飞机大战技术描述报告

对各个类进行分析，以此来设计超类  
大敌机：有坐标 x、y，宽，高，生命数，向下移动速度的属性。发射子弹，移动，越界，碰撞，返回分数等行为，  
小敌机：有坐标 x、y，宽，高，生命数，向下移动速度的属性。发射子弹，移动，越界，碰撞，返回分数等行为  
侦察机：有坐标 x、y，宽，高，生命数，x、y方向移动速度的属性。发射子弹，移动，越界，碰撞，返会奖励等行为  
boss机：有坐标 x、y，宽，高，生命数，向下移动速度的属性。发射子弹，移动，越界，碰撞，返回分数和奖励等行为  
英雄机：有坐标 x、y，宽，高，生命数，分数，火力值属性。发射子弹，移动，碰撞等行为  
天空：有坐标 x、y，宽，高，向下移动速度的属性。移动行为。  
子弹：有坐标 x、y，宽，高，向下或者向上移动速度的属性。移动行为，碰撞，越界等。  
可以看出上面的属性和行为有太多的重复，在此设计一个超类有上面共有的属性和方法

1.程序运行后有背景音，暂停时没有（鼠标移出游戏界面暂停）  
2.一发子弹可以打去一条命  
3.敌人在被击毁后有爆炸（包括声音和画面）  
4.英雄机在被子弹击中后和死亡都有音效

首先确定，所有的飞机天空子弹都是一张张背景透明的图片，每一此改变它的坐标在将它画到画板上以此形成了连续移动的感觉，就和电影是一个原理。Java的画板类JPanel就为我们提供了将图片显示出来的功能。

加载图片类Images

设计一个加载图片的类，将图片加载到方法区，这样图片只用读取一次

public class Images {

public static BufferedImage sky;

public static BufferedImage[ ] bullets;

public static BufferedImage[ ] bossairplanes;

public static BufferedImage[ ] heros;

public static BufferedImage[ ] airplanes;

public static BufferedImage[ ] bigairplanes;

public static BufferedImage[ ] bees;

static{

//天空图片的加载

sky = readImage("background1.png");

//英雄机图片的加载

heros = new BufferedImage[2];

heros[0] = readImage("hero0.png");

heros[1] = readImage("hero1.png");

//子弹图片的加载

bullets = new BufferedImage[2];

bullets[0] = readImage("bullet0.png");

bullets[1] = readImage("bullet1.png");

//boss机图片的加载

bossairplanes = new BufferedImage[5];

bossairplanes[0] = readImage("boss.png");

//小敌机图片的加载

airplanes = new BufferedImage[5];

airplanes[0] = readImage("airplane0.png");

//大敌机的加载

bigairplanes = new BufferedImage[5];

bigairplanes[0] = readImage("bigairplane0.png");

//侦察机图片的加载

bees = new BufferedImage[5];

bees[0] = readImage("bee0.png");

//爆破图片的加载

for(int i=1;i<5;i++){

bees[i] = readImage("bom"+i+".png");

airplanes[i] = readImage("bom"+i+".png");

bigairplanes[i] = readImage("bom"+i+".png");

bossairplanes[i] = readImage("bom"+i+".png");

}

}

//读取图片到内存

public static BufferedImage readImage(String fileName){

try{

BufferedImage img = ImageIO.read(FlyingObject.class.getResource(fileName));

return img;

}catch(Exception e){

e.printStackTrace();

throw new RuntimeException();

}

}

}

设计抽象类超类FlyingObject

public abstract class FlyingObject {

//共同属性

protected int width;

protected int height;

protected int x;

protected int y;

protected int life;

//状态常量

public static final int LIFE = 0;

public static final int DEAD = 1;

public static final int REMOVE = 2;

//当前状态

protected int state = LIFE;

//为敌人提供的构造方法，

FlyingObject(int width,int height,int life){

this.width = width;

this.height = height;

this.x = (int)(Math.random()\*(World.WIDTH-width));

this.y = -height;

this.life = life;

}

//为天空和英雄机子弹提供的构造方法

FlyingObject(int width,int height,int x,int y){

this.width = width;

this.height = height;

this.x = x;

this.y = y;

}

//生成子弹组数的方法

public Bullet[] shoot(){

return new Bullet[0];

}

//判断碰撞的方法

public boolean hit(FlyingObject other){

int x1 = other.x - this.width;

int x2 = other.x + other.width;

int y1 = other.y - this.height;

int y2 = other.y + other.height;

return x>=x1 && x<=x2 && y>=y1 && y<=y2;

}

//生命减一

public void subtractLife(){

life--;

}

//判断是否活着

public boolean isLife(){

return state==LIFE;

}

//判断是否死了，后面有用

public boolean isDead(){

return state==DEAD;

}

//判断状态是否为移除

public boolean isRemove(){

return state==REMOVE;

}

//将状态改为DEAD

public void goDead(){

state = DEAD;

}

//在画板上画出图片，getImage()是后面自己写的一个获取图片数据的方法

public void paintObject(Graphics g){

g.drawImage(getImage(),x,y,null);

}

//移动抽象方法，因为每个对象移动的方法不一样，又都有移动，所以写为抽象方法

public abstract void step();

//获取图片的抽象方法，为什么设置为抽象方法如上

public abstract BufferedImage getImage();

//判断是否越界的抽象方法，如上

public abstract boolean outBround();

}

**旋风无敌大大大招**

//这里实现里监听键盘的接口KeyListener，重写按键时间

@Override

public void keyPressed(KeyEvent e) {

//当状态为RUNNING时

if(state==RUNNING){

//判断是否按下空格键

switch(e.getKeyCode()){

case KeyEvent.VK\_SPACE:

//按下空格键后播放爆炸音效同时清空所有地方对象

all\_bomb.play();

enemies = new FlyingObject[0];

enemiesBullets = new Bullet[0];

break;

}

}