Java代码

1.Login类

package SnakeGameDemo;

import java.awt.Color;

import java.awt.Container;

import java.awt.Graphics;

import java.awt.GridLayout;

import java.awt.Image;

import java.awt.Label;

import java.awt.Panel;

import java.awt.TextField;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

//登陆界面

public class Login extends JFrame {

private TextField f1;//创建文本框

private TextField f2;

private JButton b1;//创建按钮

private JButton b2;

public Login() {//登陆界面

Container cp=getContentPane();//获取容器

cp.setLayout(new GridLayout(4,1));//设置布局管理为网格四行一列

Label l1=new Label("用户名");//创建标签了 l1，l2

Label l2=new Label("密 码");

Panel p1=new Panel();//创建面板，p1,p2,p3，p4

Panel p2=new Panel();

Panel p3=new Panel();

MyPanel1 p4=new MyPanel1();//创建插入图片平面 p4

f1=new TextField(10);

f2=new TextField(10);

f2.setEchoChar('\*');//回显字符为\* 用于输入密码，设置为不可见

b1=new JButton("确定");//新建按钮 b1，b2

b2=new JButton("取消");

p1.add(l1);//第一行添加 label 1

p1.add(f1);

p2.add(l2);

p2.add(f2);

p3.add(b1);

p3.add(b2);

cp.add(p4);//面板 p4，p1，p2，p3 分别加入容器 cp

cp.add(p1);

cp.add(p2);

cp.add(p3);

b1.addActionListener(new Enter());//添加按钮事件响应

b2.addActionListener(new Close());

}

class Enter implements ActionListener {

public void actionPerformed(ActionEvent e) {

JButton bt=(JButton)e.getSource();

if((f1.getText()).equals("孙定")&&(f2.getText()).equals("123456")

&&bt==b1)

{

JFrame.setDefaultLookAndFeelDecorated(true);

SnakeFrame frame=new SnakeFrame();//创建游戏主界面

frame.setSize(frame.col\*11+11,frame.row\*11+100);

frame.setResizable(false);

frame.setLocation(500,200);//设置窗体位置

frame.setTitle("贪吃蛇");

frame.setVisible(true);

}

else JOptionPane.showMessageDialog(null, "用户名或密码错误，请重新登录！");//错误用户提示

}

}

class Close implements ActionListener {//关闭登陆按钮响应

public void actionPerformed(ActionEvent e) {

JButton bt=(JButton)e.getSource();

if(bt==b2) {

System.exit(0);//退出

}

}

}

//主函数 程序开始

public static void main(String[] args) {

Login log=new Login();

log.setTitle("疯狂贪吃蛇界面登陆");

log.setBounds(500, 200, 300, 300);

log.setBackground(Color.blue);

log.setVisible(true);

}

}

class MyPanel1 extends JPanel {//插入登陆界面图片

Image

img=Toolkit.getDefaultToolkit().getImage("imgs/201.jpg");// 添 加

public void paint(Graphics g) {

g.drawImage(img,0,0,this);

}

}

2.SnakeFrame 类

package SnakeGameDemo;

import java.awt.Color;

import java.awt.Container;

import java.awt.FlowLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import javax.swing.ButtonGroup;

import javax.swing.JCheckBoxMenuItem;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JRadioButtonMenuItem;

import javax.swing.JTextField;

import SnakeGameDemo.SnakePanel;

public class SnakeFrame extends JFrame implements ActionListener{

int row=20;

int col=30;

SnakePanel p=new SnakePanel(this,row,col);

JMenuBar menubar=new JMenuBar();//菜单条

JMenu fileMenu=new JMenu("文件");//菜单

JMenuItem newgameitem=new JMenuItem("开始");//菜单条目

JMenuItem stopitem=new JMenuItem("暂停");

JMenuItem runitem=new JMenuItem("继续");

JMenuItem exititem=new JMenuItem("退出");

//"设置"菜单

JMenu optionMenu=new JMenu("设置");

/\*

\* 等级选项

\* ButtonGroup 创建一组按钮选项

\* JRadioButtonMenuItem 是属于一组菜单项中的一个菜单项，该组

中只能选择一个项。

\* 被选择的项显示其选择状态。选择此项的同时，其他任何以前被

选择的项都切换到未选择状

态。

\* Panel 是最简单的容器类。应用程序可以将其他组件放在面板提

供的空间内，这些组件包括

其他面板。

\*/

JMenu degreeMenu=new JMenu("等级");

ButtonGroup groupDegree = new ButtonGroup();

JRadioButtonMenuItem oneItem= new JRadioButtonMenuItem(" 初级");

JRadioButtonMenuItem twoItem= new JRadioButtonMenuItem(" 中级");

JRadioButtonMenuItem threeItem= new JRadioButtonMenuItem(" 高级");

// 设置“窗口大小”菜单

JMenu windowsMenu = new JMenu("窗口大小");

JMenu windowsMenu1 = new JMenu("视图");

ButtonGroup groupDegree1 = new ButtonGroup();

JRadioButtonMenuItem minItem= new JRadioButtonMenuItem(" 小");

JRadioButtonMenuItem midItem= new JRadioButtonMenuItem(" 中");

JRadioButtonMenuItem maxItem= new JRadioButtonMenuItem(" 大");

// 设置“作者信息”菜单

JMenu XinXi=new JMenu("关于开发者");

JMenuItem XinXiItem=new JMenuItem("软件相关");

//设置“帮助”菜单

JMenu helpMenu=new JMenu("帮助");

JMenuItem helpItem=new JMenuItem("操作指南");

//可以被选定或取消选定的菜单项.

final JCheckBoxMenuItem showGridItem = new

JCheckBoxMenuItem("显示网格");

JLabel scorelabel;//为分数创建标签

public JTextField scoreField;//显示分数的文本框

private long speedtime=200;

private String helpstr = "游戏说明：\n1 ：方向键控制蛇移动的方向."+

"\n2 ：单击菜单'文件->开始'开始游戏."+

"\n3 ：单击菜单'文件->暂停'或者单击键盘空格键暂停游戏."+

"\n4 ：单击菜单'文件->继续'继续游戏."+

"\n5 ：单击菜单'设置->等级'可以设置难度等级."+

"\n6 ：单击菜单'设置->显示网格'可以设置是否显示网格."+

"\n7 ：红色为食物,吃一个得 10 分同时蛇身加长."+

"\n8 ：蛇不可以出界或自身相交，否则结束游戏.";

SnakeFrame() {

setJMenuBar(menubar);

fileMenu.add(newgameitem);

fileMenu.add(stopitem);

fileMenu.add(runitem);

fileMenu.add(exititem);

menubar.add(fileMenu);

oneItem.setSelected(true); //设置可选择

groupDegree.add(oneItem);

groupDegree.add(twoItem);

groupDegree.add(threeItem);

degreeMenu.add(oneItem);

degreeMenu.add(twoItem);

degreeMenu.add(threeItem);

optionMenu.add(degreeMenu);

minItem.setSelected(true);

groupDegree1.add(minItem);

groupDegree1.add(midItem);

groupDegree1.add(maxItem);

windowsMenu1.add(minItem);

windowsMenu1.add(midItem);

windowsMenu1.add(maxItem);

windowsMenu.add(windowsMenu1);

// 风格选项

showGridItem.setSelected(true);

optionMenu.add(showGridItem);

menubar.add(optionMenu);

helpMenu.add(helpItem);

XinXi.add(XinXiItem);

menubar.add(windowsMenu);

menubar.add(XinXi);

menubar.add(helpMenu);

Container contentpane=getContentPane();

contentpane.setLayout(new FlowLayout());//设置布局管理

contentpane.add(p);

scorelabel=new JLabel("得 分: ");

scoreField=new JTextField("0",15);

scoreField.setEnabled(false);

scoreField.setHorizontalAlignment(0);//设 置 图 标 和 文 本的水平对齐方式

JPanel toolPanel=new JPanel();

toolPanel.add(scorelabel);

toolPanel.add(scoreField);

contentpane.add(toolPanel);

//事件监听 使得按钮可以响应

oneItem.addActionListener(this);

twoItem.addActionListener(this);

threeItem.addActionListener(this);

minItem.addActionListener(this);

midItem.addActionListener(this);

maxItem.addActionListener(this);

newgameitem.addActionListener(this);

stopitem.addActionListener(this);

runitem.addActionListener(this);

exititem.addActionListener(this);

XinXiItem.addActionListener(this);

helpItem.addActionListener(this);

showGridItem.addActionListener(this);

}

public void actionPerformed(ActionEvent e){

// 事件响应 从接口 ActionListener 复制的描述 ，发生操作时调用。

try{//异常捕获

if(e.getSource()==helpItem){

JOptionPane.showConfirmDialog(p,helpstr,"操 纵 说 明",JOptionPane.PLAIN\_MESSAGE);

}

else if(e.getSource()==exititem){

JOptionPane.showConfirmDialog(null,

"你确定退出游戏吗？", " ",

JOptionPane.YES\_NO\_OPTION);

}

else if(e.getSource()==newgameitem)

p.newGame(speedtime);

else if(e.getSource()==stopitem) p.stopGame();

else if(e.getSource()==runitem) p.returnGame();

else if(e.getSource()==showGridItem) {

if(!showGridItem.isSelected()) {

p.setBackground(Color.lightGray);// 设 置 背 景颜色

}else{

p.setBackground(Color.darkGray);

}

}

else if(e.getSource()==oneItem) speedtime=200;

else if(e.getSource()==twoItem) speedtime=100;

else if(e.getSource()==threeItem) speedtime=50;

else if(e.getSource()==minItem){

row=20;col=30;

p.change(this,row,col);

this.setSize(col\*11+11,row\*11+100);

}

else if(e.getSource()==XinXiItem)

JOptionPane.showMessageDialog(this,"贪吃蛇\n\n 石家庄铁道大学\n 信息科学与技术学院\n 网络工程" +"\n 姓名：孙定\n"," 软 件 相 关",JOptionPane.INFORMATION\_MESSAGE);

else if(e.getSource()==midItem){

row=30;col=45;

p.change(this,row,col);

this.setSize(col\*11+11,row\*11+100);

}

else if(e.getSource()==maxItem) {

row=40;col=60;

p.change(this,row,col);

this.setSize(col\*11+11,row\*11+100);

}

}catch(Exception ee){ee.printStackTrace();}//异常处理

}

}

3.SnakeModel 类

package SnakeGameDemo;

import java.util.LinkedList;

import javax.swing.JOptionPane;

public class SnakeModel {

private int rows,cols;//行列数

private Location snakeHead,runingDiriction;//运行方向

private LocationRO[][] locRO;//LocationRO 类数组

private LinkedList snake,playBlocks;//蛇及其它区域块

private LocationRO snakeFood;//目标食物

private int gameScore=0; //分数

private boolean AddScore=false;//加分

//获得蛇头

public LocationRO getSnakeHead(){

return (LocationRO)(snake.getLast());

}

//蛇尾

public LocationRO getSnakeTail(){

return (LocationRO)(snake.getFirst());

}

//运行路线

public Location getRuningDiriction(){

return runingDiriction;

}

//获得蛇实体区域

public LinkedList getSnake(){

return snake;

}

//其他区域

public LinkedList getOthers(){

return playBlocks;

}

//获得总分

public int getScore(){

return gameScore;

}

//获得增加分数

public boolean getAddScore(){

return AddScore;

}

//设置蛇头方向

private void setSnakeHead(Location snakeHead){

this.snakeHead=snakeHead;

}

//获得目标食物

public LocationRO getSnakeFood(){

return snakeFood;

}

//随机设置目标食物

private void setSnakeFood(){

snakeFood=(LocationRO)(playBlocks.get((int)(Math.random()\*playBlocks.size())));

}

//移动

private void moveTo(Object a,LinkedList fromlist,LinkedList

tolist){

fromlist.remove(a);

tolist.add(a);

}

//初始设置

public void init(){

playBlocks.clear();

snake.clear();

gameScore=0;

for(int i=0;i<rows;i++){

for(int j=0;j<cols;j++){

playBlocks.add(locRO[i][j]);

}

}

//初始化蛇的形状

for(int i=4;i<7;i++){

moveTo(locRO[4][i],playBlocks,snake);

}

//蛇头位置

snakeHead=new Location(4,6);

//设置随机块

snakeFood=new LocationRO(0,0);

setSnakeFood();

//初始化运动方向

runingDiriction=new Location(0,1);

}

//Snake 构造器

public SnakeModel(int rows1,int cols1){

rows=rows1;

cols=cols1;

locRO=new LocationRO[rows][cols];

snake=new LinkedList();

playBlocks=new LinkedList();

for(int i=0;i<rows;i++){

for(int j=0;j<cols;j++){

locRO[i][j]=new LocationRO(i,j);

}

}

init();

}

// /\*\*定义布尔型 move 方法，如果运行成功则返回 true,否则返回 false

// \*参数 direction 是 Location 类型，

// \*direction 的值:(-1,0)表示向上；(1,0)表示向下；

// \*(0,-1)表示向左；(0,1)表示向右；

// \*\*/

public boolean move(Location direction){

//判断设定的方向跟运行方向是不是相反

if (direction.reverse(runingDiriction)){

snakeHead.setX(snakeHead.getX()+runingDiriction.getX());

snakeHead.setY(snakeHead.getY()+runingDiriction.getY());

}else{

snakeHead.setX(snakeHead.getX()+direction.getX());

snakeHead.setY(snakeHead.getY()+direction.getY());

}

//如果蛇吃到了目标食物

try{

if ((snakeHead.getX()==snakeFood.getX())

&&(snakeHead.getY()==snakeFood.getY()))

{

moveTo(locRO[snakeHead.getX()][snakeHead.getY()],playBlocks,

snake);

setSnakeFood();

gameScore+=10;

AddScore=true;

}else{

AddScore=false;

//是否出界

if((snakeHead.getX()<rows)&&(snakeHead.getY()<cols)&&(snakeHead.getX()>=0&&

(snakeHead.getY()>=0))){

//如果不出界，判断是否与自身相交

if(snake.contains(locRO[snakeHead.getX()]

[snakeHead.getY()])){

//如果相交,结束游戏

JOptionPane.showMessageDialog(null, "GameOver!", "游戏结束", JOptionPane.INFORMATION\_MESSAGE);

return false;

}else{

//如果不相交，就把 snakeHead 加到 snake 里面，并且把尾巴移出

moveTo(locRO[snakeHead.getX()]

[snakeHead.getY()],playBlocks,snake);

moveTo(snake.getFirst(),snake,playBlocks);

}

}else{

//出界,游戏结束

JOptionPane.showMessageDialog(null, "Game Over!",

"游戏结束", JOptionPane.INFORMATION\_MESSAGE);

return false;

}

}

}catch(ArrayIndexOutOfBoundsException e){

return false;

}

//设置运行方向

if (!(direction.reverse(runingDiriction)

||direction.equals(runingDiriction))){

runingDiriction.setX(direction.getX());

runingDiriction.setY(direction.getY());

}

return true;

}

}

4. SnakePanel 类

package SnakeGameDemo;

import java.awt.Color;

import java.awt.GridLayout;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.util.Iterator;

import java.util.LinkedList;

import javax.swing.JFrame;

import javax.swing.JPanel;

public class SnakePanel extends JPanel implements

Runnable,KeyListener{

JFrame parent=new JFrame();

private int row; //网格行数 30-341 60-671 40-451

private int col; //列数

private JPanel[][] gridsPanel; //面板网格

private Location direction;//方向定位

private SnakeModel snake; //贪吃蛇

private LinkedList snakeBody; //蛇的身体

private LinkedList otherBlocks; //其他区域

private LocationRO snakeHead; //蛇的头部

private LocationRO snakeFood; //目标食物

private Color bodyColor=Color.orange;//蛇的身体颜色

private Color headColor=Color.black; //蛇的头部颜色

private Color foodColor=Color.red; //目标食物颜色

private Color othersColor=Color.green;//其他区域颜色

private int gameScore=0; //总分

private long speed; //速度（难度设置）

private boolean AddScore; //加分

private Thread t; //线程

private boolean isEnd; //暂停

private static boolean notExit;

// 构造器，初始化操作

public SnakePanel(SnakeFrame parent,int row,int col){

change(parent,row,col);

}

// 开始游戏

public void newGame(long speed){

this.speed=speed;

if (notExit) {

snake.init();//初始化

}else{

snake=new SnakeModel(row,col);

notExit=true;

t=new Thread(this);

t.start();

}

requestFocus();

direction.setX(0);

direction.setY(1);

gameScore=0;

updateTextFiled(""+gameScore);

isEnd=false;

}

// 暂停游戏

public void stopGame(){

requestFocus();

isEnd=true;

}

//继续

public void returnGame(){

requestFocus();

isEnd=false;

}

//获得总分

public int getGameScore(){

return gameScore;

}

//更新总分

private void updateTextFiled(String str){

((SnakeFrame)parent).scoreField.setText(str);

}

//更新各相关单元颜色

private void updateColors(){

// 设定蛇身颜色

snakeBody=snake.getSnake();

Iterator i =snakeBody.iterator();

while(i.hasNext()){

LocationRO t=(LocationRO)(i.next());

gridsPanel[t.getX()][t.getY()].setBackground(bodyColor);

}

//设定蛇头颜色

snakeHead=snake.getSnakeHead();

gridsPanel[snakeHead.getX()][snakeHead.getY()].setBackground

(headColor);

//设定背景颜色

otherBlocks=snake.getOthers();

i =otherBlocks.iterator();

while(i.hasNext()){

LocationRO t=(LocationRO)(i.next());

gridsPanel[t.getX()][t.getY()].setBackground(othersColor);

}

//设定临时块的颜色

snakeFood=snake.getSnakeFood();

gridsPanel[snakeFood.getX()][snakeFood.getY()].setBackground

(foodColor);

}

public boolean isFocusTraversable()

{

return true;

}

//实现 Runnable 接口

public void run(){

while(true){

try{

Thread.sleep(speed);

}catch (InterruptedException e){}

if(!isEnd){

isEnd=!snake.move(direction);

updateColors();

if(snake.getAddScore()){

gameScore+=10;

updateTextFiled(""+gameScore);

}

}

}

}

//实现 KeyListener 接口

public void keyPressed(KeyEvent event)

{

int keyCode = event.getKeyCode();

if(notExit){

if (keyCode == KeyEvent.VK\_LEFT) { //向左

direction.setX(0);

direction.setY(-1);

}

else if (keyCode == KeyEvent.VK\_RIGHT) { //向右

direction.setX(0);

direction.setY(1);

}

else if (keyCode == KeyEvent.VK\_UP) { //向上

direction.setX(-1);

direction.setY(0);

}

else if (keyCode == KeyEvent.VK\_DOWN) { //向下

direction.setX(1);

direction.setY(0);

}

else if (keyCode == KeyEvent.VK\_SPACE){ //空格键

isEnd=!isEnd;

}

}

}

public void keyReleased(KeyEvent event){}

public void keyTyped(KeyEvent event){}

public void change(SnakeFrame parent,int row2, int col2) {

this.row=row2;

this.col=col2;

this.parent=parent;

gridsPanel=new JPanel[row][col];

otherBlocks=new LinkedList();

snakeBody=new LinkedList();

snakeHead=new LocationRO(0,0);

snakeFood=new LocationRO(0,0);

direction=new Location(0,1);

// 布局

setLayout(new GridLayout(row,col,1,1));

this.row=row2;

this.col=col2;

this.removeAll();

for(int i=0;i<row;i++){

for(int j=0;j<col;j++){

gridsPanel[i][j]=new JPanel();

gridsPanel[i][j].setBackground(othersColor);

add(gridsPanel[i][j]);

}

}

addKeyListener(this);

}

}

5. LocationRO 类

package SnakeGameDemo;

public class LocationRO {

private int x;

private int y;

LocationRO(int x,int y) {

this.x=x;

this.y=y;

}

int getX() {

return x;

}

int getY(){

return y;

}

public boolean equalOrRev(LocationRO e) {

return ((e.getX()==getX())&&(e.getY()==getY()))

||((e.getX()==getX())&&(e.getY()==-1\*getY()))

||((e.getX()==-1\*getX())&&(e.getY()==getY()));

}

public boolean equals(LocationRO e) {

return(e.getX()==getX())&&(e.getY()==getY());

}

public boolean reverse(LocationRO e){

return ((e.getX()==getX())&&(e.getY()==-1\*getY()))

||((e.getX()==-1\*getX())&&(e.getY()==getY()));

}

}

6. Location 类

package SnakeGameDemo;

public class Location {

private int x;

private int y;

Location(int x,int y){

this.x=x;

this.y=y;

}

int getX(){

return x;

}

int getY(){

return y;

}

void setX(int x){

this.x=x;

}

void setY(int y){

this.y=y;

}

public boolean equalOrRev(Location e){

return ((e.getX()==getX())&&(e.getY()==getY()))

||((e.getX()==getX())&&(e.getY()==-1\*getY()))

||((e.getX()==-1\*getX())&&(e.getY()==getY()));

}

public boolean equals(Location e){

return(e.getX()==getX())&&(e.getY()==getY());

}

public boolean reverse(Location e){

return ((e.getX()==getX())&&(e.getY()==-1\*getY()))

||((e.getX()==-1\*getX())&&(e.getY()==getY()));

}

}