第9章　Flash 4的动作命令脚本语言

Flash 4的新动作命令脚本语言，使它的网上交互能力有了很大的提高。

动作命令脚本语言是Flash 4的一种新的脚本语言。由于它的加入，使得Flash 4的交互能力有了前所未有的提高。具有交互能力的脚本语言，在Flash 3中已经出现。但只是一些基于传统动作控制的简单语句，如：“Tell Target”和“Go To”等。

在Flash 3中，跟踪内部数据只能依靠一种叫“动画片段图符逻辑”（Movie Clip Logic）的方法。这种方法是通过“Tell Target”调用动画片段图符来保持计数的。一个实用的例子是图9-1所显示的猜数字游戏。用Flash 3，使用上述的方法，设计这样一个游戏，需要150K储存空间。而且游戏中的数据储存和逻辑判断都需要借助动画片段图符。而用Flash 4的动作命令脚本语言设计这个游戏，只需4K储存空间，并且可以直接访问数据库。

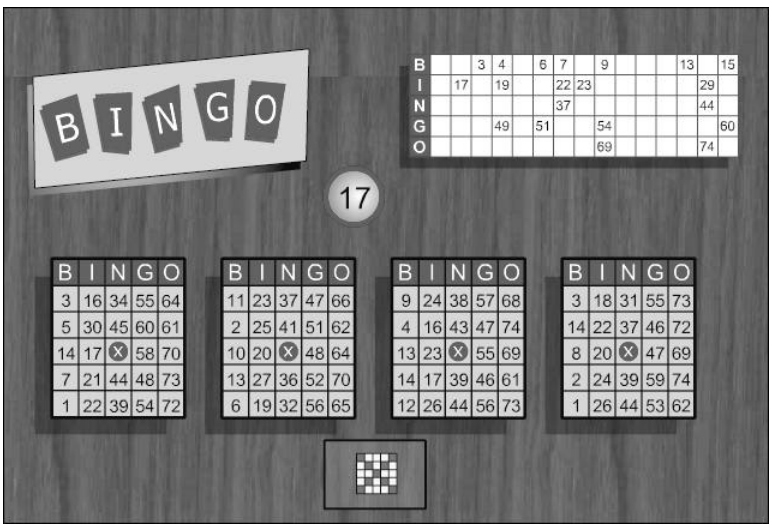

图9-1　猜数字游戏

还有一个例子，就是图9-2所显示的模拟积木游戏。它是利用Flash 4的动作命令脚本语言中的“Drag Movie Clip”和“Duplicate Movie Clip”等语句设计的。这两个例子都可以在网站：www.FlashCentral.com上找到。

Flash 4的动作命令脚本语言短小精悍，它包含了常用的数学、字符串和逻辑运算能力、少量的基本命令和内置功能及二十一种属性。Flash 4动作命令脚本语言的命令与功能是根据“能实现最大的功能，只需要最小的播放器”原则设置的。如果需要附加功能可以访问www.FlashBible.com/members网站。

Flash 4的动作命令脚本语言扩展了它的动作命令的功能。许多新程序员和非职业程序员都愿意在Flash 4提供的编程环境下进行编程。但是对于那些有经验的程序员，这个编程环境太小、而且不接受由其他文本编辑器拷贝过来的脚本，可能有些不便。Flash 4提供的编程环境，只能

将脚本拷出，不能将脚本拷入。但是在Flash 4的编程环境里，动作命令脚本是可以相互拷贝的。并且可以建立一个常用动作命令脚本库，再编程的时候可以直接从动作命令脚本库中拷贝有用的脚本。

图9-2 模拟积木游戏

9.1 变量

下面先从变量开始。简单地说，变量是信息的存储空间，存入这个存储空间的信息可以是变化的，但是信息一旦存入这个存储空间就不会再发生任何改变。

9.1.1 变量的类型

在Flash 4中，变量有三种类型：数值变量、字符串变量和逻辑变量。数值变量可以包括所有可以进行数学运算的数字。字符串变量可以是一个以上的字符或数字图符。逻辑变量只有两个值：“ True ”、“ False ”或用零表示False，用非零表示True。

在下面的例子中，将创建动画片段图符“ OriginalMC ”的副本，其副本的名字将存入字符串变量“ NewMC ”中。它的动作命令为：

```
Duplicate Movie Clip ("OriginalMC", NewMC, Leve1)
```

尽管OriginalMC和NewMC都是字符串，但是在这里，OriginalMC是字符串，而NewMC是字符串变量。在Flash 4的动作命令脚本语言中，字符串必须用双引号括起来，而字符串变量则不用。这就是字符串与字符串变量的区别。

下一个例子是利用Set Property动作命令将逻辑值false赋给名字为Globe的变量，使它成为不可见的。

```
Set Property ("/Globe", Visibility) = false
```

接下来的这个例子，是将一个逻辑变量 Visible的值赋给名字为Globe的对象。如果逻辑变量Visible的值为false，Globe动画片段图符将不可见；当逻辑变量 Visible的值为True，Globe动画片段图符将成为可见的。

```
Set Property ("/Globe", Visibility) = Visible
```

请注意在这两个例子中的等号“ = ”，它是一个运算符号。一个表达式包括变量、常量和运算符号。表达式将会得到一个运算结果。这个运算结果可以赋值给一个变量，也可以是另一个表达式的一部分。表达式和运算符号将是本章的重点。

9.1.2 输入变量

在很多脚本语言中，都必须预先设定变量的类型。某种类型的数据只能赋值给同种类型的变量。在Flash 4中，变量不必预先设定类型，它的类型由它后面跟的运算符号决定。例如：变量后面跟的是数学运算符号，它就是数值变量、变量后面跟的是逻辑运算符号，它就是逻辑变量。

图9-3 动作控制命令选择菜单

图9-4显示的是Frame Properties对话框。在Frame Properties对话框左侧的窗口，是Flash 4的编程环境。点击它左上角带有加号的按钮就可以打开动作控制命令选择菜单。图 9-3所显示的就是 Frame Properties对话框的动作控制命令选择菜单。在动作控制命令选择菜单中，可以选Set Variable命令直接给变量赋值。当Set Variable命令被选中后，它会出现在Flash 4的编程环境中（见图9-4左侧窗口），然后可以分别输入变量名和变量数值。

在变量名输入窗口（Variable），输入变量名。点击变量名输入窗口右侧的“ abc ”按钮，可以选择变量名的类型。在点“ abc ”按钮后，打开选择菜单。在该菜单中，可看到 String Literal、Expression和Expression Editor选项。String Literal选项可以将输入的变量名定义为字符串。 Expression选项可以将输入的变量名定义为变量名。Expression Editor选项可以将输入的变量名带入表达式编辑环境。

在变量值输入窗口（Value），输入变量值。按变量名输入窗口右侧的“ abc ”按钮，可以选择变量值的类型。按“ abc ”按钮后，打开选择菜单，在该菜单中有 String Literal、Expression和Expression Editor选项。String Literal选项可以将输入的变量值定义为字符串赋值给变量名。Expression选项可以将输入的变量值赋值给变量名。 Expression Editor选项可以将输入的变量值带入表达式编辑环境。

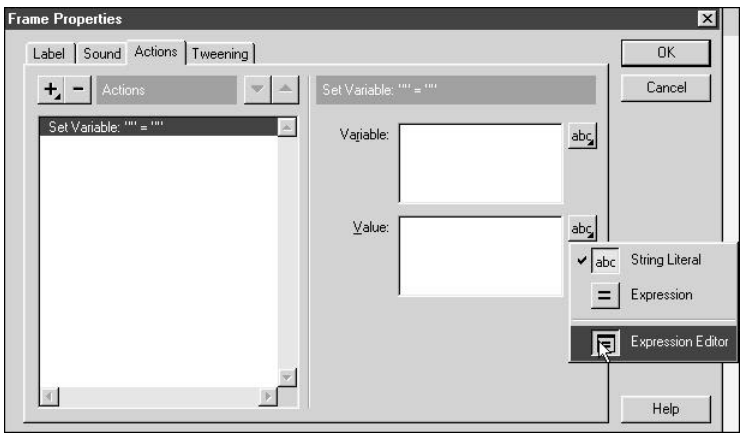

图9-4 Frame Properties对话框

变量的路径

如果所用的变量都是在同一个时间线上，在使用变量的时候可以直接使用变量名。但是所用的变量是在其他时间线上，在使用变量时除了要有变量名外，还需要指出变量的路径。变量路径的表达形式与动画片段图符的路径表达形式一样。具体的使用方法是，变量的路径在前，变量名在后，中间加冒号隔开。例如：

/movieClip1/movieClip2:variableName

9.2 运算符号

运算符号是在表达式中起连接、比较和求值的数学符号。例如，加号可以求两个数之和，连接号“ & ”可将两个字符串连接起来。图9-5就是将两个字符串连接起来的例子。

通过下面的例子可以看出，虽然参加运算的数值是相同的，但是由于运算符号不同，得出的结果也不同。图9-6是表达式编辑器，左文字框是运算符号。表9-1是Flash 4中的运算符号。

3+12=15

3&12=312

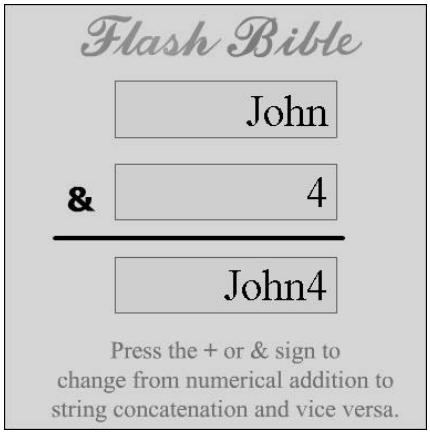

图9-5 将两个字符串连接起来

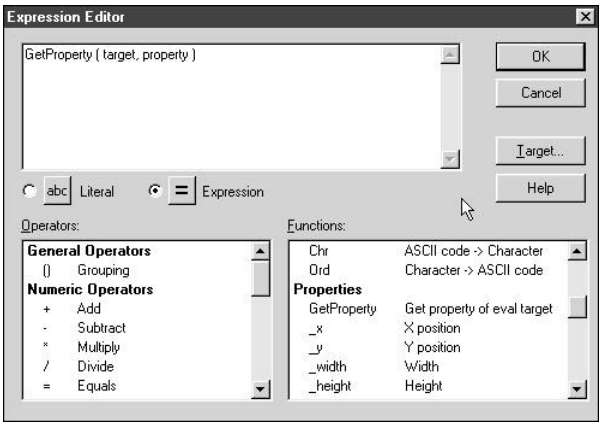

图9-6 表达式编辑器

表9-1 运算符号的类型

基本运算符号	
运算符号	**符号意义**
( )	**括号**
算术运算符号	
运算符号	**符号意义**
+	**加号**
-	**减号**
*	**乘号**
/	**除号**
数值比较运算符号	
运算符号	**符号意义**
=	**等于**

(续)

<>	不等于
<	小于
>	大于
<=	小于等于
>=	大于等于
字符串运算符号	
运算符号	**符号意义**
“ “	双引号
&	连接号
字符串比较符号	
运算符号	**符号意义**
eq	等于
ne	不等于
lt	小于
gt	大于
le	小于等于
ge	大于等于
逻辑运算符号	
运算符号	**符号意义**
AND	与
OR	或
NOT	非

9.3 表达式

在Flash 4中，有三种基本表达式：数值表达式、字符串表达式和逻辑表达式。一种表达式中可以包含另外两种表达式，但是表达式的最终结果必须是数值、字符串或逻辑值。图 9-6显示的是表达式编辑器。

9.3.1 数值表达式

数值表达式的结果一定是数值。数值表达式是由算术运算符号与以下的各种数据组成的。

- 数字。
- 数值变量。
- 返回数据的函数。
- 以上三种数据的任意组合

Flash 4数值表达式的运算遵循标准的数学运算规则，括号内的运算优先处理，先乘除后加减。图9-7所显示的是一个可以进行动态运算的动画作品。感兴趣的读者可以访问 www. FlashBible.com 网站。

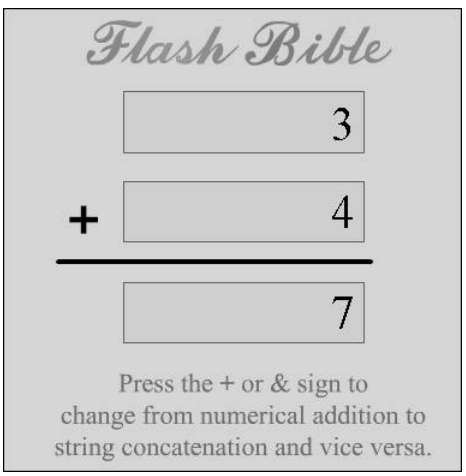

图9-7 可以进行动态运算的动画作品

下面的数值表达式，是将1加2的结果赋值给变量“ Result ”：

```
Set Variable: "Result" = 1 + 2
```

这个表达式在脚本编辑器中的情况见图9-8。

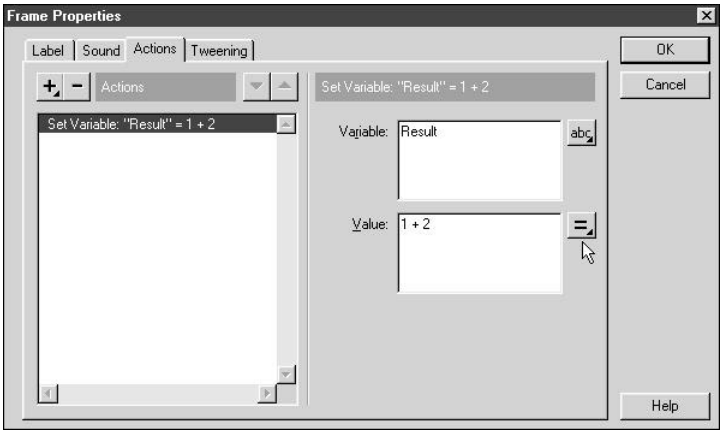

图9-8 表达式在脚本编辑器中

下面的数值表达式，是将变量“ Subtotal ”加1的结果赋值给变量“ Result ”：

```
Set Variable: "Result" = Subtotal + 1
```

如果变量“ Subtotal ”没有被预先赋值的话，它的值将被认为0。

如果一个字符串中只有数字没有其他字母，这个字符串将做为数字参加运算。如果一个字符串中除了有数字还有其他字母，这个字符串将做为数字0参加运算。

下面的数学表达式，是将24赋值给变量“ Result ”。因为字符串中没有其他字母：

```
Set Variable: "Result" = "23" + 1
```

而下面的数学表达式，是将1赋值给变量“ Result ”。因为字符串中有其他字母：

```
Set Variable: "Result" = "23B" + 1
```

下面的数学表达式，是将7.5赋值给变量“ Result ”。因为字符串中没有其他字母：

```
Set Variable: "Result" = 10 + "- 2.5"
```

而下面的数学表达式，是将10赋值给变量“ Result ”。因为字符串中有空格：

```
Set Variable: "Result" = 10 + "- 2.5  "
```

如果用0去做除数，结果将是#ERROR#。

9.3.2 字符串表达式

字符串表达式的结果一定是字符串。字符串表达式是由字符串运算符号与以下的各种字符串组成的：

- 字符串。
- 字符串变量。
- 从功能调用返回的字符串。
- 以上三种字符串的任意组合。

下面的字符串表达式，是将字符串“ John ”赋值给变量“ Result ”。

```
Set Variable: "Result" = "John"
```

这个表达式在脚本编辑器中的情况见图9-9。

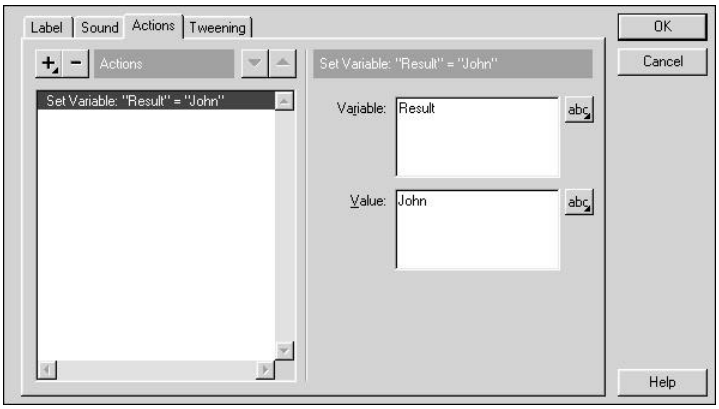

图9-9 表达式在脚本编辑器中

9.3.3 逻辑运算表达式

逻辑运算表达式的结果一定是True或False。逻辑运算表达式是由逻辑运算符号与以下的各种数据组成的：

- 参与比较的字符串。
- 参与比较的数字。
- 从功能调用返回的逻辑值。
- 以上三种字符串的任意组合。

逻辑运算表达式常被用在条件语句或循环语句中。在Flash 4中，也可以用0代表False，用1或任何非零的值代表True。任何未经过赋值的字符串将返回一个逻辑值0（False）。

9.4 比较

在Flash 4中，既可在两个数学表达式之间进行比较，又可在两个字符串表达式之间进行比较，当然也可以将数据和字符串放到逻辑运算表达式中进行比较。

9.4.1 数值比较

数值比较就是对两个数值表达式的数值进行比较。数值比较的结果是逻辑值“ True ”（1）或“ False ”（0）。如果比较结果满足比较条件，其结果为“ True ”（1）。如果比较结果不满足比较条件，其结果为“ False ”（0）。图9-10所显示的是一个抛球的演示动画。它就是利用数值比较来判断球的位置，以保证球能准确地进入垃圾桶中。对本演示动画感兴趣的读者可以访问www.FlashBible.com/members/Drag/DropTarget.htm网站。

下面的例子，是将“ True ”（1）赋值给变量“ Result ”。因为8大于5是成立的：

```
Set Variable: "Result" = "8 > 5"
```

而下面的例子，是将“ False ”(0）赋值给变量“ Result ”。因为9等于5是不成立的：

Set Variable: "Result" = "9 = 5"

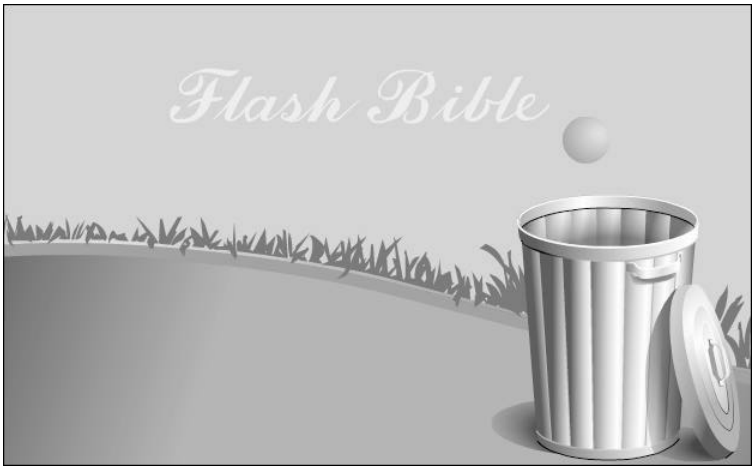

图9-10 一个抛球的演示动画

9.4.2 字符串比较

字符串比较就是对两个表达式的字符串进行比较。字符串比较的结果是逻辑值“ True ”(1）或“ False ”(0)。如果比较结果满足比较条件，其结果为“ True ”(1)。如果比较结果不满足比较条件，其结果为“ False ”(0)。字符串的比较是从每个表达式的最左侧的字母开始，由左至右依次进行。字母的值就是它的ASCII码，ASCII码越大，字母的值也就越大。

根据ASCII码的规律，大写字母的ASCII码小于小写字母的ASCII码。排在字母序列表前部的字母，其ASCII码小于排在字母序列表后部字母的ASCII码。

下面的例子，是将“ True ”(1）赋值给变量“ Result ”。因为字符串“ bats ”比字符串“ bat ”多一个字母“ s ”，满足比较条件：

Set Variable: "Result" = "bats gt bat"

而下面的例子，是将“ False ”(0）赋值给变量“ Result ”。 因为字符串“ bat ”与字符串“ bat ”相等，不满足比较条件：

Set Variable: "Result" = "bat gt bat"

下面的例子，是将“ True ”(1）赋值给变量“ Result ”。因为字符串“ bat ”比字符串“ Bat ”大，满足比较条件：

Set Variable: "Result" = "bat gt Bat"

而下面的例子，是将“ True ”(1）赋值给变量“ Result ”。因为字符串“ bet ”比字符串“ bat ”大，满足比较条件：

Set Variable: "Result" = "bet gt bat"

9.4.3 利用逻辑运算表达式进行联合比较

数值比较和字符串比较可以由逻辑运算符号连接形成联合比较。 Flash 4中逻辑运算符号有

三种：与“ AND ”、或“ OR ”、非“ NOT ”。表9-2列出了逻辑运算表达式的运算关系。

表9-2 逻辑运算结果

逻辑值1	逻辑运算符号	逻辑值2	结 果
0	AND	0	0
1	AND	0	0
0	AND	1	0
1	AND	1	1
0	OR	0	0
1	OR	0	1
0	OR	1	1
1	OR	1	1

下面的这个表达式，是由逻辑运算符号“ AND ”将两个数值比较表达式联合到了一起：

```
(Result >= 20) AND (Result <= 30)
```

如果“ Result ”的值是在20和30之间，两个数值比较条件都满足，逻辑逻辑运算的结果是“ True ”(1)。如果“ Result ”的值不在20和30之间，两个数值比较条件中，将至少有一个不能满足，逻辑逻辑运算的结果则是“ False ”(0)。

如果要判断“ Result ”的内容是否是“ Red ”或“ Blue ”。可以利用下面的逻辑运算表达式进行判断。

```
(Result eq "Red") OR (Result eq "Blue")
```

只要“ Result ”的内容是“ Red ”或“ Blue ”，表达式的结果将是“ True ”(1)。逻辑运算表达式一般用在条件语句或循环语句中。

9.5 条件语句

Flash 4中的条件语句可以用来控制动画的走向。所有的条件语句都必须包括判断条件和对应满足判断条件或不满足判断条件的下一步指令。

这里用生活中的一个实例，来说明条件语句的用法。假如，今天如果要下雨，需要带雨衣。今天如果不下雨，就把雨衣留在家中。用条件语句来描述以上事件的具体脚本如下：

```
If (Raining)
    Set Variable: "Display" = "Wear the raincoat"
End If
Else
    Set Variable: "Display" = "Leave the raincoat at home"
End If
```

图9-11所显示的拼字游戏就是以条件语句为主编写的。对本演示动画感兴趣的读者可以访问www.FlashBible.com/members/ActionScript/PlugNPlay/Strings.htm网站。

下面用一个在动画创作中常遇到的问题，来具体介绍一下如何使用条件语句。假设要创作一个交互式的动画作品，动画的进程由浏览者控制。如果浏览者选择红色“ Red ”，动画从第一帧开始播放。如果浏览者选择蓝色“ Blue ”，动画从第20帧开始播放。如果浏览者选择绿色“ Green ”，动画从第30帧开始播放。如果浏览者选择了其他颜色，动画从第40帧开始播放。

以下是利用Flash 4的脚本编辑环境，根据以上条件编写脚本的具体步骤：

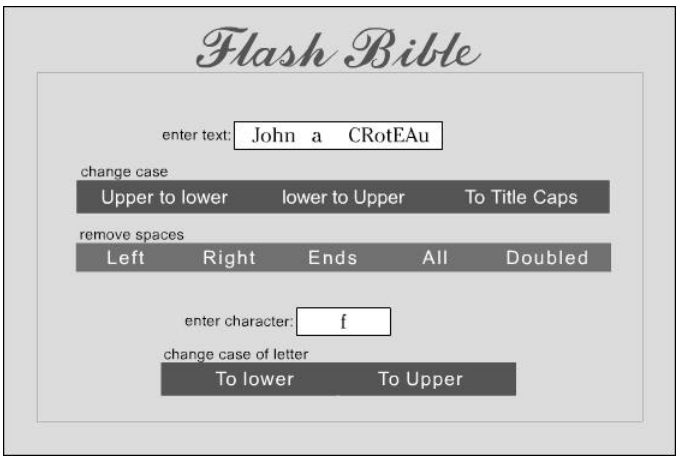

图9-11 拼字游戏

1) 进入Flash 4的脚本编辑环境，点位于脚本编辑环境左上角的加号“ + ”按钮，打开下拉菜单。在下拉菜单中选“ If ”。进入条件语句输入状态（见图 9-12）。在条件输入窗口中输入：Color eq "Red"。

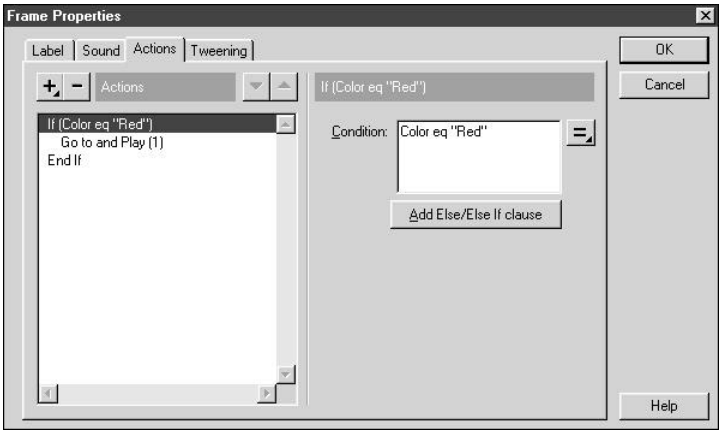

图9-12 进入条件语句输入状态

2) 点位于脚本编辑环境右上角的加号“ + ”按钮，打开下拉菜单。在下拉菜单中选Go To。进入Go To语句输入状态。在Go To语句输入状态中，选Go to and Play，并在Number输入区域中输入10。

这时脚本为：

```
If (Color eq "red")
  Go to and play ("Red")
End If
```

3) 选中第一行语句，点三次“ Add Else / Else If clause ”按钮。

4) 选中第一个“ Else ”语句，进入“Else ”输入状态。在“Else ”输入状态中，选“Else If ”。在条件输入窗口中出入“ Color eq “ Blue ””。

5) 点位于脚本编辑环境左上角的加号“ + ”按钮，打开下拉菜单。在下拉菜单中选Go To。

进入Go To语句输入状态。在Go To语句输入状态中，选Go to and Play，并在Number输入区域中输入“20”。重复4、5步骤完成脚本编写。完成后的脚本为：

```
If (Color eq "red")
    Go to and play (10)
Else If (Color eq "Blue")
    Go to and play (20)
Else If (Color eq "Green")
    Go to and play (30)
Else
    Go to and play (40)
End If
```

9.6 循环语句

在Flash 4中的循环语句有两种基本类型，While循环和If循环。循环的进程是由它的退出条件来控制的。当While后的条件判断语句得到满足时（取True值），While循环将重复进行，否则，While循环将立即退出。在Flash 4中，循环的次数被限制在200 000次。这主要是考虑到，如果循环次数太多会被浏览者误认为已经死机了。

While循环一般包括三部分，退出条件判断语句、循环计数器和动作执行语句。退出条件可以是数值，也可以是字符串。下面是While循环的基本形式。

```
Loop While (Condition)
    Action 1
    Action 2
End Loop
```

如果要将角度变量Angle中的值转换成小于360度的数值，可以用While循环来完成。具体脚本如下：

```
Loop While (Angle >360)
    Set Variable: "Angle" = Angle - 360
End Loop
```

如果将角度变量“Angle”中的值限制在小于5000 000度的范围内，具体脚本将成为以下形式：

```
If (Angle < 5000000)
    Loop While (Angle >360)
        Set Variable: "Angle" = Angle - 360
    End Loop
Else
    Set Variable: "Angle" = "That angle was ridiculously large."
End If
```

如果退出条件判断语句不起作用的时候，循环将无限地执行下去。图9-13中所显示的就是一个无限循环。它的脚本如下：

```
Loop While (Angle)
    Set Variable: "Angle" = Angle + 20
End Loop
```

遇到这种情况，Flash 4的200 000次循环限制机制将起作用。当循环次数大于200 000次时，

Flash 4将强制停止上述循环。但是有时因为某些特殊的需要，希望循环次数大于 200 000次。这时就需要使用If循环语句。

If循环语句与While循环语句一样，也是由三大部分组成：

- 退出条件判断部分。该部分可以是数学表达式或字符串表达式。
- 循环计数器和动作执行语句。
- 返回语句Go To，使进程返回循环起始点。

下面的例子是利用If循环进行两帧之间的延时。

首先，将第一帧命名为BeginUp并且无动作，在第二帧上附加以下脚本：

```
Set Variable: "/: Elapsed" = ( GetTime - Start )/1000
If ( /: Elapsed < /: Delay )
    Go to and Play ("BeginUp")
Else
    Go to and Play ("EndDelay")
End If
```

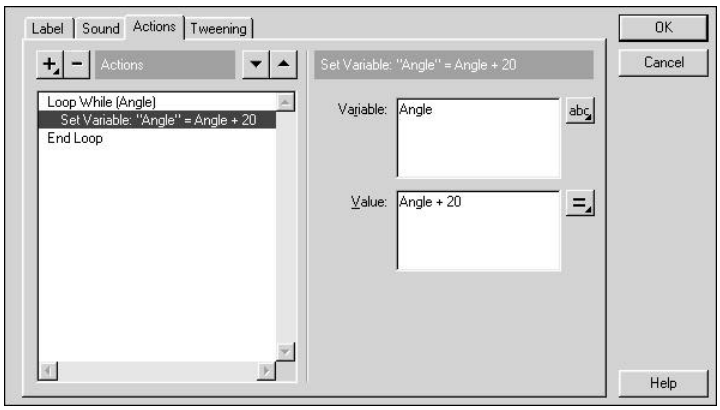

图9-13 一个无限循环

9.7 属性函数

Flash 4提供了三种属性函数，动画片段图符属性函数、全局属性函数和文本滚动属性函数。

动画片段图符属性函数可以作用于每一个动画片段图符。可以通过“ Get Property ”命令得到当前动画片段图符的状态，也可以通过“ Set Property ”命令改变动画片段图符属性函数的值，来改变动画片段图符的状态。在表9-3中列出了十六种动画片段图符的属性函数。图9-14是一段动画片段图符属性函数功能的演示动画，通过它可以了解动画片段图符属性函数是如何控制动画片段的状态的。

图9-14 动画片段图符属性函数的演示动画

全局属性函数是确定整个影片的属性的，它包括三个函数。文本滚动属性函数是用来控制文本在文

本显示窗口的滚动的，它包含两个函数。

动画片段图符属性函数和全局属性函数都是由“ _”符号开头。动画片段图符属性函数和全局属性函数只在表达式编辑状态下出现。

在Flash 4中，动画片段图符属性函数对动画片段图符状态的控制是以它的启始点做为参考点的，而不是以它的当前状态为参考点。例如某个动画片段图符现在处于旋转 45 的状况。如果再利用动画片段图符属性函数让它旋转 45°，它将不会改变目前的状态。因为它已经处于旋转45 的状况。

表9-3 属 性 函 数

属性函数变量	是否可设定	函数变量编号	注 解
动画片段图符属性函数			
_x	是	0	对象在x轴的坐标值
_y	是	1	对象在y轴的坐标值
_width		8	对象的宽度
_height		9	对象的高度
_rotation	是	10	对象的旋转角度
_target		11	对象的路径
_name	是	13	对象的名字
_url		15	对象的URL地址
_xscale	是	2	对象的x轴缩放系数
_yscale	是	3	对象的y轴缩放系数
_currentframe		4	获得当前帧位置
_totalframes		5	获取时间线上的全部帧数
_framesloaded		12	当前已装载的帧数
_alpha	是	6	对象的alpha系数
_visible	是	7	设定对象是否可见
_droptarget		14	获得动画片段图符的名字和路径
全局属性函数			
_highquality	是	16	高画质显示
_focusrectangle	是	17	显示聚焦范围
_soundbuftime	是	18	设定声音缓存时间（默认值为5秒）
文本滚动属性函数			
variable_name.scroll	是		文字显示区域中首行行号
variable_name.maxscroll			文字显示区域中首行最大可用行号

表9-3列出了属性函数变量。属性函数变量如果是可以设定的，就意味着属性函数变量是可读和可写的。属性函数的编号是用于FSCommands命令的。详细的用法将在第12章中讨论。

9.7.1 动画片段图符的坐标系

当动画片段图符被多层引用时，常常会发现一些动画片段图符不按预期的方法运动。要解决这个问题，首先要了解动画片段图符坐标系的原点和当它被其他动画片段图符引用时其坐标系的变化。

当某个动画片段图符处于主时间线上，该动画片段图符的坐标系原点在画布的左上角。在这种情况下，利用动画片段图符属性函数得到的“ _x”和“ _y”值都是相对这个原点的。而当

一个动画片段图符被另外一个动画片段图符调用，它的坐标系原点就转移到调用它的动画片段图符的中心了。这是再用动画片段图符属性函数得到的“ _x ”和“ _y ”值都是相对这个新原点的。例如，一个动画片段图符被指定做由左向右的直线运动。当它被某个顺时针方向旋转了90的动画片段图符调用时，它将不再做由左向右的直线运动，而是做由上至下的直线运动。这是因为，它的坐标系被调用它的动画片段图符沿顺时针方向旋转了90°。

图9-15所显示的是一个动画演示游戏，画面上的字母可以由鼠标拖着在画面上移动。

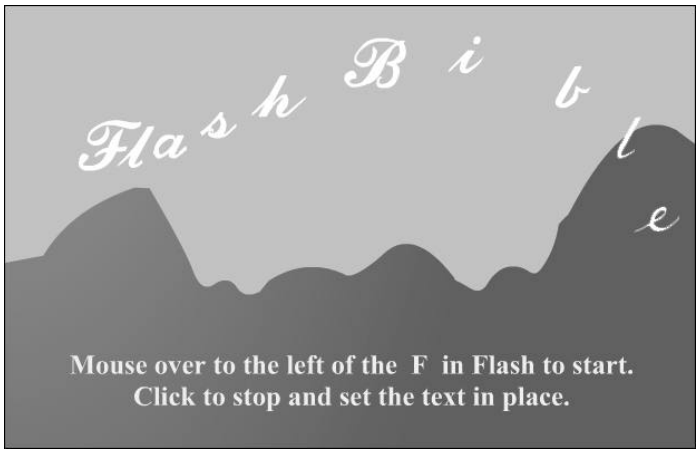

图9-15 一个动画演示游戏

9.7.2 动画片段图符属性函数

每个动画片段图符都可以通过动画片段图符属性函数得到它的属性或改变它的状态。并不是所有的动画片段图符属性函数都是可以设定的，正如表 9-3所示。不能设定的动画片段图符属性函数是只读的。只能通过执行GetProperty命令得到画片段图符当前的状态。

下面的这条语句就是要将一个名字为“ TBall ”的动画片段图符在x轴的坐标值赋值给变量“ Xpos ”。具体语句为：

```
Set Variable: "Xpos" = Getproperty ("/Tball",_x )
```

图9-16所显示的是通过脚本编辑环境，输入上述语句的情况。

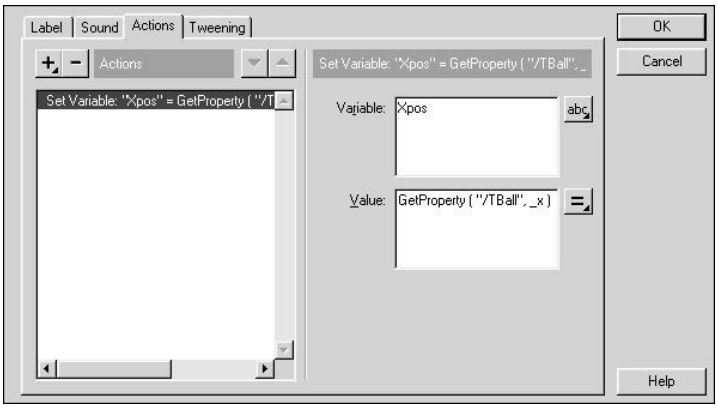

图9-16 通过脚本编辑器输入变量

动画片段图符属性函数如下：

- _alpha —— 对象的alpha系数，alpha值可以是0到100中的任何数。
- _currentframe —— 当前帧在时间线上的位置。
- _droptarget —— 当对象被拖着从其他动画片段图符上方掠过，将返回在其下方的动画片段图符的名字和路径。
- _frameloaded —— 当前已装载的帧数。
- _height和_width —— 获得对象的高和宽，单位为像素。
- _name —— 对象的名字
- _rotation —— 获得或设定对象的旋转角度。它不但作用于对象，而且对当前对象所调用的动画片段图符也起同样作用。
- _target —— 对象的路径。
- _url —— 对象的URL路径。
- _visible —— 设定对象是否可见。对象如果被设定为不可见，它只是不被显示，但依然存在。
- _x和_y —— 获得或设定对象在x轴或y轴的坐标值。它不但作用于对象，而且对当前对象所调用的动画片段图符也起同样作用。
- _xscale和_yscale —— 获得或设定对象在x轴或y轴方向的缩放系数，默认值为100%。它不但作用于对象，而且对当前对象所调用的动画片段图符也起同样作用。

Flash 4提供了下拉式的动画片段图符属性函数选择窗口，这个选择窗口中可以方便地选择需要的动画片段图符属性函，见图9-17。

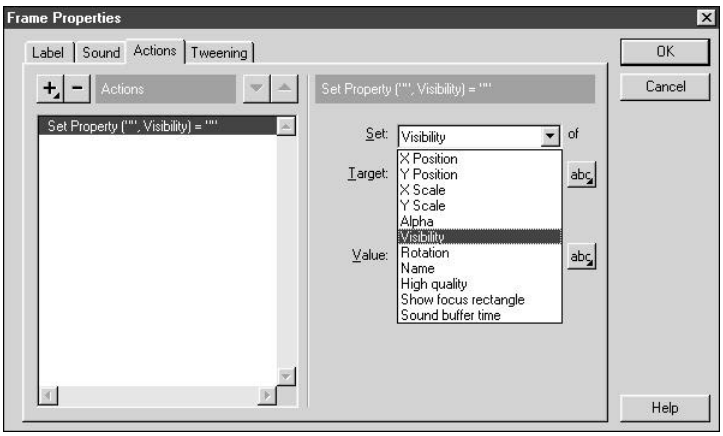

图9-17 下拉式的动画片段图符属性函数选择窗口

9.7.3 全局属性函数

全局属性函数制作用于当前影片，而对动画片段图符不起作用。

全局属性函数如下：

- _focus rectangle —— 确定是否显示聚焦方框。
- _high quality —— 设定是否打开自动边缘平滑功能。1为打开，0为关闭。

• _soundbuftime——设定声音的缓存时间。默认值为5秒。

9.7.4 文本滚动属性函数

文本滚动属性不出现在表达式编辑器，并与其他属性函数不同，不用下划线开头，以.scroll或.maxscroll结尾。

• var_name.scroll——设定文字显示区域中的文本内容从哪一行开始显示（即文字显示区域中首行是文本内容的第几行）。默认值为1，即第一行。该值可以设定。

• var_name.maxscroll——在文字显示区域中的文本内容首行最大可从哪一行开始显示。该值是由Flash 4根据文本显示区域的高度自动设定的。该函数只能读取，不能设定。

9.8 内部函数

Flash 4提供了一些内部函数，每一种内部函数都可以完成一种特定的功能。表 9-4显示了Flash 4所有的内部函数和它们的功能。

表9-4 内部函数的功能

函 数 名	注 解	结果数据类型
Int	将非整型数转换成整型数	整型数
Random	返回随机数	整型数
Get Timer	取得已播放的时间	毫秒
True	逻辑"真"	1
False	逻辑"假"	0
Newline	插入新行	增加新行
Substring	提取子字符串	字符串
Length	计算字符串长度	整型数
Chr	根据ASCII值返回字符	单个字符
Ord	根据字符返回ASCII值	整型数
Eval	返回字符串表达式的值	字符串或整型数
MBSubstring	提取双字节子字符串	字符串
MBLength	计算双字节字符串长度	整型数
MBChr	根据ASCII值返回双字节字符	单个字符
MBOrd	根据双字节字符返回ASCII值	整型数

1. 将非整型数转换成整型数

在Flash 4中的Int函数，可以将非整型数转换成整型数。当一个非整型数转换成整型数后，整型数值的绝对值将小于或等于它原来的值。将字符串数转换成整型数，其值为 0。

它的语法是：Int(Num)；“ Num ”为需转换的非整型数。

下面是“ Int ”函数实际用法。

```
Set Variable: "Integer" = Int (Num)
```

以下是上式的计算结果：

如果Num=2.2　　结果为2

如果Num=4.9　　结果为4

如果Num=-7.2　　结果为-7

如果Num=-5.9　　结果为-5

如果Num=22.4B 结果为0

2. 产生一个随机数（Random）

内部函数Random可以产生一个随机的整型数。

它的语法是：Random (Num)，Num为可产生随机整型数的上限。

下面是“随机的整型数”函数实际用法：

```
Set Variable: "Result" = Random (10) + 1
```

上式可以产生0到10之间的随机数。

3. 获得已播放的时间（Get Timer）

内部函数Get Timer可以获得当前影片已播放的时间。它的单位是毫秒。

下面是它的具体用法：

```
Set Variable: "StartTime" = GetTime
Set Variable: "Elapsed" = (GetTime - Start) / 1000
```

4. 逻辑函数（True和False）

逻辑函数True和False可以做为逻辑值，出现在逻辑表达式中。 True有时可以用1表示，False可以用0表示。

5. 插入新行(Newline)

内部函数Newline的功能，相当于按回车键，可以将它以后的内容转入下一行。通过以下的实例可以进一步了解它的功能：

```
Set Variable: "Result" = "Name" & "New line & Numb"
Set Variable: "Result" = "Name" & Chr (13) & "Number"
```

结果为：

```
Name
Number
```

6. 提取子字符串(Substring)

内部函数“Substring”可以从某个字符串中提取一个子字符串。子字符串的长度，从何处截取都可以人为设定。

它的语法是：Substring (string, index, count)

其中的“string”为主字符串、“index”的值用于指定从主字符串中的第几个字符开始截取、“count”为子字符串的长度。例如：

```
Set Variable: "NewString" = Substring ( "Four horses", 6,5 )
```

上面这个脚本将从主字符串“Four horses”的第六个字符开始，向后截取五个字符做为子字符串。从主字符串中可以看到，六个字符是“ h”，从字符“h”向后截取五个字符得到子字符串“horse”。

7. 取得字符串长度（Length）

内部函数Length的功能是提取字符串的长度。它将包括字符串中的空格和标点符号。它的语法为：Length（String），其中String为需要被提取长度的字符串。

具体用法为：Set Variable: "Result" = length ("The four horsemen.")

其结果为：18

8. 根据ASCII值得到字符（Chr）

内部函数“Chr”可以根据ASCII值得到对应的字符。

它的句法为：Chr（ASCII Code），其中“ASCII Code”是字符对应的ASCII值。“ASCII Code”的值不应大于255。如果“ASCII Code”大于255，将用它与255相除的余数来代替。

具体用法为：Set Veritable: "Result" = Chr (ASCII Code)

其结果为：

如果“ASCII Code”为74　　字符为J

如果“ASCII Code”为115　　字符为s

如果“ASCII Code”为34　　字符为#

如果“ASCII Code”为323　　字符为C

9. 根据字符得到ASCII值（Ord）

内部函数“Ord”可以根据字符得到对应的ASCII值。

它的语法为：Ord（character），其中“character”为需要获得ASCII值的字符。如果“character”是字符串，只有字符串中的头一个字符的ASCII值做为结果返回。

具体用法为：Set Veritable: "Result" = Ord (character)

其结果为：

如果字符为“f”　　ASCII值为102

如果字符为“D”　　ASCII值为68

如果字符为“*”　　ASCII值为42

如果字符为“Dog”　　ASCII值为68

10. 多字节字符串函数

在Flash 4中有四个多字节字符串函数。它们的功能与一般的字符串函数一样，只不过它们是工作在多字节系统中。在多字节系统中的字符是由两个字节的ASCII值来描述，所以字符的ASCII值是从256到65535。

四个多字节字符串函数分别是：

MBSubstring
MBLength
MBChr
MBOrd

11. 功能调用（Call）

Flash 4中的功能调用与其他的编程语言的功能一样。当功能调用开始的时候，当前的进程暂时停止，转而去执行功能调用的进程。当功能调用的进程执行完毕，再回到原进程继续向下执行。在被调用的帧上，只执行该帧上的动作命令，不显示该帧上的画面。从这个角度来看，像Tell Target、Go To这样的动作命令，不管它们位于哪一帧都可以被调用执行。

在执行功能调用“Call”时，需要确定功能调用的有关程序的具体位置。在Flash 4中，就是确定功能调用有关脚本所在的帧。如果指定功能调用的有关脚本所在的帧不在当前时间线上，在调用时还需要写明所在帧的路径。

在Flash 4中功能调用的嵌套层数没有限制，但是在每帧上最多可以执行200000次动作命令将间接地限制它嵌套层数。下面的指令是要调用一个延时子程序。设定延时变量 Delay为10（延时子程序将延时10秒钟）。延时子程序位于标签为Up的帧中的Count动画片段图符中。它位于主时间线上。这段指令为：

```
Set Variable: "Delay"=10
Call ("/Count:Up")
```

图9-18所显示的是一个利用功能调用编写的演示实例，有感兴趣的读者可以访问 www.FlashBible.com/members/ActionScript/PlugNPlay/Trig.htm。

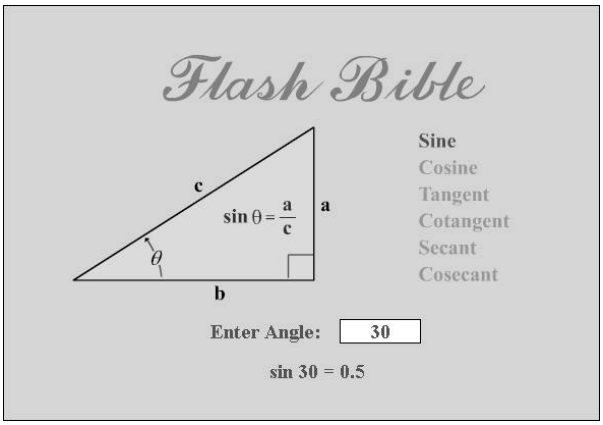

图9-18 利用功能调用编写的演示实例

12. 创建简单数组

Flash 4不提供创建数组的功能。但是在需要使用数组的时候，可以利用一组具有有序的变量名的变量来代替简单的数组。例如，Item1、Item2、....Item10。当然为这些变量赋值只能逐一进行。例如：

```
Set Variable: "Item1" = "Bear"
Set Variable: "Item2" = "Cat"
Set Variable: "Item3" = "Monkey"
```

可以看出，这是一个具有三行一列的简单数组。

13. 返回字符串表达式的值（Eval）

内部函数“Eval”可以获得字符串表达式的值。

它的语法为：Eval(String)，其中的“String”是一个字符串表达式。内部函数“Eval”返回的是一个字符串表达式最终结果。

利用内部函数“Eval”的这种特性，再加上在上一节所创建的简单数组，就可以在Flash 4的环境里进行一些简单的数组运算。下面就是应用的实例。

```
Set Variable: "count" = 1
Loop While (count <= 10)
   If (Eval ("Item" & count ) eq "Monkey" )
      Set Variable: "Display" = " There is a monkey!"
   End If
   Set Variable: "count" = count + 1
End Loop
```

请注意，在(Eval ("Item" & count)语句中，"Item" & count语句将"Item"和count组合成了变量名Item1、Item2、...Item10。然后再由Eval函数将每个变量的值取出。

本章我们进一步讨论了Flash 4的动作控制脚本语言。在下一章将讨论Flash 4与Macromedia公司其他产品的联合应用方法。