OnShow();

Spark\_prepareFlyL.OnMove();

}

}

}

else if (isFly) //飛行

{

if (rightOrLeft)

{

if (Spark\_flyR.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

Spark\_flyR.OnShow();

Spark\_flyR.OnMove();

}

else

{

if (Spark\_flyL.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

Spark\_flyL.OnShow();

Spark\_flyL.OnMove();

}

}

//自由落體

//兩腳地都必須空，且不是走在斜坡，才會顯示降落的圖

else if (!isFly && m->isEmpty(GetX1(), GetY2() + 1) && m->isEmpty(GetX2(), GetY2() + 1) && !m->isSlope(GetX1(), GetY2() + 11) && !m->isSlope(GetX2(), GetY2() + 11))

{

if (rightOrLeft)

{

Spark\_landingR.OnShow();

Spark\_landingR.OnMove();

}

else

{

Spark\_landingL.OnShow();

Spark\_landingL.OnMove();

}

}

else if (isKick && !isFly) //踢擊

{

if (rightOrLeft)

{

Spark\_downAttackR.OnMove();

Spark\_downAttackR.OnShow();

}

else

{

Spark\_downAttackL.OnMove();

Spark\_downAttackL.OnShow();

}

}

else if (isMovingDown && !m->isEmpty(GetX1() + width / 2, GetY1() + height + 1)) //縮小

{

if (rightOrLeft)

{

Spark\_downR.OnShow();

Spark\_downR.OnMove();

}

else

{

Spark\_downL.OnShow();

Spark\_downL.OnMove();

}

}

else if (!isFly && isRunning && (isMovingLeft || isMovingRight)) //跑

{

if (!rightOrLeft)

{

Spark\_runL.OnShow();

Spark\_runL.OnMove();

}

else

{

Spark\_runR.OnShow();

Spark\_runR.OnMove();

}

}

else if (!isFly && isMovingLeft) //一般向左走

{

Spark\_goL.OnShow();

Spark\_goL.OnMove();

}

else if (!isFly && isMovingRight) //一般向右走

{

Spark\_goR.OnShow();

Spark\_goR.OnMove();

}

else if (rightOrLeft) //面相右

{

Spark\_originR.OnMove();

Spark\_originR.OnShow();

}

else if (!rightOrLeft) //面相左

{

Spark\_originL.OnMove();

Spark\_originL.OnShow();

}

}

}

void Kirby::ShowFireKirby(Map \*m)

{

fire\_exhaleR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_exhaleL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_originR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_originL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_downR.SetTopLeft(m->ScreenX(x), m->ScreenY(y + 20));

fire\_downL.SetTopLeft(m->ScreenX(x), m->ScreenY(y + 20));

fire\_goR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_goL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_jumpR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_jumpL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_landingR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_landingL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_downAttackR.SetTopLeft(m->ScreenX(x - 30), m->ScreenY(y + 28));

fire\_downAttackL.SetTopLeft(m->ScreenX(x), m->ScreenY(y + 28));

fire\_flyR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_prepareFlyR.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_flyL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_prepareFlyL.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

fire\_runR.SetTopLeft(m->ScreenX(x - 35), m->ScreenY(y + 5));

fire\_runL.SetTopLeft(m->ScreenX(x), m->ScreenY(y + 5));

fire\_attackR.SetTopLeft(m->ScreenX(x - 15), m->ScreenY(y + 10));

fire\_attackL.SetTopLeft(m->ScreenX(x + 10), m->ScreenY(y + 10));

Fire\_Attack(m);

attack1.SetDelayCount(5);

attack2.SetDelayCount(5);

attack3.SetDelayCount(5);

if (!isMovingUp && !isFly) //沒吸氣也沒飛行就要reset吸氣動畫

{

fire\_prepareFlyR.Reset();

fire\_prepareFlyL.Reset();

}

if (isHurted) // 被攻擊

{

y += 40; //還原補償的高度

if (rightOrLeft)

{

hurtedR.OnMove();

hurtedR.OnShow();

}

else

{

hurtedL.OnMove();

hurtedL.OnShow();

}

if (hurtedL.IsFinalBitmap() || hurtedR.IsFinalBitmap())

{

isHurted = false;

hurtedL.Reset();

hurtedR.Reset();

}

}

else

{

if (isAttack && !isFly && !isMovingDown) //放火

{

if (rightOrLeft)

{

fire\_attackR.OnShow();

fire\_attackR.OnMove();

attack1.SetTopLeft(m->ScreenX(x + 63), m->ScreenY(y + 30));

attack2.SetTopLeft(m->ScreenX(x + 83), m->ScreenY(y + 10));

attack3.SetTopLeft(m->ScreenX(x + 133), m->ScreenY(y + 10));

}

else

{

fire\_attackL.OnShow();

fire\_attackL.OnMove();

attack1.SetTopLeft(m->ScreenX(x - 52), m->ScreenY(y + 30));

attack2.SetTopLeft(m->ScreenX(x - 72), m->ScreenY(y + 10));

attack3.SetTopLeft(m->ScreenX(x - 122), m->ScreenY(y + 10));

}

attack1.OnMove();

attack1.OnShow();

attack2.OnMove();

attack2.OnShow();

attack3.OnMove();

attack3.OnShow();

}

else if (isJump) //跳躍

{

if (rightOrLeft)

{

fire\_jumpR.OnMove();

fire\_jumpR.OnShow();

}

else

{

fire\_jumpL.OnMove();

fire\_jumpL.OnShow();

}

}

else if ((isAttack || isExhale) && isFly) //吐氣

{

// isAttack要設為false，避免吐氣完馬上吸怪的bug

isAttack = false;

isExhale = true;

exhaleDelay--;

if (rightOrLeft)

fire\_exhaleR.ShowBitmap();

else

fire\_exhaleL.ShowBitmap();

if (exhaleDelay == 0)

{

exhaleDelay = 10;

fire\_prepareFlyR.Reset();

fire\_prepareFlyL.Reset();

isFly = false;

isExhale = false;

}

}

else if (isMovingUp)

{

if (fire\_prepareFlyR.IsFinalBitmap() || fire\_prepareFlyL.IsFinalBitmap()) //飛行前吸氣

{

isFly = true;

if (rightOrLeft)

{

if (fire\_flyR.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

fire\_flyR.OnShow();

fire\_flyR.OnMove();

}

else

{

if (fire\_flyL.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

fire\_flyL.OnShow();

fire\_flyL.OnMove();

}

}

else

{

if (rightOrLeft)

{

fire\_prepareFlyR.OnShow();

fire\_prepareFlyR.OnMove();

}

else

{

fire\_prepareFlyL.OnShow();

fire\_prepareFlyL.OnMove();

}

}

}

else if (isFly) //飛行

{

if (rightOrLeft)

{

if (fire\_flyR.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

fire\_flyR.OnShow();

fire\_flyR.OnMove();

}

else

{

if (fire\_flyL.GetCurrentBitmapNumber() == 3)

CAudio::Instance()->Play(fly);

fire\_flyL.OnShow();

fire\_flyL.OnMove();

}

}

//自由落體

//兩腳地都必須空，且不是走在斜坡，才會顯示降落的圖

else if (!isFly && m->isEmpty(GetX1(), GetY2() + 1) && m->isEmpty(GetX2(), GetY2() + 1) && !m->isSlope(GetX1(), GetY2() + 11) && !m->isSlope(GetX2(), GetY2() + 11))

{

if (rightOrLeft)

{

fire\_landingR.OnShow();

fire\_landingR.OnMove();

}

else

{

fire\_landingL.OnShow();

fire\_landingL.OnMove();

}

}

else if (isKick && !isFly) //踢擊

{

if (rightOrLeft)

{

fire\_downAttackR.OnMove();

fire\_downAttackR.OnShow();

}

else

{

fire\_downAttackL.OnMove();

fire\_downAttackL.OnShow();

}

}

else if (isMovingDown && !m->isEmpty(GetX1() + width / 2, GetY1() + height + 1)) //縮小

{

if (rightOrLeft)

{

fire\_downR.OnShow();

fire\_downR.OnMove();

}

else

{

fire\_downL.OnShow();

fire\_downL.OnMove();

}

}

else if (!isFly && isRunning && (isMovingLeft || isMovingRight)) //跑

{

if (!rightOrLeft)

{

fire\_runL.OnShow();

fire\_runL.OnMove();

}

else

{

fire\_runR.OnShow();

fire\_runR.OnMove();

}

}

else if (!isFly && isMovingLeft) //一般向左走

{

fire\_goL.OnShow();

fire\_goL.OnMove();

}

else if (!isFly && isMovingRight) //一般向右走

{

fire\_goR.OnShow();

fire\_goR.OnMove();

}

else if (rightOrLeft) //面相右

{

fire\_originR.OnMove();

fire\_originR.OnShow();

}

else if (!rightOrLeft) //面相左

{

fire\_originL.OnMove();

fire\_originL.OnShow();

}

}

}

}

## Enemy.h

#pragma once

#include "Map.h"

#include "Kirby.h"

namespace game\_framework {

class Enemy

{

public:

Enemy();

int GetX1(); // 左上角 x 座標

int GetY1(); // 左上角 y 座標

void Hurted(Kirby\* kirby); // 被攻擊

void Sucked(Kirby\* kirby); // 被吸

void SetAlive(bool flag); // 設定死活

bool IsAlive(); // 是否活著

virtual int GetX2(); // 右下角 x 座標

virtual int GetY2(); // 右下角 y 座標

virtual void OnShow(Map \*m, Kirby \*kirby); // 顯示

virtual void OnMove(Map \*m, Kirby\* kirby); // 移動

virtual void LoadBitmap(); // 讀圖

virtual void Attack(Kirby\* kirby); // 攻擊

protected:

int x, y, hp; // 左上角座標.血量

bool is\_alive; // 是否活著

bool is\_sucked; // 是否被吸

bool RightOrLeft; // 判斷左右

bool Ishurted; // 受傷與否

bool IsKing; // 是不是王

bool HitRectangle(int tx1, int ty1, int tx2, int ty2); // 是否碰到參數範圍的矩形

double ComputeDistance(int tx1, int ty1); // 計算距離

};

}

## Enemy.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Enemy.h"

namespace game\_framework {

Enemy::Enemy()

{

}

int Enemy::GetX1()

{

return x;

}

int Enemy::GetY1()

{

return y;

}

int Enemy::GetX2()

{

return 0;

}

int Enemy::GetY2()

{

return 0;

}

void Enemy::Hurted(Kirby\* kirby)

{

// 被踢

if ((kirby->GetType() == 0 && HitRectangle(kirby->GetX1(), kirby->GetY1(), kirby->GetX2(), kirby->GetY2()) && kirby->IsKick()) ||((kirby->GetType() == 1 || kirby->GetType() == 2) && HitRectangle(kirby->GetX1(), kirby->GetY1() + 40, kirby->GetX2(), kirby->GetY1() + 60) && kirby->IsKick())) {

hp -= 10;

Ishurted = true;

}

// 被空氣砲擊中

else if (kirby->GetGas()->IsAlive()&&HitRectangle(kirby->GetGas()->GetX1()+10, kirby->GetGas()->GetY1() + 10, kirby->GetGas()->GetX2() - 10, kirby->GetGas()->GetY2() - 10)) {

hp -= 10;

Ishurted = true;

kirby->GetGas()->SetAlive(false);

}

// 被星星擊中

else if (kirby->GetStar()->IsAlive() && HitRectangle(kirby->GetStar()->GetX1() + 10, kirby->GetStar()->GetY1() + 10, kirby->GetStar()->GetX2() - 10, kirby->GetStar()->GetY2() - 10)) {

hp -= 20;

Ishurted = true;

kirby->GetStar()->SetAlive(false);

}

// 被電

else if (kirby->GetType() == 1 && kirby->IsAttack() && !kirby->IsDown() && HitRectangle(kirby->GetX1() - 40, kirby->GetY1() - 10, kirby->GetX1() + 90, kirby->GetY1() + 110)){ // -10補償圖片 +145.+158是雷電圖檔的大小

hp -= 1;

Ishurted = true;

}

// 被燒

else if (kirby->GetType() == 2 && kirby->IsAttack() && !kirby->IsDown() && ((kirby->IsRight() && HitRectangle(kirby->GetX1() + 63, kirby->GetY1() + 12, kirby->GetX1() + 203, kirby->GetY1() + 90)) || (!kirby->IsRight() && HitRectangle(kirby->GetX1() - 112, kirby->GetY1() + 12, kirby->GetX1() - 12, kirby->GetY1() + 90)))) {

hp -= 1;

Ishurted = true;

}

else {

Ishurted = false;

}

if (kirby->IsSuck() && (x - kirby->GetX1() > 2 || x - kirby->GetX1() < -2 || y - kirby->GetY1() > 2 || y - kirby->GetY1() < -2))

is\_alive = true;

else if (hp <= 0)

{

CAudio::Instance()->Play(enemyDie);

is\_alive = false;

}

}

void Enemy::Sucked(Kirby\* kirby)

{

if (IsKing)

return;

if (ComputeDistance(kirby->GetX1(), kirby->GetY1()) < 150.0 && kirby->IsSuck() && ((kirby->IsRight() && x - kirby->GetX1() >= 0) || (!kirby->IsRight() && x - kirby->GetX2() <= 0)))

{

hp -= 10;

if (hp <= 0)

is\_sucked = true;

}

else

is\_sucked = false;

}

void Enemy::SetAlive(bool flag)

{

is\_alive = flag;

}

bool Enemy::IsAlive() {

return is\_alive;

}

void Enemy::OnShow(Map \*m, Kirby \*kirby)

{

}

void Enemy::OnMove(Map \*m, Kirby\* kirby)

{

}

void Enemy::LoadBitmap()

{

}

void Enemy::Attack(Kirby\* kirby)

{

}

bool Enemy::HitRectangle(int tx1, int ty1, int tx2, int ty2)

{

int x1 = GetX1(); // 左上角x座標

int y1 = GetY1(); // 左上角y座標

int x2 = GetX2(); // 右下角x座標

int y2 = GetY2(); // 右下角y座標

// 檢測矩形與參數矩形是否有交集

return (tx2 >= x1 && tx1 <= x2 && ty2 >= y1 && ty1 <= y2);

}

double Enemy::ComputeDistance(int tx1, int ty1)

{

int temp = (x - tx1)\*(x - tx1) + (y - ty1)\*(y - ty1);

return sqrt(temp);

}

}

## Map.h

#pragma once

namespace game\_framework {

class Map

{

public:

Map();

int GetWidth(); //取得背景寬度

int GetHeight(); //取得背景高度

int ScreenX(int x); // 螢幕 x 座標

int ScreenY(int y); // 螢幕 y 座標

void Initialize(); // 設定為初始值

void LoadBitmap(char\*,COLORREF,char\*,string); // 載入圖形

void OnMove(int,int); // 地圖移動

void OnShow(); // 將圖形貼到畫面

void SetXY(int nx, int ny); // 設定螢幕畫面左上角的座標

bool isEmpty(int x, int y); // 判斷碰壁

bool isEmpty\_2(int x, int y); // 判斷碰壁

bool isSlope(int, int);

void SetMouse(int ,int); //設定滑鼠座標

void IsLclick(bool); //設定左鍵按下

void IsRclick(bool); //設定右鍵按下

protected:

CMovingBitmap background; // 地圖

CMovingBitmap foreground;

CMovingBitmap ball\_1,ball\_2,ball\_3; //顯示地形圖

private:

bool Lclick=false, Rclick=false; //滑鼠左鍵與右鍵是否按下 預設為否

int mx, my; //滑鼠座標

int sx, sy; //左上角座標

int mapSize\_X, mapSize\_Y; //地圖所切的格數

int map[1000][1000];

string Mapfile;

};

}

## Map.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Map.h"

#include <fstream>

namespace game\_framework {

Map::Map()

{

}

void Map::SetMouse(int x, int y) { //設定滑鼠座標

mx = x;

my = y;

}

void Map::IsLclick(bool flag) { //設定左鍵是否按下

Lclick= flag;

ofstream output(Mapfile);

for (int i = 0; i < mapSize\_Y; i++) {

for (int j = 0; j < mapSize\_X; j++) {

output << map[i][j] << " ";

}

output << endl;

}

output.close();

}

void Map::IsRclick(bool flag) { //設定右鍵是否按下

Rclick= flag;

ofstream output(Mapfile);

for (int i = 0; i < mapSize\_Y; i++) {

for (int j = 0; j < mapSize\_X; j++) {

output << map[i][j] << " ";

}

output << endl;

}

output.close();

}

int Map::GetWidth() { //取得背景寬度

return foreground.Width();

}

int Map::GetHeight() { //取得背景長度

return foreground.Height();

}

int Map::ScreenX(int x) //回傳人物在螢幕上座標

{

return x-sx;

}

int Map::ScreenY(int y) //同上

{

return y-sy;

}

void Map::Initialize()

{

//const int X\_POS = 320;

//const int Y\_POS = 240;

//sx = X\_POS;

//sy = Y\_POS;

ifstream file(Mapfile); //將資料存入陣列

for (int i = 0; i<mapSize\_Y; i++)

{

for (int j = 0; j < mapSize\_X; j++) {

file >> map[i][j];

}

}

file.close();

}

void Map::LoadBitmap(char\* fore,COLORREF color,char\* back,string file)

{

foreground.LoadBitmap(fore,color);

background.LoadBitmap(back);

Mapfile = file;

mapSize\_X = (foreground.Width() \* 10 + 5) / 100;

mapSize\_Y = (foreground.Height() \* 10 + 5) / 100;

ball\_1.LoadBitmap(".//RES//ball\_1.bmp", RGB(0, 0, 0));

ball\_2.LoadBitmap(".//RES//ball\_2.bmp", RGB(0, 0, 0));

ball\_3.LoadBitmap(".//RES//ball\_3.bmp", RGB(0, 0, 0));

}

void Map::OnMove(int x, int y) //地圖隨人物移動

{

sx = x - SIZE\_X / 2; //讓人物保持在地圖中間

sy = y - SIZE\_Y / 2;

if (x <= SIZE\_X / 2) { //設定當人物接近邊界時地圖移動停止

sx = 0;

}

if (x >= foreground.Width() - SIZE\_X / 2) {

sx = foreground.Width() - SIZE\_X;

}

if (y <= SIZE\_Y / 2) {

sy = 0;

}

if (y >= foreground.Height() - SIZE\_Y / 2) {

sy = foreground.Height() - SIZE\_Y;

}

}

void Map::OnShow()

{

background.SetTopLeft(0, 0);

background.ShowBitmap();

foreground.SetTopLeft(-sx, -sy); // 指定第(i, j)這一格的座標

foreground.ShowBitmap();

//ofstream set(Mapfile); //第一次建立地圖設立初始值

//for (int i = 0; i < mapSize\_Y; i++) {

// for (int j = 0; j < mapSize\_X; j++) {

// set << 0 << " ";

// }

// set << endl;

//}

//set.close();

//for (int i = 0; i < mapSize\_Y; i++) {

// for (int j = 0; j < mapSize\_X; j++) {

// int x = j \* 10 - sx; // 算出第(i, j)這一格的 x 螢幕座標

// int y = i \* 10 - sy; // 算出第(i, j)這一格的 y 螢幕座標

// if (mx >= x && mx <= x + 10 && my >= y && my <= y + 10) { //判斷滑鼠位置

// if (Lclick) { //左鍵為設立障礙物

// map[i][j] = 1;

// }

// if (Lclick) { //左鍵為設立可穿透的障礙物

// map[i][j] = 2;

// }

// if (Lclick) { //左鍵為設立斜坡

// map[i][j] = 3;

// }

// if (Rclick) { //右鍵為取消障礙物

// map[i][j] = 0;

// }

// }

// switch (map[i][j]) { //顯示格子

// case 1:

// ball\_1.SetTopLeft(x, y); // 指定第(i, j)這一格的座標

// ball\_1.ShowBitmap();

// break;

// case 2:

// ball\_2.SetTopLeft(x, y);

// ball\_2.ShowBitmap();

// break;

// case 3:

// ball\_3.SetTopLeft(x, y);

// ball\_3.ShowBitmap();

// break;

// }

// }

//}

}

void Map::SetXY(int nx, int ny)

{

sx = nx;

sy = ny;

}

bool Map::isEmpty(int x, int y) // (x, y) 為地圖的點座標

{

int gx = x / 10; // 轉換為X軸格座標(整數除法)

int gy = y / 10; // 轉換為Y軸格座標(整數除法)

//map[x][y]中的y表示X軸的格數，x表示Y軸

return map[gy][gx] == 0; // 假設 0 代表空的

}

bool Map::isEmpty\_2(int x, int y) // (x, y) 為地圖的點座標

{

int gx = x / 10; // 轉換為X軸格座標(整數除法)

int gy = y / 10; // 轉換為Y軸格座標(整數除法)

//map[x][y]中的y表示X軸的格數，x表示Y軸

return map[gy][gx] == 0 || map[gy][gx]==2; // 假設 0 代表空的2代表可穿越

}

bool Map::isSlope(int x, int y) {

int gx = x / 10; // 轉換為X軸格座標(整數除法)

int gy = y / 10; // 轉換為Y軸格座標(整數除法)

//map[x][y]中的y表示X軸的格數，x表示Y軸

return map[gy][gx] == 3 ;// 假設 3 代表斜坡

}

}

## Door.h

#pragma once

#include"Map.h"

#include"Kirby.h"

namespace game\_framework {

class Door

{

public:

Door();

int GetMapNum(); //取得地圖號碼

int GetWidth(); //取得背景寬度

int GetHeight(); //取得背景高度

int GetX();

int GetY();

int GetNextX();

int GetNextY();

int GetX2();

int GetY2();

void Initialize(int x,int y,int Num,int,Door\* ); // 設定為初始值

void LoadBitmap(); // 載入圖形

void OnMove(); // 地圖移動

void OnShow(Map \*); // 將圖形貼到畫面

void SetEnter(bool flag); // 設定是否正進入傳送門

bool IsEnter(Kirby\*); // 是否進入

Door GetNextDoor();

protected:

CAnimation door; // 地圖

CAnimation Gate;

private:

bool Enter=false; //是否在門按上 預設為否

int x, y; //左上角座標

int mapNum; //記錄此門傳送到幾號地圖

int map; //現在所在地圖

Door\* NextDoor;

};

}

## Door.cpp

#include "stdafx.h"

#include "Resource.h"

#include <mmsystem.h>

#include <ddraw.h>

#include "audio.h"

#include "gamelib.h"

#include "Door.h"

#include <fstream>

namespace game\_framework {

Door::Door(){

}

int Door::GetMapNum() {

return mapNum;

}

int Door::GetHeight() {

return door.Height();

}

int Door::GetWidth() {

return door.Width();

}

int Door::GetX() {

return x;

}

int Door::GetY() {

return y;

}

int Door::GetX2() {

return x + door.Width();

}

int Door::GetY2() {

return y + door.Height();

}

int Door::GetNextX() { //回傳下一個門的X座標

return NextDoor->GetX();

}

int Door::GetNextY() { //回傳下一個門的Y座標

return NextDoor->GetY();

}

Door Door::GetNextDoor() { //回傳指標

return \*NextDoor;

}

void Door::Initialize(int px,int py,int num,int m ,Door\* h) {//設定門的位置所在地圖下一個門的地圖

x = px;

y = py;

map = m;

mapNum = num;

NextDoor = h;

}

void Door::LoadBitmap() {　．．．．．　} //所有圖形的addbitmap

void Door::OnMove() {

door.OnMove();

Gate.OnMove();

Gate.SetDelayCount(7);

}

void Door::OnShow(Map \*m) {

if (map == 0) { //當為地圖0時為一般的門

door.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

door.OnShow();

}

else { //其餘為有星星的門

Gate.SetTopLeft(m->ScreenX(x), m->ScreenY(y));

Gate.OnShow();

}

}

void Door::SetEnter(bool flag) { //設定是否按上

Enter = flag;

}

bool Door::IsEnter(Kirby \*k) {

if (map == 0) { //當為地圖0時為一般的門

if ((k->GetX1() + k->GetX2()) / 2 >= x && (k->GetX1() + k->GetX2()) / 2 <= x + door.Width() && (k->GetY1() + k->GetY2()) / 2 >= y && (k->GetY1() + k->GetY2()) / 2 <= y + door.Height() && Enter)

return true;

else

return false;

}

else { //其餘為有星星的門

if ((k->GetX1() + k->GetX2()) / 2 >= x && (k->GetX1() + k->GetX2()) / 2 <= x + Gate.Width() && (k->GetY1() + k->GetY2()) / 2 >= y && (k->GetY1() + k->GetY2()) / 2 <= y + Gate.Height() && Enter)

return true;

else

return false;

}

}

}