# 臺北科技大學
# 資訊工程系


## 物件導向程式設計實習-書面報告

組別：_____4_____

題目：_____Scary Fight：放火燒完聖粉_____

組員：____黃彥穎_____學號____105590026____

____陳哲葦_____學號____105590030____

指導老師：陳偉凱

# 目錄

# 壹、 簡介

## 一、 動機：

剛開始我們在決定要做什麼類型的遊戲時，花了不少時間在討論，也玩了很多小遊戲來尋找想法，最後便找到了這個小時候也很常玩的遊戲：紫色恐怖，因為以前蠻愛玩這種類型的遊戲，剛好能藉由這次程式設計的課程，設計一個屬於我們的紫色恐怖。決定了遊戲後，再來就要思考要在什麼樣的環境下，Windows 和網頁是跟原本的遊戲方式比較相近，不過我們想說既然都要做了，不如就換個方式做吧，所以最後我們選擇了 Android 來實做這款遊戲，算是一種新的遊戲體驗。

## 二、 目的：

透過一學期的製作遊戲，首要目標不外乎是完成一個像樣的遊戲，二來是讓我們更能夠學會如何團隊合作，不在是像以往的作業是各自做各自的，埋頭幹自己的，在一個團隊，更要學會如何討論並把自己的想法表達出來、並從中統整出完整的脈絡，其重要性並不亞於自身的程式能力，提早學習、並適應未來出社會的工作方式，也是相當重要。

## 三、 分工：

由於陳哲葦對於「紫色恐怖」這款遊戲較為熟悉，因此大部分的角色功能以及遊戲方式由他設計，再進行討論；黃彥穎則主要為圖形美化輔助以及期末報告。

黃彥穎：主角滑行、起始畫面換頁以及王關進場動畫。

陳哲葦：主角攻擊、移動、屬性判斷以及怪物行為、進關轉場……等。

Git commit 情況：

| Graph | Description | Date | Author | Commit |
|---|---|---|---|---|
| | master origin/master origin/HEAD Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 16 六月 2018 21:02 | a105590026 <t10 | 73ea1d3 |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 16 六月 2018 17:46 | a105590030 <t10 | 67d497E |
| | 掉落物OK | 16 六月 2018 17:46 | a105590030 <t10 | 32faf00 |
| | no idea | 16 六月 2018 20:55 | a105590026 <t10 | 835599d |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 15 六月 2018 21:02 | a105590026 <t10 | 62ec340 |
| | 期末報告v1 | 15 六月 2018 11:55 | 105590026 <105 | 31ddd9E |
| | 增加掉落物part1 | 15 六月 2018 11:47 | a105590030 <t10 | 9a9cbc7 |
| | 期末報告v2 | 15 六月 2018 11:55 | 105590026 <105 | 920f35f |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 15 六月 2018 8:59 | a105590030 <t10 | f3a7a10 |
| | 死亡圖 | 15 六月 2018 8:56 | a105590030 <t10 | c45507c |
| | 你要的結局，我給你 | 15 六月 2018 21:01 | a105590026 <t10 | 4535ecd |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 15 六月 2018 1:31 | a105590026 <t10 | 1d98055 |
| | 王關可能OK | 15 六月 2018 1:28 | a105590030 <t10 | 6058fa0 |
| | merge | 14 六月 2018 21:19 | a105590030 <t10 | ee9e091 |
| | 王死亡動畫 | 14 六月 2018 21:19 | a105590030 <t10 | 254eafa |
| | 105590026_6/15 | 15 六月 2018 1:31 | a105590026 <t10 | 066e834 |

| | | | | |
|---|---|---|---|---|
| | 105590026_6/15 | 15 六月 2018 1:31 | a105590026 <t10 | 066e834 |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 14 六月 2018 19:36 | a105590026 <t10 | 51650a5 |
| | 王關死亡prat1 | 14 六月 2018 17:15 | a105590030 <t10 | 1c68b79 |
| | 王關遠攻 | 14 六月 2018 15:52 | a105590030 <t10 | f17a697 |
| | 雜七雜八 | 14 六月 2018 19:35 | a105590026 <t10 | d29705f |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 14 六月 2018 13:54 | a105590026 <t10 | 93806fb |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 14 六月 2018 12:11 | a105590030 <t10 | 268f379 |
| | 遠距離part1 | 14 六月 2018 12:08 | a105590030 <t10 | 3b2d7aa |
| | 再傳一次 | 14 六月 2018 13:54 | a105590026 <t10 | e0b0f65 |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 12 六月 2018 23:29 | a105590026 <t10 | 10f6d85 |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 12 六月 2018 0:57 | a105590030 <t10 | c60d67b |
| | 王關 隨機物處理 | 12 六月 2018 0:57 | a105590030 <t10 | c439ab4 |
| | 死王+王血條-改 | 12 六月 2018 23:28 | a105590026 <t10 | 556a989 |
| | 補王圖 | 11 六月 2018 23:31 | a105590026 <t10 | df9f2d2 |
| | merge | 11 六月 2018 0:26 | a105590030 <t10 | fbc178a |
| | 105590026_kingPhoto | 8 六月 2018 12:01 | a105590026 <105 | 9b18d5a |

| Graph | Description | Date | Author | Commit |
|---|---|---|---|---|
| | 105590026_kingPhoto | 8 六月 2018 12:01 | a105590026 <105 | 9b18d5a |
| | 王關動畫complete | 8 六月 2018 11:13 | a105590026 <105 | 01abf56 |
| | 0608_王進場一個move | 8 六月 2018 1:19 | a105590026 <t10 | 51ec773 |
| | 我不會 | 7 六月 2018 15:42 | 2-10\NTUTCSIE < | 6f1654d |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 7 六月 2018 1:40 | a105590026 <t10 | 45ab958 |
| | 105590026_改了一些圖 | 7 六月 2018 1:38 | a105590026 <t10 | e2cf4fc |
| | 角色和怪獸class | 11 六月 2018 0:25 | a105590030 <t10 | f302c3c |
| | Merge | 6 六月 2018 20:40 | a105590030 <t10 | f3ba49d |
| | ? | 1 六月 2018 12:20 | 105590030 <t105 | 4ae26a9 |
| | 不知道改了甚麼，好像沒吧 | 1 六月 2018 12:13 | 3-6\NTUTCSIE <t | 7c19a87 |
| | 衝擊波OK | 1 六月 2018 12:20 | 105590030 <t105 | c012959 |
| | 0601 | 1 六月 2018 2:09 | a105590026 <t10 | 54d48ba |
| | 沒事 | 6 六月 2018 20:39 | a105590030 <t10 | c0e334b |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 1 六月 2018 2:05 | a105590030 <t10 | fce48fd |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 31 五月 2018 22:43 | a105590026 <t10 | deb2975 |
| | 小改血條 | 31 五月 2018 1:48 | a105590026 <t10 | 9b58d2E |

| Graph | Description | Date | Author | Commit |
|---|---|---|---|---|
| | 小改血條 | 31 五月 2018 1:48 | a105590026 <t10 | 9b58d2E |
| | 砍破掉圖 | 1 六月 2018 2:05 | a105590030 <t10 | dca11f8 |
| | 衝擊波 | 1 六月 2018 2:05 | a105590030 <t10 | ca5c3cf |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 31 五月 2018 1:35 | a105590030 <t10 | 5f2acc7 |
| | Merge branch 'master' of https://ooplab.csie.ntut.edu.tw/git/oopcourse125 | 30 五月 2018 18:01 | a105590026 <t10 | 9840a8a |
| | 不知道這是甚麼 | 30 五月 2018 17:44 | a105590026 <t10 | 7e3ac6b |
| | 衝擊波 | 31 五月 2018 1:35 | a105590030 <t10 | 1082d38 |
| | pull | 29 五月 2018 20:05 | a105590030 <t10 | 4938ddc |
| | 105590026 5/25 hp v2 | 25 五月 2018 11:54 | 105590026 <chlb | 227ea87 |
| | 增加一些註解 和動作僵直改善 | 29 五月 2018 20:03 | a105590030 <t10 | c470c89 |
| | 推 | 25 五月 2018 9:36 | a105590030 <t10 | fe1a6e5 |
| | 5/25 105590026 monster_hp v1 | 25 五月 2018 1:15 | LAPTOP-FITAQ3C | c21352f |
| | 推 | 25 五月 2018 9:33 | a105590030 <t10 | 00e7549 |
| | Merge remote-tracking branch 'origin/master' | 24 五月 2018 22:56 | LAPTOP-FITAQ3C | 7343eaf |
| | 苦無攻擊 & 攻擊之間動畫修正 | 24 五月 2018 22:34 | a105590030 <t10 | d4f8d52 |
| | 怪獸 & 角色攻擊 PART2 | 18 五月 2018 9:41 | a105590030 <t10 | 2f5d71c |

| Graph | Description | Date | Author | Commit |
|---|---|---|---|---|
| | 怪獸 & 角色攻擊 PART2 | 18 五月 2018 9:41 | a105590030 <t10 | 2f5d71c |
| | 5/24 | 24 五月 2018 22:54 | LAPTOP-FITAQ3C | 2c103ba |
| | 只有滑行+幹大事 | 18 五月 2018 3:49 | LAPTOP-FITAQ3C | 253b7c6 |
| | 怪獸與角色攻擊PART 1 | 18 五月 2018 0:09 | a105590030 <t10 | 8e90db7 |
| | TEST | 11 五月 2018 12:28 | a105590030 <t10 | 24323e6 |
| | 1 | 4 五月 2018 9:44 | 105590026 <t105 | 11426a1 |
| | 怪獸移動 | 4 五月 2018 9:37 | a105590030 <t10 | 3abd531 |
| | 感動的第一次push | 26 四月 2018 21:59 | a105590026 <t10 | 6ee7d9e |

## 四、 指導老師：

陳偉凱 老師

## 五、 作者：

105590026 四資二 黃彥穎

105590030 四資二 陳哲葦

## 貳、 遊戲介紹

### 一、 遊戲說明：

「Scary Fight：放火燒完聖粉」這款遊戲是一個橫向卷軸遊戲，主要有三個關卡，必須將當前關卡的怪物殺完才可前往下一關，第三關是機器王關，擊敗機器王即可勝利，若無法活著擊敗機器王，則遊戲失敗。主角都市忍者必須使用武功擊敗侵襲沙鹿市夜晚寧靜的殭屍以及機器王，並且不能死亡。

在第一關以及第二關各有兩隻殭屍，必須殺完所有殭屍才能前往下一關卡。在第一關殺死所有殭屍時，會掉落補血藥水以及苦無，提供補血以及使用技能 S，而完成第二關後則會掉落衝擊波以及補血藥水，撿到衝擊波後即可使用雙擊 A 產生衝擊波。

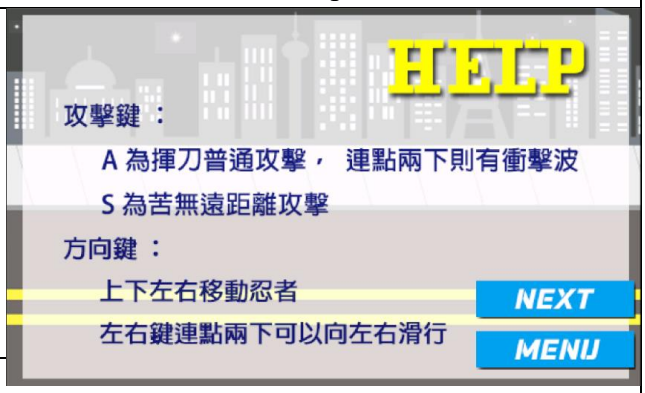
第三關則是機器王關，一進到關卡會先有一段機器王進場動畫，機器王的攻擊方式有兩種，方別為遠距離以及近距離：

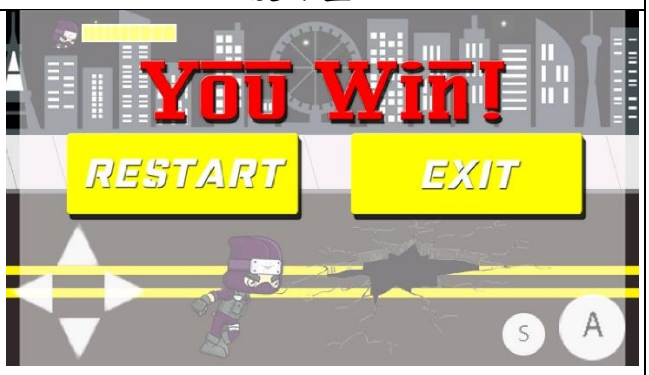
1. 遠距離攻擊：當都市忍者以及機器王平行時，機器王會發射衝擊波攻擊都市忍者。
2. 近距離攻擊：當都市忍者與機器王距離較近時，機器王則使用近距離揮刀攻擊。

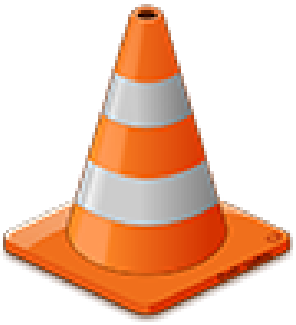
都市忍者技能：

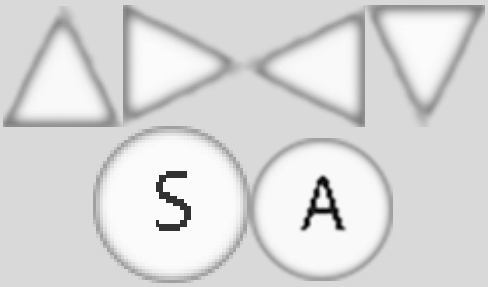
| | 技能使用方法 | 技能說明 |
|---|---|---|
| 1 | 單擊按鍵 A | 揮刀普通攻擊 |
| 2 | 雙擊按鍵 A | 揮刀並產生衝擊波，造成怪物傷害並使其後退 |
| 3 | 單擊按鍵 S | 發射苦無遠距離攻擊 |
| 4 | 方向鍵 | 移動忍者上下左右 |
| 5 | 連擊左、右方向鍵 | 使忍者向左、右滑行 |
| 6 | 方向鍵下+A+S | 密技：直接殺光此關所有怪物 |

## 二、 遊戲圖形：

| 遊戲起始畫面 | Help |
|---|---|
|  |  |
| Help-密技 | 角色 Attack |
|  |  |
| About | 角色滑行 |
|  |  |
| 角色雙擊 A 遠攻 | 角色 Smite |
|  |  |

| 角色被攻擊動畫 | 角色進關畫面 |
| --- | --- |
|  |  |
| 王關畫面 | 王關警告動畫 |
|  |  |
| 王近距離攻擊 | 王遠距離攻擊 |
|  |  |
| 失敗畫面 | 獲勝畫面 |
|  |  |

| 角色移動素材 | 遊戲 Icon |
| --- | --- |
|  |  |
| 角色攻擊素材 | 障礙物-垃圾桶 |
|  |  |
| 障礙物-街燈 | 障礙物-電話亭 |
|  |  |
| 障礙物-三角錐 | 障礙物-垃圾桶 |
|  |  |

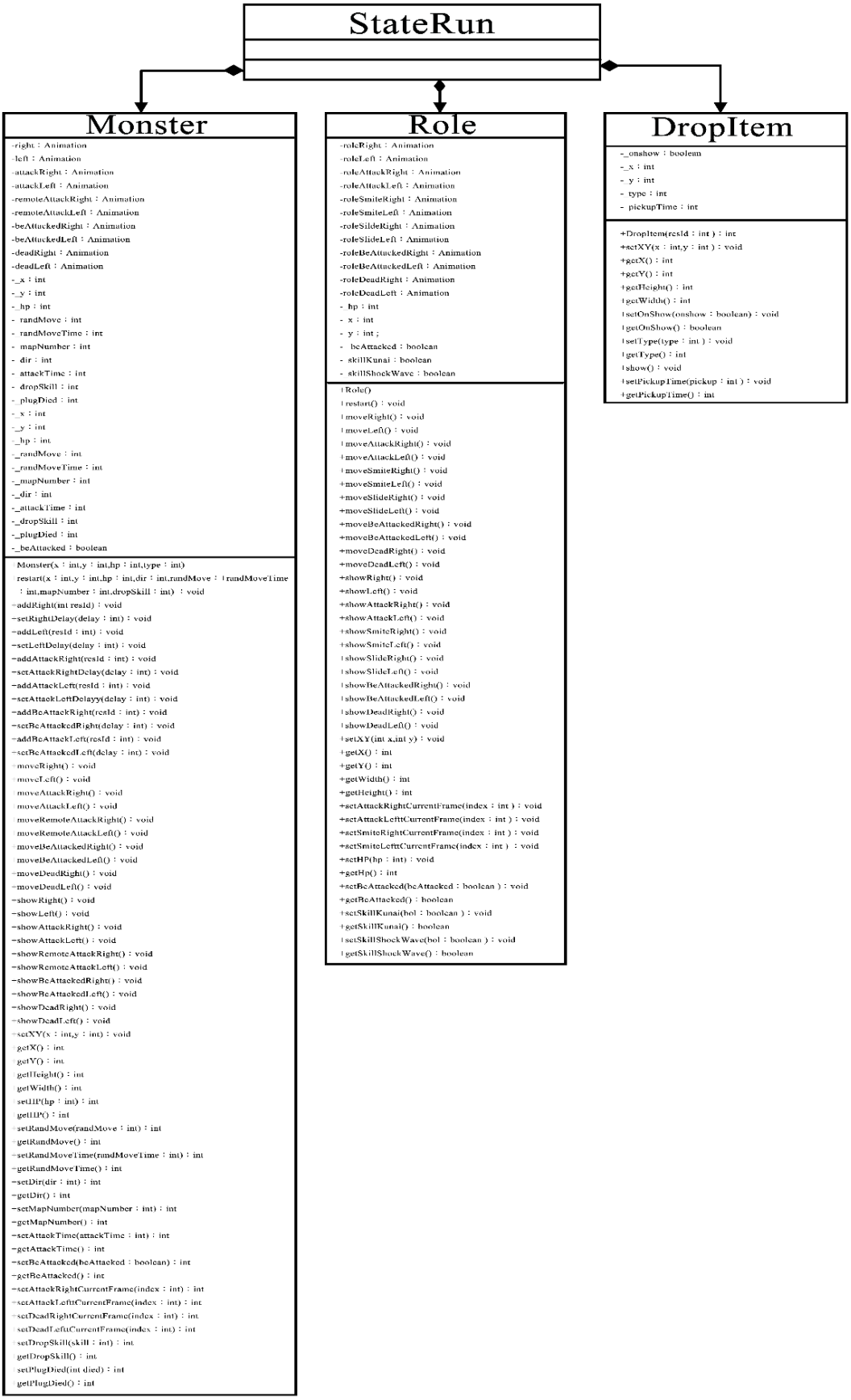
| 障礙物-跑車 | 障礙物-地板破裂 |
| --- | --- |
|  |  |
| 起始畫面按鍵 | 王關警告 |
|  |  |
| 遊戲中按鍵 | 血條 |
|  |  |
| 掉落物-道具 | 字卡提醒 |
|  | 尚未獲得苦無 請先獲得苦無道具<br>尚未獲得衝擊波 請先獲得衝擊波道 |

## 三、　遊戲音效：

| 音樂檔名 | 備註 |
| --- | --- |
| ntut.mp3 | 為本遊戲的背景音樂，使用「紫色恐怖」原始背景音樂。 |
| winmedio.mp3 | 獲勝時播放音樂。 |
| king_bgm.mp3 | 機器王關的專屬音樂。 |
| waveknife.mp3 | 角色揮刀時音效。 |

# 參、 程式設計

## 一、 程式架構：


StateRun
Monster
-right : Animation
-left : Animation
-attackRight : Animation
-attackLeft : Animation
-remoteAttackRight : Animation
-remoteAttackLeft : Animation
-beAttackedRight : Animation
-beAttackedLeft : Animation
-deadRight : Animation
-deadLeft : Animation
-_x : int
-_y : int
-_hp : int
-_randMove : int
-_randMoveTime : int
-_mapNumber : int
-_dir : int
-_attackTime : int
-_dropSkill : int
-_plugDied : int
-_x : int
-_y : int
-_hp : int
-_randMove : int
-_randMoveTime : int
-_mapNumber : int
-_dir : int
-_attackTime : int
-_dropSkill : int
-_plugDied : int
-_beAttacked : boolean
+Monster(x : int,y : int,hp : int,type : int)
+restart(x : int,y : int,hp : int,dir : int,randMove : int,randMoveTime : int,mapNumber : int,dropSkill : int) : void
-addRight(int resId) : void
-setRightDelay(delay : int) : void
-addLeft(resId : int) : void
-setLeftDelay(delay : int) : void
-addAttackRight(resId : int) : void
-setAttackRightDelay(delay : int) : void
-addAttackLeft(resId : int) : void
-setAttackLeftDelayy(delay : int) : void
-addBeAttackRight(resId : int) : void
-setBeAttackedRight(delay : int) : void
-addBeAttackLeft(resId : int) : void
-setBeAttackedLeft(delay : int) : void
+moveRight() : void
+moveLeft() : void
+moveAttackRight() : void
+moveAttackLeft() : void
+moveRemoteAttackRight() : void
+moveRemoteAttackLeft() : void
+moveBeAttackedRight() : void
+moveBeAttackedLeft() : void
-moveDeadRight() : void
-moveDeadLeft() : void
-showRight() : void
-showLeft() : void
-showAttackRight() : void
-showAttackLeft() : void
-showRemoteAttackRight() : void
-showRemoteAttackLeft() : void
-showBeAttackedRight() : void
-showBeAttackedLeft() : void
-showDeadRight() : void
-showDeadLeft() : void
-setXY(x : int,y : int) : void
+getX() : int
+getY() : int
+getHeight() : int
+getWidth() : int
-setHP(hp : int) : int
+getHP() : int
-setRandMove(randMove : int) : int
+getRandMove() : int
-setRandMoveTime(randMoveTime : int) : int
-getRandMoveTime() : int
-setDir(dir : int) : int
-getDir() : int
-setMapNumber(mapNumber : int) : int
-getMapNumber() : int
-setAttackTime(attackTime : int) : int
-getAttackTime() : int
-setBeAttacked(beAttacked : boolean) : int
-getBeAttacked() : int
-setAttackRightCurrentFrame(index : int) : int
-setAttackLeftCurrentFrame(index : int) : int
-setDeadRightCurrentFrame(index : int) : int
-setDeadLeftCurrentFrame(index : int) : int
-setDropSkill(skill : int) : int
-getDropSkill() : int
-setPlugDied(int died) : int
-getPlugDied() : int
Role
-roleRight : Animation
-roleLeft : Animation
-roleAttackRight : Animation
-roleAttackLeft : Animation
-roleSmiteRight : Animation
-roleSmiteLeft : Animation
-roleSildeRight : Animation
-roleSlideLeft : Animation
-roleBeAttackedRight : Animation
-roleBeAttackedLeft : Animation
-roleDeadRight : Animation
-roleDeadLeft : Animation
- hp : int
- x : int
- y : int ;
- beAttacked : boolean
- skillKunai : boolean
- skillShockWave : boolean
+Role()
+restart() : void
+moveRight() : void
+moveLeft() : void
+moveAttackRight() : void
+moveAttackLeft() : void
+moveSmiteRight() : void
+moveSmiteLeft() : void
+moveSlideRight() : void
+moveSlideLeft() : void
+moveBeAttackedRight() : void
+moveBeAttackedLeft() : void
+moveDeadRight() : void
+moveDeadLeft() : void
+showRight() : void
+showLeft() : void
+showAttackRight() : void
+showAttackLeft() : void
+showSmiteRight() : void
+showSmiteLeft() : void
+showSlideRight() : void
+showSlideLeft() : void
+showBeAttackedRight() : void
+showBeAttackedLeft() : void
+showDeadRight() : void
+showDeadLeft() : void
+setXY(int x,int y) : void
+getX() : int
+getY() : int
+getWidth() : int
+getHeight() : int
+setAttackRightCurrentFrame(index : int ) : void
+setAttackLefttCurrentFrame(index : int ) : void
+setSmiteRightCurrentFrame(index : int ) : void
+setSmiteLefttCurrentFrame(index : int ) : void
+setHP(hp : int) : void
+getHp() : int
+setBeAttacked(beAttacked : boolean ) : void
+getBeAttacked() : boolean
+setSkillKunai(bol : boolean ) : void
+getSkillKunai() : boolean
+setSkillShockWave(bol : boolean ) : void
+getSkillShockWave() : boolean
DropItem
-_onshow : boolean
-_x : int
- y : int
- type : int
- pickupTime : int
+DropItem(resId : int ) : int
+setXY(x : int,y : int ) : void
+getX() : int
+getY() : int
+getHeight() : int
+getWidth() : int
+setOnShow(onshow : boolean) : void
+getOnShow() : boolean
+setType(type : int ) : void
+getType() : int
+show() : void
+setPickupTime(pickup : int ) : void
+getPickupTime() : int

## 二、 程式類別：

| 類別名稱 | .java 檔行數 | 說明 |
|---|---|---|
| Monster | 398 | 所有怪物行為（移動、攻擊……等）以及怪物屬性（血量、方向判定） |
| Role | 239 | 所有角色行為（移動、滑行、攻擊）以及角色屬型（血量、方向判定） |
| DropItem | 40 | 判斷掉落物狀態 |
| 總行數 | 677 | |

## 三、 程式技術：

因為是角色扮演的遊戲，所以寫了很多副程式在處理物件與物件之間的互動判定，例如：障礙物無法通過、小怪掉落道具、以及角色與怪物之間的互動。障礙物的部分，為了寫成能夠通用的副程式，我們分為了高於角色以及低於角色兩個副程式，因為在 y 座標上會有些許變化。

多點觸控：原本程式就有寫但功能不完全，無法正常使用，我們是利用 List 的方式去把觸控的點記錄下來，然後在利用 for 迴圈將座標取出判斷是否需要使用。

# 肆、 結語

## 一、 問題與解決方法：

觸控連點功能：因為角色攻擊、移動時會使用到連點的功能，但是在一開始一直無法做出此功能，後來有上尋找的解決方法，找到了可以使用 Java 內建的函式，取得點取的系統時間，然後再和上次點擊放開的時間點相減，如果在設定時間內就是連點，最後才得以完成此功能。

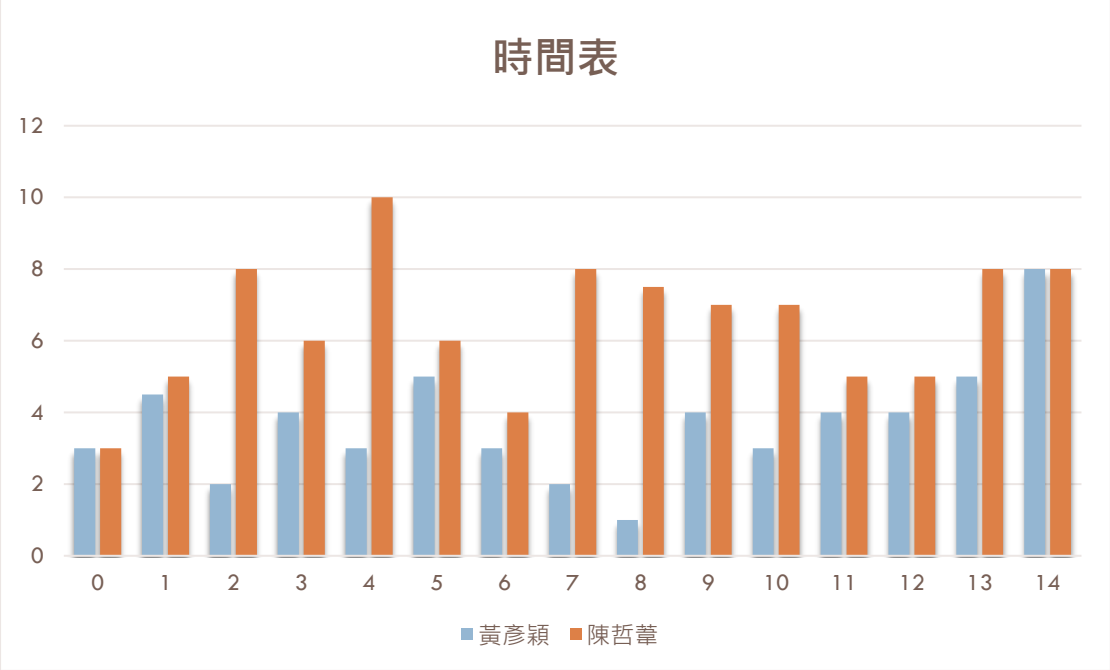
動畫與動畫之間的動作無法對齊：因為一個角色或怪獸都會有很多動作，像走路、攻擊等等， 一開始因為動畫無法正確銜接上而修改了很多次圖片，最後我們是把所有動作的”版面尺寸”都弄成一樣大。

多點觸控無法使用：原本看似程式裡面有寫多點觸控這部分的功能，但我們要使用時發現無法正常運作，所以我們就加了一個 List 和一個迴圈把觸控的點記錄下來，並透過迴圈偵測是否觸控到的點是需要的。

障礙物的判定：原本以為很簡單的判定，但實際寫起來座標其實蠻難抓的準的，所以後來我們是利用畫圖，直接做圖並確認上下左右的邊界。

2.5D 的顯示方式：因為 2.5D 會有 Z 軸前後顯示的問題，一開始沒有頭緒該怎麼解決，後來利用 Y 軸的判定方式，來當做 Z 軸前後的判斷。

## 二、 時間表（不含上課時間）：



| 週數 | 黃彥穎（Hours） | 陳哲葦（Hours） | 總共時間（Hours） |
|---|---|---|---|
| 0 | 3 | 3 | 6 |
| 1 | 4.5 | 5 | 9.5 |
| 2 | 2 | 8 | 10 |
| 3 | 4 | 6 | 10 |
| 4 | 3 | 10 | 13 |
| 5 | 5 | 6 | 11 |
| 6 | 3 | 4 | 7 |
| 7 | 2 | 8 | 10 |
| 8 | 1 | 7.5 | 8.5 |
| 9 | 4 | 7 | 11 |
| 10 | 3 | 7 | 10 |
| 11 | 4 | 5 | 9 |
| 12 | 4 | 5 | 9 |
| 13 | 5 | 8 | 13 |
| 14 | 8 | 8 | 16 |
| 總計 | 55.5 | 97.5 | 153 |

## 三、 貢獻比例：

黃彥穎：50%

陳哲葦：50%

合計：100%

## 四、 檢核表：

| | 項目 | 完成否 | 無法完成的原因 |
|---|---|---|---|
| 1 | 解決 Memory leak | ■已完成 □未完成 | |
| 2 | 自訂遊戲 Icon | ■已完成 □未完成 | |
| 3 | 全螢幕啟動 | ■已完成 □未完成 | |
| 4 | 修改 Help→About | ■已完成 □未完成 | |
| 5 | 初始畫面說明按鍵及滑鼠之用法與密技 | ■已完成 □未完成 | |
| 6 | 上傳 setup 檔 | ■已完成 □未完成 | |
| 7 | 報告字型、點數、對齊、行距、頁碼等格式正確 | ■已完成 □未完成 | |
| 8 | 報告封面、側邊格式正確 | ■已完成 □未完成 | |

## 五、 收穫：

黃彥穎：

經過這一個學期，雖然表面上看起來是要我們完成一個遊戲，但是最重要的是讓我們學習物件導向的概念，其實最難的不是撰寫程式的部分，反而是兩個人的互相搭配，這堂課不光是要我們學習程式，更要訓練我們如何團隊合作，在一個團隊

中一定有每個人各自比較擅長的部份，能夠一個團隊中每個人發揮自己所長，才能發揮出這個團隊的最大效益以及能力，這才是我在這堂課中學習到最多的部分，而這學期另外一個很重要的部分就是訓練我們自學的能力，這學期全部的程式都是我們自己摸索的，因為老師都已經寫好程式框架了，所以這次的自學跟其他比起來相對簡單許多，我們只要照著老師的框架寫，基本上不會有太大的問題。

陳哲葦：

經過這學期的實做，我學到了更多關於"物件的使用"，以及使用物件的重要性，剛開始因為沒有正確使用物件將需要用到的，像是角色、怪獸、物品的部分，都包成一個一個物件，因而造成在程式撰寫起來相當繁雜，且有人多重複的 code 是在執行類似的事情，最後包成物件後，才發現因此節省了很多多餘的 code，讓程式碼看起來比以前簡潔、易讀許多，也學到了在程式撰寫前，程式的規劃也是相當重要的。在程式撰寫時，難免會遇到許多問題，這時懂得利用網路上的資源就很重要，像這次遊戲是在 java 上寫的，一些語法跟以前在 c++上面，有一些不同的地方就要自己上網找資料，邊做邊學也學了蠻多的。除了在程式上的成長外，我更體會到像這種規模較以往大的程式，要規劃程式、還要做美術動畫、更要解決撰寫時產生的 Bug，單靠自己是無法做得很好的，所以團隊分工上是不可免的，不但能減輕本身的壓力，也可以透過討論理解為何這樣寫不行，加速遊戲的開發。

## 六、 心得：

黃彥穎：

這個學期其實物件導向程式設計實習算是作業比較重的一門課，但也因為這樣才能夠讓我們學習到更多的東西，雖然在過程中是很辛苦的，每個禮拜都有進度的壓力，但是這也讓我們主動地去自學程式，而除了在寫程式之外，還有一個最常遇到的就是團隊討論，兩個人討論總是會遇到意見分歧的時候，因此更需要透過不斷的溝通，才能得到兩個人都滿意的想法，也才能得到最好的結果！而在這學期中我們有很重要的三次 Demo，在我們以後可能也會有許多 Demo 的機會，這也讓我們學習如何在短時間內，展現出自己程式的所有優點，並且讓顧客感興趣，這是我們可以去練習、學習的，因此在這堂課中其實可以讓我們學習到許多不同的東西，而不只是在程式方面，真的是讓我收穫良多！

陳哲葦：

經過了一個學期的遊戲撰寫，透過物件導向設計的方式，著實讓程式撰寫起來更加易讀許多，也把上學期所學的東西實際應用在遊戲裡面，有點類似課程延伸的部分，雖然這次在撰寫上並沒有將物件的特性用得很好，主要是一開始程式規劃上並沒有做得很好，導致後來發現時還要花時間修改。我覺得這學期蠻充實的，一來是在程式方面的精進，二來我覺得團隊合作的方式，能讓我學習如何與人合作，這在未來出社會算是一項重要的課題，畢竟要把自己的想法轉化為語言說出來，並讓你的夥伴們都能理解，也並非易事，盡早在大學習慣這種分工方式，我覺得相當好。

## 七、 對於本課程的建議：

物件導向這門課，分為上下學期，我覺得這樣不錯，上學期學習物件導向的觀念，這學期進行實做的部分，雖然遊戲的框架是老師提供好了，但還是練習到許多關於物件的觀念及如何可以使用的比較好。撰寫遊戲時，大部分都是我們自己學習和討論，助教及老師是在每周五上課時，從中檢查進度並協助我們看我們哪邊有問題，我覺得這樣的教學方式很棒，跳脫於以往老師講，講完學生才做的上課方式，這門課更注重於學生做，做完有問題在與老師或助教們討論。

# 伍、 附錄

**StateRun.java**

```
package tw.edu.ntut.csie.game.state;
import android.util.Log;
import android.view.animation.AnimationUtils;
import android.widget.Toast;
import java.util.List;
import java.util.Map;
import tw.edu.ntut.csie.game.Game;
import tw.edu.ntut.csie.game.Pointer;
import tw.edu.ntut.csie.game.R;
import tw.edu.ntut.csie.game.core.Audio;
import tw.edu.ntut.csie.game.core.MovingBitmap;
import tw.edu.ntut.csie.game.engine.GameEngine;
import tw.edu.ntut.csie.game.extend.Animation;
import tw.edu.ntut.csie.game.extend.BitmapButton;
import tw.edu.ntut.csie.game.extend.Integer;
import static java.lang.Thread.*;
public class StateRun extends GameState {
    public static final int DEFAULT_SCORE_DIGITS = 4;
    private BitmapButton rightButton;  //右按鈕
    private BitmapButton leftButton;   //左按鈕
    private BitmapButton upButton;
    private BitmapButton downButton;
    private BitmapButton attackButton;
    private BitmapButton smiteButton;
    private BitmapButton restartButton;
    private BitmapButton exitButton;
    private MovingBitmap _background;  //道路背景 //手機顯示的背景寬為 640 高為 376
    private MovingBitmap _explosionLeft,_explosionRight;
    private MovingBitmap _kunaiRight,_kunaiLeft;
    private MovingBitmap _shockWaveRight,_shockWaveLeft;
    private MovingBitmap _bulletRight,_bulletLeft;
    private MovingBitmap[] _mhp,_fmhp,_kingHp,_hp;
    private MovingBitmap _kunaiText,_shockWaveText;
    private MovingBitmap _winPhoto,_losePhoto;
    //----補給品----
    private DropItem _dropKunai,_dropShockWave,_dropBlood;
    //----第一張地圖----
    private MovingBitmap _map1Trashcan; //垃圾桶
    private MovingBitmap _map1Telephone; //電話亭
    private MovingBitmap _map1TrafficCon;
    private Monster _map1MonsterMan1;
    private Monster _map1MonsterMan2;
    //----第二張地圖----
    private MovingBitmap _map2StreeLight;
    private MovingBitmap _map2RecycleBin;
    private MovingBitmap _map2Car;
    private Monster _map2Monster1;
    private Monster _map2Monster2;
    //----第三張地圖----
    private Monster _king;
    private MovingBitmap _broke;
    private Role role;
    private Animation nextMapGo; //進下一關的方向指示
    private Animation warning;
    //----障礙物被攻擊後消失動畫----
    private Animation _map1TrashCanAttacked;
    private Animation _map1TrafficConAttacked;
    private Animation _map1TelephoneAttacked;
    private Animation _map2StreeLightAttacked;
    private Animation _map2RecycleBinAttacked;
    private Animation _map2CarAttacked;
    private Integer _scores;
    private Integer _test;
    private boolean _grab;
    private boolean _grabLeft ,  _grabRight , _grabUp , _grabDown , _grabAttack , _grabSmite,_grabRestart,_grabExit;  //按鍵區
```

```
    private boolean
_obstacleRight,_obstacleLeft,_obstacleUp,_obstacleButton,_attackRightThing,_attackLeftThing,_attackRightMonster,_attackLeftMonster;
    private boolean _monsterObstacleR, _monsterObstacleL, _monsterObstacleU, _monsterObstacleB;
    private boolean _slideR, _slideL;
    private boolean _oneGrabAttack,_doubleGrabAttack,_detectDoubleGrabAttack;
    private boolean _kingShow,_kingMelee,_remoteAttackRight,_remoteAttackLeft;
    private boolean _skillKunai,_skillShockWave;
    private int  _bx,_by,_roleX,_roleY,_direction,_mapNumber,_backNumber,_attackTime,_simteTime,_kunaiFlyTime,_shockWaveTime;
    private int _map1MonsterMan1X,_map1MonsterMan1Y;
    private int _map1MonsterMan2X,_map1MonsterMan2Y;
    private int _map2MonsterWoman1X,_map2MonsterWoman1Y;
    private int _map2MonsterWoman2X,_map2MonsterWoman2Y;
    private int _detectLastGrab;
    private int showGO , reset ,_monsterBeAttacked, _roleBeAttacked, _monsterBeClear;
    private int _monsterAttackRole;
    private int  _diedShine,_kingDiedShine;   //障礙物消失的閃爍
    private int _kingX,_kingY;
    private int _kingTime,_kingHpShow,_bulletFlyTime;
    private int roleDead;
    private Audio _music , _winMedio,_kingBGM;
    private static long lastClickRightTime = 0;
    private static long lastClickLeftTime = 0;
    public StateRun(GameEngine engine) {
        super(engine);
    }
    @Override
    public void initialize(Map<String, Object> data) {
        rightButton = new BitmapButton(R.drawable.right_1,R.drawable.right_2,90,275);
        leftButton = new BitmapButton(R.drawable.left_1,R.drawable.left_2,-10,275);
        upButton =new BitmapButton(R.drawable.up_1,R.drawable.up_2,40,225);
        downButton =new BitmapButton(R.drawable.down_1,R.drawable.down_2,40,325);
        attackButton =new BitmapButton(R.drawable.a,R.drawable.a_2,570,300);
        smiteButton = new BitmapButton(R.drawable.s,R.drawable.s_2,510,320);
        restartButton = new BitmapButton(R.drawable.final_restart,R.drawable.final_restart_pressed,50,130);
        exitButton = new BitmapButton(R.drawable.final_exit,R.drawable.final_exit_pressed,350,130);
        _background = new MovingBitmap(R.drawable.background);
        _explosionLeft = new MovingBitmap(R.drawable.explosion_left);
        _explosionRight = new MovingBitmap(R.drawable.explosion_right);
        _kunaiRight  = new MovingBitmap(R.drawable.kunai);
        _kunaiLeft = new MovingBitmap(R.drawable.kunai_left);
        _shockWaveRight = new MovingBitmap(R.drawable.shockwave1);
        _shockWaveRight.resize(_shockWaveRight.getWidth(),_shockWaveRight.getHeight() + 30);
        _shockWaveLeft = new MovingBitmap(R.drawable.shockwaveleft1);
        _shockWaveLeft.resize(_shockWaveLeft.getWidth(),_shockWaveLeft.getHeight() + 30);
        _bulletRight = new MovingBitmap(R.drawable.bullet_000);
        _bulletLeft = new MovingBitmap(R.drawable.lbullet_000);
        _kunaiText = new MovingBitmap(R.drawable.card1,150,300);
        _shockWaveText = new MovingBitmap(R.drawable.card2_2,150,300);
        _winPhoto = new MovingBitmap(R.drawable.win);
        _losePhoto = new MovingBitmap(R.drawable.lose1);
        //-----補給品-----
        _dropBlood = new DropItem(R.drawable.blood);
        _dropBlood.setType(0);
        _dropKunai = new DropItem(R.drawable.kunai);
        _dropKunai.setType(1);
        _dropShockWave = new DropItem(R.drawable.shockwave1);
        _dropShockWave.setType(2);
        _mhp = new MovingBitmap[6];
        _mhp[0] = new MovingBitmap(R.drawable.mhp_00);
        _mhp[1] = new MovingBitmap(R.drawable.mhp_01);
        _mhp[2] = new MovingBitmap(R.drawable.mhp_02);
        _mhp[3] = new MovingBitmap(R.drawable.mhp_03);
        _mhp[4] = new MovingBitmap(R.drawable.mhp_04);
        _mhp[5] = new MovingBitmap(R.drawable.mhp_05);
        for(int i=0;i<6;i++){
            _mhp[i].setLocation(500,10);
        }
        _fmhp = new MovingBitmap[6];
        _fmhp[0] = new MovingBitmap(R.drawable.fmhp_00);
        _fmhp[1] = new MovingBitmap(R.drawable.fmhp_01);
```

```
_fmhp[2] = new MovingBitmap(R.drawable.fmhp_02);
_fmhp[3] = new MovingBitmap(R.drawable.fmhp_03);
_fmhp[4] = new MovingBitmap(R.drawable.fmhp_04);
_fmhp[5] = new MovingBitmap(R.drawable.fmhp_05);
for(int i=0;i<6;i++){
    _fmhp[i].setLocation(500,10);
}
_kingHp = new MovingBitmap[11];
_kingHp[0] = new MovingBitmap(R.drawable.king_hp000);
_kingHp[1] = new MovingBitmap(R.drawable.king_hp001);
_kingHp[2] = new MovingBitmap(R.drawable.king_hp002);
_kingHp[3] = new MovingBitmap(R.drawable.king_hp003);
_kingHp[4] = new MovingBitmap(R.drawable.king_hp004);
_kingHp[5] = new MovingBitmap(R.drawable.king_hp005);
_kingHp[6] = new MovingBitmap(R.drawable.king_hp006);
_kingHp[7] = new MovingBitmap(R.drawable.king_hp007);
_kingHp[8] = new MovingBitmap(R.drawable.king_hp008);
_kingHp[9] = new MovingBitmap(R.drawable.king_hp009);
_kingHp[10] = new MovingBitmap(R.drawable.king_hp010);
for(int i=0; i<11; i++){
    _kingHp[i].setLocation(250,10);
    _kingHp[i].resize(400,40);
}
_hp = new MovingBitmap[11];
_hp[0] = new MovingBitmap(R.drawable.hp_00);
_hp[1] = new MovingBitmap(R.drawable.hp_01);
_hp[2] = new MovingBitmap(R.drawable.hp_02);
_hp[3] = new MovingBitmap(R.drawable.hp_03);
_hp[4] = new MovingBitmap(R.drawable.hp_04);
_hp[5] = new MovingBitmap(R.drawable.hp_05);
_hp[6] = new MovingBitmap(R.drawable.hp_06);
_hp[7] = new MovingBitmap(R.drawable.hp_07);
_hp[8] = new MovingBitmap(R.drawable.hp_08);
_hp[9] = new MovingBitmap(R.drawable.hp_09);
_hp[10] = new MovingBitmap(R.drawable.hp_10);
for(int i=0;i<11;i++){
    _hp[i].setLocation(30,10);
}
//----第一張地圖----
_map1Trashcan = new MovingBitmap(R.drawable.trash_can,550,275);
_map1Trashcan.setAttribute("TrashCan",1);   //設定障礙物名子 & 血量
_map1Telephone = new MovingBitmap(R.drawable.telephone_booth,120,0);
_map1Telephone.setAttribute("Telephone",2);
_map1TrafficCon = new MovingBitmap(R.drawable.traffic_cone,350,150);
_map1TrafficCon.setAttribute("TrafficCon",1);
//-----第一張地圖怪獸----
_map1MonsterMan1 = new Monster(200,200,5,1);
_map1MonsterMan1.setRandMove(0);
_map1MonsterMan1.setRandMoveTime(0);
_map1MonsterMan1.setMapNumber(0);
_map1MonsterMan1.setDir(1);
_map1MonsterMan1.setDropSkill(1);
_map1MonsterMan2 = new Monster(500,100,5,1);
_map1MonsterMan2.setRandMove(0);
_map1MonsterMan2.setRandMoveTime(0);
_map1MonsterMan2.setMapNumber(0);
_map1MonsterMan2.setDir(1);
_map1MonsterMan2.setDropSkill(0);
//----第二張地圖----
_map2StreeLight = new MovingBitmap(R.drawable.street_light,1076,0);
_map2StreeLight.setAttribute("StreeLight",2);
_map2RecycleBin = new MovingBitmap(R.drawable.recyle_bin,700,0);
_map2RecycleBin.setAttribute("RecycleBin",2);
_map2Car = new MovingBitmap(R.drawable.car,870,290);
_map2Car.setAttribute("Car",3);
_map2Monster1 = new Monster(800,180,5,2);
_map2Monster1.setRandMove(0);
_map2Monster1.setRandMoveTime(0);
_map2Monster1.setMapNumber(1);
_map2Monster1.setDir(1);
```

```
_map2Monster1.setDropSkill(2);
_map2Monster2 = new Monster(1000,80,5,2);
_map2Monster2.setRandMove(0);
_map2Monster2.setRandMoveTime(0);
_map2Monster2.setMapNumber(1);
_map2Monster2.setDir(1);
_map2Monster2.setDropSkill(0);
//------第三張圖----
_broke = new MovingBitmap(R.drawable.broken,1450,185);//x1450
_broke.setHp(10);
warning = new Animation();
warning.addFrame(R.drawable.warning);
warning.addFrame(R.drawable.warning_2);
warning.setDelay(2);
warning.setLocation(0,0);
_king = new Monster(1500,-200,10,3); //x 1500
_king.setRandMove(0);
_king.setRandMoveTime(0);
_king.setMapNumber(2);
_king.setDir(1);
//----角色動畫區----
role = new Role();
role.setHP(10);
nextMapGo = new Animation();   //進關箭頭
nextMapGo.setLocation(550,180); //進關箭頭座標
nextMapGo.addFrame(R.drawable.go_1);
nextMapGo.addFrame(R.drawable.transparent);
nextMapGo.setDelay(5);
//----以下為障礙物消失動畫----
_map1TrashCanAttacked = new Animation();
_map1TrashCanAttacked.addFrame(R.drawable.trash_can);
_map1TrashCanAttacked.addFrame(R.drawable.transparent);
_map1TrashCanAttacked.setDelay(2);
_map1TrafficConAttacked = new Animation();
_map1TrafficConAttacked.addFrame(R.drawable.traffic_cone);
_map1TrafficConAttacked.addFrame(R.drawable.transparent);
_map1TrafficConAttacked.setDelay(2);
_map1TelephoneAttacked = new Animation();
_map1TelephoneAttacked.addFrame(R.drawable.telephone_booth);
_map1TelephoneAttacked.addFrame(R.drawable.transparent);
_map1TelephoneAttacked.setDelay(2);
_map2StreeLightAttacked = new Animation();
_map2StreeLightAttacked.addFrame(R.drawable.street_light);
_map2StreeLightAttacked.addFrame(R.drawable.transparent);
_map2StreeLightAttacked.setDelay(2);
_map2RecycleBinAttacked = new Animation();
_map2RecycleBinAttacked.addFrame(R.drawable.recyle_bin);
_map2RecycleBinAttacked.addFrame(R.drawable.transparent);
_map2RecycleBinAttacked.setDelay(2);
_map2CarAttacked = new Animation();
_map2CarAttacked.addFrame(R.drawable.car);
_map2CarAttacked.addFrame(R.drawable.transparent);
_map2CarAttacked.setDelay(2);
_bx = 0;
_by = 0;
_roleX = 0; //角色 X 座標
_roleY = 180;  //角色 Y 座標
_map1MonsterMan1X = _map1MonsterMan1.getX();
_map1MonsterMan1Y = _map1MonsterMan1.getY();
_map1MonsterMan2X = _map1MonsterMan2.getX();
_map1MonsterMan2Y = _map1MonsterMan2.getY();
_map2MonsterWoman1X = _map2Monster1.getX();
_map2MonsterWoman1Y = _map2Monster1.getY();
_map2MonsterWoman2X = _map2Monster2.getX();
_map2MonsterWoman2Y = _map2Monster2.getY();
_kingX = _king.getX();
_kingY = _king.getY();
_backNumber = 0;   //設定進關卡後 地圖捲動次數
_mapNumber = 0; //通過了幾個關卡
_direction = 1; //角色面向哪個方向 1 為右邊 0 為左邊
```

```
    _detectLastGrab = 1;   //初始先設定為 1 右 2 左 3 上 4 下
    showGO = 1;
    reset = 0;
    _monsterBeClear = 2;
    _attackTime = 0;   //使攻擊動畫完整砍完一次
    _simteTime = 0;
    _kunaiFlyTime = 0;
    _diedShine = 0;
    _monsterBeAttacked = 2;
    _roleBeAttacked = 0;
    _monsterAttackRole = 1; //1 為右邊 0 為左邊 先預設為 1
    _kingTime = -1;
    _kingHpShow = 0;
    _bulletFlyTime = 0;
    roleDead = 0;
    role.setXY(_roleX,_roleY);
    _scores = new Integer(DEFAULT_SCORE_DIGITS, _map1TrafficCon.getX(), 550, 10);
    _test = new Integer(DEFAULT_SCORE_DIGITS, _roleX, 350, 10);
    _music = new Audio(R.raw.ntut);
    _music.setRepeating(true);
    _music.play();
    _winMedio = new Audio(R.raw.winmedio);
    _winMedio.setRepeating(true);
    _kingBGM = new Audio(R.raw.king_bgm);
    _kingBGM.setRepeating(true);
    _grabLeft = false;
    _grabRight = false;
    _grabUp = false;
    _grabDown = false;
    _grabAttack = false;
    _grabSmite = false;
    _grabRestart = false;
    _grabExit = false;
    _obstacleRight = false;
    _obstacleLeft = false;
    _obstacleUp = false;
    _obstacleButton = false;
    _monsterObstacleR = false;
    _monsterObstacleL = false;
    _monsterObstacleU = false;
    _monsterObstacleB = false;
    _attackLeftThing = false;
    _attackRightThing = false;
    _attackRightMonster = false;
    _attackLeftMonster = false;
    _oneGrabAttack = false;
    _doubleGrabAttack = false;
    _detectDoubleGrabAttack = false;
    _kingShow = false;
    _kingMelee = false;
    _remoteAttackLeft = false;
    _remoteAttackRight = false;
    _skillKunai = false;
    _skillShockWave = false;
  }
  //------Restart 初始化------
  public void init(){
    _background.setLocation(0,0);
    role.restart();
    //-----補給品-----
    _dropBlood.setOnShow(false);
    _dropKunai.setOnShow(false);
    _dropShockWave.setOnShow(false);
    _map1Trashcan = new MovingBitmap(R.drawable.trash_can,550,275);
    _map1Trashcan.setAttribute("TrashCan",1);   //設定障礙物名子 & 血量
    _map1Telephone = new MovingBitmap(R.drawable.telephone_booth,120,0);
    _map1Telephone.setAttribute("Telephone",2);
    _map1TrafficCon = new MovingBitmap(R.drawable.traffic_cone,350,150);
    _map1TrafficCon.setAttribute("TrafficCon",1);
    _map1MonsterMan1.restart(200,200,5,1,0,0,0,1);
```

```
_map1MonsterMan2.restart(500,100,5,1,0,0,0,0);
_map2StreeLight = new MovingBitmap(R.drawable.street_light,1076,0);
_map2StreeLight.setAttribute("StreeLight",2);
_map2RecycleBin = new MovingBitmap(R.drawable.recyle_bin,700,0);
_map2RecycleBin.setAttribute("RecycleBin",2);
_map2Car = new MovingBitmap(R.drawable.car,870,290);
_map2Car.setAttribute("Car",3);
_map2Monster1.restart(800,180,5,1,0,0,1,2);
_map2Monster2.restart(1000,80,5,1,0,0,1,0);
_broke = new MovingBitmap(R.drawable.broken,1450,185);//x1450
_broke.setHp(10);
_king.restart(1500,-200,10,1,0,0,2,-1);
_bx = 0;
_by = 0;
_roleX = 0; //角色 X 座標
_roleY = 180;  //角色 Y 座標
_map1MonsterMan1X = _map1MonsterMan1.getX();
_map1MonsterMan1Y = _map1MonsterMan1.getY();
_map1MonsterMan2X = _map1MonsterMan2.getX();
_map1MonsterMan2Y = _map1MonsterMan2.getY();
_map2MonsterWoman1X = _map2Monster1.getX();
_map2MonsterWoman1Y = _map2Monster1.getY();
_map2MonsterWoman2X = _map2Monster2.getX();
_map2MonsterWoman2Y = _map2Monster2.getY();
_kingX = _king.getX();
_kingY = _king.getY();
_backNumber = 0;  //設定進關卡後 地圖捲動次數
_mapNumber = 0; //通過了幾個關卡
_direction = 1; //角色面向哪個方向 1 為右邊 0 為左邊
_detectLastGrab = 1;  //初始先設定為 1 右 2 左 3 上 4 下
showGO = 1;
reset = 0;
_monsterBeClear = 2;
_attackTime = 0;  //使攻擊動畫完整砍完一次
_simteTime = 0;
_kunaiFlyTime = 0;
_diedShine = 0;
_monsterBeAttacked = 2;
_roleBeAttacked = 0;
_monsterAttackRole = 1; //1 為右邊 0 為左邊 先預設為 1
_kingTime = -1;
_kingHpShow = 0;
_bulletFlyTime = 0;
roleDead = 0;
role.setXY(_roleX,_roleY);
_music.setRepeating(true);
_music.play();
_winMedio.setRepeating(true);
_winMedio.stop();
_kingBGM.stop();
_grabLeft = false;
_grabRight = false;
_grabUp = false;
_grabDown = false;
_grabAttack = false;
_grabSmite = false;
_grabRestart = false;
_grabExit = false;
_obstacleRight = false;
_obstacleLeft = false;
_obstacleUp = false;
_obstacleButton = false;
_monsterObstacleR = false;
_monsterObstacleL = false;
_monsterObstacleU = false;
_monsterObstacleB = false;
_attackLeftThing = false;
_attackRightThing = false;
_attackRightMonster = false;
_attackLeftMonster = false;
```

```
        _oneGrabAttack = false;
        _doubleGrabAttack = false;
        _detectDoubleGrabAttack = false;
        _kingShow = false;
        _kingMelee = false;
        _remoteAttackLeft = false;
        _remoteAttackRight = false;
        _skillKunai = false;
        _skillShockWave = false;
    }
    public void DetectObstacle_LessThanRole(MovingBitmap obstacle){
        if(_grabRight){    // -100 為角色寬度**主要判定 ，右邊界為角色+角色寬度，-50 為因為障礙物比角色矮要讓角色可以撞到要-50，下邊界判斷應為(上+下)除 2  -42 為減少下邊界
            if(_roleX > obstacle.getX() - 100 && _roleX < obstacle.getX() + obstacle.getWidth() - 15
                && _roleY > obstacle.getY() - 50 && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
_detectLastGrab == 1 && obstacle.getHp() > 0){
                _obstacleRight = true;
            }
        }
        else {
            _obstacleRight = false;
        }
        if(_grabLeft){     // -100 為角色寬度 ，右邊界為角色+角色寬度 ，-50 為因為障礙物比角色矮要讓角色可以撞到要-50，下邊界判斷應為(上+下)除 2  -42 為減少下邊界
            if(_roleX > obstacle.getX() - 100 && _roleX < obstacle.getX() + obstacle.getWidth()
                && _roleY > obstacle.getY() - 50 && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
_detectLastGrab == 2 && obstacle.getHp() > 0){
                _obstacleLeft = true;
            }
        }
        else {
            _obstacleLeft = false;
        }
        if(_grabUp){    // -70 為使垃圾桶左邊界往左一點 ，右邊界為角色+角色寬度 - 30 為減少右邊界避免角色有上面沒東西卻會頂到的問題，下邊界判斷應為(上+下)除 2 -26 為角色會頂到的邊界
            if(_roleX >obstacle.getX() - 70 && _roleX < obstacle.getX() + obstacle.getWidth() - 30
                && _roleY > obstacle.getY()  && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 && _detectLastGrab
== 3 && obstacle.getHp() > 0){
                _obstacleUp = true;
            }
        }
        else {
            _obstacleUp = false;
        }
        if(_grabDown){  // -70 為使垃圾桶左邊界往左一點 ，右邊界為角色+角色寬度 -30 為減少右邊界 避免角色有像在漂浮的問題， -80 為不要讓角色整個走進障礙物裡，下邊界判斷應為(上+下)除 2
            if(_roleX > obstacle.getX() - 70  && _roleX < obstacle.getX() + obstacle.getWidth() - 30
                && _roleY > obstacle.getY() - 80  && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 &&
_detectLastGrab == 4 && obstacle.getHp() > 0){
                _obstacleButton = true;
            }
        }
        else {
            _obstacleButton = false;
        }
    }
    public  void DetectObstacle(MovingBitmap obstacle){     //跟上面差不多 主要處理邊界問題，這邊處理比角色高的障礙物
        if(_grabRight){
            if(_roleX > obstacle.getX() - 100 && _roleX < obstacle.getX() + obstacle.getWidth()
                && _roleY > obstacle.getY() && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 23 && _detectLastGrab ==
1 && obstacle.getHp() > 0){
                _obstacleRight = true;
            }
        }
        else {
            _obstacleRight = false;
        }
        if(_grabLeft){
            if(_roleX > obstacle.getX() - 100 && _roleX < obstacle.getX() + obstacle.getWidth()
```

```
                && _roleY > obstacle.getY() && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 23 && _detectLastGrab ==
2 && obstacle.getHp() > 0){
            _obstacleLeft = true;
        }
    }
    else {
        _obstacleLeft = false;
    }
    if(_grabUp){
        if(_roleX >obstacle.getX() - 80 && _roleX < obstacle.getX() + obstacle.getWidth() - 30
                && _roleY > obstacle.getY()  && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 7 && _detectLastGrab ==
3 && obstacle.getHp() > 0){
            _obstacleUp = true;
        }
    }
    else {
        _obstacleUp = false;
    }
    if(_grabDown){
        if(_roleX > obstacle.getX() - 80  && _roleX < obstacle.getX() + obstacle.getWidth() - 30
                && _roleY > obstacle.getY() -80 && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 7 && _detectLastGrab
== 4 && obstacle.getHp() > 0){
            _obstacleButton = true;
        }
    }
    else {
        _obstacleButton = false;
    }
  }
  public void DetectObstacle_LessThanRole_broke(MovingBitmap obstacle){
    if(_grabRight){    // -100 為角色寬度**主要判定 ，右邊界為角色+角色寬度，-50 為因為障礙物比角色矮要讓角色可以撞到要-50，下
邊界判斷應為(上+下)除 2  -42 為減少下邊界
        if(_roleX > obstacle.getX() - 20 && _roleX < obstacle.getX() + obstacle.getWidth() - 50
                && _roleY > obstacle.getY() - 50 && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
_detectLastGrab == 1 && obstacle.getHp() > 0){
            _obstacleRight = true;
        }
    }
    else {
        _obstacleRight = false;
    }
    if(_grabLeft){     // -100 為角色寬度 ，右邊界為角色+角色寬度 ，-50 為因為障礙物比角色矮要讓角色可以撞到要-50，下邊界判斷應
為(上+下)除 2  -42 為減少下邊界
        if(_roleX > obstacle.getX() && _roleX < obstacle.getX() + obstacle.getWidth() - 30
                && _roleY > obstacle.getY() - 50 && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
_detectLastGrab == 2 && obstacle.getHp() > 0){
            _obstacleLeft = true;
        }
    }
    else {
        _obstacleLeft = false;
    }
    if(_grabUp){   // -70 為使垃圾桶左邊界往左一點 ，右邊界為角色+角色寬度 - 30 為減少右邊界避免角色有上面沒東西卻會頂到的問
題，下邊界判斷應為(上+下)除 2 -26 為角色會頂到的邊界
        if(_roleX >obstacle.getX() && _roleX < obstacle.getX() + obstacle.getWidth() - 50
                && _roleY > obstacle.getY()  && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 && _detectLastGrab
== 3 && obstacle.getHp() > 0){
            _obstacleUp = true;
        }
    }
    else {
        _obstacleUp = false;
    }
    if(_grabDown){  // -70 為使垃圾桶左邊界往左一點 ，右邊界為角色+角色寬度 -30 為減少右邊界 避免角色有像在漂浮的問題， -80 為
不要讓角色整個走進障礙物裡，下邊界判斷應為(上+下)除 2
        if(_roleX > obstacle.getX()   && _roleX < obstacle.getX() + obstacle.getWidth() - 50
                && _roleY > obstacle.getY() - 80  && _roleY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 &&
_detectLastGrab == 4 && obstacle.getHp() > 0){
            _obstacleButton = true;
        }
```

```
        }
        else {
            _obstacleButton = false;
        }
    }
    //----角色攻擊障礙物判定----//高度小於角色的障礙物
    public boolean DetectAttackThingLessThanRole(MovingBitmap thing){
        if(_roleX > thing.getX() - 130 && _roleX < thing.getX() + 20
                && _roleY > thing.getY() - 50 && _roleY < ((thing.getY() + thing.getHeight() + thing.getY()) / 2) - 42 /*&& _detectLastGrab == 1*/&&
_direction == 1 && thing.getHp() > 0){
            _attackRightThing = true;
            return true;
        }
        else{
            _attackRightThing = false;
        }
        if(_roleX > thing.getX() + 30 && _roleX < thing.getX() + thing.getWidth()
                && _roleY > thing.getY() - 50 && _roleY < ((thing.getY() + thing.getHeight() + thing.getY()) / 2) - 42 /*&& _detectLastGrab == 2*/&&
_direction == 0 && thing.getHp() > 0){
            _attackLeftThing = true;
            return true;
        }
        else{
            _attackLeftThing = false;
        }
        return false;
    }
    public boolean DetectAttackThing(MovingBitmap thing){
        if(_roleX > thing.getX() - 130 && _roleX < thing.getX() + 20
                && _roleY > thing.getY() - 50 && _roleY < ((thing.getY() + thing.getHeight() + thing.getY()) / 2) - 23 /*&& _detectLastGrab == 1*/&&
_direction == 1 && thing.getHp() > 0){
            _attackRightThing = true;
            return true;
        }
        else{
            _attackRightThing = false;
        }
        if(_roleX > thing.getX() + 30 && _roleX < thing.getX() + thing.getWidth()
                && _roleY > thing.getY() - 50 && _roleY < ((thing.getY() + thing.getHeight() + thing.getY()) / 2) - 23 /*&& _detectLastGrab == 2 */&&
_direction == 0 && thing.getHp() > 0){
            _attackLeftThing = true;
            return true;
        }
        else{
            _attackLeftThing = false;
        }
        return false;
    }
    //----角色攻擊怪獸----
    public boolean DetectAttackMonster(Monster _monster){
        if(_roleX > _monster.getX() - 130 && _roleX < _monster.getX() + 20
                && _roleY > _monster.getY() - 30 && _roleY < ((_monster.getY() + _monster.getHeight() + _monster.getY()) / 2) - 23
                && _direction == 1 && _monster.getHP() > 0 && _monster.getMapNumber() == _mapNumber){
            _attackRightMonster = true;
            _monster.setBeAttacked(true);
            _monsterBeAttacked = 10;    //設定怪物被攻擊的動畫持續時間
            return true;
        }
        else{
            _attackRightMonster = false;
            _monster.setBeAttacked(false);
        }
        if(_roleX > _monster.getX() + 30 && _roleX < _monster.getX() + _monster.getWidth()
                && _roleY > _monster.getY() - 30 && _roleY < ((_monster.getY() + _monster.getHeight() + _monster.getY()) / 2) - 23
                && _direction == 0 && _monster.getHP() > 0 && _monster.getMapNumber() == _mapNumber){
            _attackLeftMonster = true;
            _monster.setBeAttacked(true);
            _monsterBeAttacked = 10;    //設定怪物被攻擊的動畫持續時間
            return true;
        }
```

```
        else{
            _attackLeftMonster = false;
            _monster.setBeAttacked(false);
        }
        return false;
    }
    //----角色的遠程攻擊到怪物----
    public boolean DetectRemoteAttackMonster(Monster _monster, MovingBitmap _remoteAttackRight, MovingBitmap _remoteAttackLeft){
        if(_remoteAttackRight.getX() > _monster.getX() - 55 && _remoteAttackRight.getX() < _monster.getX() + 20
                && _remoteAttackRight.getY() + 20 > _monster.getY() && _remoteAttackRight.getY() < ((_monster.getY()
                + _monster.getHeight() + _monster.getY()) / 2) + 30 && _direction == 1 && _monster.getHP() > 0 && _monster.getMapNumber() ==
_mapNumber){
            _attackRightMonster = true;
            _monster.setBeAttacked(true);
            _monsterBeAttacked = 8;
            return true;
        }
        else{
            _attackRightMonster = false;
            _monster.setBeAttacked(false);
        }
        if(_remoteAttackLeft.getX() > _monster.getX() + 30 && _remoteAttackLeft.getX() < _monster.getX() + _monster.getWidth() - 30
                && _remoteAttackLeft.getY() + 20 > _monster.getY() && _remoteAttackLeft.getY() < ((_monster.getY() + _monster.getHeight()
                + _monster.getY()) / 2) + 30  && _direction == 0 && _monster.getHP() > 0 && _monster.getMapNumber() == _mapNumber){
            _attackLeftMonster = true;
            _monster.setBeAttacked(true);
            _monsterBeAttacked = 8;
            return true;
        }
        else{
            _attackLeftMonster = false;
            _monster.setBeAttacked(false);
        }
        return false;
    }
    //----怪獸攻擊角色----
    public boolean DetectAttackRole(Monster _monster,Role _role){
        if(_monster.getX() > _role.getX() - 80 && _monster.getX() < _role.getX() + 20
                && _monster.getY() > _role.getY() - 50 && _monster.getY() < ((_role.getY() + _role.getHeight() + _role.getY()) / 2) - 23 /*&&
_detectLastGrab == 1*/ && _role.getHp() > 0){
            _monster.setRandMoveTime(0);
            _monster.setDir(1);
            _role.setBeAttacked(true);
            _roleBeAttacked = 8;
            _monsterAttackRole = 1;
            _monster.setAttackTime(10);
            return true;
        }
        else{
            _role.setBeAttacked(false);
        }
        if(_monster.getX() > _role.getX() + 30 && _monster.getX() < _role.getX() + _role.getWidth() - 20
                && _monster.getY() > _role.getY() - 50 && _monster.getY() < ((_role.getY() + _role.getHeight() + _role.getY()) / 2) - 23 /*&&
_detectLastGrab == 2*/ && _role.getHp() > 0){
            _monster.setRandMoveTime(0);
            _monster.setDir(0);
            _role.setBeAttacked(true);
            _roleBeAttacked = 8;
            _monsterAttackRole = 0;
            _monster.setAttackTime(10);
            return true;
        }
        else{
            _role.setBeAttacked(false);
        }
        return false;
    }
    public boolean DetectAttackRole_king(Monster _monster,Role _role){
        if(_monster.getX() > _role.getX() - 150 && _monster.getX() < _role.getX() - 20
```

```
            && _monster.getY() > _role.getY() - 100 && _monster.getY() < ((_role.getY() + _role.getHeight() + _role.getY()) / 2) - 23 &&
_role.getHp() > 0){
        _monster.setRandMoveTime(0);
        _monster.setDir(1);
        _role.setBeAttacked(true);
        _roleBeAttacked = 8;
        _monsterAttackRole = 1;
        _monster.setAttackTime(10);
        return true;
    }
    else{
        _role.setBeAttacked(false);
    }
    if(_monster.getX() > _role.getX() && _monster.getX() < _role.getX() + _role.getWidth() - 20
            && _monster.getY() > _role.getY() - 100 && _monster.getY() < ((_role.getY() + _role.getHeight() + _role.getY()) / 2) - 23 &&
_role.getHp() > 0){
        _monster.setRandMoveTime(0);
        _monster.setDir(0);
        _role.setBeAttacked(true);
        _roleBeAttacked = 8;
        _monsterAttackRole = 0;
        _monster.setAttackTime(10);
        return true;
    }
    else{
        _role.setBeAttacked(false);
    }
    if(_monster.getX() > _role.getX() - 500 && _monster.getX() < _role.getX() - 170
            && _monster.getY() > _role.getY() - 100 && _monster.getY() < ((_role.getY() + _role.getHeight() + _role.getY()) / 2) - 50 &&
_role.getHp() > 0 && _bulletFlyTime == 0){
        _monster.setRandMoveTime(0);
        _monster.setAttackTime(12);
        _monster.setDir(1);
        _bulletFlyTime = 20;
        _bulletRight.setLocation(_monster.getX() + 130 ,_monster.getY() + 90);
        _remoteAttackRight = true;
        return true;
    }
    if(_monster.getX() > _role.getX() + 150 && _monster.getX() < _role.getX() + 500
            && _monster.getY() > _role.getY() - 100 && _monster.getY() < ((_role.getY() + _role.getHeight() + _role.getY()) / 2) - 50 &&
_role.getHp() > 0 && _bulletFlyTime == 0){
        _monster.setRandMoveTime(0);
        _monster.setAttackTime(12);
        _monster.setDir(0);
        _bulletFlyTime = 20;
        _bulletLeft.setLocation(_monster.getX() ,_monster.getY() + 90);
        _remoteAttackLeft = true;
        return true;
    }
    return false;
  }
  //----王遠距離攻擊砲彈判定----
  public boolean DetectRemoteAttackRole(Role _role, MovingBitmap _remoteAttackRight, MovingBitmap _remoteAttackLeft){
    if(_remoteAttackRight.getX() > _role.getX() - 55 && _remoteAttackRight.getX() < _role.getX() + 20
            && _remoteAttackRight.getY() + 20 > _role.getY() && _remoteAttackRight.getY() < ((_role.getY()
            + _role.getHeight() + _role.getY()) / 2) + 30 && _king.getDir() == 1 && _role.getHp() > 0){
        _role.setBeAttacked(true);
        _roleBeAttacked = 8;
        return true;
    }
    else{
        _role.setBeAttacked(false);
    }
    if(_remoteAttackLeft.getX() > _role.getX() + 30 && _remoteAttackLeft.getX() < _role.getX() + _role.getWidth() - 30
            && _remoteAttackLeft.getY() + 20 > _role.getY() && _remoteAttackLeft.getY() < ((_role.getY() + _role.getHeight()
            + _role.getY()) / 2) + 30  && _king.getDir() == 0 && _role.getHp() > 0){
        _role.setBeAttacked(true);
        _roleBeAttacked = 8;
        return true;
    }
```

```
        else{
            _role.setBeAttacked(false);
        }
        return false;
    }
    //----怪獸移動----
    public void MonsterMove(Monster _monster, int _monsterX , int _monsterY){
        if(_monster.getRandMoveTime() == 0) {
            _monster.setRandMove((int) (Math.random() * 4 + 1));   //製造 1~4 的亂數 1 為向右移動 2 為向左移動 3 為向上移動 4 為向下移動
            _monster.setRandMoveTime((int) (Math.random() * 6 + 50));  //製造 50~56 的亂數 讓角色移動的次數
        }
        //----第一張地圖 障礙物判定----
        monsterDetectObstacleLess(_map1Trashcan,_monsterX,_monsterY,_monster.getRandMove());
        monsterDetectObstacle(_map1Telephone,_monsterX,_monsterY,_monster.getRandMove());
        monsterDetectObstacleLess(_map1TrafficCon,_monsterX,_monsterY,_monster.getRandMove());
        if(_roleBeAttacked == 0 && _monster.getAttackTime() == 0 && _monster.getHP() > 0 && !_monster.getBeAttacked() &&
_monster.getMapNumber() < 2) {
            DetectAttackRole(_monster, role);
        }
        if(_roleBeAttacked == 0 && _monster.getAttackTime() == 0 && _monster.getHP() > 0 && !_monster.getBeAttacked() &&
_monster.getMapNumber() == 2) {
            DetectAttackRole_king(_monster, role);
        }
        //----第二張地圖障礙物判定----
        monsterDetectObstacle(_map2StreeLight,_monsterX,_monsterY,_monster.getRandMove());
        monsterDetectObstacle(_map2RecycleBin,_monsterX,_monsterY,_monster.getRandMove());
        monsterDetectObstacleLess(_map2Car,_monsterX,_monsterY,_monster.getRandMove());
        //----第三張圖障礙物判定
        monsterDetectObstacleLess_broke(_broke,_monsterX,_monsterY,_monster.getRandMove());
        if(_monster.getRandMoveTime() > 0 && _monster.getHP() > 0 && !_monster.getBeAttacked() && _monster.getAttackTime() == 0) {  //怪物
在被攻擊及攻擊人時不會移動 //*移動的次數未結束前不會變換行走方向
            if (_monster.getRandMove() == 1) {
                _monster.setDir(1);
                _monster.moveRight();
                if(_monsterX < 640 - _monster.getWidth() + 30 && !_monsterObstacleR) {
                    _monster.setXY(_monsterX += 5, _monsterY);
                }
                else{   //如果撞到障礙物或邊界 會變換方向
                    _monster.setRandMoveTime(1);     //設為 1 後 下面再-- 變為 0 會重新產生方向亂數
                }
                _monster.setRandMoveTime(_monster.getRandMoveTime() - 1);  //每次移動減一
            }
            else if (_monster.getRandMove() == 2) {
                _monster.setDir(0);
                _monster.moveLeft();
                if(_monsterX > 0 && !_monsterObstacleL) {
                    _monster.setXY(_monsterX -= 5, _monsterY);
                }
                else{
                    _monster.setRandMoveTime(1);
                }
                _monster.setRandMoveTime(_monster.getRandMoveTime() - 1);
            }
            else if (_monster.getRandMove() == 3) {
                if (_monster.getDir() == 1) {
                    _monster.moveRight();
                    if(_monsterY > _background.getY() + 15 && !_monsterObstacleU) {
                        _monster.setXY(_monsterX, _monsterY -= 2);
                    }
                    else{
                        _monster.setRandMoveTime(1);
                    }
                    _monster.setRandMoveTime(_monster.getRandMoveTime() - 1);
                }
                if (_monster.getDir() == 0) {
                    _monster.moveLeft();
                    if(_monsterY > _background.getY() + 15 && !_monsterObstacleU) {
                        _monster.setXY(_monsterX, _monsterY -= 2);
                    }
                    else{
```

```
                _monster.setRandMoveTime(1);
            }
            _monster.setRandMoveTime(_monster.getRandMoveTime() - 1);
        }
    }
    else if (_monster.getRandMove() == 4) {
        if (_monster.getDir() == 1) {
            _monster.moveRight();
            if(_monsterY < 376 - _monster.getHeight() + 5 && !_monsterObstacleB) {
                _monster.setXY(_monsterX, _monsterY += 2);
            }
            else{
                _monster.setRandMoveTime(1);
            }
            _monster.setRandMoveTime(_monster.getRandMoveTime() - 1);
        }
        if (_monster.getDir() == 0) {
            _monster.moveLeft();
            if(_monsterY < 376 -140 && !_monsterObstacleB) {
                _monster.setXY(_monsterX, _monsterY += 2);
            }
            else{
                _monster.setRandMoveTime(1);
            }
            _monster.setRandMoveTime(_monster.getRandMoveTime() - 1);
        }
    }
  }
  //----設定怪獸相關訊息回去----
  _monster.setXY(_monsterX,_monsterY);
}
//----怪獸障礙物判定---- //高度小於怪獸的障礙物
public void monsterDetectObstacle(MovingBitmap obstacle, int _monsterX, int _monsterY, int _randMove) {
  if (_randMove == 1){
    if (_monsterX > obstacle.getX() - 100 && _monsterX < obstacle.getX() + obstacle.getWidth()
            && _monsterY > obstacle.getY() && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 23 &&
obstacle.getHp() > 0) {
        _monsterObstacleR = true;
    }
  }
  else {
    _monsterObstacleR = false;
  }
  if(_randMove == 2) {
    if (_monsterX > obstacle.getX() - 100 && _monsterX < obstacle.getX() + obstacle.getWidth()
            && _monsterY > obstacle.getY() && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 23 &&
obstacle.getHp() > 0) {
        _monsterObstacleL = true;
    }
  }
  else {
    _monsterObstacleL = false;
  }
  if(_randMove == 3) {
    if (_monsterX > obstacle.getX() - 80 && _monsterX < obstacle.getX() + obstacle.getWidth() - 30
            && _monsterY > obstacle.getY() && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 7 &&
obstacle.getHp() > 0) {
        _monsterObstacleU = true;
    }
  }
  else {
    _monsterObstacleU = false;
  }
  if(_randMove == 4) {
    if (_monsterX > obstacle.getX() - 80 && _monsterX < obstacle.getX() + obstacle.getWidth() - 30
            && _monsterY > obstacle.getY() - 80 && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 7 &&
obstacle.getHp() > 0) {
        _monsterObstacleB = true;
    }
  }
```

```
        else {
            _monsterObstacleB = false;
        }
    }
    public void monsterDetectObstacleLess(MovingBitmap obstacle, int _monsterX, int _monsterY,int _randMove){
        if(_randMove == 1) {
            if (_monsterX > obstacle.getX() - 100 && _monsterX < obstacle.getX() + obstacle.getWidth() - 15
                    && _monsterY > obstacle.getY() - 50 && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
obstacle.getHp() > 0) {
                _monsterObstacleR = true;
            }
        }
        else {
            _monsterObstacleR = false;
        }
        if(_randMove == 2) {
            if (_monsterX > obstacle.getX() - 100 && _monsterX < obstacle.getX() + obstacle.getWidth()
                    && _monsterY > obstacle.getY() - 50 && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
obstacle.getHp() > 0) {
                _monsterObstacleL = true;
            }
        }
        else {
            _monsterObstacleL = false;
        }
        if(_randMove == 3) {
            if (_monsterX > obstacle.getX() - 70 && _monsterX < obstacle.getX() + obstacle.getWidth() - 30
                    && _monsterY > obstacle.getY() && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 &&
obstacle.getHp() > 0) {
                _monsterObstacleU = true;
            }
        }
        else {
            _monsterObstacleU = false;
        }
        if(_randMove == 4) {
            if (_monsterX > obstacle.getX() - 70 && _monsterX < obstacle.getX() + obstacle.getWidth() - 30
                    && _monsterY > obstacle.getY() - 80 && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 &&
obstacle.getHp() > 0) {
                _monsterObstacleB = true;
            }
        }
        else {
            _monsterObstacleB = false;
        }
    }
    public void monsterDetectObstacleLess_broke(MovingBitmap obstacle, int _monsterX, int _monsterY,int _randMove){
        if(_randMove == 1) {
            if (_monsterX > obstacle.getX() - 20 && _monsterX < obstacle.getX() + obstacle.getWidth() - 30
                    && _monsterY > obstacle.getY() - 150 && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
obstacle.getHp() > 0) {
                _monsterObstacleR = true;
            }
        }
        else {
            _monsterObstacleR = false;
        }
        if(_randMove == 2) {
            if (_monsterX > obstacle.getX() && _monsterX < obstacle.getX() + obstacle.getWidth() - 30
                    && _monsterY > obstacle.getY() - 150 && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 42 &&
obstacle.getHp() > 0) {
                _monsterObstacleL = true;
            }
        }
        else {
            _monsterObstacleL = false;
        }
        if(_randMove == 3) {
            if (_monsterX > obstacle.getX() && _monsterX < obstacle.getX() + obstacle.getWidth() - 50
```

```
                && _monsterY > obstacle.getY() && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 &&
obstacle.getHp() > 0) {
            _monsterObstacleU = true;
        }
    }
    else {
        _monsterObstacleU = false;
    }
    if(_randMove == 4) {
        if (_monsterX > obstacle.getX() && _monsterX < obstacle.getX() + obstacle.getWidth() - 50
                && _monsterY > obstacle.getY() - 100 && _monsterY < ((obstacle.getY() + obstacle.getHeight() + obstacle.getY()) / 2) - 26 &&
obstacle.getHp() > 0) {
            _monsterObstacleB = true;
        }
    }
    else {
        _monsterObstacleB = false;
    }
  }
  //----怪獸死亡物品掉落----
  public void dropItemDetected(Monster _monster){
    if(_monster.getHP() == -1){
        if(_monster.getDropSkill() == 0){
            _dropBlood.setOnShow(true);
            _dropBlood.setXY(_monster.getX() + 50,_monster.getY() + 70);
            _monster.setDropSkill(-1);
        }
        if(_monster.getDropSkill() == 1){
            _dropKunai.setOnShow(true);
            _dropKunai.setXY(_monster.getX() + 50,_monster.getY() + 70);
            _monster.setDropSkill(-1);
        }
        if(_monster.getDropSkill() == 2){
            _dropShockWave.setOnShow(true);
            _dropShockWave.setXY(_monster.getX() + 50,_monster.getY() + 70);
            _monster.setDropSkill(-1);
        }
    }
  }
  public void DetectGetDrop(DropItem item){
    boolean getItem = false;
    // -100 為角色寬度**主要判定 ，右邊界為角色+角色寬度，-50 為因為障礙物比角色矮要讓角色可以撞到要-50，下邊界判斷應為(上+
下)除 2  -42 為減少下邊界
    if(_roleX > item.getX() - 100 && _roleX < item.getX() + item.getWidth() - 15
            && _roleY > item.getY() - 50 && _roleY < ((item.getY() + item.getHeight() + item.getY()) / 2) - 42){
        getItem = true;
    }
    // -100 為角色寬度 ，右邊界為角色+角色寬度 ，-50 為因為障礙物比角色矮要讓角色可以撞到要-50，下邊界判斷應為(上+下)除 2  -42
為減少下邊界
    if(_roleX > item.getX() - 100 && _roleX < item.getX() + item.getWidth()
            && _roleY > item.getY() - 50 && _roleY < ((item.getY() + item.getHeight() + item.getY()) / 2) - 42 ){
        getItem = true;
    }
    // -70 為使垃圾桶左邊界往左一點 ，右邊界為角色+角色寬度 - 30 為減少右邊界避免角色有上面沒東西卻會頂到的問題，下邊界判斷
應為(上+下)除 2 -26 為角色會頂到的邊界
    if(_roleX >item.getX() - 70 && _roleX < item.getX() + item.getWidth() - 30
            && _roleY > item.getY()  && _roleY < ((item.getY() + item.getHeight() + item.getY()) / 2) - 26){
        getItem = true;
    }
    // -70 為使垃圾桶左邊界往左一點 ，右邊界為角色+角色寬度 -30 為減少右邊界 避免角色有像在漂浮的問題， -80 為不要讓角色整個
走進障礙物裡，下邊界判斷應為(上+下)除 2
    if(_roleX > item.getX() - 70  && _roleX < item.getX() + item.getWidth() - 30
            && _roleY > item.getY() - 80  && _roleY < ((item.getY() + item.getHeight() + item.getY()) / 2) - 26){
        getItem = true;
    }
    if(getItem){
        if(item.getType() == 0){
            role.setHP(role.getHp() + 5);
            if(role.getHp() > 10){
                role.setHP(10);
```

```
            }
            item.setOnShow(false);
            item.setPickupTime(12);
        }
        if(item.getType() == 1){
            role.setSkillKunai(true);
            item.setOnShow(false);
            item.setPickupTime(12);
        }
        if(item.getType() == 2){
            role.setSkillShockWave(true);
            item.setOnShow(false);
            item.setPickupTime(12);
        }
    }
}
@Override
public void move() {
    // ----設定每個動畫的位置(必續延續上次的位置) 再次設定避免位置跑掉
    role.setXY(_roleX,_roleY);
    _explosionRight.setLocation(_roleX + 5,_roleY - 10);   //攻擊到障礙物時會出現的效果
    _explosionLeft.setLocation(_roleX - 25,_roleY - 10);
    //----怪獸座標設定----
    _map1MonsterMan1.setXY(_map1MonsterMan1X,_map1MonsterMan1Y);
    _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
    _map2Monster1.setXY(_map2MonsterWoman1X,_map2MonsterWoman1Y);
    _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
    _king.setXY(_kingX,_kingY);
    //初始滑行
    _slideL = false;
    _slideR = false;
    //----第一張地圖 障礙物判定----
    DetectObstacle_LessThanRole(_map1Trashcan);
    DetectObstacle(_map1Telephone);
    DetectObstacle_LessThanRole(_map1TrafficCon);
    //----第二張地圖障礙物判定----
    DetectObstacle(_map2StreeLight);
    DetectObstacle(_map2RecycleBin);
    DetectObstacle_LessThanRole(_map2Car);
    //----第三張圖障礙物判定----
    DetectObstacle_LessThanRole_broke(_broke);
    //----以下為怪物移動----
    if(_mapNumber == 0 && _backNumber == 0) {
        MonsterMove(_map1MonsterMan1, _map1MonsterMan1X, _map1MonsterMan1Y);
        _map1MonsterMan1X = _map1MonsterMan1.getX();
        _map1MonsterMan1Y = _map1MonsterMan1.getY();
        MonsterMove(_map1MonsterMan2, _map1MonsterMan2X, _map1MonsterMan2Y);
        _map1MonsterMan2X = _map1MonsterMan2.getX();
        _map1MonsterMan2Y = _map1MonsterMan2.getY();
    }
    else if(_mapNumber == 1 && _backNumber == 0){
        MonsterMove(_map2Monster1,_map2MonsterWoman1X,_map2MonsterWoman1Y);
        _map2MonsterWoman1X = _map2Monster1.getX();
        _map2MonsterWoman1Y = _map2Monster1.getY();
        MonsterMove(_map2Monster2,_map2MonsterWoman2X,_map2MonsterWoman2Y);
        _map2MonsterWoman2X = _map2Monster2.getX();
        _map2MonsterWoman2Y = _map2Monster2.getY();
    }
    if(_mapNumber == 2 && _backNumber == 0 && _kingShow){
        MonsterMove(_king,_kingX,_kingY);
        _kingX = _king.getX();
        _kingY = _king.getY();
    }
    // ----以下為角色控制----
    if (_grabRight && _backNumber == 0 && _attackTime == 0 && _simteTime < 5 && !role.getBeAttacked() && _kingTime < 1 &&
role.getHp() > 0) {    //右鍵被按下，當攻擊鍵被按下 及 地圖在捲動時 無法移動
        long currentTime = System.currentTimeMillis(); //取得按下向右鍵的時間
        _direction = 1;    //設定方向為右
        _detectLastGrab = 1;
        role.moveRight();
```

```
        //當在 1000 ms 內連按兩下，則加速
        if ((currentTime - lastClickRightTime) <= 500) {
            role.moveSlideRight();
            if (_roleX > 150 && _background.getX() > -150 - _mapNumber * 600 && !_obstacleRight) {  //角色走過 150 這個 x 座標時，地圖會
移動，每次移動 6 最多 150
                _background.setLocation(_bx -= 6, _by);
                //----第一張地圖----
                _map1Trashcan.setLocation(_map1Trashcan.getX() - 6, _map1Trashcan.getY());
                _map1Telephone.setLocation(_map1Telephone.getX() - 6, _map1Telephone.getY());
                _map1TrafficCon.setLocation(_map1TrafficCon.getX() - 6, _map1TrafficCon.getY());
                _map1MonsterMan1X -= 6;
                _map1MonsterMan2X -= 6;
                //----第二張地圖----
                _map2StreeLight.setLocation(_map2StreeLight.getX() - 6, _map2StreeLight.getY());
                _map2RecycleBin.setLocation(_map2RecycleBin.getX() - 6, _map2RecycleBin.getY());
                _map2Car.setLocation(_map2Car.getX() - 6,_map2Car.getY());
                _map2MonsterWoman1X -= 6;
                _map2MonsterWoman2X -= 6;
                //----第三張地圖----
                _broke.setLocation(_broke.getX() - 6,_broke.getY());
                _kingX -= 6;
                //-----掉落物跟著移動-----
                _dropBlood.setXY(_dropBlood.getX() - 6,_dropBlood.getY());
                _dropKunai.setXY(_dropKunai.getX() - 6,_dropKunai.getY());
                _dropShockWave.setXY(_dropShockWave.getX() - 6,_dropShockWave.getY());
            }
            if (_roleX < 560 && !_obstacleRight) {  //螢幕最右邊的座標為 640 減掉角色寬度 80
                _slideR = true;
                role.setXY(_roleX += 15, _roleY);
            }
        }
        else {
            if (_roleX > 150 && _background.getX() > -150 - _mapNumber * 600 && !_obstacleRight) {  //角色走過 150 這個 x 座標時，地圖會
移動，每次移動 6 最多 150
                _background.setLocation(_bx -= 6, _by);
                //----第一張地圖----
                _map1Trashcan.setLocation(_map1Trashcan.getX() - 6, _map1Trashcan.getY());
                _map1Telephone.setLocation(_map1Telephone.getX() - 6, _map1Telephone.getY());
                _map1TrafficCon.setLocation(_map1TrafficCon.getX() - 6, _map1TrafficCon.getY());
                _map1MonsterMan1X -= 6;
                _map1MonsterMan2X -= 6;
                //----第二張地圖----
                _map2StreeLight.setLocation(_map2StreeLight.getX() - 6, _map2StreeLight.getY());
                _map2RecycleBin.setLocation(_map2RecycleBin.getX() - 6, _map2RecycleBin.getY());
                _map2Car.setLocation(_map2Car.getX() - 6,_map2Car.getY());
                _map2MonsterWoman1X -= 6;
                _map2MonsterWoman2X -= 6;
                //----第三張地圖----
                _broke.setLocation(_broke.getX() - 6,_broke.getY());
                _kingX -= 6;
                //-----掉落物跟著移動-----
                _dropBlood.setXY(_dropBlood.getX() - 6,_dropBlood.getY());
                _dropKunai.setXY(_dropKunai.getX() - 6,_dropKunai.getY());
                _dropShockWave.setXY(_dropShockWave.getX() - 6,_dropShockWave.getY());
            }
            if (_roleX < 560 && !_obstacleRight) {  //螢幕最右邊的座標為 640 減掉角色寬度 80
                role.setXY(_roleX += 12, _roleY);
            }
        }
    }
    if (_grabLeft && _backNumber == 0 && _attackTime == 0 && _simteTime < 5 && !role.getBeAttacked() && _kingTime < 1 &&
role.getHp() > 0) {
        long currentTime = System.currentTimeMillis();  //取得按下向左鍵的時間
        _direction = 0;
        _detectLastGrab = 2;
        role.moveLeft();
        //當在 1000 ms 內連按兩下，則加速
        if (currentTime - lastClickLeftTime <= 500) {
            role.moveSlideLeft();
            if (_roleX < 500 && _background.getX() < (0 - _mapNumber * 600) && !_obstacleLeft) {
```

```
                _background.setLocation(_bx += 6, _by);
                //----第一張地圖----
                _map1Trashcan.setLocation(_map1Trashcan.getX() + 6, _map1Trashcan.getY());
                _map1Telephone.setLocation(_map1Telephone.getX() + 6, _map1Telephone.getY());
                _map1TrafficCon.setLocation(_map1TrafficCon.getX() + 6, _map1TrafficCon.getY());
                _map1MonsterMan1X += 6;
                _map1MonsterMan2X += 6;
                //----第二張地圖----
                _map2StreeLight.setLocation(_map2StreeLight.getX() + 6, _map2StreeLight.getY());
                _map2RecycleBin.setLocation(_map2RecycleBin.getX() + 6, _map2RecycleBin.getY());
                _map2Car.setLocation(_map2Car.getX() + 6,_map2Car.getY());
                _map2MonsterWoman1X += 6;
                _map2MonsterWoman2X += 6;
                //----第三張地圖----
                _broke.setLocation(_broke.getX() + 6,_broke.getY());
                _kingX += 6;
                //-----掉落物跟著移動-----
                _dropBlood.setXY(_dropBlood.getX() + 6,_dropBlood.getY());
                _dropKunai.setXY(_dropKunai.getX() + 6,_dropKunai.getY());
                _dropShockWave.setXY(_dropShockWave.getX() + 6,_dropShockWave.getY());
            }
            if (_roleX > 0 && !_obstacleLeft) {    //螢幕最左邊的座標為 0
                _slideL = true;
                role.setXY(_roleX -= 15, _roleY);
            }
        }
        else {
            if (_roleX < 500 && _background.getX() < (0 - _mapNumber * 600) && !_obstacleLeft) {
                _background.setLocation(_bx += 6, _by);
                //----第一張地圖----
                _map1Trashcan.setLocation(_map1Trashcan.getX() + 6, _map1Trashcan.getY());
                _map1Telephone.setLocation(_map1Telephone.getX() + 6, _map1Telephone.getY());
                _map1TrafficCon.setLocation(_map1TrafficCon.getX() + 6, _map1TrafficCon.getY());
                _map1MonsterMan1X += 6;
                _map1MonsterMan2X += 6;
                //----第二張地圖----
                _map2StreeLight.setLocation(_map2StreeLight.getX() + 6, _map2StreeLight.getY());
                _map2RecycleBin.setLocation(_map2RecycleBin.getX() + 6, _map2RecycleBin.getY());
                _map2Car.setLocation(_map2Car.getX() + 6,_map2Car.getY());
                _map2MonsterWoman1X += 6;
                _map2MonsterWoman2X += 6;
                //----第三張地圖----
                _broke.setLocation(_broke.getX() + 6,_broke.getY());
                _kingX += 6;
                //-----掉落物跟著移動-----
                _dropBlood.setXY(_dropBlood.getX() + 6,_dropBlood.getY());
                _dropKunai.setXY(_dropKunai.getX() + 6,_dropKunai.getY());
                _dropShockWave.setXY(_dropShockWave.getX() + 6,_dropShockWave.getY());
            }
            if (_roleX > 0 && !_obstacleLeft) {    //螢幕最左邊的座標為 0
                role.setXY(_roleX -=12, _roleY);
            }
        }
    }
    if (_grabUp && _backNumber == 0 && _attackTime == 0 && _simteTime < 5 && !role.getBeAttacked() && _kingTime < 1 &&
role.getHp() > 0) {
        _detectLastGrab = 3;
        if (_direction == 1) {    //判斷原本面向哪個方向
            //roleRight.move();
            role.moveRight();
            if (_roleY > _background.getY() + 15 && !_obstacleUp) { //角色能移動的最上面邊界
                role.setXY(_roleX,_roleY -= 8);
            }
        } else {
            //roleLeft.move();
            role.moveLeft();
            if (_roleY > _background.getY() + 15 && !_obstacleUp) {
                role.setXY(_roleX,_roleY -= 8);
            }
        }
```

```
        }
        if(_grabDown && _backNumber == 0 && _attackTime == 0 && _simteTime < 5 && !role.getBeAttacked() && _kingTime < 1 &&
role.getHp() > 0){
            _detectLastGrab = 4;
            if(_direction == 1){
                role.moveRight();
                if(_roleY < 376 - 120 && !_obstacleButton) {   //角色能移動的最下面邊界 376 是最低減掉 120 為角色的高度
                    role.setXY(_roleX,_roleY += 8);
                }
            }
            else{
                role.moveLeft();
                if(_roleY < 376 - 120 && !_obstacleButton) { //螢幕高度減去角色高度
                    role.setXY(_roleX,_roleY += 8);
                }
            }
        }
        if(_grabAttack && _attackTime == 0 && _simteTime == 0 && _kunaiFlyTime == 0 && _backNumber == 0 && !role.getBeAttacked() &&
role.getHp() > 0){   //當攻擊按下 *攻擊動畫未結束前無法再按攻擊 *地圖捲動時也無法攻擊
            //----判斷前方是否有障礙物被打到---- *判斷分為比角色高的障礙物和比角色矮的障礙物
            if(DetectAttackThingLessThanRole(_map1Trashcan)){
                _map1Trashcan.setHp(_map1Trashcan.getHp() - 1);    //被打到後血量減 1
            }
            else if(DetectAttackThingLessThanRole(_map1TrafficCon)){
                _map1TrafficCon.setHp(_map1TrafficCon.getHp() - 1);
            }
            else if(DetectAttackThing(_map1Telephone)){
                _map1Telephone.setHp(_map1Telephone.getHp() - 1);
            }
            else if(DetectAttackThing(_map2StreeLight)){
                _map2StreeLight.setHp(_map2StreeLight.getHp() - 1);
            }
            else if(DetectAttackThing(_map2RecycleBin)){
                _map2RecycleBin.setHp(_map2RecycleBin.getHp() - 1);
            }
            else if(DetectAttackThingLessThanRole(_map2Car)){
                _map2Car.setHp(_map2Car.getHp() - 1);
            }
            //----偵測怪物是否被攻擊----
            if(DetectAttackMonster(_map1MonsterMan1)){
                _map1MonsterMan1.setHP(_map1MonsterMan1.getHP() - 1);
            }
            else if(DetectAttackMonster(_map1MonsterMan2)){
                _map1MonsterMan2.setHP(_map1MonsterMan2.getHP() - 1);
            }
            if(DetectAttackMonster(_map2Monster1)){
                _map2Monster1.setHP(_map2Monster1.getHP() - 1);
            }
            else if(DetectAttackMonster(_map2Monster2)){
                _map2Monster2.setHP(_map2Monster2.getHP() - 1);
            }
            else if(DetectAttackMonster(_king)){
                _king.setHP(_king.getHP() - 1);
            }
            //----攻擊按下時，會設定攻擊動畫時間、以及消失閃爍的時間，下面才會判斷如果有障礙物血量等於 0 會啟動閃爍
            if(_diedShine == 0) {
                _diedShine = 20;
            }
            _attackTime = 10;
        }
        if(_grabSmite &&_attackTime == 0 && _simteTime == 0 && _kunaiFlyTime == 0 && _backNumber == 0 && !role.getBeAttacked() &&
role.getSkillKunai() && role.getHp() > 0){   //*************
            if(_direction == 1){
                _kunaiRight.setLocation(_roleX + role.getWidth() - 50,_roleY + 70); //設定苦無的位置
            }
            if(_direction == 0){
                _kunaiLeft.setLocation(_roleX + 10,_roleY + 70);
            }
            //----Simte 被按下時，設定丟苦無的動畫時間、及消失閃爍時間、苦無飛行時間
            if(_diedShine == 0) {
```

```
            _diedShine = 20;
        }
        _simteTime = 10;
        _kunaiFlyTime = 10;
    }
    else if(_grabSmite &&_attackTime == 0 && _simteTime == 0 && _kunaiFlyTime == 0 && _backNumber == 0 && !role.getBeAttacked()
&& !role.getSkillKunai()){
        _kunaiText.setTextTime(10);
    }
    //----使攻擊動畫完整砍完----
    if(_attackTime > 0){
        _attackTime-- ;
        role.moveAttackRight();
        role.moveAttackLeft();
        if(_grabAttack && !_detectDoubleGrabAttack && !_oneGrabAttack && _shockWaveTime == 0){
            _oneGrabAttack = true;
            _detectDoubleGrabAttack = true;
        }
        if(_grabAttack && !_detectDoubleGrabAttack && _oneGrabAttack && role.getSkillShockWave()){
            _doubleGrabAttack = true;
            _detectDoubleGrabAttack = true;
        }
        else if(_grabAttack && !_detectDoubleGrabAttack && _oneGrabAttack && !role.getSkillShockWave()){
            _shockWaveText.setTextTime(10);
        }
        if(_attackTime == 0){  //攻擊動畫跑完時，將攻擊的動畫重新設定到第一個動畫圖
            role.setAttackRightCurrentFrame(1);
            role.setAttackLefttCurrentFrame(1);
            _oneGrabAttack = false;
            _doubleGrabAttack = false;
        }
    }
    //----連點 attack 產生衝擊波-----
    if(_doubleGrabAttack){
        _shockWaveRight.setLocation(_roleX + role.getWidth() - 50,_roleY);
        _shockWaveLeft.setLocation(_roleX,_roleY);
        _shockWaveTime = 8;
        _doubleGrabAttack =false;
        _oneGrabAttack = false;
    }
    if(_shockWaveTime > 0){
        _shockWaveTime--;
        if(_direction == 1){
            _shockWaveRight.setLocation(_shockWaveRight.getX() + 13 , _shockWaveRight.getY());
            _shockWaveRight.setLocation(_shockWaveRight.getX() + 13 , _shockWaveRight.getY());
            _shockWaveRight.setLocation(_shockWaveRight.getX() + 13 , _shockWaveRight.getY());
        }
        if(_direction == 0){
            _shockWaveLeft.setLocation(_shockWaveLeft.getX() - 13,_shockWaveLeft.getY());
            _shockWaveLeft.setLocation(_shockWaveLeft.getX() - 13,_shockWaveLeft.getY());
            _shockWaveLeft.setLocation(_shockWaveLeft.getX() - 13,_shockWaveLeft.getY());
        }
        //----判斷衝擊波是否打到怪物----
        if(DetectRemoteAttackMonster(_map1MonsterMan1,_shockWaveRight,_shockWaveLeft)){
            _map1MonsterMan1.setHP(_map1MonsterMan1.getHP() - 1);
            if(_direction == 1 && _map1MonsterMan1.getX() < 560){
                _map1MonsterMan1.setXY(_map1MonsterMan1X += 30,_map1MonsterMan1Y);
                _map1MonsterMan1.setXY(_map1MonsterMan1X += 30,_map1MonsterMan1Y);
            }
            if(_direction == 0 && _map1MonsterMan1.getX() > 0){
                _map1MonsterMan1.setXY(_map1MonsterMan1X -= 30,_map1MonsterMan1Y);
                _map1MonsterMan1.setXY(_map1MonsterMan1X -= 30,_map1MonsterMan1Y);
            }
            _shockWaveTime = 0;
        }
        if(DetectRemoteAttackMonster(_map1MonsterMan2,_shockWaveRight,_shockWaveLeft)){
            _map1MonsterMan2.setHP(_map1MonsterMan2.getHP() - 1);
            if(_direction == 1 && _map1MonsterMan2.getX() < 560){
                _map1MonsterMan2.setXY(_map1MonsterMan2X += 30,_map1MonsterMan2Y);
                _map1MonsterMan2.setXY(_map1MonsterMan2X += 30,_map1MonsterMan2Y);
```

```
            }
            if(_direction == 0 && _map1MonsterMan2.getX() > 0){
                _map1MonsterMan2.setXY(_map1MonsterMan2X -= 30,_map1MonsterMan2Y);
                _map1MonsterMan2.setXY(_map1MonsterMan2X -= 30,_map1MonsterMan2Y);
            }
            _shockWaveTime = 0;
        }
        if(DetectRemoteAttackMonster(_map2Monster1,_shockWaveRight,_shockWaveLeft)){
            _map2Monster1.setHP(_map2Monster1.getHP() - 1);
            if(_direction == 1 && _map2Monster1.getX() < 560){
                _map2Monster1.setXY(_map2MonsterWoman1X += 30,_map2MonsterWoman1Y);
                _map2Monster1.setXY(_map2MonsterWoman1X += 30,_map2MonsterWoman1Y);
            }
            if(_direction == 0 && _map2Monster1.getX() > 0){
                _map2Monster1.setXY(_map2MonsterWoman1X -= 30,_map2MonsterWoman1Y);
                _map2Monster1.setXY(_map2MonsterWoman1X -= 30,_map2MonsterWoman1Y);
            }
            _shockWaveTime = 0;
        }
        if(DetectRemoteAttackMonster(_map2Monster2,_shockWaveRight,_shockWaveLeft)){
            _map2Monster2.setHP(_map2Monster2.getHP() - 1);
            if(_direction == 1 && _map2Monster2.getX() < 560){
                _map2Monster2.setXY(_map2MonsterWoman2X += 30,_map2MonsterWoman2Y);
                _map2Monster2.setXY(_map2MonsterWoman2X += 30,_map2MonsterWoman2Y);
            }
            if(_direction == 0 && _map2Monster2.getX() > 0){
                _map2Monster2.setXY(_map2MonsterWoman2X -= 30,_map2MonsterWoman2Y);
                _map2Monster2.setXY(_map2MonsterWoman2X -= 30,_map2MonsterWoman2Y);
            }
            _shockWaveTime = 0;
        }
        if(DetectRemoteAttackMonster(_king,_shockWaveRight,_shockWaveLeft)){
            _king.setHP(_king.getHP() - 1);
            _shockWaveTime = 0;
        }
    }
    if(_simteTime > 0){
        _simteTime--;
        role.moveSmiteRight();
        role.moveSmiteLeft();
        if(_simteTime == 0){   //當丟苦無的動畫跑完時，將丟苦無動畫重新設定到第一個動畫圖
            role.setSmiteRightCurrentFrame(1);
            role.setSmiteLefttCurrentFrame(1);
        }
    }
    if(_kunaiFlyTime > 0){
        _kunaiFlyTime--;
        if(_direction == 1) {
            _kunaiRight.setLocation(_kunaiRight.getX() + 7,_kunaiRight.getY());
            _kunaiRight.setLocation(_kunaiRight.getX() + 7,_kunaiRight.getY());
            _kunaiRight.setLocation(_kunaiRight.getX() + 7,_kunaiRight.getY());
        }
        if(_direction == 0){
            _kunaiLeft.setLocation(_kunaiLeft.getX() - 7,_kunaiLeft.getY());
            _kunaiLeft.setLocation(_kunaiLeft.getX() - 7,_kunaiLeft.getY());
            _kunaiLeft.setLocation(_kunaiLeft.getX() - 7,_kunaiLeft.getY());
        }
        //----判斷苦無是否打到怪物----
        if(DetectRemoteAttackMonster(_map1MonsterMan1,_kunaiRight,_kunaiLeft)){
            _map1MonsterMan1.setHP(_map1MonsterMan1.getHP() - 1);
            _kunaiFlyTime = 0;
        }
        if(DetectRemoteAttackMonster(_map1MonsterMan2,_kunaiRight,_kunaiLeft)){
            _map1MonsterMan2.setHP(_map1MonsterMan2.getHP() - 1);
            _kunaiFlyTime = 0;
        }
        if(DetectRemoteAttackMonster(_map2Monster1,_kunaiRight,_kunaiLeft)){
            _map2Monster1.setHP(_map2Monster1.getHP() - 1);
            _kunaiFlyTime = 0;
        }
```

```
        if(DetectRemoteAttackMonster(_map2Monster2,_kunaiRight,_kunaiLeft)){
            _map2Monster2.setHP(_map2Monster2.getHP() - 1);
            _kunaiFlyTime = 0;
        }
        if(DetectRemoteAttackMonster(_king,_kunaiRight,_kunaiLeft)){
            _king.setHP(_king.getHP() - 1);
            _kunaiFlyTime = 0;
        }
    }
    //-----地圖一的怪獸一攻擊角色動畫處理----
    if(_monsterAttackRole == 1 && _map1MonsterMan1.getAttackTime() > 0){
        _map1MonsterMan1.moveAttackRight();
        _map1MonsterMan1.setAttackTime(_map1MonsterMan1.getAttackTime() - 1);
        if(_map1MonsterMan1.getAttackTime() == 0){
            role.setHP(role.getHp() - 1);
        }
    }
    else if(_monsterAttackRole == 0 && _map1MonsterMan1.getAttackTime() > 0){
        _map1MonsterMan1.moveAttackLeft();
        _map1MonsterMan1.setAttackTime(_map1MonsterMan1.getAttackTime() - 1);
        if(_map1MonsterMan1.getAttackTime() == 1){
            role.setHP(role.getHp() - 1);
        }
    }
    //-----地圖一怪獸二 攻擊角色動畫----
    if(_monsterAttackRole == 1 && _map1MonsterMan2.getAttackTime() > 0){
        _map1MonsterMan2.moveAttackRight();
        _map1MonsterMan2.setAttackTime(_map1MonsterMan2.getAttackTime() - 1);
        if(_map1MonsterMan2.getAttackTime() == 0){
            role.setHP(role.getHp() - 1);
        }
    }
    else if(_monsterAttackRole == 0 && _map1MonsterMan2.getAttackTime() > 0){
        _map1MonsterMan2.moveAttackLeft();
        _map1MonsterMan2.setAttackTime(_map1MonsterMan2.getAttackTime() - 1);
        if(_map1MonsterMan2.getAttackTime() == 1){
            role.setHP(role.getHp() - 1);
        }
    }
    //----地圖二怪獸一 攻擊角色動畫----
    if(_monsterAttackRole == 1 && _map2Monster1.getAttackTime() > 0){
        _map2Monster1.moveAttackRight();
        _map2Monster1.setAttackTime(_map2Monster1.getAttackTime() - 1);
        if(_map2Monster1.getAttackTime() == 0){
            role.setHP(role.getHp() - 1);
        }
    }
    else if(_monsterAttackRole == 0 && _map2Monster1.getAttackTime() > 0){
        _map2Monster1.moveAttackLeft();
        _map2Monster1.setAttackTime(_map2Monster1.getAttackTime() - 1);
        if(_map2Monster1.getAttackTime() == 1){
            role.setHP(role.getHp() - 1);
        }
    }
    //----地圖二怪獸二攻擊角色動畫----
    if(_monsterAttackRole == 1 && _map2Monster2.getAttackTime() > 0){
        _map2Monster2.moveAttackRight();
        _map2Monster2.setAttackTime(_map2Monster2.getAttackTime() - 1);
        if(_map2Monster2.getAttackTime() == 0){
            role.setHP(role.getHp() - 1);
        }
    }
    else if(_monsterAttackRole == 0 && _map2Monster2.getAttackTime() > 0){
        _map2Monster2.moveAttackLeft();
        _map2Monster2.setAttackTime(_map2Monster2.getAttackTime() - 1);
        if(_map2Monster2.getAttackTime() == 1){
            role.setHP(role.getHp() - 1);
        }
    }
    //-----王關攻擊
```

```
if(_monsterAttackRole == 1 && _king.getAttackTime() > 0 && !_remoteAttackRight && !_remoteAttackLeft){
    _king.moveAttackRight();
    _king.setAttackTime(_king.getAttackTime() - 1);
    if(_king.getAttackTime() == 0){
        role.setHP(role.getHp() - 1);
    }
}
else if(_monsterAttackRole == 0 && _king.getAttackTime() > 0 && !_remoteAttackRight && !_remoteAttackLeft){
    _king.moveAttackLeft();
    _king.setAttackTime(_king.getAttackTime() - 1);
    if(_king.getAttackTime() == 1){
        role.setHP(role.getHp() - 1);
    }
}
//-----王遠距離攻擊------
if(_remoteAttackRight && _king.getAttackTime() > 0){
    _king.moveRemoteAttackRight();
    _king.setAttackTime(_king.getAttackTime() - 1);
}
if(_remoteAttackRight && _bulletFlyTime > 0){
    _bulletFlyTime--;
    _bulletRight.setLocation(_bulletRight.getX() + 6, _bulletRight.getY());
    _bulletRight.setLocation(_bulletRight.getX() + 6, _bulletRight.getY());
    _bulletRight.setLocation(_bulletRight.getX() + 6 , _bulletRight.getY());

    if(DetectRemoteAttackRole(role,_bulletRight,_bulletLeft)){
        role.setHP(role.getHp() - 1);
        _bulletFlyTime = 0;
        _king.setAttackTime(0);
        _remoteAttackRight = false;
    }
    if(_bulletFlyTime == 0){
        _king.setAttackTime(0);
        _remoteAttackRight = false;
    }
}
if(_remoteAttackLeft && _king.getAttackTime() > 0){
    _test.setValue(_king.getAttackTime());
    _king.moveRemoteAttackLeft();
    _king.setAttackTime(_king.getAttackTime() - 1);
}
if(_remoteAttackLeft && _bulletFlyTime > 0){
    _bulletFlyTime--;
    _bulletLeft.setLocation(_bulletLeft.getX() - 6 , _bulletLeft.getY());
    _bulletLeft.setLocation(_bulletLeft.getX() - 6 , _bulletLeft.getY());
    _bulletLeft.setLocation(_bulletLeft.getX() - 6 , _bulletLeft.getY());

    if(DetectRemoteAttackRole(role,_bulletRight,_bulletLeft)){
        role.setHP(role.getHp() - 1);
        _king.setAttackTime(0);
        _bulletFlyTime = 0;
        _remoteAttackLeft = false;
    }
    if(_bulletFlyTime == 0){
        _king.setAttackTime(0);
        _remoteAttackLeft = false;
    }
}
//----角色被攻擊到的動畫-----
if(_roleBeAttacked > 0){
    _roleBeAttacked--;
    if(_roleBeAttacked == 3){
        role.setBeAttacked(false);
    }
}
if(role.getBeAttacked()){
    role.moveBeAttackedRight();
    role.moveBeAttackedLeft();
}
//------角色死亡------
```

```
if(role.getHp() == 0){
    roleDead = 18;
    role.setHP(role.getHp() - 1);
}
if(roleDead > 0){
    roleDead--;
    role.moveDeadRight();
    role.moveDeadLeft();
}
//-----怪獸被攻擊到的動畫----
if(_monsterBeAttacked > 0){
    _map1MonsterMan1.moveBeAttackedRight();
    _map1MonsterMan1.moveBeAttackedLeft();
    _map1MonsterMan2.moveBeAttackedRight();
    _map1MonsterMan2.moveBeAttackedLeft();
    _map2Monster1.moveBeAttackedRight();
    _map2Monster1.moveBeAttackedLeft();
    _map2Monster2.moveBeAttackedRight();
    _map2Monster2.moveBeAttackedLeft();
    _king.moveBeAttackedRight();
    _king.moveBeAttackedLeft();
    _monsterBeAttacked--;
    if(_monsterBeAttacked == 0){   //被打完要重製攻擊動畫，避免原本被攻擊到一半，下次攻擊動畫會很奇怪
        _map1MonsterMan1.setAttackRightCurrentFrame(1);
        _map1MonsterMan1.setAttackLefttCurrentFrame(1);
        _map1MonsterMan2.setAttackRightCurrentFrame(1);
        _map1MonsterMan2.setAttackLefttCurrentFrame(1);
        _map2Monster1.setAttackRightCurrentFrame(1);
        _map2Monster1.setAttackLefttCurrentFrame(1);
        _map2Monster2.setAttackRightCurrentFrame(1);
        _map2Monster2.setAttackLefttCurrentFrame(1);
        _king.setAttackRightCurrentFrame(1);
        _king.setAttackLefttCurrentFrame(1);
    }
}
//----障礙物血量等於 0 以及閃爍時間被設定且 > 0----
if(_map1Trashcan.getHp() == 0 && _diedShine > 0){
    _map1TrashCanAttacked.move();
    _diedShine -- ;
    if(_diedShine == 0){
        _map1Trashcan.setHp(_map1Trashcan.getHp() - 1);    //閃爍結束會讓障礙物血量變為-1
    }
}
if(_map1TrafficCon.getHp() == 0 && _diedShine > 0){
    _map1TrafficConAttacked.move();
    _diedShine -- ;
    if(_diedShine == 0){
        _map1TrafficCon.setHp(_map1TrafficCon.getHp() - 1);
    }
}
if(_map1Telephone.getHp() == 0 && _diedShine > 0){
    _map1TelephoneAttacked.move();
    _diedShine -- ;
    if(_diedShine == 0){
        _map1Telephone.setHp(_map1Telephone.getHp() - 1);
    }
}
if(_map2StreeLight.getHp() == 0 && _diedShine > 0){
    _map2StreeLightAttacked.move();
    _diedShine -- ;
    if(_diedShine == 0){
        _map2StreeLight.setHp(_map2StreeLight.getHp() - 1);
    }
}
if(_map2RecycleBin.getHp() == 0 && _diedShine > 0){
    _map2RecycleBinAttacked.move();
    _diedShine -- ;
    if(_diedShine == 0){
        _map2RecycleBin.setHp(_map2RecycleBin.getHp() - 1);
    }
```

```
    }
    if(_map2Car.getHp() == 0 && _diedShine > 0){
        _map2CarAttacked.move();
        _diedShine -- ;
        if(_diedShine == 0){
            _map2Car.setHp(_map2Car.getHp() - 1);
        }
    }
    if(_map1MonsterMan1.getHP() == 0 && (_diedShine > 0 || _map1MonsterMan1.getPlugDied() > 0)){
        _map1MonsterMan1.moveDeadRight();
        _map1MonsterMan1.moveDeadLeft();
        _diedShine--;
        _map1MonsterMan1.setPlugDied(_map1MonsterMan1.getPlugDied() - 1);
        if(_diedShine == 0 || _map1MonsterMan1.getPlugDied() == 0){
            _map1MonsterMan1.setDeadRightCurrentFrame(1);
            _map1MonsterMan1.setDeadLefttCurrentFrame(1);
            _monsterBeClear --;
            _map1MonsterMan1.setHP(_map1MonsterMan1.getHP() - 1);
        }
    }
    if(_map1MonsterMan2.getHP() == 0 && (_diedShine > 0 || _map1MonsterMan2.getPlugDied() > 0)){
        _map1MonsterMan2.moveDeadRight();
        _map1MonsterMan2.moveDeadLeft();
        _diedShine--;
        _map1MonsterMan2.setPlugDied(_map1MonsterMan2.getPlugDied() - 1);
        if(_diedShine == 0 || _map1MonsterMan2.getPlugDied() == 0){
            _map1MonsterMan2.setDeadRightCurrentFrame(1);
            _map1MonsterMan2.setDeadLefttCurrentFrame(1);
            _monsterBeClear --;
            _map1MonsterMan2.setHP(_map1MonsterMan2.getHP() - 1);
        }
    }
    if(_map2Monster1.getHP() == 0 && (_diedShine > 0 || _map2Monster1.getPlugDied() > 0)){
        _map2Monster1.moveDeadRight();
        _map2Monster1.moveDeadLeft();
        _diedShine--;
        _map2Monster1.setPlugDied(_map2Monster1.getPlugDied() - 1);
        if(_diedShine == 0 || _map2Monster1.getPlugDied() == 0){
            _map2Monster1.setDeadRightCurrentFrame(1);
            _map2Monster1.setDeadLefttCurrentFrame(1);
            _monsterBeClear --;
            _map2Monster1.setHP(_map2Monster1.getHP() - 1);
        }
    }
    if(_map2Monster2.getHP() == 0 && (_diedShine > 0 || _map2Monster2.getPlugDied() > 0)){
        _map2Monster2.moveDeadRight();
        _map2Monster2.moveDeadLeft();
        _diedShine--;
        _map2Monster2.setPlugDied(_map2Monster2.getPlugDied() - 1);
        if(_diedShine == 0 || _map2Monster2.getPlugDied() == 0){
            _map2Monster2.setDeadRightCurrentFrame(1);
            _map2Monster2.setDeadLefttCurrentFrame(1);
            _monsterBeClear --;
            _map2Monster2.setHP(_map2Monster2.getHP() - 1);
        }
    }
    //-----王關死亡動畫-----
    if(_king.getHP() == 0){
        _kingDiedShine = 54;
        _king.setHP(_king.getHP() - 1);
    }
    if(_kingDiedShine > 0){
        _king.moveDeadRight();
        _king.moveDeadLeft();
        _kingDiedShine--;
        if(_kingDiedShine == 0){
            _king.setDeadLefttCurrentFrame(1);
            _king.setDeadRightCurrentFrame(1);
        }
    }
```

```
//-----掉落物品-----
dropItemDetected(_map1MonsterMan1);
dropItemDetected(_map1MonsterMan2);
dropItemDetected(_map2Monster1);
dropItemDetected(_map2Monster2);
if(_dropBlood.getOnShow()){
   DetectGetDrop(_dropBlood);
}
if(_dropKunai.getOnShow()){
   DetectGetDrop(_dropKunai);
}
if(_dropShockWave.getOnShow()){
   DetectGetDrop(_dropShockWave);
}
if(_dropBlood.getPickupTime() > 0){
   _dropBlood.setPickupTime(_dropBlood.getPickupTime() - 1);
   _dropBlood.setXY(_roleX + 30,_roleY - 30);
}
if(_dropKunai.getPickupTime() > 0){
   _dropKunai.setPickupTime(_dropKunai.getPickupTime() - 1);
   _dropKunai.setXY(_roleX + 40,_roleY - 10);
}
if(_dropShockWave.getPickupTime() > 0){
   _dropShockWave.setPickupTime(_dropShockWave.getPickupTime() - 1);
   _dropShockWave.setXY(_roleX + 40,_roleY - 60);
}
if(_monsterBeClear == 0){
   showGO = 0;
}
//----以下為過關地圖捲動部分----
if(_roleX > nextMapGo.getX() && _mapNumber < 4 && showGO == 0){  //當角色碰觸到進關箭頭
   _backNumber = 30;  //地圖往後捲動 30 次
}
if(_backNumber > 0){   //過關觸碰到箭頭時 _backnumber 會設為 30
   _background.setLocation(_bx -= 15,_by); //每次往後捲動單位為 15
   role.setXY(_roleX -= 15,_roleY);
   //-----補給品-----
   _dropBlood.setXY(_dropBlood.getX() - 15 ,_dropBlood.getY());
   _dropKunai.setXY(_dropKunai.getX() - 15 ,_dropKunai.getY());
   _dropShockWave.setXY(_dropShockWave.getX() - 15 ,_dropShockWave.getY());
   //----第一張地圖----
   _map1Trashcan.setLocation(_map1Trashcan.getX() - 15 , _map1Trashcan.getY());
   _map1Telephone.setLocation(_map1Telephone.getX() - 15 , _map1Telephone.getY());
   _map1TrafficCon.setLocation(_map1TrafficCon.getX() - 15 , _map1TrafficCon.getY());
   //----第二張地圖----
   _map2StreeLight.setLocation(_map2StreeLight.getX() - 15 , _map2StreeLight.getY());
   _map2RecycleBin.setLocation(_map2RecycleBin.getX() - 15 ,_map2RecycleBin.getY());
   _map2Car.setLocation(_map2Car.getX() - 15 , _map2Car.getY());
   _map2MonsterWoman1X -= 15;
   _map2MonsterWoman2X -= 15;
   //----第三張地圖----
   _broke.setLocation(_broke.getX() - 15,_broke.getY());
   _kingX -= 15;
   _backNumber--;  //只做 30 次
   reset++ ;
}
//-----進王關動畫-----
if(_mapNumber == 2 && _kingTime == -1 && _backNumber == 0){
   _music.stop();
   _kingBGM.resume();
   _kingTime = 50;
}
if(_kingTime > 10){
   _kingTime-- ;
   warning.move();
   _kingY += 6;
}
else if(_kingTime > 0){
   if(_kingTime % 2 ==1){
      _background.setLocation(_bx-=5,_by);
```

```
            }
            else {
                _background.setLocation(_bx+=5,_by);
            }
            _kingTime--;
        }
        if(_kingTime == 0){
            _kingShow = true;
        }
        if(reset == 30){
            _monsterBeClear = 2;
            showGO = 1;
            reset = 0;
            _mapNumber++;   //過了幾關
        }
        if(_attackTime == 0){
            _attackLeftThing = false;
            _attackRightThing = false;
        }
        nextMapGo.move();
        _scores.setValue(_king.getX());
        //-----外掛------
        if(_grabDown && _grabAttack && _grabSmite){
            if(_mapNumber == 0){
                if(_map1MonsterMan1.getHP() > 0){
                    _map1MonsterMan1.setPlugDied(20);
                    _map1MonsterMan1.setHP(0);
                }
                if(_map1MonsterMan2.getHP() > 0){
                    _map1MonsterMan2.setPlugDied(20);
                    _map1MonsterMan2.setHP(0);
                }
            }
            if(_mapNumber == 1){
                if(_map2Monster1.getHP() > 0){
                    _map2Monster1.setPlugDied(20);
                    _map2Monster1.setHP(0);
                }
                if(_map2Monster2.getHP() > 0){
                    _map2Monster2.setPlugDied(20);
                    _map2Monster2.setHP(0);
                }
            }
            if(_mapNumber == 2 && _kingShow){
                if(_king.getHP() > 0){
                    _king.setHP(0);
                }
            }
        }
        //-----按下 restart 鍵-----
        if(_grabRestart){
            init();
        }
    }
    @Override
    public void show() {
        _background.show();
        //------王關顯示------
        if(_remoteAttackRight){
            _bulletRight.show();
        }
        if(_remoteAttackLeft){
            _bulletLeft.show();
        }
        if(_roleY >= _king.getY() && _broke.getY() >= _king.getY() && _kingShow && _mapNumber == 2 && _king.getHP() > 0
&& !_king.getBeAttacked()){
            _king.setXY(_kingX,_kingY);
            if(_king.getDir() == 1 && _king.getAttackTime() == 0){
                _king.showRight();
            }
```

```
            if(_king.getDir() == 1 && _remoteAttackRight && _king.getAttackTime() > 0){
                _king.showRemoteAttackRight();
            }
            else if(_king.getDir() == 1 && _king.getAttackTime() > 0){
                _king.showAttackRight();
            }
            if(_king.getDir() == 0 && _king.getAttackTime() == 0){
                _king.showLeft();
            }
            if(_king.getDir() == 0 && _remoteAttackLeft  && _king.getAttackTime() > 0){
                _king.showRemoteAttackLeft();
            }
            else if(_king.getDir() == 0 && _king.getAttackTime() > 0){
                _king.showAttackLeft();
            }
        }
        else if(_roleY >= _king.getY() && _broke.getY() >= _king.getY() && _kingShow && _mapNumber==2 && _king.getHP() > 0 &&
_king.getBeAttacked()){
            _king.setXY(_kingX,_kingY);
            if (_king.getDir() == 1) {
                _king.showBeAttackedRight();
                if(_monsterBeAttacked == 0){
                    _king.setBeAttacked(false);
                }
            }
            if (_king.getDir() == 0) {
                _king.showBeAttackedLeft();
                if(_monsterBeAttacked == 0){
                    _king.setBeAttacked(false);
                }
            }
        }
        if(_roleY >= _broke.getY() && _kingTime < 10 && _kingTime != -1){
            _broke.show();
        }
        if(_roleY >= _king.getY() && _broke.getY() < _king.getY() && _kingShow && _mapNumber==2 && _king.getHP() > 0
&& !_king.getBeAttacked()){
            _king.setXY(_kingX,_kingY);
            if(_king.getDir() == 1 && _king.getAttackTime() == 0){
                _king.showRight();
            }
            if(_king.getDir() == 1 && _remoteAttackRight  && _king.getAttackTime() > 0){
                _king.showRemoteAttackRight();
            }
            else if(_king.getDir() == 1 && _king.getAttackTime() > 0){
                _king.showAttackRight();
            }
            if(_king.getDir() == 0 && _king.getAttackTime() == 0){
                _king.showLeft();
            }
            if(_king.getDir() == 0 && _remoteAttackLeft  && _king.getAttackTime() > 0){
                _king.showRemoteAttackLeft();
            }
            else if(_king.getDir() == 0 && _king.getAttackTime() > 0){
                _king.showAttackLeft();
            }
        }
        else if(_roleY >= _king.getY() && _broke.getY() < _king.getY() && _kingShow && _mapNumber==2 && _king.getHP() > 0 &&
_king.getBeAttacked()) {
            _king.setXY(_kingX, _kingY);
            if (_king.getDir() == 1) {
                _king.showBeAttackedRight();
                if (_monsterBeAttacked == 0) {
                    _king.setBeAttacked(false);
                }
            }
            if (_king.getDir() == 0) {
                _king.showBeAttackedLeft();
                if (_monsterBeAttacked == 0) {
                    _king.setBeAttacked(false);
```

```
            }
        }
    }
    if(_map1Telephone.getHp() > 0) {   //電話亭血量大於 0 才會顯示
        _map1Telephone.show();
    }
    if(_map2StreeLight.getHp() > 0) {  //路燈血量大於 0 才會顯示
        _map2StreeLight.show();
    }
    if(_map2RecycleBin.getHp() > 0){
        _map2RecycleBin.show();
    }
    //----地圖一怪獸與障礙物前後的顯示判定----
    if(_roleY >= _map1TrafficCon.getY() && _map1MonsterMan1Y >= _map1TrafficCon.getY() && _map1MonsterMan2Y >=
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
        _map1TrafficCon.show();
    }
    if(_roleY >= _map1MonsterMan2Y && _map1MonsterMan1Y >= _map1MonsterMan2Y && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && !_map1MonsterMan2.getBeAttacked()) {
        _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
        if (_map1MonsterMan2.getDir() == 1 && _map1MonsterMan2.getAttackTime() == 0) {
            _map1MonsterMan2.showRight();
        }
        if(_monsterAttackRole == 1 && _map1MonsterMan2.getAttackTime() > 0){
            _map1MonsterMan2.showAttackRight();
        }
        if ( _map1MonsterMan2.getDir() == 0 && _map1MonsterMan2.getAttackTime() == 0) {
            _map1MonsterMan2.showLeft();
        }
        if(_monsterAttackRole == 0 && _map1MonsterMan2.getAttackTime() > 0){
            _map1MonsterMan2.showAttackLeft();
        }
    }
    else if(_roleY >= _map1MonsterMan2Y && _map1MonsterMan1Y >= _map1MonsterMan2Y && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && _map1MonsterMan2.getBeAttacked()){
        _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
        if (_map1MonsterMan2.getDir() == 1) {
            _map1MonsterMan2.showBeAttackedRight();
            if(_monsterBeAttacked == 0){
                _map1MonsterMan2.setBeAttacked(false);
            }
        }
        if (_map1MonsterMan2.getDir() == 0) {
            _map1MonsterMan2.showBeAttackedLeft();
            if(_monsterBeAttacked == 0){
                _map1MonsterMan2.setBeAttacked(false);
            }
        }
    }
    if(_roleY >= _map1TrafficCon.getY() && _map1MonsterMan1Y >= _map1TrafficCon.getY() && _map1MonsterMan2Y <
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
        _map1TrafficCon.show();
    }
    if(_roleY >= _map1MonsterMan1Y && _map1MonsterMan1.getHP() > 0 && _mapNumber == 0
&& !_map1MonsterMan1.getBeAttacked() ){
        _map1MonsterMan1.setXY(_map1MonsterMan1X,_map1MonsterMan1Y);
        if(_map1MonsterMan1.getDir() == 1 && _map1MonsterMan1.getAttackTime() == 0){
            _map1MonsterMan1.showRight();
        }
        if(_monsterAttackRole == 1 && _map1MonsterMan1.getAttackTime() > 0){
            _map1MonsterMan1.showAttackRight();
        }
        if(_map1MonsterMan1.getDir() == 0 && _map1MonsterMan1.getAttackTime() == 0) {
            _map1MonsterMan1.showLeft();
        }
        if(_monsterAttackRole == 0 && _map1MonsterMan1.getAttackTime() > 0){
            _map1MonsterMan1.showAttackLeft();
        }
    }
```

```
    else if(_roleY >= _map1MonsterMan1Y && _map1MonsterMan1.getHP() > 0 && _mapNumber == 0 &&
_map1MonsterMan1.getBeAttacked()){
        _map1MonsterMan1.setXY(_map1MonsterMan1X,_map1MonsterMan1Y);
        if(_map1MonsterMan1.getDir() == 1){
            _map1MonsterMan1.showBeAttackedRight();
            if(_monsterBeAttacked == 0) {
                _map1MonsterMan1.setBeAttacked(false);
            }
        }
        if(_map1MonsterMan1.getDir() == 0){
            _map1MonsterMan1.showBeAttackedLeft();
            if(_monsterBeAttacked == 0) {
                _map1MonsterMan1.setBeAttacked(false);
            }
        }
    }
    if(_roleY >= _map1TrafficCon.getY() && _map1MonsterMan1Y < _map1TrafficCon.getY() && _map1MonsterMan2Y >=
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
        _map1TrafficCon.show();
    }
    if(_roleY >= _map1MonsterMan2Y && _map1MonsterMan1Y < _map1MonsterMan2Y  && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && !_map1MonsterMan2.getBeAttacked()) {
        _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
        if (_map1MonsterMan2.getDir() == 1 && _map1MonsterMan2.getAttackTime() == 0) {
            _map1MonsterMan2.showRight();
        }
        if(_monsterAttackRole == 1 && _map1MonsterMan2.getAttackTime() > 0){
            _map1MonsterMan2.showAttackRight();
        }
        if (_map1MonsterMan2.getDir() == 0 && _map1MonsterMan2.getAttackTime() == 0) {
            _map1MonsterMan2.showLeft();
        }
        if(_monsterAttackRole == 0 && _map1MonsterMan2.getAttackTime() > 0){
            _map1MonsterMan2.showAttackLeft();
        }
    }
    else if(_roleY >= _map1MonsterMan2Y && _map1MonsterMan1Y < _map1MonsterMan2Y  && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && _map1MonsterMan2.getBeAttacked()) {
        _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
        if (_map1MonsterMan2.getDir() == 1) {
            _map1MonsterMan2.showBeAttackedRight();
            if(_monsterBeAttacked == 0){
                _map1MonsterMan2.setBeAttacked(false);
            }
        }
        if (_map1MonsterMan2.getDir() == 0) {
            _map1MonsterMan2.showBeAttackedLeft();
            if(_monsterBeAttacked == 0){
                _map1MonsterMan2.setBeAttacked(false);
            }
        }
    }
    //----地圖二怪獸之間與角色前後顯示判定----
    if(_roleY >= _map2MonsterWoman2Y && _map2MonsterWoman1Y >= _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && !_map2Monster2.getBeAttacked()) {
        _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
        if (_map2Monster2.getDir() == 1 && _map2Monster2.getAttackTime() == 0) {
            _map2Monster2.showRight();
        }
        if(_monsterAttackRole == 1 && _map2Monster2.getAttackTime() > 0 ){
            _map2Monster2.showAttackRight();
        }
        if (_map2Monster2.getDir() == 0 && _map2Monster2.getAttackTime() == 0) {
            _map2Monster2.showLeft();
        }
        if(_monsterAttackRole == 0 && _map2Monster2.getAttackTime() > 0 ){
            _map2Monster2.showAttackLeft();
        }
    }
```

```
        else if(_roleY >= _map2MonsterWoman2Y && _map2MonsterWoman1Y >= _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && _map2Monster2.getBeAttacked()){
            _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
            if (_map2Monster2.getDir() == 1) {
                _map2Monster2.showBeAttackedRight();
                if(_monsterBeAttacked == 0){
                    _map2Monster2.setBeAttacked(false);
                }
            }
            if (_map2Monster2.getDir() == 0) {
                _map2Monster2.showBeAttackedLeft();
                if(_monsterBeAttacked == 0){
                    _map2Monster2.setBeAttacked(false);
                }
            }
        }
        if(_roleY >= _map2MonsterWoman1Y && _map2Monster1.getHP() > 0 && _mapNumber == 1 && !_map2Monster1.getBeAttacked()) {
            _map2Monster1.setXY(_map2MonsterWoman1X,_map2MonsterWoman1Y);
            if (_map2Monster1.getDir() == 1 && _map2Monster1.getAttackTime() == 0) {
                _map2Monster1.showRight();
            }
            if(_monsterAttackRole == 1 && _map2Monster1.getAttackTime() > 0 ){
                _map2Monster1.showAttackRight();
            }
            if (_map2Monster1.getDir() == 0 && _map2Monster1.getAttackTime() == 0) {
                _map2Monster1.showLeft();
            }
            if(_monsterAttackRole == 0 && _map2Monster1.getAttackTime() > 0 ){
                _map2Monster1.showAttackLeft();
            }
        }
        else if(_roleY >= _map2MonsterWoman1Y && _map2Monster1.getHP() > 0 && _mapNumber == 1 && _map2Monster1.getBeAttacked()){
            _map2Monster1.setXY(_map2MonsterWoman1X,_map2MonsterWoman1Y);
            if (_map2Monster1.getDir() == 1) {
                _map2Monster1.showBeAttackedRight();
                if(_monsterBeAttacked == 0){
                    _map2Monster1.setBeAttacked(false);
                }
            }
            if (_map2Monster1.getDir() == 0) {
                _map2Monster1.showBeAttackedLeft();
                if(_monsterBeAttacked == 0){
                    _map2Monster1.setBeAttacked(false);
                }
            }
        }
        if(_roleY >= _map2MonsterWoman2Y && _map2MonsterWoman1Y < _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && !_map2Monster2.getBeAttacked()) {
            _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
            if (_map2Monster2.getDir() == 1 && _map2Monster2.getAttackTime() == 0) {
                _map2Monster2.showRight();
            }
            if(_monsterAttackRole == 1 && _map2Monster2.getAttackTime() > 0 ){
                _map2Monster2.showAttackRight();
            }
            if (_map2Monster2.getDir() == 0 && _map2Monster2.getAttackTime() == 0) {
                _map2Monster2.showLeft();
            }
            if(_monsterAttackRole == 0 && _map2Monster2.getAttackTime() > 0 ){
                _map2Monster2.showAttackLeft();
            }
        }
        else if(_roleY >= _map2MonsterWoman2Y && _map2MonsterWoman1Y < _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && _map2Monster2.getBeAttacked()){
            _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
            if (_map2Monster2.getDir() == 1) {
                _map2Monster2.showBeAttackedRight();
                if(_monsterBeAttacked == 0){
                    _map2Monster2.setBeAttacked(false);
                }
```

```
        }
        if (_map2Monster2.getDir() == 0) {
            _map2Monster2.showBeAttackedLeft();
            if(_monsterBeAttacked == 0){
                _map2Monster2.setBeAttacked(false);
            }
        }
    }
    if(_roleY >= _map1TrafficCon.getY() && _map1MonsterMan1Y < _map1TrafficCon.getY() && _map1MonsterMan2Y <
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
        _map1TrafficCon.show();
    }
    //----角色各式動作----
    if(_direction == 1 && !_slideR && _attackTime == 0 && _simteTime == 0 && !role.getBeAttacked() && role.getHp() > 0) {   //方向為 1 是
向右時 才顯示向右動畫
        role.showRight();
    }
    else if(_direction == 1 && role.getBeAttacked() && role.getHp() > 0){
        role.setXY(_roleX,_roleY);
        role.showBeAttackedRight();
    }
    if(_direction == 0 && !_slideL && _attackTime == 0 && _simteTime == 0 && !role.getBeAttacked() && role.getHp() > 0) {   //方向為 0 是
向左時 才顯示向左動畫
        role.showLeft();
    }
    else if(_direction == 0 && role.getBeAttacked() && role.getHp() > 0){
        role.setXY(_roleX,_roleY);
        role.showBeAttackedLeft();
    }
    if(roleDead > 0 && _direction == 1){
        role.setXY(_roleX,_roleY);
        role.showDeadRight();
    }
    if(roleDead > 0 && _direction == 0){
        role.setXY(_roleX,_roleY);
        role.showDeadLeft();
    }
    if(_direction == 1 && _attackTime > 0){
        role.showAttackRight();
    }
    if(_direction == 0 && _attackTime > 0){
        role.showAttackLeft();
    }
    //----丟苦無的動作 & 苦無的顯示----
    if(_direction == 1 && _simteTime > 0) {
        role.showSmiteRight();
    }
    if(_direction == 1 && _kunaiFlyTime > 0) {
        _kunaiRight.show();
    }
    if(_direction == 1 && _shockWaveTime > 0){
        _shockWaveRight.show();
    }
    if(_direction == 0 && _shockWaveTime > 0){
        _shockWaveLeft.show();
    }
    if(_direction == 0 && _simteTime > 0) {
        role.showSmiteLeft();
    }
    if(_direction == 0 && _kunaiFlyTime > 0) {
        _kunaiLeft.show();
    }
    if(_slideR) {   //向右滑行
        role.showSlideRight();
    }
    if(_slideL) {   //向左滑行
        role.showSlideLeft();
    }
    //----地圖一 怪獸與障礙物前後顯示判定-----
```

```
    if(_roleY < _map1TrafficCon.getY() && _map1MonsterMan1Y >= _map1TrafficCon.getY() && _map1MonsterMan2Y >=
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
        _map1TrafficCon.show();
    }
    if(_roleY < _map1MonsterMan2Y && _map1MonsterMan1Y >= _map1MonsterMan2Y  && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && !_map1MonsterMan2.getBeAttacked()) {
        _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
        if (_map1MonsterMan2.getDir() == 1 && _map1MonsterMan2.getAttackTime() == 0) {
            _map1MonsterMan2.showRight();
        }
        if(_monsterAttackRole == 1 && _map1MonsterMan2.getAttackTime() > 0){
            _map1MonsterMan2.showAttackRight();
        }
        if ( _map1MonsterMan2.getDir() == 0 && _map1MonsterMan2.getAttackTime() == 0) {
            _map1MonsterMan2.showLeft();
        }
        if(_monsterAttackRole == 0 && _map1MonsterMan2.getAttackTime() > 0){
            _map1MonsterMan2.showAttackLeft();
        }
    }
    else if(_roleY < _map1MonsterMan2Y && _map1MonsterMan1Y >= _map1MonsterMan2Y  && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && _map1MonsterMan2.getBeAttacked()) {
        _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
        if (_map1MonsterMan2.getDir() == 1) {
            _map1MonsterMan2.showBeAttackedRight();
            if(_monsterBeAttacked == 0){
                _map1MonsterMan2.setBeAttacked(false);
            }
        }
        if (_map1MonsterMan2.getDir() == 0) {
            _map1MonsterMan2.showBeAttackedLeft();
            if(_monsterBeAttacked == 0){
                _map1MonsterMan2.setBeAttacked(false);
            }
        }
    }
    if(_roleY < _map1TrafficCon.getY() && _map1MonsterMan1Y >= _map1TrafficCon.getY() && _map1MonsterMan2Y <
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
        _map1TrafficCon.show();
    }
    if(_roleY < _map1MonsterMan1Y  && _map1MonsterMan1.getHP() > 0 && _mapNumber == 0 && !_map1MonsterMan1.getBeAttacked()){
        _map1MonsterMan1.setXY(_map1MonsterMan1X,_map1MonsterMan1Y);
        if(_map1MonsterMan1.getDir() == 1 && _map1MonsterMan1.getAttackTime() == 0){
            _map1MonsterMan1.showRight();
        }
        if(_monsterAttackRole == 1 && _map1MonsterMan1.getAttackTime() > 0){
            _map1MonsterMan1.showAttackRight();
        }
        if(_map1MonsterMan1.getDir() == 0 && _map1MonsterMan1.getAttackTime() == 0) {
            _map1MonsterMan1.showLeft();
        }
        if(_monsterAttackRole == 0 && _map1MonsterMan1.getAttackTime() > 0){
            _map1MonsterMan1.showAttackLeft();
        }
    }
    else if(_roleY < _map1MonsterMan1Y  && _map1MonsterMan1.getHP() > 0 && _mapNumber == 0 &&
_map1MonsterMan1.getBeAttacked()){
        _map1MonsterMan1.setXY(_map1MonsterMan1X,_map1MonsterMan1Y);
        if(_map1MonsterMan1.getDir() == 1){
            _map1MonsterMan1.showBeAttackedRight();
            if(_monsterBeAttacked == 0) {
                _map1MonsterMan1.setBeAttacked(false);
            }
        }
        if(_map1MonsterMan1.getDir() == 0){
            _map1MonsterMan1.showBeAttackedLeft();
            if(_monsterBeAttacked == 0) {
                _map1MonsterMan1.setBeAttacked(false);
            }
        }
```

```
        }
        if(_roleY < _map1TrafficCon.getY() && _map1MonsterMan1Y < _map1TrafficCon.getY() && _map1MonsterMan2Y >=
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
            _map1TrafficCon.show();
        }
        if(_roleY < _map1MonsterMan2Y && _map1MonsterMan1Y < _map1MonsterMan2Y  && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && !_map1MonsterMan2.getBeAttacked()) {
            _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
            if (_map1MonsterMan2.getDir() == 1 && _map1MonsterMan2.getAttackTime() == 0) {
                _map1MonsterMan2.showRight();
            }
            if(_monsterAttackRole == 1 && _map1MonsterMan2.getAttackTime() > 0){
                _map1MonsterMan2.showAttackRight();
            }
            if ( _map1MonsterMan2.getDir() == 0 && _map1MonsterMan2.getAttackTime() == 0) {
                _map1MonsterMan2.showLeft();
            }
            if(_monsterAttackRole == 0 && _map1MonsterMan2.getAttackTime() > 0){
                _map1MonsterMan2.showAttackLeft();
            }
        }
        else if(_roleY < _map1MonsterMan2Y && _map1MonsterMan1Y < _map1MonsterMan2Y  && _map1MonsterMan2.getHP() > 0 &&
_mapNumber == 0 && _map1MonsterMan2.getBeAttacked()) {
            _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
            if (_map1MonsterMan2.getDir() == 1) {
                _map1MonsterMan2.showBeAttackedRight();
                if(_monsterBeAttacked == 0){
                    _map1MonsterMan2.setBeAttacked(false);
                }
            }
            if (_map1MonsterMan2.getDir() == 0) {
                _map1MonsterMan2.showBeAttackedLeft();
                if(_monsterBeAttacked == 0){
                    _map1MonsterMan2.setBeAttacked(false);
                }
            }
        }
        if(_roleY < _map1TrafficCon.getY() && _map1MonsterMan1Y < _map1TrafficCon.getY() && _map1MonsterMan2Y <
_map1TrafficCon.getY() && _map1TrafficCon.getHp() > 0){  //三角錐血量大於 0 才會顯示
            _map1TrafficCon.show();
        }
        //----地圖二 怪獸之間與角色前後顯示判定----
        if(_roleY < _map2MonsterWoman2Y && _map2MonsterWoman1Y >= _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && !_map2Monster2.getBeAttacked()) {
            _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
            if (_map2Monster2.getDir() == 1 && _map2Monster2.getAttackTime() == 0) {
                _map2Monster2.showRight();
            }
            if(_monsterAttackRole == 1 && _map2Monster2.getAttackTime() > 0 ){
                _map2Monster2.showAttackRight();
            }
            if (_map2Monster2.getDir() == 0 && _map2Monster2.getAttackTime() == 0) {
                _map2Monster2.showLeft();
            }
            if(_monsterAttackRole == 0 && _map2Monster2.getAttackTime() > 0 ){
                _map2Monster2.showAttackLeft();
            }
        }
        else if(_roleY < _map2MonsterWoman2Y && _map2MonsterWoman1Y >= _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && _map2Monster2.getBeAttacked()){
            _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
            if (_map2Monster2.getDir() == 1) {
                _map2Monster2.showBeAttackedRight();
                if(_monsterBeAttacked == 0){
                    _map2Monster2.setBeAttacked(false);
                }
            }
            if (_map2Monster2.getDir() == 0) {
                _map2Monster2.showBeAttackedLeft();
                if(_monsterBeAttacked == 0){
```

```
                _map2Monster2.setBeAttacked(false);
            }
        }
    }
    if(_roleY < _map2MonsterWoman1Y && _map2Monster1.getHP() > 0 && _mapNumber == 1 && !_map2Monster1.getBeAttacked()) {
        _map2Monster1.setXY(_map2MonsterWoman1X,_map2MonsterWoman1Y);
        if (_map2Monster1.getDir() == 1 && _map2Monster1.getAttackTime() == 0) {
            _map2Monster1.showRight();
        }
        if(_monsterAttackRole == 1 && _map2Monster1.getAttackTime() > 0 ){
            _map2Monster1.showAttackRight();
        }
        if (_map2Monster1.getDir() == 0 && _map2Monster1.getAttackTime() == 0) {
            _map2Monster1.showLeft();
        }
        if(_monsterAttackRole == 0 && _map2Monster1.getAttackTime() > 0 ){
            _map2Monster1.showAttackLeft();
        }
    }
    else if(_roleY < _map2MonsterWoman1Y && _map2Monster1.getHP() > 0 && _mapNumber == 1 && _map2Monster1.getBeAttacked()){
        _map2Monster1.setXY(_map2MonsterWoman1X,_map2MonsterWoman1Y);
        if (_map2Monster1.getDir() == 1) {
            _map2Monster1.showBeAttackedRight();
            if(_monsterBeAttacked == 0){
                _map2Monster1.setBeAttacked(false);
            }
        }
        if (_map2Monster1.getDir() == 0) {
            _map2Monster1.showBeAttackedLeft();
            if(_monsterBeAttacked == 0){
                _map2Monster1.setBeAttacked(false);
            }
        }
    }
    if(_roleY < _map2MonsterWoman2Y && _map2MonsterWoman1Y < _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && !_map2Monster2.getBeAttacked()) {
        _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
        if (_map2Monster2.getDir() == 1 && _map2Monster2.getAttackTime() == 0) {
            _map2Monster2.showRight();
        }
        if(_monsterAttackRole == 1 && _map2Monster2.getAttackTime() > 0 ){
            _map2Monster2.showAttackRight();
        }
        if (_map2Monster2.getDir() == 0 && _map2Monster2.getAttackTime() == 0) {
            _map2Monster2.showLeft();
        }
        if(_monsterAttackRole == 0 && _map2Monster2.getAttackTime() > 0 ){
            _map2Monster2.showAttackLeft();
        }
    }
    else if(_roleY < _map2MonsterWoman2Y && _map2MonsterWoman1Y < _map2MonsterWoman2Y && _map2Monster2.getHP() > 0 &&
_mapNumber == 1 && _map2Monster2.getBeAttacked()){
        _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
        if (_map2Monster2.getDir() == 1) {
            _map2Monster2.showBeAttackedRight();
            if(_monsterBeAttacked == 0){
                _map2Monster2.setBeAttacked(false);
            }
        }
        if (_map2Monster2.getDir() == 0) {
            _map2Monster2.showBeAttackedLeft();
            if(_monsterBeAttacked == 0){
                _map2Monster2.setBeAttacked(false);
            }
        }
    }
    //------王關顯示------
    if(_roleY < _king.getY() && _broke.getY() >= _king.getY() && _kingShow && _mapNumber==2 && _king.getHP() > 0
&& !_king.getBeAttacked()){
        _king.setXY(_kingX,_kingY);
```

```
        if(_king.getDir() == 1 && _king.getAttackTime() == 0){
            _king.showRight();
        }
        if(_king.getDir() == 1 && _remoteAttackRight  && _king.getAttackTime() > 0){
            _king.showRemoteAttackRight();
        }
        else if(_king.getDir() == 1 && _king.getAttackTime() > 0){
            _king.showAttackRight();
        }
        if(_king.getDir() == 0 && _king.getAttackTime() == 0){
            _king.showLeft();
        }
        if(_king.getDir() == 0 && _remoteAttackLeft  && _king.getAttackTime() > 0){
            _king.showRemoteAttackLeft();
        }
        else if(_king.getDir() == 0 && _king.getAttackTime() > 0){
            _king.showAttackLeft();
        }
    }
    else if(_roleY < _king.getY() && _broke.getY() >= _king.getY() && _kingShow && _mapNumber==2 && _king.getHP() > 0 &&
_king.getBeAttacked()) {
        _king.setXY(_kingX, _kingY);
        if (_king.getDir() == 1) {
            _king.showBeAttackedRight();
            if (_monsterBeAttacked == 0) {
                _king.setBeAttacked(false);
            }
        }
        if (_king.getDir() == 0) {
            _king.showBeAttackedLeft();
            if (_monsterBeAttacked == 0) {
                _king.setBeAttacked(false);
            }
        }
    }
    if(_roleY < _broke.getY() && _kingTime < 10 && _kingTime != -1){
        _broke.show();
    }
    if(_roleY < _king.getY() && _broke.getY() < _king.getY() && _kingShow && _mapNumber==2 && _king.getHP() > 0
&& !_king.getBeAttacked()){
        _king.setXY(_kingX,_kingY);
        if(_king.getDir() == 1 && _king.getAttackTime() == 0){
            _king.showRight();
        }
        if(_king.getDir() == 1 && _remoteAttackRight){
            _king.showRemoteAttackRight();
        }
        else if(_king.getDir() == 1 && _king.getAttackTime() > 0){
            _king.showAttackRight();
        }
        if(_king.getDir() == 0 && _king.getAttackTime() == 0){
            _king.showLeft();
        }
        if(_king.getDir() == 0 && _remoteAttackLeft){
            _king.showRemoteAttackLeft();
        }
        else if(_king.getDir() == 0 && _king.getAttackTime() > 0){
            _king.showAttackLeft();
        }
    }
    else if(_roleY < _king.getY() && _broke.getY() < _king.getY() && _kingShow && _mapNumber==2 && _king.getHP() > 0 &&
_king.getBeAttacked()) {
        _king.setXY(_kingX, _kingY);
        if (_king.getDir() == 1) {
            _king.showBeAttackedRight();
            if (_monsterBeAttacked == 0) {
                _king.setBeAttacked(false);
            }
        }
        if (_king.getDir() == 0) {
```

```
            _king.showBeAttackedLeft();
            if (_monsterBeAttacked == 0) {
                _king.setBeAttacked(false);
            }
        }
    }
    //-----王關動畫區-----
    if(_kingTime > 0) {
        _king.setXY(_kingX,_kingY);
        if (_kingTime > 20) {
            warning.show();
        }
        if(_kingTime < 11){
            _kingHp[_kingHpShow].show();
            _kingHpShow++;
        }
    _king.showLeft();
    }
    //----垃圾桶血量大於 0 才會顯示----
    if(_map1Trashcan.getHp() > 0) {
        _map1Trashcan.show();
    }
    if(_map2Car.getHp() > 0){
        _map2Car.show();
    }
   //----障礙物血量等於 0 以及 消失閃爍的時間 > 0 才會顯示 障礙物消失的動畫，閃完後障礙物就消失----
    if(_map1TrafficCon.getHp() == 0 && _diedShine > 0){
       _map1TrafficConAttacked.setLocation(_map1TrafficCon.getX(),_map1TrafficCon.getY());
       _map1TrafficConAttacked.show();
    }
    if(_map1Trashcan.getHp() == 0 && _diedShine > 0){
        _map1TrashCanAttacked.setLocation(_map1Trashcan.getX(),_map1Trashcan.getY());
        _map1TrashCanAttacked.show();
    }
    if(_map1Telephone.getHp() == 0 && _diedShine > 0){
        _map1TelephoneAttacked.setLocation(_map1Telephone.getX(),_map1Telephone.getY());
        _map1TelephoneAttacked.show();
    }
    if(_map2StreeLight.getHp() == 0 && _diedShine > 0){
        _map2StreeLightAttacked.setLocation(_map2StreeLight.getX(),_map2StreeLight.getY());
        _map2StreeLightAttacked.show();
    }
    if(_map2RecycleBin.getHp() == 0 && _diedShine > 0){
        _map2RecycleBinAttacked.setLocation(_map2RecycleBin.getX(),_map2RecycleBin.getY());
        _map2RecycleBinAttacked.show();
    }
    if(_map2Car.getHp() == 0 && _diedShine > 0){
        _map2CarAttacked.setLocation(_map2Car.getX(),_map2Car.getY());
        _map2CarAttacked.show();
    }
    //----怪物死亡----
    if(_map1MonsterMan1.getHP() == 0 && (_diedShine > 0 || _map1MonsterMan1.getPlugDied() > 0)){
        _map1MonsterMan1.setXY(_map1MonsterMan1X,_map1MonsterMan1Y);
        if(_map1MonsterMan1.getDir() == 1){
            _map1MonsterMan1.showDeadRight();
        }
        if(_map1MonsterMan1.getDir() == 0){
            _map1MonsterMan1.showDeadLeft();
        }
    }
    if(_map1MonsterMan2.getHP() == 0 && (_diedShine > 0 || _map1MonsterMan2.getPlugDied() > 0)){
        _map1MonsterMan2.setXY(_map1MonsterMan2X,_map1MonsterMan2Y);
        if(_map1MonsterMan2.getDir() == 1){
            _map1MonsterMan2.showDeadRight();
        }
        if(_map1MonsterMan2.getDir() == 0){
            _map1MonsterMan2.showDeadLeft();
        }
    }
    if(_map2Monster1.getHP() == 0 && (_diedShine > 0 || _map2Monster1.getPlugDied() > 0)){
```

```
        _map2Monster1.setXY(_map2MonsterWoman1X,_map2MonsterWoman1Y);
        if(_map2Monster1.getDir() == 1){
            _map2Monster1.showDeadRight();
        }
        if(_map2Monster1.getDir() == 0){
            _map2Monster1.showDeadLeft();
        }
    }
    if(_map2Monster2.getHP() == 0 && (_diedShine > 0 || _map2Monster2.getPlugDied() > 0)){
        _map2Monster2.setXY(_map2MonsterWoman2X,_map2MonsterWoman2Y);
        if(_map2Monster2.getDir() == 1){
            _map2Monster2.showDeadRight();
        }
        if(_map2Monster2.getDir() == 0){
            _map2Monster2.showDeadLeft();
        }
    }
    if(_kingDiedShine > 0){
        _king.setXY(_kingX,_kingY);
        if(_king.getDir() == 1){
            _king.showDeadRight();
        }
        if(_king.getDir() == 0){
            _king.showDeadLeft();
        }
    }
    //-----怪物死亡掉落物品-----
    if(_dropBlood.getOnShow() || _dropBlood.getPickupTime() > 0){
        _dropBlood.show();
    }
    if(_dropKunai.getOnShow() || _dropKunai.getPickupTime() > 0){
        _dropKunai.show();
    }
    if(_dropShockWave.getOnShow() || _dropShockWave.getPickupTime() > 0){
        _dropShockWave.show();
    }
    if(_kunaiText.getText() > 0){
        _kunaiText.show();
        _kunaiText.setTextTime(_kunaiText.getText() - 1);
    }
    if(_shockWaveText.getText() > 0){
        _shockWaveText.show();
        _shockWaveText.setTextTime(_shockWaveText.getText() - 1);
    }
    //----前方有障礙物被打到時，顯示東西被打到的動畫 小於 7 跟大於 3 是用來持續顯示的時間
    if(_attackRightThing && _attackTime < 7 && _attackTime > 3){
        _explosionRight.show();
    }
    if(_attackLeftThing && _attackTime < 7 && _attackTime > 3){
        _explosionLeft.show();
    }
    //----按鍵區----
    rightButton.show();
    leftButton.show();
    upButton.show();
    downButton.show();
    attackButton.show();
    smiteButton.show();
    if(showGO == 0) {
        nextMapGo.show();
    }
    if(_map1MonsterMan1.getBeAttacked() && _map1MonsterMan1.getHP() > 0) {
        _mhp[_map1MonsterMan1.getHP()].show();
        _mhp[_map1MonsterMan1.getHP()].show();
    }
    if(_map1MonsterMan2.getBeAttacked() && _map1MonsterMan2.getHP() > 0) {
        _mhp[_map1MonsterMan2.getHP()].show();
        _mhp[_map1MonsterMan2.getHP()].show();
    }
    if(_map2Monster1.getBeAttacked() && _map2Monster1.getHP() > 0) {
```

```
        _fmhp[_map2Monster1.getHP()].show();
        _fmhp[_map2Monster1.getHP()].show();
    }
    if(_map2Monster2.getBeAttacked() && _map2Monster2.getHP() > 0) {
        _fmhp[_map2Monster2.getHP()].show();
        _fmhp[_map2Monster2.getHP()].show();
    }
    if(_king.getHP() > 0 && _kingShow) {
        _kingHp[_king.getHP()].show();
        _kingHp[_king.getHP()].show();
    }
    if(role.getHp() > 0) {
        _hp[role.getHp()].show();
    }
    //-----輸了-----
    if(role.getHp() == -1 && roleDead == 0){
        _losePhoto.show();
        restartButton.show();
        exitButton.show();
    }
    //-----贏了-----
    if(_king.getHP() == -1 && _kingDiedShine == 0 && role.getHp() > 0){
        _winPhoto.show();
        restartButton.show();
        exitButton.show();
        _music.stop();
        _kingBGM.stop();
        _winMedio.resume();
    }
}
@Override
public void release() {
    _scores.release();
    _music.release();
    _background.release();
    rightButton.release();
    leftButton.release();
    upButton.release();
    downButton.release();
    attackButton.release();
    smiteButton.release();
    nextMapGo.release();
    _test.release();
    _explosionRight.release();
    _kunaiLeft.release();
    _kunaiRight.release();
    //----第一張地圖----
    _map1Telephone.release();
    _map1TrafficCon.release();
    _map1Trashcan.release();
    //----第二張地圖
    _map2StreeLight.release();
    _map2RecycleBin.release();
    _map2Car.release();
    //----第一張地圖
    _map1TrafficConAttacked.release();
    _map1TrashCanAttacked.release();
    _map1TelephoneAttacked.release();
    //_map1MonsterManDeadRight.release();
    //_map1MonsterManDeadLeft.release();
    //----第二張地圖----
    _map2StreeLightAttacked.release();
    _map2RecycleBinAttacked.release();
    _map2CarAttacked.release();
    _scores = null;
    _music = null;
    _background = null;
    //----按鈕區----
    rightButton = null;
    leftButton = null;
```

```
    upButton = null;
    downButton = null;
    attackButton = null;
    smiteButton = null;
                nextMapGo = null;
    _map1Trashcan = null;
    _map1Telephone = null;
    _map1TrafficCon = null;
    _map2StreeLight = null;
    _map2RecycleBin = null;
    _explosionRight = null;
    _kunaiRight = null;
    _kunaiLeft = null;
    _map1TrashCanAttacked = null;
    _map1TrafficConAttacked = null;
    _map1TelephoneAttacked = null;
    _map2StreeLightAttacked = null;
    _map2RecycleBinAttacked = null;
    _map2CarAttacked = null;
    _map2Car = null;
    _hp = null;
}
@Override
public void keyPressed(int keyCode) {
    // TODO Auto-generated method stub
}
@Override
public void keyReleased(int keyCode) {
    // TODO Auto-generated method stub
}
@Override
public void orientationChanged(float pitch, float azimuth, float roll) {}
@Override
public void accelerationChanged(float dX, float dY, float dZ) {
    // TODO Auto-generated method stub
}
@Override
public boolean pointerPressed(Pointer actionPointer, List<Pointer> pointers) {
    _grabRight = rightButton.pointerPressed(actionPointer,pointers);   //判斷是否按到右鍵，是的話回傳 true
    _grabLeft = leftButton.pointerPressed(actionPointer,pointers);     //同上 *左鍵*
    _grabUp = upButton.pointerPressed(actionPointer,pointers);
    _grabDown = downButton.pointerPressed(actionPointer,pointers);
    _grabAttack = attackButton.pointerPressed(actionPointer,pointers);
    _grabSmite = smiteButton.pointerPressed(actionPointer,pointers);
    if((role.getHp() == -1 && roleDead == 0) || (_king.getHP() == -1 && _kingDiedShine == 0 && role.getHp() > 0)) {
        _grabRestart = restartButton.pointerPressed(actionPointer, pointers);
        _grabExit = exitButton.pointerPressed(actionPointer, pointers);
    }
    return true;
}
@Override
public boolean pointerMoved(Pointer actionPointer, List<Pointer> pointers) {
    _grabRight = rightButton.pointerPressed(actionPointer,pointers);   //判斷是否按到右鍵，是的話回傳 true
    _grabLeft = leftButton.pointerPressed(actionPointer,pointers);     //同上 *左鍵*
    _grabUp = upButton.pointerPressed(actionPointer,pointers);
    _grabDown = downButton.pointerPressed(actionPointer,pointers);
    _grabAttack = attackButton.pointerPressed(actionPointer,pointers);
    _grabSmite = smiteButton.pointerPressed(actionPointer,pointers);
    if(role.getHp() == -1 && roleDead == 0 || (_king.getHP() == -1 && _kingDiedShine == 0 && role.getHp() > 0)) {
        _grabRestart = restartButton.pointerPressed(actionPointer, pointers);
        _grabExit = exitButton.pointerPressed(actionPointer, pointers);
    }
    return false;
}
public void resizeAndroidIcon() {}
@Override
public boolean pointerReleased(Pointer actionPointer, List<Pointer> pointers) {
    if(_grabRight) {
        lastClickRightTime = System.currentTimeMillis();   //取得向右鍵放開時間
    }
```

```
        if(_grabLeft) {
            lastClickLeftTime = System.currentTimeMillis();    //取得向左鍵放開時間
        }
        _detectDoubleGrabAttack = false;
        _grab = false;
        _grabRight = false;
        _grabLeft = false;
        _grabUp = false;
        _grabDown = false;
        _grabAttack = false;
        _grabSmite = false;
        rightButton.pointerReleased(actionPointer,pointers);
        leftButton.pointerReleased(actionPointer,pointers);
        upButton.pointerReleased(actionPointer,pointers);
        downButton.pointerReleased(actionPointer,pointers);
        attackButton.pointerReleased(actionPointer,pointers);
        smiteButton.pointerReleased(actionPointer,pointers);
        restartButton.pointerReleased(actionPointer,pointers);
        exitButton.pointerReleased(actionPointer,pointers);
        return false;
    }
    @Override
    public void pause() {
        _music.pause();
    }

    @Override
    public void resume() {
        _music.resume();
    }
}
```

## Monster.java

```
package tw.edu.ntut.csie.game.state;
import tw.edu.ntut.csie.game.R;
import tw.edu.ntut.csie.game.core.MovingBitmap;
import tw.edu.ntut.csie.game.extend.Animation;
/**
 * Created by User on 2018/6/7.
 */
public class Monster{
    private Animation right = new Animation();
    private Animation left = new Animation();
    private Animation attackRight = new Animation();
    private Animation attackLeft = new Animation();
    private Animation remoteAttackRight = new Animation();
    private Animation remoteAttackLeft = new Animation();
    private Animation beAttackedRight = new Animation();
    private Animation beAttackedLeft = new Animation();
    private Animation deadRight = new Animation();
    private Animation deadLeft = new Animation();
    private int _x,_y,_hp = 0,_randMove = 0,_randMoveTime = 0,_mapNumber = 0,_dir = 1,_attackTime = 0,_dropSkill,_plugDied;   //掉落的技能 0
為血瓶 1 為苦無 2 為衝擊波 -1 為已掉落物品
    private boolean _beAttacked = false;
    public Monster(int x,int y,int hp,int type){
        _hp = hp;
        _x = x;
        _y = y;
        if(type == 1) {
            right.addFrame(R.drawable.m_run001);
            right.addFrame(R.drawable.m_run002);
            right.addFrame(R.drawable.m_run003);
            right.addFrame(R.drawable.m_run004);
            right.addFrame(R.drawable.m_run005);
            right.addFrame(R.drawable.m_run007);
            right.addFrame(R.drawable.m_run008);
            right.addFrame(R.drawable.m_run09);
```

```
        right.addFrame(R.drawable.m_run010);
        right.setDelay(1);
        left.addFrame(R.drawable.m_runl001);
        left.addFrame(R.drawable.m_runl002);
        left.addFrame(R.drawable.m_runl003);
        left.addFrame(R.drawable.m_runl004);
        left.addFrame(R.drawable.m_runl005);
        left.addFrame(R.drawable.m_runl007);
        left.addFrame(R.drawable.m_runl008);
        left.addFrame(R.drawable.m_runl09);
        left.addFrame(R.drawable.m_runl010);
        left.setDelay(1);
        attackRight.addFrame(R.drawable.m_attack002);
        attackRight.addFrame(R.drawable.m_attack003);
        attackRight.addFrame(R.drawable.m_attack004);
        attackRight.addFrame(R.drawable.m_attack005);
        attackRight.addFrame(R.drawable.m_attack008);
        attackRight.setDelay(2);
        attackLeft.addFrame(R.drawable.m_attackl002);
        attackLeft.addFrame(R.drawable.m_attackl003);
        attackLeft.addFrame(R.drawable.m_attackl004);
        attackLeft.addFrame(R.drawable.m_attackl005);
        attackLeft.addFrame(R.drawable.m_attackl008);
        attackLeft.setDelay(2);
        beAttackedRight.addFrame(R.drawable.hurt);
        beAttackedRight.setDelay(6);
        beAttackedLeft.addFrame(R.drawable.hurt_l);
        beAttackedLeft.setDelay(6);
        deadRight.addFrame(R.drawable.dead_001);
        deadRight.addFrame(R.drawable.dead_002);
        deadRight.addFrame(R.drawable.dead_003);
        deadRight.addFrame(R.drawable.dead_004);
        deadRight.addFrame(R.drawable.dead_007);
        deadRight.addFrame(R.drawable.dead_008);
        deadRight.addFrame(R.drawable.dead_011);
        deadRight.addFrame(R.drawable.dead_012);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.dead_012);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.dead_012);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.dead_012);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.dead_012);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.setDelay(2);
        deadLeft.addFrame(R.drawable.dead_l_001);
        deadLeft.addFrame(R.drawable.dead_l_002);
        deadLeft.addFrame(R.drawable.dead_l_003);
        deadLeft.addFrame(R.drawable.dead_l_004);
        deadLeft.addFrame(R.drawable.dead_l_007);
        deadLeft.addFrame(R.drawable.dead_l_008);
        deadLeft.addFrame(R.drawable.dead_l_011);
        deadLeft.addFrame(R.drawable.dead_l_012);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.dead_l_012);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.dead_l_012);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.dead_l_012);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.dead_l_012);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.setDelay(2);
    }
    else if(type == 2){
        right.addFrame(R.drawable.femalewalk_001);
        right.addFrame(R.drawable.femalewalk_002);
        right.addFrame(R.drawable.femalewalk_003);
        right.addFrame(R.drawable.femalewalk_004);
```

```
right.addFrame(R.drawable.femalewalk_005);
right.addFrame(R.drawable.femalewalk_006);
right.addFrame(R.drawable.femalewalk_007);
right.addFrame(R.drawable.femalewalk_008);
right.addFrame(R.drawable.femalewalk_009);
right.addFrame(R.drawable.femalewalk_010);
right.setDelay(1);
left.addFrame(R.drawable.femalewalk_l_001);
left.addFrame(R.drawable.femalewalk_l_002);
left.addFrame(R.drawable.femalewalk_l_003);
left.addFrame(R.drawable.femalewalk_l_004);
left.addFrame(R.drawable.femalewalk_l_005);
left.addFrame(R.drawable.femalewalk_l_006);
left.addFrame(R.drawable.femalewalk_l_007);
left.addFrame(R.drawable.femalewalk_l_008);
left.addFrame(R.drawable.femalewalk_l_009);
left.addFrame(R.drawable.femalewalk_l_010);
left.setDelay(1);
attackRight.addFrame(R.drawable.femaleattack_002);
attackRight.addFrame(R.drawable.femaleattack_003);
attackRight.addFrame(R.drawable.femaleattack_004);
attackRight.addFrame(R.drawable.femaleattack_005);
attackRight.addFrame(R.drawable.femaleattack_008);
attackRight.setDelay(2);
attackLeft.addFrame(R.drawable.femaleattack_l_002);
attackLeft.addFrame(R.drawable.femaleattack_l_003);
attackLeft.addFrame(R.drawable.femaleattack_l_004);
attackLeft.addFrame(R.drawable.femaleattack_l_005);
attackLeft.addFrame(R.drawable.femaleattack_l_008);
attackLeft.setDelay(2);
beAttackedRight.addFrame(R.drawable.femalehurt);
beAttackedRight.setDelay(6);
beAttackedLeft.addFrame(R.drawable.femalehurt_l);
beAttackedLeft.setDelay(6);
deadRight.addFrame(R.drawable.femaledead_001);
deadRight.addFrame(R.drawable.femaledead_002);
deadRight.addFrame(R.drawable.femaledead_003);
deadRight.addFrame(R.drawable.femaledead_004);
deadRight.addFrame(R.drawable.femaledead_007);
deadRight.addFrame(R.drawable.femaledead_008);
deadRight.addFrame(R.drawable.femaledead_010);
deadRight.addFrame(R.drawable.femaledead_011);
deadRight.addFrame(R.drawable.femaledead_012);
deadRight.addFrame(R.drawable.transparent);
deadRight.addFrame(R.drawable.femaledead_012);
deadRight.addFrame(R.drawable.transparent);
deadRight.addFrame(R.drawable.femaledead_012);
deadRight.addFrame(R.drawable.transparent);
deadRight.addFrame(R.drawable.femaledead_012);
deadRight.addFrame(R.drawable.transparent);
deadRight.addFrame(R.drawable.femaledead_012);
deadRight.addFrame(R.drawable.transparent);
deadRight.setDelay(2);
deadLeft.addFrame(R.drawable.femaledead_l_001);
deadLeft.addFrame(R.drawable.femaledead_l_002);
deadLeft.addFrame(R.drawable.femaledead_l_003);
deadLeft.addFrame(R.drawable.femaledead_l_004);
deadLeft.addFrame(R.drawable.femaledead_l_007);
deadLeft.addFrame(R.drawable.femaledead_l_008);
deadLeft.addFrame(R.drawable.femaledead_l_010);
deadLeft.addFrame(R.drawable.femaledead_l_011);
deadLeft.addFrame(R.drawable.femaledead_l_012);
deadLeft.addFrame(R.drawable.transparent);
deadLeft.addFrame(R.drawable.femaledead_l_012);
deadLeft.addFrame(R.drawable.transparent);
deadLeft.addFrame(R.drawable.femaledead_l_012);
deadLeft.addFrame(R.drawable.transparent);
deadLeft.addFrame(R.drawable.femaledead_l_012);
deadLeft.addFrame(R.drawable.transparent);
deadLeft.addFrame(R.drawable.femaledead_l_012);
```

```
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.setDelay(2);
    }
    if(type == 3){
        right.addFrame(R.drawable.king_run_001);
        right.addFrame(R.drawable.king_run_002);
        right.addFrame(R.drawable.king_run_003);
        right.addFrame(R.drawable.king_run_004);
        right.addFrame(R.drawable.king_run_005);
        right.addFrame(R.drawable.king_run_006);
        right.addFrame(R.drawable.king_run_007);
        right.addFrame(R.drawable.king_run_008);
        right.setDelay(1);
        left.addFrame(R.drawable.king_lrun_001);
        left.addFrame(R.drawable.king_lrun_002);
        left.addFrame(R.drawable.king_lrun_003);
        left.addFrame(R.drawable.king_lrun_004);
        left.addFrame(R.drawable.king_lrun_005);
        left.addFrame(R.drawable.king_lrun_006);
        left.addFrame(R.drawable.king_lrun_007);
        left.addFrame(R.drawable.king_lrun_008);
        left.setDelay(1);
        attackRight.addFrame(R.drawable.melee_002);
        attackRight.addFrame(R.drawable.melee_003);
        attackRight.addFrame(R.drawable.melee_004);
        attackRight.addFrame(R.drawable.melee_005);
        attackRight.addFrame(R.drawable.melee_006);
        attackRight.setDelay(2);
        attackLeft.addFrame(R.drawable.l_melee_002);
        attackLeft.addFrame(R.drawable.l_melee_003);
        attackLeft.addFrame(R.drawable.l_melee_004);
        attackLeft.addFrame(R.drawable.l_melee_005);
        attackLeft.addFrame(R.drawable.l_melee_006);
        attackLeft.setDelay(2);
        remoteAttackRight.addFrame(R.drawable.shoot_001);
        remoteAttackRight.addFrame(R.drawable.shoot_002);
        remoteAttackRight.addFrame(R.drawable.shoot_003);
        remoteAttackRight.addFrame(R.drawable.shoot_004);
        remoteAttackRight.setDelay(3);
        remoteAttackLeft.addFrame(R.drawable.lshoot_001);
        remoteAttackLeft.addFrame(R.drawable.lshoot_002);
        remoteAttackLeft.addFrame(R.drawable.lshoot_003);
        remoteAttackLeft.addFrame(R.drawable.lshoot_004);
        remoteAttackLeft.setDelay(3);
        beAttackedRight.addFrame(R.drawable.hurt_king);
        beAttackedRight.setDelay(6);
        beAttackedLeft.addFrame(R.drawable.lhurt_king);
        beAttackedLeft.setDelay(6);
        deadRight.addFrame(R.drawable.king_dead000);
        deadRight.addFrame(R.drawable.king_dead001);
        deadRight.addFrame(R.drawable.king_dead002);
        deadRight.addFrame(R.drawable.king_dead003);
        deadRight.addFrame(R.drawable.king_dead004);
        deadRight.addFrame(R.drawable.king_dead005);
        deadRight.addFrame(R.drawable.king_dead006);
        deadRight.addFrame(R.drawable.king_dead007);
        deadRight.addFrame(R.drawable.king_dead008);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.king_dead008);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.king_dead008);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.king_dead008);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.addFrame(R.drawable.king_dead008);
        deadRight.addFrame(R.drawable.transparent);
        deadRight.setDelay(3);
        deadLeft.addFrame(R.drawable.lking_dead000);
        deadLeft.addFrame(R.drawable.lking_dead001);
        deadLeft.addFrame(R.drawable.lking_dead002);
```

```
        deadLeft.addFrame(R.drawable.lking_dead003);
        deadLeft.addFrame(R.drawable.lking_dead004);
        deadLeft.addFrame(R.drawable.lking_dead005);
        deadLeft.addFrame(R.drawable.lking_dead006);
        deadLeft.addFrame(R.drawable.lking_dead007);
        deadLeft.addFrame(R.drawable.lking_dead008);
        deadLeft.addFrame(R.drawable.lking_dead009);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.lking_dead009);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.lking_dead009);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.lking_dead009);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.addFrame(R.drawable.lking_dead009);
        deadLeft.addFrame(R.drawable.transparent);
        deadLeft.setDelay(3);
    }
}
public void restart(int x,int y,int hp,int dir,int randMove,int randMoveTime,int mapNumber,int dropSkill){
    _x = x;
    _y = y;
    _hp = hp;
    _dir = dir;
    _randMove = randMove;
    _randMoveTime = randMoveTime;
    _mapNumber = mapNumber;
    _dropSkill  = dropSkill;
    setXY(_x,_y);
    setAttackRightCurrentFrame(1);
    setAttackLefttCurrentFrame(1);
    setDeadRightCurrentFrame(1);
    setDeadLefttCurrentFrame(1);
    _beAttacked = false;
    _attackTime = 0;
}
public void addRight(int resId) { right.addFrame(resId);}
public void setRightDelay(int delay) {right.setDelay(delay);}
public void addLeft(int resId) { left.addFrame(resId);}
public void setLeftDelay(int delay) {left.setDelay(delay);}
public void addAttackRight(int resId) {attackRight.addFrame(resId);}
public void setAttackRightDelay(int delay) { attackRight.addFrame(delay);}
public void addAttackLeft(int resId) {attackLeft.addFrame(resId);}
public void setAttackLeftDelayy(int delay) { attackLeft.setDelay(delay);}
public void addBeAttackRight(int resId) {beAttackedRight.addFrame(resId);}
public void setBeAttackedRight(int delay) { beAttackedRight.setDelay(delay);}
public void addBeAttackLeft(int resId) {beAttackedLeft.addFrame(resId);}
public void setBeAttackedLeft(int delay) {beAttackedLeft.setDelay(delay);}
public void moveRight() { right.move();}
public void moveLeft() { left.move();}
public void moveAttackRight() { attackRight.move();}
public void moveAttackLeft() { attackLeft.move();}
public void moveRemoteAttackRight() { remoteAttackRight.move();}
public void moveRemoteAttackLeft() { remoteAttackLeft.move();}
public void moveBeAttackedRight() { beAttackedRight.move();}
public void moveBeAttackedLeft() { beAttackedLeft.move();}
public void moveDeadRight() {deadRight.move();}
public void moveDeadLeft() {deadLeft.move();}
public void showRight() { right.show();}
public void showLeft() { left.show();}
public void showAttackRight() { attackRight.show();}
public void showAttackLeft() { attackLeft.show();}
public void showRemoteAttackRight() { remoteAttackRight.show();}
public void showRemoteAttackLeft() { remoteAttackLeft.show();}
public void showBeAttackedRight() { beAttackedRight.show();}
public void showBeAttackedLeft() { beAttackedLeft.show();}
public void showDeadRight() {deadRight.show();}
public void showDeadLeft() {deadLeft.show();}
public void setXY(int x,int y){
    _x = x;
```

```
        _y = y;
        right.setLocation(_x,_y);
        left.setLocation(_x,_y);
        attackRight.setLocation(_x,_y);
        attackLeft.setLocation(_x,_y);
        remoteAttackRight.setLocation(_x,_y);
        remoteAttackLeft.setLocation(_x,_y);
        beAttackedRight.setLocation(_x,_y);
        beAttackedLeft.setLocation(_x,_y);
        deadRight.setLocation(_x,_y);
        deadLeft.setLocation(_x,_y);
    }
    public int getX(){ return _x;}
    public int getY(){return _y;}
    public int getHeight(){ return right.getHeight();}
    public int getWidth(){ return right.getWidth();}
    public void setHP(int hp){ _hp = hp;}
    public int getHP(){ return _hp;}
    public void setRandMove(int randMove) { _randMove = randMove;}
    public int getRandMove() { return _randMove;}
    public void setRandMoveTime(int randMoveTime) { _randMoveTime = randMoveTime;}
    public int getRandMoveTime() { return _randMoveTime;}
    public void setDir(int dir) { _dir = dir;}
    public int getDir() {return _dir;}
    public void setMapNumber(int mapNumber) { _mapNumber = mapNumber;}
    public int getMapNumber() {return _mapNumber;}
    public void setAttackTime(int attackTime){_attackTime = attackTime;}
    public int getAttackTime(){return _attackTime;}
    public void setBeAttacked(boolean beAttacked){_beAttacked = beAttacked;}
    public boolean getBeAttacked(){return _beAttacked;}
    public void setAttackRightCurrentFrame(int index) { attackRight.setCurrentFrameIndex(index);}
    public void setAttackLefttCurrentFrame(int index) { attackLeft.setCurrentFrameIndex(index);}
    public void setDeadRightCurrentFrame(int index) { deadRight.setCurrentFrameIndex(index);}
    public void setDeadLefttCurrentFrame(int index) { deadLeft.setCurrentFrameIndex(index);}
    public void setDropSkill(int skill) { _dropSkill = skill;}
    public int getDropSkill() { return _dropSkill;}
    public void setPlugDied(int died) { _plugDied = died;}
    public int getPlugDied() { return _plugDied;}
}
```

## Role.java

```
package tw.edu.ntut.csie.game.state;
import java.util.ArrayList;
import tw.edu.ntut.csie.game.R;
import tw.edu.ntut.csie.game.core.MovingBitmap;
import tw.edu.ntut.csie.game.extend.Animation;
/**
 * Created by User on 2018/6/6.
 */
public class Role{
    private Animation roleRight = new Animation();
    private Animation roleLeft = new Animation();
    private Animation roleAttackRight = new Animation();
    private Animation roleAttackLeft = new Animation();
    private Animation roleSmiteRight = new Animation();
    private Animation roleSmiteLeft = new Animation();
    private Animation roleSildeRight = new Animation();
    private Animation roleSlideLeft = new Animation();
    private Animation roleBeAttackedRight = new Animation();
    private Animation roleBeAttackedLeft = new Animation();
    private Animation roleDeadRight = new Animation();
    private Animation roleDeadLeft = new Animation();
    private int _hp,_x,_y;
    private boolean _beAttacked = false;
    private boolean _skillKunai = false;
    private boolean _skillShockWave = false;
    public Role(){
```

```
roleRight.addFrame(R.drawable.run_000);
roleRight.addFrame(R.drawable.run_001);
roleRight.addFrame(R.drawable.run_002);
roleRight.addFrame(R.drawable.run_003);
roleRight.addFrame(R.drawable.run_004);
roleRight.addFrame(R.drawable.run_005);
roleRight.addFrame(R.drawable.run_007);
roleRight.addFrame(R.drawable.run_008);
roleRight.addFrame(R.drawable.run_009);
roleRight.setDelay(1);
roleLeft.addFrame(R.drawable.runleft_000);
roleLeft.addFrame(R.drawable.runleft_001);
roleLeft.addFrame(R.drawable.runleft_002);
roleLeft.addFrame(R.drawable.runleft_003);
roleLeft.addFrame(R.drawable.runleft_004);
roleLeft.addFrame(R.drawable.runleft_005);
roleLeft.addFrame(R.drawable.runleft_007);
roleLeft.addFrame(R.drawable.runleft_008);
roleLeft.addFrame(R.drawable.runleft_009);
roleLeft.setDelay(1);
roleAttackRight.addFrame(R.drawable.attack__000);
roleAttackRight.addFrame(R.drawable.attack__001);
roleAttackRight.addFrame(R.drawable.attack__002);
roleAttackRight.addFrame(R.drawable.attack__003);
roleAttackRight.addFrame(R.drawable.attack__004);
roleAttackRight.addFrame(R.drawable.attack__005);
roleAttackRight.addFrame(R.drawable.attack__006);
roleAttackRight.addFrame(R.drawable.attack__007);
roleAttackRight.addFrame(R.drawable.attack__008);
roleAttackRight.addFrame(R.drawable.attack__009);
roleAttackRight.setDelay(1);
roleAttackLeft.addFrame(R.drawable.attackleft__000);
roleAttackLeft.addFrame(R.drawable.attackleft__001);
roleAttackLeft.addFrame(R.drawable.attackleft__002);
roleAttackLeft.addFrame(R.drawable.attackleft__003);
roleAttackLeft.addFrame(R.drawable.attackleft__004);
roleAttackLeft.addFrame(R.drawable.attackleft__005);
roleAttackLeft.addFrame(R.drawable.attackleft__006);
roleAttackLeft.addFrame(R.drawable.attackleft__007);
roleAttackLeft.addFrame(R.drawable.attackleft__008);
roleAttackLeft.addFrame(R.drawable.attackleft__009);
roleAttackLeft.setDelay(1);
roleSmiteRight.addFrame(R.drawable.throw__000);
roleSmiteRight.addFrame(R.drawable.throw__001);
roleSmiteRight.addFrame(R.drawable.throw__002);
roleSmiteRight.addFrame(R.drawable.throw__003);
roleSmiteRight.addFrame(R.drawable.throw__004);
roleSmiteRight.addFrame(R.drawable.throw__005);
roleSmiteRight.addFrame(R.drawable.throw__006);
roleSmiteRight.addFrame(R.drawable.throw__007);
roleSmiteRight.addFrame(R.drawable.throw__008);
roleSmiteRight.addFrame(R.drawable.throw__009);
roleSmiteRight.setDelay(1);
roleSmiteLeft.addFrame(R.drawable.throwleft__000);
roleSmiteLeft.addFrame(R.drawable.throwleft__001);
roleSmiteLeft.addFrame(R.drawable.throwleft__002);
roleSmiteLeft.addFrame(R.drawable.throwleft__003);
roleSmiteLeft.addFrame(R.drawable.throwleft__004);
roleSmiteLeft.addFrame(R.drawable.throwleft__005);
roleSmiteLeft.addFrame(R.drawable.throwleft__006);
roleSmiteLeft.addFrame(R.drawable.throwleft__007);
roleSmiteLeft.addFrame(R.drawable.throwleft__008);
roleSmiteLeft.addFrame(R.drawable.throwleft__009);
roleSmiteLeft.setDelay(1);
roleSildeRight.addFrame(R.drawable.slide__000);
roleSildeRight.addFrame(R.drawable.slide__001);
roleSildeRight.addFrame(R.drawable.slide__002);
roleSildeRight.addFrame(R.drawable.slide__003);
roleSildeRight.addFrame(R.drawable.slide__004);
roleSildeRight.addFrame(R.drawable.slide__005);
```

```
        roleSildeRight.addFrame(R.drawable.slide__006);
        roleSildeRight.addFrame(R.drawable.slide__007);
        roleSildeRight.addFrame(R.drawable.slide__008);
        roleSildeRight.addFrame(R.drawable.slide__009);
        roleSildeRight.setDelay(1);
        roleSlideLeft.addFrame(R.drawable.slideleft__000);
        roleSlideLeft.addFrame(R.drawable.slideleft__001);
        roleSlideLeft.addFrame(R.drawable.slideleft__002);
        roleSlideLeft.addFrame(R.drawable.slideleft__003);
        roleSlideLeft.addFrame(R.drawable.slideleft__004);
        roleSlideLeft.addFrame(R.drawable.slideleft__005);
        roleSlideLeft.addFrame(R.drawable.slideleft__006);
        roleSlideLeft.addFrame(R.drawable.slideleft__007);
        roleSlideLeft.addFrame(R.drawable.slideleft__008);
        roleSlideLeft.addFrame(R.drawable.slideleft__009);
        roleSlideLeft.setDelay(1);
        roleBeAttackedRight.addFrame(R.drawable.rightattacked_001);
        roleBeAttackedRight.addFrame(R.drawable.rightattacked_001);
        roleBeAttackedRight.setDelay(1);
        roleBeAttackedLeft.addFrame(R.drawable.leftattacked_001);
        roleBeAttackedLeft.addFrame(R.drawable.leftattacked_001);
        roleBeAttackedLeft.setDelay(1);
        roleDeadRight.addFrame(R.drawable.died__001);
        roleDeadRight.addFrame(R.drawable.died__002);
        roleDeadRight.addFrame(R.drawable.died__003);
        roleDeadRight.addFrame(R.drawable.died_004);
        roleDeadRight.addFrame(R.drawable.died_005);
        roleDeadRight.addFrame(R.drawable.died_006);
        roleDeadRight.addFrame(R.drawable.died_007);
        roleDeadRight.addFrame(R.drawable.died_008);
        roleDeadRight.addFrame(R.drawable.died_009);
        roleDeadRight.setDelay(2);
        roleDeadLeft.addFrame(R.drawable.dead__l001);
        roleDeadLeft.addFrame(R.drawable.dead__l002);
        roleDeadLeft.addFrame(R.drawable.dead__l003);
        roleDeadLeft.addFrame(R.drawable.dead__l004);
        roleDeadLeft.addFrame(R.drawable.dead__l005);
        roleDeadLeft.addFrame(R.drawable.dead__l006);
        roleDeadLeft.addFrame(R.drawable.dead__l007);
        roleDeadLeft.addFrame(R.drawable.dead__l008);
        roleDeadLeft.addFrame(R.drawable.dead__l009);
        roleDeadLeft.setDelay(2);
    }
    public void restart(){
        _hp = 10;
        _skillKunai = false;
        _skillShockWave = false;
        setAttackRightCurrentFrame(1);
        setAttackLefttCurrentFrame(1);
        setSmiteLefttCurrentFrame(1);
        setSmiteRightCurrentFrame(1);
    }
    public void moveRight() {roleRight.move();}
    public void moveLeft() { roleLeft.move();}
    public void moveAttackRight() { roleAttackRight.move();}
    public void moveAttackLeft() { roleAttackLeft.move();}
    public void moveSmiteRight() { roleSmiteRight.move();}
    public void moveSmiteLeft() { roleSmiteLeft.move();}
    public void moveSlideRight() { roleSildeRight.move();}
    public void moveSlideLeft() { roleSlideLeft.move();}
    public void moveBeAttackedRight() { roleBeAttackedRight.move();}
    public void moveBeAttackedLeft() { roleBeAttackedLeft.move();}
    public void moveDeadRight() {roleDeadRight.move();}
    public void moveDeadLeft() {roleDeadLeft.move();}
    public void showRight() { roleRight.show();}
    public void showLeft() { roleLeft.show();}
    public void showAttackRight() { roleAttackRight.show();}
    public void showAttackLeft() { roleAttackLeft.show();}
    public void showSmiteRight() { roleSmiteRight.show();}
    public void showSmiteLeft() { roleSmiteLeft.show();}
```

```
    public void showSlideRight() { roleSildeRight.show();}
    public void showSlideLeft() { roleSlideLeft.show();}
    public void showBeAttackedRight() { roleBeAttackedRight.show();}
    public void showBeAttackedLeft() { roleBeAttackedLeft.show();}
    public void showDeadRight() {roleDeadRight.show();}
    public void showDeadLeft() {roleDeadLeft.show();}
    public void setXY(int x,int y){
        _x = x;
        _y = y;
        roleRight.setLocation(_x,_y);
        roleLeft.setLocation(_x,_y);
        roleAttackRight.setLocation(_x,_y);
        roleAttackLeft.setLocation(_x,_y);
        roleSmiteRight.setLocation(_x,_y);
        roleSmiteLeft.setLocation(_x,_y);
        roleSildeRight.setLocation(_x,_y);
        roleSlideLeft.setLocation(_x,_y);
        roleBeAttackedRight.setLocation(_x,_y);
        roleBeAttackedLeft.setLocation(_x,_y);
        roleDeadRight.setLocation(_x,_y);
        roleDeadLeft.setLocation(_x,_y);
    }
    public int getX() { return _x;}
    public int getY() { return  _y;}
    public int getWidth() { return roleRight.getWidth();}
    public int getHeight() { return roleRight.getHeight();}
    public void setAttackRightCurrentFrame(int index) { roleAttackRight.setCurrentFrameIndex(index);}
    public void setAttackLefttCurrentFrame(int index) { roleAttackLeft.setCurrentFrameIndex(index);}
    public void setSmiteRightCurrentFrame(int index) { roleSmiteRight.setCurrentFrameIndex(index);}
    public void setSmiteLefttCurrentFrame(int index) { roleSmiteLeft.setCurrentFrameIndex(index);}
    public void setHP(int hp){
        _hp = hp;
    }
    public int getHp(){
        return _hp;
    }
    public void setBeAttacked(boolean beAttacked){_beAttacked = beAttacked;}
    public boolean getBeAttacked(){return _beAttacked;}
    public void setSkillKunai(boolean bol) { _skillKunai = bol;}
    public boolean getSkillKunai() { return _skillKunai;}
    public void setSkillShockWave(boolean bol) { _skillShockWave = bol;}
    public boolean getSkillShockWave() { return _skillShockWave;}
}
```

## Dropitem.java

```
package tw.edu.ntut.csie.game.state;
import tw.edu.ntut.csie.game.core.MovingBitmap;
/**
 * Created by NTUTCSIE on 2018/6/15.
 */
public class DropItem{
    private MovingBitmap item;
    private boolean _onshow = false;
    private int _x,_y,_type,_pickupTime;    //掉落的技能 0 為血瓶 1 為苦無 2 為衝擊波

    public DropItem(int resId){
        item = new MovingBitmap(resId);
    }
    public void setXY(int x,int y) {
        _x=x;
        _y=y;
        item.setLocation(x,y);
    }
    public int getX() {return _x;}
    public int getY() {return _y;}
    public int getHeight() {return item.getHeight();}
    public int getWidth() {return item.getWidth();}
    public void setOnShow(boolean onshow){ _onshow = onshow;}
```

```
    public boolean getOnShow(){ return _onshow;}
    public void setType(int type) { _type = type;}
    public int getType() { return _type;}
    public void show() {item.show();}
    public void setPickupTime(int pickup) { _pickupTime = pickup;}
    public int getPickupTime() { return _pickupTime;}
}
```