tankgamee.java

**package** tank;

**import** java.awt.\*;

**import** javax.swing.\*;

**import** java.util.\*;

**import** java.awt.event. \*;

**public** **class** tankgamee **extends** JFrame **implements** ActionListener,KeyListener{

**public** **static** **void** main(String[] args){

JFrame p = **new** JFrame("坦克大战");

StartPanel a = **new** StartPanel();

MyPanel b = **new** MyPanel();

b.addKeyListener(b);

JMenuBar menu = **new** JMenuBar();//添加菜单

JMenu m1 = **new** JMenu("游戏菜单");

JMenuItem m2 = **new** JMenuItem("开始新游戏");//创建菜单的项

JMenuItem m3 = **new** JMenuItem("接着上局游戏");

JMenuItem m4 = **new** JMenuItem("存盘退出");

JMenuItem m5 = **new** JMenuItem("退出游戏");

menu.add(m1);

m1.add(m2);

m1.add(m3);

m1.add(m4);

m1.add(m5);

p.setJMenuBar(menu);//将菜单添加导p上

p.add(a,BorderLayout.***CENTER***);

p.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***); //设置点击叉号即可关闭程序

ActionListener listen = **new** ActionListener(){

**public** **void** actionPerformed(ActionEvent e){

a.setVisible(**false**);

p.remove(a);//将a去除

p.add(b,BorderLayout.***CENTER***);

b.setVisible(**true**);//将b加入

b.requestFocus();

Thread ab =**new** Thread(b);//使游戏界面一直刷新，弱国不加线程，那么只有在你进行操作的时候才会刷新界面

ab.start();

}

};

m2.addActionListener(listen);//给m2添加监视器，点击m2就触发时间

p.setBounds(400,200,1000,600);

p.setVisible(**true**);

}

}

**class** StartPanel **extends** JPanel{//开始面板类框架

**public** **void** paint(Graphics g){

**super**.paint(g);

**this**.setBackground(Color.***BLACK***);

g.setColor(Color.***black***);

g.setColor(Color.***red***);

Font font = **new** Font("宋体", Font.***BOLD***, 50); // 创建字体对象

g.setFont(font);

g.drawString("Welocme To Play Tank Game",100,250);

}

}

**class** MyPanel **extends** JPanel **implements** KeyListener,Runnable{//游戏面板的创建

Hero hero=**null**;

Vector<Enemy>ets=**new** Vector<Enemy>();

**int** all=10;//子弹数量

**int** word=0;//记录第几发炮弹

**int** etsize=3;//敌人数量

**public** **void** paint(Graphics g){

**super**.paint(g);//背景的建立

**this**.setBackground(Color.***BLACK***);

**this**.setVisible(**true**);

**this**.drawTank(hero.x,hero.y,hero.direct,0,g,**true**);

**for**(**int** i=0;i<etsize;i++){

Enemy enemy = ets.get(i);//依次取出敌人的对象，然后画出来

**this**.drawTank(enemy.x,enemy.y,enemy.direct,1,g,enemy.isalive);

}

}

**public** **void** drawTank(**int** tx,**int** ty,**int** direction,**int** flag,Graphics g, **boolean** isalive){//增加了一个参数，来确定画不画坦克

**if**(isalive == **true**){

**if**(flag==0)

g.setColor(Color.***blue***);

**else**

g.setColor(Color.***red***);

**if**(direction==1){ //炮管向上

g.drawRect(tx-6,ty-10,5,30);

g.drawRect(tx+11,ty-10,5,30);

g.drawRect(tx-1,ty-6,12,20);

g.fillRect(tx+5,ty-15,2,20);

}

**else** **if**(direction==2){//炮管向右（这个坦克应该是前后除了炮管一样，左右除了炮管一样）

g.drawRect(tx-10,ty-6,30,5);

g.drawRect(tx-10,ty+11,30,5);

g.drawRect(tx-6,ty-1,20,12);

g.fillRect(tx+5,ty+4,20,2);

}

**else** **if**(direction==3){//炮管向下

g.drawRect(tx-6,ty-10,5,30);

g.drawRect(tx+11,ty-10,5,30);

g.drawRect(tx-1,ty-6,12,20);

g.fillRect(tx+5,ty+7,2,20);

}

**else** **if**(direction==4){//炮管向左

g.drawRect(tx-10,ty-6,30,5);

g.drawRect(tx-10,ty+11,30,5);

g.drawRect(tx-6,ty-1,20,12);

g.fillRect(tx-15,ty+4,20,2);

}

g.drawOval(tx, ty,10,8);//画中间的那个圆形

g.fillOval(tx, ty,10,8);//填充圆形

}

**for** (**int** i=0;i<hero.zidan1.size();i++) {//每次将所有可见的子弹画出来

Shot myShot = hero.zidan1.get(i);

**if** (myShot.isalive == **true**) {

g.setColor(Color.***blue***);

g.drawOval(myShot.x,myShot.y,10,8);

g.fillOval(myShot.x,myShot.y,10,8);

}

}

}

**public** **boolean** hitTank(Shot shot,Tank tank){//判断是否打到了坦克

**if**(((shot.x)<=(tank.x+15))&&((shot.x)>=(tank.x-15))&&((shot.y)<=(tank.y+20))&&((shot.y)>=(tank.y-20))){

**return** **true**;

}

**else**

**return** **false**;

}

**public** **void** run(){//使画面一直刷新

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

**for**(**int** i=0;i<hero.word;i++){//用for循环来看每发子弹是否打到敌方坦克

Shot biabiabia = hero.zidan1.get(i);

**if**(biabiabia.isalive == **true**){

**for**(**int** j=0;j<etsize;j++){

Enemy enemy = ets.get(j);

**if**(enemy.isalive == **true**){

**if**(hitTank(biabiabia,enemy) == **true**){

biabiabia.isalive = **false**;

enemy.isalive = **false**;

}

}

}

}

}

repaint();

}

}

**public** MyPanel(){//构造函数

hero = **new** Hero(300,300);//构造HERO

**int** x=50,y=100;

**for**(**int** i=0;i<etsize;i++){

Enemy enemy = **new** Enemy(x,y);//构造ENEMY

x+=250;

ets.add(enemy);//放到VECTOR内部

Thread ac = **new** Thread(enemy);//在此处进行线程的创建

ac.start();

}

}

@Override

**public** **void** keyPressed(KeyEvent e) {

**int** KeyCode = e.getKeyCode();//返回按键的整数值

**if**(KeyCode == 87||KeyCode == 38){//对于‘w’‘a’‘s’‘d’以及小键盘的辨别

hero.up();

}

**else** **if**(KeyCode == 65||KeyCode == 37){

hero.left();

}

**else** **if**(KeyCode == 68||KeyCode == 39){

hero.right();

}

**else** **if**(KeyCode == 83||KeyCode == 40){

hero.down();

}

**else** **if**(KeyCode == KeyEvent.***VK\_SPACE***){

hero.shotothers();

}

}

@Override

**public** **void** keyReleased(KeyEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** keyTyped(KeyEvent arg0) {

// **TODO** Auto-generated method stub

}

}

tank.java

**package** tank;

**import** java.util.Vector;

**public** **class** Tank {

**boolean** isalive=**true**;

Vector<Shot>zidan1=**new** Vector<Shot>();//子弹的Vector类

**int** zidan=10;//子弹总数

**int** x=0;

**int** y=0;

**int** direct;//方向

**int** speed=20;//速度

**public** Tank(**int** x,**int** y,**int** direct){

**this**.x=x;

**this**.y=y;

**this**.direct=direct;

}

**void** up(){

**this**.y-=10;

**this**.direct=1;

}

**void** down(){

**this**.y+=10;

**this**.direct=3;

}

**void** left(){

**this**.x-=10;

**this**.direct=4;

}

**void** right(){

**this**.x+=10;

**this**.direct=2;

}

}

**class** Hero **extends** Tank{

**int** word=0;//第几发子弹

**public** Hero(**int** x,**int** y){

**super**(x,y,0);

direct=1;

}

**public** **void** shotothers(){

**if**(word<zidan){

Shot a = **new** Shot(x, y, direct);

zidan1.add(a);

Thread ac = **new** Thread(a);//给当前子弹一个线程

ac.start();

word++;//控制子弹数目

}

}

}

**class** Enemy **extends** Tank **implements** Runnable{

**public** Enemy(**int** x,**int** y){

**super**(x,y,0);

direct=3;

}

**public** **void** run(){//控制敌方坦克的运动

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

**while**(**this**.isalive == **true**){

**if**(**this**.direct == 1){

**for**(**int** i=0;i<15;i++){//i越大，坦克改变方向的次数就会越少

**this**.y-=1;

**try** {

Thread.*sleep*(50);//休眠50后继续做循环

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**else** **if**(**this**.direct == 2){

**for**(**int** i=0;i<15;i++){

**this**.x+=1;

**try** {//这个是固定用法try catch

Thread.*sleep*(50);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**else** **if**(**this**.direct == 3){

**for**(**int** i=0;i<15;i++){

**this**.y+=1;

**try** {//这个是固定用法try catch

Thread.*sleep*(50);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**else** **if**(**this**.direct == 4){

**for**(**int** i=0;i<15;i++){

**this**.x-=1;

**try** {//这个是固定用法try catch

Thread.*sleep*(50);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

**if**(**this**.x<=20){//判断是否碰到边界，如果碰到边界就掉头

**this**.direct = 2;

}

**else** **if**(**this**.x>=950){

**this**.direct = 4;

}

**else** **if**(**this**.y<=20){

**this**.direct = 3;

}

**else** **if**(**this**.y>=550){

**this**.direct = 1;

}

**else**{//没碰到边界就随即放想想

**this**.direct =(**int**)(Math.*random*()\*4)+1;

}

**if**(**this**.isalive == **false**)//如果该坦克已经被消灭，那么就将它的线程关闭

Thread.*interrupted*();

}

}

}

}

**class** Shot **implements** Runnable{

**int** x;

**int** y;

**int** direct;

**int** speed=1;//这个和下面的sleep联合运用会使子弹的轨迹更平滑

**boolean** isalive=**true**;

**public** Shot(**int** x,**int** y,**int** direct){

**this**.x=x;

**this**.y=y;

**this**.direct=direct;

}

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

**while**(**this**.isalive==**true**){//对子弹坐标二修改

**if**(**this**.direct==1){

**this**.y-=**this**.speed;

}

**if**(**this**.direct==2){

**this**.x+=**this**.speed;

}

**if**(**this**.direct==3){

**this**.y+=**this**.speed;

}

**if**(**this**.direct==4){

**this**.x-=**this**.speed;

}

**try** {//这个是固定用法try catch

Thread.*sleep*(4);//控制在什么时候唤醒线程，在这个游戏里面，将时间缩短的很少就会减少卡顿（子弹）

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**if**(**this**.x<0||**this**.x>1000||**this**.y<0||**this**.y>600){

**this**.isalive=**false**;

Thread.*interrupted*();//如果超出那么就结束线程

}

}

}

}

本次实验主要完成就是两个功能，分别是1实现子弹击毁坦克；2敌方坦克的自主移动

在第一点上运用的就是继续重写run方法，在里面加上对子弹和坦克的遍历，用两个for循环表示，如果两者碰到一起，那么两者的isalive都变为false，每次刷新页面都要判断

第二点则是给敌方坦克添加一个线程，在enemy类后面实现runnable接口，重写run方法，在tangamee里的构造enemy时添加线程。

调试的时候遇到的问题都是些小问题，比如因为我修改了draw函数，但是我之前使用draw函数的语句里并未修改，于是就花费了时间去寻找错误；还有就是在给enemy添加线程的时候，在draw函数内部添加，以至于每次draw都会创建一个新线程，电脑直接卡住。