**Java大作业文档**

**大鱼吃小鱼**

11061077 邵帅

**目录**

[前言 2](#_Toc343702690)

[1. 概述 2](#_Toc343702691)

[1.1. 简介 2](#_Toc343702692)

[1.2. 功能描述 2](#_Toc343702693)

[1.2.1. 基本游戏功能： 2](#_Toc343702694)

[1.2.2. 界面友好等人性化的要求： 3](#_Toc343702695)

[1.2.3. 高级游戏功能： 4](#_Toc343702696)

[1.2.4. 附加功能： 4](#_Toc343702697)

[2. 技术细节 5](#_Toc343702698)

[2.1. 代码结构 5](#_Toc343702699)

[2.1.1. 代码文件 5](#_Toc343702700)

[2.1.2. 代码类及其中的重要方法 5](#_Toc343702701)

[2.2. 主要代码及解析 8](#_Toc343702702)

[2.2.1. MainFrame中的监听器 8](#_Toc343702703)

[2.2.2. MainFrame中的方法toScene() 9](#_Toc343702704)

[2.2.3. player中本体鱼图像的获取 10](#_Toc343702705)

[2.2.4. player中经验增加的方法addExp() 10](#_Toc343702706)

[2.2.5. npc中的方法initial() 12](#_Toc343702707)

[2.2.6. npc中change()内对npc鱼位置的更新 13](#_Toc343702708)

[2.2.7. npc中change()内对碰撞的判定 14](#_Toc343702709)

[2.2.8. scene中的键盘监听器 15](#_Toc343702710)

[2.2.9. npc中线程的启动，结束和运行 15](#_Toc343702711)

[2.2.10.npc中重写update()防止闪烁 16](#_Toc343702712)

[2.2.11.npc中重写paint()内在缓冲图片上绘图的过程 16](#_Toc343702713)

[2.2.12.npc重写paint()内把缓冲图片绘制到屏幕上的过程 17](#_Toc343702714)

[2.2.13.PausePanel和EndPanel中显示最后的等级经验 18](#_Toc343702715)

[3. 测试效果 19](#_Toc343702716)

[3.1. 开始菜单 19](#_Toc343702717)

[3.2. 场景一游戏显示和操作 20](#_Toc343702718)

[3.3. 游戏暂停界面 21](#_Toc343702719)

[3.4. 游戏场景切换 22](#_Toc343702720)

[3.5. 游戏结束菜单 23](#_Toc343702721)

# 前言

经过好几周的努力，终于完成我学习java以来第一个较为大型的工程。这次提交的文件除本文档外有一个jar文件和一个exe文件，都是我打包好的程序，可以直接运行，运行后直接点开始游戏进入场景一，然后就能开始通过键盘的上下左右控制鱼来避开比自己大的鱼，并且吃比自己小的鱼，吃掉比机子小的鱼就能增加经验，积累一定的经验就可以升级改变外形。同时还可以通过传送门到达别的场景。除此之外还有一个装源代码文件的文件夹src，里面除了多个代码文件外还有一个装图片素材的image文件夹，如果要在eclipse调试状态下运行此游戏，只需要在新建工程后把代码文件都添加到工程当中，然后把image文件夹放到工程生成的bin文件夹中(即二进制文件存放目录)，否则将无法正常显示图片。

# 概述

## 简介

使用java平台，设计完成游戏大鱼吃小鱼，并进行多方面的改进和功能的增加。

## 功能描述

### 基本游戏功能：

(1) 完成海底场景的实现和各种鱼类的外观设计。（通过互联网寻找原始图片并通过photoshop进行修改）。

(2) 通过键盘上下左右键控制本体鱼的运动，并伴随着屏幕显示背景的移动。

(3) 实现其他鱼类的运动方式的设定，维持其运动方向的随机性，但同时不会频繁地转向导致运动范围过小。

(4) 其他鱼类的大小设定要合理，不能使游戏难度太小或很难继续进行游戏。

(5) 完成大鱼吃小鱼的设定，即本体碰到小鱼能吃掉，并同时刷新出另一条鱼；若碰到比本体大的鱼则游戏结束，重点是鱼和鱼的碰撞判定。

### 界面友好等人性化的要求：

(1) 开始时有开始菜单，游戏中能暂停并出现暂停菜单，游戏结束有结束菜单。

(2) 其他鱼进行初始化或刷新时不能直接撞上本体鱼导致游戏直接结束，但是其他鱼的初始位置必须遵守随机性。

(3) 所有鱼都不能出现游到地图外的情况。

(4) 无论本体鱼在地图上如何运动，基本保持本体在屏幕中央，除非移动到边缘，那么也要保持本体在中间位置。

(5) 游戏中背景和开始，暂停，结束菜单的图片应美观合理。

### 高级游戏功能：

(1) 具有等级和经验功能，及吃小鱼会增加经验，而得到一定的经验就会升级。

(2) 伴随着经验的增加，本体鱼会增大。每一级的鱼的尺寸根据经验成线性增长。

(3) 当等级提升时，鱼的外观会发生改变。每一级对应一个外观。

(4) 其他鱼的尺寸和外观也同样由经验和等级决定，吃下鱼所增长的经验完全由该鱼的等级和经验决定，并且吃比自己等级低的鱼获得的经验将大大减少从而鼓励尽量吃较大的鱼。

(5) 设定多个游戏场景，每个场景里的鱼的等级分布不一样，即后面场景里的鱼普遍较大。

(6) 每个场景之间都有传送门，当本体鱼进入传送门后将自动传送到该传送门对应的场景。

(7) 每次进入新的场景鱼都会全部重新刷新，因此可以通过进入别的场景马上再回来从而解决原本场景鱼太大的问题。

(8) 每个场景采用单独的线程控制，每次场景切换时结束上一线程并开始下一线程。

(9) 画面顶端要显示本体鱼当前的等级和经验。其中等级用数字显示，经验用进度条和数字同时显示。

### 附加功能：

(1) 增加秘籍功能，即通过输入秘籍可以让本体鱼进入无敌状态，再次输入则退出无敌状态。（ps：本游戏为“s”键，因为我的姓名开头字母就是ss (\*^\_\_^\*) 嘻嘻……）

# 技术细节

## 代码结构

### 代码文件

MainGame.java 即工程入口

MainFrame.java 即主框架文件

StartPanel.java 即开始界面文件

PausePanel.java 即暂停界面文件

EndPanel.java 即结束界面文件

player.java 即本体鱼的状态文件

3个scene.java文件即场景文件

3个npc.java文件即其他鱼的状态文件

Image文件夹中的多张鱼类，场景背景，及菜单背景的jpg或gif图片

### 代码类及其中的重要方法

文件MainGame.java只有一个类，只是把框架实例化放在main方法中。

文件MainFrame.java有一个继承了JFrame的类，作为该游戏的主框架，其他组件都放在该类中，这个框架不能改变大小，并且没有使用布局管理器，所有组件采用硬性坐标布局。除此之外，该文件中还有几个内部类，都是开始、暂停、结束3个菜单中按钮的监听器，为了便于管理都放在了主框架中。还有几个场景切换的方法，内容是终止当前线程，隐藏当前场景的JPanel，重置本体鱼位置到下一场景的传送门旁边，再new一个新场景的JPanel，并将其添加到主框架中，同时启动其线程。当场景里检测到本体进入传送门时即可调用此方法来实现场景切换。

文件StartPanel.java，PausePanel.java，EndPanel.java这三个菜单的文件都有一个继承JPanel的类，在当中添加“开始游戏”、“结束游戏”、“继续游戏”，“重新开始”等按钮，其监听器已经放在MainFrame当中。此外，通过改写JPanel的paintComponent()方法，在其中调用drawImage()来把背景图片画到菜单中。

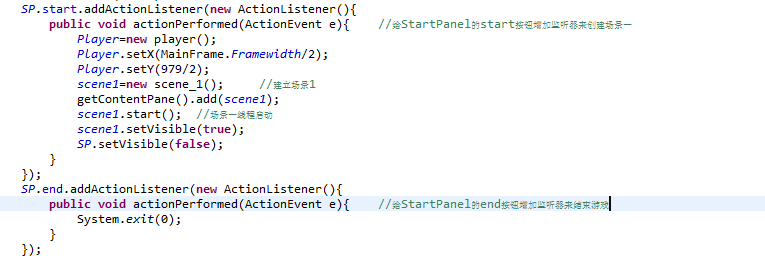
文件player.java中只有player一个类用来储存本体鱼的状态资料等。构造方法主要是初始化本体鱼的等级、经验、方向、尺寸、外形图片以及创建一个Image格式的数组并把4个等级的鱼的两个朝向的外形图片都放到该数组中。其中外形图片是根据等级经验对相应原始图片调用getScaledInstance(wid, hei, Image.SCALE\_SMOOTH)方法来进行缩放到相应尺寸。该类中有大量方法，主要都是用于让其他类来调用这些方法从而改变本体鱼状态的，或者是其他类调用这些方法来获取本体鱼状态的。其中值得注意的有setX、setY中一定要加上判断本体是否超出边界，若超出则强行改回边界内；addexp（增加经验）一定要判断是不是升级，如果升级则改变尺寸、等级、经验、外形等等；setdir（设置方向）改变方向变量同时改变图片。

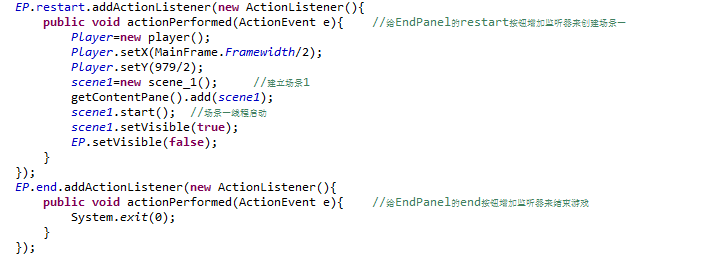
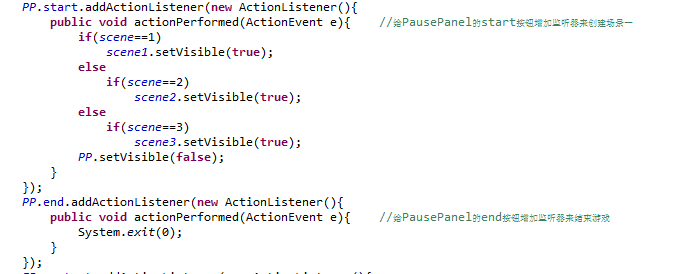
文件npc?.java(?表示1,2,3)都有一个储存npc鱼（ps：我实在想不到那些应该叫野怪的鱼应该取什么名字，结果当时就取了一个“npc”，现在才觉得不合理，但是改名字实在太麻烦了，所以就这样用算了）数据的类。该类的构造方法就是把所有的npc鱼都初始化，即调用下面的initial(int i)方法对第i条鱼进行初始化。初始化的具体内容是大量使用Math.random()来对npc鱼的位置、等级、经验、方向等进行初始化，同时根据等级经验生成相应的外形图片。其中应该注意的是初始化位置时一定要离本体鱼一定距离以外，以及设定好每个场景里每个等级鱼所占的比例（完成后经过很多测试才基本达到游戏的平衡性）。除此之外还有一个change()方法，作用是每次重画场景前都调用此方法来改变npc鱼的状态。内容是鱼一直向原来方向游动的同时有1/200的概率会改变方向，此方向包括水平的和垂直的，若撞到边缘则分别有一般的概率贴边走和向反方向走。另外和本体鱼的碰撞也放在这个方法里面，每次改变坐标后都和本体鱼进行碰撞检测（ps：由于只是采用了近似矩形的判断方法，所以在玩的时候难免会遇到没碰到鱼却死了的情况，对此我只能表示无能为力），发生碰撞后若比本体鱼大，则线程结束，弹出EndPanel；若比本体鱼小，则根据等级经验对本体鱼调用addexp()方法增加经验。

文件scene\_?.java(?表示1,2,3)都有一个extends JPanel且implements Runable的类，用于表示每个场景的JPanel。该类添加了键盘的监听器，监听上下左右以及的press和release以及秘籍的press。上下左右各对应一个boolean变量表示本体鱼的运动状态。该类的重点在于方法线程的开始、结束、运行，重写update()方法防止画面闪烁以及最重要的重写piant()方法来完成该场景中所有图像的绘制。（游戏的运行主要就在于线程不停地运行，每sleep(30)就执行一次paint()方法，从而让画面不断刷新。）在paint()里内容主要有：判断本体鱼是不是到达了该场景的传送门，如果到了就调用MainFrame里的方法；利用双重缓冲技术在巨大的缓冲图上绘制大型场景；依次绘制背景，本体鱼，所有npc鱼，以及画面顶端的等级经验显示；最重要的是根据本体鱼在场景中的位置（场景是一个很大的图，远大于框架）把合适的画面显示在JPanel中，具体方法是大部分情况让本体鱼在正中央，除非本体鱼到达场景边缘附近无法让本体鱼在正中央时则尽量让本体鱼靠近中间。

## 主要代码及解析

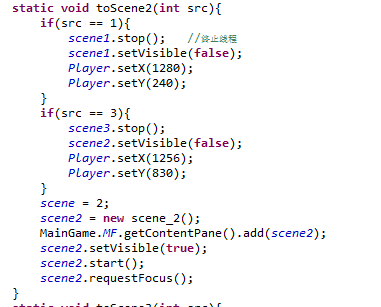
### MainFrame中的监听器





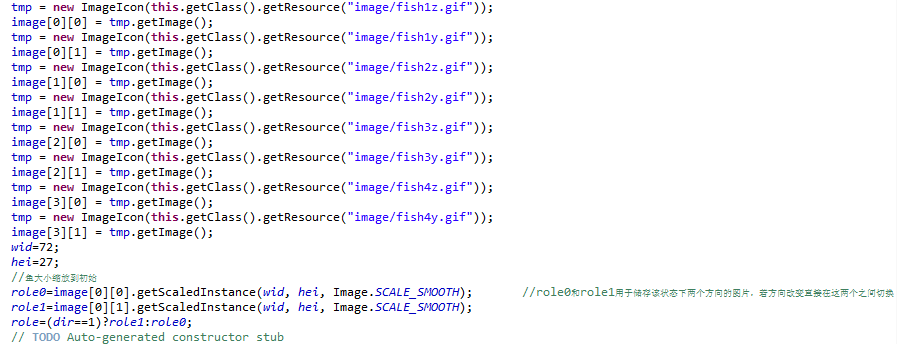
这是开始、暂停、结束三个菜单中按钮的监听器，主要内容都是第一个按钮隐藏菜单界面，显示游戏场景，第二个按钮结束游戏。其中在SP（开始菜单）和EP（结束菜单）中，第一个按钮都是开始游戏，内容还有new一个新的player（重置player的状态），new一个场景一，同时把场景一add到MainFrame中，并调用start()启动场景一的线程。

### MainFrame中的方法toScene()



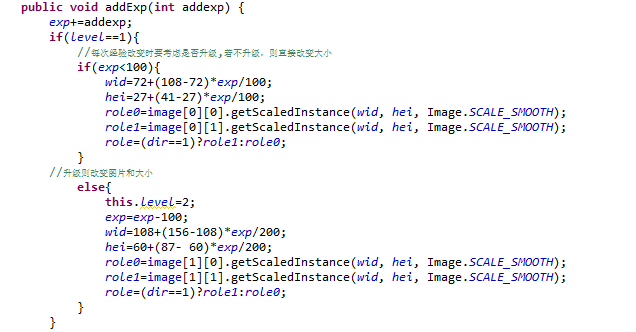
toScene1(int i),toScene2(int i),toScene3(int i)是场景切换的方法，其中参数是当前场景。主要内容就是stop()终止线程，setX，setY重置位置， new scene新建场景，add添加场景，start()启动新的线程，requestFocus()获取焦点从而使新场景的监听器生效。

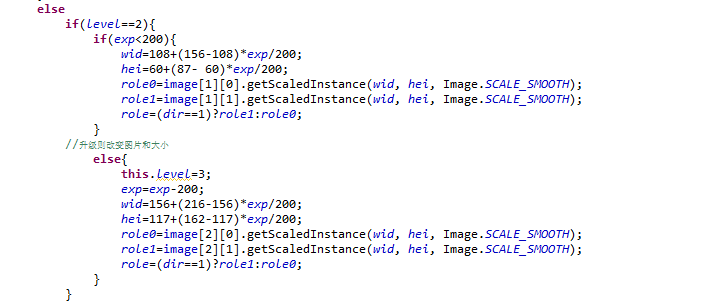
### player中本体鱼图像的获取

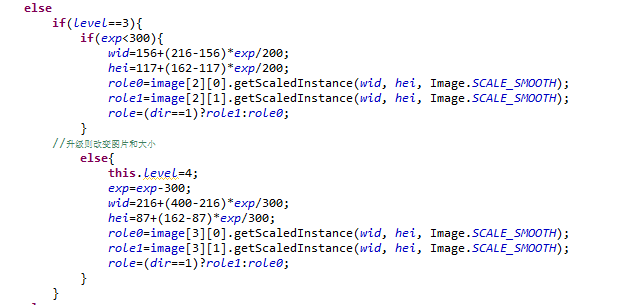


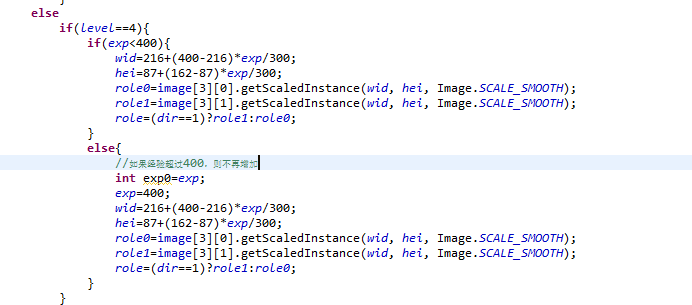
从该类的class文件同目录的image文件夹中读取相应图片到数组中。其中image[i][j]表示等级为i+1时方向为j的图片（j=0表示向左，j=1表示向右）。role为当前的本体鱼的图像。此外，role0和role1为两个Image文件，表示对应于当前尺寸的朝向不同的两个本体鱼的图片，目的是防止每次改变方向都重新进行图片放缩使得游戏性能下降。

### player中经验增加的方法addExp()



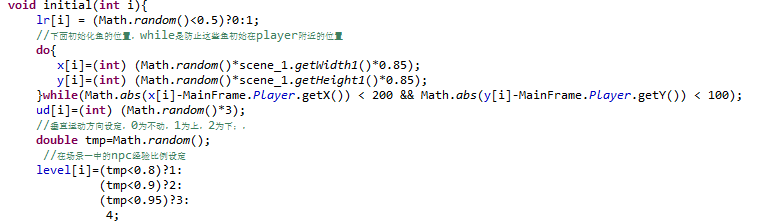






增加经验时，先进行等级的判定，然后再改变本体鱼的尺寸，具体方法是在设计游戏时就设计好每个等级的尺寸增长设定，即每级开始和结束时长宽，然后根据每级升级所需经验（第i级为i\*100）和当前经验线性确定尺寸，然后在每次改变尺寸时都根据当前尺寸刷新role0和role1，之后只要根据方向变量dir的值用role0和role1对role赋值就可。

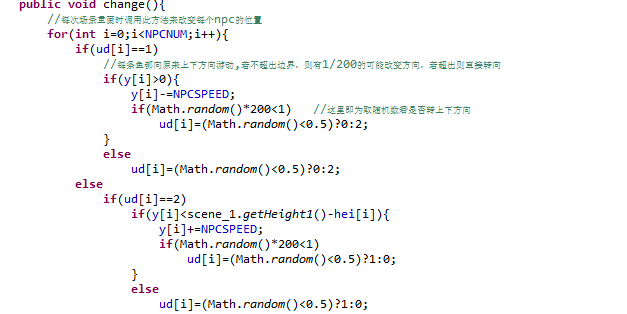
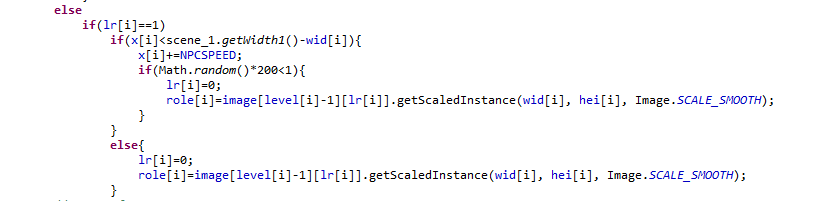
### npc中的方法initial()





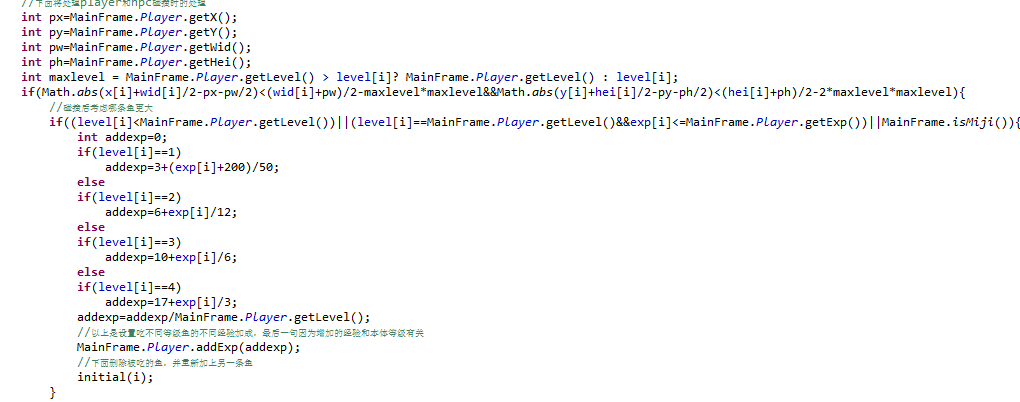
即通过Math.random()所在的范围的特殊运用来初始化npc鱼的lr（水平方向），ud（垂直方向），exp（经验），level（等级），wid（宽）,hei（高），并根据相应等级方向尺寸确定外形role[i]。值得注意的是第一个do while语句目的是使得所有npc鱼的位置初始化时都不能再本体鱼附近。

### npc中change()内对npc鱼位置的更新

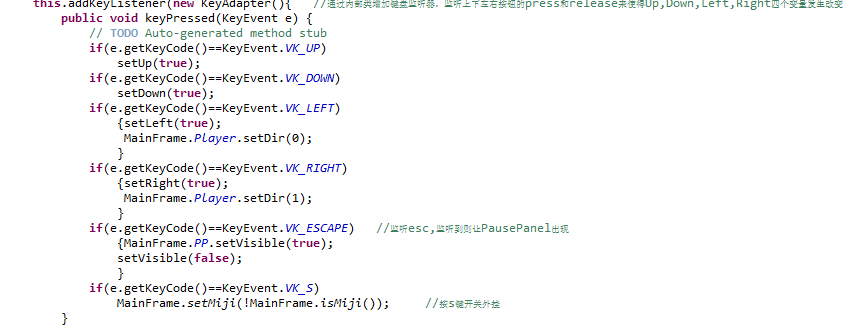
主要就是根据当前方向对x，y进行加减NPCSPEED常量的值，并判断Math.random()\*200是否小于1，若小于则改变方向，同时判断是否到达边界，若到达有一半概率回头或贴边移动。

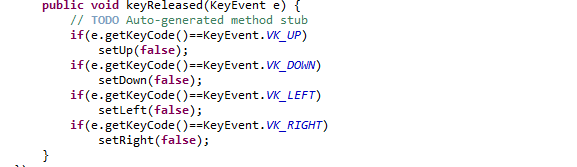
### npc中change()内对碰撞的判定

定义px，py为本体鱼的中央位置，然后maxlevel为本体鱼和发生碰撞的npc的等级较高的一个，然后通过两鱼中心坐标的abs（绝对值）进行修正后和长宽比较。其中修正的内容为加上一个maxlevel乘上一个常数，目的就是因为鱼不是矩形，而越大的鱼周围的空白越多，因此利用这个和等级挂钩的方法可以减少出错的概率。另外还或了Miji变量，就是秘籍开启的话直接把鱼吃掉。

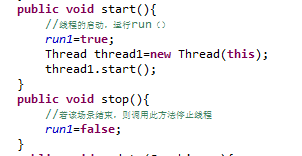
### scene中的键盘监听器





主要内容就是检测到press就对相应向量变为true，检测到release就对应向量变为false。此外“s”键作为本游戏秘籍的开关，一旦开启就会进入无敌状态，即Miji变量取反。

### npc中线程的启动，结束和运行

Start方法就是创建一个新线程，启动，并且让一个boolean变量run1令其为true。

Stop就是让run1变为false。

Run就是当run1为true时不停循环更新坐标并通过repaint调用paint（）重画。

### npc中重写update()防止闪烁

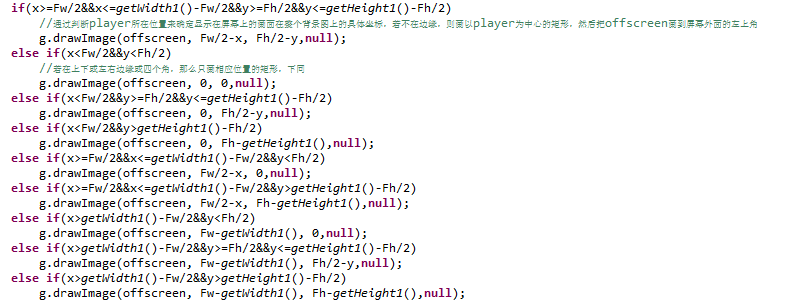


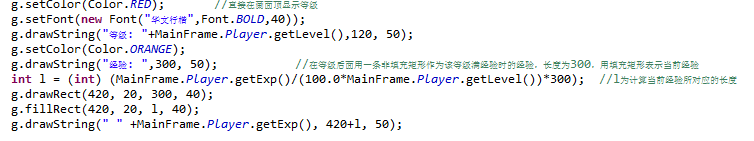
### npc中重写paint()内在缓冲图片上绘图的过程



先判断本体鱼坐标是不是在该场景的传送门内，若在就调用MainFrame中的toscene()方法。然后createImage创建一个和背景图片一样大小的缓冲图片，然后利用g1在该缓冲图片上画出背景，player和npc。

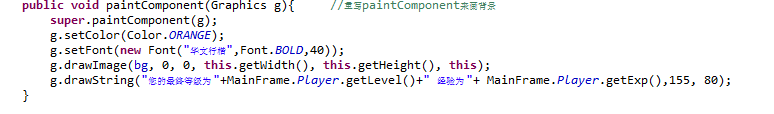
### npc重写paint()内把缓冲图片绘制到屏幕上的过程





把整个场景分为9部分，中间一部分是可以成功地让本体鱼在中央的情况，那么此时就把缓冲图片的坐标设为Fw/2-x,Fh/2-y，即放到屏幕左上角的外面，此时恰好本体鱼会在屏幕中间。此外当本体不能显示在中央的情况有周围八个区域，每个都根据特定的情况计算好然后经过调试最后成功。此外，最后再用华文行楷的红字显示等级再用橙色的显示经验，并且用空矩形画出满的经验条，再用填充矩形画出当前经验条。

### PausePanel和EndPanel中显示最后的等级经验

和StartPanel一样借助改写paintComponent()画出背景，并且用橙色笔写出等级和经验。

# 测试效果

## 开始菜单



图片，文字和按钮成功显示，并且点击效果正确。

## 场景一游戏显示和操作



正确显示背景，本体，npc以及等级经验的显示，并且吃小鱼后能正确增加经验和等级。

## 游戏暂停界面

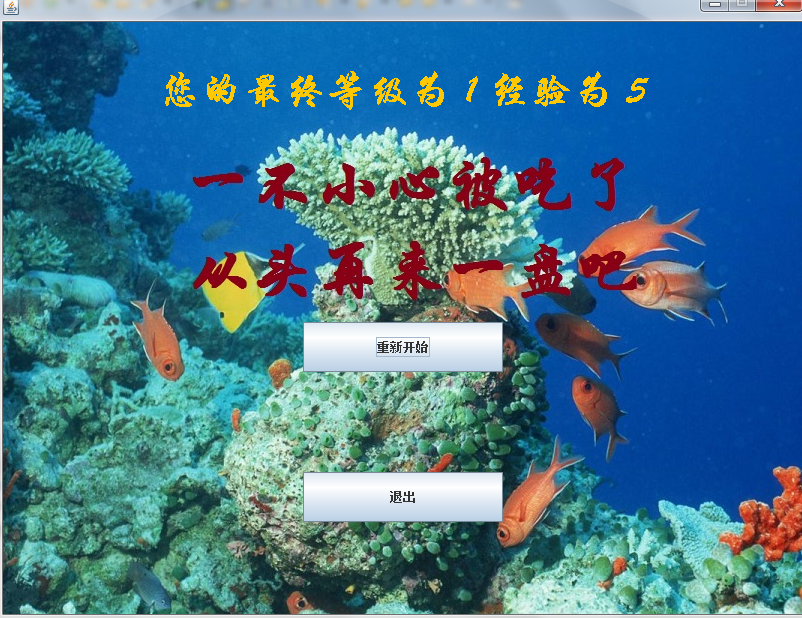


按esc键能成功进入暂停界面，且暂停界面能正确显示等级经验按钮，按钮也能起作用。

## 游戏场景切换

 能成功通过传送门进入场景二和三。

## 游戏结束菜单



碰撞后能成功显示结束界面并且显示正确，按键有效。