第 4 章　Flash 游戏基础

本章重点

- 动作、事件、对象、属性概念
- 变量、数据类型、运算符概念
- 程序控制语句的使用

本章难点

- 程序控制语句的使用
- 常用动作的使用和制作

学习目标

- 掌握 ActionScript 基本知识和概念
- 理解程序控制语句
- 掌握常用动作的使用

ActionScript（动作脚本）是 Flash 的脚本撰写语言，将其添加在 Flash 中，可以增加 Flash 文档的交互性，更好地控制 Flash 文档。使用 Flash 编程可以实现很多功能，比如按钮的响应、场景的跳转、网页的链接、动态装载 SWF 文件、交互游戏等。本章将介绍 ActionScript 语言基础知识和基本操作，讲解如何在 Flash 中添加简单的脚本。

4.1 ActionScript 基础

ActionScript 拥有语法、变量、函数等，与 JavaScript 类似，它也由许多行语句组成，每行语句又是由一些命令、运算符、分号等组成。通过应用 ActionScript，能够突破时间轴的应用而表现高级动画。ActionScript 最大的特点是实现 Flash 动画和用户之间的交互。简而言之，就是用户能够控制时间轴，跳转到指定的场景，播放特定的动画。用鼠标、键盘等，可以链接到特定的主页、发送邮件、加入多种效果。

4.1.1 ActionScript 制作效果

- 控制时间轴

应用按钮的 ActionScript 代码，可以播放或停止影片的播放。

- 跳转页面

菜单按钮，可以跳转到与菜单相关的影片。

- 键盘控制

设置特定的键盘按键，跳转到相关影片或页面。

- 更改对象属性

可以通过触发按钮等方法来控制对象的颜色、大小、位置和透明度等属性。

- 设置超链接

通过设置，链接到相关的站点。

- 拖动影片剪辑

拖动影片剪辑到指定位置。

- 制作进度条

显示影片的下载速度和进度，制作进度条动画。

- 制作游戏

制作出多样的游戏动画。

4.1.2 ActionScript 编辑窗口

ActionScript 实际是一种脚本语言，主要用来对动画进行编程，使用它可以使动画具有交互性，在一些动画中起到画龙点睛的效果。我们在 Flash 的“动作”面板中编辑代码，Flash 中的所有脚本语言都可以在这里找到。

可以选择“窗口”菜单中的“动作”命令打开“动作”面板，也可以按 F9 键打开“动作”面板，如图 4-1 所示。

“动作”面板工作界面如下：

1. 动作工具箱

动作工具箱可以选择不同动作脚本的语言类型。Flash CS3 动作面板的动作工具箱中有“ActionScript 1.0 & 2.0”和“ActionScript 3.0”等语言供用户选择。选择了语言后，在列表框中会显示出该种语言类型中的动作列表，如图 4-2 所示。

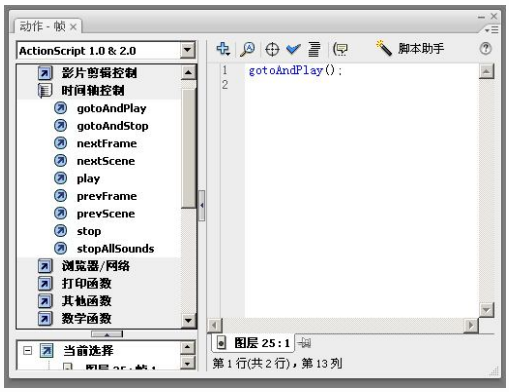

图 4-1 “动作”面板

图 4-2 动作工具箱

2. 脚本窗口

用于显示脚本和进行动作添加、语法检查、语法着色、自套用格式、代码提示、代码注释、代码折叠、自动换行等代码操作的窗口。

脚本窗口分为“手写”模式和“脚本助手”模式两种，如图 4-3 所示。

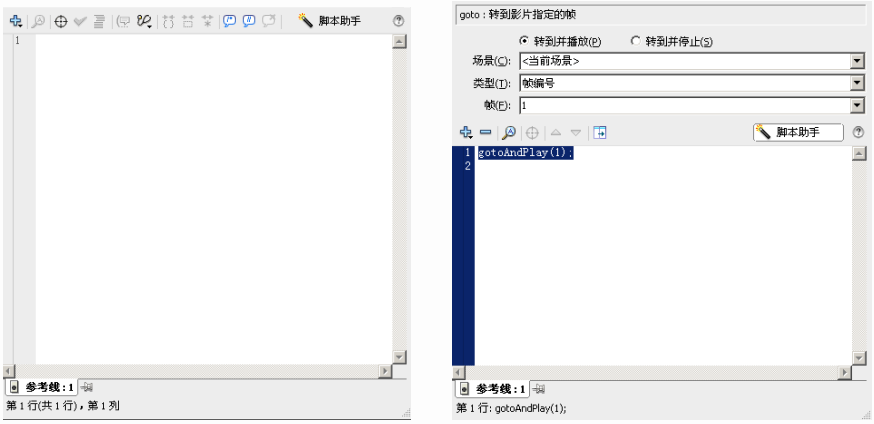

图 4-3 脚本助手

提示

脚本助手模式可以规范脚本，避免初学者编写代码时可能出现的语法和逻辑错误。

“手写”模式窗口一般是熟悉 ActionScript 用户所使用的模式，可直接输入脚本代码，如果在“首选参数”面板中设置了代码提示功能，在输入代码时会出现提示内容，如图 4-4 所示。

“脚本助手”模式窗口一般是 ActionScript 初学用户使用的模式，用户可通过选择动作工具箱中的项目来构建脚本。单击某个脚本项目，面板右上方会显示该项目的描述；双击某个项目，该项目就被添加到动作面板的“脚本”窗格中，如图 4-5 所示。

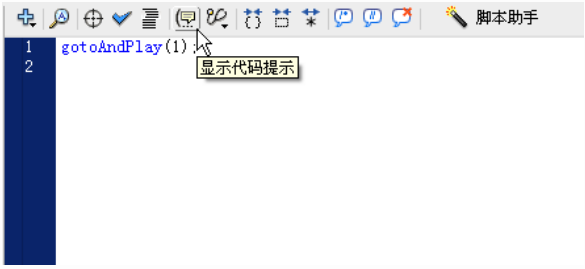

图 4-4　代码提示

3. 脚本导航器

显示当前文档中添加脚本的对象。单击脚本导航器的某一项目，与该项目相关的脚本将显示在“脚本”窗格中，并且播放头将移动到时间轴上相应位置，如图 4-6 所示。

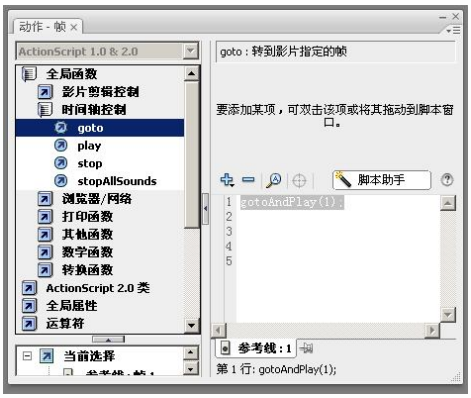

图 4-5　脚本助手

图 4-6　脚本导航器

提示

双击脚本导航器中的某一项目可固定脚本，将其锁定在当前位置，如图 4-7 所示。

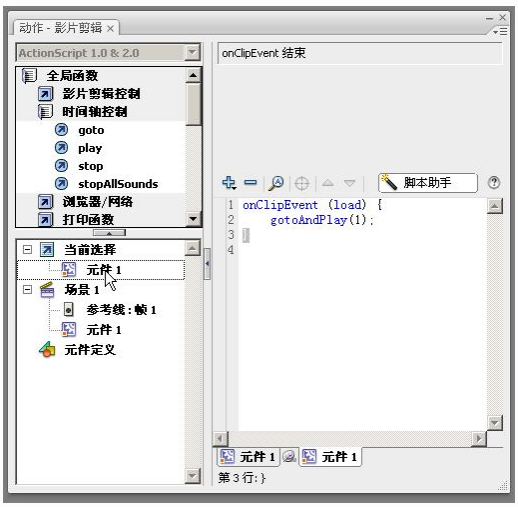

图 4-7　锁定

“动作”面板可以创建编辑对象或帧的 ActionScript 代码。选择相应的帧、影片剪辑元件或按钮元件后，再打开“动作”面板并添加动作代码。选择不同的对象添加脚本时，“动作”面板标题也会相应的命名为不同的对象动作。

在 Flash 中，我们可以对帧、影片剪辑元件和按钮元件添加动作（ActionScript 语句）。

4.2 ActionScript 编程基础

ActionScript 的语法是学习 ActionScript 编程的重点。只有对语法有了充分的了解，才能在编程过程中避免出现一些不必要的错误。下面以 ActionScript 2.0 为基础，讲解 ActionScript 编程的一些基础知识。

4.2.1 动作、事件和事件句柄

- 动作

在播放影片时指示影片执行某些任务的句子。例如：stop()。

发生某种情况：比如鼠标的单击、键盘按键或者动画播放到某一帧。

事件：在影片播放时发生的动作。

事件句柄：控制事件，应用于动作。

动作：发送命令，执行指定行为。

- 事件

播放到某帧、按鼠标或键盘中特定的键时，会触发并执行动作，将播放到某帧、按鼠标或键盘中特定的键这些事情称为事件。

帧事件：播放到某帧。

按钮事件：鼠标对按钮执行的单击或拖动等行为。

影片剪辑事件：鼠标、键盘对影片剪辑执行的各种行为以及影片剪辑的加载和卸载等。

- 事件句柄

加入了某个动作的帧、按钮和影片剪辑等发生事件时，控制和触发该动作的就是事件句柄。

帧事件句柄：

播放头进入该帧。

按钮事件句柄如图 4-8 所示。

on(press)：鼠标单击按钮时，要执行的代码或发生的事件。

on(release)：鼠标单击按钮后释放时，要执行的代码或发生的事件。

on (releaseOutside)：鼠标单击按钮，然后在外面释放时，要执行的代码或发生的事件。

on (rollOver)：将鼠标光标移动到按钮上时，要执行的代码或发生的事件。

on (rollOut)：将鼠标光标从按钮上移出时，要执行的代码或发生的事件。

on (dragOver)：鼠标单击按钮并拖动到外侧，然后重新移动到按钮上时，要执行的代码或发生的事件。

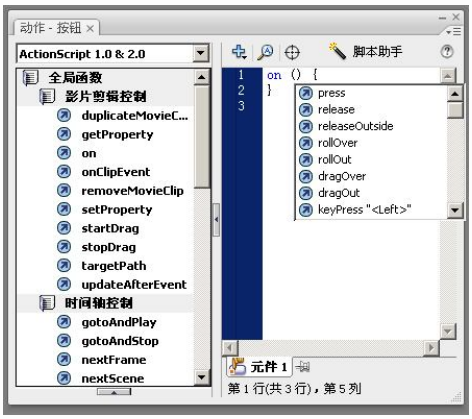

图 4-8　按钮事件句柄

on (dragOut)：单击按钮并拖动到按钮外侧时，要执行的代码或发生的事件。
on (keyPress "<Left>")：按下键盘指定键（向左方向键）时，要执行的代码或发生的事件。
影片剪辑句柄如图 4-9 所示。

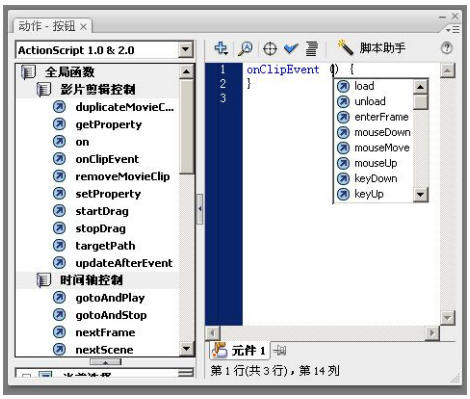

图 4-9　影片剪辑句柄

onClipEvent (load)：加载影片剪辑，执行该命令。
onClipEvent (unload)：卸载影片剪辑，执行该命令。
onClipEvent (enterFrame)：反复播放影片剪辑，执行该命令。
onClipEvent (mouseDown)：按下鼠标时，执行该命令。
onClipEvent (mouseMove)：移动鼠标时，执行该命令。
onClipEvent (mouseUp)：释放鼠标时，执行该命令。
onClipEvent (keyDown)：按下键盘时，执行该命令。
onClipEvent (keyUp)：释放键盘时，执行该命令。
onClipEvent (data)：CGI、ASP、XML 等数据传递结束时，执行该命令。

4.2.2　对象、属性和方法

对象在现实生活中很常见。比如，人、计算机、汽车都是对象，对象是有某些特性的事物的抽象。每个对象有自己的属性和行为，比如人有头、躯干、四肢，这是属性；人有行走、

思考等行为，这就是方法。

Flash 实例拥有原对象的所有属性和方法。

影片剪辑实例若要在程序中被引用，先要设定影片剪辑实例的“实例名称”，然后通过实例名来应用动作。影片剪辑实例名一般使用便于记忆的英文表示，如图 4-10 所示。

图 4-10　影片剪辑实例名称

提示

用英文、数字及空格来表示，不使用中文；不能用数字开头；不能用“+”、“-”等运算符；不能用动作语句来命名。

属性是指对象拥有的各种特性。例如影片剪辑的不透明度、颜色、大小等。属性大部分都可以在程序中进行设定。

例如：box._alpha=30; / /影片剪辑“box”的不透明度属性设置为“30%”。

方法是指赋予对象的各种行动。简单说，对象的函数就叫方法。

例如：box.gotoAndPlay(5);　/ /将“box”这个实例跳转到第 5 帧并播放。

4.2.3　变量

变量是程序设计中重要的组成部分，用来对所需的数据资料进行暂时存储。只有设置变量名称与内容，就可以产生一个变量。变量可用于记录和保存用户的操作信息、输入的资料、记录动画播放的剩余时间、判断条件是否成立等。

在游戏中，我们常常会发现，某些值和参数是随着游戏的进行不断变化的。比如人物的进攻力、防御力、武器威力、生命值等。这些值都是随时变化的，因此存储这些可变数值的量就是变量。

变量在使用之前，我们先要定义它。例如：

```
Power=50;
```

这里定义了一个名字叫“Power”的变量，意图是用来存储人物的“进攻力”的量，并且赋予这个“进攻力”的值为 50。随着程序的进行，这个值可以被重新赋予新的值。

在表达式中，“=”是赋值符号，随着“=”右边的值的变化，左边变量的值也变化。

变量的命名规则

ActionScript 中变量名必须是标识符，那么就要遵循标识符的格式和规则：

- 变量名第一个字符必须为字母、下划线、美元符号，其后的字符可以是数字、字母、下划线或美元符号。
- 变量名不能是保留关键字。例如：不能是 if、else、and 等。
- 变量名没有大小写之分。例如：BOX 和 box 被解释为同一个变量。
- 变量名不能是 ActionScript 语言中的命令名称。
- 在它的作业范围内必须是唯一的。

提示 变量的作用范围是指脚本中能够识别和引用指定变量的区域。

变量名的错误定义：

```
? Power=50;
55="good";
```

变量在使用前，一般先定义：

```
var Speed; //汽车行驶速度
var score; //玩家得分
var hiScore; //最高分
var time;  //玩家所用时间
```

提示 “//”为注释分隔符号，用于在脚本中为命令语句添加注释说明。出现在注释分隔符“//”和行结束符之间的字符，都被程序解释为注释，不会被作为脚本语句来分析执行。注释在程序设计中经常使用，方便快速理解脚本的意图。

4.2.4 数据类型

数据的种类称为数据类型。ActionScript 主要数据类型如下：

数据类型	说明
字符串（String）	字符串数据，包括字母、数字和标点符号。需要用""包围的数据。
数字（Number）	数字数据，可以表示整数、无符号整数和浮点数。可以进行算术运算（+、－、×、÷）以及比较运算（＜、＞、＝）的数据
布尔（Boolean）	布尔数据，包括 true 和 false 两个值。其他任何值都是无效的
对象（Object）	对象数据，用作所有类定义的基类。定义了属性（Property）和方法（Method）的数据
影片剪辑（MovieClip）	影片剪辑数据，允许使用 MovieClip 类的方法控制影片剪辑元件。拥有实例名的数据
未指定（null）	无值的数据

4.2.5 运算符

运算符是指定如何组合、比较或修改表达式值的字符。包括按位运算符、比较运算符、赋值、逻辑运算符、其他运算符和算术运算符。

1. 比较运算符

比较运算符用于进行变量与数值间、变量与变量间大小比较，如图 4-11 所示。

!= ：不等于运算符；

!==：不全等于运算符；

<：小于运算符；

<=：小于或等于运算符；

==：等于运算符；

===：全等于运算符；

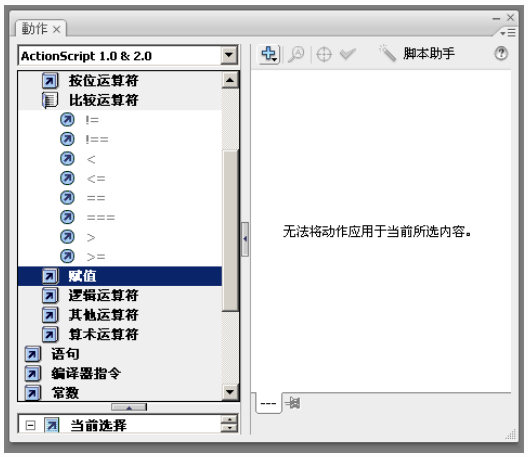

图 4-11　比较运算符

>：大于运算符；

>=：大于或等于运算符。

例如：>=（大于等于）的使用。

用于测试符号左边的表达式是否大于或等于符号右边的表达式。如果是，则结果为 true。

格式：Expr1>=Expr2

Expr1、Expr2 可以表示为数字、字符串、布尔值、变量、对象、数组或函数。

【实例 4.1】大于等于的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-12 所示。

```
var Power=500; //定义变量，升级需要的最小进攻力
var Play_power=490; //定义变量，玩家现有的进攻力
if(Play_power >= Power) {          //比较 Power 与 Play_power 的值
    trace("升级") ;                 //输出窗口中显示“升级 ”
} else {
    trace("不升级");                //输出窗口中显示“不升级 ”
}
```

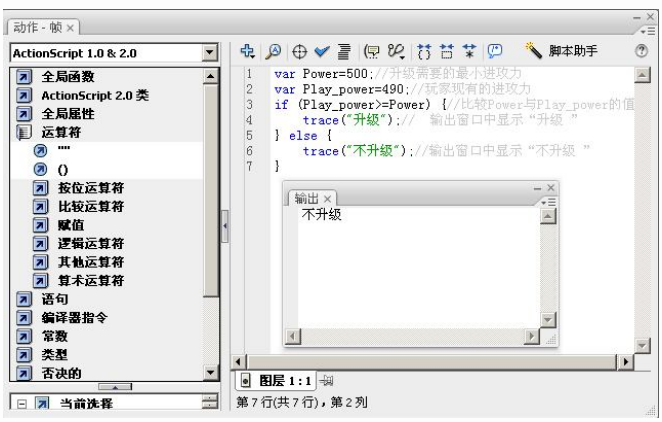

图 4-12　代码

提示

trace（表达式）的功能：将表达式的结果从“输出”面板中显示出来，但只能在 Flash 的开发环境中使用，不能在 SWF 播放器中显示结果。
本例中使用了 if 语句，将在后面章节中进行分析和讲解。
程序中使用的标点符号都必须是英文格式的。

2. 赋值运算符

赋值运算符是指执行变量赋值的运算符，如图 4-13 所示。

图 4-13　赋值运算符

赋值运算符包括“-=”、“%=”、“&=”、“*=”、“|=”、“/=”、“^=”、“+=”、“<<=”、“=”、“>>=”、“>>>=”等。最常用的是以下几种：

“=”：赋值运算

“-=”：减法赋值运算

“+=”：加法赋值运算

【实例 4.2】赋值运算

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-14 所示。

```
var Power=500; //将 500 赋值给变量 Power。
Power+=5; //一回合胜利，Power 值递加 5。
trace(Power); //输出窗口中显示 Power 的值。
```

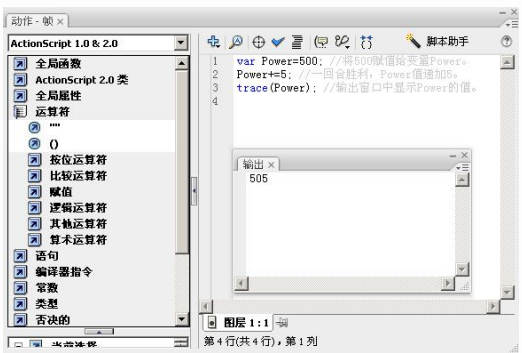

图 4-14　代码

3. 逻辑运算符

逻辑运算符可以对数字、变量等进行比较，然后得出他们的交集或并集作为输出结果，如图 4-15 所示。

逻辑运算符包括“&&”、“||”、“!”等。

“&&”：它是逻辑与运算符，对一个或两个表达式的值执行布尔运算。计算运算符左右两边的表达式，如果两边的结果都为 true，则最终结果为 true，否则最终结果为 false。即交集，如图 4-16 所示。

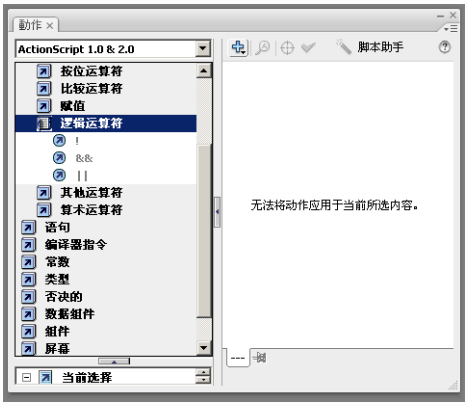

图 4-15　逻辑运算符

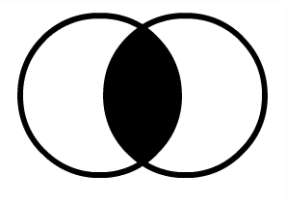

图 4-16　逻辑与

格式：Expr1　&&　Expr2

假设在游戏中要升级的条件有两个：一是角色的进攻力达到 500 或以上，二是拥有的钱币达到 1000 或以上。那么我们可以应用逻辑与运算控制是否升级。

【实例 4.3】逻辑运算

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-17 所示。

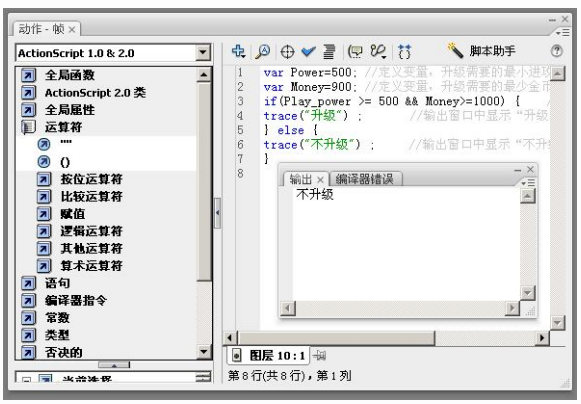

图 4-17

```
var Power=500; //定义变量，升级需要的最小进攻力
var Money=900; //定义变量，升级需要的最少金币
if(Play_power >= 500 && Money>=1000) {      //判断是否满足升级的两个条件
```

```
        trace("升级") ;          //输出窗口中显示“升级 ”
    } else {
        trace("不升级") ;        //输出窗口中显示“不升级 ”
    }
```

“||”：它是逻辑或运算符，计算符号左右两边的表达式，如果有其中一边的结果为 true，则最终结果为 ture，除非两者都为 false，最终结果才是 false，即并集，如图 4-18 所示。

格式：Expr1 || Expr2

假设在游戏中要升级的条件是满足以下条件之一：不管是角色的进攻力达到 500，还是拥有的钱币达到 1000。这种情况就可以应用逻辑或进行判断。

“!”：它是逻辑非运算符，表示取变量或表达式的布尔值的相反值，即补集，如图 4-19 所示。

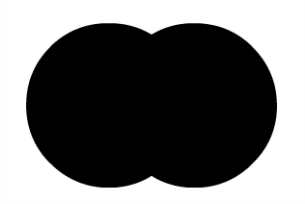

图 4-18 逻辑或

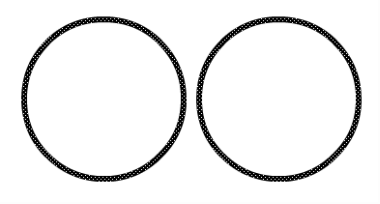

图 4-19 逻辑非

格式：! Expr1

Expr1 的值如果是 true，那么 ! Expr1 的值则为 false；Expr1 的值如果是 false，那么 ! Expr1 的值则为 true。

4. 算术运算符

算术运算符是指用于对数值、变量进行计算的各种运算符号，如“+”、“-”、“*”、“/”、“%”等，如图 4-20 所示。

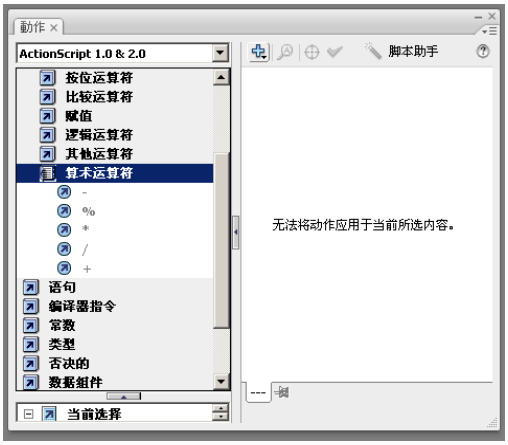

图 4-20 算术运算符

例如：+（加号）的使用

用于计算符号左边的表达式和符号右边的表达式的和。

格式：Expr1+Expr2

【实例 4.4】加号的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-21 所示。

```
var Money1=500;       //定义变量，玩家出售装备获得的金币
var Money2=490;       //定义变量，玩家消灭对手获得的金币
var Money; //定义变量，存储玩家总共拥有的金币
Money= Money1+ Money2;      //将变量 Money1 和 Money2 的值相加，赋给变量 Money
trace("您的总计金币数为："+Money) ;       //输出窗口中显示 Money 的值
```

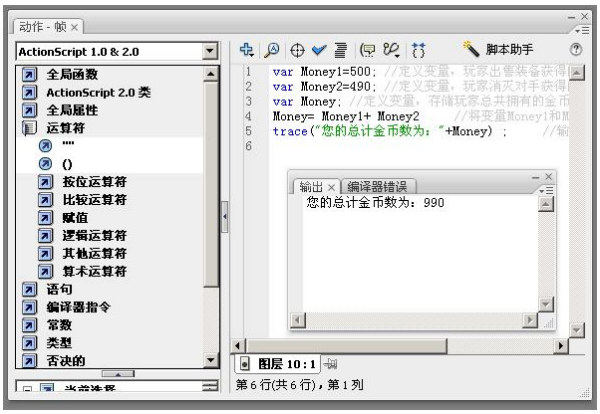

图 4-21　代码

提示

trace("您的总计金币数为："+Money)中，字符串“您的总计金币数为：”的输出需要使用引号，Money 数值的输出则不需要。同时输出字符串和数值，中间使用“+”。

5. 其他运算符

其他运算符中包括“--”递减变量、“?:”条件运算、“++”递加变量、“instanceof”返回对象与类之间继承的布尔值、“typeof”返回指定表达式的类型字符串、“void”返回 undefined 值等运算符，如图 4-22 所示。

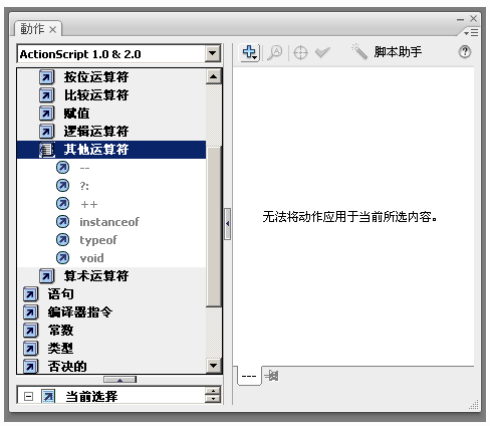

图 4-22　其他运算符

例如：++（递加）的使用

用于将表达式加 1 的预先递加或滞后递加的一元运算符。

格式：++Expr

Expr++

预先递加格式：++ Expr，表示将 Expr 加 1，然后返回结果；

滞后递加格式：Expr++，表示将 Expr 加 1，返回 Expr 的初始值。

【实例 4.5】递加的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-23 所示。

```
var   a=500;          //定义变量 a
var   b=a++;          //定义变量 b，a 滞后递加，赋值于 b
var   c=500;          //定义变量 c
var   d= ++ c;        //定义变量 d，c 预先递加，赋值于 d
trace("b="+b) ;       //输出窗口中显示 b 的值
trace("d="+d) ;       //输出窗口中显示 d 的值
```

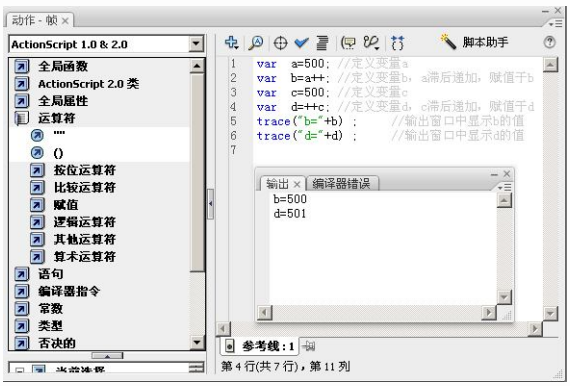

图 4-23　代码

4.3　程序控制语句

与其他程序语言类似，ActionScript 的分支和循环程序控制语句，可以控制 ActionScript 的流程。例如条件语句 if、else、else if、switch 等用来判断游戏中是否可以升级；循环语句 for、while、do while 等用来控制多次的循环操作。

4.3.1　条件语句

1. if 语句

格式：

```
if (condition) {
    statement(s);
}
……
```

if 语句通常用来判断所给的条件是否满足，如果判断结果（condition）为真（true），则 Flash 将运行大括号内的语句（statement(s)），再继续执行后面的内容。如果判断结果（condition）为假（false），则 Flash 将跳过大括号内的语句，执行大括号外的语句。

【实例 4.6】if 语句的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-24 所示。

```
var task=true;          //定义变量 task，表示游戏中任务是否完成的布尔变量
var money=0;            //定义变量 money，表示游戏中金币的值，赋初始值为 0
var Power =10;          //定义变量 Power，表示游戏中进攻力的值，赋初始值为 10
if(task==true) {        //判断任务是否完成
    money+=100;         //金币增加 100
    Power+=100;         //进攻力增加 100
}
trace("金币值为："+money);      //输出窗口中显示字符串和 money 的值
trace("进攻力为："+ Power);     //输出窗口中显示字符串和 Power 的值
```

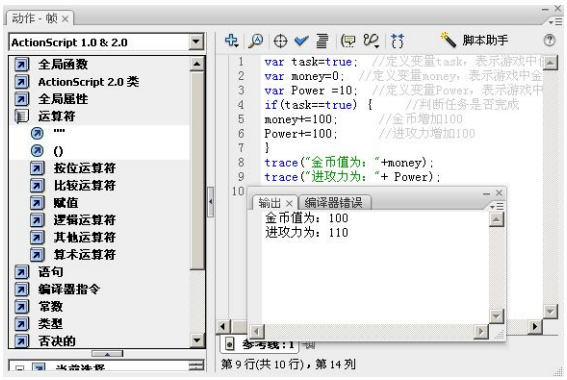

图 4-24　代码

2. if…else 语句

格式：

```
if (condition) {
    statement(s);
} else{
    statement(k);
}
……
```

如果判断结果（condition）为真（true），则 Flash 将运行语句（statement(s)），不执行语句（statement(k)）。如果判断结果（condition）为假（false），则 Flash 直接运行语句（statement(k)），不执行语句（statement(s)）。

if 与 if…else 看起来比较相似，但在控制程序流程上是有区别的。比如实例 4.6 中，我们设置游戏如果完成某任务，则增加玩家的金币数量和进攻力；如果没有完成任务，则什么也不增加。实际在游戏设置时，即使没有完成任务，进攻力也可以少量的增加，这样可以增加玩家游戏的信心。因此我们可以按实例 4.7 来设计。

【实例 4.7】if…else 语句的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-25 所示。

```
var task=false;    //定义变量 task，表示游戏中任务是否完成的布尔变量
var money=0;       //定义变量 money，表示游戏中金币的值，赋初始值为 0
var Power =10;     //定义变量 Power，表示游戏中进攻力的值，赋初始值为 10
if(task==true) {          //判断任务是否完成
    money+=100;           //金币增加 100
    Power+=100;           //进攻力增加 100
} else{
    Power+=30;            //进攻力增加 30
}
trace("金币值为："+money);       //输出窗口中显示字符串和 money 的值
trace("进攻力为："+ Power);      //输出窗口中显示字符串和 Power 的值
```

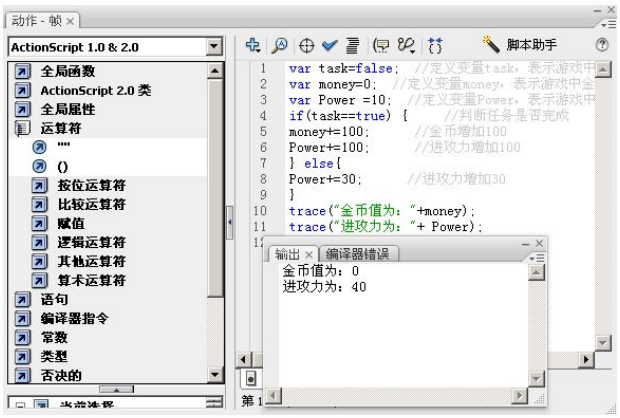

图 4-25　代码

提示

如果任务完成（task=true），则结果显示与实例 4.6 相同，如图 4-26 所示。

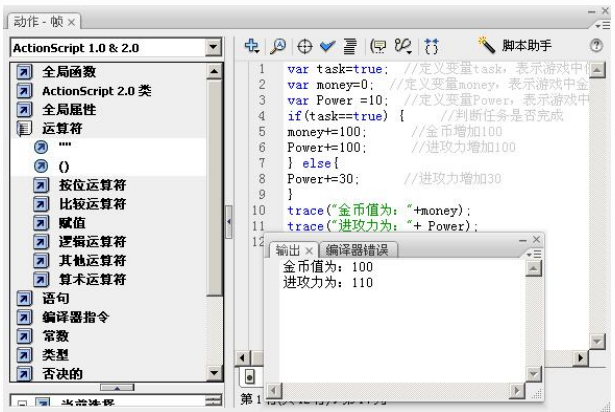

图 4-26　代码

3. if…else if…语句

格式：

```
if (condition1) {
    statement(a);
} else if(condition2){
    statement(b);
}else if(condition3){
    statement(c);
}
……
```

如果判断结果（condition1）为假（false），则 Flash 将继续判断（condition2），若仍为假，则继续判断下一个 else if 的条件式，直到某一个条件式为真（true），执行大括号中的相应 statement 后，跳过其他的 else if 语句。

比如我们在登录游戏时，经常会碰到为防止未成年人沉迷游戏的程序：

```
if(age<18){      //输入年龄是否小于 18
    fscommand("quit");  //退出 flash
}else if(age>100){        //输入年龄是否大于 100
    trace("请输入真实年龄！");  // 输出窗口中显示字符：请输入真实年龄！
} else{
    gotoAndPlay("Opening",1);    //跳到 Opening 场景的第 1 帧，并且播放
}
```

4. switch 语句

格式：

```
switch(expr){
case value:
    do something
[default:]
    do something
}
```

expr 表示任意的表达式；case 是关键字，value 是 expr 的值。

我们常常在需要检查是否满足若干条件中的一个条件时，使用 switch 语句。switch 语句测试一个条件，并在条件满足时执行语句。

例如在游戏中经常遇到完成某个任务或达到某个级别后，可以根据不同的完成情况玩家选择奖励物品。

【实例 4.8】switch 语句的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-27 所示。

```
var your_choose=1;
switch(your_choose){  //参数 your_choose，表示玩家选择项的变量
case 1:                    //your_choose 的值为 1 时
    trace("您得到了霸王盔");    //输出窗口中显示字符: 您得到了霸王盔
    break;
case 2:                    //your_choose 的值为 2 时
```

```
        trace("您得到了盘古盔");    //输出窗口中显示字符: 您得到了盘古盔
        break;
case 3:                    //your_choose 的值为 3 时
        race("您得到了恶灵盔");    //输出窗口中显示字符: 您得到了恶灵盔
        break;
default:                   //没有赋值时
        trace("您不能得到奖赏");    输出窗口中显示字符: 您不能得到奖赏
}
```

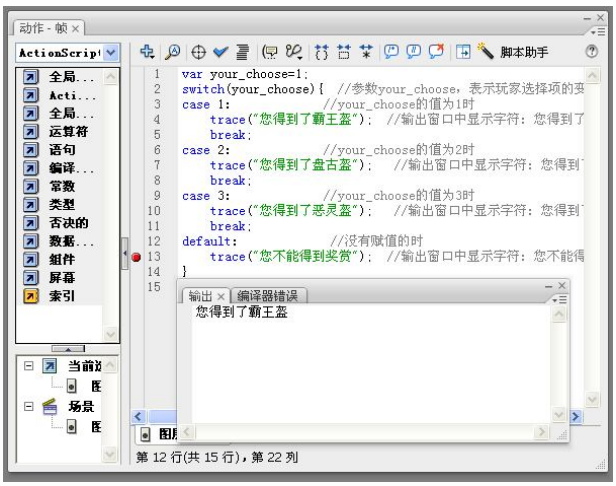

图 4-27　代码

4.3.2　循环语句

1．for 语句

格式：

```
for (init; condition; next) {
    statement(s);
}
……
```

for 语句是一个循环结构，它首先计算 init 表达式一次，只要 condition 的计算结果为真（true），则按照以下顺序开始循环序列，执行 statement，然后计算 next 表达式。

init：通常为赋值表达式。

condition：计算结果为 true 或 false 的表达式。在每次循环迭代前计算该条件，当条件的计算结果为 false 时退出循环。

next：一个在每次循环迭代后要计算的表达式；通常为使用递增（++）或递减（--）运算符的赋值表达式。

例如游戏中，要求在与 NPC 战斗时，游戏限定最多 8 个回合就要结束战斗。

【实例 4.9】for 语句的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-28 所示。

```
var i;          //定义变量 i，在循环时计数
var npc_hp=100;  //定义变量 npc_hp，表示游戏中 npc 的生命值
var it_hp=100;  //定义变量 it_hp，表示游戏中玩家的生命值
var npc_hit=5;  //定义变量 npc_hit，表示游戏中 npc 每击中游戏玩家后，玩家需减去的生命值
var it_hit=6;  //定义变量 it_hit，表示游戏中玩家每击中 npc 后，npc 需减去的生命值
for (i=0;i<12;i++) {      //for 循环，用 i 计数，i 到 10 后停止循环
    npc_hp-=it_hit;       //npc 的生命值等于原有值减去玩家对其伤害值（it_hit）
it_hp-=npc_hit;           //it 的生命值等于原有值减去 npc 对其伤害值（npc_hit）
trace("回合："+i);        //输出窗口中显示字符串和 i 的值
trace("npc 的生命值"+ npc_hp);  //输出窗口中显示字符串和 npc_hp 的值
trace("it 的生命值"+ it_hp);  //输出窗口中显示字符串和 it_hp 的值
}
```

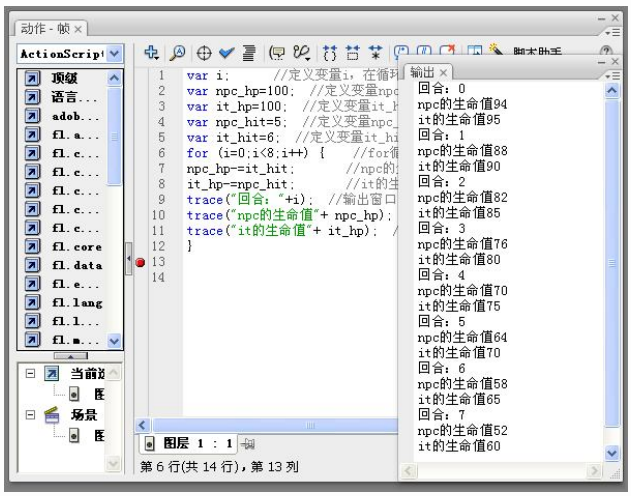

图 4-28　代码

提示

本例假设玩家和 pnc 对抗时，每个回合都是各击中对方一次。

2. while 语句

格式：

```
while (condition) {
    statement(s);
}
……
```

while 语句是一个循环结构。首先计算 condition，如果结果为 true，则执行后面的语句，如果结果为 false，则跳过大括号中的语句，继续执行之后的语句。

condition：每次执行 while 动作时都有重新计算的表达式。如果该语句的计算结果为真（true），则运行 statement。

statement(s)：条件语句结果为真时要执行的语句。

例如在游戏中指定玩家对付 npc 时，每进攻一次，获得经验值 10，如果经验值达到 100，则可以升级。

【实例 4.10】while 语句的使用

新建一个 Flash 文档，选中“图层 1”的第 1 帧，按 F9 键打开“动作”面板并输入如下代码，按组合键 Ctrl+Enter 预览效果，如图 4-29 所示。

```
var i=0;              //定义变量 i，记录进攻的次数
var it_exp=10;    //定义变量 it_exp，表示游戏中玩家的初始经验值
var it_hit=10;     //定义变量 it_hit，表示游戏中玩家每击中 npc 后，玩家获得的经验值
while (it_exp<100) {   //如果 it_exp<100 循环继续，直到大于或等于 100 退出循环
    it_exp+= it_hit;     //玩家的经验值等于原有值加上进攻一次获得的值
    i+=1;                  //进攻 1 次计数
    trace("进攻次数："+i);          //输出窗口中显示字符串和 i 的值
    trace("it 的经验"+ it_exp);      //输出窗口中显示字符串和 it_exp 的值
}
```

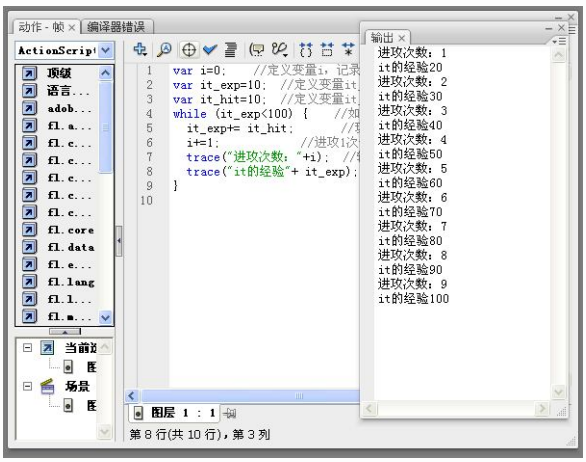

图 4-29　代码

4.4　常用动作

4.4.1　stop 和 play

stop：主要功能是停止影片的播放。如果动画中没有加入 stop 动作就会一直循环播放。若要取消动画开始的自动播放，或在某个指定的帧停止播放，可以使用 stop 动作。

格式：

```
stop();
```

如果要让某个影片剪辑停止播放，可以直接在它前面加上影片剪辑的实例名。

```
box_mc.stop();
```

play：是一个播放命令，主要的功能是控制时间轴的动画播放。

格式：

```
play();
```

如果要让已经停止播放的影片剪辑开始播放，可以直接在它前面加上影片剪辑的实例名。

```
box_mc.play();
```

【实例 4.11】播放和停止

步骤 1：新建 Flash 文档，并设置尺寸为 550px×400px，帧频为 12fps，如图 4-30 所示。

图 4-30　设置文档属性

步骤 2：修改图层名为“背景”，并在该图层的第 1 帧中绘制如图 4-31 所示的白云和天空，并分别将其转换成图形元件。

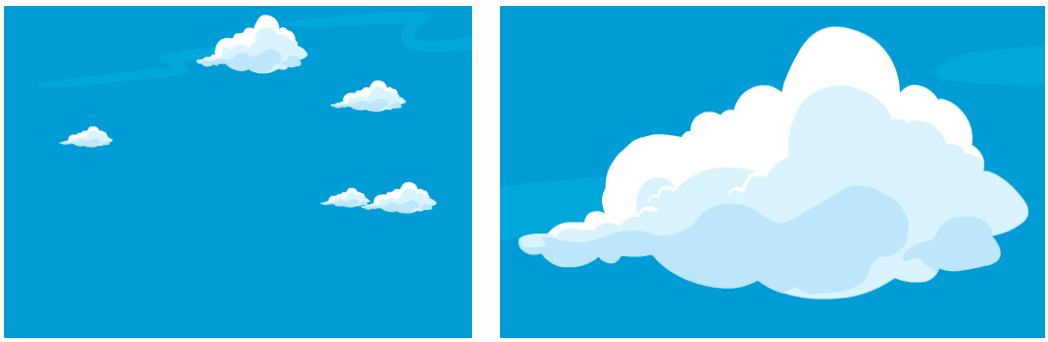

图 4-31　白云绘制

步骤 3：单击“插入图层”按钮，新建一个名为“气球”的图层，在第 1 帧分别绘制不同颜色的气球和蝴蝶结，并单独转换为元件。最后组合成气球总体，按 F8 键将其转换为影片剪辑元件，如图 4-32 所示。

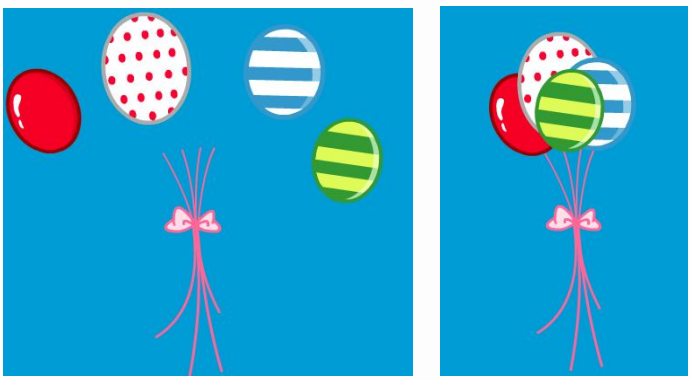

图 4-32　气球绘制

步骤 4：单击“插入图层”按钮，新建一个名为“按钮”的图层，在第 1 帧中绘制如图 4-33 所示的图形，选中后按 F8 键转换为名为“button-play”的按钮元件。

步骤 5：双击按钮元件，进入元件编辑模式，如图 4-34 所示。

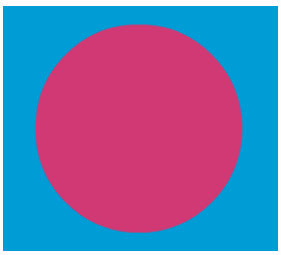

图 4-33　按钮绘制

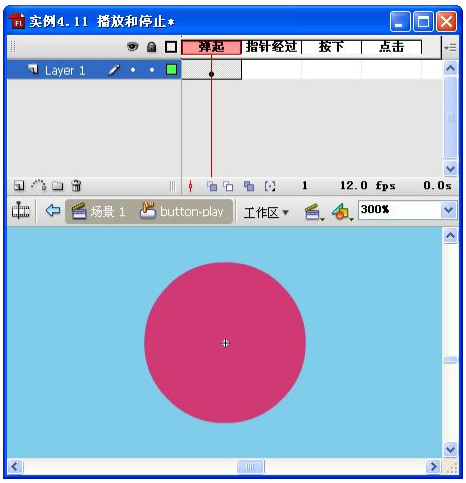

图 4-34　元件编辑

步骤 6：在第 1 帧中输入“PLAY”，如图 4-35 所示。

步骤 7：分别在“指针经过”、“按下”和“点击”帧插入关键帧，如图 4-36 所示。

图 4-35　文字输入

图 4-36　按钮设置

步骤 8：选中“指针经过”帧，将“PLAY”的颜色进行更改，并将其放大到 110%，如图 4-37 所示。

提示

步骤 8 设置了按钮动画，动画播放时，鼠标经过按钮时“PLAY”会变色并放大。

步骤 9：单击“场景 1”按钮返回到主场景。按同样的方法制作名为“button-stop”的按钮元件，如图 4-38 所示。

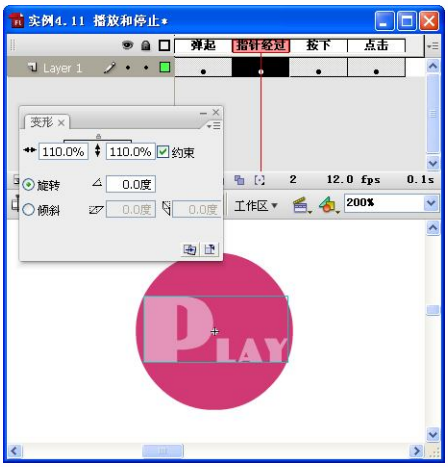

图 4-37　变形

图 4-38　按钮制作

步骤 10：分别选中“背景”层和“按钮”的第 50 帧，按 F5 键插入帧。选中“气球”层的第 50 帧，按 F6 键插入关键帧，如图 4-39 所示。

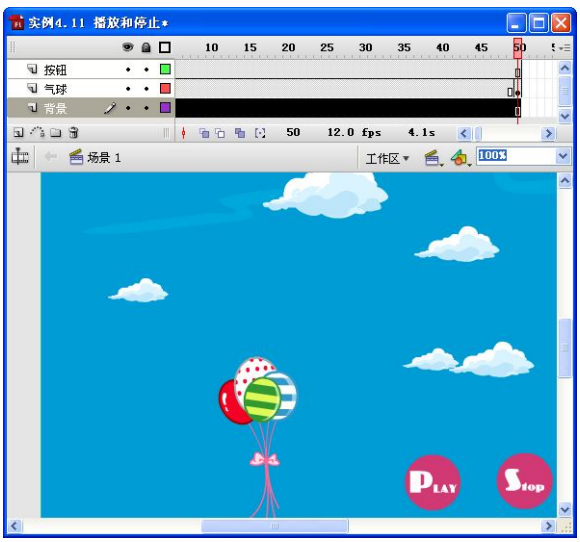

图 3-39　插入帧

步骤 11：选中“气球”层的第 50 帧，将气球影片剪辑元件拖放到顶端。在两个关键帧之间创建补间动画，形成气球上升的动画效果，如图 4-40 所示。

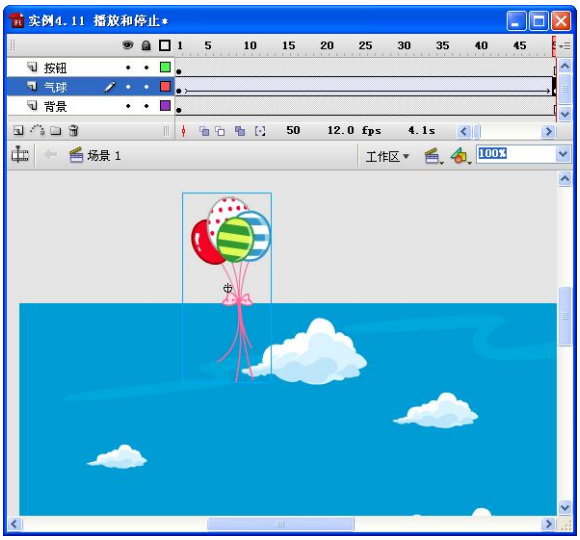

图 4-40　移动位置

步骤 12：单击“插入图层”按钮，新建一个名为“动作”的图层，在第 50 帧处插入空白关键帧。按 F9 键弹出“动作”面板，如图 4-41 所示。

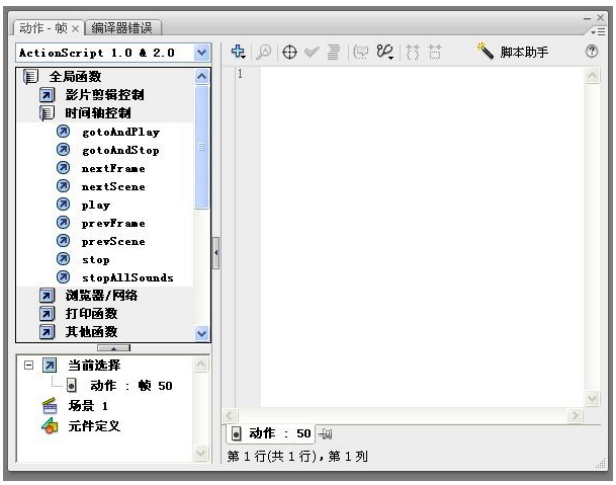

图 4-41　“动作”面板

步骤 13：单击左侧动作工具箱中的“全局函数”→“时间轴控制”，双击“stop”命令，在脚本窗格中显示“stop();”命令，如图 4-42 所示。

提示

步骤 13 给帧 50 添加了命令，动画播放到第 50 帧处将停止。通过时间轴动作脚本停止动画的播放。

步骤 14：选中“button-play”按钮，按 F9 键弹出“动作”面板。单击左侧动作工具箱中的“全局函数”→“影片剪辑控制”，双击“on”命令，在脚本窗格中显示按钮事件句柄 on，如图 4-43 所示。

步骤 15：为了设置在单击按钮时触发动作，双击“press”选项，如图 4-44 所示。

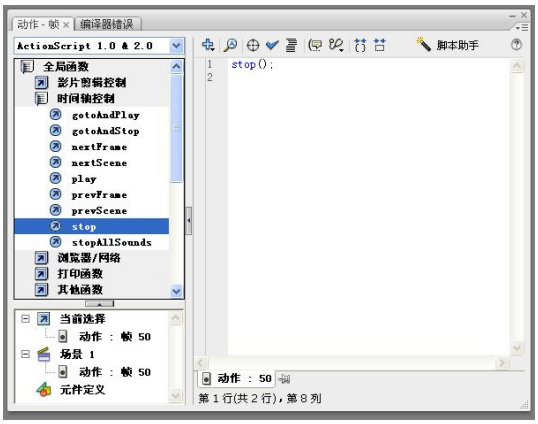

图 4-42　输入代码

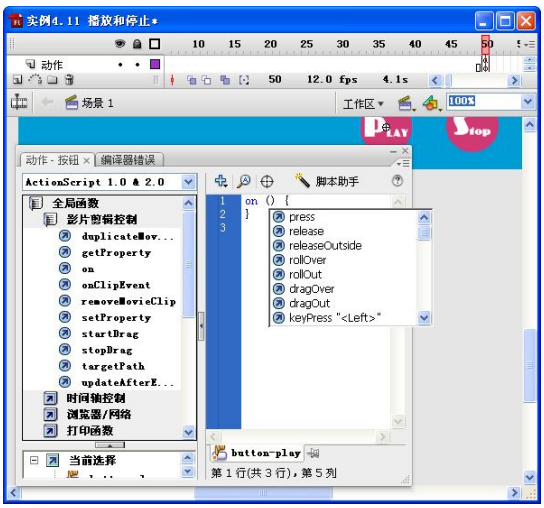

图 4-43　选择 on 命令

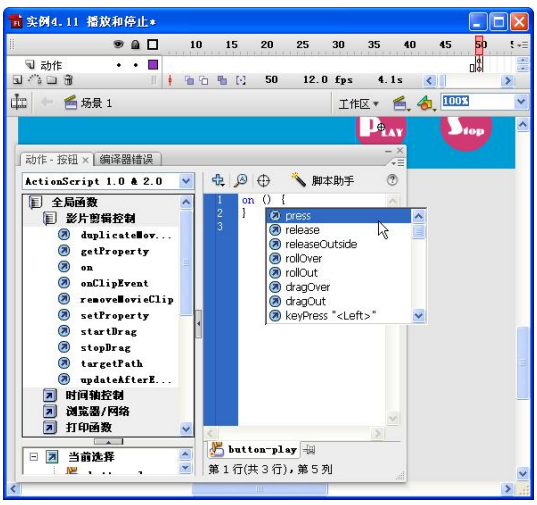

图 4-44　选择 press 命令

步骤 16：在脚本窗格中显示 on(press)，然后在{}之间按下 Enter 键，增加一行。光标移动到中间行，单击左侧动作工具箱中的“全局函数”→“时间轴控制”，双击“play”命令，动作代码如图 4-45 所示。

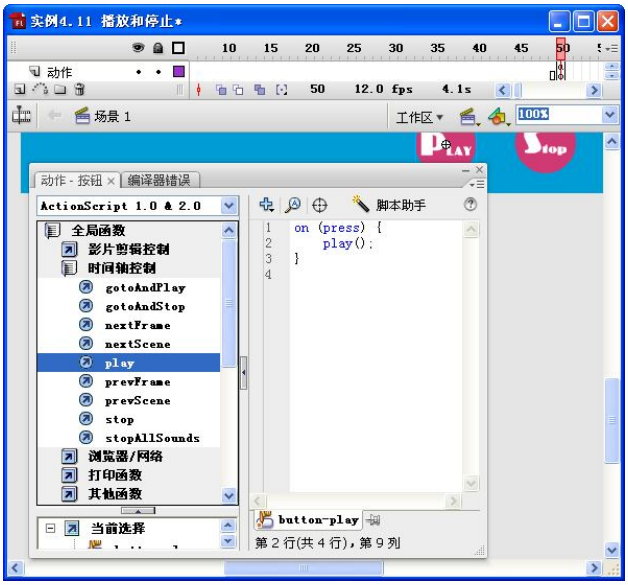

图 4-45　输入 play 命令

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

步骤 17：选中“button-stop”按钮，按同样的方法给停止按钮添加代码。不同之处应单击动作工具箱中的“全局函数”→“时间轴控制”，双击“stop”命令，代码如图 4-46 所示。

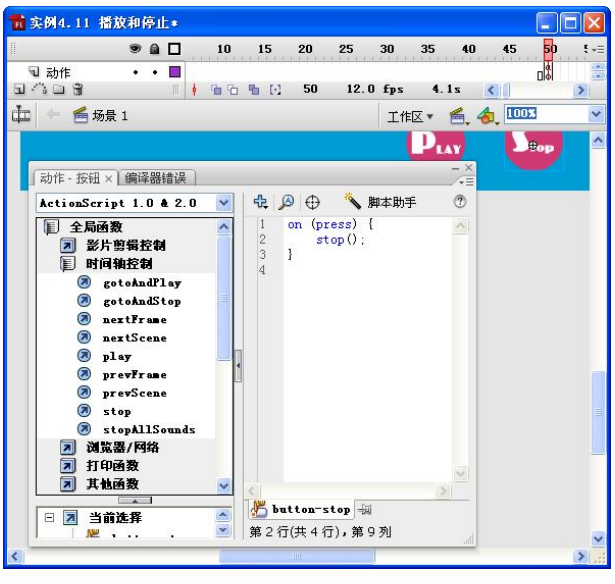

图 4-46　输入 stop 命令

提示

给按钮添加代码时，首先应选中按钮，再按 F9 键弹出“动作”面板；给帧添加代码时，首先应选中帧，再按 F9 键弹出“动作”面板。

步骤 18：按组合键 Ctrl+Enter 预览效果。气球上升的动画中，按“stop”按钮，气球停止上升；按“play”按钮，气球继续上升。到第 50 帧处动画停止，如图 4-47 所示。

图 4-47　预览效果

4.4.2　goto

goto 语句主要用于控制动画的跳转，具体可分为 gotoAndPlay 和 gotoAndStop。

gotoAndPlay：它让动画跳转到指定的帧并播放。有三种用法：跳转到指定的帧、跳转到指定的场景的帧和跳转到标签。

要控制影片剪辑，可以在影片剪辑的前面加上影片剪辑的实例名。例如：

```
box_mc.gotoAndPlay(2);  //跳转到影片剪辑 box_mc 的第 2 帧，并开始播放。
```

格式：

```
gotoAndPlay(8);                 //跳转到指定的帧（第 8 帧），并开始播放
gotoAndPlay("场景 1",8)         //跳转到指定场景（场景 1）的指定帧（第 8 帧）并开始播放
gotoAndPlay("标签 1")           //跳转到指定的标签（标签 1），并开始播放
```

gotoAndStop：它让动画跳转到指定的帧并停止。用法与 gotoAndPlay 是类似的。

要控制影片剪辑，可以在影片剪辑的前面加上影片剪辑的实例名。例如：

```
box_mc. gotoAndStop (2);  //跳转到影片剪辑 box_mc 的第 2 帧，并停止播放。
```

格式：

```
gotoAndStop(8);                 //跳转到指定的帧（第 8 帧），并停止播放
gotoAndStop("场景 1",8)         //跳转到指定场景（场景 1）的指定帧（第 8 帧）并停止播放
gotoAndStop("标签 1")           //跳转到指定的标签（标签 1）并停止播放
```

【实例 4.12】gotoAndStop

步骤 1：新建 Flash 文档，并设置尺寸为 550px×400px，帧频为 12fps，如图 4-48 所示。

步骤 2：修改图层名为“背景”，并在该图层的第 1 帧中导入素材文件夹中的“000.jpg”图片，调整图片大小，按 F8 键将其转换为名为“背景”的图形元件，设置元件 Alpha 属性为 50%，并延长帧到第 20 帧，如图 4-49 所示。

图 4-48　设置文档属性

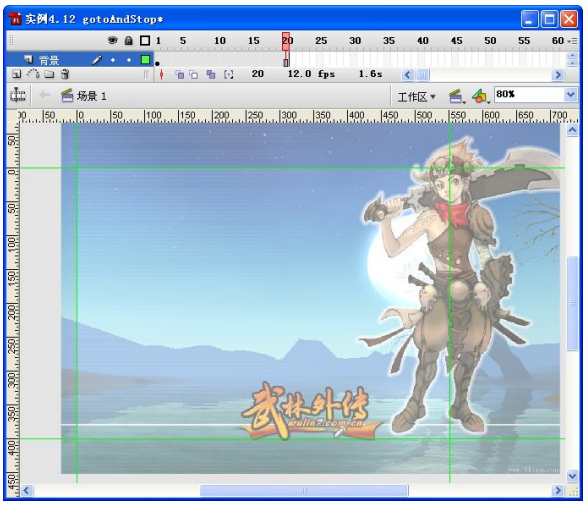

图 4-49　导入背景素材，设置背景图层

步骤 3：单击“插入图层”按钮，新建一个名为“图片”的图层。将素材文件夹中的 001.jpg、002.jpg、003.jpg、004.jpg 四张图片导入到库中，如图 4-50 所示。

图 4-50　导入素材

步骤 4：分别在“图片”图层的第 1 帧、第 5 帧、第 10 帧、第 15 帧中拖入库中的图片 001.jpg、002.jpg、003.jpg、004.jpg，如图 4-51 所示。

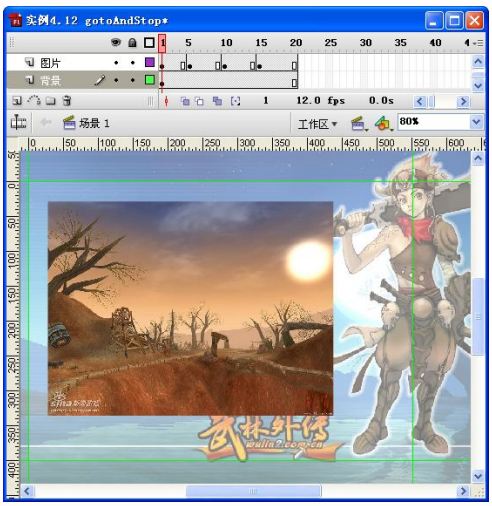

图 4-51　元件实例化

步骤 5：分别设置第 1 帧、第 5 帧、第 10 帧、第 15 帧图片的大小和位置。大小为 40%，位置为“X：28”、“Y：29”，如图 4-52 所示。

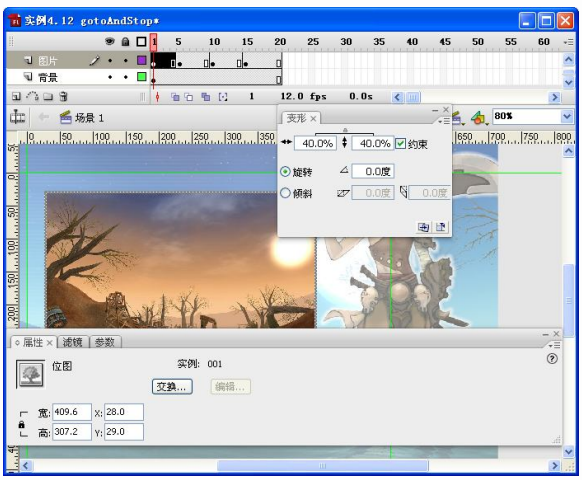

图 4-52　调整图片位置

步骤 6：单击“插入图层”按钮，新建一个名为“按钮”的图层，在第 1 帧中绘制圆角矩形，如图 4-53 所示。

图 4-53　新建按钮图层

步骤 7：选中该矩形，按 F8 键将其转换为名为“bt_1”的按钮元件。双击进入元件编辑模式，在“弹起”帧输入文字“pic 1”，如图 4-54 所示。

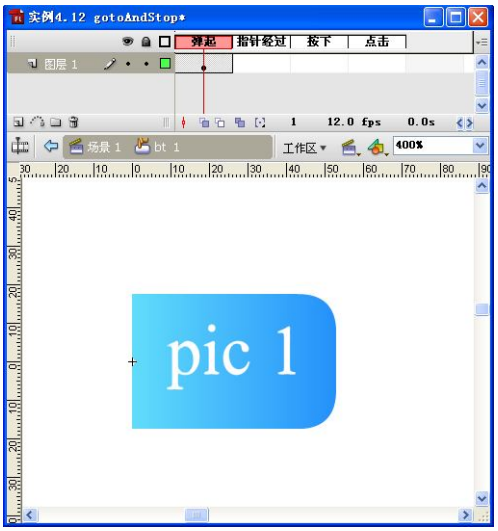

图 4-54　转换为按钮

步骤 8：单击“场景 1”按钮返回到主场景编辑模式。在库中选中“bt_1”按钮元件并右击，选择快捷菜单中的“直接复制”命令，将其更名为“bt_2”，如图 4-55 所示。

图 4-55　直接复制元件

提示

“直接复制”命令可以复制相同的另一个元件。

步骤 9：按同样的方法再复制“bt_3”和“bt_4”按钮元件。双击“bt_1”按钮元件，进入元件编辑模式。分别在“指针经过”、“按下”、“点击”帧插入关键帧，并将“指针经过”帧的对象放大 120%，如图 4-56 所示。

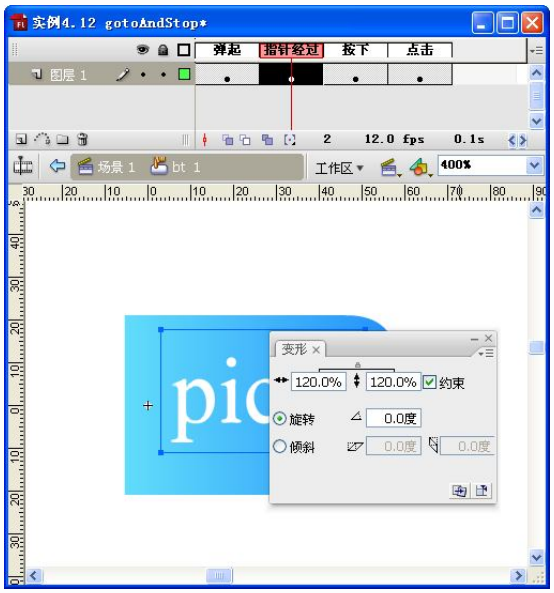

图 4-56　制作按钮动画

提示

该操作形成了按钮指针经过时放大的动画。

步骤 10：分别进入其他 3 个按钮的编辑模式，更改文本内容为“pic 2”、“pic 3、“pic 4”，并设置按钮动画，如图 4-57 所示。

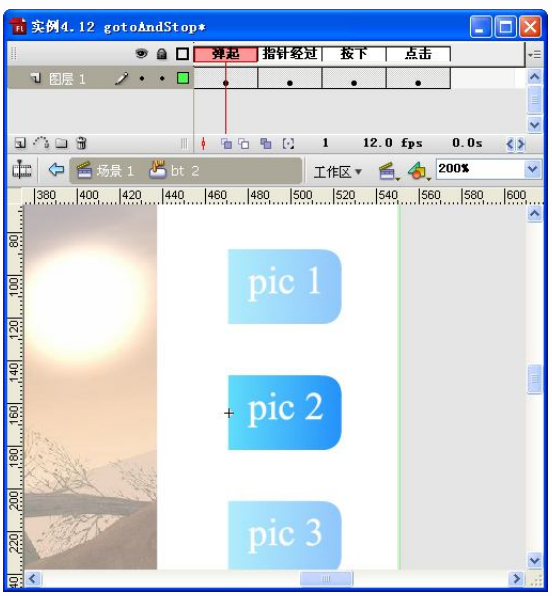

图 4-57　制作按钮动画

步骤 11：单击“场景 1”返回到主场景编辑模式。将库中的 bt_2、bt_3、bt_4 按钮拖放到“按钮”图层的第 1 帧，将 4 个按钮调整好位置和大小，并使用直线工具在左侧绘制一条

直线，延长“按钮”图层到第 20 帧，如图 4-58 所示。

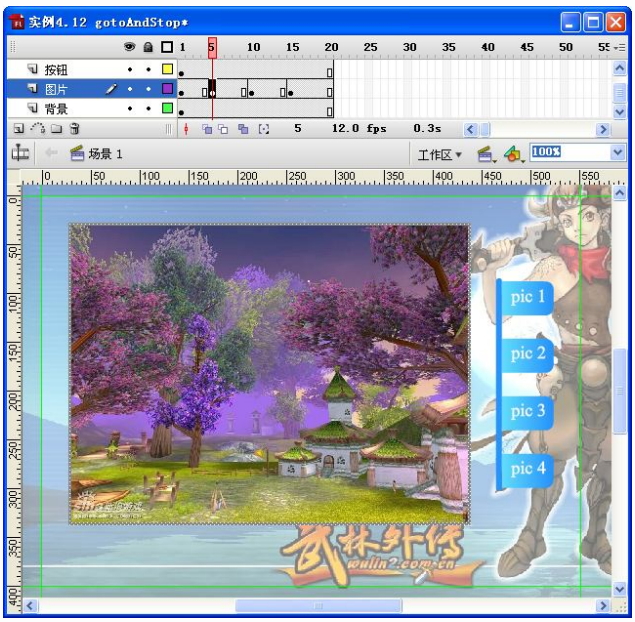

图 4-58　调整按钮位置

步骤 12：单击“插入图层”按钮，新建一个名为“as”的图层。选中第 1 帧，按 F9 键弹出“动作”面板。单击左侧动作工具箱中的“全局函数”→“时间轴控制”，双击“stop”命令，在脚本窗格中显示停止命令，如图 4-59 所示。

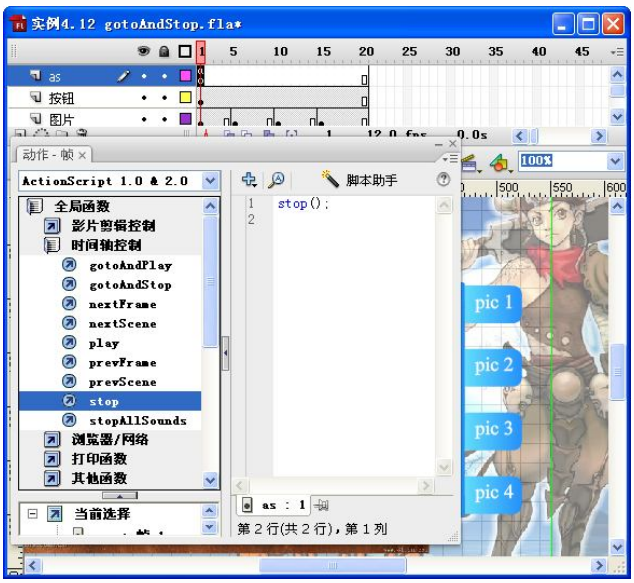

图 4-59　输入代码

步骤 13：选中“按钮”图层的第 1 帧，单击“bt_1”按钮。按 F9 键弹出“动作”面板。单击左侧动作工具箱中的“全局函数”→“影片剪辑控制”，双击“on”命令，在脚本窗格中

显示按钮事件处理函数 on，双击“release”选项后代码如图 4-60 所示。

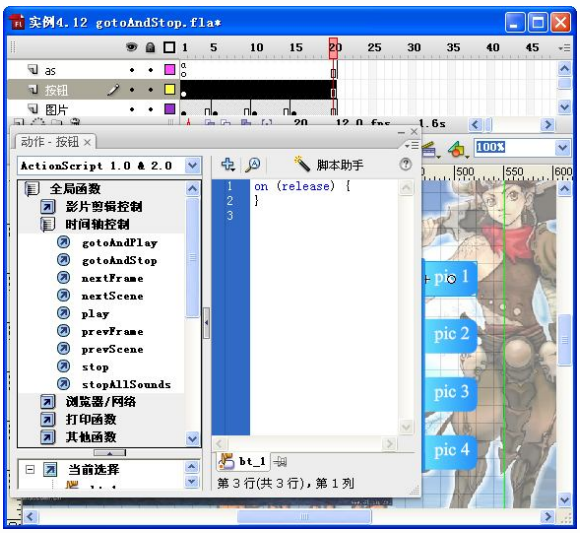

图 4-60　选择命令

步骤 14：然后在{}之间按下 Enter 键，增加一行。光标移动到中间行，单击左侧动作工具箱中的“全局函数”→“时间轴控制”，双击“gotoAndStop”命令，并在“()”中输入 1，动作代码如图 4-61 所示。

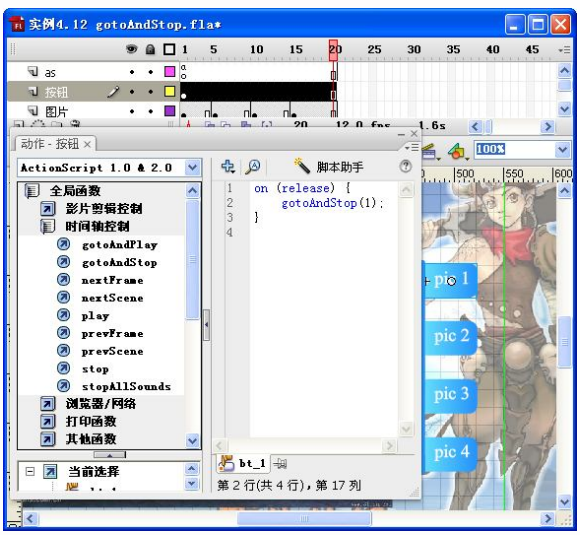

图 4-61　输入代码

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

步骤 15：按同样的方法，为按钮“bt_2”添加代码，如图 4-62 所示。

步骤 16：为按钮“bt_3”添加代码，如图 4-63 所示。

步骤 17：为按钮“bt_4”添加代码，如图 4-64 所示。

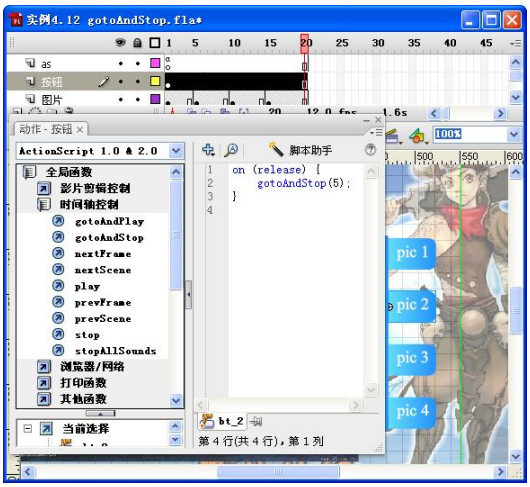

图 4-62　添加按钮代码

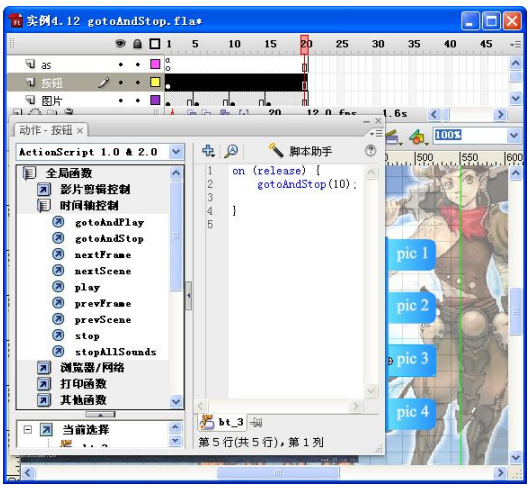

图 4-63　添加按钮代码

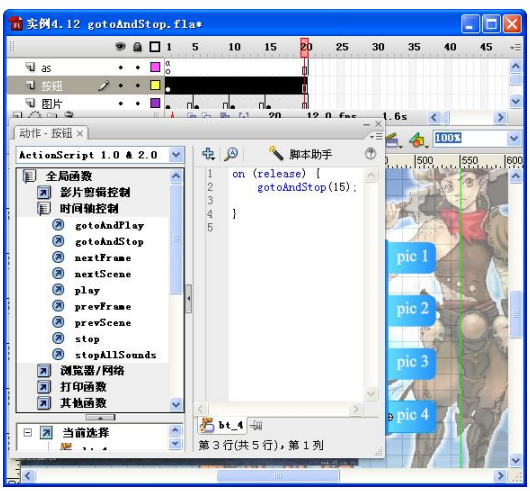

图 4-64　添加按钮代码

提示

4 个按钮的代码基本一样，只是更改括号中的帧数，分别为 1、5、10、15。

步骤 18：按组合键 Ctrl+Enter 预览效果，如图 4-65 所示。

图 4-65　预览效果

4.4.3　声音

声音可以加载到时间轴上进行同步的播放，也可以导入到库中后再通过命令将其加载到动画中。

下面通过实例来讲解声音的操作。

【实例 4.13】声音和按钮

步骤 1：新建 Flash 文档，并设置尺寸为 680px×500px，帧频为 12fps，如图 4-66 所示。

步骤 2：修改图层名称为“背景”，并在该图层的第 1 帧中绘制一个矩形，矩形高度比舞台高度稍小。矩形采用#FFFFFF 和#C2CBDE 的“线性”渐变色，如图 4-67 所示。

图 4-66　设置文档属性

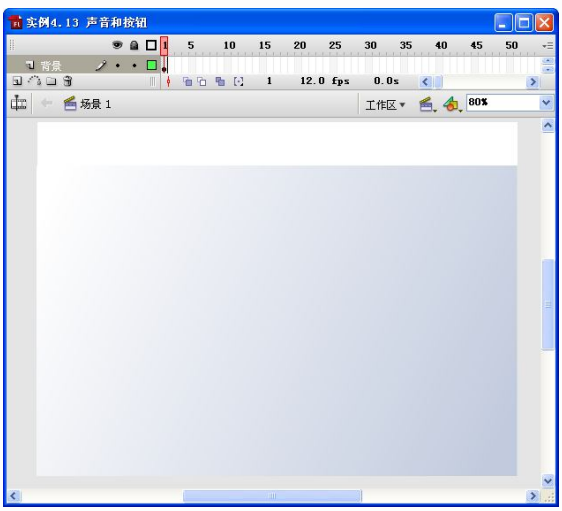

图 4-67　绘制背景

步骤 3：选用工具箱中的“文本工具”，在第 1 帧舞台的顶端输入文字“幻想三国志”和“人物介绍”，如图 4-68 所示。

图 4-68　输入文字

步骤 4：单击“插入图层”按钮，新建一个名为“人物”的图层。导入素材文件夹中的“rw1.jpg”、“rw2.jpg”、“rw3.jpg”等三张图片到库中。在“人物”图层的第 1 帧拖入库中的图片“rw1.jpg”、“rw2.jpg”、“rw3.jpg”，如图 4-69 所示。

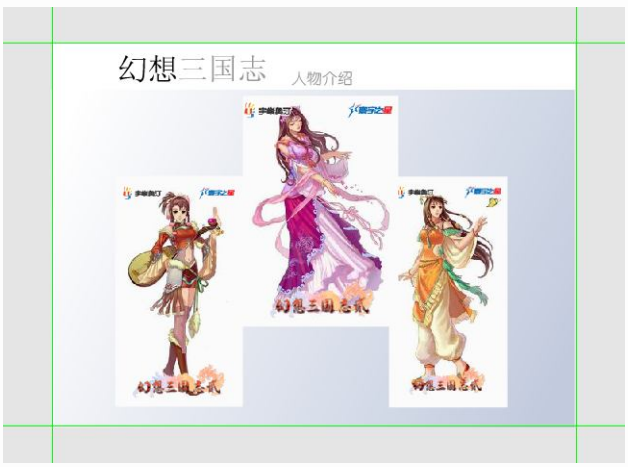

图 4-69　插入图层，导入素材并拖入到舞台

步骤 5：在第 1 帧 3 张图片空白处分别绘制 3 个渐变色的矩形，分别是#FF9900 到#CC0000 的线性渐变、#33CCFF 到#0066CC 的线性渐变、#CCFF00 到#009900 的线性渐变，如图 4-70 所示。

步骤 6：选中左上角的矩形，按 F8 键将其转换为名为“mc_1”的影片剪辑元件，选中影片剪辑后在“属性”面板中的“实例名称”中输入“sy”，如图 4-71 所示。

图 4-70　绘制矩形

图 4-71　设置影片剪辑实例名称

提示

设置影片剪辑的“实例名称”，是为了在代码中调用影片剪辑动画。这个步骤必不可少。

步骤 7：双击影片剪辑元件，进入元件编辑模式。将“图层 1”延长至第 15 帧。在“图层 1”上增加一层，命名为“名字”，并在第 1 帧输入“沈嫣”，如图 4-72 所示。

图 4-72　输入文字

步骤 8：将文字打散，按 F8 键，将其转换为名为“wz_1”的图形元件。在第 8 帧处按 F6 键插入关键帧，并将该帧中的元件 Alpha 属性设置为 0，同时将元件向右边移动。在关键帧之间创建补间动画，延长帧到第 15 帧，如图 4-73 所示。

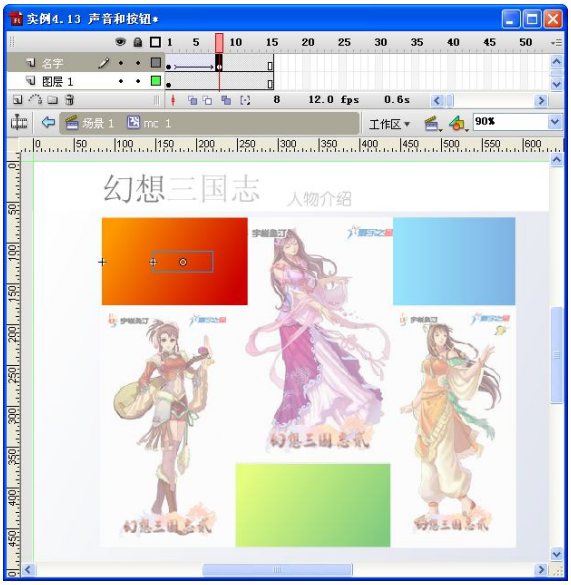

图 4-73　制作文字动画

提示

形成了文字向右并渐渐消失的动画。

步骤 9：单击“插入图层”按钮，新增一个名为“特点”的图层。在第 2 帧处插入空白关键帧，并输入“娇俏可人”，如图 4-74 所示。

图 4-74　输入文字

步骤 10：将文字打散，按 F8 键将其转换为名为“wz_2”的图形元件。在第 9 帧处按 F6 键插入关键帧，并将第 2 帧的元件 Alpha 属性设置为 0，同时将第 9 帧中的元件向左边移动。在关键帧之间创建补间动画，延长帧到第 15 帧，如图 4-75 所示。

图 4-75　制作文字动画

提示

形成了文字向左并渐渐清晰的动画。

步骤 11：单击“插入图层”按钮，新增一个名为“as”的图层。选中“as”图层的第 1 帧，按 F9 键弹出“动作”面板，单击左侧动作工具箱中的“全局函数”→“时间轴控制”，双击“stop”命令，在脚本窗格中显示“stop();”命令。在第 15 帧按 F7 键插入空白关键帧，同样给第 15 帧添加“stop();”命令，如图 4-76 所示。

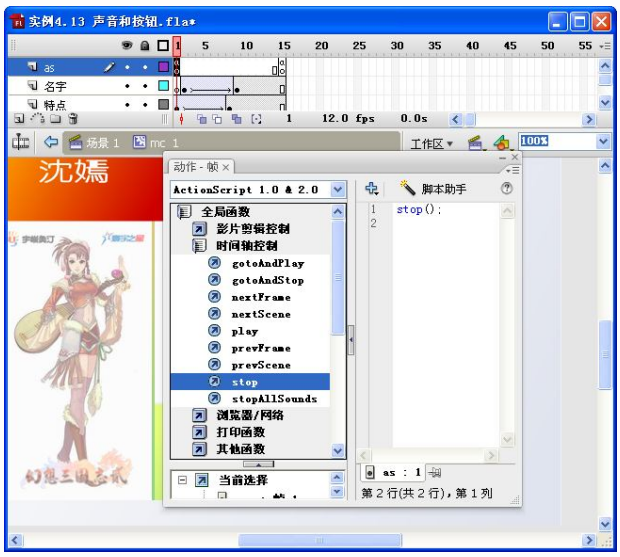

图 4-76　添加帧代码

步骤 12：按同样的方法制作“mc_2”和“mc_3”影片剪辑元件。“mc_2”设置影片剪辑

的“实例名称”为“yz”，其中的文字内容中，姓名为“瑶甄”，特点为“温柔恬美”，其他层内容与“mc_1”一致，如图 4-77 所示。

图 4-77　制作影片剪辑动画

步骤 13：“mc_3”设置影片剪辑的“实例名称”为“ht”，其中的文字内容中，姓名为“海棠”，特点为“慧黠灵巧”，其他层内容与“mc_1”一致，如图 4-78 所示。

图 4-78　制作影片剪辑动画

提示

主要动画制作完成。下面来介绍按钮和声音的制作。

步骤 14：在场景编辑模式中单击“插入图层”按钮，插入一个名为“按钮”的图层。首先在“按钮”层的第 1 帧中添加“公用库”中的播放和停止按钮，以控制动画背景音乐的播放和停止。选择“窗口”菜单“公用库”级联菜单中的“按钮”命令，弹出“库”面板，如图 4-79 所示。

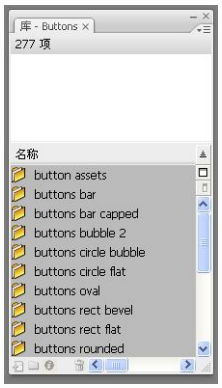

图 4-79　选择公用库命令

步骤 15：选择“classic buttons→Circle Buttons→play”和“classic buttons→Circle Buttons→stop”，拖放到“按钮”层的第 1 帧舞台上，如图 4-80 所示。

图 4-80　拖放公用库按钮

步骤 16：将素材文件夹中的“music.mp3”和“Button.wav”声音文件导入到库中，如图 4-81 所示。

图 4-81　导入声音素材

提示

.mp3 和.wav 格式的音频都可以导入到 Flash 中。

步骤 17：在库中选中声音“music.mp3”，单击鼠标右键，在弹出的菜单中选择“链接”命令打开“链接属性”对话框，勾选“为 ActionScript 导出”，在“标识符”后输入“ztq”，如图 4-82 所示。

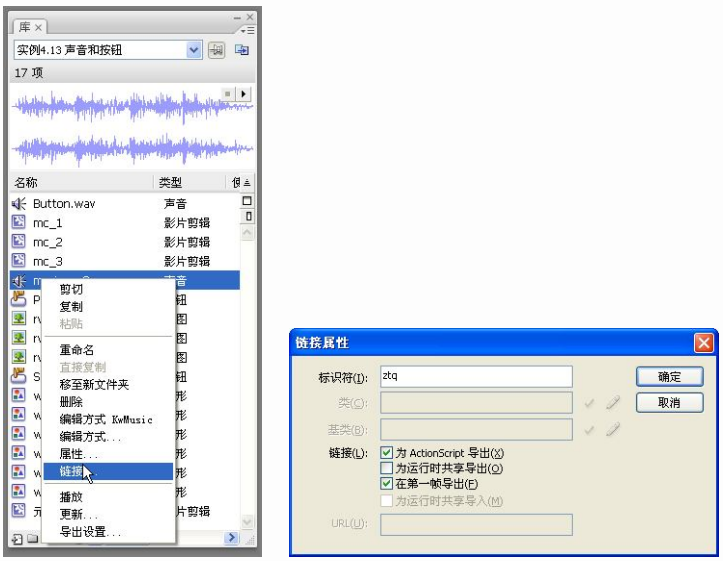

图 4-82　设置声音链接属性

提示

标识符中的“ztq”指定音频的 ID 名称，将会在程序中引用。

步骤 18：在主场景编辑模式中单击“插入图层”按钮，插入一个名为“as”的图层并在第 1 帧中按 F9 键打开“动作”面板，输入如下代码，如图 4-83 所示。

```
att_sound=new Sound();                    //创建声音对象，名为 att_sound
att_sound.attachSound ("ztq");            //附加“ztq”
```

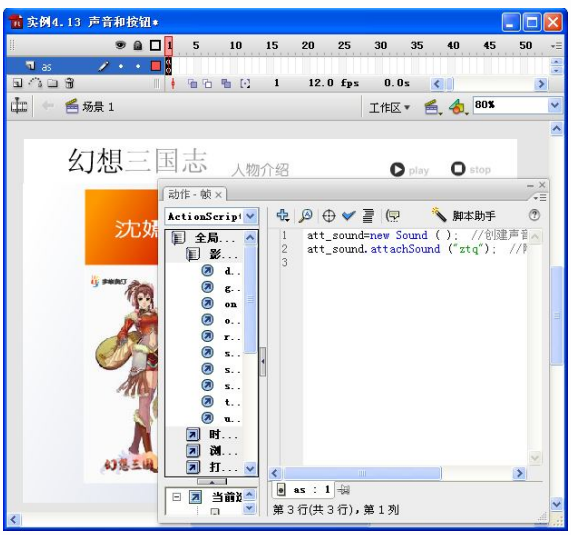

图 4-83　输入代码

提示

此时，已经将声音从库中附加到主场景中，但还不能听到声音。下面将为控制主题曲的声音添加按钮的代码。

步骤 19：单击选中“stop”按钮，按 F9 键打开“动作”面板，输入如下代码，如图 4-84 所示。

```
on (press) {
    att_sound.stop();    //单击按钮，声音停止
}
```

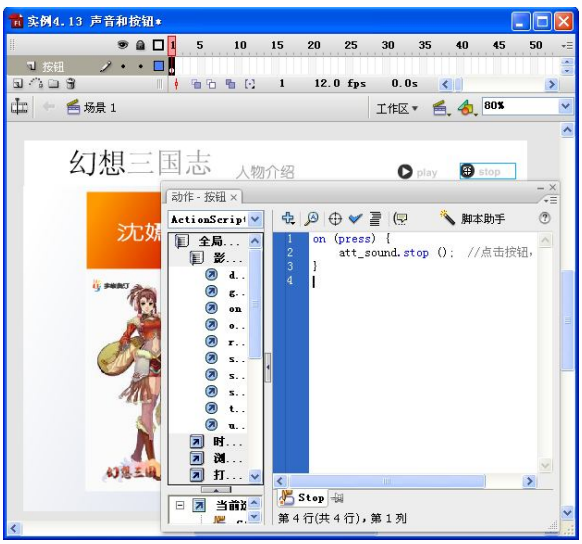

图 4-84　输入按钮代码

步骤 20：单击选中“play”按钮，按 F9 键打开“动作”面板，输入如下代码，如图 4-85 所示。

```
on (press) {
    att_sound.stop();        //单击按钮，声音停止
    att_sound.start();       //声音开始播放
}
```

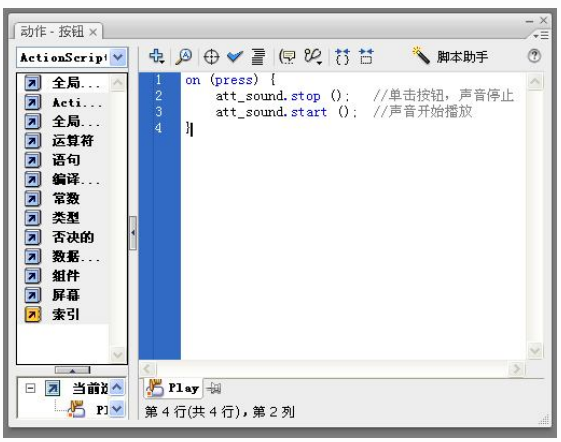

图 4-85　输入按钮代码

提示

先执行“声音停止”，再执行“声音播放”是为了同一时间只有一段声音在播放。声音附加的操作完成。下面为动画的播放添加按钮，并设置按钮声音。

步骤 21：在“按钮”层的第 1 帧，分别绘制 3 个矩形，遮盖住 3 个影片剪辑。将 3 个矩形分别转换为名为“bt_1”、“bt_2”和“bt_3”的按钮元件，如图 4-86 所示。

图 4-86　创建按钮

步骤 22：选中第 1 个按钮，按 F9 键打开“动作”面板。单击左侧动作工具箱中的“全局函数”→“影片剪辑控制”，双击“on”命令，选中“rollOver”单击。在脚本窗格中显示代码，如图 4-87 所示。

```
on (rollOver) {
}
```

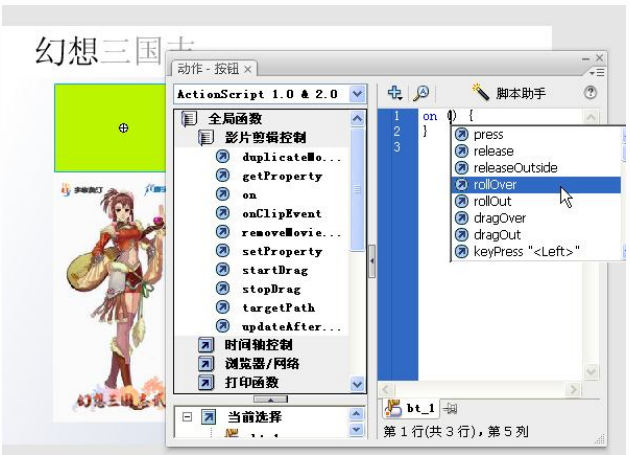

图 4-87　选择按钮命令

步骤 23：在{}之间按回车键，光标停在中间行。单击“插入目标路径”按钮弹出“插入

目标路径”对话框，如图 4-88 所示。

图 4-88　插入目标路径

步骤 24：在“插入目标路径”对话框中选择“sy”，单击“确定”按钮，如图 4-89 所示。将出现如下代码：

图 4-89　选择目标

```
on (rollOver) {
    this.sy
}
```

步骤 25：补充后面的代码“.play();”，如图 4-90 所示。

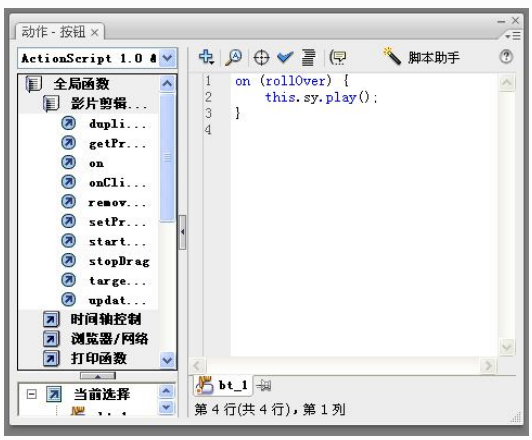

图 4-90　补充代码

```
on (rollOver) {
    this.sy.play();    //鼠标经过按钮时，播放影片剪辑“sy”。
}
```

步骤 26：继续补充代码，如图 4-91 所示。

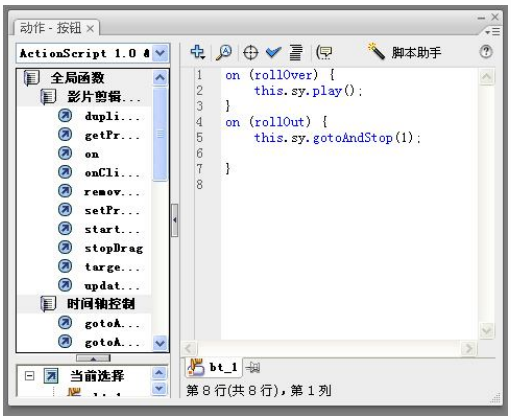

图 4-91　输入代码

```
on (rollOut) {
    this.sy.gotoAndStop(1);  //鼠标移出按钮时，影片剪辑“sy”跳转到第 1 帧并停止。
}
```

步骤 27：选中第 2 个按钮，按 F9 键弹出“动作”面板。按同样的方法输入下面的代码，如图 4-92 所示。

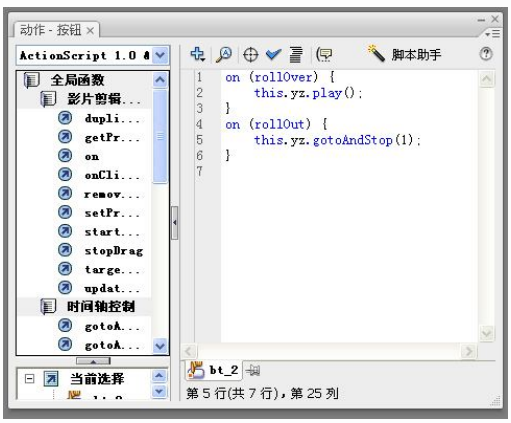

图 4-92　按钮代码

```
on (rollOver) {
    this.yz.play();    //鼠标经过按钮时，播放影片剪辑“yz”。
}
on (rollOut) {
    this.yz.gotoAndStop(1);  //鼠标移出按钮时，影片剪辑“yz”跳转到第 1 帧并停止。
}
```

步骤 28：选中第 3 个按钮，按 F9 键弹出“动作”面板。按同样的方法输入下面的代码，如图 4-93 所示。

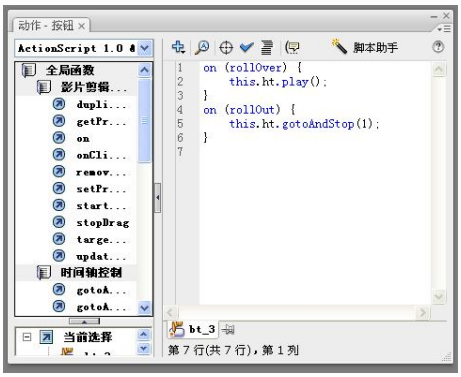

图 4-93　按钮代码

```
on (rollOver) {
        this.ht.play();                    //鼠标经过按钮时，播放影片剪辑“ht”。
}
on (rollOut) {
        this.ht.gotoAndStop(1);   //鼠标移出按钮时，影片剪辑“ht”跳转到第 1 帧并停止。
}
```

步骤 29：双击第 1 个按钮，进入元件编辑模式。插入一个图层。在“指针经过”帧插入空白关键帧。选中该帧，将库中的“Button.wav”音频拖放到舞台，如图 4-94 所示。

图 4-94　按钮声音

提示

此操作设置了当鼠标经过按钮时，发出“Button.wav”的声音。

步骤 30：返回到主场景，将“bt_1”按钮元件的 Alpha 属性设置为 0，并双击进入编辑模式，插入 3 个关键帧。如图 4-95 所示。

提示

将按钮元件的 Alpha 值设置为 0，按钮将隐形，以便可以显示影片剪辑“mc_1”的播放。

图 4-95　按钮元件 Alpha 属性设置

步骤 31：按同样的方法添加其他两个按钮的“指针经过”声音，同样设置按钮的 Alpha 值和插入关键帧，如图 4-96 所示。

图 4-96　按钮声音

步骤 32：按组合键 Ctrl+Enter 预览效果。该实例在单击“play”和“stop”按钮时可以控制主题曲的播放和停止。鼠标经过 3 个影片剪辑位置时，可以播放影片剪辑动画，同时带有声音，如图 4-97 所示。

图 4-97　预览效果

4.4.4 影片剪辑

我们可以使用影片剪辑的方法和函数，在动画播放时动态地创建影片剪辑对象和操作，下面我们将介绍利用 duplicateMovieClip 来复制影片剪辑。

【实例 4.14】影片剪辑复制

步骤 1：新建 Flash 文档，并设置尺寸为 450px×350px，帧频为 12fps，如图 4-98 所示。修改“图层 1”名称为“背景”。

步骤 2：将素材文件夹中的“bj.jpg”图片文件导入到库中，并将其拖放到“背景”图层的第 1 帧中，调整好位置和大小，如图 4-99 所示。

图 4-98 设置文档属性

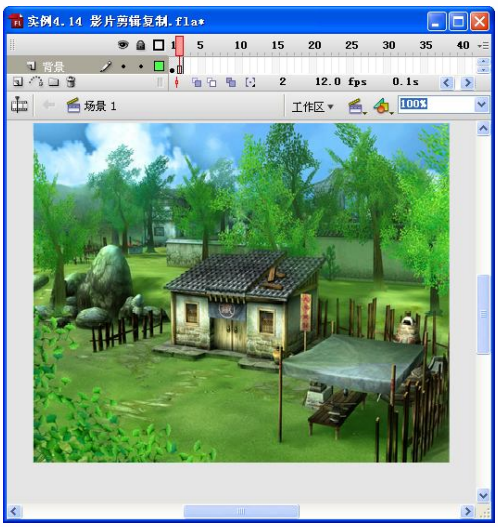

图 4-99 背景图片导入

步骤 3：单击“插入图层”按钮，新建一个名为“影片剪辑”的图层。选用工具箱中的“椭圆工具”在第 1 帧舞台绘制一个正圆。填充颜色为从#FFFFFF 到#A9E466 的“放射状”渐变色，如图 4-100 所示。

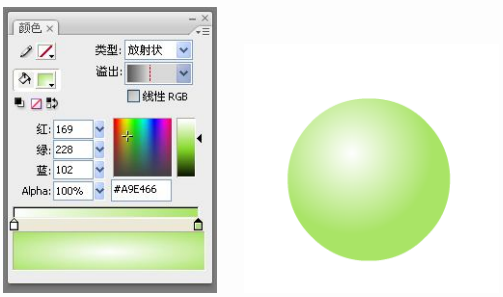

图 4-100 绘制球形

提示

绘制图形后，使用“渐变变形工具”将中心点移动到左上角。

步骤 4：选中圆，按 F8 键将其转换为名为“泡泡”的图形元件，如图 4-101 所示。

图 4-101　转化为元件

步骤 5：选中“泡泡”图形元件，按 F8 键将其转换为影片剪辑元件，命名为“泡泡动”，如图 4-102 所示。

图 4-102　转化为影片剪辑元件

步骤 6：双击“泡泡动”影片剪辑元件，进入元件编辑模式。在第 15 帧处按 F6 键插入关键帧。将第 15 帧的“泡泡”图形元件放大 200%，并设置 Alpha 属性值为 0%。在关键帧间直接创建补间动画，并设置旋转属性的值为“顺时针”和“5 次”，如图 4-103 所示。

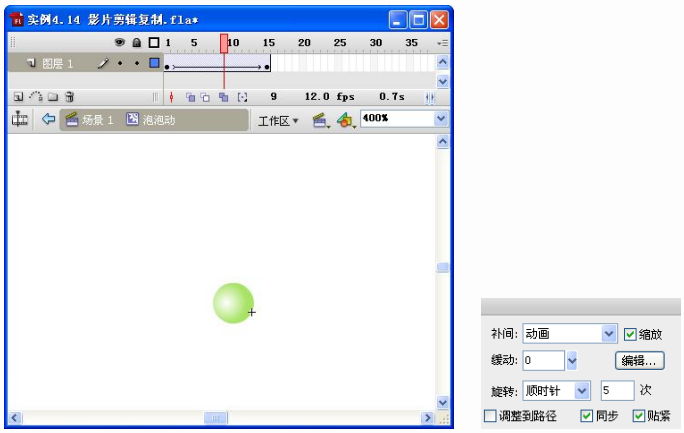

图 4-103　泡泡旋转消失动画制作

提示

此时影片剪辑和动画制作完毕，下面添加代码。

步骤 7：单击“场景 1”按钮返回场景编辑模式，将前面两个图层都延长到第 2 帧。将“泡泡动”影片剪辑拖出舞台，设置实例名称为“move”，如图 4-104 所示。

图 4-104　设置影片剪辑实例名称

提示

该步骤必不可少，在后面的代码中将会引用“move”。

步骤 8：单击“插入图层”按钮，新建一个名为“as”的图层。选中第 1 帧，按 F9 键打开“动作”面板，输入以下代码，如图 4-105 所示。

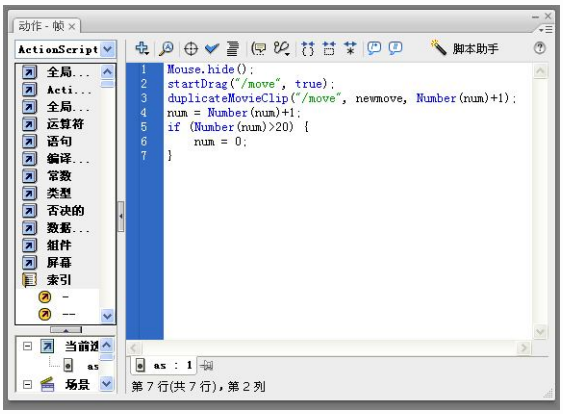

图 4-105　输入帧代码

```
Mouse.hide();      //隐藏鼠标光标，调用了 hide 函数
startDrag("/move", true);  //拖动影片剪辑实例 move，并锁定到鼠标
duplicateMovieClip("/move", newmove, Number(num)+1);
// duplicateMovieClip：复制影片剪辑。复制的目标对象为“move”，复制出的新影片剪辑的名称为“newmove”，新影片剪辑的深度为 Number(num)+1
num = Number(num)+1;    //变量 num 的值等于 Number(num)+1
if (Number(num)>20) {       //判断 Number(num)是否大于 20
        num = 0;                  //变量 num 重新赋值为 0
}
```

步骤 9：选中“as”图层的第 2 帧处，按 F7 键插入空白关键帧。按 F9 键打开“动作”面板，输入以下代码，如图 4-106 所示。

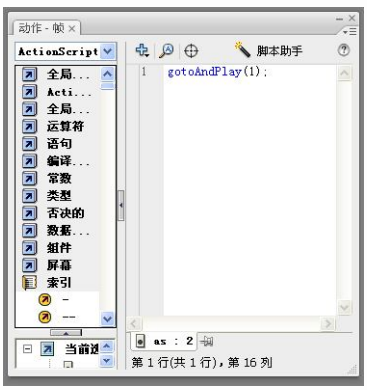

图 4-106　输入帧代码

```
gotoAndPlay(1);      //跳转到第 1 帧并播放
```

步骤 10：按组合键 Ctrl+Enter 预览效果。通过对影片剪辑的复制，鼠标在舞台上移动时，

“泡泡动”影片剪辑动画跟随鼠标重叠播放，如图 4-107 所示。

图 4-107　预览效果

4.4.5　getURL

利用 Flash 制作的动画和游戏、网页等，经常需要链接到其他网页或站点。应用 getURL 可以跳转到其他网页或站点，甚至可以发送邮件。

【实例 4.15】getURL

步骤 1：在素材文件夹中打开之前制作的广告条动画文件，如图 4-108 所示。

图 4-108　打开文件

步骤 2：单击“插入图层”按钮，在图层最顶层插入一个名为“按钮”的图层，如图 4-109 所示。

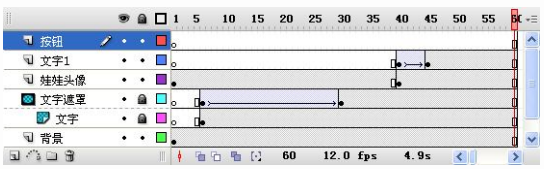

图 4-109　插入图层

步骤 3：在“按钮”图层的第 1 帧，使用“矩形工具”绘制矩形。选中矩形，按 F8 键将其转换为名为“链接”的按钮元件，如图 4-110 所示。

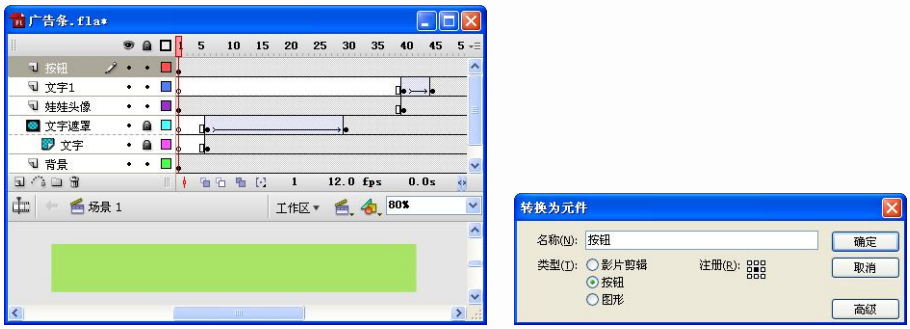

图 4-110　转换为按钮元件

提示

矩形的填充颜色可以任意选择，大小为舞台相同大小。

步骤 4：双击按钮元件，进入元件编辑模式。选中“弹起”帧向后拖动到“点击”帧，如图 4-111 所示。

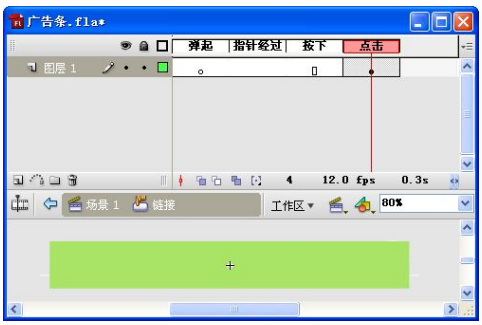

图 4-111　编辑按钮

步骤 5：单击“场景 1”按钮，返回到主场景编辑模式。选中按钮元件，按 F9 键打开“动作”面板，并输入如图 4-112 所示的代码。

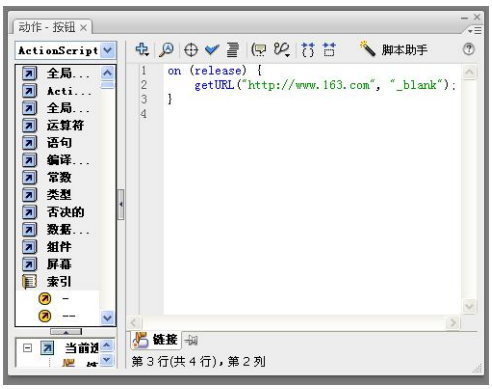

图 4-112　输入按钮代码

```
on (release) {
    getURL("http://www.163.com", "_blank");    //单击按钮链接到 http://www.163.com
}
```

步骤 6：按组合键 Ctrl+Enter 预览效果，如图 4-113 所示单击动画将链接到 http://www.163.com。

图 4-113 预览效果

本章小结

本章介绍了 ActionScript 的相关知识。主要介绍了使用 ActionScript 可以制作出的效果，介绍了 ActionScript 编辑窗口的主要构成和使用方法；讲解了 ActionScript 编程的基础，包括动作、事件、对象、属性、变量、数据类型和运算符等；分析了程序控制语句中主要的条件语句和循环语句，并设计了实例来帮助理解程序控制语句；最后介绍了常用的动作，并使用实例来说明动作的用法和作用。本章的基础知识是 Flash 编程必不可少的，因此我们应该认真地体会和掌握编程基础知识，为 Flash 游戏设计打下基础。

习题四

单选题

1．下列选项中，不能添加脚本的是（ ）。
A．帧　　B．图形元件
C．按钮元件　　D．影片剪辑

2．打开“颜色”面板的快捷方式是（ ）。
A．Shift+F7　　B．Shift+F8
C．Shift+F9　　D．Shift+F10

3．在 Flash 中，“动作”面板中 按钮的作用是（ ）。
A．脚本自动套用格式　　B．语法检查
C．显示代码提示　　D．脚本帮助

4．下列关于 Flash 中基本的数据类型说法错误的是（ ）。
A．Boolean 数据类型包括两个值：true 和 false
B．Number 数据类型可以表示整数、无符号整数和浮点数
C．undefined 数据类型包含一个值：undefined
D．Object 数据类型是由 MovieClip 类定义的

5．下列关于变量的定义，错误的是（ ）。
A．_My_name=good　　B．3_2=hello
C．goto=byby　　D．_$12=yes

多选题

1．下列关于变量类型的描述错误的是（　）。
　A．保留字不能在代码中用作标识符
　B．变量可以任意命名，不需要遵守规则
　C．在函数内声明的变量是局部变量
　D．变量名 BALLS 与 balls 不能被解释为同一变量
2．下列关于变量运算的描述正确的是（　）。
　A．表达式由运算符和操作数组成
　B．赋值运算符有两个操作数
　C．数字表达式使用数值运算符
　D．关系运算符只有一个操作数
3．下列属于 ActionScript 基本数据类型的是（　）。
　A．Boolean　　B．Number
　C．String　　D．MovieClip
4．下列关于 while 循环语句说法正确的是（　）。
　A．while 循环在条件为 true 时重复执行
　B．在不确定一段代码块循环多少次时多使用 while 循环
　C．当条件不再为 true 时，循环将退出
　D．while 循环不会出现无限循环

判断题

1．在递增运算中，a++等同于 a+=1 或 a=a+1。（　）
2．变量名称一定要区分大小写。（　）
3．String 数据类型可以包括 "" 符号。（　）
4．在“动作”面板输入保留字时，系统默认的字符串的颜色是蓝色。（　）
5．一般语言的流程控制分逻辑判断、循环控制和中断控制 3 种。（　）
6．比较字符串是否相同，可以使用关系运算符进行比较。（　）
7．if 是最常用的条件判断式。（　）