package eluosifangkuai;

public class Block {

public final int shapes[][][] = new int[][][]{ //每种方块具有四种状态

//J型方块

{{0,1,0,0,0,1,0,0,1,1,0,0,0,0,0,0},

{1,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0},

{1,1,0,0,1,0,0,0,1,0,0,0,0,0,0,0},

{1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0}},

//L型方块

{{1,0,0,0,1,0,0,0,1,1,0,0,0,0,0,0},

{1,1,1,0,1,0,0,0,0,0,0,0,0,0,0,0},

{1,1,0,0,0,1,0,0,0,1,0,0,0,0,0,0},

{0,0,1,0,1,1,1,0,0,0,0,0,0,0,0,0}},

//S型方块

{{0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,},

{1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,},

{0,1,1,0,1,1,0,0,0,0,0,0,0,0,0,0,},

{1,0,0,0,1,1,0,0,0,1,0,0,0,0,0,0,}},

//Z型方块

{{1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0},

{0,1,0,0,1,1,0,0,1,0,0,0,0,0,0,0},

{1,1,0,0,0,1,1,0,0,0,0,0,0,0,0,0},

{0,1,0,0,1,1,0,0,1,0,0,0,0,0,0,0}},

//T型方块

{{0,1,0,0,1,1,1,0,0,0,0,0,0,0,0,0},

{1,0,0,0,1,1,0,0,1,0,0,0,0,0,0,0},

{1,1,1,0,0,1,0,0,0,0,0,0,0,0,0,0},

{0,1,0,0,1,1,0,0,0,1,0,0,0,0,0,0}},

//O型方块

{{1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0},

{1,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0}},

//I型方块

{{0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0},

{0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0},

{0,0,0,0,1,1,1,1,0,0,0,0,0,0,0,0},

{0,1,0,0,0,1,0,0,0,1,0,0,0,1,0,0}}, };

/\*\* \* x,y用来记录方块4×4区域中的（0，0）点的位置，x,y中的（0，0）相对于map中是（1,0） \*/

public int x; //方块x坐标

public int y; //方块y坐标

public int blockType=0; //方块类型

public int blockState=0; //方块状态

public int initX = 5; //初始位置的X,Y值

public int initY = 0;

GamePanel game;

/\*\*\* 构造方法\*/

Block(GamePanel game){ //引用游戏画板

this.game = game;

}

/\*\*\* 新建方（初始化）\*/

public void newblock(){

blockType = (int)(Math.random()\*1000)%7; //范围0-6

blockState = (int)(Math.random()\*1000)%4; //范围0-3

x=initX; //导入初始位置的X,Y值

y=initY;

}

/\*\*\* 把需要固定的方块固定\*存放在map数组中\*/

public void add(){

int i=0;

for(int a=0;a<4;a++){

for(int b=0;b<4;b++){

if(game.map[x+1+b][y+a]==0){ // map[列][行]

game.map[x+1+b][y+a]=shapes[blockType][blockState][i]; //将方块的四个格子保存

}

i++;

}

}

}

/\*\*\* 右移\*/

public void right() {

if(!game.isCollide(x+1,y)){ //判断是否碰撞

x++; //x坐标加1

}

game.repaint();

}

/\*\*\* 下移\*/

public void down() {

if(!game.isCollide(x,y+1)){ //判断是否碰撞

y++; //y坐标减1

}else{

add(); //碰撞到底部后，把方块添加到画布上去

game.deleteLine();//判断能否消行

newblock(); //继续创建新方块

}

game.repaint();

}

/\*\*\* 左移\*/

public void left() {

if(!game.isCollide(x-1,y)){ //判断是否碰撞

x--; //x坐标减1

}

game.repaint();

}

/\*\*\* 转换状态\*/

public void turnState() {

int temp = blockState; //首先记录本状态

blockState = (blockState+1)%4; //将状态数加一

if(game.isCollide(x,y)){ //判断是否碰撞

blockState = temp; //保存新状态

}

game.repaint();

}

}

package eluosifangkuai;

import java.awt.BasicStroke;

import java.awt.Color;

import java.awt.Font;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Image;

import java.awt.Toolkit;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.Timer;

//该类主要完成内容组件的设置

public class GamePanel extends JPanel implements KeyListener{

private static final long serialVersionUID = 1L;

public int MAPCOL=15; //游戏画布的列数

public int MAPROW=23; //游戏画布的行数

public int PIXEL=40; //像素，单位格子的长、宽

public int score; //分数

public int map[][]=new int[MAPCOL][MAPROW];

public Timer timer ; //计时器

private int Level=5; //初始难度5

private Block block; //引用Block类

private TimerListener timerlistener;

private int kind; //颜色种类

private Image image; //图片

/\*\*\* 内部类，被timer触发\*\*\*/

private class TimerListener implements ActionListener{ //实现接口

@Override

public void actionPerformed(ActionEvent e) {

if(!isCollide(block.x,block.y+1)){ //方块未触底则继续下降

block.y++;

}

else{

block.add(); //碰撞到底部后，把方块添加到画布上去

deleteLine();

if(true==isGameover()){ //返回值为true结束游戏

JOptionPane.showMessageDialog(null, "游戏结束\n你的分数是: "+score);

cleanMap(); //清空画布

drawWall(); //画围墙

score=0; //重置分数

}

block.newblock(); //新的方块

}

repaint(); //重画

}

}

/\*\*\* 构造方法\*/

GamePanel(){

block = new Block(this); //引用Block类，参数是GamePanel类

cleanMap(); //清空画布

drawWall(); //画围墙

block.newblock(); //新的方块

timerlistener=new TimerListener();

timer = new Timer(800, timerlistener); //计时器，在指定时间间隔触发TimerListener()

image = Toolkit.getDefaultToolkit().getImage( "D:/手机相册/01.jpg");//获取背景图片

}

/\*\*\* 增加难度\*最高等级10\*/

private void upLevel(){

if(Level<10){

Level++;

timer.setDelay(1300-100\*Level);

repaint();

}

}

/\*\*\* 降低难度\*最低等级1\*/

private void downLevel(){

if(Level>0){

Level--;

timer.setDelay(1300-100\*Level);

repaint();

}

}

/\*\*\* 围墙内的画面置零，相当于清除画面\*/

private void cleanMap(){

for(int i=1;i<MAPCOL-2;i++){

for(int j=0;j<MAPROW-2;j++){

map[i][j]=0;

}

}

}

/\*\*\* 画围墙，用2标示为围墙 \*/

private void drawWall(){

for(int i=0;i<MAPCOL-1;i++){

map[i][MAPROW-2]=2;

}

for(int j=0;j<MAPROW-2;j++){

map[0][j]=2;

map[MAPCOL-2][j]=2;

}

}

/\*\*\* 得到画笔来画TeTirsPanel \* 画面来源 \*/

protected void paintComponent(Graphics g) {

super.paintComponent(g); //每次调用repaint()都清除原先的组件，不调用会保留原组件

setBackground(Color.WHITE); //设置背景色

if (image != null) { //若背景图片尺寸不匹配则修改图片大小

int height = image.getHeight(this);

int width = image.getWidth(this);

if (height != -1 && height > getHeight())

height = getHeight();

if (width != -1 && width > getWidth())

width = getWidth();

int x = (int) (((double) (getWidth() - width)) / 2.0);

int y = (int) (((double) (getHeight() - height)) / 2.0);

g.drawImage(image, x, y, width, height, this);

}

/\*g.drawImage(image, 0, 0, image.getWidth(this),image.getHeight(this), this);\*/

for(int j=0;j<MAPROW-1;j++){

for(int i=0;i<MAPCOL-1;i++){

if(2==map[i][j]){

g.setColor(Color.lightGray);

g.fillRect(i\*PIXEL, j\*PIXEL, PIXEL, PIXEL); //画围墙格子

}

//画出固定好的方块

if(1==map[i][j]){

g.setColor(Color.BLACK);

g.fill3DRect(i\*PIXEL, j\*PIXEL, PIXEL, PIXEL,true);

}

}

}

//画地图上的竖线

for(int i=1;i<MAPCOL-1;i++){

g.setColor(Color.lightGray);

g.drawLine(i\*PIXEL, 0, i\*PIXEL, (MAPROW-2)\*PIXEL);

}

//画地图上的横线

for(int j=0;j<MAPROW-1;j++){

g.setColor(Color.lightGray);

g.drawLine(PIXEL\*1, j\*PIXEL, (MAPCOL-2)\*PIXEL, j\*PIXEL);

}

//画未固定方块，16是shapes数组的第一维长度

//x,y是方块正处于的坐标(提示x要加1)

//让方块颜色随机变化

kind = (int)(Math.random()\*1000)%7; //生成0~6之间的随机数以对应7种颜色

switch(kind){

case 0: {for(int i=0;i<16;i++){ //黄色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.YELLOW);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true); //在预定位置用shapes数组来画出方块

}

}

}break;

case 1: {for(int i=0;i<16;i++){ //红色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.RED);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true);

}

}

}break;

case 2: {for(int i=0;i<16;i++){ //绿色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.GREEN);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true);

}

}

}break;

case 3: {for(int i=0;i<16;i++){ //粉色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.PINK);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true);

}

}

}break;

case 4: {for(int i=0;i<16;i++){ //橙色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.ORANGE);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true);

}

}

}break;

case 5:{for(int i=0;i<16;i++){ //浅灰色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.lightGray);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true);

}

}

}break;

case 6: {for(int i=0;i<16;i++){ //青色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.CYAN);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true);

}

}

}break;

case 7:{for(int i=0;i<16;i++){ //蓝色块

if(1==block.shapes[block.blockType][block.blockState][i]){

g.setColor(Color.BLUE);

g.fill3DRect((block.x+1+i%4)\*PIXEL,(block.y+i/4)\*PIXEL,PIXEL,PIXEL,true);

}

}

}break;

}

g.setColor(Color.YELLOW); //设置颜色

g.setFont(new Font("楷体", Font.BOLD, 30)); //设置字体，粗细和大小

g.drawString(" 分数："+score,MAPCOL\*PIXEL,3\*PIXEL-35);

Graphics2D g2 = (Graphics2D)g; //开始画说明栏

g2.setStroke(new BasicStroke(3.0f));

g2.drawLine(MAPCOL\*PIXEL-30, 3\*PIXEL-85, MAPCOL\*PIXEL+170, 3\*PIXEL-85); //横线1

g2.drawLine(MAPCOL\*PIXEL-30, 3\*PIXEL, MAPCOL\*PIXEL+170, 3\*PIXEL); //横线2

g2.drawLine(MAPCOL\*PIXEL-30, 3\*PIXEL-85, MAPCOL\*PIXEL-30, 3\*PIXEL); //竖线1

g2.drawLine(MAPCOL\*PIXEL+170, 3\*PIXEL-85, MAPCOL\*PIXEL+170, 3\*PIXEL); //竖线2

g2.drawLine(MAPCOL\*PIXEL-30, 5\*PIXEL, MAPCOL\*PIXEL+170, 5\*PIXEL); //横线1

g2.drawLine(MAPCOL\*PIXEL-30, 5\*PIXEL+60, MAPCOL\*PIXEL+170, 5\*PIXEL+60); //横线2

g2.drawLine(MAPCOL\*PIXEL-30, 7\*PIXEL+60, MAPCOL\*PIXEL+170, 7\*PIXEL+60); //横线3

g2.drawLine(MAPCOL\*PIXEL-30, 9\*PIXEL+60, MAPCOL\*PIXEL+170, 9\*PIXEL+60); //横线4

g2.drawLine(MAPCOL\*PIXEL-30, 11\*PIXEL+60, MAPCOL\*PIXEL+170, 11\*PIXEL+60); //横线5

g2.drawLine(MAPCOL\*PIXEL-30, 13\*PIXEL+60, MAPCOL\*PIXEL+170, 13\*PIXEL+60); //横线6

g2.drawLine(MAPCOL\*PIXEL-30, 15\*PIXEL+60, MAPCOL\*PIXEL+170, 15\*PIXEL+60); //横线7

g2.drawLine(MAPCOL\*PIXEL-30, 17\*PIXEL+60, MAPCOL\*PIXEL+170, 17\*PIXEL+60); //横线8

g2.drawLine(MAPCOL\*PIXEL-30, 19\*PIXEL+60, MAPCOL\*PIXEL+170, 19\*PIXEL+60); //横线9

g2.drawLine(MAPCOL\*PIXEL-30, 5\*PIXEL, MAPCOL\*PIXEL-30, 19\*PIXEL+60); //竖线1

g2.drawLine(MAPCOL\*PIXEL+170, 5\*PIXEL, MAPCOL\*PIXEL+170, 19\*PIXEL+60); //竖线2

g.setFont(new Font("楷体", Font.BOLD, 24)); //向说明栏填入文字

g.drawString(" 操作方法", MAPCOL\*PIXEL, 5\*PIXEL+40);

g.drawString("↑ 翻转",MAPCOL\*PIXEL,7\*PIXEL+30);

g.drawString("↓ 下降一格",MAPCOL\*PIXEL,9\*PIXEL+30);

g.drawString("← 左移",MAPCOL\*PIXEL,11\*PIXEL+30);

g.drawString("→ 右移",MAPCOL\*PIXEL,13\*PIXEL+30);

g.drawString("F1 暂停", MAPCOL\*PIXEL, 15\*PIXEL+30);

g.drawString("F2 增加难度", MAPCOL\*PIXEL, 17\*PIXEL+30);

g.drawString("F3 降低难度", MAPCOL\*PIXEL, 19\*PIXEL+30);

g.setColor(Color.WHITE);

g.drawString("当前难度："+Level+" 相当于"+(double)(1300-Level\*100)/1000+

"秒下降一格", 2\*PIXEL, 22\*PIXEL+40);

}

@Override

public void keyPressed(KeyEvent e) {

switch(e.getKeyCode()){

case KeyEvent.VK\_UP: //按下键盘上 ↑ 时，改变方块方向

block.turnState();

break;

case KeyEvent.VK\_LEFT: //按下键盘上 ← 时，方块向左移动1格

block.left();

break;

case KeyEvent.VK\_DOWN: //按下键盘上 ↓ 时，方块向下移动1格

block.down();

break;

case KeyEvent.VK\_RIGHT: //按下键盘上 → 时，方块向右移动1格

block.right();

break;

case KeyEvent.VK\_F1: //按下键盘上 F1键 时，游戏暂停

timer.stop();

JOptionPane.showMessageDialog(null, "按确认取消暂停");

timer.restart(); //鼠标单击确认开始游戏

break;

case KeyEvent.VK\_F2: //按下键盘上 F2键 时，增加游戏难度

upLevel();

break;

case KeyEvent.VK\_F3: //按下键盘上 F3键 时，减少游戏难度

downLevel();

break;

}

}

@Override

public void keyTyped(KeyEvent e) {

}

@Override

public void keyReleased(KeyEvent e) {

}

/\*\*\* 判断是否碰撞\*\*/

public boolean isCollide(int x,int y){

for(int a=0;a<4;a++){ //遍历4×4方块区域

for(int b=0;b<4;b++){

if((block.shapes[block.blockType][block.blockState][a\*4+b]==1)&&(map[x+1+b][y+a]==1)){ //判断与已有方块是否重合

return true;

}else if((block.shapes[block.blockType][block.blockState][a\*4+b]==1)&&(map[x+1+b][y+a]==2)){ //与围墙是否相撞

return true;

}

}

}

return false;

}

/\*\*\* 判断是否结束游戏，判断条件：第一行有方块\*/

public boolean isGameover(){

for(int i=1;i<MAPCOL-3;i++){

if(map[i][0]==1){ //第一行是否有方块

return true; //有方块返回true

}

}

return false; //没有方块返回false

}

/\*\*\* 消行\*/

public void deleteLine(){

int count = 0;

for(int i=0;i<MAPROW-2;i++){

for(int j=1;j<MAPCOL-2;j++){

if(map[j][i]==1){ //判断第j行的第i个格子是否有方块

count++; //记录方块数

if(count==MAPCOL-3){ //一行都满的话，总数为MAPCOL-3个，满足则消行

score+=10; //加分

Toolkit.getDefaultToolkit().beep(); //消行提示音

for(int a=i;a>0;a--){ //从第i行开始

for(int b=1;b<MAPCOL-2;b++){

map[b][a]=map[b][a-1]; //当前行等于上一行

}

}

}

}

}

count=0;

}

}

}

package eluosifangkuai;

import javax.swing.\*;//引用JFrame包

import javax.swing.JOptionPane;

public class TetrisGame extends JFrame{

private static final long serialVersionUID = 1L;

private GamePanel t;

private static int flag=0;

TetrisGame(){

setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);//按下X按钮时结束应用程序

setLocation(550,25); //设置窗口位置

setTitle("Tetris"); //设置窗口名称

setSize(800,1000); //设置窗口大小

setResizable(true); //不可缩放

String path="D:/手机相册/封面/010.jpg"; //图片的路径

ImageIcon background=new ImageIcon(path);//背景图片

JLabel label=new JLabel(background); //把背景图片放在在一个标签里面

label.setBounds(0, 0, this.getWidth(),this.getHeight());//标签的大小位置设置

JPanel imagePanel=(JPanel)this.getContentPane();//把内容窗格转化为JPanel，否则不能用方法setOpaque()来使内容窗格透明，要强制转化

imagePanel.setOpaque(false);

this.getLayeredPane().add(label,new Integer(Integer.MIN\_VALUE));//把背景图片添加到最底层作为背景

}

public void startGame(){

t = new GamePanel(); //开始新的Panel

add(t); //将Panel添加到窗口

addKeyListener(t);

t.timer.start();

}

public static void main(String[] args){

TetrisGame tetris = new TetrisGame(); //创建新游戏

tetris.setVisible(true); //开始显示窗口

flag = JOptionPane.showConfirmDialog(tetris,"按【是】开始新游戏\n按【否】退出游戏", "新游戏", JOptionPane.YES\_NO\_OPTION); //开始选择对话框

if(flag==JOptionPane.YES\_OPTION){ //选“是”则开始游戏

tetris.startGame(); //开始游戏

tetris.setVisible(true); //显示面板

}

else{ //选“否”则退出游戏

tetris.dispose(); //关闭窗口

System.exit(0);

}

}

}