Flash MX

入门与提高

计 算 机 教 程

青苹果电子图书系列

Flash MX 入门与提高

刘　淼　王传国　王　涛　等　编著

内 容 提 要

Flash 是深受广大用户喜爱的一个网页动画制作软件，Flash MX 是该软件的最新版本。

本书从 Flash MX 的基础知识讲起，系统地介绍了 Flash MX 的操作方法和使用技巧。全书分 3 篇（基础篇、提高篇、应用篇），共 12 章，包括 Flash MX 基础、绘图、颜色的使用、简单的 Flash MX 动画、复杂的 Flash 动画、交互式 Flash 动画基础、创作交互式 Flash 动画、外部文件的导入、组件的使用、Flash MX 动画的发布、Flash MX 与网络、Flash MX 动画实战等内容。另外在附录中提供了 ActionScript 脚本语言参考和 Flash MX 快捷键，以便于读者查阅。

本书内容丰富、条理清晰、通俗易懂，适合广大学习和使用 Flash MX 的用户阅读，也可供大专院校师生学习和参考。

前　言

Flash 是由 Macromedia 公司发布的一款专用于制作网页动画的软件。使用 Flash 制作的 Flash 动画体积小，传输速度快，能够满足 WWW 网络高速传输的需要，所以 Flash 得到了众多网页设计师的青睐。可以说 Flash 是当前最优秀、应用最广泛的网页矢量动画设计软件。

Flash MX 是该软件的最新版本，该版本具有以下新特性：

● 提供了更简洁的工作界面

在 Flash MX 中面板元素大大减少。Macromedia 取消了 Instance（实体）、Frame（帧）、Effect（效果）、Sound（声音）、Stroke（线条）、Fill（填充）、Text（文本）、Paragraph（段落）和 Character（字符）等面板，并将它们都移到一个新的面版——属性面板中。这个设计来自于 Macromedia 的另一产品——Dreamweaver 中一个极受广大用户欢迎的界面特性。选择一个对象，属性面板也随着所选对象不同而改变。另外，一些开发者会非常欣赏 Action 面板的改进，尤其是在交互开发环境中新的上下相关代码提示和 ActionScript 参考。

在 Flash MX 中，帧的操作变得更加直观了，感觉上就像使用其他常见的图形处理软件。例如，鼠标双击一帧将会选择一定范围的帧，而不会打开 ActionScript 面板。同样，一些对时间轴和舞台同时产生影响的快捷键，也为了避免未知的行为而被分隔开。例如，现在，delete 键仅仅会删除舞台上的对象。

● 增强的图像绘制、修改和上色工具

Flash MX 对于自由变换工具加入了比以前更灵活的功能。用户可以很容易地对一个图片对象进行缩放、旋转、倾斜、扭曲甚至是改变每个变换的中心点或轴。这个新工具在对形状、符号和图片元素进行修改时更富有创造性和灵活性。

新的 Color Mixer 面板改进了前代的 Mixer 面板并综合了 Stroke 面板。对于做动画来说，在上色阶段可以适当提高效率。在这个面板中，可以直接修改涉及颜色的所有选项，包括制作过渡色和修改/增加自定义颜色。

● 更广泛的视频支持

动态视频支持是 Flash MX 重大的新功能之一。由于在 Flash MX 中加入了动态视频支持，开发人员可以制作新形式的交互视频应用程序，提供与 Web 视频内容天衣无缝的整合。

● 多语言支持

Flash MX 可以支持多种语言，可以支持 Unicode 字符标准，在没有更换计算机语言系统的情况下，设计者也可以使用多种语言。在多语言目录里，不需要嵌入字符，从而使文件的尺寸更小。

另外，Flash MX 还新增了组件的概念，使 Flash 动画资源的共享更为广泛、简便。在 ActionScript 脚本语言中提供了比以前版本更多的函数，可以创作出更为复杂的交互式 Flash 动画。

本书由浅入深、全面系统地介绍了 Flash MX 的新增功能、操作方法和使用技巧。本书分为三大部分，第一部分为 Flash MX 基础篇（第 1 章~第 4 章），主要介绍 Flash MX 的基本功能及操作。第二部分为 Flash MX 提高篇（第 5 章~第 9 章），主要介绍 Flash MX 的高级功能。结合每一项功能的介绍都给出相应的实例，供读者参考。第三部分为 Flash MX 应用篇（第 10 章~第 12 章），主要介绍 Flash MX 动画的实现方法和应用技巧。其中第 12 章给出了几个非常实用的例子，以说明 Flash 在不同方面的具体应用。另外，本书的附录提供了 ActionScript 脚本语言参考和 Flash MX 快捷键列表。

本书由刘淼、王传国、王涛等主编。在编写过程中，得到了王晓君、周华、彭小琦、舒伟群、徐振成等同志的大力支持；张鹏、杜建成、田雅琴等参与了本书的审校工作，在此一并表示感谢。

由于时间仓促，加之作者水平有限，书中疏漏之处在所难免，敬请广大读者批评指正。如有疑问或其他意见，请与作者联系，作者的 E-mail 信箱：onyx_lj@sina.com。

作　者

目　录

第一部分　Flash MX 入门篇

第 1 章　Flash MX 基础 ····· 2

1.1　Flash MX 概述 ····· 2
1.1.1　Flash MX 的功能 ····· 2
1.1.2　Flash MX 的新特性 ····· 2
1.2　Flash MX 的安装 ····· 2
1.3　使用 Flash MX ····· 7
1.3.1　Flash MX 的启动 ····· 7
1.3.2　友好的 Flash MX 用户界面 ····· 7
1.3.3　浏览 Flash MX 动画 ····· 9

第 2 章　绘　图 ····· 11

2.1　综　述 ····· 11
2.2　Flash MX 的绘图工具 ····· 11
2.2.1　Line（线条）工具 ····· 11
2.2.2　Lasso（套索）工具 ····· 12
2.2.3　Pen（钢笔）工具 ····· 13
2.2.4　Text（文本）工具 ····· 14
2.2.5　图形工具 ····· 17
2.2.6　Pencil（铅笔）工具 ····· 18
2.2.7　Brush（刷子）工具 ····· 19
2.2.8　Eraser（橡皮）工具 ····· 21
2.3　线条及轮廓的修改 ····· 22
2.3.1　使用箭头工具修改造型 ····· 22
2.3.2　箭头选项的设置 ····· 23
2.3.3　箭头工具修改线条的其他途径 ····· 24
2.3.4　自由变形工具 ····· 25
2.3.5　曲线的优化 ····· 25
2.4　磁铁工具的妙用 ····· 26
2.5　视图工具 ····· 27
2.5.1　Zoom（缩放）工具 ····· 27
2.5.2　Hand（手形）工具 ····· 28
2.6　绘图参数的设定 ····· 28

第 3 章　颜色的使用 ····· 30

3.1　综　述 ····· 30
3.2　描绘和填充 ····· 30
3.2.1　墨水瓶工具的使用 ····· 30
3.2.2　油漆桶工具的使用 ····· 30

3.2.3 点滴器的使用 31
3.3 颜色面板的使用 31
3.3.1 信息面板 31
3.3.2 图形的变形 32
3.3.3 神奇的填充 32
3.4 编辑调色板 35
3.4.1 调色板的使用 35
3.4.2 描绘色面板 36
3.4.3 填充色面板 37
3.4.4 调色板的导入与导出 38

第 4 章 简单的 Flash MX 动画 40

4.1 综 述 40
4.2 帧的引入 40
4.2.1 帧的概念 40
4.2.2 帧的显示模式 40
4.2.3 帧的类型 42
4.2.4 帧的编辑 43
4.3 功能强大的位移运动 44
4.3.1 直线运动的物体 44
4.3.2 沿轨道运动的物体 46
4.3.3 物体的转动 49
4.3.4 物体运动参数的设定 49
4.4 神奇的图形渐变 51
4.4.1 简单示例 51
4.4.2 提示点的设定 53
4.4.3 图形渐变参数的设定 56
4.5 逐帧动画 57

第二部分 Flash MX 提高篇

第 5 章 复杂的 Flash 动画 61

5.1 综 述 61
5.2 如何使用层 61
5.2.1 层的优点 61
5.2.2 层的编辑和使用 61
5.2.3 层的应用 64
5.2.4 设定层的属性 72
5.3 使用元件 74
5.3.1 元件的概述 74
5.3.2 创建元件 74
5.3.3 影片剪辑 75
5.3.4 图形元件 80
5.3.5 按钮元件 81
5.3.6 通用元件库的使用 86

第 6 章　交互式 Flash 动画基础 ····· 87

6.1　综　述 ····· 87
6.2　使用动作面板 ····· 87
6.2.1　动作面板简介 ····· 87
6.2.2　标准模式编辑 ····· 87
6.2.3　专家模式编辑 ····· 90
6.2.4　动作面板设置菜单 ····· 91
6.3　ActionScript 脚本语言的编写 ····· 92
6.3.1　ActionScript 脚本语言的语法 ····· 92
6.3.2　数据类型 ····· 92
6.3.3　变量的使用 ····· 94
6.3.4　表达式与运算符 ····· 95
6.3.5　使用 Action 动作 ····· 97
6.3.6　控制播放语句 ····· 98
6.3.7　调用函数 ····· 100
6.3.8　自定义函数 ····· 101
6.3.9　使用对象 ····· 102
6.3.10　使用自定义对象 ····· 103
6.3.11　指明路径 ····· 105
6.3.12　数组对象的使用 ····· 106
6.3.13　跟踪调试 ····· 110
6.3.14　影片导航面板的使用 ····· 112

第 7 章　创作交互式 Flash 动画 ····· 115

7.1　综　述 ····· 115
7.2　响应事件 ····· 115
7.2.1　关于事件 ····· 115
7.2.2　按钮事件 ····· 116
7.2.3　影片剪辑事件 ····· 118
7.3　获得鼠标的位置 ····· 120
7.4　创建自己独特风格的鼠标指针 ····· 121
7.5　测试鼠标指针的移动速度 ····· 123
7.6　捕获键盘按键 ····· 124
7.7　幻影移动 ····· 126

第 8 章　组件的使用 ····· 132

8.1　综　述 ····· 132
8.2　组件的概述 ····· 132
8.3　自定义组件 ····· 133
8.3.1　创建自定义组件的步骤 ····· 133
8.3.2　为影片剪辑添加参数 ····· 133
8.3.3　通用滑动条的制作 ····· 135
8.3.4　使用自定义的组件 ····· 138

第 9 章　外部文件的导入 ····· 143

9.1 综 述 143
9.2 导入图片文件 143
9.2.1 图片文件的类型（矢量图与位图） 143
9.2.2 导入图片及图片组 144
9.2.3 把位图转换为矢量图 146
9.2.4 位图填充 147
9.2.5 编辑位图 148
9.2.6 设置位图格式 149
9.3 导入声音 149
9.3.1 Flash MX 所支持的声音文件 150
9.3.2 为动画配音 150
9.3.3 让按钮发音 151
9.3.4 声音的编辑与控制 151
9.3.5 声音格式的设定 152
9.3.6 动态调节音量 155

第三部分 Flash MX 应用篇

第 10 章 Flash MX 动画的发布 160

10.1 综 述 160
10.2 输出动画的格式 161
10.2.1 输出 SWF 动画 161
10.2.2 输出 GIF 动画 163
10.2.3 输出 JPEG 文件 165
10.2.4 输出 PNG 文件 165
10.2.5 输出 QuickTime 电影 166
10.3 动画的优化 167
10.4 将 Flash MX 动画添加到网页中 168
10.4.1 Flash 的插件 168
10.4.2 直接输出 HTML 文件 168
10.4.3 使用 Dreamweaver 添加动画 171
10.4.4 测试 Flash 动画的下载过程 173

第 11 章 Flash MX 与网络 176

11.1 综 述 176
11.2 参数、变量的传递 176
11.2.1 从外部文件装入变量 176
11.2.2 以前的方式 177
11.2.3 getURL 的使用 177
11.2.4 LoadVariables 的使用 178
11.2.5 LoadMovie 动作 183
11.2.6 XML 概述 183
11.2.7 使用 XML 对象 184
11.2.8 使用 XMLSocket 对象 186
11.3 向播放器发送消息 187

11.3.1 向 Flash player 发送消息 187
11.3.2 向网页发送消息 188

第 12 章 Flash MX 动画实战 191

12.1 动画“雪花纷飞” 191
12.1.1 原 理 191
12.1.2 实 战 191
12.1.3 总 结 197
12.1.4 扩 展 199
12.2 旋转的空间立方体顶点 200
12.2.1 介 绍 200
12.2.2 3D 变换矩阵 201
12.2.3 实 战 203
12.2.4 总 结 206
12.3 可以控制转动的立方体 209
12.3.1 介 绍 209
12.3.2 错切的实现 210
12.3.3 平面的消隐 213
12.3.4 实 战 214
12.3.5 总 结 221
12.4 制作自己的 MTV 221
12.4.1 介 绍 221
12.4.2 实 战 221
12.4.3 总 结 232
12.5 下载进度的显示 233
12.5.1 介 绍 233
12.5.2 实 战 233
12.5.3 总 结 237
12.6 用 Flash 实现讨论区 238
12.6.1 介 绍 238
12.6.2 实 战 238
12.6.3 总 结 261
12.7 桌面台历时钟 261
12.7.1 介 绍 261
12.7.2 实 战 261
12.8 打字练习软件 267
12.8.1 介 绍 267
12.8.2 实 战 267

附录 A ActionScript 脚本语言参考 272

A.1 Functions 函数 272
A.2 字符串函数 272
A.3 Properties 属性 273
A.4 对 象 274

附录 B Flash MX 快捷键 283

B.1 菜单命令 283
B.2 工具箱中的快捷键 287

第一部分

Flash MX 入门篇

第 1 章　Flash MX 基础

第 2 章　绘图

第 3 章　颜色的使用

第 4 章　简单的 Flash 动画

第 1 章　Flash MX 基础

1.1　Flash MX 概述

1.1.1　Flash MX 的功能

在 Flash 尚未诞生之前，因特网上的动画大多是 GIF 动画或 Java 动画。前者文件尺寸很大，后者要求制作者有较高的编程能力。Macromedia 公司推出的 Flash MX 提供了创作网络动画的新途径。Flash 动画具有如下特点：

- 体积小。Macromedia 公司使用了插件技术，大大缩小了 Flash 动画文件的尺寸。Flash 中的声音文件使用了 MP3 的压缩格式，在保证高质量声音的同时，也减小了 Flash 动画的尺寸。
- Flash 动画是一种 Stream（流式）动画。所以，Flash 动画在因特网上可以边下载边运行。这一特点是 Java 动画无法比拟的。
- 创建和编辑 Flash 动画的方法简单易学。通过使用关键帧和渐变的技术，Flash 简化了动画的创建过程。
- 强大的交互能力。通过交互功能使动画与网页有机地结合起来，可以创造出复杂的动画，绝不逊色于 Java。
- 可以输出多种格式的电影文件。利用 Flash MX 创建的动画，不仅可以生成 Flash 格式（*.swf）的动画，还可以输出为 GIF 动画、MOV 动画、AVI 动画以及.exe 文件或 Java 动画。

因此，越来越多的用户开始使用 Flash 创建自己所需的动画或者 Flash 版的主页。随着 Flash 的发展，Flash 动画不仅应用在因特网上，现在许多电视广告、电脑游戏的片头和片尾都是用 Flash 做成的。Flash 还可以代替 PowerPoint 制作更精巧的幻灯片进行播放。利用 Flash 可以直接输出 Windows 可执行文件（*.exe）的功能，可以便捷地制作出 Flash 游戏。使用 Flash 的插件，能把 Flash 动画直接嵌入到 VB、VC 所生成的 Windows 可执行文件（*.exe）之中。可以说，Flash 已经成为进行多媒体制作不可缺少的工具。

1.1.2　Flash MX 的新特性

与 Flash 5 相比，Flash MX 具有以下一些新特性：

- 更简洁、更专业的工作界面。
- 新增层文件夹功能，使图层成组，操作更简单、更明快。
- 增加了自由变形工具，在编辑动画时，物体变形自由自在。
- 增加了模板功能，定制动画更简单。
- 实现了对视频文件的支持，这是一个最大的改进。
- Flash MX 支持多国语言，使其代码应用更加广泛。
- 增加了组件的概念，使 Flash 资源的共享更为广泛、简便。

1.2　Flash MX 的安装

作为一个标准的 Windows 应用程序，Flash MX 有良好的安装界面。具体过程如下：

（1）启动资源管理器，在安装盘所在的驱动器中双击 Setup.exe 文件，会出现如图 1-1 所示的 Flash

MX 安装向导。

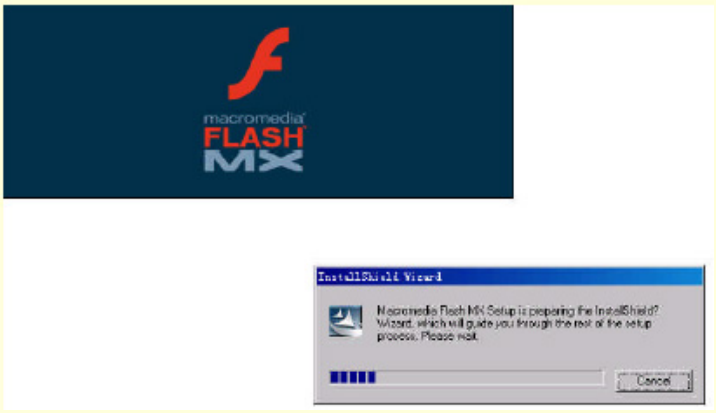


图 1-1　安装向导

（2）进入安装程序以后，将出现如图 1-2 所示的安装 Flash MX 的提示信息。

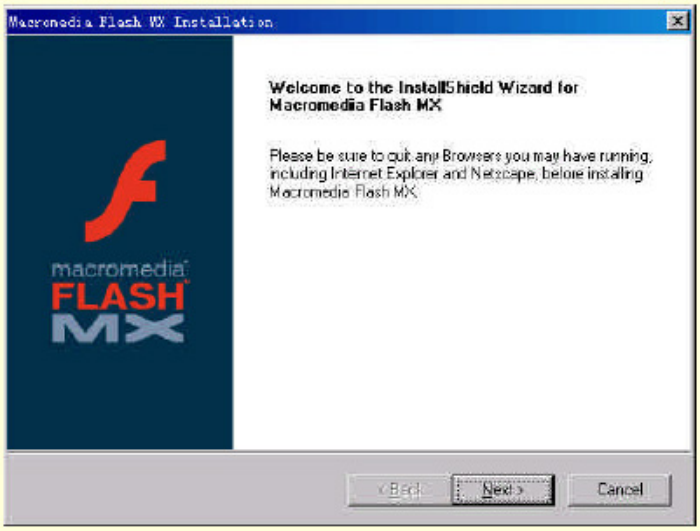


图 1-2　安装提示

（3）单击 Next（下一步）按钮，将显示如图 1-3 所示的关于软件使用许可的协议。在用户安装协议中说明了用户的权利和义务。

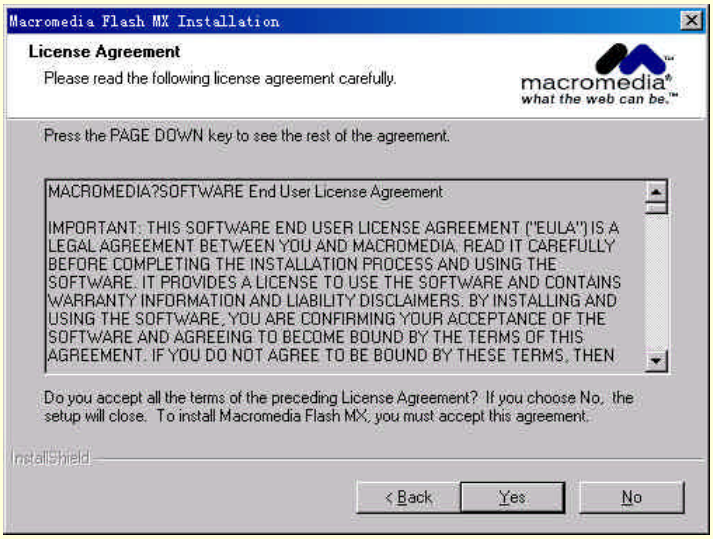


图 1-3　软件使用许可

（4）单击 Yes 按钮，将显示如图 1-4 所示的用户名和产品序列号输入对话框。输入用户名和所在组织名，再输入正确的产品序列号，下方的“继续”按钮将变为可用。

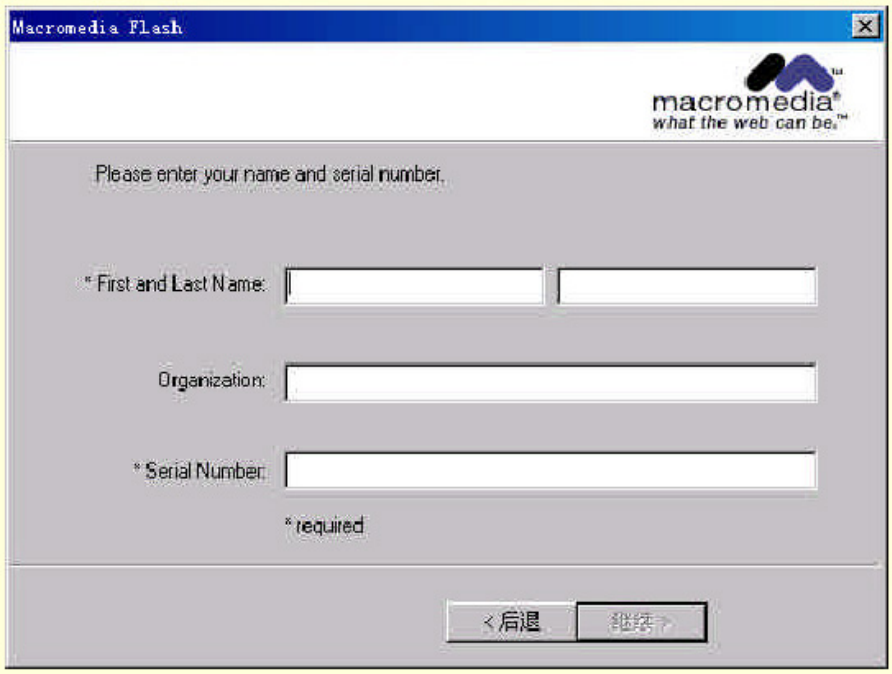


图 1-4　输入用户名和产品序列号

（5）单击“继续”按钮，进入如图 1-5 所示的安装目录选择界面，用户可以选择安装目录或使用默认目录。如果希望改变目录，单击 Browse（浏览）按钮，在弹出的对话框中键入或直接选择存在的目录。

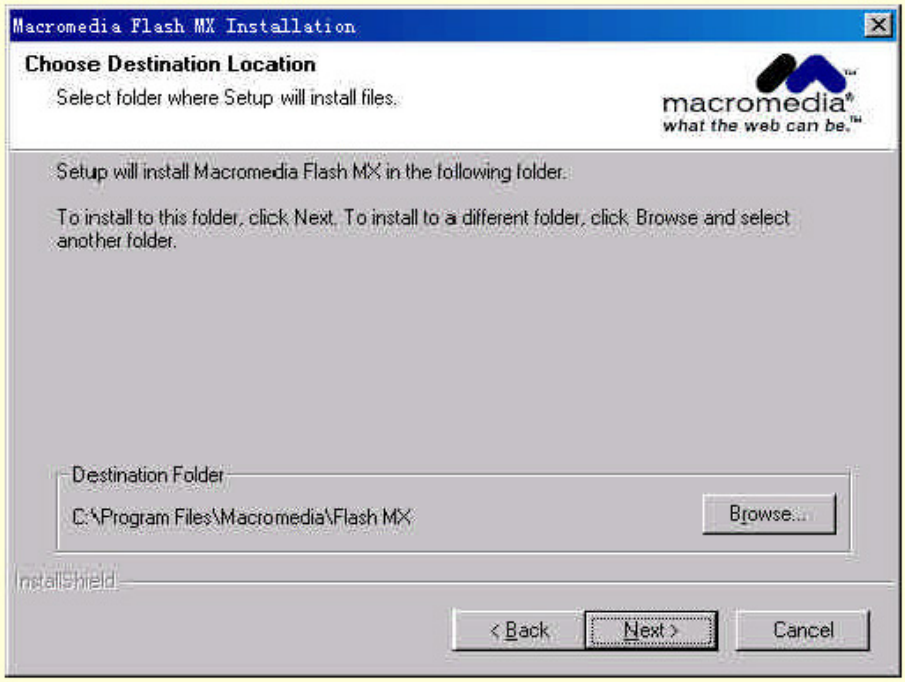


图 1-5　选择安装目录

（6）单击 Next（下一步）按钮，将出现如图 1-6 所示的 Flash MX 的浏览器插件选择界面。如果选中界面中的复选框，安装程序将自动更新浏览器中 Flash MX 的播放插件，建议选中该选项。

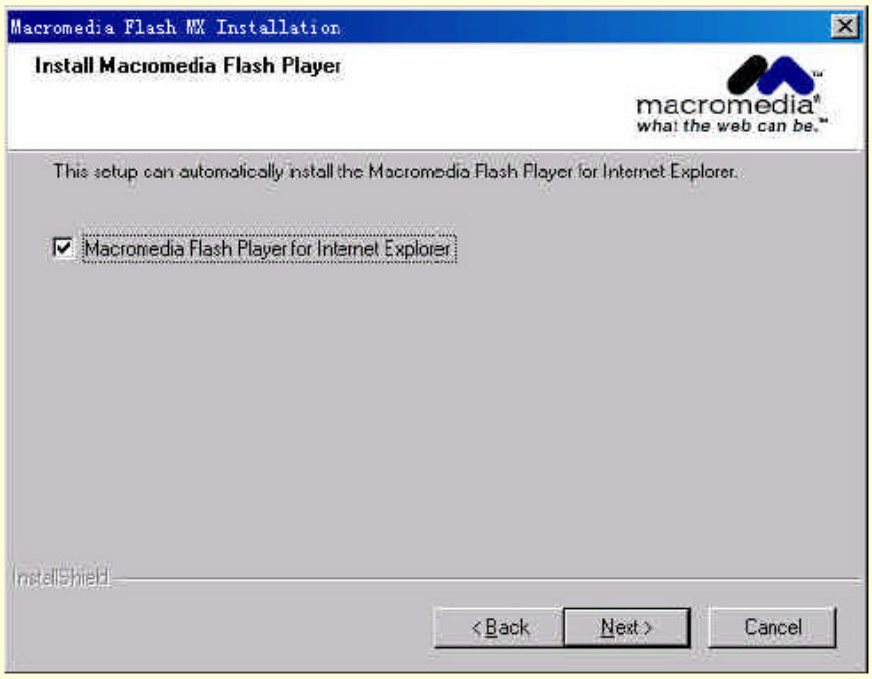


图 1-6　插件选择

（7）单击 Next（下一步）按钮，安装程序将显示如图 1-7 所示的用户所选择的配置，如果发现有不妥之处，可以逐级单击 Back（返回）按钮重新设置。

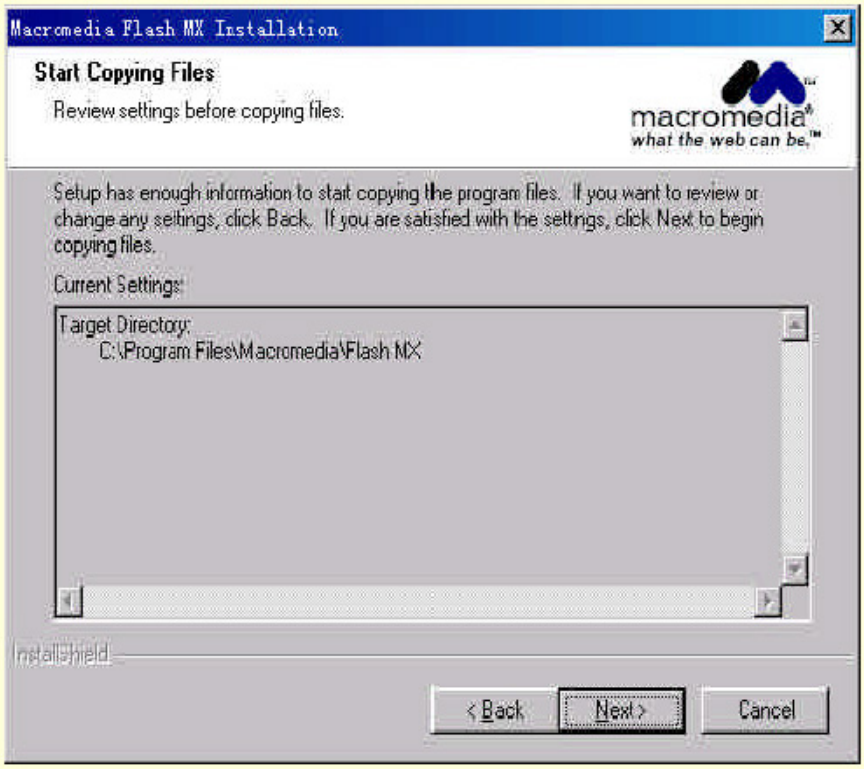


图 1-7　用户所选择的配置

（8）单击 Next（下一步）按钮，安装程序将把 Flash MX 复制到硬盘上，在复制文件的过程中可从如图 1-8 所示的进度条上直观地看到已安装部分的百分比。在复制过程中可以单击 Cancel（取消）按钮退出安装程序。

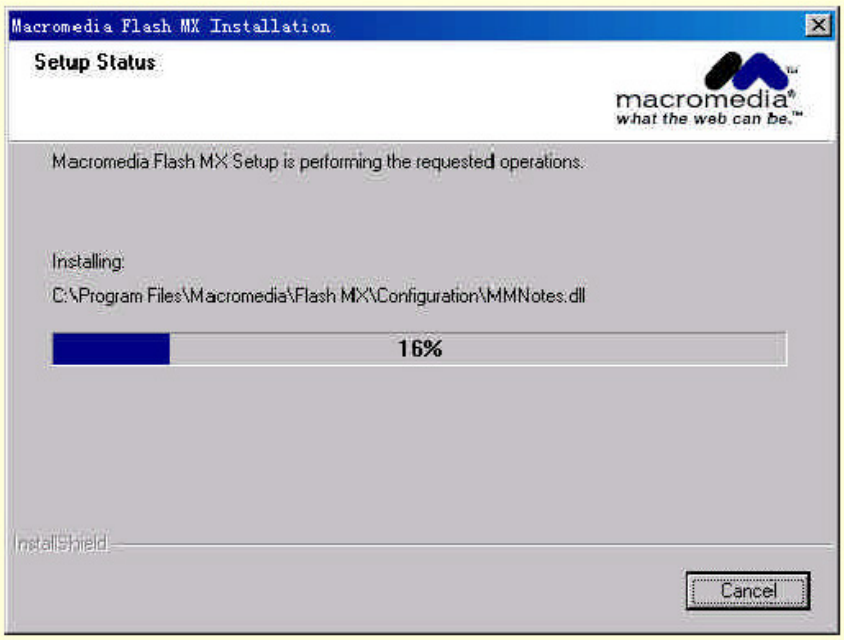


图 1-8　安装进度

（9）最后，屏幕上将出现如图 1-9 所示的安装完成的界面。安装向导提示要第一次运行 Flash MX，必须重新启动电脑。如果不想重新启动电脑，选择 No，再单击 Finish 按钮，完成 Flash MX 的安装。

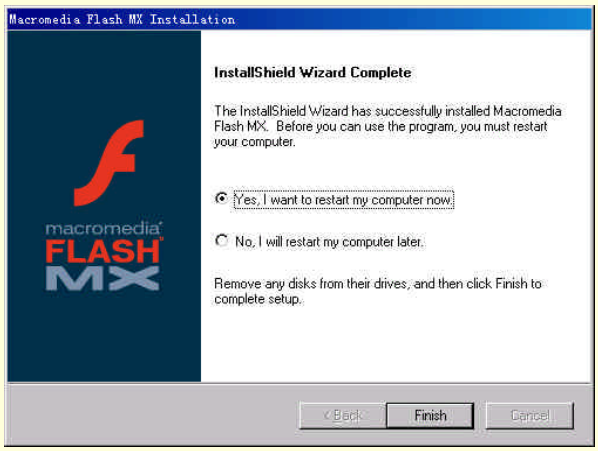


图 1-9　安装结束

（10）安装程序自动创建如图 1-10 所示的 Flash MX 程序组，并显示 ReadMe 文件。

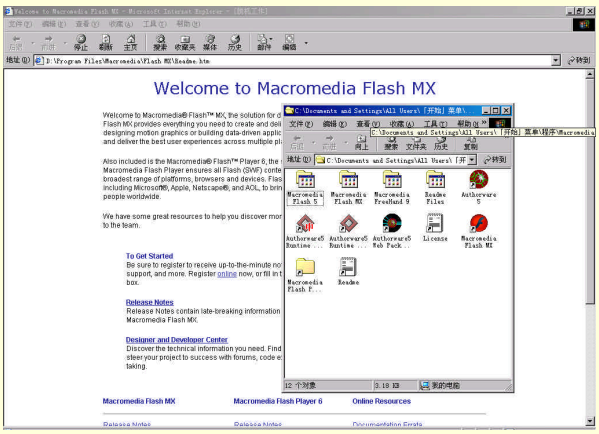


图 1-10　创建程序组并显示自述文件

1.3 使用 Flash MX

1.3.1 Flash MX 的启动

Flash MX 安装完成以后，如图 1-11 所示，可以在 Windows 的“开始”菜单中启动 Flash MX，用户也可以按照自己的习惯创建其他的快捷方式。

图 1-11 在 Windows 的开始菜单中的 Flash MX

1.3.2 友好的 Flash MX 用户界面

第一次进入 Flash MX 将出现一个带帮助信息的 Welcome（欢迎）面板，如图 1-12 所示。在这个面板中，用户可以根据自己的工作侧重点选择 Designer（设计师）、General（普通用户）、Developer（开发人员）三个选项，Flash MX 将在 Welcome 面板中列出相应的帮助信息。

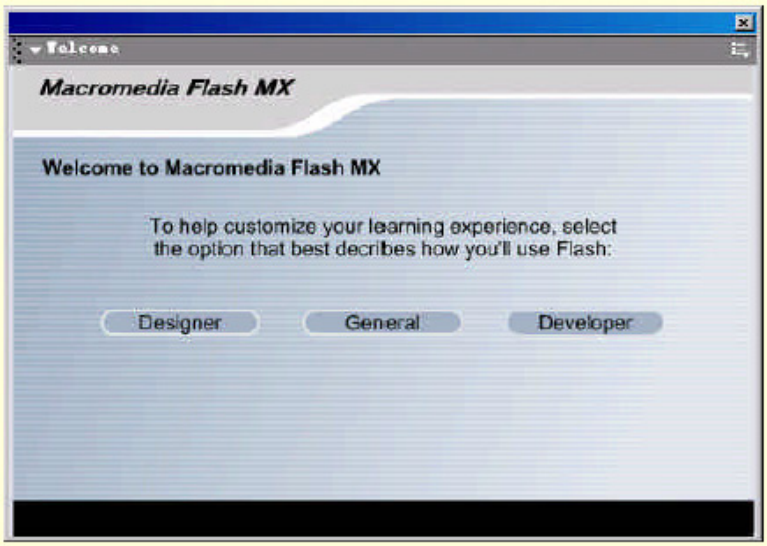


图 1-12 工作界面选择面板

单击 Welcome 面板右上角的关闭按钮，关闭 Welcome 面板，将出现如图 1-13 所示的 Flash MX 的工作界面，对其中各部分的介绍如下：

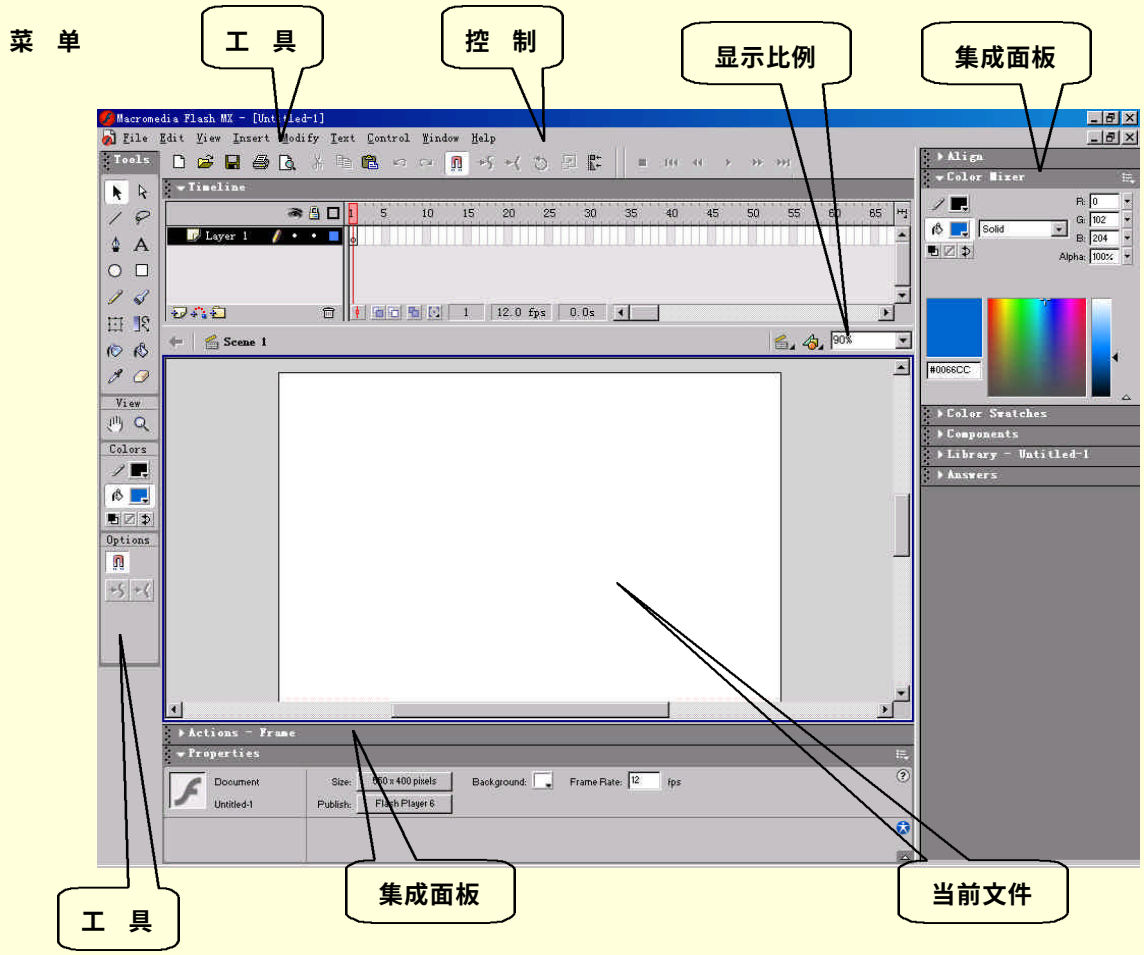


图 1-13 Flash MX 工作界面

- 菜单栏：包括 File（文件）、Edit（编辑）、View（显示）、Insert（插入）、Modify（修改）、Text（文本）、Control（控制）、Window（窗口）、Help（帮助）等 9 个菜单。Flash MX 的所有功能都可以通过单击相应的菜单命令来实现。
- 工具栏：常用的 Flash MX 命令都可以在工具栏中找到。
- 控制栏：可以控制 Flash MX 动画的 Play（播放）、Rewind（重播）、Go to end（结束）、Stop（停止）、Step back（返回）、Step Forward（前进）等。
- 工具箱
  - Tools（绘图工具）：通过各种实用的工具，实现 Flash MX 图片的绘制。关于绘图的详细说明将在第 2 章中详细说明。
  - View（显示）控制：在编辑过程中可以控制显示的比例及位置。
  - Colors（颜色）控制：使用各种着色工具，对 Flash MX 图片进行着色填充。关于着色的详细说明将在第 3 章中详细说明。
  - Options（选项）控制：它包括磁铁工具、平滑和锐化工具、缩放和旋转工具等，可以方便地实现 Flash MX 对象的修改。
- 集成面板：包括 Info（信息）面板、Transform（转换）面板、Color Mixer（调色）面板、Color Swatch（颜色样本）面板、Scene（场景）面板、Movie Exporler（影片浏览器）面板、Library（元件库）面板、Actions（动作）面板、Properties（属性）面板等。
- 当前文件：显示正在打开或编辑的文件。
- 显示比例：方便地改变当前文件显示的比例。

1.3.3 浏览 Flash MX 动画

选择如图 1-14 所示的 Flash MX 的 Control | Test Movie 菜单选项，或者直接按 Ctrl+Enter 快捷键即可输出并显示动画，动画的输出进程如图 1-15 所示。

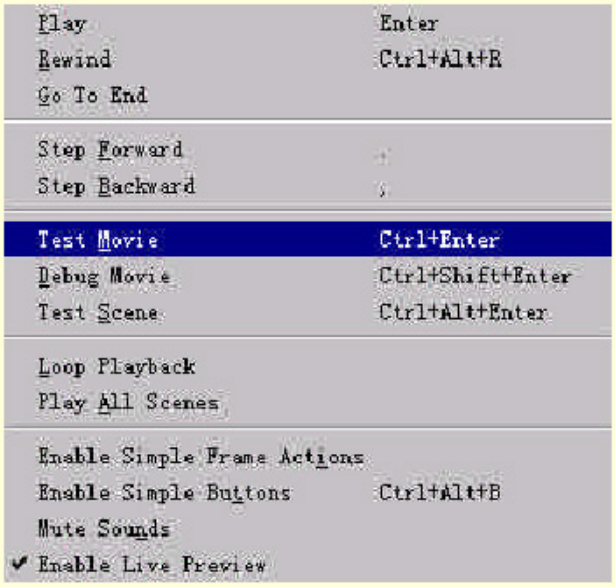


图 1-14 Control | Test Movie 菜单

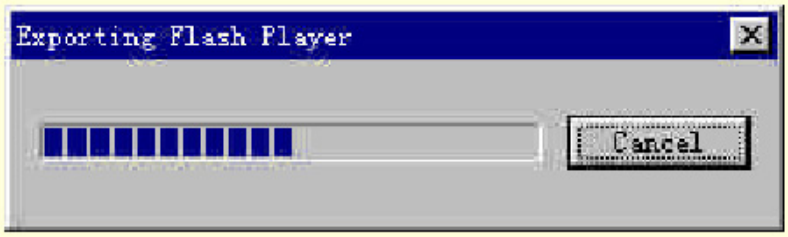


图 1-15 动画的输出进程

Flash MX 动画运行时，在动画上任意位置右击鼠标，便会弹出一个如图 1-16 所示的快捷菜单，其具体功能如下：

- Zoom（缩放）
  - Zoom In（放大）：放大显示动画
  - Zoom Out（缩小）：缩小显示动画
  - 100%：按照 1:1 的比例显示动画
  - Show All（全部显示）：自动缩放比例，显示动画的全部
- Quality（显示质量）分为 Low（低）、Medium（中）、High（高）3 个档次，当动画较复杂或者系统配置较低时，降低动画的显示质量可以提高运行速度。关于动画质量的控制，将在第 10 章中作进一步讨论。
- 播放控制可以控制 Flash MX 动画的 Play（播放）、Rewind（重播）、Loop（循环播放）、Back（返回）、Forward（前进）。
- Print（打印）：实现当前动画画面的打印输出。
- Debugger（跟踪）：跟踪 Flash MX 动画的运行，在调试动画时使用，具体内容将在第 6 章中予以介绍。

图 1-16　快捷菜单

✍ Test Movie（影片测试）的结果就是将生成的 Flash MX 动画文件（*.swf）进行播放，要想停止影片测试，直接关闭测试窗口即可。正在编辑的 Flash MX 文档（*.fla）不会被关闭。

第 2 章　绘　图

2.1　综　述

如图 2-1 所示，Flash MX 的工具栏为用户提供了各种各样的绘图工具。利用这些工具可以绘制出各种图形，并为其设置相应的属性，如颜色、大小、位置等。

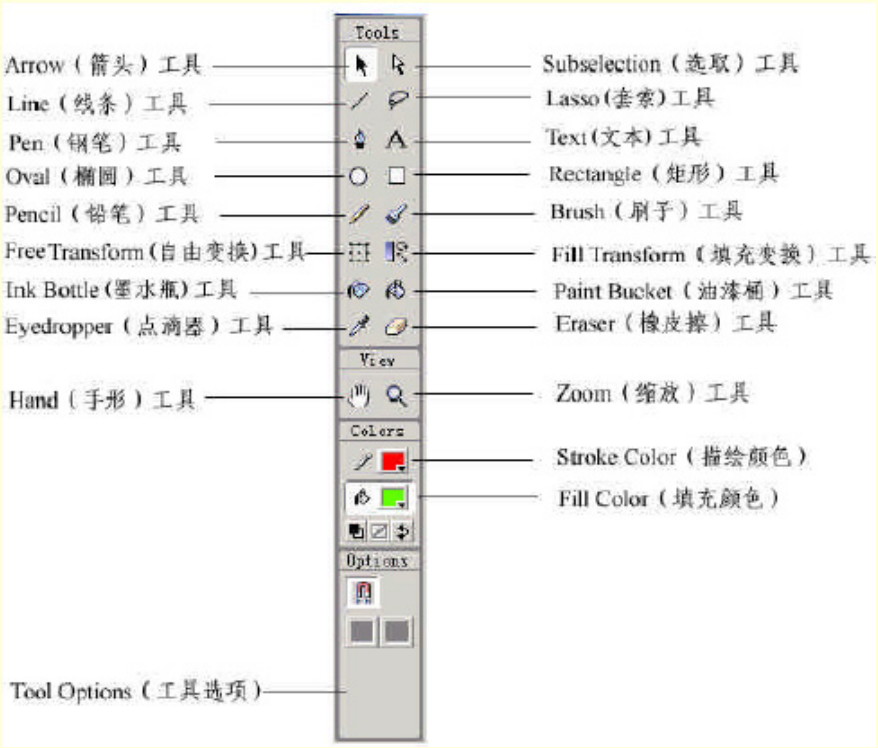


图 2-1　Flash MX 的工具栏

2.2　Flash MX 的绘图工具

2.2.1　Line（线条）工具

利用 Flash MX 提供的 Line（线条）工具可以快捷地绘制出所需的直线。

先在工具栏中单击 Line（线条）工具，使其处于如图 2-2 所示的选中状态，在 Stroke Color（描绘颜色）框中选择所需的颜色，然后就可以绘制所需的直线了。此外，也可以在 Properties 面板（如果屏幕上没有显示，可通过选择 Windows | Properties 菜单项使其显示）中选择所需直线的颜色、粗细和线型（如图 2-3 所示）。如果绘制直线时按住 Shift 键，那么绘出的将是如图 2-3 所示的水平、垂直或正负 45° 方向的直线。

图 2-2　选取线条工具

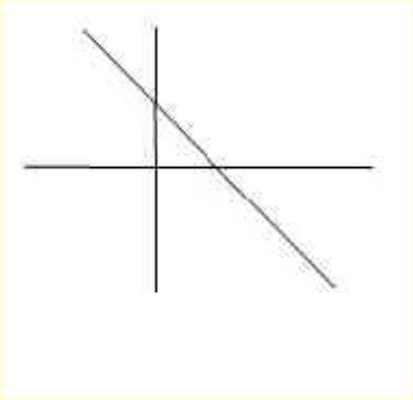


图 2-4　绘制直线

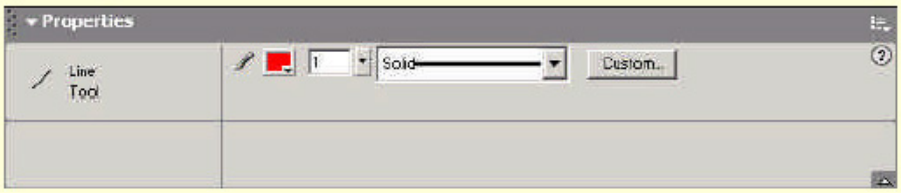


图 2-3　线条工具的属性面板

2.2.2　Lasso（套索）工具

用套索工具可以选取一定的区域，然后用其他工具对选中区域进行修改。选中工具栏中的套索工具（如图 2-5 所示），然后在工作区中圈画出要选中的区域，松开鼠标后 Flash 会自动选取套索圈定的封闭或近似封闭区域。此时放在被选中区域上的鼠标会变成箭头形状，按住鼠标可以拖动被选中的区域，原图形即被拆分（如图 2-6 所示）。

图 2-5　选取套索工具

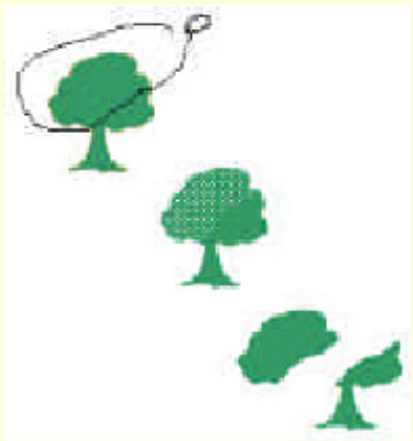


图 2-6　用套索选取

- 多边形选择模式：如图 2-7 所示的 Polygon Mode（多边形选择模式）可以实现对多边形区域的选取，选中此模式之后用鼠标单击工作区中的若干点，如图 2-8 所示，由这些点构成的多边形区域将被选中。

图 2-7　选取多边形模式

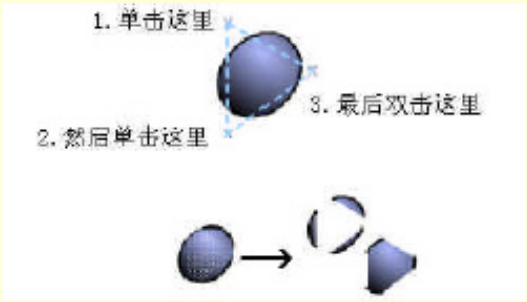


图 2-8　选取多边形区域

- 当所需选定的点都已用鼠标单击后，需双击左键结束选择。

- **魔棒模式：通过选取** Magic Wand（魔棒模式）**的各种属性，能够对套索选择方式进行各种变化。选中魔棒模式然后单击魔棒模式属性图标，会弹出如图** 2-9 **所示魔棒模式属性对话框，用户可以自行定义** Threshold（平滑限度），在 Smoothing（平滑）下拉式列表框中可以选择 Pixels（像素）、Rough（粗糙）、Normal（标准）以及 Smooth（平滑）四种属性之一。

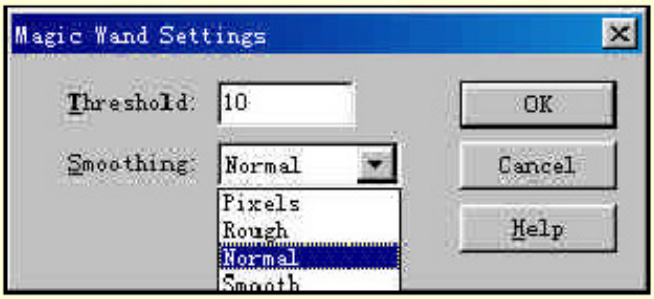


图 2-9　魔棒模式属性对话框

2.2.3　Pen（钢笔）工具

如果需要绘制精确的直线或曲线可以使用钢笔工具，先大致画出直线或曲线，然后将直线的长度、角度及曲线的倾斜度调整至所需的样子。通过对直线及曲线上的点的调整，可以将直线调整为曲线，也可以将曲线拉伸成直线。

钢笔工具还可以调整由其他 Flash 工具生成的图形上的点，例如由铅笔工具、椭圆形工具、矩形工具生成的图形点等。

- **画直线：选中如图** 2-10 **所示的工具箱中的钢笔工具，选择** Window | Properties **菜单项后，会弹出如图** 2-11 **所示的** Properties（属性）**面板，在该面板中选择合适的线型、粗细及颜色，然后在工作区中单击鼠标定义直线的起始点，之后在需要的位置再次单击鼠标定义直线的终止点。按住** Shift **键单击鼠标可以以** 45 ° **方式绘出如图** 2-12 **所示的折线。**
- **画曲线：方法类似于直线的画法，先定义起始点，在定义终止点时按住鼠标左键不放会出现一个平衡棒（即曲线端点处的切线段），移动鼠标会使平衡棒跟着转动以改变曲线的斜率，松开鼠标后曲线形状便被确定了。**

图 2-10 选取钢笔工具

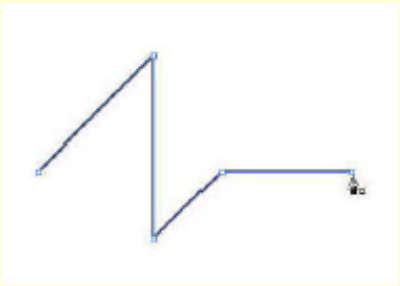


图 2-12 用钢笔工具绘直线

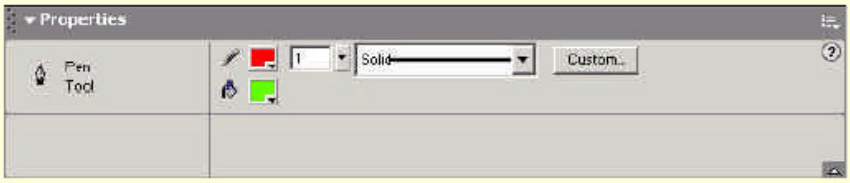


图 2-11 钢笔工具的属性面板

对于不同种类的曲线，要以不同的方法结束绘制：

- 开口曲线：双击曲线终止点、单击工具箱中的钢笔工具或在曲线外按住 Ctrl 键单击鼠标可以结束对开口曲线的绘制。
- 闭合曲线：如图 2-13 所示，将钢笔工具移至曲线起始点处，当位置正确时在钢笔工具右下角会出现一小圆圈，单击即可。
- 还可用如下方法完成对曲线的绘制：如图 2-14 所示，选择 Edit | Deselect All（编辑 | 撤消全部选择），或者在工具箱中选择任何其他工具。

图 2-13 用钢笔工具绘曲线

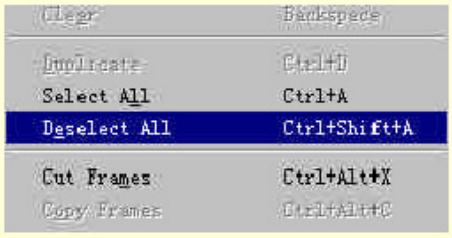


图 2-14 撤消全部选择

2.2.4 Text（文本）工具

用文本工具可以在工作区中定义文本区。文本内容可以在一行中输入，让 Flash 自动调整文本区的长度；也可以将文本内容用多行方式输入，让 Flash 自动锁定每一行的长度。

如果选中文本工具以后，如图 2-15 所示，用鼠标拖拽出一个文本区域。则 Flash 会在文本区的右上角显示一个如图 2-16 所示的小正方形手柄，表示文本区域的长度是固定的，在文本输入至每一行末尾时，Flash 会自动换行以保证文本区域的宽度不变。用户也可以根据输入文本的篇幅调整手柄以达到满意的效果。

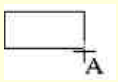


图 2-15　拖拽出文本区域

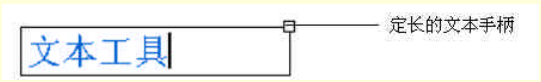


图 2-16　定长文本区域

选中文本工具以后用鼠标在要添加文本的地方单击，再添加文本，则文本区的右上角出现的是如图 2-17 所示一个圆形手柄，表示此文本区的长度是随文字自动扩展的，如果在文本中不输入回车，文本是永远不会换行的。

图 2-17　可变长文本区域

- 文本区起始点由用户根据需要自行定义，文本区域创建以后，也可以用鼠标直接将其拖动到想要的位置。

如图 2-18 所示，用鼠标拖动可变长文本的手柄，可以把可变长文本区域改变为定长文本区域；在定长文本区域的手柄上双击鼠标，可以把定长的文本区域变为可变长度的文本区域。

图 2-18　将可变长文本区域变为定长

文本属性的设置：通过此设置可以根据需要定义文本的各种属性，如字体颜色、大小、字型、与因特网的连接等。选择 Window | Properties 菜单命令，或者直接在文本区域上右击鼠标，选择快捷菜单中的 Properties 菜单项，即可显示出如图 2-19 所示的文本工具的属性面板，包括：

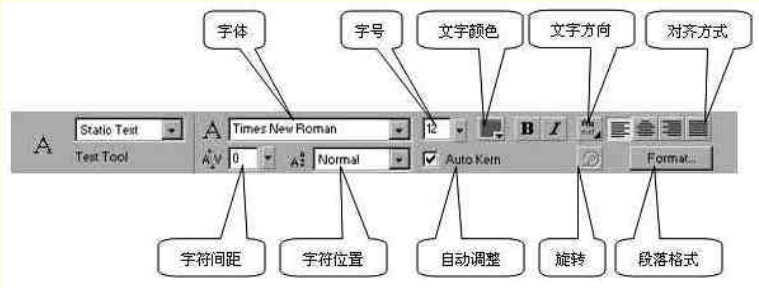


图 2-19　文本工具的属性面板

- 设定字体：在字体下拉式列表框中可选择不同的字体。
- 设定字号：在控制字号的文本框中添入数字，或者单击文本框右边的倒三角箭头，然后拖动滑块，可以调整字的大小。
- 设定文字颜色：在调色板中可以选择不同的字体颜色。
- 按钮 B 与 *I* 分别代表加粗与斜体，这与 Microsoft Word 中的相同。
- 设定文字方向：单击文字方向按钮会出现三个选项：Horizontal（水平方向）、Vertical,Left to Right（竖直方向从左至右）、Vertical,Right to Left（竖直方向从右至左），分别代表如图 2-20 所示的三种文字方向。另外，在文字方向下方的旋转按钮在文字方向为竖直的情况下将变为可选，其效果如图 2-20 所示。
- 排列文本：在 Align（排列）项中有四个按钮，当文字方向为水平时分别代表左侧、中间、右侧、分散对齐；为竖直方向时则代表上方、中间、下方、分散对齐。特别要注意的是分散对齐，他表示文本按正常方式输入且段落充满整个文本框。
- 设定字符间距：在设定字符间距的文本框中输入数字，或者单击文本框右边的倒三角箭头然后

拖动滑块，可以调整字符之间间距的大小。

- 设定字符位置：该下拉列表框中有 Normal（标准）、Superscript（上标）、Subscript（下标）三个选项，用于设定文字为上标或者下标。

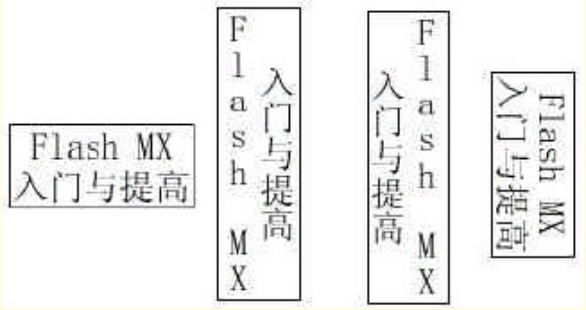


图 2-20 文字排列方向

- Auto Kern（自动调整）复选框：该复选框用于设定字符之间是否紧密排列。选中此复选框后，A 与 V 的间距将比 A 与 D 的间距小，即字体间距是可变的。
- 设定设定段落格式：单击 Format 按钮会弹出如图 2-21 所示的段落格式对话框，其中四个文本框可依次填入 Indent（首行缩进量）、Line Spacing（行间距）、Left（左边界缩进量）、Right（右边界缩进量），在框中键入数字或单击文本框右边的倒三角箭头后拖动滑块即可进行调整。

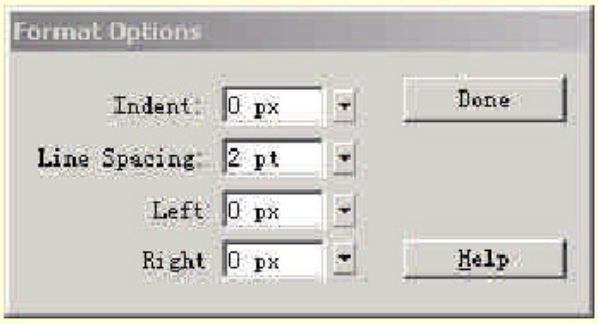


图 2-21 段落格式对话框

在文本工具属性面板最左边的下拉列表框中可以设定三种文本类型：Static Text（静态文本）、Dynamic Text（动态文本）、Input Text（输入文本）。下面分别进行说明：

- Static Text（静态文本）：图 2-22 所示的静态文本类型只有两个可选项。
  - 使用 Use Device Fonts（设备字体）。
  - Selectable（允许文本被选中）。
- Dynamic Text（动态文本）：图 2-23 所示的动态文本类型中包括如下选项。
  - 在文本类型列表框下面的下拉式列表框中可以设置文本区域为 Single Line（单行文本）或者是 Multiline（多行文本）。
  - 按下 HTML 按钮，可以设置文本区域为超文本格式。
  - 按下 Border/Bg 按钮，可以使文本区域具有边框。
  - 按下 Selectable 按钮，允许文本被选中。
  - 在 Var 文本框中可以设置文本框变量，通过 ActionScript 脚本控制在文本区域中动态显示文本。
  - Character 按钮用来在 Flash MX 文件中嵌入字体文件。单击 Character 按钮，将显示如图 2-24 所示的 Character Options 对话框，在此可以选择是否嵌入当前字体，全部嵌入还是只嵌入一部分。另外，在文本框中也可以直接键入需要嵌入的文本。

图 2-22　文本标签

图 2-23　动态文本

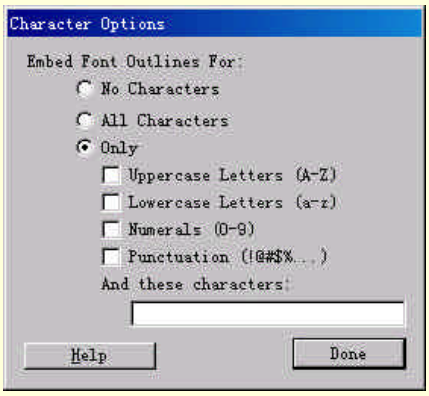


图 2-24　Character Options 对话框

- Input Text（输入文本）：输入文本类型可以在 Flash MX 动画的运行过程中输入文本，它的功能与动态文本基本相同，但多了如下两种功能。
  - 密码文本：除了 Single Line（单行文本）和 Multiline（多行文本）之外，还有密码文本的功能。它可以将用户输入的密码用“*”符号屏蔽掉。
  - 最多字符：在 Max Chars 文本框中可以设定文本框中允许输入的最多字符数。如果不需要明确指出字符数，则键入 0。

2.2.5　图形工具

Flash 的图形工具包括 Oval（椭圆）工具、Rectangle（矩形）工具等（线形工具也属于图形工具，前面已做介绍）。用这些图形工具可以生成基本的 Flash 矢量图形，并且可以对椭圆及矩形设置所需的描绘颜色及填充颜色。

✎ 对 Line（线形）工具不能设置其填充色属性，因为用线形工具绘出的图形不存在区域的概念，所以也就不存在可以填充颜色的区域。

- Oval（椭圆）工具：如图 2-25 所示，选中工具箱中的椭圆工具，按住鼠标左键即可在工作区中拖拽出一个椭圆。如果按住 Shift 键可以拖出一个圆形，如图 2-26 所示，绘制出的圆形具有预先设置的属性（描绘颜色及填充颜色）。
- Rectangle（矩形）工具：如图 2-27 所示，选中工具箱中的矩形工具，按住鼠标左键即可在工作区中拖拽出一个矩形。如果按住 Shift 键可以拖出一个正方形，如图 2-28 所示，绘制出的矩形具有预先设置的属性（描绘颜色及填充颜色）。

图 2-25 选中椭圆工具

图 2-26 绘制圆

图 2-27 选中矩形工具

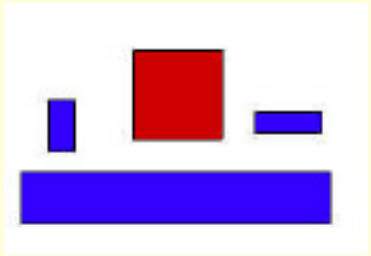


图 2-28 绘制矩形

- 在如图 2-29 所示的 Option（选项）栏中单击 Round Rectangle Radius（矩形圆角半径）选项，弹出如图 2-30 所示的角度设置对话框，在 Corner（角度）文本框中添入一定的数值会使矩形四角圆滑，如果添入的角度为零度，则矩形的四个角为直角，如图 2-31 所示。
- 在拖制矩形时如果按住鼠标同时按键盘上的上、下两个方向键，同样会使矩形的角度圆滑程度发生变化。

图 2-29 选项栏

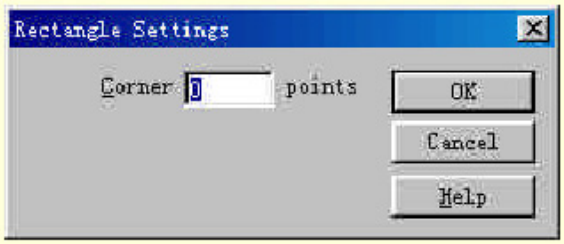


图 2-30 角度设置对话框

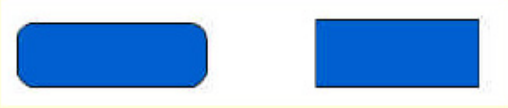


图 2-31 圆角矩形

2.2.6 Pencil（铅笔）工具

Flash 的 Pencil（铅笔）工具具有独特的功能，用它可以绘制线条、轮廓以及其他形状的圆滑曲线或折线。

- 绘制折线：如图 2-32 所示，选中 Pencil Tool（铅笔工具）及铅笔模式修改选项中的 Straighten（锐化）模式，然后如图 2-33 所示，在工作区中画出一个三角形，松开鼠标后 Flash 会自动将

此线条折线化。

- 绘制平滑曲线：如图 2-34 所示，与绘制折线类似，但此时要选中 Smooth（平滑）模式，此后绘制出的曲线会被自动圆滑。
- 绘制墨水曲线：选中如图 2-35 所示的 Ink（墨水）模式，在此模式下绘出的曲线会不经任何变化存在于工作区中。

图 2-32　用铅笔工具绘图

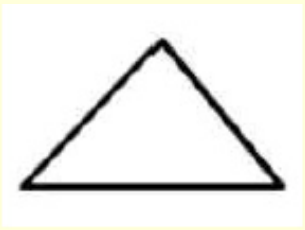


图 2-33　折线

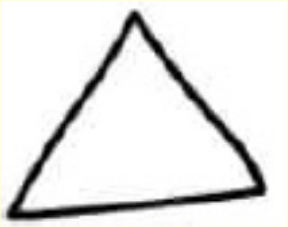


图 2-34　平滑曲线

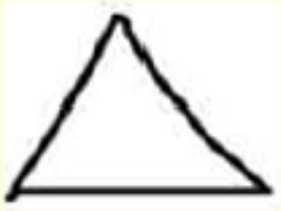


图 2-35　墨水曲线

如果绘制直线时按住 Shift 键，那么绘出的将是水平方向与垂直方向的直线。

2.2.7　Brush（刷子）工具

Flash MX 的刷子工具可以产生刷子粉刷效果的线条图形，如图 2-36 所示，选择不同的刷子形状及刷子大小可以产生各种不同的样式。

图 2-36　不同的画刷工具的绘制结果

- 刷子工具的使用：如图 2-37 所示，在工具箱中选择 Brush Tool（刷子工具），选择合适的填充色，就可以在工作区进行绘制了。在如图 2-38 所示的 Options（选项）栏可以设定刷子的各种选项，包括对 Brush Mode（刷子模式）、Brush Shape（刷子形状）及 Brush Size（刷子大小）的设定。
- 刷子形状的设定：单击 Options（选项）栏中的 Brush Shape（刷子形状）下拉式列表框右边的倒三角箭头，会出现如图 2-39 所示的刷子形状列表以供选择，选择其中一种，此后刷子将以此为基本形状。
- 刷子大小的设定：单击 Options（选项）栏中的 Brush Size（刷子大小）下拉列表框右边的倒三角箭头，会出现如图 2-40 所示刷子大小列表以供选择，选中其中一种，此后刷子大小以此为依据。

图 2-37 选择刷子

图 2-38 选项栏

图 2-39 刷子形状

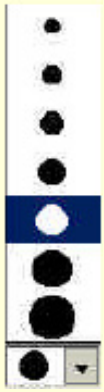


图 2-40 刷子大小

- **刷子模式的设定：单击 Options（选项）栏中的 Brush Mode（刷子模式）下拉列表框右边的倒三角箭头，会出现如图 2-41 所示的刷子模式列表以供选择，选中其中一种，此后刷子的模式以此为依据。**

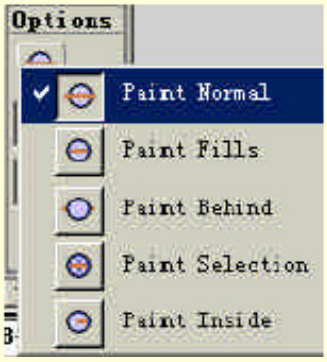


图 2-41 刷子模式

  - Paint Normal（标准喷涂）：如图 2-42 所示的标准喷涂模式可以对同一图层的线条或填充区域进行喷涂，喷涂后刷子扫过的区域将被覆盖。
  - Paint Fills（喷涂填充区域）：如图 2-43 所示的喷涂填充区域模式可以对填充区域或空白区域进行喷涂，但不会喷涂线条。
  - Paint Behind（喷涂后面）：如图 2-44 所示的喷涂后面模式可以对同一图层的空白区域进行喷涂，但不会喷涂线条及填充区域。
  - Paint Selection（喷涂所选区域）：如图 2-45 所示的喷涂所选模式可以对选中的区域进行喷涂，但不会影响线条及未被选中的区域。
  - Paint Inside（喷涂内部）：使用如图 2-46 所示的喷涂内部模式进行喷涂时，Flash 只对喷涂操作开始时所接触到的区域进行喷涂，对其他区域、线条及同一区域内刷子未扫过的区域无影响。

关于层的使用方法将在第 5 章中讲述。

图 2-42 标准喷涂

图 2-43 喷涂填充区域

图 2-44　喷涂后面　　图 2-45　喷涂所选区域　　图 2-46　喷涂内部

2.2.8　Eraser（橡皮）工具

橡皮工具可以擦除线条和填充区域，在执行擦除操作时可以选择只擦除线条、填充区域或被选中的线条、填充区域。

- Erase Normal（标准擦除）：如图 2-47 所示，选中 Eraser Tool（橡皮工具）及 Eraser Mode（擦除模式）中的 Erase Normal（标准擦除）选项，即可以如图 2-48 所示，拖动橡皮擦擦除工作区中任何线条或填充区域。
- Erase Fills（擦除填充区域）：如图 2-49 所示，选中 Erase Fills（擦除填充区域）选项，即可以拖动橡皮擦擦除工作区中任何填充区域，但保留轮廓线。用 Faucet（水龙头）方式可以快速擦除填充区域，只需选中它并在需要擦除的填充区域内单击鼠标即可，其结果如图 2-50 所示。
- Erase Lines（擦除线条）：如图 2-51 所示，选中 Erase Lines（擦除线条）选项，即可以拖动橡皮擦擦除工作区中的曲线或轮廓线，但保留填充区域。
- Erase Selected Fills（擦除选中的填充区域）：选择 Erase Selected Fills（擦除选中填充区域）选项，即可以拖动橡皮擦擦除工作区中被选中的填充区域，但未被选中的填充区域不会被擦除，其结果如图 2-52 所示。
- 内部擦除：如图 2-53 所示，内部擦除只擦除橡皮擦起始位置所在的填充区域内部，如果橡皮擦起始位置处无任何可擦除的对象，则擦除操作不起任何作用。

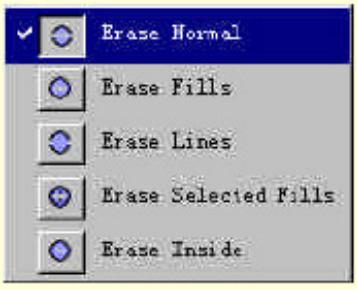


图 2-47　选取标准擦除

图 2-48　标准擦除结果

图 2-49　擦除填充区域

图 2-50　水龙头擦除结果

图 2-51 擦除线条

图 2-52 擦除选中的填充区域

图 2-53 内部擦除

2.3 线条及轮廓的修改

2.3.1 使用箭头工具修改造型

箭头工具可以用来选择工作区中的编辑对象并对其进行修改：对线条进行拉伸，对角度进行变化等。用箭头工具选择编辑对象：如图 2-54 所示，选中工具箱中的 Arrow Tool（箭头工具）。

- 用箭头工具选择填充区域：如图 2-55 所示，用鼠标单击填充区域内部，此区域内部即被选中，拖动被选中的区域时只有处于高亮状态的被选中区域被移动，而轮廓线不受影响。如果用鼠标双击填充区域内部，整个填充区域都被选中，包括轮廓线，如图 2-56 所示。此时若移动被选中的部分，则填充区域与轮廓线一同移动且相对位置不变。

图 2-54 箭头工具

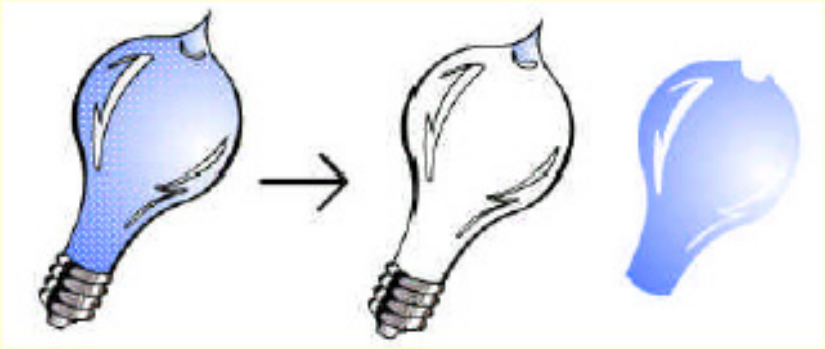


图 2-55 移动填充区

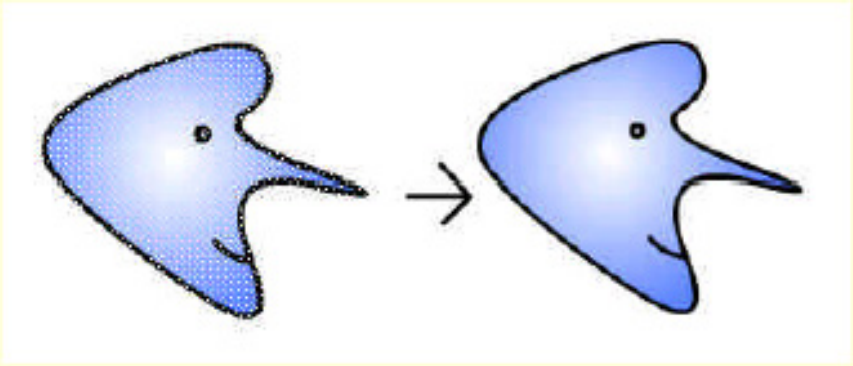


图 2-56 移动整个区域

- 用箭头工具选择线条：用箭头工具可以对单个线条或多个线条进行选择。用箭头点取某一线条时，此线条即被选中；若双击此线条，则与此线条相连的所有线条都被选中。如图 2-57 所示，线条被选中后可以对其进行移动、删除、复制等修改，也可对其属性作出必要的调整。
- 用箭头工具选择多个编辑对象：如图 2-58 所示，在用箭头工具进行选择时按住 Shift 键，此后

依次用鼠标点取的对象都将被选中，包括线条及填充区域。

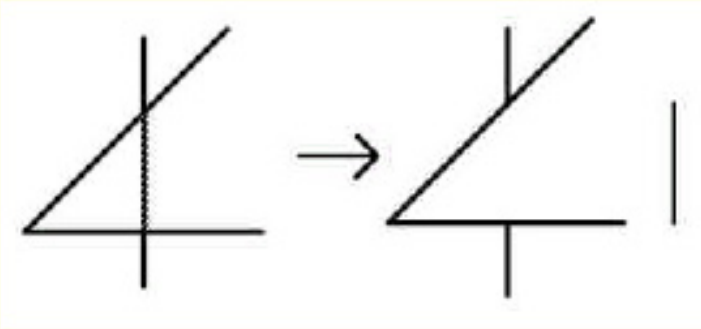


图 2-57　修改线条

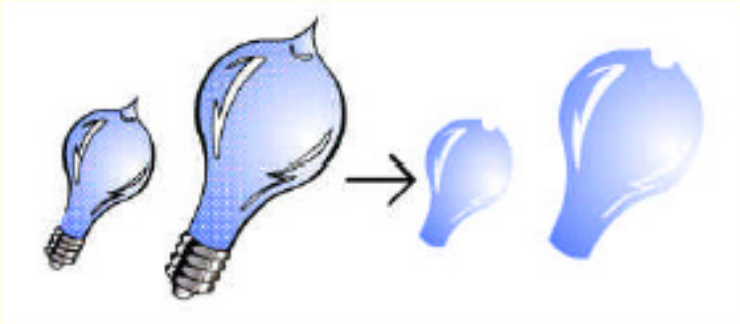


图 2-58　选择多个对象

- 用箭头工具选择某一区域内的对象：确定箭头工具处于被选中状态，如图 2-59 所示，按住左键拖动鼠标会出现一个矩形框，松开左键后，矩形框内的所有编辑对象都将被选中，包括线条与填充区域。

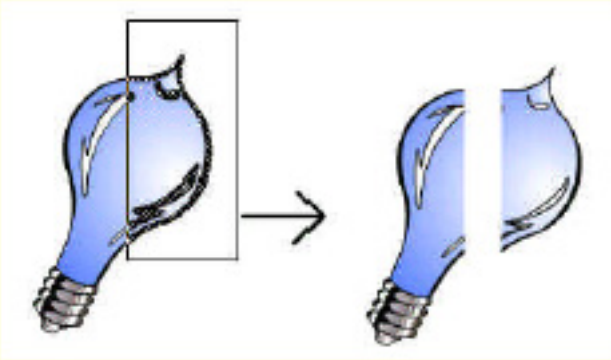


图 2-59　区域选中

2.3.2　箭头选项的设置

用箭头工具选中某一对象后，可以用 Options（选项）栏中的各种可用工具对其进行修改。

- 平滑 :先选中需要修改的对象,一般情况下是线条,然后单击如图 2-60 所示的选项栏中的 Smooth（平滑）工具，线条即被平滑，如图 2-61 所示，多次单击后曲线会逐渐变得平直。
- 锐化：先选中需要修改的对象，通常是线条，然后单击如图 2-62 所示选项栏中的 Straighten（锐化）工具，线条即被锐化。如图 2-63 所示，多次单击后曲线会逐渐变得棱角分明。

图 2-60 平滑工具

图 2-61 平滑结果

图 2-62 锐化工具

图 2-63 锐化结果

2.3.3 箭头工具修改线条的其他途径

- 将箭头移至线条端点时会出现一个如图 2-64 所示的直角标志，此时如图 2-65 所示，按住左键拖动鼠标可以将线条向任意方向拉伸，其结果如图 2-66 所示。

图 2-64 直角标志

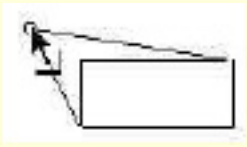


图 2-65 拉伸

图 2-66 修改结果

- 将箭头移至线条中间时会出现一个如图 2-67 所示的弧形标志，此时如图 2-68 所示，按住左键拖动鼠标可以将线条牵引变形。其结果如图 2-69 所示。

图 2-67 圆弧标志

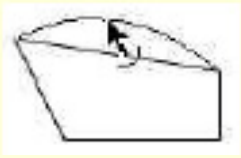


图 2-68 变形

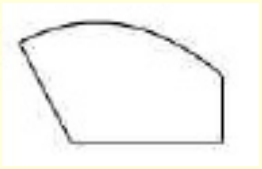


图 2-69 修改结果

- 鼠标拖动的是图形对象的两个控制点之间的部分，如果要改变图形对象的控制点的数目及位置，可以使用钢笔工具。
- 如图 2-70 所示，箭头工具还能以填充区域边界的任何一点为牵引点对轮廓进行曲化，填充区域形状随之变化，但其固有属性不变。

图 2-70 改变填充区域

2.3.4 自由变形工具

选中如图 2-71 所示的自由变形工具，可以对当前选中对象进行以下操作：

- 缩放：被选中的对象四周会出现 8 个小方块，调整这 8 个小方块即可以在水平、垂直、对角线 3 个方向对其进行整体缩放。
- 旋转与错切：鼠标箭头放在 4 个角上的小方块时会变成回转箭头形状，如图 2-73 所示，拖动鼠标可以使被编辑对象顺时针或逆时针旋转。把鼠标箭头放在其余四个小方块上时会变成双向箭头形状，如图 2-74 所示，拖动鼠标可以使被编辑对象在相应方向上发生错切变化。

图 2-71 自由变形工具

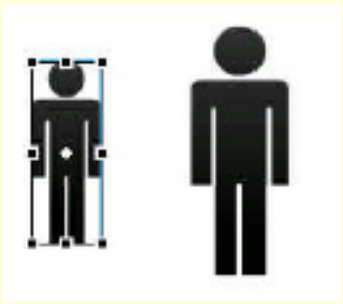


图 2-72 缩放结果

图 2-73 旋转结果

图 2-74 错切结果

2.3.5 曲线的优化

可以对曲线进行优化，选中需要优化的曲线，选择 Modify | Optimize，会弹出如图 2-75 所示的曲线优化对话框。其设定如下：

- 光滑滑块：拖动 Smoothing（光滑）滑块可以设定曲线光滑化的程度。
- 多重过滤：选中复选框中的 Use multiple passes（多重过滤）选项，可以按照设定值不断对曲线进行优化，直到不能再优化为止。其效果相当于对一个选中线条进行多次优化。
- 显示统计信息：选中 Show totals message（显示统计信息）复选框，如图 2-76 所示，可以在优化操作完成后显示摘要。图 2-76 中所显示的摘要为原有曲线 36 条，优化后曲线有 30 条，减少了 16%。

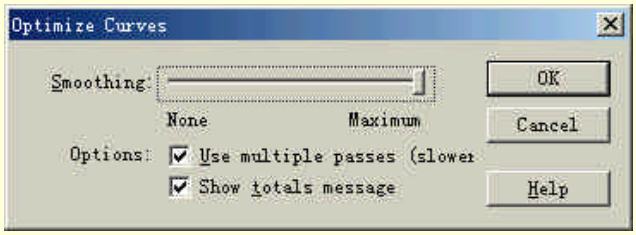


图 2-75 曲线优化对话框

图 2-76　显示摘要

2.4　磁铁工具的妙用

Flash 的磁铁工具可以自动完成线条、区域、网格点间的相互捕捉（捕捉即协调定位），用这一功能可以自动调整两物体间的相对位置。捕捉功能只有在箭头工具及磁铁工具都处于选中状态时才有效。当捕捉开始时，处于物体上的箭头附近会出现一个磁铁圆圈，将以此圆圈为参照点进行捕捉，此圆圈所处的位置大多是线段中点、线段端点、区域中心等。

- 线条间的相互捕捉：选中箭头工具及如图 2-77 所示的选项栏中的磁铁工具，选中一条线，然后在这条线的中点处按下左键，会出现一个小的磁力圆圈，如图 2-78 所示，将此线条从 1 位置拖至 2 位置后磁力圆圈会变大，表明 Flash 已经完成了捕捉，松开鼠标后线条会停在 2 位置，且中点在被捕捉的水平线上。如图 2-79 所示，用同样方法，可以完成对线条端点的捕捉。

图 2-77　磁力工具

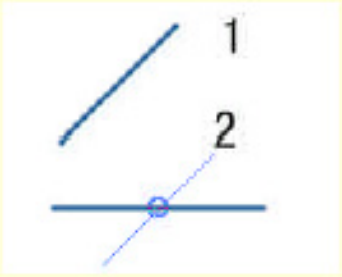


图 2-78　完成捕捉

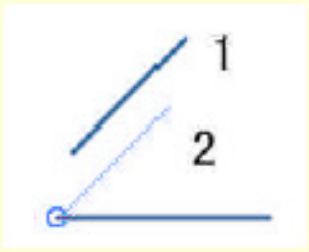


图 2-79　端点的捕捉

- 对填充区域中心的捕捉：如图 2-80 所示，用线条上的特殊点对填充区域中心进行捕捉，方法与上述线条间的相互捕捉类似，捕捉后捕捉点与区域中心重合。
- 对网格点的捕捉：如图 2-81 所示，选择 View | Grid | Show Gird（显示网格）菜单命令及 Snap to Grid（捕捉网格）菜单命令，如图 2-82 所示，选中三角形并用其直角顶点去捕捉网格点，完成后其顶点会处在被捕捉的网格点上。

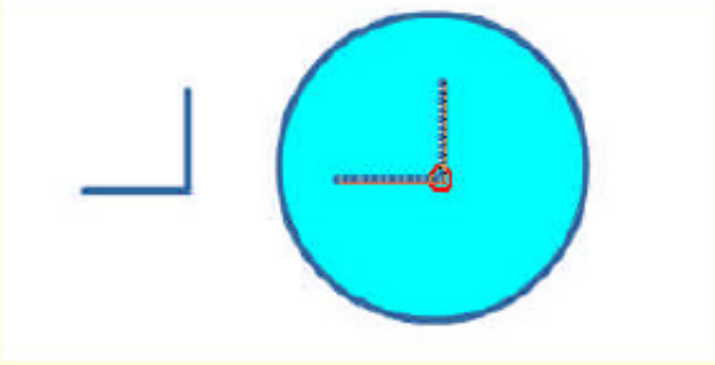


图 2-80　形心的捕捉

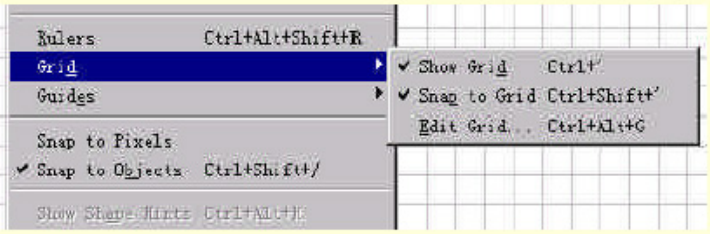


图 2-81 开启网格及捕捉

图 2-82 捕捉网格

2.5 视图工具

在创作中经常需要对舞台中的图形进行放大或缩小，这时就要用到 Flash 的视图工具了。

2.5.1 Zoom（缩放）工具

缩放工具有放大及缩小两个功能。选中缩放工具，在选项中选择 Enlarge（放大）或 Reduce（缩小）模式，在舞台中单击左键即可进行放大或缩小。

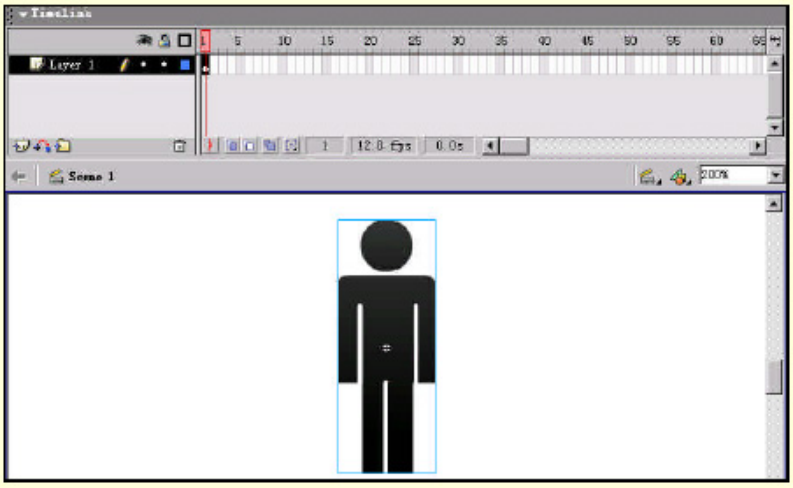


图 2-83 放大显示

- 放大以后的显示比例可以在窗口右上方的下拉列表框中看到。其实，直接在下拉列表框中选择比例或者输入比例也可以放大显示。

局部放大功能：用缩放工具可以对选中的局部区域进行放大。方法是按住左键拖动鼠标，会出现如图 2-84 所示的一个矩形框，Flash 将矩形框内的区域放大到整个舞台的范围。

- 无论在放大模式还是在缩小模式下，鼠标拖动形成的矩形区域都将被放大，不会被缩小。

图 2-84 局部放大

2.5.2 Hand（手形）工具

选中手形工具，此时舞台中的鼠标指针将变成手掌形状，用“手掌”可以拖动整个舞台内的图形。在拖动的同时，纵向滑块和横向滑块也随之移动。其实手形工具的作用也就相当于同时拖动纵向及横向滑块。

2.6 绘图参数的设定

通过设定绘图参数，可以在进行线条的平滑、锐化及运用磁力工具修改图形时产生不同的效果。选择 Edit | Preference 菜单命令，会弹出如图 2-85 所示的参数设定对话框，选择 Editing（编辑）标签。

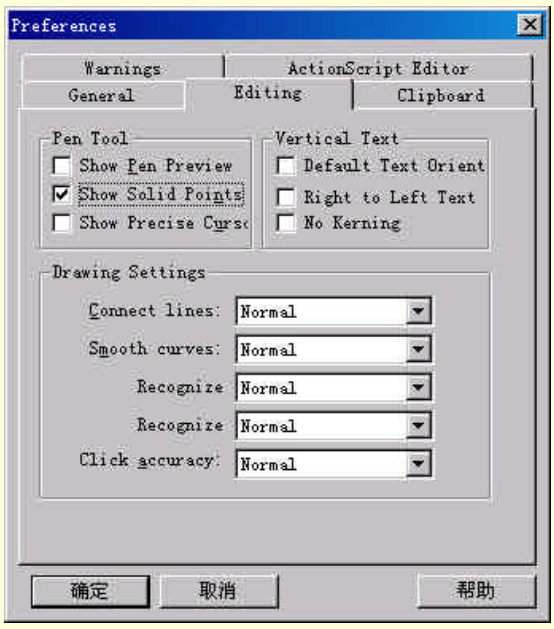


图 2-85 参数设定对话框

- 在 Pen Tool（钢笔工具）项中有 3 个复选框用于对钢笔绘图的设定。
  - Show Pen Preview（显示钢笔预览）：选中此复选框，用钢笔绘图时会有一条细曲线跟随“钢笔”移动，用户可以由此预先看到将要绘出的曲线的形状，并根据预览曲线的形状进行相应的调整。
  - Show Solid Points（显示固体点）：选中此复选框以后，用钢笔绘图时，被选中的点以中空方式显示，未被选中的点以实心方式显示；若未选中此复选框，被选中的点以实心方式显示，未被选中的点以中空方式显示。
  - Show Precise Cursors（显示精确光标）：选中此复选框后，工作区中的钢笔形光标将被斜十

字形光标所取代，以显示精确位置。

- 在 Vertical Text（竖排文本）项中有 3 个复选框用于对竖排文本的设定。
  - Default Text Orient（默认文本方向）：选中此复选框后，选择 Text（文本）工具时，文本将默认为以竖直方向排列，否则将以水平方向排列。这个选项主要是为了方便亚洲某些国家和地区（如日本和台湾）的文字编排习惯而设的。
  - Right to Left Text（自右向左排列文本）：选中此复选框后，当按水平方向排列文本时，将按从右向左的方向进行。
  - No Kerning（取消字距调整）：选中此复选框将会取消对竖排文本的字距调整。
- Drawing Settings（绘画设定）：包括 5 个下拉式列表框。
  - Connect lines（连接线）：用于设定线的捕捉范围，包括 Must Be Close（贴紧）、Normal（标准）、Can Be Distant（松散）三个选项，范围依次扩大，如选择贴紧，则只有两线条相互贴紧时捕捉才有效；若选择松散，则两线条相距一定距离时也能完成捕捉。
  - Smooth curves（平滑曲线）：用于设定曲线的平滑度，在使用铅笔工具绘图时可以看到，选择 Off（关闭）、Rough（粗糙）、Normal（标准）及 Smooth（平滑）选项时，线的平滑或锐化程度是不同的。
  - Recognize Lines（线识别）：用于设置对线条形状的识别精度，有 Off（关闭）、Strict（严格）、Normal（标准）、Tolerant（宽松）四个选项，在采用严格识别时，Flash 对铅笔工具绘出的线条作细微锐化；在宽松状态下锐化最为明显；若选择关闭，则线条的平滑效果会比较明显，此后若需对线条进行锐化，可选中此线条，再用 Modify（修改）| Straighten（锐化）菜单命令对其进行处理。
  - Recognize Shapes（图形识别）：用于设置对图形形状的识别精度，有 Off（关闭）、Strict（严格）、Normal（标准）、Tolerant（宽松）四个选项，这些选项作用与线识别的基本相同，读者只要上机一试便知。

图形识别中所指的图形包括圆、椭圆、正方形、长方形，直角、平角等。或许是软件设计的原因，线识别与图形识别两个列表框的名字均为 Recognize，实际上上面的 Recognize 代表线识别，下面的 Recognize 代表图形识别。

- Click Accuracy（单击精确度）：用于设置对编辑对象的识别范围，有 Strict（严格）、Normal（标准）、Tolerant（宽松）三个选项。如图 2-86 所示，在严格的精确度状态下，箭头尖端必须落在被选物体上才能将其选中；在宽松的精确度状态下，如图 2-87 所示，箭头尖端不必落在被选物体上也可将物体选中。

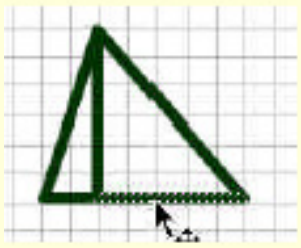


图 2-86　严格的精确度

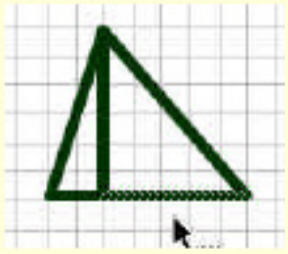


图 2-87　宽松的精确度

第 3 章　颜色的使用

3.1　综　述

Flash MX 为颜色的使用、生成及修改提供了多种方法，使用默认的调色板和自定义调色板可以设定填充色及描绘色。用描绘色描绘图形的轮廓线，用填充色填充内部区域。在描绘轮廓线时可以使用任何一种纯色，并可选择不同的线形及粗细。内部区域的填充可以用纯色、渐变色、位图等。在使用位图填充时必须在当前文件中导入位图文件。

3.2　描绘和填充

用工具箱中的描绘工具与填充工具可以方便快捷地对描绘色及填充色进行修改。

3.2.1　墨水瓶工具的使用

选择工具箱中的 Ink Bottle Tool（墨水瓶工具），如图 3-1 所示，利用颜色描绘面板设定所需的描绘色，选择合适的线型，然后单击舞台中需要修改的线条，此线条的线型及颜色便会改变，效果如图 3-2 所示。

图 3-1　墨水瓶工具

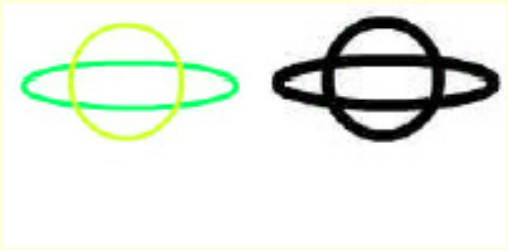


图 3-2　墨水瓶工具的应用

3.2.2　油漆桶工具的使用

油漆桶工具可以对封闭填充区域或不完全封闭区域进行填充，可以用纯色、渐变色或导入的位图图像进行填充，在填充时可以对渐变色及位图图像的填充宽度、填充方向及填充中心进行设定。通常的方法是选择工具箱中的 Paint Bucket Tool（油漆桶工具），如图 3-3 所示，选择合适的填充色，在需要填充的区域单击鼠标左键即可完成填充操作。

油漆桶选项的设置：

- Gap size（空隙大小）：在图 3-4 所示的选项栏中可以选择 Gap Size（空隙大小），以控制对不完全封闭区域进行填充时的效果。选择 Close Large Gaps（封闭大空隙）可以在区域不完全封闭

的情况下完成填充，与 Close Small Gaps（封闭小空隙）相比，此项所能封闭的缺口较大。Don't Close Gaps（不封闭空隙）需要手动将空隙封闭才能进行填充，即使空隙很小也需如此操作。

图 3-3 油漆桶工具

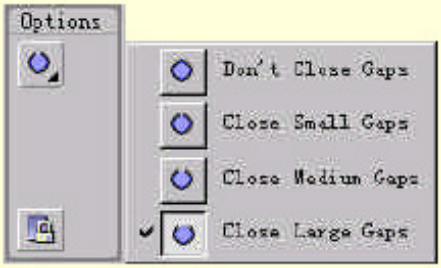


图 3-4 空隙大小

- 虽然有封闭大空隙的选项，但对于较大的空隙还需要手动封闭后再填充。
- Lock Fill（锁定填充）：此项可以锁定渐变色与位图图形的填充模式，按照拾取的模式进行填充，填充方向、填充宽度保持不变。

3.2.3 点滴器的使用

Eyedropper（点滴器）用于拾取描绘色、填充色、位图图形等，在拾取描绘色后点滴器自动变成墨水瓶工具，在拾取填充色或位图图形后自动变成油漆桶工具。拾取方法十分简单，选中此工具后在希望拾取的颜色或图形上单击左键，然后在需要修改的线条或区域中单击左键即可将拾取的颜色或位图复制过来。

- 在拾取位图图形时必须将位图图形打碎，方法是选中将要拾取的位图图形，选择 Modify（修改）| Break Apart（打碎）菜单命令。

3.3 颜色面板的使用

3.3.1 信息面板

选择 Window | Info 菜单命令，会弹出图 3-5 所示的信息面板，在此面板中可以看到鼠标所在点或区域处的图形信息。RGB 代表鼠标所在点处的颜色组成，R（Red）代表红色、G（Green）代表绿色、B（Blue）代表蓝色；旁边的 X、Y 值代表鼠标所在点处的坐标（舞台左上角点为原点）。如果鼠标所处位置为选中区域的内部，则 Shape（形状）区中的四个文本框将处于可填写状态，W 和 H 分别代表填充区域外接矩形的宽与高，修改文本框中的数字可以改变外接矩形的形状，进而改变整个区域在 X、Y 方向的尺寸；右边的 X、Y 值代表区域的外接矩形中心点或左上角点在舞台中的坐标，具体代表哪一个点要看小图表中哪个点处于黑色选中状态，修改文本框中的数字可以改变选中点的坐标，进而改变整个区域在舞台中的位置，效果如图 3-6 所示。

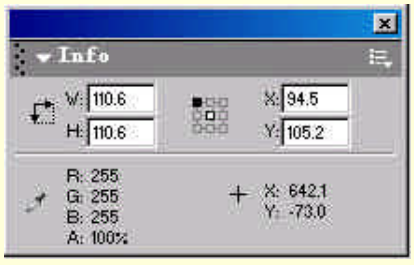


图 3-5 信息面板

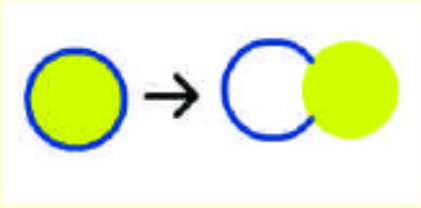


图 3-6 改变选中点的坐标

3.3.2 图形的变形

选择 Window | Transform 菜单命令会弹出如图 3-7 所示的 Transform（图形转换）面板。

- Constrain（强制转换）：选择强制转换复选框，在左边的文本框中选择合适的百分比，Flash 会自动调整填充区域的大小，并且根据用户对长（宽）的修改自动锁定宽（长）的百分比。当然，取消强制转换可以对长和宽分别进行调整。
- Rotate（旋转）：选中旋转单选钮，在右边的文本框中键入合适的角度，按回车键即可对选中区域进行相应角度的旋转，效果如图 3-8 所示。

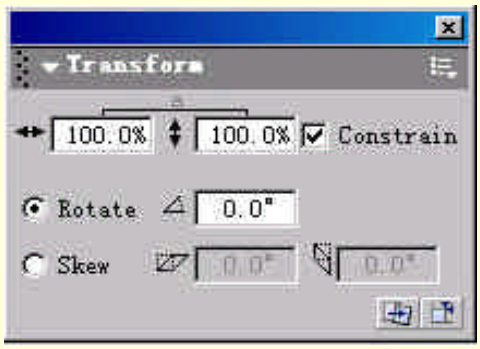


图 3-7 图形转换面板

图 3-8 图形的旋转

- Skew（错切）：选中此单选钮并在 Skew Horizontally（水平错切）及 Skew Vertically（竖直错切）文本框中添入所需的角度并按回车键，Flash 会自动对选中区域完成错切变换，效果如图 3-9 和图 3-10 所示。
- Copy and apply transform（复制并应用转换）：此按钮用于复制上一步的变换操作并进行重复。例如，对一个填充区域进行旋转操作后按下“复制并应用转换”按钮，Flash 会复制这一步转换操作，当再一次按下此按钮后，Flash 会在当前状态下重复所复制的转换操作。
- Reset（复位）：用于取消复制并应用转换的作用。

图 3-9 水平错切

图 3-10 竖直错切

选择描绘标签会弹出 Stroke（线条描绘）面板。此面板已在第 2 章作过介绍，这里不再赘述。

3.3.3 神奇的填充

选择 Window | Color Mixer 菜单命令会弹出图 3-11 所示 Color Mixer 面板。在下拉式列表框中有五个可选选项：None（无色）、Solid（实线）、Linear Gradient（线渐变）、Radial Gradient（径向渐变）、Bitmap（位图）。

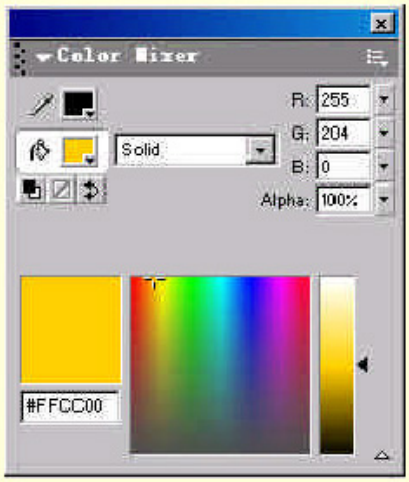


图 3-11　Color Mixer 面板

- None（无色）：当选中此项时，填充色为无色，绘出的图形只有轮廓线。
- Solid（实线)：此项用于简单的设置填充色，只能选择各种纯色，无法使用填充色。
- Linear Gradient（线渐变）：用于定义直线渐变色，在渐变彩带下方有若干个锥形滑块，分别代表若干个渐变关键色，滑块被选中时尖端会变成黑色，在右边的调色板中可以选择此滑块所控制的关键色。拖动滑块还可以改变关键色的渐变分布情况。单击右下方的磁盘按钮可以将当前的渐变色添加到填充色面板中。在渐变彩带左边有一个渐变预览窗口，可以看到最终渐变效果。如果滑块超出所需数目需要删除，只要用鼠标拖住要删除的滑块向下拉开一段距离即可；在两滑块间任意位置单击左键都可插入新的滑块。由于颜色是渐变模式，因此滑块数目不能少于两个。
- Radial Gradient（径向渐变）：用于定义径向渐变色，如图 3-12 所示，定义方法类似于线渐变，读者只需上机一试便知。
- Bitmap（位图）：用于定义位图填充，在列表框下面的窗口中列出了已经导入的位图图形微缩预览，如图 3-13 所示，选中所需的位图即可用此位图进行填充。

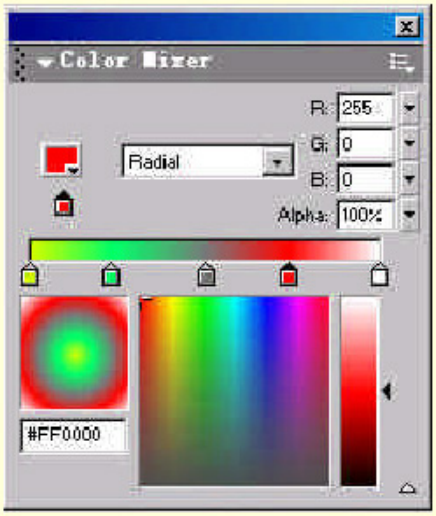


图 3-12　径向渐变色

图 3-13　位图填充

✍ 下面请读者参照范例作一朵七彩渐变花。

（1）编辑渐变色。在 Color Mixer 面板中选择径向渐变，并在渐变彩带中设置渐变色，使彩带从左到右的渐变对应于花蕊至花瓣的色彩渐变。记住要将渐变彩带最右边的渐变关键色设置为白色，如图 3-14 所示。

图 3-14　设置渐变色

（2）绘制一个花瓣。在工具箱中选择椭圆工具，在描绘色面板中选择刚才所编辑的渐变色，并将描绘色设成无色，在舞台中绘制一个椭圆形的花瓣，效果如图 3-15 所示。

（3）绘制整个花瓣。进入变换面板，选中上一步中画出的花瓣，Rotate 在（旋转）文本框中键入 45 ° 并按回车，连续按 Copy and apply transform（复制并应用转换）按钮，花瓣会不断旋转成型，效果如图 3-16 所示。

（4）当花瓣完全成型后按 Reset（复位）按钮并取消对花瓣的选择，最终效果如图 3-17 所示。

（5）绘制花萼。用类似的方法绘制四片花萼，注意颜色与花瓣的搭配，如图 3-18 及图 3-19 所示。

（6）最终成形。选中花瓣，并利用 Flash 的捕捉功能将花瓣按图 3-20 所示的方式重叠至花萼上方，取消选择后即可看到最终的成型效果，如图 3-21 所示。

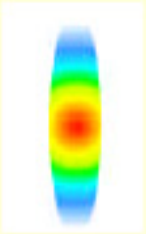


图 3-15　绘制一个花瓣

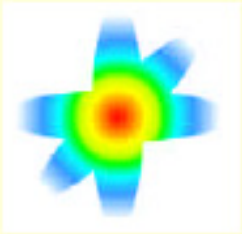


图 3-16　复制并应用转换

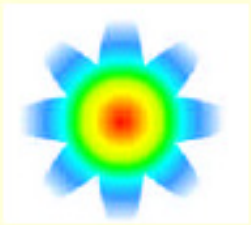


图 3-17　花朵成形

图 3-18　绘制一片花萼

图 3-19　花萼成形

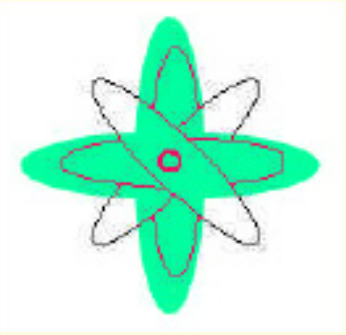


图 3-20　花瓣与花萼重叠

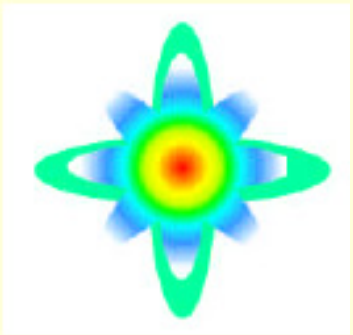


图 3-21　最终效果

3.4　编辑调色板

3.4.1　调色板的使用

可以说，Color Mixer 面板是 Flash 的颜色控制中心，它提供了两种颜色修改方案（RGB 方案和 HSB 方案）及选择方式（直观选择和量化选择）。一个动画如果没有适当的颜色就无法称为优秀的动画。所以，熟练、恰当地运用调色板是做好 Flash 动画的前提。选择 Window | Color Mixer 菜单命令将弹出图 3-22 所示的调色面板。

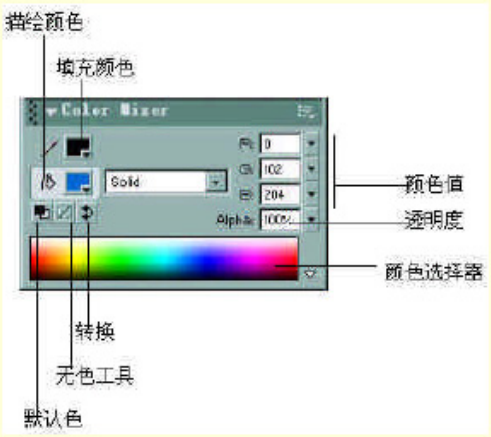


图 3-22　调色面板

调色板的使用：选择调色面板中的 Mixer（调色板标签）。

- Stroke（描绘）与 Fill（填充）：单击 Stroke color（描绘）或 Fill color（填充）图标会弹出相应的控制面板，在其中可以选择所需的描绘色与填充色。描绘色面板与填充色面板的用法将在 3.4.2、3.4.3 节中予以介绍。
- Color bar（颜色选择器）：在颜色选择器中可以选择不同的描绘色或填充色，所选的颜色用于描绘或是填充要看哪一个图标处于选中状态，图中的描绘图标处于选中状态。
- 快捷工具：No stroke/fill（无色工具）用于设定使用无色描绘或无色填充。Default stroke and fill（默认色）工具可以将描绘色与填充色分别设成黑色与白色。Swap stroke and fill colors（转换）工具能将当前的描绘色与填充色互换，如果当前的填充色为渐变色，那么 Flash 会将渐变色的关键色与描绘色互换。

当使用无色工具时，在舞台中绘出的图形将只能分辨出轮廓线，填充色与舞台背景色相同，因此无法分辨，这与填充色为无色时的情形相同但性质不同：填充色为无色时轮廓线所包

围的部分没有区域的概念，用箭头工具只能选择轮廓线，无法选择轮廓线包围的部分；填充色为白色时，用箭头工具可以选择轮廓线与轮廓线所包围的区域。涉及到描绘色时情况相同。

- 颜色参数：量化显示当前色的基本组成，即红（R）、绿（G）、蓝（B）的量化值，以及当前色的透明度。修改文本框中的数值或拖动滑块均可对当前色作出调整。

如果当前色为渐变填充色，颜色参数的文本框中将不显示任何参数。

- 颜色方案的选择：单击右上角的箭头会弹出一个菜单以供选择不同的色彩方案，包括 RGB 和 HSB 两种方案，如图 3-23 所示。
- Swatches（样本色）：选择 Windows | Color Swatches 菜单命令，将会弹出如图 3-24 所示的 Color Swatches（样本色）面板。样本色面板中以渐变方式存储各种纯色，此外还存储 Flash 默认的渐变色及用户自行定义的渐变色。

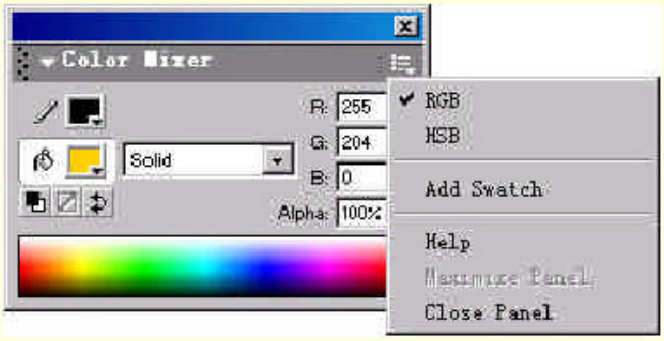


图 3-23 色彩方案

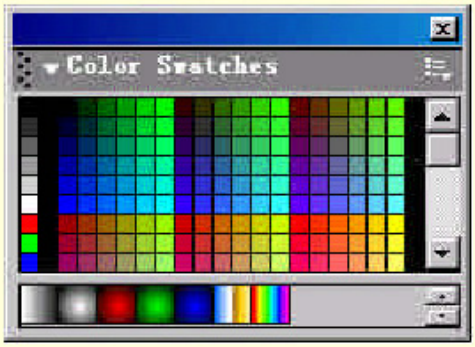


图 3-24 样本色面板

3.4.2 描绘色面板

单击图 3-25 所示的描绘色图标会弹出图 3-26 所示的描绘色面板。

在 Hex Edit text box（十六进制数值框）中填入相应数值，或在 Solid colors（纯色）栏中进行选择，均可得到所需的纯色。

单击右上角的 Color picker button（颜色选择器按钮），将会弹出颜色选择器面板，其作用将在 3.4.3 节中介绍。若在工具栏中选中 None button（无色按钮），那么绘出的图形只有填充色而无轮廓线。

图 3-25 描绘色图标

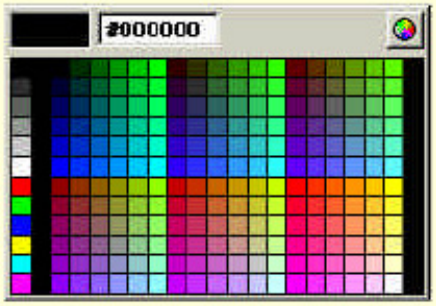


图 3-26 描绘色面板

3.4.3 填充色面板

单击填充色图标会弹出图 3-27 所示的填充色面板，在 Hex Edit text box（十六进制数值框）中填入相应数值或在 Solid colors（纯色）栏中进行选择，均可得到所需的纯色。Gradient swatches（渐变色样本）所提供的渐变色只适用于填充色，描绘色无法使用渐变效果。使用 None button（无色按钮）可以使填充色变为透明，这样绘出的图形只有轮廓线可见而内部区域透明。

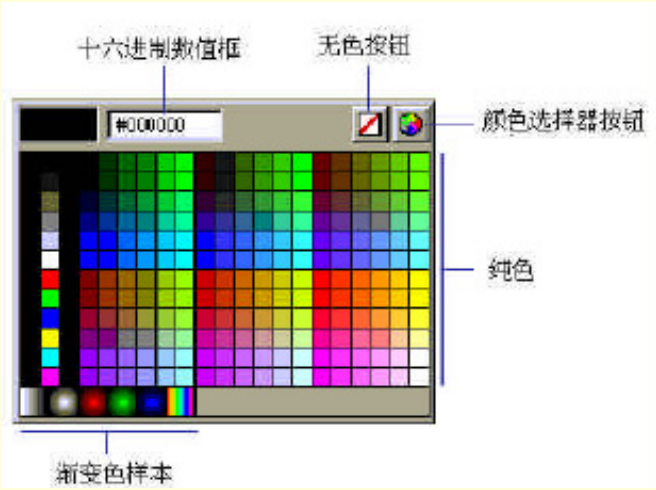


图 3-27 填充色面板

单击 Color picker button（颜色选择器按钮）会弹出如图 3-28 所示的颜色选择器面板，此面板为 Windows 内置的颜色选择器面板。

颜色选择器面板的使用：

- 添加自定义颜色：选中一个自定义颜色框用于存储自定义颜色，然后选取或自行设置一种颜色，并单击“添加到自定义颜色”按钮，自定义颜色框中会存储此自定义颜色以备用。
- 基本颜色：此框中存有 48 种基本色供选择，用户可以根据需要对一种选中的基本色的亮度以及红、绿、蓝成分进行调整。
- 颜色渐变区：在此渐变区内可以通过十字准星选择合适的颜色，拖动渐变区右边的亮度滑块可以对选中颜色的亮度进行调整。

图 3-28 颜色选择器面板

- 颜色量化控制文本框：文本框中的数据表示当前颜色的各种属性，包括色调、饱和度、亮度及红绿蓝组成，可以对文本框中的数据进行修改以获得新的颜色。读者应当了解：任何一种混合色（除红、绿、蓝以外的色）都可看作由三原色（红、绿、蓝）以一定比例混合而成的，如三原色等量相加可得到黑色。

3.4.4 调色板的导入与导出

利用调色板的这一功能可以导入与导出 RGB 颜色与渐变色，用 Color Table 文件（Color Table 为文件类型）导出的 RGB 颜色与渐变色能被 Macromedia Fireworks 和 Adobe Photoshop 所使用。

单击样本色面板右上角的箭头会弹出导入导出菜单，用它即可控制颜色的导入导出。各选项用法如下。

- Delete Swatch（删除样本色）：用于删除选定的样本色。
- Load Default Colors（加载默认颜色）：用于恢复调色板的默认设置。
- Replace Colors（替换颜色）：可以用文件中的颜色或颜色信息替换选中的样本色。用法及作用类似于加载颜色。
- Duplicate Swatch（复制样本色）：选中一种样本色并选择复制样本色，被选中的样本色将被复制到图 3-29 所示的样本色面板中备用。可以将绘图常用的颜色进行复制，以减少查找时间。

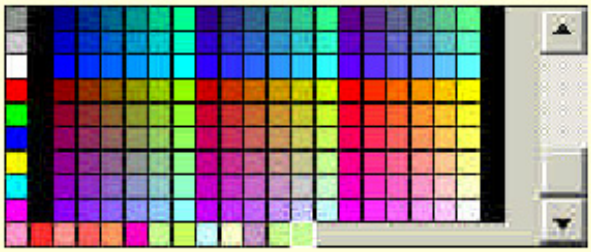


图 3-29 样本色面板

- Add Colors（加载颜色）：用于加载来自文件的颜色。选中后会弹出图 3-30 所示的 Import Color Swatch（样本色导入）对话框。

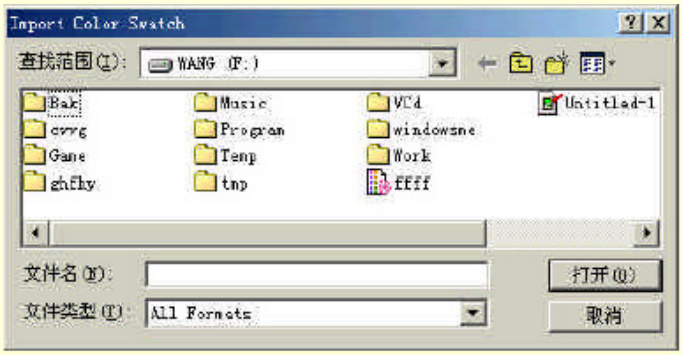


图 3-30 样本色导入对话框

导入对象可以是位图文件或 Flash Color Table 类型文件。如果导入对象是位图文件，那么加载到调色板中的将是此位图图形的所有颜色，如图 3-31 所示。

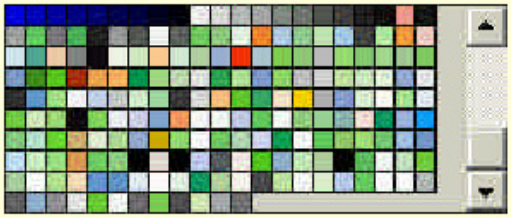


图 3-31　导入位图文件的颜色

- Save as Default（保存为默认色）：可以将当前样本色面板中的颜色保存为默认色。在下一次加载默认颜色时会将此次保存的颜色调入。选择此项后会弹出一个对话框询问是否这样做，因为这将改变原始设置，初学者应慎用。
- 216 Web（网络颜色）：调用 216 种网络颜色。
- Save Colors（保存颜色）：可以将调色板的当前颜色信息存储为 Flash Color Table 类型文件。选中后会弹出图 3-32 所示的样本色导出对话框，导出的文件可以作为 Flash 的导入文件使用，也可以被其他软件导入。

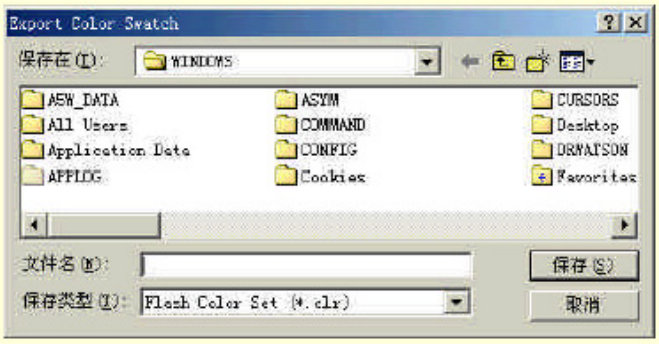


图 3-32　样本色导出对话框

- Clear Colors（清除颜色）：选择此项将清除调色板中的颜色，效果如图 3-33 所示。

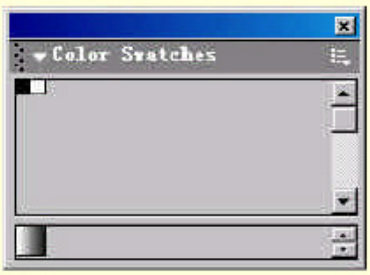


图 3-33　清除调色板中的颜色

- Sort By Color（颜色排序）：将颜色按顺序排列，如图 3-34 所示。

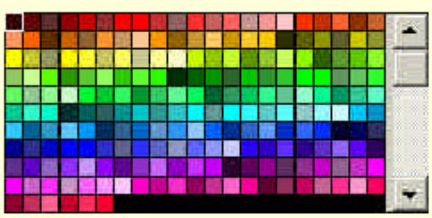


图 3-34　颜色排序

第 4 章　简单的 Flash MX 动画

4.1　综　述

在前面的章节中介绍了 Flash MX 的绘图功能，从本章开始将详细讲述 Flash MX 的动画制作过程。渐变是制作 Flash 动画的简单而又灵活的方法，本章将主要介绍 Flash MX 中两种重要的渐变形式——位置渐变和图形渐变。

4.2　帧的引入

4.2.1　帧的概念

所谓动画，就是一系列相似的图画快速地逐帧显示出来。因为人的眼睛有 0.1 秒的视觉暂留，所以看上去就像是物体运动起来了。其实，现在的电影就是以每秒 24 张的速率放映出来的图片，这每一张图片就叫作一“帧”。在 Flash MX 中，帧的概念与电影中的一样，就是一张静止的图片。“帧”是组成动画的基本单位，理解帧的概念并使用好“帧”是成功创作 Flash 动画的关键。

在 Flash MX 中创建动画有两种方法。

- 逐帧制作：像制作电影一样一帧一帧地制作动画的每一张图片。显然这种方法工作量很大，在制作时间较长的动画时，根本不现实。除非需要从外部导入其他格式的动画文件，Flash MX 才把它转化为逐帧的动画。
- 利用 Tweening（帧渐变）：Flash MX 中提供了丰富的帧渐变形式。在使用时，只需要给出首帧、末帧以及渐变形式，就可以实现很复杂的动画。比如：物体的放大、缩小、色彩渐变、移动、转动、形体变化等。具体的功能及实现方法将在本章中进行详细的说明。

4.2.2　帧的显示模式

利用如图 4-1 所示的 Timeline（时间轴）可以方便地对帧进行编辑。

时间轴上的每一个小格代表一帧。时间轴包括如下控制项。

- 帧数：当前动画所处的帧数。
- 帧频：播放动画时每秒钟刷新的帧数。默认值为 12 帧/秒。在帧频栏上双击或者选择 Modify | Document 菜单命令（快捷键 Ctrl+M），在弹出的如图 4-2 所示的 Document Properties（电影属性）对话框中可以设置帧频以及电影文件的其他属性。
  - Frame Rate（帧频）
  - Dimensions（电影尺寸）：Width（宽度）、Height（高度）

图 4-1　Flash MX 的时间轴

- Background（背景颜色）
- Ruler Units（尺寸单位）：包括：Inch（英寸）、十进制英寸、Points（点）、Centimeters（厘米）、Millimeters（毫米）、Pixels（像素）。

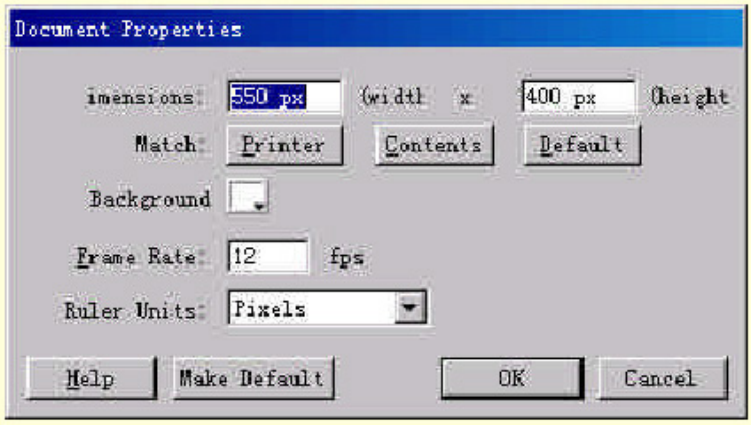


图 4-2　电影文件的属性

- 时间：从第一帧播放到当前帧所需要的时间。
- 帧工具（如图 4-3 所示）。

图 4-3　帧工具

- Center Frame（帧居中）：使选中的帧居中显示。
- Onion Skin（洋葱皮）模式：如图 4-4 所示的洋葱皮模式可以以不同的透明度显示出当前帧以及附近的帧。

图 4-4　洋葱皮模式显示

- Onion Skin Outline（洋葱皮轮廓）模式：如图 4-5 所示的洋葱皮轮廓模式可以以不同的透明度显示出当前帧附近的帧的轮廓。

图 4-5　洋葱皮轮廓模式显示

- Edit Multiple Frames（多帧编辑）：可以同时对当前帧及其附近的帧进行编辑。
- Modify Onion Markers（洋葱皮参数设定）：Always Show Markers（开启或隐藏洋葱皮显示范围）、Anchor Onion（固定洋葱皮显示范围）、设定洋葱皮显示范围（Onion 2，Onion 5，Onion All）。

洋葱皮对于制作动画有很大帮助，它可以使帧与帧之间的位置关系一目了然。在设置洋葱皮的作用范围时，也可以按照图 4-6 所示的方法用鼠标直接拖动。

图 4-6 拖动鼠标设置洋葱皮范围

- 时间轴模式：单击时间轴模式设定按钮，将弹出如图 4-7 所示的菜单。其功能如下。

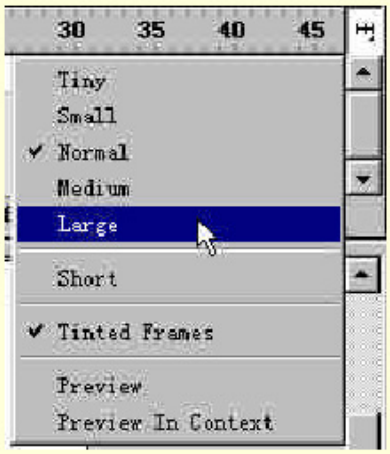


图 4-7 设定时间轴模式

  - 帧的大小：设定时间轴所显示的每一帧的大小。其中包括 Tiny（很小）、Small（小）、Normal（正常）、Medium（中）和 Large（大）。
  - Short（短帧显示）：缩短时间轴上帧的高度。
  - Tinted Frames（帧着色）：以彩色模式显示帧，以便区分不同类型的帧。
  - Preview（预览帧）：在时间轴上，显示每一帧的缩略图并自动放大。
  - Preview In Context（整体预览帧）：在时间轴上显示每一帧的整体缩略图。

4.2.3 帧的类型

在 Flash MX 的时间轴上，不同的帧有不同的表示方法：

- Blank Frame（空白帧）：帧中不包含任何 Flash 对象，相当于一张空白的影片（如图 4-8 所示）。

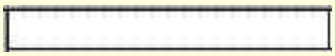


图 4-8 空白帧

- Key Frame（关键帧）：圆点表示一个 Flash MX 关键帧的开始，空心矩形表示一个 Flash MX 关键帧的结束，中间用灰色填充表示一个静止的 Flash MX 关键帧（如图 4-9 所示）。

图 4-9 关键帧

- Motion-tweened（位置渐变帧）：在两个关键帧之间用浅蓝色填充并由箭头连接的帧（如图 4-10 所示）。关于位置渐变的使用方法将在 4.3 节中说明。

图 4-10 位置渐变帧

- Shape-tweened（图形渐变帧）：在两个关键帧之间用浅绿色填充并由箭头连接的帧（如图 4-11 所示）。关于图形渐变帧的使用方法将在 4.4 节中说明。

图 4-11　图形渐变帧

- 不可渐变帧：在两个关键帧之间用浅蓝色填充并由虚线连接的帧（如图 4-12 所示）。它表示帧的位置渐变不符合要求或者没有与其相连的下一关键帧。

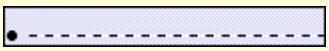


图 4-12　不可渐变帧

- Action（动作帧）：在帧中显示一个字母“ α ”（如图 4-13 所示），它表示此处有帧的动作——Flash MX 的 ActionScript 脚本程序。关于脚本程序的使用将在后续章节中进行介绍。

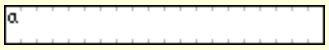


图 4-13　动作帧

- label、comment（帧标签）：用一个小红旗开头后面标注有文字的帧表示帧的标签或者注释，可以将其理解为帧的名字（如图 4-14 所示）。

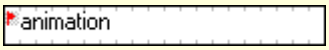


图 4-14　帧标签

4.2.4　帧的编辑

Flash MX 提供了强大的帧编辑功能。在时间轴上右击鼠标，在弹出的如图 4-15 所示的快捷菜单上列举了各种编辑帧的方法。

图 4-15　帧快捷菜单

菜单中与帧编辑有关的功能介绍如下：

- Insert Frame（插入帧）：在当前位置插入帧，其作用在于加长关键帧的作用时间。

插入帧的数目和选中帧的数目相同。

- Remove Frame（删除帧）：删除被选中的帧。

- Insert Keyframe（插入关键帧）：把前一个关键帧复制到当前位置，并将前一关键帧的作用时间延长至当前位置之前，如图 4-16 和图 4-17 所示。

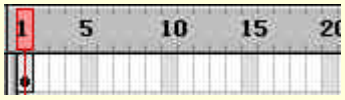


图 4-16　插入关键帧之前

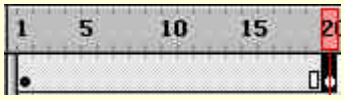


图 4-17　插入关键帧之后

- Clear Keyframe（删除关键帧）：删除被选中的关键帧。
- Insert Blank Keyframe（插入空白帧）：如图 4-18 和图 4-19 所示，在当前位置创建空白帧，并将前一关键帧的作用时间延长至当前位置之前。

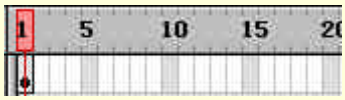


图 4-18　插入空白帧之前

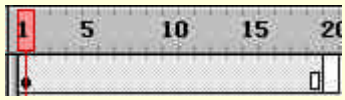


图 4-19　插入空白帧之后

- Select All（全选帧）：选中所有层上面的所有帧。
- Cut Frame（剪切帧）：剪切被选中的帧。
- Copy Frame（复制帧）：复制被选中的帧。
- Paste Frame（粘贴帧）：将被剪切或被复制的帧粘贴到当前位置。
- Reverse Frame（翻转帧）：如图 4-20 和图 4-21 所示，将所选中的帧的顺序翻转，使动画反向播放。

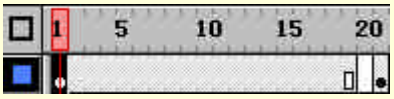


图 4-20　翻转帧之前

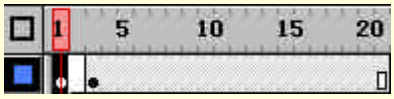


图 4-21　翻转帧之后

✍ 将鼠标移动到时间轴的某一关键帧上，鼠标指针末尾会加上如图 4-22 所示的小方框。按照图 4-23 所示的方法直接用鼠标拖动即可将该帧移动到相应的位置。此外，如图 4-24 所示，配合 Alt 键拖动也可以实现帧的复制。

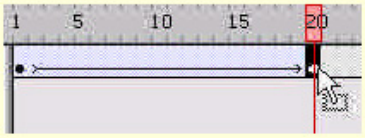


图 4-22　鼠标指针变为手形

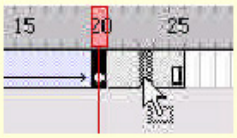


图 4-23　拖动帧

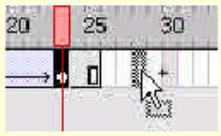


图 4-24　复制帧

4.3　功能强大的位移运动

4.3.1　直线运动的物体

利用 Flash MX 中提供的图形位移功能，只需要几个步骤就可以实现 Flash MX 对象的直线运动。具体步骤如下：

（1）选择 File | New 菜单命令新建一个 Flash MX 文件，选择椭圆工具，Stroke（描绘颜色）使用黑色，Fill（填充颜色）使用径向的黑白渐变。选中 Snap to Object（抓握）工具。在 Scene（场景）的第一帧上绘制如图 4-25 所示的圆球。

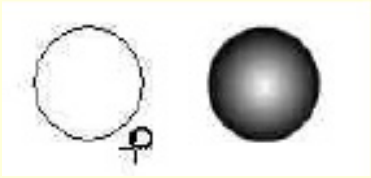


图 4-25　拖动并绘制一个圆球

（2）在 Timeline（时间轴）的第一帧上单击鼠标，这时如图 4-26 所示，第一帧上的所有对象处于选中状态。选择 Modify | Group 菜单命令或者使用快捷键 Ctrl+G 将第一帧上所选中的对象组合成一个组。这时，处于选中状态的圆球将被如图 4-27 所示的外框围住，它表示 Flash MX 将把整个圆球作为一个图像来处理。

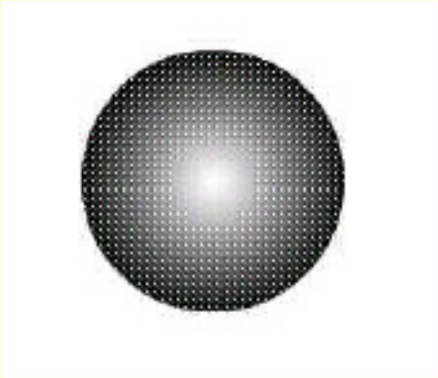


图 4-26　处于选中状态的圆球

图 4-27　组合以后的圆球

- 如果需要取消组合，可以选择 Modify | Ungroup 菜单命令或者使用快捷键 Ctrl+Shift+G。若要编辑组内的对象，则可以先选中组，再选择 Edit | Edit Selected 菜单命令或者用 Arrow Tool（箭头工具）双击选中的组。这时 Flash MX 将进入如图 4-28 所示的组编辑状态。若要结束编辑，则使用 Edit | Edit All 菜单命令或者用箭头工具在空白的编辑区域中双击。

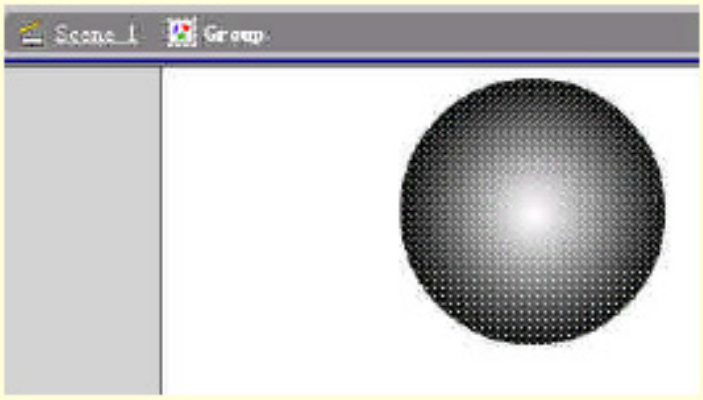


图 4-28　组编辑状态

（3）若要将此组图作为第 20 帧，则单击时间轴上的第 20 帧，选择 Insert | Keyframe 菜单命令或者按 F6 快捷键，在第 20 帧处插入关键帧，效果如图 4-29 所示。

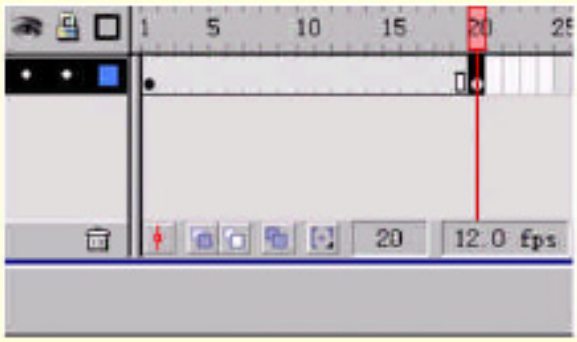


图 4-29　在第 20 帧处插入关键帧

（4）在第 20 帧上按照图 4-30 所示的方法用鼠标向右拖动圆球一段距离。

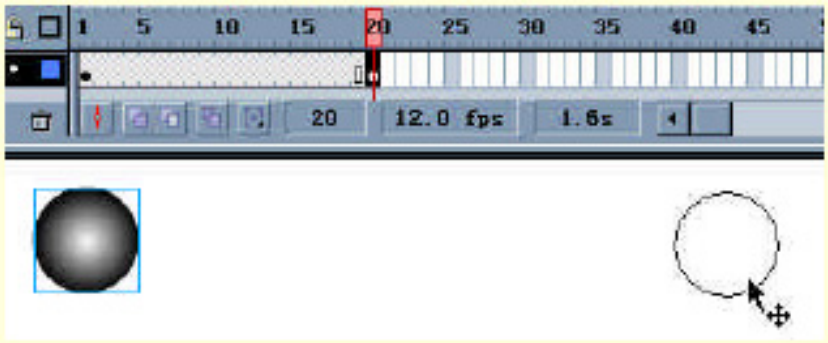


图 4-30　拖动圆球

（5）在第 1~19 帧之间单击左键，设置 Properties（属性）面板中的 Tween（变化）列表框为 Motion，其余设置均使用如图 4-31 所示的默认值。这时，在时间轴上的第 1~19 帧之间将出现一个如图 4-32 所示的蓝色箭头，表示位置渐变。

图 4-31　属性面板的设置

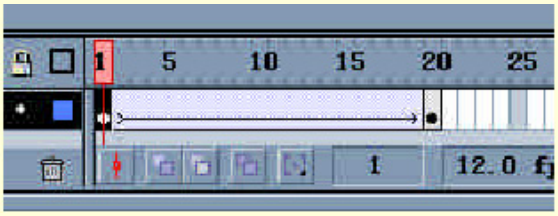


图 4-32　箭头表示位置渐变

（6）最后，按 Ctrl+Enter 键或者选择 Control | Test Movie 菜单命令，输出并运行 Flash MX 动画。这时，球将从第 1 帧的位置逐渐移动到第 20 帧，并反复运动。直线运动物体的动画制作完成。

4.3.2　沿轨道运动的物体

在直线运动的基础上，只要加上一个向导层就可以实现物体沿设定轨道的运动。

（1）选择 File | New，新建一个 Flash MX 文件，选择 Rectangle Tool（矩形工具）， Stroke（描绘）颜色和 Fill（填充）颜色随意使用。选中 Snap to Object（磁铁）工具。在 Scene（场景）的第 1 帧上绘制一个如图 4-33 所示的正方形。

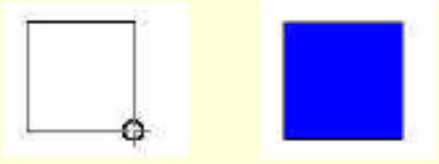


图 4-33　绘制一个正方形

（2）在 Timeline（时间轴）的第一帧上单击鼠标，这时第一帧上的所有对象处于选中状态。选择 Modify | Group 菜单命令或者使用快捷键 Ctrl+G 将第一帧上所选中的对象组合成一个组。

（3）用鼠标右击当前层，在弹出的快捷菜单中选择如图 4-34 所示的 Add Motion Guide 菜单命令，为当前层（Layer1）添加一个向导层。添加结果如图 4-35 所示。关于层的进一步介绍将在下一章中进行。

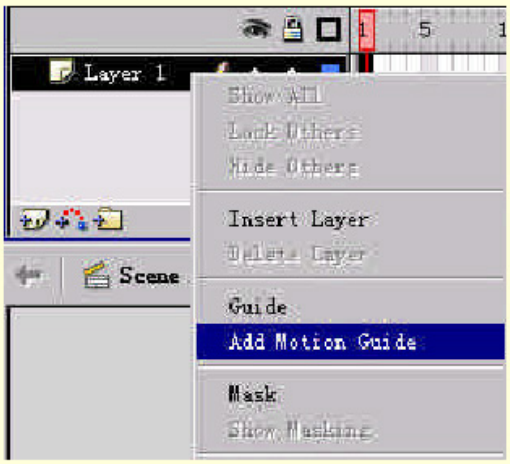


图 4-34　添加向导层

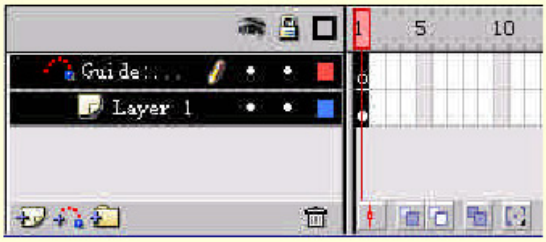


图 4-35　向导层添加结果

（4）在添加的向导层上，用 Pencil Tool（铅笔工具）随意画一条如图 4-36 所示的曲线，作为正方形对象的移动轨迹。

- 移动轨迹一定要画在向导层 Guide：Layer 上，画在其他层上则无效。在向导层绘图时层的状态如图 4-37 所示。

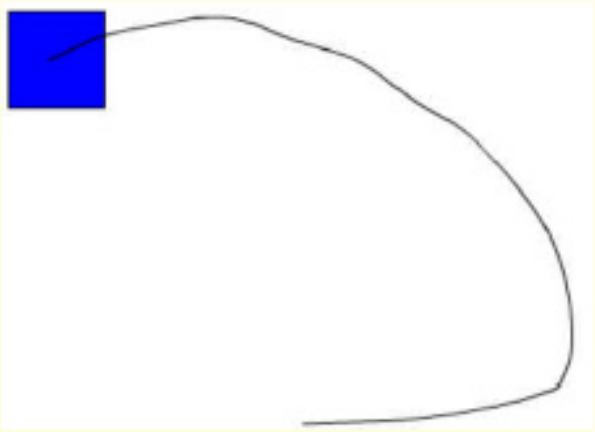


图 4-36　绘制移动轨迹

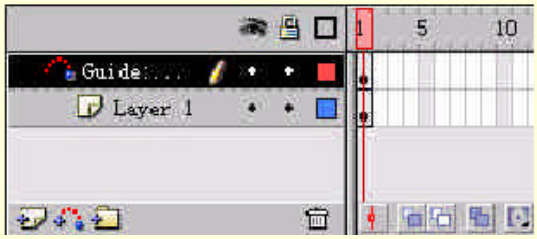


图 4-37　在向导层绘图

（5）选中 Snap to Object（抓握）工具，使用 Arrow Tool（箭头工具）在正方形的中心拖动，并将其移动到所画曲线的一个端点，这时，正方形的中心将出现一个如图 4-38 所示的加粗圆圈，表示正方形已经锁定在曲线上。

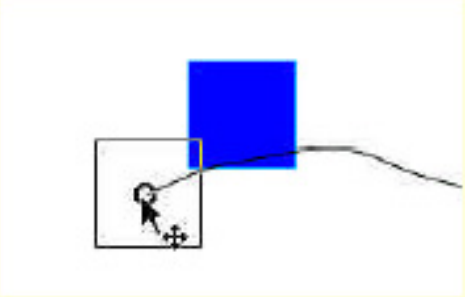


图 4-38 将正方形锁定在曲线上

（6） 在正方形所在的层 Layer 1 中的 Timeline（时间轴）的第 30 帧处选择 Insert | Keyframe 菜单命令或者按 F6 快捷键插入 Keyframe（关键帧），如图 4-39 所示。

（7）在向导层 Guide：Layer 1 中的 Timeline（时间轴）的第 30 帧处选择 Insert | frame 菜单命令或者按 F5 快捷键插入 Keyframe（关键帧），如图 4-40 所示，使向导层作用于第 1~30 帧之间。

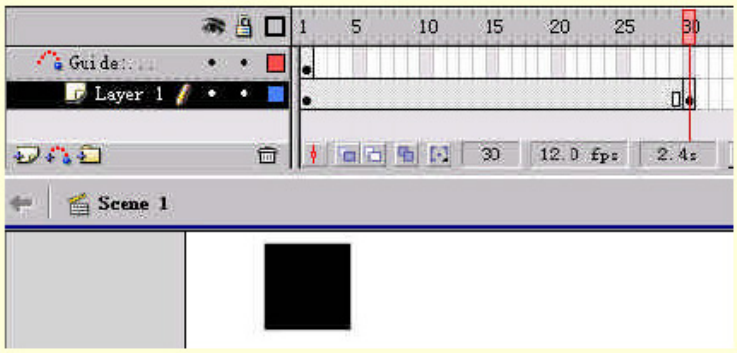


图 4-39 插入关键帧

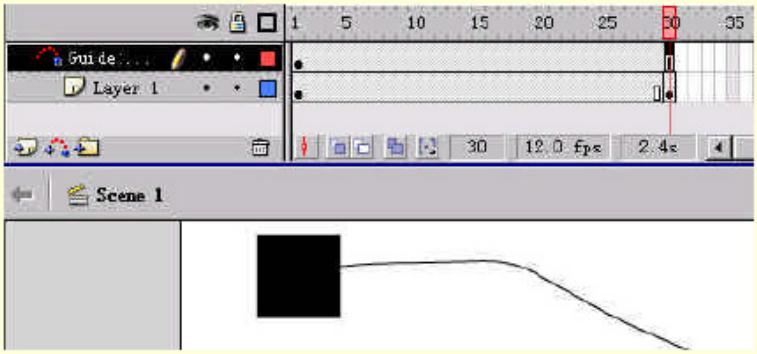


图 4-40 在向导层插入帧

✎ 请注意插入关键帧与插入帧的区别。前者是复制前一关键帧到当前帧的所在位置，并自动将被复制帧的作用范围设置到当前位置之前；而后者是直接将前一关键帧的作用范围设置到当前位置，并不创建关键帧。

（8）选中第 30 帧，按照如图 4-41 所示的方法将正方形从曲线的一个端点拖动并锁定到另一个端点。

（9）在第 1~29 帧之间单击左键，设置属性面板中的 Tween（变化）列表框为 Motion，并选中 Orient to path（适应路径）选项，其余设置均使用如图 4-42 所示的默认值。这时，在时间轴上的第 1~29 帧之间将出现一个蓝色的箭头表示位置渐变。

（10）最后，按 Ctrl+Enter 键或者选择 Control | Test Movie 菜单命令，输出并运行 Flash MX 动画。这时，正方形将沿曲线反复运动。至此，沿曲线运动物体的动画制作完成。

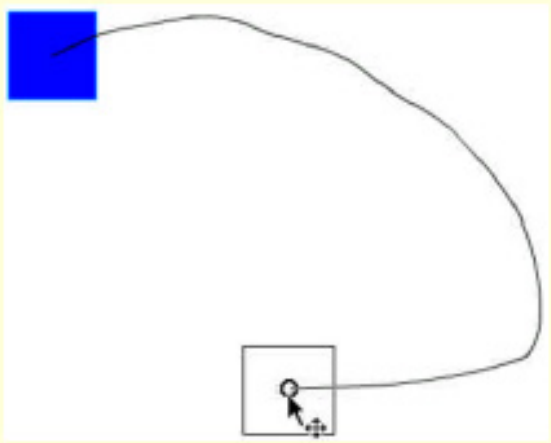


图 4-41　锁定正方形到另一个端点

图 4-42　设置帧面板

4.3.3　物体的转动

利用 Flash MX 中的位置渐变功能也可以实现物体的转动。下面是实现一个矩形原地转动的例子。

（1）新建一个 Flash MX 文件，在第 1 帧上绘制一个如图 4-43 所示的矩形，并将其组成一个组。

（2）在时间轴第 20 帧上插入 Keyframe（关键帧）。选中第 20 帧上的矩形，使用旋转操作将其旋转 180°，旋转过程如图 4-44 所示。

图 4-43　绘制一个矩形

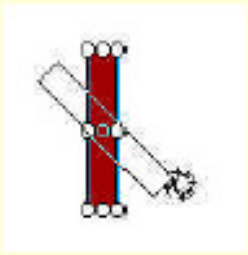


图 4-44　旋转矩形

（3）在第 1~19 帧之间单击，设置属性面板中的 Tween（变化）列表框为 Motion，其余设置均使用如图 4-45 所示的默认值。

（4）最后，按 Ctrl+Enter 键或者选择 Control | Test Movie 菜单命令，输出并运行 Flash MX 动画。这时，矩形将反复转动。至此，转动物体的动画制作完成。

图 4-45　设置帧面板

4.3.4　物体运动参数的设定

在 4.3 节的几个例子中基本上使用了 Motion（运动渐变）默认值，其他参数的功能及设置方法如下。

- Frame Label（标签）：代表当前帧的字符串，在 ActionScript 脚本语言中使用，具体做法将在后续章节中介绍。
- Scale（比例）：在渐变的过程中，允许比例缩放。图 4-46 和图 4-47 是对一个正方形使用和禁止比例缩放的结果（这里使用洋葱皮轮廓模式来表示动画的所有帧）。

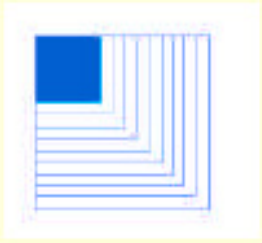


图 4-46 允许比例缩放

图 4-47 禁止比例缩放

- Ease（运动速度变化比）：调节物体渐变过程中的速度比例关系。正数表示先快后慢，负数表示先慢后快。图 4-48 和图 4-49 为正方形放大动画的洋葱皮轮廓显示。图 4-48 中 Ease=50；图 4-49 中 Ease=－50。

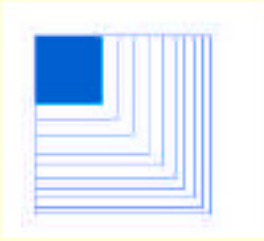


图 4-48 Ease=50

图 4-49 Ease=－50

- Rotate（旋转）：在渐变过程中转动的方式。
  - None（不旋转）：如图 4-50 所示。
  - Auto（自动）：让 Flash MX 自动调节转动角度，也可以配合沿路径移动选项使用。如图 4-51 所示。
  - CW（顺时针旋转）：如图 4-52 所示。
  - CCW（逆时针旋转）：如图 4-53 所示。
- Orient to path（适应路径）：使物体沿路径方向自动转动。在沿路径运动时，若不选用此选项和旋转选项，则物体只做平动，如图 4-54 和图 4-55 所示。

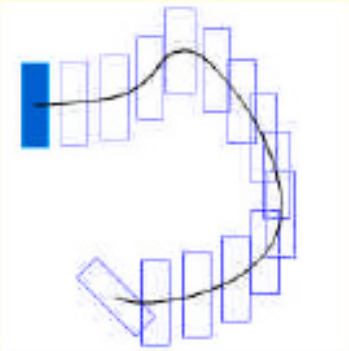


图 4-50 不旋转图

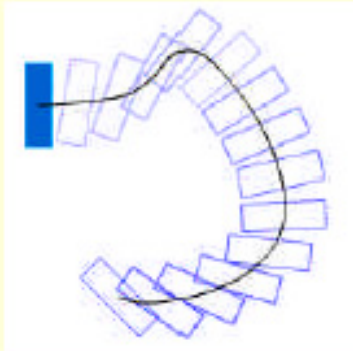


图 4-51 自动旋转

图 4-52　顺时针旋转 CW=1

图 4-53　逆时针旋转 CCW=1

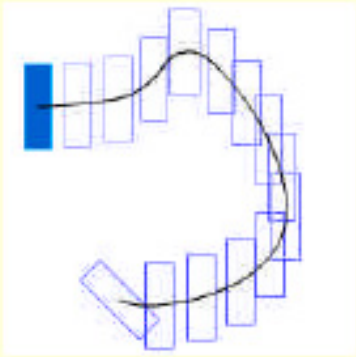


图 4-54　未适应路径

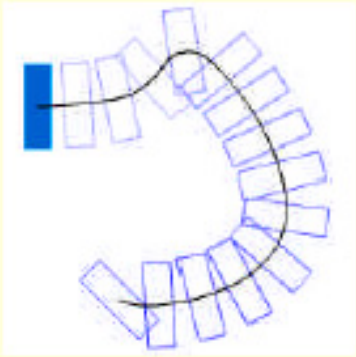


图 4-55　适应路径

4.4　神奇的图形渐变

4.4.1　简单示例

在上述位移渐变的例子中，对于每一个图形都使用了 Group（成组）命令，其目的在于让 Flash MX 对图形进行整体渐变，否则 Flash MX 将如图 4-56 所示，不能进行 Motion 渐变。

图 4-56　不能进行 Motion 渐变

本节将介绍图形内部的渐变，下面给出一个例子——变形文字。注意这里没有使用 Group（成组）命令，相反却使用了 Break Apart（分解）命令。具体步骤如下。

（1）新建一个 Flash MX 文件，用 Text Tool（文字工具）选择 Arial Black 字体，在第 1 帧上写一行如图 4-57 所示的文字“ FLASH ”。

图 4-57 输入文字

（2）使用 Arrow Tool（箭头工具）选中文字。两次选择 Modify | Break Apart 菜单命令或两次使用 Ctrl+B 快捷键将文字分解，分解后的效果如图 4-58 所示。

图 4-58 文字分解后的效果

✎ 文字一旦分解以后将作为图形处理，无法再次进行文本编辑。所以，一定要在文字编辑已经完成并且不准备再次编辑的时候，才可以将文字分解。

（3）选中时间轴第 20 帧，选择 Insert | Blank Frame 菜单命令或使用 F7 快捷键在第 20 帧处插入如图 4-59 所示的空白帧。

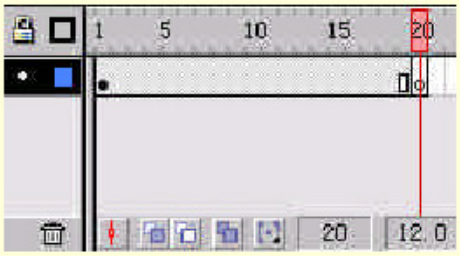


图 4-59 插入空白帧

（4）同样在第 20 帧内，使用 Text Tool（文本工具）并选择字体为宋体，写出另一行文字“变形文字”，并将其分解成图 4-60 所示的样子。

图 4-60 第 20 帧的文字

（5）开启时间轴的洋葱皮状态，拖动第 20 帧的文字使其与第 1 帧的文字重叠，效果如图 4-61 所示。

图 4-61 重叠文字

（6）选中第 1 帧，设置属性面板中的 Tween（变化）列表框为 Shape，其余设置均使用如图 4-62 所示的默认值。这时，在时间轴上的第 1~19 帧之间将出现一个如图 4-63 所示的绿色箭头表示图形渐变。

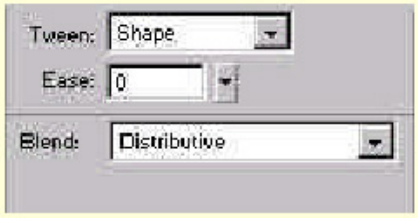


图 4-62　属性面板的设置

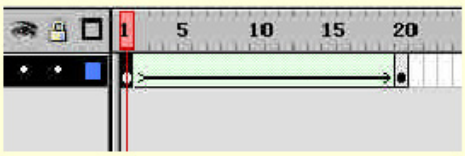


图 4-63　图形渐变的时间轴

（7）最后，按 Ctrl+Enter 键或者选择 Control | Test Movie 菜单命令，输出并运行 Flash MX 动画，变形文字动画制作完成，其结果如图 4-64 所示。

图 4-64　变形文字运行结果

4.4.2　提示点的设定

在 4.4.1 节所述的变形文字中，文字的变形过程由 Flash MX 自动控制，完全不受人为控制。这样，在某些要求不高的情况下可以满足用户的需要。但是，在很多情况下要求根据需要设定变形的具体过程，这就需要使用 Flash MX 的 Shape Hints（设置提示点）命令进行控制。下面给出一个例子：

（1）新建一个 Flash MX 文件，用 Text Tool（文字工具）选择 Impact 效果，在第 1 帧上写文字“1”，效果如图 4-65 所示。

（2）在 Timeline（时间轴）第 20 帧处用 Insert | Key Frame 菜单命令或 F6 快捷键插入关键帧。将文字“1”改为“2”，效果如图 4-66 所示。

图 4-65　第 1 帧上写“ 1 ”

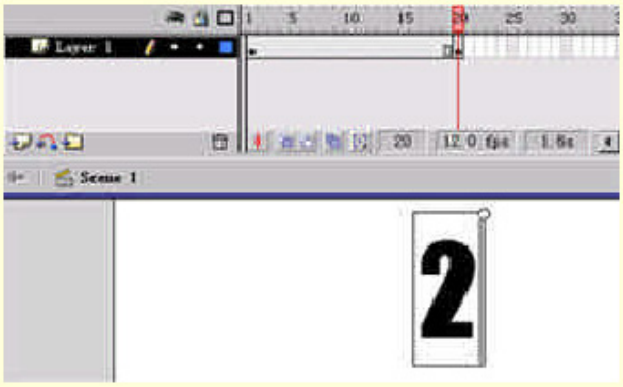


图 4-66　文字“ 1 ”改为“ 2 ”

（3）分别选中第 1 帧和第 2 帧，使用 Modify | Break Apart 菜单命令或 Ctrl+B 快捷键把文字分解成如图 4-67 所示的样子。

图 4-67　分解后的文字

（4）在第一帧的属性面板上设置 Tween（变化）为 Shape (图形渐变) (如图 4-68 所示)。

（5）按 Ctrl+Enter 键或者选择 Control | Test Movie 菜单命令，输出并运行 Flash 动画，其结果如图 4-69 所示。

（6）选中第 1 帧，选择 Modify | Shape | Add Shape Hint 菜单命令或者按 Ctrl+Shift+H 快捷键，插入一个图形渐变的提示点，在第 1 帧和第 20 帧图形的中心将出现如图 4-70 所示的红色的提示点。

（7）分别用鼠标拖动第 1 帧和第 20 帧的提示点至图 4-71 所示的位置。这时，第 1 帧的提示点变为黄色，第 20 帧的提示点变为绿色。

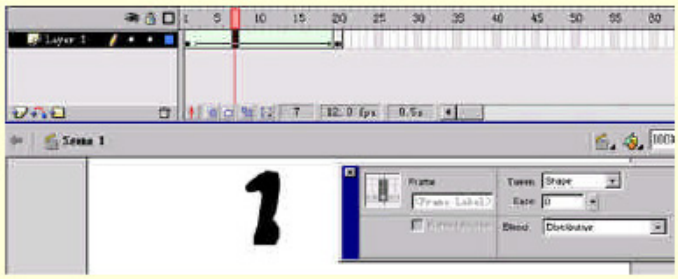


图 4-68　设置为图形渐变

图 4-69　1—2 图形渐变

图 4-70　插入的提示点

图 4-71　设置提示点的位置

（8）重复步骤（6）~（7），插入并拖动提示点，如图 4-72 所示。

图 4-72　插入并拖动第 2 个提示点

（9）按 Ctrl+Enter 键或者选择 Control | Test Movie 菜单命令，输出并运行 Flash MX 动画，其结果如图 4-73 所示。

由此可见，提示点的意义在于，它可以让用户指定图形渐变中某一点具体的变化位置。有了提示点，图形渐变的过程就变得更加明确。提示点越多，图形渐变的过程就越详细。在 Flash MX 中，对于同一个图形渐变最多可以插入 26 个提示点，并由 26 个小写的英文字母标出。这对一般的动画来说已经足够了。

- 在提示点上右击鼠标，可以弹出一个快捷菜单（如图 4-74 所示），以实现提示点的编辑操作：Add Hint（增加提示点）、Remove Hint（删除提示点）、Remove All Hints（删除所有提示点）、

Show Hints（显示提示点）。

图 4-73　运行结果

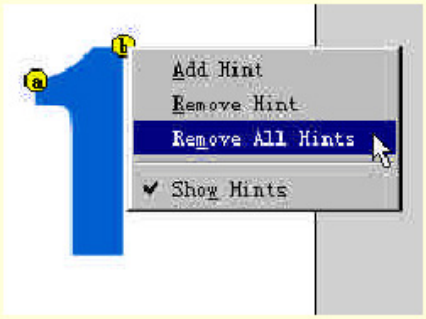


图 4-74　编辑提示点快捷菜单

4.4.3　图形渐变参数的设定

图形渐变的属性面板（如图 4-75 所示）中的参数说明如下：

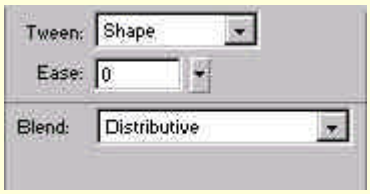


图 4-75　图形渐变的属性面板

- Ease（运动速度变化比）：与 Motion（位置渐变）中的同名选项一样，速度变化比可以调节物体渐变过程的速度比例关系。正数表示先快后慢，负数表示先慢后快。图 4-76 和图 4-77 所示的为正方形变为圆形的动画的洋葱皮轮廓显示。图 4-76 中 Ease=50；图 4-77 中 Ease=－50。

图 4-76　Ease=50

图 4-77　Ease=－50

- Blend（混合模式）
  - Distributive（分布式）：如图 4-78 所示。

- Angular（弯角式）：如图 4-79 所示。

图 4-78　分布式

图 4-79　弯角式

4.5　逐帧动画

虽然创作逐帧的 Flash MX 动画是一个很复杂的工作。但是，逐帧动画的每一帧都是独立的，因此它可以创作出很多依靠 Flash MX 的渐变功能根本无法实现的动画。下面的例子实现了一个写字过程的 Flash MX 动画，体现出了逐帧动画的优越性。

（1）新建一个 Flash MX 文件，如图 4-80 所示，用 Text（文字）工具选择 Country Blueprint 字体，在第 1 帧上写上文字“ FLASH ”。

（2）选中第 1 帧的文字，如图 4-81 所示，两次使用 Modify | Break Apart 菜单命令或 Ctrl+B 快捷键把文字分解。

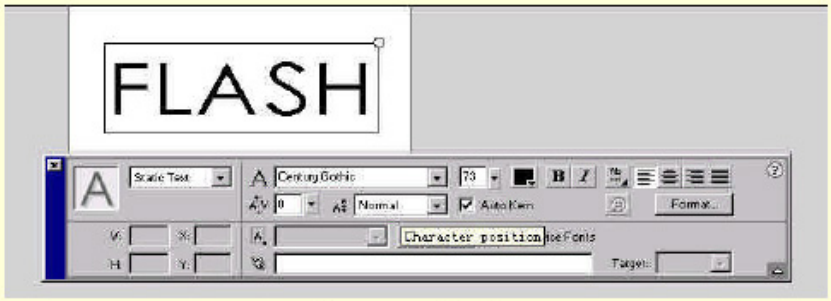


图 4-80　添加文字

图 4-81　分解以后的文字

（3）选中第 2 帧，选择 Insert | Keyframe 菜单命令或者 F6 快捷键，紧接着第 1 帧创建关键帧。此时的 Timeline（时间轴）如图 4-82 所示。

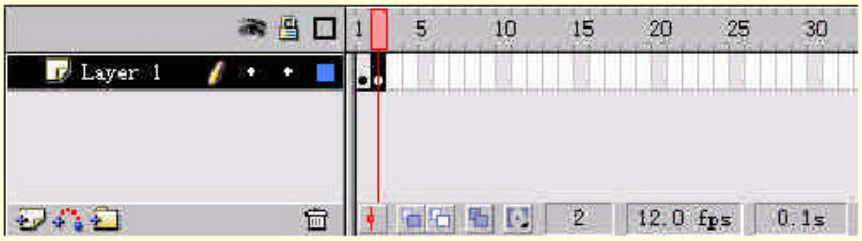


图 4-82　插入关键帧的时间轴

（4）在第 2 帧上使用橡皮工具，如图 4-83 所示，擦除文字“ FLASH ”的最后一笔的一部分。

图 4-83　擦除文字

（5）紧接着第 2 帧创建第 3 帧为关键帧。如图 4-84 所示，在第 3 帧上按照与文字“ FLASH ”笔画顺序相反的顺序，继续擦除文字。

图 4-84　继续擦除文字

（6）重复上述的创建关键帧并擦除笔画的步骤，按照与文字“ FLASH ”笔画顺序相反的顺序逐渐地将文字一点一点地擦除干净。最后将得到一个文字“ FLASH ”的笔画逐渐消失的动画。这时的时间轴如图 4-85 所示。其中，关键帧的数目由擦除文字的过程所决定，并没有严格的限制，但是，它决定了动画播放的连续性。

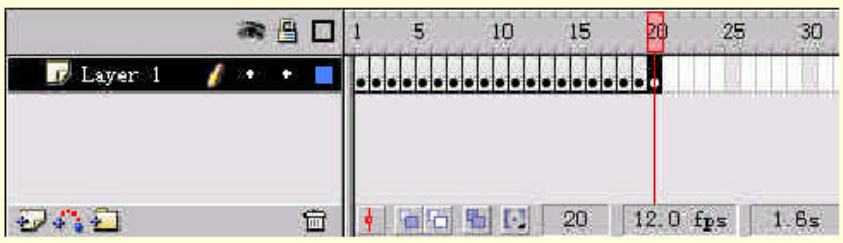


图 4-85　擦除操作创建的关键帧

✎ 在擦除并创建关键帧的过程中，注意均匀地擦除。即每次擦除笔画长度都差不多，这样才可以保证动画播放时文字书写是匀速的。

（7）选中创建好的所有关键帧。在关键帧上右击鼠标，选择 Reverse Frames 菜单项，如图 4-86 所示，将所有的关键帧的顺序反相。

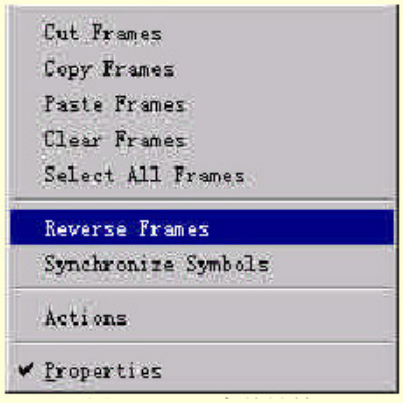


图 4-86　反序关键帧

（8）按 Ctrl+Enter 键或者选择 Control | Test Movie 菜单命令，输出并运行 Flash MX 动画，文字“FLASH”便逐渐地被书写出来，其结果如图 4-87 所示。

图 4-87　运行结果

- 如图 4-88 所示，如果在每一帧中文字笔画的末端加上一个粉笔，就制成了一个用粉笔写字的动画。

图 4-88　用粉笔写字的动画

可见，在上述动画中主要使用了插入关键帧技术。因为在逐帧的 Flash MX 动画中相邻两帧之间的差别一般都很小，而插入关键帧可以把前一帧的内容复制到所选中的位置，所以只需要对前一帧的内容作很小的改动，如此连续地一点一点地改动并创建关键帧就相当于把所做改动的过程用照相机拍下来一样，从而得到显示整个改动过程的连续的 Flash MX 动画。

- 如果制作的动画是放在网页上，则不提倡大量地使用逐帧形式的 Flash MX 动画。因为逐帧动画中记录了每一帧动画的全部内容，它所输出的文件将比相同时间的渐变动画大得多，不利于网上传输。

第二部分

Flash MX 提高篇

第 5 章　复杂的 Flash 动画

第 6 章　交互式 Flash 动画基础

第 7 章　创作交互试 Flash 动画

第8章　组件的使用

第9章　外部文件的导入

第 5 章　复杂的 Flash 动画

5.1　综　述

本章将着重讲述 Flash MX 动画中几个重要的概念及其用法。读者将学会如何使用层，如何在动画中使用向导层和屏蔽层，如何创建元件等高级动画功能。

5.2　如何使用层

5.2.1　层的优点

与大多数优秀的图形图像处理软件一样，层的使用给用户带来了极大的方便。使用层制作图片或者动画，就好像将多张透明电影胶片叠放在一起显示出来。一个新建的 Flash MX 文件仅仅包含了一个层，用户可以随意添加或删除层（如图 5-1 所示）。

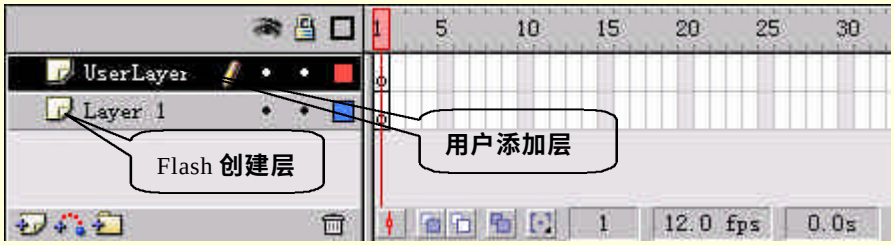


图 5-1　Flash MX 中的层

使用层的优点在于，用户可以在一个层上随意地修改该层的图形而不影响其他层，这样便于动画文件的编辑管理。同时，层也把图形进行分类，在不同层上可以分别进行不同形式的动画渐变，从而实现更复杂的 Flash MX 动画。

另外，Flash MX 还提供了两种特殊的层：Guide Layer（向导层）和 Mask Layer（屏蔽层）。前者已经在上一章中介绍过了，本章将更加深入地讲述向导层的用法。屏蔽层是很特殊的层，利用它可以实现许多特殊效果，在本章的 5.2.3 节中将做详细的说明。

> Flash MX 中对层进行了特殊的处理，用户所能添加层的数目只依赖于系统内存的多少，并没有其他的限制。而且，Flash MX 最终输出的动画文件不会因为层的增多而增加大小，这就有利于动画网上的传输。所以，建议用户尽量使用层，以方便 Flash MX 动画文件的编辑与管理。

5.2.2　层的编辑和使用

Flash MX 中对层的操作包括以下几种。

- 编辑层
  - 添加层：选择 Insert | Layer 菜单命令或者在某一层上右击鼠标，在弹出的如图 5-2 所示的快捷菜单上选择 Insert Layer 项，或者直接单击图 5-3 所示的添加层按钮，即可在当前的层上添加一个新层。层名由 Flash MX 自动给出。

图 5-2 层快捷菜单

图 5-3 编辑层的按钮

- 删除层：选中一层或多层（可配合 Shift 键），用快捷菜单中的 Delete Layer 命令或者图 5-3 所示的删除层按钮删除层。
- 重命名层：在某一层的名字上双击鼠标，即可按图 5-4 的方法改变该层名字。

图 5-4 层的重命名

- 层的移动：通常，上面层的内容会挡住下面层的内容。所以，有时需要调节层的顺序。直接用鼠标拖动某一层到指定的位置即可，如图 5-5 所示。

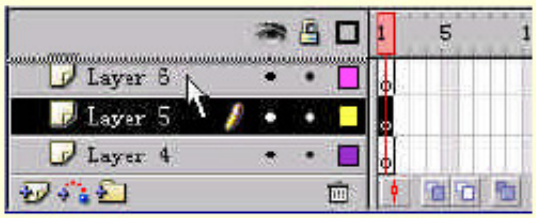


图 5-5 层的移动

- 层文件夹：针对以前的版本中出现的层的管理太凌乱的现象，Flash MX 新增加了层文件夹，可以将层分类管理，方便了用户。要新建一个层文件夹，可以选择 Insert | Layer Folder 菜单命令或单击如图 5-3 所示的添加层文件夹按钮。

- 设置层的模式：编辑 Flash MX 动画有时需要改变层的显示模式，以便于对其他层进行编辑。只需在相应层的模式设置位置上单击鼠标即可，请读者参考图 5-6。层的显示模式包括：

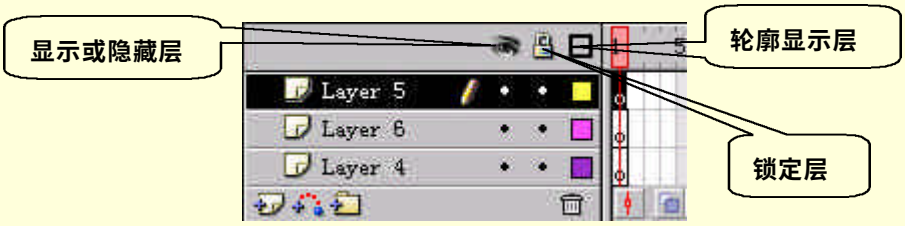


图 5-6　设置层的模式

- 层的显示及隐藏：在“眼睛”图标对应的黑点上单击鼠标，即可显示或隐藏层，情况如图 5-7 所示。在层的快捷菜单中选择 Hide Others，将隐藏当前层以外的所有层。层一旦被隐藏，在编辑的时候将不可见，也不能编辑。但是，在输出 Flash MX 动画的时候，该层仍然可见。

图 5-7　层的隐藏

- 层的锁定：在“锁定”图标对应的黑点上单击鼠标，即可将层锁定或解锁，情况如图 5-8 所示。在层的快捷菜单中选择 Lock Others，将锁定当前层以外的所有层。层一旦被锁定，在编辑的时候仍然可见，但是不能被编辑。这一功能对于不在同一个层的图形的定位很有帮助。

图 5-8　层的锁定

- 轮廓显示层：在“方框”图标对应的黑点上单击鼠标，即可以按图 5-9 所示的轮廓方式显示层。层一旦以轮廓方式显示，在编辑的时候该层图形的颜色与“方框”所对应的颜色相同。这一功能对于在同一个层中有大量复杂的图形需要编辑或者图形的前景色与背景色相同的情况很有帮助。以正常形式的轮廓方式显示的结果分别如图 5-10 及图 5-11 所示。

图 5-9　开启轮廓显示

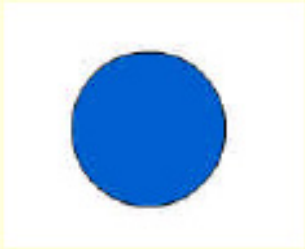


图 5-10 以正常方式显示

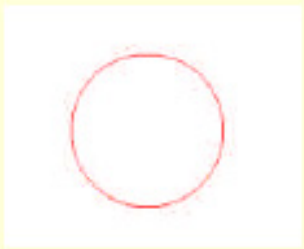


图 5-11 以轮廓方式显示

5.2.3 层的应用

Flash MX 中有两种特殊的层：Guide Layer（向导层）和 Mask Layer（屏蔽层）。

在前一章中已经给出了一个简单的使用向导层的例子。其实，向导层也是可以改变的，也就是说在不同的时刻，物体沿不同的轨道运动。下面给出一个“原子图案”的例子，在此例子中，“原子核外电子”将沿着两条不同的轨道运动，具体做法如下：

（1）新建一个 Flash MX 文件，在舞台的中央绘制一个红色的圆表示“原子核”，参考图 5-12。

图 5-12 绘制“原子核”

（2）在当前层上添加一个新层 Layer2，并在该层上以沿径向的黑白渐变为填充色绘制一个如图 5-13 所示的圆，表示“原子核外电子”。

（3）在“电子”所在层上添加一个 Guide Layer（向导层），并在 Guide Layer（向导层）上绘制一个长轴倾斜 45° 的椭圆，“原子核”为椭圆的中心，如图 5-14 所示。

（4）在 Guide Layer（向导层）第 20 帧处插入关键帧并将椭圆旋转 180°，表示“电子的另一条轨道”，绘制结果如图 5-15 所示。

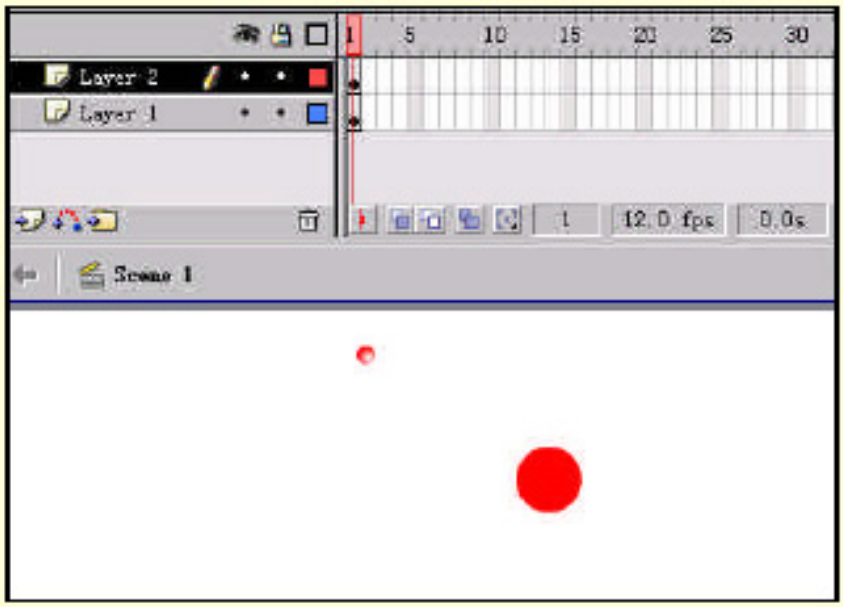


图 5-13 绘制“电子”

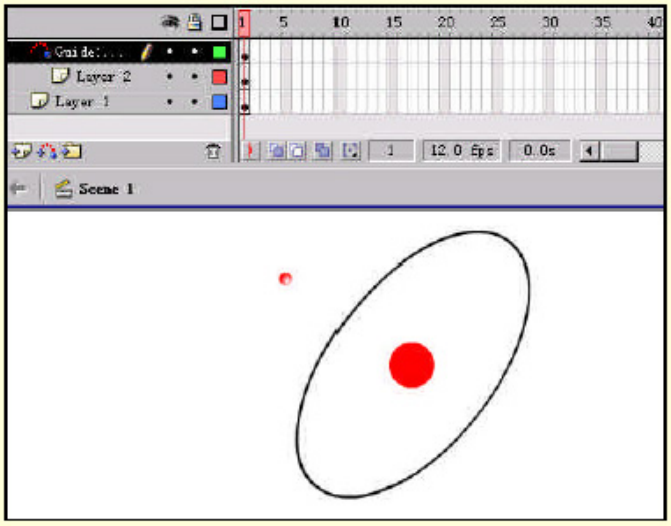


图 5-14　绘制椭圆

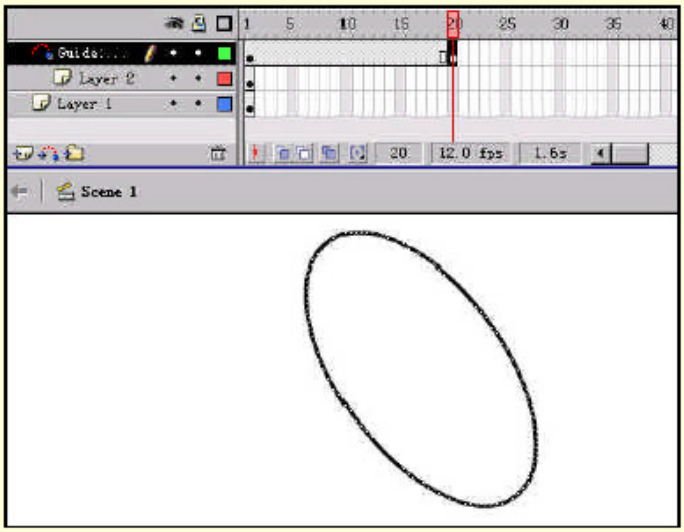


图 5-15　电子的另一条轨道

（5）开启磁铁工具和帧的洋葱皮模式。把第 1 帧的“ 电子 ”移动到两条“ 轨道的交汇处 ”，如图 5-16 所示。

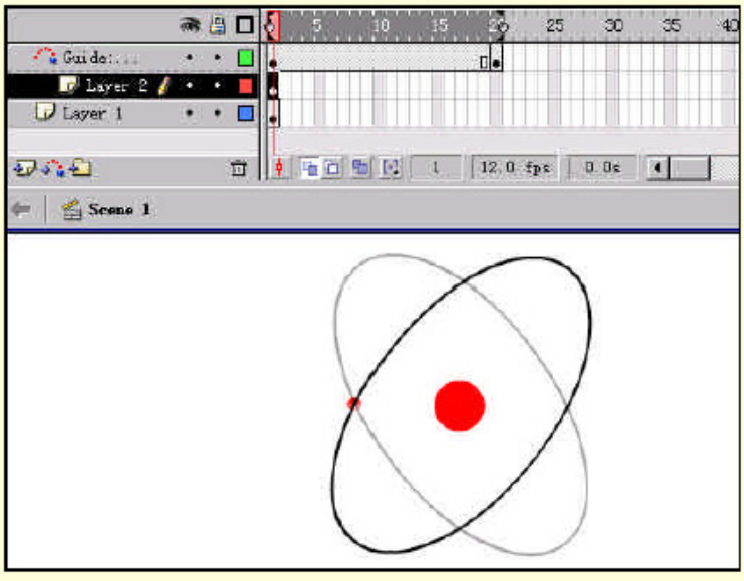


图 5-16　移动“ 电子 ”

✍ 移动“电子”时可以锁定 Guide Layer（向导层），以免对轨道产生影响。虽然锁定层以后，帧的洋葱皮功能将失去作用，但是，只要打开磁铁工具，同样可以很容易地捕捉到两条轨道的交点。

（6）在“电子”所在层的第 7 帧和第 14 帧插入关键帧，将“电子”移动到图 5-17 和图 5-18 所示的两个位置。

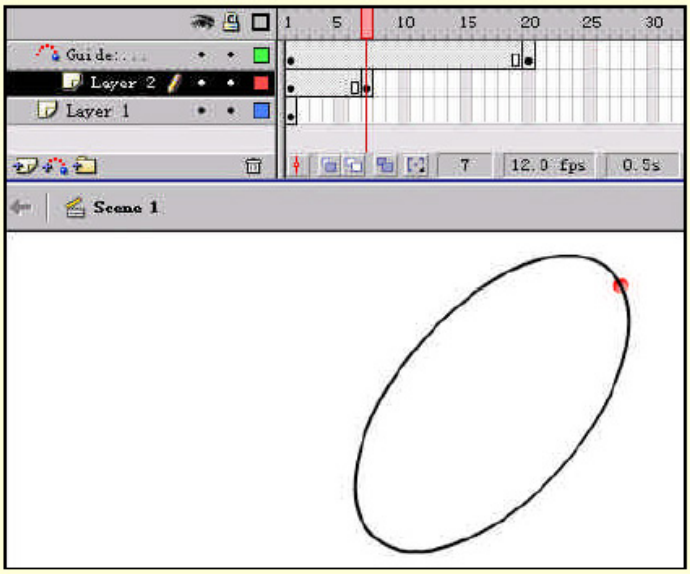


图 5-17　第 7 帧的“电子”

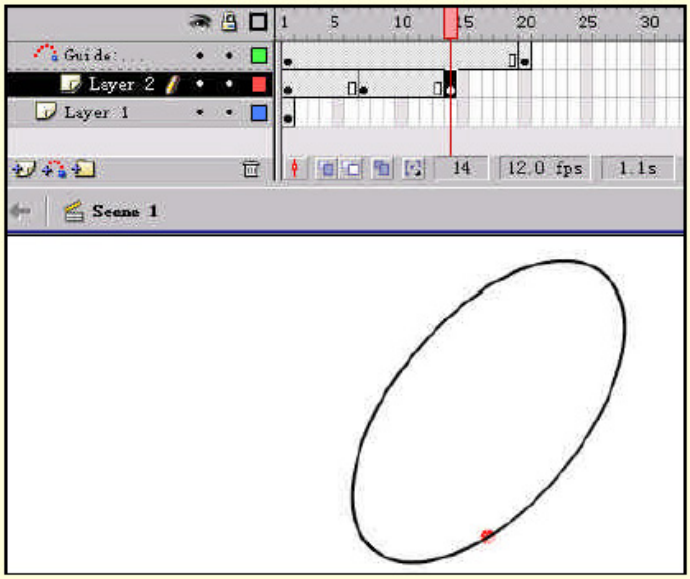


图 5-18　第 14 帧的“电子”

（7）把“电子”所在层的第 1 帧复制到第 20 帧，把 Guide Layer（向导层）的第 1 帧复制到第 40 帧。在步骤（7）操作完成后，时间轴上的情况应如图 5-19 所示。

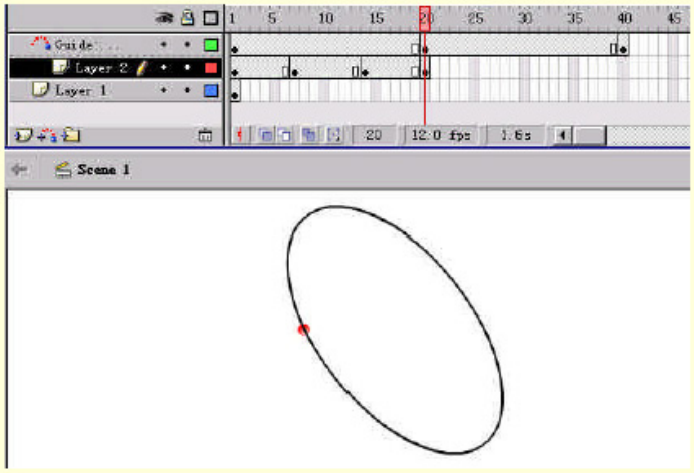


图 5-19　步骤（7）

（8）与步骤（6）相似，在“电子”所在层的第 27 帧和第 34 帧插入关键帧，并将“电子”移动到图 5-20 和图 5-21 所示的两个位置。

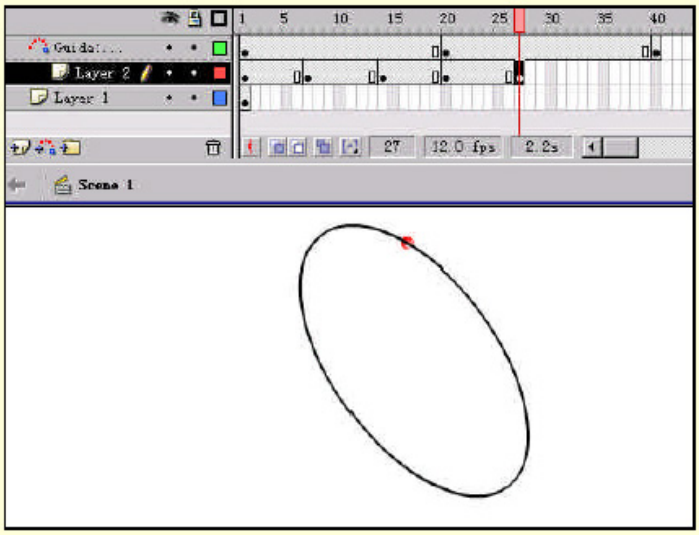


图 5-20　第 27 帧“电子”的位置

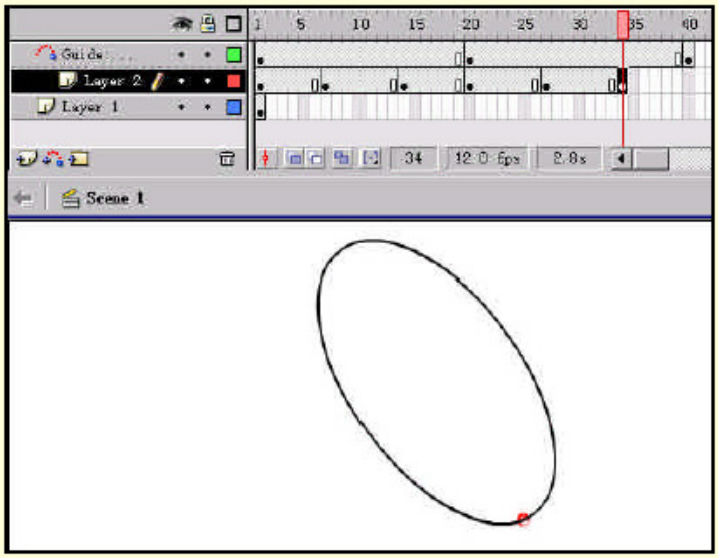


图 5-21　第 34 帧“电子”的位置

（9）把“电子”所在层的第 20 帧复制到第 40 帧，并在“原子核”所在层的第 40 帧处插入帧，使“原子核”在 1~40 帧的范围内静止显示，如图 5-22 所示。

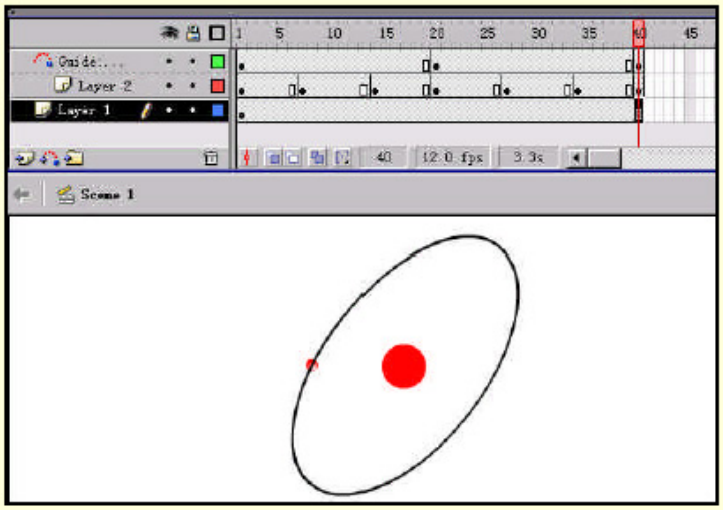


图 5-22 步骤（9）

（10）将“电子”所在层的每一帧之间设为 Motion（位置）渐变。

- 直接用鼠标右击“电子”所在层每一帧，在弹出的快捷菜单上选择 Create Motion Tween 菜单项即可。Flash MX 会把当前帧上所有内容组成 Symbol（元件），然后与下一帧进行位置渐变，请参考图 5-23。关于元件的内容将在 5.3 节中介绍。

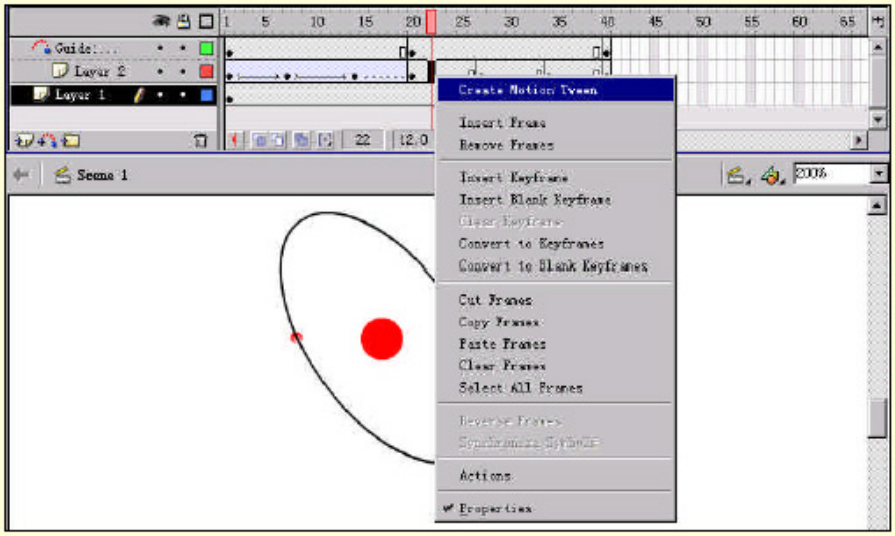
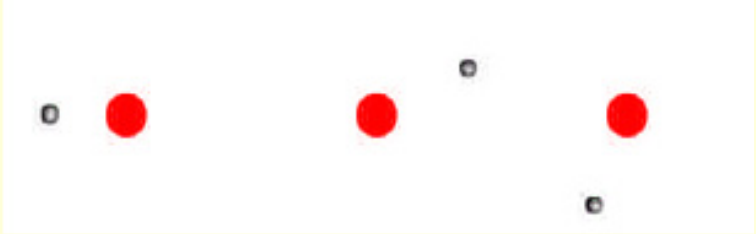


图 5-23 用快捷菜单渐变

（11）输出并运行动画。“电子”将分别沿着两条不同的轨道围绕“原子核”运动，如图 5-24 所示。



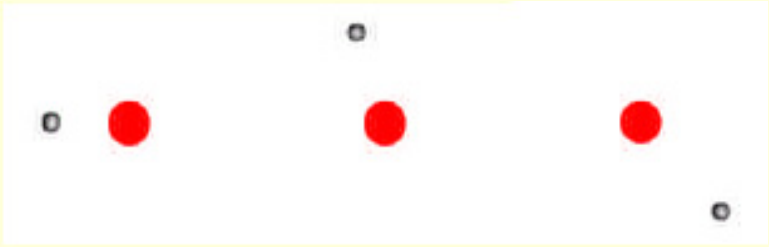


图 5-24 “电子”的环绕运动

（12）因为向导层并不显示，所以上述“电子”运动的轨道并没有显示出来。这就需要再创建一个轨道显示层，使“电子”运动的轨道按图 5-25 所示明确地显示出来。

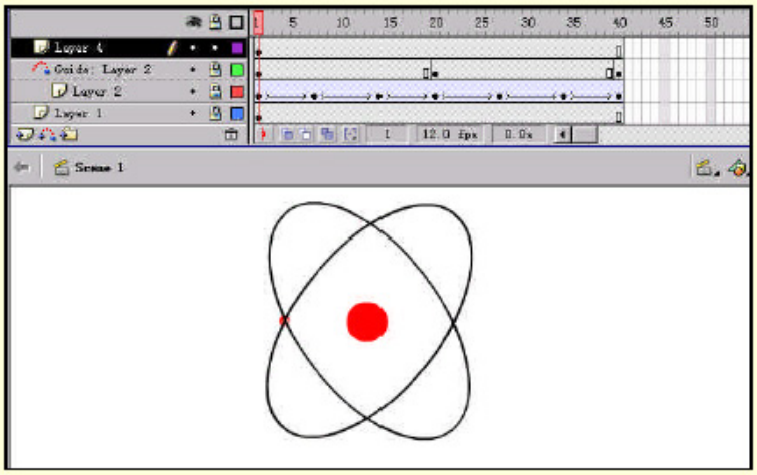


图 5-25 创建轨道显示层

（13）最后，输出并运行动画，“电子”的运动轨道也显示出来了，运行结果如图 5-26 所示。通过改变向导层使“电子”不停地沿轨道运动，这样可以实现很多轨道复杂的运动。

在制作“电子”的每一周环绕运动时都使用了 3 个关键帧。因为，在 Flash MX 的向导层中，当有两条或两条以上的轨道都可以到达下一关键帧时，Flash MX 自动将物体沿最短的轨道运动，所以，如果只是用 2 帧关键帧制作环绕动画，其结果是物体沿一条轨道运动，再沿同一条轨道返回。

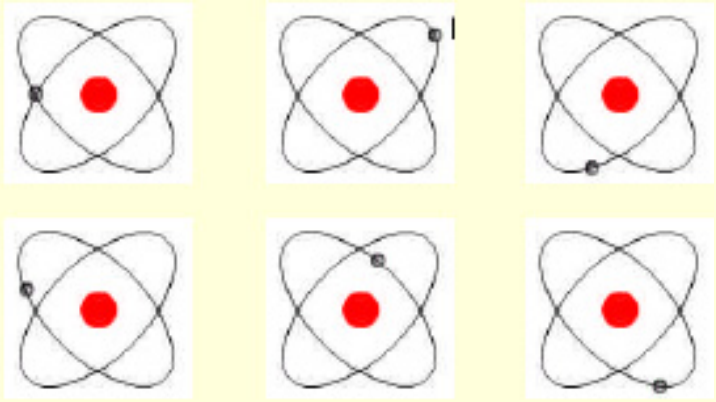


图 5-26 “电子”沿轨道的环绕运动

在 Flash MX 中另一个特殊的层是 Mask Layer（屏蔽层），下面给出一个例子。

（1）新建 Flash MX 文件，用文字工具在第 1 帧输入文字“Flash”，颜色随意，如图 5-27 所示。

图 5-27　输入文字

（2）插入一个新层 Layer2，用椭圆工具在该层上绘制一个圆，填充色使用黑白色的径向渐变，并将其组成一个组，再把此层移动到如图 5-28 所示的文字层 Layer2 之下。

图 5-28　插入新层

（3）在 Layer2 的第 20 帧和第 40 帧处插入关键帧，在 Layer1 的第 40 帧处插入帧，再将第 20 帧的圆移动到图 5-29 所示的位置。

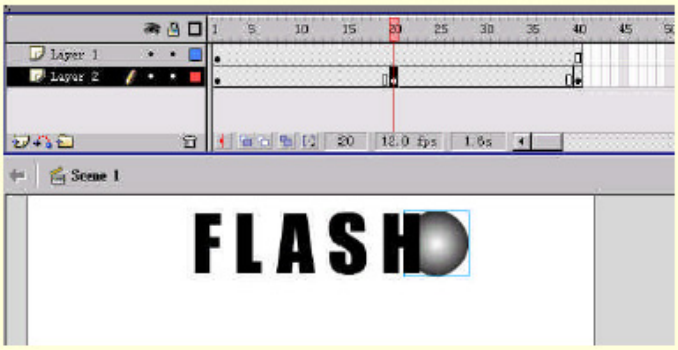


图 5-29　添加并编辑帧

（4）把 Layer2 层的前两帧都设为移动渐变，使 Layer2 层的圆产生先向左移动再向右移动的效果，如图 5-30 所示。

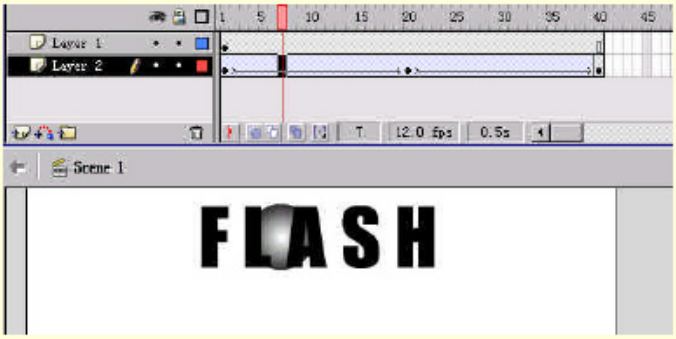


图 5-30　步骤（4）的时间轴

（5）在 Layer1 层上右击鼠标，选择 Mask（屏蔽）菜单项。其结果如图 5-31 所示。

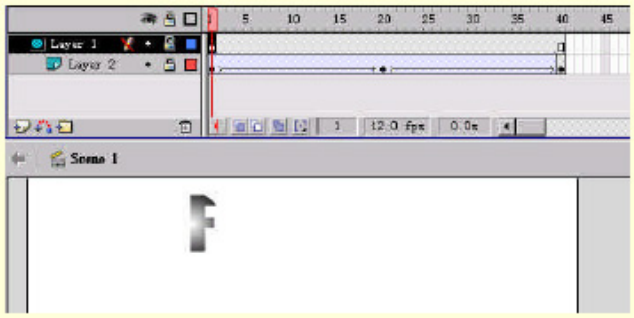


图 5-31　使用屏蔽后的效果

（6）再插入一个层 Layer3，把 Layer1 的所有帧复制到该层，并将文字改为亮灰色，然后将 Layer3 移动到最下一层，如图 5-32 所示。

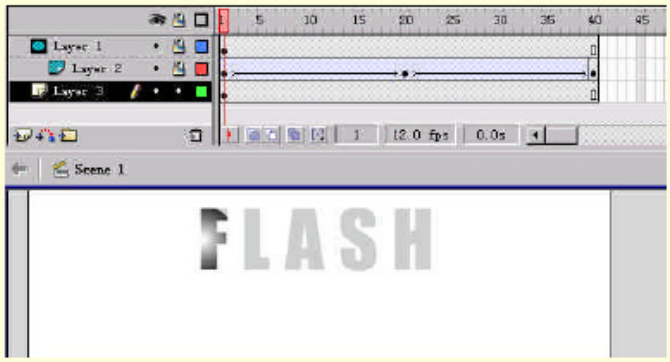


图 5-32　添加 Layer3 层

（7）选择 Modify | Document 菜单项，将电影的背景设置为灰色，如图 5-33 所示。

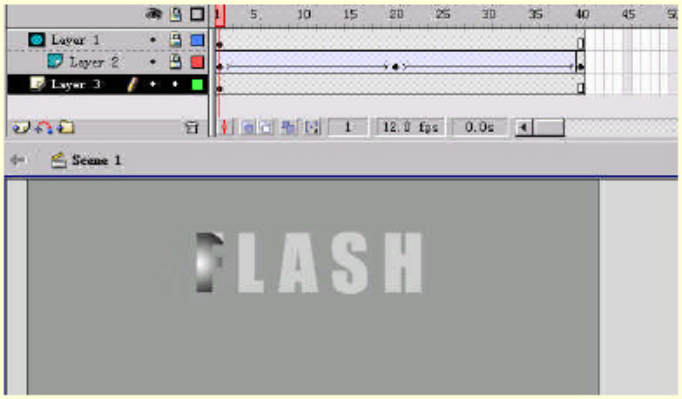


图 5-33　改变背景颜色

（8）输出并运行动画，将出现类似于一束灯光照亮文字的效果，如图 5-34 所示。

图 5-34　灯光效果

在 Flash MX 中，利用屏蔽可以创造出很多意想不到的效果。但是，刚开始使用屏蔽的时候，最让人糊涂的地方就是到底是“谁”屏蔽“谁”？其实，只要记住“屏蔽层提供外形，被屏蔽层提供颜色”，就可以随心所欲地使用屏蔽。

5.2.4 设定层的属性

在层上右击鼠标，选择快捷菜单中的 Properties（属性）选项，在弹出的如图 5-35 所示的对话框中设定层的属性。

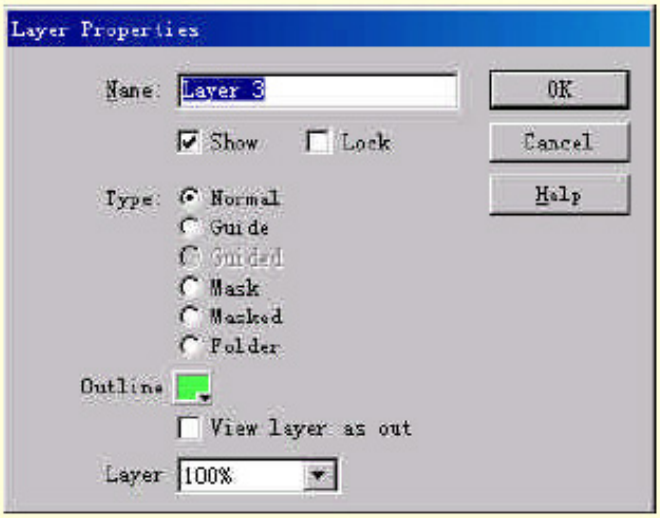


图 5-35 层的属性对话框

在对话框中可以设置：

- Name（层名）
- Show（显示层）
- Lock（锁定层）
- 层的类型
  - Normal（正常）
  - Guide（向导）或 Guided（被引导）
  - Folder（层文件夹）

向某一层添加向导层后，向导层前将显示一曲线图标，并且与被引导的层之间用虚线隔开，如图 5-36 所示。

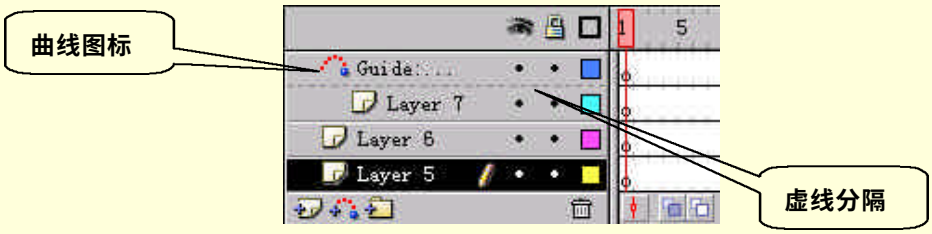


图 5-36 正常的向导层

当把被引导层删除或直接把某一层设定为向导层后，向导层的曲线图标将变成一把小锤子，如图 5-37 所示。这表明向导层没有引导对象，需要把它的下一层属性设置为 Guided（被引导）层。

图 5-37　没有引导对象

- 对于同一个向导层可以设定多个被引导层，从而使这几个被引导层共享一条路径，请参考图 5-38。

图 5-38　一个向导层和多个被引导层

- Mask（屏蔽）或 Masked（被屏蔽）

与向导层和被引导层的关系类似，屏蔽也可以设置为一层屏蔽多个层，其规律还是“屏蔽层提供外形，被屏蔽层提供颜色”，如图 5-39 所示。

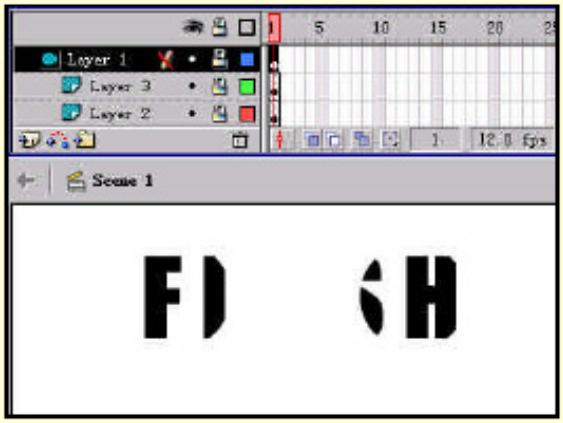


图 5-39　一层屏蔽多个层

- Outline（轮廓）显示及颜色设定。
- 层的显示大小：设定层的高度，如图 5-40 及图 5-41 所示。

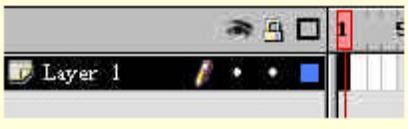


图 5-40　100%

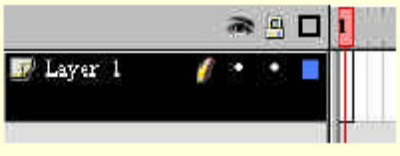


图 5-41　200%

5.3 使用元件

5.3.1 元件的概述

Symbol（元件）在 Flash MX 中具有十分重要的意义。元件一旦被创建，Flash MX 将自动把它添加到元件库中，从而可以在同一个甚至任何一个 Flash MX 文件中反复使用。如果元件被重新编辑了，那么在整个 Flash MX 电影文件的所用到的同一个元件都将自动更新，这一特点加速了 Flash MX 动画的制作与编辑过程。在元件库中的元件也可以组织分类，管理和使用都十分方便。

使用元件可以减小 Flash MX 最终输出动画文件的大小。同时，在播放 Flash MX 动画的时候，元件只下载一次，有利于动画文件在互联网上的传递与播放。

另外，将 Flash MX 文件以 Share Library（共享库）的形式打开，就可以在一个 Flash MX 文件中使用另一个 Flash MX 文件的元件。

Flash MX 中主要有三种元件：Graphic（图形）元件、Movie Clip（影片剪辑）、Button（按钮）元件。此外，从外部导入的图片、声音、视频等文件也作为元件存储在 Flash MX 文件中。关于外部文件的导入，将在后面的章节中进行介绍。

5.3.2 创建元件

选择 Insert | New Symbol 菜单命令或者使用 Ctrl+F8 快捷键，将弹出如图 5-42 所示的 Create New Symbol（创建新元件）对话框。

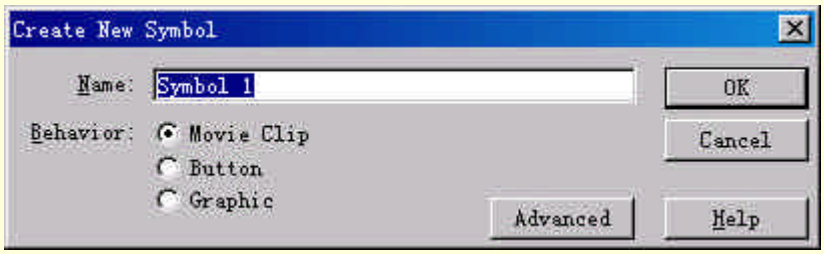


图 5-42 创建新元件对话框

在 Name（名称）的文本框中，输入要创建的元件的名字，也可以使用系统的默认值 Symbol X。然后选择所要创建元件的类型，包括 Movie Clip（影片剪辑）、Button（按钮）元件、Graphic（图形）元件。选好元件的类型后按 OK 按钮，即可创建并编辑元件。

Flash MX 还可以把当前舞台上的内容转变为元件。先选中要转变的对象，如图 5-43 所示。

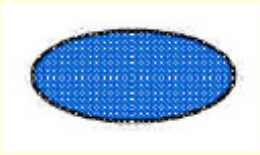


图 5-43 选中要转变的对象

然后选择 Insert | Convert to Symbol 菜单项或者使用 F8 快捷键，将弹出 Convert to Symbol（转换为元件）对话框。选择元件的类型后单击 OK 按钮，则选中的图形将转变为如图 5-44 所示的元件，并自动添加到当前位置。

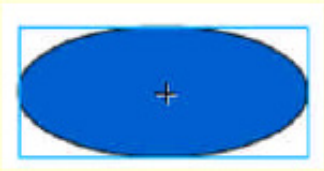


图 5-44　转变成为元件

后面几节中将分别介绍每一种元件的功能及特点。

5.3.3　影片剪辑

在影片剪辑中创建或者编辑动画与在 Flash MX 的舞台上完全相同，影片剪辑中可以使用帧渐变、层等一切 Flash MX 的动画效果，就相当于一段嵌入到 Flash 动画中的 Flash 动画，而且影片剪辑的播放与其他动画的播放无关。因此，利用它可以实现极为复杂的运动及相对运动的功能，这里将给出一个示意太阳、地球、月亮运动关系的例子。

（1）新建一个 Flash MX 文件，创建一个 Movie Clip（影片剪辑）命名为“ earth ”，如图 5-45 所示。

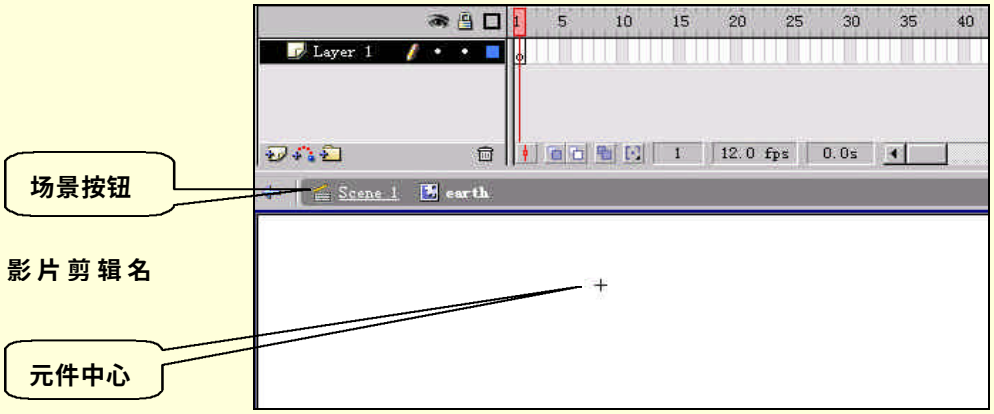


图 5-45　创建影片剪辑

✎ 进入影片剪辑编辑状态之后，Flash MX 的舞台区域将变为影片剪辑的编辑区域。区域中“ 十 ”字表示元件的中心位置，元件的缩放、旋转、镜像等效果都以此为中心。要十分注意元件的中心位置，否则，当元件被大量使用以后，调节中心将是很麻烦的事情。

（2）以元件中心为圆心，在影片剪辑的第 1 帧绘制一个填充色为蓝色/黑色径向渐变的圆表示“ 地球 ”，在影片剪辑中插入一个新层 Layer2，在“ 地球 ”的旁边绘制一个白色/黑色径向渐变的小圆表示“ 月球 ”，如图 5-46 所示。

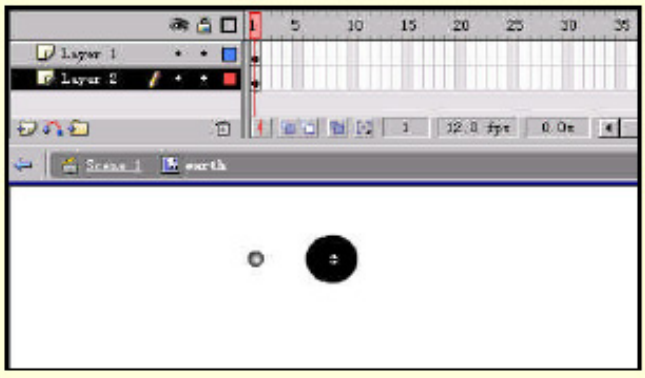


图 5-46　绘制“ 地球 ”和“ 月球 ”

（3）在“ 月球 ”所在层 Layer2 上添加 Guide Layer（向导层），并绘制一条椭圆的轨道，然后把月

球放置在“轨道”上，如图 5-47 所示。

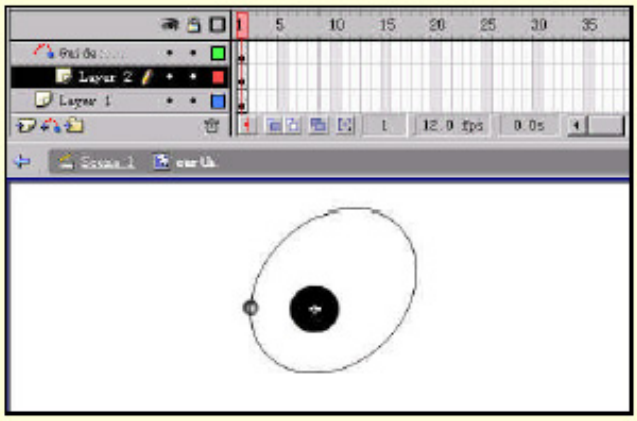


图 5-47 建立“月球”轨道

（4）右击“月球”所在层 Layer2 上的第 1 帧，在弹出的快捷菜单上选择 Create Motion Tween 菜单项，然后，分别在第 10 帧、第 20 帧、第 30 帧处创建关键帧，如图 5-48 所示。

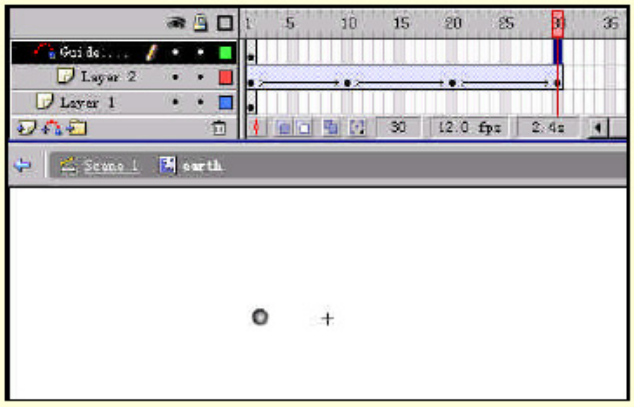


图 5-48 创建“月球”关键帧

（5）在第 30 帧处，分别为向导层和“地球”层插入帧，并将“月球”的第 10 帧和第 20 帧移动到图 5-49 和图 5-50 所示的位置。

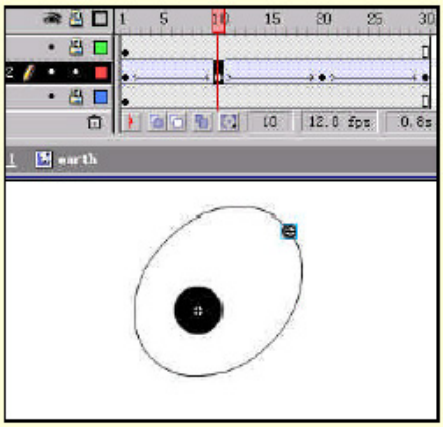


图 5-49 第 10 帧“月球”位置

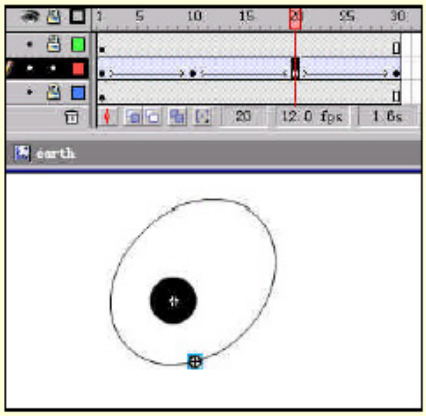


图 5-50 第 20 帧“月球”位置

（6）单击 Scene（场景）按钮，回到 Flash MX 舞台，在第 1 帧上以红色/黑色的径向渐变为填充色绘制一个大圆表示“太阳”，如图 5-51 所示。

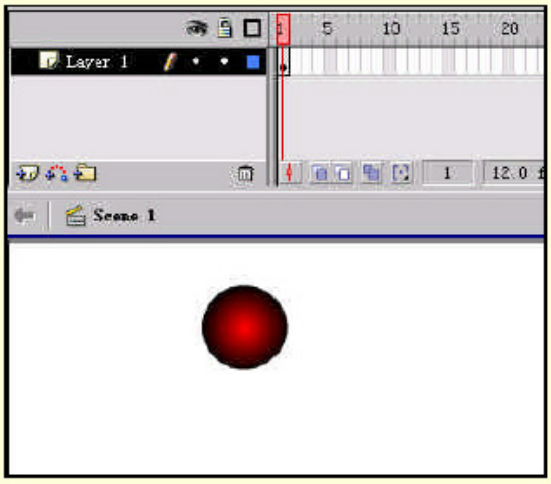


图 5-51　绘制“太阳”

（7）选择 Windows | Library 菜单项，开启如图 5-52 所示的元件库管理窗口，刚刚编辑好的影片剪辑 earth 将出现在元件库管理窗口中。单击播放按钮，可以预览影片剪辑。

- 利用元件库管理窗口按钮或 Option（选项）菜单也可以便捷地创建、删除、复制、重命名、编辑元件，或者对元件进行分类、修改属性等高级操作。其中，Option（选项）菜单的主要选项及功能见表 5-1。

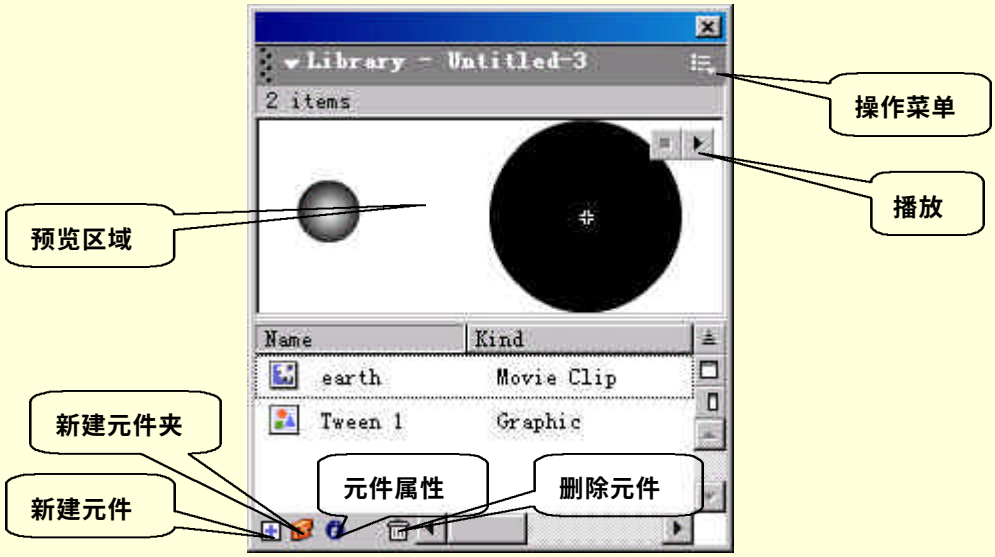


图 5-52　元件库管理窗口

表 5-1　　操作菜单的主要选项及功能

菜　单　项	说　　明
New Symbol	新建元件
New Folder	新建元件夹，它可以把元件按树状结构分类，以便于管理
New Font	新建字体元件
New Video	新建视频元件
Rename	重命名元件
Move to New Folder	把选中的元件移动到新的元件夹中
Duplicate	复制元件

(续表 5-1)

Delete	删除元件
Edit	编辑元件
Edit with	用指定的应用程序编辑从外部导入的文件
Properties	元件属性
Linkage	设置影片剪辑的连接名称
Component Definition	设置影片剪辑的参数
Select Unused Item	选中没有使用的元件
Update	更新元件
Play	播放
Expand Folder	展开当前元件夹
Collapse Folder	折叠当前元件夹
Expand All Folder	展开所有元件夹
Collapse All Folder	折叠所有元件夹
Shared Library Properties	共享库参数属性
Keep Use Counts Updated	始终刷新元件使用次数
Update Use Counts Now	立即计算元件使用次数

（8）新建一层，把 earth 影片剪辑用鼠标拖动到该层上，如图 5-53 所示。

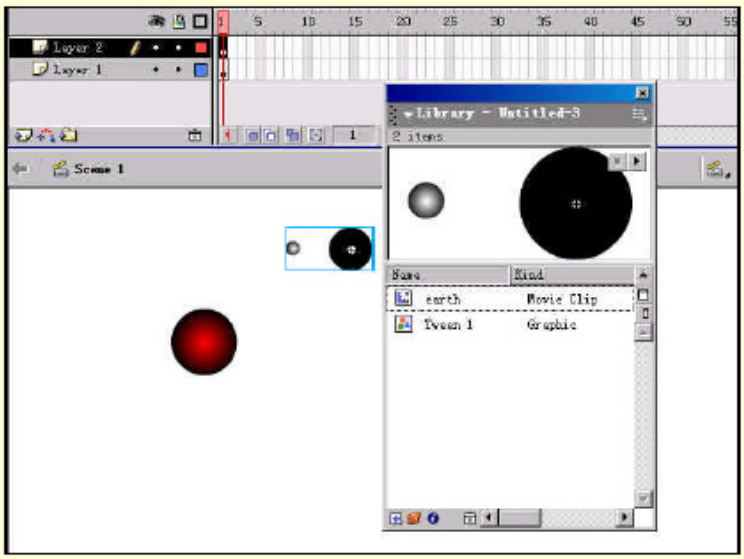


图 5-53　添加影片剪辑

（9）在 Layer2 层上创建 Guide Layer（向导层），并绘制一个椭圆代表太阳轨道，然后将 earth 电影剪辑放置在轨道上，如图 5-54 所示。

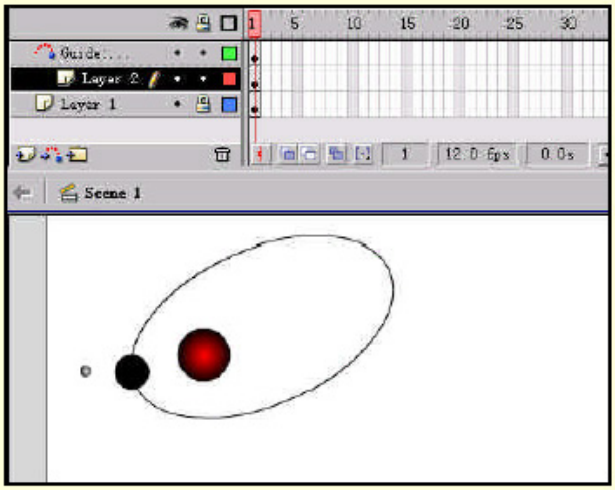


图 5-54　将“地球”放置到轨道上

（10）右击 Layer2 层上的第 1 帧，在弹出的快捷菜单中选择 Create Motion Tween 菜单项，然后，分别在第 30 帧、第 60 帧、第 90 帧处创建关键帧，在向导层和“太阳”层 Layer1 的第 90 帧处添加帧，如图 5-55 所示。

（11）将“地球”层的第 30 帧和第 60 帧移动到图 5-56 及图 5-57 所示的位置。

（12）输出并运行动画。月球在围绕地球转的同时也随地球围绕太阳转。

由此可见，影片剪辑中的动画是完全独立的，在影片剪辑运动的同时，动画的播放完全不受影响。利用这项功能可以很容易实现人走路、鸟飞翔等复合的运动。

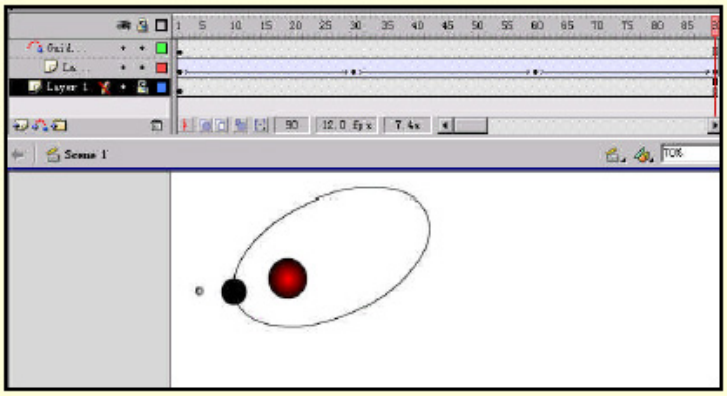


图 5-55　添加帧和关键帧

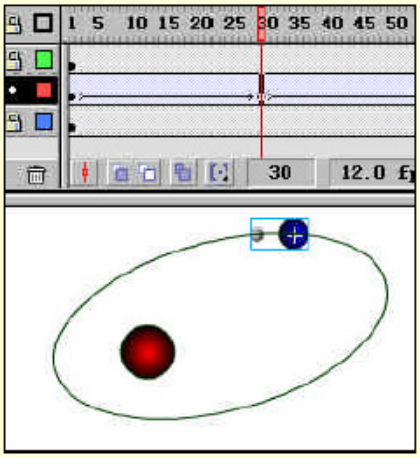


图 5-56　第 30 帧的“地球”位置

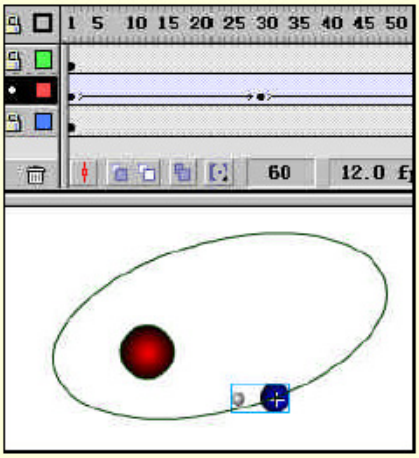


图 5-57　第 60 帧的“地球”位置

5.3.4 图形元件

与影片剪辑类似，在图形元件中也可以使用帧渐变、层等一切 Flash MX 的动画效果。它与影片剪辑最主要的区别是：影片剪辑是完全独立的，它的播放不受其他 Flash MX 动画的影响；而图形元件一旦被嵌入 Flash MX 动画中，它将随着主动画所播放的帧的相对位置而改变，即主动画播放到哪一帧，图形元件也相对地显示哪一帧。

（1）新建一个 Flash MX 文件，创建一个图形元件，并制作一段来回摆动的球的动画，摆动周期为 20 帧，如图 5-58 所示。

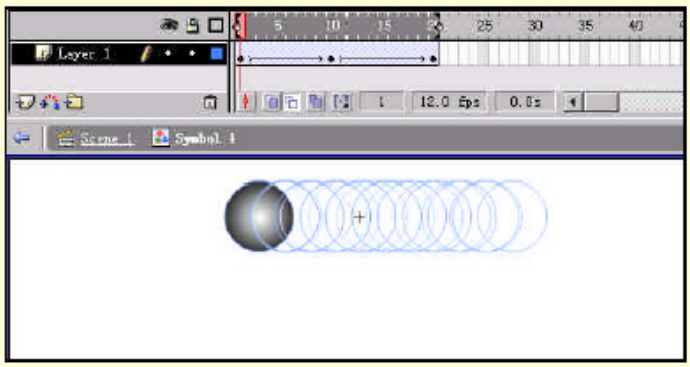


图 5-58 创建一个图形元件

（2）将图形元件移动到 Flash MX 的舞台上，输出并运行动画，但是，球并不运动，如图 5-59 所示。

图 5-59 球不运动

（3）如果在 Flash MX 舞台上的第 10 帧处插入帧，如图 5-60 所示。球只做单向运动，如图 5-61 所示。

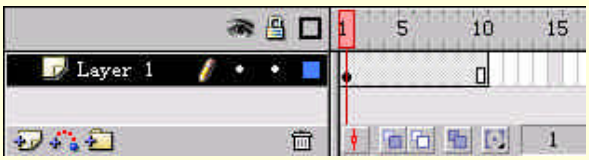


图 5-60 在第 10 帧处插入帧

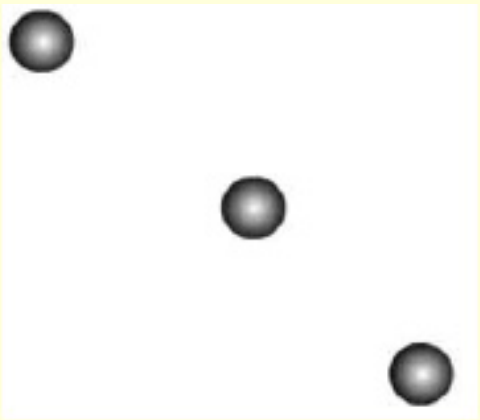


图 5-61 球只做单向运动

（4）按照图 5-62 的方法在 Flash MX 舞台上的第 20 帧处插入帧。球可以往复运动。

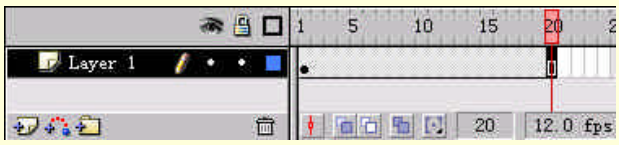


图 5-62　第 20 帧处插入帧

可见，图形元件的这种动画特性并不好掌握，有时候甚至会带来很多麻烦，所以，除非有特殊用途，一般不使用图形元件的动画功能，但是，用图形元件制作和编辑静态图片是一种好的习惯。尤其是当某些图形在同一动画文件中重复出现的时候，使用图形元件可以大大减小 Flash MX 动画的体积，而且，只要修改图形元件，此图形在动画中的所有副本都将自动更新。

5.3.5　按钮元件

按钮是 Flash MX 中特殊的元件，它可以响应鼠标操作，是创建交互式 Flash MX 动画的基本元件。新建一个按钮后，将进入到按钮的编辑状态，如图 5-63 所示。

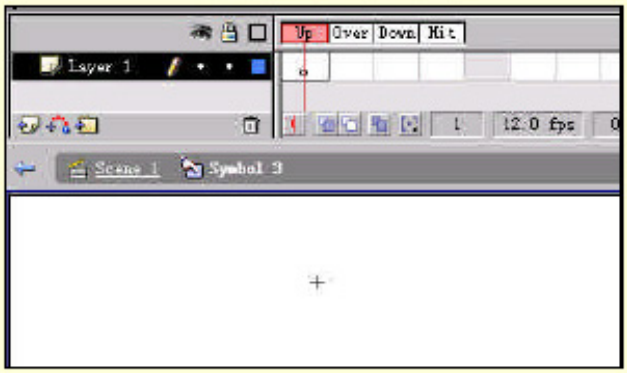


图 5-63　编辑按钮

不同于影片剪辑和图形元件，每一个按钮元件只有 4 帧。

- 前 3 帧代表按钮的 3 种不同状态：
  - Up（抬起）：在通常情况下，按钮所处的状态。
  - Over（掠过）：当有鼠标放置在按钮上面时，按钮将自动地显示这一帧。
  - Down（按下）：按钮被按下时，将自动显示这一帧。
- 第 4 帧表示按钮的响应区域：只有当鼠标进入到这一区域时，按钮才开始响应鼠标的动作。这一帧仅仅代表了一个区域，它永远不会被显示出来。

另外，在按钮中也支持层的使用，一切 Flash MX 层的效果都可以实现。这里给出一个按钮的例子。

（1）新建一个 Flash MX 图形元件，绘制一个蓝色/白色径向渐变填充色的圆，并将圆的一半选中移开一段距离，如图 5-64 所示。

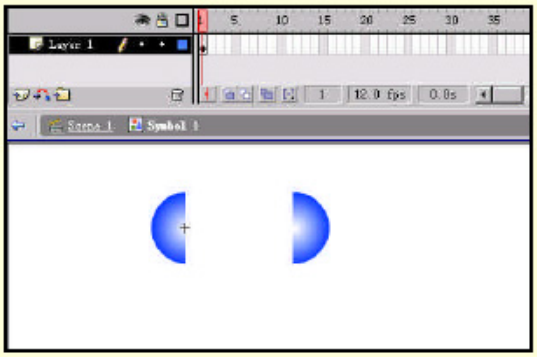


图 5-64　绘制并分开圆

（2）在两个半圆的之间，绘制一个如图 5-65 所示的白色/蓝色/白色渐变填充色的矩形。

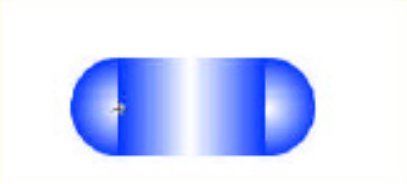


图 5-65 绘制矩形

（3）调整矩形的填充色的位置，使它按图 5-66 的方式与两个半圆相接合。

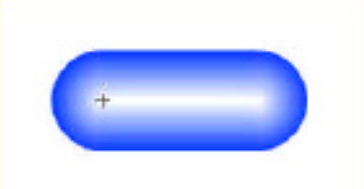


图 5-66 调整后矩形的填充色

（4）把整个图形组成一个组，并将其中心移动到“十”字处，如图 5-67 所示。

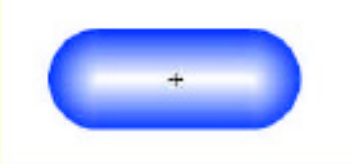


图 5-67 调整中心

使用 Window | Align 菜单命令或者 Ctrl+K 快捷键打开 Flash MX 的 Align（排列）面板，利用它可以准确地排齐元件，如图 5-68 所示。排列面板中每个按钮的功能见表 5-2。

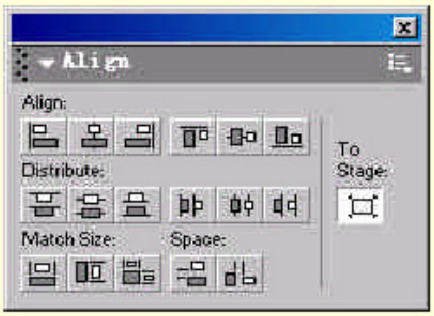


图 5-68 排列面板

表 5-2 排列面板的功能

按钮图标	注 释	说 明
Align		
	Align left edge	使选中的对象左边缘对齐
	Align horizontal Center	使选中的对象垂直居中对齐
	Align right edge	使选中的对象右边缘对齐
	Align top edge	使选中的对象上边缘对齐
	Align vertical Center	使选中的对象水平居中对齐

（续表 5-2）

	Align bottom edge	使选中的对象下边缘对齐
Distribute		
	Distribute top edge	以选中的对象左边缘为标准，垂直均匀分布各选中对象
	Distribute vertical Center	以选中的对象中间位置为标准，垂直均匀分布各选中对象
	Distribute bottom edge	以选中的对象右边缘为标准，垂直均匀分布各选中对象
	Distribute left edge	以选中的对象上边缘为标准，水平均匀分布各选中对象
	Distribute horizontal Center	以选中的对象中间位置为标准，水平均匀分布各选中对象
	Distribute right edge	以选中的对象下边缘为标准，水平均匀分布各选中对象
Match		
	Match width	使选中的对象宽度相同
	Match height	使选中的对象高度相同
	Match width and height	使选中的对象宽度、高度都相同
Space		
	Space evenly vertically	使选中的对象垂直等间距排列
	Space evenly horizontally	使选中的对象水平等间距排列
To Stage		
	Align/Distribute to stage	以场景为参考，排列或分布对象。使用此选项可以很容易地把对象排到场景的中间

（5）创建一个按钮元件，把刚刚建立的图形元件放置在按钮的 Up 帧和 Down 帧中，注意居中对齐，如图 5-69 所示。

（6）新建一个影片剪辑，在第 1 帧放入图形元件 Symbol 1，在第 15 帧、第 30 帧处插入关键帧，注意居中对齐，如图 5-70 所示。

（7）把第 15 帧的图形放大，并对第 1 帧和第 15 帧使用位置渐变，制成一个先由小变大再由大变小的影片剪辑，如图 5-71 所示。

（8）回到编辑按钮的状态，把影片剪辑插入到 Over 帧处，同时居中排齐。

可以使用元件编辑按钮实现元件之间的切换编辑，单击元件编辑按钮，在弹出的快捷菜单中，将列出当前文件中所有的可编辑 Flash MX 元件，不包括从外部导入的图形声音等元件，如图 5-72 所示。

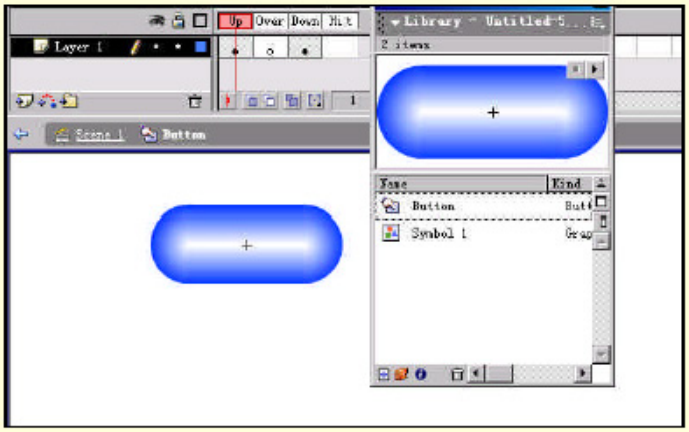


图 5-69　放置图形

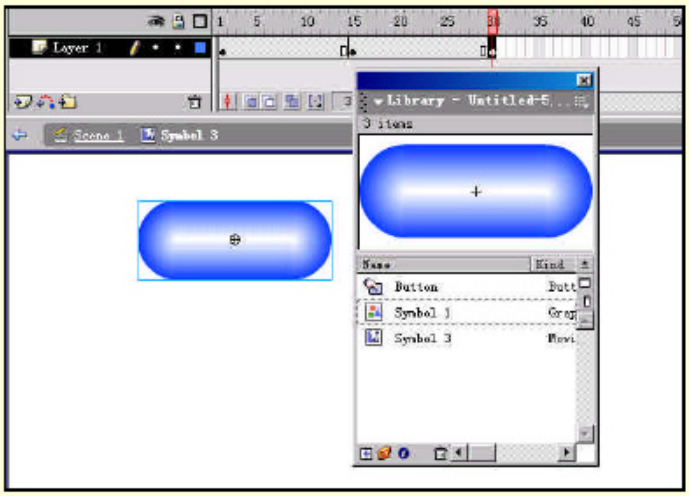


图 5-70　建立影片剪辑

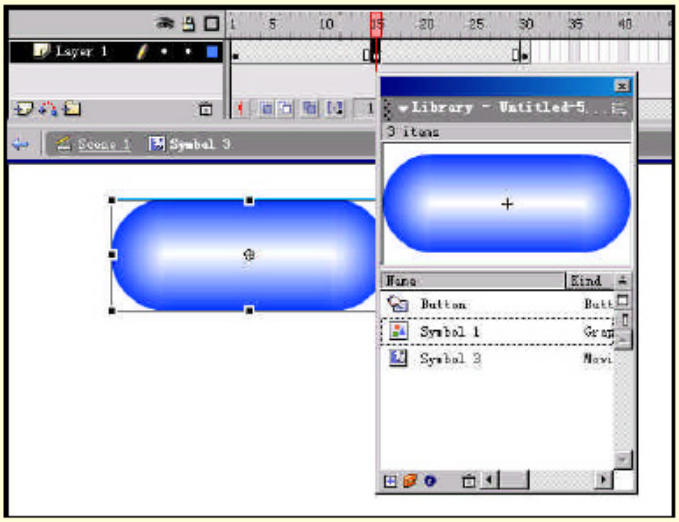


图 5-71　制成动画

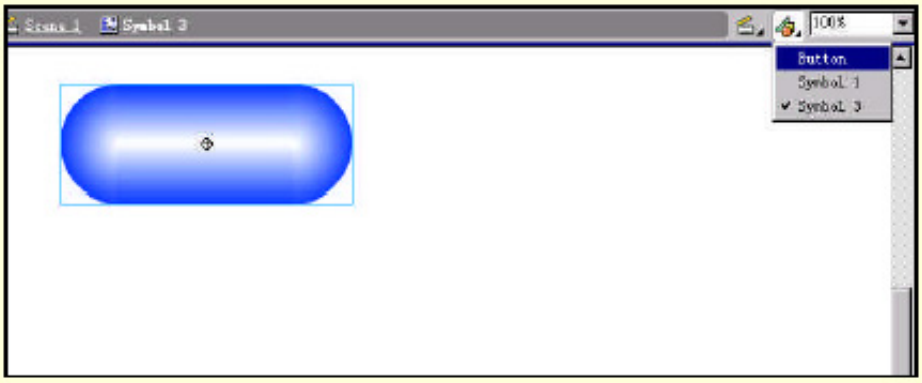


图 5-72　元件编辑按钮

（9）新建一个层，在 Up 帧处输入绿色的文字“开始”，并调整好大小和位置。把此层的 Up 帧复制到 Over 帧和 Down 帧处。将 Over 帧的文字设为橙色，Down 帧的文字设为黄色，如图 5-73 所示。

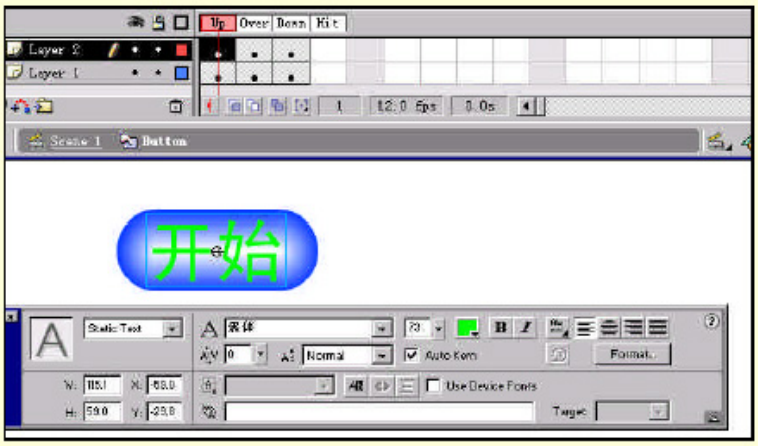


图 5-73　添加文字

（10）在 Layer1 层的 Hit 帧处插入帧，设定按钮的响应范围，如图 5-74 所示。

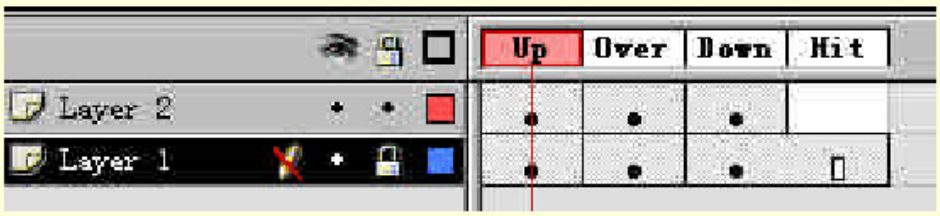


图 5-74　设定响应范围

（11）回到场景，把按钮拖拽到舞台中，输出并运行动画。按钮的 3 种状态如图 5-75、图 5-76 及图 5-77 所示。

图 5-75　抬起

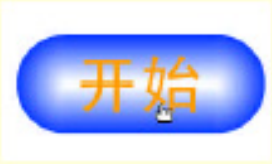


图 5-76　掠过

图 5-77 按下

可见，虽然按钮元件不具有制作动画的特性，但是通过在按钮帧中插入影片剪辑，同样可以制作出动画按钮。

5.3.6 通用元件库的使用

Flash MX 提供了几个通用元件库，搜集了很多精美的元件。用户可以将它们嵌入到自己的文件中。选择 Window | Common Library 菜单下相应的子菜单即可使用通用元件库，如图 5-78 所示。通用元件不可编辑，其他用法与使用自己建立的元件库完全相同。

图 5-78 通用元件库

第 6 章　交互式 Flash 动画基础

6.1　综　述

对于网页上的动画，交互功能是必不可少的。与 Flash 5 相比，Flash MX 增加了很多函数和动作，可以实现比 Flash 5 更为复杂的交互。在 Flash MX 中还沿用了 ActionScript 脚本语言和专家模式的动作编辑面板，这使创建交互式的 Flash MX 动画更像用 Visual Basic 编写程序。本章将详细介绍 ActionScript 脚本语言的使用方法。

6.2　使用动作面板

6.2.1　动作面板简介

在 Flash MX 中只有影片剪辑、按钮和帧可以响应动作。前二者是用户与 Flash MX 交互的基础；后者可以控制 Flash MX 动画的播放。在帧或按钮上右击鼠标，选择快捷菜单中的 Action（动作）选项，将显示出动作面板。它可以完成所有 ActionScript 脚本语言的编写。图 6-1 所示的是动作面板的标准模式，适合于初级用户或制作简单的交互。Flash MX 还可以使用专家模式的动作面板，适合于熟悉 ActionScript 脚本语言的用户制作复杂的交互。

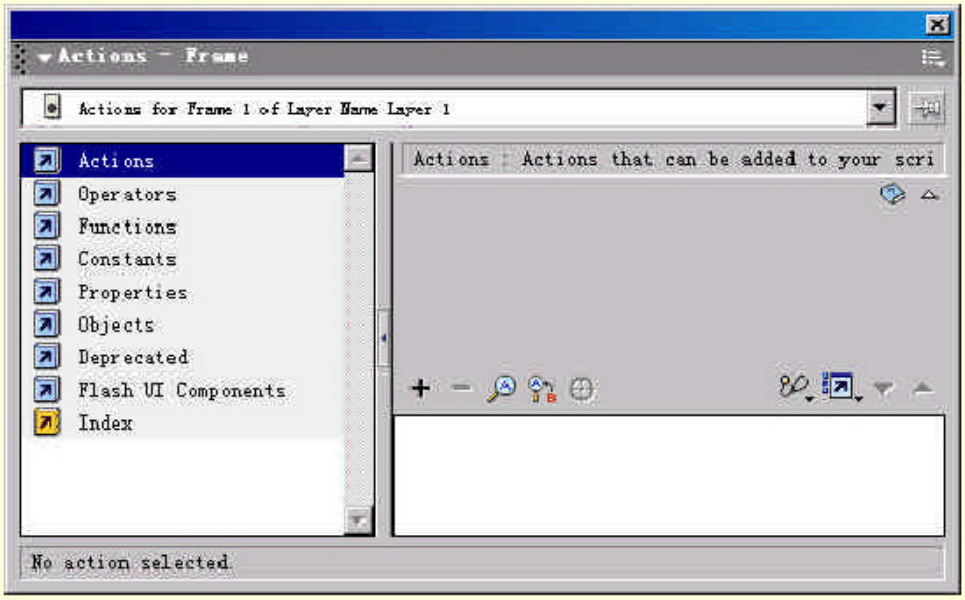


图 6-1　动作面板

6.2.2　标准模式编辑

Flash MX 动作面板的标准模式与 Flash 5 相似，用户只需要在动作列表框中指定动作及参数，就可以实现 ActionScript 语言的编写。

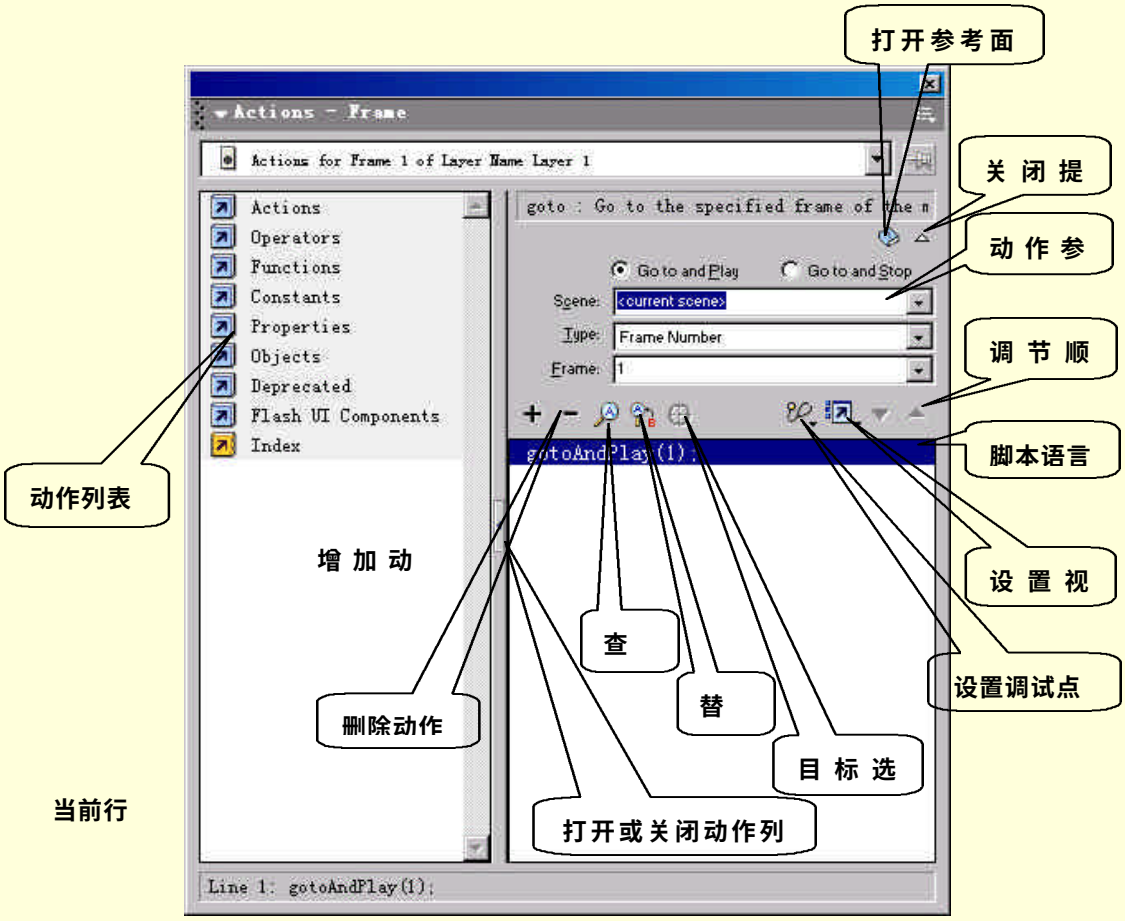


图 6-2　标准模式动作面板的功能

图 6-2 所示的 Flash MX 标准模式动作面板具有如下功能。

- 动作列表：双击列表中的选项，便可以逐层展开列表，最后双击想要的动作或函数，将其添加到脚本中。
- 增加动作：用“ + ”按钮可以按图 6-3 所示的方法以选择菜单的形式添加动作或函数。
- 删除动作：选中一行或多行脚本程序后，按“ - ”按钮或 Delete 键，便可以删除动作。
- 添加参数：有些 Flash 动作或函数需要参数，选中它们以后，参数输入区域将按照图 6-4 的样子自动更新，以便添加所需的参数。其中用红色标出的就是尚未添加参数的部分。

输入区域中的 Expression 选项用于标识参数的类型。选中此选项，则表明所添加的参数是一个表达式，反之，则表示所添加的参数是一个字符串。

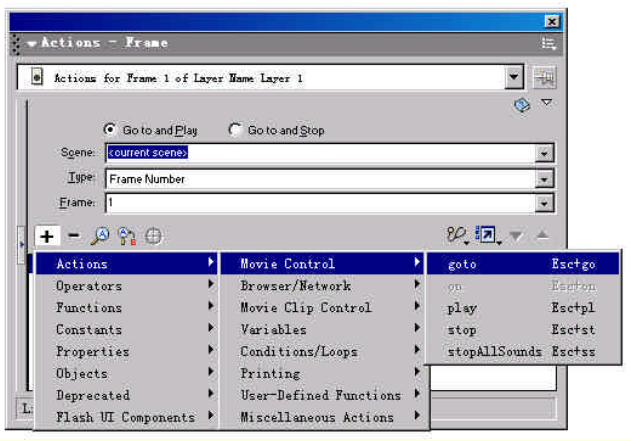


图 6-3　动作菜单

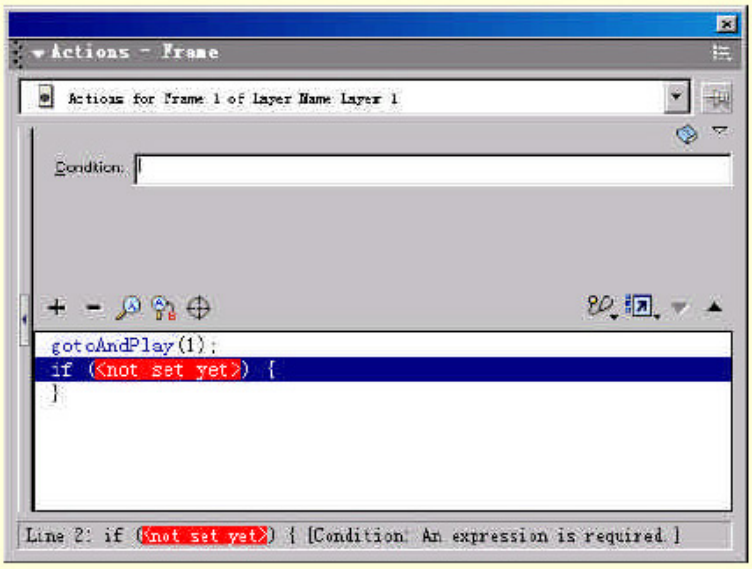


图 6-4　添加参数

- 展开/关闭提示区域：单击展开/关闭提示区域按钮，可以展开或关闭命令提示区域。
- 打开 Reference（参考）面板：单击此按钮可以打开 Flash MX 中新增加的 Reference（参考）面板，在该面板中可以看到针对当前所选命令的参考文档，如图 6-5 所示。
- 目标选择：在 Flash MX 的脚本中经常需要对动画中的对象进行操作。使用此按钮可以浏览动画中所有的对象，并将其加入到脚本中，如图 6-6 所示。
- 查找和替换：单击查找按钮，会弹出如图 6-7 所示的 Find（查找）对话框，在文本框内可输入要查找的文本；单击替换按钮，会弹出如图 6-8 所示的 Replace（替换）对话框，在两个文本框中可分别输入被替换和替换后的文本。
- 设置调试点：单击设置调试点按钮可在当前行添加或删除调试点。
- 设置视图：单击设置视图选项可以在标准模式和专家模式之间进行切换。

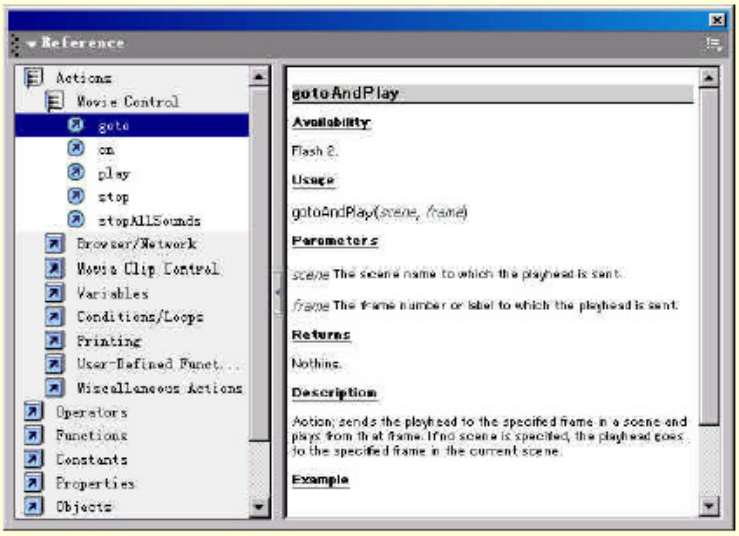


图 6-5　Reference（参考）面板

图 6-6　目标选择对话框

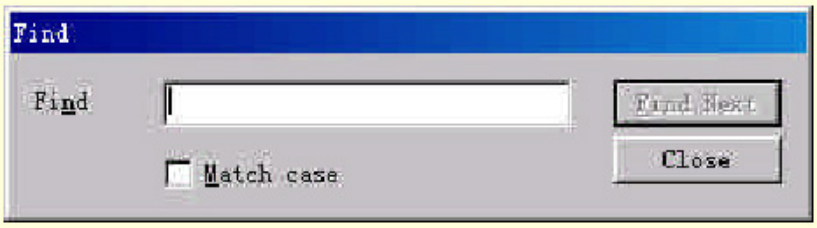


图 6-7　Find（查找）对话框

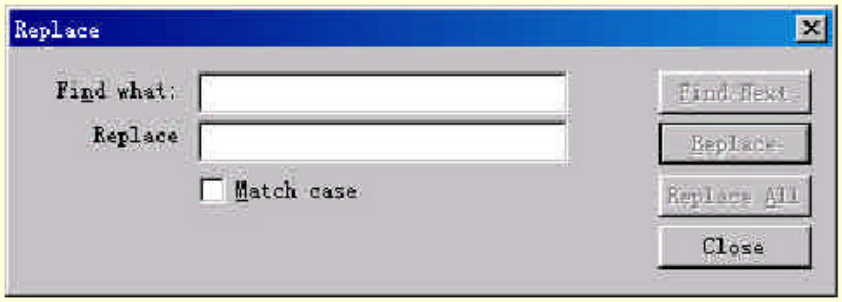


图 6-8　Replace（替换）对话框

6.2.3　专家模式编辑

按照图 6-9 所示使用视图设置菜单的 Expert Mode 选项，可以把动作面板切换到专家模式。

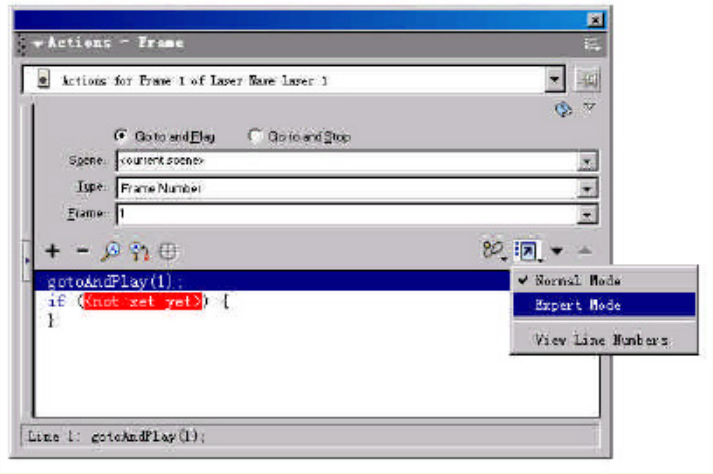


图 6-9　视图设置菜单

动作面板的专家模式同编写程序一样，可以按图 6-10 所示的模式直接键入 ActionScript 脚本语言。

这种模式更适合于编写大量的复杂的脚本。但是，它要求用户完全掌握 ActionScript 脚本语言的语法，一旦编写的 ActionScript 脚本语言有语法错误，Flash MX 动画将不能正常运行。

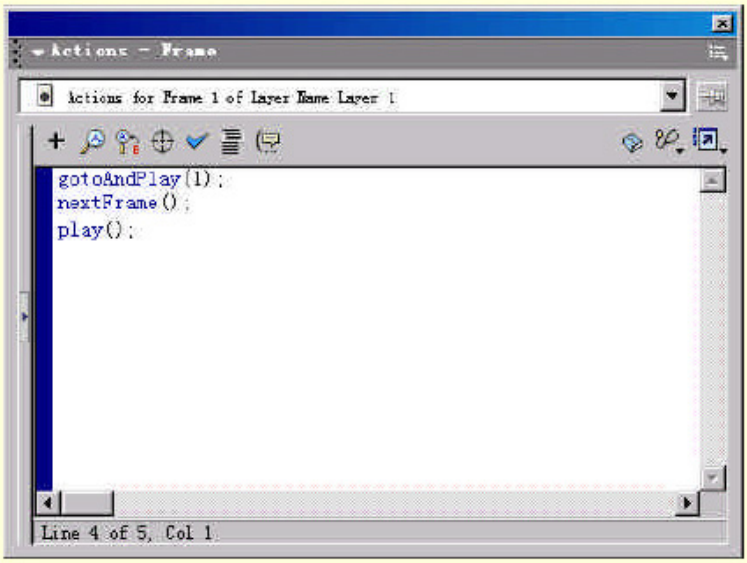


图 6-10　专家模式

6.2.4　动作面板设置菜单

图 6-11 是动作面板设置菜单，其功能如下。

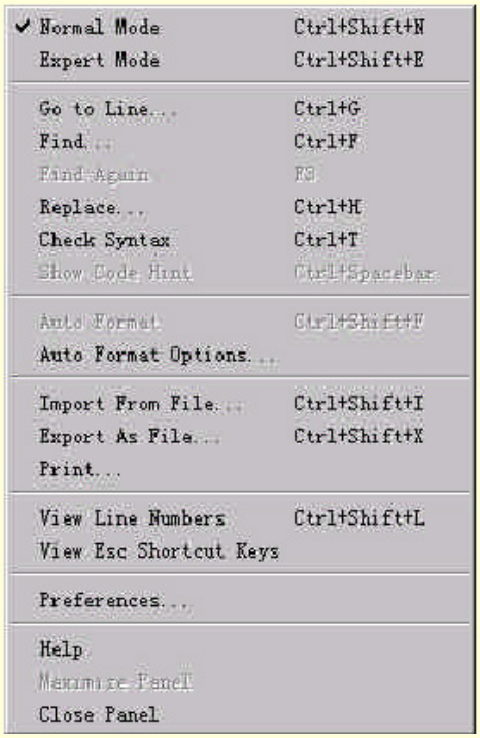


图 6-11　动作面板设置菜单

- 模式切换：进行 Normal Mode（标准模式）和 Expert Mode（专家模式）之间的切换。
- Goto Line（转移定位）：在对话框中输入行数，便可调转到指定的位置。
- Find（查找）和 Replace（替换）：查找或替换 ActionScript 脚本中的指定文本。
- Check Syntax（错误检查）：检查编写的 ActionScript 脚本中是否含有语法错误，以便修改。
- Show Code Hint（显示代码提示）：类似于 Visual Basic 中的代码自动完成功能。

- Auto Format（格式化）：以 Flash MX 默认或用户自己设定的格式对脚本进行格式化，以方便对脚本的查看、修改和调试。
- 脚本的导入与导出：ActionScript 脚本可以 Export As File（导出到文件）导出，或者以 Import From File（从指定文件导）导入。ActionScript 脚本文件的扩展名为 as。配合 Include 动作，可以使不同位置的 ActionScript 脚本共享相同的代码。
- Print（打印）。
- View Line Numbers（显示每行行数）：在脚本每一行命令前面加一个数字标注该行行数。
- Preferences（选项）：在此可以设定在命令编辑模式下的一些个人偏好选项，如代码字体、字号、代码颜色等。

6.3 ActionScript 脚本语言的编写

6.3.1 ActionScript 脚本语言的语法

使用过 Normal Mode（标准模式）的动作面板后，读者对 ActionScript 脚本语言有了初步的认识。下面再来看一段典型的 ActionScript 脚本（请暂时不要关心这些脚本实现的功能）：

```
var temp_pt = [];
var pt2d = [];
for (n=0; n<8; n++) {
    temp_pt.push([point[n][0]]);
    temp_pt[n][1] = point[n][1]*Math.cos(angle)-point[n][2]*Math.sin(angle);
    temp_pt[n][2] = point[n][1]*Math.sin(angle)+point[n][2]*Math.cos(angle);
}
for (n=0; n<8; n++) {
    pt2d.push([temp_pt[n][1]*0.707-temp_pt[n][0]*0.707]);
    pt2d[n][1] = temp_pt[n][2]*0.816-temp_pt[n][0]*0.408-temp_pt[n][1]*0.408;
    this["p" add n].num = n;
    setProperty ("p" add n, _x, pt2d[n][1]+xpos);
    setProperty ("p" add n, _y, pt2d[n][0]+ypos);
}
```

这段脚本中包括：变量的定义、循环、数组的使用以及函数的调用。每一行脚本都用分号分隔，从语言结构的角度来看，ActionScript 脚本类似于 C（或者 C++），或者更确切的说应该是 Java（ActionScript 脚本没有指针的概念）。所以，学过 C 语言或 Java 的人，很容易掌握 ActionScript 脚本语言。对于没有编程基础的人来说，使用标准模式的的动作面板，也可以逐渐熟悉并掌握 ActionScript 脚本语言。

6.3.2 数据类型

Flash MX 的 ActionScript 脚本支持的数据类型可分为两类：primitive（简单变量）和 reference（参数）。

primitive（简单变量）中的每个变量只存储一种数据，包括：

- String（字符串）

一系列字符（例如：字母、数字、标点符号等）的有序组合。当一个或几个字符被双引号引用时，ActionScript 脚本就把它作为一个字符串来处理。例如：value="Hello"表示通过‘ = ’运算符将字符串"Hello"赋值给变量 value。

在字符串中的右斜杠（\）称作“ 换码符 ”，利用“ 换码符 ”可以表示出一些无法键入或表示的符号。

例如：value="\r"表示把“回车”符赋值给变量 value。表 6-1 列出了“换码符”所表示的所有字符。熟悉 C 语言的人会明白，ActionScript 脚本的换码字符和 C 语言的规则基本相同。

表 6-1 换码字符

换码字符	含 义	换码字符	含 义
\b	退格（BackSpace）	\’	单引号
\f	走纸换页	\\	左斜杠
\n	换行	\000 - \377	1~3 位 8 进制数所代表字符
\r	回车	\0x0 - \0xFF	1~2 位 16 进制数所代表字符
\t	横向跳格（Tab）	\u0000 - \uFFFF	1~4 位 16 进制数所代表字符
\”	双引号		

- Number（数字）

ActionScript 脚本中所有的数字都是双精度的浮点数。用户可以使用基本的数学运算符处理数字。包括：加（+）、减（-）、乘（*）、除（/）、取余（%）、加 1（++）、减 1（--）。同时，用户也可以使用数学对象（Math Object）对数字进行高级运算（关于对象的使用，将在 6.3.9 节中说明）。例如：

```
value=1+2；    //将 1+2 的结果赋予变量 value
value++；      //变量 value 加 1
value=Math.sqrt(value);   //求 value 的平方根，结果为 2
```

和 C++语言的注释方法相同。在连续两个右斜杠后面的一行文字表示 ActionScript 脚本的注释。

- Boolean（布尔变量）

Boolean（布尔变量）只有两种值：true（真）、false（假）。在某些情况下，ActionScript 脚本也把 true、false 转变成 1、0。Boolean（布尔变量）经常与逻辑运算符配合使用，通过 if 动作控制动画的播放。例如：

```
onClipEvent(enterFrame) {
    if ((userName == true) && (password == true)){
        play();
    }
}
```

只有当布尔变量 userName 和 password 都为 true（真）时动画才能继续播放。

reference（参数）中可以存储多个不同类型的数据，包括：

- Object（对象）

类似于 C++中 Class（类）的概念。ActionScript 脚本的对象是多种属性的结合，每一个属性都拥有自己的名称和值。这里的属性可以是简单变量也可以是一个对象。使用点（.）运算符可以指定想要的 Object（对象）或者属性。例如：

```
Figure.Circle.Radius
```

在 ActionScript 脚本中也定义了很多 Object（对象），使用它们可以简化或实现很多操作。例如：

```
value=Math.sqrt(value);   //求 value 的平方根
```

当然，用户也可以自己定义 Object（对象）。例如，可以把一个人的名字、性别、年龄、身高、体重

等多种类型的数据信息组合到一起定义为一个对象。同样，也可以把多个自定义对象组合，定义成为更高一级的对象。关于自定义对象的声明和使用将在 6.3.10 节中说明。

- Movie Clip（影片剪辑）

在上一章中，Movie Clip（影片剪辑）作为一个独立的 Flash MX 子动画，可以实现很多复杂的动画。其实，Movie Clip（影片剪辑）本身就是一种特殊的对象。用户可以通过调用或者设定 Movie Clip（影片剪辑）对象的各种属性控制影片的播放效果。例如：

```
Movie1.play();//播放影片剪辑 Movie1
Movie2.nextFrame();//播放影片剪辑 Movie2 的下一帧
myMouse.startDrag(true);//使影片剪辑随鼠标移动
```

Movie1、Movie2 和 myMouse 为影片剪辑的名字。影片剪辑的命名不能在 ActionScript 脚本语言中，而是应该使用 Properties（属性）面板。选中一个影片剪辑之后，激活 Properties（属性）面板，直接在图 6-12 所示的 Name（名称）文本框中填入电影剪辑的名字即可。

图 6-12　设置影片剪辑的名称

✍ 作为影片剪辑变量并不要求所定义的对象一定是影片剪辑，即使图形元件或者按钮元件同样也可以定义为影片剪辑变量。这时，需要在 Instance（实例）面板中把元件的 behavior 属性改为 Movie Clip（影片剪辑）。改变之后，元件将具有电影剪辑的特性，按钮元件将不再响应用户的输入。

6.3.3　变量的使用

对于创建交互式的 Flash MX 动画，变量是必不可少的。虽然变量的名称固定不变，但是它所存储的内容是可变的。在动画播放的时候，它可以记录用户的操作动作、动画播放的位置或者是否满足某种条件（true or false）。

ActionScript 脚本中的变量可以存储所有的数据类型（包括：字符串、数字、布尔变量、对象和影片剪辑）。典型的 ActionScript 脚本变量所存储的信息包括：用户名、数学计算的结果、某一事件发生的次数、某一按钮是否被选中等。

ActionScript 脚本变量的命名必须遵循如下原则。

- 变量名必须是有效的 identifier（标示符）。包括：字母、数字和下划线。
- 变量名不可以是 ActionScript 脚本的 Keyword（关键字）或者布尔值（true、false）。
- 在变量的作用范围内，变量名必须是唯一的。

变量的作用范围是指在一定的范围内变量是可用的。ActionScript 脚本变量按作用范围可以分为：全局变量和局部变量。全局变量在整个时间轴的任何一帧上均可用。局部变量只在它自己的代码区域中有效（两个大括号之间）。

在 ActionScript 脚本中，使用 var 可以定义局部变量。例如：在循环中，变量 i 和 j 经常用作计数器。在下面的代码中，i 作为一个局部变量，仅在 Myfuntion 函数中有效（关于函数的定义和使用将在 6.3.7

和 6.3.8 两节中讲述)。

```
function Myfuntion(){
    var i
    for( i = 0; i < monthArray[month]; i++ ) {
        _root.Days.attachMovie( "DayDisplay", i, i + 2000 );
        _root.Days[i].num = i + 1;
        _root.Days[i]._x = column * _root.Days[i]._width;
        _root.Days[i]._y = row * _root.Days[i]._height;
        column = column + 1;
        if (column == 7 ) {
            column = 0;
            row = row + 1;
        }
    }
}
```

使用局部变量可以有效地防止变量冲突。在函数中使用局部变量是一个好习惯，局部变量的作用范围仅仅在函数之中，这样可以使函数内部的变量与外部无关。如果函数中使用了全局变量，一旦这个变量在函数外部被改变，它将影响整个函数的工作。

✍ 在变量定义之后直接给变量赋初始值是一个好习惯。例如：var in=9；var out = sqr（in)。

定义全局变量应使用 SetVariables 动作或者直接使用赋值操作符（=)。例如：

```
SetVariables（x，0)；
y=0；
```

在 ActionScript 脚本中最容易出错的地方就是使用赋值操作符（=)，某个变量在没有使用 var 定义的时候使用赋值操作符（=）给变量赋值，其实，就是把该变量定义成为了全局变量。在初学 ActionScript 脚本的时候一定要注意这一点。

6.3.4 表达式与运算符

ActionScript 脚本的表达式就是由一个或者多个运算符连接成的算式。它在 ActionScript 脚本的动作中出现时，ActionScript 脚本先把表达式运算出来，再用运算的结果来执行动作。例如：

```
setProperty ("ball", _alpha, 20+alpha);
```

在编写 ActionScript 脚本的时候，如果使用动作面板的标准模式，而且要使用表达式，需要选中图 6-13 所示的表达式选项（Expression)。

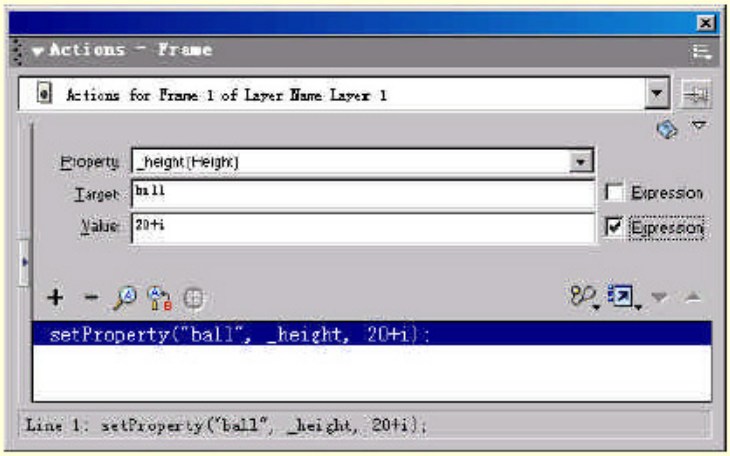


图 6-13 选中表达式选项

ActionScript 脚本的运算符主要分为：

- 算术运算符：加（+）、减（-）、乘（*）、除（/）、取余（%）、加 1（++）、减 1（--）
- 比较运算符：大于（>）、小于（<）、等于（==）、不等于（！=）、大于等于（>=）、小于等于（<=）

比较运算的结果是布尔值（true、false），它们经常被用在条件动作（if）中，例如：

```
if (x <= 100){
    loadMovie("bad.swf", 5);
}
 else {
    loadMovie("good.swf", 5);
}
```

- 字符串运算符

加号（+）或者 add 表示字符串的连接运算，例如：

```
"Congratulations, " + "Marry!"
```

运算的结果是“ Congratulations，Marry! ”。只要加号（+）的两端有一个为字符串，那么，Flash MX 将把其他所有的操作数转换成字符串。例如：

```
number=3;
string="There are "+number+"apples";
```

string 的值为“ There is 3 apple. ”。

比较运算符对字符串也同样有效，它将比较字符串在英文字母表中的先后顺序。

- 使用过 Flash 4 的用户在比较字符串的时候可能习惯于使用 eq、eg、lt 等字符串专用操作符，但是，Flash MX 的 ActionScript 脚本语言允许相同的操作符对不同类型的数据进行操作（有点类似于 C++语言中的重载的概念），所以，可以直接使用符号，而且使用符号运算更为直观。

- 逻辑预算符：非（！）、与（&&）、或（||）。

逻辑预算符可以进行布尔值之间的运算。常常利用它完成多种条件的判断。例如：

```
if ((i > 10) && (_framesloaded > 50)){
    play();
}
```

- 位运算符：位与（&）、位或（|）、位异或（^）、位反（~）、左移位（<<）、右移位（>>）、右移位添零（>>>）。

- 对于浮点数的位运算，ActionScript 脚本先把它们转换成 32 位二进制数，然后再进行位运算。

- 赋值运算符

符号（=）可以把指定的值赋予变量，例如：

```
name="anonymity";
```

另外，符号（=）可以像 C 语言一样，同时对多个变量赋值，例如：

```
a=b=c=d=name;
```

上式表示把变量 name 的值同时赋予变量 a、b、c、d。

在符号（=）之前加上其他运算符，可以构成复合运算符。如果在符号（=）前加一个加号（+）就组成了加等于符合运算（+=）。例如：

```
a+=3;          //等价于 a=a+3
x*=5       //等价于 x=x*5
```

类似的，ActionScript 脚本中的复合运算符有：+=、ー=、*=、/=、%=、<<=、>>=、>>>=、 &=、

^=、|=。

6.3.5 使用 Action 动作

Action（动作）可以控制 ActionScript 脚本动画的一切属性。在同一 Frame（帧）或者同一个物体上可以执行多个动作，例如：

```
swapDepths("mc1", "mc2");
gotoAndPlay(15);
```

表 6-2 列出了 Flash MX 的 ActionScript 脚本语言中所有的 Action（动作），其实，从它们的名称上也不难猜出它们的作用，而且有些动作同 C 语言的用法相同（例如：break、Include、For、Continue、Return、do ..while、while、if、if ..else、else if 等），请读者自行参考有关方面的书籍，这里不再介绍。在后面的章节中将针对部分主要动作做详细介绍。

表 6-2 Flash MX ActionScript 脚本语言中所有的动作

Break	getURL	prevFrame	toggleHighQuality
Call	gotoAndPlay	prevScene	trace
Call function	gotoAndStop	print	unloadMovie
clearInterval	if	printAsBitmap	unloadMovieNum
Comment	ifFrameLoaded	printAsBitmapNum	updataAfterEvent
Continue	include	removeMovieClip	var
Default	#initclip	return	with
Delete	loadMovie	set variable	while
do while	loadMovieNum	setInterval	
duplicateMovieClip	loadVariables	setProperty	
else	loadVariablesNum	startDrag	
else if	Method	stop	
#endinitclip	nextFrame	stopAllSounds	
Evaluate	nextScene	stopDrag	
for	on	swapDepth	
for..in	onClipEvent	switch	
fsCommand	play	tellTarget	

对于特定的影片剪辑使用动作或者装载影片，必须指明影片剪辑的名称或者地址，在 Flash MX 的 ActionScript 脚本中叫作 Target Path（目标路径）。下面的几个动作都需要指明 Target Path（目标路径）：

- loadMovie
- loadVariables
- unloadMovie
- setProperty
- startDrag
- duplicateMovieClip
- removeMovieClip
- print
- printAsBitmap

在 Flash MX 的 ActionScript 脚本中使用点（.）运算符来指明目标路径（Target Path）是很方便的。例如：loadMovie 动作表示从指定的互联网地址上装载一个 Flash 动画到影片剪辑中。它至少需要两个参

数：URL（统一资源定位符）、Location（目标）影片剪辑。其中，Location（目标）影片剪辑就是一个 Target Path（目标路径），使用方法如下：

```
loadMovie("http://www.mySite.com/myMovie.swf", _root.movie1);
```

此外，标识一个被装载的影片剪辑，也可以使用_levelX 方式。其中，X 是装载影片时所指定的数字。例如，一个影片被装载到了 level5，那么，它的实例名称就是_level5，利用它可以指定被装载影片的目标路径：

```
onClipEvent(load) {
    loadMovie("myMovie.swf", 5);
    level5._alpha=50;
    leve5._x=100;
    leve5._y=100;
}
```

6.3.6 控制播放语句

Stop、Play 动作可以控制影片的播放，停止播放或者继续播放。

GotoAndStop、GotoAndPlay 动作可以控制影片播放时跳转到指定的帧。如果使用 GotoAndStop，则影片跳转后将停止；如果使用 GotoAndPlay，则影片跳转后将沿 Timeline（时间轴）继续播放。

在默认情况下，Flash MX 动画启动之后将自动地循环播放。这里给出一个例子，它将在动画播放前提示开始播放，在播放结束后提示重播。

（1）随意创建一个 Flash MX 动画，或者使用以前创建过的 Flash MX 动画，把动画的第一帧按照图 6-14 的方式空出。

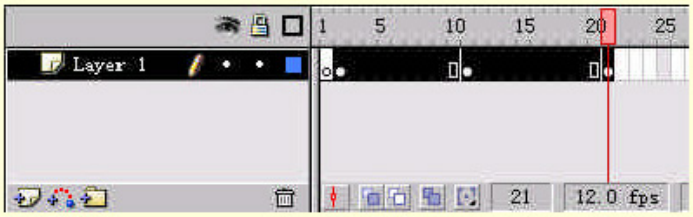


图 6-14 空出第一帧

（2）在第 1 帧的帧面板上加入 Stop 动作，阻止 Flash MX 动画启动默认的播放，请读者参考图 6-15。

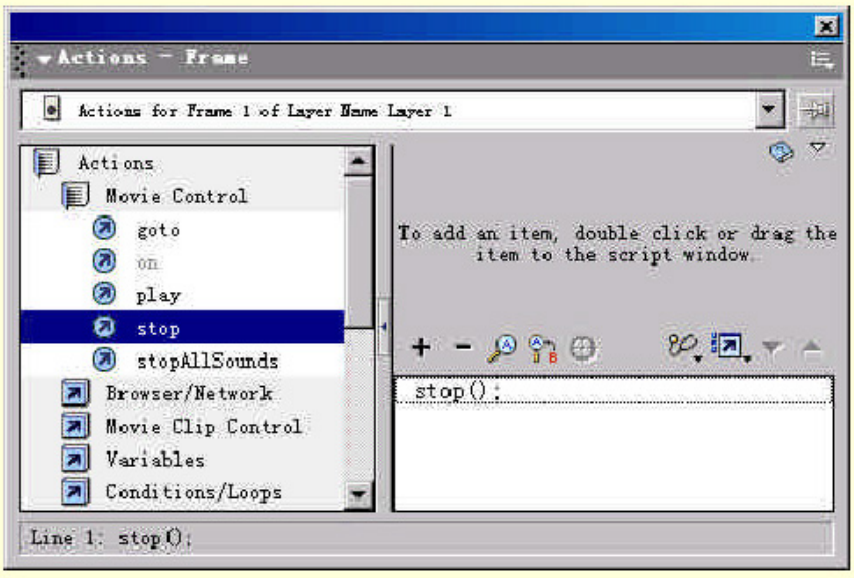


图 6-15 加入 Stop 动作

（3）在第 1 帧处加入一个开始按钮（可以直接使用通用库中的按钮）和动画片头的说明文字等信息，并在按钮上添加 Play 动作，如图 6-16 所示。

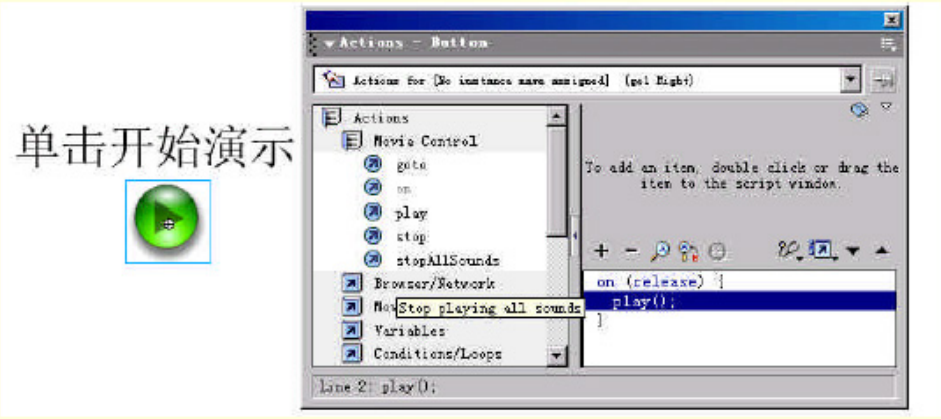


图 6-16　添加片头及按钮动作

- 按钮动作的响应不同于帧，它的格式为 on（Evens）{Actions}。Evens 是指与按钮相应的事件，Actions 为指定 Evens（事件）发生之后所执行的 ActionScript 脚本。关于按钮的使用方法将在下一章中介绍。

（4）在动画的最后一帧之后，插入空白帧，添加帧的动作 Stop，停止播放动画，并加入结束词和重播按钮，添加按钮动作，如图 6-17 所示。

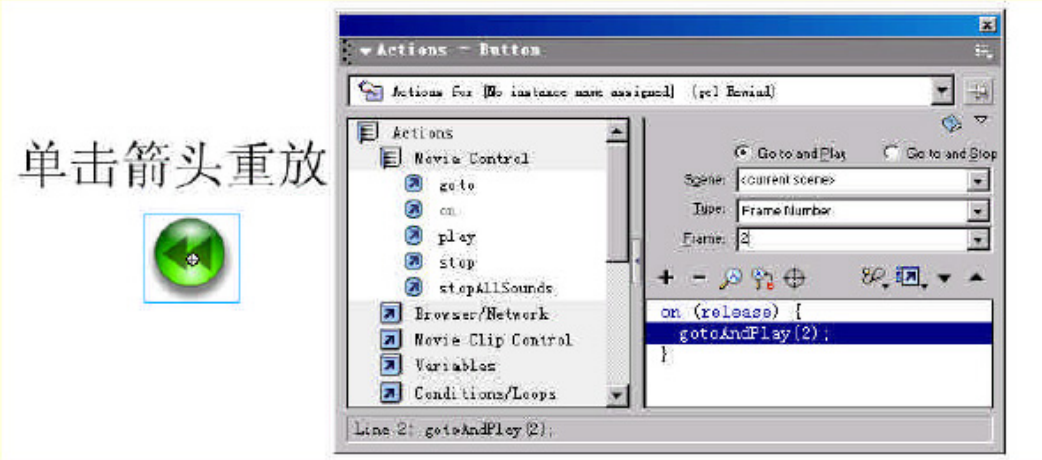


图 6-17　加入片尾和动作

- 在片尾的按钮动作中使用 gotoAndPlay（2）是指跳转到时间轴的第 2 帧处并继续播放，如果使用 gotoAndPlay（1）则将再次出现片头提示。

（5）输出并测试动画。这时，动画的播放完全受欣赏者的控制。

动作 gotoAndPlay 的用法如下：

```
gotoAndPlay(scene, frame);
```

- scene：表示跳转的场景，此参数可以省略，省略表示在本场景中跳转，其他用法如下：
  - Previous Scene：上一个场景
  - Next Scene：下一个场景
  - 场景名称：跳转到的指定的场景，此时，scene 用一个字符串表示。

- 新建的 Flash MX 文件只有一个场景（Scene1）。选择 Insert | Scene 菜单可以添加场景。Flash MX 中使用多场景就像是在一个多幕剧中换幕一样。每一个场景都有自己的时间轴，管理很

方便。在编辑的时候使用场景按钮可以在不同的场景之间按照图 6-18 的方式进行切换编辑。

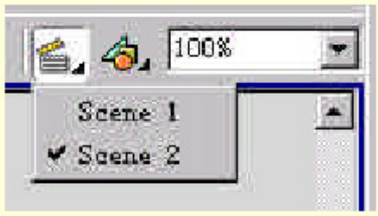
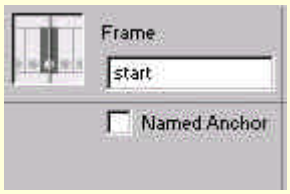


图 6-18 场景按钮

- frame：跳转到的帧。Frame 可以有几种用方法：
  - Previous Frame：上一帧。
  - Next Frame：下一帧。
  - Frame Number：跳转到的第 X 帧，此时，frame 用一个数字表示。
  - Frame Label：跳转到的指定标签的帧，此时，frame 用一个字符串表示。

选中一帧后，在 Properties（属性）面板中的 Frame Label（帧标签）文本框中可以按照图 6-19 所示方式指定帧的标签。



帧 标

图 6-19 帧的标签

gotoAndStop 的用法与 gotoAndPlay 完全相同，它们的区别在于，跳转之后是继续播放动画，还是停止播放。这里不再重复。

6.3.7 调用函数

函数是集成了的 ActionScript 脚本，它可以在动画播放的任何时候调用。有的函数调用要求传递相关的参数，函数处理这些参数，必要时函数还可以有返回值，如参数的运算结果或者函数的执行状况等。函数的参数和返回值可以是 ActionScript 脚本中的所有的数据类型。

Flash MX 中提供了很多预定义函数，用户可以直接调用。利用它们可以处理一定的信息或者执行相关的任务。在 Timeline（时间轴）的任何位置的函数都可以随时调用，甚至包括装载进来的影片中的函数。

当为函数传递参数时，如果传递了多余的参数，那么多余的部分将被 ActionScript 脚本忽略。如果传递的参数不够，剩余的参数将被赋予 undefined（未定义）类型的数据，这将在导出 Flash MX 动画的时候引起错误。表 6-3 列出了 Flash MX 的 ActionScript 脚本中所有的预定义函数。关于这些函数具体的使用方法请参见本书的附录部分。

表 6-3 ActionScript 脚本中的预定义函数

Array	GetTimer	Object	unescape
Boolean	GetVersion	ParseFloat	
Escape	isFinite	ParseInt	
Eval	isNaN	String	
GetProperty	Number	targetPath	

在编写 ActionScript 脚本的 Action（动作）面板的专家模式下，直接键入函数名称及其参数，就可

以调用函数。例如：

```
x=y=random (100);          //ActionScript 脚本预定义的随机函数，参数为 100
initialize();              //自定义的无参数函数
result=run();              //自定义的有返回值函数
```

在动作面板（Action）的标准模式下，如果调用预定义函数，可以在 function（函数）列表中直接选择；如果调用自定义函数，则应该使用 evaluate 动作新创建一行 ActionScript 脚本，并在 Expression（表达式）文本框中填入函数及其所需的参数，见图 6-20。

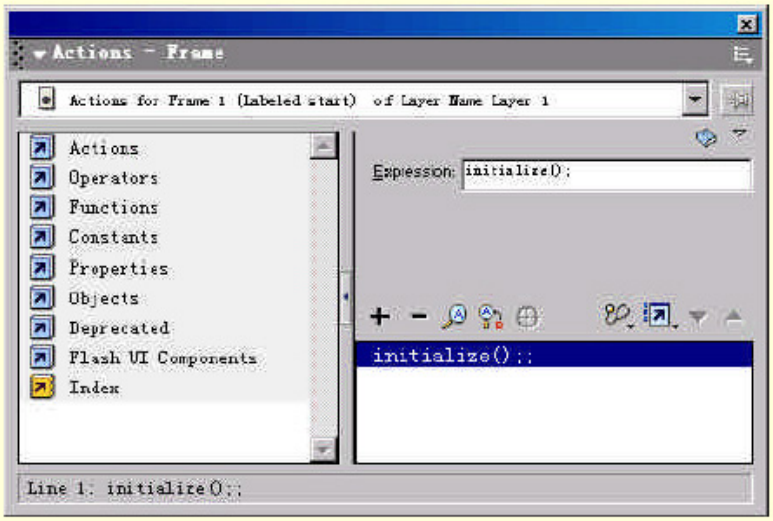


图 6-20　调用自定义函数

如果要调用其他 Timeline（时间轴）上或者其他对象内的函数，例如 Movie Clip（影片剪辑）中的函数，就要使用点（.）运算符指明目标路径，下面的代码调用了 MyfunctionsMovieClip 影片剪辑中的 initialize、calculateTax、run 函数。

```
MyfunctionsMovieClip.initialize();
MyfunctionsMovieClip.calculateTax(total);
Value=MyfunctionsMovieClip.result(50+random(50));
```

另外，也可以使用 Telltarget 动作或者 With 动作来指明目标路径。关于以上两个 Actions（动作）的具体使用方法将在 6.3.11 节中详细说明。

6.3.8　自定义函数

Flash MX 的 ActionScript 脚本语言允许用户自己定义函数完成特定的功能或者运算。与 ActionScript 脚本的预定义函数一样，自定义函数也可以传递参数或者返回值。自定义函数一旦被定义，它就可以在 Timeline（时间轴）的任何地方（包括装载的影片中）被调用。

> 一个优秀的函数要求对函数的输入输出参数有详细的注释，即使是自己写的函数，时间长了，也不可能记清楚每一个函数的每一个参数的作用。如果有了详细的注释，就可以不用关心函数的内部代码而直接使用函数。这一点对于编写大型交互式 Flash MX 动画的人来说尤为重要。

同 C++语言的注释一样，ActionScript 脚本语言中，双右斜杠后面的文字就表示注释。如果使用彩色方式显示 ActionScript 脚本，注释内容将变为灰白色。

使用 Function 动作可以定义一个函数，例如：

```
function average (x,y,z) {
    var value;
    value=(x+y+z)/2;
```

```
        return value;
    }
```

在 Function 后面的 avaerage 是自定义函数的名称（函数的命名要求同变量的相同）。紧跟在函数名称（average）之后的圆括号中的字符是自定义函数的参数，参数之间用逗号分隔。如果函数没有参数则使用空括号。自定义函数中的 return 动作给出函数的返回值（value）。脚本运行到 return 将直接返回，不再继续执行函数的其他部分。例如：

```
function getresult(score){
    if(score<60)
        return "very bad";
    if(score<80)
        return "bad";
    return "good";
}
```

函数 getresult 根据得分 score 来返回不同的结果，当 score 大于等于 60 小于 80 的时候将返回 bad，这里并没有使用 if(score<80 && score>=60)进行判断。因为如果 score 小于 60，脚本执行到 return "very bad"；就已经返回了。对于无返回值的函数，直接使用 return；就可以结束函数调用。例如：

```
function finish(){
    if(bfinish==false)
        return;
    …
}
```

6.3.9 使用对象

ActionScript 脚本中的对象类似于 C++语言中 Class（类）的概念，它可以拥有自己的变量（或属性），拥有自己的函数（或方法）。这就很大程度地简化了对某一类问题的处理。表 6-4 列出了 Flash MX 的 ActionScript 脚本中所有的预定义对象。

表 6-4 ActionScript 脚本中的预定义对象

Array	CustomActions	MovieClip	String
Accessibility	Date	Number	System
Boolean	Key	Object	TextField
Button	LoadVars	Selection	TextFormat
Capabilities	Math	Sound	XML
Color	Mouse	Stage	XMLSocket

每一个对象都有它们各自的特点和作用。有的对象的某些属性是只读的，用户不可以改变它们，例如 Key 对象中有很多属性对应着键盘按键的编码，它们都是不可改变的。本节只介绍几种 Flash MX 常用的对象，更详尽的对象使用说明，请参考附录。

使用 new 操作符可以创建一个对象。new 操作符先创建一个对象的拷贝，再为这个对象初始化属性和方法，这就好像是通过 ActionScript 脚本语言把一个影片剪辑从元件库中拖放到 Flash MX 的舞台上。例如：

```
currentDate = new Date();
myarray = new Array();
```

使用点（.）运算符可以访问对象的属性，如果对象的属性是可修改的，也可以用符号（=）为属性赋值。例如：

```
picture._x=200;
picture._y=100;
picture._alpha=50;
picture._name=mypic
angle=mypic._rotation;
```

通过点（.）运算符也可以调用对象的方法。例如：

```
s = new Sound(this);
s.setVolume(50);      //设置音量为 50%
```

- 对象的方法（Methods）实质上就是封装在对象中用于处理对象中的信息的函数，它类似于C++语言中的成员函数。

有一些 Flash MX ActionScript 脚本中的预定义对象中的方法也可以不创建对象而直接使用。例如 Math 对象中的数学函数：

```
angle = Math.random()*Math.PI;
result1 = Math.sin(angle);
angle = Math.random()*Math.PI;
result2 = Math.sin(angle);
result = Math.max(result1,result2);
```

6.3.10 使用自定义对象

Flash MX 的 ActionScript 脚本允许用户创建自己的对象并为对象添加自己的属性或方法。对象就相当于一个复杂信息的容器，它是多种数据类型的集合，每一个数据在对象中称作属性。对象的属性可以是 ActionScript 脚本的任何一种类型的数据。包括 String（字符串）、Number（数字）、Boolean（布尔值）、Object（对象）、Movie Clip（影片剪辑）。下面的对象中包含了各种不同类型的数据：

```
customer.name = "Jane Doe";
customer.age = 30;
customer.member = true;
customer.account.currentRecord = 000609;
customer.mcInstanceName._visible = true;
```

- 对象中的属性也可以是一个对象。例如在上述脚本中的第 4 行，account 是 customer 的属性，而 currentRecord 是 account 的属性，currentRecord 是数字类型的数据。可见，ActionScript 脚本这种对象中包含对象的逐层结构可以很方便地管理各种复杂的信息。

使用 new 操作符可以通过构造函数创建一个自定义对象。构造函数是与自定义对象同名的、用来初始化自定义对象的函数。例如，如果自定义对象的名称为 account，那么，它的构造函数的名称也应该是 account。下面的函数通过构造函数创建了一个新对象 MyConstructorFunction：

```
new MyConstructorFunction (argument1, argument2, ... argumentN);
```

一旦构造函数 MyConstructorFunction 被调用，ActionScript 脚本将传递一个隐藏的参数 this 给构造函数。this 本身就代表了构造函数。通过 this 可以为对象创建并初始化属性。下面的构造函数创建了一个圆，并初始化圆的各种属性：

```
function Circle(radius) {
    this.radius = radius;
    this.area = Math.PI * radius * radius;
}
```

如果要在 ActionScript 脚本中使用上述的自定义函数，可以使用 new 操作符直接创建 Circle 对象，下面的例子通过构造函数创建了一个半径为 5 的圆：

```
var myCircle = new Circle(5);
```

Flash MX 的 ActionScript 脚本在创建所有的自定义对象的同时会为它们自动建立一个 prototype 属性。使用对象 prototype 属性可以为对象本身添加方法。下面的例子中说明了如何创建一个对象并为它添加方法：

（1）创建一个自定义的圆对象的构造函数：

```
function Circle(x,y,radius) {
    this.radius = radius;
    this.x = x;
    this.y = y;
}
```

（2） 定义方法 area。此方法将用于计算圆对象的面积。

```
Circle.prototype.area = function () {
        return Math.PI * this.radius * this.radius;
};
```

✍ 此处使用的是一个赋值运算，所以在函数的大括号（{}）之后需要以分号（;）作为结尾。

（3）用自定义的圆对象定义一个变量。

```
var myCircle = new Circle(4);
```

（4）调用圆对象的 area 方法，得到圆的面积。

```
var myCircleArea = myCircle.area();
```

实际上，当调用 myCircle.area() 方法时，ActionScript 脚本先查找 Circle.area()方法是否存在，因为 Circle.area()方法不存在，所以脚本语言继续检查是否存在 Circle.prototype.area 方法，此时，自定义函数将被找到并执行。整个过程看起来很复杂，实际上就相当于直接定义并调用了 Circle.area()方法。

其实，也可以在对象的构造函数来定义对象的方法，下面的脚本也可以实现与上述脚本一样的功能：

```
function Circle(radius){
    this.radius = radius;
    this.area = function(){
        return Math.PI * this.radius * this.radius;
    }
}
```

这种方法虽然看上去很直观，但是并不提倡这样使用。因为构造函数将为变量分配内存空间，如果把方法的定义放在构造函数中，只要用此对象定义一个变量，ActionScript 脚本就会按照构造函数为它们分配内存空间，这样，如果对象定义了一个以上的变量，对象内部的方法将重复地被创建多次，而实际上，方法和属性不同，一个对象的所有变量可以共同使用同一个方法。使用 prototype 对象的创建方法效率会更高，因为 prototype 对象只为每一个变量复制一个方法的入口，真正的方法只有一个。

如果只是暂时地定义一个简单的对象，以后也不会或者很少反复地使用此对象，也可以使用 new Object()直接定义一个变量。例如：

```
customer = new Object();
customer.name = "Jane Doe";
customer.age = 30;
customer.telephone=123321;
customer.male=true;
customer.member = true;
```

上面的脚本首先定义了一个对象的变量，当使用赋值运算符（=）为对象变量的属性（包括：name、age、telephone、male、member）赋值的时候，如果 ActionScript 脚本发现其属性不存在，它将自动为此变量创建这个属性。其实，就相当于定义了一个具有 name、age、telephone、male、member 属性的变量。另外，也可以使用如下脚本创建对象变量，它的作用与上述脚本是相同的。

```
customer = {name:"Jane Doe", age:’ 30’ ,telephone:’ 123321’ ,male:true,member:true};
```

在大括号（{}）中的部分是对象的属性，属性之间用逗号（,）隔开，冒号前的部分表示属性名称，冒号后的部分表示属性的值。

大括号（{}）实质上定义的是一个对象，并把对象赋值给变量，如果按顺序执行下面的脚本：

```
customer = {name:"Jane Doe", age:’ 30’ ,telephone:’ 123321’ ,male:true,member:true};
customer = {name:"Tom White", age:’ 25’ ,member:true};
```

执行的结果将使 customer 只具有 3 个属性（而不是 5 个），这是因为，customer 变量被先后赋予了两次对象变量，而最后一次的对象只有 3 个属性，所以 customer 只具有 3 个属性。

6.3.11 指明路径

在 Flash MX 的 ActionScript 脚本中，通过点（.）运算符可以指明路径，利用 tellTarget 或者 with 动作，也可以指明目标路径。例如，下面 3 段 ActionScript 脚本的所起的作用是相同的。

```
//脚本 1：
MyMovieClip._x=100;
MyMovieClip._y=200;
MyMovieClip.play();
//脚本 2：
with(MyMovieClip){
    _x=100;
    _y=200;
    play();
}
//脚本 3：
tellTarget("MyMovieClip"){

    _x=100;
    _y=200;
    play();
}
```

可见，with 和 tellTarget 可以临时改变目标路径，当动作执行完后，在动作之外的脚本的作用对象又回到执行 with 或者 tellTarget 之前的那一个对象。

Flash MX 中推荐使用的是 with 动作，而将 tellTarget 动作归为 Deprecated（不赞成）一类。因为 with 是一个标准的 ActionScript 脚本在 ECMA-262 标准上的扩展。With 和 tellTarget 之间的主要区别在于，with 的参数是目标对象或者实体，而 tellTarget 的参数是目标的路径，所以，tellTarget 不能作用于对象。在本节中主要介绍 with 的使用。

当 with 所包含的脚本要搜索一个特定的目标时，ActionScript 脚本会根据指定的对象按照一个特定的顺序一层层地搜索，搜索的顺序称作搜索链，其顺序如下。

- 在 with 动作内指定的对象
- 在 with 动作外指定的对象
- 正被激活的对象

- 包含现在正在执行的脚本的影片剪辑
- 预定义对象

With 动作在反复使用多个对象的方法时非常有用。例如：

```
function polar(r){
    var a, x, y
    with (Math) {
        a = PI * r * r
        x = r * cos(PI)
        y = r * sin(PI/2)
    }
trace("area = " +a)
trace("x = " + x)
trace("y = " + y)
}
```

在上面的例子中，Math 对象被排在搜索链的最顶部，所以 cos、sin 和 PI 这些属性都在 Math 对象中找到了，with 动作把它们分别对应于 Math.cos、Math.sin 和 PI。而 a、x、y 和 r 并不是 Math 对象的方法或属性，with 就沿着搜索链找下去，直到在 polar 函数中找到它们。

同时，with 也支持嵌套使用。例如：

```
with (China){
    with (Beijing){
        _alpha = 80;
    }
    with (Tianjin){
        _alpha = 60;
    }
}
```

在上面的例子中，影片剪辑 Tianjin 和 Beijing 都是 China 的子剪辑。通过 with 动作改变了 Beijing 和 Tianjin 的 alpha（透明度），但没有改变 China 的透明度。它等同于如下脚本：

```
China.Beijing._alpha=80;
China.Tianjin._alpha=60;
```

6.3.12 数组对象的使用

Flash MX 的 ActionScript 脚本把数组定义成了一个对象，数组对象的属性是由数组后面中括号（[ ]）中的数字标识的，这些数字称作数组的下标。Flash MX 数组的下标都是从 0 开始，即[0]是数组中的第 1 个元素，[1]是第 2 个元素……[n] 是第 n+1 个元素。

使用运算符 new Array，可以创建一个数组，它包括 3 种格式。

- 创建一个长度为 0 的数组

```
myArray = new Array();
```

- 创建一个长度为 5 的数组

```
myArray = new Array(5);
```

- 创建一个长度为 5 的数组，并为数组赋值

```
myArray = new Array("星期一","星期二","星期三","星期四","星期五");
```

数组中的每一个元素所对应的值为：myArray[0] =“星期一”，myArray[1] = “星期二”，myArray[2] = “星期三”，myArray[3] = “星期四”，myArray[4] = “星期五”。

在 ActionScript 脚本中，中括号（[ ]）运算符表示对象的属性，所以，myArray[n]就相当于数组对象的一个属性，可以用赋值(=)运算符直接为它赋值。

数组中元素的数目（数组的长度）可以通过数组的属性 length 得到，例如：

```
myArray = new Array(5);
count=myArray.length;
```

✍ length 是数组对象的一个属性而不是方法，所以在使用时后面不应该加括号。

表 6-5 列出了 ActionScript 脚本中数组对象的所有方法及功能。

表 6-5　　数组对象的方法及功能

方　法	语　法	功　能
Concat	Array1.concat(array2);	合并多个数组
Join	Array.join(separator);	把数组中的元素合并成一个字符串
Pop	Array.pop();	使数组中的元素出栈（FILO 先入后出）
Push	Array.push(value1,value2,..valueN);	使元素入栈数组
Reverse	Array.reverse();	使数组反序
Shift	Array.shift();	使数组中的元素出队列（FIFO 先入先出）
Slice	Array.slice(start,end);	截取数组中的子串，生成新数组
Sort	Array.sort(compareFunction)	按照 compareFunction 给数组中的元素排序
SortOn	Array.sortOn(fieldName)	按照 fieldName 给数组中的元素排序
Splice	Array.splice(start,delCount,value0,..valueN);	添加或者删除范围内的元素
toString	Array.toString()	把数组中的元素合并成一个字符串，使用逗号（,）分隔
Unshift	Array.unshift(value1,value2,..valueN);	从数组头部插入一个或者多个元素

下面对数组对象的主要方法进行介绍。

- Join：把数组中的元素合并成一个字符串，参数 separator 为分隔字符串。例如：

```
myArray = new Array("星期一","星期二","星期三","星期四","星期五");
str=myArray.join(",");
```

str 的结果为：“ 星期一,星期二,星期三,星期四,星期五 ”。

- toString：把数组中的元素合并成一个字符串，使用逗号（,）分隔，它相当于 join(“,”)。
- Pop：使数组中的元素出栈（FILO 先入后出），返回数组的最后一个元素值，并将此元素从数组中删除。例如：

```
myArray = new Array("星期一", "星期二", "星期三", "星期四", "星期五");
today=myArray.pop();
count=myArray.length;
```

执行结果为：today=“ 星期五 ”，count=4。

- Push：使元素入栈数组，参数为入栈数组的元素，元素增加在数组的末尾。例如：

```
myArray = new Array("星期一","星期二","星期三","星期四","星期五");
myArray.push("星期六","星期日");
count=myArray.length;
```

执行结果为：count=7，myArray[0] =“ 星期一 ”，myArray[1] = “ 星期二 ”，myArray[2] = “ 星期三 ”，

myArray[3] = “星期四”，myArray[4] = “星期五”， myArray[5] = “星期六”， myArray[6] = “星期日”。

- Sort：按照 compareFunction 给数组中的元素排序，compareFunction(a,b)为数组排序的时候自动调用的一个有返回值的函数，当元素 a 先于元素 b 时，返回-1；当两个元素相等时，返回 0；当元素 b 先于元素 a 时，返回 1。例如：

```
var cf = function (a, b) {
        if (a < b) {
            return 1;
        }
        if (a > b) {
            return -1;
        }
    return 0;
    }
myArray = new Array("1","3","4","2","0");
myArray.Sort(cf);
result=myArray.toString();
```

执行之后的结果为：result=“4,3,2,1,0”。

另外，如果数组中的元素为可比量（如数字、字符串等），Sort 中的参数也可以省略，ActionScript 脚本将自动按照升序排列。例如：

```
myArray = new Array("1","3","4","2","0");
myArray.Sort(cf);
result=myArray.toString();
```

执行之后的结果为：result=“0,1,2,3,4”。

- Slice：截取数组中的子串，生成新数组，参数表示所要截取数组的起始下标和终止下标。返回值为新的数组。例如：

```
myArray = new Array("星期一","星期二","星期三","星期四","星期五");
subArray=myArray.slice(2,4);
result=subArray.toString();
```

执行结果为：result=“星期三,星期四”。注意，数组 subArray 不包括“星期五”，也就是说，slice(startend)方法所截取的子数组是从原数组的 start 下标开始，到 end-1 下标结束。

ActionScript 脚本的数组对象是动态可变的，定义了一个数组之后，用户可以直接为它的任何下标赋值，而不需要考虑数组的长度，如果下标超出了范围，ActionScript 脚本会自动增加数组的长度。例如：

```
myArray = new Array("星期一","星期二","星期三","星期四","星期五");
myArray[6]= "星期日";
result=myArray.toString();
count=myArray.length;
```

执行之后的结果为：result=“星期一,星期二,星期三,星期四,星期五,星期日”，count=7。

在上述结果中星期五和星期日之间有两个逗号（,），这是因为，数组起始长度为 5，下标范围 0~4，当为数组中下标为 6 的属性赋值时，ActionScript 脚本自动填充两个元素，将数组的长度增加到 7，并将 myArray[6]赋值。而 myArray[5]还没有被赋值，所以它为空值。

利用 ActionScript 脚本数组中的元素可以是对象的特性，只要将数组中的元素也定义成数组，可以很容易地创建出二维甚至多维数组。例如：

```
myArray = new Array(new Array，new Array);
```

```
for (i=0; i<2; i++) {
    for (j=0; j<3; j++) {
    myArray[i][j]=i+","+j;
    }
}
```

上面的脚本创建了一个二维数组，并为其赋值，其结果如下：

```
myArray[0][0]="0,0";    myArray[0][1]="0,1";    myArray[0][2]="0,2";
myArray[1][0]="1,0";    myArray[1][1]="1,1";    myArray[1][2]="1,2";
```

如果将上述脚本改为：

```
myArray = new Array(new Array，new Array);
for (i=0; i<3; i++) {
    for (j=0; j<3; j++) {
    myArray[i][j]=i+","+j;
    }
}
```

它的执行结果仍未改变，即 ActionScript 脚本并未自动增加二维数组的长度。这是因为，在一维数组中当使用符号（=）为其赋值时，ActionScript 脚本会自动检测符号（=）前数组的下标是否越界；同样，上述的在二维数组中，只对 myArray[i][j]进行赋值，ActionScript 脚本会自动检测数组 myArray[i]的下标是否越界，而没有检测 myArray 的下标。所以，数组 myArray 不会自动增加元素。如果改为如下脚本：

```
myArray = new Array();
for (i=0; i<3; i++)
{
    myArray[i]=new Array();
    for (j=0; j<3; j++) {
    myArray[i][j]=i+","+j;
    }
}
```

则数组会动态地增加，其执行结果为：

```
myArray[0][0]="0,0";    myArray[0][1]="0,1";    myArray[0][2]="0,2";
myArray[1][0]="1,0";    myArray[1][1]="1,1";    myArray[1][2]="1,2";
myArray[2][0]="2,0";    myArray[2][1]="2,1";    myArray[2][2]="2,2";
```

其实，使用 push 方法也可以增加二维数组的长度，下面的脚本与上面的脚本是等价的：

```
myArray = new Array();
for (i=0; i<3; i++) {
    myArray.push(new array());
    for (j=0; j<10; j++) {
        myArray[i][j] = i+","+j;
    }
}
```

使用类似的方法以可以创建出多维数组，这里不再讲述。

此外，中括号（[ ]）运算符也可以代替 new Array()定义或者初始化一个数组。例如：

```
myArray = [];
```

```
myArray = [5];
myArray = ["星期一","星期二","星期三","星期四","星期五"];
myArray[i] = [];
myArray.push([]);
```

上述脚本分别等价于：

```
myArray = new Array();
myArray = new Array(5);
myArray = new Array("星期一","星期二","星期三","星期四","星期五");
myArray[i] = new Array();
myArray.push(new Array());
```

6.3.13 跟踪调试

使用 ActionScript 脚本中的 trace 动作可以在脚本运行的时候，在 Output（输出）窗口中显示信息。其语法为：

```
trace(expression);
```

参数 expression 是一个表达式，表达式的结果将显示在如图 6-21 所示的 Output（输出）窗口中。具体地说，表达式可以是数字、字符串、布尔值或者是它们之间所组成的运算。例如：

```
trace("This is a test.");
x=5;
y=10;
trace(x);
trace(y);
trace(x+","+y)
```

上述脚本的输出结果如图 6-22 所示。

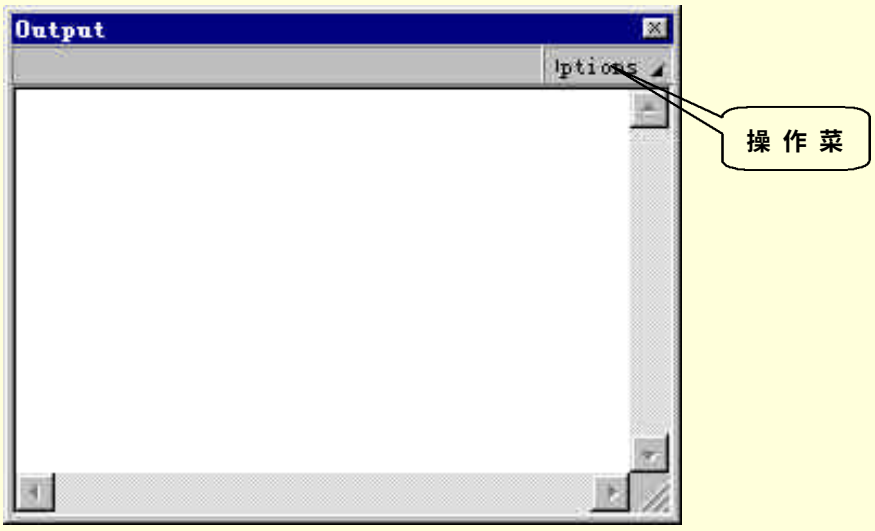


图 6-21 输出窗口

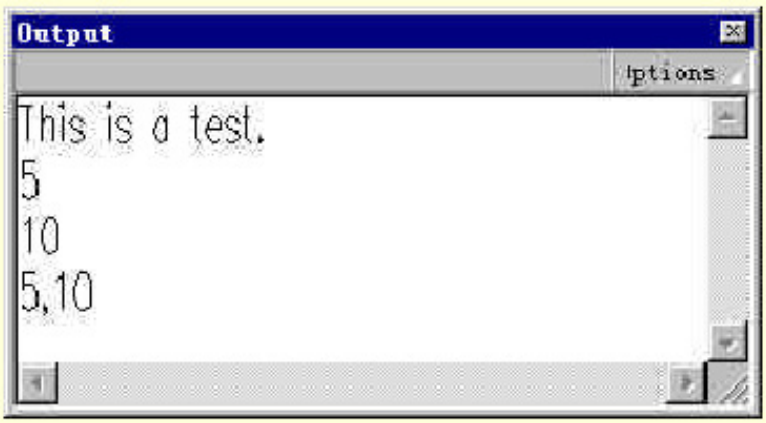


图 6-22　跟踪结果

如果 expression（表达式）是一个 Movie Clip（影片剪辑）或者一个 Object（对象），例如脚本：

```
value=new object();
trace (value);
trace (plane);          //plane 为一个影片剪辑名称
```

在输出窗口中将输出如图 6-23 所示的内容。

图 6-23　输出结果

可见，对于影片剪辑 trace 的动作将输出此影片剪辑所在的层次。在调试复杂动画，尤其是动态创建影片剪辑的动画时，这是一种很有用的功能。

在输出窗口上右击鼠标或者单击窗口右上角的 Option（操作）按钮，将会弹出如图 6-24 所示的快捷菜单。

图 6-24　输出窗口快捷菜单

其功能如下：

- Copy（复制）：复制输出窗口中选中的内容到剪贴板。
- Clear（清除）：清除输出窗口中选中的所有内容。
- Find（查找）：查找输出窗口中所需的文本。
- Find Again（再次查找）：查找输出窗口中所需文本的下一个位置。
- Save to File（保存到文件）：保存输出窗口中的内容到文本文件。
- Print（打印）：打印输出窗口中的内容。

6.3.14 影片导航面板的使用

使用 window | Movie Explorer 菜单项，可以打开如图 6-25 所示的 Movie Explor（影片导航）面板。影片导航面板包含了整个 Flash MX 动画中所有的成分。利用影片导航面板可以清晰地看到整个动画的层次，从而迅速地找到想要编辑的部分。其各部分的功能介绍如下：

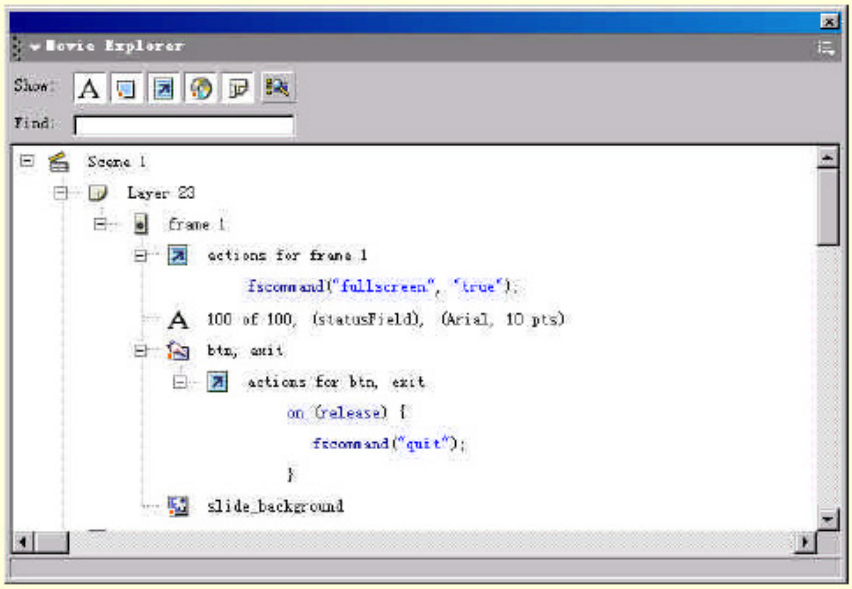


图 6-25 影片导航面板

- 显示控制：如图 6-26 所示的显示控制工具栏可以控制在影片导航面板中所显示的内容。其主要功能包括以下这些。

图 6-26 显示控制工具栏

- 显示文本：显示或隐藏整个 Flash MX 动画中的文本区域。
- 显示元件：显示或隐藏动画中的 Graphic（图形）、Button（按钮）、Movie Clip（影片剪辑）等元件。
- 显示脚本：显示或隐藏动画中的 ActionScript 脚本。
- 显示帧和层：显示或隐藏动画中每一条时间轴上的所有关键帧和层。
- 显示导入内容：显示或隐藏导入到动画中的声音、图片、电影等外部文件。
- 制定显示：选择此项后将弹出如图 6-27 所示的对话框，以便更精确地制定影片导航面板中所需要显示的内容。

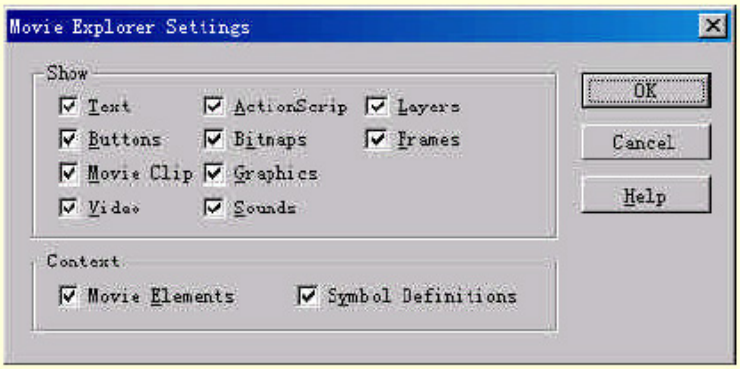


图 6-27　精确的制定影片导航面板

在上述对话框中选中的复选框为显示的内容。表 6-6 中列出了所有复选框的具体作用。

表 6-6　　制定影片导航面板

复　选　框	作　　用
Text	显示或隐藏动画中的文本区域
Button	显示或隐藏动画中所使用的按钮
Movie Clip	显示或隐藏动画中所使用的影片剪辑
Video	显示或隐藏动画中所导入的视频剪辑
ActionScript	显示或隐藏动画中的 ActionScript 脚本
Bitmaps	显示或隐藏动画中所导入的位图文件
Graphics	显示或隐藏动画中所使用的图形元件
Sounds	显示或隐藏动画中所导入的声音
Layers	显示或隐藏动画中每一条时间轴上的所有关键帧
Frames	显示或隐藏动画中所有的层
Movie Elements	显示或隐藏动画中的影片（主要指场景）
Symbol Definitons	显示或隐藏动画中的元件定义部分

- 查找控制：如图 6-28 所示，在查找控制的文本区域中输入所要寻找的字符，在影片导航面板中将显示出与其相关的部件。

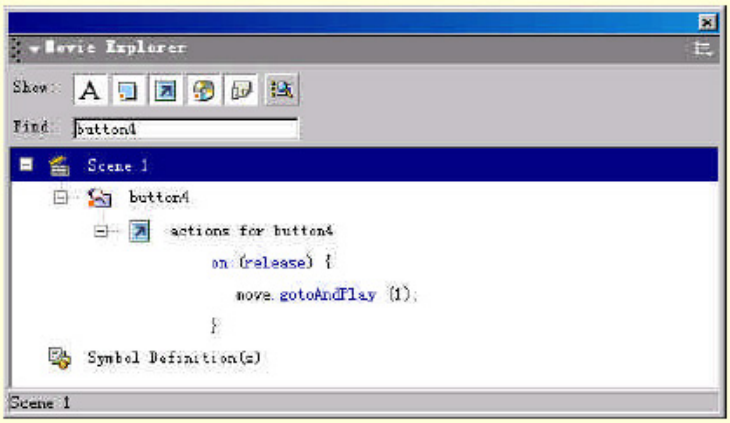


图 6-28　查找的结果

- 快捷菜单：单击快捷菜单按钮或者在 Movie Explorer（影片导航）面板中所列出的元件上右击鼠标，将会弹出一个快捷菜单。在菜单中可以完成导航面板的大多数功能。其说明见表 6-7。

表 6-7 影片导航面板快捷菜单的功能

菜 单 项	功 能 说 明
Goto Location	跳转到选中对象所在的位置进行编辑
Goto Symbol Definition	在影片导航面板中选中元件的定义部分
Select Symbol Instance	选中动画中所使用的元件
Find in Library	在元件库中选中的对象
Rename	重命名对象
Edit in Place	直接在元件所处的当前位置进行编辑，相当于双击元件
Edit in New Windows	打开一个新的窗口，编辑选中的元件
Show Movie Elements	显示或隐藏动画中的影片（主要指场景）
Show Symbol Definition	显示或隐藏动画中的元件定义部分
Show all Scenes	显示或隐藏动画中的所有场景
Copy All Text to Clipboard	复制所选中的文本到剪贴板
Cut	剪切所选中的对象
Copy	复制所选中的对象
Paste	粘贴所选中的对象
Clear	清除所选中的对象
Expand Branch	展开一层分支
Collaspe Branch	折叠一层分支
Collaspe Others	折叠其他的分支
Print	打印

✍ 直接在 Movie Explorer（影片导航）面板中双击鼠标，也可以编辑所选中的对象。

可见 Movie Explorer（影片导航）面板为用户提供了一种更快捷的方法来显示或者组织 Flash MX 动画中的所有内容。使用它可以方便地查找元件，清楚地显示整个 Flash MX 动画的层次结构。影片导航面板强大的查找定位功能在创建复杂动画的时候显得尤为重要。

第 7 章　创作交互式 Flash 动画

7.1　综　述

一个交互式的动画完全不同于普通顺序式的动画。通过键盘和鼠标用户可以主动地控制动画，比如，跳转到动画的各个部分、移动动画中的物体、向动画中输入信息并得到响应、单击按钮或者其他更复杂的交互。

在 Flash MX 中，所有的交互行为都是依靠上一章中所讲述的 ActionScript 脚本语言来实现的。实现交互具体的过程是，当用户或者 Flash MX 动画本身触发了某一 Event（事件）时，比如单击按钮、按键盘上的按键、动画播放到某一帧、影片被装载或者卸载等，事件就会执行相应的 ActionScript 脚本，从而完成交互。

本章中所创建的交互动画是比较复杂的交互，只要掌握了这些技巧，便可以创作出高级的交互式 Flash MX 动画。

7.2　响应事件

7.2.1　关于事件

当动画播放到某一帧时，在帧上的 ActionScript 脚本会自动运行。Button（按钮）或者 Movie Clip（影片剪辑）将针对用户的操作触发相应的 Event（事件），从而执行相应的 ActionScript 脚本。这里的 Button（按钮）或者 Movie Clip（影片剪辑）都是针对元件的 Behavior 属性而言的。元件的 Behavior 属性并不要求和元件本身的类型相同，它是可以改变的。例如在舞台上放置一个 Movie Clip（影片剪辑），在 Properties 面板中它默认的 Behavior 属性是 Movie Clip（影片剪辑）。如图 7-1 所示。

图 7-1　影片剪辑的作用

通过图 7-2 所示的下拉列表框可以将上述 Movie Clip（影片剪辑）的 Behavior 属性改为 Button（按钮）或者 Graphic（图形）。

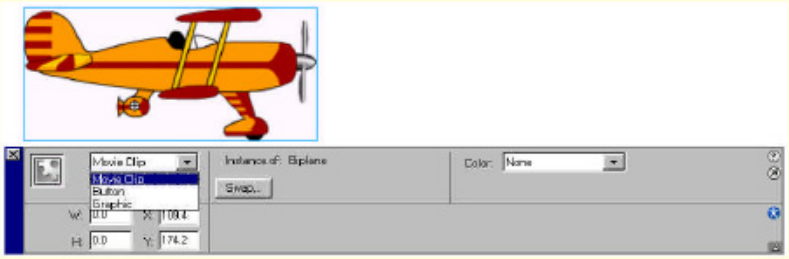


图 7-2 改变影片剪辑的作用

7.2.2 按钮事件

Flash MX 的按钮可以触发多种事件，下面的方法可以为按钮所触发的事件添加 ActionScript 脚本。拖放一个按钮到舞台上，选中此按钮，在标准模式的 Action（动作）面板中随意添加一个动作，Flash MX 就会自动添加 Release 按钮事件来执行所添加的动作，如图 7-3 所示。

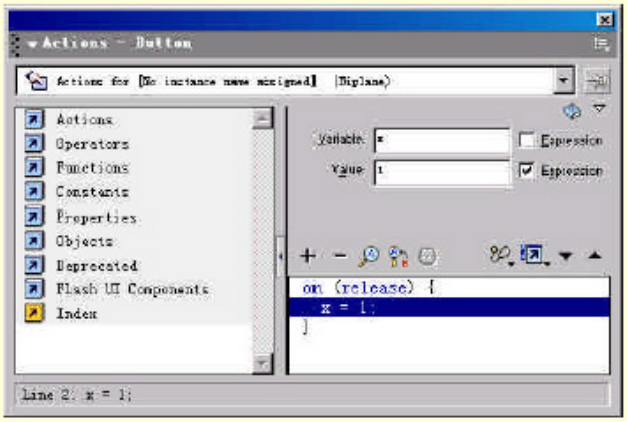


图 7-3 自动添加按钮事件

选中第一行 on(release)，标准模式的动作面板便会列出所有的按钮事件供选择，如图 7-4 所示。

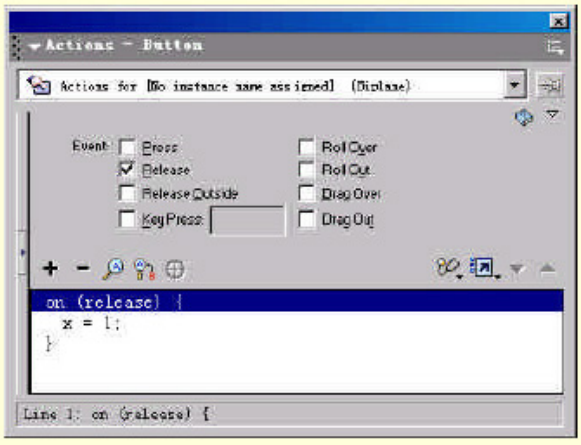


图 7-4 按钮事件

每一个事件的具体用法如下：

- Press：当鼠标移到按钮上并按鼠标按键时，触发此事件。
- Release：当鼠标移到按钮上单击后释放鼠标按键时，触发此事件（这是默认鼠标事件）。
- Release Outside：当按下按钮，而在按钮外面释放鼠标时发生此事件。
- Key Press：当指定键盘上的按键被按下时，发生此事件
- Roll Over：当鼠标放置在按钮上时发生此事件。
- Roll Out：当鼠标从按键上移出时发生此事件。

- Drag Over：当鼠标放置在按钮上的同时按住鼠标按键，然后将鼠标从按钮上拖出（依然按住鼠标按键），最后再将鼠标放回按钮时发生此事件。
- Drag Out ：当鼠标放置在按钮后，按住鼠标按键，然后将鼠标从按钮上拖出（依然按住鼠标按键）时动作发生。

下面的例子可以测试按钮的各种动作：

（1）新建 Flash MX 文件，从共享库中拖放一个如图 7-5 所示的按钮到舞台上。

图 7-5　在舞台上放一个按钮

（2）在按钮的下面创建一个文本区域，以便显示鼠标的动作。文本区域属性的设置如图 7-6 所示。

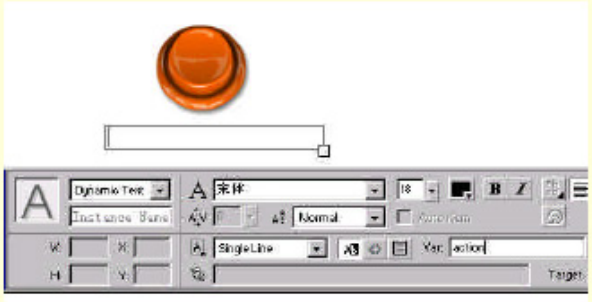


图 7-6　设置文本区域属性

（3）为按钮添加如下动作：

```
on (press) {
    action = "Press";
}
on (release) {
    action = "Release";
}
on (releaseOutside) {
    action = "Release Outside";
}
on (rollOver) {
    action = "Roll Over";
}
on (rollOut) {
    action = "Roll Out";
}
on (dragOver) {
    action = "Drag Over";
}
on (dragOut) {
    action = "Drag Out";
}
```

✎ 在 Flash MX 中，按钮事件也可以被称作鼠标事件，如果使用标准模式的动作面板，在前一个按钮动作的结尾，直接从动作列表框中选择 On 就可以按照图 7-7 的方式添加按钮的其他动作了。

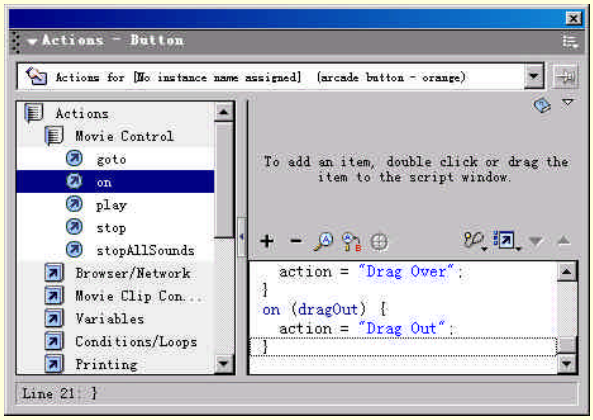


图 7-7 添加按钮事件

（4）输出并测试动画，当用户执行每一个按钮动作时便会显示出如图 7-8 所示的文字提示。

图 7-8 显示按钮动作

7.2.3 影片剪辑事件

与按钮类似，Flash MX 的 Movie Clip（影片剪辑）也可以触发多种事件。下面将为影片剪辑所触发的事件添加 ActionScript 脚本。拖放一个影片剪辑到舞台上，选中它，在标准模式的 Action（动作）面板中随意添加一个动作，Flash MX 就会自动添加 load 影片剪辑事件来执行所添加的动作。选中第一行 onClipEvent(Load)，标准模式的动作面板便会按照图 7-9 的方式列出所有的影片剪辑事件。

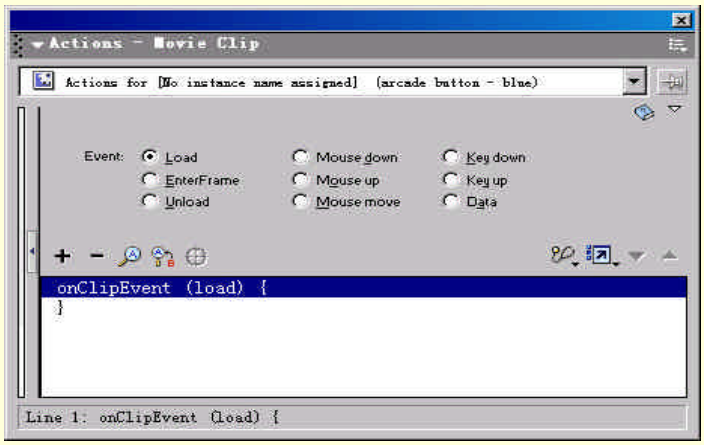


图 7-9 影片剪辑事件

✎ 影片剪辑事件与按钮事件的不同之处在于，按钮允许多个按钮事件共用同一个脚本，例如：On（release,rollOver,keyPress）{..}，而影片剪辑事件则不能共用脚本。利用按钮事件可以

共用脚本的特性，可以为完成同一任务制定多种方式或者快捷键，而对于影片剪辑则完全没有必要。

影片剪辑的各个事件说明如下：

- Load：当影片剪辑被装载进来的时候，发生此事件。
- EnterFrame：当影片剪辑播放每一帧的时候，发生此事件。
- Unload：当影片剪辑被卸载的时候，发生此事件。
- Mouse down：当鼠标左键被按下的时候，发生此事件。
- Mouse up：当鼠标左键被松开的时候，发生此事件。
- Mouse move：当鼠标移动的时候，发生此事件。
- Key down：当键盘按键被按下的时候，发生此事件。
- Key up：当键盘按键被松开的时候，发生此事件。
- Data：使用 loadVariables 或者 loadMovie 动作，当接收到数据的时候，发生此事件。

下面的例子说明了影片剪辑的各个事件的用法：

（1）新建 Flash MX 文件，拖放一个影片剪辑到舞台上，如图 7-10 所示。

图 7-10　拖放影片剪辑到舞台上

（2）在影片剪辑的下面创建两个文本区域，命名为 Action1、Action2，以便分别显示影片剪辑的动作。文本区域属性的设置如图 7-11 所示。

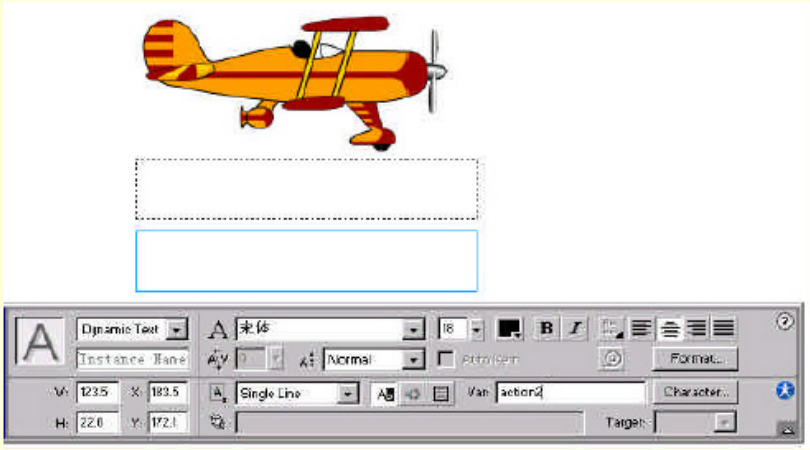


图 7-11　创建并设置文本区域

（3）为影片剪辑添加如下脚本：

```
onClipEvent (load) {
    _root.action1 = "Load";
}
onClipEvent (enterFrame) {
    _root.action1 = "EnterFrame:"+this._currentframe;
}
onClipEvent (mouseDown) {
    _root.action2 = "Mouse Down";
}
```

```
onClipEvent (mouseUp) {
    _root.action2 = "Mouse Up";
}
onClipEvent (mouseMove) {
    _root.action2 = "MouseMove";
}
onClipEvent (keyDown) {
    _root.action2 = "Key Down";
}
onClipEvent (keyUp) {
    _root.action2 = "Key Up";
}
```

✍ 因为影片剪辑不同于按钮，它将改变目标路径，也就是说，在 onClipEvent () {……}中的脚本所指的对象是影片剪辑本身，所以，上述脚本使用_root.action1="……"来显示文本。_root 所指的是主时间轴。

（4）输出并测试动画，当执行到每一个影片剪辑动作时，影片剪辑下面便会显示出相应的文字提示，如图 7-12 所示。

图 7-12 影片剪辑的动作

7.3 获得鼠标的位置

使用_xmouse 和_ymouse 可以得到鼠标在动画中的坐标。每一个影片剪辑或者动画都有_xmouse、_ymouse 属性，它们表示的是鼠标相对于当前对象的坐标。

下面的例子将显示不同参考系下的鼠标坐标：

（1）新建一个 Flash MX 文件，创建一个 Movie Clip（影片剪辑）。在影片剪辑的第 1 帧中绘制一个填充色为灰色的矩形。

（2）为影片剪辑添加一个新层，在新层上添加一个文本区域，用来显示鼠标在影片剪辑中的坐标，并设置文本区域的属性如图 7-13 所示。

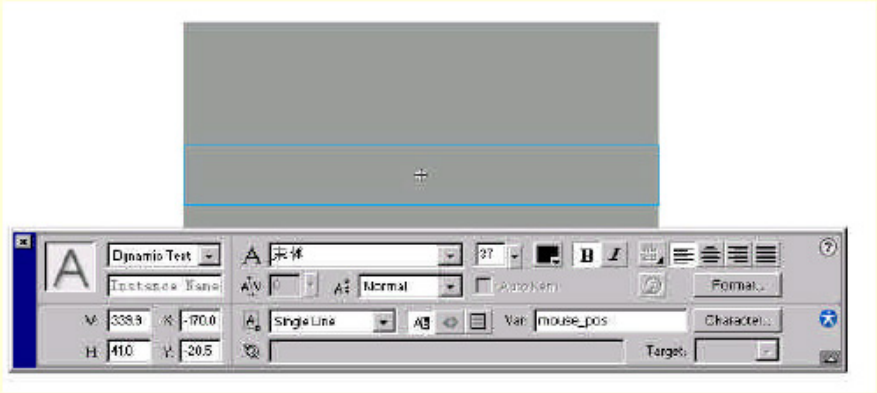


图 7-13　设置影片剪辑文本区域属性

（3）回到舞台上，将影片剪辑拖放到主场景中央。在主场景中也创建一个文本区域，以显示鼠标在主场景中的坐标，并设置文本区域的属性如图 7-14 所示。

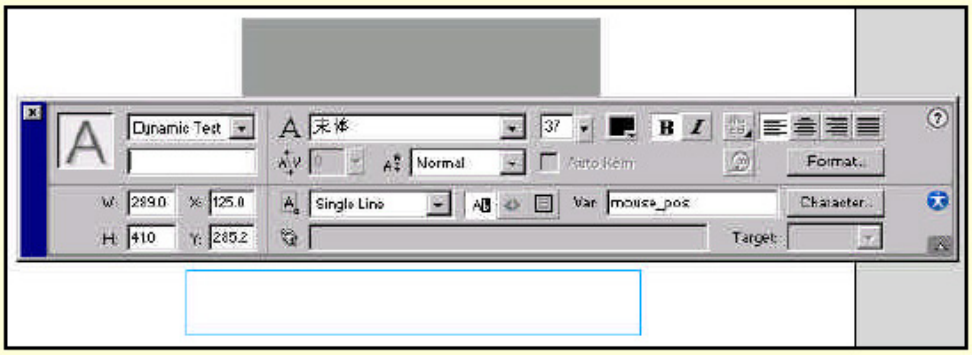


图 7-14　设置主场景文本区域属性

（4）为影片剪辑加入如下脚本：

```
onClipEvent (mouseMove) {
    mouse_pos = _xmouse + "," + _ymouse;
    _root.mouse_pos = _root._xmouse + "," + _root._ymouse;
}
```

（5）输出并执行动画，在动画区域中移动鼠标，鼠标指针的坐标将以图 7-15 所示的方式显示出来，上图所示是以影片剪辑中心为原点的坐标，下图所示是以动画右上角为原点的坐标。

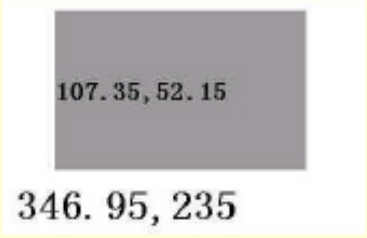


图 7-15　鼠标指针的坐标

7.4　创建自己独特风格的鼠标指针

当用户将鼠标指针移动到 Flash MX 动画或者动画的指定区域中时，鼠标指针将变成具有独特风格的动画鼠标指针。

实现这种效果的具体思路是：先将标准的鼠标指针隐藏，再开启 Movie Clip（影片剪辑）的 startDrag 动作，使它作为鼠标指针跟随用户的鼠标移动，就相当于改变了鼠标的指针。其实现过程如下：

（1）新建 Flash MX 文件，创建一个 Movie Clip（影片剪辑）命名为 Curcor，在影片剪辑的第 1 帧中使用直线工具绘制一个鼠标的轮廓，并用颜料桶工具填充为白色。影片剪辑的中心就是鼠标的焦点所在。

（2）添加两个新层 Layer2、Layer3，把 Layer1 层的第 1 帧（鼠标指针）复制到 Layer2 的第 1 帧上。在 Layer3 层上绘制一个填充色为黑色/白色/黑色条形渐变的矩形，并旋转使其与鼠标指针平行。按照图 7-16 所示调整层的顺序，其结果如图 7-17 所示。

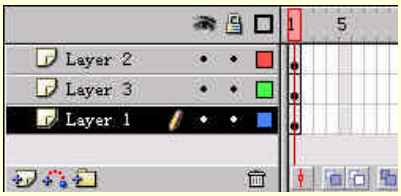


图 7-16　调整层的顺序

图 7-17　调整结果

（3）在 Layer3 层的第 1 帧上右击鼠标，选择快捷菜单中的 Create Motion Tween 菜单项，并在第 15 帧和第 30 帧处插入关键帧，把第 15 帧处的矩形放大。其时间轴如图 7-18 所示。

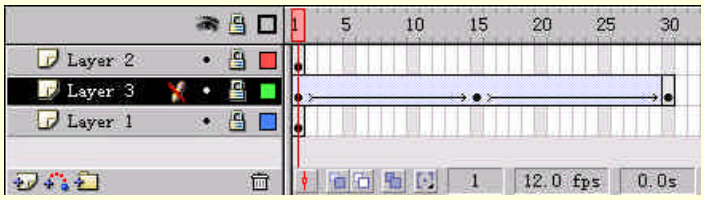


图 7-18　时间轴

（4）分别在 Layer1、Layer2 层的第 30 帧处插入帧，并将 Layer2 设为 Mask（遮罩）层（如图 7-19 所示）。这样，就完成了一个有颜色变化的鼠标指针的影片剪辑。

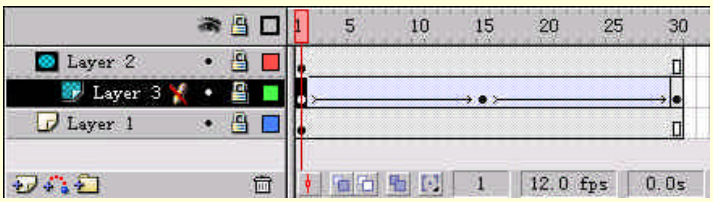


图 7-19　设置遮罩层

（5）回到舞台上，把影片剪辑 Curcor 拖放到场景中，并为它添加如下脚本：

```
onClipEvent (load) {
    Mouse.hide();
    startDrag (this, true);
}
onClipEvent (mouseMove) {
    if (_root.hitTest(_root._xmouse,_root._ymouse,false)) {
        this._visible=true;
    } else {
        this._visible=false;
```

```
        }
    }
```

✍ 在 onClipEvent (load)事件中，隐藏鼠标并开启 startDrag。onClipEvent (mouseMove)事件中，判断鼠标是否在动画的区域内，如果在则显示影片剪辑 Curcor，否则隐藏影片剪辑 Curcor。

（6）输出并运行动画。当鼠标移动到动画区域时指针将变成明暗交替变化的箭头。

7.5 测试鼠标指针的移动速度

用户在移动鼠标时，动态地测试并显示出鼠标指针的移动速度（包括横向速度、纵向速度、总速度）。

它的设计思路是：Movie Clip（影片剪辑）的 EnterFrame 事件在动画每播放一帧的时候都产生（默认频率是 12Hz），把它作为定时器，计算出每两次 EnterFrame 事件中鼠标指针所在位置之差，就可以计算出速度。实现过程如下：

（1）新建一个 Flash MX 文件，创建一个影片剪辑命名为 Timer。在影片剪辑的第 1 帧上添加 3 个图 7-20 所示的 Dynamic（动态）的文本区域，分别把 Variable（变量名称）设置为：xspeed、yspeed、speed。

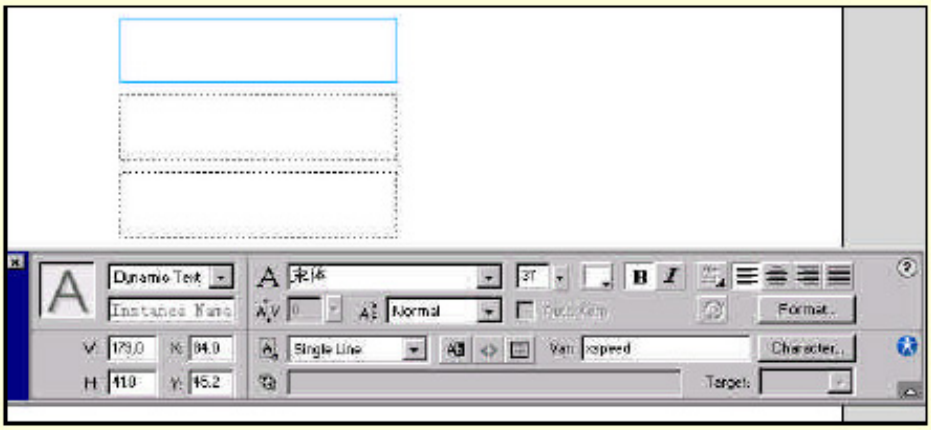


图 7-20 添加文本区域

（2）返回到舞台上，把影片剪辑 Timer 拖放到主场景中，并为它加入如下 ActionScript 脚本：

```
onClipEvent (load) {
        xpoint = _xmouse;
        ypoint = _ymouse;
}
onClipEvent (enterFrame) {
        var x,y,total;
        x = int((_xmouse-xpoint)/0.12)/100;
        y = int((_ymouse-ypoint)/0.12)/100;

        xspeed = "横向速度："+x+"像素/秒";
        yspeed = "纵向速度："+y+"像素/秒";
        total = int(Math.sqrt (x*x+y*y)*100)/100;
        speed = "总速度："+total+"像素/S  秒";
        xpoint = _xmouse;
        ypoint = _ymouse;
}
```

（3）输出并运行动画。当用户移动鼠标的时候将以图 7-21 的方式显示出鼠标指针移动的速度。

图 7-21　鼠标指针的速度

7.6　捕获键盘按键

使用 ActionScript 脚本的预定义对象 Key 可以响应键盘的事件。Key 对象没有构造函数，所以可以不需要构造 Key 对象的实例而直接使用它的方法，例如：

```
Key.getCode();
```

可以获得用户最近一次所按下的按键的扫描码。如果用户按下的是 ASCII 码键，可以使用如下脚本得到最近一次所按下的 ASCII 码键：

```
Key.getAscii();
```

扫描码直接对应了键盘的按键，键盘上的任何一个按键都有扫描码，例如：向左的箭头所对应的扫描码为 37。Key 对象中的许多只读属性对应了这些特殊按键，使用时直接调用即可。例如：

```
Key.LEFT、Key.RIGHT
```

ASCII 码对应了键盘所键入的 127 个有效字符，所以，大写字母“ A ”和小写字母“ a ”所对应的 ASCII 码不同。

下面的例子将实现控制一束灯光照在屏幕上的效果。

（1）新建一个 Flash MX 文件，创建一个图形元件，命名为 Lamp。在影片剪辑的第 1 帧上绘制一个类似于图 7-22 的填充色为白色/黑色径向渐变色的圆。

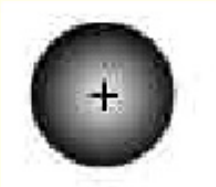


图 7-22　绘制圆

（2）回到舞台上，把图形元件 Lamp 拖放到主场景中，并在它的 Properties 面板中把它的 Behavior 属性改为 Movie Clip（影片剪辑），如图 7-23 所示。

图 7-23　将行为改为影片剪辑

（3）为影片剪辑添加如下脚本：

```
onClipEvent (keyDown) {
    if (Key.getCode()==Key.LEFT) {
        _x-=10;
```

```
        } else if (Key.getCode()==Key.RIGHT) {
            _x+=10;
        } else if (Key.getCode()==Key.UP) {
            _y-=10;
        } else if (Key.getCode()==Key.DOWN) {
            _y+=10;
        } else if (Key.getCode()==Key.PGDN) {
            _xscale*=4/5;
            _yscale*=4/5;
        } else if (Key.getCode()==Key.PGUP) {
            _xscale*=6/5;
            _yscale*=6/5;
        }
    }
```

（4）新建一个层 Layer2，在此层上用 Text Tool（文字工具）添加一行文字作为背景（如“ FLASH ”），效果如图 7-24 所示。

图 7-24　添加背景

（5）将 Layer2 层设为遮罩层，并将主场景的背景设为图 7-25 所示的黑色。

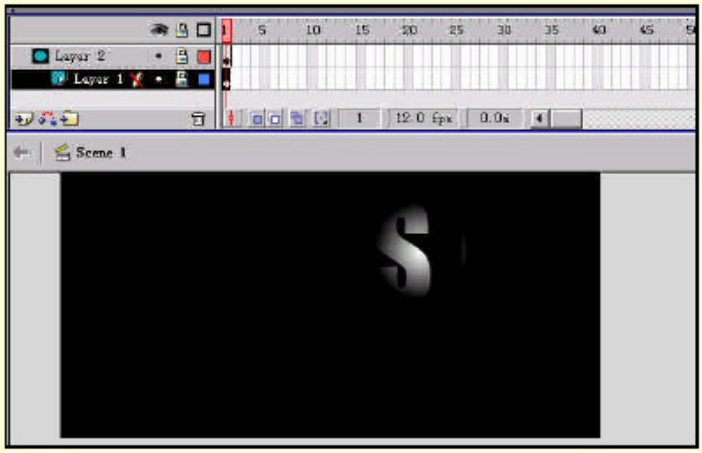


图 7-25　设 Layer2 层为遮罩层

（6）输出并运行动画。用户可以通过键盘上的方向键控制灯光向上、下、左、右移动，或者使用 PageUp、PageDown 键增大或者缩小光圈 20%，效果如图 7-26 所示。

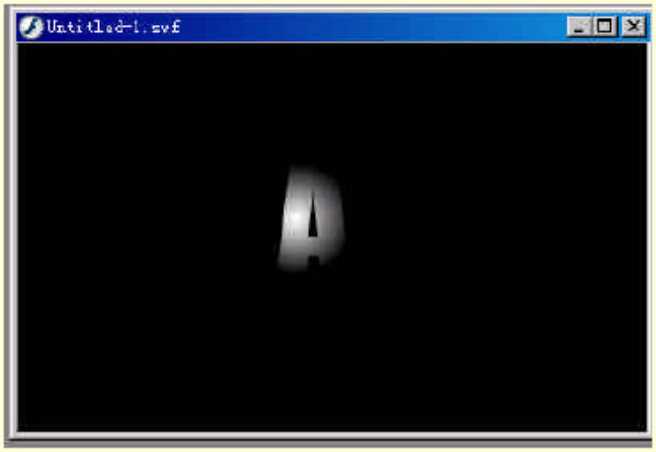


图 7-26　运行结果

7.7　幻影移动

如图 7-27 所示为幻影运动，物体在运动时在自己的运动轨迹上留下一串逐渐消失的虚影。

其实，实现这种效果有很多种方法。较复杂的方法，就是自己创建一个运动物体影片剪辑，把它们放在主场景的相邻的帧上，并使其透明度逐渐变小，使得动画在运行时在不同的时间内连续播放同一影片，即可产生此效果，但是上述过程过于麻烦。下面给出的例子中，把在不同时刻创建影片剪辑的任务交给 ActionScript 脚本完成，所以大大地简化了动画的创建过程。

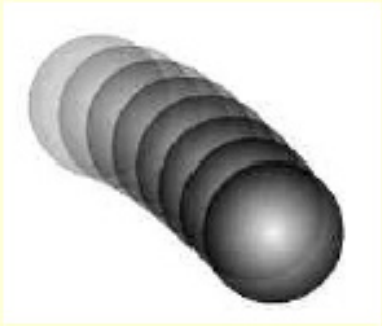


图 7-27　幻影运动的物体

（1）新建一个 Flash MX 文件，如图 7-28 所示，在主场景的第一帧上用直线工具绘制一条直线并将其拉弯。

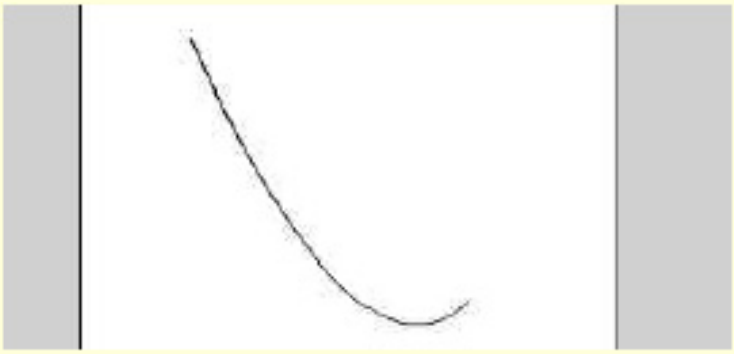


图 7-28　拉弯直线

（2）如图 7-29 所示，使用钢笔工具选中直线，并在直线上创建一个拐点。

（3）使用箭头工具将直线拉成如图 7-30 所示的形状。

虽然步骤（1）~（3）的过程只是为创建一条曲线，但是使用上述方法可以保证曲线只有一个拐点。这使得后面添加的球体的沿曲线运动的过程有沿着一个曲面向下滑落的感觉。

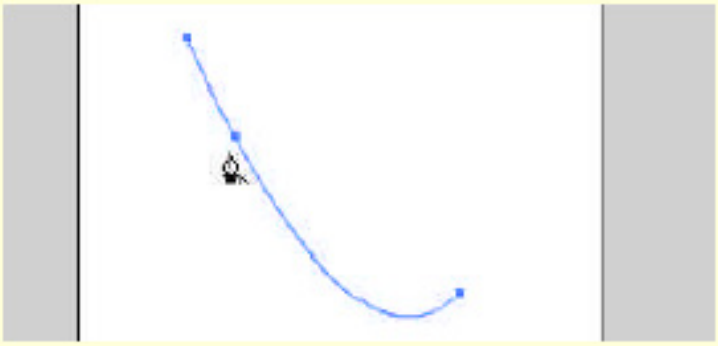


图 7-29　为直线创建拐点

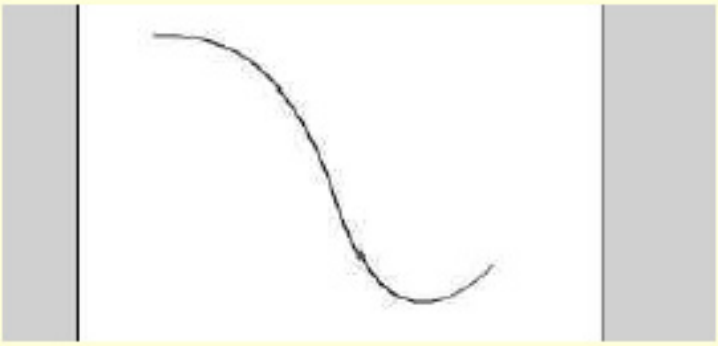


图 7-30　改变后曲线形状

（4）选中曲线，使用 Insert | Convert to Symbol 菜单命令，将曲线转换成影片剪辑。并命名为 ball。

之所以把主场景中的曲线转变成影片剪辑，而没有先创建影片剪辑再绘制曲线，是因为采用前者的方式可以预知影片剪辑在主场景中的大小和位置，为绘制曲线提供方便。

（5）双击曲线进入影片剪辑编辑状态。在影片剪辑中插入一个新层 Layer2，把它移动到曲线所在层 Layer1 下面。并将曲线所在层 Layer1 定义为 Guide（向导）层，将新层定义为 Guided（被向导层），这时的时间轴如图 7-31 所示。

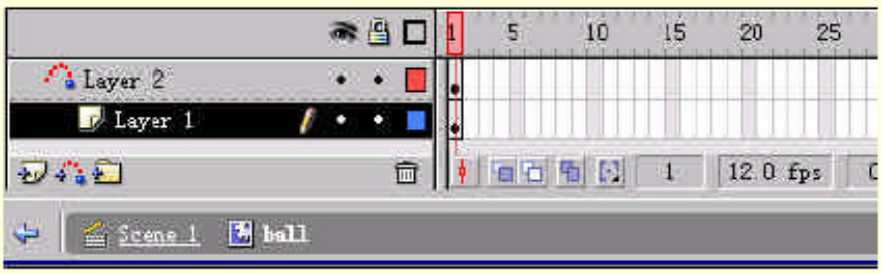


图 7-31　步骤 5 的时间轴

（6）在 Layer2 层中加入一个填充色为白色/黑色径向渐变的圆。如图 7-32 所示将此圆组合成组，并移动到曲线的上端。

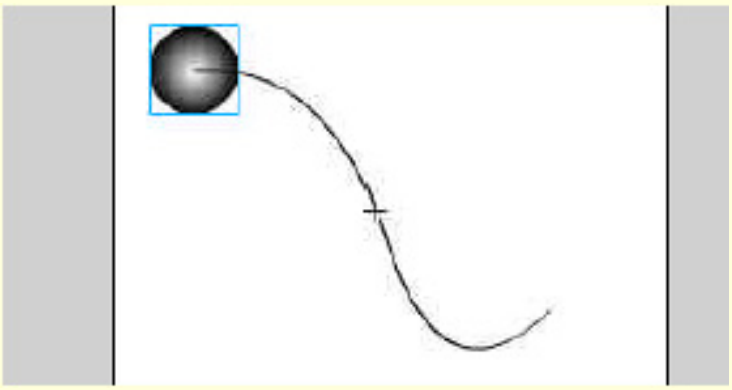


图 7-32　放置圆球

（7）在影片剪辑的 Layer2 层的第 30 帧处创建关键帧，Layer1 层的第 30 帧处插入帧。如图 7-33 所示，在第 30 帧处把圆球移动到曲线的末尾，并在第 1 帧到第 30 帧之间创建位置渐变。

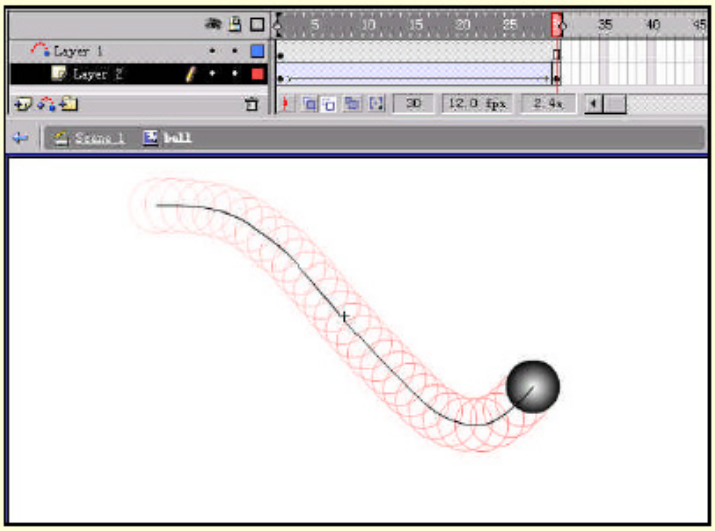


图 7-33　创建位置渐变

（8）在影片剪辑的向导层上再插入一个新层 Layer3，将向导层的曲线复制到新建的层上。如图 7-34 所示，打开时间轴的洋葱皮轮廓显示模式，并使其显示所有帧的轮廓。如图 7-35 所示，改变 Layer3 层上帧的位置及形状，使其尽量拟合圆球运动的轨迹。

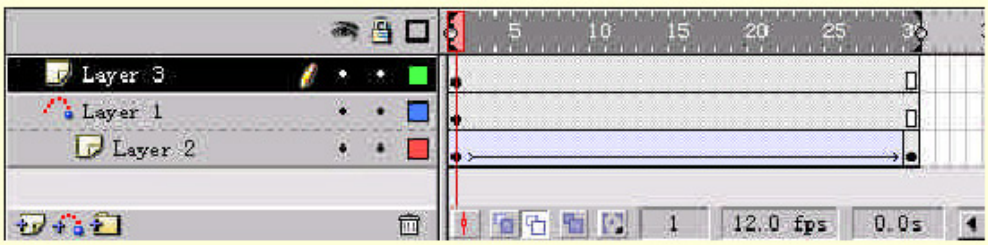


图 7-34　步骤（8）的时间轴

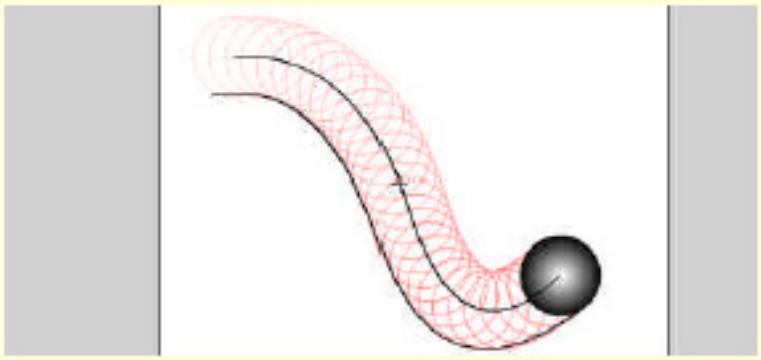


图 7-35　拟合曲线

（9）把拟合好的曲线复制到主场景的第 1 帧中，并为圆球创建如图 7-36 所示的运动曲面，同时删除影片剪辑中的 Layer3 层。

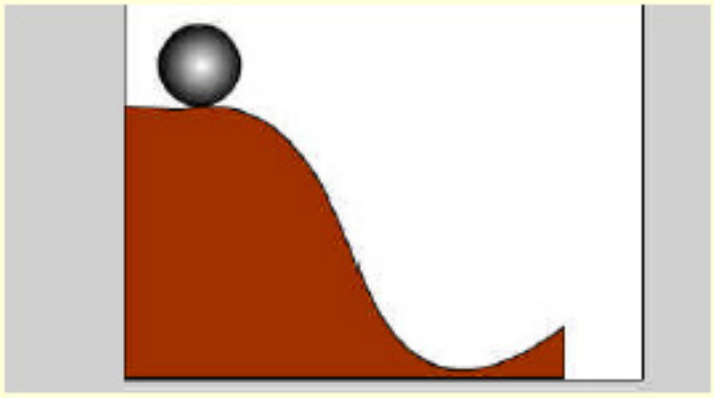


图 7-36　创建运动曲面

（10）双击回到影片剪辑中，选中 Layer2 层。如图 7-37 所示，调节位置渐变的速度比例，使圆球的运动有一种加速下滑的感觉。

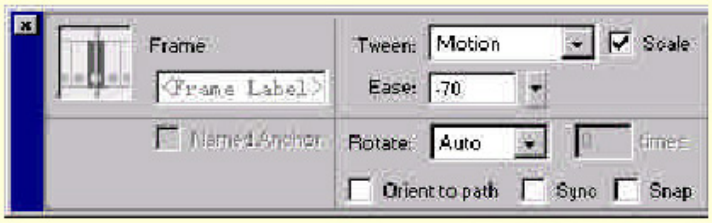


图 7-37　调节球运动的速度

✎ 在这一步骤调节球的运动速度，而没有在创建运动渐变的同时调节，是为了更容易地创建拟合曲线。

（11）把影片剪辑中所有的帧向后移动两帧，并在 Layer1 的第 2 帧处插入空白帧，这时的时间轴如图 7-38 所示。

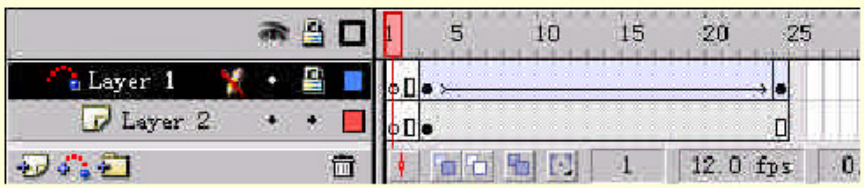


图 7-38　插入空白帧

（12）在影片剪辑 Layer1 层的第 2 帧处插入如下脚本:

```
if (--timer==0) {
    gotoAndPlay (3);
    return;
}
gotoAndPlay (1);
```

✍ 变量 timer 在这里作为一个定时器，只有当它等于 1 时影片剪辑才开始播放。所以，只要在主场景中为多个影片剪辑的 timer 变量赋予不同的数值，这样，即使是在同一帧中或者同一时刻创建的影片剪辑，这些影片剪辑也会在不同的时间开始播放。

（13）切换回主场景中，在如图 7-39 所示的位置加入一个播放按钮。

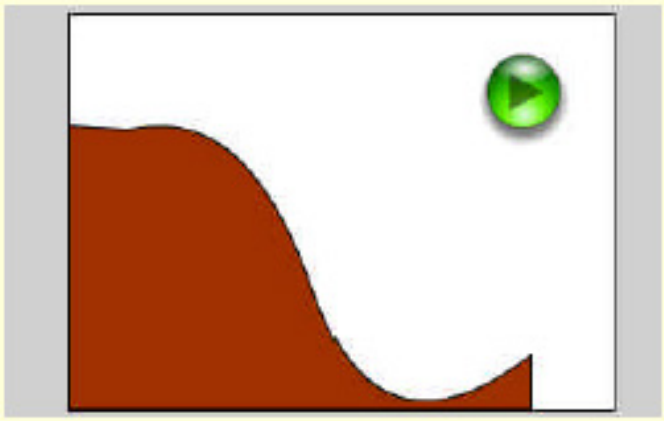


图 7-39 加入播放按钮

（14）如图 7-40 所示，将影片剪辑命名为 ball。

✍ 虽然影片剪辑的前两帧为空白帧，但是，在编辑的时候，在影片剪截中心位置会显示一个空心圆圈。

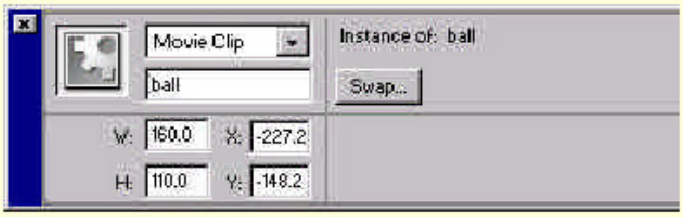


图 7-40 为影片剪辑命名

（15）在主场景的第 1 帧处加入如下脚本：

```
number=9;
for (n=0; n<number; n++) {
    ball.stop();
    alpha_step=ball._alpha=100/(number+1);
    duplicateMovieClip ("ball", "ball"+n, n);
    this["ball"+n].stop();
    this["ball"+n]._alpha=alpha_step*(n+1);
}
```

（16）为按钮添加如下脚本：

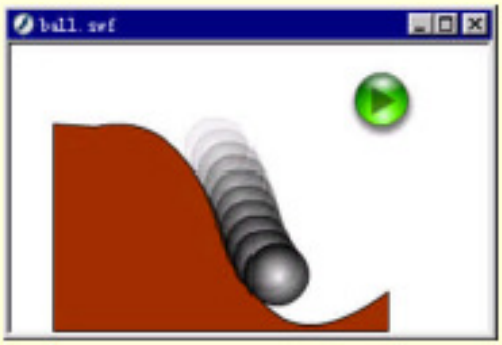
```
on (release) {
    ball.timer=number+1;
```

```
        ball.play();
        for (n=0; n<number; n++) {
            this["ball"+n].timer=number-n;
            this["ball"+n].play();
        }
    }
```

（17）为影片剪辑的最后 1 帧加入如下脚本：

```
Stop();
```

（18）输出并运行动画。如图 7-41 所示，单击播放按钮使圆球开始沿曲面幻影式地下滑。动画结束后，按播放按钮可以反复播放。



7-41 运行结果

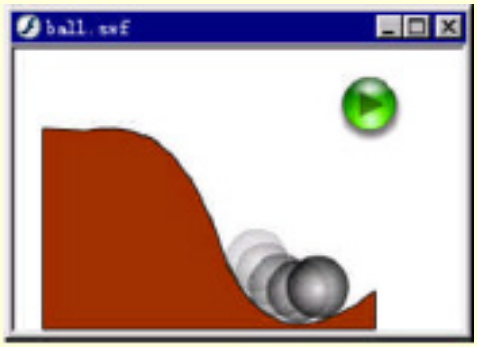


图 7-42 改变幻影数图

✍ 通过修改主场景中变量 number 的数值可以改变幻影的个数。其结果如图 7-42 所示。

第 8 章　组件的使用

8.1　综　述

随着 Flash 的普及程度越来越高，Flash 动画应用的领域越来越广，Flash 动画制作过程中的资源共享也越来越被人们所重视。这种资源共享不仅仅是简单的如图像、声音等艺术素材的共享，更重要的是具有一定功能的动画元素的共享。

正是为了满足这一要求，Flash MX 引进了在 Visual Basic 程序设计中应用的 Component（组件）这一概念（VB 中有时也称为控件），替代和扩展了 Flash 5 中的 SmartClip（智能剪辑）功能，而且在 Flash MX 中已经定制了一些常用的组件，比如 ScrollBar（滚动条）、ListBox（列表框）等我们在 HTML 页面文件中常用的功能组件，这些定制的常用组件不仅减少了开发者的开发时间，提高了工作效率，而且能给 Flash 作品带来统一的标准化的界面。

8.2　组件的概述

Flash MX 中的 Component（组件）概念是由 Flash 5 中的 SmartClip（智能剪辑）延伸而来的，是拥有某种功能的 MovieClip（影片剪辑）元件，具有以下特点：

- 可重复应用于同一个 Flash MX 文件中，也可在不同的 Flash MX 文件中多次使用。
- 对于有参数变量的组件，可通过改变参数来改变组件的属性。
- 在参数改变的情况下，组件的功能不变。
- 有可以编辑的外观。

比如一个如图 8-1 所示的拨盘组件，指针的拨动范围（最大值和最小值）是可调的参数。

图 8-1　拨盘组件

Flash MX 中已经为用户内建了一些常用的组件，放在 Components（组件）面板中，如图 8-2 所示。当然，用户也可以自定义一些常用的组件或共享他人的组件资源。

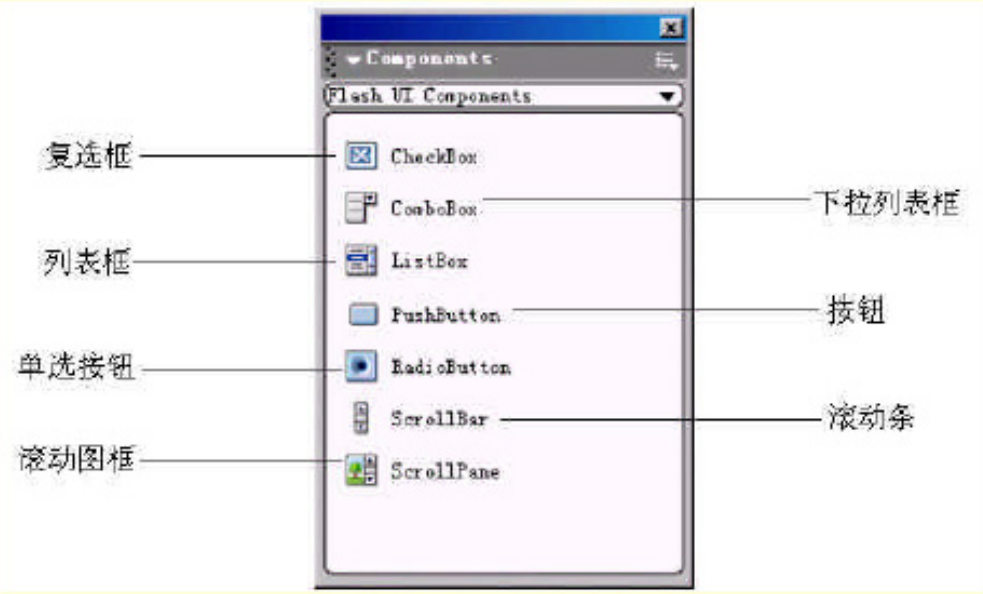


图 8-2 Flash MX 内建组建

8.3 自定义组件

8.3.1 创建自定义组件的步骤

Flash MX 的内建组件虽然使用方便，功能也很强大，但要应付越来越丰富的 Flash 动画制作显然是不够的，因此有必要根据需要创建自己的通用组件。

一般来说，创建自定义组件分以下几个步骤：

（1）创建电影剪辑：创建一个电影剪辑作为自定义组件的基础部件。

（2）定义组件类：即为上述电影剪辑编写 ActionScript 脚本语言，这是创建自定义组件的关键，这一步决定了自定义组件的功能。

（3）定义组件参数：为上述的电影剪辑添加参数，这些参数将决定自定义组件的某些属性。

（4）安装组件到 Flash MX 程序中。

在 8.4.3 节中将以制作一个通用滑动条为例，演示如何创建一个自定义组件。

8.3.2 为影片剪辑添加参数

Flash MX 的 Movie Clip（影片剪辑）本身就是一个对象，它不但有自己的属性和方法，而且还可以让用户定义自己的属性。为 Movie Clip（影片剪辑）定义属性与其他对象不同，Flash MX 对此有专门的操作。

创建一个影片剪辑以后，打开 Library（元件库）面板，在影片剪辑上右击鼠标，选中弹出菜单的 Component Definition 菜单项。将弹出图 8-3 所示的定义影片剪辑参数的对话框。

使用“+”或者“－”按钮可以为影片剪辑增加或者删除属性。用户自定义的影片剪辑属性将在图 8-4 所示的 Parameters（参数）列表框中列出。

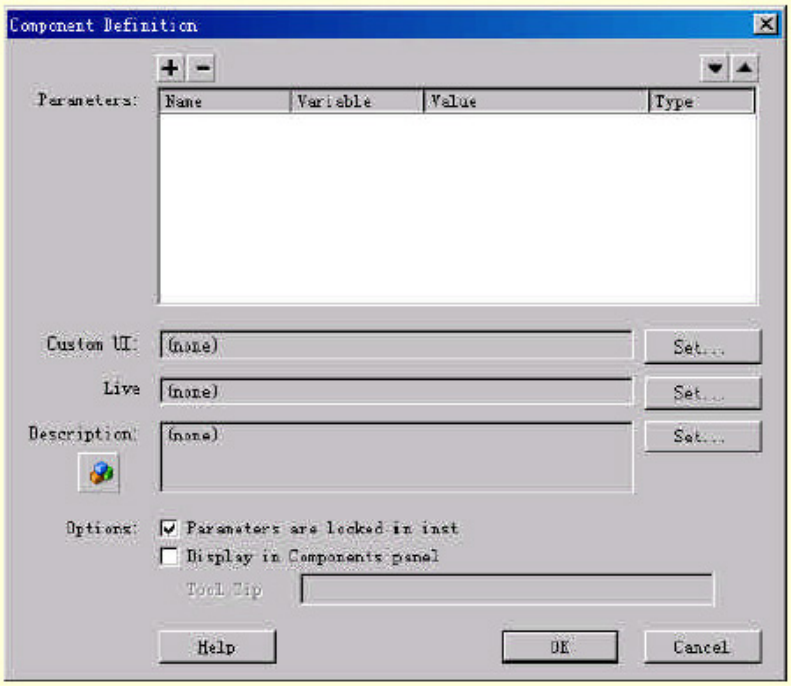


图 8-3　定义影片剪辑参数

Name	Variable	Value	Type
Label	label	Check Box	Default
Initial Value	initialValue	false	Boolean
Label Placement	labelPlacement	right	List
Change Handler	changeHandler		Default

图 8-4　参数列表

- Name 列代表属性名称。
- Variable 列代表该属性对应的 ActionScript 脚本语言中的变量的名称。
- Value 列表示属性的默认值，可以为空。如果属性的类型为 Array、Object 或者 List，双击此项后还可以弹出对话框，以便继续为其添加属性或者设置初始值，如图 8-5 所示。

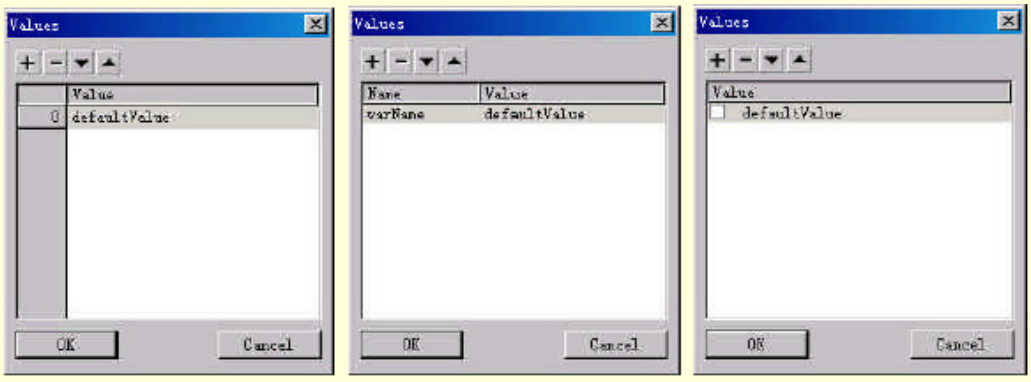


图 8-5　添加属性或者设置初始值

- Type 列表示属性的类型，包括：
  - Default：一般的 ActionScript 脚本变量，如数字、字符串、布尔值
  - Array：数组
  - Object：对象
  - List：当使用影片剪辑的时候，提供供用户选择参数的列表
  - String：字符串

- Number：数字
- Boolean：布尔值
- Font Name：字体
- Color：颜色

自定义的影片剪辑属性和对象的属性在使用时几乎没有任何区别，使用点（.）运算符可以直接访问，例如：

```
result = myMovie.myValue/3。
```

如果要在影片剪辑的内部使用它的属性，直接给属性对应的变量赋值即可，例如：

```
myValue=8;
```

如果属性的层次复杂，可以使用关键字_parent 来确定层次。类似于_root，关键字_parent 也是确定目标路径的一种方法，它表示当前层次的上一个层次（父层）对象。可见如果使用_parent 定位，所有对象的层次都是相对的，也就是说，影片剪辑可以随意地在任何地方使用而不用考虑其绝对路径。

8.3.3 通用滑动条的制作

下面的例子制作了一个类似于 Windows 风格的 Slider Bar（滑动条），用鼠标可以拖动滑块设定相应的值。

（1）新建一个 Flash MX 文件，创建一个图形元件命名为 Slider_Bar。在图形元件的中央，用矩形工具绘制一个细长的矩形，描绘颜色为黑色，宽度为 3；填充色为深灰色（#666666）。并将矩形的右边和下边的描绘色改为浅灰色（#CCCCCC），使它像一个嵌入到屏幕中的凹槽，效果如图 8-6 所示。

图 8-6 绘制凹槽

（2）再创建一个图形元件，命名为 Slider_Point。使用上述方法绘制滑块，注意居中对齐，效果如图 8-7 所示。

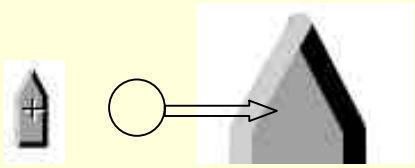


图 8-7 绘制滑块

（3）创建一个影片剪辑元件命名为 Slider。在影片剪辑中创建一个层（Layer2），把图形元件 Slider_Bar、Slider_Point 分别拖放到 Layer1、Layer2 层，并使 Slider_Point 所在层在上。所有元件均居中对齐排列，如图 8-8 所示。

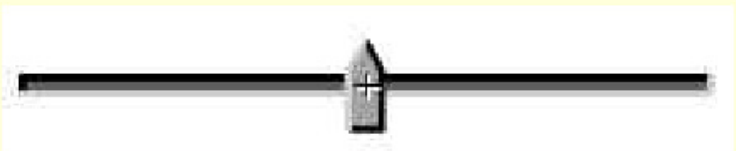


图 8-8 创建影片剪辑

（4）在 Properties（属性）面板中改变上述图形元件的 Behavior 属性为 Movie Clip（影片剪辑），并将凹槽图形 Slider_Bar 命名为 bar，滑块 Slider_Point 命名为 point，如图 8-9 所示。

图 8-9　改变作用

（5）为滑块 Slider_point 添加如下脚本：

```
onClipEvent (mouseMove) {
    if (_parent.bsetpos==false) {
        return;
    }
    _x=_parent._xmouse;
    _x=Math.min(_x,_parent.bar._width/2);
    _x=Math.max(_x,-_parent.bar._width/2);
    range=(_parent.max_range-_parent.min_range);
    _parent.pos = _parent.min_range+
        (_x+_parent.bar._width/2)*range/_parent.bar._width;
    _parent.pos = int(_parent.pos*100)/100;
}
onClipEvent (mouseDown) {
    if ((_xmouse<-_width/2) || (_xmouse>_width/2)) {
        return;
    }
    if ((_ymouse<-_height/2) || (_ymouse>_height/2)) {
        return;
    }
    _parent.bsetpos = true;
}
onClipEvent (mouseUp) {
    _parent.bsetpos = false;
}
onClipEvent (load) {
    _parent.bsetpos=false;
}
```

✎ **在上述的脚本中，使用了_parent 关键字来指明目标路径，从而使影片剪辑的脚本具有通用性。无论影片剪辑处于路径中的哪一个层次，脚本均有效。**

（6）在影片剪辑的第 1 帧中加入如下脚本：

```
Updatapos();
function setpos (position) {
    pos = position;
    Updatapos();
}
function Updatapos () {
```

```
        range = max_range-min_range;
        point._x = bar._width/range*(pos-min_range)-bar._width/2;
    }
```

（7）为影片剪辑 Slider 添加属性，bsetpos（正在移动滑块）、pos（滑块位置）、max_range（最大取值）、min_range（最小取值）。如图 8-10 所示。

Name	Variable	Value	Type
Sliding?	bsetpos	false	Boolean
Position	pos	50	Default
Max Range	max_range	100	Default
Min Range	min_range	0	Default

图 8-10　影片剪辑 Slider 的属性

（8）在 Component Definition（组件定义）对话框中还有几个选项可以为自定义组件设置一些个性化的特性，列举如下：

- Custom UI：给组件提供一个外部的.swf 文件链接，这个文件可以显示在属性面板中，通过改变.swf 文件中的参数，可以直观地反映组件的外观。
- Live：与 Custom UI 相似，给组件提供一个外部的.swf 文件链接，这个.swf 文件可以不通过播放器直接在源文件中显示组件的外观和功能，真正做到所见即所得。
- Description：说明组件的功能、作者和版权等相关信息。
- Icon：在 Description 下面有一个  按钮，单击可选择组件在组件面板中的图标。
- Option：有两个选项 Parameters are locked in inst 和 Display in Component panel，分别表示是否允许用户添加或删除组件的属性和是否在组件面板中显示该组件。
- Tool Tip：当选中 Display in Component panel 复选框后，可以在 Tool Tip 文本框中输入组件在组件面板中的提示标签。

按图 8-11 所示设置好 Component Definition（组件定义）对话框，单击 OK 按钮，完成组件属性的设置。

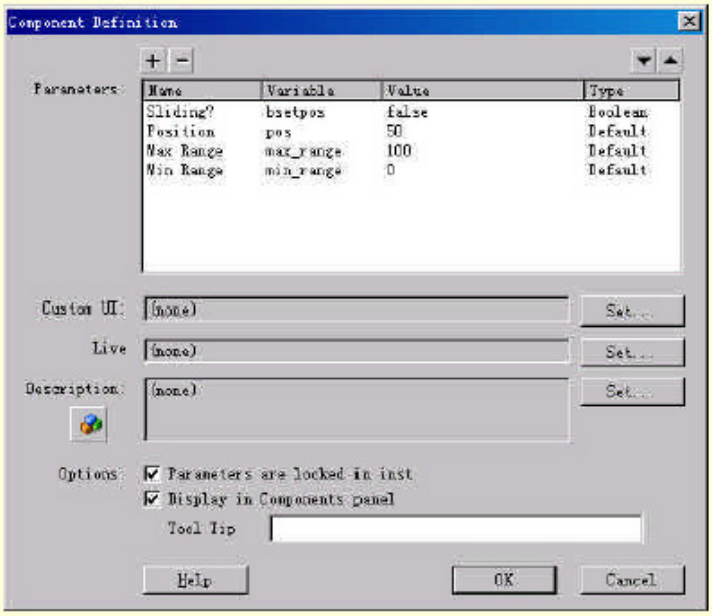


图 8-11　组件属性设置

（9）将当前的 Flash MX 文件存为 SliderBar.fla，这样一个自定义组件就基本上完成了，但是这个时候在 Components 面板中还看到这个组件，因此我们还要做最后一道“工序”，将组件的源文件放在正确

的文件夹内。

对于不同的操作系统，组件源文件的存放位置是不同的，列举如下：

- Windows 2000 / XP：C:\ Documents and Settings \ user \ Application Data\Macromedia \ Flash MX \ Configuration \ Components
- Windows 98 / ME：C:\ Windows \ Application Data \ Macromedia \ Flash MX \ Configuration \ Components
- Windows NT：Windows directory \ profiles \ user \ Application Data \ Macromedia \ Flash MX \ Configuration \ Components
- Macintosh OS X：HardDrive / Users / Library / Application Support / Macromedia / Flash MX / Configuration / Components
- Macintosh System 9.x , single user：HardDrive / System folder / Application Support / Macromedia / Flash MX / Configuration / Components
- Macintosh System 9.x , multiple users：HardDrive / Users / user / Documents / Macromedia / Flash MX / Configuration / Components

将 SliderBar.fla 文件根据操作系统类型存放到相应的文件夹内，重新启动 Flash MX，在 Components（组件）面板的下拉列表框中选择 SliderBar，就可以看到刚做好的滑动条组件了，如图 8-12 所示。

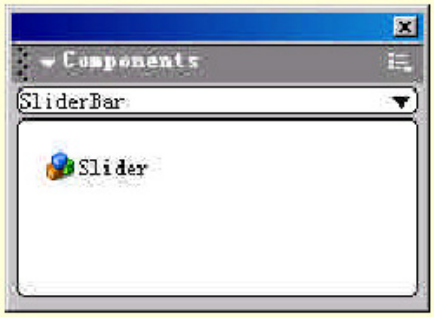


图 8-12 组件面板中的自定义组件

8.3.4 使用自定义的组件

这一节通过一个动态设置影片剪辑颜色的实例来验证上一节中通用滑动条的功能，同时也将演示如何在 Flash MX 动画文件中使用自定义组件。

使用 Flash MX 的 ActionScript 脚本所提供的 Color（颜色）对象，可以调整或者设置 Movie Clip（影片剪辑）的颜色。Color 对象提供了两种控制颜色的方法：setRGB 和 setTransform。

- setRGB 可以设置指定影片剪辑的颜色，其格式为 mycolor.setRGB(0xRRGGBB)。其中 0xRRGGBB 为红绿蓝所组成的颜色值。RR 为红色，GG 为绿色，BB 为蓝色，
- setTransform 可以改变影片剪辑中红、绿、蓝颜色的比例或者调节透明度（alpha）。其格式为 myColor.setTransform(colorTransformObject)。其中，colorTransformObject 是一个普通的对象，它必须包括以下 8 个属性：ra, rb, ga, gb, ba, bb, aa, ab。
  - ra 表示 Color（颜色）对象中红色成分的百分比（取值范围：-100~100）。
  - rb 表示 Color（颜色）对象中红色的偏移量（取值范围：-255~255）。
  - ga 表示 Color（颜色）对象中绿色成分的百分比（取值范围：-100~100）。
  - gb 表示 Color（颜色）对象中绿色的偏移量（取值范围：-255~255）。
  - ba 表示 Color（颜色）对象中蓝色成分的百分比（取值范围：-100~100）。
  - bb 表示 Color（颜色）对象中蓝色的偏移量（取值范围：-255~255）。
  - aa 表示 Color（颜色）对象中 alpha（透明度）的百分比（取值范围：-100~100）。
  - ab 表示 Color（颜色）对象中 alpha（透明度）的偏移量（取值范围：-255~255）。

下面的脚本直接生成了一个 colorTransformObject 对象，并为它的属性赋值：

```
myColorTransform = new Object();
myColorTransform.ra = 50;
myColorTransform.rb = 244;
myColorTransform.ga = 40;
myColorTransform.gb = 112;
myColorTransform.ba = 12;
myColorTransform.bb = 90;
myColorTransform.aa = 40;
myColorTransform.ab = 70;
```

其实，也可以简单地使用如下脚本为定义对象变量并为其属性赋值：

```
myColorTransform = { ra: '50', rb: '244', ga: '40', gb: '112', ba: '12', bb: '90', aa: '40', ab: '70'}
```

下面的例子将说明如何动态地控制影片剪辑的颜色。

（1）新建一个 Flash MX **文件，把影片的宽度适当加大，拖放一个图形元件到舞台上，并按照图** 8-13 **的方式添加静态文本区域。**

图 8-13　**添加文本及图形**

（2）在每一行静态文本的旁边，添加一个输入型的文本区域，由上至下分别将其变量命名为颜色、ra、rb、ga、gb、ba、bb、aa、ab。**如图** 8-14 **所示。**

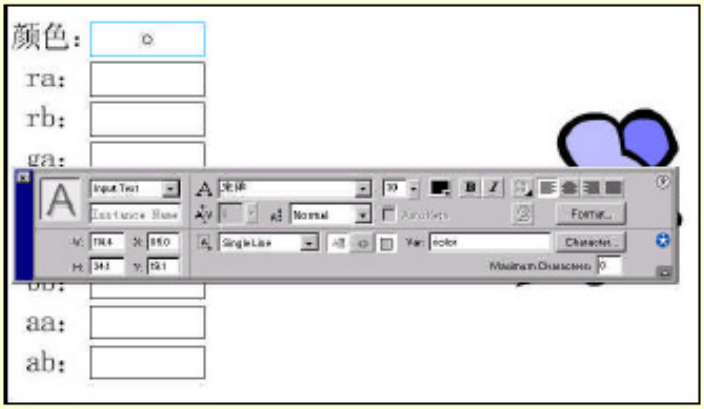


图 8-14　**输入型的文本区域**

（3）选择 Window | Components **菜单命令，将会弹出** Components**（组件）面板，把滑动条组件拖放到场景中，由上至下分别将其命名为** ra_bar、rb_bar、ga_bar、gb_bar、ba_bar、bb_bar、aa_bar、ab_bar，**如图** 8-15 **所示。**

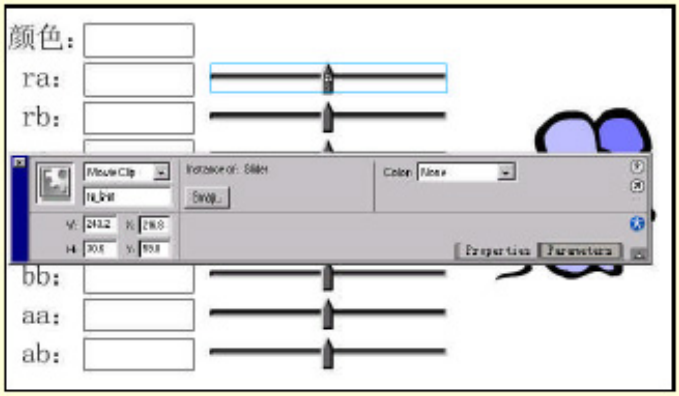


图 8-15　放置滑动条

（4）改变图形元件 Behavior 属性为 Movie Clip，并将它命名为 picture，如图 8-16 所示。

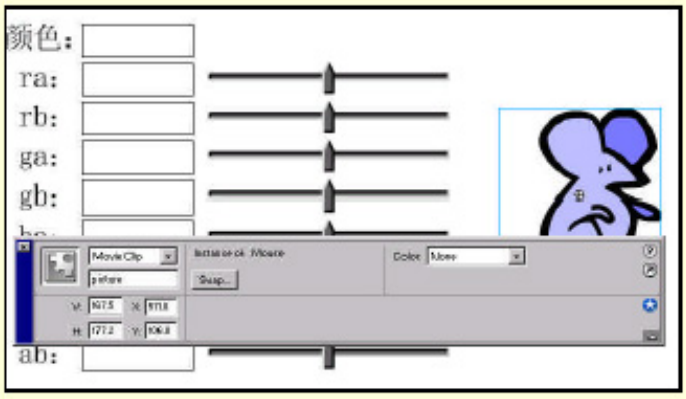


图 8-16　图形元件的作用

（5）如图 8-17 所示，选择组件的 Properties（属性）面板上的 Parameters（参数）标签，设置 ra_bar、ga_bar、ba_bar、aa_bar 组件的默认属性值如图 8-18 所示；设置 rb_bar、bb_bar、gb_bar、ab_bar 组件的默认属性值如图 8-19 所示。

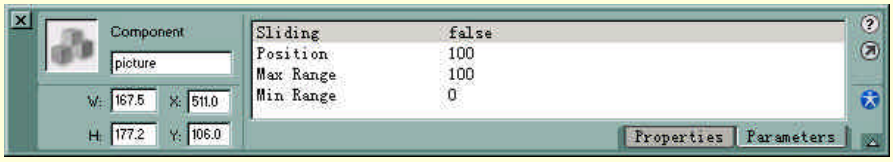


图 8-17　属性面板的 Parametres（参数）标签

Sliding?	false
Position	100
Max Range	100
Min Range	0

图 8-18　组件的默认属性值图

Sliding?	false
Position	0
Max Range	255
Min Range	-255

图 8-19　组件的默认属性值

（6）在主场景的第 1 帧中添加如下脚本：

```
c = new Color(picture);
myColorTransform = oldColorTransform=c.getTransform();
ini_text();
function ini_text () {
    colour = "x";
    ra = oldColorTransform.ra;
    rb = oldColorTransform.rb;
    ga = oldColorTransform.ga;
    gb = oldColorTransform.gb;
```

```
        ba = oldColorTransform.ba;
        bb = oldColorTransform.bb;
        aa = oldColorTransform.aa;
        ab = oldColorTransform.ab;
        Updatabar();
    }
    function Updatabar () {
        ra_bar.setpos(ra);
        rb_bar.setpos(rb);
        ga_bar.setpos(ga);
        gb_bar.setpos(gb);
        ba_bar.setpos(ba);
        bb_bar.setpos(bb);
        aa_bar.setpos(aa);
        ab_bar.setpos(ab);
    }
    function setcolor () {
        if (colour.toUpperCase() == "X") {
            Updatabar();
            myColorTransform = {ra:ra, rb:rb, ga:ga, gb:gb, ba:ba, bb:bb, aa:aa, ab:ab};
            c.setTransform(myColorTransform);
        } else {
            c.setRGB(parseInt(colour, 16));
        }
    }
```

（7）为影片剪辑 picuture **添加如下脚本：**

```
    onClipEvent (keyDown) {
        if (Key.getCode()==Key.ENTER) {
            _root.setColor();
        }
    }
```

（8）为主场景中的每一个滑动条添加如下脚本：

```
    onClipEvent (mouseMove) {
        if (bsetpos==false) {
            return;
        }
        _root.ra = pos;  //要根据滑动条所对应的文本区域，改变此行（见注意）
        _root.setColor();
    }
```

✍ 为了使不同的滑动条控制相应的文本区域，要改变上述源代码第 5 行为_root.rb =pos、_root.ga=pos、_root.gb=pos、……_root.ab=pos。

（9）从通用元件库中拖出一个按钮到主场景中，作为复位按钮，并为它添加如下脚本，效果如图 8-20 **所示。**

```
on (release) {
    _root.c.setTransform(_root.oldColorTransform);
    ini_text();
}
```

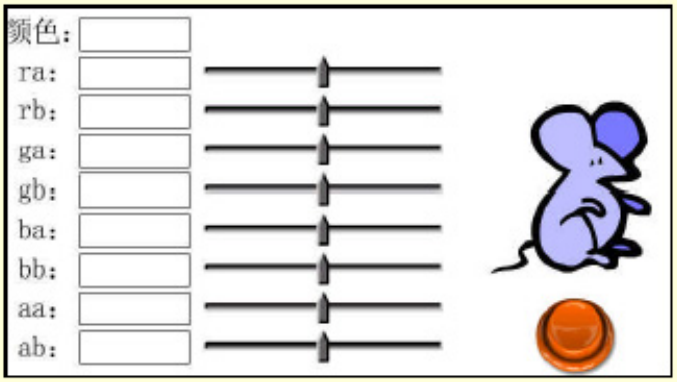


图 8-20　添加复位按钮

（10）输出并测试动画。用户可以通过滑动条或者直接设置图形对象颜色的偏移量和百分比，如图 8-21 所示。如果在颜色文本区域中输入数字，就可以改变图形对象的颜色如图 8-22 所示。使用复位按钮则恢复到图形的起始状态。

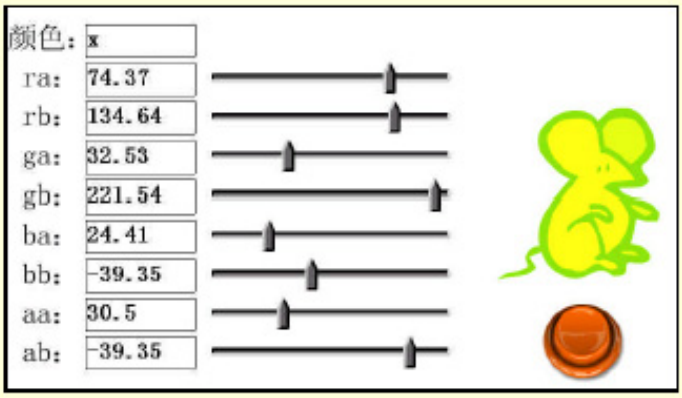


图 8-21　调节颜色偏移量和百分比

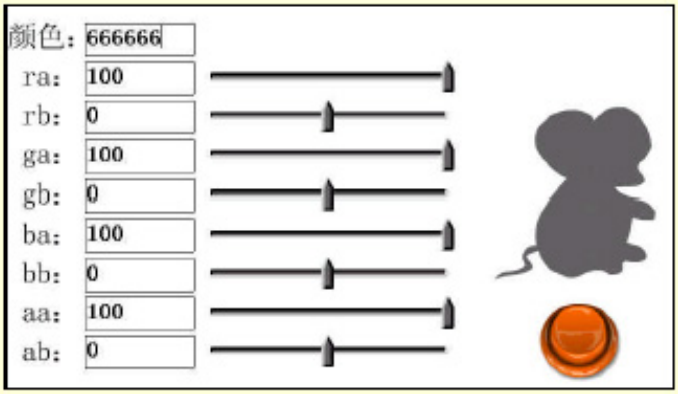


图 8-22　改变颜色

第 9 章　外部文件的导入

9.1　综　述

“巧妇难为无米之炊”——没有丰富的素材，动画的创作将受到很大的限制。但外部素材需要导入到 Flash 中方能使用，所以在学会制作动画的同时也应学会如何导入各种格式的素材。

Flash MX 中可以导入的外部文件的格式包括：

- 图片文件。Flash MX 所支持的图片文件包括：.EPS、.JPEG、.DXF、.WMF、.BMP、.GIF、等多种。
- 声音文件。Flash MX 不仅可以导入.WAV 格式的声音文件，还可以导入.MP3 格式的声音文件。
- 视频文件。Flash MX 除了可以导入它本身所生成的.SWF 动画之外，还可以导入其他格式的视频文件，如：.MOV、.AVI、.DV、.MPEG 等。

9.2　导入图片文件

Flash MX 本身可以导入多种格式的图片文件。如果系统中已装载 Quick Time 4 或以后版本， 那么还可以导入更多格式的文件。Flash MX 还可以导入其他软件的成型作品，如 FreeHand 文件及 Fireworks PNG 文件。

9.2.1　图片文件的类型（矢量图与位图）

矢量图与位图是 Flash 中较为重要的的概念，二者性质不同但联系紧密：数学上所讲的矢量是指既有大小又有方向的量，如对一条矢量线段的描述为此线段的长度有多少，仰角为多少度。计算机中所讲的矢量图是指用矢量化元素描绘的图形，而位图图形使用点来描述图形，类似于显示器用像素点来显示图画，整幅图是由一个个的点组成的，没有矢量的概念。在存储方式上，矢量图存储的是图的矢量信息，如线段长度、仰角、圆心坐标、半径等；而位图存储的是点的信息，即不同位置上点的颜色。相对于不同的图形来说，它们各有优缺点。比如，对于工程图纸和手工绘制的图片而言，用矢量方式进行描绘和存储比较方便，而且文件也较小。如果使用位图进行描绘和存储，在对图形进行放大时，如图 9-1 所示，位图会因为点间距的增大而变得模糊甚至导致图片的变形，而矢量图则不会出现这种情况。但是，如果图片对象是一个照片，对照片进行矢量描述或者绘制是相当困难的，虽然 Flash MX 也可以把位图转换成矢量图，但那将损失大量的图像信息。

基于上述特点，用 Flash 工具绘出的图都是矢量化的，同时 Flash MX 也支持对矢量图与位图的导入。

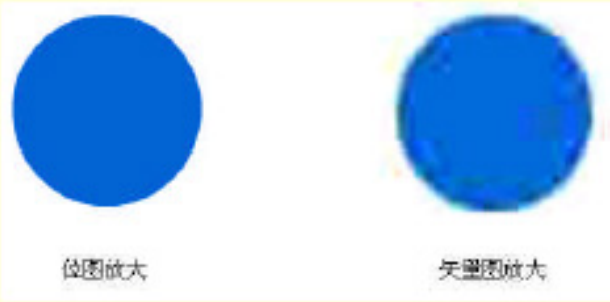


图 9-1 矢量图和位图的放大

无论系统中是否安装 QuickTime 4 或以后版本，以下格式文件均可以导入至 Flash MX 中，但有时也会受到操作系统的限制，如 Bitmap（位图）文件在 Windows 环境下可以实现导入，但在 Macintosh 上却无法导入：

- Adobe Illustrator（扩展名.eps，.ai）：Windows 及 Macintosh 均可导入。
- AutoCAD DXF（.dxf）：Windows 及 Macintosh 均可导入。
- Bitmap（.bmp）：Windows 上可以导入但 Macintosh 无法导入。
- Enhanced Windows Metafile（.emf）：Windows 可以导入但 Macintosh 无法导入。
- FreeHand（fh7,.ft7,.fh8,.ft8,.fh9,.ft9）：Windows 及 Macintosh 均可导入。
- FutureSplash Player（.spl）：Windows 及 Macintosh 均可导入。
- GIF and animated GIF（.gif）：Windows 及 Macintosh 均可导入。
- JPEG：（.jpg）Windows 及 Macintosh 均可导入。
- PICT：（.pic，.pct）Macintosh 可以导入但 Windows 无法导入。
- PNG：（.png）Windows 及 Macintosh 均可导入。
- Flash Player（.swf）：Windows 及 Macintosh 均可导入。
- Windows Metafile（.wmf）：Windows 可以导入但 Macintosh 无法导入。

以下格式的文件只有当系统装载了 QuickTime 4 或其后续版本时才能导入到 Flash MX 中。

- MacPaint（.pntg）：Windows 及 Macintosh 均可导入。
- Photoshop（.psd）：Windows 及 Macintosh 均可导入。
- PICT（.pic，.pct）：Windows 可以导入但 Macintosh 无法导入。在 Windows 环境中以位图形式存在。
- QuickTime Image（.qtif）：Windows 及 Macintosh 均可导入。
- SiliconGraphics（.sai）：Windows 及 Macintosh 均可导入。
- TGA：（.tgf）Windows 及 Macintosh 均可导入。
- TIFF：（.tiff）Windows 及 Macintosh 均可导入。

9.2.2 导入图片及图片组

1. 导入单张图片

- 在编辑 Flash MX 动画时，选择 File | Import 菜单命令，会弹出如图 9-2 所示的导入对话框。
- 选中要导入的图片，单击打开按钮。图片随即被导入至 Flash MX 中。

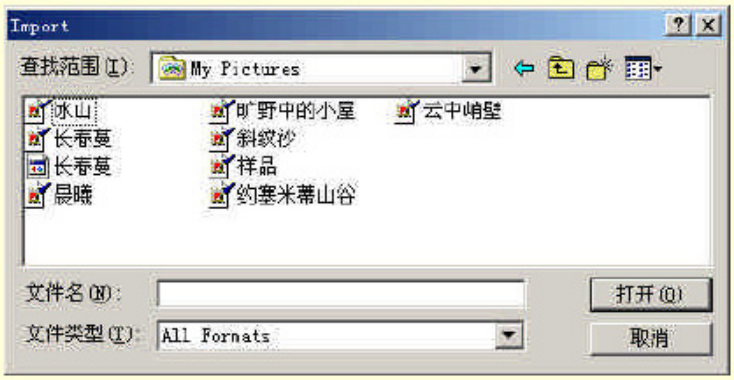


图 9-2　导入图片对话框

2．导入图片组

图片组是指一组按顺序命名的图片，如 f1.jpg，f2.jpg，f3.jpg，f4.jpg……f*n*.jpg。

导入图片组多用于将连续变化的图片导入到 Flash MX 中制成动画。这些连续变化的图片可以是扫描的电影胶片、手绘图片等。

当从一组图片中导入其中一张到 Flash 中时会弹出如图 9-3 所示的对话框。对话框询问是否将这张图片所在图片组中的所有图片全部导入，选择“是”，Flash MX 会将整个图片组导入。

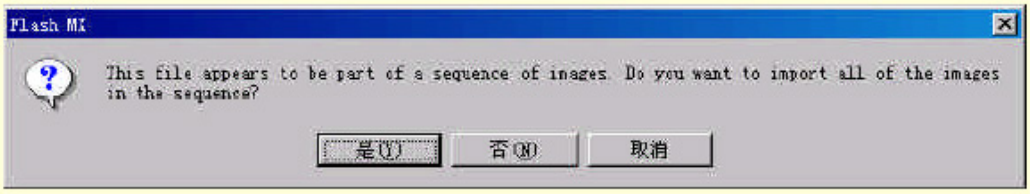


图 9-3　导入图片组提示

如图 9-4 所示，导入的图片在时间轴中按顺序排列成若干帧，不难理解，图片组中的图片数目等于时间轴中排列的帧数。

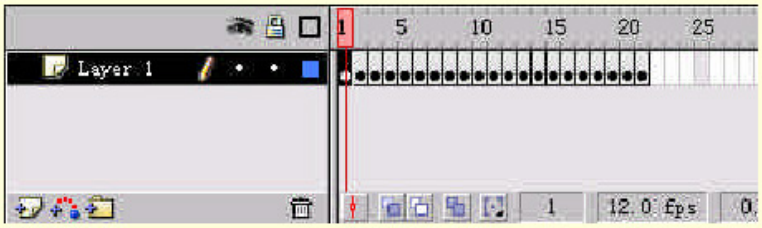


图 9-4　导入图片组

✎ 导入图片组方式与在导入对话框中同时选择多个图片一起导入的方式是完全不同的。前者，将导入的图片分别放在连续的不同的关键帧之中，而后者导入的多张图片均放在同一帧中。

如图 9-5 所示，在 Flash MX 的库中可以看到导入的图片按图片名顺序排列。

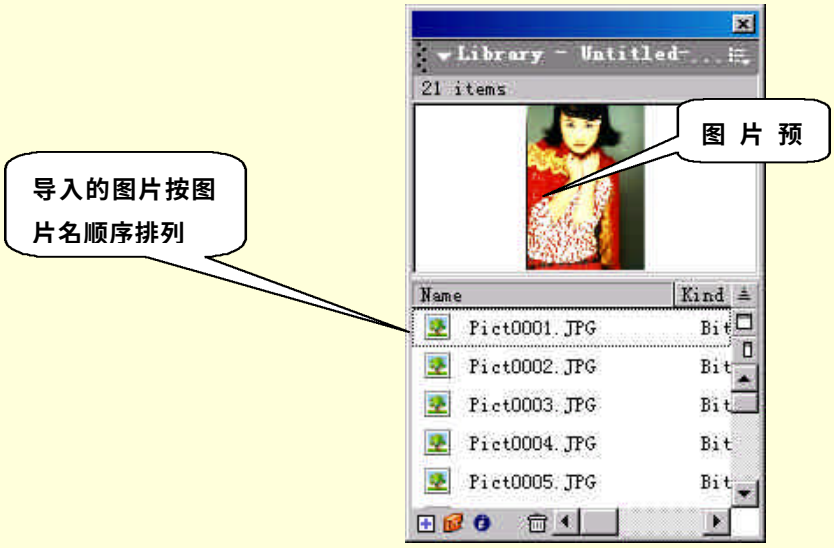


图 9-5　导入的图片

9.2.3　把位图转换为矢量图

用 Flash 的位图分析命令可以将位图转换成矢量图以便对文件进行其他的操作。位图转换成矢量图后，矢量图与库窗口中的位图 Symbol（元件）不存在连接关系。

如果导入的位图所包含的图形过于复杂，那么转换后的矢量图可能比原来的位图还大，需要更多的存储空间。此时可以在位图分析对话框中重新设置各项参数，在图像质量及文件大小之间寻求平衡。

转换步骤如下：

（1）选择需要进行转换的位图。

（2）选择 Modify | Trace Bitmap 菜单，会弹出如图 9-6 所示的位图分析对话框。

（3）在 Color（颜色）文本框中键入颜色容差值（1~500）。

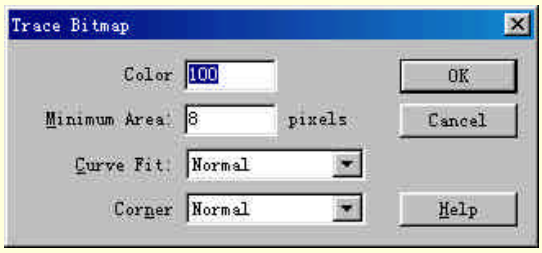


图 9-6　位图分析对话框

- 当两个像素点进行比较时，如果它们的 RGB 色彩差值小于容差值时，这两个像素点将在转换后归于同一颜色。这意味着：加大容差将减少颜色数目，相应的文件也将减小，但图的质量会下降。可以根据需要在两者之间寻求一种平衡。
- 在 Minimum Area（最小区域）选项中填入 1~1000 范围内的一个数值。以确定在进行转换时归于同一颜色的区域所包含像素点的最小值。
- 在 Curve Fit（曲线适应）列表框中选择合适的选项以确定转换时轮廓曲线的光滑程度。列表框中有六个选项：Pixels（像素）、Very Tight（非常接近）、Tight（接近）、Normal（标准）、Smooth（光滑）、Very Smooth（非常光滑）。
- 在 Corner（边角界限）列表框中选择合适的选项。以此确定在转化过程中对边角所采取的处理办法。列表框中有 3 个选项：Many corners（较多转角）、Normal（标准）、Few Corners（较少转角）。

为使转化后的矢量图与原来的位图差别较小，常采取以下设置：

- 颜色容差：10。
- 最小区域：一个像素。
- 边角界限：较多转角。
- 曲线适应：像素。

转化效果如图 9-7 和图 9-8 所示。

图 9-7　转化之前

图 9-8　转化之后

9.2.4　位图填充

分解位图：将位图分解可以将组成位图的像素点分离，位图上各区域间原有的组合形式也被打破。被分解的位图可以用 Flash MX 的绘图工具对其进行修改，修改时将修改对象作为填充区域处理。

用套索工具及其魔棒可以实现对分解位图的选择。在用位图对某一区域进行填充之前可以用油漆桶工具对已分解的位图进行旋转、错切、缩放变换。

分解位图的步骤：

（1）选中将要分解的位图。

（2）选择 Modify | Break Apart 菜单命令将位图分解。

用位图填充：

（1）按上述方法分解位图。

（2）在工具箱中选择点滴器工具，然后在位图上单击左键，点滴器会将此位图图像设置成当前的填充内容。点滴器自动变成油漆桶工具。

（3）进行填充操作，会得到如图 9-9 所示的填充效果。

图 9-9　填充前后

对分解位图的选中区域进行修改的步骤如下：

（1）选中已被分解的位图。

（2）选择套索工具及其魔棒模式。

（3）设置魔棒模式的各项参数。

（4）单击位图中需要修改的区域，符合设置参数的相连区域将被选中。在其他地方单击鼠标可以增加所选区域。

（5）设置填充色，并用油漆桶工具对选中区域进行填充。填充前后的效果对比如图 9-10 所示。

图 9-10　填充前后的效果对比

9.2.5　编辑位图

用 Fireworks 3 或更高版本及其他图形处理软件可以对导入 Flash MX 中的位图进行处理。在 Flash 工作环境中可以直接装载这些软件处理位图。

用 Fireworks 编辑位图：

（1）在库窗口中右击需要编辑的位图图标。在弹出的菜单中选择 Edit with Fireworks 命令。

（2）在 Fireworks 中对位图进行编辑。

（3）编辑完成后右击库窗口中的位图元件，选择 Update（更新）菜单项，会弹出如图 9-11 所示的对话框，选中更新的位图文件，单击 Close（关闭）按钮，Flash 中的位图会自动更新成编辑后的效果。

用其他外部软件编辑位图：

（1）在库窗口中右击需要编辑的位图图标。在弹出的菜单中选择 Edit With 命令，会弹出如图 9-12 所示的 Select External Editor（选择外部编辑器）对话框，选择需要的图形编辑软件，单击“ 打开 ”按钮。

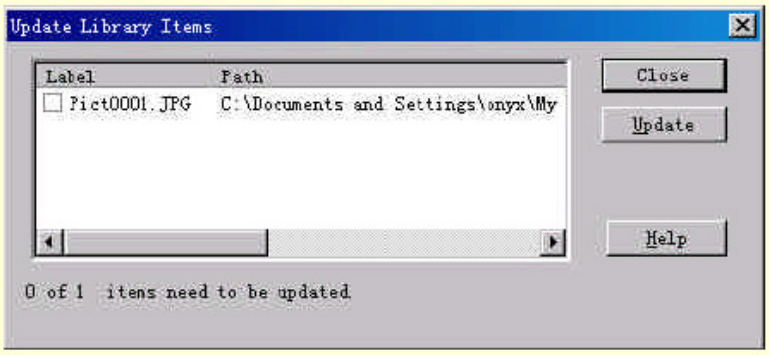


图 9-11　更新对话框

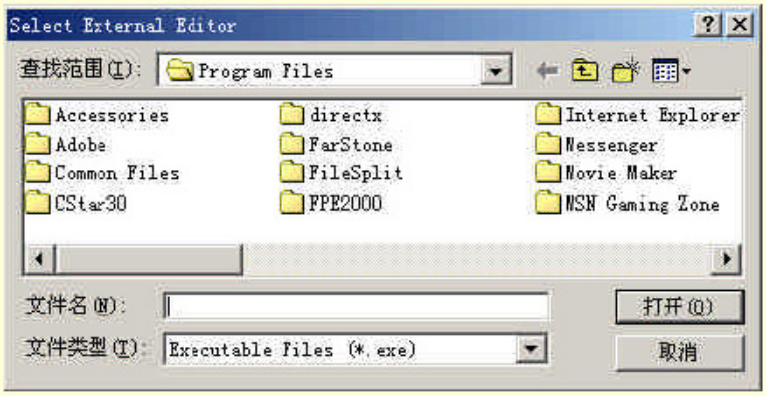


图 9-12　选择外部编辑器对话框

（2）在所选的软件环境中对位图进行编辑。

（3）编辑完成后右击库窗口中的位图元件，选择 Update（更新）菜单项。Flash 中的位图会自动更

新成编辑以后的样子。

9.2.6 设置位图格式

在位图属性对话框中可以对位图参数进行设置，以满足在网页上的发布要求，如文件大小及图像质量等。

设置属性的步骤如下：

（1）在库窗口中右击需要编辑的位图图标。在弹出的菜单中选择 Properties（属性），会弹出如图 9-13 所示的属性设置对话框。

（2）在对话框中选择 Allow Smoothing（允许平滑）。此项用于平滑位图边缘。

（3）在 Compression（压缩）下拉式列表框中有如下选择：

- Photo JPEG（JPEG 图片）：以 JPEG 格式压缩图形。选中 Use imported JPEG data（使用缺省图形质量）后无需设置图形质量参数；如不选择缺省设置，可以在 Quality（质量参数）文本框中填入合适的参数进行设定。
- Lossless PNG/GIF（无损图片）：因为 JPEG 是有损压缩格式。而选择此选项，在压缩过程中可以保持原有图片的质量，即原有图片的数据不被破坏。

对于含有渐变色或复杂颜色、色调的复杂图形应该采用 JPEG 图片模式压缩；对于较为简单的图形应该采用无损压缩。

（4）单击 Test（检测）按钮可以对照原始图片查看压缩结果，根据检测结果决定压缩参数设定的可行性。

（5）单击 OK 按钮完成设定。

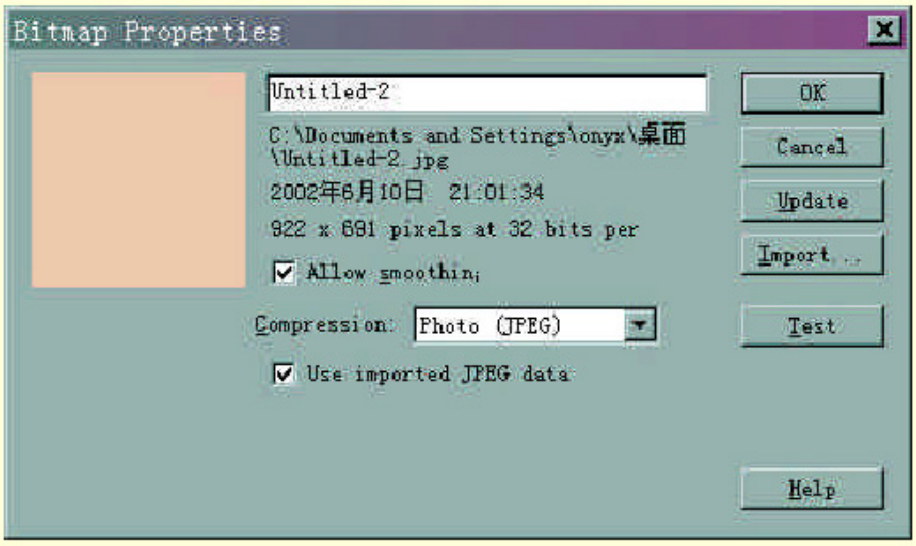


图 9-13 属性设置对话框

9.3 导入声音

Flash MX 提供了多种声音导入方法。它可以让声音持续播放而不受 Timeline（时间轴）的控制，也可以让声音与动画保持同步。声音与按钮事件可以交互，可以让声音产生逐渐变大、变小的效果。

共享库中的声音文件可以链接到动画中。用 Sound（声音）对象也可以调用声音文件，然后用 ActionScript 脚本控制声音的播放。

在 Flash MX 中可以使用两种声音：Event sounds（事件声音）及 Stream sounds（流式声音）。事件

声音需要完全下载才能播放，直到被明确终止才能停止播放；流式声音在最初几帧的数据下载完毕后就可播放，即流式声音可以边下载边播放，网页中的流式声音在播放时与时间轴保持同步。

在导出动画时使用压缩选项可以控制声音的质量及文件的大小。在声音属性对话框中可以对单个声音属性进行设置，也可以在发布设置对话框中进行修改。

9.3.1 Flash MX 所支持的声音文件

在 Flash MX 中可以导入多种格式的声音文件，包括 WAV、MP3 等。

如果系统中装有 QuickTime 4 或其更高版本，还可以导入更多格式的声音文件。类似于位图文件的导入，不同的操作系统允许导入的声音文件的格式有所差别。列举如下：

- AIFF：Windows 及 Macintosh 均可导入。
- Sound Designer II：Macintosh 可以导入但 Windows 无法导入。
- Sound Only QuickTime Movies：Windows 及 Macintosh 均可导入。
- Sun AU：Windows 及 Macintosh 均可导入。
- System 7 Sounds：Macintosh 可以导入但 Windows 无法导入。
- WAV：Windows 及 Macintosh 均可导入。

与导入的位图文件一样，Flash MX 将导入的声音文件也存在库中，如果需要在 Flash MX 动画中共享声音，可以将声音文件加到共享库中。

如果要在 Flash MX 动画中加入音效，最好导入 16 位的声音文件；倘若内存有限，可以导入 8 位的声音文件。

声音文件的导入方法与位图文件的导入方法相同，读者可以参考 9.2.2 节中的内容，这里不再赘述。

9.3.2 为动画配音

为动画配音时可以在声音面板中为声音分配一个新的层。

给动画配音的步骤如下：

（1）导入声音文件。

（2）选择 Insert | Layer 菜单命令，为声音创建一个新图层。

（3）选中所创建的图层，从库窗口中将声音图标拖到舞台中，这样声音文件就被添加到当前层中。

> 一个层中可以放置多个声音，声音与其他对象也可以放在同一层中。建议一个声音对象使用一个层，这样便于管理。当播放动画时，所有层中的声音都将被合并播放。

（4）单击声音文件所在层的任一帧，在 Properties 面板中会出现声音设置选项，如图 9-14 所示。

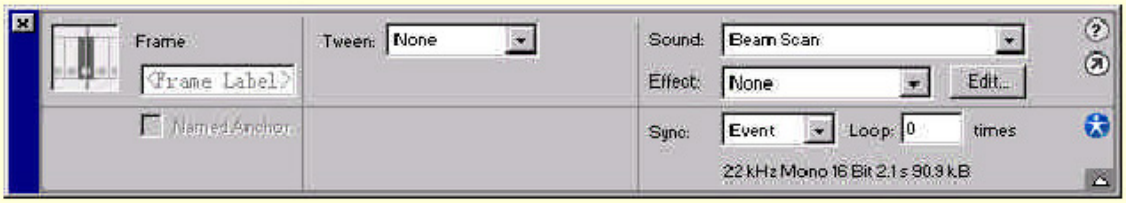


图 9-14 Properties 面板中的声音设置选项

（5）在声音列表框中选择需要设置的声音。

（6）在 Effect（效果）列表框中选择音响效果：

- None（没有）：不对声音进行任何设置。
- Left Channel（左声道）：只在左声道播放。
- Right Channel（右声道）：只在右声道播放。
- Fade Left to Right（从左向右过渡）：控制声音在播放时从左声道向右声道渐变。
- Fade Right to Left（从右向左过渡）：控制声音在播放时从右声道向左声道渐变。

- Fade In（逐渐开始）：控制声音在播放时音量不断增大。
- Fade Out（逐渐关闭）：控制声音在播放时音量不断减小。
- Custom（自定义）：可以自行编辑声音的变化效果，编辑方法将在后续章节中介绍。

（7）在 Sync（同步）列表框中选择所需的同步模式。

- Event（事件）模式：声音的播放同事件的发生同步。事件声音在其开始关键帧出现时开始播放，并且播放整个声音文件，即使动画停止声音的播放也照常进行，不受时间轴的控制。
- Start（开始）模式：与事件模式类似。区别在于开始模式不包括如下情况——当一个声音在播放时，另一个声音事件发生。
- Stop（停止）模式：使当前指定的声音停止播放。
- Stream（流式声音）模式：在网页中动画与声音同步，Flash MX 会强制控制动画的播放与流式声音的播放同步。Flash MX 如果无法快速播放动画帧以实现同步，有些帧将被忽略以保持同步。事件声音的播放不可能超过它所占据的帧的长度。

✍ 如果用 MP3 格式的声音文件作为流式声音，必须在导出之前对声音进行进一步压缩。

（8）选择 Loop（循环）次数：指定声音播放的循环次数。如果需要声音持续播放较长时间，在文本框中填入较大的数值即可。例如想让一个 15 秒长的声音播放 15 分钟，在文本框中填入 60 即可。

✍ 建议不使用流式声音的循环。因为在使用流式声音循环时动画帧数将随着循环次数的增加而增加。

9.3.3 让按钮发音

声音可以与按钮元件链接生效。因为声音同其他元件一同存储，可以与所有元件的事件进行链接。

让按钮发音的步骤如下：

（1）在库窗口中选择一个按钮。

（2）选择库窗口选项中的 Edit（编辑）命令。

（3）在按钮的时间轴中为声音创建一个新层。

（4）在声音层中，插入一个空白关键帧，使声音与需要加入声音的按钮相符合。例如：希望在按钮按下的同时播放声音，可以在按钮按下的那一帧插入一个关键帧。

（5）单击刚刚创建的关键帧。

（6）选择 Window | Properties 菜单命令。

（7）在弹出的声音面板中选中声音文件。

（8）在同步列表框中选择 Event（事件）模式。

9.3.4 声音的编辑与控制

用 Flash MX 的声音编辑控制功能可以定义声音的起始点、终止点及播放时的音量大小。这一功能可以去除声音中不用的部分以减小声音文件的大小。

编辑声音文件的步骤如下：

（1）为声音文件添加一个帧或选中一个已经包含声音文件的帧。

（2）选择 Window | Properties 菜单命令，在弹出的帧属性面板中单击 Edit 按钮。此时会弹出如图 9-15 所示的声音编辑对话框，对其进行必要的设置。

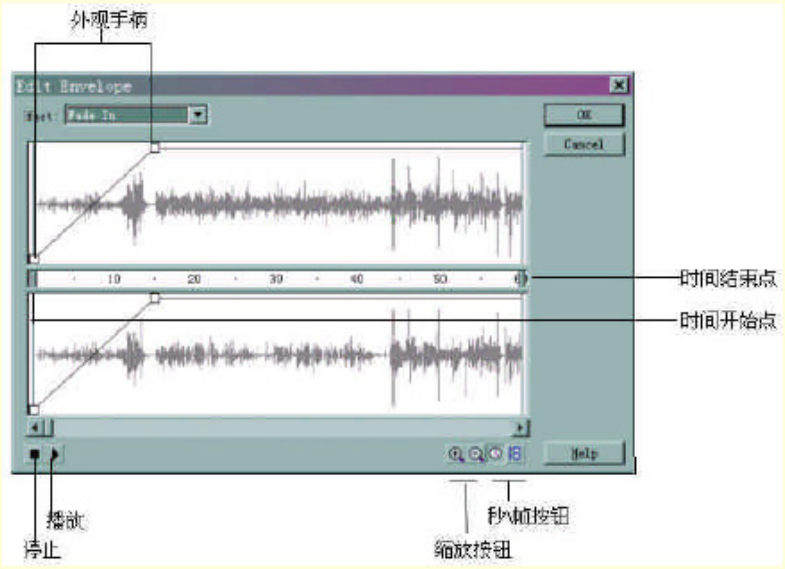


图 9-15　声音编辑对话框

- 拖动 Time In（时间开始点）及 Time Out（时间结束点）手柄可以改变声音的起始点及终止点。
- 改变声音的外观：拖动 Envelope（外观）手柄可以改变声音在播放时的音量高低。若需要增加外观手柄（最多 8 个），只需在外观控制线上单击左键；如果将手柄拖出窗口即可将其删除。
- 用 Zoom In/Out（缩放）按钮可以使窗口中的声音波形图样以缩小或放大模式显示。
- Seconds/Frames（秒/帧）按钮可以转换窗口中央的标尺，使其按秒数或者帧数来度量声音波形图样。

9.3.5　声音格式的设定

在声音属性对话框中可以对声音文件的压缩参数进行设置。对应不同的压缩方法有不同的参数选项。如果声音文件已在外部软件中编辑，可以用声音属性对话框的 Update（更新）功能将编辑后的声音文件更新到 Flash 中；用 Test（检测）功能还可以测试压缩结果，Flash 会在检测时播放压缩后的声音以供参考。

不同的 Sampling rate（采样频率）和压缩比会产生不同的压缩效果，体现在动画中的声音文件的大小及音质上有明显差别。对于一个声音文件来说，采样频率越低、压缩率越高意味着文件越小、音质越差。在创作时应当在文件大小与音质两者之间取得一种平衡，在必要时可以忽略其中一方面。

MP3 格式的声音文件在导入时已经被压缩过了，但仍可以根据需要对其进行再压缩。例如：将 MP3 音乐用于 Stream sound（流式声音），必须对其进行再压缩，因为流式声音必须在压缩后才能导出。

如果没有对声音的导出进行设置，Flash MX 会自动启用 Publish Settings（发布设置）对话框中的相关选项、参数作为声音文件的导出设置。用户也可忽略声音属性对话框中的各项设置，方法是在导出设置对话框中选取 Flash 标签，并将此标签中的忽略 Override Sound Settings（声音设置）复选框选中。此选项在需要制作较大的高保真度的音频动画或较小的低保真度网页动画时比较有用。

设置声音属性的步骤如下：

（1）先执行下列操作之一，会弹出如图 9-16 所示的声音属性对话框：

- 双击库窗口中的声音图标。
- 右击库窗口中的声音图标，在弹出的菜单中选择 Properties（属性）。
- 在库窗口中选中声音图标，在库窗口的选项 Options（菜单）中选择属性。
- 选中库窗口中的声音图标，单击库窗口中的属性按钮。

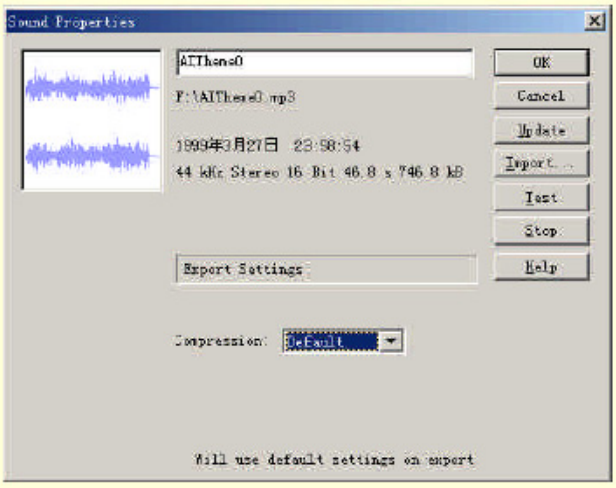


图 9-16 声音属性对话框

（2）如果声音文件已在外部软件中进行了编辑，单击 Update（更新）按钮。

（3）在 Compression（压缩）列表框中选择压缩格式，可供选择的压缩格式有 Default、ADPCM、MP3、Raw 及 Speech。下面将分别对这几种格式的设定方法进行说明。

- Default（默认格式）：选择如图所示的默认格式，可以看到属性对话框中无任何可选设置，在导出动画时用发布设置对话框中的相关选项、参数作为声音文件的最终设置。
- ADPCM 格式：选择 ADPCM 格式会出现如图 9-17 所示的导出设置选项。选中 Preprocessing（预处理）复选框可以将混合立体声转换成单声道，如果声音已经是单声道的了，那么此选项将不起作用。在 Sample Rate（采样频率）中选择合适的频率以控制音质及声音文件的大小。在 ADPCM Bits 下拉列表框中选择声音单元的存储位数，可以选择用 2bit、3bit、4bit 或 5bit 存储一个声音单元。当然，存储一个声音单元所用的比特数越多，音质越好，整个声音文件也就越大。

在压缩过程中，对不同用处的声音采取合适的采样频率可以收到较好的效果。

5 kHz：音质最差，勉强可以在对话声音中使用。

11 kHz：CD 音质的四分之一，可以用于较短音乐中的最差音质。

22 kHz：CD 音质的二分之一，对于用在网页中的音乐来说是较为折中的选择。

44 kHz：标准 CD 音质，存储时需要较大的文件。

Flash MX 无法提升（也不可能提升）导入声音文件的采样频率，如导入的声音文件为 22kHz，无法通过上述的设置将其采样频率提升至 44 kHz。

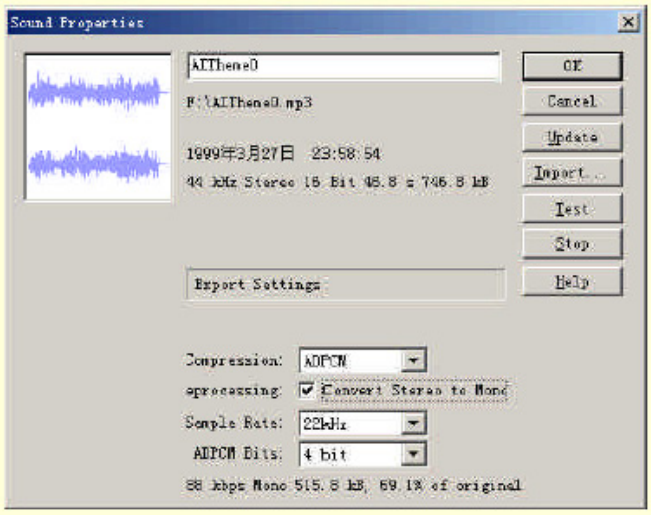


图 9-17 导出 ADPCM 格式的设置选项

- MP3 格式：选择 MP3 格式会出现如图 9-18 所示的导出设置选项。在 Bit Rate 中定义由 MP3 译码器生成的声音文件的最大位率。Flash MX 支持的位率从 8~160kbit/s 不等。在导出音乐时选择 16kbit/s 或更高位率以获得较好的效果。预处理的作用与 ADPCM 格式相同。在 Quality（质量）列表框中有快、中、慢 3 个选项，Fast（快速）可以使压缩速度加快，但会降低声音的质量；Medium（中速）压缩较慢，但音质较好；Slow（低速）压缩最慢，但音质最好。

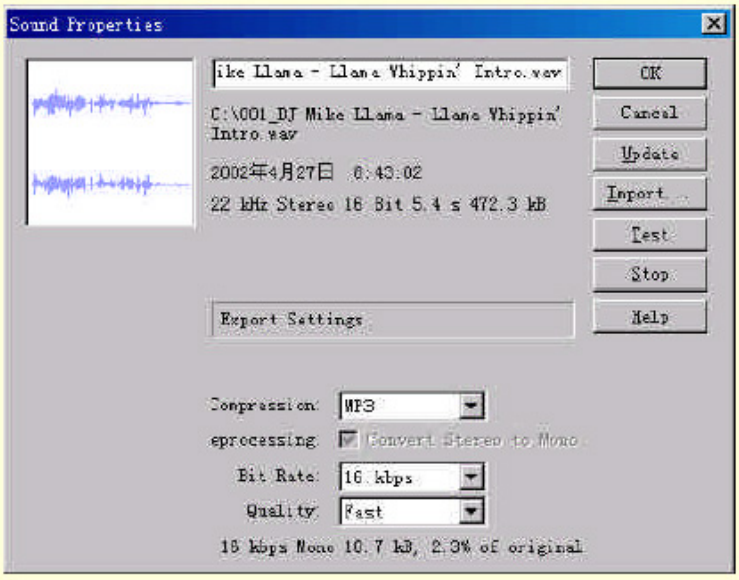


图 9-18 导出 MP3 格式的设置选项

预处理选项只有在速率等于或高于 20kbit/s 才可选。

如果选择 Use imported MP3 quality（使用导入的 MP3 文件质量），那么速率及质量列表框将无法进行设置。而且在输出动画时，Flash MX 也不对声音做特殊的处理，可以节省大量的输出动画的时间。

- Raw 格式：选择 Raw 格式会出现如图 9-19 所示的导出设置选项。Sample Rate（采样频率）及 Preprocessing（预处理）的设置方法与上述方法类似，读者可以参考 ADPCM 格式关于这两项的设置方法。
- Speech 格式：选择 Speech 格式会出现如图 9-20 所示的导出设置选项。Sample Rate（采样频率）及 Preprocessing（预处理）的设置方法与上述方法类似，读者可以参考 ADPCM 格式关于这两项的设置方法。

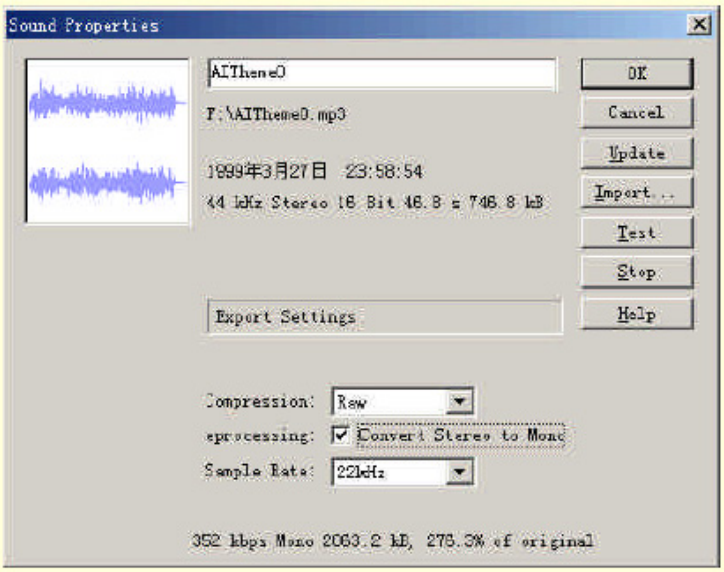


图 9-19 导出 Raw 格式的设置选项

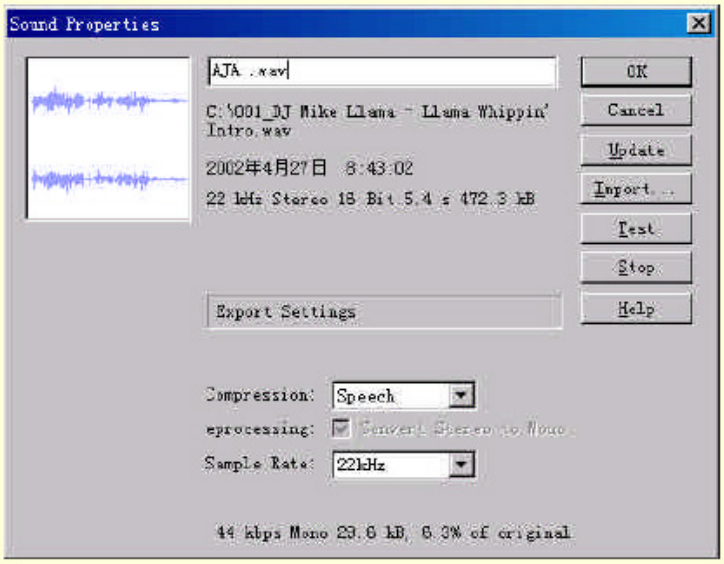


图 9-20 导出 Speech 格式的设置选项

（4）单击 Test（检测）按钮，Flash 会按照设置对声音进行处理，从如图 9-21 所示的导出进度条上可以直观地看到处理进程。处理完成后，Flash MX 会自动播放处理后的声音，单击 Stop（停止）按钮可以停止播放，声音停止后可以根据需要调整设置。

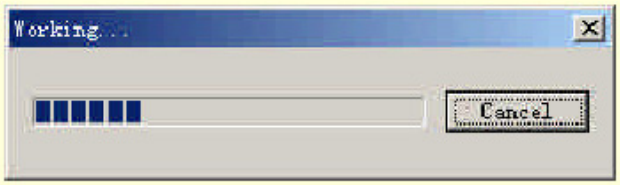


图 9-21 导出进度条

（5）设置完成后单击 OK 按钮，关闭声音属性设置对话框。

9.3.6 动态调节音量

使用 Flash MX 的 ActionScript 脚本中的 Sound 对象可以控制动画中声音的播放。Sound 对象的定义有如下两种方式：

```
new Sound();
new Sound(target);
```

其中，target 参数为一个影片剪辑，由 new Sound(target)所返回的声音对象变量可以控制 target 参数所指向的影片剪辑的声音。如果不指定 target 参数，那么，将得到指向主场景中所有声音对象的变量。

Sound 对象包括如下方法：

- Play()：开始播放声音
- Stop()：停止播放声音
- SetPan(pan)：设置声音中左右声道音量的均衡量。参数 pan 的取值范围是-100~100。当 pan=-100 时，只使用左声道；当 pan=100 时，只使用右声道；当 pan=0 时，左右声道的音量均衡。
- SetTransForm(soundTransformObject)：为声音对象设置声音的变化信息。参数 soundTransformObject 包括以下 4 各属性。
  - ll：左声道在左音箱中的输出百分比，范围-100~100。
  - lr：右声道在左音箱中的输出百分比，范围-100~100。
  - rr：右声道在右音箱中的输出百分比，范围-100~100。
  - rl：左声道在右音箱中的输出百分比，范围-100~100。

可见，最终在左右音箱中的输出音量可以按照下式计算出来：

左音箱中的输出 = 左声道 x ll + 右声道 x lr

右音箱中的输出 = 右声道 x rr + 左声道 x rl

对于立体声，因为左右的声道是分开的，所以，默认的设置为：

ll = 100; lr = 0;

rr = 100; rl = 0;

对于非立体声，默认的设置为：

ll = 100; lr = 100;

rr = 0; rl = 0;

- SetVolume(volume)：设置音量的百分比，volume 的范围 0~100。
- GetPan()：返回声音中左右声道音量的均衡量。参见 SetPan()。
- GetTransForm()：返回声音对象设置的变化信息。参见 SetTransForm()。
- GetVolume()：返回音量的百分比。参见 SetVolume()。

下面的例子给出了在 Flash MX 中控制声音的具体方法：

（1）新建一个 Flash MX 文件。随意导入一段声音如 music1.mp3。在主场景的第一帧上设置帧属性面板，设置的参数如图 9-22 所示。

图 9-22 设置帧属性面板

（2）在主场景的第一帧中添加静态及动态文本区域，并按图 9-23 所示方式进行命名。

（3）在如图 9-24 所示的位置添加第 8 章中所做的滑动条组件，由上至下分别命名为：volume_bar、pan_bar、ll_bar、lr_bar、rr_bar、rl_bar。

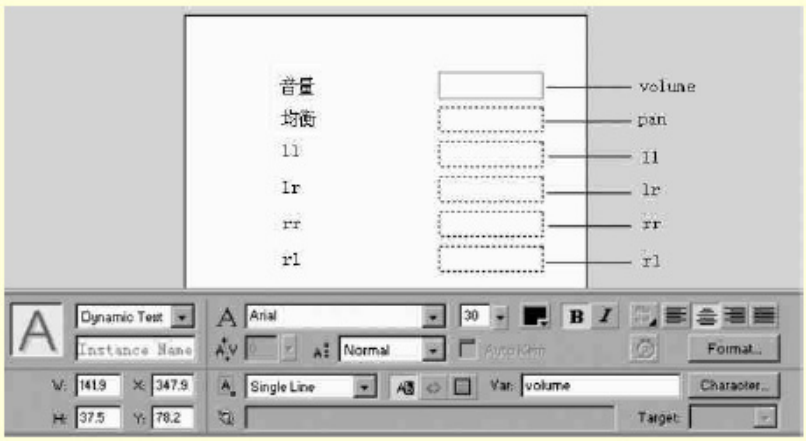


图 9-23 添加文本区域

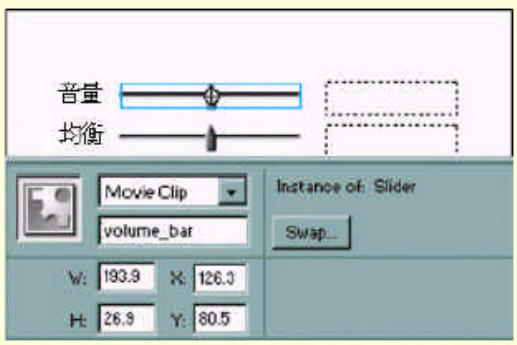


图 9-24 添加滑动条

（4）如表 9-1 所示设置各滑动条的参数。

表 9-1 滑动条的参数

滑动条	Sliding	Position	Max Range	Min Range
volume_bar	false	100	200	0
pan_bar	false	0	100	-100
ll_bar	false	100	100	-100
lr_bar	false	0	100	-100
rr_bar	false	100	100	-100
rl_bar	false	0	100	-100

虽然 Flash MX 要求 Sound.SetVolume(volume)中的参数 volume 的范围在 0~100 之间，可是，如果 volume 的值超过 100，声音也同样会继续增大，但是将产生失真。所以，并不提倡这样做。

（5）在如图 9-25 所示的位置上添加按钮。

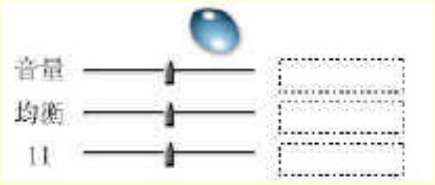


图 9-25 添加按钮

（6）为主场景的第 1 帧添加如下脚本：

```
s = new Sound();
mySoundTransform = oldSoundTransform=s.getTransform();
volume=oldvolume=s.getVolume();
volume=oldpan=s.getPan();
ini_text();
function ini_text () {
    volume = oldvolume;
    pan = oldpan;
    ll = oldSoundTransform.ll;
    lr = oldSoundTransform.lr;
    rr = oldSoundTransform.rr;
    rl = oldSoundTransform.rl;
    Updatabar();
}
function Updatabar () {
    volume_bar.setpos(volume);
    pan_bar.setpos(pan);

    ll_bar.setpos(ll);
    lr_bar.setpos(lr);
    rr_bar.setpos(rr);
```

```
        rl_bar.setpos(rl);
    }
    function setSound () {
        Updatabar();
        mySoundTransform = {ll:ll, lr:lr, rr:rr, rl:rl};
        s.setVolume(volume);
        s.setPan(pan);
        s.setTransform(mySoundTransform);
    }
```

（7）为每一个滑动条添加如下脚本：

```
    onClipEvent (mouseMove) {
        if (bsetpos==false) {
            return;
        }
        _root.volume = pos;//对于不同的滑动条只需改变为设置相应的变量
        _root.setSound();
    }
```

（8）为按钮添加如下脚本：

```
    on (release) {
        s.setTransform(oldSoundTransform);
        ini_text();
    }
```

（9）输出并运行动画，如图 9-26 所示，调整滑动条可以改变声音的设置，从而得到不同的效果，并且使用按钮可以还原设置。

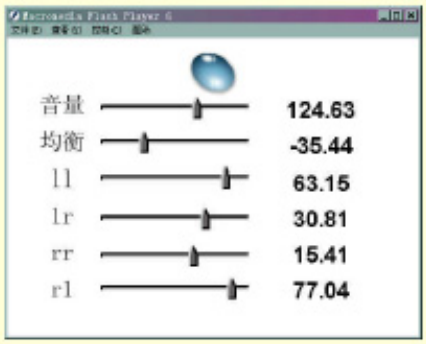


图 9-26　运行结果

第三部分

Flash MX 应用篇

第 10 章　Flash MX 动画的发布

第 11 章　Flash MX 与网络

第 12 章　Flash MX 动画实战

第 10 章　Flash MX 动画的发布

10.1　综　述

当动画制作完成后，需要将 FLA 格式的文件发布（或者叫“导出”）成 SWF 格式的文件（扩展名为 SWF，可以直接被 Flash 6 播放器播放的动画文件）用于网页播放。Flash 在发布时可以生成 SWF 文件及 HTML 文件（超文本文件），超文本文件允许在 IE 浏览器窗口中观看动画，如图 10-1 所示。

图 10-1　在 IE 浏览器中观看动画的播放

Flash MX 可以发布多种格式的动画文件，用户可以根据需要自行定义发布文件的格式。选择 File | Publish Settings 菜单命令会弹出如图 10-2 所示的发布设置对话框。在格式 Formats（标签）中可以选择发布哪些格式的动画。可供选择的发布格式很多，有 SWF、HTML、GIF、JPEG、PNG、Windows 可执行文件、Macintosh 可执行文件、QuickTime 动画文件。如果所选中的格式有额外的设置，在发布设置对话框中会新增一个标签，关于这些标签的使用将在以后进行介绍。如果使用默认的文件名，选中 Use default names（使用默认名）复选框即可，Flash MX 会自动为发布文件命名；如果想自行定义文件名，取消选择上述复选框，在 Filename（文件名）文本框中填入文件名即可。

✎ 不同格式文件的扩展名也不同，在自行定义文件名时注意不要修改扩展名。如果不小心改动了扩展名而又忘了正确的扩展名，可以先选中 Use default names 复选框，再取消选择，文件名会变为默认名（扩展名会变成正确的扩展名），然后修改文件名即可。

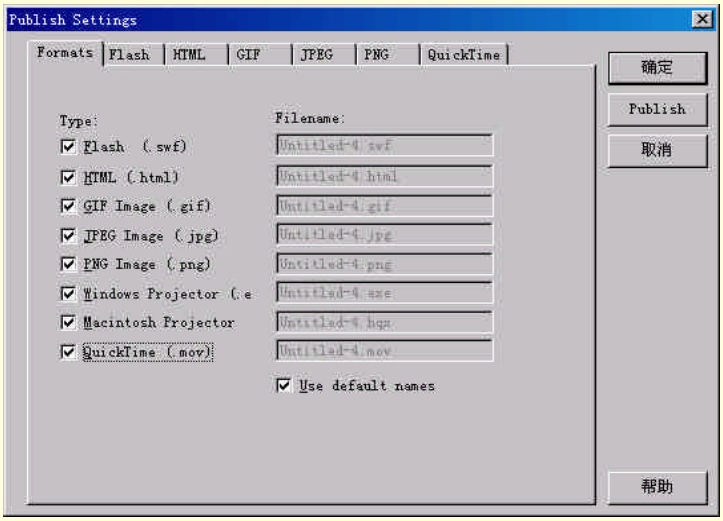


图 10-2 格式标签

完成设定后单击“确定”按钮，然后选择 File | Publish 菜单命令；或者直接使用 Publish（发布）按钮，Flash MX 会将动画文件发布到源文件所在的文件夹中。如果在更改文件名时设定存储路径，Flash MX 会将动画文件发布到路径所指向的文件夹中。如将 HTML 文件的路径设置为“F:\tmp\Mov4.html”，那么发布后所生成的 HTML 文件会存储在 F 盘的 tmp 文件夹中。

10.2 输出动画的格式

10.2.1 输出 SWF 动画

SWF 动画格式是 Flash MX 自身的动画格式，因此它是输出动画的默认形式。在输出 SWF 动画时，通过选择如图 10-3 所示的 Flash 标签，可以设定 SWF 动画的图像和声音压缩比例等参数。

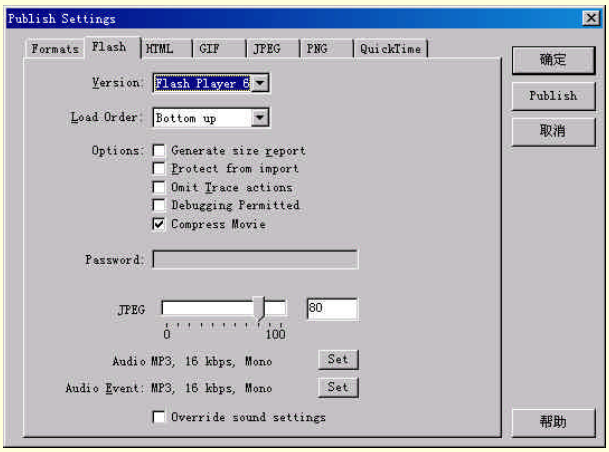


图 10-3 Flash 标签

- 选择 Version（版本）：选择所输出 Flash 动画的版本，范围从 Flash 1~Flash 6。因为 Flash MX 动画的播放是靠插件支持的。如果用户的系统中没有装高版本的 Flash 插件，那么用高版本输出的 Flash 动画在此系统中不会被正确显示出来。如果使用低版本的格式输出 Flash MX 的动画，所有 Flash MX 新增的特性或者新动作将无法正常运行。所以，除非有必要，否则不提倡使用低版本的格式输出 Flash MX 的动画。
- Load Order（装载顺序）：当动画被读入时所装载的每一层的先后顺序。当 Flash MX 动画被远程调用时，尤其是在网络传输速率较低的时候，设定 Flash MX 动画的装载顺序就显得尤为重要，因为它决定了在动画的主场景中哪一层先显示出来。但是当网络传输速率高时，或者在本地机

上欣赏 Flash MX 动画时，用户根本感觉不到动画绘制的先后顺序。此选项包括如下两种选择：

- Bottom up（底层到上层）：先载入并且绘制 Flash MX 动画的最下层，再逐渐地载入并绘制上面的层。
- Top down（上层到底层）：先载入并且绘制 Flash MX 动画的最上层，再逐渐地载入并绘制下面的层。

- Option（选项）
  - Generate size report（生成数据报告）：生成 Flash MX 动画运行的过程中传输数据的报告文件（.txt）。关于此选项将在本章下几节中予以说明。
  - Omit Trace actions（忽略 Trace 动作）：在调试 Flash MX 动画的 ActionScript 脚本时经常需要使用 Trace 动作来显示某一变量的值，或者脚本运行的位置等相关信息。当脚本运行到 Trace 动作的时候将自动弹出 Output（输出）面板，以显示相关的信息。当一个 Flash MX 动画调试完成以后，在发布动画时可以选择此选项，忽略所有的 Trace 动作并禁止 Output（输出）面板的显示。这样可以避免查找并删除 Trace 动作的过程。
  - Protect from Import（导入保护）:防止所生成的动画文件被其他人非法导入到新的 Flash MX 文件中继续编辑。当选中此选项后，对话框中的 Password（密码）文本框便被激活，在其中可以加入导入此动画文件所需要的密码。以后当文件被导入的时候，就会出现如图 10-4 所示的对话框提示输入密码。只有密码正确才允许导入此动画文件。
  - Debugging Permitted（允许跟踪）：允许在 Flash MX 文件外部跟踪动画文件。当选中此选项后，对话框中的 Password（密码）文本框也被激活，在其中可以加入跟踪所需要的密码。

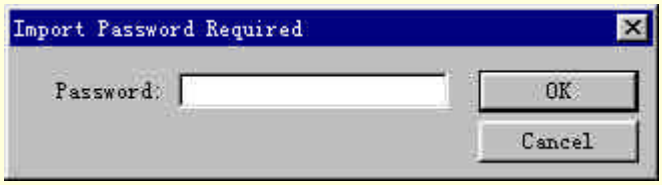


图 10-4 输入导入密码

  - Compress Movies（压缩影片）：只有在 Version 下拉列表框中选中 Flash Player 6 时此选项变为可用，选中此选项可进一步压缩要发布的动画。

- 通过调整 JPEG 滑动条，或者直接在文本框中键入数值，可以设置位图文件在 Flash MX 动画中的 JPEG 压缩比例和画质。因为 Flash MX 动画中的位图文件是以 JPEG 的格式存储的，而 JPEG 是一种有损压缩格式，当在文本框中键入数值越大，JPEG 图像的质量就越高，体积也就越大；相反，数值越小，压缩比例越高，图像质量就越差。所以需要在文件大小及画面质量之间选择一个折衷的方案。
- 设置声音采样率（格式）。如果在 Flash MX 动画编辑的时候没有为动画中的每一个声音设置声音采样率（格式），那么，可以在这里统一设置。单击 Set（设置）按钮将弹出如图 10-5 所示的声音格式设置对话框。此对话框的使用方法与设置单独的声音格式的方法完全相同，这里不再重复。在这里需要注意的是，声音格式的设置分为如下两类：

图 10-5 设置声音格式

- Stream（流式）声音：可以边下载边播放的声音。它适合于播放长时间的声音，如动画的背景音乐等。
- Event（事件）声音：必须等到下载完毕以后才可以播放的声音。它适合于播放短时间的声音。

- Override Sound Settings（忽略声音设置）：忽略在 Flash MX 动画的制作过程中对各种声音的不同设置，使它们统一成在此对话框中设置的格式，这种方式可以很方便地为高速网络或者本地机生成高保真的声音动画，为低速网络生成低质量的声音，以节省带宽。

当一切设定完毕后，单击 OK 按钮保存对输出的设置，或者使用 Publishing（发布）按钮直接发布动画。

Flash MX 把每一个文件的输出设置分别保存在 Flash MX 动画文件（*.Fla）之中，所以，改变一个文件的输出方式不会影响其他文件的输出格式的设置。

10.2.2 输出 GIF 动画

GIF 是一种较为方便的导出 Flash MX 动画的方法。选择如图 10-6 所示的 GIF 标签。

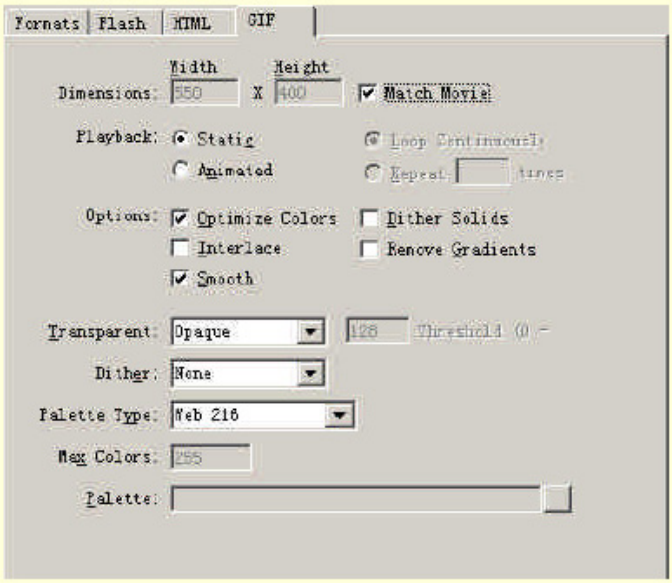


图 10-6 GIF 标签

- Dimensions（尺寸）：设定动画尺寸。
  - Match Screen（匹配影片）：此项为默认选择，导出的动画尺寸与原始尺寸相匹配。
  - 宽与高：当不选用匹配影片选项时，可以对宽与高进行设置，单位为像素。
- Playback（回放）：控制动画的播放。
  - Static（静止）：导出的动画为静止状态。
  - Animated（动画）：选中此项后可以对 Loop Continuously（连续循环）及 Repeat（循环次数）进行修改。选中连续循环可以使动画无限次连续播放；选中循环次数选项并在文本框中填入循环次数，可以让动画循环播放，当循环到设定次数后，动画便停止播放。
- Option（选项）：
  - Optimize Colors（优化颜色）：去除动画中不用的颜色。这样将在不影响画面质量的情况下将文件尺寸减小 1000~1500Byte，但会略微增加对内存的需求。默认情况下此项处于选中状态。
  - Interlace（交错）：交错可以在文件没有完全下载之前显示图片的基本内容。在连接速度较慢时会加快下载速度，但对于 GIF 动画不要使用交错。交错不是默认选择。

- Smooth（平滑）：减少位图的锯齿，使画面质量提高，但平滑会增加文件的大小。这是默认选项。
- Dither Solids（抖动立体）：使纯色产生类似渐变色的抖动效果。
- Remove Gradients（去除渐变色）：使用渐变色中的第一种颜色取代渐变色。为了避免不良结果的出现，要慎重选择渐变色中的第一种颜色。

- Transparent（透明）：用于确定动画背景的透明度。
  - Opaque（不透明）：将背景以纯色方式显示。不透明是默认选择。
  - Transparent（透明）：使背景色透明。
  - Alpha（透明度）：可以对背景透明程度进行设置。在右边的文本框中填入一个数值（0~255），所有色彩指数低于设定值的颜色都将被变得透明，高于设定值的颜色将被部分透明化。例如填入 128 可以产生半透明的效果，即色彩指数低于 128 的颜色变得透明，其余颜色变得半透明。
- Dither（抖动）：确定像素点的合并形式。抖动可以提高画面质量，但会增加动画文件的大小。
  - None（没有）：不对画面进行抖动修改。将非基础色的颜色用近似的纯色代替。这样会减小文件的尺寸，但会使色彩失真，效果如图 10-7 所示。没有抖动是 Flash 的默认设置。
  - Ordered（顺序）：能产生质量较好的抖动效果，与此同时动画文件的大小不会有太大程度的增加。产生的效果如图 10-8 所示。
  - Diffusion（扩散）：可以产生质量最高的抖动效果，但不可避免地增加文件的大小。与顺序抖动相比，扩散抖动会增加处理时间。一般情况下，只有在使用 216 色时才选用此项。产生的效果如图 10-9 所示。

图 10-7 没有抖动

图 10-8 顺序抖动

图 10-9 扩散抖动

- Palette Type（调色板类型）：在列表框中选择一种调色板用于图像的编辑。除了可以选择列举的调色板之外，还可以选择自定义的调色板。
  - Web 216（网络 216 色）：使用标准的 216 色调色板生成 GIF 图像，可以提高图像质量并且使处理速度达到最快。
  - Adaptive（适应值）：Flash MX 会对图像中的颜色进行分析，并为此图像生成一个专用的调色板，生成的调色板中包含了图像中所用到的颜色。这一选项可以为被编辑的图像生成最为精确的颜色，但与使用 216 色生成的图像文件相比，适应值生成的文件较大。减小文件尺寸的办法将在以后介绍。
  - Web Snap Adaptive（捕捉适应 Web 值）：与适应值选项的作用类似，所不同的是，当图像颜色接近 Web 216 调色板中的颜色时，捕捉适应 Web 值功能可以将图像颜色转换为 Web 216 调色板中的颜色。
  - Custom（自定义）：调用自定义调色板。选用此选项的处理速度与使用 Web 216 调色板时的处理速度相同。选中此项后，在 Palette（调色板）文本框中填入调色板的存储路径就可以调用自定义调色板了；也可以单击文本框右边的按钮，在弹出的如图 10-10 所示的对话框中选择调色板文件，打开即可。

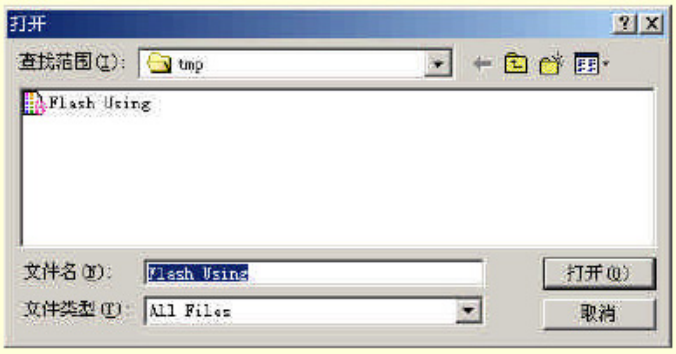


图 10-10　打开自定义调色板

- Max Colors（最大颜色）：如果在面中选择了 Adaptive（适应值）或 Web Snap Adaptive（捕捉适应 Web 值）选项，此文本框将变为可选状态。在其中可以填入 0~255 间的任何数值，可以去除超过这一设定值的颜色。设定的数值较小则可以生成较小的文件，但画面质量会较差。

10.2.3　输出 JPEG 文件

JPEG 格式可以用较高的压缩比存储 24 位的位图图像。一般来说，GIF 格式适用于线的导出，而 JPEG 适用于包含渐变色、嵌入式位图图像的导出。选择 JPEG 标签，如图 10-11 所示，对其中各项进行设置。

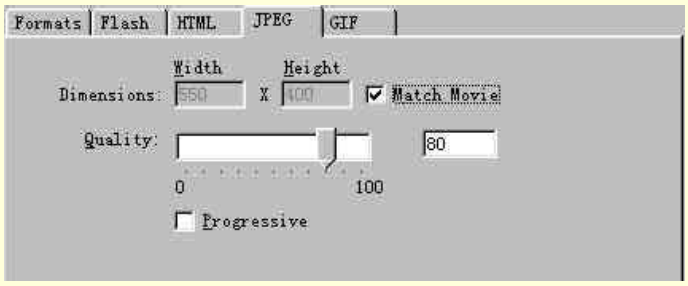


图 10-11　JPEG 标签

- Dimensions（尺寸）：用于设定播放动画的尺寸。
  - Match Movie（匹配影片）：此项为默认选择，导出的动画尺寸与原始尺寸相匹配。
  - Width Height（宽与高）：当不使用匹配影片选项，即取消选择匹配影片复选框时，可以通过设定宽与高的像素点个数来确定播放动画区域的范围。
- Quality（品质）：可以拖动滑块或在右边文本框中填入一定的数值（1~100）以控制 JPEG 文件的压缩程度。滑块与文本框中的数值同步变化。设置品质参数应注意文件的大小与动画质量成反比，为两全其美，可以多次修改参数以获得满意的效果。
- Progressive（渐进显示）：在浏览器中渐进显示图像。如果网络连接速度较慢，这一功能可以加快图片的下载速度。

可以在位图属性对话框中对位图中每个物体的品质进行详细设置。

10.2.4　输出 PNG 文件

PNG 是唯一的一种支持 Alpha（透明度）通道的跨平台位图格式，也是 Macromedia Fireworks 的默认文件格式。在没有进行设定的情况下，Flash 导出动画的第一帧是 PNG 格式。选择如图 10-12 所示的 PNG 标签，对其进行设置。

- Dimensions（尺寸）：用于设定动画的播放尺寸。
  - Match Movie（匹配影片）：作用及用法同上。
  - Width Height（宽与高）：作用及用法同上。

- Bit Depth（位深度）：用于选择像素或颜色的位深度。
  - 8 位：用于 256 色的图像。8 位即 2 的 8 次方。
  - 24 Bit（24 位）：24 位颜色模式包含了成千上万种颜色，是真彩色。
  - 24-bit with Alpha（24 位 Alpha 调色板）：含有 32 位透明度的 24 位真彩色。

✍ 位深度越高，存储动画所需的文件越大。

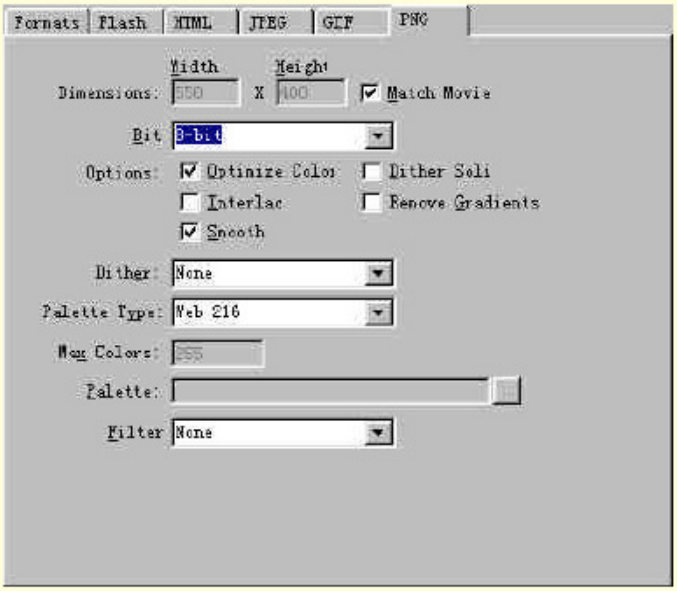


图 10-12　PNG 标签

- Options（选项）：对导出的 PNG 文件进行基本设置。
  - Optimize Colors（优化颜色）：将 PNG 文件中不用的颜色去除。这一功能可以在不影响画面质量的情况下将文件减小 1000~1500Byte，但对内存的要求会略有提高。
  - Interlace（隔行扫描）：使图像在下载时逐渐显示。可以在图像未完全下载之前先显示基本内容，而且在某些情况下可以加快下载速度。注意不要对 PNG 动画使用隔行扫描。
  - Smooth（平滑）：作用与其他标签中的平滑选项相同，用于消除锯齿。
  - Dither Solids（图像抖动）：使渐变色及纯色产生抖动效果。
  - Remove Gradients（移除渐变）：用渐变色中的第一种颜色取代整个渐变色。渐变色的使用会加大 PNG 文件的尺寸且画面质量较低；如果使用此选项，就必须慎重选择渐变色中的第一种颜色。
- Dither option（抖动选项）：确定像素点的合并形式。抖动可以提高画面质量，但会增加动画文件的大小。
  - None（没有）：不对画面进行抖动修改。
  - Ordered（顺序）：能产生质量较好的抖动效果。
  - Diffusion（扩散）：可以产生质量最高的抖动效果，但不可避免地增加文件的大小。

✍ 只有在位深度为 8 位时抖动选项才处于可选状态。

- Palette Type（调色板类型）：列表框中有四种选项，其作用和设定方法与 GIF 标签中的相同，读者可以自行参考。
- Filter（过滤器）选项：读者如果有兴趣，可以参考有关 FireWorks 的书籍，本书不对此项功能作太多说明。

10.2.5　输出 QuickTime 电影

Flash MX 可以输出 QuickTime 4 格式的电影。在 QuickTime 中播放的动画电影与在 Flash 播放器中

的播放效果是相同的，同时保留原有的交互属性。选择如图 10-13 所示的 QuickTime 标签。

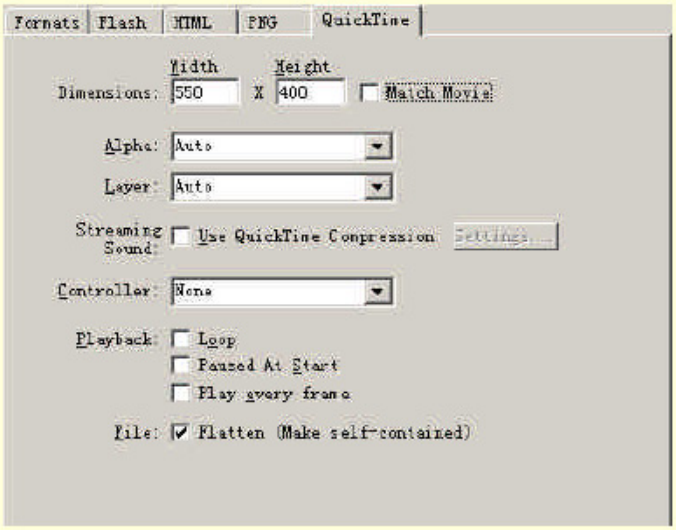


图 10-13　QuickTime 标签

- Dimensions（尺寸）：用于设定动画的播放尺寸，用法同上。
- Alpha 选项：控制 Flash 动画以 QuickTime 电影格式进行播放时的透明度。此选项的设置不会影响 Flash 动画本身的透明度。
- Layer（图层）：
  - Auto（自动）：将 Flash MX 所处的层自动放置。
  - Bottom（底部）：将 Flash MX 所处的层放置在 QuickTime 影片的底部。
  - Top（顶部）：将 Flash MX 所处的层放置在 QuickTime 影片的顶部。
- Streaming Sound（声音流）：将 Flash 动画中的流式音频导出为 QuickTime 音轨。
- Controller（控制器）：确定播放导出动画的 QuickTime 控制器类型。列表框中有三个选项，None（没有）、Standard（标准值）、QuickTimeVR（虚拟 QuickTime）。
- Playback（播放）：此选项用于控制 QuickTime 如何播放电影动画。
  - Loop（循环）：当动画播放到最后一帧时自动跳回第一帧循环播放。
  - Paused at Start（开始时停止）：使动画在播放开始时处于暂停状态，当在菜单中选择播放命令时动画才开始播放。
  - Play Every Frame（播放每一帧）：播放动画的每一帧，不跳过任何一帧。
- File（文件）：选中文件的 Make Self-Contained（单独项目）复选框，这样 Flash 的动画内容将全部合并到一个单独的 QuickTime 电影中。如果不选中此项，QuickTime 文件将不对动画进行合并，只是指明外部文件的存储路径。在使用时按照路径进行调用，如果外部动画文件丢失，动画将不能正常播放。

10.3　动画的优化

动画文件越大，下载时间越长，播放速度也就越慢。作为发布的一个步骤，Flash 会自动优化动画，例如查找动画中的相同部分，相同部分在文件中只存储一次。导出动画之前可以根据需要自行优化动画。在作出优化改动后，在不同的计算机和不同的操作系统上进行测试。

- 基本优化原则：
  - 避免在同一时间段内设置太多的动画，否则在配置较低的机器上播放动画时，会因为显存不够而出现画面停滞的情况。

- 将动画中多次出现的元素（如图形等）作为元件处理。
- 如果可能的话，使用渐变动画，这样会比使用一系列关键帧所生成的动画文件小得多。
- 尽量限制关键帧中变化区域的范围，使动画尽量在一个较小的区域中产生。
- 尽量避免在动画中使用比特速图，因为位图所占的存储空间相对较大。
- 尽可能使用 MP3 格式的声音文件。

- 元素与线的优化：
  - 尽可能地将动画元素组合。
  - 在动画过程中可能有的部分在变化而有的部分不变，将这些不同的部分分别设在不同的层中可以优化动画。
  - 使用 Modify | Optimize 菜单命令对轮廓曲线进行优化，使轮廓线所包含的曲线数目尽量减少。
- 文本与字体的优化：
  - 限制字体及字体风格的数目，尽量不使用嵌入式字体，避免增大动画文件的大小。
  - 如果使用嵌入式字体，在 Character（字符）面板中选择必需的设置即可，不必选用所有的设置。
- 颜色的优化：
  - 尽量不使用渐变色。使用渐变色填充比用纯色多需要 50KB 的存储空间。
  - 尽量不对透明度进行设定，否则将使播放速度减慢。

10.4 将 Flash MX 动画添加到网页中

10.4.1 Flash 的插件

所有的浏览器都有供用户阅读文本、查看图片、接听声音等的内置功能，但是，在早期，浏览器开发人员意识到可以通过允许其他开发人员添加功能来提高产品的可用性。这样的创意已被诸如 Macromedia Director 、FreeHand 和 Adobe Photoshop 这样的优秀的软件所采纳。

对于浏览器同样也可以使用插件，有些插件是预装的，但是如果查看某一网页所需要的插件是浏览器所没有的，便会询问你是否将它下载，还是允许它自动下载。一旦插件已下载和安装，就可以看到符合设计人员意图的网页。

要查看 Flash MX 的动画内容，同样需要安装 Flash MX 的插件。并且每一个新的 Flash 版本都带有一个新版本的插件。因为旧版本的 Flash 插件无法领会 Flash 的新的脚本编辑功能。虽然旧的功能仍然有可能正常运行，但是如果要使用 Flash 所提供的新功能，需安装新版的 Flash 的插件。

其实，Flash 的插件就是一个标准的 OCX 控件，它的名称为 Shakewave Flash Object。它不但可以集成到浏览器中，还可以直接作为控件加入到 Visual Basic 或者 Visual C++的工程中，从而使编译出来的 EXE 文件中也嵌入 Flash MX 动画。

10.4.2 直接输出 HTML 文件

输出 HTML 文件是 Flash MX 默认输出格式的一种。选择 File | Publish Settings 菜单命令，在弹出的对话框中选择 HTML 标签，如图 10-14 所示。

设定方法如下：

- Template（模板）：Flash MX 提供了各种类型的模板，可以把制作完成的 Flash MX 动画方便地加入到 HTML 页面中。在下拉式列表框中列举了 Macromedia\Flash MX \HTML 文件夹中的所有模板的名字。如果不对模板进行选择，那么 Flash 会用默认的模板。选中一个模板后单击列表框右边的 Info（信息）标签，会弹出如图 10-15 所示的模板信息对话框。对话框中列出了 Name（模

板名称）、Filename（模板文件的名称）以及 Description（对模板的描述）。下面对主要的模板进行介绍。

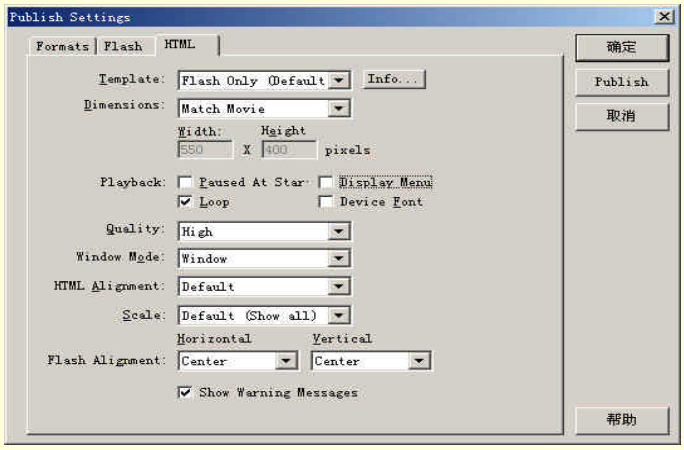


图 10-14　HTML 标签

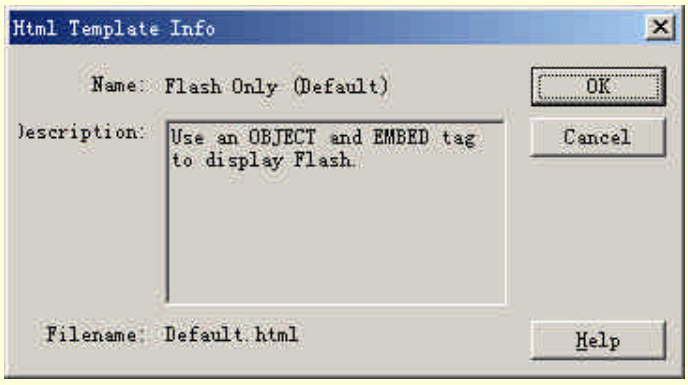


图 10-15　模板信息

- Detect for Flash 3：使用脚本语言检测 Flash Player 3 插件是否存在。
- Detect for Flash 4：使用脚本语言检测 Flash Player 4 插件是否存在。
- Detect for Flash 5：使用脚本语言检测 Flash Player 5 插件是否存在。
- Detect for Flash 6：使用脚本语言检测 Flash Player 6 插件是否存在。
- Flash Only：标准地嵌入 Flash MX 动画的 HTML 模板。
- Flash with FScommand：标准地嵌入 Flash MX 动画并自动在生成的 HTML 页面中添加处理 FScommand 的 JavaScript 脚本。
- Flash with Named Anchors：使以 Flash 6 格式发布的动画在 HTML 页面中具有书签功能。
- Image Map：在浏览器中显示图片，而不是 Flash MX 动画。使用此选项必须选中一幅同时发布的图片，如 GIF、JPEG 或者 PNG 格式的图片。
- Pocket PC 2002：发布以 Pocket PC 2002 为平台的包括 Flash 动画的 HTML 页面。
- QuickTime：播放 QuickTime 影片。

- Dimensions（尺寸）：尺寸列表框中有 3 种尺寸方案供选择：Match Movie（匹配影片）、Pixels（像素）、Percent（百分比）。
  - 匹配影片：此项为默认选择，导出的动画尺寸与原始尺寸相匹配。
  - 像素：选择此项后列表框下面的 WIDTH（宽）与 HEIGHT（高）两个文本框会变为可选状态。在其中填入数值可以确定导出动画的宽与高由多少像素点组成。
  - 百分比：选择此项后，宽与高将受百分比控制。在宽与高文本框中填入的数值用于控制动画在浏览器中所占用的比例。100%表示横向或者纵向填满浏览器窗口。这种方式可以使

Flash MX 动画自动适应浏览器窗口大小并随窗口而缩放。

- Playback（回放）：回放包括四个复选框。
  - Paused at Start（在开始时停止）：使动画在播放开始时处于暂停状态，当在菜单中选择播放时动画才开始播放。默认情况下此项不被选中，动画在载入后便会自动播放。
  - Loop（循环）：选中此项后动画在播完最后一帧后会自动跳转到第一帧循环播放；如果不循环，动画会在播放完最后一帧后停止。默认情况下此项被选中。
  - Display Menu（显示菜单）：默认情况下“显示菜单”处于选中状态。在浏览器中观看动画时单击右键，会弹出如图 10-16 所示的菜单。以便控制动画的播放，包括 Zoom In（放大）、Zoom Out（缩小）、Play（播放）、Forward（前进）、Back（后退）等。若不选中此项，单击右键弹出的菜单中只含有如图 10-17 所示的 About Macromedia Flash Player 6 一项。

在顺序播放的 Flash MX 动画之中，可以允许显示控制菜单，使用户可以随时控制动画的播放。如果在 Flash MX 动画中有脚本控制动画，使动画没有按照时间轴的顺序播放，那么，一旦用户使用此菜单控制动画的播放将产生无法预料的错误。所以应该关闭此控制菜单。

图 10-16　显示控制菜单

图 10-17　不显示控制菜单

  - Device Font（设备字体）：使用设备字体可以在字较小的时候使文本内容清晰显示，并且可以减小动画文件的大小。此选项只有在动画中含有静态文本的时候才有用。
- Quality（动画质量）：用于设定动画播放速度和动画质量之间的平衡关系。
  - Low（低）：如图 10-18 所示，以牺牲动画质量为代价来提高播放速度。选择此项后播放的动画不会消除锯齿。
  - Auto Low（自动低级）：注重动画的播放速度，在适当时候会提高画面质量。
  - Auto High（自动高级）：注重动画的播放速度，消除锯齿功能刚刚打开。当播放速度降到 Movie Properties（动画属性）标签内所设定的速度以下时，消除锯齿的功能会自动关闭。
  - Medium（中等质量）：这是一种较为折衷的选择，消除锯齿的功能有所加强，但对位图不进行优化。
  - High（较高质量）：如图 10-19 所示，这是默认选项。动画的播放速度降得更低，经常启用消除处锯齿功能。
  - Best（最高质量）：不考虑播放速度，画面质量最好。对所有输出都进行优化，即使位图也是如此。

图 10-18 消除处锯齿

图 10-19 消除处锯齿

- Window Mode（窗口模式）：只有在安装了 Flash ActiveX 控件的 Windows IE 4.0（或更高版本）之后才可用。
  - Windows（窗口）：此项为默认选项。提供最佳的播放效果，动画在 HTML 页中属于其自己的矩形区域内播放。
  - Opaque Windowless（不透明窗口）：如图 10-20 所示，在 IE 浏览器窗口中隐蔽地移动动画区域后面的元素。
  - Transparent Windowless（透明窗口）：如图 10-21 所示，使动画的背景色变为透明，这样动画所在的 HTML 页的背景就会显露出来。

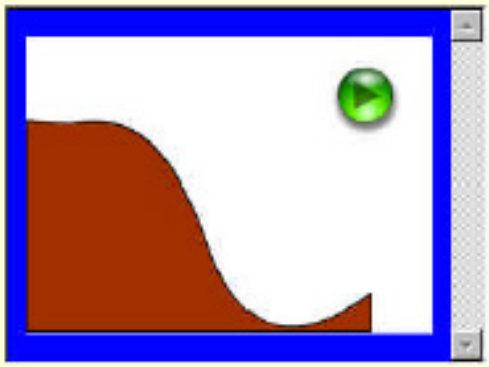


图 10-20 不透明窗口

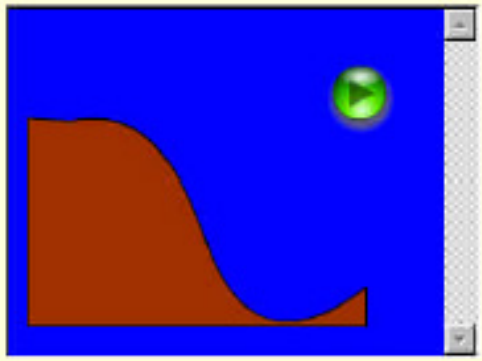


图 10-21 透明窗口

- HTML Alignment（HTML 对齐）：确定动画窗口在浏览器窗口中与其他元素的对齐关系。只有在动画附近放置了其他元素或重新编辑网页时此项才起作用，除了这两种情况外此选项作用不明显。在列表框中有五种选择：Default（默认）、Left（左）、Right（右）、Top（上）、Bottom（下）。
- Scale（比例）：确定动画在指定的动画窗口中的外观。
  - Default（默认）：即全部显示。动画元素按比例缩放使动画全部显示在动画窗口中。
  - No Border（无边框）：不显示边界，动画的一部分有可能被切掉。
  - Exact Fit（完全适合）：动画强制适应播放窗口的大小。可能会出现画面比例失调的情况，即横向和纵向的缩放比例不一样。
- Flash Alignment（Flash 对齐）：决定动画在动画窗口中的对齐情况。
  - Horizontal（水平）：此列表框有 Left（左）、Center（中）、Right（右）3 个选项。
  - Vertical（竖直）：此列表框有 Top（上）、Center（中）、Bottom（下）3 个选项。
- Show Warning Messages（显示警告信息）：选中此项会在标签设置矛盾时出现警告信息。

设定完成之后，使用 Publishing 按钮就可以把制作好的 Flash MX 动画直接发布到网页上。

10.4.3 使用 Dreamweaver 添加动画

使用上一节中 Flash MX 自带的直接发布的功能，可以把 Flash MX 动画放置在 HTML 页面中，但是，

这种方法并不能直接对 HTML 页面进行编辑，最后发布的 HTML 页面中也仅有一个 Flash MX 对象。

如果要编辑 HTML 页面，可以使用其他的主页编辑器，如 Dreamweaver。它可以使 Flash MX 对象与 HTML 页面中的其他对象相配合。因为 Dreamweaver 与 Flash 都是 Macromedia 公司的产品，所以 Dreamweaver 对 Flash 动画支持得很好。

启动 Dreamweaver 3，单击如图 10-22 所示的按钮可以直接插入 Flash MX 动画（*.SWF）。

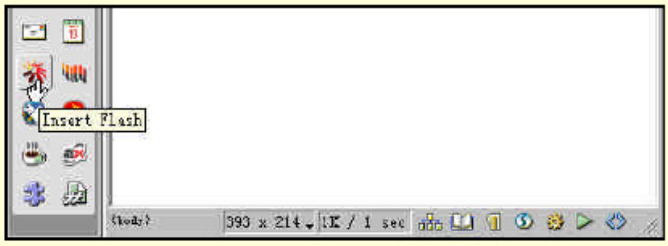


图 10-22 插入 Flash MX 动画

插入之后的 Flash MX 动画如图 10-23 所示。Dreamweaver 会自动根据在 Flash MX 中设置的影片的尺寸定义 Flash MX 动画在 HTML 页面中的尺寸。

虽然在 Dreamweaver 中编辑 Flash MX 动画，并没有显示出动画的具体内容，而只是以一个专用的图标显示动画的位置及大小。这样做可以加快编辑的速度，但是，并不影响最后的结果，最终生成的 HTML 页面在浏览器中仍然可以正常显示并播放动画。

图 10-23 添加 Flash 动画

选中 Flash MX 动画之后，可以在如图 10-24 所示的属性面板中设置动画的各种属性。

图 10-24 属性面板

- 尺寸：在 W、H 文本框中可以设置动画在浏览器中 W（横向）及 H（纵向）的尺寸，也可以直接输入百分比（如 100%）来控制动画在浏览器中的相对大小使动画随浏览器缩放。
- File（文件路径）：指定 Flash MX 动画的位置，最好使用相对路径。
- Tag（标签类型）：选择使用的标签的类型 OBJECT 或者 EMBED。
- Align（排列）：指定 Flash MX 动画在网页中的排列方式。关于此项的详细情况请参考 Dreamweaver 方面的书籍。
- BgColor（背景颜色）：设置 Flash MX 动画的背景颜色，此选项只在设定 Flash MX 动画为透明的时候才有意义。
- Border（边框）：设置 Flash MX 动画对象周围边框的宽度，如果不想设置边框，则可以不填此

项或者设为 0。

- Space（空间）：在 V、H 文本框中可以设置 Flash MX 动画在浏览器中与其他对象之间的 H（水平）和 V（垂直）空间。
- Quality（动画质量）：设置 Flash MX 动画播放时的质量。包括：Low（低）、Auto Low（自动低级）、Auto High（自动高级）、High（高）四个选项。它们的意义请参考 10.4.2 节中对输出 HTML 文件的设置。
- Scale（比例）：确定动画在指定的动画窗口中的外观。
  - Default（默认）：即 Show All（全部显示）。动画按比例缩放使动画全部显示在动画窗口中。
  - No Border（无边框）：不显示边界，动画的一部分有可能被切掉。
  - Exact Fit（完全适合）：动画强制适应播放窗口的大小。可能会出现画面比例失调的情况，即横向和纵向的缩放比例不一样。
- Loop（循环播放）：选中此项后动画在播完最后一帧后会自动跳转到第一帧循环播放；否则，动画会在播放完最后一帧后停止。
- Autoplay（自动播放）
- Parameters（参数设置）：单击 Parameters（参数）设置按钮将出现如图 10-25 所示的对话框，可以为 Flash MX 动画设置其他参数。

图 10-25　参数设置对话框

其实，上面设置的动画的属性都是通过设置 Flash MX 动画的参数而实现的。此外，Flash MX 动画的参数还包括：

- Base：指定 Flash MX 动画的地址。
- Menu：设置动画快捷菜单的模式。如果为 True 则显示完整的菜单；如果为 False 则快捷菜单中只有 About Flash 选项，与 10.4.2 节“直接输出 HTML 文件”中菜单的设置完全相同。
- Wmode：设置显示模式。可以是 Window、Opaque、Transparent 等 3 个值。具体意义如下：
  - Windows（窗口）：此项为默认选项。提供最佳的播放效果，动画在 HTML 页中属于其自己的矩形区域内播放。
  - Opaque Windowless（不透明窗口）：在 IE 浏览器窗口中隐蔽地移动动画区域后面的元素。
  - Transparent Windowless（透明窗口）：使动画的背景色变为透明，这样动画所在的 HTML 页的背景就会显露出来。

10.4.4　测试 Flash 动画的下载过程

Flash MX 动画是一种流式文件，即它可以一边下载一边播放，但是，当网络的传输速率不高时，一旦有某一部分数据没有及时被下载，那么 Flash MX 将出现播放的停顿。在测试动画的时候选择 View | Bandwidth Profiler 菜单命令，将显示出如图 10-26 所示的整个动画每一帧的下载所需要的数据。在右边的条型图中用明暗相间的数据条表示每一帧的数据及下载的时间。

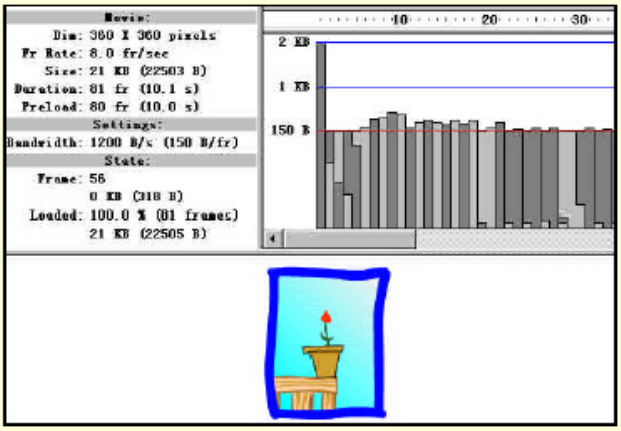


图 10-26　下载数据

默认的数据图表是 Streaming Graph（流式图），通过选择 View | Frame by Frame Graph 菜单命令可以将数据图表改变为如图 10-27 所示的逐帧数据显示图。

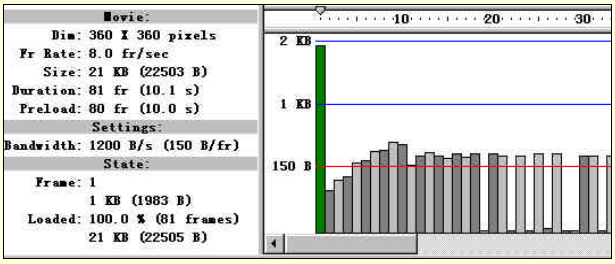


图 10-27　逐帧数据显示图

在上述图表中的红线为数据传输的上限，因此可以清楚地看到到底是哪几帧影响了动画的播放。如果要设置传输速率，可以选择如图 10-28 所示的 Debug 菜单中的选项。

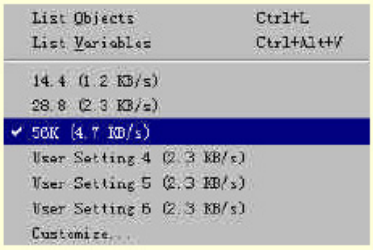


图 10-28　Debug 菜单

使用 Debug | Customize 命令，可以在弹出的如图 10-29 所示的对话框中自定义传输速率。

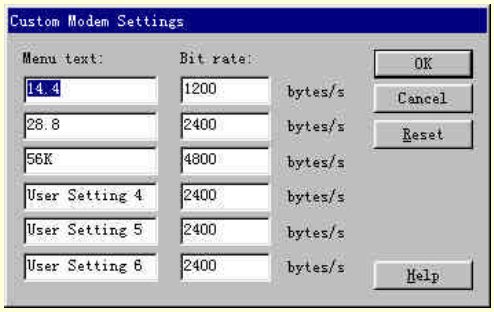


图 10-29　自定义传输速率

- Menu text：此文本框中的文本为显示在 Debug 菜单中的选项。
- Bit rate：数据传输的速率。

- 在 Bit rate 框中添加数值的单位是 bytes/s（字节/秒）。而通常所说的 Modem 的速率单位是 bit/s（比特/秒）。虽然在计算机中 1byte=8bit，但是还要算上串行通信所需要的起始位和结束位等，所以这里大概是 1byte=12bit。

选中 View | Show Streaming 菜单命令或者使用 Ctrl + Enter 快捷键，如图 10-30 所示，可以预览动画在当前传输速率下的播放情况。

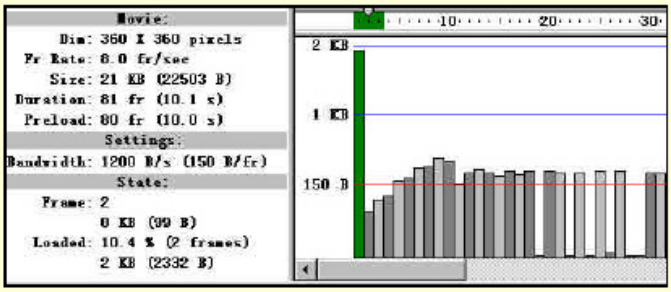


图 10-30 预览播放情况

- 时间轴中绿色的部分为已经下载完毕的数据。

第 11 章　Flash MX 与网络

11.1　综　述

Flash MX 的动画文件可以与外部（如文件或者服务器）进行信息和数据的交流。例如，使用 loadVariables 或者 getURL 动作可以发送或装入一个变量；使用 loadMovie 动作可以在 Flash MX 的播放器中装入其他文件。另外，Flash MX 引入了 XML 功能。使用 XML 或者 XMLSocket 对象，可以发送或者接收 XML 数据。同时，利用 XML 对象所提供的方法，用户也可以构造自己的 XML 数据。

同时，还可以使用文本区域或者弹出式菜单组成具有 Flash MX 风格的表单来收集用户所输入的信息，并把它提交给服务器端的应用程序进行处理。

此外，使用 Flash MX 的 fscommand 动作，可以使 Flash MX 动画与 Flash 6 播放器或者浏览器中的 JavaScript 脚本函数进行信息的交换。

11.2　参数、变量的传递

11.2.1　从外部文件装入变量

在网页上的 Flash MX 动画与普通的 HTML（超文本）页面的作用一样，它是一个接收并且显示信息的窗口。但是，与普通的 HTML（超文本）页面上其他对象不同，Flash MX 动画可以主动连续地从互联网上更新信息，而不用用户刷新或者跳转到新的页面。这一切都要归功于 Flash MX 提供了许多相关的 Actions（动作）和对象的方法。利用它们可以很容易地与服务器端的脚本、文本文件或者 XML 文件进行信息的传递。

服务器端的脚本是指运行在互联网的服务器上的程序，它并不下载到客户机上运行，但是它可以与运行在客户机上的超文本页面进行进行信息的传递。通常利用服务器端的脚本访问或者修改服务器端的数据库，并把相关的信息传送到客户机上的超文本页面上。服务器端的脚本可以用很多种语言编写，最常见的有 Perl、ASP 和 PHP 等。

把信息存储在服务器的数据库中，在需要的时候随时读出，利用这种技术可以创建出动态的个性化的 Flash MX 网络动画。例如：记录来访用户的信息和访问时间，可以在用户下一次登录此页面的时候直接进入上一次离开时的页面，或者提醒来访用户查看尚未看过的信息。

在 Flash MX 中提供了多种与外部交互信息的途径。每一种途径都有固定的交互标准和协议。例如：

- GetURL、loadVariables、loadMovie 动作：使用 HTTP 或者 HTTPS 协议以 URL 编码的形式发送信息。
- XML 对象中的 XML.send、XML.load、XML.sendAndLoad 方法：使用 HTTP 或者 HTTPS 协议以 XML 的形式发送信息。
- XMLSocket 对象中的 XMLSocket.connect、XMLSocket.send 方法：通过创建或者使用 TCP/IP Socket 连接发送信息。

图 11-1 显示了 Flash 6 播放器决定是否接受一个 HTTP 请求的过程：

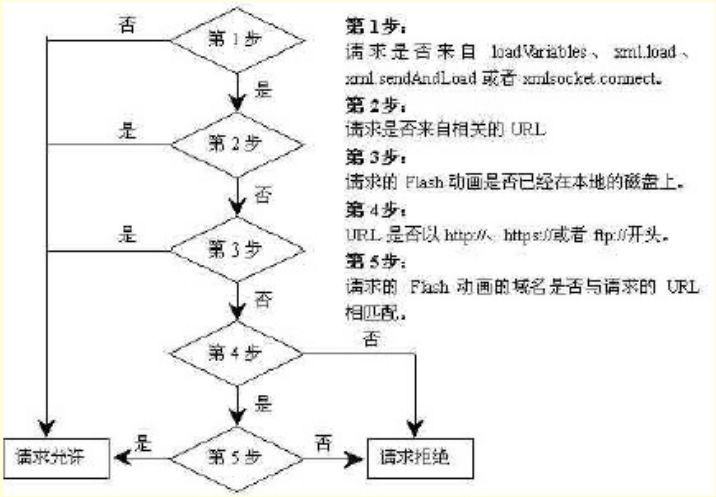


图 11-1　是否接受一个 HTTP 请求

因为 Flash MX 动画的信息交流依靠 HTTP 或者 HTTPS 协议，所以它的安全性与标准的 HTML 是完全相同的。也就是说，设计 Flash MX 动画的信息传递应该与 HTML 遵循相同的规则。例如：在 Flash MX 动画中对访问密码的支持，其过程如下：

（1）使用 Flash MX 文本区域获得用户所输入的密码。

（2）用 loadVariables 动作或者 XML.sendAndLoad 方法通过 HTTPS 协议的地址的方式发送到服务器。

（3）服务器检查密码是否有效，并返回相关信息。

可见，密码的检查是在服务器中进行的，而在客户机运行的 Flash MX 动画（SWF）文件没有任何关于密码的信息，所以，它的安全性很高。

11.2.2　以前的方式

在 Flash 以前的版本中，只能通过 GetURL、loadVariables 或者 loadMovie 动作实现与外部文件的交流。它们都是通过 HTTP 协议与服务器端的脚本进行数据的交互，其作用如下：

- getURL：返回信息到浏览器中（而不是 Flash 的播放器中）。它就相当于一个在 Flash MX 动画中使用的超级链接，经常用于打开指定的网页。
- loadVariables：装载变量到 Flash MX 动画的指定帧中。Flash MX 主要靠此动作与服务器进行信息的交互。
- loadMovie：装载其他 Flash MX 动画（*.swf）到 Flash MX 播放器的指定层次中。

其中，loadVariables 动作是 Flash MX 动画与服务器进行交互的关键，利用此动作可以实现几乎所有的数据传递的功能。其具体的使用方法将在 11.2.4 节中予以介绍。

11.2.3　getURL 的使用

其实，在 Flash MX 文本区域中设置 URL 选项也可以提供超级链接，但是，getURL 作为 Flash MX ActionScript 脚本的一个动作，它可以在任何时刻指向一个超级链接（如为按钮提供超级链接），并且提供比文本区域中更丰富的设置。

GetURL 的语法如下：

```
getURL(url [, window [, variables]]);
```

- url 是一个字符串，它指向一个超级链接的地址，如：http：//www.Macromedia.com
- window 表示打开此超级链接的窗口或者 HTML 的 Frame（框架）。它可以是一个特定的窗口（或者框架）的名称，也可以是如下预定义名称：

- _self：如图 11-3 所示，在当前动画所在的窗口中直接打开超级链接（图 11-2 为连接之前的主页）。

图 11-2　连接之前的主页

图 11-3　连接之后的主页

- _blank：如图 11-4 所示，在新窗口中打开超级链接。
- _parent：在当前动画所在窗口的父窗口中直接打开超级链接。
- _top：在当前动画所在窗口的最顶层窗口中直接打开超级链接。

图 11-4　新窗口中打开超级链接

- variables 表示传递变量的方式。包括 GET 和 POST：
  - GET 方式把变量直接加到 URL 地址后面。这种方式适合于小数据量的发送。
  - POST 方式把变量由一个独立的 HTTP 字符串发送出去。这种方式适合于发送长字符串变量。

✎ window 和 variables 参数可以省略。

11.2.4　LoadVariables 的使用

LoadVariables 动作可以从外部文件中读取数据。外部文件可以是文本文件，也可以是由 CGI 脚本、ASP（Active Server Pages）或者 PHP（Personal Home Page）等服务器端软件产生的文本文件。通常文本文件中为 Flash MX 动画或者影片剪辑设置了变量的值，这是一种主动更新变量的好办法。

其中，文本文件的 URL 地址必须是标准的 MIME 格式，而且动画文件与被装载的变量必须在同一个域名下。其语法如下：

```
loadVariables (url ,location [, variables]);
```

- url 表示其目标文件所处的地址。
- location 表示接收数据的变量所处在的层次。它可以是一个影片剪辑，也可以是一个表示层次的数字，如，0 表示_level0 层，也就是第一个被装入主场景中的影片剪辑。
- variables 表示传递变量的方式（可选参数），包括 GET 和 POST：
  - GET 方式把变量直接加到 URL 地址的后面。这种方式适合于小数据量的发送。

- POST 方式把变量由一个独立的 HTTP 字符串发送出去。这种方式，适合于发送长字符串变量。

下面的例子将从一个文本文件中更新变量：

（1）新建一个 Flash MX 文件，在如图 11-5 所示主场景的第 1 帧中添加一个文本区域。

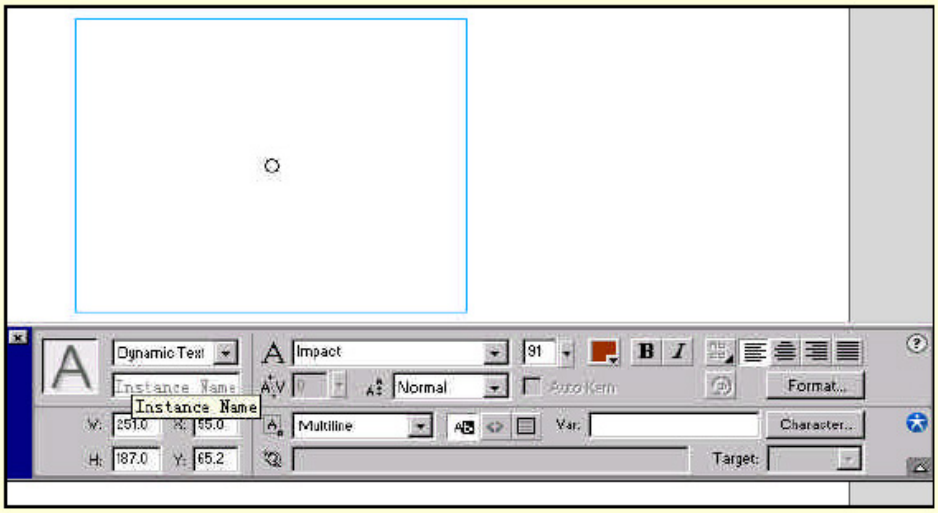


图 11-5　添加一个文本区域

（2）如图 11-6 所示，在文本区域的上方添加一个按钮。

图 11-6　添加按钮

（3）为按钮加入如下脚本：

```
on (release) {
    _root.loadVariables("data.txt", 0);
}
```

（4）保存此 Flash MX 文件，并在相同的目录创建如下文本文件，文件名称为 data.txt（与脚本中 loadVariables 动作的第一个参数相同）。

```
data=<在等号后面输入文本，此文本显示在文本区域中>……
```

（5）输出并运行动画。单击按钮之后，如图 11-7 在 data.txt 文本文件中等号后面的文本将出现在文本区域中。

图 11-7　更新文本区域

（6）如果改变文本区域为如图所示 10-8 的格式，则可以在文本区域中直接显示超文本文件的内容。

图 11-8　文本区域格式

（7）用网页编辑工具制作一个超文本文件保存之后，如图 11-9 所示使用文本编辑器打开超文本文件并在第一行加入 data=。

图 11-9　更改超文本文件

（8）输出并运行动画。单击按钮之后，如图 11-10 在文本区域中将出现超文本文件的内容。

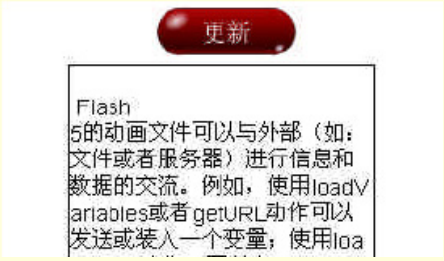


图 11-10　文本区域的超文本文件

✎ 文本区域中只显示超文本文件的内容，超文本的其他格式（如字体等格式）将丢失。

可见，利用上述方法可以在不改变 Flash MX 动画文件本身的同时，改变动画的内容或者运行结果。这是一种更新动画的快捷方法，它可以实现 Flash MX 动画网页的动态更新。

LoadVariables 与服务器脚本之间的信息交互是通过标准的 MIME 格式的 URL 地址实现的。例如：

```
loadVariables("http://www.mySite.com/scripts/score.php", _root.scoreClip, GET);
```

下面的例子通过 ASP 实现了一个通过服务器检查密码的过程：

（1）新建一个 Flash MX 文件。创建一个影片剪辑，并在影片剪辑的第 1 帧中加入如图 11-11 所示的文本区域。

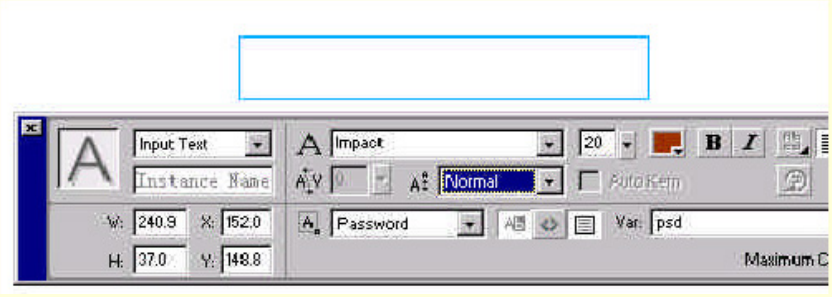


图 11-11　设置文本区域

✎ 一定要将文本区域的设置成 Passsword 类型。这样在文本区域中输入的文字将不会公开。

（2）把影片剪辑添加到主场景中，并在影片剪辑的下面放入如图 11-12 所示的按钮。

图 11-12　添加按钮

（3）如图 11-13 所示，把影片剪辑命名为 psdmov。

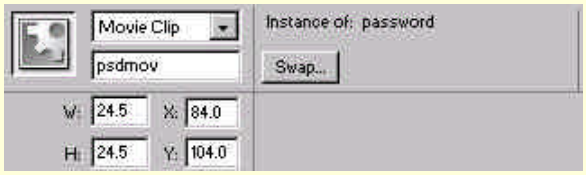


图 11-13　命名影片剪辑

（4）为按钮添加如下脚本：

```
on (release) {
    loadVariables ("check.asp?action=check&psd="+psd, "padmov", "POST");
}
```

（5）在按钮的旁边添加一个如图 11-14 所示的显示密码检查结果的文本区域。

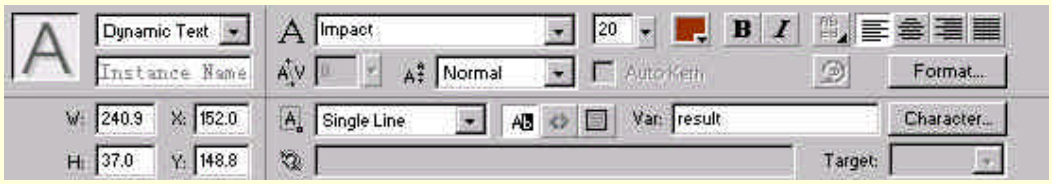


图 11-14　添加检查结果的显示区域

（6）为影片剪辑 psdmov 添加如下脚本：

```
onClipEvent (data) {
    if (password_test==true) {
        result = "正确";
    } else {
        result = "错误";
    }
}
```

（7）使用默认值将 Flash MX 直接发布到网页中。

（8）创建 check.asp 脚本文件如下：

```
<%
Dim action,passowrd,right_psd
right_psd="12345"
action = Request.QueryString("action")
password = Request.QueryString("psd")
If action = "check" Then    '检查密码
    If right_psd<>password Then    '密码错误
        Response.write("password_test=0")
    Else
```

```
            Response.write("password_test=1")
        End If
    End if
    %>
```

（9）将 Flash MX 发布的超文本文件 password.html 和 password.swf 文件复制到主页服务器的根目录中（或者使用发布工具进行发布）。在浏览器中直接输入地址 http://localhost/password.html，即可显示出如图 11-15 所示的 Flash MX 动画页面。

图 11-15　运行结果

如图 11-16 所示，如果在文本区域中输入 12345，单击“提交”按钮，则先提示用户等待检验过程，然后再显示“正确”，如果输入其他文本则显示“错误”。

图 11-16　密码正确

在上述与服务器交互信息的例子中，主要使用了如下技巧：

- 因为密码的检验是在服务器中进行的，从客户机提交密码到服务器端脚本检验密码并返回，由于网络的传输速度以及服务器的运行速度不同，将产生不定时的延时。在 Flash 5 以前的版本中，判断服务器端是否返回结果需要在时间轴上创建一个循环等待。而在 Flash MX 中为影片剪辑提供了 Data 事件，当影片剪辑获得数据的时候将产生此事件。所以只需要将 LoadVariables 动作中的目标参数指向一个影片剪辑，然后处理此影片剪辑的 Data 事件即可。
- LoadVariables 动作中的 URL 地址参数是标准的 MIME 格式。以 ASP 脚本为例，其格式为：

myASP.asp?myvalue1=xxx&myvalue2=xxx&……。例如：

score.asp?action=getscore&highScore1=54000&playerName1=rockin

在 ASP 后面所跟的几个参数之间使用&符号分隔开。因为一个 ASP 脚本可以完成许多不同的任务，所以在它的参数中使用一个 action 参数作为判断其任务的标识。

- 在 ASP 脚本中使用 Response.write(String)可以直接为 Flash MX 动画中的变量赋值。参数 String 为一个字符串，它可以通过等号运算符为客户机上运行着的 Flash MX 动画的变量赋值。例如：

  Response.write("password_test=1")

被赋值的变量 password_test 在 Flash MX 动画中所处的层次是 LoadVariables 动作中的目标参数指向的层次。如果参数为空，则变量为_root 下的全局变量。

另外，测试上述例子时，一定要在支持 ASP 的系统上运行。比如，装有 Personal Web Sever（个人主页服务器）的 Windows 9x 操作系统。或者装有 IIS 的 Windows NT\2000\XP 服务器等。否则，ASP 将不能正确运行。有关 ASP 的使用超出了本书的范围，请读者自行参考有关方面的书籍。

11.2.5 LoadMovie 动作

LoadMovie 的语法如下：

loadMovie(url [,location/target, variables]]);

- url 表示要装载的 Flash MX 动画文件*.swf 所处的位置。它可以是绝对位置，也可以是相对位置，在这里建议使用相对位置。被装载的动画文件一定要与当前动画文件处于同一个子域。例如，在 Flash 6 播放器中播放或者测试动画的模式下，所有的动画文件必须位于相同的子目录下，并且引用的文件名不能包含磁盘符或者绝对目录。例如：C:\myflash\movie.swf 应将“ C:\myflash\ ”去掉。
- target 是可选参数，它表示一个影片剪辑，装载进来的 Flash MX 动画文件将取代此影片剪辑，并且它的位置、角度、比例等参数交通此影片剪辑完全相同。
- location 是一个可选参数，并且它和 target 参数不能同时存在。它指明了装载进来的 Flash MX 动画文件所处的 level（层次）。如果此层次中没有其他的影片剪辑，则动画文件将直接装载进来，否则它将取代此层次中的影片剪辑，并继承影片剪辑中的所有属性。

  ✍ 如果想要取代原有的动画文件并且卸载所有的层，则自己装载动画文件到第 0 层。第 0 层将决定其他被装载影片的帧速率、背景颜色和影片大小。
- variables 表示传递变量的方式（可选参数）。包括 GET 和 POST：
  - GET 方式把变量直接加到 URL 地址的后面，这种方式适合于小数据量的发送。
  - POST 方式把变量由一个独立的 HTTP 字符串发送出去。这种方式适合于发送长字符串变量。

✍ 使用 unloadMovie 动作可以卸载 loadMovie 所装载的 Flash MX 动画。

11.2.6 XML 概述

XML（Extensible Markup Language）语言是一种 Internet 上的结构化数据交互标准。使用 Flash MX 引入的 XML 技术可以实现 Flash MX 动画与服务器之间的数据交互。比如说，创建一个聊天室系统。

HTML 语言一样，XML 语言也使用标签来标定每一段文本。在 HTML 语言中，使用预定义标签来表示一段文本在浏览器中显示的方式（例如<b>标签表示文本的粗体显示）。同样在 XML 语言中也可以使用标签来定义一段数据（例如，<password> VerySecret </password>）。同时，XML 语言把数据同它们的显示方式分离起来，这样，就保证了同一个 XML 文件可以在不同的环境中反复使用。

每一个 XML 标签都被称作 node（节点）。节点中可以拥有自己的属性，节点中的节点被称作 ChildNode（子节点）。这种分层的树状节点结构就叫作 DOM，Document Object ModelXML（文件对象模型），它类似于浏览器中所使用的 JavaScript DOM 结构。

下面是一段 XML 语言，请注意它的结构：

```
<PORTFOLIO>
    <HOLDING SYMBOL="RICH"
        QTY="75"
        PRICE="245.50"
        VALUE="18412.50" />
</PORTFOLIO>
```

其中，<PORTFOLIO>是父节点，它不包含任何属性，但是它包含一个子节点<HOLDING>，在子节点中包含 SYMBOL、QTY、PRICE 和 VALUE 属性。

11.2.7 使用 XML 对象

使用 ActionScript 脚本对象中所提供的方法（例如，appendChild、removeNode 和 insertBefore）可以在 Flash MX 动画中构造一个 XML 数据结构，然后把它发送到服务器；或者解释并处理从服务器端发来的 XML 数据结构。

下面的 XML 对象的方法通过 HTTP 邮递的方法可以实现与服务器之间的 XML 数据的发送或者接收。

- load：从指定的 URL 地址中接收 XML 数据，并把它存放在 ActionScript 脚本的 XML 对象中。
- send：向指定 URL 地址发送 XML 数据，返回信息将被发送到另外的浏览器窗口中。
- sendAndLoad：向指定的 URL 地址发送 XML 数据，信息返回到 XML 对象中。

例如，创建一个 Flash MX 风格的 BBS 系统，在服务器的数据库中将存储所有用户的信息（包括用户名、密码、ID 号、积分、签名档案等）。如果使用 XML 对象，则它的工作过程如下：

运行在服务器端的脚本将以 XML 的格式读写数据库并与运行在客户端的 Flash MX 动画进行信息的交互。在客户端，使用 ActionScript 脚本将用户向 Flash MX 动画中输入的信息（如用户名、密码等）转换成为 XML 对象，然后再以 XML 文件的形式发送给服务器端的脚本。同样，也可以使用 ActionScript 脚本接收并转换服务器返回的 XML 文件到 XML 对象中，以便在 Flash MX 动画中调用 XML 对象中的信息。图 11-17 表示了此过程：

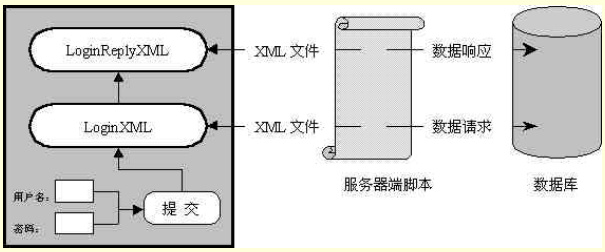


图 11-17 XML 交互流程图

下面的例子使用 XML 对象的方式传递数据来判断用户名和密码的正确性：

（1）新建一个 Flash MX 文件。直接在主场景的第 1 帧中添加如图 11-18 所示的文本区域。

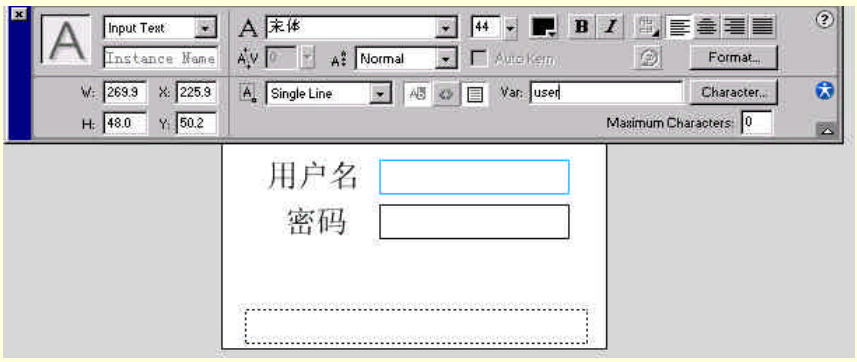


图 11-18 添加文本区域

（2）添加如图 11-19 所示的提交按钮。

图 11-19　添加按钮

（3）为提交按钮添加如下脚本：

```
on (release) {
    //使用 LOGIN 节点构造一个 XML 文件对象
    loginXML = new XML();
    loginElement = loginXML.createElement("LOGIN");
    loginElement.attributes.username = user;
    loginElement.attributes.password = psd;
    loginXML.appendChild(loginElement);

    //构造一个 XML 对象接收服务器的回复
    loginReplyXML = new XML();
    loginReplyXML.onLoad = onLoginReply;

    //向服务器发送 LOGIN 节点
    _root.result="正在处理，请稍候……";
    loginXML.sendAndLoad("https://www.myweb.com/main.cgi",loginReplyXML);
}
```

✍ 假设用户名为 Tom、密码为 12345。则上述脚本所构造的 LOGIN 节点相当于

```
<LOGIN USERNAME="Tom" PASSWORD="12345" />
```

（4）在主场景的第 1 帧中添加如下脚本：

```
function onLoginReply() {
    //获得第一个 XML 节点
    var e = this.firstChild;
    //如果的一个节点为 LOGINREPLY，并且它的值为 OK，则显示正确
    //否则显示错误
    if (e.nodeName == "LOGINREPLY" && e.attributes.status == "OK") {
        result = "正确";
    } else {
        result = "错误";
    }
}
```

✍ 假设从服务器返回的 XML 文件遵循如下原则：

如果正确则返回：

```
<LOGINREPLY STATUS="OK"/>
```

如果错误，则返回：

```
<LOGINREPLY STATUS="FAILED"/>
```

✍ 另外，函数 onLoginReply 是一个回调函数，在服务器有返回信息的时候被调用的。在提示按钮的如下脚本中提供了此函数的入口。

```
loginReplyXML.onLoad = onLoginReply;
```

（5）发布 Flash MX 的网页，并将超文本文件和*.swf 文件发布到服务器上，并调试。

✍ 上述例子中，服务器端的脚本 https://www.myweb.com/main.cgi 并没有给出，读者可自行参考相关书籍，这里不作介绍。

11.2.8 使用 XMLSocket 对象

ActionScript 脚本中提供的 XMLSocket 对象可以同服务器创建一个连续的连接。有了 Socket 连接以后，可以使服务器在任何时候向客户机发送信息，否则服务器只能等到有 HTTP 请求的时候才可以和客户机通信。这种开放式的交互加强了信息的传递，因此它被广泛地应用于创建实时的应用，如在线聊天。在 Socket 连接之间传递的信息使用的是 XML 格式，所以可以利用 XML 对象构造数据。

创建一个 Socket 连接，必须编写一个运行在服务器上的应用程序，用来等待或者接受 Socket 连接的请求并运行相应的 Flash MX 动画。这种应用程序可以用编程语言来创建，如 Java。

在客户端，使用 ActionScript 脚本 XMLSocket 对象的 connect 和 send 的方法通过 Socket 连接与服务器间传递 XML 结构的数据。XMLSocket 对象的 connect 方法是用一个固定的网络端口进行 XMLSocket 对象的连接；Send 方法则通过此端口发送 XML 结构的数据。

一旦使用 connect 方法创建了一个连接，Flash 6 播放器将始终打开一个 TCP/IP 连接，直到如下情况之一发生：

- 使用 close 方法关闭连接
- 退出 Flash 6 播放器
- 连接意外中断（例如 Modem 掉线）

下面的例子创建了一个 XML 的 Socket 连接，并且利用 XML 发送数据。

```
//创建一个 XMLSocket 对象
sock = new XMLSocket();
//使用 connect 方法通过 1024 端口建立连接
sock.connect("http://www.myserver.com", 1024);
//定义一个函数接收 Scoect 句柄
//服务器响应：如果成功则发送 myXML 对象，否则显示错误信息
function onSockConnect(success){
    if (success){
        sock.send(myXML);
    } else {
        msg="There has been an error connecting to "+serverName;
    }
}
sock.onConnect = onSockConnect;
```

✍ 使用 connect 方法建立连接所使用的端口必须与服务器端的端口统一，并且必须大于等于 1024。因为小于 1024 的端口经常被系统的其他服务所使用。例如，FTP、Telnet、HTTP 等服务。

11.3 向播放器发送消息

使用 Flash MX 的 ActionScript 脚本所提供的 fscommand 动作可以使 Flash MX 动画向它的执行环境发送信息（例如浏览器、Director 播放器或者 Flash 6 的在线播放器）。这样使 Flash MX 动画的功能进一步得到扩展。比如，通过向 HTML 网页发送信息可以引发 JaveScript 函数的调用，从而实现对浏览器的操作。

11.3.1 向 Flash player 发送消息

Fscommand 动作的语法如下：

```
fscommand(command, arguments);
```

其中参数 command、arguments 都是向播放器所发送的字符串命令。

Fscommand 动作向 Flash MX 动画的播放程序（或者环境）发送任何消息。但是如果向独立的 Flash 6 播放器发送消息，只能使用 Fscommand 动作的预定义命令和参数。例如，下面的脚本可以使播放器全屏显示。

```
on(release){
    fscommand("fullscreen", "true");
}
```

表 11-1 列出了所有的 Fscommand 动作的预定义命令和参数，它们都可以作为向独立的 Flash 6 播放器发送的消息。

表 11-1 Fscommand 动作的预定义命令和参数

命 令	参 数	说 明
Quit	None	关闭 Flash 6 播放器
Fullscreen	True/False	是否将 Flash 6 播放器切换到全屏模式 True：将 Flash 6 播放器切换到全屏模式 False：将 Flash 6 播放器切换到标准模式
Allowscale	True/False	是否允许缩放 True：使 Flash 6 动画同播放器一起缩放 False：Flash 6 动画始终以 100%的比例显示
Showmenu	True/False	如图 11-20 所示是否显示 Flash 6 播放器的菜单 True：显示 Flash 6 播放器的菜单 False :隐藏 Flash 6 播放器的菜单（除 About Flash Player.以外）
Exec	应用程序路径	运行一个指定的应用程序
Trapallkeys	True/False	是否使键盘上所有键有效

✎ 命令 Exec 的参数不要求一定是一个 Exe 文件，也可以把它指向其他文件（如*.hlp），Windows 将自动地使用相应的编辑器打开此文件。利用此项功能就可以在 Flash MX 动画中直接调用它的说明文件（readme.txt）或者帮助文件（*.hlp）。在下一章中将给出类似的例子。

图 11-20　显示与隐藏菜单

11.3.2　向网页发送消息

在网页中的 Flash MX 动画，如果使用 fscommand 动作向 HTML 网页中的脚本（如 JavaScript）发送信息，则 fscommand 的两个参数 command、arguments 可以是任何字符串。当在动画中使用 fscommand 动作的时候，在 Flash MX 动画所处网页中的 moviename _DoFSCommand 函数将被调用。此函数由 JaveScript 语言编写，其中 moviename 为 Flash MX 动画的文件名。例如，如果 Flash MX 动画的文件名为 myMovie，那么 JavaScript 相应函数名称为 myMovie_DoFSCommand。

下面给出在网页中应用 Fscommand 动作的例子：

（1）新建一个 Flash MX 动画文件。在主场景窗口中添加如图 11-21 所示的按钮。

图 11-21　添加按钮

（2）分别为两个按钮添加如下脚本：

```
//按钮 1
on (release) {
    fscommand ("messagebox", "此消息来自按钮 1");
}
//按钮 2
on (release) {
    fscommand ("messagebox", "此消息来自按钮 2");
}
```

（3）发布动画文件到网页上，其设置如图 11-22 所示。

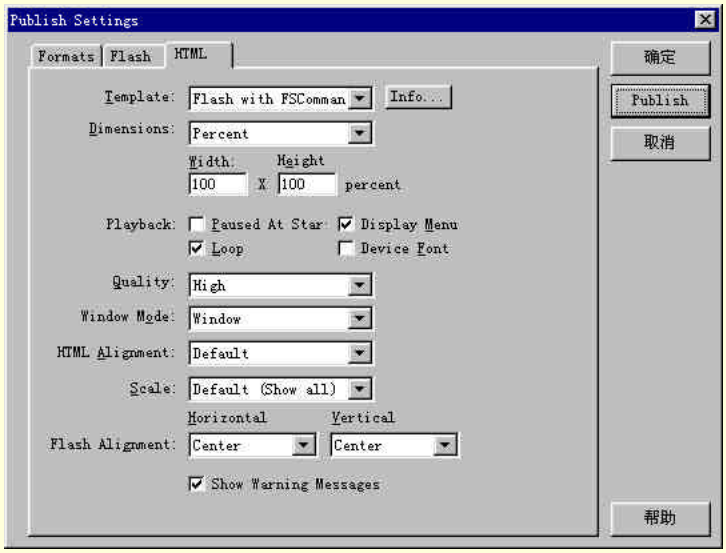


图 11-22　设置发布 HTML 方式

将网页文件（*.html）用文本编辑器打开，将其改为如下内容：

```
<HTML>
<HEAD>
<TITLE>fscommand</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<SCRIPT LANGUAGE=JavaScript>
<!--
var InternetExplorer = navigator.appName.indexOf("Microsoft") != -1;
// Handle all the the FSCommand messages in a Flash movie
function fscommand_DoFSCommand(command, args) {
  var fscommandObj = InternetExplorer ? fscommand : document.fscommand;
  //
  // Place your code here...
  //
    if(command="messagebox"){
        alert(args);
    }
}
// Hook for Internet Explorer
if (navigator.appName && navigator.appName.indexOf("Microsoft") != -1 &&
      navigator.userAgent.indexOf("Windows") != -1 && navigator.userAgent.indexOf("Windows 3.1") == -1)
 {
    document.write('<SCRIPT LANGUAGE=VBScript\> \n');
    document.write('on error resume next \n');
    document.write('Sub fscommand_FSCommand(ByVal command, ByVal args)\n');
    document.write('    call fscommand_DoFSCommand(command, args)\n');
    document.write('end sub\n');
```

```
        document.write('</SCRIPT\> \n');
    }
    //-->
    </SCRIPT>
    <!-- URL's used in the movie-->
    <A HREF=FSCommand:messagebox></A> <!-- text used in the movie-->
    <!--窗ヅ 1  窗ヅ 2 --><OBJECT classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
    codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=5,0,0,0"
     ID=fscommand WIDTH=100% HEIGHT=100%>
     <PARAM NAME=movie VALUE="fscommand.swf"> <PARAM NAME=quality VALUE=high> <PARAM
    NAME=bgcolor VALUE=#FFFFFF> <EMBED src="fscommand.swf" quality=high bgcolor=#FFFFFF
    WIDTH=100% HEIGHT=100% swLiveConnect=true NAME=fscommand
    TYPE="application/x-shockwave-flash"
    PLUGINSPAGE="http://www.macromedia.com/shockwave/download/index.cgi?P1_Prod_Version=ShockwaveFlash
    "></EMBED>
    </OBJECT>
    </BODY>
    </HTML>
```

(4) 执行上述文本文件 (*.html)，单击在浏览器中 Flash MX 动画中的两个按钮，将显示如图 11-23 和图 11-24 所示的不同的消息对话框。

图 11-23　按钮 1 的消息对话框

图 11-24　按钮 2 的消息对话框

上述例子说明，通过使用 Fscommand 动作可以使网页中的 Flash MX 动画与网页上的 JavaScript 脚本交互信息，从而实现在 Flash MX 动画中无法完成的功能（如上述的消息对话框的功能）。

第 12 章　Flash MX 动画实战

12.1　动画“雪花纷飞”

12.1.1　原　理

本例将利用 Flash MX 创作出漫天大雪的动画。显然，所有雪花的运动不可能一个一个地由手工制作并添加而完成，而需要使用 ActionScript 脚本语言在动画运行时动态地创建。然而，如果简单地复制同一个雪片落下的动画，将使雪花下落的动画过于单一，而不像自然雪。因此就需要使用 ActionScript 脚本语言随机地控制雪片下落的动画，使雪片有不同的透明度、不同的大小、不同的下落方向。

此外，制作的雪片下落的动画实际上是雪片以不同速度、沿不同轨迹下落的几段动画的集合。在播放动画时，用 ActionScript 脚本控制动画随机播放。这样，几种下雪动作掺杂起来，就形成了极其形象的“雪花纷飞”的效果。

12.1.2　实　战

（1）新建一个 Flash MX 文件，把影片的背景色设置成黑色。如果打算在全屏的模式下播放此动画，建议将电影的分辨率设置为 640×480 像素。创建一个 Graph（图形）元件命名为 snow_gra，并在舞台的中央位置绘制一个大小如图 12-1 所示的白色的圆（无描绘颜色）。

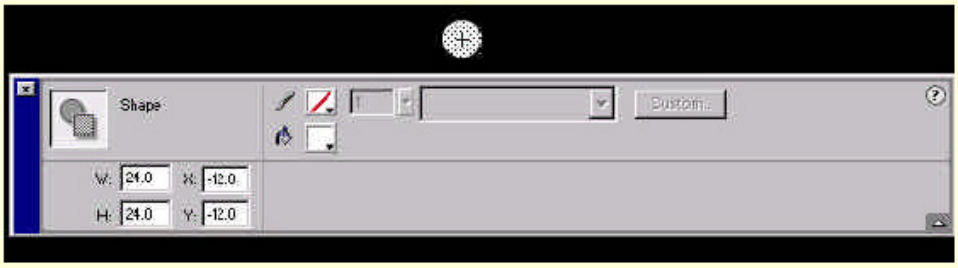


图 12-1　绘制白色圆

（2）选中圆，使用 Modify | Shape | Soften Fill Edges 菜单命令，在弹出的 Soften Edges（柔化边缘）对话框中按照图 12-2 进行设置。单击 OK 按钮，这使圆产生由边缘向内柔化的效果，具有雪花的质感，这样一片雪花便制作完成了，如图 12-3 所示。

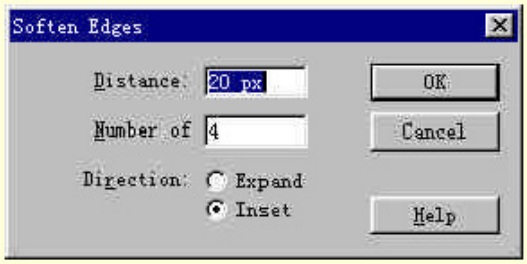


图 12-2　柔化边缘对话框

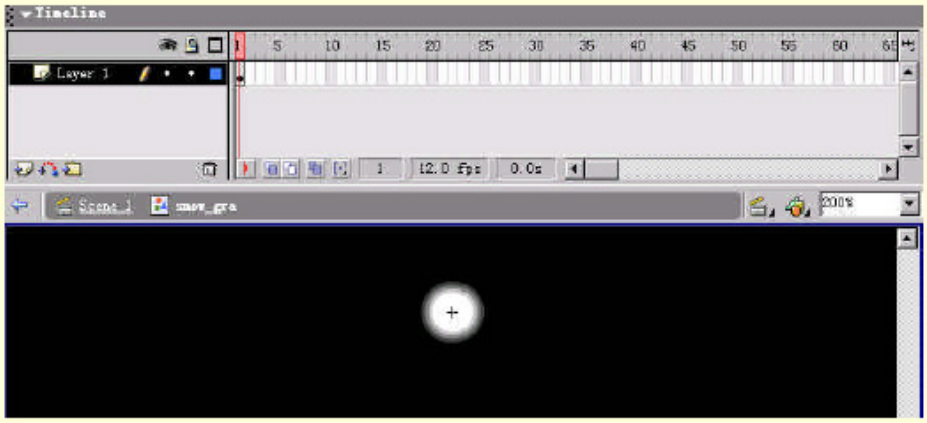


图 12-3　柔化的结果

- Soften Edges（柔化边缘）对话框中的 Number of 项决定了柔化的层次数，层次数值越大，柔化效果越好，但是这样将导致动画速度降低。尤其是在大量雪花同时飘落时或者在配置较低的计算机上显示动画时，会出现明显的停顿现象。

（3）创建一个 Movie Clip（影片剪辑），命名为 Snow_mov。在影片剪辑的第 2 帧、第 4 帧、第 5 帧处插入空白帧。把元件 Snow_gra 拖放到第 5 帧上。

（4）在影片剪辑 Snow_mov 的 Layer1 层上创建 Guide（向导）层，在向导层的第 5 帧处插入关键帧。并绘制一条如图 12-4 所示的曲线表示雪花飘落的轨迹。曲线的曲率不要太大，否则，难以产生雪花飘落的效果。

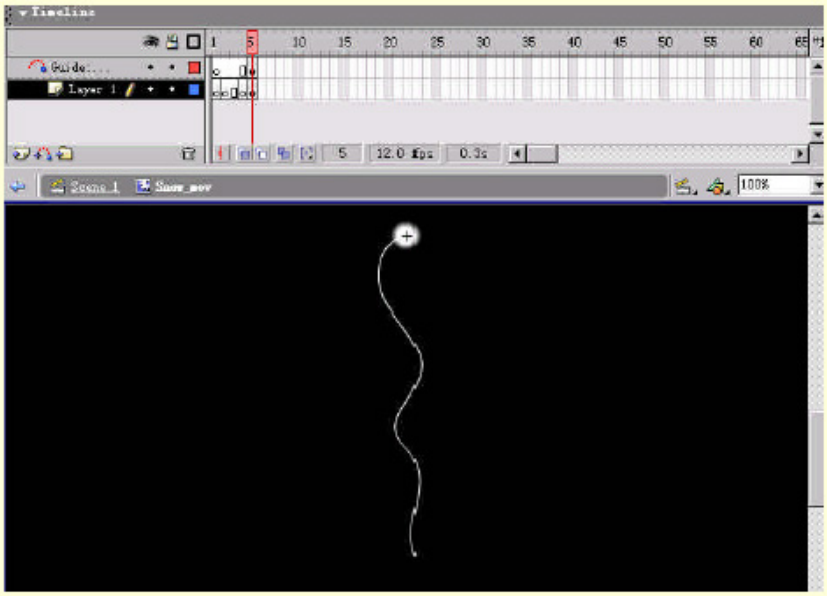


图 12-4　雪花飘落的轨迹

（5）将雪花放置在轨迹上端点，在向导层的第 40 帧处插入帧，在 Layer1 的第 40 帧处插入关键帧，并将雪花放置在轨道下端点。调节第 40 帧处的雪花的 alpha（透明度）为 0，在 Layer1 层的第 5 帧处使用位置渐变。制成雪花飘落并逐渐淡出的动画，效果如图 12-5 所示。

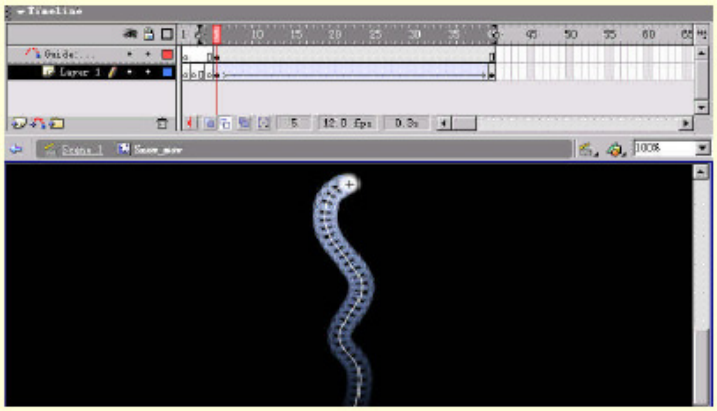


图 12-5　雪花飘落动画的洋葱皮表示

（6）将影片剪辑的 Layer1 层和向导层分别从第 5 帧开始复制到第 40 帧，并粘贴在第 41 帧处。改变第 41 帧处的向导层的轨迹形状，使雪花沿不同的轨迹以不同的形式落下，如图 12-6 所示。

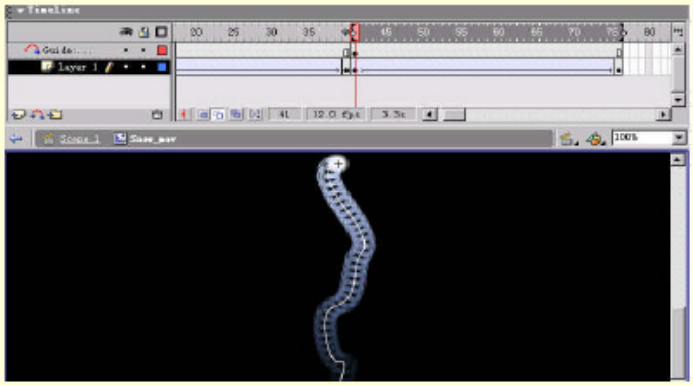


图 12-6　改变雪花下落的轨迹

（7）将影片剪辑的 Layer1 层和向导层分别从第 5 帧开始复制到第 76 帧，并粘贴在第 77 帧处。改变新拷贝的两段动画首末帧之间的距离，使雪花以不同的速度落下。效果如图 12-7 所示。

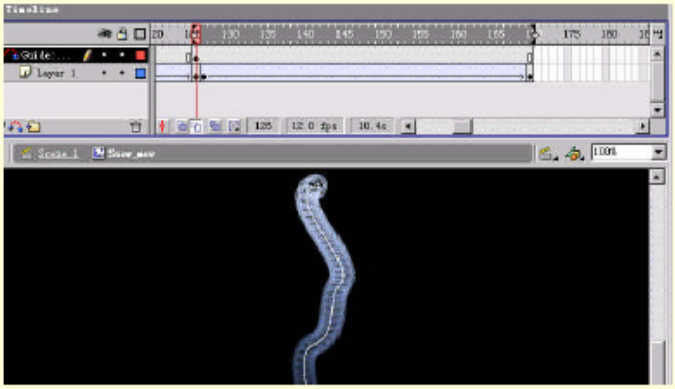


图 12-7　改变雪花飘落的速度

（8）分别在每一段动画的起始帧处（第 5 帧、第 40 帧、第 77 帧、第 127 帧）添加 Label（标签），以区分不同的动画片段。标签名称依次为：mov0、mov1、mov2、mov3。如图 12-8 所示。

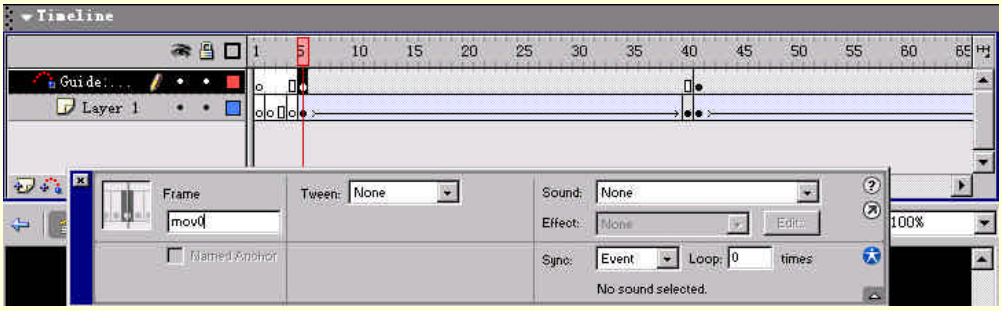


图 12-8　添加帧标签

（9）添加第 1 帧的动作如下：

```
timer = random(50);
gotoAndPlay (3);
```

（10）添加第 4 帧的动作如下：

```
if (timer == 0) {
    var mov;
    mov = "mov" add random(3);
    gotoAndPlay (mov);
    return；
}
timer--;
gotoAndPlay (3);
```

第 1 帧与第 4 帧的动作互相配合实现一个随机的延时，这样将保证在同一时刻创建的雪花动画不会同时落下。其实现原理是：首先在第 1 帧产生一个随机延时数赋予全局变量 timer，然后跳转到第 3 帧，第 3 帧没有任何动作和画面，只相当于延时了 0.2 秒。之后，执行到第 4 帧，判断 timer 是否为 0，如果非 0 则 timer 减 1，跳到第 3 帧继续延时；如果为 0，则延时结束，随机跳转到 4 段动画中的任何一段并播放。

（11）在每一段动画的最后一帧添加如下动作，使无论哪一段雪花动画播放完毕都停止在第 2 帧。

```
gotoAndStop (2);
```

（12）返回主场景 Scene1，在元件库面板的 Snow_mov 元件上右击鼠标，选择快捷菜单中的 Linkage 选项。如图 12-9 所示在对话框中进行选择。

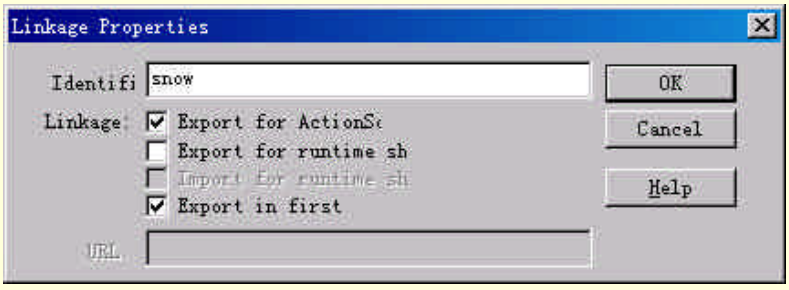


图 12-9　填写 Linkage 对话框

（13）在主场景 Scene1 的第 1 帧处加入动作：

```
snow_num = 50;
rnd_scale=100;
```

```
angle = 30;
for (n=1; n<=snow_num; n++) {
    CreateSnow("snow"+n,n);
}
function CreateSnow (sn,n) {
    var scale;
    attachMovie("snow", sn, n);
    this[sn]._x = random(600)-50;
    this[sn]._y = -10;
    scale = 20+random(rnd_scale);
    this[sn]._xscale = scale;
    this[sn]._yscale = scale;
    this[sn]._rotation = -random(angle);
    this[sn]._alpha = 20+random(80);
}
```

（14）在主场景 Scene1 的第 5 帧处加入动作：

```
var sn;
for (n=1; n<=snow_num; n++) {
    sn = "snow"+n;
    if (getProperty(sn, _currentframe) == 2) {
        removeMovieClip (sn);
        CreateSnow(sn, n);
    }
}
gotoAndPlay (2);
```

- 在主场景的第 1 帧中，通过循环调用 CreateSnow 函数创建了 50 个不同层深的雪花动画。然后，经过主场景的第 2~4 帧的延时，在第 5 帧的动作中检查这 50 片雪花中哪一片雪花落完了。雪花动画停在影片剪辑的第 2 帧上，则说明雪花落完了，应该删除雪花动画，重新创建，以便在新位置产生新的雪花继续飘落。
- CreateSnow 在创建完每一片雪花之后，都在一定范围内随机的设置雪花动画的属性（位置、转角、比例、透明度等）。其中转角使雪花在下落的过程中向一个方向倾斜，有一点微风吹雪的感觉。

（15）输出并执行动画，雪片便纷纷从屏幕上方下落，如图 12-10 所示。

图 12-10 “雪花纷飞”的效果

（16）因为所用的动画过程都是由 ActionScript 脚本创建的，所以很容易就可以控制雪花的多少。在主场景中添加一个新层 Layer2，在新层上添加 3 个如图 12-11 所示的按钮“大雪”、“中雪”、“小雪”。

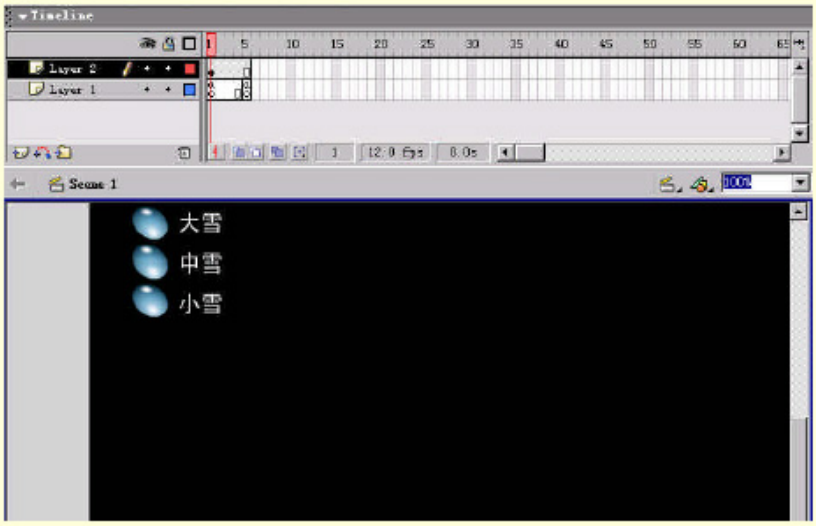


图 12-11　添加按钮

（17）分别为 3 个按钮加入如下脚本：

- “大雪”

```
on (press) {
    snow_num = 70;
    rnd_scale=120;
}
```

- “中雪”

```
on (press) {
    snow_num = 50;
    rnd_scale=100;
}
```

- “小雪”

```
on (press) {
    snow_num = 20;
    rnd_scale=70;
}
```

（18）输出并执行动画，雪的大小就可以受用户控制了，效果见图 12-12。

图 12-12　受控制的雪

12.1.3 总 结

动画“雪花纷飞”主要运用了以下几项技巧：

- 使用 attachMovie（）函数在程序运行的过程中动态地创建动画，这更适合于创建复杂的动画。
- 使用 gotoAndPlay（）来实现循环和延时。
- 使用随机函数 random 方便地产生任意区间内的整数，例如，产生 a-b 区间内的整数，则直接使用 a+random（b-a）即可。这比使用 Math 对象中的 random 方法更简便。
- 通过 gotoAndPlay（）和随机函数 random 将一个影片剪辑分成了可以随机播放的几段影片，每次播放都会有不同的结果。当然还可以创造出更多的飘落过程的动画，使雪花飘落得更为逼真。

为了增强通用性，下面把上述例子中的雪花动画封装到一个 Movie Clip（影片剪辑）中：

（1）新建一个 Flash MX 文件，设置背景色为黑色。创建一个影片剪辑命名为 snow。如图 12-13 所示把上述动画主场景中雪花的一层 Layer1 中的 1~5 帧，复制到此影片剪辑中。

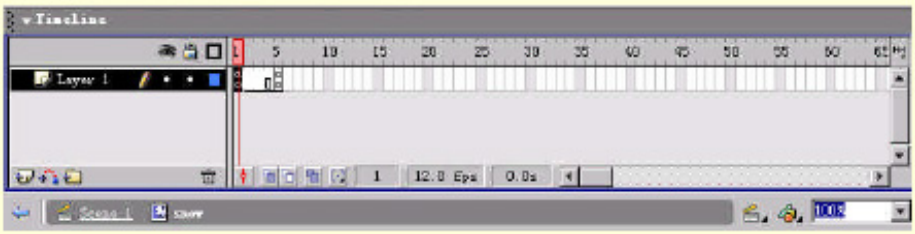


图 12-13 复制雪花层

（2）如图 12-14 所示，同时打开两个文件中的库，把上述例子中的影片剪辑 Snow_mov 拖放到新建文件的库中。

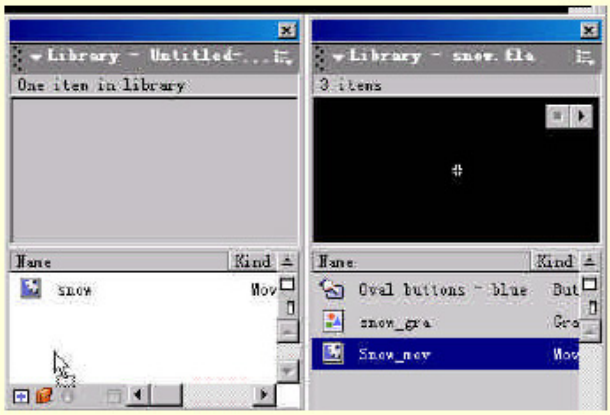


图 12-14 复制影片剪辑

因为图形元件包含在影片剪辑 Snow_mov 中，所以 Flash MX 也将它自动地复制到新文件中。

（3）在元件库中右击 snow 影片剪辑，选择快捷菜单的 Component Definition 项，如图 12-15 所示，在弹出的对话框中为影片剪辑添加参数。

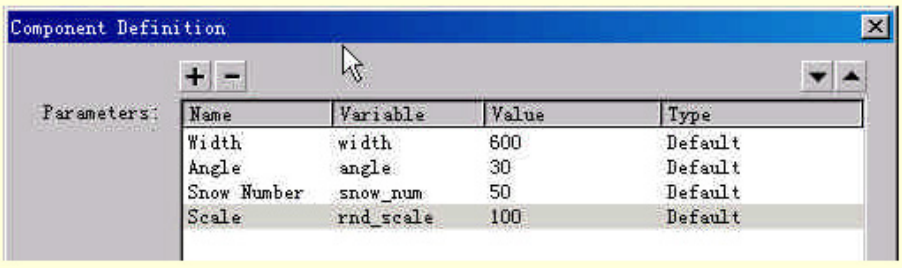


图 12-15 添加参数

（4）改变影片剪辑第一帧中的代码为：

```
for (n=1; n<=snow_num; n++) {
    CreateSnow("snow"+n,n);
}
function CreateSnow (sn,n) {
    var scale;
    attachMovie("snow", sn, n);
    this[sn]._x = random(width)-width/2;
    this[sn]._y = 0;
    scale = 20+random(rnd_scale);
    this[sn]._xscale = scale;
    this[sn]._yscale = scale;
    this[sn]._rotation = -random(angle);
    this[sn]._alpha = 20+random(80);
    }
```

（5）为了便于管理和使用，选中元件库中的 3 个元件，如图 12-16 所示使用 Move to New Folder 命令，把它们放在一个元件夹中并命名为 Snow_folder。

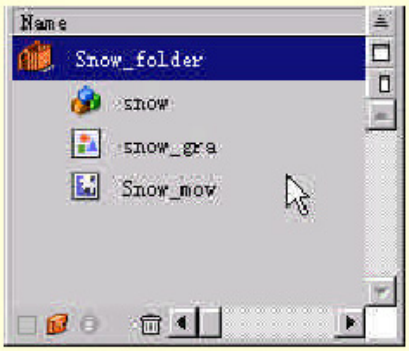


图 12-16　创建元件夹

（6）现在雪花动画已经被封装到了一个影片剪辑中，下面进行测试。如图 12-17 所示把影片剪辑 snow 拖放到主场景上边的中间位置。

图 12-17　放置雪花效果

✎ 因为雪花是在动画播放时动态地创建出来的。在编辑的时候，影片剪辑 snow 中没有任何图形，所以 Flash MX 用一个圆点表示未选中的元件的中心位置；用“ ＋ ”字表示选中的元件的中心位置。

（7）设定影片剪辑 snow 中的参数或者使用默认值。输出并运行动画。同样可以实现下雪的效果。

上述过程把“雪花纷飞”的动画放在了一个影片剪辑中，并设置了四个参数用于控制雪花飘落的风格，这样使此效果很方便地嵌入到其他动画中。其中 4 个参数的作用如下：

- width：下雪的宽度，即以影片剪辑为中心，下雪的水平范围。
- angle：下雪的随机角度的范围。顺时针为负，逆时针为正。它可以是雪产生一种被风吹的效果。

- snow_num：雪花数目。
- rnd_scale：雪花比例。

12.1.4 扩　展

在 12.1.3 节中把动画“雪花纷飞”封装到了一个影片剪辑中，可以很方便地改变其风格或者加入到其他动画中。下面将借助此影片剪辑创建一个跟随鼠标下雪的动画效果。

（1）新建一个 Flash MX 文件，设置背景色为黑色。选择 File | Open as Library 菜单项以元件库的形式打开 12.1.3 节中封装到 snow 影片剪辑的 Flash MX 动画，如图 12-18 所示选中 Snow_folder 元件夹下的所有元件并将它们拖放到此文件的元件库中。

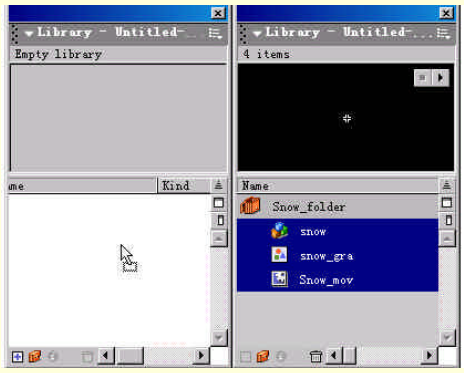


图 12-18　复制所有元件

✍ 因为影片剪辑 snow 是在脚本中创建雪花，但是雪花并不存在于影片剪辑中，所以，如果直接从打开的元件库中拖放影片剪辑 snow 到主场景中，Flash MX 不会把在 snow 的脚本中所用到的 snow_mov 元件也复制到当前文件中。因此需要手工复制。

（2）如图 12-19 所示，将影片剪辑 snow 拖放到主场景第 1 帧中的任意位置。

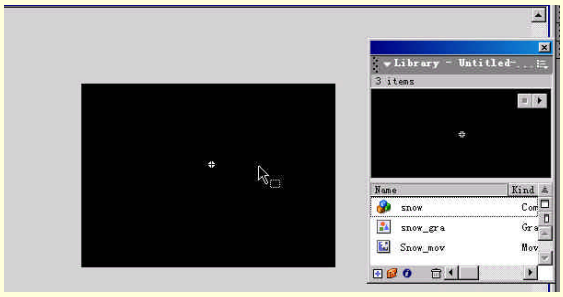


图 12-19　放置雪花

（3）设置影片剪辑 snow 的参数如图 12-20 所示。

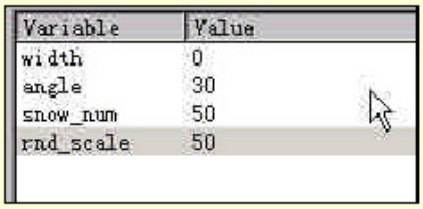


图 12-20　设置参数

（4）为影片剪辑 snow 加入如下脚本：

```
onClipEvent (load) {
    startDrag (this, true);
}
```

```
onClipEvent (mouseMove) {
    if (_root.hitTest(_root._xmouse,_root._ymouse,false)) {
        this._visible=true;
    } else {
        this._visible=false;
    }
}
```

（5）输出并运行动画。如图 12-21 所示，在鼠标所处的位置处不断地下雪。

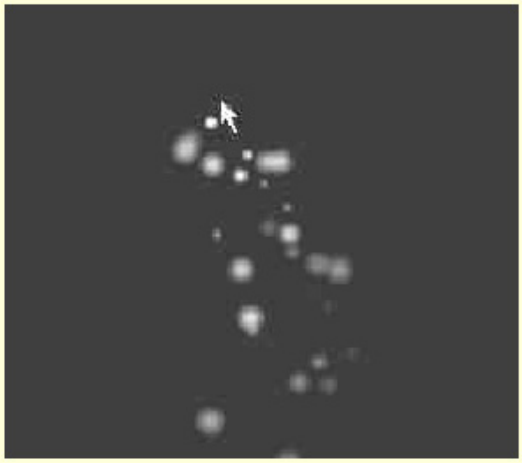


图 12-21　运行结果

上述动画只是把下雪的范围 width 参数设为 0，并适当地调整了雪花的大小和数目，配合 StartDrag 动作就可以使雪花跟随鼠标。可见，将制作成形的动画效果封装成影片剪辑，并为它设置一定的参数，是一种很方便的方法。

其实，“雪花纷飞”是一类动画的代表。利用上述技巧，可以制作出很多类似的动画，比如花瓣或者树叶的飘落、萤火虫的飞翔等随机形态的动画。相反，如果使代表“雪花”的圆球向上运动，再改变一下圆的填充色和大小，就形成了从水中冒出气泡的动画。

12.2　旋转的空间立方体顶点

12.2.1　介　绍

虽然 Flash MX 有着强大的动画功能，但是它只能制作二维的动画，如果想制作三维动画则必须借助外部软件（如 3D Max），并且配合 Flash MX 的插件，才可以制作成 Flash MX 格式的三维动画。其实这些所谓的三维动画被导入到 Flash MX 中以后不过是一系列逐帧的二维 Flash MX 动画，根本没有使用 Flash MX 的任何渐变功能。所以这些“三维”的 Flash MX 动画文件都很大，即使输出成*.swf 文件也不会很小，很不利于网上的传输。那么有没有别的方法生成更小的三维 Flash MX 动画文件呢？那就需要了解三维动画的创建过程。如图 12-22 所示，三维动画的生成主要包括：

- 三维建模：创建物体的三维模型，即把物体的空间形状告诉计算机。
- 三维运动：描述物体的空间运动过程，也可以像创建 Flash MX 动画一样以关键帧的形式创建三维运动。
- 二维投影：为了在二维的计算机屏幕上真实地显示出三维图形，必须通过某种算法把三维图形投影到平面上。

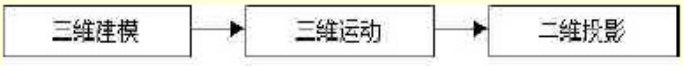


图 12-22　三维动画的创建过程

可见，如果要直接用 Flash MX 实现物体的三维运动，也必须经过上述步骤。显然直接利用 Flash MX 的简便功能是无法实现的，所以那就需要利用 Flash MX 所提供的 ActionScript 脚本通过程序进行三维建模；实现三维运动；最后进行二维投影。这就相当于用 ActionScript 脚本编写了一个简单的三维图形软件。在接下来的 12.2.2 节中将详细讲述三维图形的创建与变换，这是实现三维动画的基础。

12.2.2　3D 变换矩阵

本节主要讲述三维图形创建的理论方面的知识。如果读者对此不感兴趣完全可以跳过本节，直接阅读下面的例子，但是，如果想成功地编写三维图形的脚本，则这一节是很关键的部分。

在计算机图形学中通常使用一个四维向量[x　y　z　1]来表示三维空间的一个点向量[x　y　z]。此向量乘以一个变换矩阵所得到的新向量，就对应于三维空间的一个点经过同样的变换所得到的新点。其过程可以用下式表示：

[x'　y'　z'　1]=[x　y　z　1]·T

式中 T 为一个三维变换矩阵，它是一个 4×4 的方阵。即：

$$T=\begin{bmatrix} a & b & c & p \\ d & e & f & q \\ h & i & j & r \\ dx & dy & dz & s \end{bmatrix}$$

不同的变换对应不同的 T，下面把主要的三维变化介绍如下：

- 比例变换矩阵

$$T=\begin{bmatrix} a & 0 & 0 & 0 \\ 0 & e & 0 & 0 \\ 0 & 0 & j & 0 \\ 0 & 0 & 0 & s \end{bmatrix}$$

当 s=1 时，a、e、j 分别表示图形沿 *x*、*y*、*z* 三轴方向上的缩放比例。

当 s≠1，a=e=j=1 时，s 表示这个图形的缩放比例因子。s>1 图形缩小；0<s<1 图形放大。

- 平移变换矩阵

$$T=\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ dx & dy & dz & 1 \end{bmatrix}$$

上式中 dx、dy、dz 分别表示沿 *x*、*y*、*z* 三轴正方向上的偏移量。

- 旋转变换矩阵

三维旋转变换指绕坐标系的三根坐标轴旋转，转角 θ 的正负如图 12-23 所示按右手定则确定。

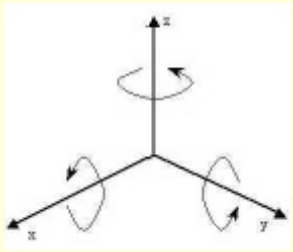


图 10-23　转角 θ 的方向

- 绕 x 轴转 θ 角

$$T=\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos q & \sin q & 0\\ 0 & -\sin q & \cos q & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

- 绕 y 轴转 θ 角

$$T=\begin{bmatrix}\cos q & 0 & -\sin q & 0\\ 0 & 0 & 0 & 0\\ \sin q & 0 & \cos q & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

- 绕 z 轴转 θ 角

$$T=\begin{bmatrix}\cos q & \sin q & 0 & 0\\ -\sin q & \cos q & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

- 正轴测投影

选择一个合适的角度，把三维图形投影到二维平面上，可以使用正轴测投影，使它具在立体感。正轴测投影的形成过程是这样的：现将空间一立体绕 z 轴正向转 θ 角，然后再绕 x 轴反向转 ϕ 角，最后让经过两次旋转后的立体向 xoz 平面投影。其整个过程可以用如下矩阵所表示：

$$T=\begin{bmatrix}\cos q & \sin q & 0 & 0\\ -\sin q & \cos q & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & \cos(-j) & \sin(-j) & 0\\ 0 & -\sin(-j) & \cos(-j) & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0 & 0\\ 0 & 0 & 0 & 0\\ 0 & 0 & 1 & \\ 0 & 0 & 0 & 1\end{bmatrix}$$

$$=\begin{bmatrix}\cos q & -\sin q\sin f & 0 & 0\\ -\sin q & -\cos q\sin j & 0 & 0\\ 0 & \cos j & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}$$

正等轴测投影（简称正等测投影）和正二轴测投影（简称正二测投影）是正轴测投影中两种常用的形式。正等轴测投影用手工绘制较为方便；正二轴测投影的立体感较好，但其手工绘制不如正等轴测投影方便。

当使用正等轴测投影时，取 θ=45°，ϕ=35°16'，则变换矩阵 T 为：

$$T=\begin{bmatrix} 0.707 & 0 & -0.408 & 0 \\ -0.707 & 0 & -0.408 & 0 \\ 0 & 0 & 0.816 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

当使用正二轴测投影时，取 θ=20°42'，ϕ=19°28'，则变换矩阵 T 为：

$$T=\begin{bmatrix} 0.935 & 0 & -0.118 & 0 \\ -0.345 & 0 & -0.312 & 0 \\ 0 & 0 & 0.943 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

通过使用上述的变换矩阵可以很方便地对建立好的三维模型进行运算，以实现物体的转动、平移、缩放等动画过程。

12.2.3 实 战

在 12.2.2 节中讲述了 3D 变换矩阵，下面将结合一个简单的空间立方体顶点旋转的例子来具体讲述 3D 变换矩阵在 ActionScript 脚本中的使用。

（1）新建一个 Flash MX 文件。创建一个影片剪辑，命名为 point。如图 12-24 所示在影片剪辑的第 1 帧中使用径向渐变填充色绘制一个立体的点，作为立方体的一个顶点。为了便于脚本的调试，在顶点的上方添加一个文本框，以便给顶点编号。

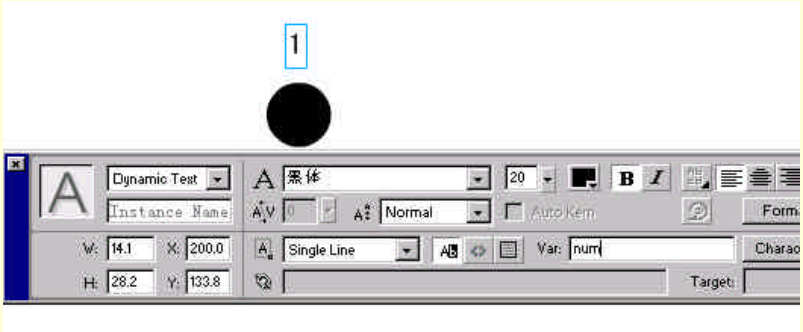


图 12-24 绘制顶点并添加编号

（2）返回主场景（Scene1），在元件库面板的 point 影片剪辑上右击鼠标，选择快捷菜单中的 Linkage 选项。按图 12-25 所示填写对话框。

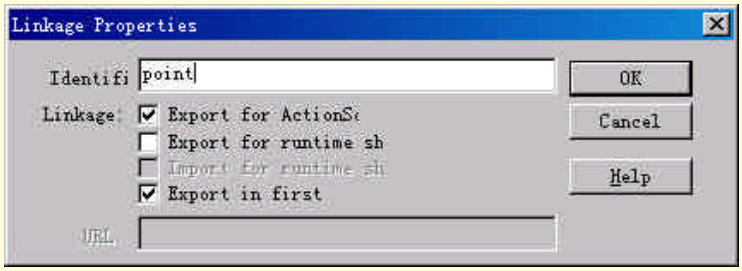


图 12-25 填写 Linkage 对话框

（3）在主场景的第 1 帧中添加如下脚本：

```
point = []; //顶点数组
var sidelen = 120;     //立方体边长
anglex = angley=anglez=0;        //沿 x、y、z 三轴所旋转的角度
```

```
step = Math.PI/40;          //立方体每次所转的角度
inicube();        //调用初始化立方体函数
//创建 8 个立方体顶点
for (i=0; i<8; i++) {
    attachMovie("point", "p"+i, i);
}
//立方体初始化函数，根据边长设置立方体的 8 个顶点坐标
function inicube () {
    point.push([sidelen/2, sidelen/2, sidelen/2]);
    point.push([-sidelen/2, sidelen/2, sidelen/2]);
    point.push([-sidelen/2, sidelen/2, -sidelen/2]);
    point.push([sidelen/2, sidelen/2, -sidelen/2]);
    for (n=0; n<4; n++) {
        point.push([point[n][0], -sidelen/2, point[n][2]]);
    }
}
//计算并显示立方体转角的函数
function calpos () {
    var temp_pt = [];       //临时点的数组
    var pt2d = [];          //将三维立方体顶点向二维坐标下投影的坐标数组
    for (n=0; n<8; n++) {       //计算转角
        //调用 rotation(顶点,x 转角,y 转角, z 转角)函数，返回旋转之后的顶点
        temp_pt[n] = rotation(point[n], anglex, angley, anglez);
    }
//将三维顶点向二维平面正等测投影
    for (n=0; n<8; n++) {
        //调用投影函数
        pt2d[n] = to2d(temp_pt[n]);
        //设置顶点号码。调试时使用，调试完成后可以去掉
        this["p" add n].num = n;
        //设置顶点位置，使立方体的中心为鼠标指针所在位置
        this["p" add n]._x = pt2d[n][0]+_xmouse;
        this["p" add n]._y = pt2d[n][1]+_ymouse;
    }
}
//旋转函数 rotation(顶点,x 转角,y 转角, z 转角)，如果转角大于 2π，则不转动
//返回旋转之后的顶点数组
function rotation (pt, anglex, angley, anglez) {
    var temppt = []; //临时数组
//如果转角大于 2π，则不转动
    if (anglex>=Math.PI*2) {
        anglex = 0;
    }
    if (angley>=Math.PI*2) {
```

```
            angley = 0;
        }
        if (anglez>=Math.PI*2) {
            anglez = 0;
        }
        temppt.push(pt);
    //分别使用旋转矩阵，开始转换
    //以 x 为轴转动
        if (anglex) {
            temppt[1] = pt[1]*Math.cos(anglex)-pt[2]*Math.sin(anglex);
            temppt[0] = pt[0];
            temppt[2] = pt[1]*Math.sin(anglex)+pt[2]*Math.cos(anglex);
            pt = temppt;
        }
    //以 y 为轴转动
        if (angley) {
            temppt[0] = pt[0]*Math.cos(angley)+pt[2]*Math.sin(angley);
            temppt[1] = pt[1];
            temppt[2] = -pt[0]*Math.sin(angley)+pt[2]*Math.cos(angley);
            pt = temppt;
        }
    //以 z 为轴转动
        if (anglez) {
            temppt[0] = pt[0]*Math.cos(anglez)-pt[1]*Math.sin(anglez);
            temppt[1] = pt[0]*Math.sin(anglez)+pt[1]*Math.cos(anglez);
            temppt[2] = pt[2];
        }
        return temppt;          //返回新的顶点数组
    }
    //正等轴测投影函数，返回二维顶点数组
    function to2d (pt) {
        var temppt=[2];
    //正等轴测投影运算
        temppt[0]=pt[1]*0.707-pt[0]*0.707;
        temppt[1] = -(pt[2]*0.816-pt[0]*0.408-pt[1]*0.408);
        return temppt; //返回二维顶点数组
    }
```

（4）在主场景的第 3 帧处插入空白帧并加入如下脚本：

```
    //调用 calpos()函数，计算并显示顶点
    calpos();
    //改变 x、y、z 三轴的转角，使立方体顶点依次沿 x、y、z 三轴转动
    if (anglex<=Math.PI*2) {
        anglex+=step;
        angley=anglez=0;
```

```
    } else {
        if (angley<=Math.PI*2) {
            angley+=step;
            anglez=0;
        } else {
            if (anglez<=Math.PI*2) {
                anglez+=step;
            } else {
                anglex=0;}
        }
    }
    gotoAndPlay (2);      //跳转到主场景的第 2 帧，以便反复地循环转动
```

（5）输出并运行动画。如图 12-26 所示，鼠标移动到哪里，立方体中心就跟到哪里，而且立方体顶点依次沿 *x*、*y*、*z* 三轴循环转动。

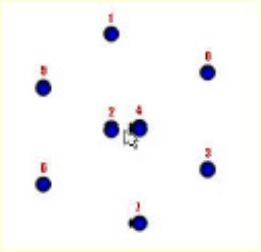


图 12-26　跟随鼠标自转的立方体

- 修改变量 sidelen 可以改变立方体的边长。修改变量 step 可以改变立方体旋转的速度。

（6）调试完毕之后，去掉影片剪辑 point 中的文本区域及其相关的脚本。输出并运行动画。隐藏了顶点编号的旋转立方体如图 12-27 所示。

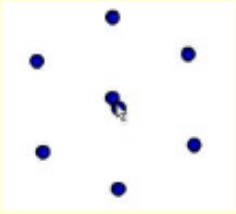


图 12-27　隐藏顶点编号

- 立方体的顶点编号在调试脚本的时候是很有用的，它可以很容易地监视并区分各个点的位置。直到立方体的所有顶点运动正常之后再去掉顶点编号，否则很难调试成功。

12.2.4　总　结

在上述的脚本中，主要使用了矩阵变换对立方体 8 个顶点的运动位置进行计算。其中，立方体 8 个顶点的三维和二维坐标使用了一个二维数组表示，并且定义了几个全局函数来处理顶点的变换。其实如果把这些过程的函数定义到一个对象之中是一种更为直观的办法。下面的脚本就是将一个三维点定义成一个对象并把它保存在 point3d.as 脚本文件中，以便在以后需要的时候可以随时使用#include 动作把它加入到 Flash MX 动画的脚本中。

```
// point3d 对象构造函数
function point3d (x, y, z) {
    this.x = x;
```

```
        this.y = y;
        this.z = z;
    }
    // 旋转函数 rotation(顶点,x 转角,y 转角, z 转角)，如果转角大于 2π，则不转动
    point3d.prototype.rotation = function (anglex, angley, anglez)
    {
        var temppt ={x：this.x, y：this.y, z：this.z};
        if (anglex) {
            temppt.x = this.x;
            temppt.y = this.y*Math.cos(anglex)-this.z*Math.sin(anglex);
            temppt.z = this.y*Math.sin(anglex)+this.z*Math.cos(anglex);
            this.x = temppt.x;
            this.y = temppt.y;
            this.z = temppt.z;
        }
        if (angley) {
            temppt.x = this.x*Math.cos(angley)+this.z*Math.sin(angley);
            temppt.y = this.y;
            temppt.z = -this.x*Math.sin(angley)+this.z*Math.cos(angley);
            this.x = temppt.x;
            this.y = temppt.y;
            this.z = temppt.z;
        }
        if (anglez) {
            temppt.x = this.x*Math.cos(anglez)-this.y*Math.sin(anglez);
            temppt.y = this.x*Math.sin(anglez)+this.y*Math.cos(anglez);
            temppt.z = this.z;
            this.x = temppt.x;
            this.y = temppt.y;
            this.z = temppt.z;
        }
    };

    // 获得正等测投影函数，返回二维顶点
    point3d.prototype.to2d = function ()
    {
        var temppt = new Object();
        temppt.x = this.y*0.707-this.x*0.707;
        temppt.y = -(this.z*0.816-this.x*0.408-this.y*0.408);
        return temppt;
    };
```

针对如上的变动，主场景的第 1 帧中的脚本应改为：

```
    point = [];// 顶点数组
    direction = "x";
```

```
var sidelen = 120;// 立方体边长
anglex = angley=anglez=0;// 沿 x、y、z 三轴转动的角度
step = Math.PI/40;// 立方体每次所转的角度
inicube();// 调用初始化立方体函数
// 创建 8 个立方体顶点
for (i=0; i<8; i++) {
    attachMovie("point", "p"+i, i);
}
// 立方体初始化函数，根据边长设置立方体的 8 个顶点坐标
function inicube () {
    point[0] = new point3d(sidelen/2, sidelen/2, sidelen/2);
    point[1] = new point3d(-sidelen/2, sidelen/2, sidelen/2);
    point[2] = new point3d(-sidelen/2, sidelen/2, -sidelen/2);
    point[3] = new point3d(sidelen/2, sidelen/2, -sidelen/2);
    for (n=0; n<4; n++) {
        point.push(new point3d(point[n].x, -sidelen/2, point[n].z));
    }
}
// 计算并显示立方体转角的函数
function calpos () {
    var pt2d = [];

    // 将三维立方体顶点向二维坐标下投影的坐标数组
    for (n=0; n<8; n++) {
        // 计算转角
        // 调用 point3d.rotation(x 转角,y 转角, z 转角)方法，返回旋转之后的顶点
        if (direction == "x") {
            point[n].rotation(step, 0, 0);
        } else if (direction == "y") {
            point[n].rotation(0, step, 0);
        } else {
            point[n].rotation(0, 0, step);
        }
    }
    // 将三维顶点向二维平面正等测投影
    for (n=0; n<8; n++) {
        // 调用投影方法
        pt2d[n] = point[n].to2d();
        // 设置顶点位置，是立方体的中心为鼠标指针所在位置
        this["p" add n]._x = pt2d[n].x+_xmouse;
        this["p" add n]._y = pt2d[n].y+_ymouse;
    }
}
```

主场景的第 3 帧中的脚本应改为：

```
// 调用 calpos()函数，计算并显示顶点
calpos();
// 改变 x、y、z 三轴的转角，使立方体顶点依次沿 x、y、z 三轴转动
if (anglex<=Math.PI*2 && direction=="x") {
//沿 x 方向旋转
    anglex += step;
    angley = anglez=0;
} else {
    direction="y";
    if (angley<=Math.PI*2 && direction=="y") {
    //沿 y 方向旋转
        angley += step;
        anglex = anglez=0;
    } else {
        direction="z";
        if (anglez<=Math.PI*2 && direction=="z") {
        //沿 z 方向旋转

            anglez += step;
            anglex = angley=0;
        } else {
            direction = "x";
        }
    }
}
gotoAndPlay (2);// 跳转到主场景的第 2 帧，以便反复地循环转动
```

因为调用自定义对象 point3d 中提供的 rotation 方法将改变自身的转角，所以在上述脚本中所使用的转角为相对转角，即相对于前一次转动的角度。而不是像上一个例子一样使用的是绝对转角。这种使用方法使对象本身独立起来不受外界环境的干扰，更有利于 ActionScript 脚本的模块化。在 12.3 节中，仍将使用本节所讲述并给出的 point3d 对象，从而将体现出相对坐标的优越性。

12.3 可以控制转动的立方体

12.3.1 介 绍

在 12.2 节中，虽然使用了 3D 变换矩阵，也实现了立方体顶点的显示。但是，动画的结果终究是几个点的运动，并没有显示出立方体的每一个面，因而无法构成一个真正的空间立方体的正等测投影图。这些问题将在本节中解决。

其实，一个立方体是由 6 个正方形平面围成的封闭几何体。当空间的立方体向二维平面投影时，如图 12-28 所示，每一个平面的正方形就会被投影成平行四边形，也就是当于正方形被错切了。

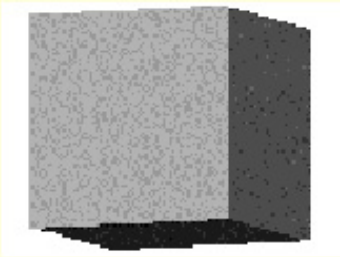


图 12-28　正方形被投影成平行四边形

在 12.2 节中，应用变换矩阵已经可以计算出空间立方体的 8 个顶点的坐标以及它们投影到二维平面上的坐标。接下来的工作就是“贴图”，即把立方体的每个正方形平面图形或者图案经过错切变换以后，使它的 4 个顶点如图 12-29 所示，刚好符合立方体中所对应的 4 个顶点在二维平面上的坐标。

图 12-29　“贴图”

12.3.2　错切的实现

虽然在编辑 Flash MX 动画的时候，如图 12-30 所示，可以对图片或者元件进行定量的错切变换。但是，在 Flash MX 的 ActionScript 脚本中，只为影片剪辑提供了旋转与缩放变换的属性，并没有提供有关错切变换的方法或者属性，所以，在 Flash MX 动画运行的时候，没有办法直接通过 ActionScript 脚本控制平面图形的错切，必须寻找间接的办法对图形进行变换。

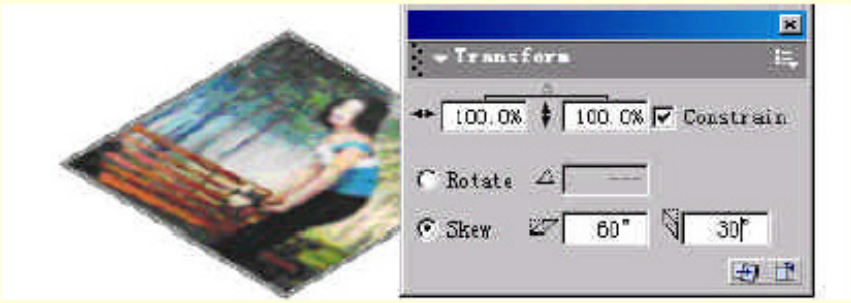


图 12-30　定量的错切变换

在这里介绍一种巧妙过程来实现错切变换，以满足我们的需要：

- 首先，将要变换的图形或者图片添加到一个图形元件中，通过把图形元件的行为设置为影片剪辑 MovieClip，如图 12-31 所示就可以用 ActionScript 脚本实现对图片旋转定量的角度（θ）。
- 然后，将上述图形元件添加到一个新的图形元件中。同样，把图形元件的行为设置为影片剪辑 MovieClip2，如图 12-32 所示就可以用 ActionScript 脚本定量的对图片进行横向和纵向两个方向进行缩放（Scalex、Scaley）和旋转角度（Φ）。
- 最后，根据已知的平行四边形（投影之后的立方体的一个面）的 4 个顶点坐标，计算两次转动所需的角度 θ、Φ，以及横向和纵向两个方向进行缩放的比例 a、d。

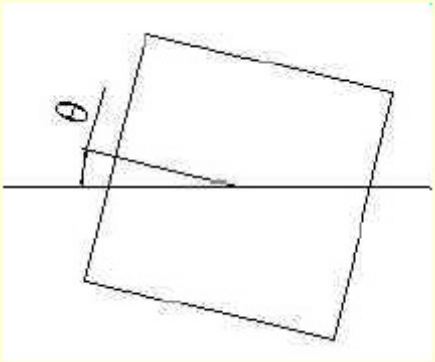
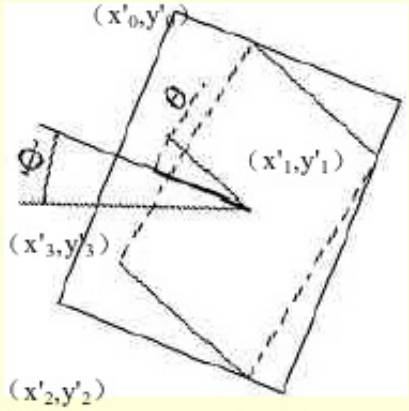


图 12-31　图片旋转定量的角度 θ



错切以后的图

图 12-32　对图片进行缩放（Scale）和旋转角度（Φ）

具体的计算方法如下：

根据平面图形理论，平面内任意一点[x　y]可以用三维向量[x　y　1]表示，则它的旋转变换矩阵为：

$$T=\begin{bmatrix}\cos a & \sin a & 0\\ -\sin a & \cos a & 0\\ 0 & 0 & 1\end{bmatrix}$$

- α 表示顺时针转角

比例变换矩阵为：

$$T=\begin{bmatrix}a & 0 & 0\\ 0 & d & 0\\ 0 & 0 & 1\end{bmatrix}$$

- a、d 分别表示沿 x、y 轴方向上的缩放比例

使平面正方形上任意一点（x,y,1）经过上述步骤变换之后的位置为（x',y',1）,则上述变换过程可以表示为：

$$[x' \quad y' \quad 1]=[x \quad y \quad 1]\begin{bmatrix}\cos q & \sin q & 0\\ -\sin q & \cos q & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\cos f & \sin f & 0\\ -\sin f & \cos f & 0\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}a & 0 & 0\\ 0 & d & 0\\ 0 & 0 & 1\end{bmatrix}$$

所以有：

$$\begin{cases} x' = ax\cos q\cos f - ay\sin q\cos f - dx\sin q\sin f - dy\cos q\sin f \\ y' = ax\cos q\sin f - ay\sin q\sin f + dx\sin q\cos f + dy\cos q\cos f \end{cases}$$

设立方体一个面的 4 个顶点按图 12-33 的顺序定义：

图 12-33　顶点顺序

则：

$$\begin{cases} x'_0 = ax_0\cos q\cos f - ay_0\sin q\cos f - dx_0\sin q\sin f - dy_0\cos q\sin f & (1) \\ y'_0 = ax_0\cos q\sin f - ay_0\sin q\sin f + dx_0\sin q\cos f + dy_0\cos q\cos f & (2) \end{cases}$$

$$\begin{cases} x'_1 = ax_1\cos q\cos f - ay_1\sin q\cos f - dx_1\sin q\sin f - dy_1\cos q\sin f & (3) \\ y'_1 = ax_1\cos q\sin f - ay_1\sin q\sin f + dx_1\sin q\cos f + dy_1\cos q\cos f & (4) \end{cases}$$

设正方形的边长为 l，且：

$$\begin{cases} x_0 - x_1 = -l \\ y_0 - y_1 = 0 \end{cases} \qquad \begin{cases} x_0 - x_3 = 0 \\ y_0 - y_3 = -l \end{cases}$$

所以(1)-(3)、(2)-(4)**得：**

$$\begin{cases} x'_0 - x'_1 = -al\cos q\cos a + dl\sin q\sin a & (5) \\ y'_0 - y'_1 = -al\cos q\sin a - dl\sin q\cos a & (6) \end{cases}$$

同理有：

$$\begin{cases} x'_0 - x'_3 = al\sin q\cos f + dl\cos q\sin f & (7) \\ y'_0 - y'_3 = al\sin q\sin f - dl\cos q\cos f & (8) \end{cases}$$

(5)+(8)、(6)+(7)**得：**

$$\begin{cases} x'_0 - x'_1 + y'_0 - y'_3 = -al\cos(q+f) - dl\cos(q+f) & (9) \\ x'_0 - x'_3 + y'_0 - y'_1 = al\sin(q-f) - dl\sin(q-f) & (10) \end{cases}$$

(5)-(8)、(6)-(7)**得：**

$$\begin{cases} x'_0 - x'_1 - y'_0 + y'_3 = -al\cos(q-f) + dl\cos(q-f) & (11) \\ -x'_0 + x'_3 + y'_0 - y'_1 = -al\sin(q+f) - dl\sin(q+f) & (12) \end{cases}$$

(12)/(9)、(10)/(11)**得：**

$$\frac{-x'_0 + x'_3 + y'_0 - y'_1}{x'_0 - x'_1 + y'_0 - y'_3} = tg(q+f) \Rightarrow q+f = arctg\frac{-x'_0 + x'_3 + y'_0 - y'_1}{x'_0 - x'_1 + y'_0 - y'_3} \quad (13)$$

$$\frac{x'_0 - x'_3 + y'_0 - y'_1}{-x'_0 + x'_1 + y'_0 - y'_3} = tg(q-f) \Rightarrow q-f = arctg\frac{x'_0 - x'_3 + y'_0 - y'_1}{-x'_0 + x'_1 + y'_0 - y'_3} \quad (14)$$

所以：

$$q=\frac{1}{2}(arctg\frac{-x'_0+x'_3+y'_0-y'_1}{x'_0-x'_1+y'_0-y'_3}+arctg\frac{x'_0-x'_3+y'_0-y'_1}{-x'_0+x'_1+y'_0-y'_3})$$

$$f=\frac{1}{2}(arctg\frac{-x'_0+x'_3+y'_0-y'_1}{x'_0-x'_1+y'_0-y'_3}-arctg\frac{x'_0-x'_3+y'_0-y'_1}{-x'_0+x'_1+y'_0-y'_3})$$

由(9)、(10)得：

$$\begin{cases}\dfrac{-x'_0+x'_1-y'_0+y'_3}{l\cos(q+f)}=a+d\ldots(15)\\ \dfrac{x'_0-x'_3+y'_0-y'_1}{l\sin(q-f)}=a-d\ldots(16)\end{cases}$$

(15)+(16)、(15)-(16)得：

$$a=\frac{-x'_0+x'_1-y'_0+y'_3}{2l\cos(q+f)}+\frac{x'_0-x'_3+y'_0-y'_1}{2l\sin(q-f)}$$

$$d=\frac{-x'_0+x'_1-y'_0+y'_3}{2l\cos(q+f)}-\frac{x'_0-x'_3+y'_0-y'_1}{2l\sin(q-f)}$$

现在所有的参数都可以由已知的平面二维坐标（x',y'）所表示出来。也就是说按要求实现了图形的错切。在 12.4.4 节中将以此变换方法为基础，对图形的错切进行计算。

12.3.3 平面的消隐

在二维坐标系下显示三维空间图形的时候，还将遇到一个问题，即“消隐”。所谓“消隐”，就是当把错切变换好的图形放置到二维平面对应的位置上时，应该让计算机计算出哪些位置的平面看得见，哪些位置的平面看不见；或者说哪些位置的平面在前面，哪些位置的平面在后面。接下来就介绍一种消隐算法。

如图 12-34 所示，设在右手的卡尔坐标系中有一已知平面 F，点 P_0（x_0, y_0, z_0）、P_1（x_1, y_1, z_1）、P_2（x_2, y_2, z_2）是在平面 F 内的不共线的三点，则 $P_0P_1P_2$ 是平面 F 内的有向线段（向量），求此向量的叉积便可得到平面 F 的法向量 N。

因为：

$$\mathbf{P_0P_1}=(x_1-x_0)i+(y_1-y_0)j+(z_1-z_0)k$$

$$\mathbf{P_0P_2}=(x_2-x_0)i+(y_2-y_0)j+(z_2-z_0)k$$

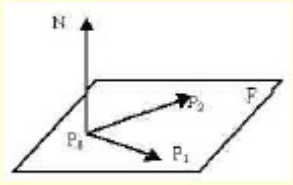


图 12-34 平面的向量

故：

$$\mathbf{N}=\mathbf{P_0P_1}\times\mathbf{P_0P_2}=\begin{vmatrix}i & j & k\\ x_1-x_0 & y_1-y_0 & z_1-z_0\\ x_2-x_0 & y_2-y_0 & z_2-z_0\end{vmatrix}=Ai+Bj+Ck$$

式中的 i、j、k 为空间三轴方向上的单位向量。

所以：

$$B=\begin{vmatrix} z_1-z_0 & x_1-x_0 \\ z_2-z_0 & x_2-x_0 \end{vmatrix}=(x_2-x_0)(z_1-z_0)-(x_1-x_0)(z_2-z_0)$$

所以，外平面的外表面法向量 N 与 y 轴的夹角 β 的余弦值可以由下式求得：

$$\cos b=\frac{B}{|\mathbf{N}|}$$

上式中，|N|表示为法向量 N 的模，它恒为正值，所以 cos β 的符号就可以由 B 值的符号来决定。即比较$(x_2-x_0)(z_1-z_0)$与$(x_1-x_0)(z_2-z_0)$之间的大小，就可以判断面的方向，从而判断面的可见性。具体的判断方法如下：

- 当$(x_2-x_0)(z_1-z_0)>(x_1-x_0)(z_2-z_0)$时，cos β＞0，0°≤β＜90°，此外法线所代表的面不可见。
- 当$(x_2-x_0)(z_1-z_0)\leq(x_1-x_0)(z_2-z_0)$时，cos β≤0，90°≤β≤180°，此外法线所代表的面潜在可见。

因此，在用 ActionScript 脚本脚本显示图片时，只需要隐藏不可见面（B＜0 的面），显示潜在可见面（B＞0 的面）即可。

✍ 在这里，为了简化问题，仍然以空间的立方体为例，所以只考虑空间的凸多面体，对于任意多面体的消隐问题请读者参考有关方面的书籍。

12.3.4 实　战

按照 12.3.2、12.3.3 两节中所讲述的方法，完全可以实现对一个已知顶点坐标的空间立方体进行贴图操作。在下面的例子中将利用上述知识，实现一个具有表面贴图的空间立方体的自由旋转。

（1）新建一个 Flash MX 文件。如图 12-35 所示，从外部导入事先制作好的 6 张正方形图片，作为立方体的 6 个面。同时调整图片至合适的大小。

图 12-35　导入图片

✍ 图片的大小必须相同。如果导入的图片大小不同，则使用排列面板将它们设置成相同的尺寸。

（2）选中一个图片，使用 Insert | Convert to Symbol 菜单命令把此图片转换成一个图形元件，并命名为 side_gra1。

（3）双击创建的图形元件，如图 12-36 所示进入元件的编辑状态。再次用 Insert | Convert to Symbol 菜单命令把元件内部的图片转换成一个新图形元件，并命名为 side_gra1_in。

图 12-36　再次转换成一个新图形元件

（4）如图 12-37 所示，在第一层图形元件 side_gra1 中，改变其内部的图形元件 side_gra1 _in 的作用为影片剪辑，并命名为 pic。

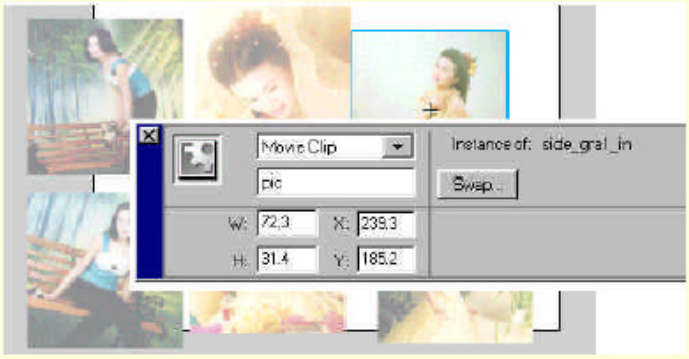


图 12-37　改变图形元件的作用

（5）回到主场景中，重复步骤（2）~（4），把所有的图片都改变成有两层元件嵌套组成的图形元件。外层的元件命名为 side_gra?；内层的元件命名为 side_gra?_in，同时将其作用都改为影片剪辑（Movie Clip），并命名为 pic。

（6）在主场景中将转化为图形元件的图片按如图 12-38 所示的位置进行排列，排列的结果就像是一个被展开的立方体盒子。

- 因为立方体是由 ActionScript 脚本创建出来的，所以它并不依赖于编辑时候图片之间的位置关系。但是按照上述位置排列，很容易想象出图片在空间中（立方体上）的位置，有利于 ActionScript 脚本的调试。

图 12-38　排列元件

（7）如图 12-39 所示，分别改变图形元件的作用为 Movie Clip（影片剪辑），并按照被展开的立方体盒子各个面的位置关系，为它们命名为“ front ”、“ left ”、“ top ”、“ right ”、“ bottom ”、“ back ”（“ 前面 ”、“ 左面 ”、“ 上面 ”、“ 右面 ”、“ 下面 ”、“ 后面 ”）。

图 12-39　命名各个面

（8）新建一个影片剪辑，命名为 hand。如图 12-40 所示，在影片剪辑的第 1 帧上，使用铅笔工具绘制出一个手。

（9）在影片剪辑的第 2 帧处插入空白帧，并使用铅笔工具绘制出一个如图 12-41 所示的握紧的手。

图 12-40　绘制出一个手的形状

图 12-41　握紧的手

（10）在影片剪辑 hand 的两帧上都加入如下脚本：

```
stop ();
```

（11）返回主场景，插入一个新层 Layer2，如图 12-42 所示，在第 1 帧中放入手的影片剪辑，并命名为 hand。

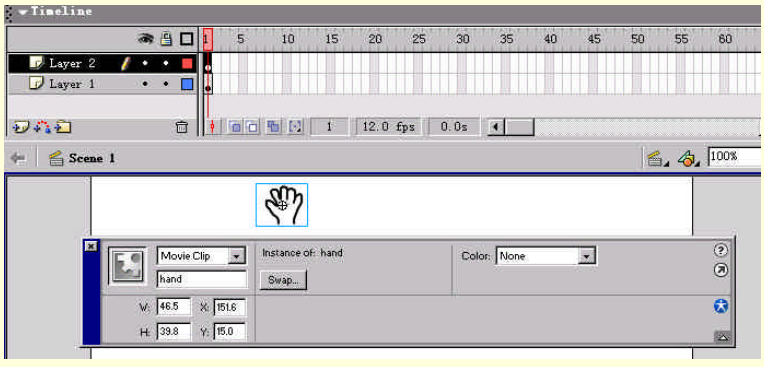


图 12-42　放入并设置影片剪辑

（12）在主场景的第 1 帧中加入如下脚本：

```
#include "point3d.as"        //同 12.2 节中所给出的脚本
stop ();     //停止影片的播放，防止反复运行此脚本
hand._visible=false;  //隐藏“ 手 ”
point = []; //空间顶点数组
```

```
pt2d = []; //空间顶点数组的平面投影点
var sidelen = front._width;        // 立方体边长
//设置正方体的中心与正方体前面的相同
xpos = front._x;
ypos = front._y;
anglex = angley=anglez=0;// 沿 x、y、z 三轴所旋转的相对角度
// 调用初始化立方体函数
inicube();
//计算并显示贴图结果
calpos();
// 立方体初始化函数，根据边长设置立方体的 8 个顶点坐标
function inicube () {
    point[0] = new point3d(sidelen/2, sidelen/2, sidelen/2);
    point[1] = new point3d(-sidelen/2, sidelen/2, sidelen/2);
    point[2] = new point3d(-sidelen/2, sidelen/2, -sidelen/2);
    point[3] = new point3d(sidelen/2, sidelen/2, -sidelen/2);
    for (n=0; n<4; n++) {
        point.push(new point3d(point[n].x, -sidelen/2, point[n].z));
    }
}
// 计算立方体转角并显示贴图的函数
function calpos () {
    // 将三维立方体顶点向二维坐标下投影的坐标数组
    for (n=0; n<8; n++) {
        // 计算转角
        // 调用 rotation(x 转角,y 转角, z 转角)函数
        point[n].rotation(anglex, angley, anglez);
    }
    // 将三维顶点向二维平面正等测投影
    for (n=0; n<8; n++) {
        // 调用投影函数
        pt2d[n] = point[n].to2d();
    }
//对立方体的六个面进行贴图
    setpic(0, 3, 1, "front");
    setpic(4, 7, 0, "left");
    setpic(4, 0, 5, "top");
    setpic(1, 2, 5, "right");
    setpic(3, 7, 2, "bottom");
    setpic(7, 4, 6, "back");
}
//对立方体的六个面进行贴图函数。Setpic(点 0，点 1，点 3,所对应的面)
function setpic (pt0num, pt1num, pt3num, side) {
    //见提示 1
```

```
        pt0 = pt2d[pt0num];
        pt1 = pt2d[pt3num];
        pt3 = pt2d[pt1num];
        //按照 12.2 节所给出的错切公式进行计算
        temp_ang1 = Math.atan2(-pt0.x+pt3.x+pt0.y-pt1.y, pt0.x-pt1.x+pt0.y-pt3.y);
        temp_ang2 = Math.atan2(pt0.x-pt3.x+pt0.y-pt1.y, -pt0.x+pt1.x+pt0.y-pt3.y);
        rot_ang1 = (temp_ang1+temp_ang2)/2;      //θ角
        rot_ang2 = (temp_ang1-temp_ang2)/2;      //φ角
        xscale = 100;    //横向缩放比例 a
        yscale = 100;    //纵向缩放比例 d
        if (Math.sin(rot_ang1-rot_ang2) != 0) {//分母不为零
            //计算 a、d
            xay = (-pt0.x+pt1.x-pt0.y+pt3.y)/Math.cos(rot_ang1+rot_ang2)/sidelen*100;
            xdy = (pt0.x-pt3.x+pt0.y-pt1.y)/Math.sin(rot_ang1-rot_ang2)/sidelen*100;
            xscale = (xay+xdy)/2;        //横向缩放比例 a
            yscale = (xay-xdy)/2;        //纵向缩放比例 d
        }
        //计算错切以后的图形中心位置
        tempx = xpos+(pt1.x+pt3.x)/2;
        tempy = ypos+(pt1.y+pt3.y)/2;
    //按照计算结果设置图片（对正方体表面贴图）
        //设置图形中心位置
        this[side]._x = tempx;
        this[side]._y = tempy;
        //内部旋转θ角
        this[side].pic._rotation = rot_ang1*180/Math.PI;
        //设置外部图形横纵的比例
        this[side]._xscale = xscale;
        this[side]._yscale = yscale;
        //外部旋转φ角
        this[side]._rotation = rot_ang2*180/Math.PI;
    //按照 12.2 节所给出的公式，判断消隐
        //设置空间点，见提示 2
        pt0_3d = point[pt0num].to2d();
        pt1_3d = point[pt1num].to2d();
        pt3_3d = point[pt3num].to2d();
        //计算 B 的值
        var b = (pt3_3d.x-pt0_3d.x)*(pt1_3d.y-pt0_3d.y)-
                        (pt1_3d.x-pt0_3d.x)*(pt3_3d.y-pt0_3d.y);
        //判断消隐
        if (b<=0) {
            this[side]._visible = false;
        } else {
            this[side]._visible = true;
```

```
    }
}
```

✍ **提示 1：**

脚本：

```
pt1 = pt2d[pt3num];
pt3 = pt2d[pt1num];
```

把第 3 点和第 1 点交换位置，是因为 pt?表示错切变形以后的正方形的顶点，而 12.2 节中计算错切时，是按照顺时针的方向设置坐标的，它与判断消隐时的坐标方向刚好相反，所以要交换。

✍ **提示 2：**

脚本：

```
pt0_3d = point[pt0num].to2d();
pt1_3d = point[pt1num].to2d();
pt3_3d = point[pt3num].to2d();
```

判断消隐时所使用的 B 应该是在正等轴测旋转变换之后计算出来的。正等轴测的变换矩阵为：

$$T = \begin{bmatrix} \cos q & \sin q & 0 & 0 \\ -\sin q & \cos q & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(-j) & \sin(-j) & 0 \\ 0 & -\sin(-j) & \cos(-j) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos q & \sin q \cos j & -\sin q \sin j & 0 \\ -\sin q & \cos q \sin j & -\cos q \sin j & 0 \\ 0 & 0 & \cos j & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

因为在 B 的计算中只需要 x 和 z，与 y 无关，所以它刚好和向 xOz 平面投影的结果相同。所以可以直接调用正等轴测投影的函数。

（13）随意选中一个主场景中的影片剪辑（如“ front ”），为它添加如下脚本：

```
onClipEvent (mouseDown) {
    _root.hand.gotoAndStop(2);      //变换手型
    mouse_down = true;
    md_pt={x：_root._xmouse,y：_root._ymouse};        //获取按下鼠标时的坐标
}
onClipEvent (load) {
    Mouse.hide();   //隐藏鼠标指针
    startDrag (_root.hand, true);            //启动手作为鼠标指针
    //获得 Flash MX 动画的中心坐标
    cen_pt={x：_root._width/2,y：_root._height/2};
    mouse_down=false;
}
onClipEvent (mouseUp) {
```

```
        _root.hand.gotoAndStop(1);      //变换手型
        mouse_down = false;
    }
    onClipEvent (mouseMove) {
        //判断鼠标是否在动画之内
        if (_root.hitTest(_root._xmouse,_root._ymouse,false)) {
            //鼠标在动画之内，显示“手”
            _root.hand._visible=true;
        } else {
            //鼠标不在动画之内，隐藏“手”
            _root.hand._visible=false;
        }
        if (mouse_down==false) {
            _root.hand.gotoAndStop(1);
            return;
        }
        //获取当前鼠标的坐标
        var md_pt1={x：_root._xmouse,y：_root._ymouse};
        //以影片中心点为原点，变换当前鼠标的坐标
        tmp_pt={x：md_pt1.x-cen_pt.x,y：md_pt1.y-cen_pt.y};
        //计算转动的半径
        var rad=Math.sqrt (_root.sidelen*_root.sidelen
                                +tmp_pt.x*tmp_pt.x+tmp_pt.y*tmp_pt.y);
    //计算转角
        _root.anglez=(md_pt1.x-md_pt.x)/rad;
    //当在动画的左半部垂直拖动鼠标时，立方体沿 y 轴转动
        if (tmp_pt.x<=0) {
            _root.angley=(md_pt1.y-md_pt.y)/rad;
        } else {//当在动画的右半部垂直拖动鼠标时，立方体沿 x 轴转动
            _root.anglex=-(md_pt1.y-md_pt.y)/rad;
        }
    //计算并显示转动以后的结果
        _root.calpos();
    //保存当前鼠标坐标，以便下一次作为参考
        md_pt=md_pt1;
    }
```

（14）输出并运行动画。如图 12-43 所示，平面的正方形经过错切变换后将被贴到相应的位置，从而组成了空间的立方体。使用鼠标拖动可以转动立方体。

图 12-43　运行结果

12.3.5　总　结

在 12.3、12.4 两节中，通过使用计算机绘图方面的知识，实现了在 Flash MX 中显示三维物体的功能。在本节中，通过双层嵌套的元件，巧妙地实现了平面正方形的错切变换，从而圆满解决了空间立方体的贴图问题。利用类似的方法可以创造出其他空间多面体的 Flash MX 动画。

上述方法需要编写大量的 ActionScript 脚本，对编程水平要求较高，因此不适合于初学者。

12.4　制作自己的 MTV

12.4.1　介　绍

Flash 最初是用来制作网页动画的，但随着 Flash 版本的不断更新，其功能也逐渐增强。它的易用性以及制作动画的高效性使得它可以被用来制作电视和电影的动画、游戏的片头和片尾等。尤其是用 Flash 制作自己的 MTV 已经被广泛地应用。用 Flash 制作的 MTV 都很小（1MB 左右），适合在网上的传播。另外，用 Flash 制作 MTV 的一大优点是用户可以把自己的照片扫描到计算机内，制作出以自己为主题的 MTV。

其实，MTV 是一个顺序播放的动画，不涉及到任何的交互功能（当然也可以在 MTV 增加简单的交互）所以，表面上看它没有交互动画那么复杂，通过简单的 Flash MX 的渐变功能也可以制作出完整的 MTV。由于一首好的 MTV 是多种 Flash MX 动画特效的综合表现，因此只靠简单的 Flash MX 渐变是无法实现的，此外还需要使用 ActionScript 脚本来控制动画。

12.4.2　实　战

介绍一个 MTV，它应用了很典型的 Flash MX 动画特效，供读者在制作 MTV 时参考。读者应该注重制作的过程，不必一步步地照搬。

（1）新建一个 Flash MX 文件，把背景设置为黑色。将事先选好的图片导入到文件中。如图 12-44 所示，挑选两张图片分别到插入主场景的第 1 帧的两层中，将它们转化为图形元件。添加向导层，使它引导两个原有的层，并绘制出两条轨道。将图片缩小之后放置在轨道的一端。

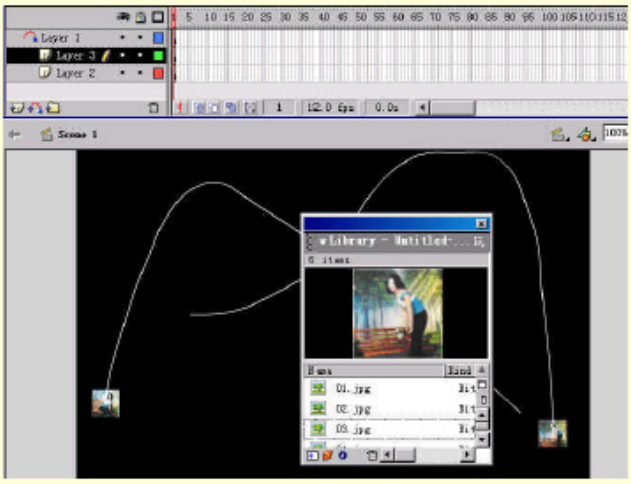


图 12-44　入场动画

✎ 因为一个 MTV 的时间由所选的歌曲决定，一般是 3~4 分钟左右（180~240 秒），所以为了编辑的方便，可以将时间轴的显示模式设置成 tiny 模式。

（2）如图 12-45 所示在图片所在两层的第 90 帧处插入关键帧，并在向导层的第 60 帧处插入帧。移动并放大图片至轨道的另一端，对图片层使用位移渐变。

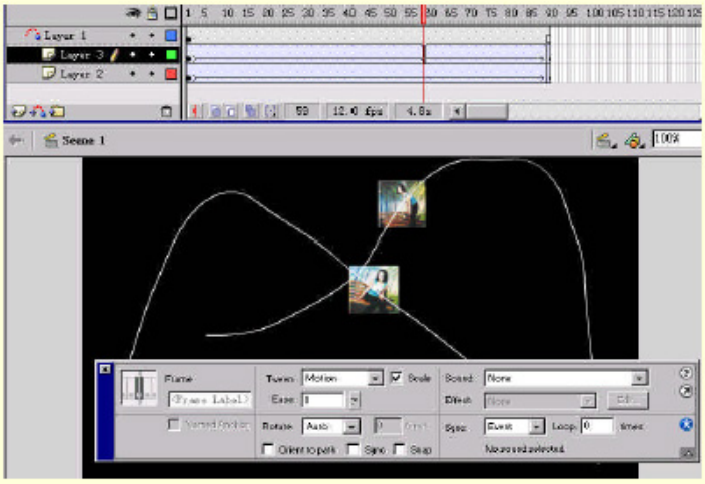


图 12-45　使用位移渐变

（3）分别在两个图形所在层的第 120 帧处插入关键帧，并改变图形元件的透明度为 0，如图 12-46 所示同样使用位移渐变，使图片渐隐。

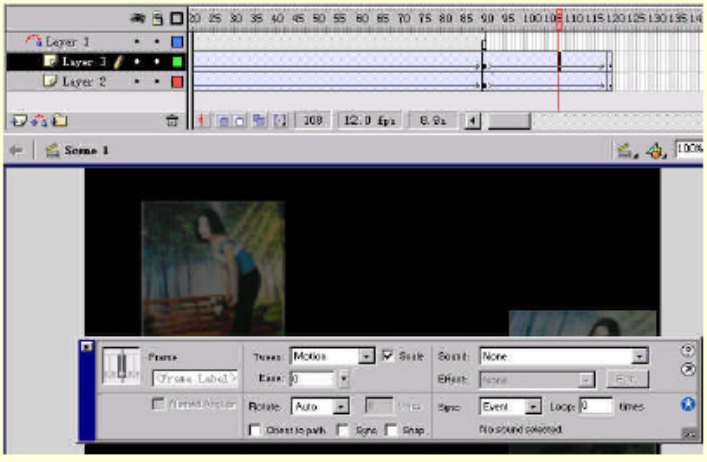


图 12-46　渐隐图片

（4）插入一个新层 Layer4（不被向导层所作用），在第 100 帧处插入空白帧，并加入一个新的图片。把此图片放置在场景的正中间，同时缩小图片。此时的时间轴如图 12-47 所示。

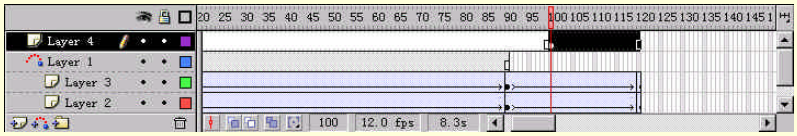


图 12-47　加入新层

（5）如图 12-48 所示在 Layer4 的第 130 帧处插入关键帧，把图片放大，同时使用位移渐变。

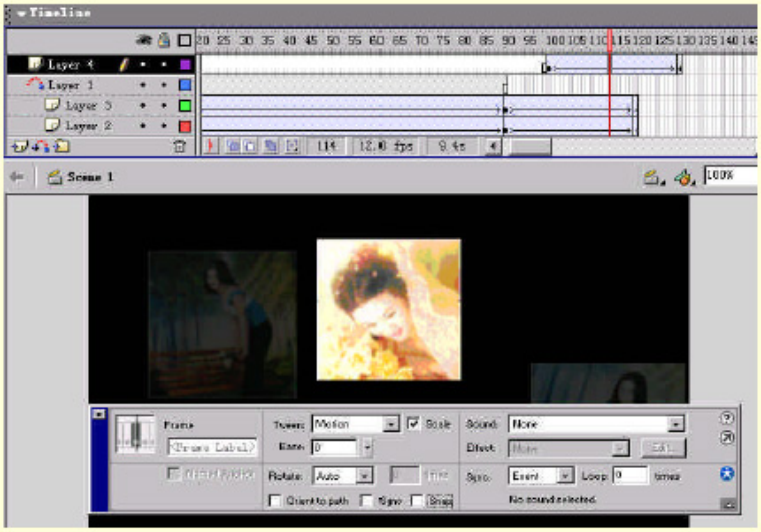


图 12-48　放大图片

（6）在 Layer4 的第 131 帧处插入关键帧。在此帧上选中图片，并将其转换为图形元件。双击图形元件，进入元件的编辑状态。如图 12-49 所示选中图形使用 Modify | Break Apart 菜单命令将图形分解。

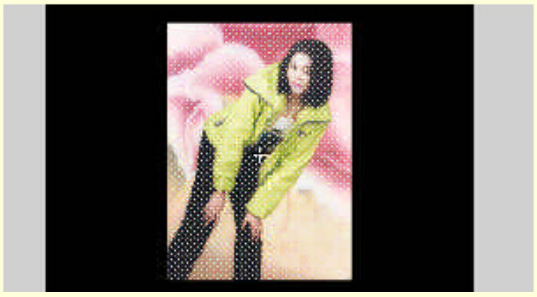


图 12-49　分解图片

（7）如图 12-50 所示，用箭头工具选中图片的下三分之一处。使用 Cut 命令删除选中部分；再使用 Paste to new layer（粘贴到一个新层）命令，将图片分为上下两个部分，并把图片的下半部移回原处。

图 12-50　分为上下两个部分

（8）与上述过程相似，将图形的上半部分也分开到两个层中并移回原处。这时图形将被分解成上、中、下 3 个部分，它们分别处于 3 层中。

（9）如图 12-51 所示，纵向选中图片，并将它们分开。这时图形已经被分解成了 9 部分。

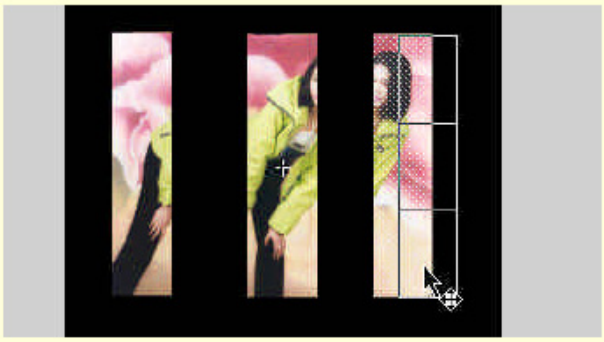


图 12-51　纵向分解图片

（10）把上述分解的图片的 9 个部分分别放在 9 个层中，将 9 块图片重新组合。这时的时间轴如图 12-52 所示，组合后的图片如图 12-53 所示。

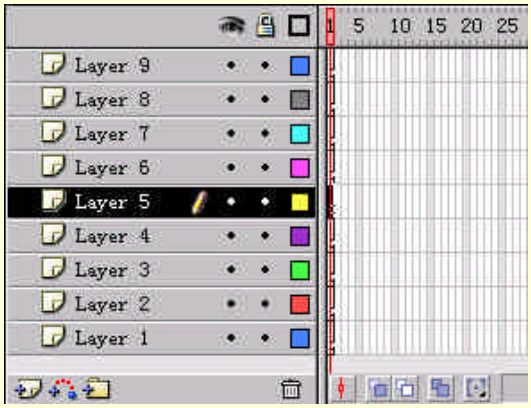


图 12-52　9 层的时间轴

图 12-53　组合后的图片

（11）分别在每一层的第 60 帧处创建关键帧。如图 12-54 所示，随意改变每一块图形的边缘。最后得到如图 12-55 所示的破碎图形。

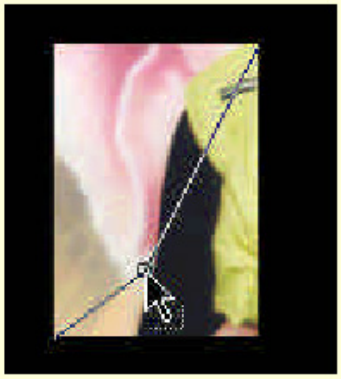


图 12-54　修改图形边缘

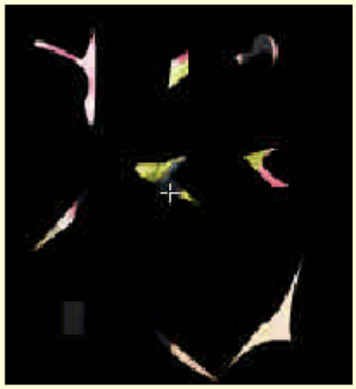


图 12-55　破碎的图形

（12）对每一帧都使用图形渐变，将得到一个如图 12-56 所示的展示图片的破碎过程的动画。

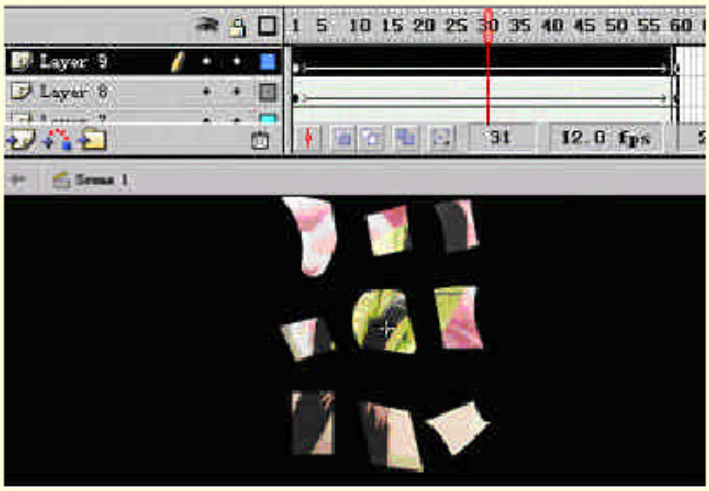


图 12-56　图片的破碎过程

（13）返回主场景，如图 12-57 所示在 Layer4 层的第 190 帧处插入帧。使图形元件自动播放至此。

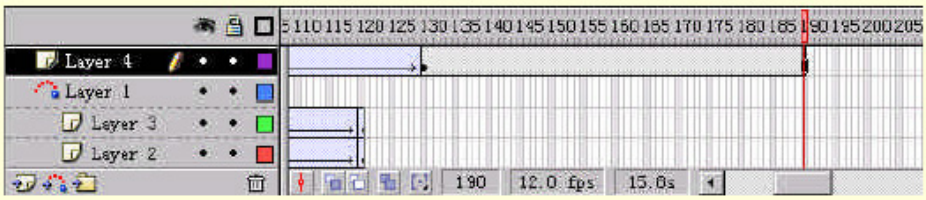


图 12-57　Layer4 层第 190 帧处插入帧

（14）如图 12-58 所示，在 Layer3 层的第 121 和 130 帧处插入空白帧，并在第 130 帧处加入新的图片。

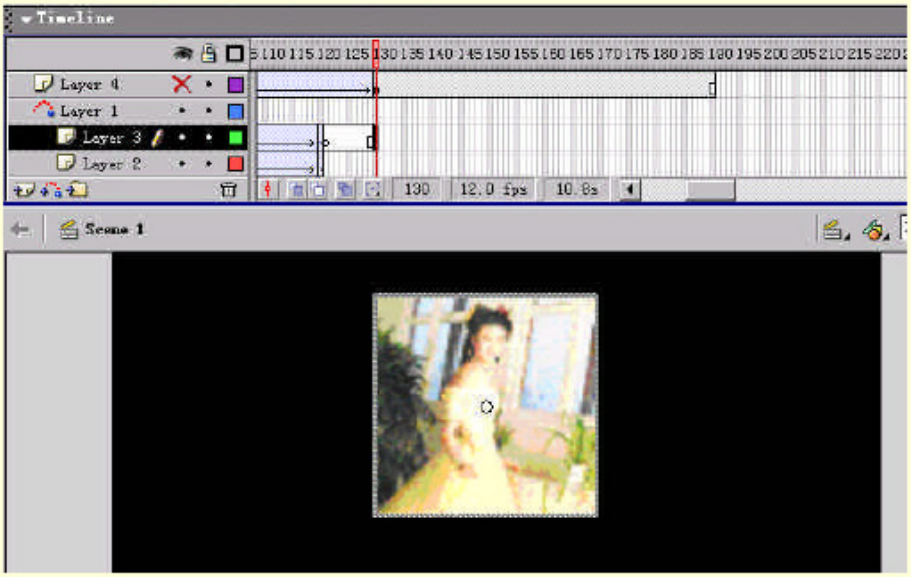


图 12-58　加入图片

（15）在 Layer3 层的第 200 帧处插入关键帧，使用 Insert | Convert to Symbol 菜单命令将图片转化为图形元件。此时的时间轴如图 12-59 所示。

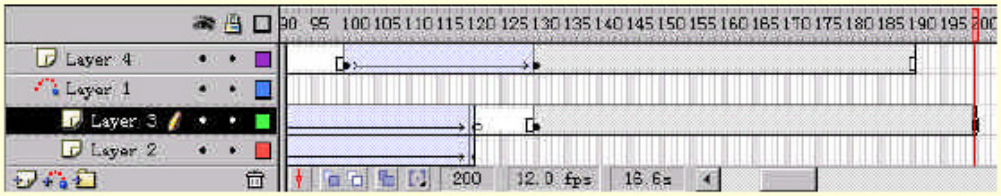


图 12-59　步骤（15）的时间轴

（16）双击 Layer3 层的第 200 帧处的图形元件，进入元件的编辑状态。如图 12-60 所示，在当前层上插入一个新层 Layer2，并在此层上绘制一个圆，将下一层的图片遮住。

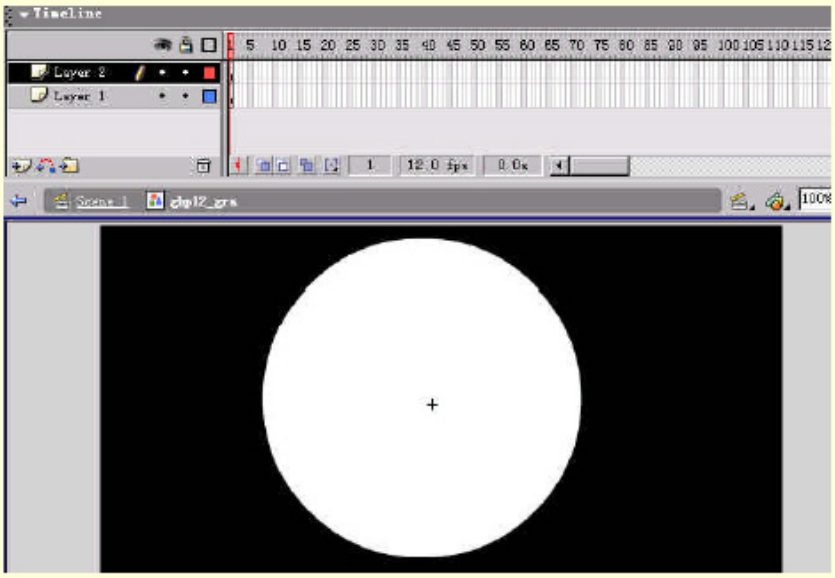


图 12-60　编辑图形元件

（17）如图 12-61 所示在元件的编辑模式中，选中圆区域，并将它组合。在 Layer2 层的第 60 帧中插入关键帧，将此帧中的圆缩小，并使用位移渐变。同时，在 Layer1 层的第 60 帧处插入帧。

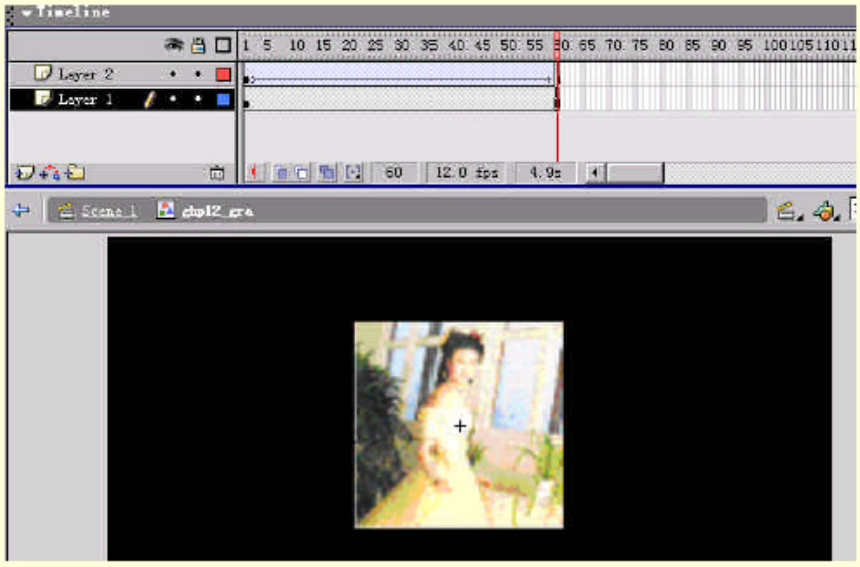


图 12-61　缩小圆

（18）如图 12-62 所示，在 Layer2 层上使用遮照，产生圆形收缩的动画。

（19）返回主场景在 Layer3 层的第 259 帧处插入帧。在 Layer2 的第 121 帧和第 200 帧处插入空白

帧。然后在第 200 帧处添加一个新图片，并将其转换为图形元件。最后在第 260 帧和第 320 帧处插入关键帧，同时调整第 320 帧图片的透明度为 0，使用位置渐变使其渐隐。

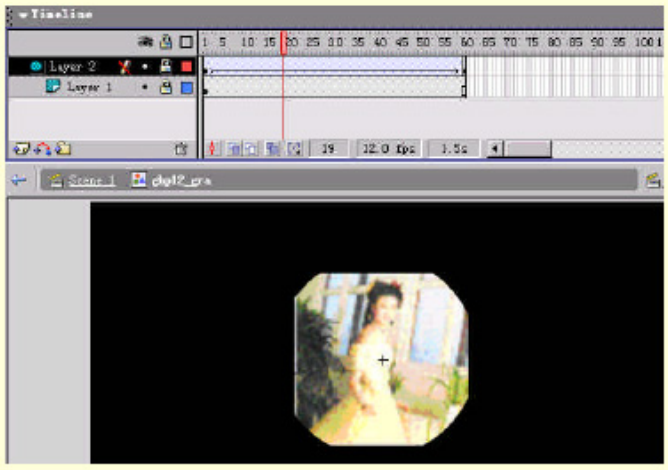


图 12-62　Layer2 层上使用遮照

（20）移动 Layer2 层的第 260~320 帧的动画至 Layer3 的对应位置。这时的时间轴如图 12-63 所示。

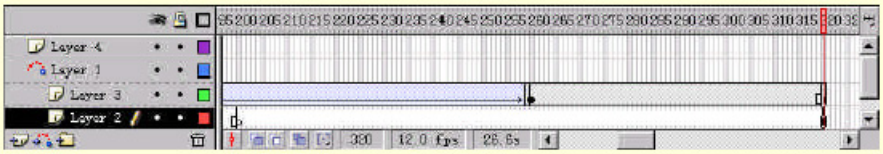


图 12-63　步骤（20）的时间轴

（21）创建一个影片剪辑，命名为 cube。如图 12-64 所示，把 12.4 节中旋转立方体的主场景的第 1 帧复制到此影片剪之中。删除其中 front 影片剪辑的所有 ActionScript 脚本。

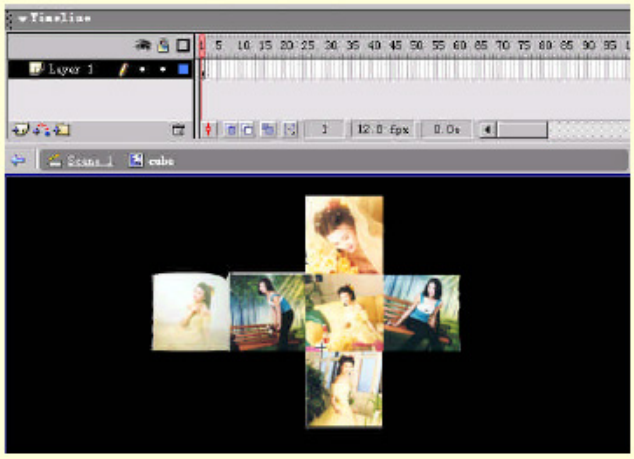


图 12-64　创建影片剪辑 cube

（22）在影片剪辑 cube 中添加一个新层 Layer2，在此层的第 4 帧插入空白帧。并加入如下脚本：

```
calpos();
if (angle<=Math.PI*2) {
      angle += step;
```

```
} else {
    angle = 0;
    if (direction=="z") {
        bfinish=true
        stop ();
        return;
    }
    if (direction=="y") {
        direction="z";
        anglez = step;
        anglex = angley=0;
    }
    if (direction=="x") {
        direction="y";
        angley = step;
        anglex = anglez=0;
    }
}
gotoAndPlay (2);
```

（23）在 Layer1 层的第 4 帧处插入帧，并修改 Layer1 层的帧脚本如下：

```
#include "point3d.as"
stop ();
bfinish=false;
point = [];// 顶点数组
pt2d = [];
direction = "x";
step=Math.PI/40;//每次旋转角度
// 立方体边长
var sidelen = front._width;
pos = {x：front._x,y：front._y};
anglex = angley=anglez=0;// 沿 x、y、z 三轴所旋转的向对角度
// 调用初始化立方体函数
// 立方体初始化函数，根据边长设置立方体的 8 个顶点坐标
inicube();
calpos();
anglex = step;
function inicube () {
    point[0] = new point3d(sidelen/2, sidelen/2, sidelen/2);
    point[1] = new point3d(-sidelen/2, sidelen/2, sidelen/2);
    point[2] = new point3d(-sidelen/2, sidelen/2, -sidelen/2);
    point[3] = new point3d(sidelen/2, sidelen/2, -sidelen/2);
    for (n=0; n<4; n++) {
        point.push(new point3d(point[n].x, -sidelen/2, point[n].z));
    }
```

```
}
// 计算立方体转角的函数
function calpos () {
    // 将三维立方体顶点向二维坐标下投影的坐标数组
    for (n=0; n<8; n++) {
        // 计算转角
        // 调用 rotation(顶点,x 转角,y 转角, z 转角)函数，返回旋转之后的顶点
        point[n].rotation(anglex, angley, anglez);
    }
    // 将三维顶点向二维平面正等测投影
    for (n=0; n<8; n++) {
        // 调用投影函数
        pt2d[n] = point[n].to2d();
    }
    setpic(0, 3, 1, "front");
    setpic(4, 7, 0, "left");
    setpic(4, 0, 5, "top");
    setpic(1, 2, 5, "right");
    setpic(3, 7, 2, "bottom");
    setpic(7, 4, 6, "back");
}
function setpic (pt0num, pt1num, pt3num, side) {
    pt0 = pt2d[pt0num];
    pt1 = pt2d[pt3num];
    pt3 = pt2d[pt1num];
    temp_ang1 = Math.atan2(-pt0.x+pt3.x+pt0.y-pt1.y, pt0.x-pt1.x+pt0.y-pt3.y);
    temp_ang2 = Math.atan2(pt0.x-pt3.x+pt0.y-pt1.y, -pt0.x+pt1.x+pt0.y-pt3.y);
    rot_ang1 = (temp_ang1+temp_ang2)/2;
    rot_ang2 = (temp_ang1-temp_ang2)/2;
    xscale = 100;
    yscale = 100;

    if (Math.sin(rot_ang1-rot_ang2) != 0) {
        xay = (-pt0.x+pt1.x-pt0.y+pt3.y)/Math.cos(rot_ang1+rot_ang2)/sidelen*100;
        xdy = (pt0.x-pt3.x+pt0.y-pt1.y)/Math.sin(rot_ang1-rot_ang2)/sidelen*100;
        xscale = (xay+xdy)/2;
        yscale = (xay-xdy)/2;
    }
    tempx = pos.x+(pt1.x+pt3.x)/2;
    tempy = pos.y+(pt1.y+pt3.y)/2;
    this[side]._x = tempx;
    this[side]._y = tempy;
    this[side].pic._rotation = rot_ang1*180/Math.PI;
    this[side]._xscale = xscale;
```

```
        this[side]._yscale = yscale;
        this[side]._rotation = rot_ang2*180/Math.PI;
        pt0_3d = point[pt0num].to2d();
        pt1_3d = point[pt1num].to2d();
        pt3_3d = point[pt3num].to2d();
        b = (pt3_3d.x-pt0_3d.x)*(pt1_3d.y-pt0_3d.y)-(pt1_3d.x-pt0_3d.x)*(pt3_3d.y-pt0_3d.y);
        if (b<=0) {
            this[side]._visible = false;
        } else {
            this[side]._visible = true;
        }
    }
```

（24）如图 12-65 所示，返回主场景，在 Layer2 层的第 260 帧处加入 Cube 影片剪辑，并命名为 Cube。

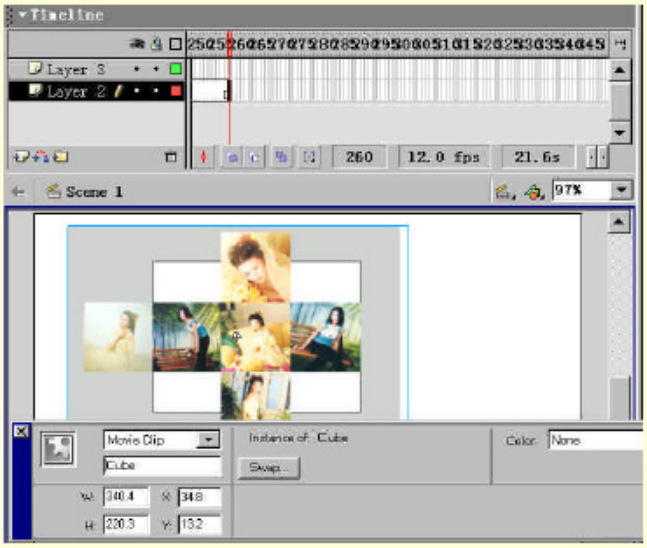


图 12-65　加入 Cube 影片剪辑

（25）在 Layer3 层的第 320 帧处加入如下脚本：

```
cube.play();
```

（26）在 Layer3 层的第 321、322 帧处插入空白帧，如图 12-66 所示，为第 321 帧添加标签 wait。并在第 322 帧处加入如下脚本：

```
if (cube.bfinish==false) {
    gotoAndPlay ("wait");
    return;
}
```

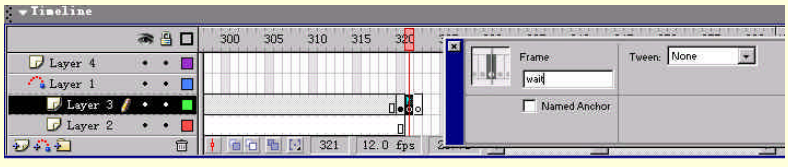


图 12-66　添加帧标签

此步骤创建一个时间轴上的循环，等待自动旋转的立方体动画播放完毕以后再继续播放。

（27）在 Layer2 帧的第 323 帧和第 380 帧处插入关键帧，并在它们之间使用位置渐变，改变第 380 帧上的影片剪辑的透明度为 0。这时的时间轴如图 12-67 所示。

图 12-67　步骤（27）的时间轴

（28）在所有的层之上新建一个层 Layer5。以元件库的形式打开 12.2.3 节中的下雪的动画。如图 12-68 所示，把库中的所有元件复制到当前文件中。

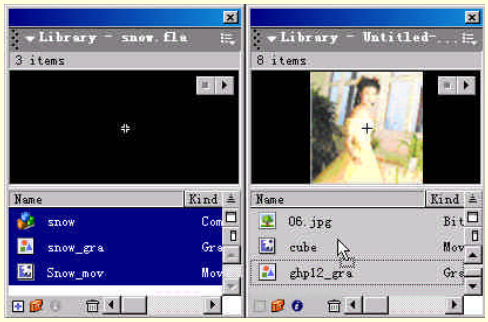


图 12-68　复制元件

（29）把影片剪辑 snow 放置到 Layer5 层，并移动到主场景中的上边中间的位置。snow 的属性都使用默认值。

（30）使用 File | Import 菜单，导入所需要的歌曲。新建一个影片剪辑，命名为 music。在影片剪辑的 Layer1 层中设置 Properties（属性）面板，如图 12-69 所示。其中，声音 Effect（效果）的设置如图 12-70 所示。

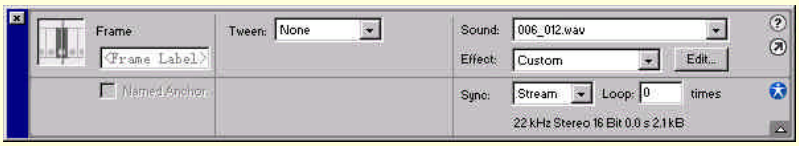


图 12-69　设置声音

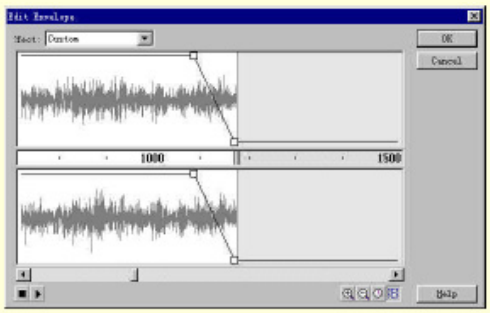


图 12-70　设置声音效果

（31）在设置声音效果的时候记下声音所播放的帧数，则如图 12-71 所示在 Layer1 层的相应位置处插入帧，使声音的波形在时间轴上完全显示出来。

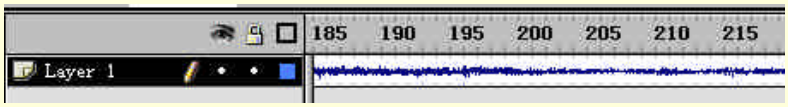


图 12-71　在声音层上插入帧

（32）在 music 影片剪辑中插入新层 Layer2，使用 Flash MX 的各种渐变形式为歌曲加入歌词或者其他语歌曲相关的内容。

- 可以以大模式显示帧，将时间轴放大，声音的波形拉开。这样，可以根据音乐的波形判断每一句歌词的起始和结束帧。
- 另外，点击控制栏上的播放按钮，或者按回车键可以在当前帧的所在位置开始播放音乐。这种功能也便于添加歌词。

（33）回到主场景中，把 music 影片剪辑添加到 Layer5 层，并居中对齐。选中出 Layer5 种以外的所有帧，将它们向后移动一帧。并在主场景的第 1 帧中添加脚本 Stop()。这时的时间轴如图 12-72 所示。

图 12-72　步骤（32）的时间轴

（34）一般的音乐都有段前奏，在 music 影片剪辑中前奏播完的那一帧处加入如下脚本：

```
_root.play();
```

（35）调整主场景中动画渐变的时间或者起始位置，使动画的播放配合音乐。

- 利用步骤（32）~（33）的 Stop-Play 方法可控制主场景中的动画配合音乐播放。

（36）调试完成之后，输出并运行动画。如图 12-73 所示，自己的 MTV 制作完成。

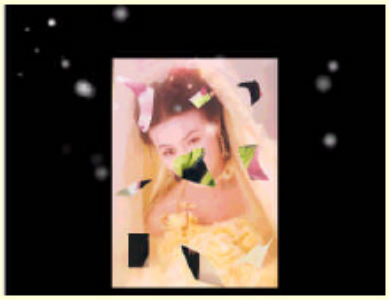


图 12-73　运行结果

12.4.3　总　结

上述的 MTV 是一段较复杂的 Flash MX 动画，也是多种 Flash MX 特效的综合表现，在制作过程中需要反复调试。其中，运用的技巧主要有：

- 把歌词和音乐放在一个影片剪辑中，可以使它们独立起来，即歌词和音乐的播放不因主场景播放的停止或者调转而受到影响。
- 在主场景或者其他动画中使用 Stop 动作，在音乐影片剪辑中使用 Play 动作可以使其他动画配合音乐开始或者停止播放。

12.5 下载进度的显示

12.5.1 介 绍

虽然 Flash MX 动画是一种流式动画文件，即在互联网上，它可以一边下载一边播放。但是，如果某些帧比较大下载的时间过长，将导致动画播放极不连续，甚至会失去某些效果。所以，当播放较大的动画文件时，最好是下载完成之后再播放，而在下载的过程中又不能没用任何反应，所以最好可以提供一个进度条显示下载工作的进度。

Flash MX 中提供了两种动作：getBytesLoaded()和 getBytesTotal()。前者可以得到已经下载的字节数；后者可以得到此文件一共有多少字节，所以，要计算出下载进度或者预计下载时间是很容易的。

12.5.2 实 战

本节给出的例子用于在下载较大 Flash MX 动画时显示下载进度并且估算预计下载时间。

（1）新建一个 Flash MX 文件。创建一个图形元件，命名为 left_gra。如图 12-74 所示使用椭圆工具在图形元件中绘制一个圆，并居中对齐。圆的填充色为白色/黑色的径向渐变色，且无描绘色。

图 12-74 绘制圆

（2）如图 12-75 所示选择圆的一半，将其转换为图形元件，并命名为 right_gra。同时删除在图形元件 left_gra 中的元件。

图 12-75 分割圆

（3）如图 12-76 所示，进入图形元件 right_gra 的编辑状态，选中右半圆，使半圆的左边居中。

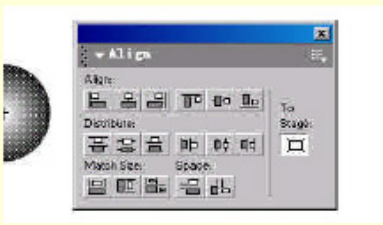


图 12-76 编辑右半圆

（4）新建以图形元件命名为 mid_gra，如图 12-77 所示绘制一个无描绘色，填充色为黑色/白色/黑色渐变的矩形。使矩形右边居中。

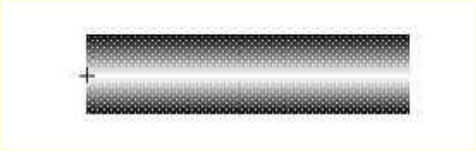


图 12-77 绘制矩形

(5)创建一个影片剪辑,命名为 bar。在第 1 帧中放入上述 3 个图形元件(left_gra、mid_gra、right_gra)。如图 12-78 所示，改变所有图形元件的 Behavior 属性为 Movie Clip（影片剪辑），并分别命名为：left、mid、right。

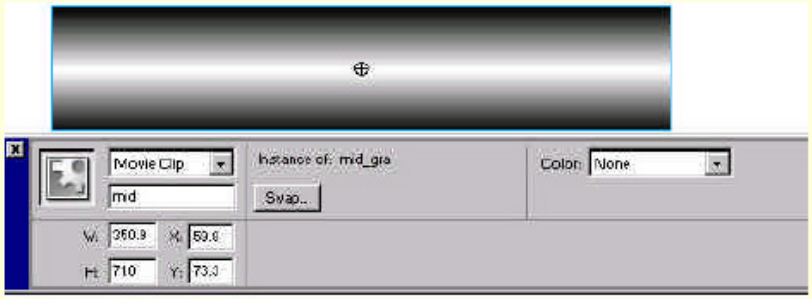


图 12-78 创建影片剪辑 bar

（6）调整上述 3 个元件的位置及大小，如图 12-79 所示。

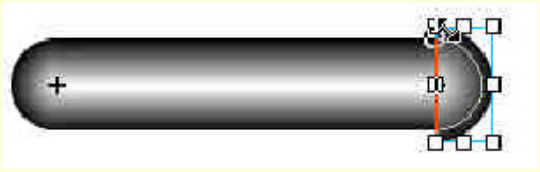


图 12-79 调整元件

调整时，使 left 右边居中，mid 左边居中，right 紧靠在 mid 的右边。

（7）在 Layer1 层上创建一个新层 Layer2，使用椭圆工具及线条工具绘制如图 12-80 所示的边框。线宽度均为 10。

图 12-80 绘制线框

（8）选中线框，使用 Modify | Shape | Convert to Filled 菜单命令将线条转化为填充区域，并按照如图 12-81 所示的方式填充黑/白渐变色。

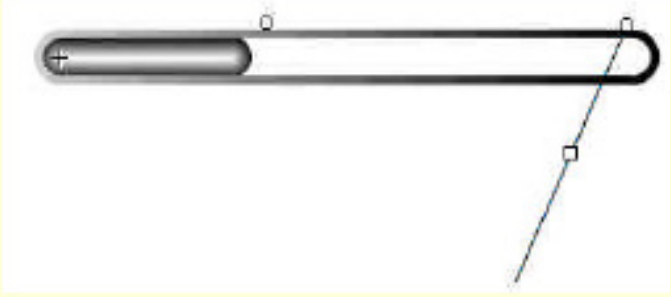


图 12-81 填充颜色

（9）选中整个边框，将其转换成为图形元件，如图 12-82 所示，改变图形元件的 Behavior 属性为 Movie Clip（影片剪辑），并命名为 outline。

图 12-82　改变图形元件的作用

（10）如图 12-83 所示为影片剪辑添加参数 percent。

Name	Variable	Value	Type
percent	percent	50	Default

图 12-83　添加参数

（11）在 bar 影片剪辑的第 1 帧中添加如下脚本：

```
show();
function show () {
    mid._width=(outline._width-outline._height)*percent/100;
    right._x=mid._width;
}
```

（12）回到主场景中，如图 12-84 所示，将影片剪辑 bar 放入主场景的第 1 帧中，并命名为 loadbar。设置其参数 percent=100。

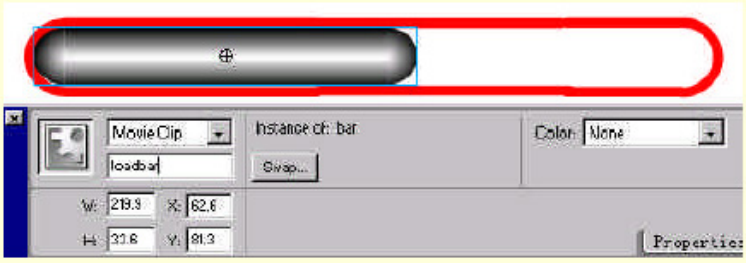


图 12-84　主场景中的影片剪辑 bar

（13）为第 1 帧添加如图 12-85 所示的文字。

图 12-85　添加文字

（14）如图 12-86 所示，在主场景的第 1 帧中添加两个文本区域，分别命名为 process 和 time。

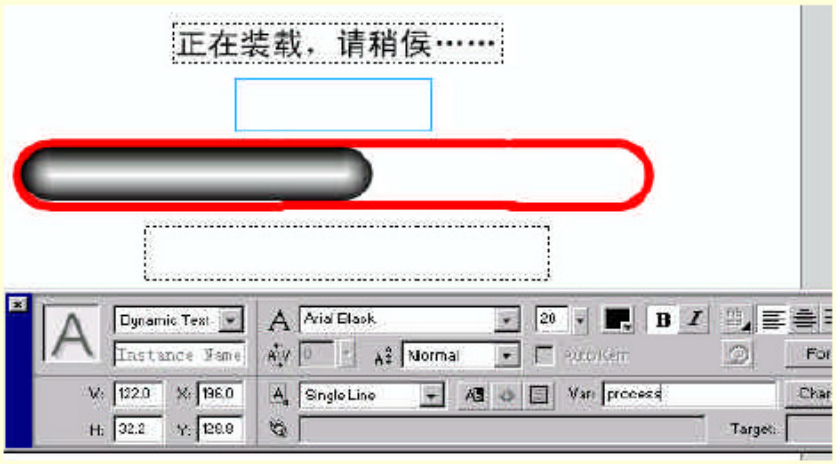


图 12-86　添加一个文本区域

（15）如图 12-87 所示，新建一个层 Layer2，在此层的第 10 帧处插入空白帧，并在 Layer1 层的第 10 帧处插入帧。

图 12-87　新建层

（16）在 Layer2 层的第 10 帧加入如下脚本：

```
bytesloaded = _root.getBytesLoaded();//已经下载的字节数
bytestotal = _root.getBytesTotal();//动画文件的总字节数
//计算完成的百分比
percent = int(bytesloaded/bytestotal*100);
nowtime = getTimer();//获得从动画启动到现在所经过的时间（单位：毫秒）
//设置并刷新进度条
loadbar.percent = percent;
loadbar.show();
process = percent+"%";//显示百分比
//计算装载过程中所经历的时间（单位：毫秒）
loadtime = int(nowtime/1000);
totaltime = int(loadtime/percent*100);//预计总时间
remaintime = totaltime-loadtime;//计算剩余时间
time = "剩余时间："+remaintime+"秒";//显示剩余时间
//如果下载完成，则播放下一帧
if (bytestotal==bytesloaded) {
    return;
}
//如果下载没有完成，则跳转到第 2 帧，循环等待。
gotoAndPlay (2);
```

- 剩余下载时间使用如下公式估算：

$$剩余时间 = 总时间 - 已用时间 = \frac{已用时间}{已下载的百分比} - 已用时间$$

（17）打开要播放的比较大的动画（如 12.4 节中的旋转立方体，或者 12.5 节中的 MTV）。如图 12-88 所示把动画的主场景中的所有帧和层复制到此文件中主场景的第 10 帧之后，或者将它们复制到新层中并移动到第 10 帧之后。

（18）输出并运行动画。因为要测试下载的过程，所以需要选择 View | Show Streaming 的模式测试动画的下载进度。其结果如图 12-89 所示。

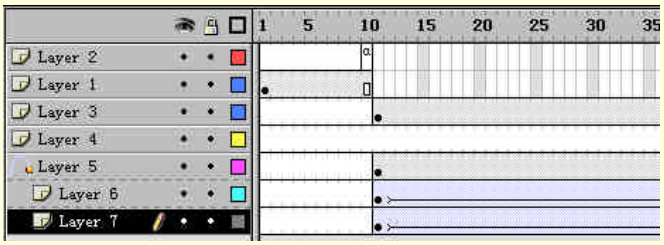


图 12-88　添加动画

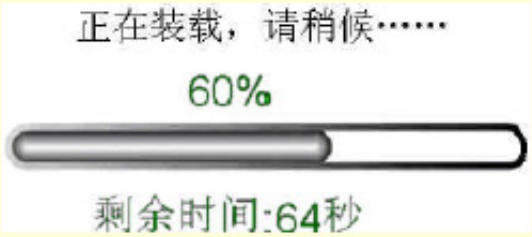


图 12-89　下载进度

✍ 如图 12-90 所示，在运行调试的时候，可以选中 View | Bandwidth Profiler 菜单项来监视下载过程，以检验进度条显示的正确性。

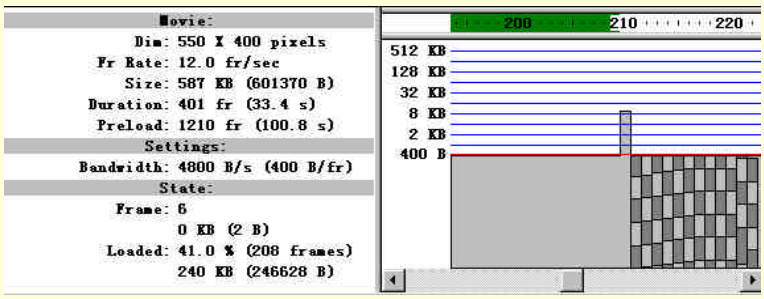


图 12-90　监视下载过程

12.5.3　总　结

可见，使用 Flash MX 中提供的 getBytesLoaded()和 getBytesTotal()两种动作，可以制作出精确的下载进度显示。这种下载进度显示功能在互联网，尤其是速度不高的网络中具有极为重要的作用。试想一下，用户在互联网上打开一个 Flash 主页之前，将会见到一段 Loading 的进度动画，这会使用户觉得不像是在浏览一个主页，而像是在玩一个游戏，即使下载的速度很慢，用户也愿意等下去。

12.6 用 Flash 实现讨论区

12.6.1 介 绍

Flash MX 动画可以与外部文件，尤其是运行在网络上服务器端的脚本（如 ASP、PHP、CGI 等）相互传递信息。我们完全可以制作出一个 Flash 风格的讨论区。

运行在服务器端的脚本与其他形式的讨论区几乎没有差别，它的主要任务就是根据客户端的请求验证用户身份、读写数据库等。

运行在客户机的程序是用 Flash MX 制成的动画。与用其他脚本语言写成的网页一样，它的主要任务是与用户交互，响应用户的请求并发送给服务器，然后接收从服务器端返回的信息。

12.6.2 实 战

下面的例子是一个 Flash MX 风格的讨论区，服务器端的脚本是用 ASP 写成的。数据库使用的是 Access。虽然它并不完善，但足以完成一些讨论区的基本工作，比如：注册用户、匿名登录、发言、回复、修改密码、使用签名档案等。

（1）新建一个 Flash MX 文件，设置动画尺寸为 550×340 像素。在主场景的第 1 帧中加入如图 12-91 所示的静态文本区域及输入文本区域。将两个输入文本区域分别命名为 user 和 password。

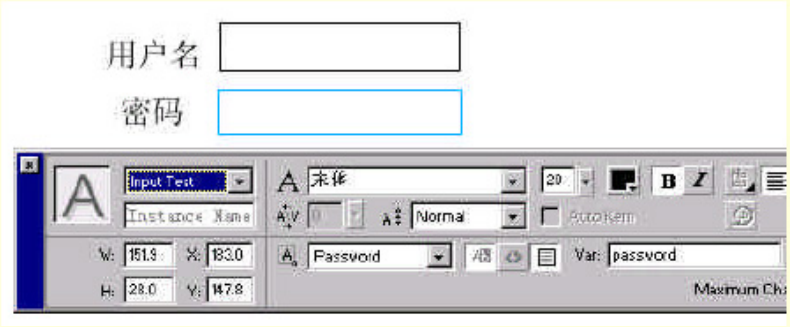


图 12-91 添加文本

（2）如图 12-92 所示，选中当前的所有内容，将它们转换为影片剪辑，并命名为 input。

（3）如图 12-93 所示，在影片剪辑的下面添加两个按钮（“进入”、“新用户”）和一个动态文本区域（命名为 result）。

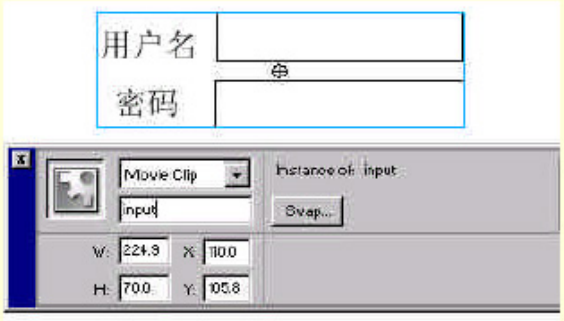


图 12-92 转换为影片剪辑

图 12-93　添加按钮和文本区域

（4）在第 2 帧上插入空白帧，设置帧标签为 new。并添加如图 12-94 所示的静态文本、动态文本、输入文本区域以及“ 注册 ”和“ 取消 ”按钮。将 3 个输入文本区域及动态文本分别命名为 user、password1、password2、newresult。

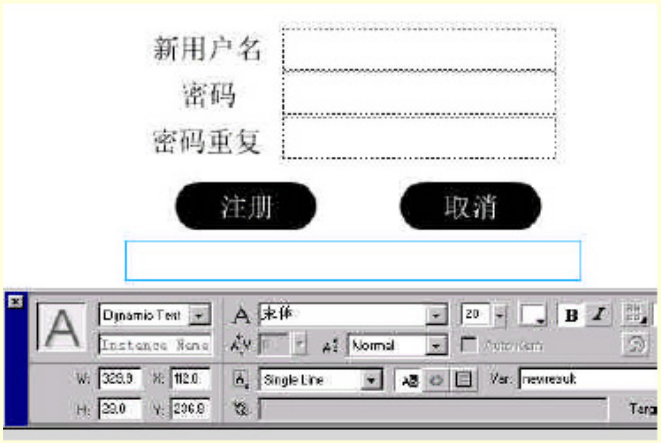


图 12-94　第 3 帧

（5）选中如图 12-95 所示的文本区域，将它们转化为影片剪辑并命名为 newiput。

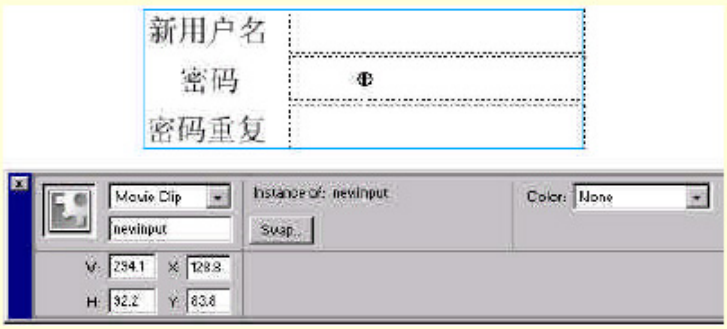


图 12-95　转化为影片剪辑

（6）加入脚本。

- 在第 1 帧中键入如下脚本：

```
stop ();
user_name="guest";
resulte="";
server_url="";//服务器地址
```

- 为“ 进入 ”按钮加入如下脚本：

```
on (release,KeyPress "<Enter>") {
    if (_root.input.user == "guest") {
        // 匿名登录
        gotoAndStop ("discuss");
        user_name = "guest";
        return;
    }
    if (length(_root.input.user) < 0) {
        fscommand ("messagebox", "请输入用户名");
        return;
    }
    if (length(_root.input.password) == 0) {
        fscommand ("messagebox", "请输入密码");
        return;
    }
    // 发送到服务器端运行的 ASP，服务器端的 ASP 脚本将在后面部分给出
    loadVariables(server_url+"discuss.asp?action=test_password&user="
        +_root.input.user+"&password="+_root.input.password, input, "POST");
    result = "正在检查密码，请稍候……";
}
```

- 为影片剪辑 input 加入如下脚本：

```
onClipEvent (data) {
//服务器端返回检验结果
    if (password_test==0) {//密码无效
        _root.result="密码无效，请重试";
        return;
    }
//密码正确
    _root.user_name=user;//用户名
    _root.gotoAndStop ("discuss");//跳转到讨论区的首帧
}
```

- 为“新注册”按钮加入如下脚本：

```
on (release) {
//跳转到新注册帧
    gotoAndStop ("new");
}
```

- 为“注册”按钮加入如下脚本：

```
on (release) {
    if (length(newinput.user)==0) {//用户名为空

        fscommand ("messagebox", "请输入用户名");
        return;
    }
```

```
        if (length(newinput.password1)==0) {//无密码
            fscommand ("messagebox", "请输入密码");
            return;
        }
        if (newinput.password1!=newinput.password2) {//密码不同
            newresult="密码不同，请重试";
            return;
        }
        // 发送到服务器端运行的 ASP，进行注册
        loadVariables(server_url+"discuss.asp?action=new_user&user="+newinput.user
                +"&password="+newinput.password1, "newinput", "POST");
        newresult = "正在注册，请稍候……";
    }
```

- 为“取消”按钮加入如下脚本：

```
on (release) {
//返回登录帧（首帧）
    gotoAndStop (1);
}
```

- 为影片剪辑 newinput 加入如下脚本：

```
onClipEvent (data) {
    if (new_user_test == false) {//用户名重复
        _root.newresult = "用户名重复";
        return;
    }
//注册成功
    fscommand ("messagebox", "恭喜 !! 注册成功!");
    _root.result = "恭喜 !! 注册成功!";
    _root.gotoAndStop(1);
}
onClipEvent (load) {
    new_user_test = false;
}
```

（7）在主场景的第 3 帧处插入空白帧,命名帧标签为 discuss。创建一个影片剪辑 mov_discuss。在影片剪辑中 Layer1 层上插入一个新层 Layer2 并添加文本“讨论区”,在影片剪辑 Layer1 层中绘制如图 12-96 所示的三个矩形（两边的为蓝色，中间的为黄色）并将它们组成一组。

图 12-96　编辑影片剪辑 mov_discuss

（8）在 Layer2 层的第 10 帧处插入帧，在 Layer1 层的第 10 帧处插入关键帧。移动 Layer 1 层的第 10 帧图形到图 12-97 所示的位置。

图 12-97　编辑 Layer1 层

（9）在 Layer1 层的第 1~9 帧之间使用位置渐变，并使用 Layer2 层 Mask（遮罩）Layer1 层。从而得到如图 12-98 所示的闪烁的文字。

图 12-98　创建闪烁文字

（10）将影片剪辑 mov_discuss 添加到主场景的第 3 帧中。然后在 Layer1 层之下为主场景添加 4 个层，由上到下依次命名为 text、hightlight、others、backgroud，并在每一层的第 3 帧处都插入空白帧。如图 12-99 所示。

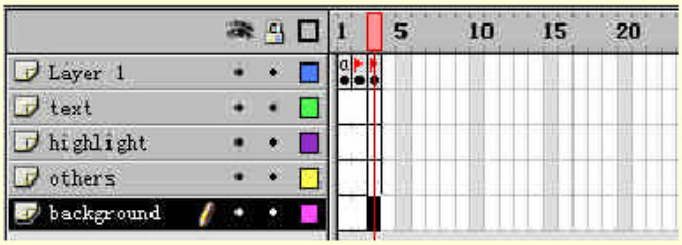


图 12-99　添加层

（11）在 text 层的第 3 帧中，添加如图 12-100 所示的文本区域。

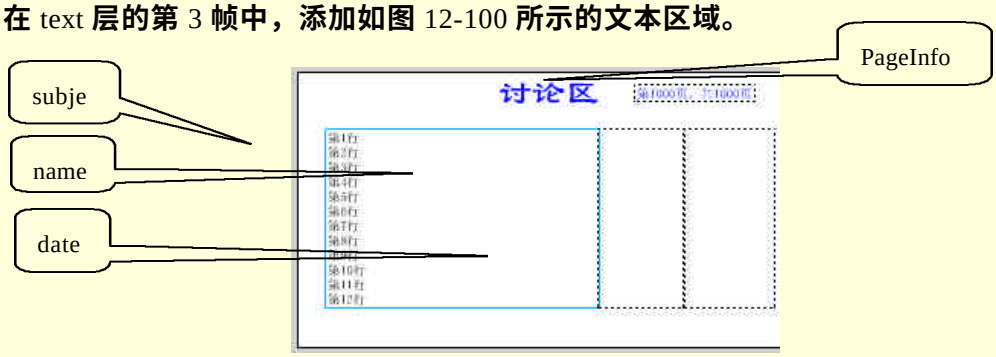


图 12-100　添加文本区域

- 在 Multiline（多行文本）框中预先输入的文字用于在编辑的时候调整多行文本区域的高度和计算文本的行数。在动画的运行中它们会先被 ActionScript 脚本删除，以免被显示出来。

（12）在 highlight 层的第 3 帧中，绘制如图 12-101 所示的浅绿色矩形，作为高亮度条。将图形转化为影片剪辑，并命名为 SelectBar。

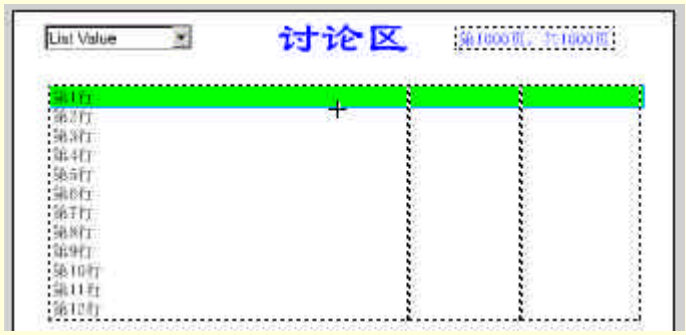


图 12-101　创建高亮度条

（13）创建一个如图 12-102 所示的三角形按钮（从左至右依次为：Up、Over、Down、Hit 帧）。命名为 But_nextpage。

图 12-102　创建按钮 But_nextpage

（14）如图 12-103 所示创建一个与上述按钮方向相反的按钮，命名为 But_prepage。

图 12-103　创建按钮 But_prepage

（15）创建一个图形元件命名为 Gra_btbk，绘制如图 12-104 所示的蓝色/白色渐变的图形。

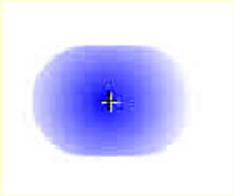


图 12-104　图形元件 Gra_btbk

（16）创建一个影片剪辑命名为 Mov_btbk，在影片剪辑中使用位置渐变，再做出一个如图 12-105 所示的使图形元件 Gra_btbk 先放大后缩小的动画。

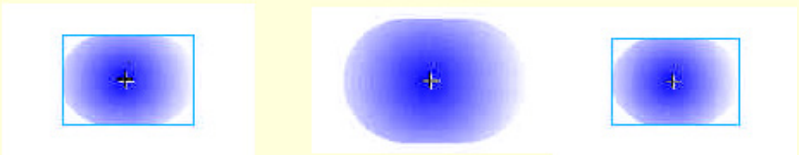


图 12-105　影片剪辑 Mov_btbk

（17）创建一个按钮命名为 But_say。在按钮的 Up、Down 和 Hit 帧中加入图形元件 Gra_btbk；在按钮的 Over 帧中加入影片剪辑 Mov_btbk。然后在按钮的 Up、Over、Down 帧的图形之上加入不同颜色的文字“发言”。 类似的，制作出如图 12-106 所示的按钮。

图 12-106　制作按钮

（18）再创建一个按钮，命名为 But_go，如图 12-107 所示，随意编辑一个按钮即可。

图 12-107　制作 go 按钮

（19）回到主场景中，把上述按钮放置在 others 层的第 3 帧中如图 12-108 所示的位置。

（20）在 backgroud 层的第 3 帧中绘制如图 12-109 所示的浅蓝色背景。并将背景转化为影片剪辑。

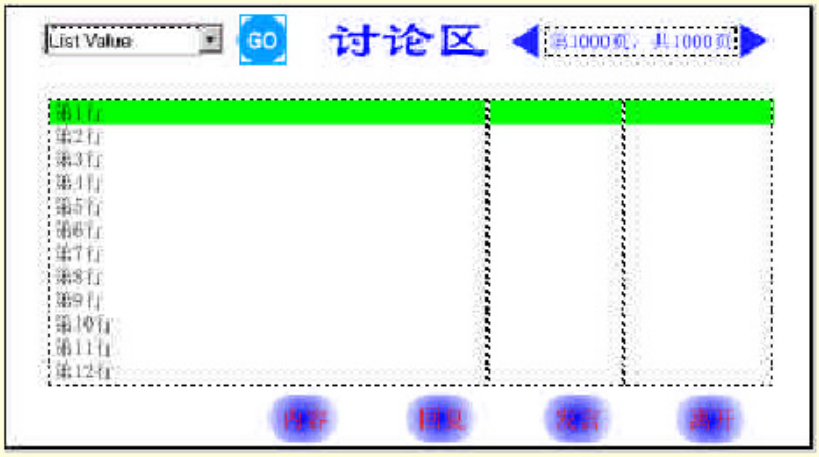


图 12-108　排放按钮

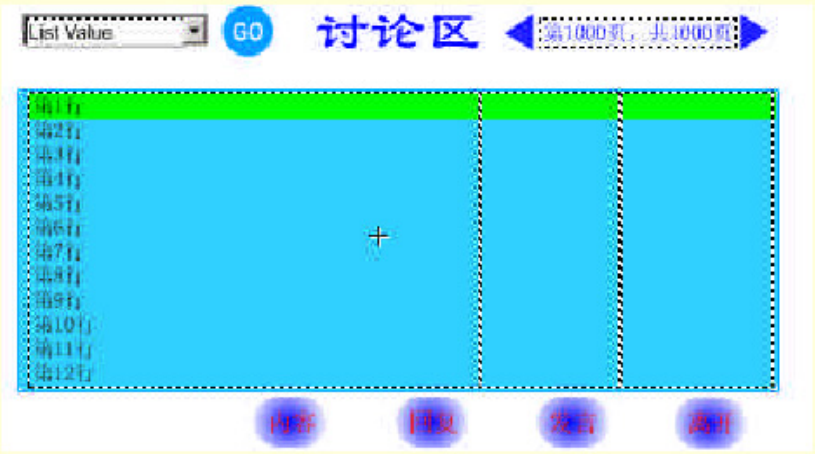


图 12-109　绘制背景

（21）加入 ActionScript 脚本

- 为 text 层的第 3 帧加入如下脚本：

```
all_line = 12;//当前页中的行数
name = "loading...";//发言者
date = "loading...";//发言日期
PageInfo = "loading...";//页码信息
subject = "loading...";//主题
total_line = 12;//所有行数（标题数）
at_line = 1;//高亮度条所在位置
at_page = 1;//所在页码
```

```
total_page = 1;//总页码
//计算高亮度条的最上边的位置
hlpos = background._y-background._height/2+5+SelectBar._height;
hlstep = 17;// 亮度条每一次移动的距离，若不合适，可以调整
stop ();
Update();//调用更新函数
//计算亮度条的位置
SelectBar._y = hlpos+(at_line-1)*hlstep;
//更新函数，从服务器中读取的讨论内容
function Update () {
    loadVariables(server_url+"discuss.asp?action=load&total_line="
        +total_line+"&at_page="+at_page, "", "POST");
}
//亮度条上移一行
function Roll_up () {
    y = SelectBar._y-hlstep;
    at_line = at_line-1;
    if (at_line>=1) {
        SelectBar._y = y;
    } else {
        at_line = 1;
    }
}
//亮度条下移一行
function Roll_down () {
    y = SelectBar._y+hlstep;
    at_line++;
    if (at_line<=all_line) {
        SelectBar._y = y;
    } else {
        at_line = all_line;
    }
}
//显示下一页
function Next_page () {
    if (at_page<total_page) {
        at_page++;
        at_line = 1;
        Update();
        SelectBar._y=hlpos+(at_line-1)*hlstep;
    }
}
//显示上一页
function Pre_page () {
```

```
        if (at_page>1) {
            at_page--;
            at_line = 1;
            Update();
            SelectBar._y=hlpos+(at_line-1)*hlstep;
        }
    }
    //修改密码
    function change_pd () {
        if (user_name!="guest") {
            ps_old_pd = "";
            ps_password2 = "";
            ps_password1 = "";
            ps_pd_comment = "";
            gotoAndStop ("discuss_change_pd");
        } else {
            fscommand ("messagebox", "你使用匿名登录，没有密码");
        }
    }
    //修改签名档案
    function edit_personal_info () {
        if (user_name!="guest") {
            ps_name=user_name;
            ps_sign1="loading...";
            ps_sign2="loading...";
            ps_sign3="loading...";
            ps_comment = "";
            loadVariables (server_url+"discuss.asp?action=load_ps&user="
                    +user_name, "", "POST");
            gotoAndStop ("ps_sign1");
        } else {
            fscommand ("messagebox", "你使用匿名登录，没有签名档案");
        }
    }
    //显示签名档案
    function show_sign () {
        gotoAndStop ("ps_sign"+sign_word);
    }
    //插入签名
    function Insert_sign () {
        if (user_name!="guest") {
            loadVariables(server_url+"discuss.asp?action=insert_sign&user="
                +user_name+"&sign="+dis_sign_number, "", "POST");
        } else {
```

```
            fscommand ("messagebox", "你使用匿名登录，没有签名档案");
        }
    }
```

- **为背景影片剪辑加入如下脚本：**

```
onClipEvent (keyDown) {
//判断方向键，控制高亮度条滚动
    if (Key.getCode()==Key.UP) {
     //向上移动高亮度条
        _root.Roll_up();
    } else if (Key.getCode()==Key.DOWN) {
     //向下移动高亮度条
        _root.Roll_down();
    }
}
```

- **为“内容”按钮加入如下脚本：**

```
on (release, keyPress "<Right>") {
//装载指定主题的内容
    loadVariables(server_url+"discuss.asp?action=bodyload&line="+at_line
            +"&at_page="+at_page, "", "POST");
    body_subject = "主题...";
    body_body = "loading...";
    gotoAndStop ("Discuss_text");
}
```

- **为“回复”按钮加入如下脚本：**

```
on (release, keyPress "r") {
//判断是否匿名登录
    if (user_name!="guest") {
     //非匿名登录
        reply = true;
        gotoAndStop ("Discuss_say");
    } else {
        fscommand ("messagebox", "你使用匿名登录，无权发言");
    }
}
```

- **为“发言”按钮加入如下脚本：**

```
on (release) {
//判断是否匿名登录
    if (user_name!="guest") {
     //非匿名登录
        gotoAndStop ("Discuss_say");
        reply = false;
    } else {
```

```
            fscommand ("messagebox", "你使用匿名登录，无权发言");
        }
    }
```

- **为“离开”按钮加入如下脚本：**

```
on (release) {
//退出讨论区，重新提示登录
    gotoAndStop (1);
}
```

- **为“Go”按钮加入如下脚本：**

```
on (release) {
    if (menu.currentValue==menu.items[0]) {//编辑签名档案
        edit_personal_info();
    } else if (menu.currentValue==menu.items[1]) {
    //修改密码
        change_pd();
    }
}
```

- **为“上一页”按钮加入如下脚本：**

```
on (release, keyPress "<PageDown>")
{
    Pre_page();
}
```

- **为“下一页”按钮加入如下脚本：**

```
on (release, keyPress "<PageDown>")
{
    Next_page();
}
```

（22）在 Layer1 层的第 9 帧处插入帧。分别在 text、others、background 层的第 4 帧处插入空白帧。把 text 层第 4 帧处的帧标签命名为 Discuss_say。并在 text 层的第 4 帧处加入如图 12-111 所示的文本区域。

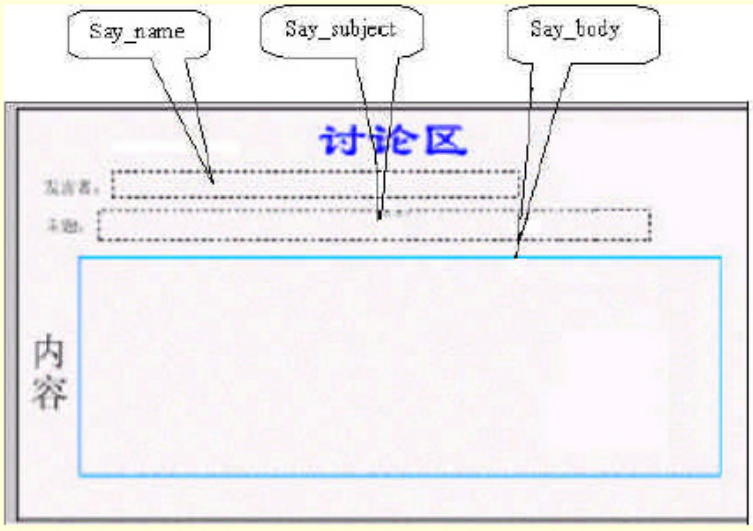


图 12-111　加入文本区域

（23）分别在 otehrs、background 层的第 4 帧处加入如图 12-112 所示的按钮及背景。

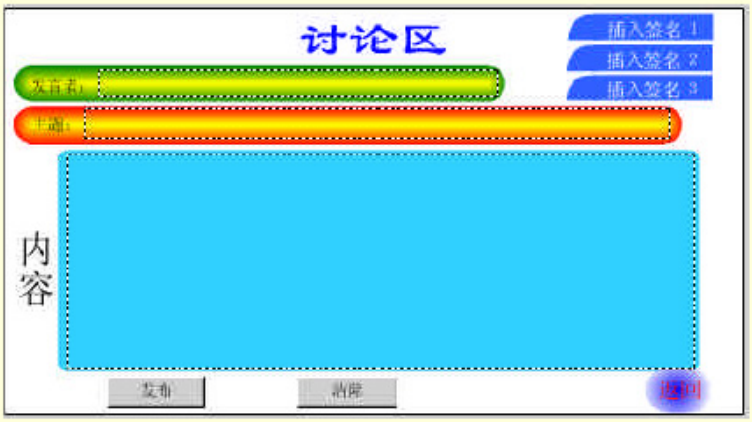


图 12-112　加入背景及按钮

（24）添加 ActionScript 脚本：

- 在 text 层的第 4 帧处加入如下脚本：

```
stop ();
//判断是否为回复，如果是则装载文章内容
if (reply==true) {//是回复
    loadVariables(server_url+"discuss.asp?action=reply&line="+at_line
                                + "&at_page="+ at_page, "", "POST");
    say_subject = "loading...";
    say_body = "loading...";
} else {//不是回复
    say_subject = "";
    say_body = "";
}
say_name = user_name;//设置用户名
```

- 为“发布”按钮加入如下脚本：

```
on (press) {
//判断文章是否有主题，无主题的文章不允许发布
    if (length(say_subject)<>0) {
        loadVariables(server_url+"discuss.asp?action=send&reply="+reply,
                        "", "POST");
        gotoAndStop ("Discuss");//返回讨论区主界面
    } else {
        fscommand ("messagebox", "请填写主题");//提示填写文章主题
    }
}
```

- 为“清除”按钮加入如下脚本：

```
on (release) {
    say_body = "";//清除文章内容
}
```

- 为“返回”按钮加入如下脚本：

```
on (release) {
```

```
    gotoAndStop ("Discuss");//返回讨论区主界面
}
```

- **为“插入签名 1”按钮加入如下脚本：**

```
on (release) {
//调用函数 Insert_sign(签名号)在文章结尾添加用户签名档案
    Insert_sign(1);
}
```

- **为“插入签名 2”按钮加入如下脚本：**

```
on (release) {
    Insert_sign(2);
}
```

- **为“插入签名 3”按钮加入如下脚本：**

```
on (release) {
    Insert_sign(3);
}
```

（25）分别在 text、others、background **层的第** 5 **帧处插入空白帧。把** text **层第** 5 **帧处的帧标签命名为** Discuss_text**。并在** text **层的第** 4 **帧处加入如图** 12-113 **所示文本区域。**

图 12-113　**添加文本区域**

（26）分别在 otehrs、background **层的第** 5 **帧处加入如图** 12-114 **所示的按钮及背景。**

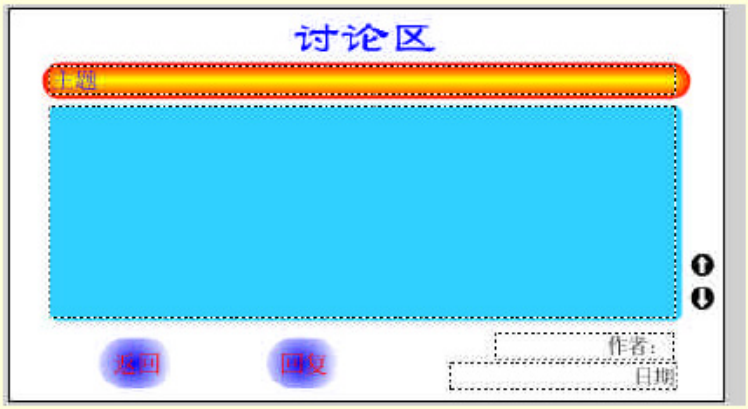


图 12-114　**加入按钮及背景**

（27）添加 ActionScript **脚本：**

- **为“返回”按钮加入如下脚本：**

```
on (release, keyPress "<Left>") {
```

```
        gotoAndStop ("Discuss");
    }
```

- **为“回复”按钮加入如下脚本：**

```
on (release, keyPress "r") {
//判断是否匿名登录
    if (user_name!="guest") {//不是匿名登录，允许回复
        reply = true;
        gotoAndStop ("Discuss_say");
    } else {
        fscommand ("messagebox", "你使用匿名登录，无权发言");
    }
}
```

- **为“向上滚动”按钮加入如下脚本：**

```
on (press, keyPress "<Up>") {
//向上滚动文章主体的文本区域
    body_body.scroll--;
}
```

- **为“向下滚动”按钮加入如下脚本：**

```
on (press, keyPress "<Down>") {
//向下滚动文章主体的文本区域
    body_body.scroll++;
}
```

（28）分别在 text、others、background 层的第 6 帧处插入空白帧。把 text 层第 5 帧处的帧标签命名为 Discuss_text，并分别在 text、otehrs 层的第 6 帧处加入如图 12-115 所示的文本区域和按钮。

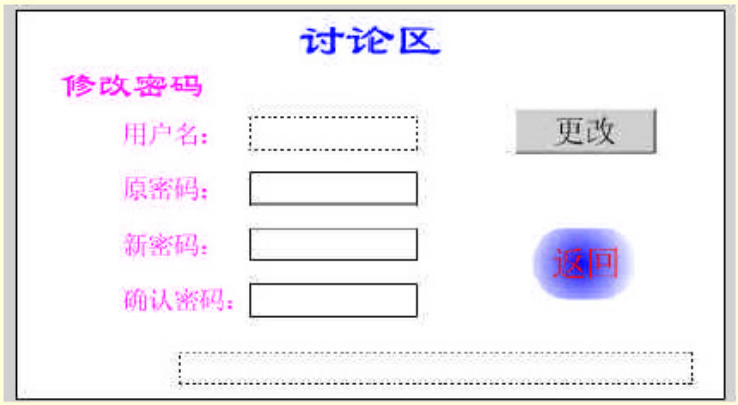


图 12-115　加入文本区域和按钮

（29）添加 ActionScript 脚本：

- **为 text 层的第 6 帧加入如下脚本：**

```
ps_name = user_name;
```

- **为“更改”按钮加入如下脚本：**

```
on (release) {
//如果两次输入的密码相同，并且密码不为空才允许改动
    if ((ps_password1 == ps_password2) and length(ps_password1)<>0) {
        loadVariables(server_url+"discuss.asp?action=change_pd&user="+ps_name
```

```
                +"&old_pd="+ps_old_pd+ "&password="+ps_password1, "", "POST");
            ps_pd_comment = "请稍候...";
        } else {
            ps_pd_comment = "新密码设置不一致，请重试";
        }
    }
```

- 为“返回”按钮加入如下脚本：

```
on (release) {
    gotoAndStop ("Discuss");
}
```

（30）回到主场景中，分别在 text、others、background 层的第 7 帧处插入空白帧。命名 text 层的帧标签为 ps_sign1。并在相应的层中添加如图 12-116 所示的文本区域、按钮及背景。

（31）创建一个影片剪辑，命名为 Mov_discuss_sign。在第 1 帧、第 2 帧中分别使用不同的渐变色添加如图 12-117 所示的图形元件。

（32）把上述第 1 帧的图形元件的 Behavior 属性改为按钮并为它添加如下脚本：

```
on (release) {
    _root.sign_word = sign;
    _root.show_sign();
}
```

图 12-116　加入文本区域、按钮及背景

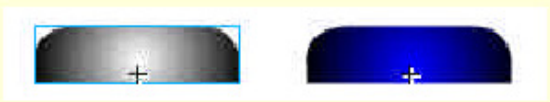


图 12-117　绘制图形

（33）在影片剪辑的两帧中均加入脚本 Stop()。并创建一个新层 Layer2，在此层上为按钮加入如图 12-118 所示的动态文字区域 sign_text。

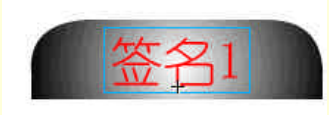


图 12-118　加入动态文字区域

（34）如图 12-119 所示，为影片剪辑 Mov_discuss_sign 添加参数 sign。

Name	Variable	Value	Type
Sign	sign	1	Default

图 12-119　添加参数

（35）为 Layer2 层添加如下脚本：

```
//根据参数 sing 动态地改变文字
sign_text="签名"+sign;
```

（36）如图 12-120 所示把上述影片剪辑放置到主场景的 others 层的第 7 帧中分别命名为 sign1、sign2、sign3，并设置参数 sign 分别为 1、2、3。

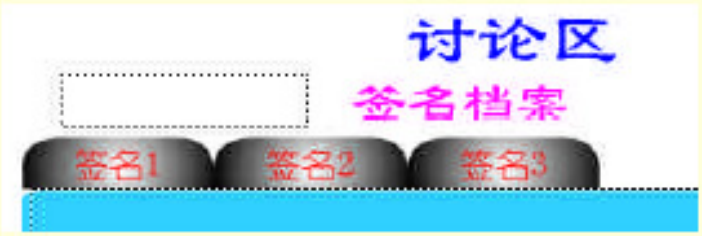


图 12-120　加入影片剪辑

（37）在 text 层的第 8 帧、第 9 帧处插入关键帧。命名帧标签为 ps_sign2、ps_sign3。分别在 others、background 层的第 9 帧处插入帧。这时的时间轴如图 12-121 所示。

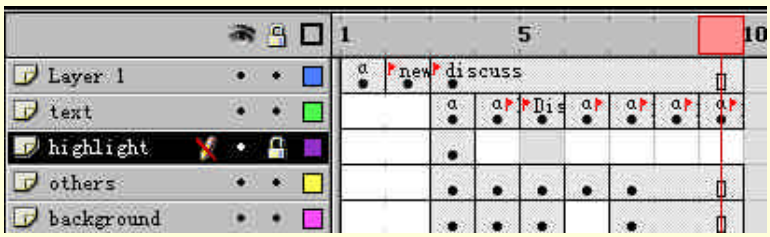


图 12-121　步骤（35）的时间轴

（38）分别改变 text 层的第 8 帧、第 9 帧处的 ps_sign1 文本区域的名称为 ps_sign2、ps_sign3。

（39）添加 ActionScript 脚本：

- 为 text 层的第 7 帧加入如下脚本：

```
stop ();
sign1.gotoAndStop(2);
sign2.gotoAndStop(1);
sign3.gotoAndStop(1);
```

- 为 text 层的第 8 帧加入如下脚本：

```
stop ();
sign1.gotoAndStop(1);
sign2.gotoAndStop(2);
sign3.gotoAndStop(1);
```

- 为 text 层的第 9 帧加入如下脚本：

```
stop ();
sign1.gotoAndStop(1);
sign2.gotoAndStop(1);
sign3.gotoAndStop(2);
```

- **为“保存”按钮添加如下脚本：**

```
on (release) {
    loadVariables (server_url+"discuss.asp?action=change_ps&user="+ps_name,
                "", "POST");
    ps_comment = "正在保存，请稍后...";
}
```

- **为“返回”按钮添加如下脚本：**

```
on (release) {
    gotoAndStop ("Discuss");
}
```

（40）如图 12-122 所示，使用 Publish Settings 对话框设置并将 Flash MX 动画输出到网页中。

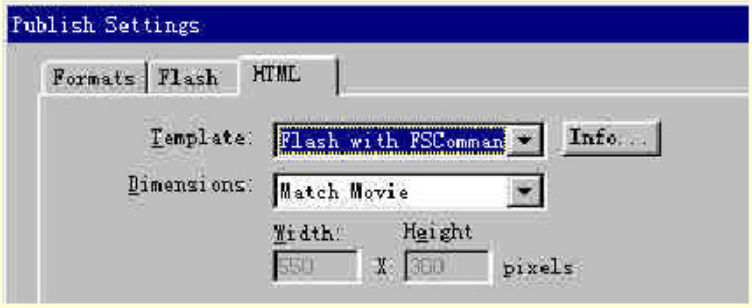


图 12-122　设置并输出 Flash MX 动画

（41）以文本方式打开上面输出的 HTML 页面，加入如下脚本：

```
function discuss_DoFSCommand(command, args) {
  var discussObj = InternetExplorer ? discuss ： document.discuss;
  //
  // Place your code here...
  //
    if(command=="messagebox"){
        alert(args);
    }
}
```

✍ **至此，客户机的软件制作完成，但是没有服务器的支持无法正常工作。下面开始服务器方面脚本的制作。**

（42）编写 ASP 脚本（discuss.asp）如下：

```
<% '定义变量
Dim action,filename,iLength,name,subject,Date,line,i,body,reply,password

Dim total_line,at_page,pageinfo,total_page,sign1,sign2,sign3, Page_size
'为回复正文的每一行添加“>”
Sub ReplyBody()
    Dim tmpbody,tmp
    iLength=Len(body)
    tmpbody=">"
    For i=1 To iLength
        tmp=Mid(body,i,1)
```

```
            If tmp=Chr(13) And i<>iLength Then
                tmpbody=tmpbody+tmp+">"
            Else
                tmpbody=tmpbody+tmp
            End If
        Next
        body=tmpbody+Chr(13)
End Sub
'获得 action 参数
action = Request.QueryString("action")
'得到服务器中 discuss.mdb 数据库的目录
filename=server.mappath("/discuss.mdb")+";"
Page_size=12    ' 每页行数
'打开数据库
strProvider="Driver={Microsoft Access Driver (*.mdb)}; DBQ="+filename
set objConn = server.createobject("ADODB.Connection")
objConn.Open strProvider
'执行不同的动作
If action = "load" Then              '装载主题
    total_line=CInt(Request.QueryString("total_line"))
    at_page=CInt(Request.QueryString("at_page"))
'以时间排序打开数据库中的 Discuss 表格
    Set rs=Server.CreateObject("ADODB.Recordset")
    rs.Open "SELECT * FROM Discuss_Date",strProvider,1,3
'按每页 Page_size 行分页
    rs.PageSize= Page_size
    rs.AbsolutePage=at_page
    total_page=rs.PageCount
    pageinfo="第" & at_page & "页，"+"共" & total_page & "页"
'更新 Flash MX 动画中的文本区域
    i=0
    Do While i<total_line And Not rs.EOF
        subject=subject+rs("Subject")+Chr(13)
        name=name+rs("name")+Chr(13)
        date=date+FormatDateTime(rs("date"),0)+Chr(13)
        rs.MoveNext
        i=i+1
    Loop
'返回参数
    Response.write("subject="+subject+"&name="+name+"&date="+date+"&PageInfo="
        +pageinfo+"&all_line="&i&"&total_page="&total_page)
    rs.Close
    Set rs=nothing
ElseIf action = "bodyload" Then              '装载正文
```

```
        line=CInt(Request.QueryString("line"))
        at_page=CInt(Request.QueryString("at_page"))
'以时间排序打开数据库中的 Discuss 表格
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM Discuss_Date",strProvider,1,3
'按每页 Page_size 行分页
        rs.PageSize=Page_size
        rs.AbsolutePage=at_page
'找到第 at_page 页中的第 line 行
        i=1
        Do While i<line And Not rs.EOF
            i=i+1
            rs.MoveNext
        Loop
'读数据
        subject=rs("Subject")
        name=rs("Name")
        Date=rs("Date")
        date=FormatDateTime(Date,1)+FormatDateTime(Date,3)
        body=rs("Body")
'返回参数
        Response.write("body_subject="+subject+"&body_body="+body+"&body_date="+date
                +"&body_writer="+name)
        rs.Close
        Set rs=nothing
ElseIf action = "send" Then             '发表文章

        reply=Request.QueryString("reply")
        Name=RTrim(Request.form("say_Name"))
        Date = ( month( now ) & "/" & day( now ) & "/" & year( now ) & " " & time() )
        Subject=RTrim(Request.form("say_Subject"))
        Body=RTrim(Request.form("say_Body"))

        If Body = "" Then
            iLength=255
        Else
            iLength = Len(Body)
        End If
'打开数据表格 Discuss
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM Discuss",strProvider,1,3
'创建新的数据表单
        rs.AddNew
            rs("Name")=Name
```

```
            rs("Date")=Date
            rs("Subject")=Subject
            rs("Body")=Body
            rs("Reply")=reply
        rs.Update
        rs.Close
        Set rs=nothing
ElseIf action = "reply" Then              '回复作者
        line=CInt(Request.QueryString("line"))
        at_page=CInt(Request.QueryString("at_page"))
'以时间排序打开数据库中的 Discuss 表格
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM Discuss_Date",strProvider,1,3
'按每页 Page_size 行分页
        rs.PageSize=Page_size
        rs.AbsolutePage=at_page
'找到第 at_page 页中的第 line 行
        i=1
        Do While i<line And Not rs.EOF
            i=i+1
            rs.MoveNext
        Loop
'生成回复文本
        subject=rs("Subject")
        If Left(subject,3)<>"回复：" Then
            subject="回复："+subject
        End If
        body=rs("Body")
        reply=rs("discussID")
        ReplyBody()
'返回参数
        Response.write("say_subject="+subject+"&say_body="+body+"&reply="&reply)
        rs.Close
        Set rs=nothing
ElseIf action = "test_password" Then        '验证密码
        name=LTrim(RTrim(Request.QueryString("user")))
        password=Request.QueryString("password")
'打开数据库中的 UserInfo 表格中的 name 项
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM UserInfo WHERE User='"&name&"'",strProvider,1,3
        If rs.EOF Then'                     无此用户
            Response.write("password_test=0")
        Else
            If rs("Password")<>password Then    '密码错误
```

```
                Response.write("password_test=0")
            Else
                Response.write("password_test=1")
            End If
        End If
        rs.Close
        Set rs=nothing
    ElseIf action = "new_user" Then        '新注册
        name=LTrim(RTrim(Request.QueryString("user")))
        password=Request.QueryString("password")
        Date = ( month( now ) & "/" & day( now ) & "/" & year( now ) & " " & time() )
    '打开数据库中的 UserInfo 表格中的 name 项
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM UserInfo WHERE User='"&name&"'",objConn,1,3
        If rs.EOF Then          '新用户
            rs.Close
            '打开 UserInfo 表格，并添加信息
            Set rs=Server.CreateObject("ADODB.Recordset")
            rs.Open "SELECT * FROM UserInfo",objConn,1,3
            Response.write("new_user_test=1")
            rs.AddNew
                rs("User")=Name
                rs("Date")=Date
                rs("Password")=password
            rs.Update
        Else        '用户名重复
            Response.write("new_user_test=0")
        End If
        rs.Close
        Set rs=nothing
    ElseIf action = "change_pd" Then      '更改密码
        name=LTrim(RTrim(Request.QueryString("user")))
        password=Request.QueryString("password")
        old_pd=Request.QueryString("old_pd")
    '打开数据库中的 UserInfo 表格中的 name 项
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM UserInfo WHERE User='"&name&"'",strProvider,1,3
        If name=rs("User") Then
            If old_pd=rs("Password") Then
                rs("Password")=password
                rs.Update
                Response.write("ps_pd_comment=密码更改成功")
            Else
                Response.write("ps_pd_comment=密码输入错误")
```

```
            End If
        End If
        rs.Close
        Set rs=nothing
ElseIf action = "load_ps" Then  '读取用户信息（签名档案）
        name=LTrim(RTrim(Request.QueryString("user")))
'打开数据库中的 UserInfo 表格中的 name 项
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM UserInfo WHERE User='"&name&"'",strProvider,1,3
        If name=rs("User") Then
            sign1=RTrim(rs("Sign1"))
            sign2=RTrim(rs("Sign2"))
            sign3=RTrim(rs("Sign3"))
            Response.write("ps_sign1="+sign1+"&ps_sign2="+sign2+"&ps_sign3="+sign3)
        End If
        rs.Close
        Set rs=nothing
ElseIf action = "change_ps" Then      '更改用户信息（签名档案）
        name=LTrim(RTrim(Request.QueryString("user")))
        sign1=RTrim(Request.form("ps_sign1"))
        sign2=RTrim(Request.form("ps_sign2"))
        sign3=RTrim(Request.form("ps_sign3"))
'打开数据库中的 UserInfo 表格中的 name 项
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM UserInfo WHERE User='"&name&"'",strProvider,1,3
        If name=rs("User") Then
            If Len(sign1)<>0 Then
                rs("Sign1")=sign1
            Else
                rs("Sign1")=" "
            End If
            If Len(sign2)<>0 Then
                rs("Sign2")=sign2
            Else
                rs("Sign2")=" "
            End If
            If Len(sign3)<>0 Then
                rs("Sign3")=sign3
            Else
                rs("Sign3")=" "
            End If
            rs.Update
            Response.write("ps_comment=存储完毕")
        End If
```

```
        rs.Close
        Set rs=nothing
    ElseIf action = "insert_sign" Then     '插入签名
        name=LTrim(RTrim(Request.QueryString("user")))
        sign="Sign"+Request.QueryString("sign")
        body=Request.form("say_Body")
    '打开数据库中的 UserInfo 表格中的 name 项
        Set rs=Server.CreateObject("ADODB.Recordset")
        rs.Open "SELECT * FROM UserInfo WHERE User='"&name&"'",strProvider,1,3
        If name=rs("User") Then
            Response.write("say_body="+body+RTrim(rs(sign)))
        End If
        rs.Close
        Set rs=nothing
    End If
    objConn.Close
    Set objConn=nothing%>
```

（43）建立 Microsoft Access 数据库。在数据库 Discuss.mdb 中有两个表格 Disscuss 和 UserInfo。它们的定义如表 12-1 和表 12-2 所示。

表 12-1　　Discuss 的定义

字段名称	数据类型	说　明
DiscussID	自动编号	发布文章的标识 ID 号
Name	文本	发布文章的作者名
Date	日期/时间	发布文章的时间
Subject	文本	发布文章的主题
Body	备注	发布文章的内容
Reply	数字	如果为 0 则不是回复，否则为恢复文章的 discussID

表 12-2　　UserInfo 的定义

字段名称	数据类型	说　明
ID	自动编号	讨论区用户的 ID
User	文本	讨论区的用户名
Password	文本	讨论区的用户密码
Date	日期/时间	用户申请的时间
Sign1	备注	讨论区的用户签名档案 1
Sign2	备注	讨论区的用户签名档案 2
Sign3	备注	讨论区的用户签名档案 3

（44）为数据库定义一个查询 Discuss_Date，按日期顺序升序排列表个 Discuss 的项。

（45）发布数据库、ASP 脚本、网页、*.Swf 动画到服务器上，讨论区便可以正常运行了。

12.6.3 总 结

虽然上述讨论区还不够完善，但它是 Flash MX 动画与 ASP 交互的典型应用。如果读者用 ASP 和 JavaScript 脚本创建过讨论区，就会发现，它们与 Flash MX 中对 ASP 的请求格式和过程基本相同，所以完全可以仿造上述例子再制作一个讨论区的 Flash 版本，而对于服务器端的 ASP 程序根本不用修改。

12.7 桌面台历时钟

12.7.1 介 绍

一般来说，Windows 系统的桌面都是静止的图片，虽然这种桌面具有很强的观赏性，但实用性不是很强。Flash 的出现可以很好地解决这个问题，因为 Flash 动画可以嵌入到网页中，而如果将嵌有 Flash 动画文件的网页设为 Windows 系统的桌面，就将得到一个具有动感或者具有一定功能的桌面。

本例将介绍一个能即时显示当前时间和日期的 Flash 动画，将它嵌入网页并将网页设为 Windows 系统的桌面，最终得到一个美观实用的桌面时钟台历。

12.7.2 实 战

（1）新建一个 Flash MX 文档，命名为“ 时钟台历 ”，设置文档的大小为 360×360 像素，背景为白色，如图 12-123 所示。

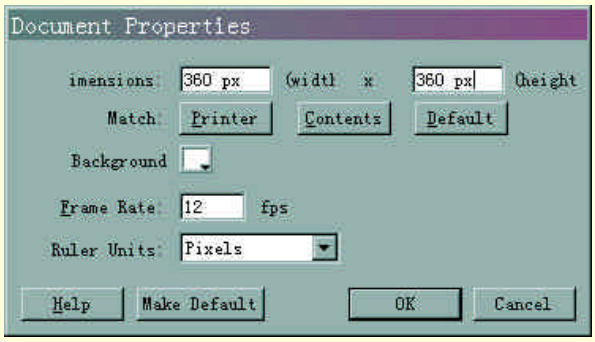


图 12-123 进行舞台设置

（2）使用 File | Import 菜单命令，导入一幅事先已处理好的背景图片，如图 12-124 所示。

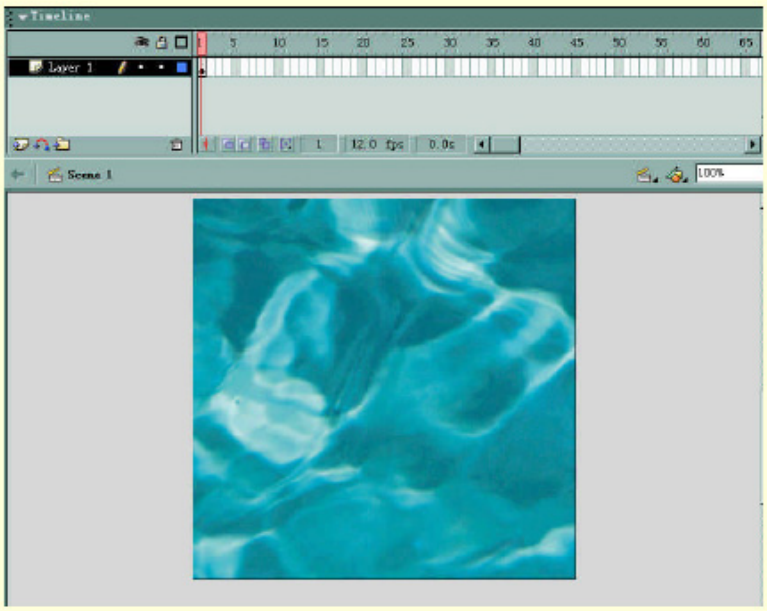


图 12-124 导入背景图片

（3）使用 Insert | New Symbol 菜单命令创建一个新的电影剪辑元件，命名为“秒针”，单击 OK 按钮，进入电影剪辑元件编辑状态。按住 Shift 键在舞台中绘制一根垂直直线，线宽为 1 像素，描绘色为红色，如图 12-125 所示。这根直线的底端一定要对准舞台的中心，这样，在旋转指针时，指针才会围绕时钟中心旋转。

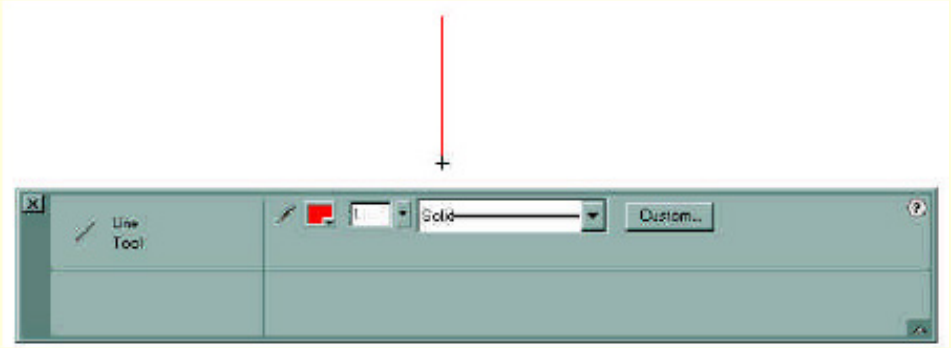


图 12-125　绘制秒针

（4）新建一个电影剪辑元件，命名为“分针”。在“分针”电影剪辑元件的编辑状态下，在舞台中绘制一根线宽为 2 像素、颜色为绿色的垂直直线，如图 12-126 所示。同样，这根直线的底端也要对准舞台的中心，而且这根直线要稍短于“秒针”电影剪辑元件中的直线。

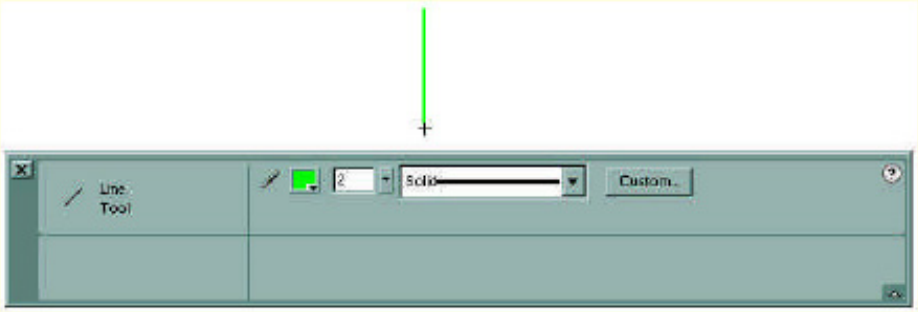


图 12-126　绘制分针

（5）再新建一个电影剪辑元件，命名为“时针”。在“时针”电影剪辑元件的编辑状态下，在舞台中绘制一根线宽为 4 像素、颜色为黄色的垂直直线，如图 12-127 所示。同样，这根直线的底端也要对准舞台的中心，而且这根直线要稍短于“分针”电影剪辑元件中的直线。

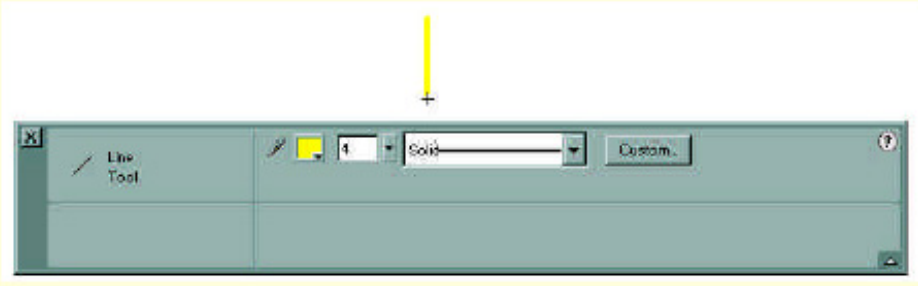


图 12-127　绘制时针

（6）创建一个新的电影剪辑元件，命名为“钟”，进入“钟”电影剪辑元件的编辑状态。先用 Oval（椭圆）工具在舞台中绘制两个大小不同的椭圆，然后分别令其对齐到舞台中央，再用 Text（文字）工具在相应位置写上时间刻度，如图 12-128 所示。

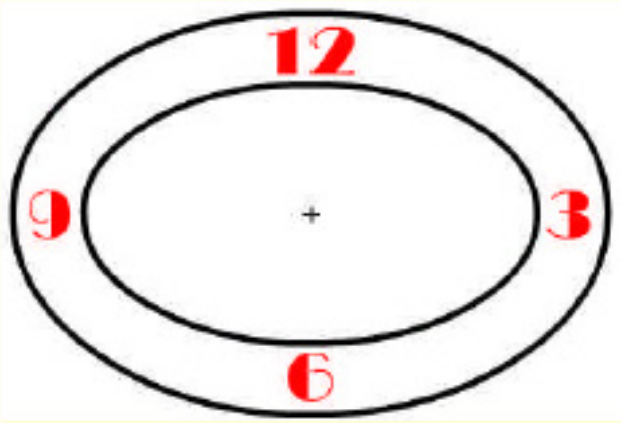


图 12-128　绘制表盘

（7）在“钟”电影剪辑元件的时间轴上新建一个层，使用 Window | Library 菜单命令调出元件库面板，依次将“时针”、“分针”、“秒针”这三个电影剪辑元件从元件库面板中拖曳到新建层的舞台中，位置如图 12-129 所示。这三个电影剪辑元件的中心“+”字一定要对准当前舞台的中心。

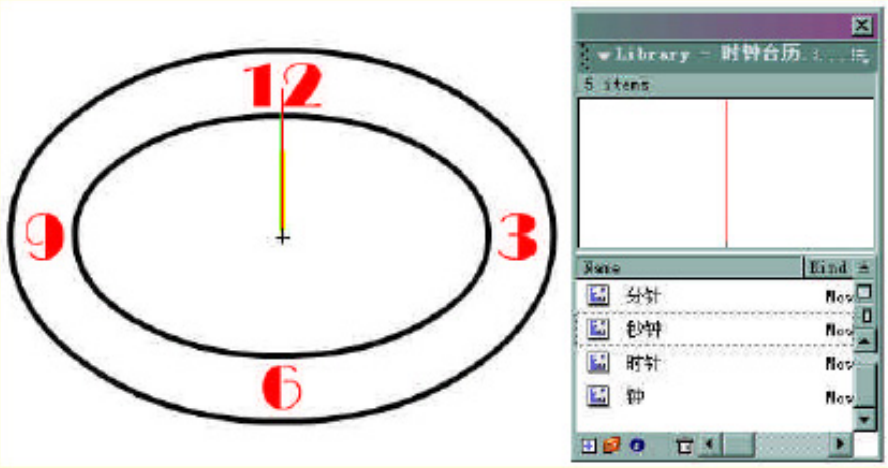


图 12-129　放置时针、分针和秒针

（8）调出 Properties（属性）面板，选中舞台中的“时针”电影剪辑元件的实例，在 Properties 面板的 Instance Name（实例名）文本框内输入 hourhand，将“时针”电影剪辑元件的实例命名为 hourhand。然后依照同样的步骤将“分针”、“秒针”的电影剪辑元件的实例依次命名为 minutehand、secondhand，如图 12-130 所示。

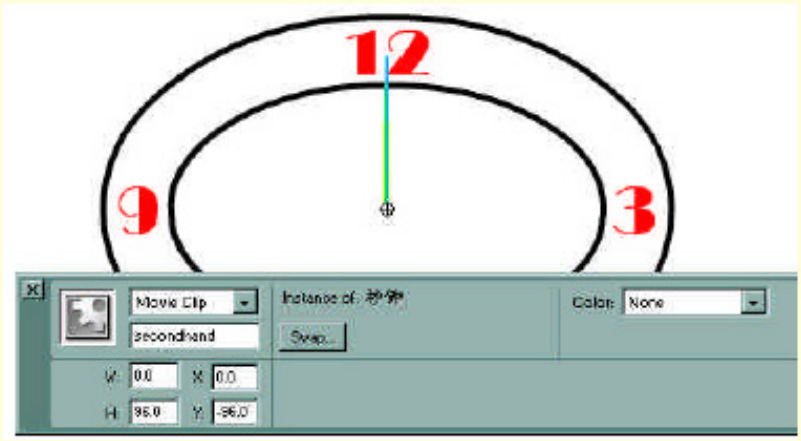


图 12-130　命名电影剪辑实例

（9）在“钟”电影剪辑元件的时间轴上新建一个层，选中新建层的第一帧，按 F9 快捷键调出 Actions

面板，在脚本编辑区中加入如下代码：

```
mytime = new Date ();
sec = mytime.getSeconds ();
min = mytime.getMinutes ();
minutehand._rotation = min*6;
secondhand._rotation = sec*6;
hour = mytime.getHours ()-1;
if (hour>12) {
    hour = hour-12;
}
hourhand._rotation = hour*30;
```

然后，单击当前时间轴的第 2 帧，按 F5 快捷键插入一个普通帧。

（10）回到主场景，在主场景的时间轴上新建一个层，将“钟”电影剪辑元件从元件库中拖曳到主场景舞台中适当位置，如图 12-131 所示，这样桌面时钟台历的时钟部分便做好了。

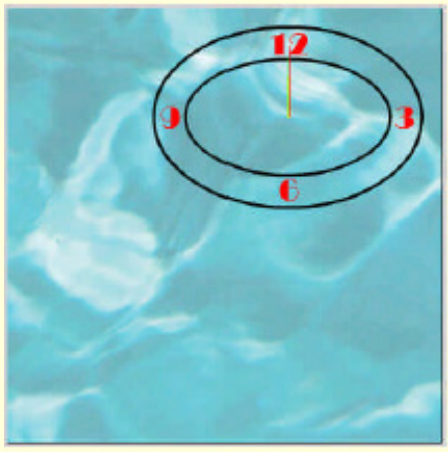


图 12-131　放置表盘

（11）下面的步骤是制作台历。在主场景的时间轴上新建一个层，用 Text（文字）工具在舞台的适当位置输入一些提示性文字，如图 12-132 所示。分别在“年”、“月”、“日”三个字前面和“星期”后面加一个动态文本框，依次命名为 year、month、day、week，如图 12-133 所示。

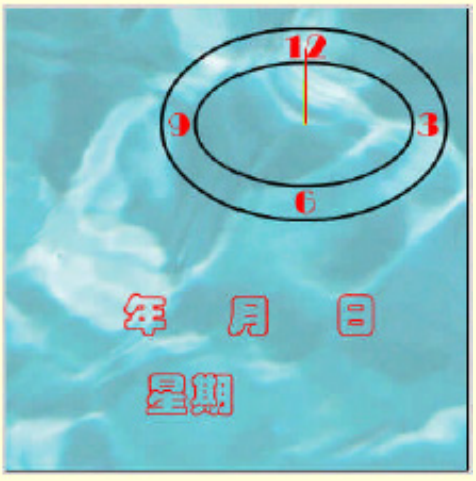


图 12-132　输入提示性文字

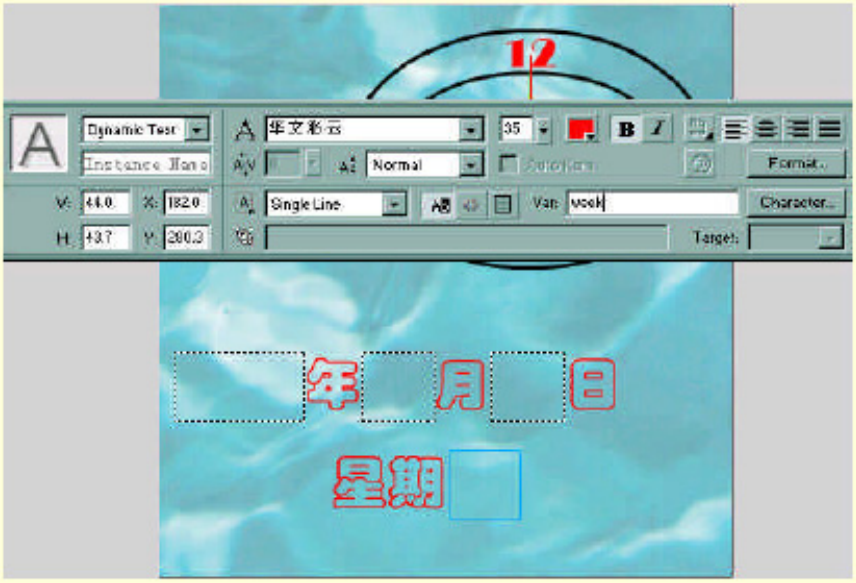


图 12-133　放置动态文本框

（12）调出 Actions 面板，然后单击层的第一帧，在脚本编辑区加入如下代码：

```
mydate = new Date ();
myarray = new Array ("日","一","二","三","四","五","六");
_root.week = myarray[mydate.getDay ()];
_root.year = mydate.getFullYear ();
_root.month = mydate.getMonth ()+1;
_root.day = mydate.getDate ();
```

这样，桌面时钟台历的台历部分也做好了。

（13）接下来要做的是将 Flash MX 动画嵌入到网页中。执行 File | Publish Settings 菜单命令，将会弹出 Publish Settings（发布设置）对话框。在 Formats（格式）标签中选中 HTML 项和 Flash 项，如图 12-134 所示。

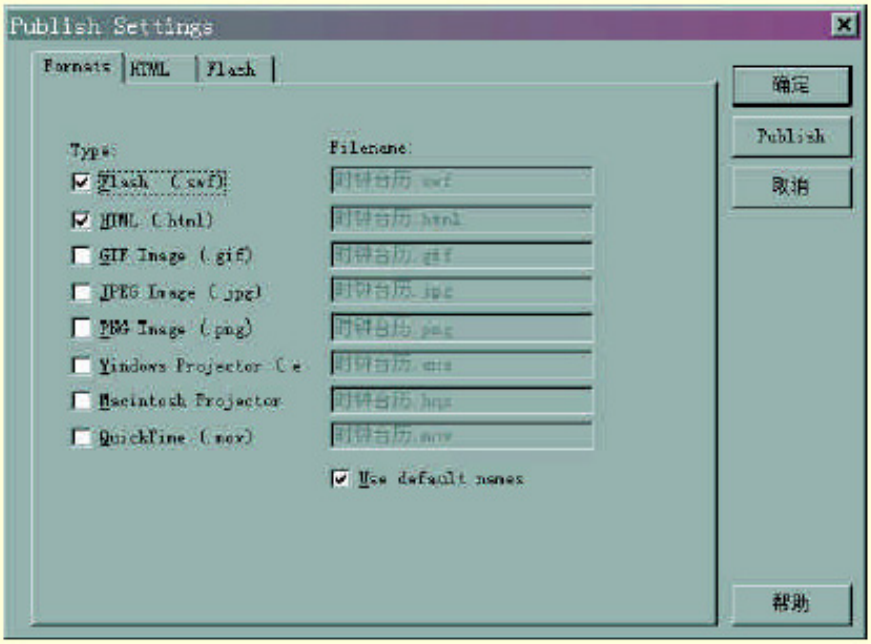


图 12-134　选择发布格式

（14）单击 HTML 标签，按图 12-135 所示进行 HTML 页面的发布设置，再单击 Flash 标签，按图 12-136 所示进行 Flash 动画文件的发布设置。设置好所有选项之后，单击 Publish Settings 对话框右边的 Publish 按钮，完成作品的发布，Flash MX 会将生成的 SWF 动画文件和 HTML 文件保存到当前 Flash MX 文档所在的目录中。

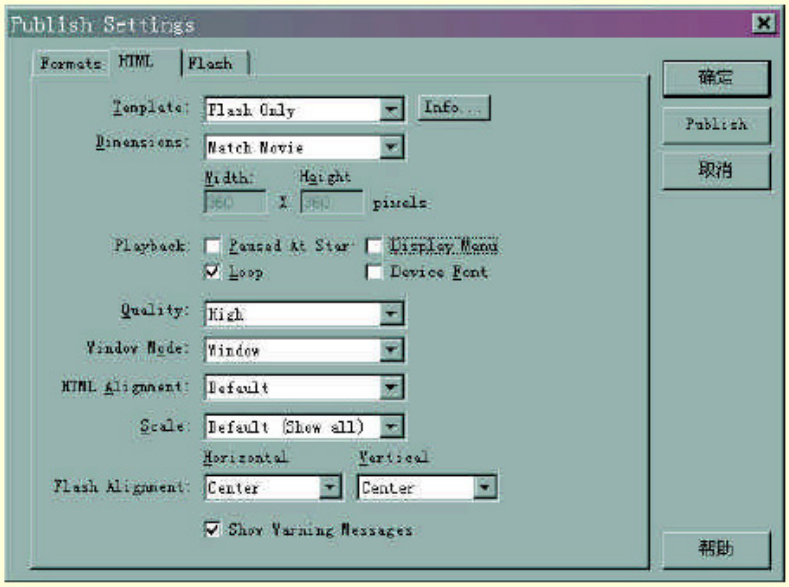


图 12-135　HTML 页面发布设置

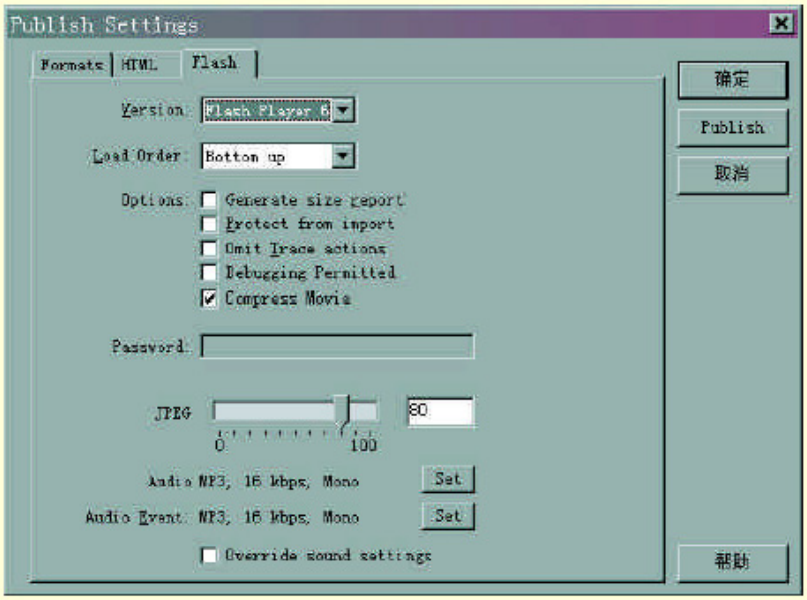


图 12-136　SWF 文件发布设置

（15）下面完成最后一步——将制做好的时钟台历放到 Windows 系统的桌面上。在 Windows 桌面上单击鼠标右健，然后在弹出的快捷菜单中选择“属性”，系统将弹出如图 12-137 所示的显示属性对话框，单击“浏览”按钮，选择刚发布的 HTML 页面文件，单击“确定”按钮，时钟台历就放到桌面上了。最终效果如图 12-138 所示。

图 12-137 设置 Windows 系统桌面

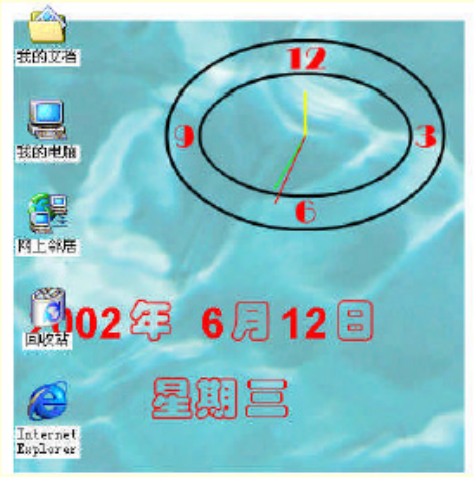


图 12-138 桌面时钟台历最终效果

12.8 打字练习软件

12.8.1 介 绍

本例将介绍如何使用 Flash MX 制作一个英文打字练习软件。首先，打字机产生 20 个随机字母，然后按回车键，练习者在通过键盘输入对应的字母，同时打字机的模拟键盘的对应健还会变成黄色，提示练习者该按哪个键。该软件还会记录练习者输入错误的字数，显示在屏幕中“ wrong :”的后面。打字机的界面如图 12-139 所示。

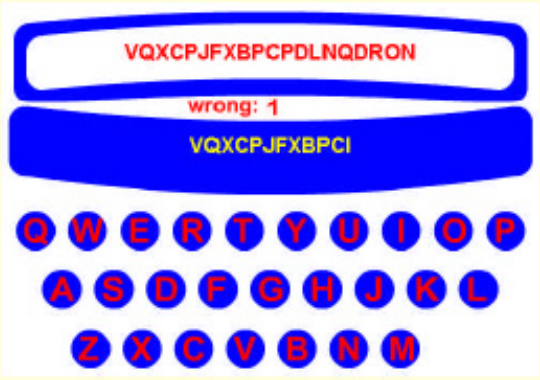


图 12-139 打字练习软件界面

12.8.2 实 战

（1）新建一个 Flash MX 文档，舞台设置使用 Flash MX 默认值即可。

（2）使用 Insert | New Symbol 菜单命令，新建一个电影剪辑元件，命名为“ A ”，作为打字机模拟键盘上的 A 字母键。

进入“ A ”电影剪辑元件的编辑状态，用 Oval（椭圆）工具在舞台中央画一个蓝色的圆，然后用 Text（文字）工具在圆形中央写一个“ A ”，如图 12-140 所示。然后单击当前时间轴的第二帧，按 F6 插入关键帧，再将圆形的填充色变为黄色，如图 12-141 所示。接下来在第一帧上加入一条 ActionScript 脚本代码：

```
Stop();
```

（3）按照上述方法，再制作其他 25 个字母键。

图 12-140 模拟键盘按键 A

图 12-141 按键 A 按下时的状态

（4）新建一个电影剪辑元件，命名为“ keyboard ”，然后将制作完成的 26 个字母键电影剪辑元件从元件库中拖曳到舞台中，按图 12-142 排列好。接下来将每个电影剪辑元件的实例命名为相应的键名，如将“ A ”电影剪辑元件实例命名为“ a ”。这样打字机的模拟键盘就制作完成了。

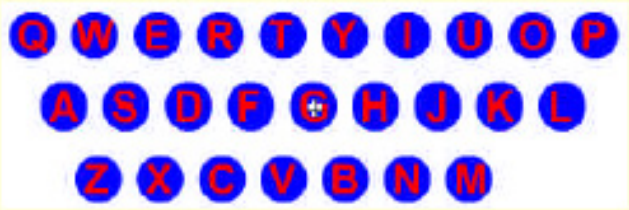


图 12-142 模拟键盘排列

（5）新建一个电影剪辑元件，命名为“ screen ”。然后进入其编辑状态，在舞台中绘制如图 12-143 所示的图像，颜色为蓝色。

图 12-143 绘制打字机的屏幕

（6）回到主场景的编辑状态中，将当前层命名为“ typewriter ”，再将“ keyboard ”电影剪辑元件和“ screen ”电影剪辑元件从元件库中拖曳到舞台中。然后在这一层的第 21 帧加入一个普通帧。如图 12-144 所示。

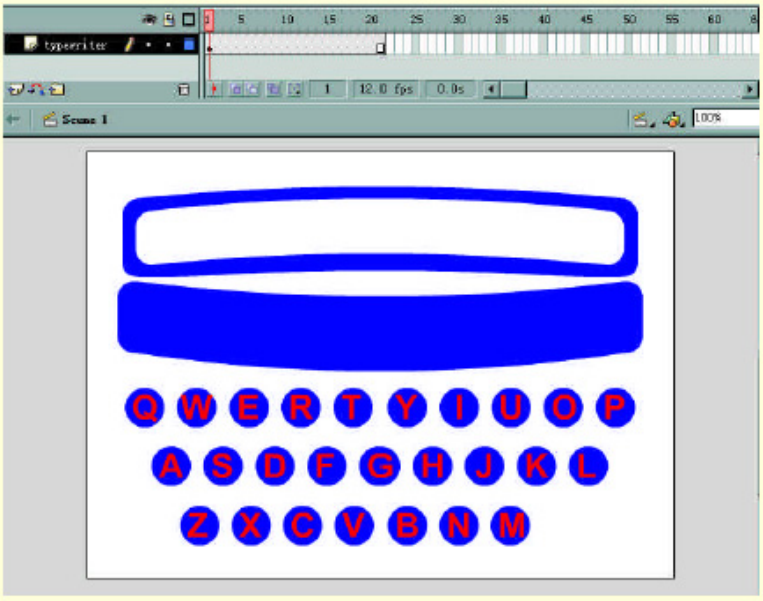


图 12-144 打字机界面

（7）新建一个层，命名为“ wrong ”，在如图 12-145 所示的位置上写上提示性单词“ wrong ”，再放置一个动态文本框，将其命名为“ wrong ”。wrong 变量用来记录练习者输入错误的次数。

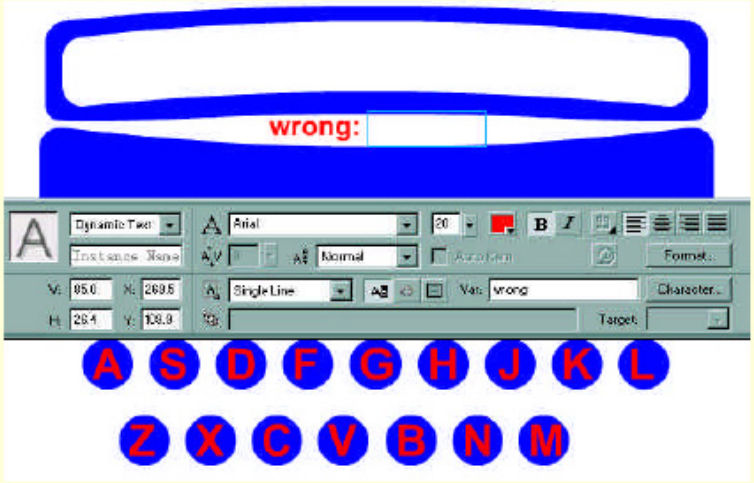


图 12-145　绘制错误提示界面

（8）再新建一个层，命名为“ text ”。在如图 12-146 所示的位置放置两个动态文本框，其中位于上面的命名为“ output ”，用来显示电脑随机产生的 20 个英文字母；下面的命名为“ input ”，用来显示练习者输入的英文字母。

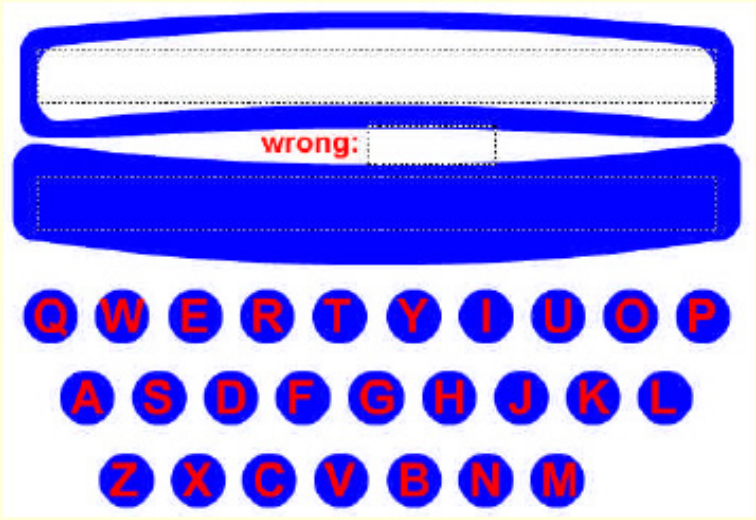


图 12-146　放置动态文本框

在“ text ”层的第 2~21 帧的每一帧新建一个关键帧，并在第 1~21 帧的每个关键帧上加入如下 ActionScript 代码：

```
generator ();
```

此时的时间轴如图 12-147 所示。

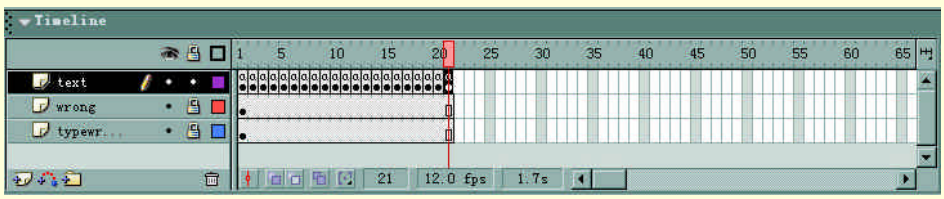


图 12-147　第（8）步的时间轴

（9）新建一个“ Action ”层用来放置 ActionScript 代码。选中 Action 层的第 1 帧，然后调出 Action

面板，在脚本编辑区输入如下代码：

```
keyArray = new Array (26);
keyString = new Array ();
output = new String ();
input = new String ();
inputState = “start”;
wrong = 0;
var n;
for (n=0;n<26;n++) {
keyArray[n] = chr (n+65);
}
function generator () {
var temp;
if (inputState = = “start”) {
temp = random (26);
keyString.push (keyArray[temp]);
output = output.concat (keyArray[temp]);
}
}
```

单击“ action ”层的第 21 帧，调出 Actions 面板，加入如下代码：

```
stop ();
inputState = ”input”;
```

（10）选中舞台中的“ keyboard ”电影剪辑元件的实例，然后，调出 Actions 面板，加入如下代码：

```
onClipEvent (keyDown) {
x = Key.getCode ();
this[chr (x)].play ();
if (Key.isDown(Key.Enter))
_root.play ();
if ((_root.inputState = = “input”)&&(_root.input.length<20)) {
_root.input = _root.input.concat (chr(x));
temp = _root.keyString.shift ();
if (temp<>chr(x))
_root.wrong++;
}
}
```

（11）将当前动画文件发布为 SWF 文件，然后再运行该文件，就可以进行打字练习了。如图 12-148 所示。

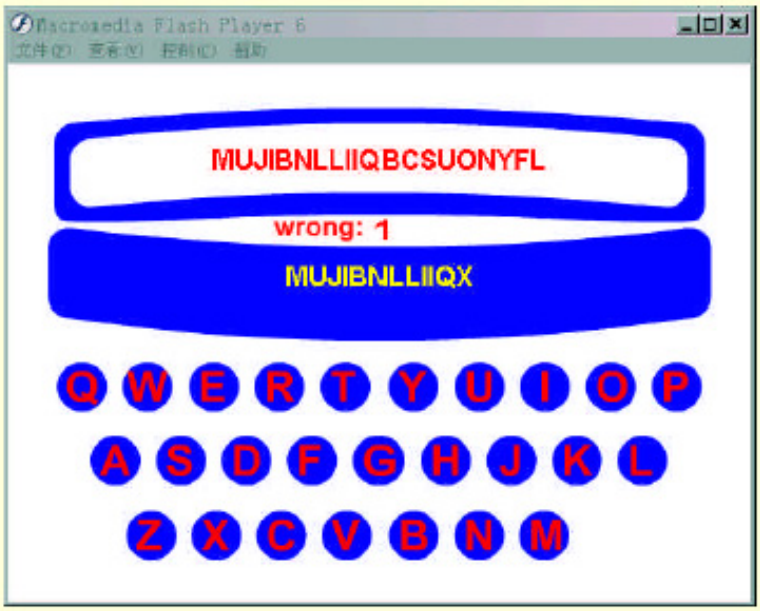


图 12-148　**运行打字练习软件**

附录 A　ActionScript 脚本语言参考

A.1　Functions 函数

函 数 名	功　　能	语　　法
Boolean	取表达式的布尔值（True 或 False）	
escape	除去 URL 串中的非法字符	escape(expression);
eval	返回由表达式命名的变量的值	eval(expression);
false	布尔“假”，值等于 0	
getProperty	获取对象的属性	getProperty(instancename , property);
getTimer	获取从电影开始播放到现在的总播放时间（毫秒数）	getTimer();
getVersion	获取浏览器插件的 FlashPlayer 的版本号	getVersion();
int	把数值强制转换成整数	int(value);
isFinite	测试数值是否为有限数	isFinite(expression);
isNaN	测试是否为非数值	isNaN(expression);
maxscroll	文本框的最大长度	variable_name.maxscroll = x
newline	换行符	
number	将参数转换成数值	Number(expression);
pareseFloat	将字符串转换成浮点数	parseFloat(string);
parseInt	将字符串转换成小数	parseInt(expression, radix);
random	产生 0 到指定数间的随机数	random(value);
scroll	文本框中的当前行	variable_name.scroll = x
String	将参数转换成字符串	String(expression);
targetPath	返回指定实体 Movie Clip 的路径字符串	targetpath(movieClipObject);
true	布尔“真”值，等于 0	
unescape	保留字符串中的%XX 格式的十六进制字符	unescape(x);

A.2　字符串函数

函 数 名	功　　能	语　　法
chr	将 ASCII 码转化成相应字符	chr(number);
length	返回字符串的长度	length(expression);或 length(variable);

续 表

函 数 名	功 能	语 法
mbchr	将 ASCII/S-JIS 编码转换成相应的多字节字符	mbchr(number);
mblength	返回多字节字符串的长度	mblength(string);
mbord	将多字节字符转换成相应的 ASCII/S-JIS 编码	mbord(character);
mbsubstring	截取多字节字符串中的字串	mbsubstring(value, index, count);
ord	将字符转换成 ASCII 码	ord(character);
substring	截取字符串中的字串	substring(string, index, count);

A.3 Properties 属性

函 数 名	功 能	语 法
_alpha	Aplha 值（透明度值）	instancename._alpha instancename._alpha = value;
_currentframe	在 Movie Clip 中的当前帧数	instancename._currentframe
_droptarget	正在拖动的 Movie Clip 是否播完	draggableInstanceName._droptarget
_focusrect	焦点矩形框的显示与否（True 显示/Flase 不显示）	instancename._focusrect = Boolean;
_framesloaded	载入的影帧数	instancename._framesloaded
_height	Movie Clip 的高度	instancename._height nstancename._height = value;
_highquality	画质的高低（True 高画质/False 低画质）	_highquality = value;
_name	Movie Clip 的实体名	instancename._name instancename._name = value;
_quality	当前画质（字符串值）LOW,MEDIUM,HIGH,BEST	_quality _quality = x;
_rotation	Movie Clip 旋转的角度（单位：度）	instancename._rotation instancename._rotation = integer;
_soundbuftime	声音的缓冲区大小（默认值 5，单位：秒）	_soundbuftime = integer;
_target	Movie Clip 的路径	instancename._target
_totalframes	总帧数	instancename._totalframes
_url	电影被调用的 URl 地址	instancename._url
visible	是否可视（True 可视/False 不可视）	instancename._visible instancename._visible =Boolean;
_width	Movie Clip 的宽度	instancename._width instancename._width =value;

续 表

函 数 名	功 能	语 法
_x	Movie Clip 的 x 坐标	instancename._x instancename._x = integer;
_xmouse	鼠标的 x 坐标	instancename._xmouse
_xscale	Movie Clip 的 x 轴向缩放度	instancename._xscale instancename._xscale = percentage;
_y	Movie Clip 的 y 坐标	instancename._y instancename._y = integer;
_ymouse	鼠标的 y 坐标	instancename._ymouse
_yscale	Movie Clip 的 y 轴向缩放度	instancename._yscale instancename._yscale = percentage;

A.4 对 象

Array（数组）

函 数 名	功 能	语 法
concat	合并多个数组	myArray.concat(value0,value1,...valueN);
join	合并数组元素位字符串	myArray.join(); myArray.join(separator);
lenth	返回数组长度	myArray.length;

new Array	新建数组物件	new Array();
		new Array(length);
		new Array(element0, element1, element2,...elementN);

pop	出栈（FILO 先入后出）	myArray.pop();
push	入栈（入队列）	myArray.push(value,...);
reverse	反相（颠倒数组，第一个元素和最后一个元素互换.....）	myArray.reverse();
shift	出队列（FILO 先入后出）	myArray.shift();
slice	截取数组中的子串生成新的数组	myArray.slice(start, end);
sort	数组元素的排序	myArray.sort(); myArray.sort(orderfunc);
splice	从数组中指定元素起删除指定个数的元素或者删除指定元素	myArray.splice(start, deleteCount, value0,value1...valueN);
unshift	从数组头部插入一个元素	myArray.unshift(value1,value2,...valueN);

布尔数

函 数 名	功 能	语 法
new Boolean	新建布尔型物件	new Boolean(); new Boolean(x);
toString	将布尔型物件的值转换成字符串值	Boolean.toString();
valueOf	获取布尔型物件的值（返回值为布尔型）	Boolean.valueOf();

Color（颜色）

函 数 名	功 能	语 法
getRGB	获取颜色值的 RGB 分量（返回值 0xRRGGBB 十六进制）	myColor.getRGB();
getTransform	获取颜色值的转换量（参数为 cxform 型）	myColor.getTransform();
new Color	新建一颜色物件	new Color(target);
setRGB	设置颜色值的 RGB 分量(参数为 0xRRGGBB 十六进制）	myColor.setRGB(0xRRGGBB);
setTransform	设置颜色值的转换量（参数为 cxform 型）	myColor.setTransform(colorTransformObject);

Date（时间和日期）

函 数 名	功 能	语 法
getDate	获取当前日期（本月的几号）	myDate.getDate();
getDay	获取今天是星期几（0——Sunday,1——Monday...）	myDate.getDay();
getFullYear	获取当前年份（四位数字）	myDate.getFullYear();
getHours	获取当前小时数（24 小时制，0~23）	myDate.getHours();
getMilliseconds	获取当前毫秒数	myDate.getMilliseconds();
getMinutes	获取当前分钟数	myDate.getMinutes();
getMonth	获取当前月份（从 0 开始：0——Jan，1——Feb...）	myDate.getMonth();
getSeconds	获取当前秒数	myDate.getSeconds();
getTime	以自 1970.1.1 0：00 以来的秒数为基准，获取 UTC	myDate.getTime();
getTimezoneOffset	获取当前时间和 UTC 格式的偏移值（以分钟为单位）	mydate.getTimezoneOffset();
getUTCDate	获取 UTC 格式的当前日期（本月的几号）	myDate.getUTCDate();
getUTCDay	获取 UTC 格式的今天是星期几（0——Sunday,1——Monday...）	myDate.getUTCDay();
getUTCFullYear	获取 UTC 格式的当前年份（四位数字）	myDate.getUTCFullYear();
getUTCHours	获取 UTC 格式的当前小时数（24 小时制，0~23）	myDate.getUTCHours();
getUTCMilliseconds	获取 UTC 格式的当前毫秒数	myDate.getUTCMilliseconds();
getUTCMinutes	获取 UTC 格式的当前分钟数	myDate.getUTCMinutes();

续 表

函 数 名	功 能	语 法
getUTCMonth	获取 UTC 格式的当前月份 （从 0 开始：0——Jan，1——Feb...）	myDate.getUTCMonth();
getUTCSeconds	获取 UTC 格式的当前秒数	myDate.getUTCSeconds();
getYear	获取当前缩写年份（当前年份减去 1900）	myDate.getYear();
new Date	新建日期时间物件	new Date(); new Date(year [, month [, date [, hour [, minute [, second [, millisecond]]]]]]);
setDate	设置当前日期（本月的几号）	myDate.setDate(date);
setFullYear	设置当前年份（四位数字）	myDate.setFullYear(year [, month [, date]]);
setHours	设置当前小时数（24 小时制，0~23）	myDate.setHours(hour);
setMilliseconds	设置当前毫秒数	myDate.setMilliseconds(millisecond);
setMinutes	设置当前分钟数	myDate.setMinutes(minute);
setMonth	设置当前月份（从 0 开始：0——Jan，1——Feb...）	myDate.setMonth(month[,date]);
etSeconds	设置当前秒数	myDate.setSeconds(second);
setTime	以自 1970.1.1 0：00 以来的秒数为基准，设置 UTC	myDate.setTime(millisecond);
setUTCDate	设置 UTC 格式的当前日期（本月的几号）	myDate.setUTCDate(date);
setUTCFullYear	设置 UTC 格式的当前年份（四位数字）	myDate.setUTCFullYear(year [, month [, date]]);
setUTCHours	设置 UTC 格式的当前分钟数	myDate.setUTCHours(hour [, minute [, second [, millisecond]]]));
setUTCMilliseconds	设置 UTC 格式的当前毫秒数	myDate.setUTCMinutes(minute [, second [, millisecond]]));
setUTCMinutes	设置 UTC 格式的当前分钟数	myDate.setUTCMinutes(minute [, second [, millisecond]]));
setUTCMonth	设置 UTC 格式的当前月份（从 0 开始：0——Jan，1——Feb...）	myDate.setUTCMonth(month [, date]);
setUTCSeconds	设置 UTC 格式的当前秒数	myDate.setUTCSeconds(second [, millisecond]));
setYear	设置当前缩写年份（前年份减去 1900）	myDate.setYear(year);
toString	将日期时间值转换成“日期/时间”形式的字符串值	myDate.toString();
UTC	返回指定的 UTC 格式日期时间的固定时间值	Date.UTC(year,month[,date[,hour[,minute[,second[,millisecond]]]]]);

Key（键盘）

函 数 名	功　　能	语　　法
BACKSPACE	Backspace 键	Key.BACKSPACE
CAPSLOCK	CapsLock 键	Key.CAPSLOCK
CONTROL	Ctrl 键	Key.CONTROL
DELETEKEY	Delete(Del)键	Key.DELETEKEY
DOWN	方向下键	Key.DOWN
END	End 键	Key.END
ENTER	Enter（回车）键	Key.ENTER
ESCAPE	Esc 键	Key.ESCAPE
getAscii	获取最后一个按下或松开的键的对应字符的 ASCII 码	Key.getAscii();
getCode	获取最后一个按下或松开的键的键盘扫描码	Key.getCode();
HOME	Home 键	Key.HOME
INSERT	Insert（Ins）键	Key.INSERT
isDown	当指定键被按下时返回 True 值	Key.isDown(keycode);
isToggled	当指定键被锁定时返回 True 值	Key.isToggled(keycode)
keycode	返回键盘按下键的键值	
LEFT	方向左键	Key.LEFT
PGDN	PageDown 键	Key.PGDN
PGUP	PageUp 键	Key.PGUP
RIGHT	方向右键	Key.RIGHT
SHIFT	Shift 键	Key.SHIFT
SPACE	空格键	Key.SPACE
TAB	Tab 键	Key.TAB
UP	方向上键	Key.UP

Math（数学函数）

函 数 名	功　　能	语　　法
abs	abs（n）取 n 的绝对值	Math.abs(x);
acos	acos（n）n 的反余弦（返回值单位：弧度）	Math.acos(x);
asin	asin（n）n 的反正弦（返回值单位：弧度）	Math.asin(x);
atan	atan（n）n 的反正切（返回值单位：弧度）	Math.atan(x);
atan2	atan2（x,y）计算 x/y 的反正切	Math.atan2(y, x);
ceil	ceil（n）取靠近 n 的上限的整数(向上取整)	Math.ceil(x);
cos	cos（n）取 n 的余弦（n 单位：弧度）	Math.cos(x);
E	Euler（欧拉）指数（约为 2.718）	Math.E

续 表

函 数 名	功 能	语 法
exp	指数	Math.exp(x);
floor	floor（n）取靠近 n 的下限的整数(向下取整)	Math.floor(x);
LN10	ln10（约等于 2.302）	Math.LN10
LN2	ln2（约等于 0.693）	Math.LN2
log	取自然对数	Math.log(x);
LOG10E	10 为底取 E 的对数（约等于 0.434）	Math.LOG2E
LOG2E	2 为底取 E 的对数（约等于 1.443）	Math.LOG10E
max	返回两参数中的最大值	Math.max(x , y);
min	返回两参数中的最小值	Math.min(x , y);
PI	圆周率（约等于 3.1415926）	Math.PI
pow	pow（x,y）x 的 y 次方	Math.pow(x , y);
random	产生 0~1 间的随机数	Math.random();
round	四舍五入取整	Math.round(x);
sin	sin（n）取 n 正弦（n 单位：弧度）	Math.sin(x);
sqrt	开根号	Math.sqrt(x);
SQRT1_2	0.5 开根号（约等于 0.707）	Math.SQRT1_2
SQRT2	2 开根号（约等于 1.414）	Math.SQRT2
tan	tan（n）取 n 的正切（n 单位：弧度）	Math.tan(x);

Movie Clip（电影剪辑）

函 数 名	功 能	语 法
attachMovie	绑定一个电影，产生一个库中 Movie Clip 的实体	anyMovieClip.attachMovie(idName, newname, depth);
dupicateMovieClip	复制当前 Movie Clip 为新的 Movie Clip	anyMovieClip.duplicateMovieClip(newname, depth);
getURL	使浏览器浏览指定页面	anyMovieClip.getURL(URL[,window, variables]]);
globalToLocal	场景（Scene）中的坐标转换成 Movie Clip 中的坐标	anyMovieClip.globalToLocal(point);
gotoAndPlay	跳转到指定帧并播放	anyMovieClip.gotoAndPlay(frame);
gotoAndStop	跳转到指定帧并停止播放	anyMovieClip.gotoAndStop(frame);
hitTest	测试一个点或者 Movie Clip 是否同另一个交叉	anyMovieClip.hitTest(x, y, shapeFlag); anyMovieClip.hitTest(target);
loadMovie	引入一个外部电影到指定层	anyMovieClip.loadMovie(url [,variables]);
loadVariables	引入外部文件中的变量值	anyMovieClip.loadVariables(url, variables);

续 表

函 数 名	功 能	语 法
localToGlobal	Movie Clip 中的坐标转换成场景(Scene)中的坐标	anyMovieClip.localToGlobal(point);
nextFrame	下一帧	anyMovieClip.nextFrame();
play	播放	anyMovieClip.play();
prevFrame	上一帧	anyMovieClip.prevFrame();
removeMovieClip	删除用 dupicateMovieClip 创建的 Movie Clip	anyMovieClip.removeMovieClip();
startDrag	开始拖动 Movie Clip	anyMovieClip.startDrag([lock, left, right, top, bottom]);
stop	停止 Movie Clip 的播放	anyMovieClip.stop();
stopDrag	停止拖动 Movie Clip	anyMovieClip.stopDrag();
unloadMovie	卸载由 Movie Clip 引入的 Movie	anyMovieClip.unloadMovie();

Mouse（鼠标）

函 数 名	功 能	语 法
hide	隐藏鼠标指针	Mouse.hide();
show	显示鼠标指针	Mouse.show();

Number（数值）

函 数 名	功 能	语 法
MAX_VALUE	FLASH5 所允许的最大数值 1.79769313486231e308	Number.MAX_VALUE
MIN_VALUE	FLASH5 所允许的最小数值 5e-324	Number.MIN_VALUE
NaN	是否为非数值（Not a Number）	Number.NaN
NEGATIVE_INFINITY	是否为负数	Number.NEGATIVE_INFINITY
new Number	新建数值物件	myNumber = new Number(value);
POSITIVE_INFINITY	是否为正数	Number.POSITIVE_INFINITY
toString	将数值转换成字符串	myNumber.toString(radix);

Object（对象）

函 数 名	功 能	语 法
new Object	新建一个对象	new Object();new Object(value);
toString	转换对象为字符串	myObject.toString();
valueOf	返回对象的值	myNumber.valueOf();

Selection（选择区）

函 数 名	功 能	语 法
getBeginIndex	获取可编辑文本区的起始位置，-1 表示无可编辑文本区	Selection.getBeginIndex();
getCareIndex	获取当前的编辑位置，-1 表示无可编辑文本区	Selection.getCaretIndex();
getEndIndex	获取可编辑文本区的结束位置，-1 表示无可编辑文本区	Selection.getEndIndex();
getFocus	获取当前的激活文本区的文本变量名	Selection.getFocus();
setFocus	设置当前的激活文本区	Selection.setFocus(variable);
setSelection	设置可编辑文本的起始位置和终止位置	Selection.setSelection(start, end);

Sound（声音）

函 数 名	功 能	语 法
attachSound	绑定库中的一个声音	mySound.attachSound("idName");
getPan	获取声音的混音值	mySound.getPan();
getTransform	获取当前声音的变换量（返回值类型：sxform）	mySound.getTransform();
getVolume	获取当前声音的音量（百分比）	mySound.getVolume();
new sound	新建声音物件	new Sound(); new Sound(target);
setPan	设置声音的混音值	mySound.setPan(pan);
setTransform	设置当前声音的变换量（参数类型：sxform）	mySound.setTransform(soundTransformObject);
setVloume	设置当前声音的音量（百分比）	mySound.setVolume(volume);
start	开始播放当前声音	mySound.start(); mySound.start([secondOffset, loop]);
stop	停止播放当前声音	mySound.stop(); mySound.stop(["idName"]);

String（字符串）

函 数 名	功 能	语 法
charAt	在指定的索引表中返回一个字符	myString.charAt(index);
charCodeAt	在指定的索引表中返回一个字符的代码	myString.charCodeAt(index);
concat	连接合并多个字符串	myString.concat(value1,...valueN);
fromCharcode	从字符代码组构造出一个新的字符串	myString.fromCharCode(c1,c2,...cN);
indexOf	在字符串中寻找子串，返回字串起始位置或-1（为找到）	myString.indexOf(value); myString.index of (value, start);
lastIndexOf	在字符串中寻找子串，返回字串终止位置或-1（为找到）	myString.lastIndexOf(substring); myString.lastIndexOf(substring, start);

续 表

函 数 名	功 能	语 法
length	返回字符串的长度	string.length
new String	新建字符串物件	new String(value);
slice	返回字符串中指定截取的子串	myString.slice(start, end);
split	根据限定符将字符串转换成一个数组	myString.split(delimiter);
substr	substr(start,length)返回从 start 开始共 lenth 长的子串	myString.substr(start, length);
substring	substring（indexA,indexB）返回 indexA...indexB 之间的子串	myString.substring(from, to);
toLowerCase	将字符串的大写字符全部转换成小写	myString.toLowerCase();

XML（可扩展标记语言）

函 数 名	功 能	语 法
appendChild	添加一个子节点到指定的 XML 元素	myXML.appendChild(childNode);
attributes	XML 元素的属性数组	myXML.attributes;
childNodes	一个 XML 元素的子节点数组	myXML.childNodes;
cloneNode	复制当前节点	myXML.cloneNode(deep);
createElement	新建一个 XML 元素	myXML.createElement(name);
createTextNode	新建一个 XML 文本节点	myXML.createTextNode(text);
firstChild	返回当前 XML 节点的第一个子节点	myXML.firstChild;
hasChildNodes	当前 XML 节点是否有子节点（True 有子节点/False 无子节点）	myXML.hasChildNodes();
insertBefore	在指定的 XML 元素的子节点前插入一个新的子节点	myXML.insertBefore(childNode, beforeNode);
lastChild	返回当前 XML 的最后一个子节点	myXML.lastChild;
load	从指定的 URL 把 XML 元素引入 Flash 中	myXML.load(url);
loaded	当 XML 元素引入或是发送同时引入 Flash 中后，返回 True 值	myXML.loaded;
new XML	新建一个 XML 物件	new XML(); new XML(source);
nextSibling	当前 XML 节点的下一个节点	myXML.nextSibling;
nodeName	返回当前 XML 节点的名字	myXML.nodeName;
nodeType	返回当前 XML 节点的类型	myXML.nodeType;
4411nodeValue	返回当前 XML 节点的值	myXML.nodeValue;
onLoad	当引入或发送同时引入触发事件	myXML.onLoad(success);
parentNode	返回当前 XML 节点的父节点	myXML.parentNode;
parseXML	将 XML 字符串转换成 XML 物件	myXML.parseXML(source);

续 表

函 数 名	功 能	语 法
previousSibling	当前 XML 节点的前一个节点	myXML.previousSibling;
removeNode	从 XML 文本中删除节点	myXML.removeNode();
send	从 Flash 中把一个 XML 元素发送到指定的 URL 地址	myXML.send(url); myXML.send(url, window);
sendAndLoad	从 Flash 中把一个 XML 元素发送到指定的 URL 地址，同时引入 XML 结果	myXML.sendAndLoad(url,targetXMLobject);
toString	把 XML 物件转换为 XML 字符串表达	myXML.toString();

XMLSocket（XML 套接口）

函 数 名	功 能	语 法
close	关闭一个 XML 套接口	myXMLSocket.close();
connect	连接一个 XML 套接口，指定 URL 并定义其端口号	myXMLSocket.connect(host, port);
new XMLSocket	建立一个新的 XML 套接口	new XMLSocket();
onClose	关闭 XML 套接口时触发事件	myXMLSocket.onClose();
onConnect	连接 XML 套接口时触发事件	myXMLSocket.onConnect(success);
onXML	从服务器上获取 XML 时触发事件	myXMLSocket.onXML(object);
send	发送 XML 给服务器	myXMLSocket.send(object);

附录B　Flash MX 快捷键

B.1　菜单命令

File

命令名	快捷键	作　用
New	Ctrl+N	新建一个文件
Open	Ctrl+O	打开一个文件
Open as Library	Ctrl+Shift+O	以图库方式打开
Close	Ctrl+W	关闭一个文件
Save	Ctrl+S	存储一个文件
Save as	Ctrl+Shift+S	另存为
Import	Ctrl+R	导入外部动画素材
Export Movie	Ctrl+Alt+Shift+S	导出动画
Publish Settings	Ctrl+Alt+F12	发布设置
Publish	Alt+F12	发布
Print	Ctrl+P	打印
Exit	Ctrl+Q	退出 Flash

Edit

命令名	快捷键	作　用
Undo	Ctrl+Z	取消上一步操作
Redo	Ctrl+Y	重复上一步操作
Cut	Ctrl+x	剪切
Copy	Ctrl+C	复制
Paste	Ctrl+V	粘贴
Paste in Place	Ctrl+Alt+V	粘贴到新位置
Clear	Backspace	清除
Duplicate	Ctrl+D	复制
Select All	Ctrl+A	全部选择
Deselect All	Ctrl+Alt+A	取消选择
Cut Frames	Ctrl+Alt+X	剪切帧
Copy Frames	Ctrl+Alt+C	复制帧
Paste Frames	Ctrl+Alt+V	粘贴帧
Clear Frames	Alt+Backsapce	清除帧

续 表

命令名	快捷键	作 用
Select All Frames	Ctrl+Alt+A	选择所有帧
Edit Symbols	Ctrl+E	编辑元件
Preferences	Ctrl+U	设置

View

命令名	快捷键	作 用
Goto | First	Home	转到最前的场景
Goto | Previous	Page Up	转到前一个场景
Goto | Next	Page Down	转到下一个场景
Goto | Last	End	转到最后的场景
Zoom In	Ctrl+=	放大
Zoom Out	Ctrl+ー	缩小
Magnification | 100%	Ctrl+1	原始尺寸
Magnification | Show Frame	Ctrl+2	显示结构
Magnification | Show All	Ctrl+3	全部显示
Outlines	Ctrl+Alt+Shift+O	外边框
Fast	Ctrl+Alt+Shift+F	高速显示
Antialias	Ctrl+Alt+Shift+A	消除锯齿
Antialias Text	Ctrl+Alt+Shift+T	清除文字锯齿
Timeline	Ctrl+Alt+T	时间轴
Work Area	Ctrl+Shift+W	工作区
Rulers	Ctrl+Alt+Shift+R	标尺
Grid | Show Grid	Ctrl+’	显示网格
Grid | Snap to Grid	Ctrl+Shift+’	捕捉网格
Grid | Edit Grid	Ctrl+Alt+G	编辑网格
Guides | Show Guides	Ctrl+;	显示引导线
Guides | Lock Guides	Ctrl+Alt+;	锁定引导线
Guides | Snap to Guides	Ctrl+Shift+;	贴紧引导线
Guides | Edit Guides	Ctrl+Alt+Shift+G	编辑引导线
Snap to Objects	Ctrl+Shift+/	贴紧对象
Show Shape Hints	Ctrl+Alt+H	显示图形提示
Hide Edges	Ctrl+H	隐含锯齿
Hide Panels	F4	隐含面板

Insert

命令名	快捷键	作 用
Convert to Symbol	F8	转换成元件
New Symbol	Ctrl+F8	新建元件

续 表

命令名	快捷键	作 用
Frame	F5	帧
Remove Frames	Shift+F5	移除帧
ClearKeyframe	Shift+F6	清除关键帧

Modify

命令名	快捷键	作 用
Scene	Shift+F2	场景
Document	Ctrl+J	影片
Optimize	Ctrl+Shift+Alt+C	最优化
Shape | Add Shape Hint	Ctrl+Shift+H	添加形状提示
Transform | Scale and Rotate	Ctrl+Alt+S	比例旋转
Transform | Rotate 90 CW	Ctrl+Shift+9	顺时针旋转 90 度
Transform | Rotate 90 CCW	Ctrl+Shift+7	逆时针旋转 90 度
Transform | Remove Transform	Ctrl+Shift+Z	移除转换
Arrange | Bring to Front	Ctrl+Shift+Up	移到最前
Arrange | Bring Forward	Ctrl+Up	向前移动
Arrange | Send Backward	Ctrl+Down	向后移动
Arrange | Send to Back	Ctrl+Shift+Down	移到最后
Arrange | Lock	Ctrl+Alt+L	锁定
Arrange | Unlock All	Ctrl+Alt+Shift+L	全部解除锁定
Align | Left	Ctrl+Alt+1	左对齐
Align | Center Vertical	Ctrl+Alt+2	垂直居中对齐
Align | Right	Ctrl+Alt+3	右对齐
Align | Top	Ctrl+Alt+4	顶对齐
Align | Center Horizontal	Ctrl+Alt+5	水平居中对齐
Align | Bottom	Ctrl+Alt+6	底对齐
Align | Distribute Width	Ctrl+Alt+7	按宽度分布
Align | Distribute Height	Ctrl+Alt+9	按高度分布
Align | Make Same Width	Ctrl+Alt+Shift+7	宽度相同
Align | Make Same Height	Ctrl+Alt+Shift+9	高度相同
Align | To Stage	Ctrl+Alt+8	对齐场景
Group	Ctrl+G	群组
Ungroup	Ctrl+Shift+G	撤消群组
Break Apart	Ctrl+B	分解
Distribute to Layers	Ctrl+Shift+D	分布到层

Text

命令名	快捷键	作　用
Style | Plain	Ctrl+Shift+P	正常
Style | Bold	Ctrl+Shift+B	粗体
Style | Italic	Ctrl+Shift+I	斜体
Align | Left	Ctrl+Shift+L	左侧排列
Align | Center	Ctrl+Shift+C	居中排列
Align | Right	Ctrl+Shift+R	右侧排列
Align | Justify	Ctrl+Shift+J	正中间
Tracking | Increase	Ctrl+Alt+Right	增大
Tracking | Decrease	Ctrl+Alt+Left	减小
Tracking | Reset	Ctrl+Alt+Up	重置

Control

命令名	快捷键	作　用
Play	Enter	播放
Rewind	Ctrl+Alt+R	重复
Step Forward	.	快进
Step Backward	,	快退
Test Movie	Ctrl+Enter	测试影片
Debug Movie	Ctrl+Shift+Enter	调试影片
Test Scene	Ctrl+Alt+Enter	测试场景
Enable Simple Buttons	Ctrl+Alt+B	启用简单按钮

Window

命令名	快捷键	作　用
New Window	Ctrl+Alt+N	新建窗口
Tools	Ctrl+F2	工具栏
Timeline	Ctrl+Alt+T	时间轴
Properties	Ctrl+F3	属性面板
Answers	Alt+F1	Answers 面板
Align	Ctrl+K	对齐面板
Color Mixer	Shift+F9	调色面板
Color Swatches	Ctrl+F9	样本色面板
Info	Ctrl+I	信息面板
Scene	Shift+F2	场景面板
Transform	Ctrl+T	变形面板
Actions	F9	动作面板
Debugger	Shift+F4	调试面板

续 表

命令名	快捷键	作 用
Movie Explorer	Alt+F3	影片浏览器面板
Reference	Shift+F1	参考面板
Output	F2	输出面板
Accessibility	Alt+F2	Accessibility 面板
Component	Ctrl+F7	组件面板
Component Parameters	Alt+F7	组件参数面板
Library	F11	图库面板

B.2 工具箱中的快捷键

命令名	快捷键	作 用
Arrow Tool	V	箭头工具
Subselect Tool	A	选取工具
Line Tool	N	线条工具
Lasso Tool	L	套索工具
Pen Tool	P	钢笔工具
Text Tool	T	文本工具
Rectangle Tool	R	矩形工具
Oval Tool	O	椭圆工具
Pencil Tool	Y	铅笔工具
Brush Tool	B	画笔工具
Free Transform Tool	Q	自由变换工具
Fill Transform Tool	F	填充变换工具
Ink Bottle Tool	S	墨水瓶工具
Paint Bucket Tool	K	油漆桶工具
Dropper Tool	I	点滴器工具
Eraser Tool	E	橡皮擦工具
Hand Tool	H	手形工具
oom Tool	M、Z	放大镜工具