本章我们将轻松一下，一起来学习怎样使用 Java 编写一些小游戏。这些游戏是 Java 多媒体知识的具体应用，同时也是对 Java 监听器知识的进一步深化。Java 作为一门灵活的面向对象编程语言，已经成功定义了成千上万种好玩的游戏，所以就让我们一起进入 Java 的游戏世界，一起来体验 Java 程序带给我们的快乐吧！

知 识 点	难度指数	占用时间
欢乐斗地主	4	2
打豆豆	3	2
动感魔方	4	3
俄罗斯方块	6	3
贪吃蛇	5	3
拼图	4	2
寻找宝藏	6	3
开窗	4	2
猜价格	5	3
中国象棋	6	3
Java 身份证信息解读	5	3
万年历	5	2

实例 314　欢乐斗地主

【实例描述】

斗地主是一种扑克游戏。游戏最少由 3 个玩家进行，用一副 54　　连鬼牌），其中一方为地主，其余两家为另一方（农民），双方对战，先出完牌的一方获胜。地主先出牌，可以出一　　或一组合法的牌型。按逆时针顺序，下一个玩家要么不出（Pass），要么出　　和类型都相同但比上一组牌更大的牌。出牌将这样沿着牌桌持续多轮，直到连续两个玩家不出为止。具体的运行效果如图 19.1 所示。

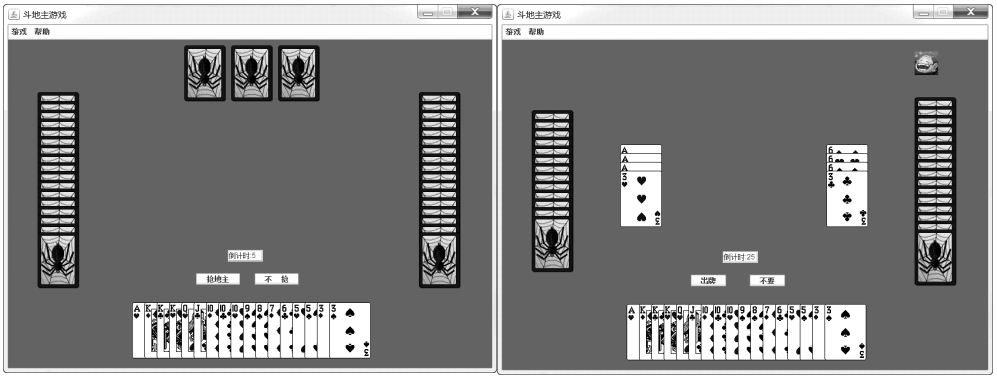

图 19.1　欢乐斗地主

【实现过程】

在 Eclipse 中新建项目 DouLandlord，并在其中创建一个 DouLandlord.java 文件用来实现斗地主游戏的基本功能。核心代码如下所示：

```
public class Card extends JLabel implements MouseListener{
    Main main;                                          // Main 类的引用
    String name;                                        // 图片 url 名字
    boolean up;                                         // 是否正反面
    boolean canClick=false;                             // 是否可被点击
    boolean clicked=false;                                  // 是否点击过
    public Card(Main m,String name,boolean up){
        this.main=m;
        this.name=name;
        this.up=up;
       if(this.up)
        this.turnFront();
       else {
            this.turnRear();
        }
        this.setSize(71, 96);
        this.setVisible(true);
        this.addMouseListener(this);
    }
    // 正面
    public void turnFront() {
        this.setIcon(new ImageIcon("images/" + name + ".gif"));
        this.up = true;
    }
    // 反面
    public void turnRear() {
        this.setIcon(new ImageIcon("images/rear.gif"));
        this.up = false;
    }
    @Override
    public void mouseClicked(MouseEvent e) {
    }
    public void mouseEntered(MouseEvent arg0) {}
    public void mouseExited(MouseEvent arg0) {}
    public void mousePressed(MouseEvent e) {
        if(canClick)
        {
            Point from=this.getLocation();
```

```
                int step;                                                   // 移动的距离
                if(clicked)
                    step=-20;
                else {
                    step=20;
                }
                clicked=!clicked;                                           // 反向
                // 当被选中的时候，向前移动一步/后退一步
                Common.move(this,from,new Point(from.x,from.y-step));
            }
        }
        public void mouseReleased(MouseEvent arg0) {
        }
}
public class Main extends JFrame implements ActionListener{
        public Container container = null;                                  // 定义容器
        JMenuItem start, exit, about;                                       // 定义菜单按钮
        JButton landlord[]=new JButton[2];                                  // 抢地主按钮
        JButton publishCard[]=new JButton[2];                               // 出牌按钮
        int dizhuFlag;                                                      // 地主标志
        int turn;
        JLabel dizhu;                                                       // 地主图标
        List<Card> currentList[] =new Vector[3];                            // 当前的出牌
        List<Card> playerList[] = new Vector[3];                            // 定义 3 个玩家表
        List<Card> lordList;                                                // 地主牌
        Card card[] = new Card[56];                                         // 定义 54 张牌
        JTextField time[]=new JTextField[3];                                // 计时器
        Time t;                                                             // 定时器（线程）
        boolean nextPlayer=false;                                           // 转换角色
        public Main(){
            Init();                                                         // 初始化
            SetMenu();                                  // 创建菜单 按钮（抢地主，发牌,计时器）
            this.setVisible(true);
            CardInit();                                                     // 发牌
            getLord();                                                      // 发完牌开始抢地主
            time[1].setVisible(true);
            // 线程安全性,把非主线程的 UI 控制放到里面
            t=new Time(this,10);                                            // 从 10 开始倒计时
            t.start();
        }
        // 抢地主
        public void getLord(){
            System.out.println(CardType.c0.toString());
            for(int i=0;i<2;i++)
                landlord[i].setVisible(true);
        }
        // 发牌洗牌
        public void CardInit() {
            int count = 1;
            // 初始化牌
            for (int i = 1; i <= 5; i++) {
                for (int j = 1; j <= 13; j++) {
                    if ((i == 5) && (j > 2))
                        break;
                    else {
                        card[count] = new Card(this, i + "-" + j, false);
                        card[count].setLocation(350, 50);
                        container.add(card[count]);
                        count++;
```

```
                }
            }
        }
        // 打乱顺序
        for(int i=0;i<100;i++){
            Random random=new Random();
            int a=random.nextInt(54)+1;
            int b=random.nextInt(54)+1;
            Card k=card[a];
            card[a]=card[b];
            card[b]=k;
        }
        for(int i=0;i<3;i++)                                    // 发完牌排序，从大到小
        {
            Common.order(playerList[i]);
            Common.rePosition(this,playerList[i],i);    // 重新定位
        }
        dizhu=new JLabel(new ImageIcon("images/dizhu.gif"));
        dizhu.setVisible(false);
        dizhu.setSize(40, 40);
        container.add(dizhu);
    }
    public void SetMenu() {                                     // 创建菜单功能按钮
        JMenuBar jMenuBar = new JMenuBar();
        JMenu game = new JMenu("游戏");
        JMenu help = new JMenu("帮助");
        start = new JMenuItem("新游戏");
        exit = new JMenuItem("退出");
        about = new JMenuItem("关于");
        landlord[0]=new JButton("抢地主");
        landlord[1]=new JButton("不抢");
        publishCard[0]= new JButton("出牌");
        publishCard[1]= new JButton("不要");
        for(int i=0;i<2;i++)
        {
            publishCard[i].setBounds(320+i*100, 400, 60, 20);
            landlord[i].setBounds(320+i*100, 400,75,20);
            container.add(landlord[i]);
            landlord[i].addActionListener(this);
            landlord[i].setVisible(false);
            container.add(publishCard[i]);
            publishCard[i].setVisible(false);
            publishCard[i].addActionListener(this);
        }
        for(int i=0;i<3;i++){
            time[i]=new JTextField("倒计时:");
            time[i].setVisible(false);
            container.add(time[i]);
        }
        for(int i=0;i<3;i++)
        {
            currentList[i]=new Vector<Card>();
        }
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == exit) {
            this.dispose();
        }
        if (e.getSource() == about) {
            JOptionPane.showMessageDialog(this, "");
```

```
            }
            ……
        public static void main(String args[]) {

            new Main();
        }
    }
        public void AI_4(List<String> model1, List<String> model2, List<Card> player,
List<String> list, int role) {
            // 排序按重复数
            player = Common.getOrder2(player);
            int len1 = model1.size();
            int len2 = model2.size();
            if (len1 < 1 || len2 < 1)
                return;
            for (int i = 0; i < len1; i++) {
                String[] s = model1.get(i).split(",");
                String[] s2 = model2.get(0).split(",");
                if ((s.length / 3 <= len2)
                        && (s.length * (3 + s2.length) == player.size())
                        && getValueInt(model1.get(i)) > Common.getValue(player
                                .get(0))) {
                    list.add(model1.get(i));
                    for (int j = 1; j <= s.length / 3; j++)
                        list.add(model2.get(len2 - j));
                }
            }
        }
        ……
        // 单牌、对子，3 个、4 个通用
        public void AI_1(List<String> model, List<Card> player, List<String> list,
                int role) {
            for (int len = model.size(), i = len - 1; i >= 0; i--) {
                if (getValueInt(model.get(i)) > Common.getValue(player.get(0))) {
                    list.add(model.get(i));
                    break;
                }
            }
        }
        // 3 带 1、2,4 带 1、2
        public void AI_2(List<String> model1, List<String> model2,
                List<Card> player, List<String> list, int role) {
            // model1 是主牌，model2 是带牌，player 是玩家出的牌，list 是准备回的牌
            // 排序按重复数
            player = Common.getOrder2(player);
            int len1 = model1.size();
            int len2 = model2.size();
            // 如果有王直接炸
            if (len1 > 0 && model1.get(0).length() < 10) {
                list.add(model1.get(0));
                System.out.println("王炸");
                return;
            }
            if (len1 < 1 || len2 < 1)
                return;
            for (int len = len1, i = len - 1; i >= 0; i--) {
                if (getValueInt(model1.get(i)) > Common.getValue(player.get(0))) {
                    list.add(model1.get(i));
                    break;
                }
            }
            list.add(model2.get(len2 - 1));
```

```
        if (list.size() < 2)
            list.clear();
    }
    // 延时，模拟时钟
    public void timeWait(int n, int player) {
        if (main.currentList[player].size() > 0)
            Common.hideCards(main.currentList[player]);
        if (player == 1)                        // 如果是我，10 秒到后直接下一家出牌
        {
            int i = n;
            while (main.nextPlayer == false && i >= 0) {
                // main.container.setComponentZOrder(main.time[player], 0);
                main.time[player].setText("倒计时:" + i);
                main.time[player].setVisible(true);
                second(1);
                i--;
            }
            if (i == -1 && player == 1) {
                // 如果我超时
                ShowCard(1);
            }
            main.nextPlayer = false;
        } else {
            for (int i = n; i >= 0; i--) {
                second(1);
                // main.container.setComponentZOrder(main.time[player], 0);
                main.time[player].setText("倒计时:" + i);
                main.time[player].setVisible(true);
            }
        }
        main.time[player].setVisible(false);
    }
    // 通过 name 估值
    public int getValueInt(String n) {
        String name[] = n.split(",");
        String s = name[0];
        int i = Integer.parseInt(s.substring(2, s.length()));
        if (s.substring(0, 1).equals("5"))
            i += 3;
        if (s.substring(2, s.length()).equals("1")
                || s.substring(2, s.length()).equals("2"))
            i += 13;
        return i;
    }
    // 判断输赢
    public boolean win() {
        for (int i = 0; i < 3; i++) {
            if (main.playerList[i].size() == 0) {
                String s;
                if (i == 1) {
                    s = "恭喜你，胜利了!";
                } else {
                    s = "恭喜电脑" + i + ",赢了! 你的智商有待提高哦";
                }
                for (int j = 0; j < main.playerList[(i + 1) % 3].size(); j++)
                    main.playerList[(i + 1) % 3].get(j).turnFront();
                for (int j = 0; j < main.playerList[(i + 2) % 3].size(); j++)
                    main.playerList[(i + 2) % 3].get(j).turnFront();
                JOptionPane.showMessageDialog(main, s);
                return true;
            }
        }
```

```
            return false;
        }
    }
```

【代码解析】

本例主要考查监听器的相关知识。实现事件处理，首先要实现监听器接口，然后调用事件源对象中的方法来添加一个监听器对象。当事件发生后，事件源会调用监听器接口中的方法，通过将事件对象传递给相应的监听器方法来实现对事件的处理。

每种事件类都有对应的事件监听器，它是事件监听器类的接口。各种事件类的接口描述如表 19-1 所示。

表 19-1　各种事件类的接口描述

事件类别	描述信息	接 口 名	方　　法
ActionEvent	激活组件	ActionListener	actionPerformed(ActionEvent)
ItemEvent	选择了某些项目	ItemListener	itemStateChanged(ItemEvent)
MouseEvent	鼠标移动	MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
	鼠标单击等	MouseListener	mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mouseClicked(MouseEvent)
KeyEvent	键盘输入	KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
FocusEvent	组件收到或失去焦点	FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
WindowEvent	窗口收到窗口级事件	WindowListener	windowClosing(WindowEvent) windowOpened(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent)

实例 315　打豆豆游戏

【实例描述】

本例中的打豆豆游戏是一种射击游戏。开枪或开炮的游戏都称为射击游戏，地上的、空中的都可以。射击游戏现在基本是朝着 3D 的方向发展。配置要求也越来越高。较早接触的那种小霸王中的“开坦克”也许是射击游戏最早的版本了。本实例我们也来看一种简单的射击游戏——打豆豆游戏，具体的运行效果如图 19.2 所示。

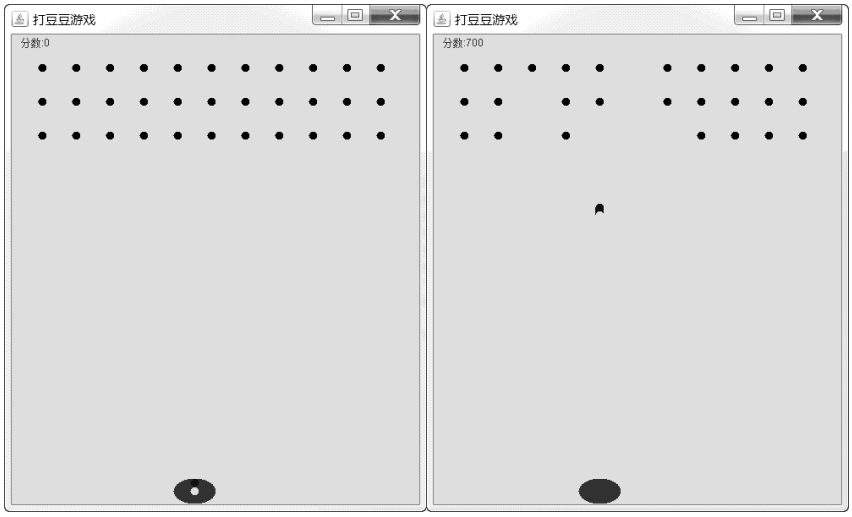

图 19.2　打豆豆游戏

【实现过程】

在 Eclipse 中新建项目 TestHoney，并在其中创建一个 TestHoney.java 文件。在该类的主方法中使用 paint()方法完成图片界面图形的变换。核心代码如下所示：

```
class Cannonball{
    static int  y=560,score=0;
    int temp = 240;
    ClassLoader classLoader = this.getClass().getClassLoader();
    public void paint(Graphics g,int x2){
        int t = 0;
        if(y==560){
            temp = x2;
        }
        g.setColor(Color.red);
        g.fillOval(temp+20, y,10, 10);
        if(y<560)
        y--;
        g.setColor(Color.LIGHT_GRAY);
        g.fillOval(temp+20, y+10, 10, 10);
        if(((temp+20)%40==0  &&  y==70  &&  TestHoney.a[0][(temp+20)/40-1]==1)||
((temp+20)%40==0 && y==110&& TestHoney.a[1][(temp+20)/40-1]==1)||((temp+20)%40==0 &&
y==150&& TestHoney.a[2][(temp+20)/40-1]==1)){
            AudioClip au = JApplet.newAudioClip(classLoader.getResource("112.wav"));
            au.play();
            g.setColor(Color.LIGHT_GRAY);
            g.fillRect(temp+20, y, 20, 30);
            if(y==70){
                t=0;
            }else if(y==110){
                t=1;
            }else if(y==150){
                t=2;
            }
            TestHoney.a[t][(temp+20)/40-1]=0;
            score+=100;
            y=560;
        }
        if(y==0){
```

```
                y=560;
            }
        }
    }
    public class TestHoney extends Frame{
        static int x1=200;
        static int [] []a={{1,1,1,1,1,1,1,1,1,1,1},
                {1,1,1,1,1,1,1,1,1,1,1},
                {1,1,1,1,1,1,1,1,1,1,1},};
        ClassLoader classLoader = this.getClass().getClassLoader();
        public TestHoney(){
            AudioClip au = JApplet.newAudioClip(classLoader.getResource("start.wav"));
            au.play();
            addWindowListener(new WindowAdapter(){
                public void windowClosing(WindowEvent e){
                    dispose();
                    System.exit(0);
                }
            });
            addKeyListener(new KeyAdapter() {
                public void keyPressed(KeyEvent e) {
                    int keycode = e.getKeyCode();
                    if (keycode == KeyEvent.VK_LEFT) {
                        x1 = x1 - 10;
                    } else if (keycode == KeyEvent.VK_RIGHT) {
                        x1 = x1 + 10;
                    }
                    else if (keycode == KeyEvent.VK_SPACE){
                        if(Cannonball.y==560){
                            AudioClip   au   =   JApplet.newAudioClip(classLoader.
getResource("BONG.wav"));
                            au.play();
                            Cannonball.y=559;
                        }else
                        {}
                    }
                    repaint();
                }
            });
        }
        public void paint(Graphics g){
            int num;
            g.setColor(Color.BLUE);
            g.drawString("分数:"+Cannonball.score, 20, 50);
            g.fillOval(x1, 560, 50, 30);
            g.setColor(Color.BLACK);
            num=0;
            for(int i=0;i<11;i++){
                if(a[0][i]==1)
                    g.fillOval(num=num+40, 70, 10, 10);
                else
                num = num+40;
            }num=0;
            for(int i=0;i<11;i++){
                if(a[1][i]==1)
                g.fillOval(num=num+40, 110, 10, 10);
                else
                num = num+40;
            }num=0;
            for(int i=0;i<11;i++){
                if(a[2][i]==1)
                g.fillOval(num=num+40, 150, 10, 10);
                else
```

```
                num = num+40;
            }
        }
        public static void main(String [] args){
            TestHoney th = new TestHoney();
            th.setBackground(Color.LIGHT_GRAY);
            th.setSize(500, 600);
            th.setTitle("打豆豆游戏");
            th.setVisible(true);
            Graphics g = th.getGraphics();
            Cannonball cb =new Cannonball();
            while(true){
                try {
                    Thread.sleep(4);
                } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
                cb.paint(g,x1);
            }
        }
    }
```

【代码解析】

本实例在设置窗体的背景图片时，继承 JPanel 自定义了自己的面板控件，并重写了面板绘制方法，为自己绘制了背景图片。面板绘制方法的声明格式如图 19.3 所示。

图 19.3　面板绘制方法声明格式

实例 316　动感魔方游戏

【实例描述】

魔方，又叫魔术方块。是匈牙利布达佩斯建筑学院厄尔诺·鲁比克教授在 1974 年发明的。魔方是由富于弹性的硬塑料制成的 6 面正方体，是一种世界范围内流行的游戏玩具，本实例我们使用 Java 来编写一个镶嵌在平面中的三维立体魔方，我们同样可以对其进行横向和竖向的旋转来使每一面成为一样的颜色。具体的运行效果如图 19.4 所示。

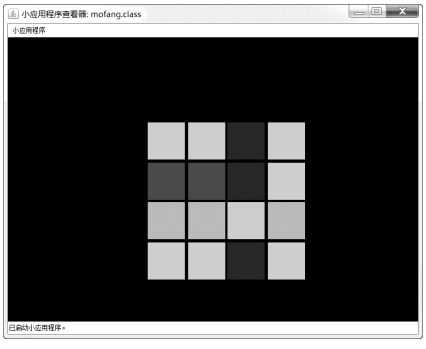

图 19.4　动感魔方游戏

【实现过程】

在 Eclipse 中新建项目 mofang，并在其中创建一个 mofang.java 文件用于实现动感模仿游戏的基本功能。核心代码如下所示：

```
    public class mofang extends Applet implements ActionListener,MouseListener,
MouseMotionListener{
        private int[][] xyz;                                    // 4 个坐标构成的面
        private double[] x,y,z;                                 // 原始点
        private double[] x1,y1,z1;                              // 旋转后的点
        private double[][] mxy={{1,0,0},{0,1,0},{0,0,1}};       // 旋转矩阵
        private int time=0;
        private int[] colors={0x70e33e,0x65f0e4,0xf20f2f,0xffff00,0x454545,
        0xaaaaaa};                                              // 六面色
        private  int 数量=4;                                    // 在此设置是（4*4）的魔方
        private int 视距=800;                                   // 越大越远
        private int 鼠标点=-1;                  // 装的是点击后得到的方块在 x、y、z 里的索引
        private int[] 鼠标点击={-1,-1};                         // 点击时的鼠标坐标
        private int[] 鼠标移动={-1,-1};
        private double[][] nou;                                 // 旋转矩阵
        private BufferedImage bi;
        private Graphics2D big;
        public void actionPerformed(ActionEvent e){            // 计时器
            if(time>0){
                for(int i=0;i<数量*数量*6;i++){
                    if(get 轴(i)){
                        for(int u=0;u<4;u++){
                            double[] hh={x[xyz[i][u]],y[xyz[i][u]],z[xyz[i][u]]};
                            double[] hjh=setMxy(hh,nou);
                            x[xyz[i][u]]=hjh[0];
                            y[xyz[i][u]]=hjh[1];
                            z[xyz[i][u]]=hjh[2];
                        }
                    }
                }
                time--;
            }
            repaint();
        }
        public void init(){                                     // 初始化数据
            double[] 临时点=new double[3];
            int i1,i2,i3,i4;
            int 直径=100;
            int 边距=8;
            int 中心点=((直径+边距)*数量-边距)/2;
            x=new double[数量*数量*数量<<3];
            y=new double[数量*数量*数量<<3];
            z=new double[数量*数量*数量<<3];
            x1=new double[数量*数量*数量<<3];
            y1=new double[数量*数量*数量<<3];
            z1=new double[数量*数量*数量<<3];
            xyz=new int[数量*数量*数量<<1][4];
            double[][] X 轴矩阵={{1,0,0},{0,0,1},{0,-1,0}};
            double[][] Y 轴矩阵={{0,0,1},{0,1,0},{-1,0,0}};
            for(i1=0;i1<数量;i1++){
                i4=i1<<2;
                x[i4+3]=x[i4]=i1*(直径+边距)-中心点;
```

```
                x[i4+1]=x[i4+2]=x[i4]+直径;
                y[i4+1]=y[i4]=中心点;
                y[i4+2]=y[i4+3]=y[i4]-直径;
                z[i4]=z[i4+1]=z[i4+2]=z[i4+3]=中心点;
            }
            ……
            addMouseMotionListener(this);
            addMouseListener(this);
            Timer t=new Timer(40,this);
            setBackground(new Color(0x00ff00));
            bi= new BufferedImage(800, 600, BufferedImage.TYPE_INT_RGB);
            big = bi.createGraphics();
            t.start();
        }
        public void paint(Graphics g){                          // 生成图像
            int i,u,o;
            Graphics2D g2=(Graphics2D)g;
            big.setColor(new Color(0x000000));
            big.fillRect(0,0,800,600);
            double[] xx1=new double[3];
            for(i=0;i<数量*数量*24;i++){
                xx1[0]=x[i];
                xx1[1]=y[i];
                xx1[2]=z[i];
                xx1=setMxy(xx1,mxy);
                x1[i]=xx1[0];
                y1[i]=xx1[1];
                z1[i]=xx1[2];
            }
            ……
                big.setColor(new Color(colors[i/数量/数量]));
                if(getabc(lx1[0],ly1[0],lx1[1],ly1[1],lx1[3],ly1[3]))
{big.fillPolygon(lx1,ly1,4);}
            }
            g2.drawImage(bi,0,0,null);
        }
        public boolean get轴(int i){
            if(取轴==0){
                return ((x[xyz[i][0]]>段号-2 && x[xyz[i][0]]<段号+2)||(x[xyz[i][1]]>
段号-2 && x[xyz[i][1]]<段号+2)  || (x[xyz[i][2]]>段号-2 && x[xyz[i][2]]<段号+2));
            }else if(取轴==1){
                return ((y[xyz[i][0]]>段号-2 && y[xyz[i][0]]<段号+2) || (y[xyz[i][1]]>
段号-2 && y[xyz[i][1]]<段号+2)  || (y[xyz[i][2]]>段号-2 && y[xyz[i][2]]<段号+2));
            }else{
                return ((z[xyz[i][0]]>段号-2 && z[xyz[i][0]]<段号+2) || (z[xyz[i][1]]>
段号-2 && z[xyz[i][1]]<段号+2)  || (z[xyz[i][2]]>段号-2 && z[xyz[i][2]]<段号+2));
            }
        }
        public double[] setMxy(double[] l,double[][] m){            // 坐标旋转
            double xx2[]={0,0,0};
            for(int u=0;u<3;u++){
                for(int o=0;o<3;o++){
                    xx2[u]+=l[o]*m[o][u];
                }
            }
            return xx2;
        }
        public double[][] getMxy(double[][] xx,double[][] yy){       // 矩阵乘法
            int i=0,u=0,o=0;
            double xx1[][]={{0,0,0},{0,0,0},{0,0,0}};
```

```
                for(i=0;i<3;i++){
                    for(u=0;u<3;u++){
                        for(o=0;o<3;o++){
                            xx1[i][u]+=xx[i][o]*yy[o][u];
                        }
                    }
                }
                return xx1;
            }
            // 判断是否是顺时针方向排列
            public boolean  getabc(double ax,double ay,double bx,double by,double cx,double
cy){
                double cax=cx-ax;
                double cay=cy-ay;
                double bcx=bx-cx;
                double bcy=by-cy;
                return cax*bcy>cay*bcx;
            }
            public void mouseClicked(MouseEvent e){}
            public void mouseEntered(MouseEvent e){}
            public void mouseExited(MouseEvent e){}
            public void mousePressed(MouseEvent e){                    // 鼠标按下时的动作
                if(e.getButton()==1){
                        鼠标点=getf(e.getX(),e.getY());
                        鼠标点击[0]=e.getX();
                        鼠标点击[1]=e.getY();
                }else if(e.getButton()==3){
                    鼠标移动[0]=e.getX();
                    鼠标移动[1]=e.getY();
                }
            }
            public double 取点(int n,double[] o){
                double li=o[xyz[n][0]];
                if(li>0){
                    for(int i=1;i<4;i++){li=(li>o[xyz[n][i]])?li:o[xyz[n][i]];}
                }else{
                    for(int i=1;i<4;i++){li=(li<o[xyz[n][i]])?li:o[xyz[n][i]];}
                }
                return li;
            }
            ……
            }
    }
```

【代码解析】

本例考查 Java 焦点事件的相关知识。焦点事件类（FocusEvent）是指用户程序界面的组件焦点发生改变（即焦点从一个对象转移到另外一个对象）时，就会发生焦点事件。得到焦点事件的组件处于激活状态。使用焦点事件必须给组件增加一个 FocusListener 接口的事件处理器，该接口包含如下两个方法，如图 19.5 所示。

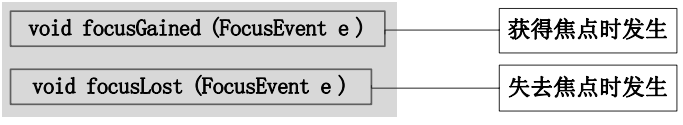

图 19.5　FocusListener 接口的方法

实例 317　俄罗斯方块游戏

【实例描述】

俄罗斯方块是一款风靡全球的游戏，它由俄罗斯人阿列克谢·帕基特诺夫发明，故得此名。俄罗斯方块的基本规则是移动、旋转和摆放游戏自动输出的各种方块，使之排列成完整的一行或多行，从而消除这些行来增加得分数。由于上手简单、老少皆宜，从而家喻户晓、风靡世界。本实例就使用 Java 来设计这样一款游戏，具体的运行效果如图 19.6 所示。

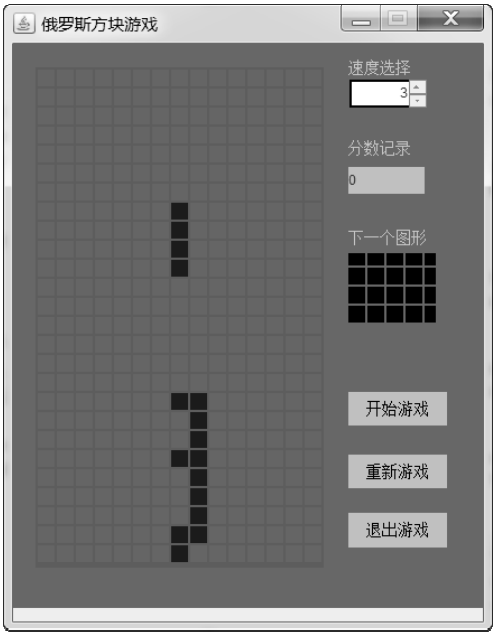

图 19.6　俄罗斯方块游戏

【实现过程】

在 Eclipse 中新建项目 RussiaGame，并编写以下几个类。每个类都单独保存在对应的.java 文件中。文件名需要和类名一致。各个类的核心代码如下所示：

```
public class GameAreaPanel extends Panel {
   private Timer timer;                          // 计时器，用于控制下落时间间隔
   private int nTime = 1000;                     // 速度，创建计时器时使用
   private Root root;                            // 要落下的方块的引用，即根类
   private GameTable gTable;                     // 创建一个游戏桌
   private int nWhich;                           // 标志创建哪一个要下落的方块
   private int tempnWhich;                       // 标志下一个要创建的方块，即提前显示下落块
   private int nScore = 0;                       // 每消一行加 10 分，用来记录总分数
   private boolean canMove = false;              // 标志是否响应键盘
   private Image myImage0, myImage1;             // 加载两个图片，用来覆盖背景和组成方块
   private boolean isRun = true;                 // 标志是开始还是暂停，一钮两用
   public GameAreaPanel(GameTable gTable) {
      myImage0 = getToolkit().getImage("b0.jpg");
      myImage1 = getToolkit().getImage("b1.jpg");
```

```
        gTable.myTable = new int[gTable.x][gTable.y];
        for (int i = 0; i < gTable.x; i++)
            for (int j = 0; j < gTable.y; j++)
                gTable.myTable[i][j] = 0;
    }
    public void Init() {
        for (int i = 0; i < gTable.x; i++)              // 重新给游戏桌置 0 标志
            for (int j = 0; j < gTable.y; j++) {
                gTable.myTable[i][j] = 0;
            }
        repaint();
    }

    public void paint(Graphics g) {
        for (int i = 0; i < gTable.x; i++)              // 根据桌子上的标记决定是否画方块
            for (int j = 0; j < gTable.y; j++) {
                if (gTable.myTable[i][j] == 1) {        // 有，画蓝方块
                    g.drawImage(myImage1, 0 + i * (15 + 2) + 2,
                              0 + j * (15 + 2) + 2, this);
                }
                else if (gTable.myTable[i][j] == 0) {   // 无，画白方块
                    g.drawImage(myImage0, 0 + i * (15 + 2) + 2,
                              0 + j * (15 + 2) + 2, this);
                }
            }
    }
    public void actionPerformed(ActionEvent e) {
        if (isRun) {                                    // 是开始
            timer.start();
            isRun = !isRun;
            canMove = true;                             // 可以响应键盘
        }
        else {                                          // 是暂停
            timer.stop();
            isRun = !isRun;
            canMove = false;                            // 不可以响应键盘
        }
    }
    public void UpdateGraph(GameTable gameTable) {
        this.gTable = gameTable;
        this.repaint();
    }
    public void update(Graphics g) {
        paint(g);                                       // 防止闪烁
    }
}
public class MainFrame
    extends JFrame {
    private Timer timer;                                // 方块下落时间间隔控制类
    private int nTime = 1000;                           // 方块下落速度
    private Root root;                                  // 方块
    private GameTable gGameAreaTable;                   // 方块下落区
    private PreviewTable gPreviewTable;                 // 方块预览区
    private int intGraph;                               // 标志创建哪一个要下落的方块
    private int intNextGraph;                           // 标志下一个要创建的方块
    private int intTotalScore = 0;                      // 用来记录总分数
    private boolean bCanMove = false;                   // 标志是否响应键盘
    private int intSpeed = 1;                           // 速度等级
    private Panel jMainPanel = new Panel();             // 添加键盘监听的 Panel
    private GameAreaPanel jGamePanel;                   // 方块下落区域
```

```
private PreviewPanel jNextGraphPanel;              // 方块预览区域
private XYLayout xyLayout = new XYLayout();        // 界面布局
private JButton jStartButton = new JButton();
……
public MainFrame() {
   try {
      setTitle("俄罗斯方块游戏");
      setLocation(200, 130);
      setSize(600, 400);
      jbInit();
      getContentPane().add(jMainPanel, "North");
      getContentPane().addKeyListener(new KeyAdapter() { // 主窗口添加键盘监听器
         public void keyPressed(KeyEvent e) {
            ProcessKeyEvent(e);
         }
      });
      addWindowListener(new WindowAdapter() {
         public void windowClosing(WindowEvent e) {
            dispose();
         }
      });
   }
   catch (Exception e) {
      e.printStackTrace();
   }
}
public static void main(String[] args) {
   MainFrame objMainFrame = new MainFrame();
   objMainFrame.pack();
   objMainFrame.setVisible(true);
   objMainFrame.validate();
   objMainFrame.setResizable(false);
}
private void jbInit() throws Exception {
  ……
   intGraph = ( ( (int) Math.round(Math.random() * 12345)) % 8) + 1;
                                                   // 产生随机方块
   switch (intGraph) {                             // 初始化产生的方块
   ……
   }
   jNextGraphPanel.PreviewGraph(intGraph);
   root.begin();                                   // 占据桌子
   intNextGraph = ( ( (int) Math.round(Math.random() * 12345)) % 8) + 1;
                                                   // 产生下一个提前显示的方块
   intGraph = intNextGraph;
   timer = new Timer(nTime, new MyRun());          // 创建计时器，调用 MyRun()
   addKeyListener(new KeyAdapter() {               // 主窗口添加键盘监听器
      public void keyPressed(KeyEvent e) {
         ProcessKeyEvent(e);
      }
   });
}
public class MyRun
   implements ActionListener {                     // 计时器调用的监听器
   public void actionPerformed(ActionEvent e) {
      if (!bCanMove)
         return;
      if (!root.down()) {       // 将方块下落一格，如果不能再下降了，就调用下面语句
         DeleteLine();                             // 消行
         jNextGraphPanel.PreviewGraph(intGraph);
```

```
            switch (intGraph) {                          // 产生新的方块
               case 1:
                  root = new One();
                  break;
            ……
               }
               intNextGraph = ( ( (int) Math.round(Math.random() * 12345)) % 8) +
                  1;
               intGraph = intNextGraph;
               if (!root.begin()) {                      // 如果失败，游戏结束
                  jGamePanel.UpdateGraph(gGameAreaTable);
                  GameResetGamePara();
               }
               else
                  jGamePanel.UpdateGraph(gGameAreaTable);
            }
            else {
               jGamePanel.UpdateGraph(gGameAreaTable);
            }
         }
      }
      public void ProcessKeyEvent(KeyEvent e) {          // 判断输入的是哪个键，并做出响应
         jNextGraphPanel.PreviewGraph(intGraph);
         switch (e.getKeyCode()) {
            case KeyEvent.VK_DOWN:          // 如果是向下键，就一下到底，并产生一个新的方块
               root.downTo();
               DeleteLine();                             // 消去添满的一行，并加 10 分
               switch (intGraph) {                       // 产生新的方块
                  case 1:
                     root = new One();
               ……
               }
               intNextGraph = ( ( (int) Math.round(Math.random() * 12345)) % 8) +
                  1;
               intGraph = intNextGraph;
               if (!root.begin()) {          // 如果新的方块产生失败，则游戏结束
                  GameResetGamePara();
               }
               else {
                  jGamePanel.UpdateGraph(gGameAreaTable);
               }
               break;
          ……
         }
      }

      public void DeleteLine() {                         // 消去添满的一行，并加 10 分
         boolean isCan;                                  // 标志是否循环消一行
         boolean isContinue = true;                      // 标志是否还有待消的行
         int k = gGameAreaTable.y - 1;
         while (isContinue) {
            isCan = true;
            while (isCan) {
               for (int i = 0; i < gGameAreaTable.x; i++) {
                  if (gGameAreaTable.myTable[i][k] != 1)
                     isCan = false;
               }
               if (isCan) {                              // 满足条件，开始消行
                  for (int i = 0; i < gGameAreaTable.x; i++)
                     gGameAreaTable.myTable[i][k] = 0;
                  for (int j = k - 1; j >= 0; j--)
```

```
                for (int i = 0; i < gGameAreaTable.x; i++) {
                    if (gGameAreaTable.myTable[i][j] == 1) {
                        gGameAreaTable.myTable[i][j + 1] = 1;
                        gGameAreaTable.myTable[i][j] = 0;
                    }
                }
            jGamePanel.UpdateGraph(gGameAreaTable);
            intTotalScore += 10;                        // 加成绩
            UpdateScore(intTotalScore);
        }
      }
      k--;
      if (k < 1)
         isContinue = false;
   }
}
public void GameResetGamePara() {                       // 游戏结束
   timer.stop();
   bCanMove = false;
   JOptionPane anOptionPane = new JOptionPane();        // 弹出对话框
   anOptionPane.showMessageDialog(this, "游戏结束，再来一次？", "Game Over！",
                              JOptionPane.WARNING_MESSAGE);
   bCanMove = false;
   ResetGamePara();                                     // 处理各个变量，以便重新开始
}
// 游戏结束后，处理各个变量，以便重新开始
public void ResetGamePara() {
   timer.stop();
   for (int i = 0; i < gGameAreaTable.x; i++)           // 重新给游戏桌置 0 标志
      for (int j = 0; j < gGameAreaTable.y; j++) {
         gGameAreaTable.myTable[i][j] = 0;
      }
   jGamePanel.UpdateGraph(gGameAreaTable);
   bCanMove = false;
   intTotalScore = 0;
   jScoreTextField.setText( (new Integer(intTotalScore)).toString());
   nTime = 1000;
   intSpeed = 1;
   jSpeedSpinner.setValue(new Integer(intSpeed));
   switch (intGraph) {                                  // 产生新的方块
      case 1:
         root = new One();
         break;
    ......
   }
   intNextGraph = ( ( (int) Math.round(Math.random() * 12345)) % 8) + 1;
   intGraph = intNextGraph;
   jGamePanel.UpdateGraph(gGameAreaTable);
   root.begin();
}
void jStartButton_actionPerformed(ActionEvent e) {
   jStartButton.setFocusable(false);
   jRestartButton.setFocusable(false);
   jExitButton.setFocusable(false);
   this.getContentPane().setFocusable(true);
   this.getContentPane().setFocusCycleRoot(true);
   bCanMove = true;
   timer = new Timer(nTime, new MyRun());
                                  // 根据选择的 nTime 间隔重新设置计时器，以变换速度
   timer.start();
}
void jRestartButton_actionPerformed(ActionEvent e) {
```

```
            timer.stop();
            jGamePanel.Init();
            bCanMove = false;
        }
        void jExitButton_actionPerformed(ActionEvent e) {
            System.exit(0);
        }
        public class SpeedChangeListener
            implements ChangeListener {                    // 速度控件监听器
            public void stateChanged(ChangeEvent e) {
                if (e.getSource() == jSpeedSpinner) {
                    String strSpeed = new String(jSpeedSpinner.getValue().toString());
                    intSpeed = Integer.parseInt(strSpeed);
                    if (intSpeed < 1)
                        intSpeed = 1;
                    if (intSpeed > 9)
                        intSpeed = 9;
                    jSpeedSpinner.setValue(new Integer(intSpeed));
                }
                UpdateSpeed();
            }
        }
        public void UpdateSpeed() {
            jSpeedSpinner.setValue(new Integer(intSpeed));
            String strSpeed = new String(jSpeedSpinner.getValue().toString());
            intSpeed = Integer.parseInt(strSpeed);
            nTime = 1000 - (intSpeed * 110);
            timer = new Timer(nTime, new MyRun());
                                        // 根据选择的 nTime 间隔重新设置计时器，以变换速度
            if (bCanMove)
                timer.start();
        }
        public void UpdateScore(int intScore) {
            if (intScore > intSpeed * 10)
                intSpeed++;
            if (intSpeed > 9)
                intSpeed = 1;
            UpdateSpeed();
            jScoreTextField.setText( (new Integer(intScore)).toString());
        }
    }
    class MainFrame_jStartButton_actionAdapter
        implements java.awt.event.ActionListener {
        MainFrame adaptee;
        MainFrame_jStartButton_actionAdapter(MainFrame adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.jStartButton_actionPerformed(e);
        }
    }
    class MainFrame_jRestartButton_actionAdapter
        implements java.awt.event.ActionListener {
        MainFrame adaptee;
        MainFrame_jRestartButton_actionAdapter(MainFrame adaptee) {
            this.adaptee = adaptee;
        }
        public void actionPerformed(ActionEvent e) {
            adaptee.jRestartButton_actionPerformed(e);
        }
    }
    class MainFrame_jExitButton_actionAdapter
        implements java.awt.event.ActionListener {
```