本电子书为某站收费 **VIP** 资源

由 灰鸽子网络资源分享社区 免费共享

灰鸽子网络资源分享社区,致力为大家提供最新最全最优秀的网络资源！
论坛主要负责提供各类优秀视频，优秀书籍以及各类优秀源码资源。
网赚，编程，逆向，建站，安全这几个大方向。其中以 网赚，编程，逆向为主打方向。

http://www.huigezi.cc < 近期论坛经常被某些收费站 **DDOS** 攻击 >

http://www.duominuo.cc < 如果论坛无法访问，请尝试其他域名 >

http://www.52dmn.com < 最好加群，第一时间获知论坛动向 >

官方群： 55059882（ 500 人群，未满 ）

（千人群，暂不开放，500 人群满后再开放）：134624974

KH北京科海 25年·IT技术出版专家

严格遴选的典型范例
方便快捷的参考手册

台湾资深开发和教育专家凝聚多年经验
挑选出最具代表性的1000多个范例

一网打尽Java日常应用开发技巧
涉及Java语法基础、图形、动画、网络、数据库、Web等领域应用

超值光盘，超量源码

- 精选**350**分钟多媒体语音教学视频，重点讲解技术难点
- 全系列**6**本书**1000**多个精华范例，全面覆盖Java开发，一查即得
- **6**个项目案例源代码，了解企业项目开发方法
- 全系列图书电子教案（PPT）
- 最新JDK及中文文档、Tomcat服务器等软件

Java 典型应用彻查1000例

图形与网络游戏开发

贾蓉生 胡大源 林金池 编著
（中国台湾）

多媒体语音教学视频
1000多个精华范例

科学出版社
北京科海电子出版社
www.khp.com.cn

Java

典型应用彻查1000例

图形与网络游戏开发

贾蓉生 胡大源 林金池 编著
（中国台湾）

科学出版社
北京科海电子出版社
www.khp.com.cn

内 容 提 要

“Java 典型应用彻查 1000 例”系列丛书以提出并解决问题为导向，通过超过 1000 个开发范例，全面介绍 Java 语言从基础到网络、数据库、游戏和 Web 开发的特性和实现方法。本系列丛书共六册，每册可独立学习，若能全部融会贯通，则效果更佳。

本书是丛书第三册，内容涵盖 Java 基础动画到网络在线游戏，循序渐进地介绍了基础图文动画、事件处理、在线游戏、2D 绘图设计、3D 绘图设计、Java Applet 与网页等知识，并通过大量范例详尽演示理论知识的实际应用。

本系列丛书的作者是中国台湾地区长期从事 Java 教学的知名教授。本书实例丰富，编排合理，可以让有初级 Java 基础的读者，从陌生到完全熟练地设计网络游戏，进而掌握 3D 立体绘图方法，适合作为 Java 网络游戏开发课程的教材。

本书光盘不仅包含全部范例源代码以及习题答案，还含有 350 分钟的教学视频，特别适合自学之用。

图书在版编目（CIP）数据

Java 典型应用彻查 1000 例. 图形与网络游戏开发/贾蓉生，胡大源，林金池编著. —北京：科学出版社，2009

ISBN 978-7-03-024486-4

I. J… II. ①贾…②胡…③林… III.①JAVA 语言—程序设计 ②计算机网络—游戏—应用程序—程序设计 IV. TP312 G899

中国版本图书馆 CIP 数据核字（2009）第 063283 号

责任编辑：何立兵 / 责任校对：杨慧芳

责任印刷：科 海 / 封面设计：林 陶

科学出版社出版

北京东黄城根北街 16 号

邮政编码：100717

http://www.sciencep.com

北京市艺辉印刷有限公司印刷

科学出版社发行 各地新华书店经销

*

2009 年 7 月 第 一 版 开本：16 开

2009 年 7 月第一次印刷 印张：21.25

印数：0 001~3 000 字数：517 000

定价：36.00 元（含 1DVD 价格）

（如有印装质量问题，我社负责调换）

推 荐 序

Java 从诞生至今已经超过 15 年，近几年来，Java 一直牢牢占据编程语言排行榜的榜首位置。时至今日，Java 已经成为每一个 IT 人无法回避的主流技术，即便是身处对面的阵营中的许多人，也会或多或少涉猎 Java 的技术。这样的形势对所有计算机及相关专业的学生，以及绝大部分的 IT 职场新人而言，并不存在要不要学 Java 的问题，而只存在如何学习 Java 的问题了。

拿到这套“Java 典型应用彻查 1000 例”，我觉得至少有两点值得高兴的地方。其一，这套书是台湾地区的大学教授根据他们在当地开设的 Java 课程而编写的，其课程涵盖了 Java 语言入门、网络应用、游戏开发、数据库应用、Web 开发等，共 6 门课程。虽然内地的大学还没有开设过切分得如此细致的 Java 课程，但这也给我带来了未来能有类似课程规模的希望。其二，这套书的编写方式，令我想起 O'Reilly 出版社久负盛名的 *Java in a Nutshell*。*Java in a Nutshell* 中文版似乎并没有得到其应有的销量，但其英文版确实一直稳居 Java 领域的前 5 名。“Java 典型应用彻查 1000 例”这套书正是采用了类似的“例说式”教学，用范例来解释每一个重要的语言或技术特性。很多抽象的概念或特性用语言描述是很不容易的，但只要附上实例，读者只要花一点点时间钻研，即可豁然开朗。本套书的覆盖面很广，其他书籍中没有提及的很多概念和特性在这里都能够找到答案。从这个意义上来说，这套书是 Java 领域中少见的“分类辞典”。

个人认为，写一本给初学者的入门书的难度不低于写作某个专门概念的高端专著。具体到 Java 语言的入门书，我自己有以下几个要求：（1）能让读者理解并掌握重要的抽象概念；（2）能让读者编写出能实际应用的程序；（3）能让读者在遇到问题的时候知道如何去查找资料并解决问题。

“Java 典型应用彻查 1000 例”这套书不仅讲解深入，范例覆盖面广，而且给出了每一个重要概念的英文名称，等于给了读者在以英文为主的技术资料海洋中冲浪的钥匙，完全可以作为要在 Java 的各个应用领域（游戏、网络、数据库、Web 开发）中入门的读者的首选读物。更可喜的是，这套书的配套光盘也是基于入门读者的普遍需求，将所有范例整合成易于查询的方式，并附带了如何下载和设置开发工具的教学视频。

这确实是一套集中了两岸作者和编辑对 Java 的诚意的作品，我诚挚向读者推荐。

罗亮
HP软件开发项目经理
2009年6月12日

前　言

自 1990 年 Sun 公司开始研发 Java 以来，至今 Java 俨然已成为最具潜力的网络程序设计语言。目前，大多数网络游戏、网上银行都是用 Java 编写的。

学 Java 程序设计首先要做的事情就是选择一本适合自己的参考书，然而令人感到尴尬的是，虽然书店中关于 Java 的书琳琅满目，但很难找到一本适合初学者系统学习的书。因为从企业对 Java 程序员的要求来看，只掌握基本的语法是远远不够的，还需要了解网络、数据库、Web 等相关知识，而目前市场上关于 Java 的书大都只介绍某一方面，不成体系。

为了让 Java 初学者少走弯路，笔者总结了 Java 教学经验，并结合多年的开发实践，编写了一系列最新 Java 实例应用丛书，包括：《Java 典型应用彻查 1000 例——Java 入门》、《Java 典型应用彻查 1000 例——网络应用开发》、《Java 典型应用彻查 1000 例——图形与网络游戏开发》、《Java 典型应用彻查 1000 例——数据库应用基础》、《Java 典型应用彻查 1000 例——网站数据库设计》、《Java 典型应用彻查 1000 例——Web 应用开发》等，让没有编程基础的 Java 读者，从最简单的语法学起，渐进到网络、数据库和 Web 应用，进而融入程序设计的主流。

主要内容

本书是本系列丛书的第三册，主要介绍 Java 绘图和网络在线游戏的开发知识，对于没有相关背景的读者，只要依次学习各章节、练习范例和习题，就能编写出比较专业的 Java 网络游戏程序。本书内容包括：

PART 01　基础图文动画

探讨 Java 基础图文动画程序的开发，包括文字的绘制、基本图形的绘制、图像文件的引用和设计简单动画。学完这部分内容后，读者具备创建网络游戏动画程序的基础框架的能力。

PART 02　事件处理

在设计网络游戏动画程序时，应考虑使用交互功能来提升精彩度，譬如使用鼠标单击位置，使用键盘键改变移动方向等。本部分内容就是以提高游戏的交互性为目的，内容包括 Java 底层事件、鼠标事件应用、键盘事件应用、消除图像闪烁和音效处理。

PART 03　网络在线游戏

本部分以弈棋对阵游戏和射击对阵游戏为例，详细介绍了网络在线游戏的开发方法。弈棋对阵游戏较为静态，射击对阵游戏较为动态，若能完整设计出这两类游戏，即表示已有能力设计实际开发中的大部分网络在线游戏。笔者强烈建议，读者在学习这部分内容之前，务必先阅读本系列丛书

第二册《Java 典型应用彻查 1000 例——网络应用开发》，掌握 Java 网络程序设计的原理，再来了解在线游戏设计的方法。

PART 04　2D 绘图设计

本部分介绍了使用 Java 进行 2D 绘图的方法，包括屏幕/视图坐标的互换，点、直线、向量、内积、法线等绘图概念，以及缩放/旋转。目的是以 2D 环境来复习必要数学知识，进而支持后面的 3D 绘图程序设计。

PART 05　3D 绘图设计

本部分介绍了 3D 绘图设计的方法，如果使用软件包，可以很轻松地设计出需要的图形效果，但终究是在使用别人的东西，无法窥得 3D 绘图的奥秘，以后在设计上也无发展潜力。以笔者多年的教学经验，能使用别人精心设计的软件包，是应当的也是必须的，但从基础入手，进行深度解析也是不可或缺的，尤其是大专院校的学生，毕业后多少要肩负一些研发的责任，更是需要厚实的基础知识。为了让读者清晰了解 3D 程序的设计过程，本书在描述上尽量按步骤进行分析，凡是涉及软件包的部分，均力求避免，以免读者错过设计的每一个细节。

PART 06　Java Applet 与网页

Java Applet 是用 Java 语言编写的一些小的应用程序，这些程序直接嵌入到页面中，由支持 Java 的浏览器（IE 或 Firefox）解释执行，能够产生特殊效果。它可大大提高 Web 交互能力和动态执行能力。本部分内容就是探讨如何开发 Applet，包括基础图文处理、动画和事件处理。

本书特色

1　**轻松入门**：本书以 Java 初学者的观点切入网络在线游戏设计，配合范例轻松解说游戏设计的基本概念和绘图开发包的用法，让零 Java 基础的读者也能轻松了解相关概念，顺利入门。

2　**范例练习**：以编程实例来介绍基础图文动画、事件处理、2D 绘图设计、3D 绘图设计、Applet 与网页等游戏开发知识，88 个实用范例包括了实际工作中需要用到的大部分问题，完整清晰地解析网络游戏开发的知识。

3　**层次深入**：运用最新的 Java 编程技术开发单机版和网络版游戏，每章都以前述内容为基础，层层深入地揭示了 Java 游戏的开发过程，使全书成为一个有机的整体。

光盘使用

本书附带 1DVD 超值光盘，内容有 Java 安装程序、范例程序、多媒体语音教学视频、PPT 电子教案和习题解答。

1　**范例程序**：放在配书光盘的\BookJavaVol_3\Program 目录中，同时还提供了本系列其他图书的源代码，涉及 Java 语法基础、图形、游戏动画、网络、数据库、Web 等应用领域。

2 **多媒体语音教学视频**：精选 350 分钟多媒体语音教学视频，放在配书光盘的\Video 目录中，重点讲解技术难点，让读者轻松掌握相关概念和操作技巧，提高学习效率。

3 **电子教案**：本书可以作为 Java 培训课程的教材或辅导书，所以特别制作了电子教案（PPT），放在配书光盘的\BookJavaVol_3\Ppt 目录中，以方便教师教学使用。

4 **习题解答**：放在配书光盘的\BookJavaVol_3\Ex 目录中，列出本书各章习题解答，随时可以测试学习效果。

5 **Java 安装程序**：放在配书光盘的\System 目录中，来源于 Sun 公司官方网站上的 Java 6.0 安装程序和中文帮助文档，无须读者下载。

读者对象

- 想进入软件开发行业，但对图形和游戏开发知识缺乏了解的初级读者。
- 具有一定的 Java 语言基础，想掌握 Java 游戏开发的专业人员。
- “Java 软件开发工程师”、“Web 开发工程师”等专业培训课程的教学用书。

本书得以编写成功，须感谢学校同仁给予的鼓励及指正，尤其感谢胡大源、林金池两位老师协助系统安装、程序测试与操作验证等工作；感谢妻子马元春协助打字、编校等工作。

贾蓉生　chia@mail.tf.edu.tw

PART

01

基础图文动画

本书是“Java典型应用彻查1000例”丛书的第三册，专门探讨Java动画游戏程序的设计，从初学入门的角度来编写，内容从Java基础绘图到网络在线游戏。笔者建议，读者在阅读本书之前，先完整阅读本系列丛书第一册《Java典型应用彻查1000例——Java入门》和第二册《Java典型应用彻查1000例——网络应用开发》，为Java程序设计打好扎实的基础。

本部分探讨Java基础图文动画程序，内容包括图案绘制、图像文件的引用与创建简易动画。

目　录

PART 01　基础图文动画

PART 02 事件处理

PART 04 2D 绘图设计

PART 05 3D 绘图设计

Chapter 01 文字绘制

本书探讨的是动画游戏，包括文字、图像和动画，这些都需要一个环境来显示。常用的显示工具是框架（Frame）与浏览器（Browser），前者可用于单机显示或多机网络对阵；后者可用于单机网络显示，本书将对它们进行详细介绍。本章将使用框架来显示基础图文。为了让读者尽早熟悉在线游戏程序的设计方式，这里介绍的虽然是简单的图文绘制，也以线程（Threads）的方式设计。

Chapter 02 基础绘图

在基础图文绘制方面，除了介绍上一章所探讨的文字绘制之外，本章将介绍基础图形的绘制方法。Graphics类提供了用于绘制各种图形的方法，包括直线、长方形、椭圆形绘制、弧线、多边形和图形处理等。

Chapter 03 图像文件引用

在前面各章中，我们探讨的图文来自临场自行绘制，而本章探讨的图文来自已经完成的图像文件（Images），包括文字、图片、照片等文件。

Chapter 04 基础动画

在前面各章中，我们探讨的是静态图片，而本章将介绍动态图片的设计，以便为在线游戏设计做准备，包括在框架中使用绘制的图案或图像；运行单幅动画或多幅动画。

Chapter 01 文字绘制

1-1 简介

本书探讨的是动画游戏，包括文字、图像和动画，这些都需要一个环境来显示。常用的显示工具是框架（Frame）与浏览器（Browser），前者可用于单机显示或多机网络对阵；后者可用于单机网络显示，本书将对它们进行详细介绍。本章将使用框架来显示基础图文。

为了让读者尽早熟悉在线游戏程序的设计方式，这里介绍的虽然是简单的图文绘制，也以线程（Threads）的方式设计。

1-2 Frame 类

java.awt.Frame 继承（extends）自 Window→Container→Component→Object，此类对象可以创建一个窗口，配合 Java 程序以提供单机图文或多机网络对阵图文的显示。

1 构造函数（用于创建新对象，请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》）

```
public Frame();
public Frame(String title);   （如范例 1）
```

其中，参数 title 是窗口左上方显示的名称。

2 实例方法（必须配合新对象使用，请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》）

```
public int getCursorType();
```

返回光标的类型。

```
public int getTitle();
```

返回框架的标题名称。

```
public boolean isResizable();
```

返回框架是否可缩放。

```
public void setCursor(int cursorType);
```

设置光标的类型。

public void **setTitle(String title);**

设置框架的标题名称。

public void **setResizable(boolean resizale);**

设置框架是可缩放的。

public void **setSize(int w, int h);** （如范例 1）

设置框架的宽度和高度。

public void **setVisible(true);** （如范例 1）

显示窗口框架。

public void **dispose();**

关闭框架，并释放有关资源。

范例 01 文件 Ex1_2_1.java 的功能是解释窗口框架的创建。

```
01 import java.awt.*;

02 class Ex1_2_1 {
03     public Ex1_2_1() {
04         Frame frame = new Frame("Ex1_2_1");
05         frame.setSize(350, 350);
06         frame.setVisible(true);
07     }

08      public static void main(String[] args) {
09             Ex1_2_1 workstart = new Ex1_2_1();
10      }
11 }
```

行 01　导入系统包 java.awt.*。
行 09　调用行 03～07 的构造函数 Ex1_2_1()。
行 04　以 Frame 类生成框架新对象 frame，并在框架左上方标示 "Ex1_2_1"。
行 05　设置框架的宽度和高度。
行 06　显示窗口框架。

运行结果

运行结果如图 1-1 所示。在 DOS 窗口中按 Ctrl+C 键可关闭框架。

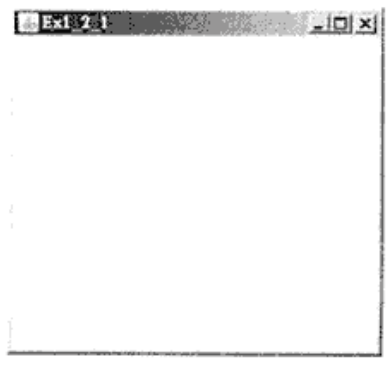


图 1-1

如果设计一个继承 Frame 类的程序，则可省略生成新对象的过程，而直接运行具有相同功能的程序代码。设计范例 2，以继承 Frame 类，注意它与范例 1 的不同之处。

范例 02 文件 Ex1_2_2.java 的功能是解释继承 Frame 类的使用方法。

```
01 import java.awt.*;

02 public class Ex1_2_2 extends Frame {
03      public Ex1_2_2() {
04           super("Ex1_2_2");
05           setSize(350, 350);
06           setVisible(true);
07      }

08      public static void main(String args[]) {
09         Ex1_2_2 workStart=new Ex1_2_2();
10      }
11 }
```

行 02　本例程序继承 Frame 类，以 Frame 类为父类，拥有其所有的成员。
行 04　设置父类的标示，即在框架左上方标示“Ex1_2_2”。

运行结果

运行结果如图 1-2 所示。

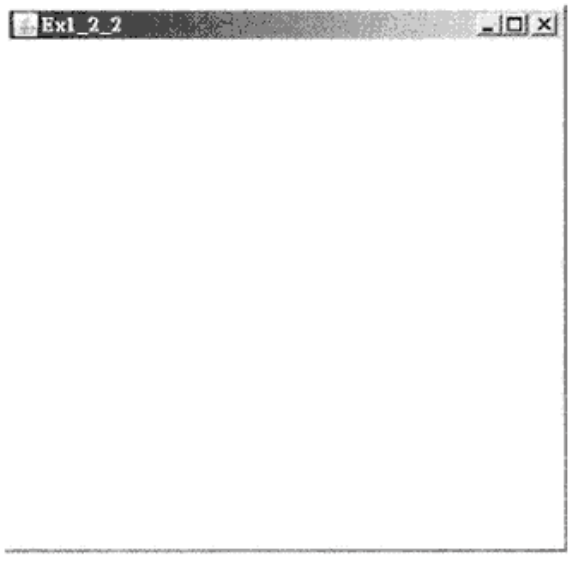


图 1-2

1-3 线程绘图流程

在绘图或游戏应用上，发生事件的区域范围不仅广大，且数量也多不胜数。在程序设计中，我们必须考虑事件线程同步并行的情况，当 CPU 能力允许时各线程竞争进入 CPU 运行，以提高运行效率。为了达到这些要求，我们使用 Thread 类、Runnable 接口和 Component 类。

每一个绘图动作即为一个线程（Threads）。在同一时刻，CPU 仅能处理一个工作。当有多个工作同时要进入 CPU 时，CPU 将其本身分割成多个工作时段，并适当地分配各工作时段。即当有多个工作同时要进入 CPU 时，哪一个工作的条件最好，该工作就可先抢到 CPU 的工作时段，这样抢 CPU 的工作时段就是线程的意义。

Runnable 接口可定义线程的行为，它与线程的关系与运行步骤如下：

（1）设计一个实现 Runnable 接口的类，其中有 run()方法。

（2）以该类生成一个新对象。

（3）将该对象指定给 Thread 构造函数的一个参数，并产生一个线程。

（4）当该线程通过 start() 开始运行时，立即调用（1）中的 run()方法。

Component 类提供了 paint(Graphics g)方法，当图像第一次显示或需要更新时，可调用此方法（请参考 2-2 节）。

Component 类还提供了 repaint()方法，触发一个重绘动作时会调用 paint(Graphics g)方法进行图案绘制。

有关上述主题更详细的说明请参考本系列丛书第二册《Java 典型应用彻查 1000 例——网络应用开发》的第 16 章。线程绘图流程如下（当读者阅读 6-6 节时，将更能体会到线程并行效果的精髓）：

```
01 public class myWork extends Frame implements Runnable {
02     public static void main(String args[]) {
03        myWork workStart=new myWork ();
04     }

05     public myWork () {
06           super("myWork");
07           setSize(350, 350);
08           setVisible(true);

09           new Thread(this).start();
10     }

11     public void run() {
12        …
13        repaint();
14     }

15     public void paint(Graphics g) {
16           …
17     }
18 }
```

行 01	设计程序 myWork。它继承 Frame 类，用于创建窗口框架；还实现了 Runnable 接口，用于启用线程。
行 02～04	调用行 05～10 的构造函数 myWork()。
行 06～08	创建窗口框架。
行 09	以线程方式并行调用行 11～14 的 run()方法。
行 13	以 repaint() 更新调用行 15～17 的 paint()方法。

1-4 Font 类

java.awt.Font 继承 (extends) 自 Object，此类对象定义字体的规格，包括字体名称（如 Times New Roman）、字体模式（如 BOLD）和字号（如 10）等。

1 构造函数（用于创建新对象，请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》）

```
public Font(String name, int style, int size);  （如范例 3）
```

定义字体的格式，其中参数 name 代表字体名称，如 Batang、Times New Roman、楷体等；参数 style 代表字体模式，如 BOLD（加粗）、ITALIC（斜体）、PLAIN（标准）；参数 size 代表字号，如 10。

2 类常量（不需要配合新对象，可直接使用，请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》）

```
public final static int BOLD;
public final static int ITALIC;
public final static int PLAIN;  （如范例 3）
```

定义字体模式的常量，BOLD（加粗）、ITALIC（斜体）、PLAIN（标准）。

3 实例方法（必须配合新对象使用，请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》）

```
public String getName();
public String getStyle();
public String getSize();
public boolean isBold();
public boolean isItalic();
public boolean isPlain();
```

返回字体的六个基本信息（如范例 4）。

范例 03 文件 Ex1_4_1.java 的功能是解释如何以 Font 类来创建字体对象？

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;
04 import java.awt.Font;

05 public class Ex1_4_1 extends Frame implements Runnable {
06     Font messageFont;
07     String message;

08     public static void main(String args[]) {
09        Ex1_4_1 workStart=new Ex1_4_1();
10     }

11     public Ex1_4_1() {
12          super("Ex1_4_1");
13          setSize(350, 350);

14          setVisible(true);
15          new Thread(this).start();
16     }

17     public void run() {
18        messageFont = new Font("TimesRoman", Font.PLAIN, 20);
19        message = "This is a test string";
20        repaint();
21     }

22     public void paint(Graphics g) {
23          g.setFont(messageFont);
```

```
24          g.drawString(message, 5, 50);
25      }
26 }
```

行 01～16	请参考 1–3 节。
行 15	以线程方式调用行 17～21 的 run()方法。
行 18	以 Font 类创建字体新对象 messageFont。其中，字体名称为 TimesRoman；字体模式为 PLAIN；字号为 20。
行 19	设置字符串。
行 20	更新调用行 22～25。
行 23	设置字符串的字体（参考 2–2 节）。
行 24	绘出字符串，其中参数的用法请参考 2–2 节。

运行结果

运行结果如图 1–3 所示。

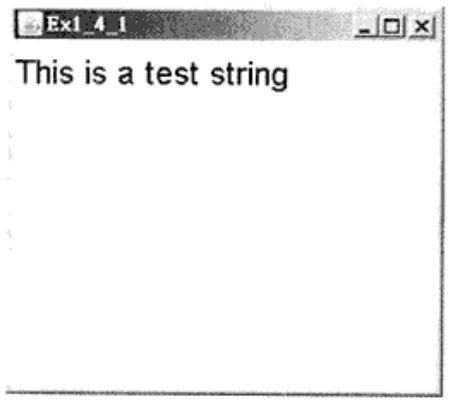


图 1-3

范例 04 文件 Ex1_4_2.java 的功能是解释 Font 类各实例方法的应用？

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;
04 import java.awt.Font;

05 public class Ex1_4_2 extends Frame implements Runnable {
06      Font messageFont;
07      String message;

08      public static void main(String args[]) {
09         Ex1_4_2 workStart=new Ex1_4_2();
10      }

11      public Ex1_4_2() {
12            super("Ex1_4_2");
13            setSize(350, 350);

14            setVisible(true);
15            new Thread(this).start();
16      }

17      public void run() {
18         messageFont = new Font("TimesRoman", Font.PLAIN, 20);
```

```
19          message = "This is a test string";

20          System.out.println("getName() : " + messageFont.getName());
21          System.out.println("getStyle() : " + messageFont.getStyle());
22          System.out.println("getSize() : " + messageFont.getSize());
23          System.out.println("isBold() : " + messageFont.isBold());
24          System.out.println("isItalic() : " + messageFont.isItalic());
25          System.out.println("isPlain() : " + messageFont.isPlain());

26          repaint();
27      }

28      public void paint(Graphics g) {
29          g.setFont(messageFont);
30          g.drawString(message, 5, 50);
31      }
32  }
```

行 20～25　返回字体的六个基本信息。

运行结果

运行结果如图 1-4 所示。

```
getName() : TimesRoman
getStyle() : 0
getSize() : 20
isBold() : false
isItalic() : false
isPlain() : true
```

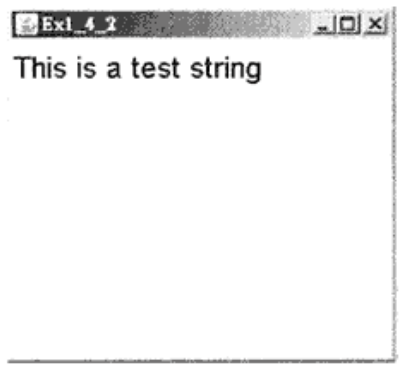


图 1-4

1-5 Color 类

java.awt.Color 继承自 Object，为 public final Class，此类对象用于定义颜色值。因为它是 final 类，所以不能被继承使用。

1　构造函数

```
public Color(int r, int g, int b);
public Color(int rgb);
public Color(float r, float g, float b);
```

其中，参数 r 代表红色；g 代表绿色；b 代表蓝色。当以 int 表示时，其值为 0～255；当以 float 表示时，其值为 0.0～1.0（如范例 5）。

2 类常量

```
public final static Color black;
public final static Color blue;
public final static Color cyan;
public final static Color darkGray;
public final static Color gray;
public final static Color lightGray;
public final static Color magenta;
public final static Color orange;
public final static Color pink;
public final static Color red;
public final static Color white;
public final static Color yellow;
```

为各颜色值的常量，如范例 6。

3 实例方法

```
public int getRed();
public int getGreen();
public int getBlue();
public int getRGB();
```

设计范例 5，解释构造函数参数的颜色值。下面三种形式均表示相同的颜色。

(1) 颜色的 int 值为 R=212，G=255，B=0。

(2) RGB= 0xd4ff00 或 RGB= −2818304。

(3) 颜色的 float 值为 R=0.83f，G=1.0f，B=0.0f。

范例 05 文件 Ex1_5_1.java 的功能是解释 Color 类的构造函数参数颜色值的设置。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;
04 import java.awt.Font;

05 public class Ex1_5_1 extends Frame implements Runnable {
06     Font messageFont;
07     String message;
08     Color color1, color2, color3;

09     public static void main(String args[]) {
10        Ex1_5_1 workStart=new Ex1_5_1();
11     }

12     public Ex1_5_1() {
13           super("Ex1_5_1");
14           setSize(350, 350);
15           setVisible(true);

16           new Thread(this).start();
17     }

18     public void run() {
19        color1 = new Color(212, 255, 0);
20        color2 = new Color(0xd4ff00);
21        color3 = new Color(0.83f, 1.0f, 0.0f);
22        messageFont = new Font("TimesRoman", Font.PLAIN, 20);
23        message = "This is a test string";
```

```
24          repaint();
25      }

26      public void paint(Graphics g) {
27          g.setFont(messageFont);
28          g.setColor(color1);
29          g.drawString(message, 5, 50);

30          g.setFont(messageFont);
31          g.setColor(color2);
32          g.drawString(message, 5, 100);

33          g.setFont(messageFont);
34          g.setColor(color3);
35          g.drawString(message, 5, 150);
36      }
37 }
```

行 19～21	配合构造函数生成新对象 color1、color2、color3，并以三种不同的方式来设置参数的颜色值。
行 28	设置字符串的颜色（参考 2-2 节）。

运行结果

运行结果如图 1-5 所示。虽然本书是黑色印刷，不能显示其他颜色，但读者可在计算机屏幕中观察。

图 1-5

范例 06 文件 Ex1_5_2.java 的功能是解释 Color 类的类常量。

```
import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class Ex1_5_2 extends Frame implements Runnable {
    Font messageFont;
    String message;

    public static void main(String args[]) {
        Ex1_5_2 workStart=new Ex1_5_2();
    }
```

```
12      public Ex1_5_2() {
13          super("Ex1_5_2");
14          setSize(350, 350);

15          setVisible(true);
16          new Thread(this).start();
17      }

18      public void run() {
19        messageFont = new Font("TimesRoman", Font.PLAIN, 30);
20        message = "This is a test string";
21        repaint();
22      }

23      public void paint(Graphics g) {
24          g.setFont(messageFont);
25          g.setColor(Color.blue);
26          g.drawString(message, 5, 50);
27      }
28  }
```

行 25　blue 是蓝色值的常量，因为是类常量，所以不需配合新对象的实例，可直接以 Color.blue 的形式使用。

运行结果

运行结果如图 1-6 所示。

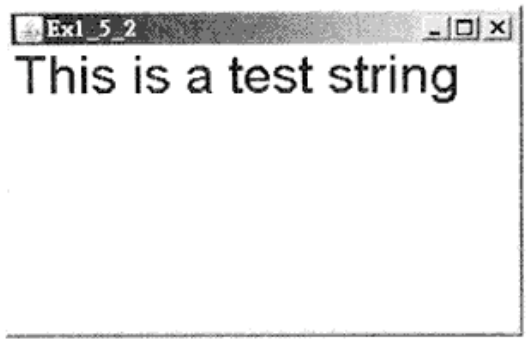


图 1-6

范例 06.1　文件 Ex1_5_3.java 的功能是解释 Color 类读取颜色值的实例方法，运行时必须配合新对象的实例才可使用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;
04 import java.awt.Font;

05 public class Ex1_5_3 extends Frame implements Runnable {
06      Font messageFont;
07      String message;
08      Color color;

09      public static void main(String args[]) {
10        Ex1_5_3 workStart=new Ex1_5_3();
11      }

12      public Ex1_5_3() {
```

```
13          super("Ex1_5_3");
14          setSize(350, 350);
15          setVisible(true);

16          new Thread(this).start();
17      }

18      public void run() {
19          color = new Color(212, 255, 0);
20          messageFont = new Font("TimesRoman", Font.PLAIN, 20);
21          message = "This is a test string";

22          System.out.println("getRed() : " + color.getRed());
23          System.out.println("getGreen() : " + color.getGreen());
24          System.out.println("getBlue() : " + color.getBlue());
25          System.out.println("getRGB() : " + color.getRGB());
26          repaint();
27      }

28      public void paint(Graphics g) {
29          g.setFont(messageFont);
30          g.setColor(color);
31          g.drawString(message, 5, 50);
32      }
33  }
```

行 19　　生成新对象 color。

行 22～25　　读取新对象 color 的颜色值。因为是实例方法，所以必须配合新对象的实例才可使用。

运行结果

运行结果如图 1-7 所示。

```
getRed() : 212
getGreen() : 255
getBlue() : 0
getRGB() : -2818304
```

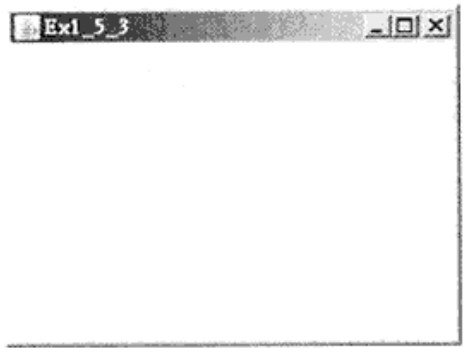


图 1-7

1-6 中文处理

在 1-4 节已经介绍了 Font 类的功能，其生成的对象可定义字体的格式，构造函数参数 name 为字体名称，如 Batang、Times New Roman 楷体等；参数 style 为字体模式，如 BOLD（加粗）、ITALIC（斜体）、PLAIN（标准）；参数 size 为字号，如 10、12 等。

当设置中文字体时，应使用中文字体名称字符串，如"宋体"、"楷体"等；至于字体模式和字号，与英文字体相同。

范例 07 文件 Ex1_6_1.java 的功能是解释中文字体的绘制。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;
04 import java.awt.Font;
05 import java.awt.Color;

06 public class Ex1_6_1 extends Frame implements Runnable {
07     Font messageFont1;
08     Font messageFont2;
09     String message;

10     public static void main(String args[]) {
11        Ex1_6_1 workStart=new Ex1_6_1();
12     }

13     public Ex1_6_1() {
14          super("Ex1_6_1");
15          setSize(350, 350);

16          setVisible(true);
17          new Thread(this).start();
18     }

19     public void run() {
20        messageFont1 = new Font("宋体", Font.PLAIN, 30);
21        messageFont2 = new Font("楷体", Font.PLAIN, 30);
22        message = "中文字符串";
23        repaint();
24     }

25     public void paint(Graphics g) {
26          g.setFont(messageFont1);
27          g.setColor(Color.blue);
28          g.drawString(message, 5, 50);

29          g.setFont(messageFont2);
30          g.setColor(Color.green);
31          g.drawString(message, 5, 100);
32     }
33 }
```

行 20　设置中文字体名称。
行 26　设置字符串的字体。

运行结果

运行结果如图1-8所示。

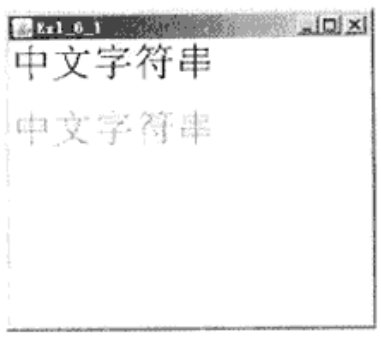


图 1-8

同时设置中文/英文的字体名称或两种字体模式时，Java 可以用“+”号来兼容两种设置，如“宋体”+TimesRoman 或 Font.ITALIC + Font.PLAIN。

范例 08 文件 Ex1_6_2.java 的功能是解释同时设置中文/英文的字体名称或两种字体模式。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;
04 import java.awt.Font;
05 import java.awt.Color;

06 public class Ex1_6_2 extends Frame implements Runnable {
07      Font messageFont1, messageFont2, messageFont3, messageFont4;
08      String message;

09      public static void main(String args[]) {
10          Ex1_6_2 workStart=new Ex1_6_2();
11      }

12      public Ex1_6_2() {
13          super("Ex1_6_2");
14          setSize(350, 350);

15          setVisible(true);
16          new Thread(this).start();
17      }

18      public void run() {
19          messageFont1 = new Font("宋体"+"TimesRoman", Font.PLAIN, 30);
20          messageFont2 = new Font("宋体"+"Monotype Corsiva", Font.PLAIN, 30);
21          messageFont3 = new Font("Monotype Corsiva", Font.PLAIN, 30);
22          messageFont4 = new Font("楷体"+"Monotype Corsiva", Font.ITALIC + Font.PLAIN, 30);
23          message = "中文字符串  English String";

24          repaint();
25      }

26      public void paint(Graphics g) {
27          g.setFont(messageFont1);
28          g.setColor(Color.blue);
29          g.drawString(message, 5, 50);

30          g.setFont(messageFont2);
31          g.setColor(Color.gray);
32          g.drawString(message, 5, 100);

33          g.setFont(messageFont3);
34          g.setColor(Color.red);
```

```
35          g.drawString(message, 5, 150);

36          g.setFont(messageFont4);
37          g.setColor(Color.yellow);
38          g.drawString(message, 5, 200);
39      }
40 }
```

行 19～22　同时设置中文/英文的字体名称或两种字体模式。
行 27　设置字符串的字体。

运行结果

运行结果如图 1–9 所示。

图 1-9

讨论事项

1　设置中英综合字体名称（如行 19、行 20）时，仅中文字体起作用，英文字体无作用。
2　若仅设置英文字体名称，则无法显示中文字符串（如行 21）。
3　当设置综合字体模式时（如行 22），对中英两种字体均起作用。

1-7 习题

1．Frame 类的功能是什么？
2．构造函数 public Frame(String title) 的参数有何意义？
3．如果设计一个继承 Frame 类的程序，与以 Frame 类生成新对象相比，它有何优势？
4．试论述线程的意义。
5．当线程以 start() 开始运行时，将调用哪一个方法运行？
6．Component 类提供的 paint(Graphics g)方法的功能是什么？
7．Font 类的功能是什么？
8．Color 类的功能是什么？
9．如何设置颜色值？
10．设置中文字体时，如何进行字体设置？
11．同时设置中文/英文字体时，如何运行？
12．同时设置中文/英文字体时，有哪些注意事项？

Chapter 02 基础绘图

2-1 简介

在基础图文绘制方面，除了介绍上一章所探讨的文字绘制之外，本章将介绍基础图形的绘制方法。Graphics 类提供了用于绘制各种图形的方法，包括直线、长方形、椭圆形绘制、弧线、多边形和图形处理等。

2-2 Graphics 类

java.awt.Graphics 继承（extends）自 Object，为 abstract public class（有关抽象类的内容请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》），此类提供了所有 Java 的绘图功能。因为它是抽象类，所以无法借助构造函数生成新的实例对象直接绘图，但可通过 Component 类的 paint()方法参与绘制。下面介绍 Graphics 类的实例方法。

1 字符串绘制

public abstract void **drawString(String str, int x, int y);**

绘制字符串。其中，参数 str 是设置的字符串；(x，y) 是字符串起始点的坐标（如范例 5）。

2 直线绘制

public abstract **void drawLine(int x1, int y1, int x2, int y2);**

绘制直线。其中，参数(x1，y1)是直线起始点的坐标；(x2，y2)是直线终止点的坐标（如范例 9）。

3 长方形绘制

public abstract void **drawRect(int x, int y, int width, int height);**
public abstract void **fillRect(int x, int y, int width, int height);**

drawRect() 绘制长方形。其中，参数(x，y)是长方形左上角的坐标；width 是长方形的宽；height 是长方形的高。

fillRect()将前景色填入长方形中（如范例 10)。

4 椭圆形绘制

```
public abstract void drawOval(int x, int y, int width, int height);
public abstract void fillOval(int x, int y, int width, int height);
```

drawOval() 绘制椭圆。其中，参数(x，y)是椭圆长方形框左上角的坐标；width 是椭圆的宽；height 是椭圆形的高。

fillOval()将前景色填入椭圆中（如范例 11)。

5 弧线绘制

```
public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle);
public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle);
```

drawArc() 绘制弧线。其中，参数(x，y)类似于长方形左上角的坐标；width 是椭圆的宽；height 是椭圆的高；startAngle 是弧线的起始角度；arcAngle 是截取的椭圆边缘的角度。

fillArc()将前景色填入扇形中（如范例 12)。

6 多边形绘制

```
public abstract void drawPolygon(int[] xpoints, int[] ypoints, int npoints);
public abstract void fillPolygon(int[] xpoints, int[] ypoints, int npoints);
```

drawPolygon() 绘制多边形。其中，参数 xpoints 是各点的 x 轴坐标；ypoints 是各点的 y 轴坐标；npoints 是多边形点的数量。

fillPolygon()将前景色填入多边形中（如范例 13)。

7 清除图形

```
public abstract void clearRect(int x, int y, int width, int height);
```

以背景色清除长方形区域内的图形。其中，参数(x，y)是长方形左上角的坐标；width 是长方形的宽；height 是长方形的高。

8 剪裁图形

```
public abstract void clipRect(int x, int y, int width, int height);
```

截取长方形区域内的图形。其中，参数(x，y)是长方形左上角的坐标；width 是长方形的宽；height 是长方形的高（如范例 14)。

9 复制图形

```
public abstract void copyArea(int x, int y, int width, int height, int dx, int dy);
```

复制长方形区域内的图形到(x+dx,y+dy)。其中，参数(x,y)是原长方形左上角的坐标；(x+dx,y+dy)是复制长方形左上角的坐标；width 是长方形的宽；height 是长方形的高（如范例 15)。

10 设置颜色

```
public abstract void setColor(Color c);
public abstract void getColor();
```

设置/读取当前的前景色（如范例 5)。

11 设置字体

```
public abstract void setFont(Font fount);
public abstract void getFont();
```

设置/读取当前的字体（如范例 3）。

2-3 直线绘制

用 Graphics 类的方法 public abstract void drawLine(int x1, int y1, int x2, int y2)来绘制直线。其中，参数(x1, y1)是直线起始点的坐标；(x2, y2)是直线终止点的坐标。

范例 09 文件 Ex2_3.java 的功能是解释如何绘制直线。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex2_3 extends Frame implements Runnable {
05     public static void main(String args[]) {
06       Ex2_3 workStart=new Ex2_3();
07     }

08     public Ex2_3() {
09         super("Ex2_3");
10         setSize(350, 350);

11         setVisible(true);
12         new Thread(this).start();
13     }
14     public void run() {
15       repaint();
16     }

17     public void paint(Graphics g) {
18       g.drawLine(50,50,120,100);

19       g.drawLine(10,250,50,200);
20       g.drawLine(50,200,120,270);
21       g.drawLine(120,270,200,160);
22     }
23 }
```

行 18　　绘制直线。
行 19－21　绘制折线（由 3 条直线组成）。

运行结果

运行结果如图 2-1 所示。

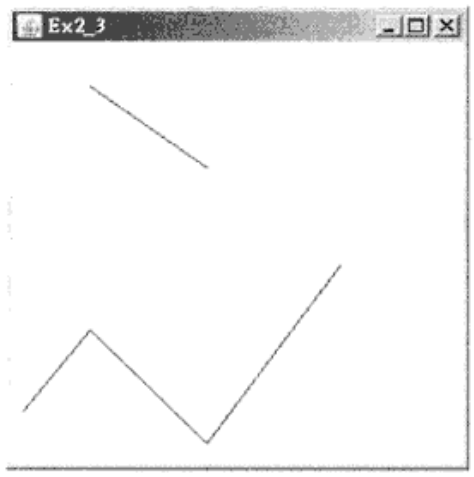


图 2-1

2-4 长方形绘制

用 Graphics 类的方法 public abstract void drawRect(int x, int y, int width, int height)来绘制长方形。其中，参数(x, y)是长方形左上角的坐标；width 是长方形的宽；height 是长方形的高。

通过 public abstract void fillRect(int x, int y, int width, int height)可以将前景色填入长方形中。

范例 10 文件 Ex2_4.java 的功能是解释如何绘制长方形。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex2_4 extends Frame implements Runnable {
05     public static void main(String args[]) {
06        Ex2_4 workStart=new Ex2_4();
07     }

08     public Ex2_4() {
09           super("Ex2_4");
10           setSize(350, 350);

11           setVisible(true);
12           new Thread(this).start();
13     }

14     public void run() {
15        repaint();
16     }

17     public void paint(Graphics g) {
18        g.drawRect(90,50,150,100);
19        g.fillRect(90,200,150,100);
20     }
21 }
```

行 18 绘制长方形。

行 19 绘制以前景色填充的长方形。

运行结果

运行结果如图 2-2 所示。

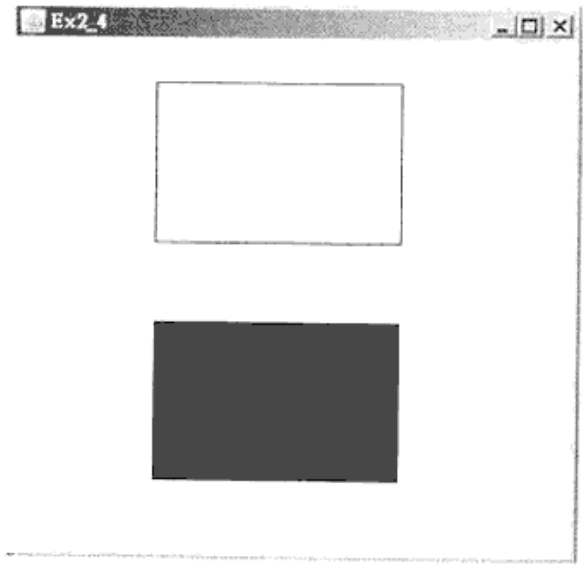


图 2-2

2-5 椭圆形绘制

用 Graphics 类的方法 public abstract void drawOval(int x, int y, int width, int height)来绘制椭圆。其中，参数(x, y)是椭圆长方形框左上角的坐标；width 是椭圆的宽；height 是椭圆的高。

通过 public abstract void fillOval(int x, int y, int width, int height)可以将前景色填入椭圆中。

范例 11 文件 Ex2_5.java 的功能是解释如何绘制椭圆。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex2_5 extends Frame implements Runnable {
05     public static void main(String args[]) {
06        Ex2_5 workStart=new Ex2_5();
07     }

08     public Ex2_5() {
09          super("Ex2_5");
10          setSize(350, 350);

11          setVisible(true);
12          new Thread(this).start();
13     }

14     public void run() {
15        repaint();
16     }

17     public void paint(Graphics g) {
18        g.drawOval(35,65,100,60);

19        g.drawOval(170,65,100,60);
20        g.drawRect(170,65,100,60);
```

```
21        g.fillOval(35,200,100,60);
22    }
23 }
```

行 18	绘制椭圆。
行 19～20	绘制椭圆及其长方形框架。
行 21	绘制以前景色填充的椭圆。

运行结果

运行结果如图 2-3 所示。

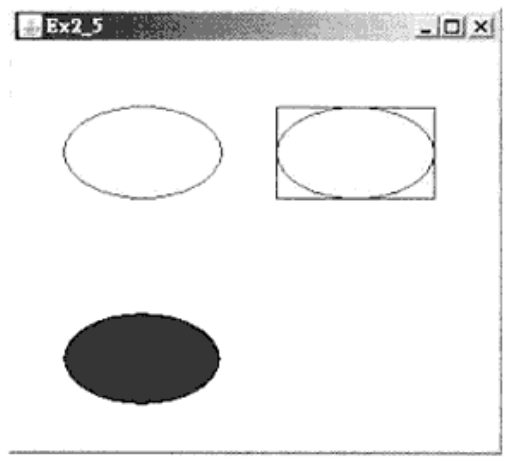


图 2-3

2-6 弧线绘制

用 Graphics 类的方法 public abstract void drawArc(int x, int y, int width, int height, int startAngle, int arcAngle)来绘制弧线。其中，参数(x, y) 类似于长方形左上角的坐标；width 是椭圆的宽；height 是椭圆的高；startAngle 是弧线的起始角度；arcAngle 是截取的椭圆边缘的角度。

通过 public abstract void fillArc(int x, int y, int width, int height, int startAngle, int arcAngle) 可以将前景色填入扇形中。

范例 12 文件 Ex2_6.java 的功能是解释如何绘制弧线。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex2_6 extends Frame implements Runnable {
05     public static void main(String args[]) {
06        Ex2_6 workStart=new Ex2_6();
07     }

08     public Ex2_6() {
09          super("Ex2_6");
10          setSize(350, 350);

11          setVisible(true);
12          new Thread(this).start();
13     }
```

```
14     public void run() {
15         repaint();
16     }

17     public void paint(Graphics g) {
18         g.drawArc(35,65,100,60,10,90);

19         g.drawArc(170,65,100,60,10,90);
20         g.drawRect(170,65,100,60);

21         g.fillArc(35,200,100,60,10,90);
22     }
23 }
```

行 18	绘制弧线。
行 19～20	绘制弧线及其长方形框架。
行 21	绘制以前景色填充的扇形。

运行结果

运行结果如图 2-4 所示。

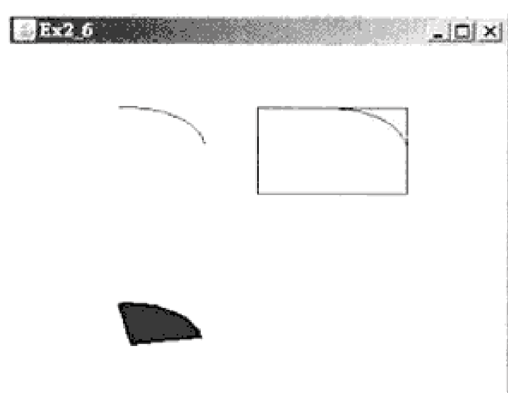


图 2-4

2-7 多边形绘制

用 Graphics 类的方法 public abstract void drawPolygon(int[] xpoints, int[] ypoints, int npoints) 来绘制多边形。其中，参数 xpoints 是各点的 x 轴坐标；ypoints 是各点的 y 轴坐标；npoints 是多边形点的数量。

通过 public abstract void fillPolygon(int[] xpoints, int[] ypoints, int npoints)可以将前景色填入多边形中。

范例 13 文件 Ex2_7.java 的功能是解释如何绘制多边形。

```
import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;

public class Ex2_7 extends Frame implements Runnable {
    int[] pgx = {90,130,180,230,180,130,90};
    int[] pgy = {100,60,60,100,150,150,100};
    int pgn = 7;
```

```
08      int[] fpgx = {90,130,180,230,180,130,90};
09      int[] fpgy = {250,210,210,250,300,300,250};
10      int fpgn = 7;

11      public static void main(String args[]) {
12         Ex2_7 workStart=new Ex2_7();
13      }

14      public Ex2_7() {
15            super("Ex2_7");
16            setSize(350, 350);

17            setVisible(true);
18            new Thread(this).start();
19      }

20      public void run() {
21         repaint();
22      }

23      public void paint(Graphics g) {
24         g.drawPolygon(pgx, pgy, pgn);
25         g.fillPolygon(fpgx, fpgy, fpgn);
26      }
27 }
```

行 05～07	多边形各点声明与初始化。
行 05	各点 x 轴坐标的数组声明与初始化。
行 06	各点 y 轴坐标的数组声明与初始化。
行 07	多边形点的数量声明与初始化。
行 08～10	填充多边形各点声明与初始化，意义同行 05～07。
行 24	绘制多边形。
行 25	绘制以前景色填充的多边形。

运行结果

运行结果如图 2-5 所示。

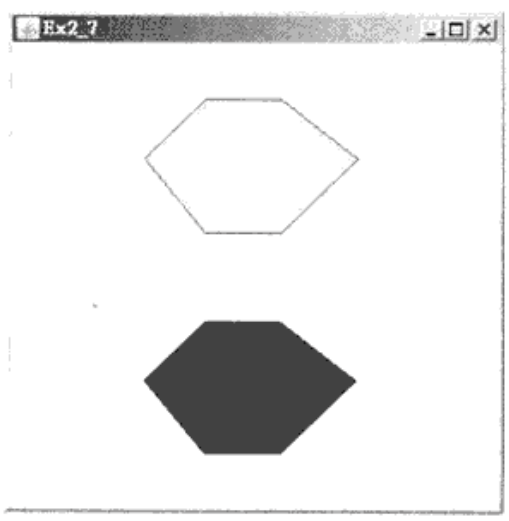


图 2-5

2-8 图形剪裁

用 Graphics 类的方法 public abstract void clipRect(int x, int y, int width, int height)来截取长方形区域内的图形。其中，参数(x, y)是长方形左上角的坐标；width 是长方形的宽；height 是长方形的高。

范例 14 参考范例 13，文件 Ex2_8.java 的功能是解释如何进行图形剪裁。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex2_8 extends Frame implements Runnable {
05      int[] pgx = {90,130,180,230,180,130,90};
06      int[] pgy = {100,60,60,100,150,150,100};
07      int pgn = 7;

08      int[] fpgx = {90,130,180,230,180,130,90};
09      int[] fpgy = {250,210,210,250,300,300,250};
10      int fpgn = 7;

11      public static void main(String args[]) {
12         Ex2_8 workStart=new Ex2_8();
13      }

14      public Ex2_8() {
15            super("Ex2_8");
16            setSize(350, 350);

17            setVisible(true);
18            new Thread(this).start();
19      }

20      public void run() {
21         repaint();
22      }

23      public void paint(Graphics g) {
24         g.drawRect(100,80,120,50);
25         g.drawPolygon(pgx, pgy, pgn);

26         g.clipRect(100,230,120,50);
27         g.fillPolygon(fpgx, fpgy, fpgn);
28      }
29 }
```

行 24～25　绘出图形剪裁的参考范围。
行 26～27　剪裁图形并进行填充。

运行结果

运行结果如图 2-6 所示。

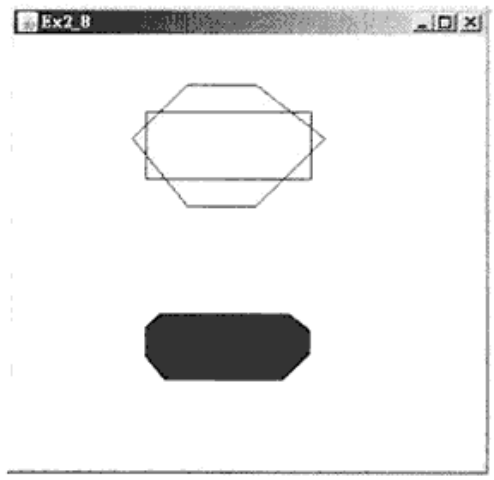


图 2-6

2-9 图形复制

用 Graphics 类的方法 public abstract void copyArea(int x, int y, int width, int height, int dx, int dy)复制长方形区域内的图形到(x+dx, y+dy)。其中，参数(x, y)是原长方形左上角的坐标；(x+dx, y+dy)是复制长方形左上角的坐标；width 是长方形的宽；height 是长方形的高。

范例 15 参考范例 11，文件 Ex2_9.java 的功能是解释如何进行图形复制。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex2_9 extends Frame implements Runnable {
05      public static void main(String args[]) {
06        Ex2_9 workStart=new Ex2_9();
07      }

08      public Ex2_9() {
09          super("Ex2_9");
10          setSize(350, 350);

11          setVisible(true);
12          new Thread(this).start();
13      }

14      public void run() {
15        repaint();
16      }

17      public void paint(Graphics g) {
18        g.drawOval(35,65,100,60);

19        g.drawOval(170,65,100,60);
20        g.drawRect(170,65,100,60);

21        g.fillOval(35,200,100,60);
22        g.copyArea(35,200,100,60,135,0);
23      }
24 }
```

行 22　将填充椭圆复制到右下角。

运行结果

运行结果如图 2-7 所示。

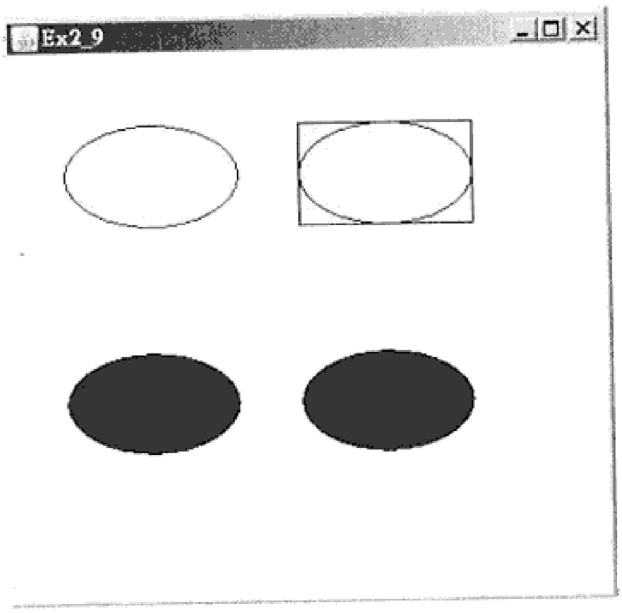


图 2-7

2-10 习题

1．Graphics 类有哪些功能？

2．Graphics 类是抽象类，所以无法生成实例对象直接绘图，那么它是如何绘图的？

3．drawString(String str，int x，int y) 绘制字符串，其中的参数各表示什么？

4．drawLine(int x1，int y1，int x2，int y2) 绘制直线，其中的参数各表示什么？

5．drawRect(int x，int y，int width，int height) 绘制长方形，其中的参数各表示什么？

6．drawOval(int x，int y，int width，int height) 绘制椭圆，其中的参数各表示什么？

7．drawArc(int x，int y，int width，int height，int startAngle，int arcAngle) 绘制弧线，其中的参数各表示什么？

8．drawPolygon(int[] xpoints，int[] ypoints，int npoints) 绘制多边形，其中的参数各表示什么？

9．clearRect(int x，int y，int width，int height) 以背景色清除长方形区域内的图形，其中的参数各表示什么？

10．clipRect(int x，int y，int width，int height) 截取长方形区域内的图形，其中的参数各表示什么？

11．copyArea(int x，int y，int width，int height，int dx，int dy) 复制长方形区域内的图形到 (x+dx，y+dy)，其中的参数各表示什么？

Chapter 03 图像文件引用

3-1 简介

在前面各章中，我们探讨的图文来自临场自行绘制，而本章探讨的图文来自已经完成的图像文件（Images），包括文字、图片、照片等文件。

3-2 图像文件格式

一般显示在计算机屏幕上的图像文件格式有两种：点阵格式（Raster）与向量格式（Vector）。前者以点阵描绘图像，每一个像素对应图像中特定的位置，当图像放大时密度会降低，图像趋于模糊；后者以几何图形描绘图像，当图像放大时不会改变图像的质量。目前 Java 可使用的图像文件格式为点阵格式，包括 GIF、PNG 和 JPEG(JPG)。

- GIF 格式提供透明与不透明的图像显示功能，具有 8 位（256）以下的调色功能，因为其色谱幅度不大，所以压缩时颜色不会轻易失真，可进行非破坏性压缩。此类格式适合绘制图形（如卡通动画）。
- PNG 格式支持透明、半透明与不透明的图像显示功能，具有 8～24 位的调色功能，与 GIF 相同可进行非破坏性压缩，因此可用来替代 GIF 格式。
- JPEG(JPG) 格式仅支持不透明的图像显示功能和 24 位调色功能，所以压缩时颜色会轻易失真，成为破坏性压缩。此类格式适合图像照片。

3-3 图像读取与 Toolkit 类

图像的显示环境有两种：本机框架（Frame）与 Web 浏览器（Browser），前者可用于单机显示或多机网络对阵；后者可用于单机网络显示。本章将先介绍前者。Toolkit 类提供图像读取功能。

Java.awt.Toolkit 继承自 Object，为 Abstract public class，因为它是抽象类，所以无法借助构造函数生成新对象读取图像，但可借助类方法 getDefaultToolkit()创建新对象，再以实例方法 getImage()读取图像。

1 类方法

```
public static Toolkit getDefaultToolkit();
```

因为它是类方法，所以不需配合构造函数生成新对象来使用，况且 Toolkit 类也没有构造函数（系统赋予的空白隐性构造函数不在此列，请参考本系列丛书第一册《Java 典型应用彻查 1000

例——Java 入门》)。但此方法可直接生成一个引用对象，如下面的 tk。

Toolkit tk = Toolkit.getDefaultToolkit()；　（如范例 16）

2 实例方法

```
public abstract Image getImage(String filename);
public abstract Image getImage(URL url);
```

读取图像文件。其中，参数 filename 是 GIF、PNG、JPEG（JPG）文件的名称；url 用于 Web 浏览器（另在后面详述）。下面示范配合类方法生成的对象 tk 来读取图像文件。

Image = tk.getImage(String filename)；（如范例 16）

3-4 图像绘制与 Graphics 类

在 2-2 节中，我们已经详述了 Graphics 类的绘图功能，除了基础绘图方法之外，Graphics 类还支持图像图片绘制，相应的实例方法如下：

```
public abstract void drawImage(Image img, int x, int y, this);    （如范例 16）
public abstract void drawImage(Image img, int x, int y, int width, int height, this);
```

其中，参数 img 是图像图片；(x, y)是图片对应长方形框左上角的坐标；width 是图片的宽；height 是图片的高；this 是在代码块内对图片设置内容的关键字。

设计范例 16，使用实例方法 drawImage(Image img, int x, int y, this)绘制图像，因为没有设置图像的宽与高，所以无法按比例绘制图像全貌。

范例 16 文件 Ex3_4_1.java 的功能是解释 drawImage(Image img, int x, int y, this) 的绘制功能。

本例所用的图像文件是 Sunset.jpg，如图 3-1 所示。

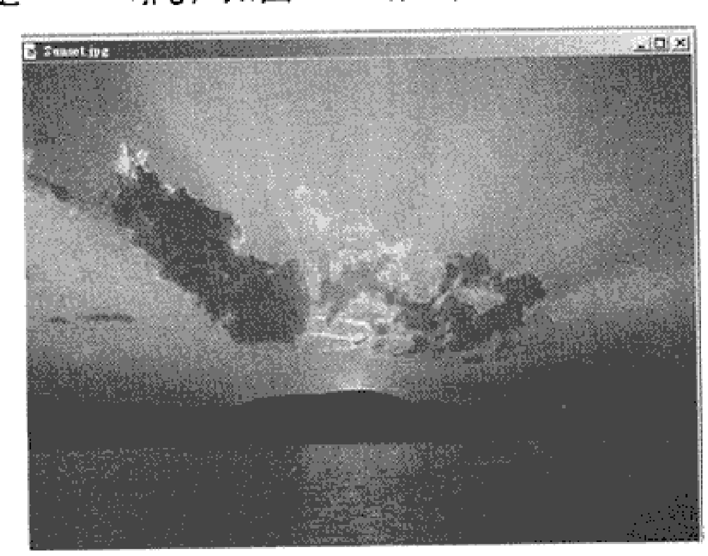


图 3-1

```
import java.awt.*;
import java.awt.event.*;
import java.awt.Graphics;

public class Ex3_4_1 extends Frame implements Runnable {
    Image image;

    public static void main(String args[]) {
        Ex3_4_1 workStart=new Ex3_4_1();
    }
```

```
09      public Ex3_4_1() {
10          super("Ex3_4_1");
11          setSize(350, 350);

12          Toolkit tk = Toolkit.getDefaultToolkit();
13          image = tk.getImage("Sunset.jpg");

14          setVisible(true);
15          new Thread(this).start();
16      }

17      public void run() {
18         repaint();
19      }

20      public void paint(Graphics g) {
21         g.drawImage(image, 0, 0, this);
22      }
23  }
```

行 12～13　根据 3-3 节的内容，读取图像文件 Sunset.jpg。
行 21　绘制图片，因没有设置图像的宽与高，所以无法按比例绘制图像全貌。

运行结果

运行结果如图 3-2 所示。

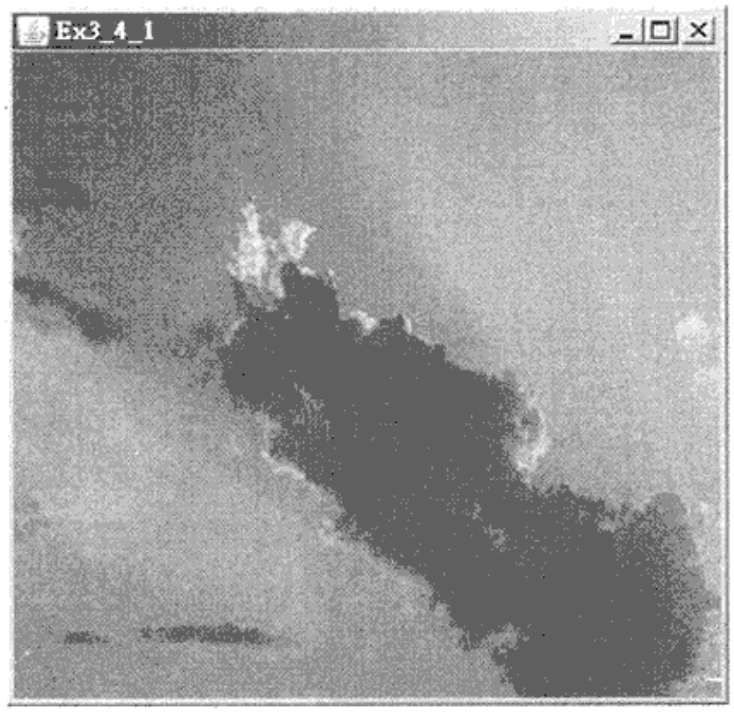


图 3-2

设计范例 16.1，使用实例方法 drawImage(Image img, int x, int y, int width, int height, this) 绘制图像，因为设置了图像的宽与高，所以可按比例绘制图像全貌。

范例 16.1　文件 Ex3_4_2.java 的功能是解释 drawImage(Image img, int x, int y, int width, int height, this) 的绘制功能。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex3_4_2 extends Frame implements Runnable {
05      Image image;

06      public static void main(String args[]) {
07         Ex3_4_2 workStart=new Ex3_4_2();
08      }

09      public Ex3_4_2() {
10            super("Ex3_4_2");
11            setSize(350, 350);

12            Toolkit tk = Toolkit.getDefaultToolkit();
13            image = tk.getImage("Sunset.jpg");

14            setVisible(true);
15            new Thread(this).start();
16      }

17      public void run() {
18         repaint();
19      }

20      public void paint(Graphics g) {
21         g.drawImage(image, 0, 0, 200, 200, this);
22      }
23 }
```

行 21　因为设置了图像的宽与高，所以可按比例绘制图像全貌。

运行结果

运行结果如图 3–3 所示。

图 3-3

3-5 习题

1. 目前 Java 可使用的图像文件格式有哪些？
2. GIF 格式的特点有哪些？
3. PNG 格式的特点有哪些？
4. JPEG（JPG）格式的特点有哪些？
5. 图像的显示环境有哪两种？
6. 如何利用 Toolkit 类读取图像文件？
7. 方法 drawImage(Image img, int x, int y, this) 在绘图上有何特点？
8. 方法 drawImage(Image img, int x, int y, int width, int height, this)在绘图上有何特点？

Chapter 04 基础动画

4-1 简介

在前面各章中，我们探讨的是静态图片，本章将介绍动态图片的设计，以便为在线游戏设计做准备，包括在框架中使用绘制的图案或图片；运行单幅动画或多幅动画。

4-2 动态图案

要显示一幅动态的图案，在程序设计上要克服以下两个问题：

(1) 不断地改变图案坐标。
(2) 当在新位置显示图案时，要清除旧位置上的图案。

1 改变图案坐标：置坐标变化量(dx，dy)，创建图案位置坐标(x+dx，y+dy)，这样图案将随新坐标的改变而移动。

2 清除旧位置上的图案：当图案移往新坐标时，必须清除旧位置上的图案，否则新旧图案重叠，图形会显得混乱而无法辨识。repaint()方法有更新图案与清除旧图案的功能。

设计范例 17，绘制红色球形，由框架的左端移向右端，使用(x+dx，y+dy) 改变球形的位置坐标，使用 repaint()方法更新图案与清除旧图案。

范例 17 文件 Ex4_2_1.java 的功能是解释如何移动图案。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex4_2_1 extends Frame implements Runnable {
05     int x=0, y=160;
06     int dx=5, dy=0;

07     public static void main(String args[]) {
08        Ex4_2_1 workStart=new Ex4_2_1();
09     }

10     public Ex4_2_1() {
11        super("Ex4_2_1");
12        setSize(350, 350);
13        setVisible(true);
```

```
14        new Thread(this).start();
15    }

16    public void run() {
17        while(true) {
18            x = x + dx;
19            y = y + dy;
20            repaint();

21            try{Thread.sleep(250);}
22            catch(InterruptedException e) {;}
23        }
24    }

25    public void paint(Graphics g) {
26        g.setColor(Color.red);
27        g.fillOval(x, y, 50, 50);
28    }
29 }
```

行 05	声明坐标(x，y)，并初始化。
行 06	声明坐标变化量(dx，dy)，并初始化。
行 17～23	while 循环，每循环一次，图案移动到一个新位置。
行 18～19	新坐标位置。
行 20	调用行 25～28，更新图案与清除旧图案。
行 21	设置每循环一次时的停顿时间。
行 27	在新位置上绘制红色球形。

运行结果

运行结果如图 4–1 所示。

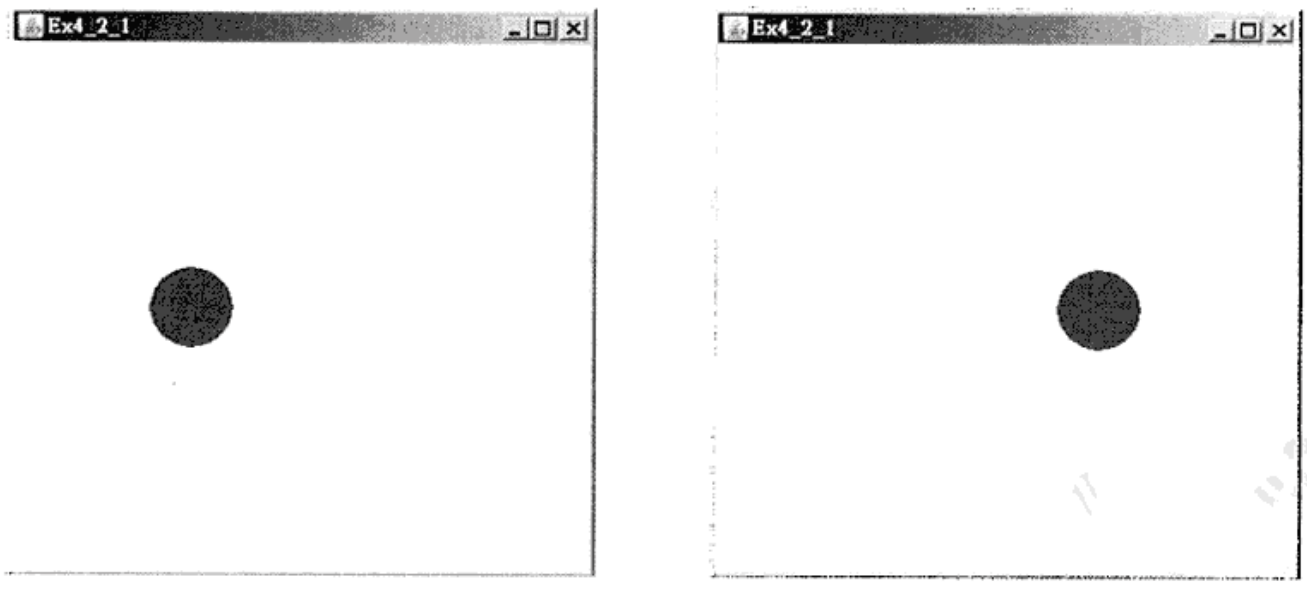


图 4-1

讨论事项

在本例中，当球形移出框架边缘时将一去不返，我们可增设其折返功能（如范例 18）。

java.awt.Component 类提供了 getWidth()和 getHeight()方法，它们可读取框架的宽与高。设计范例 18，当图案移动到框架边缘时会立即回弹。

范例 18 文件 Ex4_2_2.java 的功能是解释图案回弹的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex4_2_2 extends Frame implements Runnable {
05     int x=0, y=160;
06     int dx=5, dy=0;

07     public static void main(String args[]) {
08        Ex4_2_2 workStart=new Ex4_2_2();
09     }

10     public Ex4_2_2() {
11        super("Ex4_2_2");
12        setSize(350, 350);
13        setVisible(true);
14        new Thread(this).start();
15     }

16     public void run() {
17        while(true) {
18           x = x + dx;
19           y = y + dy;
20           repaint();

21           if(x<=0) dx = 5;
22           else if((x + 50) >= getWidth()) dx = -5;

23           if(y<=0) dy = 5;
24           else if((y + 50) >= getHeight()) dy = -5;

25           try{Thread.sleep(250);}
26           catch(InterruptedException e) {;}
27        }
28     }

29     public void paint(Graphics g) {
30        g.setColor(Color.red);
31        g.fillOval(x, y, 50, 50);
32     }
33 }
```

行 21　当图案移动到框架左边线时，设置向右移动。
行 22　当图案移动到框架右边线时，设置向左移动。通过 getWidth()来读取右边线位置。
行 23　当图案移动到框架上边线时，设置向下移动。
行 24　当图案移动到框架下边线时，设置向上移动。通过 getHeight()来读取下边线位置。

运行结果

运行结果如图 4-2 所示。

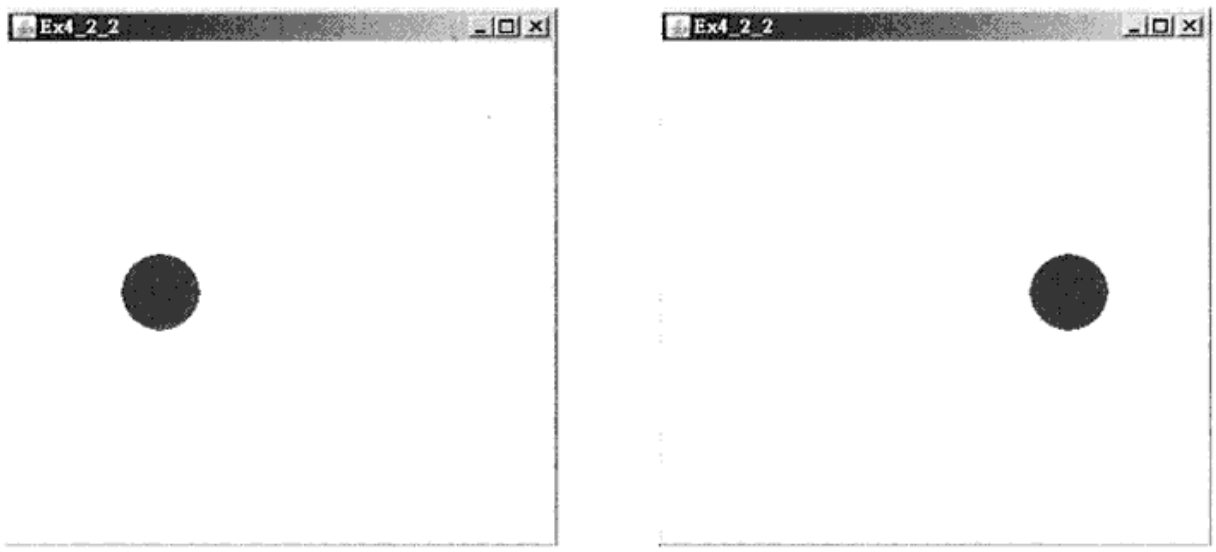


图 4-2

4-3 动态图像

参考 3–3 节和 3–4 节，通过 Toolkit 类的类方法 getDefaultToolkit()创建新对象，再以其实例方法 getImage()读取图像。通过 Graphics 类进行图像图片绘制。设计范例 18.1，读取单幅图像文件并实现动画移动效果。

范例 18.1 文件 Ex4_3_1.java 的功能是解释单幅动画的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex4_3_1 extends Frame implements Runnable {
05     int x=0, y=100;
06     int dx=5, dy=5;
07     Image img;

08     public static void main(String args[]) {
09       Ex4_3_1 workStart=new Ex4_3_1();
10     }

11     public Ex4_3_1() {
12       super("Ex4_3_1");
13       setSize(350, 350);

14       Toolkit tk = Toolkit.getDefaultToolkit();
15       img = tk.getImage("fly.gif");

16       setVisible(true);
17       new Thread(this).start();
18     }

19     public void run() {
20       while(true) {
21         x = x + dx;
22         y = y + dy;
23         repaint();

24         if(x<=0) dx = 5;
25         else if((x + 50) >= getWidth()) dx = -5;
```

```
26            if(y<=0) dy = 5;
27            else if((y + 50) >= getHeight()) dy = -5;

28            try{Thread.sleep(250);}
29            catch(InterruptedException e) {;}
30         }
31     }

32     public void paint(Graphics g) {
33         g.drawImage(img, x, y, this);
34     }
35 }
```

行 14～15	根据 3–3 节的内容，读取图像文件 fly.gif。
行 33	绘制图片。

运行结果

运行结果如图 4–3 所示。

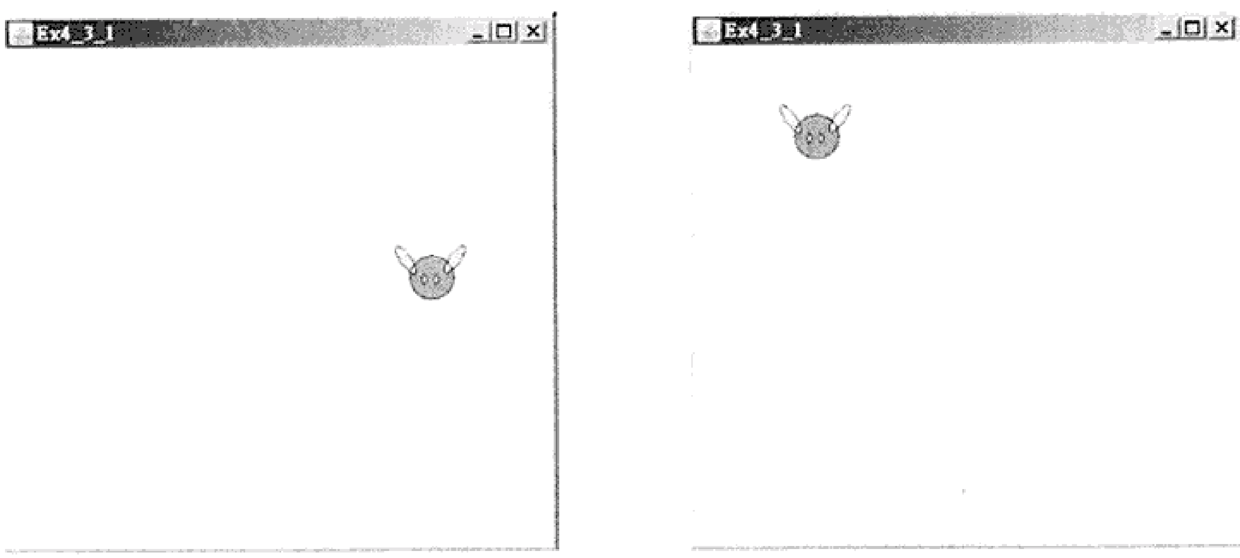


图 4-3

单幅动画不够灵活生动，要使动画生动就必须运行多幅动画。其设计方法如下：

（1）设置图像框架坐标，随用户的需要变化该坐标。

（2）设计多幅连续动作的图像文件，轮流置入移动框架坐标。

范例 19 文件 Ex4_3_2.java 的功能是解释多幅动画的应用。

```
import java.awt.*;
import java.awt.event.*;

public class Ex4_3_2 extends Frame implements Runnable {
    int num=0, flag;
    int x=0, y=100, dx=5, dy=5;
    Image img0, img1, img2;

    public static void main(String args[]) {
        Ex4_3_2 workStart=new Ex4_3_2();
    }
```

```
10    public Ex4_3_2() {
11        super("Ex4_3_2");
12        setSize(350, 350);

13        Toolkit tk = Toolkit.getDefaultToolkit();
14        img0 = tk.getImage("fly0.gif");
15        img1 = tk.getImage("fly1.gif");
16        img2 = tk.getImage("fly2.gif");

17        setVisible(true);
18        new Thread(this).start();
19    }

20    public void run() {
21       while(true) {
22           x = x + dx;
23           y = y + dy;
24           flag = num % 3;
25           repaint();
26           num = num + 1;

27           if(x <= 0) dx = 5;
28           else if((x+60) >= 350) dx = -5;

29           if(y<=0) dy = 5;
30           else if((y + 50) >= getHeight()) dy = -5;

31           try{Thread.sleep(250);}
32           catch(InterruptedException e) {;}
33       }
34    }

35    public void paint(Graphics g) {
36        if(flag == 0)
37            g.drawImage(img0, x, y, this);
38        else if(flag == 1)
39            g.drawImage(img1, x, y, this);
40        else if(flag == 2)
41            g.drawImage(img2, x, y, this);
42    }
43 }
```

行 04	声明变量 flag 与 num，用于选择图像文件。
行 06	声明图像变量 img0、img1、img2。
行 13～16	读取 3 幅图像。
行 24	"flag = num % 3;" 循环使用 3 幅图像，将 num 除以 3 取其余数，若余数为 0，则选用 img0；若余数为 1，则选用 img1；若余数为 2，则选用 img2。
行 26	每次循环将 num 加 1，配合行 24 选择一幅图像。
行 36～41	如行 24 所述，依序循环选择各幅图像来显示。

运行结果

运行结果如图 4–4 所示。

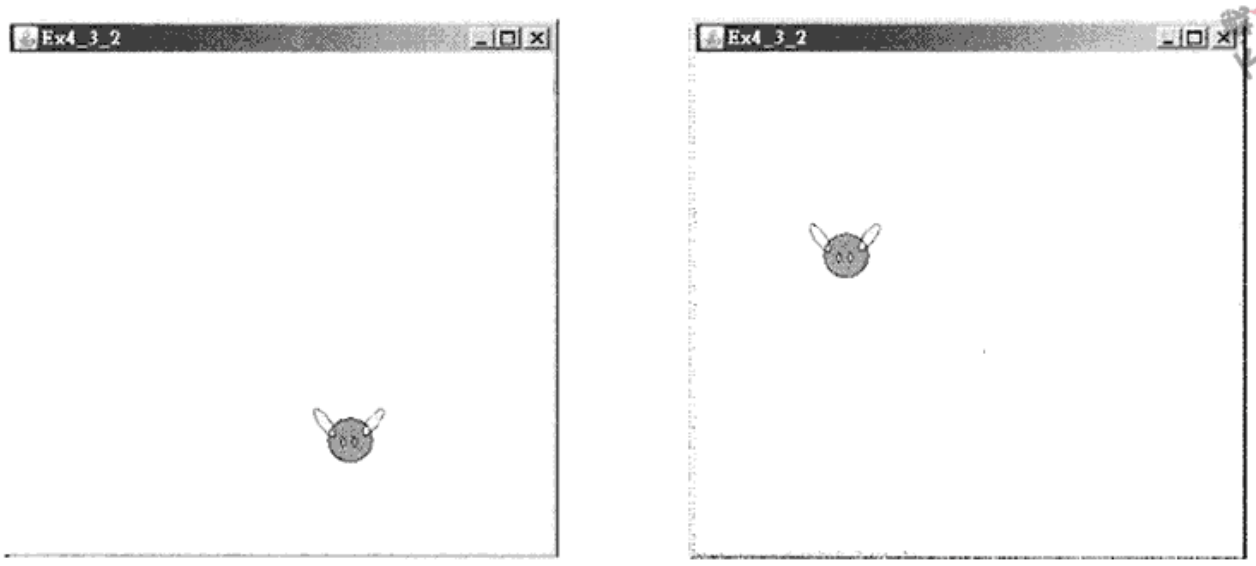


图 4-4

4-4 数组与动画

在处理生动的动画时，都需读取/绘制多幅图像文件，为使程序简洁且具有高可读性，我们应选择使用数组。其详细的使用方法请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》。设计范例 20，使用数组读取 10 幅图像文件，并以静态方式依序绘出这 10 个数字（图像文件的内容）。

范例 20 文件 Ex4_4_1.java 的功能是解释数组与静态动画的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex4_4_1 extends Frame implements Runnable {
04    int num=0, flag, i;
05    Image[] img;

06    public static void main(String args[]) {
07       Ex4_4_1 workStart=new Ex4_4_1();
08    }

09    public Ex4_4_10 {
10          super("Ex4_4_1");
11          setSize(350, 350);

12          Toolkit tk = Toolkit.getDefaultToolkit();
13          img = new Image[10];
14          for(i=0; i<10; i++) {
15              img[i] = tk.getImage("char" + i + ".png");
16          }

17          setVisible(true);
18          new Thread(this).start();
19    }

20    public void run() {
21       while(true) {
22          flag = num % 10;
23          repaint();
24          num = num + 1;

25          try{Thread.sleep(250);}
```

```
26              catch(InterruptedException e) {;}
27          }
28      }

29      public void paint(Graphics g) {
30          g.drawImage(img[flag], 50, 50, this);
31      }
32 }
```

行 05	声明图像数组。
行 13	因为数组属于引用类型（请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》），所以必须为其新对象设置长度。本例中新对象的长度为 10。
行 14～16	读取图像文件 char0.png～char9.png，分别存放在数组元素 img[0]～img[9]中。
行 22	设置 flag 值。
行 30	当 flag 为 0 时显示 img[0]，当 flag 为 1 时显示 img[1]，依此类推。

运行结果

运行结果如图 4-5 所示。

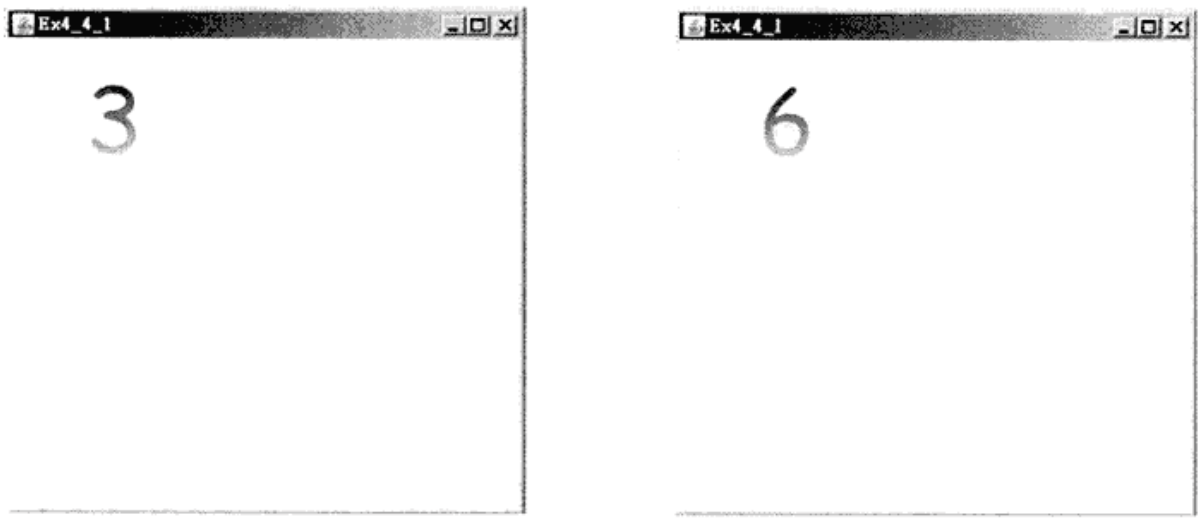


图 4-5

设计范例 21，修改范例 19，以数组方式读取图像文件。

范例 21 参考范例 19，文件 Ex4_4_2.java 的功能是解释以数组方式设计多幅动画的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex4_4_2 extends Frame implements Runnable {
04      int num=0, flag, i;
05      int x=0, y=100, dx=5, dy=5;
06      Image[] img;

07      public static void main(String args[]) {
08          Ex4_4_2 workStart=new Ex4_4_2();
09      }

10      public Ex4_4_2() {
11          super("Ex4_4_2");
12          setSize(350, 350);
```

```
13          Toolkit tk = Toolkit.getDefaultToolkit();
14          img = new Image[3];
15          for(i=0; i<3; i++) {
16              img[i] = tk.getImage("fly" + i + ".gif");
17          }

18          setVisible(true);
19          new Thread(this).start();
20      }

21      public void run() {
22        while(true) {
23            x = x + dx;
24            y = y + dy;
25            flag = num % 3;
26            repaint();
27            num = num + 1;

28            if(x <= 0) dx = 5;
29            else if((x+60) >= 350) dx = -5;

30            if(y<=0) dy = 5;
31            else if((y + 50) >= getHeight()) dy = -5;

32            try{Thread.sleep(250);}
33            catch(InterruptedException e) {;}
34        }
35      }

36      public void paint(Graphics g) {
37          g.drawImage(img[flag], x, y, this);
38      }
39  }
```

参考范例 20 的解说。

运行结果

运行结果如图 4–6 所示。

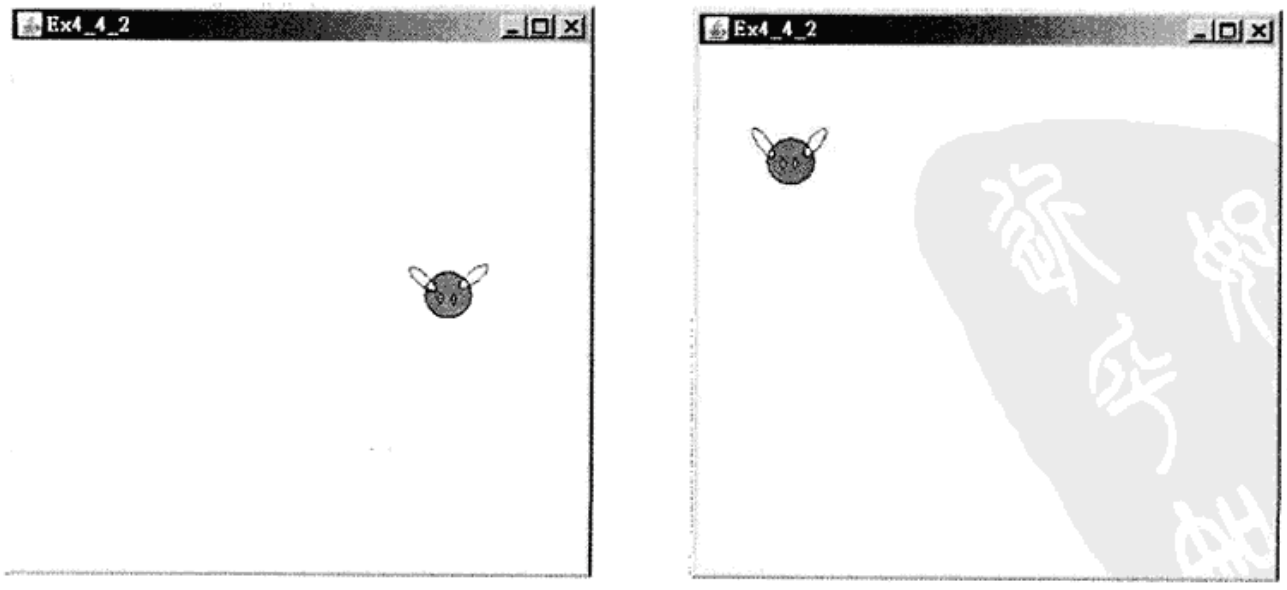


图 4-6

4-5 习题

1. 设计动态图案，在程序设计上要克服哪些问题？
2. 如何取得框架的宽与高？
3. 如何读取图像？
4. 设计多幅动画时有哪些重点？

PART

02

事件处理

在动画游戏设计上，我们应考虑以交互功能来提升精彩度，譬如使用鼠标单击选择位置，使用键盘键改变移动方向。

底层事件（Low Level Events）的启动是将底层交互输入用于图形界面（如同鼠标、键盘键在框架图形界面内的作用），Java的API定义了一组事件类（Event Classes），用于生成各种事件对象（Event Objects）。当事件发生时，这些对象可在事件源（Event Source）与事件侦听器（Event Listener）之间传递。Java支持的底层事件类有：ComponentEvent、ContainerEvent、FocusEvent、KeyEvent、MouseEvent、WindowEvent。所有底层事件类都是由AWTEvent类派生而来的。

当默认事件发生时，无论程序在何处，进行何种动作，它都将暂停，由系统将适当的资源（包括CPU与时间）分配给它，以优先支持该事件的运行。

Chapter 05　底层事件

在前面各章中，我们探讨了如何绘图？如何读取图片？但仍缺少精彩的交互性，譬如使用键盘键改变移动方向，这些都是令人兴奋的交互操作。虽然读者可能是资深玩家，但如果能彻底了解其中程序设计的奥妙，那将更有意义。

Chapter 06　鼠标事件应用

常用的鼠标事件行为有按键（PRESSED）、单击（CLICKED）和释放（RELEASED）。每次发生鼠标事件时，我们可以立即读取该事件的源（Source）、标识码（ID）、发生时间（When）、x轴坐标和y轴坐标。由于上述信息是伴随事件的发生而产生的，所以我们可利用这些信息引导后续操作的执行，如移动（Moved）、拖动（Dragged）、选择（Selected）、随动（Followed）、棋盘（Chessboard）等。回顾1—3节，线程绘图流程是以Thread驱动run()方法，再以repaint()调用paint()方法，这样可获得线程并行的效果。

Chapter 07　键盘事件应用

常用的键盘事件有按键（PRESSED）和释放（RELEASED）。每次发生键盘事件时，我们可以立即读取该事件的源（Source）、标识码（ID）、发生时间（When）与按键的ASCII码。由于上述信息是伴随事件的发生而产成的，所以我们可利用这些信息引导后续操作的执行，如键盘数据、图案移动方向控制等。回顾1—3节，线程绘图流程是以Thread驱动run()方法，再以repaint()调用paint()方法，这样可获得线程并行的效果。

Chapter 08 消除图像闪烁

细心的读者可能已经注意到，上述各章的图像动画都有闪烁漂浮的情况，这是因为新的图像随时替换旧的图像，如此才能使图像在窗口中显示出位置的变化或姿态的变化。如果图像替换得太快，其速度超过了计算机系统的负荷，有些部分先到位，有些部分后到位，则图像将会显得闪烁而不稳定，这也是计算机游戏设计不希望看到的。本章将介绍如何消除这样的闪烁。

Chapter 09 音效处理

如果动画与电玩没有音效的陪衬，将使画面失色许多，Sun公司在设计Applet时已支持音效的使用，但在当时并不适用于窗口框架。本书在第1章开宗明义地提到，动画游戏有文字、有图像、有动画，这些都需要一个环境来显示，常用的显示环境是窗口框架（Frame）与浏览器（Browser），前者可用于单机显示或多机网络对阵；后者可用于单机网络显示。为了使窗口框架具有适当的音响效果，Sun公司特别将音效由Applet扩展到窗口框架系列。

Chapter 05 底层事件

5-1 简介

在前面各章中，我们探讨了如何绘图？如何读取图片？但仍缺少精彩的交互性，譬如使用键盘改变移动方向，这些都是令人兴奋的交互操作。虽然读者可能是资深玩家，但如果能彻底了解其中程序设计的奥妙，那将更有意义。

5-2 Java 事件架构

当默认事件发生时，无论程序在何处，进行何种动作，它都将暂停，由系统将适当的资源（包括 CPU 与时间）分配给它，以优先支持该事件的运行。

底层事件的启动是将底层交互输入用于图形界面（如同鼠标、键盘在框架图形界面内的作用），Java 的 API 定义了一组事件类，用于生成各种事件对象。当事件发生时，这些对象可在事件源与事件侦听器之间传递。Java 支持的底层事件类有：

- ComponentEvent
- ContainerEvent
- FocusEvent
- KeyEvent
- MouseEvent
- WindowEvent

所有底层事件类都是由 AWTEvent 类派生而来的，同时将各事件类共享的特性封装聚集起来，它们的关系如图 5–1 所示。

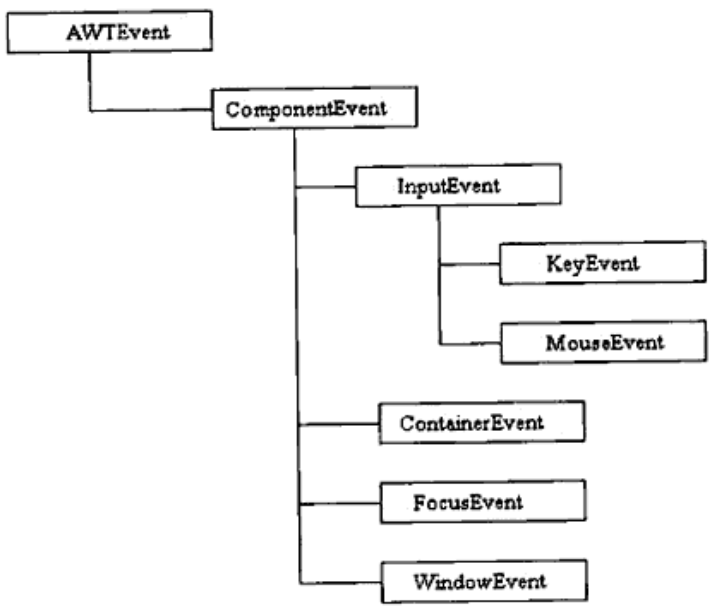


图 5-1

5-3 AWTEvent 类

java.awt.AWTEvent 继承自 EventObject → Object，此类是所有底层事件类的基础类。

1 构造函数

public AWTEvent(Object source, int id);

根据事件源对象与标识码创建 AWT 事件。其中，source 是事件源对象；id 是标识码。

2 类常量

public static final long **ACTION_EVENT_MASK;**
public static final long **CONPONENT_EVENT_MASK;** （如范例 22）
public static final long **CONTAINER_EVENT_MASK;**
public static final long **FOCUS_EVENT_MASK;** （如范例 28）
public static final long **KEY_EVENT_MASK;** （如范例 24）
public static final long **MOUSE_EVENT_MASK;** （如范例 26）
public static final long **MOUSE_MOTION_EVENT_MASK;**
public static final long **WINDOW_EVENT_MASK;** （如范例 30）

这些常量是 AWTEvent 类支持的事件对象的类型。

3 类方法

public void **enableEvents(AWTEvent constant);**

启动 AWTEvent 类所属的对象。

enableEvents(AWTEvent.COMPONENT_EVENT_MASK);

启动 Component 事件（如范例 22）。

enableEvents(AWTEvent.MOUSE_EVENT_MASK);

启动鼠标事件（如范例 26）。

enableEvents(AWTEvent.KEY_EVENT_MASK);

启动键盘事件（如范例 24）。

enableEvents(AWTEvent.CONTAINER_EVENT_MASK);

启动容器事件。

enableEvents(AWTEvent.FOCUS_EVENT_MASK);

启动焦点事件（如范例 28）。

enableEvents(AWTEvent.WINDOW_EVENT_MASK);

启动窗口事件（如范例 30）。

4 实例方法

public int getID();

配合构造函数生成的新对象，返回 AWT 事件的标识码。

5-4 ComponentEvent 类

java.awt.event.ComponentEvent 继承自 AWTEvent → EventObject → Object，此类将组件层次的事件封装起来，如移动（Move）、改变大小（Resize）、隐藏（Hid）或显示（Show）等。

1 构造函数

public ComponentEvent(Component source, int id);

其中，参数 source 是一个与指定组件相关的事件对象；id 是该事件对象的标识码（如范例 22）。

2 类常量

public static final int **COMPONENT_MOVED;** （如范例 22）
public static final int **COMPONENT_RESIZED;** （如范例 23）
public static final int **COMPONENT_HIDDEN;**
public static final int **COMPONENT_SHOWN;**

这些常量代表显示组件事件的类型。

3 类方法

process**ComponentEvent(ComponentEvent e);**

运行 Component 事件（如范例 22）。

4 实例方法

public Component **getComponent();**

配合构造函数生成的新对象，返回组件事件。

public Object **getSource();**
public int **getID();**

配合构造函数生成的事件对象，返回其中参数的内容（如范例 22）。

设计范例 22，解释如何在框架中使用方法 enableEvents（AWTEvent.COMPONENT_EVENT_MASK）启动 Component 事件；如何使用方法 public void processComponentEvent(ComponentEvent e) 运行 Component 事件，同时读取其构造函数各参数的内容。本例中，Component 事件是移动框架事件。

范例 22 文件 Ex5_4_1.java 的功能是解释移动框架的 Component 事件。

```
01 import java.awt.*;
02 import java.awt.event.*;
```

```
03 public class Ex5_4_1 extends Frame {
04      public Ex5_4_1() {
05          super("Ex5_4_1");
06          setSize(350, 350);
07          setVisible(true);
08          enableEvents(AWTEvent.COMPONENT_EVENT_MASK);
09      }

10      public void processComponentEvent(ComponentEvent e) {
11          if(e.getID() == ComponentEvent.COMPONENT_MOVED) {
12              System.out.println("getSource() : " + e.getSource());
13              System.out.println("getID() : " + e.getID());
14          }
15      }

16      public static void main(String args[]) {
17        Ex5_4_1 workStart=new Ex5_4_1();
18      }
19 }
```

行 08　启动 Component 事件（参考 5–3 节的类方法）。

行 10　运行 Component 事件（参考 5–4 节的类方法）。

行 11　检查是不是移动框架的 Component 事件（参考 5–4 节的 getID()方法与类常量 COMPONENT_MOVED）。

行 12　读取事件源对象 source。

行 13　读取标识码 id。

运行结果

移动框架时将产生下列结果，如图 5–2 所示。

```
getSource() :
Ex5_4_1[frame0,0,0,350x350,layout=java.awt.BorderLayout,title=Ex5_4_1,resizable,normal]
getID() : 100
getSource() :
Ex5_4_1[frame0,94,28,350x350,layout=java.awt.BorderLayout,title=Ex5_4_1,resizable,normal]
getID() : 100
```

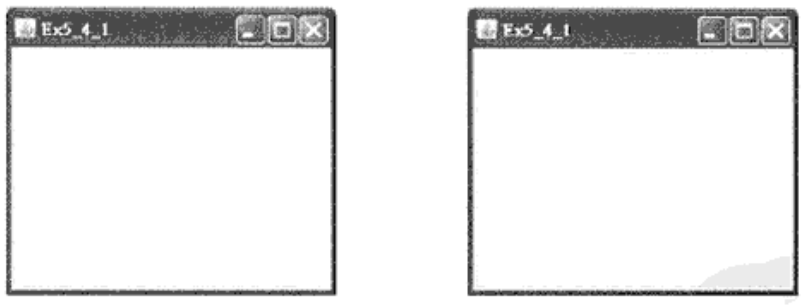


图 5-2

讨论事项

从运行结果中可观察到框架坐标的改变，本例从(0，0)移至(94，28)。

设计范例 23，解释当改变框架大小的 Component 事件发生时，如何读取其构造函数各参数的内容。

范例 23　文件 Ex5_4_2.java 的功能是解释改变框架大小的 Component 事件。

```
01 import java.awt.*;
```

```
02 import java.awt.event.*;

03 public class Ex5_4_2 extends Frame {
04      public Ex5_4_2() {
05           super("Ex5_4_2");
06           setSize(350, 350);
07           setVisible(true);
08           enableEvents(AWTEvent.COMPONENT_EVENT_MASK);
09      }

10      public void processComponentEvent(ComponentEvent e) {
11           if(e.getID() == ComponentEvent.COMPONENT_RESIZED) {
12                System.out.println("getSource() : " + e.getSource());
13                System.out.println("getID() : " + e.getID());

14           }
15      }

16      public static void main(String args[]) {
17        Ex5_4_2 workStart=new Ex5_4_2();
18      }
19 }
```

行 11　检查是不是改变框架大小的 Component 事件（参考 5—4 节的 getID()方法与类常量 COMPONENT_RSIZED）。

运行结果

改变框架大小时将产生下列结果，如图 5—3 所示。

```
getSource() :
Ex5_4_2[frame0,0,0,350x350,layout=java.awt.BorderLayout,title=Ex5_4_2,resizable,normal]
getID() : 101
getSource() :
Ex5_4_2[frame0,67,15,247x240,layout=java.awt.BorderLayout,title=Ex5_4_2,resizable,normal]
getID() : 101
```

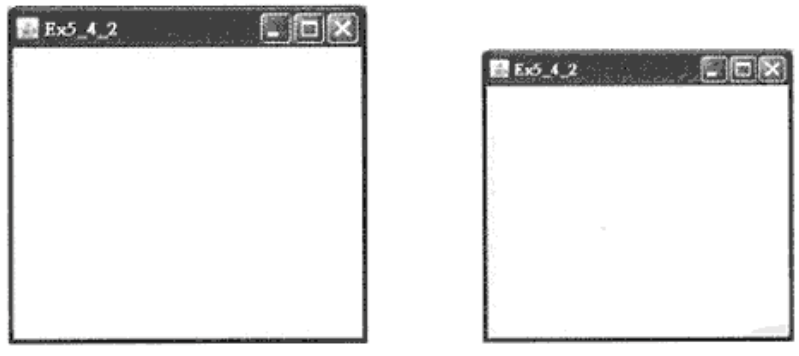


图 5-3

讨论事项

从运行结果中可观察到框架大小的改变，本例从 350x350 缩小为 247x240。

5-5 KeyEvent 类

java.awt.event.KeyEvent 继承自 InputEvent → ComponentEvent → AWTEvent → EventObject → Object，此类将与按键有关的事件封装起来，击打键盘是按键事件的来源（KeyEvent Source），它还

实现了 KeyListener 接口，即一个 KeyEvent 会被送到 KeyListener 中进行处理。

1 构造函数

```
public KeyEvent(Component source, int id, long when, int keyCode);
```

其中，参数 source 是与按键事件相关的对象；id 是事件标识码；when 是按键的时间；keyCode 是按键的 ASCII 码（如范例 24）。

2 类常量

```
public static final int KEY_TYPED, KEY_PRESSED, KEY_RELEASED;
public static final int VK_LEFT, VK_RIGHT, VK_UP, VK_DOWN;
public static final int VK_0, VK_1, VK_2, VK_3, VK_4, VK_5, VK_6,
                         VK_7, VK_8, VK_9;
public static final int VK_A, VK_B, VK_C, VK_D, VK_E, VK_F, VK_G,
                         VK_H, VK_I, VK_J, VK_K, VK_L, VK_M, VK_N,
                         VK_O, VK_P, VK_Q, VK_R, VK_S, VK_T, VK_U,
                         VK_V, VK_W, VK_X, VK_Y, VK_Z;
```

上述都是常用的键盘常量，因为数量众多，所以其他的请读者参考附录 A。

3 类方法

```
public void processKeyEvent(KeyEvent e);
```

运行键盘事件（如范例 24）。

4 实例方法

```
public Object getSource();
public int getID();
public int getWhen();
public int getKeyCode();
```

配合构造函数生成的事件对象，返回其中参数的内容（如范例 24）。

设计范例 24，解释如何在框架中使用方法 enableEvents（AWTEvent.KEY_EVENT_MASK）启动键盘事件；如何使用方法 public void processKeyEvent(KeyEvent e) 运行键盘事件，同时读取按键事件构造函数各参数的内容。

范例 24 文件 Ex5_5_1.java 的功能是解释以键“A”作为 KEY_PRESSED 按键事件时，如何读取按键事件构造函数参数的内容？

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex5_5_1 extends Frame {
04     public Ex5_5_1() {
05         super("Ex5_5_1");
06         setSize(350, 350);
07         setVisible(true);
08         enableEvents(AWTEvent.KEY_EVENT_MASK);
09     }

10     public void processKeyEvent(KeyEvent e) {
11         if(e.getID() == KeyEvent.KEY_PRESSED) {
12             System.out.println("getSource() : " + e.getSource());
13             System.out.println("getID() : " + e.getID());
14             System.out.println("getWhen() : " + e.getWhen());
15             System.out.println("getKeyCode() : " + e.getKeyCode());
```

```
16          }
17      }

18      public static void main(String args[]) {
19          Ex5_5_1 workStart=new Ex5_5_1();
20      }
21  }
```

行 08　启动键盘事件（参考 5-3 节的类方法）。
行 10　运行键盘事件（参考 5-5 节的类方法）。
行 11　检查是否是按键事件（参考 5-5 节的 getID()方法与类常量 KEY_PRESSED）。
行 12　读取按键事件的 source。
行 13　读取事件标识码 id。
行 14　读取按键的时间 when。
行 15　读取按键的 ASCII 码。

运行结果

本例是在打开框架时运行的，而不是在打开 DOS 时运行的。运行结果如图 5-4 所示。

```
getSource() :
Ex5_5_1[frame0,0,0,350x350,layout=java.awt.BorderLayout,title=Ex5_5_1,resizable,normal]
getID() : 401
getWhen() : 1194100216937
getKeyCode() : 65
```

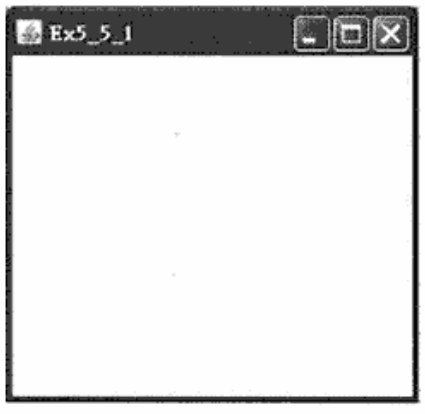


图 5-4

讨论事项

（1）标识码为 401。
（2）有关事件发生时间的部分请参考本系列丛书第一册《Java 典型应用彻查 1000 例——Java 入门》。
（3）'A'的 ASCII 码为 65，读者可试试其他按键，能得到不同的结果。

设计范例 25，当指定的按键事件发生时，测试其类常量与事件的关系，本例以键'A'的 VK_A 为例。

范例 25　文件 Ex5_5_2.java 的功能是解释 KeyEvent 类的类常量 VK_A 与事件的关系。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 public class Ex5_5_2 extends Frame {
```

```
04    public Ex5_5_2() {
05        super("Ex5_5_2");
06        setSize(350, 350);
07        setVisible(true);
08        enableEvents(AWTEvent.KEY_EVENT_MASK);
09    }

10    public void processKeyEvent(KeyEvent e) {
11        if(e.getID() == KeyEvent.KEY_PRESSED) {
12          if(e.getKeyCode() == KeyEvent.VK_A)
13            System.out.println("This is VK_A event");
14          else
15            System.out.println("This is not VK_A event");
16        }
17    }

18    public static void main(String args[]) {
19      Ex5_5_2 workStart=new Ex5_5_2();
20    }
21 }
```

行 12～16　测试键'A'与 VK_A 的关系，当击打键'A'时会显示“This is VK_A event”，否则会显示“This is not VK_A event”。

运行结果

运行结果如图 5-5 所示。

```
This is VK_A event
This is not VK_A event
```

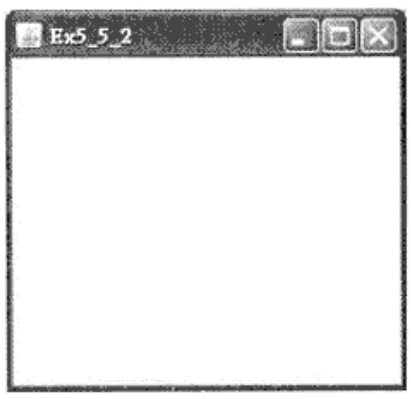


图 5-5

5-6 MouseEvent 类

java.awt.event.MouseEvent 继承自 InputEvent→ComponentEvent→AWTEvent → EventObject → Object，此类将与鼠标有关的事件封装起来，包括单击、移动等。鼠标是鼠标事件源（MouseEvent Source），MouseEvent 还实现了 MouseListener 接口，即一个 MouseEvent 会被送到 MouseListener 中进行处理。

1　构造函数

public **MouseEvent(Component source, int id, long when, int x, int y, int clickCount);**

其中，参数 source 是与鼠标事件相关的对象；id 是事件标识码；when 是鼠标事件的时间；（x，y）是鼠标指针的位置坐标；clickCount 是鼠标的单击次数（如范例 26）。

2 类常量

```
public static final int MOUSE_CLICKED;
public static final int MOUSE_ENTERED;   （如范例 27）
public static final int MOUSE_EXITED;    （如范例 27）
public static final int MOUSE_PRESSED;   （如范例 26）
public static final int MOUSE_RELEASED;
```

因为是类常量，所以不需配合构造函数生成的事件对象，可直接使用，用于标示鼠标事件的类型。

3 类方法

```
public void processMouseEvent(MouseEvent e);
```

运行鼠标事件（如范例 26）。

4 实例方法

```
public Object getSource();
public int getID();
public int getWhen();
public int getX();
public int getY();
public int getClickCount();
```

配合构造函数生成的事件对象，返回其中参数的内容（如范例 26）。

设计范例 26，解释如何在框架中使用方法 enableEvents（AWTEvent.MOUSE_EVENT_MASK）启动鼠标事件；如何使用方法 public void processMouseEvent(MouseEvent e）运行鼠标事件，同时读取鼠标事件构造函数各参数的内容。

范例 26 文件 Ex5_6_1.java 的功能是解释 MouseEvent 类的类常量 MOUSE_PRESSED 与如何读取鼠标事件构造函数参数的内容？

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex5_6_1 extends Frame {
04     public Ex5_6_1() {
05         super("Ex5_6_1");
06         setSize(350, 350);
07         setVisible(true);
08         enableEvents(AWTEvent.MOUSE_EVENT_MASK);
09     }

10     public void processMouseEvent(MouseEvent e) {
11         if(e.getID() == MouseEvent.MOUSE_PRESSED) {
12             System.out.println("getSource() : " + e.getSource());
13             System.out.println("getID() : " + e.getID());
14             System.out.println("getWhen() : " + e.getWhen());
15             System.out.println("getX() : " + e.getX());
16             System.out.println("getY() : " + e.getY());
17             System.out.println("getClickCount() : " + e.getClickCount());
18         }
19     }
```

```
20      public static void main(String args[]) {
21          Ex5_6_1 workStart=new Ex5_6_1();
22      }
23 }
```

行 08　启动鼠标事件（参考 5-3 节的类方法）。
行 10　运行鼠标事件（参考 5-6 节的类方法）。
行 11　检查是不是鼠标事件（参考 5-6 节的 getID()方法与类常量 MOUSE_PRESSED）。
行 12　读取鼠标事件的 source。
行 13　读取事件标识码 id。
行 14　读取事件的时间 when。
行 15　读取鼠标指针的 x 轴坐标。
行 16　读取鼠标指针 y 轴坐标。
行 17　读取鼠标的单击次数。

运行结果

运行结果如图 5-6 所示。

```
getSource() :
Ex5_6_1[frame0,0,0,350x350,layout=java.awt.BorderLayout,title=Ex5_6_1,resizable,normal]
getID() : 501
getWhen() : 1194251609812
getX() : 239
getY() : 291
getClickCount() : 1
```

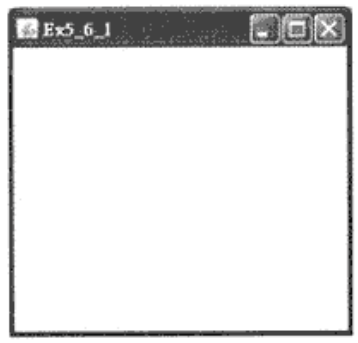


图 5-6

设计范例 27，使用类常量 MOUSE_ENTERED，当鼠标指针进入框架时，读取当时各参数的内容；使用类常量 MOUSE_EXITED，当鼠标指针离开框架时，读取当时各参数的内容。

范例 27 文件 Ex5_6_2.java 的功能是解释类常量 MOUSE_ENTERED 和 MOUSE_EXITED 的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex5_6_2 extends Frame {
04      public Ex5_6_2() {
05          super("Ex5_6_2");
06          setSize(350, 350);
07          setVisible(true);
08          enableEvents(AWTEvent.MOUSE_EVENT_MASK);
09      }
```

```
10      public void processMouseEvent(MouseEvent e) {
11          if(e.getID() == MouseEvent.MOUSE_ENTERED) {
12              System.out.println("ENTER DATA ::");
13              System.out.println("getSource() : " + e.getSource());
14              System.out.println("getID() : " + e.getID());
15              System.out.println("getWhen() : " + e.getWhen());
16              System.out.println("getX() : " + e.getX());
17              System.out.println("getY() : " + e.getY());
18              System.out.println("getClickCount() : " + e.getClickCount());
19          }

20          if(e.getID() == MouseEvent.MOUSE_EXITED) {
21              System.out.println("EXIT DATA ::");
22              System.out.println("getSource() : " + e.getSource());
23              System.out.println("getID() : " + e.getID());
24              System.out.println("getWhen() : " + e.getWhen());
25              System.out.println("getX() : " + e.getX());
26              System.out.println("getY() : " + e.getY());
27              System.out.println("getClickCount() : " + e.getClickCount());
28          }
29      }

30      public static void main(String args[]) {
31         Ex5_6_2 workStart=new Ex5_6_2();
32      }
33 }
```

行 11～19　使用类常量 MOUSE_ENTERED，当鼠标指针进入框架时，读取当时各参数的内容。

行 20～29　使用类常量 MOUSE_EXITED，当鼠标指针离开框架时，读取当时各参数的内容。

运行结果

运行结果如图 5-7 所示。

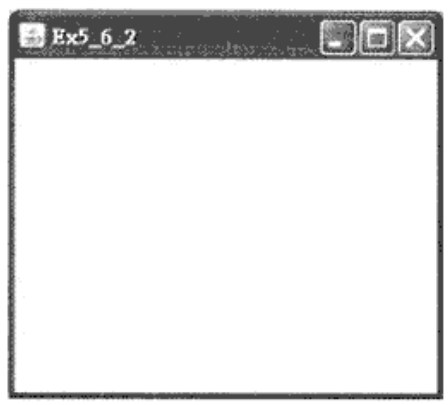


图 5-7

```
ENTER DATA ::
getSource() :
Ex5_6_2[frame0,0,0,350x350,layout=java.awt.BorderLayout,title=Ex5_6_2,resizable,normal]
getID() : 504
getWhen() : 1194256538687
getX() : 333
getY() : 282
getClickCount() : 0
EXIT DATA ::
getSource() :
Ex5_6_2[frame0,0,0,350x350,layout=java.awt.BorderLayout,title=Ex5_6_2,resizable,normal]
```

getID() : 505
getWhen() : 1194256549547
getX() : 192
getY() : 346
getClickCount() : 0

5-7 ContainerEvent 类

java.awt.event.ContainerEvent 继承自 ComponentEvent → AWTEvent → EventObject → Object，此类将与容器有关的事件封装起来，包括组件的添加与删除，当此类事件发生时会立即启动 ContainerEvent，并配合 ContainerListener 运行。

1 构造函数

public **ContainerEvent(Component source, int id);**

其中，参数 source 是与容器事件相关的对象；id 是事件标识码。

2 类常量

public static final int **COMPONENT_ADDED;**
public static final int **COMPONENT_REMOVED;**

因为是类常量，所以不需配合构造函数生成的事件对象，可直接使用，用于标示容器事件的类型。

3 类方法

public void **processContainerEvent(ContainerEvent e);**

运行容器事件。

4 实例方法

public Object **getSource();**
public int **getID();**

配合构造函数生成的事件对象，返回其中参数的内容。

5-8 FocusEvent 类

java.awt.event.FocusEvent 继承自 ComponentEvent→AWTEvent→EventObject→Object，此类将与输入焦点有关的事件封装起来，当此类事件发生时会立即启动 FocusEvent，并配合 FocusListener 运行。

1 构造函数

public **FocusEvent(Component source, int id, boolean temporary);**

其中，参数 source 是与焦点事件相关的对象；id 是事件标识码；temporary 表示是否为暂时性的焦点事件。

2 类常量

public static final int **FOCUS_GAINED;** （如范例 28）
public static final int **FOCUS_LOST;** （如范例 29）

因为是类常量，所以不需配合构造函数生成的事件对象，可直接使用，用于标示焦点事件的类型。

3 类方法

```
public void processFocusEvent(FocusEvent e);   （如范例 28）
```

运行焦点事件。

4 实例方法

```
public Object getSource();
public int getID();
public boolean isTemporary();
```

配合构造函数生成的事件对象，返回其中参数的内容（如范例 28）。

设计范例 28，解释如何在框架中使用方法 enableEvents（AWTEvent.FOCUS_EVENT_MASK）启动焦点事件；如何使用方法 public void processFocusEvent(FocusEvent e) 运行焦点事件，同时读取其事件构造函数各参数的内容。

范例 28 文件 Ex5_8_1.java 的功能是解释单击选中框架的 Focus 事件。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex5_8_1 extends Frame {
04     public Ex5_8_1() {
05         super("Ex5_8_1");
06         setSize(350, 350);
07         setVisible(true);
08         enableEvents(AWTEvent.FOCUS_EVENT_MASK);
09     }

10     public void processFocusEvent(FocusEvent e) {
11         if(e.getID() == FocusEvent.FOCUS_GAINED) {
12             System.out.println("getSource() : " + e.getSource());
13             System.out.println("getID() : " + e.getID());
14             System.out.println("isTemporary() : " + e.isTemporary());
15         }
16     }

17     public static void main(String args[]) {
18       Ex5_8_1 workStart=new Ex5_8_1();
19     }
20 }
```

行 08　启动 Focus 事件（参考 5-3 节的类方法）。
行 10　运行 Focus 事件（参考 5-8 节的类方法）。
行 11　检查是不是单击选中框架的 Focus 事件（参考 5-8 节的 getID()方法与类常量 FOCUS_GAINED）。
行 12　读取事件的 source。
行 13　读取标识码 id。
行 14　读取是否为暂时性的焦点事件。

运行结果

单击选中框架时将产生下列结果，如图 5-8 所示。

```
getSource() :
Ex5_8_1[frame0,55,43,350x350,layout=java.awt.BorderLayout,title=Ex5_8_1,resizable,normal]
getID() : 1004
isTemporary() : false
```

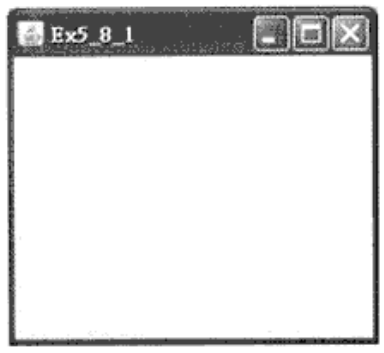


图 5-8

设计范例 29，解释当单击选中其他框架的 Focus 事件发生时（本例中单击 Ex5_8_2 之外的框架），如何读取其构造函数各参数的内容。

范例 29 文件 Ex5_8_2.java 的功能是解释单击其他框架的 Focus 事件。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex5_8_2 extends Frame {
04      public Ex5_8_2() {
05           super("Ex5_8_2");
06           setSize(350, 350);
07           setVisible(true);
08           enableEvents(AWTEvent.FOCUS_EVENT_MASK);
09      }
10      public void processFocusEvent(FocusEvent e) {
11           if(e.getID() == FocusEvent.FOCUS_LOST) {
12                System.out.println("getSource() : " + e.getSource());
13                System.out.println("getID() : " + e.getID());
14                System.out.println("isTemporary() : " + e.isTemporary());
15           }
16      }

17      public static void main(String args[]) {
18         Ex5_8_2 workStart=new Ex5_8_2();
19      }
20 }
```

行 11　检查是不是单击其他框架的 Focus 事件（参考 5-8 节的 getID() 方法与类常量 FOCUS_LOST）。

运行结果

单击其他框架时，即单击 Ex5_8_2 之外的框架时将产生下列结果，如图 5-9 所示。

```
getSource() :
Ex5_8_2[frame0,38,26,206x197,layout=java.awt.BorderLayout,title=Ex5_8_2,resizable,normal]
getID() : 1005
isTemporary() : true
```

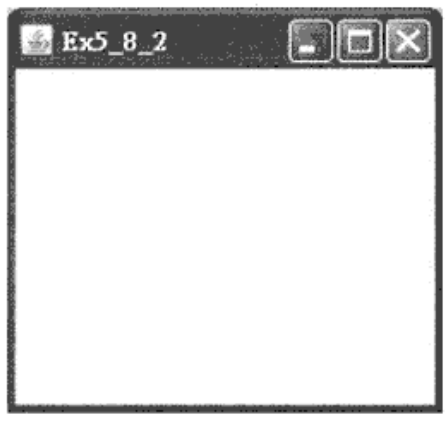


图 5-9

5-9 WindowEvent 类

java.awt.event.WindowEvent 继承自 ComponentEvent → AWTEvent → EventObject → Object，此类将与窗口有关的事件封装起来，当此类事件发生时会立即启动 WindowEvent，并配合 WindowListener 运行。

1 构造函数

public WindowEvent(Window source, int id);

其中，参数 source 是与窗口相关的对象；id 是事件标识码。

2 类常量

public static final int **WINDOW_CLOSING;** （如范例 30）
public static final int **WINDOW_ACTIVATED;**
public static final int **WINDOW_DEACTIVATED;**
public static final int **WINDOW_ICONIFIED;**
public static final int **WINDOW_DEICONIFIED;**

因为是类常量，所以不需配合构造函数生成的事件对象，可直接使用，用于标示窗口事件的类型。

3 类方法

public void **processWindowEvent(WindowEvent e);** （如范例 30）

运行窗口事件。

4 实例方法

public Window **getSource();**
public int **getID();**

配合构造函数生成的事件对象，返回其中参数的内容（如范例 30）。

设计范例 30，解释如何在框架中使用方法 enableEvents（AWTEvent.WINDOW_EVENT_ MASK）启动窗口事件；如何使用方法 public void processWindowEvent(WindowEvent e) 运行窗口事件，同时读取其事件构造函数各参数的内容。

范例 30 文件 Ex5_9_1.java 的功能是解释使用类常量 WINDOW_CLOSING 与单击框架右上方的“×”按钮的运行结果。

```
01 import java.awt.*;
02 import java.awt.event.*;
```

```
03 public class Ex5_9_1 extends Frame {
04      public Ex5_9_1() {
05          super("Ex5_9_1");
06          setSize(350, 350);
07          setVisible(true);
08          enableEvents(AWTEvent.WINDOW_EVENT_MASK);
09      }

10      public void processWindowEvent(WindowEvent e) {
11          if(e.getID() == WindowEvent.WINDOW_CLOSING) {
12              System.out.println("getSource() : " + e.getSource());
13              System.out.println("getID() : " + e.getID());
14          }
15      }

16      public static void main(String args[]) {
17        Ex5_9_1 workStart=new Ex5_9_1();
18      }
19 }
```

行 08　启动 Window 事件（参考 5–3 节的类方法）。

行 10　运行 Window 事件（参考 5–9 节的类方法）。

行 11　检查是不是单击框架右上方的“×”按钮的 Window 事件（参考 5–9 节的 getID()方法与类常量 WINDOW_CLOSING）。

行 12　读取事件的 source。

行 13　读取标识码 id。

运行结果

单击框架右上方的“×”按钮时将产生下列结果，如图 5–10 所示。

```
getSource() :
Ex5_9_1[frame0,48,23,350x350,layout=java.awt.BorderLayout,title=Ex5_9_1,resizable,normal]
getID() : 201
```

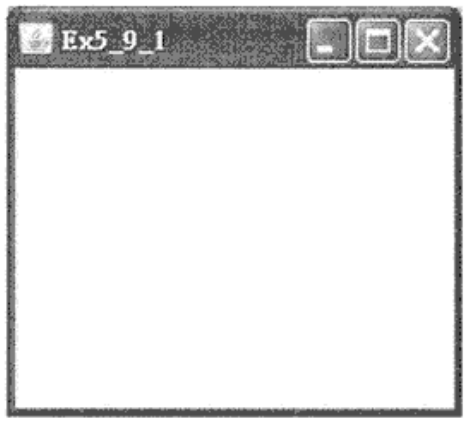


图 5-10

到目前为止，本书各范例所创建的窗口框架都以 Ctrlt+C 来关闭，无法使用框架右上方的“×”按钮来关闭。范例 31 将以类常量 WINDOW_CLOSING 与 System 类的类方法 public static void exit(int status)来关闭框架。

范例 31　文件 Ex5_9_2.java 的功能是解释如何通过单击框架右上方的“×”按钮来关闭窗口。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex5_9_2 extends Frame {
04      public Ex5_9_2() {
05           super("Ex5_9_2");
06           setSize(350, 350);
07           setVisible(true);
08           enableEvents(AWTEvent.WINDOW_EVENT_MASK);
09      }

10      public void processWindowEvent(WindowEvent e) {
11           if(e.getID() == WindowEvent.WINDOW_CLOSING) {
12               System.exit(0);
13              }
14      }

15      public static void main(String args[]) {
16         Ex5_9_2 workStart=new Ex5_9_2();
17      }
18 }
```

行 11　检查是不是单击框架右上方的“×”按钮的 Window 事件（参考 5–9 节的 getID()方法与类常量 WINDOW_CLOSING）。

行 12　关闭窗口框架。

运行结果

当单击框架右上方的“×”按钮时，将关闭窗口框架。运行时窗口如图 5–11 所示。

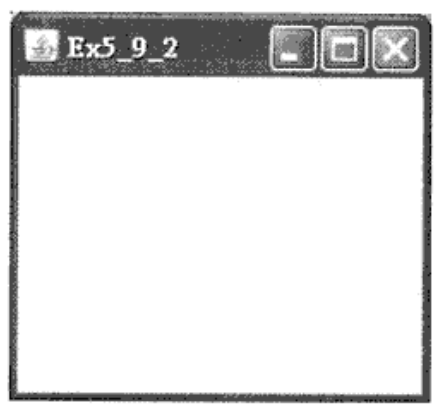


图 5-11

读者也可参考本系列丛书第二册《Java 典型应用彻查 1000 例——网络应用开发》的 19--5 节，在其中我们曾使用程序代码：

```
addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent evt) {
          System.exit(0);
      }
  });
```

它们与范例 31 具有相同的功能。

范例 32　文件 Ex5_9_3.java 的功能是解释如何通过单击框架右上方的“X”按钮来关闭窗口（参考本系列丛书第二册《Java 典型应用彻查 1000 例——网络应用开发》的 19--5 节）。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 class Ex5_9_3 extends Frame {
04     public Ex5_9_3() {
05         super("Ex5_9_3");
06         setSize(350, 350);
07         setVisible(true);

08         addWindowListener(new WindowAdapter() {
09             public void windowClosing(WindowEvent evt) {
10                 System.exit(0);
11             }
12         });
13     }

14     public static void main(String[] args) {
15         Ex5_9_3 workstart = new Ex5_9_3();
16     }
17 }
```

行 08～12　通过 WindowListener 接口、WindowAdapter 类与 WindowEvent 类来设计程序代码，以关闭窗口（单击窗口右上方的“×”按钮来关闭窗口）。

运行结果

运行时窗口如图 5-12 所示。

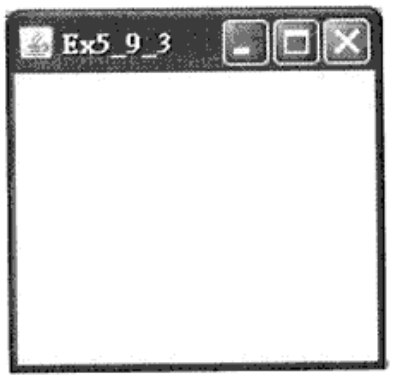


图 5-12

5-10 习题

1. Java 所支持的底层事件类有哪些？
2. AWTEvent 类的功能有哪些？
3. ComponentEvent 类的功能有哪些？
4. KeyEvent 类的功能有哪些？
5. MouseEvent 类的功能有哪些？
6. ContainerEvent 类的功能有哪些？
7. FocusEvent 类的功能有哪些？
8. WindowEvent 类的功能有哪些？

Chapter 06 鼠标事件应用

6-1 简介

常用的鼠标事件行为有按键（PRESSED）、单击（CLICKED）和释放（RELEASED）。每次发生鼠标事件时，我们可以立即读取该事件的源（Source）、标识码（ID）、发生时间（When）、x 轴坐标和 y 轴坐标。

由于上述信息是伴随事件的发生而产生的，所以我们可利用这些信息引导后续操作的执行，如移动（Moved）、拖动（Dragged）、选择（Selected）、随动（Followed）、棋盘（Chessboard）等。

回顾 1–3 节，线程绘图流程是以 Thread 驱动 run()方法，再以 repaint()调用 paint()方法，这样可获得线程并行的效果。

6-2 移动

回顾范例 26，当在窗口框架内任一点单击鼠标左键时会发生鼠标事件，此时我们可读取鼠标指针当前的位置坐标(x, y)。再参考范例 16，我们可将图像对应长方形的左上角绘制于新位置坐标。这样图案将随鼠标指针的位置而移动。

范例 33 文件 Ex6_2.java 的功能是解释图像移动（Moved）的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex6_2 extends Frame implements Runnable {

04     int x=50, y=50;
05     Image img;

06     public static void main(String args[]) {
07         Ex6_2 workStart=new Ex6_2();
08     }
```

```
09    public Ex6_2() {
10        super("Ex6_2");
11        setSize(350,350);

12        Toolkit tk = Toolkit.getDefaultToolkit();
13        img = tk.getImage("img7_2.jpg");

14        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
15        enableEvents(AWTEvent.MOUSE_EVENT_MASK);

16        setVisible(true);
17        new Thread(this).start();
18    }

19    public void processWindowEvent(WindowEvent e) {
20        if(e.getID() == WindowEvent.WINDOW_CLOSING) {
21            System.exit(0);
22          }
23    }

24    public void processMouseEvent(MouseEvent e) {
25        if(e.getID() == MouseEvent.MOUSE_PRESSED) {
26            x = e.getX();
27            y = e.getY();
28        }
29    }

30    public void run() {
31      while(true) {
32          repaint();
33          try{Thread.sleep(250);}
34          catch(InterruptedException e) {;}
35        }
36    }

37    public void paint(Graphics g) {
38        g.drawImage(img, x, y, this);
39    }
40 }
```

行 15	启动鼠标事件（如 5–3 节）。
行 24～29	单击鼠标左键时会发生鼠标事件，同时记录鼠标指针当前的位置坐标(x，y)（如范例 26）。
行 38	将图像绘制于新位置坐标(x，y)（如范例 16）。
行 17	以线程驱动行 30～36 的 run()方法（参考 1–3 节）。
行 32	以 repaint()方法驱动行 37～39 的 paint()方法，进行更新绘制。

运行结果

运行结果如图 6–1 所示。

当在窗口框架内任一点单击鼠标左键时，图像将随鼠标指针的位置而移动。

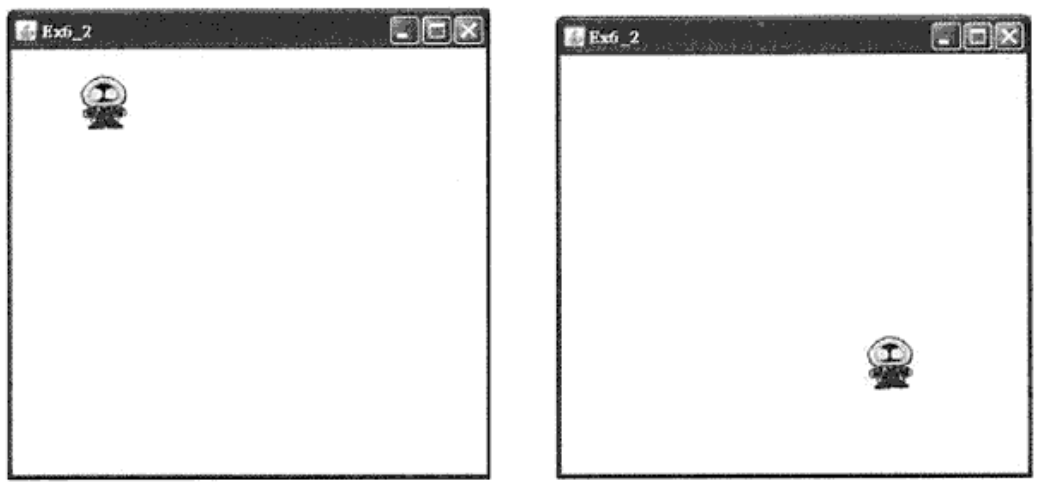


图 6-1

6-3 拖动

在图像上单击鼠标左键，拖动后释放鼠标左键，图像将随鼠标指针当前的位置而移动，这样的运行方式也就是所谓的“拖动”。

在单击图像时需先设置图像单击范围。如图 6-2 所示，假设图像宽为 w，高为 h，图像对应长方形左上角的坐标为(x1，y1)。

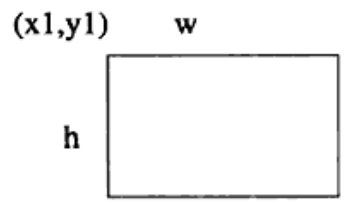


图 6-2

则图像单击范围(x，y)为

在 x 轴：　(x >= x1) && (x <= (x1 + w))
在 y 轴：　(y >= y1) && (y <= (y1 + h))

范例 34 文件 Ex6_3.java 的功能是解释图像拖动（Dragged）的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex6_3 extends Frame implements Runnable {

04     int x=50, y=50, flag=0;
05     Image img;

06     public static void main(String args[]) {
07        Ex6_3 workStart=new Ex6_3();
08     }

09     public Ex6_3() {
10          super("Ex6_3");
11          setSize(350,350);

12          Toolkit tk = Toolkit.getDefaultToolkit();
13          img = tk.getImage("img7_3.jpg");

14          enableEvents(AWTEvent.WINDOW_EVENT_MASK);
```

```
15          enableEvents(AWTEvent.MOUSE_EVENT_MASK);

16          setVisible(true);
17          new Thread(this).start();
18      }

19      public void processWindowEvent(WindowEvent e) {
20          if(e.getID() == WindowEvent.WINDOW_CLOSING) {
21              System.exit(0);
22            }
23      }

24      public void processMouseEvent(MouseEvent e) {
25          if(e.getID() == MouseEvent.MOUSE_PRESSED)
26              if(((e.getX() >= x) && (e.getX() <= (x+40))) &&
27                 ((e.getY() >= y) && (e.getY() <= (y+40))))   {
28                  flag = 1;
29               }

30              if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag == 1)) {
31                  x = e.getX();
32                  y = e.getY();
33                  flag = 0;
34              }
35       }

36      public void run() {
37        while(true) {
38          repaint();
39          try{Thread.sleep(250);}
40          catch(InterruptedException e) {;}
41        }
42      }

43      public void paint(Graphics g) {
44          g.drawImage(img, x, y, this);
45      }
46  }
```

行 15	启动鼠标事件。
行 24	当有鼠标事件发生时。
行 25～27	设置图像单击范围（本例中图像大小为 40x40），检查是否是在图像单击范围内单击鼠标左键。
行 28	设置 flag 为 1。
行 30～32	拖动鼠标后，如果释放鼠标时 flag 为 1，则读取鼠标指针当前的位置坐标(x，y)。
行 44	将图像放在新位置坐标(x，y)。
行 17	以线程驱动行 36～42 的 run()方法（参考 1—3 节）。
行 38	以 repaint()方法驱动行 43～45 的 paint()方法，进行更新绘制。

运行结果

运行结果如图 6—3 所示。

在图像上单击鼠标左键，拖动后释放鼠标，图像将随鼠标指针的位置而移动。

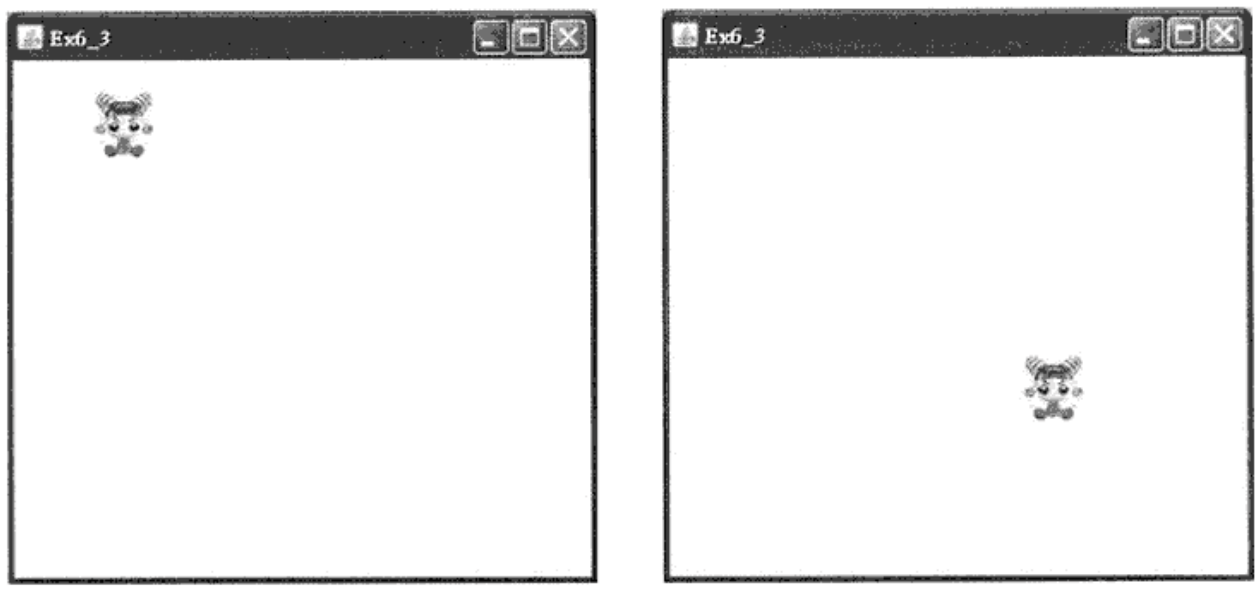


图 6-3

6-4 选择

在 6–3 节中，我们已实现单击一幅图像，执行拖动使之移动。本节将介绍如何在多幅图像中单击其中一幅，执行拖动使之移动，而不干扰其他图像。即在多幅图像中选择需要的图像，执行需要的操作。

范例 35 文件 Ex6_4.java 的功能是解释选择图像（Selected）与拖动（Dragged）的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex6_4 extends Frame implements Runnable {

04    int x1=50, y1=50, x2=100, y2=100, x3=150, y3=150;
05    int flag=0;
06    Image img1, img2, img3;

07    public static void main(String args[]) {
08       Ex6_4 workStart=new Ex6_4();
09    }

10    public Ex6_4() {
11       super("Ex6_4");
12       setSize(350,350);

13       Toolkit tk = Toolkit.getDefaultToolkit();
14       img1 = tk.getImage("img7_4_01.jpg");
15       img2 = tk.getImage("img7_4_02.jpg");
16       img3 = tk.getImage("img7_4_03.jpg");

17       enableEvents(AWTEvent.WINDOW_EVENT_MASK);
18       enableEvents(AWTEvent.MOUSE_EVENT_MASK);

19       setVisible(true);
20       new Thread(this).start();
```

```
}
public void processWindowEvent(WindowEvent e) {
    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
    }
}

public void processMouseEvent(MouseEvent e) {
   if(e.getID() == MouseEvent.MOUSE_PRESSED)
       if(((e.getX() >= x1) && (e.getX() <= (x1+40))) &&
          ((e.getY() >= y1) && (e.getY() <= (y1+40))))   {
                flag = 1;
    }

    if(((e.getX() >= x2) && (e.getX() <= (x2+40))) &&
        ((e.getY() >= y2) && (e.getY() <= (y2+40))))   {
                flag = 2;
     }

     if(((e.getX() >= x3) && (e.getX() <= (x3+40))) &&
         ((e.getY() >= y3) && (e.getY() <= (y3+40))))   {
                flag = 3;
      }

      if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag == 1)) {
          x1 = e.getX();
          y1 = e.getY();
          flag = 0;
     }

     if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag == 2)) {
          x2 = e.getX();
          y2 = e.getY();
          flag = 0;
     }

     if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag == 3)) {
          x3 = e.getX();
          y3 = e.getY();
          flag = 0;
     }
 }

 public void run() {
    while(true) {
       repaint();
       try{Thread.sleep(250);}
       catch(InterruptedException e) {;}
    }
 }

 public void paint(Graphics g) {
```

```
65          g.drawImage(img1, x1, y1, this);
66          g.drawImage(img2, x2, y2, this);
67          g.drawImage(img3, x3, y3, this);
68      }
69 }
```

行 14～16	读取 3 幅图像。
行 18	启动鼠标事件。
行 27	当有鼠标事件发生时。
行 28～32	设置图像选择范围（本例中图像大小为 40x40），如果在第一幅图像的选择范围内单击鼠标左键，则设置 flag 为 1。
行 41～45	拖动鼠标后，如果释放鼠标时 flag 为 1，则读取鼠标指针当前的位置坐标(x1，y1)。
行 65	将图像放在新位置坐标(x1，y1)。
行 20	以线程驱动行 57～63 的 run()方法（参考 1-3 节）。
行 59	以 repaint()方法驱动行 64～68 的 paint()方法，进行更新绘制。

运行结果

运行结果如图 6-4 所示。

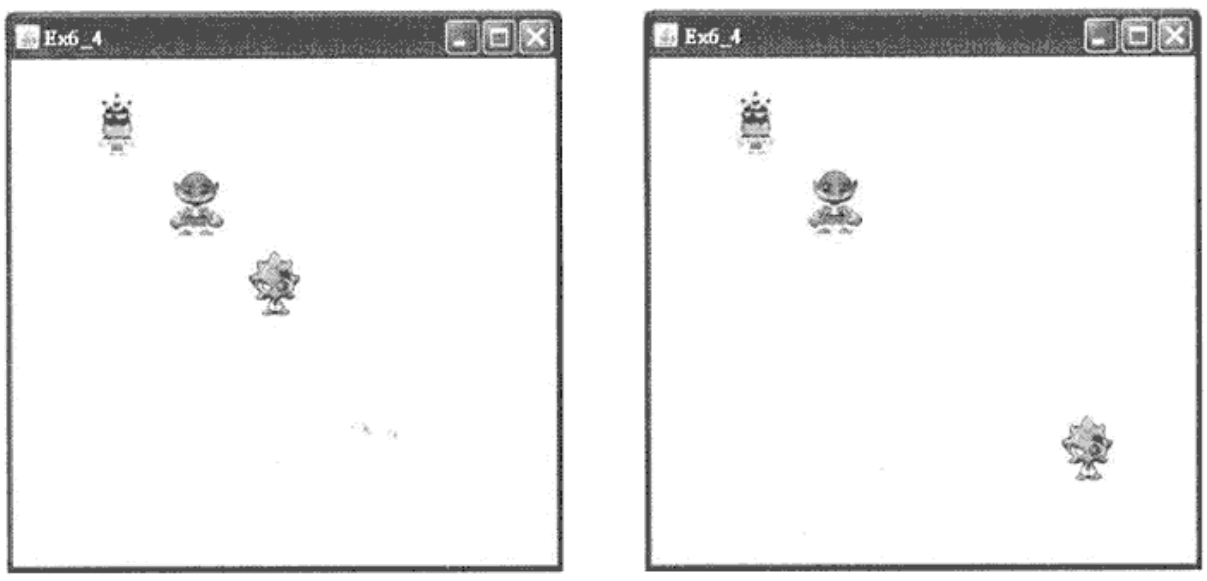


图 6-4

在三幅图像中单击其中一幅，单击鼠标左键，拖动后释放鼠标，图像将随鼠标指针当前的位置而移动，运行过程中不会干扰另两幅图像。

6-5 随动

单击鼠标左键选择图像，拖动鼠标后再释放鼠标时，读取鼠标指针当前的随动坐标(fx，fy)。比较原始坐标(x，y)与随动坐标(fx，fy)，将它们的差除以 n，用于设置(dx，dy)，当图像随动到坐标(fx，fy)时，设置(dx，dy) = (0，0)，使图像不再移动。参考范例 18，在每一阶段的新位置坐标(x，y)上更新绘制图像，这样的运行方式也就是所谓的“随动 (Followed)”。

在比较原始坐标(x,y)与随动坐标(fx,fy)时，如果图像向右移动，由(x,y)随动到(fx,fy)，即 if((dx>0)&&((x+w/2)>=fx))；或图像向左移动，由(x,y)随动到(fx,fy)，即 if((dx<0)&&((x+w/2)

<= fx))，此时设置 dx =0，使图像不再左右移动。

如果图像向下移动，由(x，y) 随动到(fx，fy)，即 if ((dy > 0) && ((y+ h/2) >= fy))；或图像向上移动，由(x，y) 随动到(fx，fy)，即 if ((dy < 0) && ((y+ h/2) <= fy))，此时设置 dy =0，使图像不再上下移动。

范例 36 文件 Ex6_5_1 .java 的功能是解释单幅图像与随动 (Followed) 的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex6_5_1 extends Frame implements Runnable {

04      int x=50, y=50, fx=0, fy=0, dx=0, dy=0, flag=0;
05      Image img;

06      public static void main(String args[]) {
07         Ex6_5_1 workStart=new Ex6_5_1();
08      }

09      public Ex6_5_1() {
10         super("Ex6_5_1");
11         setSize(350,350);

12         Toolkit tk = Toolkit.getDefaultToolkit();
13         img = tk.getImage("fly.GIF");

14         enableEvents(AWTEvent.WINDOW_EVENT_MASK);
15         enableEvents(AWTEvent.MOUSE_EVENT_MASK);
16         setVisible(true);
17         new Thread(this).start();
18      }

19      public void processWindowEvent(WindowEvent e) {
20         if(e.getID() == WindowEvent.WINDOW_CLOSING) {
21            System.exit(0);
22         }
23      }

24      public void processMouseEvent(MouseEvent e) {
25         if(e.getID() == MouseEvent.MOUSE_PRESSED)
26            if(((e.getX() >= x) && (e.getX() <= (x+60))) &&
27               ((e.getY() >= y) && (e.getY() <= (y+50))))   {
28                  flag = 1;
29            }

30            if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag == 1)) {
31               fx = e.getX();
32               fy = e.getY();

33               dx = (fx - x) / 10;
34               dy = (fy - y) / 10;
35               flag = 0;
36            }
37         }

38      public void run() {
39         while(true) {
```

```
40          x = x + dx;
41          y = y + dy;

42          if(((dx > 0) && ((x+30) >= fx)) || ((dx < 0) && ((x+30) <= fx))) dx = 0;
43          if(((dy > 0) && ((y+25) >= fy)) || ((dy < 0) && ((y+25) <= fy))) dy = 0;

44          repaint();
45          try{Thread.sleep(250);}
46          catch(InterruptedException e) {;}
47        }
48      }

49    public void paint(Graphics g) {
50          g.drawImage(img, x, y, this);
51      }
52 }
```

行 24～29	利用鼠标左键单击图像，设置 flag 为 1。
行 30～32	拖动鼠标后，如果释放鼠标时 flag 为 1，则读取鼠标指针当前的随动坐标(fx，fy)。
行 33～35	比较原始坐标(x，y) 与随动坐标(fx，fy)，将它们的差除以 n（本例中 n 为 10），用于设置(dx，dy)，同时将 flag 还原成 0。
行 40～41	参考范例 18，设置每一阶段的新位置坐标(x，y)。
行 42～43	当图像随动至坐标(fx，fy) 时，设置(dx，dy) = (0，0)，使图像不再移动。
行 17	以线程驱动行 38～48 的 run()方法（参考 1-3 节）。
行 44	以 repaint()方法驱动行 49～51 的 paint()方法，进行更新绘制。

运行结果

运行结果如图 6-5 所示。

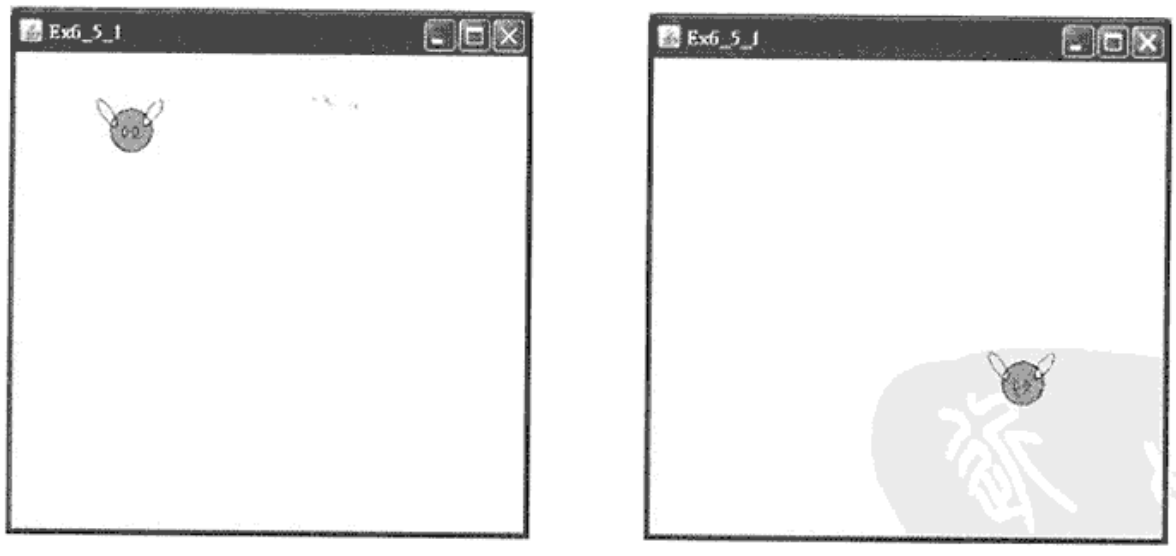


图 6-5

利用鼠标左键单击图像，拖动鼠标后再释放鼠标时，读取鼠标指针当前的随动坐标(fx，fy)，此时图像将以动画方式从原位置随动至坐标位置(fx，fy)。

范例 37 文件 Ex6_5_2.java 的功能是解释多幅图像与随动（Followed）的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
```

```
03 public class Ex6_5_2 extends Frame implements Runnable {

04    int x=50, y=50, fx=0, fy=0, dx=0, dy=0;
05    int flag_type_img=0, flag_num_img=0, num=0;
06    Image img0, img1, img2;

07    public static void main(String args[]) {
08       Ex6_5_2 workStart=new Ex6_5_2();
09    }

10    public Ex6_5_2() {
11          super("Ex6_5_2");
12          setSize(350,350);

13          Toolkit tk = Toolkit.getDefaultToolkit();
14          img0 = tk.getImage("fly0.GIF");
15          img1 = tk.getImage("fly1.GIF");
16          img2 = tk.getImage("fly2.GIF");

17          enableEvents(AWTEvent.WINDOW_EVENT_MASK);
18          enableEvents(AWTEvent.MOUSE_EVENT_MASK);

19          setVisible(true);
20          new Thread(this).start();
21     }

22     public void processWindowEvent(WindowEvent e) {
23          if(e.getID() == WindowEvent.WINDOW_CLOSING) {
24             System.exit(0);
25          }
26    }

27    public void processMouseEvent(MouseEvent e) {
28          if(e.getID() == MouseEvent.MOUSE_PRESSED)
29             if(((e.getX() >= x) && (e.getX() <= (x+80))) &&
               ((e.getY() >= y) && (e.getY() <= (y+60))))   {
30                flag_type_img = 1;
31             }

32             if((e.getID() == MouseEvent.MOUSE_RELEASED) &&
                        (flag_type_img == 1)) {
33                fx = e.getX();
34                fy = e.getY();

35                dx = (fx - x) / 10;
36                dy = (fy - y) / 10;
37                flag_type_img = 0;
38             }
39     }

40    public void run() {
41       while(true) {
42          x = x + dx;
43          y = y + dy;

44          flag_num_img = num % 3;

45          if(((dx > 0) && ((x+40) >= fx)) || ((dx < 0) && ((x+40) <= fx))) dx = 0;
46          if(((dy > 0) && ((y+30) >= fy)) || ((dy < 0) && ((y+30) <= fy))) dy = 0;
```

```
47            repaint();
48            num = num + 1;
49            try{Thread.sleep(250);}
50            catch(InterruptedException e) {;}
51         }
52    }

53    public void paint(Graphics g) {
54       if(flag_num_img == 0) g.drawImage(img0, x, y, this);
55       if(flag_num_img == 1) g.drawImage(img1, x, y, this);
56       if(flag_num_img == 2) g.drawImage(img2, x, y, this);
57    }
58 }
```

代码说明请参考范例 36 和范例 19 的解说。

运行结果

运行结果如图 6–6 所示。

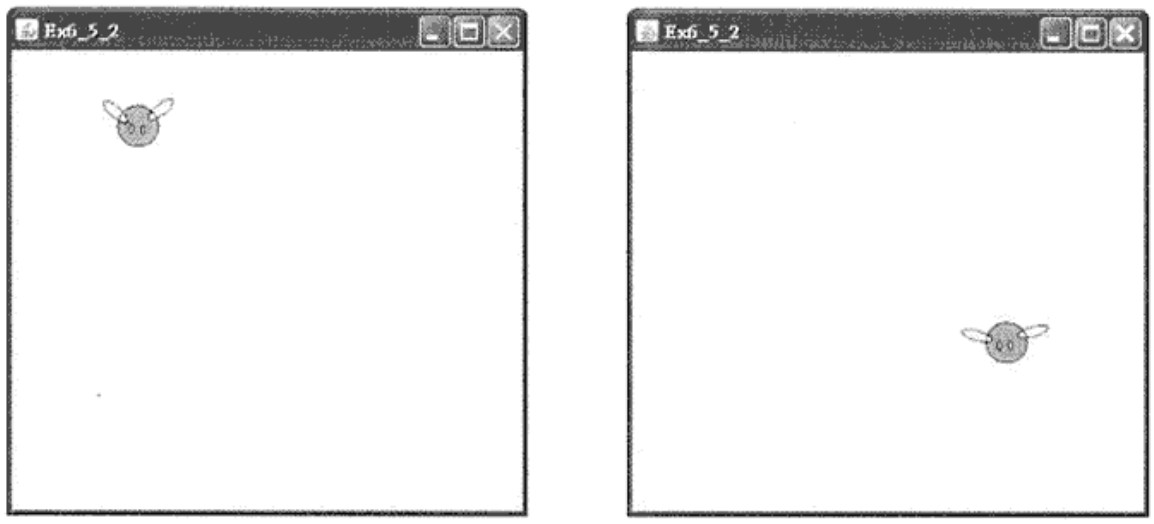


图 6-6

利用鼠标左键单击图像，拖动鼠标后再释放鼠标时，读取鼠标指针当前的随动坐标(fx，fy)，此时图像将以生动的动画方式从原位置随动至坐标位置(fx，fy)。

6-6 线程

我们一再强调，设计 Java 动画游戏必须参考 1–3 节，原因是通过它可同时运行多个图像的移动，多个图像并行各自的操作，互不干扰。

参考范例 36 设计范例 38，使用两个单幅图像，并实现两个图像同时并行移动的功能。单击第一个图像，拖动后释放鼠标；立即再单击第二个图像，拖动后释放鼠标。两个图像将同时并行，各自移动到预定的位置，且互不干扰。

范例 38 参考范例 36，文件 Ex6_6_1.java 的功能是解释多个单幅图像与并行的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex6_6_1 extends Frame implements Runnable {
```

```
04  int x1=50, y1=50, fx1=0, fy1=0, dx1=0, dy1=0, flag1=0;
05  int x2=100, y2=100, fx2=0, fy2=0, dx2=0, dy2=0, flag2=0;
06  Image img1, img2;

07  public static void main(String args[]) {
08    Ex6_6_1 workStart=new Ex6_6_1();
09  }

10  public Ex6_6_1() {
11    super("Ex6_6_1");
12    setSize(350,350);

13    Toolkit tk = Toolkit.getDefaultToolkit();
14    img1 = tk.getImage("fly1.GIF");
15    img2 = tk.getImage("fly2.GIF");

16    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
17    enableEvents(AWTEvent.MOUSE_EVENT_MASK);

18    setVisible(true);
19    new Thread(this).start();
20  }

21  public void processWindowEvent(WindowEvent e) {
22    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
23        System.exit(0);
24    }
25  }

26  public void processMouseEvent(MouseEvent e) {
27    if(e.getID() == MouseEvent.MOUSE_PRESSED)
28      if(((e.getX() >= x1) && (e.getX() <= (x1+80))) &&
29        ((e.getY() >= y1) && (e.getY() <= (y1+60))))  {
30          flag1 = 1;
31      }
32      if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag1 == 1)) {
33        fx1 = e.getX();
34        fy1 = e.getY();
35        dx1 = (fx1 - x1) / 50;
36        dy1 = (fy1 - y1) / 50;
37        flag1 =0;
38      }

39      if(((e.getX() >= x2) && (e.getX() <= (x2+80))) &&
40        ((e.getY() >= y2) && (e.getY() <= (y2+60))))  {
41          flag2 = 1;
42      }
43      if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag2 == 1)) {
44        fx2 = e.getX();
45        fy2 = e.getY();
46        dx2 = (fx2 - x2) / 50;
47        dy2 = (fy2 - y2) / 50;
48        flag2 = 0;
49      }
50  }

51  public void run() {
52    while(true) {
53      x1 = x1 + dx1;
54      y1 = y1 + dy1;
```

```
55          if(((dx1 > 0) && ((x1+40) >= fx1)) || ((dx1 < 0) && ((x1+40) <= fx1))) dx1 = 0;
56          if(((dy1 > 0) && ((y1+30) >= fy1)) || ((dy1 < 0) && ((y1+30) <= fy1))) dy1 = 0;

57          x2 = x2 + dx2;
58          y2 = y2 + dy2;
59          if(((dx2 > 0) && ((x2+40) >= fx2)) || ((dx2 < 0) && ((x2+40) <= fx2))) dx2 = 0;
60          if(((dy2 > 0) && ((y2+30) >= fy2)) || ((dy2 < 0) && ((y2+30) <= fy2))) dy2 = 0;

61          repaint();
62          try{Thread.sleep(250);}
63          catch(InterruptedException e) {;}
64        }
65    }

66    public void paint(Graphics g) {
67        g.drawImage(img1, x1, y1, this);
68        g.drawImage(img2, x2, y2, this);
69    }
70 }
```

参考范例 36 的解说，本例中使用两个单幅图像，并实现了两个图像同时并行移动的功能。

运行结果

运行结果如图 6-7 所示。

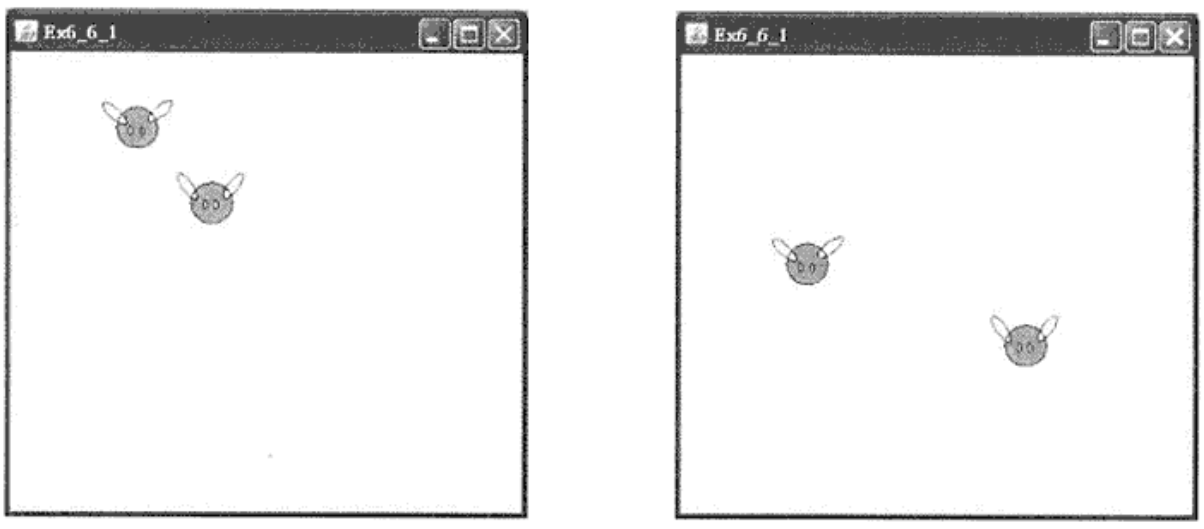


图 6-7

以鼠标选择第一个图像，拖动后释放鼠标；立即再选择第二个图像，拖动后释放鼠标。两个图像将同时并行，各自移动至预定的位置，且互不干扰。

范例 39 参考范例 37，文件 Ex6_6_2.java 的功能是解释多个多幅图像与并行的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex6_6_2 extends Frame implements Runnable {

04    int x1=50, y1=50, fx1=0, fy1=0, dx1=0, dy1=0;
05    int flag1_type_img=0, flag1_num_img=0, num1=0;
06    Image img10, img11, img12;

07    int x2=100, y2=100, fx2=0, fy2=0, dx2=0, dy2=0;
```

```java
int flag2_type_img=0, flag2_num_img=0, num2=0;
Image img20, img21, img22;

public static void main(String args[]) {
  Ex6_6_2 workStart=new Ex6_6_2();
}

public Ex6_6_2() {
  super("Ex6_6_2");
  setSize(350,350);

  Toolkit tk = Toolkit.getDefaultToolkit();
  img10 = tk.getImage("fly10.GIF");
  img11 = tk.getImage("fly11.GIF");
  img12 = tk.getImage("fly12.GIF");

  img20 = tk.getImage("fly20.GIF");
  img21 = tk.getImage("fly21.GIF");
  img22 = tk.getImage("fly22.GIF");

  enableEvents(AWTEvent.WINDOW_EVENT_MASK);
  enableEvents(AWTEvent.MOUSE_EVENT_MASK);

  setVisible(true);
  new Thread(this).start();
}

public void processWindowEvent(WindowEvent e) {
  if(e.getID() == WindowEvent.WINDOW_CLOSING) {
     System.exit(0);
  }
}

public void processMouseEvent(MouseEvent e) {
  if(e.getID() == MouseEvent.MOUSE_PRESSED)
    if(((e.getX() >= x1) && (e.getX() <= (x1+80))) &&
      ((e.getY() >= y1) && (e.getY() <= (y1+60))))   {
          flag1_type_img = 1;
    }
    if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag1_type_img == 1)) {
        fx1 = e.getX();
        fy1 = e.getY();
        dx1 = (fx1 - x1) / 50;
        dy1 = (fy1 - y1) / 50;
        flag1_type_img = 0;
    }

     if(((e.getX() >= x2) && (e.getX() <= (x2+80))) &&
       ((e.getY() >= y2) && (e.getY() <= (y2+60))))   {
           flag2_type_img = 1;
     }
     if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag2_type_img == 1)) {
         fx2 = e.getX();
         fy2 = e.getY();
         dx2 = (fx2 - x2) / 50;
         dy2 = (fy2 - y2) / 50;
         flag2_type_img = 0;
     }
  }
```

```
58  public void run() {
59     while(true) {
60        x1 = x1 + dx1;
61        y1 = y1 + dy1;
62        flag1_num_img = num1 % 3;
63        if(((dx1 > 0) && ((x1+40) >= fx1)) || ((dx1 < 0) && ((x1+40) <= fx1))) dx1 = 0;
64        if(((dy1 > 0) && ((y1+30) >= fy1)) || ((dy1 < 0) && ((y1+30) <= fy1))) dy1 = 0;

65        x2 = x2 + dx2;
66        y2 = y2 + dy2;
67        flag2_num_img = num2 % 3;
68        if(((dx2 > 0) && ((x2+40) >= fx2)) || ((dx2 < 0) && ((x2+40) <= fx2))) dx2 = 0;
69        if(((dy2 > 0) && ((y2+30) >= fy2)) || ((dy2 < 0) && ((y2+30) <= fy2))) dy2 = 0;

70        repaint();
71        num1 = num1 + 1;
72        num2 = num2 + 1;
73        try{Thread.sleep(250);}
74        catch(InterruptedException e) {;}
75     }
76  }

77  public void paint(Graphics g) {
78     if(flag1_num_img == 0) g.drawImage(img10, x1, y1, this);
79     if(flag1_num_img == 1) g.drawImage(img11, x1, y1, this);
80     if(flag1_num_img == 2) g.drawImage(img12, x1, y1, this);

81     if(flag2_num_img == 0) g.drawImage(img20, x2, y2, this);
82     if(flag2_num_img == 1) g.drawImage(img21, x2, y2, this);
83     if(flag2_num_img == 2) g.drawImage(img22, x2, y2, this);
84  }
85 }
```

参考范例 37 的解说，本例中使用两个多幅图像，并实现了两幅图像同时并行移动的功能。

运行结果

运行结果如图 6–8 所示。

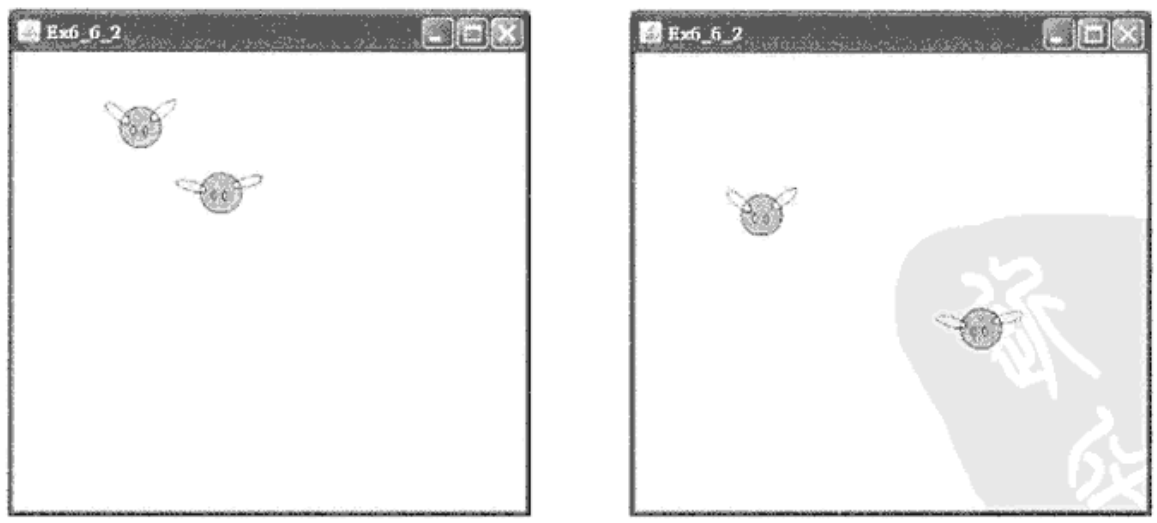


图 6-8

以鼠标选择第一幅图像，拖动后释放鼠标；立即再选择第二幅图像，拖动后释放鼠标。两幅图像将同时并行，各自移动到预定的位置，且互不干扰。

6-7 棋盘

从程序设计的角度看，弈棋游戏可分为以下两类。

（1）落子弈棋（如五子棋、围棋等），在棋盘预定位置显示放置的棋子。在设计上，我们可参考范例 33，在鼠标单击的位置上绘制棋子。

（2）叫吃弈棋（如象棋、国际象棋等），选择一个棋子，从原位置拖动至新位置，如果新位置与敌方棋子冲突，比较这两个棋子的优势，胜者保存，败者消失。在设计上，我们可参考范例 35，以鼠标选择棋子并拖动至新位置。

本书的重点是介绍入门程序，为了让读者清楚简单的设计概念，本书仅介绍落子弈棋（包括本书第 3 部分“在线游戏”）。至于叫吃弈棋，笔者将另编于本系列丛书高级动画游戏中，敬请指教。在落子弈棋程序设计上，除了范例 33 已克服的问题之外，还有一些问题需要克服，如落子范围的设置，落子图案的显示等。

落子范围的设置：一个未设置落子范围的棋盘，其棋子的显示可能如图 6–9 所示。

在这种情况下，棋子的位置是由鼠标指针的位置坐标决定的，因此棋子很难排列整齐。改进的方法是将棋格中任何坐标均更换成该棋格的中心点坐标，如图 6–10 所示。

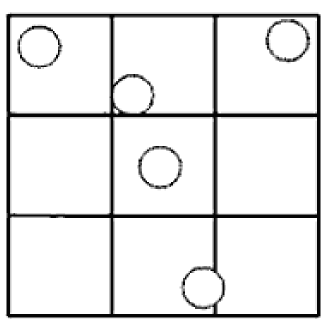


图 6-9

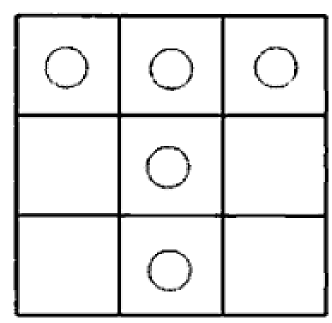


图 6-10

假设有棋格如图 6–11 所示，四个边角的坐标分别为 (x_1, y_1)、(x_2, y_1)、(x_1, y_2)、(x_2, y_2)，则其中心点坐标的 x 轴坐标应为 $(x_1+x_2)/2$，y 轴坐标应为 $(y_1+y_2)/2$。若考虑棋子图案对应正方形的宽“w”和高“h”，则图案的 x 轴坐标应为 $((x_1+x_2)/2)-(w/2)$，y 轴坐标应为 $((y_1+y_2)/2)-(h/2)$。这些数据可供参考，在程序设计时应视实际环境而定。

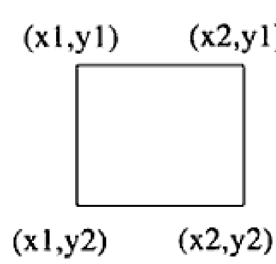


图 6-11

落子图案的显示：设置落子标示 flag，当某棋格被落子时，立即设置其标示 flag 为 1，若棋格的 flag 为 1，则在任何时刻都要显示棋子图案。

范例 40 文件 Ex6_7.java 的功能是解释井字棋盘的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
```

```
03 public class Ex6_7 extends Frame implements Runnable {

04    int x, y;
05    Image img;
06    int[] area_flag = new int[9];
07    int i;

08    public static void main(String args[]) {
09       Ex6_7 workStart=new Ex6_7();
10    }

11    public Ex6_7() {
12       super("Ex6_7");
13       setSize(250,280);

14       Toolkit tk = Toolkit.getDefaultToolkit();
15       img = tk.getImage("Circle.GIF");

16       enableEvents(AWTEvent.WINDOW_EVENT_MASK);
17       enableEvents(AWTEvent.MOUSE_EVENT_MASK);

18       setVisible(true);
19       new Thread(this).start();
20    }

21    public void processWindowEvent(WindowEvent e) {
22       if(e.getID() == WindowEvent.WINDOW_CLOSING) {
23            System.exit(0);
24       }
25    }

26    public void processMouseEvent(MouseEvent e) {
27       if(e.getID() == MouseEvent.MOUSE_PRESSED) {
28            x = e.getX();
29            y = e.getY();
30       }
31    }

32    public void run() {
33       while(true) {
34          if((x>30)&&(x<90)&&(y>70)&&(y<130)&&(area_flag[0]==0))
                 {area_flag[0] = 1;}
35          else if ((x>90)&&(x<150)&&(y>70)&&(y<130)&&(area_flag[1]==0))
                 {area_flag[1] = 1;}
36          else if ((x>150)&&(x<210)&&(y>70)&&(y<130)&&(area_flag[2]==0))
                 {area_flag[2] = 1;}
37          else if((x>30)&&(x<90)&&(y>130)&&(y<190)&&(area_flag[3]==0))
                 {area_flag[3] = 1;}
38          else if((x>90)&&(x<150)&&(y>130)&&(y<190)&&(area_flag[4]==0))
                 {area_flag[4] = 1;}
39          else if((x>150)&&(x<210)&&(y>130)&&(y<190)&&(area_flag[5]==0))
                 {area_flag[5] = 1;}
40          else if((x>30)&&(x<90)&&(y>190)&&(y<250)&&(area_flag[6]==0))
                 {area_flag[6] = 1;}
41          else if((x>90)&&(x<150)&&(y>190)&&(y<250)&&(area_flag[7]==0))
                 {area_flag[7] = 1;}
42          else if((x>150)&&(x<210)&&(y>190)&&(y<250)&&(area_flag[8]==0))
                 {area_flag[8] = 1;}
43          repaint();
44          try{Thread.sleep(250);}
```

```
45          catch(InterruptedException e) {;}
46      }
47  }

48  public void paint(Graphics g) {
49      g.drawLine(90,50,90,230);
50      g.drawLine(150,50,150,230);
51      g.drawLine(30,110,210,110);
52      g.drawLine(30,170,210,170);

53      for (i=0; i<9; i++) {
54          if (area_flag[0] == 1)
              g.drawImage(img, 42, 60, this);
55          if (area_flag[1] == 1)
              g.drawImage(img, 102, 60, this);
56          if (area_flag[2] == 1)
                g.drawImage(img, 164, 60, this);
57          if (area_flag[3] == 1)
                g.drawImage(img, 42, 125, this);
58          if (area_flag[4] == 1)
                g.drawImage(img, 102, 125, this);
59          if (area_flag[5] == 1)
                g.drawImage(img, 164, 125, this);
60          if (area_flag[6] == 1)
                g.drawImage(img, 42, 187, this);
61          if (area_flag[7] == 1)
                g.drawImage(img, 102, 187, this);
62          if (area_flag[8] == 1)
                g.drawImage(img, 164, 187, this);
63      }
64  }
65 }
```

行 06	声明井字棋盘中 9 个棋子的标示数组。
行 15	读取棋子图案（本例为圆圈）。
行 34～42	设置每个棋格的范围，且当该范围有落子时，设置其棋格标示为 1（如 area_flag[i] = 1）。
行 49～52	绘制井字棋盘的四条线。
行 53～63	当棋格标示为 1 时，在该棋格的中心位置绘制棋子图案。

运行结果

运行结果如图 6-12 所示。

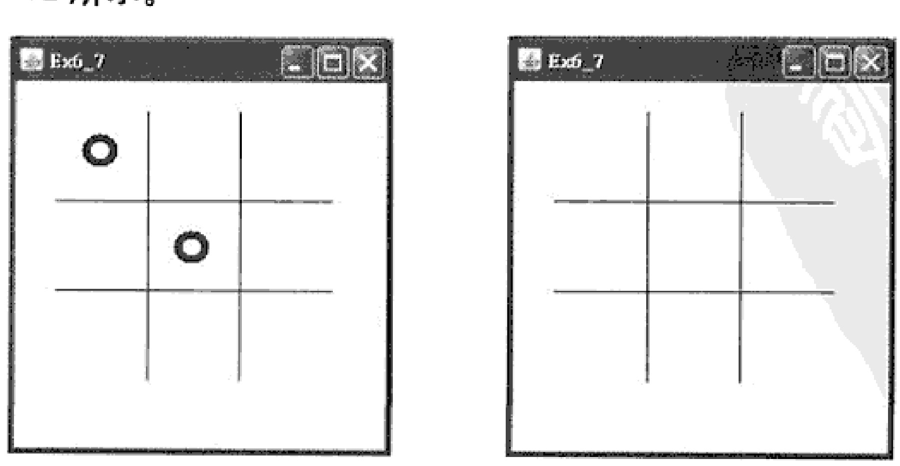


图 6-12

在棋格范围内单击鼠标左键，则在该范围预定义的中心位置上将显示棋子图案。

6-8 习题

1. 什么是鼠标事件的移动（Moved）？
2. 什么是鼠标事件的拖动（Dragged）？
3. 如何设置图像选择范围？
4. 什么是鼠标事件的选择（Selected）？
5. 什么是鼠标事件的随动（Followed）？
6. 从程序设计的角度看，弈棋游戏可分为哪两类？

Chapter 07 键盘事件应用

7-1 简介

常用的键盘事件有按键（PRESSED）和释放（RELEASED）。每次发生键盘事件时，我们可以立即读取该事件的源（Source）、标识码（ID）、发生时间（When）与按键的 ASCII 码。

由于上述信息是伴随事件的发生而产生的，所以我们可利用这些信息引导后续操作的执行，如键盘数据、图案移动方向控制等。

回顾 1–3 节，线程绘图流程是以 Thread 驱动 run()方法，再以 repaint()调用 paint()方法，这样可获得线程并行的效果。

7-2 键盘数据

回顾 5–5 节，KeyEvent 类将与按键有关的事件封装起来，击打键盘是按键事件的源（KeyEvent Source），它还实现了 KeyListener 接口，即将一个 KeyEvent 送到 KeyListenerl 中进行处理。

当生成键盘事件的对象后，我们可以使用其实例方法来读取构造函数各参数的数据。参考范例 24，当按任一键盘键时都可读取该键的 Source、ID、When 和 KeyCode。

7-3 静态方向控制

当按键盘键时，键盘事件随之发生，此时可读取按键的信息，不同的按键有不同的 KeyCode，我们可利用这些不同的信息来执行不同的操作。例如范例 41，当按“→”键时，图案向右移一步；当按“←”键时，图案向左移一步；当按“↑”键时，图案向上移一步；当按“↓”键时，图案向下移一步。

范例 41 文件 Ex7_3_1.java 的功能是解释单幅图案静态方向控制的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex7_3_1 extends Frame implements Runnable {
04    int x=150, y=235, dx, dy;
05    Image img;
```

```
06  public static void main(String args[]) {
07      Ex7_3_1 workStart=new Ex7_3_1();
08  }

09  public Ex7_3_1() {
10      super("Ex7_3_1");
11      setSize(350,350);

12      Toolkit tk = Toolkit.getDefaultToolkit();
13      img = tk.getImage("img8_3_1.JPG");

14      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
15      enableEvents(AWTEvent.KEY_EVENT_MASK);

16      setVisible(true);
17      new Thread(this).start();
18  }

19  public void processWindowEvent(WindowEvent e) {
20      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
21          System.exit(0);
22      }
23  }

24  public void processKeyEvent(KeyEvent e) {
25      if(e.getID() == KeyEvent.KEY_PRESSED) {
26          switch(e.getKeyCode()) {
27              case KeyEvent.VK_RIGHT:
28                  dx = 5; dy = 0;
29                  break;
30              case KeyEvent.VK_LEFT:
31                  dx = -5; dy = 0;
32                  break;
33              case KeyEvent.VK_UP:
34                  dx = 0; dy = -5;
35                  break;
36              case KeyEvent.VK_DOWN:
37                  dx = 0; dy = 5;
38                  break;
39          }
40          x = x + dx;
41          y = y + dy;
42      }
43  }

44  public void run() {
45      while(true) {
46          repaint();
47          try{Thread.sleep(10);}
48          catch(InterruptedException e) {;}
49      }
50  }

51  public void paint(Graphics g) {
```

```
52        g.drawImage(img, x, y, this);
53    }
54 }
```

行 15	启动键盘事件。
行 24	当有键盘事件发生时立即运行行 25～42。
行 25	如果键盘事件是按键事件，则运行行 26～39。
行 27～38	当按“→”键时，设置 dx = 5、dy = 0，使图案向右移一步； 当按“←”键时，设置 dx = −5、dy = 0，使图案向左移一步； 当按“↑”键时，设置 dx = 0、dy = −5，使图案向上移一步； 当按“↓”键时，设置 dx = 0、dy = 5，使图案向下移一步。
行 40～41	设置图案新位置坐标。
行 17	以线程驱动行 44～50 的 run()方法（参考 1–3 节）。
行 46	以 repaint()方法驱动行 51～53 的 paint()方法，进行更新绘制。

运行结果

运行结果如图 7–1 所示。

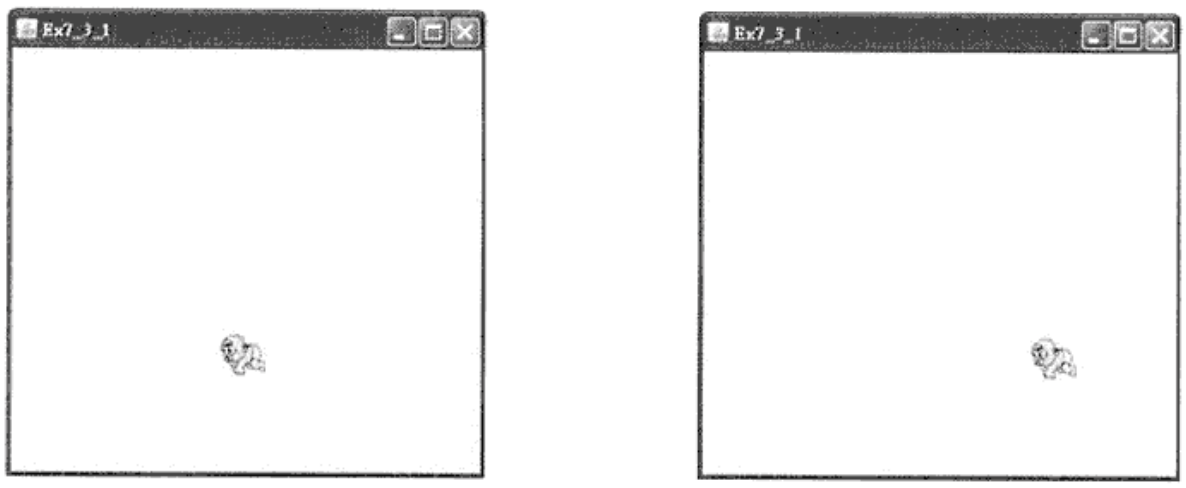


图 7-1

当按“→”键时，图案向右移一步；当按“←”键时，图案向左移一步；当按“↑”键时，图案向上移一步；当按“↓”键时，图案向下移一步。

范例 42 文件 Ex7_3_2.java 的功能是解释多幅图案静态方向控制的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex7_3_2 extends Frame implements Runnable {
04    int x=150, y=235, dx=0, dy=0, flag_direction=0;
05    Image img0, img1, img2, img3;

06    public static void main(String args[]) {
07       Ex7_3_2 workStart=new Ex7_3_2();
08    }

09    public Ex7_3_2() {
```

```
10      super("Ex7_3_2");
11      setSize(350,350);

12      Toolkit tk = Toolkit.getDefaultToolkit();
13      img0 = tk.getImage("img000.JPG");
14      img1 = tk.getImage("img090.JPG");
15      img2 = tk.getImage("img180.JPG");
16      img3 = tk.getImage("img270.JPG");

17      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
18      enableEvents(AWTEvent.KEY_EVENT_MASK);

19      setVisible(true);
20      new Thread(this).start();
21  }

22  public void processWindowEvent(WindowEvent e) {
23      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
24          System.exit(0);
25      }
26  }

27  public void processKeyEvent(KeyEvent e) {
28      if(e.getID() == KeyEvent.KEY_PRESSED) {
29          switch(e.getKeyCode()) {
30              case KeyEvent.VK_RIGHT:
31                  dx = 5; dy = 0;
32                  flag_direction = 0;
33                  break;
34              case KeyEvent.VK_LEFT:
35                  dx = -5; dy = 0;
36                  flag_direction = 2;
37                  break;
38              case KeyEvent.VK_UP:
39                  dx = 0; dy = -5;
40                  flag_direction = 1;
41                  break;
42              case KeyEvent.VK_DOWN:
43                  dx = 0; dy = 5;
44                  flag_direction = 3;
45                  break;
46          }
47          x = x + dx;
48          y = y + dy;
49      }
50  }

51  public void run() {
52      while(true) {
53          repaint();
54          try{Thread.sleep(10);}
55          catch(InterruptedException e) {;}
56      }
57  }

58  public void paint(Graphics g) {
59      if(flag_direction == 0) g.drawImage(img0, x, y, this);
60      if(flag_direction == 1) g.drawImage(img1, x, y, this);
61      if(flag_direction == 2) g.drawImage(img2, x, y, this);
62      if(flag_direction == 3) g.drawImage(img3, x, y, this);
```

```
63  }
64 }
```

行 13～16	读取 4 幅图案。
行 32	当按“→”键时，设置标示 flag_direction 为 0。
行 59	如果标示 flag_direction 为 0，则显示第一幅图案。

运行结果

运行结果如图 7-2 所示。

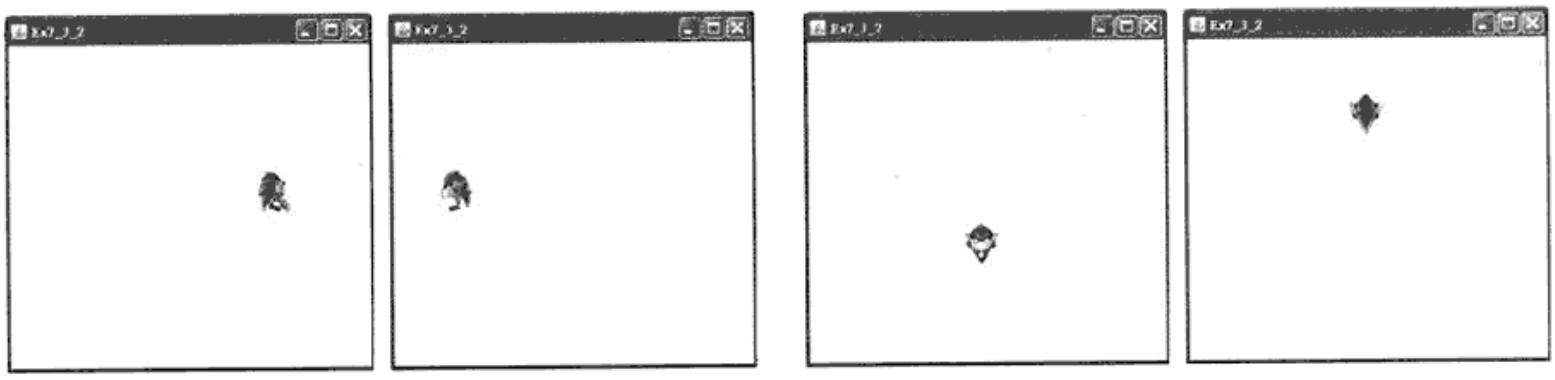


图 7-2

当按→键时，使用第一幅图，图案向右移一步；当按“←”键时，使用第三幅图，图案向左移一步；当按“↑”键时，使用第二幅图，图案向上移一步；当按“↓”键时，使用第四幅图，图案向下移一步。

7-4 动态方向控制

所谓动态，是指通过循环使图案的位置坐标不断地变换，每循环一次，即更新图案坐标一次，图案也随着不断地移动。

范例 43 文件 Ex7_4_1.java 的功能是解释单幅图案动态方向控制的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex7_4_1 extends Frame implements Runnable {
04   int x=0, y=100;
05   int dx=0, dy=0;
06   Image img;

07   public static void main(String args[]) {
08     Ex7_4_1 workStart=new Ex7_4_1();
09   }

10   public Ex7_4_1() {
11     super("Ex7_4_1");
12     setSize(350, 350);

13     Toolkit tk = Toolkit.getDefaultToolkit();
14     img = tk.getImage("fly.gif");

15     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
```

```
16      enableEvents(AWTEvent.KEY_EVENT_MASK);

17      setVisible(true);
18      new Thread(this).start();
19  }

20  public void processWindowEvent(WindowEvent e) {
21      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
22          System.exit(0);
23      }
24  }

25  public void processKeyEvent(KeyEvent e) {
26      if(e.getID() == KeyEvent.KEY_PRESSED) {
27          switch(e.getKeyCode()) {
28              case KeyEvent.VK_RIGHT:
29                  dx = 5; dy = 0;
30                  break;
31              case KeyEvent.VK_LEFT:
32                  dx = -5; dy = 0;
33                  break;
34              case KeyEvent.VK_UP:
35                  dx = 0; dy = -5;
36                  break;
37              case KeyEvent.VK_DOWN:
38                  dx = 0; dy = 5;
39                  break;
40          }
41      }
42  }

43  public void run() {
44      while(true) {
45          x = x + dx;
46          y = y + dy;
47          repaint();

48          if(x<=0) dx = 5;
49          else if((x + 50) >= getWidth()) dx = -5;

50          if(y<=0) dy = 5;
51          else if((y + 50) >= getHeight()) dy = -5;

52          try{Thread.sleep(250);}
53          catch(InterruptedException e) {;}
54      }
55  }

56  public void paint(Graphics g) {
57      g.drawImage(img, x, y, this);
58  }
59 }
```

行 44～55　随着循环的执行，图案的位置坐标会不断地变换。

行 45～46　每循环一次，更新图案坐标一次。

行 48～49　当图案触碰到左右边线时立即回头移动。

行 50～51　当图案触碰到上下边线时立即回头移动。

运行结果

运行结果如图 7–3 所示。

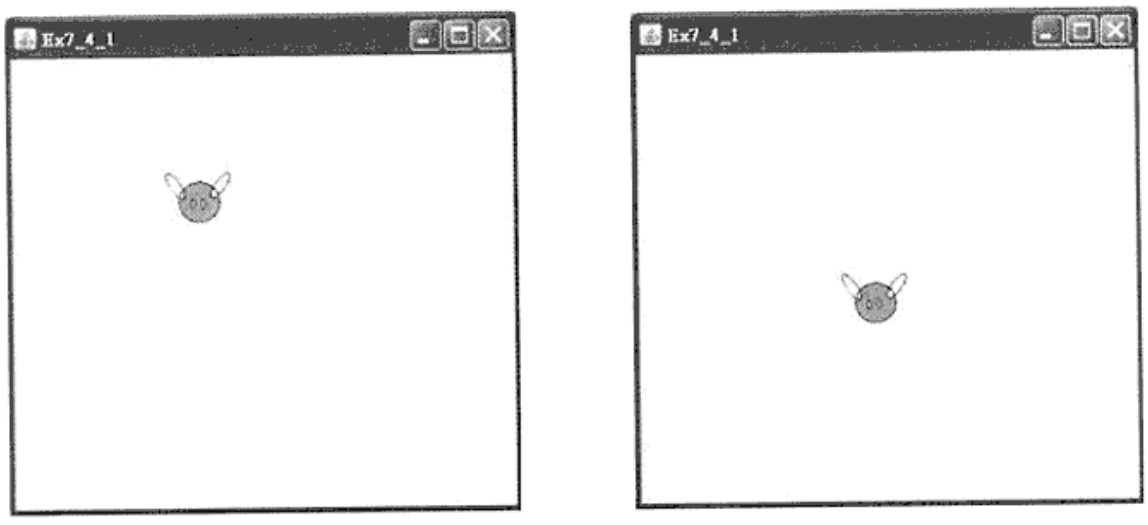


图 7-3

当按“→”键时，图案向右移动，触碰到右边线时立即回头向左移动；当按“←”键时，图案向左移动，触碰到左边线时立即回头向右移动；当按“↑”键时，图案向上移动，触碰到上边线时，立即回头向下移动；当按“↓”键时，图案向下移动，触碰到下边线时立即回头向上移动。

范例 44 文件 Ex7_4_2.java 的功能是解释多幅图案动态方向控制的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 public class Ex7_4_2 extends Frame implements Runnable {
04   int x=0, y=170, dx=0, dy=0;
05   int num=1, flag;
06   Image img0, img1, img2;

07   public static void main(String args[]) {
08     Ex7_4_2 workStart=new Ex7_4_2();
09   }

10   public Ex7_4_2() {
11     super("Ex7_4_2");
12     setSize(350, 350);

13     Toolkit tk = Toolkit.getDefaultToolkit();
14     img0 = tk.getImage("fly0.gif");
15     img1 = tk.getImage("fly1.gif");
16     img2 = tk.getImage("fly2.gif");

17     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
18     enableEvents(AWTEvent.KEY_EVENT_MASK);

19     setVisible(true);
20     new Thread(this).start();
21   }

22   public void processWindowEvent(WindowEvent e) {
23     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
24         System.exit(0);
25     }
```

```
26  }

27  public void processKeyEvent(KeyEvent e) {
28    if(e.getID() == KeyEvent.KEY_PRESSED) {
29      switch(e.getKeyCode()) {
30          case KeyEvent.VK_RIGHT:
31              dx = 5; dy = 0;
32              break;
33          case KeyEvent.VK_LEFT:
34              dx = -5; dy = 0;
35              break;
36          case KeyEvent.VK_UP:
37              dx = 0; dy = -5;
38              break;
39          case KeyEvent.VK_DOWN:
40              dx = 0; dy = 5;
41              break;
42      }
43    }
44  }

45  public void run() {
46    while(true) {
47        x = x + dx;
48        y = y + dy;
49        flag = num % 3;
50        repaint();
51        num = num + 1;

52        if(x <= 0) dx = 5;
53        else if((x+60) >= 350) dx = -5;
54        if((y-10) <= 0) dy = 5;
55        else if((y+50) >= 350) dy = -5;

56        try{Thread.sleep(170);}
57        catch(InterruptedException e) {;}
58    }
59  }

60  public void paint(Graphics g) {
61    if(flag == 0)   g.drawImage(img0, x, y, this);
62    if(flag == 1)   g.drawImage(img1, x, y, this);
63    if(flag == 2)   g.drawImage(img2, x, y, this);
64  }
65 }
```

行 14～16　读取多幅图案。

行 49　计算 num 除以 3 的余数，用于设置 flag。

行 61～63　根据 flag 的值轮流显示三幅图案，使其移动得生动。

运行结果

运行结果如图 7-4 所示。

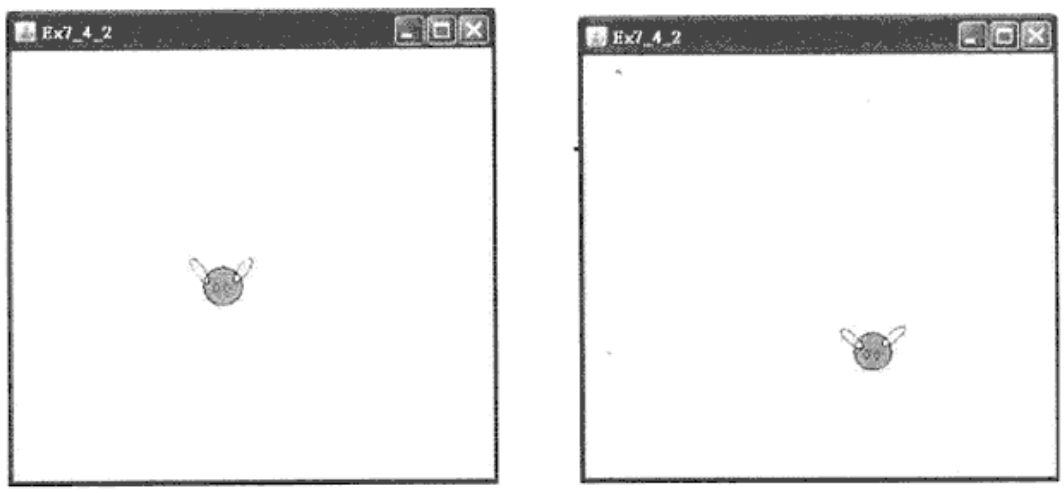


图 7-4

如同范例 43，当按“→”键时，图案向右移动，触碰到右边线时立即回头向左移动；当按“←”键时，图案向左移动，触碰到左边线时立即回头向右移动；当按“↑”键时，图案向上移动，触碰到上边线时立即回头向下移动；当按“↓”键时，图案向下移动，触碰到下边线时立即回头向上移动。

7-5 基础射击

如果能设计弈棋游戏与射击游戏，则可设计大部分的计算机游戏。我们已在 6–7 节中简单介绍了弈棋游戏的设计方法，本节将介绍基础射击的设计方法。这些方法将用于本书第 3 部分“在线游戏”的设计。

回顾 1–3 节，可使多个图像同时并行移动，在 6–6 节的范例 38 和范例 39 中实现了多个图像同时并行，各自移动，互不干扰。

本节将通过实例介绍基础射击的设计，发射器与子弹同时并行，互不干扰的实现方法。本书的重点是介绍入门程序，为了让读者清楚简单的设计概念，本书仅介绍单发子弹射击。至于多发子弹射击，笔者将另编于本系列丛书高级动画游戏中，敬请指教。

设计范例 45，当按“→”键时，发射器向右移一步；当按“←”键时，发射器向左移一步；当按“↑”键时，发射器向上移一步；当按“↓”键时，发射器向下移一步。按 Space 键时，发射一发子弹。

范例 45 设计文件 Ex7_5.java，其功能是解释基础射击的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex7_5 extends Frame implements Runnable {
04   int x=150, y=235, dx, dy;
05   int bx, by, dbx=0, dby=-5, bflag=0;
06   Image img;

07   public static void main(String args[]) {
08     Ex7_5 workStart=new Ex7_5();
09   }

10   public Ex7_5() {
11     super("Ex7_5");
12     setSize(350,350);
```

```
13      Toolkit tk = Toolkit.getDefaultToolkit();
14      img = tk.getImage("car090.gif");

15      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
16      enableEvents(AWTEvent.KEY_EVENT_MASK);

17      setVisible(true);
18      new Thread(this).start();
19  }

20  public void processWindowEvent(WindowEvent e) {
21      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
22          System.exit(0);
23      }
24  }

25  public void processKeyEvent(KeyEvent e) {
26      if(e.getID() == KeyEvent.KEY_PRESSED) {
27          switch(e.getKeyCode()) {
28              case KeyEvent.VK_RIGHT:
29                  dx = 5; dy = 0;
30                  break;
31              case KeyEvent.VK_LEFT:
32                  dx = -5; dy = 0;
33                  break;
34              case KeyEvent.VK_UP:
35                  dx = 0; dy = -5;
36                  break;
37              case KeyEvent.VK_DOWN:
38                  dx = 0; dy = 5;
39                  break;
40              case KeyEvent.VK_SPACE:
41                  dx = 0; dy = 0;
42                  bx = x + 30;
43                  by = y - 5;
44                  bflag = 1;
45                  break;
46              default:
47                  dx = 0; dy = 0;
48          }
49          x = x + dx;
50          y = y + dy;
51      }
52  }

53  public void run() {
54      while(true) {
55          if(by <= 0) bflag = 0;
56          if(bflag == 1) by = by + dby;

57          repaint();
58          try{Thread.sleep(30);}
59          catch(InterruptedException e) {;}
60      }
61  }

62  public void paint(Graphics g) {
63      g.drawImage(img, x, y, this);
64      g.fillRect(bx, by, 3, 5);
```

```
65  }
66 }
```

行 05	声明子弹的坐标变量和标示变量 bflag。
行 40～45	当按 Space 键时，设置子弹的起始坐标，设置 bflag 为 1。
行 55	当子弹越过框架上边线时，设置 bflag 为 0。
行 56	当 bflag 为 1 时，设置子弹坐标（即按 Space 键且子弹未越过框架上边线时）。
行 64	当按 Space 键且子弹未越过框架上边线时，根据行 56 设置的子弹坐标绘制子弹。

运行结果

运行结果如图 7–5 所示。

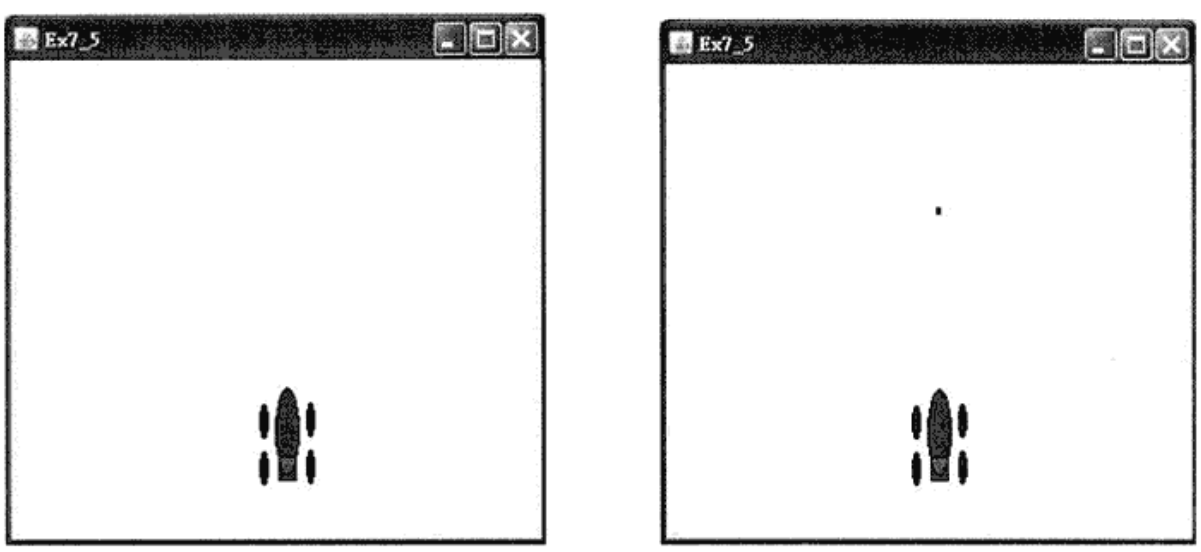


图 7-5

7-6 习题

1. 如何读取键盘事件的信息？
2. 如何以键盘键控制图案的移动方向？
3. 如何使图案动态移动？
4. 如何设计发射器与子弹，使它们并行，且互不干扰？

Chapter 08 消除图像闪烁

8-1 简介

细心的读者可能已经注意到，上述各章的图像动画都有闪烁漂浮的情况，这是因为新的图像随时替换旧的图像，如此才能使图像在窗口中显示出位置的变化或姿态的变化。如果图像替换得太快，其速度超过了计算机系统的负荷，有些部分先到位，有些部分后到位，则图像将会显得闪烁而不稳定，这也是计算机游戏设计不希望看到的。本章将介绍如何消除这样的闪烁。

8-2 设计方法

为了消除图像闪烁，我们可设置一个缓冲页（Buffer Page），将新图像先置入缓冲页，等图像各部分都绘制完成后，再将缓冲页显示在窗口中，这样就可消除图像的闪烁。程序设计过程中我们将使用 Image、Component、Graphics 等类的方法。

8-2-1 创建缓冲页与 Image 类

java.awt.Image 继承自 Object，此类对象可使图像驻留在文件或缓冲区中，与创建缓冲页有关的实例方法有：

```
public abstract Graphics getGraphics();
```

读取当前正在绘制的图像，用于复制到内存中（故可用于复制到缓冲页）。

```
public abstract Graphics getInsets();
```

读取缓冲页的范围，通常以 getInsets().left 读取其左端的 x 轴坐标；以 getInsets().right 读取其右端的 x 轴坐标；以 getInsets().top 读取其上端的 y 轴坐标；以 getInsets().bottom 读取其下端的 y 轴坐标。

在处理文件图像与缓冲区图像上，由于驻留环境不同，所以两者的处理过程也有所不同。

1 文件图像：如 3–3 节，我们使用 Toolkit 类，借助其类方法 getDefaultToolkit() 创建新对象，再以实例方法 getImage()读取文件图像。如范例 16：

```
Image image;
Toolkit tk = Toolkit.getDefaultToolkit();
image = tk.getImage("Sunset.jpg");
```

其中，Sunset.jpg 是文件图像。

2 缓冲区图像：我们使用 Image 类，借助其实例方法 getGraphics()在缓冲存储器中读取图像数据。例如：

```
Image bufferPage;                    （声明图像缓冲页对象）
Graphics bufferg;                    （声明图像对象）
bufferPage = createImage(w, h);      （创建图像缓冲页对象，createImage(w, h) 参考 8-2-2 节）
bufferg = bufferPage.getGraphics();  （读取当前的图像）
```

8-2-2 创建缓冲页与 Component 类

java.awt.Component 继承自 Object，此类对象的方法可直接或间接地支持其他抽象类的运行，在此用于创建缓冲页的实例方法有：

public Image **createImage(int width, int height);**

生成一个图像对象，用于创建一个放置图像的缓冲区，但并不显示在屏幕上。

public void **paint(Graphics g);**

以参数对象 g 绘制图案或显示图像。

public void **repaint();**

调用 paint(Graphics g)进行重绘。

public void **update(Graphics g);**

调用 paint(Graphics g)进行重绘，通常与 repaint()并行运行。

8-2-3 创建缓冲页与 Graphics 类

在 2—2 节和 3—4 节中，我们曾详述 Graphics 类的绘图功能，除了基础绘图、图像图案绘制之外，Graphics 类也支持缓冲页的创建。其相关实例方法有：

public abstract void **drawImage(Image img, int x, int y, this);**

其中，参数 img 是图像；(x，y)是图像对应长方形框左上角的坐标；this 是在代码块内对图像设置内容的关键字。

public abstract void **dispose();**

立即释放与该 Graphics 对象有关的所有资源。

8-2-4 创建缓冲页的设计方法

经过上述有关 Image 类、Component 类与 Graphics 类的讨论，我们可创建一组消除图像闪烁的程序代码。

```
Image bufferPage = null;                    （声明缓冲页）
public void run() {repaint();}              （调用 paint()方法）
public void update(Graphics g) {paint(g);}  （调用 paint()方法，参考 8-2-2 节）
public void paint(Graphics g) {
    Graphics bufferg;
    if(bufferpage == null)
        bufferpage = createImage(w, h);     （创建缓冲页，参考 8-2-2 节）
    bufferg = bufferPage.getGraphics();     （启动缓冲页绘制图像机制，见 8-2-1 节）
```

```
        bufferg.drawImage(img, x, y, this);          （在缓冲页中绘制图像，见 8-2-3 节）
        bufferg.dispose();                           （当完成缓冲页全图绘制后，释放有关资源，见 8-2-3 节）
        g.drawImage(bufferPage, getInsets().left,
        getInsets().top, this);          （在屏幕上显示缓冲页中的图像，见 8-2-1 节）
    }
```

8-3 消除动画闪烁

为了消除图像闪烁，我们可设置一个缓冲页（Buffer Page），将新图像先置入缓冲页，等图像各部分都绘制完成后，再将缓冲页显示在窗口中，这样就可消除图像的闪烁。根据 8—2—4 节中消除图像闪烁的程序代码，设计范例 46，消除单幅动画闪烁的现象。

范例 46 参考范例 18.1，设计文件 Ex8_3_1.java，其功能是解释消除单幅动画闪烁的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex8_3_1 extends Frame implements Runnable {
04      int x=0, y=100;
05      int dx=5, dy=5;
06      Image img, bufferPage=null;

07      public static void main(String args[]) {
08        Ex8_3_1 workStart=new Ex8_3_1();
09      }

10      public Ex8_3_1() {
11        super("Ex8_3_1");
12        setSize(350, 350);

13        Toolkit tk = Toolkit.getDefaultToolkit();
14        img = tk.getImage("fly.gif");

15        enableEvents(AWTEvent.WINDOW_EVENT_MASK);

16        setVisible(true);
17        new Thread(this).start();
18    }

19    public void processWindowEvent(WindowEvent e) {
20          if(e.getID() == WindowEvent.WINDOW_CLOSING) {
21              System.exit(0);
22            }
23    }

24    public void run() {
25      while(true) {
26          x = x + dx;
27          y = y + dy;
28          repaint();

29          if(x<=0) dx = 5;
30          else if((x + 50) >= getWidth()) dx = -5;

31          if(y<=0) dy = 5;
32          else if((y + 50) >= getHeight()) dy = -5;

33          try{Thread.sleep(250);}
```

```
34            catch(InterruptedException e) {;}
35         }
36     }

37     public void update(Graphics g) {
38         paint(g);
39     }

40     public void paint(Graphics g) {
41         Graphics bufferg;
42         if(bufferPage == null)
43             bufferPage = createImage(350, 350);
44         bufferg =bufferPage.getGraphics();

45         bufferg.drawImage(img, x, y, this);

46         bufferg.dispose();
47         g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
48     }
49 }
```

行 42～43	创建缓冲页。
行 44	启动缓冲页绘制图像机制。
行 45	在缓冲页中绘制图像。
行 46	当完成缓冲页全图绘制后，释放有关资源。
行 47	在屏幕上显示缓冲页中的图像。

运行结果

运行结果如图 8-1 所示。

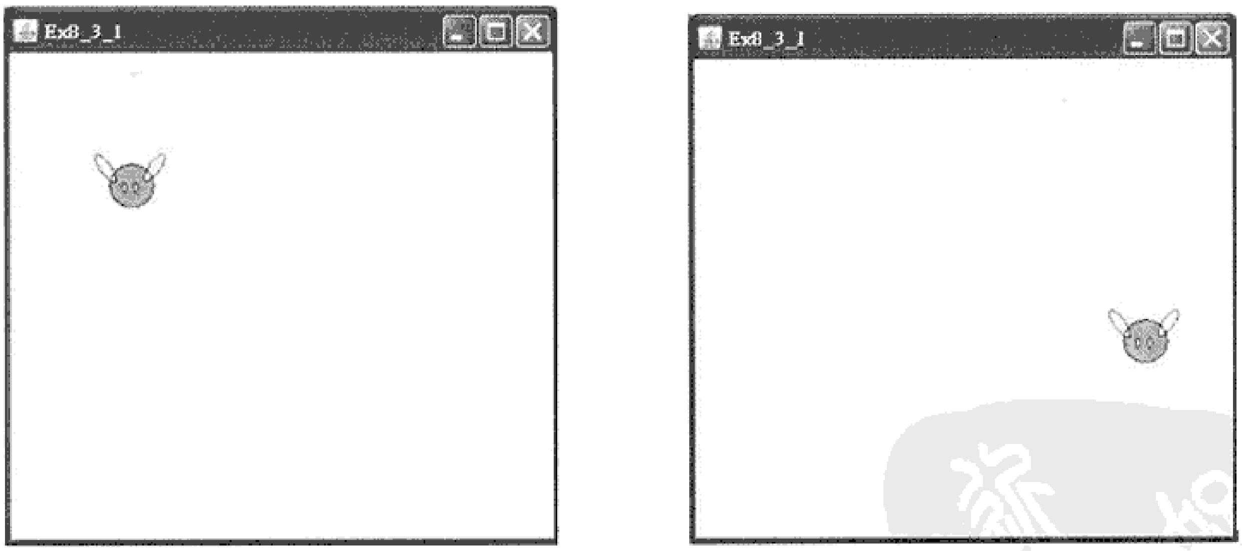


图 8-1

讨论事项

比较范例 18.1，本例中将不会再出现闪烁的图像。

根据 8-2-4 节中消除图像闪烁的程序代码，设计范例 47，消除多幅动画闪烁的现象。

范例 47 参考范例 19，文件 Ex8_3_2.java，其功能是解释消除多幅动画闪烁的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex8_3_2 extends Frame implements Runnable {
04      int num=0, flag;
05      int x=0, y=100, dx=5, dy=5;
06      Image img0, img1, img2, bufferPage = null;

07      public static void main(String args[]) {
08         Ex8_3_2 workStart=new Ex8_3_2();
09      }

10      public Ex8_3_2() {
11           super("Ex8_3_2");
12           setSize(350, 350);

13           Toolkit tk = Toolkit.getDefaultToolkit();
14           img0 = tk.getImage("fly0.gif");
15           img1 = tk.getImage("fly1.gif");
16           img2 = tk.getImage("fly2.gif");

17           enableEvents(AWTEvent.WINDOW_EVENT_MASK);

18           setVisible(true);
19           new Thread(this).start();
20      }

21      public void processWindowEvent(WindowEvent e) {
22           if(e.getID() == WindowEvent.WINDOW_CLOSING) {
23                System.exit(0);
24              }
25      }

26    public void run() {
27       while(true) {
28            x = x + dx;
29            y = y + dy;
30            flag = num % 3;
31            repaint();
32            num = num + 1;

33            if(x <= 0) dx = 5;
34            else if((x+60) >= 350) dx = -5;

35            if(y<=0) dy = 5;
36            else if((y + 50) >= getHeight()) dy = -5;

37            try{Thread.sleep(250);}
38            catch(InterruptedException e) {;}
39       }
40    }

41   public void update(Graphics g) {
42        paint(g);
43   }

44    public void paint(Graphics g) {
45        Graphics bufferg;
46        if(bufferPage == null)
47             bufferPage = createImage(350, 350);
```

```
48          bufferg = bufferPage.getGraphics();

49          if(flag == 0)
50              bufferg.drawImage(img0, x, y, this);
51          else if(flag == 1)
52              bufferg.drawImage(img1, x, y, this);
53          else if(flag == 2)
54              bufferg.drawImage(img2, x, y, this);

55          bufferg.dispose();
56          g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
57      }
58 }
```

行 46～47　创建缓冲页。
行 48　启动缓冲页绘制图像机制。
行 49～54　在缓冲页中轮流绘制多幅图像。
行 55　当完成缓冲页全图绘制后，释放有关资源。
行 56　在屏幕上显示缓冲页中的图像。

运行结果

运行结果如图 8–2 所示。

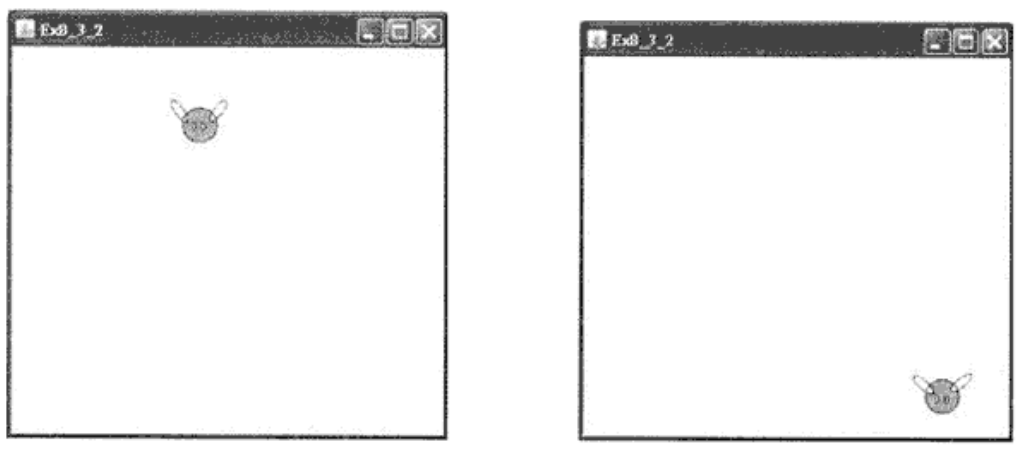


图 8-2

讨论事项

比较范例 19，本例中将不会再出现闪烁的图像。

8-4 消除棋盘闪烁

根据 8–2–4 节中消除图像闪烁的程序代码，设计范例 48，消除棋盘闪烁的现象，包括棋盘网格线与棋子图片。

范例 48　参考范例 40，设计文件 Ex8_4.java，其功能是解释消除棋盘闪烁的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;

03 public class Ex8_4 extends Frame implements Runnable {
```

```
04  int x, y;
05  Image img, bufferPage;
06  int[] area_flag = new int[9];
07  int i;

08  public static void main(String args[]) {
09     Ex8_4 workStart=new Ex8_4();
10  }

11  public Ex8_4() {
12     super("Ex8_4");
13     setSize(250,280);

14     Toolkit tk = Toolkit.getDefaultToolkit();
15     img = tk.getImage("Circle.GIF");

16     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
17     enableEvents(AWTEvent.MOUSE_EVENT_MASK);

18     setVisible(true);
19     new Thread(this).start();
20  }

21  public void processWindowEvent(WindowEvent e) {
22     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
23        System.exit(0);
24     }
25  }

26  public void processMouseEvent(MouseEvent e) {
27     if(e.getID() == MouseEvent.MOUSE_PRESSED) {
28        x = e.getX();
29        y = e.getY();
30     }
31  }

32  public void run() {
33     while(true) {
34        if((x>30)&&(x<90)&&(y>70)&&(y<130)&&(area_flag[0]==0))
                {area_flag[0] = 1;}
35        else if ((x>90)&&(x<150)&&(y>70)&&(y<130)&&(area_flag[1]==0))
                {area_flag[1] = 1;}
36        else if ((x>150)&&(x<210)&&(y>70)&&(y<130)&&(area_flag[2]==0))
                {area_flag[2] = 1;}
37        else if((x>30)&&(x<90)&&(y>130)&&(y<190)&&(area_flag[3]==0))
                {area_flag[3] = 1;}
38        else if((x>90)&&(x<150)&&(y>130)&&(y<190)&&(area_flag[4]==0))
                {area_flag[4] = 1;}
39        else if((x>150)&&(x<210)&&(y>130)&&(y<190)&&(area_flag[5]==0))
                {area_flag[5] = 1;}
40        else if((x>30)&&(x<90)&&(y>190)&&(y<250)&&(area_flag[6]==0))
                {area_flag[6] = 1;}
41        else if((x>90)&&(x<150)&&(y>190)&&(y<250)&&(area_flag[7]==0))
                {area_flag[7] = 1;}
42        else if((x>150)&&(x<210)&&(y>190)&&(y<250)&&(area_flag[8]==0))
                {area_flag[8] = 1;}
43        repaint();
44        try{Thread.sleep(250);}
45        catch(InterruptedException e) {;}
```

```
46    }
47  }

48  public void update(Graphics g) {
49      paint(g);
50  }

51  public void paint(Graphics g) {
52      Graphics bufferg;
53      if(bufferPage == null)
            bufferPage = createImage(250, 250);
54      bufferg = bufferPage.getGraphics();

55      bufferg.drawLine(90,50,90,230);
56      bufferg.drawLine(150,50,150,230);
57      bufferg.drawLine(30,110,210,110);
58      bufferg.drawLine(30,170,210,170);

59      for (i=0; i<9; i++) {
60          if (area_flag[0] == 1)
              bufferg.drawImage(img, 42, 60, this);
61          if (area_flag[1] == 1)
              bufferg.drawImage(img, 102, 60, this);
62          if (area_flag[2] == 1)
              bufferg.drawImage(img, 164, 60, this);
63          if (area_flag[3] == 1)
              bufferg.drawImage(img, 42, 125, this);
64          if (area_flag[4] == 1)
              bufferg.drawImage(img, 102, 125, this);
65          if (area_flag[5] == 1)
              bufferg.drawImage(img, 164, 125, this);
66          if (area_flag[6] == 1)
              bufferg.drawImage(img, 42, 187, this);
67          if (area_flag[7] == 1)
              bufferg.drawImage(img, 102, 187, this);
68          if (area_flag[8] == 1)
              bufferg.drawImage(img, 164, 187, this);
69      }

70      bufferg.dispose();
71      g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
72    }
73  }
```

行 53	创建缓冲页。
行 54	启动缓冲页绘制图像机制。
行 55～58	在缓冲页中绘制棋盘上的四条线。
行 59～69	在缓冲页中绘制棋子图片。
行 70	当完成缓冲页全图绘制后，释放有关资源。
行 71	在屏幕上显示缓冲页中的图像。

运行结果

运行结果如图 8–3 所示。

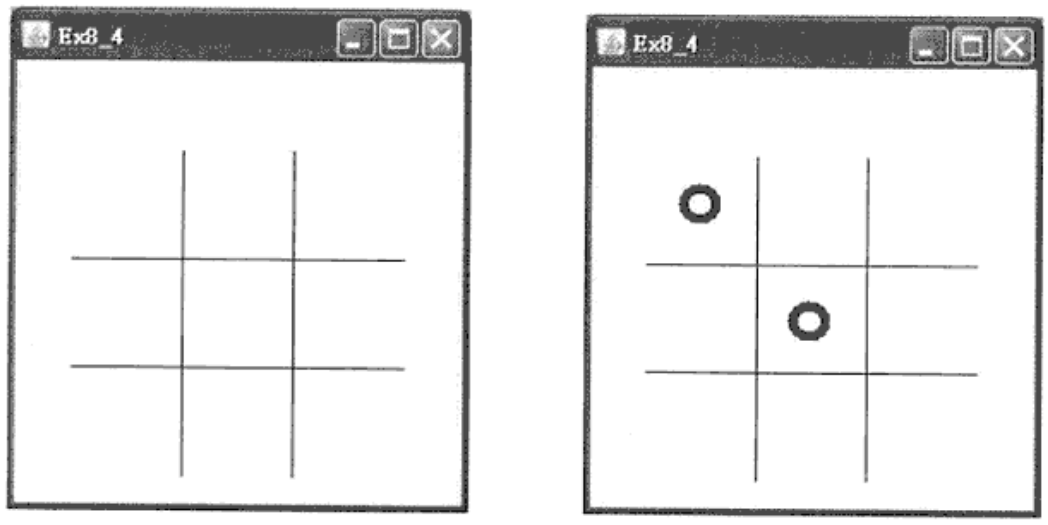


图 8-3

讨论事项

比较范例 40，本例中将不会再出现闪烁的图像。

8-5 消除射击图像闪烁

根据 8-2-4 节中消除图像闪烁的程序代码，设计范例 49，消除射击图像闪烁的现象，包括发射器与子弹。

范例 49 参考范例 45，设计文件 Ex8_5.java，其功能是解释消除射击图像闪烁的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 public class Ex8_5 extends Frame implements Runnable {
04   int x=150, y=235, dx, dy;
05   int bx, by, dbx=0, dby=-5, bflag=0;
06   Image img, bufferPage=null;

07   public static void main(String args[]) {
08     Ex8_5 workStart=new Ex8_5();
09   }

10   public Ex8_5() {
11     super("Ex8_5");
12     setSize(350,350);

13     Toolkit tk = Toolkit.getDefaultToolkit();
14     img = tk.getImage("car090.gif");

15     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
16     enableEvents(AWTEvent.KEY_EVENT_MASK);

17     setVisible(true);
18     new Thread(this).start();
19   }

20   public void processWindowEvent(WindowEvent e) {
21     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
22       System.exit(0);
```

```
    }
}

public void processKeyEvent(KeyEvent e) {
  if(e.getID() == KeyEvent.KEY_PRESSED) {
    switch(e.getKeyCode()) {
         case KeyEvent.VK_RIGHT:
              dx = 5; dy = 0;
              break;
         case KeyEvent.VK_LEFT:
              dx = -5; dy = 0;
              break;
         case KeyEvent.VK_UP:
              dx = 0; dy = -5;
              break;
         case KeyEvent.VK_DOWN:
              dx = 0; dy = 5;
              break;
         case KeyEvent.VK_SPACE:
              dx = 0; dy = 0;
              bx = x + 30;
              by = y - 5;
              bflag = 1;
              break;
          default:
              dx =0; dy = 0;
    }
    x = x + dx;
    y = y + dy;
  }
}

public void run() {
  while(true) {
    if(by <= -10) bflag = 0;
    if(bflag == 1) by = by + dby;

    repaint();
    try{Thread.sleep(30);}
    catch(InterruptedException e) {;}
  }
}

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
  Graphics bufferg;
  if(bufferPage == null)
      bufferPage = createImage(350, 350);
  bufferg = bufferPage.getGraphics();

  bufferg.drawImage(img, x, y, this);
  bufferg.fillRect(bx, by, 3, 5);
  bufferg.setColor(Color.white);
  bufferg.fillRect(bx, by+5, 3, 5);

  bufferg.dispose();
  g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
```

```
75  }
76 }
```

行 67	创建缓冲页。
行 68	创建缓冲页绘制图像对象。
行 69	在缓冲页中绘制发射器。
行 70～72	在缓冲页中绘制子弹图片。因为是在缓冲页中绘制，要等到全图完成，所以会造成一条直线，而不是一颗子弹，因此以行 69～70 修饰。
行 73	当完成缓冲页全图绘制后，释放有关资源。
行 74	在屏幕上显示缓冲页中的图像。

运行结果

运行结果如图 8–4 所示。

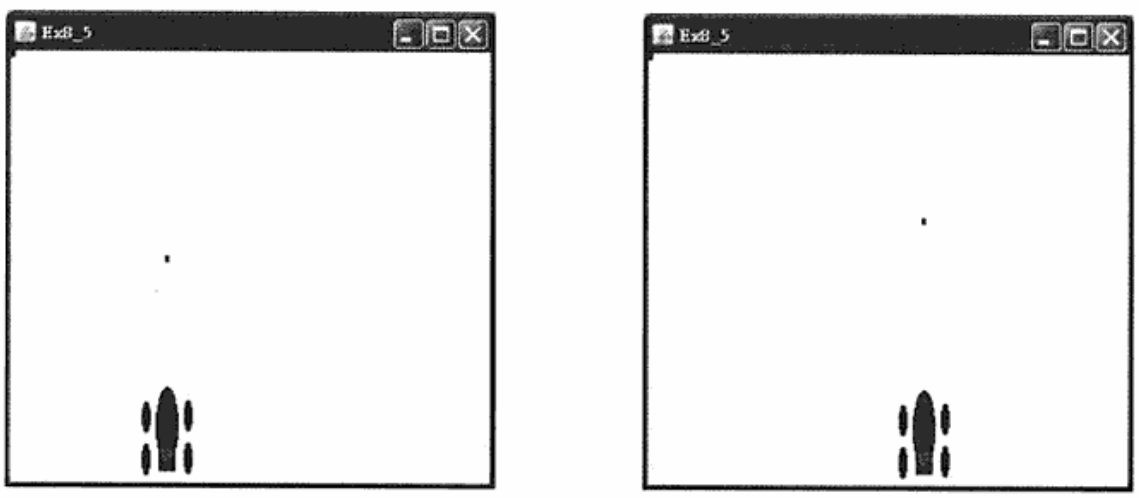


图 8-4

讨论事项

比较范例 45，本例中将不会再出现闪烁的图像。

8-6 习题

1. 为何会产生图像闪烁的现象？
2. 如何消除图像闪烁现象？
3. 消除图像闪烁现象的程序应如何设计？

Chapter 09 音效处理

9-1 简介

如果动画与电玩没有音效的陪衬，将使画面失色许多，Sun 公司在设计 Applet 时已支持音效的使用，但在当时并不适用于窗口框架。

本书在第 1 章开宗明义地提到，动画游戏有文字、有图像、有动画，这些都需要一个环境来显示，常用的显示环境是窗口框架（Frame）与浏览器（Browser），前者可用于单机显示或多机网络对阵；后者可用于单机网络显示。

为了使窗口框架具有适当的音响效果，Sun 公司特别将音效由 Applet 扩展到窗口框架系列。

9-2 音效设计方法

Java 提供 AIFF、AU 和 WAV 三种格式的声音文件，本书暂时采用 AU 格式，因为它比较简单好用。

AU 格式原本是为 Applet 设计的，并不适用于窗口框架。因为它使用方便，所以 Sun 公司又将它扩展到窗口框架，但仍在 Applet 范畴内。故在使用前须导入 applet 包：

```
import java.applet.*;
```

并以其中 AudioClip 类声明：

```
AudioClip   sound;
```

其中，sound 是音效变量，用于读取声音文件：

```
sound = Applet.newAudioClip(getClass().getResource("sound.au"));
```

其中，Applet.newAudioClip()是专为窗口框架设计的音效接口，sound.au 是任一 AU 格式的声音文件。可以使用

```
sound.play( );
sound.loop( );
sound.stop( );
```

分别执行播放、连续播放与停止播放等操作。

9-3 背景音效

如 9—2 节所述，以 Applet.newAudioClip(getClass().getResource("sound.au"))读取声音文件 sound.au，设计范例 50，以 sound.loop( ) 连续播放声音文件，从而形成背景音乐。

范例 50 参考范例 47，设计文件 Ex9_3.java，其功能是解释动画背景音效的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.applet.*;

04 public class Ex9_3 extends Frame implements Runnable {
05   int num=0, flag;
06   int x=0, y=100, dx=5, dy=5;
07   Image img0, img1, img2, bufferPage = null;
08   AudioClip sound;

09   public static void main(String args[]) {
10     Ex9_3 workStart=new Ex9_3();
11   }

12   public Ex9_3() {
13     super("Ex9_3");
14     setSize(350, 350);

15     Toolkit tk = Toolkit.getDefaultToolkit();
16     img0 = tk.getImage("fly0.gif");
17     img1 = tk.getImage("fly1.gif");
18     img2 = tk.getImage("fly2.gif");

19     sound = Applet.newAudioClip(getClass().getResource("sound.au"));
20     sound.loop();

21     enableEvents(AWTEvent.WINDOW_EVENT_MASK);

22     setVisible(true);
23     new Thread(this).start();
24   }

25   public void processWindowEvent(WindowEvent e) {
26     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
27       System.exit(0);
28     }
29   }

30   public void run() {
31     while(true) {
32       x = x + dx;
33       y = y + dy;
34       flag = num % 3;
35       repaint();
36       num = num + 1;

37       if(x <= 0) dx = 5;
38       else if((x+60) >= 350) dx = -5;

39       if(y<=0) dy = 5;
40       else if((y + 50) >= getHeight()) dy = -5;
```

```
41          try{Thread.sleep(250);}
42          catch(InterruptedException e) {;}
43      }
44  }

45  public void update(Graphics g) {
46      paint(g);
47  }

48  public void paint(Graphics g) {
49      Graphics bufferg;
50      if(bufferPage == null)
51          bufferPage = createImage(350, 350);
52      bufferg = bufferPage.getGraphics();

53      if(flag == 0)
54          bufferg.drawImage(img0, x, y, this);
55      else if(flag == 1)
56          bufferg.drawImage(img1, x, y, this);
57      else if(flag == 2)
58          bufferg.drawImage(img2, x, y, this);

59      bufferg.dispose();
60      g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
61  }
62 }
```

行 03　由系统导入 java.applet 包。
行 08　声明音效变量。
行 19　读取声音文件。
行 20　连续播放声音文件。

运行结果

运行结果如图 9–1 所示。

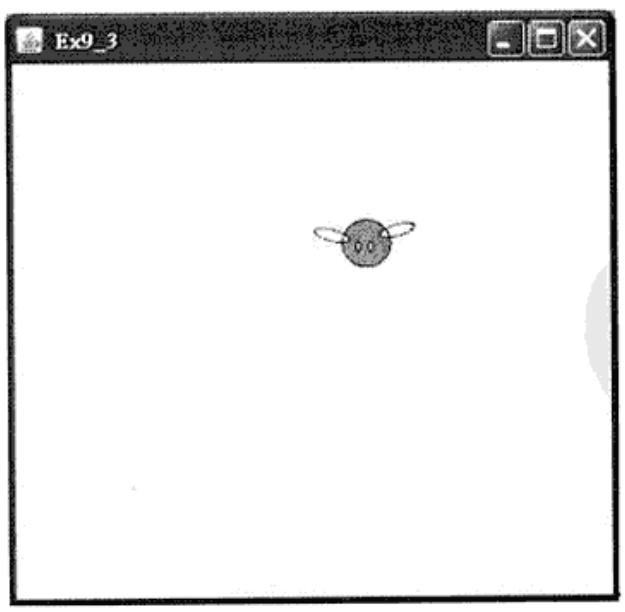


图 9-1

当打开窗口时，音乐伴随播放。

9-4 音效控制

如 9-2 节所述，以 Applet.newAudioClip(getClass().getResource("sound.au"))读取声音文件 sound.au，设计范例 51，以键盘控制音乐的播放，当按 L 键时进行连续播放（loop）方法；当按 P 键时播放一次（play）方法；当按 S 键时停止播放（stop）方法。

范例 51 参考范例 50，文件 Ex9_4.java 的功能是解释动画音效控制的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.applet.*;

04 public class Ex9_4 extends Frame implements Runnable {
05    int num=0, flag;
06    int x=0, y=100, dx=5, dy=5;
07    Image img0, img1, img2, bufferPage = null;
08    AudioClip sound;

09    public static void main(String args[]) {
10       Ex9_4 workStart=new Ex9_4();
11    }

12    public Ex9_4() {
13       super("Ex9_4");
14       setSize(350, 350);

15       Toolkit tk = Toolkit.getDefaultToolkit();
16       img0 = tk.getImage("fly0.gif");
17       img1 = tk.getImage("fly1.gif");
18       img2 = tk.getImage("fly2.gif");
19       sound = Applet.newAudioClip(getClass().getResource("sound.au"));

20       enableEvents(AWTEvent.WINDOW_EVENT_MASK);
21       enableEvents(AWTEvent.KEY_EVENT_MASK);

22       setVisible(true);
23       new Thread(this).start();
24    }

25    public void processWindowEvent(WindowEvent e) {
26       if(e.getID() == WindowEvent.WINDOW_CLOSING) {
27          System.exit(0);
28       }
29    }

30    public void processKeyEvent(KeyEvent e) {
31       if(e.getID() == KeyEvent.KEY_PRESSED) {
32          switch(e.getKeyCode()) {
33             case KeyEvent.VK_L:
34                sound.loop();
35                break;
36             case KeyEvent.VK_P:
37                sound.play();
38                break;
39             case KeyEvent.VK_S:
40                sound.stop();
41                break;
```

```
42          }
43      }
44  }

45  public void run() {
46      while(true) {
47          x = x + dx;
48          y = y + dy;
49          flag = num % 3;
50          repaint();
51          num = num + 1;

52          if(x <= 0) dx = 5;
53          else if((x+60) >= 350) dx = -5;

54          if(y<=0) dy = 5;
55          else if((y + 50) >= getHeight()) dy = -5;

56          try{Thread.sleep(250);}
57          catch(InterruptedException e) {;}
58      }
59  }

60  public void update(Graphics g) {
61      paint(g);
62  }

63  public void paint(Graphics g) {
64      Graphics bufferg;
65      if(bufferPage == null)
            bufferPage = createImage(350, 350);
66      bufferg = bufferPage.getGraphics();
67      if(flag == 0)
            bufferg.drawImage(img0, x, y, this);
68      else if(flag == 1)
            bufferg.drawImage(img1, x, y, this);
69      else if(flag == 2)
            bufferg.drawImage(img2, x, y, this);

70      bufferg.dispose();
71      g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
72  }
73 }
```

行 03	由系统导入 java.applet 包。
行 08	声明音效变量。
行 19	读取声音文件。
行 33～41	以键盘控制音乐的播放。
行 33～34	当按 L 键时进行连续播放。
行 36～37	当按 P 键时播放一次。
行 39～40	当按 S 键时停止播放。

运行结果

运行结果如图 9–2 所示。

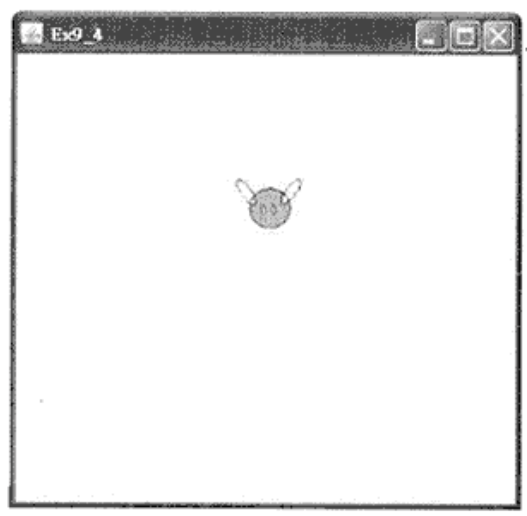


图 9-2

9-5 弈棋音效

如 9–2 节所述，以 Applet.newAudioClip(getClass().getResource("sound.au"))读取声音文件 sound.au，设计范例 52，当单击鼠标左键落棋子时播放音乐；当 3 枚棋子成一直线时为赢，此时播放音乐并显示胜利信息。

范例 52 参考范例 40，文件 Ex9_5.java 的功能是解释弈棋音效的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.applet.*;

04 public class Ex9_5 extends Frame implements Runnable {
05   int x, y;
06   Image img, bufferPage;
07   int[] area_flag = new int[9];
08   int i;
09   AudioClip soundchess1, soundwin;
10   String message = "";

11   public static void main(String args[]) {
12     Ex9_5 workStart=new Ex9_5();
13   }

14   public Ex9_5() {
15     super("Ex9_5");
16     setSize(250,280);

17     Toolkit tk = Toolkit.getDefaultToolkit();
18     img = tk.getImage("Circle.GIF");

19     soundchess1 = Applet.newAudioClip(getClass().getResource("soundchess1.au"));
20     soundwin = Applet.newAudioClip(getClass().getResource("soundwin.au"));

21     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
22     enableEvents(AWTEvent.MOUSE_EVENT_MASK);
```

```
23      setVisible(true);
24      new Thread(this).start();
25  }

26  public void processWindowEvent(WindowEvent e) {
27      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
28          System.exit(0);
29      }
30  }

31  public void processMouseEvent(MouseEvent e) {
32      if(e.getID() == MouseEvent.MOUSE_PRESSED) {
33          x = e.getX();
34          y = e.getY();
35          soundchess1.play();
36      }
37  }

38  public void run() {
39   while(true) {
40      if((x>30)&&(x<90)&&(y>70)&&(y<130)&&(area_flag[0]==0))
                {area_flag[0] = 1;}
41      else if ((x>90)&&(x<150)&&(y>70)&&(y<130)&&(area_flag[1]==0))
                {area_flag[1] = 1;}
42      else if ((x>150)&&(x<210)&&(y>70)&&(y<130)&&(area_flag[2]==0))
                {area_flag[2] = 1;}
43      else if((x>30)&&(x<90)&&(y>130)&&(y<190)&&(area_flag[3]==0))
                {area_flag[3] = 1;}
44      else if((x>90)&&(x<150)&&(y>130)&&(y<190)&&(area_flag[4]==0))
                {area_flag[4] = 1;}
45      else if((x>150)&&(x<210)&&(y>130)&&(y<190)&&(area_flag[5]==0))
                {area_flag[5] = 1;}
46      else if((x>30)&&(x<90)&&(y>190)&&(y<250)&&(area_flag[6]==0))
                {area_flag[6] = 1;}
47      else if((x>90)&&(x<150)&&(y>190)&&(y<250)&&(area_flag[7]==0))
                {area_flag[7] = 1;}
48      else if((x>150)&&(x<210)&&(y>190)&&(y<250)&&(area_flag[8]==0))
                {area_flag[8] = 1;}

49      if ((area_flag[0]==1)&&(area_flag[1]==1)&&(area_flag[2]==1)&&
                (message==null)) {
                message = "User win!!";
                soundwin.play();
        }
50      else if ((area_flag[3]==1)&&(area_flag[4]==1)&&(area_flag[5]==1)&&
                (message==null)) {
                message = "User win!!";
                soundwin.play();
        }
51      else if ((area_flag[6]==1)&&(area_flag[7]==1)&&(area_flag[8]==1)&&
                (message==null)) {
                message = "User win!!";
                soundwin.play();
        }
52      else if ((area_flag[0]==1)&&(area_flag[3]==1)&&(area_flag[6]==1)&&
                (message==null)) {
                message = "User win!!";
                soundwin.play();
        }
```

```
53      else if ((area_flag[1]==1)&&(area_flag[4]==1)&&(area_flag[7]==1)&&
                    (message=="")) {
                    message = "User win!!";
                    soundwin.play();
        }
54      else if ((area_flag[2]==1)&&(area_flag[5]==1)&&(area_flag[8]==1)&&
                    (message=="")) {
                    message = "User win!!";
                    soundwin.play();
        }
55      else if ((area_flag[0]==1)&&(area_flag[4]==1)&&(area_flag[8]==1)&&
                    (message=="")) {
                    message = "User win!!";
                    soundwin.play();
        }
56      else if ((area_flag[2]==1)&&(area_flag[4]==1)&&(area_flag[6]==1)&&
                    (message=="")) {
                    message = "User win!!";
                    soundwin.play();
        }

57      repaint();
58      try{Thread.sleep(250);}
59      catch(InterruptedException e) {;}
60    }
61  }

62  public void update(Graphics g) {
63     paint(g);
64  }

65  public void paint(Graphics g) {
66     Graphics bufferg;
67     if(bufferPage == null)
68        bufferPage = createImage(250, 250);
69     bufferg = bufferPage.getGraphics();

70     bufferg.drawString(message, 20, 20);
71     bufferg.drawLine(90,50,90,230);
72     bufferg.drawLine(150,50,150,230);
73     bufferg.drawLine(30,110,210,110);
74     bufferg.drawLine(30,170,210,170);

75     for (i=0; i<9; i++) {
76        if (area_flag[0] == 1)
              bufferg.drawImage(img, 42, 60, this);
77        if (area_flag[1] == 1)
              bufferg.drawImage(img, 102, 60, this);
78        if (area_flag[2] == 1)
              bufferg.drawImage(img, 164, 60, this);
79        if (area_flag[3] == 1)
              bufferg.drawImage(img, 42, 125, this);
80        if (area_flag[4] == 1)
              bufferg.drawImage(img, 102, 125, this);
81        if (area_flag[5] == 1)
              bufferg.drawImage(img, 164, 125, this);
82        if (area_flag[6] == 1)
              bufferg.drawImage(img, 42, 187, this);
83        if (area_flag[7] == 1)
              bufferg.drawImage(img, 102, 187, this);
```

```
84          if (area_flag[8] == 1)
                bufferg.drawImage(img, 164, 187, this);
85      }

86      bufferg.dispose();
87      g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
88   }
89 }
```

行 03	由系统导入 java.applet 包。
行 09	声明音效变量。
行 10	声明字符串变量。
行 19～20	读取声音文件。
行 35	当单击鼠标左键落棋子时播放音乐。
行 49～56	当 3 枚棋子成一直线时为赢，此时播放音乐并显示胜利信息。

运行结果

运行结果如图 9-3 所示。

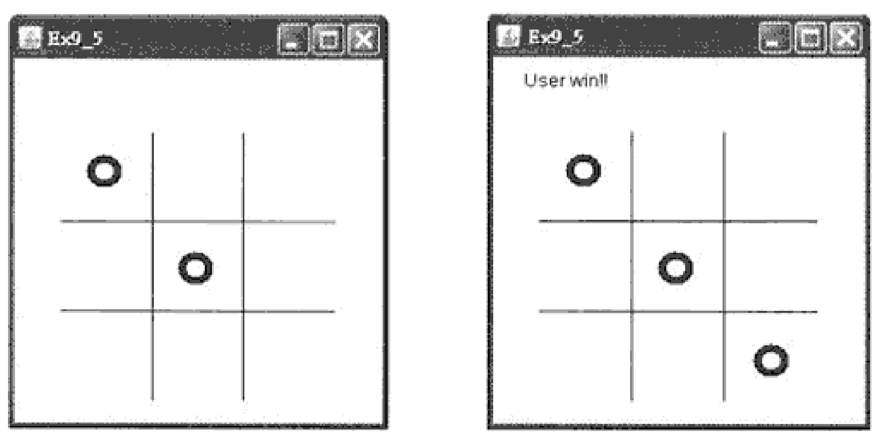


图 9-3

9-6 射击音效

如 9-2 节所述，以 Applet.newAudioClip(getClass().getResource("sound.au"))读取声音文件 shoot1.au 和 explode.au，设计范例 53，按"←"、"→"、"↑"、"↓"键移动发射器；当发射子弹时播放射击音乐；按 Space 键发射子弹，当子弹进入靶车范围时播放击中音乐，并将靶车图片改成爆炸图片，显示中弹信息。

范例 53 参考范例 45，文件 Ex9_6.java 的功能是解释射击音效的应用。

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class Ex9_6 extends Frame implements Runnable {
   int x1=150, y1=225, dx1, dy1, x2=150, y2=10;
   int bx, by, dbx=0, dby=-5, bflag=0, hitflag2=0;
   Image img1, img2, img3, bufferPage=null;
   AudioClip shoot1, explode;
   String message="";
```

```
10  public static void main(String args[]) {
11      Ex9_6 workStart=new Ex9_6();
12  }

13  public Ex9_6() {
14      super("Ex9_6");
15      setSize(350,350);

16      Toolkit tk = Toolkit.getDefaultToolkit();
17      img1 = tk.getImage("car090.gif");
18      img2 = tk.getImage("car180.gif");
19      img3 = tk.getImage("hit2.gif");

20      shoot1 = Applet.newAudioClip(getClass().getResource("shoot1.au"));
21      explode = Applet.newAudioClip(getClass().getResource("explode.au"));

22      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
23      enableEvents(AWTEvent.KEY_EVENT_MASK);

24      setVisible(true);
25      new Thread(this).start();
26  }

27  public void processWindowEvent(WindowEvent e) {
28      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
29          System.exit(0);
30      }
31  }

32  public void processKeyEvent(KeyEvent e) {
33      if(e.getID() == KeyEvent.KEY_PRESSED) {
34          switch(e.getKeyCode()) {
35              case KeyEvent.VK_RIGHT:
36                  dx1 = 5; dy1 = 0;
37                  break;
38              case KeyEvent.VK_LEFT:
39                  dx1 = -5; dy1 = 0;
40                  break;
41              case KeyEvent.VK_UP:
42                  dx1 = 0; dy1 = -5;
43                  break;
44              case KeyEvent.VK_DOWN:
45                  dx1 = 0; dy1 = 5;
46                  break;
47              case KeyEvent.VK_SPACE:
48                  dx1 = 0; dy1 = 0;
49                  bx = x1 + 30;
50                  by = y1 - 5;
51                  bflag = 1;
52                  shoot1.play();
53                  break;
54              default:
55                  dx1 = 0; dy1 = 0;
56          }
57          x1 = x1 + dx1;
58          y1 = y1 + dy1;
59      }
60  }

61  public void run() {
62      while(true) {
63          if(by <= -10) bflag = 0;
```

```
64          if(bflag == 1) by = by + dby;

65          if ((bx>=x2)&&(bx<=x2+60)&&(by<=y2)&&(message==""))){
66              hitflag2=1;
67              explode.play();
68              message="User Win!!";
69          }

70          repaint();
71          try{Thread.sleep(30);}
72          catch(InterruptedException e) {;}
73      }
74  }

75  public void update(Graphics g) {
76      paint(g);
77  }

78  public void paint(Graphics g) {
79      Graphics bufferg;
80      if(bufferPage == null)
            bufferPage = createImage(350, 350);
81      bufferg = bufferPage.getGraphics();

82      if (hitflag2==1) img2=img3;
83      bufferg.drawString(message, 20, 20);

84      bufferg.drawImage(img1, x1, y1, this);
85      bufferg.drawImage(img2, x2, y2, this);

86      bufferg.fillRect(bx, by, 3, 5);
87      bufferg.setColor(Color.white);
88      bufferg.fillRect(bx, by+5, 3, 5);

89      bufferg.dispose();
90      g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
91  }
92 }
```

行 03	由系统导入 java.applet 包。
行 05	声明靶车的位置变量。
行 06	声明靶车的击中标示。
行 08	声明射击与靶车被击中的音效变量。
行 09	声明字符串变量。
行 18	读取靶车图片。
行 19	读取靶车被击中的图片（爆炸图片）。
行 20	读取射击声音文件。
行 21	读取击中靶车的声音文件。
行 52	播放射击音乐。
行 65～69	如果子弹进入靶车范围且尚未显示中弹信息，则设置击中标示，播放击中音乐设置中弹信息。
行 82	如果击中标示为 1，则将靶车图片改成爆炸图片。
行 83	显示中弹信息。

运行结果

运行结果如图 9–4 所示。

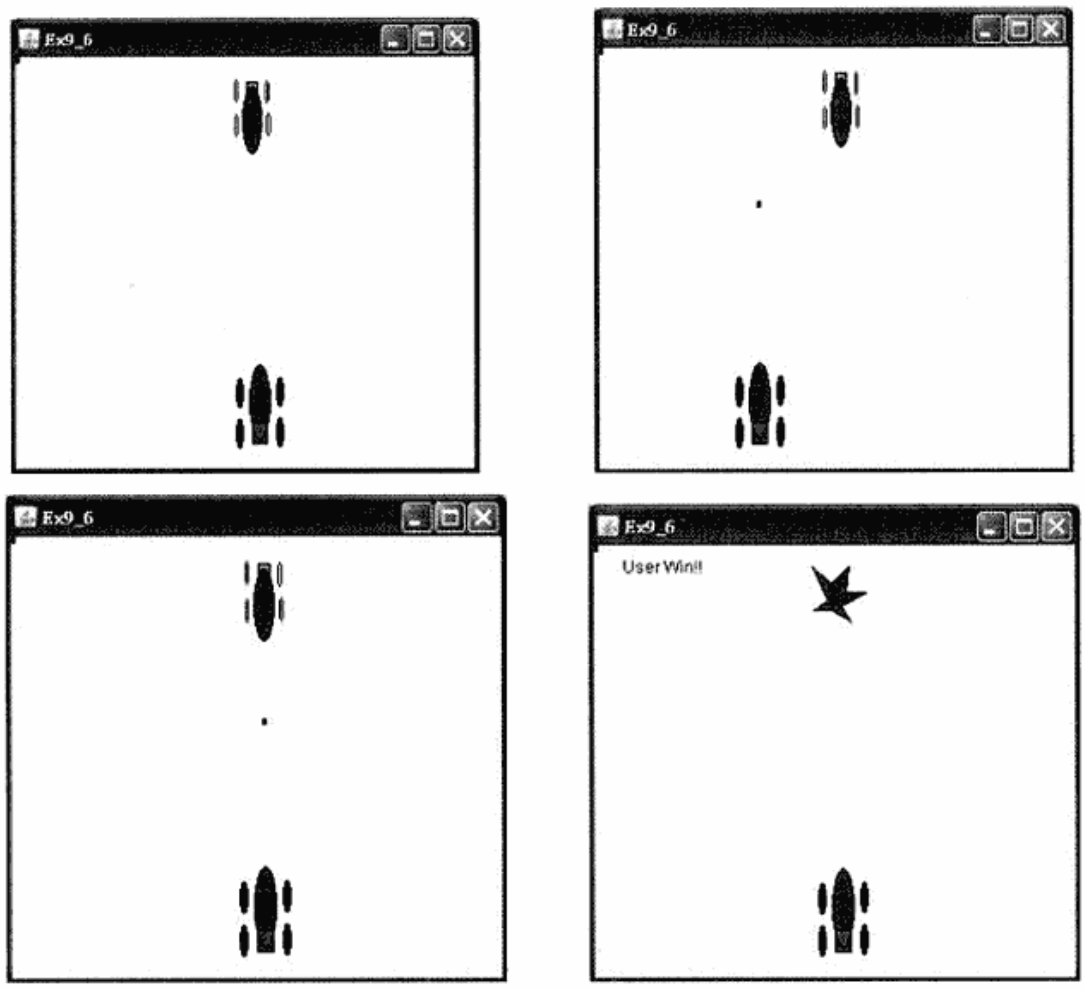


图 9-4

9-7 习题

1. Java 提供哪三种格式的声音文件？
2. AU 格式原本是为哪种环境而设计？
3. 在窗口框架中使用 AU 格式的声音文件时，为何必须导入 applet 包？
4. 如何为窗口框架设计音效接口，用于读取声音文件？
5. sound.play( )方法有什么功能？
6. sound.loop( )方法有什么功能？
7. sound.stop( )方法有什么功能？

PART 03

在线游戏

想来读者都有过玩在线游戏的经历，是否也曾梦想有朝一日玩玩自己设计的在线游戏呢？当然了，我们可以找到一些设计好的软件，只要设置好相关参数就可完成一个电子游戏，笔者以为那并不是我们真正想要的。我们需要的是一点一滴积累的实力，从最基础的了解到能够自主创建的设计能力，只有这样的过程才能磨炼出我们研发的潜力，本部分正是以此作为编写的方向。

Java在网络应用上有着令人震撼的功能，无须像其他语言那样费时费力地搭建网站，Java只要简单数行程序代码即可构建网络平台，这也是为什么目前绝大多数网络游戏或网络应用软件都是以Java编写的，我们也因此不得不学习Java。

在网络对阵方面，本书以弈棋对阵与射击对阵为模型，笔者认为若能完整设计这两类模型，则可设计所有的网络对阵模型。其中，弈棋对阵较为静态，射击对阵较为动态，本部分将以它们为模型详尽介绍其设计过程。

笔者强烈建议，读者在阅读研讨本部分之前，务必先阅读本系列丛书第二册《Java典型应用彻查1000例——网络应用开发》，为了完整了解在线游戏的程序设计方法，必须先了解Java网络程序设计的原理。

Java

Chapter 10　在线命令消息

回顾本系列丛书第二册《Java典型应用彻查1000例——网络应用开发》，其中第3部分“组播”的内容正是在线游戏的设计架构，用户A由客户端向服务器端发起连接，用户B由另一客户端向服务器端发起连接，这样用户A与用户B即可在网络上对阵。在线游戏最主要的问题是如何创建对阵各方的同步画面。如果将Client1的画面通过网络传递给Client2，如此大量的数据传递将无法维持游戏的流畅运行，也会因速度降低而无法保证游戏的趣味性。因此在线游戏各方的同步画面是不能通过传递图像来完成的。实际上，在线游戏各方的同步画面是通过传递动画命令来完成的。其步骤如下：

(1) 在各客户端创建相同的动画组件。

(2) 以在线流将动画命令传递给其他客户端，控制各动画组件做同步操作。

Chapter 11　在线弈棋对阵

一般而言，弈棋为两人对阵（本章中使用User1与User2），网络上两人各使用一台计算机，两台计算机显示相同的棋盘，相同的棋子位置。设计的关键处是当User1在某棋盘位置落子时，远程的User2也可在其计算机上看到同样的动作，反之亦然。解决棋盘画面同步、音效处理、消除闪烁等问题是本章的核心内容。

Chapter 12　在线射击对阵

在线对阵游戏最重要的两个项目就是：弈棋对阵与射击对阵。读者如果能将这两种在线游戏弄清楚，那么其他任何对阵类的游戏均可轻易克服。在第11章我们已探讨了弈棋设计，本章将探讨在线射击对阵的设计。射击对阵是动态的，操作频繁，如果要让对阵双方的窗口图像做同步移动，势必有大量的数据需要传递，不仅会造成设计上的困难，也会降低图像移动的灵活度。如第11章所述，为了解决此问题，对阵双方各自设计图像，不传递数据，仅传递移动命令流，这样可减少传递数据量，也可达到双方图像同步移动的效果。射击对阵是动态的，比较困难，除了要考虑命令流的传递外，还必须考虑射击体、子弹、击中等问题。

Chapter 10 在线命令消息

10-1 简介

回顾本系列丛书第二册《Java 典型应用彻查 1000 例——网络应用开发》，其中第 3 部分“组播”的内容正是在线游戏的设计架构，用户 A 由客户端向服务器端发起连接，用户 B 由另一客户端向服务器端发起连接，这样用户 A 与用户 B 即可在网络上对阵。

在线游戏最主要的问题是如何创建对阵各方的同步画面。如果将客户端 Client1 的画面通过网络传递给客户端 Client2，如此大量的数据传递将无法维持游戏的流畅运行，也会因速度降低而无法保证游戏的趣味性。因此在线游戏各方的同步画面是不能通过传递图像来完成的。

实际上，在线游戏各方的同步画面是通过传递动画命令来完成的。其步骤如下：

(1) 在各客户端创建相同的动画组件。

(2) 以在线流将动画命令传递给其他客户端，控制各动画组件做同步操作。

10-2 在线命令流

当多个动画命令进入网络后，因网络的路径复杂，命令将不会按一定次序抵达目的地，此时我们将无法识别这些命令。本书以余数方法创建识别系数（Factor），在传递命令之前，以识别系数封装（Pack）命令；抵达目的地之后，先辨认命令，解封装（Unpack）识别系数，再使用命令。例如：

1 假设有坐标命令(x，y)，则我们以余数方法创建识别系数，封装命令：

```
x_send = x * 100 + 0;
y_send = y * 100 + 1;
```

2 抵达目的地之后，先辨认命令，解封装识别系数，再使用命令。

假设 rcv 是接收到的命令。

如果 rcv % 100 为 0，则 x = (rcv–0)/100；如果 rcv % 100 为 1，则 y = (rcv–1)/100。

如图 10–1 所示，在线命令流的流程如下：

(1) Client1 与 Client2 分别向 Server 发起连接。

(2) Client1 将本机的动画命令以识别系数封装后，通过 Server 传递给 Client2。

(3) Client2 将接收到的命令经过辨认，解封装识别系数后再运行。同样地，Client2 也会传递动画命令到 Client1。

(4) Client1 与 Client2 将保持画面同步。

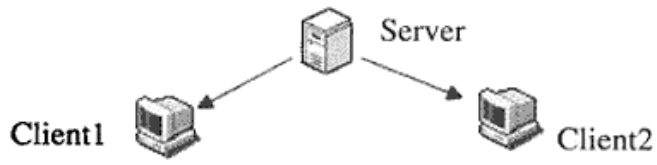


图 10-1

笔者强烈建议，读者在阅读研讨本部分之前，务必先阅读本系列丛书第二册《Java 典型应用彻查 1000 例——网络应用开发》的第 3 部分“组播”，详细了解如何在服务器端（Server）创建网站套接字（ServerSocket）/连接套接字（Socket），创建哈希表（Hash Table），创建线程？如何在发送端（Speaker）创建连接套接字（Socket），创建网络输出流（DataOutputStream）？如何在接收端（Receiver）创建连接套接字（Socket），创建网络输入流（DataInputStream）？

10-3 鼠标命令流

如图 10–1 所示，在 Client1 和 Client2 中创建相同的动画组件，以在线流将鼠标动画命令传递给对方，控制各动画组件做同步操作。其运行步骤如下：

(1) Client1 与 Client2 分别向 Server 发起连接。
(2) Client1 将本机的鼠标动画命令通过 Server 传递给 Client2。
(3) Client2 将本机的鼠标动画命令通过 Server 传递给 Client1。
(4) Client1 与 Client2 将保持画面同步。

范例 54 参考范例 33，文件 Server10_3.java、Client10_3_user1.java、Client10_3_user2.java 的功能是解释鼠标命令流与同步图像的应用。

文件 Server10_3.java：创建网站套接字（ServerSocket）/连接套接字（Socket）、输入/输出流对象（DataStream）、哈希表（Hash Table）、同步线程（Synchronized Threads）。

```
001 import java.net.*;
002 import java.io.*;
003 import java.util.*;

004 public class Server10_3 {
005   private static ServerSocket SS;
006   private static int port;
007   private Hashtable ht = new Hashtable();
008   Socket socket;

009   public Server10_3() {
010     try {
011         SS = new ServerSocket(port);
012         System.out.println("Server is created and waiting Client to connect...");

013         while (true) {
014             socket = SS.accept();
015             System.out.println("Client IP = " +
                                    socket.getInetAddress().getHostAddress());
016             DataOutputStream outstream = new
                                    DataOutputStream(socket.getOutputStream());
017             ht.put(socket, outstream);
```

```
018                 Thread thread = new Thread(new ServerRunnable(socket, ht));
019                 thread.start();
020               }
021       }
022       catch (IOException e) {
023             System.out.println(e.getMessage());
024       }
025     }

026   public static void main(String[] args) {
027       if (args.length < 1) {
028          System.out.println("Usage: java Server10_3 [port]");
029          System.exit(1);
030       }

031       port=Integer.parseInt(args[0]) ;
032       Server10_3 ServerStart=new Server10_3();
033   }
034 }

035 class ServerRunnable implements Runnable {
036   private Socket socket;
037   private Hashtable ht;

038   public ServerRunnable(Socket socket, Hashtable ht) {
039       this.socket = socket;
040       this.ht = ht;
041   }

042   public void run() {
043       DataInputStream instream;
044     try {
045         instream = new DataInputStream(socket.getInputStream());

046         while (true) {
047             int message = instream.readInt();

048             synchronized(ht) {
049                 Enumeration en = ht.elements();
050                 while(en.hasMoreElements()) {
051                     DataOutputStream outstream =
052                         (DataOutputStream)en.nextElement();

053                     try {
054                         outstream.writeInt(message);
055                     }
056                     catch (IOException e) {
057                         System.out.println(e.getMessage());
058                     }
059                 }
060             }
061         }
062   }
063   catch (IOException e) {
064         System.out.println(e.getMessage());
065     }
066     finally {
067         synchronized(ht) {
068             System.out.println("Remove connection: " + socket);
```

```
069                 ht.remove(socket);
070                 try {
071                     socket.close();
072                 }
073                 catch (IOException e) {
074                 }
075             }
076         }
078     }
079 }
```

行 007	创建哈希表 ht。
行 027～031	以命令参数设置端口值（port）。
行 032	调用行 009～025 的构造函数。
行 011	创建服务器端的网站套接字 SS。
行 013～020	以 while 循环与各客户端创建各类关系。
行 014	当各客户端触发网络连接时，立即创建与各客户端的连接套接字 socket。
行 016	以各客户端的连接套接字 socket 创建各自的输出流 outstream。
行 017	将各客户端的 socket、outstream 存储在哈希表 ht 中。
行 018	创建各客户端的线程，以 ServerRunnable 类（行 035～079）生成的对象作为参数。
行 019	启动各线程。
行 035～079	创建实现 Runnable 接口的 ServerRunnable 类。
行 038	将行 014 的 socket、行 017 的 ht 作为参数传递给行 038 的构造函数 ServerRunnable()。
行 045	创建网络输入流 instream，等待发送端送来的信息。
行 048～060	使用关键字 synchronized 设置临界区，同一时刻仅允许一个线程进入运行。
行 050～060	使用哈希表中连接各客户端的网络流分别将数据信息传递给各客户端。

文件 Client10_3_user1.java：读取鼠标指针的坐标(x，y)；以识别系数封装(x，y)成在线命令消息；以网络输出流对象将命令消息输出到网络；读取从网络传来的在线命令消息；解封装在线命令消息的识别系数；绘制图像。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;

086 public class Client10_3_user1 extends Frame implements Runnable {

087   int x=50, y=50;
088   Image img;

089   Socket socket;
090   static String iaddr;
091   static int port;
092   DataOutputStream   outstream;
093   DataInputStream   instream;
094   int x_send, y_send, rcv;
```

```
095  public static void main(String args[]) {
096     if (args.length < 2){
097        System.out.println("USAGE: java Client10_3_user1 [iaddr] [port]");
098        System.exit(1);
099     }

100     iaddr = args[0];
101     port=Integer.parseInt(args[1]);
102     Client10_3_user1 workStart=new Client10_3_user1();
103  }

104  public Client10_3_user1() {
105     super("Client10_3_user1");
106     setSize(350,350);

107     Toolkit tk = Toolkit.getDefaultToolkit();
108     img = tk.getImage("img11_3.jpg");

109     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
110     enableEvents(AWTEvent.MOUSE_EVENT_MASK);

111     setVisible(true);

112     try{
113        socket=new Socket(InetAddress.getByName(iaddr),port);
114        outstream = new DataOutputStream(socket.getOutputStream());
115        instream = new DataInputStream(socket.getInputStream());
116        new Thread(this).start();
117     }
118     catch (Exception e) {
119        e.printStackTrace();
120     }
121  }

122  public void processWindowEvent(WindowEvent e) {
123     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
125        System.exit(0);
126     }
127  }

128  public void processMouseEvent(MouseEvent e) {
129     if(e.getID() == MouseEvent.MOUSE_PRESSED) {
130        try{
131           x = e.getX();
132           y = e.getY();

133           x_send = x * 100 + 0;
134           y_send = y * 100 + 1;
135           outstream.writeInt(x_send);
136           outstream.writeInt(y_send);
137        }
138        catch (Exception f) {
139           f.printStackTrace();
140        }
141     }
142  }

143  public void run() {
144     while(true) {
145        try{
```

```
146             rcv = instream.readInt();
147             if(rcv % 100 == 0) x = (rcv-0)/100;
148             if(rcv % 100 == 1) y = (rcv-1)/100;

149             repaint();
150         }
151         catch (Exception f) {
152             f.printStackTrace();
153         }
154     }
155 }

156 public void paint(Graphics g) {
157     g.drawImage(img, x, y, this);
158 }
159 }
```

行 089～094　有关在线命令各变量的声明。
行 113　以服务器端的 IP、port 创建连接 Server 的套接字，用于驱动行 014。
行 114　创建网络输出流对象 outstream。
行 115　创建网络输入流对象 instream。
行 131～132　由鼠标事件读取鼠标指针的坐标(x，y)。
行 133～134　以识别系数（a*100+n）封装(x，y)成在线命令消息(x_send，y_send)，其中 a 为原始值，n 为顺序值。如 x_send = x*100+0，y_send=y*100+1。
行 135～136　以网络输出流对象 outstream 将命令消息(x_send，y_send）输出到网络。
行 146　由网络输入流对象 instream 读取从网络传来的在线命令消息，并存储在 rcv 中。
行 147～148　解封装在线命令消息的识别系数，如果 rcv%100 为 0，则 x=(rcv−0)/100；如果 rcv%100 为 1，则 y=(rcv−1)/100，这样可求出原始坐标(x，y)。
行 157　以在线命令坐标(x，y）绘制图像。

文件 Client10_3_user2.java：

```
160 import java.awt.*;
161 import java.awt.event.*;
162 import java.math.*;

163 import java.io.*;
164 import java.net.*;
165 import java.util.*;

166 public class Client10_3_user2 extends Frame implements Runnable {

167   int x=50, y=50;
168   Image img;

169   Socket socket;
170   static String iaddr;
171   static int port;
172   DataOutputStream   outstream;
173   DataInputStream   instream;
174   int x_send, y_send, rcv;

175   public static void main(String args[]) {
176     if (args.length < 2){
```

```
177         System.out.println("USAGE: java Client10_3_user2 [iaddr] [port]");
178         System.exit(1);
179       }

180       iaddr = args[0];
181       port=Integer.parseInt(args[1]);
182       Client10_3_user2 workStart=new Client10_3_user2();
183    }

184    public Client10_3_user2() {
185       super("Client10_3_user2");
186       setSize(350,350);

187       Toolkit tk = Toolkit.getDefaultToolkit();
188       img = tk.getImage("img11_3.jpg");

189       enableEvents(AWTEvent.WINDOW_EVENT_MASK);
190       enableEvents(AWTEvent.MOUSE_EVENT_MASK);

191       setVisible(true);

192       try{
193          socket=new Socket(InetAddress.getByName(iaddr),port);
194          outstream = new DataOutputStream(socket.getOutputStream());
195          instream = new DataInputStream(socket.getInputStream());
196          new Thread(this).start();
197       }
198       catch (Exception e) {
199          e.printStackTrace();
200       }
201    }

202    public void processWindowEvent(WindowEvent e) {
203       if(e.getID() == WindowEvent.WINDOW_CLOSING) {
204          System.exit(0);
205       }
206    }

207    public void processMouseEvent(MouseEvent e) {
208       if(e.getID() == MouseEvent.MOUSE_PRESSED) {
209          try{
210             x = e.getX();
211             y = e.getY();

212             x_send = x * 100 + 0;
213             y_send = y * 100 + 1;
214             outstream.writeInt(x_send);
215             outstream.writeInt(y_send);
216          }
217          catch (Exception f) {
218             f.printStackTrace();
219          }
220       }
201    }

202    public void run() {
203       while(true) {
```

```
204         try{
205             rcv = instream.readInt();
206             if(rcv % 100 == 0) x = (rcv-0)/100;
207             if(rcv % 100 == 1) y = (rcv-1)/100;

208             repaint();
209         }
210         catch (Exception f) {
211             f.printStackTrace();
212         }
213     }
214 }

215 public void paint(Graphics g) {
216     g.drawImage(img, x, y, this);
217 }
218 }
```

参考 Client10_3_user1.java 的解说。

编译程序

输入“javac *.java”进行编译，如图 10-2 所示。

图 10-2

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server10_3 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 10-3 所示（其中，1234 为自定义 port，读者应自定义自己的 port。但要注意这里不

能使用公用的 port)。

图 10-3

2 Client1 向 Server 发起连接。

在图 10—4 所示的“命令提示符”窗口中输入“java Client10_3_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 10—5 所示。

图 10-4

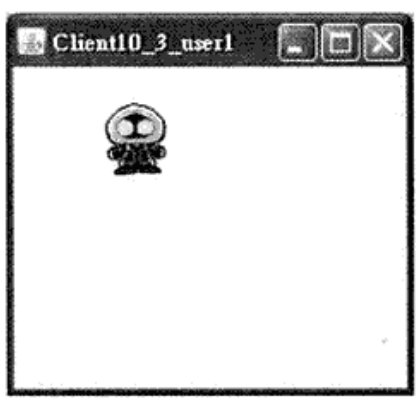


图 10-5

3 Client2 向 Server 发起连接。

在图 10—6 所示的“命令提示符”窗口中输入“java Client10_3_user2 222.28.113.169 1234”则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 10—7 所示。

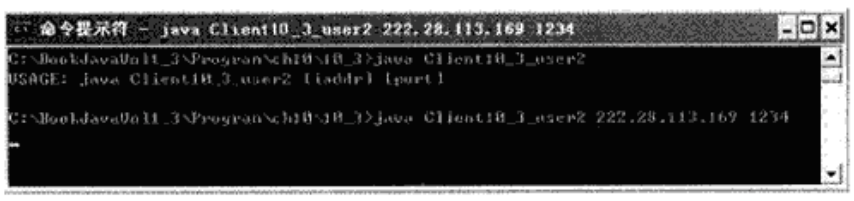


图 10-6

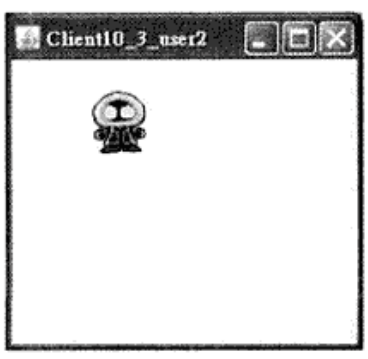


图 10-7

4 运行程序。

在 Client1 上以鼠标左键单击框架内某位置，则图像立即移往新位置，同时远程 Client2 的图像也会同步移动，如图 10-8 所示。在 Client2 上操控图像时，Client1 中也会同步移动。

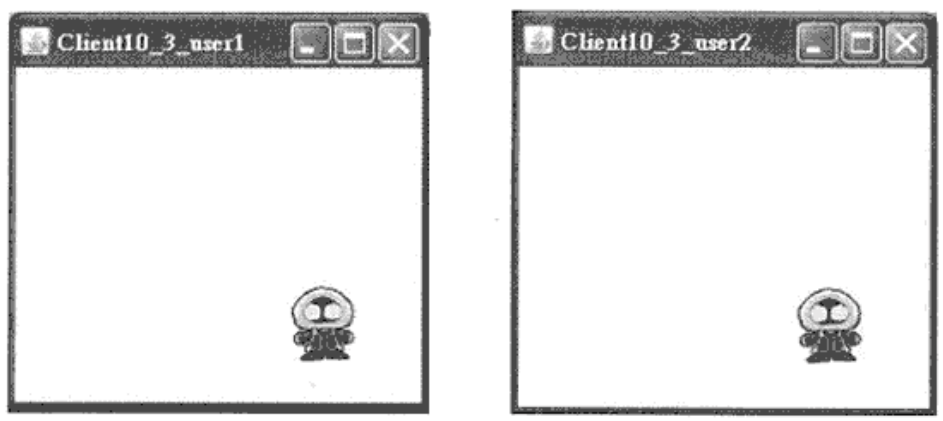


图 10-8

10-4 键盘命令流

如图 10-1 所示，在 Client1 和 Client2 中创建相同的动画组件，以在线流将键盘动画命令传递给对方，控制各动画组件做同步操作。其运行步骤如下：

(1) Client1 与 Client2 分别向 Server 发起连接。
(2) Client1 将本机的键盘动画命令通过 Server 传递给 Client2。
(3) Client2 将本机的键盘动画命令通过 Server 传递给 Client1。
(4) Client1 与 Client2 将保持画面同步。

范例 55 参考范例 41，文件 Server10_4.java、Client10_4_user1.java、Client10_4_user2java 的功能是解释键盘命令流与同步图像的应用。

文件 Server10_4.java：内容与 Server10_3.java 相同，请参考范例 54。

文件 Client10_4_user1.java：以键盘事件改变图像的移动方向。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;

086 public class Client10_4_user1 extends Frame implements Runnable {
087   int x=150, y=235, dx, dy;
088   Image img;

089   Socket socket;
090   static String iaddr;
091   static int port;
092   DataOutputStream   outstream;
093   DataInputStream   instream;
094   int x_send, y_send, rcv;

095   public static void main(String args[]) {
096     if (args.length < 2){
```

```
097         System.out.println("USAGE: java Client10_4_user1 [iaddr] [port]");
098         System.exit(1);
099       }

100       iaddr = args[0];
101       port=Integer.parseInt(args[1]);
102       Client10_4_user1 workStart=new Client10_4_user1();
103   }

104   public Client10_4_user1() {
105       super("Client10_4_user1");
106       setSize(350,350);

107       Toolkit tk = Toolkit.getDefaultToolkit();
108       img = tk.getImage("img11_4.JPG");

109       enableEvents(AWTEvent.WINDOW_EVENT_MASK);
110       enableEvents(AWTEvent.KEY_EVENT_MASK);

111       setVisible(true);

112       try{
113             socket=new Socket(InetAddress.getByName(iaddr),port);
114             outstream = new DataOutputStream(socket.getOutputStream());
115             instream = new DataInputStream(socket.getInputStream());
116             new Thread(this).start();
117       }
118       catch (Exception e) {
119             e.printStackTrace();
120       }
121   }

122   public void processWindowEvent(WindowEvent e) {
123       if(e.getID() == WindowEvent.WINDOW_CLOSING) {
124           System.exit(0);
125       }
126   }

127   public void processKeyEvent(KeyEvent e) {
128       if(e.getID() == KeyEvent.KEY_PRESSED) {
129             switch(e.getKeyCode()) {
130                   case KeyEvent.VK_RIGHT:
131                         dx = 5; dy = 0;
132                         break;
133                   case KeyEvent.VK_LEFT:
134                         dx = -5; dy = 0;
135                         break;
136                   case KeyEvent.VK_UP:
137                         dx = 0; dy = -5;
138                         break;
139                   case KeyEvent.VK_DOWN:
140                         dx = 0; dy = 5;
141                         break;
142                   default:
143                         dx = 0; dy = 0;
144             }
145             try{
146                 x = x + dx;
147                 y = y + dy;
```

```
148                 x_send = x * 100 + 0;
149                 y_send = y * 100 + 1;

150                 outstream.writeInt(x_send);
151                 outstream.writeInt(y_send);
152             }
153             catch (Exception f) {
154                 f.printStackTrace();
155             }
156         }
157     }

158     public void run() {
159       while(true) {
160         try {
161             rcv = instream.readInt();

162             if (rcv % 100 == 0)
                    x = (rcv - 0) / 100;
163             else if (rcv % 100 == 1)
                    y = (rcv - 1) / 100;

164             repaint();
165         }
166         catch (Exception f) {
167             f.printStackTrace();
168         }
169       }
170     }

171     public void paint(Graphics g) {
172         g.drawImage(img, x, y, this);
173     }
174 }
```

行 089～094	有关在线命令各变量的声明。
行 113	以服务器端的 IP、port 创建连接 Server 的套接字，用于驱动行 014。
行 114	创建网络输出流对象 outstream。
行 115	创建网络输入流对象 instream。
行 129～144	以键盘事件控制图像的移动方向。按“→”键，图像向右移一步；按“←”键，图像向左移一步；按“↑”键，图像向上移一步；按“↓”键，图像向下移一步。
行 148～149	以识别系数（a*100+n）封装(x，y）成在线命令消息(x_send，y_send)，其中 a 为原始值，n 为顺序值。如 x_send = x*100+0，y_send=y*100+1。
行 150～151	以网络输出流对象 outstream 将命令消息(x_send，y_send）输出到网络。
行 161	由网络输入流对象 instream 读取从网络传来的在线命令消息，并存储在 rcv 中。
行 162～163	解封装在线命令消息的识别系数，如果 rcv%100 为 0，则 x=(rcv−0)/100；如果 rcv%100 为 1，则 y=(rcv−1)/100，这样可求出原始坐标(x，y)。
行 172	以在线命令坐标(x，y）绘制图像。

文件 Client10_4_user2.java：

```
175 import java.awt.*;
176 import java.awt.event.*;
```

```
import java.math.*;
import java.io.*;
import java.net.*;
import java.util.*;

public class Client10_4_user2 extends Frame implements Runnable {
  int x=150, y=235, dx, dy;
  Image img;

  Socket socket;
  static String iaddr;
  static int port;
  DataOutputStream    outstream;
  DataInputStream    instream;
  int x_send, y_send, rcv;

  public static void main(String args[]) {
    if (args.length < 2){
      System.out.println("USAGE: java Client10_4_user2 [iaddr] [port]");
      System.exit(1);
    }

    iaddr = args[0];
    port=Integer.parseInt(args[1]);
    Client10_4_user2 workStart=new Client10_4_user2();
  }

  public Client10_4_user2() {
    super("Client10_4_user2");
    setSize(350,350);

    Toolkit tk = Toolkit.getDefaultToolkit();
    img = tk.getImage("img11_4.JPG");

    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    enableEvents(AWTEvent.KEY_EVENT_MASK);

    setVisible(true);

    try{
        socket=new Socket(InetAddress.getByName(iaddr),port);
        outstream = new DataOutputStream(socket.getOutputStream());
        instream = new DataInputStream(socket.getInputStream());
        new Thread(this).start();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
  }

  public void processWindowEvent(WindowEvent e) {
    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
       System.exit(0);
    }
  }

  public void processKeyEvent(KeyEvent e) {
    if(e.getID() == KeyEvent.KEY_PRESSED) {
      switch(e.getKeyCode()) {
```

```
225                 case KeyEvent.VK_RIGHT:
226                     dx = 5; dy = 0;
227                     break;
228                 case KeyEvent.VK_LEFT:
229                     dx = -5; dy = 0;
230                     break;
231                 case KeyEvent.VK_UP:
232                     dx = 0; dy = -5;
233                     break;
234                 case KeyEvent.VK_DOWN:
235                     dx = 0; dy = 5;
236                     break;
237                 default:
238                     dx = 0; dy = 0;
239                 }
240                 try{
241                     x = x + dx;
242                     y = y + dy;

243                     x_send = x * 100 + 0;
244                     y_send = y * 100 + 1;

245                     outstream.writeInt(x_send);
246                     outstream.writeInt(y_send);
247                 }
248                 catch (Exception f) {
249                     f.printStackTrace();
250                 }
251         }
252 }

253 public void run() {
254     while(true) {
255         try {
256                 rcv = instream.readInt();

257                 if (rcv % 100 == 0)
                    x = (rcv - 0) / 100;
258                 else if (rcv % 100 == 1)
                    y = (rcv - 1) / 100;

259                 repaint();
260         }
261         catch (Exception f) {
262             f.printStackTrace();
263         }
264     }
265 }

266 public void paint(Graphics g) {
267         g.drawImage(img, x, y, this);
268 }
269 }
```

参考 Client10_4_user1 .java 的解说。

编译程序

输入 "javac *.java" 进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个 "命令提示符" 窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个 "命令提示符" 窗口）。

1 在服务器端创建网站。

输入 "java Server10_4 1234"，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 10-9 所示。

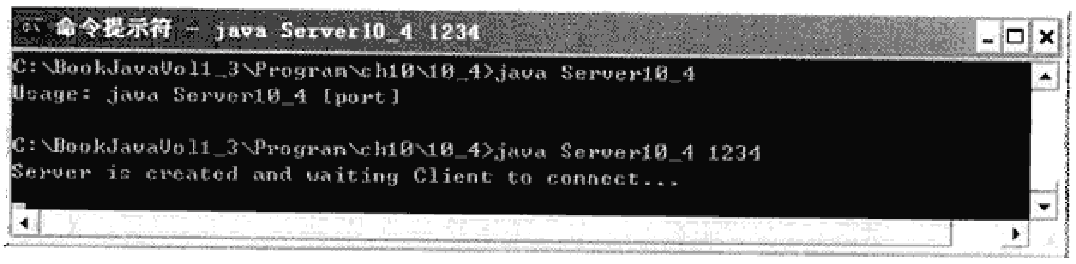


图 10-9

2 Client1 向 Server 发起连接。

在图 10-10 所示的 "命令提示符" 窗口中输入 "java Client10_4_user1 222.28.113.169 1234" 则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 10-11 所示。

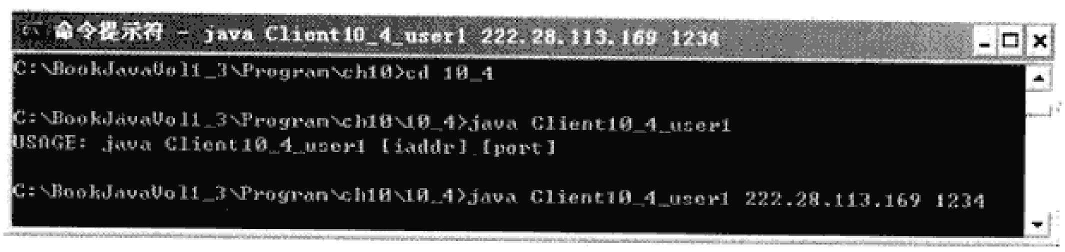


图 10-10

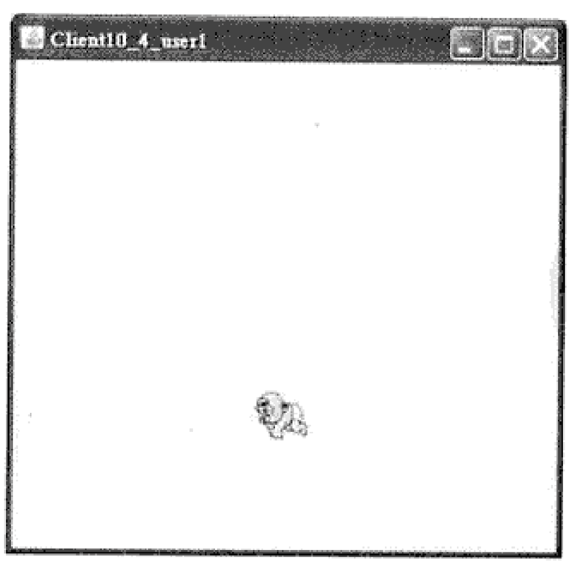


图 10-11

3 Client2 向 Server 发起连接。

在图 10–12 所示的“命令提示符”窗口中输入“java Client10_4_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 10–13 所示。

图 10-12

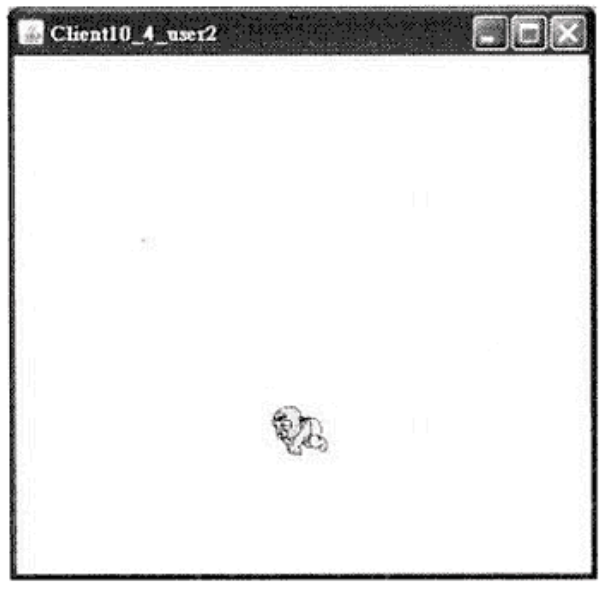


图 10-13

4 运行程序。

在 Client1 上按键盘，则图像立即随设置方向移动，同时远程 Client2 的图像也会同步移动，如图 10–14 所示。反之亦然（按“→”键，图像向右移一步；按“←”键，图像向左移一步；按“↑”键，图像向上移一步；按“↓”键，图像向下移一步）。

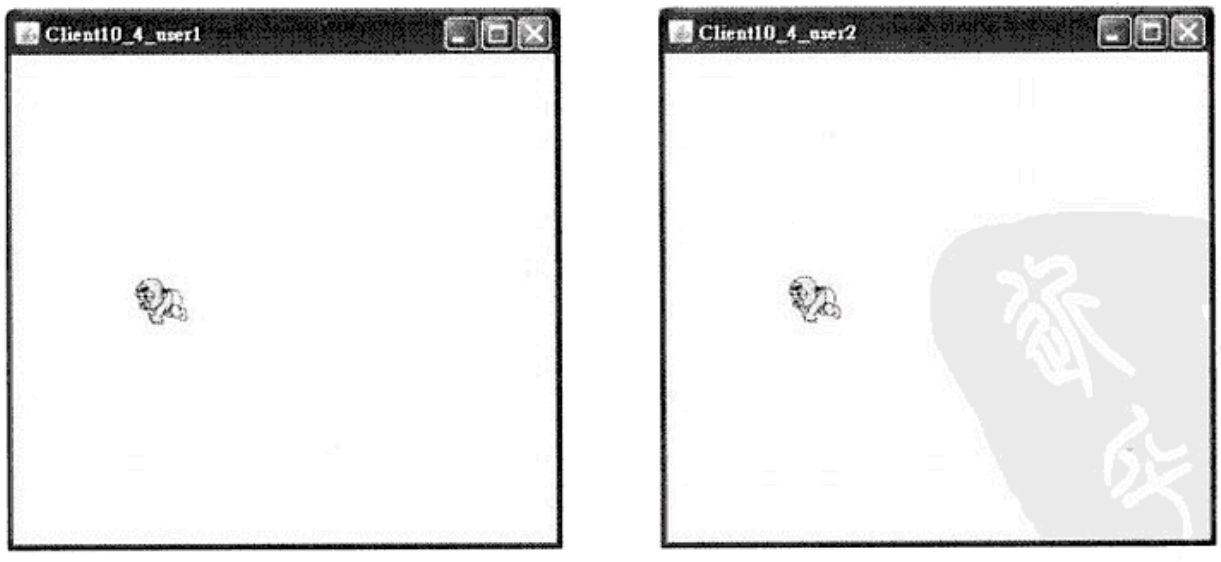


图 10-14

10-5 习题

1. 在线游戏最主要的问题是什么？
2. 为了维护对阵各端的同步画面，为何不能通过传递图像来完成？
3. 如何创建对阵各端的同步画面？
4. 如何辨识网络命令？
5. 如何以余数方法创建识别系数？
6. 在网络对阵中，最少需要创建哪些网络连接套接字？

Chapter 11　在线弈棋对阵

11-1　简介

一般而言，弈棋为两人对阵（本章中使用 User1 与 User2），网络上两人各使用一台计算机，两台计算机显示相同的棋盘，相同的棋子位置。设计的关键处是当 User1 在某棋盘位置落子时，远程的 User2 也可在其计算机上看到同样的动作，反之亦然。解决棋盘画面同步、音效处理、消除闪烁等问题是本章的核心内容。

11-2　网络命令流

在 Client1、Client2 中创建相同的棋盘组件，以在线流将命令传递给对方，控制同步落子位置。其运行步骤如下：

(1) Client1 与 Client2 分别向 Server 报到。
(2) Client1 将本机的落子命令通过 Server 传递给 Client2。
(3) Client2 将本机的落子命令通过 Server 传递给 Client1。
(4) Client1 与 Client2 同步落子。

范例 56　参考范例 40 和范例 54，文件 Server11_2.java、Client11_2_user1、Client11_2_user2 的功能是解释弈棋命令流与同步图像的应用。

文件 Server11_2.java：接受 Client1 送来的命令信息，再转发给 Client2，反之亦然。

```
001 import java.net.*;
002 import java.io.*;
003 import java.util.*;

004 public class Server11_2 {
005   private static ServerSocket SS;
006   private static int port;
007   private Hashtable ht = new Hashtable();
008   Socket socket;

009   public Server11_2() {
010     try {
011         SS = new ServerSocket(port);
012         System.out.println("Server is created and waiting Client to connect...");
```

```
013             while (true) {
014                 socket = SS.accept();
015                 System.out.println("Client IP = " +
                                        socket.getInetAddress().getHostAddress());
016                 DataOutputStream outstream = new
                                        DataOutputStream(socket.getOutputStream());
017                 ht.put(socket, outstream);
018                 Thread thread = new Thread(new ServerRunnable(socket, ht));
019                 thread.start();
020             }
021         }
022         catch (IOException e) {
023             System.out.println(e.getMessage());
024         }
025     }

026     public static void main(String[] args) {
027         if (args.length < 1) {
028             System.out.println("Usage: java Server11_2 [port]");
029             System.exit(1);
030         }

031         port=Integer.parseInt(args[0]) ;
032         Server11_2 ServerStart=new Server11_2();
033     }
034 }

035 class ServerRunnable implements Runnable {
036     private Socket socket;
037     private Hashtable ht;

038     public ServerRunnable(Socket socket, Hashtable ht) {
039         this.socket = socket;
040         this.ht = ht;
041     }

042     public void run() {
043         DataInputStream instream;

044         try {
045             instream = new DataInputStream(socket.getInputStream());

046             while (true) {
047                 int message = instream.readInt();

048                 synchronized(ht) {
049                     Enumeration en = ht.elements();
050                     while(en.hasMoreElements()) {
051                         DataOutputStream outstream =
052                                 (DataOutputStream)en.nextElement();

053                         try {
054                             outstream.writeInt(message);
055                         }
056                         catch (IOException e) {
057                             System.out.println(e.getMessage());
058                         }
059                     }
060                 }
```

```
061          }
062     }
063     catch (IOException e) {
064               System.out.println(e.getMessage());
065       }
066       finally {
067               synchronized(ht) {
068                    System.out.println("Remove connection: " + socket);
069                    ht.remove(socket);
070                    try {
071                         socket.close();
072                    }
073                    catch (IOException e) {
074                    }
075               }
076       }
078     }
079 }
```

行 007　创建哈希表 ht。
行 027～031　以命令参数设置端口值（port）。
行 032　调用行 009～025 的构造函数。
行 011　创建 Server 的网站套接字 SS。
行 013～020　以 while 循环与各客户端创建各类关系。
行 014　当各客户端触发网络连接时，立即创建与各客户端的连接套接字 socket。
行 016　以各客户端的连接套接字 socket 创建各自的输出流 outstream。
行 017　将各客户端的 socket、outstream 存储在哈希表 ht 中。
行 018　创建各客户端的线程，以 ServerRunnable 类（行 035～079）生成的对象作为参数。
行 019　启动各线程。
行 035～079　创建实现 Runnable 接口的 ServerRunnable 类。
行 038　将行 014 的 socket、行 017 的 ht 作为参数传递给行 038 的构造函数 ServerRunnable()。
行 045　创建网络输入流 instream，等待发送端送来的信息。
行 048～060　利用关键字 synchronized 设置临界区，同一时刻仅允许一个线程进入运行。
行 050～060　使用哈希表中连接各客户端的网络流分别将数据信息传递给各客户端。

文件 Client11_2_user1.java：向 Server 报到，将命令信息传递给远程 User2，使棋盘同步落子。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;

086 public class Client11_2_user1 extends Frame implements Runnable {

087   int x, y;
088   Image img;
089   int[] area_flag = new int[9];
090   int i;
```

```
091  Socket socket;
092  static String iaddr;
093  static int port;
094  DataOutputStream   outstream;
095  DataInputStream   instream;
096  int x_send, y_send, rcv, w;

097  public static void main(String args[]) {
098     if (args.length < 2){
099        System.out.println("USAGE: java Client11_2_user1 [iaddr] [port]");
100        System.exit(1);
101     }

102     iaddr = args[0];
103     port=Integer.parseInt(args[1]);
104     Client11_2_user1 workStart=new Client11_2_user1();
105  }

106  public Client11_2_user1() {
107     super("Client11_2_user1");
108     setSize(250,280);

109     Toolkit tk = Toolkit.getDefaultToolkit();
110     img = tk.getImage("Circle.GIF");

111     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
112     enableEvents(AWTEvent.MOUSE_EVENT_MASK);

113     setVisible(true);

114     try{
115        socket=new Socket(InetAddress.getByName(iaddr),port);
116        outstream = new DataOutputStream(socket.getOutputStream());
117        instream = new DataInputStream(socket.getInputStream());
118        new Thread(this).start();
119     }
120     catch (Exception e) {
121        e.printStackTrace();
122     }
123  }

124  public void processWindowEvent(WindowEvent e) {
125     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
126        System.exit(0);
127     }
128  }

129  public void processMouseEvent(MouseEvent e) {
130     if(e.getID() == MouseEvent.MOUSE_PRESSED) {
131        try{
132           x = e.getX();
133           y = e.getY();

134           x_send = x * 100 + 0;
135           y_send = y * 100 + 1;
136           outstream.writeInt(x_send);
137           outstream.writeInt(y_send);
138        }
139        catch (Exception f) {
140           f.printStackTrace();
```

```
141          }
142      }
143  }

144  public void run() {
145      while(true) {
146          try{
147              rcv = instream.readInt();
148              if(rcv % 100 == 0) x = (rcv-0)/100;
149              if(rcv % 100 == 1) y = (rcv-1)/100;
150              w = w+1;

151              if((x>30)&&(x<90)&&(y>70)&&(y<130)&&
                          (area_flag[0]==0)&&(w%2==0))
                         {area_flag[0] = 1;}
152              else if ((x>90)&&(x<150)&&(y>70)&&(y<130)&&
                          (area_flag[1]==0)&&(w%2==0))
                         {area_flag[1] = 1;}
153              else if ((x>150)&&(x<210)&&(y>70)&&(y<130)&&
                          (area_flag[2]==0)&&(w%2==0))
                         {area_flag[2] = 1;}
154              else if((x>30)&&(x<90)&&(y>130)&&(y<190)&&
                          (area_flag[3]==0)&&(w%2==0))
                         {area_flag[3] = 1;}
155              else if((x>90)&&(x<150)&&(y>130)&&(y<190)&&
                          (area_flag[4]==0)&&(w%2==0))
                         {area_flag[4] = 1;}
156              else if((x>150)&&(x<210)&&(y>130)&&(y<190)&&
                          (area_flag[5]==0)&&(w%2==0))
                         {area_flag[5] = 1;}
157              else if((x>30)&&(x<90)&&(y>190)&&(y<250)&&
                          (area_flag[6]==0)&&(w%2==0))
                         {area_flag[6] = 1;}
158              else if((x>90)&&(x<150)&&(y>190)&&(y<250)&&
                          (area_flag[7]==0)&&(w%2==0))
                         {area_flag[7] = 1;}
159              else if((x>150)&&(x<210)&&(y>190)&&(y<250)&&
                          (area_flag[8]==0)&&(w%2==0))
                         {area_flag[8] = 1;}

160              repaint();
161              Thread.sleep(250);
162          }
163          catch (Exception f) {
164              f.printStackTrace();
165          }
166      }
167  }

168  public void paint(Graphics g) {
169      g.drawLine(90,50,90,230);
170      g.drawLine(150,50,150,230);
171      g.drawLine(30,110,210,110);
172      g.drawLine(30,170,210,170);

173      for (i=0; i<9; i++) {
174          if (area_flag[0] == 1)
                 g.drawImage(img, 42, 60, this);
175          if (area_flag[1] == 1)
                 g.drawImage(img, 102, 60, this);
```

```
176             if (area_flag[2] == 1)
                    g.drawImage(img, 164, 60, this);
177             if (area_flag[3] == 1)
                    g.drawImage(img, 42, 125, this);
178             if (area_flag[4] == 1)
                    g.drawImage(img, 102, 125, this);
179             if (area_flag[5] == 1)
                    g.drawImage(img, 164, 125, this);
180             if (area_flag[6] == 1)
                    g.drawImage(img, 42, 187, this);
181             if (area_flag[7] == 1)
                    g.drawImage(img, 102, 187, this);
182             if (area_flag[8] == 1)
                    g.drawImage(img, 164, 187, this);
183         }
184     }
185 }
```

行 091～096　有关在线命令各变量的声明。
行 115　以服务器端的 IP、port 创建连接 Server 的套接字，用于驱动行 014。
行 116　创建网络输出流对象 outstream。
行 117　创建网络输入流对象 instream。
行 132～133　由鼠标事件读取鼠标指针的坐标(x，y)。
行 134～135　以识别系数（a*100+n）封装(x，y)成在线命令信息(x_send，y_send)，其中 a 为原始值，n 为顺序值。如 x_send = x*100+0，y_send=y*100+1。
行 136～137　以网络输出流对象 outstream 将命令信息(x_send，y_send)输出到网络。
行 147　由网络输入流对象 instream 读取从网络传来的在线命令信息，并存储在 rcv 中。
行 148～149　解封装在线命令信息的识别系数，如果 rcv%100 为 0，则 x=(rcv−0)/100；如果 rcv%100 为 1，则 y=(rcv−1)/100，可求出原始坐标(x，y)。
行 150　避免前次落子坐标(x1，y1)与这次落子坐标(x2，y2)造成（x1，y2)的错误，以每循环一次运行 w=w+1 与行 151～159 的 if((w%2) == 0)防止错误发生。
行 151～159　设置 9 个棋盘格的范围。
行 169～172　绘制棋盘网格线。
行 173～183　绘出同步落子图像。

文件 Client11_2_user2.java：向 Server 报到，将命令信息传递给远程 User1，使棋盘同步落子。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

public class Client11_2_user2 extends Frame implements Runnable {

    int x, y;
    Image img;
    int[] area_flag = new int[9];
    int i;
```

```
Socket socket;
static String iaddr;
static int port;
DataOutputStream   outstream;
DataInputStream   instream;
int x_send, y_send, rcv, w;

public static void main(String args[]) {
   if (args.length < 2){
       System.out.println("USAGE: java Client11_2_user2 [iaddr] [port]");
       System.exit(1);
   }

   iaddr = args[0];
   port=Integer.parseInt(args[1]);
   Client11_2_user2 workStart=new Client11_2_user2();
}

public Client11_2_user2() {
      super("Client11_2_user2");
      setSize(250,280);

      Toolkit tk = Toolkit.getDefaultToolkit();
      img = tk.getImage("Circle.GIF");

      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
      enableEvents(AWTEvent.MOUSE_EVENT_MASK);

      setVisible(true);

      try{
          socket=new Socket(InetAddress.getByName(iaddr),port);
          outstream = new DataOutputStream(socket.getOutputStream());
          instream = new DataInputStream(socket.getInputStream());
          new Thread(this).start();
      }
      catch (Exception e) {
          e.printStackTrace();
      }
}

public void processWindowEvent(WindowEvent e) {
     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
      }
}

public void processMouseEvent(MouseEvent e) {
     if(e.getID() == MouseEvent.MOUSE_PRESSED) {
       try{
          x = e.getX();
          y = e.getY();

          x_send = x * 100 + 0;
          y_send = y * 100 + 1;
          outstream.writeInt(x_send);
          outstream.writeInt(y_send);
       }
       catch (Exception f) {
          f.printStackTrace();
```

```
        }
      }
}

public void run() {
   while(true) {
    try{
       rcv = instream.readInt();
       if(rcv % 100 == 0) x = (rcv-0)/100;
       if(rcv % 100 == 1) y = (rcv-1)/100;
       w = w+1;

       if((x>30)&&(x<90)&&(y>70)&&(y<130)&&
          (area_flag[0]==0)&&(w%2==0))
                {area_flag[0] = 1;}
       else if ((x>90)&&(x<150)&&(y>70)&&(y<130)&&
            (area_flag[1]==0)&&(w%2==0))
                {area_flag[1] = 1;}
       else if ((x>150)&&(x<210)&&(y>70)&&(y<130)&&
            (area_flag[2]==0)&&(w%2==0))
                {area_flag[2] = 1;}
       else if((x>30)&&(x<90)&&(y>130)&&(y<190)&&
            (area_flag[3]==0)&&(w%2==0))
                {area_flag[3] = 1;}
       else if((x>90)&&(x<150)&&(y>130)&&(y<190)&&
            (area_flag[4]==0)&&(w%2==0))
                {area_flag[4] = 1;}
       else if((x>150)&&(x<210)&&(y>130)&&(y<190)&&
            (area_flag[5]==0)&&(w%2==0))
                {area_flag[5] = 1;}
       else if((x>30)&&(x<90)&&(y>190)&&(y<250)&&
            (area_flag[6]==0)&&(w%2==0))
                {area_flag[6] = 1;}
        else if((x>90)&&(x<150)&&(y>190)&&(y<250)&&
             (area_flag[7]==0)&&(w%2==0))
                {area_flag[7] = 1;}
        else if((x>150)&&(x<210)&&(y>190)&&(y<250)&&
             (area_flag[8]==0)&&(w%2==0))
                {area_flag[8] = 1;}

        repaint();
        Thread.sleep(250);
       }
       catch (Exception f) {
            f.printStackTrace();
       }
   }
}

public void paint(Graphics g) {
    g.drawLine(90,50,90,230);
    g.drawLine(150,50,150,230);
    g.drawLine(30,110,210,110);
    g.drawLine(30,170,210,170);

    for (i=0; i<9; i++) {
        if (area_flag[0] == 1)
            g.drawImage(img, 42, 60, this);
        if (area_flag[1] == 1)
            g.drawImage(img, 102, 60, this);
```

```
                if (area_flag[2] == 1)
                    g.drawImage(img, 164, 60, this);
                if (area_flag[3] == 1)
                    g.drawImage(img, 42, 125, this);
                if (area_flag[4] == 1)
                    g.drawImage(img, 102, 125, this);
                if (area_flag[5] == 1)
                    g.drawImage(img, 164, 125, this);
                if (area_flag[6] == 1)
                    g.drawImage(img, 42, 187, this);
                if (area_flag[7] == 1)
                    g.drawImage(img, 102, 187, this);
                if (area_flag[8] == 1)
                    g.drawImage(img, 164, 187, this);
            }
        }
    }
```

参考 Client11_2_user1.java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server11_2 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 11-1 所示。

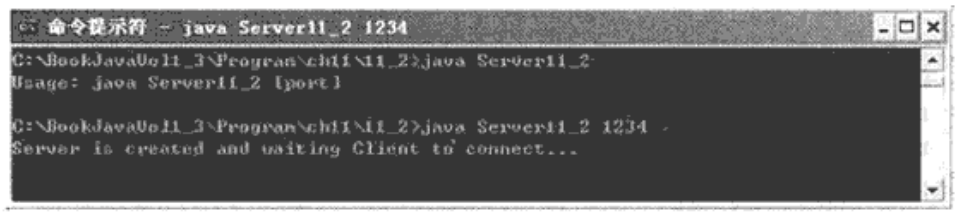


图 11-1

2 Client1 向 Server 报到。

在图 11-2 所示的“命令提示符”窗口中输入“java Client11_2_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 11-3 所示。

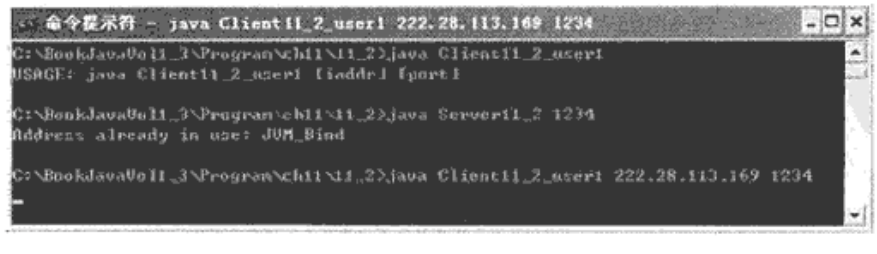


图 11-2

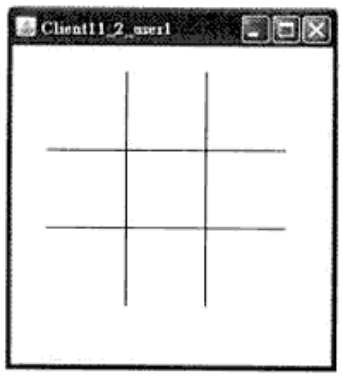


图 11-3

3 Client2 向 Server 报到。

在图 11-4 所示的“命令提示符”窗口中输入“java Client11_2_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 11-5 所示。

图 11-4

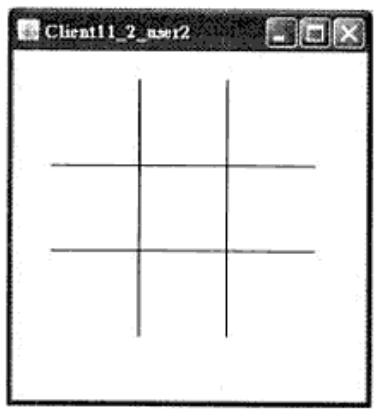


图 11-5

4 运行程序。

在 Client1 上以鼠标左键单击棋盘格某位置，则棋子图像立即显示于该位置，同时远程 Client2 的棋子图像也会同步显示于相应的位置，如图 11-6 所示。反之亦然。

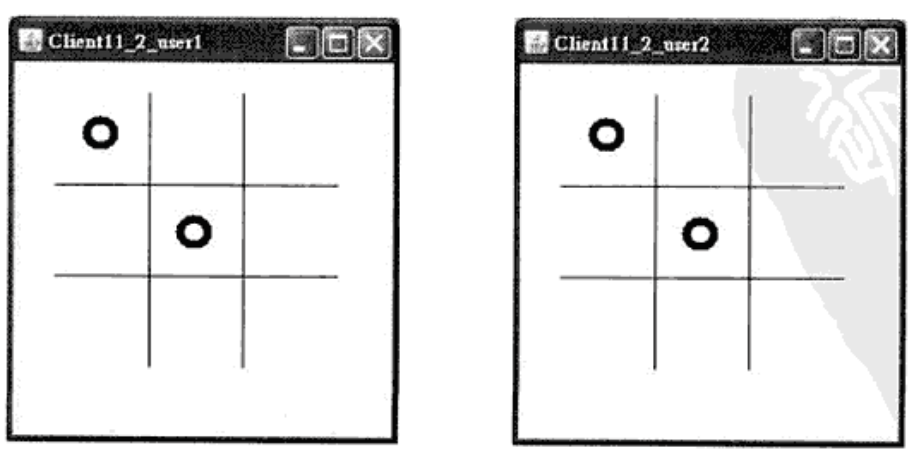


图 11-6

11-3 对阵同步图像

在 Client1 和 Client2 中创建相同的棋盘组件，包括 Circle 棋子图像与 Cross 棋子图像，以在线流将棋子命令传递给对方，控制同步落子位置。其运行步骤如下：

(1) Client1 与 Client2 分别向 Server 报到。

(2) Client1 将 Circle 棋子图像的落子命令通过 Server 传递给 Client2。

(3) Client2 将 Cross 棋子图像的落子命令通过 Server 传递给 Client1。

(4) Client1 与 Client2 对阵同步落子。

范例 57 参考范例 56，文件 Server11_3.java、Client11_3_user1.java、Client11_3_user2.java 的功能是解释弈棋命令流与对阵的应用。

文件 Server11_3.java：内容与 Server11_2.java 相同，请参考范例 56。

文件 Client11_3_user1.java：向 Server 报到，将 Circle 棋子命令信息传递给远程 User2，也接收 User2 传来的 Cross 棋子命令，使棋盘同步落子。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

public class Client11_3_user1 extends Frame implements Runnable {
  int x1, y1, x2, y2;
  Image img1, img2;
  int[] area_flag = new int[9];
  int i;

  Socket socket;
  static String iaddr;
  static int port;
  DataOutputStream    outstream;
  DataInputStream    instream;
  int x1_send, y1_send, x2_send, y2_send, rcv, w;

  public static void main(String args[]) {
    if (args.length < 2){
      System.out.println("USAGE: java Client11_3_user1 [iaddr] [port]");
      System.exit(1);
    }

    iaddr = args[0];
    port=Integer.parseInt(args[1]);
    Client11_3_user1 workStart=new Client11_3_user1();
  }

  public Client11_3_user1() {
    super("Client11_3_user1");
    setSize(250,280);

    Toolkit tk = Toolkit.getDefaultToolkit();
    img1 = tk.getImage("Circle.GIF");
```

```
111     img2 = tk.getImage("Cross.GIF");

112     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
113     enableEvents(AWTEvent.MOUSE_EVENT_MASK);

114     setVisible(true);

115     try{
116         socket=new Socket(InetAddress.getByName(iaddr),port);
117         outstream = new DataOutputStream(socket.getOutputStream());
118         instream = new DataInputStream(socket.getInputStream());
119         new Thread(this).start();
120     }
121     catch (Exception e) {
122         e.printStackTrace();
123     }
124 }

125 public void processWindowEvent(WindowEvent e) {
126     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
127         System.exit(0);
128     }
129 }

130 public void processMouseEvent(MouseEvent e) {
131     if(e.getID() == MouseEvent.MOUSE_PRESSED) {
132         try{
133             x1 = e.getX();
134             y1 = e.getY();

135             x1_send = x1 * 100 + 0;
136             y1_send = y1 * 100 + 1;
137             outstream.writeInt(x1_send);
138             outstream.writeInt(y1_send);
139         }
140         catch (Exception f) {
141             f.printStackTrace();
142         }
143     }
144 }

145 public void run() {
146     while(true) {
147         try{
148             rcv = instream.readInt();
149             if(rcv % 100 == 0) x1 = (rcv-0)/100;
150             if(rcv % 100 == 1) y1 = (rcv-1)/100;
151             if(rcv % 100 == 2) x2 = (rcv-2)/100;
152             if(rcv % 100 == 3) y2 = (rcv-3)/100;
153             w = w+1;

154             if((x1>30)&&(x1<90)&&(y1>70)&&(y1<130)&&
                    (area_flag[0]==0)&&(w%2==0))
                        {area_flag[0] = 1;}
155             else if ((x1>90)&&(x1<150)&&(y1>70)&&(y1<130)&&
                        (area_flag[1]==0)&&(w%2==0))
                            {area_flag[1] = 1;}
156             else if ((x1>150)&&(x1<210)&&(y1>70)&&(y1<130)&&
                        (area_flag[2]==0)&&(w%2==0))
                            {area_flag[2] = 1;}
```

```
157             else if((x1>30)&&(x1<90)&&(y1>130)&&(y1<190)&&
                        (area_flag[3]==0)&&(w%2==0))
                            {area_flag[3] = 1;}
158             else if((x1>90)&&(x1<150)&&(y1>130)&&(y1<190)&&
                        (area_flag[4]==0)&&(w%2==0))
                            {area_flag[4] = 1;}
159             else if((x1>150)&&(x1<210)&&(y1>130)&&(y1<190)&&
                        (area_flag[5]==0)&&(w%2==0))
                            {area_flag[5] = 1;}
160             else if((x1>30)&&(x1<90)&&(y1>190)&&(y1<250)&&
                        (area_flag[6]==0)&&(w%2==0))
                            {area_flag[6] = 1;}
161             else if((x1>90)&&(x1<150)&&(y1>190)&&(y1<250)&&
                        (area_flag[7]==0)&&(w%2==0))
                            {area_flag[7] = 1;}
162             else if((x1>150)&&(x1<210)&&(y1>190)&&(y1<250)&&
                        (area_flag[8]==0)&&(w%2==0))
                            {area_flag[8] = 1;}

163             if((x2>30)&&(x2<90)&&(y2>70)&&(y2<130)&&
                        (area_flag[0]==0)&&(w%2==0))
                            {area_flag[0] = 2;}
164             else if ((x2>90)&&(x2<150)&&(y2>70)&&(y2<130)&&
                        (area_flag[1]==0)&&(w%2==0))
                            {area_flag[1] = 2;}
165             else if ((x2>150)&&(x2<210)&&(y2>70)&&(y2<130)&&
                        (area_flag[2]==0)&&(w%2==0))
                            {area_flag[2] = 2;}
166             else if((x2>30)&&(x2<90)&&(y2>130)&&(y2<190)&&
                        (area_flag[3]==0)&&(w%2==0))
                            {area_flag[3] = 2;}
167             else if((x2>90)&&(x2<150)&&(y2>130)&&(y2<190)&&
                        (area_flag[4]==0)&&(w%2==0))
                            {area_flag[4] = 2;}
168             else if((x2>150)&&(x2<210)&&(y2>130)&&(y2<190)&&
                        (area_flag[5]==0)&&(w%2==0))
                            {area_flag[5] = 2;}
169             else if((x2>30)&&(x2<90)&&(y2>190)&&(y2<250)&&
                        (area_flag[6]==0)&&(w%2==0))
                            {area_flag[6] = 2;}
170             else if((x2>90)&&(x2<150)&&(y2>190)&&(y2<250)&&
                        (area_flag[7]==0)&&(w%2==0))
                            {area_flag[7] = 2;}
171             else if((x2>150)&&(x2<210)&&(y2>190)&&(y2<250)&&
                        (area_flag[8]==0)&&(w%2==0))
                            {area_flag[8] = 2;}

178              repaint();
179              Thread.sleep(250);
180             }
181             catch (Exception f) {
182                 f.printStackTrace();
183             }
184         }
185     }

186     public void paint(Graphics g) {
187         g.drawLine(90,50,90,230);
188         g.drawLine(150,50,150,230);
189         g.drawLine(30,110,210,110);
```

```
190         g.drawLine(30,170,210,170);

191         for (i=0; i<9; i++) {
192             if (area_flag[0] == 1)
                    g.drawImage(img1, 42, 60, this);
193             if (area_flag[1] == 1)
                    g.drawImage(img1, 102, 60, this);
194             if (area_flag[2] == 1)
                    g.drawImage(img1, 164, 60, this);
195             if (area_flag[3] == 1)
                    g.drawImage(img1, 42, 125, this);
196             if (area_flag[4] == 1)
                    g.drawImage(img1, 102, 125, this);
197             if (area_flag[5] == 1)
                    g.drawImage(img1, 164, 125, this);
198             if (area_flag[6] == 1)
                    g.drawImage(img1, 42, 187, this);
199             if (area_flag[7] == 1)
                    g.drawImage(img1, 102, 187, this);
200             if (area_flag[8] == 1)
                    g.drawImage(img1, 164, 187, this);
201         }

202         for (i=0; i<9; i++) {
203             if (area_flag[0] == 2)
                    g.drawImage(img2, 42, 60, this);
204             if (area_flag[1] == 2)
                    g.drawImage(img2, 102, 60, this);
205             if (area_flag[2] == 2)
                    g.drawImage(img2, 164, 60, this);
206             if (area_flag[3] == 2)
                    g.drawImage(img2, 42, 125, this);
207             if (area_flag[4] == 2)
                    g.drawImage(img2, 102, 125, this);
208             if (area_flag[5] == 2)
                    g.drawImage(img2, 164, 125, this);
209             if (area_flag[6] == 2)
                    g.drawImage(img2, 42, 187, this);
210             if (area_flag[7] == 2)
                    g.drawImage(img2, 102, 187, this);
211             if (area_flag[8] == 2)
                    g.drawImage(img2, 164, 187, this);
212         }
213     }
214 }
```

行 110　　读取 Circle 棋子图像（用于 User1）。
行 111　　读取 Cross 棋子图像（用于 User2）。
行 135～138　当 User1 以鼠标单击某棋盘格时，将该棋盘格的坐标(x1，y1)输出到网络。
行 149～150　读取 User1 单击的棋盘格坐标(x1，y1)。
行 151～152　读取 User2 单击的棋盘格坐标(x2，y2)。
行 154～162　设置 9 个棋盘格的范围（用于 Circle 图像）。
行 163～171　设置 9 个棋盘格的范围（用于 Cross 图像）。
行 191～201　将 Circle 图像显示于棋盘格坐标(x1，y1)。
行 202～212　将 Cross 图像显示于棋盘格坐标(x2，y2)。

文件 Client11_3_user2.java：向 Server 报到，将 Cross 棋子命令信息传递给远程 User1，也接收 User1 传来的 Circle 棋子命令，使棋盘同步落子。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

public class Client11_3_user2 extends Frame implements Runnable {

    int x1, y1, x2, y2;
    Image img1, img2;
    int[] area_flag = new int[9];
    int i;

    Socket socket;
    static String iaddr;
    static int port;
    DataOutputStream   outstream;
    DataInputStream   instream;
    int x1_send, y1_send, x2_send, y2_send, rcv, w;

    public static void main(String args[]) {
      if (args.length < 2){
          System.out.println("USAGE: java Client11_3_user2 [iaddr] [port]");
          System.exit(1);
      }

      iaddr = args[0];
      port=Integer.parseInt(args[1]);
      Client11_3_user2 workStart=new Client11_3_user2();
}

public Client11_3_user2() {
      super("Client11_3_user2");
      setSize(250,280);

      Toolkit tk = Toolkit.getDefaultToolkit();
      img1 = tk.getImage("Circle.GIF");
      img2 = tk.getImage("Cross.GIF");

      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
      enableEvents(AWTEvent.MOUSE_EVENT_MASK);

      setVisible(true);

      try{
          socket=new Socket(InetAddress.getByName(iaddr),port);
          outstream = new DataOutputStream(socket.getOutputStream());
          instream = new DataInputStream(socket.getInputStream());
          new Thread(this).start();
       }
       catch (Exception e) {
          e.printStackTrace();
       }
}
```

```
public void processWindowEvent(WindowEvent e) {
      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
          System.exit(0);
        }
}

public void processMouseEvent(MouseEvent e) {
      if(e.getID() == MouseEvent.MOUSE_PRESSED) {
        try{
           x2 = e.getX();
           y2 = e.getY();

           x2_send = x2 * 100 + 2;
           y2_send = y2 * 100 + 3;
           outstream.writeInt(x2_send);
           outstream.writeInt(y2_send);
        }
        catch (Exception f) {
           f.printStackTrace();
        }
      }
}

public void run() {
   while(true) {
     try{
        rcv = instream.readInt();
        if(rcv % 100 == 0) x1 = (rcv-0)/100;
        if(rcv % 100 == 1) y1 = (rcv-1)/100;
        if(rcv % 100 == 2) x2 = (rcv-2)/100;
        if(rcv % 100 == 3) y2 = (rcv-3)/100;
        w = w+1;

        if((x1>30)&&(x1<90)&&(y1>70)&&(y1<130)&&
              (area_flag[0]==0)&&(w%2==0))
                 {area_flag[0] = 1;}
        else if ((x1>90)&&(x1<150)&&(y1>70)&&(y1<130)&&
              (area_flag[1]==0)&&(w%2==0))
                 {area_flag[1] = 1;}
        else if ((x1>150)&&(x1<210)&&(y1>70)&&(y1<130)&&
              (area_flag[2]==0)&&(w%2==0))
                 {area_flag[2] = 1;}
        else if((x1>30)&&(x1<90)&&(y1>130)&&(y1<190)&&
              (area_flag[3]==0)&&(w%2==0))
                 {area_flag[3] = 1;}
        else if((x1>90)&&(x1<150)&&(y1>130)&&(y1<190)&&
              (area_flag[4]==0)&&(w%2==0))
                 {area_flag[4] = 1;}
        else if((x1>150)&&(x1<210)&&(y1>130)&&(y1<190)&&
              (area_flag[5]==0)&&(w%2==0))
                 {area_flag[5] = 1;}
        else if((x1>30)&&(x1<90)&&(y1>190)&&(y1<250)&&
              (area_flag[6]==0)&&(w%2==0))
                 {area_flag[6] = 1;}
        else if((x1>90)&&(x1<150)&&(y1>190)&&(y1<250)&&
              (area_flag[7]==0)&&(w%2==0))
                 {area_flag[7] = 1;}
        else if((x1>150)&&(x1<210)&&(y1>190)&&(y1<250)&&
              (area_flag[8]==0)&&(w%2==0))
```

```
                {area_flag[8] = 1;}

if((x2>30)&&(x2<90)&&(y2>70)&&(y2<130)&&
        (area_flag[0]==0)&&(w%2==0))
            {area_flag[0] = 2;}
else if ((x2>90)&&(x2<150)&&(y2>70)&&(y2<130)&&
        (area_flag[1]==0)&&(w%2==0))
            {area_flag[1] = 2;}
else if ((x2>150)&&(x2<210)&&(y2>70)&&(y2<130)&&
        (area_flag[2]==0)&&(w%2==0))
            {area_flag[2] = 2;}
else if((x2>30)&&(x2<90)&&(y2>130)&&(y2<190)&&
        (area_flag[3]==0)&&(w%2==0))
            {area_flag[3] = 2;}
else if((x2>90)&&(x2<150)&&(y2>130)&&(y2<190)&&
        (area_flag[4]==0)&&(w%2==0))
            {area_flag[4] = 2;}
else if((x2>150)&&(x2<210)&&(y2>130)&&(y2<190)&&
        (area_flag[5]==0)&&(w%2==0))
            {area_flag[5] = 2;}
else if((x2>30)&&(x2<90)&&(y2>190)&&(y2<250)&&
        (area_flag[6]==0)&&(w%2==0))
            {area_flag[6] = 2;}
else if((x2>90)&&(x2<150)&&(y2>190)&&(y2<250)&&
        (area_flag[7]==0)&&(w%2==0))
            {area_flag[7] = 2;}
else if((x2>150)&&(x2<210)&&(y2>190)&&(y2<250)&&
        (area_flag[8]==0)&&(w%2==0))
             {area_flag[8] = 2;}

repaint();
Thread.sleep(250);
        }
        catch (Exception f) {
            f.printStackTrace();
        }
    }
}

public void paint(Graphics g) {
    g.drawLine(90,50,90,230);
    g.drawLine(150,50,150,230);
    g.drawLine(30,110,210,110);
    g.drawLine(30,170,210,170);

    for (i=0; i<9; i++) {
        if (area_flag[0] == 1)
            g.drawImage(img1, 42, 60, this);
        if (area_flag[1] == 1)
            g.drawImage(img1, 102, 60, this);
        if (area_flag[2] == 1)
            g.drawImage(img1, 164, 60, this);
        if (area_flag[3] == 1)
            g.drawImage(img1, 42, 125, this);
        if (area_flag[4] == 1)
            g.drawImage(img1, 102, 125, this);
        if (area_flag[5] == 1)
            g.drawImage(img1, 164, 125, this);
        if (area_flag[6] == 1)
            g.drawImage(img1, 42, 187, this);
```

```
            if (area_flag[7] == 1)
                g.drawImage(img1, 102, 187, this);
            if (area_flag[8] == 1)
                g.drawImage(img1, 164, 187, this);
        }

        for (i=0; i<9; i++) {
            if (area_flag[0] == 2)
                g.drawImage(img2, 42, 60, this);
            if (area_flag[1] == 2)
                g.drawImage(img2, 102, 60, this);
            if (area_flag[2] == 2)
                g.drawImage(img2, 164, 60, this);
            if (area_flag[3] == 2)
                g.drawImage(img2, 42, 125, this);
            if (area_flag[4] == 2)
                g.drawImage(img2, 102, 125, this);
            if (area_flag[5] == 2)
                g.drawImage(img2, 164, 125, this);
            if (area_flag[6] == 2)
                g.drawImage(img2, 42, 187, this);
            if (area_flag[7] == 2)
                g.drawImage(img2, 102, 187, this);
            if (area_flag[8] == 2)
                g.drawImage(img2, 164, 187, this);
        }
    }
}
```

参考 Client11_3_user1 . java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server11_3 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 11-7 所示。

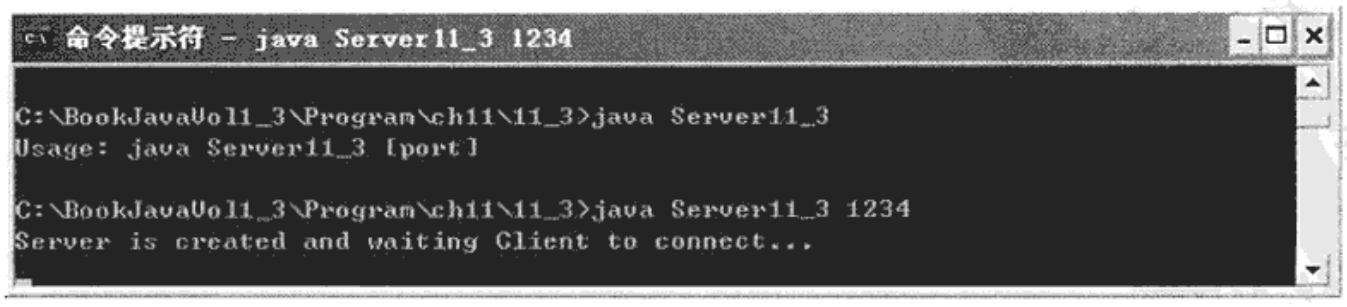


图 11-7

2 Client1 向 Server 报到。

在图 11—8 所示的“命令提示符”窗口中输入“java Client11_3_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 11—9 所示。

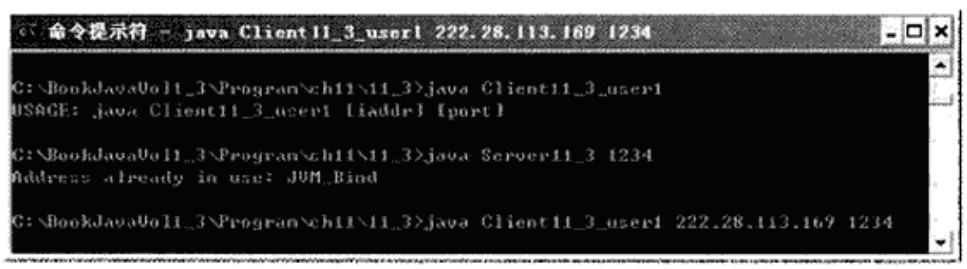


图 11-8

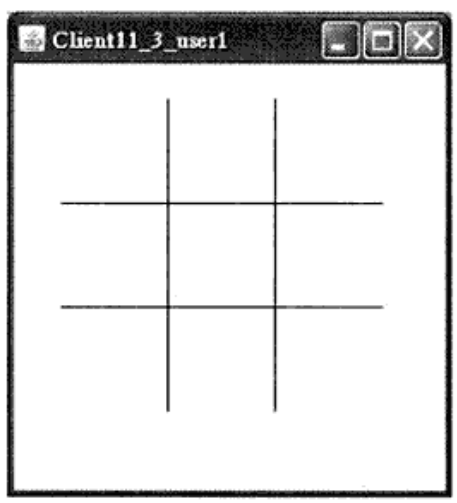


图 11-9

3 Client2 向 Server 报到。

在图 11—10 所示的“命令提示符”窗口中输入“java Client11_3_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 11—11 所示。

图 11-10

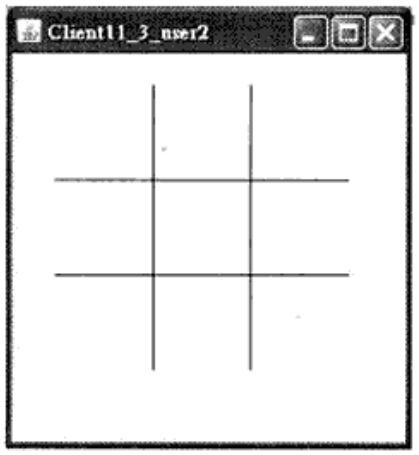


图 11-11

4 运行程序。

在 Client1 上以鼠标左键单击棋盘格某位置，Circle 棋子图像立即显示于该位置，同时远程 Client2 的 Circle 棋子图像也会同步显示于相应的位置；反之 Cross 棋子图像也是一样的，如图 11-12 所示。

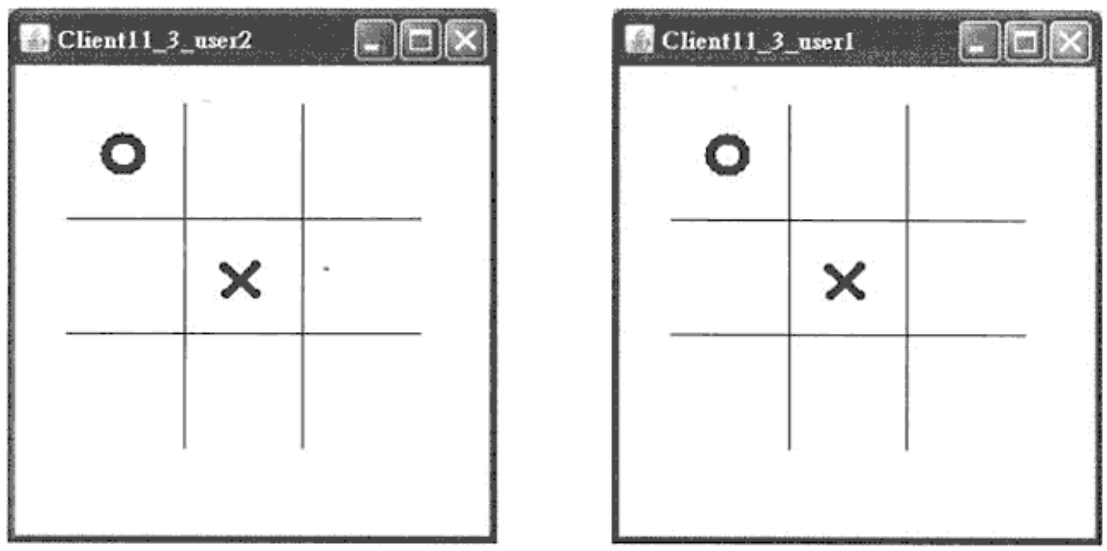


图 11-12

11-4 输赢评定与音效

井字棋有 9 个棋盘格，以 0～8 进行编号；当单击鼠标左键落棋子时，播放落子音效；当弈棋某一方的棋子先成一直线时，则评定他是胜方，此时播放胜利音效并显示胜利信息。

范例 58 参考范例 52 和范例 57，文件 Server11_4.java、Client11_4_user1.java、Client11_4_user2.java 的功能是解释弈棋输赢评定与音效的应用。

文件 Server11_4.java：内容与 Server11_2.java 相同，请参考范例 56。

文件 Client11_4_user1.java：向 Server 报到，将 Circle 棋子命令信息传递给远程 User2，落子时播放音效，当弈棋某一方的棋子先成一直线时，播放胜利音效并显示胜利信息。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;

086 import java.applet.*;

087 public class Client11_4_user1 extends Frame implements Runnable {

088   int x1, y1, x2, y2;
089   Image img1, img2;
090   int[] area_flag = new int[9];
091   int i;

092   Socket socket;
093   static String iaddr;
094   static int port;
095   DataOutputStream   outstream;
```

```
096  DataInputStream   instream;
097  int x1_send, y1_send, x2_send, y2_send, rcv, w;

098  AudioClip sound1, sound2, soundwin;
099  String message = "";

100  public static void main(String args[]) {
101     if (args.length < 2){
102        System.out.println("USAGE: java Client11_4_user1 [iaddr] [port]");
103        System.exit(1);
104     }

105     iaddr = args[0];
106     port=Integer.parseInt(args[1]);
107     Client11_4_user1 workStart=new Client11_4_user1();
108  }

109  public Client11_4_user1() {
110     super("Client11_4_user1");
111     setSize(250,280);

112     Toolkit tk = Toolkit.getDefaultToolkit();
113     img1 = tk.getImage("Circle.GIF");
114     img2 = tk.getImage("Cross.GIF");

115     sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
116     sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
117     soundwin = Applet.newAudioClip(getClass().getResource("soundwin.au"));

118     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
119     enableEvents(AWTEvent.MOUSE_EVENT_MASK);
120     setVisible(true);

121     try{
122          socket=new Socket(InetAddress.getByName(iaddr),port);
123          outstream = new DataOutputStream(socket.getOutputStream());
124          instream = new DataInputStream(socket.getInputStream());
125          new Thread(this).start();
126       }
127       catch (Exception e) {
128          e.printStackTrace();
129       }
130  }

131  public void processWindowEvent(WindowEvent e) {
132     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
133        System.exit(0);
134     }
135  }

136  public void processMouseEvent(MouseEvent e) {
137     if(e.getID() == MouseEvent.MOUSE_PRESSED) {
138        try{
139           x1 = e.getX();
140           y1 = e.getY();
141           sound1.play();

142           x1_send = x1 * 100 + 0;
143           y1_send = y1 * 100 + 1;
```

```
144             outstream.writeInt(x1_send);
145             outstream.writeInt(y1_send);
146           }
147           catch (Exception f) {
148             f.printStackTrace();
149           }
150       }
151   }

152   public void run() {
153     while(true) {
154       try{
155           rcv = instream.readInt();
156           if(rcv % 100 == 0) x1 = (rcv-0)/100;
157           if(rcv % 100 == 1) y1 = (rcv-1)/100;
158           if(rcv % 100 == 2) x2 = (rcv-2)/100;
159           if(rcv % 100 == 3) y2 = (rcv-3)/100;
160           w = w+1;

161           if((x1>30)&&(x1<90)&&(y1>70)&&(y1<130)&&
                    (area_flag[0]==0)&&(w%2==0))
                        {area_flag[0] = 1;}
162           else if ((x1>90)&&(x1<150)&&(y1>70)&&(y1<130)&&
                    (area_flag[1]==0)&&(w%2==0))
                        {area_flag[1] = 1;}
163           else if ((x1>150)&&(x1<210)&&(y1>70)&&(y1<130)&&
                    (area_flag[2]==0)&&(w%2==0))
                        {area_flag[2] = 1;}
164           else if((x1>30)&&(x1<90)&&(y1>130)&&(y1<190)&&
                    (area_flag[3]==0)&&(w%2==0))
                        {area_flag[3] = 1;}
165           else if((x1>90)&&(x1<150)&&(y1>130)&&(y1<190)&&
                    (area_flag[4]==0)&&(w%2==0))
                        {area_flag[4] = 1;}
166           else if((x1>150)&&(x1<210)&&(y1>130)&&(y1<190)&&
                    (area_flag[5]==0)&&(w%2==0))
                        {area_flag[5] = 1;}
167           else if((x1>30)&&(x1<90)&&(y1>190)&&(y1<250)&&
                    (area_flag[6]==0)&&(w%2==0))
                        {area_flag[6] = 1;}
168           else if((x1>90)&&(x1<150)&&(y1>190)&&(y1<250)&&
                    (area_flag[7]==0)&&(w%2==0))
                        {area_flag[7] = 1;}
169           else if((x1>150)&&(x1<210)&&(y1>190)&&(y1<250)&&
                    (area_flag[8]==0)&&(w%2==0))
                        {area_flag[8] = 1;}

170           if((x2>30)&&(x2<90)&&(y2>70)&&(y2<130)&&
                    (area_flag[0]==0)&&(w%2==0))
                        {area_flag[0] = 2;}
171           else if ((x2>90)&&(x2<150)&&(y2>70)&&(y2<130)&&
                    (area_flag[1]==0)&&(w%2==0))
                        {area_flag[1] = 2;}
172           else if ((x2>150)&&(x2<210)&&(y2>70)&&(y2<130)&&
                    (area_flag[2]==0)&&(w%2==0))
                        {area_flag[2] = 2;}
173           else if((x2>30)&&(x2<90)&&(y2>130)&&(y2<190)&&
                    (area_flag[3]==0)&&(w%2==0))
                        {area_flag[3] = 2;}
174           else if((x2>90)&&(x2<150)&&(y2>130)&&(y2<190)&&
```

```
                (area_flag[4]==0)&&(w%2==0))
                    {area_flag[4] = 2;}
175         else if((x2>150)&&(x2<210)&&(y2>130)&&(y2<190)&&
                (area_flag[5]==0)&&(w%2==0))
                    {area_flag[5] = 2;}
176         else if((x2>30)&&(x2<90)&&(y2>190)&&(y2<250)&&
                (area_flag[6]==0)&&(w%2==0))
                    {area_flag[6] = 2;}
177         else if((x2>90)&&(x2<150)&&(y2>190)&&(y2<250)&&
                (area_flag[7]==0)&&(w%2==0))
                    {area_flag[7] = 2;}
178         else if((x2>150)&&(x2<210)&&(y2>190)&&(y2<250)&&
                (area_flag[8]==0)&&(w%2==0))
                    {area_flag[8] = 2;}

179         if ((area_flag[0]==1)&&(area_flag[1]==1)&&
                (area_flag[2]==1)&&(message=="")) {
                 message = "User1 win!!";
                 soundwin.play();
            }
180         else if ((area_flag[3]==1)&&(area_flag[4]==1)&&
                (area_flag[5]==1)&&(message=="")) {
                     message = "User1 win!!";
                     soundwin.play();
            }
181         else if ((area_flag[6]==1)&&(area_flag[7]==1)&&
                (area_flag[8]==1)&&(message=="")) {
                     message = "User1 win!!";
                     soundwin.play();
            }
182         else if ((area_flag[0]==1)&&(area_flag[3]==1)&&
                (area_flag[6]==1)&&(message=="")) {
                     message = "User1 win!!";
                     soundwin.play();
            }
183         else if ((area_flag[1]==1)&&(area_flag[4]==1)&&
                 (area_flag[7]==1)&&(message=="")) {
                     message = "User1 win!!";
                     soundwin.play();
            }
184         else if ((area_flag[2]==1)&&(area_flag[5]==1)&&
                 (area_flag[8]==1)&&(message=="")) {
                     message = "User1 win!!";
                     soundwin.play();
            }
185         else if ((area_flag[0]==1)&&(area_flag[4]==1)&&
                 (area_flag[8]==1)&&(message=="")) {
                     message = "User1 win!!";
                     soundwin.play();
            }
186         else if ((area_flag[2]==1)&&(area_flag[4]==1)&&
                 (area_flag[6]==1)&&(message=="")) {
                     message = "User1 win!!";
                     soundwin.play();
            }

187         if ((area_flag[0]==2)&&(area_flag[1]==2)&&
                 (area_flag[2]==2)&&(message=="")) {
                  message = "User2 win!!";
                  soundwin.play();
```

```
            }
            else if ((area_flag[3]==2)&&(area_flag[4]==2)&&
                    (area_flag[5]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[6]==2)&&(area_flag[7]==2)&&
                    (area_flag[8]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[0]==2)&&(area_flag[3]==2)&&
                    (area_flag[6]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[1]==2)&&(area_flag[4]==2)&&
                    (area_flag[7]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[2]==2)&&(area_flag[5]==2)&&
                    (area_flag[8]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[0]==2)&&(area_flag[4]==2)&&
                    (area_flag[8]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[2]==2)&&(area_flag[4]==2)&&
                    (area_flag[6]==2)&&(message=="")) {
                        message = "User2 win!!";
                    soundwin.play();
            }

            repaint();
            Thread.sleep(250);
          }
          catch (Exception f) {
              f.printStackTrace();
          }
    }
}

public void paint(Graphics g) {
    g.drawString(message, 20, 45);
    g.drawLine(90,50,90,230);
    g.drawLine(150,50,150,230);
    g.drawLine(30,110,210,110);
    g.drawLine(30,170,210,170);

    for (i=0; i<9; i++) {
        if (area_flag[0] == 1)
            g.drawImage(img1, 42, 60, this);
        if (area_flag[1] == 1)
            g.drawImage(img1, 102, 60, this);
        if (area_flag[2] == 1)
            g.drawImage(img1, 164, 60, this);
```

```
213             if (area_flag[3] == 1)
                    g.drawImage(img1, 42, 125, this);
214             if (area_flag[4] == 1)
                    g.drawImage(img1, 102, 125, this);
215             if (area_flag[5] == 1)
                    g.drawImage(img1, 164, 125, this);
216             if (area_flag[6] == 1)
                    g.drawImage(img1, 42, 187, this);
217             if (area_flag[7] == 1)
                    g.drawImage(img1, 102, 187, this);
218             if (area_flag[8] == 1)
                    g.drawImage(img1, 164, 187, this);
219         }

220         for (i=0; i<9; i++) {
221             if (area_flag[0] == 2)
                    g.drawImage(img2, 42, 60, this);
222             if (area_flag[1] == 2)
                    g.drawImage(img2, 102, 60, this);
223             if (area_flag[2] == 2)
                    g.drawImage(img2, 164, 60, this);
224             if (area_flag[3] == 2)
                    g.drawImage(img2, 42, 125, this);
225             if (area_flag[4] == 2)
                    g.drawImage(img2, 102, 125, this);
226             if (area_flag[5] == 2)
                    g.drawImage(img2, 164, 125, this);
227             if (area_flag[6] == 2)
                    g.drawImage(img2, 42, 187, this);
228             if (area_flag[7] == 2)
                    g.drawImage(img2, 102, 187, this);
229             if (area_flag[8] == 2)
                    g.drawImage(img2, 164, 187, this);
230             }
231         }
232 }
```

行 086　　导入 applet 包，用于音效处理。
行 115～117　读取声音文件。
行 141　　当 User1 单击棋盘格时，播放 Circle 音效。
行 179～186　当 User1 赢时，播放胜利音效并显示胜利信息。
行 187～194　当 User2 赢时，播放胜利音效并显示胜利信息。

文件 Client11_4_user2.java：向 Server 报到，将 Cross 棋子命令信息传递给远程 User1，落子时播放音效，当弈棋某一方的棋子先成一直线时，播放胜利音效并显示胜利信息。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

import java.applet.*;

public class Client11_4_user2 extends Frame implements Runnable {
```

```
int x1, y1, x2, y2;
Image img1, img2;
int[] area_flag = new int[9];
int i;

Socket socket;
static String iaddr;
static int port;
DataOutputStream   outstream;
DataInputStream   instream;
int x1_send, y1_send, x2_send, y2_send, rcv, w;

AudioClip sound1,.sound2, soundwin;
String message = "";

public static void main(String args[]) {
   if (args.length < 2){
        System.out.println("USAGE: java Client11_4_user2 [iaddr] [port]");
        System.exit(1);
   }

   iaddr = args[0];
   port=Integer.parseInt(args[1]);
   Client11_4_user2 workStart=new Client11_4_user2();
}

public Client11_4_user2() {
      super("Client11_4_user2");
      setSize(250,280);

      Toolkit tk = Toolkit.getDefaultToolkit();
      img1 = tk.getImage("Circle.GIF");
      img2 = tk.getImage("Cross.GIF");

      sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
      sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
      soundwin = Applet.newAudioClip(getClass().getResource("soundwin.au"));

      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
      enableEvents(AWTEvent.MOUSE_EVENT_MASK);

      setVisible(true);

      try{
            socket=new Socket(InetAddress.getByName(iaddr),port);
            outstream = new DataOutputStream(socket.getOutputStream());
            instream = new DataInputStream(socket.getInputStream());
            new Thread(this).start();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
}

public void processWindowEvent(WindowEvent e) {
      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
           System.exit(0);
         }
}
```

```
public void processMouseEvent(MouseEvent e) {
    if(e.getID() == MouseEvent.MOUSE_PRESSED) {
      try{
        x2 = e.getX();
        y2 = e.getY();
        sound2.play();

        x2_send = x2 * 100 + 2;
        y2_send = y2 * 100 + 3;
        outstream.writeInt(x2_send);
        outstream.writeInt(y2_send);
      }
      catch (Exception f) {
        f.printStackTrace();
      }
    }
}

public void run() {
  while(true) {
   try{
     rcv = instream.readInt();
     if(rcv % 100 == 0) x1 = (rcv-0)/100;
     if(rcv % 100 == 1) y1 = (rcv-1)/100;
     if(rcv % 100 == 2) x2 = (rcv-2)/100;
     if(rcv % 100 == 3) y2 = (rcv-3)/100;
     w = w+1;

     if((x1>30)&&(x1<90)&&(y1>70)&&(y1<130)&&
             (area_flag[0]==0)&&(w%2==0))
                {area_flag[0] = 1;}
     else if ((x1>90)&&(x1<150)&&(y1>70)&&(y1<130)&&
             (area_flag[1]==0)&&(w%2==0))
                {area_flag[1] = 1;}
     else if ((x1>150)&&(x1<210)&&(y1>70)&&(y1<130)&&
             (area_flag[2]==0)&&(w%2==0))
                {area_flag[2] = 1;}
     else if((x1>30)&&(x1<90)&&(y1>130)&&(y1<190)&&
             (area_flag[3]==0)&&(w%2==0))
                {area_flag[3] = 1;}
     else if((x1>90)&&(x1<150)&&(y1>130)&&(y1<190)&&
             (area_flag[4]==0)&&(w%2==0))
                {area_flag[4] = 1;}
     else if((x1>150)&&(x1<210)&&(y1>130)&&(y1<190)&&
             (area_flag[5]==0)&&(w%2==0))
                {area_flag[5] = 1;}
     else if((x1>30)&&(x1<90)&&(y1>190)&&(y1<250)&&
             (area_flag[6]==0)&&(w%2==0))
                {area_flag[6] = 1;}
     else if((x1>90)&&(x1<150)&&(y1>190)&&(y1<250)&&
             (area_flag[7]==0)&&(w%2==0))
                {area_flag[7] = 1;}
     else if((x1>150)&&(x1<210)&&(y1>190)&&(y1<250)&&
             (area_flag[8]==0)&&(w%2==0))
                {area_flag[8] = 1;}

     if((x2>30)&&(x2<90)&&(y2>70)&&(y2<130)&&
             (area_flag[0]==0)&&(w%2==0))
                {area_flag[0] = 2;}
```

```
else if ((x2>90)&&(x2<150)&&(y2>70)&&(y2<130)&&
        (area_flag[1]==0)&&(w%2==0))
            {area_flag[1] = 2;}
else if ((x2>150)&&(x2<210)&&(y2>70)&&(y2<130)&&
        (area_flag[2]==0)&&(w%2==0))
            {area_flag[2] = 2;}
else if((x2>30)&&(x2<90)&&(y2>130)&&(y2<190)&&
        (area_flag[3]==0)&&(w%2==0))
            {area_flag[3] = 2;}
else if((x2>90)&&(x2<150)&&(y2>130)&&(y2<190)&&
        (area_flag[4]==0)&&(w%2==0))
            {area_flag[4] = 2;}
else if((x2>150)&&(x2<210)&&(y2>130)&&(y2<190)&&
        (area_flag[5]==0)&&(w%2==0))
            {area_flag[5] = 2;}
else if((x2>30)&&(x2<90)&&(y2>190)&&(y2<250)&&
        (area_flag[6]==0)&&(w%2==0))
            {area_flag[6] = 2;}
else if((x2>90)&&(x2<150)&&(y2>190)&&(y2<250)&&
        (area_flag[7]==0)&&(w%2==0))
            {area_flag[7] = 2;}
else if((x2>150)&&(x2<210)&&(y2>190)&&(y2<250)&&
        (area_flag[8]==0)&&(w%2==0))
            {area_flag[8] = 2;}

if ((area_flag[0]==1)&&(area_flag[1]==1)&&
        (area_flag[2]==1)&&(message=="")) {
        message = "User1 win!!";
        soundwin.play();
}
else if ((area_flag[3]==1)&&(area_flag[4]==1)&&
        (area_flag[5]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[6]==1)&&(area_flag[7]==1)&&
        (area_flag[8]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[0]==1)&&(area_flag[3]==1)&&
        (area_flag[6]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[1]==1)&&(area_flag[4]==1)&&
        (area_flag[7]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[2]==1)&&(area_flag[5]==1)&&
        (area_flag[8]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[0]==1)&&(area_flag[4]==1)&&
        (area_flag[8]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
```

```
else if ((area_flag[2]==1)&&(area_flag[4]==1)&&
        (area_flag[6]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}

if ((area_flag[0]==2)&&(area_flag[1]==2)&&
        (area_flag[2]==2)&&(message=="")) {
        message = "User2 win!!";
        soundwin.play();
}
else if ((area_flag[3]==2)&&(area_flag[4]==2)&&
        (area_flag[5]==2)&&(message=="")) {
            message = "User2 win!!";
            soundwin.play();
}
else if ((area_flag[6]==2)&&(area_flag[7]==2)&&
        (area_flag[8]==2)&&(message=="")) {
            message = "User2 win!!";
            soundwin.play();
}
else if ((area_flag[0]==2)&&(area_flag[3]==2)&&
        (area_flag[6]==2)&&(message=="")) {
            message = "User2 win!!";
            soundwin.play();
}
else if ((area_flag[1]==2)&&(area_flag[4]==2)&&
        (area_flag[7]==2)&&(message=="")) {
            message = "User2 win!!";
            soundwin.play();
}
else if ((area_flag[2]==2)&&(area_flag[5]==2)&&
            (area_flag[8]==2)&&(message=="")) {
                message = "User2 win!!";
                soundwin.play();
        }
        else if ((area_flag[0]==2)&&(area_flag[4]==2)&&
                (area_flag[8]==2)&&(message=="")) {
                    message = "User2 win!!";
                    soundwin.play();
        }
        else if ((area_flag[2]==2)&&(area_flag[4]==2)&&
                (area_flag[6]==2)&&(message=="")) {
                    message = "User2 win!!";
                    soundwin.play();
        }

        repaint();
        Thread.sleep(250);
       }
       catch (Exception f) {
            f.printStackTrace();
       }
    }
}

public void paint(Graphics g) {
    g.drawString(message, 20, 45);
    g.drawLine(90,50,90,230);
    g.drawLine(150,50,150,230);
```

```
        g.drawLine(30,110,210,110);
        g.drawLine(30,170,210,170);

        for (i=0; i<9; i++) {
            if (area_flag[0] == 1)
                g.drawImage(img1, 42, 60, this);
            if (area_flag[1] == 1)
                g.drawImage(img1, 102, 60, this);
            if (area_flag[2] == 1)
                g.drawImage(img1, 164, 60, this);
            if (area_flag[3] == 1)
                g.drawImage(img1, 42, 125, this);
            if (area_flag[4] == 1)
                g.drawImage(img1, 102, 125, this);
            if (area_flag[5] == 1)
                g.drawImage(img1, 164, 125, this);
            if (area_flag[6] == 1)
                g.drawImage(img1, 42, 187, this);
            if (area_flag[7] == 1)
                g.drawImage(img1, 102, 187, this);
            if (area_flag[8] == 1)
                g.drawImage(img1, 164, 187, this);
        }

        for (i=0; i<9; i++) {
            if (area_flag[0] == 2)
                g.drawImage(img2, 42, 60, this);
            if (area_flag[1] == 2)
                g.drawImage(img2, 102, 60, this);
            if (area_flag[2] == 2)
                g.drawImage(img2, 164, 60, this);
            if (area_flag[3] == 2)
                g.drawImage(img2, 42, 125, this);
            if (area_flag[4] == 2)
                g.drawImage(img2, 102, 125, this);
            if (area_flag[5] == 2)
                g.drawImage(img2, 164, 125, this);
            if (area_flag[6] == 2)
                g.drawImage(img2, 42, 187, this);
            if (area_flag[7] == 2)
                g.drawImage(img2, 102, 187, this);
            if (area_flag[8] == 2)
                g.drawImage(img2, 164, 187, this);
        }
    }
}
```

参考 Client11_4_user1.java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。

若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server11_4 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 11-13 所示。

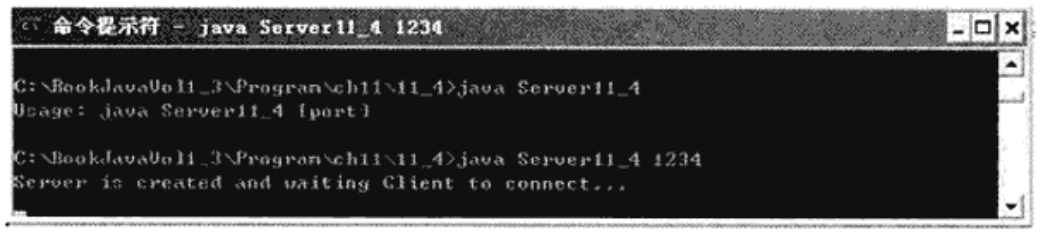


图 11-13

2 Client1 向 Server 报到。

在图 11-14 所示的“命令提示符”窗口中输入“java Client11_4_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 11-15 所示。

图 11-14

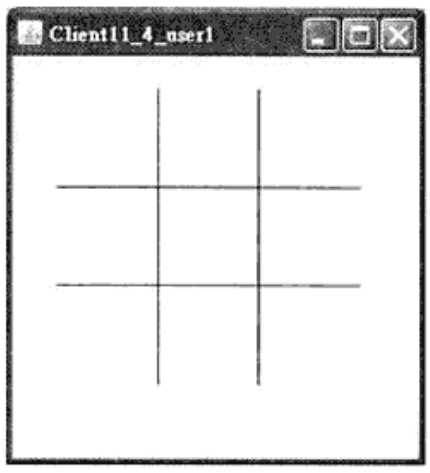


图 11-15

3 Client2 向 Server 报到。

在图 11-16 所示的“命令提示符”窗口中输入“java Client11_4_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 11-17 所示。

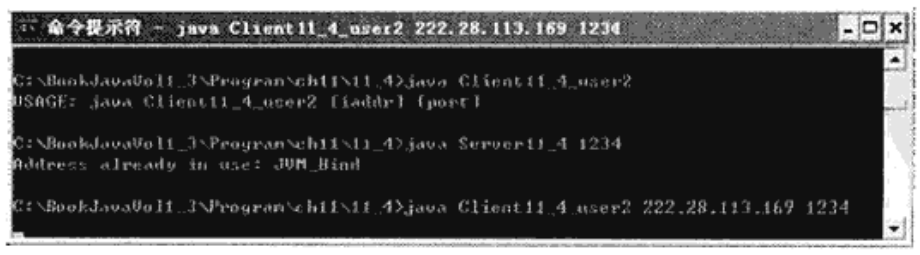


图 11-16

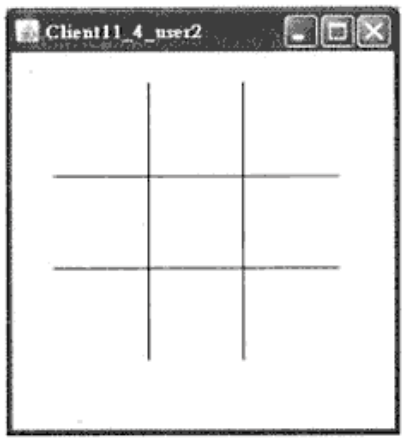


图 11-17

4　运行程序。

单击鼠标左键落棋子时，播放落子音效；当弈棋某一方的棋子先成一直线时，评定他是胜方，此时播放胜利音效并显示胜利信息，如图 11—18 所示。

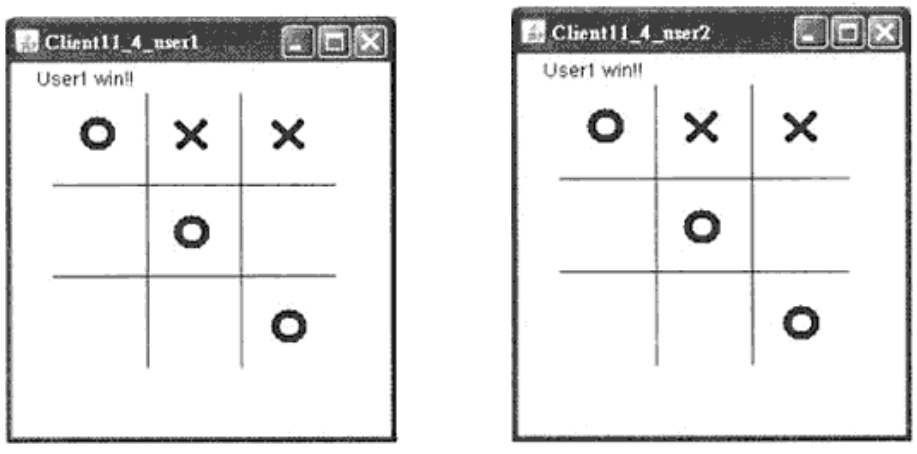


图 11-18

11-5 消除闪烁

如第 8 章所述，因为新图像随时替换旧图像，如果替换得太快，有些部分先到位，有些部分后到位，图像将会显得闪烁而不稳定。为了消除图像闪烁，我们可设置一个缓冲页（Buffer Page），将新图像先置入缓冲页，等待图像各部分全部绘制完成后，再将缓冲页显示在窗口中，这样就可消除图像的闪烁。根据 8—2—4 节消除图像闪烁的程序代码，设计范例 59，消除在线弈棋中图像闪烁的现象。

范例 59　参考范例 48 和范例 58，文件 Server11_5.java、Client11_5_user1.java、Client11_5_user2.java 的功能是解释消除弈棋闪烁的应用。

文件 Server11_5.java：内容与 Server11_2.java 相同，请参考范例 56。

文件 Client11_5_user1.java：作用是消除 User1 端的弈棋棋盘闪烁。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;
```

```
086 import java.applet.*;

087 public class Client11_5_user1 extends Frame implements Runnable {

088   int x1, y1, x2, y2;
089   Image img1, img2, bufferPage;
090   int[] area_flag = new int[9];
091   int i;

092   Socket socket;
093   static String iaddr;
094   static int port;
095   DataOutputStream   outstream;
096   DataInputStream   instream;
097   int x1_send, y1_send, x2_send, y2_send, rcv, w;

098   AudioClip sound1, sound2, soundwin;
099   String message = "";

100   public static void main(String args[]) {
101     if (args.length < 2){
102       System.out.println("USAGE: java Client11_5_user1 [iaddr] [port]");
103       System.exit(1);
104     }

105     iaddr = args[0];
106     port=Integer.parseInt(args[1]);
107     Client11_5_user1 workStart=new Client11_5_user1();
108   }

109   public Client11_5_user1() {
110     super("Client11_5_user1");
111     setSize(250,280);

112     Toolkit tk = Toolkit.getDefaultToolkit();
113     img1 = tk.getImage("Circle.GIF");
114     img2 = tk.getImage("Cross.GIF");

115     sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
116     sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
117     soundwin = Applet.newAudioClip(getClass().getResource("soundwin.au"));

118     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
119     enableEvents(AWTEvent.MOUSE_EVENT_MASK);

120     setVisible(true);

121     try{
122       socket=new Socket(InetAddress.getByName(iaddr),port);
123       outstream = new DataOutputStream(socket.getOutputStream());
124       instream = new DataInputStream(socket.getInputStream());
125       new Thread(this).start();
126     }
127     catch (Exception e) {
128       e.printStackTrace();
129     }
130   }

131   public void processWindowEvent(WindowEvent e) {
132     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
```

```
133         System.exit(0);
134       }
135   }

136   public void processMouseEvent(MouseEvent e) {
137     if(e.getID() == MouseEvent.MOUSE_PRESSED) {
138       try{
139           x1 = e.getX();
140           y1 = e.getY();
141           sound1.play();

142           x1_send = x1 * 100 + 0;
143           y1_send = y1 * 100 + 1;
144           outstream.writeInt(x1_send);
145           outstream.writeInt(y1_send);
146         }
147        catch (Exception f) {
148           f.printStackTrace();
149         }
150     }
151   }

152   public void run() {
153     while(true) {
154       try{
155           rcv = instream.readInt();
156           if(rcv % 100 == 0) x1 = (rcv-0)/100;
157           if(rcv % 100 == 1) y1 = (rcv-1)/100;
158           if(rcv % 100 == 2) x2 = (rcv-2)/100;
159           if(rcv % 100 == 3) y2 = (rcv-3)/100;
160           w = w+1;

161         if((x1>30)&&(x1<90)&&(y1>70)&&(y1<130)&&
                (area_flag[0]==0)&&(w%2==0))
                  {area_flag[0] = 1;}
162       else if ((x1>90)&&(x1<150)&&(y1>70)&&(y1<130)&&
                (area_flag[1]==0)&&(w%2==0))
                  {area_flag[1] = 1;}
163       else if ((x1>150)&&(x1<210)&&(y1>70)&&(y1<130)&&
                (area_flag[2]==0)&&(w%2==0))
                  {area_flag[2] = 1;}
164       else if((x1>30)&&(x1<90)&&(y1>130)&&(y1<190)&&
                (area_flag[3]==0)&&(w%2==0))
                  {area_flag[3] = 1;}
165       else if((x1>90)&&(x1<150)&&(y1>130)&&(y1<190)&&
                (area_flag[4]==0)&&(w%2==0))
                  {area_flag[4] = 1;}
166       else if((x1>150)&&(x1<210)&&(y1>130)&&(y1<190)&&
                (area_flag[5]==0)&&(w%2==0))
                  {area_flag[5] = 1;}
167       else if((x1>30)&&(x1<90)&&(y1>190)&&(y1<250)&&
                (area_flag[6]==0)&&(w%2==0))
                  {area_flag[6] = 1;}
168       else if((x1>90)&&(x1<150)&&(y1>190)&&(y1<250)&&
                (area_flag[7]==0)&&(w%2==0))
                  {area_flag[7] = 1;}
169       else if((x1>150)&&(x1<210)&&(y1>190)&&(y1<250)&&
                (area_flag[8]==0)&&(w%2==0))
                  {area_flag[8] = 1;}
```

```
170         if((x2>30)&&(x2<90)&&(y2>70)&&(y2<130)&&
                (area_flag[0]==0)&&(w%2==0))
                    {area_flag[0] = 2;}
171         else if ((x2>90)&&(x2<150)&&(y2>70)&&(y2<130)&&
                (area_flag[1]==0)&&(w%2==0))
                    {area_flag[1] = 2;}
172         else if ((x2>150)&&(x2<210)&&(y2>70)&&(y2<130)&&
                (area_flag[2]==0)&&(w%2==0))
                    {area_flag[2] = 2;}
173         else if((x2>30)&&(x2<90)&&(y2>130)&&(y2<190)&&
                (area_flag[3]==0)&&(w%2==0))
                    {area_flag[3] = 2;}
174         else if((x2>90)&&(x2<150)&&(y2>130)&&(y2<190)&&
                (area_flag[4]==0)&&(w%2==0))
                    {area_flag[4] = 2;}
175         else if((x2>150)&&(x2<210)&&(y2>130)&&(y2<190)&&
                (area_flag[5]==0)&&(w%2==0))
                    {area_flag[5] = 2;}
176         else if((x2>30)&&(x2<90)&&(y2>190)&&(y2<250)&&
                (area_flag[6]==0)&&(w%2==0))
                    {area_flag[6] = 2;}
177         else if((x2>90)&&(x2<150)&&(y2>190)&&(y2<250)&&
                (area_flag[7]==0)&&(w%2==0))
                    {area_flag[7] = 2;}
178         else if((x2>150)&&(x2<210)&&(y2>190)&&(y2<250)&&
                (area_flag[8]==0)&&(w%2==0))
                    {area_flag[8] = 2;}

179         if ((area_flag[0]==1)&&(area_flag[1]==1)&&
                (area_flag[2]==1)&&(message=="")) {
                  message = "User1 win!!";
                  soundwin.play();
            }
180         else if ((area_flag[3]==1)&&(area_flag[4]==1)&&
                    (area_flag[5]==1)&&(message=="")) {
                        message = "User1 win!!";
                        soundwin.play();
            }
181         else if ((area_flag[6]==1)&&(area_flag[7]==1)&&
                    (area_flag[8]==1)&&(message=="")) {
                        message = "User1 win!!";
                        soundwin.play();
            }
182         else if ((area_flag[0]==1)&&(area_flag[3]==1)&&
                    (area_flag[6]==1)&&(message=="")) {
                        message = "User1 win!!";
                        soundwin.play();
            }
183         else if ((area_flag[1]==1)&&(area_flag[4]==1)&&
                    (area_flag[7]==1)&&(message=="")) {
                        message = "User1 win!!";
                        soundwin.play();
            }
184         else if ((area_flag[2]==1)&&(area_flag[5]==1)&&
                    (area_flag[8]==1)&&(message=="")) {
                        message = "User1 win!!";
                        soundwin.play();
            }
185         else if ((area_flag[0]==1)&&(area_flag[4]==1)&&
                    (area_flag[8]==1)&&(message=="")) {
```

```
                message = "User1 win!!";
                soundwin.play();
            }
186         else if ((area_flag[2]==1)&&(area_flag[4]==1)&&
                (area_flag[6]==1)&&(message=="")) {
                    message = "User1 win!!";
                    soundwin.play();
            }

187         if ((area_flag[0]==2)&&(area_flag[1]==2)&&
                (area_flag[2]==2)&&(message=="")) {
                 message = "User2 win!!";
                 soundwin.play();
            }
188         else if ((area_flag[3]==2)&&(area_flag[4]==2)&&
                (area_flag[5]==2)&&(message=="")) {
                    message = "User2 win!!";
                    soundwin.play();
            }
189         else if ((area_flag[6]==2)&&(area_flag[7]==2)&&
                (area_flag[8]==2)&&(message=="")) {
                    message = "User2 win!!";
                    soundwin.play();
            }
190         else if ((area_flag[0]==2)&&(area_flag[3]==2)&&
                (area_flag[6]==2)&&(message=="")) {
                    message = "User2 win!!";
                    soundwin.play();
            }
191         else if ((area_flag[1]==2)&&(area_flag[4]==2)&&
                (area_flag[7]==2)&&(message=="")) {
                    message = "User2 win!!";
                    soundwin.play();
             }
192          else if ((area_flag[2]==2)&&(area_flag[5]==2)&&
                    (area_flag[8]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
             }
193          else if ((area_flag[0]==2)&&(area_flag[4]==2)&&
                    (area_flag[8]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
             }
194          else if ((area_flag[2]==2)&&(area_flag[4]==2)&&
                    (area_flag[6]==2)&&(message=="")) {
                        message = "User2 win!!";
                    soundwin.play();
             }

195          repaint();
196          Thread.sleep(250);
197         }
198         catch (Exception f) {
199             f.printStackTrace();
200         }
201     }
202 }

203 public void update(Graphics g) {
```

```
    paint(g);
}

public void paint(Graphics g) {
  Graphics bufferg;
  if(bufferPage == null)
      bufferPage = createImage(250, 280);
  bufferg = bufferPage.getGraphics();

  bufferg.drawString(message, 20, 20);
  bufferg.drawLine(90,50,90,230);
  bufferg.drawLine(150,50,150,230);
  bufferg.drawLine(30,110,210,110);
  bufferg.drawLine(30,170,210,170);

  for (i=0; i<9; i++) {
       if (area_flag[0] == 1)
           bufferg.drawImage(img1, 42, 60, this);
       if (area_flag[1] == 1)
           bufferg.drawImage(img1, 102, 60, this);
       if (area_flag[2] == 1)
           bufferg.drawImage(img1, 164, 60, this);
       if (area_flag[3] == 1)
           bufferg.drawImage(img1, 42, 125, this);
       if (area_flag[4] == 1)
           bufferg.drawImage(img1, 102, 125, this);
       if (area_flag[5] == 1)
           bufferg.drawImage(img1, 164, 125, this);
       if (area_flag[6] == 1)
           bufferg.drawImage(img1, 42, 187, this);
       if (area_flag[7] == 1)
           bufferg.drawImage(img1, 102, 187, this);
       if (area_flag[8] == 1)
           bufferg.drawImage(img1, 164, 187, this);
  }

  for (i=0; i<9; i++) {
       if (area_flag[0] == 2)
           bufferg.drawImage(img2, 42, 60, this);
       if (area_flag[1] == 2)
           bufferg.drawImage(img2, 102, 60, this);
       if (area_flag[2] == 2)
           bufferg.drawImage(img2, 164, 60, this);
       if (area_flag[3] == 2)
           bufferg.drawImage(img2, 42, 125, this);
       if (area_flag[4] == 2)
           bufferg.drawImage(img2, 102, 125, this);
       if (area_flag[5] == 2)
           bufferg.drawImage(img2, 164, 125, this);
       if (area_flag[6] == 2)
           bufferg.drawImage(img2, 42, 187, this);
       if (area_flag[7] == 2)
           bufferg.drawImage(img2, 102, 187, this);
       if (area_flag[8] == 2)
           bufferg.drawImage(img2, 164, 187, this);
  }

  bufferg.dispose();
  g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
```

```
240   }
241 }
```

行 089	声明缓冲页。
行 203～205	更换缓冲页。
行 207～210	创建缓冲页。
行 211	缓冲页显示胜利信息。
行 212～215	缓冲页显示棋盘网格线。
行 216～226	缓冲页显示 Circle 棋子。
行 227～237	缓冲页显示 Cross 棋子。
行 238	释放缓冲页资源。
行 239	补充绘制。

文件 Client11_5_user2.java：作用是消除 user2 端的弈棋棋盘闪烁。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

import java.applet.*;

public class Client11_5_user2 extends Frame implements Runnable {

    int x1, y1, x2, y2;
    Image img1, img2, bufferPage;
    int[] area_flag = new int[9];
    int i;

    Socket socket;
    static String iaddr;
    static int port;
    DataOutputStream   outstream;
    DataInputStream   instream;
    int x1_send, y1_send, x2_send, y2_send, rcv, w;

    AudioClip sound1, sound2, soundwin;
    String message = "";

    public static void main(String args[]) {
      if (args.length < 2){
          System.out.println("USAGE: java Client11_5_user2 [iaddr] [port]");
          System.exit(1);
      }

      iaddr = args[0];
      port=Integer.parseInt(args[1]);
      Client11_5_user2 workStart=new Client11_5_user2();
    }

    public Client11_5_user2() {
        super("Client11_5_user2");
```

```
        setSize(250,280);

        Toolkit tk = Toolkit.getDefaultToolkit();
        img1 = tk.getImage("Circle.GIF");
        img2 = tk.getImage("Cross.GIF");

        sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
        sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
        soundwin = Applet.newAudioClip(getClass().getResource("soundwin.au"));

        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        enableEvents(AWTEvent.MOUSE_EVENT_MASK);

        setVisible(true);

        try{
            socket=new Socket(InetAddress.getByName(iaddr),port);
            outstream = new DataOutputStream(socket.getOutputStream());
            instream = new DataInputStream(socket.getInputStream());
            new Thread(this).start();
         }
         catch (Exception e) {
            e.printStackTrace();
         }
}

public void processWindowEvent(WindowEvent e) {
      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
          System.exit(0);
        }
}

public void processMouseEvent(MouseEvent e) {
      if(e.getID() == MouseEvent.MOUSE_PRESSED) {
        try{
          x2 = e.getX();
          y2 = e.getY();
          sound2.play();

          x2_send = x2 * 100 + 2;
          y2_send = y2 * 100 + 3;
          outstream.writeInt(x2_send);
          outstream.writeInt(y2_send);
        }
        catch (Exception f) {
          f.printStackTrace();
        }
      }
}

public void run() {
   while(true) {
     try{
       rcv = instream.readInt();
       if(rcv % 100 == 0) x1 = (rcv-0)/100;
       if(rcv % 100 == 1) y1 = (rcv-1)/100;
       if(rcv % 100 == 2) x2 = (rcv-2)/100;
       if(rcv % 100 == 3) y2 = (rcv-3)/100;
       w = w+1;
```

```
if((x1>30)&&(x1<90)&&(y1>70)&&(y1<130)&&
        (area_flag[0]==0)&&(w%2==0))
            {area_flag[0] = 1;}
else if ((x1>90)&&(x1<150)&&(y1>70)&&(y1<130)&&
        (area_flag[1]==0)&&(w%2==0))
            {area_flag[1] = 1;}
else if ((x1>150)&&(x1<210)&&(y1>70)&&(y1<130)&&
        (area_flag[2]==0)&&(w%2==0))
            {area_flag[2] = 1;}
else if((x1>30)&&(x1<90)&&(y1>130)&&(y1<190)&&
        (area_flag[3]==0)&&(w%2==0))
            {area_flag[3] = 1;}
else if((x1>90)&&(x1<150)&&(y1>130)&&(y1<190)&&
        (area_flag[4]==0)&&(w%2==0))
            {area_flag[4] = 1;}
else if((x1>150)&&(x1<210)&&(y1>130)&&(y1<190)&&
        (area_flag[5]==0)&&(w%2==0))
            {area_flag[5] = 1;}
else if((x1>30)&&(x1<90)&&(y1>190)&&(y1<250)&&
        (area_flag[6]==0)&&(w%2==0))
            {area_flag[6] = 1;}
else if((x1>90)&&(x1<150)&&(y1>190)&&(y1<250)&&
        (area_flag[7]==0)&&(w%2==0))
            {area_flag[7] = 1;}
else if((x1>150)&&(x1<210)&&(y1>190)&&(y1<250)&&
        (area_flag[8]==0)&&(w%2==0))
            {area_flag[8] = 1;}

if((x2>30)&&(x2<90)&&(y2>70)&&(y2<130)&&
        (area_flag[0]==0)&&(w%2==0))
            {area_flag[0] = 2;}
else if ((x2>90)&&(x2<150)&&(y2>70)&&(y2<130)&&
        (area_flag[1]==0)&&(w%2==0))
            {area_flag[1] = 2;}
else if ((x2>150)&&(x2<210)&&(y2>70)&&(y2<130)&&
        (area_flag[2]==0)&&(w%2==0))
            {area_flag[2] = 2;}
else if((x2>30)&&(x2<90)&&(y2>130)&&(y2<190)&&
        (area_flag[3]==0)&&(w%2==0))
            {area_flag[3] = 2;}
else if((x2>90)&&(x2<150)&&(y2>130)&&(y2<190)&&
        (area_flag[4]==0)&&(w%2==0))
            {area_flag[4] = 2;}
else if((x2>150)&&(x2<210)&&(y2>130)&&(y2<190)&&
        (area_flag[5]==0)&&(w%2==0))
            {area_flag[5] = 2;}
else if((x2>30)&&(x2<90)&&(y2>190)&&(y2<250)&&
        (area_flag[6]==0)&&(w%2==0))
            {area_flag[6] = 2;}
else if((x2>90)&&(x2<150)&&(y2>190)&&(y2<250)&&
        (area_flag[7]==0)&&(w%2==0))
            {area_flag[7] = 2;}
else if((x2>150)&&(x2<210)&&(y2>190)&&(y2<250)&&
        (area_flag[8]==0)&&(w%2==0))
            {area_flag[8] = 2;}

if ((area_flag[0]==1)&&(area_flag[1]==1)&&
        (area_flag[2]==1)&&(message=="")) {
        message = "User1 win!!";
        soundwin.play();
```

```
}
else if ((area_flag[3]==1)&&(area_flag[4]==1)&&
        (area_flag[5]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[6]==1)&&(area_flag[7]==1)&&
        (area_flag[8]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[0]==1)&&(area_flag[3]==1)&&
        (area_flag[6]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[1]==1)&&(area_flag[4]==1)&&
        (area_flag[7]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[2]==1)&&(area_flag[5]==1)&&
        (area_flag[8]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[0]==1)&&(area_flag[4]==1)&&
        (area_flag[8]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}
else if ((area_flag[2]==1)&&(area_flag[4]==1)&&
        (area_flag[6]==1)&&(message=="")) {
            message = "User1 win!!";
            soundwin.play();
}

if ((area_flag[0]==2)&&(area_flag[1]==2)&&
        (area_flag[2]==2)&&(message=="")) {
        message = "User2 win!!";
        soundwin.play();
}
else if ((area_flag[3]==2)&&(area_flag[4]==2)&&
        (area_flag[5]==2)&&(message=="")) {
            message = "User2 win!!";
            soundwin.play();
}
else if ((area_flag[6]==2)&&(area_flag[7]==2)&&
        (area_flag[8]==2)&&(message=="")) {
            message = "User2 win!!";
            soundwin.play();
}
else if ((area_flag[0]==2)&&(area_flag[3]==2)&&
        (area_flag[6]==2)&&(message=="")) {
            message = "User2 win!!";
            soundwin.play();
}
else if ((area_flag[1]==2)&&(area_flag[4]==2)&&
         (area_flag[7]==2)&&(message=="")) {
            message = "User2 win!!";
```

```
                soundwin.play();
            }
            else if ((area_flag[2]==2)&&(area_flag[5]==2)&&
                    (area_flag[8]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[0]==2)&&(area_flag[4]==2)&&
                    (area_flag[8]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }
            else if ((area_flag[2]==2)&&(area_flag[4]==2)&&
                    (area_flag[6]==2)&&(message=="")) {
                        message = "User2 win!!";
                        soundwin.play();
            }

            repaint();
            Thread.sleep(250);
          }
          catch (Exception f) {
               f.printStackTrace();
          }
       }
}

public void update(Graphics g) {
    paint(g);
}

public void paint(Graphics g) {
Graphics bufferg;
if(bufferPage == null)
    bufferPage = createImage(250, 280);
bufferg = bufferPage.getGraphics();

bufferg.drawString(message, 20, 20);
bufferg.drawLine(90,50,90,230);
bufferg.drawLine(150,50,150,230);
bufferg.drawLine(30,110,210,110);
bufferg.drawLine(30,170,210,170);

for (i=0; i<9; i++) {
      if (area_flag[0] == 1)
          bufferg.drawImage(img1, 42, 60, this);
      if (area_flag[1] == 1)
          bufferg.drawImage(img1, 102, 60, this);
      if (area_flag[2] == 1)
          bufferg.drawImage(img1, 164, 60, this);
      if (area_flag[3] == 1)
          bufferg.drawImage(img1, 42, 125, this);
      if (area_flag[4] == 1)
          bufferg.drawImage(img1, 102, 125, this);
      if (area_flag[5] == 1)
          bufferg.drawImage(img1, 164, 125, this);
      if (area_flag[6] == 1)
          bufferg.drawImage(img1, 42, 187, this);
      if (area_flag[7] == 1)
          bufferg.drawImage(img1, 102, 187, this);
```

```
            if (area_flag[8] == 1)
                bufferg.drawImage(img1, 164, 187, this);
        }

        for (i=0; i<9; i++) {
            if (area_flag[0] == 2)
                bufferg.drawImage(img2, 42, 60, this);
            if (area_flag[1] == 2)
                bufferg.drawImage(img2, 102, 60, this);
            if (area_flag[2] == 2)
                bufferg.drawImage(img2, 164, 60, this);
            if (area_flag[3] == 2)
                bufferg.drawImage(img2, 42, 125, this);
            if (area_flag[4] == 2)
                bufferg.drawImage(img2, 102, 125, this);
            if (area_flag[5] == 2)
                bufferg.drawImage(img2, 164, 125, this);
            if (area_flag[6] == 2)
                bufferg.drawImage(img2, 42, 187, this);
            if (area_flag[7] == 2)
                bufferg.drawImage(img2, 102, 187, this);
            if (area_flag[8] == 2)
                bufferg.drawImage(img2, 164, 187, this);
        }

        bufferg.dispose();
        g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
    }
}
```

参考 Client11_5_user1.java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server11_5 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的消息，如图 11-19 所示。

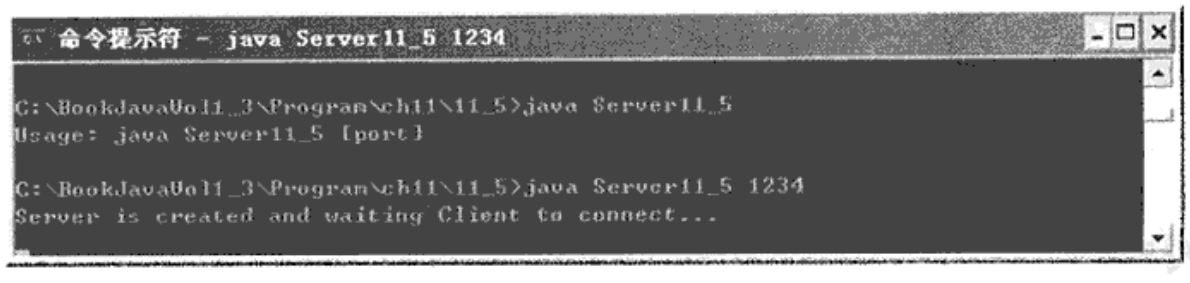


图 11-19

2 Client1 向 Server 报到。

在图 11—20 所示的“命令提示符”窗口中输入“java Client11_5_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 11—21 所示。

图 11-20

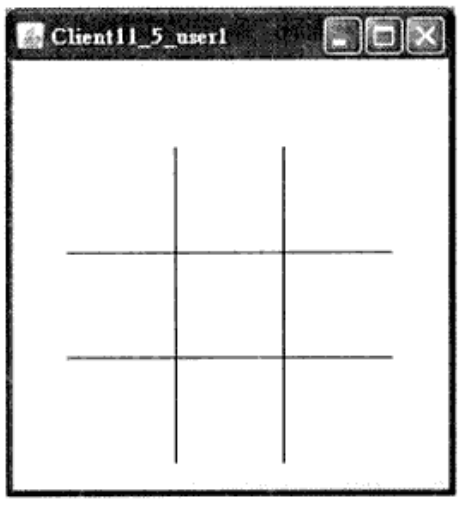


图 11-21

3 Client2 向 Server 报到。

在图 11—22 所示的“命令提示符”窗口中输入“java Client11_5_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 11—23 所示。

图 11-22

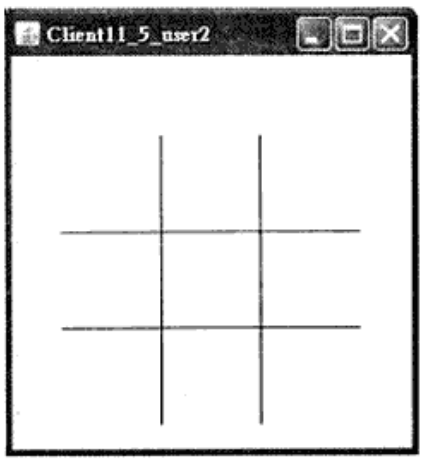


图 11-23

4 运行程序。

将消除上述各范例的闪烁画面，如图 11-24 所示。

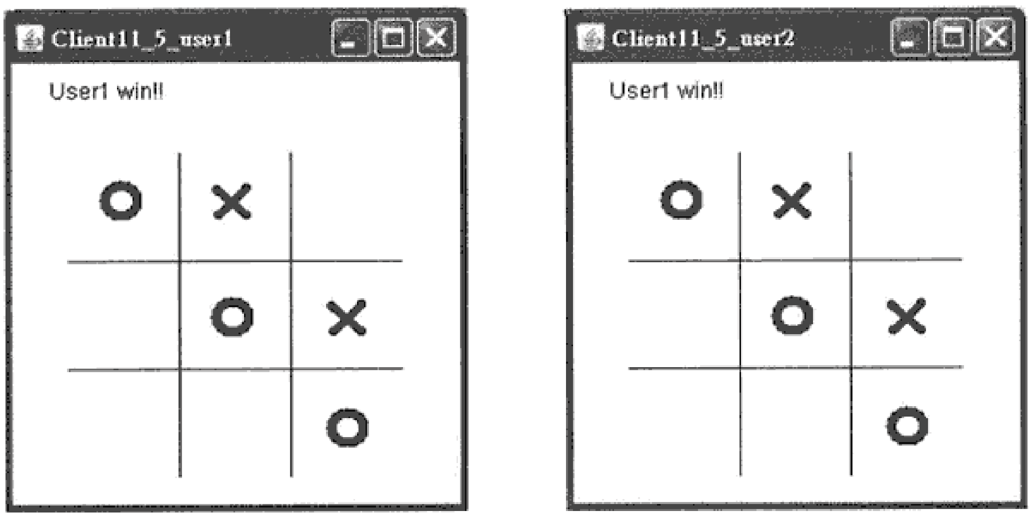


图 11-24

11-6 习题

1. 弈棋对阵同步图像的意义是什么？
2. 弈棋输赢评定与音效的意义是什么？
3. 请设计一个五子棋对阵游戏。

Chapter 12　在线射击对阵

12-1 简介

在线对阵游戏最重要的两个项目就是：弈棋对阵与射击对阵。读者如果能将这两种在线游戏悟透，那么其他任何对阵类的游戏均可轻易克服。在第 11 章我们已探讨了弈棋设计，本章将探讨在线射击对阵的设计。

射击对阵是动态的，操作频繁，如果要让对阵双方的窗口图像做同步移动，势必有大量的数据需要传递，不仅会造成设计上的困难，也会降低图像移动的灵活度。如第 11 章所述，为了解决此问题，对阵双方各自设计图像，不进行数据传递，仅传递移动命令流，这样可减少传递数据量，也可达到双方图像同步移动的效果。

射击对阵是动态的，比较困难，除了要考虑命令流的传递外，还必须考虑射击体、子弹、击中等问题。

12-2 网络命令流

在 Client1、Client2 上创建相同的射击体与子弹组件，以在线流将命令传递给对方，控制图像同步移动。其运行步骤如下：

(1) Client1 与 Client2 分别向 Server 报到。
(2) Client1 将本机的射击体与子弹移动命令通过 Server 传递给 Client2。
(3) Client2 将本机的射击体与子弹移动命令通过 Server 传递给 Client1。
(4) Client1 与 Client2 的画面同步更新。

范例 60　参考范例 45 和范例 55，文件 Server12_2.java、Client12_2_user1、Client12_2_user2 的功能是解释射击命令流与同步图像的应用。

文件 Server12_2.java：接收 Client1 送来的命令信息，再转发给 Client2，反之亦然。

```
001 import java.net.*;
002 import java.io.*;
003 import java.util.*;

004 public class Server12_2 {
005   private static ServerSocket SS;
```

```
006  private static int port;
007  private Hashtable ht = new Hashtable();
008  Socket socket;

009  public Server12_2() {
010    try {
011        SS = new ServerSocket(port);
012        System.out.println("Server is created and waiting Client to connect...");

013        while (true) {
014          socket = SS.accept();
015          System.out.println("Client IP = " +
                                  socket.getInetAddress().getHostAddress());
016          DataOutputStream outstream = new
                                  DataOutputStream(socket.getOutputStream());
017          ht.put(socket, outstream);
018          Thread thread = new Thread(new ServerRunnable(socket, ht));
019          thread.start();
020        }
021    }
022    catch (IOException e) {
023        System.out.println(e.getMessage());
024    }
025  }

026  public static void main(String[] args) {
027    if (args.length < 1) {
028      System.out.println("Usage: java Server12_2 [port]");
029      System.exit(1);
030    }

031    port=Integer.parseInt(args[0]) ;
032    Server12_2 ServerStart=new Server12_2();
033  }
034 }

035 class ServerRunnable implements Runnable {
036   private Socket socket;
037   private Hashtable ht;

038   public ServerRunnable(Socket socket, Hashtable ht) {
039     this.socket = socket;
040     this.ht = ht;
041   }

042   public void run() {
043     DataInputStream instream;

044     try {
045       instream = new DataInputStream(socket.getInputStream());

046       while (true) {
047         int message = instream.readInt();

048         synchronized(ht) {
049           Enumeration en = ht.elements();
050           while(en.hasMoreElements()) {
051             DataOutputStream outstream =
052               (DataOutputStream)en.nextElement();
```

```
053                         try {
054                             outstream.writeInt(message);
055                         }
056                         catch (IOException e) {
057                             System.out.println(e.getMessage());
058                         }
059                     }
060                 }
061         }
062     }
063     catch (IOException e) {
064             System.out.println(e.getMessage());
065       }
066       finally {
067             synchronized(ht) {
068                 System.out.println("Remove connection: " + socket);
069                 ht.remove(socket);
070                 try {
071                     socket.close();
072                 }
073                 catch (IOException e) {
074                 }
075             }
076       }
078   }
079 }
```

行 007	创建哈希表 ht。
行 027～031	以命令参数设置端口值（port）。
行 032	调用行 009～025 的构造函数。
行 011	创建 Server 的网站套接字 SS。
行 013～020	以 while 循环与各客户端创建各类关系。
行 014	当各客户端触发网络连接时，立即创建与各客户端的连接套接字 socket。
行 016	以各客户端的连接套接字 socket 创建各自的输出流 outstream。
行 017	将各客户端的 socket、outstream 存储在哈希表 ht 中。
行 018	创建各客户端的线程，以 ServerRunnable 类（行 035～079）生成的对象作为参数。
行 019	启动各线程。
行 035～079	创建实现 Runnable 接口的 ServerRunnable 类。
行 038	将行 014 的 socket、行 017 的 ht 作为参数传递给行 038 的构造函数 ServerRunnable()。
行 045	创建网络输入流 instream，等待发送端送来的信息。
行 048～060	利用关键字 synchronized 设置临界区，同一时刻仅允许一个线程运行。
行 050～060	使用哈希表中连接各客户端的网络流分别将数据信息传递给各客户端。

文件 Client12_2_user1.java：向 Server 报到，将命令信息传递给远程 User2，使射击同步移动。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;
```

```
086 public class Client12_2_user1 extends Frame implements Runnable {
087     int x=150, y=235, dx, dy;
088     int bx, by, dbx=0, dby=-5, bflag=0;
089     Image img;

090     Socket socket;
091     static String iaddr;
092     static int port;
093     DataOutputStream   outstream;
094     DataInputStream   instream;
095     int x_send, y_send, bx_send, by_send, Ibflag=0, bflag_send, rcv;

096     public static void main(String args[]) {
097         if (args.length < 2){
098            System.out.println("USAGE: java Client12_2_user1 [iaddr] [port]");
099            System.exit(1);
100         }

101         iaddr = args[0];
102         port=Integer.parseInt(args[1]);
103         Client12_2_user1 workStart=new Client12_2_user1();
104     }

105     public Client12_2_user1() {
106          super("Client12_2_user1");
107          setSize(350,350);

108          Toolkit tk = Toolkit.getDefaultToolkit();
109          img = tk.getImage("car090.gif");

110          enableEvents(AWTEvent.WINDOW_EVENT_MASK);
111          enableEvents(AWTEvent.KEY_EVENT_MASK);

112          setVisible(true);

113          try{
114               socket=new Socket(InetAddress.getByName(iaddr),port);
115               outstream = new DataOutputStream(socket.getOutputStream());
116               instream = new DataInputStream(socket.getInputStream());
117               new Thread(this).start();
118           }
119           catch (Exception e) {
120               e.printStackTrace();
121           }
122     }

123     public void processWindowEvent(WindowEvent e) {
124          if(e.getID() == WindowEvent.WINDOW_CLOSING) {
125             System.exit(0);
126           }
127     }

128     public void processKeyEvent(KeyEvent e) {
129          if(e.getID() == KeyEvent.KEY_PRESSED) {
130                switch(e.getKeyCode()) {
131                     case KeyEvent.VK_RIGHT:
                              dx = 5; dy = 0;
                              break;
132                     case KeyEvent.VK_LEFT:
```

```
                    dx = -5; dy = 0;
                    break;
                case KeyEvent.VK_UP:
                    dx = 0; dy = -5;
                    break;
                case KeyEvent.VK_DOWN:
                    dx = 0; dy = 5;
                    break;
                case KeyEvent.VK_SPACE:
                    dx = 0; dy = 0;
                    bx = x + 30;
                    by = y - 5;
                    Ibflag = 1;
                    break;
                default:
                    dx = 0; dy = 0;
            }
            try{
                x = x + dx;
                y = y + dy;

                x_send = x * 100 + 0;
                y_send = y * 100 + 1;
                bx_send = bx * 100 + 2;
                by_send = by * 100 + 3;
                bflag_send = Ibflag * 100 + 4;

                outstream.writeInt(x_send);
                outstream.writeInt(y_send);
                outstream.writeInt(bx_send);
                outstream.writeInt(by_send);
                outstream.writeInt(bflag_send);
            }
            catch (Exception f) {
                f.printStackTrace();
            }
        }
    }

    public void run() {
        while(true) {
            try {
                rcv = instream.readInt();

                if (rcv % 100 == 0)
                    x = (rcv - 0) / 100;
                else if (rcv % 100 == 1)
                    y = (rcv - 1) / 100;
                else if (rcv % 100 == 2)
                    bx = (rcv - 2) / 100;
                else if (rcv % 100 == 3)
                    by = (rcv - 3) / 100;
                else if (rcv % 100 == 4)
                    bflag = (rcv - 4) / 100;

                while(bflag==1) {
                    if(by <= -10) bflag = 0;
                    by = by + dby;

                    repaint();
```

```
170                     Thread.sleep(5);
171                   }
172                repaint();
173             }
174             catch (Exception f) {
175                 f.printStackTrace();
176             }
177         }
178     }

179     public void paint(Graphics g) {
180             g.drawImage(img, x, y, this);
181             g.fillRect(bx, by, 3, 5);
182     }
183 }
```

行 090～095　有关在线命令各变量的声明。

行 114　以服务器端的 IP、port 创建连接 Server 的套接字，用于驱动行 014。

行 115　创建网络输出流对象 outstream。

行 116　创建网络输入流对象 instream。

行 130～137　以键盘事件控制发射器图像的移动方向。按“→”键，图像向右移一步；按“←”键，图像向左移一步；按“↑”键，图像向上移一步；按“↓”键，图像向下移一步。按 Space 键时，发射子弹并设置 bflag 为 1。

行 141～145　以识别系数（a*100+n）封装(x，y）成在线命令信息(x_send，y_send)，其中 a 为原始值，n 为顺序值。如发射器 x_send = x*100+0，y_send=y*100+1；子弹 bx_send = bx * 100 + 2，by_send = by * 100 + 3，bflag_send = lbflag * 100 + 4。

行 146～150　以网络输出流对象 outstream 将命令信息(a_send）输出到网络。

行 160　由网络输入流对象 instream 读取从网络传来的在线命令信息，并存储在 rcv 中。

行 161～165　解封装在线命令信息的识别系数，如果 rcv%100 为 n，则 a=(rcv−n)/100，这样可求出发射器坐标(x，y)，子弹坐标(bx，by)，子弹标示（bflag）。

行 180～181　绘出发射器、子弹图像。

文件 Client12_2_user2.java：向 Server 报到，将命令信息传递给远程 User1，使射击同步移动。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

public class Client12_2_user2 extends Frame implements Runnable {
    int x=150, y=235, dx, dy;
    int bx, by, dbx=0, dby=-5, bflag=0;
    Image img;

    Socket socket;
    static String iaddr;
    static int port;
    DataOutputStream   outstream;
    DataInputStream   instream;
```

```
int x_send, y_send, bx_send, by_send, lbflag=0, bflag_send, rcv;

public static void main(String args[]) {
    if (args.length < 2){
      System.out.println("USAGE: java Client12_2_user2 [iaddr] [port]");
      System.exit(1);
    }

    iaddr = args[0];
    port=Integer.parseInt(args[1]);
  Client12_2_user2 workStart=new Client12_2_user2();
}

public Client12_2_user2() {
      super("Client12_2_user2");
      setSize(350,350);

      Toolkit tk = Toolkit.getDefaultToolkit();
      img = tk.getImage("car090.gif");

      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
      enableEvents(AWTEvent.KEY_EVENT_MASK);

      setVisible(true);
      try{
            socket=new Socket(InetAddress.getByName(iaddr),port);
            outstream = new DataOutputStream(socket.getOutputStream());
            instream = new DataInputStream(socket.getInputStream());
            new Thread(this).start();
       }
       catch (Exception e) {
            e.printStackTrace();
       }
}

public void processWindowEvent(WindowEvent e) {
      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
          System.exit(0);
       }
}

public void processKeyEvent(KeyEvent e) {
      if(e.getID() == KeyEvent.KEY_PRESSED) {
            switch(e.getKeyCode()) {
                  case KeyEvent.VK_RIGHT:
                        dx = 5; dy = 0;
                        break;
                  case KeyEvent.VK_LEFT:
                        dx = -5; dy = 0;
                        break;
                  case KeyEvent.VK_UP:
                        dx = 0; dy = -5;
                        break;
                  case KeyEvent.VK_DOWN:
                        dx = 0; dy = 5;
                        break;
                  case KeyEvent.VK_SPACE:
                        dx = 0; dy = 0;
                        bx = x + 30;
                        by = y - 5;
```

```
                    Ibflag = 1;
                    break;
                default:
                    dx = 0; dy = 0;
            }
            try{
                x = x + dx;
                y = y + dy;

                x_send = x * 100 + 0;
                y_send = y * 100 + 1;
                bx_send = bx * 100 + 2;
                by_send = by * 100 + 3;
                bflag_send = Ibflag * 100 + 4;

                outstream.writeInt(x_send);
                outstream.writeInt(y_send);
                outstream.writeInt(bx_send);
                outstream.writeInt(by_send);
                outstream.writeInt(bflag_send);
            }
            catch (Exception f) {
                f.printStackTrace();
            }
        }
    }

    public void run() {
      while(true) {
        try {
            rcv = instream.readInt();

            if (rcv % 100 == 0)
                x = (rcv - 0) / 100;
            else if (rcv % 100 == 1)
                y = (rcv - 1) / 100;
            else if (rcv % 100 == 2)
                bx = (rcv - 2) / 100;
            else if (rcv % 100 == 3)
                by = (rcv - 3) / 100;
            else if (rcv % 100 == 4)
                bflag = (rcv - 4) / 100;

            while(bflag==1) {
              if(by <= -10) bflag = 0;
              by = by + dby;

              repaint();
              Thread.sleep(5);
             }
          repaint();
        }
        catch (Exception f) {
            f.printStackTrace();
        }
      }
    }

    public void paint(Graphics g) {
        g.drawImage(img, x, y, this);
```

```
            g.fillRect(bx, by, 3, 5);
        }
    }
```

参考 Client12_2_user1.java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2 端（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server12_2 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 12—1 所示。

图 12-1

2 Client1 向 Server 报到。

在图 12—2 所示的“命令提示符”窗口中输入“java Client12_2_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 12—3 所示。

图 12-2

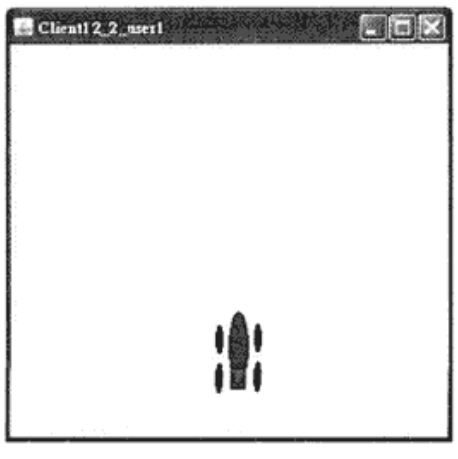


图 12-3

3 Client2 向 Server 报到。

在图 12-4 所示的“命令提示符”窗口中输入“java Client12_2_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 12-5 所示。

图 12-4

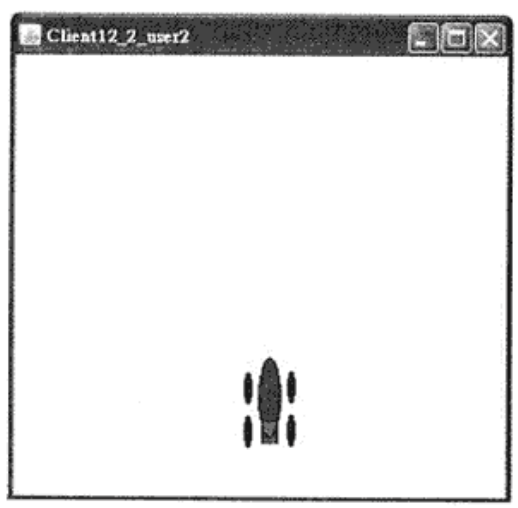


图 12-5

4 运行程序。

在 Client1 上按键盘键时，发射器图像立即随设置方向移动，同时远程 Client2 的图像也同步移动；反之亦然（按“→”键，图像向右移一步；按“←”键，图像向左移一步；按“↑”键，图像向上移一步；按“↓”键，图像向下移一步）。按 Space 键时发射子弹，如图 12-6 所示。

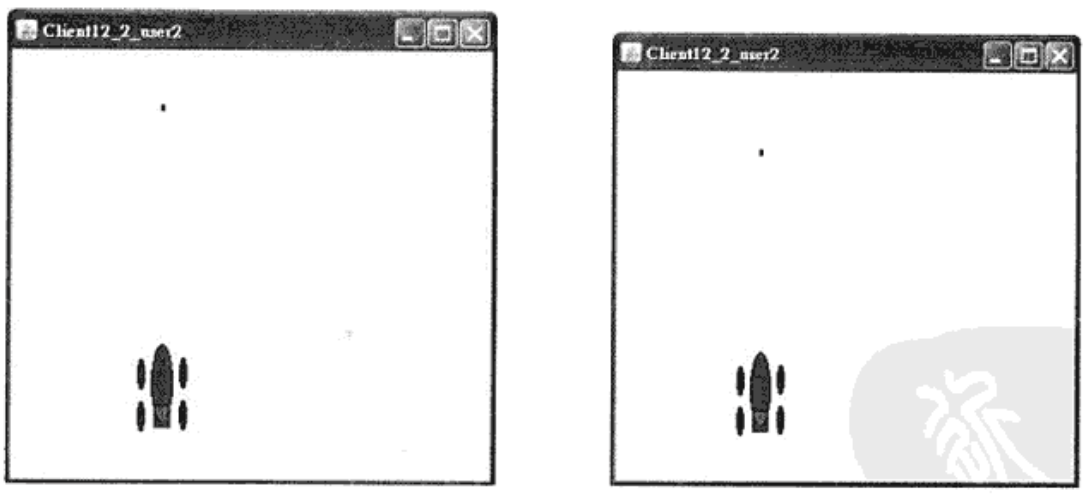


图 12-6

12-3 对阵同步图像

在 Client1 和 Client2 中创建相同的发射器组件，包括 User1 的绿色发射器与 User2 的红色发射器，以在线流将命令传递给对方，控制图像同步移动。其运行步骤如下：

(1) Client1 与 Client2 分别向 Server 报到。

(2) Client1 将绿色发射器与子弹的图像移动命令通过 Server 传递给 Client2。

(3) Client2 将红色发射器与子弹的图像移动命令通过 Server 传递给 Client1。

(4) Client1 与 Client2 对阵图像同步移动。

范例 61 参考范例 60，文件 Server12_3.java、Client12_3_user1.java、Client12_3_user2.java 的功能是解释射击命令流与对阵的应用。

文件 Server12_3.java：内容与 Server12_2.java 相同，请参考范例 60。

文件 Client12_3_user1.java：向 Server 报到，将绿色发射器与子弹的图像移动命令传递给远程 User2，也接收 User2 传来的对阵命令，使射击同步移动。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;

086 public class Client12_3_user1 extends Frame implements Runnable {
087     int x1=150, y1=235, dx1, dy1;
088     int x2=150, y2=45, dx2, dy2;
089     int bx1, by1, dbx1=0, dby1=-5, bflag1=0;
090     int bx2, by2, dbx2=0, dby2=5, bflag2=0;
091     Image img1, img2;

092     Socket socket;
093     static String iaddr;
094     static int port;
095     DataOutputStream   outstream;
096     DataInputStream   instream;
097     int x1_send, y1_send, bx1_send, by1_send, Ibflag1=0, bflag1_send, rcv;
098     int x2_send, y2_send, bx2_send, by2_send, blfag2_send;

099     public static void main(String args[]) {
100         if (args.length < 2){
101             System.out.println("USAGE: java Client12_3_user1 [iaddr] [port]");
102             System.exit(1);
103         }

104         iaddr = args[0];
105         port=Integer.parseInt(args[1]);
106       Client12_3_user1 workStart=new Client12_3_user1();
107     }

108     public Client12_3_user1() {
109         super("Client12_3_user1");
110         setSize(350,350);

111         Toolkit tk = Toolkit.getDefaultToolkit();
112         img1 = tk.getImage("car090.gif");
113         img2 = tk.getImage("car180.gif");

114         enableEvents(AWTEvent.WINDOW_EVENT_MASK);
115         enableEvents(AWTEvent.KEY_EVENT_MASK);

116         setVisible(true);
```

```
117         try{
118             socket=new Socket(InetAddress.getByName(iaddr),port);
119             outstream = new DataOutputStream(socket.getOutputStream());
120             instream = new DataInputStream(socket.getInputStream());
121             new Thread(this).start();
122          }
123          catch (Exception e) {
124             e.printStackTrace();
125          }
126     }

127     public void processWindowEvent(WindowEvent e) {
128         if(e.getID() == WindowEvent.WINDOW_CLOSING) {
129             System.exit(0);
130           }
131     }

132     public void processKeyEvent(KeyEvent e) {
133         if(e.getID() == KeyEvent.KEY_PRESSED) {
134                 switch(e.getKeyCode()) {
135                     case KeyEvent.VK_RIGHT:
                            dx1 = 5; dy1 = 0;
                            break;
136                     case KeyEvent.VK_LEFT:
                            dx1 = -5; dy1 = 0;
                            break;
137                     case KeyEvent.VK_UP:
                            dx1 = 0; dy1 = -5;
                            break;
138                     case KeyEvent.VK_DOWN:
                            dx1 = 0; dy1 = 5;
                            break;
139                     case KeyEvent.VK_SPACE:
                            dx1 = 0; dy1 = 0;
                            bx1 = x1 + 30;
                            by1 = y1 - 5;
                            Ibflag1 = 1;
                            break;
140                     default:
                            dx1 = 0; dy1 = 0;
141             }
142             try{
143                 x1 = x1 + dx1;
144                 y1 = y1 + dy1;

145                 x1_send = x1 * 100 + 0;
146                 y1_send = y1 * 100 + 1;
147                 bx1_send = bx1 * 100 + 2;
148                 by1_send = by1 * 100 + 3;
149                 bflag1_send = Ibflag1 * 100 + 4;

150                 outstream.writeInt(x1_send);
151                 outstream.writeInt(y1_send);
152                 outstream.writeInt(bx1_send);
153                 outstream.writeInt(by1_send);
154                 outstream.writeInt(bflag1_send);
155             }
156             catch (Exception f) {
157                 f.printStackTrace();
```

```
158                 }
159             }
160     }

161     public void run() {
162       while(true) {
163         try {
164                 rcv = instream.readInt();

165                 if (rcv % 100 == 0)
                        x1 = (rcv - 0) / 100;
166                 else if (rcv % 100 == 1)
                    y1 = (rcv - 1) / 100;
167                 else if (rcv % 100 == 2)
                        bx1 = (rcv - 2) / 100;
168                 else if (rcv % 100 == 3)
                        by1 = (rcv - 3) / 100;
169                 else if (rcv % 100 == 4)
                        bflag1 = (rcv - 4) / 100;

170                 if (rcv % 100 == 5)
                        x2 = (rcv - 5) / 100;
171                 else if (rcv % 100 == 6)
                        y2 = (rcv - 6) / 100;
172                 else if (rcv % 100 == 7)
                        bx2 = (rcv - 7) / 100;
173                 else if (rcv % 100 == 8)
                        by2 = (rcv - 8) / 100;
174                 else if (rcv % 100 == 9)
                        bflag2 = (rcv - 9) / 100;

175                 while((bflag1==1)&&(bflag2==0)) {
                      if(by1 <= -10) bflag1 = 0;
                      by1 = by1 + dby1;
                      repaint();
                      Thread.sleep(5);
                     }

176                 while((bflag1==0)&&(bflag2==1)) {
                      if(by2 >= 360) bflag2 = 0;
                      by2 = by2 + dby2;
                      repaint();
                      Thread.sleep(5);
                     }

177                 while((bflag1==1)&&(bflag2==1)) {
                      if(by1 <= -10) bflag1 = 0;
                      if(by2 >= 360) bflag2 = 0;
                      by1 = by1 + dby1;
                      by2 = by2 + dby2;
                      repaint();
                      Thread.sleep(5);
                     }

178             repaint();
179         }
180         catch (Exception f) {
181             f.printStackTrace();
182         }
183     }
```

```
184     }

185     public void paint(Graphics g) {
186             g.drawImage(img1, x1, y1, this);
187             g.drawImage(img2, x2, y2, this);
188             g.fillRect(bx1, by1, 3, 5);
189             g.fillRect(bx2, by2, 3, 5);
190     }
191 }
```

行 112　读取 User1 的绿色发射器图像文件。
行 113　读取 User2 的红色发射器图像文件。
行 170～174　读取 User2 的网络命令，解封装识别系数，包括红色发射器与子弹的坐标。
行 176～177　当 User2 的子弹标示（bflag2）为 1 时，改变其子弹的移动坐标。
行 187　绘制 User2 的红色发射器图像。
行 189　绘制 User2 的子弹图像。

文件 Client12_3_user2.java：向 Server 报到，将红色发射器与子弹的图像移动命令传递给远程 User1，也接收 User1 传来的命令，使射击同步移动。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

public class Client12_3_user2 extends Frame implements Runnable {
    int x1=150, y1=235, dx1, dy1;
    int x2=150, y2=45, dx2, dy2;
    int bx1, by1, dbx1=0, dby1=-5, bflag1=0;
    int bx2, by2, dbx2=0, dby2=5, bflag2=0;
    Image img1, img2;

    Socket socket;
    static String iaddr;
    static int port;
    DataOutputStream   outstream;
    DataInputStream   instream;
    int x1_send, y1_send, bx1_send, by1_send, bflag1_send, rcv;
    int x2_send, y2_send, bx2_send, by2_send, Ibflag2=0, bflag2_send;

    public static void main(String args[]) {
        if (args.length < 2){
           System.out.println("USAGE: java Client12_3_user2 [iaddr] [port]");
           System.exit(1);
        }

        iaddr = args[0];
        port=Integer.parseInt(args[1]);
      Client12_3_user2 workStart=new Client12_3_user2();
    }

    public Client12_3_user2() {
         super("Client12_3_user2");
```

```
        setSize(350,350);

        Toolkit tk = Toolkit.getDefaultToolkit();
        img1 = tk.getImage("car090.gif");
        img2 = tk.getImage("car180.gif");

        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        enableEvents(AWTEvent.KEY_EVENT_MASK);

        setVisible(true);

        try{
            socket=new Socket(InetAddress.getByName(iaddr),port);
            outstream = new DataOutputStream(socket.getOutputStream());
            instream = new DataInputStream(socket.getInputStream());
            new Thread(this).start();
         }
         catch (Exception e) {
            e.printStackTrace();
         }
}

public void processWindowEvent(WindowEvent e) {
      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
          System.exit(0);
        }
}

public void processKeyEvent(KeyEvent e) {
      if(e.getID() == KeyEvent.KEY_PRESSED) {
            switch(e.getKeyCode()) {
                  case KeyEvent.VK_RIGHT:
                        dx2 = 5; dy2 = 0;
                        break;
            case KeyEvent.VK_LEFT:
                        dx2 = -5; dy2 = 0;
                        break;
            case KeyEvent.VK_UP:
                        dx2 = 0; dy2 = -5;
                        break;
            case KeyEvent.VK_DOWN:
                        dx2 = 0; dy2 = 5;
                        break;
            case KeyEvent.VK_SPACE:
                        dx2 = 0; dy2 = 0;
                        bx2 = x2 + 30;
                        by2 = y2 - 5;
                        Ibflag2 = 1;
                        break;
            default:
                        dx2 = 0; dy2 = 0;
}
try{
    x2 = x2 + dx2;
    y2 = y2 + dy2;

    x2_send = x2 * 100 + 5;
    y2_send = y2 * 100 + 6;
    bx2_send = bx2 * 100 + 7;
    by2_send = by2 * 100 + 8;
```

```
                bflag2_send = Ibflag2 * 100 + 9;

                outstream.writeInt(x2_send);
                outstream.writeInt(y2_send);
                outstream.writeInt(bx2_send);
                outstream.writeInt(by2_send);
                outstream.writeInt(bflag2_send);
            }
            catch (Exception f) {
                f.printStackTrace();
            }
        }
    }

    public void run() {
      while(true) {
        try {
            rcv = instream.readInt();

            if (rcv % 100 == 0)
                x1 = (rcv - 0) / 100;
            else if (rcv % 100 == 1)
                y1 = (rcv - 1) / 100;
            else if (rcv % 100 == 2)
                bx1 = (rcv - 2) / 100;
            else if (rcv % 100 == 3)
                by1 = (rcv - 3) / 100;
            else if (rcv % 100 == 4)
                bflag1 = (rcv - 4) / 100;

            if (rcv % 100 == 5)
                x2 = (rcv - 5) / 100;
            else if (rcv % 100 == 6)
                y2 = (rcv - 6) / 100;
            else if (rcv % 100 == 7)
                bx2 = (rcv - 7) / 100;
            else if (rcv % 100 == 8)
                by2 = (rcv - 8) / 100;
            else if (rcv % 100 == 9)
                bflag2 = (rcv - 9) / 100;

            while((bflag1==1)&&(bflag2==0)) {
              if(by1 <= -10) bflag1 = 0;
              by1 = by1 + dby1;
              repaint();
              Thread.sleep(5);
             }

            while((bflag1==0)&&(bflag2==1)) {
              if(by2 >= 360) bflag2 = 0;
              by2 = by2 + dby2;
              repaint();
              Thread.sleep(5);
             }

            while((bflag1==1)&&(bflag2==1)) {
              if(by1 <= -10) bflag1 = 0;
              if(by2 >= 360) bflag2 = 0;
              by1 = by1 + dby1;
              by2 = by2 + dby2;
```

```
                repaint();
                Thread.sleep(5);
              }

            repaint();
          }
          catch (Exception f) {
            f.printStackTrace();
          }
        }
      }

      public void paint(Graphics g) {
          g.drawImage(img1, x1, y1, this);
          g.drawImage(img2, x2, y2, this);
          g.fillRect(bx1, by1, 3, 5);
          g.fillRect(bx2, by2, 3, 5);
      }
}
```

参考 Client12_3_user1 .java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2 端（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server12_3 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 12-7 所示。

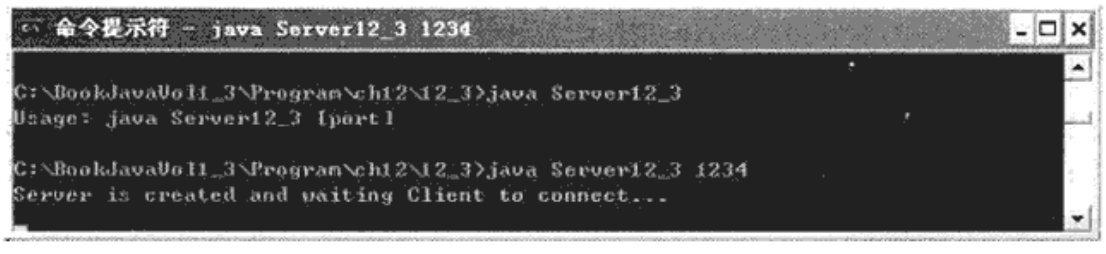


图 12-7

2 Client1 向 Server 报到。

在图 12-8 所示的“命令提示符”窗口中输入“java Client12_3_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 12-9 所示。

图 12-8

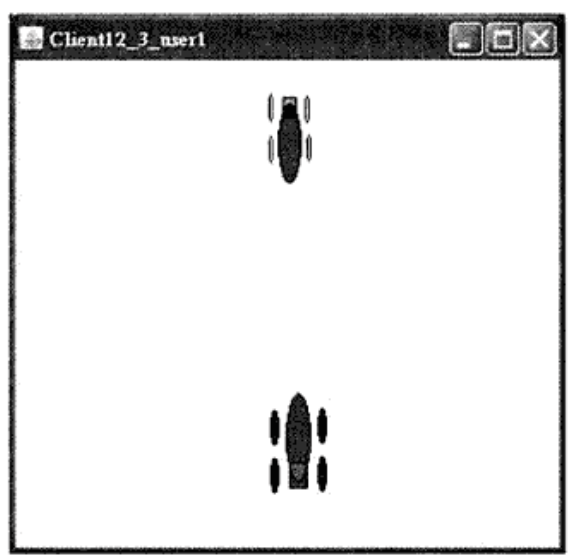


图 12-9

3 Client2 向 Server 报到。

在图 12-10 所示的“命令提示符”窗口中输入“java Client12_3_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 12-11 所示。

图 12-10

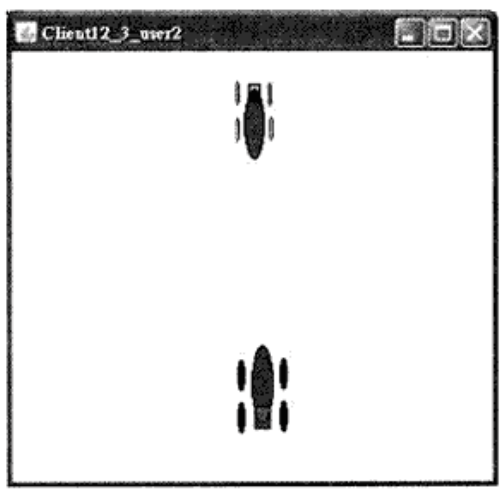


图 12-11

4 运行程序。

在 Client1 上按键盘键时，绿色发射器图像立即随设置方向移动，同时远程 Client2 的绿色发射

器图像也同步移动；在 Client2 上按键盘键时，红色发射器图像立即随设置方向移动，同时远程 Client1 的红色发射器图像也同步移动（按“→”键，图像向右移一步；按“←”键，图像向左移一步；按“↑”键，图像向上移一步；按“↓”键，图像向下移一步）。按 Space 键时发射子弹，如图 12-12 所示。

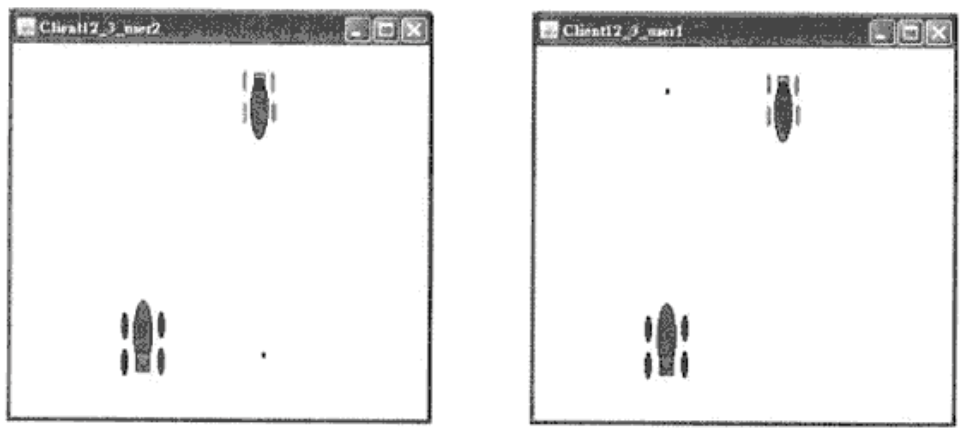


图 12-12

12-4 输赢评定与音效

按“←”、“→”、“↑”、“↓”键时，分别可以向左、右、上、下移动发射器，从而实现闪躲与追击；当按 Space 键时发射子弹并播放发射音效；当子弹击中对方发射器时，将对方换成爆炸图像，播放爆炸音效并显示胜利信息。

范例 62 参考范例 53 和范例 61，文件 Server12_4.java、Client12_4_user1.java、Client12_4_user2.java 的功能是解释射击输赢评定与音效的应用。

文件 Server12_4.java：内容与 Server12_2.java 相同，请参考范例 60。

文件 Client12_4_user1.java：向 Server 报到，按“←”、“→”、“↑”、“↓”键时，分别可以向左、右、上、下移动发射器；按 Space 键时发射子弹并播放发射音效；当子弹击中对方发射器时，将对方换成爆炸图像、播放爆炸音效并显示胜利信息。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;

086 import java.applet.*;

087 public class Client12_4_user1 extends Frame implements Runnable {
088   int x1=150, y1=235, dx1, dy1;
089   int x2=150, y2=45, dx2, dy2;
090   int bx1, by1, dbx1=0, dby1=-5, bflag1=0;
091   int bx2, by2, dbx2=0, dby2=5, bflag2=0;
092   Image img1, img2, img3, img4;

093   Socket socket;
094   static String iaddr;
095   static int port;
096   DataOutputStream   outstream;
```

```
097  DataInputStream   instream;
098  int x1_send, y1_send, bx1_send, by1_send, Ibflag1=0, bflag1_send, rcv;
099  int x2_send, y2_send, bx2_send, by2_send, blfag2_send;

100  int hitflag1=0, hitflag2=0;
101  AudioClip sound1, sound2, soundexplode;
102  String message="";

103  public static void main(String args[]) {
104     if (args.length < 2){
105        System.out.println("USAGE: java Client12_4_user1 [iaddr] [port]");
106        System.exit(1);
107     }

108     iaddr = args[0];
109     port=Integer.parseInt(args[1]);
110     Client12_4_user1 workStart=new Client12_4_user1();
111  }

112  public Client12_4_user1() {
113   super("Client12_4_user1");
114   setSize(350,350);

115   Toolkit tk = Toolkit.getDefaultToolkit();
116   img1 = tk.getImage("car090.gif");
117   img2 = tk.getImage("car180.gif");
118   img3 = tk.getImage("hit1.gif");
119   img4 = tk.getImage("hit2.gif");

120   sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
121   sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
122   soundexplode = Applet.newAudioClip(getClass().getResource("soundexplode.au"));
123   enableEvents(AWTEvent.WINDOW_EVENT_MASK);
124   enableEvents(AWTEvent.KEY_EVENT_MASK);

125   setVisible(true);

126   try{
127        socket=new Socket(InetAddress.getByName(iaddr),port);
128        outstream = new DataOutputStream(socket.getOutputStream());
129        instream = new DataInputStream(socket.getInputStream());
130        new Thread(this).start();
131   }
132   catch (Exception e) {
133        e.printStackTrace();
134   }
135  }

136  public void processWindowEvent(WindowEvent e) {
137     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
138        System.exit(0);
139     }
140  }

141  public void processKeyEvent(KeyEvent e) {
142     if(e.getID() == KeyEvent.KEY_PRESSED) {
143        switch(e.getKeyCode()) {
144              case KeyEvent.VK_RIGHT:
                    dx1 = 5; dy1 = 0;
                    break;
```

```
145                 case KeyEvent.VK_LEFT:
                        dx1 = -5; dy1 = 0;
                        break;
146                 case KeyEvent.VK_UP:
                        dx1 = 0; dy1 = -5;
                        break;
147                 case KeyEvent.VK_DOWN:
                        dx1 = 0; dy1 = 5;
                        break;
148                 case KeyEvent.VK_SPACE:
                        dx1 = 0; dy1 = 0;
                        bx1 = x1 + 30;
                        by1 = y1 - 5;
                        Ibflag1 = 1;
                        sound1.play();
                        break;
149                  default:
                        dx1 = 0; dy1 = 0;
150         }
151         try{
152             x1 = x1 + dx1;
153             y1 = y1 + dy1;

154             x1_send = x1 * 100 + 0;
155             y1_send = y1 * 100 + 1;
156             bx1_send = bx1 * 100 + 2;
157             by1_send = by1 * 100 + 3;
158             bflag1_send = Ibflag1 * 100 + 4;

159             outstream.writeInt(x1_send);
160             outstream.writeInt(y1_send);
161             outstream.writeInt(bx1_send);
162             outstream.writeInt(by1_send);
163             outstream.writeInt(bflag1_send);
164         }
165         catch (Exception f) {
166             f.printStackTrace();
167         }
168     }
169 }

170 public void run() {
171     while(true) {
172         try {
173             rcv = instream.readInt();

174             if (rcv % 100 == 0)
                    x1 = (rcv - 0) / 100;
175             else if (rcv % 100 == 1)
                    y1 = (rcv - 1) / 100;
176             else if (rcv % 100 == 2)
                    bx1 = (rcv - 2) / 100;
177             else if (rcv % 100 == 3)
                    by1 = (rcv - 3) / 100;
178             else if (rcv % 100 == 4)
                    bflag1 = (rcv - 4) / 100;

179             if (rcv % 100 == 5)
                    x2 = (rcv - 5) / 100;
180             else if (rcv % 100 == 6)
```

```
            y2 = (rcv - 6) / 100;
181         else if (rcv % 100 == 7)
            bx2 = (rcv - 7) / 100;
182         else if (rcv % 100 == 8)
            by2 = (rcv - 8) / 100;
183         else if (rcv % 100 == 9)
            bflag2 = (rcv - 9) / 100;

184         while((bflag1==1)&&(bflag2==0)) {
185           if(by1 <= -10) bflag1 = 0;
186           by1 = by1 + dby1;
187           if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message=="")){
                  hitflag2=1;
                  soundexplode.play();
                  message="User1 Win!!";
188           }
189           repaint();
190           Thread.sleep(5);
191         }

192         while((bflag1==0)&&(bflag2==1)) {
193           if(by2 >= 360) bflag2 = 0;
194           by2 = by2 + dby2;
195           if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message=="")){
                  hitflag1=1;
                  soundexplode.play();
                  message="User2 Win!!";
196           }
197           repaint();
198           Thread.sleep(5);
199         }

200           while((bflag1==1)&&(bflag2==1)) {
201             if(by1 <= -10) bflag1 = 0;
202             if(by2 >= 360) bflag2 = 0;
203             by1 = by1 + dby1;
204             by2 = by2 + dby2;
205             if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message=="")){
                    hitflag2=1;
                    soundexplode.play();
                    message="Both died!!";
206             }
207             if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message=="")){
                    hitflag1=1;
                    soundexplode.play();
                    message=""Both died!!"";
208             }
209             repaint();
210             Thread.sleep(5);
211           }

212         repaint();
213       }
214       catch (Exception f) {
215         f.printStackTrace();
216       }
217     }
218   }

219   public void paint(Graphics g) {
```

```
220      g.drawImage(img1, x1, y1, this);
221      g.drawImage(img2, x2, y2, this);
222      g.fillRect(bx1, by1, 3, 5);
223      g.fillRect(bx2, by2, 3, 5);

224      if (hitflag1==1) img1=img3;
225      if (hitflag2==1) img2=img4;

226      g.drawString(message, 20, 45);
227   }
228 }
```

行 086	由系统导入 java.applet 包（用于音效处理）。
行 092	声明击中图像变量。
行 100	声明击中标示。
行 101	声明音效变量。
行 102	声明字符串变量。
行 118～119	读取击中图像。
行 120～122	读取声音文件。
行 148	按 Space 键时射出子弹并播放音效。
行 187～188	如果 User1 的子弹进入对方红色发射器范围，且尚未显示中弹信息，则设置击中标示，播放击中音效，设置 "User1 Win!!" 信息。
行 195～196	如果 User2 的子弹进入对方绿色发射器范围，且尚未显示中弹信息，则设置击中标示，播放击中音效，设置 "User2 Win!!" 信息。
行 205～208	如果对阵双方的子弹同时进入对方发射器范围，且尚未显示中弹信息，则设置击中标示，播放击中音效，设置 "Both died!!" 信息。
行 224～225	如果击中标示为 1，则将该标示所属的发射器图像改成爆炸图像。
行 226	显示输赢信息。

文件 Client12_4_user2.java：参考 Client12_4_user1.java。

```
import java.awt.*;
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

import java.applet.*;

public class Client12_4_user2 extends Frame implements Runnable {
    int x1=150, y1=235, dx1, dy1;
    int x2=150, y2=45, dx2, dy2;
    int bx1, by1, dbx1=0, dby1=-5, bflag1=0;
    int bx2, by2, dbx2=0, dby2=5, bflag2=0;
    Image img1, img2, img3, img4;

    Socket socket;
    static String iaddr;
    static int port;
    DataOutputStream   outstream;
```

```
DataInputStream   instream;
int x1_send, y1_send, bx1_send, by1_send, bflag1_send, rcv;
int x2_send, y2_send, bx2_send, by2_send, Ibflag2=0, bflag2_send;

int hitflag1=0, hitflag2=0;
AudioClip sound1, sound2, soundexplode;
String message="";

public static void main(String args[]) {
   if (args.length < 2){
      System.out.println("USAGE: java Client12_4_user2 [iaddr] [port]");
      System.exit(1);
   }

   iaddr = args[0];
   port=Integer.parseInt(args[1]);
   Client12_4_user2 workStart=new Client12_4_user2();
}

public Client12_4_user2() {
   super("Client12_4_user2");
   setSize(350,350);

   Toolkit tk = Toolkit.getDefaultToolkit();
   img1 = tk.getImage("car090.gif");
   img2 = tk.getImage("car180.gif");
   img3 = tk.getImage("hit1.gif");
   img4 = tk.getImage("hit2.gif");

   sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
   sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
   soundexplode = Applet.newAudioClip(getClass().getResource("soundexplode.au"));

   enableEvents(AWTEvent.WINDOW_EVENT_MASK);
   enableEvents(AWTEvent.KEY_EVENT_MASK);

   setVisible(true);

   try{
        socket=new Socket(InetAddress.getByName(iaddr),port);
        outstream = new DataOutputStream(socket.getOutputStream());
        instream = new DataInputStream(socket.getInputStream());
        new Thread(this).start();
    }
    catch (Exception e) {
        e.printStackTrace();
    }
}

public void processWindowEvent(WindowEvent e) {
     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
         System.exit(0);
        }
}

public void processKeyEvent(KeyEvent e) {
     if(e.getID() == KeyEvent.KEY_PRESSED) {
            switch(e.getKeyCode()) {
                 case KeyEvent.VK_RIGHT:
                        dx2 = 5; dy2 = 0;
```

```
                    break;
                case KeyEvent.VK_LEFT:
                    dx2 = -5; dy2 = 0;
                    break;
                case KeyEvent.VK_UP:
                    dx2 = 0; dy2 = -5;
                    break;
                case KeyEvent.VK_DOWN:
                    dx2 = 0; dy2 = 5;
                    break;
                case KeyEvent.VK_SPACE:
                    dx2 = 0; dy2 = 0;
                    bx2 = x2 + 30;
                    by2 = y2 - 5;
                    Ibflag2 = 1;
                    sound2.play();
                    break;
                default:
                    dx2 = 0; dy2 = 0;
            }
            try{
                x2 = x2 + dx2;
                y2 = y2 + dy2;

                x2_send = x2 * 100 + 5;
                y2_send = y2 * 100 + 6;
                bx2_send = bx2 * 100 + 7;
                by2_send = by2 * 100 + 8;
                bflag2_send = Ibflag2 * 100 + 9;

                outstream.writeInt(x2_send);
                outstream.writeInt(y2_send);
                outstream.writeInt(bx2_send);
                outstream.writeInt(by2_send);
                outstream.writeInt(bflag2_send);
            }
            catch (Exception f) {
                f.printStackTrace();
            }
        }
    }

    public void run() {
      while(true) {
        try {
            rcv = instream.readInt();

            if (rcv % 100 == 0)
                x1 = (rcv - 0) / 100;
            else if (rcv % 100 == 1)
                y1 = (rcv - 1) / 100;
            else if (rcv % 100 == 2)
                bx1 = (rcv - 2) / 100;
            else if (rcv % 100 == 3)
                by1 = (rcv - 3) / 100;
            else if (rcv % 100 == 4)
                bflag1 = (rcv - 4) / 100;

            if (rcv % 100 == 5)
                x2 = (rcv - 5) / 100;
```

```
else if (rcv % 100 == 6)
    y2 = (rcv - 6) / 100;
else if (rcv % 100 == 7)
    bx2 = (rcv - 7) / 100;
else if (rcv % 100 == 8)
    by2 = (rcv - 8) / 100;
else if (rcv % 100 == 9)
    bflag2 = (rcv - 9) / 100;

while((bflag1==1)&&(bflag2==0)) {
  if(by1 <= -10) bflag1 = 0;
  by1 = by1 + dby1;
  if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message==""){
        hitflag2=1;
        soundexplode.play();
        message="User1 Win!!";
  }
  repaint();
  Thread.sleep(5);
 }

while((bflag1==0)&&(bflag2==1)) {
  if(by2 >= 360) bflag2 = 0;
  by2 = by2 + dby2;
  if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message==""){
        hitflag1=1;
        soundexplode.play();
        message="User2 Win!!";
  }
  repaint();
  Thread.sleep(5);
 }

            while((bflag1==1)&&(bflag2==1)) {
              if(by1 <= -10) bflag1 = 0;
              if(by2 >= 360) bflag2 = 0;
              by1 = by1 + dby1;
              by2 = by2 + dby2;
              if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message==""){
                    hitflag2=1;
                    soundexplode.play();
                    message="Both died!!";
              }
              if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message==""){
                    hitflag1=1;
                    soundexplode.play();
                    message="Both died!!";
              }
              repaint();
              Thread.sleep(5);
            }

        repaint();
      }
      catch (Exception f) {
          f.printStackTrace();
      }
  }
}
```

```
    public void paint(Graphics g) {
            g.drawImage(img1, x1, y1, this);
            g.drawImage(img2, x2, y2, this);
            g.fillRect(bx1, by1, 3, 5);
            g.fillRect(bx2, by2, 3, 5);

        if (hitflag1==1) img1=img3;
        if (hitflag2==1) img2=img4;

        g.drawString(message, 20, 45);
    }
}
```

参考 Client12_4_user1.java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server12_4 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 12-13 所示。

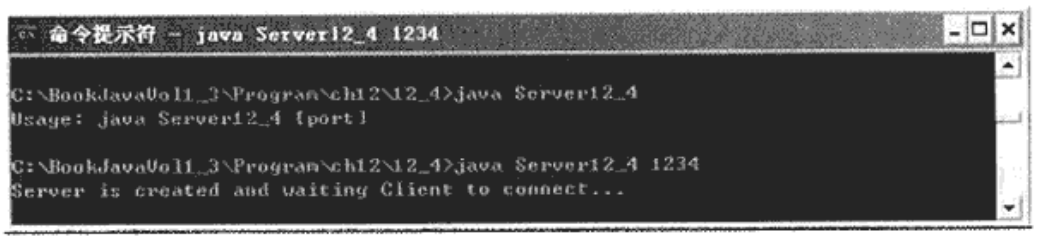


图 12-13

2 Client1 向 Server 报到。

在图 12-14 所示的“命令提示符”窗口中输入“java Client12_4_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 12-15 所示。

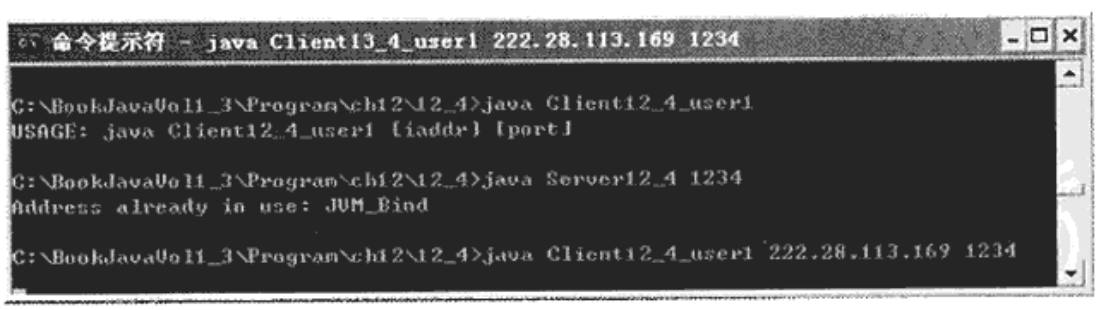


图 12-14

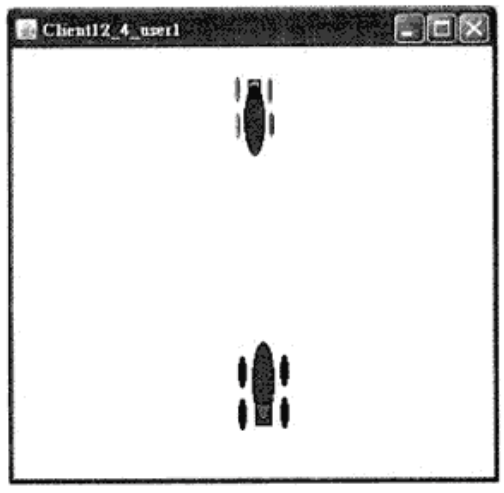


图 12-15

3 Client2 向 Server 报到。

在图 12—16 所示的“命令提示符”窗口中输入“java Client12_4_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 12—17 所示。

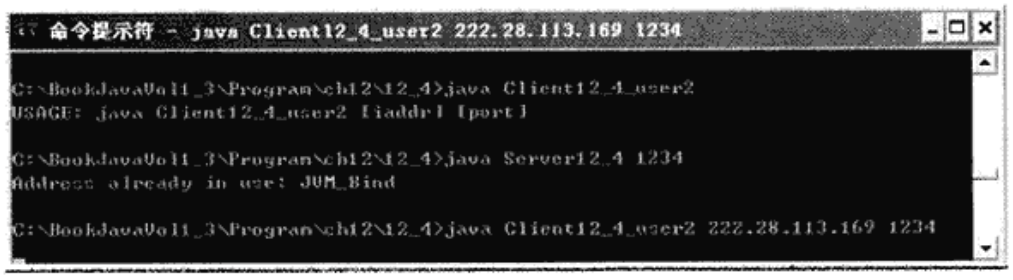


图 12-16

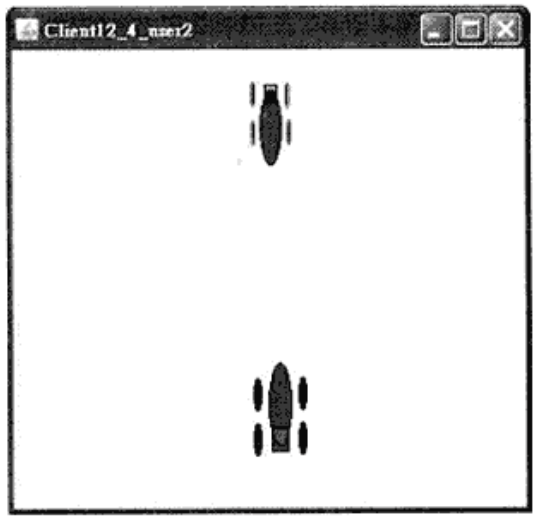


图 12-17

4 运行程序。

User1 按 Space 键时发射子弹，当子弹进入红色发射器范围且尚未显示中弹信息时，则将红色发射器图像改成爆炸图像，播放击中音效，显示“User1 Win!!”信息，如图 12—18 所示。

User2 按 Space 键时发射子弹，当子弹进入绿色发射器范围且尚未显示中弹信息时，则将绿色发射器图像改成爆炸图像，播放击中音效，显示“User2 Win!!”信息，如图 12—19 所示。

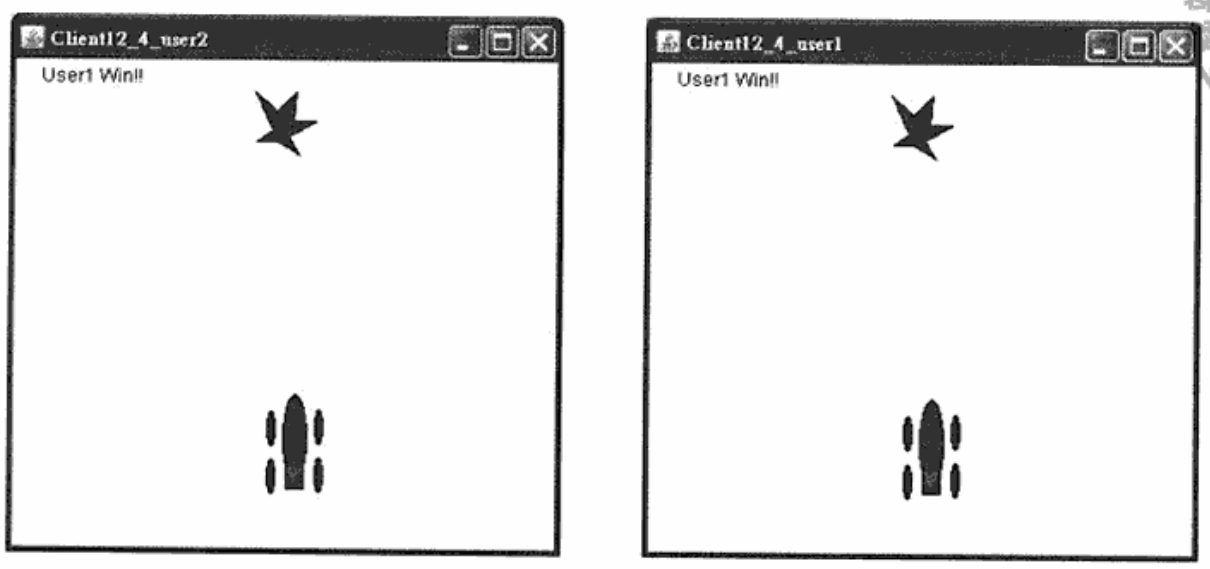


图 12-18

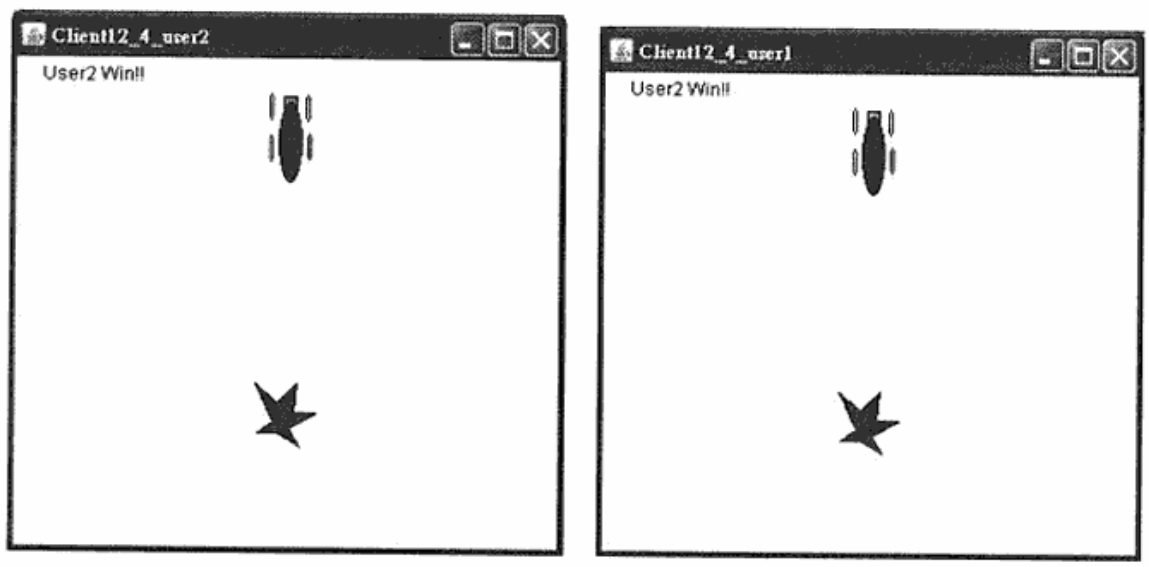


图 12-19

12-5 消除闪烁

如第 8 章所述，因为新图像随时替换旧图像，如果替换得太快，有些部分先到位，有些部分后到位，图像将会显得闪烁而不稳定。为了消除图像闪烁，我们可设置一个缓冲页（Buffer Page），将新图像先置入缓冲页，等待图像各部分全部绘制完成后，再将缓冲页显示在窗口中，这样就可消除图像的闪烁。

范例 63 参考范例 49 和范例 62，文件 Server12_5.java、Client12_5_user1.java、Client12_5_user2.java 的功能是解释消除射击闪烁的应用。

文件Server12_5.java：内容与Server12_2.java相同，请参考范例60。

文件 Client12_5_user1.java：作用是消除 User1 端的射击图像闪烁。

```
080 import java.awt.*;
081 import java.awt.event.*;
082 import java.math.*;

083 import java.io.*;
084 import java.net.*;
085 import java.util.*;
```

```
086 import java.applet.*;

087 public class Client12_5_user1 extends Frame implements Runnable {
088   int x1=150, y1=225, dx1, dy1;
089   int x2=150, y2=35, dx2, dy2;
090   int bx1, by1, dbx1=0, dby1=-5, bflag1=0;
091   int bx2, by2, dbx2=0, dby2=5, bflag2=0;
092   Image img1, img2, img3, img4, bufferPage=null;

093   Socket socket;
094   static String iaddr;
095   static int port;
096   DataOutputStream   outstream;
097   DataInputStream   instream;
098   int x1_send, y1_send, bx1_send, by1_send, Ibflag1=0, bflag1_send, rcv;
099   int x2_send, y2_send, bx2_send, by2_send, blfag2_send;

100   int hitflag1=0, hitflag2=0;
101   AudioClip sound1, sound2, soundexplode;
102   String message="";

103   public static void main(String args[]) {
104     if (args.length < 2){
105     System.out.println("USAGE: java Client12_5_user1 [iaddr] [port]");
106     System.exit(1);
107    }

108    iaddr = args[0];
109    port=Integer.parseInt(args[1]);
110    Client12_5_user1 workStart=new Client12_5_user1();
111  }

112   public Client12_5_user1() {
113     super("Client12_5_user1");
114     setSize(350,350);

115     Toolkit tk = Toolkit.getDefaultToolkit();
116     img1 = tk.getImage("car090.gif");
117     img2 = tk.getImage("car180.gif");
118     img3 = tk.getImage("hit1.gif");
119     img4 = tk.getImage("hit2.gif");

120     sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
121     sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
122     soundexplode = Applet.newAudioClip(getClass().getResource("soundexplode.au"));

123     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
124     enableEvents(AWTEvent.KEY_EVENT_MASK);

125     setVisible(true);

126     try{
127           socket=new Socket(InetAddress.getByName(iaddr),port);
128           outstream = new DataOutputStream(socket.getOutputStream());
129           instream = new DataInputStream(socket.getInputStream());
130           new Thread(this).start();
131     }
132     catch (Exception e) {
133           e.printStackTrace();
134     }
135  }

136  public void processWindowEvent(WindowEvent e) {
```

```
137     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
138       System.exit(0);
139     }
140   }

141   public void processKeyEvent(KeyEvent e) {
142     if(e.getID() == KeyEvent.KEY_PRESSED) {
143       switch(e.getKeyCode()) {
144             case KeyEvent.VK_RIGHT:
                    dx1 = 5; dy1 = 0;
                    break;
145             case KeyEvent.VK_LEFT:
                    dx1 = -5; dy1 = 0;
                    break;
146             case KeyEvent.VK_UP:
                    dx1 = 0; dy1 = -5;
                    break;
147             case KeyEvent.VK_DOWN:
                    dx1 = 0; dy1 = 5;
                    break;
148             case KeyEvent.VK_SPACE:
                    dx1 = 0; dy1 = 0;
                    bx1 = x1 + 30;
                    by1 = y1 - 5;
                    Ibflag1 = 1;
                    sound1.play();
                    break;
149             default:
                    dx1 = 0; dy1 = 0;
150       }
151       try{
152           x1 = x1 + dx1;
153           y1 = y1 + dy1;
154           x1_send = x1 * 100 + 0;
155           y1_send = y1 * 100 + 1;
156           bx1_send = bx1 * 100 + 2;
157           by1_send = by1 * 100 + 3;
158           bflag1_send = Ibflag1 * 100 + 4;

159           outstream.writeInt(x1_send);
160           outstream.writeInt(y1_send);
161           outstream.writeInt(bx1_send);
162           outstream.writeInt(by1_send);
163           outstream.writeInt(bflag1_send);
164       }
165       catch (Exception f) {
166           f.printStackTrace();
167       }
168     }
169   }

170   public void run() {
171     while(true) {
172       try {
173             rcv = instream.readInt();

174             if (rcv % 100 == 0)
                    x1 = (rcv - 0) / 100;
175             else if (rcv % 100 == 1)
                    y1 = (rcv - 1) / 100;
176             else if (rcv % 100 == 2)
                    bx1 = (rcv - 2) / 100;
```

```
177             else if (rcv % 100 == 3)
                    by1 = (rcv - 3) / 100;
178             else if (rcv % 100 == 4)
                    bflag1 = (rcv - 4) / 100;

179             if (rcv % 100 == 5)
                    x2 = (rcv - 5) / 100;
180             else if (rcv % 100 == 6)
                    y2 = (rcv - 6) / 100;
181             else if (rcv % 100 == 7)
                    bx2 = (rcv - 7) / 100;
182             else if (rcv % 100 == 8)
                    by2 = (rcv - 8) / 100;
183             else if (rcv % 100 == 9)
                    bflag2 = (rcv - 9) / 100;

184             while((bflag1==1)&&(bflag2==0)) {
185               if(by1 <= -10) bflag1 = 0;
186               by1 = by1 + dby1;
187               if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message==""){
                      hitflag2=1;
                      soundexplode.play();
                      message="User1 Win!!";
188               }
189               repaint();
200               Thread.sleep(3);
201             }

202             while((bflag1==0)&&(bflag2==1)) {
203               if(by2 >= 360) bflag2 = 0;
204               by2 = by2 + dby2;
205               if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message==""){
                      hitflag1=1;
                      soundexplode.play();
                      message="User2 Win!!";
206               }
207               repaint();
208               Thread.sleep(3);
209             }

210             while((bflag1==1)&&(bflag2==1)) {
211               if(by1 <= -10) bflag1 = 0;
212               if(by2 >= 360) bflag2 = 0;
213               by1 = by1 + dby1;
214               by2 = by2 + dby2;
215               if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message==""){
                      hitflag2=1;
                      soundexplode.play();
                      message="Both died!!";
216               }
217               if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message==""){
                  hitflag1=1;
                      soundexplode.play();
                      message="Both died!!";
                  }
                  repaint();
218               Thread.sleep(3);
219             }

220           repaint();
221         }
222         catch (Exception f) {
223           f.printStackTrace();
```

```
224         }
225     }
226 }

227 public void update(Graphics g) {
228         paint(g);
229 }

230 public void paint(Graphics g) {
231     Graphics bufferg;
232     if(bufferPage == null)
            bufferPage = createImage(350, 350);
233     bufferg = bufferPage.getGraphics();

234     if (hitflag1==1) img1=img3;
235     if (hitflag2==1) img2=img4;

236     bufferg.drawString(message, 20, 20);

237     bufferg.drawImage(img1, x1, y1, this);
238     bufferg.drawImage(img2, x2, y2, this);

239     if(bflag1==1) {
            bufferg.setColor(Color.black);
            bufferg.fillRect(bx1, by1, 3, 5);
            bufferg.setColor(Color.white);
            bufferg.fillRect(bx1, by1+5, 3, 5);
240     }

241     if (bflag2==1) {
            bufferg.setColor(Color.black);
            bufferg.fillRect(bx2, by2, 3, 5);
            bufferg.setColor(Color.white);
            bufferg.fillRect(bx2, by2-5, 3, 5);
242     }

243     bufferg.dispose();
244     g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
245   }
246 }
```

参考 8-2-4 节消除图像闪烁的程序代码。

行 092　　声明缓冲页的变量。
行 231　　声明绘图对象的变量。
行 232　　创建缓冲页。
行 233　　创建缓冲页绘制图像对象。
行 236　　在缓冲页中显示字符串信息。
行 237～238　在缓冲页中绘制双方的发射器。
行 239～240　在缓冲页中绘制 User1 射出的子弹图像。
行 241～242　在缓冲页中绘制 User2 射出的子弹图像。
行 243　　完成缓冲页全图绘制后释放有关资源。
行 244　　在屏幕上显示缓冲页中的图像。

文件 Client12_5_user2.java：作用是消除 User2 端的射击图像闪烁。

```
import java.awt.*;
```

```
import java.awt.event.*;
import java.math.*;

import java.io.*;
import java.net.*;
import java.util.*;

import java.applet.*;

public class Client12_5_user2 extends Frame implements Runnable {
    int x1=150, y1=225, dx1, dy1;
    int x2=150, y2=35, dx2, dy2;
    int bx1, by1, dbx1=0, dby1=-5, bflag1=0;
    int bx2, by2, dbx2=0, dby2=5, bflag2=0;
    Image img1, img2, img3, img4, bufferPage=null;

    Socket socket;
    static String iaddr;
    static int port;
    DataOutputStream   outstream;
    DataInputStream   instream;
    int x1_send, y1_send, bx1_send, by1_send, bflag1_send, rcv;
    int x2_send, y2_send, bx2_send, by2_send, Ibflag2=0, bflag2_send;

    int hitflag1=0, hitflag2=0;
    AudioClip sound1, sound2, soundexplode;
    String message="";

    public static void main(String args[]) {
        if (args.length < 2){
            System.out.println("USAGE: java Client12_5_user2 [iaddr] [port]");
            System.exit(1);
        }

        iaddr = args[0];
        port=Integer.parseInt(args[1]);
        Client12_5_user2 workStart=new Client12_5_user2();
    }

    public Client12_5_user2() {
        super("Client12_5_user2");
        setSize(350,350);

        Toolkit tk = Toolkit.getDefaultToolkit();
        img1 = tk.getImage("car090.gif");
        img2 = tk.getImage("car180.gif");
        img3 = tk.getImage("hit1.gif");
        img4 = tk.getImage("hit2.gif");

        sound1 = Applet.newAudioClip(getClass().getResource("sound1.au"));
        sound2 = Applet.newAudioClip(getClass().getResource("sound2.au"));
        soundexplode = Applet.newAudioClip(getClass().getResource("soundexplode.au"));

        enableEvents(AWTEvent.WINDOW_EVENT_MASK);
        enableEvents(AWTEvent.KEY_EVENT_MASK);

        setVisible(true);

        try{
            socket=new Socket(InetAddress.getByName(iaddr),port);
```

```
            outstream = new DataOutputStream(socket.getOutputStream());
            instream = new DataInputStream(socket.getInputStream());
            new Thread(this).start();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
}

public void processWindowEvent(WindowEvent e) {
    if(e.getID() == WindowEvent.WINDOW_CLOSING) {
        System.exit(0);
      }
}

public void processKeyEvent(KeyEvent e) {
    if(e.getID() == KeyEvent.KEY_PRESSED) {
            switch(e.getKeyCode()) {
                case KeyEvent.VK_RIGHT:
                        dx2 = 5; dy2 = 0;
                        break;
                case KeyEvent.VK_LEFT:
                        dx2 = -5; dy2 = 0;
                        break;
                case KeyEvent.VK_UP:
                        dx2 = 0; dy2 = -5;
                        break;
                case KeyEvent.VK_DOWN:
                        dx2 = 0; dy2 = 5;
                        break;
                case KeyEvent.VK_SPACE:
                        dx2 = 0; dy2 = 0;
                        bx2 = x2 + 30;
                        by2 = y2 - 5;
                        Ibflag2 = 1;
                        sound2.play();
                        break;
                default:
                        dx2 = 0; dy2 = 0;
            }
            try{
                x2 = x2 + dx2;
                y2 = y2 + dy2;

                x2_send = x2 * 100 + 5;
                y2_send = y2 * 100 + 6;
                bx2_send = bx2 * 100 + 7;
                by2_send = by2 * 100 + 8;
                bflag2_send = Ibflag2 * 100 + 9;

                outstream.writeInt(x2_send);
                outstream.writeInt(y2_send);
                outstream.writeInt(bx2_send);
                outstream.writeInt(by2_send);
                outstream.writeInt(bflag2_send);
            }
            catch (Exception f) {
                f.printStackTrace();
            }
    }
```

```
}

public void run() {
  while(true) {
    try {
          rcv = instream.readInt();

          if (rcv % 100 == 0)
              x1 = (rcv - 0) / 100;
          else if (rcv % 100 == 1)
              y1 = (rcv - 1) / 100;
          else if (rcv % 100 == 2)
              bx1 = (rcv - 2) / 100;
          else if (rcv % 100 == 3)
              by1 = (rcv - 3) / 100;
          else if (rcv % 100 == 4)
              bflag1 = (rcv - 4) / 100;

          if (rcv % 100 == 5)
              x2 = (rcv - 5) / 100;
          else if (rcv % 100 == 6)
              y2 = (rcv - 6) / 100;
          else if (rcv % 100 == 7)
              bx2 = (rcv - 7) / 100;
          else if (rcv % 100 == 8)
              by2 = (rcv - 8) / 100;
          else if (rcv % 100 == 9)
              bflag2 = (rcv - 9) / 100;

          while((bflag1==1)&&(bflag2==0)) {
            if(by1 <= -10) bflag1 = 0;
            by1 = by1 + dby1;
            if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message=="")){
                  hitflag2=1;
                  soundexplode.play();
                  message="User1 Win!!";
            }
            repaint();
            Thread.sleep(3);
          }

          while((bflag1==0)&&(bflag2==1)) {
            if(by2 >= 360) bflag2 = 0;
            by2 = by2 + dby2;
            if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message=="")){
                  hitflag1=1;
                  soundexplode.play();
                  message="User2 Win!!";
            }
            repaint();
            Thread.sleep(3);
          }

          while((bflag1==1)&&(bflag2==1)) {
            if(by1 <= -10) bflag1 = 0;
            if(by2 >= 360) bflag2 = 0;
            by1 = by1 + dby1;
            by2 = by2 + dby2;
            if ((bx1>=x2)&&(bx1<=x2+60)&&(by1<=y2)&&(message=="")){
                  hitflag2=1;
```

```
                soundexplode.play();
                message="Both died!!";
            }
            if ((bx2>=x1)&&(bx2<=x1+60)&&(by2>=y1)&&(message==""))){
                hitflag1=1;
                soundexplode.play();
                message="Both died!!";
            }
            repaint();
            Thread.sleep(3);
          }

        repaint();
      }
      catch (Exception f) {
          f.printStackTrace();
      }
   }
}

public void update(Graphics g) {
      paint(g);
}

public void paint(Graphics g) {
    Graphics bufferg;
    if(bufferPage == null)
        bufferPage = createImage(350, 350);
    bufferg = bufferPage.getGraphics();

    if (hitflag1==1) img1=img3;
    if (hitflag2==1) img2=img4;
        bufferg.drawString(message, 20, 20);

        bufferg.drawImage(img1, x1, y1, this);
        bufferg.drawImage(img2, x2, y2, this);

        if(bflag1==1) {
            bufferg.setColor(Color.black);
            bufferg.fillRect(bx1, by1, 3, 5);
            bufferg.setColor(Color.white);
            bufferg.fillRect(bx1, by1+5, 3, 5);
        }

        if (bflag2==1) {
            bufferg.setColor(Color.black);
            bufferg.fillRect(bx2, by2, 3, 5);
            bufferg.setColor(Color.white);
            bufferg.fillRect(bx2, by2-5, 3, 5);
        }

        bufferg.dispose();
        g.drawImage(bufferPage, getInsets().left, getInsets().top, this);
    }
}
```

参考 Client12_5_user1.java 的解说。

编译程序

输入“javac *.java”进行编译。

运行步骤

挑选三台网络计算机，选择一台作为 Server；另选一台作为 Client1；再另选一台作为 Client2。若读者仅有一台计算机可用，可打开三个“命令提示符”窗口，一个模拟服务器端，另两个模拟 Client1 与 Client2（本例采用一台计算机 222.28.113.169，三个“命令提示符”窗口）。

1 在服务器端创建网站。

输入“java Server12_5 1234”，则窗口会显示已创建网站信息，并等待各客户端来连接的信息，如图 12-20 所示。

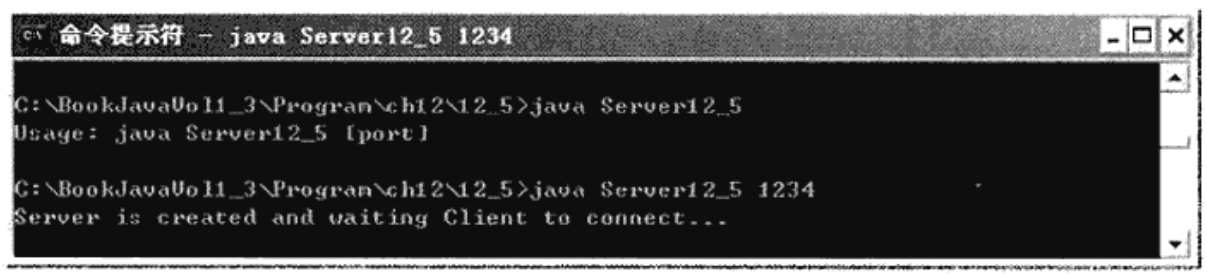


图 12-20

2 Client1 向 Server 报到。

在图 12-21 所示的“命令提示符”窗口中输入“java Client12_5_user1 222.28.113.169 1234”，则立即由 Client1 网络连接 Server，同时显示窗口框架，如图 12-22 所示。

图 12-21

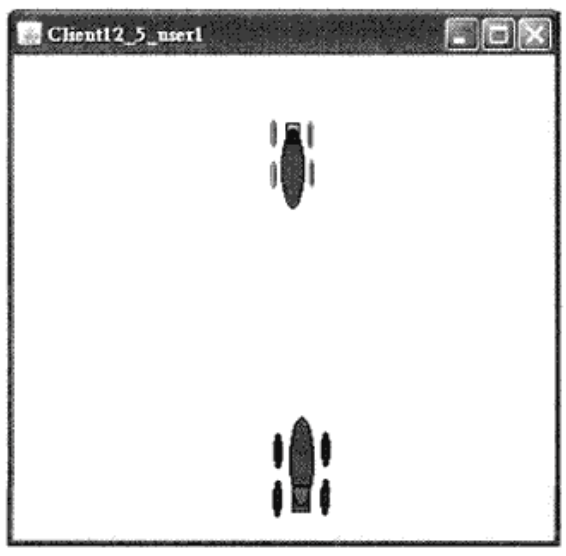


图 12-22

3 Client2 向 Server 报到。

在图 12—23 所示的“命令提示符”窗口中输入“java Client12_5_user2 222.28.113.169 1234”，则立即由 Client2 网络连接 Server，同时显示窗口框架，如图 12—24 所示。

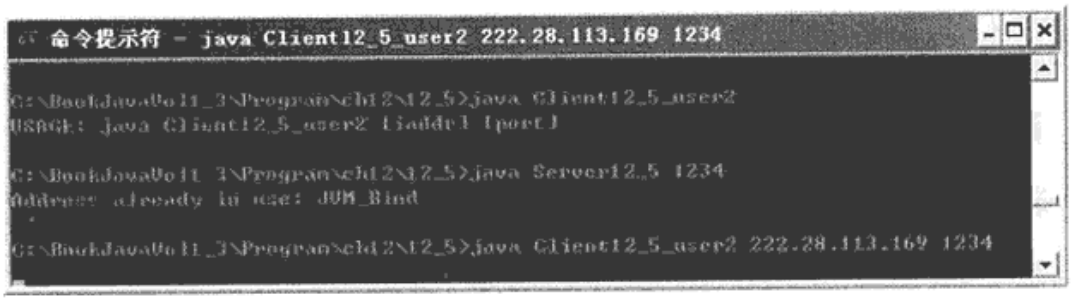


图 12-23

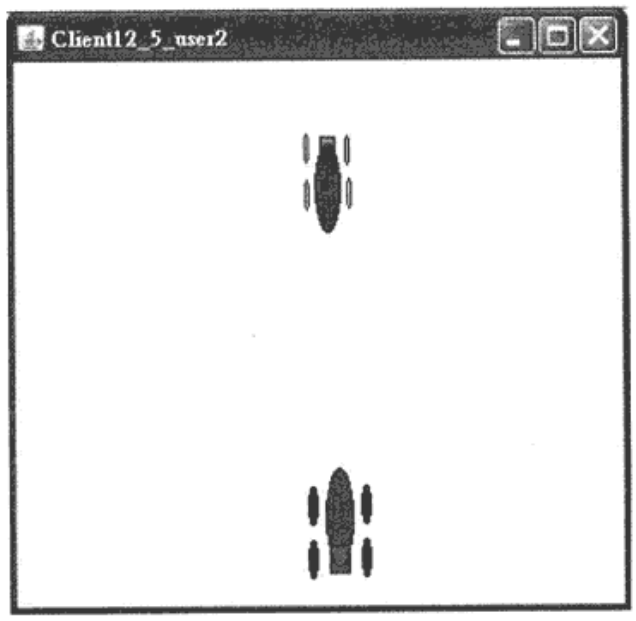


图 12-24

4 运行程序。

User1 按 Space 键时发射子弹，当子弹进入红色发射器范围且尚未显示中弹信息时，则将红色发射器图像改成爆炸图像，播放击中音效，显示“User1 Win!!”信息，如图 12—25 所示。

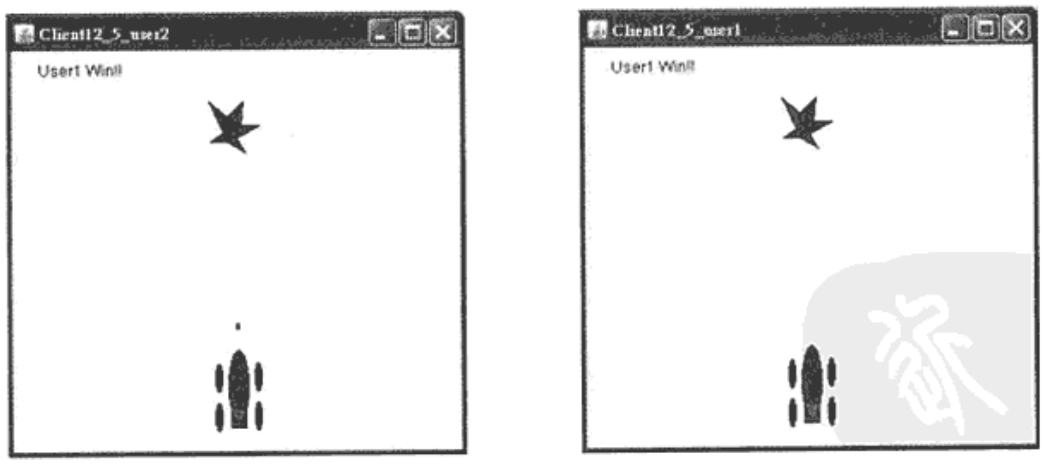


图 12-25

User2 按 Space 键时发射子弹，当子弹进入绿色发射器范围且尚未显示中弹信息时，则将绿色发射器图像改成爆炸图像，播放击中音效，显示“User2 Win!!”信息，如图 12—26 所示。

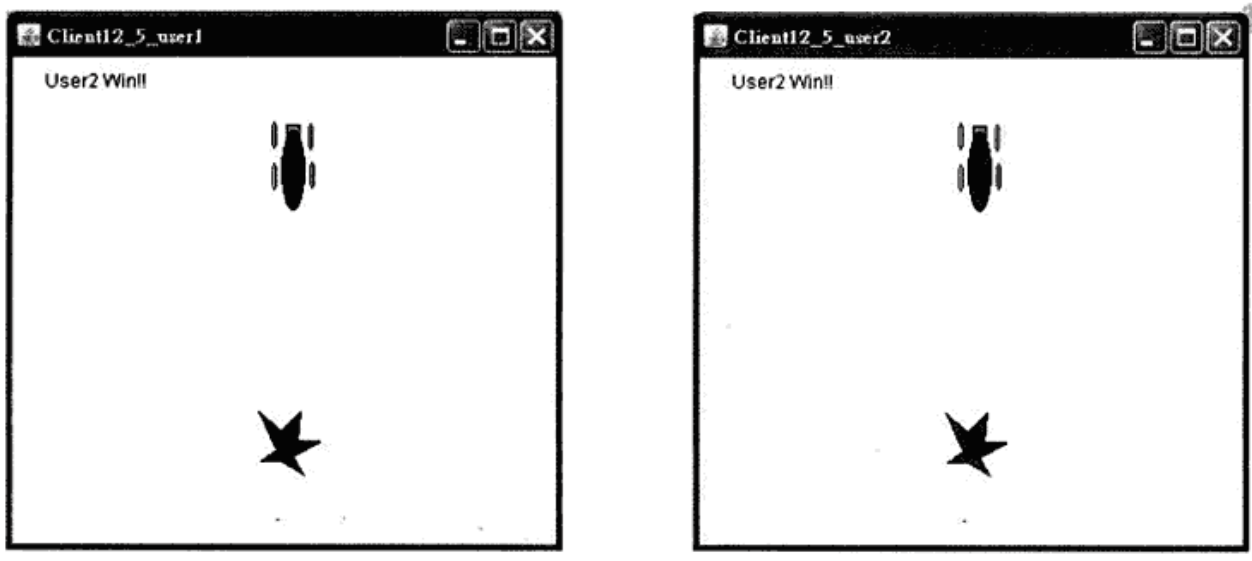


图 12-26

12-6 习题

1. 试设计射击对阵，包括3个发射器。
2. 试模拟射击对阵现象，当User1的子弹碰撞到User2的子弹时，双方的子弹均被击毁。

PART 04

2D绘图设计

对我们而言，2D平面绘图并不是一件困难的事，而3D绘图就不是那么简单了。本书在绘图方面的描述，最终目的是让读者认识3D绘图，使用2D平面来解说一些数学公式比较容易，故本部分先以2D环境进行必要的数学复习，进而支持本书第5部分“3D绘图设计”。本部分内容包括屏幕/视图坐标的互换，点、直线、向量、内积、法线等概念，缩放/旋转的设计。

Chapter 13　屏幕坐标与视图坐标

2D平面坐标是建立在x轴和y轴基础上的。在屏幕上：原点在左上角，x坐标是从左到右，y坐标是从上到下，称为“屏幕坐标（Screen Coordinates）”。在视觉上：则习惯原点在中央，x坐标是从左到右，y坐标是从下到上，称为“视图坐标（View Coordinates）”。因此在2D平面绘图中，设计时以视图坐标进行思考，设计完成后再将视图坐标转换成对应的屏幕坐标，在屏幕上显示图案。

Chapter 14　绘图概念

对读者而言，2D平面绘图也许比较容易，而3D绘图并不简单。本书在绘图方面的描述，最终目的是让读者认识3D绘图。在3D绘图设计上，需要一些数学知识来支持。

由于以2D平面来解说这些数学知识比较容易，所以本章先以2D环境复习点（Point）、直线（Line）、向量（Vector）、内积（Inner Product）、法线（Normal）等概念，进而支持本书第5部分“3D绘图设计”。

Chapter 15　缩放与旋转

笔者一再强调，本书的绘图重点是3D绘图，本部分的主题虽然是2D平面绘图，但它是为3D立体绘图做准备。为了介绍如何进行3D立体图形的缩放与旋转，本章先介绍2D图形的缩放与旋转，以较简单的层次让读者了解其中的部分原理及程序设计方法。

Chapter 13　屏幕坐标与视图坐标

13-1 简介

2D 平面坐标是建立在 x 轴和 y 轴基础上的。在屏幕上：原点在左上角，x 坐标是从左到右，y 坐标是从上到下，称为“屏幕坐标（Screen Coordinates）”。在视觉上：则习惯原点在中央，x 坐标是从左到右，y 坐标是从下到上，称为“视图坐标（View Coordinates）”。因此在 2D 平面绘图中，设计时以视图坐标进行思考，设计完成后再将视图坐标转换成对应的屏幕坐标，在屏幕上显示图案。

13-2 屏幕坐标

在屏幕上，x 轴坐标大小是由左向右递增，y 轴坐标大小是由上向下递增，左上方为原点(x=0, y=0)，如图 13–1 所示。本节将以屏幕坐标来解说 2D 绘图。

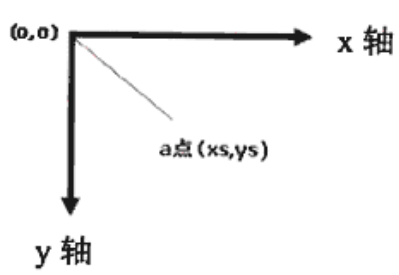


图 13-1

范例 64　设计文件 Ex13_2.java，其功能是解释屏幕坐标的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex13_2 extends Frame implements Runnable {
05   int xs=100, ys=100;

06   public static void main(String args[]) {
07     Ex13_2 workStart=new Ex13_2();
08   }
```

```
09  public Ex13_2() {
10     super("Ex13_2");
11     setSize(350, 350);

12     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
13     setVisible(true);
14     new Thread(this).start();
15  }

16  public void processWindowEvent(WindowEvent e) {
17     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
18        System.exit(0);
19     }
20  }

21  public void run() {
22     repaint();
23  }

24  public void paint(Graphics g) {
25     g.drawLine(0,0,xs,ys);
26  }
27 }
```

行 05 声明屏幕坐标变量。
行 25 以屏幕坐标绘制直线。

运行结果

以屏幕坐标表示的两点(0，0)、(xs，ys)来绘制直线，如图 13–2 所示。

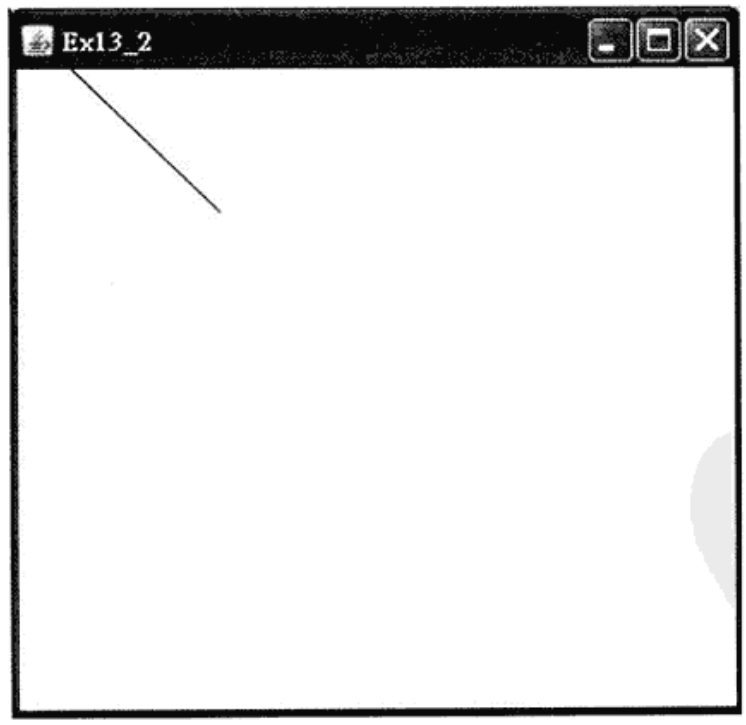


图 13-2

13-3 视图坐标

在视觉上，我们习惯以视图坐标进行思考，原点在窗口中央，原点右端的 x 坐标为正，从左到右递增，原点左端的 x 坐标为负，从右到左递减；原点上端的 y 坐标为正，从下到上递增，原点下端的 y 坐标为负，从上到下递减。它也称为“视图坐标（View Coordinates）”。

因此在 2D 平面绘图中，设计时以视图坐标进行思考，设计完成后再将视图坐标转换成对应的屏幕坐标，在屏幕上显示图案。视图坐标与屏幕坐标的关系如图 13-3 所示。

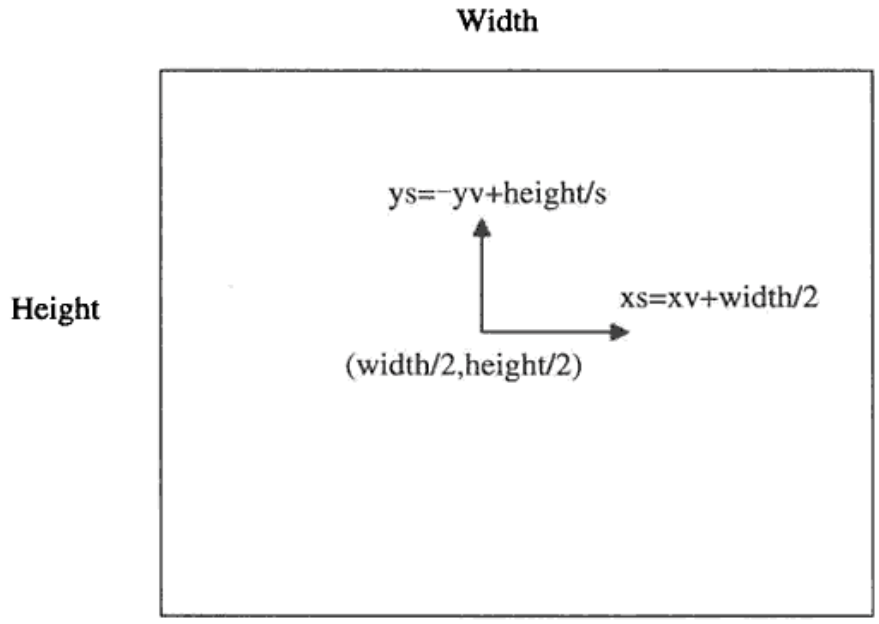


图 13-3

其中，Width 为屏幕的宽；Height 为屏幕的高；(xv，yv)为视图坐标；(xs，ys)为屏幕坐标。

范例 65 在视图坐标下，以原点(xv1，yv1)与 a 点(xv2，yv2)来绘制直线，其中原点为(0，0)，a 点为(100，100)，如图 13-4 所示，则其屏幕坐标是多少？请设计程序 Ex13_3.java 进行验证。

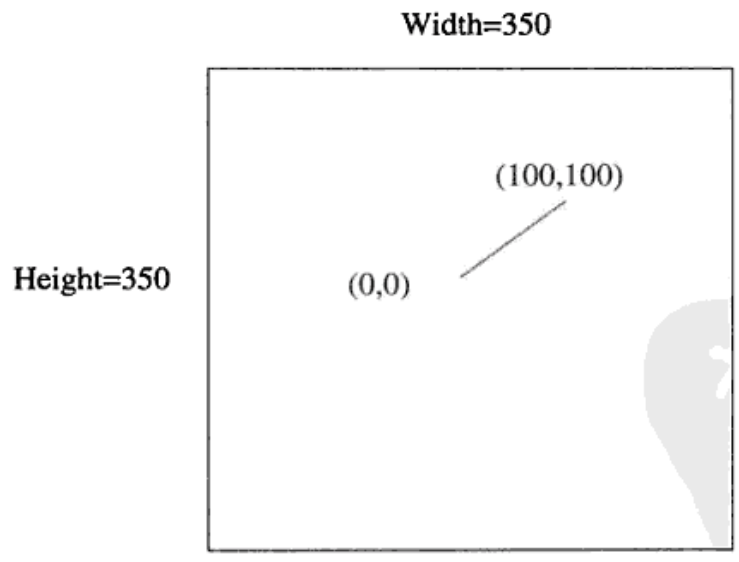


图 13-4

解：由图 13-3 可知，其屏幕坐标为

(1) 原点：xs1 = 350/2 = 175，ys1 = 350/2 = 175。

(2) a 点：xs2 = 100 + 175 = 275，ys2 = −100 + 175 = 75。

程序验证：（视图坐标转换成屏幕坐标的应用）

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.awt.Graphics;

04 public class Ex13_3 extends Frame implements Runnable {
05   int xv1=0, yv1=0, xv2=100, yv2=100;
06   int xs1, ys1, xs2, ys2;

07   public static void main(String args[]) {
08     Ex13_3 workStart=new Ex13_3();
09   }

10   public Ex13_3() {
11     super("Ex13_3");
12     setSize(350, 350);

13     xs1 = 350/2;
14     ys1 = 350/2;

15     xs2 = xv2 + 350/2;
16     ys2 = -yv2 + 350/2;

17     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
18     setVisible(true);
19     new Thread(this).start();
20   }

21   public void processWindowEvent(WindowEvent e) {
22     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
23       System.exit(0);
24     }
25   }

26   public void run() {
27     repaint();
28   }

29   public void paint(Graphics g) {
30     g.drawLine(xs1, ys1, xs2, ys2);
31   }
32 }
```

行 05	声明视图坐标变量并设置坐标值。
行 06	声明屏幕坐标变量。
行 13～16	将视图坐标转换成屏幕坐标。
行 30	以屏幕坐标绘出直线。

运行结果

运行结果如图 13–5 所示。

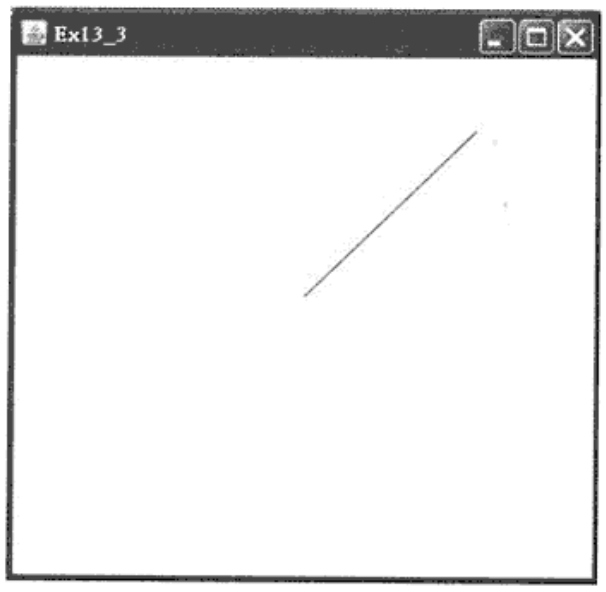


图 13-5

13-4 包应用

增加程序功能时，其中的程序代码也将随之增多，这时为了保证程序的可读性，我们可将某些具有特别功能的程序代码单独编写在某个包程序中。有关该主题的说明可参考本系列丛书第二册《Java 典型应用彻查 1000 例——网络应用开发》的第 6 章。其优点如下：

（1）减少编程的代码量。可将具有特别功能的程序代码单独编写在包程序中，以提高整体的可读性与设计性。

（2）缩短计算时间。可将各类计算分派到各包程序中先计算，当主程序需要时再直接读取使用，这会使绘图流畅运行。

范例 66 设计文件Ex13_4.java，其功能是解释如何使用math2D包与将视图坐标转换成屏幕坐标。

包文件 view_To_screen_coordinates.java：将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch13\13_4\math2D。

```
01 package math2D;

02 import java.awt.*;

03 public class view_To_screen_coordinates {
04   public int viewX_To_screenX(int x) {
05     return x + 350/2;
06   }

07   public int viewY_To_screenY(int y) {
08     return -y + 350/2;
09   }
10 }
```

行 01	创建 math2D 包。
行 03	创建类。
行 04～06	是一个方法，将视图坐标 x 转换成屏幕坐标 x。
行 07～09	是一个方法，将视图坐标 y 转换成屏幕坐标 y。

文件 Ex13_4.java：使用行 01～10 的包，将视图坐标转换成屏幕坐标，再以屏幕坐标绘出直线。

```
11 import math2D.*;

12 import java.awt.*;
13 import java.awt.event.*;
14 import java.awt.Graphics;

15 public class Ex13_4 extends Frame implements Runnable {
16    int xv1=0, yv1=0, xv2=100, yv2=100;
17    int xs1, ys1, xs2, ys2;

18    public static void main(String args[]) {
19       Ex13_4 workStart=new Ex13_4();
20    }

21    public Ex13_4() {
22       super("Ex13_4");
23       setSize(350, 350);

24       view_To_screen_coordinates vTs = new view_To_screen_coordinates();
25       xs1 = vTs.viewX_To_screenX(xv1);
26       ys1 = vTs.viewY_To_screenY(yv1);
27       xs2 = vTs.viewX_To_screenX(xv2);
28       ys2 = vTs.viewY_To_screenY(xv2);

29       enableEvents(AWTEvent.WINDOW_EVENT_MASK);
30       setVisible(true);
31       new Thread(this).start();
32    }

33    public void processWindowEvent(WindowEvent e) {
34       if(e.getID() == WindowEvent.WINDOW_CLOSING) {
35          System.exit(0);
36       }
37    }

38    public void run() {
39       repaint();
40    }

41    public void paint(Graphics g) {
42       g.drawLine(xs1, ys1, xs2, ys2);
43    }
44 }
```

行 11	导入行 01～10 的 math2D 包。
行 16	声明视图坐标变量并设置坐标值。
行 17	声明屏幕坐标变量。
行 24	以行 03～10 的类程序创建新对象 vTs。
行 25～28	以对象 vTs 的 vTs.viewX_To_screenX()方法将视图坐标 x 转换成屏幕坐标 x，以 vTs.viewY_To_screenY()方法将视图坐标 y 转换成屏幕坐标 y。
行 42	以屏幕坐标绘出直线。

编译程序

输入“javac Ex13_4.java”进行编译，如图 13-6 所示。

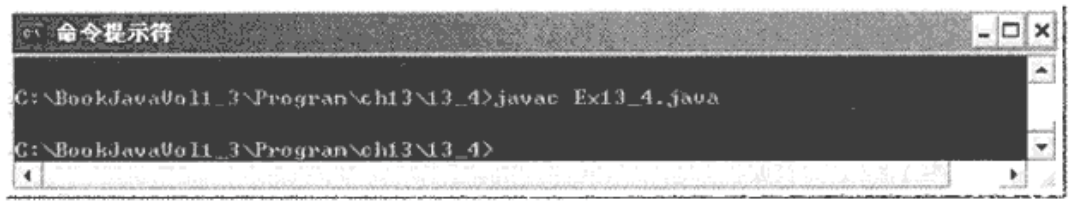


图 13-6

运行程序

输入“java Ex13_4”，如图 13-7 所示。

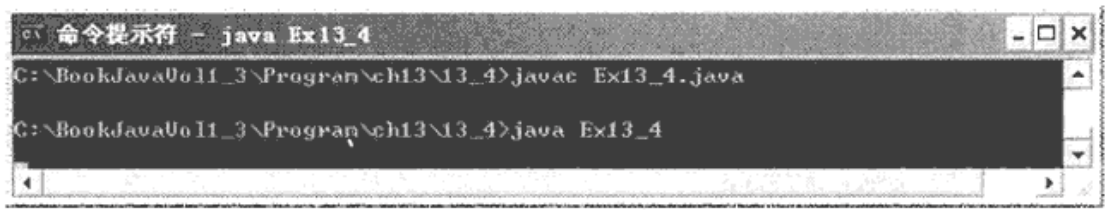


图 13-7

运行结果

运行结果如图 13-8 所示。

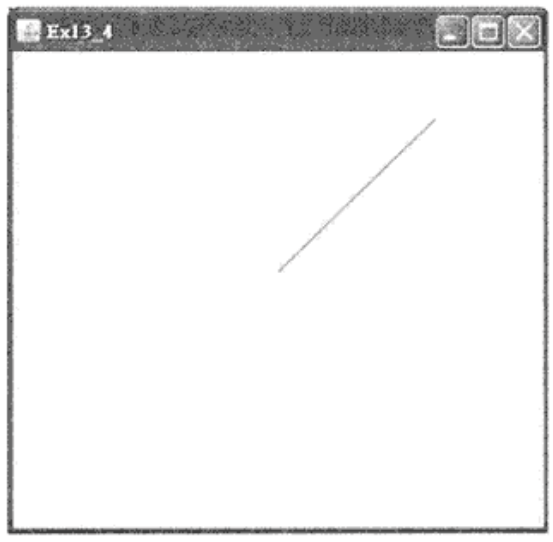


图 13-8

13-5 习题

1. 屏幕坐标的意义是什么？
2. 视图坐标的意义是什么？
3. 视图坐标与屏幕坐标之间有何关系？
4. 屏幕坐标与视图坐标各在什么情况下使用？
5. 为何要使用包？它有什么优点？

Chapter 14 绘图概念

14-1 简介

对读者而言，2D 平面绘图也许比较容易，但是 3D 绘图并不简单。本书在绘图方面的描述，最终目的是让读者认识 3D 绘图。在 3D 绘图设计上，需要一些数学知识来支持。

由于以 2D 平面来解说这些数学知识比较容易，所以本章先以 2D 环境复习点（Point）、直线（Line）、向量（Vector）、内积（Inner Product）、法线（Normal）等概念，进而支持本书第 5 部分"3D 绘图设计"。

14-2 点与线

点/线（Point/Line）是绘图组件中最基本的，也是最重要的两个组件。通常点以坐标(x，y)表示，如图 14–1 所示；线以两点坐标((x1，y1)，(x2，y2))表示，如图 14–2 所示。

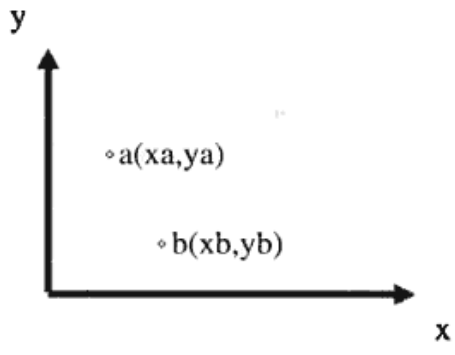


图 14-1

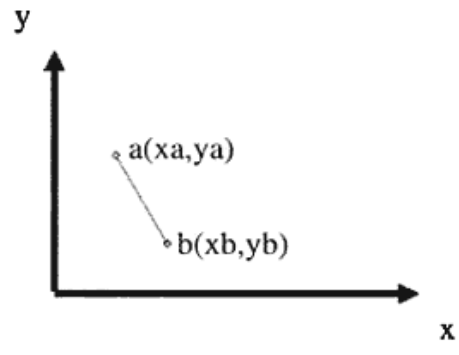


图 14-2

14-3 向量

向量（Vector）是两点的连接，但与"线（Line）"不同的是，其起始点为原点(0,0)，且具有方向与长度，如图 14–3 所示。

1 向量 A：长度为 $(xa^2+ya^2)^{1/2}$，表示符号为 A(xa，ya)。
2 向量 B：长度为 $(xb^2+yb^2)^{1/2}$，表示符号为 B(xb，yb)。
3 向量 AB：长度为 $((xb-xa)^2+(yb-ya)^2)^{1/2}$，表示符号为 AB(xb–xa，yb–ya)。

如图 14–4 所示，向量和与向量差为 C = A ± B。即 C(xc,yc) = A(xa,ya) ± B(xb,yb)。其中，xc = xa ± xb；yc = ya ± yb。也即 C(xa±xb, ya±yb) = A(xa,ya) ± B(xb,yb)。

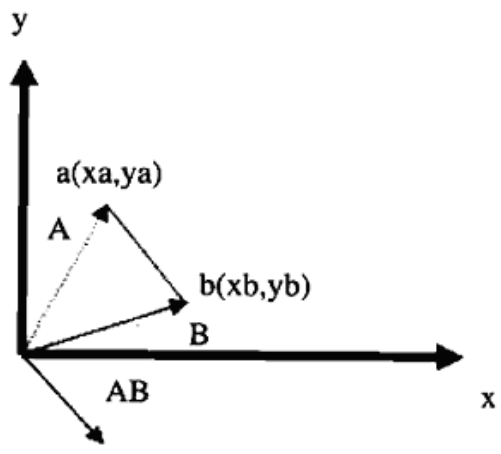


图 14-3

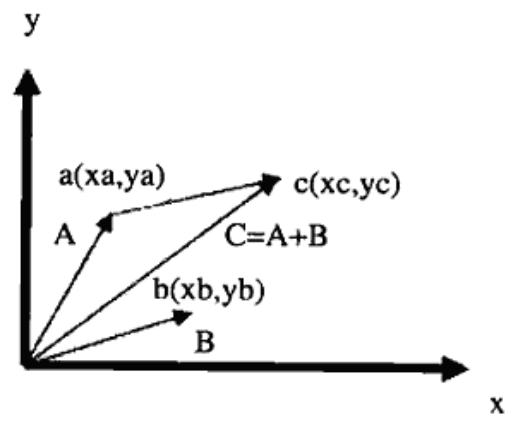


图 14-4

设计范例 67，如图 14–4 所示，设计文件 Ex14_3.java 绘出向量 C=A+B，并在 math2D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标，创建文件 vector_calculator.java，运行坐标的四则运算。

范例 67 设计文件 Ex14_3.java，其功能是解释 math2D 包与向量和的应用。

包文件 view_To_screen_coordinates.java：将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch14\14_3\math2D。

```
01 package math2D;
02 import java.awt.*;

03 public class view_To_screen_coordinates {
04   public int viewX_To_screenX(int x) {
05     return x + 350/2;
06   }

07   public int viewY_To_screenY(int y) {
08     return -y + 350/2;
09   }
10 }
```

行 01	创建 math2D 包。
行 03	创建类。
行 04～06	是一个方法，将视图坐标 x 转换成屏幕坐标 x。
行 07～09	是一个方法，将视图坐标 y 转换成屏幕坐标 y。

包文件 vector_calculator.java：运行坐标的四则运算，本例目录为 C：\BookJavaVol1_3\Program\ch14\14_3\math2D。

```
11 package math2D;
12 import java.awt.*;

13 public class vector_calculator {
14   private int x, y;
```

```
15   public void setPt(int x, int y) {
16       this.x = x; this.y = y;
17   }

18   public void vectAdd(int x, int y) {
19       this.x += x; this.y += y;
20   }

21   public void vectSubtract(int x, int y) {
22       this.x -= x; this.y -= y;
23   }

24   public void vectMultiply(int x, int y) {
25       this.x *= x; this.y *= y;
26   }

27   public void vectDivide(int x, int y) {
28       this.x /= x; this.y /= y;
29   }

30   public int getPtX() {return x;}
31   public int getPtY() {return y;}
32 }
```

行 11	创建 math2D 包。
行 13	创建类。
行 14	声明坐标值变量。
行 15～17	是一个方法，设置点的坐标。
行 18～20	是一个方法，计算参数坐标与行 15～17 所设置的坐标的向量和，存储在行 14 所声明的变量中。
行 21～23	是一个方法，计算参数坐标与行 15～17 所设置的坐标的向量差，存储在行 14 所声明的变量中。
行 24～26	是一个方法，计算参数坐标与行 15～17 所设置的坐标的向量积，存储在行 14 所声明的变量中。
行 27～29	是一个方法，计算参数坐标与行 15～17 所设置的坐标的向量商，存储在行 14 所声明的变量中。
行 30	读取行 14 的 x 值。
行 31	读取行 14 的 y 值。

文件 Ex14_3.java：绘出向量 C=A+B。

```
33 import math2D.*;

34 import java.awt.*;
35 import java.awt.event.*;
36 import java.awt.Graphics;

37 public class Ex14_3 extends Frame implements Runnable {
38   int x0=0, y0=0, xa=20, ya=40, xb=100, yb=10, xc, yc;
39   int x0s, y0s, xas, yas, xbs, ybs, xcs, ycs;

40   public static void main(String args[]) {
41     Ex14_3 workStart=new Ex14_3();
```

```
42  }

43  public Ex14_3() {
44     super("Ex14_3");
45     setSize(350, 350);

46     vector_calculator vc = new vector_calculator();
47     vc.setPt(xa, ya);
48     vc.vectAdd(xb, yb);
49     xc = vc.getPtX();
50     yc = vc.getPtY();

51     view_To_screen_coordinates vTs = new view_To_screen_coordinates();
52     x0s = vTs.viewX_To_screenX(x0);
53     y0s = vTs.viewY_To_screenY(y0);
54     xas = vTs.viewX_To_screenX(xa);
55     yas = vTs.viewY_To_screenY(ya);
56     xbs = vTs.viewX_To_screenX(xb);
57     ybs = vTs.viewY_To_screenY(yb);
58     xcs = vTs.viewX_To_screenX(xc);
59     ycs = vTs.viewY_To_screenY(yc);

60     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
61     setVisible(true);
62     new Thread(this).start();
63     }

64  public void processWindowEvent(WindowEvent e) {
65     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
66        System.exit(0);
67     }
68  }

69  public void run() {
70        repaint();
71  }

72  public void paint(Graphics g) {
73     g.drawLine(x0s, y0s, xas, yas);
74     g.drawLine(x0s, y0s, xbs, ybs);
75     g.drawLine(x0s, y0s, xcs, ycs);
76     }
77 }
```

行 33	导入 math2D 包。
行 38	声明各点的视图坐标变量。
行 39	声明各点的屏幕坐标变量。
行 46	以行 11～32 的类创建新对象 vc。
行 47～48	以对象 vc 计算 A 点与 B 点的向量和 C。
行 49～50	以对象 vc 读取 C 点的坐标值。
行 51	以行 01～10 的类创建新对象 vTs。
行 52～59	以对象 vTs 将各点的视图坐标转换成屏幕坐标。
行 73～75	以各点的屏幕坐标绘出向量 A、B、C。

编译程序

输入 "javac Ex14_3.java" 进行编译，如图 14-5 所示。

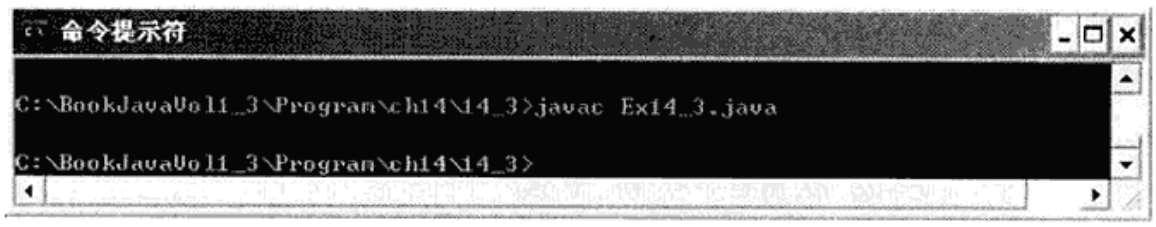


图 14-5

运行程序

输入 "java Ex14_3"，如图 14-6 所示。

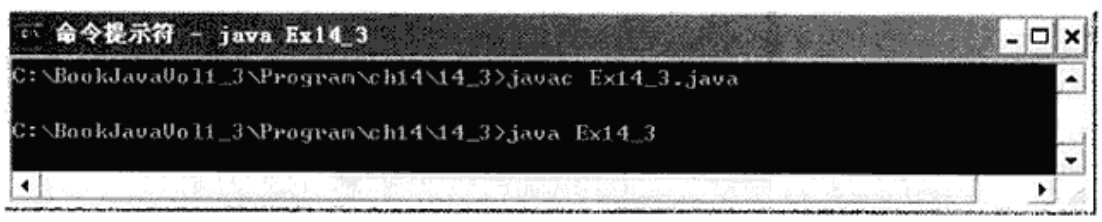


图 14-6

运行结果

运行结果如图 14-7 所示。

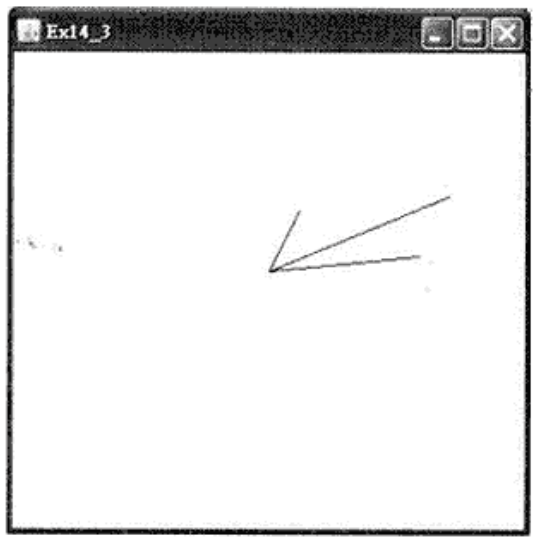


图 14-7

14-4 内积与法线

读者也许会好奇，2D、3D、4D（移动的立体）绘图与内积（Inner Product）、法线（Normal）有何关系？在本书第 5 部分 "3D 绘图设计" 中，内积与法线在消除隐藏线上都是极为重要的。

1 内积：指两向量的乘积，如图 14-8 所示的 AxB。

$A \times B = xa \times xb + ya \times yb$

（因为|A|为向量 A 的长度，所以 $xa = \cos\emptyset_A$；依此类推）

$$= |A| \times \cos\varnothing_A \times |B| \times \cos\varnothing_B + |A| \times \sin\varnothing_A \times |B| \times \sin\varnothing_B$$
$$= |A| \times |B| \times (\cos\varnothing_A \times \cos\varnothing_B) + |A| \times |B| \times (\sin\varnothing_A \times \sin\varnothing_B)$$
$$= |A| \times |B| \times (\cos\varnothing_A \times \cos\varnothing_B) + (\sin\varnothing_A \times \sin\varnothing_B)$$
(因为 $\cos(\alpha-\beta) = \cos\alpha \times \cos\beta + \sin\alpha \times \sin\beta$)
$$= |A| \times |B| \times \cos(\varnothing_A - \varnothing_B)$$
(因为 $\varnothing = \varnothing_A - \varnothing_B$)
$$= |A| \times |B| \times \cos\varnothing$$

故 $\cos\varnothing = (A \times B) / (|A| \times |B|)$

当 $A \times B = 0$ 时，$\cos\varnothing$ 必为 0，$\angle\varnothing$ 为 90°

即当内积 $A \times B = 0$ 时，A、B 两向量必然互相垂直。

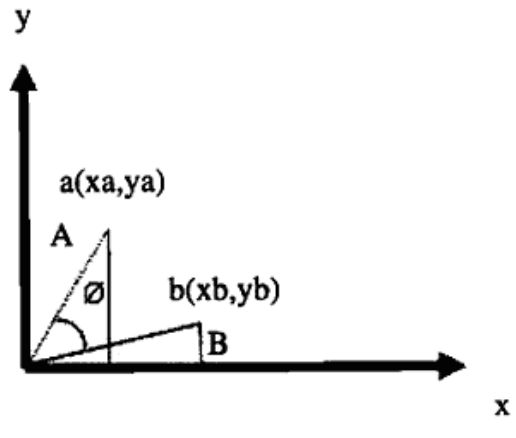


图 14-8

2 线段向量、法线向量：如图 14—9 所示的线段 ab、向量 AB 与法线向量 N_{AB}。

向量 AB 的长度为 $((xb-xa)^2+(yb-ya)^2)^{1/2}$，表示符号为 AB((xb−xa)，(yb−ya))，其法线向量 N_{AB} = (−(yb−ya)，(xb−xa)) 或 N_{AB}=((yb−ya)，−(xb−xa))。

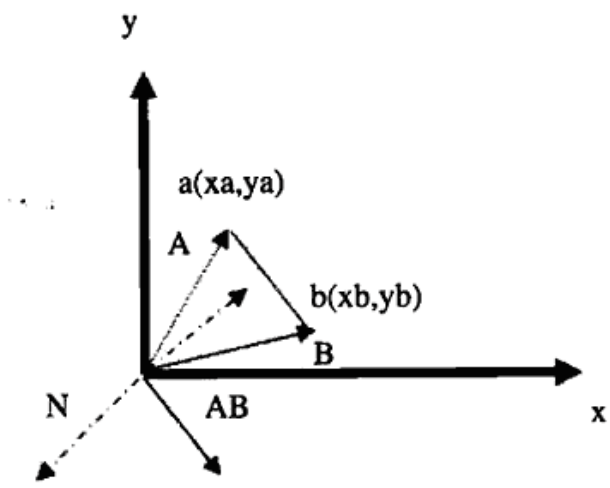


图 14-9

设计范例 68，如图 14—9 所示，设计文件 Ex14_4.java 绘出线段 ab 的向量 AB 与其法线向量 N（上端 N1 与下端 N2）。在 math2D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标，创建文件 vector_calculator.java，运行坐标的四则运算。

范例 68 设计文件 Ex14_4.java，其功能是解释如何使用 math2D 包与绘出线段的法线向量。

包文件 view_To_screen_coordinates.java：参考范例 67，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch14\14_4\math2D。

包文件 vector_calculator.java：参考范例 67，运行坐标的四则运算，本例目录为 C：\

BookJavaVol1_3\Program\ch14\14_4\math2D。

文件 Ex14_4.java：绘出线段 ab 的向量 AB 与其法线向量 N（上端 N1 与下端 N2）。

```
33 import math2D.*;

34 import java.awt.*;
35 import java.awt.event.*;
36 import java.awt.Graphics;

37 public class Ex14_4 extends Frame implements Runnable {
38   int x0=0, y0=0, xa=20, ya=40, xb=100, yb=10, xc, yc;
39   int xN1, yN1, xN2, yN2;
40   int x0s, y0s, xcs, ycs, xN1s, yN1s, xN2s, yN2s;

41   public static void main(String args[]) {
42     Ex14_4 workStart=new Ex14_4();
43   }

44   public Ex14_4() {
45     super("Ex14_4");
46     setSize(350, 350);

47     vector_calculator vc = new vector_calculator();
48     vc.setPt(xb, yb);
49     vc.vectSubtract(xa, ya);
50     xc = vc.getPtX();
51     yc = vc.getPtY();

52     xN1 = -yc;   yN1 = xc;
53     xN2 = yc;    yN2 = -xc;

54     view_To_screen_coordinates vTs = new view_To_screen_coordinates();
55     x0s = vTs.viewX_To_screenX(x0);
56     y0s = vTs.viewY_To_screenY(y0);
57     xcs = vTs.viewX_To_screenX(xc);
58     ycs = vTs.viewY_To_screenY(yc);
59     xN1s = vTs.viewX_To_screenX(xN1);
60     yN1s = vTs.viewY_To_screenY(yN1);
61     xN2s = vTs.viewX_To_screenX(xN2);
62     yN2s = vTs.viewY_To_screenY(yN2);

63     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
64     setVisible(true);
65     new Thread(this).start();
66     }

67   public void processWindowEvent(WindowEvent e) {
68     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
69       System.exit(0);
70     }
71   }

72   public void run() {
73     repaint();
74   }

75   public void paint(Graphics g) {
76     g.drawLine(x0s, y0s, xcs, ycs);
77     g.drawLine(x0s, y0s, xN1s, yN1s);
```

```
78      g.drawLine(x0s, y0s, xN2s, yN2s);
79      }
80 }
```

行 33	导入 math2D 包。
行 38～39	声明各点的视图坐标变量。
行 40	声明各点的屏幕坐标变量。
行 47	以行 11～32 的类创建新对象 vc。
行 48～51	以对象 vc 计算线段 ab 的向量 AB 的坐标。
行 52～53	以向量 AB 的坐标求出法线向量 N1 与 N2 的坐标。
行 54	以行 01～10 的类创建新对象 vTs。
行 55～62	以对象 vTs 将各点的视图坐标转换成屏幕坐标。
行 76～78	以各点的屏幕坐标绘出线段 ab 的向量 AB 与法线 N1、N2。

编译程序

输入“javac Ex14_4.java”进行编译，如图 14-10 所示。

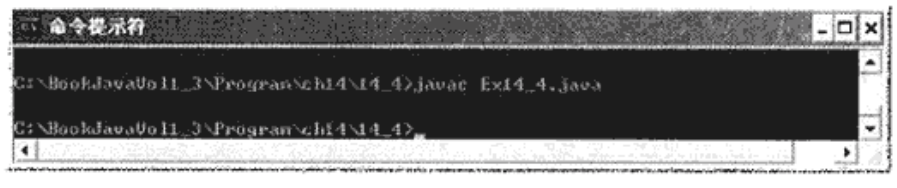


图 14-10

运行程序

输入“java Ex14_4”，如图 14-11 所示。

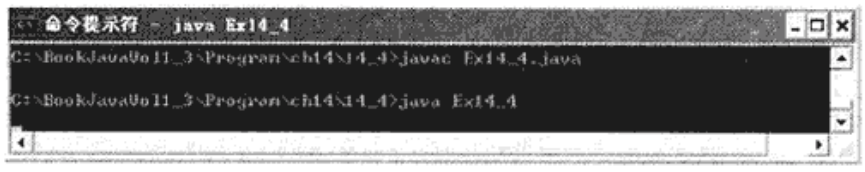


图 14-11

运行结果

运行结果如图 14-12 所示。

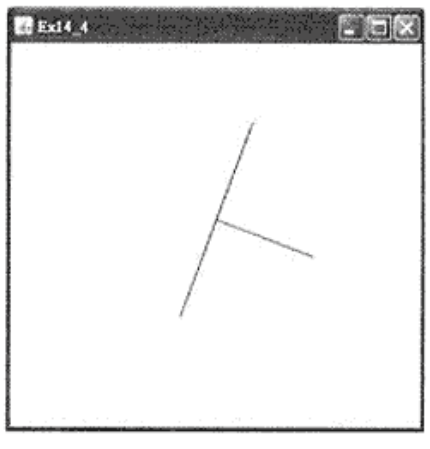


图 14-12

14-5 多边形

三个（含）以上的点所包含的范围称为“多边形（Polygon）”。在 2D、3D 绘图中，多边形是不可或缺的绘图组件，是图案画面的基本单元（Basic Unit），每一幅图案都是由不同的多个多边形组成的。掌握了多边形的特性，也就掌握了图案的特性，如旋转（Rotate）、缩放（Zoom）、隐藏（Hide）等。

如图 14-13 所示，多边形是由多条直线连接而成的封闭区域，每条直线有其起点与终点，掌握每个起点与终点的变化就可掌握图案的变化。

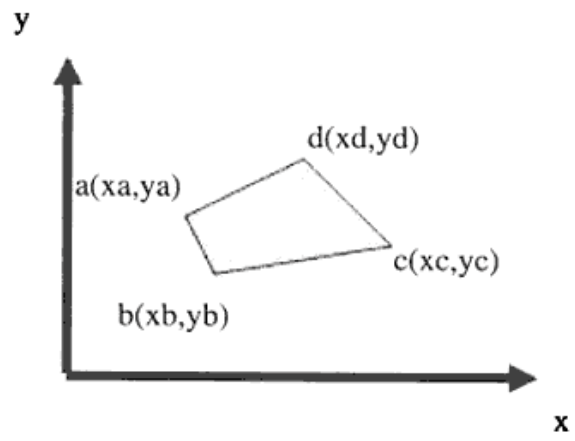


图 14-13

设计范例 69，如图 14-13 所示，设计文件 Ex14_5.java 绘出多边形 abcd。在 math2D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标。

范例 69 设计文件 Ex14_5.java，其功能是解释如何使用 math2D 包与绘出多边形。

包文件 view_To_screen_coordinates.java：参考范例 67，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch14\14_5\math2D。

文件 Ex14_5.java：绘出多边形 abcd。

```
11 import math2D.*;

12 import java.awt.*;
13 import java.awt.event.*;
14 import java.awt.Graphics;

15 public class Ex14_5 extends Frame implements Runnable {
16   int xa=20, ya=40, xb=30, yb=10, xc=100, yc=20, xd=80, yd=60;
17   int xas, yas, xbs, ybs, xcs, ycs, xds, yds;

18   public static void main(String args[]) {
19     Ex14_5 workStart=new Ex14_5();
20   }

21   public Ex14_5() {
22     super("Ex14_5");
23     setSize(350, 350);

24     view_To_screen_coordinates vTs = new view_To_screen_coordinates();
25     xas = vTs.viewX_To_screenX(xa);
26     yas = vTs.viewY_To_screenY(ya);
```

```
27      xbs = vTs.viewX_To_screenX(xb);
28      ybs = vTs.viewY_To_screenY(yb);
29      xcs = vTs.viewX_To_screenX(xc);
30      ycs = vTs.viewY_To_screenY(yc);
31      xds = vTs.viewX_To_screenX(xd);
32      yds = vTs.viewY_To_screenY(yd);

33      enableEvents(AWTEvent.WINDOW_EVENT_MASK);
34      setVisible(true);
35      new Thread(this).start();
36      }

37    public void processWindowEvent(WindowEvent e) {
38      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
39          System.exit(0);
40      }
41    }

42    public void run() {
43          repaint();
44    }

45    public void paint(Graphics g) {
46      int pgx[] = {xas,xbs,xcs,xds};
47      int pgy[] = {yas,ybs,ycs,yds};

48      g.fillPolygon(pgx, pgy, 4);
49      }
50  }
```

行 11	导入行 01～10 的 math2D 包。
行 16	声明视图坐标变量并设置坐标值。
行 17	声明屏幕坐标变量。
行 24	以行 03～09 的类创建新对象 vTs。
行 25～32	以对象 vTs 的 vTs.viewX_To_screenX()方法将视图坐标 x 转换成屏幕坐标 x，以 vTs.viewY_To_screenY()方法将视图坐标 y 转换成屏幕坐标 y。
行 46～47	声明多边形坐标数组变量，并以行 25～32 的屏幕坐标值来初始化。
行 48	以屏幕坐标绘制多边形。

编译程序

输入“javac Ex14_5.java”进行编译，如图 14-14 所示。

图 14-14

运行程序

输入“java Ex14_5”，如图 14-15 所示。

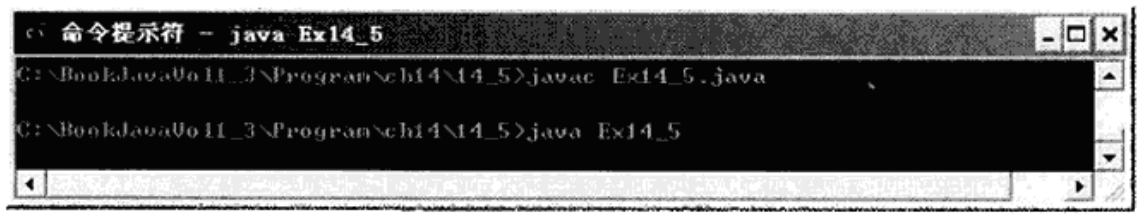


图 14-15

运行结果

运行结果如图 14–16 所示。

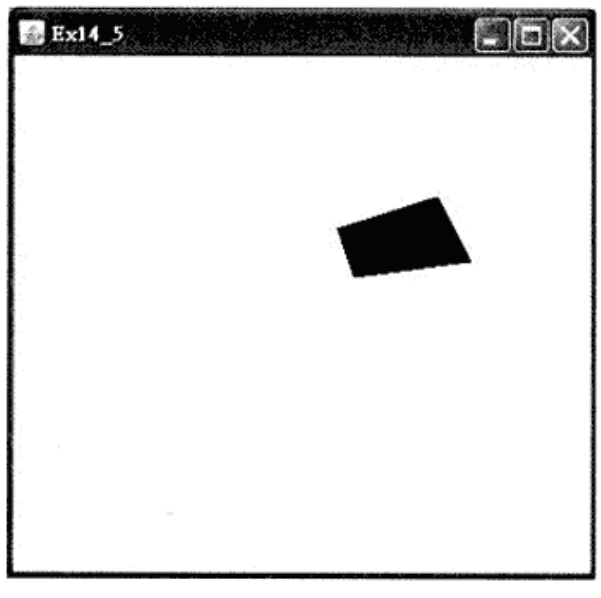


图 14-16

14-6 习题

1. 点/线（Point/Line）如何表示？
2. 向量（Vector）与线（Line）有何不同？
3. 设有向量 A(xa，ya)，其长度如何计算？
4. 设有向量 A(xa，ya)、向量 B(xb，yb)，则向量 AB 如何表示？
5. 绘图与内积（Inner Product）、法线（Normal）有何关系？
6. 两向量的内积与其夹角有何关系？
7. 什么是多边形？
8. 绘图与多边形有何关系？

Chapter 15 缩放与旋转

15-1 简介

笔者一再强调，本书的绘图重点是 3D 绘图，本部分的主题虽然是 2D 平面绘图，但它是为 3D 立体绘图做准备。为了介绍如何进行 3D 立体图形的缩放与旋转，本章先介绍 2D 图形的缩放与旋转，以较简单的层次让读者了解其中的部分原理及程序设计方法。

15-2 多边形缩放

从视觉上来说，缩放是指观察对象时观察点与对象间距离的变化；从绘图设计上来说，缩放却是将对象的坐标乘以某个系数，使对象变大或变小。本节中的多边形缩放是配合键盘事件实现的，按 "↑" 键放大图案，按 "↓" 键缩小图案。

设计范例 70，创建文件 Ex15_2.java 缩放多边形。在 math2D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标，创建文件 zoom_calculator.java，以缩放系数实现多边形缩放。

范例 70 设计文件 Ex15_2.java，其功能是解释如何使用 math2D 包与缩放多边形。

包文件 view_To_screen_coordinates.java：将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch15\15_2\math2D。

```
01 package math2D;

02 import java.awt.*;

03 public class view_To_screen_coordinates {
04   public int viewX_To_screenX(int x) {
05     return x + 350/2;
06   }

07   public int viewY_To_screenY(int y) {
08     return -y + 350/2;
09   }
10 }
```

行 01	创建 math2D 包。
行 03	创建类。
行 04～06	是一个方法，将视图坐标 x 转换成屏幕坐标 x。
行 07～09	是一个方法，将视图坐标 y 转换成屏幕坐标 y。

包文件 zoom_calculator.java：以缩放系数实现多边形缩放，本例目录为 C：\Book-JavaVol1_3\Program\ch15\15_2\math2D。

```
11 package math2D;
12 import java.awt.*;

13 public class zoom_calculator {
14   private double x, y;

15   public void zoomChange(double x, double y, double zf) {
16       this.x = x * zf;
17       this.y = y * zf;
18   }

19   public double getPtX() {return x;}
20   public double getPtY() {return y;}
21 }
```

行 11	创建 math2D 包。
行 14	声明坐标变量。
行 15～18	以缩放系数改变坐标位置。
行 19～20	返回经过缩放处理的坐标。

文件 Ex15_2.java：绘出缩放多边形，按“↑”键放大图案，按“↓”键缩小图案。

```
22 import math2D.*;

23 import java.awt.*;
24 import java.awt.event.*;
25 import java.awt.Graphics;

26 public class Ex15_2 extends Frame implements Runnable {
27   double xa=20, ya=40, xb=30, yb=10, xc=100, yc=20, xd=80, yd=60;
28   int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
29   int xas, yas, xbs, ybs, xcs, ycs, xds, yds;
30   double zf=1;

31   view_To_screen_coordinates vTs = new view_To_screen_coordinates();
32   zoom_calculator zc = new zoom_calculator();

33   public static void main(String args[]) {
34     Ex15_2 workStart=new Ex15_2();
35   }

36   public Ex15_2() {
37     super("Ex15_2");
38     setSize(350, 350);

39     enableEvents(AWTEvent.WINDOW_EVENT_MASK);
40     enableEvents(AWTEvent.KEY_EVENT_MASK);

41     xam=(int)xa; yam=(int)ya;
42     xbm=(int)xb; ybm=(int)yb;
43     xcm=(int)xc; ycm=(int)yc;
44     xdm=(int)xd; ydm=(int)yd;

45     setVisible(true);
```

```
46      new Thread(this).start();
47      }

48  public void processWindowEvent(WindowEvent e) {
49      if(e.getID() == WindowEvent.WINDOW_CLOSING) {
50          System.exit(0);
51      }
52  }

53  public void processKeyEvent(KeyEvent e) {
54      if(e.getID() == KeyEvent.KEY_PRESSED) {
55          switch(e.getKeyCode()) {
56              case KeyEvent.VK_UP:
57                  if(zf<1) zf = 1;
58                  zf = zf + 0.01;
59                  break;
60              case KeyEvent.VK_DOWN:
61                  if(zf>0) zf = 1;
62                  zf = zf - 0.01;
63                  break;
64          }
65          zc.zoomChange(xa,ya,zf);
66          xa=zc.getPtX(); ya=zc.getPtY();

67          zc.zoomChange(xb,yb,zf);
68          xb=zc.getPtX(); yb=zc.getPtY();

69          zc.zoomChange(xc,yc,zf);
70          xc=zc.getPtX(); yc=zc.getPtY();

71          zc.zoomChange(xd,yd,zf);
72          xd=zc.getPtX(); yd=zc.getPtY();

73          xam=(int)xa; yam=(int)ya;
74          xbm=(int)xb; ybm=(int)yb;
75          xcm=(int)xc; ycm=(int)yc;
76          xdm=(int)xd; ydm=(int)yd;
77      }
78  }

79  public void run() {
80      while(true) {
81          xas = vTs.viewX_To_screenX(xam);
82          yas = vTs.viewY_To_screenY(yam);
83          xbs = vTs.viewX_To_screenX(xbm);
84          ybs = vTs.viewY_To_screenY(ybm);
85          xcs = vTs.viewX_To_screenX(xcm);
86          ycs = vTs.viewY_To_screenY(ycm);
87          xds = vTs.viewX_To_screenX(xdm);
88          yds = vTs.viewY_To_screenY(ydm);

89          repaint();
90          try{Thread.sleep(150);}
91          catch(InterruptedException e) {;}
92      }
93  }

94  public void paint(Graphics g) {
95      int pgx[] = {xas, xbs, xcs, xds};
96      int pgy[] = {yas, ybs, ycs, yds};
```

```
97      g.drawPolygon(pgx, pgy, 4);
98    }
99 }
```

行 22	导入 math2D 包。
行 27	声明各点的视图坐标变量并设置初值。
行 28	声明各点的视图坐标更换值变量（从 double 更换成 int）。
行 29	声明各点的屏幕坐标变量。
行 30	声明缩放系数 zf 并设置初值为 1。
行 31	以行 01～10 的类创建新对象 vTs，用于将视图坐标转换成屏幕坐标。
行 32	以行 11～21 的类创建新对象 zc，用于以缩放系数实现多边形缩放。
行 41～44	设置各点的视图坐标更换值的初值。
行 53～64	设置键盘事件，按 "↑" 键放大图案，按 "↓" 键缩小图案。
行 65～72	使用 zc 对象，以缩放系数改变各点的视图坐标。
行 73～76	设置各点的视图坐标的更换值。
行 81～88	使用 vTs 对象将视图坐标的更换值转换成屏幕坐标。
行 95～97	使用经过缩放处理的屏幕坐标绘制多边形。

编译程序

输入 "javac Ex15_2.java" 进行编译，如图 15-1 所示。

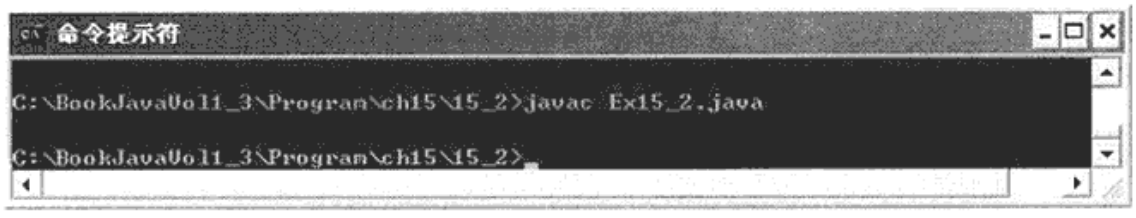


图 15-1

运行程序

输入 "java Ex15_2"，如图 15-2 所示。

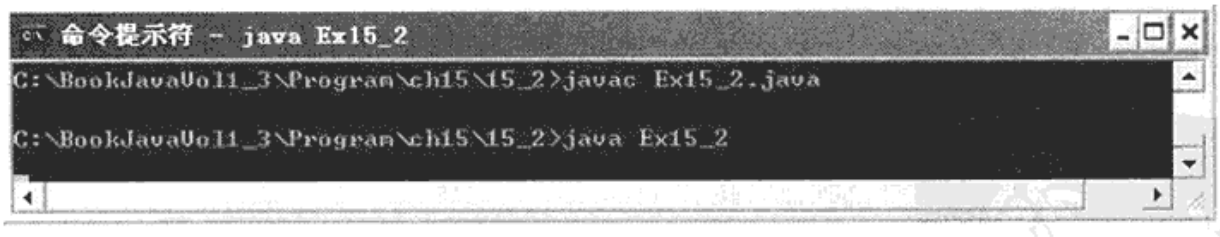


图 15-2

运行结果

运行结果如图 15-3 所示。按 "↑" 键放大图案，按 "↓" 键缩小图案。

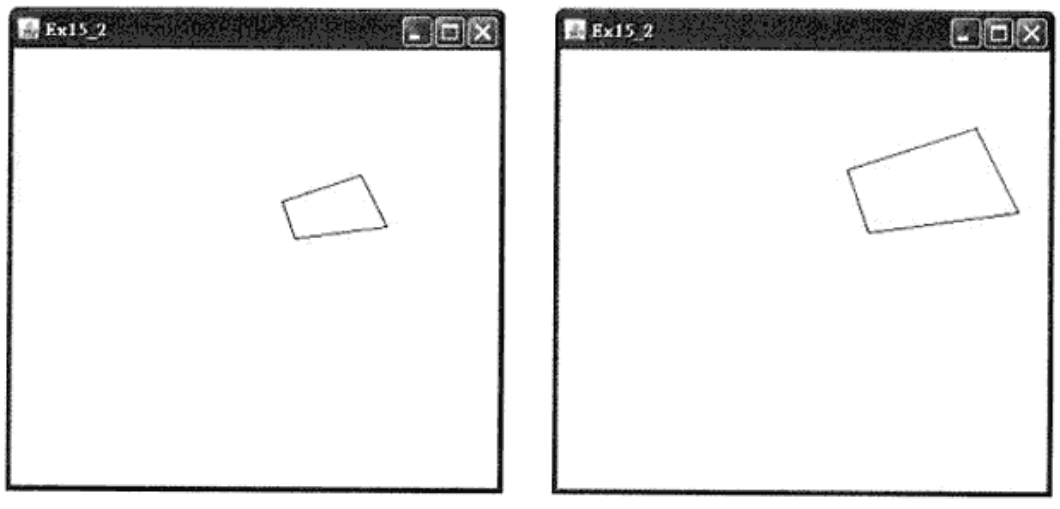


图 15-3

15-3 多边形旋转

在绘制图形中，旋转也是一种重要的坐标转换。如图 15—4 所示，如何从 a 点旋转至 b 点？这里我们将使用三角形的正弦定理、余弦定理来求解。

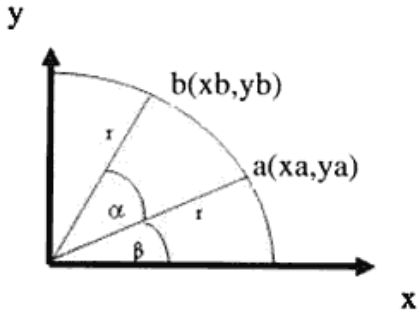


图 15-4

由图 15—4 可得：$xa = r \times \cos\beta$，$ya = r \times \sin\beta$

$xb = r \times \cos(\alpha + \beta)$，$yb = r \times \cos(\alpha + \beta)$

由三角形的正弦定理和余弦定理可得：

$$\begin{aligned} xb &= r \times \cos(\alpha + \beta) \\ &= r \times (\cos\alpha \cos\beta - \sin\alpha \sin\beta) \\ &= r \times \cos\beta \cos\alpha - r \times \sin\beta \sin\alpha \\ &= xa \times \cos\alpha - ya \times \sin\alpha \end{aligned}$$

$$\begin{aligned} yb &= r \times \sin(\alpha + \beta) \\ &= r \times (\sin\alpha \cos\beta + \cos\alpha \sin\beta) \\ &= r \times \cos\beta \sin\alpha + r \times \sin\beta \cos\alpha \\ &= xa \times \sin\alpha + ya \times \cos\alpha \end{aligned}$$

设计范例 71，创建文件 Ex15_3.java 旋转多边形。在 math2D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标，创建文件 rotate_calculator.java，以上述数学公式实现坐标的旋转变换。

范例 71 设计文件 Ex15_3.java，其功能是解释 math2D 包与多边形围绕中心点旋转的应用。

包文件 view_To_screen_coordinates.java：参考范例 70，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch15\15_3\math2D。

包文件 rotate_calculator.java：实现坐标的旋转变换，本例目录为 C：\BookJavaVol1_3\Program\ch15\15_3\math2D。

```
11 package math2D;
12 import java.awt.*;

13 public class rotate_calculator {
14   private double x, y;
15   private double sinAngle, cosAngle;

16   public void rotateChange(double x, double y, double angle) {
17     this.sinAngle = (double)Math.sin(angle);
18     this.cosAngle = (double)Math.cos(angle);

19     this.x = x*cosAngle - y*sinAngle;
20     this.y = x*sinAngle + y*cosAngle;
21   }

22   public double getPtX() {return x;}
23   public double getPtY() {return y;}
24 }
```

行 11	创建 math2D 包。
行 14	声明坐标变量。
行 15	声明三角函数 sinα、cosα 的变量。
行 16～21	将参数坐标(x，y) 旋转 α 后，计算新坐标(this.x，this.y)。其中，α 为参数 angle。
行 22～23	返回新坐标。

文件 Ex15_3.java：绘出旋转多边形。

```
25 import math2D.*;

26 import java.awt.*;
27 import java.awt.event.*;
28 import java.awt.Graphics;

29 public class Ex15_3 extends Frame implements Runnable {
30   double xa=20, ya=40, xb=30, yb=10, xc=100, yc=20, xd=80, yd=60;
31   int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
32   int xas, yas, xbs, ybs, xcs, ycs, xds, yds;

33   view_To_screen_coordinates vTs = new view_To_screen_coordinates();
34   rotate_calculator rc = new rotate_calculator();

35   public static void main(String args[]) {
36     Ex15_3 workStart=new Ex15_3();
37   }

38   public Ex15_3() {
39     super("Ex15_3");
40     setSize(350, 350);

41     enableEvents(AWTEvent.WINDOW_EVENT_MASK);

42     xam=(int)xa; yam=(int)ya;
```

```
43     xbm=(int)xb; ybm=(int)yb;
44     xcm=(int)xc; ycm=(int)yc;
45     xdm=(int)xd; ydm=(int)yd;

46     setVisible(true);
47     new Thread(this).start();
48     }

49   public void processWindowEvent(WindowEvent e) {
50     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
51         System.exit(0);
52     }
53   }

54   public void run() {
55     while(true) {
56       rc.rotateChange(xa,ya,0.03);
57       xa=rc.getPtX(); ya=rc.getPtY();

58       rc.rotateChange(xb,yb,0.03);
59       xb=rc.getPtX(); yb=rc.getPtY();

60       rc.rotateChange(xc,yc,0.03);
61       xc=rc.getPtX(); yc=rc.getPtY();

62       rc.rotateChange(xd,yd,0.03);
63       xd=rc.getPtX(); yd=rc.getPtY();

64       xam=(int)xa; yam=(int)ya;
65       xbm=(int)xb; ybm=(int)yb;
66       xcm=(int)xc; ycm=(int)yc;
67       xdm=(int)xd; ydm=(int)yd;

68       xas = vTs.viewX_To_screenX(xam);
69       yas = vTs.viewY_To_screenY(yam);
70       xbs = vTs.viewX_To_screenX(xbm);
71       ybs = vTs.viewY_To_screenY(ybm);
72       xcs = vTs.viewX_To_screenX(xcm);
73       ycs = vTs.viewY_To_screenY(ycm);
74       xds = vTs.viewX_To_screenX(xdm);
75       yds = vTs.viewY_To_screenY(ydm);

76       repaint();
77       try{Thread.sleep(150);}
78       catch(InterruptedException e) {;}
79     }
80   }

81   public void paint(Graphics g) {
82     int pgx[] = {xas, xbs, xcs, xds};
83     int pgy[] = {yas, ybs, ycs, yds};
84     g.drawPolygon(pgx, pgy, 4);
85   }
86 }
```

行 56～63	调用行 11～24，将多边形各点旋转 0.03。
行 81～85	显示出连续旋转的多边形。

编译程序

输入“javac Ex15_3.java”进行编译，如图 15-5 所示。

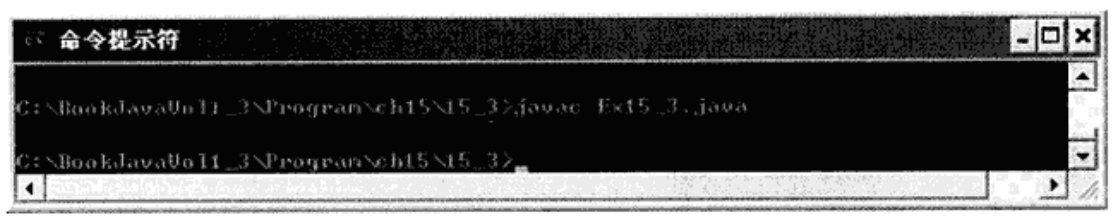


图 15-5

运行程序

输入“java Ex15_3”，如图 15-6 所示。

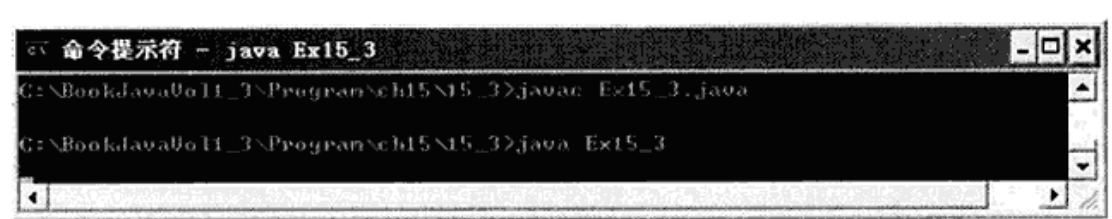


图 15-6

运行结果

运行结果如图 15-7 所示。

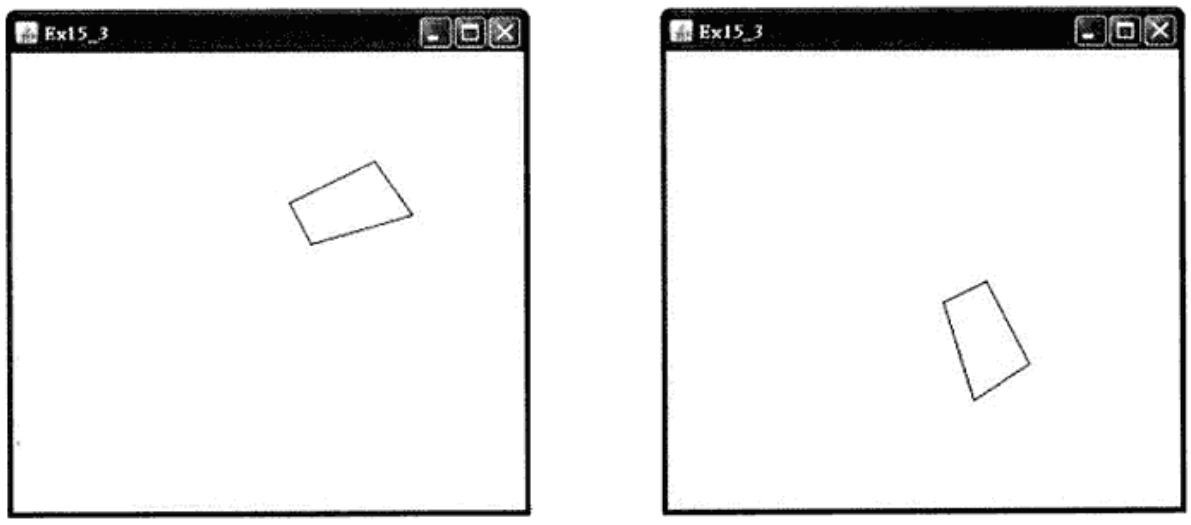


图 15-7

15-4 习题

1. 从视觉上来说，缩放有何意义？
2. 从绘图设计上来说，缩放有何意义？
3. 什么是三角形的正弦定理、余弦定理？
4. 如何利用三角形的正弦定理、余弦定理来产生旋转坐标？

PART 05

3D绘图设计

本部分内容是3D绘图设计，如果我们使用软件包，则可以在非常短的时间内上手，但终究是在使用别人的东西，并无法窥得3D绘图的奥秘，以后在设计上也无发展潜力。

以笔者多年的教学经验，熟练使用别人精心设计的软件包是必须的，但从基础入手，进行深度解析也是不可或缺的，尤其是大专院校的学生，毕业后多少要肩负一些研发的责任，更需要厚实的基础认识。

为了让读者深刻理解3D绘图的基础原理与程序设计过程，本书在解说时对步骤过程进行了清晰的刻画，凡是涉及软件包的部分均力求避免，以免读者错过设计的每一道痕迹。

Chapter 16　3D坐标

图案放置之处是一张纸、是屏幕、是照片等，无论是何物，都有一些共同的特征，那就是2D平面。如果要绘制3D立体图案，首先要解决的就是如何将3D坐标投影转换成2D坐标。

Chapter 17　3D图形的旋转

在第16章中，我们探讨了如何将3D图形投影到2D平面上，其表现方式是静态的。本章将讨论动态的3D图形，它是随时间变化的，因此也有人称之为4D图形。通过三角函数表达式，可计算围绕x、y、z轴旋转的位置坐标。

Chapter 18　法线与隐藏线

实例对象并非都是透明的。若不是透明的，则我们只能看到其正面，无法看到被遮挡的背面，在绘图上称它为隐藏部分，因此在绘制时必须将隐藏部分做消除或浅细处理。在设计时，必须先判别哪一些是正面显示部分，哪一些是背面隐藏部分。本章将通过范例进行详尽解说。

Chapter 16　3D 坐标

16-1　简介

图案放置之处是一张纸、是屏幕、是照片等，无论是何物，都有一些共同的特征，那就是 2D 平面。如果要绘制 3D 立体图案，首先要解决的就是如何将 3D 坐标投影转换成 2D 坐标。

16-2　立体空间坐标

3D 对象的绘制，是将对象各点的立体坐标(x,y,z)投影到二维空间的平面坐标 (x',y')。本节将列出有关的原理及公式，同时以范例绘制 3D 水平多边形、3D 立体多边形，读者能轻易地了解其中的奥秘。

16-2-1　z 轴坐标

在本书第 4 部分中，我们已详细探讨了 2D 绘图，它只有 x 与 y 方向的坐标，而 3D 图案必须增设 z 方向的坐标，其投影在 2D 平面上的模式如图 16–1 所示。

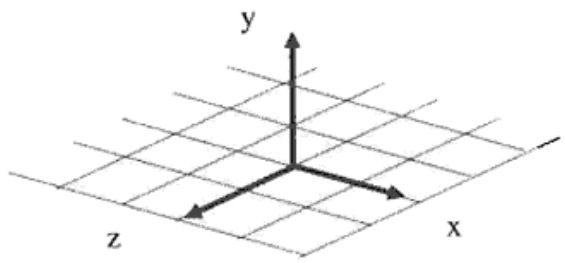


图 16-1

在立体空间中以视图坐标而言，x 轴越向右，其值越大；y 轴越向上，其值越大；z 轴指向观看者（读者）的眼睛，越靠近观看者，其值越大。

16-2-2　视图窗口

如图 16–2 所示，镜头（Camera）是观看者的位置，视图窗口（View Window）是 3D 空间中的一个屏幕。

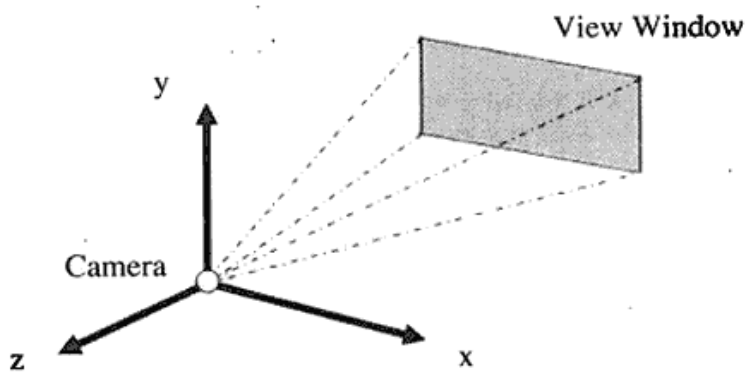


图 16-2

将 3D 对象投影到视图窗口，可假设该对象各点到镜头有连接虚线（如图 16–3 所示），该虚线与视图窗口的交点所呈现的图案，即为 3D 图案。对象距离镜头越近，图像越大；对象距离镜头越远，图像越小，呈现视觉景深（Field Depth）。

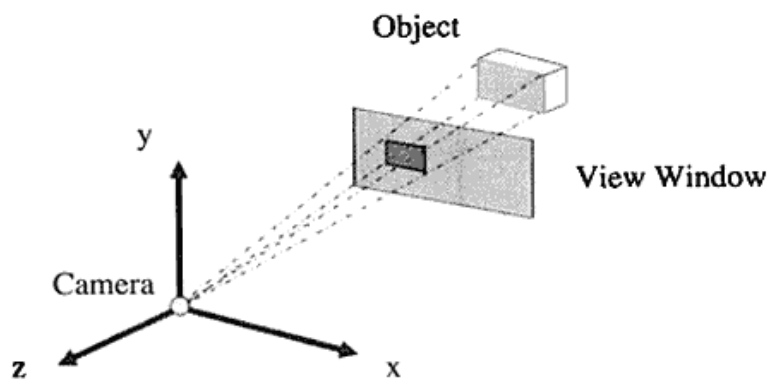


图 16-3

16-2-3 投影坐标

1 外端投影（Outer Project）：由图 16–3 可得图 16–4，3D 对象的各点 v(x,y,z)从外端投影到 2D 视图窗口的各点 v'（x',y'），在此为了解说方便，我们以 v 代表 3D 对象的各点(x,y,z)；以 v'代表 2D 视图窗口的各点(x',y')。

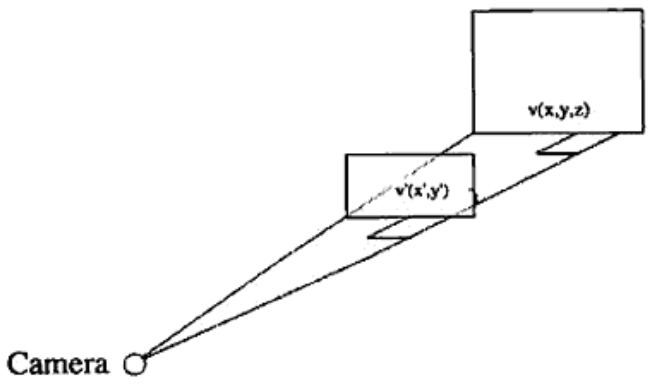


图 16-4

如图 16–5 所示，设置 2D 视图窗口的 Z 轴位置为 "0"，则观看者到视图窗口的距离是 "d"，对象的 z 轴位置是 "–z"。由此可得：

$$v'/d = v/(d-(-z))$$

即

$$v' = vd / (d-(-z))$$

故 3D 坐标 v（x,y,z）投影到 2D 平面坐标 v'（x',y'）的关系为

$$x' = xd / (d-(-z)) \qquad \text{(式 16-1)}$$
$$y' = yd / (d-(-z))$$

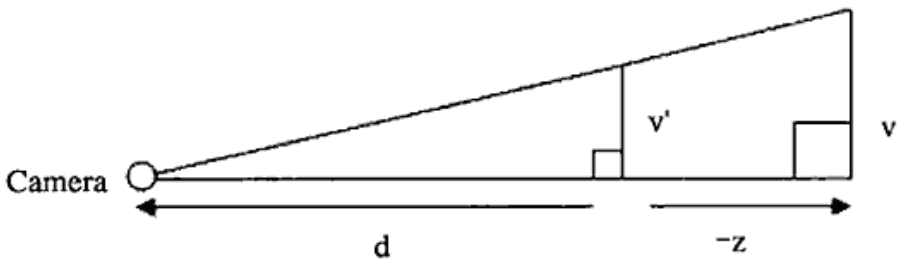


图 16-5

2　内端投影（Inner Project）：由图 16—6 可得图 16—7，3D 对象的各点 v(x,y,z)从内端投影到视图 2D 窗口的各点 v'（x',y'）。

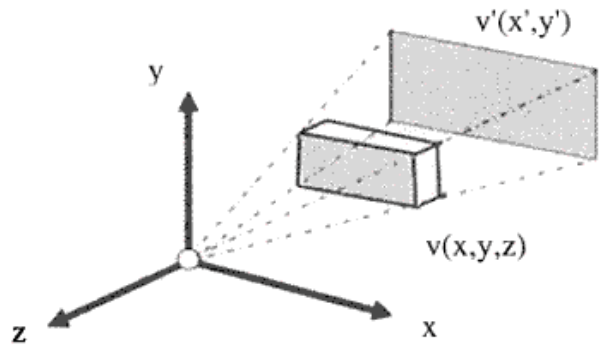


图 16-6

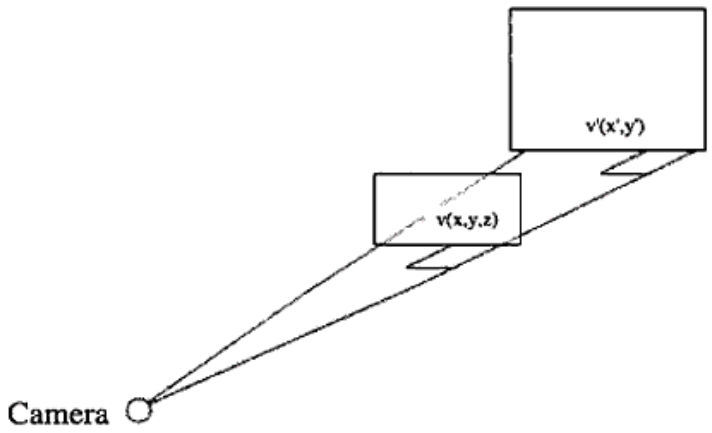


图 16-7

如图 16—8 所示，假设 2D 视图窗口的 z 轴位置为 "0"，则观看者到视图窗口的距离是 "d"，对象的 z 轴位置是 "z"。由此可得：

$$v'/d = v/(d-z)$$

即

$$v' = vd/(d-z)$$

故 3D 坐标 v(x,y,z)投影到 2D 平面坐标 v'（x',y'）的关系为

$$x' = xd / (d-z) \qquad \text{(式 16-2)}$$
$$y' = yd / (d-z)$$

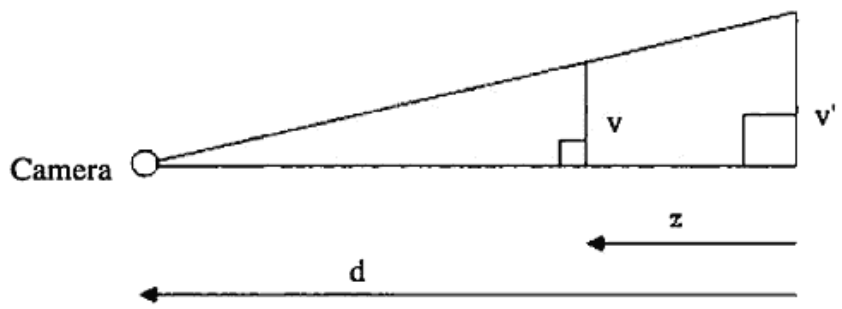


图 16-8

式 16–1 和式 16–2 中的 x、y、z 均为已知，只要 d 也是已知的，就可求出 v（x,y,z）投影在 2D 平面的坐标 v'（x',y'）。我们将在 16–2–4 节讨论如何计算 d。

16-2-4 视图宽与视距

如图 16–9 所示，w 为视图的宽度，d 为视图窗口到观看者间的距离，θ 为观看者的水平视角，水平视角一般介于 45° ～90°，本部分的范例中使用 80°。

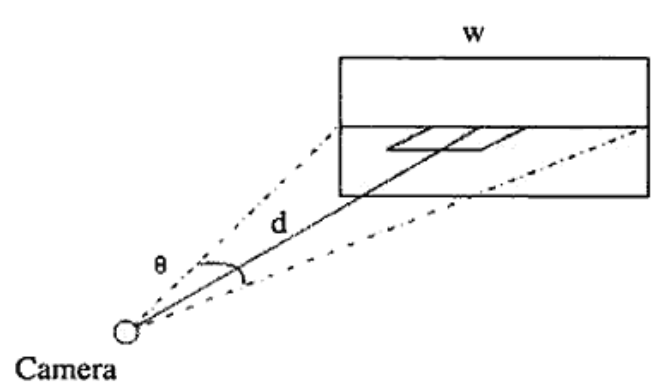


图 16-9

如图 16–10 所示，由三角形直角公式可得：

$$\tan(\theta/2) = (w/2) / d$$

即 $$d = (w/2) / \tan(\theta/2)$$ （式 16–3）

由于 w、θ 已知，可求出 d。

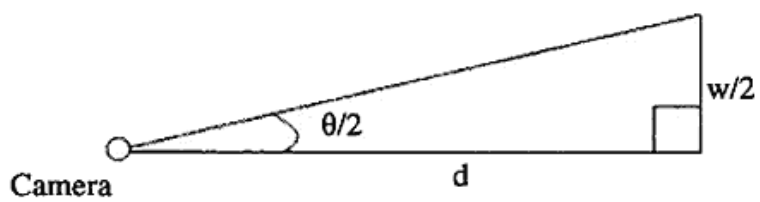


图 16-10

在式 16–1、式 16–2、式 16–3 中的 x、y、z、d 均为已知，因此可求得 v（x,y,z）投影在 2D 平面的坐标 v'（x',y'）。

16-3 绘制 3D 水平多边形

依照 16–2 节所述的概念原理，本节将其融入程序设计，使用 Java 绘制第一个 3D 图形。为了解说方便，本节选择绘制一个简单的 3D 水平多边形，如图 16–11 所示。

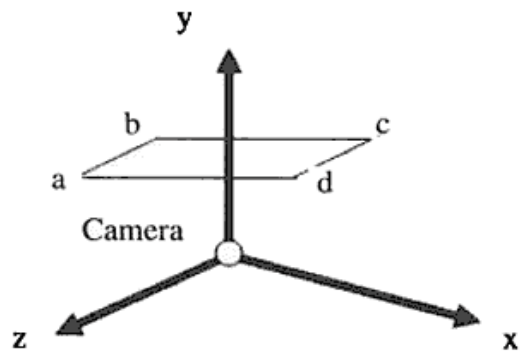


图 16-11

其绘制结果将如图 16—12 所示。沿 z 轴观看，较近的线段 ad 较长；较远的线段 bc 较短。

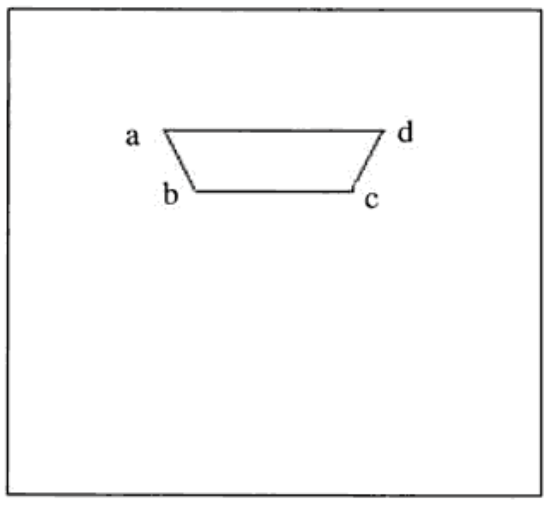


图 16-12

设计范例 72，创建文件 Ex16_3.java 绘制立体的水平多边形（如图 16—11 所示）。在 math3D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标，创建文件 project3D_To_2D.java，将 3D 图形各点的坐标投影到 2D 平面上。

范例 72 设计文件 Ex16_3.java，其功能是解释如何使用 math3D 包与绘制立体的水平多边形（运行结果如图 16—12 所示）。

包文件 view_To_screen_coordinates.java：将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch16\16_3\math3D。

```
01 package math3D;

02 import java.awt.*;

03 public class view_To_screen_coordinates {
04   public int viewX_To_screenX(int x) {
05     return x + 400/2;
06   }

07   public int viewY_To_screenY(int y) {
08     return -y + 400/2;
09   }
```

```
10 }
```

行 01	创建 math3D 包。
行 03	创建类。
行 04～06	将视图坐标 x 转换成屏幕坐标 x。
行 07～09	将视图坐标 y 转换成屏幕坐标 y。

包文件 project3D_To_2D.java：将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v' (x',y')，本例目录为 C：\BookJavaVol1_3\Program\ch16\16_3\math3D。

```
11 package math3D;
12 import java.awt.*;

13 public class project3D_To_2D {
14   private double x, y;
15   private double d;

16   public double projectPtX(double x, double y, double z) {
17     d = ((double)400/2)/(double)Math.tan(80/2);
18     this.x = d*x/(d- (-z));
19     return this.x;
20   }

21   public double projectPtY(double x, double y, double z) {
22     d = ((double)400/2)/(double)Math.tan(80/2);
23     this.y = d*y/(d- (-z));
24     return this.y;
25   }
26 }
```

行 11	创建 math3D 包。
行 14	声明 2D 平面坐标(x, y) 的变量。
行 15	声明视图窗口视距的变量。
行 16～20	将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v' (x',y')，运行完毕后返回 x'（如式 16-3 与式 16-1）。
行 21～24	将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v' (x',y')，运行完毕后返回 y'（如式 16-3 与式 16-1）。

文件 Ex16_3.java：绘出立体的水平多边形。

```
27 import math3D.*;

28 import java.awt.*;
29 import java.awt.event.*;
30 import java.awt.Graphics;

31 public class Ex16_3 extends Frame implements Runnable {
32   double xa=-50, ya=50, za=50, xb=-50, yb=50, zb=-50;
33   double xc=50, yc=50, zc=-50, xd=50, yd=50, zd=50;
34   int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
35   int xas, yas, xbs, ybs, xcs, ycs, xds, yds;

36   view_To_screen_coordinates vTs = new view_To_screen_coordinates();
```

```
project3D_To_2D pr = new project3D_To_2D();

public static void main(String args[]) {
   Ex16_3 workStart=new Ex16_3();
}

public Ex16_3() {
   super("Ex16_3");
   setSize(400, 400);

   enableEvents(AWTEvent.WINDOW_EVENT_MASK);

   xam = (int)pr.projectPtX(xa,ya,za);
   yam = (int)pr.projectPtY(xa,ya,za);

   xbm = (int)pr.projectPtX(xb,yb,zb);
   ybm = (int)pr.projectPtY(xb,yb,zb);

   xcm = (int)pr.projectPtX(xc,yc,zc);
   ycm = (int)pr.projectPtY(xc,yc,zc);

   xdm = (int)pr.projectPtX(xd,yd,zd);
   ydm = (int)pr.projectPtY(xd,yd,zd);

   xas = vTs.viewX_To_screenX(xam);
   yas = vTs.viewY_To_screenY(yam);

   xbs = vTs.viewX_To_screenX(xbm);
   ybs = vTs.viewY_To_screenY(ybm);

   xcs = vTs.viewX_To_screenX(xcm);
   ycs = vTs.viewY_To_screenY(ycm);

   xds = vTs.viewX_To_screenX(xdm);
   yds = vTs.viewY_To_screenY(ydm);

   setVisible(true);
   new Thread(this).start();
   }

public void processWindowEvent(WindowEvent e) {
   if(e.getID() == WindowEvent.WINDOW_CLOSING) {
       System.exit(0);
   }
}

public void run() {
      repaint();
}

public void paint(Graphics g) {
   int pgx[] = {xas, xbs, xcs, xds};
   int pgy[] = {yas, ybs, ycs, yds};
   g.drawPolygon(pgx, pgy, 4);
}
```

```
77 }
```

行 27	导入 math3D 包。
行 32～33	声明水平多边形各点坐标的变量，并设置其坐标值。
行 34	声明各点辅助坐标的变量。
行 36	以行 01～10 的类创建新对象 vTs，用于将视图坐标转换成屏幕坐标。
行 37	以行 11～26 的类创建新对象 pr，用于将 3D 对象各点的坐标投影到 2D 平面。
行 45～52	将水平多边形的 3D 坐标投影到 2D 平面，再以辅助坐标(xm，ym) 读取各点的 2D 坐标。
行 53～60	将各点的视图坐标转换成屏幕坐标。
行 72～76	以屏幕坐标绘制水平多边形。

编译程序

输入“javac Ex16_3.java”进行编辑，如图 16-13 所示。

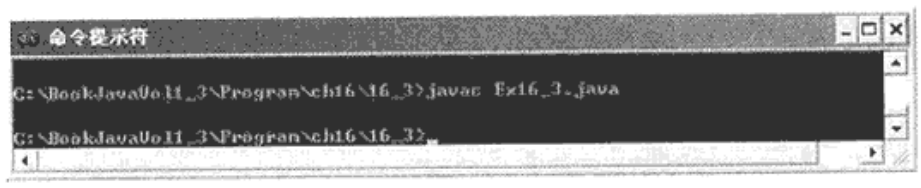


图 16-13

运行程序

输入“java Ex16_3”，如图 16-14 所示。

图 16-14

运行结果

运行结果如图 16-15 所示。

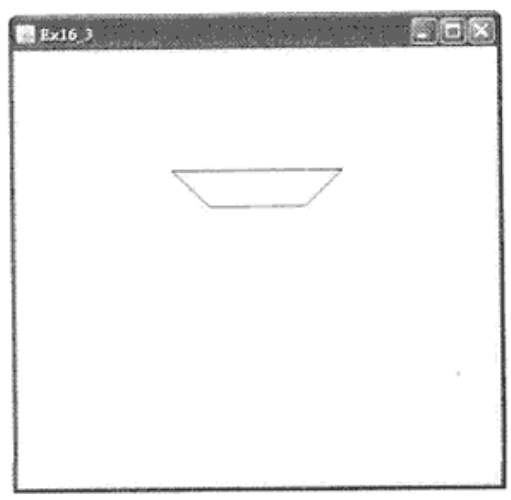


图 16-15

16-4 绘制 3D 立体多边形

继 16–3 节绘制水平多边形后，本节将介绍如何绘制 3D 立体多边形。当完成 3D 立体多边形的绘制时，即已了解简易 3D 立体图形的设计方法。如图 16–16 所示，原点(0,0,0)位于立方体中央，各轴距为 50 个单位长度（如 a 点为(−50,50,50)，b 点为(−50,50,−50)，其他点依此类推）。

其绘制结果将如图 16–17 所示。沿 z 轴观看，近端面 adhe 将较宽长；远端面 bcgf 将较短窄。

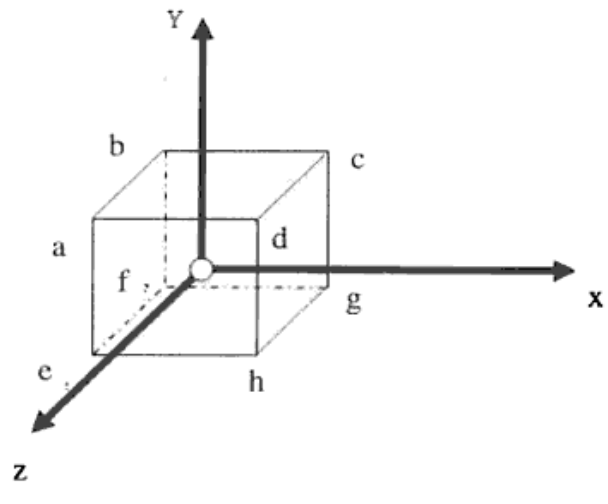


图 16-16

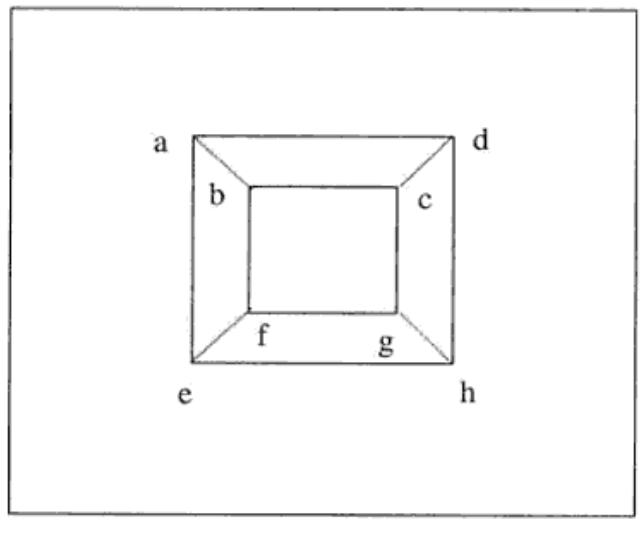


图 16-17

设计范例 73，创建文件 Ex16_4.java 绘制 3D 立体多边形（如图 16–16 所示）。在 math3D 包创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标，创建文件 project3D_To_2D.java，将 3D 图形各点的坐标投影到 2D 平面上。

范例 73 设计文件 Ex16_4.java，其功能是解释如何使用 math3D 包与绘制 3D 立体多边形（运行结果将如图 16–17 所示）。

包文件 view_To_screen_coordinates.java：参考范例 72，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch16\16_4\math3D。

包文件 project3D_To_2D.java：参考范例 72，将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v′ (x′,y′)，本例目录为 C：\BookJavaVol1_3\Program\ch16\16_4\math3D。

文件 Ex16_4.java：绘出立体多边形。

```
027 import math3D.*;

028 import java.awt.*;
029 import java.awt.event.*;
030 import java.awt.Graphics;

031 public class Ex16_4 extends Frame implements Runnable {
032   double xa=-50, ya=50, za=50, xb=-50, yb=50, zb=-50;
033   double xc=50, yc=50, zc=-50, xd=50, yd=50, zd=50;
034   double xe=-50, ye=-50, ze=50, xf=-50, yf=-50, zf=-50;
035   double xg=50, yg=-50, zg=-50, xh=50, yh=-50, zh=50;

036   int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
037   int xem, yem, xfm, yfm, xgm, ygm, xhm, yhm;

038   int xas, yas, xbs, ybs, xcs, ycs, xds, yds;
039   int xes, yes, xfs, yfs, xgs, ygs, xhs, yhs;
```

```
040   view_To_screen_coordinates vTs = new view_To_screen_coordinates();
041   project3D_To_2D pr = new project3D_To_2D();

042   public static void main(String args[]) {
043      Ex16_4 workStart=new Ex16_4();
044   }

045   public Ex16_4() {
046      super("Ex16_4");
047      setSize(400, 400);

048      enableEvents(AWTEvent.WINDOW_EVENT_MASK);

049      xam = (int)pr.projectPtX(xa,ya,za);
050      yam = (int)pr.projectPtY(xa,ya,za);

051      xbm = (int)pr.projectPtX(xb,yb,zb);
052      ybm = (int)pr.projectPtY(xb,yb,zb);

053      xcm = (int)pr.projectPtX(xc,yc,zc);
054      ycm = (int)pr.projectPtY(xc,yc,zc);

055      xdm = (int)pr.projectPtX(xd,yd,zd);
056      ydm = (int)pr.projectPtY(xd,yd,zd);

057      xem = (int)pr.projectPtX(xe,ye,ze);
058      yem = (int)pr.projectPtY(xe,ye,ze);

059      xfm = (int)pr.projectPtX(xf,yf,zf);
060      yfm = (int)pr.projectPtY(xf,yf,zf);

061      xgm = (int)pr.projectPtX(xg,yg,zg);
062      ygm = (int)pr.projectPtY(xg,yg,zg);

063      xhm = (int)pr.projectPtX(xh,yh,zh);
064      yhm = (int)pr.projectPtY(xh,yh,zh);

065      xas = vTs.viewX_To_screenX(xam);
066      yas = vTs.viewY_To_screenY(yam);

067      xbs = vTs.viewX_To_screenX(xbm);
068      ybs = vTs.viewY_To_screenY(ybm);

069      xcs = vTs.viewX_To_screenX(xcm);
070      ycs = vTs.viewY_To_screenY(ycm);

071      xds = vTs.viewX_To_screenX(xdm);
072      yds = vTs.viewY_To_screenY(ydm);

073      xes = vTs.viewX_To_screenX(xem);
074      yes = vTs.viewY_To_screenY(yem);

075      xfs = vTs.viewX_To_screenX(xfm);
076      yfs = vTs.viewY_To_screenY(yfm);

077      xgs = vTs.viewX_To_screenX(xgm);
078      ygs = vTs.viewY_To_screenY(ygm);

079      xhs = vTs.viewX_To_screenX(xhm);
080      yhs = vTs.viewY_To_screenY(yhm);
```

```
081     setVisible(true);
082     new Thread(this).start();
083     }

084  public void processWindowEvent(WindowEvent e) {
085     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
086         System.exit(0);
087     }
088  }

089  public void run() {
090        repaint();
091  }

092  public void paint(Graphics g) {
093     g.drawLine(xas, yas, xbs, ybs);
094     g.drawLine(xbs, ybs, xcs, ycs);
095     g.drawLine(xcs, ycs, xds, yds);
096     g.drawLine(xds, yds, xas, yas);

097     g.drawLine(xes, yes, xfs, yfs);
098     g.drawLine(xfs, yfs, xgs, ygs);
099     g.drawLine(xgs, ygs, xhs, yhs);
100     g.drawLine(xhs, yhs, xes, yes);

101     g.drawLine(xas, yas, xes, yes);
102     g.drawLine(xbs, ybs, xfs, yfs);
103     g.drawLine(xcs, ycs, xgs, ygs);
104     g.drawLine(xds, yds, xhs, yhs);
105  }
106 }
```

行 027	导入 math3D 包。
行 032～035	声明立体多边形各点坐标的变量，并设置其坐标值。
行 036～037	声明各点辅助坐标的变量。
行 040	以行 01～10 的类创建新对象 vTs，用于将视图坐标转换成屏幕坐标。
行 041	以行 11～26 的类创建新对象 pr，用于将 3D 对象各点的坐标投影到 2D 平面。
行 049～064	将立体多边形的 3D 坐标投影到 2D 平面，再以辅助坐标(xm，ym) 读取各点的 2D 坐标。
行 065～080	将各点的视图坐标转换成屏幕坐标。
行 093～104	以屏幕坐标绘制立体多边形。

编译程序

输入“javac Ex16_4.java”进行编译，如图 16-18 所示。

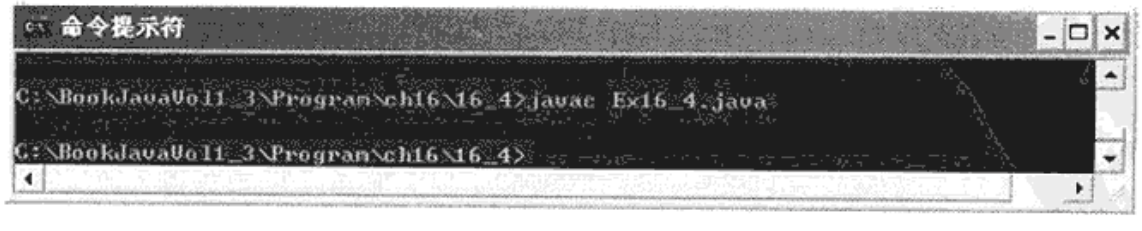


图 16-18

运行程序

输入“java Ex16_4”，如图 16-19 所示。

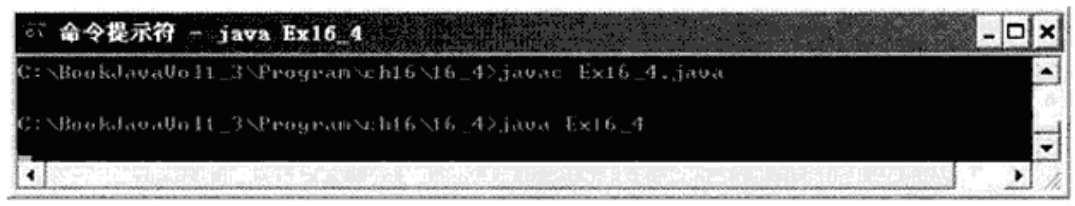


图 16-19

运行结果

运行结果如图 16-20 所示。

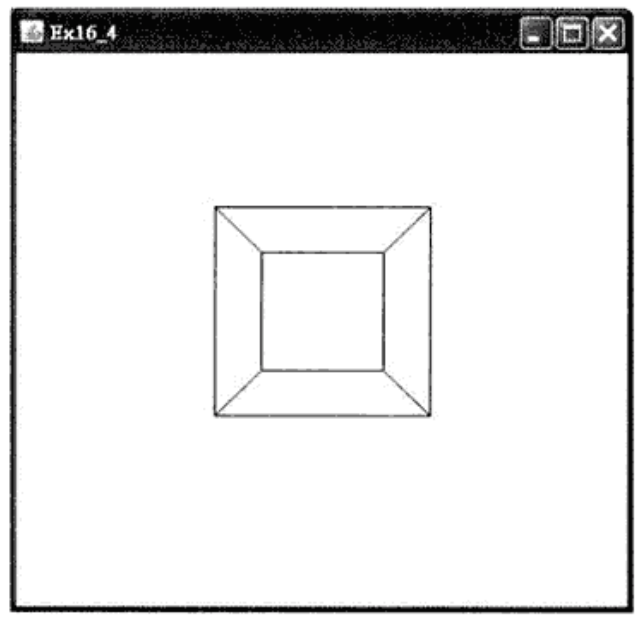


图 16-20

16-5 习题

1. 试述 3D 绘图的意义。
2. 试述 3D 绘图中 x 轴、y 轴、z 轴的意义。
3. 什么是视觉景深(Field Depth)？
4. 外端投影中 3D 坐标 v（x,y,z）与 2D 坐标 v'（x',y'）的关系是怎么样的？
5. 内端投影中 3D 坐标 v（x,y,z）与 2D 坐标 v'（x',y'）的关系是怎么样的？
6. 如何计算视距？

Chapter 17　3D 图形的旋转

17-1 简介

在第 16 章中，我们探讨了如何将 3D 图形投影到 2D 平面上，其表现方式是静态的。本章将讨论动态的 3D 图形，它是随时间变化的，因此也有人称之为 4D 图形。通过三角函数表达式，可计算围绕 x、y、z 轴旋转的位置坐标。

17-2 立体图形的旋转

在 15–3 节，我们已对 2D 平面多边形的旋转进行了详尽的探讨，本节将对立体图形的旋转做进一步的讨论。参考 15–3 节的计算公式，图 17–1 可视为围绕 z 轴旋转，因为 xa 移至 xb，ya 移至 yb，而 zb=za，即 z 轴坐标无改变。

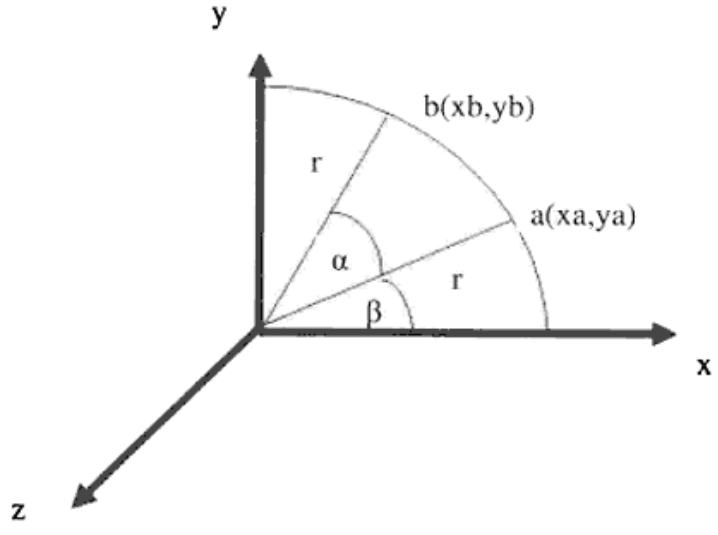


图 17-1

由图 17–1 可得：　$xa = r \times \cos\beta$，　$ya = r \times \sin\beta$

$xb = r \times \cos(\alpha+\beta)$，　$yb = r \times \cos(\alpha+\beta)$

依三角形的正弦定理、余弦定理可得：

$$\begin{aligned} xb &= r \times \cos(\alpha+\beta) \\ &= r \times (\cos\alpha \cos\beta - \sin\alpha \sin\beta) \\ &= r \times \cos\beta \cos\alpha - r \times \sin\beta \sin\alpha \\ &= xa \times \cos\alpha - ya \times \sin\alpha \end{aligned}$$

$$
\begin{aligned}
yb &= r \times \sin(\alpha+\beta) \\
&= r \times (\sin\alpha \cos\beta + \cos\alpha \sin\beta) \\
&= r \times \cos\beta \sin\alpha + r \times \sin\beta \cos\alpha \\
&= xa \times \sin\alpha + ya \times \cos\alpha \\
zb &= za
\end{aligned}
$$

同理，围绕 y 轴旋转的表达式为

$$
\begin{aligned}
zb &= za \times \cos\alpha - xa \times \sin\alpha \\
xb &= za \times \sin\alpha + xa \times \cos\alpha \\
yb &= ya
\end{aligned}
$$

围绕 x 轴旋转的表达式为

$$
\begin{aligned}
yb &= ya \times \cos\alpha - za \times \sin\alpha \\
zb &= ya \times \sin\alpha + za \times \cos\alpha \\
xb &= xa
\end{aligned}
$$

17-3 绕 y 轴旋转

根据 17—2 节的计算公式设计范例 74，创建文件 Ex17_3.java 绘制 3D 立体多边形围绕 y 轴旋转。在 math3D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标；创建文件 project3D_To_2D.java，将 3D 图形各点的坐标投影到 2D 平面上；创建文件 rotate_calculator.java，将 3D 图形围绕 x 轴（或 y 轴、或 z 轴）旋转 α 角度。

范例 74 设计文件 Ex17_3.java，其功能是解释如何使用 math3D 包与绘制 3D 立体多边形围绕 y 轴旋转。

包文件 view_To_screen_coordinates.java：将视图坐标转换成屏幕坐标，本例目录为 C：\Book-JavaVol1_3\Program\ch17\17_3\math3D。

```
01 package math3D;

02 import java.awt.*;

03 public class view_To_screen_coordinates {
04   public int viewX_To_screenX(int x) {
05     return x + 400/2;
06   }

07   public int viewY_To_screenY(int y) {
08     return -y + 400/2;
09   }
10 }
```

行 01	创建 math3D 包。
行 03	创建类。
行 04～06	将视图坐标 x 转换成屏幕坐标 x。
行 07～09	将视图坐标 y 转换成屏幕坐标 y。

包文件 project3D_To_2D.java：将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v' (x',y')，本例目录为 C：\BookJavaVol1_3\Program\ch17\17_3\math3D。

```
11 package math3D;
12 import java.awt.*;

13 public class project3D_To_2D {
14   private double x, y;
15   private double d;

16   public double projectPtX(double x, double y, double z) {
17     d = ((double)400/2)/(double)Math.tan(80/2);
18     this.x = d*x/(d- (-z));
19     return this.x;
20   }

21   public double projectPtY(double x, double y, double z) {
22     d = ((double)400/2)/(double)Math.tan(80/2);
23     this.y = d*y/(d- (-z));
24     return this.y;
25   }
26 }
```

行 11	创建 math3D 包。
行 14	声明 2D 平面坐标(x，y) 的变量。
行 15	声明平面视图窗口视距的变量。
行 16～20	将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v' (x',y')，运行完毕后返回 x'。
行 21～25	将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v' (x',y')，运行完毕后返回 y'。

包文件 rotate_calculator.java：将 3D 图形围绕 x 轴（或 y 轴、或 z 轴)旋转 α 角度，本例目录为 C：\BookJavaVol1_3\Program\ch17\17_3\math3D）。

```
27 package math3D;
28 import java.awt.*;

29 public class rotate_calculator {
30   private double x, y, z;
31   private double sinAngle, cosAngle;

32   public void rotateX(double x, double y, double z, double angle) {
33     this.sinAngle = (double)Math.sin(angle);
34     this.cosAngle = (double)Math.cos(angle);
35     this.y = y*cosAngle - z*sinAngle;
36     this.z = y*sinAngle + z*cosAngle;
37     this.x = x;
38   }

39   public void rotateY(double x, double y, double z, double angle) {
40     this.sinAngle = (double)Math.sin(angle);
41     this.cosAngle = (double)Math.cos(angle);
42     this.z = z*cosAngle - x*sinAngle;
43     this.x = z*sinAngle + x*cosAngle;
44     this.y = y;
45   }

46   public void rotateZ(double x, double y, double z, double angle) {
```

```
47      this.sinAngle = (double)Math.sin(angle);
48      this.cosAngle = (double)Math.cos(angle);
49      this.x = x*cosAngle - y*sinAngle;
50      this.y = x*sinAngle + y*cosAngle;
51      this.z = z;
52    }

53    public double getPtX() {return x;}
54    public double getPtY() {return y;}
55    public double getPtZ() {return z;}
56 }
```

行 27	创建 math3D 包。
行 30	声明坐标变量。
行 31	声明三角函数 sinα、cosα 的变量。
行 32～38	将参数坐标(x，y，z)围绕 x 轴旋转 α 后，计算新坐标(this.x，this.y，this.z)，其中 α 为参数 angle。
行 39～45	将参数坐标(x，y，z)围绕 y 轴旋转 α 后，计算新坐标(this.x，this.y，this.z)，其中 α 为参数 angle。
行 46～52	将参数坐标(x，y，z)围绕 z 轴旋转 α 后，计算新坐标(this.x，this.y，this.z)，其中 α 为参数 angle。
行 53～55	返回新坐标。

文件 Ex17_3.java：绘制 3D 图形围绕 y 轴旋转。

```
057 import math3D.*;
058 import java.awt.*;
059 import java.awt.event.*;
060 import java.awt.Graphics;

061 public class Ex17_3 extends Frame implements Runnable {
062   double xa=-50, ya=50, za=50, xb=-50, yb=50, zb=-50;
063   double xc=50, yc=50, zc=-50, xd=50, yd=50, zd=50;
064   double xe=-50, ye=-50, ze=50, xf=-50, yf=-50, zf=-50;
065   double xg=50, yg=-50, zg=-50, xh=50, yh=-50, zh=50;

066   int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
067   int xem, yem, xfm, yfm, xgm, ygm, xhm, yhm;

068   int xas, yas, xbs, ybs, xcs, ycs, xds, yds;
069   int xes, yes, xfs, yfs, xgs, ygs, xhs, yhs;

070   view_To_screen_coordinates vTs = new view_To_screen_coordinates();
071   project3D_To_2D pr = new project3D_To_2D();
072   rotate_calculator rc = new rotate_calculator();

073   public static void main(String args[]) {
074     Ex17_3 workStart=new Ex17_3();
075   }

076   public Ex17_3() {
077     super("Ex17_3");
078     setSize(400, 400);
```

```
079     enableEvents(AWTEvent.WINDOW_EVENT_MASK);

080     setVisible(true);
081     new Thread(this).start();
082     }

083  public void processWindowEvent(WindowEvent e) {
084     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
085         System.exit(0);
086     }
087  }

088  public void run() {
089     while(true) {
090        rc.rotateY(xa,ya,za, 0.03);
091        xa=rc.getPtX(); ya=rc.getPtY(); za=rc.getPtZ();
092        rc.rotateY(xb,yb,zb, 0.03);
093        xb=rc.getPtX(); yb=rc.getPtY(); zb=rc.getPtZ();
094        rc.rotateY(xc,yc,zc, 0.03);
095        xc=rc.getPtX(); yc=rc.getPtY(); zc=rc.getPtZ();
096        rc.rotateY(xd,yd,zd, 0.03);
097        xd=rc.getPtX(); yd=rc.getPtY(); zd=rc.getPtZ();
098        rc.rotateY(xe,ye,ze, 0.03);
099        xe=rc.getPtX(); ye=rc.getPtY(); ze=rc.getPtZ();
100       rc.rotateY(xf,yf,zf, 0.03);
101       xf=rc.getPtX(); yf=rc.getPtY(); zf=rc.getPtZ();
102       rc.rotateY(xg,yg,zg, 0.03);
103       xg=rc.getPtX(); yg=rc.getPtY(); zg=rc.getPtZ();
104       rc.rotateY(xh,yh,zh, 0.03);
105       xh=rc.getPtX(); yh=rc.getPtY(); zh=rc.getPtZ();

106       xam = (int)pr.projectPtX(xa,ya,za);
107       yam = (int)pr.projectPtY(xa,ya,za);
108       xbm = (int)pr.projectPtX(xb,yb,zb);
109       ybm = (int)pr.projectPtY(xb,yb,zb);
110       xcm = (int)pr.projectPtX(xc,yc,zc);
111       ycm = (int)pr.projectPtY(xc,yc,zc);
112       xdm = (int)pr.projectPtX(xd,yd,zd);
113       ydm = (int)pr.projectPtY(xd,yd,zd);
114       xem = (int)pr.projectPtX(xe,ye,ze);
115       yem = (int)pr.projectPtY(xe,ye,ze);
116       xfm = (int)pr.projectPtX(xf,yf,zf);
117       yfm = (int)pr.projectPtY(xf,yf,zf);
118       xgm = (int)pr.projectPtX(xg,yg,zg);
119       ygm = (int)pr.projectPtY(xg,yg,zg);
120       xhm = (int)pr.projectPtX(xh,yh,zh);
121       yhm = (int)pr.projectPtY(xh,yh,zh);

122       xas = vTs.viewX_To_screenX(xam);
123       yas = vTs.viewY_To_screenY(yam);
124       xbs = vTs.viewX_To_screenX(xbm);
125       ybs = vTs.viewY_To_screenY(ybm);
126       xcs = vTs.viewX_To_screenX(xcm);
127       ycs = vTs.viewY_To_screenY(ycm);
128       xds = vTs.viewX_To_screenX(xdm);
129       yds = vTs.viewY_To_screenY(ydm);
130       xes = vTs.viewX_To_screenX(xem);
131       yes = vTs.viewY_To_screenY(yem);
132       xfs = vTs.viewX_To_screenX(xfm);
133       yfs = vTs.viewY_To_screenY(yfm);
```

```
134         xgs = vTs.viewX_To_screenX(xgm);
135         ygs = vTs.viewY_To_screenY(ygm);
136         xhs = vTs.viewX_To_screenX(xhm);
137         yhs = vTs.viewY_To_screenY(yhm);

138         repaint();
139         try{Thread.sleep(150);}
140         catch(InterruptedException e) {;}
141      }
142   }

143   public void paint(Graphics g) {
144      g.drawLine(xas, yas, xbs, ybs);
145      g.drawLine(xbs, ybs, xcs, ycs);
146      g.drawLine(xcs, ycs, xds, yds);
147      g.drawLine(xds, yds, xas, yas);

148      g.drawLine(xes, yes, xfs, yfs);
149      g.drawLine(xfs, yfs, xgs, ygs);
150      g.drawLine(xgs, ygs, xhs, yhs);
151      g.drawLine(xhs, yhs, xes, yes);

152      g.drawLine(xas, yas, xes, yes);
153      g.drawLine(xbs, ybs, xfs, yfs);
154      g.drawLine(xcs, ycs, xgs, ygs);
155      g.drawLine(xds, yds, xhs, yhs);

156      int Xcdhg[] = {xcs, xds, xhs, xgs};
157      int Ycdhg[] = {ycs, yds, yhs, ygs};
158      g.setColor(Color.red);
159      g.fillPolygon(Xcdhg, Ycdhg, 4);
160   }
161 }
```

行 090～105　以行 27～56 的类创建的新对象 rc，将 3D 图形围绕 y 轴旋转 α 角度。
行 106～121　以行 11～26 的类创建的新对象 pr，将 3D 图形各点的坐标投影到 2D 平面。
行 122～137　以行 01～10 的类创建的新对象 vTs，将视图坐标转换成屏幕坐标。
行 156～159　以红色填充面 cdhg。

编译程序

输入 "javac Ex17_3.java" 进行编译，如图 17-2 所示。

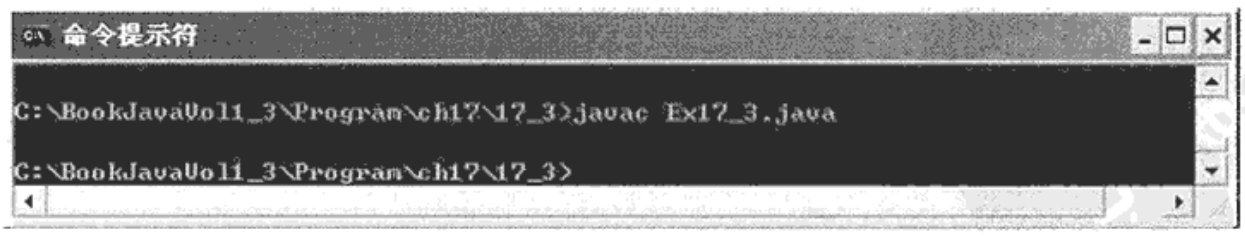


图 17-2

运行程序

输入 "java Ex17_3"，如图 17-3 所示。

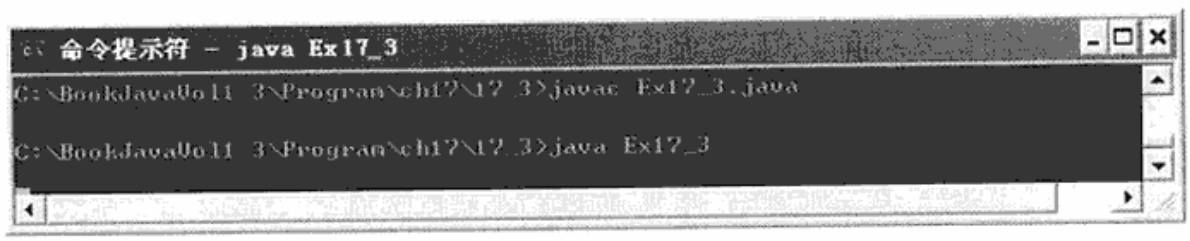


图 17-3

运行结果

绘制 3D 图形围绕 y 轴旋转，如图 17—4 所示。

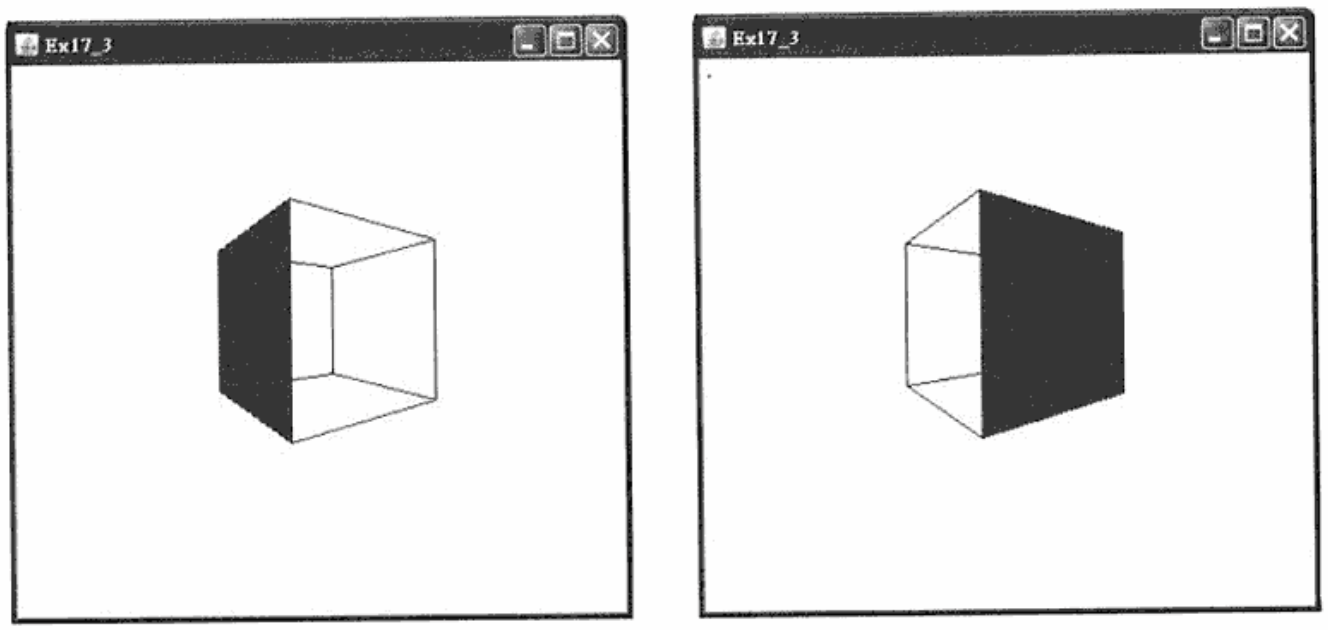


图 17-4

17-4 绕 x 轴旋转

范例 75 设计文件 Ex17_4.java，其功能是解释如何使用 math3D 包与绘制 3D 立体多边形围绕 x 轴旋转。

包文件 view_To_screen_coordinates.java：参考范例 74，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch17\17_4\math3D。

包文件 project3D_To_2D.java：参考范例 74，将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v'（x',y'），本例目录为 C:\BookJavaVol1_3\Program\ch17\17_4\math3D。

包文件 rotate_calculator.java：参考范例 74，将 3D 图形围绕 x 轴（或 y 轴、或 z 轴）旋转 α 角度，本例目录为 C：\BookJavaVol1_3\Program\ch17\17_4\math3D。

文件 Ex17_4.java：绘制 3D 立体多边形围绕 x 轴旋转。

```
057 import math3D.*;

058 import java.awt.*;
059 import java.awt.event.*;
060 import java.awt.Graphics;

061 public class Ex17_4 extends Frame implements Runnable {
062    double xa=-50, ya=50, za=50, xb=-50, yb=50, zb=-50;
063    double xc=50, yc=50, zc=-50, xd=50, yd=50, zd=50;
```

```
064  double xe=-50, ye=-50, ze=50, xf=-50, yf=-50, zf=-50;
065  double xg=50, yg=-50, zg=-50, xh=50, yh=-50, zh=50;

066  int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
067  int xem, yem, xfm, yfm, xgm, ygm, xhm, yhm;

068  int xas, yas, xbs, ybs, xcs, ycs, xds, yds;
069  int xes, yes, xfs, yfs, xgs, ygs, xhs, yhs;

070  view_To_screen_coordinates vTs = new view_To_screen_coordinates();
071  project3D_To_2D pr = new project3D_To_2D();
072  rotate_calculator rc = new rotate_calculator();

073  public static void main(String args[]) {
074     Ex17_4 workStart=new Ex17_4();
075  }

076  public Ex17_4() {
077     super("Ex17_4");
078     setSize(400, 400);

079     enableEvents(AWTEvent.WINDOW_EVENT_MASK);

080     setVisible(true);
081     new Thread(this).start();
082     }

083  public void processWindowEvent(WindowEvent e) {
084     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
085          System.exit(0);
086     }
087  }

088  public void run() {
089     while(true) {
090        rc.rotateX(xa,ya,za, 0.03);
091        xa=rc.getPtX(); ya=rc.getPtY(); za=rc.getPtZ();
092        rc.rotateX(xb,yb,zb, 0.03);
093        xb=rc.getPtX(); yb=rc.getPtY(); zb=rc.getPtZ();
094        rc.rotateX(xc,yc,zc, 0.03);
095        xc=rc.getPtX(); yc=rc.getPtY(); zc=rc.getPtZ();
096        rc.rotateX(xd,yd,zd, 0.03);
097        xd=rc.getPtX(); yd=rc.getPtY(); zd=rc.getPtZ();
098        rc.rotateX(xe,ye,ze, 0.03);
099        xe=rc.getPtX(); ye=rc.getPtY(); ze=rc.getPtZ();
100        rc.rotateX(xf,yf,zf, 0.03);
101        xf=rc.getPtX(); yf=rc.getPtY(); zf=rc.getPtZ();
102        rc.rotateX(xg,yg,zg, 0.03);
103        xg=rc.getPtX(); yg=rc.getPtY(); zg=rc.getPtZ();
104        rc.rotateX(xh,yh,zh, 0.03);
105        xh=rc.getPtX(); yh=rc.getPtY(); zh=rc.getPtZ();

106        xam = (int)pr.projectPtX(xa,ya,za);
107        yam = (int)pr.projectPtY(xa,ya,za);
108        xbm = (int)pr.projectPtX(xb,yb,zb);
109        ybm = (int)pr.projectPtY(xb,yb,zb);
110        xcm = (int)pr.projectPtX(xc,yc,zc);
111        ycm = (int)pr.projectPtY(xc,yc,zc);
112        xdm = (int)pr.projectPtX(xd,yd,zd);
113        ydm = (int)pr.projectPtY(xd,yd,zd);
```

```
114         xem = (int)pr.projectPtX(xe,ye,ze);
115         yem = (int)pr.projectPtY(xe,ye,ze);
116         xfm = (int)pr.projectPtX(xf,yf,zf);
117         yfm = (int)pr.projectPtY(xf,yf,zf);
118         xgm = (int)pr.projectPtX(xg,yg,zg);
119         ygm = (int)pr.projectPtY(xg,yg,zg);
120         xhm = (int)pr.projectPtX(xh,yh,zh);
121         yhm = (int)pr.projectPtY(xh,yh,zh);

122         xas = vTs.viewX_To_screenX(xam);
123         yas = vTs.viewY_To_screenY(yam);
124         xbs = vTs.viewX_To_screenX(xbm);
125         ybs = vTs.viewY_To_screenY(ybm);
126         xcs = vTs.viewX_To_screenX(xcm);
127         ycs = vTs.viewY_To_screenY(ycm);
128         xds = vTs.viewX_To_screenX(xdm);
129         yds = vTs.viewY_To_screenY(ydm);
130         xes = vTs.viewX_To_screenX(xem);
131         yes = vTs.viewY_To_screenY(yem);
132         xfs = vTs.viewX_To_screenX(xfm);
133         yfs = vTs.viewY_To_screenY(yfm);
134         xgs = vTs.viewX_To_screenX(xgm);
135         ygs = vTs.viewY_To_screenY(ygm);
136         xhs = vTs.viewX_To_screenX(xhm);
137         yhs = vTs.viewY_To_screenY(yhm);

138         repaint();
139         try{Thread.sleep(150);}
140         catch(InterruptedException e) {;}
141       }
142   }
143   public void paint(Graphics g) {
144       g.drawLine(xas, yas, xbs, ybs);
145       g.drawLine(xbs, ybs, xcs, ycs);
146       g.drawLine(xcs, ycs, xds, yds);
147       g.drawLine(xds, yds, xas, yas);

148       g.drawLine(xes, yes, xfs, yfs);
149       g.drawLine(xfs, yfs, xgs, ygs);
150       g.drawLine(xgs, ygs, xhs, yhs);
151       g.drawLine(xhs, yhs, xes, yes);

152       g.drawLine(xas, yas, xes, yes);
153       g.drawLine(xbs, ybs, xfs, yfs);
154       g.drawLine(xcs, ycs, xgs, ygs);
155       g.drawLine(xds, yds, xhs, yhs);

156       int Xaehd[] = {xas, xes, xhs, xds};
157       int Yaehd[] = {yas, yes, yhs, yds};
158       g.setColor(Color.red);
159       g.fillPolygon(Xaehd, Yaehd, 4);
160   }
161 }
```

行 090～105　以行 27～56 的类创建的新对象 rc，将 3D 图形围绕 x 轴旋转 α 角度。
行 156～159　以红色填充面 aehd。

编译程序

输入“javac Ex17_4.java”进行编译，如图 17–5 所示。

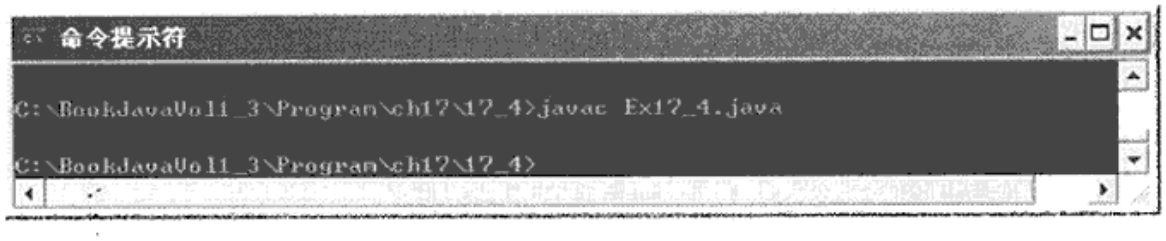


图 17-5

运行程序

输入“java Ex17_4”，如图 17–6 所示。

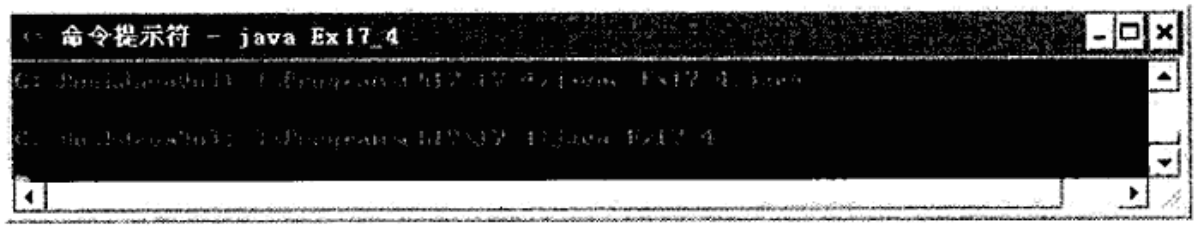


图 17-6

运行结果

绘制 3D 立体多边形围绕 x 轴旋转，如图 17–7 所示。

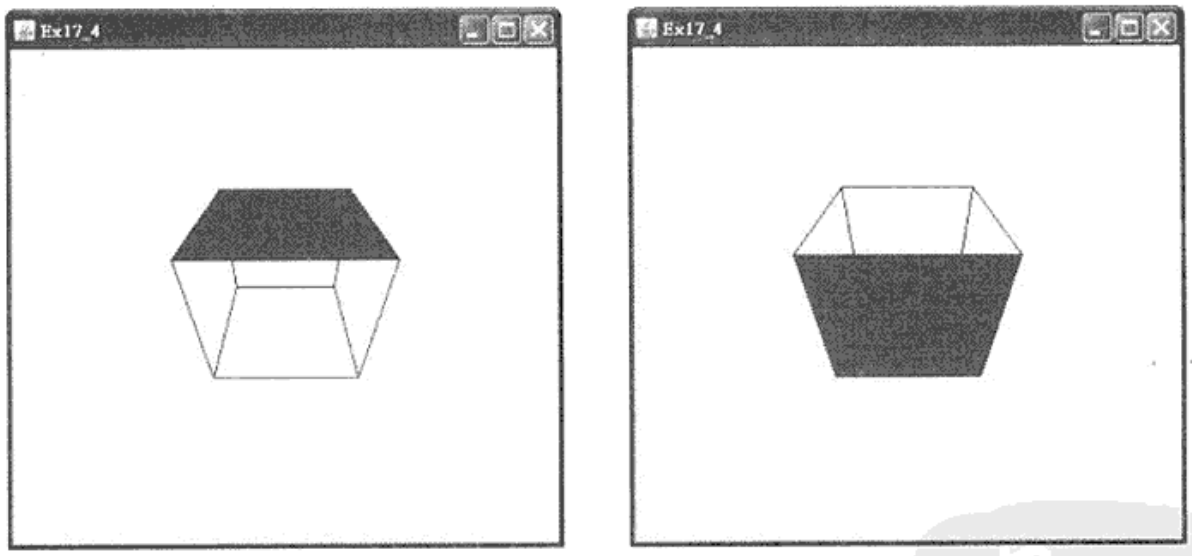


图 17-7

17-5 习题

1. 3D 图形围绕 x 轴旋转的表达式为何？
2. 3D 图形围绕 y 轴旋转的表达式为何？
3. 3D 图形围绕 z 轴旋转的表达式为何？
4. 试设计一个立方体围绕 z 轴旋转的程序。

Chapter 18　法线与隐藏线

18-1　简介

实例对象并非都是透明的。若不是透明的，则我们只能看到其正面，无法看到被遮挡的背面，在绘图上称它为隐藏部分，因此在绘制时必须将隐藏部分做消除或浅细处理。

在设计时，必须先判别哪一些是正面显示部分，哪一些是背面隐藏部分。本章将通过范例进行详尽解说。

18-2　隐藏线

立体对象的显示与平面对象的显示有所不同，平面对象是尽收眼底，而立体对象因其层次性，无法一览而尽。也就是说，我们只能看到前端的部分，无法看到后端被遮挡的部分，如图 18–1 所示。如果我们沿 z 轴凝视，我们只能看到前端平面 daeh，而无法看到被遮挡的后端平面 bcgf（读者是否注意到：笔者在描述平面各点时，都是面向平面逆时针描述，这是为了配合法线的方向应用，将在 18–2–1 节说明）。

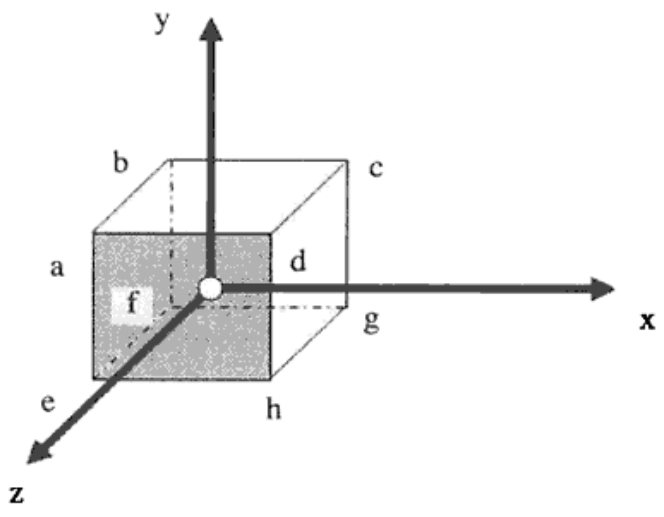


图 18-1

18-2-1 立体图像的法线

与多边形平面相垂直的线段的向量，是该平面的法线向量，立体对象的表面由多个多边形组成，每一个多边形都有一个法线向量。

1 如图 18–2 所示，线段 n 与多边形 abcd 垂直，将线段 n 平行移动到线段 N，则线段 N 是多边形 abcd 的法线向量（注意多边形 abcd 的各点必须逆时针依序选择）。

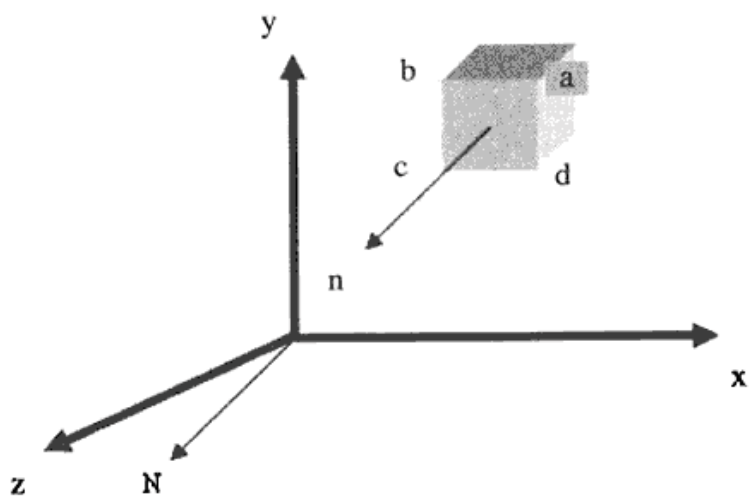


图 18-2

2 在 14–4 节中，我们探讨过“内积与法线向量”，设有 A、B 两线段向量，Ø 为夹角，当 A * B = 0 时，cosØ 必为 0，∠Ø 为 90°。在本节中，我们将讨论“多边形与法线”，如图 18–2 所示。

设线段 ab 的向量为 U（即 U_x=xa–xb，U_y=ya–yb，U_z=za–zb）；线段 bc 的向量为 V（即 V_x=xb–xc，V_y=yb–yc，V_z=zb–zc）。

若 N 同时与 U、V 垂直，则 N 必与多边形 abcd 垂直。

即 U*N = 0 且 V*N = 0，则多边形 abcd 的法线向量必为 N。

也即 $U_x*N_x + U_y*N_y + U_z*N_z = 0$ 且 $V_x*N_x + V_y*N_y + V_z*N_z = 0$。

3 以上方程组有解，其中有一个解可由线性代数的行列式求得。其法线向量 N 如下：

$$N_x = U_y*V_z - U_z*V_y$$
$$N_y = U_z*V_x - U_x*V_z$$
$$N_z = U_x*V_y - U_y*V_x$$

4 多边形 abcd 与法线向量 N 有以下关系。

If ((U*N < 0) and (V*N < 0)) then (Ø > 90°)
If ((U*N = 0) and (V*N = 0)) then (Ø = 90°)
If ((U*N > 0) and (V*N > 0)) then (Ø < 90°)

如果我们沿着向量 N 观看多边形 abcd，只有当 Ø<= 90°时，我们才能看到多边形 abcd；当 Ø > 90°时，我们将看不到多边形 abcd，此时应将其隐藏起来。

5 3D 法线的应用。

除非有特殊的设计要求，观看者的视线一般均与屏幕垂直，即沿着 z 轴观看屏幕，故屏幕的法线可定为（0，0，N_z），继而屏幕上 3D 图案的法线亦可设为（0，0，N_z），由本节以上知识可求得屏幕上 3D 图案的法线 N 为（0，0，($U_x*V_y - U_y*V_x$)）。

因此由本节以上知识可知，当 N_z >= 0 时，Ø <= 90°，此时我们可看到 3D 图案，否则我们看

不到 3D 图案，应将其隐藏起来。故 $N_z >= 0$ 可视为“隐藏/显示”的条件，即遮挡判别条件。

18-2-2 隐藏线处理

根据 18–2–1 节的计算公式设计范例 76，创建文件 Ex18_2.java 绘制 3D 立体多边形围绕 y 轴旋转，并对隐藏线进行浅细处理。在 math3D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标；创建文件 project3D_To_2D.java，将 3D 图形各点的坐标投影到 2D 平面上；创建文件 rotate_calculator.java，将 3D 图形围绕 x 轴（或 y 轴、或 z 轴）旋转 α 角度；创建文件 normal_hidden.java，计算隐藏条件 Nz。

范例 76 设计文件 Ex18_2.java，其功能是解释如何使用 math3D 包与绘制 3D 立方体围绕 y 轴旋转，并对隐藏线进行浅细处理。

包文件 view_To_screen_coordinates.java：参考范例 74，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_2\math3D。

包文件 project3D_To_2D.java：参考范例 74，将 3D 对象的各点 v(x,y,z)投影到 2D 平面的各点 v'(x',y')，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_2\math3D。

包文件 rotate_calculator.java：参考范例 74，将 3D 图形围绕 x 轴（或 y 轴、或 z 轴）旋转 α 角度，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_2\math3D。

包文件 normal_hidden.java：计算 3D 图形的隐藏条件 Nz，当 Nz >= 0 时，Ø <= 90°，此时我们可看到 3D 图案，否则我们看不到 3D 图案，应进行隐藏处理，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_2\math3D。

```
57 package math3D;
58 import java.awt.*;

59 public class normal_hidden {
60   private double Nx, Ny, Nz;

61   public void vectNormal(double xa, double ya, double za,
                            double xb, double yb, double zb,
                            double xc, double yc, double zc) {

62     double Ux=xa-xb; double Uy=ya-yb; double Uz=za-zb;
63     double Vx=xb-xc; double Vy=yb-yc; double Vz=zb-zc;

64     this.Nx=Uy*Vz - Uz*Vy;
65     this.Ny=Uz*Vx - Ux*Vz;
66     this.Nz=Ux*Vy - Uy*Vx;
67   }

68   public double getHidden() {return Nz;}
69 }
```

行 57	创建 math3D 包。
行 60	声明法线向量的坐标变量。
行 61～67	以两线段坐标为参数，根据 18–2–1 节的计算公式求这两线段平面的法线向量。
行 68	返回隐藏条件 Nz。

文件 Ex18_2.java：绘制 3D 立方体围绕 y 轴旋转，并将隐藏线做浅细处理。

```
070 import math3D.*;

071 import java.awt.*;
072 import java.awt.event.*;
073 import java.awt.Graphics;

074 public class Ex18_2 extends Frame implements Runnable {
075    double xa=-50, ya=50, za=50, xb=-50, yb=50, zb=-50;
076    double xc=50, yc=50, zc=-50, xd=50, yd=50, zd=50;
077    double xe=-50, ye=-50, ze=50, xf=-50, yf=-50, zf=-50;
078    double xg=50, yg=-50, zg=-50, xh=50, yh=-50, zh=50;

079    int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
080    int xem, yem, xfm, yfm, xgm, ygm, xhm, yhm;

081    int xas, yas, xbs, ybs, xcs, ycs, xds, yds;
082    int xes, yes, xfs, yfs, xgs, ygs, xhs, yhs;

083    view_To_screen_coordinates vTs = new view_To_screen_coordinates();
084    project3D_To_2D pr = new project3D_To_2D();
085    rotate_calculator rc = new rotate_calculator();

086    normal_hidden P1 = new normal_hidden();
087    normal_hidden P2 = new normal_hidden();
088    normal_hidden P3 = new normal_hidden();
089    normal_hidden P4 = new normal_hidden();
090    normal_hidden P5 = new normal_hidden();
091    normal_hidden P6 = new normal_hidden();

092    public static void main(String args[]) {
093       Ex18_2 workStart=new Ex18_2();
094    }

095    public Ex18_2() {
096       super("Ex18_2");
097       setSize(400, 400);

098       enableEvents(AWTEvent.WINDOW_EVENT_MASK);

099       setVisible(true);
100       new Thread(this).start();
101       }

102    public void processWindowEvent(WindowEvent e) {
103       if(e.getID() == WindowEvent.WINDOW_CLOSING) {
104          System.exit(0);
105       }
106    }

107    public void run() {
108       while(true) {
109          rc.rotateY(xa,ya,za, 0.03);
110          xa=rc.getPtX(); ya=rc.getPtY(); za=rc.getPtZ();
111          rc.rotateY(xb,yb,zb, 0.03);
112          xb=rc.getPtX(); yb=rc.getPtY(); zb=rc.getPtZ();
113          rc.rotateY(xc,yc,zc, 0.03);
114          xc=rc.getPtX(); yc=rc.getPtY(); zc=rc.getPtZ();
115          rc.rotateY(xd,yd,zd, 0.03);
116          xd=rc.getPtX(); yd=rc.getPtY(); zd=rc.getPtZ();
117          rc.rotateY(xe,ye,ze, 0.03);
118          xe=rc.getPtX(); ye=rc.getPtY(); ze=rc.getPtZ();
```

```
        rc.rotateY(xf,yf,zf, 0.03);
        xf=rc.getPtX(); yf=rc.getPtY(); zf=rc.getPtZ();
        rc.rotateY(xg,yg,zg, 0.03);
        xg=rc.getPtX(); yg=rc.getPtY(); zg=rc.getPtZ();
        rc.rotateY(xh,yh,zh, 0.03);
        xh=rc.getPtX(); yh=rc.getPtY(); zh=rc.getPtZ();

        P1.vectNormal(xd, yd, zd, xa, ya, za, xe, ye, ze);
        P2.vectNormal(xc, yc, zc, xd, yd, zd, xh, yh, zh);
        P3.vectNormal(xb, yb, zb, xc, yc, zc, xg, yg, zg);
        P4.vectNormal(xa, ya, za, xb, yb, zb, xf, yf, zf);
        P5.vectNormal(xc, yc, zc, xb, yb, zb, xa, ya, za);
        P6.vectNormal(xh, yh, zh, xe, ye, ze, xf, yf, zf);

        xam = (int)pr.projectPtX(xa,ya,za);
        yam = (int)pr.projectPtY(xa,ya,za);
        xbm = (int)pr.projectPtX(xb,yb,zb);
        ybm = (int)pr.projectPtY(xb,yb,zb);
        xcm = (int)pr.projectPtX(xc,yc,zc);
        ycm = (int)pr.projectPtY(xc,yc,zc);
        xdm = (int)pr.projectPtX(xd,yd,zd);
        ydm = (int)pr.projectPtY(xd,yd,zd);
        xem = (int)pr.projectPtX(xe,ye,ze);
        yem = (int)pr.projectPtY(xe,ye,ze);
        xfm = (int)pr.projectPtX(xf,yf,zf);
        yfm = (int)pr.projectPtY(xf,yf,zf);
        xgm = (int)pr.projectPtX(xg,yg,zg);
        ygm = (int)pr.projectPtY(xg,yg,zg);
        xhm = (int)pr.projectPtX(xh,yh,zh);
        yhm = (int)pr.projectPtY(xh,yh,zh);

        xas = vTs.viewX_To_screenX(xam);
        yas = vTs.viewY_To_screenY(yam);
        xbs = vTs.viewX_To_screenX(xbm);
        ybs = vTs.viewY_To_screenY(ybm);
        xcs = vTs.viewX_To_screenX(xcm);
        ycs = vTs.viewY_To_screenY(ycm);
        xds = vTs.viewX_To_screenX(xdm);
        yds = vTs.viewY_To_screenY(ydm);
        xes = vTs.viewX_To_screenX(xem);
        yes = vTs.viewY_To_screenY(yem);
        xfs = vTs.viewX_To_screenX(xfm);
        yfs = vTs.viewY_To_screenY(yfm);
        xgs = vTs.viewX_To_screenX(xgm);
        ygs = vTs.viewY_To_screenY(ygm);
        xhs = vTs.viewX_To_screenX(xhm);
        yhs = vTs.viewY_To_screenY(yhm);

        repaint();
        try{Thread.sleep(150);}
        catch(InterruptedException e) {;}
    }
}

public void paint(Graphics g) {
    g.setColor(new Color(225,225,225));
    g.drawLine(xas, yas, xbs, ybs);
    g.drawLine(xbs, ybs, xcs, ycs);
    g.drawLine(xcs, ycs, xds, yds);
    g.drawLine(xds, yds, xas, yas);

    g.drawLine(xes, yes, xfs, yfs);
    g.drawLine(xfs, yfs, xgs, ygs);
```

```
176     g.drawLine(xgs, ygs, xhs, yhs);
177     g.drawLine(xhs, yhs, xes, yes);

178     g.drawLine(xas, yas, xes, yes);
179     g.drawLine(xbs, ybs, xfs, yfs);
180     g.drawLine(xcs, ycs, xgs, ygs);
181     g.drawLine(xds, yds, xhs, yhs);

182     g.setColor(new Color(0,0,0));
183     if (P1.getHidden() > 0.0) {
184        g.drawLine(xds, yds, xas, yas);
185        g.drawLine(xas, yas, xes, yes);
186        g.drawLine(xes, yes, xhs, yhs);
187        g.drawLine(xhs, yhs, xds, yds);
188     }

189     if (P2.getHidden() > 0.0) {
190        g.drawLine(xcs, ycs, xds, yds);
191        g.drawLine(xds, yds, xhs, yhs);
192        g.drawLine(xhs, yhs, xgs, ygs);
193        g.drawLine(xgs, ygs, xcs, ycs);
194     }

195     if (P3.getHidden() > 0.0) {
196        g.drawLine(xbs, ybs, xcs, ycs);
197        g.drawLine(xcs, ycs, xgs, ygs);
198        g.drawLine(xgs, ygs, xfs, yfs);
199        g.drawLine(xfs, yfs, xbs, ybs);
200     }

201     if (P4.getHidden() > 0.0) {
202        g.drawLine(xas, yas, xbs, ybs);
203        g.drawLine(xbs, ybs, xfs, yfs);
204        g.drawLine(xfs, yfs, xes, yes);
205        g.drawLine(xes, yes, xas, yas);
206     }

207     if (P5.getHidden() > 0.0) {
208        g.drawLine(xcs, ycs, xbs, ybs);
209        g.drawLine(xbs, ybs, xas, yas);
210        g.drawLine(xas, yas, xds, yds);
211        g.drawLine(xds, yds, xcs, ycs);
212     }

213     if (P6.getHidden() > 0.0) {
214        g.drawLine(xhs, yhs, xes, yes);
215        g.drawLine(xes, yes, xfs, yfs);
216        g.drawLine(xfs, yfs, xgs, ygs);
217        g.drawLine(xgs, ygs, xhs, yhs);
218     }
219  }
220 }
```

行 086～091	以行 57～69 的类创建新对象 P1、P2、P3、P4、P5 和 P6，它们是立方体的六个平面。
行 125～130	设置六个平面的法线。
行 169	设置浅细线的绘制。
行 170～181	以浅细线绘制立方体的各线段。
行 183～188	如果平面 P1 返回的隐藏条件大于 0，则以深色线绘制该平面的各线段。

编译程序

输入“javac Ex18_2.java”进行编译，如图 18–3 所示。

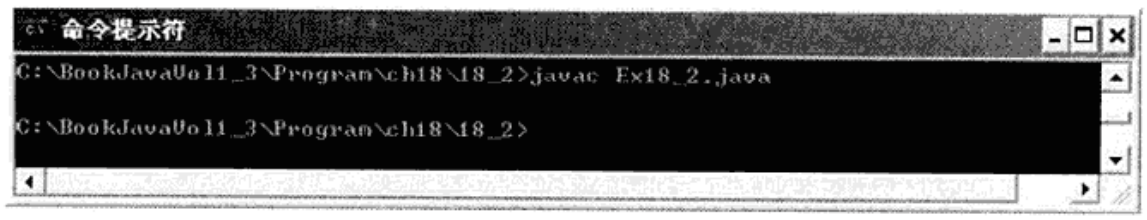


图 18-3

运行程序

输入“java Ex18_2”，如图 18–4 所示。

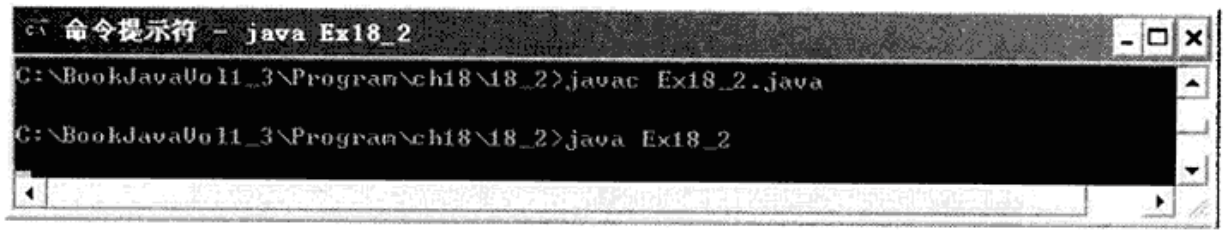


图 18-4

运行结果

绘制 3D 立方体围绕 y 轴旋转，并将隐藏线做浅细处理，如图 18–5 所示。

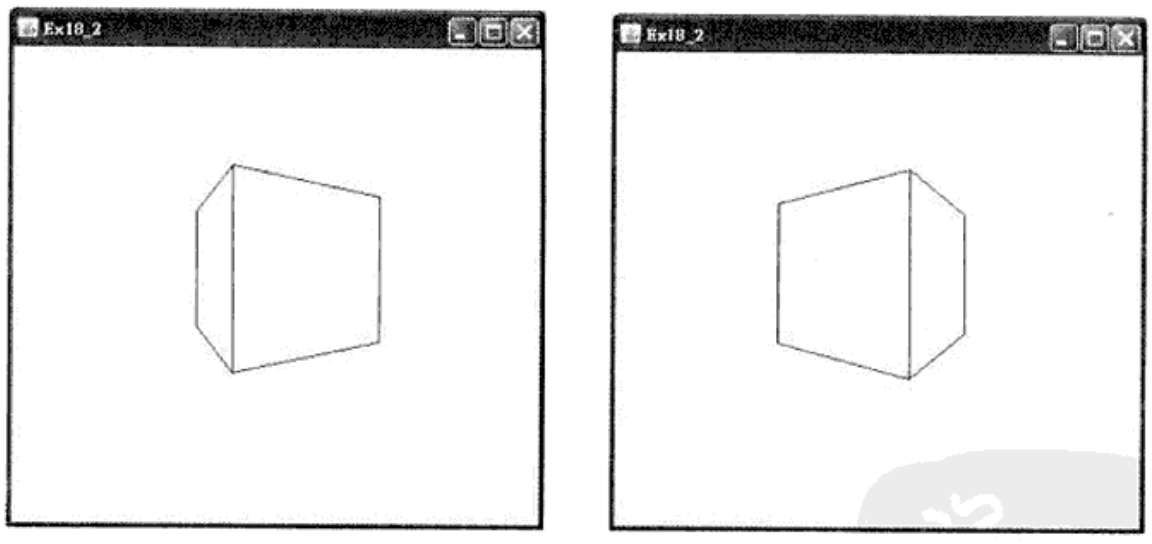


图 18-5

讨论事项

立方体在旋转时，仍有小部分隐藏线无法隐藏（如图 18–6 所示），这是视角误差造成的，将在 18–4 节中讨论消除方法。

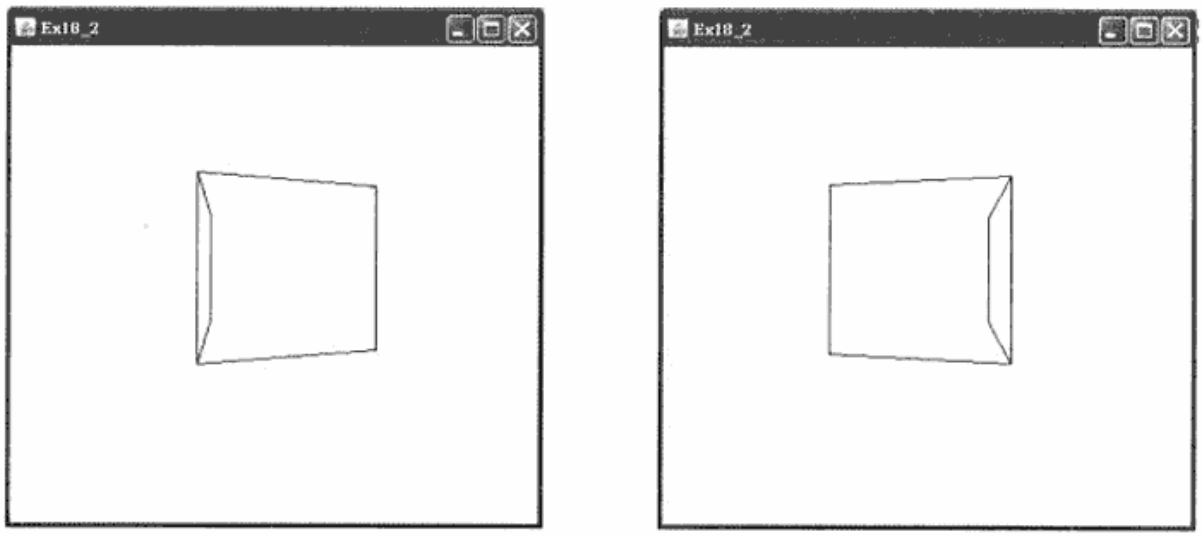


图 18-6

18-3 视角误差

如图 18–7 所示，当我们沿 z 轴垂直观看 3D 图案时，是以 N1 为法线。但事实上，我们是以 N2 为法线，故在视觉上必有一视角误差 α，它随 Ø 值与视距的变动而变动，因此在消除隐藏线的设计上需做些许修正。

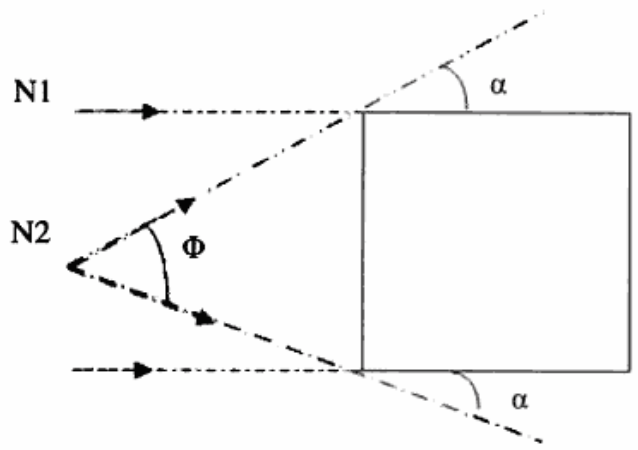


图 18-7

18-4 修正视角误差

在 18–2–2 节范例 76 最后两个运行结果中，当某平面已旋转到立方体的背面且已达隐藏线部位时，按理我们不应看到该平面有深色线段。其实则不然，其原因是 18–3 节所讨论的视角误差。因为并非垂直视线，平面部分看似已旋转至背面，事实上，其法线向量的 Nz 仍大于 0，即若以垂直视线观看，部分尚未旋转至背面。

视角误差的修正，应考虑其视角与视距的变化（如图 18–7 所示）。为了便于介绍，笔者采用“观测法”，观察并测试 Nz 值，当其小于 0 时，一定看不到；当其大于 0 时，可看到，但有视角误差。我们可在程序运行时观察其临界值，以范例 77 为例，Nz > 2400.0 为临界值。因此以 Nz > 2400.0 替换 Nz >= 0。

根据 18–3 节设计范例 77，创建文件 Ex18_4.java 修正视角误差（以 Nz>2400.0 替换 Nz >= 0）。在 math3D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标；创建文件

project3D_To_2D.java，将 3D 图形各点的坐标投影到 2D 平面上；创建文件 rotate_calculator.java，将 3D 图形围绕 x 轴（或 y 轴、或 z 轴）旋转 α 角度；创建文件 normal_hidden.java，计算隐藏条件 Nz。

范例 77 设计文件 Ex18_4.java，其功能是解释如何使用 math3D 包与绘制 3D 立方体围绕 y 轴旋转，将隐藏线做浅细处理并修正视角误差。

包文件 view_To_screen_coordinates.java：参考范例 74，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_4\math3D。

包文件 project3D_To_2D.java：参考范例 74，将 3D 对象的各点 v (x,y,z)投影到 2D 平面的各点 v' (x',y')，本例目录为 C:\BookJavaVol1_3\Program\ch18\18_4\math3D。

包文件 rotate_calculator.java：参考范例 74，将 3D 图形围绕 x 轴(或 y 轴、或 z 轴)旋转 α 角度，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_4\math3D。

包文件 normal_hidden.java：参考范例 76，计算 3D 图形的隐藏条件 Nz，当 Nz >= 2400.0 时，Ø<=90°，此时我们可看到 3D 图案，否则我们看不到 3D 图案，应做隐藏处理，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_4\math3D。

文件 Ex18_4.java：绘制 3D 立方体围绕 y 轴旋转，将隐藏线做浅细处理并修正视角误差。

```
070 import math3D.*;

071 import java.awt.*;
072 import java.awt.event.*;
073 import java.awt.Graphics;

074 public class Ex18_4 extends Frame implements Runnable {
075    double xa=-50, ya=50, za=50, xb=-50, yb=50, zb=-50;
076    double xc=50, yc=50, zc=-50, xd=50, yd=50, zd=50;
077    double xe=-50, ye=-50, ze=50, xf=-50, yf=-50, zf=-50;
078    double xg=50, yg=-50, zg=-50, xh=50, yh=-50, zh=50;

079    int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
080    int xem, yem, xfm, yfm, xgm, ygm, xhm, yhm;

081    int xas, yas, xbs, ybs, xcs, ycs, xds, yds;
082    int xes, yes, xfs, yfs, xgs, ygs, xhs, yhs;

083    view_To_screen_coordinates vTs = new view_To_screen_coordinates();
084    project3D_To_2D pr = new project3D_To_2D();
085    rotate_calculator rc = new rotate_calculator();

086    normal_hidden P1 = new normal_hidden();
087    normal_hidden P2 = new normal_hidden();
088    normal_hidden P3 = new normal_hidden();
089    normal_hidden P4 = new normal_hidden();
090    normal_hidden P5 = new normal_hidden();
091    normal_hidden P6 = new normal_hidden();

092    public static void main(String args[]) {
093       Ex18_4 workStart=new Ex18_4();
094    }

095    public Ex18_4() {
096       super("Ex18_4");
097       setSize(400, 400);
```

```
098     enableEvents(AWTEvent.WINDOW_EVENT_MASK);

099     setVisible(true);
100     new Thread(this).start();
101     }

102  public void processWindowEvent(WindowEvent e) {
103     if(e.getID() == WindowEvent.WINDOW_CLOSING) {
104         System.exit(0);
105     }
106  }

107  public void run() {
108     while(true) {
109        rc.rotateY(xa,ya,za, 0.03);
110        xa=rc.getPtX(); ya=rc.getPtY(); za=rc.getPtZ();
111        rc.rotateY(xb,yb,zb, 0.03);
112        xb=rc.getPtX(); yb=rc.getPtY(); zb=rc.getPtZ();
113        rc.rotateY(xc,yc,zc, 0.03);
114        xc=rc.getPtX(); yc=rc.getPtY(); zc=rc.getPtZ();
115        rc.rotateY(xd,yd,zd, 0.03);
116        xd=rc.getPtX(); yd=rc.getPtY(); zd=rc.getPtZ();
117        rc.rotateY(xe,ye,ze, 0.03);
118        xe=rc.getPtX(); ye=rc.getPtY(); ze=rc.getPtZ();
119        rc.rotateY(xf,yf,zf, 0.03);
120        xf=rc.getPtX(); yf=rc.getPtY(); zf=rc.getPtZ();
121        rc.rotateY(xg,yg,zg, 0.03);
122        xg=rc.getPtX(); yg=rc.getPtY(); zg=rc.getPtZ();
123        rc.rotateY(xh,yh,zh, 0.03);
124        xh=rc.getPtX(); yh=rc.getPtY(); zh=rc.getPtZ();

125        P1.vectNormal(xd, yd, zd, xa, ya, za, xe, ye, ze);
126        P2.vectNormal(xc, yc, zc, xd, yd, zd, xh, yh, zh);
127        P3.vectNormal(xb, yb, zb, xc, yc, zc, xg, yg, zg);
128        P4.vectNormal(xa, ya, za, xb, yb, zb, xf, yf, zf);
129        P5.vectNormal(xc, yc, zc, xb, yb, zb, xa, ya, za);
130        P6.vectNormal(xh, yh, zh, xe, ye, ze, xf, yf, zf);

131        xam = (int)pr.projectPtX(xa,ya,za);
132        yam = (int)pr.projectPtY(xa,ya,za);
133        xbm = (int)pr.projectPtX(xb,yb,zb);
134        ybm = (int)pr.projectPtY(xb,yb,zb);
135        xcm = (int)pr.projectPtX(xc,yc,zc);
136        ycm = (int)pr.projectPtY(xc,yc,zc);
137        xdm = (int)pr.projectPtX(xd,yd,zd);
138        ydm = (int)pr.projectPtY(xd,yd,zd);
139        xem = (int)pr.projectPtX(xe,ye,ze);
140        yem = (int)pr.projectPtY(xe,ye,ze);
141        xfm = (int)pr.projectPtX(xf,yf,zf);
142        yfm = (int)pr.projectPtY(xf,yf,zf);
143        xgm = (int)pr.projectPtX(xg,yg,zg);
144        ygm = (int)pr.projectPtY(xg,yg,zg);
145        xhm = (int)pr.projectPtX(xh,yh,zh);
146        yhm = (int)pr.projectPtY(xh,yh,zh);

147        xas = vTs.viewX_To_screenX(xam);
148        yas = vTs.viewY_To_screenY(yam);
149        xbs = vTs.viewX_To_screenX(xbm);
150        ybs = vTs.viewY_To_screenY(ybm);
151        xcs = vTs.viewX_To_screenX(xcm);
```

```
152         ycs = vTs.viewY_To_screenY(ycm);
153         xds = vTs.viewX_To_screenX(xdm);
154         yds = vTs.viewY_To_screenY(ydm);
155         xes = vTs.viewX_To_screenX(xem);
156         yes = vTs.viewY_To_screenY(yem);
157         xfs = vTs.viewX_To_screenX(xfm);
158         yfs = vTs.viewY_To_screenY(yfm);
159         xgs = vTs.viewX_To_screenX(xgm);
160         ygs = vTs.viewY_To_screenY(ygm);
161         xhs = vTs.viewX_To_screenX(xhm);
162         yhs = vTs.viewY_To_screenY(yhm);

163         repaint();
164         try{Thread.sleep(150);}
165         catch(InterruptedException e) {;}
166      }
167   }

168   public void paint(Graphics g) {
169      g.setColor(new Color(225,225,225));
170      g.drawLine(xas, yas, xbs, ybs);
171      g.drawLine(xbs, ybs, xcs, ycs);
172      g.drawLine(xcs, ycs, xds, yds);
173      g.drawLine(xds, yds, xas, yas);

174      g.drawLine(xes, yes, xfs, yfs);
175      g.drawLine(xfs, yfs, xgs, ygs);
176      g.drawLine(xgs, ygs, xhs, yhs);
177      g.drawLine(xhs, yhs, xes, yes);

178      g.drawLine(xas, yas, xes, yes);
179      g.drawLine(xbs, ybs, xfs, yfs);
180      g.drawLine(xcs, ycs, xgs, ygs);
181      g.drawLine(xds, yds, xhs, yhs);

182      g.setColor(new Color(0,0,0));
183      if (P1.getHidden() > 2400.0) {
184         g.drawLine(xds, yds, xas, yas);
185         g.drawLine(xas, yas, xes, yes);
186         g.drawLine(xes, yes, xhs, yhs);
187         g.drawLine(xhs, yhs, xds, yds);
188      }

189      if (P2.getHidden() > 2400.0) {
190         g.drawLine(xcs, ycs, xds, yds);
191         g.drawLine(xds, yds, xhs, yhs);
192         g.drawLine(xhs, yhs, xgs, ygs);
193         g.drawLine(xgs, ygs, xcs, ycs);
194      }

195      if (P3.getHidden() >2400.0) {
196         g.drawLine(xbs, ybs, xcs, ycs);
197         g.drawLine(xcs, ycs, xgs, ygs);
198         g.drawLine(xgs, ygs, xfs, yfs);
199         g.drawLine(xfs, yfs, xbs, ybs);
200      }

201      if (P4.getHidden() >2400.0) {
202         g.drawLine(xas, yas, xbs, ybs);
203         g.drawLine(xbs, ybs, xfs, yfs);
```

```
204         g.drawLine(xfs, yfs, xes, yes);
205         g.drawLine(xes, yes, xas, yas);
206       }

207       if (P5.getHidden() > 2400.0) {
208         g.drawLine(xcs, ycs, xbs, ybs);
209         g.drawLine(xbs, ybs, xas, yas);
210         g.drawLine(xas, yas, xds, yds);
211         g.drawLine(xds, yds, xcs, ycs);
212       }

213       if (P6.getHidden() >2400.0) {
214         g.drawLine(xhs, yhs, xes, yes);
215         g.drawLine(xes, yes, xfs, yfs);
216         g.drawLine(xfs, yfs, xgs, ygs);
217         g.drawLine(xgs, ygs, xhs, yhs);
218       }
219     }
220 }
```

行 183　采用“观测法”，以 Nz > 2400.0 替换 Nz >= 0。

编译程序

输入“javac Ex18_4.java”进行编译，如图 18-8 所示。

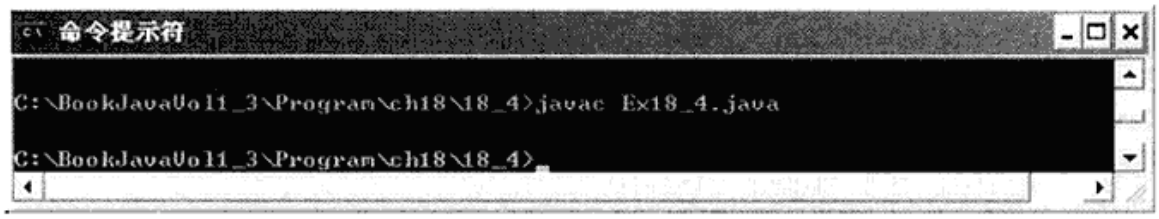


图 18-8

运行程序

输入“java Ex18_4”，如图 18-9 所示。

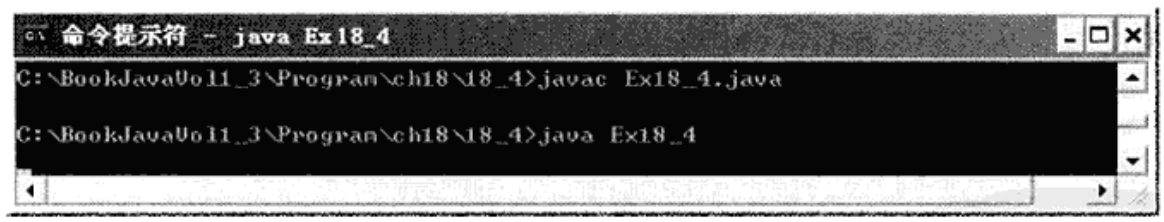


图 18-9

运行结果

绘制 3D 立方体围绕 y 轴旋转，并将隐藏线做浅细处理，如图 18-10 所示。

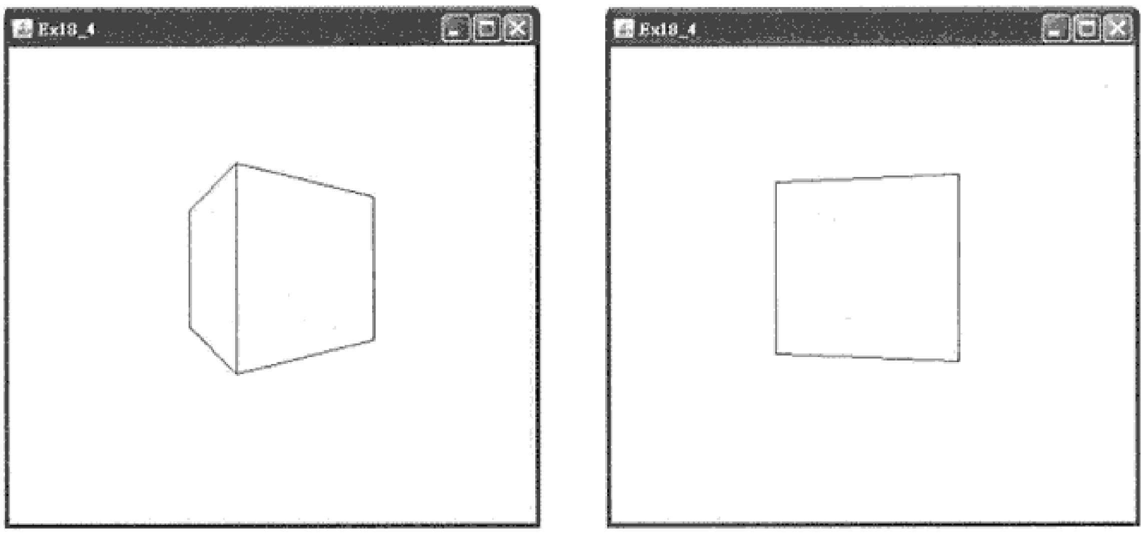


图 18-10

讨论事项

采用“观测法”，以 Nz > 2400.0 替换 Nz >= 0，修正视角误差。

18-5 光影变化

在 18–4 节中，我们以 Nz 值作为立方体的遮挡判别条件，本节仍以其作为光影的亮度条件。在不同的环境中有不同的 Nz 值，假设光线是从正面入射，则 Nz 值越大，该平面应越明亮。

由 Color 类可知，RGB 颜色值介于 0～255，值越大越明亮。由范例 77 的观测结果可知，其可见平面的 Nz 值介于 2400～10000。如果设置 n 值且 2400/n ～ 10000/n 介于 0～255，则 n 值为可行系数，范例 78 将设置 n = 52。

设计范例 78，创建文件 Ex18_5.java 设置立方体在旋转时的光影变化。在 math3D 包中创建文件 view_To_screen_coordinates.java，将视图坐标转换成屏幕坐标；创建文件 project3D_To_2D.java，将 3D 图形各点的坐标投影到 2D 平面上；创建文件 rotate_calculator.java，将 3D 图形围绕 x 轴（或 y 轴、或 z 轴）旋转 α 角度；创建文件 normal_hidden.java，计算隐藏条件 Nz。

范例 78 设计文件 Ex18_5.java，其功能是解释如何使用 math3D 包与绘制 3D 立方体围绕 y 轴旋转，设置立方体在旋转时的光影变化。

包文件 view_To_screen_coordinates.java：参考范例 74，将视图坐标转换成屏幕坐标，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_5\math3D。

包文件 project3D_To_2D.java：参考范例 74，将 3D 对象的各点 v（x,y,z)投影到 2D 平面的各点 v'（x',y'），本例目录为 C：\BookJavaVol1_3\Program\ch18\18_5\math3D。

包文件 rotate_calculator.java：参考范例 74，将 3D 图形围绕 x 轴（或 y 轴、或 z 轴）旋转 α 角度，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_5\math3D。

包文件 normal_hidden.java：参考范例 76，计算 3D 图形的隐藏条件 Nz，当 Nz >= 2400.0 时，Ø<=90°，此时我们可看到 3D 图案，否则我们看不到 3D 图案，应做隐藏处理，本例目录为 C：\BookJavaVol1_3\Program\ch18\18_5\math3D。

文件 Ex18_5.java：设置立方体在旋转时的光影变化。

```
import math3D.*;
```

```
071 import java.awt.*;
072 import java.awt.event.*;
073 import java.awt.Graphics;

074 public class Ex18_5 extends Frame implements Runnable {
075    double xa=-50, ya=50, za=50, xb=-50, yb=50, zb=-50;
076    double xc=50, yc=50, zc=-50, xd=50, yd=50, zd=50;
077    double xe=-50, ye=-50, ze=50, xf=-50, yf=-50, zf=-50;
078    double xg=50, yg=-50, zg=-50, xh=50, yh=-50, zh=50;

079    int xam, yam, xbm, ybm, xcm, ycm, xdm, ydm;
080    int xem, yem, xfm, yfm, xgm, ygm, xhm, yhm;

081    int xas, yas, xbs, ybs, xcs, ycs, xds, yds;
082    int xes, yes, xfs, yfs, xgs, ygs, xhs, yhs;

083    view_To_screen_coordinates vTs = new view_To_screen_coordinates();
084    project3D_To_2D pr = new project3D_To_2D();
085    rotate_calculator rc = new rotate_calculator();

086    normal_hidden P1 = new normal_hidden();
087    normal_hidden P2 = new normal_hidden();
088    normal_hidden P3 = new normal_hidden();
089    normal_hidden P4 = new normal_hidden();
090    normal_hidden P5 = new normal_hidden();
091    normal_hidden P6 = new normal_hidden();

092    public static void main(String args[]) {
093       Ex18_5 workStart=new Ex18_5();
094    }

095    public Ex18_5() {
096       super("Ex18_5");
097       setSize(400, 400);

098       enableEvents(AWTEvent.WINDOW_EVENT_MASK);

099       setVisible(true);
100       new Thread(this).start();
101       }

102    public void processWindowEvent(WindowEvent e) {
103       if(e.getID() == WindowEvent.WINDOW_CLOSING) {
104          System.exit(0);
105       }
106    }

107    public void run() {
108       while(true) {
109          rc.rotateY(xa,ya,za, 0.03);
110          xa=rc.getPtX(); ya=rc.getPtY(); za=rc.getPtZ();
111          rc.rotateY(xb,yb,zb, 0.03);
112          xb=rc.getPtX(); yb=rc.getPtY(); zb=rc.getPtZ();
113          rc.rotateY(xc,yc,zc, 0.03);
114          xc=rc.getPtX(); yc=rc.getPtY(); zc=rc.getPtZ();
115          rc.rotateY(xd,yd,zd, 0.03);
```

```
116         xd=rc.getPtX(); yd=rc.getPtY(); zd=rc.getPtZ();
117         rc.rotateY(xe,ye,ze, 0.03);
118         xe=rc.getPtX(); ye=rc.getPtY(); ze=rc.getPtZ();
119         rc.rotateY(xf,yf,zf, 0.03);
120         xf=rc.getPtX(); yf=rc.getPtY(); zf=rc.getPtZ();
121         rc.rotateY(xg,yg,zg, 0.03);
122         xg=rc.getPtX(); yg=rc.getPtY(); zg=rc.getPtZ();
123         rc.rotateY(xh,yh,zh, 0.03);
124         xh=rc.getPtX(); yh=rc.getPtY(); zh=rc.getPtZ();

125         P1.vectNormal(xd, yd, zd, xa, ya, za, xe, ye, ze);
126         P2.vectNormal(xc, yc, zc, xd, yd, zd, xh, yh, zh);
127         P3.vectNormal(xb, yb, zb, xc, yc, zc, xg, yg, zg);
128         P4.vectNormal(xa, ya, za, xb, yb, zb, xf, yf, zf);
129         P5.vectNormal(xc, yc, zc, xb, yb, zb, xa, ya, za);
130         P6.vectNormal(xh, yh, zh, xe, ye, ze, xf, yf, zf);

131         xam = (int)pr.projectPtX(xa,ya,za);
132         yam = (int)pr.projectPtY(xa,ya,za);
133         xbm = (int)pr.projectPtX(xb,yb,zb);
134         ybm = (int)pr.projectPtY(xb,yb,zb);
135         xcm = (int)pr.projectPtX(xc,yc,zc);
136         ycm = (int)pr.projectPtY(xc,yc,zc);
137         xdm = (int)pr.projectPtX(xd,yd,zd);
138         ydm = (int)pr.projectPtY(xd,yd,zd);
139         xem = (int)pr.projectPtX(xe,ye,ze);
140         yem = (int)pr.projectPtY(xe,ye,ze);
141         xfm = (int)pr.projectPtX(xf,yf,zf);
142         yfm = (int)pr.projectPtY(xf,yf,zf);
143         xgm = (int)pr.projectPtX(xg,yg,zg);
144         ygm = (int)pr.projectPtY(xg,yg,zg);
145         xhm = (int)pr.projectPtX(xh,yh,zh);
146         yhm = (int)pr.projectPtY(xh,yh,zh);

147         xas = vTs.viewX_To_screenX(xam);
148         yas = vTs.viewY_To_screenY(yam);
149         xbs = vTs.viewX_To_screenX(xbm);
150         ybs = vTs.viewY_To_screenY(ybm);
151         xcs = vTs.viewX_To_screenX(xcm);
152         ycs = vTs.viewY_To_screenY(ycm);
153         xds = vTs.viewX_To_screenX(xdm);
154         yds = vTs.viewY_To_screenY(ydm);
155         xes = vTs.viewX_To_screenX(xem);
156         yes = vTs.viewY_To_screenY(yem);
157         xfs = vTs.viewX_To_screenX(xfm);
158         yfs = vTs.viewY_To_screenY(yfm);
159         xgs = vTs.viewX_To_screenX(xgm);
160         ygs = vTs.viewY_To_screenY(ygm);
161         xhs = vTs.viewX_To_screenX(xhm);
162         yhs = vTs.viewY_To_screenY(yhm);

163         repaint();
164         try{Thread.sleep(150);}
165         catch(InterruptedException e) {;}
166     }
167 }
```

```
168  public void paint(Graphics g) {
169      int P1_cf = (int)P1.getHidden()/52;
170      int Xdaeh[] = {xds, xas, xes, xhs};
171      int Ydaeh[] = {yds, yas, yes, yhs};
172      int Ndaeh = 4;

173      int P2_cf = (int)P2.getHidden()/52;
174      int Xcdhg[] = {xcs, xds, xhs, xgs};
175      int Ycdhg[] = {ycs, yds, yhs, ygs};
176      int Ncdhg = 4;

177      int P3_cf = (int)P3.getHidden()/52;
178      int Xbcgf[] = {xbs, xcs, xgs, xfs};
179      int Ybcgf[] = {ybs, ycs, ygs, yfs};
180      int Nbcgf = 4;

181      int P4_cf = (int)P4.getHidden()/52;
182      int Xabfe[] = {xas, xbs, xfs, xes};
183      int Yabfe[] = {yas, ybs, yfs, yes};
184      int Nabfe = 4;

185      int P5_cf = (int)P5.getHidden()/52;
186      int Xadcb[] = {xas, xds, xcs, xbs};
187      int Yadcb[] = {yas, yds, ycs, ybs};
188      int Nadcb = 4;

189      int P6_cf = (int)P6.getHidden()/52;
190      int Xhefg[] = {xhs, xes, xfs, xgs};
191      int Yhefg[] = {xhs, xes, xfs, xgs};
192      int Nhefg = 4;

193      if (P1.getHidden() > 2400.0) {
194          g.setColor(new Color(P1_cf, P1_cf, P1_cf));
195          g.fillPolygon(Xdaeh, Ydaeh, Ndaeh);
196      }

197      if (P2.getHidden() > 2400.0) {
198          g.setColor(new Color(P2_cf, P2_cf, P2_cf));
199          g.fillPolygon(Xcdhg, Ycdhg, Ncdhg);
200      }

201      if (P3.getHidden() > 2400.0)    {
202          g.setColor(new Color(P3_cf, P3_cf, P3_cf));
203          g.fillPolygon(Xbcgf, Ybcgf, Nbcgf);
204      }

205      if (P4.getHidden() > 2400.0) {
206          g.setColor(new Color(P4_cf, P4_cf, P4_cf));
207          g.fillPolygon(Xabfe, Yabfe, Nabfe);
208      }

209      if (P5.getHidden() > 2400.0) {
210          g.setColor(new Color(P5_cf, P5_cf, P5_cf));
211          g.fillPolygon(Xadcb, Yadcb, Nadcb);
212      }

213      if (P6.getHidden() > 2400.0) {
```

```
214          g.setColor(new Color(P6_cf, P6_cf, P6_cf));
215          g.fillPolygon(Xhefg, Yhefg, Nhefg);
216       }
217    }
218 }
```

行 169　　声明平面 P1 RGB 色值的变量，并以 Nz/n 设置其值。
行 170　　声明多边形 x 轴坐标的数组变量，并以 P1 各点坐标设置其值。
行 171　　声明多边形 y 轴坐标的数组变量，并以 P1 各点坐标设置其值。
行 172　　声明多边形各点数量的变量，并以 P1 各点数量设置其值。
行 173～192　与行 169～172 相同。
行 193～196　以 169～172 设置的值绘制填色多边形 P1。
行 197～216　与行 193～196 相同。

编译程序

输入“javac Ex18_5.java”进行编译，如图 18-11 所示。

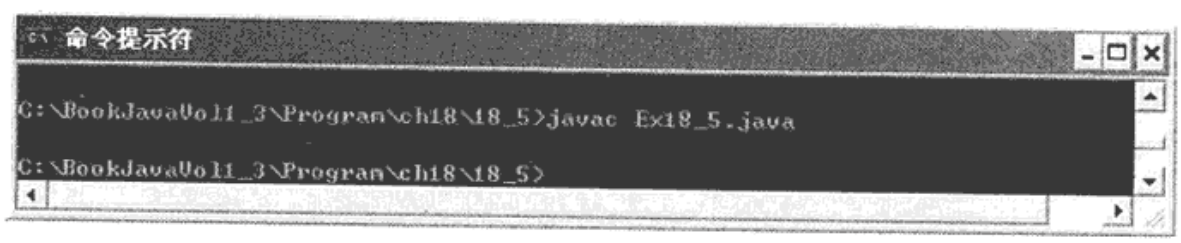


图 18-11

运行程序

输入“java Ex18_5”，如图 18-12 所示。

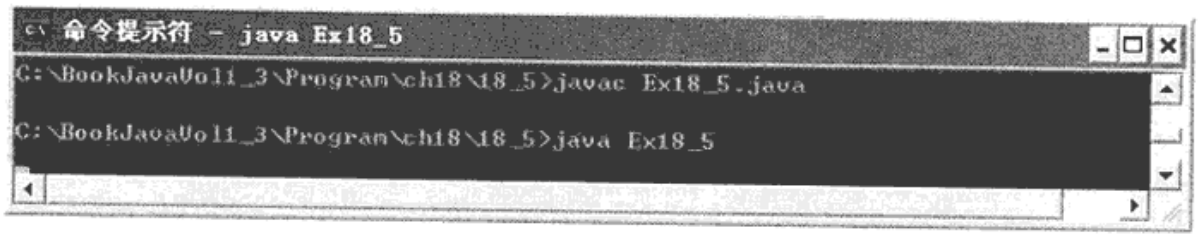


图 18-12

运行结果

绘制 3D 立方体围绕 y 轴旋转，显示在旋转时的光影变化，如图 18-13 所示。

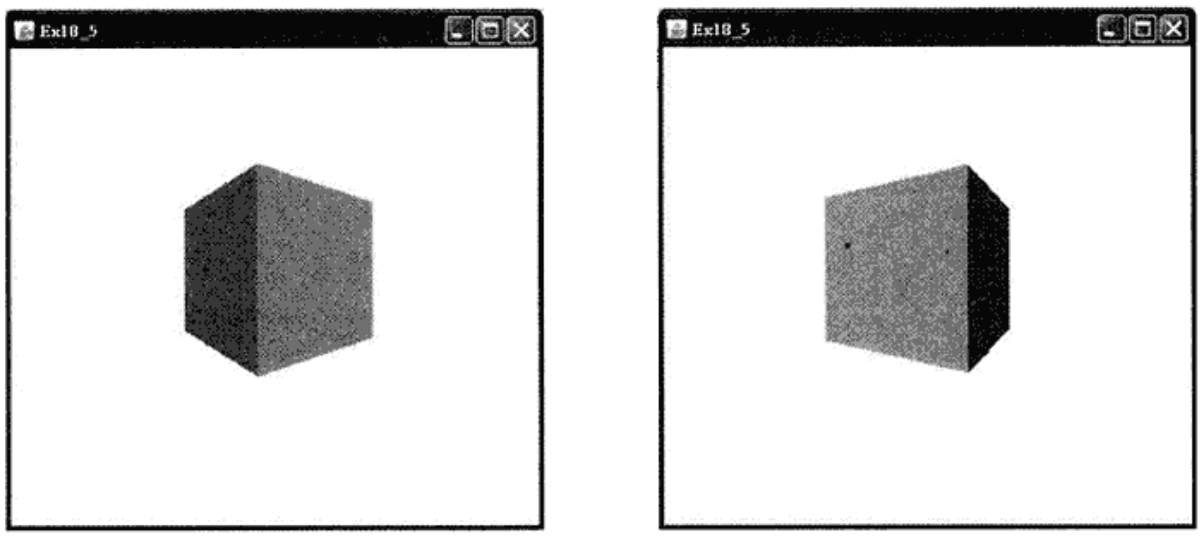


图 18-13

18-6 习题

1. 立体对象的显示与平面对象的显示有何不同？
2. 什么是法线向量？
3. 如何求多边形的法线向量？
4. 设有一个由向量 U 与 V 组成的平面，则该平面的法线如何表示？
5. 在视觉上，法线代表什么意义？
6. 如何以法线设置隐藏条件？
7. 为何要进行视角修正？
8. 试修改范例 78，消除其图像闪烁。

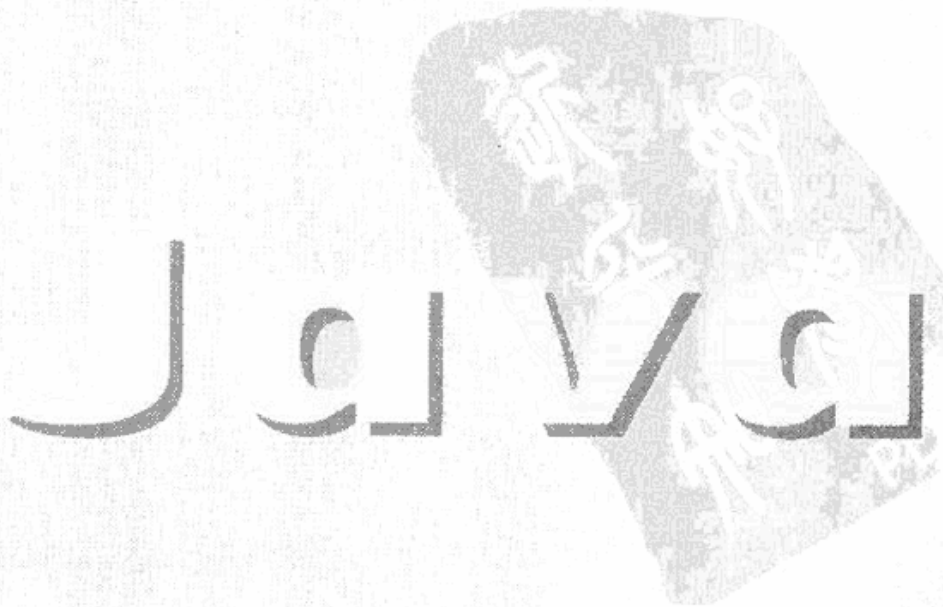
PART

06

Java Applet与网页

本书探讨的是动画游戏，有文字、图像、动画，这些都需要一个环境来显示，常用的显示工具是框架（Frame）与浏览器（Browser），前者可用于单机显示或多机网络对阵；后者可用于单机网络显示。

在第1部分～第5部分的内容中均以框架作为显示环境，无论是单机显示或多机网络对阵，我们都已体会到了框架的强大功能。Java早期的游戏环境却是浏览器环境，用于单机网络显示，属于网络广播性质，但它无法显示网络在线对阵游戏。



Chapter 19　第一个Java Applet程序

Sun公司开发的Java Applet可以嵌入到网页应用程序中，当网页被用户浏览时，网页的文字、图案随Applet自动通过网络运行，具有交互功能。即当游戏在作为服务器端的网站上创建完成后，用户可在网络的任一台客户端计算机上打开浏览器，输入该网站网址，以键盘或鼠标在客户端上交互控制服务器端的游戏操作。

Chapter 20　基础图文处理

参考本书第1章和第2章，我们曾以框架环境进行图文处理，本章将在网络浏览器中运行Java Applet程序进行基础图文处理，读者可观察两者间的异同，前者强调本机运行，后者强调网络广播。

Chapter 21　动画与事件

对于框架环境，本书已在第4章中讨论了框架动画，在第6章中讨论了框架鼠标事件，在第7章中讨论了框架键盘事件。本章将探讨网络浏览器环境中的动画与事件，我们将看到远程客户端如何以事件（Events）控制服务器端的图案。

Chapter 19 第一个 Java Applet 程序

19-1 简介

Sun 公司开发的 Java Applet 可以嵌入到网页应用程序中，当网页被用户浏览时，网页的文字、图案随 Applet 自动通过网络运行，具有交互功能。即当游戏在作为服务器端的网站上创建完成后，用户可在网络的任一台客户端计算机上打开浏览器，输入该网站网址，以键盘或鼠标在客户端上交互控制服务器端的游戏操作。

19-2 编写 Java Applet 与 HTML 程序

在网络上运行 Java Applet 程序时，必须配合 HTML 程序，即利用 HTML 强大的网络能力来管理屏幕画面(Screen Manager)。本节将介绍如何编写一个最基础的 Java Applet 与 HTML 程序。

设计范例 79，创建第一个 Java Applet 程序，用于显示字符串；另创建一个 HTML 程序，用于配合 Java Applet 程序的运行。

范例 79 设计文件 firstApplet.java、Ex19_2.html，其功能是解释 Java Applet 应用。

文件 firstApplet.java：编写在记事本中，功能是显示字符串。

```
01 import java.awt.Graphics;
02 import java.applet.*;

03 public class firstApplet extends Applet {
04     String message = "My firstApplet WWW Program";

05     public void paint(Graphics g){
06         g.drawString(message,100,45);
07     }
08 }
```

行 01	导入系统内置的 java.awt.Graphics 包，在行 05～07 中将会使用到。
行 02	导入系统内置的 java.applet.*包，将使用于本程序。
行 03～08	本例主体。
行 03	firstApplet 类继承自 java.applet.Applet。
行 04	声明字符串变量 message 并设置其初值。

行 05～07	paint 方法，使用行 01 的 java.awt.Graphics 包。
行 05	使用默认参数 Graphics g。
行 06	使用内置方法 g.drawString 显示 message 的内容。其中，100 为 x 轴的位置坐标；45 为 y 轴的位置坐标。

编译文件 firstApplet.java，生成 firstApplet.class，配合 Ex19_2.html 运行，如图 19-1 所示。

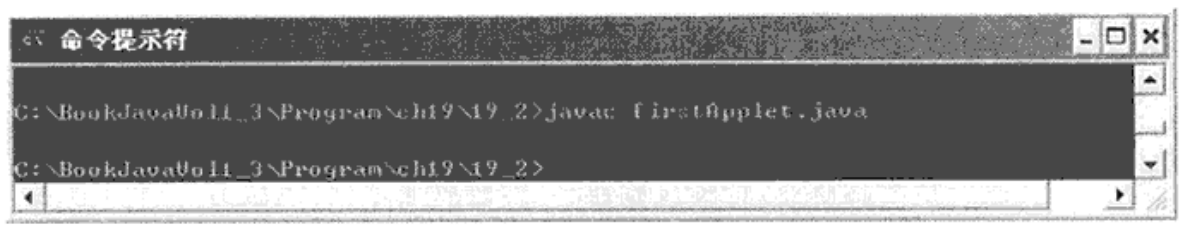


图 19-1

文件 Ex19_2.html：编写在记事本中，功能是使 firstApplet.class 能在网络中运行。

```
09 <HTML>
10 <BODY BGCOLOR="#eeeeee">
11 <CENTER>
12 <APPLET CODE="firstApplet" WIDTH=350 HEIGHT=350>
13 </APPLET>
14 </HTML>
```

行 09～14	为 HTML 程序，其功能是使 Java Applet 程序能在网页上运行。有关内容位于标签<HTML>与</HTML>之间。
行 10	设置屏幕背景色。
行 11	将显示区设置在屏幕中间。
行 12	CODE 为导引的.class 文件的名称 xxx.class，但可省略扩展名.class（本例为 firstApplet.class 的 firstApplet）。WIDTH 为显示区的宽度，HEIGHT 为显示区的高度。

运行结果

以鼠标双击文件 Ex19_2.html，如图 19-2 所示。

图 19-2

讨论事项

本例的运行结果仅显示在目录 C：\BookJavaVol1_3\Program\ch19\19_2\Ex19_2.html 上，而不是显示在网络上，我们将在 19-3 节中探讨如何在网站的网页上应用 Java Applet 程序。

19-3 网站上应用 Applet

完成 Java Applet 与 HTML 程序的编写后，必须在操作系统中创建网站虚拟目录，将设计内容以网页的形式发布出去，也称为“架设网站”。

19-3-1 网站架设

如前所述，Java 语言在网络应用上有其震撼的功能，无须像其他语言那样费时费力地搭建网站，Java 只需简单数行程序代码即可构建网络平台，如第 3 部分“在线游戏”。但本部分的 Java Applet 程序需与浏览器搭配使用，所以仍应在操作系统中创建网站。其步骤如下（以 Windows XP 为例）：

1 选择【开始】→【控制面板】命令，在弹出的窗口中双击【管理工具】，如图 19-3 所示。

图 19-3

2 单击【Internet 信息服务】，如图 19-4 所示。

图 19-4

3 展开【本地计算机】→【网站】，右击【默认网站】，在弹出的快捷菜单中选择【新建】→【虚拟目录】，如图 19-5 所示。

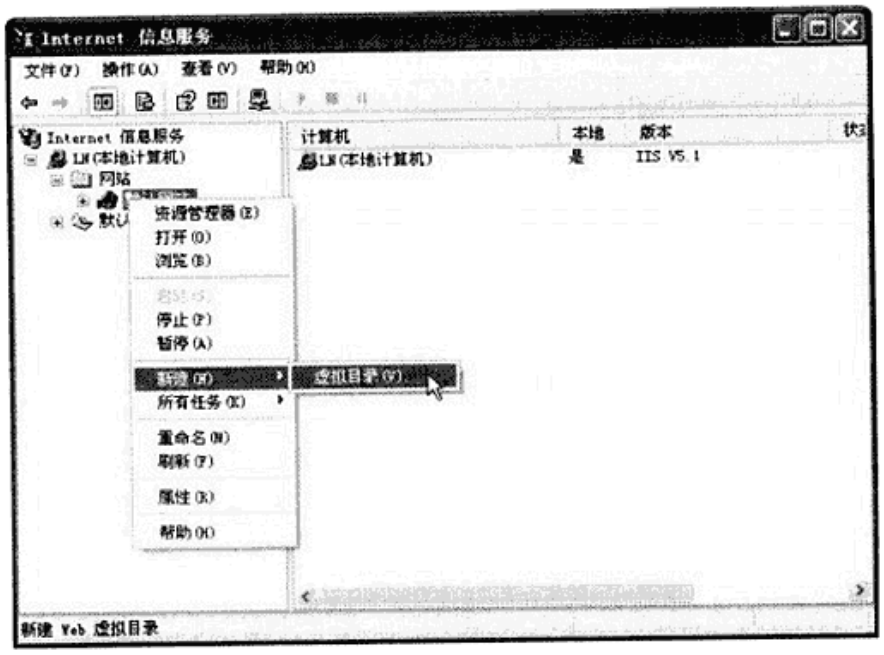


图 19-5

4 在弹出的对话框中单击【下一步】。

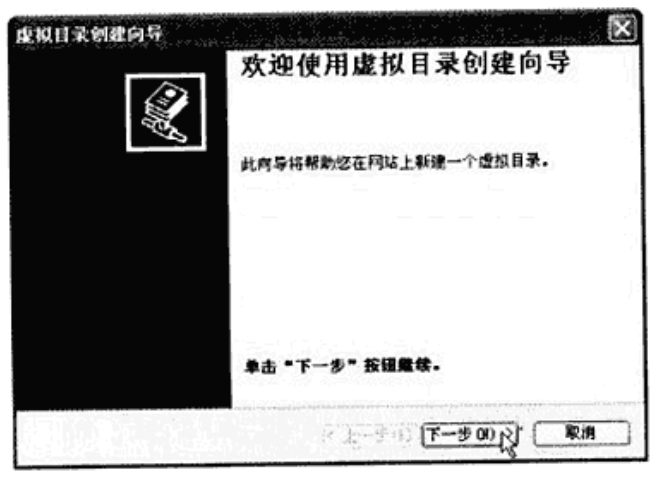


图 19-6

5 在“别名”文本框中输入虚拟目录的别名（本例为“ch19”），然后单击【下一步】，如图 19-7 所示。

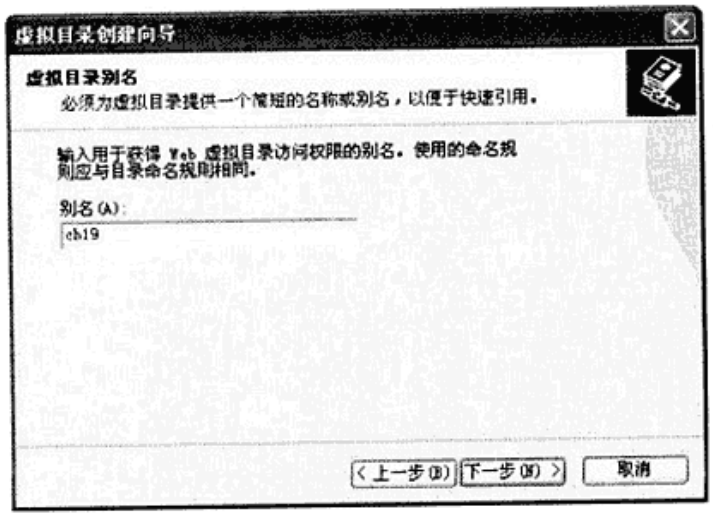


图 19-7

6 单击【浏览】，如图 19–8 所示。

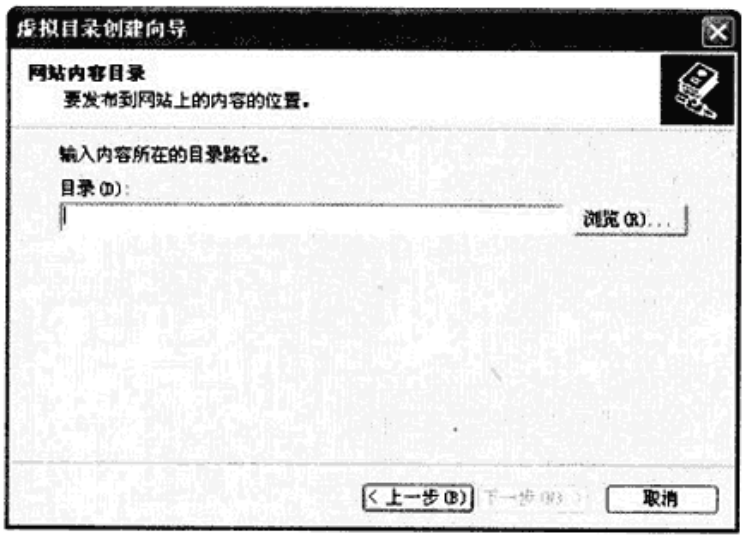


图 19-8

7 选择 HTML 文件的目录（本例为 "C：\java\Program\ch19\19_2"），然后单击【确定】，如图 19–9 所示。

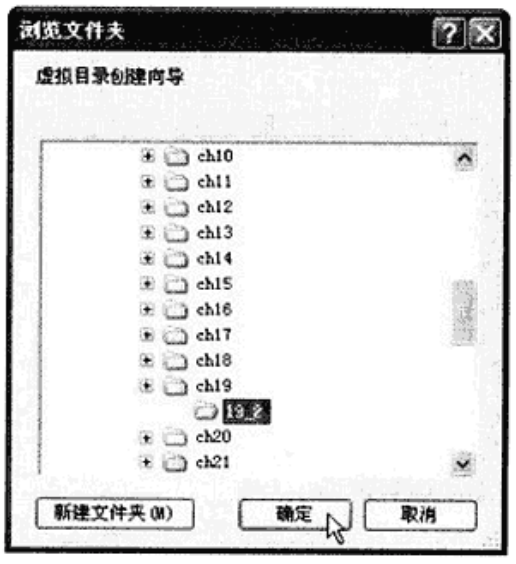


图 19-9

8 返回原对话框后单击【下一步】，如图 19–10 所示。

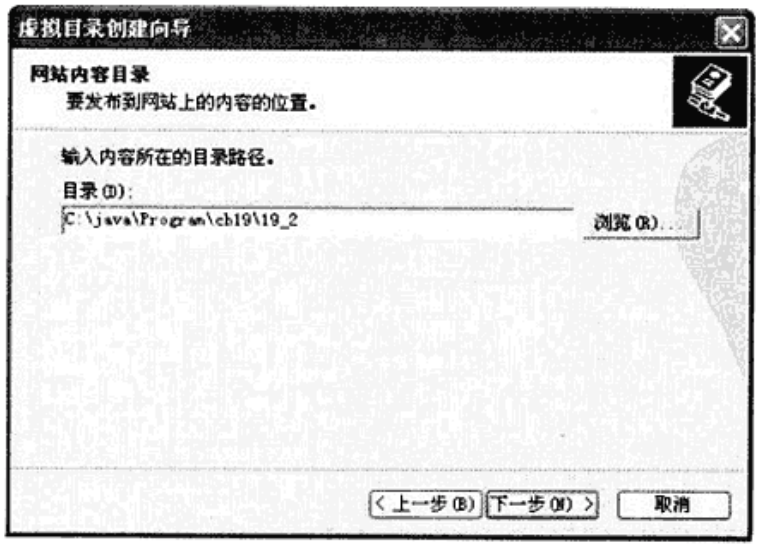


图 19-10

9 单击【下一步】，如图 19–11 所示。

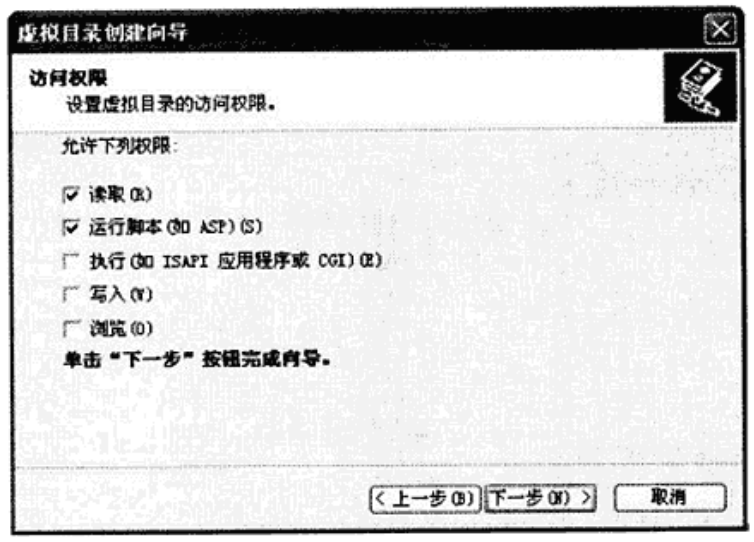


图 19-11

10 单击【完成】，将 HTML 文件创建成网站虚拟目录，如图 19-12 所示。

图 19-12

19-3-2 查看本机的 IP 地址

读者若尚不知道本机的 IP 地址，则可在控制面板中按下列步骤查看。

1 选择【开始】→【控制面板】命令，在弹出的窗口中双击【网络连接】，如图 19-13 所示。

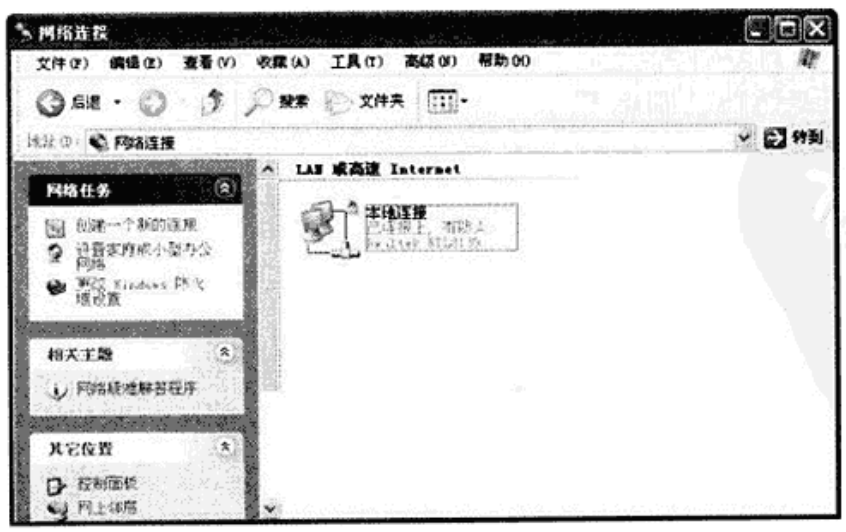


图 19-13

2 右击【本地连接】，在弹出的快捷菜单中选择【属性】命令，如图 19-14 所示。

图 19-14

3　在【常规】选项卡中单击【Internet 协议（TCP/IP)】→【属性】，如图 19–15 所示。

图 19-15

4　通过“IP 地址”文本框查看本机的 IP 地址（本例中为 192.168.1.3)，如图 19–16 所示。

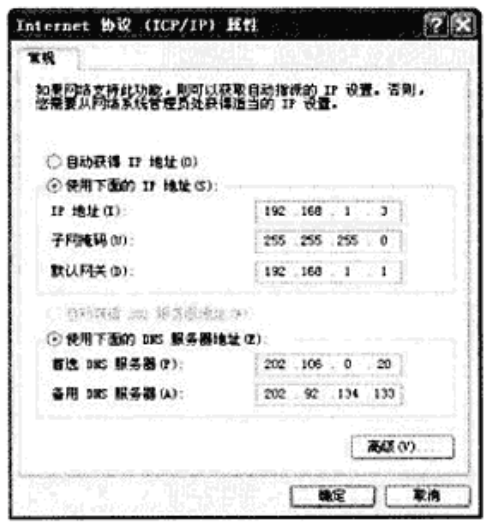


图 19-16

19-3-3　网络浏览器

1　在任意网络计算机中打开浏览器。

2　输入网址“http：//本地计算机 IP/虚拟目录/HTML 程序”，本例中为“http：//

192.168.1.3/ch19/Ex19_2.html"。打开的网页如图 19-17 所示。

其中，网站计算机 IP：本例中为 192.168.1.3，参考 19-3-2 节；虚拟目录：本例中为 ch19，参考 19-3-1 节；HTML 程序：本例中为 Ex19_2.html。

图 19-17

19-4 习题

1. 在图像显示上，框架（Frame）与浏览器（Browser）的功能有何差异？
2. Java Applet 程序需配合 HTML 程序使用，试述连接两者的程序代码是什么？
3. 在 Windows XP 操作系统中如何创建网站？
4. 在任意网络计算机中打开浏览器后，若要使用新建网站的网页，则应如何输入网址？

Chapter 20 基础图文处理

20-1 简介

参考本书第 1 章和第 2 章，我们曾以框架环境进行图文处理，本章将在网络浏览器中运行 Java Applet 程序进行基础图文处理，读者可观察两者间的异同，前者强调本机运行，后者强调网络广播。

20-2 文字处理

参考第 1 章，设计范例 80，用于文字处理，创建 Java Applet 程序来显示字符串；另创建 HTML 程序，用于运行 Java Applet 程序（参考第 19 章）。

范例 80 设计文件 Ex20_2.java、Ex20_2.html，其功能是解释如何使用 Java Applet 在网络中进行文字处理。

文件 Ex20_2.java：编写在记事本中，功能是显示中英文字符串。

```
01 import java.awt.Graphics;
02 import java.awt.Font;
03 import java.awt.Color;
04 import java.applet.*;


05 public class Ex20_2 extends Applet {
06   Font messageFont1 = new Font("TimesRoman", Font.PLAIN, 30);
07   Font messageFont2 = new Font("宋体", Font.PLAIN, 30);
08   Font messageFont3 = new Font("楷体", Font.PLAIN, 30);

09   String messageEnglish = "English test string";
10   String messageChinese = "中文测试字符串";

11   public void paint(Graphics g) {
12     g.setFont(messageFont1);
13     g.setColor(Color.red);
14     g.drawString(messageEnglish, 10, 50);

15     g.setFont(messageFont2);
16     g.setColor(Color.green);
17     g.drawString(messageChinese, 10, 100);

18     g.setFont(messageFont3);
19     g.setColor(Color.blue);
```

```
20      g.drawString(messageChinese, 10, 150);
21    }
22 }
```

行 06	设置英文字体。
行 07～08	设置中文字体。
行 09	设置英文字符串。
行 10	设置中文字符串。
行 12～14	显示英文字符串。
行 15～20	显示中文字符串。

编译文件 Ex20_2.java，生成 Ex20_2.class，配合 Ex20_2.html 运行，如图 20–1 所示。

图 20-1

文件 Ex20_2.html：编写在记事本中，功能是使 Ex20_2.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex20_2" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1　参考 19–3 节运行，结果如图 20–2 所示。

本例中设置虚拟目录的别名为 ch20；网站内容目录为 C：\Java\Program\ch20\20_2。

2　本例中网站的网址为 http：//192.168.1.3/ch20_2/Ex20_2.html。

注意：以上为参考数据，读者应根据自己的计算机环境参考 19–3 节进行测试。

图 20-2

20-3 图案绘制

参考第 2 章，设计范例 81，用于图案绘制，创建 Java Applet 程序来绘制基础图形；另创建 HTML 程序，用于运行 Java Applet 程序（参考第 19 章）。

范例 81 设计文件 Ex20_3.java、Ex20_3.html，其功能是解释如何使用 Java Applet 在网络中进行图案绘制。

文件 Ex20_3.java：编写在记事本中，功能是绘制基础图形。

```
01 import java.awt.Graphics;
02 import java.applet.Applet;

03 public class Ex20_3 extends Applet {
04    public void paint(Graphics g) {
05      g.drawLine(100,35,250,35);

06      g.drawRect(60,70,100,50);
07      g.fillRect(200,70,100,50);

08      g.drawOval(30,160,80,60);
09      g.drawOval(140,160,80,60);
10      g.drawRect(140,160,80,60);
11      g.fillOval(250,160,80,60);

12      g.drawArc(30,260,80,60,10,90);
13      g.drawArc(140,260,80,60,10,90);
14      g.drawRect(140,260,80,60);
15      g.fillArc(250,260,80,60,10,90);
16   }
17 }
```

行 05　绘制直线。
行 06～07　绘制长方形。
行 08～11　绘制椭圆。
行 12～15　绘制弧形。

编译文件 Ex20_3.java，生成 Ex20_3.class，配合 Ex20_3.html 运行，如图 20–3 所示。

图 20-3

文件 Ex20_3.html：编写在记事本中，功能是使 Ex20_3.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex20_3" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1 参考 19–3 节运行，结果如图 20–4 所示。
本例中设置虚拟目录的别名为 ch20；网站内容目录为 C：\Java\Program\ch20\20_3。

2 本例中网站的网址为 http：//192.168.1.3/ch20/Ex20_3.html。

注意：以上为参考数据，读者应根据自己的计算机环境参考 19–3 节来操作。

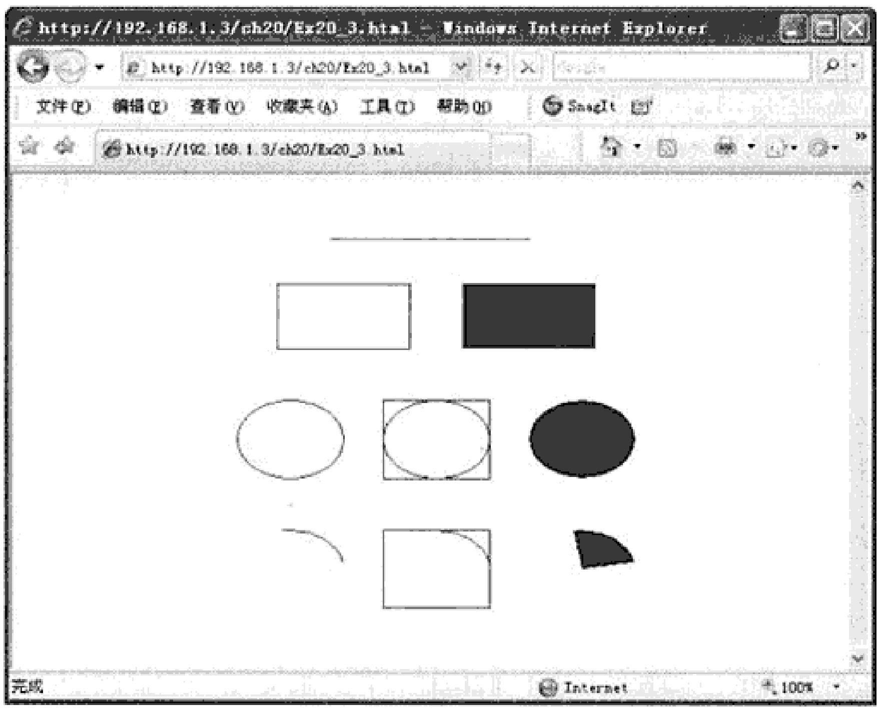


图 20-4

20-4 图片引用

参考第 3 章，设计范例 82，用于图片引用，创建 Java Applet 程序来显示图片；另创建 HTML 程序，用于运行 Java Applet 程序（参考第 19 章）。

范例 82 设计文件 Ex20_4.java、Ex20_4.html，其功能是解释如何使用 Java Applet 在网络中进行图片引用。

文件 Ex20_4.java：编写在记事本中，功能是图像文件的引用。

```
import java.awt.Graphics;
import java.awt.Image;
import java.applet.Applet;

public class Ex20_4 extends Applet {
  Image Img;

  public void init() {
    Img = getImage(getDocumentBase(), "pic.JPG");
  }

  public void paint(Graphics g) {
    g.drawImage(Img, 170, 170, this);
  }
```

```
12 }
```

行 05	声明图像文件变量。
行 06～08	依惯例将程序的初始内容放在此方法中。
行 07	读取图像文件。
行 10	在指定的坐标上显示图像文件。

编译文件 Ex20_4.java，生成 Ex20_4.class，配合 Ex20_4.html 运行，如图 20–5 所示。

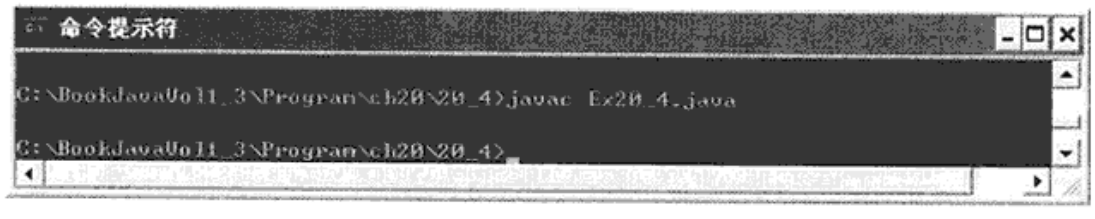


图 20-5

文件 Ex20_4.html：编写在记事本中，功能是使 Ex20_4.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex20_4" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1. 参考 19–3 节运行，结果如图 20–6 所示。
   本例中设置虚拟目录的别名为 ch20；网站内容目录为 C：\Java\Program\ch20\20_4。
2. 本例中网站的网址为 http：//192.168.1.3/ch20/Ex20_4.html。

注意：以上为参考数据，读者应根据自己的计算机环境参考 19–3 节来操作。

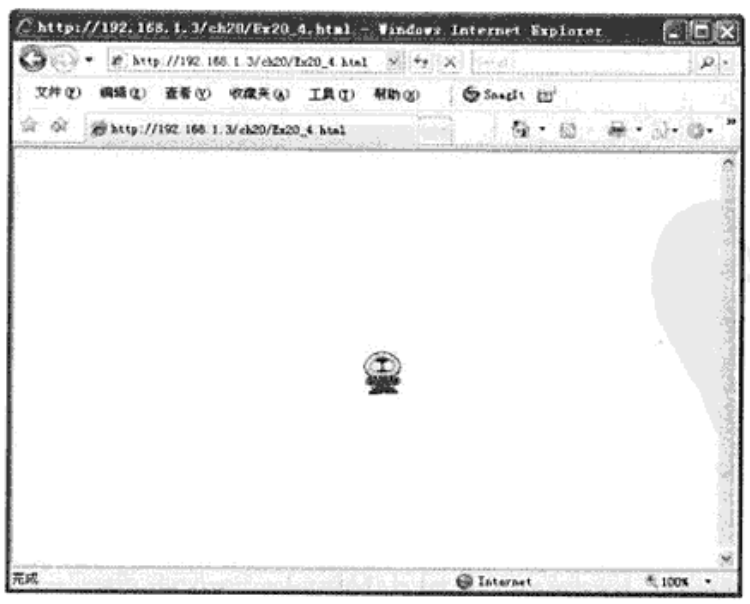


图 20-6

20-5 习题

1. 试使用 Java Applet 编写一个能绘制多边形的程序。
2. 无论是长方形、椭圆、弧形或图片，其绘制坐标在图案的哪个部位？
3. 设计 Java Applet 程序时，它读取图片的程序代码是什么？

Chapter 21 动画与事件

21-1 简介

对于框架环境，本书已在第 4 章中讨论了框架动画，在第 6 章中讨论了框架鼠标事件，在第 7 章中讨论了框架键盘事件。本章将探讨网络浏览器环境中的动画与事件，我们将看到远程客户端如何以事件（Events）控制服务器端的图案。

21-2 线程工作流程

在 1–3 节中，我们曾谈到框架环境中的线程工作流程，并描述流程的格式。本章将就 Applet 与网络浏览器环境，讨论其线程工作流程，并描述流程的格式。

每一个动画动作即为一个线程（Thread）。在同一时刻内，CPU 仅能处理一个工作。当有多个工作同时进入时，CPU 将其本身分割成多个工作时段，适当地分配给各工作运行。即当有多个工作同时进入时，哪一个工作的条件最好，该工作就可先抢到 CPU 的工作时段，如此抢 CPU 的工作时段既是线程的意义，也是并行工作的开始。

为了让多个线程井然有序地并行，Applet 网络浏览器环境中有如下的流程格式。

```
01 import java.awt.*;
02 import java.applet.*;

03 public class testProgram extends Applet implements Runnable {
04     …
05     public void init () {
06         …
07     }

08     public void start() {
09         …
10         new Thread(this).start();
11     }

12     public void run () {
13         …
14         repaint();
15     }

16     public void paint(Graphics g) {
```

```
17          …
18      }
19  }
```

行 03～19	创建类，继承自 Applet，实现了 Runnable 接口。
行 04	变量声明区。
行 05～07	设置初始数据或初始动作。
行 08～11	启动线程工作机制，使多个线程并行，并由行 10 驱动行 12～15。
行 12～15	运行各项工作，并由行 14 驱动行 16～18。
行 16～18	绘制有关图案。

21-3 动画设计

参考范例 18.1，设计 Applet 网络浏览器环境中单幅动画的应用，并考虑当图案移动触及边线时立即折返移动。

范例 83 设计文件 Ex21_3_1.java、Ex21_3_1.html，其功能是解释 Java Applet 在网络中运行单幅动画的应用。

文件 Ex21_3_1.java：编写在记事本中，功能是制作单幅动画。

```
01 import java.awt.*;
02 import java.applet.*;

03 public class Ex21_3_1 extends Applet implements Runnable {
04      int x=0, y=100;
05      int dx=5, dy=5;
06      Image img;

07      public void init() {
08          img = getImage(getDocumentBase(), "fly.gif");
09      }

10      public void start() {
11          new Thread(this).start();
12      }

13      public void run() {
14          while(true) {
15              x = x + dx;
16              y = y + dy;
17              repaint();

18              if(x<=0) dx = 5;
19              else if((x + 50) >= getWidth()) dx = -5;

20              if(y<=0) dy = 5;
21              else if((y + 50) >= getHeight()) dy = -5;

22              try{Thread.sleep(250);}
23              catch(InterruptedException e) {;}
24          }
25      }
```

```
26      public void paint(Graphics g) {
27          g.drawImage(img, x, y, this);
28      }
29 }
```

行 08	读取图像文件。
行 11	启动线程工作机制，驱动行 13～25。
行 14～24	为 while 循环，每循环一次就改变图像坐标一次。
行 15～16	设置每次循环的新坐标。
行 18～19	当图像抵达左右边线时，设置折返移动坐标变化，以 getWidth()读取右端边线位置。
行 20～21	当图像抵达上下边线时，设置折返移动坐标变化，以 getHeight()读取下端边线位置。
行 17	更新图像，驱动行 26～28。
行 27	绘制图片。

编译文件 Ex21_3_1.java，生成 Ex21_3_1.class，配合 Ex21_3_1.html 运行，如图 21-1 所示。

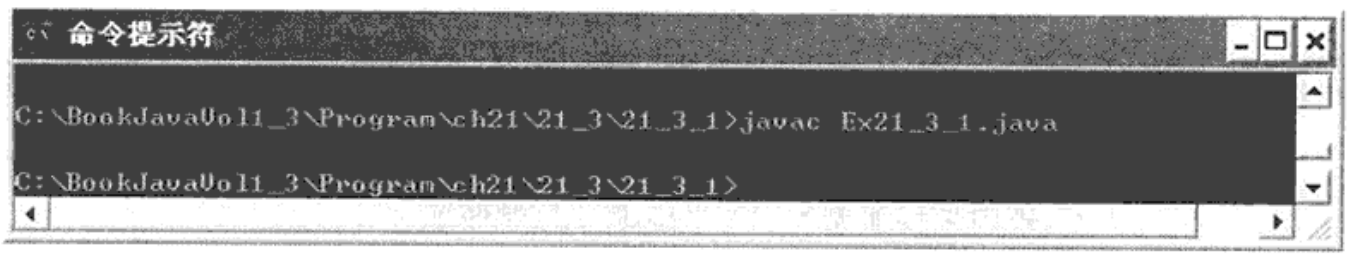


图 21-1

文件 Ex21_3_1.html：编写在记事本中，功能是使 Ex21_3_1.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex21_3_1" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1. 参考 19-3 节运行，结果如图 21-2 所示。
   本例中设置虚拟目录的别名为 ch21_3_1；网站内容目录为 C：\Java\Program\ch21\21_3\21_3_1。
2. 本例中网站的网址为 http：//192.168.1.3/ch21_3_1/Ex21_3_1.html。

注意：以上为参考数据，读者应根据自己的计算机环境参考 19-3 节来操作。

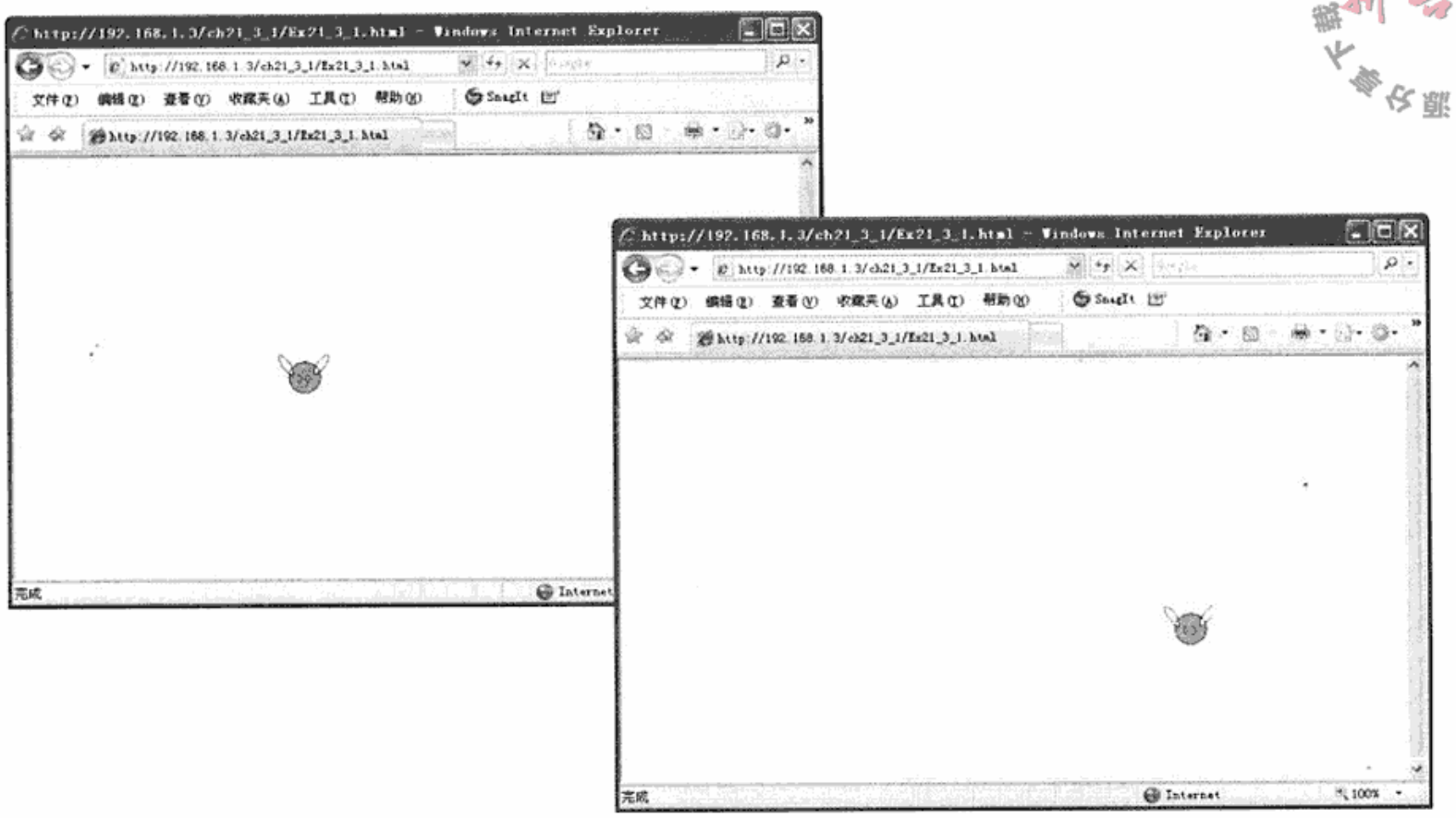


图 21-2

单幅动画不够灵活生动，要使动画生动就需运行多幅动画。其设计方法如下：

(1) 设置图像坐标，随用户的需要而变化该坐标。

(2) 设计多幅连续动作的图像文件，轮流显示并移动坐标，即可产生可爱生动的动画。

参考范例 19，设计 Applet 网络浏览器环境多幅动画的应用，并考虑当图像移动触及边线时立即折返移动。

范例 84 设计文件 Ex21_3_2.java、Ex21_3_2.html，其功能是解释 Java Applet 在网络中运行多幅动画的应用。

文件 Ex21_3_2.java：编写在记事本中，功能是制作多幅动画。

```
01 import java.awt.*;
02 import java.applet.*;

03 public class Ex21_3_2 extends Applet implements Runnable {
04     int num=0, flag;
05     int x=0, y=100, dx=5, dy=5;
06     Image img0, img1, img2;

07     public void init() {
08        img0 = getImage(getDocumentBase(), "fly0.gif");
09        img1 = getImage(getDocumentBase(), "fly1.gif");
10        img2 = getImage(getDocumentBase(), "fly2.gif");
11     }

12     public void start() {
13        new Thread(this).start();
14     }

15     public void run() {
```

```
16          while(true) {
17              x = x + dx;
18              y = y + dy;
19              flag = num % 3;
20              repaint();
21              num = num + 1;

22              if(x <= 0) dx = 5;
23              else if((x+60) >= 350) dx = -5;

24              if(y<=0) dy = 5;
25              else if((y + 50) >= getHeight()) dy = -5;

26              try{Thread.sleep(250);}
27              catch(InterruptedException e) {;}
28          }
29      }

30      public void paint(Graphics g) {
31          if(flag == 0)
32              g.drawImage(img0, x, y, this);
33          else if(flag == 1)
34              g.drawImage(img1, x, y, this);
35          else if(flag == 2)
36              g.drawImage(img2, x, y, this);
37      }
38  }
```

行 06	声明图像变量 img0、img1、img2。
行 08～10	读取 3 组图像。
行 19	循环使用 3 组图像，将 num 除以 3，取其余数，若余数为 0，则取用 img0；若余数为 1，则取用 img1；若余数为 2，则取用 img2。
行 21	每次循环将 num 加 1，配合行 19 选择一幅图像。
行 31～36	如行 19 所述，依序循环选择各幅图像并显示。

编译文件 Ex21_3_2.java，生成 Ex21_3_2.class，配合 Ex21_3_2.html 运行，如图 21-3 所示。

图 21-3

文件 Ex21_3_2.html：编写在记事本中，功能是使 Ex21_3_2.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex21_3_2" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1 参考 19–3 节运行，结果如图 21–4 所示。
本例中设置虚拟目录的别名为 ch21_3_2；网站内容目录为 C：\Java\Program\ch21\21_3\21_3_2。

2 本例中网站的网址为 http：//192.168.1.3/ch21_3_2/Ex21_3_2.html。

注意：以上为参考数据，读者应根据自己的计算机环境参考 19–3 节来操作。

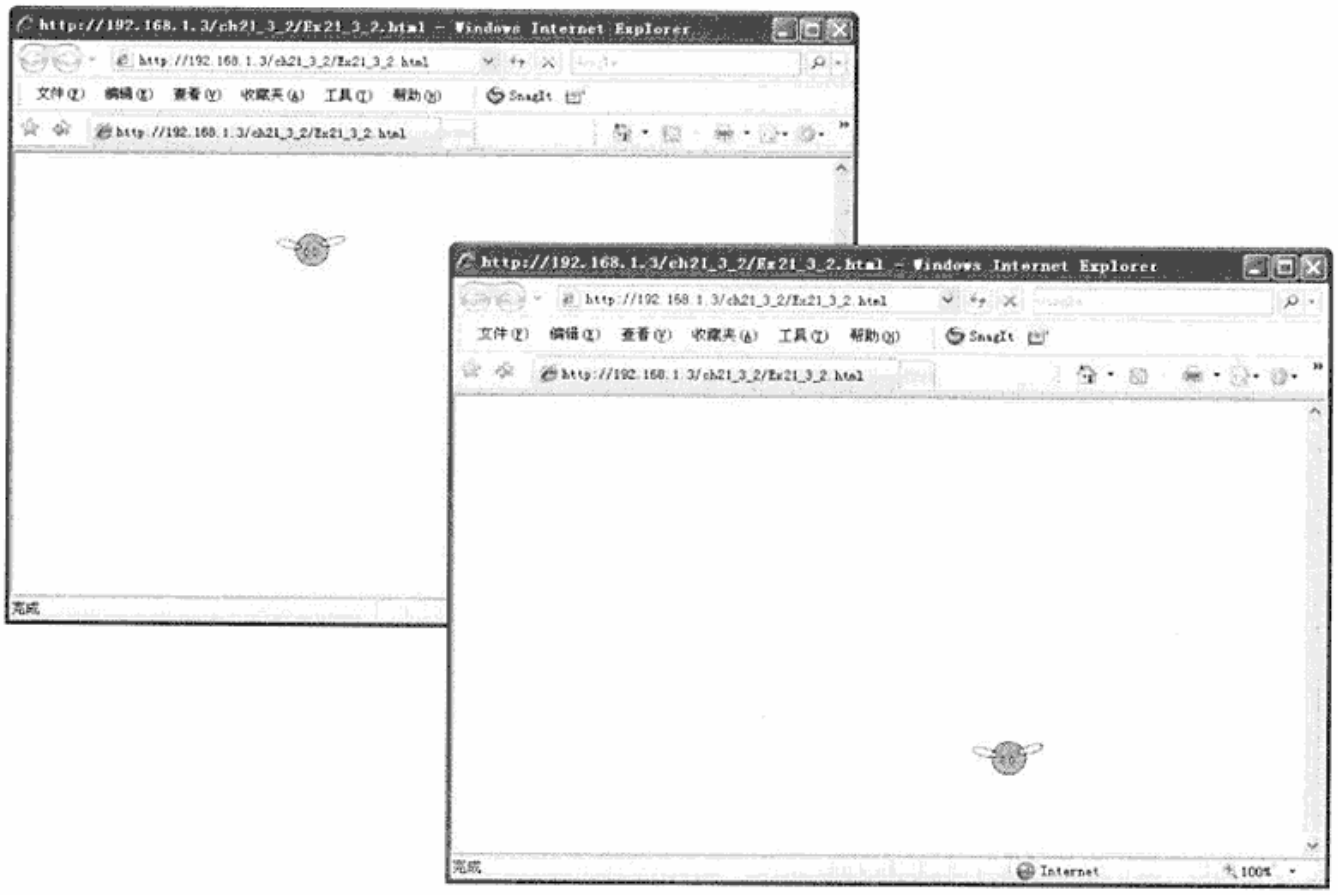


图 21-4

21-4 鼠标事件

回顾 5–6 节，每次发生鼠标事件时，我们可立即读取该事件的源（Source）、标识码（ID）、发生时间（When）、x 轴坐标、y 轴坐标。因为这些信息是伴随事件的发生而产生的，所以我们可利用这些信息来执行想要的后续操作，如移动（Moved）、拖动（Dragged）、选择（Selected）、随动（Followed）等。本节将它们用于 Applet 网页设计。

参考范例 33，设计 Applet 网络浏览器环境中鼠标事件的应用，当单击鼠标左键时发生鼠标事件，此时我们可读取鼠标指针当前的位置坐标(x，y)，并将图像显示在该坐标上，使图像随鼠标指针的位置而移动。

范例 85 设计文件 Ex21_4_1.java、Ex21_4_1.html，其功能是解释 Java Applet 在网络中运行鼠标事件的应用。

文件 Ex21_4_1.java：编写在记事本中，功能是解释鼠标事件的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
```

```
03 import java.applet.*;

04 public class Ex21_4_1 extends Applet implements Runnable {

05     int x=50, y=50;
06     Image img;

07     public void init() {
08        img = getImage(getDocumentBase(), "imgBoy.jpg");
09        enableEvents(AWTEvent.MOUSE_EVENT_MASK);
10     }

11     public void processMouseEvent(MouseEvent e) {
12           if(e.getID() == MouseEvent.MOUSE_PRESSED) {
13                x = e.getX();
14                y = e.getY();
15           }
16     }

17     public void start() {
18        new Thread(this).start();
19     }

20     public void run() {
21        while(true) {
22           repaint();
23           try{Thread.sleep(250);}
24           catch(InterruptedException e) {;}
25        }
26     }

27     public void paint(Graphics g) {
28           g.drawImage(img, x, y, this);
29     }
30 }
```

行 09	启动鼠标事件。
行 11～16	当单击鼠标左键时发生鼠标事件，同时记录鼠标指针当前的位置坐标(x，y)。
行 28	将图像显示于新位置坐标(x，y)。
行 18	以线程驱动行 20～26 的 run()。
行 22	以 repaint()方法驱动行 27～29 的 paint()方法，进行更新绘制。

编译文件 Ex21_4_1.java，生成 Ex21_4_1.class，配合 Ex21_4_1.html 运行，如图 21-5 所示。

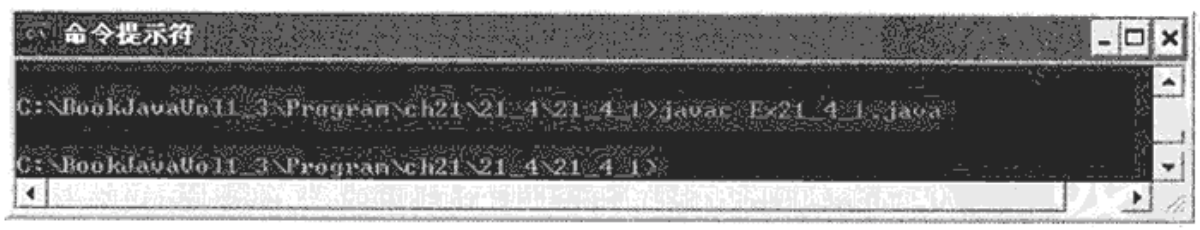


图 21-5

文件 Ex21_4_1.html：编写在记事本中，功能是使 Ex21_4_1.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#cccccc">
```

```
<CENTER>
<APPLET CODE="Ex21_4_1" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1 参考 19—3 节运行，结果如图 21—6 所示。
本例中设置虚拟目录的别名为 ch21_4_1；网站内容目录为 C：\Java\Program\ch21\21_4\21_4_1。

2 本例中网站的网址为 http：//192.168.1.3/ch21_4_1/Ex21_4_1.html。

3 当单击鼠标左键时，图像将随鼠标指针的位置而移动。

注意：以上为参考数据，读者应根据自己的计算机环境参考 19—3 节来操作。

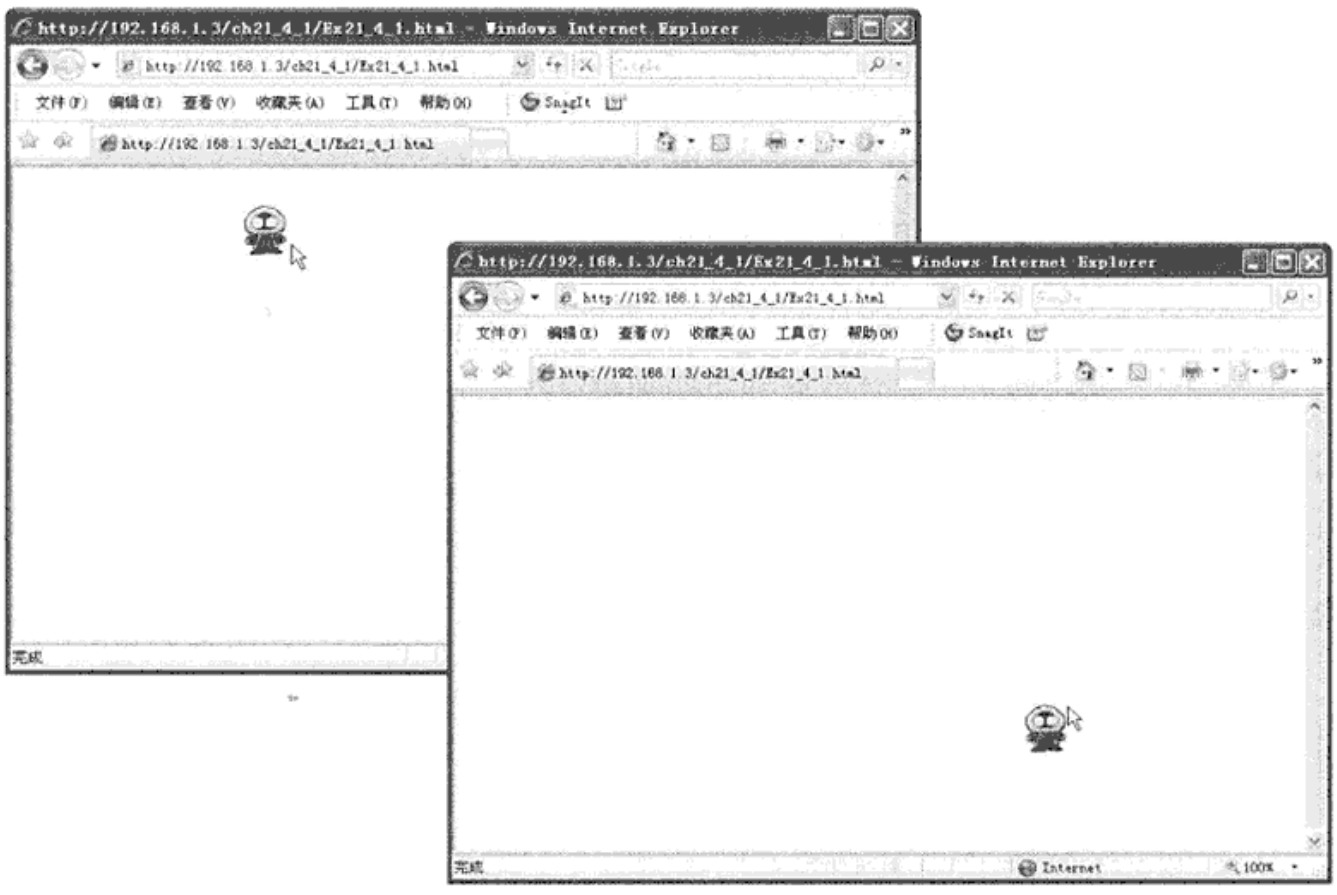


图 21-6

要再次强调的是，设计 Applet 网络浏览器程序时必须参考 21—2 节的线程工作流程，原因是利用它可同时运行多个图案的移动，多个图案并行各自的动作，互不干扰。

参考范例 39，设计范例 86，使用两组多幅图像，实现两个图像同时并行移动的功能。以鼠标单击第一个图像，拖动后释放鼠标；立即再单击第二个图像，拖动后释放鼠标。两个图像将同时并行，各自移动至预定的位置，互不干扰。

范例 86 设计文件 Ex21_4_2.java、Ex21_4_2.html，其功能是解释 Java Applet 在网络中运行鼠标事件与多个多幅图像并行的应用。

文件 Ex21_4_2.java：编写在记事本中，功能是解释鼠标事件与多个多幅图像并行的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.applet.*;

04 public class Ex21_4_2 extends Applet implements Runnable {
05    int x1=50, y1=50, fx1=0, fy1=0, dx1=0, dy1=0;
06    int flag1_type_img=0, flag1_num_img=0, num1=0;
07    Image img10, img11, img12;

08    int x2=100, y2=100, fx2=0, fy2=0, dx2=0, dy2=0;
09    int flag2_type_img=0, flag2_num_img=0, num2=0;
10    Image img20, img21, img22;

11    public void init() {
12       img10 = getImage(getDocumentBase(), "fly0.gif");
13       img11 = getImage(getDocumentBase(), "fly1.gif");
14       img12 = getImage(getDocumentBase(), "fly2.gif");
15       img20 = getImage(getDocumentBase(), "fly0.gif");
16       img21 = getImage(getDocumentBase(), "fly1.gif");
17       img22 = getImage(getDocumentBase(), "fly2.gif");
18       enableEvents(AWTEvent.MOUSE_EVENT_MASK);
19    }

20    public void processMouseEvent(MouseEvent e) {
21       if(e.getID() == MouseEvent.MOUSE_PRESSED)
22          if(((e.getX() >= x1) && (e.getX() <= (x1+80))) &&
23             ((e.getY() >= y1) && (e.getY() <= (y1+60))))   {
24                flag1_type_img = 1;
25          }
26          if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag1_type_img == 1)) {
27                fx1 = e.getX();
28                fy1 = e.getY();
29                dx1 = (fx1 - x1) / 50;
30                dy1 = (fy1 - y1) / 50;
31                flag1_type_img = 0;
32          }

33          if(((e.getX() >= x2) && (e.getX() <= (x2+80))) &&
34             ((e.getY() >= y2) && (e.getY() <= (y2+60))))   {
35                flag2_type_img = 1;
36          }
37          if((e.getID() == MouseEvent.MOUSE_RELEASED) && (flag2_type_img == 1)) {
38                fx2 = e.getX();
39                fy2 = e.getY();
40                dx2 = (fx2 - x2) / 50;
41                dy2 = (fy2 - y2) / 50;
42                flag2_type_img = 0;
43          }
44    }

45 public void start() {
46    new Thread(this).start();
47 }


48 public void run() {
49    while(true) {
50       x1 = x1 + dx1;
51       y1 = y1 + dy1;
```

```
52        flag1_num_img = num1 % 3;
53        if(((dx1 > 0) && ((x1+40) >= fx1)) || ((dx1 < 0) && ((x1+40) <= fx1))) dx1 = 0;
54        if(((dy1 > 0) && ((y1+30) >= fy1)) || ((dy1 < 0) && ((y1+30) <= fy1))) dy1 = 0;

55        x2 = x2 + dx2;
56        y2 = y2 + dy2;
57        flag2_num_img = num2 % 3;
58        if(((dx2 > 0) && ((x2+40) >= fx2)) || ((dx2 < 0) && ((x2+40) <= fx2))) dx2 = 0;
59        if(((dy2 > 0) && ((y2+30) >= fy2)) || ((dy2 < 0) && ((y2+30) <= fy2))) dy2 = 0;

60        repaint();
61        num1 = num1 + 1;
62        num2 = num2 + 1;
63        try{Thread.sleep(250);}
64        catch(InterruptedException e) {;}
65     }
66  }

67  public void paint(Graphics g) {
68     if(flag1_num_img == 0) g.drawImage(img10, x1, y1, this);
69     if(flag1_num_img == 1) g.drawImage(img11, x1, y1, this);
70     if(flag1_num_img == 2) g.drawImage(img12, x1, y1, this);

71     if(flag2_num_img == 0) g.drawImage(img20, x2, y2, this);
72     if(flag2_num_img == 1) g.drawImage(img21, x2, y2, this);
73     if(flag2_num_img == 2) g.drawImage(img22, x2, y2, this);
74  }
75 }
```

行 12～17　读取图像文件。
行 18　启动鼠标事件。
行 20　当有鼠标事件发生时执行。
行 21～24　设置图像单击范围（本例中图像为 80x60），如果在图像单击范围内单击鼠标左键，则设置 flag1_type_img 为 1。
行 26～28　拖动鼠标后，如果释放鼠标时 flag1_type_img 为 1，则读取鼠标指针当前的随动坐标(fx1，fy1)。
行 29～31　比较原始坐标(x1，y1）与随动坐标(fx1，fy1)，将它们的差除以 n（本例中为 50），用于设置(dx1，dy1)，同时将 flag1_type_img 还原成 0。
行 52　配合行 61 将第 1 组的 3 幅图像轮流显示。
行 53～54　当第一组图像随动至坐标(fx1,fy1)时，设置(dx,dy)=(0,0)，使图像不再移动。
行 60　以 repaint()方法驱动行 67～74 的 paint()方法，进行更新绘制。

编译文件 Ex21_4_2.java，生成 Ex21_4_2.class，配合 Ex21_4_2.html 运行，如图 21-7 所示。

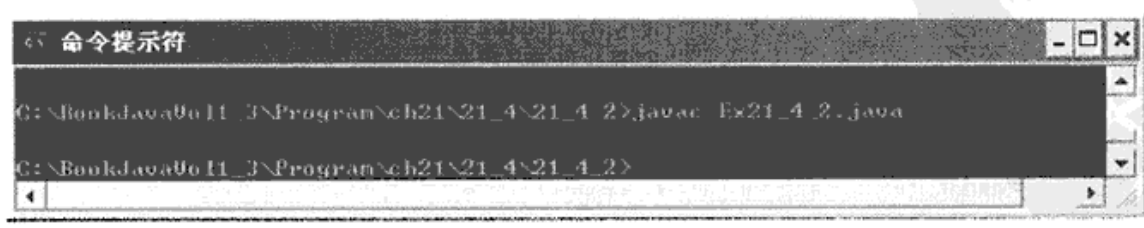


图 21-7

文件 Ex21_4_2.html：编写在记事本中，功能是使 Ex21_4_2.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex21_4_2" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1 参考 19-3 节运行，结果如图 21-8 所示。
本例中设置虚拟目录的别名为 ch21_4_2；网站内容目录为 C：\Java\Program\ch21\21_4\21_4_2。

2 本例中网站的网址为 http：//192.168.1.3/ch21_4_2/Ex21_4_2.html。

3 以鼠标单击第一组图像，拖动后释放鼠标；立即再单击第二组图像，拖动后释放鼠标。两组图像将同时并行，各自移动至预定的位置，互不干扰。

注意：以上为参考数据，读者应根据自己的计算机环境参考 19-3 节来操作。

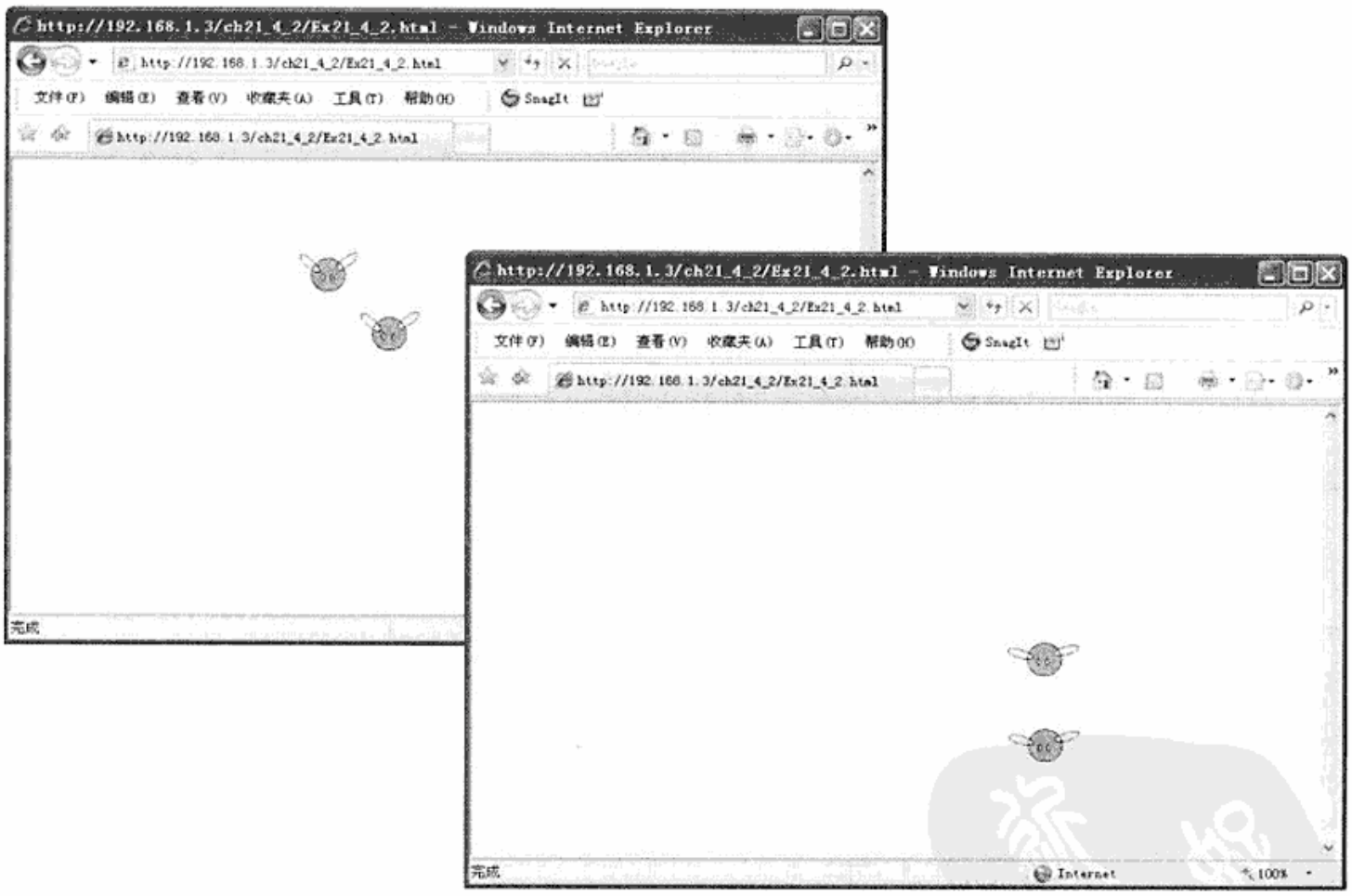


图 21-8

21-5 键盘事件

当按键盘键时，键盘事件随之发生，此时可读取按键的信息，不同的键有不同的 KeyCode，我们可利用这些信息来执行不同的操作。

参考范例 42，设计范例 87，运行多幅静态方向控制，当按“→”键时，图像向右移一步；当按“←”键时，图像向左移一步；当按“↑”键时，图像向上移一步；当按“↓”键时，图案向下移一步。

范例 87 设计文件 Ex21_5_1.java、Ex21_5_1.html，其功能是解释 Java Applet 在网络中运行键盘事件与多幅静态方向控制的应用。

文件 Ex21_5_1.java：编写在记事本中，功能是解释键盘事件与多幅静态方向控制的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.applet.*;

04 public class Ex21_5_1 extends Applet implements Runnable {
05     int x=150, y=235, dx=0, dy=0, flag_direction=0;
06     Image img0, img1, img2, img3;

07     public void init() {
08        img0 = getImage(getDocumentBase(), "img000.JPG");
09        img1 = getImage(getDocumentBase(), "img090.JPG");
10        img2 = getImage(getDocumentBase(), "img180.JPG");
11        img3 = getImage(getDocumentBase(), "img270.JPG");
12        enableEvents(AWTEvent.KEY_EVENT_MASK);
13     }

14     public void processKeyEvent(KeyEvent e) {
15         if(e.getID() == KeyEvent.KEY_PRESSED) {
16                 switch(e.getKeyCode()) {
17                     case KeyEvent.VK_RIGHT:
18                             dx = 5; dy = 0;
19                             flag_direction = 0;
20                             break;
21                     case KeyEvent.VK_LEFT:
22                             dx = -5; dy = 0;
23                             flag_direction = 2;
24                             break;
25                     case KeyEvent.VK_UP:
26                             dx = 0; dy = -5;
27                             flag_direction = 1;
28                             break;
29                     case KeyEvent.VK_DOWN:
30                             dx = 0; dy = 5;
31                             flag_direction = 3;
32                             break;
33                 }
34             x = x + dx;
35             y = y + dy;
36         }
37     }

38     public void start() {
39       new Thread(this).start();
40     }

41     public void run() {
42       while(true) {
43           repaint();
44           try{Thread.sleep(150);}
45           catch(InterruptedException e) {;}
46       }
47     }
```

```
48      public void paint(Graphics g) {
49          if(flag_direction == 0) g.drawImage(img0, x, y, this);
50          if(flag_direction == 1) g.drawImage(img1, x, y, this);
51          if(flag_direction == 2) g.drawImage(img2, x, y, this);
52          if(flag_direction == 3) g.drawImage(img3, x, y, this);
53      }
54 }
```

行 08～11	读取图像文件。
行 12	启动键盘事件。
行 14	当有键盘事件发生时立即运行行 15～37。
行 15	如果键盘事件是按键事件，则运行行 16～33。
行 16～33	当按“→”键时设置 dx = 5、dy = 0，使图像向右移一步，设置标示 flag_direction 为 0； 当按“←”键时设置 dx = −5、dy = 0，使图像向左移一步，设置标示 flag_direction 为 2； 当按“↑”键时设置 dx = 0、dy = −5，使图像向上移一步，设置标示 flag_direction 为 1； 当按“↓”键时设置 dx = 0、dy = 5，使图像向下移一步，设置标示 flag_direction 为 3。
行 34～35	设置图像新位置坐标。
行 39	以线程驱动行 41～47 的 run()方法。
行 43	以 repaint()方法驱动行 48～53 的 paint()方法，进行更新绘制。
行 49	如果标示 flag_direction 为 0，则显示第一幅图。

编译文件 Ex21_5_1.java，生成 Ex21_5_1.class，配合 Ex21_5_1.html 运行，如图 21-9 所示。

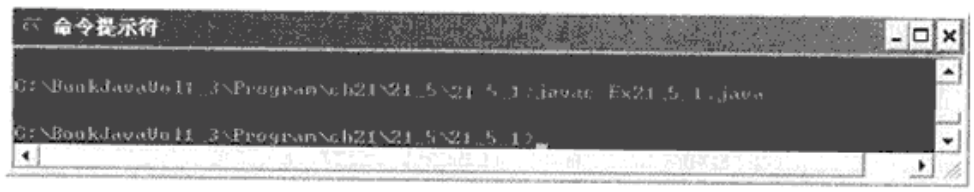


图 21-9

文件 Ex21_5_1.html：编写在记事本中，功能是使 Ex21_5_1.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex21_5_1" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1　参考 19-3 节运行，结果如图 21-10 所示。

本例中设置虚拟目录的别名为 ch21_5_1；网站内容目录为 C：\Java\Program\ch21\21_5\21_5_1。

2 本例中网站的网址为 http：//192.168.1.3/ch21_5_1/Ex21_5_1.html。

3 在显示区单击鼠标左键启动键盘事件，当按“→”键时，图像向右移一步；当按“←”键时，图像向左移一步；当按“↑”键时，图像向上移一步；当按“↓”键时，图像向下移一步。

注意事项

(1) 运行 Applet 网络浏览器的事件时，需在显示区单击鼠标左键启动键盘事件。

(2) 以上为参考数据，读者应根据自己的计算机环境参考 19-3 节来操作。

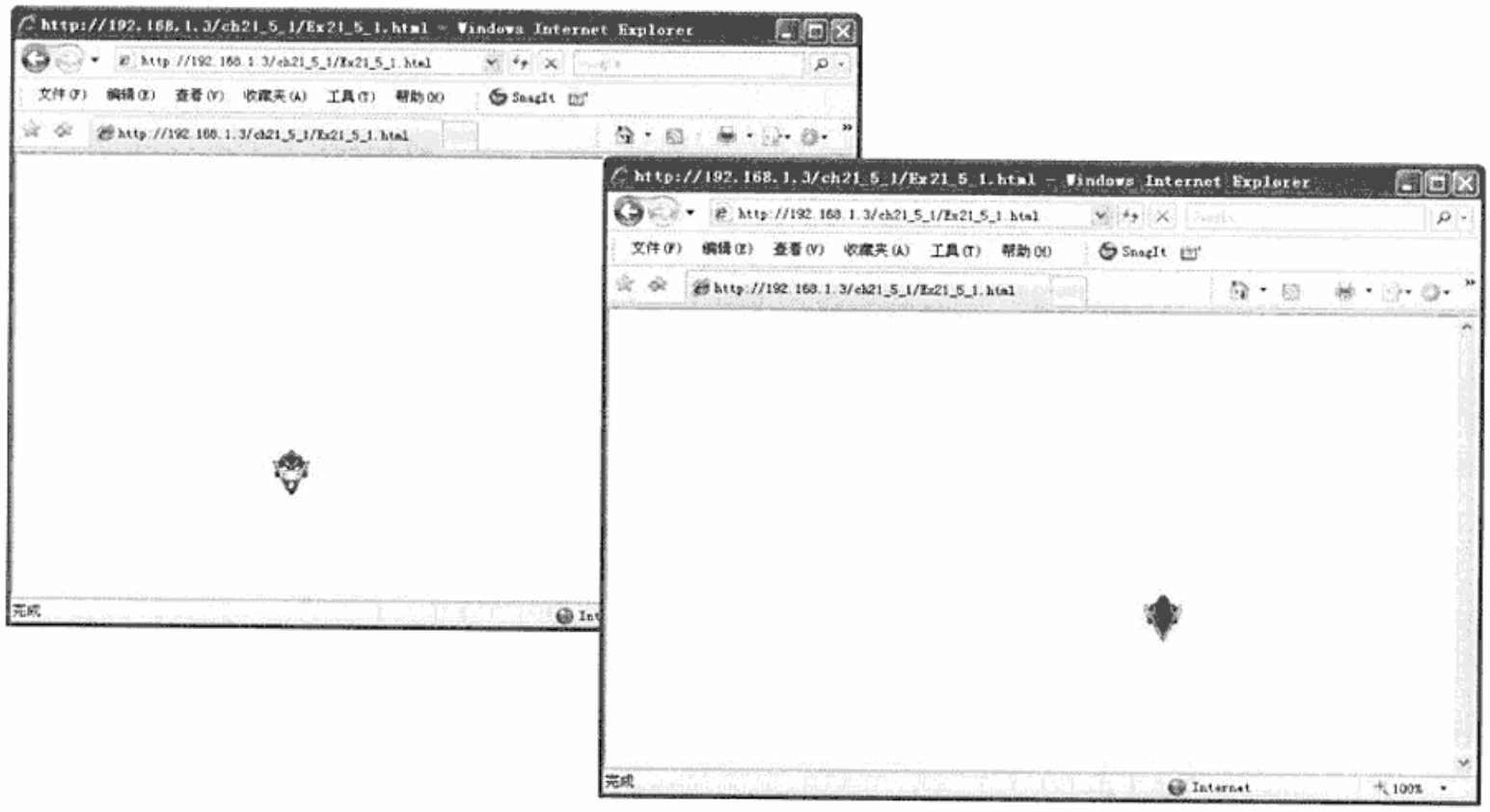


图 21-10

参考范例 44，设计范例 88，运行多幅动态方向控制，当按“→”键时，图像向右连续移动；当按“←”键时，图像向左连续移动；当按“↑”键时，图像向上连续移动；当按“↓”键时，图像向下连续移动。

所谓动态，是指通过循环使图像的位置坐标不停地变换，每循环一次，即更新图像坐标一次，图像也随着不停地移动。

范例 88 设计文件 Ex21_5_2.java、Ex21_5_2.html，其功能是解释 Java Applet 在网络中运行键盘事件与多幅动态方向控制的应用。

文件 Ex21_5_2.java：编写在记事本中，功能是解释键盘事件与多幅动态方向控制的应用。

```
01 import java.awt.*;
02 import java.awt.event.*;
03 import java.applet.*;

04 public class Ex21_5_2 extends Applet implements Runnable {
05      int x=0, y=170, dx=0, dy=0;
06      int num=1, flag;
07      Image img0, img1, img2;
```

```
08    public void init() {
09       img0 = getImage(getDocumentBase(), "fly0.gif");
10       img1 = getImage(getDocumentBase(), "fly1.gif");
11       img2 = getImage(getDocumentBase(), "fly2.gif");

12       enableEvents(AWTEvent.KEY_EVENT_MASK);
13    }

14    public void processKeyEvent(KeyEvent e) {
15          if(e.getID() == KeyEvent.KEY_PRESSED) {
16                switch(e.getKeyCode()) {
17                      case KeyEvent.VK_RIGHT:
18                            dx = 5; dy = 0;
19                            break;
20                      case KeyEvent.VK_LEFT:
21                            dx = -5; dy = 0;
22                            break;
23                      case KeyEvent.VK_UP:
24                            dx = 0; dy = -5;
25                            break;
26                      case KeyEvent.VK_DOWN:
27                            dx = 0; dy = 5;
28                            break;
29                }
30          }
31    }

32    public void start() {
33       new Thread(this).start();
34    }

35    public void run() {
36       while(true) {
37          x = x + dx;
38          y = y + dy;
39          flag = num % 3;
40          repaint();
41          num = num + 1;

42          if(x <= 0) dx = 5;
43          else if((x+60) >= 350) dx = -5;
44          if((y-10) <= 0) dy = 5;
45          else if((y+50) >= 350) dy = -5;

46          try{Thread.sleep(170);}
47          catch(InterruptedException e) {;}
48       }
49    }

50    public void paint(Graphics g) {
51       if(flag == 0)   g.drawImage(img0, x, y, this);
52       if(flag == 1)   g.drawImage(img1, x, y, this);
53       if(flag == 2)   g.drawImage(img2, x, y, this);
54    }
55 }
```

行 37～38　使图像连续移动。
行 39　配合行 41 使三幅图像轮流显示。

编译文件 Ex21_5_2.java，生成 Ex21_5_2.class，配合 Ex21_5_2.html 运行，如图 21-11 所示。

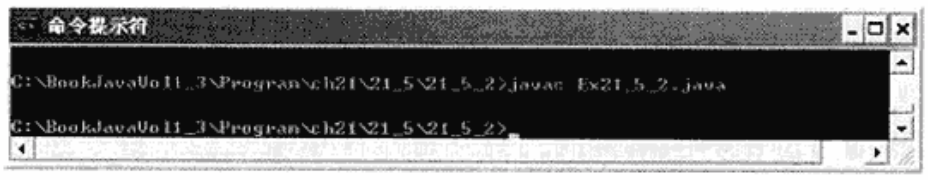


图 21-11

文件 Ex21_5_2.html：编写在记事本中，功能是使 Ex21_5_2.class 能在网络中运行。

```
<HTML>
<BODY BGCOLOR="#eeeeee">
<CENTER>
<APPLET CODE="Ex21_5_2" WIDTH=350 HEIGHT=350>
</APPLET>
</HTML>
```

参考范例 79 中文件 Ex19_2.html 的解说。

网络浏览器运行

1 参考 19—3 节运行，结果如图 21—12 所示。
本例中设置虚拟目录的别名为 ch21_5_2；网站内容目录为 C：\Java\Program\ch21\21_5\21_5_2。

2 本例中网站的网址为 http：//192.168.1.3/ch21_5_2/Ex21_5_2.html。

3 在显示区单击鼠标左键启动键盘事件，当按“→”键时，图像向右连续移动；当按“←”键时，图像向左连续移动；当按“↑”键时，图像向上连续移动；当按“↓”键时，图像向下连续移动。

图 21-12

注意事项

(1) 运行 Applet 网络浏览器的事件时，需在显示区单击鼠标左键启动键盘事件。

(2) 以上为参考数据，读者应根据自己的计算机环境参考 19-3 节来操作。

21-6 习题

1. Applet 网络浏览器环境的设计流程是怎样的？
2. 运行 Applet 网络浏览器的键盘事件时，要如何启动键盘事件？
3. 参考 18-5 节的范例 78，使用 Applet 网络浏览器环境设计立方体光影变化。

Appendix A 键盘事件类常量

```
public static final int KEY_FIRST, KEY_LAST, KEY_TYPE;
public static final int KEY_PRESSED, KEY_RELEASED;
public static final int VK_ENTER;
public static final int VK_BACK_SPACE;
public static final int VK_TAB;
public static final int VK_CANCEL;
public static final int VK_CLEAR;
public static final int VK_SHIFT, VK_CONTROL, VK_ALT;
public static final int VK_PAUSE;
public static final int VK_CAPS_LOCK;
public static final int VK_ESCAPE;
public static final int VK_SPACE;
public static final int VK_PAGE_UP, VK_PAGE_DOWN;
public static final int VK_END;
public static final int VK_HOME;
public static final int VK_LEFT, VK_RIGHT;
public static final int VK_UP, VK_DOWN;
public static final int VK_COMMA;
public static final int VK_MINUS;
public static final int VK_PERIOD;
public static final int VK_SLASH;
public static final int VK_0, VK_1, VK_2, VK_3, VK_4, VK_5, VK_6, VK_7, VK_8, VK_9;
public static final int VK_SEMICOLON;
public static final int VK_EQUALS;
public static final int VK_A, VK_B, VK_C, VK_D, VK_E, VK_F, VK_G, VK_H, VK_I, VK_J, VK_K,
                        VK_L, VK_M, VK_N, VK_O, VK_P, VK_Q, VK_R, VK_S, VK_T, VK_U,
                        VK_V, VK_W, VK_X, VK_Y, VK_Z;
public static final int VK_OPEN_BRACKET;
public static final int VK_BACK_SLASH;
public static final int VK_CLOSE_BRACKET;
public static final int VK_NUMPAD0, VK_NUMPAD1, VK_NUMPAD2, VK_NUMPAD3, VK_NUMPAD4,
                        VK_NUMPAD5, VK_NUMPAD6, VK_NUMPAD7, VK_NUMPAD8, VK_NUMPAD9;
public static final int VK_MULTIPLY;
public static final int VK_ADD;
public static final int VK_SEPARATER;
public static final int VK_SUBTRACT;
public static final int VK_DECIMAL;
```

```
public static final int VK_DIVIDE;
public static final int VK_DELETE;
public static final int VK_NUM_LOCK, VK_SCROLL_LOCK;
public static final int VK_F1, VK_F2, VK_F3, VK_F4, VK_F5, VK_F6, VK_F7, VK_F8, VK_F9,
                         VK_F10, VK_F11, VK_F12;
public static final int VK_PRINTSCREEN;
public static final int VK_INSERT;
public static final int VK_HELP;
public static final int VK_META;
public static final int VK_BACK_QUOTE, VK_QUOTE;
public static final int VK_KP_UP, VK_KP_DOWN;
public static final int VK_KP_RIGHT, VK_KP_LEFT;
public static final int VK_AMPERSAND;
public static final int VK_ASTERISK;
public static final int VK_QUOTEDBL;
public static final int VK_LESS;
public static final int VK_GREATER;
public static final int VK_BRACELEFT;
public static final int VK_BRACERIGHT;
public static final int VK_AT;
public static final int VK_COLON;
public static final int VK_CIRCUNFLER;
public static final int VK_DOLLAR;
public static final int VK_EURO_SIGN;
public static final int VK_EXCLANATION_MARK;
public static final int VK_INVERTED_EXCLANATION_MARK;
public static final int VK_NUMBER_SIGN;
public static final int VK_PLUS;
public static final int VK_RIGHT_PARENTHESIS;
public static final int VK_UNDERSCORE;
public static final int VK_FINAL;
public static final int VK_CONVERT;
public static final int VK_NONCONVRT;
public static final int VK_ACCEPT;
public static final int VK_MODECHANGE;
public static final int VK_CUT;
public static final int VK_COPY;
public static final int VK_PASTE;
public static final int VK_UNDO;
public static final int VK_AGAIN;
```

```
public static final int VK_FIND;
public static final int VK_PROPS;
public static final int VK_STOP;
public static final int VK_COMPOSE;
public static final int VK_ALT_GRAPH;
public static final int VK_UNDEFINED;
```