

**厦门大学**

**JAVA程序设计报告**

**课题：绝地求生游戏**

小组成员：

赖劲垲 17420152200532

林竟鹏 17420152200559

高咏鸿 17420152200501

吴优 17420152200664

张鹏程 17420152200715

付识为 17420152200498

日期：2017/12/23

**目录**

[一、需求分析 4](#_Toc501843011)

[（一）开发背景 4](#_Toc501843012)

[（二）功能需求 4](#_Toc501843013)

[（三）开发环境 5](#_Toc501843014)

[二、设计调试 6](#_Toc501843015)

[（一）概要设计 6](#_Toc501843016)

[（二）详细设计 6](#_Toc501843017)

[三、运行界面 7](#_Toc501843018)

[（一）程序加载界面 7](#_Toc501843019)

[（二）主界面 7](#_Toc501843020)

[（三）登录与注册界面 8](#_Toc501843021)

[（四）玩法与个人信息 9](#_Toc501843022)

[（五）设置与开始游戏 10](#_Toc501843023)

[（六）游戏界面 11](#_Toc501843024)

[四、主要代码 12](#_Toc501843025)

[（一）Gui部分 12](#_Toc501843026)

[1、基本思路： 12](#_Toc501843027)

[2、UML示意图 15](#_Toc501843028)

[（二）地图部分 15](#_Toc501843029)

[1、基本思路 15](#_Toc501843030)

[2、具体描述 17](#_Toc501843031)

[地图显示 19](#_Toc501843032)

[（三）资源部分 24](#_Toc501843033)

[1、基本思路 24](#_Toc501843034)

[2、具体描述 24](#_Toc501843035)

[（四）AI部分 30](#_Toc501843036)

[1、基本思路 30](#_Toc501843037)

[2、具体描述 31](#_Toc501843038)

[（五）人物部分 39](#_Toc501843039)

[1、基本思路 39](#_Toc501843040)

[2、具体描述 40](#_Toc501843041)

[（六）其他部分 46](#_Toc501843042)

[1、数据库 46](#_Toc501843043)

[2、毒圈 47](#_Toc501843044)

[3、音乐 49](#_Toc501843045)

[4、计时器 51](#_Toc501843046)

[五、程序评价 52](#_Toc501843047)

[（一）优势与特色 52](#_Toc501843048)

[（二）缺点与不足 52](#_Toc501843049)

[六、心得体会 52](#_Toc501843050)

[七、参考资料 53](#_Toc501843051)

# 

# 一、需求分析

## （一）开发背景

近来，《绝地求生》游戏火遍各大社交圈与直播平台，该游戏是一款突破性的战术竞技类游戏，凭借其独特创新的玩法模式，写实风格带来的代入感和沉浸感深受玩家喜爱。

为了解该游戏运行与设计背后的机制、在实践中学习与提高Java编程能力，我们模仿《绝地求生》，设计了一款大逃杀类型的射击类单机游戏。

## （二）功能需求

游戏主要实现以下几个方面的功能

1. 用户的登录与注册
2. 主界面的各种游戏选项的设置
3. 玩家信息查看
4. 多种不同的游戏模式
5. 多种游戏地图
6. 不同难度等级的敌人
7. 资源随机生成与消失
8. 多种武器攻击方式
9. 自动缩小的毒圈

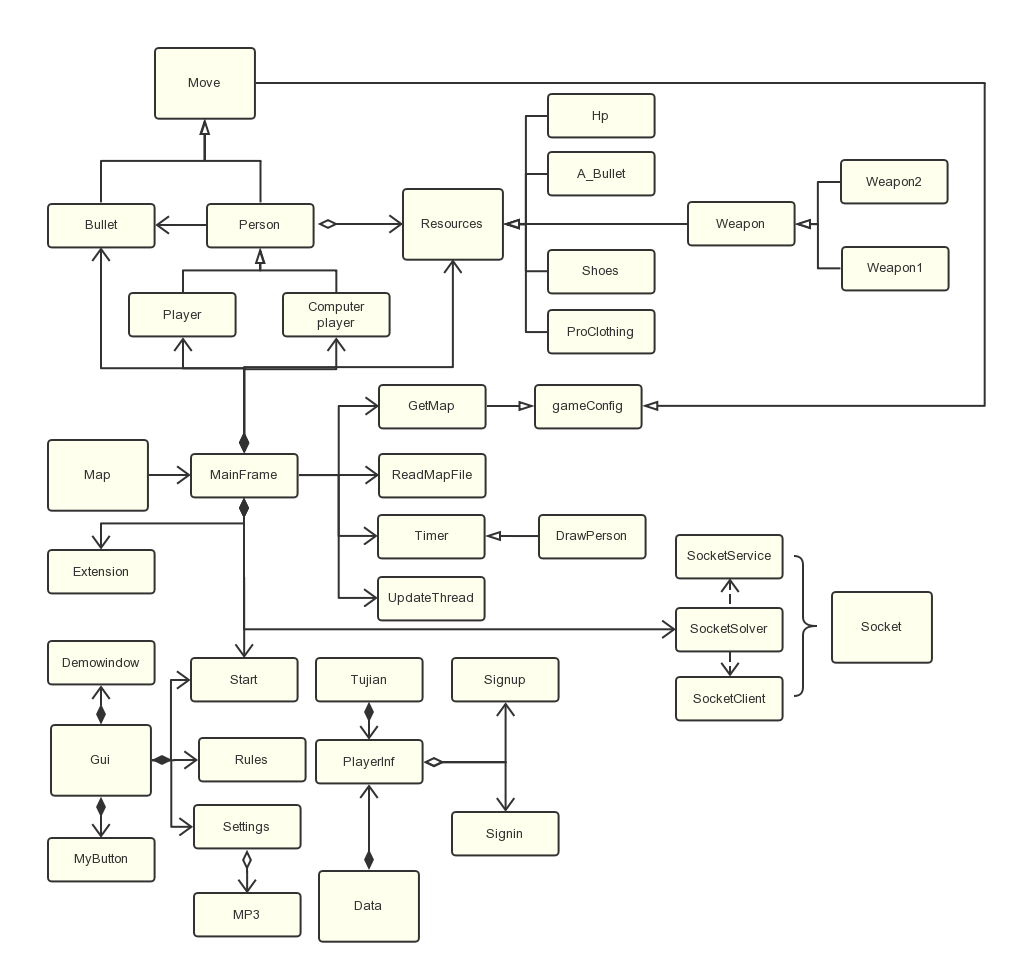
除此之外尚有许多零星小点，所有功能都将在下文代码部分进行详细解释。

## （三）开发环境

Windows8.1/10操作系统，eclipse版本号：Oxygen Release (4.7.0)

# 二、设计调试

## （一）概要设计



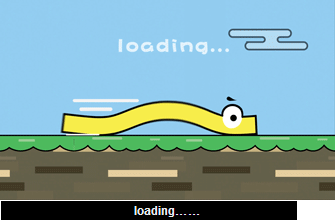
以上为我们程序的UML图

## （二）详细设计

代码主要分为数据库（data），主界面（GUI），玩家与AI（man）,地图（map），毒圈以及音乐和计时器等（others），资源（resources），网络传输（socket）几个部分，每个部分下设若干类，主要代码解释见下文。

# 三、运行界面

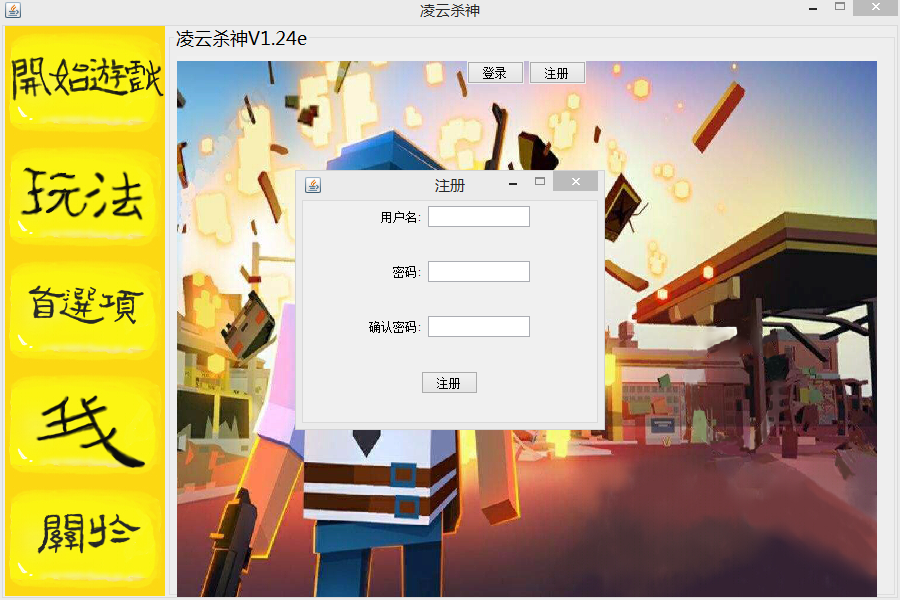
## （一）程序加载界面

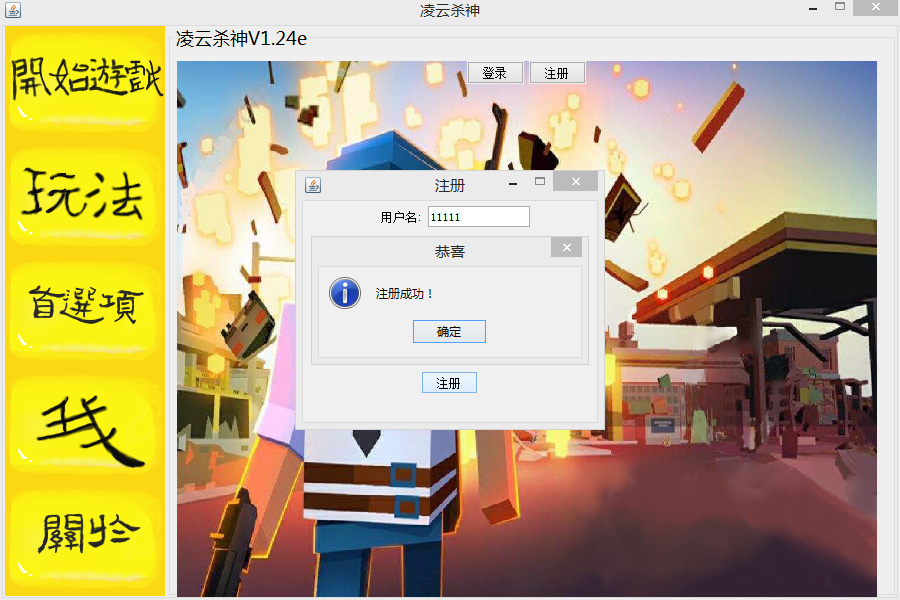


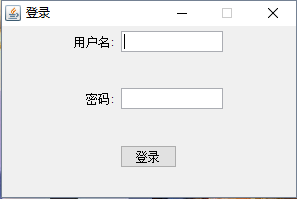
## （二）主界面



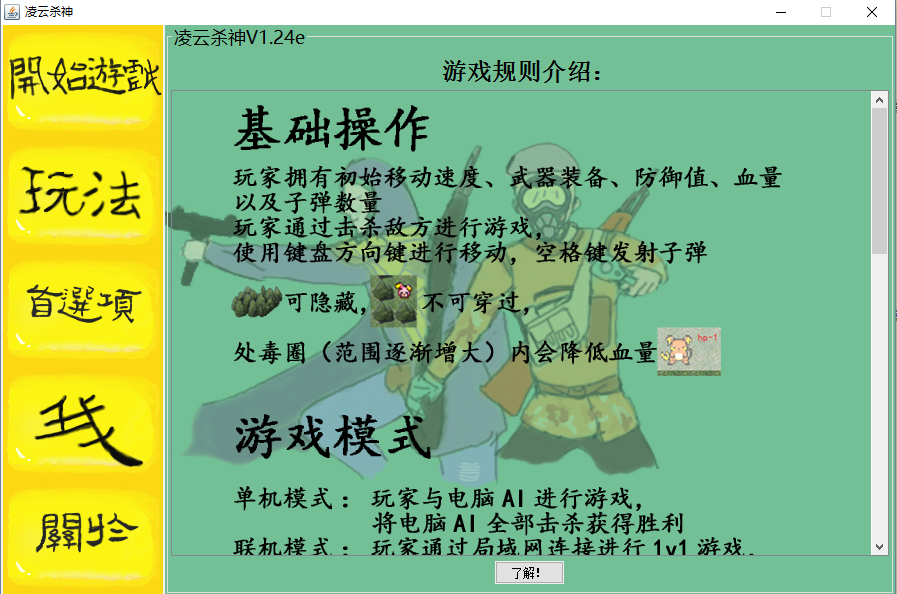
## （三）登录与注册界面





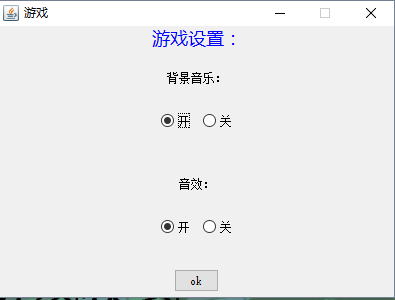


## （四）玩法与个人信息



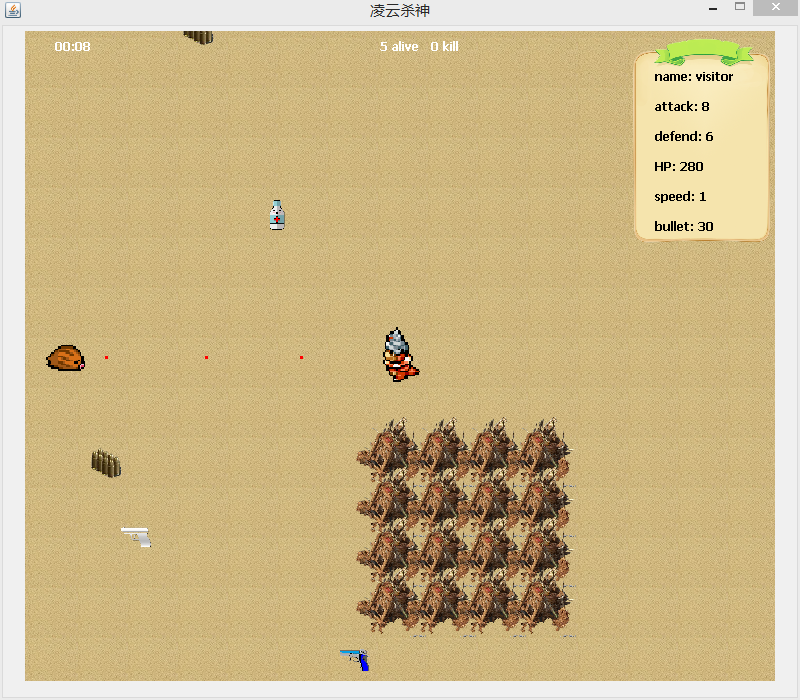


## （五）设置与开始游戏





## （六）游戏界面





# 四、主要代码

## （一）Gui部分

### 1、基本思路：

Gui部分主要设计游戏菜单界面，为其他进程提供入口，获取用户输入等。

运行游戏后，首先呈现的是一个悬浮的加载进度窗口，如下图：

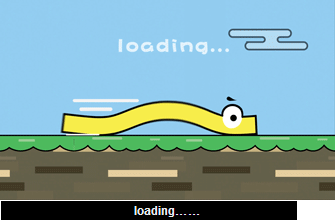


图1.1

实现的类为DemoWindow，主要代码包括：

**public** **class** DemoWindow **extends** JWindow **implements** Runnable {

**private** JProgressBar progress = **new** JProgressBar(1,100);

ImagePanel ip = **new** ImagePanel();

**public** DemoWindow(){

add(ip,BorderLayout.***CENTER***);

add(progress,BorderLayout.***SOUTH***);

//进度条

这部分声明了窗口的两部分组成，上面是图片，下面是进度条。

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

**try**{

**for**(**int** i = 0;i < 100;i ++){

Thread.*sleep*(20);

progress.setValue(progress.getValue()+1);

}

}**catch**(Exception e){

System.***out***.println(e.getMessage());

}

**this**.dispose();

这个类声明了一个函数run()，用来实现进度条的运行，这里设置的是按时间来加载。当加载完成之后，用dispose()关闭并释放加载窗口，进入下面的代码：

Gui gui = **new** Gui();

gui.setTitle("凌云杀神");

gui.setLocationRelativeTo(**null**);

gui.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

gui.setResizable(**false**);

gui.setVisible(**true**);

这部分实例化Gui类，产生一个新的游戏菜单窗口，并设置其各项参数。

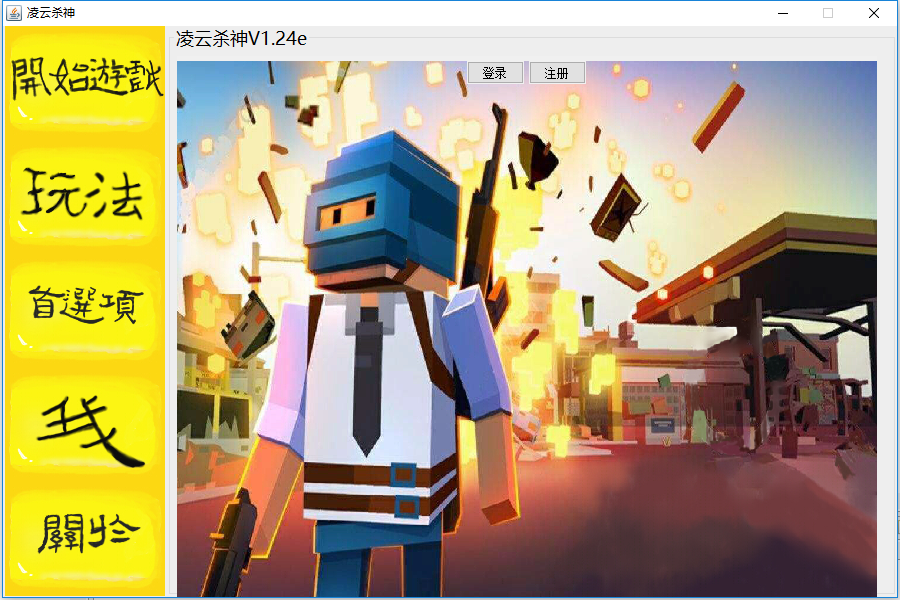


图1.5

简要介绍一下主界面的构成。整个窗体布局为BorderLayout，西侧为按钮部分：

buttonPanel.add(b01);

buttonPanel.add(b02);

buttonPanel.add(b03);

buttonPanel.add(b04);

buttonPanel.add(b05);

buttonPanel.setLayout(**new** GridLayout(5,1));

add(buttonPanel,BorderLayout.***WEST***);

剩下的页面为卡片页cardPanel占据，它包含3个部分，默认显示的layerPane、点击“玩法”按钮后出现的规则页面panel1和点击“我”之后的个人信息界面pi。

//玩法介绍

panel1 = **new** Rules();

panel1.back.addActionListener(**this**);

//个人信息

pi = **new** PlayerInf(*sql*);

pi.back.addActionListener(**this**);

pi.tujian.addActionListener(**this**);

cardPanel.add(cardName[0],layeredPane);

cardPanel.add(cardName[1],panel1);

cardPanel.add(cardName[2], pi);

card.show(cardPanel, cardName[0]);

add(cardPanel,BorderLayout.***CENTER***);

另外还有三个按钮，分别是“开始游戏”、“首选项”和“关于”，点击会弹出独立的小窗口。“关于”窗口比较简单，直接写在了Gui大类里面。另外两个小窗口写在独立的类Start和Settings里。

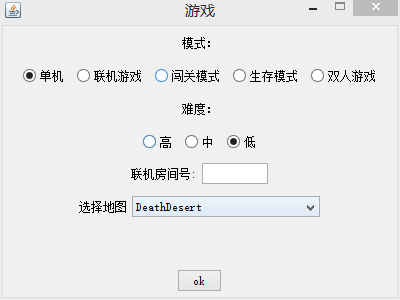


图1.8

另外Gui部分也为其他功能提供了入口，比如注册窗口对应的类Signin，它提供了name()方法和password()方法，方便其他类来调用用户的输入数据，并存储在数据库内。

### 2、UML示意图



## （二）地图部分

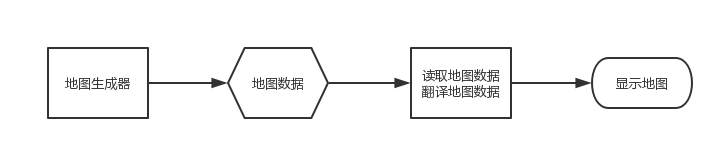
1、基本思路

本部分为地图部分，主要解决的问题是：地图的显示、人物（物品）的显示、人物的移动。

首先，对于地图的显示而言它是由许多个元素拼接而成，用二维数组作为地图中各个元素储存的媒介是十分方便的。因为每一个元素均是一个小方块，其在地图中有一个可以用 (x , y) 坐标表示的相对其他元素的位置，而二维数组可以很自然地定位其中储存的每一个数相对其他数用 (x , y)坐标表示的相对位置。所以，我们只需要按照元素在地图中存在的相应位置将其按照顺序存放入一个二位数组，需要显示地图时再按照顺序将其读出即可。其次，我们要做的是按照二位数组中存放的信息将各个元素拼接起来。二位数组可以定位每个元素的位置，却不方便直接存放图片，因此我们将所需的元素按照一定的原则编码（如：000代表“000地板.png”）再将其储存入数组。在需要显示地图时，根据先前的编码原则将从数组中读取的数字“翻译”成对应的元素，再将其按照顺序拼接起来即可。

对于人物（物品）的显示而言，我们只需要在画出地图之后，再在地图的背景上画出人物（物品即可）。

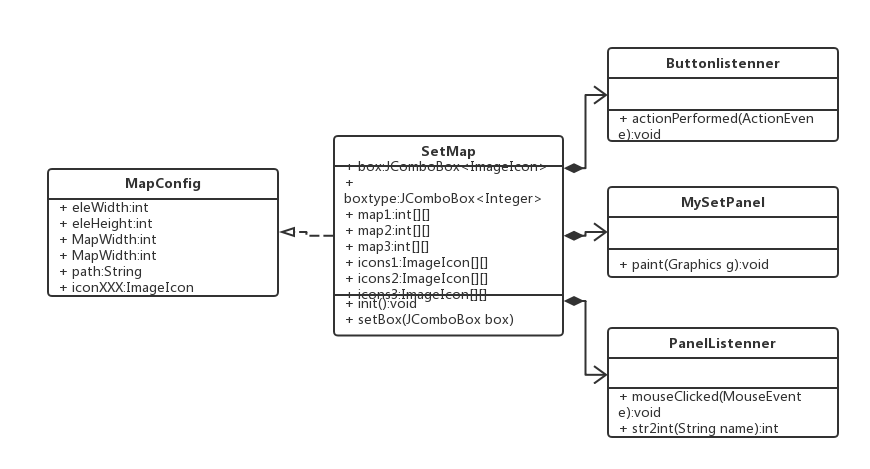
对于人物的移动而言，很自然的想法是在一个固定的背景上移动人物，但是这样做会产生一些麻烦。因为我们一个窗口显示的地图的范围是有限的，我们为了能够在一个有限的窗口中实现在一个很大的地图上走动，在移动人物的同时我们也会需要移动地图。这样就会造成人物与地图两者之间在二位数组的某个范围内需要发生相对移动，这样需要考虑的变量与情况将会比较多。但是为了产生移动人物的效果，我们可以让人物固定在画布的某一点，我们只需要判断人物在二位数组中的相对位置然画出周围一定范围内的地图即可。



### 2、具体描述

#### 地图生成

为了方便地图的生成我们制作了一个地图生成器，我们可以通过直接通过对图形界面的操作来设计、生成地图。其UML图如下：



MapConfig 为一个接口，在其中规定了每个元素的长和宽，地图总的长和宽，生成地图的名字，每种元素的名字。

//元素大小

**int** ***eleWidth*** = 50;

**int** ***eleHeight*** = 50;

//地图大小

**int** ***MapWidth*** = 2000;

**int** ***MapHeight***= 2000;

//图片元素

ImageIcon ***icon0*** = **new** ImageIcon("000草地.png");

ImageIcon ***icon002*** = **new** ImageIcon("002普通地砖.png");

SetMap 为实现了MapConfig的一个类。其中规定了三个数组map1[][]，map2[][]，map3[][]。这三个数组均为40\*40的整数型，其目的是储存地图中的元素的相对位置。map1[][]储存人物脚下的地图（地板），map2[][]储存与人物同意平面的地图（障碍物），map3[][]储存人物透顶的地图（遮蔽物）。

Buttonlistener中设置了一个“地图储存”的按钮，设计完成之后使用FileOutputString以及DataOutputString于指定的路径生成保存有地图信息的文件。

**class** Buttonlistenner **implements** ActionListener{

@Override

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

**if**(e.getActionCommand().equals("create")){

**try**{

System.***out***.println("地图成长");

FileOutputStream fos = **new** FileOutputStream(***path***);

DataOutputStream dos = **new** DataOutputStream(fos);

**int** i = ***MapHeight***/***eleHeight***;

**int** j = ***MapWidth***/***eleWidth***;

dos.writeInt(i);

dos.writeInt(j);

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

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

dos.writeInt(*map1*[ii][jj]);

dos.writeInt(*map2*[ii][jj]);

dos.writeInt(*map3*[ii][jj]);

}

}

dos.flush();

dos.close();

System.***out***.println("OK");

}**catch**(Exception ef){

ef.printStackTrace();

}

}

}

}

MySetPanel 中在模拟的地图界面上画出各种元素，使得我们可以对图片进行操作，从而简单、直观地设计地图。

PanelListenner 中我们使用了str2int(String name):int 方法，其目的是将个图片元素翻译为对应的int型数据，从而可以储存在数组中。而mouseClicked（MouseEvent e）:void方法中，我们点击模拟地图上的一个位置，就将确定元素对应的int型数据储存在数组的相应位置。

**public** **int** str2int(String numstr){

**for**(**int** i=0;i<3;i++){

**if**(numstr.charAt(i)!=0){

numstr = numstr.substring(i);

**int** num = Integer.*parseInt*(numstr);

**return** num;

}

}

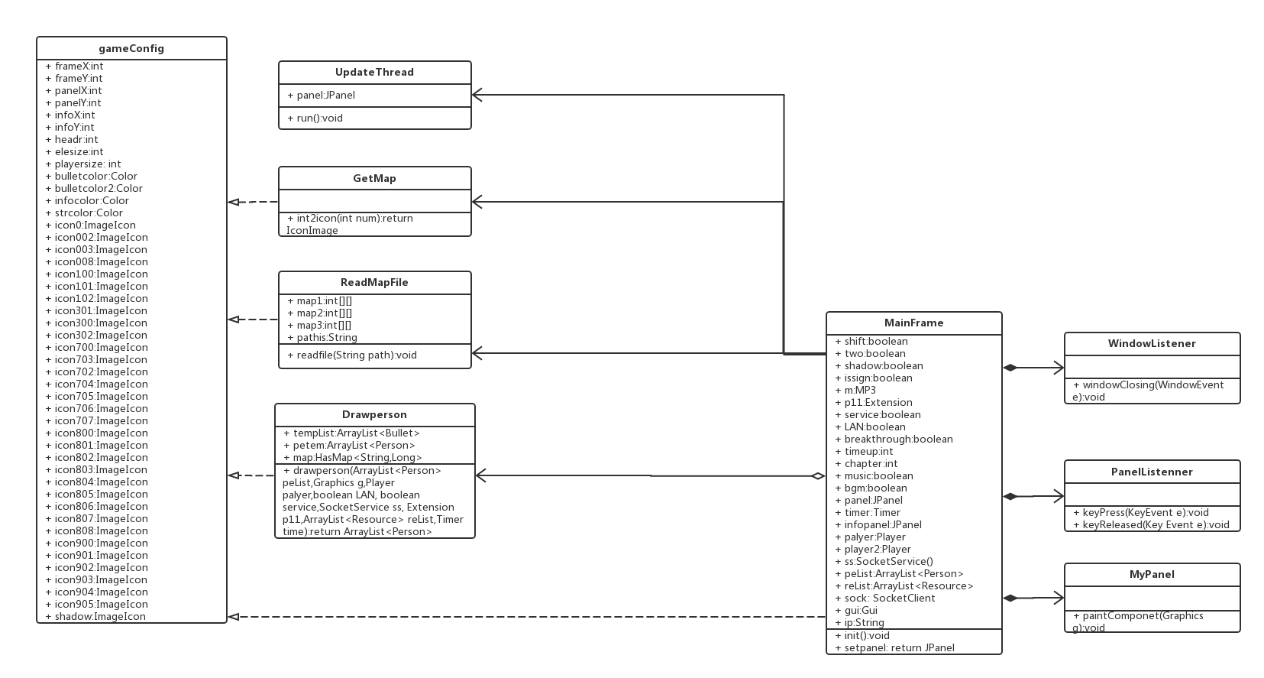
numstr = numstr.substring(2);

**int** num = Integer.*parseInt*(numstr);

**return** num;

}

#### **地图显示**

实现地图显示以及人物显示的代码位于map包中，其UML如下图所示：

gameConfig 为一个接口，图中的UpdateThread类、GetMap类、ReadMapFile类、Drawperson类、MainFrame类均实现了本接口。在这个接口中，类似于地图生成部分中的MapConfig接口，在此处确定了游戏中有关地图各种元素的一数据，方便在实现这个接口的类中直接使用。同时为了增加游戏的难度，在游戏素材中我们加入了阴影镜头来限制游戏者的可视范围。

//游戏主题窗口大小

**int** ***frameX*** = 800;

**int** ***frameY*** = 700;

//游戏面板大小

**int** ***panelX*** = 750;

**int** ***panelY*** = 650;

//信息面板大小

**int** ***infoX*** = 150;

**int** ***infoY*** = 220;

//头像边长

**int** ***headr*** = 50;

//游戏素材大小（正方形）

**int** ***elesize*** = 50;

//人物大小（一格一格移动）

**int** ***playersize*** = 50;

Color ***bulletcolor***=Color.***YELLOW***;

Color ***bulletcolor2***=Color.***RED***;

Color ***infocolor***=Color.***BLACK***;

Color ***strcolor***=Color.***WHITE***;

//------------[游戏素材]----------

ImageIcon ***icon0***= **new** ImageIcon("mapresource/000草地.png");

ImageIcon ***icon002*** = **new** ImageIcon("mapresource/002普通地砖.png");

//限制视角的镜头

ImageIcon ***shadow*** = **new** ImageIcon("镜头阴影2.png");

UpdateThreads 类为面板刷新线程类，其休眠时间为10，在MainFrame类中有使用到。

ReadMapFile 类中的readFile(String path):void方法是对之前地图生成部分的生成的地图文件从指定的path读入，得到文件输入流，之后再将文件流包装成基本数据输入流，再将得到的数据数据存放至相应的数组内。

**public** **static** **void** readfile(String path){

**try**{

*pathis*=path;

//从指定的路径path读入地图，以后可以换地图

FileInputStream fis = **new** FileInputStream(path);

//翻译地图，将文件输入数据流包装成基本数据流

DataInputStream dis = **new** DataInputStream(fis);

//读取三个地图数组

**int** i = dis.readInt();

**int** j = dis.readInt();

*map1* = **new** **int**[i][j];

*map2* = **new** **int**[i][j];

*map3* = **new** **int**[i][j];

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

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

*map1*[ii][jj] = dis.readInt();

*map2*[ii][jj] = dis.readInt();

*map3*[ii][jj] = dis.readInt();}

}

dis.close();

fis.close();

}**catch**(Exception e){

e.printStackTrace();

}}}

getMap类是对之前地图生成部分中PanelListenner类中str2int(String name):int 方法的反向翻译，其目的是将从数组中读取的int型整数翻译成对应的元素图片。

MainFrame类以及DrawPerson类以及与之相关的windowlistenner类，PanelListenner类，MyPannel类的目的是画出地图以及游戏界面的各个部分，包括人物（人物的四种朝向），相关的物品（随机生成的，游戏玩家可拾取的子弹、血包等；射出的子弹），游戏的相关的信息框（子弹数，Hp值等），地图的各个元素（人物脚下的、同一平面的、头顶遮蔽的），随着时间流逝扩大的毒圈，为增大难度而限制的玩家视野，以及游戏的背景音乐。因为这一部分代码众多，本处只展现代表逻辑的重要代码。代码如下：

//人物脚下与同一层次的地图

**for**(**int** i=player.getI()-8;i<=player.getI()+8;i++){

**for**(**int** j=player.getJ()-9;j<=player.getJ()+9;j++){

**if**(i>=0&&j>=0&&i<ReadMapFile.*map1*.length&&j<ReadMapFile.*map1*[0].length){

ImageIcon icon1 = GetMap.*int2icon*(ReadMapFile.*map1*[i][j]);

g.drawImage(icon1.getImage(), (player.px-***elesize***/2)+((j-player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+((i-player.getI())\****elesize***)-(player.my%***elesize***), ***elesize***, ***elesize***, **null**);

**if**(ReadMapFile.*map2*[i][j]!=0)

{

ImageIcon icon2 = GetMap.*int2icon*(ReadMapFile.*map2*[i][j]);

System.***out***.println(ReadMapFile.*map2*[i][j]);

g.drawImage(icon2.getImage(), (player.px-***elesize***/2)+((j-player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+((i-player.getI())\****elesize***)-(player.my%***elesize***), ***elesize***, ***elesize***, **null**);

}

}

//player.px-elesize/2为主角在屏幕上开始画的坐标

//(j-player.getJ())\*elesize为主角与该地图块之间的水平距离

//player.mx%elesize代表主角相对与脚下地图块的x起始点的差值，即主角离脚下地图块左上角的水平距离

**else**//在地图的外围画上海洋

{

g.drawImage(***icon008***.getImage(), (player.px-***elesize***/2)+((j-player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+((i-player.getI())\****elesize***)-(player.my%***elesize***), ***elesize***, ***elesize***, **null**);

}

}

}

//获取毒圈 玩家信息

**if**(per.alive)

{

**if**(!LAN||service)

{

**if**(MainFrame.*timeup*==0)

((ComputerPlayer) per).getInformation(p11.wlimit/2-p11.width,peList, reList );

**else**

{ArrayList<Person> pp=**new** ArrayList<Person>();

pp.add(player);

((ComputerPlayer) per).getInformation(p11.wlimit/2-p11.width,pp, reList );

}

}

((ComputerPlayer) per).draw(g,player);

*petem*.add(per);

}

//人物头顶的第三层的掩盖层

**for**(**int** i=player.getI()-7;i<=player.getI()+7;i++)

{

**for**(**int** j=player.getJ()-8;j<=player.getJ()+8;j++)

{ **if**(i>=0&&j>=0&&i<ReadMapFile.*map3*.length&&j<ReadMapFile.*map3*[0].length&&ReadMapFile.*map3*[i][j]!=0)

{

ImageIcon icon3 = GetMap.*int2icon*(ReadMapFile.*map3*[i][j]);

Image ico=icon3.getImage();

g.drawImage(ico, (player.px-***elesize***/2)+((j-player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+((i-player.getI())\****elesize***)-(player.my%***elesize***), ***elesize***, ***elesize***, **null**);

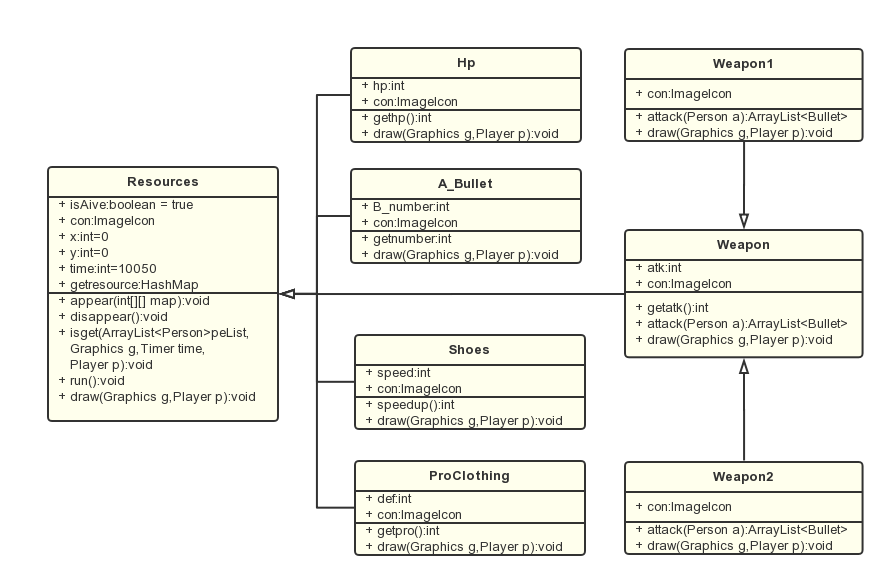
}

}

}

## （三）资源部分

1、基本思路

游戏资源包（Package resources）中有类Resources、Hp、A\_Bullet、Shoes、ProClothing、Weapon、Weapon1以及Weapon2八个类，其中Resources为所有类的父类，Weapon为Weapon1以及Weapon2的父类，具体UML图示如下。

### 2、具体描述

首先是类Resources，

appear(int[][] map)方法用来在地图上画出血包、鞋子等各种资源，通过map[][]传递整个地图的大小。当确保资源位置会处于游戏地图内时，通过随机函数随机画出每个资源的位置（x、y值）。并且按照游戏设定，每个资源都有在固定范围上的随机数值，因此在通过随机函数对所有画在地图上的资源进行属性数值上的赋值。其中，因为武器拥有不同的攻击方式，所以设定的攻击力范围也不同。具体细节代码设计思路体现在下面代码及解释中。

**public** **void** appear(**int**[][] map)

{

**int** xlength=(map[0].length-1)\****elesize***; //防止将资源画出地图外或卡在地图上

**int** ylength=(map[0].length-1)\****elesize***; //防止将资源画出地图外或卡在地图上

Random rand = **new** Random();//随机函数

x=y=0;

**while**(ReadMapFile.*map2*[y/***elesize***][x/***elesize***]!=0)//随机画出资源的位置x、y

{

x = rand.nextInt(xlength);

y = rand.nextInt(ylength);

}

//为每个资源分配随机的数值

**switch**(**this**.getClass().getName())

{

**case** "resources.Hp":

((Hp)**this**).hp=rand.nextInt(15)+10;

**break**;

**case** "resources.Shoes":

((Shoes)**this**).speed=1;

**break**;

**case** "resources.Weapon":

((Weapon)**this**).atk=rand.nextInt(9)+10;//武器攻击方式不同，武器的攻击力数值范围也不同

**break**;

**case** "resources.Weapon1":

((Weapon)**this**).atk=rand.nextInt(9)+5;

**break**;

**case** "resources.Weapon2":

((Weapon)**this**).atk=rand.nextInt(9)+3;

**break**;

**case** "resources.ProClothing":

((ProClothing)**this**).def=rand.nextInt(3)+1;

**break**;

**case** "resources.A\_Bullet":

((A\_Bullet)**this**).B\_number=rand.nextInt(10)+10;

**break**;

**default**:

**break**;

}

}

Run()方法用来调动每个资源的出现时间，当资源出现（调用appear()方法）后会在原地出现10.05秒，当资源被person拾取后，会在经历过1/4 time的时间后再随机出现在其他位置。

int map[][]=new int[ReadMapFile.map1.length][ReadMapFile.map1[0].length];//传入地图大小

public void run() {

while(true){

if(!this.isAlive)//当资源被拾取后

{try {

Thread.sleep(time/4);

} catch (InterruptedException e) {

e.printStackTrace();

}}

this.isAlive=true;

appear(map);//画出资源

try {

Thread.sleep(time);//资源出现时间

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

isget()方法用来判断person拾取到资源后的属性值增加。当person的位置可以拾取到资源时（若该person为游戏玩家，则打开拾取资源的游戏音效），通过switch函数判断该person拾取到的资源种类，在不超过每样数值的规定上限下，增加person的属性数值，并出现person拾取资源的提示。

**public** **void** isget(ArrayList<Person> peList,Graphics g,Timer time,Player p)

{ //当地图为沙漠或者雪地是改变子弹颜色

**if**(ReadMapFile.*pathis*.equals("map/SnowWorld.map")||ReadMapFile.*pathis*.equals("map/DeathDesert.map"))

g.setColor(***bulletcolor2***);

**else**

g.setColor(***bulletcolor***);

**for**(Person per:peList)

{

**if**((per.x+***elesize***/2>**this**.x&&per.x-***elesize***/2<**this**.x) && (per.y+***elesize***/2>**this**.y&&per.y-***elesize***/2<**this**.y))//判断person是否可以拾取到资源

{

**if**(per.personid==p.personid&&MainFrame.*music*)

{

MP3 ss=**new** MP3("get.mp3");

ss.isloop=**false**;

ss.start();

}//打开拾取资源的游戏音效

**int** num;

**switch**(**this**.getClass().getName())//增加属性数值

{

**case** "resources.Hp":

**if**(per.hp+((Hp)**this**).gethp()>per.hplimit)//判断是否超过上限

{

num=per.hplimit-per.hp;

per.hp+=num;

}

**else**

{

num=((Hp)**this**).gethp();

per.hp+=num;

}

*getresource*.put(**new** Object[]{per.name+" HP+"+num,**this**.x,**this**.y},time.whattime());//出现提示

**break**;

**case** "resources.Shoes":

**if**(per.step<4)

{per.step+=((Shoes)**this**).speedup();

num=((Shoes)**this**).speedup();

}

**else**

num=0;//防止速度过快失去游戏平衡

per.speedup+=500;

*getresource*.put(**new** Object[]{per.name+" SPEED+"+num,**this**.x,**this**.y},time.whattime());

**break**;

**case** "resources.Weapon":

per.bulletnum+=10;//武器附带子弹

per.wp=(Weapon) **this**;

per.atk=((Weapon)**this**).getatak();

*getresource*.put(**new** Object[]{per.name+" get normal gun",**this**.x,**this**.y},time.whattime());

**break**;

**case** "resources.Weapon1":

per.bulletnum+=10;

per.wp=(Weapon) **this**;

per.atk=((Weapon)**this**).getatak();

*getresource*.put(**new** Object[]{per.name+" get scatter gun",**this**.x,**this**.y},time.whattime());

**break**;

**case** "resources.Weapon2":

per.bulletnum+=10;

per.wp=(Weapon) **this**;

per.atk=((Weapon)**this**).getatak();

*getresource*.put(**new** Object[]{per.name+" get strongger gun",**this**.x,**this**.y},time.whattime());

**break**;

**case** "resources.ProClothing":

**if**(per.def<18) {

per.def+=((ProClothing)**this**).getPro();

num=((ProClothing)**this**).getPro();

}

**else**

num=0;//防止防御值过高失去游戏平衡

*getresource*.put(**new** Object[]{per.name+" DEF+"+num,**this**.x,**this**.y},time.whattime());

**break**;

**case** "resources.A\_Bullet":

per.bulletnum+=((A\_Bullet)**this**).getnumber();

num=((A\_Bullet)**this**).getnumber();

*getresource*.put(**new** Object[]{per.name+" BULLET+"+num,**this**.x,**this**.y},time.whattime());

**break**;

**default**:

**break**;

}

HashMap<Object[],Long> tmap=**new** HashMap<Object[],Long>();

**for**(Object[] str:*getresource*.keySet())//画出person拾取到资源后的提示

{

**if**(*getresource*.get(str)>time.whattime())

{

*getresource*.clear();

**break**;

}

**if**(time.whattime()-*getresource*.get(str)<1000)

{

g.drawString(str[0].toString(),(p.px-***elesize***/2)+(Integer.*parseInt*(str[1].toString())-***elesize***/2-(p.getJ())\****elesize***)-(p.mx%***elesize***), (p.py-***elesize***/2)+(Integer.*parseInt*(str[2].toString())-***elesize***/2-(p.getI())\****elesize***)-(p.my%***elesize***)); tmap.put(str,*getresource*.get(str));

tmap.put(str,*getresource*.get(str));

}

}

*getresource*.clear();

*getresource*.putAll(tmap);

**this**.disappear();

**return**;

}

}

HashMap<Object[],Long> tmap=**new** HashMap<Object[],Long>();

**for**(Object[] str:*getresource*.keySet())

{

**if**(*getresource*.get(str)>time.whattime())

{

*getresource*.clear();

**break**;

}

**if**(time.whattime()-*getresource*.get(str)<1000)

{

g.drawString(str[0].toString(),(p.px-***elesize***/2)+(Integer.*parseInt*(str[1].toString())-***elesize***/2-(p.getJ())\****elesize***)-(p.mx%***elesize***), (p.py-***elesize***/2)+(Integer.*parseInt*(str[2].toString())-***elesize***/2-(p.getI())\****elesize***)-(p.my%***elesize***));

tmap.put(str,*getresource*.get(str));

}

}

*getresource*.clear();

*getresource*.putAll(tmap);

}

Draw()方法就是通过x、y值将资源画出来。

Resources的子类中，Hp、A\_Bullet、Shoes、ProClothing、Weapon、Weapon1以及Weapon2中的属性及方法较为简单，即定义属性和图片样式，返回属性值的方法和覆盖父类draw()方法外没有其他方法。

下面介绍下weapon中的主要代码，即射出子弹的代码。

attack(person a)方法返回一个bullet的arraylist，用来定义武器的攻击方式。Weapon中的attack()方法定义了最基础的攻击方式，首先是new一个新的bullet，通过start()调用Bullet中的方法，并将该bullet传进定义的arraylist里面。

在weapon1以及weapon2中，则是增加了子弹的个数，weapon1改变了子弹的朝向，weapon2改变了子弹的位置，于是产生了不同的子弹攻击方式

**public** ArrayList<Bullet> attack(Person a)//射出子弹

{

ArrayList<Bullet> k=**new** ArrayList<Bullet>();

Bullet b1=**new** Bullet(a);

b1.start();

**if**((a.up&&a.down)||(a.left&&a.right)||!a.isalive())

{

b1.alive=**false**;

}

**else**

k.add(b1);

**return** k;

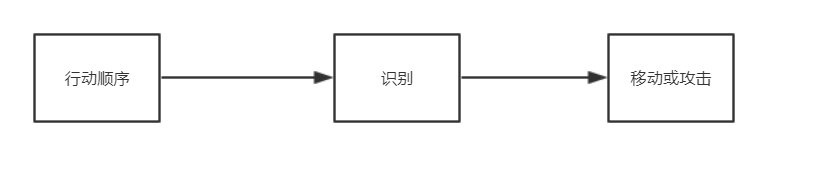
}

## （四）AI部分

### 1、基本思路

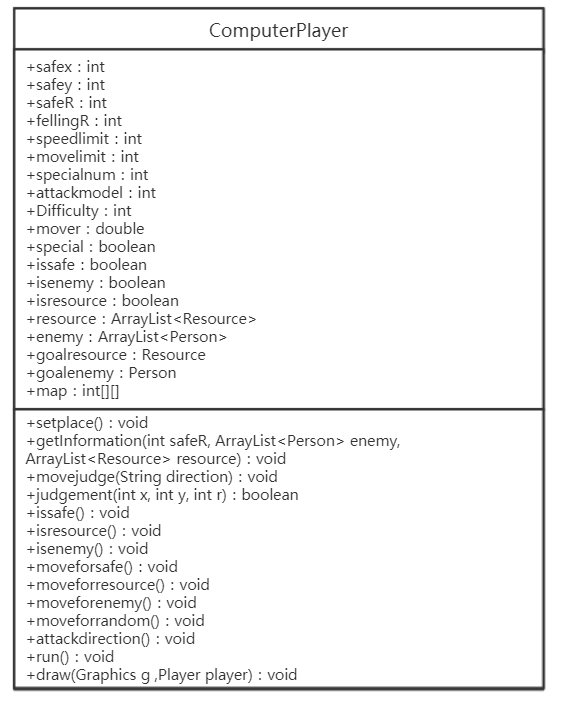
本部分为电脑玩家（以下简称AI）部分，主要解决的问题是：电脑玩家的行为模式。

对于AI来说，其行为模式包括识别，移动模式，攻击模式，行动顺序。先通过行动顺序决定识别的先后顺序，在识别过程中，设定一个目标，然后向目标进行移动或攻击，从而完成行动。



### 2、具体描述

本部分代码位于man包下的ComputerPlayer类，其UML图如下



#### 识别

识别部分包括是否安全，是否有敌人，是否有资源，是否处于特殊状态的判断。

是否安全通过issafe()判断，判断AI是否在毒圈中，从而改变issafe的值，safex，safey为毒圈的中心点，safeR为半径。

**public** **void** issafe() {

issafe = **true**;

**if** (! (x<safex+safeR && x>safex-safeR && y<safey+safeR && y>safey-safeR) )

issafe = **false**;

}

是否有敌人通过isenemy()判断，通过enemy这个list获得所有人物的信息，如果人物在AI的感知范围内(fellingR)且在毒圈内，则设成目标敌人，之后再进行墙体的相关判断，如果移动过程中会碰到墙体（死路或原地不动），或者子弹射出后会碰到墙体则放弃目标敌人，最后进行草丛的判断，如果敌人处于草丛，则放弃。这部分代码主要工作是进行墙体的判断（分成八个方向，判断敌人与AI间是否有墙）。代码在文件中213到334行。下面贴出部分

//目标敌人的设定

**public** **void** isenemy() {

isenemy = **false**;

goalenemy = **null**;

**for** (Person per :enemy) {

**if** (per.personid != **this**.personid)

**if** (per.isalive())

**if** (judgement (per.x, per.y, feelingR))

**if** (per.x<safex+safeR && per.x>safex-safeR && per.y<safey+safeR && per.y>safey-safeR){

goalenemy = per;

isenemy = **true**;

//左右移动有墙则放弃目标敌人，ReadMapFile.map[][]=2为墙

**if**(y/***elesize***==per.y/***elesize*** && x/***elesize***==per.x/***elesize***) {

}**else** {

**if**(per.x > x) {

**if**(ReadMapFile.*map2*[y/***elesize***][x/***elesize***+1]!=0) {

goalenemy =**null**;

isenemy = **false**;

special=**true**;

}

}**else** **if**(per.x < x){

**if**(ReadMapFile.*map2*[y/***elesize***][x/***elesize***-1]!=0) {

goalenemy =**null**;

isenemy = **false**;

special=**true**;

}

}

//目标敌人在AI的左上角，如果与AI间有墙则放弃

}**else** **if**( x > goalenemy.x+100 && y > goalenemy.y+100){

**for** (**int** i=0;i<8;i++)

**if** (ReadMapFile.*map2*[y/***elesize***-i][x/***elesize***-i]!=0 ||ReadMapFile.*map2*[y/***elesize***-i-1][x/***elesize***-i]!=0||ReadMapFile.*map2*[y/***elesize***-i][x/***elesize***-i-1]!=0) {

isenemy=**false**;

goalenemy=**null**;

special=**true**;

**break**;

}

//目标敌人在草丛中则放弃 map[][]记录草丛的信息

**try** {

**if** (map[goalenemy.y/***elesize***][goalenemy.x/***elesize***]!=0) {

isenemy=**false**;

goalenemy=**null**;

}

}**catch**(NullPointerException e){

}

是否有资源通过isresource()判断，通过resource这个list传递所有的资源信息，如果资源在AI的感知范围内，且在毒圈内，则设为目标资源，之后再进行墙体的相关判断，如果在获取资源的过程中，会碰到墙体（死路或原地不动），则放弃。本部分代码在文件中136到210行，主要也是进行墙体判断（分成8种情况）

//设定目标资源

isresource = **false**;

goalresource = **null**;

**for** (Resource res :resource) {

**if** (res.isAlive)

**if** (judgement (res.x, res.y, feelingR))

**if** (res.x<safex+safeR && res.x>safex-safeR && res.y<safey+safeR && res.y>safey-safeR) {

goalresource = res;

isresource = **true**;

//资源在人物右下角，如果人物左边，下边都有墙，则放弃

**if**(y/***elesize***==res.y/***elesize*** && x/***elesize***==res.x/***elesize***) {

}**else** {

**if**(res.x > x) {

**if**(ReadMapFile.*map2*[y/***elesize***][x/***elesize***+1]!=0) {

**if**(res.y >y) {

**if**(ReadMapFile.*map2*[y/***elesize***+1][x/***elesize***]!=0) {

goalresource = **null**;

isresource = **false**;

special=**true**;

}

是否处在特殊情况通过special这个属性判断，特殊情况包括AI因某些原因无法动弹，AI在不同识别状态中不断切换，special属性穿插于issafe(),isresource(),isenemy()，moveforresource（）等方法中。

#### 移动或攻击模式

这部分包括向资源移动，向敌人移动，向安全地点移动，攻击模式，移动判断，随机移动。

移动判断即让AI在移动中不会出现穿墙行为，分成上下左右四个方向进行判断。本部分代码在59到118行。下面贴出部分。

//向上移动，如果上一格是墙，则坐标不变，如果不是墙则可以变。

**if**(direction=="up"){

**if**(y/***elesize***-1>0)

{

**if**(ReadMapFile.*map2*[y/***elesize***-1][x/***elesize***]==0) {

y=y-step;

}

**else** **if**(ReadMapFile.*map2*[y/***elesize***-1][x/***elesize***]!=0){

**int** y1 = (y/***elesize***-1)\****elesize***+***elesize***/2;

**if**((y-y1)\*(y-y1)>=***elesize***\****elesize***){

y=y-step;

}

}

}

}

向安全地点移动通过moveforsafe方法实现，向资源移动通过moveforresource方法实现，两部分代码相似，都是给定目标的x，y坐标，然后进行直线移动，同时设定朝向（facetoxx决定了模型图像的四种状态），下面只贴出moveforsafe方法的部分代码。代码在336到382行

//目标的x坐标大于AI的x坐标，则AI向右移动，反之则相反。

**if** (safex-x > 0){

movejudge("right");

facetoright = **true**;

}**else** **if** (safex-x < 0){

movejudge("left");

facetoleft = **true**;

}

攻击方式即决定AI的攻击方式，将AI周围区域分成八块，然后判断敌人在哪块，从而决定攻击方式和后续的移动方式，如敌人在AI东北方向，则AI将向右上斜射，同时移动方向与攻击方向垂直，通过attackdirection()方法实现。这部分代码在493到556行。下面贴出部分。

//难度为2 如果敌人在AI左边，且y坐标在上下100范围内，攻击模式为1，即向左攻击

**if** (*Difficulty*==2) {

**if** ( x > goalenemy.x && y > goalenemy.y-100 && y < goalenemy.y+100) {

attackmodel = 1;

facetoleft = **true**;

}

//难度为1 只进行左右的判断，即向左或向右射击

**else** **if** (*Difficulty*==1) {

**if** (x>goalenemy.x) {

facetoleft=**true**;

}**else** {

facetoright=**true**;

}

}

向敌人的移动方式通过moveforenemy()实现，attackmodel不同则移动方式不同，移动方向与射击方向垂直。代码在384到463行。同时设置了难度Difficuity，

低难度只会上下移动，左右射击，高难度各方向都可以。

//难度2 第五种攻击方式 敌人在AI左上角，AI需向左下或右上移动

**case** 5:

**if**(Math.*pow*((goalenemy.y-y), 2)/Math.*pow*((goalenemy.x-x), 2) <1) {

movejudge("left");

movejudge("down");

}**else** **if**(Math.*pow*((goalenemy.y-y), 2)/Math.*pow*((goalenemy.x-x), 2) >1) {

movejudge("right");

movejudge("up");

}

**break**;

//难度1 只上下移动

**else** **if** (*Difficulty*==1) {

**if** (y>goalenemy.y) {

movejudge("up");

}**else** **if**(y<goalenemy.y) {

movejudge("down");

}

}

随机移动通过moveforrandom()实现，设定一个mover，并赋予随机值，对不同的值进行不同方向的移动。代码在465到492行

//随机值在0.25到0.5,向左移动

**else** **if** (mover>0.25 & mover<=0.5) {

movejudge("left");

facetoleft = **true**;

}

#### 行动顺序

在AI行为过程中，先进行是否处于特殊情况的判断，如果处于特殊情况，则随机移动100次。

如果为否，再进行是否安全的判断，如果不安全，则moveforsafe（）。

如果安全，进行是否有敌人的判断，如果有敌人，进行射击方向的判断attackdirection()，并准备射击(**this**.buList.addAll(**this**.attack())，如果血量大于20，且有子弹，则向目标敌人移动，如果血量少于20或子弹，则进行资源判断，如果有资源，则moveforresource（）。如果没有资源，进行随机移动。

如果没有敌人，进行是否有资源的判断，如果有资源，则moveforresource（）。

如果没有资源，进行随机移动。

通过run（）实现AI完整的行为模式。代码在558到623行。

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

**while**(isalive()){

**if**(special) {

moveforrandom();

movelimit +=1;

**if** (movelimit%100 == 0) {

mover = Math.*random*();

movelimit=0;

}

specialnum++;

**if** (specialnum==100) {

special=**false**;

specialnum=0;

}

}**else** {

issafe();

**if** (!issafe) {

moveforsafe();

}**else** {

isenemy();

**if** (isenemy ) {

attackdirection();

**if** (speedlimit%20 == 0) { //限制子弹发射速度

**this**.buList.addAll(**this**.attack());

speedlimit=0;

}

speedlimit +=1;

**if** (hp>20 && **this**.bulletnum != 0) {

moveforenemy();

}**else** {

isresource();

**if** (isresource) {

moveforresource();

}**else** {

moveforrandom();

movelimit +=1;

**if** (movelimit%50 == 0) {

mover = Math.*random*();

movelimit=0;

}

}

}

}**else** {

isresource();

**if** (isresource) {

moveforresource();

}**else** {

moveforrandom();

movelimit +=1;

**if** (movelimit%50 == 0) {

mover = Math.*random*();

movelimit=0;

}

}

}

}

}

**try** {

Thread.*sleep*(10);

} **catch** (InterruptedException e) {

e.printStackTrace();

}

}

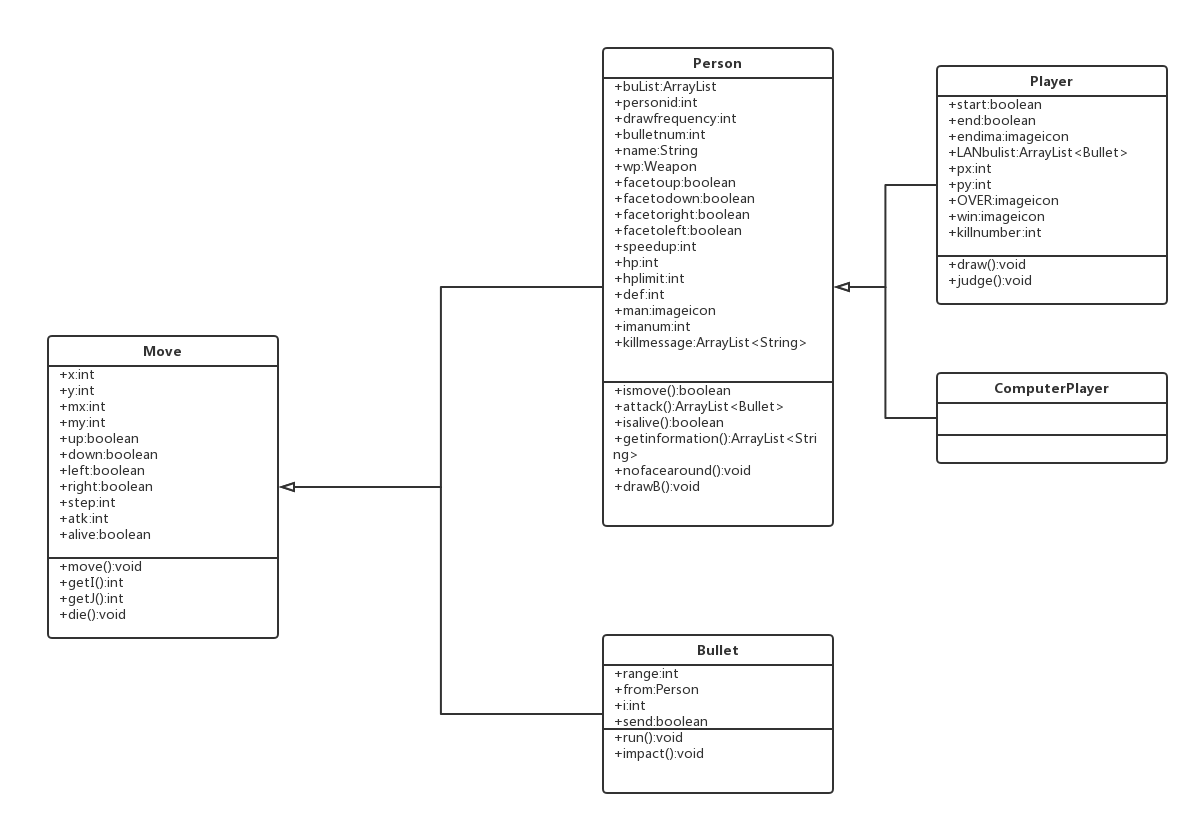
}

## （五）人物部分

### 1、基本思路

先设置一个所有可移动对象的父类Move，其下设两个子类person与bullet，分别代表人物与子弹，并设置up down left right 等boolean值来确定各对象的移动方向

### 2、具体描述



首先从Move类看起，Move类中的x、y属性代表对象在地图上的绝对位置，相当于坐标值，mx与my则是用来实现对象流畅移动的偏量位移，主要用以处理角色坐标相较于脚下地图块左上角坐标突出部分，up、down等方向boolean值则是为了判定角色移动方向，step与atk分别代表速度与攻击力，alive则是用以判断角色是否死亡、线程继续运行的变量。Move类的主要方法为move()，是子弹与玩家移动的基础，其中的部分代码如下

**if**(up){

**if**(ReadMapFile.*map2*[y/***elesize***-1][x/***elesize***]==0) {

//改变角色在地图中的位置

y=y-step;

//改变角色相对固定点的偏量位移

my=my-step;

}

**else** **if**(ReadMapFile.*map2*[y/***elesize***-1][x/***elesize***]!=0){

**int** y1 = (y/***elesize***-1)\****elesize***+***elesize***/2;

**if**((y-y1)\*(y-y1)>=***elesize***\****elesize***){

//改变角色小于角色的物体在地图中的位置

y=y-step;

//改变小于角色相对固定点的偏量位移

my=my-step;

} **else** **if**(**this**.getClass().getName()=="man.Bullet")

{

**this**.alive=**false**;

}

}

}

先通过判断方向bool值来判断运动方向，然后判断移动方向的下一格是不是墙壁（map2！=0），若是墙，人则不再前进，子弹则消失。

至于person类，则主要有以下属性

**public** **final** **int** personid;//用以区分其他玩家

**public** **int** drawfrequency;//移动时画出动作变化的频率

**public** **int** bulletnum=30;//子弹数

**public** String name;

**public** Weapon wp=**new** Weapon();//武器

**public** **boolean** facetoup=**false**;//判断角色面朝哪里

**public** **boolean** facetodown=**true**;//判断角色面朝哪里

**public** **boolean** facetoright=**false**;//判断角色面朝哪里

**public** **boolean** facetoleft=**false**;//判断角色面朝哪里

**public** **int** speedup=0;//用以标记加速鞋子的时间

**public** **int** hp;//血量

**public** **int** hplimit;

**public** **int** def;//防御力

在实例化时通过构造函数中的以下代码来随机生成角色图

**while**(man.getImage().getWidth(**null**)!=man.getImage().getHeight(**null**))

{

file=**new** File("hhhhh");

**while**(!file.exists())

{str = String.*format*("%03d", (rand.nextInt(386)+1));

file = **new** File("man/"+str+".png");}

man=**new** ImageIcon("man/"+str+".png");

man.setDescription(str);

}

角色通过attack（）函数来射出子弹，并在MainFrame中将 attck（）返回的ArrayList加入到角色的buList中，buList保存有角色所有的已发射并且尚在运动中的子弹，并在drawperson中将其画出。

**public** ArrayList<Bullet> attack()//射出子弹

{

ArrayList<Bullet> b=**new** ArrayList<Bullet>();

**if**(bulletnum>0)

{b=wp.attack(**this**);

**if**(b.size()>0)

bulletnum-=1;

}

**return** b;

}

getinformation（）则返回角色信息并在MainFrame中将之画出

最后，则是使用drawB（）函数将角色画出（若角色为玩家且不为主玩家）

**public** **void** drawB(Graphics g,Player player)//画出人物

{

Image im=man.getImage();

**if**(!ismove())

{

**if** (facetoright)

g.drawImage(im,(player.px-***elesize***/2)+(x-***elesize***/2-5-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(y-***elesize***/2-5-(player.getI())\****elesize***)-(player.my%***elesize***),(player.px-***elesize***/2)+(x+***elesize***/2+5-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(y+***elesize***/2+5-(player.getI())\****elesize***)-(player.my%***elesize***),0,im.getHeight(**null**)/2,im.getWidth(**null**)/4, 3\*im.getHeight(**null**)/4, **null**);

/**\*\*\***

**\*\*\*/**

**else** **if** (up)

g.drawImage(im,(player.px-***elesize***/2)+(x-***elesize***/2-5-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(y-***elesize***/2-5-(player.getI())\****elesize***)-(player.my%***elesize***),(player.px-***elesize***/2)+(x+***elesize***/2+5-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(y+***elesize***/2+5+5-(player.getI())\****elesize***)-(player.my%***elesize***),((imanum/drawfrequency)%4)\*im.getWidth(**null**)/4,3\*im.getHeight(**null**)/4,((imanum/drawfrequency)%4+1)\*im.getWidth(**null**)/4, im.getHeight(**null**), **null**);

/**\*\*\***

**\*\*\*/**

imanum+=1;

}

}

由于drawB()函数代码偏长，故截取其中一部分进行分析,省略中间部分代码，采用drawimage函数将图片画出，共有十个参数，其中第2-5个参数分别为图片在屏幕上的起始点与结束点坐标，第6-9个参数则为图片在原图中的起始点及结束点坐标。由于角色图片都是采用4\*4的角色移动图，故后四个参数可以相对固定，主要难点在于确定第2-5个参数的值，从前面map部分画出地图块可以得到启发，得到(player.px-***elesize***/2)+(x-***elesize***/2-5-(player.getJ())\****elesize***)-(player.mx%***elesize***)，即玩家在屏幕上的中心点加上角色坐标与玩家坐标之差值，再减去玩家偏移量，其他几个参数同理。至于移动时的图，则加上了(imanum/drawfrequency)%4+1这一变量，以此实现循环移动改变图片。

至于Player部分，首先是通过random随机生成初始坐标，Player的draw（）函数与person的drawB（）略有不同，但更为易懂，故不加以赘述。Player还有judge（）函数，以此判断游戏胜利与否，判断方式主要为player的hp值以及目前存活角色数（pe.size()，在MainFrame中存在一个ArrayList用来存放存活的角色信息）,并且在游戏结束后通过sql语句将游戏信息存入数据库

此外则是Bullet类，主要属性包括from（判断射出者，避免击中自己）、range（射程，超过射程后自动消亡），Bullet有个impact函数用以判断是否击中角色。

**public** **void** impact(ArrayList<Person> list,Player player)

{

**for**(Person i :list)

{

**if**(!i.equals(from))

{

**if**(**this**.x<=(i.x+***elesize***/2)&&(**this**.x>=(i.x-***elesize***/2))&&**this**.y<=(i.y+***elesize***/2)&&(**this**.y>=(i.y-***elesize***/2))&&i.hp>0)

{

**if**(i.personid!=player.personid&&i **instanceof** Player&&MainFrame.*LAN*)

{

{

Bullet bb=**new** Bullet(**this**.from);

bb.x=i.x;

bb.y=i.y;

bb.i=0;

bb.send=**true**;

bb.up=**this**.up;

bb.down=**this**.down;

bb.left=**this**.left;

bb.right=**this**.right;

bb.atk=**this**.atk;

((Player)i).LANbulist.add(bb);

}

}

//此处的作用是在联机时将所有子弹射击命中的判断都放在服务器端进行

**else**

{

**if**(**this**.atk-i.def>=1)

{i.hp-=**this**.atk-i.def;

}

**else**

{i.hp-=1;}

**if** (i.hp<=0) {

//播放击杀音效

i.killmessage.add(from.name+" kill "+i.name) ;

**if** (from **instanceof** Player)

{((Player) from).killnumber++;

String s;

**if**(((Player) from).killnumber<6)

{s = String.*valueOf*(((Player) from).killnumber);}

**else**

{s="6";}

**if**(MainFrame.*music*)

{

MP3 ss=**new** MP3(s+".mp3");

ss.isloop=**false**;

ss.start();

}

}

}

//击退效果实现

**if**(**this**.up)

{

**if**(ReadMapFile.*map2*[y/***elesize***-1][x/***elesize***]==0)

{

i.y-=5;

i.my-=5;

}

}

**if**(**this**.down)

{

**if**(y/***elesize***+1<40)

**if**(ReadMapFile.*map2*[y/***elesize***+1][x/***elesize***]==0)

{

i.y+=5;

i.my+=5;

}

}

**if**(**this**.right)

{

**if**(x/***elesize***+1<40)

**if**(ReadMapFile.*map2*[y/***elesize***][x/***elesize***+1]==0){

i.x+=5;

i.mx+=5;

}

}

**if**(**this**.left)

{**if**(ReadMapFile.*map2*[y/***elesize***][x/***elesize***-1]==0)

{

i.x-=5;

i.mx-=5;

}

}

}

**this**.alive=**false**;

**return**;

}

}

}

}

# （六）其他部分

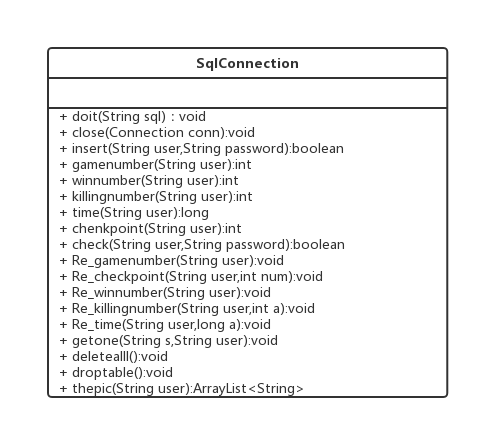
### 1、数据库

为了实现用户的注册、登录、游戏数据的存储等功能，在前期使用mysql的基础上引入sqlite（因其可转移且方便等优点）。用户的数据包括账号、密码、游戏局数、游戏总赢数（转化为游戏胜率）、总杀敌数、游戏时间以及最佳闯关成绩，同时，我们设定图鉴模式，当玩家完成游戏后（完成游戏指赢或输了游戏）可获得当场游戏的人物图鉴。

insert()用来注册用户信息，当游戏账号被注册时不得注册该账号（通过select语句搜索），返回false；当账号注册成功后（通过insert语句插入），返回true。

gamenumber()、winnunber()等方法通过select语句返回相对应的数值。

Re\_gamenumber()、Re\_winnumber()等方法通过update语句修改相对应的数值。

thepic()则用来将每次使用的玩家图片add进定义的arraylist里面，形成图鉴。

### 2、毒圈

首先是毒圈的外形，我们组认为可以将一张半透明的图片覆盖在地图的四条边上，上下两边的宽度为地图的宽度，高度为0，左右两边的宽度为0，高度为地图的高度。随着时间的增加，四张图片逐渐增加高度或宽度，最终汇聚到中心点。

半透明图片如图（黑色部分为突显白色的背景）：



然后是毒圈的判定，我们组根据人相对于毒圈的范围，判定人受到毒圈的伤害，并且添加人被毒圈毒死的信息。

代码实现：

画出毒圈：

**public** **void** draw(Graphics g,Player player)

{

g.drawImage(poison.getImage(),(player.px-***elesize***/2)+(x-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(y-(player.getI())\****elesize***)-(player.my%***elesize***), width, hlimit,**null**);

g.drawImage(poison.getImage(),(player.px-***elesize***/2)+(width-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(y-(player.getI())\****elesize***)-(player.my%***elesize***), wlimit-width\*2, height,**null**);

g.drawImage(poison.getImage(),(player.px-***elesize***/2)+(wlimit-width-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(y-(player.getI())\****elesize***)-(player.my%***elesize***), width, hlimit,**null**);

g.drawImage(poison.getImage(),(player.px-***elesize***/2)+(width-(player.getJ())\****elesize***)-(player.mx%***elesize***), (player.py-***elesize***/2)+(wlimit-height-(player.getI())\****elesize***)-(player.my%***elesize***), wlimit-width\*2, height,**null**);

}

毒圈的扩散：

**public** **void** Extensions(){

**if**(width<wlimit/2)

width+=speed;

**if**(height<hlimit/2)

height+=speed;

}

毒圈的判定：

**public** **void** isinpoison(ArrayList<Person> peList)

{

**for**(Person per:peList)

{

**if**(per.x<width||per.y<height||per.x>wlimit-width||per.y>hlimit-height)

{

per.hp-=damage;

**if** (per.hp <=0)

per.killmessage.add(per.name + " was killed by poison") ;

}

}

}

我们之所以将毒圈缩至中心点，是因为我们构造地图时在中心点绘制了更多障碍物，可以提供玩家更多的操作空间。

### 3、音乐

我们认为给程序加入音乐网上有许多教程，所以直接在网上参考了相关代码。最终我们选择了如下代码，这个代码可以添加mp3音乐，但需要导入一个jar包，这个包可以从网站上下载<http://www.javazoom.net/javalayer/sources.html>。我们添加了两首BGM，一首用于主界面，在Gui包中启动线程，另一首用于游戏界面，在MainFrame包中启动线程。

代码实现：

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

**do**

{

**try** {

BufferedInputStream buffer = **new** BufferedInputStream(

**new** FileInputStream(filename));

player = **new** Player(buffer);

player.play();

} **catch** (Exception e) {

System.***out***.println(e);

}

}**while**(isloop);

}

**public** **void** close() {

isloop=**false**;

player.close();

}

}

最开始我们选用的是如下代码：

1. **public** **class** PlayMusic {
2. **public** **static** AudioClip loadSound(String filename) {
3. URL url = **null**;
4. **try** {
5. url = **new** URL("file:" + filename);
6. }
7. **catch** (MalformedURLException e) {;}
8. **return** JApplet.newAudioClip(url);
9. }
10. **public** **void** play() {
11. AudioClip christmas = loadSound("music/bg.wav");
12. christmas.play();
13. }
14. }

但是发现这个代码只能导入wav格式的音乐，由于wav格式文件太大，而且现在主流音乐的格式是mp3，所以我们放弃了这个代码。网上搜索得知java不支持mp3音乐的导入，所以只能去网站上下载第三方jar包，才能使用。

### 4、计时器

根据网上的方法，我们得知计时器都是利用系统的时间进行计算的，我们只需将游戏界面开始的时间赋予一个变量中，启动线程后便会实时显示时间，同时还需要有一个调整时间格式的方法。我们在MainFrame包中启动了线程。

代码实现：

计时：

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

{

// 记录程序开始时间

**long** programStart = System.*currentTimeMillis*();

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

**if** (!stopped) {

**long** elapsed = System.*currentTimeMillis*() - programStart;

time=elapsed;

}

**try** {

*sleep*(1); // 1毫秒更新一次显示

} **catch** (InterruptedException e) {

e.printStackTrace();

System.*exit*(1);

}

}

}

将时间格式化：

**public** **static** String format(**long** elapsed)

{

**int** minute, second;

elapsed=elapsed / 1000;

second = (**int**) (elapsed % 60);

elapsed = elapsed / 60;

minute = (**int**) (elapsed % 60);

elapsed = elapsed / 60;

**return** String.*format*("%02d:%02d",minute, second);

}

# 五、程序评价

## （一）优势与特色

本游戏包括多种游戏模式、多种难度、多种地图，实现了实时射击、采集资源，并且可以与数据库连接获取玩家信息。具有大部分小型单机游戏的主要功能，并且游戏性较强，主界面完善

## （二）缺点与不足

游戏的联机模式尚不完善，无法在同一局域网下两台电脑间稳定传输。此外，程序解耦不足，各个类之间往往相互调用属性，导致代码的扩展性可能相对较差。

# 六、心得体会

此次Java大作业使我们的Java程序设计应用水平得到提升，因为大作业所需要设计到的知识面相对较广，包括数据库、多线程、GUI、网络传输、多态与继承等多方面知识，所以在编程过程中难免常常遇到问题，这时候善用网络资源就成了一项必备技能，通过查询阅读百度知道、CSDN、StackOverflow、segmentfault、ITeye、知乎等众多站点，使用百度、谷歌等搜索引擎来解决自己的问题，是在实现本程序的过程中的常态。对于编程知识，了解得越多，便会觉得自己所知道的越少。不仅如此，有时不够细致认真，常常导致进展缓慢，漏洞百出。有时候对着一个BUG调试上半天求解不得，却在睡前闲想的时候灵光一闪获得思路，这其中的激动与喜悦，或许也是程序设计的乐趣所在。

# 七、参考资料

·地图与人物的一些基础部分参考了该博客的代码

<http://y-1746119035.iteye.com/blog/2147785>

·部分玩法及设定参考《绝地求生·大逃杀》、《僵尸危机》、《口袋妖怪》等游戏

·人物图片来源<http://www.aigei.com/view/273.html>

·其余参考项多为csdn等技术网站上的代码，由于零散繁多，故不一一列出