**《程序设计》课程设计**

题目名称： 坦克战争

学 院： 计算机学院

专业班级： 软件工程1607班

学生姓名： 李林育

学 号： 14164801087

指导教师：

完成日期:2017年12月27日

目录：

1. 前言
2. 坦克战争游戏的初步分析。
   1. 流程图
   2. 类图
   3. 设计思路
3. 核心设计分析
   1. 核心算法
   2. 核心思想
4. 编码实现
   1. 详细说明核心部分代码
   2. 展示整个游戏完整代码
5. 游戏测试与问题的发现
6. 总结心得

前言

课程设计实验环境：

Windows10（家庭中文版64位）

+manjaro Linux

均可以编译，但在manjaro Linux下面没有

GUI-icon。

使用开发工具IDE : Intellij idea + Eclipse

均可以使用。

初次接触：

这是本人第一次写有关Java图形界面且比较大的一个程序，当时看到有课程设计的题目第一感觉就想做一个小游戏出来，毕竟我从大一到现在还没有写过完整的图形界面程序，然后就选择了坦克战争，本来这个游戏将会更加庞大和漂亮，但是由于时间仓促，以及临近期末考试，这个坦克战争只能说是一个还没有完全成熟的一个版本吧，但是应该已经符合课程设计的要求。

李林育

2017年12月28日

流程图：

开始

画出游戏界面

初始化我方坦克，敌方坦克，画出墙面，画出游戏操作方法

游戏结束！

判断胜负

产生爆炸效果

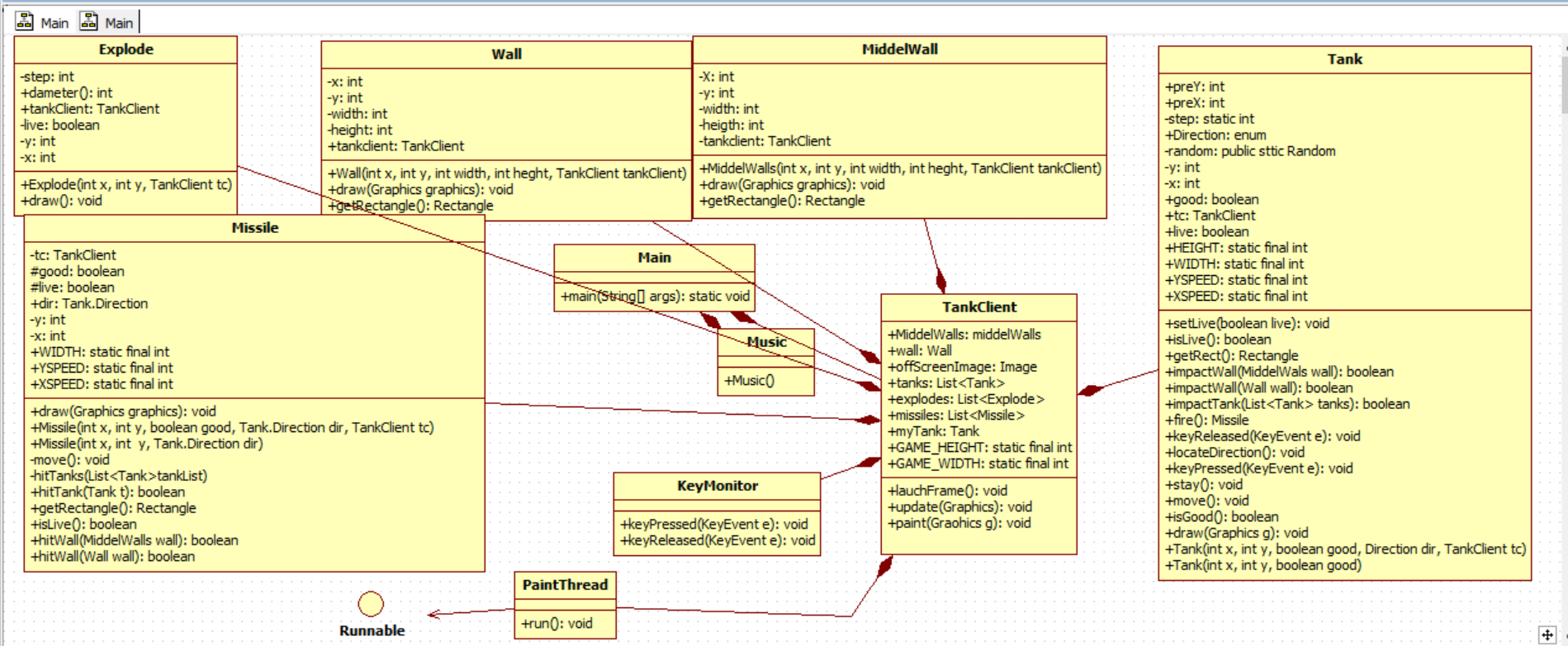
击中坦克

没有击中坦克

判断碰撞

开火（Fire）

UML类图：



设计思路：

1. 窗口界面的设计：用Java的GUI图形界面将游戏的整体窗口画出来，然后在再将一系列与游戏相关的东西画进去：坦克，墙壁，子弹等等..但是由于这些东西都是静态的，所以我们要用到重画rePaint()利用多线程的知识均匀的重复画出这些东西才能展示出动态的效果出来。
2. 坦克类：坦克类需要考虑坦克的坐标，坦克的方向，坦克的运动（往八个方向移动），坦克的炮筒方向，坦克的显示（八张图片每一张图片代表坦克现在的状态），坦克需要开火发射导弹，坦克要分辨敌我关系，分辨好坏关系，以及坦克往八个方向运动的速度大小。以及用来装敌对坦克的坦克容器(TankList) 。 同时坦克要设定对键盘监听和具备响应键盘的事件（KeyPress和KeyRelesed），每当你按下一次方向键之后坦克就会移动，还要设定一个抬起键盘的事件，每次抬起一次X键就让坦克开炮。坦克还会爆炸，当敌对坦克或者自己坦克受到了相互敌对的坦克的子弹碰撞的时候就会产生爆炸效果同时将坦克移除画面，敌对坦克还不能穿墙，以及坦克之间不能够重叠，这里我们设置了一个prestep用来记录前一步的属性，这样就可以防止他们相互重叠和穿越墙壁，具体功能会在后面讲到。
3. 子弹类：同样的子弹类要具备速度，并且子弹的速度要比坦克更快，因为….（总不能让坦克追着子弹走吧….），子弹还需要坐标来控制子弹的位置，子弹也会消亡，需要一个子弹容器来装子弹(MissileList)，子弹要设置碰撞函数（测定子弹和坦克的碰撞以及子弹和墙壁的碰撞），子弹不能够通过墙壁，遇到墙壁之后要自动销毁（从子弹容器里面移除来）,子弹需要根据坦克炮筒的方向来确定开炮的方向，子弹需要设置一个判断好坏的函数来判读两个坦克之间是不是敌对关系，（这个可以用来防止“坦克自杀”，也可以用来防止坦克之间“伤害友军”）
4. 墙类：有两种墙，一种是普通的墙，还有一种是放在中间的墙，原本这两个种墙壁一种的，但是由于这样嵌入的图片太多游戏玩起来会感觉到比较卡，所以我就将几张图片PS到了一起去了，
5. 爆炸类：爆炸类用一个数组来存放爆炸所需要表示时的画面，这里我没有嵌入图片到窗口中，而是直接用一系列颜色像同而大小不同的圆圈来代替并且爆炸的效果展示还是挺不错的，
6. 音乐类：开始是可以完好的播放的，但是后来由于嵌入的图片逐渐增多之后对CPU和磁盘的占用率越来越大之后就将音乐这个功能给取消了，但是音乐类并没有删除而是保留着。

核心设计分析：

1.将图片插入到窗口中：

**public static** BufferedImage *tank\_up*;

**static** {  
 **try** {  
 *tank\_up*= ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-up.jpg"**));

} **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
}此处举出一张图片为例，其他的就不多演示了

2.建立监听，让窗口可以被关掉，！（运用匿名类：当类比较简单，不涉及一些重要的逻辑以及不会再继续扩展的时候使用匿名类是十分方便的）：

**this**.addWindowListener(**new** WindowAdapter() {  
 **public void** windowClosing(WindowEvent e) {  
 System.*exit*(0);  
 }  
});

3.画图像时要注意窗口坐标的分布：

Y

X

4.利用线程让坦克和子弹“动起来”：

**private class** PaintThread **implements** Runnable {  
 **public void** run() {  
 **while**(**true**) {  
 *//重画自动可以自动调用paint函数* repaint();  
 **try** {  
 Thread.*sleep*(50);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

5.在重画之后就出现了一个问题，那就是双闪烁问题（当你在运行这个GUI界面的时候你会发现见面有频繁的闪烁显现）得出问题原因：

刷新的频率太快，paint方法还没有完成，下一个方法又来了...

解决方法：

首先我们要了解paint（）是由update()调用出来的，而update（）是由repaint（）调用出来的

在awt中对于窗体画布的重绘其条用顺序是repaint() —>update()—>paint()

所以只要我们修改update代码就可以了，。在 updateupdate updateupdate方法 里，先 new new出来 一张虚拟的图片， 得 到这个图片的画笔， Graphics gOffScreen = offScreenImage.getGraphics();，保存 画笔的颜色， 设置 这支画笔的颜色与游戏背景相同， 使用这支画笔 画一个 与游戏窗口大小相 同的虚拟图片， 同的虚拟图片， 同的虚拟图片， 同的虚拟图片， 同的虚拟图片， 同的虚拟图片， 同的虚拟图片， 再 还原来的颜色 还原来的颜色 还原来的颜色 还原来的颜色 还原来的颜色 还原来的颜色 还原来的颜色 。 将 这 支画笔传给 支画笔传给 支画笔传给 支画笔传给 支画笔传给 paint paintpaint方法 print(gOffScreen);，这样 paint paint方法 画出来的就在这张虚拟图片上 。最后 使用 游戏窗口的这支画笔 g将这张虚拟图片 画到游戏窗口中，代码如下：

public void update(Graphics g) {  
if(offScreenImage == null) {  
offScreenImage = this.createImage(800, 600);  
}  
//拿到这个图片的画笔  
Graphics gOffScreen = offScreenImage.getGraphics();  
Color c = gOffScreen.getColor();  
gOffScreen.setColor(Color.GREEN);  
gOffScreen.fillRect(0, 0, 800, 600);  
gOffScreen.setColor(c);  
print(gOffScreen);  
g.drawImage(offScreenImage, 0, 0, null);

}

但是这样就不能自己添加背景图片了，只能用纯色...但是解决了频闪的问题。

1. 定义一个确定行走的方向的函数：

*//用来确定当前方向***void** locateDirection() {  
 **if**(**bL** && !**bU** && !**bR** && !**bD**) **dir** = Direction.***L***;  
 **else if**(**bL** && **bU** && !**bR** && !**bD**) **dir** = Direction.***LU***;  
 **else if**(!**bL** && **bU** && !**bR** && !**bD**) **dir** = Direction.***U***;  
 **else if**(!**bL** && **bU** && **bR** && !**bD**) **dir** = Direction.***RU***;  
 **else if**(!**bL** && !**bU** && **bR** && !**bD**) **dir** = Direction.***R***;  
 **else if**(!**bL** && !**bU** && **bR** && **bD**) **dir** = Direction.***RD***;  
 **else if**(!**bL** && !**bU** && !**bR** && **bD**) **dir** = Direction.***D***;  
 **else if**(**bL** && !**bU** && !**bR** && **bD**) **dir** = Direction.***LD***;  
 **else if**(!**bL** && !**bU** && !**bR** && !**bD**) **dir** = Direction.***STOP***;  
}

1. 碰撞检测函数：

*//为图形碰撞做准备，给出坦克的方位，用于判断是否与子弹碰撞***public** Rectangle getRect()  
{  
 **return new** Rectangle(**x**, **y**, ***WIDTH***, ***HEIGHT***);  
}

**public boolean** hitTank(Tank t) {  
 **if**(**this**.**live**&&**this**.getRect().intersects(t.getRect()) && t.isLive()&&**this**.**good**!=t.isGood()) {  
 t.setLive(**false**);  
 **this**.**live** = **false**;  
 Explode explode = **new** Explode(**x**, **y**, **tc**);  
 **tc**.**explodes**.add(explode);  
 **return true**;  
 }  
 **return false**;  
}

1. 画出敌军坦克，：
2. **for** (**int** i=0;i<3;i++) {  
    **tanks**.add(**new** Tank(200 , 150+ 50 \* i, **false**, Tank.Direction.***D***, **this**));  
   }  
   **for** (**int** i=0;i<3;i++) {  
    **tanks**.add(**new** Tank(920, 150 + 50 \* i, **false**, Tank.Direction.***D***, **this**));  
   }  
   **for**(**int** i=0;i<3;i++) {  
    **tanks**.add((**new** Tank(351 + 50 \* i, 668, **false**, Tank.Direction.***D***, **this**)));  
   }  
   **tanks**.add((**new** Tank(449, 60, **false**, Tank.Direction.***D***, **this**)));

10：让敌军更加智能，而不是智能傻傻的站在那里：

**if** (!**good**) {  
 *//将枚举类型用.values()方法之间转化为数组类型* Direction[]directions=Direction.*values*();//!!!!!  
 **if** (*step* == 0) {  
 *step*=*random*.nextInt(13)+2;  
 **int** randomNumber = *random*.nextInt(directions.**length**);  
 **dir** = directions[randomNumber];  
 }  
 *step*--;  
 **if** (*random*.nextInt(40)>38)  
 **this**.fire();  
}

利用随机数来控制敌军的移动，但是必须要设定一个敌军移动的范围，**if** (*random*.nextInt(40)>38)

不然敌军的向各个方向移动的概率是相等的，这将会导致敌军像“一群热锅上的蚂蚁”那样一直围着一个点动来动去..而且移动的频率特别快。

源代码展示：

**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
**import** java.awt.image.BufferedImage;  
**import** java.util.List;  
**import** java.util.Random;  
  
**public class** Tank {  
 *//用于存放要画出坦克的图像* **public static** BufferedImage *tank\_up*;  
 **public static** BufferedImage *tank\_down*;  
 **public static** BufferedImage *tank\_left*;  
 **public static** BufferedImage *tank\_right*;  
 **public static** BufferedImage *tank\_ul*;  
 **public static** BufferedImage *tank\_ur*;  
 **public static** BufferedImage *tank\_dl*;  
 **public static** BufferedImage *tank\_dr*;  
 *//读取坦克八个方向的图像* **static** {  
 **try** {  
 *tank\_up*= ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-up.jpg"**));  
 *tank\_down*=ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-down.jpg"**));  
 *tank\_left*=ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-left.jpg"**));  
 *tank\_right*=ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-right.jpg"**));  
 *tank\_ul*=ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-ul.jpg"**));  
 *tank\_ur*=ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-ur.jpg"**));  
 *tank\_dl*=ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-dl.jpg"**));  
 *tank\_dr*=ImageIO.*read*(  
 Tank.**class**.getResource(**"tank-dr.jpg"**));  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
 *//设置坦克的速度* **public static final int *XSPEED*** = 5;  
 **public static final int *YSPEED*** = 5;  
 *//设置坦克的大小* **public static final int *WIDTH*** = 30;  
 **public static final int *HEIGHT*** = 30;  
  
 *//确定坦克是否还活着* **private boolean live** = **true**;  
 *//定义一个对象引用方便来传递参数* TankClient **tc**;  
 *//区分是自己的坦克还是别人的坦克* **private boolean good**;  
 *//设置自己坦克的坐标* **private int x**, **y**;  
 *//设置一个随机数，用来让坦克随机移动* **private static** Random *random*=**new** Random();  
 *//定义坦克沿着四个斜线方向移动* **private boolean bL**=**false**, **bU**=**false**, **bR**=**false**, **bD** = **false**;  
 *//用枚举类型来定义坦克行走的八个方位* **enum** Direction {***L***, ***LU***, ***U***, ***RU***, ***R***, ***RD***, ***D***, ***LD***, ***STOP***};  
 *//设置默认方向为停止* **private** Direction **dir** = Direction.***STOP***;  
 *//设置炮筒方向，默认向下* **private** Direction **ptDir** = Direction.***D***;  
 **private static int** *step*=*random*.nextInt(13)+2;  
 *//设置一个先前位置点，用来记录上一个位置坦克的位置* **int preX**,**preY**;  
  
 **public** Tank(**int** x, **int** y, **boolean** good) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**good** = good;  
 }  
 *//坦克的构造方法* **public** Tank(**int** x, **int** y, **boolean** good, Direction dir,TankClient tc) {  
 **this**(x, y, good);  
 **this**.**dir** = dir;  
 **this**.**tc** = tc;  
 }  
 *//画出坦克* **public void** draw(Graphics g) {  
 **if**(!**live**) {  
 *//如果是敌方坦克被销毁则将它从list中移除，以免占用内存* **if** (!**good**) {  
 **tc**.**tanks**.remove(**this**);  
 }**else** {  
 *//我们操控的坦克已经被炸毁....游戏结束//Tahoma 宋体* g.setFont(**new** Font(**"Tahoma"**, Font.***BOLD***, 40));  
 g.drawString(**"Game Over !"**,422,370);  
 }  
 }  
 **if**(**live**){  
 *//如果活着，根据键盘设置的方向画出不同坦克的图片* **switch**(**ptDir**) {  
 **case *U***:  
 g.drawImage(*tank\_up*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 **case *D***:  
 g.drawImage(*tank\_down*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 **case *L***:  
 g.drawImage(*tank\_left*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 **case *R***:  
 g.drawImage(*tank\_right*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 **case *LU***:  
 g.drawImage(*tank\_ul*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 **case *RU***:  
 g.drawImage(*tank\_ur*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 **case *LD***:  
 g.drawImage(*tank\_dl*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 **case *RD***:  
 g.drawImage(*tank\_dr*, **x**, **y**, 50, 30, **null**);  
 **break**;  
 }  
 }  
 move();  
 }  
  
 **public boolean** isGood() {  
 **return good**;  
 }  
  
 **void** move() {  
 **this**.**preX** = **x**;  
 **this**.**preY** = **y**;  
  
 **switch**(**dir**) {  
 **case *L***:  
 **x** -= ***XSPEED***;  
 **break**;  
 **case *LU***:  
 **x** -= ***XSPEED***;  
 **y** -= ***YSPEED***;  
 **break**;  
 **case *U***:  
 **y** -= ***YSPEED***;  
 **break**;  
 **case *RU***:  
 **x** += ***XSPEED***;  
 **y** -= ***YSPEED***;  
 **break**;  
 **case *R***:  
 **x** += ***XSPEED***;  
 **break**;  
 **case *RD***:  
 **x** += ***XSPEED***;  
 **y** += ***YSPEED***;  
 **break**;  
 **case *D***:  
 **y** += ***YSPEED***;  
 **break**;  
 **case *LD***:  
 **x** -= ***XSPEED***;  
 **y** += ***YSPEED***;  
 **break**;  
 **case *STOP***:  
 **break**;  
 }  
   
 **if**(**this**.**dir** != Direction.***STOP***) {  
 **this**.**ptDir** = **this**.**dir**;  
 }  
   
 **if**(**x** < 0) **x** = 0;  
 **if**(**y** < 30) **y** = 30;  
 **if**(**x** + Tank.***WIDTH*** > TankClient.***GAME\_WIDTH***) **x** = TankClient.***GAME\_WIDTH*** - Tank.***WIDTH***;  
 **if**(**y** + Tank.***HEIGHT*** > TankClient.***GAME\_HEIGHT***) **y** = TankClient.***GAME\_HEIGHT*** - Tank.***HEIGHT***;  
 *//让敌对坦克动起来，并且稍微智能一点，并且火力不能太猛* **if** (!**good**) {  
 *//将枚举类型用.values()方法之间转化为数组类型* Direction[]directions=Direction.*values*();  
 **if** (*step* == 0) {  
 *step*=*random*.nextInt(13)+2;  
 **int** randomNumber = *random*.nextInt(directions.**length**);  
 **dir** = directions[randomNumber];  
 }  
 *step*--;  
 **if** (*random*.nextInt(40)>38)  
 **this**.fire();  
 }  
 }  
 *//让坦克回到上一次移动点位置* **public void** stay() {  
 **x**=**preX**;  
 **y** = **preY**;  
 }  
 *//响应键盘事件，当键盘按下去的时候响应键盘事件* **public void** keyPressed(KeyEvent e) {  
 **int** key = e.getKeyCode();  
 **switch**(key) {  
 **case** KeyEvent.***VK\_LEFT*** :  
 **bL** = **true**;  
 **break**;  
 **case** KeyEvent.***VK\_UP*** :  
 **bU** = **true**;  
 **break**;  
 **case** KeyEvent.***VK\_RIGHT*** :  
 **bR** = **true**;  
 **break**;  
 **case** KeyEvent.***VK\_DOWN*** :  
 **bD** = **true**;  
 **break**;  
 }  
 locateDirection();  
 }  
 *//用来确定当前方向* **void** locateDirection() {  
 **if**(**bL** && !**bU** && !**bR** && !**bD**) **dir** = Direction.***L***;  
 **else if**(**bL** && **bU** && !**bR** && !**bD**) **dir** = Direction.***LU***;  
 **else if**(!**bL** && **bU** && !**bR** && !**bD**) **dir** = Direction.***U***;  
 **else if**(!**bL** && **bU** && **bR** && !**bD**) **dir** = Direction.***RU***;  
 **else if**(!**bL** && !**bU** && **bR** && !**bD**) **dir** = Direction.***R***;  
 **else if**(!**bL** && !**bU** && **bR** && **bD**) **dir** = Direction.***RD***;  
 **else if**(!**bL** && !**bU** && !**bR** && **bD**) **dir** = Direction.***D***;  
 **else if**(**bL** && !**bU** && !**bR** && **bD**) **dir** = Direction.***LD***;  
 **else if**(!**bL** && !**bU** && !**bR** && !**bD**) **dir** = Direction.***STOP***;  
 }  
 *//键盘响应事件,当键盘抬起来的时候触发事件* **public void** keyReleased(KeyEvent e) {  
 **int** key = e.getKeyCode();  
 **switch**(key) {  
 **case** KeyEvent.***VK\_X***:  
 fire();  
 **break**;  
 **case** KeyEvent.***VK\_LEFT*** :  
 **bL** = **false**;  
 **break**;  
 **case** KeyEvent.***VK\_UP*** :  
 **bU** = **false**;  
 **break**;  
 **case** KeyEvent.***VK\_RIGHT*** :  
 **bR** = **false**;  
 **break**;  
 **case** KeyEvent.***VK\_DOWN*** :  
 **bD** = **false**;  
 **break**;  
 }  
 locateDirection();   
 }  
 *//控制坦克开火* **public** Missile fire() {  
 **if** (!**live**) {  
 **return null**;  
 }  
 *//控制子弹出现的方位地址* **int** x = **this**.**x** + Tank.***WIDTH***/2 - Missile.***WIDTH***/2;  
 **int** y = **this**.**y** + Tank.***HEIGHT***/2 - Missile.***HEIGHT***/2;  
 Missile m = **new** Missile(x, y,**good**, **ptDir**, **this**.**tc**);  
 **tc**.**missiles**.add(m);  
 **return** m;  
 }  
 *//防止坦克之间相互重叠* **public boolean** impactTank(List<Tank> tanks) {  
 **for** (**int** i=0;i<tanks.size();i++) {  
 Tank tank = tanks.get(i);  
 **if** (tank != **this**) {  
 **if** (**this**.**live** && **this**.getRect().intersects(tank.getRect())) {  
 stay();  
 **return true**;  
 }  
 }  
 }  
 **return false**;  
 }  
 *//不能让敌对坦克穿越墙面* **public boolean** impactWall(Wall wall) {  
 **if** (**this**.**live** && **this**.getRect().intersects(wall.getRectangle())) {  
 **this**.stay();  
 **return true**;  
 }  
 **return false**;  
 }  
 **public boolean** impactWall(MiddelWalls wall) {  
 **if** (**this**.**live** && **this**.getRect().intersects(wall.getRectangle())) {  
 **this**.stay();  
 **return true**;  
 }  
 **return false**;  
 }  
 *//为图形碰撞做准备，给出坦克的方位，用于判断是否与子弹碰撞* **public** Rectangle getRect()  
 {  
 **return new** Rectangle(**x**, **y**, ***WIDTH***, ***HEIGHT***);  
 }  
 *//确定坦克是否还活着* **public boolean** isLive() {  
 **return live**;  
 }  
 *//设置坦克是否还活着* **public void** setLive(**boolean** live) {  
 **this**.**live** = live;  
 }  
}

**import** javax.sound.sampled.\*;  
**import** java.applet.Applet;  
**import** java.applet.AudioClip;  
**import** java.io.File;  
**import** java.io.IOException;  
**import** java.net.MalformedURLException;  
**import** java.net.URL;  
*//由于CPU和磁盘占用率过大....就不在界面里面添加音乐了***public class** Music {  
 **public** Music()  
 {  
 *// URL url = getClass().getResource("start.wav");  
 // AudioClip clip = java.applet.Applet.newAudioClip(url);  
 // clip.play();* }  
 **static void** playMusic(){*//背景音乐播放* **try** {  
 URL cb;  
 File f = **new** File(**"start.wav"**); *// 引号里面的是音乐文件所在的路径* cb = f.toURL();  
 AudioClip aau;  
 aau = Applet.*newAudioClip*(cb);  
  
 aau.play();  
 aau.loop();*//循环播放* System.***out***.println(**"可以播放"**);  
 *// 循环播放 aau.play()  
 //单曲 aau.stop()停止播放* } **catch** (MalformedURLException e) {  
  
 e.printStackTrace();  
 }  
 }  
   
}

**import** javax.imageio.ImageIO;  
**import** java.awt.\*;  
**import** java.awt.image.BufferedImage;  
  
**public class** MiddelWalls {  
 **private int x**,**y**,**width**,**height**;  
 TankClient **tankClient**;  
  
  
 **public** MiddelWalls(**int** x, **int** y, **int** width, **int** height, TankClient tankClient) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**width** = width;  
 **this**.**height** = height;  
 **this**.**tankClient** = tankClient;  
 }  
  
 **public void** draw(Graphics graphics) {  
 *// graphics.fillRect(x, y, width, height);* **try** {  
 BufferedImage walls= ImageIO.*read*(  
 Tank.**class**.getResource(**"middleWalls.jpg"**));  
 graphics.drawImage(walls, **x**, **y**, **width**, **height**,**null**);  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
 **public** Rectangle getRectangle() {  
 **return new** Rectangle(**x**, **y**, **width**, **height**);  
 }  
}

**import** javax.imageio.ImageIO;  
**import** javax.swing.\*;  
**import** java.awt.\*;  
**import** java.awt.event.\*;  
**import** java.awt.image.BufferedImage;  
**import** java.io.IOException;  
**import** java.util.List;  
**import** java.util.ArrayList;  
  
**public class** TankClient **extends** Frame {  
 **public static final int *GAME\_WIDTH*** = 1100;  
 **public static final int *GAME\_HEIGHT*** = 850;  
 *//构造出一架我们可以操控的坦克* Tank **myTank** = **new** Tank(50, 50, **true**,Tank.Direction.***STOP***, **this**);  
  
 List<Missile> **missiles** = **new** ArrayList<Missile>();  
 List<Explode> **explodes** = **new** ArrayList<Explode>();  
 List<Tank> **tanks** = **new** ArrayList<Tank>();  
 Image **offScreenImage** = **null**;  
 *//左下三堵墙* Wall **wall\_1** = **new** Wall(103, 546, 40, 90,**this**);  
 Wall **wall\_2** = **new** Wall(103, 546+90, 40, 90,**this**);  
 Wall **wall\_3** = **new** Wall(103 + 50, 546 + 90 + 90, 40, 90, **this**);  
 *//右下三堵墙* Wall **wall\_4** = **new** Wall(896, 546, 40, 90,**this**);  
 Wall **wall\_5** = **new** Wall(896, 546+90, 40, 90,**this**);  
 Wall **wall\_6** = **new** Wall(896 - 50, 546 + 90 + 90, 40, 90, **this**);  
 *//中间上墙  
 // Wall wall\_7 = new Wall(410, 139, 80, 40,this);  
 // Wall wall\_8 = new Wall(410+80, 139, 80, 40,this);  
 // Wall wall\_9 = new Wall(410+80+80, 139, 80, 40,this);* MiddelWalls **middelWalls1** = **new** MiddelWalls(410, 139, 80 \* 3, 40, **this**);  
  
 *//中间中墙  
 // Wall wall\_10 = new Wall(410, 289, 80, 40,this);  
 // Wall wall\_11 = new Wall(410+80, 289, 80, 40,this);  
 // Wall wall\_12 = new Wall(410+80+80, 289, 80, 40,this);* MiddelWalls **middelWalls2** = **new** MiddelWalls(410, 139+150, 80 \* 3, 40, **this**);  
 *//中间下墙  
 // Wall wall\_13 = new Wall(410, 289+150, 80, 40,this);  
 // Wall wall\_14 = new Wall(410+80, 289+150, 80, 40,this);  
 // Wall wall\_15 = new Wall(410+80+80, 289+150, 80, 40,this);* MiddelWalls **middelWalls3** = **new** MiddelWalls(410, 139+300, 80 \* 3, 40, **this**);  
  
 **public void** paint(Graphics g) {  
 *//用于记录现在余留在内存中的导弹，爆炸效果，和敌对坦克数量* g.setFont(**new** Font(**"Tahoma"**, Font.***BOLD***, 20));  
 g.drawString(**"↑↓← →(MOVE) X (Fire) :"**, 10, 50);  
 *// g.drawString("explodes count:" + explodes.size(), 10, 70);* g.drawString(**"Enemy Tank Count:"** + **tanks**.size(), 10, 80);  
 *// explode.draw(g);* **for**(**int** i=0; i<**missiles**.size(); i++) {  
 Missile m = **missiles**.get(i);  
 m.draw(g);  
 *//子弹击中敌人的坦克* m.hitTanks(**tanks**);  
 m.hitTank(**myTank**);  
 m.hitWall(**wall\_1**);  
 m.hitWall(**wall\_2**);  
 m.hitWall(**wall\_3**);  
 m.hitWall(**wall\_4**);  
 m.hitWall(**wall\_5**);  
 m.hitWall(**wall\_6**);  
 m.hitWall(**middelWalls1**);  
 m.hitWall(**middelWalls2**);  
 m.hitWall(**middelWalls3**);  
  
 *// m.hitWall(wall\_7);  
 // m.hitWall(wall\_8);  
 // m.hitWall(wall\_9);  
 // m.hitWall(wall\_10);  
 // m.hitWall(wall\_11);  
 // m.hitWall(wall\_12);  
 // m.hitWall(wall\_13);  
 // m.hitWall(wall\_14);  
 // m.hitWall(wall\_15);  
 //if(!m.isLive()) missiles.remove(m);  
 //else m.draw(g);* }  
 **for** (**int** i=0;i<**explodes**.size();i++) {  
 Explode explode = **explodes**.get(i);  
 explode.draw(g);  
 }  
 *//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*用foreach会产生异常！！！！！！！！！！！！！！！！！！！！！！！  
// for (Explode explode : explodes) {  
// explode.draw(g);  
// }* **myTank**.draw(g);  
 *//画出多辆敌人坦克* **for** (**int** i=0;i<**tanks**.size();i++) {  
 Tank tank = **tanks**.get(i);  
 tank.impactWall(**wall\_1**);  
 tank.impactWall(**wall\_2**);  
 tank.impactWall(**wall\_3**);  
 tank.impactWall(**wall\_4**);  
 tank.impactWall(**wall\_5**);  
 tank.impactWall(**wall\_6**);  
 tank.impactWall(**middelWalls1**);  
 tank.impactWall(**middelWalls2**);  
 tank.impactWall(**middelWalls3**);  
  
 *// tank.impactWall(wall\_7);  
 // tank.impactWall(wall\_8);  
 // tank.impactWall(wall\_9);  
 // tank.impactWall(wall\_10);  
 // tank.impactWall(wall\_11);  
 // tank.impactWall(wall\_12);  
 // tank.impactWall(wall\_13);  
 // tank.impactWall(wall\_14);  
 // tank.impactWall(wall\_15);* tank.impactTank(**tanks**);  
 tank.draw(g);  
 }  
 **if** (**tanks**.size() == 0) {  
 *//Tahoma 宋体* g.setFont(**new** Font(**"Tahoma"**, Font.***BOLD***, 40));  
 g.drawString(**"You Are Winner !"**,422,373);  
 }  
 **wall\_1**.draw(g);  
 **wall\_2**.draw(g);  
 **wall\_3**.draw(g);  
 **wall\_4**.draw(g);  
 **wall\_5**.draw(g);  
 **wall\_6**.draw(g);  
 **middelWalls1**.draw(g);  
 **middelWalls2**.draw(g);  
 **middelWalls3**.draw(g);  
 *// wall\_7.draw(g);  
 // wall\_8.draw(g);  
 // wall\_9.draw(g);  
 // wall\_10.draw(g);  
 // wall\_11.draw(g);  
 // wall\_12.draw(g);  
 // wall\_13.draw(g);  
 // wall\_14.draw(g);  
 // wall\_15.draw(g);* }  
 *//不断填充背景色，同时要方式闪烁显现，用背景色填充的方式来消除GUI中闪烁显现* **public void** update(Graphics g) {  
 **if**(**offScreenImage** == **null**) {  
 **offScreenImage** = **this**.createImage(***GAME\_WIDTH***, ***GAME\_HEIGHT***);  
 }  
 Graphics gOffScreen = **offScreenImage**.getGraphics();  
 Color c = gOffScreen.getColor();  
 gOffScreen.setColor(Color.***GRAY***);  
 gOffScreen.fillRect(0, 0, ***GAME\_WIDTH***, ***GAME\_HEIGHT***);  
 gOffScreen.setColor(c);  
 paint(gOffScreen);  
 g.drawImage(**offScreenImage**, 0, 0, **null**);  
 }  
 *//画出整个游戏的窗口，并且画出游戏上个各个元素(敌对坦克，等等..)* **public void** lauchFrame() {  
 *//设置GUI的logo* **try** {  
 BufferedImage icon= ImageIO.*read*(  
 Tank.**class**.getResource(**"TankIcon.png"**));  
 setIconImage(icon);  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
  
 *//this.setLocation(400, 300);  
 //注意添加放置的位置，放在添加背景后面则会抛出异常！* **for** (**int** i=0;i<3;i++) {  
 **tanks**.add(**new** Tank(200 , 150+ 50 \* i, **false**, Tank.Direction.***D***, **this**));  
 }  
 **for** (**int** i=0;i<3;i++) {  
 **tanks**.add(**new** Tank(920, 150 + 50 \* i, **false**, Tank.Direction.***D***, **this**));  
 }  
 **for**(**int** i=0;i<3;i++) {  
 **tanks**.add((**new** Tank(351 + 50 \* i, 668, **false**, Tank.Direction.***D***, **this**)));  
 }  
 **tanks**.add((**new** Tank(449, 60, **false**, Tank.Direction.***D***, **this**)));  
 **this**.setSize(***GAME\_WIDTH***, ***GAME\_HEIGHT***);  
 **this**.setTitle(**"TankWar 姓名：李林育 班级：16级软件工程2班 学号：14164801087"**);  
 *//设置窗口关闭* **this**.addWindowListener(**new** WindowAdapter() {  
 **public void** windowClosing(WindowEvent e) {  
 System.*exit*(0);  
 }  
 });  
 **this**.setResizable(**false**);  
 *//画出背景颜色为灰色* **this**.setBackground(Color.***GRAY***);  
   
 **this**.addKeyListener(**new** KeyMonitor());  
 *//构造出10辆坦克* setVisible(**true**);  
 *//调用线程开始循坏* **new** Thread(**new** PaintThread()).start();  
 }  
  
 **private class** PaintThread **implements** Runnable {  
 **public void** run() {  
 **while**(**true**) {  
 *//重画自动可以自动调用paint函数* repaint();  
 **try** {  
 Thread.*sleep*(50);  
 } **catch** (InterruptedException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 }  
 *//设置键盘监听，一个按下监听和一个抬起的监听* **private class** KeyMonitor **extends** KeyAdapter {  
 **public void** keyReleased(KeyEvent e) {  
 **myTank**.keyReleased(e);  
 }  
 **public void** keyPressed(KeyEvent e) {  
 **myTank**.keyPressed(e);  
 }  
 }  
}

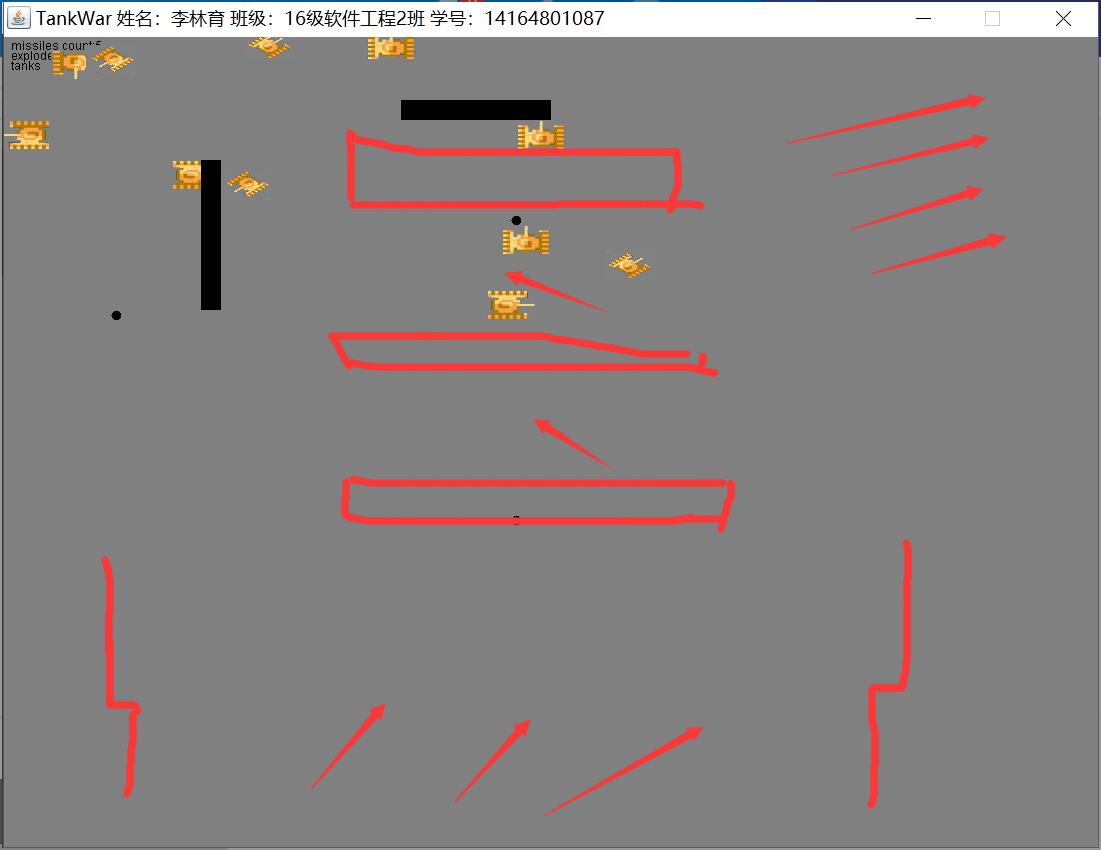
**import** com.sun.corba.se.impl.orbutil.graph.Graph;  
  
**import** javax.imageio.ImageIO;  
**import** java.awt.\*;  
**import** java.awt.image.BufferedImage;  
  
**public class** Wall {  
 **private int x**,**y**,**width**,**height**;  
 TankClient **tankClient**;  
  
  
 **public** Wall(**int** x, **int** y, **int** width, **int** height, TankClient tankClient) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**width** = width;  
 **this**.**height** = height;  
 **this**.**tankClient** = tankClient;  
 }  
  
 **public void** draw(Graphics graphics) {  
 *// graphics.fillRect(x, y, width, height);* **try** {  
 BufferedImage walls= ImageIO.*read*(  
 Tank.**class**.getResource(**"walls.gif"**));  
 graphics.drawImage(walls, **x**, **y**, **width**, **height**,**null**);  
 } **catch** (Exception ex) {  
 ex.printStackTrace();  
 }  
 }  
  
 **public** Rectangle getRectangle() {  
 **return new** Rectangle(**x**, **y**, **width**, **height**);  
 }  
}

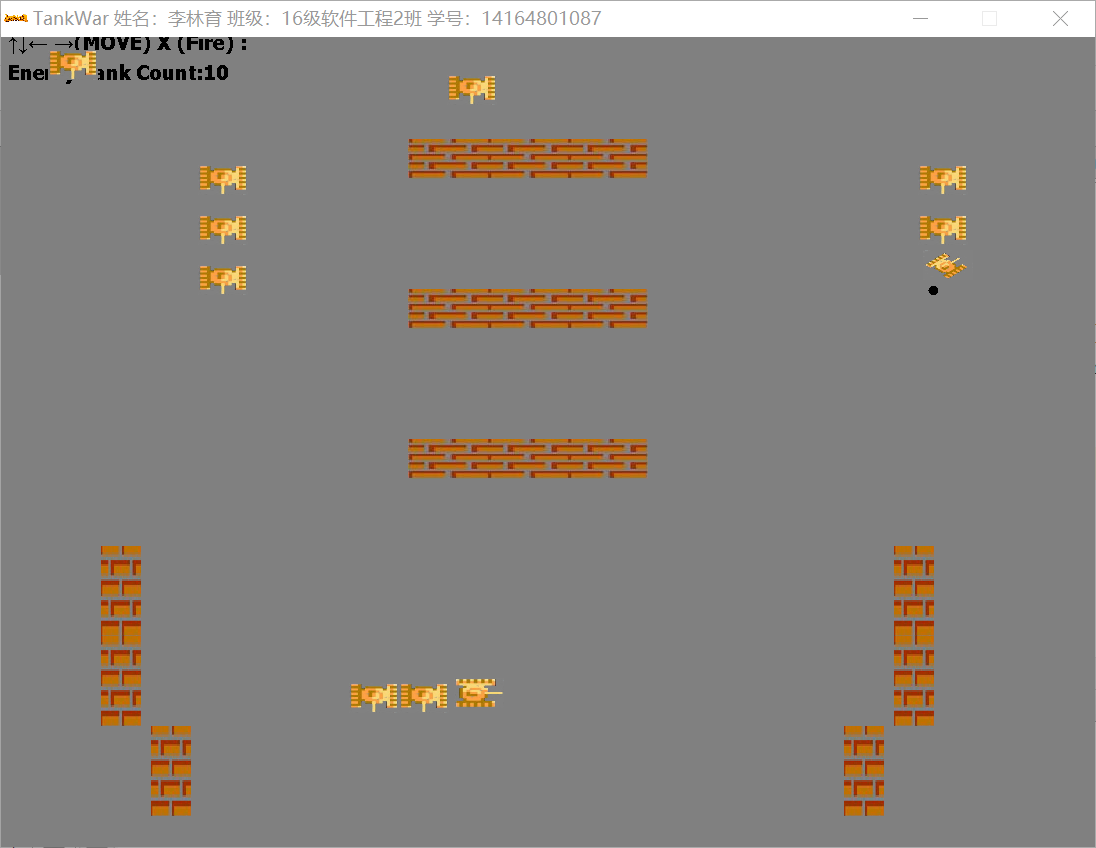
**import** java.awt.\*;  
  
**public class** Explode {  
 **private int x**, **y**;  
 **private boolean live**=**true**;  
 TankClient **tankClient**;  
 **private int step** = 0;  
 **int**[] **diameter** = {4, 7, 12, 18, 26, 32, 49, 30, 14, 6};  
  
 **public** Explode(**int** x, **int** y, TankClient tankClient) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**tankClient** = tankClient;  
 }  
  
  
 **public void** draw(Graphics graphics) {  
 **if** (!**live**) {  
 **tankClient**.**explodes**.remove(**this**);  
 **return**;  
 }  
 **if** (**step** == **diameter**.**length**) {  
 **live** = **false**;  
 **step** = 0;  
 **return** ;  
 }  
 Color c = graphics.getColor();  
 graphics.setColor(Color.***ORANGE***);  
 graphics.fillOval(**x**,**y**,**diameter**[**step**],**diameter**[**step**]);  
 graphics.setColor(c);  
 **step**++;  
 }  
  
}

**import** java.awt.\*;  
**import** java.util.List;  
**public class** Missile {  
 *//定义子弹飞行的速度* **public static final int *XSPEED*** = 10;  
 **public static final int *YSPEED*** = 10;  
 *//定义子弹的大小* **public static final int *WIDTH*** = 10;  
 **public static final int *HEIGHT*** = 10;  
 *//定义子弹的位置* **int x**, **y**;  
 *//定义子弹的方向* Tank.Direction **dir**;  
 *//定义子弹的存在状态* **private boolean live** = **true**;  
 *//用来防止子弹"自杀或者杀害友军"* **private boolean good**;  
 *//定义一个引用* **private** TankClient **tc**;  
   
 **public** Missile(**int** x, **int** y, Tank.Direction dir) {  
 **this**.**x** = x;  
 **this**.**y** = y;  
 **this**.**dir** = dir;  
 }  
   
 **public** Missile(**int** x, **int** y,**boolean** good, Tank.Direction dir, TankClient tc) {  
 **this**(x, y, dir);  
 **this**.**good**=good;  
 **this**.**tc** = tc;  
 }  
 *//话子弹* **public void** draw(Graphics g) {  
 *//如果子弹已经死亡，则将子弹移除子弹集合* **if**(!**live**) {  
 **tc**.**missiles**.remove(**this**);  
 **return**;  
 }  
   
 Color c = g.getColor();  
 g.setColor(Color.***BLACK***);  
 g.fillOval(**x**, **y**, ***WIDTH***, ***HEIGHT***);  
 g.setColor(c);  
   
 move();  
 }  
 **private void** move() {  
 **switch**(**dir**) {  
 **case *L***:  
 **x** -= ***XSPEED*** ;  
 **break**;  
 **case *LU***:  
 **x** -= ***XSPEED***;  
 **y** -= ***YSPEED***;  
 **break**;  
 **case *U***:  
 **y** -= ***YSPEED***;  
 **break**;  
 **case *RU***:  
 **x** += ***XSPEED***;  
 **y** -= ***YSPEED***;  
 **break**;  
 **case *R***:  
 **x** += ***XSPEED***;  
 **break**;  
 **case *RD***:  
 **x** += ***XSPEED***;  
 **y** += ***YSPEED***;  
 **break**;  
 **case *D***:  
 **y** += ***YSPEED***;  
 **break**;  
 **case *LD***:  
 **x** -= ***XSPEED***;  
 **y** += ***YSPEED***;  
 **break**;  
 **case *STOP***:  
 **break**;  
 }  
 *//挡子弹射出到游戏框外面的时候则子弹死亡* **if**(**x** < 0 || **y** < 0 || **x** > TankClient.***GAME\_WIDTH*** || **y** > TankClient.***GAME\_HEIGHT***) {  
 **live** = **false**;  
 }  
 }  
 *//判断子弹是否死亡* **public boolean** isLive() {  
 **return live**;  
 }  
 *//设置碰撞函数，用来判断是否击中敌对坦克的* **public** Rectangle getRect() {  
 **return new** Rectangle(**x**, **y**, ***WIDTH***, ***HEIGHT***);  
 }  
 *//如果击中设置爆炸* **public boolean** hitTank(Tank t) {  
 **if**(**this**.**live**&&**this**.getRect().intersects(t.getRect()) && t.isLive()&&**this**.**good**!=t.isGood()) {  
 t.setLive(**false**);  
 **this**.**live** = **false**;  
 Explode explode = **new** Explode(**x**, **y**, **tc**);  
 **tc**.**explodes**.add(explode);  
 **return true**;  
 }  
 **return false**;  
 }  
 *//判断子弹是否击中相互敌对的坦克* **public boolean** hitTanks(List<Tank> tankList) {  
 **for** (**int** i=0;i<tankList.size();i++) {  
 **if** (hitTank(tankList.get(i))) {  
 **return true**;  
 }  
 }  
 **return false**;  
 }  
 *//判断子弹是否击中墙壁* **public boolean** hitWall(Wall wall) {  
 **if** (**this**.**live** && **this**.getRect().intersects(wall.getRectangle())) {  
 **this**.**live** = **false**;  
 **return true**;  
 }**else** {  
 **return false**;  
 }  
 }  
 *//判断子弹是否击中中间的墙壁* **public boolean** hitWall(MiddelWalls wall) {  
 **if** (**this**.**live** && **this**.getRect().intersects(wall.getRectangle())) {  
 **this**.**live** = **false**;  
 **return true**;  
 }**else** {  
 **return false**;  
 }  
 }  
}

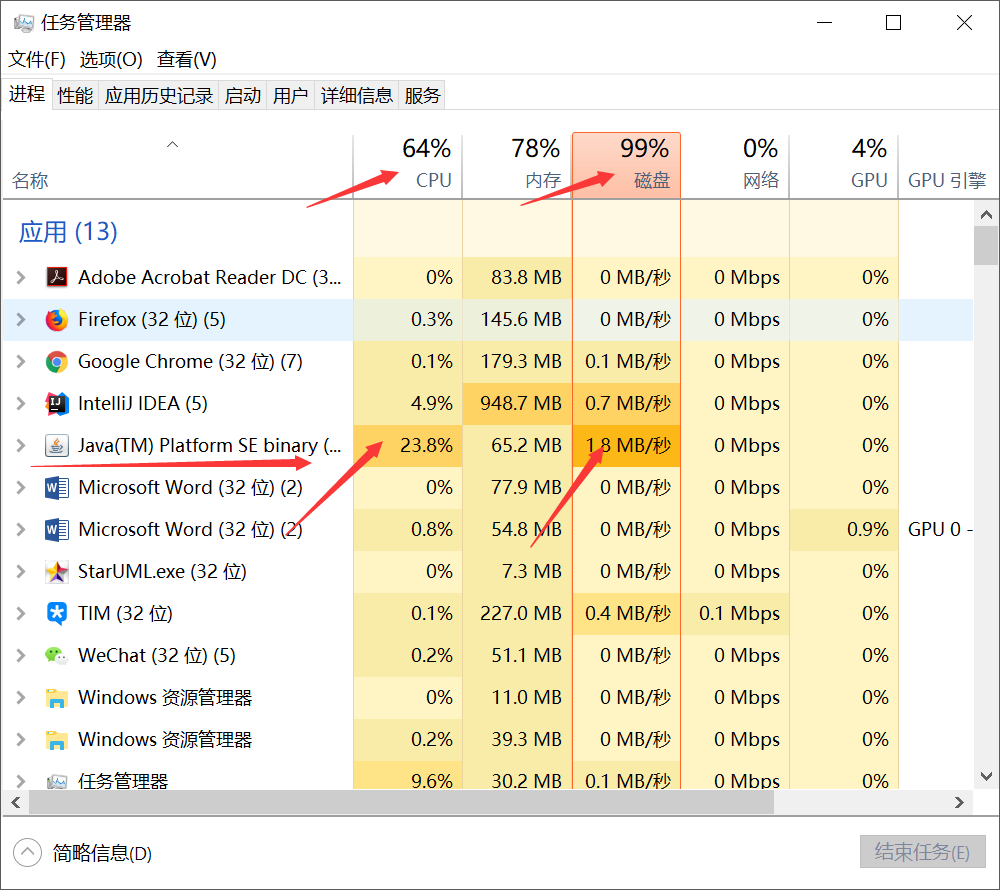
**public class** Main {  
  
 **public static void** main(String[] args) {  
 TankClient tc = **new** TankClient();  
 *//由于CPU和磁盘占用率过大....就不在界面里面添加音乐了* Music.*playMusic*();  
 tc.lauchFrame();  
 *// music.controlVaue();* }  
}

游戏的测试与发现：

1.游戏元素中的坦克，墙壁的定位需要慢慢的去找位置，利用截图工具一个一个找像素点定位：

定位成功之后：

一开始，完全把握不好插入图片的位置，在慢慢插入了几张图片之后，自己感觉对于在什么地方画出墙，在什么地方画出坦克的方法熟练了不少。

2.游戏的CPU和磁盘占用率太高了.. 

一开始我插入了16张图片但是发现CPU和磁盘占用率太高，而且游戏运行起来比较卡…后来查阅了下百度，也发现了一个关于磁盘读写的问题，就是每次读很多个小文件的总速度要比读一个很大的文件要慢的多。然后这让我重新想办法来解决这个问题，我用P图软件将几张图片和成一张图片加进去，这样做之后效果比之前的要好一些了。

总结心得：

总的来说这次课程设计耗费了我不少的时间和精力，同样的我感觉这次得到的收获也是不菲的。

从一开始的选择要做哪个题目，到选择了一个从来没有做过的图形界面，再到最后完成这次的课程设计，真的感觉自己的处理问题的能力以及分析东西的能力有了些长进，比如说要到网上面去查找各种各样的资料，和图片。这次的课程设计也让我充分的意识到了P图的重要性…一个从来不P图的我开始向用PS去修改下那个坦克的样式…然后那真的是下不了手啊，完全不会操作，后来下了个“美图秀秀”还是无济于事，最后还是请教了我的一个同学才弄好了这个图片的制作。

在编写这个“坦克战争”的过程中我着实遇到的很多的问题，但好在问题都还是一一解决了，并且在Windows和Linux两个系统上面都跑起来了。

通过这个小游戏的编写也让我真正感受到了效率问题，这个效率分为两个效率：一个是编译器的效率，一开始我用eclipse然后发现编写效率一般远没有intellij idea快，而且intellij idea的自动补全功能让我们基本上不可能再犯敲错关键字的尴尬，第二个效率就是在程序运行起来时候的效率问题，也就是上面曾经提到过的效率问题，插入大量的小图片相对而言要比添加少量大图片要慢。

最后我想说这个版本还不是最终的版本，我还有许多许多可以去改进的地方：

1. 整个游戏界面还不够美观…有一点low。
2. 没有能够添加一系列不同类型的墙体（可以被炮弹销毁的）以及没有给敌军一个和我们自身坦克有区别的图片样式。
3. 程序的敌对坦克还不能做到足够只能，不能自动的讲炮口对准我们操控的坦克。
4. 还没有实现多人对战，以及网络联机对战。

在以后的日子里我会将这个坦克大战变的更加完美漂亮。