國立臺北科技大學

2020 Spring 資工系物件導向程式實習

期末報告

遊戲名稱(TwinShot)

第 6 組

107820004 上官昌華

107820017 李昆哲

目錄

一、 簡介

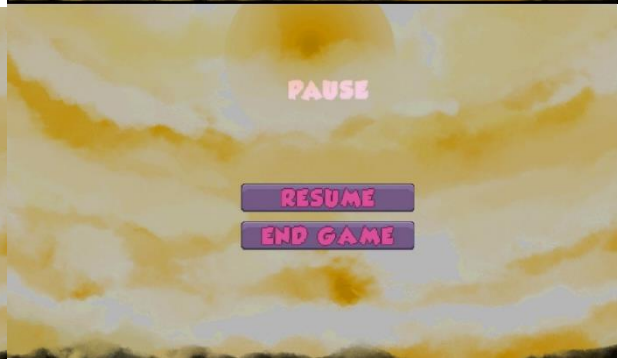
1. 動機: 小時候最常玩的遊戲之一，想利用所學重現它。
2. 分工:

	負責部分
李昆哲	程式開發、少部分截圖及美術
上官昌華	大部分截圖及美術，音效，關卡地圖建置

二、 遊戲介紹

1. 遊戲說明: 滑動、點擊螢幕來移動、跳躍、攻擊，殺死所有怪物即至下一關；共有三條命，碰到怪物一次會扣一條命；密技：重置角色位置，快速通關(殺死所有怪物)。

2. 遊戲圖形:



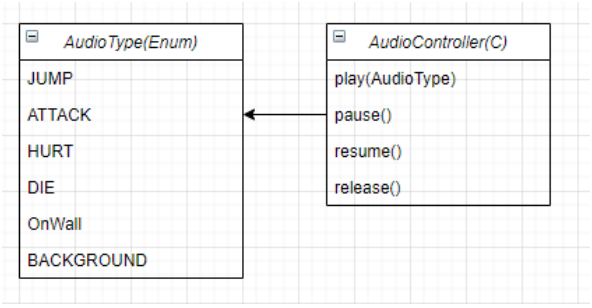
3. 遊戲音效:

跳躍	https://taira-komori.jpn.org/game01tw.html (jump 04)
攻擊	https://taira-komori.jpn.org/attack01tw.html (刺)
箭碰到牆	https://taira-komori.jpn.org/attack01tw.html (損壞 3)
扣血	https://sc.chinaz.com/yinxiao/160501185873.htm
死亡	https://www.youtube.com/watch?v=HOf4VqGkRys&ab_channel=MusicSoundEffects

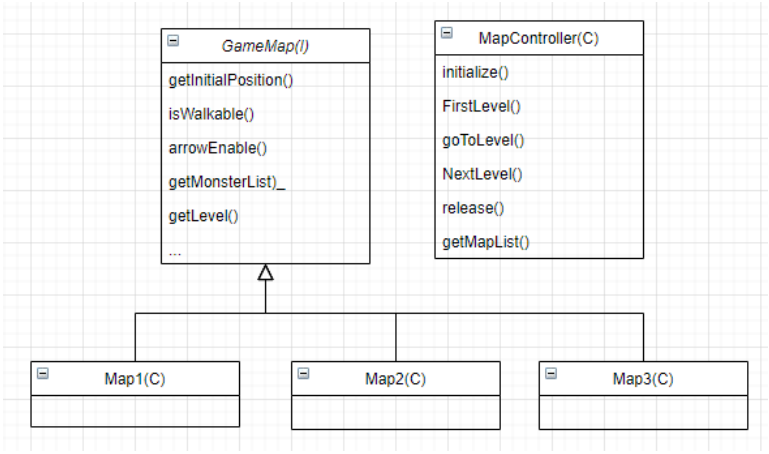
三、 程式設計

1. 程式架構:

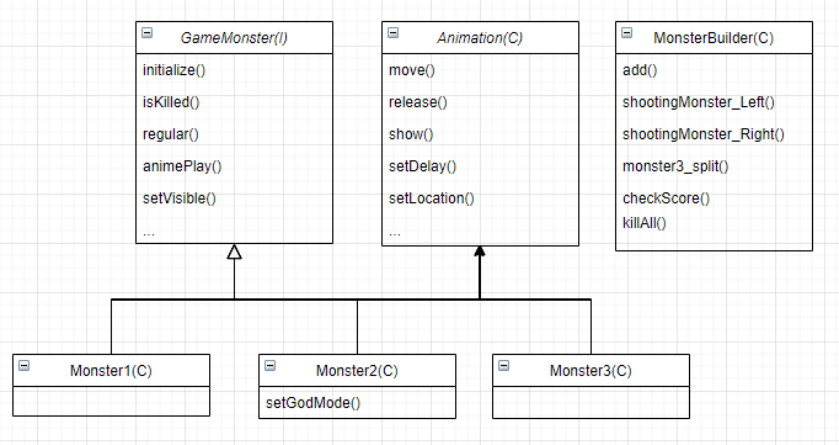
音效:利用 Enum 創建音效列表，用 controller 控制



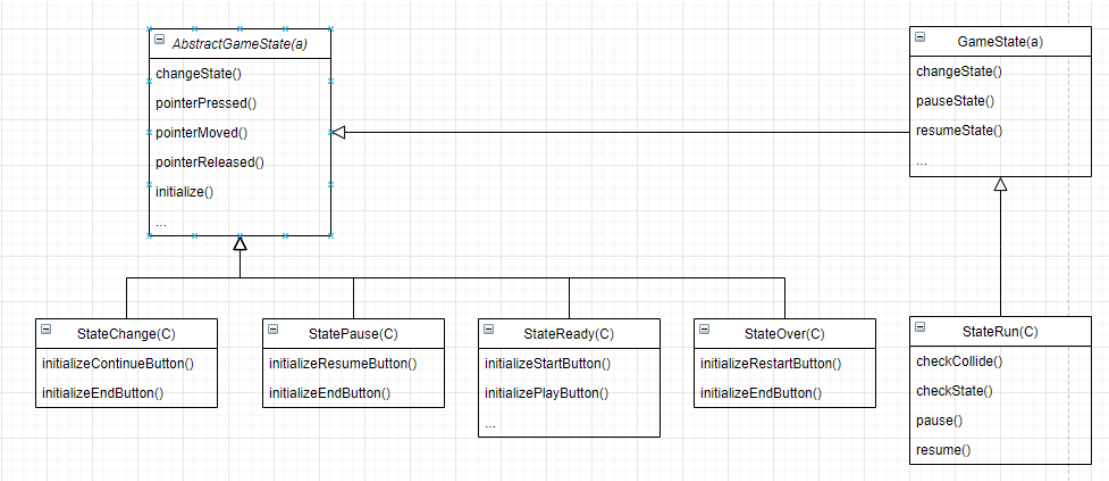
地圖:3 個關卡各繼承 GameMap，用 MapController 控制關卡



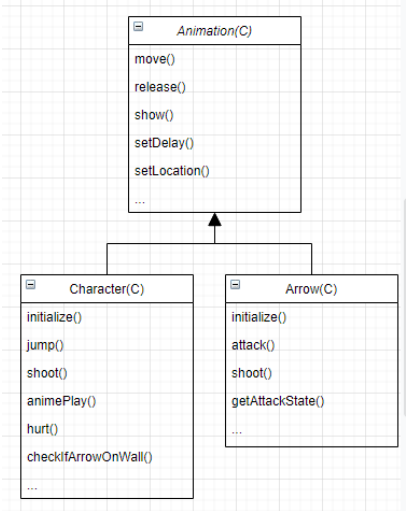
怪物:三種怪物，各繼承 GameMonster，也繼承 FrameWork 中的 Animation，再利用 MonsterBuilder 控制擊殺怪物，分數確認，產生分裂怪物，金手指(擊殺全部怪物)



遊戲狀態:在原有的 FrameWork 中的 AbstractGameState 中加入 StateChange 以及 StatePause，以及修改用於主要畫面呈現的 StateRun



角色與箭:建立主要角色以及跳躍、攻擊、站在箭上等功能



2. 程式類別:

類別名稱	.java 檔行數	說明
AudioController	71	控制音效播放
AudioType	10	音效類別
GameMap	25	關卡地圖介面
Map1	197	關卡 1
Map2	248	關卡 2
Map3	244	關卡 3
MapController	58	控制關卡轉換時進入的關卡
GameMonster	21	怪物介面
Monster1	201	怪物 1
Monster2	235	需攻擊兩次的怪物 2
Monster3	214	怪物 3，由 Builder 創造分裂效果
MonsterBuilder	188	用於建立怪物，確認分數，檢查怪物擊殺
Arrow	188	攻擊手段(箭)的相關功能
Character	388	角色的跳躍、移動、攻擊
StateChange	87	狀態切換(關卡切換)
StatePause	86	暫停狀態
StateRun	495	主要畫面呈現
StateReady	235	初始畫面
StateOver	76	死掉畫面
總行數	3267	

3. 程式技術:

套用了 design pattern 中的 Builder Pattern，讓 StateRun 不需要過度依賴其他的 Class，可以透過取用各 Controller 或是 Builder 來控制所有需要的元件，且讓程式盡量符合 DIP 原則。

四、 結語

1. 問題及解決方法: 在初期我們嘗試過使用網頁版進行開發，但是鑒於 debug 較為困難，因此後來我們改用 android studio 進行開發，android studio 版本太新好像也會有各式各樣的問題，需要一直尋找解決辦法，因此最後我們是用 4.0 做開發。同時間我們也在學習 java，因此可以邊學邊用，開發到後來很容易因為各種原因發生黑屏，因此我們採用 Builder Pattern 來幫助我們讓 class 之間的依賴性降低，同時設中斷點尋找原因，這讓我們在開發過程中的 debug 找問題找得很快，除此之外遇到的最大問題應該是 Timer 跟 TimerTask 的應用，因為很多的功能需要間隔固定時間去執行，我們查到可以利用 Timer 來實現，但是同一個 class 之內如果 new 了兩個 Timer 會報錯，且其他方法像是 Thread 都是會把整個遊戲的畫面暫停，而這不是我們想要的結果，還好最後有發現 framework 中的 move 跟 show 函式就有自帶間隔一小段時間重複執行的功能，才能順利的把各種功能開發出來。

2. 時間表:

週次	上官昌華 (小時)	李昆哲 (小時)	說明
1 (3/05)	5	5	Git tutorial
2 (3/12)	3	3	Game tutorial
3 (3/19)	5	5	素材收集
4 (3/26)	10	10	素材收集、角色移動
5 (4/09)	17	21	怪物、跳躍、動畫
6 (4/16)	10	11.5	關卡 1
7 (4/23)	10.5	10.5	關卡 2
8 (4/30)	8.5	8.5	關卡 3
9 (5/07)	10	10	角色、怪物功能完善
10 (5/14)	9	9	攻擊完善、暫停功能
11 (5/21)	6	6	新怪物、bug 修復
12 (5/28)	6	6	Bug 修復、程式優化
13 (6/04)	5	5	程式優化
14 (6/11)	5	5	音效、加強完整度
15 (6/18)	5	5	背景音樂
總和	115	120.5	

3. 貢獻比例: 各 50%

4. 自我檢核表:

週次	項目	完成否	無法完成原因
1	解決 Memort leak	已完成	
2	自訂遊戲 Icon	已完成	
3	有 About 畫面	已完成	
4	初始畫面說明按鍵	已完成	
5	之用法與密技	已完成	
6	上傳 apk 檔	已完成	
7	apk 可正確執行	已完成	
8	報告字型、點數、對齊	已完成	
9	行距、頁碼等格式正確	已完成	
10	全螢幕啟動	已完成	

5. 收穫:

李昆哲: 熟悉 Android Studio，Java 語言更加熟練，學到 timer、thread、synchronize 等概念，以及降低依賴性及設中斷點等。

上官昌華: 熟悉 Android Studio、Java 語言(interface、extend)

6. 心得:

李昆哲: 因為同時間也在學習 Java，因此能邊學邊應用是一個很好的體驗，且每個禮拜丟了一堆時間進去，真的是必要的，可以有效的精進自己所選的程式語言的熟練度，最後看到成品慢慢成形時，其實還滿開心的。

上官昌華: 因為沒有找到這個遊戲的素材，所以內容都是一個個從原遊戲截下的，耗費了不少時間。原本是打算用 windows 做，不過我和另一個組員都不熟悉語法，就改成了 android studio。在製作遊戲的過程中也是學到了很多 java 相關的語法，收穫蠻多的。

7. 建議:無

附錄

```
// AudioController.java
public class AudioController {
    private Audio hurt, jump, attack, die, onWall, background;
    private List<Audio> audioList;

    public AudioController(){
        hurt = new Audio(R.raw.hurt);
        jump = new Audio(R.raw.jump);
        attack = new Audio(R.raw.shoot);
        die = new Audio(R.raw.gameover);
        onWall = new Audio(R.raw.towall);
        background = new Audio(R.raw.ntut);
        createAudioList();
    }

    public void play(AudioType audioType){ //播放指定音效
        switch (audioType){
            case HURT:
                hurt.play();
                break;
            case JUMP:
                jump.play();
                break;
            case ATTACK:
                attack.play();
                break;
            case DIE:
                die.play();
                break;
            case OnWall:
                onWall.play();
                break;
            case BACKGROUND:
                background.setRepeating(true);
                background.play();
                break;
        }
    }

    public void release(){ //釋放記憶體
        for(Audio audio: audioList){
            audio.release();
        }
    }

    public void pause(){ //暫停播放
        background.pause();
    }

    public void resume(){ //恢復播放
        background.resume();
    }

    private void createAudioList(){ //建立音效列表
        audioList = new ArrayList<>();
        audioList.add(hurt);
        audioList.add(jump);
        audioList.add(attack);
        audioList.add(die);
        audioList.add(onWall);
        audioList.add(background);
    }
}
```

```java
// AudioType.java
public enum AudioType { //音效種類
	JUMP,
	ATTACK,
	HURT,
	DIE,
	OnWall,
	BACKGROUND
}

// GameMap.java
public interface GameMap extends GameObject { //關卡地圖介面
	public int getInitialPositionX();
	public int getInitialPositionY();
	public boolean isWalkable_down(int x, int y);
	public boolean isWalkable_up(int x, int y);
	public boolean isWalkable_right(int x, int y);
	public boolean isWalkable_left(int x, int y);
	public boolean arrowEnable_left(int x, int y, Arrow arrow);
	public boolean arrowEnable_right(int x, int y, Arrow arrow);
	public List<GameMonster> getMonsterList();
	public MonsterBuilder getMonsterBuilder();
	public int getLevel();
	public default boolean superJump(int x, int y){ return false; }
}

//Map1.java
public class Map1 implements GameObject, GameMap {
	private MovingBitmap block, block1, block2;
	private MovingBitmap pillar, pillar1, pillar2, pillar3;
	private MovingBitmap scores;
	private MonsterBuilder monsterBuilder;
	private int[][] map = {{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}, //地圖陣列
			{0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,3,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,1,3,2,2,3,1,2,3,1,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0},
			{0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0},
			{0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0},
			{0,8,0,0,0,0,0,0,0,0,1,3,1,1,2,2,1,3,1,0,0,0,0,0,0,0,0,8,0},
			{0,8,0,0,0,0,0,0,0,0,6,0,0,6,0,6,0,0,6,0,0,0,0,0,0,0,0,8,0},
			{0,1,3,1,3,1,3,2,1,2,0,0,0,0,0,0,0,0,0,1,1,3,3,1,2,2,3,1,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
			{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};

	//大地圖左上角座標
	private final int X = 0;
	private final int Y = 0;
	//小地圖寬度與高度
	private final int MW = 23;
	private final int MH = 23;

	public Map1(){
		block = new MovingBitmap(R.drawable.block);
		block1 = new MovingBitmap(R.drawable.block1);
		block2 = new MovingBitmap(R.drawable.block2);
		pillar = new MovingBitmap(R.drawable.pillar);
		pillar1 = new MovingBitmap(R.drawable.pillar1);
		pillar2 = new MovingBitmap(R.drawable.pillar2);
		pillar3 = new MovingBitmap(R.drawable.pillar3);
		scores = new MovingBitmap(R.drawable.scores);
		monsterBuilder = new MonsterBuilder();
		monsterBuilder.add(1, 80,435, 290, 323);
		monsterBuilder.add(1, 80,43, 290, 323);
```

```
        scores.setLocation(453,348);
    }

    @Override
    public int getInitialPositionX(){ //回傳人物初始位置
        return 300;
    }

    @Override
    public int getInitialPositionY() { //回傳人物初始位置
        return 230;
    }

    @Override
    public void release(){ //釋放記憶體
        block.release();
        block1.release();
        block2.release();
        pillar.release();
        pillar1.release();
        pillar2.release();
        pillar3.release();
        monsterBuilder.release();
        scores.release();

        block = null;
        block1 = null;
        block2 = null;
        pillar = null;
        pillar1 = null;
        pillar2 = null;
        pillar3 = null;
        scores = null;
    }

    @Override
    public void move(){ //使怪物規律移動
        monsterBuilder.move();
    }

    @Override
    public void show(){ //依照地圖陣列顯示地圖
        scores.show();
        for(int i = 0; i < 17; i++){
            for(int j = 0; j<29; j++){
                switch(map[i][j]){
                    case 1:
                        block.setLocation(X+(MW*j), Y+(MH*i));
                        block.show();
                        break;
                    case 2:
                        block1.setLocation(X+(MW*j), Y+(MH*i));
                        block1.show();
                        break;
                    case 3:
                        block2.setLocation(X+(MW*j), Y+(MH*i));
                        block2.show();
                        break;
                    case 4:
                        pillar.setLocation(X+(MW*j), Y+(MH*i));
                        pillar.show();
                        break;
                    case 5:
                        pillar1.setLocation(X+(MW*j), Y+(MH*i));
                        pillar1.show();
                        break;
                    case 6:
                        pillar2.setLocation(X+(MW*j), Y+(MH*i));
                        pillar2.show();
```

```
                        break;
                    case 7:
                        pillar3.setLocation(X+(MW*j), Y+(MH*i));
                        pillar3.show();
                        break;
                    default:
                        break;
                }
            }
        }
        monsterBuilder.show();
    }


    public boolean isWalkable_right(int x, int y){ //確認是否可以行走
        if (x > 35 && x < 587) {
            if (x >= 200 && x <= 430 && y < 322-43 && y > 276-43){
                return false;
            }else {
                return true;
            }
        }else {
            return false;
        }
    }
    public boolean isWalkable_left(int x, int y){
        if (x > 35 && x < 587) {
            if (x > 200 && x <= 437 && y < 322-43 && y > 276-43) {
                return false;
            }else {
                return true;
            }
        }else {
            return false;
        }
    }
    public boolean isWalkable_down(int x, int y){
        if ((x <= 200 || x >= 432) && y < 322-43){
            return true;
        }else if (x > 200 && x < 432 && y < 276-43){
            return true;
        }else {
            return false;
        }
    }
    public boolean isWalkable_up(int x, int y){
        return true;
    }

    public boolean arrowEnable_left(int x, int y, Arrow arrow){ //  確認是否箭矢可以繼續前進
        int i = y/23;
        int j = x/23+1;
        return !monsterBuilder.shootingMonster_Left(x, y, arrow) && map[i][j] == 0;
    }

    public boolean arrowEnable_right(int x, int y, Arrow arrow){
        int i = y/23;
        int j = x/23+1;
        return !monsterBuilder.shootingMonster_Right(x, y, arrow) && map[i][j] == 0;
    }

    @Override
    public List<GameMonster> getMonsterList(){ //回傳怪物列表
        return monsterBuilder.getMonsterList();
    }

    @Override
    public MonsterBuilder getMonsterBuilder(){ //回傳 builder
        return monsterBuilder;
```

```
    }

    public int getLevel() { return 1; } //回傳關卡數
}

//Map2.java
public class Map2 implements GameObject, GameMap {
    private MovingBitmap block, block1, block2;
    private MovingBitmap pillar, pillar1, pillar2, pillar3, pillar4;
    private MonsterBuilder monsterBuilder;
    private MovingBitmap scores;
    private int[][] map = { //地圖陣列
            {0,0,1,1,3,1,1,3,1,1,3,3,2,2,1,3,3,3,1,1,1,3,1,2,3,1,1,1,0,0},
            {0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0},
            {0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,1,1,1,1,3,2,1,2,1,1,1,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0},
            {0,0,2,2,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,2,3,1,0,0},
            {0,0,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,0,0},
            {0,0,9,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,1,1,2,3,3,2,1,1,1,3,1,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,9,0,0,0,0,0,0,0,0,0,9,0,0,0,0,0,0,0,9,0,0},
            {0,0,9,0,0,0,0,0,0,9,0,0,0,0,0,0,0,0,0,9,0,0,0,0,0,0,0,9,0,0},
            {0,0,1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,1,3,3,1,2,2,3,0,0},
            {0,0,5,0,0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,5,0,0},
            {0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0}};

    //大地圖左上角座標
    private final int X = 0;
    private final int Y = 0;
    //小地圖寬度與高度
    private final int MW = 23;
    private final int MH = 23;

    public Map2(){
        block = new MovingBitmap(R.drawable.block);
        block1 = new MovingBitmap(R.drawable.block1);
        block2 = new MovingBitmap(R.drawable.block2);
        pillar = new MovingBitmap(R.drawable.pillar);
        pillar1 = new MovingBitmap(R.drawable.pillar1_1);
        pillar2 = new MovingBitmap(R.drawable.pillar2);
        pillar3 = new MovingBitmap(R.drawable.pillar3);
        pillar4 = new MovingBitmap(R.drawable.pillar4);
        scores = new MovingBitmap(R.drawable.scores);
        monsterBuilder = new MonsterBuilder();
        monsterBuilder.add(2, 110,210, 80, 113);
        monsterBuilder.add(2, 110,210, 196, 229);
        scores.setLocation(453,348);
    }

    @Override
    public int getInitialPositionX(){ //回傳人物初始位置
        return 490;
    }

    @Override
    public int getInitialPositionY() { return 279; }

    @Override
    public void release(){ //釋放記憶體
        block.release();
        block1.release();
        block2.release();
        pillar.release();
        pillar1.release();
        pillar2.release();
```

```
        pillar3.release();
        pillar4.release();
        monsterBuilder.release();
        scores.release();

        block = null;
        block1 = null;
        block2 = null;
        pillar = null;
        pillar1 = null;
        pillar2 = null;
        pillar3 = null;
        pillar4 = null;
        scores = null;
}

@Override
public void move(){ //使怪物規律移動
        monsterBuilder.move();
}

@Override
public void show(){ //依照地圖陣列顯示地圖
        scores.show();
        for(int i = 0; i < 17; i++){
                for(int j = 0; j<29; j++){
                        switch(map[i][j]){
                                case 1:
                                        block.setLocation(X+(MW*j), Y+(MH*i));
                                        block.show();
                                        break;
                                case 2:
                                        block1.setLocation(X+(MW*j), Y+(MH*i));
                                        block1.show();
                                        break;
                                case 3:
                                        block2.setLocation(X+(MW*j), Y+(MH*i));
                                        block2.show();
                                        break;
                                case 4:
                                        pillar.setLocation(X+(MW*j), Y+(MH*i));
                                        pillar.show();
                                        break;
                                case 5:
                                        pillar1.setLocation(X+(MW*j), Y+(MH*i));
                                        pillar1.show();
                                        break;
                                case 6:
                                        pillar2.setLocation(X+(MW*j), Y+(MH*i));
                                        pillar2.show();
                                        break;
                                case 7:
                                        pillar3.setLocation(X+(MW*j), Y+(MH*i));
                                        pillar3.show();
                                        break;
                                case 8:
                                        pillar4.setLocation(X+(MW*j), Y+(MH*i));
                                        pillar4.show();
                                        break;
                                default:
                                        break;
                        }
                }
        }
        monsterBuilder.show();
}

public boolean isWalkable_right(int x, int y){ //確認是否可以行走
        if (x > 165 && x <= 184 && y <= 279 && y > 210) {
```

```
            return false;
        }else if (x >= 560) {
            return false;
        }else {
            return true;
        }
}
public boolean isWalkable_left(int x, int y){
        if (x <= 455 && x > 445 && y <= 279 && y > 210) {
            return false;
        }else if (x <= 56) {
            return false;
        }else {
            return true;
        }
}
public boolean isWalkable_down(int x, int y){
        if (x < 167){
            if (x < 115){
                if (y <= 279 && y > 184){
                    return true;
                }else if (y <= 118){
                    return true;
                }else {
                    return false;
                }
            }else {
                if (y <= 279){
                    return true;
                }else {
                    return false;
                }
            }
        }else if (x > 450){
            if (x > 506){
                if (y <= 279 && y > 184){
                    return true;
                }else if (y <= 118){
                    return true;
                }else {
                    return false;
                }
            }else {
                if (y <= 279){
                    return true;
                }else {
                    return false;
                }
            }
        }else {
            if (y <= 187 && y > 138){
                return true;
            }else if (y <= 71){
                return true;
            }else {
                return false;
            }
        }
}

public boolean isWalkable_up(int x, int y){
        if (x < 138 && y < 184 && y >= 181){
            return false;
        }else if (x > 506 && y < 184 && y >= 181){
            return false;
        }else if (x > 167 && x < 450 && y < 26 && y >= 23){
            return false;
        }else {
            return true;
```

```
        }
    }

    public boolean arrowEnable_left(int x, int y, Arrow arrow){ //確認是否可以行走
        int i = y/23;
        int j = x/23+1;
        return !monsterBuilder.shootingMonster_Left(x, y, arrow) && map[i][j] == 0;
    }

    public boolean arrowEnable_right(int x, int y, Arrow arrow){
        int i = y/23;
        int j = x/23+1;
        return !monsterBuilder.shootingMonster_Right(x, y, arrow) && map[i][j] == 0;
    }

    @Override
    public List<GameMonster> getMonsterList(){ //回傳怪物列表
        return monsterBuilder.getMonsterList();
    }

    @Override
    public MonsterBuilder getMonsterBuilder(){ //回傳 builder
        return monsterBuilder;
    }

    public int getLevel() { return 2; } //回傳關卡數
}

//Map3.java
public class Map3 implements GameObject, GameMap {
    private MovingBitmap block, block1, block2, block3, block4;
    private MovingBitmap pillar, pillar1, pillar2, pillar3;
    private MovingBitmap scores;
    private MonsterBuilder monsterBuilder;
    private int[][] map = {
            {0,0,1,8,0,0,0,0,8,8,8,8,8,8,8,8,8,8,8,8,8,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,9,9,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,8,9,9,8,1,0,0},
            {0,0,1,8,8,8,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,8,8,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,1,3,2,1,1,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,8,1,0,0},
            {0,0,1,8,0,0,0,0,8,8,8,8,8,8,8,8,8,8,8,8,8,0,0,0,0,8,1,0,0}};

    //大地圖左上角座標
    private final int X = 0;
    private final int Y = 0;
    //小地圖寬度與高度
    private final int MW = 23;
    private final int MH = 23;

    public Map3(){
        block = new MovingBitmap(R.drawable.block);
        block1 = new MovingBitmap(R.drawable.block1);
        block2 = new MovingBitmap(R.drawable.block2);
        block3 = new MovingBitmap(R.drawable.block3);
        block4 = new MovingBitmap(R.drawable.block4);
        pillar = new MovingBitmap(R.drawable.pillar);
        pillar1 = new MovingBitmap(R.drawable.pillar1);
        pillar2 = new MovingBitmap(R.drawable.pillar2);
```

```
        pillar3 = new MovingBitmap(R.drawable.pillar3);
        scores = new MovingBitmap(R.drawable.scores);
        monsterBuilder = new MonsterBuilder();
        monsterBuilder.add(3, 230,95, 130, 160);
        scores.setLocation(453,348);
    }

    @Override
    public int getInitialPositionX(){ //回傳人物初始位置
        return 310;
    }

    @Override
    public int getInitialPositionY() {
        return 321;
    }

    @Override
    public void release(){ //釋放記憶體
        block.release();
        block1.release();
        block2.release();
        block3.release();
        block4.release();
        pillar.release();
        pillar1.release();
        pillar2.release();
        pillar3.release();
        monsterBuilder.release();
        scores.release();

        block = null;
        block1 = null;
        block2 = null;
        block3 = null;
        block4 = null;
        pillar = null;
        pillar1 = null;
        pillar2 = null;
        pillar3 = null;
        scores = null;
    }

    @Override
    public void move() { //使怪物規律移動
        monsterBuilder.move();
    }

    @Override
    public void show(){ //依照地圖陣列顯示地圖
        scores.show();
        for(int i = 0; i < 17; i++){
            for(int j = 0; j<29; j++){
                switch(map[i][j]){
                    case 1:
                        block.setLocation(X+(MW*j), Y+(MH*i));
                        block.show();
                        break;
                    case 2:
                        block1.setLocation(X+(MW*j), Y+(MH*i));
                        block1.show();
                        break;
                    case 3:
                        block2.setLocation(X+(MW*j), Y+(MH*i));
                        block2.show();
                        break;
                    case 4:
                        pillar.setLocation(X+(MW*j), Y+(MH*i));
                        pillar.show();
```

```
                        break;
                    case 5:
                        pillar1.setLocation(X+(MW*j), Y+(MH*i));
                        pillar1.show();
                        break;
                    case 6:
                        pillar2.setLocation(X+(MW*j), Y+(MH*i));
                        pillar2.show();
                        break;
                    case 7:
                        pillar3.setLocation(X+(MW*j), Y+(MH*i));
                        pillar3.show();
                        break;
                    case 8:
                        block3.setLocation(X+(MW*j), Y+(MH*i));
                        block3.show();
                        break;
                    case 9:
                        block4.setLocation(X+(MW*j), Y+(MH*i));
                        block4.show();
                        break;
                    default:
                        break;
                }
            }
        }
        monsterBuilder.show();
    }

    public boolean isWalkable_right(int x, int y){ //確認是否可以行走
        if (x > 92 && x < 529) {
            if (y < 322 && y > 184) {
                return true;
            } else if (y > 23 && y < 161 - 43) {
                return true;
            } else {
                return false;
            }
        }else {
            return false;
        }
    }
    public boolean isWalkable_left(int x, int y){
        if (x > 92 && x < 529) {
            if (y < 322 && y > 184) {
                return true;
            }else if (y > 23 && y < 161 - 43) {
                return true;
            }else {
                return false;
            }
        }else {
            return false;
        }
    }
    public boolean isWalkable_down(int x, int y) {
        if (y < 23 && (x > 460 || x < 184)){
            return true;
        }else if (y > 23 && y < 161-43){
            return true;
        }else if(y > 161){
            if ((x <= 253 && x > 161) || (x > 368 && x < 460)){
                if (y < 322) {
                    return true;
                }else {
                    return false;
                }
            }else if (x < 184 || x > 460){
                    return true;
```

```
                }else {
                    if (y < 322 && y > 235){
                        return true;
                    }else if (y < 231){
                        return true;
                    }else {
                        return false;
                    }
                }
            }else {
                return false;
            }
        }
        public boolean isWalkable_up(int x, int y) {
            if (y <= 23 && (x > 506 || x < 138)){
                return true;
            }else if (y > 23 && y < 161-43){
                return true;
            }else if(y > 161){
                return true;
            }else {
                return false;
            }
        }

        public boolean arrowEnable_left(int x, int y, Arrow arrow){ //確認箭矢是否可以前進
            int i = y/23;
            int j = x/23+1;
            return !monsterBuilder.shootingMonster_Left(x, y, arrow) && map[i][j] == 0;
        }

        public boolean arrowEnable_right(int x, int y, Arrow arrow){
            int i = y/23;
            int j = x/23+1;
            return !monsterBuilder.shootingMonster_Right(x, y, arrow) && map[i][j] == 0;
        }

        @Override
        public boolean superJump(int x, int y) { //地圖邊緣的彈跳台
            return y <= 161 - 43 && y >= 115 && ((x > 506 || x < 120));
        }
        public List<GameMonster> getMonsterList(){ //回傳怪物列表
            return monsterBuilder.getMonsterList();
        }

        @Override
        public MonsterBuilder getMonsterBuilder() { //回傳 builder
            return monsterBuilder;
        }

        public int getLevel() { return 3; } //回傳關卡數
    }

    //MapController.java
    public class MapController {
        private List<GameMap> mapList;
        private int currentLevel;

        public MapController(){
            mapList = new ArrayList<>();
        }

        public void initialize(){ //初始化，建立地圖
            GameMap map1 = new Map1();
            GameMap map2 = new Map2();
            GameMap map3 = new Map3();
            mapList.add(map1);
            mapList.add(map2);
            mapList.add(map3);
```

```
    }

    public void release(){ //釋放記憶體
        for(GameMap map: mapList){
            if(map != null){
                map.release();
                map = null;
            }
        }
        mapList = null;
    }

    public GameMap FirstLevel(){ //回傳第一關
        currentLevel = 1;
        return mapList.get(currentLevel-1);
    }

    public GameMap NextLevel(){ //回傳下一關
        currentLevel += 1;
        return mapList.get(currentLevel-1);
    }

    public GameMap goToLevel(int level){ //回傳指定關卡
        currentLevel = level-1; // need to be fix
        for(GameMap map: mapList){
            if(map != null && map.getLevel() == currentLevel){
                return map;
            }
        }
        return null;
    }

    public List<GameMap> getMapList(){ //回傳地圖列表
        return mapList;
    }
}

//GameMonster.java
public interface GameMonster {
    public void initialize(int monsterStep, int height);
    public void setIskilled();
    public boolean isKilled();
    public void setLocation(int x, int y);
    public int getX();
    public int getY();
    public int getWidth();
    public int getHeight();
    public void regular();
    public void animePlay();
    public void release();
    public void move();
    public void show();
    public void setVisible(boolean visible);
    public void setDirection(int direction);
    public void setCurrentFrame(int frame);
    public boolean getVisible();
}

//Monster1.java
public class Monster1 extends Animation implements GameMonster {
    private Animation monster;
    public boolean iskilled;
    private int step;
    private int initStep;
    private int direction;
    private double speed;
    boolean turned;
    private int jumpStep, basicHeight;
    private int fallStep;
```

```
public Monster1() { //載入動畫每禎的圖
	monster = new Animation();
	monster.setLocation(0, 0);
	monster.addFrame(R.drawable.monsterleft1);
	monster.addFrame(R.drawable.monsterleft2);
	monster.addFrame(R.drawable.monsterleft3);
	monster.addFrame(R.drawable.monsterleft4);
	monster.addFrame(R.drawable.monsterleft5);
	monster.addFrame(R.drawable.monsterleft6);
	monster.addFrame(R.drawable.monsterleft7);
	monster.addFrame(R.drawable.monsterleft8);
	monster.addFrame(R.drawable.monsterleft9);
	monster.addFrame(R.drawable.monsterleft10);
	monster.addFrame(R.drawable.monsterleft11);
	monster.addFrame(R.drawable.monsterright1);
	monster.addFrame(R.drawable.monsterright2);
	monster.addFrame(R.drawable.monsterright3);
	monster.addFrame(R.drawable.monsterright4);
	monster.addFrame(R.drawable.monsterright5);
	monster.addFrame(R.drawable.monsterright6);
	monster.addFrame(R.drawable.monsterright7);
	monster.addFrame(R.drawable.monsterright8);
	monster.addFrame(R.drawable.monsterright9);
	monster.addFrame(R.drawable.monsterright10);
	monster.addFrame(R.drawable.monsterright11);
	monster.addFrame(R.drawable.d1);
	monster.addFrame(R.drawable.d2);
	monster.addFrame(R.drawable.d3);
	monster.addFrame(R.drawable.d4);
	monster.addFrame(R.drawable.d5);
	monster.addFrame(R.drawable.d6);
	monster.addFrame(R.drawable.d7);
	monster.addFrame(R.drawable.d8);
	monster.addFrame(R.drawable.d9);
	monster.addFrame(R.drawable.d10);
	monster.addFrame(R.drawable.d11);
	monster.addFrame(R.drawable.d12);
	monster.addFrame(R.drawable.d13);
	monster.addFrame(R.drawable.d14);
	monster.addFrame(R.drawable.d15);
	monster.addFrame(R.drawable.d16);
	monster.addFrame(R.drawable.d17);
	monster.addFrame(R.drawable.d18);
	monster.addFrame(R.drawable.d19);
	monster.addFrame(R.drawable.d20);
	monster.addFrame(R.drawable.d21);
	monster.addFrame(R.drawable.d22);
	monster.addFrame(R.drawable.d23);
	monster.addFrame(R.drawable.d24);

	monster.setDelay(2);
}

public void initialize(int monsterStep, int height) { //初始化
	iskilled = false;
	step = monsterStep;
	initStep = monsterStep;
	basicHeight = height;
	jumpStep = 5;
	fallStep = 30;
	direction = 0;
	turned = false;
}

public void setIskilled() { //使怪物死亡
	iskilled = true;
}
```

```
public boolean isKilled() { //回傳是否死亡
	return iskilled;
}

public void setLocation(int x, int y) { //設定位置
	monster.setLocation(x, y);
}

public int getX() { //回傳位置
	return monster.getX();
}

public int getY() {
	return monster.getY();
}

@Override
public int getWidth(){ //回傳圖片寬與高
	return monster.getWidth();
}

@Override
public int getHeight(){
	return monster.getHeight();
}

@Override
public void release() { //釋放記憶體
	monster.release();
	monster = null;
}

@Override
public void move() { //規律移動及動畫播放
	if (monster != null) {
		monster.move();
		animePlay();
		adjustHeight();
	}
}

@Override
public void show() { //顯示
	monster.show();
}

@Override
public void setDirection(int direction) {

}

@Override
public void setCurrentFrame(int frame) {

}

@Override
public boolean getVisible(){ //回傳是否可見
	return monster.getVisible();
}

public void regular() { //規律移動函式
	if (monster != null && !iskilled) {
		speed = 2;
		if (step <= 0 && direction == 0) {
			direction = 1;
			step = initStep;
		} else if (step <= 0 && direction == 1) {
			direction = 0;
```

```
                step = initStep;
            } else {
                if (direction == 0) {
                    monster.setLocation((int) (monster.getX() + speed), monster.getY());
                } else {
                    monster.setLocation((int) (monster.getX() - speed), monster.getY());
                }
            }
            step--;
        }
    }

    public void animePlay() { //動畫播放
        if (iskilled) {
            if (monster.getCurrentFrameIndex() <= 21) {
                monster.setCurrentFrameIndex(22);
            }
            speed++;
            if (jumpStep <= 0) {
                fallStep--;
                monster.setLocation(monster.getX(), (int) (monster.getY() + speed));
                if (monster.getY() + speed >= 368){
                    monster.setVisible(false);
                    monster.setLocation(650, 400);
                }
            } else {
                jumpStep--;
                monster.setLocation(monster.getX(), (int) (monster.getY() - speed));
            }
        }else {
            if (direction == 0) {
                if (monster.getCurrentFrameIndex() <= 10 || monster.getCurrentFrameIndex() >= 21) {
                    monster.setCurrentFrameIndex(11);
                }
            } else if (direction == 1) {
                if (monster.getCurrentFrameIndex() >= 11) {
                    monster.setCurrentFrameIndex(0);
                }
            }
        }
    }

    public void adjustHeight(){ //使怪物保持在同一水平線上
        if(!iskilled) {
            monster.setLocation(monster.getX(), basicHeight - monster.getHeight());
        }
    }
}


//Monster2.java
public class Monster2 extends Animation implements GameMonster {
    private Animation monster;
    public boolean iskilled;
    private int step;
    private int initStep;
    private int direction;
    private double speed;
    boolean turned;
    private int jumpStep, fallStep;
    private boolean firstShot, godMode;
    private boolean shotAgain;
    private int stay, basicHeight;
    private int initY;

    public Monster2() { //載入動畫每禎的圖
        monster = new Animation();
        monster.setLocation(0, 0);
        monster.addFrame(R.drawable.monster1_2);
```

```
        monster.addFrame(R.drawable.monster1_1);
        monster.addFrame(R.drawable.monster1_2);
        monster.addFrame(R.drawable.monster1_3);
        monster.addFrame(R.drawable.monster1_shot1); //4
        monster.addFrame(R.drawable.monster1_shot2);
        monster.addFrame(R.drawable.monster1_shot3);
        monster.addFrame(R.drawable.monster1_shot4);
        monster.addFrame(R.drawable.monster1_shot5);
        monster.addFrame(R.drawable.monster1_shot6);
        monster.addFrame(R.drawable.monster1_shot7);
        monster.addFrame(R.drawable.monster1_shot8);
        monster.addFrame(R.drawable.monster1_shot9);
        monster.addFrame(R.drawable.monster1_shot10); //13
        monster.addFrame(R.drawable.monster1_shot11);
        monster.addFrame(R.drawable.monster1_shot12);
        monster.addFrame(R.drawable.monster1_shot13);
        monster.addFrame(R.drawable.monster1_shot14);
        monster.addFrame(R.drawable.d1); //18
        monster.addFrame(R.drawable.d2);
        monster.addFrame(R.drawable.d3);
        monster.addFrame(R.drawable.d4);
        monster.addFrame(R.drawable.d5);
        monster.addFrame(R.drawable.d6);
        monster.addFrame(R.drawable.d7);
        monster.addFrame(R.drawable.d8);
        monster.addFrame(R.drawable.d9);
        monster.addFrame(R.drawable.d10);
        monster.addFrame(R.drawable.d11);
        monster.addFrame(R.drawable.d12);
        monster.addFrame(R.drawable.d13);
        monster.addFrame(R.drawable.d14);
        monster.addFrame(R.drawable.d15);
        monster.addFrame(R.drawable.d16);
        monster.addFrame(R.drawable.d17);
        monster.addFrame(R.drawable.d18);
        monster.addFrame(R.drawable.d19);
        monster.addFrame(R.drawable.d20);
        monster.addFrame(R.drawable.d21);
        monster.addFrame(R.drawable.d22);
        monster.addFrame(R.drawable.d23);
        monster.addFrame(R.drawable.d24);
        monster.setDelay(2);
    }

    public void initialize(int monsterStep, int height) { //初始化
        iskilled = false;
        step = monsterStep;
        initStep = monsterStep;
        basicHeight = height;
        jumpStep = 5;
        fallStep = 30;
        direction = 0;
        turned = false;
        firstShot = false;
        shotAgain = false;
        godMode = false;
        stay = 40;
    }

    public void setIskilled() { //使怪物死亡
        iskilled = true;
    }

    public void setFirstShot() { //攻擊第一下後脫離無敵
        firstShot = true;
    }

    public boolean getFirstShot() { //回傳狀態
        return firstShot;
```

```
}

public boolean isKilled() { //使怪物死亡
	return iskilled;
}

public void setLocation(int x, int y) { //設置位置
	monster.setLocation(x, y);
}

public int getX() { //回傳位置
	return monster.getX();
}

public int getY() {
	return monster.getY();
}

@Override
public int getWidth(){ //回傳寬與高
	return monster.getWidth();
}

@Override
public int getHeight(){
	return monster.getHeight();
}

@Override
public void release() { //釋放記憶體
	monster.release();
	monster = null;
}

@Override
public void move() { //使怪物規律移動與播放動畫
	if (monster != null) {
		monster.move();
		animePlay();
		adjustHeight();
	}
}

@Override
public void show() { //顯示
	monster.show();
}

@Override
public void setDirection(int direction) {

}

@Override
public void setCurrentFrame(int frame) {

}

@Override
public boolean getVisible(){ //回傳是否可見
	return monster.getVisible();
}

public void regular() { //規律移動
	if (!godMode){
		if (monster != null && !iskilled) {
			speed = 2;
			if (step <= 0 && direction == 0) {
				direction = 1;
```

```
                step = initStep;
            } else if (step <= 0 && direction == 1) {
                direction = 0;
                step = initStep;
            } else {
                if (direction == 0) {
                    monster.setLocation((int) (monster.getX() + speed), monster.getY());
                } else {
                    monster.setLocation((int) (monster.getX() - speed), monster.getY());
                }
            }
            step--;
        }
    }
}

public void setGodMode(){ //無敵
    godMode = true;
}

public boolean isGodMode(){
    return godMode;
}

public void animePlay() { //動畫播放
    if (iskilled) {
        if (monster.getCurrentFrameIndex() <= 21) {
            monster.setCurrentFrameIndex(22);
        }
        speed++;
        if (jumpStep <= 0) {
            fallStep--;
            monster.setLocation(monster.getX(), (int) (monster.getY() + speed));
            if (monster.getY() + speed >= 368) {
                monster.setVisible(false);
                monster.setLocation(650, 400);
            }
        } else {
            jumpStep--;
            monster.setLocation(monster.getX(), (int) (monster.getY() - speed));
        }
    } else if (godMode) {
        if (stay > 0) {
            stay--;
            if (monster.getCurrentFrameIndex() >= 12) {
                firstShot = true;
                monster.setCurrentFrameIndex(12);
            }
        } else {
            firstShot = false;
            if(monster.getCurrentFrameIndex() <= 12){
                monster.setCurrentFrameIndex(13);
            }
            if (monster.getCurrentFrameIndex() >= 17) {
                monster.setCurrentFrameIndex(0);
                godMode = false;
                stay = 40;
            }
        }
    } else {
        if (monster.getCurrentFrameIndex() >= 3) {
            monster.setCurrentFrameIndex(0);
        }
    }
}

public void adjustHeight(){ //使怪物在同一水平線上
    if(!iskilled) {
        monster.setLocation(monster.getX(), basicHeight - monster.getHeight());
```

```
            }
        }
}

//Monster3.java
public class Monster3 extends Animation implements GameMonster {
        private Animation monster;
        public boolean iskilled;
        private int direction;
        private double speed;
        boolean turned;
        private int step, initStep, jumpStep, fallStep, basicHeight;

        public Monster3() {
                monster = new Animation();
                monster.setLocation(0, 0);
                monster.addFrame(R.drawable.monster2_left1);
                monster.addFrame(R.drawable.monster2_left2);
                monster.addFrame(R.drawable.monster2_left3);
                monster.addFrame(R.drawable.monster2_left4);
                monster.addFrame(R.drawable.monster2_left5);
                monster.addFrame(R.drawable.monster2_left6);
                monster.addFrame(R.drawable.monster2_left7);
                monster.addFrame(R.drawable.monster2_left8);
                monster.addFrame(R.drawable.monster2_left9);
                monster.addFrame(R.drawable.monster2_left10);
                monster.addFrame(R.drawable.monster2_right1);
                monster.addFrame(R.drawable.monster2_right2);
                monster.addFrame(R.drawable.monster2_right3);
                monster.addFrame(R.drawable.monster2_right4);
                monster.addFrame(R.drawable.monster2_right5);
                monster.addFrame(R.drawable.monster2_right6);
                monster.addFrame(R.drawable.monster2_right7);
                monster.addFrame(R.drawable.monster2_right8);
                monster.addFrame(R.drawable.monster2_right9);
                monster.addFrame(R.drawable.monster2_right10);
                monster.addFrame(R.drawable.d1);
                monster.addFrame(R.drawable.d2);
                monster.addFrame(R.drawable.d3);
                monster.addFrame(R.drawable.d4);
                monster.addFrame(R.drawable.d5);
                monster.addFrame(R.drawable.d6);
                monster.addFrame(R.drawable.d7);
                monster.addFrame(R.drawable.d8);
                monster.addFrame(R.drawable.d9);
                monster.addFrame(R.drawable.d10);
                monster.addFrame(R.drawable.d11);
                monster.addFrame(R.drawable.d12);
                monster.addFrame(R.drawable.d13);
                monster.addFrame(R.drawable.d14);
                monster.addFrame(R.drawable.d15);
                monster.addFrame(R.drawable.d16);
                monster.addFrame(R.drawable.d17);
                monster.addFrame(R.drawable.d18);
                monster.addFrame(R.drawable.d19);
                monster.addFrame(R.drawable.d20);
                monster.addFrame(R.drawable.d21);
                monster.addFrame(R.drawable.d22);
                monster.addFrame(R.drawable.d23);
                monster.addFrame(R.drawable.d24);
                monster.setDelay(2);
        }

        public void initialize(int monsterStep, int height) { //初始化
                iskilled = false;
                step = monsterStep;
                initStep = monsterStep;
                basicHeight = height;
                jumpStep = 5;
```

```
        fallStep = 30;
        direction = 0;
        turned = false;
    }

    public void setIskilled() { //使怪物死亡
        if(monster.getVisible()){
            iskilled = true;
            monster.setLocation(-30, -30);
        }
    }

    public boolean isKilled() {
        return iskilled;
    }

    public void setLocation(int x, int y) { //設定位置
        monster.setLocation(x, y);
    }

    public int getX() { //回傳位置
        return monster.getX();
    }

    public int getY() {
        return monster.getY();
    }

    @Override
    public int getWidth(){ //回傳寬與高
        return monster.getWidth();
    }

    @Override
    public int getHeight(){
        return monster.getHeight();
    }

    @Override
    public void release() { //釋放記憶體
        monster.release();
        monster = null;
    }

    @Override
    public void move() { //規律移動與動畫播放
        if (monster != null) {
            monster.move();
            animePlay();
            adjustHeight();
        }
    }

    @Override
    public void show() { //顯示
        monster.show();
    }

    public void regular() { //規律移動
        if (monster != null && !iskilled) {
            System.out.println("step: "+step+", direction: "+direction+", x: "+monster.getX()+", y: "+monster.getY());
            speed = 2;
            if (step <= 0 && direction == 0) {
                direction = 1;
                step = initStep;
            } else if (step <= 0 && direction == 1) {
                direction = 0;
                step = initStep;
            } else {
```

```
                if (direction == 0) {
                    monster.setLocation((int) (monster.getX() + speed), monster.getY());
                } else {
                    monster.setLocation((int) (monster.getX() - speed), monster.getY());
                }
                if (!(monster.getX() <= 529) || !(monster.getX() >= 92)) {
                    step = 0;
                }
            }
            step--;
        }
    }

    public void animePlay() { //動畫播放
        if (iskilled) {
            if (monster.getCurrentFrameIndex() <= 19) {
                monster.setCurrentFrameIndex(20);
            }
            speed = 5;
            if (jumpStep <= 0) {
                fallStep--;
                monster.setLocation(monster.getX(), (int) (monster.getY() + speed));
                if (monster.getY() + speed >= 368){
                    monster.setVisible(false);
                    monster.setLocation(650, 400);
                }
            } else {
                jumpStep--;
                monster.setLocation(monster.getX(), (int) (monster.getY() - speed));
            }
        }else {
            if (direction == 0) {
                if (monster.getCurrentFrameIndex() <= 9 || monster.getCurrentFrameIndex() >= 20) {
                    monster.setCurrentFrameIndex(10);
                }
            } else if (direction == 1) {
                if (monster.getCurrentFrameIndex() >= 10) {
                    monster.setCurrentFrameIndex(0);
                }
            }
        }
    }

    public void adjustHeight(){ //使怪物在同一水平線上
        if(!iskilled) {
            if(monster.getCurrentFrameIndex() ==5 || monster.getCurrentFrameIndex() == 6 ||
            monster.getCurrentFrameIndex() == 15 || monster.getCurrentFrameIndex() == 16){
                monster.setLocation(monster.getX(), basicHeight - monster.getHeight()-20);
            }else{
                monster.setLocation(monster.getX(), basicHeight - monster.getHeight());
            }
        }
    }

    @Override
    public void setVisible(boolean visible){
        monster.setVisible(visible);
    }

    @Override
    public void setDirection(int d){
        direction = d;
    }

    @Override
    public void setCurrentFrame(int frame){ //設定動畫張數位置
        monster.setCurrentFrameIndex(frame);
    }
```

```
    @Override
    public boolean getVisible(){
        return monster.getVisible();
    }

}

//MonsterBuilder.java
public class MonsterBuilder {
    private List<GameMonster> MonsterList;

    public MonsterBuilder(){
        MonsterList = new ArrayList<>();
    }

    public void add(int monsterNum, int monsterStep, int x, int y, int height){ //加入怪物
        switch (monsterNum){
            case 1:
                GameMonster m = new Monster1();
                m.initialize(monsterStep, height);
                m.setLocation(x, y);
                MonsterList.add(m);
                break;
            case 2:
                GameMonster m2 = new Monster2();
                m2.initialize(monsterStep, height);
                m2.setLocation(x, y);
                MonsterList.add(m2);
                break;
            case 3:
                GameMonster m3 = new Monster3();
                GameMonster m3_1 = new Monster3();
                GameMonster m3_2 = new Monster3();
                GameMonster m3_3 = new Monster3();
                GameMonster m3_4 = new Monster3();
                GameMonster m3_5 = new Monster3();
                GameMonster m3_6 = new Monster3();
                MonsterList.add(m3);
                MonsterList.add(m3_1);
                MonsterList.add(m3_2);
                MonsterList.add(m3_3);
                MonsterList.add(m3_4);
                MonsterList.add(m3_5);
                MonsterList.add(m3_6);
                for(int i=0; i<MonsterList.size(); i++) {
                    MonsterList.get(i).initialize(monsterStep, height);
                    MonsterList.get(i).setLocation(x, y);
                    if(i != 0){
                        MonsterList.get(i).setVisible(false);
                    }
                }
                break;
            default:
                break;
        }
    }

    public void show(){ //顯示
        for(GameMonster m: MonsterList){
            m.show();
            if(m instanceof Monster3){
                monster3_split();
            }
        }
    }

    public void move(){ //使怪物規律移動
        for(GameMonster m: MonsterList){
            if(m != null){
```

```
                m.move();
                m.regular();
            }
        }
    }

    public void release(){ //釋放記憶體
        for(GameMonster m: MonsterList){
            m.release();
            m = null;
        }
    }

    public boolean shootingMonster_Left(int x, int y, Arrow arrow){ //確認怪物是否被擊中
        for(GameMonster monster: MonsterList){
            if(x > monster.getX()+monster.getWidth() || x < monster.getX()){
                continue;
            }else if(y > monster.getY()+monster.getHeight() || y+arrow.getHeight() < monster.getY()){
                continue;
            }else{
                if(!arrow.noPower){
                    if (!(monster instanceof Monster2)){
                        monster.setIskilled();
                        arrow.hitMonster = true;
                        return true;
                    }else{
                        ((Monster2) monster).setGodMode();
                        arrow.hitMonster = true;
                        if(((Monster2) monster).getFirstShot()) {
                            monster.setIskilled();
                            return true;
                        }
                        return false;
                    }
                }
            }
        }
        return false;
    }

    public boolean shootingMonster_Right(int x, int y, Arrow arrow){
        for(GameMonster monster: MonsterList){
            if(x+arrow.getWidth() > monster.getX()+monster.getWidth() || x+arrow.getWidth() < monster.getX()){
                continue;
            }else if(y > monster.getY()+monster.getHeight() || y+arrow.getHeight() < monster.getY()){
                continue;
            }else{
                if(!arrow.noPower){
                    if (!(monster instanceof Monster2)){
                        monster.setIskilled();
                        arrow.hitMonster = true;
                        return true;
                    }else{
                        ((Monster2) monster).setGodMode();
                        arrow.hitMonster = true;
                        if(((Monster2) monster).getFirstShot()) {
                            monster.setIskilled();
                            return true;
                        }
                        return false;
                    }
                }
            }
        }
        return false;
    }

    public void monster3_split(){ //創造分裂的怪物 3
        for(int i=0; i<MonsterList.size(); i++){
```

```
                if(i==0 && MonsterList.get(i).isKilled() && !MonsterList.get(i+1).isKilled()){
                    if(!MonsterList.get(i+1).isKilled() && !MonsterList.get(i+2).isKilled()){
                        MonsterList.get(i+1).setVisible(true);
                        MonsterList.get(i+1).setDirection(0);
                        MonsterList.get(i+3).setDirection(0);
                        MonsterList.get(i+4).setDirection(0);
                        MonsterList.get(i+3).setLocation(MonsterList.get(i+1).getX(), MonsterList.get(i+1).getY());
                        MonsterList.get(i+4).setLocation(MonsterList.get(i+1).getX(), MonsterList.get(i+1).getY());
                        MonsterList.get(i+2).setVisible(true);
                        MonsterList.get(i+2).setDirection(1);
                        MonsterList.get(i+5).setDirection(1);
                        MonsterList.get(i+6).setDirection(1);
                        MonsterList.get(i+5).setLocation(MonsterList.get(i+2).getX(), MonsterList.get(i+2).getY());
                        MonsterList.get(i+6).setLocation(MonsterList.get(i+2).getX(), MonsterList.get(i+2).getY());
                    }
                }else if(i == 1 && MonsterList.get(i).isKilled()){
                    if(!MonsterList.get(i+2).getVisible()){
                        MonsterList.get(i+2).setVisible(true);
                        MonsterList.get(i+3).setVisible(true);
                        MonsterList.get(i+2).setDirection(0);
                        MonsterList.get(i+3).setDirection(1);
                    }
                }else if(i == 2 && MonsterList.get(i).isKilled()){
                    if(!MonsterList.get(i+3).getVisible()){
                        MonsterList.get(i+3).setVisible(true);
                        MonsterList.get(i+4).setVisible(true);
                        MonsterList.get(i+3).setDirection(0);
                        MonsterList.get(i+4).setDirection(1);
                    }
                }
            }
        }

        public List<GameMonster> getMonsterList(){
            return MonsterList;
        }

        public int checkScore(){ //確認分數
            int score = 0;
            for(GameMonster m: MonsterList){
                if(m.isKilled()){score += 100;}
            }
            return score;
        }

        public void killAll(){ //金手指，殺死全怪物
            for(GameMonster m: MonsterList){
                m.setIskilled();
            }
        }
    }

    //Arrow.java
    public class Arrow extends Animation {
        private Animation arrow;
        private Timer timer;
        private TimerTask timerTask;
        private String direction;
        private GameMap gameMap;
        private int fallCount, disappearCount;
        private boolean attack;
        public boolean noPower, onWall, hitMonster;

        public Arrow(GameMap map){
            arrow = new Animation();
            direction = null;
            gameMap = map;
            fallCount = 0;
            disappearCount = 50;
```

```
        arrow.setVisible(false);
        attack = false;
        hitMonster = false;
    }

    public void initializeLeft(){ //初始化左方向箭矢
        arrow.addFrame(R.drawable.arrowleft);
        arrow.setDelay(2);
        direction = "Left";
        noPower = false;
        onWall = false;
    }

    public void initializeRight(){ //初始化右方向箭矢
        arrow.addFrame(R.drawable.arrowright);
        arrow.setDelay(2);
        direction = "Right";
        noPower = false;
        onWall = false;
    }

    public void attack(int mainX, int mainY, int delay){ //發送攻擊指令
        attack = true;
        arrow.setLocation(mainX, mainY+15);
        fallCount = delay;
        arrow.setVisible(true);
        noPower = false;
        hitMonster = false;
    }

    public void shoot(){ //攻擊
        if(attack){
            if(direction.contains("Left")){
                if(gameMap.arrowEnable_left(arrow.getX()-20, arrow.getY(), this) && fallCount > 0 && !hitMonster){
                    arrow.setLocation(arrow.getX()-20, arrow.getY());
                    fallCount--;
                }else if(gameMap.arrowEnable_left(arrow.getX()-20, arrow.getY()+18, this) && !hitMonster){
                    arrow.setLocation(arrow.getX()-20, arrow.getY()+6);
                }else if(gameMap.arrowEnable_left(arrow.getX()-20, arrow.getY(), this) && !hitMonster){
                    arrow.setLocation(arrow.getX()-20, arrow.getY());
                    noPower = true;
                }else{
                    noPower = true;
                    disappearCount--;
                    if(hitMonster){
                        arrow.setVisible(false);
                        attack = false;
                    }else if(disappearCount != 0){
                        onWall = true;
                        if(disappearCount%2==0){
                            arrow.setVisible(false);
                        }else{
                            arrow.setVisible(true);
                        }
                    }else{
                        attack = false;
                        onWall = false;
                    }
                }
                if(arrow.getX() < -20){
                    arrow.setLocation(600, arrow.getY());
                }
            }
            if(direction.contains("Right")){
                if(gameMap.arrowEnable_right(arrow.getX()+20, arrow.getY(), this) && fallCount > 0 && !hitMonster){
                    arrow.setLocation(arrow.getX()+20, arrow.getY());
                    fallCount--;
                }else if(gameMap.arrowEnable_right(arrow.getX()+20, arrow.getY()+18, this) && !hitMonster){
                    arrow.setLocation(arrow.getX()+20, arrow.getY()+6);
```

```
                }else if(gameMap.arrowEnable_right(arrow.getX()+20, arrow.getY(), this) && !hitMonster){
                    arrow.setLocation(arrow.getX()+20, arrow.getY());
                    noPower = true;
                }else{
                    noPower = true;
                    disappearCount--;
                    if(hitMonster){
                        arrow.setVisible(false);
                        attack = false;
                    }else if(disappearCount != 0){
                        onWall = true;
                        if(disappearCount%2==0){
                            arrow.setVisible(false);
                        }else{
                            arrow.setVisible(true);
                        }
                    }else{
                        attack = false;
                        onWall = false;
                    }
                }
                if(arrow.getX() > 600){
                    arrow.setLocation(0, arrow.getY());
                }
            }
        }

}

@Override
public int getX(){ //回傳箭矢位置
    return arrow.getX();
}

@Override
public int getY(){
    return arrow.getY();
}

@Override
public int getWidth(){ //回傳箭矢寬與高
    return arrow.getWidth();
}

@Override
public int getHeight(){
    return arrow.getHeight();
}

public boolean getAttackState(){
    return attack;
}

@Override
public void move(){ //箭矢移動
    arrow.move();
    shoot();
}

@Override
public void show(){ //顯示
    arrow.show();
}


private void stopTimer(){ //停止 timer
    if(timerTask!= null){
        timerTask.cancel();
        timerTask = null;
```

```
            }
            if(timer != null){
                timer.cancel();
                timer = null;
            }
        }

        @Override
        public void release(){ //釋放記憶體
            arrow.release();
            arrow = null;
        }
}

//Character.java
public class Character implements GameObject {
        private Animation main;
        public Arrow arrowLeft, arrowRight;
        private Timer timer;
        private TimerTask timerTask;
        public int life, score;
        private int godModeCount, jumpHeight;
        private GameMap gameMap;
        private String direction;
        private boolean jumping, falling, shooting, godMode, onArrow, rightCheck, leftCheck;
        public boolean dead;

        public Character(){
            main = new Animation();
            arrowLeft = null;
            arrowRight = null;
            timer = null;
            timerTask = null;
            jumpHeight = 5;
            godModeCount = 50;
            life = 3;
            score = 0;
            direction = "standingRight";
            godMode = false;
            falling = false;
            shooting = false;
            dead = false;
            rightCheck = false;
            leftCheck = false;
        }

        public void initialize(GameMap map){ //初始化
            main.setLocation(map.getInitialPositionX(), map.getInitialPositionY());
            main.addFrame(R.drawable.mainright);
            main.addFrame(R.drawable.mainright);
            main.addFrame(R.drawable.mainright1);
            main.addFrame(R.drawable.mainright2);
            main.addFrame(R.drawable.mainright3);
            main.addFrame(R.drawable.mainright4);
            main.addFrame(R.drawable.mainright5);
            main.addFrame(R.drawable.mainright6);
            main.addFrame(R.drawable.mainright7);
            main.addFrame(R.drawable.mainleft);
            main.addFrame(R.drawable.mainleftshake);
            main.addFrame(R.drawable.mainleft1);
            main.addFrame(R.drawable.mainleft2);
            main.addFrame(R.drawable.mainleft3);
            main.addFrame(R.drawable.mainleft4);
            main.addFrame(R.drawable.mainleft5);
            main.addFrame(R.drawable.mainleft6);
            main.addFrame(R.drawable.mainleft7);
            main.addFrame(R.drawable.shoot_left1);
            main.addFrame(R.drawable.shoot_left2);
            main.addFrame(R.drawable.shoot_left3);
```

```
        main.addFrame(R.drawable.shoot_left4);
        main.addFrame(R.drawable.shoot_right1_1);
        main.addFrame(R.drawable.shoot_right2_1);
        main.addFrame(R.drawable.shoot_right3_1);
        main.addFrame(R.drawable.shoot_right4_1);
        main.addFrame(R.drawable.jumpleft);
        main.addFrame(R.drawable.jumpright);
        main.addFrame(R.drawable.fallleft);
        main.addFrame(R.drawable.fallright);
        main.setDelay(2);
        gameMap = map;
    }

    public boolean isJumping(){ //回傳是否跳躍中
        return jumping;
    }

    public void jump(int height){ //跳躍
        jumping = true;
        onArrow = false;
        jumpHeight = height*5;
        if(timer == null){ timer = new Timer(); }
        timer.scheduleAtFixedRate(timerTask = new TimerTask() {
            @Override
            public void run() {
                if(jumping){
                    if(jumpHeight>0){
                        if(gameMap.isWalkable_up(main.getX(), main.getY()-5)){
                            main.setLocation(main.getX(), main.getY() - 5);
                            jumpHeight--;
                        }else{
                            jumpHeight = 0;
                        }

                    }else{
                        falling = true;
                        if(gameMap.isWalkable_down(main.getX(), main.getY()+5)){
                            main.setLocation(main.getX(), main.getY()+5);
                            jumpHeight--;
                            if(checkIfLeftArrowOnWall(main.getX(), main.getY())){
                                jumping = false;
                                falling = false;
                                onArrow = true;
                                direction = "standingLeft";
                                stopTimer();
                            }
                            if(checkIfRightArrowOnWall(main.getX(), main.getY())){
                                jumping = false;
                                falling = false;
                                onArrow = true;
                                direction = "standingRight";
                                stopTimer();
                            }

                        }else{
                            jumping = false;
                            falling = false;
                            onArrow = false;
                            stopTimer();
                        }
                    }
                }
            }
        }, 100, 20);
    }

    public void shoot(){ //攻擊
        shooting = true;
        if(direction.contains("Right")){
```

```
                arrowRight = new Arrow(gameMap);
                arrowRight.initializeRight();
                arrowRight.attack(main.getX(), main.getY(), 20);
                direction = "shootingRight";
            }
            if(direction.contains("Left")){
                arrowLeft = new Arrow(gameMap);
                arrowLeft.initializeLeft();
                arrowLeft.attack(main.getX(), main.getY(), 20);
                direction = "shootingLeft";
            }
        }

        public void checkGravity(){ //加入重力
            if(!jumping && onArrow){
                if(!checkIfLeftArrowOnWall(main.getX(), main.getY()) && !checkIfRightArrowOnWall(main.getX(),
main.getY())){
                    jump(0);
                    onArrow = false;
                }
            }
        }

        public void setDirection(String d){ //設定方向
            direction = d;
        }

        @Override
        public void move(){ //移動與動畫播放
            main.move();
            if(arrowRight != null){
                arrowRight.move();
                if(!arrowRight.getAttackState()){
                    arrowRight.release();
                    arrowRight = null;
                }
            }
            if(arrowLeft != null){
                arrowLeft.move();
                if(!arrowLeft.getAttackState()){
                    arrowLeft.release();
                    arrowLeft = null;
                }
            }
            animePlay(direction);
            if(main.getY() > 400){
                main.setLocation(main.getX(), 0);
            }
            if (!falling && main.getY() < 10){
                main.setLocation(main.getX(), 391);
            }
            if (gameMap.superJump(main.getX(),main.getY())){
                jump(11);
            }
            checkGravity();
        }

        @Override
        public void show(){ //顯示
            main.show();
            if(arrowRight != null){ arrowRight.show(); }
            if(arrowLeft != null){ arrowLeft.show(); }
        }

        @Override
        public void release(){ //釋放記憶體
            main.release();
            main = null;
        }
```

```
public int getX(){ //回傳角色位置
	return main.getX();
}

public int getY(){
	return main.getY();
}

public int getWidth(){ //回傳角色寬與高
	return main.getWidth();
}

public int getHeight(){
	return main.getHeight();
}

public void setLocation(int x, int y){ //設定位置
	main.setLocation(x, y);
}

public String getDirection() {return direction;} //回傳方向

public int getJumpHeight(){ //回傳跳躍高度
	return jumpHeight;
}

public void reset(){ //重置角色動畫
	main.reset();
}

public void animePlay(String s){ //動畫播放
	if(godMode && godModeCount != 0){
		if(main.getVisible()){
			main.setVisible(false);
		}else{
			main.setVisible(true);
		}
		godModeCount--;
		if(godModeCount == 0){
			godMode = false;
			godModeCount = 50;
		}
	}
	if(falling && !shooting){
		if(s.contains("Left")){
			if(main.getCurrentFrameIndex() != 28){
				main.setCurrentFrameIndex(28);
			}
		}else{
			main.setCurrentFrameIndex(29);
		}
	}else if(jumping && !shooting && s.contains("Left")){
		if(main.getCurrentFrameIndex() != 26){
			main.setCurrentFrameIndex(26);
		}
	}else if(jumping && !shooting && s.contains("Right")){
		if(main.getCurrentFrameIndex() != 27){
			main.setCurrentFrameIndex(27);
		}
	}else if(s.equals("Right")){
		if(main.getCurrentFrameIndex() >= 8 || main.getCurrentFrameIndex() < 2){
			main.setCurrentFrameIndex(2);
		}
	}else if(s.equals("Left")){
		if(main.getCurrentFrameIndex() >= 17 || main.getCurrentFrameIndex() < 11){
			main.setCurrentFrameIndex(11);
		}
	}else if(s.equals("standingRight")){
```

```
            if(main.getCurrentFrameIndex() >= 2){
                main.setCurrentFrameIndex(0);
            }
        }else if(s.equals("standingLeft")){
            if(main.getCurrentFrameIndex() != 9){
                main.setCurrentFrameIndex(9);
            }
        }else if(s.equals("shootingLeft")){
            if(main.getCurrentFrameIndex() < 18 || main.getCurrentFrameIndex() >= 22){
                main.setCurrentFrameIndex(18);
            }
            if(main.getCurrentFrameIndex() == 21){
                shooting = false;
                direction = "standingLeft";
            }
        }else if(s.equals("shootingRight")){
            if(main.getCurrentFrameIndex() < 22 || main.getCurrentFrameIndex() >= 26){
                main.setCurrentFrameIndex(22);
            }
            if(main.getCurrentFrameIndex() == 25){
                shooting = false;
                direction = "standingRight";
            }
        }
    }


    private void stopTimer(){ //停止 timer
        if(timer != null){
            timer.cancel();
            timer = null;
        }
        if(timerTask != null){
            timerTask.cancel();
            timerTask = null;
        }
        jumpHeight = 5;
    }

    public void hurt(boolean collide){ //受傷以及無敵時間
        if(collide){
            if(!godMode && life > 0){
                jump(5);
                godMode = true;
                life--;
            }
        }
        if(life == 0){
            godMode = true;
            if(main.getY() < 390){
                main.setLocation(main.getX(), main.getY()+5);
                if(main.getY() > 385){
                    dead = true;
                }
            }
        }
    }

    public boolean checkIfLeftArrowOnWall(int mainX, int mainY){ //若左箭矢在牆上則確認角色位置
        int ArrowX;
        int ArrowY;
        if(arrowLeft != null && arrowLeft.onWall){
            ArrowX = arrowLeft.getX();
            ArrowY = arrowLeft.getY();
            if(mainX+46 > ArrowX && mainX <= ArrowX+46 && mainY+40 < ArrowY && mainY+40 > ArrowY - 10){
                return true;
            }
        }
        return false;
```

```
    }

    public boolean checkIfRightArrowOnWall(int mainX, int mainY){ //若右箭矢在牆上則確認角色位置
        int ArrowX;
        int ArrowY;
        if(arrowRight != null && arrowRight.onWall){
            ArrowX = arrowRight.getX();
            ArrowY = arrowRight.getY();
            if(mainX+46 > ArrowX && mainX <= ArrowX+46 && mainY+40 < ArrowY && mainY+40 > ArrowY - 10){
                return true;
            }
        }
        return false;
    }

    public boolean checkIfArrowOnWall(){ //確認角色是否可以站在箭上
        if(arrowRight != null && arrowRight.onWall && !rightCheck){
            rightCheck = true;
            return true;
        }
        if(rightCheck && arrowRight != null && !arrowRight.onWall){
            rightCheck = false;
        }
        if(arrowLeft != null && arrowLeft.onWall && !leftCheck){
            leftCheck = true;
            return true;
        }
        if(rightCheck && arrowLeft != null && !arrowLeft.onWall){
            leftCheck = false;
        }
        return false;
    }
}

//StateChange.java
public class StateChange extends AbstractGameState {
    private MovingBitmap _complete;
    private MovingBitmap _background;
    private BitmapButton _continue;
    private BitmapButton _end;
    private Map<String, Object> Nextlevel;
    private int currentLevel;

    public StateChange(GameEngine engine) {
        super(engine);
    }

    @Override
    public void initialize(Map<String, Object> data) { //初始化頁面按鈕
        currentLevel = (int) data.get("level");
        Nextlevel = new HashMap<>();
        Nextlevel.put("level", currentLevel+1);
        Nextlevel.put("score", data.get("score"));

        addGameObject(_complete = new MovingBitmap(R.drawable.complete));
        addGameObject(_background = new MovingBitmap(R.drawable.levelbackground3));
        _background.setLocation(0,0);
        _complete.setLocation(225,80);
        initializeContinueButton();
        initializeEndButton();
    }

    private void initializeContinueButton() { //繼續按鈕
        addGameObject(_continue = new BitmapButton(R.drawable.continue_button, R.drawable.continue_pressed, 250,
190));
        _continue.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                changeState(Game.RUNNING_STATE, Nextlevel);
```

```
            }
        });
        addPointerEventHandler(_continue);
    }

    private void initializeEndButton() { //回到首頁按鈕
        addGameObject(_end = new BitmapButton(R.drawable.endgame, R.drawable.endgame_pressed, 250, 230));
        _end.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                changeState(Game.INITIAL_STATE);
            }
        });
        addPointerEventHandler(_end);
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }
}

//StateOver.java
public class StateOver extends AbstractGameState {
    private MovingBitmap _levelFailed;
    private MovingBitmap _background;

    private BitmapButton _end;
    private BitmapButton _restartButton;

    private int currentLevel;
    private Map<String, Object> Nextlevel;

    public StateOver(GameEngine engine) {
        super(engine);
    }

    @Override
    public void initialize(Map<String, Object> data) { //初始化按鈕
        currentLevel = (int) data.get("level");
        Nextlevel = new HashMap<>();
        Nextlevel.put("level", currentLevel);
        Nextlevel.put("score", data.get("score"));

        addGameObject(_levelFailed = new MovingBitmap(R.drawable.levelfailed));
        addGameObject(_background = new MovingBitmap(R.drawable.levelbackground3));
        _background.setLocation(0,0);
        _levelFailed.setLocation(249,80);
        initializeRestartButton();
        initializeEndButton();
    }

    private void initializeRestartButton() { //重新挑戰按鈕
        addGameObject(_restartButton = new BitmapButton(R.drawable.restartlevel, R.drawable.restartlevel_pressed, 219,
                                                        190));
        _restartButton.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                changeState(Game.RUNNING_STATE, Nextlevel);
            }
        });
        addPointerEventHandler(_restartButton);
    }

    private void initializeEndButton() { //回到首頁按鈕
        addGameObject(_end = new BitmapButton(R.drawable.endgame2, R.drawable.endgame2_pressed, 219, 230));
```

```
        _end.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                changeState(Game.INITIAL_STATE);
            }
        });
        addPointerEventHandler(_end);
    }

    @Override
    public void move() {
        super.move();
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }
}

//StatePause.java
public class StatePause extends AbstractGameState {
    private MovingBitmap _pause;
    private MovingBitmap _background;
    private BitmapButton _resume;
    private BitmapButton _end;
    private GameState temp;
    private int currentLevel;

    public StatePause(GameEngine engine) {
        super(engine);
    }

    @Override
    public void initialize(Map<String, Object> data) { //初始化按鈕
        temp = (GameState) data.get("state");
        addGameObject(_pause = new MovingBitmap(R.drawable.paused));
        addGameObject(_background = new MovingBitmap(R.drawable.levelbackground3));
        _background.setLocation(0,0);
        _pause.setLocation(300,80);
        initializeResumeButton();
        initializeEndButton();
//          initializeSoundButton();
    }

    private void initializeResumeButton() { //繼續遊戲按鈕
        addGameObject(_resume = new BitmapButton(R.drawable.resume, R.drawable.resume_pressed, 250, 190));
        _resume.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                resumeState(2, null);
                temp.resume();
            }
        });
        addPointerEventHandler(_resume);
    }

    private void initializeEndButton() { //回到首頁按鈕
        addGameObject(_end = new BitmapButton(R.drawable.endgame, R.drawable.endgame_pressed, 250, 230));
        _end.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                changeState(Game.INITIAL_STATE);
            }
        });
        addPointerEventHandler(_end);
```

```
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }
}

//StateReady.java
public class StateReady extends AbstractGameState {
    private MovingBitmap _mode;
    private MovingBitmap _levels;
    private MovingBitmap _background;
    private MovingBitmap _about;
    private BitmapButton _play;
    private BitmapButton _back1, _back2, _back3;
    private BitmapButton _startButton, _aboutButton;
    private BitmapButton _level1, _level2, _level3;
    private Map<String, Object> level1, level2, level3;
    private Map<String, Object> chooseLevel;

    public StateReady(GameEngine engine) {
        super(engine);
    }

    @Override
    public void initialize(Map<String, Object> data) { //初始化按鈕
        addGameObject(_mode = new MovingBitmap(R.drawable.select));
        addGameObject(_background = new MovingBitmap(R.drawable.background3));
        addGameObject(_levels = new MovingBitmap(R.drawable.levels));
        addGameObject(_about = new MovingBitmap(R.drawable.about_info));
        _background.resize((int)(_background.getWidth()*1.124),(int)(_background.getHeight()*1.045));
        _about.resize((int)(_about.getWidth()*1.124),(int)(_about.getHeight()*1.045));
        _background.setLocation(55,0);
        _mode.setLocation(55,0);
        _levels.setLocation(55,0);
        _about.setLocation(55,0);
        initializeStartButton();
        initializeAboutButton();
        initializeBack1Button();
        initializePlayButton();
        initializeBack2Button();
        initializeBack3Button();
        initializeLevel1Button();
        initializeLevel2Button();
        initializeLevel3Button();
        setVisibility(false, false,false);

        level1 = new HashMap<>();
        level1.put("level", 2);
        level1.put("score", 0);
        level2 = new HashMap<>();
        level2.put("level", 3);
        level2.put("score", 0);
        level3 = new HashMap<>();
        level3.put("level", 4);
        level3.put("score", 0);
    }

    private void initializeBack2Button() { //回上一頁
        addGameObject(_back2 = new BitmapButton(R.drawable.back, R.drawable.back_pressed, 257, 280));
        _back2.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                setVisibility(true, false,false);
            }
```

```
        });
        addPointerEventHandler(_back2);
    }

    private void initializeBack3Button() { //回上一頁
        addGameObject(_back3 = new BitmapButton(R.drawable.back, R.drawable.back_pressed, 257, 280));
        _back3.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                setVisibility(false, false, false);
            }
        });
        addPointerEventHandler(_back3);
    }

    private void initializePlayButton() { //開始關卡
        addGameObject(_play = new BitmapButton(R.drawable.play, R.drawable.play_pressed, 492, 120));
        _play.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                setVisibility(false, true,false);
            }
        });
        addPointerEventHandler(_play);
    }

    private void initializeBack1Button() { //回上一頁
        addGameObject(_back1 = new BitmapButton(R.drawable.back, R.drawable.back_pressed, 257, 319));
        _back1.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                setVisibility(false, false,false);
            }
        });
        addPointerEventHandler(_back1);
    }

    private void initializeLevel1Button() { //關卡 1 按鈕
        addGameObject(_level1 = new BitmapButton(R.drawable.level1, R.drawable.level1_pressed, 76, 112));
        _level1.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
//                  changeState(Game.RUNNING_STATE);
                changeState(Game.RUNNING_STATE, level1);
            }
        });
        addPointerEventHandler(_level1);
    }

    private void initializeLevel2Button() { //關卡 2 按鈕
        addGameObject(_level2 = new BitmapButton(R.drawable.level2, R.drawable.level2_pressed, 127, 112));
        _level2.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                changeState(Game.RUNNING_STATE, level2);
            }
        });
        addPointerEventHandler(_level2);
    }

    private void initializeLevel3Button() { //關卡 3 按鈕
        addGameObject(_level3 = new BitmapButton(R.drawable.level3, R.drawable.level3_pressed, 179, 112));
        _level3.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                changeState(Game.RUNNING_STATE, level3);
            }
        });
        addPointerEventHandler(_level3);
```

```
    }

    private void initializeStartButton() { //首頁的 start 按鈕
        addGameObject(_startButton = new BitmapButton(R.drawable.clickstart, R.drawable.clickstart_pressed, 220, 331));
        _startButton.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                setVisibility(true, false,false);
            }
        });
        addPointerEventHandler(_startButton);
    }

    private void initializeAboutButton() { //about 頁面
        addGameObject(_aboutButton = new BitmapButton(R.drawable.about, R.drawable.about_pressed, 500, 320));
        _aboutButton.addButtonEventHandler(new ButtonEventHandler() {
            @Override
            public void perform(BitmapButton button) {
                setVisibility(false, false,true);
            }
        });
        addPointerEventHandler(_aboutButton);
    }

    @Override
    public void pause() {
    }

    @Override
    public void resume() {
    }

    private void setVisibility(boolean showMode, boolean showLevels, boolean showAbout) { //設定按鈕出現時機
        boolean showMenu = !showLevels && !showMode && !showAbout;
        _background.setVisible(showMenu);
        _mode.setVisible(showMode);
        _levels.setVisible(showLevels);
        _about.setVisible(showAbout);

        _startButton.setVisible(showMenu);
        _aboutButton.setVisible(showMenu);
        _play.setVisible(showMode);
        _back1.setVisible(showMode);
        _back2.setVisible(showLevels);
        _back3.setVisible(showAbout);
        _level1.setVisible(showLevels);
        _level2.setVisible(showLevels);
        _level3.setVisible(showLevels);
    }
}

//StateRun.java
public class StateRun extends GameState {
    public static final int DEFAULT_SCORE_DIGITS = 4;
    private MovingBitmap _background, _button, _message, _message_attack, _message_jump, _pauseButton;
    private GameMap gameMap;
    private MonsterBuilder monsterBuilder;
    private List<GameMonster> MonsterList;
    private Character character;
    private MovingBitmap _life1, _life2, _life3;
    private MovingBitmap _black1, _black2, _black3;
    private MovingBitmap s, t, a, g, e, level, refreshButton, killMonster;
    private Integer _scores;
    private int currentScore, beginWordsAnimationCount;
    private boolean _grab;
    public boolean pausing;
    private AudioController audioController;
    private Pointer _pointer1, _pointer2;
    private Map<String, Object> changelevel;
```

```
public StateRun(GameEngine engine) {
    super(engine);
}

@Override
public void initialize(Map<String, Object> data) { //初始化所有所需物件
    MapController mapController = new MapController();
    mapController.initialize();
    if (data != null) {
        gameMap = mapController.goToLevel((int) data.get("level"));
        currentScore = (int) data.get("score");
    } else {
        gameMap = mapController.FirstLevel();
        currentScore = 0;
    }
    _background = new MovingBitmap(R.drawable.levelbackground3);
    _background.setLocation(0, 0);
    _message = new MovingBitmap(R.drawable.message_move, 130, 150);
    _message_attack = new MovingBitmap(R.drawable.message_attack, 490, 60);
    _message_jump = new MovingBitmap(R.drawable.message_jump, 490, 230);
    s = new MovingBitmap(R.drawable.s);
    t = new MovingBitmap(R.drawable.t);
    a = new MovingBitmap(R.drawable.a);
    g = new MovingBitmap(R.drawable.g);
    e = new MovingBitmap(R.drawable.e);
    s.setLocation(571, 80);//271
    t.setLocation(596, 80);//296
    a.setLocation(621, 80);//321
    g.setLocation(646, 80);//346
    e.setLocation(671, 80);//371
    beginWordsAnimationCount = 30;
    switch (gameMap.getLevel()) {
        case 1:
            level = new MovingBitmap(R.drawable.one);
            break;
        case 2:
            level = new MovingBitmap(R.drawable.two);
            break;
        case 3:
            level = new MovingBitmap(R.drawable.three);
            break;
    }
    level.setLocation(696, 80);
    _button = new MovingBitmap(R.drawable.button);
    _button.setLocation(100, 200);
    _pauseButton = new MovingBitmap(R.drawable.pause);
    _pauseButton.setLocation(590, 10);
    refreshButton = new MovingBitmap(R.drawable.refresh_button);
    refreshButton.setLocation(560, 10);
    killMonster = new MovingBitmap(R.drawable.sword_button);
    killMonster.setLocation(530, 10);
    changelevel = new HashMap<>();
    changelevel.put("level", gameMap.getLevel() + 1);
    monsterBuilder = gameMap.getMonsterBuilder();
    MonsterList = monsterBuilder.getMonsterList();
    _scores = new Integer(DEFAULT_SCORE_DIGITS, 0, 460, 350);
    character = new Character();
    character.initialize(gameMap);
    audioController = new AudioController();
    audioController.play(AudioType.BACKGROUND);
    _grab = false;
    pausing = false
    _pointer1 = null;
    _pointer2 = null;
    _black3 = new MovingBitmap(R.drawable.blacklife);
    _black3.setLocation(600, 350);
    _black2 = new MovingBitmap(R.drawable.blacklife);
    _black2.setLocation(577, 350);
```

```
        _black1 = new MovingBitmap(R.drawable.blacklife);
        _black1.setLocation(554, 350);
        _life3 = new MovingBitmap(R.drawable.lifewithoutframe);
        _life3.setLocation(603, 353);
        _life2 = new MovingBitmap(R.drawable.lifewithoutframe);
        _life2.setLocation(580, 353);
        _life1 = new MovingBitmap(R.drawable.lifewithoutframe);
        _life1.setLocation(557, 353);
    }

    @Override
    public void move() { //角色移動、確認碰撞、進關動畫、確認角色狀態
        if (!pausing) {
            character.move();
            character.hurt(checkCollide());
            gameMap.move();
            checkState();
            if (monsterBuilder != null) {
                _scores.setValue(monsterBuilder.checkScore() + currentScore);
            }
            levelBeginWordsAnimationPlay();
            if(character.checkIfArrowOnWall()){
                audioController.play(AudioType.OnWall);
            }
        }
    }

    @Override
    public void show() { //顯示所有物件
        // 順序為貼圖順序
        _background.show();
        character.show();
        if (gameMap.getLevel() == 1) {
            _message.show();
            _message_attack.show();
            _message_jump.show();
        }
        _button.show();
        gameMap.show();
        _pauseButton.show();
        refreshButton.show();
        killMonster.show();
        _scores.show();
        showLife();
        showLevelBeginWords();
    }

    @Override
    public void release() { //釋放記憶體
        _background.release();
        _scores.release();
        _button.release();
        character.release();
        _message.release();
        _message_attack.release();
        _message_jump.release();
        audioController.release();
        gameMap.release();
        _pauseButton.release();
        refreshButton.release();
        killMonster.release();
        _black1.release();
        _black2.release();
        _black3.release();
        if (character.life == 3) {
            _life1.release();
            _life2.release();
            _life3.release();
```

```
            } else {
                if (_life2 != null) {
                    _life2.release();
                }
                if (_life3 != null) {
                    _life3.release();
                }
            }
            s.release();
            t.release();
            a.release();
            g.release();
            e.release();
            level.release();

            _background = null;
            _scores = null;
            _button = null;
            character = null;
            _message = null;
            _message_attack = null;
            _message_jump = null;
            audioController = null;
            gameMap = null;
            _pauseButton = null;
            refreshButton = null;
            killMonster = null;
            changelevel = null;
            _black1 = null;
            _black2 = null;
            _black3 = null;
            _life1 = null;
            _life2 = null;
            _life3 = null;
            s = null;
            t = null;
            a = null;
            g = null;
            e = null;
            level = null;
        }

        @Override //按下實體按鍵(用不到)
        public void keyPressed(int keyCode) {
            // TODO Auto-generated method stub
        }

        @Override //放開實體按鍵(用不到)
        public void keyReleased(int keyCode) {
            // TODO Auto-generated method stub
        }

        @Override
        public void orientationChanged(float pitch, float azimuth, float roll) {
//            if (roll > 15 && roll < 60 && _cx > 50)
//                _cx -= 2;
//            if (roll < -15 && roll > -60 && _cx + _cloud.getWidth() < 500)
//                _cx += 2;
        }

        @Override
        public void accelerationChanged(float dX, float dY, float dZ) {
            // TODO Auto-generated method stub
        }

        @Override
        public boolean pointerPressed(Pointer actionPointer, List<Pointer> pointers) { //觸碰螢幕
            _message.setVisible(false);
            _message_attack.setVisible(false);
```

```
        _message_jump.setVisible(false);
        int touchX = actionPointer.getX();
        int touchY = actionPointer.getY();
        if (touchX > 560 && touchX < 590 && touchY < 20) {
            character.setLocation(300, 182);
        }
        if (touchX > 530 && touchX < 560 && touchY < 20) {
            monsterBuilder.killAll();
        }
        if (touchX > 590 && touchY < 20) {
            pause();
        }
        if (touchX > 325 && touchY > 185) {
            if (character.getJumpHeight() == 5) {
                character.jump(5);
                audioController.play(AudioType.JUMP);
            }
        }
        if (touchX > 325 && touchY < 185 && touchY > 20) {
            if(character.life>0){
                character.shoot();
                audioController.play(AudioType.ATTACK);
            }
        }

        if (touchX > _button.getX() && touchX < _button.getX() + _button.getWidth() &&
                touchY > _button.getY() && touchY < _button.getY() + _button.getHeight()) {
            _grab = true;
        } else {
            _grab = false;
            if (_pointer1 == null) {
                _pointer1 = actionPointer;
            } else if (_pointer2 == null) {
                _pointer2 = actionPointer;
            }
        }
        return true;
    }

    @Override
    public boolean pointerMoved(Pointer actionPointer, List<Pointer> pointers) { //滑動螢幕
        if (_grab) {
            int moveX = actionPointer.getX();
            int moveY = actionPointer.getY();
            if (moveX > _button.getX()) {
                if (gameMap.isWalkable_right(character.getX() + 5, character.getY())) {
                    character.setLocation(character.getX() + 5, character.getY());
                    character.setDirection("Right");
                }
                if (gameMap.isWalkable_down(character.getX(), character.getY() + 5)) {
                    if (!character.isJumping()) {
                        character.jump(0);
                    }
                }
            } else if (moveX < _button.getX() + _button.getWidth()) {
                if (gameMap.isWalkable_left(character.getX() - 5, character.getY())) {
                    character.setLocation(character.getX() - 5, character.getY());
                    character.setDirection("Left");
                }
                if (gameMap.isWalkable_down(character.getX(), character.getY() + 5)) {
                    if (!character.isJumping()) {
                        character.jump(0);
                    }
                }
            }
        }

        if (_pointer1 != null && _pointer2 != null) {
            if (actionPointer.getID() == _pointer1.getID()) {
```

```
                _pointer1 = actionPointer;
            } else if (actionPointer.getID() == _pointer2.getID()) {
                _pointer2 = actionPointer;
            }
        }
        return false;
    }

    @Override
    public boolean pointerReleased(Pointer actionPointer, List<Pointer> pointers) { //放開螢幕
        _grab = false;
        String d = character.getDirection();
        if (d.contains("Right") && !d.equals("shootingRight")) {
            character.setDirection("standingRight");
        } else if (d.contains("Left") && !d.equals("shootingLeft")) {
            character.setDirection("standingLeft");
        }
        return false;
    }

    @Override
    public void pause() { //遊戲暫停
        audioController.pause();
        pauseState(5, this, null);
        pausing = true;
    }

    @Override
    public void resume() { //恢復遊戲
        audioController.resume();
        pausing = false;
    }

    public boolean checkCollide() { //確認角色碰撞
        for (GameMonster monster : MonsterList) {
            if (character.getX() > monster.getX() + monster.getWidth() || character.getX()+character.getWidth() <
monster.getX()) {
                continue;
            } else if (character.getY() > monster.getY()+monster.getHeight() || character.getY()+character.getHeight() <
monster.getY()) {
                continue;
            } else {
                if (!monster.isKilled()) {
                    switch (character.life) {
                        case 2:
                            if (_life1 != null) {
                                _life1.release();
                                _life1 = null;
                                audioController.play(AudioType.HURT);
                            }
                            break;
                        case 1:
                            if (_life2 != null) {
                                _life2.release();
                                _life2 = null;
                                audioController.play(AudioType.HURT);
                            }
                            break;
                        case 0:
                            if (_life3 != null) {
                                _life3.release();
                                _life3 = null;
                                audioController.play(AudioType.HURT);
                                audioController.play(AudioType.DIE);
                            }
                            break;
                        }
                        return true;
                    }
```

```
            }
            }
        return false;
}

private void checkState () { //確認遊戲狀態
        int monsterClear = 0;
        for (GameMonster monster : MonsterList) {
            if (!monster.isKilled() && monster.getVisible()) {
                monsterClear += 1;
            }else if(monster.isKilled() && monster.getVisible()){
                monsterClear += 1;
            }
        }
        if (monsterClear == 0) {
            _scores.setValue(monsterBuilder.checkScore() + currentScore);
            changelevel.put("score", _scores.getValue());
            changeState(Game.CHANGE_STATE, changelevel);
        }
        if (character.dead) {
            changelevel.put("score", currentScore);
            changeState(Game.OVER_STATE, changelevel);
        }
}

private void showLife () { //顯示生命值
        _black1.show();
        _black2.show();
        _black3.show();
        if (_life1 != null) {
            _life1.show();
        }
        if (_life2 != null) {
            _life2.show();
        }
        if (_life3 != null) {
            _life3.show();
        }
}

private void showLevelBeginWords () { //顯示進關動畫的物件
        s.show();
        t.show();
        a.show();
        g.show();
        e.show();
        level.show();
}

private void levelBeginWordsAnimationPlay () { //顯示進關動畫
        if (beginWordsAnimationCount > 0) {
            if (beginWordsAnimationCount <= 30 && s != null && s.getX() > 271) {
                s.setLocation(s.getX() - 15, s.getY());
            }
            if (beginWordsAnimationCount <= 28 && t != null && t.getX() > 296) {
                t.setLocation(t.getX() - 15, t.getY());
            }
            if (beginWordsAnimationCount <= 26 && a != null && a.getX() > 321) {
                a.setLocation(a.getX() - 15, a.getY());
            }
            if (beginWordsAnimationCount <= 24 && g != null && g.getX() > 346) {
                g.setLocation(g.getX() - 15, g.getY());
            }
            if (beginWordsAnimationCount <= 22 && e != null && e.getX() > 371) {
                e.setLocation(e.getX() - 15, e.getY());
            }
            if (beginWordsAnimationCount <= 20 && level != null && level.getX() > 396) {
                level.setLocation(level.getX() - 15, level.getY());
            }
```

```
                beginWordsAnimationCount--;
            } else if (beginWordsAnimationCount > -80) {
                if (beginWordsAnimationCount <= -10 && s != null && s.getX() > -30) {
                    s.setLocation(s.getX() - 15, s.getY());
                }
                if (beginWordsAnimationCount <= -12 && t != null && t.getX() > -30) {
                    t.setLocation(t.getX() - 15, t.getY());
                }
                if (beginWordsAnimationCount <= -14 && a != null && a.getX() > -30) {
                    a.setLocation(a.getX() - 15, a.getY());
                }
                if (beginWordsAnimationCount <= -16 && g != null && g.getX() > -30) {
                    g.setLocation(g.getX() - 15, g.getY());
                }
                if (beginWordsAnimationCount <= -18 && e != null && e.getX() > -30) {
                    e.setLocation(e.getX() - 15, e.getY());
                }
                if (beginWordsAnimationCount <= -20 && level != null && level.getX() > -30) {
                    level.setLocation(level.getX() - 15, level.getY());
                }
                beginWordsAnimationCount--;
            }

        }
    }
```