**目錄**

**一、簡介**

**1.動機 1**

**2.分工 1**

**二、遊戲介紹**

**1.遊戲說明 2**

**2.遊戲圖形 3**

**3.遊戲音效 6**

**三、程式設計**

**1.程式架構 7**

**2.程式類別 9**

**3.程式技術 11**

**四、結語**

**1.問題及解決方法 16**

**2.時間表 17**

**3.貢獻比例 18**

**4.檢核表 18**

**5.收穫 18**

**6.心得、感想 21**

**7.課程建議 24**

**五、附錄 25**

**一、簡介**

**1.動機**

我們從寒假開始構思OOP實習題目，考慮範圍涵蓋網頁FLASH遊戲及任天堂經典遊戲。我們希望選定的題目能夠支援2P模式以增加趣味性並且能夠充分發揮OOP物件導向的精神以便讓我們在修習完這門OOP物件導向實習課程能夠獲得良好的學習成果。

Chip 'n Dale Rescue Rangers這款遊戲能夠滿足我們的需求，並且具有鮮明的遊戲特色。不僅是單純俱有玩家與怪物間的互動，更有玩家跟玩家之間的互動關係。因此，我們決定選擇Chip 'n Dale Rescue Rangers作為此次課程的主題。

**2.分工**

學習前期主要以共同開發程式碼為主，後期分工越趨專業。

遊戲分工共分為三項，以下分別詳述：

* 遊戲圖檔
  + 謝宗廷：遊戲截圖、圖片去背
  + 陳科銘：圖片調整大小、背景組合與調整、背景裁切
* 遊戲音效
  + 謝宗廷：遊戲音樂擷取、音樂剪接
  + 陳科銘：音效尋找
* 程式設計
  + 謝宗廷 ：
    - 音效控制
    - 人物控制、動畫處理
    - 分數顯示
    - STATE架構
  + 陳科銘：
    - 怪物AI設計
    - 地圖管理
    - 轉場特效
    - 遊戲物件設計
    - 地圖編輯器模式

**二、遊戲介紹**

**1.遊戲說明**

Chip 'n Dale Rescue Rangers是一款經典超任遊戲。遊戲是以雙向2D捲軸方式呈現。玩家可以扮演奇奇與蒂蒂開始救援隊的冒險之旅。

* 遊戲開始畫面可以選擇玩家人數及扮演角色

以左右方向鍵選擇、Enter鍵為確認，按下空白鍵可以查看遊戲說明

* 1P模式

1P：上下左右：方向鍵 A鍵：Z B鍵：X 查看分數：TAB

* 2P模式

1P：上下左右：方向鍵 A鍵：N B鍵：M

2P：上下左右：WASD A鍵：Z B鍵：X 查看分數：TAB

* 地圖編輯器模式

上下左右：方向鍵 切換模式：TAB

切換物件：Z , X 放置：ENTER

儲存：ESC (若不儲存而離開，所做的更動只有一次性的效果)

* 密技

密技只支援1P模式，且無分大小寫。

* + GOD：切換到GOD模式，取消物件和怪物對人物的互動關係，人物可以騰空飛翔並且為無敵狀態。
  + EDIT ：切換地圖編輯器模式 / 一般模式
  + NTUT：遵循傳統將BOSS頭像更換為老師的頭像
  + 9999：人物生命值上限由3改為9999
* 快捷鍵
  + Ctrl + Q：暫停
  + Ctrl + F：切換全螢幕 / 視窗模式

**2.遊戲圖形**

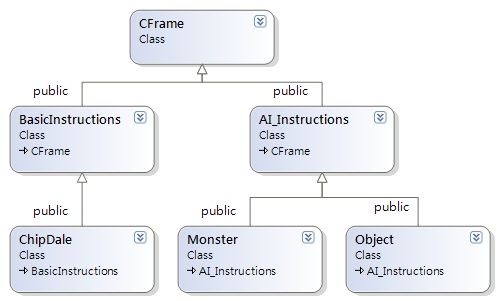
|  |  |  |  |
| --- | --- | --- | --- |
| **Chip** | | | |
| Faint_1R.bmpHurt_R.bmpjump_1R.bmpResurrect_R.bmprun_1R.bmprun_2R.bmprun_3R.bmprun_4R.bmpjump_2R.bmpstand_R.bmpTake_Jump_R.bmpTake_R.bmpTake_Run_1R.bmpTake_Run_2R.bmp | | | |
| **Dale** | | | |
| Faint_1R.bmpHurt_R.bmpjump_1R.bmpResurrect_R.bmprun_1R.bmprun_2R.bmprun_3R.bmprun_4R.bmpSquat_1R.bmpstand_R.bmpTake_Jump_R.bmpTake_R.bmpTake_Run_1R.bmpTake_Run_2R.bmp | | | |
| **MachineDog** | | | |
| **mechine_dog_run_1R.bmpmechine_dog_run_2R.bmpmechine_dog_stand_1R.bmpmechine_dog_stand_2R.bmp** | | | |
| **Mouse** | | | |
| **mouse_Jump_R.bmpmouse_Run_1R.bmpmouse_Run_2R.bmpmouse_Run_3R.bmp** | | | |
| **Greedy** | **Wasp** | | **Explosion** |
| **Greedy_1R.bmpGreedy_2R.bmp** | **Wasp_1R.bmpWasp_2R.bmp** | | **Explosion_1.bmpExplosion_2.bmpExplosion_3.bmpExplosion_4.bmp** |
| **Centipede** | | | |
| **Centipede_body_normal.bmpCentipede_body_die.bmphead.bmphead_die.bmpL_Fist.bmpL_Cloth.bmpR_Cloth.bmpR_Fist.bmpLightning.bmp** | | | |
| **Angel** | | **Electric** | |
| **bee_1L.bmpbee_1R.bmp** | | **Electric1.bmpElectric2.bmp** | |
| **一般物件** | | | |
| **6.bmp0.bmp1.bmp2.bmp13.bmp17.bmp3.bmp14.bmp15.bmp16.bmp12.bmp7.bmp8.bmp10.bmp11.bmp9.bmp18.bmp**  **5.bmpStone_R.bmp4.bmp19.bmpCheese.bmp** | | | |
| **單人遊戲截圖** | | | |
| **1_cr.png2_cr.png3_cr.png4_cr.png5_cr.png6_cr.png7_cr.png8_cr.png9_cr.png10_cr.png11_cr.png12_cr.png13_cr.png14_cr.png15_cr.png16_cr.png17_cr.png18_cr.png** | | | |
| **雙人遊戲截圖** | | | |
| **19_cr.png20_cr.png21_cr.png** | | | |

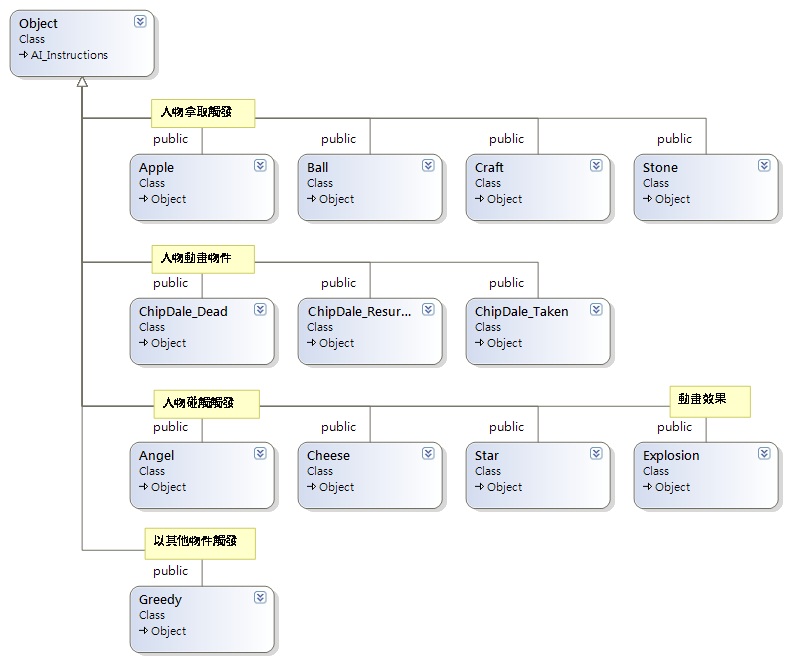
**3.遊戲音效**

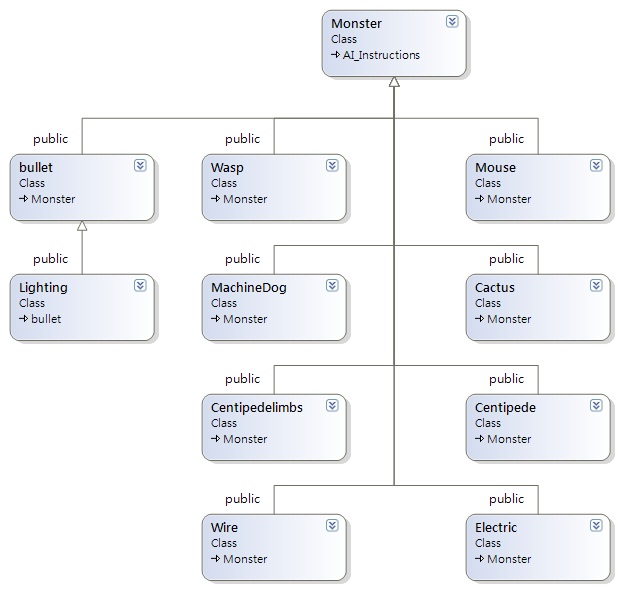
|  |  |
| --- | --- |
| 音效 | 來源 |
| 遊戲開場音樂 | 原作截取 |
| 關卡A背景音樂(A段) | 原作截取 |
| 關卡A背景音樂(B段) | 原作截取 |
| 關卡A BOSS音樂 | 原作截取 |
| 地圖編輯器模式音樂 | Rag Time On the Rag (黃金傳說料理背景音樂) |
| 遊戲結束音樂 | 原作截取 |
| 按鈕音效 | 網路搜尋 |
| 死亡音效 | 原作截取 |
| 過關音效 | 原作截取 |

**三、程式設計**

**1.程式架構**



****

****

****

**2.程式類別**

|  |  |  |  |
| --- | --- | --- | --- |
| 類別名稱 | .h檔行數 | .cpp檔行數 | 說明 |
| CGameStateInit | 12 | 70 | 遊戲的開頭畫面 |
| CGameStateRun | 29 | 432 | 遊戲的執行主要物件 |
| CGameStateBonus | 10 | 19 | 遊戲的獎勵狀態 |
| CGameStateOver | 11 | 43 | 遊戲的結束畫面 |
| CAudio | 15 | 86 | 音效介面及掌管遊戲音效控制 |
| ToolCDC | 24 | 96 | 提供介面處理畫面淡出淡入 |
| GameMapEdit | 23 | 148 | 地圖編輯器 |
| GameScore | 16 | 75 | 遊戲計分器 |
| MapManage | 38 | 442 | 地圖移動及碰撞系統 |
| CFrame | 24 | 99 | 基礎框架物件(抽象類別) |
| AI\_Instructions | 16 | 88 | AI指令集(設計怪物AI) |
| BasicInstructions | 20 | 245 | 基本指令集(按鍵觸發) |
| ChipDale | 69 | 609 | 遊戲人物 |
| Object | 20 | 77 | 遊戲物件(抽象類別) |
| ChipDale\_Dead | 14 | 81 | 人物死亡動畫物件 |
| ChipDale\_Resurrect | 11 | 59 | 人物復活狀態物件 |
| ChipDale\_Taken | 11 | 69 | 人物被抓狀態物件 |
| Angel | 13 | 94 | 小天使(人物觸發後無敵) |
| Apple | 10 | 49 | 蘋果物件(可拿起) |
| Ball | 11 | 111 | 球物件(可拿起) |
| Cheese | 10 | 53 | 起司物件(可吃，觸發Greedy) |
| Craft | 12 | 65 | 小箱子(可拿起) |
| Explosion | 11 | 82 | 爆炸物件(觸發後new新物件) |
| Greedy | 13 | 79 | 貪婪老鼠(觸發後在地圖鑽洞) |
| Star | 10 | 40 | 星星(分數物件，可吃) |
| Stone | 17 | 126 | 石頭(可拿起) |
| Monster | 20 | 75 | 遊戲怪物(抽象類別) |
| Bullet | 9 | 32 | 子彈(抽現類別) |
| Cactus | 8 | 28 | 仙人掌 |
| Centipede | 16 | 93 | 蜈蚣本體(第一關BOSS) |
| Centipedelimbs | 14 | 125 | 蜈蚣四肢 |
| Electric | 13 | 44 | 電(本體為Wasp) |
| Lighting | 8 | 22 | 閃電(繼承子彈) |
| MachineDog | 10 | 102 | 機器狗 |
| Mouse | 11 | 144 | 老鼠 |
| Wire | 13 | 52 | 線圈(界定Electric移動範圍) |
| Wasp | 11 | 86 | 虎頭蜂 |
| 總計 | 603 | 4240 |  |

**3.程式技術**

* 移動運算帶動二維螢幕
  + 建立10x10的二維陣列( MapXY ) ，其Index以地圖格座標運算。此陣列記錄地圖移動路線，預設值為0。螢幕需要帶動的地方則須先轉換成地圖格座標並在MapXY對應地方的填入介於1~15之數字。
  + MapXY陣列每格所代表的意義是一張螢幕，大小為640x480。換言之，每個物件的絕對座標介於(0,0)~(6400x4800,6400x480)。
  + 絕對座標(x,y) 轉地圖格座標(Mx,My)的算法為

Mx = x / 640 , My = y / 480 。

* + 螢幕顯示主要以絕對座標進行運算。考量硬體配備，一次最多只顯示四張地圖(左上、左下、右上、右下)。根據螢幕左上角之絕對座標計算出需顯示之地圖，若對應之MapXY陣列值為0或地圖格座標超出10x10的範圍則不顯示該背景圖片。
  + MapXY所記錄的地圖移動方向值介於0~15，編碼方式以 bit為單位。使用1,2,4,8進行編碼，其代表意義分別為，上、下、左、右。Ex：(5&1==1)為true，以及 (5&4==4)為true，所以在MapXY填入5表示該處可以向上及向左移動。

如此一來，不需要重複寫八個if判斷，只需得到對應的MapXY值就可以得知MapXY的移動方向。

* 物件資訊建立
  + 物件資訊主要由兩種資料組成，物件格座標-(MapObjXY) 與 碰撞資料陣列表-(Obstacle)組成。
  + 物件格座標每單位為8x3，每格代表此物件在絕對座標系統上的位置。物件格座標(ox,oy)轉絕對座標(x,y)之轉換方式為

x=ox\*8 ,y=oy\*3。

* + 碰撞資料表記錄物件格座標體系的物件資訊，每格儲存物件代碼、以及該物件整體的左上角格座標。其運算方式為

[(物件代碼)\*(2^22)+(oy+1)\*(2^11)+(ox+1)] 。

記錄左上角格座標是為了判斷物件與人物碰撞之後的後續處理動作。採用這樣的編碼能讓傳值與儲存更為方便，一個int就記錄所有物件資訊。

* + 由於物件格座標介於(0,0)~(640\*10/8,480\*10/3), 記錄與儲存物件碰撞表消耗資源過大。所以我們讀取採用讀寫檔的方式進行，並且先從MapXY判斷該位置是否有地圖路線。只有大於零的情況，才從Binary檔案中讀入或寫入資訊。
* CFrame Class
  + 定義一個物件的框架，有記錄最基本的wx,wy,width,height。
  + FixXY函式，做出相對運動的效果。當地圖被帶動後，此函式會依據地圖移動的方向與距離，對人物、物件、怪物做適當的修正。
  + IfCollision函式，傳入twx,twy,twidth,theight，傳回是否與自己有發生碰撞。由傳入的的資訊算出四個角的座標，與自己做四次判斷，如果有任何一次點座標在自己的方框內，則傳回true否則還要再判斷自己的四個角與對方是否在他方框內，如果有則傳回true；沒有則傳回false。
* AI\_Instruction 提供基本移動指令集以便於設計 Object和怪物 AI。
* Object Class

未被觸發之前的物件都是以資料的形式儲存在Map碰撞表，觸發之後會轉為object class。

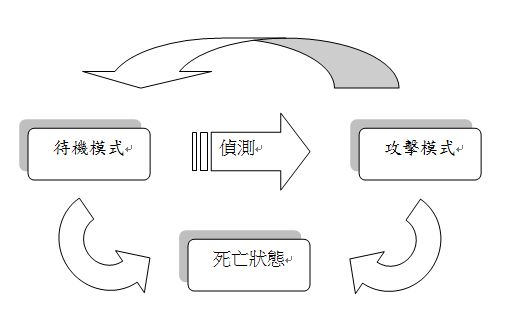
特性：Object都具有自己的主人(Owner)

每個物件都有基本的變數Direct, CanAttackMode, NowAct。

* + Direct變數，記錄物件該往那個方向移動，一樣採用1,2,4,8編碼。Object會自動依據Direct來呼叫相對應的AI\_Instruction指令。
  + CanAttackMode表示此物件可以攻擊的對象，搭配NowAct以產生物件的行為模式。
    - CanAttackMode = 0 => 任何人 + 怪物
    - CanAttackMode = 1 => 除了自己的擁有者外
    - CanAttackMode = 2 => 不攻擊任何人跟怪物
    - CanAttackMode = 3 => 只攻擊怪物
  + NowAct記錄物件的狀態
    - NowAct = 0 => 停屍間。

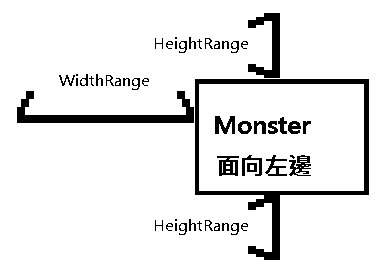
直接操作delete物件指標若有其他物件參考此物件則會有程式崩潰的危險。加入停屍間的緩衝區可以避免此情況。

在CGameStateRun中如果發現NowAct = 0再統一做delete指標的動作，並把參考指標轉向NULL。

* + - NowAct = 1 => 表示拿在人物手中。
    - NowAct = 2 or 4 => 表示此物體正處於運動狀態。其差別為NowAct = 2 物件顯示順序在怪物之前NowAct = 4 顯示在怪物之後。
    - NowAct = 3 => 表示此物件靜止在地上，該狀態會把物件加入物件碰撞表中進行碰撞判斷。
* Monster Class
  + 
  + 怪物使用螢幕格座標(wx,wy)進行運算。一開始建構時，會由目前螢幕位置去換算怪物的相對螢幕位置，並把NowAct設為0，而遊戲在執行的時候Monster.OnMove()每次都會呼叫FixXY()，當怪物自己發現自己的wx ,wy皆在640x480內的話，則會把NowAct 設為 1。
  + 怪物類別中提供Detect函式給怪物偵測人物的運算。呼叫時傳入\*player[]、WidthRange、HeightRange，並與怪物面向方向運算即可判斷並傳回離怪物最近的人物指標並傳回xy座標及距離，若沒有則傳回NULL。

P.S. ：WidthRange < 0則偵測怪物前後半個螢幕寬，而HeightRange < 0則偵測怪物目前所在地圖格座標中的一張螢幕高度範圍。

Ex ：



* + NowAct = 0 => 怪物尚未真實出現在遊戲中。
  + NowAct = 1 => 表示怪物活起來，並開始有各種行為。
  + NowAct 1~99 => 可以自行設計怪物。

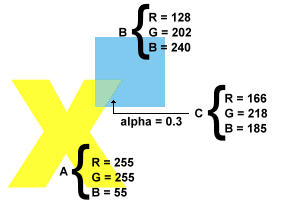
Ex：NowAct為多少時，會有甚麼樣的動作？並呼叫相對應的AI\_Instruct函式。

* + NowAct 100 , 101 => 可自行設計怪物死亡之後的該做的事情。100為怪物右邊被擊中死亡，101為怪物左邊被擊中死亡。
  + NowAct = -1 => 表示怪物死去。
* ToolCDC

Static介面以提供fade in、fade out轉場效果。

以半透明演算法搭配巨集函式實作alpha blending函式。

* + 半透明演算法



A = 原底圖  
B = 遮罩圖  
C = 新成像圖  
alpha = 透光率(0 ≦ alpha ≦ 1, 0為全遮住, 1為全透明)  
rgb = red, green, blue  
Cr = Br × (1 - alpha) + Ar × alpha  
Cg = Bg × (1 - alpha) + Ag × alpha  
Cb = Bb × (1 - alpha) + Ab × alpha

* + ALPHA(alpha , i , j )巨集函式

(((unsigned long)(((color[j][i]>>16)&0xFF)\*alpha)<<16) |

((unsigned long)(((color[j][i]>>8)&0xFF)\*alpha)<<8) |

((unsigned long)(((color[j][i])&0xFF)\*alpha)))

* 利用父類別的指標來操作子類別的指標，來達到操作多型函式。

**四、結語**

**1.問題及解決方法**

|  |  |
| --- | --- |
| 問題 | 解決辦法 |
| Class越來越龐大。 | 使用繼承來解決，把相同的概念抽出來，寫成Class並繼承他。 |
| 用switch實做多型，case越來越多寫到有點煩。 | 用父類別指標操作子類別指標，並配合virtual function做出OOP的多型技巧。 |
| 在Wire Class中宣告Election Class，且程式也有delete Wire Class指標，但在程式結束卻發生memory leak。 | 有時多型使用Virtual function，須要另外宣告此class的virtual解構元。 |
| MapManage在換關時，無法重新Loading新的背景圖片。 | 把Map改成指標，刪掉再重新動態配置就可以重新Loading。 |
| 當背景音樂停止，播放跳躍音效時程式會有Delay現象。 | 老師把CAudio改成thread方式執行，減少程式等待時間。 |
| MapManage要建立物件表的值，放在.cpp中會讓編譯速度變很慢。 | 程式開啟後另外從檔案loading進來即可。 |
| cpp大小越來越龐大 | 使用cpp、h分頁技巧。 |
| 人物的FallngDown 運算看程式碼看了好久還是找不出問題在哪邊。 | 使用Trace技巧，很簡單的就可以把值找出來，並加以修正。 |
| 不知道該把Basic 和 AI Instruction分成兩個Class還是應該寫成同一個。 | 方向一直都是錯的，這兩個根本就不應該被繼承，應該寫成一個instruction Class，並用has a的觀念讓人物、怪物、物件使用。雖然知道怎麼寫了，但是因為時間不夠，我們還沒來的及改正。 |
| 遊戲暫停後再開始，發現所有音效重播。 | 修改老師架構並且新增CAudio Class function 以滿足需求。 |

**2.時間表**

每周課餘撰寫程式時間(不含每個禮拜三小時的上課時間)

|  |  |  |  |
| --- | --- | --- | --- |
| 周次 | 陳科銘 | 謝宗廷 | 小組合計 |
| 1 | - | - | - |
| 2 | 8：45 | 11：30 | 20：15 |
| 3 | 11：40 | 13：00 | 24：40 |
| 4 | 15：30 | 16：00 | 31：30 |
| 5 | 12：30 | 10：10 | 22：40 |
| 6 | 04：30 | 04：00 | 08：30 |
| 7 | 07：15 | 07：00 | 14：15 |
| 8 | 09：55 | 10：30 | 20：25 |
| 9 | 05：00 | 6：30 | 11：30 |
| 10 | 20：45 | 12：40 | 33：25 |
| 11 | 13：40 | 14：20 | 28：00 |
| 12 | 00：15 | 09：30 | 09：45 |
| 13 | - | - | - |
| 14 | 03：40 | - | 03：40 |
| 15 | 20：50 | 22：45 | 43：35 |
| 16 | 13：00 | 17：00 | 30：00 |
| 總計 | 147：15 | 155：25 | 302：40 |
| 平均 | 9：12 | 9：42 | 18：54 |

**3.貢獻比例**

陳科銘：謝宗廷　=> 50%：50%

**4.檢核表**

|  |  |  |  |
| --- | --- | --- | --- |
|  | 項目 | 完成否 | 未完成原因 |
| 1 | 自定遊戲Icon | □已完成 □未完成 |  |
| 2 | 全螢幕啟動 | □已完成 □未完成 |  |
| 3 | 修改Help->About | □已完成 □未完成 |  |
| 4 | 初始畫面說明按鍵及滑鼠之用法與密技 | □已完成 □未完成 |  |
| 5 | 上傳setup 檔 | □已完成 □未完成 |  |
| 6 | 報告封面、側邊格式正確 | □已完成 □未完成 |  |

**5.收穫**

謝宗廷

* OOP的精神-繼承、多型、封裝
* TRACE及中斷點搭配逐步執行與不進入函式等偵錯技巧
* Friend Class 使用時機
* 繼承的使用時機(需要操作多型，父類別與子類別是is-a的關係)
* reuse程式碼的方式：has-a 水平呼叫、is-a 垂直繼承
* .h .cpp分檔技巧

.cpp檔注意include標頭檔的先後順序

* 以父類別指標操作多型

以父類別指標刪除子類別的物件，並不會呼叫到子類別的解構元。程式會執行父類別的解構元來delete。因此留下的是沒有父類別成員的子類別。不過在程式結束之後，主程式會回收這些殘缺不全的類別所使用的記憶體空間。如果子類別有另外使用動態配置則會出現memory leak。此時要父類別要提供一個virtual的解構元來呼叫子類別的解構元。但並不是表示virtual解構元可以濫用，程式運行這些virtual函式需要把指標存入stack，不必要的濫用會導致成本過高。

* virtual函式意義及使用方式

事實上virtual主要的使用時機是操作多型。但因為我們繼承架構的設計錯誤，我們需要讓父類別存取子類別的成員。我們錯誤的使用virtual函式傳子類別變數及指標給父類別使用。這點是需要避免的。

* delete效率不佳避免濫用

原本的想法是以delete再重新動態配置跑建構元的方式來初始化只有一個instance的類別。我們認為這樣意義上較為直覺以及方便乾淨。但我們並沒有考慮到程式在執行delete所處理的程序是很複雜的。由於遊戲是在PC上執行，記憶體成本相對較小。所以我們決定以記憶體換取效率

* 巨集函數優缺點

巨集函式能夠避免撰寫重複的CODE以節省開發成本，但過度使用巨集反而容易忽略OOP的精神。

* const int與define差異

均不占記憶體，前者是有資料形態的(區域)常數，後者為全域常數

* static修飾子使用及避免濫用

static使用在函式可以把變數隸屬於函式裡，編寫密技以及宣告一些counter很好用。Static使用在宣告class的member時，可以把變數隸屬於類別。對於把圖檔宣告成static可以避免instance過多而重複load圖。一些變數也可以宣告成static而賦予其他新的意義。Static與private constructor搭配組合是為singleton的基礎。

即使static實用性很高，還是不可以濫用static。程式部分已有濫用Public static的現象。Public static等效全域變數並不符封裝特性，要盡量避免。

* singleton design pattern 使用及了解優缺點

優：保證唯一的instance 缺：彈性小，回復singleton程序複雜

* 訓練出研究API的能力(mciSendcommandString, alphablending)

研究API讓我對STL容器有更深一層的認識，如list、iterator、map等。最後自己也有利用map

* 團隊溝通技巧

陳科銘

* 學習物件導向繼承觀念和public、 protected、 private的區別，也同時瞭解Has a 跟 Is a 應該怎麼分配與設計。
* Debug 的技巧，從 用眼睛慢慢 -> trace -> 下中斷點逐步執行，還學到this在debug中能看到許許多多的資訊。
* 物件導向與C語言的不同點在於用父類別儲存子類別指標，搭配virtual可以很簡單的實現在C語言中要寫一堆switch(){case：}的繁雜程式碼。
* #define和const的區別，兩者都是取代，只是第一種是沒有型別的，而第二種確有，除非逼不得已，才會使用#define，而const還可以分辨是這整個class要用，還是只有在這函式中要用，彈性比較夠。
* virtual的解構式的使用時機，並不是每個virtual fn都需要，因為創造這種解構式會多花上一倍記憶體，所以真的有memory leak時再用即可。
* CDC的各點存取與運算操作，與CDC效能差時，可以用將解析度的方式來增加效率，此舉動對視覺的影響並不大的觀念。
* static函式的使用。用在loading圖片檔可以減少loading時間，與對記憶體不必要的浪費，一個class大家用的圖片都一樣，就沒必要一人一份。還有singleton的設計，讓建構與解構都放在private只讓自己使用，並在class 宣告時並自己，能確保在這程式中是唯一且不滅。但這技術也不能濫用，須全面考慮清楚再使用，以免造成災難。
* 寫程式時遇到一個小問題但並不影響程式的功能，不要急著去把這問題解決，要先考慮成本與是否真的必要，還是學個經驗下次不要再犯即可。
* STL容器的使用，使程式創作更簡單方便，裡面有各式各樣的實用的函式等著我們去利用它，不用在自己寫。
* 記錄檔的用binary來做，這樣可以防止其他人輕易看出裡面的內容與修改其值。然而因Win7 資料夾權限問題，使得自己寫的程式如果要修改program file裡面的檔案會出現error。
* 團隊溝通上也有很大的進步，自己的想表達想法夥伴不一定一聽就懂，而這時的溝通，就需要用引導式的對話，帶著他慢慢往你的想法邁進，會比較有效率的完成兩個人的共識。

**6.心得、感想**

陳科銘：

一下時我對物件導向程式設計真的一點都不了解，老師在台上講的跟C語言沒差多少，就把scanf和printf改成cin 和cout，而物件導向只有稍微提到一些基本概念，沒有很多實際的操作與練習，大部分的練習跟C語言差不多。

這次的OOP實習，大概算是我第一次接觸真正的C++吧！靠著參考老師的範例，用破到不能再破的設計觀念，慢慢得寫出這次的作業，途中歷經了許多的困難與問題，漸漸的瞭解許多以前不會的語法與觀念，果然親自去體會從錯誤中學習，會是進步最快的方法。

在遊戲設計中，我們重新寫了好幾遍，總是覺得設計得不夠好、不夠周詳，而且許多觀念也都還停留在Ｃ語言，居然還想要用switch跟struct union實做出C++最好用的東西，用父類別指標來存取子類別指標的多型操作，還好寫沒幾個就請老師幫我們看看，不然這樣繼續做下去，很難想像我們還能加多少物件跟怪物進去，這個架構就爆炸了。

而我也會謹記著老師講過的觀念，有時路上或路旁有顆小石頭，不需去把它搬離，當真的石頭大到擋住所有去路，非不得已再把它移開，寫程式要考慮成本，該得過且過，下次注意不要再犯即可。但這次也許是練習吧，是還可以追求完美時期，我們花了許多時間與精神去研究它們與清除它們，雖然看起來很魯莽，但是能學到東西並能馬上加以操作練習並增加印象，感覺真得很棒。

我也非常喜歡老師這種教學，不奮不啟，不悱不發，這才有讀大學的感覺，不然什麼事情都被安排得好好，考試、上課枯燥乏味的教學，跟高中沒什麼兩樣？感覺修這門課真的是選對了！

謝宗廷：

這門課是OOP實習，但老實說，在第一次上課以前我不敢說我了解OOP。雖然一下已經先修了C++課程，但其實前半學期也只是將一些C語言用的語法轉換成使用C++函式庫。真正接觸物件導向的精隨是在後半學期，老師上課有講解封裝、多型、多載等物件導向原理，但在台下的我聽得很模糊。如老師在FrameWork的前言所說的：「寫再多的小程式，都很難讓學生深刻的體會到物件的用法與物件的互動，而且無法培養系統的觀念，更遑論開發軟體的能力。」

的確，修完C++的我並無法了解物件與物件間的互動精隨。對於那些名詞也僅有非常淺的認識。因此，修習這們課之前我已經有相當的心理準備。我要藉由這次的機會深入了解物件導向。

相較於我的組員陳科銘，我的程式基礎並沒有很穩固。但令人驚奇的，我們遇到的問題大部分能自己找出方法解決。我不會的，陳科銘可以教我。陳科銘不會的，我也能大概說個思考方向以便網路查詢進而解決問題。一開始我們就設立了非常高的目標，我們殷切的希望我們能夠達成。而學期開始不久後我們也真的下了很大的功夫撰寫程式，花費時數幾乎都領著全班高標移動。

我們很有熱情，但是我們對於物件導向的基礎不好。起初沒有經驗，架構除形還沒有完全考慮周全就開始動手CODING。當發現架構有很大的問題的時候程式已經寫了幾百行。幸好我與組員有相同的默契：砍掉重練。我們幾乎是不考慮時間成本只為了追求程式品質。學期到了第三周時，我們遊戲架構做了大幅度的修正三次。直到這時候我們程式才有了繼承架構。

由於這個學期我與組員也修了C語言，導致我們過度依賴C的用法。像是define全域常數以及使用switch實作多型。而忘了C++的const int與多型操作這麼重要的概念。

此時的我們，看似問題解決，實則危機四伏。包括不成熟的繼承架構，以及錯誤的virtual用法、還有程式中摻雜了許多C語言用法。但最危險的是─組員間的溝通開始產生了些問題。關於繼承架構，彼此的觀念差異開始浮現。彼此對於Code堅持的原則也不一。我對於程式的架構以及使用註解比較堅持。陳科銘則是著重於效率及演算法。還好我們克服了！我們是一個團隊，一個團隊就該是一個『整體』。所以我們要把彼此的想法完完全全讓對方理解。講出可能會產生不平衡的地方讓對方了解。溝通可以讓對方認同你的想法，或者找出自己的想法有哪些地方需要改進。於是，我們發明了「引導式溝通」。引導對方跟著你原先的思考步驟思考，我們發現這樣可以讓對方更快了解狀況，也能迅速產生共鳴。藉由這樣的溝通方式，我們克服了彼此觀念不同的問題。甚至在一次次的溝通中，我發現這個團隊由兩個人漸漸變成一個人了！

隨著深入研究遊戲架框，我們每次都會發現新的寫法新的驚奇。AlphaBlending、Virtual destructor 、mciCommandstring、多執行序、singleton design pattern等。知道越多就越覺得當初的技術有多麼不成熟。當初千心萬苦修改三次的主架構，後來發現也是錯的繼承架構。感謝老師在旁指導重要觀念。讓我們知道需考量的不只是這麼單純。

在你往前的道路上有一個十分礙眼的石頭，如果她沒有大到能夠阻礙你往前，那就略過她吧！越接近學期尾聲越能體會這句話的意思。因為要做的事太多了，所以需要考量時間成本。

雖然最後只有做完一關，雖然程式碼很明顯能看出不同時期的拙劣技巧， 我還是能夠大聲的說「我沒有遺憾」。也許是學期初的那份固執，也許是對於程式的熱情。總之，我收穫很多，真的！

**7.對於本課程的建議**

* 希望能增加alpha Blending介面，或者是支援png圖檔，讓我們有透明度可以選擇。

**五、附錄**

**mygame.h**

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 OnKeyDown(UINT, UINT, UINT); // 處理鍵盤Down的動作

protected:

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

int order;

CMovingBitmap pic[4];

};

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);

static void ToBonus(){TimeToBonus=PassMusicTime;}

static int multiplayer,Chip\_Dale,NowLevel; // multiplay 1= 1P ,2= 2P nowplayer 0 =Chip, 1= Dale

static ChipDale \*player[2];

static bool TimeToGo;

static bool IfViewer;

static bool IsBoss;

protected:

void OnMove(); // 移動遊戲元素

void OnShow(); // 顯示這個狀態的遊戲畫面

private:

CMovingBitmap saved,loading; // 背景圖

CMovingBitmap Health[3],ShowWho[3];

CFrame Frame;

MapManage \*Maps;

GameMapEdit EditPointer; //編輯箭頭

Monster \*AllMonster[ONE\_LEVEL\_MONSTER\_NUM]; // 全部怪物

Object \*AllThrow[CanTakeNum];

int EditMoveMode;

bool show\_save,show\_load;

static int TimeToBonus;

};

class CGameStateBonus : public CGameState {

public:

CGameStateBonus(CGame \*g);

~CGameStateBonus();

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

};

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

};

class CAudio {

public:

//自定義函式

void PauseById(int id);

void SetSpeedByID(unsigned int id,int speed);

void PlayOnSpe(unsigned,bool repeat=false,int time=0);

void Resume\_for\_CGame();

//Functions for Game\_ChipDale

void PlayLevelMusic(int level,bool stop\_and\_play=false); //限於GameStateRun使用，會自動偵測是否該播放BOSS音樂

void PauseLevelMusic(int level); //限於GameStateRun使用，會自動偵測是否該播放BOSS音樂

void StopLevelMusic(int level); //限於GameStateRun使用，背景音樂與BOSS音樂全部stop

void LevelMusicOnit(); //LevelMusic對照Map初始化(只能執行一次，有防呆)

private:

static map<int,int> LevelMusic;

};

class ToolCDC{

public:

ToolCDC();

static void Loading(); //為 ShowProgressBar 的圖片

static void Fadeout(){NowState=1;} //setState =1

static void Fadein(){NowState=4;} //setState =1

static void SaveCDC(); //記錄目前螢幕的 Colors

static int ReturnStage(){return NowState;} //傳回目前該做甚麼事情

static void ToNextState(); //移到下一個state

static void CDDraw();

static void ShowProgressBar(int WhichLevel,int percent);

private:

static COLORREF color[MHEIGHT][MWIDTH];

static CAnimation MovePointer;

static CMovingBitmap loading\_BG,loading\_Bar,loading\_Mask,level,letter[LEVELNUM];

static int NowState;

//0 未設定 ,

//1 下一次要OnShow要save & 鎖人物移動恩螢幕顯示, (淡出)

//2 saved 跟onMove說可以換人物位置 & 鎖人物移動恩螢幕顯示 ,

//3 鎖人物移動恩螢幕顯示

//4 下一次OnShow要save & 鎖人物移動恩螢幕顯示, (淡入)

//5 鎖人物移動恩螢幕顯示

static double alpha; //目前的aplha值

};

class GameMapEdit{

public:

GameMapEdit();

void Loading();

void OnShow(MapManage \*);

void SetOrder(int);

int nowOrder(){return Order;};

void OnMove(MapManage \*map);

void SetWxWy(int setWx,int setWy);

void SetMove(bool,bool,bool,bool,bool,int mode=0);

void SetObject(MapManage \*map); //去wx wy 位置放置obj

int ReturnWX(){return wx;}

int ReturnWY(){return wy;}

void FixXY(MapManage \*); // 修正螢幕位置

private:

int wx,wy;

int Order;

int ThisMoveTimesX,ThisMoveTimesY; // X、Y方向的移動次數

int ContinueMode; // TAB切換微調模式 0 1 2

bool move[4];

CMovingBitmap frame\_pic;

CMovingBitmap frame\_background[OrderSize+MonsterOrderSize];

};

class GameScore{

public:

static GameScore\* Instance();

~GameScore();

void Loading();

void ShowScroe();

void onShow();

bool Switch;

private:

GameScore();

static GameScore Game\_Score;

CMovingBitmap Head[2];

CMovingBitmap F,S;

CMovingBitmap L\_C,L\_D;

CInteger\* score\_p;

};

class CFrame{

public:

CFrame();

void SetWxWy(int wx,int wy,bool shiftmode=false);

virtual void FixXY(MapManage \*); // 修正螢幕位置

virtual void ReFixXY(MapManage \*); // 反修正螢幕位置

virtual void FixMapMove(int fixX,int fixY);

void SetWidthHeight(int tempW,int tempH);

virtual int IfCollision(int twx,int twy,int twidth,int theight);

int IfCollision(int Direct,int passSpeed); //與地圖做運算碰撞

virtual void CollisionReact(int setDirect,CFrame \*which){;}

int ReturnWX(){return wx;} // 回傳螢幕x座標

int ReturnWY(){return wy;} // 回傳螢幕y座標

int ReturnWidth(){return width;}

int ReturnHeight(){return height;}

bool ReturnLR(){return LRflag;}

static int Jump\_Fix;

private:

protected:

int wx,wy; // 框框左上角的螢幕座標 window x,y

int width,height;

bool flag[6]; // 上下左右AB 按鍵旗標

bool LRflag; // 0 左 1 右 標示方向

};

**mapmanage.h**

class MapManage{

public:

MapManage(int WhichLevel);

void OnShow();

void OnShowObject();

void LevelLoading(int WhichLevel,bool NotResetAll=false); //設定關卡 NotResetAll是否重設此關所有變數

int MoveMap(int direct,int MoveOneX=SPEED,int MoveOneY=SPEED);//move 1,2,4,8 ...

void SetMapXY(int,int); //設定 map 顯示位置

void SetObstacle(); //設定障礙物資料

void SetMonster(Monster \*monster[]); //設定怪物資料

void SetObj(int value,int wx,int wy); //wx,wy 螢幕座標

void SetRecord(){recordx=x;recordy=y;} //設定目前地圖左上角為紀錄點

int ReturnNowX(){return x;} //傳回螢幕左上角在地圖的x位置

int ReturnNowY(){return y;} //傳回螢幕左上角在地圖的y位置

int ReturnOBJ(int ox,int oy){return MapObjXY[oy][ox];} //回傳 Obj 的值

void SaveObj(int WhichLevel); //儲存 Obj 資訊

void ShowObstacle(int i,int j); //顯示指定的物件 ps: 給edit用的

void ClearisMoveMap(bool flag); //清除 isMoveMap 紀錄

void Set\_toRecord();

int IfCollision(int wx,int wy,int PicWidth,int PicHeight,bool ignore\_obj\_2=false,bool ignore\_eat=false,bool findNext=false,bool Reverse=false); //計算是否碰撞

int GetRoute(){return Route[y/MHEIGHT][x/MWIDTH];}

int GetRoute(int rx,int ry){return Route[ry][rx];}

void ClearObstacle(int Value); //把目標物件值告訴它 讓他直接去刪除碰撞機制

int FillObstacle(int Value,int setWx,int setWy,bool Visible=true); //把目標物件值告訴它 讓他直接去增加碰撞機制

static int isMoveMap; //地圖移動方向 1,2,4,8 上下左右

static int Teleport;

static int LRMargin;

friend class ChipDale;

private:

CMovingBitmap frame\_obstacle[OrderSize+MonsterOrderSize+UnVisibleOrderSize];

protected:

int Route[MAX\_YN][MAX\_XN]; //地圖路線移動方向 1,2,4,8 方向 16 出生點 32~ 64 ~boss

int MapObjXY[MAX\_OY][MAX\_OX]; //物件座標~顯示讀寫檔用

int Obstacle[MAX\_OY][MAX\_OX]; //障礙物 判斷碰撞用 //第一部分：種類 第二：左上Y 第三 左上X

CMovingBitmap MapSP[MAX\_YN][MAX\_XN];

int x,y; //目前螢幕左上角在地圖的座標

int recordx,recordy; //死亡起始位置

};

**instruction.h**

class AI\_Instructions : public CFrame{

public:

AI\_Instructions();

bool MoveUp(MapManage\* map);

bool MoveDown(MapManage\* map);

bool MoveRight(MapManage\* map);

bool MoveLeft(MapManage\* map);

void Jump();

int FallingDown(MapManage\* map);

protected:

int UpSpeed,MoveSPEED,JumpSPEED;

int LR\_Space;

bool IsJump;

bool NoCollision; // NoCollision = true 不要判斷碰撞

bool NoIgnore\_2,NoIgnore\_eat;

};

class BasicInstructions : public CFrame{

public:

BasicInstructions();

bool MoveRight(MapManage\* map);

bool MoveLeft(MapManage\* map);

bool Jump();

void Squat();

int FallingDown(MapManage\* map);

virtual ChipDale\* getPartner(){return NULL;}

virtual int CollisionChipDale(int Direct,int passSpeed,int Squeezemode=1){return false;}

virtual void TriggerObj(int Derict){TRACE("TriggerObj error\n");} // Derict吃花的方向

virtual int SetNowTaken(int Value){return -1;} //設定NowTaken

virtual ChipDale\* Instance(){return NULL;}

protected:

int UpSpeed,time\_jump; // time\_jump 避免二段跳

bool IsJump,IsRun,IsLessCollision; // isLessCollison = true 不要判斷地板2的碰撞(下跳觸發)

int ani\_jump\_count; // animation 跳轉蹲的計數器

int Reduce\_UP\_VELOCITY;

static bool canMoveMapX,canMoveMapY;

};

**ChipDale.h**

class ChipDale : public BasicInstructions{

public:

ChipDale(int isDale);

static void Loading();

void OnMove();

void GodMove();

void OnShow();

ChipDale\* Instance(){return this;}

int ReturnHideComplete(){return (animation[IsDale][1][4][LRflag].GetCurrentBitmapNumber()>=4)&&NowAct==4;}

int ReturnNowAct(){return NowAct;}

bool ReturnIsHurting(){return Hurt;}

int ReturnLastAct(){return LastAct;}

int ReturnIsTaken(){if(NowTaken>0)return 1;else return 0;}

int ReturnTakenByPartner(){if(Partner!=NULL&&Partner->NowTaken==ChipDale\_taken)return 1 ;else return 0;}

int ReturnHealth(){return Health;}

int ReturnLife(){return Life;}

int ReturnInvincible(){return Invincible;}

int ReturnIsDale(){return IsDale;}

int CollisionChipDale(int Direct,int passSpeed,int Squeezemode=1);

Object\* ReturnNowTakeObj(){return NowTakeObj;}

ChipDale\* getPartner(){return Partner;}

void setFlag(bool value,bool,bool,bool,bool,bool,bool); //值 上下左右AB

void SetState(); //設定Act LRdirect

void SetPartner(ChipDale \*setPlayer){Partner = setPlayer;}

void SetInvincible(int value){Invincible=value;} //設定Invincible 1無敵 0正常

void ReleaseNowTakeObj();

void FixSelf\_Onto\_ObjectTop();

void InitialWidthHeight(); //初始化高度

void Reset(int Wx,int Wy,bool LR=1,bool FullHealth=1);

void ResetScore(){Score\_Flower=Score\_Star=0;}

void Dead();

void Lock(bool value = true ){LOCK=value;}

void GetHurt();

void Faint();

void GodMode();

virtual int SetNowTaken(int Value); //設定NowTaken

virtual void TriggerObj(int Derict); //Derict吃花的方向

int Score\_Flower;

int Score\_Star;

static Object \*\*CanThrow;

static MapManage \*Maps;

static Monster \*\*AllMonster;

friend class BasicInstructions;

friend class ChipDale\_Dead;

friend class ChipDale\_Resurrect;

friend class ChipDale\_Taken;

private:

static CAnimation animation[2][2][ACTION\_NUM][2]; // 動作~方向

static CAnimation ani\_sweat[2],ani\_dizzy,ani\_god[2];

int IsDale;

int ani\_sweat\_x;

int ani\_sweat\_wy;

int ani\_sweat\_count;

int Health;

int Life;

int NowAct,LastAct; // Act 0站 1走 2跳 3丟 4蹲 5受傷 6暈眩

int NowTaken,LastTaken; // 0空手 1拿普通東西 6蘋果... (<0 作為再次拿取延遲器)

int freeze; // 丟東西的冷卻時間

bool Hurt; // 是否受傷

int IsFaint; // 是否暈眩 ,counter合一

bool ani\_Hide\_freeze\_jump; // 控制HIDE動畫未完成不能下跳，設定在CHIPDALE的動畫切換

int Last\_flag[6]; // 上一次按鍵的旗標，搭配flag做出放開按鍵判斷

int Invincible; // 1無敵 0正常 (>1作為ONSHOW的COUNTER)

Object \*NowTakeObj;

ChipDale \*Partner;

bool IsGod;

bool LOCK; // 鎖setFlag

bool Alive; // 鎖除了setFlag以外的函式

};

**object.h**

class Object : public AI\_Instructions{

public:

Object();

virtual ~Object(){;}

virtual void OnShow(MapManage\* map){;}

virtual void OnMove(MapManage\* map){;}

virtual void Throw(int setDirect){;}

int ReturnObjValue(){return ThisObjValue;}

int ReturnNowAct(){return NowAct;}

void SetNowAct(int setNowAct) {NowAct = setNowAct;}

void RecoverObj(ChipDale \*player); //重新拾獲時需呼叫

void CollisionMonster(Monster \*\*monster);

void CollisionChipDale(ChipDale \*player);

protected:

int CanAttackMode; // 攻擊對象 0 任何人 1 除了主人(預設值),2 None 3 只攻擊怪物

int Direct,LRflag,ThisObjValue,NowAct;

//ThisObjValue用來記錄刪除obstacle的編號

//NowAct =0 停屍間 1 拿在手上 2運動狀態 3放在地上 4打完怪物

ChipDale \*Owener;

};

class ChipDale\_Dead:public Object{

public:

ChipDale\_Dead(ChipDale\* chip);

~ChipDale\_Dead();

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

friend class ChipDale\_Resurrect;

private:

int isDale;

int Shine\_Count;

static CMovingBitmap frame\_pic[2][2];

int Wait\_A\_Minute;

};

class ChipDale\_Resurrect:public Object{

public:

ChipDale\_Resurrect(ChipDale\_Dead\* chip);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int isDale;

int timeCount;

static CMovingBitmap frame\_pic[2][2];

};

class ChipDale\_Taken:public Object{

public:

ChipDale\_Taken(ChipDale\* chip);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

private:

int isDale;

static CAnimation animation[2][2];

};

class Star : public Object{

public:

Star(int setWx,int setWy);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int lastUpSpeed;

static CMovingBitmap frame\_pic;

};

class Cheese : public Object{

public:

Cheese(int setWx,int setWy);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int lastUpSpeed;

static CMovingBitmap frame\_pic;

};

class Explosion : public Object{

public:

Explosion(int setOx,int setOy,int setChangeWhat,ChipDale \*player); //ChangeWhat 填代碼 (跟顯示物件代碼一樣)

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int ChangeWhat;

bool IfNeedReFix;

static CAnimation frame\_pic;

};

class Angel : public Object{

public:

Angel(int setWx,int setWy,ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int NowAttack;

bool OutLRflag;

const int TheBeeMaxSpeed;

static CAnimation frame\_pic[2];

static Monster \*\*monster;

};

class Apple : public Object{

public:

Apple(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

private:

static CMovingBitmap frame\_pic[2];

};

class Ball : public Object{

public:

Ball(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

private:

int Rebound\_times;

static CMovingBitmap frame\_pic[2];

};

class Craft : public Object{

public:

Craft(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

void CollisionReact(int setDirect,CFrame \*which);

~Craft(){TRACE("Craft destuctor run\n");}

private:

static CMovingBitmap frame\_pic[2];

};

class Stone : public Object{

public:

Stone(ChipDale \*player);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

void Throw(int setDirect);

void CollisionReact(int setDirect,CFrame \*which);

bool MoveRight(MapManage\* map);

bool MoveLeft(MapManage\* map);

void ReBound(MapManage\* map);

private:

bool HideThrow;

static CMovingBitmap frame\_pic[2];

int ReboundLR; // 0 無反彈 1 反彈右邊 -1反彈左邊

int ReboundSpeed;

};

class Greedy : public Object{

public:

Greedy(int setWx,int setWy,Object \*setRemoveObj);

static void Loading();

void OnShow(MapManage\* map);

void OnMove(MapManage\* map);

private:

int PassWx;

bool showMouse;

static CAnimation frame\_pic[2];

static CMovingBitmap frame\_hole;

Object \*CheeseObj;

};

**monster.h**

class Monster : public AI\_Instructions{

public:

Monster();

virtual ~Monster(){;}

void SetNowAct(int Value){NowAct = Value;}

int ReturnNowAct(){return NowAct;}

bool ReturnCanTrace(){return CanTrace;}

virtual void CollisionChipDale(ChipDale \*player);

virtual void OnShow(MapManage \*map){;}

virtual bool OnMove(MapManage \*map,ChipDale \*\*player){return false;}

virtual void SetMonster(MapManage \*map,int SetOx,int SetOy){;}

virtual bool KillMonster(int Direct){return false;}

ChipDale\* Detect(ChipDale \*\*player,int \*WLength,int \*HLength,int WRange,int HRange,bool IfTraceInvincible=false);

friend class ChipDale;

protected:

int LR\_flag,NowAct; //NowAct 正常介於1-100 100UP作為彈飛動畫

int lastWy,wait;

int Health;

bool CanTrace; //是否可以被跟蹤 給其他OBJ用 (ex: Object::Angel)

};

class bullet : public Monster{

public:

bullet();

virtual void OnShow(MapManage \*map){;}

bool OnMove(MapManage \*map);

int Collision(ChipDale \*player){return player->IfCollision(wx,wy,width,height);}

protected:

int MoveLR,MoveUD;

};

class MachineDog : public Monster{

public:

MachineDog(MapManage \*map,int SetOx,int SetOy);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

bool KillMonster(int Direct);

private:

static CAnimation frame\_monster[2][2]; //NowAct LRflag

};

class Cactus : public Monster{

public:

Cactus(MapManage \*map,int SetOx,int SetOy);

static void Loading();

bool OnMove(MapManage \*map,ChipDale \*\*player);

private:

static CMovingBitmap frame\_monster;

};

class Electric : public Monster{

public:

Electric(MapManage \*map,int SetWx,int SetWy,int SetWireOLength);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

int ElectricCollision(ChipDale \*player){return player->IfCollision(wx,wy,width,height);}

private:

int MoveLength;

int WireLength;

const int ChangeSpeed,MaxMoveSpeed;

static CMovingBitmap frame\_monster[2];

};

class Wire : public Monster{

public:

Wire(MapManage \*map,int SetOx,int SetOy,int SetWireLength);

~Wire(){delete(real\_monster);}

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

void CollisionChipDale(ChipDale \*player);

void FixMapMove(int fixX,int fixY);

private:

int WireLength;

Electric \*real\_monster;

};

class Mouse : public Monster{

public:

Mouse(MapManage \*map,int SetOx,int SetOy);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

bool KillMonster(int Direct);

private:

ChipDale \*tracePlayer;

static CAnimation frame\_monster[3][2]; //NowAct LRflag

};

class Wasp : public Monster{

public:

Wasp(MapManage \*map,int SetOx,int SetOy);

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

bool KillMonster(int Direct);

private:

int tLRMoveSPEED,tDownMoveSPEED;

static CAnimation frame\_monster[2];

};

class Lighting : public bullet{

public:

Lighting(MapManage \*map,int SetWx,int SetWy,int LR,int UD);

static void Loading();

void OnShow(MapManage \*map);

private:

static CMovingBitmap frame\_monster;

};

class Centipedelimbs : public Monster{

public:

Centipedelimbs(MapManage \*map,int SetWx,int SetWy,int SetSelect);

~Centipedelimbs();

static void Loading();

void OnShow(MapManage \*map,int countFlicker);

bool OnMove(MapManage \*map,ChipDale \*\*player);

int CentipedelimbsCollision(ChipDale \*player);

private:

static CAnimation frame\_monster[3];

bullet \*bullets[CentipedeLightingNum];

int Select;

const int TotalBullTime,ReleaseBullTime,BulletMaxSpeed,WRandBullet;

};

class Centipede : public Monster{

public:

Centipede(MapManage \*map,int SetOx,int SetOy);

~Centipede();

static void Loading();

void OnShow(MapManage \*map);

bool OnMove(MapManage \*map,ChipDale \*\*player);

void CollisionChipDale(ChipDale \*player);

void FixMapMove(int fixX,int fixY);

int IfCollision(int twx,int twy,int twidth,int theight){return real\_monster[0]->IfCollision(twx,twy,twidth,theight);}

bool KillMonster(int Direct);

private:

Centipedelimbs \*real\_monster[3];

int countFlicker;

static CAnimation frame\_monster;

};

**mygame.cpp**

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

GameScore::Instance()->Loading();

//logo.LoadBitmap(IDB\_BACKGROUND);

Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此Sleep

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

CAudio::Instance()->Load(AUDIO\_BEGIN, "sounds\\begin.mp3");

CAudio::Instance()->Load(AUDIO\_BUTTON, "sounds\\button.mp3");

pic[0].LoadBitmapA("Bitmaps\\BeginState\\1.bmp");

pic[1].LoadBitmapA("Bitmaps\\BeginState\\2.bmp");

pic[2].LoadBitmapA("Bitmaps\\BeginState\\3.bmp");

pic[3].LoadBitmapA("Bitmaps\\BeginState\\4.bmp");

}

void CGameStateInit::OnBeginState()

{

order=0;

}

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

{

const char KEY\_ESC = 27;

const char KEY\_ENTER = 13;

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

if(nChar == KEY\_RIGHT){

if(order==0){

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

order=1;

}

if(order==2){

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

order=3;

}

}

if(nChar == KEY\_LEFT){

if(order==1){

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

order=0;

}

if(order==3){

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

order=2;

}

}

if(nChar == KEY\_ENTER){

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

if(order==0)order=2;

else {

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

if(order==1){

CGameStateRun::multiplayer = 2;

GotoGameState(GAME\_STATE\_RUN);

}

else{

CGameStateRun::multiplayer = 1;

if(order==2)CGameStateRun::Chip\_Dale=0;

else CGameStateRun::Chip\_Dale=1;

GotoGameState(GAME\_STATE\_RUN);

}

}

}

}

void CGameStateInit::OnShow()

{

pic[order].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

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

int CGameStateRun::multiplayer=1;

int CGameStateRun::NowLevel=1;

int CGameStateRun::Chip\_Dale=0;

int CGameStateRun::TimeToBonus=0;

bool CGameStateRun::TimeToGo;

bool CGameStateRun::IfViewer=false;

bool CGameStateRun::IsBoss=false;

ChipDale\* CGameStateRun::player[2];

CGameStateRun::CGameStateRun(CGame \*g)

: CGameState(g)

{

IfViewer = false;

NowLevel=1;

show\_save = false;

show\_load = false;

EditMoveMode = 0;

Maps = NULL;

player[0] = NULL;

player[1] = NULL;

for(int i=0;i<CanTakeNum;i++) //這只需寫一次 初始化陣列

AllThrow[i]=NULL;

for(int i=0;i<ONE\_LEVEL\_MONSTER\_NUM;i++) //這只需寫一次 初始化陣列

AllMonster[i]=NULL;

ChipDale::CanThrow = AllThrow;

ChipDale::AllMonster = AllMonster;

}

CGameStateRun::~CGameStateRun()

{

if(Maps!=NULL) delete(Maps);

for(int i=0;i<ONE\_LEVEL\_MONSTER\_NUM && AllMonster[i]!=NULL;i++){

delete(AllMonster[i]);

}

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

if(AllThrow[i]==NULL)continue;

delete(AllThrow[i]);

}

for(int i=0;i<2 && player[i]!=NULL;i++)

delete(player[i]);

}

void CGameStateRun::OnBeginState()

{

int initial\_Y;

TimeToGo=false;

EditPointer.SetWxWy(0,0);

Maps = new MapManage(NowLevel);

CAudio::Instance()->PlayLevelMusic(0,true);

CAudio::Instance()->PauseLevelMusic(0);

CAudio::Instance()->Play(AUDIO\_1\_1A);

ChipDale::Maps = Maps;

Maps->SetObstacle();

Maps->SetMonster(AllMonster);

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

if(AllThrow[i]!=NULL){

delete AllThrow[i];

AllThrow[i]=NULL;

}

}

if(player[0]!=NULL)delete player[0];

if(player[1]!=NULL)delete player[1];

player[0] = new ChipDale(Chip\_Dale);

if(multiplayer==2){

player[1] = new ChipDale(1);

}

else

player[1]=NULL;

player[0]->SetPartner(player[1]);

if(multiplayer==2) player[1]->SetPartner(player[0]);

initial\_Y = (MapManage::Teleport>>7)/DesSize;//得到出生的高 , DesSize 表示 Des 最大的值(128)

TRACE("initial\_Y: %d\n",initial\_Y);

player[0]->InitialWidthHeight();

player[0]->SetWxWy(80,initial\_Y-player[0]->ReturnHeight());

if(multiplayer==2){

player[1]->InitialWidthHeight();

player[1]->SetWxWy(80+player[1]->ReturnWidth(),initial\_Y-player[1]->ReturnHeight());

}

}

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

{

// 如果希望修改cursor的樣式，則將下面程式的commment取消即可

// SetCursor(AfxGetApp()->LoadCursor(IDC\_GAMECURSOR));

//TRACE("1\n");

int des,initial\_Y,fixX,fixY;

int toolCDC\_State = ToolCDC::ReturnStage();

static int music\_count=0;

if(music\_count<6\*33-5)music\_count++;

if(music\_count==6\*33-5){

CAudio::Instance()->PlayLevelMusic(NowLevel,true);

music\_count++;

}

if(GameScore::Instance()->Switch)return;

if(TimeToGo||TimeToBonus==1){

CAudio::Instance()->StopLevelMusic(NowLevel);

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

if(TimeToGo){

player[0]->ResetScore();

if(player[1]!=NULL)player[1]->ResetScore();

TimeToGo=false;

GotoGameState(GAME\_STATE\_OVER);

}else{

TimeToBonus=false;

GotoGameState(GAME\_STATE\_BONUS);

}

}

if(TimeToBonus>0){

if(TimeToBonus==PassMusicTime)

{

CAudio::Instance()->StopLevelMusic(NowLevel);

CAudio::Instance()->Play(AUDIO\_1\_PASS);

}

TimeToBonus--;

}

if(toolCDC\_State==0){

if(!IfViewer){

//人物移動

if(!TimeToBonus){

player[0]->OnMove();

if(multiplayer==2)

player[1]->OnMove();

}

//取消注解此行可得人物Y軸

//static int lastNowY = 0;int tempNowY = //player[0]->ReturnWY()+player[0]->ReturnHeight();if(lastNowY!=tempNowY){TRACE("now\_Y: //%d\n",tempNowY);lastNowY=tempNowY;}

//人物修正

player[0]->FixXY(Maps);

if(multiplayer==2)

player[1]->FixXY(Maps);

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

if(AllThrow[i]==NULL)continue;

if(AllThrow[i]->ReturnNowAct()==0){

delete(AllThrow[i]);

AllThrow[i]=NULL;

continue;

}

if(AllThrow[i]->ReturnNowAct()>=0)//有時候不想FIX修正位置 可以讓 NowAct 為負的!!!

AllThrow[i]->FixXY(Maps);

AllThrow[i]->OnMove(Maps);

}

//怪物移動+修正+碰撞測試

for(int i=0;i<ONE\_LEVEL\_MONSTER\_NUM && AllMonster[i]!=NULL;i++){

if(AllMonster[i]->OnMove(Maps,player)){

AllMonster[i]->CollisionChipDale(player[0]);

if(multiplayer==2)

AllMonster[i]->CollisionChipDale(player[1]);

}

}

//碰撞 物件 怪物 人物 測試

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

if(AllThrow[i]==NULL)continue;

AllThrow[i]->CollisionMonster(AllMonster);

AllThrow[i]->CollisionChipDale(player[0]);

if(multiplayer==2)

AllThrow[i]->CollisionChipDale(player[1]);

}

}

else{

EditPointer.OnMove(Maps);

EditPointer.FixXY(Maps);

}

}

else if(toolCDC\_State==2){

des = (Maps->Teleport)%DesSize;//得到出生的高 , DesSize 表示 Des 最大的值(128)

initial\_Y = (Maps->GetRoute((des-1)%10,(des-1)/10)>>7)/DesSize;

TRACE("initial\_Y: %d\n",initial\_Y);

fixX = -Maps->ReturnNowX()+(des-1)%MAX\_XN\*MWIDTH;

fixY = -Maps->ReturnNowY()+(des-1)/MAX\_YN\*MHEIGHT;

Maps->SetMapXY((des-1)%MAX\_XN\*MWIDTH ,(des-1)/MAX\_YN\*MHEIGHT);

Maps->SetRecord();

for(int i=0;i<ONE\_LEVEL\_MONSTER\_NUM && AllMonster[i]!=NULL;i++){

AllMonster[i]->FixMapMove(fixX,fixY);

}

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

if(AllThrow[i]==NULL)continue;

if(AllThrow[i]->ReturnNowAct()!=1){

delete(AllThrow[i]);

AllThrow[i]=NULL;

}

}

player[0]->Reset(80,initial\_Y-player[0]->ReturnHeight(),true,false);

if(multiplayer==2)

player[1]->Reset(125,initial\_Y-player[1]->ReturnHeight(),true,false);

ToolCDC::ToNextState();

}else if(toolCDC\_State==4){

if(Maps->GetRoute()&64){

IsBoss=true;

CAudio::Instance()->PlayLevelMusic(NowLevel,true);

}else{

IsBoss=false;

}

}

}

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

//

Health[0].LoadBitmapA("Bitmaps/StateRun/health\_1.bmp",PURPLE);

Health[1].LoadBitmapA("Bitmaps/StateRun/health\_2.bmp",PURPLE);

Health[2].LoadBitmapA("Bitmaps/StateRun/health\_3.bmp",PURPLE);

ShowWho[0].LoadBitmapA("Bitmaps/StateRun/chip\_L.bmp",PURPLE);

ShowWho[1].LoadBitmapA("Bitmaps/StateRun/dale\_L.bmp",PURPLE);

ShowWho[2].LoadBitmapA("Bitmaps/StateRun/dale\_R.bmp",PURPLE);

ShowInitProgress(40);

ChipDale::Loading();

EditPointer.Loading();

ToolCDC::Loading();

ShowInitProgress(45);

//怪物圖片loading

MachineDog::Loading();

Cactus::Loading();

Wire::Loading();

Mouse::Loading();

Wasp::Loading();

Centipede::Loading();

//End Loading

ShowInitProgress(50);

//物件圖片loading

Apple::Loading();

Stone::Loading();

Craft::Loading();

Star::Loading();

Explosion::Loading();

Angel::Loading();

ChipDale\_Dead::Loading();

ChipDale\_Resurrect::Loading();

ChipDale\_Taken::Loading();

Greedy::Loading();

Cheese::Loading();

Ball::Loading();

//End Loading

ShowInitProgress(75);

saved.LoadBitmapA("Bitmaps/object/save.bmp",0xffffff);

loading.LoadBitmapA("Bitmaps/object/loading.bmp",0xffffff);

// 完成部分Loading動作，提高進度

ShowInitProgress(80);

// 繼續載入其他資料

CAudio::Instance()->LevelMusicOnit();

CAudio::Instance()->Load(AUDIO\_JUMP, "sounds/jump.mp3");

CAudio::Instance()->Load(AUDIO\_EDIT, "sounds/edit.mp3");

CAudio::Instance()->Load(AUDIO\_DEAD, "sounds/dead.mp3");

CAudio::Instance()->Load(AUDIO\_1\_1, "sounds/state/1/level\_1B.mp3");

CAudio::Instance()->Load(AUDIO\_1\_BOSS,"sounds/state/1/boss\_1B.mp3");

CAudio::Instance()->Load(AUDIO\_1\_1A, "sounds/state/1/level\_1A.mp3");

CAudio::Instance()->Load(AUDIO\_1\_PASS,"sounds/state/1/level\_1pass.mp3");

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

}

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

{

//密技區 宣告

static char GodMode[]="GOD";static int GodModeI = 0;

static char EditMode[]="EDIT";static int EditModeI = 0;

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

//密技區 實做

if(nChar==GodMode[GodModeI]){

GodModeI++;

if(GodMode[GodModeI]==0){

player[0]->GodMode();

GodModeI=0;

}

}

else GodModeI=0;

if(nChar==EditMode[EditModeI]){// CapsLock 切換人物與造物者模式

EditModeI++;

if(EditMode[EditModeI]==0){

if(player[0]->ReturnHealth()<=0||multiplayer==2){EditModeI=0;return;}

IfViewer = !IfViewer;

if(IfViewer){

CAudio::Instance()->PlayLevelMusic(0);

}else{

CAudio::Instance()->PlayLevelMusic(NowLevel);

}

player[0]->Reset(EditPointer.ReturnWX(),EditPointer.ReturnWY(),player[0]->ReturnLR(),false);

if(multiplayer==2){

player[1]->Reset(EditPointer.ReturnWX()+player[1]->ReturnWidth(),EditPointer.ReturnWY(),player[1]->ReturnLR(),false);

}

EditPointer.SetMove(false,true,true,true,true,EditMoveMode);

if(IfViewer){

Maps->LevelLoading(NowLevel,true);

Maps->SetMapXY(Maps->ReturnNowX()/ONEOBJX\*ONEOBJX,Maps->ReturnNowY()/ONEOBJY\*ONEOBJY);

}

else{

Maps->SetMapXY(Maps->ReturnNowX()/SPEED\*SPEED,Maps->ReturnNowY()/SPEED\*SPEED);

Maps->SetObstacle();

Maps->SetMonster(AllMonster);

player[0]->ReleaseNowTakeObj();

if(multiplayer==2)

player[1]->ReleaseNowTakeObj();

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

if(AllThrow[i]==NULL)continue;

if(AllThrow[i]->ReturnNowAct()>=0){

delete(AllThrow[i]);

AllThrow[i]=NULL;

}

}

}

EditModeI = 0;

}

}else EditModeI = 0;

//一般按鍵觸發

if(!IfViewer){

if(multiplayer==1)

player[0]->setFlag(true,(nChar==KEY\_UP),(nChar == KEY\_DOWN),(nChar == KEY\_LEFT),(nChar == KEY\_RIGHT),(nChar == 'Z'),(nChar == 'X'));

if(multiplayer==2){

player[0]->setFlag(true,(nChar==KEY\_UP),(nChar == KEY\_DOWN),(nChar == KEY\_LEFT),(nChar == KEY\_RIGHT),(nChar == 'N'),(nChar == 'M'));

player[1]->setFlag(true,(nChar==87),(nChar == 83),(nChar == 65),(nChar == 68),(nChar == 90),(nChar == 88));

}

if(nChar==9)GameScore::Instance()->ShowScroe();

}

else{

show\_save = false;

EditPointer.SetMove(true,(nChar == KEY\_UP),(nChar == KEY\_DOWN),(nChar == KEY\_LEFT),(nChar == KEY\_RIGHT),EditMoveMode);

if(nChar=='Z') EditPointer.SetOrder(EditPointer.nowOrder()-1);

if(nChar=='X') EditPointer.SetOrder(EditPointer.nowOrder()+1);

if(nChar==13) EditPointer.SetObject(Maps); // Enter 放置

if(nChar==9) EditMoveMode = (EditMoveMode+1)%3; // TAB 切換模式

if(nChar==27) {// ESC 儲存

Maps->SaveObj(NowLevel);

show\_save=true;

}

}

}

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

{

//TRACE("3\n");

const char KEY\_LEFT = 0x25; // keyboard左箭頭

const char KEY\_UP = 0x26; // keyboard上箭頭

const char KEY\_RIGHT = 0x27; // keyboard右箭頭

const char KEY\_DOWN = 0x28; // keyboard下箭頭

if(!IfViewer){

if(multiplayer==1)

player[0]->setFlag(false,(nChar==KEY\_UP),(nChar == KEY\_DOWN),(nChar == KEY\_LEFT),(nChar == KEY\_RIGHT),(nChar == 'Z'),(nChar == 'X'));

if(multiplayer==2){

player[0]->setFlag(false,(nChar==KEY\_UP),(nChar == KEY\_DOWN),(nChar == KEY\_LEFT),(nChar == KEY\_RIGHT),(nChar == 'N'),(nChar == 'M'));

player[1]->setFlag(false,(nChar==87),(nChar == 83),(nChar == 65),(nChar == 68),(nChar == 90),(nChar == 88));

}

if(nChar==9)GameScore::Instance()->ShowScroe();

}

else{

EditPointer.SetMove(false,(nChar == KEY\_UP),(nChar == KEY\_DOWN),(nChar == KEY\_LEFT),(nChar == KEY\_RIGHT));

}

}

void CGameStateRun::OnShow()

{

// 注意：Show裡面千萬不要移動任何物件的座標，移動座標的工作應由Move做才對，

// 否則當視窗重新繪圖時(OnDraw)，物件就會移動，看起來會很怪。換個術語

// 說，Move負責MVC中的Model，Show負責View，而View不應更動Model。

// 貼上背景圖、撞擊數、球、擦子、彈跳的球

int toolCDC\_State = ToolCDC::ReturnStage();

if(GameScore::Instance()->Switch){

GameScore::Instance()->onShow();

return;

}

if(toolCDC\_State==0 || toolCDC\_State==1 || toolCDC\_State==4){

Maps->OnShow();

if(!IfViewer){

for(int i=0;i<CanTakeNum;i++){//物件顯示靜止 NowAct=3 顯是在怪物之後

if(AllThrow[i]==NULL)continue;

if(AllThrow[i]->ReturnNowAct()!=3)continue;

AllThrow[i]->OnShow(Maps);

}

for(int i=0;i<ONE\_LEVEL\_MONSTER\_NUM && AllMonster[i]!=NULL;i++){//怪物顯示

AllMonster[i]->OnShow(Maps);

}

for(int i=0;i<CanTakeNum;i++){//物件顯示靜止 NowAct=5 靜止在地上 顯是在怪物之前

if(AllThrow[i]==NULL)continue;

if(AllThrow[i]->ReturnNowAct()!=5)continue;

AllThrow[i]->OnShow(Maps);

}

Maps->OnShowObject();

//物件顯示在手中 NowAct=1

if(player[0]->ReturnNowTakeObj()!=NULL)

player[0]->ReturnNowTakeObj()->OnShow(Maps);

if(multiplayer==2 && player[1]->ReturnNowTakeObj()!=NULL){

player[1]->ReturnNowTakeObj()->OnShow(Maps);

}

player[0]->OnShow();//人物顯示

if(multiplayer==2)

player[1]->OnShow();

for(int i=0;i<CanTakeNum;i++){//物件顯示運動中 NowAct = 2 4

if(AllThrow[i]==NULL)continue;

if(AllThrow[i]->ReturnNowAct()!=2 && AllThrow[i]->ReturnNowAct()!=4)continue;

AllThrow[i]->OnShow(Maps);

}

ShowWho[Chip\_Dale].SetTopLeft(0,0);

ShowWho[Chip\_Dale].ShowBitmap();

switch(player[0]->ReturnHealth()){//血量顯示

case 1 : Health[0].SetTopLeft(0,0);Health[0].ShowBitmap();break;

case 2 : Health[1].SetTopLeft(0,0);Health[1].ShowBitmap();break;

case 3 : Health[2].SetTopLeft(0,0);Health[2].ShowBitmap();break;

}

if(multiplayer==2){

ShowWho[2].SetTopLeft(MWIDTH-ShowWho[2].Width(),0);

ShowWho[2].ShowBitmap();

switch(player[1]->ReturnHealth()){//血量顯示

case 1 : Health[0].SetTopLeft(MWIDTH-ShowWho[2].Width(),0);Health[0].ShowBitmap();break;

case 2 : Health[1].SetTopLeft(MWIDTH-ShowWho[2].Width(),0);Health[1].ShowBitmap();break;

case 3 : Health[2].SetTopLeft(MWIDTH-ShowWho[2].Width(),0);Health[2].ShowBitmap();break;

}

}

}

else

EditPointer.OnShow(Maps);

//ClearisMoveMap 一定要放在所有要 show 物件的最後面

Maps->ClearisMoveMap(IfViewer);

if(show\_save)

saved.ShowBitmap();

if(toolCDC\_State==1||toolCDC\_State==4){

ToolCDC::SaveCDC();

ToolCDC::ToNextState();

if(toolCDC\_State==4) ToolCDC::CDDraw();

}

}

else{

ToolCDC::CDDraw();

ToolCDC::ToNextState();

}

}

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

// 這個class為遊戲的結束狀態(Game Bonus)

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

CGameStateBonus::CGameStateBonus(CGame\* g)

: CGameState(g)

{

TRACE("BonusConstruct\n");

}

CGameStateBonus::~CGameStateBonus(){

TRACE("~CGameStateBonus()\n");

}

void CGameStateBonus::OnInit(){

TRACE("BonusInit\n");

}

void CGameStateBonus::OnBeginState(){

TRACE("BonusOnBeginState\n");

}

void CGameStateBonus::OnMove(){}

void CGameStateBonus::OnShow(){}

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

// 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){

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

GotoGameState(GAME\_STATE\_INIT);

}

}

void CGameStateOver::OnBeginState()

{

counter = 30 \* 5; // 5 seconds

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

}

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(90); // 接個前一個狀態的進度，此處進度視為66%

// 開始載入資料

//Sleep(300); // 放慢，以便看清楚進度，實際遊戲請刪除此Sleep

// 最終進度為100%

ShowInitProgress(100);

CAudio::Instance()->Load(AUDIO\_GAMEOVER,"sounds\\gameover.mp3");

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

}

void CGameStateOver::OnShow()

{

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

}

void CAudio::Resume\_for\_CGame()

{

int state=CGame::Instance()->WhichState();

if(state==GAME\_STATE\_RUN){

if(CGameStateRun::IfViewer)

PlayLevelMusic(0);

else

{

PlayLevelMusic(CGameStateRun::NowLevel);

}

}else

Resume();

}

void CAudio::PlayLevelMusic(int level,bool stop\_and\_play){

if (!Instance()->isOpened)

return;

int Notboss;

if(CGameStateRun::IsBoss)Notboss=0;

else Notboss=1;

if(level==0)Notboss=0;

char command[MAX\_MCI\_COMMAND\_SIZE];

if(stop\_and\_play)

sprintf(command, "play device%d from 0 repeat", LevelMusic.at(level\*2-Notboss));

else

sprintf(command, "resume device%d", LevelMusic.at(level\*2-Notboss));

Instance()->SendMciCommand(command);

if(level!=0){

Instance()->PauseById(LevelMusic[0]);

if(Notboss==0)Instance()->PauseById(LevelMusic[level\*2-1]);

}

else{

Instance()->PauseById(LevelMusic[CGameStateRun::NowLevel\*2-1]);

Instance()->PauseById(LevelMusic[CGameStateRun::NowLevel\*2]);

}

}

void CAudio::PauseLevelMusic(int level){

if (!Instance()->isOpened)

return;

int Notboss;

if(CGameStateRun::IsBoss)Notboss=0;

else Notboss=1;

if(level==0)Notboss=0;

char command[MAX\_MCI\_COMMAND\_SIZE];

sprintf(command, "pause device%d wait", LevelMusic.at(level\*2-Notboss));

Instance()->SendMciCommand(command);

}

void CAudio::StopLevelMusic(int level){

if (!Instance()->isOpened)

return;

char command[MAX\_MCI\_COMMAND\_SIZE];

sprintf(command, "stop device%d", LevelMusic.at(level\*2-1));

Instance()->SendMciCommand(command);

sprintf(command, "stop device%d", LevelMusic.at(level\*2));

Instance()->SendMciCommand(command);

}

void CAudio::LevelMusicOnit(){

static bool IsExecuted=false;

if(IsExecuted)return;

LevelMusic.insert(pair<int,int>(0,AUDIO\_EDIT));

LevelMusic.insert(pair<int,int>(1,AUDIO\_1\_1));

LevelMusic.insert(pair<int,int>(2,AUDIO\_1\_BOSS));

IsExecuted=true;

}

void CAudio::PauseById(int id)

{

int j=0;

if (!isOpened)

return;

map<int, Info>::iterator i;

i=info.find(id);

if (i->second.isGood) {

char command[MAX\_MCI\_COMMAND\_SIZE];

sprintf(command, "pause device%d wait", i->first);

SendMciCommand(command);

}

}

void CAudio::SetSpeedByID(unsigned int id,int speed){

if (!isOpened)

return;

GAME\_ASSERT(info.find(id) != info.end(), "Can not play back audio: incorrect Audio ID!");

if (!info[id].isGood)

return;

char command[400];

sprintf(command, "set device%d speed %d", id,speed);

SendMciCommand(command);

}

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

// ToolCDC

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

COLORREF ToolCDC::color[MHEIGHT][MWIDTH];

CAnimation ToolCDC::MovePointer;

CMovingBitmap ToolCDC::loading\_BG,ToolCDC::loading\_Bar,ToolCDC::loading\_Mask;

CMovingBitmap ToolCDC::letter[LEVELNUM],ToolCDC::level;

int ToolCDC::NowState=0;

double ToolCDC::alpha=0.0;

ToolCDC::ToolCDC(){}

void ToolCDC::Loading(){

char name[100];

MovePointer.SetDelayCount(20);

MovePointer.AddBitmap("Bitmaps/action/","Chip/","run\_1R.bmp",PURPLE);

MovePointer.AddBitmap("Bitmaps/action/","Chip/","run\_2R.bmp",PURPLE);

MovePointer.AddBitmap("Bitmaps/action/","Chip/","run\_3R.bmp",PURPLE);

MovePointer.AddBitmap("Bitmaps/action/","Chip/","run\_4R.bmp",PURPLE);

loading\_BG.LoadBitmapA("Bitmaps/BeginState/loadingBG.bmp",PURPLE);

loading\_Bar.LoadBitmapA("Bitmaps/BeginState/loadingBar.bmp",PURPLE);

loading\_Mask.LoadBitmapA("Bitmaps/BeginState/loadingMask.bmp",PURPLE);

level.LoadBitmapA("Bitmaps/BeginState/Level.bmp",PURPLE);

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

sprintf(name,"Bitmaps/BeginState/Letter%d.bmp",i+1);

letter[i].LoadBitmapA(name,PURPLE);

}

}

void ToolCDC::CDDraw()

{

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

for(int i=0;i<MWIDTH;i+=1){

for(int j=0;j<MHEIGHT;j+=1){

pDC->SetPixel(i,j,ALPHA(alpha,i,j));

}

}

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

}

void ToolCDC::SaveCDC()

{

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

for(int i=0;i<MWIDTH;i+=1){

for(int j=0;j<MHEIGHT;j+=1){

if(i%2==0&&j%2==0)

color[j][i] =pDC->GetPixel(i,j);

else

color[j][i] = color[j/2\*2][i/2\*2];

}

}

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

if(NowState<=2) alpha = 0.75;

if(NowState>=3) alpha = 0.25;

}

void ToolCDC::ToNextState()

{

switch(NowState){

case 1:case 2:case 4:

NowState += 1;

break;

case 3:

if(alpha!=0.0){

alpha -= 0.25;

if(alpha<0.0)alpha=0.0;

}

else{

NowState += 1;

}

break;

case 5:

if(alpha!=1.0){

alpha += 0.25;

if(alpha>1.0)alpha=1.0;

}

else{

NowState = 0;

}

break;

}

}

void ToolCDC::ShowProgressBar(int WhichLevel,int percent){

int width = MovePointer.Width(),height = MovePointer.Height();

if(percent>100)percent = 100;

CDDraw::BltBackColor(DEFAULT\_BG\_COLOR); // 將 Back Plain 塗上預設的顏色

loading\_Mask.SetTopLeft(95,275);

loading\_Mask.ShowBitmap();

loading\_Bar.SetTopLeft(95-450\*(100-percent)/100,275);

loading\_Bar.ShowBitmap();

loading\_BG.SetTopLeft(0,0);

loading\_BG.ShowBitmap();

MovePointer.SetBottomLeft(95+450\*percent/100-(width)/2,275-(height-30)/2,height);

MovePointer.OnShow();

MovePointer.OnMove();

level.SetTopLeft(160,140);

level.ShowBitmap();

letter[WhichLevel-1].SetTopLeft(415,140);

letter[WhichLevel-1].ShowBitmap();

CDDraw::BltBackToPrimary(); // 將 Back Plain 貼到螢幕

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//GameMapEdit實作 \*/

// \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

GameMapEdit::GameMapEdit(){

Order = 0;

wx = wy = 0;

for(int i=0;i<4;i++)

move[i]=false;

ContinueMode = 0;

}

void GameMapEdit::Loading(){

char temp[100];

frame\_pic.LoadBitmapA("Bitmaps/object/Pointer.bmp");

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

sprintf(temp,"Bitmaps/object/%d.bmp",i);

frame\_background[i].LoadBitmapA(temp,PURPLE);

}

for(int i=-1;i>=-MonsterOrderSize;i--){

sprintf(temp,"Bitmaps/monster/%d.bmp",-i);

frame\_background[-i+OrderSize-1].LoadBitmapA(temp,PURPLE);

}

}

void GameMapEdit::OnShow(MapManage \*map){

int temp;

//顯示 目前螢幕上的OBJ

for(int i=0;i<=(MWIDTH+2\*WinShowBuffer)/ONEOBJX;i++)

for(int j=0;j<=(MHEIGHT+2\*WinShowBuffer)/ONEOBJY;j++)

map->ShowObstacle(i,j);

//顯示 目前位置

temp = (Order<0)\*(-Order+OrderSize-1) + (Order>=0)\*(Order);

if(Order==0)

frame\_background[0].SetTopLeft(wx-16,wy-16);

else

frame\_background[temp].SetTopLeft(wx,wy);

frame\_background[temp].ShowBitmap();

frame\_pic.SetTopLeft(wx,wy);

frame\_pic.ShowBitmap();

}

void GameMapEdit::SetOrder(int value){

if (value < OrderSize && value >= -MonsterOrderSize)

Order = value;

if (value < -MonsterOrderSize)

Order = OrderSize-1;

if (value >=OrderSize)

Order = -MonsterOrderSize;

}

void GameMapEdit::OnMove(MapManage \*map)

{

int width,height,temp; //目前圖片高度和寬度

int timesX,timesY;

if(ContinueMode==0) {timesX=1; timesY=3; }

else if(ContinueMode==1){timesX=1; timesY=1; }

else if(ContinueMode==2){timesX=-1;timesY=-1;}

temp = (Order<0)\*(-Order+OrderSize-1) + (Order>=0)\*(Order);

width = frame\_background[temp].Width();

height = frame\_background[temp].Height();

if(timesX==-1||timesY==-1){

timesX = width/ONEOBJX;

timesY = height/ONEOBJY;

}

static int limitX = (MWIDTH + width)/2;

static int limitY = (MHEIGHT + height)/2;

if(move[0]){

wy -= ONEOBJY\*timesY;

if(wy < 0){

wy += ONEOBJY\*timesY;

}

if(wy < limitY){

if((map->GetRoute()&14) && ((map->ReturnNowY())%MHEIGHT)<ONEOBJY\*timesY && ((map->ReturnNowY())%MHEIGHT)>0)

timesY =((map->ReturnNowY())%MHEIGHT)/ONEOBJY;

map->MoveMap(1,ONEOBJX\*timesX,ONEOBJY\*timesY);

}

ThisMoveTimesX=timesX;

ThisMoveTimesY=timesY;

}

if(move[1]){

wy += ONEOBJY\*timesY;

if(wy > MHEIGHT-height){

wy -= ONEOBJY\*timesY;

}

if(wy > limitY){

if((map->GetRoute()&13) && MHEIGHT-((map->ReturnNowY())%MHEIGHT)<ONEOBJY\*timesY && MHEIGHT-((map->ReturnNowY())%MHEIGHT)>0)

timesY =(MHEIGHT-((map->ReturnNowY())%MHEIGHT))/ONEOBJY;

map->MoveMap(2,ONEOBJX\*timesX,ONEOBJY\*timesY);

}

ThisMoveTimesX=timesX;

ThisMoveTimesY=timesY;

}

if(move[2]){

wx -= ONEOBJX\*timesX;

if(wx < 0){

wx += ONEOBJX\*timesX;

}

if(wx < limitX)

map->MoveMap(4,ONEOBJX\*timesX,ONEOBJY\*timesY);

ThisMoveTimesX=timesX;

ThisMoveTimesY=timesY;

}

if(move[3]){

wx += ONEOBJX\*timesX;

if(wx > MWIDTH-width){

wx -= ONEOBJX\*timesX;

}

if(wx > limitX)

map->MoveMap(8,ONEOBJX\*timesX,ONEOBJY\*timesY);

ThisMoveTimesX=timesX;

ThisMoveTimesY=timesY;

}

if(ContinueMode>0){

move[0] = move[1] = move[2] = move[3] = false;

ContinueMode =0;

}

}

void GameMapEdit::SetWxWy(int setWx,int setWy)

{

wx = setWx;

wy = setWy;

}

void GameMapEdit::SetMove(bool flag,bool IsUp,bool IsDown,bool IsLeft,bool IsRight,int mode)

{

if(IsUp) move[0]=flag;

if(IsDown) move[1]=flag;

if(IsLeft) move[2]=flag;

if(IsRight) move[3]=flag;

if(mode) ContinueMode = mode;

}

void GameMapEdit::FixXY(MapManage \*map)

{

int temp = map->MoveMap(0);

if(temp&1)

wy += ONEOBJY\*ThisMoveTimesY;

if(temp&2)

wy -= ONEOBJY\*ThisMoveTimesY;

if(temp&4)

wx += ONEOBJX\*ThisMoveTimesX;

if(temp&8)

wx -= ONEOBJX\*ThisMoveTimesX;

ThisMoveTimesX=0;

ThisMoveTimesY=0;

}

void GameMapEdit::SetObject(MapManage \*map)

{

map->SetObj(Order,wx,wy);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//GameScore實作 \*/

// \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

GameScore GameScore::Game\_Score;

GameScore::GameScore(){

Switch=false;

score\_p=new CInteger(2);

}

GameScore::~GameScore(){

delete(score\_p);

TRACE("~GameScore()\n");

}

GameScore\* GameScore::Instance()

{

return &Game\_Score;

}

void GameScore::Loading(){

Head[0].LoadBitmapA("Bitmaps/score/chiphead.bmp");

Head[1].LoadBitmapA("Bitmaps/score/dalehead.bmp");

F.LoadBitmapA("Bitmaps/score/F.bmp",PURPLE);

S.LoadBitmapA("Bitmaps/score/S.bmp",PURPLE);

L\_C.LoadBitmapA("Bitmaps/score/L\_C.bmp",PURPLE);

L\_D.LoadBitmapA("Bitmaps/score/L\_D.bmp",PURPLE);

score\_p->LoadBitmap();

}

void GameScore::ShowScroe(){

if(Switch==true)Switch=false;

else Switch=true;

}

void GameScore::onShow(){

if(Switch==false)return;

if(CGameStateRun::player[0]->ReturnIsDale()==0)

{

Head[0].SetTopLeft(MWIDTH/2-100-Head[0].Width(),100);

Head[0].ShowBitmap();

F.SetTopLeft(MWIDTH/2-100-Head[0].Width(),240);

S.SetTopLeft(MWIDTH/2-100-Head[0].Width(),300);

L\_C.SetTopLeft(MWIDTH/2-100-Head[0].Width(),360);

F.ShowBitmap();

S.ShowBitmap();

L\_C.ShowBitmap();

score\_p->SetInteger(CGameStateRun::player[0]->Score\_Flower);

score\_p->SetTopLeft(MWIDTH/2-50-Head[0].Width(),250);

score\_p->ShowBitmap();

score\_p->SetInteger(CGameStateRun::player[0]->Score\_Star);

score\_p->SetTopLeft(MWIDTH/2-50-Head[0].Width(),310);

score\_p->ShowBitmap();

score\_p->SetInteger(CGameStateRun::player[0]->ReturnLife());

score\_p->SetTopLeft(MWIDTH/2-50-Head[0].Width(),370);

score\_p->ShowBitmap();

}

if(CGameStateRun::multiplayer==2||CGameStateRun::player[0]->ReturnIsDale())

{

ChipDale\* p=CGameStateRun::player[0];

if(CGameStateRun::multiplayer==2)p=CGameStateRun::player[1];

Head[1].SetTopLeft(MWIDTH/2+100,100);

Head[1].ShowBitmap();

F.SetTopLeft(MWIDTH/2+100,240);

S.SetTopLeft(MWIDTH/2+100,300);

L\_D.SetTopLeft(MWIDTH/2+100,360);

F.ShowBitmap();

S.ShowBitmap();

L\_D.ShowBitmap();

score\_p->SetInteger(p->Score\_Flower);

score\_p->SetTopLeft(MWIDTH/2+150,250);

score\_p->ShowBitmap();

score\_p->SetInteger(p->Score\_Star);

score\_p->SetTopLeft(MWIDTH/2+150,310);

score\_p->ShowBitmap();

score\_p->SetInteger(p->ReturnLife());

score\_p->SetTopLeft(MWIDTH/2+150,370);

score\_p->ShowBitmap();

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//CFrame實作 \*/

// \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int CFrame::Jump\_Fix = 0;

CFrame::CFrame()

{

wx=wy=0;

for(int i=0;i<6;i++)

flag[i]=false;

LRflag=1;

}

void CFrame::SetWxWy(int setX,int setY,bool shiftmode)

{

if(shiftmode){

if((wx+width+setX)<=MWIDTH&&(wx+setX)>=0) wx += setX;

if((wy+setY)<MHEIGHT) wy += setY;

}

else{

if(setX>=0&&(width+setX)<MWIDTH) wx = setX;

if(setY<MHEIGHT) wy = setY;

}

}

void CFrame::SetWidthHeight(int tempW,int tempH)

{

width = tempW;

height = tempH;

}

void CFrame::FixXY(MapManage \*map)

{

int temp = map->MoveMap(0);

if(temp&1)

wy += Jump\_Fix;

if(temp&2)

wy -= Jump\_Fix;

if(temp&4)

wx += MapManage::LRMargin;

if(temp&8)

wx -= MapManage::LRMargin;

}

void CFrame::ReFixXY(MapManage \*map)

{

int temp = map->MoveMap(0);

if(temp&1)

wy -= Jump\_Fix;

if(temp&2)

wy += SPEED;

if(temp&4)

wx -= MapManage::LRMargin;

if(temp&8)

wx += MapManage::LRMargin;

}

void CFrame::FixMapMove(int fixX,int fixY)

{

wx -= fixX;

wy -= fixY;

}

int CFrame::IfCollision(int twx,int twy,int twidth,int theight)

{

//左上 1

int cx,cy;

for(int i=1;i<=4;i++){

switch(i){

case 1:

cx = twx;

cy = twy;

break;

case 2:

cx = twx+twidth;

cy = twy;

break;

case 3:

cx = twx+twidth;

cy = twy+theight;

break;

case 4:

cx = twx;

cy = twy+theight;

break;

}

if(wx<=cx && wx+width >= cx && wy<=cy && wy+height >= cy) return i;

}

if(twx<=wx && twx+twidth >= wx && twy<=wy && twy+theight >= wy) return 5;

return 0;

}

int CFrame::IfCollision(int Direct,int passSpeed){

MapManage \*map = ChipDale::Maps;

passSpeed = abs(passSpeed);

switch(Direct){

case 1: return map->IfCollision(wx,wy-passSpeed,width,passSpeed,true);

case 4:

if(wx-passSpeed < 0) return 1;

return map->IfCollision(wx-passSpeed,wy,passSpeed,height,true);

case 8:

if(wx+width+passSpeed > MWIDTH) return 1;

return map->IfCollision(wx+width,wy,passSpeed,height,true);

}

return 0;

}

**mapmanage.cpp**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//MapManage實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#define SET\_INITIAL(Level,LOCATE,INITIAL\_Y) limit[(Level-1)\*MAX\_YN+LOCATE/10][(Level-1)\*MAX\_YN+LOCATE%MAX\_XN-1] += ((INITIAL\_Y\*DesSize)<<7);

#define DELIVER(Level,LOCATE,DES) limit[(Level-1)\*MAX\_YN+LOCATE/10][(Level-1)\*MAX\_YN+LOCATE%MAX\_XN-1] += (DES<<7);

int MapManage::isMoveMap = 0;

int MapManage::Teleport = 0;

int MapManage::LRMargin = 0;

MapManage::MapManage(int WhichLevel)

{

LevelLoading(WhichLevel);

}

void MapManage::LevelLoading(int WhichLevel,bool NotResetAll){

char temp[100];

int count=0,OnePercent = (MAX\_YN\*MAX\_XN\*2 + OrderSize + MonsterOrderSize + 1)/100;

FILE \*file;

static int limit[MAX\_YN\*2][MAX\_XN] = {//第一關

{15, 0,15+64, 0, 0, 0, 0, 0, 0, 0},

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{ 8, 8, 8, 8, 8, 8, 0, 0, 0, 0},

{ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

{ 9, 8, 8, 8, 8, 8, 8+16, 0, 0, 0},

{ 1, 0, 0, 0, 0, 0 ,0 ,0, 0, 0},

{ 5, 4, 4, 4, 4, 5, 0, 0, 0, 0},

{ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},

{ 0, 0, 0, 0, 0, 1, 0, 0, 0, 0},

{ 8, 8, 8, 8, 8, 9, 0, 0, 0, 0},

//第二關...(待開發)

},IfSetLimit=0;

//initial\_Y 是要輸入 地板的左上角y位置 不是人物的左上角y位置

if(!IfSetLimit){//防止跑兩次以上 因為 limit 是 static!!!! (有修正過)

//第一關

SET\_INITIAL(1,91,135);

SET\_INITIAL(1,21,326);

SET\_INITIAL(1,1,395);

SET\_INITIAL(1,3,395);

DELIVER(1,47,21);

DELIVER(1,26,1);

DELIVER(1,1,3);

//第二關...(待開發)

//關卡設定結束

IfSetLimit=1;

}

if(!NotResetAll){

x=0;y=0;

ToolCDC::ShowProgressBar(WhichLevel,count/OnePercent);

for(int i=0;i<MAX\_XN;i++){

for(int j=0;j<MAX\_YN;j++){

Route[j][i] = limit[(WhichLevel-1)\*MAX\_YN+j][i];

if(Route[j][i]>0){

sprintf(temp,"Bitmaps/0x%d/%d.bmp",WhichLevel,j\*10+i+1);

MapSP[j][i].LoadBitmap(temp);

if(Route[j][i]&16){

x = i\*MWIDTH;

y = j\*MHEIGHT;

recordx=x;

recordy=y;

Teleport = Route[j][i];

}

}

count++;

ToolCDC::ShowProgressBar(WhichLevel,count/OnePercent);

}

}

}

/\* NotResetAll 有跟無差在底下SET \*/

sprintf(temp,"ObjInfo/0x%d.ChipDale",WhichLevel);

file = fopen(temp,"rb");

for(int j=0;j<MAX\_OY;j++)for(int i=0;i<MAX\_OX;i++)MapObjXY[j][i]=0;

if(file){

for(int j=0;j<MAX\_YN;j++){

for(int i=0;i<MAX\_XN;i++){

if(Route[j][i]){

for(int k=0;k<MAX\_OY/MAX\_YN;k++){

fread(&MapObjXY[j\*MAX\_OY/MAX\_YN+k][i\*MAX\_OX/MAX\_XN],MAX\_OX/MAX\_XN,sizeof(int),file);

}

}

if(!NotResetAll){

count++;

ToolCDC::ShowProgressBar(WhichLevel,count/OnePercent);

}

}

}

fclose(file);

}

else{

SaveObj(WhichLevel);

if(!NotResetAll){

count += MAX\_YN\*MAX\_XN;

ToolCDC::ShowProgressBar(WhichLevel,count/OnePercent);

}

}

if(!NotResetAll){

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

sprintf(temp,"Bitmaps/object/%d.bmp",i);

frame\_obstacle[i].LoadBitmapA(temp,PURPLE);

count++;

ToolCDC::ShowProgressBar(WhichLevel,count/OnePercent);

}

for(int i=-1;i>=-MonsterOrderSize;i--){

sprintf(temp,"Bitmaps/monster/%d.bmp",-i);

frame\_obstacle[-i+OrderSize-1].LoadBitmapA(temp,PURPLE);

count++;

ToolCDC::ShowProgressBar(WhichLevel,count/OnePercent);

}

frame\_obstacle[OrderSize+MonsterOrderSize + 0].LoadBitmapA("Bitmaps/object/StoneUnVisible.bmp",PURPLE);

count++;

ToolCDC::ShowProgressBar(WhichLevel,100);

}

}

void MapManage::OnShow(){

int gx = x/MWIDTH,gy = y/MHEIGHT; //地圖左上角轉換成螢幕格座標

int ox = x/ONEOBJX,oy = y/ONEOBJY;

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

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

if(gx+i>=0 && gx+i<MAX\_XN && gy+j>=0 && gy+j<MAX\_YN && Route[gy+j][gx+i]>0){

MapSP[gy+j][gx+i].SetTopLeft((gx+i)\*MWIDTH-x,(gy+j)\*MHEIGHT-y);

MapSP[gy+j][gx+i].ShowBitmap();

}

}

}

}

void MapManage::OnShowObject(){

int gx = x/MWIDTH,gy = y/MHEIGHT; //地圖左上角轉換成螢幕格座標

int ox = x/ONEOBJX,oy = y/ONEOBJY;

for(int j=-(WinShowBuffer/ONEOBJY);j<MHEIGHT/ONEOBJY;j++){

if(oy+j<0)j=-oy;

for(int i=-(WinShowBuffer/ONEOBJX);i<MWIDTH/ONEOBJX;i++){

if(ox+i<0)i=-ox;

if(MapObjXY[oy+j][ox+i]>=OrderSize||MapObjXY[oy+j][ox+i]<=2)

continue;

frame\_obstacle[MapObjXY[oy+j][ox+i]].SetTopLeft( (ox+i)\*ONEOBJX-x , (oy+j)\*ONEOBJY-y );

frame\_obstacle[MapObjXY[oy+j][ox+i]].ShowBitmap();

}

}

}

void MapManage::ShowObstacle(int i,int j){

int ox = (x-WinShowBuffer)/ONEOBJX,oy = (y-WinShowBuffer)/ONEOBJY,temp=0;

if(ox+i>=0 && ox+i<MAX\_OX && oy+j>=0 && oy+j<MAX\_OY && MapObjXY[oy+j][ox+i]!=0){

if(MapObjXY[oy+j][ox+i]>=OrderSize || MapObjXY[oy+j][ox+i] < -MonsterOrderSize)

return;

temp = (MapObjXY[oy+j][ox+i]<0)\*(-MapObjXY[oy+j][ox+i]+OrderSize-1) + (MapObjXY[oy+j][ox+i]>=0)\*(MapObjXY[oy+j][ox+i]);

frame\_obstacle[ temp ].SetTopLeft((ox+i)\*ONEOBJX-x,(oy+j)\*ONEOBJY-y);

frame\_obstacle[ temp ].ShowBitmap();

}

}

int MapManage::MoveMap(int move,int MoveOneX,int MoveOneY){

if((move&4) || (move&8)){

LRMargin = MoveOneX;

}

if((move&1) || (move&2)){

CFrame::Jump\_Fix = MoveOneY;

}

if( (move&1) && !(isMoveMap&1)){

if(x%MWIDTH == 0 && ( (Route[y/MHEIGHT][x/MWIDTH]&1) || ((Route[y/MHEIGHT][x/MWIDTH]&2)&&(isMoveMap&16)))){

y -=MoveOneY;

isMoveMap |= 1;

if(y<0 || Route[y/MHEIGHT][x/MWIDTH]==0){

y +=MoveOneY;

isMoveMap &= 30;

}

}

else if(x%MWIDTH <= SPEED && (Route[y/MHEIGHT][x/MWIDTH]&1)){

y -=MoveOneY;

x -= x%MWIDTH;

isMoveMap |= (1+4);

LRMargin = x%MWIDTH;

if(y<0 || Route[y/MHEIGHT][x/MWIDTH]==0){

y +=MoveOneY;

isMoveMap &= 30;

}

}

else if((x+SPEED)%MWIDTH <= SPEED && (Route[y/MHEIGHT][(x+SPEED)/MWIDTH]&1)){

y -=MoveOneY;

x += (x+SPEED)/MWIDTH\*MWIDTH - x;

isMoveMap |= (1+8);

LRMargin = (x+SPEED)/MWIDTH\*MWIDTH - x;

if(y<0 || Route[y/MHEIGHT][x/MWIDTH]==0){

y +=MoveOneY;

isMoveMap &= 30;

}

}

}

if( (move&2) && !(isMoveMap&2)){

if(x%MWIDTH == 0 && ( (Route[y/MHEIGHT][x/MWIDTH]&2) || ((Route[y/MHEIGHT][x/MWIDTH]&1)&&(isMoveMap&16)))){

y +=MoveOneY;

isMoveMap |= 2;

if(y+MHEIGHT-1>MHEIGHT\*MAX\_YN || Route[(y+MHEIGHT-1)/MHEIGHT][x/MWIDTH]==0){

y -=MoveOneY;

isMoveMap &= 29;

}

}

}

if( (move&4) && !(isMoveMap&4)){

if(y%MHEIGHT == 0 && ( (Route[y/MHEIGHT][x/MWIDTH]&4) || ((Route[y/MHEIGHT][x/MWIDTH]&8)&&(isMoveMap&16)))){

x -=MoveOneX;

isMoveMap |= 4;

if(x<0 || Route[y/MHEIGHT][x/MWIDTH]==0){

x +=MoveOneX;

isMoveMap &= 27;

}

}

}

if( (move&8) && !(isMoveMap&8)){

if(y%MHEIGHT == 0 && ( (Route[y/MHEIGHT][x/MWIDTH]&8) || ((Route[y/MHEIGHT][x/MWIDTH]&4)&&(isMoveMap&16)))){

x +=MoveOneX;

isMoveMap |= 8;

if(x+MWIDTH-1>MWIDTH\*MAX\_XN || Route[y/MHEIGHT][(x+MWIDTH-1)/MWIDTH]==0){

x -=MoveOneX;

isMoveMap &= 23;

}

}

}

return isMoveMap;

}

void MapManage::SetMapXY(int setX,int setY){

x = setX;

y = setY;

}

void MapManage::Set\_toRecord(){

Object \*\*AllThrow = ChipDale::CanThrow;

LevelLoading(CGameStateRun::NowLevel,true);

SetMapXY(recordx,recordy);

SetObstacle();

SetMonster(ChipDale::AllMonster);

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

if(AllThrow[i]!=NULL)

AllThrow[i]->SetNowAct(0);

}

}

void MapManage::SetObj(int value,int wx,int wy){

MapObjXY[(wy+y)/ONEOBJY][(wx+x)/ONEOBJX] = value;

}

void MapManage::SaveObj(int WhichLevel){

char temp[100];

FILE \*file;

sprintf(temp,"ObjInfo/0x%d.ChipDale",WhichLevel);

file = fopen(temp,"wb");

for(int j=0;j<MAX\_YN;j++){

for(int i=0;i<MAX\_XN;i++){

if(Route[j][i]){

for(int k=0;k<MAX\_OY/MAX\_YN;k++){

fwrite(&MapObjXY[j\*MAX\_OY/MAX\_YN+k][i\*MAX\_OX/MAX\_XN],MAX\_OX/MAX\_XN,sizeof(int),file);

}

}

}

}

fclose(file);

}

void MapManage::ClearisMoveMap(bool WatchMode)

{

if(WatchMode)isMoveMap = 16;

else isMoveMap=0;

}

int MapManage::IfCollision(int wx,int wy,int PicWidth,int PicHeight,bool ignore\_obj\_2,bool ignore\_eat,bool findNext,bool Reverse)

{

int start\_ox,start\_oy,end\_ox,end\_oy,tj;

if(wy<0){

PicHeight += wy;

wy=0;

}

start\_ox = (x+wx)/ONEOBJX;

start\_oy = (y+wy)/ONEOBJY;

if(start\_ox<0)start\_ox = 0;

if(start\_oy<0)start\_oy = 0;

end\_ox = (x+wx+PicWidth)/ONEOBJX;

end\_oy = (y+wy+PicHeight)/ONEOBJY;

if(end\_ox>=MAX\_OX)end\_ox = MAX\_OX-1;

if(end\_oy>=MAX\_OY)end\_oy = MAX\_OY-1;

for(int j=end\_oy-1;j>=start\_oy;j--){

if(Reverse) tj = (start\_oy+end\_oy-1)-j;

else tj = j;

for(int i=start\_ox;i<end\_ox;i++){

if(!ignore\_eat && ( Obstacle[tj][i]/E8==4 || Obstacle[tj][i]/E8>=7 && Obstacle[tj][i]/E8<=13 || Obstacle[tj][i]/E8==17) )//花

continue;

if(ignore\_obj\_2 && Obstacle[tj][i]==2)

continue;

if(Obstacle[tj][i]>0){

if(Obstacle[tj][i]!=1&&Obstacle[tj][i]!=2)

if(findNext==true){

j=(Obstacle[tj][i]%E8)/E4-1;

findNext=false;

break;

}

return Obstacle[tj][i];

}

}

}

return 0;

}

void MapManage::SetObstacle()

{

for(int i=0;i<MAX\_OX;i++)

for(int j=0;j<MAX\_OY;j++)

Obstacle[j][i]=0;

int ox\_end,oy\_end,ox\_start,oy\_start;

for(int i=0;i<MAX\_OX;i++)

{

for(int j=0;j<MAX\_OY;j++)

{

if(MapObjXY[j][i] < OrderSize && MapObjXY[j][i] > 0){

ox\_start=i;

oy\_start=j;

switch(MapObjXY[j][i]){

case 1:

case 3:case 4:case 5:case 6:case 7:case 8:

case 12:case 13:case 14:case 15:case 16:case 17:case 18:case BALL:

ox\_end = ox\_start + frame\_obstacle[MapObjXY[j][i]].Width()/ONEOBJX;

oy\_end = oy\_start + frame\_obstacle[MapObjXY[j][i]].Height()/ONEOBJY;

break;

case 9:case 10:case 11:

ox\_end = ox\_start + frame\_obstacle[MapObjXY[j][i]].Width()/ONEOBJX;

oy\_end = oy\_start + frame\_obstacle[MapObjXY[j][i]].Height()/ONEOBJY;

oy\_start += 5;

ox\_start += 2;

ox\_end -= 2;

break;

case 2:

ox\_end = ox\_start + frame\_obstacle[MapObjXY[j][i]].Width()/ONEOBJX;

oy\_end = oy\_start + 1;

break;

default:

ox\_end = ox\_start;oy\_end = oy\_start;

break;

}

for(int l=oy\_start;l<oy\_end;l++){

for(int k=ox\_start;k<ox\_end;k++){

if(MapObjXY[j][i]==1||MapObjXY[j][i]==2)

Obstacle[l][k] = MapObjXY[j][i];

else

Obstacle[l][k] = MapObjXY[j][i]\*E8+(j+1)\*E4+(i+1);

}

}

switch(MapObjXY[j][i]){

case 9:case 10:case 11:

MapObjXY[j][i]=18;

break;

case 14:case 15:case 16:

MapObjXY[j][i]=3;

break;

case 13:case 17:

MapObjXY[j][i]=0;

break;

}

}

}

}

}

void MapManage::SetMonster(Monster \*monster[])

{

int lengthO,k=0;

for(int i=k;i<ONE\_LEVEL\_MONSTER\_NUM;i++){

if(monster[i]!=NULL){

delete(monster[i]);

}

}

for(int j=0;j<MAX\_OY;j++){

for(int i=0;i<MAX\_OX;i++){

switch(MapObjXY[j][i]){

case -1:

monster[k] = new MachineDog(this,i,j);

k++;break;

case -2:

monster[k] = new Cactus(this,i,j);

k++;break;

case -3:

for(lengthO=i+1;lengthO<i+50;lengthO++){

if(MapObjXY[j][lengthO] == -3){

MapObjXY[j][lengthO] = 0;

break;

}

}

monster[k] = new Wire(this,i,j,lengthO-i);

k++;break;

case -4:

monster[k] = new Mouse(this,i,j);

k++;break;

case -5:

monster[k] = new Wasp(this,i,j);

k++;break;

case -6:

monster[k] = new Centipede(this,i,j);

k++;break;

}

}

}

if(k>=ONE\_LEVEL\_MONSTER\_NUM){

TRACE("SetMonster Overflow ( %d )!!!\n",k);

Sleep(10000);

}

else{

for(int i=k;i<ONE\_LEVEL\_MONSTER\_NUM;i++)

monster[i] = NULL;

}

}

void MapManage::ClearObstacle(int Value)

{

int ox\_start,oy\_start;

int ox\_end,oy\_end;

int ObjetID;

ox\_start = Value%E4-1; Value /= E4;

oy\_start = Value%E4-1; Value /= E4;

ObjetID = Value;

if(Value!=0){

ox\_end = ox\_start + frame\_obstacle[ObjetID].Width()/ONEOBJX;

oy\_end = oy\_start + frame\_obstacle[ObjetID].Height()/ONEOBJY;

for(int k=ox\_start;k<ox\_end;k++)

for(int l=oy\_start;l<oy\_end;l++)

Obstacle[l][k] = 0;

MapObjXY[oy\_start][ox\_start]=0;

if(Value==5)

MapObjXY[oy\_start+1][ox\_start]=0;

}

}

int MapManage::FillObstacle(int Value,int setWx,int setWy,bool Visible)

{

int minusX=0,minusY=0;

int ox\_start,oy\_start;

int ox\_end,oy\_end;

int ObjetID;

ox\_start = (setWx + x)/ONEOBJX;

oy\_start = (setWy + y)/ONEOBJY;

ObjetID = Value;

if(Value!=0){

ox\_end = ox\_start + frame\_obstacle[ObjetID].Width()/ONEOBJX;

oy\_end = oy\_start + frame\_obstacle[ObjetID].Height()/ONEOBJY;

switch(ObjetID){

case 4:case 7:case 8:

minusX = 1;

minusY = 1;

break;

}

for(int l=oy\_start+minusY;l<oy\_end-minusY;l++)

for(int k=ox\_start+minusX;k<ox\_end-minusX;k++)

Obstacle[l][k] = Value\*E8+(oy\_start+1)\*E4+(ox\_start+1);

if(Visible)

MapObjXY[oy\_start][ox\_start] = Value;

}

return Value\*E8+(oy\_start+1)\*E4+(ox\_start+1);

}

**instruction.cpp**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//AI\_Instructions實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

AI\_Instructions::AI\_Instructions()

{

UpSpeed=0;

IsJump = false;

LR\_Space = 0;

NoIgnore\_2 = true;

NoIgnore\_eat = false;

}

bool AI\_Instructions::MoveUp(MapManage\* map){

if(NoCollision || !map->IfCollision(wx,wy-MoveSPEED,width,height,true)){

wy -= MoveSPEED;

return true;

}

return false;

}

bool AI\_Instructions::MoveDown(MapManage\* map){

if(NoCollision || !map->IfCollision(wx,wy+MoveSPEED,width,height,true)){

wy += MoveSPEED;

return true;

}

return false;

}

bool AI\_Instructions::MoveLeft(MapManage\* map){

if(NoCollision || !map->IfCollision(wx-MoveSPEED,wy,MoveSPEED,height,true)){

wx -= MoveSPEED;

return true;

}

return false;

}

bool AI\_Instructions::MoveRight(MapManage\* map){

if(NoCollision || !map->IfCollision(wx+width,wy,MoveSPEED,height,true)){

wx += MoveSPEED;

return true;

}

return false;

}

void AI\_Instructions::Jump()

{

UpSpeed = JumpSPEED;

}

int AI\_Instructions::FallingDown(MapManage\* map)

{

int i=0;

int fixY=0,fixX=0;

int IsLand=false;

//修正落下時左右兩邊應該空多少不判斷

wx += LR\_Space;

width -= 2\*LR\_Space;

//最後面記得要回復原本設定

//落下過程的FLAG初始化

if(IsJump || !map->IfCollision(wx+fixX,wy+height+fixY,width,1,false,NoIgnore\_eat/\*ONEOBJY\*/)){

UpSpeed -= GRAVITY;

IsJump = true;

}

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

//以下部分為操作速度的實作過程

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

if(UpSpeed<0 && map->IfCollision(wx+fixX,wy+height+fixY,width,abs(UpSpeed),false,NoIgnore\_eat)){

//落地動作

while(!map->IfCollision(wx+fixX,wy+height+i+fixY,width,1,false,NoIgnore\_eat))i++;/\* 碰撞的修正 \*/

wy += i;

UpSpeed = 0;

IsJump = false;

IsLand=true;

}

else if(UpSpeed>0 && map->IfCollision(wx+fixX,wy+fixY-abs(UpSpeed),width,abs(UpSpeed),NoIgnore\_2)){

/\*向上跳過程遇到障礙物\*/

i=0;while(!map->IfCollision(wx+fixX,wy-i+fixY-1,width,1,NoIgnore\_2))i++;// 碰撞的修正

wy -= i;

UpSpeed = 0;

}

else{

//跳躍過程

wy -= UpSpeed;

}

if(map->IfCollision(wx+fixX,wy+height+fixY,width,1,false,NoIgnore\_eat/\*ONEOBJY\*/)){

IsLand=true;

IsJump = false;

}

//修正落下時左右兩邊應該空多少不判斷 回復原本設定

wx -= LR\_Space;

width += 2\*LR\_Space;

return IsLand;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//BasicInstructions實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bool BasicInstructions::canMoveMapX = true;

bool BasicInstructions::canMoveMapY = true;

BasicInstructions::BasicInstructions()

{

UpSpeed=0;

time\_jump=0;

IsJump = false;

IsRun = false;

IsLessCollision = false;

ani\_jump\_count=TIMEFOR\_ANI\_JUMP+1;

Reduce\_UP\_VELOCITY = 0;

}

bool BasicInstructions::MoveLeft(MapManage\* map){

int i,limitX = (MWIDTH + width)/2,temp,testTemp;

int UnVisibleStone = OrderSize+MonsterOrderSize +0;

TriggerObj(3);

if(ToolCDC::ReturnStage()!=0)return false; //防止已經存圖又移動

//抓人

if(Instance()->Partner!=NULL&&CollisionChipDale(4,2,0)&&!Instance()->ReturnIsTaken()&&Instance()->A\_FLAG

&&!Instance()->Partner->ReturnIsTaken()&&!Instance()->Partner->ReturnInvincible()){

Object\*\* CanThrow=ChipDale::CanThrow;

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

if(CanThrow[i]==NULL){

//if(Partner->Alive)break; //註解掉此行會有殘影效果

CanThrow[i]=new ChipDale\_Taken(Instance());

Instance()->NowTakeObj=CanThrow[i];

Instance()->NowTaken=ChipDale\_taken;

Instance()->A\_FLAG = false;

break;

}

}

}

//拿東西

if(A\_FLAG && !SetNowTaken(-1)){

temp = map->IfCollision(wx-SPEED-ONEOBJX,wy,ONEOBJX,height,true,false);

testTemp = map->IfCollision(wx-SPEED-ONEOBJX,wy,ONEOBJX,height,true,false,true);

if(testTemp/E8==5 || temp/E8==5 && testTemp/E8!=5 && testTemp/E8!=0) temp = testTemp;

if(testTemp/E8==UnVisibleStone || temp/E8==UnVisibleStone && testTemp/E8!=UnVisibleStone && testTemp/E8!=0) temp = testTemp;

if(temp!=1&&temp!=2&&temp!=0&&temp/E8!=4){

map->ClearObstacle(temp);

SetNowTaken(temp);

A\_FLAG = false;

}

}

temp = CollisionChipDale(4,SPEED,0);

//移動

if(!map->IfCollision(wx-SPEED,wy,SPEED,height,true) && !temp){

SetWxWy(-SPEED,0,true);

if(canMoveMapX && wx < limitX)

map->MoveMap(4);

return true;

}

else if(temp) {

//FIRST 腳色完全貼近

i=SPEED-1;while(i>=0 && CollisionChipDale(4,i,0))i--;

SetWxWy(-i,0,true);

//移動剩餘額度 SPEED-i

//貼近之後推擠對方

if(!getPartner()->ReturnHideComplete()&&!IfCollision(4,1))

CollisionChipDale(4,1);

//帶動螢幕

if(canMoveMapX && wx < limitX)

map->MoveMap(4,1);

return true;

}

else {

//撞到地圖障礙物修正

i=0;while(!map->IfCollision(wx-i,wy,width,height,true))i++;

wx -=(i!=0)\*(i-1)+(i==0)\*(-SPEED);

}

return false;

}

bool BasicInstructions::MoveRight(MapManage\* map){

int i,limitX = (MWIDTH + width)/2,temp,testTemp;

int UnVisibleStone = OrderSize+MonsterOrderSize +0;

TriggerObj(4);

if(ToolCDC::ReturnStage()!=0)return false; //防止已經存圖又移動

//抓人

if(Instance()->Partner!=NULL&&CollisionChipDale(8,2,0)&&!Instance()->ReturnIsTaken()&&Instance()->A\_FLAG

&&!Instance()->Partner->ReturnIsTaken()&&!Instance()->Partner->ReturnInvincible()){

Object\*\* CanThrow=ChipDale::CanThrow;

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

if(CanThrow[i]==NULL){

CanThrow[i]=new ChipDale\_Taken(Instance());

Instance()->NowTakeObj=CanThrow[i];

Instance()->NowTaken=ChipDale\_taken;

Instance()->A\_FLAG = false;

break;

}

}

}

//拿東西

if(A\_FLAG && !SetNowTaken(-1)){

temp = map->IfCollision(wx+width+SPEED,wy,ONEOBJX,height,true,false);

testTemp = map->IfCollision(wx+width+SPEED,wy,ONEOBJX,height,true,false,true);

if(testTemp/E8==5 || temp/E8==5 && testTemp/E8!=5 && testTemp/E8!=0) temp = testTemp;

if(testTemp/E8== UnVisibleStone|| temp/E8==UnVisibleStone && testTemp/E8!=UnVisibleStone && testTemp/E8!=0) temp = testTemp;

if(temp!=1&&temp!=2&&temp!=0&&temp/E8!=4){//箱子...

map->ClearObstacle(temp);

SetNowTaken(temp);

A\_FLAG = false;

}

}

//移動

temp = CollisionChipDale(8,SPEED,0);

if(!map->IfCollision(wx+width,wy,SPEED,height,true) && !temp){

SetWxWy(SPEED,0,true);

if(canMoveMapX && wx > limitX)

map->MoveMap(8);

return true;

}

else if(temp){

//FIRST 腳色完全貼近

i=SPEED-1;while(i>=0 && CollisionChipDale(8,i,0))i--;

SetWxWy(i,0,true);

//移動剩餘額度 SPEED-i

//貼近之後推擠對方

if(!getPartner()->ReturnHideComplete()&&!IfCollision(8,1))

CollisionChipDale(8,1);

//帶動螢幕

if(canMoveMapX && wx > limitX)

map->MoveMap(8,1);

return true;

}

else{

i=0;while(!map->IfCollision(wx+i,wy,width,height,true))i++;

wx +=(i!=0)\*(i-1)+(i==0)\*(-SPEED);

}

return false;

}

bool BasicInstructions::Jump()

{

if(ToolCDC::ReturnStage()!=0)return false; //防止已經存圖又移動

if(time\_jump < TIMEFORJUMP && !IsLessCollision){

if(B\_FLAG){

UpSpeed = UP\_VELOCITY - (time\_jump-1)\*GRAVITY -Reduce\_UP\_VELOCITY;

time\_jump++;

}

else{

time\_jump = TIMEFORJUMP;

UpSpeed = UP\_VELOCITY - (TIMEFORJUMP-1)\*GRAVITY -Reduce\_UP\_VELOCITY;

}

return true;

}

return false;

}

int BasicInstructions::FallingDown(MapManage\* map)

{

int i=0,limitY = (MHEIGHT )/2- height;

int If\_Reset\_State=false,tempFixY=0,tempFixX=0;

if(ToolCDC::ReturnStage()!=0)return If\_Reset\_State; //防止已經存圖又移動

//暫時性修正　讓遊戲在未修正前也能判斷正常

if(map->MoveMap(0)&1)tempFixY = Jump\_Fix;

if(map->MoveMap(0)&4)tempFixX = SPEED;

if(map->MoveMap(0)&8)tempFixX = -SPEED;

//落下過程的FLAG初始化 // 預設是不進行跳躍

if(!IsLessCollision && !map->IfCollision(wx+tempFixX,wy+height+tempFixY,width,1/\*ONEOBJY\*/)

&&!CollisionChipDale(2,1,0)){

UpSpeed -= GRAVITY;

IsJump = true;

If\_Reset\_State = true;

}

//如果下方為藍色地板以外的物件則不能下跳

if(IsLessCollision && map->IfCollision(wx+tempFixX,wy+height+tempFixY,width,1/\*ONEOBJY\*/,true )){

IsLessCollision= false;

IsJump=false;

B\_FLAG = false;

UpSpeed=0;

}

//等到MoveRL修好了再啟用

if(CollisionChipDale(2,0,0)){

i=0;

while(CollisionChipDale(2,1,0)){

wy--;

i++;/\* 碰撞的修正 \*/

}

}

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

//以下部分為操作速度的實作過程

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

//吃花

if(UpSpeed<=0)

TriggerObj(1);

else

TriggerObj(2);

if(IsLessCollision){

//下跳穿越地板

UpSpeed -= GRAVITY;

wy -= UpSpeed;

if(!map->IfCollision(wx+tempFixX,wy+height+tempFixY,width,1))

IsLessCollision= false;

IsJump = true;

If\_Reset\_State = true;

}

else if(UpSpeed<0 && map->IfCollision(wx+tempFixX,wy+height+tempFixY,width,abs(UpSpeed))){

//落地動作

while(!map->IfCollision(wx+tempFixX,wy+height+i+tempFixY,width,1))i++;/\* 碰撞的修正 \*/

wy += i;

UpSpeed = 0;

time\_jump = 0;

If\_Reset\_State = true;

IsJump = false;

B\_FLAG = false;

ani\_jump\_count=0;

}

else if(UpSpeed<0 && CollisionChipDale(2,UpSpeed,0)){

i=0;

while(CollisionChipDale(2,i,0)!=1){

i++;/\* 碰撞的修正 \*/

}

wy += i-1;

UpSpeed = 0;

time\_jump = 0;

If\_Reset\_State = true;

IsJump = false;

B\_FLAG = false;

ani\_jump\_count=0;

}

else if(UpSpeed>0 && map->IfCollision(wx+tempFixX,wy+tempFixY-abs(UpSpeed),width,abs(UpSpeed),true)){

/\*向上跳過程遇到障礙物\*/

i=0;while(!map->IfCollision(wx+tempFixX,wy-i+tempFixY-1,width,1,true))i++;// 碰撞的修正

wy -= i;

UpSpeed = 0;

time\_jump = TIMEFORJUMP;

}

else{//跳躍過程,持續呼叫

//2P碰撞

if(UpSpeed>0)CollisionChipDale(1,UpSpeed,1);

wy -= UpSpeed;

}

/\*跳躍帶動螢幕\*/

if(canMoveMapY && wy < limitY && UpSpeed>0 && !(map->MoveMap(0)&1)){

map->MoveMap(1,SPEED,UpSpeed/SPEED\*SPEED);

Jump\_Fix =UpSpeed/SPEED\*SPEED;

}

return If\_Reset\_State;

}

**ChipDale.cpp**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ChipDale實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Object \*\*ChipDale::CanThrow;

MapManage \*ChipDale::Maps;

Monster \*\*ChipDale::AllMonster;

CAnimation ChipDale::animation[2][2][ACTION\_NUM][2]; // 動作~方向

CAnimation ChipDale::ani\_sweat[2];

CAnimation ChipDale::ani\_dizzy;

CAnimation ChipDale::ani\_god[2];

ChipDale::ChipDale(int isDale){

IsDale = isDale;

Life=3;

Reset(10,10);//必須先有Life才能reset

IsGod=false;

ResetScore();

}

void ChipDale::Reset(int Wx,int Wy,bool LR,bool FullHealth){

if(Life<=0)return;

wx=Wx;

wy=Wy;

LRflag=LR;

if(FullHealth)Health=3;

LOCK=false;

time\_jump = 0;

NowAct=LastAct=0;

NowTaken=0;

freeze=0;

Reduce\_UP\_VELOCITY = 0;

Invincible = IsFaint=0;

Hurt=false;

NowTakeObj=NULL;

ani\_sweat\_count=0;

ani\_Hide\_freeze\_jump=false;

setFlag(false,true,true,true,true,true,true);

Alive=true;

UpSpeed=0;

time\_jump=0;

IsJump = false;

IsRun = false;

IsLessCollision = false;

ani\_jump\_count=TIMEFOR\_ANI\_JUMP+1;

Reduce\_UP\_VELOCITY = 0;

};

void ChipDale::Loading(){

char name[6]="chip/";

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

animation[i][0][0][0].AddBitmap("Bitmaps/action/",name,"stand\_L.bmp",PURPLE);

animation[i][0][0][1].AddBitmap("Bitmaps/action/",name,"stand\_R.bmp",PURPLE);

animation[i][0][1][0].AddBitmap("Bitmaps/action/",name,"run\_1L.bmp",PURPLE);

animation[i][0][1][0].AddBitmap("Bitmaps/action/",name,"run\_2L.bmp",PURPLE);

animation[i][0][1][0].AddBitmap("Bitmaps/action/",name,"run\_3L.bmp",PURPLE);

animation[i][0][1][0].AddBitmap("Bitmaps/action/",name,"run\_4L.bmp",PURPLE);

animation[i][0][1][1].AddBitmap("Bitmaps/action/",name,"run\_1R.bmp",PURPLE);

animation[i][0][1][1].AddBitmap("Bitmaps/action/",name,"run\_2R.bmp",PURPLE);

animation[i][0][1][1].AddBitmap("Bitmaps/action/",name,"run\_3R.bmp",PURPLE);

animation[i][0][1][1].AddBitmap("Bitmaps/action/",name,"run\_4R.bmp",PURPLE);

animation[i][0][2][0].AddBitmap("Bitmaps/action/",name,"jump\_1L.bmp",PURPLE);

animation[i][0][2][1].AddBitmap("Bitmaps/action/",name,"jump\_1R.bmp",PURPLE);

animation[i][0][3][0].AddBitmap("Bitmaps/action/",name,"throw\_L.bmp",PURPLE);

animation[i][0][3][1].AddBitmap("Bitmaps/action/",name,"throw\_R.bmp",PURPLE);

animation[i][0][4][0].AddBitmap("Bitmaps/action/",name,"Squat\_1L.bmp",PURPLE);

animation[i][0][4][1].AddBitmap("Bitmaps/action/",name,"Squat\_1R.bmp",PURPLE);

animation[i][0][5][0].AddBitmap("Bitmaps/action/",name,"Hurt\_L.bmp",PURPLE);

animation[i][0][5][1].AddBitmap("Bitmaps/action/",name,"Hurt\_R.bmp",PURPLE);

animation[i][0][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_1L.bmp",PURPLE);

animation[i][0][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_2L.bmp",PURPLE);

animation[i][0][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_3L.bmp",PURPLE);

animation[i][0][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_4L.bmp",PURPLE);

animation[i][0][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_1R.bmp",PURPLE);

animation[i][0][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_2R.bmp",PURPLE);

animation[i][0][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_3R.bmp",PURPLE);

animation[i][0][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_4R.bmp",PURPLE);

animation[i][1][0][0].AddBitmap("Bitmaps/action/",name,"Take\_L.bmp",PURPLE);

animation[i][1][0][1].AddBitmap("Bitmaps/action/",name,"Take\_R.bmp",PURPLE);

animation[i][1][1][0].AddBitmap("Bitmaps/action/",name,"Take\_Run\_1L.bmp",PURPLE);

animation[i][1][1][0].AddBitmap("Bitmaps/action/",name,"Take\_Run\_2L.bmp",PURPLE);

animation[i][1][1][1].AddBitmap("Bitmaps/action/",name,"Take\_Run\_1R.bmp",PURPLE);

animation[i][1][1][1].AddBitmap("Bitmaps/action/",name,"Take\_Run\_2R.bmp",PURPLE);

animation[i][1][2][0].AddBitmap("Bitmaps/action/",name,"Take\_Jump\_L.bmp",PURPLE);

animation[i][1][2][1].AddBitmap("Bitmaps/action/",name,"Take\_Jump\_R.bmp",PURPLE);

animation[i][1][3][0].AddBitmap("Bitmaps/action/",name,"throw\_L.bmp",PURPLE);

animation[i][1][3][1].AddBitmap("Bitmaps/action/",name,"throw\_R.bmp",PURPLE);

animation[i][1][4][0].SetDelayCount(2);

animation[i][1][4][0].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_1L.bmp",PURPLE);

animation[i][1][4][0].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_2L.bmp",PURPLE);

animation[i][1][4][0].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_3L.bmp",PURPLE);

animation[i][1][4][0].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_4L.bmp",PURPLE);

animation[i][1][4][0].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_5L.bmp",PURPLE);

animation[i][1][4][0].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_6L.bmp",PURPLE);

animation[i][1][4][1].SetDelayCount(2);

animation[i][1][4][1].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_1R.bmp",PURPLE);

animation[i][1][4][1].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_2R.bmp",PURPLE);

animation[i][1][4][1].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_3R.bmp",PURPLE);

animation[i][1][4][1].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_4R.bmp",PURPLE);

animation[i][1][4][1].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_5R.bmp",PURPLE);

animation[i][1][4][1].AddBitmap("Bitmaps/action/",name,"Hide\_crate\_6R.bmp",PURPLE);

animation[i][1][5][0].AddBitmap("Bitmaps/action/",name,"Hurt\_L.bmp",PURPLE);

animation[i][1][5][1].AddBitmap("Bitmaps/action/",name,"Hurt\_R.bmp",PURPLE);

animation[i][1][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_1L.bmp",PURPLE);

animation[i][1][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_2L.bmp",PURPLE);

animation[i][1][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_3L.bmp",PURPLE);

animation[i][1][6][0].AddBitmap("Bitmaps/action/",name,"Faint\_4L.bmp",PURPLE);

animation[i][1][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_1R.bmp",PURPLE);

animation[i][1][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_2R.bmp",PURPLE);

animation[i][1][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_3R.bmp",PURPLE);

animation[i][1][6][1].AddBitmap("Bitmaps/action/",name,"Faint\_4R.bmp",PURPLE);

sprintf(name,"%s","dale/");

}

ani\_sweat[0].AddBitmap("Bitmaps/action/ANI\_ChipDale/sweat\_L.bmp",PURPLE);

ani\_sweat[1].AddBitmap("Bitmaps/action/ANI\_ChipDale/sweat\_R.bmp",PURPLE);

ani\_dizzy.SetDelayCount(7);

ani\_dizzy.AddBitmap("Bitmaps/action/ANI\_ChipDale/Dizzy1.bmp",PURPLE);

ani\_dizzy.AddBitmap("Bitmaps/action/ANI\_ChipDale/Dizzy2.bmp",PURPLE);

ani\_dizzy.AddBitmap("Bitmaps/action/ANI\_ChipDale/Dizzy3.bmp",PURPLE);

ani\_dizzy.AddBitmap("Bitmaps/action/ANI\_ChipDale/Dizzy4.bmp",PURPLE);

ani\_god[0].AddBitmap("Bitmaps/action/ANI\_ChipDale/god\_L.bmp",PURPLE);

ani\_god[1].AddBitmap("Bitmaps/action/ANI\_ChipDale/god\_R.bmp",PURPLE);

}

void ChipDale::OnShow()

{

int Last\_ani\_height,fix\_hieght;

if(!Alive&&!IsGod)return;

//特定動畫重設循環

if(NowAct!=4){

if(animation[IsDale][1][4][0].GetCurrentBitmapNumber()>=4){

animation[IsDale][1][4][0].SetDelayCount(2);/\*SET完記得RESET，不然第一張圖會用到舊的DELAY\_COUNTER\*/

animation[IsDale][1][4][0].Reset();

}

if(animation[IsDale][1][4][1].GetCurrentBitmapNumber()>=4){

animation[IsDale][1][4][1].SetDelayCount(2);

animation[IsDale][1][4][1].Reset();

}

}

if(LastAct==2){

animation[IsDale][0][2][0].Reset();

animation[IsDale][0][2][1].Reset();

}

if(ani\_Hide\_freeze\_jump==0)ani\_Hide\_freeze\_jump=1;

//動畫切9=換 及 重設height wxwy

if(!ReturnHideComplete()

Last\_ani\_height=height;

if(NowAct!=2)

animation[IsDale][TAKEN\_FLAG][NowAct][LRflag].OnMove();

if(Last\_ani\_height!=animation[IsDale][TAKEN\_FLAG][NowAct][LRflag].Height()){

height=animation[IsDale][TAKEN\_FLAG][NowAct][LRflag].Height();

fix\_hieght=Last\_ani\_height-height;

SetWxWy(wx,wy+fix\_hieght);

}

}else {

ani\_Hide\_freeze\_jump=0;

if(TAKEN\_FLAG&&NowAct==4){

if(!animation[IsDale][1][4][LRflag].IsFinalBitmap()){

animation[IsDale][1][4][LRflag].SetDelayCount(5);

animation[IsDale][1][4][LRflag].OnMove();

}else{

animation[IsDale][1][4][LRflag].SetDelayCount(100);

animation[IsDale][1][4][LRflag].OnMoveToNum(4);

}

}

}

//顯示人物

//人物無敵效果

if((Invincible && (Invincible %5==2||Invincible %5==3)) || !Invincible || (NowAct==4&&TAKEN\_FLAG) ){

if(IsGod){

IsJump=false;

if(LRflag)

ani\_god[LRflag].SetBottomLeft(wx+(-ani\_god[LRflag].Width()+15),wy,ani\_god[LRflag].Height());

else

ani\_god[LRflag].SetBottomLeft(wx+width-20,wy,ani\_god[LRflag].Height());

ani\_god[LRflag].OnShow();

}

if(ani\_jump\_count<TIMEFOR\_ANI\_JUMP){

//跳轉蹲動畫

Last\_ani\_height=height;

height=animation[IsDale][0][4][LRflag].Height();

fix\_hieght=Last\_ani\_height-height;

SetWxWy(wx,wy+fix\_hieght);

animation[IsDale][0][4][LRflag].SetBottomLeft(wx,wy,height);

animation[IsDale][0][4][LRflag].OnShow();

}else{

//一般狀態動畫顯示

animation[IsDale][TAKEN\_FLAG][NowAct][LRflag].SetBottomLeft(wx,wy,height);

animation[IsDale][TAKEN\_FLAG][NowAct][LRflag].OnShow();

}

//流汗動畫顯示

if(ani\_sweat\_count>0){

int tempx=ani\_sweat\_x+10-Maps->ReturnNowX();/\*-((Maps->MoveMap(0)&8)>0)\*SPEED\*ani\_sweat\_count\*/;

if(!LRflag)tempx=ani\_sweat\_x+width-20-Maps->ReturnNowX();/\*+((Maps->MoveMap(0)&4)>0)\*SPEED\*ani\_sweat\_count\*/;

ani\_sweat[LRflag].SetBottomLeft(tempx,ani\_sweat\_wy+ani\_sweat\_count\*7/\*汗的位移\*/,ani\_sweat[LRflag].Height());

ani\_sweat[LRflag].OnShow();

}

}

//暈眩顯示

if(NowAct==6){

ani\_dizzy.SetBottomLeft(wx,wy-ani\_dizzy.Height(),ani\_dizzy.Height());

ani\_dizzy.OnShow();

ani\_dizzy.OnMove();

}

}

void ChipDale::Dead(){

Object \*\*CanThrow = ChipDale::CanThrow;

Life--;

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

if(CanThrow[i]!=NULL)continue;

CanThrow[i] = new ChipDale\_Dead(this);

break;

}

ReleaseNowTakeObj();

Alive=false;

Health=0;

}

void ChipDale::GetHurt(){

if(!Alive||IsGod)return;

Hurt=true;

if(UpSpeed>=0)UpSpeed = 40;

else UpSpeed+=40;

SetState();

Health--;

if(Health<=0){

Dead();

}

}

void ChipDale::Faint(){

if(!Alive)return;

IsFaint=1;

SetState();

}

int ChipDale::SetNowTaken(int Value){

if(!Alive)return 0;

if(Value==-1)

return NowTaken!=0;

else{

NowTaken = Value/E8;

if(NowTaken == OrderSize + MonsterOrderSize + 0){

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

if(CanThrow[i]==NULL)continue;

if(CanThrow[i]->ReturnObjValue() == Value){

NowTakeObj = CanThrow[i];

NowTakeObj->RecoverObj(this);

break;

}

if(i+1==CanTakeNum){TRACE("SetNowTaken: No Find In CanThrow\n");Sleep(100000);}

}

}

else{

switch(NowTaken){

case 14:

NowTaken = 3;

Maps->FillObstacle(7,(Value%E4-1)\*ONEOBJX-Maps->ReturnNowX(),(Value%E8/E4-1)\*ONEOBJY-Maps->ReturnNowY());

break;

case 15:

NowTaken = 3;

Maps->FillObstacle(8,(Value%E4-1)\*ONEOBJX-Maps->ReturnNowX(),(Value%E8/E4-1)\*ONEOBJY-Maps->ReturnNowY());

break;

case 16:

NowTaken = 3;

Maps->FillObstacle(4,(Value%E4-1)\*ONEOBJX-Maps->ReturnNowX(),(Value%E8/E4-1)\*ONEOBJY-Maps->ReturnNowY());

break;

}

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

if(CanThrow[i]==NULL){

switch(NowTaken){

case 3:

CanThrow[i] = new Craft(this);

break;

case 5:

CanThrow[i] = new Stone(this);

break;

case 6:

CanThrow[i] = new Apple(this);

break;

case BALL:

CanThrow[i] = new Ball(this);

break;

}

NowTakeObj = CanThrow[i];

break;

}

if(i+1==CanTakeNum){TRACE("SetNowTaken: No Find In CanThrow\n");Sleep(100000);}

}

}

return Value;

}

}

void ChipDale::FixSelf\_Onto\_ObjectTop(){

int i=0;

if(!Alive)return;

while(Maps->IfCollision(wx,wy+i,width,height,true))i--;

wy+=i;

if(i<0)TRACE("fix\n");

}

void ChipDale::SetState(){

int LastHeight=height;

LastAct = NowAct;

//設定LRflag

if(!(LastAct==4)||(Last\_flag[1]&&!DOWN\_FLAG)){

//蹲下不可以變換方向

if(LEFT\_FLAG&&!RIGHT\_FLAG) LRflag=0;

if(!LEFT\_FLAG&&RIGHT\_FLAG) LRflag=1;

}

//state切換

if(LEFT\_FLAG==RIGHT\_FLAG){ NowAct=0;IsRun=false;} //左右同時按 或都不按 Act=站

else{NowAct=1;if(!DOWN\_FLAG||(NowTaken==6))IsRun=true;}

if(IsJump) NowAct=2;

if(DOWN\_FLAG&&NowAct!=2){

if(NowTaken!=APPLE&&NowTaken!=ChipDale\_taken){

NowAct=4;

IsRun=0;

}

}

if(A\_FLAG && (TAKEN\_FLAG)) NowAct=3;

if(Hurt)NowAct=5;

if(IsFaint)NowAct=6;

//重設wx wy

if((NowAct==4/\*||LastAct==4\*/)&&TAKEN\_FLAG){

SetWxWy(wx,wy-34+height);

}

else

SetWxWy(wx,wy-animation[IsDale][TAKEN\_FLAG][NowAct][LRflag].Height()+height);

//重設height

if(NowAct==4&&TAKEN\_FLAG&&animation[IsDale][0][4][LRflag].GetCurrentBitmapNumber()==0)height=34;

else SetWidthHeight(45,animation[IsDale][TAKEN\_FLAG][NowAct][LRflag].Height());

//蹲下放開需要調整2P位置

if(LastAct==4&&Partner!=NULL){

if(CollisionChipDale(0,0,0))

Partner->SetWxWy(0,LastHeight-height,true);

}

}

void ChipDale::setFlag(bool value,bool up,bool down,bool left,bool right,bool A,bool B){

if(LOCK) return; //!Alive必須能控制 ChipDale\_

if(IsFaint&&value) return;

if(up) UP\_FLAG = value;

if(down) DOWN\_FLAG = value;

if(left) LEFT\_FLAG = value;

if(right) RIGHT\_FLAG = value;

if(A) A\_FLAG = value;

if(B){

if(!B\_FLAG && time\_jump==0 && value && !DOWN\_FLAG){

B\_FLAG = true;

IsJump = true;

}

if(!value)

B\_FLAG = false;

if(DOWN\_FLAG && value && !IsJump ){ if((TAKEN\_FLAG&&(!ani\_Hide\_freeze\_jump||NowTaken==APPLE||NowTaken==ChipDale\_taken))||!TAKEN\_FLAG)//HIDE動畫冷卻時間

IsLessCollision = true;

}

}

SetState();

//設定LAST\_FLAG

for(int i=0;i<6;i++){

Last\_flag[i]=flag[i];

}

};

void ChipDale::GodMode(){

//GodMode不支援2P

if(CGameStateRun::multiplayer!=2)

IsGod=!IsGod;

}

void ChipDale::GodMove(){

int limitX = (MWIDTH + width)/2;

int limitY = (MHEIGHT + height)/2;

if(LEFT\_FLAG&&wx-SPEED\*2>=0){

SetWxWy(-SPEED\*2,0,true);

if(wx < limitX)Maps->MoveMap(4,20);

}else if(RIGHT\_FLAG&&wx+width+SPEED\*2<=MWIDTH){

SetWxWy(SPEED\*2,0,true);

if(wx > limitX)Maps->MoveMap(8,20);

}

if(UP\_FLAG&&wy-SPEED\*2>=0){

SetWxWy(0,-SPEED\*2,true);

if(wy < limitY){

Maps->MoveMap(1,10,20);

}

}else if(DOWN\_FLAG&&wy+height+SPEED\*2<=MHEIGHT){

SetWxWy(0,SPEED\*2,true);

if(wy > limitY){

Maps->MoveMap(2,10,20);

}

}

}

void ChipDale::OnMove(){

int temp,tDerict,LRMargin=MapManage::LRMargin;

if(!Alive)return;

if(wy>MHEIGHT+50){Dead();return;}

if(freeze){

HOLD(3,freeze){freeze=0;}; //HOLD值 2~3為佳 (1為沒效果)

}

TriggerObj(0);

//運動

if(!IsGod){

if((!freeze&&IsRun&&!LRflag&&!Hurt||Hurt&&LRflag))MoveLeft(Maps);

if(!freeze&&IsRun&& LRflag&&!Hurt||Hurt&&!LRflag)MoveRight(Maps);

if(IsJump)Jump();

if(FallingDown(Maps)){SetState();}

}else{

GodMove();

}

if(NowTaken==6) Reduce\_UP\_VELOCITY=10;//增加重力加速度

//丟的冷卻時間

if(NowTaken < 0)NowTaken++;

//丟實作

if(NowAct==3 && NowTakeObj!=NULL){

tDerict = 1\*(UP\_FLAG)+4\*(!LRflag)+8\*(LRflag);

if(UP\_FLAG && !IsJump) UpSpeed += 20;//上丟會稍微往上跳

NowTakeObj->Throw(tDerict);

NowTakeObj=NULL;

if(NowTaken==6)Reduce\_UP\_VELOCITY=0;//改成正常重力加速度

NowTaken = -6;

}

//丟得僵直時間

if(NowTaken==-3){setFlag(0,0,0,0,0,1,0);freeze=1;} //決定丟動畫的延遲時間 NowTaken判斷可為-1 ~-5

//跳轉蹲動畫切換變數設定

if(ani\_jump\_count<TIMEFOR\_ANI\_JUMP){

if(ani\_jump\_count < 1){

//clock\_t t=clock();

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

//TRACE("CAudio %d\n",clock()-t);

}

if(ani\_jump\_count==2)SetState();

//TRACE("%d wy=%d height=%d \n",ani\_jump\_count,wy,height);

ani\_jump\_count++;

}

//受傷及無敵的變數設定

const int Hurt\_time=10;

const int Hurt\_Shine\_time=50;

const int invincibility\_time=240;

const int Hurt\_start=invincibility\_time-Hurt\_Shine\_time;

const int Hurt\_end=Hurt\_start+Hurt\_time;

if(Invincible)Invincible++;

if(Hurt&&Invincible<Hurt\_start)Invincible=Hurt\_start;

if(Invincible>invincibility\_time){

SetInvincible(0);

if(Partner==NULL||!Partner->ReturnInvincible())

CAudio::Instance()->SetSpeedByID(AUDIO\_1\_1,1000);

}

//受傷回復設定

if(Invincible==Hurt\_end){Hurt=0;SetState();};

//暈眩變數設定

if(IsFaint){

if(NowTakeObj!=NULL)ReleaseNowTakeObj();

//石頭暈眩修正

if(IsFaint<7){//暈眩圖片寬度不一，此條件防止連續修正

int i=0,j=0;

while(Maps->IfCollision(wx,wy+i,width,height,true))i--;//先修正WY 如修正超過一個物件高則修正WX

if(-i>34){

i=0;

while(Maps->IfCollision(wx-LRflag\*j+!LRflag\*j,wy,width,height,true))j++;

//根據人物面對方向往反方向修正

if(j>45){

j=0;

//反方向修正超過一個物件寬則朝相同方向修正

while(Maps->IfCollision(wx-LRflag\*j+!LRflag\*j,wy,width,height,true))j--;

if(-j>45)j=0;//三種方向修正都超過一個物件長寬則不修正(卡在石頭裡)

}

wx-=(LRflag\*j-!LRflag\*j);

}

wy+=i;

}

IsFaint++;

if(IsFaint>50){IsFaint=0;SetState();}

setFlag(0,1,1,1,1,1,1);

}

//流汗動畫顯示變數設定

if(NowTaken==APPLE)ani\_sweat\_count++;

else ani\_sweat\_count=0;

if(ani\_sweat\_count>5/\*週期\*/)ani\_sweat\_count=1; //更改週期亦需改變每顆汗流下的位移(onshow)

if(ani\_sweat\_count==1){

ani\_sweat\_x=wx+Maps->ReturnNowX();

ani\_sweat\_wy=wy;

}

//2P移動鎖螢幕

canMoveMapX=true;

canMoveMapY=true;

temp = Maps->GetRoute();

if((temp&8) && wx<=LRMargin || (temp&4) && (wx>=MWIDTH-width-LRMargin))

canMoveMapX = false;

if((temp&2) && wy<=Jump\_Fix || (temp&1) && wy>=(MHEIGHT-2\*height-Jump\_Fix))

canMoveMapY = false;

}

void ChipDale::InitialWidthHeight(){

width = animation[IsDale][0][0][0].Width();

height = animation[IsDale][0][0][0].Height();

}

int ChipDale::CollisionChipDale(int Direct,int passSpeed,int mode){

int temp=0,ObjFix=0;

const int LRspace=5;

if(Partner == NULL||!Partner->Alive)return 0;

passSpeed = abs(passSpeed);

if(Partner->ReturnNowTakeObj()!=NULL && !Partner->ReturnHideComplete()){

ObjFix = Partner->ReturnNowTakeObj()->ReturnHeight();

wy += ObjFix;

}

if(Direct==0) temp = Partner->IfCollision(wx+LRspace,wy,width-2\*LRspace,height);

else if(Direct==1) temp = Partner->IfCollision(wx+LRspace,wy-passSpeed,width-2\*LRspace,passSpeed);

else if(Direct==2) temp = Partner->IfCollision(wx+LRspace,wy+height,width-2\*LRspace,passSpeed);

else if(Direct==4) temp = Partner->IfCollision(wx+LRspace-passSpeed,wy,passSpeed,height);

else if(Direct==8) temp = Partner->IfCollision(wx-LRspace+width,wy,passSpeed,height);

if(ObjFix!=0)

wy -= ObjFix;

if(temp>0){

if(!Partner->IfCollision(Direct,passSpeed)){

if(mode){

if(Direct==1) Partner->SetWxWy(0,-passSpeed,true);

else if(Direct==4) Partner->SetWxWy(-passSpeed,0,true);

else if(Direct==8) Partner->SetWxWy(passSpeed,0,true);

}

return 1;//發生碰撞，推擠2P

}

return 2;//發生碰撞，不能推擠2P

}

return 0;//無發生碰撞

}

void ChipDale::TriggerObj(int Derict){

int temp=0,fixX=0,fixY=0,ox,oy;

if(!Alive)return;

if(Maps->MoveMap(0)&1)fixY = Jump\_Fix;

if(Maps->MoveMap(0)&4)fixX = SPEED;

if(Maps->MoveMap(0)&8)fixX = -SPEED;

while(true){

if(Derict==0) temp = Maps->IfCollision(wx+fixX,wy+fixY,width,height,true,true);

else if(Derict==1) temp = Maps->IfCollision(wx+fixX,wy+fixY+height,width,abs(UpSpeed),false,true,true);

else if(Derict==2) temp = Maps->IfCollision(wx+fixX,wy+fixY-UpSpeed,width,abs(UpSpeed),true,true);

else if(Derict==4) temp = Maps->IfCollision(wx+width+fixX,wy+fixY,ONEOBJX,height,true,true);

else if(Derict==3) temp = Maps->IfCollision(wx-ONEOBJX+fixX,wy+fixY,ONEOBJX,height,true,true);

switch(temp/E8){

case FLOWER:

Score\_Flower++;

Maps->ClearObstacle(temp);

break;

case STAR:

Score\_Star++;

Maps->ClearObstacle(temp);

break;

case 8:

Health=3;

Maps->ClearObstacle(temp);

break;

case 9:

CAudio::Instance()->SetSpeedByID(AUDIO\_1\_1,1500);

Maps->ClearObstacle(temp);

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

if(CanThrow[i]==NULL){

CanThrow[i] = new Explosion(temp%E4-1,temp%E8/E4-1,9,this);

break;

}

if(i+1==CanTakeNum){TRACE("SetNowTaken: No Find In CanThrow\n");Sleep(100000);}

}

break;

case 10://Cheese Box

Maps->ClearObstacle(temp);

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

if(CanThrow[i]==NULL){

CanThrow[i] = new Explosion(temp%E4-1,temp%E8/E4-1,CHEESE,this);

break;

}

if(i+1==CanTakeNum){TRACE("SetNowTaken: No Find In CanThrow\n");Sleep(100000);}

}

break;

case 11:

Maps->ClearObstacle(temp);

break;

case 12:

Maps->ClearObstacle(temp);

break;

case DOOR:

Lock();

if(Partner!=NULL)Partner->Lock();

ToolCDC::Fadeout();

ox = temp%E4-1;

oy = temp%E8/E4-1;

Maps->Teleport = Maps->GetRoute(ox\*ONEOBJX/MWIDTH,oy\*ONEOBJY/MHEIGHT)>>7;

Lock(0);

if(Partner!=NULL)Partner->Lock(0);

return;

case 17:

Maps->ClearObstacle(temp);

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

if(CanThrow[i]==NULL){

CanThrow[i] = new Explosion(temp%E4-1,temp%E8/E4-1,7,this);

CanThrow[i]->ReFixXY(Maps);

break;

}

if(i+1==CanTakeNum){TRACE("SetNowTaken: No Find In CanThrow\n");Sleep(100000);}

}

return;

default:

return ;

}

}

}

void ChipDale::ReleaseNowTakeObj(){

if(NowTakeObj==NULL||!Alive) return;

if(NowTaken==STONE || NowTaken==MonsterOrderSize+OrderSize+0)

NowTakeObj->Throw(0);

else

NowTakeObj->Throw(1);

NowTakeObj=NULL;

NowTaken=-6;

}

**object.cpp**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Object實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Object::Object()

{

NowAct = 1; //石頭拿起

ThisObjValue = 0;

wx = wy = 0;

CanAttackMode = 1;

Owener = NULL;

Direct = 0;

}

void Object::RecoverObj(ChipDale \*player){

Owener = player;//記住自己的主人

NowAct = 1;

ThisObjValue = 0;

}

void Object::CollisionMonster(Monster \*\*monster)

{

int temp;

if(NowAct==0 || Owener==NULL || CanAttackMode==2) return; //NowAct==0 準備投胎 , 沒有主人不行 , CanAttackMode==2 不攻擊任何人

if(NowAct==1 && Owener->ReturnHideComplete()){//NowAct==1 且主人蹲著 須要先修正wx wy (目前與主人關西)

wx = Owener->ReturnWX();

wy = Owener->ReturnWY()+Owener->ReturnHeight()-height;

}

else if(NowAct!=2)return; //NowAct==2 為運動狀態才能攻擊人 , 但如果NowAct==1 有例外 主人蹲著 要判斷碰撞

for(int i=0;i<ONE\_LEVEL\_MONSTER\_NUM && monster[i]!=NULL;i++){

if(monster[i]->ReturnNowAct()<=0) continue; //怪物運動狀態為 0 表示尚未遇到 , -1 已死亡

temp = monster[i]->IfCollision(wx,wy,width,height);//測試碰撞

switch(temp){

case 1:case 4://箱子的左上左下碰到怪物

if(monster[i]->KillMonster(4)){

CollisionReact(temp,monster[i]);

if(NowAct==1 && Owener->ReturnHideComplete()) Owener->ReleaseNowTakeObj();//打完怪物丟出此物品

NowAct=4;//表示已經丟出去打到怪物

i=ONE\_LEVEL\_MONSTER\_NUM;//跳出此迴圈 以防double kill

}

break;

case 2:case 3://箱子的右上右下碰到怪物

if(monster[i]->KillMonster(8)){

CollisionReact(temp,monster[i]);

if(NowAct==1 && Owener->ReturnHideComplete())Owener->ReleaseNowTakeObj();//打完怪物丟出此物品

NowAct=4;//表示已經丟出去打到怪物

i=ONE\_LEVEL\_MONSTER\_NUM;//跳出此迴圈 以防double kill

}

break;

case 5://箱子整個在怪物裡面 幾乎不可能發生

if(monster[i]->KillMonster(1)){

CollisionReact(temp,monster[i]);

if(NowAct==1 && Owener->ReturnHideComplete())Owener->ReleaseNowTakeObj();//打完怪物丟出此物品

NowAct=4;//表示已經丟出去打到怪物

i=ONE\_LEVEL\_MONSTER\_NUM;//跳出此迴圈 以防double kill

}

break;

}

}

}

void Object::CollisionChipDale(ChipDale \*player){

if(Owener==NULL)return;//沒有主人的物件不攻擊人

if(CanAttackMode==2 || CanAttackMode==3)return;//CanAttackMode = 2 表示此物件不具攻擊性) , 3 表示此物件只攻擊怪物

if(player->ReturnInvincible())return;//此玩家為無敵 不攻擊

if(NowAct==2||NowAct==4){//NowAct==2 表示此物件在運動的過程 且只有運動過程才會打傷人

if(Owener==player){

//TRACE("Self\n");

if(CanAttackMode==1)return;//CanAttackMode == 1 表示此物件不會攻擊自己的主人

if(CanAttackMode==0 && Owener->ReturnNowAct()>3 && Owener->ReturnNowAct()!=4

&& !Owener->ReturnHideComplete()&&!(Direct&1))return;

if((Direct&32)&&Owener->ReturnNowAct()==4)return;

//當CanAttackMode==0時 且箱子自己的主人的NowAct>=3 則不打它主人

}

int ThrowFix=(Owener==player)\*(4\*(LRflag)-4\*(!LRflag));

if(player->IfCollision(wx+ThrowFix,wy,width,height)){

player->Faint();

}

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ChipDale\_Dead實作 \*/

// \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap ChipDale\_Dead::frame\_pic[2][2];

ChipDale\_Dead::ChipDale\_Dead(ChipDale\* chip){

isDale =chip->IsDale;

LRflag =chip->LRflag;

wx =chip->wx;

wy =chip->wy;

Owener =chip;

CanAttackMode=2;

height=frame\_pic[0][0].Height();

NowAct=2; //運動

NoCollision = true;

UpSpeed = 0;

JumpSPEED = 30;

MoveSPEED = SPEED/2;

Wait\_A\_Minute=-1;

CAudio::Instance()->PauseLevelMusic(CGameStateRun::NowLevel);

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

}

ChipDale\_Dead::~ChipDale\_Dead(){

CAudio::Instance()->PlayLevelMusic(CGameStateRun::NowLevel);

}

void ChipDale\_Dead::Loading(){

frame\_pic[0][0].LoadBitmapA("Bitmaps/action/chip/Hurt\_L.bmp",PURPLE);

frame\_pic[0][1].LoadBitmapA("Bitmaps/action/chip/Hurt\_R.bmp",PURPLE);

frame\_pic[1][0].LoadBitmapA("Bitmaps/action/dale/Hurt\_L.bmp",PURPLE);

frame\_pic[1][1].LoadBitmapA("Bitmaps/action/dale/Hurt\_R.bmp",PURPLE);

}

void ChipDale\_Dead::OnMove(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

//死亡之等待時間

if(Wait\_A\_Minute>=0){

if(Wait\_A\_Minute>0)Wait\_A\_Minute--;

else {

NowAct=0;

CAudio::Instance()->PlayLevelMusic(CGameStateRun::NowLevel);

if((Owener->Partner==NULL||Owener->Partner->Life<=0)&&Owener->Life>0){

//回紀錄點

map->Set\_toRecord();

Owener->Reset(10,10);

}else if(Owener->Life<=0&&(Owener->Partner==NULL||Owener->Partner->Life<=0)){

//GameOver

CGameStateRun::TimeToGo=true;

}

}

return;

}

//Dead物件生命週期結束設定

if(wy+frame\_pic[isDale][LRflag].Height()>MHEIGHT+100 ){

Wait\_A\_Minute=33;

if(Owener->Life>0&&Owener->Partner!=NULL&&Owener->Partner->Life>0){

//復活

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

if(ChipDale::CanThrow[i]!=NULL)continue;

ChipDale::CanThrow[i] = new ChipDale\_Resurrect(this);

break;

}

}else if((Owener->Partner==NULL||Owener->Partner->Life<=0)&&Owener->Life>0){

//回紀錄點

}else if(Owener->Life<=0&&(Owener->Partner==NULL||Owener->Partner->Life<=0))

{

//GameOver

}

}

if(!IsJump && wy+height<MHEIGHT){

IsJump = true;

Jump();

}

if(!LRflag&&wx+width+MoveSPEED<MWIDTH) MoveRight(map);

else if(LRflag&&wx-MoveSPEED>0) MoveLeft(map);

FallingDown(map);

}

void ChipDale\_Dead::OnShow(MapManage\* map){

frame\_pic[isDale][LRflag].SetTopLeft(wx,wy);

frame\_pic[isDale][LRflag].ShowBitmap();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ChipDale\_Resurrect實作 \*/

// \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap ChipDale\_Resurrect::frame\_pic[2][2];

ChipDale\_Resurrect::ChipDale\_Resurrect(ChipDale\_Dead\* chip){

isDale =chip->isDale;

LRflag =chip->LRflag;

wx =chip->wx;

wy =chip->wy;

Owener =chip->Owener;

timeCount=0;

CanAttackMode=2;

height=frame\_pic[isDale][LRflag].Height();

width=frame\_pic[isDale][LRflag].Width();

Direct =1;//上升

NowAct=2; //運動

NoCollision = true;

MoveSPEED = SPEED;

if(wx<0)wx=0;

if(wx+width>MWIDTH)wx=MWIDTH-width;

}

void ChipDale\_Resurrect::Loading(){

frame\_pic[0][0].LoadBitmapA("Bitmaps/action/chip/Resurrect\_L.bmp",PURPLE);

frame\_pic[0][1].LoadBitmapA("Bitmaps/action/chip/Resurrect\_R.bmp",PURPLE);

frame\_pic[1][0].LoadBitmapA("Bitmaps/action/dale/Resurrect\_L.bmp",PURPLE);

frame\_pic[1][1].LoadBitmapA("Bitmaps/action/dale/Resurrect\_R.bmp",PURPLE);

}

void ChipDale\_Resurrect::OnMove(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

if(Owener->RIGHT\_FLAG){

LRflag=1;

if(wx+width+MoveSPEED<MWIDTH)MoveRight(map);

}

else if(Owener->LEFT\_FLAG){

LRflag=0;

if(wx-MoveSPEED>0 )MoveLeft(map);

}

//自動上下

if(Direct==1)MoveUp(map);

else if(Direct==2)MoveDown(map);

if(wy<=0)Direct=2;

else if(wy+height>MHEIGHT) Direct=1;

//召喚(1秒後按A手動召喚or10秒過後自動召喚)

if(((Owener->A\_FLAG&&timeCount>33\*1)||timeCount>33\*10)&&(wy+height<MHEIGHT)){

Owener->Reset(wx,wy,LRflag==1);

//Owener->SetWxWy(wx,wy);

NowAct=0;

}else timeCount++;

//防止被推出場

if(wx<0)wx=0;

if(wx+width>MWIDTH)wx=MWIDTH-width;

}

void ChipDale\_Resurrect::OnShow(MapManage\* map){

frame\_pic[isDale][LRflag].SetTopLeft(wx,wy);

//1秒前有閃爍效果

if(timeCount>33||(timeCount%5==2||timeCount%5==3))

frame\_pic[isDale][LRflag].ShowBitmap();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//ChipDale\_Taken實作 \*/

//Note::此物件是以chip的NowTakeObj創造 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CAnimation ChipDale\_Taken::animation[2][2];

ChipDale\_Taken::ChipDale\_Taken(ChipDale\* chip){

isDale =chip->Partner->IsDale;

LRflag =chip->LRflag;

height=animation[0][0].Height();

width=animation[0][0].Width();

wx =chip->wx;

wy =chip->wy-height;

Owener =chip;

Owener->Partner->Alive=false;

CanAttackMode=2;

NowAct=1; //拿在手上

NoCollision = true;

UpSpeed = 0;

JumpSPEED = 30;

MoveSPEED = SPEED;

}

void ChipDale\_Taken::Loading(){

animation[0][0].SetDelayCount(5);

animation[0][1].SetDelayCount(5);

animation[1][0].SetDelayCount(5);

animation[1][1].SetDelayCount(5);

animation[0][0].AddBitmap("Bitmaps/action/chip/taken\_1L.bmp",PURPLE);

animation[0][0].AddBitmap("Bitmaps/action/chip/taken\_2L.bmp",PURPLE);

animation[0][0].AddBitmap("Bitmaps/action/chip/taken\_3L.bmp",PURPLE);

animation[0][1].AddBitmap("Bitmaps/action/chip/taken\_1R.bmp",PURPLE);

animation[0][1].AddBitmap("Bitmaps/action/chip/taken\_2R.bmp",PURPLE);

animation[0][1].AddBitmap("Bitmaps/action/chip/taken\_3R.bmp",PURPLE);

animation[1][0].AddBitmap("Bitmaps/action/dale/taken\_1L.bmp",PURPLE);

animation[1][0].AddBitmap("Bitmaps/action/dale/taken\_2L.bmp",PURPLE);

animation[1][0].AddBitmap("Bitmaps/action/dale/taken\_3L.bmp",PURPLE);

animation[1][1].AddBitmap("Bitmaps/action/dale/taken\_1R.bmp",PURPLE);

animation[1][1].AddBitmap("Bitmaps/action/dale/taken\_2R.bmp",PURPLE);

animation[1][1].AddBitmap("Bitmaps/action/dale/taken\_3R.bmp",PURPLE);

}

void ChipDale\_Taken::Throw(int setDirect){

LRflag=Owener->LRflag;

NowAct=2;

Jump();

}

void ChipDale\_Taken::OnMove(MapManage\* map){

if(NowAct==1){

wx=Owener->wx;

wy=Owener->wy-height;

LRflag=Owener->LRflag;

}

else if(NowAct==2){

if(LRflag&&wx+width<MWIDTH)

MoveRight(map);

else if(!LRflag&&wx>0)

MoveLeft(map);

IsJump = true;

FallingDown(map);

if(UpSpeed<=0){

NowAct=0;

Owener->Partner->Reset(wx,wy,LRflag==1,false);

}

}

}

void ChipDale\_Taken::OnShow(MapManage\* map){

animation[isDale][LRflag].SetBottomLeft(wx,wy,height);

animation[isDale][LRflag].OnMove();

animation[isDale][LRflag].OnShow();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Apple實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap Apple::frame\_pic[2];

Apple::Apple(ChipDale \*player){

width = frame\_pic[0].Width();

height = frame\_pic[0].Height();

Owener = player;//記住自己的主人

}

void Apple::Loading(){

frame\_pic[0].LoadBitmapA("Bitmaps/object/Apple\_L.bmp",PURPLE);

frame\_pic[1].LoadBitmapA("Bitmaps/object/Apple\_R.bmp",PURPLE);

}

void Apple::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

else if(NowAct==1){//拿在手上frame\_pic[Owener->ReturnLR()].SetTopLeft(Owener->ReturnWX()-(width-Owener->ReturnWidth())/2,Owener->ReturnWY()-height\*(!Owener->ReturnHideComplete()));

frame\_pic[Owener->ReturnLR()].ShowBitmap();

}

else{//丟出與落地

frame\_pic[LRflag].SetTopLeft(wx,wy);

frame\_pic[LRflag].ShowBitmap();

}

}

void Apple::OnMove(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

if(NowAct==1) return;//還在人物手上

if(wx < -WinShowBuffer || wx > MWIDTH || wy < -WinShowBuffer){

NowAct = 0;

return;

}

if(Direct&1) MoveUp(map);

if(Direct&4) MoveLeft(map);

if(Direct&8) MoveRight(map);

}

void Apple::Throw(int setDirect){

NowAct=2;

LRflag = Owener->ReturnLR();

Direct = setDirect;

JumpSPEED = 0;

MoveSPEED = SPEED \*2;

NoCollision = true;

IsJump = false;

if(Direct&1)Direct=1;

else Direct = Direct&12;

wx = Owener->ReturnWX()-(width-Owener->ReturnWidth())/2;

wy = Owener->ReturnWY()+Owener->ReturnHeight()-height\*(!Owener->ReturnHideComplete())-40;//離地面 40 公分

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Angel實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CAnimation Angel::frame\_pic[2];

Monster \*\*Angel::monster;

Angel::Angel(int setWx,int setWy,ChipDale \*player):TheBeeMaxSpeed(SPEED\*2){//Angel X Y 方向的移動速度最高為 SPEED\*2

width = frame\_pic[0].Width();

height = frame\_pic[0].Height();

Owener = player;//記住自己的主人

wx = setWx;

wy = setWy;

NowAttack=-1;

CanAttackMode = 3;

LRflag=0;

NoCollision = true;

}

void Angel::Loading(){

frame\_pic[0].AddBitmap("Bitmaps/action/Angel/bee\_1R.bmp",PURPLE);

frame\_pic[1].AddBitmap("Bitmaps/action/Angel/bee\_1L.bmp",PURPLE);

monster = ChipDale::AllMonster;

}

void Angel::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

frame\_pic[LRflag].SetBottomLeft(wx,wy,height);

frame\_pic[LRflag].OnShow();

}

void Angel::OnMove(MapManage\* map){

int tempX,tempY,tempT;

if(NowAct==0){

return;//死亡準備投胎

}

NowAct=2;

if(!Owener->ReturnInvincible()){

if(wx < -WinShowBuffer || wx > MWIDTH || wy < -WinShowBuffer){

NowAct = 0;

return;

}

MoveSPEED = TheBeeMaxSpeed;

if(((ChipDale::Maps->GetRoute())&8)==8) MoveLeft(map);

else MoveRight(map);

MoveUp(map);

NowAct=4;

return;

}

if(NowAttack!=-1 && (monster[NowAttack]->ReturnNowAct()<=0 || monster[NowAttack]->ReturnNowAct()>=100)) NowAttack=-1;//Look if Monster is Alive?

if(NowAttack==-1){//不知道要去攻擊哪一個怪物 搜尋目標

for(int i=0;i<ONE\_LEVEL\_MONSTER\_NUM && monster[i]!=NULL;i++){

if(monster[i]->ReturnNowAct()>=1 && monster[i]->ReturnNowAct()<100 && monster[i]->ReturnCanTrace()){

NowAttack = i;

break;

}

}

}

if(NowAttack==-1){

LRflag = !LRflag;

if(LRflag) tempX = (Owener->ReturnWX()+(Owener->ReturnWidth()-width)/2-5)-wx;//離人物中心偏左5px

else tempX = (Owener->ReturnWX()+(Owener->ReturnWidth()-width)/2+5)-wx;//離人物中心偏左5px

tempY = (Owener->ReturnWY()-height)-wy;

}

else{

tempX = monster[NowAttack]->ReturnWX()-wx;

tempY = monster[NowAttack]->ReturnWY()-wy;

if(tempX<0) LRflag = 1;

else LRflag = 0;

}

//華麗移動(新版)

if(tempX>tempY){

tempT = tempX/TheBeeMaxSpeed;

if(abs(tempX)>=TheBeeMaxSpeed) MoveSPEED=TheBeeMaxSpeed;

else MoveSPEED=abs(tempX);

if(tempX<0) MoveLeft(map);

else if(tempX>0) MoveRight(map);

if(abs(tempT)==0)tempT=1;

if(abs(tempY)>=TheBeeMaxSpeed) MoveSPEED=TheBeeMaxSpeed;

else MoveSPEED=abs(tempY/tempT);

if(tempY<0) MoveUp(map);

else if(tempY>0) MoveDown(map);

}

else{

tempT = tempY/TheBeeMaxSpeed;

if(abs(tempY)>=TheBeeMaxSpeed) MoveSPEED=TheBeeMaxSpeed;

else MoveSPEED=abs(tempY);

if(tempY<0) MoveUp(map);

else if(tempY>0) MoveDown(map);

if(abs(tempT)==0)tempT=1;

if(abs(tempX)>=TheBeeMaxSpeed) MoveSPEED=TheBeeMaxSpeed;

else MoveSPEED=abs(tempX/tempT);

if(tempX<0) MoveLeft(map);

else if(tempX>0) MoveRight(map);

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Ball實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap Ball::frame\_pic[2];

Ball::Ball(ChipDale \*player){

width = frame\_pic[0].Width();

height = frame\_pic[0].Height();

Owener = player;//記住自己的主人

CanAttackMode = 1;

LRflag=0;

NowAct=1;

Rebound\_times=0;

IsJump=false;

NoCollision=false;

MoveSPEED=SPEED\*2;

UpSpeed=0;

JumpSPEED=40;

}

void Ball::Loading()

{

frame\_pic[0].LoadBitmapA("Bitmaps/object/Ball\_L.bmp",PURPLE);

frame\_pic[1].LoadBitmapA("Bitmaps/object/Ball\_R.bmp",PURPLE);

}

void Ball::OnMove(MapManage\* map)

{

if(NowAct==0) return;//死亡準備投胎

if(NowAct==1) return;//還在人物手上

if(NowAct==3) {

CanAttackMode=1;

return; //在地上

}

if(wx < -WinShowBuffer || wx > MWIDTH || wy < -WinShowBuffer){

NowAct = 0;

return;

}

if(NowAct==2||NowAct==4) //運動狀態

{

if(Direct&4){

MoveLeft(map);

LRflag=0;

if(wx<=0){

wx=0;

Direct=8;

Rebound\_times--;

}

}

else if(Direct&8){

MoveRight(map);

LRflag=1;

if(wx+width>=MWIDTH){

wx=MWIDTH-width;

Direct=4;

Rebound\_times--;

}

}else if(Direct&1){

MoveUp(map);

if(wy<0){

Direct=2;

Rebound\_times--;

}

}else if(!MoveDown(map)){

Rebound\_times--;

}

if(Rebound\_times<=0){

//Direct=0;

IsJump=true;

FallingDown(map);

CanAttackMode=0;

if(IsJump==false)

{

ThisObjValue = map->FillObstacle(OrderSize+MonsterOrderSize + 0,wx,wy);

wx = (ThisObjValue%E4-1)\*ONEOBJX-map->ReturnNowX();

wy = (ThisObjValue%E8/E4-1)\*ONEOBJY-map->ReturnNowY();

NowAct=5;

}

}

else if(Rebound\_times==1)CanAttackMode=0;

}

}

void Ball::OnShow(MapManage\* map)

{

if(NowAct==0) return;//死亡準備投胎

else if(NowAct==1){//拿在手上frame\_pic[Owener->ReturnLR()].SetTopLeft(Owener->ReturnWX()-(width-Owener->ReturnWidth())/2,Owener->ReturnWY()-height\*(!Owener->ReturnHideComplete()));

frame\_pic[Owener->ReturnLR()].ShowBitmap();

}

else{//丟出與落地

frame\_pic[LRflag].SetTopLeft(wx,wy);

frame\_pic[LRflag].ShowBitmap();

}

}

void Ball::Throw(int setdirect)

{

NowAct=2;

CanAttackMode=1;

Rebound\_times=2;

wx = Owener->ReturnWX()+(Owener->ReturnWidth()-width)/2;

if(Owener->ReturnLastAct()==4)

wy = Owener->ReturnWY()+Owener->ReturnHeight()-height;

else

wy = Owener->ReturnWY();

if(setdirect&1)

Direct=1;

else if(setdirect&4)

Direct=4;

else if(setdirect&8)

Direct=8;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Craft實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap Craft::frame\_pic[2];

Craft::Craft(ChipDale \*player){

width = frame\_pic[0].Width();

height = frame\_pic[0].Height();

Owener = player;//記住自己的主人

}

void Craft::Loading(){

frame\_pic[0].LoadBitmapA("Bitmaps/object/Craft\_L.bmp",PURPLE);

frame\_pic[1].LoadBitmapA("Bitmaps/object/Craft\_R.bmp",PURPLE);

}

void Craft::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

else if(NowAct==1){//拿在手上

frame\_pic[Owener->ReturnLR()].SetTopLeft(Owener->ReturnWX()-(width-Owener->ReturnWidth())/2,Owener->ReturnWY()-height\*(!Owener->ReturnHideComplete()));

frame\_pic[Owener->ReturnLR()].ShowBitmap();

}

else{//丟出與落地

frame\_pic[LRflag].SetTopLeft(wx,wy);

frame\_pic[LRflag].ShowBitmap();

}

}

void Craft::OnMove(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

if(NowAct==1) return;//還在人物手上

if(wx < -WinShowBuffer || wx > MWIDTH || wy < -WinShowBuffer){

NowAct = 0;

return;

}

if(Direct&1) MoveUp(map);

if(Direct&4) MoveLeft(map);

if(Direct&8) MoveRight(map);

}

void Craft::Throw(int setDirect){

NowAct=2;

LRflag = Owener->ReturnLR();

Direct = setDirect;

JumpSPEED = 0;

MoveSPEED = SPEED \*2;

NoCollision = true;

IsJump = false;

if(Direct&1)Direct=1;

else Direct = Direct&12;

wx = Owener->ReturnWX()-(width-Owener->ReturnWidth())/2;

if(Owener->ReturnLastAct()==4)

wy = Owener->ReturnWY()+Owener->ReturnHeight()-height;

else

wy = Owener->ReturnWY()+Owener->ReturnHeight()-height\*(!Owener->ReturnHideComplete())-40;//離地面 40 公分

}

void Craft::CollisionReact(int setDirect,CFrame \*which){

switch(setDirect){

case 1:case 4:

Direct = 9;

break;

case 2:case 3:

Direct = 5;

break;

case 5:

Direct = 1;

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Explosion實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CAnimation Explosion::frame\_pic;

Explosion::Explosion(int setOx,int setOy,int setChangeWhat,ChipDale \*player){

MapManage \*map = ChipDale::Maps;

width = frame\_pic.Width();

height = frame\_pic.Height();

ChangeWhat = setChangeWhat;

wx = setOx\*ONEOBJX-map->ReturnNowX();

wy = setOy\*ONEOBJY-map->ReturnNowY();

frame\_pic.Reset();

CanAttackMode = 2;

NowAct=2;

Owener = player;

IfNeedReFix = false;

}

void Explosion::Loading(){

frame\_pic.SetDelayCount(2);

frame\_pic.AddBitmap("Bitmaps/action/Explosion/Explosion\_1.bmp",PURPLE);

frame\_pic.AddBitmap("Bitmaps/action/Explosion/Explosion\_2.bmp",PURPLE);

frame\_pic.AddBitmap("Bitmaps/action/Explosion/Explosion\_3.bmp",PURPLE);

frame\_pic.AddBitmap("Bitmaps/action/Explosion/Explosion\_4.bmp",PURPLE);

}

void Explosion::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

frame\_pic.SetBottomLeft(wx,wy,height);

frame\_pic.OnShow();

}

void Explosion::OnMove(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

Object \*\*CanThrow = ChipDale::CanThrow;

bool IfNeedReFix=false;

if(frame\_pic.IsFinalBitmap()){

switch(ChangeWhat){ //改變成甚麼要寫在這邊

case STAR:

for(int i=0,IfNeedReFix=false;i<CanTakeNum;i++){

if(CanThrow[i]==this) IfNeedReFix=true;

if(CanThrow[i]==NULL){

CanThrow[i] = new Star(wx,wy);

if(IfNeedReFix) CanThrow[i]->ReFixXY(map);

break;

}

if(i+1==CanTakeNum){

TRACE("CanThrow OverFlow!!!\n");

Sleep(10000);

}

}

break;

case BOX\_ANGLE:

for(int i=0,IfNeedReFix=false;i<CanTakeNum;i++){

if(CanThrow[i]==this) IfNeedReFix=true;

if(CanThrow[i]==NULL){

CanThrow[i] = new Angel(wx,wy,Owener);

break;

}

if(i+1==CanTakeNum){

TRACE("CanThrow OverFlow!!!\n");

Sleep(10000);

}

}

Owener->SetInvincible(1);

break;

case CHEESE:

for(int i=0,IfNeedReFix=false;i<CanTakeNum;i++){

if(CanThrow[i]==this) IfNeedReFix=true;

if(CanThrow[i]==NULL){

CanThrow[i] = new Cheese(wx,wy);

if(IfNeedReFix) CanThrow[i]->ReFixXY(map);

break;

}

if(i+1==CanTakeNum){

TRACE("CanThrow OverFlow!!!\n");

Sleep(10000);

}

}

}

NowAct=0;

}

frame\_pic.OnMove();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Star實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap Star::frame\_pic;

Star::Star(int setWx,int setWy){

width = frame\_pic.Width();

height = frame\_pic.Height();

wx = setWx;

wy = setWy;

CanAttackMode = 2;

NowAct=2;

UpSpeed=30;

IsJump = false;

}

void Star::Loading(){

frame\_pic.LoadBitmapA("Bitmaps/object/7.bmp",PURPLE);

}

void Star::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

frame\_pic.SetTopLeft(wx,wy);

frame\_pic.ShowBitmap();

}

void Star::OnMove(MapManage\* map){

if(NowAct==-1){

NowAct=2;

return;

}

if(NowAct==0) return;//死亡準備投胎

lastUpSpeed=UpSpeed;

FallingDown(map);

if(!IsJump && lastUpSpeed==0){

map->FillObstacle(7,wx,wy);

NowAct=0;

}

else if(!IsJump){

UpSpeed = abs(lastUpSpeed)\*2/3;//落地彈跳 2/3為系數

IsJump=true;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Cheese實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap Cheese::frame\_pic;

Cheese::Cheese(int setWx,int setWy){

width = frame\_pic.Width();

height = frame\_pic.Height();

wx = setWx;

wy = setWy;

CanAttackMode = 2;

NowAct=2;

UpSpeed=30;

IsJump = false;

}

void Cheese::Loading(){

frame\_pic.LoadBitmapA("Bitmaps/object/Cheese.bmp",PURPLE);

}

void Cheese::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

frame\_pic.SetTopLeft(wx,wy);

frame\_pic.ShowBitmap();

}

void Cheese::OnMove(MapManage\* map){

Object \*\*CanThrow = ChipDale::CanThrow;

if(NowAct==-1){

NowAct=2;

return;

}

if(NowAct==0) return;//死亡準備投胎

lastUpSpeed=UpSpeed;

FallingDown(map);

if(!IsJump && lastUpSpeed==0&&NowAct!=3){

//map->FillObstacle(7,wx,wy);

//NowAct=0;

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

if(CanThrow[i]==NULL){

CanThrow[i] = new Greedy(0,wy+height,this);

TRACE("Greedy: %d %d\n",wx,wy);

break;

}

if(i+1==CanTakeNum){

TRACE("CanThrow OverFlow!!!\n");

Sleep(10000);

}

}

NowAct=3;

}

else if(!IsJump){

UpSpeed = abs(lastUpSpeed)\*2/3;//落地彈跳 2/3為系數

IsJump=true;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Stone實作 \*/

//此Class中的Direct編碼 => 1上 , 2下 , 4左 , 8右 , \*/

// 16跑一次 , 32ReFix \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap Stone::frame\_pic[2];

Stone::Stone(ChipDale \*player){

width = frame\_pic[0].Width();

height = frame\_pic[0].Height();

Owener = player;//記住自己的主人

CanAttackMode = 0;

HideThrow=false;

ReboundLR=0;

ReboundSpeed=0;

NoIgnore\_eat = true;

}

void Stone::Loading(){

frame\_pic[0].LoadBitmapA("Bitmaps/object/Stone\_L.bmp",PURPLE);

frame\_pic[1].LoadBitmapA("Bitmaps/object/Stone\_R.bmp",PURPLE);

}

void Stone::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

else if(NowAct==1){//拿在手上frame\_pic[Owener->ReturnLR()].SetTopLeft(Owener->ReturnWX()-(width-Owener->ReturnWidth())/2,Owener->ReturnWY()-height\*(!Owener->ReturnHideComplete()));

frame\_pic[Owener->ReturnLR()].ShowBitmap();

}

else{//丟出與落地

frame\_pic[LRflag].SetTopLeft(wx,wy);

frame\_pic[LRflag].ShowBitmap();

//TRACE("show!!\n");

}

}

bool Stone::MoveLeft(MapManage\* map){

if(!map->IfCollision(wx-MoveSPEED,wy,MoveSPEED,height,true)){

wx -= MoveSPEED;

return true;

}else{

int i=0;

while(!map->IfCollision(wx-i,wy,MoveSPEED,height,true))i++;

wx-=i;

Jump();

ReboundLR=1;

ReboundSpeed=(MoveSPEED-i)/4;

}

return false;

}

bool Stone::MoveRight(MapManage\* map){

if(!map->IfCollision(wx+width,wy,MoveSPEED,height,true)){

wx += MoveSPEED;

return true;

}else{

int i=0;

while(!map->IfCollision(wx+width,wy,i,height,true))i++;

wx+=i;

Jump();

ReboundLR=-1;

ReboundSpeed=(MoveSPEED-i)/4;

}

return false;

}

void Stone::ReBound(MapManage\* map){

if(!IsJump){

ReboundLR=0;

ReboundSpeed=0;

}

if(ReboundLR==-1){

if(!map->IfCollision(wx-MoveSPEED,wy,MoveSPEED,height,true))

wx-=ReboundSpeed;

}

else if(ReboundLR==1){

if(!map->IfCollision(wx+width,wy,MoveSPEED,height,true))

wx+=ReboundSpeed;

}

}

void Stone::OnMove(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

if(NowAct==1) return;//還在人物手上

if(wx < -WinShowBuffer || wx > MWIDTH || wy < -WinShowBuffer){

if(NowAct==3){

map->ClearObstacle(ThisObjValue);

TRACE("clear!!!\n");

}

NowAct = 0;

return;

}

if(NowAct==3) return;

if(!IsJump){

IsJump = false;

ThisObjValue = map->FillObstacle(OrderSize+MonsterOrderSize + 0,

wx/\*+(Direct&4)/4\*(5)\*(!HideThrow)+(Direct&8)/8\*(-5)\*(!HideThrow)\*/,wy);

//(Direct&4)/4\*(5)+(Direct&8)/8\*(-5) 丟出落地後往回拉 不然一開始不丟遠 會打到自己

wx = (ThisObjValue%E4-1)\*ONEOBJX-map->ReturnNowX();

wy = (ThisObjValue%E8/E4-1)\*ONEOBJY-map->ReturnNowY();

if(Direct&32){//有蹲下砸到怪物 會修正人物位置 用 Direct 32 當編碼

Owener->FixSelf\_Onto\_ObjectTop();

Direct &= 31;

}

NowAct = 3;

return;

}

if(Direct&4 && !(Direct&16)) {MoveLeft(map);Direct|=16;}

if(Direct&8 && !(Direct&16)) {MoveRight(map);Direct|=16;}

ReBound(map);

if(FallingDown(map))ReboundLR=0;;

}

void Stone::Throw(int setDirect){

NowAct=2;

LRflag = Owener->ReturnLR();

Direct = setDirect | (Direct&32);

if(Direct&1)Direct=1;

UpSpeed = 0;

MoveSPEED = 45-5;

JumpSPEED = 30;

NoCollision = false;

IsJump = true;

if(Direct&1) Jump();

//TRACE("Direct: %d\n",Direct);

wx = Owener->ReturnWX()+(Owener->ReturnWidth()-width)/2;

if(Owener->ReturnLastAct()==4)

wy = Owener->ReturnWY()+Owener->ReturnHeight()-height;

else

wy = Owener->ReturnWY()-height\*(!Owener->ReturnHideComplete());

}

void Stone::CollisionReact(int setDirect,CFrame \*which){

if(NowAct==1) Direct |= 32;//有蹲下砸到怪物 會修正人物位置 用 Direct 32 當編碼

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Greedy實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CAnimation Greedy::frame\_pic[2];

CMovingBitmap Greedy::frame\_hole;

Greedy::Greedy(int setWx,int setBottonWy,Object \*setCheeseObj){

width = frame\_pic[1].Width();

height = frame\_pic[1].Height();

wx = setWx;

wy = setBottonWy-height;

Owener = NULL;//記住自己的主人

CanAttackMode = 2;

if(setWx<=MWIDTH/2){

LRflag=1;

wx -= width;

}

else

LRflag=0;

NowAct = 2;

MoveSPEED = SPEED\*4/5;

CheeseObj = setCheeseObj;

NoCollision = false;

showMouse=true;

}

void Greedy::Loading(){

frame\_pic[0].SetDelayCount(5);

frame\_pic[0].AddBitmap("Bitmaps/action/Greedy/Greedy\_1L.bmp",PURPLE);

frame\_pic[0].AddBitmap("Bitmaps/action/Greedy/Greedy\_2L.bmp",PURPLE);

frame\_pic[1].SetDelayCount(5);

frame\_pic[1].AddBitmap("Bitmaps/action/Greedy/Greedy\_1R.bmp",PURPLE);

frame\_pic[1].AddBitmap("Bitmaps/action/Greedy/Greedy\_2R.bmp",PURPLE);

frame\_hole.LoadBitmapA("Bitmaps/object/hole.bmp",PURPLE);

}

void Greedy::OnShow(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

if(showMouse && NowAct>0){//拿在手上

frame\_pic[LRflag].SetBottomLeft(wx,wy,height);

frame\_pic[LRflag].OnShow();

frame\_pic[LRflag].OnMove();

}

else if(!showMouse){

frame\_hole.SetTopLeft(wx,wy);

frame\_hole.ShowBitmap();

}

}

void Greedy::OnMove(MapManage\* map){

if(NowAct==0) return;//死亡準備投胎

if(NowAct==1) return;//還在人物手上

if(!showMouse) return;//變成洞了

if(wx>=-width && wx<=width+MWIDTH){

if(!(!LRflag&&MoveLeft(map)) && !(LRflag&&MoveRight(map))){

if(!NoCollision){

NoCollision = true;

if(!LRflag){

PassWx = wx;

MoveLeft(map);

}

else{

PassWx = wx+width;

MoveRight(map);

}

}

}

else if(CheeseObj!=NULL && CheeseObj->IfCollision(wx+(width/2)\*(!LRflag),wy,width/2,height)){//只運算老鼠的一半寬度 這樣看起來比較像有吃到

CheeseObj->SetNowAct(0);

CheeseObj=NULL;

}

else if(NoCollision && (!LRflag&&wx<=PassWx || LRflag&&wx>=PassWx)){

showMouse = false;

map->FillObstacle(13,PassWx,wy-11,false);//11為寫死的值 兩個圖片(Greedy & hole)高的差值

map->FillObstacle(13,PassWx,wy+height-34,false);//34為寫死的 過關門的高度

wx = PassWx;

wy -= 11;

}

}

else //自動消失 (離開顯示螢幕)

NowAct=0;

}

**monster.cpp**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Monster 實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

Monster::Monster(){

CanTrace = true;

wait=0;

MoveSPEED = SPEED;

JumpSPEED=0;

NoCollision=false;

LR\_Space = 0;

wait = 0;

UpSpeed = 0;

LR\_flag = 0;

NowAct = 0;

IsJump = false;

Health = 1;

}

void Monster::CollisionChipDale(ChipDale \*player){

if(NowAct>=1&&NowAct<100){

if(player->ReturnInvincible())return;

if(player->ReturnHideComplete())return;

if(player->IfCollision(wx,wy,width,height)){

player->GetHurt();

}

}

}

ChipDale\* Monster::Detect(ChipDale \*\*player,int \*WLength,int \*HLength,int WRange,int HRange,bool IfTraceInvincible){

ChipDale \*result = NULL;

int twx,twy,twidth,theight,tWLength,tHLength;

MapManage \*Maps = ChipDale::Maps;

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

if(player[i]==NULL || player[i]->ReturnTakenByPartner()) continue;

if(player[i]->ReturnHealth()<=0)continue;

if(!IfTraceInvincible && player[i]->ReturnInvincible())continue;

twidth = 0;

theight = 0;

twx = wx + width/2;

twy = wy;

if(WRange<0){

twx -= MWIDTH;

twidth = 2\*MWIDTH + width;

}

else{

twidth = WRange;

if(!LRflag)

twx -= WRange;

}

if(HRange<0){

twy = (Maps->ReturnNowY()+twy)/MHEIGHT\*MHEIGHT-Maps->ReturnNowY()-WinShowBuffer;

theight = MHEIGHT+WinShowBuffer;

}

else{

twy -= HRange;

theight = 2\*HRange + height;

}

if(player[i]->IfCollision(twx,twy,twidth,theight)){

if(result!=NULL){

tWLength = player[i]->ReturnWX() - wx;

tHLength = player[i]->ReturnWY() - wy;

if(tWLength\*tWLength+tHLength\*tHLength < (\*WLength)\*(\*WLength)+(\*HLength)\*(\*HLength)){

\*WLength = tWLength;

\*HLength = tHLength;

result = player[i];

}

}

else{

\*WLength = player[i]->ReturnWX() - wx;

\*HLength = player[i]->ReturnWY() - wy;

result = player[i];

}

}

}

return result;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//MachineDog 實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CAnimation MachineDog::frame\_monster[2][2];

MachineDog::MachineDog(MapManage \*map,int SetOx,int SetOy){

MoveSPEED = SPEED\*3/5;

JumpSPEED=30;

NoCollision=false;

LR\_Space = 15;

wait = 15;

width = frame\_monster[0][0].Width();

height = frame\_monster[0][0].Height();

wx = SetOx\*ONEOBJX - map->ReturnNowX();

wy = SetOy\*ONEOBJY - map->ReturnNowY();

lastWy = wy;

UpSpeed = 0;

LR\_flag = 0;

NowAct = 0;

IsJump = false;

Health = 1;

}

void MachineDog::Loading(){

frame\_monster[0][0].SetDelayCount(5);

frame\_monster[0][0].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_stand\_1L.bmp",PURPLE);

frame\_monster[0][0].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_stand\_2L.bmp",PURPLE);

frame\_monster[0][1].SetDelayCount(5);

frame\_monster[0][1].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_stand\_1R.bmp",PURPLE);

frame\_monster[0][1].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_stand\_2R.bmp",PURPLE);

frame\_monster[1][0].SetDelayCount(5);

frame\_monster[1][0].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_run\_1L.bmp",PURPLE);

frame\_monster[1][0].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_run\_2L.bmp",PURPLE);

frame\_monster[1][1].SetDelayCount(5);

frame\_monster[1][1].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_run\_1R.bmp",PURPLE);

frame\_monster[1][1].AddBitmap("Bitmaps/action/","MachineDog/","mechine\_dog\_run\_2R.bmp",PURPLE);

}

void MachineDog::OnShow(MapManage \*map){

if(NowAct>0){

frame\_monster[NowAct>wait][LR\_flag].SetBottomLeft(wx,wy,height);

frame\_monster[NowAct>wait][LR\_flag].OnShow();

frame\_monster[NowAct>wait][LR\_flag].OnMove();

}

}

bool MachineDog::OnMove(MapManage \*map,ChipDale \*\*player){

int MLength,HLength;

if(NowAct<0)return false;

FixXY(map);

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0){

if(Detect(player,&MLength,&HLength,-1,-1,true)!=NULL){

LR\_flag = MLength >= 0;

NowAct=1;//啟動

}

}

else if(NowAct==100){

MoveLeft(map);

MoveUp(map);

}

else if(NowAct==101){

MoveRight(map);

MoveUp(map);

}

else if(NowAct>=1){

if(NowAct <= wait){

NowAct++;

}

else if(NowAct >= wait+1){

if(!IsJump && NowAct >= wait+4){

LR\_flag = !LR\_flag;

NowAct = wait+1;

}

if(!IsJump && wy!=lastWy){

NowAct=wait+1;

lastWy = wy;

LR\_Space = 15;

}

if((LR\_flag==0&&!MoveLeft(map)||LR\_flag==1&&!MoveRight(map))&&!IsJump){

IsJump = true;

LR\_Space = 0;

Jump();

NowAct++;

}

}

FallingDown(map);

}

return true;

}

else if(NowAct>0)

NowAct=-1;//表示活起來後又離開螢幕死掉

return true;

}

bool MachineDog::KillMonster(int Direct){

if(Direct&4)

NowAct=100;

else

NowAct=101;

IsJump=false;

MoveSPEED = SPEED\*2;

NoCollision=true;

Health = 0;

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Cactus 實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CMovingBitmap Cactus::frame\_monster;

Cactus::Cactus(MapManage \*map,int SetOx,int SetOy){

wx = SetOx\*ONEOBJX - map->ReturnNowX();

wy = SetOy\*ONEOBJY - map->ReturnNowY();

width = frame\_monster.Width();

height = frame\_monster.Height();

NowAct = 0;

CanTrace=false;

}

void Cactus::Loading(){

frame\_monster.LoadBitmapA("Bitmaps/action/Cactus/Cactus.bmp",PURPLE);

}

bool Cactus::OnMove(MapManage \*map,ChipDale \*\*player){

if(NowAct<0)return false;

FixXY(map);

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0){

NowAct=1;//啟動

}

}

else if(NowAct>0)

NowAct=-1;//表示活起來後又離開螢幕死掉

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Electric & Wire 實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

//Electric 屬於 Wire 架框的實際怪物

CMovingBitmap Electric::frame\_monster[2];

Electric::Electric(MapManage \*map,int SetWx,int SetWy,int SetWireLength):ChangeSpeed(4),MaxMoveSpeed((int)(SPEED\*1.5)){

MoveLength=0;//移動距離 和 顯示時使用

NoCollision=true;

width = frame\_monster[0].Width();

height = frame\_monster[0].Height();

wx = SetWx;

wy = SetWy;

NowAct = 1;

WireLength = SetWireLength;

Health = 1;

LRflag = 1;//電流的流動方向

CanTrace=false;

}

void Electric::Loading(){

frame\_monster[0].LoadBitmapA("Bitmaps/action/Electric/Electric1.bmp",PURPLE);

frame\_monster[1].LoadBitmapA("Bitmaps/action/Electric/Electric2.bmp",PURPLE);

}

void Electric::OnShow(MapManage \*map){

if(NowAct>0){

frame\_monster[(MoveLength%(ChangeSpeed\*2))>=ChangeSpeed].SetTopLeft(wx,wy);

frame\_monster[(MoveLength%(ChangeSpeed\*2))>=ChangeSpeed].ShowBitmap();

}

}

bool Electric::OnMove(MapManage \*map,ChipDale \*\*player){

if(NowAct<0)return false;

if(WireLength == MoveLength){

LRflag = !LRflag;

MoveLength = 0;

}

if(WireLength-MoveLength >= MaxMoveSpeed) MoveSPEED = MaxMoveSpeed;

else MoveSPEED = WireLength-MoveLength;

MoveLength += MoveSPEED;

if(!LRflag) MoveLeft(map);

else MoveRight(map);

return true;

}

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

//Wire

Wire::Wire(MapManage \*map,int SetOx,int SetOy,int SetWireOLength){

NoCollision=true;

WireLength = SetWireOLength\*ONEOBJX;

wx = SetOx\*ONEOBJX - map->ReturnNowX();

wy = SetOy\*ONEOBJY - map->ReturnNowY();

NowAct = 0;

Health = 1;

CanTrace=false;

real\_monster = new Electric(map,wx,wy,WireLength);

}

void Wire::Loading(){

Electric::Loading();

}

void Wire::OnShow(MapManage \*map){

if(NowAct>0){

real\_monster->OnShow(map);

}

}

bool Wire::OnMove(MapManage \*map,ChipDale \*\*player){

if(NowAct<0)return false;

FixXY(map);

real\_monster->FixXY(map);

if(wx > -WinShowBuffer-WireLength && wx < MWIDTH+WireLength && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0){

NowAct=1;//啟動

}

if(NowAct==1){

real\_monster->OnMove(map,player);

}

return true;

}

else if(NowAct>0){

NowAct=-1;//表示活起來後又離開螢幕死掉

}

return true;

}

void Wire::CollisionChipDale(ChipDale \*player){

if(NowAct>=1&&NowAct<100){

if(player->ReturnInvincible())return;

if(real\_monster->ElectricCollision(player)){

player->GetHurt();

}

}

}

void Wire::FixMapMove(int fixX,int fixY)

{

wx -= fixX;

wy -= fixY;

real\_monster->FixMapMove(fixX,fixY);

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Mouse 實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CAnimation Mouse::frame\_monster[3][2];

Mouse::Mouse(MapManage \*map,int SetOx,int SetOy){

MoveSPEED = SPEED\*3/5;

NoCollision=false;

LR\_Space = 0;

width = frame\_monster[0][0].Width();

height = frame\_monster[0][0].Height();

wx = SetOx\*ONEOBJX - map->ReturnNowX();

wy = SetOy\*ONEOBJY - map->ReturnNowY();

UpSpeed = 0;

LR\_flag = 0;

NowAct = 0;

IsJump = false;

Health = 1;

}

void Mouse::Loading(){

frame\_monster[0][0].SetDelayCount(5);

frame\_monster[0][0].AddBitmap("Bitmaps/action/","Mouse/","mouse\_stand\_L.bmp",PURPLE);

frame\_monster[0][1].SetDelayCount(5);

frame\_monster[0][1].AddBitmap("Bitmaps/action/","Mouse/","mouse\_stand\_R.bmp",PURPLE);

frame\_monster[1][0].SetDelayCount(5);

frame\_monster[1][0].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_1L.bmp",PURPLE);

frame\_monster[1][0].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_2L.bmp",PURPLE);

frame\_monster[1][0].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_3L.bmp",PURPLE);

frame\_monster[1][0].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_4L.bmp",PURPLE);

frame\_monster[1][1].SetDelayCount(5);

frame\_monster[1][1].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_1R.bmp",PURPLE);

frame\_monster[1][1].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_2R.bmp",PURPLE);

frame\_monster[1][1].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_3R.bmp",PURPLE);

frame\_monster[1][1].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Run\_4R.bmp",PURPLE);

frame\_monster[2][0].SetDelayCount(5);

frame\_monster[2][0].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Jump\_L.bmp",PURPLE);

frame\_monster[2][1].SetDelayCount(5);

frame\_monster[2][1].AddBitmap("Bitmaps/action/","Mouse/","mouse\_Jump\_R.bmp",PURPLE);

}

void Mouse::OnShow(MapManage \*map){

int showAct;

if(NowAct>0){

if(!IsJump) showAct=0;

else showAct=2;

if(NowAct>=50) showAct=1;

frame\_monster[showAct][LR\_flag].SetBottomLeft(wx,wy,height);

frame\_monster[showAct][LR\_flag].OnShow();

frame\_monster[showAct][LR\_flag].OnMove();

}

}

bool Mouse::OnMove(MapManage \*map,ChipDale \*\*player){

int MLength,HLength;

if(NowAct<0)return false;

FixXY(map);

if(wx > -WinShowBuffer && wx < MWIDTH+WinShowBuffer && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0 && wx > 0 && wx < MWIDTH && wy > 0 && wy < MHEIGHT){

if(Detect(player,&MLength,&HLength,-1,-1,true)!=NULL){

LR\_flag = MLength >= 0;

NowAct=1;//啟動

}

NowAct=1;//啟動

}

else if(NowAct==100){

MoveLeft(map);

MoveUp(map);

}

else if(NowAct==101){

MoveRight(map);

MoveUp(map);

}

else if(NowAct>=1){

tracePlayer = Detect(player,&MLength,&HLength,-1,0);

if(tracePlayer!=NULL){

LR\_flag = MLength >= 0;

NowAct = 50;

}

if(NowAct <= wait){

NowAct++;

}

else if(NowAct >= wait+1 && NowAct < 50){

if(!IsJump){

if(NowAct >= wait+5\*4){

LR\_flag = !LR\_flag;

NowAct = wait+1;

}

if((NowAct-wait)%4==1){

IsJump = true;

JumpSPEED=30;

Jump();

}

NowAct++;

}

else{

if(LR\_flag==0){

if(!MoveLeft(map)){

LR\_flag = !LR\_flag;

NowAct = wait+2;

}

}

else if(LR\_flag==1){

if(!MoveRight(map)){

LR\_flag = !LR\_flag;

NowAct = wait+2;

}

}

}

}

else if(NowAct >= 50){

if((LR\_flag==0&&!MoveLeft(map)||LR\_flag==1&&!MoveRight(map)) && !IsJump){

IsJump = true;

JumpSPEED=30;

Jump();

if(NowAct==52){

LR\_flag = !LR\_flag;

NowAct = 50;

}

NowAct++;

}

if(Detect(player,&MLength,&HLength,-1,height)==NULL){

NowAct = wait+1;

//LR\_flag = !LR\_flag;

}

}

FallingDown(map);

}

return true;

}

else if(NowAct>0)

NowAct=-1;//表示活起來後又離開螢幕死掉

return true;

}

bool Mouse::KillMonster(int Direct){

if(Direct&4)

NowAct=100;

else

NowAct=101;

IsJump=false;

MoveSPEED = SPEED\*2;

NoCollision=true;

Health = 0;

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Wasp 實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

CAnimation Wasp::frame\_monster[2];

Wasp::Wasp(MapManage \*map,int SetOx,int SetOy){

NoCollision=true;

width = frame\_monster[0].Width();

height = frame\_monster[0].Height();

wx = SetOx\*ONEOBJX - map->ReturnNowX();

wy = SetOy\*ONEOBJY - map->ReturnNowY();

NowAct = 0;

Health = 1;

MoveSPEED = 3;

}

void Wasp::Loading(){

frame\_monster[0].SetDelayCount(5);

frame\_monster[0].AddBitmap("Bitmaps/action/","Wasp/","Wasp\_1L.bmp",PURPLE);

frame\_monster[0].AddBitmap("Bitmaps/action/","Wasp/","Wasp\_2L.bmp",PURPLE);

frame\_monster[1].SetDelayCount(5);

frame\_monster[1].AddBitmap("Bitmaps/action/","Wasp/","Wasp\_1R.bmp",PURPLE);

frame\_monster[1].AddBitmap("Bitmaps/action/","Wasp/","Wasp\_2R.bmp",PURPLE);

}

void Wasp::OnShow(MapManage \*map){

if(NowAct>0){

frame\_monster[LR\_flag].SetBottomLeft(wx,wy,height);

frame\_monster[LR\_flag].OnShow();

frame\_monster[LR\_flag].OnMove();

}

}

bool Wasp::OnMove(MapManage \*map,ChipDale \*\*player){

int MLength,HLength;

if(NowAct<0)return false;

FixXY(map);

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0){

if(Detect(player,&MLength,&HLength,-1,-1,true)!=NULL){

LR\_flag = MLength >= 0;

NowAct=1;//啟動

}

}

else if(NowAct==100){

MoveLeft(map);

MoveUp(map);

}

else if(NowAct==101){

MoveRight(map);

MoveUp(map);

}

else if(NowAct>=1){

if(NowAct <= wait){

NowAct++;

}

else if(NowAct >= wait+1){

if(NowAct == wait+1){

MoveDown(map);

if(Detect(player,&MLength,&HLength,-1,height/2)){

NowAct++;

}

}

else{

MoveSPEED = 1;

MoveDown(map);

MoveSPEED = SPEED;

if(LR\_flag==0) MoveLeft(map);

if(LR\_flag==1) MoveRight(map);

}

}

}

return true;

}

else if(NowAct>0)

NowAct=-1;//表示活起來後又離開螢幕死掉

return true;

}

bool Wasp::KillMonster(int Direct){

if(Direct&4)

NowAct=100;

else

NowAct=101;

IsJump=false;

MoveSPEED = SPEED\*2;

NoCollision=true;

Health = 0;

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//bullet 實作 怪物所有用的子彈 \*/

// 需要另外做onshow \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

bullet::bullet(){

NoCollision=true;

NowAct = 1;

Health = 1;

LRflag = 1;

CanTrace=false;

}

bool bullet::OnMove(MapManage \*map){

if(NowAct<0)return false;

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0){

NowAct=1;//啟動

}

if(NowAct==1){

MoveSPEED = abs(MoveLR);

if(MoveLR<0) MoveLeft(map);

if(MoveLR>0) MoveRight(map);

MoveSPEED = abs(MoveUD);

if(MoveUD<0) MoveUp(map);

if(MoveUD>0) MoveDown(map);

}

return true;

}

else if(NowAct>0){

NowAct=-1;//表示活起來後又離開螢幕死掉

}

return true;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Centipede & Centipedelimbs 實作 \*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

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

//Lighting

CMovingBitmap Lighting::frame\_monster;

Lighting::Lighting(MapManage \*map,int SetWx,int SetWy,int LR,int UD){

wx = SetWx;

wy = SetWy;

MoveLR=LR;

MoveUD=UD;

}

void Lighting::Loading(){

frame\_monster.LoadBitmapA("Bitmaps/action/Centipede/Lightning.bmp",PURPLE);

}

void Lighting::OnShow(MapManage \*map){

if(NowAct>0){

frame\_monster.SetTopLeft(wx,wy);

frame\_monster.ShowBitmap();

}

}

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

//Centipedelimbs

CAnimation Centipedelimbs::frame\_monster[3];

Centipedelimbs::Centipedelimbs(MapManage \*map,int SetWx,int SetWy,int SetSelect):ReleaseBullTime(15),TotalBullTime(50),BulletMaxSpeed(10),WRandBullet(100){

NoCollision=true;

wx = SetWx;

wy = SetWy;

NowAct = 0;

Health = 1;

CanTrace=false;

Select = SetSelect;

width = frame\_monster[Select].Width();

height = frame\_monster[Select].Height();

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

bullets[i] = NULL;

}

Centipedelimbs::~Centipedelimbs(){

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

if(bullets[i]!=NULL)

delete(bullets[i]);

}

void Centipedelimbs::Loading(){

frame\_monster[0].SetDelayCount(2);

frame\_monster[0].AddBitmap("Bitmaps/action/","Centipede/","head.bmp",PURPLE);

frame\_monster[0].AddBitmap("Bitmaps/action/","Centipede/","head\_die.bmp",PURPLE);

frame\_monster[1].SetDelayCount(1);

frame\_monster[1].AddBitmap("Bitmaps/action/","Centipede/","L\_Fist.bmp",PURPLE);

frame\_monster[1].AddBitmap("Bitmaps/action/","Centipede/","L\_Cloth.bmp",PURPLE);

frame\_monster[2].SetDelayCount(1);

frame\_monster[2].AddBitmap("Bitmaps/action/","Centipede/","R\_Fist.bmp",PURPLE);

frame\_monster[2].AddBitmap("Bitmaps/action/","Centipede/","R\_Cloth.bmp",PURPLE);

}

void Centipedelimbs::OnShow(MapManage \*map,int countFlicker){

if(NowAct<0)return;

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

switch(Select){

case 0:

if(countFlicker==0)

frame\_monster[0].Reset();

frame\_monster[0].SetBottomLeft(wx,wy,height);

frame\_monster[0].OnShow();

frame\_monster[0].OnMove();

break;

case 1:case 2:

if(TotalBullTime-NowAct<=ReleaseBullTime) frame\_monster[Select].OnMoveToNum(1);

else frame\_monster[Select].OnMoveToNum(0);

frame\_monster[Select].SetBottomLeft(wx,wy,height);

frame\_monster[Select].OnShow();

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

if(bullets[i]!=NULL)

bullets[i]->OnShow(map);

}

break;

}

}

}

bool Centipedelimbs::OnMove(MapManage \*map,ChipDale \*\*player){

int WLength,HLength,UD,LR;

if(NowAct<0)return false;

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

if(bullets[i]!=NULL)

bullets[i]->FixXY(map);

}

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0){

NowAct=1;//啟動

}

if(NowAct>=1){

if(Select>=1){//Select 1 & 2 左手跟右手 , 0 是頭

if(TotalBullTime-NowAct==ReleaseBullTime){

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

if(bullets[i]!=NULL){

if(bullets[i]->ReturnNowAct()<0){

delete(bullets[i]);

bullets[i]=NULL;

}

continue;

}

else{

Detect(player,&WLength,&HLength,-1,-1,true);

WLength += rand()%(WRandBullet\*2)-WRandBullet;

if(WLength==0&&HLength==0) continue;

if(abs(WLength)>abs(HLength)){

LR = WLength>0 ? BulletMaxSpeed:-BulletMaxSpeed;

UD = HLength/(WLength/LR);

}

else{

UD = HLength>0 ? BulletMaxSpeed:-BulletMaxSpeed;

LR = WLength/(HLength/UD);

}

}

if(Select==1){

bullets[i] = new Lighting(map,wx+36,wy+32,LR,UD);

break;

}

else if(Select==2){

bullets[i] = new Lighting(map,wx+10,wy+23,LR,UD);

break;

}

}

}

NowAct++;

if(NowAct>50)NowAct=1;

}

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

if(bullets[i]!=NULL)

bullets[i]->OnMove(map);

}

}

return true;

}

else if(NowAct>0){

NowAct=-1;//表示活起來後又離開螢幕死掉

}

return true;

}

int Centipedelimbs::CentipedelimbsCollision(ChipDale \*player){

if(player->IfCollision(wx,wy,width,height))

return true;

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

if(bullets[i]!=NULL && bullets[i]->Collision(player))

return true;

}

return false;

}

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

//Centipede

CAnimation Centipede::frame\_monster;

Centipede::Centipede(MapManage \*map,int SetOx,int SetOy){

NoCollision=true;

wx = SetOx\*ONEOBJX - map->ReturnNowX();

wy = SetOy\*ONEOBJY - map->ReturnNowY();

width = frame\_monster.Width();

height = frame\_monster.Height();

TRACE("Centipede!!! wx %d , wy %d\n",wx,wy);

NowAct = 0;

Health = 5;

CanTrace=false;

real\_monster[0] = new Centipedelimbs(map,wx+180,wy-32,0);

real\_monster[1] = new Centipedelimbs(map,wx-22,wy+52,1);

real\_monster[2] = new Centipedelimbs(map,wx+321,wy-24,2);

countFlicker=0;

}

Centipede::~Centipede(){

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

delete(real\_monster[i]);

}

void Centipede::Loading(){

frame\_monster.SetDelayCount(2);

frame\_monster.AddBitmap("Bitmaps/action/","Centipede/","Centipede\_body\_normal.bmp",PURPLE);

frame\_monster.AddBitmap("Bitmaps/action/","Centipede/","Centipede\_body\_die.bmp",PURPLE);

Lighting::Loading();

Centipedelimbs::Loading();

}

void Centipede::OnShow(MapManage \*map){

if(NowAct<0)return;

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

if(countFlicker==0)

frame\_monster.Reset();

else

countFlicker--;

frame\_monster.SetBottomLeft(wx,wy,height);

frame\_monster.OnShow();

frame\_monster.OnMove();

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

real\_monster[i]->OnShow(map,countFlicker);

}

}

bool Centipede::OnMove(MapManage \*map,ChipDale \*\*player){

if(NowAct<0)return false;

FixXY(map);

real\_monster[0]->FixXY(map);

real\_monster[1]->FixXY(map);

real\_monster[2]->FixXY(map);

if(wx > -WinShowBuffer && wx < MWIDTH && wy > -WinShowBuffer && wy < MHEIGHT){

if(NowAct==0){

NowAct=1;//啟動

}

if(NowAct==1){

for(int i=1;i<3;i++)//real\_monster[0] 基本上不會有OnMove

real\_monster[i]->OnMove(map,player);

}

return true;

}

else if(NowAct>0){

NowAct=-1;//表示活起來後又離開螢幕死掉

}

return true;

}

void Centipede::CollisionChipDale(ChipDale \*player){

if(NowAct>=1&&NowAct<100){

if(player->ReturnInvincible())return; if(real\_monster[1]->CentipedelimbsCollision(player)||real\_monster[2]->CentipedelimbsCollision(player)){

player->GetHurt();

}

}

}

void Centipede::FixMapMove(int fixX,int fixY)

{

wx -= fixX;

wy -= fixY;

real\_monster[0]->FixMapMove(fixX,fixY);

real\_monster[1]->FixMapMove(fixX,fixY);

real\_monster[2]->FixMapMove(fixX,fixY);

}

bool Centipede::KillMonster(int Direct){

if(countFlicker!=0) return false;

Health--;

if(Health>0){

countFlicker = 50;

return false;

}

else{

CGameStateRun::ToBonus();

NowAct=-1;

return true;

}

}