Java 程 序 设 计 课 程 报 告

病毒大作战

课程名称：Java 程序设计

指导老师：　傅馨　

学　　院：　管理学院　

系　　别：　管理科学　

二〇二〇年十二月二十日

目录

一、实验内容

1.1 游戏介绍

游戏发生在患者的人体系统内，现有大量的各种细菌病毒入侵人体，这些细菌病毒种类不同，体积也各不相同。在本游戏中，玩家将扮演该患者体内的吞噬细胞（游戏内形象为女战士），在保证自己生存的前提下，吞噬比自身体积小的病毒，不断增大自己的体积壮大实力，直到消灭所有来犯之敌。

我们设计的游戏为关卡型，玩家通过鼠标移动人物吞噬病毒以获得积分（营养素）。游戏内设商店，玩家可通过获得营养素在商店中购买血包、无敌卡、药丸、福袋等道具，丰富游戏玩法及游戏体验。

1.2 游戏要点

本游戏开发的关键在于角色和病毒的生成、移动、碰撞后得分或减少生命值以及存档和商店功能的实现。这些关键功能的具体代码将会在报告正文中逐一介绍。

二、实验要求

2.1 基本要求

2.1.1 玩家注册与登录

在正式进入游戏前，提供注册与登录界面，已有账号的玩家可直接输入账号与密码进入游戏，新玩家需要注册账号才能进入游戏。

2.1.2 预游戏界面

登录账号后会进入预游戏界面，在预游戏界面，玩家可以选择新游戏或继续游戏，同时预游戏界面还提供帮助，用于查看游戏规则，以及可以选择返回注册与登录界面。

2.1.3 初始化游戏

正式进入游戏界面后，会生成一幅游戏地图，从不同方向随机生成各种大小不一且形状不同的病毒。生成主角即游戏玩家，同时生成玩家血量 hp、得分、营养素、关卡、最高分以及玩家名称，一旦玩家接触病毒，主角的血量会相应减少。游戏界面同时设置了商店、存档以及结束游戏的按钮。

2.1.4 游戏基本过程

①玩家需要通过移动鼠标来控制主角消灭病毒，即碰到体积小于主角的病毒，主角会吞噬病毒，自身体积会相应变大，并且得分会增加；

②如果玩家碰到体积大于主角的病毒，则主角的血量会相应减少，减少到零即游戏结束；

③随着主角体积增大，玩家可以吞噬更大的病毒；

④每吞噬一个病毒，主角的营养素可增加 2，营养素可以用于在商店购买道具，增强主角的生存能力和作战能力；

⑤当得分达到指定分数时，既可以进入到下一关卡，不同关卡的游戏难度不同。

2.1.5 血量

玩家在游戏开始时拥有血量 hp=100，每当碰到大于主角的病毒时，hp 减

5；可以通过在商店购买道具补血，当 hp=0 时，游戏结束。

2.1.6 商店系统

主角可以通过使用营养素在商店购买道具，增加血量以及提高主角的作战能力。

2.1.7 保存与退出游戏

玩家可以在游戏界面存档，存档后会记录下玩家的游戏数据；玩家可以退出游戏，退出游戏时可以选择存档或者不存档，退出游戏后返回游戏预界面。

2.2 进阶要求

2.2.1 动画视频

在进入游戏时，会自动弹出一个播放抗击疫情的动画视频，随后进入游戏的注册与登录界面。

2.2.2 继续游戏

在游戏的预界面，玩家可以选择新游戏或者继续游戏，若选择继续游戏，则之前登录过游戏账号的玩家可以继续进行上一次的游戏。

2.2.3 帮助界面

在预游戏界面，通过点击“帮助”按钮可以查看本游戏的游戏规则，帮助玩家熟悉基本操作。

2.2.4 道具种类

除基本的血包道具（购买后可以增加主角血量）外，商店还有：无敌、福袋以及药丸。购买无敌道具后，主角可以无视体积大小吞噬 5 个比自己大的病毒；购买福袋道具后，可随机增加主角的营养素；购买药丸道具后，当主角血量少于 100 时，游戏界面会随机出现药丸，主角可以通过吞噬药丸来增加血量，最多只能吞噬 5 个药丸。

2.2.5 病毒种类

病毒种类有三种：普通病毒、流感病毒以及新冠病毒。玩家可以吞噬普通病毒，但是碰到更大体积的普通病毒时，血量会减少；当玩家碰到流感病毒时，主角的体积会回到初始大小，并且得分会减少；当玩家碰到新冠病毒时，主角会死亡。

2.2.6 背景音乐

进入游戏界面后，会自动播放背景音乐。

2.2.7 音效

当玩家吞噬病毒、药丸、或者碰到更大的病毒时，会产生音效。

2.2.8 最高分

此游戏会记录下玩家玩这个游戏以来所获得的最高分。

三、系统设计

3.1 业务逻辑

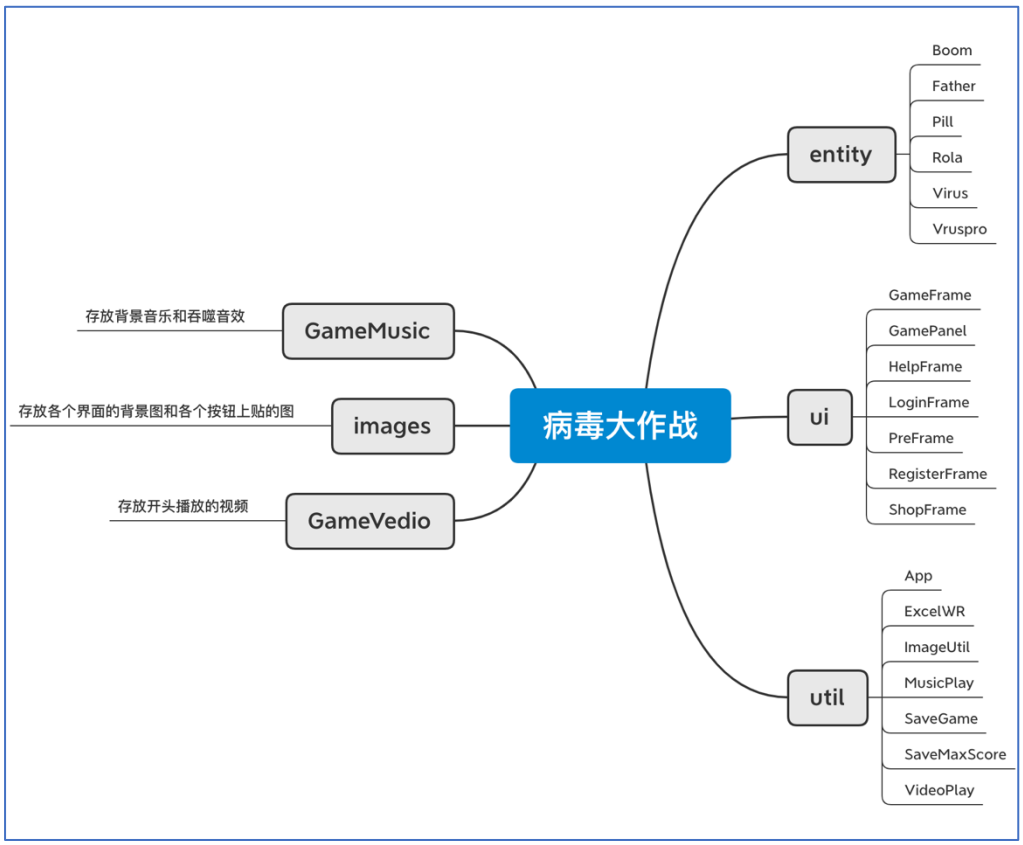

图 1 包与类

该程序一共有 5 个包，GameMusic、images、GameVedio 三个包中分别放的是所需的音乐、图片和视频。entity 包中放的是各种病毒类和玩家类。Ui 包中放置的是游戏中所需的各种界面，页面切换由面板的切换实现，GameFrame 为该程序中唯一的界面，GamePanel、HelpFrame、LoginFrame、PreFrame、RegisterFrame、ShopFrame 都为面板，当需要退出一个面板进入另一个面板时，用 remove()方法移除当前面板， 再用 add()方法添加想要打开的面板。Util 包中的类是定义的各种初始值和封装的各种方法。

3.2 类设计 UML

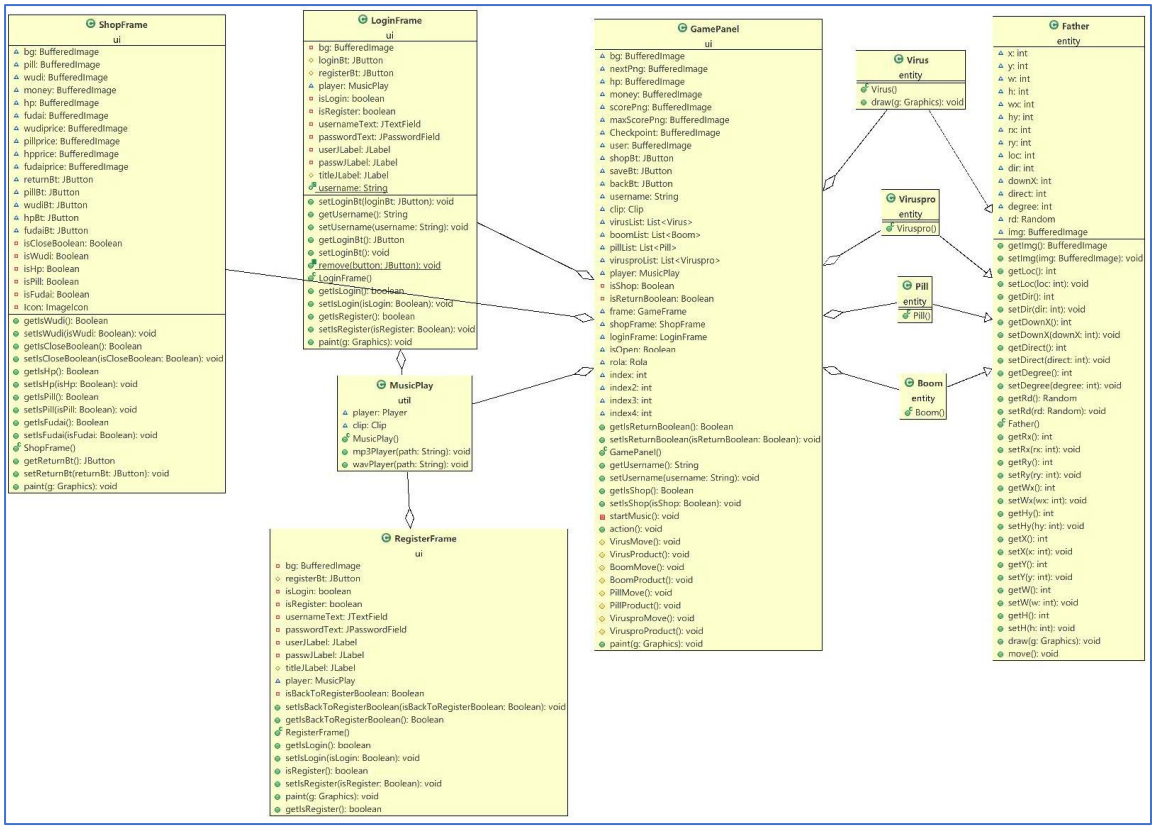

图 2 UML

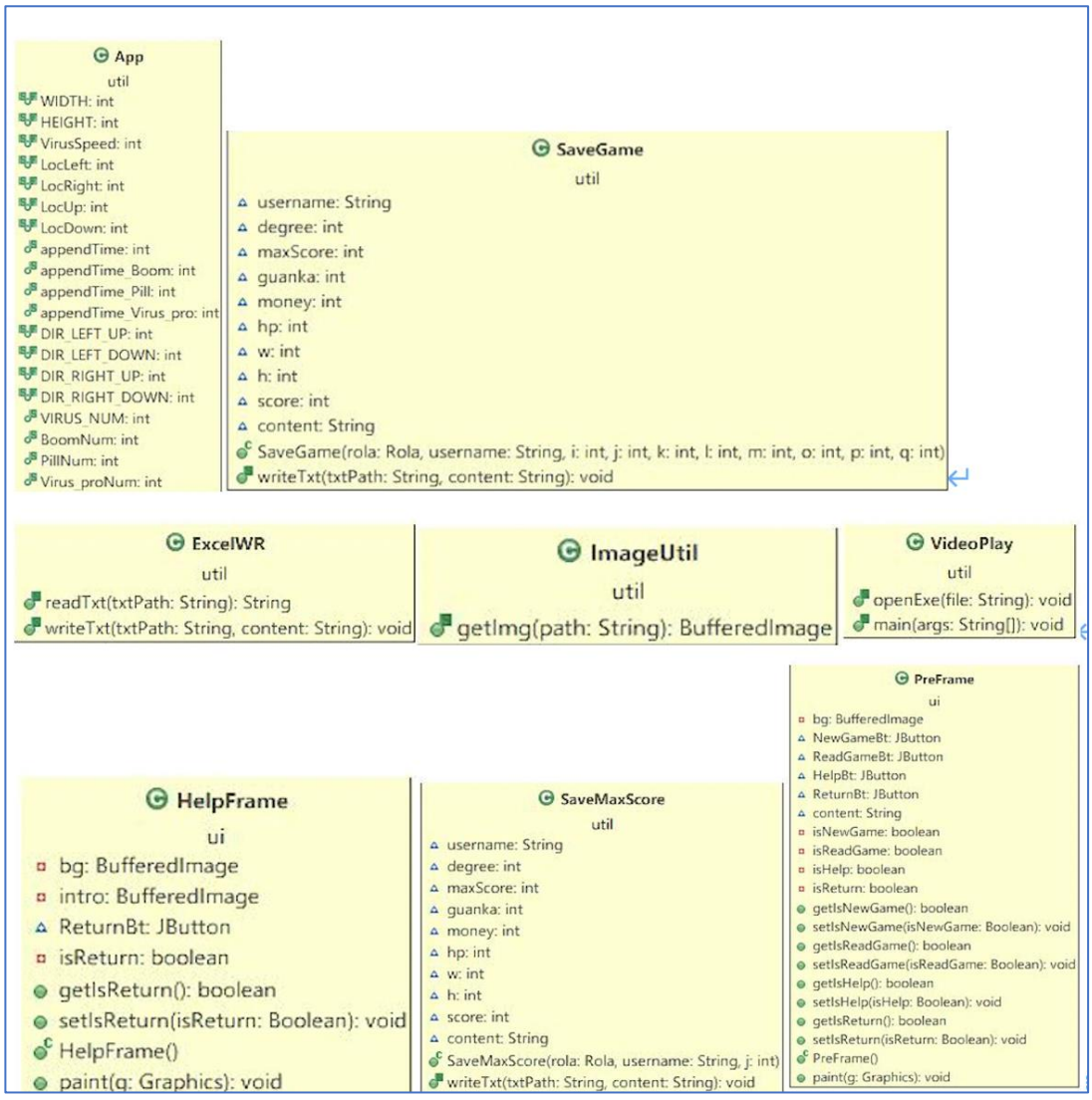

图 3 UML

四、实验过程

4.1 用户注册登录

4.1.1 用户注册

注册类 RegisterFrame 首先定义了注册、返回按钮和 JTextField 型变量 usernameText 用于账号输入以及一个 JPasswordField 型变量

passwordText 用于密码输入。同时设置布尔类型变量 isLogin 和 isRegister 用以判断是否注册或登录成功。

对注册、返回按钮添加监听器，若点击注册按钮，读取输入的账号 usernameText 和密码 passwordText 内容。如果输入的账号不曾被注册（即不存在于 usermesg 中），且账号密码都不为空，则将账号密码与初始游戏数据写入 usermesg 中，将 isRegister 设为 true。

```
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    String username=usernameText.getText();
    String password=passwordText.getText();
    String usermesg=wr.readTxt("./usermesg.txt");
    System.out.println(usermesg);
    String[] usermesg_split=usermesg.split("#");
    ArrayList<String> userlist = new ArrayList<String>();
    System.out.print("username:"+username);
    for(String li : usermesg_split) {
        if (li=="\t\n") {
            continue;
        }
        try {
            String[] li_spilt=li.split("\t");
            userlist.add(li_spilt[0]);
        } catch (Exception e2) {
            continue;
            // TODO: handle exception
        }
    }
```

图 4 读取账号密码内容

```
if (!userlist.contains(username)&&!username.equals("")&&!password.equals("")) {
    System.out.println("writer success~~");
    //初始化人物等级 关卡等
    String degree="1";
    String maxScore="0";
    String guanka="1";
    String money="100";
    String hp="100";
    String w="100";
    String h="100";
    String score="0";

    wr.writeTxt("./usermesg.txt", username+"\t"+password+"\t"+degree+"\t"+maxScore+"\t"+guanka+
            "\t"+money+"\t"+hp+"\t"+w+"\t"+h+"\t"+score);
    JOptionPane.showMessageDialog(null,"注册成功 ","注册提示",JOptionPane.INFORMATION_MESSAGE);
    isRegister= true;
}
```

图 5 新用户数据写入 usermesg 中

若 username 或 password 为空，则提示用户名或密码为空。

```
else if (username.equals("")||password.equals("")) {
    JOptionPane.showMessageDialog(null,"注册失败,用户名或者密码为空 ","注册提示",JOptionPane.INFORMATION_MESSAGE);

}
```

图 6 用户名或密码为空

若 userlist 中已存在注册的用户名，则提示用户名已存在。

```
else {

    JOptionPane.showMessageDialog(null,"注册失败,该用户已存在 ","注册提示",JOptionPane.INFORMATION_MESSAGE);
}
```

图 7 用户名已存在

4.1.2 用户登录

登录类 LoginFrame 与注册类布局结构类似，定义两个按钮分别为登录与注册。对登录和注册按钮添加了监听器，若点击登录，则分别读取 usernameText 和 passwordText。若账号和密码数据已存在于 usermesg 中，则将 Boolean 类型 isLogin 值设为 true。

```
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    //登陆成功
    //获取用户名密码
    String username=usernameText.getText();
    String password=passwordText.getText();
    String usermesg=wr.readTxt("./usermesg.txt");
    System.out.println(usermesg);
    String[] usermesg_split=usermesg.split("#");
    ArrayList<String> userlist = new ArrayList<String>();
    for(String li : usermesg_split) {
        if (li=="\t\n") {
            continue;
        }
        try {
            String[] li_spilt=li.split("\t");
            String li_password=li_spilt[1].replace("\\\n", "").replace("\n", "").replace("\\\\n", "");
            String li_usernameString=li_spilt[0];
            System.out.println("username:"+li_usernameString);
            System.out.println("password:"+li_password);
```

图 8 读取账号密码内容

```
if (username.equals(li_usernameString)&&password.equals(li_password)) {
    setUsername(username);
    System.out.println("login success~~");
    isLogin=true;
    break;
}
```

图 9 判断账号密码是否正确

若账号密码不匹配，则提示账号或密码错误。

```
if (isLogin==false&&username.equals("")==false&&password.equals("")==false) {
    JOptionPane.showMessageDialog(null,"登陆失败,账号或密码错误 ","登陆提示",JOptionPane.INFORMATION_MESSAGE);
}
```

图 10 账号或密码错误

若 username 或 password 为空，则提示用户名或密码为空。

```
if (username.equals("")||password.equals("")) {
    JOptionPane.showMessageDialog(null,"请输入账户和密码","登陆提示",JOptionPane.INFORMATION_MESSAGE);
}
```

图 11 账号或密码为空

4.2 预界面

预界面类 PreFrame 中定义四个按钮，新游戏 NewGameBt、继续游戏 ReadGameBt、帮助 HelpBt 和返回 ReturnBt。同时设置四个初始值为 false 的布尔值 isNewGame、isReadGame、isHelp 和 isReturn 用于后续面板控制。对每个按钮设置监听器。点击按钮后其对应的布尔值变为 true。

```
ReadGameBt.addActionListener(new ActionListener()
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        isReadGame=true;
    }
});

HelpBt.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        isHelp=true;
    }
});

ReturnBt.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        isReturn=true;
    }
});
```

图 12 添加监听器更改布尔值

若点击新游戏按钮，则更新该用户名下的 usermesg 中的存档内容，更新为最初值并存档。

```
public void actionPerformed(ActionEvent e) {
    // TODO Auto-generated method stub
    isNewGame=true;
    ExcelWR wr=new ExcelWR();
    String usermesg=wr.readTxt("./usermesg.txt");
    System.out.println(usermesg);
    String[] usermesg_split=usermesg.split("#");
    ArrayList<String> userlist = new ArrayList<String>();
    for(String li : usermesg_split) {
        if (li=="\t\n") {
            continue;
        }
        try {
            String[] li_spilt=li.split("\t");
            System.out.println(li_spilt[0]);
            System.out.println(username);
            rola.setHp(100);
            rola.setDegree(1);
            rola.setGuanka(1);
            rola.setMo(100);
            rola.setW(50);
            rola.setH(50);
            rola.setSc(0);
            rola.setMaxScore(Integer.parseInt(li_spilt[3]));
            SaveGame saveGame=new SaveGame(rola,username, rola.getDegree(), rola.getMaxScore(),
                    rola.getGuanka(), rola.getMo(),rola.getHp(),rola.getW(),rola.getH(),rola.getSc());
        } catch (Exception e2) {
            continue;
            // TODO: handle exception
        }

    }
}
```

图 13 新游戏按钮监听以及存档数据更新

4.3 游戏初始化

4.3.1 游戏角色的生成

角色类 Rola 的构造函数内定义了角色初始的大小、位置、等级、关卡、血量、分数、营养素，如果开始新游戏，各变量会按照初始化大小生成；如果是继续游戏，会按照角色名读档角色的历史信息，各变量会按照历史大小生成。

角色的绘制通过 drawImage()方法将图像按指定位置裁剪后复制到画布。

4.3.2 病毒及道具的生成

三个病毒类以及道具类所包含的属性和方法比较类似，所以采用了继承

的思想。父类 Father 中包含了大小、等级、下落位置、移动方向、图片裁剪位置等变量；定义了读取图片以及将图片复制到画布上的方法、移动的方法，以及各种获取和重置位置、大小的方法。

普通病毒 Virus 类、流感病毒 Viruspro 类、新冠病毒 Boom 类和药丸 Pill 类作为子类，继承了父类中的变量和方法，并在构造方法里设置了各个变量的值以及普通病毒出现的方向。其中普通病毒会随机生成 15 种大小，流感病毒、新冠病毒、药丸的大小为固定值。

病毒和药丸生产、移动和绘制的方法均相同，只是产生的速度和数量参数设置不同。生成的普通病毒、流感病毒、新冠病毒、药丸都采用 List 存储，并通过不同参数控制产生的数量和速度。

```
//生产病毒
int index=0;
protected void VirusProduct() {
// TODO Auto-generated method stub
    index++;//计数器用于控制球产生的速度
    if (index>App.appendTime && virusList.size()<App.VIRUS_NUM) {
        Virus virus=new Virus();
        virusList.add(virus);
        index=0;
    }

 }
```

图 14 生产病毒的方法

```
// 移动的方法
public void move() {
    // TODO Auto-generated method stub

    if(loc==App.LocLeft) {
        x+=App.VirusSpeed;
        if (x>downX) {
            dir=App.DIR_RIGHT_DOWN;
        }
    }else if(loc==App.LocRight) {
        x-=App.VirusSpeed;
        if (x<(App.WIDTH-downX)) {
            dir=App.DIR_LEFT_DOWN;
        }
    }else if(loc==App.LocUp) {
        x-=App.VirusSpeed;
        if (x<(App.WIDTH-downX)) {
            dir=App.DIR_LEFT_DOWN;
        }
    }else if(loc==App.LocDown) {
        x-=App.VirusSpeed;
        if (x<(App.WIDTH-downX)) {
            dir=App.DIR_LEFT_DOWN;
        }
    }
    if (dir==App.DIR_LEFT_DOWN) {

        if (direct==0) {
            x-=App.VirusSpeed/2;
            y+=App.VirusSpeed;
        }else {
            x-=App.VirusSpeed/2;
            y-=App.VirusSpeed+h;
        }

    }
    if (dir==App.DIR_RIGHT_DOWN) {

        if (direct==0) {
            x+=App.VirusSpeed/2;
            y+=App.VirusSpeed;
        }else {
            x+=App.VirusSpeed/2;
            y-=App.VirusSpeed+h;
        }

    }
}
```

图 15 移动的方法

4.4 游戏过程

4.4.1 角色的移动

对角色建立鼠标监听，使角色跟随鼠标移动。首先获取鼠标的坐标，再将角色的坐标重置为鼠标的坐标，实现鼠标监听。

```
//创建鼠标适配器
MouseAdapter adapter=new MouseAdapter() {
    @Override
    public void mouseMoved(MouseEvent e) {
        // TODO Auto-generated method stub
        //获取鼠标位置
        int mx=e.getX();
        int my=e.getY();
        rola.moveTO(mx, my);
        repaint();

    }
};
将适配器加监听器中；
addMouseListener(adapter);
addMouseMotionListener(adapter);
```

图 16 对角色鼠标监听

4.4.2 病毒和药丸的移动

创建线程，调用生产和移动函数，使病毒和药丸动起来。在调用前设置判断语句，控制不同病毒出现的关卡，普通病毒在游戏一开始便可以生成，流感病毒在第二关生成，新冠病毒在第三关生成，药丸在商店购买后才会生成。

```
//开启线程让病毒和药丸动起来
new Thread() {
    public void run() {
        while (true) {
         //生产各种病毒和药丸
            try {
                Thread.sleep(20);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            //普通病毒
            VirusProduct();
            VirusMove();
            repaint();

            //病毒pro
            if(rola.getGuanka()>=2) {
                VirusproProduct();
                VirusproMove();
                repaint();
            }
            //新冠病毒
            if(rola.getGuanka()>=3) {
                BoomProduct();
                BoomMove();
                repaint();
            }

            //药丸
            if(rola.getIsPill()==true) {
                PillProduct();
                PillMove();
                repaint();
            }

                }

            };

}.start();
```

图 17 病毒、药丸的生成和移动

4.4.3 碰撞过程

通过坐标判断角色是否与病毒碰撞，角色与普通病毒碰撞后，先判断二者大小，如果角色比普通病毒大，角色吞噬普通病毒，并且增加体积、分数、营养素。当体积变大后，可以吞噬更大的病毒；当分数达到一定数量后，角色等级会增加，关卡也会增加，不同的关卡出现的病毒种类不同，病毒出现的频率会数量也不同；营养素可以用于在商店购买道具。

如果角色比普通病毒小，判断是否购买无敌道具，如果购买无敌道具，角色的生命值不发生变化，无敌道具使用五次后将失效。如果未购买无敌功能，角色的生命值会降低 5，当生命值降为 0 时，游戏结束。

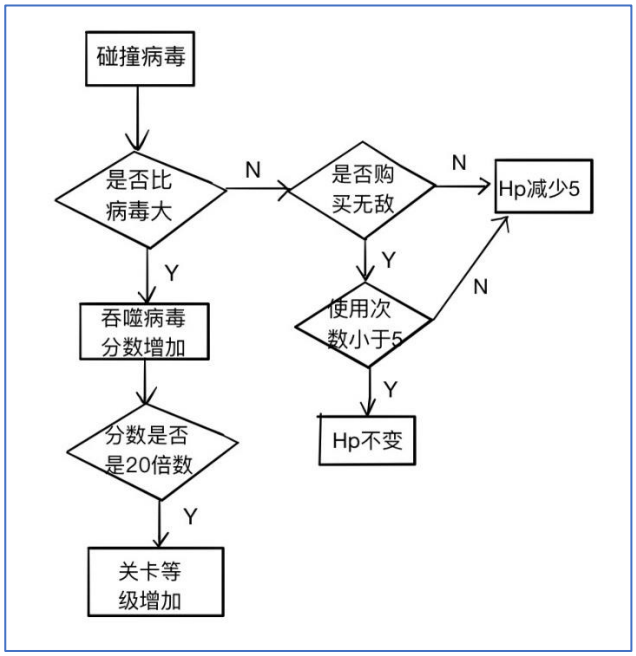

图 18 碰撞普通病毒的逻辑流程图

角色碰撞流感病毒后，大小变为初始值 50，Hp 值减少 5；角色碰撞新冠病毒后，Hp 变为 0，游戏结束。

角色碰撞药丸后，判断生命值是否小于 100，如果小于 100，判断吞噬的药丸数量是否小于 5，如果小于 5，吞噬药丸，Hp 值增加 5；如果吞噬药丸数量大于 5，药丸消失。如果生命值等于 100，无法吞噬药丸。

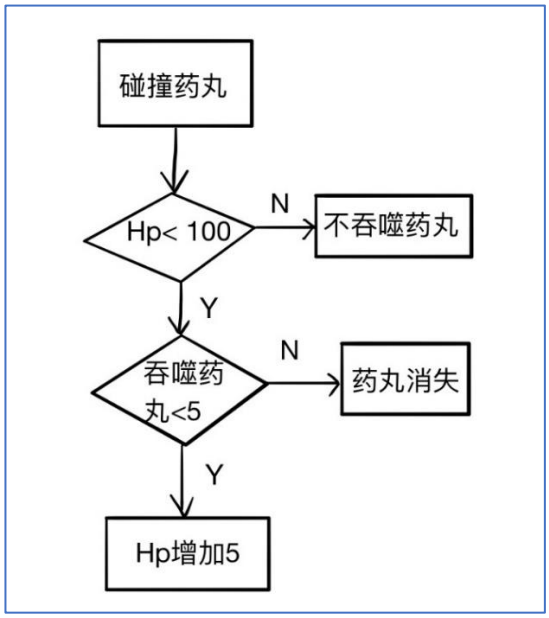

图 19 碰撞药丸的逻辑流程图

4.5 商店系统

商店里一共有四种道具：药丸、无敌、血包和福袋，对应的营养素价格分别是 30、20、10、30。

4.5.1 药丸道具

对药丸的购买按钮添加监听器，当点击购买药丸后，营养素会减少 30，Boolean 类型变量 IsPill 会变为 True。当 IsPill 为 True 时，游戏界面会生成药丸，角色碰撞药丸后吞噬药丸，生命值加 5，吞噬 5 个药丸后，IsPill 变为 False，药丸消失。

```
//药丸的事件监听器
pillBt.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        setIsPill(true);
        JOptionPane.showMessageDialog(null,"购买药丸成功","商店",JOptionPane.INFORMATION_MESSAGE);
    }

});
```

```
//检测是否点击药丸
  new Thread() {
      public void run() {
          while (true) {
              try {
                  Thread.sleep(50);
              } catch (InterruptedException e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
              }
              if (shopFrame.getIsPill()) {
                  panel.rola.setIsPill(true);
                  panel.rola.setMo(panel.rola.getMo()-30);//消耗30营养素
                  System.out.println("购买药丸");
                  break;
              }
          }
      }
  }.start();
```

图 20、21 购买药丸

4.5.2 无敌道具

对无敌的购买按钮添加监听器，当点击购买无敌后，营养素会减少 20，Boolean 类型变量 IsWudi 会变为 True。当 IsWudi 为 True 时，角色碰到比自己大的普通病毒时，生命值不会改变，当碰到五个比自己大的普通病毒时，IsWudi 变为 False，无敌道具失效。为无敌购买按钮添加监听器与购买药丸类似，故此处不贴出代码。

```
//检测是否点击无敌
new Thread() {
    public void run() {
        while (true) {
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            if (shopFrame.getIsWudi()) {
                panel.rola.setIswudi(true);
                panel.rola.setMo(panel.rola.getMo()-20);//消耗20营养素
                System.out.println("购买无敌");
                break;
            }
        }
    }
}.start();
```

图 22 购买无敌

4.5.3 血包道具

对血包的购买按钮添加监听器，当点击购买血包后，营养素会减少 10，Boolean 类型变量 IsHp 会变为 True。当 IsHp 为 True，且血量小于等于 100 时，角色的生命值会增加 10。

```
//检测是否点击买血包
new Thread() {
     public void run() {
         while(true) {
             try {
                 Thread.sleep(50);
             } catch (InterruptedException e) {
                 // TODO Auto-generated catch block
                 e.printStackTrace();
             }

              if (shopFrame.getIsHp()) {
                  panel.rola.setIsHp(true);
                  if(panel.rola.getHp()<=90) {
                         System.out.println("购买血包");
                         panel.rola.setHp(panel.rola.getHp()+10);//生命值增加10
                         panel.repaint();
                         panel.rola.setMo(panel.rola.getMo()-10);//消耗10营养素
                     }
              break;

              }
         }
     }
 }.start();
```

图 23 购买血包

4.5.4 福袋道具

对血包的购买按钮添加监听器，当点击购买血包后，营养素会减少 30，Boolean 类型变量 IsFudai 会变为 True。当 IsFudai 为 True，角色会随机增加 0-50 的营养素。

```
//检测是否点击买福袋
new Thread() {
    public void run() {
        while(true) {
            try {
                Thread.sleep(50);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }

            if (shopFrame.getIsFudai()) {
                System.out.println("购买福袋成功");
                Random rd= new Random();
                panel.rola.setMo(panel.rola.getMo()+rd.nextInt(51));//0-50
                panel.rola.setMo(panel.rola.getMo()-30);//花费30
                panel.repaint();
            break;

            }
        }
    }
}.start();
```

图 24 购买福袋

4.6 用户账户信息保存

4.6.1 用户信息存入 txt

ExcelWR 用于写入和返回用户密码和游戏数据。

```
public static String readTxt(String txtPath) {
        File file = new File(txtPath);
        if(file.isFile() && file.exists()){
            try {
                FileInputStream fileInputStream = new FileInputStream(file);
                InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream);
                BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

                StringBuffer sb = new StringBuffer();
                String text = null;
                while((text = bufferedReader.readLine()) != null){
                    sb.append(text);
                }
                return sb.toString();
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        return null;
    }
```

图 15 传入 txt 路径，读取 txt 内容

```
public static void writeTxt(String txtPath,String content){
    FileOutputStream fileOutputStream = null;
    File file = new File(txtPath);
    try {
        if(!file.exists()){
            //判断文件是否存在，如果不存在就新建一个txt
            file.createNewFile();
        }
        else {
            fileOutputStream = new FileOutputStream(file,true);
        }
        content=content+"#";
        fileOutputStream.write(content.getBytes());
        fileOutputStream.flush();
        fileOutputStream.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

图 26 使用 FileOutputStream 来写入 txt 文件

4.6.2 用户密码与游戏数据保存

游戏数据保存类SaveGame定义了 username、degree、maxScore、guanka、money、hp、w、h、score成员变量。

```
public class SaveGame {
    String username;
    int degree;
    int maxScore;
    int guanka;
    int money;
    int hp;
    int w;
    int h;
    int score;
    String  content ;
    public SaveGame(Rola rola,String username,int i, int j, int k,int l,int m,int o,int p,int q)
```

图 27 SaveGameL 类成员变量

该类用于数据保存，先通过ExcelWR中的readTxt读取usermesg文件，并且筛选出与当前账号用户名匹配的账户数据。然后将当前 Rola 的各项值存为数组然后利用writeTxt重新写入usermesg中。

```
System.out.println(content);
writeTxt("./usermesg.txt", content);
```

图 28、29 将当前数值存入 txt

4.6.3 最高分保存

与 SaveGame 类类似，SaveMaxScore 在读取当前账户存档后，更新存档内最高分 maxScore，即将 maxScore 值赋予 li_split[3]然后存入 txt 中。

```
if (username.equals(li_spilt[0])) {
    System.out.println("pipei success~");
    System.out.println(li_spilt[6]);
            li_spilt[3]=this.maxScore+"";
}

for (String ji : li_spilt) {
    content+=ji+"\t";
}
content+="#";
```

图 30 保存当前最高分

同时因为最高分应该是实时更新而非点击存档才可更新的，所以在 GamePanel 内每一次成功吃掉病毒后添加一个判断语句并调用 SaveMaxScore(Rola rola,String username,int j)以自动更新最高分。

```
if(rola.getSc()>rola.getMaxScore()) {
    rola.setMaxScore(rola.getSc());
    SaveMaxScore savems=new SaveMaxScore(rola,getUsername(),rola.getSc());
}
```

图 31 每次分数增加后判断、更新最高分

4.7 背景音乐和游戏音效

4.7.1 背景音乐

采用 Clip 类播放背景音乐，背景音乐用 wav 格式存储。播放音乐的方法设置为播放一次循环播放，当游戏开始时调用该方法播放背景音乐。

```
//播放音乐的方法
private void startMusic() {
    // TODO Auto-generated method stub
    try {
        clip=AudioSystem.getClip();
        AudioInputStream in=AudioSystem.getAudioInputStream(GamePanel.class.getResource("../GameMusic/BGM_002.wav"));
        clip.open(in);

        clip.start();//只播放一次
        clip.loop(Clip.LOOP_CONTINUOUSLY);//循环播放 count=次数;Clip.LOOP_CONTINUOUSLY无限循环
    } catch (Exception e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

图 32 播放背景音乐

4.7.2 游戏音效

角色与病毒、道具的碰撞都设置了音效，播放音效的方法与播放背景音乐的方法相同。

```
public class MusicPlay {
    Player player;
    Clip clip;
    public MusicPlay() {

    }

    public void wavPlayer(String path) {
        try {
            clip=AudioSystem.getClip();
            AudioInputStream in=AudioSystem.getAudioInputStream(GamePanel.class.getResource(path));
            clip.open(in);

            clip.start();//只播放一次
        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
```

图 33 播放音乐的方法

```
if(rola.getH()>virus.getH()&&rola.getW()>virus.getW()) {//判断角色大小和病毒大小

    player.wavPlayer("../GameMusic/SYS_017.wav");
    virusList.remove(virus);//吞噬病毒
```

图 34 吞噬病毒音效

4.8 视频播放

```
new Thread() {
    public void run() {
        while (true) {
            try {
                Thread.sleep(20);
            } catch (InterruptedException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            File file = new File("GameVideo\\video.mp4");
            String path;
            path = file.getAbsolutePath();//由相对路径得到该文件的绝对路径
            String p = "\""+path.toString()+"\"";//解决路径中空格问题，
            System.out.println(p);
            System.out.println("准备播放");
            video.openExe(p);
            System.out.println("播放成功");
            break;
        }
            };
        }.start();
```

图 35 播放视频的方法

为了在登录时自动播放一段视频，如游戏的 CG，可以在 GameFrame 的 class 里，因为 class 里线程都是平行的，所以直接创建一个线程。在运行电脑中预先存入要播放的视频，注意由相对路径得到该文件的绝对路径。最终在主界面运行游戏后，会直接跳转到与播放视频界面。

五、游戏操作

5.1 登录界面

点击运行后即可进入游戏，本游戏第一个界面为登录界面。登录界面主要由五个部分组成，分别为：登录界面文字说明框、用户名输入框、密码输入框、登录按钮以及注册按钮。

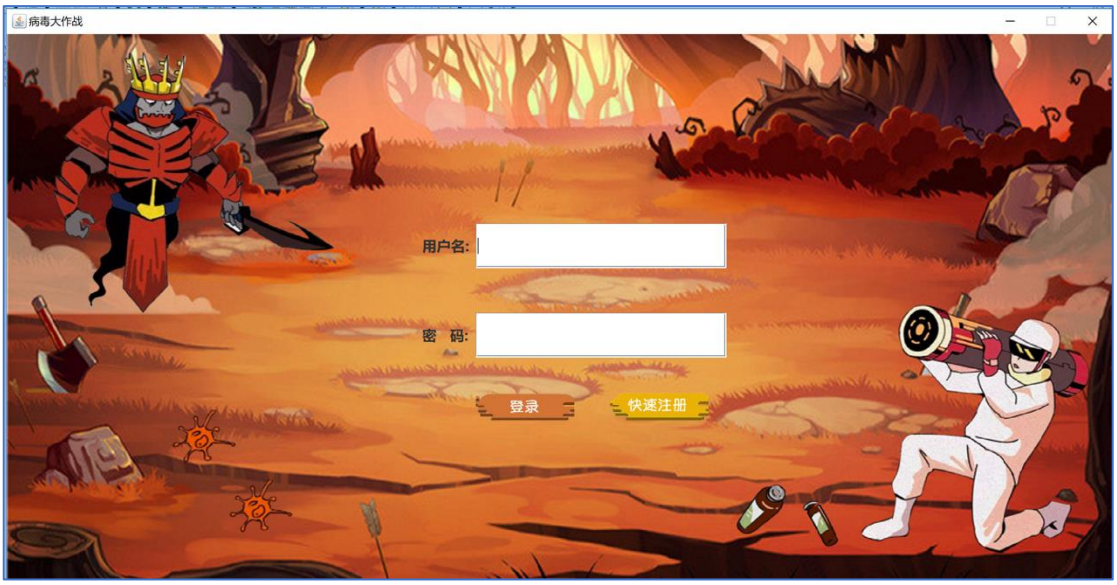

图 36 登录界面

本界面的主要功能为让已注册好的账号进行游戏登录，并为未注册的用户提供账号注册的功能。

（1）登录界面文字说明框提供界面信息显示功能。

（2）用户可在用户名输入框输入已注册的账号，在密码输入框输入账号所对应的密码。

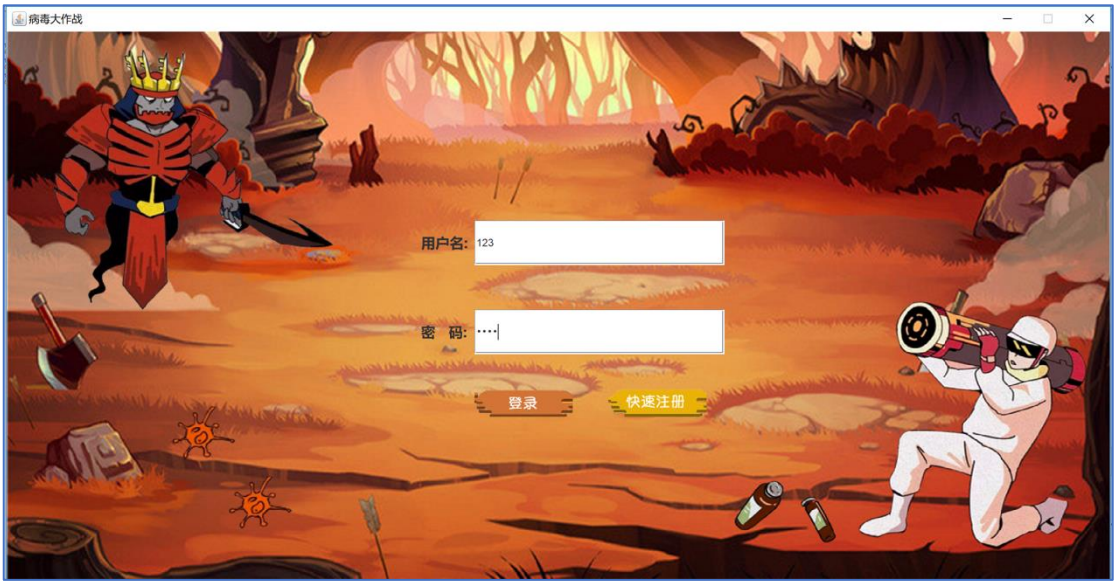

图 37 输入用户名和密码

（3）若账号密码输入正确则进入预界面。

（4）若账号密码输入错误则弹出“登录失败，账号或密码错误”。

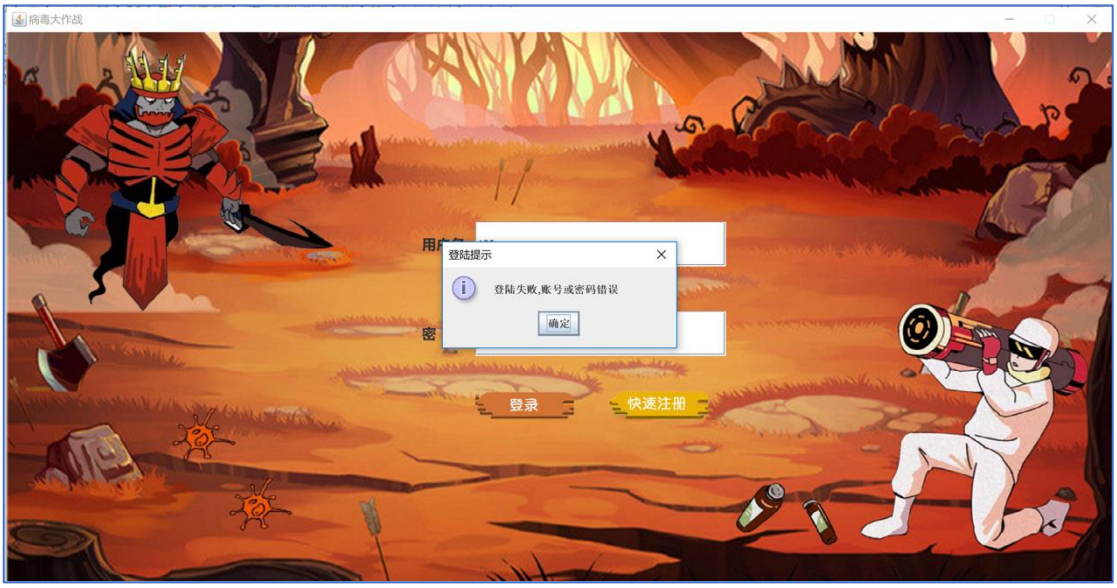

图 38 账户或密码输入错误

（5）若为新用户，未拥有游戏账号，则可以点击注册按钮进入注册界面进行注册。

5.2 注册界面

通过登录界面中点击注册按钮，即可进入注册界面。注册界面主要由五个部分组成，分别为：注册界面文字说明框、用户名输入框、密码输入框、注册按钮以及返回按钮。

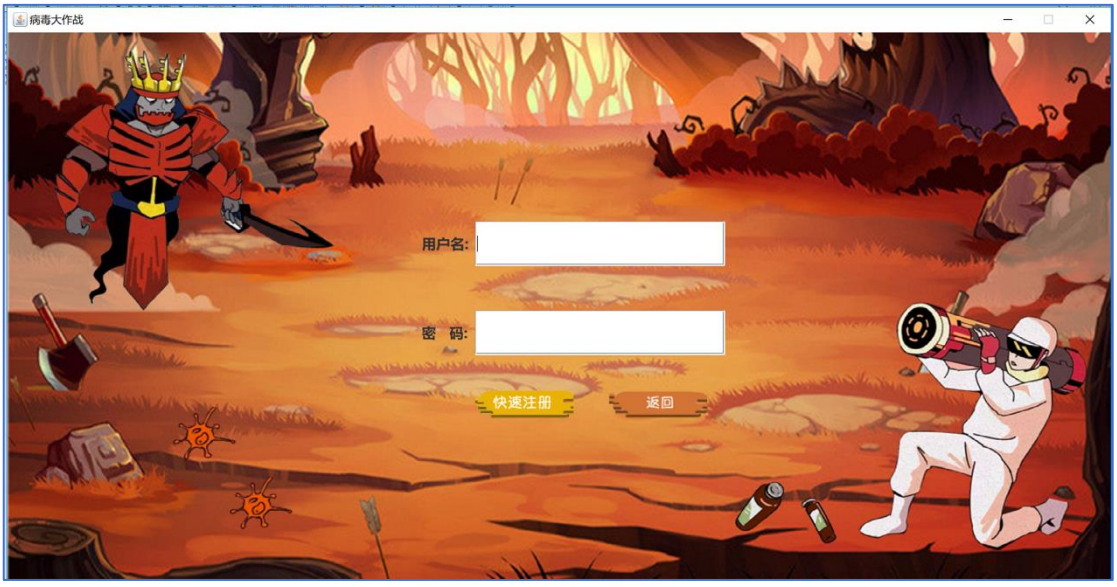

图 39 注册界面

本界面的主要功能是为未注册用户提供账号注册功能。

（1）注册界面文字说明框提供注册界面信息显示功能。

（2）用户可在用户名输入框中输入任意字符作为自己的账号名称。

（3）用户可在密码输入框中设置与账号相对应的密码。

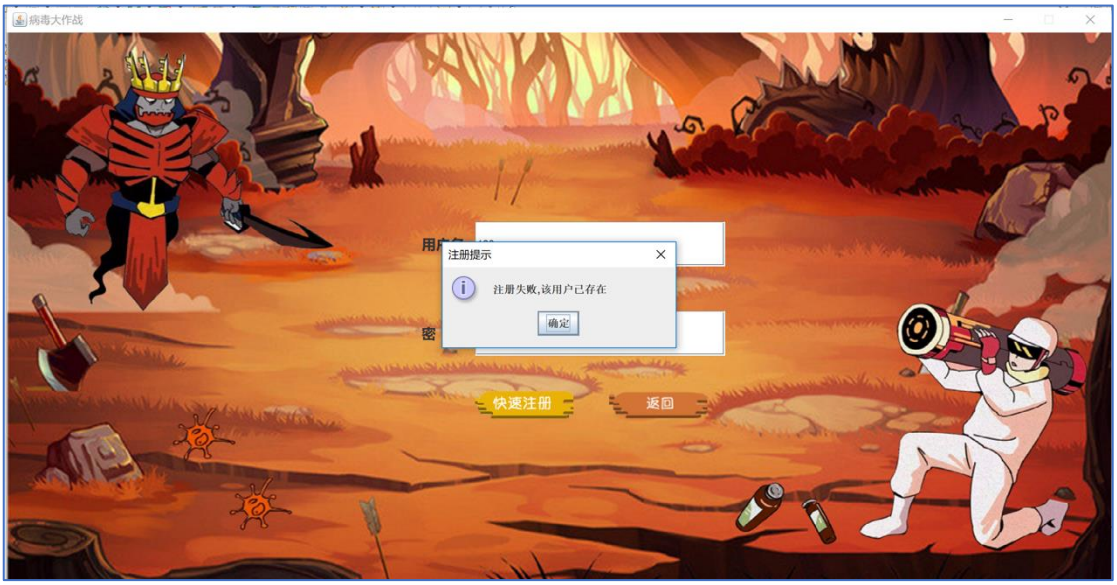

图 40 用户已存在

（4）若用户输入的注册信息正确，则提示 “注册成功”信息

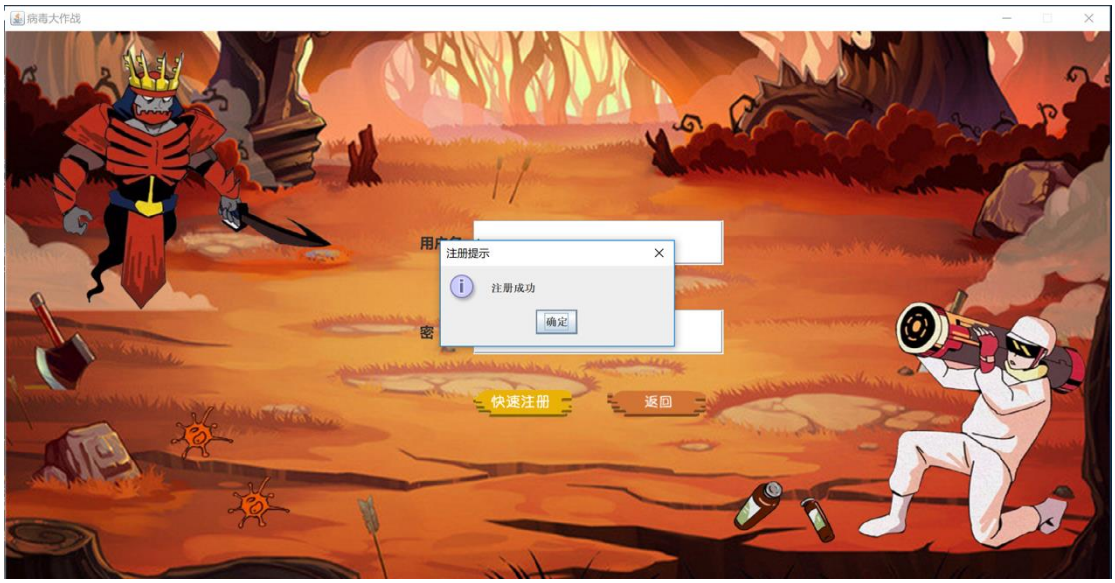

图 41 注册成功

（5）若用户输入的注册信息为空，则提示 “注册失败，用户名或密

码为空”信息。

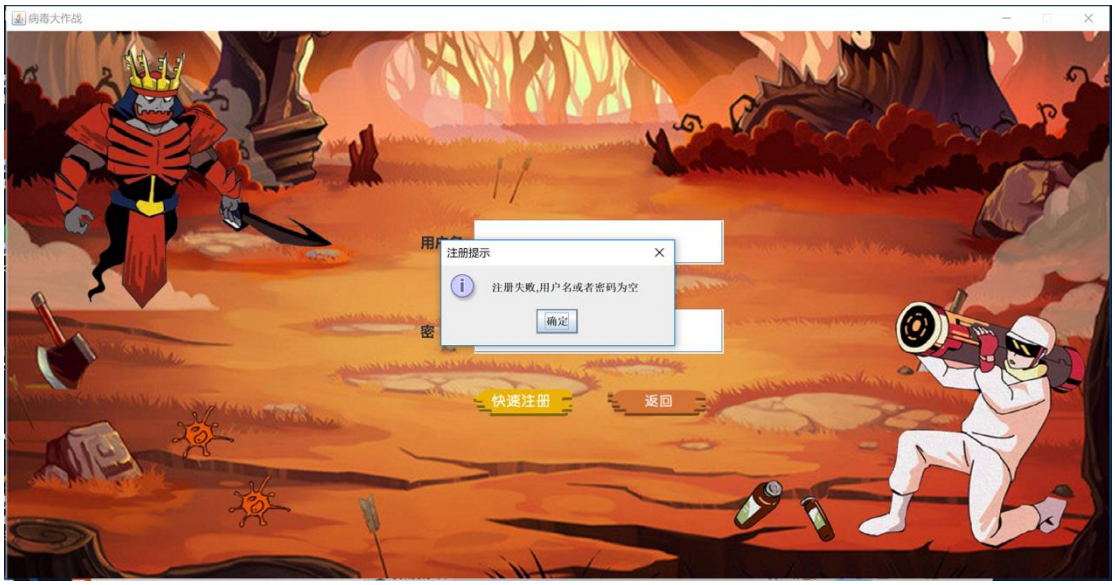

图 42 注册信息为空

（6）若正在注册的用户已存在，则提示“注册失败，该用户已存在”信息

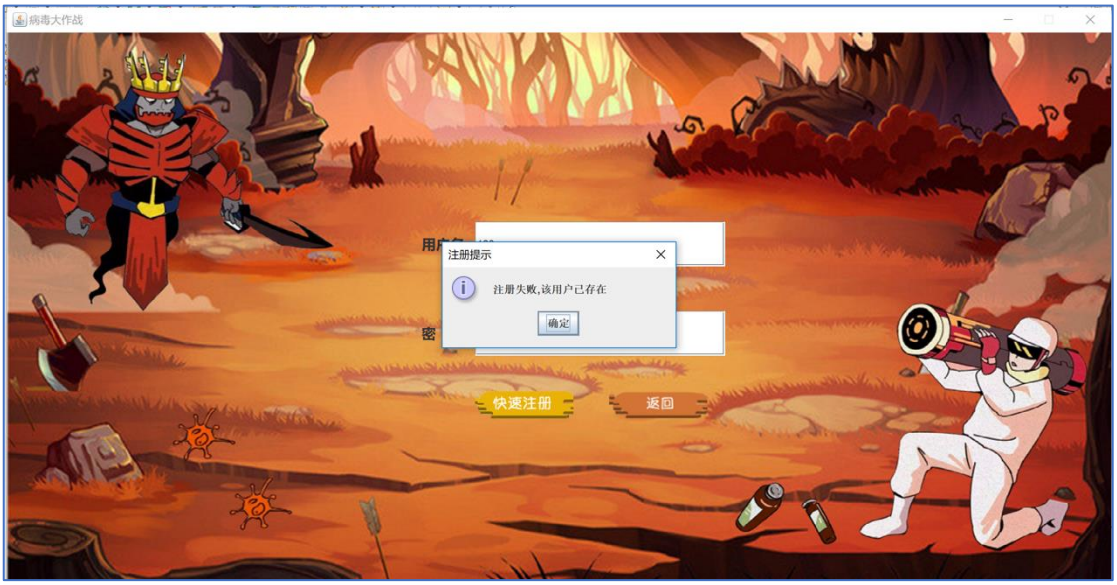

图 43 用户已存在

（7）点击返回按钮可返回登录界面进行登录

5.3 预界面

通过登录界面中点击登录按钮，即可进入游戏预界面。预界面主要由四个部分组成，分别为：新游戏按钮、继续游戏按钮、帮助按钮以及返回按钮。

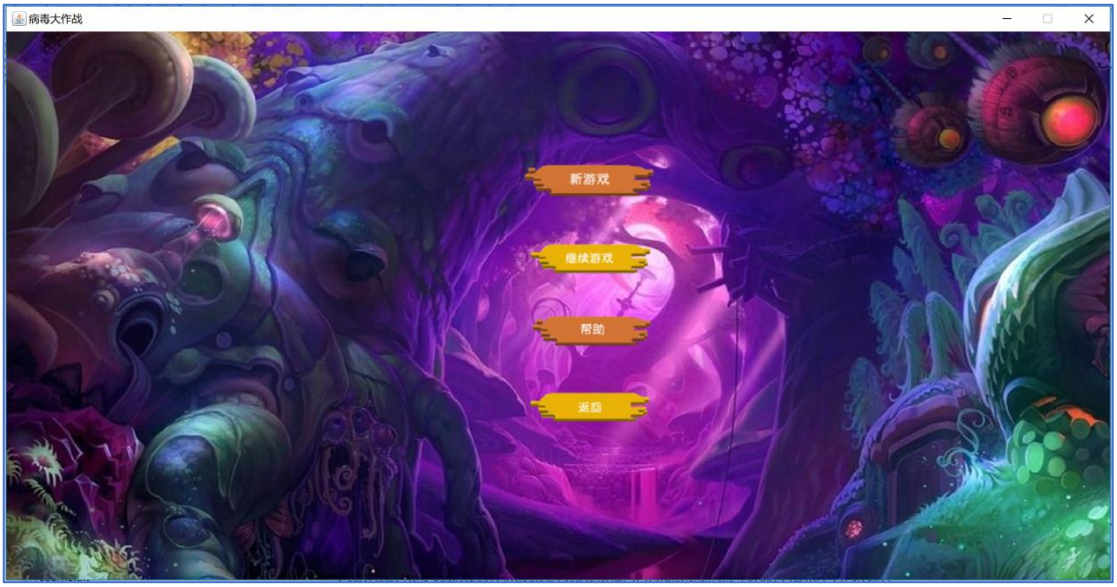

图 44 预界面

本界面的主要功能是为已登录用户提供游戏内选项的功能。

(1) 用户可通过点击新游戏按钮开始新游戏，同时覆盖上次游戏的存档。

(2) 用户可通过点击继续游戏按钮继续上次游戏。

(3) 用户可通过点击帮助按钮获取帮助信息。

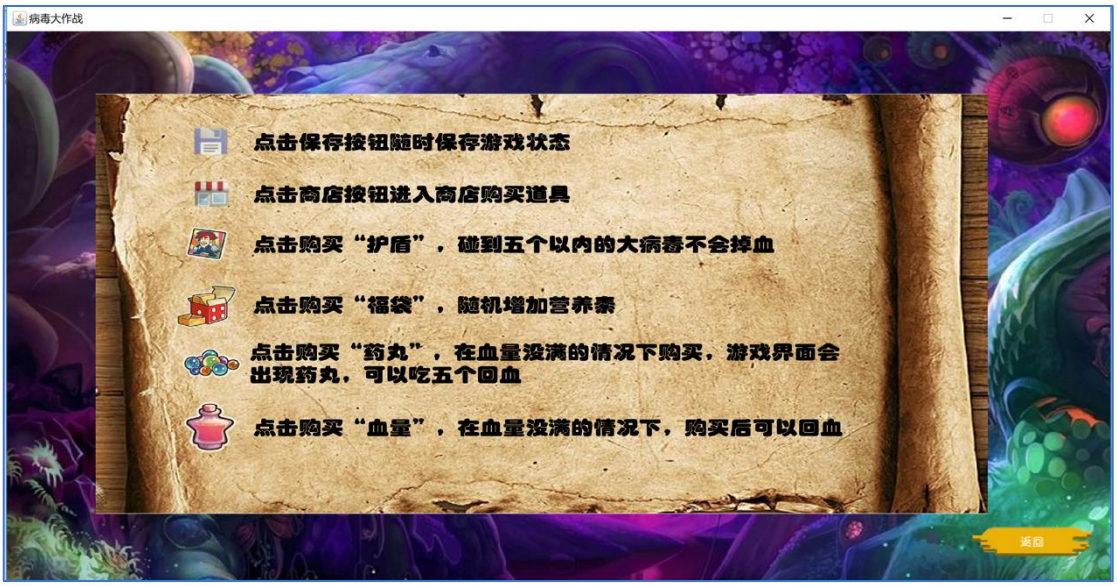

图 45 帮助界面

用户可通过点击返回按钮返回登录界面

5.4 游戏界面

通过在预界面中点击新游戏或继续游戏按钮，即可进入游戏界面。游戏界面主要由三个部分组成，分别为：游戏内信息、游戏内选项以及游戏操作画面。

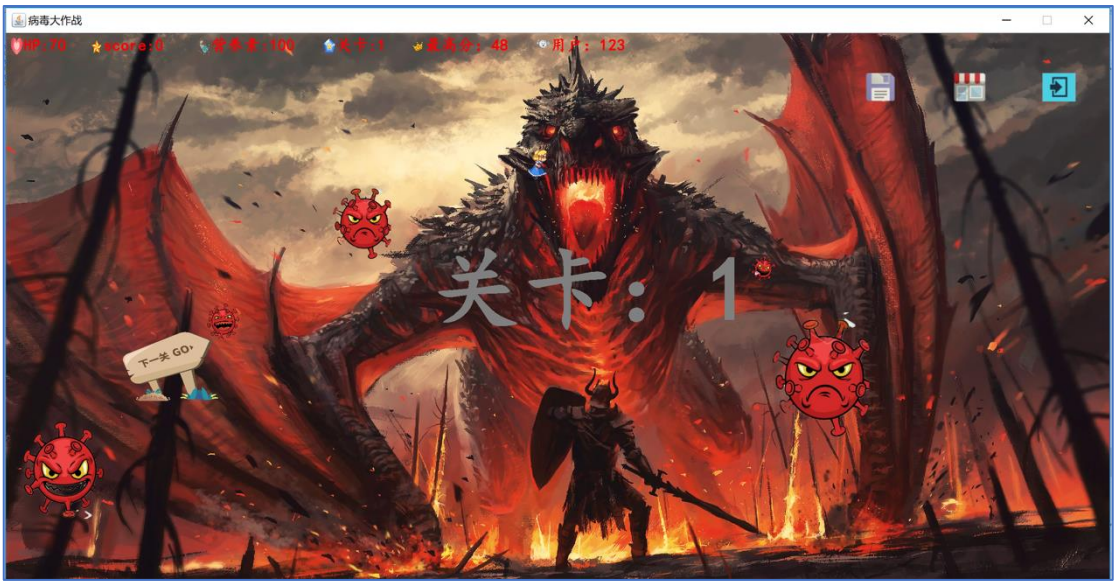

图 46 游戏界面

（1）游戏内信息分别显示玩家血量、得分、营养素、关卡、最高分、游戏进度以及用户名。

（2）游戏内选项为存档按钮、商店按钮以及返回按钮。

- 点击存档按钮可以存储游戏进度
- 点击商店按钮进入游戏内商店界面
- 点击返回弹出退出游戏框弹出“是否存档并退出游戏”的弹窗

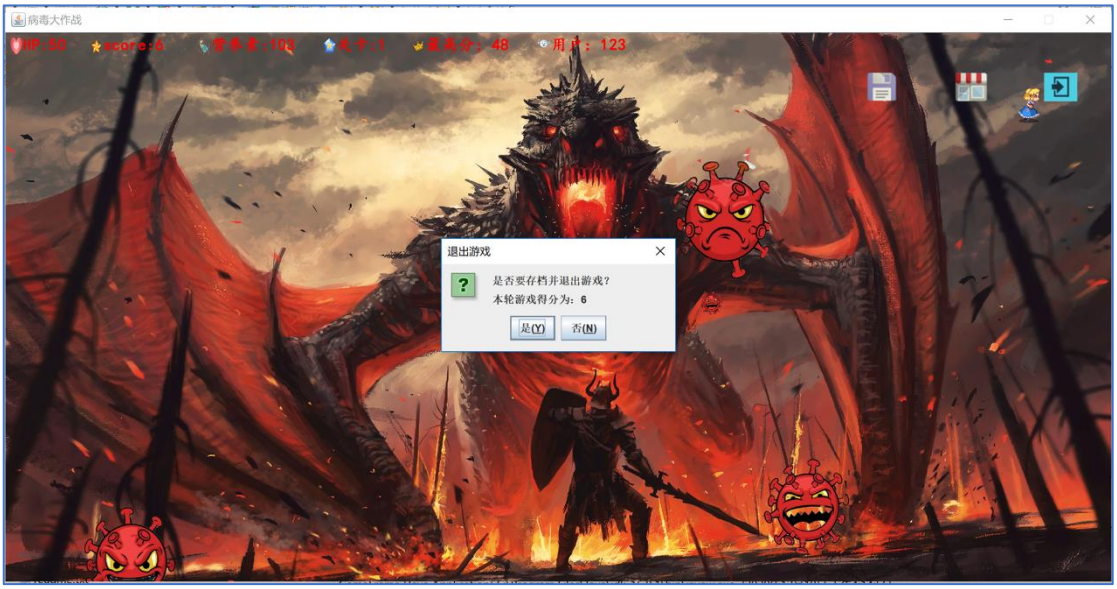

图 47 退出游戏

（3）本游戏通过鼠标的移动来移动角色，靠近比角色等级低的病毒则可以吞噬，角色同时增加 2 积分。当玩家操控游戏角色碰到比角色等级高的病毒则减少 5 点血量。玩家同时可以在商店中购买商品来提高游戏角色的生存能力。

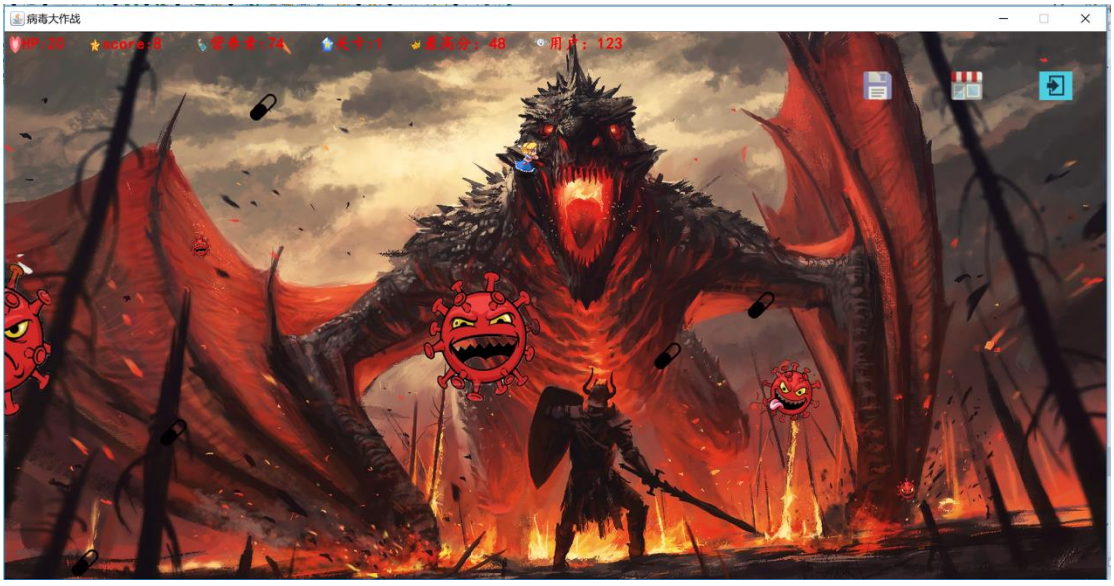

图 48 出现药丸

5.5 商店界面

通过在游戏界面中点击商店按钮，即可进入商店界面。游戏界面主要由四个商品和一个返回按钮组成。

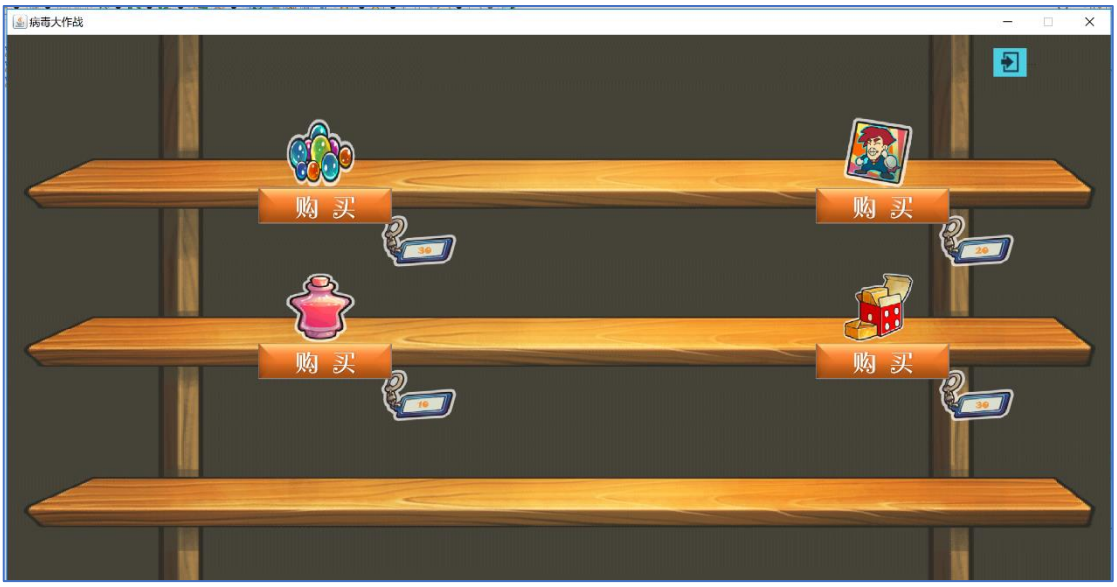

图 49 游戏界面

商店按钮的主要功能是为游戏中的角色提供游戏内的道具购买的功能，每一个道具拥有不同的游戏效果。

（1） 药丸：游戏中随机出现药丸，玩家可以吞噬药丸，每粒药丸增加 5 点血量，药丸最多可以吞噬 5 次，花费 30 营养素。

（2） 护盾：购买后为游戏角色增加五次护盾免除伤害，花费 20 营养素。

（3） 血包：购买后为游戏角色增加 10 点血量，花费 10 营养素。

（4） 随机福袋：购买后为游戏角色增加 X 点的血量值，0≤X≤50，花费 30 营养素。

5.5 游戏结束

当游戏角色成功吞噬一定的病毒，获得一定数量的积分时，则进入下一关卡。

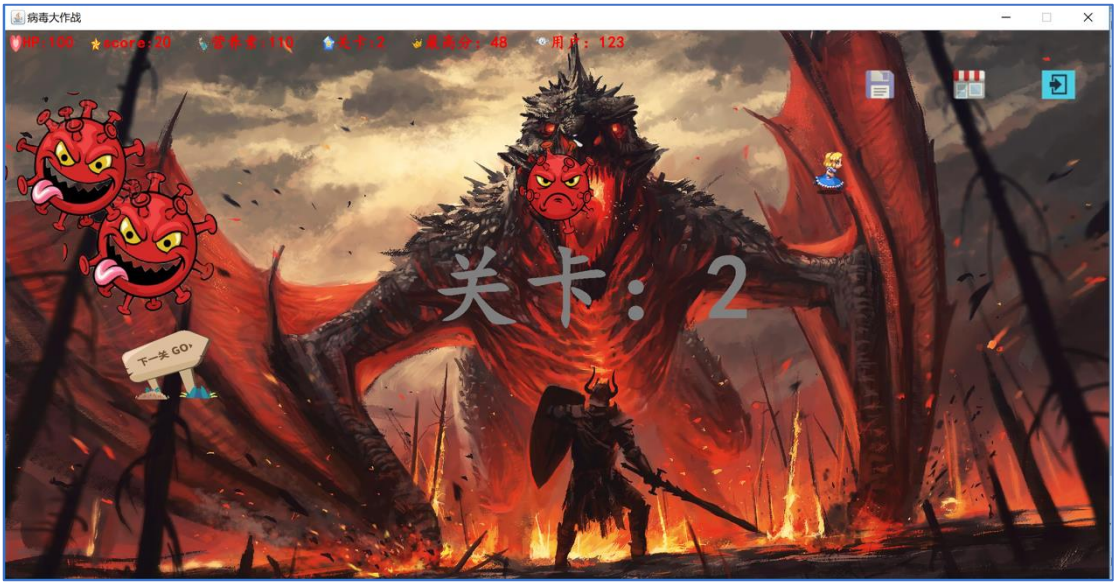

图 50 下一关

当游戏角色的血量降为 0 时，游戏结束，游戏界面清空所有内容，并显示“GAME OVER”的字样。

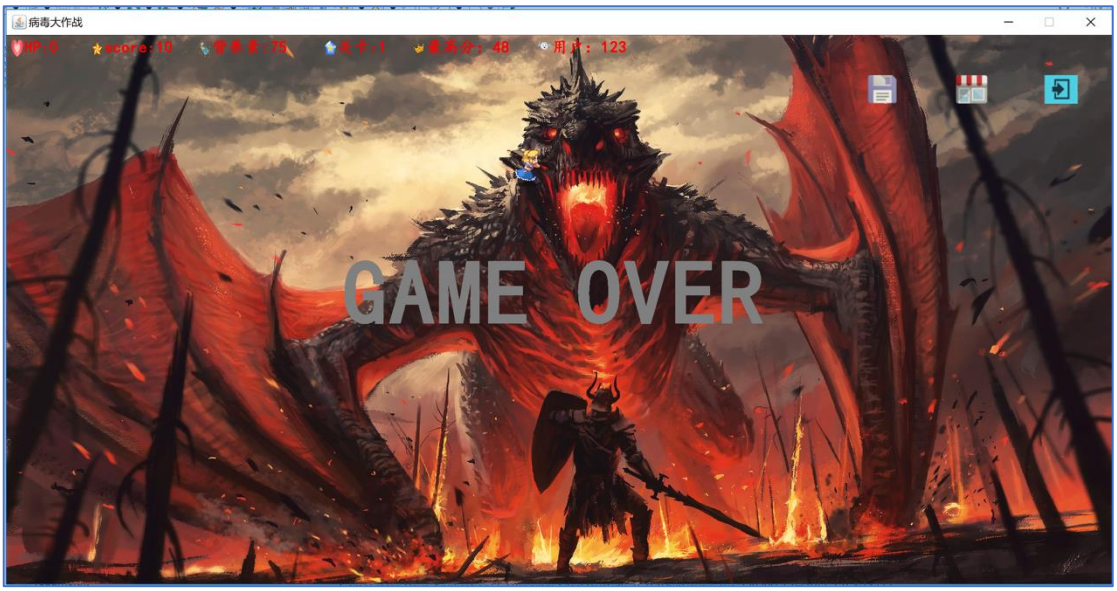

图 51 游戏结束

六、实验心得与体会

6.1 遇到的问题：

问题一：

界面上的按钮会闪烁或者只有当鼠标滑过按钮的位置，按钮才能正常显现出来。

解决方法：

加上 this.setLayout(null);和 super.paintComponents(g);两句，第一句使得按钮不会采用默认的流式布局，所有的组件可以用 setBounds()来控制大小和位置；第二句使得背景图片位于最底下的一层，不会将组件给覆盖。

问题二：

用于切换界面的按钮只能点进去一次，返回后不能再次进入，比如：注册按钮和商店按钮等。

解决方法：

以商店为例，在监测“是否点击游戏界面中的商店”这一线程中，将 break;换成 shopFrame.setIsCloseBoolean(false);这样第一次点击游戏界面中的商店按钮后就不会跳出线程，而只是将商店的状态设为关闭。当再次点击时，商店的状态又会变成打开，这样就能实现重复进入商店，其他部分的重复进入问题也以同样的方法解决。

问题三：

遇到空指针异常（null pointer exception）。

解决方法：

一般是图片的获取路径出了问题，要仔细核对图片的路径和格式，不随

意删图。

问题四：

有几个类的属性和方法都是相似的，比如 Virus、Boom、Viruspro 等，这些类中有大量重复的代码。

解决方法：

构造一个父类 Father，让 Virus、Boom、Viruspro 这些类继承这个父类，能大大减少代码的冗余。

问题五：

一开始的游戏太过于简单，游戏的趣味性和挑战性不强。

解决方法：

一点点修改游戏的初始参数，使得病毒出现的数量和频率合适，第二关和第三关在第一关的基础上加上更加厉害的病毒，比如第二关除了第一关的病毒外，还加上了碰到就会变为初始大小的病毒，第三关又在第二关的基础上加上了一碰到就会死的病毒，使得游戏的趣味性大大增强。

问题六：

游戏只有输了之后才能重新开始新游戏，如果没输只能继续游戏。

解决方法：

在监听点击“继续游戏”按钮的线程中，将 rola 和 virus 等重置为初始值，然后再调用 action()方法，这样就能实现继续游戏。

问题七：

一开始将普通病毒、流感病毒、药丸和新冠病毒分别创建了一个线程让它们产生和移动，但是运行之后它们不能正常出现。

解决方法：

由于普通病毒、流感病毒、药丸和新冠病毒的属性和方法类似，把他们都放在一个线程中之后就能够正常出现了。

6.2 项目亮点

6.2.1 游戏的趣味性：

该游戏是一个吞噬类的休闲小游戏，考虑到如果游戏过程中只是吞噬病毒的话，游戏会很无聊。所以，我们在原有普通病毒的基础上增加了流感病毒和新冠病毒，碰到流感病毒，玩家的大小会变为初始状态，碰到流感病毒的话，玩家会立即死亡，除此之外，我们还添加了能够购买各种道具的功能，大大增加了游戏的难度，也增强了游戏的趣味性。

6.2.2 视频播放：

在程序运行之后，会跳出一个有关抗击新冠病毒的视频，让玩家了解本游戏的设计理念，符合游戏主题。

6.2.3 帮助界面：

在预界面上有一个“帮助”按钮，点击“帮助”按钮会出现一个帮助界面，上面描述了不同按钮的功能，可以帮助玩家更快的熟悉游戏。

6.2.4 继续游戏和新游戏：

玩家退出游戏并存档后再次进入游戏可以选择“继续游戏”或者“新游戏”，继续游戏

6.3 小组成员及分工

6.3.1 小组成员：

6.3.2 小组分工：

七、总结

老师发布任务以来，从前期的茫然无措，到中后期的紧赶慢赶，我们既经历了一无所知的挫败，也经历了程序顺利运行的喜悦，更经历了修改 bug 烦躁与不安，期间面临了很多的纠结与取舍，但最终我们还是克服了种种困难，顺利完成了这次的大作业，但是我们明白这个游戏还存在很多可以改进的地方，我们希望在日后能提高自己的编程水平，写出更高质量的程序。