**设计与开发文档**

**题目：**基于Swing和MySQL的Crazy Snake游戏设计

目录

[一、 需求分析 1](#_Toc134966148)

[**1.1、开发环境** 1](#_Toc134966149)

[**1.2、功能需求** 1](#_Toc134966150)

[二、 系统分析与设计 1](#_Toc134966151)

[**2.1、本程序需解决的关键技术问题** 1](#_Toc134966152)

[**2.2、总体功能图** 2](#_Toc134966153)

[**2.3、类图** 3](#_Toc134966154)

[**2.4、程序流程图** 4](#_Toc134966155)

[三、 程序设计与实现 9](#_Toc134966156)

[**3.1、项目文件组织结构** 9](#_Toc134966157)

[**3.2、程序设计** 10](#_Toc134966158)

[**3.3、程序实现** 12](#_Toc134966159)

[四、 系统测试 13](#_Toc134966160)

[五、 总结与展望 14](#_Toc134966161)

[六、 作品效果 15](#_Toc134966162)

[七、 代码 21](#_Toc134966163)

# 需求分析

## **1.1、开发环境**

操作系统及版本：Windows 11

开发工具及版本：IntelliJ IDEA 2021.2.1

## **1.2、功能需求**

1．使用GUI组件，实现游戏窗口。进入游戏后为菜单界面。

2．游戏分为单人模式和双人模式两种。单人模式需要一人控制，通过不断吃到食物来获得分数，同时也会增加蛇身长度，碰到墙壁和自己就会游戏失败；双人模式需要两人控制，吃到食物的一方可以增加身体长度，碰到墙壁或自己或互相碰撞都会使游戏结束。

3．游戏要实现音乐播放功能。在菜单页面可以开启或关闭音乐。可以切换音乐歌曲。

4．游戏菜单有游戏帮助和关于按钮，可以点击打开。

5．单人模式要有登入系统，用于接收玩家信息。

6．游戏要实现最高分记录的功能。通过数据库读写，记录最高分和用户信息。

7．游戏中实现暂停和继续和重新开始的功能。在游戏进行中可以暂停或继续，或选择重新开始。

# 系统分析与设计

## **2.1、本程序需解决的关键技术问题**

1．窗口、子窗口的构建问题

2．界面的设计和切换问题。

3．按键监听，组件监听的问题

4．蛇身的构造（两条蛇的构造）

5．蛇的移动（两条蛇的移动，键盘控制移动）

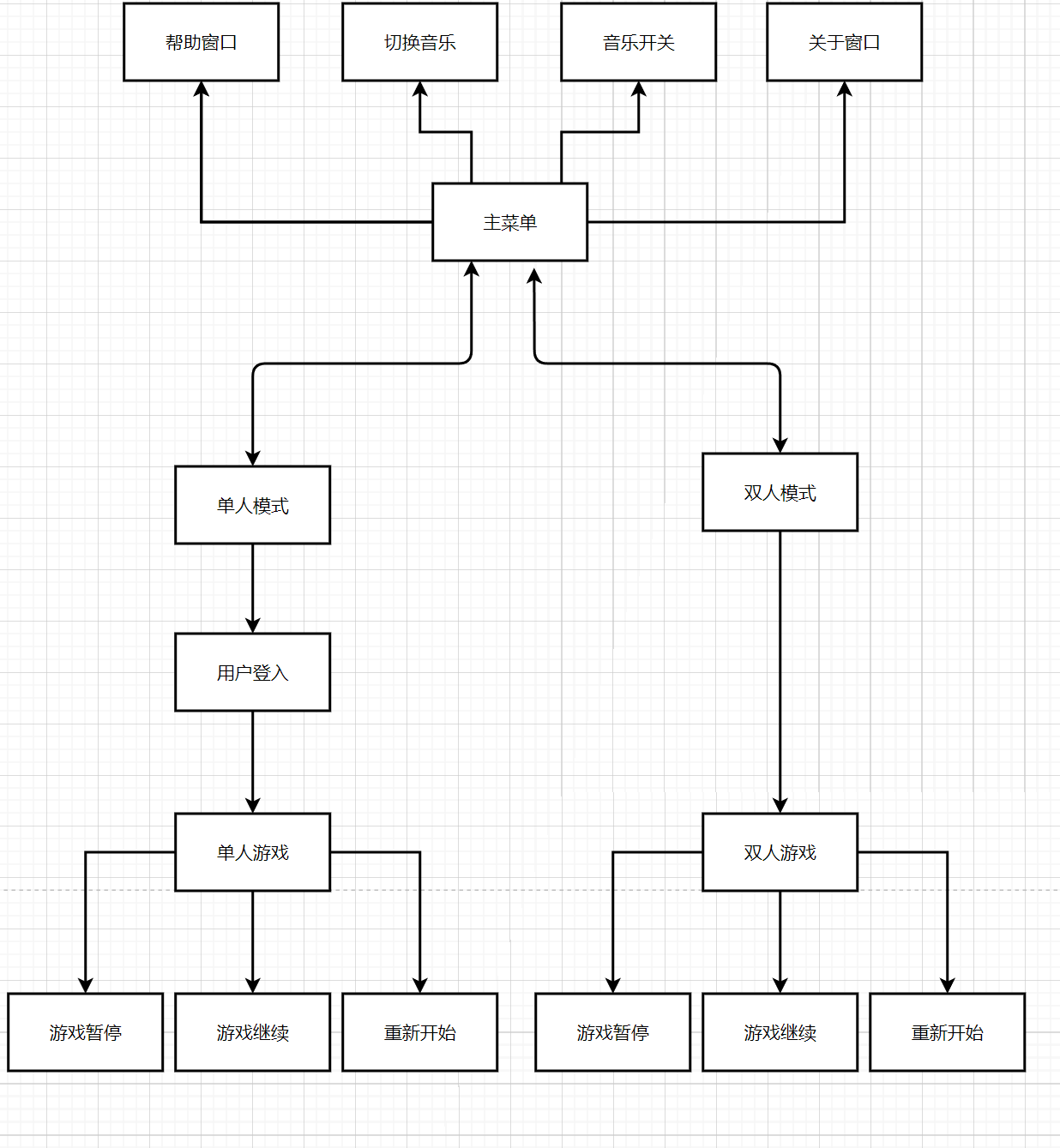
6．食物的构造（随机出现且不能出现在蛇身上）

7．游戏结束的条件判断（撞墙或咬到自己，双人模式要考虑互相碰撞）

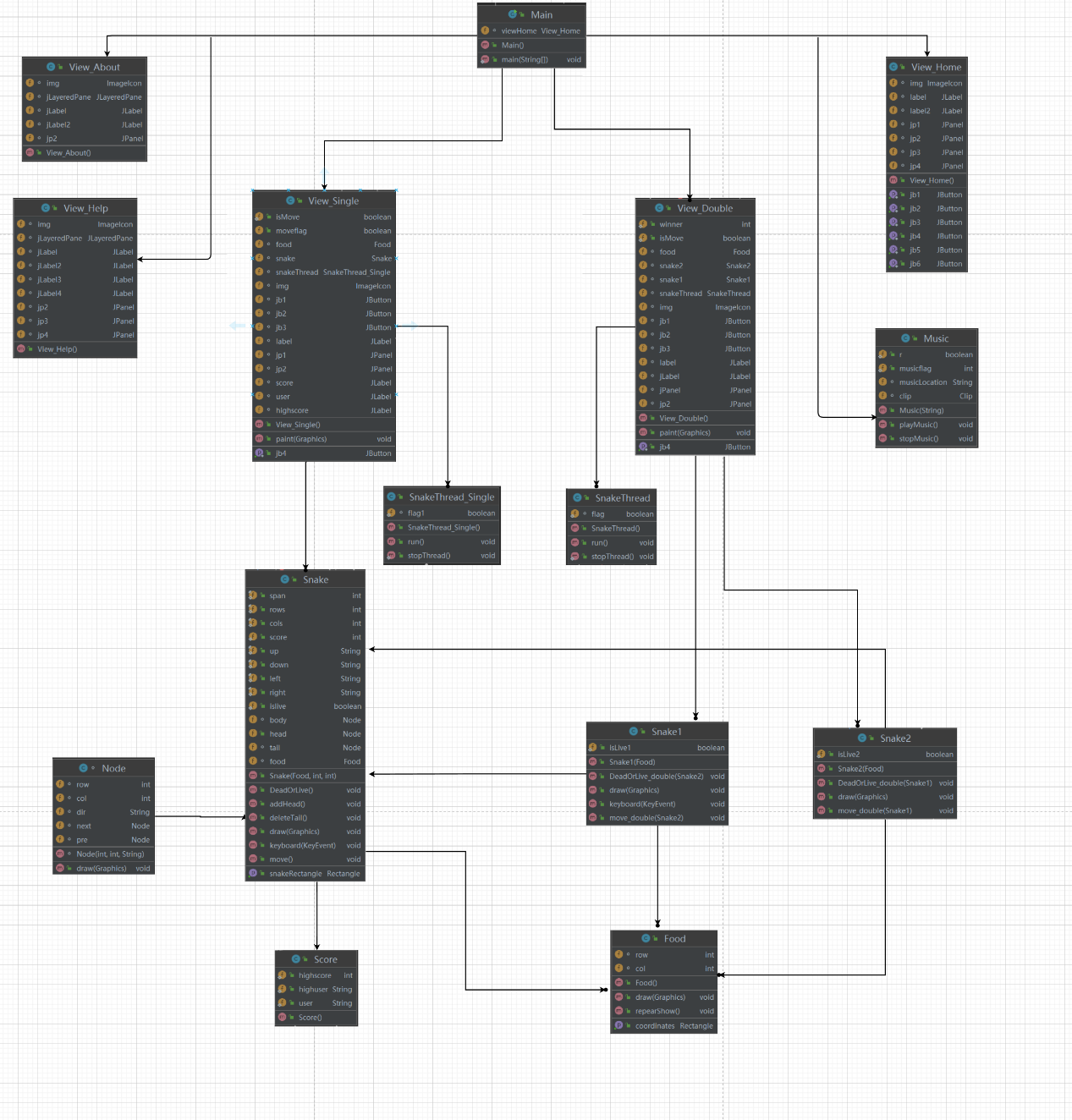
8．游戏数据的存储（多玩家的游戏记录）

9．音乐的切换、播放与关闭。

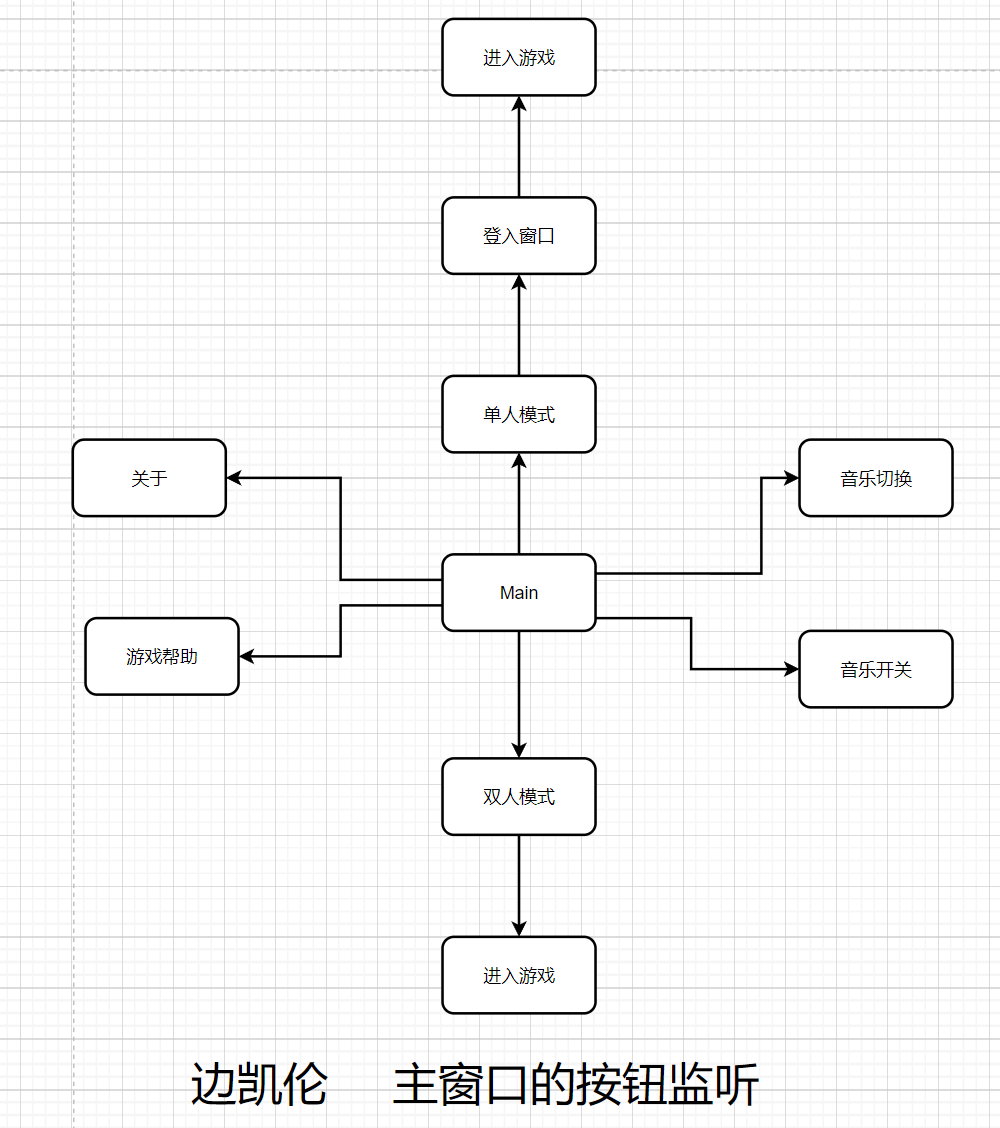
## **2.2、总体功能图**



## **2.3、类图**



## **2.4、程序流程图**



主窗口通过切换不同面板实现界面切换，主菜单界面有许多按钮，功能为：单人模式、双人模式、帮助窗口、关于窗口、音乐开关、音乐切换。

每个按钮都有按键监听，点击后会执行相应操作：

1．点击单人游戏，首先弹出登入窗口，填写信息后进入游戏，将单人游戏面板放置到主窗口，实现界面切换。

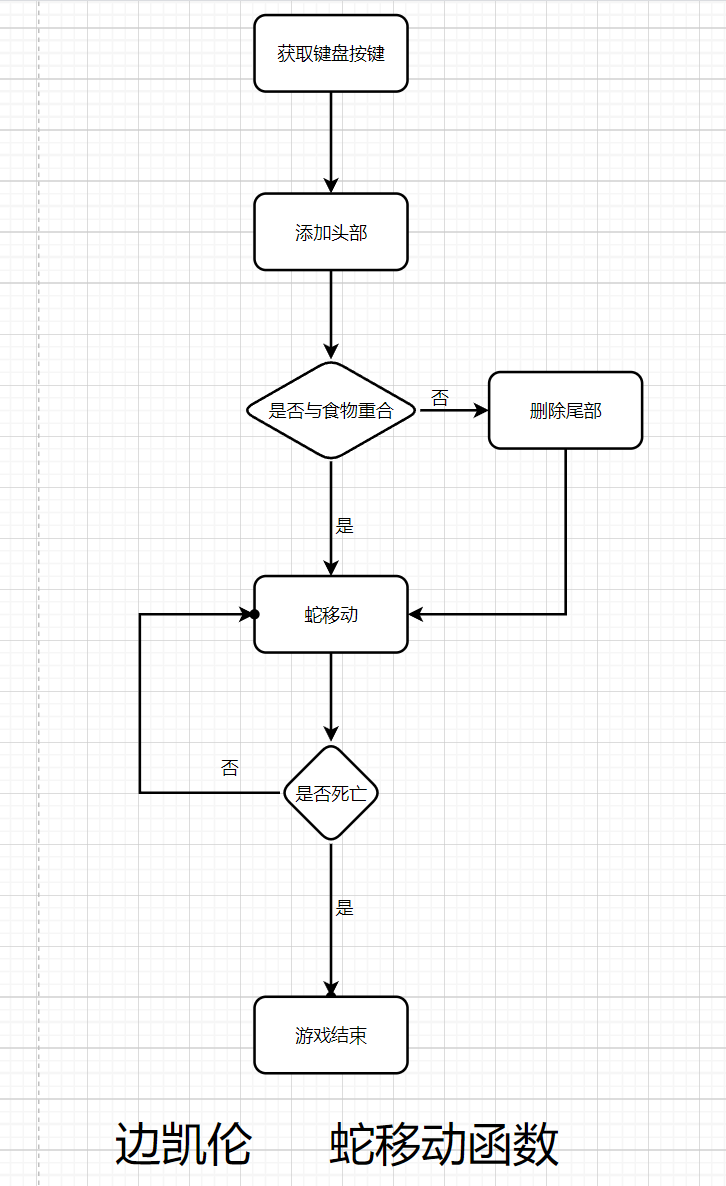
2．点击双人游戏，将双人游戏面板放置到主窗口，实现界面切换。

3．点击游戏帮助，弹出子窗口，显示游戏帮助信息。

4．点击关于，弹出子窗口，显示游戏关于信息。

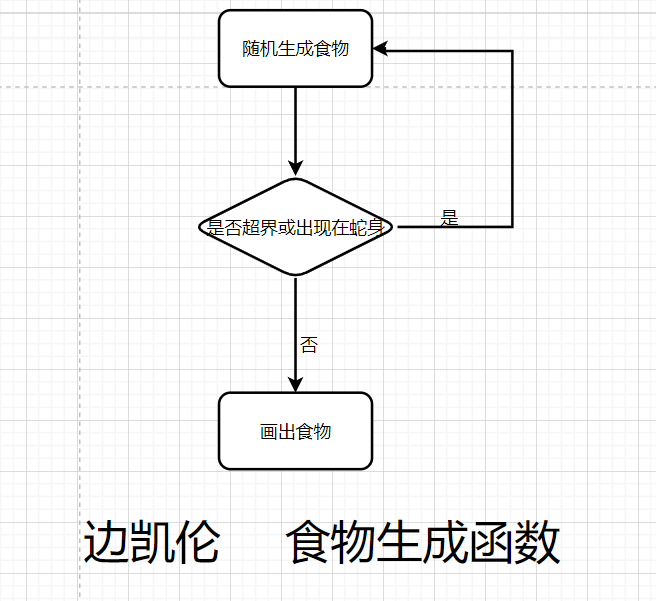
5．点击切换音乐，音乐换曲。

6．点击音乐开关，判断音乐播放情况，实现音乐开关功能。

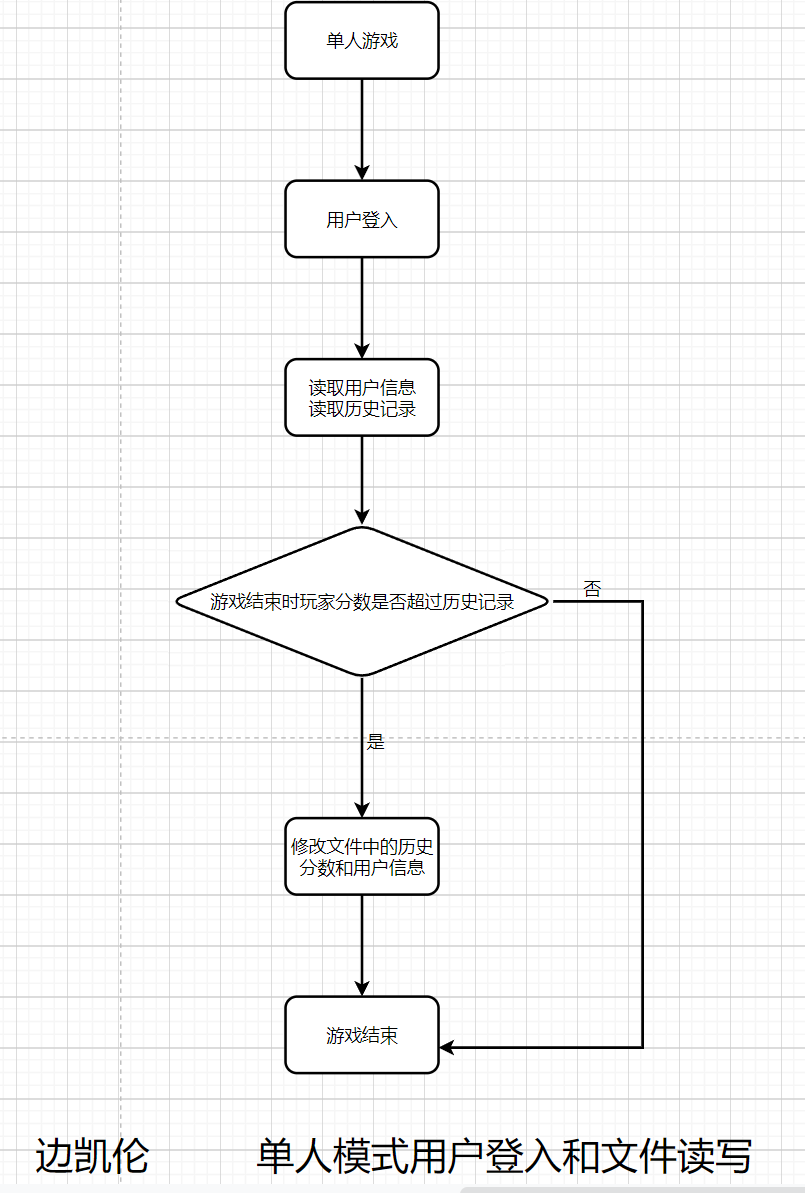


蛇移动函数，通过键盘监听获取按键信息，判断移动方向，向指定方向添加头部，如果头部位置有食物，即吃到食物，则继续移动，蛇长度边长。如果没有食物，即未吃到食物，则删除蛇尾，蛇长度不变。

每移动一步判断蛇是否死亡（撞到自己、撞到墙壁、多人模式撞到对方），死亡后结束游戏，反馈不同游戏结果。

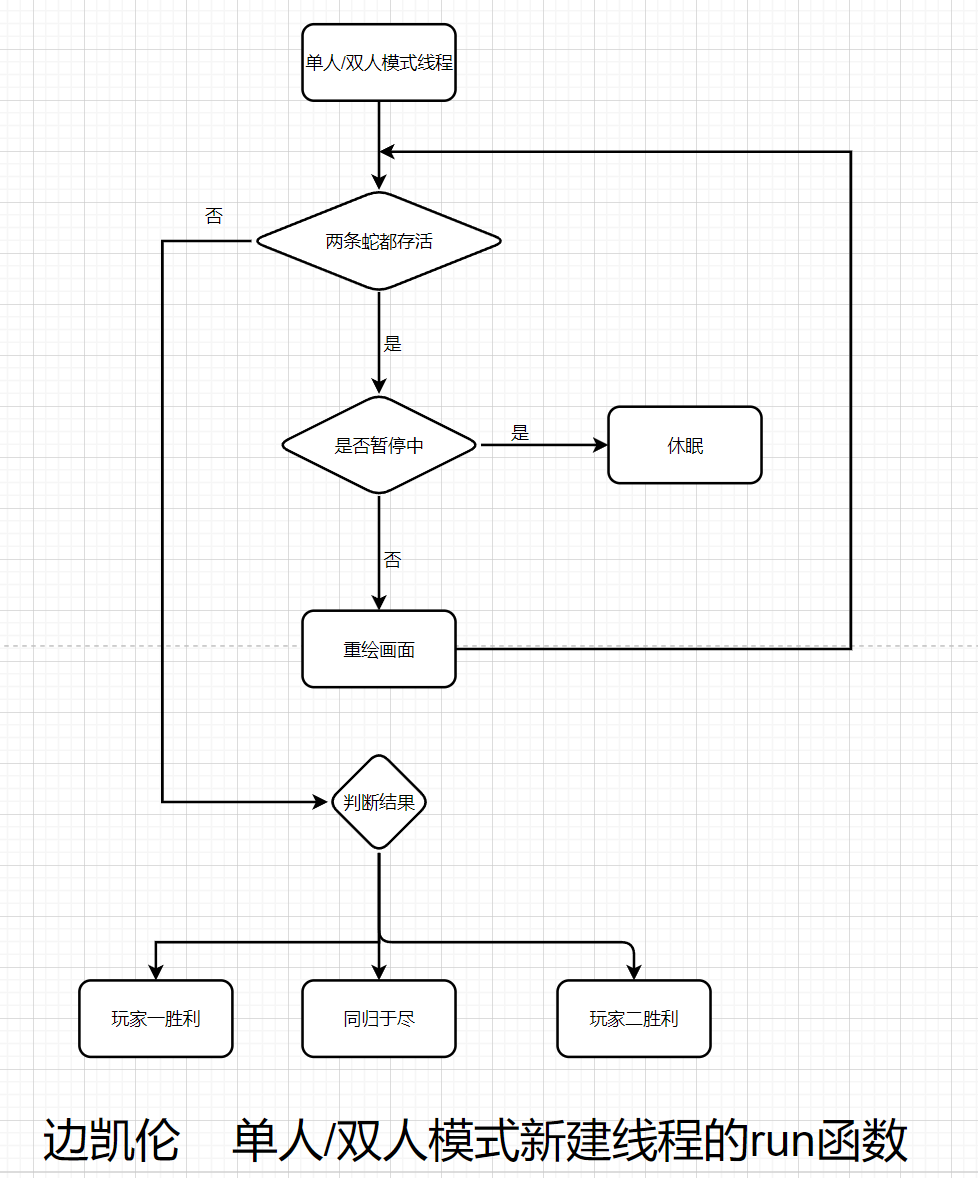


食物生成函数，先随机生成食物位置，判断食物位置是否合理（没有出现在边界外，没有生成在蛇身上），若合理则画出食物，若不合理则重新生成。



在主菜单点击单人模式后，弹出登入窗口，输入用户信息，将用户信息显示在游戏界面上。读取历史记录和得分者信息，也显示在游戏界面上。

生成游戏界面，在游戏结束后判断该用户得分是否高于历史记录，若打破记录，则修改历史记录和获得者信息。

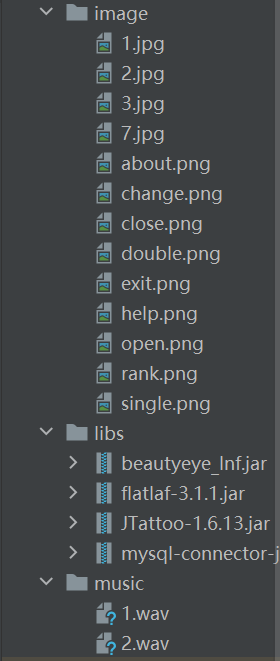
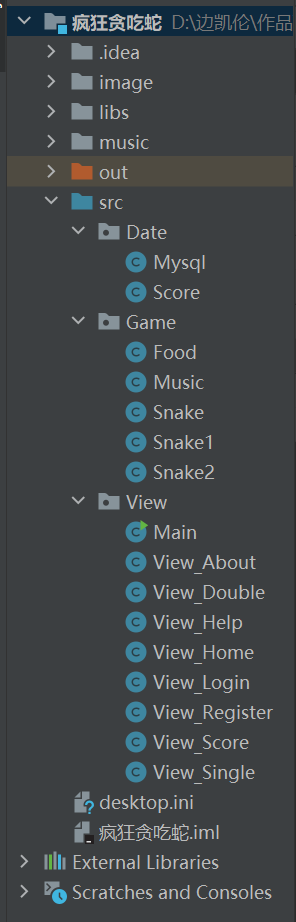


单人/双人模式的另一线程的run函数。当蛇存活且移动时，先重绘蛇身，再休眠一段时间，实现蛇移动的视觉效果。如果游戏暂停，则不执行重绘。

游戏结束之后会判断结果，双人模式存在玩家一胜利，玩家二胜利，同归于尽等结果。

# 程序设计与实现

## **3.1、项目文件组织结构**



项目名：Crazy Snake

包名：Date，Game，View

类名：如图所示

其他文件：图片文件 1.jpg，2.jpg，7.jpg等

音乐文件 1.wav，2.wav

数据库文件 user\_t.db

文件关系如图所示

## **3.2、程序设计**

* 1. Date包，与游戏数据和得分信息有关。
     + 1. Score类

实现数据库读写功能，用户登入时读取历史记录和历史记录获得者信息。游戏结束时判断用户得分是否打破历史记录，判断是否改写历史记录和获得者信息。

其中包含全局静态变量highscore、highuser、user分别表示历史记录、历史记录获得者、当前用户。

* + - 1. Mysql类

实现MySQL数据库的连接、读取、更新、插入、关闭等功能。

* 1. Game包，与游戏逻辑有关。

Food类

用于创建食物，通过随机数生成食物坐标，判断食物位置是否合理（不超边界，不出现在蛇身），则绘制食物。

1. Music类

实现音乐播放功能，构造函数用变量代替，实现播放多首歌曲的功能。包含playMusic()、stopMusic()函数，实现音乐播放和暂停。

全局变量musicflag用来判断正在播放哪首音乐。

1. Snake类（包含内部类Node）

涉及蛇身构造、蛇移动、死亡判断等功能，通过内部类Node实现双向链表功能构造蛇身，通过draw（）函数绘制蛇身。通过键盘监听获得键盘按键，控制蛇移动方向，向移动方向执行addhead（）添加头部，再执行move（）函数判断蛇是否与食物坐标重合，若两者重合，则重绘蛇身，若不重合，则先执行deleteTail（）删去尾部，再绘制蛇身。每走一步通过DeadOrLive判断蛇是否死亡。

成员变量score用来记录蛇的得分，全局静态变量islive判断蛇是否死亡。

1. Snake1类

该类为Snake的子类，重载了父类的keyboard（）方法，将用户一的按键改为W，S，A，D，添加了子类成员islive1判断蛇一存活情况。

重写了draw（）方法和构造方法改变颜色和位置，重载了移动和死亡判断函数，适用于双人模式。

1. Snake2类

该类为Snake的子类，添加了子类成员islive2判断蛇二存活情况。

重写了draw（）方法和构造方法改变颜色和位置，重载了移动和死亡判断函数，适用于双人模式。

* 1. View包，与游戏界面有关。

Main类

该类继承自JFrame类，实现了游戏的主窗口。通过将不同的界面放到此窗口实现界面的切换。

* 1. View\_About类

该类继承自JDialog类，实现了游戏关于子窗口。此类包含两个标签，一个放置背景图片，一个放置文字。包含一个面板，放置标签。包含一个层级面板，底层放置背景图片，上层放置文字面板。

* 1. View\_Double类（含内部类SnakeThread）

该类继承自JLayeredPane类，实现了双人模式游戏界面。该界面右侧为文字和按钮，实现暂停、继续、重新开始、返回等功能。该类包含内部类SnakeThread，实现两条蛇的移动。

在该类中添加了键盘监听，实现两条蛇的移动。成员变量winner用于判断游戏结果。静态全局变量isMove用于判断是否在暂停还是在游戏。

* 1. View\_Help类

类继承自JDialog类，实现了游戏帮助子窗口。此类包含文字标签和背景图片标签。用面板存放这些标签，用分层面板分层放置这些面板。

* 1. View\_Home类

该类继承自JLayeredPane类，实现了游戏的菜单界面，为所有界面的中转站。在此面板上有多个按钮。最下方有单人模式和双人模式的按钮，点击可进入游戏。（进入单人模式前会弹出登入窗口）。在界面右侧为帮助按钮，关于按钮，音乐切换按钮和音乐开关按钮。界面最中间为软件名称和背景。

在单人模式和双人模式中，点击返回按钮时，切换至此界面。

* 1. View\_Single类

该类继承自JLayeredPane类，实现了单人模式游戏界面。此界面右侧为用户信息和当前得分，以及历史得分和得分者信息。还有按钮暂停、继续、重新开始和返回。

静态全局变量isMove用于判断是否在暂停还是在游戏。成员变量moveflag用于判断是否已按键，实现键盘监听只监听一次，防止出现蛇逆向的漏洞。

## **3.3、程序实现**

首先创建一个窗口，放置主界面的层级面板。此界面作为枢纽，可以打开帮助子窗口、关于子窗口，可以切换音乐和开关音乐。

如果点击单人游戏，先弹出登入窗口，然后在主窗口放置单人模式的层级面板，进入单人模式界面，若点返回可以在主界面放置主界面的层级面板。

如果点击双人游戏，在主窗口放置双人模式的层级面板，进入双人模式界面，若点返回可以在主界面放置主界面的层级面板。

游戏中会通过新建的线程，每走一步先休眠一段时间，再调用repaint（）画出蛇身，实现蛇移动的视觉效果。其中paint（）函数会调用Snake或其子类Snake1、Snake2的move（）函数和draw（）函数，以及Food的draw（）函数。

**特色算法：**

1.使用双向链表实现了蛇身的构建。蛇身的每一个节点都用内部类保存，内部类中包含了前后两个节点。前后节点连接形成蛇身。

2. 使用了多线程，利用paint（）和repaint（）函数绘制蛇和食物，线程运行时先进行休眠，实现蛇移动的视觉效果。

3.使用Java Swing设计了GUI界面，用户可读性更高，操作更简单方便。利用多个层次面板实现了菜单与游戏界面的切换。

4.利用MySQL数据库的读写，实现用户信息存入和历史记录的读写。

5.使用层级面板，将背景和其他组件以及蛇身分离，界面更加友好。

6.使用异常类处理了可能存在的问题，防止程序意外崩溃。

**创新的地方：**

1．游戏实现了单人模式和双人模式两种模式的玩法。

2．单人模式加入登入系统，读取用户信息，玩家通过吃食物得分，最终判断得分与历史记录，从而进行历史记录的改写。单人模式更注重休闲性

3．人模式通过消灭对方取胜，吃到食物可以让身体边长，想方设法围住对手并让对方撞到自己取胜。双人模式更注重竞技性。

4．单人模式时，在键盘监听处加入判断，使蛇在没步内只能调转一次方向且不能逆向，消除了在一步内因快速点击键盘导致蛇头逆向的bug。

5．游戏界面设计友好，布局合理，界面切换流畅，功能性多。

6．游戏加入了音乐系统。可以切换音乐和开关音乐。

7．游戏结束判定结果时，充分考虑所有结果，并通过弹出子窗口给予不同的反馈信息。

# 系统测试

**1．测试目的：**

（1）通过与系统的需求定义作比较，发现软件与系统定义不符合或与之矛盾的地方。

（2）系统测试的测试用例根据需求分析来设计，并在实际使用环境中运行。

**2．测试范围：**整个程序

**3．参与测试人员：**边凯伦、李奕慧歆

**4．测试花费时间：**3小时

**5．测试内容：**

**（1）功能测试**

功能测试主要是为了发现以下几类错误：

1）是否有不正确或遗漏了的功能？

2）功能实现是否满足用户需求和系统设计的隐藏需求？

3）能否正确的接受输入？能否正确的输出结果？

功能测试需要注意以下几点：

1）单一功能，例如打电话功能、听歌功能等；

2）功能组合，例如对手机测试，可以测试接听电话的时候是否可以查看信息、通讯录等；

**（2）安全性测试**

功能测试主要是为了发现以下几类错误：

1）各级用户权限划分是否合理？

2）错误和文件访问是否适当的被记录？

3）系统配置数据是否能正确保存，系统故障时是否能恢复？

**（3）GUI测试**

GUI测试是针对软件系统GUI界面进行的测试，是一种可优化性测试，测试内容有以下方面：

1）测试界面实现和界面设计的吻合情况；

2）确认界面处理的正确性。

3）简单界面元素，指功能和属性相对比较单一的界面区域。例如：一个菜单项、一个按钮、一个快捷键等；

4）界面组合元素，主要指一些复杂的界面元素。例如：工具栏，菜单栏、组合栏、表格等；

5）完整界面（窗口），指有一系列界面组合元素组成的能够完成一个完整的输入输出功能的界面属性组合，具有自己的视图。例如：对话框等。

# 总结与展望

**总结：**

1．利用JavaSwing，GUI界面非常友好，可在不同界面之间自由切换。（会熟练创建窗口，面板，按钮，标签，层次面板，可以熟练设置它们的属性，设置监听。也会之间从Object类中之间调用已经设计好的窗口）

2．游戏功能丰富，可以打开游戏帮助，打开游戏关于，切换音乐，开关音乐。共开发了两种模式：单人得分模式和双人对抗模式。游戏可玩性高，目前暂无bug。

3．面向对象编程，功能分布合理，类与类之间的关系明晰紧凑。充分使用了所学的知识（使用了内部类，异常类，多线程，文件读写字符流。体现了继承、多态和封装）

4．使用了数据结构知识，利用双向链表创建蛇身，代码简洁，可读性高。

5．花费了大量时间进行思考、编写和测试修改。对细节控制的比较完美。（a.单人模式为得分模式，进入单人模式之前会弹出登入窗口，读取用户信息。b.音乐切换和音乐开关设计合理，每点击一下都会做出相应反映，不存在切换音乐后再点开关会出现混乱的情况。c.单人模式打破记录、双人模式分出胜负之后会弹出不同的游戏结果。d.从游戏界面返回主界面后再进入游戏不会受上局影响，程序独立性强）

6.使用MySQL数据库，用户数据得以保存。可以通过排行榜查看用户最高记录。游戏结束后可以根据情况对数据进行更新或插入。

6.代码工程量大，代码书写规范、简洁。

**展望：**

1.更多创新的玩法：随着游戏技术的不断进步，游戏开发者可能会探索更多创新的玩法，例如实时对战、多人在线游戏、虚拟现实等等。这些新的游戏玩法将为玩家带来更加多样化的游戏体验。

2.更加智能化的敌人：未来的贪吃蛇游戏可能会更加智能化，敌人会更具攻击性和策略性，玩家需要不断提高自己的游戏技巧才能胜利。

3.更加多样化的游戏场景：游戏场景的多样化将可以为玩家带来更加多样化的游戏体验。游戏开发者可能会引入更多不同的场景，例如森林、沙漠、海洋等等，这些场景可以带来更多的挑战和乐趣。

4.更加智能化的 AI:未来的贪吃蛇游戏可能会引入更加智能化的 AI,AI 将可以更好地理解游戏规则和策略，并更具攻击性和策略性地与玩家对抗。

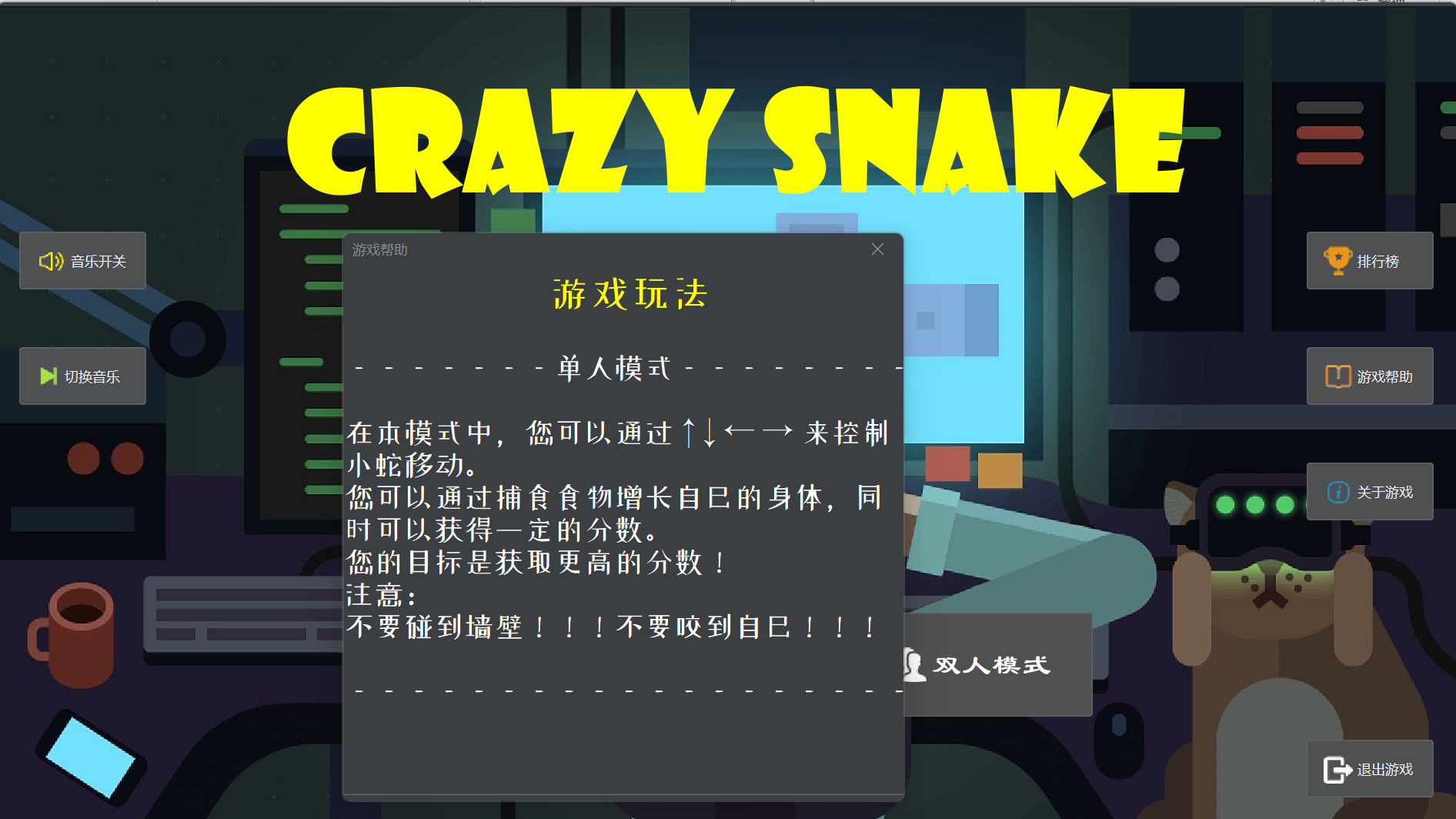
5.更加多样化的游戏设备：随着移动游戏的普及，未来的贪吃蛇游戏可能会更加多样化，支持更多不同的游戏设备，例如手机、平板电脑、智能手表等等。

总之，贪吃蛇游戏是一个非常经典的游戏，未来它可能会不断演变和发展，为玩家带来更加多样化的游戏体验。

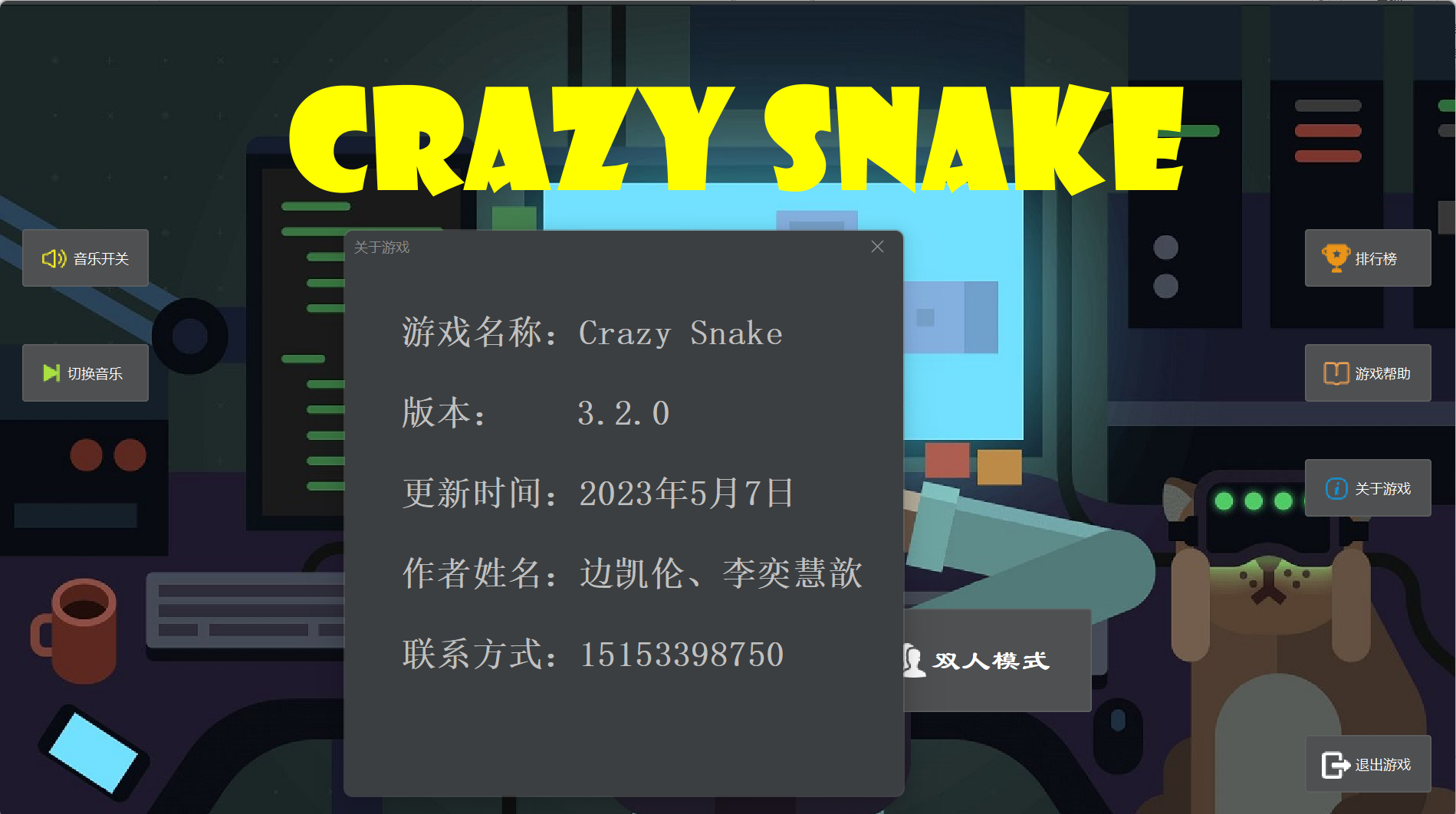
# 作品效果



**主界面**



**游戏帮助**



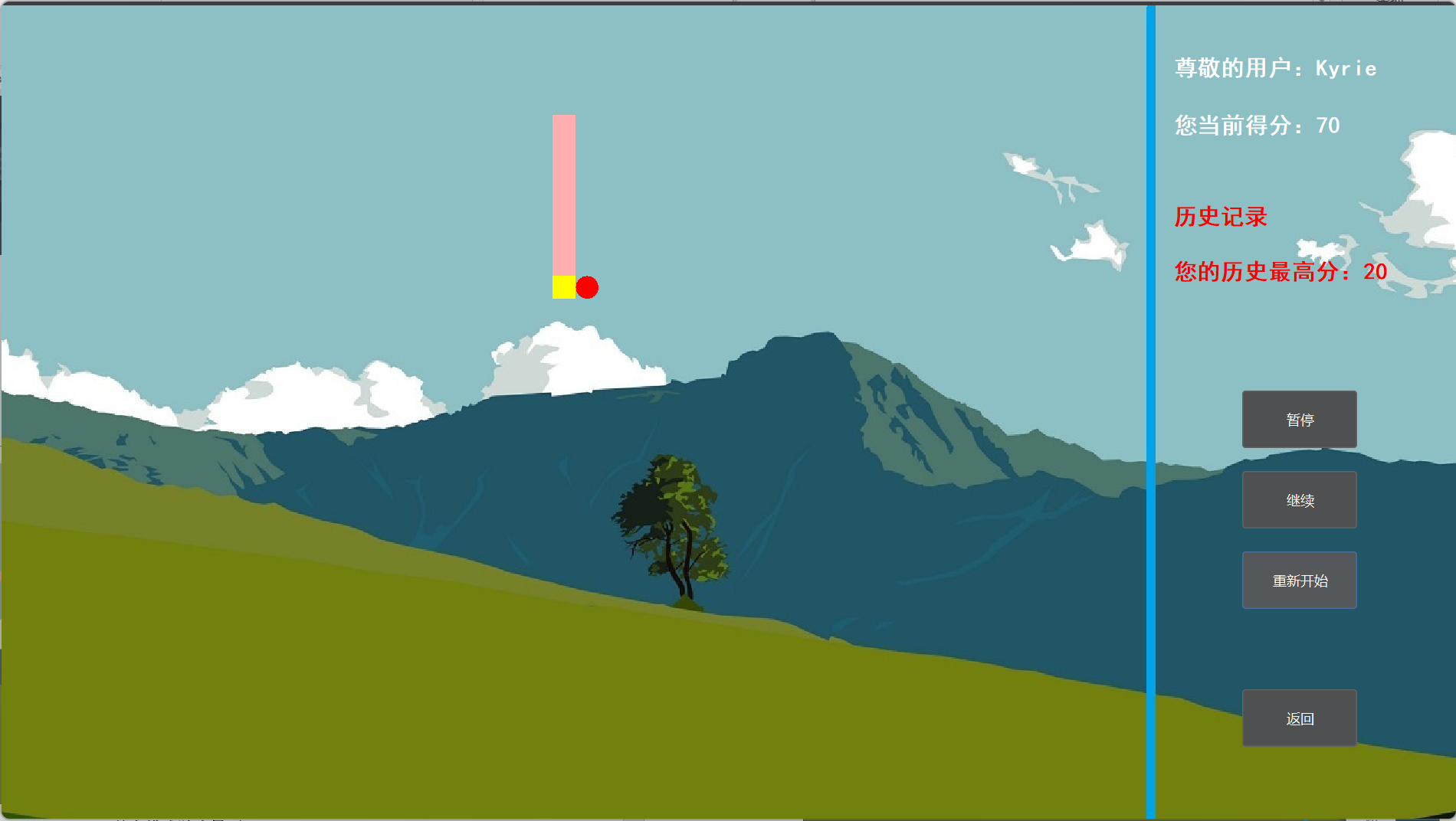
**关于游戏**



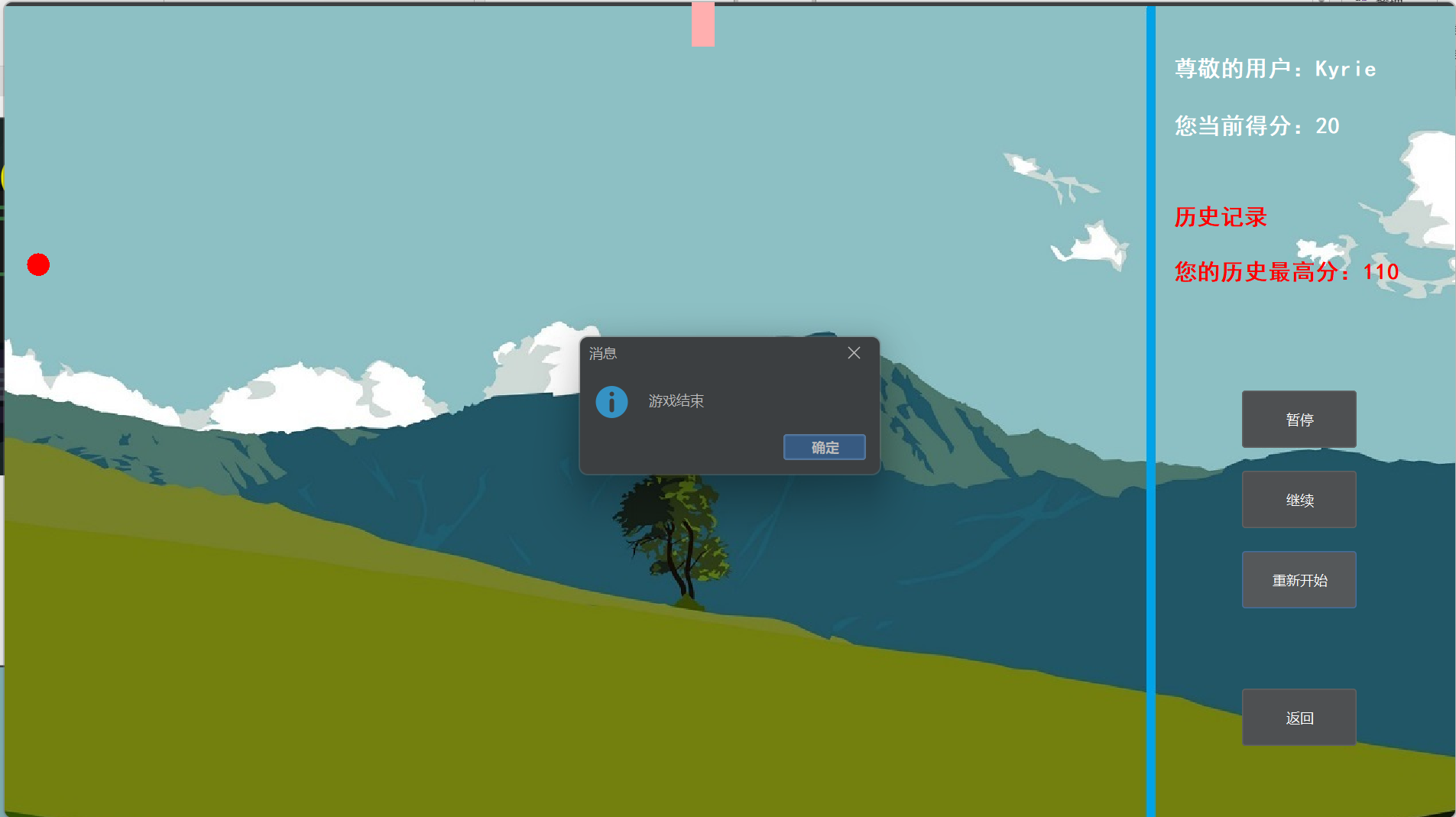
**单人模式登入**



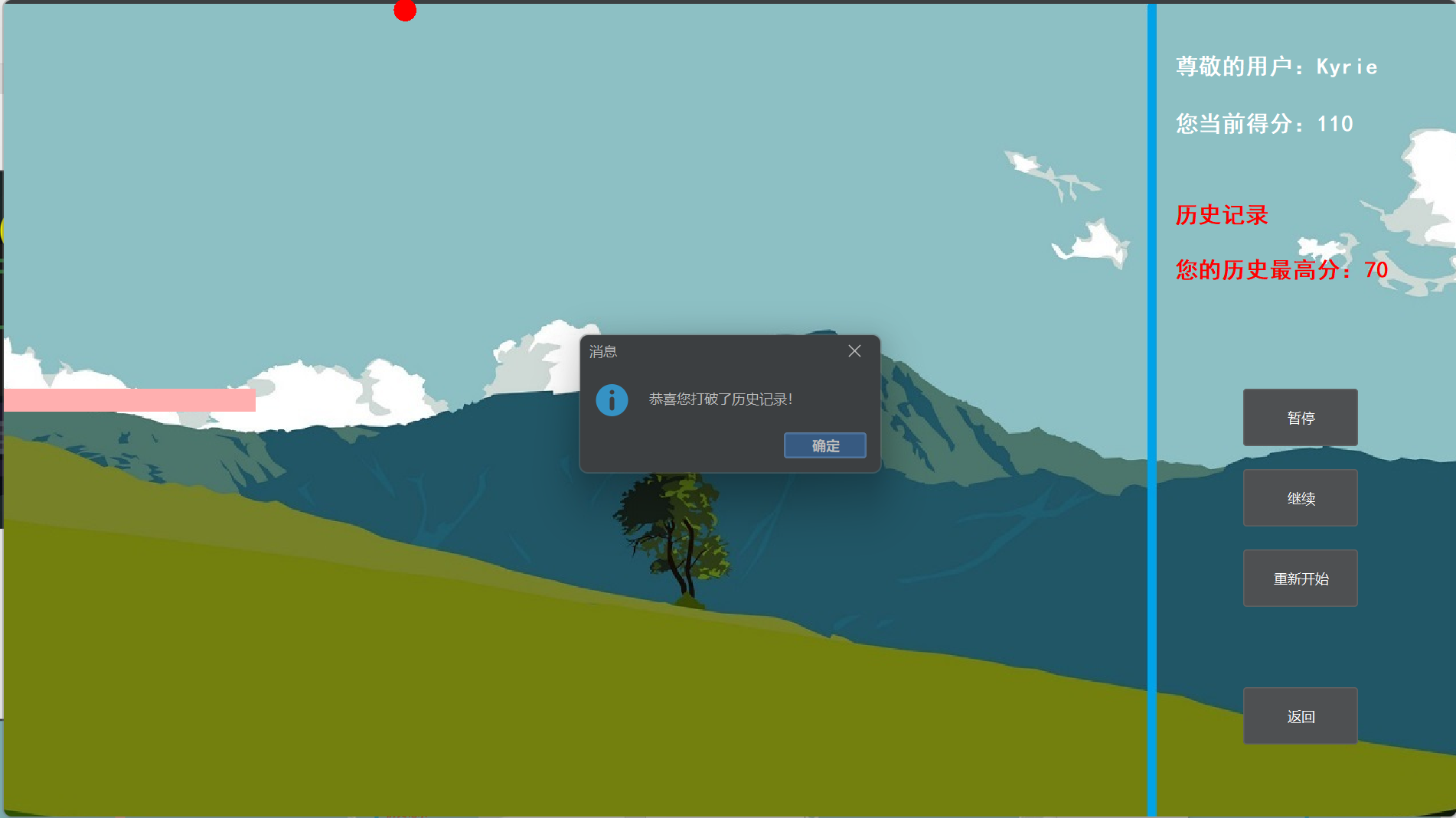
**排行榜**



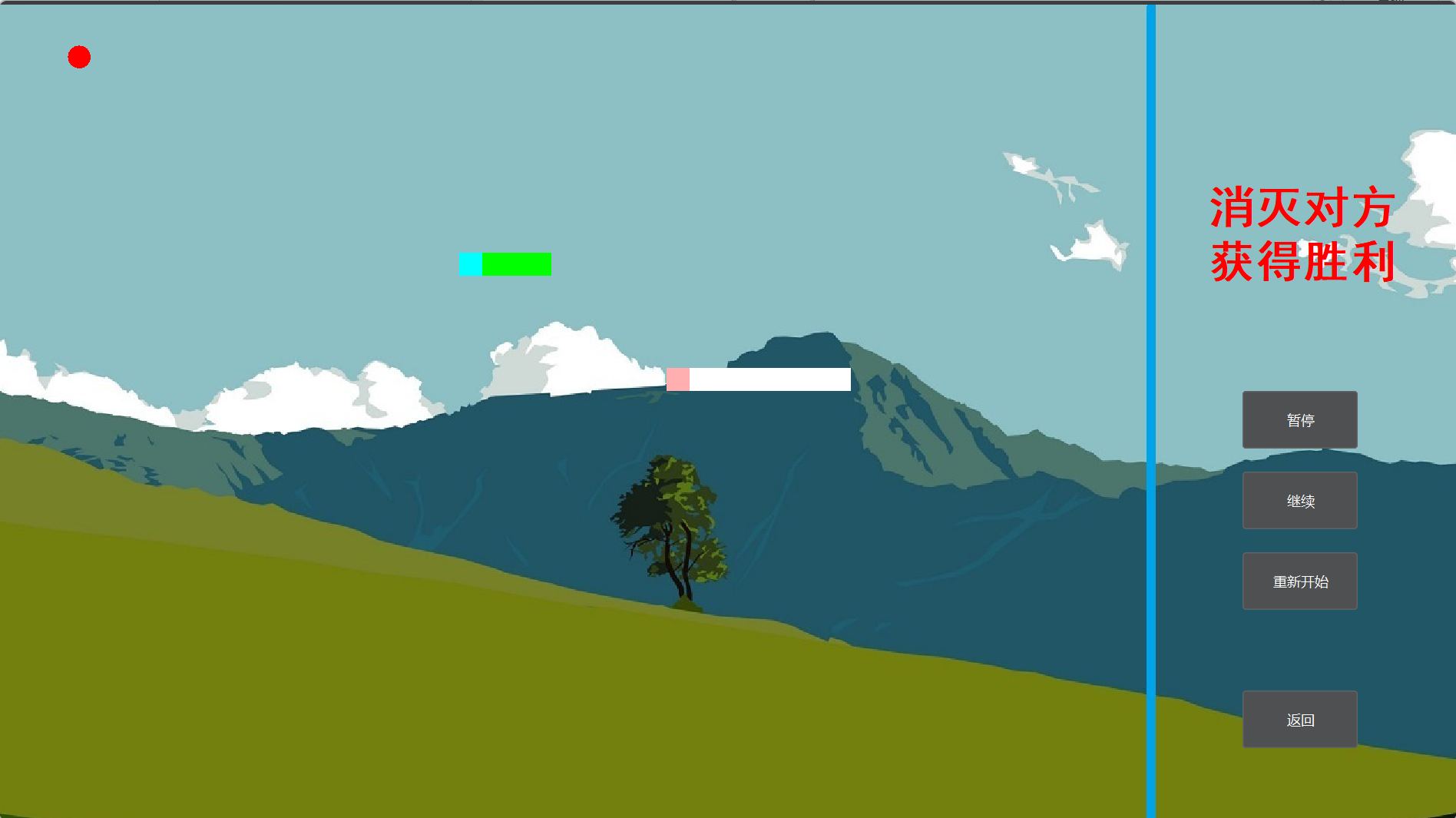
**单人模式游戏界面**



**单人模式游戏结束**



**单人模式打破记录**



**双人模式游戏界面**



**双人模式游戏结束**

# 代码

**Main.java**

package View;

import Date.Score;

import Game.Music;

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

//主窗口

public class Main extends JFrame {

View\_Home viewHome = new View\_Home();

public Main() {

Music music1 =new Music("C:\\Users\\15869\\Desktop\\疯狂贪吃蛇\\1.wav"); //音乐1

Music music2 =new Music("C:\\Users\\15869\\Desktop\\疯狂贪吃蛇\\2.wav"); //音乐2

music2.stopMusic();

this.setBounds(300, 100, 1280, 720); //设置窗口位置

this.setResizable(false);

this.setLayout(null); //设置窗口布局

this.setVisible(true);

this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

this.setLayeredPane(viewHome); //放置主界面

JButton single\_mode = View\_Home.getJb1(); //单人模式按钮

single\_mode.addActionListener(new AbstractAction() {

@Override

public void actionPerformed(ActionEvent e) {

Score.user = JOptionPane.showInputDialog( //用户登入窗口

null,

"输入你的名字:",

"用户登入",

3

);

new Score();

View\_Single viewSingle = new View\_Single();

setLayeredPane(viewSingle);

setVisible(true);

setFocusable(true);

requestFocus();

viewSingle.requestFocusInWindow();

}

});

JButton double\_mode = View\_Home.getJb2(); //双人模式按钮

double\_mode.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

View\_Double viewDouble = new View\_Double();

setLayeredPane(viewDouble);

setVisible(true);

setFocusable(true);

requestFocus();

viewDouble.requestFocusInWindow();

}

});

JButton single\_return = View\_Single.getJb4(); //单人模式返回主界面按钮

single\_return.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

View\_Single.SnakeThread\_Single.stopThread();

View\_Home viewHome = new View\_Home();

setLayeredPane(viewHome);

setVisible(true);

viewHome.requestFocusInWindow();

}

});

JButton double\_return = View\_Double.getJb4(); //双人模式返回主界面按钮

double\_return.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

View\_Double.SnakeThread.stopThread();

View\_Home viewHome = new View\_Home();

setLayeredPane(viewHome);

setVisible(true);

viewHome.requestFocusInWindow();

}

});

JButton help =View\_Home.getJb3(); //帮助窗口按钮

help.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

View\_Help view\_help=new View\_Help();

setVisible(true);

setFocusable(true);

requestFocus();

view\_help.requestFocusInWindow();

}

});

JButton changemusic =View\_Home.getJb4(); //切换音乐按钮

changemusic.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if(Music.musicflag==1) {

music1.stopMusic();

music2.playMusic();

Music.r=true;

Music.musicflag=2;

}

else

{

Music.musicflag=1;

music2.stopMusic();

music1.playMusic();

Music.r=true;

}

}

});

JButton music =View\_Home.getJb5(); //音乐开关按钮

music.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

if(Music.musicflag==1)

{

if(Music.r){

music1.stopMusic();

Music.r=false;

}

else {

music1.playMusic();

Music.r=true;

}

}

else {

if(Music.r){

music2.stopMusic();

Music.r=false;

}

else {

music2.playMusic();

Music.r=true;

}

}

}

});

JButton about =View\_Home.getJb6(); //关于窗口按钮

about.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

View\_About view\_about=new View\_About();

setVisible(true);

setFocusable(true);

requestFocus();

view\_about.requestFocusInWindow();

}

});

}

public static void main(String[] args) {

Main m = new Main();

} //主函数

}

**View\_About.java**

package View;

import javax.swing.\*;

import java.awt.\*;

import static javax.swing.JLayeredPane.DEFAULT\_LAYER;

import static javax.swing.JLayeredPane.MODAL\_LAYER;

//关于窗口

public class View\_About extends JDialog {

ImageIcon img = new ImageIcon("C:\\Users\\15869\\Desktop\\疯狂贪吃蛇\\1.jpg");

JLayeredPane jLayeredPane = new JLayeredPane(); //分层面板

JLabel jLabel = new JLabel(img); //放置图片的标签

JLabel jLabel2 = new JLabel(); //放置文字的标签

JPanel jp2 = new JPanel(); //放置标签的面板

public View\_About(){

this.setAlwaysOnTop(true); //设置子窗口的属性

this.setTitle("关于");

this.setBounds(600,300,500,500);

this.setVisible(true);

this.setLayout(null);

this.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

jLabel2.setText("<html><br>程序名称：疯狂贪吃蛇（Java版）<br><br>版本：　　3.2.0<br><br>完成时间：2022年6月7日<br><br>作者姓名：边凯伦<br><br>学号：　　8008121291<br><br>班级：　　计算机Ⅱ类2110班<br><html>");

jLabel2.setFont(new Font("宋体",Font.BOLD,30));

jLabel2.setForeground(Color.BLUE);

jp2.add(jLabel2);

jLabel.setBounds(0,0,500,500);

jp2.setBounds(0,0,500,500);

jLayeredPane.add(jLabel,DEFAULT\_LAYER); //将背景图片放底层

jLayeredPane.add(jp2,MODAL\_LAYER); //将文字面板放上层

jp2.setOpaque(false); //文字背景透明

this.setContentPane(jLayeredPane);

}

}

**View\_Double.java**

package View;

import Game.Food;

import Game.Snake2;

import Game.Snake1;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

import static Game.Snake.islive;

//双人模式

public class View\_Double extends JLayeredPane {

public static int winner = 0; //判断获胜方

public static boolean isMove = true; //判断游戏是否暂停

Food food = new Food(); //创建食物

Snake2 snake2 = new Snake2(food); //创建蛇1

Snake1 snake1 = new Snake1(food); //创建蛇2

SnakeThread snakeThread = new SnakeThread(); //创建一个线程

ImageIcon img = new ImageIcon("C:\\Users\\15869\\Desktop\\疯狂贪吃蛇\\3.jpg");

JButton jb1 = new JButton("暂停");

JButton jb2 = new JButton("继续");

JButton jb3 = new JButton("重新开始");

static JButton jb4 = new JButton("返回");

JLabel label = new JLabel(img);

JLabel jLabel= new JLabel("<html>消灭对方<br>获得胜利<html>");

JPanel jPanel = new JPanel(); //放置背景图片的面板

JPanel jp2 = new JPanel(); //放置按钮的面板

public View\_Double(){

jLabel.setFont(new Font("宋体",Font.BOLD,40));

jLabel.setForeground(Color.red);

jp2.setBounds(1000,0,270,720);

jPanel.setBounds(0,0,1280, 720);

label.setBounds(0,0,1280, 720);

jLabel.setBounds(50,50,270,300);

jb1.setBounds(80,340,100,50);

jb2.setBounds(80,410,100,50);

jb3.setBounds(80,480,100,50);

jb4.setBounds(80,600,100,50);

jp2.add(jb1);

jp2.add(jb2);

jp2.add(jb3);

jp2.add(jb4);

jp2.add(jLabel);

jp2.setLayout(null);

jp2.setOpaque(false);

jPanel.add(label);

this.add(jPanel,DEFAULT\_LAYER); //将背景放置下层

this.add(jp2,MODAL\_LAYER); //将其他东西放上层

snakeThread.start(); //启动线程

this.addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent a) { //添加按键监听，控制蛇移动

snake1.keyboard(a);

snake2.keyboard(a);

}

});

jb1.addActionListener(new ActionListener() { //暂停按钮监听

@Override

public void actionPerformed(ActionEvent e) {

isMove=false;

}

});

jb2.addActionListener(new ActionListener() { //继续按钮监听

@Override

public void actionPerformed(ActionEvent e) {

isMove= true;

setFocusable(true);

requestFocus();

}

});

jb3.addActionListener(new ActionListener() { //重新开始按钮监听

@Override

public void actionPerformed(ActionEvent e) {

snakeThread.stopThread(); //让线程死亡

food = new Food();

snake2 =new Snake2(food);

snake1 =new Snake1(food);

Snake2.isLive2=true;

Snake1.isLive1=true;

isMove=true;

SnakeThread st =new SnakeThread(); //创建新线程

snakeThread=st;

st.start();

setFocusable(true);

requestFocus();

}

});

}

public static JButton getJb4()

{

return jb4;

} //获得返回按钮

public void paint(Graphics g) {

super.paint(g);

snake2.move\_double(snake1); //蛇移动

snake1.move\_double(snake2); //蛇移动

snake2.draw(g);

snake1.draw(g);

food.draw(g);

}

public class SnakeThread extends Thread{

static boolean flag = true;

public SnakeThread(){

flag=true;

}

@Override

public void run() {

while(Snake1.isLive1 &&Snake2.isLive2&&flag){

try {

Thread.sleep(150);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

if(Snake1.isLive1&&Snake2.isLive2&&isMove){ //蛇存活且没有暂停则重绘画面

repaint();

}

}

if(Snake1.isLive1==false|| Snake2.isLive2==false){ //游戏结束后的结果判定

if(winner==1)

{

JOptionPane.showMessageDialog(View\_Double.this, "恭喜玩家一白蛇获胜！");

}

else if(winner==2)

{

JOptionPane.showMessageDialog(View\_Double.this, "恭喜玩家二青蛇获胜！");

}

else

{

JOptionPane.showMessageDialog(View\_Double.this, "你们同归于尽了！");

}

flag=false;

}

}

public static void stopThread(){

flag=false;

}

}

}

**View\_Help.java**

package View;

import javax.swing.\*;

import java.awt.\*;

import static javax.swing.JLayeredPane.DEFAULT\_LAYER;

import static javax.swing.JLayeredPane.MODAL\_LAYER;

//帮助窗口

public class View\_Help extends JDialog {

ImageIcon img = new ImageIcon("C:\\Users\\15869\\Desktop\\疯狂贪吃蛇\\2.jpg");

JLayeredPane jLayeredPane = new JLayeredPane(); //分层面板

JLabel jLabel = new JLabel(img); //放置图片的标签

JLabel jLabel2 = new JLabel(); //放置文字的标签

JLabel jLabel3 = new JLabel();

JLabel jLabel4 = new JLabel();

JPanel jp2 = new JPanel(); //放置标签的面板

JPanel jp3 = new JPanel();

JPanel jp4 = new JPanel();

public View\_Help(){

this.setAlwaysOnTop(true); //设置子窗口属性

this.setTitle("游戏帮助");

this.setBounds(600,300,500,500);

this.setVisible(true);

this.setLayout(null);

this.setDefaultCloseOperation(DISPOSE\_ON\_CLOSE);

jLabel.setBounds(0,0,500,500);

jLabel2.setText("游戏玩法");

jLabel3.setText("<html>单人模式<br>↑ ↓ ← → 控制蛇移动<br>吃到食物获得分数<br>不要碰到墙<br>不要咬到自己</html>");

jLabel4.setText("<html>双人模式<br>玩家一通过 W S A D 控制蛇移动<br>玩家二通过 ↑ ↓ ← → 控制蛇移动<br>您可以通过让对方撞到您杀死对方<br>从而获取胜利</html>");

jLabel2.setFont(new Font("宋体",Font.BOLD,30));

jLabel3.setFont(new Font("宋体",Font.PLAIN,15));

jLabel4.setFont(new Font("宋体",Font.PLAIN,15));

jLabel2.setForeground(Color.yellow);

jLabel3.setForeground(Color.red);

jLabel4.setForeground(Color.red);

jLabel3.setVerticalAlignment(SwingConstants.CENTER);

jLabel4.setVerticalAlignment(SwingConstants.CENTER);

jp2.add(jLabel2);

jp3.add(jLabel3);

jp4.add(jLabel4);

jp2.setBounds(0,20,500,100);

jp3.setBounds(0,100,250,400);

jp4.setBounds(250,100,250,400);

jLayeredPane.add(jLabel,DEFAULT\_LAYER); //背景图片放下层

jLayeredPane.add(jp2,MODAL\_LAYER); //文字放上层

jLayeredPane.add(jp3,MODAL\_LAYER);

jLayeredPane.add(jp4,MODAL\_LAYER);

jp2.setOpaque(false); //设置文字背景透明

jp3.setOpaque(false);

jp4.setOpaque(false);

this.setContentPane(jLayeredPane);

}

}

**View\_Home.java**

package View;

import javax.swing.\*;

import java.awt.\*;

//主菜单界面

public class View\_Home extends JLayeredPane {

ImageIcon img = new ImageIcon("C:\\Users\\15869\\Desktop\\疯狂贪吃蛇\\7.jpg");

static JButton jb1 = new JButton("单人模式");

static JButton jb2 = new JButton("双人模式");

static JButton jb3 = new JButton("游戏帮助");

static JButton jb4 = new JButton("切换音乐");

static JButton jb5 = new JButton("音乐开关");

static JButton jb6 = new JButton("关于");

JLabel label = new JLabel(img);

JLabel label2 = new JLabel();

JPanel jp1 = new JPanel(); //放置背景图片的面板

JPanel jp2 = new JPanel(); //放置文字的面板

JPanel jp3 = new JPanel(); //放置按钮的面板

JPanel jp4 = new JPanel(); //放置按钮的面板

public View\_Home(){

jp2.setOpaque(false); //设置面板透明

jp3.setOpaque(false);

jp4.setOpaque(false);

jp2.setLayout(null); //设置面板布局

jp3.setLayout(null);

jp4.setLayout(null);

label2.setText("疯狂贪吃蛇");

label2.setFont(new Font("隶书",Font.BOLD,100));

label2.setForeground(Color.red);

jp1.setBounds(0,0,1280, 720); //设置位置

jp2.setBounds(0,0,1280,480);

jp3.setBounds(0,480,1280,240);

jp4.setBounds(960,0,270,480);

label.setBounds(0,0,1280, 720);

label2.setBounds(360,0,800,240);

jb1.setBounds(320,50,200,100);

jb2.setBounds(760,50,200,100);

jb3.setBounds(140,50,100,50);

jb4.setBounds(140,150,100,50);

jb5.setBounds(140,250,100,50);

jb6.setBounds(140,350,100,50);

jp1.add(label);

jp2.add(label2);

jp3.add(jb1);

jp3.add(jb2);

jp4.add(jb3);

jp4.add(jb4);

jp4.add(jb5);

jp4.add(jb6);

this.add(jp1,DEFAULT\_LAYER); //设置图层位置

this.add(jp2,MODAL\_LAYER);

this.add(jp3,MODAL\_LAYER);

this.add(jp4,MODAL\_LAYER);

}

public static JButton getJb1() {

return jb1;

} //获得按钮

public static JButton getJb2() {

return jb2;

}

public static JButton getJb3() {

return jb3;

}

public static JButton getJb4() {

return jb4;

}

public static JButton getJb5() {

return jb5;

}

public static JButton getJb6() {

return jb6;

}

}

**View\_Single.java**

package View;

import Date.Score;

import Game.Food;

import Game.Snake;

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyAdapter;

import java.awt.event.KeyEvent;

//单人模式

public class View\_Single extends JLayeredPane {

public static boolean isMove = true; //判断是否游戏暂停

public boolean moveflag=true; //判断键盘监听是否获得内容的标志

Food food = new Food(); //创建食物

Snake snake = new Snake(food,5,20); //创建蛇

SnakeThread\_Single snakeThread = new SnakeThread\_Single(); //创建线程

ImageIcon img = new ImageIcon("C:\\Users\\15869\\Desktop\\疯狂贪吃蛇\\3.jpg");

JButton jb1 = new JButton("暂停");

JButton jb2 = new JButton("继续");

JButton jb3 = new JButton("重新开始");

static JButton jb4 = new JButton("返回");

JLabel label = new JLabel(img);

JPanel jp1 = new JPanel();

JPanel jp2 = new JPanel();

JLabel score = new JLabel("您当前得分："+ Snake.score);

JLabel user = new JLabel("尊敬的用户："+Score.user);

JLabel highscore = new JLabel("<html>历史记录<br>用户名：<html>"+Score.highuser+"<html><br>得分：<html>"+Score.highscore);

public View\_Single(){

isMove=true;

score.setFont(new Font("宋体",Font.BOLD,20));

score.setForeground(Color.blue);

user.setFont(new Font("宋体",Font.BOLD,20));

user.setForeground(Color.blue);

highscore.setFont(new Font("宋体",Font.BOLD,20));

highscore.setForeground(Color.red);

user.setBounds(20,0,270,120);

score.setBounds(20,50,270,120);

highscore.setBounds(20,150,270,120);

label.setBounds(0,0,1280, 720);

jb1.setBounds(80,340,100,50);

jb2.setBounds(80,410,100,50);

jb3.setBounds(80,480,100,50);

jb4.setBounds(80,600,100,50);

jp1.setBounds(0,0,1280, 720);

jp2.setBounds(1000,0,280,720);

jp1.add(label);

jp2.add(jb1);

jp2.add(jb2);

jp2.add(jb3);

jp2.add(jb4);

jp2.add(score);

jp2.add(user);

jp2.add(highscore);

jp2.setLayout(null);

jp2.setOpaque(false);

this.add(jp1,DEFAULT\_LAYER);

this.add(jp2,MODAL\_LAYER);

snakeThread.start(); //启动线程

this.addKeyListener(new KeyAdapter() {

@Override

public void keyPressed(KeyEvent e) { //键盘监听控制蛇移动

if(moveflag==true) { //通过加入布尔值让蛇每次移动只能获取一个按键

snake.keyboard(e);

}

moveflag=false;

}

});

jb1.addActionListener(new ActionListener() { //暂停按钮监听

@Override

public void actionPerformed(ActionEvent e) {

isMove=false;

}

});

jb2.addActionListener(new ActionListener() { //继续按钮监听

@Override

public void actionPerformed(ActionEvent e) {

isMove= true;

setFocusable(true);

requestFocus();

}

});

jb3.addActionListener(new ActionListener() { //重新开始按钮监听

@Override

public void actionPerformed(ActionEvent e) {

snakeThread.stopThread(); //让线程死亡

food = new Food();

snake=new Snake(food,5,20);

Snake.islive=true;

isMove=true;

SnakeThread\_Single st =new SnakeThread\_Single(); //新建一个线程

snakeThread=st;

st.start();

setFocusable(true);

requestFocus();

}

});

}

public static JButton getJb4()

{

return jb4;

} //获得返回按钮

public void paint(Graphics g) {

super.paint(g);

snake.move(); //蛇移动

snake.draw(g);

food.draw(g);

moveflag=true;

}

public class SnakeThread\_Single extends Thread{

static boolean flag1 = true;

public SnakeThread\_Single(){

flag1=true;

}

@Override

public void run() {

while(Snake.islive &&flag1){

try {

Thread.sleep(150);

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

if(Snake.islive&&isMove){

repaint();

score.setText("您当前得分："+ Snake.score);

}

}

if(Snake.islive==false){ //游戏结束结果判断

if(Snake.score> Score.highscore)

{

JOptionPane.showMessageDialog(View\_Single.this, "恭喜您打破了历史记录！");

}

else {

JOptionPane.showMessageDialog(View\_Single.this, "游戏结束");

}

new Score();

highscore.setText("<html>历史记录<br>用户名：<html>"+Score.highuser+"<html><br>得分：<html>"+Score.highscore);

flag1=false;

}

}

public static void stopThread(){

flag1=false;

}

}

}

**Food.java**

package Game;

import java.awt.\*;

public class Food {

int row;

int col;

public Food(){

row = 20; //创建食物的大小

col =20;

}

public void repearShow(){

row = (int)(Math.random()\*18); //生成随机数 乘以食物的大小可以得到坐标

col = (int)(Math.random()\*32);

if(row>51||row<0||col<0||col>51) repearShow();

}

public void draw(Graphics g) { //把食物画出来

g.setColor(Color.red);

g.fillOval(col\*20, row\*20, 20, 20); //表示坐标

}

public Rectangle getCoordinates() {

return new Rectangle(col \* 20, row \* 20, 20, 20); //获得食物的坐标

}

}

**Music.java**

package Game;

import javax.sound.sampled.AudioInputStream;

import javax.sound.sampled.AudioSystem;

import javax.sound.sampled.Clip;

import java.io.File;

public class Music {

public static boolean r=true;

public static int musicflag=1;

String musicLocation;

Clip clip;

public Music(String musicLocation){

this.musicLocation=musicLocation;

File music = new File(musicLocation);

try

{

if(music.exists())

{

AudioInputStream audioInput = AudioSystem.getAudioInputStream(music);

clip = AudioSystem.getClip();

clip.open(audioInput);

}

}

catch(Exception ex)

{

ex.printStackTrace();

}

playMusic();

}

public void playMusic() //播放音乐

{

clip.start();

clip.loop(Clip.LOOP\_CONTINUOUSLY);

}

public void stopMusic() //关闭音乐

{

clip.stop();

}

}

**Snake.java**

package Game;

import java.awt.\*;

import java.awt.event.KeyEvent;

public class Snake {

public static final int span=20;

public static final int rows=34;

public static final int cols=50;

public static int score= 0;

public static final String up="u";

public static final String down="d";

public static final String left="l";

public static final String right="r";

public static boolean islive=true;

Node body;

public Node head;

Node tail;

Food food;

public Snake(Food food,int row,int col){

islive=true;

score = 0;

body = new Node(row,col,left);

head = body;

tail = body;

this.food=food;

}

class Node{

int row;

int col;

String dir;

Node next;

Node pre;

Node(int row,int col,String dir) {

this.row=row;

this.col=col;

this.dir=dir;

}

public void draw(Graphics g){

g.fillRect(col\*span,row\*span,span,span);

}

}

public void draw(Graphics g) {

g.setColor(Color.yellow);

for (Node n = head; n != null; n = n.next) {

n.draw(g);

g.setColor(Color.pink);

}

}

public void keyboard(KeyEvent e) {

switch(e.getKeyCode()){

case KeyEvent.VK\_UP:

if(head.dir.equals(down)){

break;

}

head.dir=up;

break;

case KeyEvent.VK\_DOWN:

if(head.dir.equals(up)){

break;

}

head.dir=down;

break;

case KeyEvent.VK\_LEFT:

if(head.dir.equals(right)){

break;

}

head.dir=left;

break;

case KeyEvent.VK\_RIGHT:

if(head.dir.equals(left)){

break;

}

head.dir=right;

break;

default:

break;

}

}

public void addHead(){

Node node = null;

switch (head.dir){

case "l":

node = new Node(head.row,head.col-1,head.dir);

break;

case "r":

node = new Node(head.row,head.col+1,head.dir);

break;

case "u":

node = new Node(head.row-1,head.col,head.dir);

break;

case "d":

node = new Node(head.row+1,head.col,head.dir);

break;

default:

break;

}

node.next=head;

head.pre=node;

head=node;

}

public void deleteTail(){

tail.pre.next=null;

tail=tail.pre;

}

public void move() {

addHead();

if(this.getSnakeRectangle().intersects(food.getCoordinates())){ //当蛇头与食物重合的时候 蛇吃食物 食物刷新，不再删除尾巴，达到一种蛇增长的要求

score=score+10;

food.repearShow();

while(true) {

Node n;

for (n = head; n != null; n = n.next) {

if (food.col == n.col && food.row == n.row) {

food.repearShow();

break;

}

}

if(n==null)

{

break;

}

}

}

else{

deleteTail();

}

DeadOrLive(); //每移动一步都要判断一下是否存活

}

public Rectangle getSnakeRectangle(){ //获取蛇头的坐标

return new Rectangle(head.col\*span,head.row\*span,span,span);

}

public void DeadOrLive() { //超出边框范围 蛇头撞到身体 游戏结束

if (head.row < 0 || head.row > rows-1 || head.col < 0 || head.col > cols-1 ) {

islive = false;

}

for (Node n = head.next; n != null; n = n.next) {

if (n.col == head.col && n.row == head.row) {

islive = false;

}

}

}

}

**Snake1.java**

package Game;

import View.View\_Double;

import java.awt.\*;

import java.awt.event.KeyEvent;

public class Snake1 extends Snake {

public static boolean isLive1=true;

public Snake1(Food food) {

super(food, 20,20);

isLive1=true;

}

public void keyboard(KeyEvent e) { //双人模式下的键盘按键

switch (e.getKeyCode()) {

case KeyEvent.VK\_W:

if (head.dir.equals(down)) {

break;

}

head.dir = up;

break;

case KeyEvent.VK\_S:

if (head.dir.equals(up)) {

break;

}

head.dir = down;

break;

case KeyEvent.VK\_A:

if (head.dir.equals(right)) {

break;

}

head.dir = left;

break;

case KeyEvent.VK\_D:

if (head.dir.equals(left)) {

break;

}

head.dir = right;

break;

default:

break;

}

}

public void move\_double(Snake2 snake2) {

addHead();

if(this.getSnakeRectangle().intersects(food.getCoordinates())){

score=score+10;

food.repearShow();

}else{

deleteTail();

}

DeadOrLive\_double(snake2);

}

public void DeadOrLive\_double(Snake2 snake2) { //双人模式下的死亡判断

if (head.row < 0 || head.row > rows - 1 || head.col < 0 || head.col > cols - 1) {

isLive1 = false;

View\_Double.winner=2;

}

for (Node n = head.next; n != null; n = n.next) {

if (n.col == head.col && n.row == head.row) {

isLive1 = false;

View\_Double.winner=2;

}

}

for (Node n =head.next;n!=null;n=n.next)

{

if (n.col==snake2.head.col && n.row==snake2.head.row){

isLive1=false;

View\_Double.winner=1;

}

}

if(head.row==snake2.head.row&&head.col==snake2.head.col){

isLive1=false;

View\_Double.winner=3;

}

}

public void draw(Graphics g) {

g.setColor(Color.pink);

for (Node n = head; n != null; n = n.next) {

n.draw(g);

g.setColor(Color.white);

}

}

}

**Snake2.java**

package Game;

import View.View\_Double;

import java.awt.\*;

public class Snake2 extends Snake{

public static boolean isLive2=true;

public Snake2(Food food) {

super(food, 20,30);

isLive2=true;

}

public void move\_double(Snake1 snake1) {

addHead();

if(this.getSnakeRectangle().intersects(food.getCoordinates())){

score=score+10;

food.repearShow();

}else{

deleteTail();

}

DeadOrLive\_double(snake1);

}

public void DeadOrLive\_double(Snake1 snake1) { //双人模式下的死亡判断

if (head.row < 0 || head.row > rows - 1 || head.col < 0 || head.col > cols - 1) {

isLive2 = false;

View\_Double.winner=1;

}

for (Node n = head.next; n != null; n = n.next) {

if (n.col == head.col && n.row == head.row) {

isLive2 = false;

View\_Double.winner=1;

}

}

for (Node n =head.next;n!=null;n=n.next)

{

if (n.col==snake1.head.col && n.row==snake1.head.row){

isLive2=false;

View\_Double.winner=2;

}

}

}

public void draw(Graphics g) {

g.setColor(Color.cyan);

for (Node n = head; n != null; n = n.next) {

n.draw(g);

g.setColor(Color.green);

}

}

}

**Score.java**

package Date;

import Game.Snake;

import java.sql.\*;

public class Score {

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

public static int highscore;

public static String highuser;

public static String user;

public Score(String n){

user=n;

try {

String USER="root";

String PASS="winner.123";

Class .forName("com.mysql.cj.jdbc.Driver");

ct= DriverManager.getConnection("jdbc:mysql://localhost:3306/game?user="+USER+"&password="+PASS + "&useUnicode=true&characterEncoding=gbk&useSSL=false&serverTimezone=GMT");

if(highscore< Snake.score){

ps=ct.prepareStatement("update user\_t set highscore =?,time=curdate() where name =?");

ps.setInt(1, Snake.score);

ps.setString(2,user);

ps.executeUpdate();

}

ps=ct.prepareStatement("select highscore from user\_t where name=?");

ps.setString(1, user);

rs=ps.executeQuery();

while (rs.next()){

highscore=rs.getInt(1);

}

}catch (Exception e){

e.printStackTrace();

} finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

}

**Mysql.Java**

package Date;

import View.View\_Login;

import View.View\_Register;

import javax.swing.\*;

import java.sql.\*;

import java.text.SimpleDateFormat;

import java.util.Vector;

public class Mysql {

private Vector<Vector<Object>> rowData = new Vector<Vector<Object>>();

static SimpleDateFormat sdf= new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

PreparedStatement ps=null;

Connection ct=null;

ResultSet rs=null;

public int loginin(){

int h=-1;

try {

String USER="root";

String PASS="winner.123";

Class .forName("com.mysql.cj.jdbc.Driver");

ct= DriverManager.getConnection("jdbc:mysql://localhost:3306/game?user="+USER+"&password="+PASS + "&useUnicode=true&characterEncoding=gbk&useSSL=false&serverTimezone=GMT");

ps=ct.prepareStatement("select highscore from user\_t where name=? and password=?");

ps.setString(1,View\_Login.getn());

ps.setString(2,View\_Login.getp());

rs=ps.executeQuery();

while (rs.next()){

h=rs.getInt(1);

}

}catch (Exception e){

e.printStackTrace();

} finally{

try {

if(rs!=null){

rs.close();

}

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

return h;

}

public int regist(){

int a=0;

try {

String USER="root";

String PASS="winner.123";

Class .forName("com.mysql.cj.jdbc.Driver");

ct= DriverManager.getConnection("jdbc:mysql://localhost:3306/game?user="+USER+"&password="+PASS + "&useUnicode=true&characterEncoding=gbk&useSSL=false&serverTimezone=GMT");

ps=ct.prepareStatement("insert into user\_t(name,password) values('" + View\_Register.getn() + "','" + View\_Register.getp() + "')");

a=ps.executeUpdate();

if(a == 1){

JOptionPane.showMessageDialog(null,"注册成功！");

}

}catch (Exception e){

JOptionPane.showMessageDialog(null,"对不起，该用户名已被注册！");

e.printStackTrace();

} finally{

try {

if(ps!=null){

ps.close();

}

if(ct!=null){

ct.close();

}

} catch (SQLException e) {

e.printStackTrace();

}

}

return a;

}

}