Java 大作业文档

大鱼吃小鱼

11061077　　邵帅

目录

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。

1. 概述

1.1. 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。

1.2. 功能描述

1.2.1. 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。

1.2.2. 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。

1.2.3. 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。

1.2.4. 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……）

2. 技术细节

2.1. 代码结构

2.1.1. 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

2.1.2. 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。

2.2. 主要代码及解析

2.2.1. MainFrame 中的监听器

```
SP.start.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){    //给StartPanel的start按钮增加监听器来创建场景一
        Player=new player();
        Player.setX(MainFrame.Framewidth/2);
        Player.setY(979/2);
        scene1=new scene_1();     //建立场景1
        getContentPane().add(scene1);
        scene1.start();  //场景一线程启动
        scene1.setVisible(true);
        SP.setVisible(false);
    }
});
SP.end.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){    //给StartPanel的end按钮增加监听器来结束游戏
        System.exit(0);
    }
});
PP.start.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){    //给PausePanel的start按钮增加监听器来创建场景一
        if(scene==1)
            scene1.setVisible(true);
        else
            if(scene==2)
                scene2.setVisible(true);
        else
            if(scene==3)
                scene3.setVisible(true);
        PP.setVisible(false);
    }
});
PP.end.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){    //给PausePanel的end按钮增加监听器来结束游戏
        System.exit(0);
    }
});
```

```
EP.restart.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){    //给EndPanel的restart按钮增加监听器来创建场景一
        Player=new player();
        Player.setX(MainFrame.Framewidth/2);
        Player.setY(979/2);
        scene1=new scene_1();     //建立场景1
        getContentPane().add(scene1);
        scene1.start();  //场景一线程启动
        scene1.setVisible(true);
        EP.setVisible(false);
    }
});
EP.end.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){    //给EndPanel的end按钮增加监听器来结束游戏
        System.exit(0);
    }
});
```

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。

2.2.2. MainFrame 中的方法 toScene()

```
static void toScene2(int src){
    if(src == 1){
        scene1.stop();    //终止线程
        scene1.setVisible(false);
        Player.setX(1280);
        Player.setY(240);
    }
    if(src == 3){
        scene3.stop();
        scene2.setVisible(false);
        Player.setX(1256);
        Player.setY(830);
    }
    scene = 2;
    scene2 = new scene_2();
    MainGame.MF.getContentPane().add(scene2);
    scene2.setVisible(true);
    scene2.start();
    scene2.requestFocus();
}
```

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。

2.2.3. player 中本体鱼图像的获取

```
tmp = new ImageIcon(this.getClass().getResource("image/fish1z.gif"));
image[0][0] = tmp.getImage();
tmp = new ImageIcon(this.getClass().getResource("image/fish1y.gif"));
image[0][1] = tmp.getImage();
tmp = new ImageIcon(this.getClass().getResource("image/fish2z.gif"));
image[1][0] = tmp.getImage();
tmp = new ImageIcon(this.getClass().getResource("image/fish2y.gif"));
image[1][1] = tmp.getImage();
tmp = new ImageIcon(this.getClass().getResource("image/fish3z.gif"));
image[2][0] = tmp.getImage();
tmp = new ImageIcon(this.getClass().getResource("image/fish3y.gif"));
image[2][1] = tmp.getImage();
tmp = new ImageIcon(this.getClass().getResource("image/fish4z.gif"));
image[3][0] = tmp.getImage();
tmp = new ImageIcon(this.getClass().getResource("image/fish4y.gif"));
image[3][1] = tmp.getImage();
wid=72;
hei=27;
//鱼大小缩放到初始
role0=image[0][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);       //role0和role1用于储存该状态下两个方向的图片，若方向改变直接在这两个
role1=image[0][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
role=(dir==1)?role1:role0;
// TODO Auto-generated constructor stub
```

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。

2.2.4. player 中经验增加的方法 addExp()

```
public void addExp(int addexp) {
    exp+=addexp;
    if(level==1){
        //每次经验改变时要考虑是否升级，若不升级，则直接改变大小
        if(exp<100){
            wid=72+(108-72)*exp/100;
            hei=27+(41-27)*exp/100;
            role0=image[0][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role1=image[0][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role=(dir==1)?role1:role0;
        }
    //升级则改变图片和大小
        else{
            this.level=2;
            exp=exp-100;
            wid=108+(156-108)*exp/200;
            hei=60+(87- 60)*exp/200;
            role0=image[1][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role1=image[1][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role=(dir==1)?role1:role0;
        }
    }
```

```
        else
            if(level==2){
                if(exp<200){
                    wid=108+(156-108)*exp/200;
                    hei=60+(87- 60)*exp/200;
                    role0=image[1][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role1=image[1][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role=(dir==1)?role1:role0;
                }
                //升级则改变图片和大小
                else{
                    this.level=3;
                    exp=exp-200;
                    wid=156+(216-156)*exp/200;
                    hei=117+(162-117)*exp/200;
                    role0=image[2][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role1=image[2][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role=(dir==1)?role1:role0;
                }
            }
        else
            if(level==3){
                if(exp<300){
                    wid=156+(216-156)*exp/200;
                    hei=117+(162-117)*exp/200;
                    role0=image[2][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role1=image[2][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role=(dir==1)?role1:role0;
                }
                //升级则改变图片和大小
                else{
                    this.level=4;
                    exp=exp-300;
                    wid=216+(400-216)*exp/300;
                    hei=87+(162-87)*exp/300;
                    role0=image[3][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role1=image[3][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
                    role=(dir==1)?role1:role0;
                }
            }
    }
else
    if(level==4){
        if(exp<400){
            wid=216+(400-216)*exp/300;
            hei=87+(162-87)*exp/300;
            role0=image[3][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role1=image[3][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role=(dir==1)?role1:role0;
        }
        else{
            //如果经验超过400，则不再增加
            int exp0=exp;
            exp=400;
            wid=216+(400-216)*exp/300;
            hei=87+(162-87)*exp/300;
            role0=image[3][0].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role1=image[3][1].getScaledInstance(wid, hei, Image.SCALE_SMOOTH);
            role=(dir==1)?role1:role0;
        }
    }
```

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。

2.2.5. npc 中的方法 initial()

```
void initial(int i){
    lr[i] = (Math.random()<0.5)?0:1;
    //下面初始化鱼的位置，while是防止这些鱼初始在player附近的位置
    do{
       x[i]=(int) (Math.random()*scene_1.getWidth1()*0.85);
       y[i]=(int) (Math.random()*scene_1.getHeight1()*0.85);
    }while(Math.abs(x[i]-MainFrame.Player.getX()) < 200 && Math.abs(y[i]-MainFrame.Player.getY()) < 100);
    ud[i]=(int) (Math.random()*3);
    //垂直运动方向设定，0为不动，1为上，2为下；。
    double tmp=Math.random();
     //在场景一中的npc经验比例设定
    level[i]=(tmp<0.8)?1:
             (tmp<0.9)?2:
             (tmp<0.95)?3:
              4;

   if(level[i]==1){
       exp[i]=(int) (-150+250*Math.random());
        //等级为1的npc尺寸根据经验的设定，注意1级时经验不能从0开始考虑，否则游戏开始时一条鱼也吃不了。
       wid[i]=72+(108-72)*exp[i]/100;
       hei[i]=27+(41-27)*exp[i]/100;
       role[i]=image[0][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);
   }
   else
       if(level[i]==2){
           exp[i]=(int) (200*Math.random());
           wid[i]=108+(156-108)*exp[i]/200;
           hei[i]=60+(87-60)*exp[i]/200;
           role[i]=image[1][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);;
       }
   else
       if(level[i]==3){
           exp[i]=(int) (300*Math.random());
           wid[i]=156+(216-156)*exp[i]/300;
           hei[i]=117+(162-117)*exp[i]/300;
           role[i]=image[2][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);;
       }
   else
       if(level[i]==4){
           exp[i]=(int) (400*Math.random());
           wid[i]=216+(400-216)*exp[i]/400;
           hei[i]=87+(162-87)*exp[i]/400;
           role[i]=image[3][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);
```

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。

2.2.6. npc 中 change()内对 npc 鱼位置的更新

```
public void change(){
    //每次场景重画时调用此方法来改变每个npc的位置
    for(int i=0;i<NPCNUM;i++){
        if(ud[i]==1)
            //每条鱼都向原来上下方向游动,若不超出边界，则有1/200的可能改变方向，若超出则直接转向
            if(y[i]>0){
                y[i]-=NPCSPEED;
                if(Math.random()*200<1)    //这里即为取随机数看是否转上下方向
                    ud[i]=(Math.random()<0.5)?0:2;
            }
            else
                ud[i]=(Math.random()<0.5)?0:2;
        else
            if(ud[i]==2)
                if(y[i]<scene_1.getHeight1()-hei[i]){
                    y[i]+=NPCSPEED;
                    if(Math.random()*200<1)
                        ud[i]=(Math.random()<0.5)?1:0;
                }
                else
                    ud[i]=(Math.random()<0.5)?1:0;

     else
         if(ud[i]==0&&Math.random()*200<1){
             //若原来状态为水平，则改变方向前必须考虑改变方向后是否会超出边界
             if(y[i]>0&&y[i]<scene_1.getHeight1()-hei[i]){
                     ud[i]=(Math.random()<0.5)?1:2;
             }
             else
                 if(y[i]>0)
                         ud[i]=2;
                 else
                     if(y[i]<scene_1.getHeight1()-hei[i])
                             ud[i]=1;
         }
  if(lr[i]==0)
      //左右方向的改变同上，只是改变方向的话就必须同时改变鱼的图
      if(x[i]>0){
          x[i]-=NPCSPEED;
          if(Math.random()*200<1){
              lr[i]=1;
              role[i]=image[level[i]-1][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);
          }
      }
      else{
          lr[i]=1;
          role[i]=image[level[i]-1][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);
      }



else
    if(lr[i]==1)
        if(x[i]<scene_1.getWidth1()-wid[i]){
            x[i]+=NPCSPEED;
            if(Math.random()*200<1){
                lr[i]=0;
                role[i]=image[level[i]-1][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);
            }
        }
        else{
            lr[i]=0;
            role[i]=image[level[i]-1][lr[i]].getScaledInstance(wid[i], hei[i], Image.SCALE_SMOOTH);
        }
```

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。

2.2.7. npc 中 change()内对碰撞的判定

```
//下面将处理player和npc碰撞时的处理
int px=MainFrame.Player.getX();
int py=MainFrame.Player.getY();
int pw=MainFrame.Player.getWid();
int ph=MainFrame.Player.getHei();
int maxlevel = MainFrame.Player.getLevel() > level[i]? MainFrame.Player.getLevel() : level[i];
if(Math.abs(x[i]+wid[i]/2-px-pw/2)<(wid[i]+pw)/2-maxlevel*maxlevel&&Math.abs(y[i]+hei[i]/2-py-ph/2)<(hei[i]+ph)/2-2*maxlevel*maxlevel){
    //碰撞后考虑哪条鱼更大
    if((level[i]<MainFrame.Player.getLevel())||(level[i]==MainFrame.Player.getLevel()&&exp[i]<=MainFrame.Player.getExp())||MainFrame.isMiji()){
        int addexp=0;
        if(level[i]==1)
            addexp=3+(exp[i]+200)/50;
        else
        if(level[i]==2)
            addexp=6+exp[i]/12;
        else
        if(level[i]==3)
            addexp=10+exp[i]/6;
        else
        if(level[i]==4)
            addexp=17+exp[i]/3;
        addexp=addexp/MainFrame.Player.getLevel();
        //以上是设置吃不同等级鱼的不同经验加成，最后一句因为增加的经验和本体等级有关
        MainFrame.Player.addExp(addexp);
        //下面删除被吃的鱼，并重新加上另一条鱼
        initial(i);
    }
```

```
        //下面考虑player被吃掉
        else{
            MainFrame.scene1.stop();
            MainFrame.EP.setVisible(true);
            MainFrame.scene1.setVisible(false);
        }
    }
```

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。

2.2.8. scene 中的键盘监听器

```
this.addKeyListener(new KeyAdapter(){   //通过内部类增加键盘监听器，监听上下左右按钮的press和release来使得Up,Down,Left,Right四个变量发生改变
    public void keyPressed(KeyEvent e) {
        // TODO Auto-generated method stub
        if(e.getKeyCode()==KeyEvent.VK_UP)
            setUp(true);
        if(e.getKeyCode()==KeyEvent.VK_DOWN)
            setDown(true);
        if(e.getKeyCode()==KeyEvent.VK_LEFT)
            {setLeft(true);
             MainFrame.Player.setDir(0);
            }
        if(e.getKeyCode()==KeyEvent.VK_RIGHT)
            {setRight(true);
             MainFrame.Player.setDir(1);
            }
        if(e.getKeyCode()==KeyEvent.VK_ESCAPE)   //监听esc,监听到则让PausePanel出现
            {MainFrame.PP.setVisible(true);
            setVisible(false);
            }
        if(e.getKeyCode()==KeyEvent.VK_S)
            MainFrame.setMiji(!MainFrame.isMiji());     //按S键开关外挂
    }
        public void keyReleased(KeyEvent e) {
            // TODO Auto-generated method stub
            if(e.getKeyCode()==KeyEvent.VK_UP)
                setUp(false);
            if(e.getKeyCode()==KeyEvent.VK_DOWN)
                setDown(false);
            if(e.getKeyCode()==KeyEvent.VK_LEFT)
                setLeft(false);
            if(e.getKeyCode()==KeyEvent.VK_RIGHT)
                setRight(false);
        }
```

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。

2.2.9. npc 中线程的启动，结束和运行

```
public void start(){
    //线程的启动，运行run()
    run1=true;
    Thread thread1=new Thread(this);
    thread1.start();
}
public void stop(){
    //若该场景结束，则调用此方法停止线程
    run1=false;
}
```

```
public void run(){
    //线程run中加入对键盘上下左右按下状态的判断，从而改变坐标，通过sleep来使得每隔30毫秒执行一次，并调用repaint对窗面进行重画
    while(run1){
        if(isUp())
            MainFrame.Player.setY(MainFrame.Player.getY()-MainFrame.Player.pspeed);
        if(isDown())
            MainFrame.Player.setY(MainFrame.Player.getY()+MainFrame.Player.pspeed);
        if(isLeft())
            MainFrame.Player.setX(MainFrame.Player.getX()-MainFrame.Player.pspeed);
        if(isRight())
            MainFrame.Player.setX(MainFrame.Player.getX()+MainFrame.Player.pspeed);
        repaint();
        try{
            Thread.sleep(30);
        }catch(InterruptedException e){
            e.printStackTrace();
        }
    }
```

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。

2.2.10. npc 中重写 update()防止闪烁

```
public void update(Graphics g){
    //重写update防止窗面闪烁
    paint(g);
}
```

2.2.11. npc 中重写 paint()内在缓冲图片上绘图的过程

```
public void paint(Graphics g){
    //重写面板的paint来显示窗面
    int x=MainFrame.Player.getX()+MainFrame.Player.getWid()/2;
    int y=MainFrame.Player.getY()+MainFrame.Player.getHei()/2;
    int Fw=this.getWidth();
    int Fh=this.getHeight();
    //判断是否进入传送门
    if(x>1399&&x<1445&&y>710&&y<753){
        MainFrame.toScene2(1);
    }
    Image offscreen=createImage(getWidth1(),getHeight1());
    //利用双重缓冲，创建offscreen作为缓冲图像，先用g1在offscreen上画，画好后再画到屏幕上，注意offscreen是一个很大的图片，到时候必须画在屏幕外面的左上角然后让特定的部分显示在屏幕上
    Graphics g1=offscreen.getGraphics();
    g1.drawImage(background1, 0, 0, null);
    g1.drawImage(MainFrame.Player.getRole(),MainFrame.Player.getX(),MainFrame.Player.getY(),null);
    npc.change();
    //每次画npc对先执行change方法更新位置
    for(int i=0;i<npc.NPCNUM;i++){
        g1.drawImage(npc.getRole(i),npc.getX(i),npc.getY(i),null);
    }
```

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。

2.2.12. npc 重写 paint()内把缓冲图片绘制到屏幕上的过程

```
if(x>=Fw/2&&x<=getWidth1()-Fw/2&&y>=Fh/2&&y<=getHeight1()-Fh/2)
    //通过判断player所在位置来确定显示在屏幕上的画面在整个背景图上的具体坐标，若不在边缘，则画以player为中心的矩形，然后把offscreen画到屏幕外面的左上角
    g.drawImage(offscreen, Fw/2-x, Fh/2-y,null);
else if(x<Fw/2&&y<Fh/2)
    //若在上下或左右边缘或四个角，那么只画相应位置的矩形，下同
    g.drawImage(offscreen, 0, 0,null);
else if(x<Fw/2&&y>=Fh/2&&y<=getHeight1()-Fh/2)
    g.drawImage(offscreen, 0, Fh/2-y,null);
else if(x<Fw/2&&y>getHeight1()-Fh/2)
    g.drawImage(offscreen, 0, Fh-getHeight1(),null);
else if(x>=Fw/2&&x<=getWidth1()-Fw/2&&y<Fh/2)
    g.drawImage(offscreen, Fw/2-x, 0,null);
else if(x>=Fw/2&&x<=getWidth1()-Fw/2&&y>getHeight1()-Fh/2)
    g.drawImage(offscreen, Fw/2-x, Fh-getHeight1(),null);
else if(x>getWidth1()-Fw/2&&y<Fh/2)
    g.drawImage(offscreen, Fw-getWidth1(), 0,null);
else if(x>getWidth1()-Fw/2&&y>=Fh/2&&y<=getHeight1()-Fh/2)
    g.drawImage(offscreen, Fw-getWidth1(), Fh/2-y,null);
else if(x>getWidth1()-Fw/2&&y>getHeight1()-Fh/2)
    g.drawImage(offscreen, Fw-getWidth1(), Fh-getHeight1(),null);

g.setColor(Color.RED);        //直接在画面顶显示等级
g.setFont(new Font("华文行楷",Font.BOLD,40));
g.drawString("等级: "+MainFrame.Player.getLevel(),120, 50);
g.setColor(Color.ORANGE);
g.drawString("经验: ",300, 50);            //在等级后面用一条非填充矩形作为该等级满经验时的经验，长度为300，用填充矩形表示当前经验
int l = (int) (MainFrame.Player.getExp()/(100.0*MainFrame.Player.getLevel()))*300);  //l为计算当前经验所对应的长度
g.drawRect(420, 20, 300, 40);
g.fillRect(420, 20, l, 40);
g.drawString(" " +MainFrame.Player.getExp(), 420+l, 50);
```

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。

2.2.13. PausePanel 和 EndPanel 中显示最后的等级经验

```
public void paintComponent(Graphics g){      //重写paintComponent来画背景
    super.paintComponent(g);
    g.setColor(Color.ORANGE);
    g.setFont(new Font("华文行楷",Font.BOLD,40));
    g.drawImage(bg, 0, 0, this.getWidth(), this.getHeight(), this);
    g.drawString("您的最终等级为"+MainFrame.Player.getLevel()+" 经验为"+ MainFrame.Player.getExp(),155, 80);
}
```

```
public void paintComponent(Graphics g){      //重写paintComponent来画背景
    super.paintComponent(g);
    g.setColor(Color.ORANGE);
    g.setFont(new Font("华文行楷",Font.BOLD,40));
    g.drawImage(bg, 0, 0, this.getWidth(), this.getHeight(), this);
    g.drawString("您的最终等级为"+MainFrame.Player.getLevel()+" 经验为"+ MainFrame.Player.getExp(),155, 80);
}
```

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。

3. 测试效果

3.1. 开始菜单

图片，文字和按钮成功显示，并且点击效果正确。

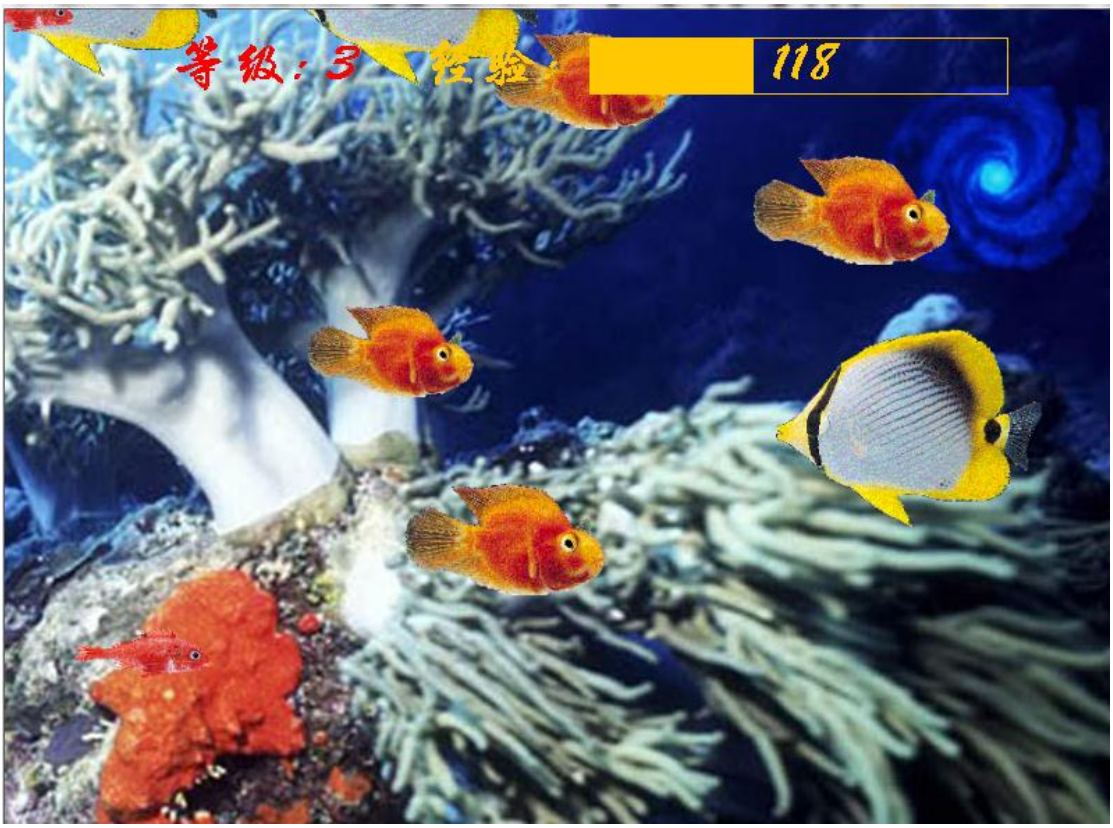
3.2. 场景一游戏显示和操作

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。

3.3. 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。

3.4. 游戏场景切换



能成

功通过传送门进入场景二和三。

3.5. 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。