# Java素选期末报告

姓名：周廷宇 专业：软件工程 学号：20151120028 项目名称：飞机大战

### 项目定位与设计：

我们打算用java开发一个小型飞机游戏，同已有的飞机类游戏一样，玩家可以操控飞机的移动和射击，以通关的方式进行游戏，击落敌人可以得分，最后得分最高者被记录。

项目实现思路：

利用java swing制作游戏界面，不断调用JPanel的PaintComponent方法达到动画效果，游戏中的敌人和自己用贴图完成。

### 项目分工：

由于组内有两个人，所以讲一下我个人的分工是除了界面、基本框架、游戏素材、音频调用、图片管理以外的所有部分，其中主要包括游戏所有素材的算法设计、功能实现以及碰撞检测的修改作业。

### 项目主要算法讲解：

飞机大战类的游戏是不同于战旗类游戏的一个动态游戏，制作的时候比较困难，需要考虑游戏过程中的诸多因素，以防游戏的不智能、卡机等问题，下面我来讲解一下游戏中的几个主要算法和问题处理的机制。

1. 重绘算法。

将重绘的方法放进一个线程中，目的是不断重绘画板，如果玩家阵亡则停止重绘。

**public** **class** PaintThread **extends** Thread

{

**public** **void** run()

{

**while**(**true**)

{

**if**(me.isOver()){

over();

repaint();

repaint();

repaint();

**break**;

}

repaint();

**try** {

Thread.*sleep*(50);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

}

}

1. 游戏进程辅助算法。

**public** **void** drawfirechange(Graphics g) //如果游戏过程中有掉落道具则画出

{

**for**(**int** j=0;j<props.size();j++)

{

ChangeFireProp cfp=(ChangeFireProp)props.get(j);

cfp.drawSelf(g);

}

}

**public** **void** drawplayershoot(Graphics g)//画出玩家发射的子弹

{

**for**(**int** j=0;j<playerBullets.size();j++)

{

Bullet b=playerBullets.get(j);

b.drawSelf(g);

}

}

**public** **void** drawenemyone(Graphics g,**long** time,**boolean** ishasprop,**int** type)

{

**for**(**int** j=0;j<enemy.size();j++) //画出敌机一号，time是游戏进程时间，ishasprop决定该敌人被击落后是否有奖励，type是敌人的攻击方式

{

Plane p=enemy.get(j);

**if**(p.life<=0&&ishasprop){

**this**.cfp=**new** ChangeFireProp(p.xpos,p.ypos,firechange,props);

props.add(cfp);

}

p.drawSelf(g);

**if**(time%p.frequency==1)

p.shoot(type,5);

}

}

**public** **void** drawenemytwo(Graphics g,**long** time,**boolean** ishasprop)

{

**for**(**int** j=0;j<enemy2.size();j++) //画出敌人二号，参数同上

{

Plane p=enemy2.get(j);

**if**(p.life<=0&&ishasprop){

**this**.cfp=**new** ChangeFireProp(p.xpos,p.ypos,firechange,props);

props.add(cfp);

}

p.drawSelf(g);

**if**(time%p.frequency==1)

p.shoot(1,5);

}

}

**public** **void** drawenemythree(Graphics g,**long** time,**boolean** ishasprop)

{

**for**(**int** j=0;j<enemy3.size();j++) //画出敌人三号，参数同上

{

Plane p=enemy3.get(j);

**if**(p.life<=0&&ishasprop){

**this**.cfp=**new** ChangeFireProp(p.xpos,p.ypos,firechange,props);

props.add(cfp);

}

p.drawSelf(g);

**if**(time%p.frequency==1)

p.shoot(1,5);

}

}

**public** **void** drawenemyfour(Graphics g,**long** time,**boolean** ishasprop)

{

**for**(**int** j=0;j<enemy4.size();j++) //画出敌人四号，参数同上

{

Plane p=enemy4.get(j);

**if**(p.life<=0&&ishasprop){

**this**.cfp=**new** ChangeFireProp(p.xpos,p.ypos,firechange,props);

props.add(cfp);

}

p.drawSelf(g);

**if**(time%p.frequency==1)

p.shoot(1,5);

}

}

**public** **void** drawenemyfive(Graphics g,**long** time,**boolean** ishasprop)

{

**for**(**int** j=0;j<enemy5.size();j++) //画出敌人五号，参数同上

{

Plane p=enemy5.get(j);

**if**(p.life<=0&&ishasprop){

**this**.cfp=**new** ChangeFireProp(p.xpos,p.ypos,firechange,props);

props.add(cfp);

}

p.drawSelf(g);

**if**(time%p.frequency==1)

p.shoot(1,5);

}

}

**public** **void** drawenemyshoot(Graphics g) //画出敌人的攻击

{

**for**(**int** j=0;j<enemyBullets.size();j++)

{

Bullet b=enemyBullets.get(j);

b.getplayerlocation(me);

b.drawSelf(g);

}

}

**public** **void** drawExplosion(Graphics g) //画出爆炸效果

{

**for**(**int** j=0;j<noticeAndExplosion.size();j++)

{

Plane p=(Plane)(noticeAndExplosion.get(j));

PlaneGameUtil.*drawExplosion*(p, g,**this**.noticeAndExplosion);

}

}

**public** **void** drawsinglebullet(Graphics g)//画出特殊子弹

{

**for**(**int** j=0;j<single.size();j++)

{

Bullet b=single.get(j);

b.getplayerlocation(me);

singleOneNeverLocate=**false**;

b.drawSelf(g);

}

**for**(**int** j=0;j<single2.size();j++)

{

Bullet b=single2.get(j);

b.getplayerlocation(me);

singleTwoNeverLocate=**false**;

b.drawSelf(g);

}

}

**public** **void** checkSingleToChangeNeverLocate()//用来依据容器状态改变变量值

{

**if**(single.size()==0){

singleOneNeverLocate=**true**;

}

**if**(single2.size()==0){

singleTwoNeverLocate=**true**;

}

}

**public** **void** createsinglebulletanywhere(**int** style)//随机生成子弹

{

Bullet sb=**new** Bullet(**new** Random().nextInt(1000), **new** Random().nextInt(1000), 20, style, **this**.single, **true**);

single.add(sb);

Bullet sb2=**new** Bullet(**new** Random().nextInt(1000), **new** Random().nextInt(1000), 20, style, **this**.single2, **true**);

single2.add(sb2);

}

**public** **void** createsinglebullet(**int** sbx,**int** sby,**int** sb2x,**int** sb2y,**int** type)

{

**if**(type==1){ //在某指定位置生成子弹

**if**(never){

Bullet sb=**new** Bullet(sbx, sby, 30, Constants.***SPECIAL\_BULLET2***, **this**.single, **true**);

single.add(sb);

Bullet sb2=**new** Bullet(sb2x, sb2y, 30, Constants.***SPECIAL\_BULLET2***, **this**.single2, **true**);

single2.add(sb2);

never=**false**;

}

}**else**{

Bullet sb=**new** Bullet(sbx, sby, 30, Constants.***SPECIAL\_BULLET2***, **this**.single, **true**);

single.add(sb);

Bullet sb2=**new** Bullet(sb2x, sb2y, 30, Constants.***SPECIAL\_BULLET2***, **this**.single2, **true**);

single2.add(sb2);

}

}

**public** **void** recovernever(){ //修改变量

never=**true**;

}

**public** **void** RemoveSingleBulletList() //清空子弹容器

{

single.removeAll(single);

single2.removeAll(single2);

}

**public** **void** RemoveEnemy() //清空敌人容器

{

enemy.removeAll(enemy);

enemy2.removeAll(enemy2);

enemy3.removeAll(enemy3);

enemy4.removeAll(enemy4);

enemy5.removeAll(enemy5);

}

**public** **int** getbosslife(List<Plane> enemy){ //获取boss生命值

**return** enemy.get(0).life;

}

**public** **void** over(){ //判断游戏是否结束

**if**(me.isOver()){

noticeAndExplosion.add(me);

isStop=**true**;

**int** option=EndFrame.*showDialog*(**this**, scd.getendtime(), ds.score);

**if**(option==EndFrame.***YES\_OPTION***){

bgm.stopSound();

**new** CornFrame().launchFrame();

}**else**{

System.*exit*(0);

}

}

}

**public** **void** setLocationAndChangeMovetype(List<Plane> enemy,**int** xpos, **int** ypos, **int** movetype){ //即时修改敌人坐标

**if**(changetype){

enemy.get(0).xpos=xpos;

enemy.get(0).ypos=ypos;

enemy.get(0).movetype=movetype;

changetype=**false**;

}

}

**public** **void** run() {// 游戏开始时间

*start*=starttime.getTime();

}

**public** **void** checktime(){ //在游戏进行的过程中不断检测当前时间

**if**(**this**.iscontinue){

Date processtime=**new** Date();

*time*=(processtime.getTime()-*start*)/1000;

// System.out.println(time);

}

}

**public** **int** getendtime(){ //游戏结束时间

Date endtime=**new** Date();

**long** time=(endtime.getTime()-*start*)/1000;

**return** (**int**)(time);

// System.out.println("this time is: "+time);

}

**public** **long** getthismonment(){ //返回当前时间

**return** **this**.*time*;

}

**public** **void** drawbg(Graphics g)//背景图滚动

{

g.drawImage(background.get(0), drx, dry, drx+667, dry+2000, **null**);

dry+=0.5;

**if**(dry==0){

dry=-520;

}

}

**public** **void** drawdatabj(Graphics g)//画游戏背景

{

g.drawImage(databj, databjx, databiy, 333, 1000, **null**);

}

**public** **void** shoot(**int** type,**int** number) { //飞机射击方法，对于玩家和敌机加载弹药的方式不同

**int** move=5;

**if**(**this**.style==Constants.***ME***)

{

**if**(fire){

**if**(**this**.bulletStyle==Constants.***FIRE\_ONE***){

Bullet b=**new** Bullet(**this**.xpos,**this**.ypos,10,**this**.bulletStyle,bullets,**false**);

bullets.add(b);

**new** Music("src/music/fire.wav").playSound();

}**else** **if**(**this**.bulletStyle==Constants.***FIRE\_TWO***){

Bullet b=**new** Bullet(**this**.xpos,**this**.ypos,20,**this**.bulletStyle,bullets,**false**);

bullets.add(b);

**new** Music("src/music/fire.wav").playSound();

} **else**{//this.bulletStyle==Constants.Firethree

Bullet b=**new** Bullet(**this**.xpos,**this**.ypos,50,**this**.bulletStyle,bullets,**false**);

bullets.add(b);

} /\*else if(this.bulletStyle==Constants.FireFour){

Bullet b=new Bullet(this.xpos,this.ypos,50,this.bulletStyle,bullets,false);

bullets.add(b);

//System.out.println("锟斤拷佣锟斤拷锟斤拷拥锟�");

} \*/

}

}**else**{

**if**(type==1){ //杩欓噷鐨則ype浠呮敼鍙樺瓙寮规暟閲�

**for**(**int** i=0;i<number;i++)

{

Bullet b=**new** Bullet(**this**.xpos+60,**this**.ypos+70,10,**this**.bulletStyle,bullets,**false**);

bullets.add(b);

}

}

**else** **if**(type==2){ //杩欓噷鐨則ype浠呮敼鍙樺瓙寮规暟閲�

**for**(**int** i=0;i<number;i++)

{

Bullet b=**new** Bullet(**this**.xpos+60,**this**.ypos+70,25,**this**.bulletStyle,bullets,**false**);

bullets.add(b);

}

}

}

}

**public** **void** addmove(KeyEvent e) { //玩家移动控制

**switch**(e.getKeyCode())

{

**case** 38:

up=**true**;

**break**;

**case** 40:

down=**true**;

**break**;

**case** 37:

left=**true**;

**break**;

**case** 39:

right=**true**;

**break**;

**default**:

**break**;

}

}

**public** **void** releasemove(KeyEvent e) //玩家移动控制

{

**switch**(e.getKeyCode())

{

**case** 38:

up=**false**;

**break**;

**case** 40:

down=**false**;

**break**;

**case** 37:

left=**false**;

**break**;

**case** 39:

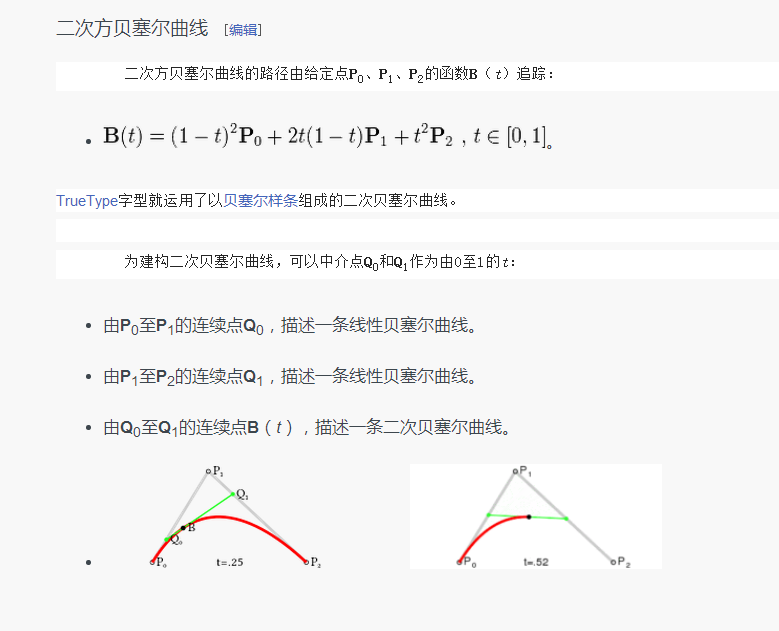
right=**false**;

**break**;

**default**: **break**;

}

}



飞机的智能飞行轨迹算法采用的是以上二次方贝塞尔曲线的变换，二次贝塞尔曲线的轨迹是由三个点确定，当确定p0和p1点，轨迹随p2点的变化而变化，这里我们p2点看成玩家，设想一下，如果p0和p1为定点，那么轨迹就有可能停止，因为玩家不一定是一直移动的，所以就这一点，我改变p0p1点的获取方式，即在游戏界面动态获取两个点，也就是说每一个时刻这两个点的坐标都在变化。

**public** **class** GetBezierCurvesPoint { //这个类是动态获取p0p1点的类

**public** **int** y1=0;

**public** **boolean** up1=**false**; //初始化设定了两个点的起始位置分别是左上角和右下角

**public** **int** y2=880;

**public** **boolean** up2=**true**;

**public** **void** movepoint() //这个方法在重绘中调用，使两个点移动起来

{

**if**(**this**.y1<=0){

up1=**false**;

y1+=5;

}**else** **if**(**this**.y1>=880){

up1=**true**;

y1-=5;

}**else**{

move();

}

**if**(**this**.y2<=0){

up2=**false**;

y2+=5;

}**else** **if**(**this**.y2>=880){

up2=**true**;

y2-=5;

}**else**{

move();

}

}

**private** **void** move()

{

**if**(up1){

y1-=5;

}**else**{

y1+=5;

}

**if**(up2){

y2-=5;

}**else**{

y2+=5;

}

}

**public** **int** gety1(){ //获取p0点

**return** **this**.y1;

}

**public** **int** gety2(){//获取p1点

**return** **this**.y2;

}

}

**public** **void** getpoint() //这个方法则在画出只能轨迹前调用，目的是获取两个点坐标

{

**for**(**int** j=0;j<enemy.size();j++)

{

Plane p=enemy.get(j);

p.getpoint(gbcp,me);

}

}

**public** **void** move() //这是飞机的移动方法

{

**if**(**this**.style==Constants.***ME***)

{

**if**(left&&xpos>0){

xpos-=Constants.***PLANE\_SPEED***;

}

**if**(right&&xpos<598){

xpos+=Constants.***PLANE\_SPEED***;

}

**if**(up&&ypos>0){

ypos-=Constants.***PLANE\_SPEED***;

}

**if**(down&&ypos<880){

ypos+=Constants.***PLANE\_SPEED***;

}

}**else**{

**if**(**this**.style==Constants.***ENEMY\_ONE***) //当敌机的类型是enemyone时采用智能飞行轨迹，if中是计算轨迹的算法

{

**if**(xpos<598&&ypos<900)

{

xpos=(**int**)(0.25\***this**.bcx3+0.5\***this**.bcx1+0.25\***this**.bcx2);

ypos=(**int**)(0.25\***this**.bcy3+0.5\***this**.bcy1+0.25\***this**.bcy2);

}

**else**{

planes.remove(**this**);

}

}**else**{

**if**(xpos<598&&ypos<900)

{

simplemove();

}

**else**{

planes.remove(**this**);

}

}

}

}

**public** **void** drawSelf(Graphics g) { //这两个方法是不同飞机射击的子弹方法

**if**(single){

g.drawImage(image, xpos,ypos,**null**);

move();

outDetection();

}**else**{

**if**(**this**.style==Constants.***FIRE\_ONE***){

g.drawImage(image, xpos-24,ypos,**null**);

g.drawImage(image, xpos+9,ypos,**null**);

move();

outDetection();

}**else** **if**(**this**.style==Constants.***FIRE\_TWO***){

g.drawImage(image, xpos-4,ypos,**null**);

g.drawImage(image, xpos+29,ypos,**null**);

g.drawImage(image,**this**.leftx+20 ,ypos,**null**);

g.drawImage(image,**this**.rightx+20,ypos,**null**);

move();

outDetection();

}**else** **if**(**this**.style==Constants.***FIRE\_THREE***){

g.drawImage(image, xpos+1,ypos,**null**);

g.drawImage(image, xpos+34,ypos,**null**);

g.drawImage(image,**this**.leftx+25 ,ypos,**null**);

g.drawImage(image,**this**.rightx+25,ypos,**null**);

g.drawImage(image,**this**.leftxx+25 ,ypos,**null**);

g.drawImage(image,**this**.rightxx+25,ypos,**null**);

move();

outDetection();

}**else** **if**(**this**.style==Constants.***ENEMY\_BULLET\_ONE***){

g.drawImage(image, xpos-24,ypos,**null**);

g.drawImage(image, xpos+9,ypos,**null**);

move();

outDetection();

}

**else** **if**(**this**.style==Constants.***ENEMY\_BULLET\_TWO***){

g.drawImage(image, xpos-24,ypos,**null**);

g.drawImage(image, xpos+9,ypos,**null**);

g.drawImage(image, xpos-40,ypos,**null**);

g.drawImage(image, xpos+25,ypos,**null**);

move();

outDetection();

}

}

**private** **void** move0() {

ypos-=speed;

speed+=speedincrease;

}

**private** **void** move1() {

ypos-=(speed+3);

speed+=(speedincrease+1);

**this**.leftx-=4;

**this**.rightx+=4;

}

**private** **void** move2() {

ypos+=speed;

speed+=speedincrease;

}

**private** **void** specialmove()

{

xpos+=speed\*(player.xpos-**this**.xpos)/Math.*sqrt*(Math.*pow*((player.xpos-**this**.xpos), 2)+Math.*pow*((player.ypos-**this**.ypos),2));

ypos+=speed\*(player.ypos-**this**.ypos)/Math.*sqrt*(Math.*pow*((player.xpos-**this**.xpos), 2)+Math.*pow*((player.ypos-**this**.ypos),2));

}

**private** **void** singlebulletmove()

{

xpos+=(speed+5)\*(player.xpos-**this**.xpos)/Math.*sqrt*(Math.*pow*((player.xpos-**this**.xpos), 2)+Math.*pow*((player.ypos-**this**.ypos),2));

ypos+=(speed+5)\*(player.ypos-**this**.ypos)/Math.*sqrt*(Math.*pow*((player.xpos-**this**.xpos), 2)+Math.*pow*((player.ypos-**this**.ypos),2));

}

/\*\*

\* remove itself when the bullet is out of bounds.

\*/

**private** **void** outDetection() {

**if** (xpos < 0 || xpos > 630) {

bullets.remove(**this**);

}

**if** (ypos < 0 || ypos > 900) {

bullets.remove(**this**);

}

}

### 项目总结：

项目中我所写的主要方法在上一模块已经给出，还有一些次要的方法就不提了，那么对于这一个项目我最后想说一下制作过程中的问题和解决心得。

首先是我们项目在编写框架的时候有些草率，因为许多参数没有想到，导致后续制作有增添不少方法。

其次是游戏运行的时候大量实例化后不用的类如果不清除会占用内存，减慢游戏速度，所以就这一点，结合java虚拟机的垃圾回收机制，我们采用传容器的方法解决，所有的类放入容器，不用的时候便remove掉，这样确保每次运行的时候不会存在额外不用的类占据空间。

最后是由于是两个人的作品，所以这也考验了我们所写代码的规范和整合的操作，提高了团队合作能力。