QQ截图未命名

**18级软件工程专业 初级软件实作课**

**实作报告**

**实作名称：合金弹头**

**组长：赵治球**

**课题组成员：劳庆昌、罗通、符齐贤**

**完成日期：2020.7**

**华南师范大学软件学院**

**2020.7**

目录

[1、实作项目简介 3](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc15152_WPSOffice_Level1)

[2、 前期准备 3](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc27525_WPSOffice_Level1)

[3、项目设计 4](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc30427_WPSOffice_Level1)

[3.1总体架构设计（系统结构图 / 分析类图） 4](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc9096_WPSOffice_Level2)

[3.2类设计（类图） 4](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc1633_WPSOffice_Level2)

[3.2.1. controller包 5](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc25707_WPSOffice_Level3)

[3.2.2. element包 5](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc13054_WPSOffice_Level3)

[3.2.3. GameStart包 11](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc22494_WPSOffice_Level3)

[3.2.4. manager包 1](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc5992_WPSOffice_Level3)2

[3.2.5. show包 13](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc28283_WPSOffice_Level3)

[4、系统实现 13](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc716_WPSOffice_Level1)

[4.1、界面 13](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc6248_WPSOffice_Level2)

[4.1.1GameJFrame类 13](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc29020_WPSOffice_Level3)

[4.2、控制 16](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc29520_WPSOffice_Level2)

[4.2.1 GameThread类 16](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc14557_WPSOffice_Level3)

[4.3、资源管理 20](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc22186_WPSOffice_Level2)

[4.3.1ElementManager类 20](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc17216_WPSOffice_Level3)

[5.游戏画面展示 22](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc15118_WPSOffice_Level1)

[图1. 游戏开始界面 23](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc5922_WPSOffice_Level2)

[图2. 解救人质界面 23](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc992_WPSOffice_Level2)

[图3. 飞机轰炸界面 24](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc6245_WPSOffice_Level2)

[图4. BOSS出现界面 24](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc29070_WPSOffice_Level2)

[图5. 小兵出现界面 25](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc27928_WPSOffice_Level2)

[6、总结 25](file:///C:\Users\lenovo\Desktop\生死救援%20初级软件实作报告（组长姓名）参考模板.doc#_Toc17371_WPSOffice_Level1)

# 1、实作项目简介

《合金弹头》是一款曾风靡全国的小游戏，游戏类型为射击类游戏，操作简单，趣味性强。在我们的设定中主角是一个极富正义感的中美混血男性，游戏背景是，部分无辜群众被邪恶的军阀所绑架控制，主角凭借自己一身过人本领与邪恶的军阀对抗，主角需要击倒军阀的小兵来解救被军阀控制的人质，击倒小兵和解救人质后，在游戏最后主角需要直接面对军阀的统帅，也就是最终的BOSS，主角需要通过击倒BOSS来摧毁整个军阀。游戏中，玩家需要操作主角移动和击倒小兵去拯救人质，玩家每拯救一个人质，可以获得奖品，奖品随机出现，可以为主角回血或者增加主角的攻击力，玩家操作主角打倒最终BOSS即可获得游戏胜利

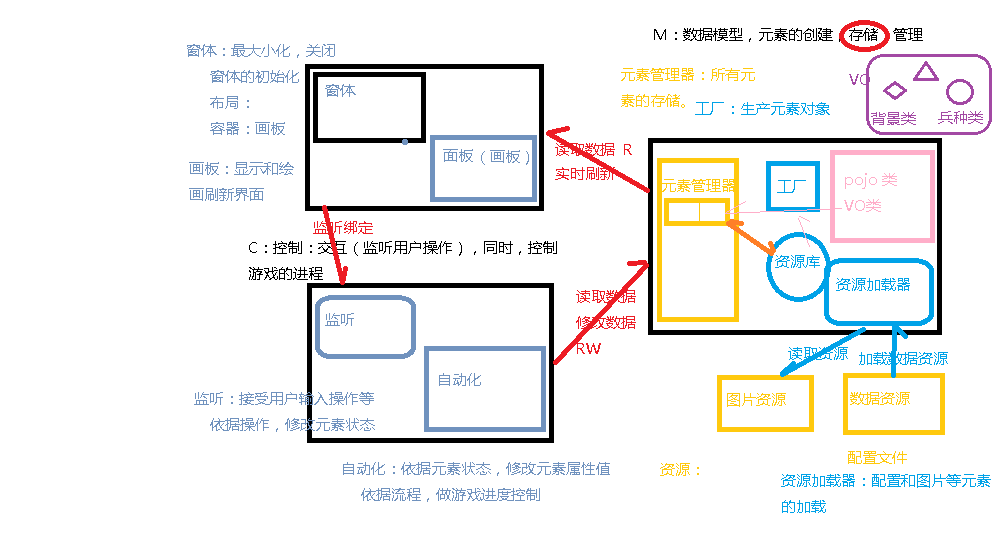
项目使用Java编程语言，采用MVC设计模式。

# 前期准备

在进行游戏设计之前我们小组全体成员进行了一周的初级软件实作学习，充分了解了MVC设计模式，进一步拓展了java的相关知识，学会了基本的游戏设计与制作本领。在课堂上，我们紧跟老师的脚步，和老师一起敲代码，积累了开发经验，遇到不懂的地方就在课堂上或者群里提问，并且时常在课后通过观看回放来巩固知识。在了解实作题目和内容后，组员们积极交流，明确各自分工，投入到各自工作中

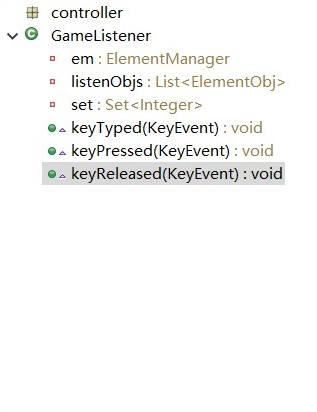
# 3、项目设计

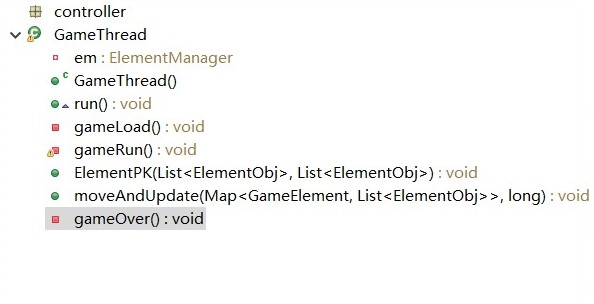
## 3.1总体架构设计（系统结构图 / 分析类图）



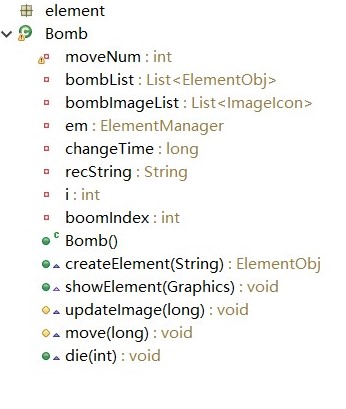
## 3.2类设计（类图）

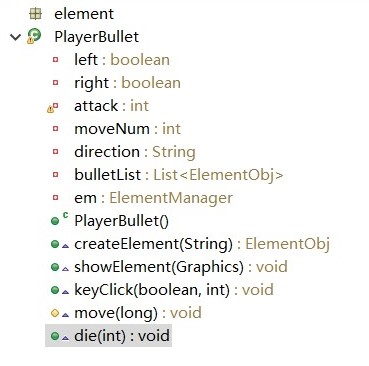
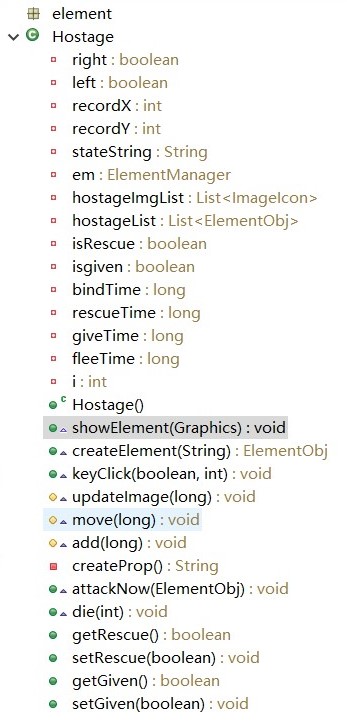
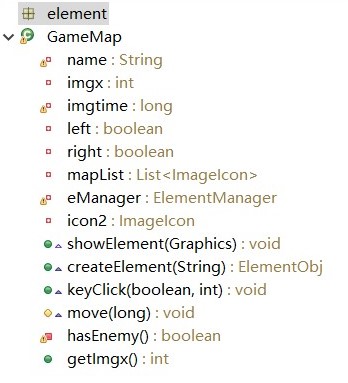
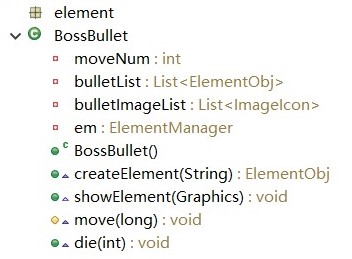
### 3.2.1.controller包

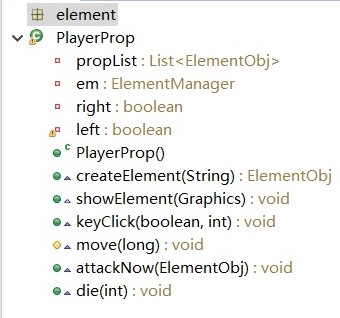




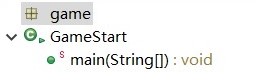
**3.2.2.****element包**



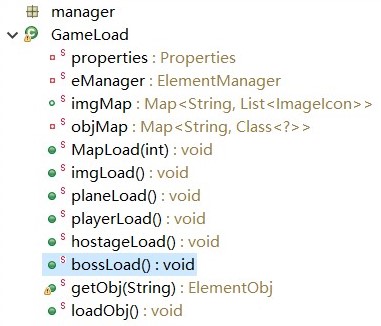
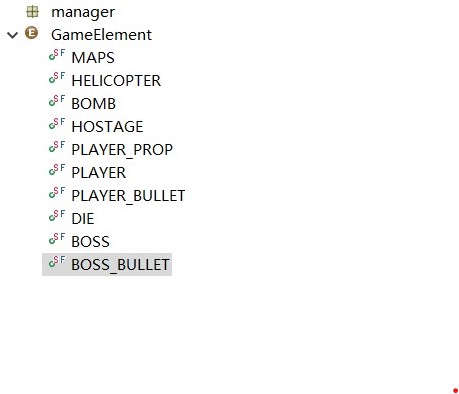
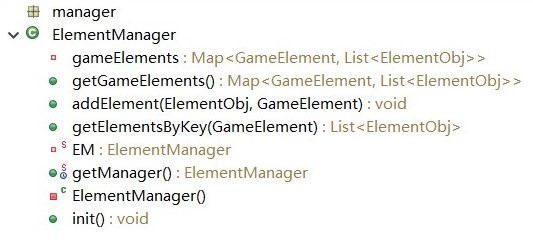




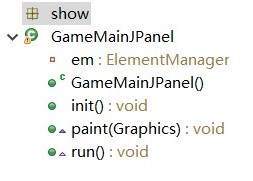
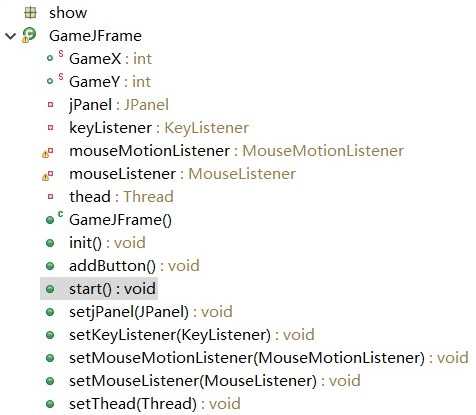
**3.2.3.****GameStart包**



**3.2.4.****manager包**



**3.2.5.****show包**



# 4、系统实现

## 4.1、界面

**4.1.1 GameJFrame类**

package show;

import java.awt.event.KeyListener;

import java.awt.event.MouseListener;

import java.awt.event.MouseMotionListener;

import javax.swing.JFrame;

import javax.swing.JPanel;

/\*\*

\* @说明 游戏窗体 主要实现的功能：关闭，显示，最大最小化

\* @author renjj

\* @功能说明 需要嵌入面板,启动主线程等等

\* @窗体说明 swing awt 窗体大小（记录用户上次使用软件的窗体样式）

\*

\* @分析 1.面板绑定到窗体

\* 2.监听绑定

\* 3.游戏主线程启动

\* 4.显示窗体

\*/

public class GameJFrame extends JFrame{

public static int GameX = 800;//游戏窗口的长度

public static int GameY = 480;//游戏窗口的宽度

private JPanel jPanel =null; //正在实现的面板

private KeyListener keyListener=null;//键盘监听

private MouseMotionListener mouseMotionListener=null; //鼠标监听

private MouseListener mouseListener=null;

private Thread thead=null; //游戏主线程

public GameJFrame() {

init();

}

public void init() {

this.setSize(GameX, GameY); //设置窗体大小

this.setTitle("合金弹头");

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//设置退出并且关闭

this.setLocationRelativeTo(null);//屏幕居中显示

setResizable(false);

//还可以继续设计窗口的其他属性

}

/\*窗体布局: 可以存档，读档，使用button 给大家扩展的\*/

public void addButton() {

// this.setLayout(manager);//布局格式，可以添加控件

}

/\*\*

\* 启动方法

\*/

public void start() {

if(jPanel!=null) {

this.add(jPanel);

}

if(keyListener !=null) {

this.addKeyListener(keyListener);

}

if(thead !=null) {

thead.start();//启动线程

}

this.setVisible(true);//显示界面

// 如果jp 是 runnable的 子类实体对象

// 如果这个判定无法进入就是 instanceof判定为 false 那么 jpanel没有实现runnable接口

if(this.jPanel instanceof Runnable) {

// 已经做类型判定，强制类型转换不会出错

// new Thread((Runnable)this.jPanel).start();

Runnable run=(Runnable)this.jPanel;

Thread th=new Thread(run);

th.start();//

System.out.println("是否启动");

}

}

/\*set注入：等大家学习ssm 通过set方法注入配置文件中读取的数据;讲配置文件

\* 中的数据赋值为类的属性

\* 构造注入：需要配合构造方法

\* spring 中ioc 进行对象的自动生成，管理。

\* \*/

public void setjPanel(JPanel jPanel) {

this.jPanel = jPanel;

}

public void setKeyListener(KeyListener keyListener) {

this.keyListener = keyListener;

}

public void setMouseMotionListener(MouseMotionListener mouseMotionListener) {

this.mouseMotionListener = mouseMotionListener;

}

public void setMouseListener(MouseListener mouseListener) {

this.mouseListener = mouseListener;

}

public void setThead(Thread thead) {

this.thead = thead;

}

}

## 4.2、控制

### 4.2.1 GameThread类

package controller;

import java.util.List;

import java.util.Map;

import element.ElementObj;

import manager.ElementManager;

import manager.GameElement;

import manager.GameLoad;

/\*\*

\* @说明 游戏的主线程，用于控制游戏加载，游戏关卡，游戏运行时自动化

\* 游戏判定；游戏地图切换 资源释放和重新读取。。。

\* @author renjj

\* @继承 使用继承的方式实现多线程(一般建议使用接口实现)

\*/

public class GameThread extends Thread{

private ElementManager em;

public GameThread() {

em=ElementManager.getManager();

}

@Override

public void run() {//游戏的run方法 主线程

while(true) { //扩展,可以讲true变为一个变量用于控制结束

// 游戏开始前 读进度条，加载游戏资源(场景资源)

gameLoad();

// 游戏进行时 游戏过程中

gameRun();

// 游戏场景结束 游戏资源回收(场景资源)

gameOver();

try {

sleep(50);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 游戏的加载

\*/

private void gameLoad() {

//加载游戏所需的所有图片

GameLoad.imgLoad();

//加载地图（参数可以为变量，每一关都重新加载）

GameLoad.MapLoad(5);

//加载直升飞机

GameLoad.planeLoad();

//加载主角

GameLoad.playerLoad(); //也可以带参数，单机还是双人

//加载人质

GameLoad.hostageLoad();

//加载敌人/NPC等

//全部加载完成，游戏才启动

}

/\*\*

\* @说明 游戏进行时

\* @任务说明 游戏过程中需要做的事情：1.自动化玩家的移动，碰撞，死亡

\* 2.新元素的增加(NPC死亡后出现道具)

\* 3.暂停等等。。。。。

\* 先实现主角的移动

\* \*/

private void gameRun() {

long gameTime=5L;//int类型就可以

while(true) {// 预留扩展 true可以变为变量，用于控制管关卡结束等

Map<GameElement, List<ElementObj>> all = em.getGameElements();

//获取主角元素

List<ElementObj> player=em.getElementsByKey(GameElement.PLAYER);

//获取道具元素

List<ElementObj> playerProps=em.getElementsByKey(GameElement.PLAYER\_PROP);

//获取敌人元素

// List<ElementObj> enemys = em.getElementsByKey(GameElement.ENEMY);

//获取炸弹元素

List<ElementObj> bombs=em.getElementsByKey(GameElement.BOMB);

//获取BOSS元素

List<ElementObj> bossbullets=em.getElementsByKey(GameElement.BOSS\_BULLET);

//获取子弹元素

List<ElementObj> bullets = em.getElementsByKey(GameElement.PLAYER\_BULLET);

//获取人质元素

List<ElementObj> hostages=em.getElementsByKey(GameElement.HOSTAGE);

//获取地图元素

List<ElementObj> maps = em.getElementsByKey(GameElement.MAPS);

//游戏元素自动化方法

moveAndUpdate(all,gameTime);

// //将主角子弹与敌人做碰撞判定

// ElementPK(bullets,enemies);

//将主角子弹与地图做碰撞判定（是否超出界面）

ElementPK(maps, bullets);

//将主角与人质做判定

ElementPK(player, hostages);

//将主角与道具做判定

ElementPK(player, playerProps);

//主角与飞机扔的炸弹做判定

ElementPK(player, bombs);

//唯一的时间控制

gameTime++; //1s自增了100次

try {

sleep(10);//默认理解为 1秒刷新100次

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

}

/\*\*

\* 为两种不同类型的元素的所有对象开启碰撞检测

\* @param listA

\* @param listB

\*/

public void ElementPK(List<ElementObj> listA,List<ElementObj> listB) {

//使用循环，做一对一判定，如果为真，就设置2个对象的死亡状态

for(int i=0;i<listA.size();i++) {

ElementObj objA=listA.get(i);

for(int j=0;j<listB.size();j++) {

ElementObj objB=listB.get(j);

if(objA.pk(objB)) {

// 问题： 如果是boss，那么也一枪一个吗？？？？

// 将 setLive(false) 变为一个受攻击方法，还可以传入另外一个对象的攻击力

// 当受攻击方法执行时，如果血量减为0，再进行设置生存状态为 false

//留给大家扩展

objA.attackNow(objB);

objB.attackNow(objA);

// if (objB instanceof Bomb) {//为炸弹类时

// if (objB.getY()>=380) {

// objA.setLive(false);

// }

// }

//objB.setLive(false);

break;

}

}

}

}

//游戏元素自动化方法

public void moveAndUpdate(Map<GameElement, List<ElementObj>> all,long gameTime) {

// GameElement.values();//隐藏方法 返回值是一个数组,数组的顺序就是定义枚举的顺序

for(GameElement ge:GameElement.values()) {

List<ElementObj> list = all.get(ge);

// 编写这样直接操作集合数据的代码建议不要使用迭代器。

// for(int i=0;i<list.size();i++) {

for(int i=list.size()-1;i>=0;i--){

ElementObj obj=list.get(i);//读取为基类

if(!obj.getLive()) {//如果死亡

// list.remove(i--); //可以使用这样的方式

// 启动一个死亡方法(方法中可以做事情例如:死亡动画/掉装备/人质奔跑)

obj.die(i);//需要大家自己补充

// list.remove(i); //应该在每个元素类的die()里remove

// continue;

}

obj.model(gameTime);//调用的模板方法model()而不是move()

}

}

}

/\*\*游戏切换关卡\*/

private void gameOver() {

}

// public void load() {

//// 图片导入

// ImageIcon icon=new ImageIcon("image/tank/play1/player1\_up.png");

// ElementObj obj=new Play(100,100,50,50,icon);//实例化对象

//// 讲对象放入到 元素管理器中

//// em.getElementsByKey(GameElement.PLAY).add(obj);

// em.addElement(obj,GameElement.PLAY);//直接添加

// }

}

## 4.3、资源管理

### 4.3.1ElementManager类

package manager;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import element.ElementObj;

/\*\*

\* @说明 本类是元素管理器，专门存储所有的元素，同时，提供方法

\* 给予视图和控制获取数据

\* @author renjj

\* @问题一：存储所有元素数据，怎么存放？ list map set 3大集合

\* @问题二：管理器是视图和控制要访问，管理器就必须只有一个，单例模式

\*/

public class ElementManager {

/\*

\* String 作为key 匹配所有的元素 play -> List<Object> listPlay

\* enemy ->List<Object> listEnemy

\* 枚举类型，当做map的key用来区分不一样的资源，用于获取资源

\* List中元素的泛型 应该是 元素 基类

\* 所有元素都可以存放到 map集合中，显示模块只需要获取到 这个map就可以

\* 显示是有的界面需要显示的元素(调用元素基类的 showElement())

\*/

private Map<GameElement,List<ElementObj>> gameElements;

// 本方法一定不够用

public Map<GameElement, List<ElementObj>> getGameElements() {

return gameElements;

}

// 添加元素(多半由加载器调用)

public void addElement(ElementObj obj,GameElement ge) {

// List<ElementObj> list = gameElements.get(ge);

// list.add(obj);

gameElements.get(ge).add(obj);//添加对象到集合中，按key值就行存储

}

// 依据key返回 list集合，取出某一类元素

public List<ElementObj> getElementsByKey(GameElement ge){

//一个延伸的小问题：String str = new String("abcd");

//在这语句中请问共产生几个变量？分别是什么？分别存储在哪里？

//一共产生3个变量：1.对象引用str，存储在栈 2.字符串常量"abcd"，存储在常量池（静态存储区）

//3.new出来的一个字符串变量String，存储在堆

//当一块内存没有任何一个引用指向的时候，会被GC回收。

return gameElements.get(ge);

}

/\*\*

\* 单例模式：内存中有且只有一个实例。

\* 饿汉模式-是启动就自动加载实例

\* 饱汉模式-是需要使用的时候才加载实例

\*

\* 编写方式：

\* 1.需要一个静态的属性(定义一个常量) 单例的引用

\* 2.提供一个静态的方法(返回这个实例) return单例的引用

\* 3.一般为防止其他人自己使用(类是可以实例化),所以会私有化构造方法

\* ElementManager em=new ElementManager();

\*/

private static ElementManager EM=null; //引用

// synchronized线程锁->保证本方法执行中只有一个线程

public static synchronized ElementManager getManager() {

if(EM == null) {//控制判定

EM=new ElementManager();

}

return EM;

}

private ElementManager() {//私有化构造方法

init(); //实例化方法

}

// static { //饿汉实例化对象 //静态语句块是在类被加载的时候直接执行

// EM=new ElementManager(); //只会执行一次

// }

/\*\*

\* 本方法是为 将来可能出现的功能扩展，重写init方法准备的。

\*/

public void init() {//实例化在这里完成

// hashMap hash散列

gameElements=new HashMap<GameElement,List<ElementObj>>();

// 将每种元素集合都放入到 map中

// gameElements.put(GameElement.PLAY, new ArrayList<ElementObj>());

// gameElements.put(GameElement.MAPS, new ArrayList<ElementObj>());

// gameElements.put(GameElement.ENEMY, new ArrayList<ElementObj>());

// gameElements.put(GameElement.BOSS, new ArrayList<ElementObj>());

//当使用循环读取的方式时，是按枚举类型变量在枚举类中的声明顺序来进行的

for(GameElement ge:GameElement.values()) { //通过循环读取枚举类型的方式添加集合

gameElements.put(ge,new ArrayList<ElementObj>());

}

// 道具，子弹，爆炸效果，死亡效果

}

}

# 5.游戏画面展示



图1.游戏开始界面



图2.解救人质界面



图3.飞机轰炸界面



图4.BOSS出现界面



图5.小兵出现界面

# 6、总结

## 6.1设计亮点

### 6.1.1角色人物多样，包括了主角、人质、小兵、boss和飞机等独立角色，游戏可玩性高

### 6.1.2游戏画面丰富、观赏度高，有众多玩家操作主角与其他人物的触发效果

### 6.1.3游戏中主角可由玩家操控自由活动，灵活度高

### 6.1.4游戏中小兵、boss、飞机和人质等具有较好的自动化运动

## 6.2不足

### 6.2.1游戏内容还较少，游玩时间较短

### 6.2.2目前还只有一个关卡，只有一个boss

### 6.2.3由于时间原因，游戏中的音效暂时还没有实现