新概念中文 Flash MX 教程

成　昊　编著

北京科海电子出版社

读者对象：

- ☑ 爱好动画制作的初、中级用户
- ☑ 各类计算机培训班

达成目标：

- ☑ 掌握制作 Flash 动画的整个流程
- ☑ 学会各种动画的制作方法
- ☑ 熟练运用脚本制作动画

本书特点：

- ☑ 融入作者多年教学经验
- ☑ 理论讲解与实例制作完美结合
- ☑ 提供大量实际应用的经验与技巧
- ☑ 语言精炼、讲解透彻

内容提要

这是由资深培训讲师根据多年教学经验编写的中文版 Flash MX 动画制作教程，全书的 11 章内容环环相扣，讲解由浅入深，循序渐进，真正实现了理论讲解与实例制作的完美结合。

书中首先介绍 Flash MX 的应用及基本操作，然后在各章中分专题介绍动画脚本制作、多媒体对象制作、图像动画制作、文字动画制作、组件应用、游戏设计等内容，最后还介绍了如何导出并发布 Flash 作品。讲解过程中，不仅用实例展示实战经验和技巧，还深入分析问题产生的根源，提示读者如何规避错误。

本书是一本经过教学实践检验的 Flash 动画制作培训教材，也适合爱好动画制作的初、中级用户自学，对高级设计人员也有一定的参考价值。

光盘内容：书中实例的多媒体演示、全部范例动画的 FLA 源文件、SWF 动画文件以及相关素材。另外，还提供了供授课用的 PowerPoint 幻灯片。

品　　名：新概念中文 Flash MX 教程
作　　者：成　昊
责任编辑：潘秀燕
排　　版：芳　芳
出　　品：北京科海电子出版社
印 刷 者：北京耀华印刷有限公司
发 行 者：新华书店总店北京发行所
开　　本：787×1092 1/16　印张：16.75　字数：407 千字
版　　次：2003 年 5 月第 1 版　2003 年 5 月第 1 次印刷
印　　数：0001~5000
盘　　号：ISBN 7-900372-31-8
定　　价：26.00 元（1CD/配套手册）

前　言

因特网飞速发展的今天，形式各异的网站层出不穷，在众多个人主页中崭露头角是每个网络爱好者追求的梦想。Macromedia 公司推出的矢量化动画制作软件 Flash MX 可以助您实现这个梦想，它已成为大多数动画爱好者设计和制作动画的首选工具。

网络中流传着缤纷多彩的 Flash 动画，如精彩 MV、动画短剧、交互课件、益智游戏等，这些都是 Flash 技术应用的最佳体现。当然，这些动画也融入制作者的创意与构思，正是这些创意与构思更进一步地推动了 Flash 动画的盛行。

由于 Flash 的流行，图书市场上出现了大量与 Flash 相关的书籍，其质量也是良莠不齐，使读者在挑选适合自己学习的图书时眼花缭乱、不知取舍。纵观整个图书市场，所有 Flash 书籍主要分为两类，其一是以理论知识为主，缺乏足够的实践内容，读之犹如空中建楼、纸上谈兵；其二就是以实例堆砌而成，缺乏必要的基础知识，观之往往一头雾水、不知所云。

基于市场需求的有效分析，作为具有多年教学经验的培训讲师，我编写了本书，将理论与实践有机地融为一体，相辅相成、取长补短。本书在介绍软件用法的同时提供了大量实际应用的经验和技巧，提示您应该规避哪些错误，并深入分析问题产生的根本原因，使您在掌握 Flash 操作的同时，能够全面系统地理解 Flash。

本书导读

读者定位	学习建议
希望系统学习 Flash，完成一般任务	循序渐进、系统完整地学习 Flash，您应该重点学习第 1 章至第 5 章和第 11 章。通过这些章节的学习，可以在较短的时间内掌握制作 Flash 动画的整个流程。 在第 1 章中可以了解 Flash 动画的特点和应用领域。 在第 2 章中可以掌握 Flash 中工具的使用并绘制矢量图形。 在第 3 章中可以掌握 Flash 各类动画制作的基本步骤。 在第 4 章中可以掌握 Flash 中动画元件和库面板的使用。 在第 5 章中可以掌握 Flash 中基本的动画脚本。 在第 11 章中可以掌握 Flash 动画的发布和导出。

读者定位	学习建议
希望熟练使用 Flash，提升专业能力	除了讲述以上基础知识，其余章节针对 Flash 中的各种对象或动画进行分类，使您能够比较清楚地了解各种形式的动画的制作方法。 在第 6 章中可以掌握多媒体对象的使用。 在第 7 章中可以掌握图像的使用和相关动画的制作。 在第 8 章中可以掌握文字的使用和相关动画的制作。 在第 9 章中可以掌握组件的使用和制作。 在第 10 章中可以了解游戏的制作。

读者对象

本书结合理论和实例介绍 Flash 的使用技巧和方法，整体结构非常适合培训教师讲课，是不可多得的电脑培训教程。

本书语言精练，内容由浅入深、循序渐进，从最基本的工具使用到各类动画的制作，再深入到 Flash 动画脚本的编写，力求做到讲解清楚、易学易用，适合于从事网页动画设计和制作的初、中级人员学习使用，同时对高级动画设计人员也有一定的借鉴作用。

光盘说明

光盘中提供了全部范例动画的 FLA 源文件和 SWF 动画文件，对每个范例动画进行多媒体演示，手把手地教您制作。光盘中还包含了范例动画中涉及的图片素材、声音文件和视频文件，方便您按照书中的范例进行模仿制作。另外，光盘中还提供了教师授课用的 PowerPoint 幻灯片。

编　者

2003 年 4 月

目　录

第 1 章　Flash MX 概述

Flash 动画的出现，为网络世界注入了勃勃生机。在网页动画的设计和制作中，Flash 无疑是其中的佼佼者。Flash MX 可以将矢量图形的精确性、灵活性与位图、声音、动画、视频交互应用，完美地融为一体，创作出极具吸引力的、动态的、高效的、交互的 Web 页面动画。

本章主要介绍了 Flash 动画的特点和功能，并对 Flash 动画的应用领域进行了简单的分类。

通过本章的学习，应该达到如下目的：

- 认识 Flash 动画的特点
- 介绍 Flash 的强大功能
- 了解 Flash 的应用领域

1.1　动 画 特 点

随着网络技术的不断成熟，网络宽带逐渐深入家庭用户，Web 页面上不再只是单纯的文字内容、精美图片，而且这些也不能再满足广大用户的需求。1995 年，Sun 公司开发了 Java 程序，并且采用既定的 Applet 程序在网页上设计出动态的多媒体程序，一时风靡全球。但是，由于 Java 程序的专业性强，使得众多网页设计者深感力不从心，对于 Java 制作的动态页面只能是望而却步。

著名的多媒体制作公司 Macromedia 在 98 年推出 Flash 版本 3.0 时，就引起了强烈的反响，广大页面设计人员真正体会到了网页动画创作的简便；随之又连续推出了 Flash 的 4.0 以及 5.0 版本，使得 Flash 动画创作功能越来越强大，同时也由单纯的动画设计开始逐步分为动画设计（Design）、应用开发（Develop）两部分；在 2002 年 3 月中旬，Macromedia 公司又强力推出了最新版本的 Flash MX，整个动画创作环境使广大动画设计人员感觉耳目一新的同时，也深深体会到 Macromedia 公司的良苦用心——动画创作的简单易行。Flash MX 不仅是在创作环境上给人以新鲜的感觉，同时更增强了众多功能，尤其是应用于开发的脚本（ActionScript）。

Flash 自其推出以来，就一直受到广大动画爱好者的热烈支持，究其原因不仅仅是在动画创作时的简单易行，同时 Flash 还具有以下几点重要的因素：

1.1.1　图形矢量化

计算机图形主要分为两大类——位图图像和矢量图形。位图图像（技术上称为栅格图像或点阵图像）使用颜色网格（也就是常说的像素）来表现图像。每个像素都有自己特定的位置和颜色值。例如，一幅位图图像中的自行车轮胎就是由该位置的像素拼合在一起组

成的。当您在处理位图图像时，所编辑的是像素而不是对象或形状。

位图图像是连续色调图像（如照片或数字绘画）最常用的电子媒介，因为它们可以表现阴影和颜色的细微层次。位图图像与分辨率有关，也就是说，它们包含固定数目的像素。因此，如果对位图图像进行缩放操作，就会丢失位图图像中的细节，呈现出锯齿状，如图 1.1 所示。

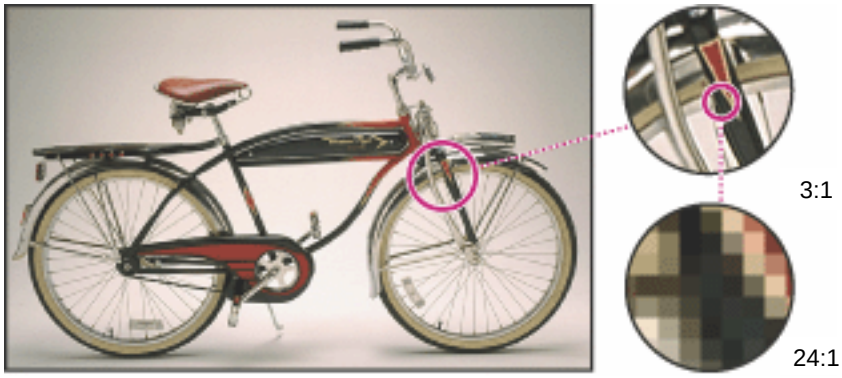

图 1.1　不同放大级别的位图图像

矢量图形由被称为矢量的数学对象定义的线条和曲线组成，根据图像的几何特性描绘图像。例如，一幅矢量图形中的自行车轮胎是由一个圆的数学定义组成的，这个圆按某一半径绘制，放置在特定的位置处并填充特定的颜色。移动轮胎、调整尺寸或更改颜色时都不会降低矢量图形的品质。

矢量图形与分辨率无关，也就是说可以将其缩放为任意尺寸而不会丢失其细节部分或降低清晰度，如图 1.2 所示。

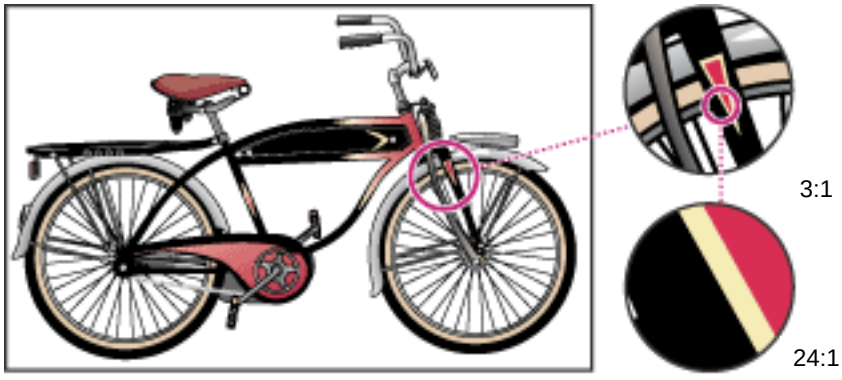

图 1.2　不同放大级别的矢量图形

Flash 动画中的图形对象就是基于矢量方式进行存储的，可以将动画无极缩放而不会影响画面质量，而且动画文件所占的容量较小。当然，在 Flash 中也同样支持位图图像，并且能对置入的位图图像进行优化以减小动画文件的容量；或者直接将位图图像转换为矢量图形，使其既具有矢量图形的精确和灵活，又表现出位图图像的精美和细腻。

1.1.2　浏览流式化

早期经常在网上冲浪的用户通常只是捕捉各大站点中的文本页面，其原因主要是实在忍受不了浏览华丽页面时痛苦等待的煎熬。由于只有当页面中的所有内容都完全下载完成后，才能一窥其华丽的面貌，致使很多用户无奈地放弃这些站点的浏览。

而今，随着网络技术的成熟，宽带使用的推广，这种状况有所好转，但是浏览这些充满位图图像的华丽页面时还是免不了一定时间的等待。如果您曾有在线观摩电影的经历，可知通常在点播后短时间内就可以观看。众所周知，一部电影的容量是相当庞大的，在这么短暂的时间内是不可能将整个电影全部下载完成的，那为什么会有如此效果呢？实际上在线观摩电影与电影下载是同时发生的。这就是流式技术使用的典型例子。

随着 Flash 动画制作技术的成熟和大量 Flash 动画设计团队的创建，Flash 动画开始逐渐具备故事情节、播放时间越来越长、动画画面也越来越美。但是在浏览这些动画时，基本感觉不到动画在浏览的同时也在下载。这正是由于 Flash 动画也应用了流式技术，但是为了避免动画浏览过程中“跳帧”现象的发生，通常在制作大型动画时会制作下载进度提示条，以告知浏览用户所需的等待时间。

1.1.3　控制交互化

Flash 动画最大的特点就是具备强大的交互控制功能。浏览动画时，用户可以参与到动画的播放过程之中，如单击 Play 按钮可以观看动画播放，如图 1.3 所示，或者单击 Replay 按钮再次浏览动画。当然，Flash 动画的交互并不局限于此，需要在以后的学习过程中逐渐领悟！

图 1.3　动画中的 Play 按钮

1.1.4　技术共享化

虽然 Flash 动画最初是为应用于 Web 页面而产生的，但随后并非只应用于 Web 页面，有时还需要将 Flash 动画植入到其他应用程序中作为素材使用，如多媒体制作 Director 等，甚至是编程软件 VB、VC 中。

基于上述情况，Flash 提供了相应的 FS Command 语句，使装载 Flash 动画的应用程序能够与 Flash 动画进行信息交换。图 1.4 所展示的就是植入 Flash 动画且由 VB 编写生成的应用程序，其中有动画播放控制按钮和视图控制按钮等。

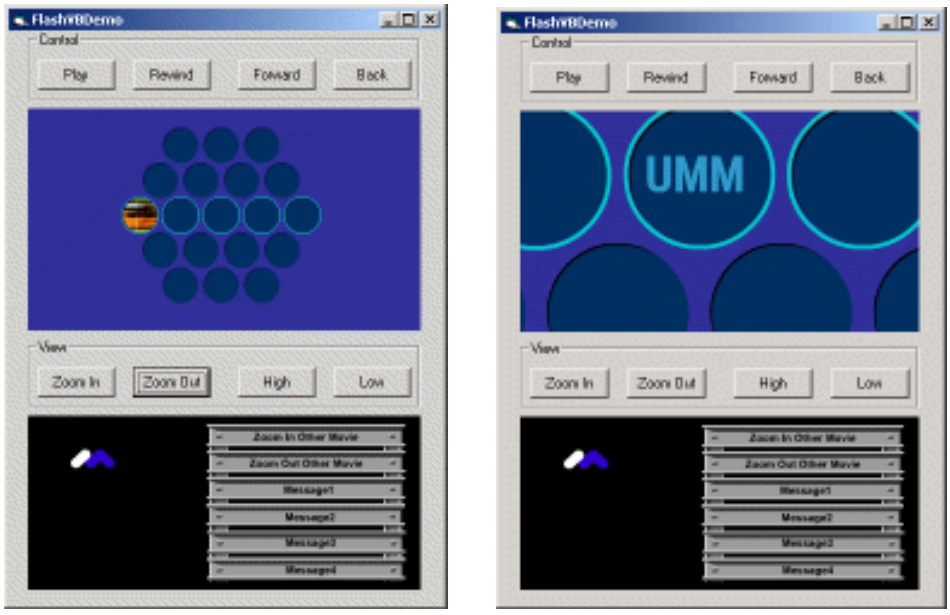

图 1.4 Flash 动画在 VB 编程中的应用

Flash 动画播放器也可由 FS Command 进行控制，主要控制 Flash 动画播放器的窗口及其显示，例如全屏显示（fullscreen）、缩放控制（allowscale）、显示菜单（showmenu）和退出动画（quit）等。

虽然 Flash 动画具有上述众多的优点，但是也存在最致命的弱点——浏览用户计算机的网页浏览器中必须安装 Flash 相关插件。如果您没有安装这类插件，那么在您浏览嵌套有 Flash 动画的 Web 页面时就会提示用户下载并安装该插件，否则就无法观看到 Web 页面嵌套的 Flash 动画。

1.2 功 能 介 绍

Flash MX 版本增强了 Flash 的易用性、创造性和功能性。以创造性为基础的强大功能给动画设计人员提供了一个更加可靠、更加简便且令人振奋的创作环境；而加强后的高级脚本、调试工具和预设组件使得开发人员能够迅速部署丰富多彩的 Web 应用程序！

1.2.1 界面简洁明了

Flash MX 创作界面得到很大程度的改变，主要是由于采用了可以根据选择内容的不同而智能调整的“属性”面板，通常位于整个界面的中下方，如图 1.5 所示。“属性”面板中包含了动画创建过程中大量的对象属性，根据制作人员选择对象的不同来调整相应对象属性的显示，从而消除了以往所有版本中大量窗口、面板和对话框的调用。

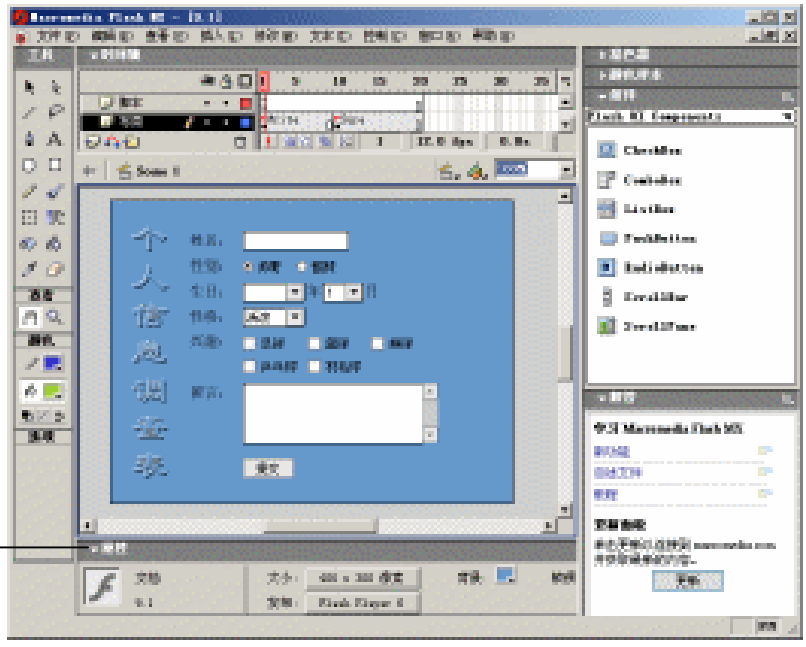

图 1.5　Flash MX 界面

而其他频繁使用的面板主要集中显示于右侧的折叠式面板区中，这些面板或固定在 Flash MX 应用程序窗体的右侧、或浮动在 Flash MX 制作环境中。动画制作人员可以根据习惯对全部面板进行合理地分布，使得动画的创作更加得心应手。

1.2.2　视频影像支持

Flash MX 对视频影像的支持使得 Flash 动画的扩展成为可能。但是导入视频文件时，如果需要视频画面质量较好，可能导致动画文件的大小成倍增加。

常用的视频影像文件格式在 Flash MX 中都得到较好的支持，可以导入视频片断以丰富动画的表现效果，图 1.6 就粗略地显示了 Flash MX 对视频文件的支持。

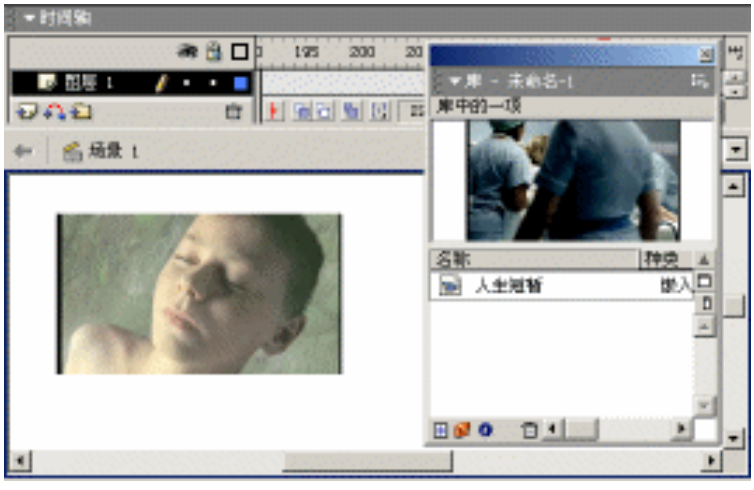

图 1.6　视频影像的使用

1.2.3 开发工具增强

Flash MX 中的“动作”面板更加完善、更加强大，如图 1.7 所示，这使得 Flash 动画设计人员（无论是新手还是老将）体会到 Flash 脚本的强大功能。

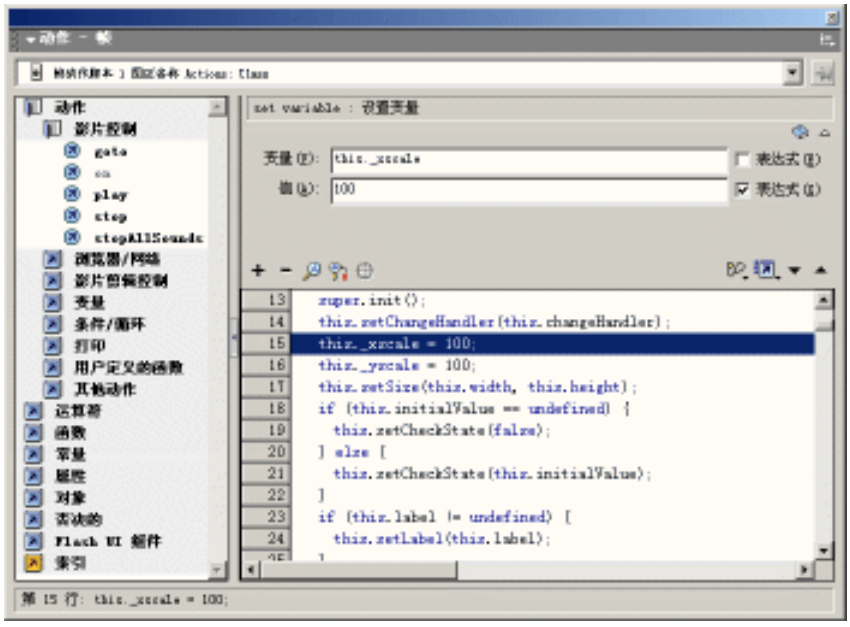

图 1.7 “动作”面板

图形编程接口中提供了一系列关于影片剪辑实例的图形绘制功能，增强了 Flash 脚本面向对象的编程理念，允许开发人员采用程序绘制 Flash 动画中的矢量图形。

增强的脚本功能允许开发人员在 Flash 动画中动态载入位图图像、MP3 声音等。修改时就可以不必重新进行动画发布，只要将相应的位图图像、MP3 声音等文件进行替换即可，可谓又方便又快捷。

Flash 中的脚本采用事件模式，使得整个脚本代码更加简单、易懂。事件模式可以进行更高级别的使用，以响应浏览用户的不同操作（如鼠标的移动和按键等）。

1.2.4 图形任意变形

任意变形工具可以将设计人员的创造性思维发挥得淋漓尽致，使得简简单单的对象变化出各种形式的图形，大大减轻了动画设计人员求助矢量图形处理软件的麻烦。Flash MX 提供的任意变形工具有旋转和倾斜、缩放、扭曲以及封套等功能，其中扭曲和封套功能是专门针对 Flash 动画中的矢量图形的。图 1.8 就是任意变形工具对矢量图形进行变形调整的示例。

1.2.5 保持向下兼容

Flash MX 保持向下兼容的功能，可以将 Flash MX 文档存储为 Flash 5 的文档格式。只要使用“文件”|“另存为”命令，就可以将 Flash MX 创建的动画文档存储为 Flash 5 的文

件格式。这样，就可以实现 Flash MX 与 Flash 5 两个应用程序之间的文件共享，确保与 Flash 5 开发人员共同完成动画制作的项目。

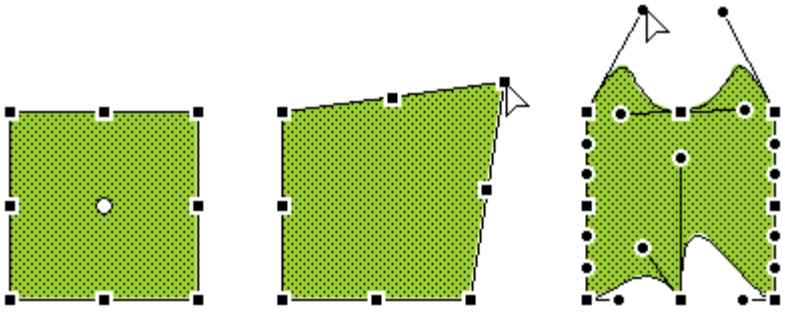

图 1.8　矢量图形变形示例

1.2.6　完善的时间轴

如果动画中涉及大量的动画元素，迫使时间轴中的图层大量增加，此时图层的管理就会相当烦琐。Flash MX 提供的图层文件夹可以将相关的多个图层分类置于多个不同的文件夹中，使得整个时间轴中的图层合理分布。将图层文件夹折叠可减少屏幕占有空间，将图层文件夹展开可以查看其中的图层内容，如图 1.9 所示。

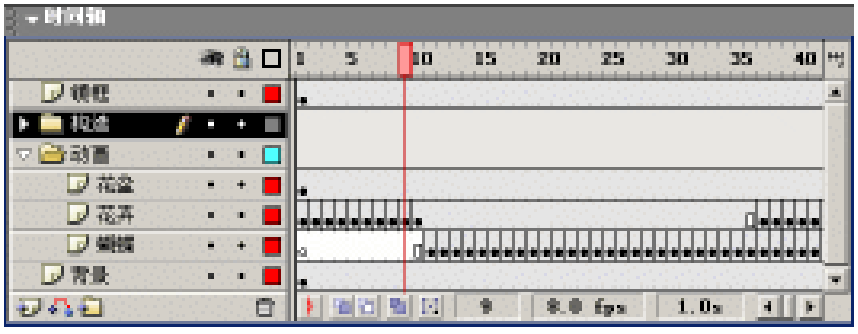

图 1.9　时间轴内文件夹

1.2.7　“混色器”面板

“混色器”面板的功能非常完善，不再是以往版本中的纯色，而且可以直接进行线性渐变或者放射渐变填充的创建、编辑与应用，同时还有更加精美的位图填充，如图 1.10 所示。

1.2.8　遮罩功能增强

Flash MX 中的遮罩功能更加强大，以往版本的 Flash 中如果在遮罩层中放置影片剪辑，只会看到影片剪辑第 1 帧的静态遮罩效果，而不会看到预想的遮罩动画效果，并且也不支持脚本编写对遮罩层中的影片剪辑实例的控制。

但是，所有这些问题都在 Flash MX 中得到完全彻底的解决，而且还公布了利用脚本

编写实现两个影片剪辑实例的遮罩效果。

图 1.10 “混色器”面板

1.2.9 像素级别网格

如果要精确控制动画中的对象，可以在“信息”面板中调整对象的尺寸和坐标，还可以根据 Flash MX 提供的像素级别网格实现各种对象的精确控制。

选中“视图”|“对齐像素”命令后，将缩放比率调整到大于 400%，此时就可以观看到像素级别的网格，如图 1.11 所示，方便设计人员绘制图形时的尺寸控制和位置分布。

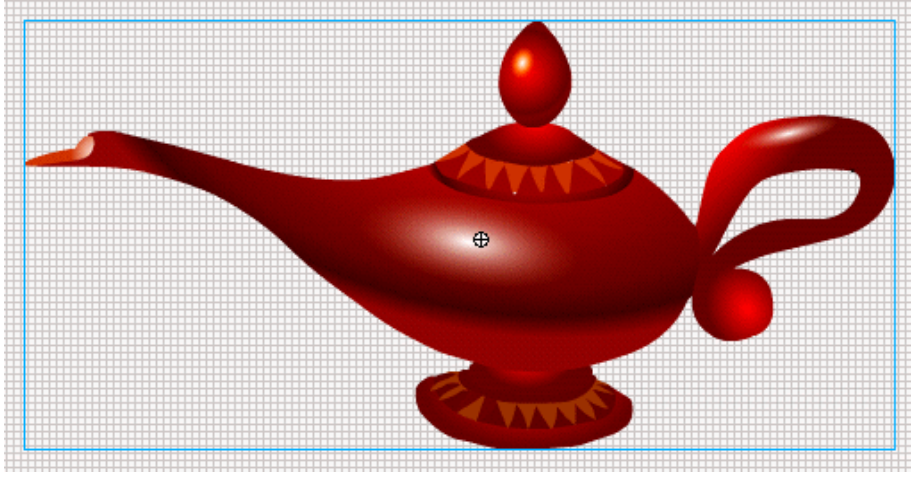

图 1.11 像素级网格

1.2.10 完善的“库”面板

Flash MX 中的“库”面板功能更加完善，消除了单一文档中库项目创建和处理的瓶颈，允许在多个动画文档间利用简单的拖曳方法，直接移动库项目或文件夹以创建文档的“库”面板。当动画文档的现有“库”面板中存在相同名称文件夹时，将会弹出“解决库冲突”

对话框，如图 1.12 所示，要求设计人员根据需要做出相应的处理。

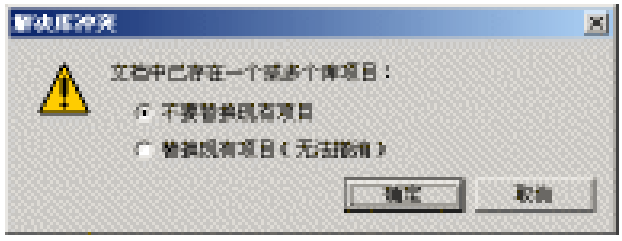

图 1.12　“ 解决库冲突 ” 对话框

1.2.11　模板功能

Flash MX 提供了动画模板，消除了新建文档时经常操作的相同的多个步骤，使得新建文档更加简单。当然也可以根据个人习惯，创建具有个性化的动画模板。

新建文档时，选择 “ 文件 ” | “ 从模板新建 ” 命令，就可以弹出 “ 新文档 ” 对话框，如图 1.13 所示，选择其中的 “ 类别 ” 和 “ 类别项目 ” 即可。

图 1.13　“ 新文档 ” 对话框

如果您要创建自己的模板，可以创建一个 Flash 动画文档后，选择 “ 文件 ” | “ 另存为模板 ” 命令，在弹出 “ 另存为模板 ” 对话框中输入 “ 名称 ” 等内容，如图 1.14 所示，单击 “ 保存 ” 按钮即可将当前文档存储为模板。

1.2.12　预设组件

对于开发人员而言，Flash MX 提供了功能强大的脚本编写和测试工具，内置 “ 脚本参考 ” 面板以及预设 Flash UI Components 的 “ 组件 ” 面板，从而可快速开发丰富的 Web 应用程序。

Flash MX 中提供的大量的预设组件是开发人员所频繁应用的，只要将这些组件拖放到

动画文档中进行参数设置即可，从而缩短 Web 应用程序的开发时间，达到提高工作效率的目的。选择“窗口”|“组件”命令，可以显示“组件”面板，如图 1.15 所示，其中提供了复选框（CheckBox）、下拉菜单（ComboBox）、列表框（ListBox）、单选框（RadioButton）等组件。

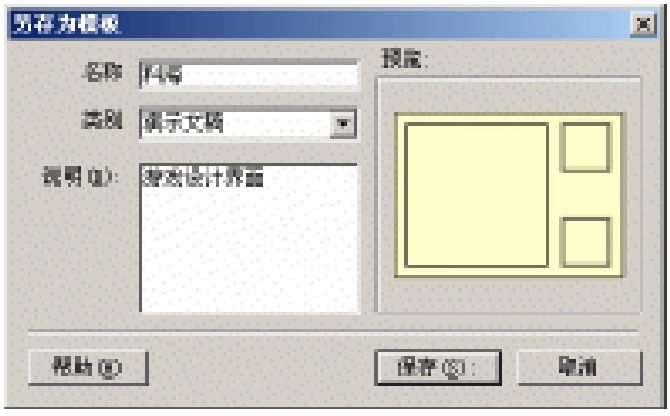

图 1.14 “另存为模板”对话框

图 1.15 “组件”面板

1.2.13 文本功能强大

Flash MX 中的文本功能非常强大，不仅可以仿效古代书籍的纵向编排，而且可以在纵向文本中将局部字符进行旋转操作，使整个文本更具特色，但这些功能目前还只能在静态文本中得以实现。

虽然动态文本（或输入文本）并没有相当程度的突破，但是要对动态文本（或输入文本）进行实例命名，开发人员就可以利用脚本控制其文本内容、格式等。

1.3 应 用 领 域

Flash 这款优秀的动画制作软件虽然最初产生并应用于 Web 页面的动画创建，但是随着软件的逐步推广，所涉及的领域也就越来越广泛。下面将 Flash 动画的应用领域进行简

单分类，以清楚 Flash 究竟可以用来做些什么？

1.3.1　主页动画

互联网时代的今天，盛行的 Flash 动画成为一种时尚的技术，几乎每天都有成千上百优秀的 Flash 作品在网上流传。在浏览 Web 页面时，只要在动画上单击右键，如果弹出的快捷菜单中显示有 Macromedia Flash Player 的字样，那么该动画肯定是 Flash 动画。

盛行 Flash 的今天，Web 页面上几乎随处可以见到 Flash 动画的身影，如 Web 页面中的 LOGO 标志、Banner 广告条、甚至整个页面中全部是 Flash 动画。

当然 Flash 也完全具备独立完成整个站点创建的能力，图 1.16 所展示的是国外设计人员制作的 Flash 动画站点，整个站点由近 20 个 Flash 动画文件构成，这些动画相互嵌套、调用成为动态交互的 Web 站点。整个站点所展示的声音、交互都可谓是顶尖高手的作品，赏心悦目的同时也体会到 Flash 动画的无限创作。

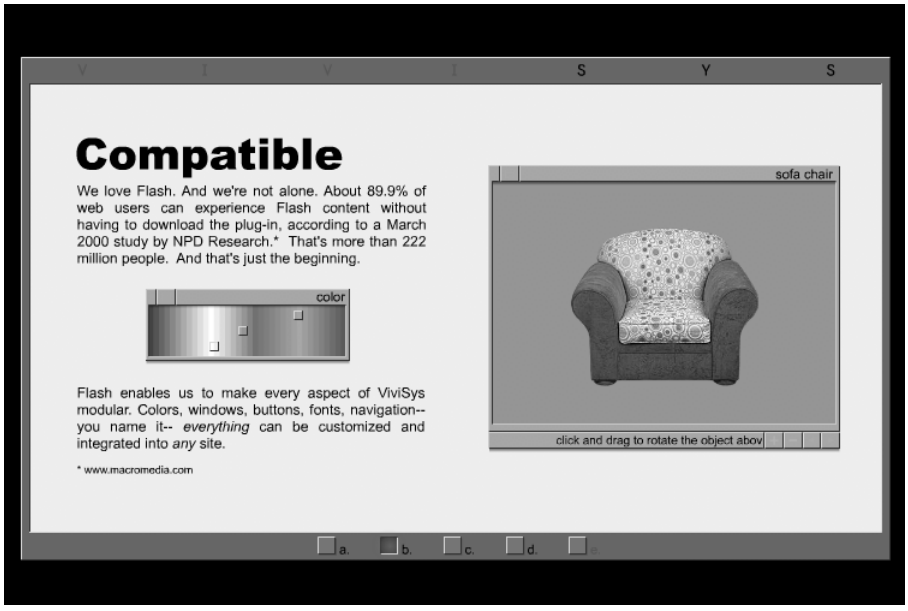

图 1.16　Flash 站点

1.3.2　MV 创意

MP3 并不是什么新鲜的东西，相信您已经聆听过 MP3 所带来的动听音乐。虽然 MP3 文件小且音质佳、便于随身携带聆听，但是美中不足的是当你聆听时无法观看到随着 MP3 节奏而变化的视频效果和歌词字幕。Flash 则可以将 MP3 和动画画面融为一体，实现音乐画面的完美结合，达到赏心悦目的效果！

网络上流传了大量的由 Flash 制作的 MV，您可以尝试着上网搜索几部 MV。经典的英文名曲《Pretty Boy》经高手配上精美的画面，如图 1.17 所示，再次聆听时真是倍感亲切。

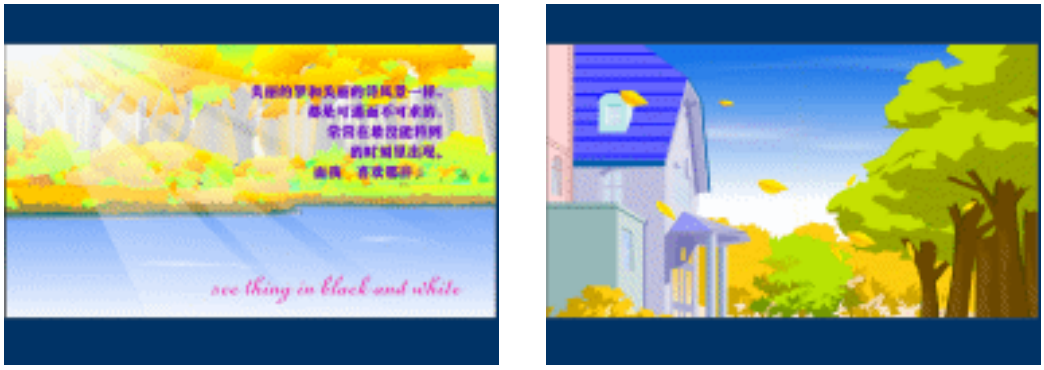

图 1.17 Flash MV《Pretty Boy》

1.3.3 电子贺卡

每逢过年过节时，朋友之间总喜欢相互问候一声。但是这样就显得比较枯燥，而且平淡得无法表达朋友之间的那份深情，此时您是否考虑利用 Flash 制作动画贺卡呢？

如果您不想独具匠心，也可以选择网上的电子贺卡进行发送。网上已经发布了大量这样的电子贺卡，其中有圣诞卡、新年卡等众多类别。图 1.18、1.19 所示的就是在网上获取的圣诞卡和新年卡。

图 1.18 圣诞卡

图 1.19 新年卡

1.3.4 交互游戏

游戏恐怕是最能体现交互性的使用了。那么 Flash 是否具备制作游戏的能力呢？回答当然是十分肯定的——可以！但是需要涉及大量有关游戏制作的知识理论，更需要在脚本编写方面具有一定的功底。

或许您曾经沉醉于掌中宝中一款叫“俄罗斯方块”的游戏，现在您可以在计算机上开始 Flash 版俄罗斯方块的拼杀，如图 1.20 所示；图 1.21 所示的黑白棋也是非常风靡的益智类游戏，工作闲暇时可以玩玩，达到劳逸结合的目的！

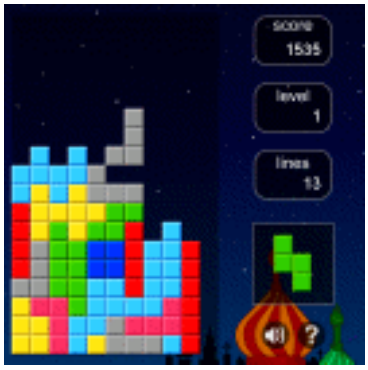

图 1.20　俄罗斯方块

图 1.21　黑白棋

1.3.5　教学课件

软件的学习一直受到广大迫切需要获取相关知识的读者的关心，希望能够很有成效地进行学习。如果软件教程单纯地使用文字叙述，这无疑是令人头疼的方法——难以应用上手、缺少实际操作！

如果利用 Flash 的交互性制作软件的学习教程，可以完美地将图文结合，实现手把手的教学模式。相信这绝对是一种很好的解决方法！图 1.22 所示的是关于 FreeHand 学习的课件教学动画，完全具备交互操作、手把手传递知识的目的！

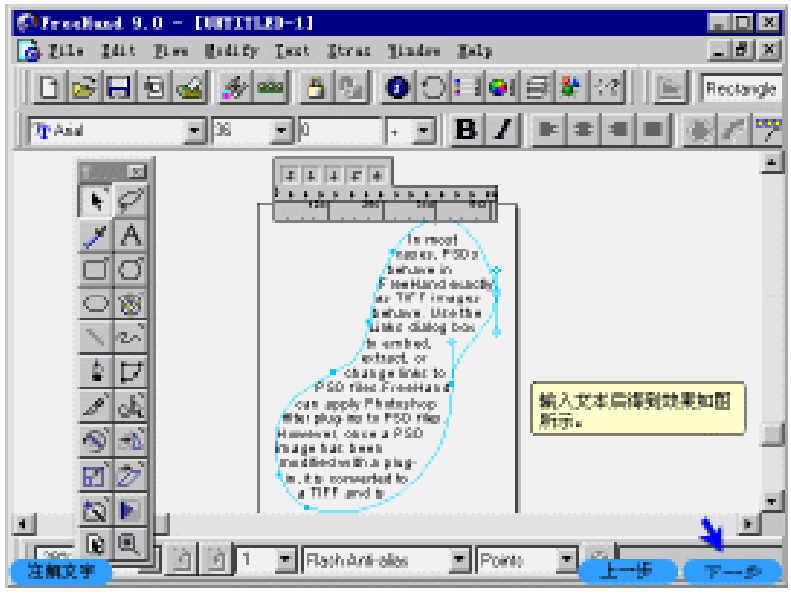

图 1.22　教学课件

1.3.6　产品广告

作为公司的对外产品宣传演示，利用 Flash 制作动画可以说是一种行之有效的方法。根据产品演示内容的不同，或者注重宣传如何正确使用该类产品，或者抽象展示产品的应

用范畴等。

图 1.23 所示的是中华牙膏对外宣传的 Flash 动画广告，采用歌曲贯穿整个动画、夸张地展示牙齿受到的种种侵袭，转而将画面切换到中华牙膏的使用。整个动画比较诙谐，可以给浏览用户留下较深的印象，达到广告宣传的目的！

图 1.23　产品广告

1.4　小结与练习

本章介绍了 Flash MX 具有的功能，其中界面布局的整体设计使得动画制作更加简便易行；同时由 Flash 动画所具有的特点探讨了 Flash 盛行的原因；最后粗略地对 Flash 动画的应用领域进行分类。

本章只是初步接触 Flash，或许您还感到有些陌生。那么在以后章节的学习中，您将与 Flash 进行更亲密地接触……

通过本章的学习，希望能够完成以下练习：

1．逢年过节免不了要和远方的朋友问声好，发送形式各样的电子贺卡传递这份友情。请访问网易的电子贺卡中心 http://cards.163.com，选择其中精美的 Flash 电子贺卡，给您远方的朋友发份 Mail 吧！！

2．闪客帝国（http://www.flashempire.com）是国内著名的 Flash 网站，广大 Flash 动画爱好者、设计人员蜂拥而至，相互之间进行经验交流和疑难解答。而且在该网站中还提供了具有大量精彩 Flash 动画的排行榜，多浏览其中的动画作品可以增加您对动画的理解！

3．Flash 软件的诞生地——Macromedia 公司，该公司的网站一定不要错过，访问网站 http://www.macromedia.com 可下载试用版的 Flash MX，同时还可以了解到 Macromedia 公司的其他软件产品。如果您的英文并不是太好，也没有必要着急，只要访问网站 http://www.macromediachina.com 即可。

第 2 章　工 具 应 用

尽管 Flash 只是一款小型的动画制作软件，但是 Flash 所提供给动画设计人员和制作人员的空间是无限的。真正要创作出一部经典的 Flash 动画作品，不只是单纯地依靠 Flash 技术的应用，而更多的是创作人员本身所具备的对动画制作知识的悟性！

如果要利用 Flash 设计出具有相当水准的动画作品，整个创作环境的熟悉和基本命令的应用是必不可少的。否则，就会多走弯路或进入岔道，从而导致完全偏离最初的设计构思。本章在熟悉创作环境的同时，深入讲解了 Flash 的工具及其应用。

通过本章的学习，应该达到如下目的：

- 熟悉创作环境
- 学会绘制矢量图形
- 掌握工具应用技巧
- 了解辅助工具使用

2.1　认识创作环境

Flash MX 在创作环境的整体界面上进行了很大程度的改善，似乎是为了更加贴近 Windows XP 亮丽的操作环境。但无可置疑的是，Flash MX 创作环境的调整，不仅体现了应用软件的人性化，而且更方便 Flash 动画的创建！！

整个创作界面大致可以分为 6 大部分，分别为菜单栏、工具箱、时间轴、工作区、“属性”面板、面板区，如图 2.1 所示。现将这 6 大部分分别进行较为详细地讲解。

2.1.1　菜单栏

菜单栏是由“文件”、“编辑”、“查看”、“插入”、“修改”、“文本”、“控制”、“窗口”以及“帮助”菜单组成，汇集了动画创作过程中的大量命令。

为了更加快捷地创建动画，部分经常涉及的菜单命令相应设置了快捷键。在本书中将以快捷键方式调用常用的菜单命令，以加快 Flash 动画的创建速度。

2.1.2　工具箱

通常，工具箱位于整个界面的左侧，其中放置了动画创建过程中的图形绘制、文本创建、视图查看以及颜色设置等工具。工具箱中各个工具的名称如图 2.2 所示，其中括号中的字母表示切换到相应工具的快捷键。

工具箱底端为当前所选工具的附加选项，并非所有工具都有相应的选项，譬如部分选取工具、直线工具、钢笔工具等就没有相应的附加选项。图 2.2 中所示的当前所选工具是箭头工具，其相应的附加选项有 3 个，分别是对齐对象、平滑和伸直。

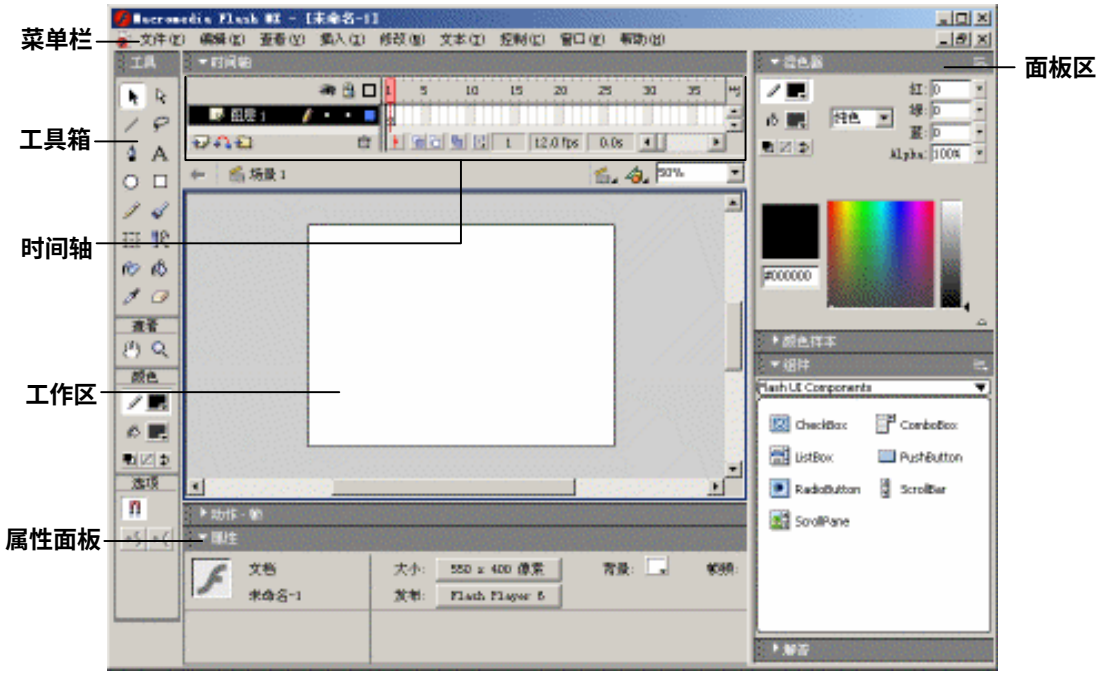

图 2.1 Flash MX 主界面

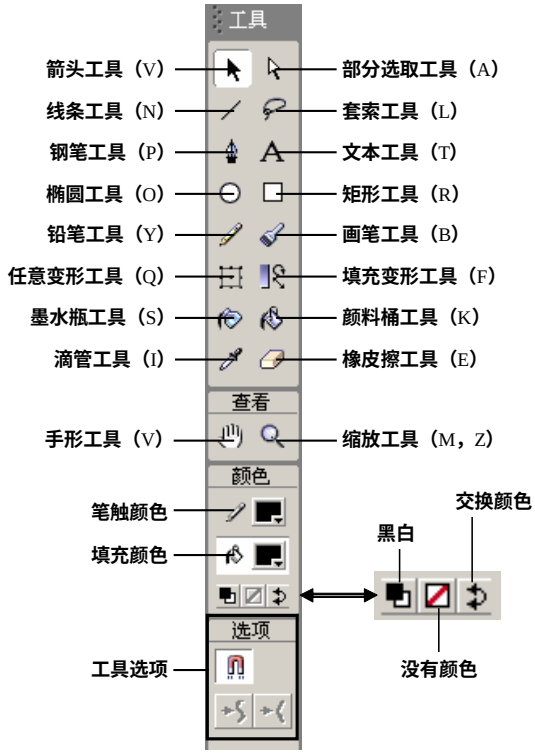

图 2.2 工具箱

注意：利用 Flash 创作动画的过程中，如果发现需要应用的按钮或菜单项呈现灰色，就应该确定使用该命令的条件还没有成立。譬如图 2.2 中所示的平滑按钮和伸直按钮呈灰色显示，这是由于这两个按钮使用时需要选中要调整的矢量图形。

提示：Flash 所提供的图形绘制功能并不是太强，毕竟其主要功能是动画制作与合成。通常在绘制复杂的、精美的图案时都需要借助其他矢量图形处理软件，如 FreeHand、Illustrator、CorleDraw 等，这些软件的最新版本都支持 Flash 动画的 SWF 导出格式。

2.1.3　时间轴

时间轴在动画创作中具有相当重要的地位，或许你对时间轴还比较陌生。其实，现实生活中经常接触到类似于时间轴的概念。如每天的日程安排，在这段时间中先完成什么事情，而在另外一段时间中又处理哪些事务；还有在观看电影时，总有反派主角与正面主角针锋相对的镜头。

由此可见，时间轴就是由时间与深度构成的二维空间，其作用就是合理地安排动画中各个角色的登台时间、表演内容等。然而 Flash 动画制作过程中，时间轴并不是简单的日程安排，它可能记录更多的内容。例如调用动画脚本、确定关键帧的标识名称、调整图层的叠放次序等等。图 2.3 中较详细地体现了 Flash 动画的时间轴应用。

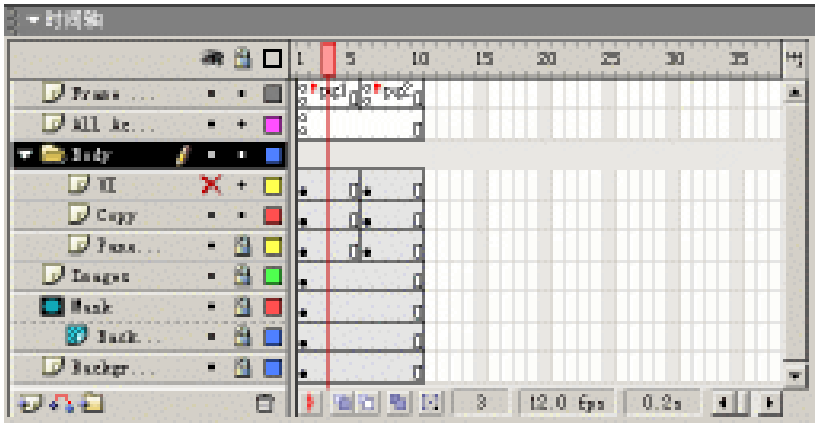

图 2.3　时间轴

注意：时间轴上的数值 1、5、10…是为了便于动画帧的计数，其中的红色矩形称之为播放头，在动画创建过程中可以前后移动，以查看动画画面是否连贯。播放头只能在已编辑的帧范围内移动，图 2.3 中最终的编辑帧数为 10，也就是说播放头在 1～10 帧之间移动，而不会出现在第 10 帧以后。

2.1.4　工作区

在 Flash 动画画面中将会出现各式各样的角色，工作区就是这些角色分布的区域。默认情况下，白色区域通常称之为舞台，也就是生成 Flash 动画的显示区域，而舞台外的部

分是不可见的。

工作区的右上角放置了 3 个下拉选框，如图 2.4 所示，分别是编辑场景、编辑元件以及缩放比例。通过对它们的选择，可在动画创建过程中自由地切换所要编辑的对象、适当调整放大倍数。

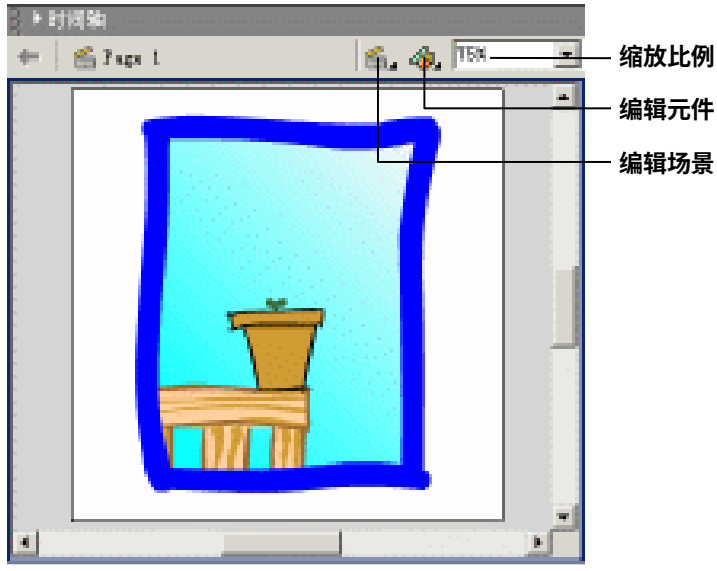

图 2.4 工作区

提示：工作区中的舞台显示有两种模式——工作区模式和非工作区模式。可以通过选择“查看”|“工作区”命令进行切换，相应的快捷键为 Ctrl+Shift+W。这两种模式略有差别，当取消“工作区”时舞台显示处于非工作区模式，此时如果角色处于舞台外的上部或左侧，那么该角色在编辑过程中是看不到的。因此，在此推荐使用工作区模式进行动画的创建。

2.1.5 “属性”面板

“属性”面板是智能化极强的功能设置面板，其主要作用是根据选择对象的不同，提供相关的属性设置内容。在没有选择任何对象的情况下，“属性”面板中通常显示文档属性的相关内容，如图 2.5 所示，其右下角的三角形按钮可以部分折叠或完全展开“属性”面板。

图 2.5 “属性”面板

当选择工具箱中的椭圆工具时，“ 属性 ”面板中显示椭圆工具需要的笔触颜色、笔触高度、笔触样式以及填充颜色等，如图 2.6 所示。

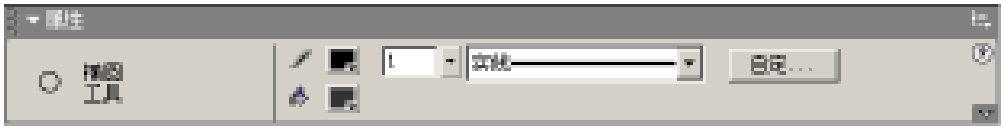

图 2.6　“ 属性 ” 面板（椭圆工具）

如果选择其他工具或选定不同的对象，“ 属性 ”面板又将呈现另外一番面貌。总而言之，善变的 “ 属性 ” 面板在动画的制作过程中将会带来极大的方便。而有关 “ 属性 ” 面板的应用将在进一步的学习中逐步了解并掌握。

2.1.6　面板区

整个界面的右侧纵向排列着多个面板，这些面板有序地布置在该区域中，极大地节约了屏幕的使用空间，同时也方便地显示了各个面板的设置内容。单击面板区中深灰色的文字标签条，可以折叠或展开相应的面板。图 2.7 显示了面板区中不同面板展开的情况。

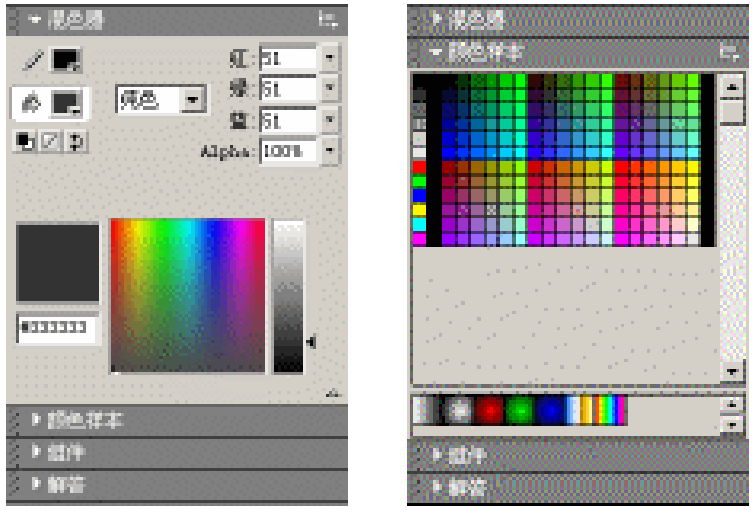

图 2.7　智能折叠的面板

提示：为了充分利用屏幕空间，不致使时间轴、“ 属性 ”面板等影响工作区的屏幕使用空间，可以采用同样的方法单击深灰色的文字标签条进行折叠。如果需要更大屏幕空间的工作区，可以选择 “ 查看 ”|“ 隐藏面板 ” 命令将所有面板都隐藏，相应的快捷键为 F4 ；再次选择该命令可以恢复所有面板的显示。

2.1.7　工具栏

许多用户都习惯 Windows 系统的图形化操作，例如 Office 应用软件常用工具栏中的新建、打开等图标按钮，只要鼠标轻松一点就可以执行相关命令，而不必在菜单中去寻找相应的命令。其实，Flash 中也提供了部分工具栏，只是默认情况下没有显示。选中 “ 窗口 ”

|“ 工具栏 ”|“ 主要栏 ”命令，即可在屏幕中显示主要栏。另外，还可以选中其中的“ 状态栏 ”、“ 控制器 ”显示状态栏和控制器，如图 2.8 所示。

图 2.8 主要栏、状态栏以及控制器

2.2 绘制矢量图形

矢量图形的绘制是制作 Flash 动画制作中必须掌握的基本知识，Flash 动画画面中的图形基本上都是在 Flash 中绘制生成的。因此，绘制矢量图形就是对 Flash 工具的熟练应用。

现在就利用 Flash 所提供的工具制作一些简单的图形图案。

2.2.1 长方体

现实生活中，长方体或许是最为简单的物体了，如图 2.9 所示。那么，在 Flash 中究竟如何绘制长方体呢？具体步骤如下：

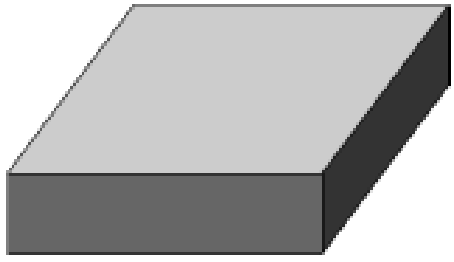

图 2.9 长方体

（1）选择“ 文件 ”|“ 新建 ”命令新建文档，选中工具箱中的矩形工具，单击颜色区中的“ 黑白 ”图标按钮设置默认颜色，选择填充颜色后单击“ 没有颜色 ”按钮取消填充颜色，如图 2.10 所示。将鼠标光标移动到舞台上，按住鼠标左键斜向拖动，释放鼠标左键后绘制得到黑色线条的矩形，如图 2.11 所示。

图 2.10 设置颜色

图 2.11 绘制矩形

（2）按 V 键切换到箭头工具，双击黑色线条矩形的任意一边，选中矩形的全部线

条；按住 Alt 键的同时斜向拖动，复制得到另一个矩形，如图 2.12 所示。

（3）按 N 键切换到线条工具，将两个矩形的顶点一一对应地连接起来；再次切换到箭头工具，判断长方体各个面的可见性，将不可见的线条删除，结果如图 2.13 所示。

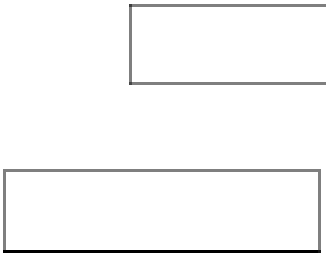

图 2.12　复制矩形

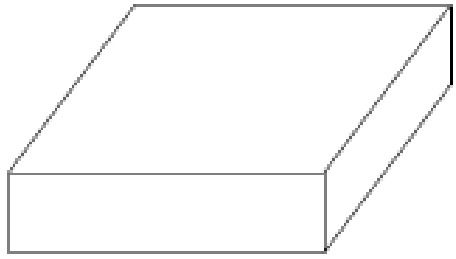

图 2.13　长方体框架

（4）按 K 键切换到颜料桶工具，单击填充颜色框弹出颜色样本，如图 2.14 所示选择浅灰色，单击后选中颜色；将光标移动到长方形的顶面上单击填充浅灰色。相同的操作方法，在各个面上填充不同的颜色，得到色彩丰富、明暗分明的长方体。

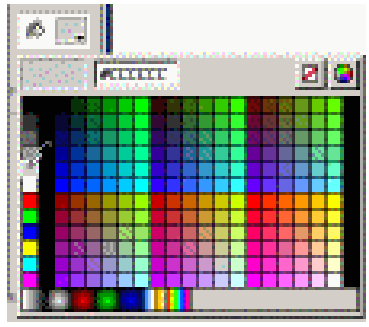

图 2.14　设置填充颜色

2.2.2　星形图案

由颜色的深浅呈现事物的三维效果，如图 2.15 所示的星形图案就是其中一个典型的例子。注意生成星形图案的基本图形，这是制作的关键所在。

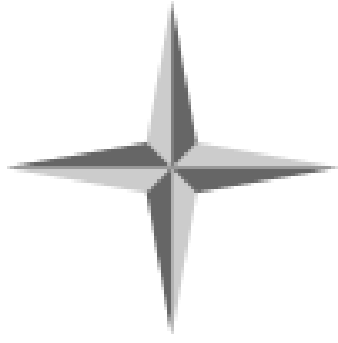

图 2.15　星形图案

具体制作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按 R 键切换到矩形工具，在“属性”面板中设置笔触颜色为黑色（#000000）、笔触高度为 0.25、笔触样式为实线，并取消填充颜色的设置，如图 2.16 所示。

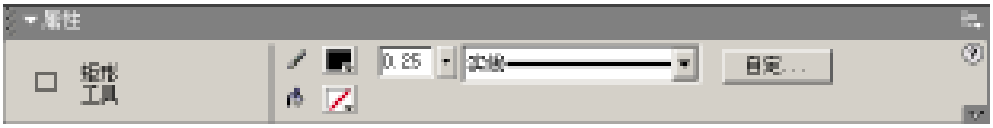

图 2.16 设置矩形工具属性

（2）按住 Shift 键的同时拖动，绘制得到黑色线条的正方形；按 N 键切换到线条工具，连接正方形的对角顶点，如图 2.17 所示。

（3）按 V 键切换到箭头工具，沿对角线的方向拖动正方形的右上顶点，得到燕形图案，如图 2.18 所示。

（4）双击选中燕形图案中的任一线条，选中全部线条后按 Q 键切换到任意变形工具，在燕形图案上将出现 8 个方形句柄和 1 个圆形控制点。该圆形控制点就是用于确定图形的旋转中心。将圆形控制点拖动到左下顶点，如图 2.19 所示。

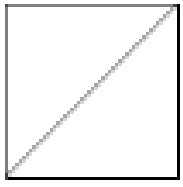

图 2.17 正方形

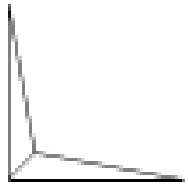

图 2.18 燕形图形

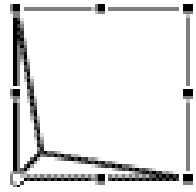

图 2.19 设置旋转中心

（5）选择“窗口”|“变形”命令，弹出“变形”面板，设置旋转角度为 90 度，如图 2.20 所示，其他参数保持不变。连续单击“变形”面板右下角的“拷贝并应用变形”按钮，直至得到如图 2.21 所示的星形图案框架。

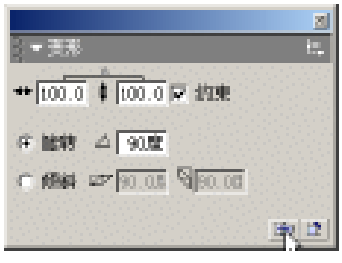

图 2.20 “变形”面板设置服

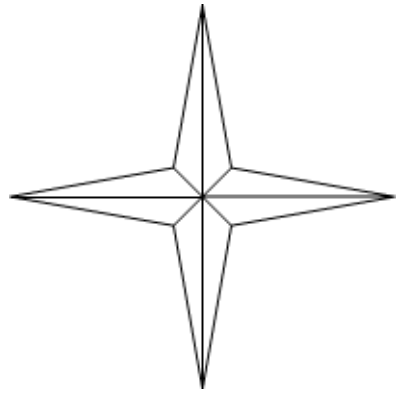

图 2.21 星形图案框架

（6）按 K 键切换到颜料桶工具，选择浅灰色（#CCCCCC）和深灰色（#666666）分别填充星形图案的各个区域。最后，再将起分割作用的线条全部删除。

2.2.3　黑白双鱼

太极文化源远流长，其标志性的图案——黑白双鱼也极具特色，如图 2.22 所示。虽然在众多平面设计软件中都有相应的教程，但是在 Flash 中制作黑白双鱼又有不同的方法，这主要是由于 Flash 中的线条具有相互分割的作用。

图 2.22　黑白双鱼

下面介绍制作黑白双鱼的一种简单方法，步骤如下：

（1）按快捷键 Ctrl+N 新建 Flash 动画文档，按 O 键切换到椭圆工具，设置笔触颜色为黑色（#000000）、取消填充颜色的设置；在舞台上斜向拖动，得到黑色环状线条，如图 2.23 所示。

（2）按 V 键切换到箭头工具，单击环状线条将之选定，选择“窗口”|“信息”命令，弹出“信息”面板，在该面板中将宽度和高度均设置为 200；单击右侧中间的方块确定以选定对象的中心点为参考坐标，同样设置 X、Y 坐标均为 200，如图 2.24 所示。

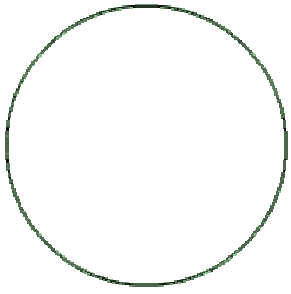

图 2.23　绘制圆

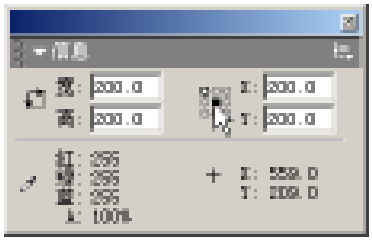

图 2.24　设置圆的尺寸及坐标

提示：“信息”面板中存在两种参考坐标，即左上点和中心点。选择不同的参考点，选定对象相应的坐标是不同的。图 2.24 所示中心点方格被黑色填充，说

明当前选定对象的坐标是以中心点为参考的坐标。在“信息”面板中还可以获取光标所在位置的坐标以及“红绿蓝 A”值。

（3）按快捷键 Ctrl+T 调用“变形”面板，选中“约束”复选框锁定纵横比，并在其左侧的宽度文本框中输入 50%，单击该面板右下角的“拷贝并应用变形”按钮，如图 2.25 所示，得到原有圆环直径一半的另一个圆环；继续在“变形”面板的宽度文本框中输入 20%，同样单击“拷贝并应用变形”按钮得到更小的圆环，形成同心圆，如图 2.26 所示。

提示：在“变形”面板中还可以设置旋转角度、倾斜角度，使所选的对象发生旋转和倾斜变形。如果图形的变形并不要求与原有图形之间保持得非常精确，完全可以利用工具箱中的任意变形工具进行操作完成。

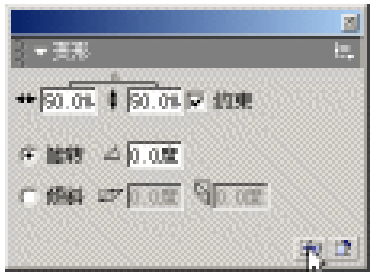

图 2.25 “变形”面板

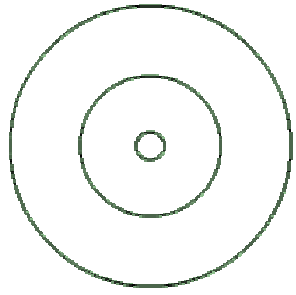

图 2.26 同心圆

（4）按住 Shift 键逐个单击两个小圆，选择“修改”|“组合”命令将这两个小圆组合成一个对象，以防止操作过程中的误操作引起圆环的变形和圆环之间位置的偏移。按快捷键 Ctrl+I 调用“信息”面板，设置组合圆环中心点的参考坐标为（150，200），如图 2.27 所示，调整组合圆环后的图形如图 2.28 所示。

（5）按住 Alt 键水平向右拖动组合圆环，复制得到另一个组合圆环，同样在“信息”面板中设置其中心点的参考坐标为（250，200），如图 2.29 所示。至此，黑白双鱼的框架就基本呈现出来了，如图 2.30 所示。

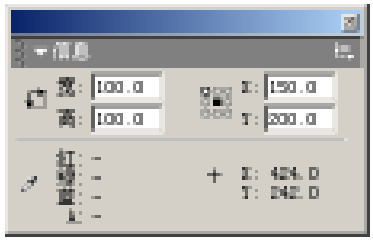

图 2.27 设置坐标（左）

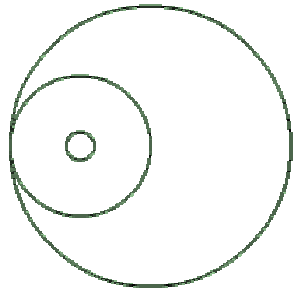

图 2.28 左侧组合圆环

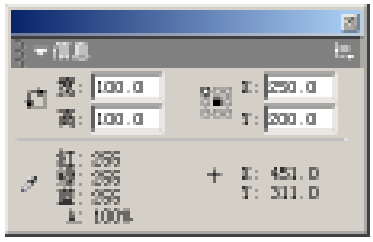

图 2.29　设置坐标（右）

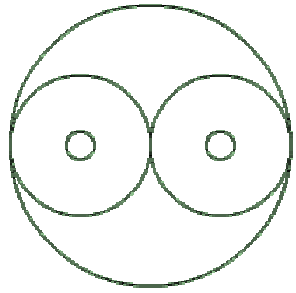

图 2.30　黑白双鱼框架

（6）由于黑白双鱼是以黑色和白色作为填充颜色的，因此舞台的背景颜色为白色将会影响黑白双鱼的颜色显示。单击舞台的空白区域，此时“ 属性 ”面板中显示有关文档的属性设置，单击其中的背景颜色框，调整背景颜色为#99CC99，如图 2.31 所示。

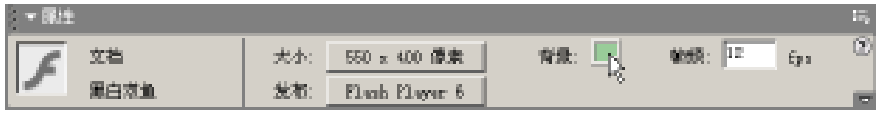

图 2.31　设置背景颜色

（7）选择“ 编辑 ”|“ 全选 ”命令选中舞台上的所有圆环，随后再选择“ 修改 ”|“ 取消组合 ”命令取消圆环的组合状态。由于线条之间的分割作用，大圆环内被小圆环分成多个区域，此时所有的线条都处于选定状态。

（8）按 K 键切换到颜料桶工具，设置填充颜色为黑色（#000000），在线条之间的区域内逐个填充；再设置填充颜色为白色（#FFFFFF），填充其他区域。最后，按 Delete 键删除黑白双鱼之间起分割作用的线条。

这样，太极文化的标志性图案就制作完成了。

2.2.4　心形像框

心形像框的制作主要由钢笔工具来完成。利用钢笔工具进行矢量图形的绘制，既具有一定的随意性，又可以更加透彻地掌握矢量图形的本质。本例中为了在静态效果中体现心形像框“ 闪 ”的效果，在其边缘以杂乱的短小线条加以点缀，如图 2.32 所示。

图 2.32　心形像框

具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建 Flash 动画文档，按 P 键切换到钢笔工具，“属性”面板中呈现钢笔工具的相关属性设置，在其中设置笔触颜色为红色（#FF0000）、笔触样式为“极细”以及填充颜色为红色（#FF0000），如图 2.33 所示。

图 2.33　设置钢笔工具属性

（2）在舞台上单击，此时出现一小圆圈称之为节点；将光标向左横向移动一定距离后，按住鼠标左键斜向下拖动，此时在第 1、2 节点之间显示一段弧线，如图 2.34 所示，其中的直线部分称之为节点的控制句柄。将光标移动到第 1 个节点的正下方，单击鼠标左键确定心形的第 3 个节点；再将光标移动到第 1 个节点的右侧，按住鼠标左键斜向上拖动，如图 2.35 所示。最后，将光标定位在第 1 个节点处单击，使得心形图形闭合，如图 2.36 所示。

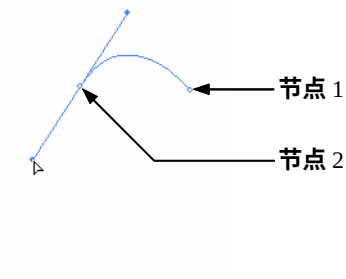

图 2.34　确定节点（1）

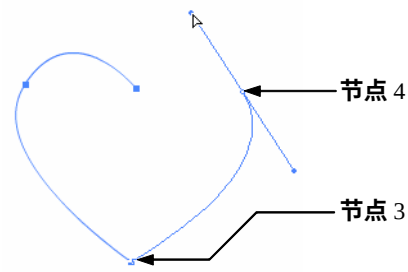

图 2.35　确定节点（2）

（3）利用钢笔工具绘制得到的心形图形在外形上存在些许的不足，还需进行细微地调整。按 A 键切换到部分选取工具，在舞台上按住鼠标左键拖动出一矩形框，完全框住整个心形图形，此时会看到心形图形的 4 个节点以及相应的控制句柄。只要在需要调整的节点上，单击选定该节点后，按住鼠标左键拖动即可；而心形曲率的调整可对左右 2 个节点的控制句柄进行拖动，最终得到的心形图形如图 2.37 所示。

图 2.36　心形图形

图 2.37　调整心形图形

（4）心形图形已经圆满完成，但是通常不采用这样单调的图形。可对此图形稍加修饰，

以使其更具实用性，譬如作个心形镜框等。

按 V 键切换到箭头工具，单击心形填充部分后，按 Delete 键将之删除。双击心形线条将之全部选定，在“属性”面板中设置笔触高度为 10pts，并单击右侧的“自定”按钮，在弹出的“线型”对话框中选择类型为“斑马线”，其他参数设置如图 2.38 所示。

图 2.38　“线型”对话框

（5）单击“确定”按钮后，心形线条也就发生相应的变化，如图 2.39 所示。在实际使用中可以在心形像框中置入相片，如图 2.40 所示。关于其制作方法，在进一步的学习中将会有所领悟。

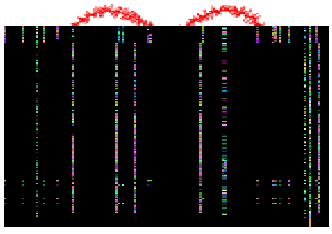

图 2.39　心形外框

图 2.40　置入相片

2.2.5　虚化矩形

利用工具箱中的矢量绘图工具得到的边界通常比较硬朗，为此 Flash 中特别增加了虚化边界的功能。通过该功能可以容易地得到边界比较柔和的矢量图形，由实到虚存在一定过渡，比一般的矢量图形更容易接受。步骤如下：

（1）双击工具箱中的矩形工具，弹出“矩形设置”对话框，在“角半径”框中输入数值 20，如图 2.41 所示，单击“确定”按钮确认。

> 提示：“矩形设置”对话框也可以在工具箱中选中矩形工具后，单击其工具选项区中的圆角矩形半径进行调用。

“属性”面板中显示矩形工具的相关设置，如图 2.42 所示，设置笔触颜色为无，填充颜色为黑色（#000000），也就是说只要矩形的填充部分。

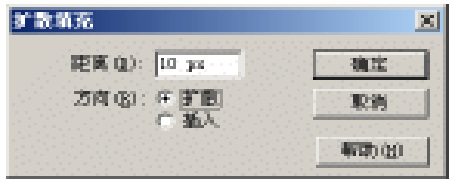

图 2.45　“扩散填充”对话框

原始矩形　　扩散 30 px　　插入 30 px

图 2.46　矩形扩散前后比较

2.3　工具应用技巧

2.3.1　箭头工具

在动画创建过程中，箭头工具主要应用于对象的选取和移动，同时还具备使矢量图形变形的功能。通常，使用箭头工具时因光标所处的位置不同而呈现以下 4 种形式。

框选光标　光标所在位置不存在可编辑对象时呈现此光标形式。此时，按住鼠标左键拖动出一定的矩形区域，如果区域内包含可编辑对象，则该对象被选中，如图 2.47 所示。

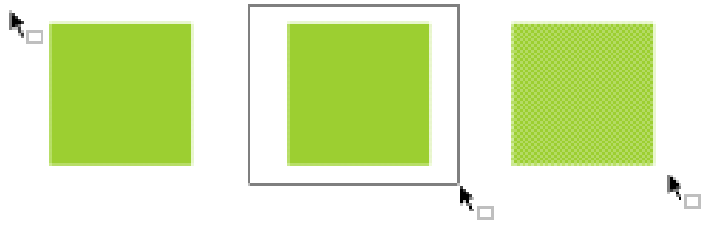

图 2.47　框选光标的使用

移动光标　光标所在位置存在可编辑对象时呈现此光标形式。此时，按住鼠标左键拖动，可以改变该对象的位置，如图 2.48 所示。

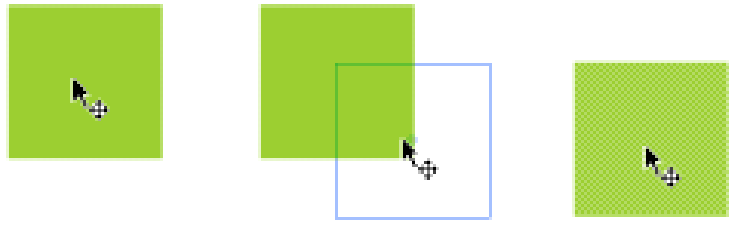

图 2.48　移动光标的使用

直角光标 光标所在位置为矢量图形的尖角节点（如矩形顶点、直线两端）时呈现此光标形式。此时，按住鼠标左键拖动，可以改变相应节点的位置，如图 2.49 所示。

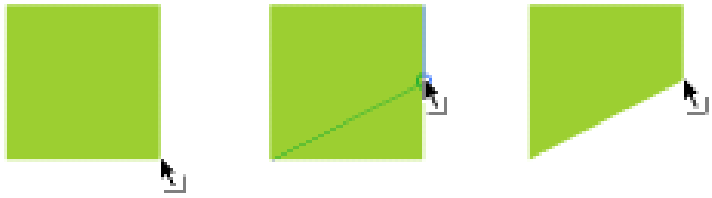

图 2.49 直角光标的使用

弧形光标 光标所在位置为矢量图形的边界时呈现此光标形式。此时，按住鼠标左键拖动，可以调整矢量图形边界的弧度，如图 2.50 所示。

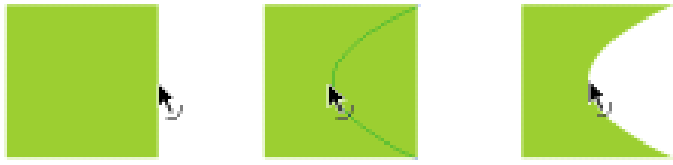

图 2.50 弧形光标的使用

提示：当光标处于矢量图形的边界处呈现弧形光标时，按住 Alt 键的同时按住鼠标左键拖动，可以调整矢量图形边界的折角，如图 2.51 所示。

图 2.51 弧形光标的使用（按住 Alt 键）

2.3.2 铅笔工具

铅笔工具在使用过程中，除了应该在“属性”面板中设置绘制线条的笔触颜色、笔触高度以及笔触样式外，还应该注意铅笔工具使用模式的选择。选中铅笔工具后，在工具箱底端的工具选项中单击其中的按钮，弹出铅笔工具使用模式的选项菜单，如图 2.52 所示，在此可选择“伸直”模式、“平滑”模式或者“墨水”模式。

图 2.52 铅笔工具使用模式

现在，就针对铅笔工具 3 种不同使用模式进行比较。为了更加清楚地了解铅笔工具在不同模式下所绘制得到的线条效果，使用铅笔工具时均设置笔触颜色为黑色

（#000000）、笔触高度为 1pts、笔触样式为“ 实线 ”。

使用铅笔工具任意绘制线条，比较鼠标左键释放前后的变化。“ 伸直 ”模式下绘制的线条在尽量符合绘制轨迹的情况下将之局部直线化，使线条具有较少的节点 ;“ 平滑 ”模式下绘制的线条在尽量符合绘制轨迹的情况下将之局部曲线化，使线条具有较少的节点的同时整个线条比较平滑 ;“ 墨水 ”模式下绘制的线条是最贴近绘制轨迹的，但得到的最终线条具有较多的节点。“ 伸直 ”模式、“ 平滑 ”模式和“ 墨水 ”模式这 3 种模式下绘制线条时释放鼠标左键前后的比较，如图 2.53 所示。

“ 伸直 ”模式　　“ 平滑 ”模式　　“ 墨水 ”模式

图 2.53　不同模式下的铅笔绘制结果

2.3.3　橡皮擦工具

橡皮擦工具可以对矢量图形进行擦除操作，以去除矢量图形中多余的部分。通常，橡皮擦工具应用于矢量图形的修正工作，因为擦除矢量图形的部分就可以看到其底层的对象或舞台的颜色。

单击工具箱中的橡皮擦工具，在工具选项中显示橡皮擦工具的相应附加选项，分别为橡皮擦模式、水龙头工具和橡皮擦形状，如图 2.54 所示。

单击“ 橡皮擦模式 ”按钮，弹出橡皮擦模式菜单，如图 2.55 所示，共有标准擦除、擦除填充、擦除线条、擦除所选填充以及内部擦除 5 种模式。

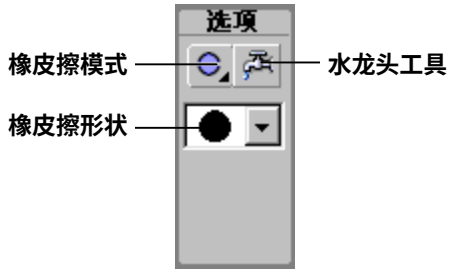

图 2.54　橡皮擦工具选项

图 2.55　橡皮擦模式

使用橡皮擦工具前，首先利用矩形工具绘制正方形填充，再利用椭圆工具在正方形填充中绘制圆环。按 E 键切换到橡皮擦工具，选择标准擦除模式，将光标定位在正方形填充的左上角，按住鼠标左键斜向下拖动（其他橡皮擦模式采用相同的操作方法），释放左键时可以看到矢量图形中的填充和线条部分都被擦除，如图 2.56 所示。

图 2.56 擦除模式比较

擦除填充模式：矢量图形中填充部分被擦除，但是线条没有影响；

擦除线条模式：矢量图形中线条部分被擦除，但是填充没有影响；

擦除所选填充：矢量图形中圆环内的填充是选中的，只有圆环内的填充部分被擦除，而线条部分和圆环外的填充并没有受影响；

内部擦除模式：矢量图形中鼠标拖动起始位置所在的填充部分被擦除，但是线条部分和其他填充部分并不受影响。

使用橡皮擦工具的水龙头工具前，首先利用矩形工具绘制正方形填充，再利用椭圆工具在正方形填充中间部分绘制圆环，并利用直线工具沿着正方形的对角线绘制直线以分割圆环。切换到橡皮擦工具，在工具选项中选中水龙头工具，将光标定位在填充部分单击，这时将删除由直线和圆环分割的半圆形填充；将光标定位在线条部分单击，则删除由直线分割的半个圆环，如图 2.57 所示。

图 2.57 使用水龙头工具

由此可见，水龙头工具对于矢量图形的删除是根据分割区域而逐个去除的。虽然操作比较简便，但如果要删除相互交叉的线条，利用箭头工具双击交叉线条中的任意一段线条后按 Delete 键进行删除的方法比较快捷。当然这两种方法在删除交叉线条时各有优缺点，关键是在创作过程中的熟练应用和灵活使用。

2.3.4　画笔工具

笔刷可以快速将矢量图形涂抹上不同的色彩，以创作出比较特殊的临摹效果。选中工具箱中的画笔工具，可以在“ 属性 ”面板中设置画笔工具的填充颜色，更重要的是在工具选项中可以选择画笔模式、画笔尺寸、画笔形状和锁定填充，如图 2.58 所示。

单击“ 画笔模式 ”按钮，弹出画笔模式菜单，如图 2.59 所示，与橡皮擦模式有些相似，也有 5 种模式，分别为标准绘画、颜料填充、后面绘画、颜料选择和内部绘画。

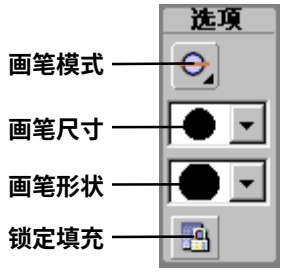

图 2.58　画笔工具选项

图 2.59　画笔模式菜单

使用画笔工具前，首先利用矩形工具绘制正方形填充，再利用椭圆工具在正方形填充中绘制圆环。按 B 键切换到画笔工具，选择画笔模式菜单中的标准绘画模式，将光标定位在正方形填充的左侧，按住鼠标左键横向拖动（其他画笔模式采用相同的操作方法），释放左键时可以看到矢量图形中的填充部分和线条部分都被画笔所绘制的填充覆盖，如图 2.60 所示。

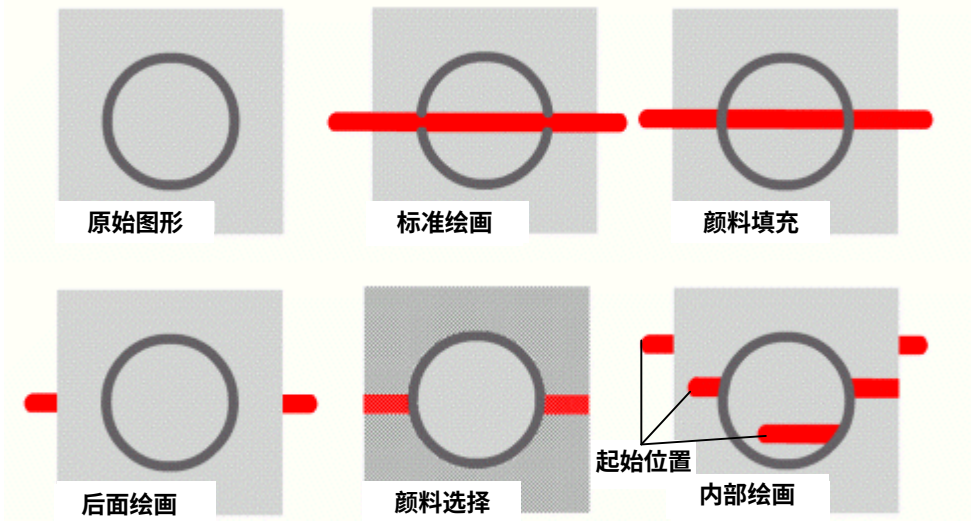

图 2.60　不同画笔模式的使用比较

颜料填充模式：矢量图形中的填充部分被画笔所绘制的填充覆盖，而线条部分并没有任何影响；

后面绘画模式：矢量图形中的线条部分和填充部分都没有任何影响，只有处于矢量图

形外的部分可以观看到画笔所绘制的填充；

颜料选择模式：矢量图形中圆环外的填充部分是选中的，画笔所绘制的填充覆盖这部分的填充，而其余部分并没有受到画笔所绘制的填充影响；

内部绘画模式：矢量图形中鼠标拖动起始位置处的填充部分由笔刷所绘制的填充覆盖，而其他部分并没有受到任何影响。

关于画笔尺寸和画笔形状的使用，相信您一试就可以清楚了！但对画笔工具选项中的“锁定填充”功能或许有些迷惑，现在就帮您解开这个谜团！

为了更加清楚地观看到锁定填充绘图的效果，使用画笔工具时设置填充颜色为色谱线性渐变，画笔模式为标准绘画模式。首先使用画笔工具横向进行较长距离的拖动，绘制出条状色谱填充；随后选中锁定填充按钮，在原先的色谱填充周围进行绘制，注意绘制得到的填充中的颜色变化，如图 2.61 所示。

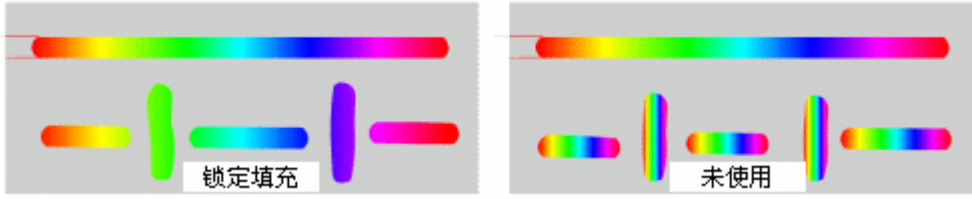

图 2.61 锁定填充的使用

如果在完成条状色谱填充的绘制后，并没有选中锁定填充按钮，采用相同的操作方法在原先的色谱填充周围进行绘制，那么此时绘制得到的填充中的颜色仍为色谱线性渐变填充，而不是锁定填充时色谱线性渐变中的局部颜色。

2.4 辅助工具一览

在动画制作过程中，通常需要些工具进行辅助创作，这会使整个动画在创建过程中具有比较合理的结构和编排，从而动画也就显得相当地有条理。

在 Flash 动画的创建环境中，主要提供了标尺、网格、辅助线以及快捷键等辅助工具。

2.4.1 标尺

标尺是丈量物体尺寸的工具，在 Flash 中调用标尺可以获知光标所在的坐标位置、动画角色放置的坐标位置、大概预测动画角色的大小尺寸；同时使动画设计人员更加清楚 Flash 创作环境的坐标系规定。

因此，标尺的使用对于动画创建过程中对象的定位是具有相当作用的，但是标尺在默认情况下是没有显示的。选中“查看”|“标尺”命令启动标尺功能，此时在动画工作区的左侧、顶端都出现相应的标尺，如图 2.62 所示。

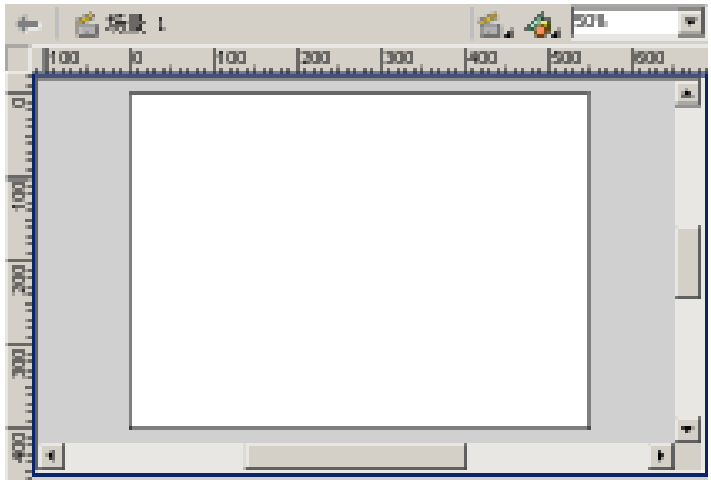

图 2.62　标尺

当标尺功能显示时，可以清楚地获知两标尺刻度（0，0）所代表的是舞台左上角的坐标，也就是说 Flash 中的坐标系的设置是以舞台的左上点为原点，水平向右为 X 轴正方向，垂直向下为 Y 轴正方向。

2.4.2　网格

曾经使用过坐标纸进行绘图的用户，一看网格就基本明白其用处。在 Flash 创作环境中，网格同样具有控制对象定位的功能，同时也为用户在绘制矢量图形时提供了方便。利用网格功能，可以比较轻松地实现在 Flash 界面中的机械制图，当然必须先要熟练掌握工具箱中的矢量绘图工具。

选中“查看”|“网格”|“显示网格”命令（相应的快捷键为 Ctrl+’），在动画角色编辑的舞台上就可以显示类似于坐标纸的小方格，如图 2.63 所示。

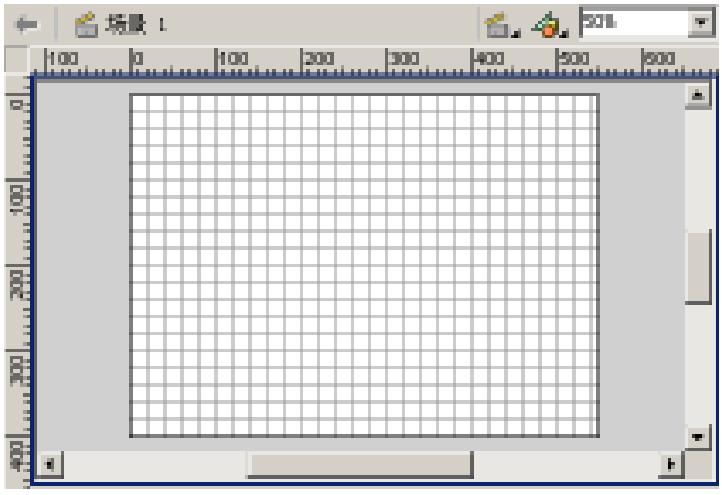

图 2.63　显示网格后的舞台

但是在动画创建时，对于不同的动画而言，所需的网格尺寸或许不一样。因此选择“查看”|“网格”|“编辑网格”命令（相应的快捷键为 Ctrl+Alt+G），弹出“网格”对话框，如图 2.64 所示。

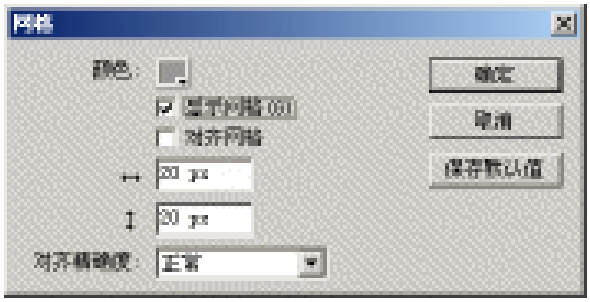

图 2.64 “网格”对话框

提示：除了在“网格”对话框中激活“对齐网格”功能外，还可以通过选择“查看”|“网格”|“对齐网格”命令（相应的快捷键为 Ctrl+Shift+G）的方法进行激活或屏蔽。

在该对话框中可以设置网格颜色、横向尺寸及纵向尺寸，同时可以选择复选框以激活“显示网格”和“对齐网格”功能。如果所做的这些设置是经常应用的，可以单击右侧的“保存默认值”按钮，以后新建的 Flash 动画文档的网格设置就会与之保持一致。

注意：Flash MX 中不仅提供了可以方便设置的网格功能，同时还提供了更加精密的像素级网格。只要选中“查看”|“对齐像素”命令，并将缩放比例调整到 400% 以上，就可以清晰地看到像素级网格，以利于设计人员绘制图形时尺寸的精确控制、各个对象位置的精确分布。

2.4.3 辅助线

网格显示时总是覆盖整个舞台，使得整个舞台显得比较繁琐。毕竟在大部分动画创作过程中是不需要这么多的网格线作为依据的。因此，Flash 同时提供了辅助线功能，在动画制作过程中放置几条辅助线就足以应付整个动画的创建，同时辅助线是可以随时进行移动定位的。

在显示标尺的情况下，将鼠标定位在标尺上按住鼠标左键，向舞台位置进行拖动，释放鼠标左键即可添加辅助线。重复上述操作就可以在舞台工作区添加多条辅助线，如图 2.65 所示。

在动画创建过程中，由于辅助线颜色与舞台上对象颜色相近或一致，体现不了辅助线的作用。此时，可以选择“查看”|“辅助线”|“编辑辅助线”命令，弹出“辅助线”对话框，如图 2.66 所示。

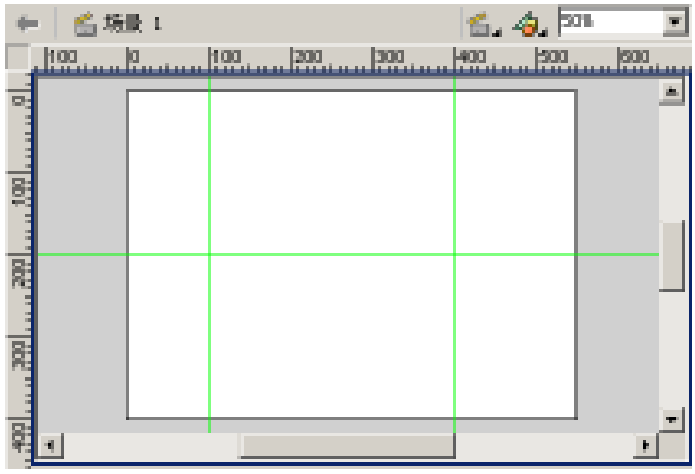

图 2.65　添加辅助线

图 2.66　“ 辅助线 ” 对话框

在该对话框中可以设置辅助线颜色；激活或屏蔽辅助线的显示、对齐以及锁定等功能；同时可以单击 “ 全部清除 ” 按钮一次性删除舞台上的所有辅助线。

> 提示：除了在 “ 辅助线 ” 对话框中激活或屏蔽辅助线的显示、对齐、锁定功能外，还可以通过选择 “ 查看 ”|“ 辅助线 ”|“ 显示辅助线 ” 命令激活或屏蔽辅助线的显示，相应的快捷键为 “ Ctrl+; ”；选择 “ 对齐辅助线 ” 命令激活或屏蔽辅助线的对齐功能，相应的快捷键为 “ Ctrl+Shift+; ”；选择 “ 锁定辅助线 ” 命令激活或屏蔽辅助线的锁定功能，相应的快捷键为 “ Ctrl+Alt+; ”。

2.4.4　快捷键

快捷键的使用是提高动画创建速度的有效途径，对于设计人员而言，一般都会精通多种图形图像设计软件，软件之间常用的快捷键基本相似，如新建 Ctrl+N、打开 Ctrl+O、保存 Ctrl+S 等等；但是有些工具或命令使用时功能等价，而不同软件之间的快捷键是不同的，这样在不同的创作环境中就比较容易相互混淆。

对此，Flash MX 就显得相当出色，允许用户设定不同的快捷键，使动画设计人员倍感亲切。选择 “ 编辑 ”|“ 快捷键 ” 命令，弹出 “ 快捷键 ” 对话框，如图 2.67 所示。Flash MX 内置了图形图像处理软件中的多个版本的快捷键，如 Fireworks 4、FreeHand 10、PhotoShop 6 等，同时也保留了 Flash 5.0 版本的快捷键设置。

图 2.67 “快捷键”对话框

提示：在 Flash MX 中内置的快捷键设置是不能修改的，必须先单击右上角的“复制设置”按钮得到相应的快捷键设置，随后就可以进行修改以适应自身操作习惯。

2.5 小结与练习

本章首先初步熟悉整个 Flash 创作环境，对于以后的动画设计和制作具有非常重要的作用。然后主要通过多个矢量图形的绘制掌握工具箱中绘图工具的使用，同时也掌握了部分菜单命令和面板应用方法。

通常，使用软件时经常会忽略辅助工具的应用。其实，辅助工具对于 Flash 动画的创建具有相当有益的帮助。

通过本章的学习，希望能够完成以下练习：

1．尝试使用工具箱中的工具进行简单图形的绘制、适当采用菜单命令，时刻注意“属性”面板中所发生的变化，争取最大程度地熟悉 Flash 动画的创作环境！

2．仔细观看星形图案，提取其中的基本形状，利用矩形工具、线条工具、任意变形工具、箭头工具、颜料桶工具以及“变形”面板等，采用另一种方法进行星形图案的绘制！

提示：绘制星形图案的另一种简便方法，其制作示意图如图 2.68 所示：

1．绘制尺寸较小的正方形，并连接其中一条对角线；

2．将具有对角线的正方形旋转 45 度；

3．将正方形的上顶点向上垂直拖动呈指针状；

4．填充指针部分再进行旋转复制即可生成星形图案。

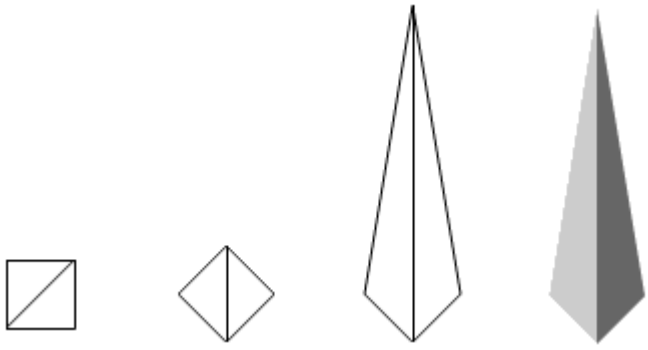

图 2.68　绘制星形图案示意图

3．创作并设计卡通人物，利用钢笔工具或者铅笔工具勾出卡通人物的基本轮廓，然后再利用颜料桶工具填充各个区域即可，如图 2.69 所示。

图 2.69　卡通人物绘制

提示：通常，设计人员先在白纸上进行手绘，再利用扫描仪将手绘图导入到 Flash 中，最后利用钢笔工具勾出轮廓进行颜色填充等工作。之所以采用手绘导入的方法，主要是由于手绘比较容易控制，不像鼠标绘图那样难以驾驭。当然，您也可以采用电脑的辅助设备——手写笔，基本类似于手绘，可以直接将图形输入到电脑。

第 3 章　动 画 类 型

动画制作都是基于时间轴形成的，本章在熟悉时间轴的同时对 Flash 动画进行了简单的分类，并穿插各类动画的经典范例进行详细介绍。通常，根据 Flash 动画制作方法的不同分为 5 大类型——逐帧动画、形变动画、运动动画、轨迹动画以及遮罩动画。

本章结合各类动画的理论知识和相应的简单案例制作，详细讲解上述 5 类动画，使您能够真正掌握各类 Flash 动画制作的全过程，为综合动画的设计和制作奠定扎实的基础。

通过本章的学习，应该达到如下目的：

- 了解 Flash 动画的基本知识
- 熟悉 Flash 动画的制作步骤
- 掌握 Flash 动画的快捷技巧

3.1　熟悉时间轴

所有的动画制作软件中，时间轴都是必不可少的。时间轴中记录了动画播放过程中的所有静态画面，类似于现实生活中观看电影时所用的胶片。当然，这种认识对于 Flash 时间轴的强大功能来说是相当肤浅的，并没有真正完整、全面地认识 Flash 动画中的时间轴。其实，在时间轴中记录着更多的数据信息，如图 3.1 所示。

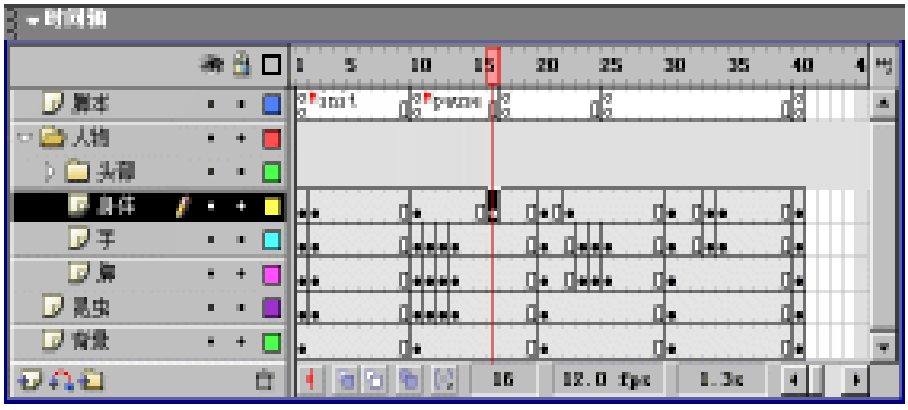

图 3.1　时间轴

注意：如果时间轴并没有显示在整个界面中，那么可以选择“查看”|“时间轴”命令，或者直接按快捷键 Ctrl+Alt+T 键显示时间轴；采用同样的方法可以隐藏时间轴。

时间轴主要分为图层编辑区和帧编辑区两个部分。图层编辑区位于时间轴左侧，在此可进行插入图层、删除图层、更改图层叠放次序等操作；而帧编辑区位于时间轴右侧，它是动画制作过程中最为重要的编辑区，所以帧操作在动画制作过程中是相当重要的。关于帧的所有操作将在动画创建的过程中讲解。

图层就像透明的醋酸纤维薄片一样，一层一层地向上叠加。图层可以帮助您组织文档中的各种对象。可以在图层上绘制和编辑对象，而不会影响其他图层上的对象。如果某个图层上没有任何内容，那么就可以透过它直接看到下面的图层。

新建文档后，在时间轴中就会包含一个图层。可以添加更多的图层，以便在文档中组织各种动画对象，图层数目的增加不会增大生成动画文件的大小。

另外，使用引导层可以使得运动物体沿着特定轨迹移动，而使用遮罩层可以让你创建出更为复杂的动画效果，相关内容将在本章其他小节进行介绍。

3.1.1 图层编辑

图层编辑包括一些比较常用的图层操作，如图层的选择、重命名、新建、复制以及删除等。现在就其中的各个操作进行具体讲解。

选择图层

要选择图层，可以进行下述的任一操作：

- 单击时间轴中的图层名称。
- 在时间轴中单击要选择的图层的任意一帧。
- 在舞台中选择要选择的图层上的任意一个对象。

要选择多个图层，可以进行下述操作：

- 选择连续的多个图层，按住 Shift 键单击时间轴中的图层名称。
- 选择不连续的多个图层，按住 Ctrl 键单击时间轴中的图层名称。

重命名图层

默认情况下，图层名称以“图层 1”、“图层 2”、“图层 3”命名，为了清楚地了解到相应图层中的大概内容，应该对图层名称进行具体的命名。

要对图层进行重命名，只要双击相应图层的图层名称，反白显示后输入新的图层名称即可。虽然重命名图层极为简单，但是对于日后动画的修改相当有益！

新建图层

要新建图层，可以进行下述的任一操作：

- 单击时间轴左下角的“插入图层”按钮。
- 选择“插入”|“图层”命令。
- 右击时间轴中的图层名称，选择弹出菜单中的“插入图层”命令。

复制图层

如果要创建一系列相似的图层，就可以进行复制图层操作，步骤如下：

- 单击相应的图层名称选择图层。
- 选择“编辑”|“拷贝帧”命令。
- 单击插入图层按钮新建图层。
- 选择“编辑”|“粘贴帧”命令。

删除图层

在动画制作过程中，难免会创建多余的图层，此时应该予以删除。值得注意的是，删除图层后所有包含的内容都将被删除。删除图层必须先选择图层，然后进行下述的任一操作：

- 单击时间轴底端的“删除图层”按钮。
- 将选定图层拖动到“删除图层”按钮。
- 右键单击选定的图层名称，选择弹出菜单中的“删除图层”命令。

3.1.2 图层状态

在动画创建过程中，为了消除动画中相互重叠对象之间的影响，通常需要隐藏相应的图层，以观看到该图层下被覆盖的动画对象。隐藏的图层或文件夹名称旁将显示红色的标志，但是在发布动画时将会自动取消图层的隐藏状态，以完整地显示整个动画内容。

如果要显示或隐藏图层，请执行以下任一操作：

- 单击图层名称右侧的“眼睛”列隐藏该图层，再次单击即可显示该图层。
- 单击“眼睛”图标可以隐藏所有的图层，再次单击就显示所有的图层。
- 在“眼睛”列中拖动可以显示或隐藏多个图层。
- 按住 Alt 键单击图层名称右侧的“眼睛”列可以隐藏所有其他的图层，按住 Alt 键再次单击可以显示所有的图层。

为了防止图层中对象的误修改，此时可以将相应的图层进行锁定；当对该图层中的对象进行调整时，再将该图层解除锁定即可。锁定或解除锁定的操作方法与显示或隐藏图层基本相似。

为了帮助区分各个图层中的对象，可以采用轮廓显示图层中的所有对象。当然，您可以更改各个图层所使用的轮廓颜色，相关内容将在随后的“图层属性”中进行讲解。轮廓显示或取消轮廓显示的操作方法与显示或隐藏图层基本相似。

图 3.2 所示的是图层或文件夹的各种状态——隐藏、锁定或轮廓显示。关于文件夹状态的设置基本和图层操作相同，所不同的是对文件夹设置状态时，该文件夹中的所有图层都将被设置为相应的状态。

提示：处于编辑状态的图层在时间轴图层名称的右侧显示铅笔图标。当图层锁定或隐藏时，就不能对该图层中的对象进行编辑，此时显示为带有红色禁止标识的铅笔图标。在 Flash 动画的创建过程中，应该时刻注意当前编辑的是哪个图层！

图 3.2　时间轴中的状态

3.1.3　图层属性

图层的各种状态也可以在“图层属性”对话框中进行设置。在该对话框中还可以更改图层的类型，如轨迹动画中的引导层和被引导层、遮罩动画中的遮罩层和被遮罩层，并且还可以控制图层高度。

如果要调用“图层属性”对话框，可以执行以下任一操作：

- 双击时间轴中图层名称左侧的图层图标。
- 右击图层名称，选择弹出菜单中的“属性”命令。
- 选中时间轴中的图层，选择“修改”|“图层”命令。

弹出的“图层属性”对话框如图 3.3 所示，尝试着更改该对话框中的各项参数，单击“确定”按钮后查看改变参数后所发生的变更。

图 3.3　“图层属性”对话框

3.2 逐帧动画

或许，您也曾经痴迷于动画王国之中，会如数家珍地搬出一些经典动画——《米老鼠与唐老鸭》、《宝莲灯》、《哪吒闹海》、《大闹天宫》等。其实，这些动画的制作基本上就是采用逐帧动画的方法，将大量的静态图片在时间轴中进行合成。由于相邻的图片之间的变化极其微小，而眼睛对于所看到的画面具有展示保留的特点。因此，快速且连续地切换这些图片就会观看到动画效果。

就逐帧动画来说，时间轴就是用于记载动画播放过程中的所有画面。为了了解时间轴与动画之间的关系，在 Flash 中导入 GIF 动画，观看 GIF 动画中各个画面在时间轴中的分布情况。具体步骤如下：

（1）按快捷键 Ctrl+N 新建文档，选择“文件”|“导入”命令，在弹出的“导入”对话框中选中 GIF 动画“KH”，单击“打开”按钮。此时，GIF 动画“KH”中的各个画面自动以关键帧的形式分布在时间轴上，如图 3.4 所示。拖动时间轴中红色的播放头，就可以观看到动画效果了。

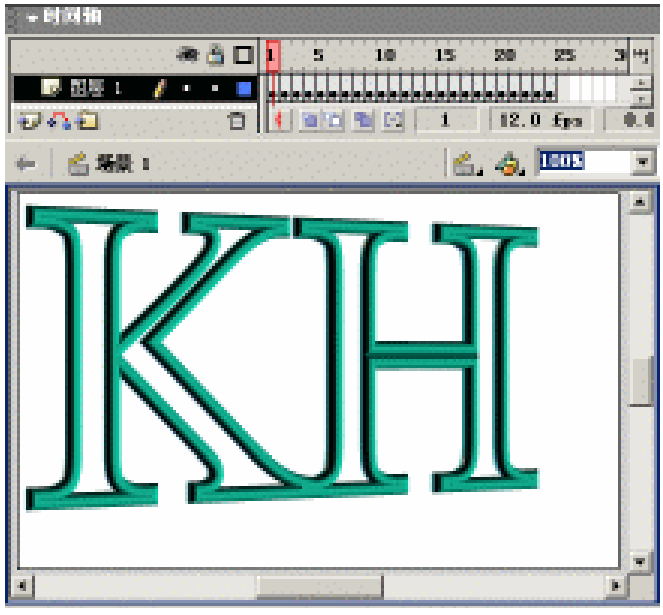

图 3.4 导入 GIF 动画后的时间轴显示

提示：关键帧就是在时间轴中记录动画画面变化的帧。由于关键帧记录着动画画面的变化，因此没有关键帧也就无法浏览动画效果，看到的只是一张静止的图片。在 Flash 动画制作过程中，关键帧所起的作用是至关重要的。它在时间轴中显示为圆点标识。当关键帧中没有内容时，该圆点为空心圆点，称为空白关键帧；而当关键帧中存在内容时，该圆点为黑色圆点。

（2）选择“修改”|“文档”命令，弹出“文档属性”对话框，设置尺寸为 400px×250px，

如图 3.5 所示，将舞台尺寸调整为与文字“ KH ”尺寸接近，单击“ 确定 ”按钮确认。

（3）单击时间轴右上角的按钮，弹出时间轴下拉菜单如图 3.6 所示，选择其中的“ 预览 ”命令。

图 3.5　“ 文档属性 ”对话框

图 3.6　时间轴下拉菜单

（4）时间轴显示发生较大变化，各个关键帧不再以黑色圆点作为标识，取而代之的是 GIF 动画“ KH ”中的各个画面，如图 3.7 所示。

图 3.7　预览状态下的时间轴显示

预览状态下的时间轴是不是有点像电影胶片？由此可见，逐帧动画的制作类似于电影的拍摄，所不同的是 Flash 中逐帧动画的制作基本是由设计人员手绘而成，而电影的拍摄是利用摄影机记录而成。

因此，逐帧动画的制作是非常费时费力的艰苦工作，但是利用该方法可以制作出极其精美的动画效果。当然，这要求您有扎实的绘画功底。

3.3　形 变 动 画

Flash 动画创建过程中，形变动画是较为常用的创作方法，其制作得到的动画效果通常表现为由一种图形逐渐转换为另一种图形。形变动画之所以较为常用，主要是由于其制作过程相对于逐帧动画来说要简单得多，最大程度地缩短了动画的创作时间。

3.3.1　形变动画原则

创建形变动画时，只要确定首、末两个关键帧中的图形即可，而不必像逐帧动画那样在每个关键帧中绘制图形。随后再利用“ 形状补间 ”在首、末关键帧之间自动生成图形转

换的中间帧，从而缩短了大量的制作时间。

但是，放置在形变动画首、末关键帧中的对象是有条件的——必须是形状，这也是形变动画创建成功与否的必要条件。所以，创建形变动画前应该先对形变动画首、末关键帧中的对象进行检验，确定其中所放置的对象是形状。

通常有效的简便方法就是单击形变动画的关键帧，选中该关键帧中的所有对象后，查看其选中状态，如果呈现点状显示则表明该关键帧中的对象为形状，如图 3.8 所示；此时再在选中的对象上单击，如果选中的对象为形状时，“ 属性 ”面板中将会给出提示——文字“ 形状 ”及相应的图标，如图 3.9 所示。

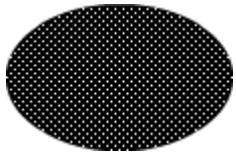

图 3.8 选中呈点状显示

图 3.9 “ 属性 ”面板显示

提示：如果确定形变动画首、末关键帧中的对象不是形状，此时可以选择“ 修改 ”|“ 分离 ”命令，使得关键帧中的对象分离为形状。有时需要经过多次分离操作，才能使关键中的对象完全成为形状。

3.3.2 基本形变动画

清楚形变动画原则后，现在就熟悉一下形变动画创建的基本步骤。概括来说，形变动画的制作只要三步就能够完成：（1）确定形变动画的首、末关键帧，（2）检验关键帧中的对象，（3）创建形变动画。

然而，在形变动画的实际制作过程中通常忽略第 2 步操作，因为确定关键帧时已经同时完成了关键帧中对象的检验。

为了清楚形变动画的制作过程，在下面的范例动画中采用最简单的图形，整个动画表现为由圆变为方形，再变为三角形，最后变回圆的过程，如图 3.10 所示。

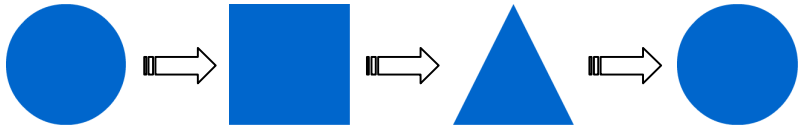

图 3.10 形变动画中各关键帧的图形

具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建文件，按快捷键 Ctrl+J 调用“ 文档属性 ”对话框，调整尺寸为 300px×300px，单击“ 确定 ”按钮确认。

（2）按 O 键选择椭圆工具，在舞台中央绘制蓝色填充的圆形，如图 3.11 所示；然后在第 10 帧处按 F7 键插入空白关键帧，按 R 键切换到矩形工具，在舞台中央绘制蓝色填充的正方形，如图 3.12 所示。

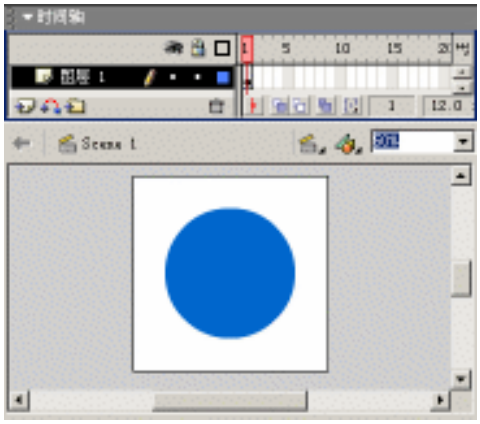

图 3.11　绘制圆形

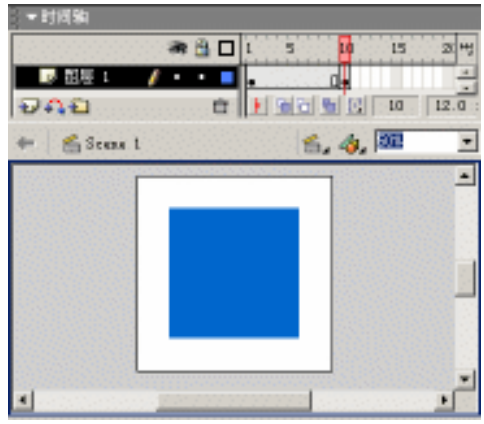

图 3.12　绘制正方形

（3）单击首、末关键帧间的任意一帧，此时“ 属性 ”面板中显示帧的相关设置，选择“ 补间 ”列表框中的“ 形状渐变 ”，如图 3.13 所示。这样，形变动画就制作完成了。此时，时间轴中显示带有箭头的直线段，并且舞台上的图形也发生相应的变化，如图 3.14 所示。

提示：当“ 补间 ”设置为“ 形状渐变 ”时，“ 简易 ”和“ 混合 ”属性也被激活。“ 简易 ”属性可以控制形状变形的快慢，也就是说变形是先快后慢还是先慢后快；“ 混合 ”属性可以调整形变动画的变形方式，不管是“ 分布式 ”还是“ 角形 ”，其过渡帧都是自动生成的。

图 3.13　设置形变动画

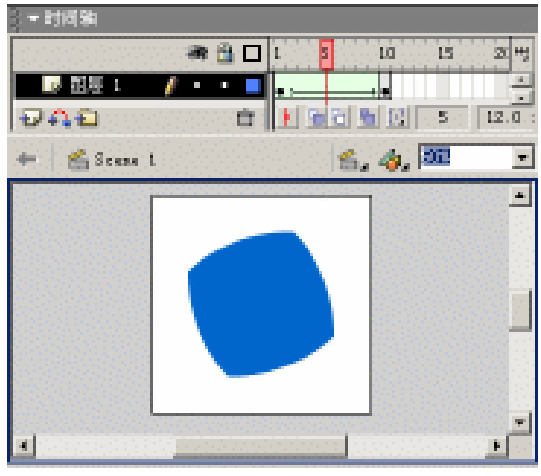

图 3.14　形变动画时间轴

（4）按 Enter 键在场景编辑状态下测试动画播放情况，此时会看到一段由圆形向正方形变形的动画，也就是说形变动画制作成功。下面就继续变形动画的其余部分制作……

（5）在第 16 帧处按 F6 键插入关键帧，使正方形在舞台上持续一段时间；在第 25 帧处按 F6 键插入关键帧，利用箭头工具将正方形调整为三角形，完成正方形变为三角形的首、末关键帧的确定；在“属性”面板中选择补间方式为“形状渐变”。

（6）在第 31 帧处按 F6 键插入关键帧，使三角形在舞台上停留一段时间；为了使动画循环变形，在第 1 帧上右击，选择弹出菜单中的“拷贝帧”命令复制该关键帧，然后在第 40 帧处右击，选择弹出菜单中的“粘贴帧”命令粘贴刚才复制的关键帧，并使之延续到第 45 帧；同样的方法创建第 31 帧到第 40 帧之间的形变动画，如图 3.15 所示。

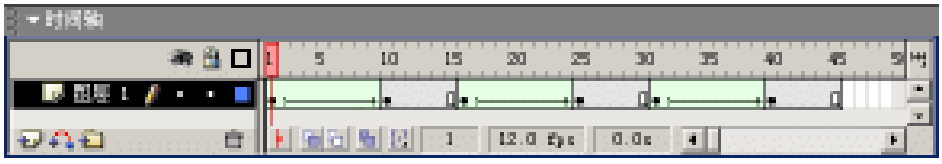

图 3.15　创建多个形变

至此，就可以看到圆形→正方形→三角形→圆形之间循环变形的形变动画。按快捷键 Ctrl+Enter 测试动画，欣赏一下形变动画带来的魅力吧！！

思考：如果测试动画时，并没有发现关键帧图形间的变形过程，也就是说形变动画没有制作成功，那么问题会出现在哪些环节呢？

3.3.3　干涉形变动画

上一节制作的形变动画中，完全自动生成过渡画面，基本上没有太多的人为因素干涉。但是有些形变动画如果没有人为控制，过渡帧中的画面几乎是乱作一团，根本不能达到设计意图。下面就尝试制作三棱锥围绕其中轴线进行转动的动画效果，为了更加清楚动画的制作过程，先完成线框三棱锥的转动，然后再对三棱锥进行渐变填充，使之更加真实。具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 400px×300px。选择“查看”|“网格”|“编辑网格”命令，弹出“网格”对话框，选中“显示网格”和“对齐网格”复选框，同时设置网格尺寸为 20px×20px，如图 3.16 所示。单击“确定”按钮确认后，舞台上显示灰色网格。

（2）按 N 键切换到线条工具，在“属性”面板中设置笔触颜色为红色（#FF0000）、笔触类型为“极细”，在舞台上绘制等腰三角形，并以右侧腰为边绘制一窄条三角形，如图 3.17 所示。

（3）在第 24 帧处按 F6 键插入关键帧，此时关键帧中的三棱锥线条均为选中状态，选择“修改”|“变形”|“水平翻转”命令翻转图形，最后创建两关键帧间的形变动画，如图 3.18 所示。

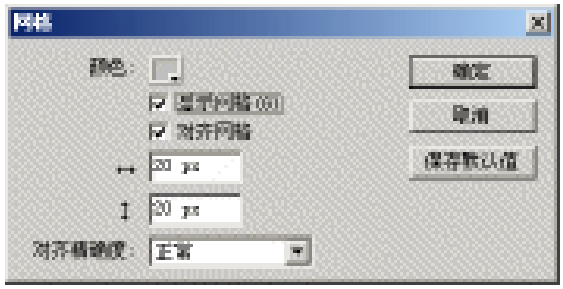

图 3.16　“网格”对话框

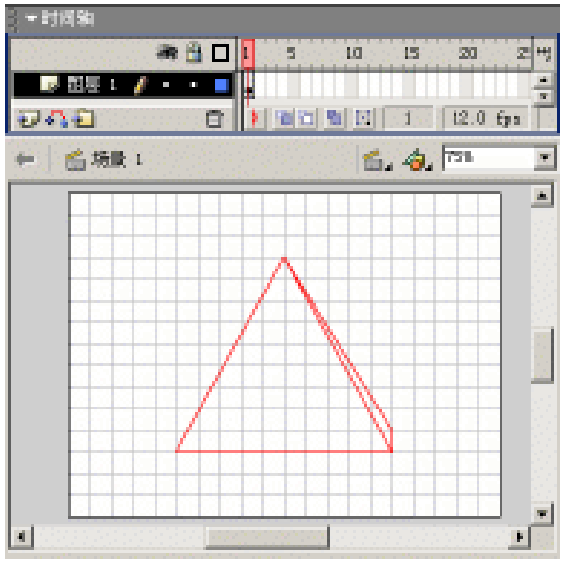

图 3.17　第 1 帧

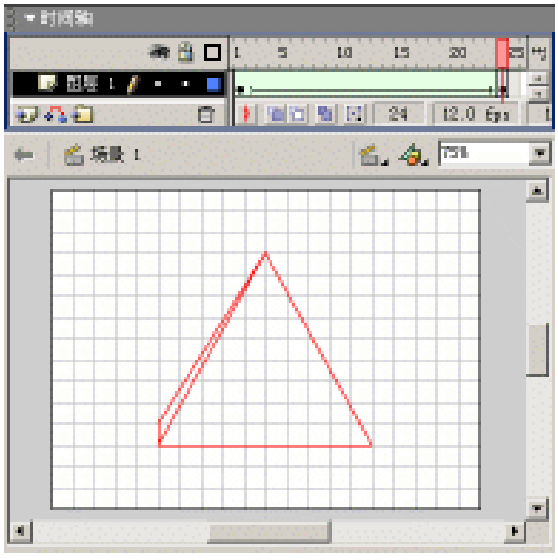

图 3.18　第 24 帧

（4）按 Enter 键在场景编辑状态中测试动画，结果是大失所望——三棱锥的所有线条都像散了架似的乱作一团，根本没有预期绕中心轴转动的一丝踪迹。其实，上述步骤只是本例动画创建的部分内容，主要是奠定基本框架，而下面的步骤才是最为重要的。

（5）切换到箭头工具后单击第 1 帧，选择“修改”|“形状”|“添加形状提示”命令，此时在舞台上出现标识为“a”的小圆点，也就是所谓的“形状提示”。连续按快捷键 Ctrl+Shift+H 四次，再增加 4 个形状提示，将这 5 个形状提示分别放置在三棱锥的 4 个顶点以及中间那条边上，具体分布如图 3.19 所示。

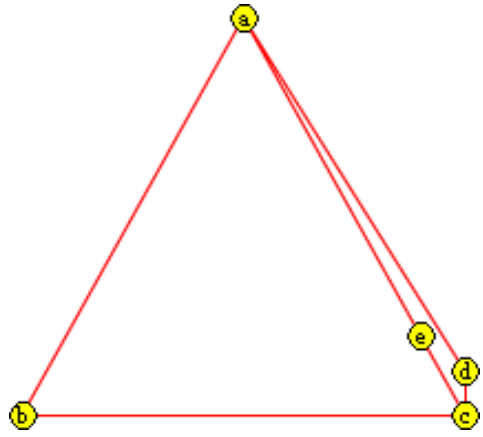

图 3.19　第 1 帧形状提示

注意：形状提示的添加是以形变动画的成功创建为前提的。因此，如果没有完成形变动画的创建，那么“添加形状提示”命令就会呈现灰色状态，表示该命令当前状态是不可用的！！

（6）单击第 24 帧，此时同样可以发现舞台上显示着 5 个形状提示，与第 1 帧中的形状提示一一对应地布置该关键帧中的形状提示，如图 3.20 所示。这样，三棱锥绕其中轴线转动的动画基本完成，按 Enter 键在场景编辑状态中测试，已经没有原先那杂乱的踪迹了！

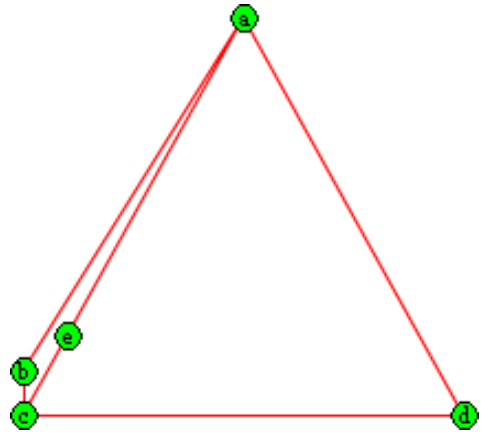

图 3.20　第 24 帧形状提示

（7）现在应该对三棱锥面进行填充，以便更加真实地体现三棱锥的转动。为了体现转动过程中光的变化，应该使用线性渐变填充。切换到颜料桶工具，选择“窗口”|“混色器”命令调用“混色器”面板，在其中设置填充样式为“线性”，同时调整渐变条上的两个色块，如图 3.21 所示。

图 3.21　设置线性渐变

（8）单击第 1 帧，利用颜料桶工具填充三棱锥的面后，再使用填充变形工具对三棱锥面的填充效果进行调整，最后删除起分割作用的所有线条，如图 3.22 所示。同样，对第 24 帧中三棱锥也进行填充等操作。

图 3.22　填充三棱锥面

至此，整个动画就全部制作完成，按快捷键 Ctrl+Enter 测试动画，效果不错吧！整个动画中涉及到的“添加形状提示”功能是形变动画所特有的，利用该功能控制形状按照预先构思的方式进行变形需要不断地尝试和探索！

提示：就线性渐变而言，或许你只知道颜料桶工具的填充功能，但是并不知道颜料桶工具也可以调整线性填充的方向，如图 3.23 所示；如果需要对图形填充进一步调整，此时可以选择填充变形工具，在具有线性渐变的图形填充上单击，

就会显示线性渐变的 3 个控制句柄，利用这 3 个句柄就可以对图形进行任意调整，如图 3.24 所示。

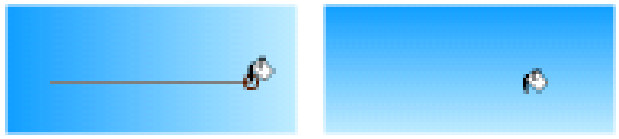

图 3.23　颜料桶工具的使用

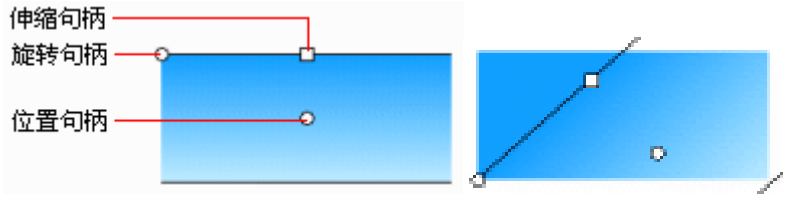

图 3.24　填充变形工具的使用

3.4　运 动 动 画

日常生活中会遇到大量的运动物体，如汽车的移动、足球的滚动等。在 Flash 中所涉及的运动动画不仅包括物体的位移变化，而且还包括物体的尺寸缩放、颜色交替等。运动动画的应用可以最大程度地简化动画制作步骤。当然，动画的制作并不只依靠这种动画制作方法，但是运动动画是 Flash 制作动画中使用最为频繁的。

3.4.1　运动动画原则

就运动动画而言，不管运动的过程中实物发生如何复杂的变化，但是其最根本的内在东西是不变的。其实，运动动画的创建与形变动画基本相似，但是创建运动动画时，必须确定其首、末关键帧中的对象都是实例。所谓实例，就是为了减小动画的文件容量以及方便动画的修改而创建元件，并将之在动画制作中加以引用。

对实例、元件等概念可能比较陌生，但是在运动动画的创建过程中不可避免地要涉及这些概念，本章重点在于熟悉运动动画的制作步骤，有关元件、实例等概念及两者间的联系，将在第 4 章中重点讲解。

在继续学习下面的内容前，再次强调——创建运动动画的首、末关键帧中的对象一定都是实例。而检验的简便方法就是单击运动动画的关键帧，选中该关键帧中的对象后，查看其选中状态，如果呈现框架显示同时具有十字线⊕标志，则表明该关键帧中的对象为实例，如图 3.25 所示；此时再在选中的对象上单击，如果选中的对象为实例时，“ 属性 ” 面板中将会给出相应的提示 “ 实例 :”，而左端显示为列表框，如图 3.26 所示。

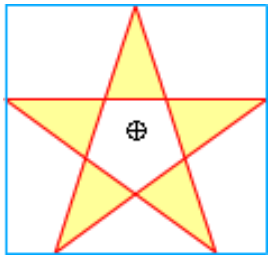

图 3.25　实例选中状态

图 3.26　利用“属性”面板判断实例

3.4.2　基本运动动画

如果您对上面的运动动画原则还是有些不解，那么就以小球弹动的动画实例进行讲解吧！在您清楚运动动画制作步骤的同时，应该与形变动画的制作进行比较，以便更加明确这两大动画类型制作过程中的异同点。

具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，选择椭圆工具，取消线条颜色、设置填充颜色为黑白放射状渐变，如图 3.27 所示；在舞台上绘制圆形，再利用颜料桶工具改变圆形填充的高光点位置，以更加清楚地体现球体的光泽感，如图 3.28 所示。

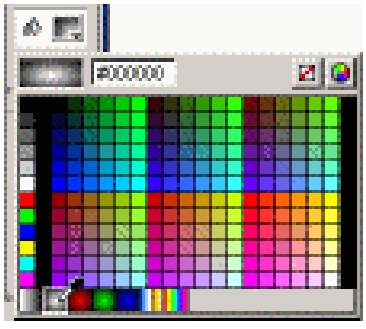

图 3.27　设置绘制颜色

图 3.28　设置绘制颜色

提示：就放射状渐变而言，使用颜料桶工具在图形填充中拖动可以调整放射状渐变的中心位置，如图 3.29 所示；如果需要进一步调整具有放射状渐变的图形填充，可以选择填充变形工具，然后在图形填充上单击，会出现放射状渐变的 4 个控制句柄，这时就可以任意调整，如图 3.30 所示。

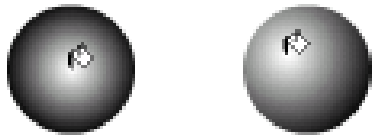

图 3.29　颜料桶工具的使用

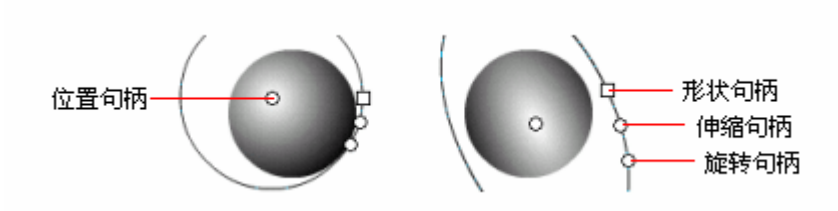

图 3.30　填充变形工具的使用

（2）使用箭头工具选定圆形填充，选择“插入”|“转换为元件”命令调用“转换为元件”对话框，输入元件名称为“Ball”，选中行为为“图形”单选按钮，如图 3.31 所示。单击“确定”按钮确认，此时发现选中的圆形不再是点状显示，而是框架显示，同时中心位置呈现十字线⊕标志，如图 3.32 所示。

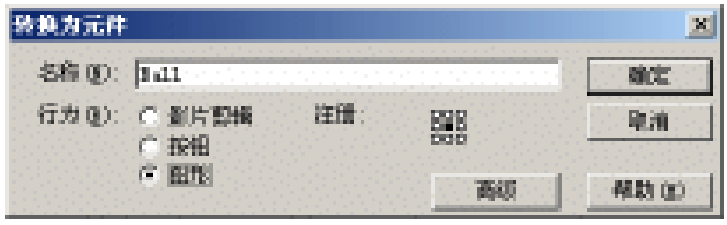

图 3.31　转换为符号

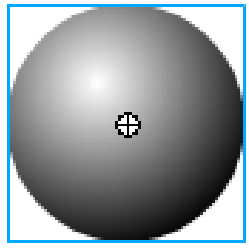

图 3.32　舞台上实例

（3）将球体置于舞台的左上角；在第 15 帧处按 F6 键插入关键帧，将球体置于舞台的中下方；在第 30 帧处按 F6 键插入关键帧，将球体置于舞台的右上角。为了使您清楚 3 个关键帧中的球体位置，单击时间轴下方的“编辑多个帧”按钮，再在显示标记中选择“绘制全部”项，时间轴及舞台显示如图 3.33 所示。

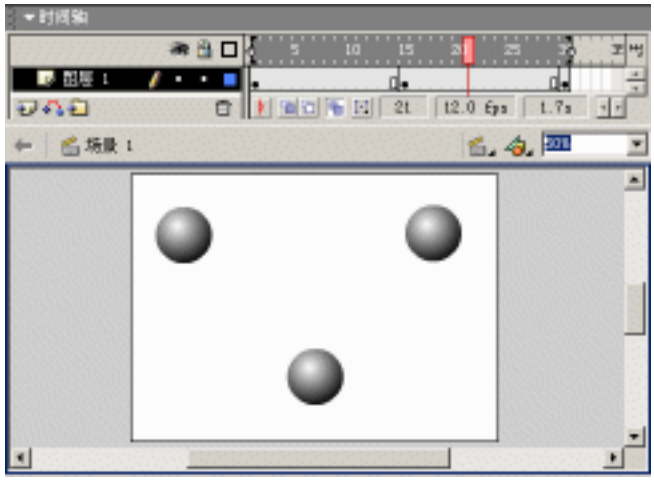

图 3.33　各关键帧中小球位置

（4）完成关键帧的创建后，在第 1 帧处右击，选择弹出菜单中的“ 创建补间动画 ”命令，此时在第 1 帧到第 15 帧之间的时间轴显示出现直线的方向箭头；同样，在第 15 帧处单击右键，选择弹出菜单中的“ 创建补间动画 ”命令。

为了使读者了解球体的运动方向，同时选中时间轴下方的“ 绘图纸外观轮廓 ”按钮、“ 编辑多个帧 ”按钮，并在显示标记中选择“ 绘制全部 ”项，此时可以看到小球由舞台的左上角斜向下落后又斜向弹起的运动过程，如图 3.34 所示。

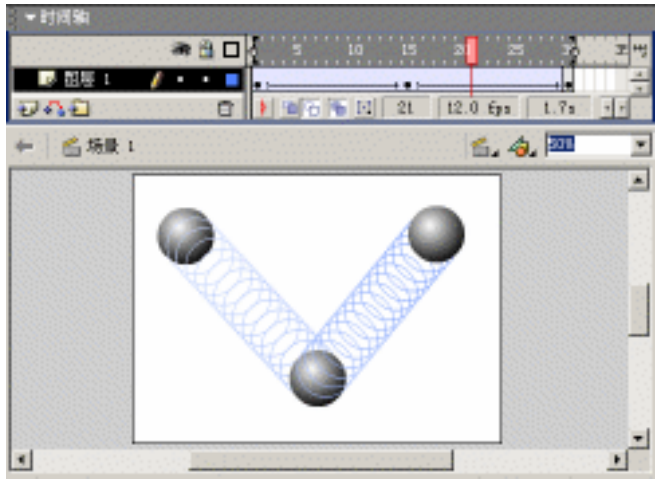

图 3.34　创建运动动画

至此，球体弹动的运动动画就制作完成，相信您已掌握并清楚运动动画的制作过程。按快捷键 Ctrl+Enter 测试动画，看看球体的运动方向，并感觉一下动画制作的乐趣吧！

3.4.3　设置运动动画

本例的动画效果是指针在罗盘上绕着罗盘圆心顺时针旋转。具体的制作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 300px×300px、帧频为 6fps；将层名称更改为“罗盘”，在舞台上绘制颜色相间的圆形，并在第 13 帧处按 F5 键插入帧进行画面延迟，如图 3.35 所示。

（2）插入图层“指针”，在舞台上绘制如图 3.36 所示的指针。

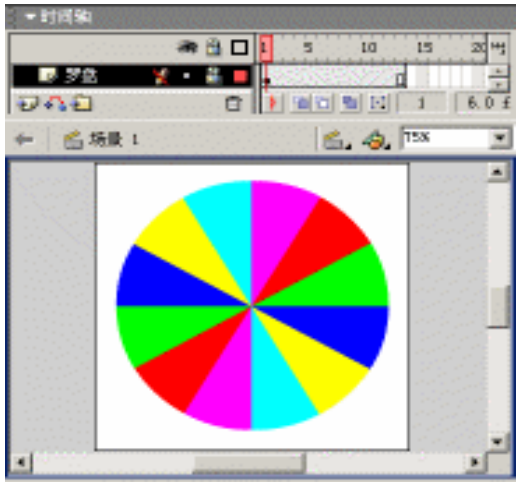

图 3.35 “罗盘”图层

图 3.36 绘制指针

（3）单击“指针”图层中的关键帧，选中整个指针图形后按 F8 键调用“转换为元件”对话框，在“名称”文本框中输入元件名称为“指针”，选中行为为“图形”单选按钮，同时设置注册为中下点，如图 3.37 所示。单击“确定”按钮后，可以发现指针呈框架显示，同时在其底端呈现十字线⊕标志，如图 3.38 所示。

图 3.37 “转换为元件”对话框

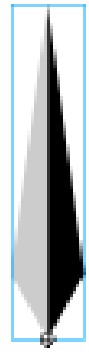

图 3.38 转换后的指针

（4）拖动“指针”十字线标志到罗盘中心位置处，在第 13 帧处按 F6 键插入关键帧，创建两个关键帧间的运动动画。此时，如果按 Enter 键在场景编辑状态下测试动画，只能看到静止的指针和罗盘，完全没有一点动画的效果。

（5）单击“指针”图层的第 1 帧，此时“属性”面板中显示帧的相关设置，“补间”列表框中为“运动渐变”，这就是创建运动动画必要的设置。选择“旋转”列表框中的“顺

时针”，同时在右侧激活的文本框中输入数值 1，如图 3.39 所示。所有这些操作表示指针绕着其十字线顺时针旋转 1 圈。

图 3.39　设置旋转运动

注意：设置运动动画时，“简易”属性调整的是运动的加速度，动画测试时可以看出物体移动的速度变化；“旋转”属性可以选择旋转方向是“顺时针”、“逆时针”、“自动”或“无”。这些属性的设置可以自行尝试，并进行总结，以提高对软件学习的积极性！！

至此，指针在罗盘上转动的动画效果就制作完成了，按快捷键 Ctrl+Enter 进行动画测试，浏览动画效果如何？

3.5　轨 迹 动 画

现实生活中，直线运动或原地转动毕竟是少数，而大部分物体的运动还是具有一定的轨迹，或者是曲线，或者是折线。但是，不管物体运动的线路多么复杂，只要能够描绘其运动轨迹，在 Flash 中都可以精确地沿着该轨迹进行移动。

轨迹动画就是针对运动线路比较复杂的物体运动产生的。同样还是先来了解一下创建轨迹动画的基本原则。

3.5.1　轨迹动画原则

其实，轨迹动画是运动动画的扩展，两者之间的区别就是轨迹动画中的物体沿着一定的线路进行移动。因此，在创建轨迹动画的过程中必然会完成运动动画的制作，也就是说，轨迹动画的制作通常需要两个图层才能得以完成。其中一个图层是轨迹层，用于确定运动物体的移动线路；而另一个图层就是被引导层，用于制作运动物体的运动动画。

需要注意的是，在创建轨迹动画时，这两个图层之间必须相互关联，时间轴中图层之间的具体关系显示如图 3.40 所示。

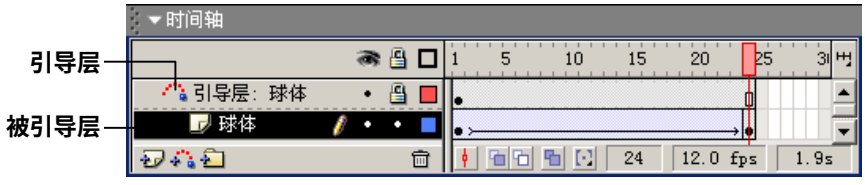

图 3.40　轨迹动画的图层关系

3.5.2 基本轨迹动画

下面通过制作球体沿着弧线滚动的动画效果，来检验轨迹动画的基本知识，同时掌握轨迹动画的制作步骤，如下所示：

（1）按快捷键 Ctrl+N 新建文档，将图层名称更改为“球体”，利用上节所学的内容在第 1 帧～第 24 帧制作一段球体移动的运动动画，如图 3.41 所示。

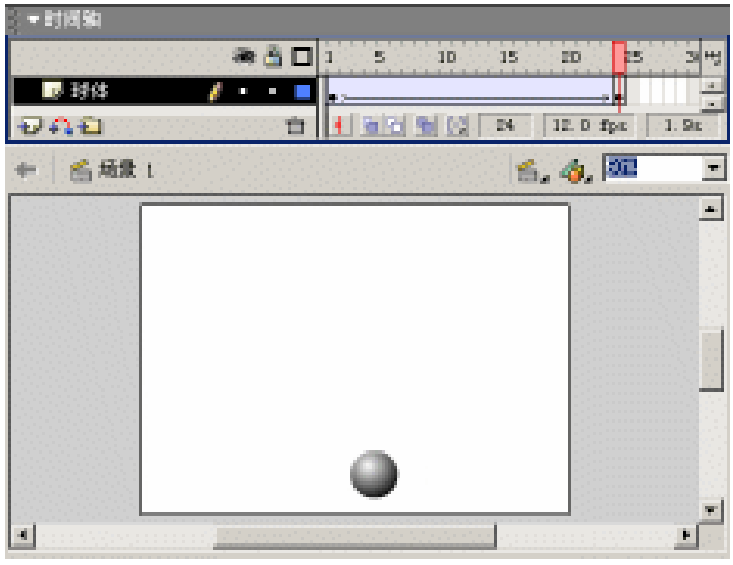

图 3.41 创建运动动画

（2）单击时间轴左下角的“添加运动引导层”按钮，时间轴中的“球体”图层上将显示“引导层：球体”图层，并且“球体”图层与之关联，如图 3.42 所示。

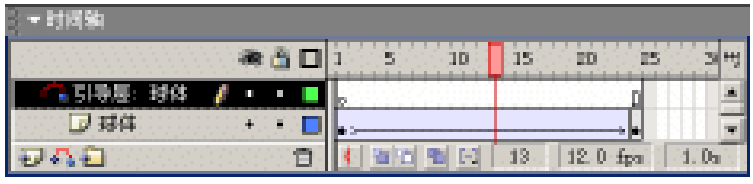

图 3.42 添加引导层

（3）利用铅笔工具或按钢笔工具在“引导层：球体”图层的舞台上绘制一条弧线后，锁定该图层，防止以后的操作引起弧线变形。

（4）单击第 1 帧，拖动舞台上的十字线标志吸附到弧线的一个端点；单击第 24 帧，拖动舞台上的十字线标志吸附到弧线的另一个端点，如图 3.43 所示。

至此，球体的弧线移动就制作完成了，按快捷键 Ctrl+Enter 测试动画。

3.5.3 高级轨迹动画

轨迹动画的制作步骤基本已经了解，可以看出轨迹动画其实就是运动动画的扩展。通

常，轨迹动画是由一个引导层和一个被引导层构成的。如果在创建的多个轨迹动画中，所有引导层中轨迹都是相同的曲线，那么就没有必要采用多个引导层。因为，单个引导层可以同时与多个被引导层关联。

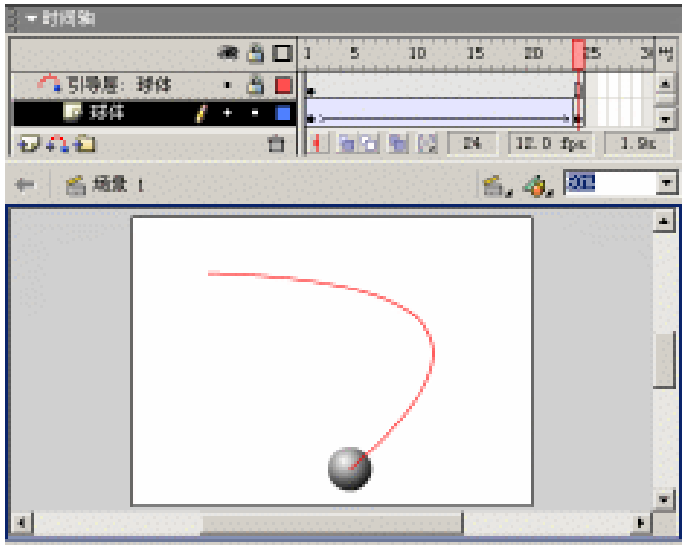

图 3.43　确定球体运动轨迹

下面就制作两个模型沿着同一段弧线移动的动画，但是这两个模型的移动方向却是相反的。步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 500px×400px。将图层名称更改为“引导层”，在舞台上绘制椭圆环，并使之倾斜一定角度。利用箭头工具框选椭圆环的一小部分，按 Delete 键将之删除，使得椭圆环变为非封闭路径，使之延续到第 34 帧，最后锁定该图层，如图 3.44 所示。

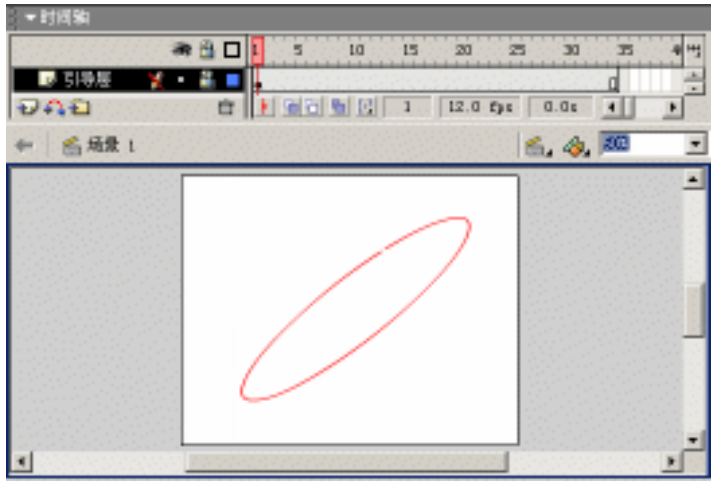

图 3.44　“引导层”图层

（2）插入图层“模型 1”，并将该图层拖至“引导层”的下方，在舞台上绘制模型并将之转换为元件“模型 1”，在第 34 帧处插入关键帧，最后创建两关键帧间的运动动画，如图 3.45 所示。

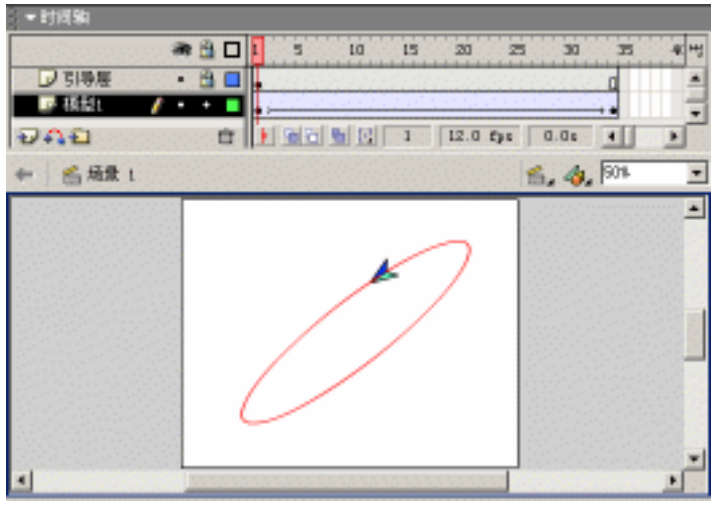

图 3.45 “模型 1”图层

（3）右击“引导层”的图层标志，选择弹出菜单中的“引导层”命令，将“引导层”由正常层转换为引导层，而图层标志也变为锤子图案，如图 3.46 所示。此时，“引导层”下的“模型 1”并没有与之关联。

提示：轨迹动画制作过程中，如果只有引导层而没有被引导层，那么引导层的图层标志为锤子图案。其实，没有关联的引导层可以作为辅助线使用，其功能比辅助线更强。引导层中的对象可以是斜线，也可以是弧线，还可以是一些特殊的形状。生成 SWF 动画时，引导层中的对象是看不到的。

图 3.46 没有关联的引导层

（4）双击“引导层”的图层标志，弹出“图层属性”对话框，在“类型”中选中“被引导”单选按钮，如图 3.47 所示，单击“确定”按钮确认后，可以发现“模型 1”图层与“引导层”建立了关联关系。

（5）适当调整“模型 1”图层中首、末关键帧中的模型，拖动它们使之吸附到椭圆环小缺口的两端，同时适当调整角度使得模型适应椭圆环。插入图层“模型 2”，与“模型 1”制作方法相同，所不同的是 2 个模型移动的方向相反而已！最终完成的时间轴显示和舞台对象如图 3.48 所示。

图 3.47　设置图层属性

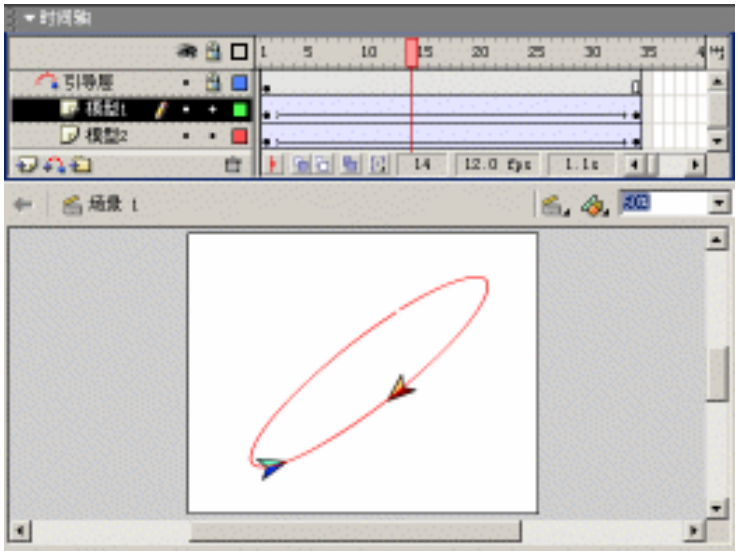

图 3.48　多个被引导层

按快捷键 Ctrl+Enter 测试动画，效果不错吧！！

思考：如果在引导层中创建多条运动轨迹，是否可以使多个被引导层中物体沿着不同的轨迹进行移动呢？如果可能，那么在放置运动层的实例对象时需要注意些什么问题呢？

3.6　遮 罩 动 画

浏览动画时，经常看到类似于探照灯、放大镜的动画效果。如果采用一般的方法就难以实现，幸亏 Flash 提供了专门针对上述效果的遮罩技术。

3.6.1 遮罩动画原则

遮罩就其本质来说就是确定显示范围。遮罩动画的制作类似于轨迹动画，至少需要两个图层才能达到探照灯等效果。通常处于上面的图层为遮罩层，而下面的图层为被遮罩层。只有当这两个图层相互关联、共同作用，才会生成遮罩效果。

遮罩层中的对象就是显示区域，显示被遮罩层中的对象。通过下面的示例，可加深对遮罩技术的理解。图 3.49 为创建遮罩前的显示效果，其中“图像”图层中放置了一张人物图像，而“形状”图层中绘制了一个圆角矩形，圆角矩形遮挡了其下面的部分人物图像；图 3.50 为创建遮罩后的显示效果，同样的图层内容却只显示了圆角矩形遮挡住的部分人物图像。

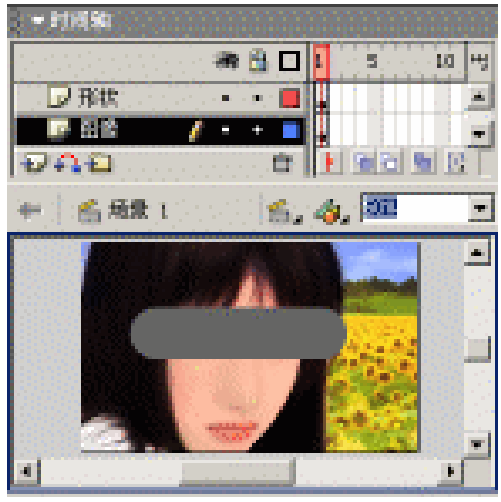

图 3.49 创建遮罩前

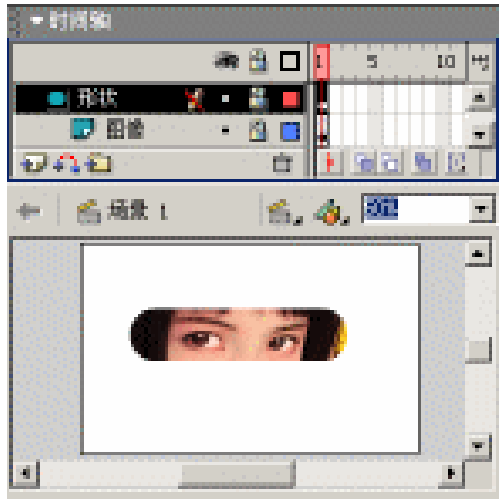

图 3.50 创建遮罩后

3.6.2　基本遮罩动画

了解遮罩技术的显示原则后，现在就实践制作一下探照灯效果。操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 400px×250px。将图层名称更改为“暗图”，选择“文件”|“导入”命令，在弹出的“导入”对话框中导入“人物.JPG”图像；选中人物图像后按 F8 键将之转换为元件“图像”，并在“属性”面板中设置“亮度”为-50%进行暗化，效果如图 3.51 所示，最后将之延长至第 25 帧。

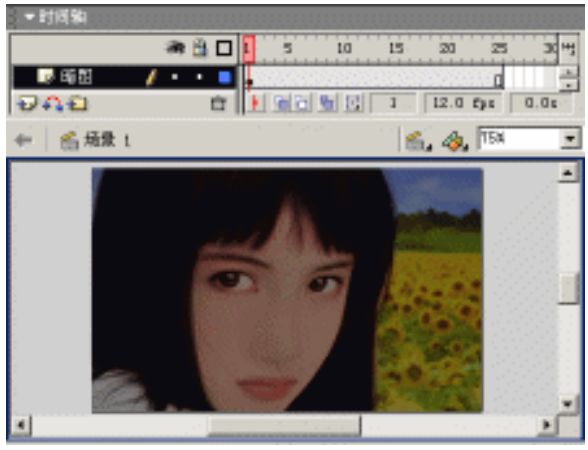

图 3.51　“暗图”图层

（2）右击“暗图”图层的关键帧，选择弹出菜单中的“拷贝帧”命令；插入图层“明图”后在第 1 帧处右击，选择弹出菜单中的“粘贴帧”命令；选中“明图”图层中的人物图像，同样在“属性”面板中设置亮度为 25%进行亮化，效果如图 3.52 所示。这样，图层“明图”、“暗图”中的人物图像是完全重合的。

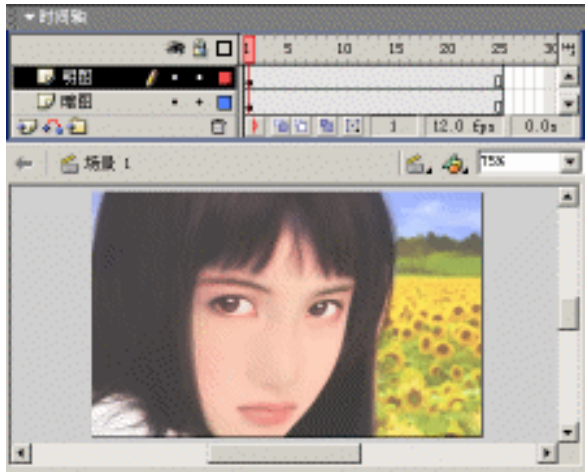

图 3.52　“明图”图层

思考：在动画制作过程中，图层“明图”、“暗图”中的人物图像也可以利用其他图像软件，如 Adobe PhotoShop、Macromedia Fireworks 等制作生成两张图像，再导入到 Flash 文档中。那么，在动画制作中为什么利用“属性”面板中的亮度设置具有图像的元件？

（3）插入图层“形变”，在该图层中可以根据个人喜好创建一段形变动画。然后在图层标志上右击，选择弹出菜单中的“遮罩层”命令创建遮罩，可以发现时间轴中的图层显示和舞台上的屏幕效果发生相应的变化，如图 3.53 所示。

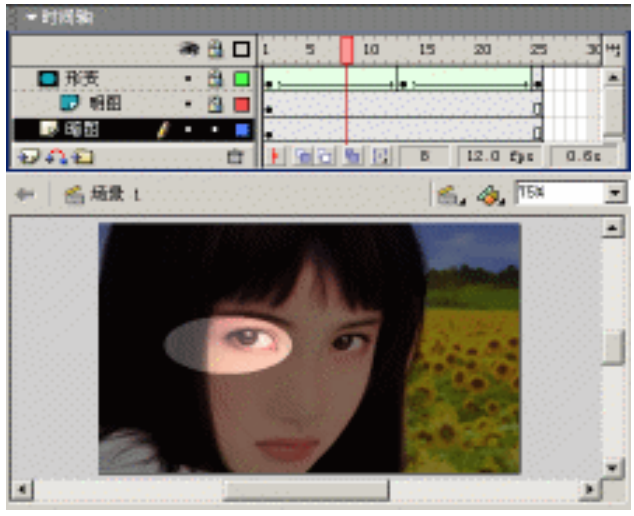

图 3.53 探照灯效果

至此，探照灯效果的制作完毕，按快捷键 Ctrl+Enter 测试动画。

3.6.3 高级遮罩动画

通过探照灯效果的制作，基本了解了遮罩动画的创建过程，也就更加明确遮罩效果的实现至少需要两个图层。与轨迹动画的制作一样，遮罩动画也同样可以实现单个遮罩层和多个被遮罩层之间的关联。步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 450px×200px。将图层名称更改为“背景”，按快捷键 Ctrl+R 导入图像“晚霞.JPG”，使该帧延续到 29 帧并锁定该图层。插入图层“形状”，利用文本工具、矩形工具等制作具有边框的文本，如图 3.54 所示。

提示：在“形状”图层中的文本必须进行分离处理，否则最后制作得到的遮罩效果将不会显示其中的文本内容，而只能看到外部的边框部分。究其原因，就是因为“形状”图层中的边框与文本不是处于同一级别。只要选中“形状”图层中的全部对象，按快捷键 Ctrl+B 进行分离处理即可。所谓分离，就是将层叠对象（呈框架显示）破碎成为舞台级对象（矢量图形，呈点状显示）。

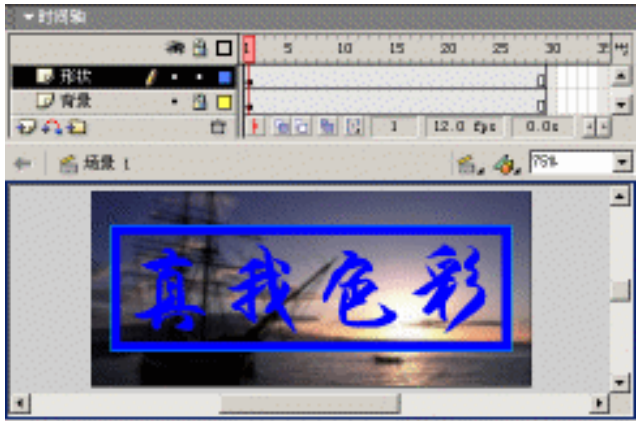

图 3.54　舞台布置

（2）插入图层“色谱 1”，在舞台上绘制色谱线性渐变的矩形，选中矩形后将之转换为元件“色谱”，分别在第 15、29 帧处按 F6 键插入关键帧，随后在各个关键帧之间创建运动动画。将“色谱 1”图层拖至“形状”图层的下方，右键单击“形状”的图层标志，选择弹出菜单中的“遮罩层”命令创建遮罩，如图 3.55 所示。

图 3.55　创建基本遮罩

提示：绘制色谱矩形时，该矩形的尺寸要略大于“形状”图层的边框，并应该参考光盘中提供的源文件，控制“色谱 1”图层各关键帧中色谱矩形的坐标位置。

（3）单击“色谱 1”图层后，插入图层“色谱 2”、“色谱 3”，复制“色谱 1”图层中的第 15 帧～第 29 帧到“色谱 2”图层的第 1 帧；复制“色谱 1”图层中的第 1 帧～第 15 帧到“色谱 3”图层的第 15 帧，最终的时间轴布置和舞台显示如图 3.56 所示。

至此，循环交替且没有间断和停顿感的色谱显示动画就制作完毕，按快捷键 Ctrl+Enter 测试动画。

图 3.56 创建高级遮罩

3.7 小结与练习

本章详细讲解了动画制作中涉及的图层操作，并将所有的动画制作方法精练为 5 大类动画——逐帧动画、形变动画、运动动画、轨迹动画和遮罩动画，目的就是使您对 Flash 动画的制作有一个较为具体、全面的认识。

通过本章的学习，希望能够完成以下练习：

1.将本章列举的所有实例按部就班地制作一遍,从中体会这 5 类动画制作的基本步骤。当您清楚这些制作规则后，可以适当扩展以力求创新和个性！

2．综合应用这 5 类动画的制作方法，设计并制作一段动画。然后再将您的作品 Mail 给您的朋友，分享一下 Flash 动画创作的乐趣！记住，只有当理论知识应用到实践中时，才能真正得到本质的突破！

3．发挥您的创意和构思，结合本章所学到的动画制作方法，利用 Flash 表达您对世间百态的理解。记住，Flash 动画制作时，创意才是整个动画的灵魂所在、魅力所在！

第 4 章　动 画 元 件

在第 3 章中介绍运动动画时，初步接触了有关动画元件的概念。为了强调动画元件在 Flash 动画创建中的重要性，特别将动画元件在本章中进行讲解。

通过本章的学习，应该达到如下目的：

- 清楚动画元件的分类
- 了解“ 库 ”面板的使用
- 掌握动画元件的使用
- 认识动画元件间的区别

4.1　元 件 类 型

元件的应用对于减小 Flash 动画的容量起到极为重要的作用，使得 Flash 动画能够在拥挤的因特网上快速流传。

所谓元件，就是指具有独立时间轴的特殊对象，同时可以在动画制作中重复使用而不会大幅度增加文档容量。根据其使用方式的不同，元件主要分为 3 类——图形元件、影片剪辑以及按钮元件，现在就分别加以介绍。

4.1.1　图形元件

图形元件中通常就放置一些静态的图形，它具有与主时间轴同步运行的功能。除了通过转换为元件生成图形元件外，还可以通过创建新元件得到图形元件。具体操作步骤如下：

（1）选择“ 插入 ”|“ 创建新元件 ”命令，弹出“ 创建新元件 ”对话框，输入元件名称“ 图形元件 ”，选中“ 图形 ”单选按钮，如图 4.1 所示。

图 4.1　创建新的图形元件

（2）单击“ 确定 ”按钮确认后，由场景编辑状态进入图形元件编辑状态，其中可以根据前面所学的内容制作任何类型的动画，如图 4.2 所示。

注意：元件编辑状态中的十字线标志为元件编辑区的中心，通常将绘制得到的图形置于该十字线附近。如果在图形元件中使用交互式控件和声音，则交互式控件和声音是不会起任何作用的。

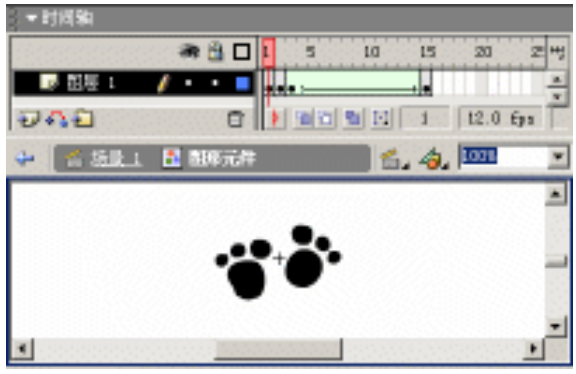

图 4.2 编辑图形元件

（3）单击编辑区上方的“场景 1”链接，返回到场景编辑状态。还可以通过选择“编辑”|“编辑文档”命令，或者直接按快捷键 Ctrl+E 切换到场景编辑状态。

提示：“编辑文档”和“编辑元件”使用同一个菜单项，根据编辑状态交替显示其中的一个命令。当处于场景编辑状态时，则显示为“编辑元件”，选择该命令后进入元件编辑状态；当处于元件编辑状态时，则显示为“编辑文档”，选择该命令后进入场景编辑状态。

4.1.2 影片剪辑

使用影片剪辑可以创建可重复使用的动画片断。影片剪辑拥有独立于主动画的时间轴进行播放的多帧时间轴，也就是说可以将影片剪辑视为主动画中的子动画，其中可以包括交互式控件、声音甚至其他元件。

创建影片剪辑的步骤如下：

（1）按快捷键 Ctrl+F8 调用“创建新元件”对话框，输入元件名称“影片剪辑”，选中“影片剪辑”单选按钮，如图 4.3 所示。

图 4.3 创建新的影片剪辑

（2）单击“确定”按钮确认后，由场景编辑状态进入影片剪辑编辑状态，其中可以根据前面所学的内容制作任何类型的动画，如图 4.4 所示。

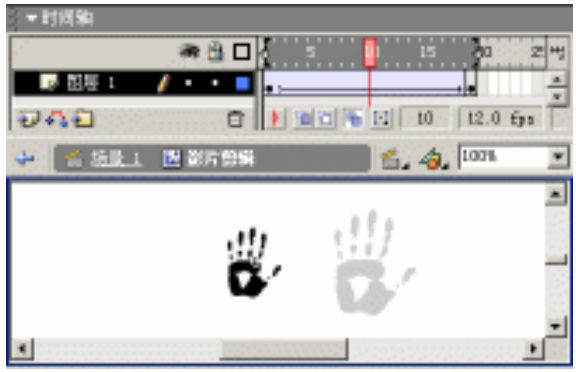

图 4.4　编辑影片剪辑

4.1.3　按钮元件

使用按钮元件可以在动画中创建响应鼠标单击、滑过或其他事件的交互式按钮。创建按钮元件的步骤如下：

（1）按快捷键 Ctrl+F8 调用“创建新元件”对话框，输入元件名称“按钮元件”，选中“按钮”单选按钮，如图 4.5 所示。

图 4.5　创建新的按钮元件

（2）单击“确定”按钮确认后，由场景编辑状态进入按钮元件编辑状态，可以发现其时间轴比较特殊，以“弹起”、“指针经过”、“按下”、“点击”进行标示，在“弹起”帧中绘制足形，连续按 F6 键插入关键帧并调整各帧中足形的颜色，如图 4.6 所示。

图 4.6　编辑按钮元件

在此需要说明的是“点击”帧的图形绘制。示例按钮各帧中的图形是通过插入关键帧得到的，因此，各帧中的图形基本相同，只是颜色上略有差别。此时，按钮的有效区域就是双足所构成的区域。

测试按钮时，当鼠标指针处于按钮有效区域外时，显示的是“弹起”帧中的图形；当鼠标指针移动到有效区域上时，显示的是“指针经过”帧中的图形；当鼠标指针在有效区域内时，按住鼠标左键就显示“按下”帧中的图形，如图 4.7 所示。

图 4.7 按钮的三种状态

重新对按钮进行编辑，将“点击”帧中的右足图形删除，只剩下左足图形。此时，按钮的有效区域就是左足所构成的区域。再次测试按钮时，当鼠标指针移动到右足上时，并没有发生按钮的切换，仍旧显示“弹起”帧中的图形；当鼠标指针移动到左足上时，则发生按钮的切换，显示“指针经过”帧中的图形，如图 4.8 所示。

图 4.8 按钮的不同点击区域

4.2 库 应 用

Flash 动画文档中的“库”面板存放的库项目包括了动画制作过程中的所有元件和由外部导入的素材资源（如视频、声音、位图等）。“库”面板显示为一个滚动列表，其中包含所有库项目，方便您在动画制作过程中随时查看和组织这些库项目。

每个 Flash 动画文档都拥有独立的库。但是在动画制作过程中，可以打开任意 Flash 文档的“库”面板，将该文档的库项目应用于当前文档中。

Flash 提供了多个公共库，其中有常用的库项目，如按钮元件、声音素材、学习交互等内容。当然，您也可以将常用的库项目作为公共库显示。

4.2.1　认识库

“ 库 ”面板是动画制作过程中的重要组成部分。新建 Flash 文档，选择“ 窗口 ”|“ 库 ”命令，弹出的“ 库 ”面板中并没有任何库项目，如图 4.9 所示。

图 4.9　“ 库 ”面板（空）

库项目创建

库项目包括两部分——动画元件和资源素材。在此重点介绍一下“ 库 ”面板中动画元件的库项目的创建方法，而有关资源素材的库项目的创建可以参考第 6、7 章。库项目的创建有以下几种方法：

- 单击“ 库 ”面板中“ 新建元件 ”按钮或直接按快捷键 Ctrl+F8，弹出“ 创建新元件 ”对话框，如图 4.10 所示，设置各项参数后单击“ 确定 ”按钮，进入元件编辑状态进行编辑即可完成动画元件的创建。

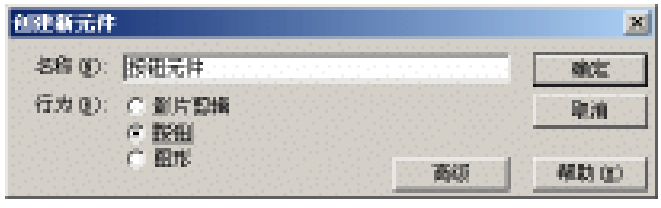

图 4.10　“ 创建新元件 ”对话框

- 选中舞台上的对象后，如图 4.11 所示将其拖动到“ 库 ”面板中或者直接按 F8 键，弹出“ 转换为元件 ”对话框，设置各项参数后单击“ 确定 ”按钮即可。

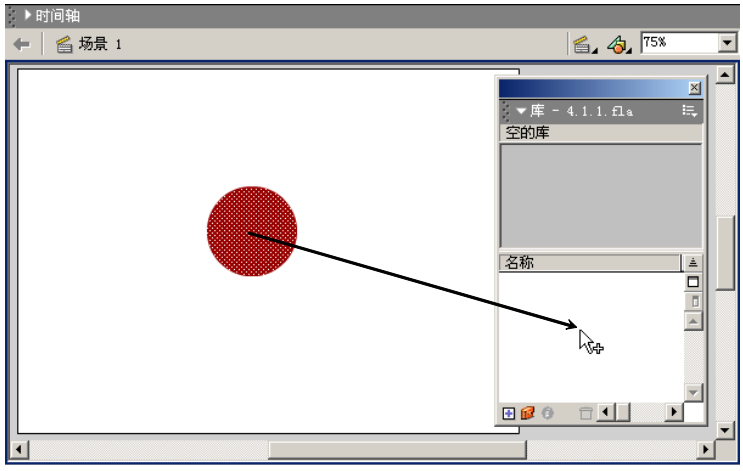

图 4.11 拖动对象到“库”面板

- 选择“文件”|“作为库打开”命令，弹出其他文档的“库”面板，将其中的库项目拖放到“库”面板中，如图 4.12 所示。或者直接将库项目拖动到舞台上进行应用，当然相应的库项目也显示在“库”面板中。

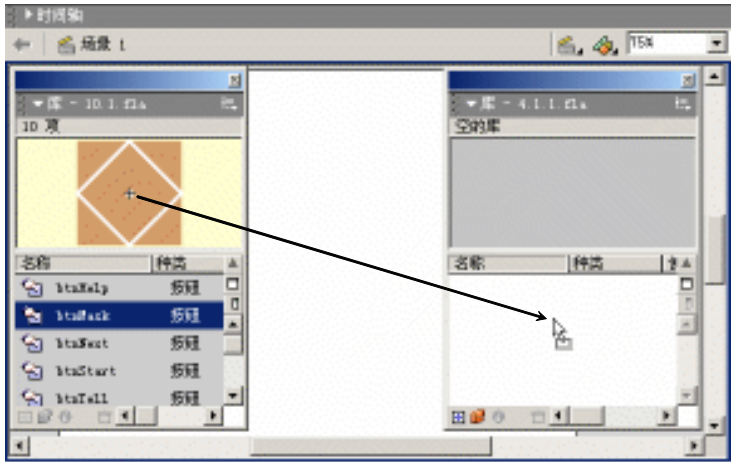

图 4.12 “库”面板间的库项目拖动

注意：采用该方法在“库”面板之间拖动库项目，当拖动的库项目在“库”面板中有相同名称的库项目时，将会弹出“解决库冲突”对话框，如图 4.13 所示，征询您是否替换现有项目。

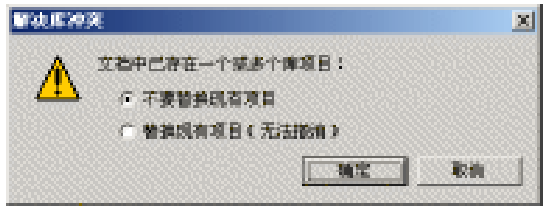

图 4.13 “解决库冲突”对话框

经过上述一系列库项目的创建生成，此时您可以发现“ 库 ”面板中放置了大量的库项目，单击“ 宽库视图 ”按钮，可显示库项目更为详细的内容，如图 4.14 所示。

图 4.14　具有元件的“ 库 ”面板

库项目排序

“ 切换排序 ”按钮和处于同一处交替出现，用于调整“ 库 ”面板中库项目的排列顺序。排序可以针对“ 库 ”面板中的各个字段——“ 名称 ”、“ 种类 ”、“ 使用次数 ”、“ 链接 ”以及“ 修改日期 ”进行调整。

默认情况下，“ 库 ”面板的“ 使用次数 ”字段并没有显示，选择“ 选项菜单 ”中的“ 保持最新使用次数 ”命令，那么每使用一次库项目就会自动更新一次库项目的使用次数。

库项目属性

不同的库项目具有不同的属性设置，但是所有动画元件的属性是相同的。选中“ 库 ”面板中的元件后，选择“ 选项菜单 ”中的“ 属性 ”命令或者直接单击“ 属性 ”按钮，弹出“ 元件属性 ”对话框，如图 4.15 所示，从中可以改变元件的名称和类型，单击其中的“ 编辑 ”按钮可以编辑所选元件。

图 4.15　“ 元件属性 ”对话框

关于视频、声音、位图等资源素材的属性都有所区别，具体内容可以参考第 6、7 章中的相关内容。

库项目删除

删除“ 库 ”面板中的库项目后，文档中所有库项目的应用也就相继删除。选择“ 库 ”面板中的库项目后，单击“ 删除 ”按钮或者直接按 Delete 键，此时弹出如图 4.16 所示的“ 删除 ”对话框，单击“ 删除 ”按钮即可。

图 4.16 “删除”对话框

注意：当库项目在舞台上应用时，“删除”对话框中的“删除元件实例”复选框将激活，撤选该复选框则删除元件时保留舞台上的实例。

4.2.2 公共库

Flash MX 提供了一些公共库，以简化动画的设计和制作。选择“窗口”|“公共库”级联菜单中的命令即可调用相应的公共库，应用其中的库项目。

- 按钮元件在动画制作中主要体现了人机的交互功能，类似于 Windows 操作系统中的各种按钮，但是 Flash MX 提供的按钮元件更加丰富、多姿多彩！选择“窗口”|“公共库”|“按钮”命令，弹出的“库”面板中显示了大量的按钮元件，如图 4.17 所示。
- 声音可以使动画更加完美，赏目的同时也可以悦耳。选择“窗口”|“公共库”|“声音”命令，弹出的“库”面板中显示了大量声音素材，如图 4.18 所示。

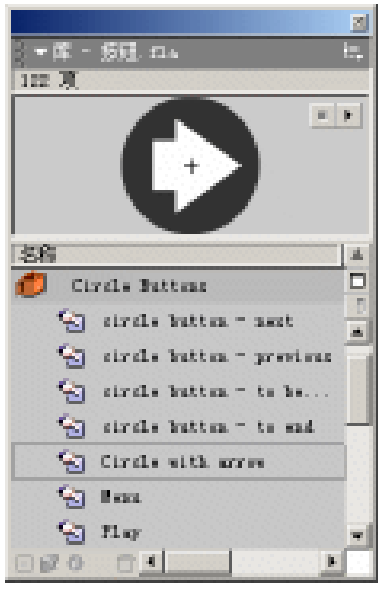

图 4.17 “按钮”公共库

图 4.18 “声音”公共库

- 最具有实际应用功能的莫属“学习交互”公共库。选择“窗口”|“公共库”|“学习交互”命令，弹出的“库”面板中显示了大量应用交互的案例，如图 4.19 所示。

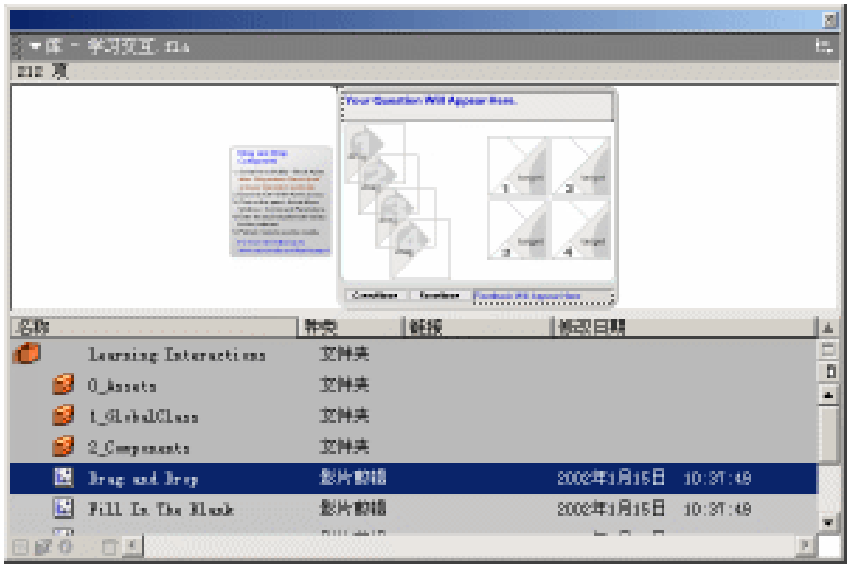

图 4.19　“ 学习交互 ” 公共库

虽然 Flash MX 提供了以上比较实用的公共库，但是这些库项目在动画制作过程中是远远不够的。其实，您完全可以创建自己的公共库，根据自己的需要调整公共库中的库项目，然后再将这些库项目应用到动画的制作过程中。方法如下：

首先新建文档并创建一系列库项目，然后存储为“ 科海.FLA ”，而保存位置确定为“ Flash MX 安装目录\First Run\Libraries ”，其中存放了原先的 3 个公共库文件——按钮、声音和学习交互。关闭 Flash MX 应用程序后重新启动，选择“ 窗口 ”|“ 公共库 ” 命令，查看其级联菜单中的菜单项，如图 4.20 所示。

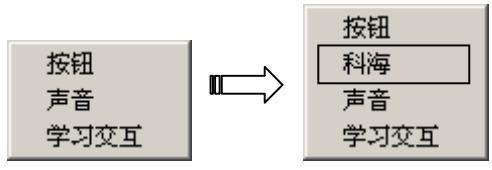

图 4.20　更改前后的“ 公共库 ” 菜单

4.3　元 件 使 用

通过上面 2 节的介绍，掌握了动画元件的分类和创建，了解了“ 库 ” 面板的操作和实用，现在就介绍一下元件使用的相关知识。元件创建完成后自动被放置在“ 库 ” 面板中，只要将“ 库 ” 面板中的元件拖放到舞台上即可应用，如图 4.21 所示。此时，放置在舞台上的对象只是对“ 库 ” 面板中相应元件的引用，通常被称之为实例。

图 4.21 使用元件

选中处于舞台上的实例，在“属性”面板中将显示该实例的引用——“实例：按钮元件”，如图 4.22 所示。

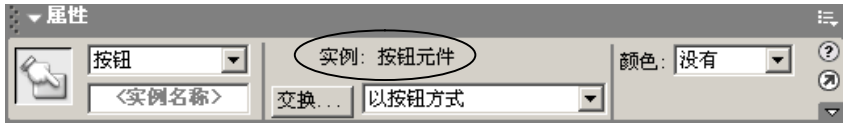

图 4.22 选中实例的“属性”面板

在舞台上分布着同一个元件的 5 个实例，如图 4.23 所示。双击舞台上的任一实例，进入该实例的元件编辑状态，删除右脚板图形后在空白处双击退出元件编辑，可以发现舞台上的 5 个实例全部发生相应的变化，如图 4.24 所示。

图 4.23 原始的舞台显示

图 4.24 修改后的舞台显示

当然，直接修改“库”面板中的元件，也会使得舞台上与之相应的实例发生自动调整。由此可见，元件是所有实例的根本。无论该元件在舞台上使用形成多少实例，元件在“库”

面板中只需存储一次。

一方面是当元件发生更改时，所有引用该元件的实例都会发生相应的变化；另一方面是各个实例之间允许存在差别，Flash MX 只保存这些实例发生变化的特殊信息——形状、位置和颜色等。

4.3.1　调整实例颜色

实例颜色是独立于元件的特殊属性，通过调整实例颜色可以使得实例呈现不同的效果。选中实例的情况下，“ 属性 ”面板的右侧为调整实例颜色的列表框，如图 4.25 所示。选中不同的选项，可对实例设置不同的颜色效果，如下：

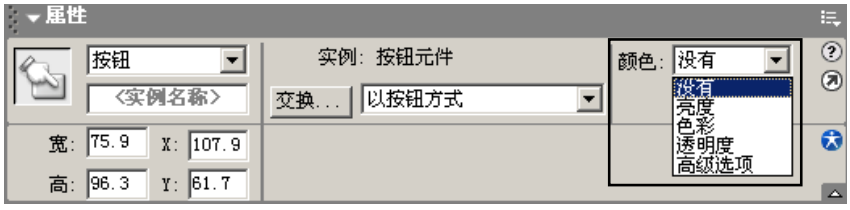

图 4.25　选中实例的“ 属性 ”面板

- 选中舞台上红色的“ 双足 ”实例，在“ 属性 ”面板的“ 颜色 ”列表框中选择“ 亮度 ”项，并将其参数设置为 100%，如图 4.26 所示，此时红色的“ 双足 ”实例呈白色显示。图 4.27 所示的就是调整实例亮度前后效果的比较。

图 4.26　亮度设置

图 4.27　调整实例亮度前后的效果

- 选中舞台上红色的“ 双足 ”实例，在“ 属性 ”面板的“ 颜色 ”列表框中选择“ 色彩 ”项，并按照图 4.28 所示设置所有参数，此时红色的“ 双足 ”实例呈亮黄色。图 4.29 所示的就是调整实例色彩前后效果的比较。

图 4.28　色彩设置

图 4.29　调整实例色彩前后的效果

- 选中舞台上红色的“双足”实例，在“属性”面板的“颜色”列表框中选择“透明度”项，并将其参数设置为 60%，如图 4.30 所示，此时透过红色的“双足”实例可以略微看到其下方的图像内容。图 4.31 所示的就是调整实例透明度前后效果的比较。

图 4.30 透明度设置

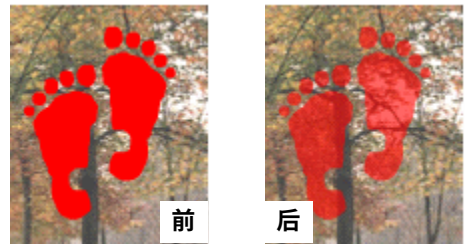

图 4.31 调整实例透明度前后的效果

- 选中舞台上的“图像”实例，在“属性”面板的“颜色”列表框中选择“高级选项”项，单击右侧的“设置”按钮弹出“高级效果”对话框，如图 4.32 所示设置各项参数，此时“图像”实例中的颜色发生本质的变化。图 4.33 所示的就是调整实例高级效果前后效果的比较。

图 4.32 “高级效果”对话框

图 4.33 调整实例高级效果前后的效果

通过上述 4 种实例颜色的调整方法，可以使得相同元件的不同实例呈现不同的效果，以使动画演示得更加绚丽多彩。当然，这需要结合本章上两节所讲知识和第 3 章的动画制作进行设计和创作！

4.3.2 变换实例形状

实例形状也是独立于元件的特殊属性，通过调整实例形状可以使实例呈现不同的外貌。Flash MX 中的任意变形工具就是专门用于调整舞台对象的，可以对实例进行倾斜、旋转、拉伸和缩放等操作。

倾斜实例

（1）选中舞台上的“双足”实例，按 Q 键切换到任意变形工具，同时单击工具选

项区中的“旋转和倾斜”按钮，此时实例外围呈现 8 个方形手柄。

（2）将光标移至中间的句柄呈现⇋时，按住鼠标左键横向拖动，虚线显示部分就是实例倾斜后的结果；释放鼠标左键后，就可以看到倾斜实例的最终结果。

倾斜实例操作的全过程如图 4.34 所示。

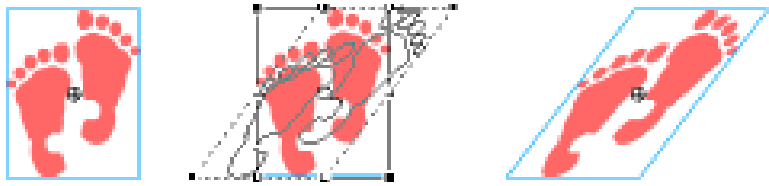

图 4.34　实例的倾斜

旋转实例

（1）选中舞台上的“双足”实例，按 Q 键切换到任意变形工具，同时单击工具选项区中的“旋转和倾斜”按钮，此时实例外围呈现 8 个方形手柄。

（2）将光标移至拐角的句柄呈现↻时，按住鼠标左键旋转拖动，虚线显示部分就是实例旋转后的结果；释放鼠标左键后，就可以看到旋转实例的最终结果。

旋转实例操作的全过程如图 4.35 所示。

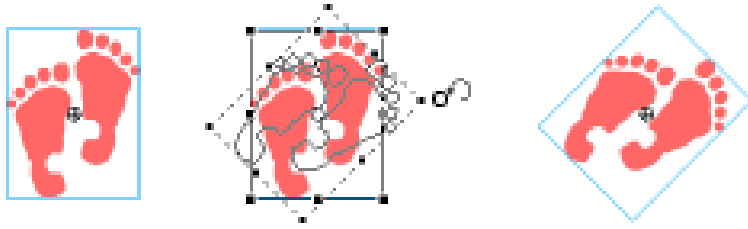

图 4.35　实例的旋转

拉伸实例

（1）选中舞台上的“双足”实例，按 Q 键切换到任意变形工具，同时单击工具选项区中的“缩放”按钮，此时实例外围呈现 8 个方形手柄。

（2）将光标移至中间的句柄呈现↔时，按住鼠标左键横向拖动，虚线显示部分就是实例拉伸后的结果；释放鼠标左键后，就可以看到拉伸实例的最终结果。

拉伸实例操作的全过程如图 4.36 所示。

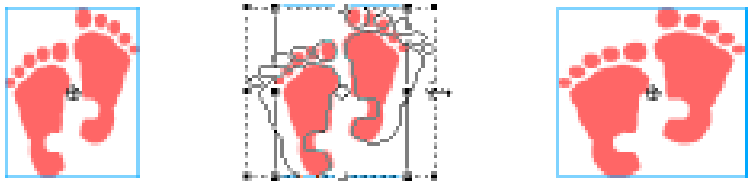

图 4.36　实例的拉伸

缩放实例

（1）选中舞台上的“双足”实例，按 Q 键切换到任意变形工具，同时单击工具选项区中的“缩放”按钮，此时实例外围呈现 8 个方形手柄。

（2）将光标移至拐角的句柄呈现↙时，按住鼠标左键斜向拖动，虚线显示部分就是实例等比例缩放后的结果；释放鼠标左键后，就可以看到缩放实例的最终结果。

缩放实例操作的全过程如图 4.37 所示。

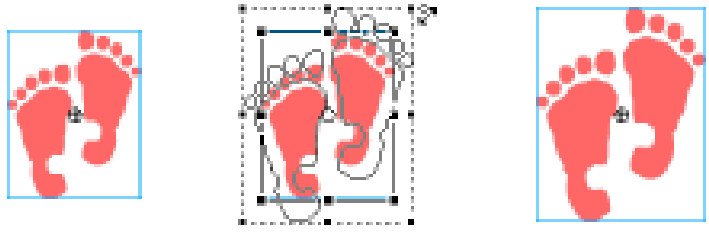

图 4.37　实例的缩放

4.3.3　更改实例行为

创建元件时，每个元件都有自己的行为，但是这并不局限于实例使用的行为。如果“库”面板中存在一个图形元件，但是动画制作过程中需要一个与图形元件中图形完全相同的影片剪辑。此时就没有必要再重新创建影片剪辑，而只要更改实例的使用行为即可。具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，选择“修改”|“文档”命令调用“文档属性”对话框，调整尺寸为 400px×300px。

（2）选择“插入”|“新建元件”命令，在弹出“创建新元件”对话框中输入名称为“计数”、确定行为为“图形”，如图 4.38 所示。

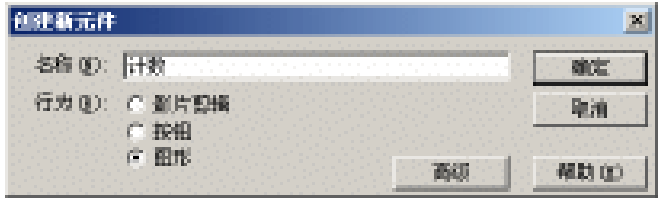

图 4.38　“创建新元件”对话框

（3）单击“确定”按钮确认，进入图形元件“计数”的编辑状态。按 T 键切换到文本工具，在舞台区上创建静态文本“0”，连续按 F6 键插入 9 个关键帧，依次将各个关键帧中静态文本的内容更改为 1、2、3、…、9，如图 4.39 所示。

（4）选中第 1～10 帧后，选择“编辑”|“拷贝帧”命令；依次在第 11、21、31、…、91 帧处按快捷键 Ctrl+Alt+V 粘贴帧。这样，在第 1～100 帧就交替依次显示数值 0～9。

（5）添加图层，在第 1 帧静态文本“ 0 ”的左侧再创建静态文本“ 0 ”，随后在第 11、21、31、…、91 帧处按 F6 键插入关键帧并修改静态文本为 0～9，如图 4.40 所示。这样，在第 1～100 帧就会显示显示数值 00～99。

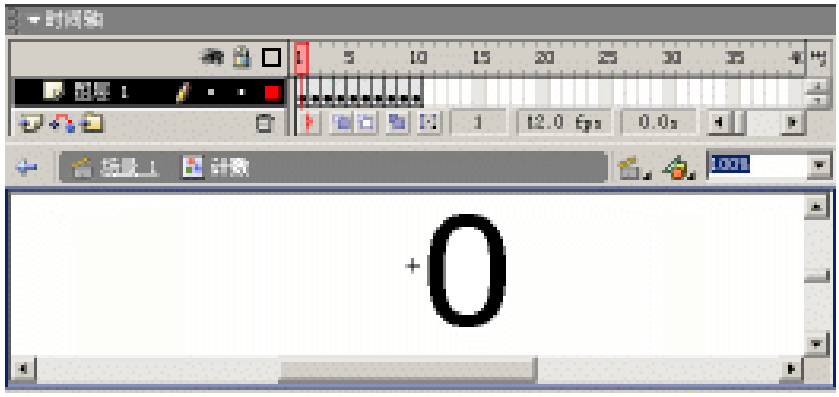

图 4.39　图形元件“ 计数 ”（1）

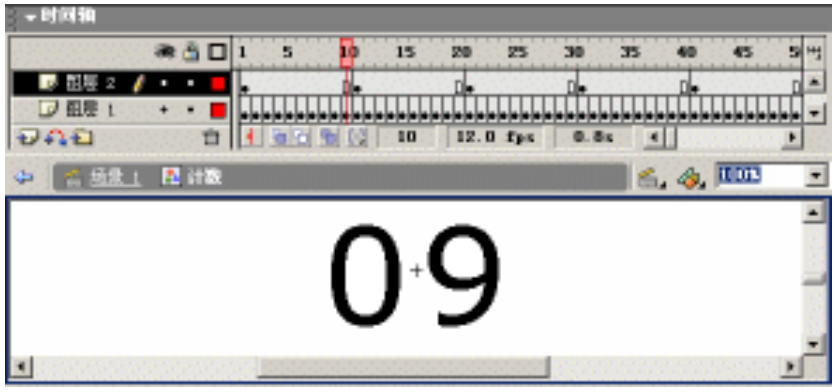

图 4.40　图形元件“ 计数 ”（2）

（6）按快捷键 Ctrl+E 切换到场景编辑状态，更改图层名称为“ 背景 ”，在舞台上绘制线性渐变的填充矩形，并持续到第 20 帧。添加图层“ 实例 ”，按 F11 键调用“ 库 ”面板，并将其中的图形元件“ 计数 ”拖放到舞台上（左右各放置 1 个），再创建多个静态文本，如图 4.41 所示。

（7）选中左侧的实例后，可以在“ 属性 ”面板中发现该实例行为为“ 图形 ”，如图 4.42 所示，保持原有的实例行为不作更改。

（8）选中右侧的实例后，可以在“ 属性 ”面板中发现该实例行为也是“ 图形 ”，单击列表框按钮并选择实例行为为“ 影片剪辑 ”，如图 4.43 所示。

至此，整个动画就制作完成，按快捷键 Ctrl+Enter 测试动画。仔细比较左右两侧数字的播放情况，可以发现左端数字始终在 00～19 之间变化，而右侧数字却在 00～99 之间完全显示，如图 4.44 所示。

图 4.41 舞台布置

图 4.42 左侧实例的“属性”面板

图 4.43 右侧实例的“属性”面板

图 4.44 效果预览

范例动画中，舞台上的两个实例虽然引用的是同一个图形元件，但是使用时采用不同的实例行为。因此在动画测试时，两个实例呈现不同的播放情况。

左侧数字的实例行为为“ 图形 ”，那么该实例的播放与主时间轴是同步进行的，也就是说该实例的播放与实例在主时间轴中持续帧数息息相关。范例动画中，左侧数字实例在主时间轴中延续了 20 帧，因此，左侧数字始终在 00～19 之间循环显示。

右侧文字的实例行为为“ 影片剪辑 ”，那么该实例的播放是独立于主时间轴的。因此，右侧数字不是在 00～19 之间循环显示，而是在 00～99 之间全部显示。

4.3.4　变换元件

有时，需要将已有实例的引用替换为其他元件，以保留实例颜色、形状和行为等特殊信息。现在就对实例进行元件交换，体会 Flash MX 中“ 变换元件 ”的功能。具体步骤如下：

（1）选择舞台上的黑色“ 双足 ”实例，将其进行水平倾斜处理，然后在“ 属性 ”面板中调整实例色调为 100%红色，实例显示效果前后变化如图 4.45 所示。

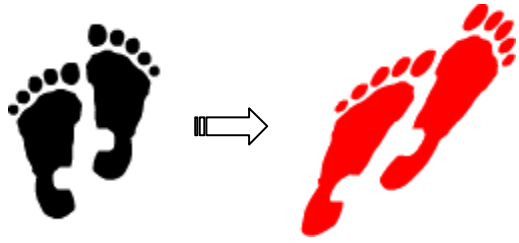

图 4.45　实例调整

（2）选中“ 双足 ”实例的情况下，再次注意“ 属性 ”面板的中间部分“ 实例：按钮元件-1 ”，如图 4.46 所示，表示该实例引用的是“ 按钮元件-1 ”。

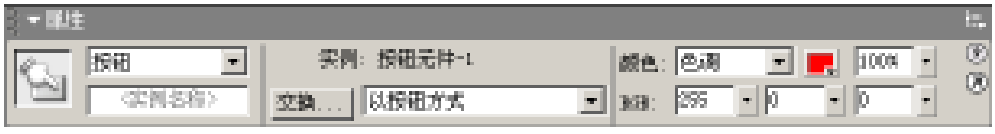

图 4.46　“ 属性 ”面板

（3）单击“ 属性 ”面板中的“ 交换 ”按钮，或者选择“ 修改 ”|“ 交换元件 ”命令，弹出“ 交换元件 ”对话框，选择其中的图形元件 Hand，如图 4.47 所示。

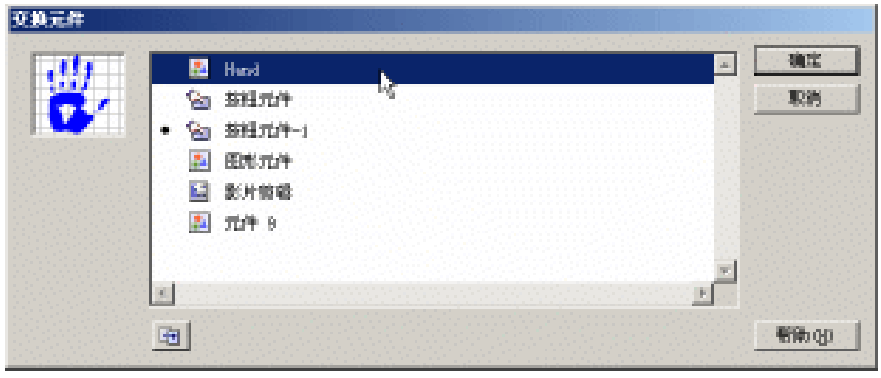

图 4.47　“ 交换元件 ”对话框

（4）单击“确定”按钮后，可以发现“属性”面板的中间部分显示为“实例：Hand”，如图 4.48 所示，而其他设置没有任何改变。与此同时，舞台上的实例也相应发生变化，如图 4.49 所示。左侧为图形元件 Hand；右侧就是在舞台上的实例，仍保留着原有形状和颜色等特殊信息。

图 4.48 “属性”面板

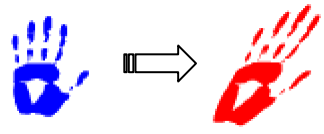

图 4.49 交换元件后的实例

4.4 实 战 演 练

动画元件是 Flash 动画中的重要部分，在介绍完前面的理论知识后，下面就来制作两个范例动画，以更加清楚地掌握动画元件的使用和相互之间的区别。

4.4.1 聚焦文字

浏览动画时，可以发现画面中始终放置着空心文字 khflash，字母逐个闪入覆盖在画面中的固定文字上，如图 4.50 所示。

如果您在浏览动画时比较细心的话，可以发现字母的闪入效果基本相同——都是由左上角移动到固定字母上，且都是由大逐渐缩小。

图 4.50 效果预览

下面来介绍此动画效果的制作步骤：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 400px×200px，帧频为 24fps，单击“确定”按钮确认。

（2）选择“插入”|“新建元件”命令，弹出“创建新元件”对话框，输入名称为 khflash，

选择行为栏中的“ 图形 ”单选按钮，如图 4.51 所示。

图 4.51　“ 创建新元件 ”对话框

(3) 单击“ 确定 ”按钮进入图形元件 khflash 的编辑状态，选择工具箱中的文本工具，在舞台中央创建字体为“ 华文彩云 ”的英文小写字母“ k ”。随后连续按 F6 键插入 6 个关键帧，分别在各个关键帧中依次输入 khflash 中的一个字母，如图 4.52 所示。

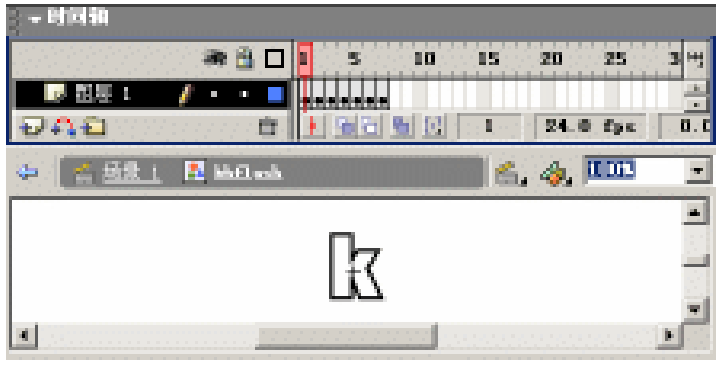

图 4.52　图形元件 khflash

(4) 按快捷键 Ctrl+F8 创建影片剪辑 k，更改图层名称为“ 背景 ”，按 F11 键调用“ 库 ”面板，并将其中的图形元件 khflash 拖放舞台上，如图 4.53 所示。

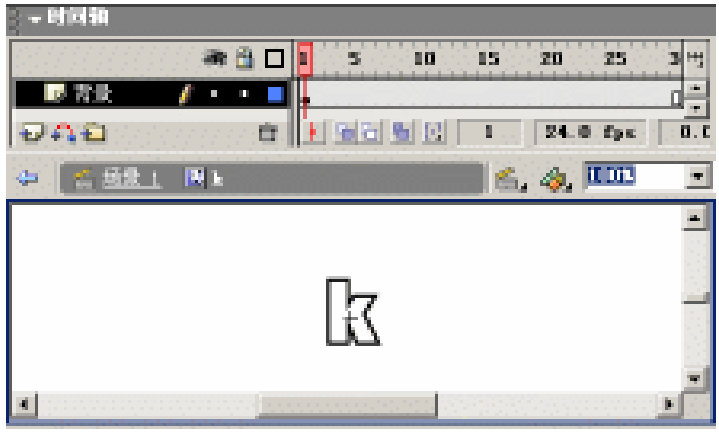

图 4.53　影片剪辑 k——“ 背景 ”图层

(5) 选中图形元件 khflash 实例的情况下，“ 属性 ”面板中的“ 图形选项 ”列表框中显示为“ 循环 ”，如图 4.54 所示。

图 4.54 修改前的“属性”面板

（6）单击“图形选项”列表框按钮，选择弹出列表中的“单帧”项，并确认右侧的“第一帧”文本框中为数值 1，如图 4.55 所示。

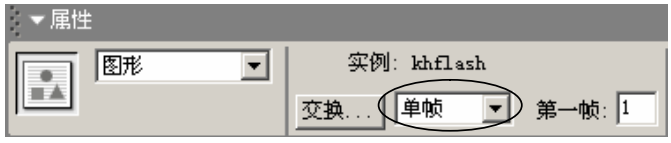

图 4.55 修改后的“属性”面板

（7）按快捷键 Ctrl+C 复制舞台上的图形元件 khflash 的实例，添加图层“闪烁”，选择“编辑”|“粘贴到当前位置”命令进行粘贴，右击该关键帧并选择弹出菜单中的“创建补间动画”命令创建运动动画，随即在第 6 帧处按 F6 键插入关键帧。

（8）选中“闪烁”图层第 1 帧中的实例，并将其适当放大（约 150%），设置实例透明度为 0%；选中第 6 帧中的实例并设置实例色调为 100%白色。此时，影片剪辑 k 的图层“闪烁”中生成了文字闪烁的动画效果，如图 4.56 所示。

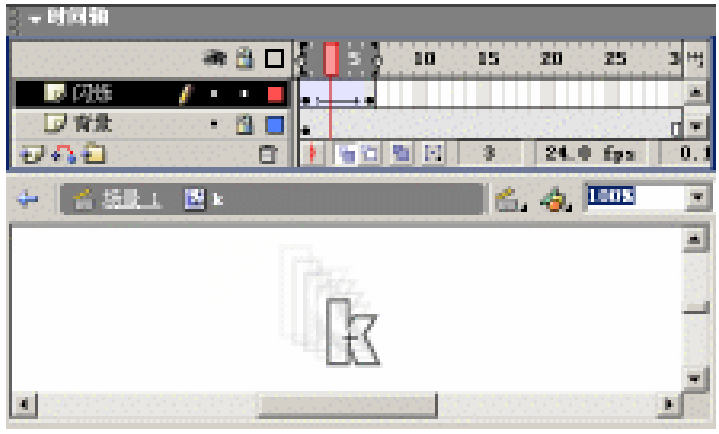

图 4.56 完成后的影片剪辑 k

（9）右击“库”面板中的影片剪辑 k，选择弹出菜单中的“复制”命令，在弹出的“复制元件”对话框中输入名称为“h1”，如图 4.57 所示，单击“确定”按钮确认。

（10）双击“库”面板中的影片剪辑 h1 进行编辑修改，将该影片剪辑中的所有图形元件实例设置图形选项为“单帧”、“第一帧”文本框中的数值为 2。此时，影片剪辑 h1 中的整个动画由字母“k”变为“h”，为了与影片剪辑 k 存在时间的延迟，将“闪烁”图层中的运动动画向后移动 3 帧，如图 4.58 所示。

图 4.57　“ 复制元件 ” 对话框

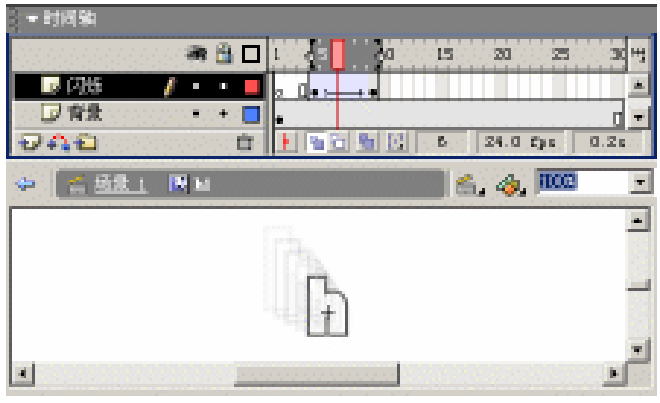

图 4.58　影片剪辑 h1

（11）按照步骤（9）、（10）继续创建影片剪辑 “ f ”、“ l ”、“ a ”、“ s ”、“ h2 ”，需要对各个影片剪辑进行调整的是 “ 第一帧 ” 文本框中的数值和影片剪辑之间的时间延迟。

（12）按快捷键 Ctrl+E 切换到场景编辑状态，更改图层名称为 “ 背景 ”，在舞台上绘制覆盖整个舞台的线性渐变的填充矩形。添加图层字母，将 “ 库 ” 面板中的所有影片剪辑拖放到舞台上，按照字母 “ khflash ” 进行排列组合，如图 4.59 所示。

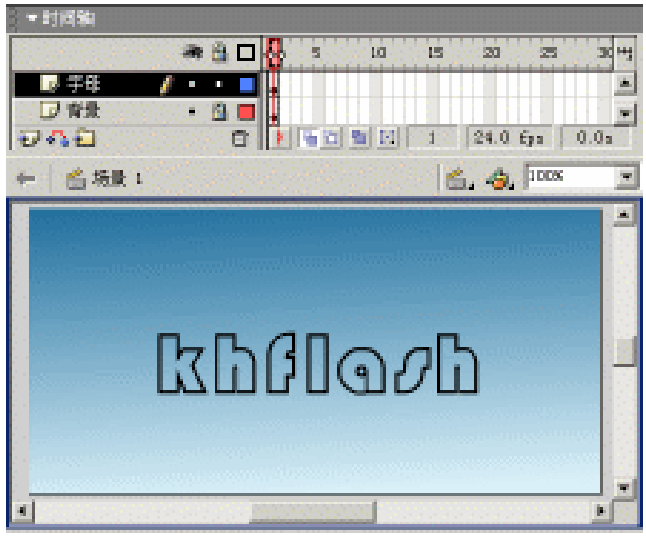

图 4.59　舞台布置

至此，整个动画就制作完成了，按快捷键 Ctrl+Enter 测试动画。

4.4.2 系列按钮

动画界面中放置了 3 个按钮——简单按钮、动态按钮和音效按钮，如图 4.60 所示。当鼠标光标移动到按钮上时，都会发生图形的轮换。如果将光标移动到音效按钮上时，可以听到短促的声音；按下鼠标左键，会发现图形轮换的同时也发出声响。

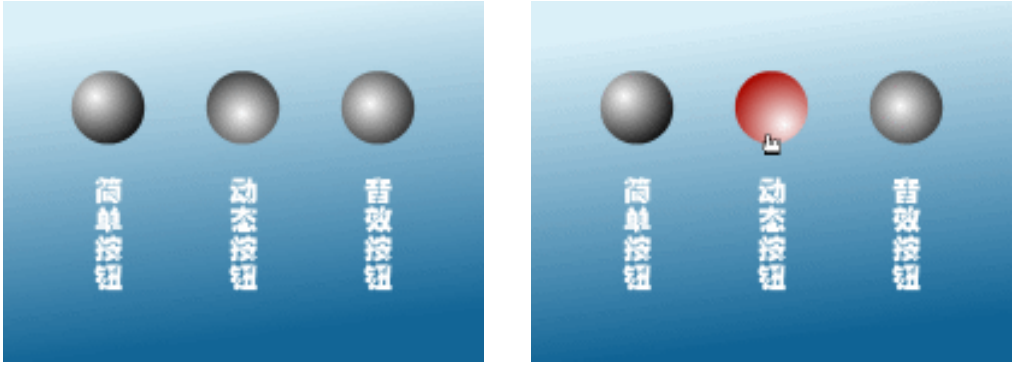

图 4.60 效果预览

按钮元件是 Flash 中比较特殊的对象，按钮元件的时间轴并不是以序数定义的，而是按照按钮的状态——“ 弹起 ”、“ 指针经过 ”、“ 按下 ” 和 “ 点击 ” 等进行命名的。

为了更加清楚地掌握复杂按钮元件制作的全过程（这里涉及的复杂并不是按钮图形的精美，而是在按钮状态中置入动画和声音），将按钮的复杂过程分为 3 个部分进行讲解，即简单按钮、动态按钮和音效按钮！

简单按钮

通过简单按钮的制作，可以更加清楚按钮元件中各帧与按钮各种状态之间的相互关系，并熟练掌握按钮的制作方法。具体步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+F8 调用 “ 创建新元件 ” 对话框，在 “ 名称 ” 文本框中输入 “ 简单按钮 ”，选中 “ 行为 ” 栏中的 “ 按钮 ” 单选按钮，如图 4.61 所示。

图 4.61 “ 创建新元件 ” 对话框

（2）单击 “ 确定 ” 按钮进入按钮元件 “ 简单按钮 ” 的编辑状态，利用椭圆工具 在 “ 弹起 ” 帧的舞台上绘制黑白放射渐变的填充圆形，并使用颜料桶工具 调整填充圆形的高光点，如图 4.62 所示。

图 4.62　简单按钮——“ 弹起 ”帧

（3）按 F6 键插入关键帧，在“ 混色器 ”面板中设置红白放射渐变，利用颜料桶工具将“ 指针经过 ”帧中的填充圆形调整为红白放射渐变，如图 4.63 所示。

图 4.63　简单按钮——“ 指针经过 ”帧

（4）按 F6 键插入关键帧，利用颜料桶工具将“ 按下 ”帧中填充圆形的高光点调整到右下角，如图 4.64 所示。

图 4.64　简单按钮——“ 按下 ”帧

（5）按 F6 键插入关键帧，任意设置“ 点击 ”帧中圆形的填充颜色，如图 4.65 所示。

至此，按钮元件“ 简单按钮 ”就制作完成了，随后在按钮元件中植入动画，使之成为具有动画效果的按钮元件。

图 4.65 简单按钮——“点击”帧

动态按钮

按钮的每个状态只占有一帧，而动画制作需要多帧才能实现，如何在按钮状态中实现动画效果呢？解决的惟一方法就是在按钮的各帧中置入影片剪辑，因为影片剪辑是独立于时间轴播放的。现在，就以按钮弹起状态中实现动画效果为例介绍动态按钮的制作。

（6）右击“库”面板中的按钮元件“简单按钮”，选择弹出菜单中的“复制”命令，在弹出的“复制元件”对话框中输入名称为“动态按钮”，如图 4.66 所示。

图 4.66 “复制元件”对话框

（7）单击“确定”按钮，“库”面板中即显示按钮元件“动态按钮”，双击该元件进入编辑状态进行修改。选中“弹起”帧中的填充圆形，按快捷键 Ctrl+X 进行剪切。

（8）新建影片剪辑“滚动动画”，在该影片剪辑编辑状态中按快捷键 Ctrl+Shift+V 原位粘贴刚才的填充圆形，分别在第 13、25 帧处按 F6 键插入关键帧，并将第 13 帧处填充圆形的高光点置于右下角。最后，再在这 3 个关键帧之间创建形变动画，如图 4.67 所示。这样，圆形中高光点滚动的动画效果就制作完成了。

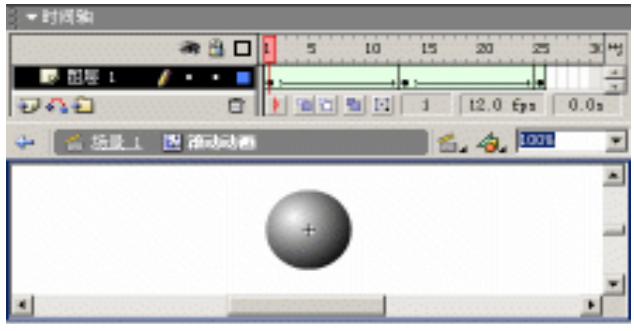

图 4.67 影片剪辑“滚动动画”

（9）再次双击“ 库 ”面板中的按钮元件“ 动态按钮 ”对其进行编辑修改，单击“ 弹起 ”帧后拖动 “ 库 ” 面板中的影片剪辑 “ 滚动动画 ” 到舞台上，如图 4.68 所示。

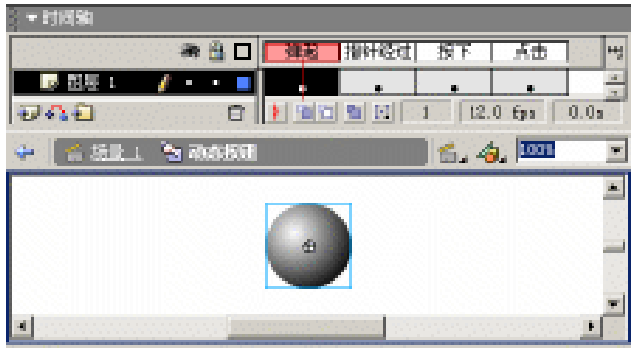

图 4.68　动态按钮

至此，按钮元件 “ 动态按钮 ” 就制作完成了。随后，在按钮元件 “ 动态按钮 ” 的基础上添加声音生成 “ 音效按钮 ”，使之成为既具有动画效果、又可以聆听声音的按钮元件。

音效按钮

如果单击按钮时没有声音效果，似乎有点单调且缺少生机，可以说是比较遗憾的。或许，您并不在意按钮状态中的声音效果，但是实践证明，使用音效的按钮确实增色不少。

（10）右击 “ 库 ” 面板中的按钮元件 “ 动态按钮 ”，选择弹出菜单中的 “ 复制 ” 命令，并在弹出的 “ 复制元件 ” 对话框中输入名称为 “ 音效按钮 ”，单击 “ 确定 ” 按钮确认。

（11）按快捷键 Ctrl+R 弹出“ 导入 ”对话框，选择声音素材“ Over.wav ”、“ Down.wav ”后单击“ 打开 ”按钮即可导入，此时 “ 库 ” 面板中就会显示种类为 “ 声音 ” 的两个库项目，如图 4.69 所示。

图 4.69　“ 库 ” 面板

（12）双击“ 库 ”面板中的按钮元件“ 音效按钮 ”进行编辑修改，在“ 指针经过 ”帧处单击选中该关键帧，在“ 属性 ”面板的“ 声音 ”列表框中选择 Over 项，如图 4.70 所示；同样在“ 按下 ”帧中选择“ 声音 ”列表框中的 Over 项。

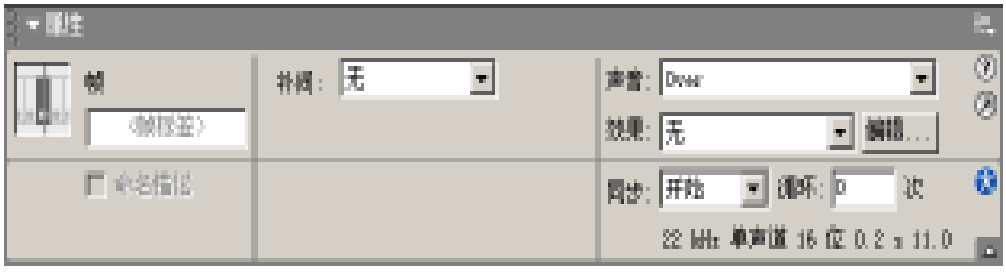

图 4.70　设置声音

（13）此时，观察按钮元件“ 音效按钮 ”的时间轴，可以发现在“ 指针经过 ”和“ 按下 ”帧中呈现声波折线，如图 4.71 所示。

图 4.71　音效按钮

至此，按钮元件“ 音效按钮 ”也就制作完成了。

舞台布置

（14）按快捷键 Ctrl+E 切换到场景编辑状态，更改图层名称为“ 背景 ”，在舞台上绘制覆盖整个舞台的线性渐变的填充矩形。

（15）添加图层“ 元件 ”，将“ 库 ”面板中的按钮元件拖放到舞台上横向布置，由左至右依次为“ 简单按钮 ”、“ 动态按钮 ”和“ 音效按钮 ”，并在各个按钮下创建相应的文本，如图 4.72 所示。

至此，系列按钮就制作完成了，按快捷键 Ctrl+Enter 测试动画。

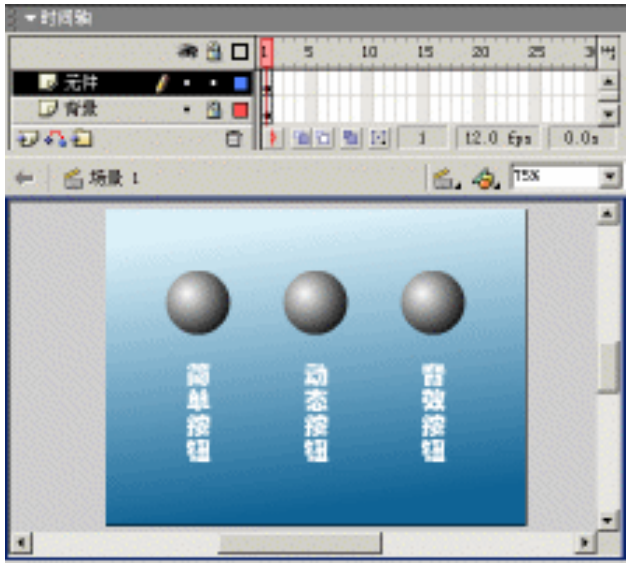

图 4.72　舞台布置

4.5　小结与练习

本章重点介绍了 Flash 动画涉及的所有动画元件——影片剪辑、按钮元件和图形元件，同时讲解了这些动画元件的存储位置——“ 库 ”面板。本章将动画元件的创建、使用和调整等经常涉及的问题一一讲解透彻。

最后结合两个范例动画，使您对动画元件应用的整个流程更加清楚，尤其是范例“ 聚焦文字 ”还介绍了制作演示效果相同的动画的一种快捷方法。

通过本章的学习，希望能够完成以下练习：

1．根据元件行为的不同，将其应用在动画制作中，充分体会影片剪辑、按钮元件和图形元件之间的差别！

2．设法制作一段沿曲线运动的形变动画。

3．制作既具有动画又具有音效的按钮元件，要求将鼠标光标移动到按钮上时，发出一声爆炸声响的同时图形也变得四分五裂；移动离开时又可以看到完整的按钮形状。

第5章 动 画 脚 本

Flash 提供了其独有的脚本编写语言，它是 Flash MX 非常重要的组成部分。动作脚本的语法和风格与 JavaScript 极其相似。本书秉承 Flash MX 开发设计的思想，对动画脚本进行了较为深入的探讨。

本章重点介绍作为面向对象的脚本编写语言，并解释了动画脚本术语以及基本编程概念，包括函数、变量、语句、运算符、条件和循环。随后以丰富的实例充分剖析了 Flash 动画中较为常用的动画脚本，由最简单的在单帧单对象设置脚本开始，逐步深入到多帧多对象的脚本设置，从而使您充分体会动画脚本所具备的强大交互性。

通过本章的学习，应该达到如下目的：

- 认识动画脚本的编写环境
- 掌握动画脚本的控制结构
- 熟悉使用大量的脚本语句
- 制作多个精彩的动画实例

5.1 脚本基本知识

Flash MX 的脚本编写环境更加人性化，不仅提供了脚本对象间快速切换的简便功能，而且还提供了脚本语句简单、扼要的实时帮助。如果要了解相关脚本更为详细的使用方法等内容，可以求助于功能强大、面面俱到的“脚本参考”面板。

开始学习动画脚本的编写前，应该先来学习有关脚本编写的基本知识，为以后的脚本动画制作奠定扎实的基础。首先，就来认识和熟悉一下脚本编写环境——“动作”面板，同时了解一下脚本编写中的一些术语和结构化控制语句。

5.1.1 认识“动作”面板

“动作”面板是动画制作过程中进行脚本编写的重要场所，Flash MX 为动画开发人员提供了更为人性化的脚本编写环境。整个面板的界面分布更加合理，如图 5.1 所示。

Flash 所提供的脚本编写环境是半自动化的。首先，在“动作”面板的动作选择区中选择脚本语句，双击该脚本语句后即会出现在脚本编写区；然后，选中脚本编写区中的该脚本语句时，会在参数设置区中显示该脚本语句的相关参数设置；最后，对这些参数进行设置就完成一句脚本语句的输入。因此，Flash 中的脚本编写没有必要记住所有的脚本语句，只要能够在动作选择区中快速找到需要的脚本语句即可。

“动作”面板中会实时显示所选脚本语句简明扼要的提示内容。如果您还不是太理解所选脚本语句的使用方法，那么可以选择“窗口”|“脚本参考”命令，在弹出的“脚本参考”面板中获取帮助，如图 5.2 所示。

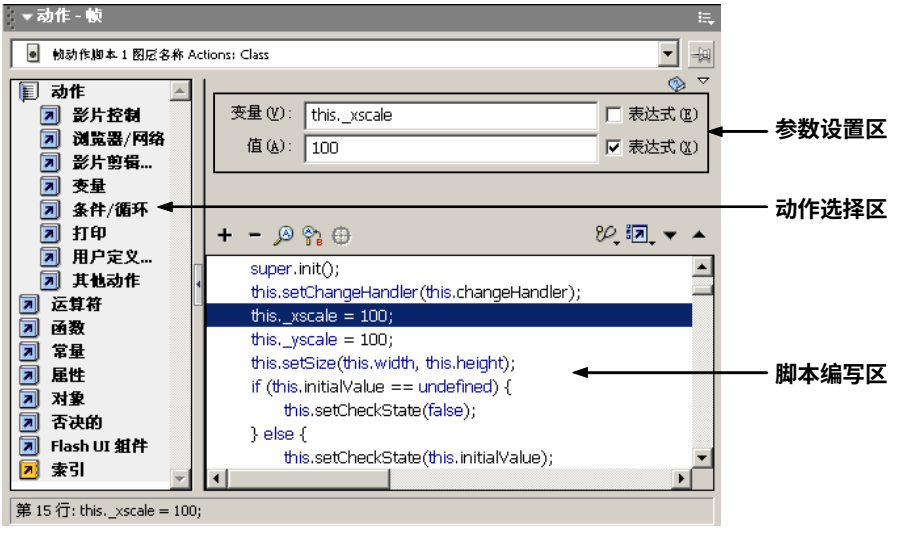

图 5.1　“动作”面板

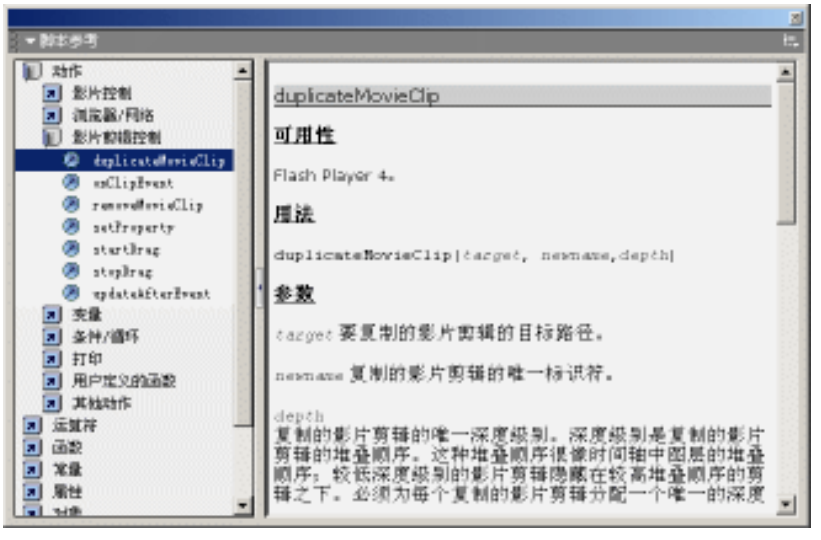

图 5.2　“脚本参考”面板

5.1.2　函数及运算符

函数

Flash MX **中包含了预定义函数和自定义函数两种。预定义函数是软件开发人员在脚本中集成的内部函数，这种函数已经完成了定义的过程，需要时直接使用即可。在** Flash **中提供了** 3 **类预定义函数——常用函数、数学函数和转换函数。各类函数的名称、语法以及功能说明可以参考表** 5.1**、表** 5.2 **和表** 5.3**。**

表 5.1 常用函数

名称	语法	功能说明
eval	eval (X)	返回表达式 X 的更新值
getTimer	getTimer ( )	获取播放时间
escape	escape (X)	转换字符串编码
unescape	unescape(X)	取消转换编码
getProperty	getProperty(MCName, property)	获取影片剪辑的属性值
getVersion	getVersion ( )	获取操作系统版本
targetPath	targetPath(MCName)	获取 MC 实例名称的字符串

表 5.2 数学函数

名称	语法	功能说明
isFinite	isFinite(expression)	检验表达式是否为有穷数
isNaN	isNaN(expression)	检验表达式是否为无穷数
parseFloat	parseFloat(string)	将字符串转换为浮点数
parseInt	parseInt(expression, radix)	将字符串转换为整数

表 5.3 转换函数

名称	语法	功能说明
Boolean	Boolean (expression)	返回表达式的布尔值
Number	Number(expression)	将表达式强制转换为数值
string	string(expression)	将表达式转换为字符串

除了集成预定义函数外，Flash 还允许用户自定义函数以满足脚本编写的需要。与预定义函数一样，自定义函数也可以传递参数、返回函数值，可以在定义后任意调用。以下脚本代码中定义了一个名为 square 的函数，返回参数 x 的平方值，而在函数外可以直接调用。

```
function square(x) {
  return x*x;
}
tmp = square(4);
```

自定义函数可以在任意位置处定义，但是必须要在函数被调用之前完成定义，否则就会引起函数调用错误。上述脚本代码中，函数 square 会将参数 x 作为具体的值进行运算。在函数调用时，参数 x 被赋值为数值 4 ，所以变量 tmp 的最终结果是 16。

运算符

脚本编写过程中，通常需要大量的数值运算或字符串比较，因此运算符的使用是必不可少的。为了清楚各个运算符的使用，将所有运算符分为以下 6 种——算术运算符、比较运算符、逻辑运算符、位运算符、赋值运算符和补充运算符。

算术运算符除包括常用的“加”、“减”、“乘”、“除”外，还包括“取余”运算，关于这些运算符的具体使用语法结构参考表 5.4。

表 5.4　算术运算符

运算符	语法	说明	备注
+	X + Y	加法运算	
-	X - Y	减法运算	
*	X * Y	乘法运算	
/	X / Y	除法运算	表达式 Y 不能等于 0
%	X % Y	取余运算	X 除以 Y 得到的余数

比较运算符通常应用于语句 if 的条件中，当表达式 X、Y 间的比较成立时，返回布尔值 true，否则就返回布尔值 false。比较运算符的使用方法可以参考表 5.5。

表 5.5　比较运算符

运算符	语法	说明	运算符	语法	说明
>	X > Y	大于	!=	X != Y	不等于
<	X < Y	小于	==	X == Y	等于
>=	X >= Y	不小于	<>	X <> Y	不等于
<=	X <= Y	不大于			

逻辑运算符通常应用于语句 if 中多个条件的组合之间，以条件 X、Y 而言，逻辑运算符得到的相应结果可以参考表 5.6。

表 5.6　逻辑运算符

运算符	语法	说明	备注
&&	X && Y	逻辑与	当条件 X、Y 均返回布尔值 true 时，则返回布尔值 true；否则都为布尔值 false
||	X || Y	逻辑或	当条件 X、Y 均返回布尔值 false 时，则返回布尔值 false；否则都为布尔值 true
!	! X	逻辑非	如果条件 X 返回的布尔值为 true，那么最终得到布尔值 false；也就是说条件 X 的值与最终得到的布尔值恰好相反

位运算符是将数值表达式 X 转换为 32 位整数（二进制数）后再进行相关操作，它包括的运算符参考表 5.7。

表 5.7　位运算符

运算符	语法	说明	备注
&	X & Y	位与	
<<	X << Y	逐位左移	表达式 Y 处于 0～31 间的整数
>>	X >> Y	逐位右移	表达式 Y 处于 0～31 间的整数
>>>	X >>> Y	逐位右移（无元件）	表达式 Y 处于 0～31 间的整数
^	X ^ Y	逐位异或	
|	X | Y	逐位或	
~	~X	逐位取非	

赋值运算符通常应用于变量的初始化和重新赋值等，由此可分为简单赋值运算与复杂赋值运算，具体使用语法参考表 5.8。

表 5.8 赋值运算符

运算符	语法	说明	备注
=	X = 11	简单赋值运算	
+=	X += Y	复杂赋值运算	等价于 X＝X+Y
-=	X -= Y	复杂赋值运算	等价于 X＝X-Y
*=	X *= Y	复杂赋值运算	等价于 X＝X*Y
/=	X /= Y	复杂赋值运算	等价于 X＝X/Y
%=	X %= Y	复杂赋值运算	等价于 X＝X%Y
&=	X &= Y	复杂赋值运算	等价于 X＝X&Y
<<=	X <<= Y	复杂赋值运算	等价于 X＝X<<Y
>>=	X >>= Y	复杂赋值运算	等价于 X＝X>>Y
>>>=	X >>>= Y	复杂赋值运算	等价于 X＝X>>>Y
^=	X ^= Y	复杂赋值运算	等价于 X＝X^Y
|=	X |= Y	复杂赋值运算	等价于 X＝X|Y

除了上述各种运算符外，其他所有运算符都归纳在补充运算符中。关于补充运算符的使用方法及其说明可以参考表 5.9。

表 5.9 补充运算符

运算符	语法	说明	备注
?:	X ? Y : Z	条件选择符	判断条件 X 确定选择 Y、Z 中之一
++	X++	后置递加运算符	等价于 X＝X+1
	++X	前置递加运算符	等价于 X＝X+1
--	X--	后置递减运算符	等价于 X＝X-1
	-- X	前置递减运算符	等价于 X＝X-1
type of	typeof (X)	一元运算符	检验 X 得出表达式 X 的类型
void	void (X)	一元运算符	摒弃 X 的携带值，返回未定义值

表 5.9 中应明确 X++与++X 之间的区别，假设两个数值变量 X、Y，初始化变量 X=2，如果变量 Y 的赋值表达式为 Y=X++，则最后得到的运行结果为 X=3、Y=2；如果变量 Y 的赋值表达式为 Y=++X，则最后得到的运行结果为 X=3、Y=3。

由上述描述可以得出结论：后置递加运算符 X++、前置递加运算符++X 间的区别就在于对另一个变量的赋值。后置递加运算符 X++是先对变量 Y 赋值后进行递加，前置递加运算符 X++是先进行递加后再对变量 Y 赋值。

5.1.3 判断结构

判断结构主要是对条件（Condition）做出判断，根据返回的布尔值（即 true 或 false）确定是否执行指定脚本。判断结构的脚本语句参考表 5.10。

表 5.10　判断结构

语句	用途说明
if	判断条件，返回 true 值执行 if 中脚本
else	判断条件，返回 false 值执行 else 中脚本
else if	判断条件，返回 false 值进行再次判断

if

语句 if 对条件 Condition 进行判断，确定是否执行指定的脚本 Statement。如果条件 Condition 成立，则返回布尔值 true，同时执行语句 if 中的指定脚本；否则返回布尔值 false，并忽略语句 if 中的指定脚本。具体语法结构如下：

```
if ( Condition ) {
  Statement;
}
```

如果语句 if 的条件 Condition 中涉及变量时，应该预先对变量进行定义并且赋值：

```
i = 15;
if (i<20) {
  trace ("该数值小于 20");
}
```

上述脚本中首先定义变量 i，并且初始化为数值 15；再利用语句 if 确定是否执行相应的脚本。很明显，数值 15 当然小于数值 20，也就是说经过条件判断成立时，编译执行指定脚本，在测试动画时会自动弹出“输出”窗口，并显示文字内容“该数值小于 20”。

假设变量 i 初始数值大于 20，那么在测试动画时就不会弹出“输出”窗口，也不会显示上述文字内容，也就是说，当语句中条件不成立时，编译就不执行指定脚本。

if…else…

单独使用语句 if，只有当条件成立时才会执行语句中的指定脚本。假设要求在条件不成立时，也要求显示相应信息，那么就只有联合使用语句 if 和 else 了，以构成条件判断不成立时的分支。对于语句 if 与 else 联合使用，具体采用语法如下：

```
if ( Condition ) {
  Statement 1;
} else {
  Statement 2;
}
```

语句 if 和 else 的联合使用相当于将条件 Condition 一分为二，即条件成立（返回布尔值为 true）时，执行语句 if 中的指定脚本；条件不成立（返回布尔值为 false）时，执行语句 else 中的指定脚本。

如果在语句 if 中示例的基础上添加语句 else，这样条件 Condition 经判断不管成立与否，都可以执行相应的脚本，具体代码设置如下：

```
i = 15;
if (i<20) {
```

```
  trace ("该数值小于 20");
} else {
  trace ("该数值不小于 20");
}
```

上述脚本代码中首先声明变量 i，并初始化为数值 15；再利用语句 if 确定是否执行相应的脚本。很明显，数值 15 当然小于数值 20，条件判断肯定成立，编译时执行 if 语句中的指定脚本，自动弹出“输出”窗口并显示文字内容“该数值小于 20”。

假设变量 i 初始数值大于 20，那么在编译时同样会弹出“输出”窗口，但显示的不是“该数值小于 20”，而是“该数值不小于 20”。

由示例可以得出这样的结论：语句 if…else…就是将条件 Condition 一分为二，也就是对应于布尔值的 true 或 false。

if…else if…

语句 else if 通常用于多次判断，也就是说在第 1 个条件不成立时再进行第 2 个条件的判断。语句 else if 与语句 if 联合使用，具体采用语法如下：

```
if ( Condition 1 ) {
  Statement 1;
} else if ( Condition 2 ) {
  Statement 2;
}
```

其中参数 Condition 1、Condition 2 就是用于判断的条件。当条件 Condition 1 成立时，执行语句 if 中的指定脚本 Statement 1；否则就再次对条件 Condition 2 进行判断，如果条件成立就执行语句 else if 中的指定脚本 Statement 2。

如果要确定某个变量处于哪个区域，在此假设由三个区域供选择（-∞, 20）、[20, 40]、（40, +∞），可以采用如下脚本程序加以区分：

```
if (i<20) {
  trace ("该数值小于 20");
} else if (i>40) {
  trace ("该数值大于 40");
} else {
  trace ("该数值处于 20～40 之间");
}
```

编译上述脚本时，如果变量 i 被分成三个区域，分别表现为：小于 20 就输出文字内容“该数值小于 20”；大于 40 就输出文字内容“该数值大于 40”；而处于数值 20～40 之间就输出文字内容“该数值处于 20～40 之间”。

这样能够比较明确地知道变量 i 所处的区域范围，如果要使得范围变得更小、更明确，可以采用更多的语句 else if。但是当使用多重 else if 语句时，势必影响程序的运行速度，因此应该将区域进行分段后再逐渐细化。

```
if ( i<20 ) {
  Statement 1;
} else if ( i<40 ) {
  Statement 2;
```

```
if ( i<40 ) {
  if ( i<20 ) {
    Statement 1;
  } else {
```

```
} else if ( i<60 ) {
  Statement 3;
} else {
  Statement 4;
}
```

```
    Statement 2;
  }
} else {
  if ( i<60 ) {
    Statement 3;
  } else {
    Statement 4;
  }
}
```

在上述左右两段脚本程序中，虽然运行后得到的结果基本相同，但是相对而言，采用右边的脚本程序是较为明智的。这是由于右边的脚本在编写结构上要比左边脚本简洁明了，在运行速度上右边脚本相对也要快得多。

5.1.4 循环结构

在循环结构中，可以反复执行指定脚本，其中包括了各类循环语句和应用于循环结构的中断语句，具体内容参考表 5.11。

表 5.11 循环结构

语句	用途简介	语句	用途简介
while	while 循环	for in	for in 循环
do while	do while 循环	break	中断整个循环
for	for 循环	continue	中断某次循环

while、do while

语句 while 首先判断条件 Condition，只有当条件 Condition 成立（即返回值为 true）时，执行循环体中指定的脚本 Statement，同时对条件中的变量进行一定的变更，回到语句 while 的头部继续检测条件以决定是否执行循环体中的脚本；否则跳出语句 while 的循环体。语句 while 的表达方式与语句 if 基本类似，两者的区别在于语句 if 最多只能运行一次，而语句 while 可以反复运行多次，直到条件不成立（返回值为 false）。语句 while 采用语法如下：

```
while( Condition ) {
  Statement;
}
```

其中参数 Condition 为语句 while 用于判断的条件，当条件成立（即返回值为 true）即执行语句 while 中的指定脚本 Statement（值得注意的是，在该指定脚本中必须包含条件 Condition 中变量改变的赋值语句）。

在使用语句 while 之前，首先要初始化变量，如条件中的变量 i 以及脚本中需要的变量 sum；然后应用语句 while 并且设置其用于检测的条件。在循环体中尤为重要的是条件变量 i 的重新定义或者变更，如果在语句中没有该脚本语句，那么一旦条件成立就构成“死循环”（即无穷循环）。

```
i = 1;
sum = 0;
```

```
while (i<5) {
  sum = sum+i;
  i = i+1;
  trace (sum);
}
```

上述所有脚本运行过程中的变量变化以及相应的输出内容可以参考表 5.12 中的内容。

表 5.12 语句 while 运行状况

运行状况	变量 sum	变量 i	输出内容
运行前	0	1	
1	1	2	1
2	3	3	3
3	6	4	6
4	10	5	10
跳出循环	10	5	

语句 do while 与语句 while 在本质上没有任何区别，都是对条件进行判断后相应运行指定脚本。但是语句 do while 与语句 while 有细微差别——语句 do while 先运行指定脚本后判断条件，而语句 while 正好相反，是先判断条件后运行指定脚本。所以在运行前如果条件不成立，语句 while 是不执行指定脚本的，而语句 do while 则执行一次指定脚本。语句 do while 采用语法如下：

```
do {
  Statement;
} while ( Condition );
```

由于语句 do while 与语句 while 基本相似，所以在语句 while 中的例子也可以由语句 do while 完成。

for

语句 for 可以说是对语句 while 进行了精简，将条件以及其中变量的初始化、变更都放置在语句 for 中作为多重参数。因此语句 for 采用语法如下：

```
for( Init; Condition; Next ) {
  Statement;
}
```

其中参数 Init 为循环体变量的初始化，通常以变量赋值的形式出现；参数 Condition 为循环体判断的条件，在循环时首先进行条件判断，当条件成立（即返回值为 true）时就执行循环体中指定脚本，否则跳出循环体；参数 Next 为循环体变量的循环步长，通常以 ++（递加）或--（递减）运算符进行重新赋值。

当然语句 for 中的参数 Init、Next 也可以移动到其他位置，这样整个语句的结构与语句 while 基本无差别，如下所示：

```
Init;
for( ; Condition; ) {
```

```
  Next;
  Statement;
}
```

在语句 while 中所列举的脚本内容，可以应用语句 for 进行改写，编译这段脚本程序得到的结果与语句 while 的完全相同，可以参考表 5.12。经语句 for 改写后的脚本程序内容如下：

```
for ( i=1, sum=0; i<5; i++ ) {
  sum = sum+i;
  trace (sum);
}
```

最后需要强调的是语句 for 与语句 while 一样先进行条件检测后执行指定脚本，而不像语句 do while 先执行指定脚本后进行条件检测。当检测条件不成立（即返回值为 false）时，就完全跳出整个语句 for 的循环体。

for in

语句 for in 应用于判断迭代对象 Object 中是否存在变量 Variable，条件成立返回布尔值 true，同时使得迭代对象中每个变量都执行指定脚本 Statement，否则跳出循环体。因此，语句 for in 采用的语法如下：

```
for( Variable in Object ){
  Statement;
}
```

下面举例说明语句 for in 的具体使用情况。假设在数组 myObject 中设置了 3 个属性 name、age、city 以及相应的属性值。利用 for in 循环反复跟踪数组对象 myObject 中的各个属性值。

```
myObject = { name:'Tara', age:27, city:'San Francisco' };
for (name in myObject) {
  trace ("myObject." + name + " = " + myObject[name]);
}
```

显然属性 name 存在于数组对象 myObject 中，因此编译时也就反复执行指定脚本跟踪数组对象 myObject 中的各个属性，具体输出内容如下：

```
myObject.name = Tara
myObject.age = 27
myObject.city = San Francisco
```

以上所举的例子是关于数组对象，下面再列举关于影片剪辑的 for in 循环应用。假设在动画主时间轴中放置了影片剪辑 YXL，其中包含了影片剪辑 ABC、123。影片剪辑 ABC、123 都是关键帧动画，依次放置为英文字母、阿拉伯数字。

```
for (ABC in YXL) {
  YXL[ABC].gotoAndStop(3);
}
```

上述脚本编译时首先判断影片剪辑实例 ABC 是否存在于影片剪辑 YXL 中，由假设的

文字内容可以得知返回的布尔值一定是 true，执行指定脚本使得影片剪辑 YXL 中的所有影片剪辑（包括 ABC、123）中的播放指针都跳转到第 3 帧处停止。

break、continue

脚本语句 break 通常出现在语句 for、for…in、do...while 或 while 等循环体中，用于中断语句的循环运行，从而使得语句在判断条件成立的情况下一旦符合某一条件时就直接中断循环、跳出循环体。因此脚本语句 break 通常与语句 if 联合出现。下面以语句 for 为例，讲述语句 break 的使用，具体内容如下：

```
for ( i=1, sum = 0; i<5; i++ ){
  if ( i == 3 ) {
    break;
  }
  sum = sum+i;
  trace (sum);
}
```

上述所有脚本运行过程中的变量变化以及相应的输出内容可以参考表 5.13 中的内容。

表 5.13 语句 break 对语句 for 的影响

运行状况	变量 sum	变量 i	输出内容
运行前	0	1	
1	1	2	1
2	3	3	3
跳出循环	3	3	

脚本语句 continue 与脚本语句 break 相似，同样应用于语句 for、for…in、do...while 或 while 等循环体中，但是脚本语句 continue 并不是完全中断循环运行，而是中断当前循环体中语句 continue 后的指定脚本，跳到语句的相应位置。

对于不同的语句，脚本语句 continue 中断本次语句后跳到语句的部位是不同的，具体内容的比较参考表 5.14。

表 5.14 语句 continue 对循环语句的影响

语句	continue 中断后跳到的位置
while	中断剩下循环体后跳到循环语句头部（即条件 Condition）
do while	中断剩下循环体后跳到循环语句尾部（即条件 Condition）
for	中断剩下循环体后跳到循环赋值位置（即步长赋值 Next）
for in	中断剩下循环体后运行下一对应变量的循环头部

因此脚本语句 continue 通常与语句 if 联合出现。下面以语句 for 为例讲述语句 continue 的使用，具体内容如下：

```
for ( i=1, sum = 0; i<5; i++ ){
  if ( i == 3 ) {
    continue;
  }
```

```
  sum = sum+i;
  trace (sum);
}
```

上述所有脚本运行过程中的变量变化以及相应的输出内容可以参考表 5.15 中的内容。

表 5.15　语句 continue 对语句 for 的影响

运行状况	变量 sum	变量 i	输出内容
运行前	0	1	
1	1	2	1
2	3	3	3
3	3	4	
4	7	5	7
跳出循环	7	5	

5.2　连续反馈按钮

所谓连续反馈按钮，就是当您在按钮上按住鼠标左键时，可使动画中的指定对象持续发生变化；但是当在按钮上释放鼠标左键或离开按钮区域时，动画中的指定对象随即停止变化。

本节利用两种方法、通过两个范例讲解连续反馈按钮的制作，并从中体会 Flash 中按钮元件的具体使用方法；并对按钮的响应事件进行详细剖析，更加清楚地掌握 Flash 中按钮交互性的强大功能。同时，还讲解了动画播放过程中的播放指针跳转和影片剪辑实例的属性设置等内容。

5.2.1　按钮响应事件

语句 on 是按钮元件实例所特有的脚本语句。通常，选中按钮实例对其添加脚本 Stamement 时，会发现脚本 Stamement 自动添加在语句 on 程序块中，具体形式如下：

```
on ( mouseEvent ) {
  Stamement;
}
```

其中 mouseEvent 就是响应按钮的鼠标事件，默认为“释放（release）”，也就是在 Windows 操作系统中常用的鼠标单击。

选中语句 on 时，在参数设置区中可以选择 8 种不同的响应事件，如图 5.3 所示。只要选中事件栏中各个响应事件的复选框，就会在语句 on 中添加相应的响应事件。

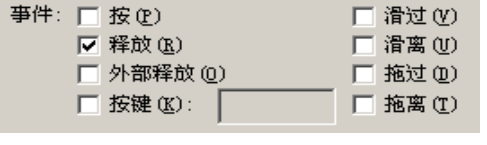

图 5.3　语句 on 的参数设置区

或许，你对语句 on 参数设置区中的响应事件不太清楚，各个事件究竟代表怎样的按钮操作呢？关于按钮响应事件的具体内容可以参考表 5.16，而其实际应用将在以后的范例动画制作中加以剖析！

表 5.16 按钮的响应事件

响应事件	事件代码	说明
按	press	在按钮上按住鼠标左键
释放	release	在按钮上按住鼠标左键后，释放左键
外部释放	releaseOutside	在按钮上按住鼠标左键，移动到按钮区域外再释放鼠标左键
按键	keyPress	按下键盘上预设的按键
滑过	rollOver	鼠标光标由按钮区域外移动到按钮区域内
滑离	rollOut	鼠标光标由按钮区域内移动到按钮区域外
拖过	dragOver	在按钮上按住鼠标左键后，移动到按钮区域外再返回按钮区域内
拖离	dragOut	在按钮上按住鼠标左键后，移动到按钮区域外

5.2.2 播放指针跳转

语句 goto 用于控制播放指针在动画中的跳转，通常采用如下语法：

```
gotoAndPlay ( Scene, Frame );
```

或

```
gotoAndStop ( Scene, Frame );
```

但是，由于参数 Scene 和 Frame 的不同，该语句又会呈现不同的形式。

选中语句 gotoAndPlay 时，可以在其参数设置区先确定是使用“转到并播放（gotoAndPlay）”还是使用“转到并停止（gotoAndStop）”，同时还应该在“场景”、“类型”以及“帧”列表框中进行设置，如图 5.4 所示。

图 5.4 语句 goto 的参数设置区

当“场景”列表框中设置为“<当前场景>”时，语句 goto 将被简化为如下形式：

```
gotoAndPlay ( Frame );
```

或

```
gotoAndStop ( Frame );
```

其实，语句 goto 还有 4 种更简化的形式。但是，这 4 种形式均由“gotoAndStop (Scene, Frame);”进行简化得到，具体形式可以参考表 5.17。

表 5.17　表格名称

列表选项	代码形式	说明
<下一场景>	nextScene();	播放指针跳转到下一个场景的第 1 帧，并停止播放
<上一场景>	prevScene();	播放指针跳转到前一个场景的第 1 帧，并停止播放
<下一帧>	nextFrame();	播放指针移动到当前场景的下一帧，并停止播放
<上一帧>	prevFrame();	播放指针移动到当前场景的前一帧，并停止播放

5.2.3　设置属性

语句 setProperty 用于设置影片剪辑实例的大小、位置等属性，利用脚本实现对影片剪辑实例的精确控制。语句 setProperty 使用语法如下：

```
setProperty ( MCName, Property, Value );
```

选中语句 setProperty 可以在参数设置区的“属性”列表框中选择需要调整的属性，在“目标”文本框中输入影片剪辑实例的路径及名称，而在“值”文本框中输入确定属性调整后的数值，如图 5.5 所示。

图 5.5　语句 setProperty 的参数设置区

对于影片剪辑（MC）而言，使用语句 setProperty 可以调整影片剪辑实例的多个属性，这些属性的具体内容可以参考表 5.18。

表 5.18　影片剪辑的属性

属性	属性描述	度量单位	设置范围
_alpha	影片剪辑的透明度	%	0～100
_rotation	影片剪辑的旋转角度	度	
_visible	影片剪辑的可视性		true 或 false
_height、_width	影片剪辑的高度、宽度	像素	正整数
_x、_y	影片剪辑的 X、Y 轴坐标	像素	整数
_xscale、_yscale	影片剪辑的 X、Y 轴缩放系数	%	任意数

使用语句 setProperty 时，应该预先完成影片剪辑实例名称的设置。否则，在设置语句 setProperty 时将不能正确地输入影片剪辑的实例名称。单击语句 setProperty 参数设置区中的“目标”文本框，此时其下方的插入目标路径图标激活。单击该图标，将弹出“插入目标路径”对话框，从中选择影片剪辑实例名称，同时选择“记号”及“模式”栏中的单选按钮即可，如图 5.6 所示。单击“确定”按钮后，在语句 setProperty 的“目标”文本框中就自动输入了影片剪辑实例的路径及名称。

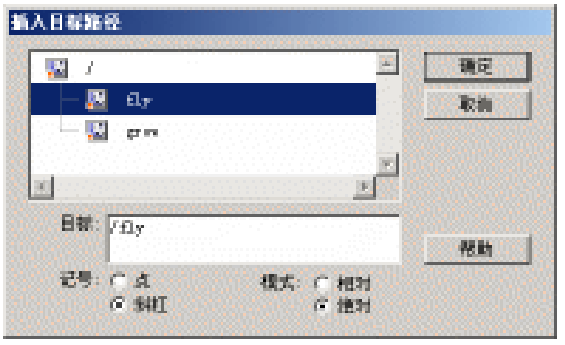

图 5.6 “插入目标路径”对话框

语句 setProperty 用于设置影片剪辑实例的各项属性。如果要在获取影片剪辑实例的各项属性后,再有目的地调整其中的部分属性,那么如何准确地获取影片剪辑实例的属性呢？此时就应该使用 Flash 提供的函数 getProperty，其使用语法如下：

```
getProperty ( MCName, Property )
```

5.2.4 范例：缩放蝴蝶图案

当光标定位在左侧的按钮上并按住鼠标左键时，画面中的蝴蝶图案将会动态地逐渐放大；当光标定位在右侧的按钮上并按住鼠标左键时，画面中的蝴蝶图案将会动态地逐渐缩小，但到一定尺寸后将不再发生变化，如图 5.7 所示。如果要将蝴蝶图案恢复至原始尺寸大小，单击中间的按钮即可。

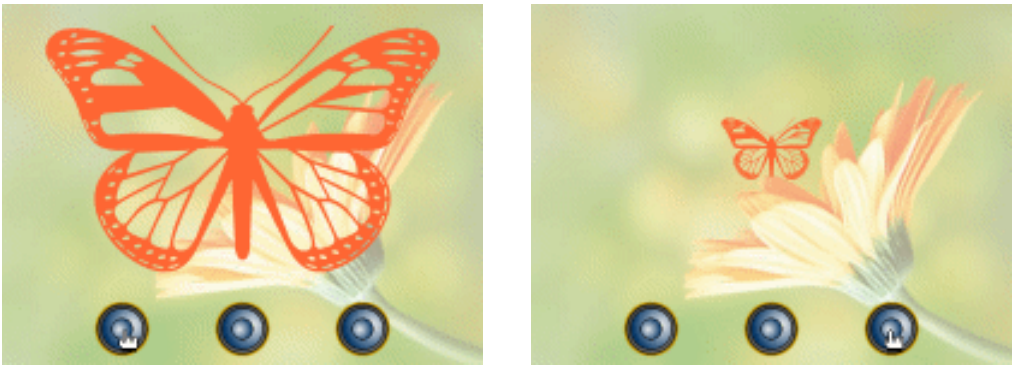

图 5.7 效果预览

在动画浏览过程中可以尝试如下操作：在左、右两侧的按钮上按住鼠标左键后移动到按钮区域外再返回到按钮区域内，查看这个过程中蝴蝶图案的缩放状况。

当在按钮上按住鼠标左键后，就会持续不断地运行指定的脚本内容，控制蝴蝶图案连续发生尺寸的变化。本例主要通过按钮元件 btn 控制影片剪辑 mcGrow 中播放指针的跳转，再由影片剪辑的播放不受主时间轴影响的特点促使影片剪辑 mcFly 不断放大或缩小。因此，本例中最为重要的动画元件就是影片剪辑 mcGrow。

通过上述简短的分析后，已经能够比较清楚地了解连续反馈按钮的制作，现在就按照

分析的思路进行连续反馈按钮的动画制作，步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 300px×400px。单击“确定”按钮确认后，创建影片剪辑 mcFly，在舞台中央处绘制蝴蝶图案，如图 5.8 所示。

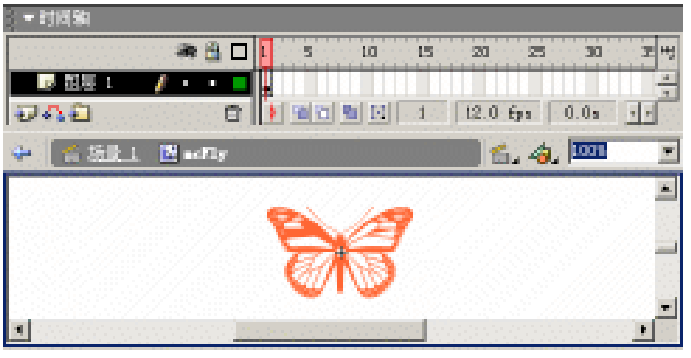

图 5.8　影片剪辑 mcFly

（2）创建按钮元件 btn，在按钮编辑状态的“弹起”帧中绘制图形，如图 5.9 所示。这样，没有画面轮换的简单按钮就制作完成了。

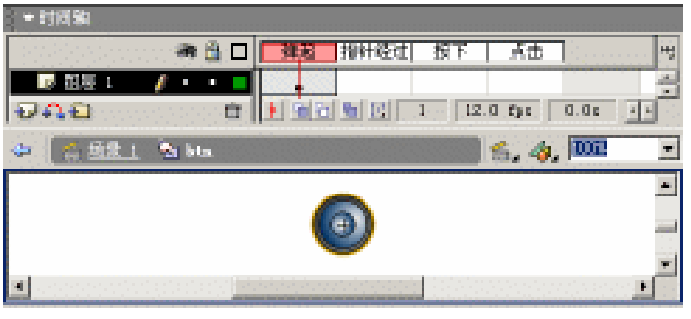

图 5.9　按钮元件 btn

（3）切换到场景编辑状态，将图层名称更改为“背景”，导入 JPG 图像“花卉”；添加图层“蝴蝶”，将“库”面板中的影片剪辑 mcFly 拖放到舞台上，并在“属性”面板中设置该影片剪辑的实例名称为 fly，如图 5.10 所示；添加图层“按钮”，连续三次将“库”面板中的按钮元件 btn 拖放到舞台上，并使之处于舞台底端，如图 5.11 所示。

（4）创建影片剪辑 mcGrow，在该影片剪辑中并不绘制任何图形，只用于控制蝴蝶图案的缩放。将图层名称更改为“标签”，单击第 1 帧后在“属性”面板的帧文本框中输入帧标签 Pause，如图 5.12 所示。此时，在时间轴中的第 1 帧处显示一个红色的旗帜。同样，在第 10、20 帧处设置帧标签 growUp、growDown。

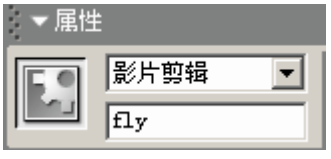

图 5.10 设置实例名称

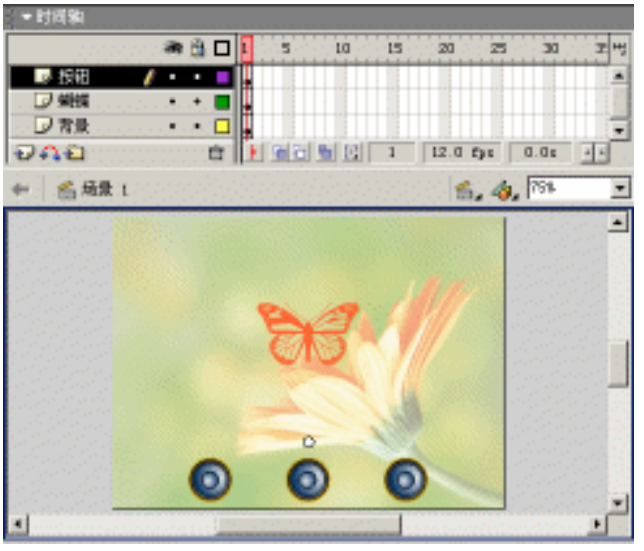

图 5.11 舞台对象布置

（5）添加图层“ 脚本 ”，在时间轴的多处设置帧的脚本内容。设置帧脚本后的关键帧处显示字母 a，设置帧脚本的各帧分布如图 5.13 所示，各帧中的具体脚本内容如下。

图 5.12 设置帧标签

图 5.13 影片剪辑 mcGrow

第 1 帧中的脚本内容如下：

```
stop ();
```

提示：影片剪辑的播放是不受时间轴的影响的，该语句主要是为了防止影片剪辑 mcGrow 的自动播放。

第 10 帧中的脚本内容如下：

```
temp = getProperty ("/fly",_xscale) * 1.1;
setProperty("/fly", _xscale, temp);
setProperty("/fly", _yscale, temp);
```

第 11 帧中的脚本内容如下：

```
gotoAndPlay("growUp");
```

提示：在第 10 帧中的脚本调整影片剪辑实例 fly 的缩放系数，使之在原有的基础上放大 10%；与第 11 帧中的脚本联合作用，使得播放指针始终处于第 10～11 帧之间，从而构成无限循环。

第 20 帧中的脚本内容如下：

```
temp = getProperty ("/fly" , _xscale) * 0.9;
if (temp < 20) {
  temp = 20;
}
setProperty("/fly", _xscale, temp);
setProperty("/fly", _yscale, temp);
```

第 21 帧中的脚本内容如下：

```
gotoAndPlay("growDown");
```

提示：在第 20 帧中的脚本调整影片剪辑实例 fly 的缩放系数，使之在原有的基础上缩小 10%；与第 21 帧中的脚本联合作用，使得播放指针始终处于第 20～21 帧之间，从而构成无限循环。
其中 if 语句控制影片剪辑实例 fly 缩放系数的最小数值，以防止影片剪辑实例 fly 缩小到一定程度后尺寸变化不明显。

（6）切换到场景编辑状态，将“库”面板中的影片剪辑 mcGrow 拖放到舞台上，并在“属性”面板中设置实例名称为 grow。由于影片剪辑 mcGrow 中没有绘制任何图形，因此当没有选中时，该影片剪辑实例在舞台上显示为白色圆形。

左侧的按钮起到放大的作用，其中具体的脚本内容如下：

```
on (press, dragOver) {
  tellTarget ("/grow") {
    gotoAndPlay("growUp");
  }
}
on (release, dragOut) {
  tellTarget ("/grow") {
    gotoAndStop("Pause");
  }
}
```

提示：当按钮响应事件 press、dragOver 触发后，将影片剪辑实例 grow 的播放指针跳转到帧标签 growUp 并进行播放；
当按钮响应事件 release、dragOut 触发后，将影片剪辑实例 grow 的播放指针跳转到帧标签 Pause 并停止播放。

中间的按钮起到复原的作用，其中具体的脚本内容如下：

```
on (release) {
  setProperty("/fly", _xscale, 100);
  setProperty("/fly", _yscale, 100);
}
```

提示：由于影片剪辑实例 fly 被拖放到舞台上后并没有进行任何变形，因此其 X、Y 轴上的缩放系数就应该是 100。上述脚本表示：当按钮响应事件 release 触发后，就直接将影片剪辑实例 fly 还原到原始尺寸。

右侧的按钮起到缩小的作用，其中具体的脚本内容如下：

```
on (press, dragOver) {
  tellTarget ("/grow") {
    gotoAndPlay("growDown");
  }
}
on (release, dragOut) {
  tellTarget ("/grow") {
    gotoAndStop("Pause");
  }
}
```

提示：当按钮响应事件 press、dragOver 触发后，将影片剪辑实例 grow 的播放指针跳转到帧标签 growDown 并进行播放；
当按钮响应事件 release、dragOut 触发后，将影片剪辑实例 grow 的播放指针跳转到帧标签 Pause 并停止播放。

至此，利用连续反馈按钮对蝴蝶图案进行缩放的动画就全部制作完成，按快捷键 Ctrl+Enter 测试动画效果。

5.2.5 范例：移动圣诞头像

当光标定位在左下角的方向盘上按住鼠标左键时，可以尝试着移动到各个方向按钮上查看圣诞头像的移动情况。当在某方向按钮上停顿时，圣诞头像就沿着箭头的指向不断移动，如图 5.14 所示。单击方向盘中的方形按钮，可以将圣诞头像复原到舞台中央。

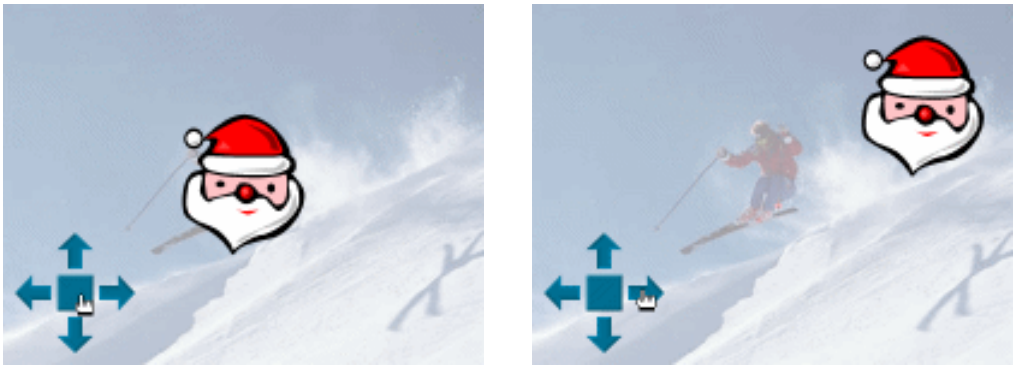

图 5.14 效果预览

本例动画中的连续反馈按钮并不是采用上一节中影片剪辑的方法得到，而是利用按钮元件所特有的选项“音轨作为菜单项”（英文版中为 Track as Menu Item）进行设置。当然，只认识到这一点是远远不够的，具体的使用方法在下面的制作步骤中加以体会：

（1）新建文档后，调用“文档属性”对话框，调整尺寸为 400px×300px。将图层名称更改为“背景”，导入 JPG 图像“滑雪”；添加图层“头像”后，在舞台上绘制圣诞头像的图形，如图 5.15 所示，并将之转换为影片剪辑 mcMan，同时在“属性”面板中设置其实例名称为 man。

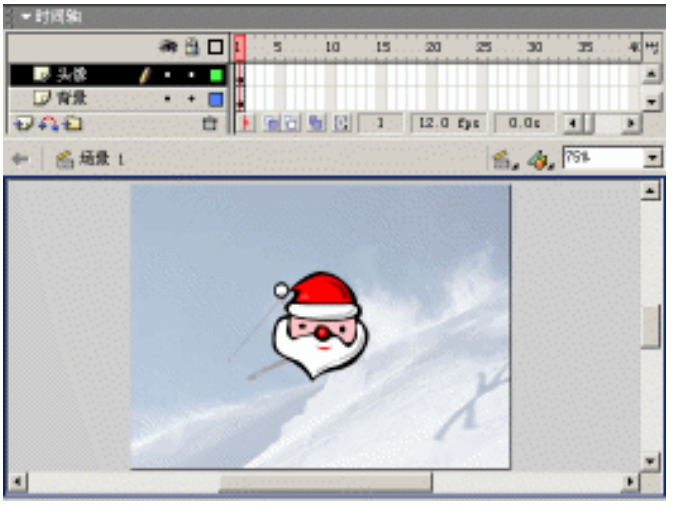

图 5.15　放置舞台对象

（2）创建按钮元件 btnRest，在按钮元件编辑状态的“弹起”帧中绘制方形图形，如图 5.16 所示。这样，没有图形轮换的按钮元件 btnRest 就制作完成。

（3）创建按钮元件 btnArrow，在舞台上绘制箭头图形，如图 5.17 所示。这样，没有图形轮换的按钮元件 btnArrow 就制作完成。

图 5.16　按钮元件 btnRest

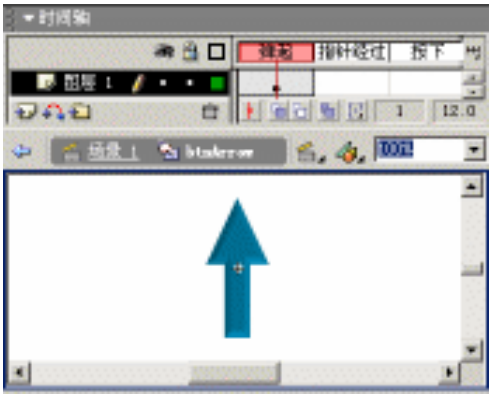

图 5.17　按钮元件 btnArrow

（4）创建影片剪辑 mcUp，将“库”面板中的按钮元件 btnArrow 拖放到舞台上，并在“属性”面板中设置按钮元件实例的选项为“以菜单条方式”，如图 5.18 所示。

图 5.18 设置按钮选项

选中按钮元件 btnArrow 实例的情况下，在“动作”面板中设置脚本内容如下：

```
on (press, dragOver) {
  temp = _root.Man._y - 5;
  if (temp < _root.man._height/2) {
    temp = _root.man._height/2;
  }
  setProperty(_root.Man, _y, temp);
}
```

提示：当按钮响应事件 press、dragOver 触发后，将影片剪辑实例 man 向上移动 5 个像素。其中 if 语句用于控制影片剪辑实例 man 向上移动时 Y 轴的最小值，以防止影片剪辑实例 man 移动到动画区域外。

右击第 1 帧，选择弹出菜单中的“复制帧”命令；添加图层后右击新图层的第 2 帧，选择弹出菜单中的“粘贴帧”命令。此时，时间轴中关键帧的分布如图 5.19 所示。由于这两个关键帧是通过复制帧–粘贴帧得到的，因此在两个关键帧中按钮元件所在的位置是完全重叠的。

（5）复制“库”面板中的影片剪辑元件 mcUp，得到影片剪辑 mcDown 并进行编辑，将各关键帧中的按钮元件实例进行垂直翻转，如图 5.20 所示，同时将按钮元件实例的脚本内容进行如下更改：

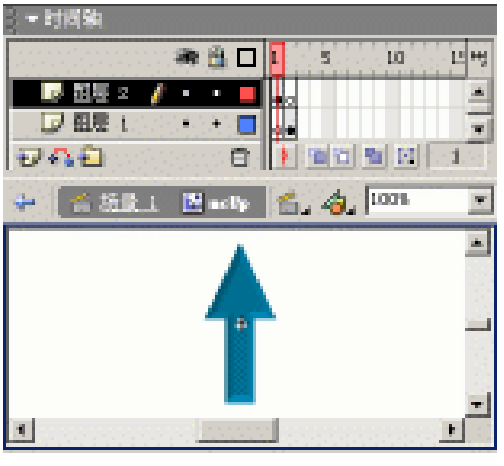

图 5.19 影片剪辑 mcUp

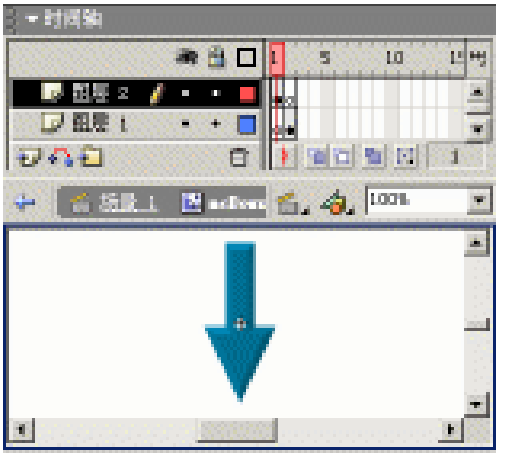

图 5.20 影片剪辑 mcDown

```
on (press, dragOver) {
  temp = _root.Man._y + 5;
  if (temp > Stage.height-_root.Man._height/2) {
```

```
    temp = Stage.height-_root.Man._height/2;
  }
  setProperty(_root.Man, _y, temp);
}
```

提示：当按钮响应事件 press、dragOver 触发后，将影片剪辑实例 man 向下移动 5 个像素。其中 if 语句用于控制影片剪辑实例 man 向上移动时 Y 轴的最大值，以防止影片剪辑实例 man 移动到动画区域外。

（6）复制“库”面板中的影片剪辑元件 mcUp，得到影片剪辑 mcLeft 并进行编辑，将各关键帧中的按钮元件实例向左旋转 90°，如图 5.21 所示，同时将按钮元件实例的脚本内容进行如下更改：

```
on (press, dragOver) {
  temp = _root.Man._x - 5;
  if (temp < _root.man._width/2) {
    temp = _root.man._width/2;
  }
  setProperty(_root.Man, _x, temp);
}
```

提示：当按钮响应事件 press、dragOver 触发后，将影片剪辑实例 man 向左移动 5 个像素。其中 if 语句用于控制影片剪辑实例 man 向左移动时 X 轴的最小值，以防止影片剪辑实例 man 移动到动画区域外。

（7）复制“库”面板中的影片剪辑元件 mcUp，得到影片剪辑 mcRight 并进行编辑，将各关键帧中的按钮元件实例向右旋转 90°，如图 5.22 所示，同时将按钮元件实例的脚本内容进行如下更改：

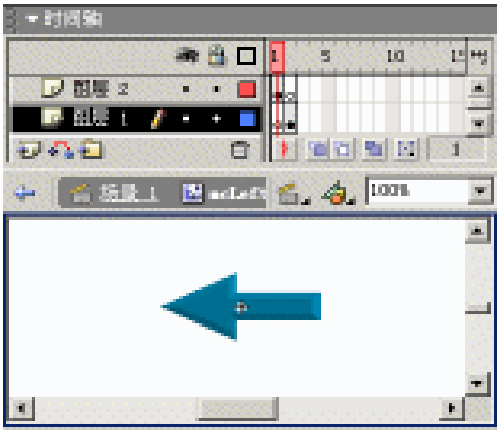

图 5.21　影片剪辑 mcLeft

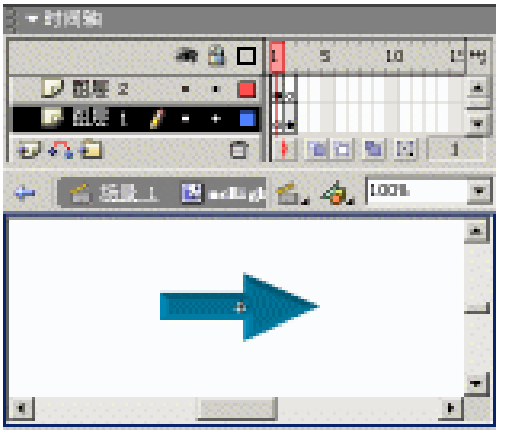

图 5.22　影片剪辑 mcRight

```
on (press, dragOver) {
  temp = _root.Man._x + 5;
  if (temp > Stage.width-_root.man._width/2) {
    temp = Stage.width-_root.man._width/2;
```

```
  }
  setProperty(_root.Man, _x, temp);
}
```

提示：当按钮响应事件 press、dragOver **触发后，将影片剪辑实例** man **向右移动** 5 **个像素。其中** if **语句用于控制影片剪辑实例** man **向左移动时** X **轴的最大值，以防止影片剪辑实例** man **移动到动画区域外。**

（8**）切换到场景编辑状态，添加图层“元件”，将“库”面板中的影片剪辑** mcUp、mcDown、mcLeft、mcRight **以及按钮元件** btnRest **拖放到舞台上，构成游戏中的方向盘形式，如图** 5.23 **所示。**

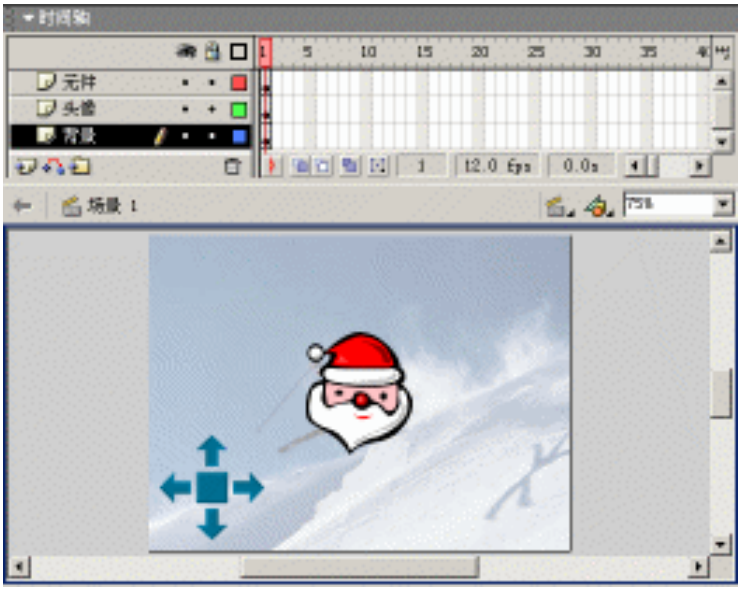

图 5.23 **放置方向盘**

选中按钮元件实例 btnRest **的情况下，在“动作”面板中设置脚本内容如下：**

```
on (release) {
  setProperty(_root.Man, _x, Stage.width/2);
  setProperty(_root.Man, _y, Stage.height/2);
}
```

提示：由于影片剪辑实例 man **至于舞台中央，其坐标就应该是舞台尺寸的一半，即** Stage.width/2 **和** Stage.height/2**。因此，按钮** btnRest **响应事件** release **触发后，就强制将影片剪辑实例** man **置于舞台中央。**

至此，利用连续反馈按钮的另一种方法控制圣诞头像移动就制作完成了，按快捷键 Ctrl+Enter **测试动画。在各个方向按钮上进行按钮操作，查看圣诞头像的移动效果？**

5.3　动态绘制图案

为了提高脚本编写的学习兴趣，现在就利用脚本程序实现某些图案的动态绘制，扩展 Flash 动画制作中的思路。首先，还是先来了解一下影片剪辑的事件设置和复制及移除等脚本编写知识，然后再通过两个范例动画展示图案的动态绘制。

5.3.1　设置影片剪辑事件

语句 onClipEvent 是影片剪辑实例所特有的脚本语句，通常，对影片剪辑实例添加脚本 Stamement 时，会发现脚本 Stamement 自动添加在语句 onClipEvent 程序块中，具体形式如下：

```
onClipEvent ( Event ) {
   Stamement;
}
```

其中 Event 就是响应影片剪辑的事件，默认为“加载（load）”，也就是当动画中载入影片剪辑时的响应事件。

选中语句 onClipEvent 时，在其参数设置区中可以选择 9 种不同的响应事件，如图 5.24 所示。只要选中其中的单选按钮，就会在语句 onClipEvent 中更改相应的响应事件。

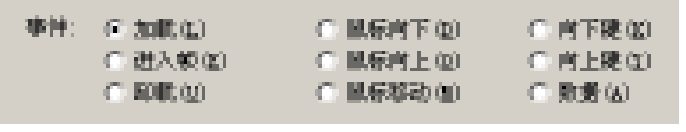

图 5.24　语句 onClipEvent 的参数设置区

关于影片剪辑响应事件的具体内容可以参考表 5.19，而其实际应用将在以后的范例动画制作中加以剖析！

表 5.19　影片剪辑的响应事件

事件选项	触发事件	事件描述
加载	load	影片剪辑载入到时间轴中
进入帧	enterFrame	影片剪辑处于时间轴的播放帧中（近似于在影片剪辑中附加指定脚本）
卸载	unload	影片剪辑从时间轴中移除
鼠标向下	mouseDown	影片剪辑处于时间轴中时，按住鼠标左键
鼠标向上	mouseUp	影片剪辑处于时间轴中时，释放鼠标左键
鼠标移动	mouseMove	影片剪辑处于时间轴中时，移动鼠标光标
向下键	keyDown	影片剪辑处于时间轴中时，按住键盘键
向上键	keyUp	影片剪辑处于时间轴中时，释放键盘键
数据	date	利用语句 loadVariables、loadMovie 获取数据 Date。当 loadVariables 载入变量时只响应一次；而 loadMovie 随每个数据段的获取，可重复响应

5.3.2 影片剪辑的复制及移除

通过语句 duplicateMovieClip，可以对已有的影片剪辑实例进行复制得到更多的影片剪辑实例，以实现在动画中存在多个相同影片剪辑实例的效果，具体语法如下：

```
duplicateMovieClip( MCName, newName, Depth );
```

其中参数 MCName 就是已有影片剪辑实例的路径及名称。参数 newName 则是复制得到的影片剪辑实例名称。影片剪辑实例名称是独一无二的，也是确定影片剪辑的惟一标识。参数 Depth 就是影片剪辑实例的堆栈深度（即上下叠放次序），用于确定影片剪辑实例间发生重叠时如何进行显示。

选中语句 duplicateMovieClip，在其参数设置区的“目标”文本框中输入已有影片剪辑的实例名称；“新名称”文本框中输入复制得到的影片剪辑实例名称；而“深度”文本框中输入复制得到的影片剪辑实例的堆栈深度，如图 5.25 所示。

图 5.25 语句 duplicateMovieClip 的参数设置区

通常 duplicateMovieClip 语句复制得到的影片剪辑实例就直接置于舞台上。如果不再需要这些复制得到的影片剪辑，就应该及时移除，具体使用方法如下：

```
removeMovieClip ( newName );
```

5.3.3 范例：绘制星光图案

浏览动画时可以看到最终图案为星形网格，但是由于该图案是由单个方框逐渐演变而成，如图 5.26 所示，其过程也就显得较为新奇。

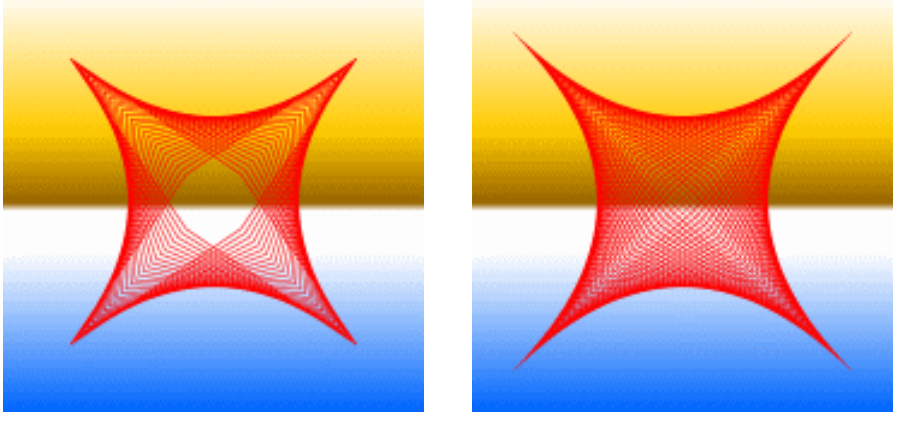

图 5.26 效果预览

动画过程中的星光绘制就是通过播放指针的跳转（gotoAndPlay）实现的。不过由语句

if 确定动画的播放是否终止。如果仔细揣摩星光的绘制过程，可以发现星光其实就是由多个方框通过变形得到的。具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，调用“ 文档属性 ”对话框，调整尺寸为 350px×350px。将图层名称更改为 “ 背景 ”，在舞台上绘制线性渐变的矩形填充。

（2）添加图层 “ 元件 ”，在舞台中央绘制红色笔触的方框，并将其旋转 45 ° 后转换为影片剪辑 mcRect，同时在 “ 属性 ” 面板中设置实例名称 rect。

（3）添加图层 “ 脚本 ”，连续按 F6 键插入关键帧 3 次，分别在各个关键帧中设置帧脚本内容。星光图案的舞台布置及时间轴分布如图 5.27 所示。

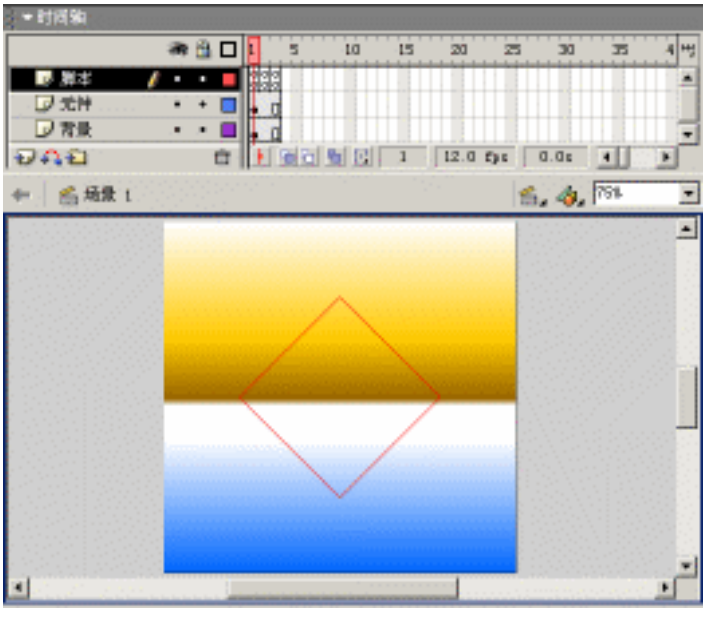

图 5.27　舞台对象布置

第 1 帧中的脚本内容如下：

```
step = 5;
angle = 45;
i = (100-int(step/2))%step;
setProperty("rect", _visible, false);
```

提示：这是各个变量的初始化，其中变量 step 表示缩放步长、变量 angle 表示旋转角度、变量 i 表示方框的调整偏差。最后，将影片剪辑实例 rect 设为不可见，防止已有的影片剪辑实例 rect 影响动画界面。

第 2 帧中的脚本内容如下：

```
temp = 1000-2*i;
duplicateMovieClip("rect", "rect"+temp, temp);
setProperty("rect"+temp, _xscale, 100+i);
setProperty("rect"+temp, _yscale, 100-i);
setProperty("rect"+temp, _rotation, angle);
```

```
temp = 999-2*i;
duplicateMovieClip("rect", "rect"+temp, temp);
setProperty("rect"+temp, _xscale, 100-i);
setProperty("rect"+temp, _yscale, 100+i);
setProperty("rect"+temp, _rotation, angle);
```

提示：利用语句 duplicateMovieClip **复制影片剪辑两次，使得星光中的方框沿着两个方向发生变形。这些变形就是通过语句** setProperty **进行调整的，由于变量** i **的值不断发生变化，所以星光中的方框也发生着相应的调整。**

第 3 **帧中的脚本内容如下：**

```
if (i<100) {
  gotoAndPlay(_currentframe-1);
  i = i+step;
} else {
  stop();
}
```

提示：利用 if **语句确定是否进行播放指针的跳转。其中作为判断条件的变量** i **逐步增加，否则就会形成死循环；属性**_currentframe **表示播放指针所在帧的帧序数，即时间轴上的序数** 1、2、3…

至此，星光图案的动态绘制就制作完成了，按快捷键 Ctrl+Enter **测试动画。调整第** 1 **帧中的变量** step **及** angle**，就会得到不同效果的星光。如果对影片剪辑** mcRect **中的图形进行更改，可以生成更多效果的图案。**

5.3.4 范例：绘制心形图案

本例的动画效果是：动画浏览时，画面中逐渐散出一些圆环。虽然起初并不清楚这些圆环会构成什么图案，但是在最终完成时就会赫然呈现心形图案，如图 5.28 **所示。**

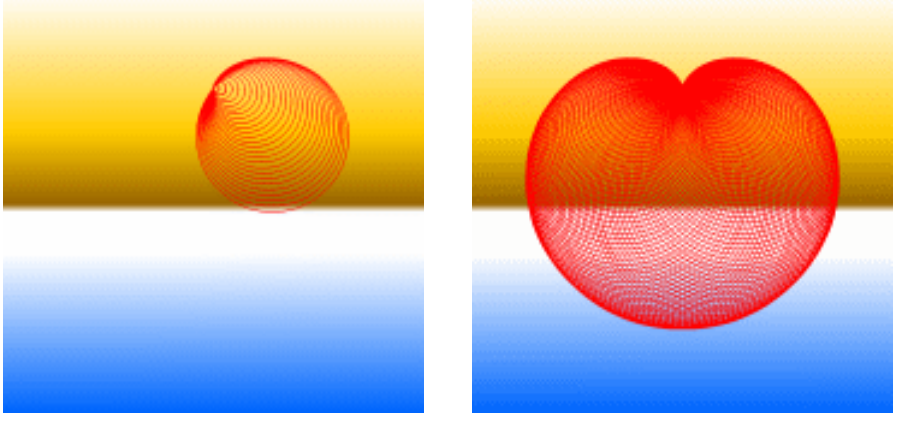

图 5.28 **效果预览**

本例动画中心形图案的绘制并没有采用播放指针跳转的方法，而是利用影片剪辑实例

所特有的语句 onClipEvent 进行编写。

在动画制作前，应该先了解一下心形图案绘制的基本原理。将半径为 r 的虚线圆形（如图 5.29 所示）进行 N 等分，则每部分的角度 theta 为 2π/N（脚本表示：theta=2*Math.PI/N）。

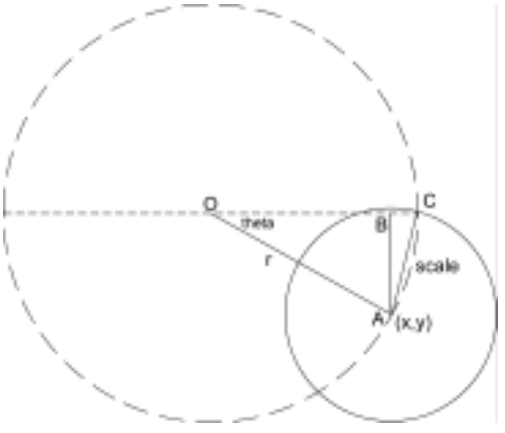

图 5.29　心形图案绘制原理图

因此，角度∠AOC 就是 theta 的 I 倍；然后以 A 点作为圆心，以 AC 段作为半径绘制圆。当数值 I 变化时，绘制的圆也就发生相应的变化，最终可得到心形图案。

如果虚线圆中圆心 O 的坐标为（0, 0），则 A 点的坐标（x, y）可以表示为：

```
x = OA*cos（∠AOC）=R*cos（I* theta）
y = OA*sin（∠AOC）=R*sin（I* theta）
```

脚本表示：

```
x= R* Math.cos(i*theta)
y= R* Math.sin(i*theta)
```

而线段 AC 的长度可以表示为：

$$AC=\sqrt{AB*AB+BC*BC}=\sqrt{y*y+(R-x)*(R-x)}$$

脚本表示：

```
AC=Math.sqrt((x-r)*(x-r)+y*y)
```

绘制心形图案的具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，调用“文档属性”对话框，调整尺寸为 350px×350px。创建影片剪辑 mcCircle，在舞台中央绘制红色笔触圆环，并利用“信息”面板设置圆环的尺寸为 200px×200px，并以圆环中心点作为参考坐标（0, 0），如图 5.30 所示。

（2）按 F11 键调用“库”面板，右击影片剪辑 mcCircle，选择弹出菜单中的“链接”命令弹出“链接属性”对话框，在“标识符”文本框中输入该影片剪辑的表示名称 circle，同时选中“链接”栏中的“为动作脚本导出”和“在第一帧导出”复选框，如图 5.31 所示，单击“确定”按钮确认。

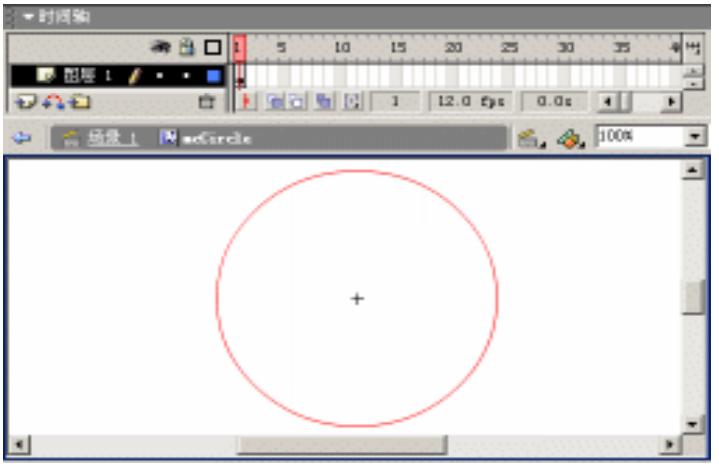

图 5.30 影片剪辑 mcCircle

图 5.31 “链接属性”对话框

（3）创建影片剪辑 mcHeart，其中不作任何设置。切换到场景编辑状态，将图层名称更改为“背景”，在舞台上绘制线性渐变的矩形填充，如图 5.32 所示。

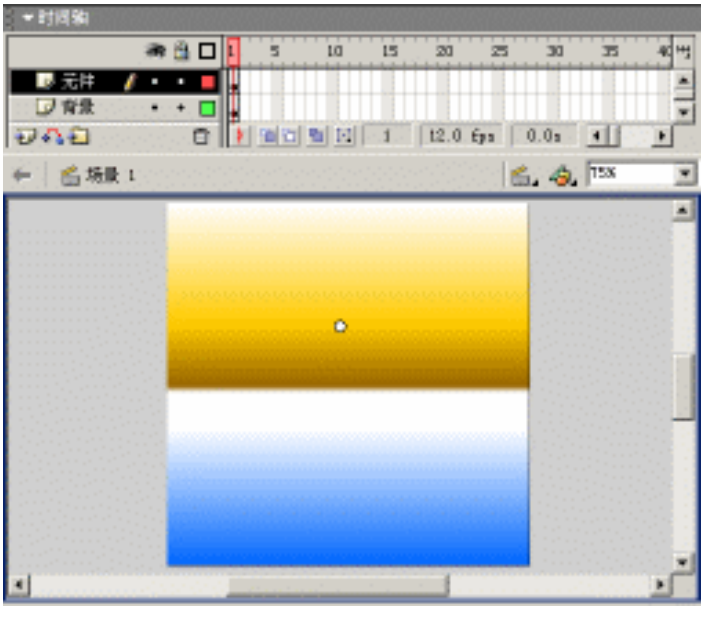

图 5.32 舞台对象布置

（4）添加图层“元件”，将“库”面板中的影片剪辑 mcHeart 拖放到舞台上。由于影片剪辑 mcHeart 中没有任何图形，因此在舞台上表现为一个圆点。选中影片剪辑 mcHeart（圆点中呈十字显示）实例后，在“动作”面板中设置脚本内容如下：

```
onClipEvent (load) {
  r=50;
  n=120;
  i=0;
  this._x = 175;
  this._y=130;
  this._rotation=-90;
  theta=2*Math.PI/n;
}
```

提示：加载影片剪辑 mcHeart 时，进行变量初始化并调整实例的坐标及旋转角度。其中变量 r 表示虚线圆的半径、变量 n 表示虚线圆的等分数目、变量 i 为计数变量、变量 theta 为角度步长。

```
onClipEvent (enterFrame) {
  if (i<=n) {
    x=r*Math.cos(i*theta);
    y=r*Math.sin(i*theta);
    scale=Math.sqrt ( (x-r)*(x-r)+y*y );
    this.attachMovie( "circle", "circle"+i, i );
    setProperty( "circle"+i, _x, x);
    setProperty( "circle"+i, _y, y);
    setProperty( "circle"+i, _xscale, scale);
    setProperty( "circle"+i, _yscale, scale);
    i +=1;
  }
}
```

提示：影片剪辑 mcHeart 处于播放帧时，通过 if 语句确定是否添加圆环。如果变量 i 小于变量 n，那么就链接“库”面板中标识符为 circle 的影片剪辑生成圆环，再确定圆环在虚线圆上的坐标及该圆环的缩放系数。

至此，心形图案的动画制作就全部制作完成了，按快捷键 Ctrl+Enter 测试动画。

思考：将星光图案及心形图案进行对比，是否体会到影片剪辑事件 load 和 enterFrame 使用的优点。以上两种方法可以相互转换使用，这样，既能熟悉图案动态绘制的动画制作，又能强化动画脚本编写的使用程度。

5.4　鼠标跟随效果

所谓鼠标跟随，就是动画中的部分对象与鼠标光标之间存在关联。简而言之，就是鼠标光标处于什么位置，那么动画对象也就相应出现在该位置处。由此可见，鼠标跟随效果的制作通常有两种方法：其一就是直接使用 startDrag 语句拖曳影片剪辑实例；其二就是先

获取鼠标光标的位置（_xmouse, _ymouse），再设置影片剪辑实例的位置。相对而言，使用 startDrag 语句实现鼠标跟随效果要简单一些。

5.4.1 startDrag、stopDrag

观看各种动画效果时，经常会发现鼠标光标后跟随着一长串的图案，其中大部分动画都是通过 startDrag 语句实现的。由此可见，startDrag 语句具有拖曳影片剪辑实例的功能。实际使用时，startDrag 语句通常呈现以下三种表现形式：

```
startDrag( MCName );
startDrag( MCName, Lock );
startDrag( MCName, Lock, Left, Top, Right, Bottom );
```

选中 startDrag 语句时，在其参数设置区显示 startDrag 语句的相关选项，如图 5.33 所示，其中“目标”文本框中输入要拖曳的影片剪辑的实例名称，此时 startDrag 语句就呈现第一种表现形式。

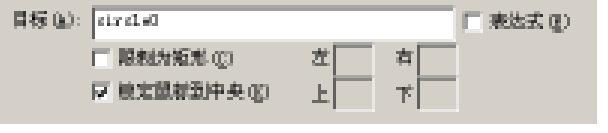

图 5.33 语句 startDrag 的参数设置区

如果选中“锁定鼠标到中央”复选框，则将影片剪辑实例的中心十字线锁定在鼠标光标上，此时 startDrag 语句就呈现为第二种表现形式。参数 Lock 取值为布尔值 true 或 false。

如果选中“限制为矩形”复选框，此时右侧的 4 个文本框被激活，输入数值以确定影片剪辑拖曳的有效区域。此时 startDrag 语句就呈现为第三种表现形式。

使用 startDrag 语句只能拖曳一个影片剪辑实例。但是，在某些动画中却可以看到鼠标光标后面跟随了大量对象，这是否会产生矛盾呢？其实不然，这主要是通过 setProperty 语句和 getProperty 函数的组合使用得到的。

使用 startDrag 语句拖曳影片剪辑后，如果不再拖曳影片剪辑就应该取消其拖曳状态。stopDrag 语句就是取消影片剪辑实例的拖曳状态，也就是说影片剪辑实例与光标之间不再存在关联，鼠标光标移动时影片剪辑实例并没有发生任何位置的变化。

5.4.2 范例：神龙摆尾

动画浏览时，将光标固定于某处，则只能观看到色彩变化的球体。如果移动光标时，可以发现球体后面隐藏了大量的小球体，好似“神龙摆尾”。有意识地控制光标移动的轨迹及速度，就会发现所有球体依次分布在光标经过的轨迹上，如图 5.34 所示。

动画预览时可以发现，小球体隐藏在大球体的后面，所有球体具有同时发生颜色变化的共性。由此就应该联想到利用 duplicateMovieClip 语句进行影片剪辑实例的循环复制，需要注意的是各个球体的深度设置。具体步骤如下：

（1）按快捷键 Ctrl+N 新建文档，调用“文档属性”对话框，调整尺寸为 400px×300px、背景颜色为黑色（#000000），帧频为 30fps。

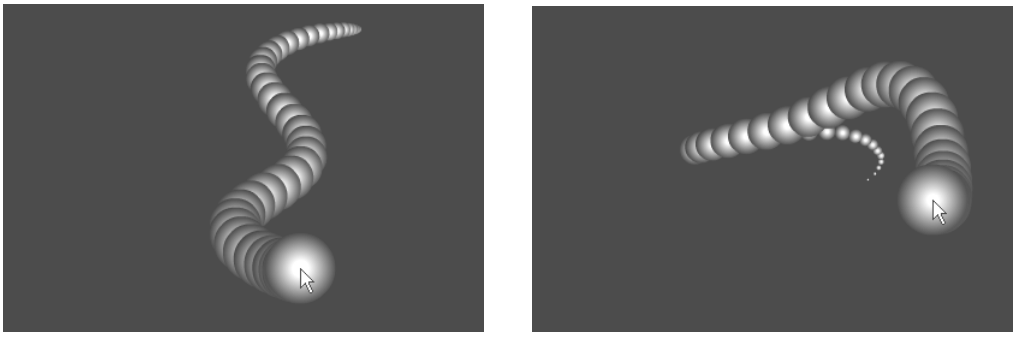

图 5.34　**效果预览**

（2**）创建影片剪辑** mcShape**，在舞台中央绘制放射状渐变的圆形填充，并在第** 12、24、36 **帧处插入关键帧，调整各关键帧中圆形填充的放射状渐变颜色，最后创建各关键帧之间的形变动画，如图** 5.35 **所示。**

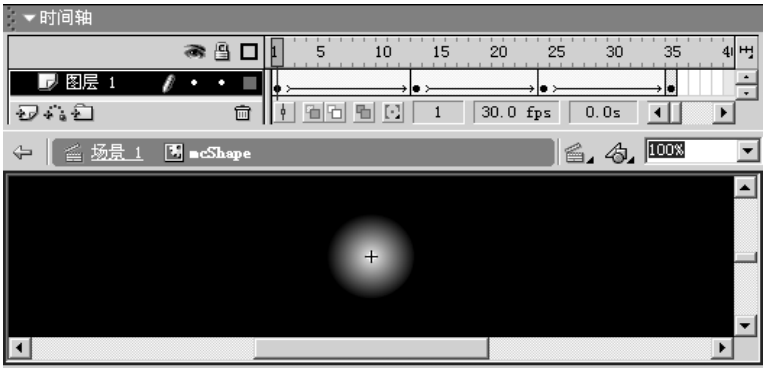

图 5.35　**影片剪辑** mcShape

（3**）切换到场景编辑状态，更改图层名称为“ 元件 ”，将“ 库 ”面板中的影片剪辑** mcShape **拖放到舞台上，如图** 5.36 **所示，并在“ 属性 ”面板中设置实例名称** shape0**。**

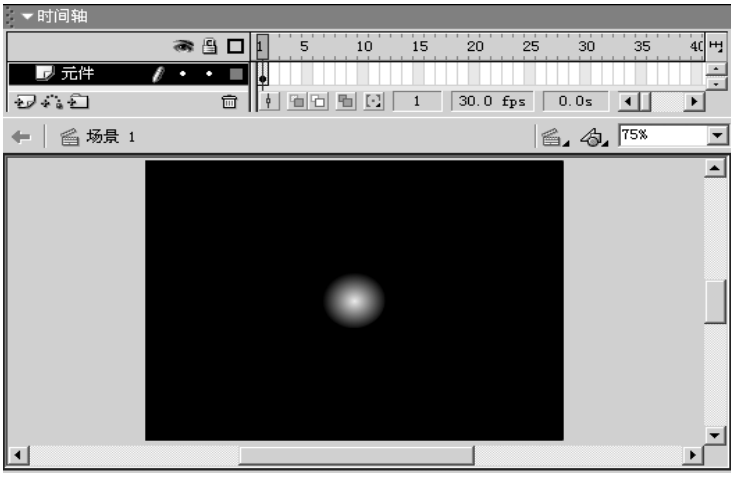

图 5.36　**舞台对象布置**

（4）添加图层“脚本”，在“动作”面板中设置脚本内容如下：

```
01  m = 3;
02  totalNum = 50;
03  scale = int(100/totalNum);
04  for (i=1; i<totalNum; i++) {
05    duplicateMovieClip("shape0", "shape" + i, -i);
06  }
07  startDrag("shape0", true);
08  setProperty("shape0", _visible, false);
```

提示：第 01～03 行：变量初始化；
第 04～06 行：通过 for 语句循环复制生成大量球体；
第 07 行：拖曳影片剪辑实例 shape0；
第 08 行：隐藏影片剪辑实例 shape0。

（5）选中舞台上的影片剪辑实例 shape0 后，在“动作”面板中设置影片剪辑实例的脚本内容如下：

```
01  onClipEvent (load) {
02    xpos = getProperty("", _x);
03    ypos = getProperty("", _y);
04    name = getProperty ("",_name);
05    i = Number (substring (name,6,2));
06    setProperty("", _xscale, 100-i*_root.scale);
07    setProperty("", _yscale, 100-i*_root.scale);
08  }
09  onClipEvent (enterFrame) {
10    if (i>0) {
11      xspace = getProperty ( "/shape"+(i-1), _x )-this.xpos;
12      yspace = getProperty ( "/shape"+(i-1), _y )-this.ypos;
13      xpos = xpos + xspace/_root.m;
14      ypos = ypos + yspace/_root.m;
15      setProperty("", _x, xpos);
16      setProperty("", _y, ypos);
17    }
18  }
```

提示：第 01～08 行：加载影片剪辑时，获取当前影片剪辑的位置并赋予变量 xpos、ypos；从当前影片剪辑的实例名称中截取数值赋予变量 i；最后根据变量 i 确定当前影片剪辑的缩放系数。
第 09～18 行：当影片剪辑处于播放帧中时，获取相邻两个球体之间的水平间距 xspace 和垂直间距 yspace；最后更新当前影片剪辑的位置（xpos, ypos）。

至此，神龙摆尾的动画就制作完成了，按快捷键 Ctrl+Enter 测试动画。移动鼠标光标观看一下动画界面中球体的运动方式吧！

5.4.3　范例：灯光闪烁

动画浏览时，可以看到黑暗中有灯光在闪烁，伴随着灯光显示局部文字。当在动画区域内按住鼠标左键进行拖动，可以发现灯光随着光标移动而且显示不同区域中的文字。一旦释放鼠标左键，则灯光就停留在释放处闪烁显示文字，如图 5.37 所示。

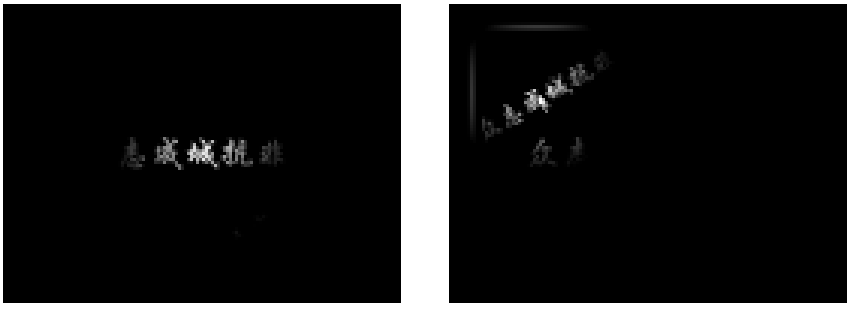

图 5.37　效果预览

在遮罩动画中使用 startDrag 语句拖曳影片剪辑实例时，既允许影片剪辑实例处于遮罩组中的被遮罩层中，也允许影片剪辑实例处于遮罩组中的遮罩层中。具体制作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，调用“ 文档属性 ”对话框，调整尺寸为 400px×300px、背景颜色为黑色（#000000）。创建影片剪辑 mcLight，在舞台中央绘制放射状渐变的圆形填充，分别在第 12、24 帧处插入关键帧，创建形变动画显示灯光闪烁的效果，如图 5.38 所示。

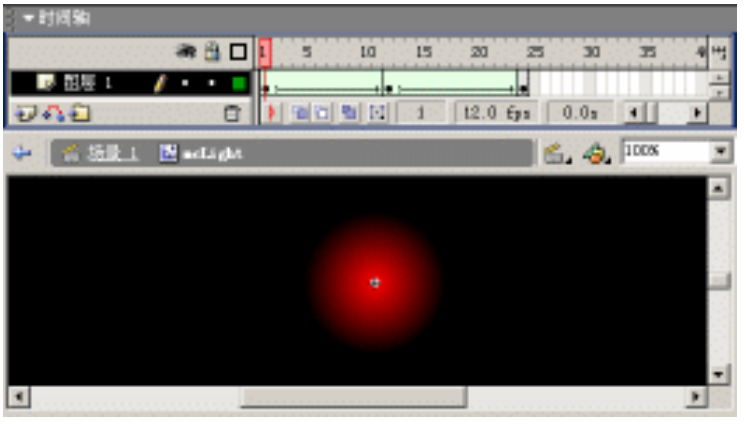

图 5.38　影片剪辑 mcLight

（2）切换到场景编辑状态，更改图层名称为“ 元件 ”，将“ 库 ”面板中的影片剪辑 mcLight 拖放到舞台上，并在“ 动作 ”面板中设置该影片剪辑实例的脚本内容如下：

```
01  onClipEvent (mouseDown) {
02    this.startDrag( true );
03    this.stop();
04  }
05  onClipEvent (mouseUp) {
```

```
06    this.stopDrag();
07    this.play();
08  }
```

提示：第 01～04 行：动画内按住鼠标左键时，将当前影片剪辑的中心锁定在光标上进行拖动，并使当前影片剪辑的播放指针停止播放；
第 05～08 行：动画内释放鼠标左键时，将当前影片剪辑脱离光标，固定在光标释放处继续播放，以显示灯光的闪烁。

按快捷键 Ctrl+Enter 测试动画，在动画区域中按住鼠标左键并拖动，观看影片剪辑的播放情况；释放鼠标左键再比较影片剪辑的播放情况。

（3）添加图层“遮罩”，在舞台上创建文字块，并绘制粗线条的矩形框。右击“遮罩”图层，选择弹出菜单中的“遮罩层”命令创建遮罩，如图 5.39 所示。

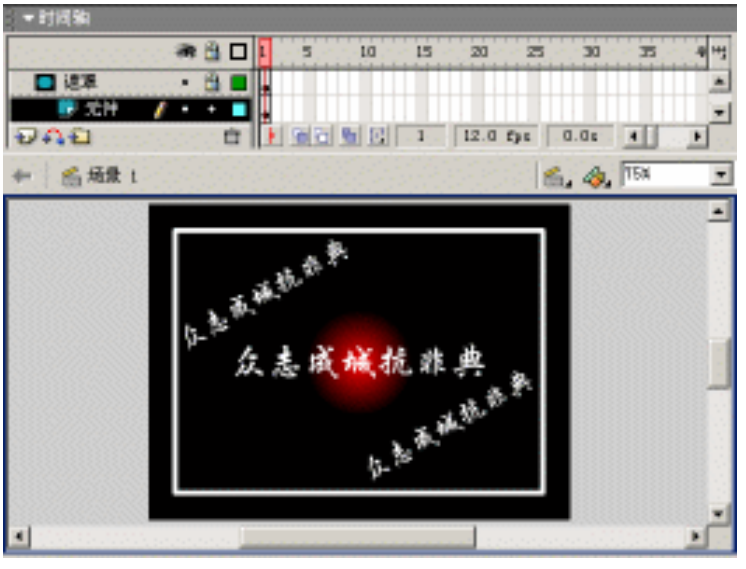

图 5.39 创建遮罩

注意：“遮罩”图层中的文本块及矩形框必须都要转换为填充。文本块可以选择“修改”|“分离”命令转换为填充；矩形线条可以选择“修改”|“形状”|“将线条转换为填充”命令。如果没有经过此步操作，将不能显示灯光闪烁的效果。

至此，整个动画就制作完成了，按快捷键 Ctrl+Enter 测试动画。

Flash 支持遮罩层中影片剪辑实例的拖曳，下面就对本例动画进行适当的修改。首先取消两个图层的遮罩关系，然后翻转两个图层的上下关系，最后再创建两个图层的遮罩关系，如图 5.40 所示。再次测试动画，比较一下动画效果。

5.4.4 认识坐标系

坐标系通常应用于分析各种数据，而 Flash 动画制作中也经常使用坐标系。Flash 中坐标系的 X 轴正向为水平向右方向，Y 轴正向为垂直向下方向，如图 5.41 所示。而其原点 O

（0, 0）随编辑环境的变化进行调整。

场景编辑状态中的原点 O 就是舞台的左上角，而元件编辑状态中的原点 O 就是舞台中的十字线。因此制作动画时清楚原点 O 的位置是相当重要的。

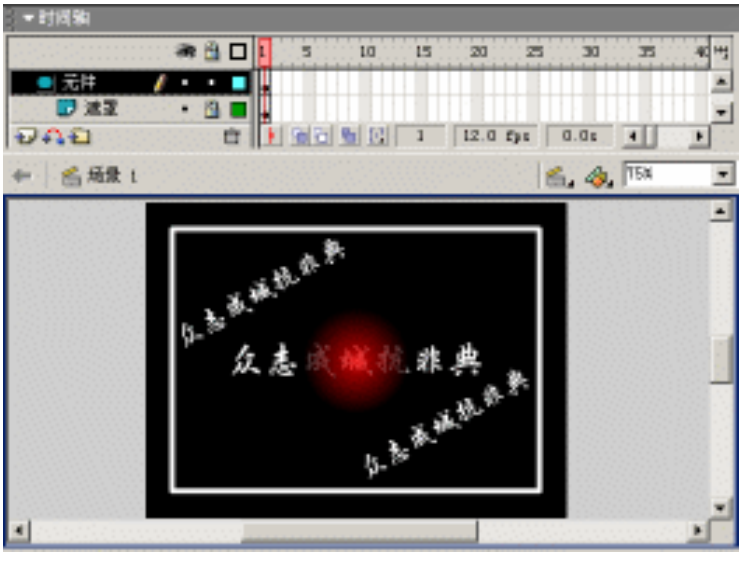

图 5.40　修改后的动画

技巧：如果你不清楚坐标系的正向及原点 O，可以在“信息”面板中查看光标移动时该面板中数值的动态变化，如图 5.42 所示。利用这种方法可以快速、轻松地掌握 Flash 中的坐标系。

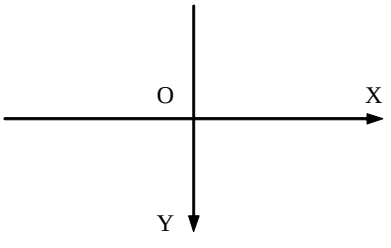

图 5.41　认识坐标系

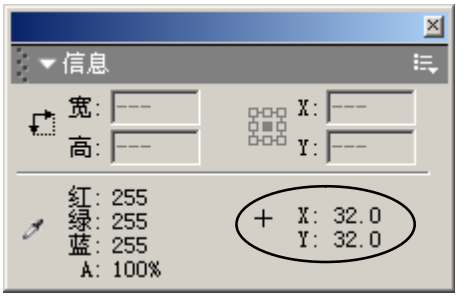

图 5.42　查看光标位置

5.4.5　fscommand

使用 Flash 动画时，通常并不是将它作为独立的动画作品进行使用，而是将其置入到 Web 页面、PowerPoint 课件中，甚至是作为 VB、VC 等编程工具的素材。

基于装载 Flash 动画作品与应用程序交换信息的需要，Flash 提供的语句 fscommand 可以大大体现 Flash 动画的多应用性。同时语句 fscommand 也控制着动画播放器播放 Flash 动画时的窗口外观。语句 fscommand 具体使用的语法如下：

```
fscommand ( Command, Arguments );
```

（6）锁定“枯树”图层，以防止更改其中的图形。添加图层“背景”，并将之移动到“枯树”图层的下方；绘制覆盖整个舞台的深灰色（#333333）矩形填充，选中该矩形后按 F8 键转换为图形元件“背景色块”，如图 6.19 所示。

图 6.19 “转换为元件”对话框

（7）在“背景”图层的第 4、7、10、13、16、19 帧处按 F6 键插入关键帧，并调整第 4、10、16 帧中的实例色调为 100%浅灰色（#CCCCCC），如图 6.20 所示，使整个背景呈现闪电所带来的忽明忽暗的效果。

图 6.20 设置背景色块的实例颜色

至此，整个动画就制作完成了，舞台布置及时间轴的分布如图 6.21 所示。按快捷键 Ctrl+Enter 测试动画，聆听雷电交加的声音，观看忽明忽暗的画面。

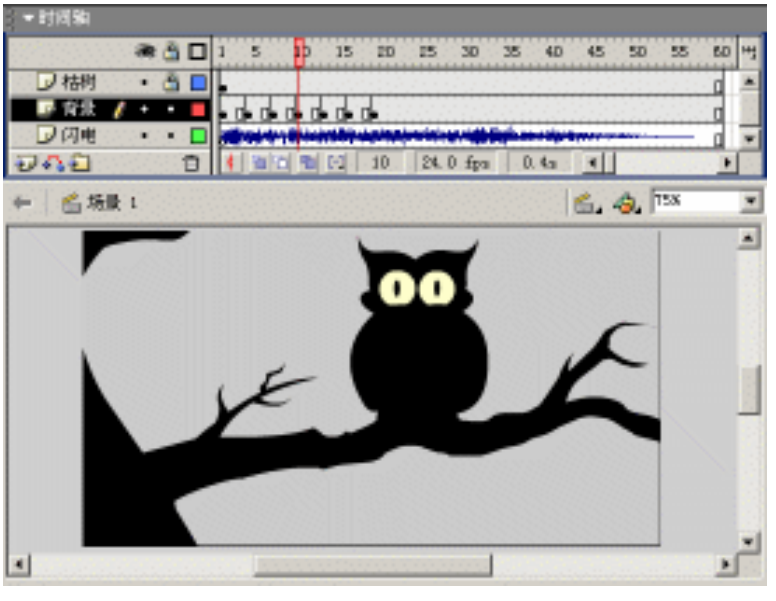

图 6.21 最终时间轴显示

通过本例的制作，基本掌握了 Flash 动画中声音的导入和使用，深刻体会声音和画面

相互结合、协调的重要性。假设将本例动画中的声音删除，只能观看到画面背景的忽明忽暗，怎么可能感受这是雷电交加的夜晚呢？

6.3　范例：大话西游

在动画中置入声音，与动画协调整合，动画的整体效果将得到较大幅度的提升。如果只通过时间轴添加声音，那么可以肯定的是您低估了 Flash 对声音的处理能力。现在，就来演示一下利用脚本实现对声音各种效果的控制。

首先最好带上耳机试听一下《大话西游》中的插曲，享受音乐的同时拖动画面中“音量”、“声道”的滑块，如图 6.22 所示，可以发现歌曲的音量大小和声道均衡都发生着相应变化。单击“停止”按钮和“重播”按钮可以使该插曲停止播放或重新开始播放。

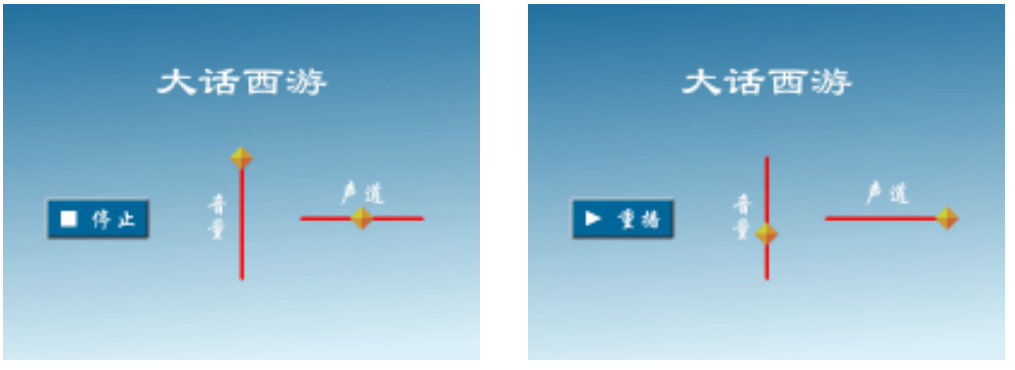

图 6.22　效果预览

具体制作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，选择“文件”|“导入”命令导入声音文件“大话西游.mp3”，按 F11 键调用“库”面板，可以发现预览窗中显示了两条波形折线，如图 6.23 所示，表示该声音是双声道的。

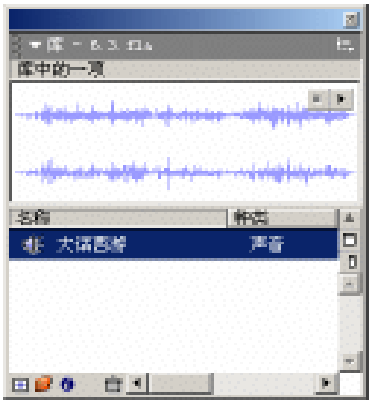

图 6.23　“库”面板

（2）右击“ 库 ”面板中的声音素材，选择弹出菜单中的“ 链接 ”命令调用“ 链接属性 ”对话框，选中“ 链接 ”栏中的“ 为动作脚本导出 ”复选框，并在激活的“ 标识符 ”文本框中输入 dhxy，如图 6.24 所示，单击“ 确定 ”按钮确认。

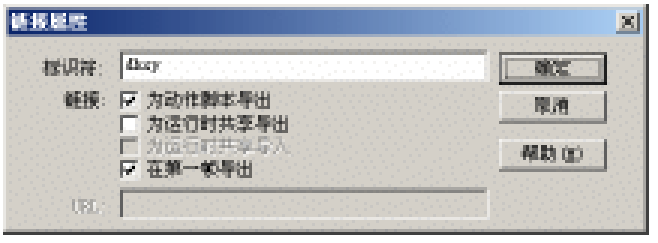

图 6.24 “ 链接属性 ”对话框

（3）按快捷键 Ctrl+J 调用“ 文档属性 ”对话框，调整尺寸为 400px×300px。更改图层名称为“ 背景 ”，绘制覆盖整个舞台的线性填充矩形，如图 6.25 所示；添加图层“ 脚本 ”，在该图层中设置帧脚本内容如下：

```
snd = new Sound();
snd.attachSound("dhxy");
snd.start();
```

提示：利用构造函数 new 创建 Sound 对象 snd，并将其链接到“ 库 ”面板中标识符为 dhxy 的声音对象，最后使用方法 start 播放 Sound 对象 snd。

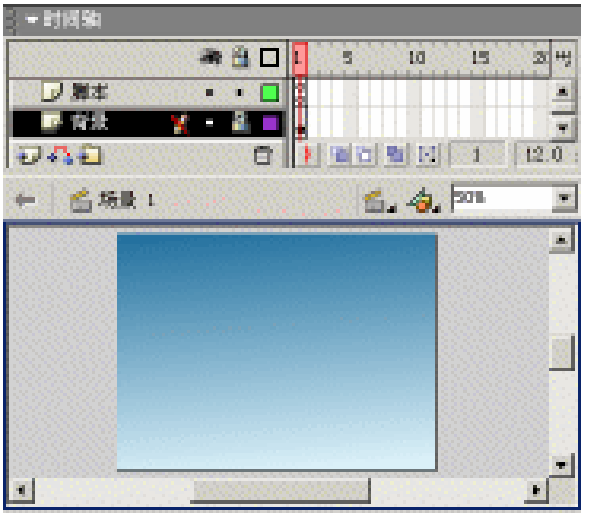

图 6.25 舞台最初布置

（4）按快捷键 Ctrl+F8 创建影片剪辑 mcScroll，在舞台上绘制不同填充颜色构成的菱形，如图 6.26 所示，并将该菱形填充转换为按钮元件 btnScroll，再设置该按钮实例的脚本内容如下：

```
01  on (press) {
```

```
02    startDrag("", false, left, top, right, bottom);
03    dragging = true;
04  }
05  on (release, releaseOutside) {
06    stopDrag();
07    dragging = false;
08  }
```

提示：第 01～04 **行：当按住（**press**）按钮时，使得当前影片剪辑在（**left, top**）和（**right, bottom**）构成的矩形区域中移动，并设置变量** dragging **为布尔值** true **；第** 05～08 **行：当释放（**release**）或外部释放（**releaseOutside**）按钮时，使得当前影片剪辑固定在某位置处，并设置变量** dragging **为布尔值** false**。**

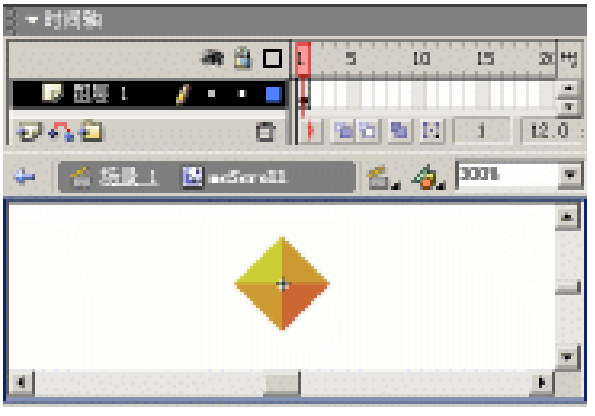

图 6.26　**影片剪辑** mcScroll

（5**）按快捷键** Ctrl+F8 **创建影片剪辑** mcControl**，在舞台上绘制填充矩形并将之转换为按钮元件** btnControl**，在该按钮实例上创建白色文本“ 停止 ”、绘制白色填充方形；单击关键帧后在“ 属性 ”面板中设置帧标签为** stop**，此时的时间轴显示和舞台布置如图** 6.27 **所示。**

（6**）在第** 10 **帧处按** F6 **键插入关键帧，修改文本内容为重播，修改图形为白色三角，并设置帧标签为** play**，如图** 6.28 **所示。选中该关键帧中的按钮实例，在“ 动作 ”面板中设置按钮实例的脚本内容如下：**

```
on (release) {
  _root.snd.start();
  gotoAndStop("stop");
}
```

提示：当单击（release**）按钮时，控制主时间轴中的** Sound **对象** snd **开始播放，并使得播放头停止在帧标签为** stop **的关键帧处，从而实现“ 停止 ”按钮和“ 重播 ”按钮的交替出现。**

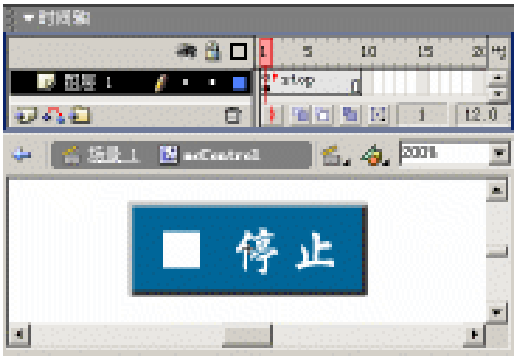

图 6.27 影片剪辑 mcControl——帧 stop

图 6.28 影片剪辑 mcControl——帧 play

（7）切换到第 1 帧处，选中其中的按钮实例并设置脚本内容如下：

```
on (release) {
  _root.snd.stop();
  gotoAndStop("play");
}
```

提示：当单击（release）按钮时，使主时间轴中的 Sound 对象 snd 停止播放，并使播放头停止在帧标签为 play 的关键帧处，从而实现“停止”按钮和“重播”按钮的交替出现。

（8）为了防止影片剪辑 mcControl 的自动播放，在其中的 2 个关键帧处均设置帧脚本语句“stop();”。

（9）按快捷键 Ctrl+E 切换到场景编辑状态，添加图层“元件”，将“库”面板中的影片剪辑 mcControl、mcScroll 拖放到舞台上。在舞台中央绘制长度为 100px 的垂直直线，将影片剪辑 mcScroll 实例拖放到垂直直线的顶端；再在舞台右侧绘制长度为 100px 的水平直线，复制影片剪辑 mcScroll 实例并将其置于水平直线的中点。

（10）添加图层“文字”，在舞台上创建动画名称的文本“大话西游”，并在竖线和横线的位置处分别创建说明文本“音量”和“声道”，如图 6.29 所示。

（11）选中控制音量的影片剪辑 mcScroll 实例，在“动作”面板中设置脚本内容如下：

```
01  onClipEvent (load) {
02    top = this._y;
03    left = this._x;
04    right = this._x;
05    bottom = this._y+100;
06  }
07  onClipEvent (enterFrame) {
08    if (dragging == true) {
09      _root.snd.setVolume(100-(this._y-top));
10    }
11  }
```

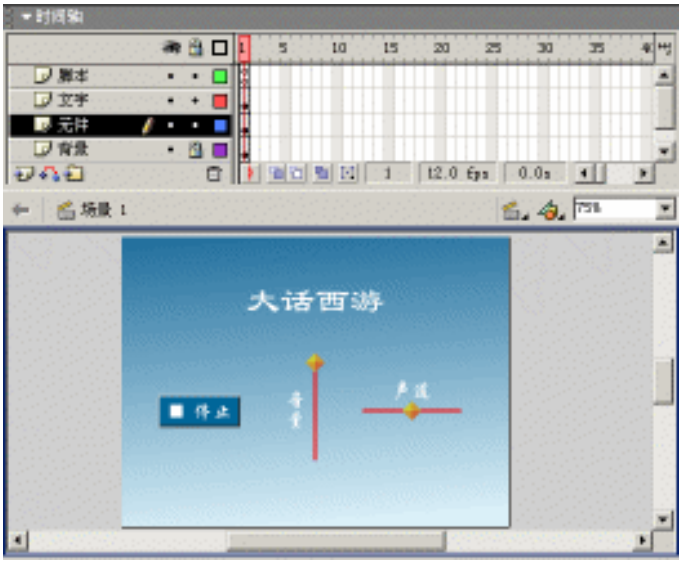

图 6.29　最终舞台布置

提示：第 01～06 行：载入（load）影片剪辑 mcScroll 实例时，初始化确定移动范围的 4 个变量；
第 07～11 行：影片剪辑 mcScroll 实例处于播放帧（enterFrame）中时，根据变量 dragging 确定是否对 Sound 对象 snd 进行音量调整（setVolume）。

（12）选中控制声道的影片剪辑 mcScroll 实例，在“动作”面板中设置脚本内容如下：

```
01  onClipEvent (load) {
02    top = this._y;
03    bottom = this._y;
04    left = this._x-50;
05    right = this._x+50;
06    center = this._x;
07  }
08  onClipEvent (enterFrame) {
09    if (dragging == true) {
10      _root.snd.setPan((this._x-center)*2);
11    }
12  }
```

提示：第 01～07 行：载入（load）影片剪辑 mcScroll 实例时，初始化确定移动范围的 4 个变量和存储实例最初位置的变量 center；
第 08～12 行：影片剪辑 mcScroll 实例处于播放帧（enterFrame）中时，根据变量 dragging 确定是否对 Sound 对象 snd 进行声道均衡的调整（setPan）。

至此，利用脚本调用声音并控制声音音量、声道均衡的动画就制作完成了，按快捷键 Ctrl+Enter 测试动画，好好享受一下美妙的音乐吧！

6.4 视 频 对 象

Flash 对视频对象的支持使动画功能的扩展成为可能,将视频文件导入到 Flash 动画中，既可以展示 Flash 动画的绚丽，又可以体验视频影像的精彩。

6.4.1 使用视频对象

要使用视频对象，可按如下步骤进行操作：

（1）按快捷键 Ctrl+N 新建文档，选择“文件”|“导入”命令，在“导入”对话框中选择视频文件“人生短暂.wmv”，单击“打开”按钮后弹出“导入视频设置”对话框，如图 6.30 所示。

图 6.30 “导入视频设置”对话框

该对话框中的参数说明如下：

- “品质” 控制导入视频所显示的画面质量，可以对视频剪辑进行压缩。当品质设置较低时，导入的视频就相对较小，但是却破坏了整个视频的画面质量，呈现马赛克；当品质设置较高时，导入的视频画面质量较好，但是却使得整个视频变得非常庞大。因此，品质设置可以进行多次尝试，以求得到画面质量和视频容量最佳配合的导入方案。通常品质可以设置为 70%～80% 。
- “关键帧间隔” 控制视频中的关键帧（具有完整数据的帧）的帧频。假设，关键帧间隔为 30，则意味着剪辑中每 30 帧存储一个完整的帧。在间隔之间的帧只存储与前一帧之间发生变化的数据。关键帧间隔越小，所存储的完整帧就越多。这样就能够在视频中进行更快的搜索，但是产生的文件也会更大。
- “缩放” 控制视频的画布尺寸。较小的缩放比例使得 Flash 动画减小的同时也改

善了视频的播放效果。假设数字视频的画面尺寸为 640px×480px，将该参数缩小到原有的 25%，将会提高 Flash 动画中视频的播放性能。

- “将视频同步到 Macromedia Flash 文档帧频”复选框　选中该复选框，将使导入视频的播放速度与 Flash 动画的播放速度同步。取消该选项将禁止帧频同步。
- “对于一定数量的 Macromedia Flash 帧要编码的视频帧数量”列表框　选择其中的比例可以指定导入视频帧与 Flash 动画帧之间的比率。例如，要对于每一个 Flash 动画帧播放一个导入的视频帧，则选择“1:1”；对于每两个 Flash 动画帧播放一个导入的视频帧，则选择“1:2”；依此类推。
- “导入音频”复选框　选中该复选框可以将音轨（如果有音频存在）包含在导入的视频中。

（2）单击“确定”按钮后，弹出“正在导入”对话框，显示视频的导入进度。经过一定时间后，弹出如图 6.31 所示的对话框，告知完整播放该视频所需的帧数，并征询您是否自动插入帧。

图 6.31　系统提示对话框

（3）单击“是”按钮，在时间轴中就会自动插入视频所需的帧，按 Enter 键在场景下进行播放即可预览视频的显示画面；按 F11 键调用“库”面板，可以发现导入视频的踪影，如图 6.32 所示。

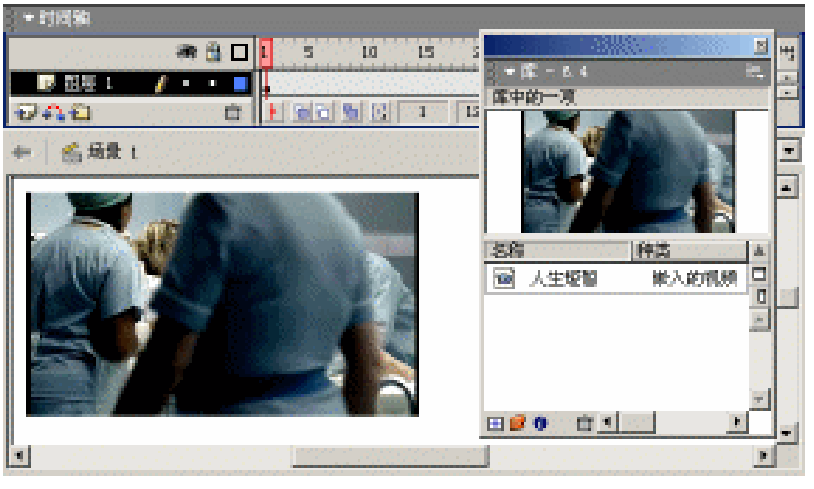

图 6.32　插入视频对象

（4）按快捷键 Ctrl+Enter 测试动画，此时就可以欣赏到声像完美结合的视频了。

6.4.2 视频属性设置

利用“嵌入视频属性”可以更新已经导入的视频对象，或者将导入的视频对象替换为其他外部视频。

右击“库”面板中的视频素材，选择弹出菜单中的“属性”命令，调用“嵌入视频属性”对话框，如图 6.33 所示。从该对话框中可以获知视频导入时的存储路径以及视频尺寸和容量等信息。

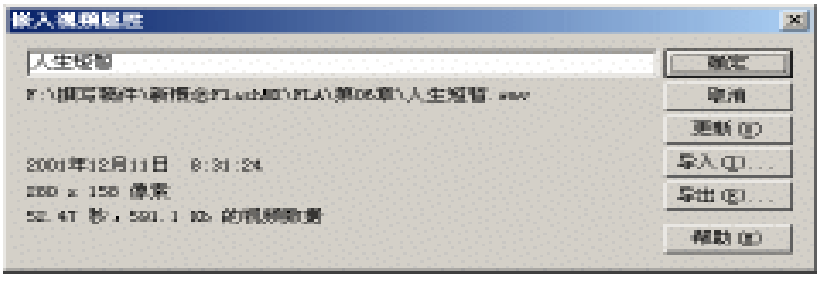

图 6.33 “嵌入视频属性”对话框

单击“导入”按钮弹出“导入”对话框，仍选择视频文件“人生短暂.wmv”，单击“打开”按钮就会弹出“导入视频设置”对话框，从而可以更改视频导入时的各个选项。

在此可比较一下不同品质导入视频的画面质量。图 6.34 是品质为 100%导入的视频画面，整体画面变化比较平滑；而图 6.35 是品质为 30%导入的视频画面，可以很明显地看到图像中的马赛克迹象。

由此可见，品质的设置直接影响着导入视频的画面质量。因此，综合考虑导入视频的大小和画面质量等要素，通常在导入时将品质设置为 70%～80%，可以获得较好的画面质量，同时文件也较小。

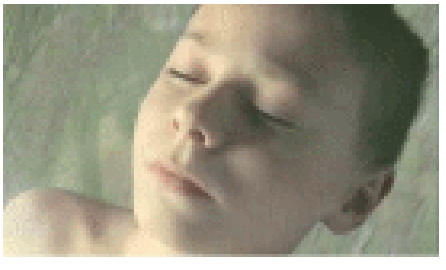

图 6.34 品质为 100%的画面

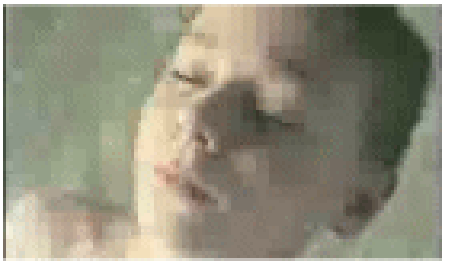

图 6.35 品质为 30%的画面

6.5 范例：视频遥控

相信您一定观看过视频墙，多个屏幕同时播放相同的内容，这是何等的气势！随着视

频动画的播放，右下角的数值不断发生着变化，告知您当前所播放的帧数；而另一个数值就是视频完全播放的总帧数，如图 6.36 所示。单击画面右侧纵向排列的按钮，可以对动画中的视频加以控制，尝试一下吧！

图 6.36　效果预览

6.5.1　界面布置

（1）按快捷键 Ctrl+N 新建文档，选择“修改”|“文档”命令调用“文档属性”对话框，调整尺寸为 500px×300px，更改图层名称为“背景”，绘制覆盖整个舞台的线性渐变填充矩形。

（2）添加图层“视频”，按快捷键 Ctrl+R 导入视频文件“番茄酱.mpeg”，弹出“导入视频设置”对话框，设置品质为 90%、缩放为 60%，其他参数如图 6.37 所示。

图 6.37　“导入视频设置”对话框

（3）单击“确定”按钮确认即可导入视频，在弹出的对话框中确认自动插入帧以完全显示整个视频。复制舞台上的视频实例得到 9 个相同的视频实例，将这些实例按 3×3 的方式排列在舞台上，形成视频墙。

（4）添加图层“元件”，选择“窗口”|“公共库”|“按钮”命令，调用“库”面板，如图 6.38 所示，打开其中的 Playback 文件夹，将其中一系列方形按钮拖放到舞台上，纵向排列在舞台的右侧区域。

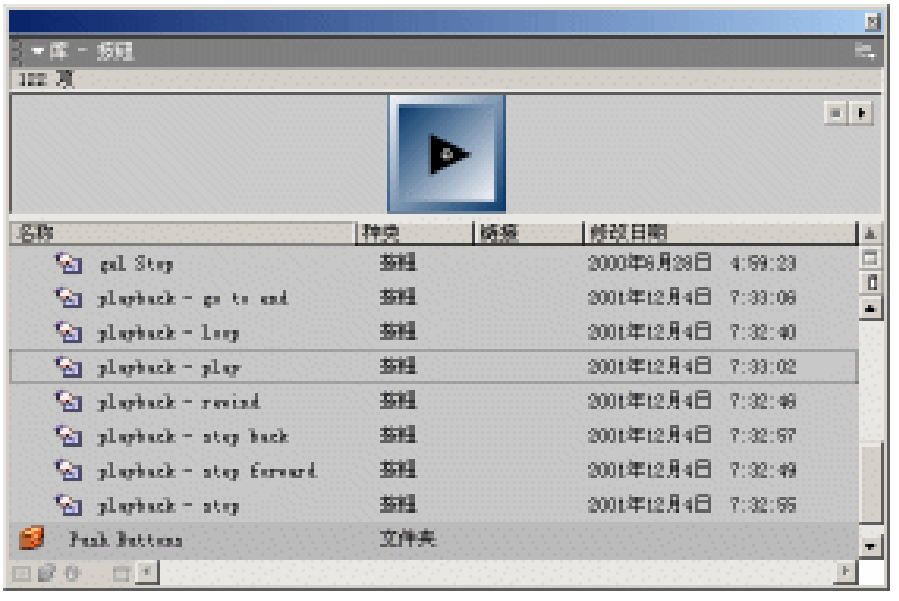

图 6.38 “库”面板——按钮

（5）利用文本工具 A 在舞台的右下角创建具有一定宽度的动态文本，在“属性”面板中设置动态文本的格式，尤其重要的是在“变量”文本框中输入 curFrame，如图 6.39 所示。关于文本的创建和使用等具体内容可以参考第 8 章。

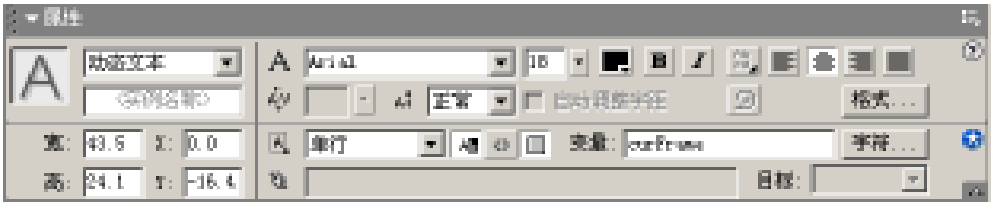

图 6.39 动态文本的“属性”面板

（6）在该动态文本的左侧创建文本“第”，右侧创建文本“帧”；复制动态文本置于下方，并在“属性”面板中重新定义“变量”为 totFrame，随后在其两侧创建文本“共”、“帧”，如图 6.40 所示。

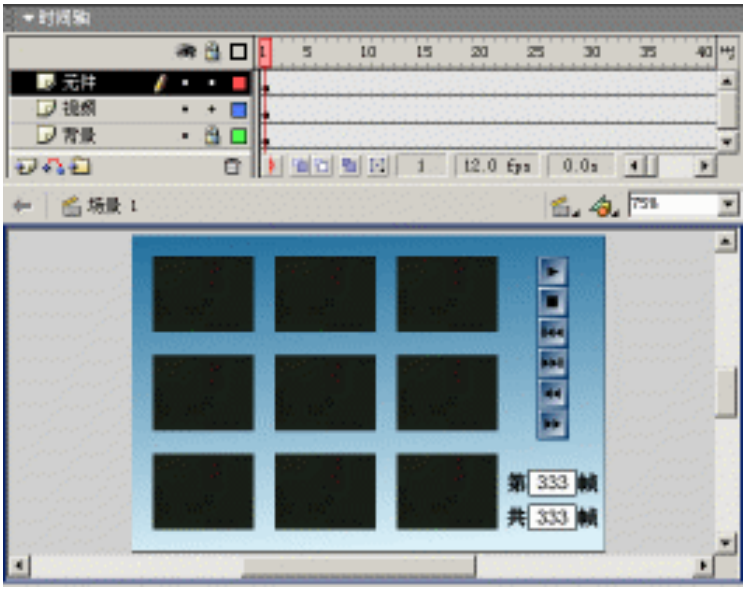

图 6.40　舞台布置

6.5.2　脚本编写

（7）选中步骤（5）、（6）中的所有文本对象，按 F8 键将其转换为影片剪辑 mcFrame。选择“窗口”|“动作”命令调用“动作”面板，设置该影片剪辑实例的脚本内容如下：

```
01  onClipEvent (load) {
02    curFrame = _root._currentframe;
03    totFrame = _root._totalframes;
04  }
05  onClipEvent (enterFrame) {
06    curFrame = _root._currentframe;
07  }
```

提示：第 01～04 行：载入（load）该影片剪辑实例时，初始化变量 curFrame、totFrame，分别用于存储主时间轴播放头所在的帧数（_currentframe）和主时间轴的总帧数（_totalframes）；
第 05～07 行：当该影片剪辑实例处于播放帧（enterFrame）时，更新变量 curFrame 的值。

（8）单击“动作”面板顶端的列表框按钮，如图 6.41 所示，由下拉列表中选择“动作[未指定实例名称]（playback - play）”项，即可切换到播放按钮的脚本编辑，具体脚本内容如下：

```
on (release) {
  play();
}
```

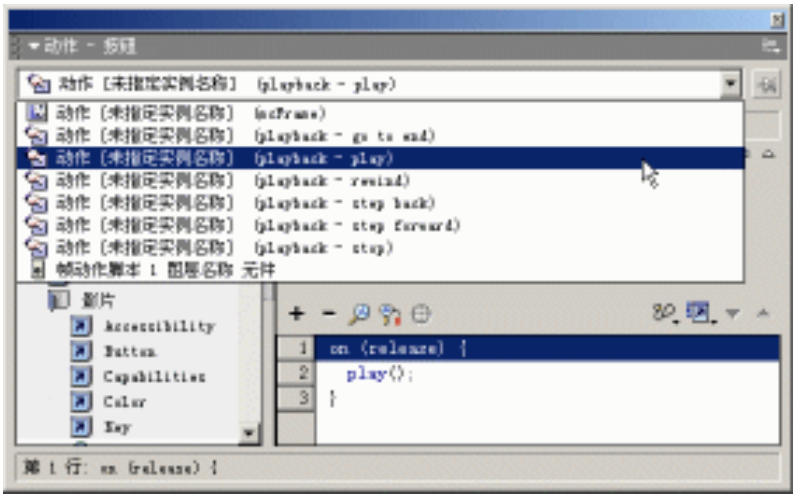

图 6.41 “动作”面板——定位到其他脚本

（9）采用相同的方法，设置其他按钮实例的脚本内容如下：

按钮元件 playback – stop 实例的脚本内容：

```
on (release) {
  stop();
}
```

按钮元件 playback - rewind 实例的脚本内容：

```
on (release) {
  gotoAndStop(1);
}
```

按钮元件 playback - go to end 实例的脚本内容：

```
on (release) {
  gotoAndStop(_totalframes);
}
```

按钮元件 playback - step back 实例的脚本内容：

```
on (release) {
  gotoAndStop(_currentframe - 10);
}
```

按钮元件 playback - step forward 实例的脚本内容：

```
on (release) {
  gotoAndStop(_currentframe + 10);
}
```

至此，视频墙的动画制作就全部完成了，按快捷键 Ctrl+Enter 测试动画，观赏一下这个小型化的视频墙，效果不错吧！可惜的是生成动画需要耐心地等待一番。

6.6　小结和练习

本章详细介绍了声音对象的导入、使用和润色等功能，并通过范例“午夜惊闪”的制作进一步强调声音应用在动画制作中的重要性；再通过范例“大话西游”介绍了脚本对声音的调用和控制。

同时也清楚地讲述了视频对象的导入和使用，并通过范例“视频遥控”的制作介绍了 Flash 动画中对视频播放的控制，同时展示了小型视频墙。

通过本章的学习，希望能够完成以下练习：

1．选择一首自己喜欢的 MP3，利用各种动画制作方法和技巧为该 MP3 添加上动画画面，制作生成 Flash MV。

提示：在 Flash MV 的动画制作中，应该将导入的 MP3 声音设置为数据流声音。

2．根据范例“大话西游”实践一下利用脚本控制声音的方法，将其中控制音量和声道的滚动条适当进行美化！

3．改编范例“视频遥控”，要求将动画中以数字方式显示视频的播放情况改成以进度条的方式显示视频的播放。

提示：制作两个尺寸相同、颜色不同的矩形条，将其中一个矩形条转换为影片剪辑 Bar，并定义实例名称为 Bar，将该实例覆盖在另一个矩形条上，设置该影片剪辑实例的脚本内容如下即可！

```
onClipEvent (enterFrame) {
  this._xscale = 100*_root._currentframe/_root._totalframes ;
}
```

第 7 章　图像动画篇

虽然 Flash 动画是基于矢量图形的，但是位图具有颜色丰富、形象逼真等众多特点，这些都是矢量图形所无法比拟的。因此，在 Flash 动画的设计和制作过程中，适当使用位图图像，可以使动画效果增色不少！

通过本章的学习，应该达到如下目的：

- 掌握图像的导入及处理
- 了解动画中的光标更换
- 掌握画面切换的制作方法
- 学会图像的无缝切割技术

7.1　图 像 应 用

7.1.1　导入图像对象

由于位图图像的种种优势，在 Flash 动画制作中经常需要导入一些位图图像，作为整个动画的靓丽背景或穿插出现在动画中。因此在 Flash 中支持大多数图像格式，包括 JPG 格式、GIF 格式、TIFF 格式、PNG 格式等等。

导入图像的步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+R 调用“导入”对话框，选择要导入图像文件的所在文件夹，如图 7.1 所示，双击要导入的图像文件 F001.JPG 即可导入该位图图像。

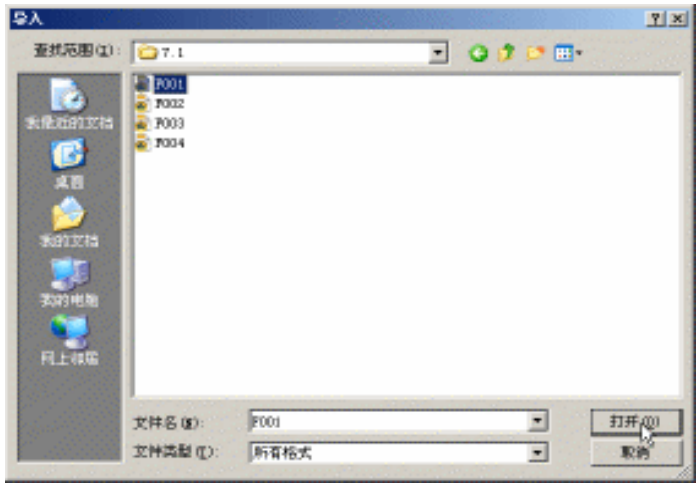

图 7.1　“导入”对话框

提示：Flash 支持多个文件的同时导入，如果需要一次性导入多张图像，可以按住 Ctrl 键后逐个单击需要导入的图像文件，最后再单击该对话框中的“打开”按钮即可。

（2）图像导入后返回到动画编辑状态，此时可以发现在舞台上就放置着该图像对象，且呈选中状态（注意图像选中时边界为灰色）；按 F11 键调用“库”面板，可以发现在“库”面板中也放置着该图像，如图 7.2 所示。

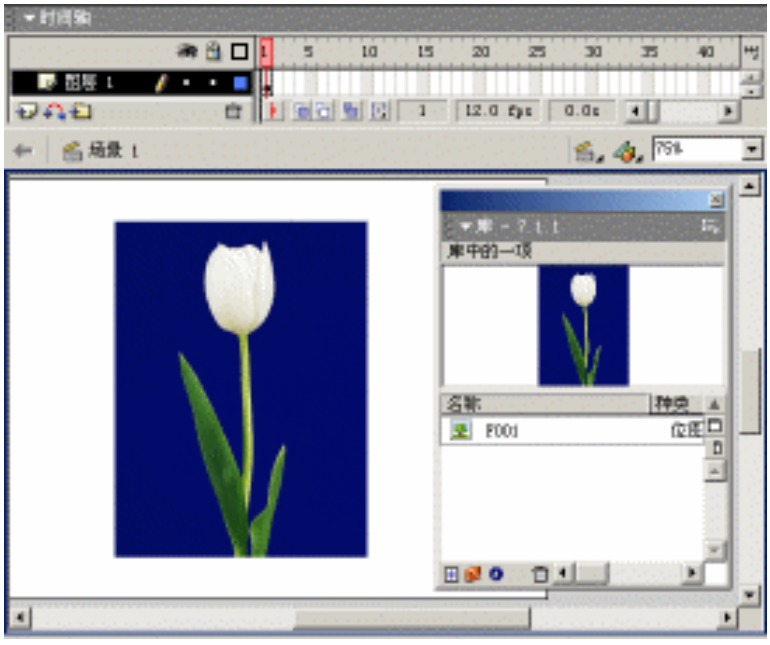

图 7.2　成功导入图像

为了使 Falsh 文件容量尽可能地减小，凡是导入的图像都被放置在“库”面板中，而且在动画制作中可以方便地重复使用这些图像。

当图像导入到 Flash 文件中，“库”面板中的图像才是实实在在的图像对象，记录着该图像的全部像素颜色；而舞台上放置的图像只是“库”面板中图像的实例。因此，在舞台上放置多个相同的图像时，文件容量并不会有太大的增加。如果将“库”面板中的图像对象删除，那么舞台上也就不会再显示相应的图像。

在实际使用过程中，由于这样的讲解比较繁琐，通常将存储在“库”面板中的图像对象和应用在舞台上的图像实例进行概念模糊，统称为图像对象。

提示：导入图像时，如果文件夹中有多个序列命名的图像文件（如 F01、F02、F03…），而只导入其中的某幅图像，此时将会弹出 Flash MX 对话框，如图 7.3 所示，询问用户是否导入序列中的所有图像。如果选择“是”，则将这些序列命名的图像全部导入并依次置于连续的关键帧中；选择“否”就只导入指定的图像文件。

图 7.3 Flash MX 对话框

提示：位图图像的导入也可以使用复制－粘贴的方法。首先在 PhotoShop、Fireworks 等图像处理软件中复制图像的全部或部分，然后切换到 Flash 中选择“编辑”|“粘贴”命令将存放在剪切板中的位图图像粘贴到舞台上，当然“库”面板中也同样显示相应的图像对象。

7.1.2 使用图像对象

位图图像导入到 Flash 文档中，就图像的使用方式可以概括为 3 种，即直接应用、分离处理和矢量化。图像对象的直接应用，就是导入图像后直接将“库”面板中的图像对象拖放到舞台上，通常会适当进行缩放、旋转、倾斜等变形处理。而图像的分离处理和矢量化就非常有必要进行一番探讨和学习。

分离处理

如果图像导入后只想使用图像中的部分内容，可以用第 3 章所学到的遮罩技术将所需部分的图像予以抠出，但是如果导入的图像存在大面积的纯色背景，就没有必要采用遮罩技术绘制抠图图形那么费时费力，此时只要将图像进行分离处理即可轻松得到抠图的目的。

分离处理图像的步骤如下：

（1）图像导入后，利用箭头工具选中舞台上的图像，选择“修改”|“分离”命令进行分离处理，此时的图像呈点状显示，如图 7.4 所示。

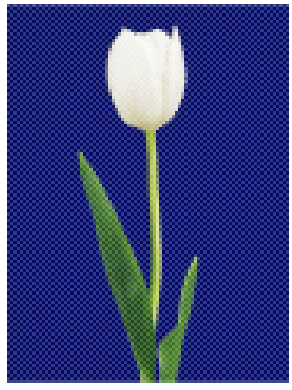

图 7.4 分离处理的图像

注意：严格来说，此时的图像已不是图像而是形状，只不过这类形状比较特殊，具有图像填充而已。导入图像后绘制任意形状，选择颜料桶工具，并在“混色器”面板中设置填充样式为“位图”，即可得到具有图像填充的形状。

（2）按 L 键切换到套索工具，选中工具选项区中的魔术棒工具，再单击其右侧的魔术棒属性按钮，弹出“魔术棒设置”对话框，如图 7.5 所示，适当进行参数设置后单击“确定”按钮确认。

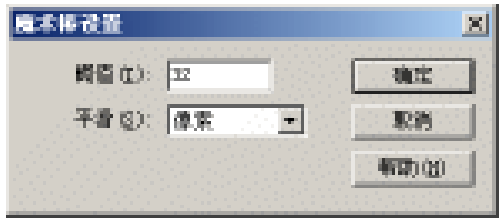

图 7.5　设置魔术棒属性

（3）此时光标显示为魔术棒图形，在图像中大面积的深蓝色背景上单击，选中全部深蓝色背景，如图 7.6 所示；按 Delete 键删除选定的深蓝色背景，此时舞台上就只剩下花卉及其枝叶部分，如图 7.7 所示。

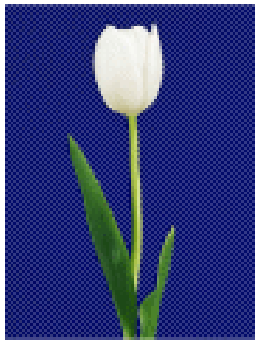

图 7.6　使用魔术棒选取

图 7.7　抠取的图像

提示：经过分离处理后的图像，就可以使用填充变形工具。单击该图像后，在其周围可显示伸缩句柄、旋转句柄、扭曲句柄以及位置句柄，将光标定位到各句柄上，按住鼠标左键进行拖动，可以获得重复填充的图形，如图 7.8 所示。

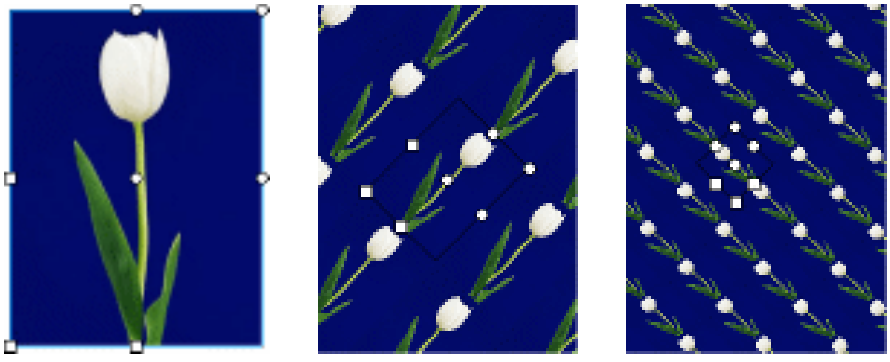

图 7.8　填充调整工具

矢量化

虽然位图图像经过分离处理后变为形状，具有矢量特征，但是该形状必须依靠“库”面板中的图像对象才能显示图像外观。当删除“库”面板中的图像对象时，形状的图像外观也就不复存在而恢复形状纯色填充的本来面貌。如果对具有图像填充的形状进行较大倍数的放大，将会露出位图放大后点阵分布色值的弊端。

鉴于上述问题，Flash 同时提供了对位图图像处理的另外一种方法，那就是位图图像的矢量化。这样，既可以摆脱“库”面板中的图像对象而独立存在，又可以实现矢量图形的无限级放大，并且也可以实现抠图功能。具体操作步骤如下：

（1）导入图像后，选中舞台上的图像对象，选择“修改”|“转换位图为矢量图”命令，弹出“转换位图为矢量图”对话框，如图 7.9 所示。

图 7.9 “转换位图为矢量图”对话框

在该对话框中有 4 个选项，各参数所代表的意义分别如下：

- 颜色阈值　确定图像中像素色值的近似化的程度，其取值范围为 1～500。将两个像素进行比较，如果它们在 RGB 颜色值上的差异低于该颜色阈值，则认为这两个像素的颜色相同。增大颜色阈值，就意味着减少颜色的数量。
- 最小区域　控制图像矢量化后的精细程度，其取值范围为 1～1000。该参数的设置对于最终得到的矢量图形所模拟源图像的效果具有很大的影响。假设将该参数设置为 30 像素，系统将会以最小为 30 像素的区域对源图像进行区域分割，如 1×30、2×15、3×10 等。
- 曲线拟合　确定图像矢量化的轮廓平滑程度，提供了 6 个级别的选项，分别为像素、非常紧密、紧密、正常、平滑以及非常平滑。
- 角阈值　确定图像矢量化时是保留锐边还是进行平滑处理，提供了 3 个选项，分别为较多转角、正常和较少转角。

（2）参数设置完毕后，单击“确定”按钮，此时将会弹出转换进度框。在舞台空白处单击，取消形状的选定状态，如图 7.10 所示。利用箭头工具 单击深蓝色部分，按 Delete 键删除选定部分，如图 7.11 所示。

通过矢量化转换得到的形状就完全脱离了“库”面板中相应图像对象的控制，此时删除“库”面板中的图像对象，也不会影响舞台上花卉及其枝叶的显示。

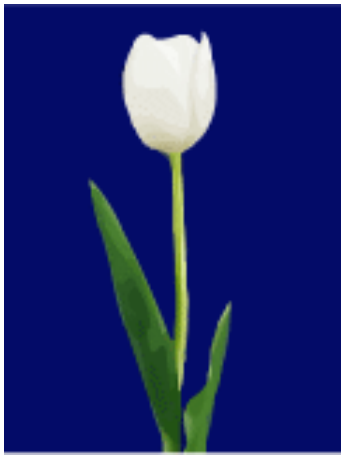

图 7.10　矢量化后的图形效果

图 7.11　抠除背景的图形

7.1.3　交换图像对象

图像对象的交换功能可以快速、方便地实现图像对象的替换，以改变动画制作中的背景等。但是，使用该功能时必须在文档中导入多幅图像对象，而且不允许对舞台上的图像对象进行图像的分离处理和矢量化处理。

交换图像对象的操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+R 导入图像，此时在舞台上就放置了该图像，如图 7.12 所示。当然“ 库 ”面板中也存在该图像对象，可以按 F11 键调用“ 库 ”面板进行查看。

（2）选择“ 文件 ”|“ 导入到库 ”命令，弹出“ 导入到库 ”对话框，选中其他要导入的图像文件后，单击“ 打开 ”按钮即可导入到当前文档的“ 库 ”面板中，如图 7.13 所示。

图 7.12　选中要替换的图像

图 7.13　导入图像的“ 库 ”面板

注意：“导入到库”功能在导入外部文件时不会将该对象置于舞台上；而“导入”功能在导入外部文件时将会自动把该对象置于舞台上。两者各有其优缺点，在使用过程中需要灵活掌握。

（3）选中舞台上的图像对象，选择“修改”|“交换位图”命令，弹出“交换位图”对话框，其中显示了“库”面板中的所有图像对象，单击图像文件 F003，在其左侧框中会自动显示 F003 缩略图，如图 7.14 所示。单击“确定”按钮确认，此时舞台上选中的图像就被替换为“库”面板中名为 F003 的图像对象，如图 7.15 所示。

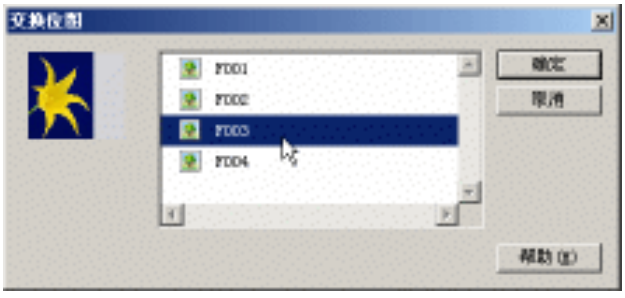

图 7.14 “交换位图”对话框

图 7.15 替换后的图像

使用交换位图功能确实比较简便，但是在“库”面板中将会遗留一些没有使用的图像对象，从而使得动画的文件变得相对要庞大些；而且交换位图也不能应用于经过分离处理的图像对象。现在就针对上述问题，探讨一下快速实现的方法：

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+R 导入图像文件 F001.JPG；复制舞台上的图像对象，对其中一个图像进行分离处理，并使用填充变形工具调整填充效果；而另一个图像则保留为图像对象，如图 7.16 所示。

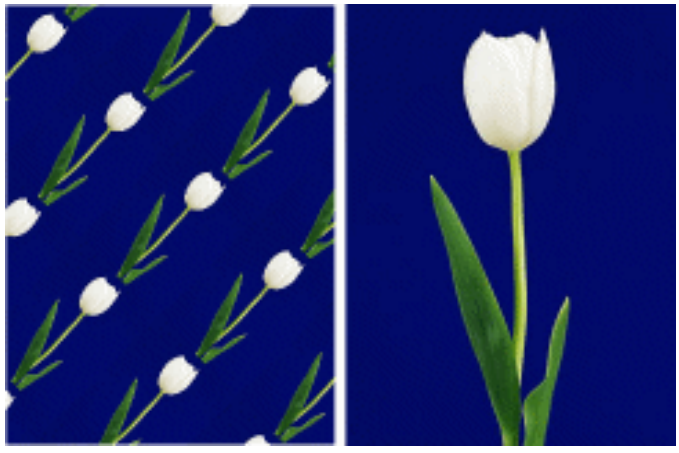

图 7.16 形状及图像

（2）按 F11 键调用“ 库 ”面板，右击其中的图像对象 F001，选择弹出菜单中的“ 属性 ”命令，弹出“ 位图属性 ”对话框，如图 7.17 所示。在该对话框中可以获取该图像的相对路径（.\素材\7.1\F001.jpg）以及一些相关的基本属性。

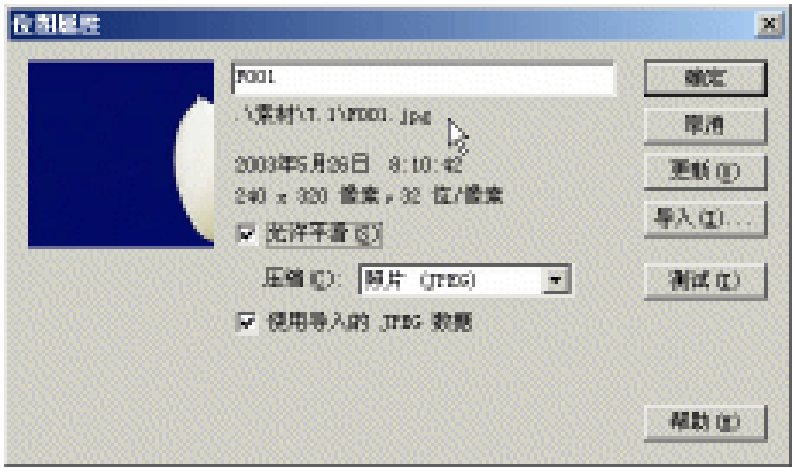

图 7.17　“ 位图属性 ”对话框（一）

（3）单击“ 导入 ”按钮，弹出“ 导入 ”对话框，选中另一幅图像 F003.jpg 并导入，返回“ 位图属性 ”对话框，注意对话框中所发生的变化——导入图像的路径指向了新的位置，如图 7.18 所示。

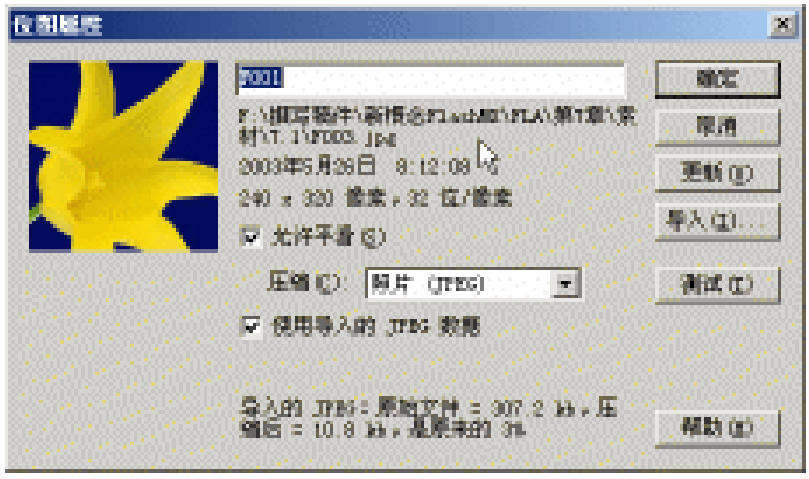

图 7.18　“ 位图属性 ”对话框（二）

（4）单击“ 确定 ”按钮确认，在舞台上经过分离处理的图像及图像对象都发生了相应的变化，如图 7.19 所示。此时再查看“ 库 ”面板中的图像对象，仍然是惟一的图像对象，基本没有任何变化，不同之处就是图像预览也发生了相应的调整。

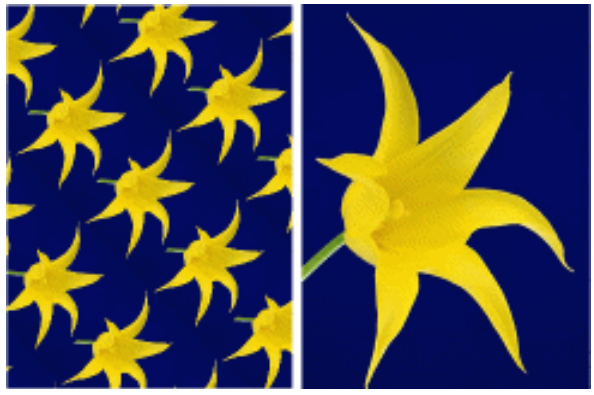

图 7.19　替换后的形状和图像

提示：本例中的替换不同于交换位图功能，主要是由于在“位图属性”对话框中“导入”就是将图像重新导入以替换当前图像，也就是导入图像的重新定位。因此，“库”面板中不会出现多余的图像对象，而且舞台上的图像对象以及形状填充都发生相应调整。

7.2　范例：图像形变

本例的动画效果是图像一幅一幅地移动到舞台上再快速变小，但变小的过程中图案基本上没有变化。这种方式通常应用在动画载入初期，可以采用遮罩技术，也可以采用位图填充方法。本例就是采用位图填充的方法实现的，具体制作步骤如下：

（1）按快捷键 Ctrl+N **新建文档，调用“文档属性”对话框调整尺寸为** 500px×300px、**背景颜色为黑色（**#000000**）、播放速率设置为** 24fps**。**

（2）按快捷键 Ctrl+F8 **创建影片剪辑** mcLoading**，利用文本工具 创建文字块** Loading**，分别在第** 6、11、16 **帧处插入关键帧，并在文字块后依次增加** 1 **个“.”，如图** 7.20 **所示。**

图 7.20　影片剪辑 mcLoading

（3）创建图形元件 graImage**，按快捷键** Ctrl+R **导入一系列** JPG **图像** LX0001、LX0002、…、LX0006**，如图** 7.21 **所示。**

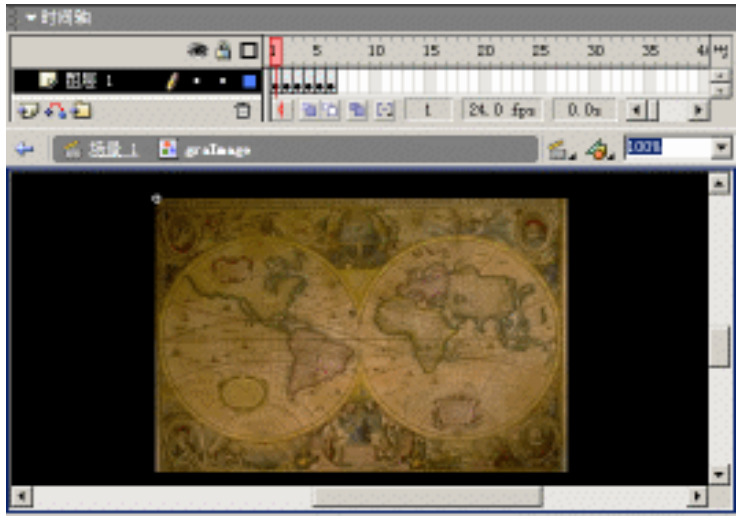

图 7.21　图形元件 graImage

（4）切换到场景编辑状态，将图层名称更改为“图像”，拖动“库”面板中的图形元件 graImage 到舞台上，在“属性”面板的图形选项列表中选择“单帧”，并在其右侧的文本框中输入数值 1；在第 10 帧处插入关键帧，在这两个关键帧之间创建由舞台外移动到舞台中央的运动动画，如图 7.22 所示。

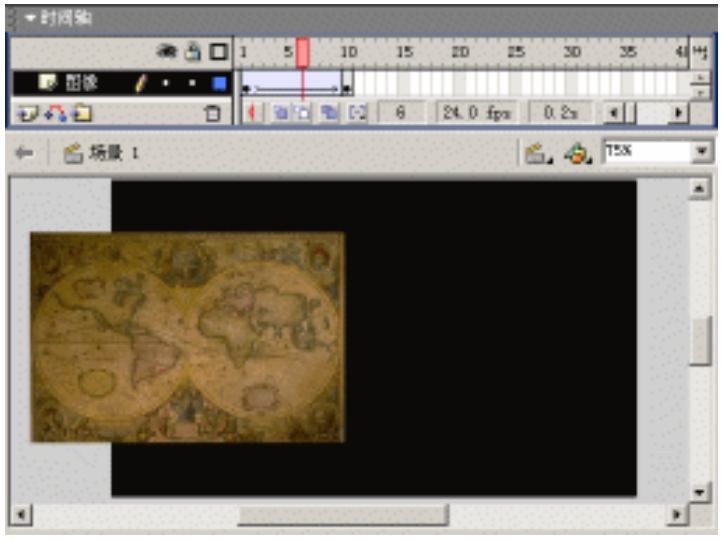

图 7.22　运动动画

（5）在第 11 帧处按 F6 键插入关键帧，连续按两次快捷键 Ctrl+B，将图像完全分离为具有图像填充的图形。在第 20 帧处插入关键帧，利用滴管工具获取填充样式，此时自动切换到颜料桶工具，同时选中了工具选项区的锁定填充；利用“变形”面板将图形尺寸等比例缩小为原有的 10%，再利用颜料桶工具进行锁定填充；然后在这两个关键帧之间创建形变动画，如图 7.23 所示。

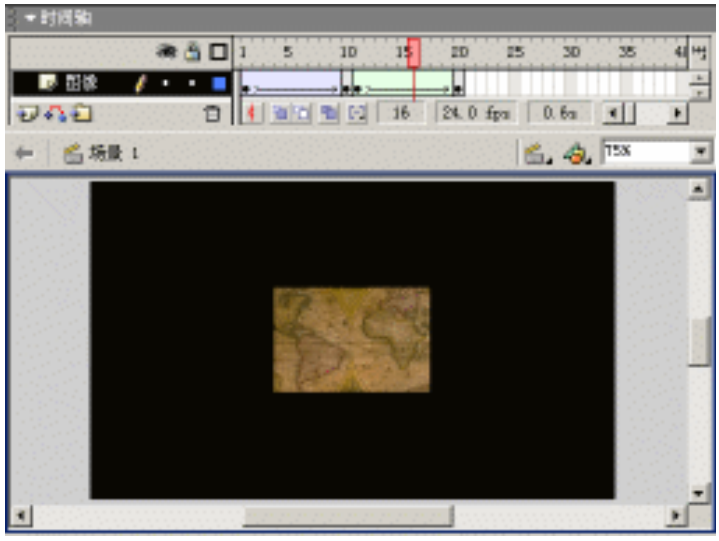

图 7.23 形变动画

注意：本步骤中的操作顺序是不能颠倒的，否则就不会浏览到图像裁剪的动画效果。这主要是由于形变动画中的形状具有相同的位图填充，采用锁定填充就是基于这个目的。

（6）采用上述步骤，重复创建运动动画和形变动画相互间隔的动画效果，所不同的是图像滑入的方向和图像显示的内容进行了调整。最后添加图层 Loading，**将“库”面板中的影片剪辑** mcLoading **拖放到舞台右下角，同时再在舞台左下角创建文本块‘旅行者之家’，如图** 7.24 **所示。**

图 7.24 主时间轴分布

7.3　范例：馋嘴兔子

广阔的草原上有两只馋嘴的兔子，已经饿了好几天……突然，一支萝卜从天而降，兔子的眼睛始终直勾勾地盯着那支萝卜，如图 7.25 所示。如果您比较细心，可以发现动画界面中鼠标光标突然消失，伴随鼠标移动所运动的是动画区域中的萝卜图案。

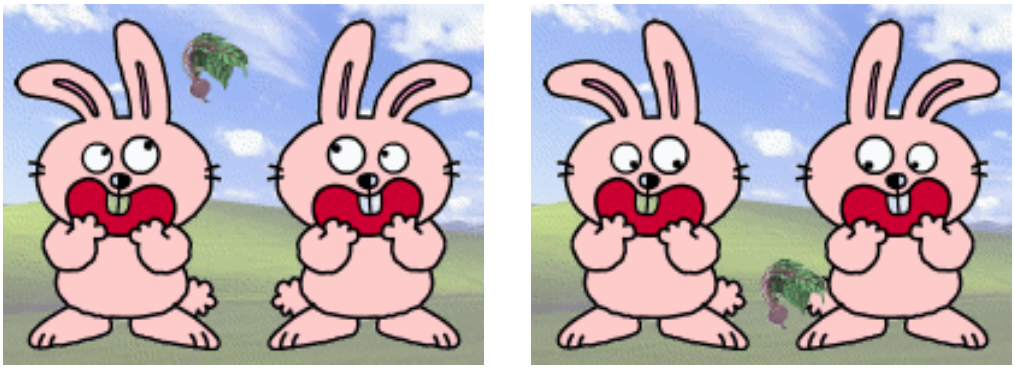

图 7.25　效果预览

7.3.1　动画构思

浏览动画的过程中，兔子的眼睛始终注视着萝卜，而不管萝卜处于任何位置。在动画制作过程中需要解决的两个问题是：如何将光标替换为萝卜图案？如何使得兔子眼睛与光标之间建立直接的联系？

所谓光标的更换，就是将操作系统默认的光标进行隐藏，再拖动动画中的某个影片剪辑。Flash 提供了 mouse.hide 方法用于隐藏光标，而拖动萝卜图案相信您一定会知道使用 startDrag 方法。

兔子眼睛的转动主要通过设置影片剪辑的属性_rotation 来实现。而该角度就是两条直线的夹角，一条是坐标系中的 X 轴，另一条就是由光标所在的坐标与影片剪辑所在的坐标形成的直线。

7.3.2　元件制作

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整背景颜色为绿色（#00CC66），尺寸为 400px×300px。

（2）按快捷键 Ctrl+F8 创建影片剪辑 mcBall，在舞台上绘制黑色圆形，并将其置于十字线的右侧，如图 7.26 所示。

（3）再创建影片剪辑 mcEye，按 F11 键调用“库”面板，将影片剪辑 mcBall 拖放到舞台上，使 2 个十字线相互重叠，并在“属性”面板中设置其实例名称为 eye；再在舞台上绘制白色笔触、黑色填充的圆形作为眼眶部分，如图 7.27 所示。

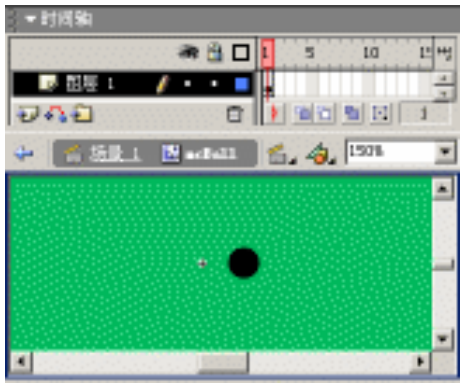

图 7.26 影片剪辑 mcBall

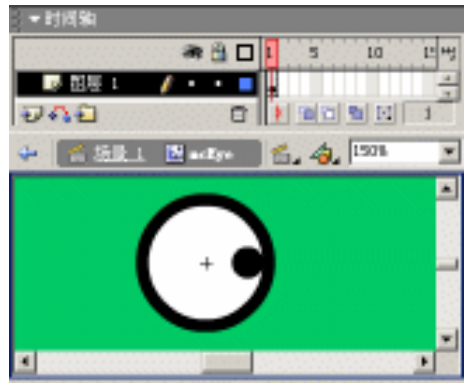

图 7.27 影片剪辑 mcEye

（4）创建影片剪辑 mcCursor，导入萝卜图像，并将其置于舞台中央，如图 7.28 所示。至此，整个动画的全部元件制作完成，“ 库 ”面板中元件列表如图 7.29 所示。

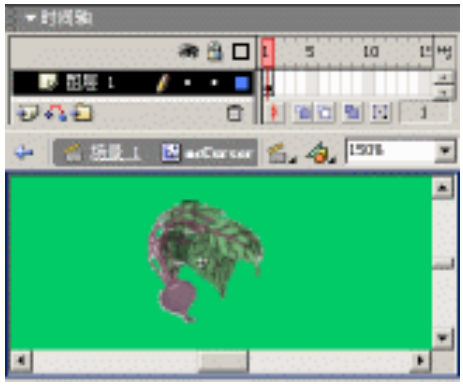

图 7.28 影片剪辑 mcCursor

图 7.29 “ 库 ”面板

7.3.3 脚本编写

（5）切换到场景编辑状态，将图层名称更改为“ 背景 ”，导入背景图像和没有眼睛的兔子图像，适当分布这些图像之间的位置，如图 7.30 所示。

（6）添加图层“ 元件 ”，将“ 库 ”面板中的影片剪辑 mcEye 拖放到舞台上，置于兔子的眼睛部位，按 F9 键调用“ 动作 ”面板，设置影片剪辑 mcEye 的实例脚本内容如下：

```
01  onClipEvent (mouseMove) {
02    arcToangle = 180/Math.PI;
03    arc = Math.atan2(_root._ymouse-this._y, _root._xmouse-this._x);
04    this.eye._rotation = arc * arcToangle;
05  }
```

提示：第 01 行：当在动画区域中移动光标时，触发执行其中的脚本块；
第 02 行：变量 arcToangle 为弧度转换为角度的比率；

第 03 行：变量 arc 是将光标与眼珠构成的线段转换成弧度值；
第 04 行：将弧度转换成角度确定眼珠的转动。

（7）复制眼睛 4 个，将其置于兔子的相应部位；再将“ 库 ”面板中的影片剪辑 mcCursor 拖放到舞台上，并在“ 属性 ”面板中设置实例名称为 cursor。

（8）添加图层“ 脚本 ”，并在该层中设置帧脚本内容，此时的时间轴设置和舞台布置如图 7.31 所示。“ 脚本 ”图层中的帧脚本内容如下：

```
01  Mouse.hide();
02  startDrag("cursor", true);
```

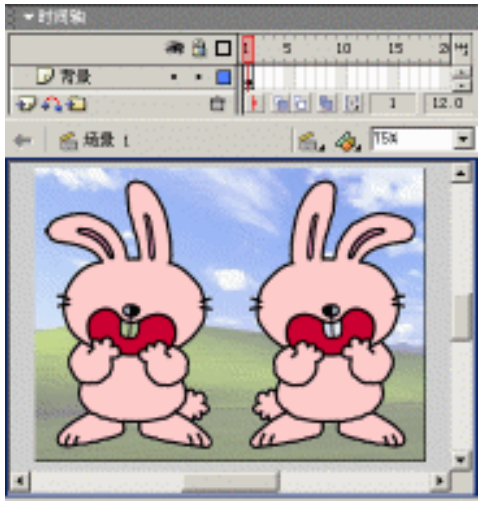

图 7.30　舞台对象布置

图 7.31　最终时间轴设置

提示：第 01 行：隐藏默认的鼠标光标；
第 02 行：将影片剪辑实例 cursor 的中心锁定在光标上进行拖曳。
短短的两行脚本语句就可以实现动画中的光标更换。

至此，馋嘴兔子的动画制作就全部完成，按快捷键 Ctrl+Enter 测试动画，应该注意鼠标移动时，动画区域中萝卜的移动和兔子眼珠的转动。

7.4　范例：蜡烛燃烧

新婚燕尔，点燃喜庆的蜡烛，这是何等的浪漫！蜡烛的火焰伴随着两颗激动的心不停地跳动，如图 7.32 所示。为了体现蜡烛的真实性，拖动蜡烛时，仔细观察一下火焰是否同时移动。

7.4.1　效果构思

整个动画的制作主要分为两部分——蜡烛和火焰。由于蜡烛可以进行拖曳，因此应将蜡烛制作为按钮元件 btnCandle 置于影片剪辑 mcCandle 中，再在按钮实例中使用 startDrag 和 stopDrag 控制蜡烛的拖动。

图 7.32 效果预览

而火焰就是多个影片剪辑 mcFlame 的叠加，由 duplicateMovieClip 语句的复制影片剪辑得到，再利用 setProperty 语句对影片剪辑进行细微的偏移，以实现火焰的抖动。

整个动画中影片剪辑实例的定位相当重要，在制作过程中应该加以重视。

7.4.2 元件制作

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整背景颜色为黑色（#000000）、尺寸为 500px×350px、帧频为 30fps。

（2）按快捷键 Ctrl+F8 创建影片剪辑 mcFlame，利用椭圆工具绘制放射状渐变的圆形填充，值得注意的是放射状渐变的左、右色块的 Alpha 值分别为 100%、0%；将其转换为图形元件 graFlame，并在“属性”面板中设置实例的色调效果为 100%的褐黄色（#E6DC32），如图 7.33 所示。

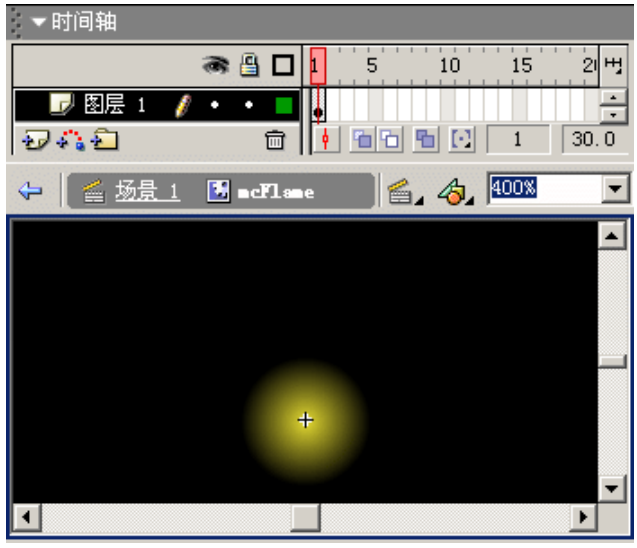

图 7.33 影片剪辑 mcFlame 第 1 帧

（3）在第 12 帧处按 F6 键插入关键帧，适当将实例进行挤压变形并向上移动，同时重新调整实例的色调效果为 100%的红色（#FF3232），如图 7.34 所示。最后再在这两个关键帧之间创建运动动画，形成火焰升起的效果。

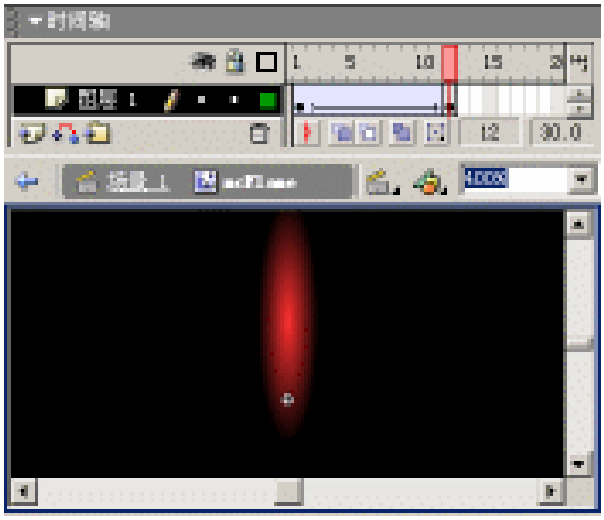

图 7.34　影片剪辑 mcFlame 第 12 帧

（4）创建影片剪辑 mcCandle，导入 PNG 图像“蜡烛”，弹出“Fireworks PNG 导入设置”对话框，选中“作为单个扁平化的位图导入”复选框，如图 7.35 所示。

图 7.35　插图名称

（5）单击“确定”按钮，将导入的蜡烛图像置于舞台的十字线的下方，如图 7.36 所示。按 F8 键将蜡烛图像转换为按钮元件 btnCandle，并设置按钮实例的脚本内容如下：

```
01  on (press) {
02    startDrag("");
03  }
04  on (release) {
05    stopDrag();
06  }
```

提示：第 01～03 行：在蜡烛上按住鼠标左键（press），就可以开始拖曳蜡烛了；
第 04～06 行：在蜡烛上释放鼠标左键（release），也就停止对蜡烛的拖曳。

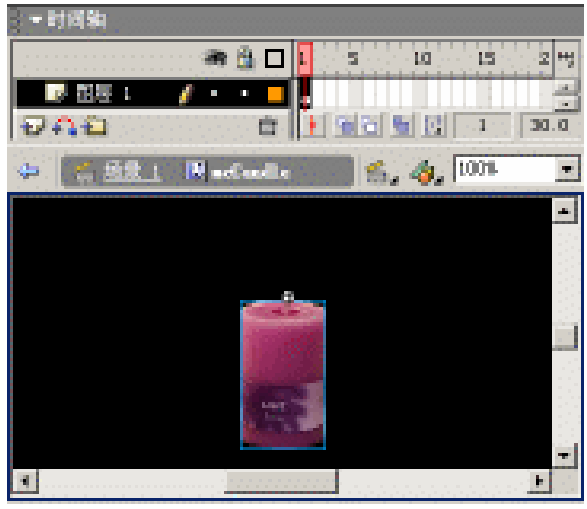

图 7.36　影片剪辑 mcCandle

（6）创建影片剪辑 mcBurn，按 F11 键调用“ 库 ”面板，将其中的影片剪辑 mcCandle、mcFlame 拖放到舞台上，如图 7.37 所示，并在“ 属性 ”面板中分别设置这两个影片剪辑的实例名称为 candle、flame。

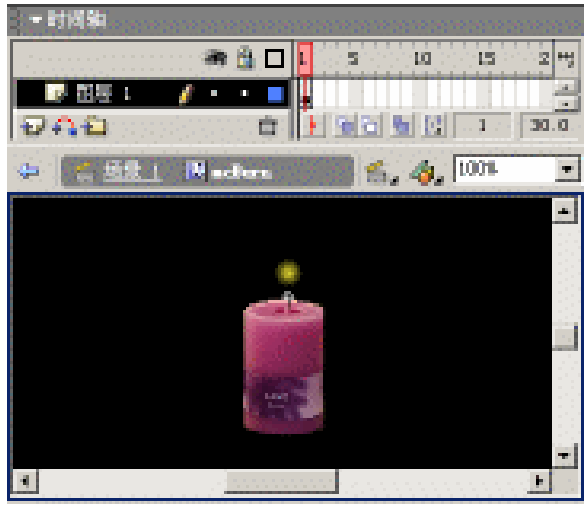

图 7.37　影片剪辑 mcBurn

7.4.3　脚本编写

（7）切换到场景编辑状态，将图层名称更改为“ 背景 ”，导入 JPG 图像“ 婚庆 ”，覆盖整个舞台；添加图层“ 蜡烛 ”，将“ 库 ”面板中的影片剪辑 mcBurn 拖放到舞台左下角的暗处，并在“ 属性 ”面板中设置该影片剪辑的实例名称为 burn。

（8）选中影片剪辑实例 burn 后，调用“ 动作 ”面板设置脚本内容如下：

```
01  onClipEvent (enterFrame) {
02    if (i < 1000) {
03      duplicateMovieClip("flame", "flame"+i, i);
04      this["flame"+i]._x = this.candle._x;
05      this["flame"+i]._y = this.candle._y;
06      i = i + 1;
07    } else {
08      i = 1;
09    }
10  }
```

提示：第 02～07 行：根据变量 i 不断地复制影片剪辑实例 flame（即火焰），再控制复制得到的火焰位置，使其始终处于蜡烛芯的位置处。
此时如果按快捷键 Ctrl+Enter 测试动画，则看到的火焰相当呆板——火焰既没有抖动效果，又遮挡部分蜡烛。

（9）双击影片剪辑实例 burn 进入影片剪辑 mcBurn 编辑状态，选中影片剪辑实例 flame，并在“ 动作 ”面板设置脚本内容如下：

```
01  onClipEvent (load) {
02    xMov = (.5-Math.random())*.5;
03    yMov = -2-Math.random();
04    Alpha = 100;
05  }
06  onClipEvent (enterFrame) {
07    this._x = this._x + xMov;
08    this._y = this._y + yMov;
09    this._alpha = Alpha;
10    Alpha = Alpha-10;
11    if (Alpha < 0) {
12      removeMovieClip("");
13    }
14  }
```

提示：第 01～05 行：当影片剪辑载入（load）时，进行变量的声明及赋值。其中变量 xMov、yMov 分别控制火焰在 X、Y 轴上的偏移，而变量 Alpha 调整火焰的透明度。
第 06～14 行：当影片剪辑处于当前帧（enterFrame）时，不断地调整火焰在 X、Y 轴上的坐标位置及 Alpha 透明度。直到火焰透明度小于 0 时将之移除。

（10）按快捷键 Ctrl+E 切换到场景编辑状态，添加图层“ 脚本 ”，并在该图层中设置帧脚本内容如下：

```
01  duplicateMovieClip("burn", "burn1", 1);
02  burn._x = 50;
```

```
03  burn1._x = 450;
```

思考：通过复制影片剪辑实例 burn 得到一对燃烧的蜡烛，并使得这两支蜡烛处于水平的不同位置处。请思考一下，怎样可以获得更多数目、分布在不同位置处的蜡烛？

最终的时间轴设置和舞台布置如图 7.38 所示。

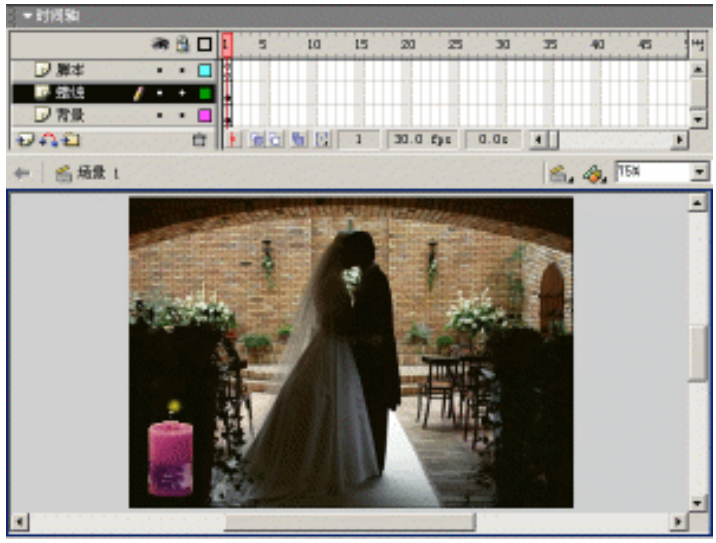

图 7.38　最终的时间轴设置

至此，蜡烛燃烧的动画效果就制作完成了，按快捷键 Ctrl+Enter 测试动画。火焰在不停地跳动，你的思维也伴随着碰撞出火花了吧！！

7.5　范例：画面切换

本例的动画效果基本模拟了百叶窗切换效果。首先可以全无遮挡地观看到绿荫荫的森林图像，随着时间的推移，类似于百叶窗打开逐渐显示出光秃秃的山丘图像；山丘图像完全显示持续一段时间后，再类似于百叶窗逐渐恢复原先的森林图像，如图 7.39 所示。简而言之，百叶窗式的画面切换就是在森林和山丘两幅图像之间循环更替。

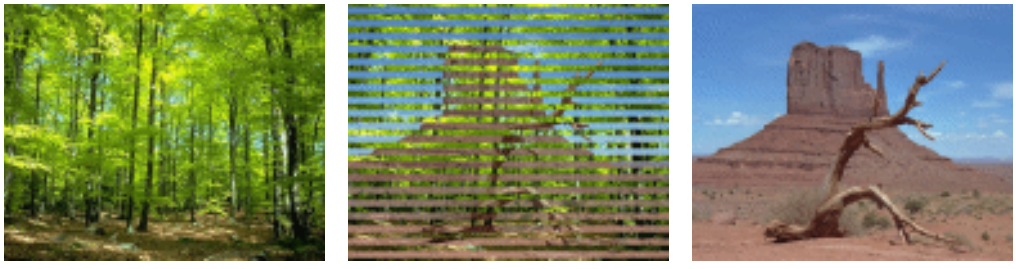

图 7.39　效果预览

7.5.1　效果分析

现实生活中经常会接触到形式各样的画面切换作品，其实就是两幅图像穿插地同时显示，这不仅使作品的前后具有连贯性，而且使作品的观赏性大大提高。

在很多应用软件中都提供了这样的画面切换功能，如幻灯片制作 PowerPoint、视频处理 Premier、课件集成 Director 以及动画制作 GifAnimator 等。但是，Flash 中还没有内置这样的功能，这对广大设计人员来说就比较遗憾了。

在 Flash 中可以模拟其他软件中的画面切换进行制作，如飞入飞出、缩放回旋等切换效果可以使用运动动画来实现。而百叶窗、棋盘式等切换效果就比较棘手了，如果采用一般方法就比较困难，同时将会涉及到大量的遮罩图层或影片剪辑。其弊端就是制作过程相当烦琐，容易出现错误，同时动画文档变得相当庞大；而且对日后的动画修改也不太方便，尤其是图像尺寸存在差异时就更加不易。

最简单的方法就是从 Flash 的脚本功能开始着手。如果仔细比较百叶窗式的画面切换，可以在其中找到各个页片之间的共同点——百叶窗式切换的各个页片同时缩放、而且尺寸基本相同。而惟一的不同之处就是各个页片中显示的图像不同而已！

鉴于百叶窗式画面切换的共性，可以联想到 duplicateMovieClip 语句对影片剪辑的复制功能，而每个页片的图像就是由遮罩组构成的影片剪辑。要使每个页片中显示的图像内容不同，可以利用 setProperty 语句控制影片剪辑中被遮罩层的图像位置。

完成分析后，应该大致清楚百叶窗式画面切换的制作步骤：首先创建影片剪辑 mcShutter 供 duplicateMovieClip 语句使用；影片剪辑 mcShutter 中包含了一个遮罩组，而被遮罩层中图像也必须定义为影片剪辑 mcImage 供 setProperty 语句控制页片中的图像位置。

7.5.2　元件制作

（1）新建文档后按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 500px×400px。

（2）创建影片剪辑 mcShutter，将图层名称更改为“图像”，导入 JPG 图像“风景 2”。将导入的图像转换为影片剪辑 mcImage，并在“属性”面板中设置实例名称为 image。

（3）添加图层“遮罩”，在舞台中央绘制尺寸为 520×20 的矩形填充，第 20 帧处插入关键帧并调整矩形填充的尺寸为 520×1，再在这两个关键帧之间创建形变动画；复制第 1～20 帧后，在第 21 帧处插入关键帧，将矩形尺寸调整为 1×1，随后在第 31 帧处粘贴帧并将第 31～50 帧进行翻转，最后将帧延长至第 60 帧处。

（4）确定图层“图像”和“遮罩”之间的遮罩关系，得到百叶窗一个页片展开和闭合的动画效果，如图 7.40 所示。

> 注意：此处绘制的矩形填充的高度（20）直接影响到最终百叶窗页片的数目，并且也确定了百叶窗页片之间的间距为 20；所绘制的矩形填充在确定坐标时，以中心作为参考的坐标（0,0）。

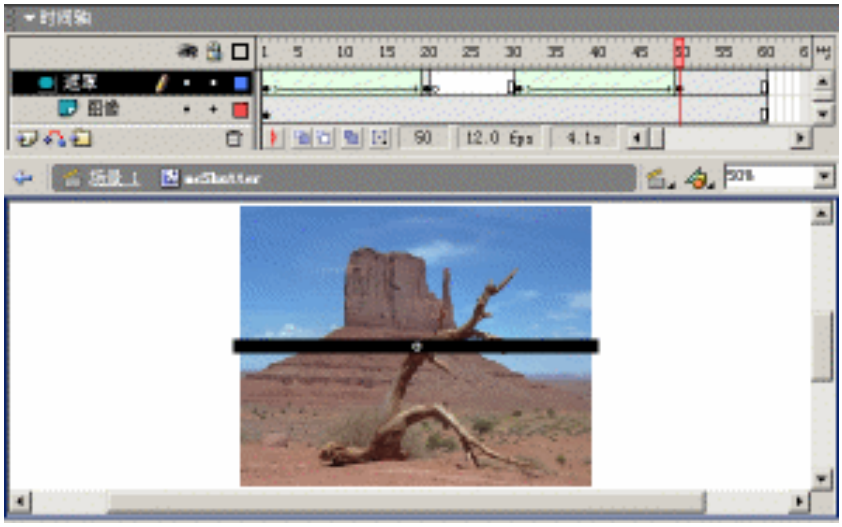

图 7.40 影片剪辑 mcShutter

7.5.3 脚本编写

（5）按快捷键 Ctrl+E 切换到场景编辑状态，将图层名称更改为“背景”，导入 JPG 图像“风景 1”，将其置于舞台中央以覆盖整个舞台。

（6）添加图层“页片”，将“库”面板中的影片剪辑 mcShutter 拖放到舞台上，并在“属性”面板中设置其实例名称为 shutter，最后在“动作”面板中设置该影片剪辑实例的脚本内容如下：

```
01  onClipEvent (enterFrame) {
02    setProperty("image", _x, 0);
03    setProperty("image", _y, 200-this._y);
04  }
```

提示：第 02～03 行：设置当前影片剪辑实例中的实例 image 在 X、Y 轴的坐标位置。如果不进行脚本控制，那么测试动画时就会发现百叶窗在打开和闭合时，只是重复地显示图像的局部，而并不是图像的全部。

（7）添加图层“脚本”，并在该图层中设置帧脚本内容如下：

```
01  setProperty("shutter", _visible, false);
02  for (i=0; i<20; i++) {
03    duplicateMovieClip("shutter", "shutter"+i, i);
04    setProperty("shutter"+i, _x, 250);
05    setProperty("shutter"+i, _y, 20*i+10);
06  }
```

提示：第 01 行：设置影片剪辑实例 shutter 为不可见，防止百叶窗中露出瑕疵；第 02～06 行：利用 for 循环语句复制影片剪辑实例 shutter 20 次，由上至下地紧密排列，最终拼合成百叶窗完整的动画效果。

最终的时间轴设置和舞台布置如图 7.41 所示。

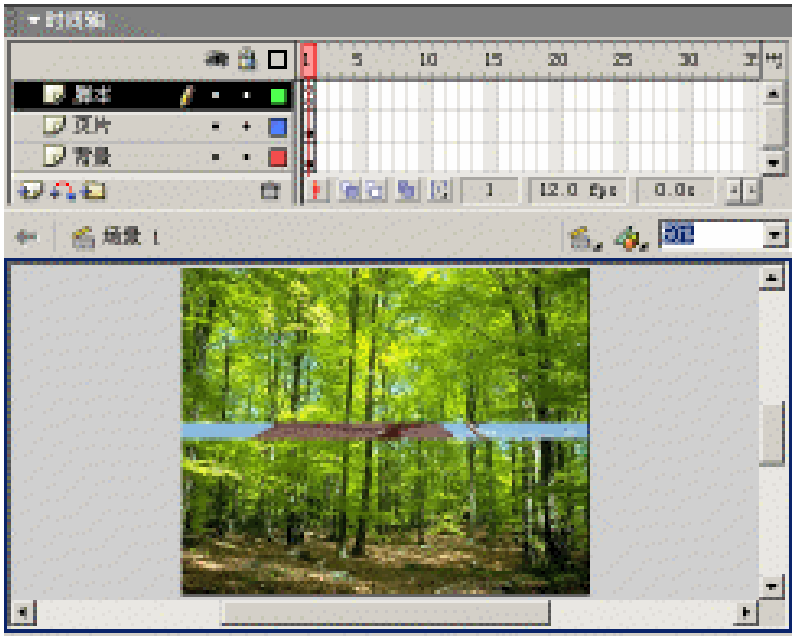

图 7.41　布置舞台对象

至此，两幅图像之间百叶窗式的切换效果就制作完成了，按快捷键 Ctrl+Enter 测试动画。如果有兴趣，可以在此基础上进行深入，制作多幅图像之间的切换效果。

7.6　范例：图像蠕变

浏览本书附带光盘中此例的动画，可以发现光标所在的局部图像扩大、而其他图像却相应缩小。伴随着光标的移动，图像的缩放也同时发生转移；并且图像的缩放过程并不是瞬间发生，而是逐渐地呈液态的转变。

如果将光标移到动画区域外，就可以观看到整个图像恢复到正常状态；如果只看一幅图像的变形比较单调，那么可以在图像上单击鼠标左键即可更换图像，如图 7.42 所示。

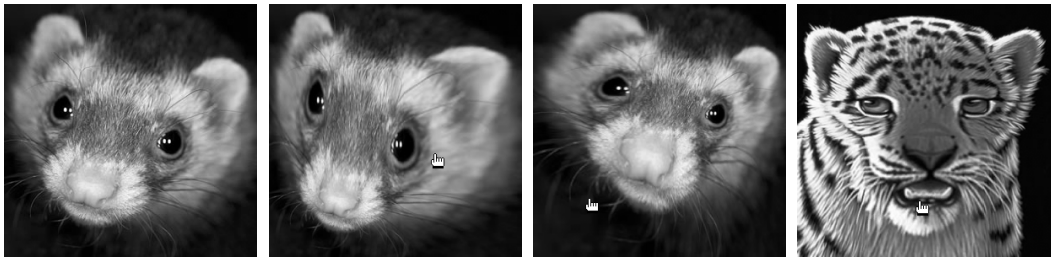

图 7.42　效果预览

7.6.1　效果构思

在前一节中利用脚本实现了百叶窗式的画面切换，相信您可以在本节的动画中察觉出

这种技术应用的一点端倪。本节动画中的局部图像动态变形，的确是利用遮罩技术对图像进行无缝切割。也就是说，动画中的图像被分割为多块小图像后再按照一定顺序拼接而成，结果会使用户在思维上产生只是一幅图像的错觉。

动画中图像的无缝分割只是整个动画制作的部分工作，更重要的是如何促使局部图像发生动态的变形。其实，光标所在的小图像扩大，就是通过隐形按钮进行变量的传递，再确定各个小图像的缩放比例来实现。

7.6.2 元件制作

（1）新建文档后按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 300px×300px。

（2）创建影片剪辑 mcImage，导入以序列命名的 JPG 图像 pic01、pic02、pic03、pic04，并自动将这 4 幅图像置于 4 个关键帧中，如图 7.43 所示。为了防止使用影片剪辑 mcImage 时的自动播放，在各个关键帧中设置帧脚本语句“stop ();”。

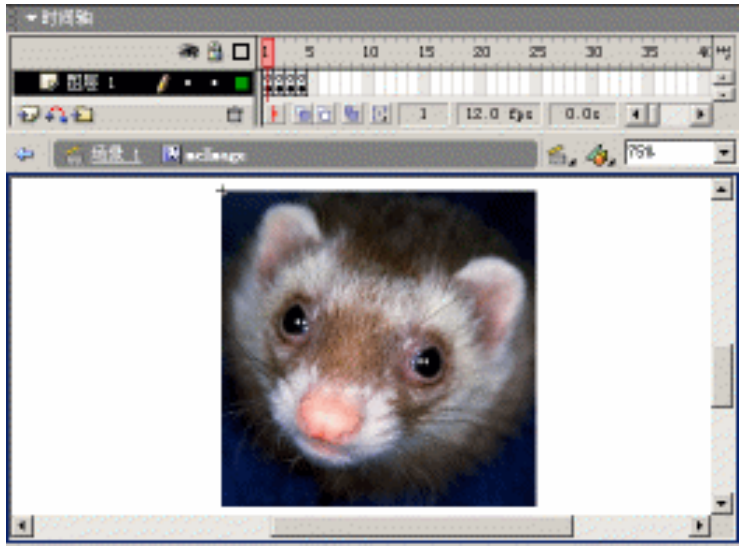

图 7.43 影片剪辑 mcImage

注意：图像导入时，图像的左上角自动与影片剪辑的十字线重叠。如果要将“库”面板中的图像拖放到影片剪辑中，那么调整图像位置时要以图像左上角作为参考的坐标（0, 0），而且图像的尺寸至少为 300×300。

（3）创建按钮元件 btnMask，在“点击”帧中绘制尺寸为 100×100 的矩形填充，如图 7.44 所示。这样，隐形按钮 btnMask 就制作完成了。

（4）创建影片剪辑 mcMask，将图层名称更改为“图像”，将“库”面板中的影片剪辑 mcImage 拖放到舞台上，并在“属性”面板中设置该影片剪辑的实例名称为 image。

（5）添加图层“遮罩”，在舞台上绘制尺寸为 101×101 的矩形填充，矩形填充的左上角与影片剪辑的十字线完全重叠，如图 7.45 所示。右击“遮罩”图层，选择弹出菜单中的“遮罩层”命令创建遮罩。

图 7.44　影片剪辑 btnMask

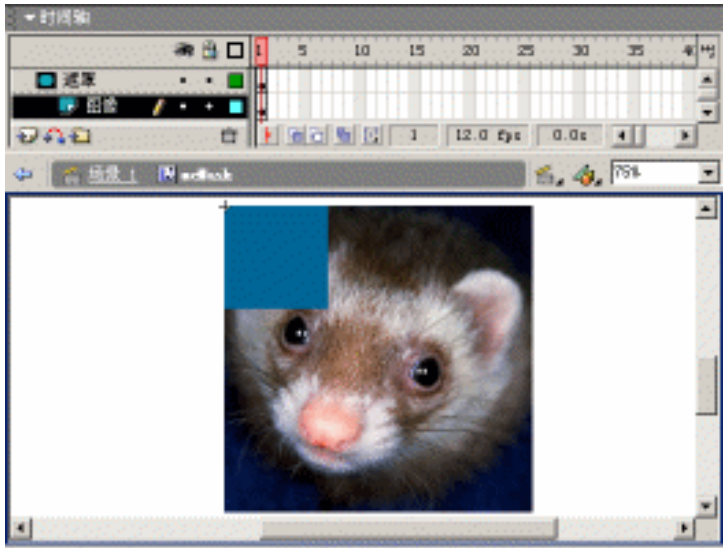

图 7.45　影片剪辑 mcMask

注意“遮罩”图层中矩形填充的尺寸要略微大于按钮元件 btnMask 中的矩形填充，否则最终得到的图像蠕动中的图像块之间存在间隙。

（6）创建影片剪辑 mcSlice，将“库”面板中的影片剪辑 mcMask 拖放到舞台上，并在“属性”面板中定义该影片剪辑的实例名称为 mask。注意，舞台上十字线处于影片剪辑实例 mask 的左上角。

（7）将“库”面板中的按钮元件 btnMask 拖放到影片剪辑实例 mask 上，如图 7.46 所示，再在“动作”面板中设置按钮实例的脚本内容如下：

```
01  on (rollOver) {
02    _root.oi = i;
03    _root.ok = k;
04  }
05  on (rollOut) {
06    _root.oi = 0;
```

```
07    _root.ok = 0;
08  }
```

提示：第 01～04 行：当光标移到按钮区域内（rollOver）时，设置主时间轴中的变量 oi、ok 分别等于当前影片剪辑中的变量 i、k；
第 05～08 行：当光标移到按钮区域外（rollOut）时，将主时间轴中的变量 oi、ok 设置为数值 0。

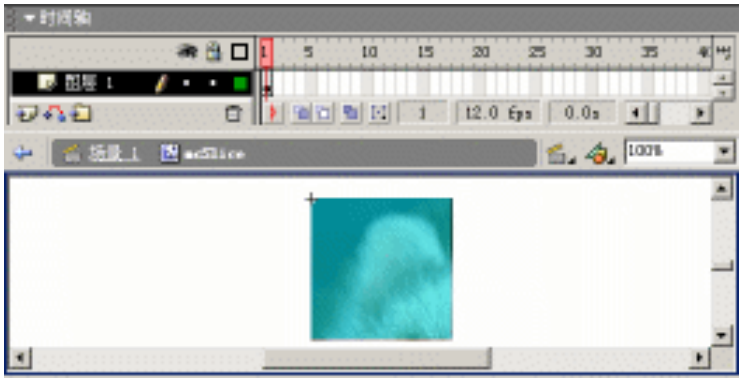

图 7.46　影片剪辑 mcSlice

7.6.3　脚本编写

（8）切换到场景编辑状态，将图层名称更改为“元件布置”，再将“库”面板中的影片剪辑 mcSlice 拖放到舞台上；并在“属性”面板中定义实例名称为 slice。

（9）选中影片剪辑实例 slice，在“动作”面板中设置该影片剪辑实例的脚本内容如下：

```
01  onClipEvent (mouseDown) {
02    this.mask.image.play();
03  }
04  onClipEvent (enterFrame) {
05    if (Number(_root.oi) <> 0) {
06      if (Number(_root.ok) == Number(k)) {
07        newxscale = _root.zoomin;
08      } else {
09        newxscale = _root.zoomout;
10      }
11      if (Number(_root.oi) == Number(i)) {
12        newyscale = _root.zoomin;
13      } else {
14        newyscale = _root.zoomout;
15      }
16    } else {
17      newxscale = _root.cellsize;
18      newyscale = _root.cellsize;
```

```
19   }
20   xstep = (newxscale-xscale)/6;
21   ystep = (newyscale-yscale)/6;
22   if (Number(xscale)<>Number(newxscale)) {
23     xscale = Number(xscale)+Number(xstep);
24   } else {
25      xstep = 0;
26   }
27   if (Number(yscale)<>Number(newyscale)) {
28     yscale = Number(yscale)+Number(ystep);
29   } else {
30     ystep = 0;
31   }
32   setProperty("", _xscale, xscale);
33   setProperty("", _yscale, yscale);
34 }
```

提示：第 01～03 **行：当按下鼠标左键（**mouseDown**）时，使得当前影片剪辑中实例** mask **内嵌套的实例** image **播放。**
第 05～19 **行：根据主时间轴中的变量** oi、ok **确定各个小图像的缩放尺寸。**
第 20～21 **行：设置小图像尺寸转换时的变形幅度** xstep、ystep**，使这种变形逐渐缓慢地发生。**

（10**）添加图层“ 动画脚本 ”，设置帧脚本内容如下：**

```
01 setProperty("slice", _visible, false);
02 block = 5;
03 cellsize = 300/block;
04 zoomin = cellsize+40;
05 zoomout = (block*cellsize-zoomin)/(block-1);
06 temp = 100/cellsize;
07 for (i=1; i<=block; i++) {
08   for (k=1; k<=block; k++) {
09     duplicateMovieClip("slice", "slice"+i+k, 100+i*block+k);
10     setProperty("slice"+i+k+"/mask/image", _xscale, temp*100);
11     setProperty("slice"+i+k+"/mask/image", _yscale, temp*100);
12     setProperty("slice"+i+k+"/mask/image", _x, -cellsize*(k-1)*temp);
13     setProperty("slice"+i+k+"/mask/image", _y, -cellsize*(i-1)*temp);
14     set("slice"+i+k+".i", i);
15     set("slice"+i+k+".k", k);
16     set("slice"+i+k+".xscale", cellsize);
17     set("slice"+i+k+".yscale", cellsize);
18   }
19 }
20 _root.onEnterFrame = function() {
21   for (ypos=0,i=1; i<=block; i++) {
22     for (xpos=0,k=1; k<=block; k++) {
```

```
23        setProperty("slice"+i+k, _x, xpos);
24        setProperty("slice"+i+k, _y, ypos);
25        xpos = xpos+eval("slice"+i+k+".xscale");
26      }
27      ypos = ypos+eval("slice"+i+"1.yscale");
28    }
29  };
```

提示：第 02～06 **行：变量的声明及赋值，其中变量** block **为图像切割的数目（如** 5×5**）、变量** cellsize **确定每个小图像的尺寸大小、变量** zoomin **表示光标所在小图像的尺寸大小、变量** zoomout **表示其余图像区域的尺寸大小。**
第 07～19 **行：利用双重** for **循环语句对整个图像进行分割，再使用** setProperty **语句控制不同图像内容的显示，最后再声明各个复制得到的影片剪辑中的变量。**
第 20～29 **行：当光标所在小图像引起尺寸动态变化时，再利用各个小图像中的变量（**xscale、yscale**）动态确定各个小图像的分布位置（**xpos、ypos**）。**

最终的时间轴设置和舞台布置如图 7.47 **所示。**

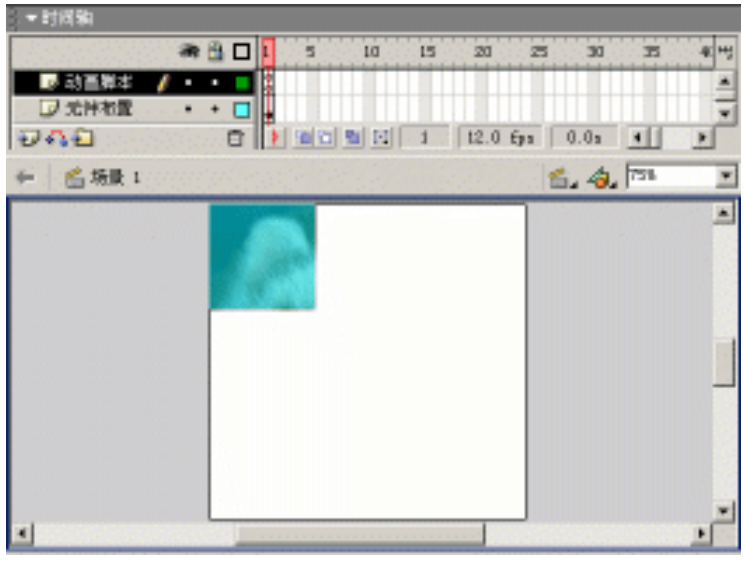

图 7.47 **布置舞台对象**

至此，整个图像蠕动变形的动画就制作完成了，按快捷键 Ctrl+Enter **测试动画，光标在动画区域内移动并停留数秒，看看动画中光标所在的图像区域是否放大了？如果看腻了这幅图像，在动画区域内单击，更换一幅图像看看效果如何？**

7.7 小结和练习

本章详细讲解了图像的导入和使用，尤其是图像的后期处理——分离处理、矢量化和交换等。最后通过 5 **个范例展示了极具图像魅力的动画效果。**

通过本章的学习，希望能够完成以下练习：

1．改变“馋嘴兔子”中作为光标的萝卜图案，使之具有轮换功能。当光标在兔子嘴外部移动时显示整个萝卜图案，当光标在兔子嘴部移动时显示残缺的萝卜图案。

2．改编动画“燃烧蜡烛”，要求在动画画面中放置 3 支蜡烛，调整蜡烛内焰的颜色为紫色、外焰的颜色为蓝色。

3．根据制作百叶窗的解决方案，设法创建另一种场景转换效果——棋盘式切换！

第 8 章　文字动画篇

本章详细介绍了有关文字动画的制作知识。首先对文本对象进行合理的分类，使您能够对 Flash 中涉及的文本类型有较为全面的认识。然后利用“分离”功能对文字进行个性化处理，以求得到标新立异的目的。最后通过多个动画效果展示文字动画的魅力，其中部分效果也可以移植到图像动画中。

通过本章的学习，应该达到如下目的：

- 了解文本对象类型
- 熟悉文本属性设置
- 灵活使用遮罩技术
- 掌握并应用动态文本

8.1　文 本 对 象

虽然生活中的信息传递已经趋于多元化，但是在任何传播媒介中，文字总是起着非常重要的作用。无可置疑的是，虽然图像也可以传递大量的信息，但始终没有文字所表达得迅速和准确。

在动画制作过程中，文字的应用也起着相当重要的作用。而且其中的文字比书籍阅读时所看到的文字具有更加生动的效果。在 Flash 中可以将文字以多种效果淋漓尽致地呈现，如循环滚动、若隐若现、伸缩旋转，甚至是十分逼真的爆炸效果。同时需要清楚的是，Flash 中的文字也具有交互功能。

8.1.1　文本对象类型

Flash 提供了 3 种类型的文本对象，即静态文本、动态文本和输入文本。这三种文本对象的区别主要在于其功能的不同。静态文本创建时，可以决定其文本内容及其外观；动态文本应用时，通常在动画过程中动态更新其文本内容；而输入文本的使用主要是允许动画浏览用户在各种表单中根据需要输入相应的文本内容，从而体现人机交互功能。

在动画制作过程中，完成上述 3 种文本类型的创建后，没有选中的状态下各自的外观显示略有不同，静态文本只显示文本内容，而动态文本和输入文本在显示文本内容的同时其外围呈现点状线框，如图 8.1 所示。

图 8.1　不同文本类型的外观

现在再来看看不同文本类型的文本编辑状态，选择工具箱中的文本工具，在“ 属性 ”面板中选择文本类型为 “ 静态文本 ”，如图 8.2 所示。

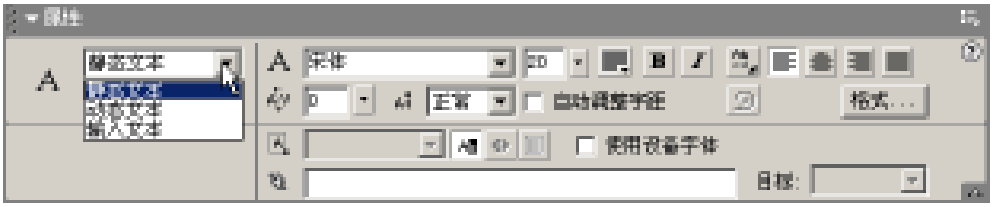

图 8.2　使用文本工具时的 “ 属性 ” 面板

如果在舞台上单击，将呈现右上角为空心圆圈的矩形框，在此输入文本内容即可。随着文字的输入，矩形框横向逐渐伸长。如果在舞台上按住鼠标左键横向拖动，就会呈现右上角为空心方形的矩形框，在此输入文本内容即可。随着文字的输入，矩形框将纵向伸长。上述 2 种方法输入静态文本中的文字内容时的手柄显示如图 8.3 所示。

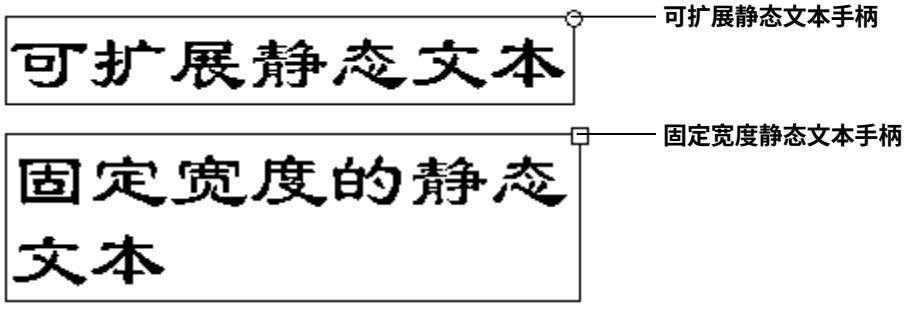

图 8.3　静态文本的各种手柄显示

技巧：静态文本的可扩展方式和固定宽度方式是可以相互转换的。横向拖动可扩展方式静态文本右上角的空心圆圈，就会自动变为空心方形，也就是转换成固定宽度方式的静态文本；双击固定宽度方式静态文本右上角的空心方形，就会自动变为空心圆圈，也就是转换成可扩展方式的静态文本。

选择工具箱中的文本工具，在 “ 属性 ” 面板中选择文本类型为 “ 动态文本 ”。如果在舞台上单击，就会出现右下角为空心方形的矩形框，表示该文本为固定宽度的动态文本。

选择 “ 文本 ”|“ 可滚动 ” 命令，此时右下角的空心方形变为黑色方块，表示该文本为可滚动的动态文本。可滚动的动态文本通常与滚动条组件联合使用，从而使得文本可随意翻屏。

双击固定宽度的动态文本右下角的空心方形，就会自动变为空心圆圈，也就是转换成可扩展的动态文本。

上述 3 种方法输入动态文本中的文字内容时的手柄显示如图 8.4 所示。“ 输入文本 ”的各种手柄显示与动态文本完全相同，在此就不再重复介绍了。

技巧：通过选择 “ 文本 ”|“ 可滚动 ” 命令可改变动态文本的可滚动属性。其实还有更为简便的方法，那就是按住 Shift 键双击动态文本的手柄。

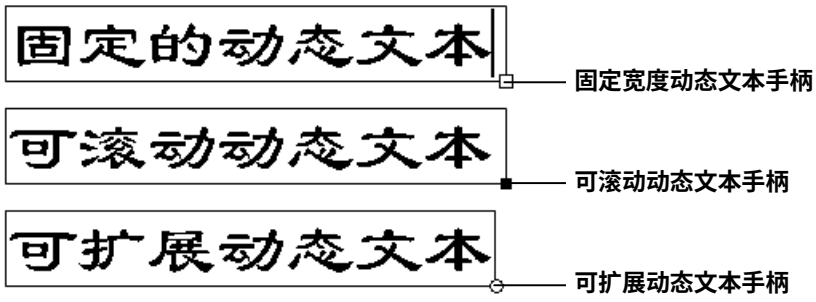

图 8.4 动态文本的各种手柄显示

8.1.2 文本属性设置

虽然 Flash 的文字处理并不如专业排版软件 PageMaker、Word 的功能强大，但是在动画制作过程中具有这样的文字处理功能就已经足够了。Flash MX 允许静态文本以古书纵排方式进行排版布局。

静态文本属性设置的具体操作步骤如下：

（1）按快捷键 Ctrl+N 新建文档，按 T 键切换到文本工具，在舞台上创建固定宽度的静态文本，并输入文本内容。随后在如图 8.5 所示的“属性”面板中适当设置静态文本的字体、字号、文本颜色等属性，最终得到的静态文本如图 8.6 所示。

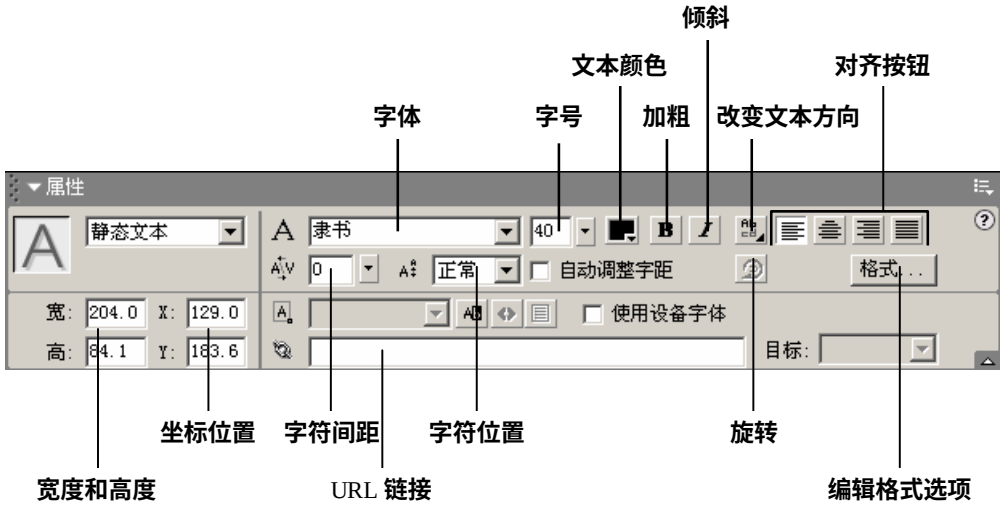

图 8.5 静态文本的“属性”面板

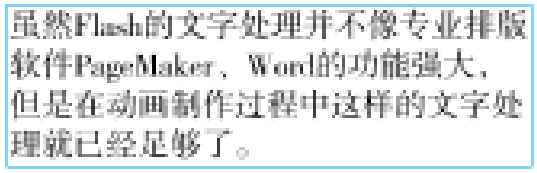

图 8.6 格式化后的静态文本

（2）选中静态文本的情况下，选择“属性”面板中的“改变文本方向”按钮，在弹出的下拉菜单中选择“垂直，从右向左”命令，适当调整静态文本的高度。此时，静态文本中的英文单词显得零散且不协调，选中其中的英文单词后选择“属性”面板中的“旋转”按钮，英文单词旋转 90 度后就比较紧凑了，如图 8.7 所示。

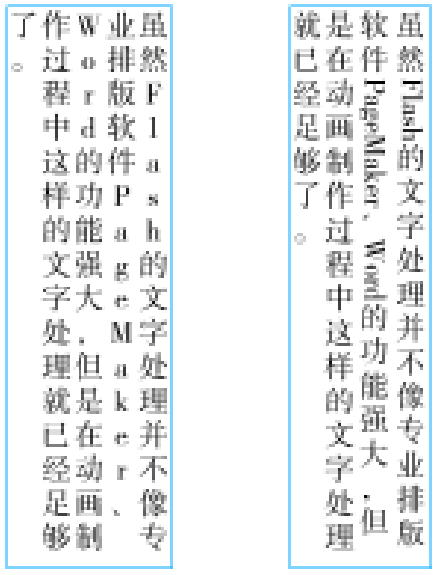

图 8.7　纵向排列的静态文本

（3）通常每段的起始位置要缩进 2 个字符的位置，下面来实现缩进效果。选中静态文本的情况下，单击“属性”面板中的“格式”按钮，弹出“格式选项”对话框。在该对话框中设置缩进为 46px，如图 8.8 所示。

图 8.8　“格式选项”对话框

（4）单击“完成”按钮后，静态文本中的文字纵向缩进 2 个文字，如图 8.9 所示。

提示：如果静态文本调整为“水平”横向文本，“属性”面板中的对齐按钮也会相应发生变化。而且“格式选项”对话框中的参数也相应发生调整，如图 8.10 所示。

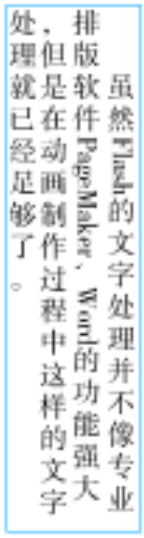

图 8.9 纵向缩进

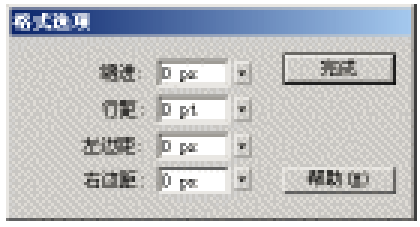

图 8.10 “格式选项”对话框（水平）

提示：静态文本的“属性”面板中的字符位置列表框 正常 中有 3 中选项，即正常、上标和下标。利用该功能可以在静态文本中设置得到化学方程式 $2H_2O=2H_2+O_2$ 和数学公式 $aX^3+bX^2+cX+d=0$ 等效果，如图 8.11 所示。

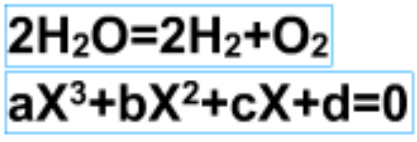

图 8.11 应用字符位置

上面介绍了静态文本的“属性”面板，下面再了解一下动态文本的“属性”面板。其实，动态文本的“属性”面板中的设置与静态文本的“属性”面板基本相同，如字体、字号、文本颜色等。不同的是，“改变文本方向”按钮 和“旋转”按钮 是静态文本所特有的，而动态文本也有其特有的属性，如实例名称、行类型、变量名称等，如图 8.12 所示。

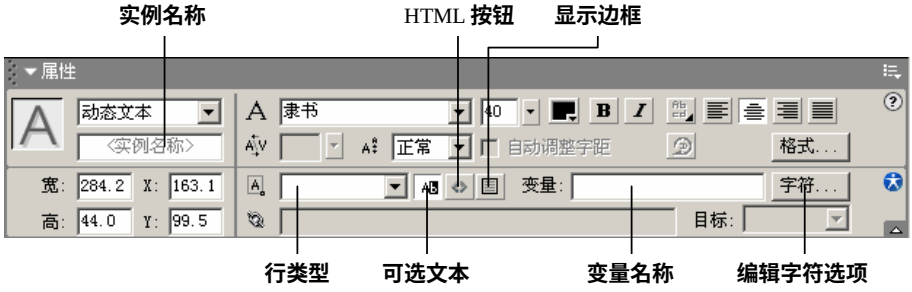

图 8.12 动态文本的“属性”面板

8.2　范例：花样文字

像霹雳体、妞妞体、竹节体等个性化字体的使用打破了文字一成不变的呆板局面，也拓展了设计人员的创作思维。如果计算机中没有安装这些字体，就不能使用这些字体了，难道在 Flash 中就仍旧忍受那些呆板的效果吗？

其实不然，利用 Flash 提供的“分离”功能可以轻松实现文字的新花样。现在就在 Flash 中制作两种花样文字效果——立体文字和图案文字，以拓展您在 Flash 中的创作思维！

8.2.1　立体文字

Flash 创建的动画是二维的平面动画，那么如何创建立体文字呢？这主要是利用不同明暗程度的颜色块拼接形成三维效果，通常称为假三维。立体文字就是基于这种方法制作得到的。具体步骤如下：

（1）按快捷键 Ctrl+N 新建文档，在舞台上创建字体为 Arail Black、字号为 200 的静态文本 KH。选中文本 KH 的情况下，将“属性”面板中的“字符间距”列表框 设置为 10，适当放大这两个字符之间的间距。

> 提示：字符间距除了可以在“属性”面板中设置外，还可以选择“文本”|“间距”级联菜单中的命令。扩大字符间距的快捷键为 Ctrl+Alt+→，缩小字符间距的快捷键为 Ctrl+Alt+←，恢复字符间距的快捷键为 Ctrl+Alt+↑。

（2）选中文本 KH 的情况下，连续按快捷键 Ctrl+B 两次将文本完全分离为矢量图形。按 S 键切换到墨水瓶工具，设置笔触颜色为红色（#FF0000）、笔触样式为“极细”，在分离后文本的周围勾勒文字 KH 线框，如图 8.13 所示。选择文字 KH 中间的填充部分后，按 Delete 键删除，剩下文字 KH 线框，如图 8.14 所示。

图 8.13　分离文本

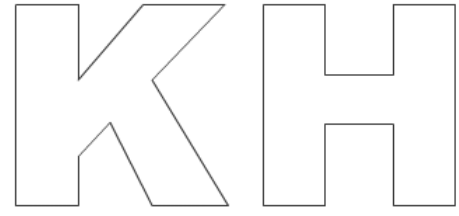

图 8.14　文字线框

（3）按 V 键切换到箭头工具，选中文字 KH 的全部的线框，按快捷键 Ctrl+G 进行组合。随后复制文字 KH 线框，使两个文字 KH 线框交叉放置，如图 8.15 所示。

（4）按 N 键切换到线条工具，选中“视图”|“对齐对象”命令，将这两个线框的相同顶点连接起来，如图 8.16 所示。

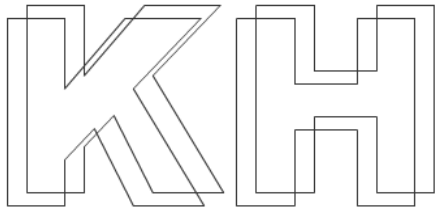

图 8.15 复制文字线框

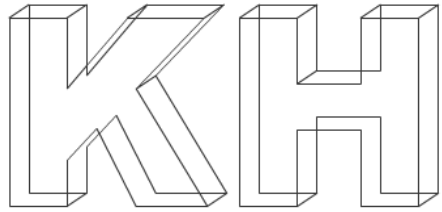

图 8.16 连接对应顶点

（5）再次切换到箭头工具，按快捷键 Ctrl+A 选中全部对象，按快捷键 Ctrl+Shift+G 取消线框的组合状态，再根据立体文字的可见面，逐个删除线框中不可见的线条。删除全部不必要的线条后，立体文字的框架也就基本呈现出来了，如图 8.17 所示。

（6）选择颜料桶工具，在各个线框区域内填充不同明暗程度的颜色。最后将确定立体文字的线框全部删除，如图 8.18 所示。

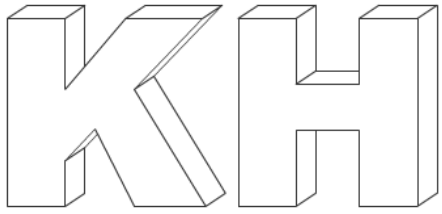

图 8.17 删除不可见线条

图 8.18 立体文字效果

通过立体文字的制作，应该基本了解 Flash 创作环境中的“ 3D 建模 ”所采用的手法，这通常需要您具有很强的空间感。

8.2.2 图案文字

在文字中均匀地分布同一张精美的图像，文字的视觉效果即增色不少。或许，您已经想到采用遮罩技术实现这种图案文字，那么必须先创建具有均匀分布相同图案的图像。

其实，完全没有必要采用遮罩技术，因为 Flash 提供了位图填充的功能。利用它，图案文字的制作只要导入任意图像即可得到。具体步骤如下：

（1）按快捷键 Ctrl+N 新建文档，在舞台上创建字体为 Arail Black、字号为 200 的静态文本 KH，连续按快捷键 Ctrl+B 直到完全分离为矢量图形，如图 8.19 所示。

（2）选择“ 文件 ”|“ 导入到库 ”命令，将 JPG 图像 F0001、F0002、F0003、F0004 等导入到“ 库 ”面板中。按 K 键切换到颜料桶工具，并选中工具选项区中的“ 锁定填充 ”按钮，展开“ 混色器 ”面板并选择填充样式为“ 位图 ”，此时导入到“ 库 ”面板中的 JPG 图像以缩略图的方式排列在该面板中，如图 8.20 所示。

（3）选择其中的“ 向日葵 ”位图后，逐个单击分离文字的内部进行填充，此时文字中图案填充并不是太令人满意，如图 8.21 所示。

图 8.19　分离后的文字

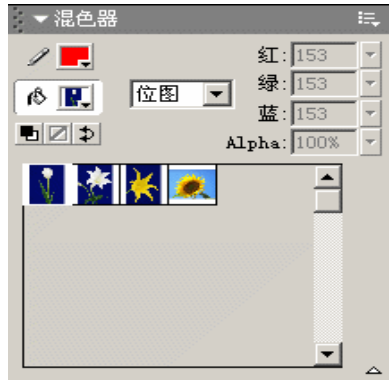

图 8.20　“混色器”面板

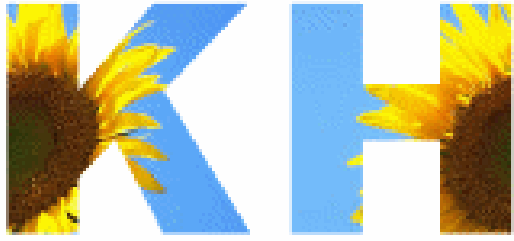

图 8.21　添加图案填充的文字

（4）按 F 键切换到填充变形工具，单击分离文字内的任意位置，显示位图填充的调整手柄，将位图填充适当缩小、旋转并移动位置，最终得到均匀分布精美图案的文字效果，如图 8.22 所示。

图 8.22　调整图案填充的文字

（5）通过图案文字的制作，掌握位图填充的同时也更熟悉颜料桶工具和填充变形工具的使用方法。由于文字 KH 已经被分离成矢量图形，因此可以利用箭头工具调整文字 KH 的顶点及边，如图 8.23 所示。

图 8.23 变形后的图案文字

8.3 范例：幻影文字

本例的效果是，动画浏览时，淡红色的文字“幻影文字”处不断地闪出光影，如图 8.24 所示。实现此效果的具体操作步骤如下：

图 8.24 效果预览

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框并调整尺寸为 400px×200px。更改图层名称为“文字 1”，按 T 键切换到文本工具，在舞台上创建红色的静态文本“幻影文字”。

（2）选中静态文本块的情况下，按 F8 键调用“转换为元件”对话框，输入名称为“文字”，确定元件行为为“图形”，如图 8.25 所示，单击“确定”按钮确认。

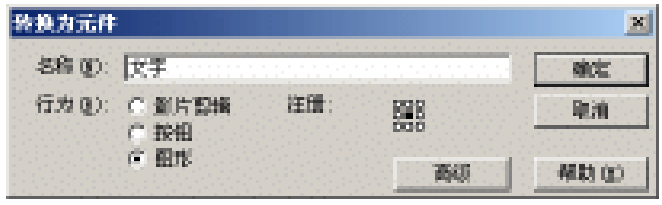

图 8.25 转换为图形元件“文字”

（3）选中图形元件“文字”实例的情况下，在“属性”面板中设置 Alpha 效果为 50%，随后在第 19 帧处按 F5 键插入帧使之延长到第 19 帧，如图 8.26 所示。

（4）选中“文字 1”中的实例，按快捷键 Ctrl+C 进行复制；添加新图层“文字 2”，在第 4 帧处按 F7 键插入空白关键帧，再按快捷键 Ctrl+Shift+V 进行原位粘贴；随后在第 13 帧处按 F6 插入关键帧，将该帧中的实例适当放大（约为 125%），Alpha 效果调整为 0%，最后在这两个关键帧之间创建运动动画。

图 8.26　图层“文字 1”

（5）选中图层“文字 2”，选择“编辑”|“拷贝帧”命令复制该图层中的所有帧，添加新图层后再选择“编辑”|“粘贴帧”命令粘贴刚才复制的所有帧。更改图层名称为“文字 3”，并将运动动画后移至第 7 帧。

技巧：选中图层中的连续的动画帧后，将光标定位在所选的帧上，按住 Alt 键后拖动鼠标到其他图层中即可完成帧的复制和粘贴。

（6）为了突出动画效果的层次感，再添加新图层进行帧的粘贴，并将之延迟到第 10 帧，最终得到的时间轴分布显示如图 8.27 所示。

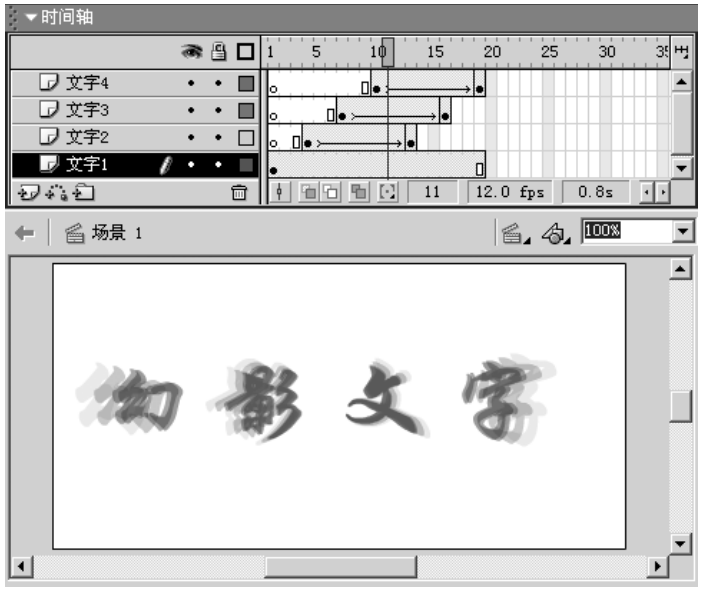

图 8.27　最终的时间轴分布

在本例动画制作的过程中，可以深刻体会到时间轴对动画制作的重要性。虽然图层“文字 2”、“文字 3”和“文字 4”中的运动动画是完全相同的，但是由于各个图层中所处的时间轴先后次序不同，使得最终可观看到文字幻影的动画效果。

8.4 范例：文字光效

动画浏览时，可以在画面中观看到光芒不断发生变化的文字效果。随着文字光芒的左右摆动，文字光芒的颜色也不断地发生着变化，如图 8.28 所示。

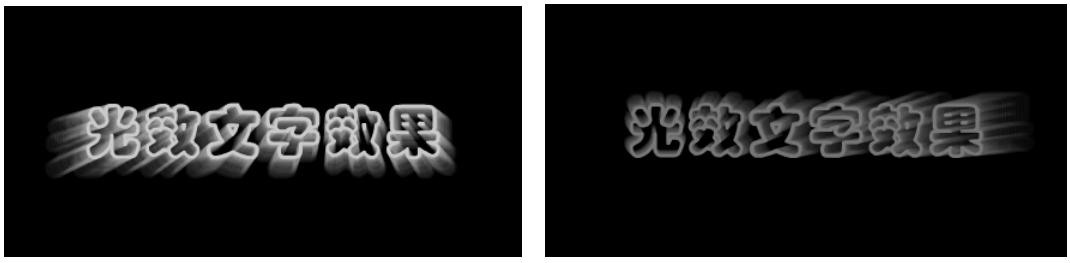

图 8.28 效果预览

文字光效的制作并不是太复杂，主要由一些简单脚本合理编写而成。如果您仔细观察动画效果，可以从中寻找出一些蛛丝马迹来。文字闪出的光效与中间的文字基本相似，所不同的就是尺寸和透明度。因此，可以从中了解到文字的光效其实就是很多不同透明度、不同尺寸的相同的文字在位置上不同程度的偏移叠加而成，由此可见脚本编写中肯定少不了 duplicateMovieClip、setProperty 等语句的使用。

除了上述脚本语句的使用外，动画中所谓的光效是众多文字叠加而成，而且光效部分始终处于一定的范围内移动。因此，必然会使用到循环语句 for 和判断语句 if。

8.4.1 元件制作

（1）新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，尺寸默认为 550px×400px，背景颜色更改为黑色（#000000），帧频调整为 24fps。

（2）创建影片剪辑 mcText，在舞台中央创建静态文本“光效文字效果”。将该文本转换为图形元件 graLight，随后创建多个关键帧的运动动画，并调整各关键帧中实例的色调效果，如图 8.29 所示。

提示：静态文本“光效文字效果”所采用的字体是“汉仪彩云体简”。如果您没有这种字体，但又要得到镂空文字的效果，可以先将文字分离成形状，然后再利用墨水瓶工具形成文字线框，最后删除文字形状即可。

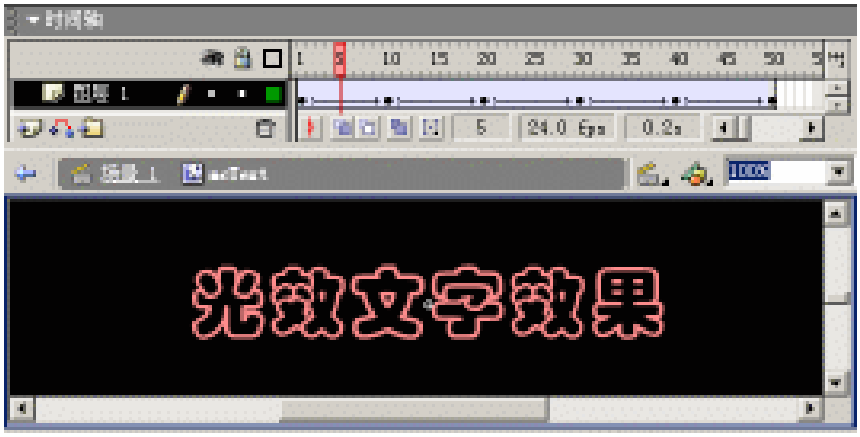

图 8.29　影片剪辑 Light

8.4.2　元件布置

（3）切换到场景编辑状态，更改图层名称为“元件”，将“库”面板中的影片剪辑 mcText 拖放到舞台上，并在“属性”面板中定义实例名称为 text0。

8.4.3　脚本编写

（4）添加图层“脚本”，在该图层中设置帧脚本内容如下：

```
_root.onLoad = function() {
  nMax = 25;
  xMov = .03;
  yMov = .01;
  xMax = .1;
  yMax = .05;
  for (i=1; i<=nMax; i++) {
    duplicateMovieClip("text0", "text"+i, i);
    _root["text"+i]._xscale = _root["text"+(i-1)]._xscale+.05*i;
    _root["text"+i]._yscale = _root["text"+(i-1)]._yscale+.05*i;
    _root["text"+i]._alpha = 20-.5*i;
  }
};
_root.onEnterFrame = function() {
  xTmp = xTmp+xMov;
  if (xTmp>xMax || xTmp<-xMax) {
    xMov = -xMov;
    yTmp = yTmp+yMov;
    if (yTmp>yMax || yTmp<-yMax) {
      yMov = -yMov;
    }
  }
  for (i=1; i<=nMax; i++) {
    _root["text"+i]._x = _root["text"+(i-1)]._x+i*xTmp;
```

```
25      _root["text"+i]._y = _root["text"+(i-1)]._y+i*yTmp;
26    }
27  };
```

提示：第 07～12 行：利用 for 语句循环复制生成多个电影剪辑 mcText 的实例，设置各实例的不同尺寸及透明度，以形成光芒效果；
第 15～22 行：使用两个 if 判断分别改变光芒在 X 轴和 Y 轴上的移动方向；
第 23～26 行：利用 for 语句控制各实例的位置，由于处于 onEnterFrame 程序块中，因此各实例的位置是动态变化的，也就形成了文字光芒移动的效果。

最终的时间轴设置和舞台布置如图 8.30 所示。

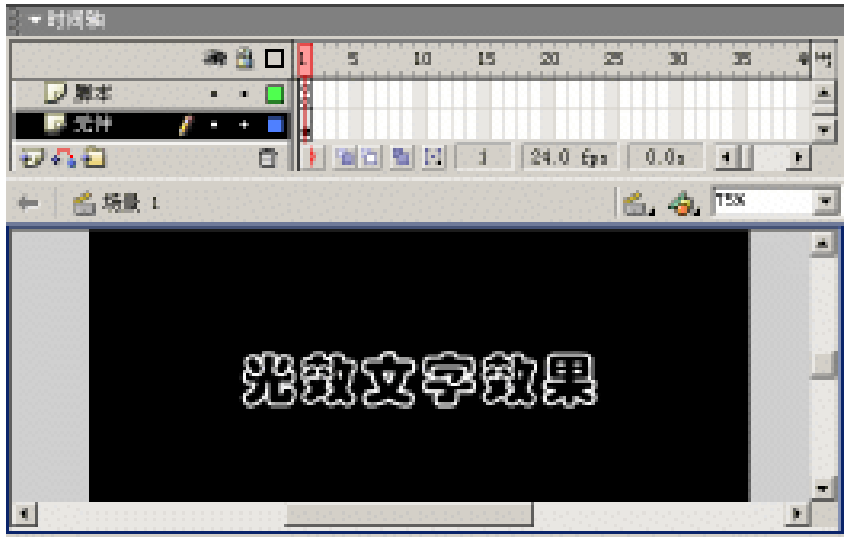

图 8.30　最终的时间轴设置和舞台布置

至此，光效文字效果的动画制作就全部完成了，按快捷键 Ctrl+Enter 测试动画！由于动画中的脚本运行频率较高，而且每次所发生的变化也较小，所以观看光效文字效果会有些滞留感。可以通过更改其中的变量 xMov、yMov、xMax、yMax 的数值，达到调整文字光芒的移动间距，以改善动画的滞留感。

8.5　范例：光束成字

漆黑的世界一无所有，突然一束光线划破这片黑暗。伴随着光束的入射和移动，逐渐显示出“光束成字效果”的字样，如图 8.31 所示。如果您浏览动画时比较细心，就会发现入射的光束会根据显示文字内容的不同而相应发生变化。

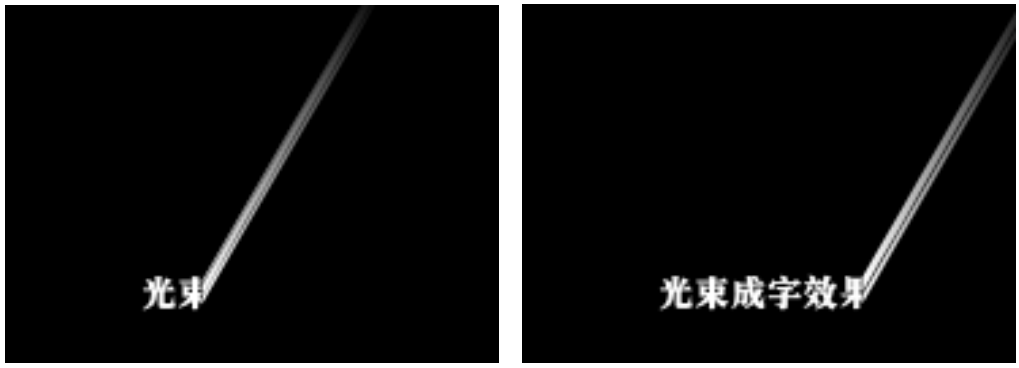

图 8.31　效果预览

光束成字效果中的制作难点就在于光束部分的形成——根据显示文字部分的不同，光效实时发生着变化。其实，这只是遮罩技术更深层次、更为灵活的应用结果。通过动画的预览，应该不难发现文字显示部分采用了遮罩技术。也许，您还不能确定光束是如何产生的。其实，光束的形成也是使用遮罩的结果——在文字中截取 1 个像素宽度的形状进行很大程度的拉伸。

本例中的脚本编写比较简单，主要利用语句 serProperty 实时调整文字的显示和光束的形成。光束成字效果的产生完全是多个影片剪辑之间协同控制的结果。

为了减少脚本代码的编写，范例动画的制作过程中需要精确控制各个影片剪辑的位置，灵活使用“信息”面板。只要在制作过程中有一点疏忽，最终得到的动画效果或许就是另外一种结果。

8.5.1　元件制作

（1）新建文档后按快捷键 Ctrl+J 调用“文档属性”对话框，尺寸默认为 550px×400px，背景颜色更改为黑色（#000000），帧频设置为 30fps。

（2）创建影片剪辑 mcText，在舞台上创建静态文本“光束成字效果”，如图 8.32 所示。

图 8.32　影片剪辑 mcText

注意：影片剪辑 mcText 中的文本位置在动画制作中非常重要。在“信息”面板中可以获知文本的宽和高，调整文本的位置使文本左端的中点置于舞台的十字线处。

（3）创建影片剪辑 mcRect，在舞台上绘制黑白线性渐变的矩形填充，如图 8.33 所示。同样需要利用“信息”面板调整填充矩形的尺寸（略大于文本的尺寸）及位置（左端的中点置于舞台的十字线处）。

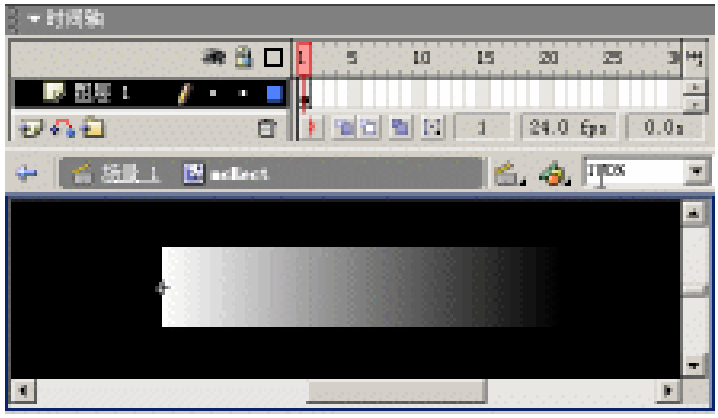

图 8.33 影片剪辑 mcRect

（4）创建影片剪辑 mcLight，更改图层名称为“渐变条”，将“库”面板中的影片剪辑 mcRect 拖放到舞台上，并定义实例名称为 rect；添加图层“文字”，将“库”面板中的影片剪辑 mcText 拖放到舞台上，并定义实例名称为 text；右击“文字”图层，选择弹出菜单中的“遮罩层”命令创建遮罩效果，如图 8.34 所示。

（5）复制“库”面板中的影片剪辑 mcLight 得到影片剪辑 mcMask 并进行修改，更改“渐变条”图层名称为“色块”，选中“色块”图层中的影片剪辑 mcRect 实例，调整实例的色调为白色，如图 8.35 所示。

图 8.34 影片剪辑 mcLight

图 8.35　影片剪辑 mcMask

8.5.2　元件布置

（6）切换到场景编辑状态，更改图层名称为“元件”，将“库”面板中的影片剪辑 mcMask **拖放到舞台上，并在“属性”面板中定义实例名称为** mask**。**

（7）将“库”面板中的影片剪辑 mcLight **拖放到舞台上，并在“属性”面板中定义实例名称为** light**；按快捷键** Ctrl+T **调用“变形”面板，在垂直倾斜文本框中输入数值**−60 **度。**

8.5.3　脚本编写

（8）添加图层“脚本”，在该图层中设置帧脚本内容如下：

```
01  xPos = 150;
02  yPos = 320;
03  move = 1;
04  _root.onLoad = function() {
05    setProperty(light, _x, xPos+1);
06    setProperty(light, _y, yPos);
07    setProperty(light, _xscale, 40000);
08    setProperty(light.rect, _width, 1);
09    setProperty(mask, _x, xPos);
10    setProperty(mask, _y, yPos);
11    setProperty(mask.rect, _xscale, -100);
12  };
13  _root.onEnterFrame = function() {
14    if (light._x < light.text._width+160) {
15      setProperty(light, _x, light._x+move);
16      setProperty(light.text, _x, light.text._x-move);
17      setProperty(mask.rect, _x, mask.rect._x+move);
18    }
19  };
```

提示：第 01～03 **行：变量的声明，其中变量** xPos、yPos **用于控制影片剪辑实例** mask **和** light **的坐标位置，变量** move **用于控制光束成字显示的速度；**

第 04～12 行：当动画开始载入（onLoad）时，利用 setProperty 控制影片剪辑实例 mask 和 light 的位置，并调整这两个影片剪辑实例中的矩形宽度。
第 13～19 行：当动画正在播放（onEnterFrame）时，利用 setProperty 不断调整影片剪辑实例 mask 和 light 中矩形及文本的相对位置。

最终的时间轴设置、舞台布置以及影片剪辑实例 light 的倾斜设置如图 8.36 所示。

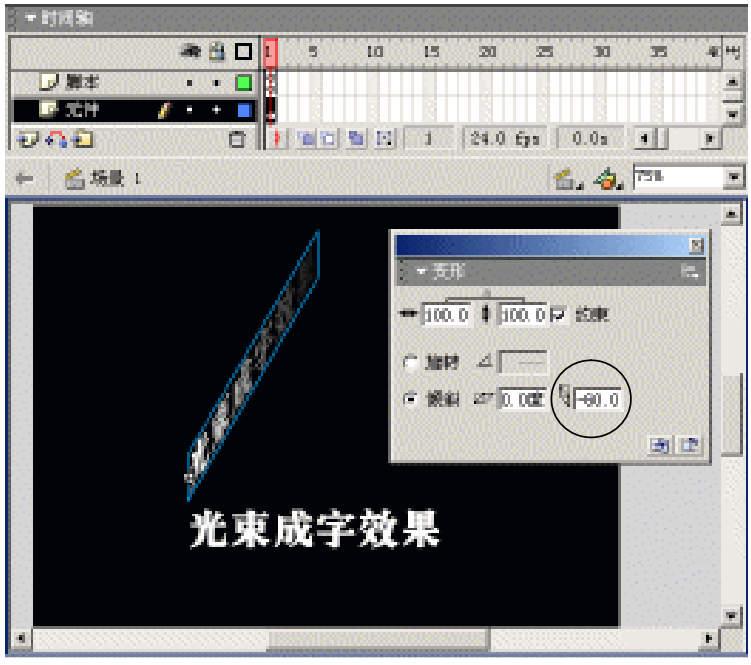

图 8.36　最终的时间轴设置

至此，光束成字效果的动画就全部制作完毕，按快捷键 Ctrl+E 测试动画，效果不错吧！

8.6　范例：模拟打字

有些动画制作软件只要通过简单的命令操作就可以实现模拟打字的动画效果。但是在 Flash MX 中并没有提供这种现成的、简便的命令操作。可以利用脚本编写方法，来实现这种动画效果。

动画播放时，伴随着短促的激光声逐个呈现《大话西游》中周星驰的那段经典对白，当所有文字都显示后自动切换到海中冲浪的画面，如图 8.37 所示。

打字效果的制作主要是截取字符串的字符进行显示，Flash 提供了两种截取字符串中字符的方法，那就是函数 mbsubstring 和方法 substr。

其中函数 mbsubstring 使用的语法结构如下：

```
mbsubstring( 源字符串, 起始字符的编号, 需要截取的字符数)
```

图 8.37　效果预览

利用该函数截取“源字符串”中的一定字符赋值给动画中的动态文本即可显示文字内容，当“需要截取的字符数”发生变化时，动态文本中的文字也相应发生变化，这就形成了打字效果。

在 Flash MX 中，函数 mbsubstring 已经不再推荐使用，取而代之的是方法 substr。其实两者的使用基本相似，只是参数的位置分布有所不同。substr 方法结构如下：

```
源字符串.substr(起始字符的编号，需要截取的字符数)
```

其打字效果的形成原理与函数 mbsubstring 基本相同，改变参数“需要截取的字符数”即可实现打字效果。最后再判断是否所有的字符都已经显示，以实现画面的切换，这就使用到了属性 length 获取“源字符串”的长度。

8.6.1　界面布置

（1）新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，尺寸更改为 400px×300px，背景颜色调整为黄色（#6699CC），帧频默认为 12fps。

（2）更改图层名称为“布局”，在“属性”面板中设置帧标签为 type，随后在舞台上创建具有相当区域的动态文本，并设置变量名称为 type，其中的文本内容为《大话西游》中的经典对白，如图 8.38 所示。

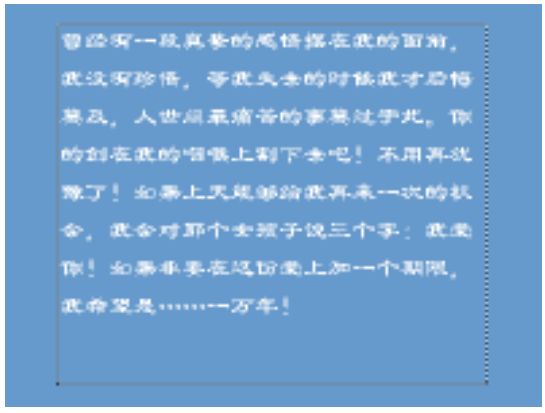

图 8.38　帧标签为 type 的舞台布置

（3）在第 6 帧处插入空白关键帧，并在“属性”面板中设置帧标签为 pic，导入 JPG 图像“冲浪”覆盖在整个舞台上，如图 8.39 所示。

图 8.39　帧标签为 pic 的舞台布置

8.6.2　脚本编写

（4）添加图层“声音”，导入 WAV 声音 Laser，在该图层中设置声音后进行适当的效果调整。

（5）添加图层“脚本”，在该图层的第 1、4 帧处分别插入空白关键帧并设置帧脚本。完成整个动画制作的最终时间轴分布如图 8.40 所示。

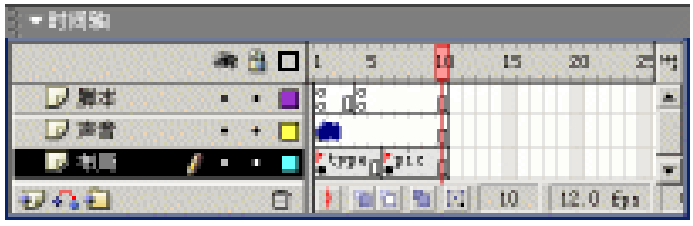

图 8.40　最终完成的时间轴显示

第 1 帧设置的脚本内容如下：

```
01  if (count==null) {
02    count = 0;
03    text = type;
04  }
05  type = text.substr( 0, count );
06  if (count>text.length) {
07    gotoAndStop("pic");
08  }
09  count = count+1;
```

提示：第 01～04 行：利用 if 语句判断计数变量 count，由于动画运动时并没有该变量的声明和赋值，所以会执行其中的赋值语句（但只执行一次）；
第 05 行 :截取字符串变量 text 的前 count 个字符并显示在变量文本为 type 的动态文本中。由于计数变量 count 的递增，所以就形成了打字效果。
第 06～08 行：将计数变量 count 与字符串变量 text 的长度进行比较，决定是否跳转停止在帧标签为 pic 的关键帧处。

第 4 帧中设置的脚本内容如下：

```
gotoAndPlay("type");
```

提示：当计数变量 count 的值小于字符串变量 text 的长度值时，动画指针就不会发生跳转。依次播放至第 4 帧处，将会促使播放指针跳转到帧标签为 type 的关键帧处（即第 1 帧）继续播放。

至此，模拟打字效果的动画制作就全部完成了，按快捷键 Ctrl+Enter 测试动画，效果不错吧！

8.7　小结和练习

本章详细介绍了 Flash 动画中的 3 种文本类型——静态文本、动态文本和输入文本，并利用静态文本进行简单排版以了解文本属性的设置。最后利用 5 个范例列举了比较经典的有关文字方面的动画效果，如幻影文字、光束成字和模拟打字等。

通过本章的学习，希望能够完成以下练习：

1．利用“ 分离 ”功能将文字完全分离为矢量图形，再将这些文字根据自己的喜好制作出极富个性化的文字外形。

2．改编范例动画“ 光束成字 ”，要求在文字显示的同时其光束会左右摆动。提示，此处不宜采用设置属性_rotation 控制影片剪辑实例的旋转，而应该利用运动动画控制光束部分的倾斜角度即可。

3．根据范例动画“ 模拟打字 ”制作一段有关个人简历介绍的打字动画效果。

第 9 章　组件应用篇

组件对于开发人员来说是一种福音，预设组件的使用不仅减少了开发设计的时间，提高了工作效率，而且还可以保证在多个 Flash 文档中的风格统一，同时也避免了动画设计人员对脚本编写的恐惧。当然，还可以根据需要开发和设计组件，以扩展组件的使用。

通过本章的学习，应该达到如下目的：

- 掌握预设组件的使用
- 了解组件的创建
- 能够开发实时预览的组件

9.1　组件的认识

组件其实就是具有参数设置的复杂的影片剪辑，这些参数可以在动画制作中随意设置。同时，组件具有一定的脚本，应用于执行参数设置和其他选项。如果您曾经接触过 Flash 的早期版本，就可以知道组件其实就是原先的智能剪辑（SmartClip）。

选择“窗口”|“组件”命令可调用“组件”面板，如图 9.1 所示。Flash MX 提供的“组件”面板中预置了 7 种组件，分别为复选框（CheckBox）、下拉列表框（ComboBox）、列表框（ListBox）、按钮（PushButton）、单选按钮（RadioButton）、滚动条（ScrollBar）以及滚动窗口（ScrollPane）等。

图 9.1　“组件”面板

通过鼠标拖曳的简单操作可以将其中的组件添加到 Flash 文档中。选中舞台上的组件实例后，可以使用“属性”面板或“组件参数”面板了解组件的属性并设置组件的参数。

选中组件实例时的“属性”面板如图 9.2 所示，可以发现，“属性”面板右上角有“属

性”、“参数”两个标签，其“属性”选项卡中的设置与影片剪辑实例的“属性”面板完全相同，而“参数”选项卡中则显示了该组件实例可设置的参数。

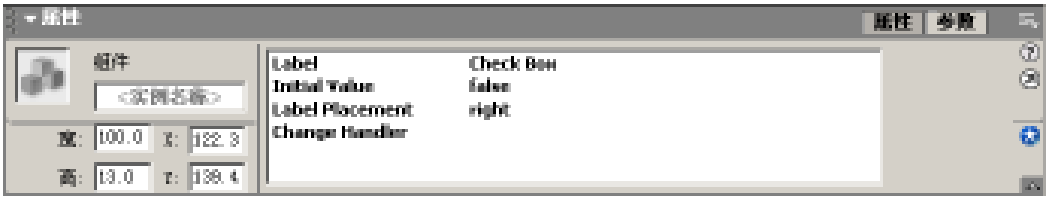

图 9.2　“属性”面板

选中组件实例后，选择“窗口”|“组件参数”命令调用“组件参数”面板，如图 9.3 所示，由该面板中可以获知该组件的可设置参数，同时可在该面板中进行参数设置。其中的组件参数会因为所选组件实例的不同而不同。

图 9.3　“组件参数”面板

您可以尝试着对组件参数进行设置，从组件的变化中了解各个参数的意义所在。如果您要了解各个参数更深层次的内容，可以求助于 Flash MX 提供的帮助文件。在下一节中，就来探讨这些预设组件的使用。

9.2　范例：信息调查

Flash MX 提供了一系列的实用的组件，方便用户创建图形化界面。本节利用 Flash MX 提供的这些组件制作了一个小型的信息调查平台，其中使用的组件有单选按钮、复选框、列表框等，如图 9.4 所示。

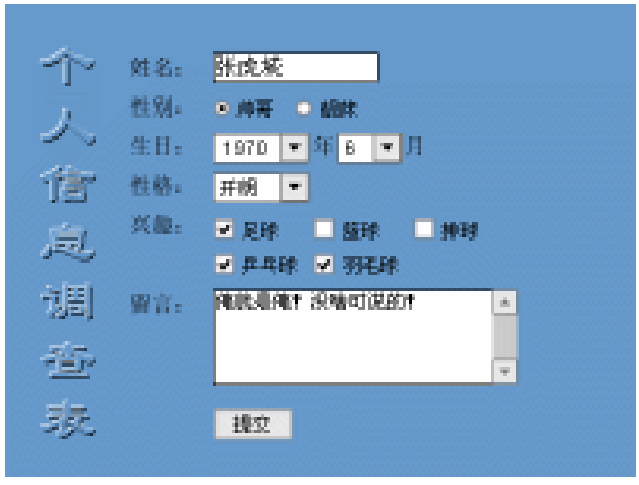

图 9.4　效果预览

9.2.1 组件布置

（1）按快捷键 Ctrl+N 新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，调整尺寸为 400px×300px、背景颜色为蔚蓝色（#6699CC）。

（2）更改图层名称为“布局”，单击第 1 帧后在“属性”面板中设置帧标签为 Write，将“组件”面板中的单选按钮、复选框、列表框和提交按钮等组件拖放到舞台上，如图 9.5 所示。下面就将各部分的布局设计进行具体讲解。

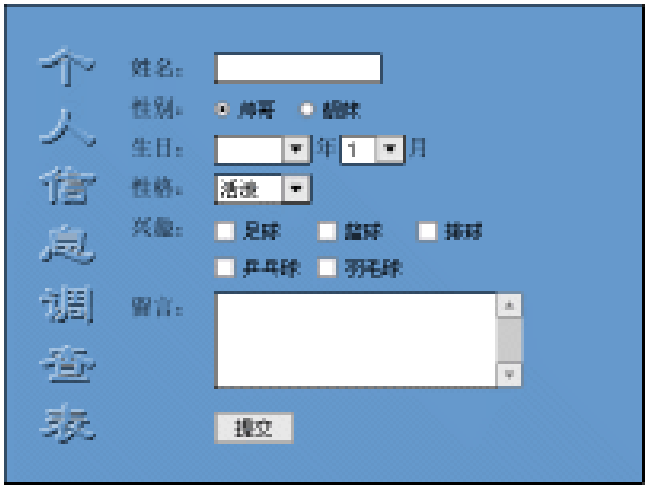

图 9.5 帧标签为 Write 的舞台布局

（3）创建静态文本“姓名：”，在其右侧创建具有一定宽度的输入文本，并在“属性”面板中定义实例名称为 Name，设置行类型为“单行”，如图 9.6 所示。

图 9.6 设置输入文本

（4）创建静态文本“性别：”，在“属性”面板中设置单选按钮组件的参数 Label 为“帅哥”、Group Name 为 sex、Data 为“帅哥”，如图 9.7 所示。同样设置另一个单选按钮的参数 Label 为“靓妹”、Group Name 为 sex、Data 为“靓妹”。

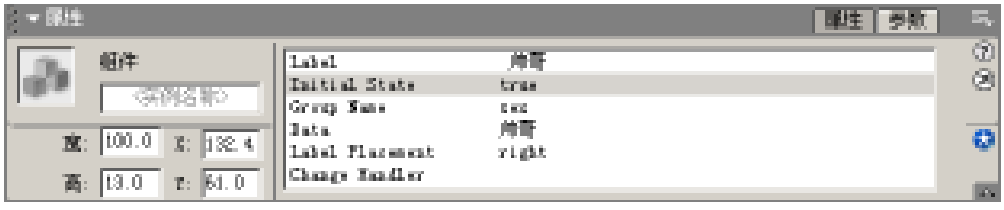

图 9.7 设置单选按钮组件

（5）创建静态文本“生日：”，在“属性”面板中设置“年”列表框组件的实例名称为 Year，双击参数 Labels 的值，在弹出的“值”对话框中单击添加按钮 + 增加数值，将数值依次设置为 1970、1971、…、1990 等，如图 9.8 所示，单击“确定”按钮确认。

图 9.8　设置参数值

由于添加的年份并没有全部包括，因此将参数 Editable 设置为 true，允许用户输入没有包含的年份，其他参数默认，如图 9.9 所示。

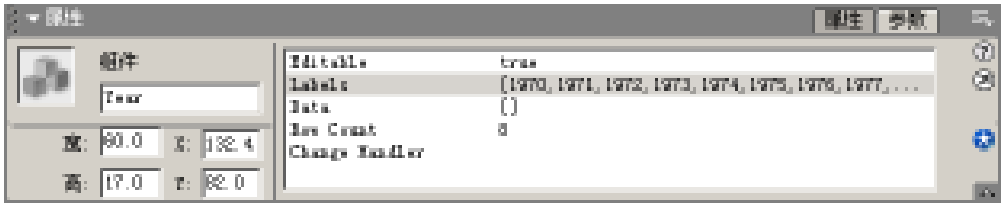

图 9.9　设置列表框组件

（6）采用相同的方法设置“月”列表框组件的实例名称为 Month，并将参数 Labels 设置为 1、2、…、12，而参数 Editable 设置为 false；设置“性格：”列表框组件的实例名称为 Kidney，参数 Labels 设置为文字“活泼”、“开朗”、“羞涩”等。

（7）创建静态文本“兴趣：”，设置复选框组件的实例名称为“Like_1”，并设置参数 Label 为“足球”，如图 9.10 所示。同样，设置其他 4 个复选框组件的实例名称为“Like_2”、…、“Like_5”，并设置参数 Label 为“篮球”、“排球”、“乒乓球”、“羽毛球”。

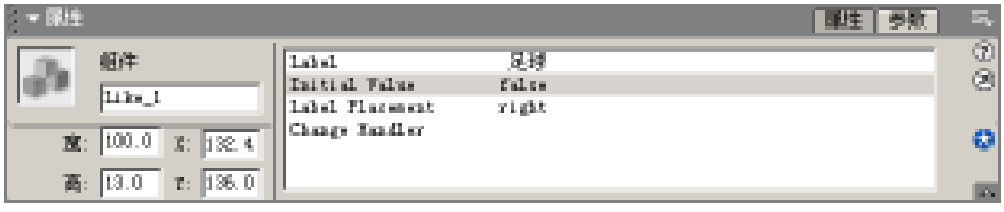

图 9.10　设置复选框组件

（8）在静态文本“留言：”右侧创建具有一定区域的输入文本，在“属性”面板中定义实例名称为 words，选择行类型为“多行”，选中“在文本周围显示边框”按钮，如图 9.11 所示。由于留言内容是由用户决定的，而创建的输入文本容纳的字数是有限的。因此，将“组件”面板中的滚动条（ScrollBar）组件拖动到舞台上输入文本 words 的右端，则该滚动条就自动吸附在其右侧。

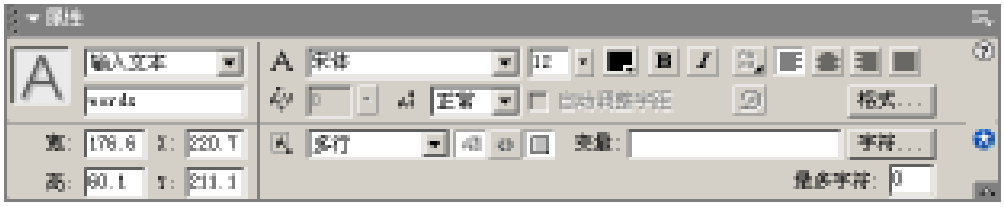

图 9.11 设置输入文本 words

（9）将“组件”面板中的 PushButton 组件拖放到舞台底端，在“属性”面板中设置参数 Label 为“提交”、Click Handler（单击时执行的函数）为 summit，如图 9.12 所示。

图 9.12 设置按钮组件——提交

（10）在第 10 帧处按 F7 键插入空白关键帧，同时在“属性”面板中设置该关键帧的帧标签为 Sure。在舞台上创建动态文本，并在“属性”面板中定义实例名称为 Result，选择行类型为“多行”，选中“在文本周围显示边框”按钮。为了使动态文本显示更多的内容，将“库”面板中的组件 ScrollBar 拖放并吸附到动态文本的右侧，如图 9.13 所示。

图 9.13 帧标签为 Sure 的舞台布局

（11）将“库”面板中的组件 PushButton 拖放到舞台底端，并在“属性”面板中设置参数 Label 为“返回”，Click Handler 为 Return，如图 9.14 所示。

图 9.14　设置按钮组件——返回

9.2.2　脚本编写

（12）添加图层“脚本”，在该图层中设置帧的脚本内容如下：

```
01  stop();
02  function Summit() {
03    for (like="", i=1; i<=5; i++) {
04      tmp = _root["Like_"+i];
05      if (tmp.getValue()==true) {
06        like = like + tmp.getLabel();
07      }
08    }
09    tmp = "姓名："+Name.text;
10    tmp = tmp+"\r 性别："+Sex.getData();
11    tmp = tmp+"\r 生日："+Year.getValue()+"年";
12    tmp = tmp+Month.getValue()+"月";
13    tmp = tmp+"\r 性格："+Kidney.getValue();
14    tmp = tmp+"\r 兴趣："+like;
15    tmp = tmp+"\r 留言："+words.text;
16    gotoAndStop("Sure");
17    result.text = tmp;
18  }
19  function Return() {
20    gotoAndStop("Write");
21  }
```

提示：第 02～18 行：自定义函数 Summit，应用于 Write 帧中的按钮，其主要功能是获取所有填写的内容组合生成字符串 tmp，最后显示在 Sure 帧中的动态文本中。

第 19～21 行：自定义函数 Return，应用于 Sure 帧中的按钮，其主要功能就是跳转到 Write 帧。

至此，有关信息调查的动画就制作完成了，按快捷键 Ctrl+Enter 测试动画。适当修改自定义函数 Summit、Return 即可进行实际应用。

9.3 范例：碰壁球体

动画界面中放置了 7 个大小不等、颜色不同的球体，如图 9.15 所示。所有球体都可以进行拖动，快速释放鼠标左键将球体扔出，那么球体就会在动画界面的四壁之间来回碰撞，同时伴随着碰撞的声音，球体组件逐渐变慢最后停止。

其实，动画界面中放置的 7 个球体都是由同一个组件生成的，使用时只要在“ 属性 ”面板中设置球体的尺寸和颜色即可。现在，就来开发和制作这种“ 碰壁球体 ”的组件吧！

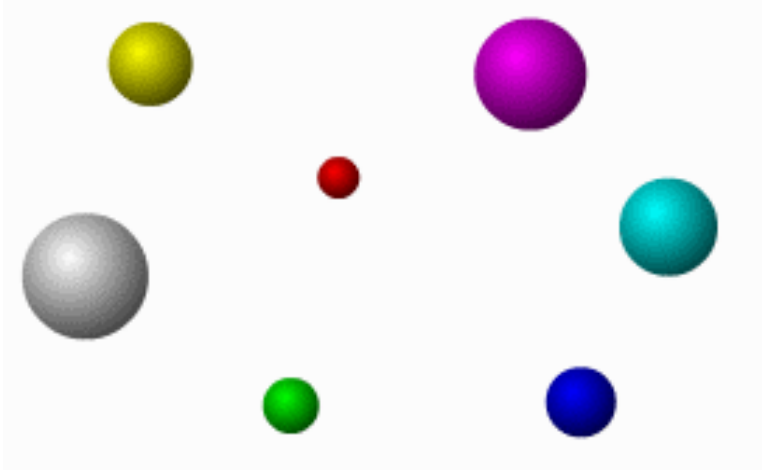

图 9.15 效果预览

9.3.1 组件创建

（1）按快捷键 Ctrl+N 新建文档，调用“ 文档属性 ”对话框设置帧频为 30fps。按快捷键 Ctrl+R 导入声音文件 sndBen.mp3，右击“ 库 ”面板中的声音 sndBen，选择弹出菜单中的“ 链接 ”命令调用“ 链接属性 ”对话框，选中链接栏中的“ 为动作脚本导出 ”复选框，并在标识符文本框中输入 sndBen，如图 9.16 所示，单击“ 确定 ”按钮确认。

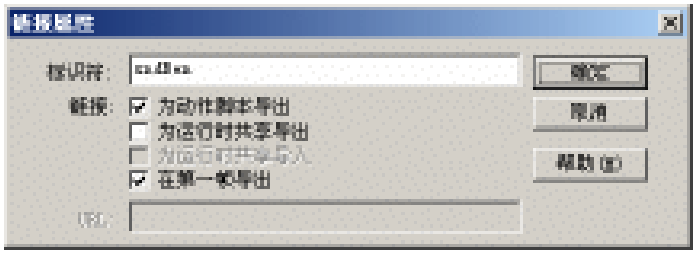

图 9.16 设置声音链接标识符

（2）按快捷键 Ctrl+F8 创建影片剪辑 mcBall，更改图层名称为“ 色块 ”，在舞台上绘制黄色的填充圆形并将之转换为影片剪辑 mcBlock，同时在“ 属性 ”面板中定义实例名称为 mcBlock，如图 9.17 所示。

图 9.17　定义实例名称

（3）复制影片剪辑实例 mcBlock，添加图层“球体”并进行原位粘贴，按快捷键 Ctrl+B 将之进行分离处理；选择颜料桶工具并在“混色器”面板中设置填充样式为“放射状”渐变，如图 9.18 所示，左侧色块为 0%的黑色、右侧色块为 75%的黑色；然后填充圆形区域，与图层“色块”重叠形成黄色球体。

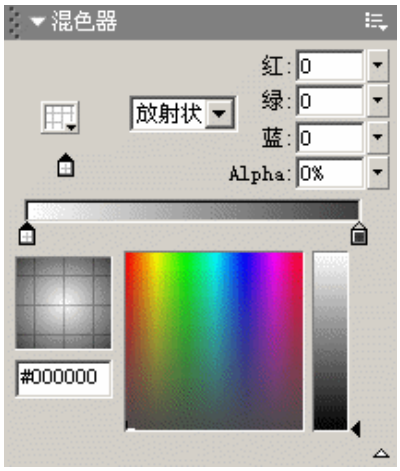

图 9.18　“混色器”面板

（4）添加图层“脚本”，并在该图层中设置帧脚本内容如下：

```
this._height = Diam;
this._width = Diam;
BallColor = new Color(mcBlock);
BallColor.setRGB(RGB);
BallSound = new Sound();
BallSound.attachSound("sndBen");
this.onPress = function() {
  this.startDrag();
  drag = 1;
};
this.onRelease = function() {
  this.stopDrag();
  drag = 0;
  xspeed = newX-oldX;
  yspeed = newY-oldY;
};
this.onReleaseOutside = this.onRelease;
```

```
18 this.onEnterFrame = function() {
19   if (drag == 1) {
20     oldX = newX;
21     oldY = newY;
22     newX = this._x;
23     newY = this._y;
24   } else {
25     this._x += xspeed;
26     this._y += yspeed;
27     xspeed *= .9;
28     yspeed *= .9;
29   }
30   if (this._x<this._width/2) {
31     xspeed = -xspeed;
32     this._x = _width/2;
33     BallSound.start();
34   }
35   if (this._x>Stage.width-this._width/2) {
36     xspeed = -xspeed;
37     this._x = Stage.width-this._width/2;
38     BallSound.start();
39   }
40   if (this._y<this._height/2) {
41     yspeed = -yspeed;
42     this._y = this._height/2;
43     BallSound.start();
44   }
45   if (this._y>Stage.height-this._height/2) {
46     yspeed = -yspeed;
47     this._y = Stage.height-this._height/2;
48     BallSound.start();
49   }
50 };
```

提示：第 01～04 行：将变量 Diam、RGB 应用到当前影片剪辑中，分别控制球体的尺寸和颜色；
第 05～06 行：利用构造函数 new 创建 Sound 对象 BallSound，并将该对象链接到“库”面板中标识符为 sndBen 的声音上；
第 19～29 行：判断球体是否处于拖曳状态，如果处于拖曳状态就对一系列变量进行赋值，否则就确定球体所处的坐标位置；
第 30～49 行：判断球体是否接触四壁，如果条件成立就改变球体的移动速度，并发出球体与墙壁的碰撞声。

影片剪辑 mcBall 制作完成后，最终的时间轴设置和舞台布置，如图 9.19 所示。

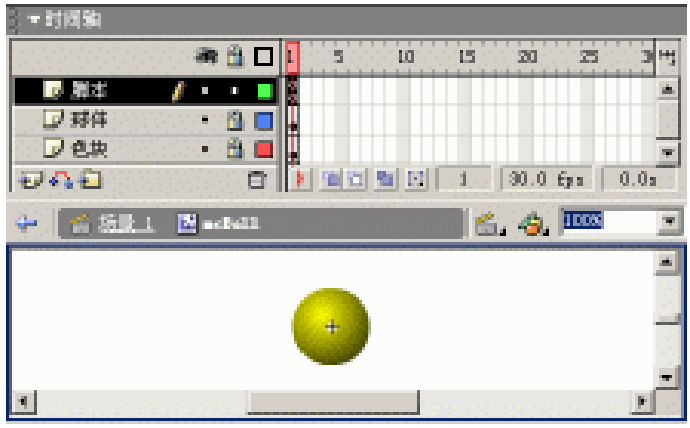

图 9.19　影片剪辑 mcBall

（5）右击“库”面板中的影片剪辑 mcBall，选择弹出菜单中的“组件定义”命令调用“组件定义”对话框，单击左上角的添加按钮 + 添加两个参数，具体的参数内容设置如图 9.20 所示，单击“确定”按钮确认。

> 提示：两个参数在变量字段中设置为 Diam、RGB，是与影片剪辑 mcBall 中脚本内容中的前 4 行代码相对应的。否则，参数设置就不会影响到组件最终的显示效果。

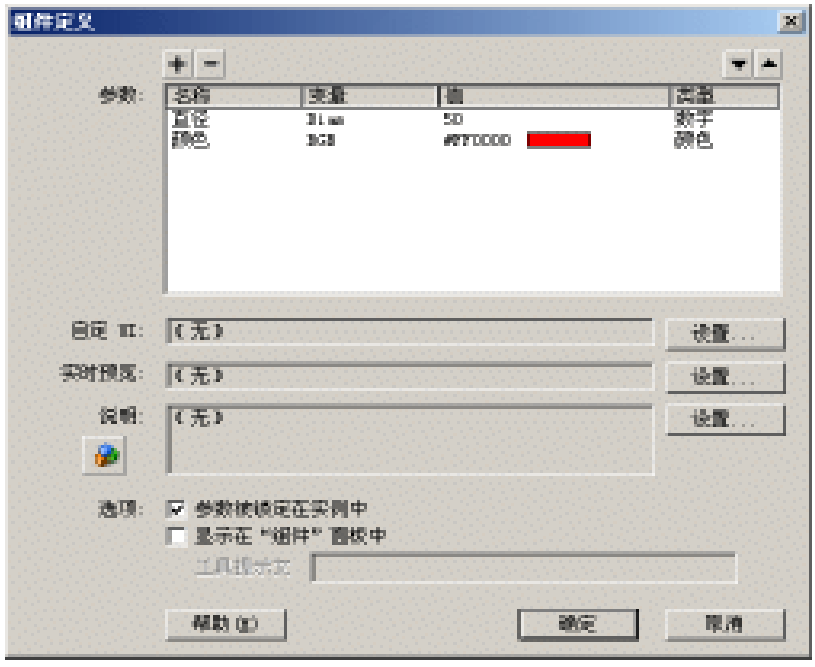

图 9.20　“组件定义”对话框

至此，“碰壁球体”的组件就制作完成了。仔细查看并比较一下“库”面板中的影片剪辑 mcBall 的图标有何变化？

9.3.2 组件布置

（6）按快捷键 Ctrl+E 切换到场景编辑状态，将“ 库 ”面板中的组件 mcBall 拖放到舞台上，复制多个并均匀地分布在舞台上，如图 9.21 所示。

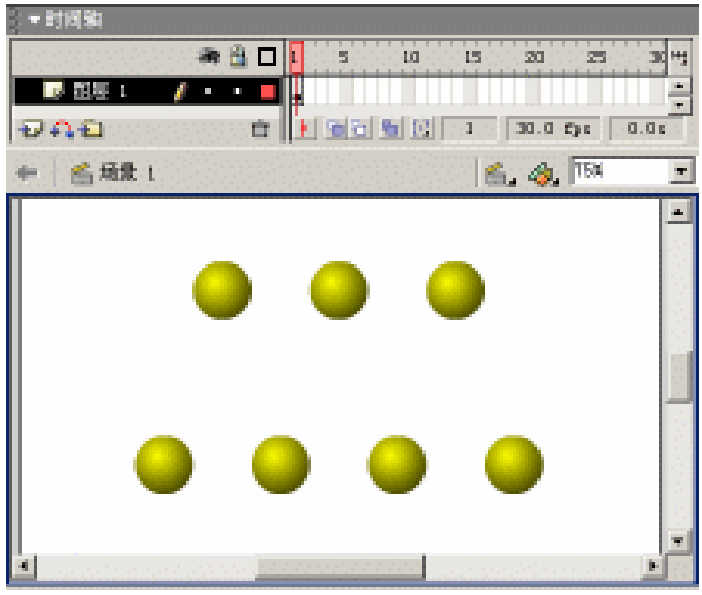

图 9.21 组件分布

（7）选中舞台上的组件实例，此时“ 属性 ”面板上显示“ 属性 ”和“ 参数 ”两个标签，在“ 参数 ”选项卡中可以设置“ 直径 ”和“ 颜色 ”，如图 9.22 所示。将各个组件实例设置不同的直径和颜色，就可以观看到大小不等、颜色不一的球体了。

图 9.22 设置组件参数

按快捷键 Ctrl+Enter 测试动画，看看效果如何？

通过本例动画的制作，应该掌握 Flash MX 中组件创建的基本方法——将“ 库 ”面板中的影片剪辑进行组件定义并设置相关参数，而且要在影片剪辑中获取这些参数并加以应用。

本例动画中球体制作方法的构思比较巧妙，利用两个图形对象的叠加显示得到球体效果——底层放置颜色随意变化的影片剪辑实例，而顶层放置相同尺寸的半透明对象。这样，通过改变底层影片剪辑实例的颜色，就可以得到另一种颜色的球体。

9.4　范例：绘制图形

仔细观察图 9.23 中所示的图形，您或许并不能发现其中的奥秘！其实，这些图形是由同一个组件的不同参数设置生成的。

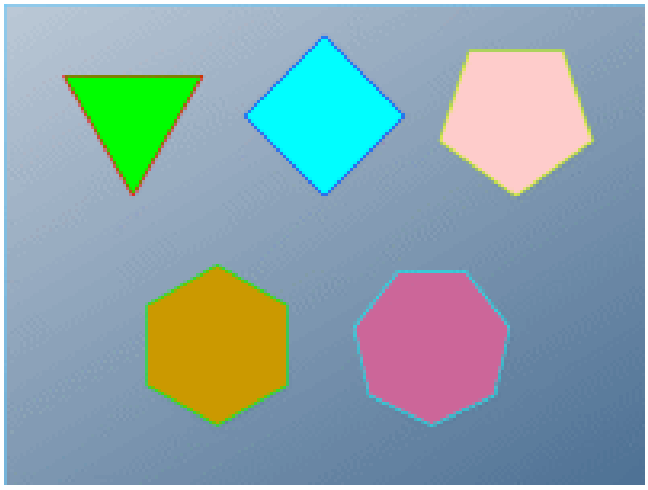

图 9.23　效果预览

与组件“碰壁球体”所不同的是，本例的组件在舞台上编辑时就可以预览最终生成的效果。下面就来开始制作这个具有“实时预览”功能的组件。

9.4.1　绘图方法

在本例中将会利用 Flash MX 提供的脚本绘图方法绘制简单的图形。Flash MX 提供脚本绘图功能，对于爱好程序编写的设计人员来说无疑是福音！

在“动作”面板中选择“对象”|“影片”|MovieClip|“绘图方法”，可以查看到所有脚本绘图的方法，关于这些方法的简要说明可以参考表 9.1。

表 9.1　绘图方法

方法	说明
beginFill	开始绘制填充
beginGradientFill	开始绘制渐变填充
Clear	移除全部的绘制指令
curveTo	以当前线型绘制弧线
endFill	终止填充
lineStyle	定义线型
lineTo	以当前线型绘制直线
moveTo	移动当前绘制点

9.4.2 组件创建

（1）按快捷键 Ctrl+N 新建文档，调用“文档属性”对话框设置尺寸为 100px×100px，如图 9.24 所示。在该文档中设置帧脚本内容如下：

```
01 function onUpdate() {
02   xPos = 50;
03   yPos = 50;
04   size = 50;
05   function Draw(n, LineRGB, FillRGB) {
06     with (this) {
07       clear();
08       lineStyle(1, LineRGB, 100);
09       beginFill(FillRGB, 100);
10       x = xPos+size*Math.sin(0);
11       y = yPos+size*Math.cos(0);
12       moveTo(x, y);
13       for (var i = 1; i<=n; i++) {
14         angle = i*2*math.pi/n;
15         x = xPos+size*math.sin(angle);
16         y = yPos+size*math.cos(angle);
17         lineTo(x, y);
18       }
19       endFill();
20     }
21   }
22   Draw(xch.n, xch.LineRGB, xch.FillRGB);
23 }
```

提示：第 01～04 行：自定义函数 onUpdate，用于实时更新组件的预览效果；第 05～21 行：自定义函数 Draw，利用 Flash MX 提供的绘画方法绘制等边多边形；第 22 行：使用自定义函数 Draw，其中的参数 n.LineRGB 和 FillRGB 会在接下来的步骤中进行具体设置。

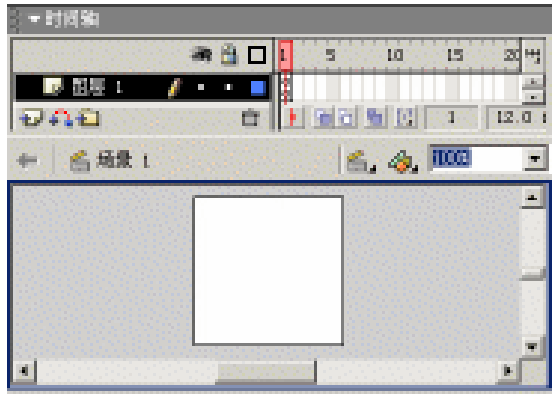

图 9.24 用于实时预览的 Flash 文档

（2）按快捷键 Ctrl+S 存储为 Flash 源文件“ 9.3.1.fla ”，并按快捷键 Ctrl+Enter 测试动画，生成动画文件“ 9.3.1.swf ”，该文件在以后的动画制作中将会使用到。

（3）按快捷键 Ctrl+N 新建文档，调用“ 文档属性 ”对话框，设置尺寸为 400px×300px。按快捷键 Ctrl+F8 创建影片剪辑 Draw，在舞台上绘制笔触颜色为 0%红色、笔触样式为“ 极细 ”、尺寸为 100px×100px 的方形线条。而且该方形线条是相对于舞台呈水平居中、垂直居中放置的。

（4）添加图层，并设置帧脚本内容如下：

```
01  size = 50;
02  function Draw(n, LineRGB, FillRGB) {
03    with (this) {
04      clear();
05      lineStyle(1, LineRGB, 100);
06      beginFill(FillRGB, 100);
07      x = size*Math.sin(0);
08      y = size*Math.cos(0);
09      moveTo(x, y);
10      for (var i = 1; i<=n; i++) {
11        var angle = i*2*math.pi/n;
12        x = size*math.sin(angle);
13        y = size*math.cos(angle);
14        lineTo(x, y);
15      }
16      endFill();
17    }
18  }
19  Draw(n, LineRGB, FillRGB);
```

提示：第 02～18 行：自定义函数 Draw，利用 Flash MX 的绘画方法绘制等边多边形；
第 19 行：使用自定义函数 Draw。

至此，影片剪辑 Draw 制作完成，最终的时间轴设置和舞台布置，如图 9.25 所示。

（5）按 F11 键调用“ 库 ”面板，右击其中的影片剪辑 Draw，选择弹出菜单中的“ 组件定义 ”命令调用“ 组件定义 ”对话框，添加 3 个参数，分别控制等边多边形的边数、线条颜色和填充颜色。具体的参数内容可以参考图 9.26 进行设置。其中“ 变量 ”字段中的 n、lineRGB 和 fillRGB 是影片剪辑 Draw 中所必须调用的。

（6）单击“ 组件定义 ”对话框中“ 实时预览 ”右侧的“ 设置 ”按钮，弹出“ 实时预览 ”对话框，选择“ .fla 文件中嵌入的.swf 文件的实时预览 ”单选按钮，并在“ 实时预览.swf 文件 ”中输入前面生成的“ 9.3.1.swf ”的具体路径（可以通过单击“ 浏览 ”按钮后选择），如图 9.27 所示。

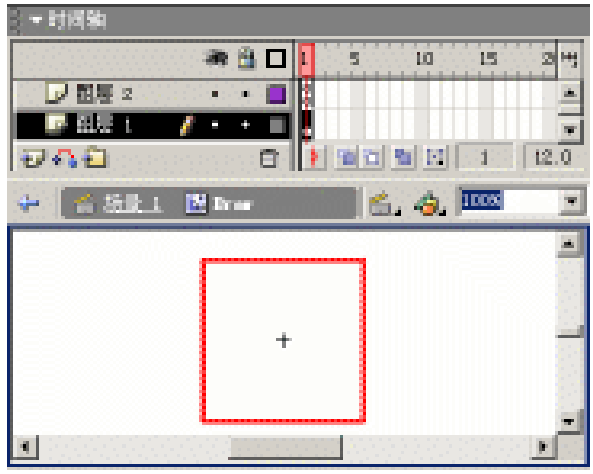

图 9.25 影片剪辑 Draw

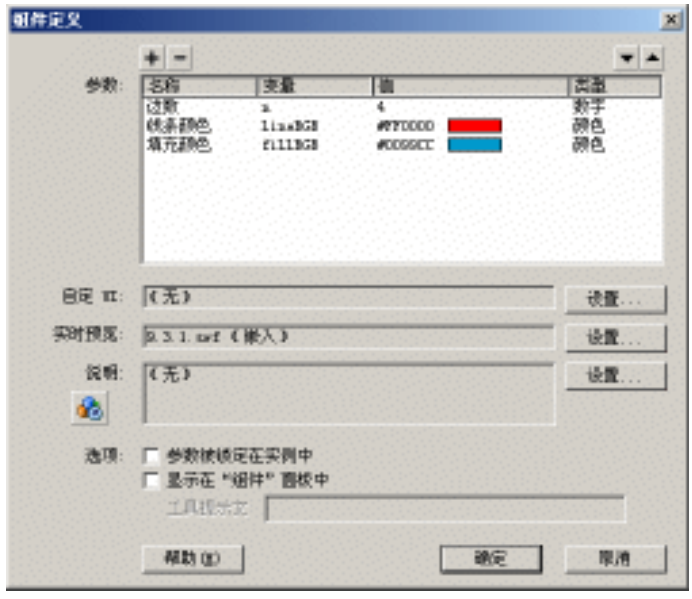

图 9.26 “组件定义”对话框

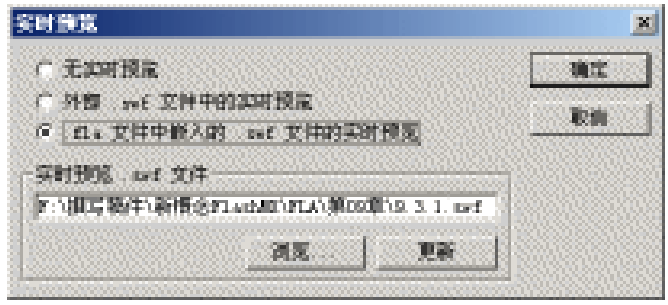

图 9.27 “实时预览”对话框

（7）连续单击“确定”按钮进行确认。至此，具有实时预览功能的组件就已经制作完成了，现在就将其应用到舞台上测试一下。

9.4.3　组件布置

（8）按快捷键 Ctrl+E 切换到场景编辑状态，更改图层名称为“背景”，绘制具有线性渐变的填充矩形，覆盖在整个舞台上。

（9）添加图层“组件”，将“库”面板中的组件 Draw 拖放到舞台上，复制生成多个组件实例。选中组件实例后，在“属性”面板中设置边数、线条颜色和填充颜色，如图 9.28 所示。此时，在舞台上的组件实例随着参数的修改会做出相应的调整。

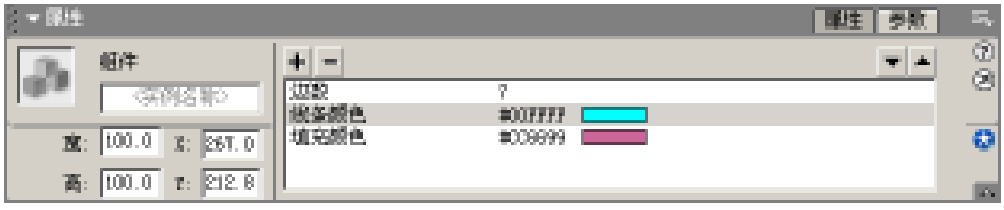

图 9.28　设置组件参数

（10）当各个组件实例设置不同的边数、线条颜色和填充颜色时，就可以在舞台上观看到不同效果的多边形，如图 9.29 所示。

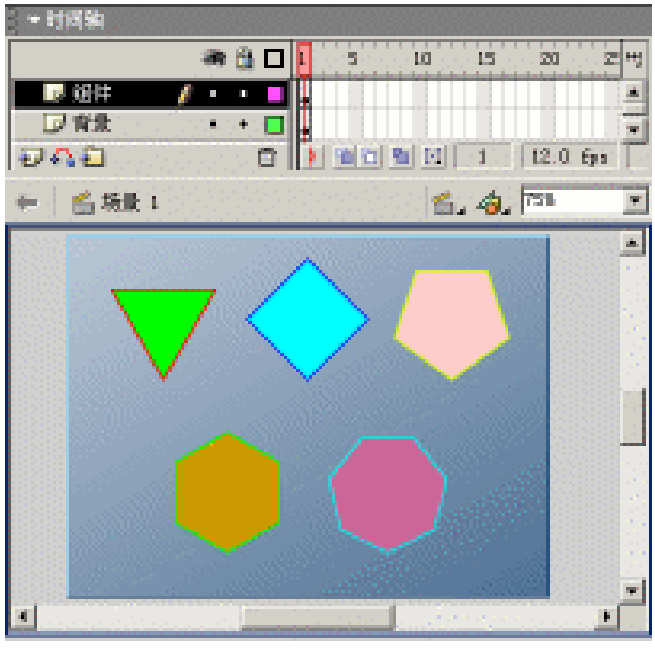

图 9.29　实时预览不同参数设置的组件

至此，具有“实时预览”功能的组件创建和应用就制作完成了，按快捷键 Ctrl+Enter 测试动画，比较一下测试时绘制的图形是否与舞台上放置的图形相同。

9.5 小结与练习

通过本章的学习，可以了解到组件给动画设计和开发人员带来的便利。预设组件的使用轻松地实现了信息调查的功能。如果没有这些预设组件的话，制作这样的信息调查将会花费大量宝贵的时间进行脚本编写等开发工作。最后，通过两个范例动画引导您设计和创建功能相对简单的组件，使您了解组件开发的整个流程。

通过本章的学习，希望能够完成以下练习：

1．根据 9.2 节中的范例动画，结合自己的工作需要或兴趣爱好，利用 Flash MX 提供的预设组件设计并制作一个信息调查表。

2．选择“窗口”|“公共库”|“按钮”命令调用“库”面板，如图 9.30 所示，打开 Component Buttons 文件夹，将其中的组件 arcade button 拖曳到 Flash 文档中，再在“属性”面板中设置组件的不同参数，可以得到颜色不同、标志有别的各种按钮，如图 9.31 所示。

图 9.30 “库”面板

图 9.31 不同参数设置的组件实例

3．如果想了解组件 arcade button 的设计和制作，可以在“Flash MX 安装目录\Samples\FLA\”文件夹中找到 Arcade_button_preview.fla、Arcade_button_ui.fla、Live_preview-custom_UI.fla 等 3 个文件，认真分析就可以大大提升 Flash 脚本编写的能力。

第 10 章　游戏设计篇

游戏是最能体现 Flash 动画交互功能的试练场了。通过在 Flash 创作环境中制作游戏，并了解游戏设计的思维过程，以及游戏过程中的各种自定义函数，可充分体会 Flash 动画中极强的交互功能。本章就以猜猜汉字、神龙曼蛇、拼图游戏和排序游戏等 4 个小游戏的设计和制作来展示 Flash 动画脚本的强大功能。

通过本章的学习，应该达到如下目的：

- 了解游戏的设计
- 创建自定义函数
- 清楚变量的传递

10.1　猜 猜 汉 字

中国汉字博大精深，只显示汉字的局部内容来猜出究竟是什么字，真的有些难度！单击“开始”按钮进入游戏开始计分，汉字上方覆盖了 8×8 共 64 块挡板，单击挡板即可揭开挡板显示其下方的局部文字，在右侧的文本框中输入您猜测的汉字，如果没有猜对将会在“剩余得分”中减掉 20 分；如果猜对的话，就会将“剩余得分”全部汇总到“合计得分”中，如图 10.1 所示。

图 10.1　效果预览

10.1.1　元件制作

（1）新建文档，按快捷键 Ctrl+J 调用“文档属性”对话框，尺寸默认为 550px×400px，背景颜色更改为浅黄色（#FFFFCC），帧频调整为 12fps。

（2）创建按钮元件 btnStart，在舞台中央绘制笔触颜色为浅灰色（#CCCCCC）、填充颜色为黑色、尺寸为 50×25 的矩形；将矩形的填充部分剪切并在“点击”帧中进行原位粘

贴；添加新层并在“弹起”帧中创建浅灰色文本“开始”；再在“指针经过”帧中将文本颜色调整为黑色，同时略微放大文本尺寸；最后保持“按下”帧中的文本颜色，恢复文本尺寸。按钮元件 btnStart 的“弹起”帧中的显示及时间轴中的关键帧分布如图 10.2 所示。

（3）按 F11 键调用“库”面板，复制其中的按钮元件 btnStart 得到按钮元件 btnHelp，并将按钮元件 btnHelp 各关键帧中的文本内容调整为“帮助”，如图 10.3 所示，用于获取游戏规则。相同的方法，复制得到按钮元件 btnTell 和 btnNext，分别用于判断猜测的文字正确与否以及猜字正确后的游戏继续。

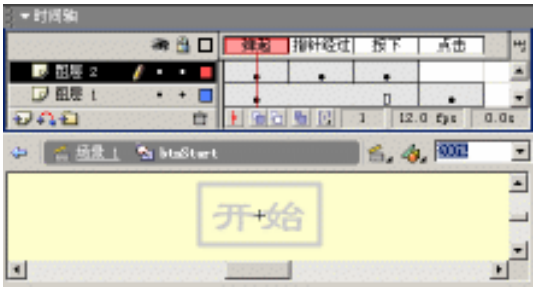

图 10.2 按钮元件 btnStart

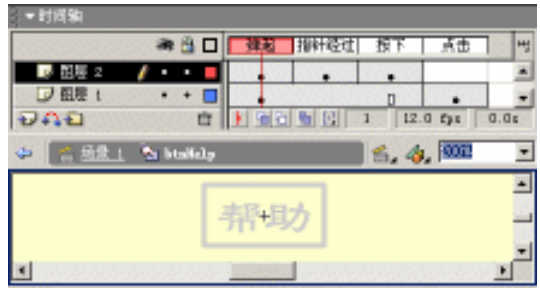

图 10.3 按钮元件 btnHelp

（4）创建影片剪辑 mcMain，在舞台中央绘制尺寸为 340px×340px 的黑色填充矩形，再利用“柔化填充边缘”命令将填充矩形向内柔化 20px，再删除中间的黑色填充矩形；选择文本工具 A 在舞台中央创建变量名称为 txt、较大字号的动态文本，如图 10.4 所示。

（5）创建影片剪辑 mcHelp，并在舞台中央创建宽度不超过 300px 的静态文本，其中的文本内容是该游戏的游戏规则和得分规则，如图 10.5 所示。

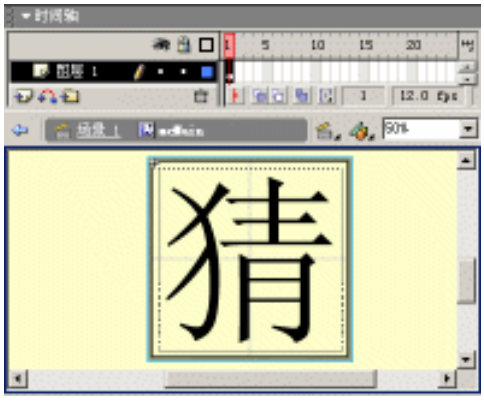

图 10.4 影片剪辑 mcMain

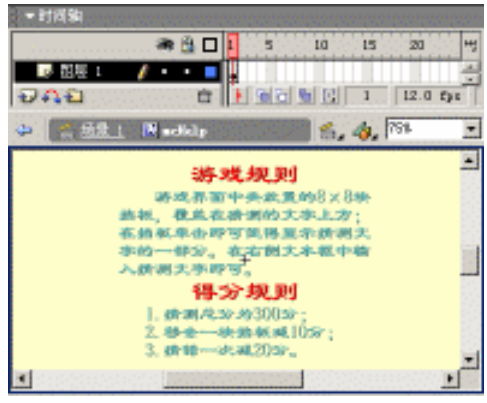

图 10.5 影片剪辑 mcHelp

（6）创建影片剪辑 mcMask，在舞台中央绘制白色线条、棕色填充、尺寸为 40px×40px 的正方形；将白色线条部分旋转 45° 并缩小至棕色填充内，如图 10.6 所示。选中全部图形后按 F8 键将之转换为按钮元件 btnMask，随后在“动作”面板中设置按钮元件 btnMask 实例的脚本内容如下：

```
on (release) {
  if (_root.remscore>=10) {
    setProperty("", _visible, 0);
    _root.remscore = _root.remscore-10;
  }
}
```

提示：单击挡板时，判断主时间轴中的“剩余得分”(remscore)，以确定是否将该挡板隐藏并将“剩余得分”递减 10 分。

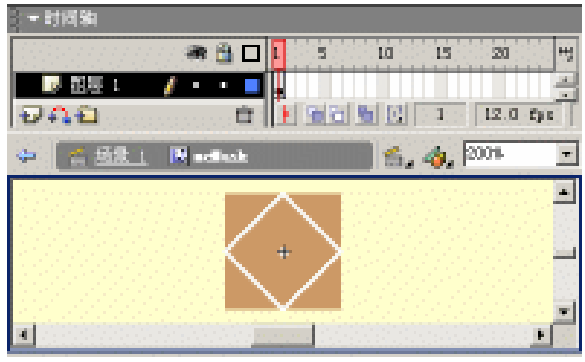

图 10.6　影片剪辑 mcMask

(7) 右击“库”面板中的影片剪辑 mcMask，选择弹出菜单中的“链接”命令，在“链接属性”对话框中选中“为动作脚本导出”复选框，并设置标识符为 mcMask。

(8) 创建影片剪辑 mcOver，在舞台中央创建静态文本“GAME OVER !”，并将之转换为图形元件 graOver，随后创建由小变大的运动动画，如图 10.7 所示。最后在运动动画的首、末关键帧中都设置帧脚本“stop();”，以防止影片剪辑 mcOver 的自动播放。

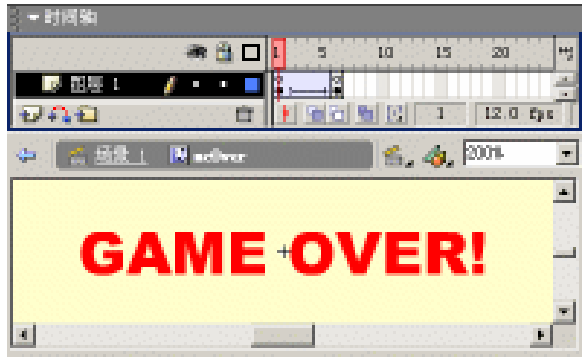

图 10.7　影片剪辑 mcOver

10.1.2　舞台布置

(9) 切换到场景编辑状态，将图层名称更改为“背景”，将舞台分成多个区域。舞台左侧部分是“库”面板中影片剪辑 mcMain 的实例，右上角放置着游戏的名称，右下角放置着游戏过程中计分的动态文本，如图 10.8 所示。其中“剩余得分”下的动态文本中显示

某轮猜字的剩余分值，变量名称为 remscore ；“ 合计得分 ”下的动态文本中显示猜字游戏的总得分，变量名称为 totscore。

图 10.8 “ 背景 ”图层

（10）添加图层“ 元件 ”，将“ 库 ”面板中的影片剪辑 mcOver、mcHelp 拖放到舞台左侧的方框内，并分别定义实例名称为 mcOver、mcHelp ；在舞台右侧的中间部分创建实例名称为 txtGuess、变量名称为 guess 并具有相当宽度的输入文本，并选中“ 在文本周围显示边框 ”按钮。

（11）将“ 库 ”面板中的按钮元件 btnStart、btnTell、btnNext、btnHelp 拖放到舞台右侧的中间部分，并分别定义实例名称为 btnStart、btnTell、btnNext、btnHelp。其中按钮 btnStart、btnTell 和 btnNext 重叠，如图 10.9 所示。

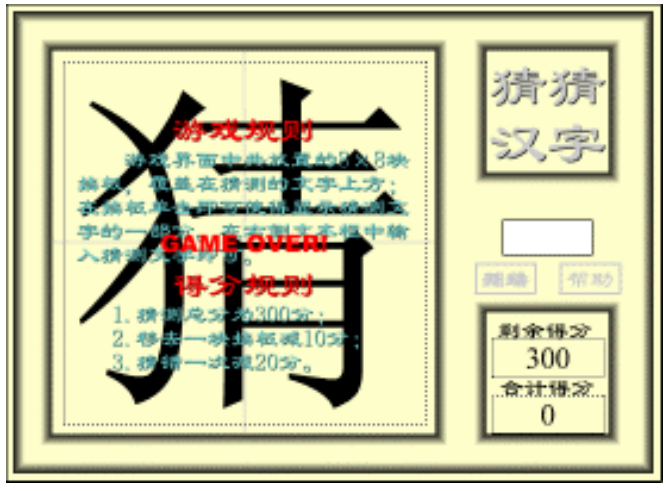

图 10.9 “ 元件 ”图层

10.1.3　脚本编写

（12）添加图层“ 脚本 ”，分别在第 5、10 帧处插入关键帧，同时在第 5、10 帧处设置帧标签为 new、tell，并在该图层的各关键帧中设置帧脚本。最后，通过插入帧将所有图层都延续到第 14 帧，如图 10.10 所示。

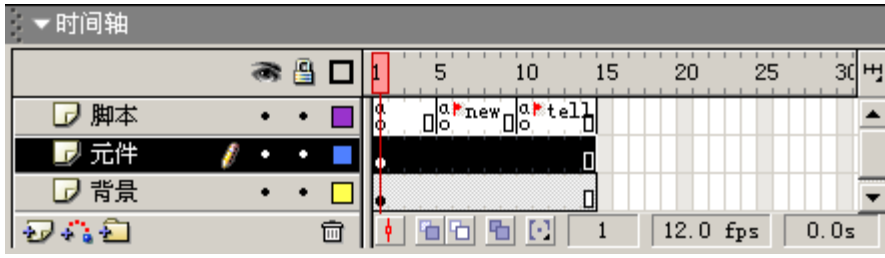

图 10.10　时间轴分布

第 1 帧中的脚本内容如下：

```
function showMask() {
  for (i = 0; i<main.num; i++) {
    for (j= 0; j<main.num; j++) {
      _root.main["mask"+i+j]._visible = 1;
    }
  }
}
function hideMask() {
  for (i = 0; i<main.num; i++) {
    for (j= 0; j<main.num; j++) {
      _root.main["mask"+i+j]._visible = 0;
    }
  }
}
stop();
mcOver._visible = 0;
mcHelp._visible = 0;
btnTell._visible = 0;
btnNext._visible = 0;
txtGuess._visible = 0;
btnStart.onRelease = function() {
  totscore = 0;
  remscore = 300;
  showMask();
  mcOver._visible = 0;
  mcOver.gotoAndStop(1);
  mcHelp._visible = 0;
  btnStart._visible = 0;
  btnTell._visible = 1;
```

```
  txtGuess._visible = 1;
  gotoAndPlay("new");
};
btnHelp.onRelease = function() {
  hideMask();
  mcOver._visible = 0;
  mcHelp._visible = 1;
  btnStart._visible = 1;
  btnNext._visible = 0;
  btnTell._visible = 0;
  main.txt = "";
};
btnTell.onRelease = function() {
  if (guess != "") {
    if (main.txt == guess) {
      hideMask();
      btnNext._visible = 1;
      btnTell._visible = 0;
      totscore = totscore+remscore;
      remscore = 0;
    } else {
      remscore = remscore-20;
      guess = "错！";
      if (remscore<0) {
        remscore = 0;
        hideMask();
        btnStart._visible = 1;
        btnTell._visible = 0;
        mcOver._visible = 1;
        mcOver.play();
      }
    }
  }
};
btnNext.onRelease = function() {
  showMask();
  btnNext._visible = 0;
  btnTell._visible = 1;
  gotoAndPlay("new");
};
```

提示：第 01～07 行：自定义函数 showMask，用于显示全部挡板以遮挡猜测的汉字；
第 08～14 行：自定义函数 hideMask，用于隐藏全部挡板以公布猜测正确的汉字；
第 21～69 行：各个按钮单击时需要执行的脚本。

第 5 帧中的脚本内容如下：

```
t0 = "你我他日吕唱晶品";
t1 = "估伸作底位体何明";
t2 = "希龟系卵别删利判";
t3 = "扶技拒批抄扮郭仙";
t4 = "坛坏址坚坐坊块皓";
t5 = "求严更束丽两来每";
t6 = "住伴佛评证识陆际";
t7 = "芬苍花苏呈吴呆员";
t = eval("t"+random(8));
temp = random(length(t)+1);
main.txt = substring(t, temp, 1);
guess = "";
remscore = 300;
```

提示：第 01～08 **行：通过一系列字符串变量存储提供游戏猜测使用的汉字；第** 09～11 **行：利用函数** random **随机截取（**substring**）字符串变量中的汉字。**

第 10 **帧中的脚本内容如下：**

```
stop();
```

至此，整个动画的界面设计和脚本编写就全部完成了，按快捷键 Ctrl+Enter **测试动画！**

10.2　神 龙 曼 蛇

神龙曼蛇（也叫贪食蛇）是一款比较流行的小型游戏。游戏过程中，通过按方向键控制神龙曼蛇的移动，吃到界面中闪动的方形食物可以长大，同时右下角的计分会叠加，如图 10.11 **所示。**

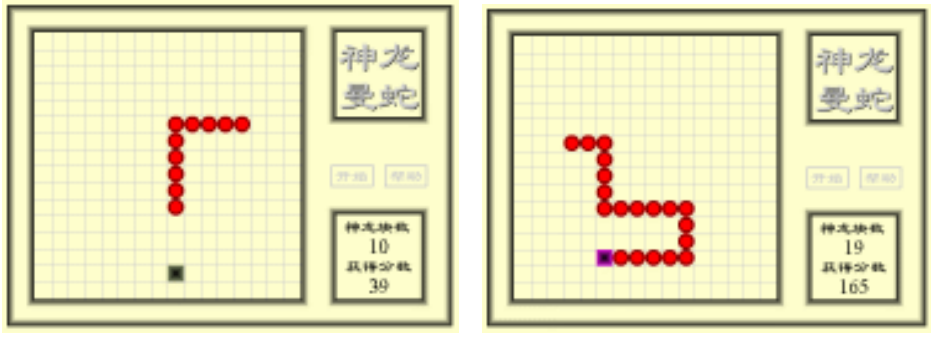

图 10.11　**效果预览**

10.2.1　元件制作

（1**）新建文档，按快捷键** Ctrl+J **调用‘ 文档属性 ’对话框，将尺寸默认为** 550px×400px**，背景颜色更改为浅黄色（**#FFFFCC**），帧频调整为** 30fps**。**

（2）创建影片剪辑 mcDragon，作为神龙曼蛇的身体部分。利用椭圆工具在舞台中央绘制红色填充、尺寸为 20px×20px 的圆形；再绘制相同尺寸、黑色填充的圆形，利用“填充边缘柔化”命令向内柔化 6px，并删除中间的黑色填充，如图 10.12 所示。

（3）创建影片剪辑 mcFood，作为该游戏中闪动的食物部分。利用矩形工具在舞台中央绘制尺寸为 20px×20px 的方形填充，分别在第 13、25 帧处插入关键帧，依次调整各关键帧中方形填充的颜色为#00FF00、#FF00FF、#00FF00，创建各关键帧之间的形状过渡；添加图层绘制相同尺寸、黑色填充的方形，利用“填充边缘柔化”命令向内柔化 20px，如图 10.13 所示，丰富色彩变化效果的层次感。

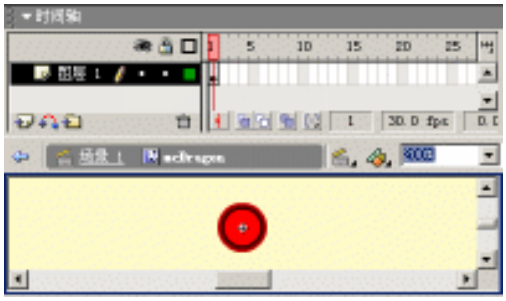

图 10.12 影片剪辑 mcDragon

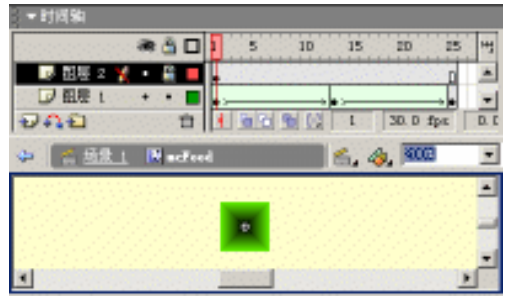

图 10.13 影片剪辑 mcFood

（4）选择“文件”|“作为库打开”命令打开“猜字游戏”的 fla 源文件，将“猜字游戏”的“库”面板中的按钮元件 btnStart、btnHelp 以及影片剪辑 mcMain、mcHelp、mcOver 拖放到本文档的“库”面板中。利用“链接”命令设置影片剪辑 mcOver 的标识符为 mcOver；修改影片剪辑 mcHelp 中的文本内容，以符合神龙曼蛇的游戏规则，如图 10.14 所示；修改影片剪辑 mcMain，删除其中的动态文本，将整个区域分割为 20px×20px 的网格，如图 10.15 所示。

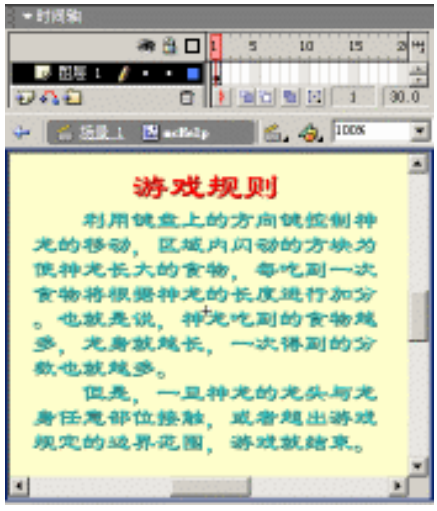

图 10.14 影片剪辑 mcHelp

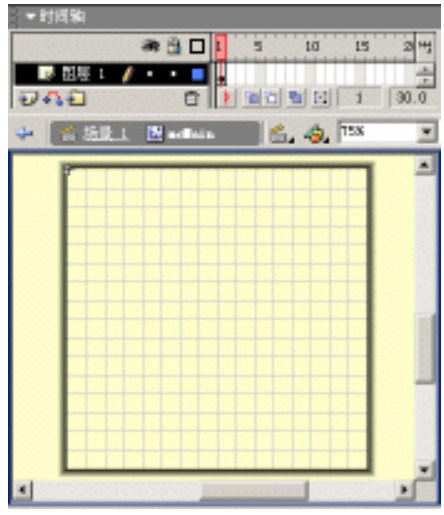

图 10.15 影片剪辑 mcMain

10.2.2　舞台布置

（5）切换到场景编辑状态，将图层名称更改为“背景”，将舞台分成多个区域。舞台左侧部分是“库”面板中影片剪辑 mcMain 的实例，右上角放置着游戏的名称，右下角放置着游戏过程中计分的动态文本，如图 10.16 所示。其中“神龙块数”下的动态文本中显示神龙曼蛇的身长，变量名称为 num；“获得分数”下的动态文本中显示游戏运行的总分，变量名称为 score。

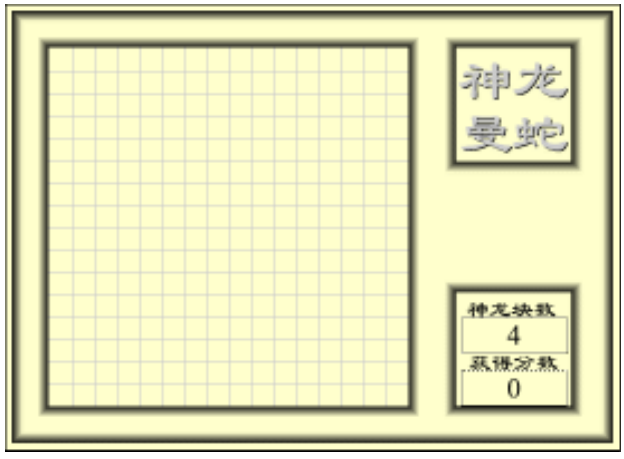

图 10.16　“背景”图层

（6）添加图层“元件”，将“库”面板中的影片剪辑 mcDragon、mcFood 拖动到舞台的左侧，并在“属性”面板中设置实例名称为 dragon、food；将“库”面板中的按钮元件 btnStart、btnHelp 拖放到舞台右侧的中间部分，如图 10.17 所示，并定义实例名称为 btnStart、btnHelp。

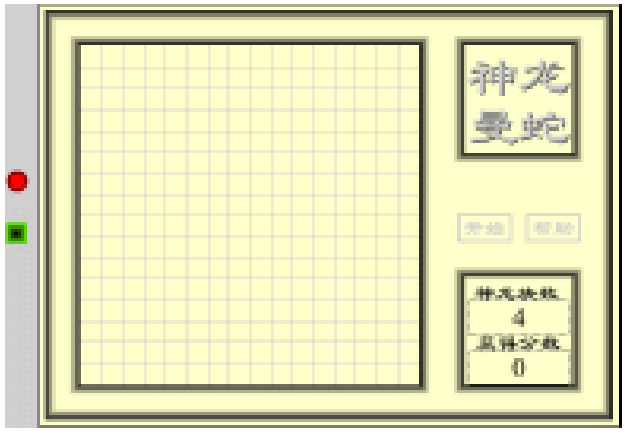

图 10.17　“元件”图层

10.2.3　脚本编写

（7）创建按钮元件 btnHit，在“点击”帧处按 F6 键插入关键帧，在舞台中央绘制任

意图形，如图 10.18 所示。该按钮元件为隐形按钮，主要用于控制神龙曼蛇的移动方向。

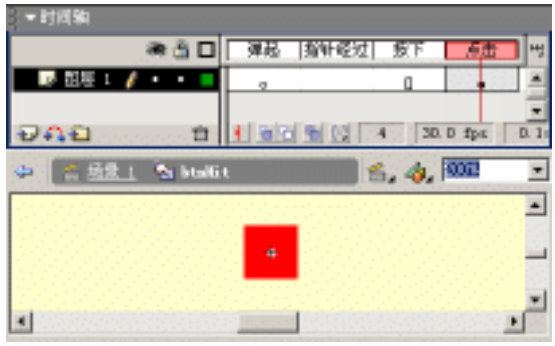

图 10.18　按钮元件 btnHit

（8）创建影片剪辑 mcControl，这是整个游戏的灵魂部分。将“库”面板中的按钮元件 btnHit 拖放到舞台中央；添加图层连续插入 4 个关键帧，并在各关键帧中设置帧脚本。影片剪辑 mcControl 的时间轴分布与舞台布置如图 10.19 所示。

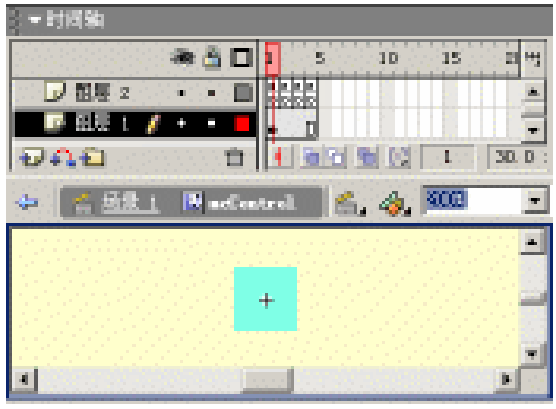

图 10.19　影片剪辑 mcControl

隐形按钮 btnHit 实例的脚本内容如下：

```
on (keyPress "<Up>") {
  if (Ym!=20) {
    Xm = 0;
    Ym = -20;
  }
}
on (keyPress "<Left>") {
  if (Xm!=20) {
    Xm = -20;
    Ym = 0;
  }
}
on (keyPress "<Down>") {
```

```
14   if (Ym!=-20) {
15     Xm = 0;
16     Ym = 20;
17   }
18 }
19 on (keyPress "<Right>") {
20   if (Xm!=-20) {
21     Xm = 20;
22     Ym = 0;
23   }
24 }
```

提示：按方向键时，隐形按钮实例会根据所按的方向键执行不同的脚本，其中变量 Xm、Ym 控制着神龙曼蛇的移动。Xm 为正时向左移动、为负时向右移动；Ym 为正时向下移动、为负时向上移动。

第 1 帧中的脚本内容如下：

```
01 function setFoodPos() {
02   Xf = minX+20*random((maxX-minX)/20)+10;
03   Yf = minY+20*random((maxY-minY)/20)+10;
04   for (i=1; i<n; i++) {
05     if (eval("Xc"+i) == Xf && eval("Yc"+i) == yf) {
06       SetFoodPos();
07     }
08   }
09 }
10 function setFood() {
11   SetFoodPos();
12   setProperty("/food", _x, Xf);
13   setProperty("/food", _y, Yf);
14 }
15 function eatFood() {
16   if (xc0+xm == Xf && yc0+ym == yf) {
17     _root.score = _root.score+n;
18     duplicateMovieClip("/dragon", "dragon"+n, n);
19     tmp = getProperty("/dragon"+(n-1), _x);
20     setProperty("/dragon"+n, _x, tmp);
21     tmp = getProperty("/dragon"+(n-1), _y);
22     setProperty("/dragon"+n, _y, tmp);
23     n = n+1;
24     _root.num = n;
25     SetFood();
26   }
27 }
28 function HideAll() {
29   for (k=0; k<=n; k++) {
```

```
30     removeMovieClip("/dragon"+k);
31   }
32   setProperty("/food", _visible, false);
33 }
34 function GameOver() {
35   for (j=1; j<n; j++) {
36     tmp = Xc0+xm==eval("Xc"+j);
37     tmp = tmp && Yc0+ym==eval("Yc"+j);
38     tmp = tmp || Xc0+xm>maxx;
39     tmp = tmp || Xc0+xm<minx;
40     tmp = tmp || Yc0+ym>maxy;
41     tmp = tmp || Yc0+ym<miny;
42     if (tmp) {
43       stop();
44       HideAll();
45       this.attachMovie("mcOver", "mcOver", 1);
46       setProperty("mcOver", _x, minX+160);
47       setProperty("mcOver", _y, minY+160);
48     }
49   }
50 }
```

提示：自定义一系列函数，函数 setFoodPos 防止放置食物时食物与神龙曼蛇的重叠；函数 setFood 用于放置食物；函数 eatFood 用于判断神龙曼蛇是否吃到食物，并在吃到食物时执行第 17～25 行的脚本；函数 HideAll 用于隐藏神龙曼蛇和食物部分；函数 GameOver 用于判断游戏是否结束，并在游戏结束时执行第 43～47 行的脚本。

第 2 帧中的脚本内容如下：

```
01 startTime  = getTimer();
02 delay  = 300;
03 minX  = _root.main._x;
04 minY  = _root.main._y;
05 maxX  = _root.main._y+320;
06 maxY  = _root.main._y+320;
07 Xm   = 20;
08 Ym   = 0;
09 n     = _root.num;
10 setProperty("/food", _visible, 1);
11 setProperty("/dragon", _visible, 0);
12 setFood();
13 for (i=0; i<n; i++) {
14   duplicateMovieClip("/dragon", "dragon"+i, i);
15   setProperty("/dragon"+i, _x, minX+150);
16   setProperty("/dragon"+i, _y, minY+150);
17   set("Xc"+i, getProperty("/dragon"+i, _x));
```

```
18    set("Yc"+i, getProperty("/dragon"+i, _y));
19  }
```

提示：第 01 行：利用函数 getTimer 获取动画已播放时间，并赋予变量 startTime；
第 02 行：初始化神龙曼蛇每次移动的时间间隔；
第 07～09 行：初始化神龙曼蛇移动量 Xm、Ym 以及身长 n；
第 13～19 行：利用 for 语句生成神龙曼蛇的最初长度。

第 3 帧中的脚本内容如下：

```
01  if (getTimer()-startTime>delay) {
02    startTime = getTimer();
03    this.Xc0 = _root.dragon0._x;
04    this.Yc0 = _root.dragon0._y;
05    _root.dragon0._x = Xc0 + Xm;
06    _root.dragon0._y = Yc0 + Ym;
07    gameOver();
08    eatFood();
09    for (i=1; i<n; i++) {
10      set("Xc"+i, getProperty("/dragon"+i, _x));
11      set("Yc"+i, getProperty("/dragon"+i, _y));
12      setProperty("/dragon"+i, _x, this["Xc"+(i-1)]);
13      setProperty("/dragon"+i, _y, this["Yc"+(i-1)]);
14    }
15  }
```

提示：第 01 行：判断当前已播放时间（getTimer）减去变量 startTime 的值是否大于神龙曼蛇每次移动的时间间隔（delay）；如果条件成立，就执行第 02～14 行的脚本；
第 03～06 行：设置变量 Xc0、Yc0 的值，更新神龙曼蛇头部的移动；
第 10～14 行：利用 for 语句循环获取神龙曼蛇身体部分各个方块的坐标（getProperty），再使神龙曼蛇身体部分前移。

第 4 帧中的脚本内容如下：

```
gotoAndPlay(_currentframe-1);
```

提示：跳转到前一帧继续播放，这样反复运行第 3 帧中的脚本，使得神龙曼蛇能够进行移动、吃食物等游戏操作。

（9）右击“库”面板中的影片剪辑 mcControl，选择弹出菜单中的“链接”命令，在“链接属性”对话框中设置标识符为 Control，同时选中“为动作脚本导出”和“在第一帧导出”复选框。

（10）切换到场景编辑状态，添加图层“脚本”，在该图层中设置帧脚本内容如下：

```
01  _root.onLoad = function() {
02    _root.attachMovie( "mcHelp", "mcHelp", 999 );
03    setProperty("mcHelp", _x, 200);
```

```
04   setProperty("mcHelp", _y, 200);
05 };
06 btnStart.onRelease = function() {
07   Control.HideAll();
08   score = 0;
09   num = 4;
10   _root.attachMovie( "Control", "Control", 999 );
11 };
12 btnHelp.onRelease = function() {
13   Control.HideAll();
14   _root.attachmovie( "mchelp", "mcHelp", 999 );
15   setProperty("mcHelp", _x, 200);
16   setProperty("mcHelp", _y, 200);
17 };
```

提示：第 01～05 **行：动画开始时显示游戏的帮助信息；**
第 06～11 **行：单击“ 开始 ”按钮（**btnStart**）后初始化游戏变量，并将控制游戏运行的影片剪辑** Control **置于主时间轴中；**
第 12～17 **行：单击“ 帮助 ”按钮（**btnHelp**）后，隐藏所有游戏中的物体，并显示游戏的帮助信息。**

至此，“ 神龙曼蛇 ”游戏就全部制作完成，按快捷键 Ctrl+Enter **测试动画。单击动画界面中的“ 开始 ”按钮后，按方向键开始游戏吧！**

10.3 拼 图 游 戏

该游戏界面有“ 初级 ”、“ 中级 ”、“ 高级 ”和“ 顶级 ” 4 **种难度级别。单击“ 开始 ”按钮后，在舞台右下角的方框内放置了切割后的多个小图片，将这些小图片拖放到左侧大方框的相应网格中，如图** 10.20 **所示，最终可以生成一幅非常精美的图像。**

图 10.20 **效果预览**

10.3.1 元件制作

（1**）新建文档，按快捷键** Ctrl+J **调用“ 文档属性 ”对话框，尺寸默认为** 550px×400px**，**

背景颜色更改为浅黄色（#FFFFCC），帧频调整为 12fps。

（2）创建影片剪辑 mcImage，这是分割为多个小图片的原始整图。为了增强拼图游戏的可玩性，所以在该影片剪辑中导入一系列精美的图像（尺寸均为 300px×300px）置于各个关键帧中，如图 10.21 所示。右击“ 库 ”面板中的影片剪辑 mcImage，选择弹出菜单中的“ 链接 ”命令后设置该影片剪辑的标识符为 mcImage。

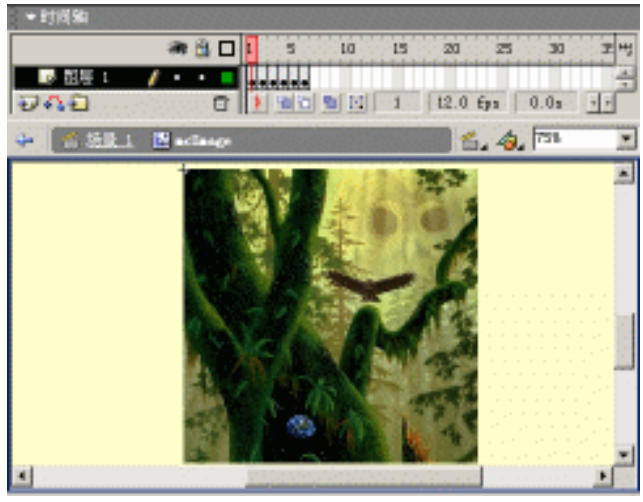

图 10.21　影片剪辑 mcImage

（3）创建影片剪辑 mcMask，更改图层名称为“ 图像 ”，将“ 库 ”面板中的影片剪辑 mcImage 拖放到舞台上，并在“ 属性 ”面板中定义实例名称为 image。添加图层“ 遮罩 ”，在舞台上绘制尺寸为 101×101 的矩形填充，矩形填充的左上角与影片剪辑的十字线完全重叠。右击“ 遮罩 ”图层，选择弹出菜单中的“ 遮罩层 ”命令创建遮罩，如图 10.22 所示。

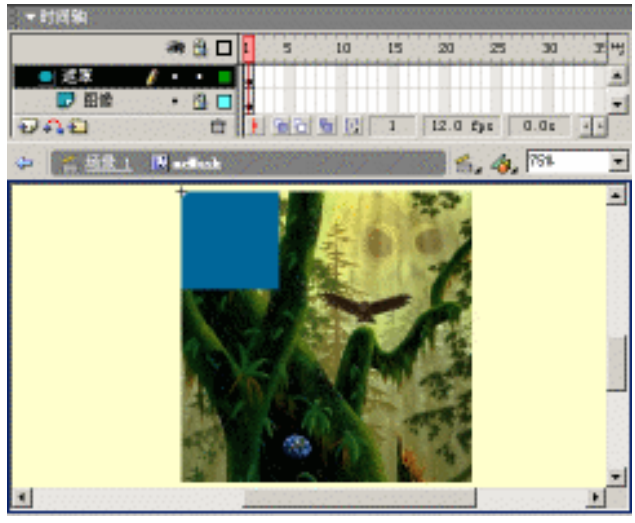

图 10.22　影片剪辑 mcMask

（4）创建按钮元件 btnHit，在“点击”帧的舞台上绘制尺寸为 100×100 的矩形填充，如图 10.23 所示。这样，隐形按钮 btnHit 就制作完成了。

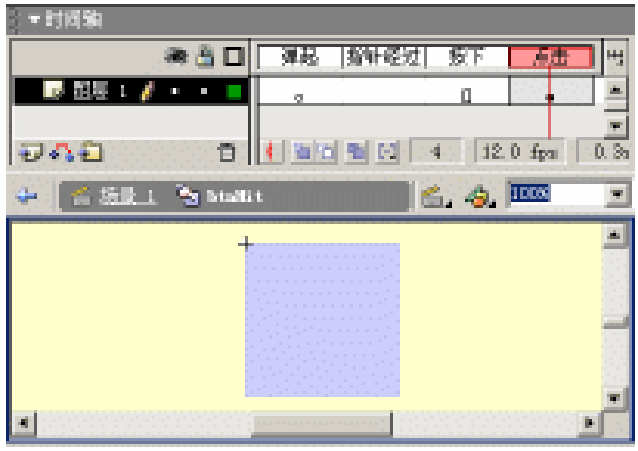

图 10.23　按钮元件 btnHit

（5）创建影片剪辑 mcSlice，将“库”面板中的影片剪辑 mcMask 拖放到舞台上，并在“属性”面板中定义实例名称为 mask。添加图层后将“库”面板中的按钮元件 btnHit 置于影片剪辑实例 mask 上，并定义实例名称为 btnHit，如图 10.24 所示，

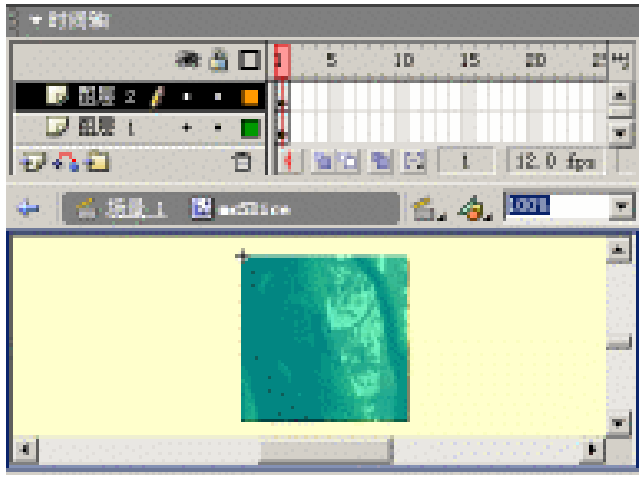

图 10.24　影片剪辑 mcSlice

（6）选择“文件”|“作为库打开”命令打开“猜字游戏”的 fla 源文件，将“猜字游戏”的“库”面板中的按钮元件 btnStart、btnHelp 以及影片剪辑 mcMain、mcHelp 拖放到本文档的“库”面板中。修改影片剪辑 mcHelp 中的文本内容，以符合拼图的游戏规则，如图 10.25 所示；修改影片剪辑 mcMain，删除其中的动态文本，并适当调整尺寸，如图 10.26 所示。

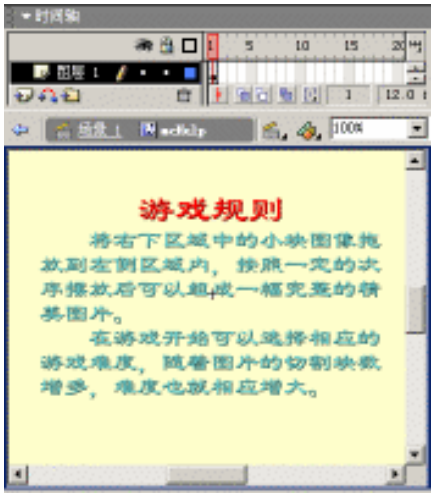

图 10.25　影片剪辑 mcHelp

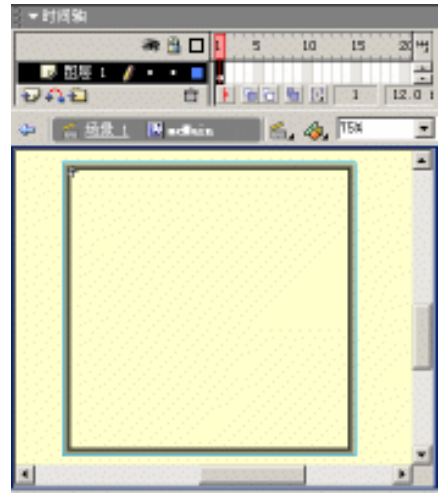

图 10.26　影片剪辑 mcMain

（7）复制“库”面板中的按钮元件 btnStart 得到按钮元件 btnReturn，修改按钮元件 btnReturn 中的文本为“返回”，如图 10.27 所示。

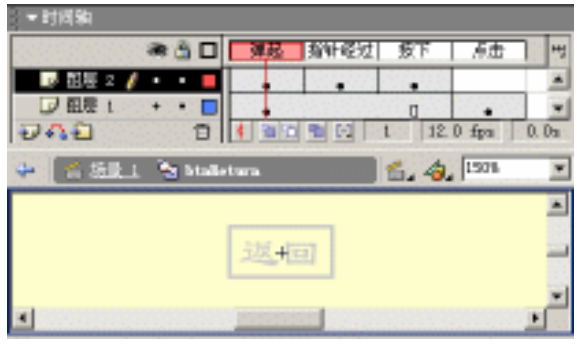

图 10.27　按钮元件 btnReturn

（8）创建影片剪辑 mcSmall，绘制尺寸为 120px×120px 的黑色填充，利用“填充边缘柔化”命令向内插入柔化后删除中间的黑色填充，如图 10.28 所示。

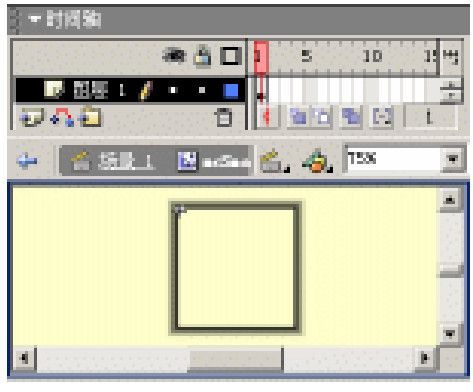

图 10.28　影片剪辑 mcSmall

（9）创建影片剪辑 mcRadio，将“组件”面板中的单选按钮 RadioButton 拖放到舞台上，设置组件参数 Label、Group Name 和 Data 如图 10.29 所示。复制多个单选按钮，并对各个单选按钮组件参数的 Label 及 Data 进行调整。最终，分别代表不同游戏难度的单选按钮分布状况如图 10.30 所示。

图 10.29 设置组件参数

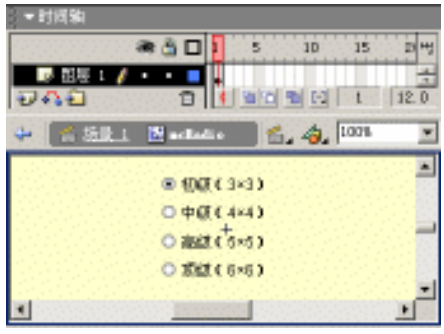

图 10.30 影片剪辑 mcRadio

（10）创建影片剪辑 mcGuide，利用线条工具在舞台中央绘制浅灰色（#CCCCCC）、笔触样式为“极细”的水平直线，并在“属性”面板中精确控制直线长度为 300px，同时直线左端点处于舞台的十字线处，如图 10.31 所示。右击“库”面板中的影片剪辑 mcGuide，选择弹出菜单中的“链接”命令，设置标识符为 mcGuide。

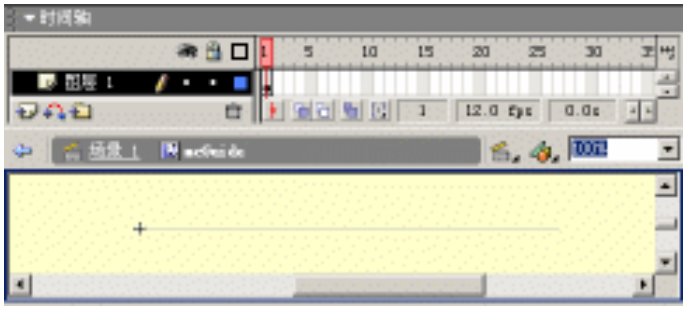

图 10.31 影片剪辑 mcGuide

10.3.2 舞台布置

（11）切换到场景编辑状态，更改图层名称为“背景”，将舞台分为多个区域。拖放“库”面板中的影片剪辑 mcMain 到舞台的左侧，并定义实例名称为 mcMain；舞台的右上角放置游戏的名称；拖放“库”面板中的影片剪辑 mcSmall 到舞台的右下角，并定义实例名称为 mcSmall。整个舞台的布局如图 10.32 所示。

（12）添加图层“元件”，将“库”面板中的影片剪辑 mcRadio、mcHelp 拖放到舞台左侧的方框中，并分别定义实例名称为 mcRadio、mcHelp；将“库”面板中的影片剪辑 mcSlice 拖放到舞台右下角的方框中，并定义实例名称为 slice；将“库”面板中的按钮元件 btnStart、btnReturn 以及 btnHelp 拖放到舞台右侧的中间，并定义实例名称为 btnStart、btnReturn、btnHelp，其中按钮 btnStart 和 btnReturn 重叠。放置所有元件后的舞台布置如图

10.33 所示。

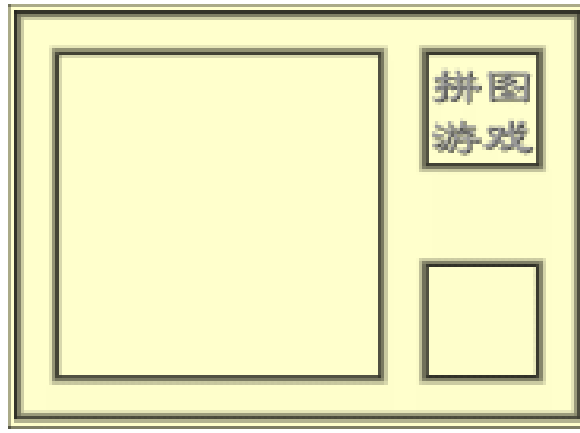

图 10.32　舞台布局

图 10.33　放置元件后的舞台布置

10.3.3　脚本编写

（13）添加图层“ 脚本 ”，在该图层中设置帧脚本如下：

```
01 function setPic() {
02   randNum = random(_root.slice.mask.image._totalframes)+1;
03   _root.mcMain.attachMovie("mcImage", "image", 1);
04   _root.mcMain.image._alpha = 10;
05   _root.mcMain.image.gotoAndStop(randNum);
06   cellsize = 300/num;
07   rate = 100/cellsize;
08   for (i=1; i<=num; i++) {
09     for (k=1; k<=num; k++) {
10       duplicateMovieClip("slice", "slice"+i+k, 100+num*i+K);
11       _root["slice"+i+k].mask.image.gotoAndStop(randNum);
```

```
      _root["slice"+i+k].mask.image._xscale = 100*rate;
      _root["slice"+i+k].mask.image._yscale = 100*rate;
      _root["slice"+i+k].mask.image._x = -100*(k-1);
      _root["slice"+i+k].mask.image._y = -100*(i-1);
      _root["slice"+i+k]._xscale = cellsize;
      _root["slice"+i+k]._yscale = cellsize;
      _root["slice"+i+k]._x = _root.mcSmall._x+random(100-cellsize);
      _root["slice"+i+k]._y = _root.mcSmall._y+random(100-cellsize);
    }
  }
  for (i=1; i<=num; i++) {
    for (k=1; k<=num; k++) {
      _root["slice"+i+k].swapDepths(101+random(num*num));
    }
  }
}
function removePic() {
  for (i=1; i<=num; i++) {
    for (k=1; k<=num; k++) {
      removeMovieClip("slice"+i+k);
    }
  }
}
function showGuide() {
  for (i=1; i<num; i++) {
    _root.attachMovie("mcGuide", "Guide"+i+"0", 10*i);
    _root["Guide"+i+"0"]._x = _root.mcMain._x;
    _root["Guide"+i+"0"]._y = _root.mcMain._y+i*cellsize;
    _root.attachMovie("mcGuide", "Guide"+"0"+i, i);
    _root["Guide"+"0"+i]._x = _root.mcMain._x+i*cellsize;
    _root["Guide"+"0"+i]._y = _root.mcMain._y;
    _root["Guide"+"0"+i]._rotation = 90;
  }
}
function hideGuide() {
  for (i=1; i<num; i++) {
    removeMovieClip("Guide"+i+"0");
    removeMovieClip("Guide"+"0"+i);
  }
}
btnStart.onRelease = function() {
  btnStart._visible = 0;
  btnReturn._visible = 1;
  mcRadio._visible = 0;
  mcHelp._visible = 0;
  num = _root.mcRadio.grpSlice.getData();
```

```
58    _root.setPic();
59    _root.showGuide();
60  };
61  btnReturn.onRelease = function() {
62    _root.mcMain.Image.removeMovieClip();
63    btnStart._visible = 1;
64    btnReturn._visible = 0;
65    mcRadio._visible = 1;
66    _root.removePic();
67    _root.hideGuide();
68  };
69  btnHelp.onRelease = function() {
70    btnStart._visible = 1;
71    btnReturn._visible = 0;
72    mcRadio._visible = 0;
73    mcHelp._visible = 1;
74    _root.removePic();
75    _root.hideGuide();
76  };
77  btnStart._visible = 1;
78  btnReturn._visible = 0;
79  mcHelp._visible = 0;
80  slice._visible = 0;
```

提示：第 01～51 **行：创建一系列自定义函数，函数** setPic **将整图进行切割并将之置于右下角的方框中；函数** removePic **用于移除游戏界面中的所有图片；函数** showGuide、hideGuide **分别用于显示网格和隐藏网格。**
第 52～76 **行：设置各个按钮实例单击时的执行脚本。**

（14**）选中舞台上的影片剪辑** mcSlice **实例后，设置该影片剪辑实例的脚本内容如下：**

```
01  onClipEvent (load) {
02    startX = _root.mcMain._x;
03    startY = _root.mcMain._y;
04    cellsize = _root.cellsize;
05  }
```

提示：影片剪辑 mcSlice **实例载入（**load**）时，初始化该影片剪辑中的变量，用于控制各个影片剪辑实例放置的位置。**

（15**）双击舞台上的影片剪辑** mcSlice **实例，进入影片剪辑** mcSlice **的编辑状态，选中按钮元件** btnHit **实例并设置该按钮的脚本内容如下：**

```
01  on (press) {
02    startDrag(this);
03    name = getProperty (this,_name);
04    i = Number( substring( name,6,1 ) );
05    k = Number( substring( name,7,1 ) );
```

```
06    this.swapDepths(999);
07  }
08  on (release) {
09    stopDrag();
10    if (_root._xmouse>startX+(k-1)*cellsize
          && _root._xmouse<startX+k*cellsize
          && _root._ymouse>startY+(i-1)*cellsize
          && _root._ymouse<startY+i*cellsize) {
11      lock = new Sound();
12      lock.attachSound("lock");
13      lock.start();
14      btnHit.enabled = false;
15      this._x = startX+(k-1)*cellsize;
16      this._y = startX+(i-1)*cellsize;
17    }
18  }
```

提示：第 01～07 行：在隐形按钮上按住鼠标左键时，拖动包含隐形按钮的影片剪辑实例，并在实例名称中提取数值赋予变量 i、k，再将该影片剪辑实例置于游戏的顶层。

第 08～18 行：在隐形按钮上释放鼠标左键时，停止拖动影片剪辑实例，判断小图片是否移动到特定位置。如果条件成立就发出声音并锁定在该位置处。

至此，“拼图游戏”就全部制作完成，按快捷键 Ctrl+Enter 测试动画。选择游戏难度后单击“开始”按钮进行游戏吧！

10.4 排 序 游 戏

游戏开始时，左侧方框中方块有序地排列着。单击其中的任一方块，就会打乱所有方块的排序，移动左侧方框中空格周围的方块，进行位置调整，如图 10.34 所示。最终排成最初的方块位置即宣告游戏胜利！单击“开始”按钮即可恢复原始状态。

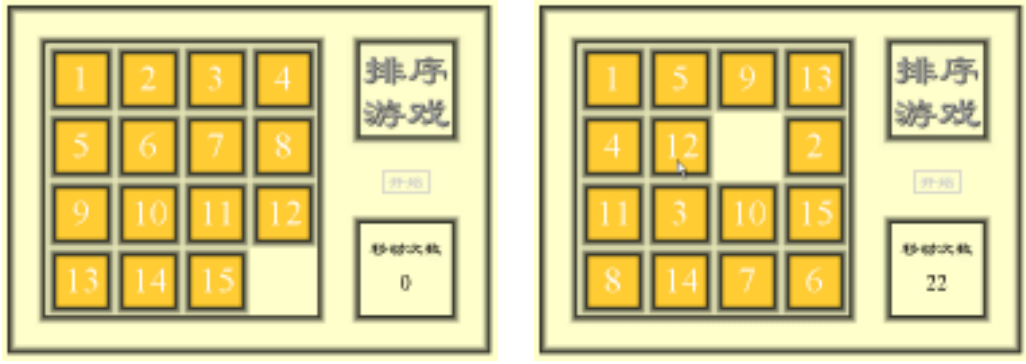

图 10.34 效果预览

10.4.1　舞台布置

（1）新建文档，按快捷键 Ctrl+J 调用“ 文档属性 ”对话框，尺寸默认为 550px×400px，背景颜色更改为浅黄色（#FFFFCC），帧频调整为 12fps。

（2）更改图层名称为“ 背景 ”，将舞台分为多个区域。舞台左侧 320px×320px 的区域是游戏的主要部分，右上角放置着游戏的名称，右下角放置着移动游戏过程中方块移动的次数，如图 10.35 所示。其中“ 移动次数 ”下动态文本的变量名称为“ clickSum ”。

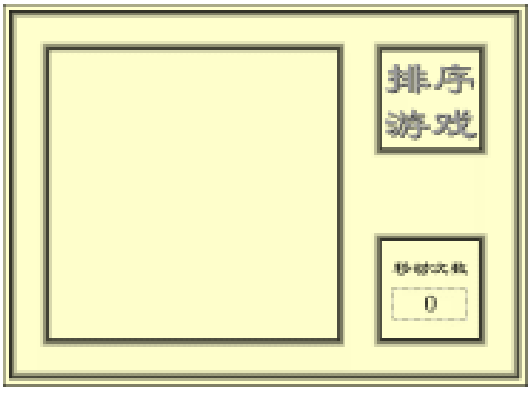

图 10.35　舞台布置

（3）添加图层“ 元件 ”，将“ 猜字游戏 ”的“ 库 ”面板中的按钮元件 btnStart 拖放到舞台右侧的中间部位，并定义实例名称为 btnStart。

（4）制作边缘柔化、尺寸为 75px×75px 的方形框；在该方形框中创建具有 2 个数字宽度的动态文本，在其中放置数值 1，并设置变量名称为 pieceNum。选中方形框和动态文本后将之转换为影片剪辑 mcBlock，并在“ 属性 ”面板中定义实例名称为 block0。放置元件后的舞台如图 10.36 所示。

图 10.36　元件分布

10.4.2　脚本编写

（5）添加图层“ 脚本 ”，在该图层中设置帧脚本如下：

```
01 xyPos = block0._x;
02 space = block0._width;
03 function pieceXY(xyNum) {
04   return xyPos+xyNum*space;
05 }
06 function Initialize() {
07   block0._x = xyPos;
08   block0._y = xyPos;
09   for (i=1; i<15; i++) {
10     block0.duplicateMovieClip("block"+i, i);
11     _root["block"+i]._x = pieceXY(i%4);
12     _root["block"+i]._y = pieceXY(int(i/4));
13     _root["block"+i].pieceNum = i+1;
14   }
15   posArray = new Array();
16   for (i=0; i<15; i++) {
17     posArray[i] = i;
18   }
19   empty = numCells-1;
20 }
21 function isWinner() {
22   for (i = 0; i<15; i++) {
23     if (posArray[i] != i) {
24       return false;
25     }
26   }
27   return true;
28 }
29 function Click(clickNum) {
30   clickNum--;
31   if (isWinner()) {
32     shuffle();
33   } else {
34     pos     = posArray[clickNum];
35     emptyRow = int(empty/4);
36     emptyCol = empty%4;
37     clickRow = int(pos/4);
38     clickCol = pos%4;
39     rowDiff  = Math.abs(clickRow-emptyRow);
40     colDiff  = Math.abs(clickCol-emptyCol);
41     if ((rowDiff+colDiff) == 1) {
42       clickSum = number(clickSum)+1;
43       _root["block"+clickNum]._x = pieceXY(emptycol);
44       _root["block"+clickNum]._y = pieceXY(emptyrow);
45       posArray[clickNum] = empty;
46       empty = pos;
```

```
47      }
48    }
49  }
50  function Shuffle() {
51    sorted = function (x, y) { if (x[1]<y[1]) {return -1;}
                                 else if (x[1]>y[1]) {return 1;}
                                 else {return 0;}
                               };
52    cell  = new Array();
53    for (i=0; i<=15; i++) {
54      cell.push([i, Math.random()]);
55    }
56    cell.sort(sorted);
57    for (i=0; i<=15; i++) {
58      piece = cell[i][0];
59      if (piece == 15) {
60        empty = i;
61      } else {
62        posArray[piece] = i;
63        _root["block"+piece]._x = pieceXY(i%4);
64        _root["block"+piece]._y = pieceXY(int(i/4));
65      }
66    }
67  }
68  Initialize();
69  btnStart.onRelease = function() {
70    clickSum = 0;
71    Initialize();
72  };
```

提示：第 03～05 行：创建带有参数的自定义函数 pieceXY，用于返回方块的放置位置。
第 06～20 行：创建自定义函数 Initialize，生成其他方块并进行初始化。
第 21～28 行：创建自定义函数 isWinner，判断是否所有方块都在相应的位置。
第 29～49 行：创建带有参数的自定义函数 Click，用于控制空格及其周围方块的位置交换。
第 50～67 行：创建自定义函数 Shuffle，用于将左侧方框中的所有方块顺序打乱。

（6）选中影片剪辑 mcBlock 实例的情况下，设置该影片剪辑实例的脚本内容如下：

```
01  onClipEvent (mouseDown) {
02    if (hitTest(_root._xmouse, _root._ymouse, false)) {
03      _root.Click(pieceNum);
04    }
05  }
```

提示：在影片剪辑 mcBlock 实例上按下鼠标左键时，判断光标位置是否处于该影片剪辑实例中。如果条件成立，就执行带有参数的自定义函数 Click。

至此，“排序游戏”就全部制作完成了，按快捷键 Ctrl+Enter 测试动画。

10.5 小结和练习

本章通过 4 个小游戏的设计和制作，比较深入地学习了 Flash MX 的脚本编写。在游戏的试练场中增加脚本编写的兴趣，可以逐步提高脚本编写的能力。

通过本章的学习，希望能够完成以下练习：

1．如何扩充“猜猜汉字”游戏中需要猜测汉字的数量？

2．怎样将“拼图游戏”中的图像全部改变为自己喜欢的图片？

3．改编“排序游戏”范例动画，设法将其中的数字方块替换为图像方块，完成排序的所有图像方块可以构成一幅完整的图像。

第 11 章　动画的发布、导出

完成 Flash 动画的设计和制作后，通常需要将 Flash 动画进行发布或导出，毕竟 Flash 动画源文件是不能直接作为素材应用的。虽然在动画制作时，按快捷键 Ctrl+Enter 测试动画就会自动生成 SWF 格式的动画（Flash 默认的导出格式），但是其中的奥秘并没有完全、透彻地被揭开。同时，Flash 动画还有多种导出格式，使得 Flash 动画技术应用在更加广泛的领域内。本章就来介绍与动画的发布、导出相关的知识。

通过本章的学习，应该达到如下目的：

- 了解动画发布的参数设置
- 区别动画导出的两种方式
- 重点掌握 SWF 动画发布
- Flash 动画应用程序的生成

11.1　动画的发布

虽然 SWF 格式文件是 Flash 动画的一种导出格式，但是通过按快捷键 Ctrl+Enter 得到的 SWF 文件是按照已定的参数设置生成的，而且 Flash 动画还有大量的导出格式。因此，对 Flash 动画作品发布等相关知识的学习是十分必要的。

11.1.1　动画发布

关于 Flash 动画的发布，通常通过选择“文件”|“发布设置”、“发布预览”、“发布”等命令开始进行的。

（1）按快捷键 Ctrl+O 打开以往创建的动画文件，选择“文件”|“发布设置”命令，相应的快捷键为 Ctrl+Shift+F12。在弹出的“发布设置”对话框中单击“格式”标签，在该选项卡中可以选择需要发布的文件格式，如图 11.1 所示。

为了较为全面地认识 Flash 动画的所有发布格式，可以将“格式”选项卡中的所有复选框都选中，此时就会依次显示各种发布格式的标签。

注意：通常，由动画发布生成的所有文件与 fla 源文件处于同一目录中。如果要控制生成文件的存储位置、文件名称，可以撤选“使用默认名称”复选框。此时，所有的“文件名”文本框都被激活，可以在该文本框中输入发布文件的文件名称与存储路径。该路径可以是绝对路径，如“C:\xcroom.swf　”；也可以是相对路径，如“..\xcroom.swf　”。

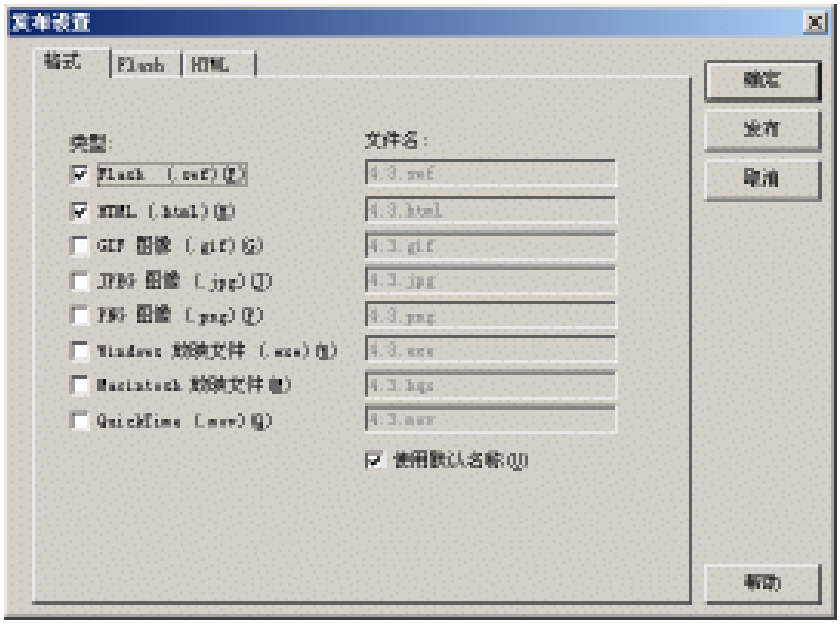

图 11.1 “发布设置”对话框——“格式”选项卡

（2）单击 Flash 标签，在 Flash 选项卡中显示了 Flash 动画发布格式 SWF 的所有参数设置，如图 11.2 所示。SWF 文件格式是 Flash 默认的发布格式，可以完全展示 Flash 动画中创建的全部效果。

图 11.2 “发布设置”对话框——Flash 选项卡

关于 Flash 选项卡中的各个参数设置的具体内容，可以参考表 11.1。

表 11.1　Flash 选项卡的参数设置

参数设置	参数介绍
版本	允许选择 Flash 播放器的版本，默认为 Flash Player 6，采用低版本的播放器将不能展示 Flash MX 新增功能部分生成的动画效果部分
加载顺序	动画中图层的加载次序，有“自下而上”和“自上至下”两种方式
生成大小报告	生成动画报告文件，其中记录了发布过程中的一系列数据
防止导入	将生成的 SWF 文件加以保护，防止他人剽窃你辛苦创作的作品。此时，“密码”文本框被激活，输入密码即可
省略跟踪动作	忽略脚本编写中的 trace 语句
允许调试	调试许可，用于检测动画中的脚本的正确与否
压缩影片	压缩动画容量，在动画中存在大量文本及脚本时尤其有效，该功能只有选中 Flash Player 6 时才支持
JPEG 品质	图像压缩控制，通常使用 70%～80%的设置，这样可以使得图像质量和文件容量能够具有较好的匹配
音频流	声音压缩控制，通常采用 MP3、8 kbps，将声音对动画容量增大的影响降为最低
音频事件	注意只有在选中“覆盖声音设置”复选框时，才能忽略在“库”面板中进行的声音设置

（3）单击 HTML 标签，在 HTML 选项卡中显示 HTML 发布格式的所有参数设置，如图 11.3 所示，可以快速地按照模板确定的方式生成 Web 页面。

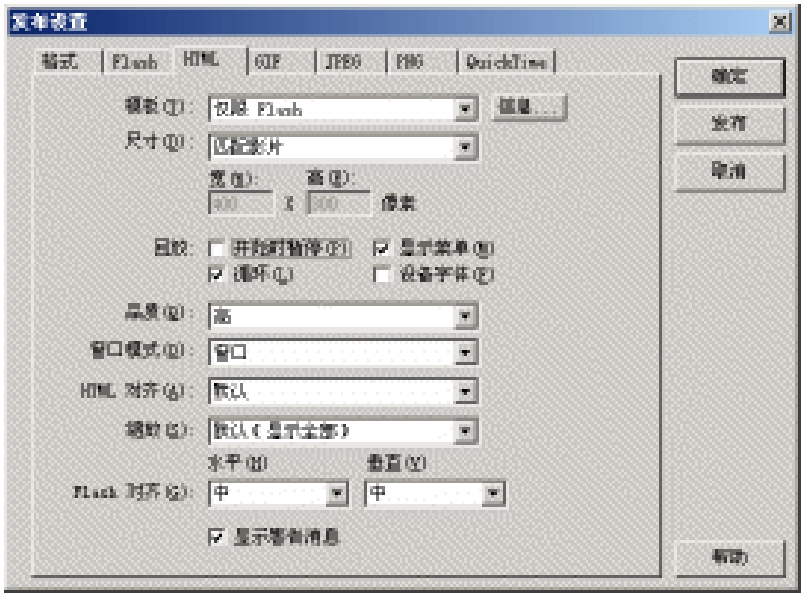

图 11.3　“发布设置”对话框——HTML 选项卡

关于 HTML 选项卡中的各个参数设置的具体内容，可以参考表 11.2。

表 11.2 HTML 选项卡的参数设置

参数设置	参数说明
模板	指定应用模板，单击右侧的“信息”按钮可获取模板的相关信息
尺寸	调整舞台尺寸，通常情况下默认为“匹配影片”
开始时暂停	动画一开始就处于暂停状态，当浏览 Web 页面时在动画区域内单击，或右击并选择弹出菜单中的“播放”命令开始播放
循环	循环播放控制，当动画播放到末帧后自动跳转到第一帧继续播放
显示菜单	快捷菜单显示，确定右击嵌入 Web 页面的动画时可以弹出快捷菜单
品质	动画播放质量，分别为“高”、“中”、“低”等选项
窗口模式	窗口模式设置，方便调整 Flash 动画在 Web 页面中的透明度及定位，但是只有安装 Flash ActiveX 控件的 IE 浏览器才可以看到相应效果
HTML 对齐	HTML 对齐方式可以确定 Flash 动画在 Web 页面中的定位方式，分别为“默认”、“左”、“右”、“顶端”、“底边”等
缩放	动画的缩放比例控制，通常采用“默认（显示全部）”，除非您需要设置 Flash 动画的高度和宽度
Flash 对齐	动画对齐方式，分为“水平”和“垂直”两部分

（4）单击 GIF 标签，在 GIF 选项卡中显示 GIF 发布格式的参数设置，如图 11.4 所示，可以生成嵌入 Web 页面中的 GIF 图像或动画。但是这就会损失掉 Flash 动画中的交互功能和声音等效果。通常，只有简单的 Flash 动画才发布为 GIF 动画。

图 11.4 “发布设置”对话框——GIF 选项卡

关于 GIF 选项卡中的各个参数设置的具体内容，可以参考表 11.3。

表 11.3　GIF 选项卡的参数设置

参数设置	说明
尺寸	调整画面尺寸，通常默认为“匹配影片”
回放	GIF 类型控制，确定发布为 GIF 图像（“静态”）还是 GIF 动画（“动画”）；当选为 GIF 动画时，还可以设置是否循环播放或重复多少次
优化颜色	优化颜色就是将 GIF 文件颜色表中不使用的颜色予以删除，可以在没有损失图像质量的情况下，使 GIF 图像相对减小 1～1.5KB，但是可能占用一定的内存。需要注意的是，如果使用 Adaptive 调色板，将会使得优化颜色命令失效
交错	图像交错显示，使得 GIF 图像在浏览器中下载的同时显示已下载的部分，通常 GIF 图像在完全下载前可以观看到基本的图像内容
平滑	平滑效果处理，消除图像边界的锯齿，从而产生高质量的位图图像
抖动实底	抖动处理既应用于单色，也应用于渐变填充。当 GIF 图像调色板中颜色不足时，多余的颜色就相应由调色板中的颜色模拟生成
删除渐变色	移除渐变，将 GIF 图像中渐变填充转换为单色，所使用的单色就是渐变填充中的第一个颜色
透明	透明处理，GIF 格式是支持透明效果的。当选择为“不透明”时，则图像背景不透明；当选择为“透明”时，则图像背景完全透明；而选择“透明度”时，可以设置透明值，介于透明与不透明之间
抖动	利用调色板中的颜色模拟调色板中没有的颜色。当选择“没有”时，关闭抖动功能，采用调色板中最接近的颜色代替调色板中没有的颜色，这可以减小文件容量，但是大大降低了颜色质量；当选择“量化”时，在尽量不增加文件容量的情况下，提供具有较高图像质量的抖动处理；当选择“扩散”时，提供最佳的抖动处理，但是会增加文件的容量，而且处理时需要较长的时间
调色板类型	图像调色板类型，通常默认为 Web 216 色，因为该调色板是浏览器安全调色板，同时产生的图像质量较好，并且处理速度快

（5）单击 PNG 标签，在 PNG 选项卡中显示 PNG 发布格式的参数设置，如图 11.5 所示，可以生成 Web 页面中的新型图像类型，但是也需要安装相应的图像插件。PNG 文件格式是 Macromedia 公司 Fireworks 图像处理软件的默认存储格式。

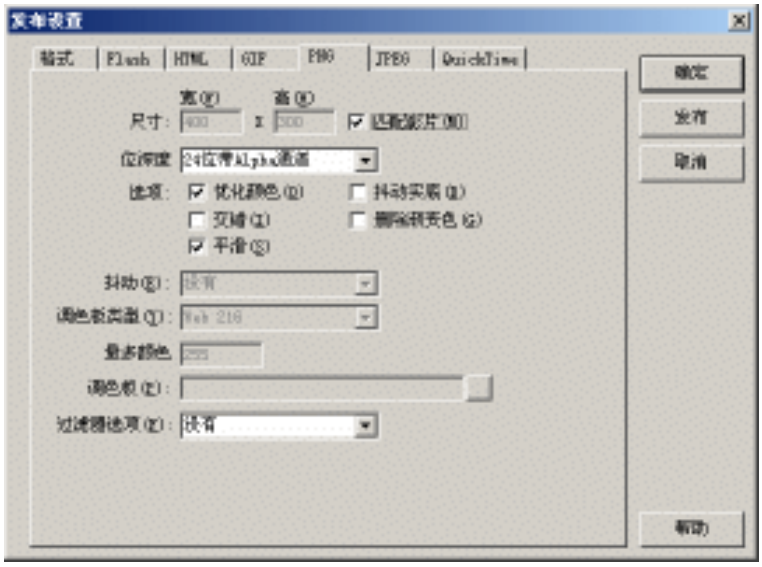

图 11.5 “发布设置”对话框——PNG 选项卡

关于 PNG 选项卡中的各个参数设置的具体内容，可以参考表 11.4。

表 11.4 PNG 选项卡的参数设置

参数设置	参数介绍
尺寸	调整舞台尺寸，通常默认为“匹配影片”
位深度	控制 PNG 图像的颜色，对于 256 色的图像，可以选择“8 位”；若要求图像颜色丰富，可以选择“24 位”；若要求图像具有丰富颜色的同时还具有 Alpha 透明，则可以选择“24 位带 Alpha 通道”。但是当 Bit 值选择越大时，生成的 PNG 文件也就越大
优化颜色	优化颜色设置就是将 PNG 文件颜色表中不使用的颜色进行删除，可以在没有损失图像质量的情况下，使得图像相对减小 1～1.5KB，但是可能占用一定的内存。需要注意的是，如果使用 Adaptive 调色板，将会使得优化颜色命令失效
交错	图像交错显示，使得 PNG 图像在浏览器中下载的同时显示已下载的部分，通常 PNG 图像在完全下载前可以观看到基本的图像内容
平滑	平滑效果处理，消除图像边界的锯齿，从而产生高质量的位图图像
抖动实底	抖动处理既应用于单色，也应用于渐变填充。当 PNG 图像调色板中颜色不足的情况下，多余的颜色就相应由调色板中的颜色模拟生成
删除渐变色	移除渐变，将 PNG 图像中渐变填充转换为单色，所使用的单色就是渐变填充中的第一个颜色
抖动	利用调色板中的颜色模拟调色板中没有的颜色，该选项只有在位深度为“8 位”才被激活。当选择“没有”时，就关闭抖动功能，采用调色板中最接近的颜色代替调色板中没有的颜色，这可以减小文件容量，但是大大降低了颜色质量；当选择“量化”时，在尽量不增加文件容量的情况下，提供具有较高图像质量的抖动处理；当选择“扩散”时，提供最佳的抖动处理，但是会增加文件的容量，而且处理时需要较长的时间

（续表）

参数设置	参数介绍
调色板	图像调色板类型，通常默认为 Web 216 色，因为该调色板是浏览器安全色板，同时产生的图像质量较好，并且处理速度快
过滤器选项	图像过滤方式，图像可以适当地压缩处理

（6）单击 JPEG 标签，在 JPEG 选项卡中显示 JPEG 发布格式的相关参数设置，如图 11.6 所示，可以生成 Web 页面中常用的 JPG 图像。

> 提示：JPG 图像、GIF 图像都是 Web 页面中较为常用的图像类型，两者在图像压缩上各有特点。前者是在保持图像颜色不损失的情况下采用高压缩比，而后者则是通过减少图像中的颜色数目进行压缩。因此，JPG 图像适用于 Web 页面中颜色丰富的照片级图像，而 GIF 图像则适合于颜色数相对较少、边界又分明的图像。

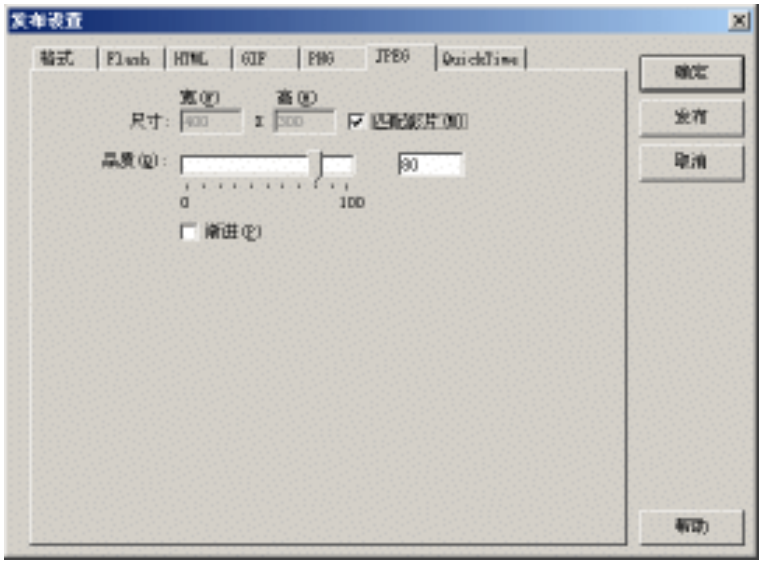

图 11.6　“发布设置”对话框——JPEG 选项卡

关于 JPEG 选项卡中的各个参数设置的具体内容，可以参考表 11.5。

表 11.5　JPEG 选项卡的参数设置

参数设置	参数介绍
尺寸	调整舞台尺寸，通常默认为“匹配影片”
品质	控制图像质量，通常取值范围处于 70%～80%之间
渐进	图像渐进显示，选中该复选框，在浏览器中逐渐显示 JPG 图像，对于网速较慢的用户而言，可以提前看到 JPG 图像的部分内容，类似于 GIF、PNG 图像中的交错显示

（7）单击“确定”按钮，返回 Flash MX 编辑状态。选择“文件”|“发布预览”命令，

此时可以看到其级联菜单中各种格式的发布命令都处于激活状态。选中其中的命令就可以导出相应格式的文件，并调用 IE 浏览器或 Flash Player 6 进行预览。

如果觉得选择“发布预览”级联菜单中的命令比较麻烦，则可以选择“文件”|“发布”命令，一次性将所有设置有发布格式的文件全部发布。

> 注意：Flash 动画进行发布时，对于静态图像 GIF、JPG、PNG 格式而言，三者之间存在略微的差别。其中发布生成的 GIF、PNG 格式图像是 Flash 动画中的第 1 帧，而 JPG 图像则是 Flash 动画中播放头所在帧的图像。

至此，对 Flash 动画进行发布的整个过程就全部完成。

11.1.2 应用程序

在“发布设置”对话框的“格式”选项卡中选择发布格式时，如果选中“Windows 放映文件”和“Macintosh 放映文件”，这两种格式并没有任何参数设置。究其原因，主要是由于放映文件是在 Flash 动画 SWF 文件格式的基础上结合播放器生成的。

浏览 Flash 动画需要得到播放器的支持，或安装有相关插件的 IE 浏览器。如果都没有，利用放映文件也可以观赏到精彩的 Flash 动画。

通过发布方式可以生成放映文件，通常在网上获得的 SWF 动画并没有提供 fla 源文件，这种情况下怎么才能得到放映文件呢？具体操作方法如下：

（1）打开 SWF 格式的 Flash 动画，当然这是在 Flash Player 中进行播放了。此时，如果 Flash 播放器中有菜单栏存在，如图 11.7 所示，那么您就很庆幸了！这是可以成功生成放映文件的必要条件！关于 Flash 播放器中菜单栏的隐藏，主要是使用了 fscommand 语句，具体内容可以参考第 5 章中的相关内容。

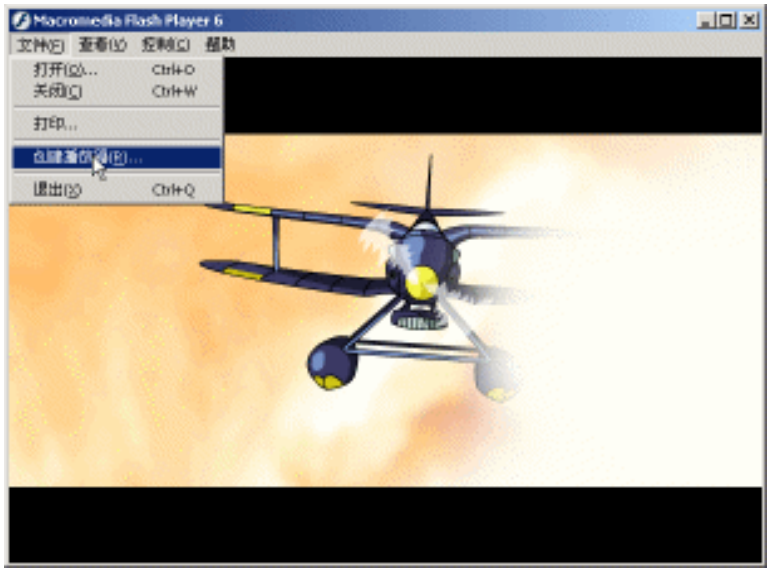

图 11.7 生成应用程序

（2）单击 Flash 播放器的“文件”|“创建播放器”命令，弹出“另存为”对话框，如图 11.8 所示。选择保存位置、设置文件名称，单击“保存”按钮即可生成该 SWF 动画的放映文件。

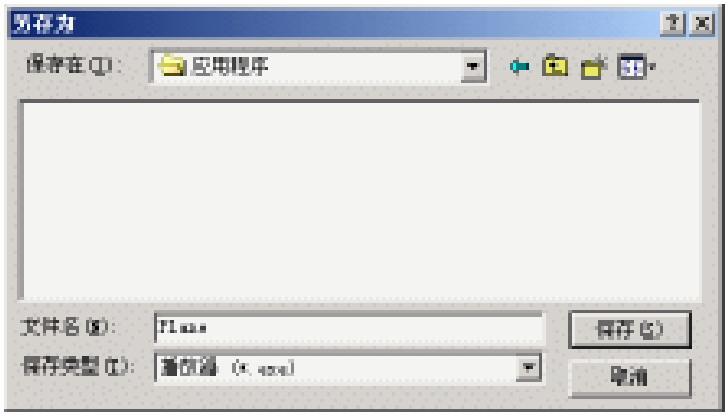

图 11.8　保存应用程序

生成的放映文件保留了 SWF 动画的所有特点，但是相对于 SWF 动画来说，文件要大得多。这主要是由于放映文件中集成了 Flash 播放器。因此，建议在浏览 Flash 动画时，不要拒绝在 IE 浏览器中安装 Flash 插件，或安装 Flash 播放器，否则会牺牲非常多的硬盘空间。

11.2　动画的导出

动画的导出与发布基本相似，两者之间的区别就在于：动画导出只是将 Flash 动画针对某种格式进行导出，一次操作只能导出一种格式；而动画发布可以将 Flash 动画同时发布为多种格式。因此，动画导出与动画发布时的参数设置是相同的。在动画导出操作中将不再对参数设置进行重复阐述，在此就动画导出操作过程中的问题加以强调。

动画的导出通常分为两种方式——导出影片和导出图像。

11.2.1　导出影片

所谓导出影片，就是将整个 Flash 动画的所有帧中的对象全部导出。具体步骤如下：

（1）按快捷键 Ctrl+O 打开以往创建的动画文件，选择“文件”|“导出影片”命令，弹出“导出影片”对话框，单击“保存类型”列表框，可以从中选择导出影片的文件格式，如图 11.9 所示。

其中格式主要有 SWF、GIF、JPG、AVI 等，SWF 格式是 Flash 动画默认的导出格式，AVI 是 Windows 操作系统的视频格式，GIF 格式也具有 GIF 动画。但是，JPG 格式完全是静态图像，怎么也可以导出呢？继续进行操作。

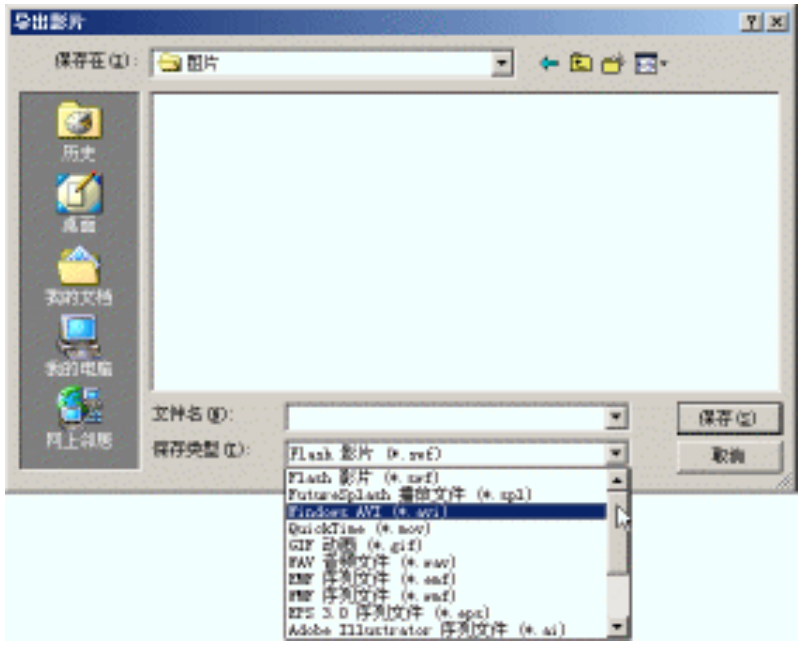

图 11.9 “导出影片”对话框

（2）在“导出影片”对话框中选择保存类型为“JPG 格式”，确定保存位置和文件名称后，单击“保存”按钮，弹出“导出 JPEG”对话框，如图 11.10 所示。其中的参数设置可以参考动画发布中的相关内容。

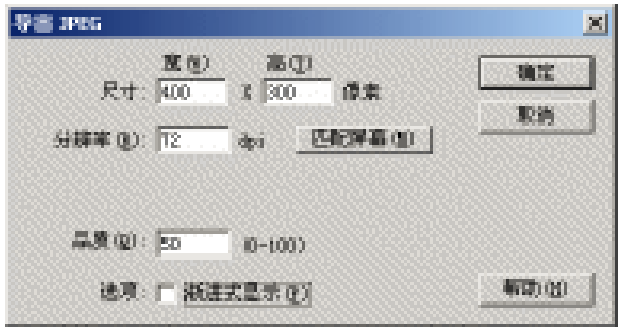

图 11.10 “导出 JPEG”对话框

（3）单击“确定”按钮就可以进行动画的导出。完成导出动画操作后，注意保存位置文件夹中的所有文件，可以发现其中保存了一系列序列形式命名的 JPG 文件。

由此可见，在使用“导出影片”命令导出静态图像 JPG、GIF、PNG 等格式时，就是将 Flash 动画的所有帧中的图形分别导出，同时将这些图像按照 Flash 动画中帧的序数进行依次命名。

11.2.2 导出图像

导出图像就是将 Flash 动画中播放头所在帧的对象进行导出。其操作步骤基本与导出影片相似，只要选择“导出图像”命令即可，此时弹出“导出图像”对话框，如图 11.11

所示。

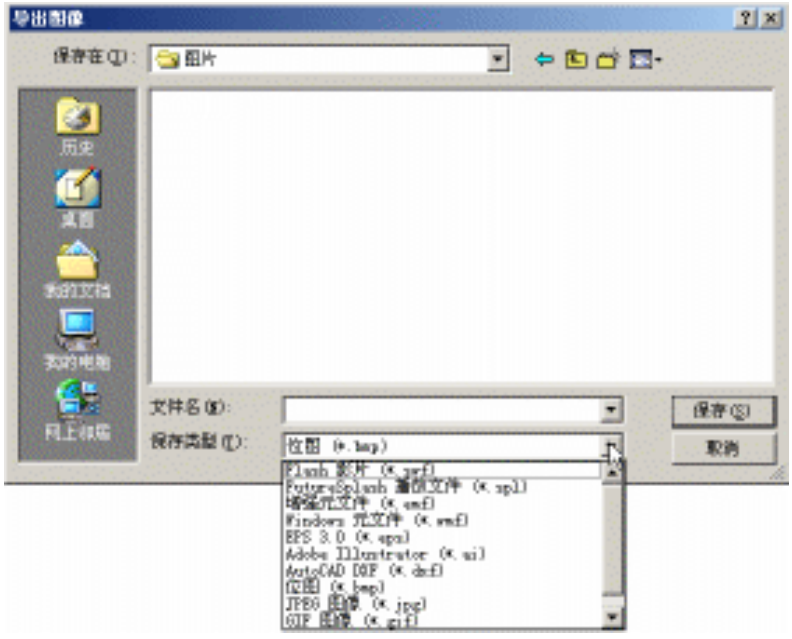

图 11.11　“ 导出图像 ” 对话框

其中保存类型同样有 SWF、GIF、JPG、AVI 等，但是需要指出的是，当导出 SWF 格式时，如果该帧中具有 Flash 的交互，则在 SWF 格式中仍保持交互功能，但是这种交互或许存在错误，因为只是将 Flash 动画中的某一帧存储为 SWF 格式。

11.3　动画的优化

虽然网络技术越来越成熟，宽带也逐渐深入家庭用户，即便如此也不该将网络资源无情地浪费。Flash 动画的优化是获取最佳播放效果的有效途径，主要表现在动画发布和导出时的参数设置。

尽管在动画发布中对 SWF 动画格式的参数设置进行了较为详细、深入的讲解，可以说 SWF 格式的参数设置是动画优化较为重要的部分。但是如果在动画创建过程中不注意适当地优化处理，则最终得到的 SWF 动画将仍旧会相当庞大，从而导致动画播放过程中发生 “ 跳帧 ” 现象，或发生 “ 停滞 ” 现象。

因此，在动画创建的过程中应该注意如下内容：

1．对于动画中出现多于一次的对象，应尽量采用动画元件，并非只有运动动画中才使用动画元件。元件是减小 Flash 动画的有效途径。

2．避免在动画文件中植入大量的位图图像，当压缩较大时会出现 “ 马赛克 ”，影响画面质量。通常位图图像只作为动画的背景出现。

3．声音在动画中通常会占有较大的比例，将声音进行大量的压缩可以极其有效地减小

Flash 动画的容量，通常采用 MP3、8 位压缩声音。

4．尽量避免大量的文本同时出现，而且尽量不要使用太多的字体，否则会影响动画的播放速度。当设置文本为嵌入字体时，选定需要的文字即可，没有必要包括全部的字符。

5．限制特殊类型线条的数量，在实际动画制作中得出的经验是实线占有内存较小，而虚线、点划线以及其他各种类型的线型则占有较大内存。

11.4 小结和练习

本章主要介绍了 Flash 动画发布和导出的各种格式，使您能根据实际要求生成所需格式的文件；同时掌握 Falsh 动画放映文件的生成，最后在动画创建过程中注意优化操作，尽量减小 Flash 动画的大小。

通过本章的学习，希望能够完成以下练习：

1．网上冲浪时，肯定会遇到些精彩的 Flash 动画，选择其中的 Flash 动画，将之转换为放映文件。

2．自创一段 Flash 动画，利用“发布”等命令将其发布为各种形式的文件，注意比较各种发布格式之间的区别。

3．分别利用“导出影片”、“导出图像”命令将 Flash 动画源文件进行发布和导出，注意两者在动画导出时的区别。