第11章 交 互 性

人类需要相互进行交流沟通。人们通过动作来激发反应。人们愿意扮小丑来博取孩子的一笑。人们喜欢遥控是因为喜欢按下按钮然后静观事态发生的那种感觉。沉默的观众、没有电池的遥控器、冷酷的计算机，所有这一切都会让人们发疯。如果人们无法与某件事物进行交互，便会另外寻找新的乐趣。

而这恰恰是我们不希望我们的观众所做的。虽然我们可以通过添加声音和动画来吸引观众的注意力，但是如果想真正抓住观众，唯一的办法就是允许他们进行操作。

交互性是电影和观众之间的纽带。可以用简单的方法来允许观众控制电影的放映和外观，或者可以一种更高级的方式来创建游戏、可自定义界面、窗体等等。但是，为了成功地实现交互，应了解一些高级交互背后的逻辑知识。不要害怕，你不必像一位计算机程序设计员那样设计复杂的程序，而只需在演示文稿中添加一些简单的交互作用。对交互作用的原理稍做了解可以受益非浅。

11.1 Flash中的交互作用

Flash中的交互作用由三个因素组成：触发动作的**事件**、事件所触发的**动作**以及**目标**或对象，也就是执行动作或事件所影响的主体。

想一想现实生活中的闹钟。上闹钟的过程实际就是一个设计交互作用的过程。潜在的逻辑关系是：

事件：当到达设定时间时，闹钟便发出声音 (将动作设定为一种运动)。

目标：闹钟(事件影响的对象)。

动作：闹钟发声(对象所执行的动作)。

如果你会设置闹钟，那么你一定也会成为一位设计交互的高手。

要用Flash创建交互，需使用 ActionScript语言。该语言包含一组简单的指令，用以定义事件、目标和动作。

11.1.1 事件

在演示文稿中添加交互时，需要定义的第一件事情就是事件。你可以两种方式来触发事件：一种是基于动作的，即通过单击鼠标、拨号或者敲打键盘开始一个事件；一种是基于时间的，即当到达一定的时间时自动激发事件。在 Flash中这两种方式称为鼠标 /键盘事件(由用户点击鼠标或敲打键盘激发)和帧事件(在播放过程中当时间线到达某一帧时激发)。

1. 鼠标事件(按钮动作)

当观众操作电影中的一个按钮时便发生鼠标事件。这种事件也被称为按钮动作，因为它们总涉及到一个按钮且总能触发一个动作。用户可以通过光标以下面任意一种方式来触发鼠标事件(见图11-1)：

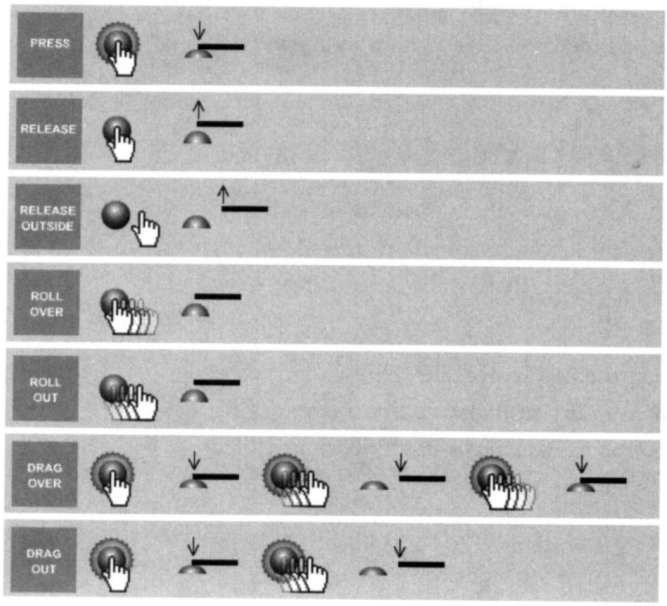

图11-1 向下箭头表示鼠标按钮按下；向上箭头表示鼠标释放。无箭头表示不要求鼠标按钮按下或释放的鼠标事件

- Press：当用户将光标移到电影按钮并按鼠标按键时，动作触发。
- Release：当用户将光标放在电影按钮并单击且释放鼠标按键时，动作触发。 (这是大多数动作的默认鼠标事件。)
- Release Outside：当用户按电影按钮，而在按钮外面释放鼠标时动作发生。
- Roll Over：当用户将光标放置在鼠标按键上时动作发生。
- Roll Out：当用户将光标从按键上移出时动作发生。
- Drag Over：当用户将光标放置在电影按键上的同时按住鼠标按钮，然后将光标从电影按钮上拖出(依然按住鼠标按钮)，最后再将光标放回电影按键时动作发生。
- Drag Out：当用户将光标放置在电影按钮后，按住鼠标按键，然后将光标从电影按钮上拖出(依然按住鼠标按钮)时动作发生。

按钮是电影中唯一受这些事件影响的对象。

要定义一个触发动作的鼠标事件，应如下操作：

1) 双击一个按钮，或者选择一个按钮，然后从 Modify菜单选择Instance。Instance Properties对话框出现。

2) 单击Action选项卡。

3) 单击“ + ”号分配一个对应于鼠标事件的动作。

4) 本例中选择Stop停止电影放映。

Action面板显示出完成的 ActionScript(见图11-2)。它表示当按钮释放时，动作发生。如不具体指定，该事件将是默认的鼠标事件。如果你不想选择此鼠标事件，或者想使用多个鼠标事件，也可以自己设置。

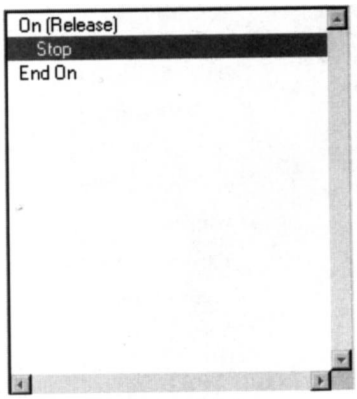

图11-2 停止电影放映的鼠标事件的ActionScript

5) 在Action面板选择On (Release)语句。

该语句突出显示，同时 Action对话框的右边出现鼠标事件参数 (参见图11-3)。

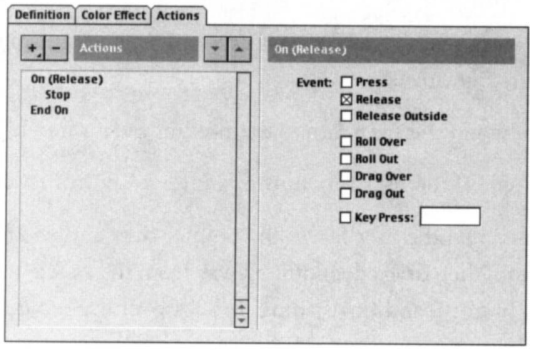

图11-3 鼠标事件参数

6) 选择任意鼠标事件来触发此动作。

选择不同的事件时， Parameter面板中的On ()语句便更新。

7) 单击OK。

当电影放映时，若已选的鼠标事件触发，则执行分配给此按钮的动作。

提示 分配给某按钮的一个实例的鼠标事件不影响该按钮的其它实例，即便这些按钮实例同时都位于舞台上也是这样。每个按钮实例都可分配不同的事件和动作。

提示 如果你愿意，可以在分配动作之前先分配鼠标事件。执行以上步骤 1和步骤 2，但在步骤 3选择On MouseEvent来定义鼠标事件。然后从同一菜单中选择某个动作(见图11-4)。

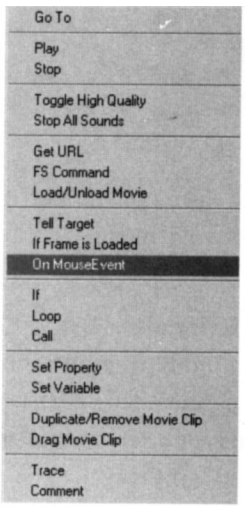

图11-4 设置某动作之前，先选择On MouseEvent选项分配鼠标事件

提示 许多按钮动作不能在编辑环境中测试。要完整地测试按钮，选择Control|Test Movie。

2. 键盘事件

当用户按字母、数字、标点、符号、箭头、回格键、插入键、Home键、End键、Page Up键、Page Down键时，键盘事件发生。键盘事件区分大小写，也就是说，A不等同于a。因此，如果你按A来触发一个动作，那么按a则不能。

键盘事件与按钮实例相连。虽然你不需要操作按钮实例，但是它必须存在于一个场景中才能使键盘事件起作用(虽然键盘事件不要求按钮可见或存在于舞台上)。它甚至可以位于帧的工作区以使它在电影导出时不可见(见图11-5)。

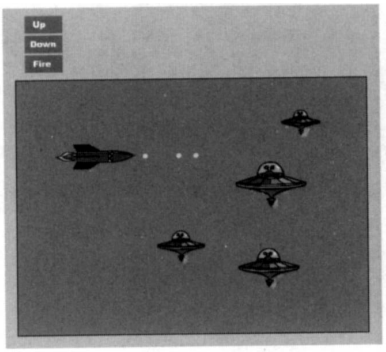

图11-5 某些包含键盘事件的隐藏按钮位于电影导出时不可见的区域。例如，舞台区之外

要定义一个触发动作的键盘事件，应如下操作：

1) 执行以上步骤1到5(添加一个鼠标事件)。

2) 在Action对话框的Parameter面板，选择键击事件。

3) 在键击选项旁边的小文本框中，键入触发动作的键名(见图11-6)。

4) 单击OK。

电影放映时，当用户按此键时，将执行分配给此按钮实例的动作。

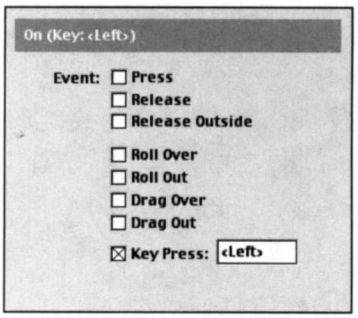

图11-6 按键盘键设置触发动作的键

3. 帧事件(帧动作)

与鼠标和键盘事件类似，时间线触发帧事件。因为帧事件与帧相连，并总是触发某个动作，所以也称帧动作。

帧事件总是设置在关键帧，可用于在某个时间点触发一个特定动作。例如， Stop动作停止电影放映，而Go To动作则使电影跳转到时间线上的另一帧或场景。

要创建一个触发动作的帧事件，应如下操作：

1) 双击你想设置帧事件的关键帧。

2) 单击Action选项卡。

3) 单击“ + ”号分配动作。当时间线到达此关键帧时，将触发此动作。

4) 本例中选择Stop动作。此动作将停止电影放映。

Action面板显示完整的ActionScript。请注意即便此动作与我们前面配置的鼠标事件相同，但是脚本也不一样。这里缺少On (Release)和End On语句。这两个语句只有在定义鼠标或键盘事件(因为这两种事件可有多种形式)时才需要。而帧事件只能以一种方式触发，也就是当时间线到达该帧时触发。

5) 单击OK。

放映电影时，当时间线到达此关键帧时，将执行相应的动作。

11.1.2 目标

现在你已知道如何用事件来触发动作。接下来需要了解如何指定将受所发生事件影响的对象或目标。事件控制 3个主要目标：当前电影及其时间线、其它电影及其时间线 (例如电影剪辑实例)和外部应用程序(例如浏览器)。以下ActionScript范例显示如何用这些目标创建交互。详细讲解如下：

在以下脚本中，当前电影(目标)中的Roll Over(事件)将使电影的时间线停止放映(动作)。

```
On (Roll Over)
  Stop
End On
```

在以下范例中，当前电影的Roll Over(事件)使得另一电影，即电影剪辑实例MyMovieClip(目标)的时间线停止放映(动作)。

```
On (Roll Over)
  Begin Tell Target ("/MyMovieClip")
    Stop
  End Tell Target
```

```
End On
```

以下ActionScript打开用户的默认浏览器 (目标)，并在Roll Over(事件)触发时加载指定的URL(动作)。

```
On (Roll Over)
  Get URL ("http://www.crazyraven.com")
End On
```

提示 有关ActionScripting句法的详细信息，请参见本章后面11.3节“ ActionScripting ”。

1. 当前电影(默认目标)

当前电影是一个**相对目标**，也就是说它包含触发某个动作的按钮或帧。因此，如果将某个鼠标事件分配给某个按钮，而该事件影响包含此按钮的电影或时间线，那么目标便是当前电影(见图11-7)。但是，如果将某个鼠标事件分配给某个按钮，而该按钮所影响的电影并不包含该按钮本身，那么目标便是一个**传达目标**(Tell Target)。

对于帧动作也是如此。除非指定传达目标为目标，否则大多数情况下， ActionScripts默认将当前电影作为目标。如下例所示：

```
On (Roll Over)
  Go To and Stop (Scene 5,20)
End On
```

以上ActionScript表示鼠标事件触发此动作。当光标滚过 (事件)电影中的按钮时，当前电影的时间线(目标)将跳转到场景5，帧20，然后从此处开始放映。

如果将以上ActionScript与主电影中的一个按钮相连，并且不用传达目标来引用另一电影，那么主电影便被视为当前电影。但是，如果将此 ActionScript与电影剪辑图符中的一个按钮相连，并且不用传达目标来引用另一电影，那么此电影剪辑便是当前电影 (见图11-8)。只需记住：当前电影(也就是触发事件开始的地方)在任何ActionScript中都是相对目标。

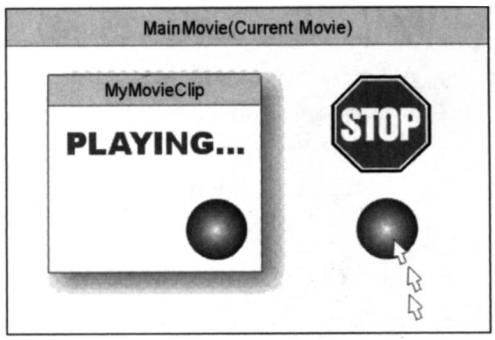

图11-7 当一个事件(例如鼠标事件)引起它自身的时间线进行某些操作(如停止放映)，那么该事件的目标被视为当前电影

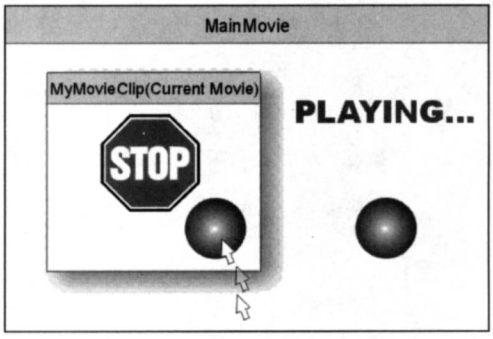

图11-8 上下文决定当前电影(根据事件触发的位置)

2. 其它电影(传达目标)

传达目标是由另一电影中的事件控制的电影。因此，如果你将一个鼠标事件分配给一个电影剪辑按钮，以便影响不包含此按钮的电影剪辑或时间线，那么你的目标便是一个传达目标(见图11-9)。以下AcitonScript用于控制一个传达目标，可将它与前例中用于控制当前电影的AcitonScript进行比较。

```
On (Roll Over)
  Begin Tell Target ("/MyMovieClip")
    Go To Stop (Scene 5, 20)
  End Tell Target
End On
```

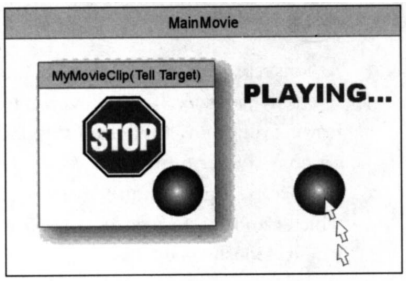

图11-9 当一个事件(如鼠标事件)引起另一动画时间线产生动作(如停止)，则该事件的目标称为传达目标

这里的ActionScript表示一个鼠标事件触发此动作。当光标Roll Over(事件)电影中的按钮时，另一电影的时间线(电影剪辑MyMovieClip)(目标)将跳到场景5，帧20，然后从此处开始放映(动作)。

如果你觉得通过一部电影控制另一电影这一概念容易混乱，那么可暂时将它搁在一边。我们将在整章中继续讨论这些概念。有关的详细信息，请参见本章后面 11.2节“处理多部电影”或者参考光盘中的交互式教程。

提示 只能用传达目标控制电影剪辑实例或者在 Flash Player窗口中用Load/Unload Movie动作加载的电影，而对于同一HTML页中两个具有单独的〈object〉或〈embed〉标记的电影，则不能用传达目标在它们之间进行通信。

2. 外部应用程序(例如浏览器或投影程序)

外部目标位于电影区域之外，例如，对于 Get URL动作，你需要一个Web浏览器才能实际打开指定的 URL。有三个动作可以引用外部源： Get URL、FS Command和Load/Unload Movie。这三个动作都需要外部应用程序的帮助。这些动作的目标可以是 Web浏览器、Flash投影程序、Web服务器或其它应用程序 (有关的详细信息，请参见 11.1.3节“动作”)。以下ActionScript以Flash投影程序窗口为目标：

```
On (Roll Over)
  FS Command ("fullscreen", "true")
End On
```

这里的ActionScript表示鼠标事件触发此动作。当光标 Roll Over(事件)电影中的按钮时，投影程序窗口将变为全屏。投影程序窗口被视为应用程序窗口，因此，是一个外部目标。

11.1.3 动作

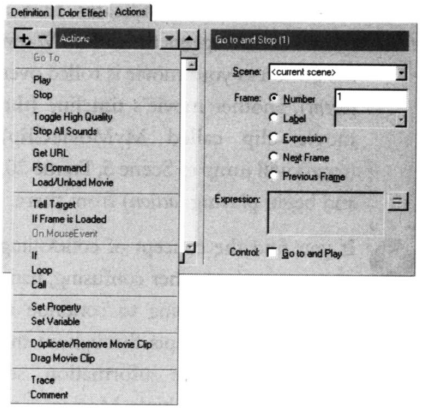

图11-10 单击Add Action按钮将出现一列可用动作

动作是组成交互作用的最后一个部分。它们引导电影 (或外部应用程序)执行任务。一个事件可以触发多个动作，且这多个动作可以在不同的目标上同时执行。

我们将先简单了解Flash中的动作，然后通过一些具体应用程序对它们进行深入分析。单击Aciton选项卡中的Add Aciton按钮，将出现如表11-1中的选项(见图11-10)。

表11-1 动作

动 作	描 述
Go To	使电影跳到时间线上指定的帧或场景，然后从此处停止或开始放映
Play	使电影从时间线上的当前点开始放映
Stop	使电影停止放映
Toggle High Quality	打开和关闭消锯齿功能
Stop All Sounds	停止当前播放的所有声音
Get URL	打开一个加载了指定URL的浏览器窗口或者将变量发送给指定的URL
FS Command	将数据发送给服务于Flash电影的应用程序(例如浏览器、投影程序、Director电影)
Load/Unload Movie	在Flash放映程序中，将Flash电影加载于指定的URL，或者卸载以前加载的电影。你还可以用它将远程文件中的变量加载于一部电影
Tell Target	一种用作电影识别方式的命令，可以用它执行某个动作
If Frame Is Loaded	确定一部特定电影是否已经加载，如果是，则执行某一动作
On Mouse Event	鼠标事件实际上是触发器而不是动作。有关的详细信息，请参见本章前面11.1.1节“事件”
If	检查某个条件语句是否为真，如果是，则执行某个动作
Loop	只要条件满足便连续执行某个动作或某组动作。当条件不再满足时，循环结束
Call	执行一组与特定帧相关的动作
Set Property	设置不同的电影属性
Set Variable	创建一个新的变量或者更新一个已有变量
Duplicate/Remove Movie Clip	动态地创建或删除电影剪辑实例
Drag Movie Clip	使电影剪辑实例可相对于光标的运动拖动
Trace	显示动作执行时的自定义信息。主要用于测试交互性。详细信息，请参见第12章“测试”
Comment	在电影中加注释，以帮助理解复杂的逻辑关系

11.1.4 高级动作

要使用更高级的Flash功能，需要加深对Flash动作的了解，而不能仅停留在一知半解的水平。在这里，我们将功能详细展示你或观众可使用的控制演示文稿的方法。

我们以下将经常用到**表达式**和**求值**这两个术语。现在做一下简单介绍。 Flash中的大多数动作都带有参数，必须正确设置这些参数才能保证动作按你的设想发生。例如，在将学到的Go To动作中有一个参数，此参数要求标识一个将转到的帧。可以用一个帧编号，也可以用一个标记。可以输入一个准确的值，如 25，也可以根据表达式所求出的值来设置参数。你也许会问，什么时候输入值，什么时候输入表达式呢？我们说，在要求你指定要转到的帧编号，而不仅仅是输入某一个值的方框中，可以输入表达式。例如 18+7就是一个表达式。表达式就

是等于某一特定值的运算符或短语，在 18+7这个表达式中特定值就是 25。使用表达式来设置参数的值，其效果同输入某一特定的值相同。表达式可以有多种方式，并且可以等于多种类型的值，包括数字和文本。使用表达式的作用可能不是很明显，但是它会使你的电影具有动态效果。有关的详细信息，请参见本章后面 11.3.2节“ 使用表达式 ”。

提示 大多数动作的效果不能在Flash编辑环境中查看或测试。要测试交互性，从菜单栏中选择Control|Test Movie。

提示 在下面的内容中，时间线、电影和电影剪辑可以互换。

11.1.5 Go To

Go To(跳转)动作用来控制电影时间线的位置，使它跳转到一个特定的帧编号、帧标记或场景，并从该处停止或开始放映 (根据具体设置而定)。

1. 参数

Go To动作可使用以下参数(见图11-11)：

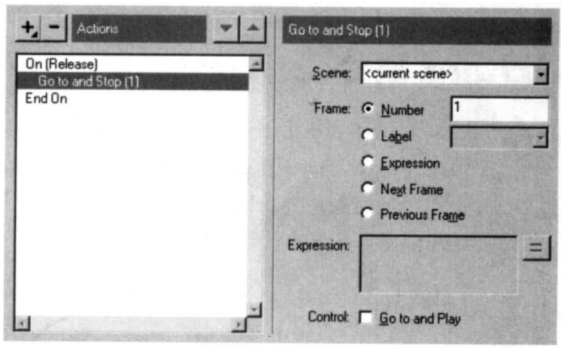

图11-11 Go To动作可使用的参数

Scene(场景)：可选择一个场景作为Go To动作的起点。一旦已定义一个场景，你就可以在该场景中定义一个帧编号或标记。该场景参数不能用于图符。可用选项包括：

- <current scene>(当前场景)：允许你从当前场景选择一个帧编号或标记 (参见下面的“ 帧 ”)作为将转到的时间线上的点。
- <next scene>(下一场景)：引起动作转到下一场景的第 1帧。如果选择此选项，便不能跳转到该场景中特定的帧编号或标记。要实现这一点，应使用 Scene_Name。
- <previous scene>(上一场景)：引起动作转到上一场景的第 1帧。如果选择此选项，你便不能转到该场景中特定的帧编号或标记。要实现这一点，应使用 Scene_Name。
- Scene_Name：在此出现一列场景名；从出现的列表中进行选择。

Frame(帧)：允许你在选择了场景的基础上，再选择将转到的特定帧。

- Number(编号)：选择将转到的帧编号。
- Label(标记)：选择该场景中要转到的帧标记。 (出现的列表仅提供当前场景中可用的帧标记。)
- Expression(表达式)：允许你根据表达式的值动态地设置目的帧。要了解详细信息请参见本章后面11.3.2节“ 使用表达式 ”。

- Next Frame(下一帧)：使时间线跳转到触发动作帧的下一帧。此选项只有在选择了<current scene>之后才可用。
- Previous Frame(上一帧)：使时间线跳转到触发动作帧的上一帧。此选项只有在选择了<current scene>之后才可用。

Expression(表达式)：此文本框允许你键入一个表达式，以便根据表达式的值动态地设置将跳转到的帧。单击等号(“=”)按钮将激活Expression Editor。有关的详细信息，请参见本章后面11.3.2节“使用表达式”。

Control(控制)：此选项允许你指定在时间线跳转到特定帧后，电影是在该帧停止放映还是从该帧继续放映。如果不选此框，电影将根据默认停止放映。

2. 脚本范例

以下脚本展示了一个鼠标事件，它使当前电影跳转到场景 Intro的第10帧，然后开始放映：

```
On (Release)
  Go To and Play (Intro, 10)
End On
```

以下脚本展示了当前电影中的一个鼠标事件，它使得另一个名为 Brooks的电影剪辑跳转到标记为Sleep的帧，然后停止放映：

```
On (Release)
  Begin Tell Target ("Brooks")
    Go To and Stop ("Sleep")
  End Tell Target
End On
```

3. 实际应用

可以用这种方法在Flash电影中创建超链接。

11.1.6 Play

Play(放映)动作使得电影从它的当前位置开始放映。如果电影由于 Stop动作或Go To and Stop动作而停止，那么你只能使用 Play动作启动才能使电影继续放映。

1. 参数

Play动作没有参数。

2. 脚本范例

以下脚本展示了一个鼠标事件，它使当前电影从当前位置开始放映。

```
On (Press)
  Play
End On
```

以下脚本展示了当前电影中的一个帧事件，它使得另一电影，即名为 Brooks的电影剪辑开始放映(如果它还未开始放映)：

```
Begin Tell Target ("/Brooks")
    Play
End Tell Target
```

3. 实际应用

可用来创建开始/停止按钮，即On (Press)Play、On (Release) Stop。

11.1.7 Stop

Stop(停止)动作使得电影停止放映。对于一部你希望延长显示时间的电影，可以在其中任意处使用此动作。

1. 参数

Stop动作没有参数。

2. 脚本范例

以下脚本展示了一个鼠标时间，它使得当前电影停止放映：

```
On (Press)
  Stop
End On
```

以下脚本展示了当前电影中的帧事件，它使得另一个名为 Brooks的电影剪辑停止放映(如果它还未停止放映)：

```
Begin Tell Target ("/Brooks")
  Stop
End Tell Target
```

3. 实际应用

可用它来创建开始/停止按钮。将此动作放置在应停止的电影的任意帧。

11.1.8 Toggle High Quality

Toggle High Quality(切换高质量)动作切换消锯齿功能(见图11-12)，它影响视觉效果和重放速度。打开消锯齿功能，视觉质量提高，而较早计算机上的重放速度减慢。关闭消锯齿功能，则出现相反的效果。此动作不能影响单个目标。它影响当前在 Flash Player中放映的所有电影和电影剪辑。

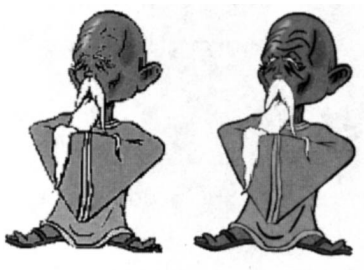

图11-12 Toggle High Quality功能可以影响电影中图形的光滑程度

1. 参数

此动作没有参数。如果已经打开消锯齿功能，设置此动作则将它关闭，否则，便打开此功能。

2. 脚本范例

以下脚本展示了一个鼠标事件，它切换当前在 Flash Player中放映的所有电影的消锯齿功能：

```
On (Release)
  Toggle High Quality
End On
```

3. 实际应用

用于确定视觉质量，并关闭电影中动画比较集中的部分的消锯齿功能。

11.1.9 Stop All Sounds

Stop All Sounds(停止所有音轨)动作停止当前在Flash Player中播放的所有声音。此动作不影响电影的视觉效果。

1. 参数

此动作没有参数。

2. 脚本范例

以下脚本展示了一个鼠标事件，它停止当前在 Flash Player中放映的所有电影和电影剪辑中的声音。

```
On (Release)
  Stop All Sounds
End On
```

3. 实际应用

用于关闭声音(声音打开/关闭按钮)，并使音轨静音。

11.1.10 Get URL

Get URL(设置URL)动作从事以下两项工作：将指定的 URL加载到浏览器窗口，或者将变量数据发送给指定的URL。例如，变量数据可以发送给 CGI脚本，以便按照HTML窗体的处理方式进行处理。请注意只发送“ 当前 ”电影的变量 (请参见本章后面11.2节“ 处理多部电影 ”)。

虽然Get URL动作主要用于将 Flash电影放置在Web页上，但也可以将它用在 Flash运行程序中，以便自动打开浏览器窗口并显示指定的 URL。

1. 参数

Get URL具有以下参数(见图11-13)：

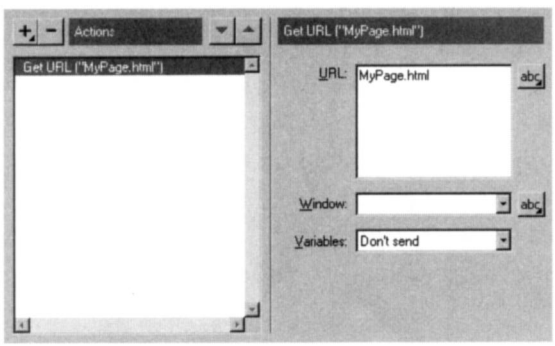

图11-13 Get URL动作的参数

URL：你可在此处定义Get URL动作的URL。它可以是一个相对路径，例如 mypage.html，也可以是一个绝对路径，例如 http://www.mydomain.com/mypage.html。

如果你获得的URL是一个CGI脚本或Cold Fusion模板，那么它应分别为 http://www.mydomain.com/cgi-bin/myscript.cgi或者http://www.mydomain.com/mycftemplate.cfm。

如果你的Flash电影位于HTML页，可以用它来定义一个当事件触发时将调用的 JavaScript函数。

可以用表达式来动态地设置 URL。单击“ abc ”按钮激活Expression Editor。详细信息，请参见本章后面11.3.2节“ 使用表达式 ”。

Window(窗口)：指定加载并显示指定的 URL的浏览器窗口或HTML帧。如果你已为一个HTML窗口或者帧定义了名称，并希望指定的 URL加载到该窗口，只需在此框中键入该窗口名。或者，你也可以选择以下选项：

- self ：将指定的URL加载到当前Flash电影所在的窗口或帧。
- blank：打开一个新的浏览器窗口，并将指定的 URL加载到此窗口。
- parent：将URL打开在当前帧的父帧中。
- top：如果具有Get URL动作的Flash电影在一个HTML帧中，那么此选项将删除该帧并将URL加载到浏览器窗口。

可以通过表达式来动态地设置加载 URL的窗口或帧。单击“ abc ”按钮激活 Expression Editor。要了解详细信息，请参见本章后面 11.3.2节“ 使用表达式 ”。

Variables(变量)：选择当使用Get URL动作时如何处理当前电影中的变量。有以下几种选择：

- Don ’t send(不发送)：不发送变量。是只打开URL的最佳方法。
- Send using GET(用GET发送)：发送附在指定的URL后面的变量。例如，如果传递两个变量，即姓名和年龄，那么使用 GET方法将使URL显示为：http:// www. mydomain. com/mypage.html?name=Derek+Franklin&age=unkown。因此，如果Flash电影存在于mypage.html，应将姓名和年龄的值传递给它，且它应能以自己的方式响应这些值。换句话说，你可以用此选项将位于HTML页上的Flash电影中的变量值发送给另一HTML页上的Flash电影(这只能很好地用于少量变量的发送)。
- Send using POST(用POST发送)：发送与URL隔开的变量，可以发送多个变量。在常规的HTML页上，这种方法主要用于将从一个窗体收集来的信息投递到服务器上的 CGI脚本。它还可以同样的方法将变量值发送给 CGI脚本进行处理。

提示 发送变量时，只有当前电影中的变量才能发送(而不是Flash电影窗口中展现的所有电影中的变量)。

提示 有关变量的详细信息，请参见本章后面11.3节“ ActionScripting ”。

2. 脚本范例

以下脚本展示了一个鼠标事件，它打开新窗口中的 URL。

```
On (Release)
  Get URL ("http://www.mydomain.com/mypage.htmlc", window="_blank")
End On
```

以下脚本展示了一个鼠标事件，它将变量投递到服务器上的 CGI脚本。

```
On (Release)
  Get URL ("http://www.mydomain.com/CGI-bin/myscript.cgi", vars=POST)
End On
```

3. 实际应用

用于Flash窗体以及与Flash内容相连的HTML网页。

11.1.11 FS Command

FS Command(FS命令)允许你的Flash电影与其它程序通信，例如Web浏览器或者可用于你的Flash电影的任何程序。主程序只是一个允许你在其中嵌入 Flash电影的程序。此命令通常用于使Flash可与HTML页中的JavaScript交互。

以下展示了如何用FS Command打开自定义警告框，请放心，这无须太多的技术知识：

1) 创建一部带按钮的Flash电影，该按钮包括一个触发FS Command动作的鼠标事件。

2) 当设置FS Command时，在Command(命令)框中键入InfoBox，在Argument(参数)框中键入We ’ re Doing OK(见图11-14)。

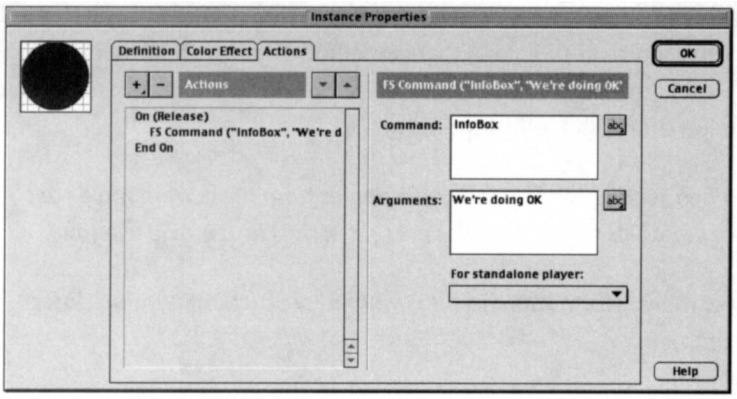

图11-14 你可以根据FS Command命令将执行的动作，在Command和Argument文本框中键入相应的内容

命令名可以任意设置，但必须唯一。

3) 如果你愿意，还可以再创建一个带FS Command命令的按钮。对于第二个按钮，只需在Command框中键入InfoBox，而在Argument框中键入We ’ re not so good。现在你具有两个命令相同而参数不同的按钮。

当你将电影放置在HTML页时，还可以将它放置在一个JavaScript函数中，当电影中的FS Command激活时，该函数可进行检测。在本例中，我们特意设置此JavaScript函数以便以某种方法评估FS Command命令，例如“ 如果命令等于InfoBox，则创建一个警告框以读取FS Command的参数 ”。因此，当单击某一按钮时，将打开警告框“ We ’ re doing OK ”，或“ We ’ re not so good ”。

当然，你可以用FS Command来完成更多任务，但这通常涉及到更多的JavaScript。对于大多数用户，Flash的新增内部脚本(ActionSciript)已足以应付大多数任务。

11.1.12 Parameters

此动作具有以下参数：

Command(命令)：它是分配给FS Command的唯一名称。可以任意设置。

你可以使用表达式来动态地设置命令。单击等号“ = ”按钮激活Expression Editor。要了解详细信息，请参见本章11.3.2节“ 使用表达式 ”。

Argument(参数)：如果该命令要求为JavaScript函数传递某一信息，必须在此输入这一信息。它可以是文本字符串的形式，如“ Hey Bob ”，也可以是数值，如35。

你可以使用表达式来动态地设置参数。单击等号“ = ”按钮激活Expression Editor。要了解详细信息，请参见本章11.3.2节“ 使用表达式 ”。

For Stand-Alone Player(用于单机播放程序)：当将电影作为单机应用程序来发放时，可以使用以下设置来控制Flash执行程序：

- fullscreen(全屏)：要全屏显示一部执行程序，选择“ true ”。“ False ”则以 Movie Properties对话框中设置的大小显示执行程序。
- allowscale(允许缩放)：选择“ true ”将允许用户调整执行程序窗口的大小。“ False ”则防止改变窗口的大小。
- showmenu(显示菜单)：若选择“ true ”，则当用户右击(Windows)或者按住Control键单击(Macintosh)执行程序窗口时，将出现执行程序菜单。“ False ”则不会使此菜单出现。
- quit(退出)：退出执行程序，并关闭其窗口。
- exec(执行)：使用此命令在Flash中打开一个外部程序。在 Argument框中输入该应用程序的目录路径。

1. 脚本范例

以下脚本展示了一个帧事件，它将执行程序窗口打开至全屏显示：

```
FS Command ("fullscreen", true)
```

2. 实际应用

控制执行程序，并用JavaScript与执行程序进行交互。

11.1.13 Load/Unload Movie

Load/Unload Movie(加载/不加载电影)动作允许你进行以下几项工作：

- 将一部新的电影加载到 Flash电影以替换原有的电影，也就是说，无须加载不同的 HTML页就可以显示一部新的电影。
- 在原有电影的基础上，加载一部新的电影(见图11-15)。
- 将已加载电影的变量发送给 CGI脚本进行处理。
- 卸载以前用Load Movie动作加载到电影窗口的电影。
- 将一组变量加载到时间线，以便时间线可以根据这些变量的结果进行相应的动作。

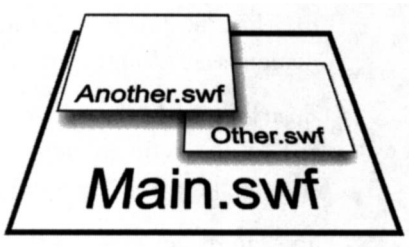

图11-15 使用Load Movie动作加载电影，可以一次在Flash Player窗口中展示多个.swf文件

因为Load/Unload Movie菜单项/动作实际上是一组动作，所以我们不使用脚本范例，而直接向你展示如何执行各项任务。

1. Parameters

此动作具有以下参数(见图11-16)：

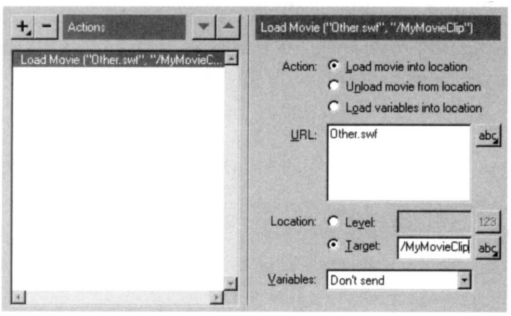

图11-16 Load/Unload Movie动作可用的参数

Action(动作)：Load/Unload Movie动作/菜单项执行三个主要动作，它们由以下按钮确定：

- Load movie into location(加载电影至某一位置)：将一个新的.swf文件加载到指定的级层或目标。这使得 Flash Player窗口一次可显示多个 .swf文件。此动作与 Variable参数的GET或POST设置结合使用，可以将当前或目标时间线的变量发送给在 URL参数中指定的CGI脚本。
- Unload movie from location(卸载某一位置的电影)：卸载以前用Load Movie动作加载的电影。
- Load variables into location(加载变量至某一位置)：允许你将一组变量从远程文本文件加载到电影，该远程文本文件是你创建并放置在服务器上的文件或者是由 CGI脚本产生的文件(请参见下面的文字框“补充内容”)。将此动作与Variables参数的GET或POST设置结合使用，可以将在该位置参数中指定的电影的变量发送给在 URL参数中指定的CGI脚本，然后该脚本根据服务器的响应，将新的变量值加载回原来的电影。

补充内容

加载远程文本文件的变量要求文本文件中的信息使用一种特殊格式，以便 Flash可读。这种格式称为URLformencoded，可手写或通过CGI脚本、Cold Fusion、ASP或者其它服务器脚本编辑语言输出。例如：

```
name=John+Doe&age=25&favoritefood=Pizza&married=false
```

name、age、favorite和married均表示变量名，应与文本文件将加载到的电影的变量名相一致。加号表示变量值中的空格。因此， John+Doe加载到电影中后将变为John Doe。不同的变量或变量值用“ & ”隔开。因此，上例中有4个单独的变量。

URL：当用于“ Load movie into location ”时，它是要加载的.swf文件的目录路径。它可以是一个相对路径，如mymovie.swf，也可以是一个绝对路径，如http:// www. mydomain.com /mymovie.swf。当将此选项与Variables参数的GET或POST设置结合使用时，此参数指定当前或目标时间线的变量将发送到的CGI脚本的位置。当选择“ Unload movie from location ”时，此选项将变灰。

当用于“ Load variables into location ”时，它是要加载的.txt文件的目录路径。它可以是一个相对路径，如 myvariables.txt，也可以是一个绝对路径，如 http://www. mydomain. com/myvariables.txt。当使用Variables参数的GET或POST设置时，此参数指定已加载的电影的变量将发送到的CGI脚本的位置。

你可以使用表达式来动态地设置URL。单击“ abc ”按钮激活Expression Editor。

Location(位置)：此参数定义将受指定动作影响的级层或目标。

- Level(级层)：除非加载到某个目标(它用整部电影[.swf文件]替换当前电影中的电影剪辑实例)，否则，加载的电影将放置在某个级层并分配一个级层编号。级层可看作为堆叠在Flash Player窗口中的各个.swf文件的图层(见图11-17)。分配给各个级层的编号决定它与其它所有级层的相对位置。最底部的 .swf文件的级层为0，它通常表示原始电影。电影可加载到已包含另一电影的级层。这样便仅替换该级层上已有的 .swf文件。
- Target(目标)：允许你将整个.swf加载到当前由一个电影剪辑实例占据的空间。这样，加载的.swf文件将继承该电影剪辑当前的所有属性，包括名称、目标路径、大小和位置 (见图11-18)。

图11-17 将电影加载到一个级层，其内容将在该级层下面的所有级层之上

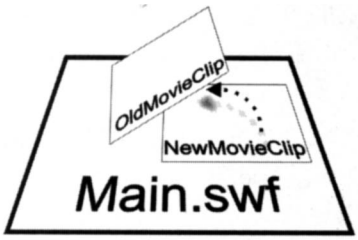

图11-18 将电影加载到某个目标可允许.swf文件替换原来由电影剪辑占据的空间

你可以通过求取表达式的值来动态地设置电影将加载到的级层或目标名称。单击等号(“=”)按钮激活Expression Editor。

Variables(变量)：允许你选择如何将电影中的变量发送给服务器。可用以下选项：

- Don’t send(不发送)：如果选择了Load Movie或Load Variables动作，则不会发送电影中的变量。
- Send using GET(用GET发送)：发送附在指定的URL后面的变量。要了解详细信息，请参见11.1.10节Get URL。
- Send using POST(用POST发送)：发送与URL分开的变量，这意味着可以发送大量变量。要了解详细信息，请参见11.1.10节“Get URL”。

要将一部新的电影加载到Flash电影窗口以替换原有的电影，应如下操作：

1) 选择“Load movie into location”选项。

2) 对于URL参数，输入要加载的.swf文件的目录路径。

3) 对于Location参数，输入当前由另一个电影占据的级层编号，或者选择要替换的电影剪辑目标。

4) 在Variables参数中选择“Don’t send”选项。

要在原有电影的基础上加载一部新的电影，应如下操作：

1) 选择“Load movie into location”选项。

2) 对于URL参数，输入要加载的.swf文件的目录路径。

3) 对于Location参数，输入当前未由另一电影占据的级层编号。

当将电影加载到级层时，你所选的级层编号并不要求连续。可以将一部电影加载到级层 6，而将下一部电影加载到87。

4) 在Variables参数选择“Don’t send”选项。

要将已加载电影的变量发送给CGI脚本进行处理，应如下操作：

1) 选择“Load movie into location”选项。

2) 对于URL参数，输入想将变量发送给它进行处理的 CGI脚本的目录路径。

3) 对于Location参数，输入你想发送其变量的电影的级层编号或目标路径。

4) 在Variables参数中，选择“Send using GET”或“Send using POST”选项。

要卸载以前用Load Movie动作加载到电影窗口的电影，应如下操作：

1) 选择“Unload movie from location”选项。

URL参数变灰。

2) 对于Location参数，输入要卸载的电影的级层编号或目标路径。

Variables参数变灰。

要将一组变量加载到时间线，以便时间线可以根据这些变量的结果采取相应的动作，应如下操作：

1) 选择“ Load variables into location ”选项。

2) 对于URL参数，输入要加载的变量所在的文本文件的目录路径。

如果文本将从CGI脚本产生，则输入脚本路径。

3) 对于Location参数，输入将接收新变量的电影的级层编号或目标路径。

4) 在Variables参数中，如果只想接收远程文件或CGI脚本的变量，则选择“ Don ’ t send ”。

如果选择“ Send using GET ”或“ Send using POST ”选项，则被指定电影的当前变量会被发送至服务器(到CGI脚本)进行处理。然后，基于服务器响应的新变量将被加载回该电影。

2. 实际应用

在不加载附加的HTML页的情况下查看多个Flash电影，以及Flash和基于服务器的信息和处理之间的连通性使得处理基于 Flash的格式成为可能，或者可以很容易在 Flash中显示动态产生的内容。

11.1.14 Tell Target

Tell Target(传达目标)命令将动作引向除当前时间线之外的任意时间线。可以用 Tell Target命令(它总是与动作结合起来使用)控制除当前电影之外的电影、设置或改变另一时间线上的变量或者设置特定电影剪辑实例的某一个属性。

1. 参数

此命令只有一个参数。

Target(目标)：在此定义所有后续动作所在的电影。出现一个可用的电影剪辑实例目标的目录(见图11-19)。电影剪辑实例名称旁边的加号“ + ”表示它包含一个“ 子电影剪辑 ”，即该电影剪辑内部的一个电影剪辑 (请参见本章11.2节“ 处理多部电影 ”)。单击加号展开目录。如果你希望某一目标作为目标输入，则双击它的名称。

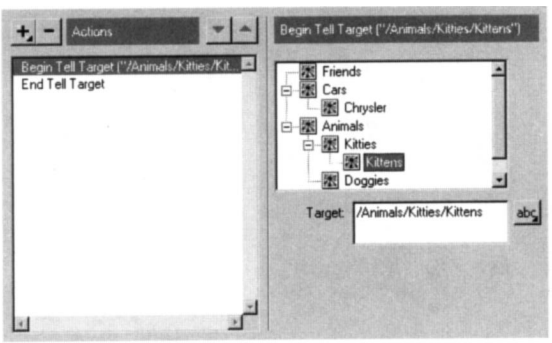

图11-19 使用电影剪辑实例的目录可轻易地将电影剪辑设置为目标，只需双击电影剪辑的名称就可选择它

提示 在图11-19的对话框中只出现当前电影的子电影剪辑实例。如果你希望以一个父电影剪辑或另一个子电影剪辑为目标，则必须输入其路径。要了解有关的详细信息，

请参见本章后面11.2节“ 处理多部电影 ”。

可以根据表达式所求出的值来动态地选择目标。单击“ abc ”按钮激活Expression Editor。

2. 脚本范例

以下脚本展示了一个鼠标事件，它使得电影剪辑实例 (MyMovieClip)的时间线跳转至帧标记MyFrameLabel，然后停止：

```
On (Release)
  Begin Tell Target ("/MyMovieClip")
    Go to and Stop ("MyFrameLabel")
  End Tell Target
End On
```

以下脚本演示了两种使用目标来执行某一动作的方法。上面的脚本用 Tell Target命令来定义目标；下面的脚本用Set Property命令来定义目标。两个脚本均执行相同的动作：

```
On (Release)
  Begin Tell Target ("/MyMovieClip")
     Set Property (" ", X Scale) = "50"
  End Tell Target
End On
```

第二个脚本：

```
On (Release)
  Set Property ("/MyMovieClip", X Scale) = "50"
End On
```

3. 实际应用

通过一部电影来控制另一电影。

11.1.15 If Frame Is Loaded

If Frame Is Loaded(如果某帧加载)是另一个用于引导动作的命令 (并总是与动作结合起来使用)。其逻辑关系应为：如果帧 x加载，则进行这些动作。如果帧 x未加载，则忽略“ If Frame Is Loaded ”命令。这被称为条件语句，即只有满足某条件的情况下，才执行该动作。

此命令通常用于创建一个循环，以连续检查是否已完全下载一部电影。此命令也被称为加载检测器。

1. 参数

此动作具有以下参数(见图11-20)：

Scene(场景)：为If Frame Is Loaded命令选择一个场景作为起点。一旦定义了一个场景，就可以定义该场景内的帧编号或标记。此场景参数不可用于图符。可用的选项包括：

- <current scene>：从当前场景选择一个帧编号或标记作为要检查的帧 (请参见下面的帧参数)。
- Scene_Name：从出现的场景名称列表中进行选择。

Frame(帧)：此参数可以根据所选的场景选项确定一个特定场景是否已经加载。

- Number(编号)：选择一个帧编号。
- Label(标记)：选择场景中的一个帧标记。(所出现的列表仅提供当前场景中可用的帧标记。)

Expression(表达式)：此参数可以键入一个表达式，以便根据表达式所求出的值，动态地

设置要检查加载状态的帧。单击等号“ = ”激活Expression Editor。要了解详细信息，请参见本章11.3.2节“ 使用表达式 ”。

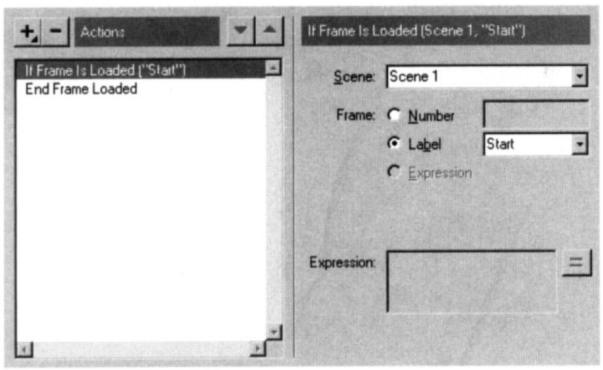

图11-20 “ If Frame Is Loaded ”动作可用的参数

2. 脚本范例

以下脚本展示了如何设置加载检测器。电影的第 1帧上的帧事件检查场景 MainScene(此场景包含电影的主动画)中标记为Start帧是否已经加载。如果是，则执行 Go To动作，从而使得时间线从该标记往前放映。如果还未加载，则忽略此命令，时间线继续第 2帧：

```
If Frame Is Loaded (MainScene, "Start")
   Go To and Play (MainScene, "Start")
End Frame Loaded
```

以下脚本展示了第2帧上的一个帧事件，它仅仅将时间线倒回至电影的第 1帧：

```
Go To and Play (1)
```

它触发第一个脚本的“ If Frame Is Loaded ”命令。如果Start帧还未加载，则再次忽略此命令，时间线继续，并再次触发第 2帧上的Go To and Play动作，从而重复整个过程。这就是所谓的循环。直到Start帧加载，也就是第一个脚本中所展示的第 1帧中的Go To and Play动作触发，此循环才停止。

提示 创建此功能的一种更加动态的方法将在If 命令的脚本范例中演示。

3. 实际应用

用于创建加载检测器，或者防止某些动作在必要的内容下载完之前触发。

11.1.16 On Mouse Event

请参见本章前面11.1节“ 事件 ”中“ 鼠标事件 ”小节。

11.1.17 If

使用If命令创建条件语句：如果某个条件满足，则执行某个动作；如该条件未满足，则执行另一个动作。如以下的条件语句：

```
If (Outside = "rain")
  Go To and Stop ("Bed")
Else
```

```
    Go To and Play ("Park")
End If
```

此条件语句表示，如果Outside等于rain，则执行go to bed；否则，执行go to the park。这可能不是最佳逻辑。如果是snowing或者如果有tornado该怎么办呢？很简单，我们只需再添加几个条件，代码如下所示：

```
If (Outside = "rain")
  Go To and Stop ("Bed")
Else If (Outside = "Sun")
  Go To and Play ("Park")
Else If (Outside = "Snow")
  Go To and Play ("SkiResort")
Else If (Outside = "Tornado")
  Go To and Stop ("Basement")
Else
  Go To and Play ("TV")
  Set Variable: "LifeIsGood" = True
End If
```

此条件语句再一次检查 Outside的值；但是，这一次它有多种选择。在新的条件语句中，根据所满足的条件会产生各种可能的动作。这是因为在条件语句中添加了其它的 Else If。例如，如果Outside等于“ rain ”，则回去睡觉。如果等于“ sun ”，则去公园玩耍。如果等于“ Snow ”，则去滑雪场。如果等于“ Tornado ”，则去打棒球。而如果以上条件均不满足，则去看电视并将LifeIsGood的值更新为true(请参见本章后面11.1.21节)。

显然，条件语句可以十分复杂。详细信息，请参见本章 11.3节“ ActionScript ”部分。

1. 参数

条件语句的不同部分要求不同的参数。

对于它的If ()部分：

Condition(条件)：输入你想检验的条件。单击等号 (“ = ”)按钮激活Expression Editor。要了解详细信息，请参见11.3.2节“ 使用表达式 ”。

Else/Else If：默认情况下，条件语句不包括 Else或Else If子句。按Add Else/Else If clause按钮添加子句，数量不限(见图11-21)。

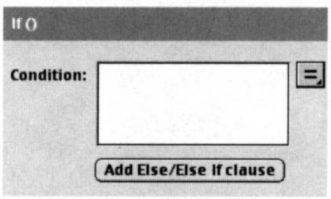

图11-21 按Add Else/Else If clause按钮为If语句添加条件逻辑

如果要为条件语句添加一个 Else或Else If子句，可以在Action面板中单击该语句，此时将出现以下参数：

Action(动作)：选择添加Else子句还是Else If子句。

如果是Else If子句，则可用以下参数。

Condition(条件)：输入要检验的条件。单击等号 (“ = ”)按钮激活Expression Editor。

2. 脚本范例

以下脚本展示了第2帧上的一个帧事件，它查看电影中已加载的帧的数量，然后采取相应的动作。此条件语句的第一部分的作用是：如果还未加载到 100帧，则返回到第1帧重新开始放映。重新放映第1帧后，用此条件语句重新检查第 2帧。如果此时加载的帧数仍然少于 100，时间线将继续返回第1帧并重复此过程。一旦已加载的帧数达到或多于100，则触发Go To动作，代码如下：

```
If (_framesloaded<100)
  Go to and Play (1)
Else
  Go to and Play (MainScene, "Start")
End If
```

3. 实际应用

在Flash电影中添加逻辑，以便一旦满足特定条件，便可执行某些动作。

11.1.18 Loop While

如果条件为真，则可使用 Loop While命令执行一系列动作。 Loop While语句所使用的逻辑可能为：只要 x等于10，则反复执行这些动作 (循环)；否则，停止执行这些动作，而执行 Loop While语句后面的动作。因为只要条件为真， Loop While语句便持续执行你设置的一系列任务，所以如果你不提供一种使条件最后为假的方式，循环将无法终止，而这也使得电影无法正确运行。要了解详细信息，请参见下面的脚本范例。 Loop While语句一般用在Flash的高级脚本中。

1. 参数

Loop While语句有一个参数：

Condition(语句)：必须输入为真的条件才能使循环继续。单击等号 (“ = ”)激活Expression Editor。要了解详细信息，请参见 11.3.2节 “ 使用表达式 ”。

2. 脚本范例

用Loop While动作可制作一些复杂的脚本，以下脚本揭示了 Loop While的概念。首先它设置一个名为Count的变量，并将该变量的初始值设为 1。然后，就是Loop While语句的主体，即当Count小于或等于10时，执行随后的动作。其中就有每循环一次， Count的值就加1。这样，循环10次后， Count将等于11，循环便终止。

```
Set Variable: "Count" = 1
Loop While (Count <=10)
     Action...
      Set Variable: "Count"= Count + 1
End Loop
```

提示 此Loop动作并非像Flash中其它类型的循环一样基于时间间隔：也就是说 Loop While动作并非每秒或每3秒循环一次，而通常是瞬间进行，且不管循环多少次。主要用此动作进行重复运算或创建任务。

3. 实际应用

动态创建变量和电影剪辑实例，并检验数据。

11.1.19 Call

Call(调用)动作可创建可反复使用的 ActionScript。例如，如果按钮组中的每一个按钮都需要执行相同的动作组，可以只创建一次动作组，以后便只需引用此动作组 (请参见下面的脚本范例)。只需在一个关键帧创建一组动作，并为该关键帧分配一个帧标记，这样，无论你何时需要运行该组动作，只需 “ 调用 ” 此帧标记即可。当鼠标 /键盘事件或帧事件触发一个 Call动

作时，时间线并不跳转到该帧，但执行相应的动作。

使用Call动作时应注意几个事项：

- 被调用的帧标记可以位于 Flash Player中的任意时间线上(请参见本章11.2节“ 处理多部电影 ”)。
- 如果Call动作引用一个尚未加载的帧标记，则忽略该 Call动作。
- 如果一个时间线上的变量将由另一个时间线的帧标记上的一组动作来计算 (基本上是Call动作的任务)，变量值必须首先传递给使用 Set Variable动作的时间线。相反，如果 Call动作创建的任何新的值都将由进行调用的时间线来使用，则必须将这些值先传递回来 (请参见以下的脚本范例)。要了解详细信息，请参见本章后面 11.3节中“ 设置或更新不同时间线的值 ”小节。
- 在执行了该组动作之后，将继续 Call动作后面的动作。

1. 参数

Call动作只有一个参数：

Frame(帧)：这是你想使用其动作的帧的路径和帧标记。要了解详细信息，请参见本章后面11.3.2节。

可以根据表达式的值来动态地选择要调用的帧标记，单击“ abc ”按钮激活 Expression Editor。要了解详细信息，请参见本章 11.3.2节“ 使用表达式 ”。

2. 脚本范例

以下脚本展示了一个鼠标事件，它创建一个值为 45的变量UseInCall。关键帧ActionSet 1上的一组动作 (Call动作)将计算此变量的值，并以此设置引导时间线的变量 ActSet1Value的值(见图11-22)。

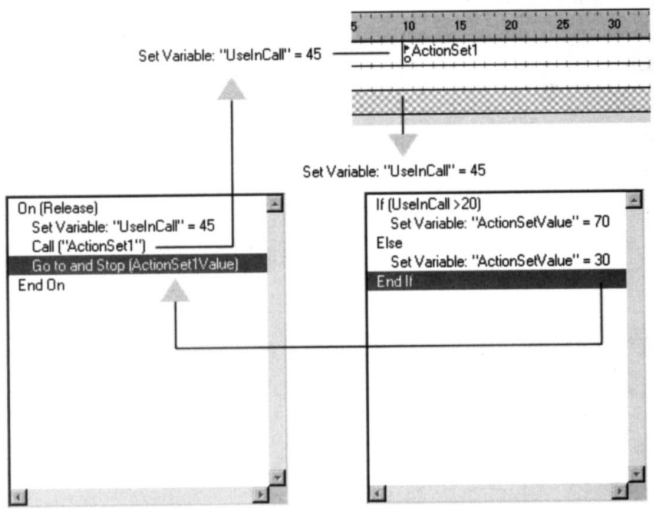

图11-22 Call动作工作原理图解

```
On (Release)
  Set Variable: "UseInCall" = 45
  Call ("ActionSet1")
  Go To and Stop (ActionSet1Value)
End On
```

提示 存放Call中所设置的动作组的关键帧可以在时间线上的任何位置，甚至可以作为一个用户从来不查看或访问的单独场景。这是一种很不错的方法，可将动作组存放在一个很方便的位置。

3. 实际应用

用于可重复使用的动作(与JavaScript函数类似)。

11.1.20 Set Property

你可以使用Set Property(设置属性)动作，在电影放映过程中调整Flash Player中任意电影的位置、大小、旋转、透明度、可见性和名称。

1. 参数

此动作有以下参数(见图11-23)：

1) Set(设置)：此动作可选择要调整的属性，其中有：

- X Position(X位置)：设置电影的水平位置。值以像素为基础，并以原始电影剪辑实例的中心点为参照点(请参见本章11.2节“处理多部电影”)。
- Y Position(Y位置)：设置电影的垂直位置。值以像素为基础，并以原始电影剪辑实例的中心点为参照点。

提示 通常情况下，X和Y位置是以原始电影剪辑实例中心点的左上角为参照点。若选择一个电影剪辑实例，然后在Object Instance中选择Use Center Point，则将参考点改为电影剪辑实例的中心点(见图11-24)。

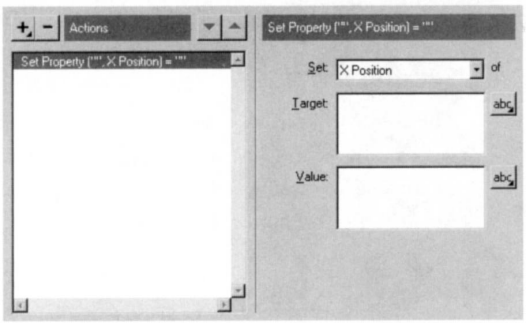

图11-23 Set Property动作可用的参数

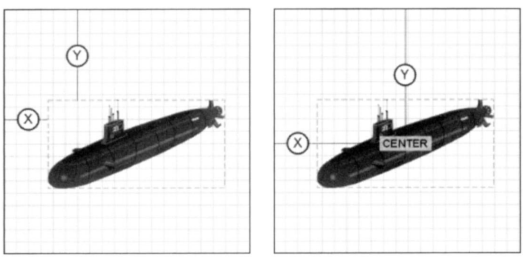

图11-24 若选择电影剪辑实例，然后选择Use Center Point，将改变确定电影的X和Y位置的参考点

- X Scale(X轴缩放)：设置电影在水平方向上的缩放比例 (用百分比表示)。小于100的值表示缩小；大于100的值表示放大。
- Y Scale(Y轴缩放)：设置电影在垂直方向上的缩放比例 (用百分比表示)。小于100的值表示缩小；大于100的值表示放大(见图11-25)。
- Alpha(透明度)：设置电影的透明度。值0使电影剪辑实例不可见。值100则使电影剪辑完全透明。完全透明的电影中的元素保持激活状态，也就是说仍然可以对它们进行交互。
- Visibility(可见性)：设置电影的可见性。True值，或任何非0的值，都使电影剪辑实例可

见；False值，或0，则使电影剪辑实例不可见。你无法与不可见电影中的元素进行交互。

- Rotation(旋转)：设置电影剪辑实例的旋转(用度数表示)。负值使电影剪辑反方向旋转。
- Name(名称)：设置电影剪辑实例的名称，这使得你可以在电影放映期间，为电影重新命名。虽然改变电影剪辑实例的名称，但不改变它与其它电影的相对目录路径(请参见本章中11.2节“处理多部电影”)。

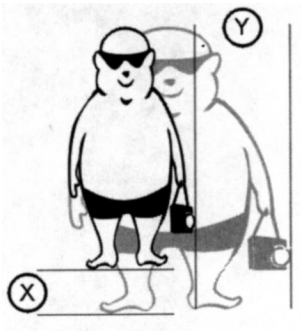

图11-25 X轴和Y轴的缩放属性可以在某一对象原有大小的基础上，对其进行放大或缩小。缩放值用百分比表示

有三个属性影响Flash电影窗口中所有电影的放映。因为它们是全局属性，所以无法只改变单个电影的属性(这就是为什么当选择其中一部电影时，Target参数会变灰)。

- High Quality(高质量)：电影的视觉效果和重放质量可设置为三个级别：0(低质量)，1(高质量)或2(最佳质量)。
- Show Focus Rectangle(显示矩形焦点)：当你使用Tab键对导出的电影中的按钮进行导航时，它们周围将出现一个黄色的矩形焦点(见图11-26)。值True或任何非零的值，都使矩形焦点可见；而值False或0，则使它不可见。
- Sound Buffer Time(声音缓冲时间)：设置流式声音在开始播放前的下载时间(用秒计)。

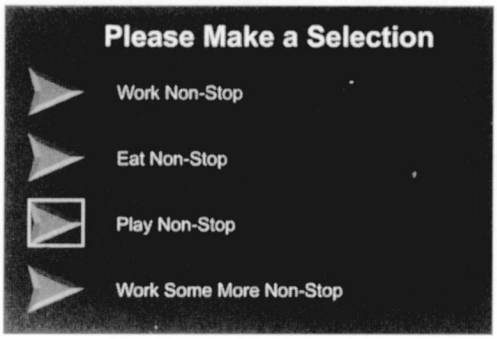

图11-26 当用Tab键在按钮之间导航时，按钮周围将出现矩形焦点

2) Target(目标)：定义要改变属性的电影剪辑实例。

你可以根据表达式的值来选择目标。单击“ abc ”按钮激活Expression Editor。

3) Value(值)：为所选的属性设置值。

2. 脚本范例

以下脚本展示了一个鼠标事件，它将名为MyMovieClip的电影剪辑实例的透明度设置为50%：

```
On (Release)
  Set Property ("/MyMovieClip", Alpha) ="50"
End On
```

以下脚本执行相同的动作。Tell Target命令用来标识要改变其属性的电影剪辑实例。你无须在Set Property动作中标识电影剪辑实例：

```
On (Release)
  Begin Tell Target ("/MyMovieClip")
    Set Property ("", Alpha) ="50"
  End Tell Target
End On
```

3. 实际应用

用于可交互的对象和界面，并可用于缩放。

11.1.21 Set Variable

Set Variable(设置变量)动作用你分配的初始值创建一个新的变量或更新已有变量的值。变量的基本作用是容纳值。相信许多读者在数学课上都学到过这个概念。例如，已知 x=20或y=20。则等式x+y=z的值很容易求得。显然， z=30。大概你绝没有想到中学期间这些乏味的数学课会为你日后成为一位Flash高手打下基础吧！

在Flash中，用变量值来动态地设置其它动作中的不同参数值。例如，将跳转到的帧编号、电影剪辑实例的属性值甚至文本字段中的文本 (请参见下面的脚本范例)。

1. 参数

此动作具有以下参数：

1) Variable(变量)：要创建或更新的变量名。所有的变量名必须以字符开始；但是，随后的字符可以是字母、数字或下划线。

可以根据表达式的值，动态地选择要创建或更新的变量。单击“ abc ”按钮激活Expression Editor。要了解详细信息，请参见本章11.3.2节“使用表达式”。

2) Value(值)：要创建或要更新的变量的值。

可以根据表达式的值，动态地设置变量的值。单击“ abc ”按钮激活Expression Editor。

2. 脚本范例

以下脚本展示了一个鼠标事件，它将 See Through的值设为45。设置好后，该变量的值将用来设置电影剪辑实例MyMovieClip的透明度：

```
On (Release)
  Set Variable: "See Through" ="45"
  Set Property ("MyMovieClip", Alpha) = SeeThrough
End On
```

3. 实际应用

用于动态地产生文本字段中的文本、动态地设置动作参数的值以及跟踪事件已触发的次数。

11.1.22 Duplicate/Remove Movie Clip

Duplicate/Remove Movie Clip(复制/删除电影剪辑)动作在电影放映期间，动态地创建或删除电影剪辑实例。复制电影剪辑实例时，新剪辑继承原剪辑的属性，但不继承它的变量 (见图11-27)(请参见本章11.2节“处理多部电影”)。创建时，副本总是从第1帧开始放映，而不管复制动作触发时原电影剪辑实例正处在哪一帧。

只有用Duplicate动作创建的电影剪辑实例可以用此动作删除；原来就为 Flash电影的一部分的电影剪辑实例不能用此动作删除。

1. 参数

此动作具有以下参数(见图11-28)：

1) Action(动作)：因为你可以用此菜单选项复制或删除电影剪辑实例，所以此参数允许你选择将执行的动作如下：

• Duplicate Movie Clip(复制电影剪辑)：制作一个目标的电影剪辑实例副本，该目标将在

下一个参数中指定。

- Remove Movie Clip(删除电影剪辑)：删除前面创建的副本。因为此动作删除一个电影剪辑，所以当选择它时，“ New Name ”和“ Depth ”参数将变灰。

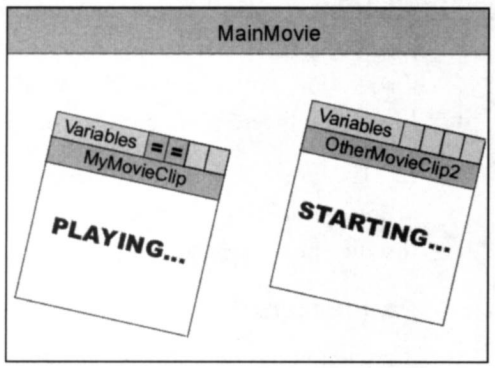

图11-27 复制的电影剪辑实例继承了原剪辑的全部属性，但不继承它的变量

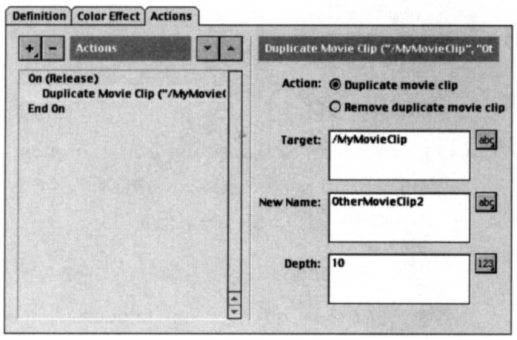

图11-28 Duplicate/Remove Movie Clip动作可用的参数

2) Target(目标)：定义要复制或删除的电影剪辑实例的路径。有关路径的详细信息，请参见本章中11.2节“ 处理多部电影 ”。

可以根据表达式的值，动态地选择目标。单击“ abc ”按钮，激活Expression Editor。详细信息，请参见本章11.3.2节“ 使用表达式 ”。

3) New Name(新建名称)：设置复制的电影剪辑实例的名称。只需输入名称，而无需输入目录路径。复制的电影剪辑继承原电影剪辑的相对路径。

可以根据表达式的值，动态地为复制的电影剪辑命名。单击“ abc ”按钮，激活Expression Editor。

4) Depth(深度)：深度是一个数值，它表示电影剪辑实例副本与同一电影加载级层中其它复本的相对堆叠深度。例如，主电影或底层电影表示级层 0。其它的所有电影都位于主电影的上面。如果其它电影位于级层1、2、3(请参见11.1.13节“ Load/Unload Movie ”)，则副本不会替换这些级层中的任何加载电影，即便你将副本复制到深度 1、2、3也同样。副本总是出现在原电影的上面。

可以根据表达式的值，动态地选择复制的电影剪辑实例的深度。单击“ abc ”按钮激活Expression Editor。

2. 脚本范例

以下脚本展示了一个鼠标事件，其作用是复制名为 MyMovieClip的电影剪辑实例。将名为MyMovieClipClone的副本放置在级层2。创建副本时，改变它的x位置，以便它不会直接出现在原剪辑的上面。

```
On (Release)
  Duplicate Movie Clip ("/MyMovieClip", "MyMovieClipClone", 2)
Set Property ("/MyMovieClipClone", X Position) = 200
End On
```

以下脚本展示了另一按钮上的鼠标事件，其作用是删除 MyMovieClipClone。

```
On (Release)
```

```
  Remove Movie Clip ("/MyMovieClipClone")
End On
```

3. 实际应用

用于游戏，或在任何想动态地创建电影元素的时候，都可使用此动作。

11.1.23 Drag Movie Clip

Drag Movie Clip(拖拽电影剪辑)动作允许用户在电影窗口中任意移动电影剪辑实例的位置，这意味着你可以在电影运行期间，重新安排电影元素的位置。电影剪辑实例随着鼠标光标移动(见图11-29)。

1. 参数

此动作具有以下参数(见图11-30)：

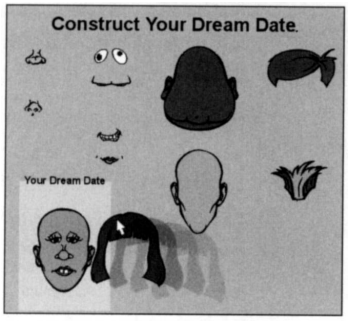

图11-29 Flash支持拖放动作

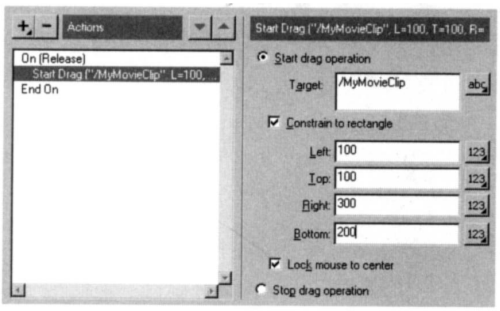

图11-30 Drag Movie Clip动作可用的参数

1) Start Drag Operation(开始拖动)：使目标电影剪辑实例拖动(参见以下内容)。然后此电影剪辑实例将保持可拖动，直到某个事件触发“ Stop drag operation ”或使另一个电影剪辑可拖动。(一次只能拖动一个电影剪辑实例。)

2) Target(目标)：定义要拖动的电影剪辑实例。

可以根据表达式的值，动态地选择要拖动的目标。单击“ abc ”按钮，激活 Expression Editor。详细信息，请参见本章11.3.2节“使用表达式”。

3) Constrain to rectangle(限制为矩形)：默认情况下，拖拽动作允许在电影窗口内部任意拖动电影剪辑实例。使用此选项，可以将它的运动限制在一个矩形区域内。左、上、右和下的值用像素表示，且以父电影剪辑的中心点为参照点(有关父电影剪辑的详细信息，参见本章 11.2节“处理多部电影”)。当使用此选项时，不能将拖动的电影剪辑实例的中心点拖到此矩形之外(见图11-31)。

可以根据表达式的值，动态地设置限制矩形的坐标。单击“ abc ”按钮，激活 Expression Editor。

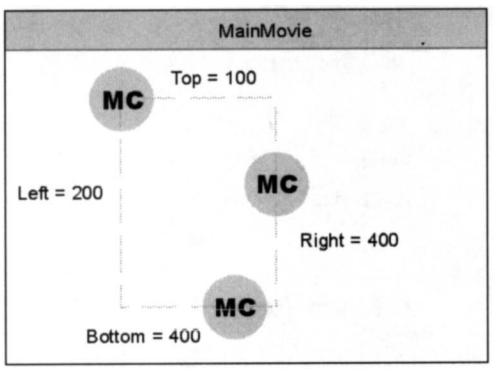

图11-31 将电影剪辑的运动限制在一个矩形之内，以防止它的中心超出此矩形之外。矩形的大小可以设定

4) Lock Mouse to Center(将鼠标锁定在中心)：在默认情况下，当拖动操作开始时，被拖

动的电影剪辑保持它与用户光标的相对位置。选择此选项，则在拖动过程中，将拖动的电影剪辑实例直接放置在光标中心的下面。

5) Stop Drag Operation(停止拖动操作)：停止拖动操作，电影剪辑实例将停留在当时的位置。

2. 脚本范例

以下脚本展示了一个鼠标事件，它拖动名为 MyMovieClip的电影剪辑实例，并在拖动期间，将该电影剪辑锁定在光标的中心。只要此鼠标事件继续，电影剪辑便继续拖动，直到第二个鼠标事件发生为止：

```
On (Press)
  Start Drag ("/MyMovieClip", lockcenter)
  End On
  On (Release Outside)
Stop Drag
End On
```

以下脚本展示了一个鼠标事件，它在一个矩形范围内拖动 MyMovieClip，并在拖动期间，将该电影剪辑锁定在光标的中心。只要此鼠标事件继续，拖动就继续进行，直到第二个鼠标事件发生为止：

```
On (Press)
    Start Drag ("/MyMovieClip", L=250, T=200, R=450, B=400, lockcenter)
  End On
  On (Release Outside)
    Stop Drag
End On
```

3. 实际应用

用于游戏、拖动功能、可自定义界面、拖动栏和滑块。

11.1.24 Trace

此动作用于输出某一动作执行时的自定义信息，它不出现在最终电影中，且不影响其它脚本。此动作可用于查看脚本的运行情况，并主要测试交互性。要了解详细信息，请参见第12章“测试”。

1. 参数

除了所创建的信息外，Trace再无其它参数。

可以根据表达式的值动态地创建信息。单击“ abc ”按钮，激活Expression Editor。要了解详细信息，参见本章11.3.2节“使用表达式”。

2. 脚本范例

以下脚本展示了一个鼠标事件，它使得 Count变量随着鼠标事件每发生一次便加 1，并且(如果在Flash的测试环境中进行测试)使跟踪信息输出到Output窗口：

```
On (Release)
  Set Variable: "Count" = Count+1
  Trace ("The variable is now " & Count)
End On
```

要使Trace命令输出到输出窗口，选择Control|Test Scene或者Test Movie。无论与跟踪相关

的事件何时触发，测试环境中的 Output框都自动打开，并显示跟踪信息。

在测试环境中，以上脚本中的跟踪动作将输出如图 11-32所示的信息。

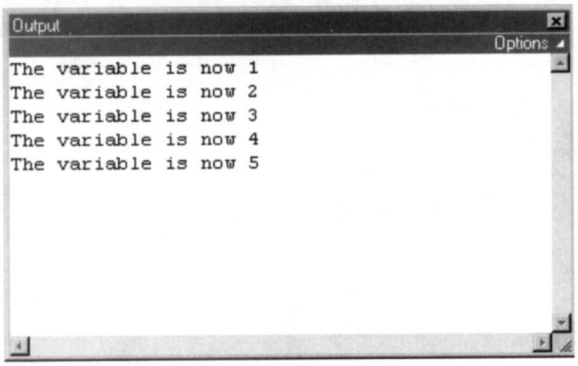

图11-32 Flash测试环境中的Output框将显示你设置的任意Trace信息，以帮助测试ActionScript的流程和执行

3. 实际应用

确保变量正确更新，并检查 ActionScript的功能。

11.1.25 注释

可以在脚本中设置一些注释，以帮助说明复杂的逻辑关系。这些注释并不导出，且对电影没有影响。

11.1.26 处理动作

你已经了解到，动作可以在电影中执行许多强大功能。了解一些动作处理的基本原理，会帮助你创建出更强大的功能。

1. 添加动作

在事件中添加一个动作，是使你的 Flash电影成为可交互的一个基本功能。虽然添加动作不是一件很难的事情，但是仍然需要注意几个事项：

首先，Flash可以为可能发生的任意事件执行多个动作。例如，如果希望一个鼠标事件在将当前时间线发送到第 15帧的同时，还将电影剪辑实例 MyMovieClip的透明属性设为50%，那么你需要遵循以下步骤。

要在一个事件中添加多个动作，应如下操作：

1) 双击按钮，设置鼠标事件触发方式，或者双击关键帧，设置帧事件触发方式。

在本例中，选择按钮。出现 Instance Property对话框。

2) 单击Action选项卡。

3) 单击加号(“ + ”)按钮，显示动作菜单。

4) 因为我们希望用Roll Over鼠标事件触发动作，所以选择 On Mouse Event菜单项。

该菜单项的参数出现在 Action对话框的右侧。

5) 选择Roll Over复选框。

现在鼠标事件设置成功。 Action面板突出显示我们的鼠标事件。

6) 再次单击加号(“ + ”)按钮，然后选择Go To动作。

此动作的参数出现在Action对话框的右侧。

7) 选择Frame Number(帧编号)单选按钮，然后在相邻的文本框中输入 15。

8) 单击加号(“ + ”)按钮显示动作列表，然后选择 Set Property。

此动作的参数出现在Action对话框的右侧。

9) 从Parameters面板上部的Set列表中，选择Alpha。

10) 在下面的Target框中，输入要改变其属性的目标。

在本例中，输入目标的路径和名称： /MyMovieClip(有关目标名称和路径的详细信息，请参见本章后面11.2节“ 处理多个时间线 ”)。

11) 在Value框中，输入数字50。

12) 单击OK。

现在按钮已被配置成一次可执行多个动作。

还可以在一个按钮实例上设置多个执行不同动作的鼠标事件。以下脚本继续使用上面的例子，但是，它不用Roll Over事件触发两个动作，而是用 Roll Over事件触发一个动作，再用Roll Out事件触发另一个动作。

```
On (Roll Over)
  Go To and Stop (15)
  End On
On (Roll Out)
  Set Property ("/MyMovieClip", Alpha) = 50
End On
```

2. 有关嵌套动作的简述

嵌套并不是一个很难的概念，但却很重要。某些 Flash动作本身并不具什么功能。例如 Tell Target动作允许你设置执行某个动作的目标；但是，辅助动作必须明确告诉目标要执行的功能。在一个动作中使用另一个动作即被称为“ 嵌套 ”。嵌套动作在Action面板中缩格表示，因此你可以清楚地找到它们(见图11-33)。

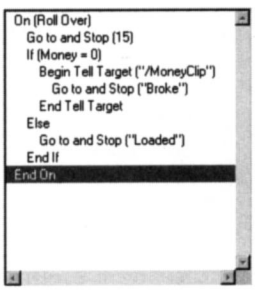

图11-33 Action面板中的内嵌动作用缩格表示，易于查找

内嵌动作如何影响 ActionScript的执行呢？请看以下脚本：

```
On (Release)
   Tell Target ("/MyMovieClip")
     Go To and Play (20)
  End Tell Target
End On
```

上面的脚本中， Go To动作嵌套在Tell Target动作的内部，而Tell Target动作则嵌套在On (Release)鼠标事件的内部。此脚本使得 MyMovieClip的时间线在On (Release)鼠标事件发生时跳转至第20帧，并放映该帧。将此脚本与以下脚本进行比较：

```
On (Release)
Tell Target ("/MyMovieClip")
End Tell Target
Go To and Play (20)
End On
```

虽然此脚本与前一脚本的内容相同，但是因为它的动作不嵌套，所以运行情况不同。此脚本由一个帧事件 (而不是鼠标事件)触发，该事件使得当前时间线 (而不是MyMovieClip时间线)跳转至第20帧，并放映该帧。此脚本中的鼠标事件和 Tell Target动作无用。

以下动作是可嵌套的，也就是说，它们通常需要第二个嵌套的动作才能影响脚本。可嵌套的动作总是带有一个End语句，该语句自动插入脚本，如下所示：

```
Tell Target
  Tell target to do what?
End Tell Target

If Frame is Loaded
  Then do what?
End Frame Loaded

On Mouse Event
  Do What?
End On

If (something is True or False)
  Do What?
End If

Loop While (something is True or False)
    Repeatedly this:
End Loop
```

提示 有关如何在脚本中向上或向下移动动作以进行嵌套的详细信息，请参见后面的小节“ 动作顺序 ”。

3. 删除动作

如果你发现动作放置的位置不对，可以很容易地将它从Action面板中删除。

要删除某个动作应如下操作：

在Action面板中选择该动作，然后单击它上面的“ - ”按钮。

提示 如果你选择一个包括嵌套动作的命令 (例如 On Mouse Event，Tell Target或者If语句)，则将同时删除所有嵌套的动作(见图11-34)。

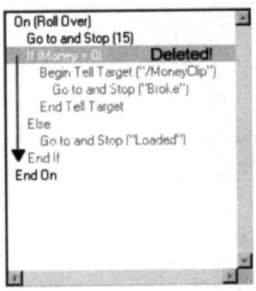

图11-34 若删除嵌套中的最上一级动作，则它下面嵌套的所有动作也将删除

4. 动作顺序

在Flash中就像在生活中一样，动作需要按特定的顺序进行，才能使事情的发生连贯且合乎逻辑。 Flash按照动作在Action面板中的顺序，依次执行它们。如果动作的顺序不正确，则脚本的执行可能不会如你所愿。

例如，下面的两个脚本中，第一个脚本的顺序是正确的：

```
On (Release)
  Set Variable: "DynamicFrame"= 20
  Go To and Play (DynamicFrame)
```

```
End On
```

此脚本展示了一个鼠标事件，它创建一个值为 20的变量DynamicFrame。该变量设置好之后，再设置一个Go To动作以跳转至以该变量的值为基础的帧编号。因为已为 DynamicFrame分配值20，所以Go To动作将跳转值第20帧。这种顺序是正确的。而以下脚本的顺序则出现了一点问题：

```
On (Release)
  Go To and Play (DynamicFrame)
  Set Variable: "DynamicFrame" = 20
End On
```

此脚本展示了一个鼠标事件，它触发一个 Go To动作以跳转至以DynamicFrame的值为基础的帧编号。但问题是，按照这种动作顺序， DynamicFrame尚未创建出来。事实上，它是在Go To动作执行之后才创建出来的；因此， Go To动作没有任何效果。

提示 如果用户再次按鼠标按钮，则Go To and Play动作将跳转至第20帧。这是因为第一次单击创建了变量 DynamicFrame，并将它的值设为 20。这样，再单击按钮时，DynamicFrame的值就可用了。

这里的基本思想就是要意识到动作执行的顺序，这可能关系到你脚本运行的成败。

要重新安排动作的顺序，应如下操作：

1) 在Action面板中，单击一次以选择要重排的动作。

2)单击Action面板右上角的向上或向下箭头，以便将该动作在整个顺序中向上或向下移动 (见图11-35)。

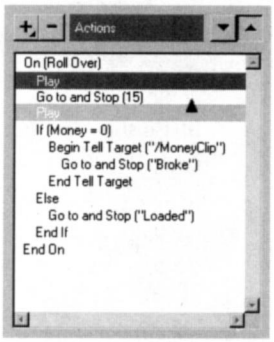

图11-35 按Action面板上部的向上或向下箭头以重排动作

5. 剪切、复制和粘贴动作

如果你已经在一个关键帧或按钮上创建出具有合理顺序的动作，并想将它用在别处，那么你无需重复这些工作，而只需进行剪切(或复制)和粘贴。

要剪切(或复制)和粘贴动作，应如下操作：

1) 在Action面板中，选择要剪切或复制的动作：

要选择某个动作，只需单击它。

要选择多个动作，可按住 Control键(Windows)或Command键(Macintosh)，单击多个动作。

要选择一个动作范围，只需单击该范围中的第一个动作，然后按住Shift键单击该范围中的最后一个动作 (见图11-36)。

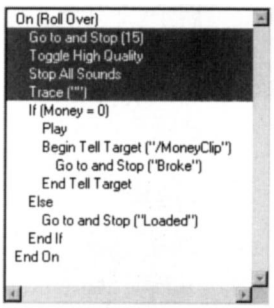

图11-36 可以选择剪切、复制和粘贴的范围

2) 右击(Windows)或者按住Control键单击(Macintosh)某个被选动作，然后从出现的菜单中选择 Cut(剪切)或Copy(复制)，将该动作放置在系统剪贴板中。

3) 单击OK关闭Action对话框。

4) 双击要粘贴该动作的按钮或关键帧。

5) 单击Action选项卡将它激活。

6) 在Action面板，右击(Windows)或者按住Control键单击(Macintosh)，然后从出现的菜单

中选择Paste(粘贴)。

被剪切或复制的动作出现在当前 Action面板中。

11.2 处理多部电影

显而易见，一部包含多部电影 (主电影、任何电影剪辑实例或者任何用 Load Movie动作加载到主电影的电影)可以包括多个时间线。每部电影都是一个具有自己的时间线、变量和属性的单独实体，并且可唯一识别 (见图11-37)。

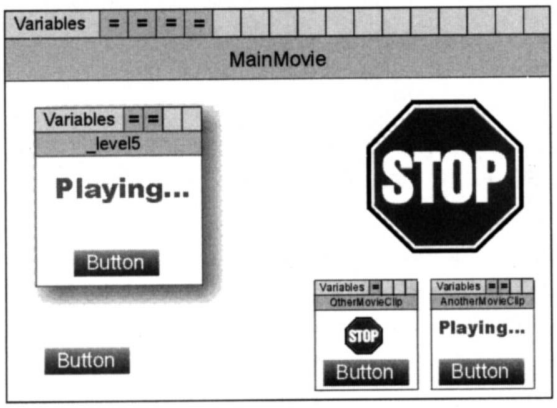

图11-37 主电影可独立于任何电影剪辑或级层而动作，反之亦然

你已经了解到，一个时间线上的鼠标和帧事件可以影响所出现的其它任何时间线上的运动、属性和变量。这样，当你同时使用多个电影剪辑时，可以多个时间线为目标来执行一个动作，从而提供无穷无尽的多样性。

只有显示出来的电影剪辑才能用作目标。例如，如果你的电影只有 40帧能使用某一个电影剪辑，那么该电影剪辑只能在这 40帧中被当作目标。

而且，如果某部电影要响应另一个电影中的事件，那么你必须特别指定以它为目标。我们将详细讲述如何实现这一点。

提示 有关如何为电影剪辑实例分配名称以便将它用作目标的方法，已在第7章“ 图符 ”中讲到。

要以一个特定的时间线为目标，你需要对它进行标识。方法有四种：空格、名称、级别编号或者级别和名称。这样归纳显然比较笼统，下面我们就略作描述。

1. 空格

如果动作的前面未加目标名称或级别编号，则目标就被视为当前的电影或时间线。有关如何确定当前电影的详细信息，请参见本章前面的“ 目标 ”。

以下脚本就是以当前电影为目标：

```
On (Release)
  Set Variable: "Derek" = 32
End On
```

2. 名称

可以为电影剪辑实例分配名称，以便在ActionScript中对它们进行标识(请参见第7章)。这样，

便有可能通过ActionScript对它们进行控制。主电影中的事件可以使电影剪辑跳转至它时间线上的一帧、使电影剪辑不可见以及对它进行旋转等等。相反，未命名的电影剪辑虽然可以放映，但却不能作为目标。你可以为同一个电影剪辑的不同实例设立不同的名称，这样它们可以分别用作目标(见图11-38)。

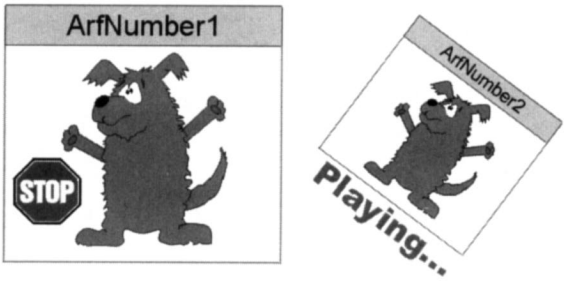

图11-38 如果给予不同的名称，相同电影剪辑的各个实例可以分别用作目标，且独立动作

同时，因为电影剪辑本质上是一个具有自己的图形、按钮、声音和时间线的 Flash 电影，所以一个电影剪辑可以通过按钮 /键盘和帧事件控制另一个电影剪辑。电影剪辑甚至可以控制主电影。

在ActionScript中以一个电影剪辑为目标时，必须正确拼写它的名称，但是不一定区分大小写(也就是说， MyMovieClip等同于mymovieclip。)

以某个电影剪辑实例为目标的脚本应如下所示：

```
On (Press)
  Tell Target ("/MyMovieClip")
   Go To and Stop (20)
  End Tell Target
End On
```

要以另一电影剪辑实例中的电影剪辑实例为目标，则应如下所示：

```
On (Press)
  Tell Target ("/MyMovieClip/AnotherMovieClip")
   Go To and Stop (20)
  End Tell Target
End On
```

电影剪辑名称前面的“ / ”表示电影剪辑的目录路径。详细信息，请参见本小节后面的“目标路径”小节。

3. 级层

使用Load/Unload Movie动作的本质是将一个 .swf文件加载到一个已有的 .swf文件。此动作将该文件加载到特定级层，例如 1或20。通过标识电影所在的级层，可以将电影的时间线(以及该电影中任意电影剪辑的时间线)用作目标。例如，如果将一个电影加载到级层 5，那么它的内容将出现在级层0到级层4的上面(0是原电影或主电影)。

若要以某级层的主时间线为目标，则应设置以下脚本：

```
On (Release)
  Tell Target ("_level5")
    Go To and Stop (25)
  End Tell Target
End On
```

若要以另一级层中的电影剪辑为目标，则应设置以下脚本 (见图11-39)：

```
On (Release)
  Tell Target ("_level5/MyMovieClip")
   Go To and Stop (25)
End Tell Target
End On
```

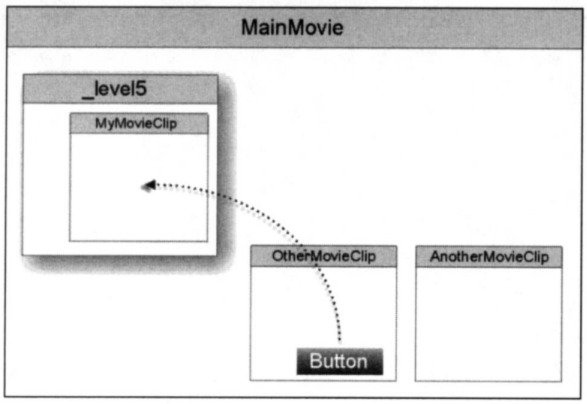

图11-39 可以另一级层中的电影剪辑为目标

4. 目标路径

除了设置目标名称，还需要设置地址 (或者目标路径)。如果你知道如何在Web页上设置超链接，那么目标路径对于你来说不过是小菜一碟。

Flash中的目标路径以分层结构为基础，这与计算机或服务器上文件的组织方式一样。

为帮助你理解目标路径的来源，不妨进行这样一个类比。可以将 Flash项目想象成一个家庭结构，主电影是母亲，现在，母亲有几个孩子，分别叫做 Joe、Lucy和Bob。在Flash中，这些孩子表示主电影中的电影剪辑实例，并被称为子电影 (参见下面的文字框“父子关系”)。

父子关系

就像一个家庭有父母、孩子以及孩子的孩子一样， Flash电影也可以包含几部电影，而这几部电影又可以包括几部电影。所有这些电影之间的关系被称为父子关系。父电影是包含其它电影或子电影的电影。例如，父电影可以包含子电影，哪怕该电影本身也是其它电影的子电影。

在此需意识到的重要一点是，当你改变父电影的属性时，它的子电影继承相同的属性。例如，如果你使一部父电影透明，那么它所有的子电影都将具有透明属性。但是，反过来却并非如此：改变子电影的属性将不会影响其父电影的属性。

下面的脚本展示了Flash中的“家庭”结构(见图11-40)：

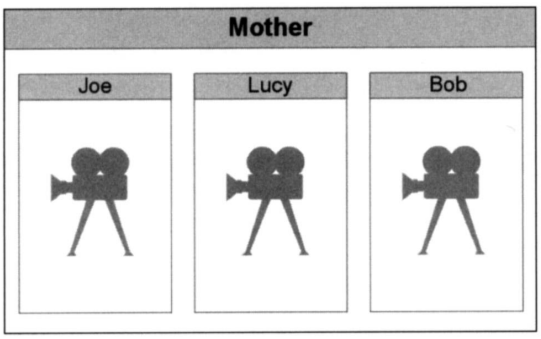

图11-40 任何时候出现在Flash Player窗口中的各个时间线都可当作一个家庭结构，即一种母电影包含其它电影或子电影的家庭结构

```
Mother(_level0)
        /Joe
        /Lucy
        /Bob
```

如果单击母电影中的一个按钮时Lucy不可见，则脚本中的目录路径可能应如下所示：

```
On (Release)
  Tell Target ("/Lucy")
    Set Property ("",Visibility) = 0
  End Tell Target
End On
```

以下脚本与前一脚本的功能基本相同，但没有 Tell Target命令。事实上，目录路径包括在Set Property语句中(两种方法实际运行情况一样)：

```
On (Release)
  Set Property ("/Lucy", Visibility) = 0
End On
```

现在，假设Joe自己有两个孩子。他决定分别给他们起名为 Junior和Youngster。如果Joe表示一个电影剪辑实例，他的孩子则表示电影剪辑实例中的电影剪辑实例。

整个家庭结构应如下所示：

```
Mother(_level0)
      /Joe
              /Junior
              /Youngster
      /Lucy
      /Bob
```

记住以上家庭结构，单击 Mother电影中的一个按钮，以使 Joe的孩子Junior不可见，则此目录路径在脚本中的表示应如下所示：

```
On (Release)
  Tell Target ("/Joe/Junior")
     Set Property (" " , Visibility) = 0
End Tell Target
End On
```

现在，我们开始一些家庭活动：在这里， Mother不是唯一的主宰。 Joe可以吩咐Lucy；Bob可以吩咐Junior；甚至Youngster也可以吩咐Mother(Mother实际上是Youngster的祖母)。

请看下面的目标路径范例(见图11-41)：

对于Mother中的鼠标或帧事件：

Joe、Lucy或Bob：目标路径应为“ /Joe ”或“ /Lucy ”或“ Bob ”。

Junior或Youngster：目标路径应为“ /Joe/Junior ”或“ /Joe/Youngster ”。

对于Bob中的鼠标或帧事件：

Mother：目标应为“ ../ ”。

Lucy或Joe：目标路径应为“ /Lucy ”或“ /Joe ”。

Junior或Youngster：目标路径应为“ Joe/Junior ”或“ /Joe/Youngster ”。

对于Lucy中的鼠标或帧事件：

Mother：目标应为“ ../ ”。

Bob或Joe：目标路径应为“ /Bob ”或“ /Joe ”。
Junior或Youngster：目标路径应为“ /Joe/Junior ”或“ /Joe/Youngster ”。
对于Joe中的鼠标或帧事件：
Mother：目标应为“ ../ ”。
Lucy或Bob：目标路径应为“ /Lucy ”或“ /Bob ”。
Junior或Youngster：目标路径应为“ Junior ”或“ Youngster ”。
对于Junior中的鼠标或帧事件：
Mother：目标应为“ ../../ ”。
Lucy或Bob：目标路径应为“ ../..Lucy ”或“ ../..Bob ”。
Joe：目标路径应为“ ../ ”。
Youngster：目标路径应为“ ../Youngster ”。
对于Youngster中的鼠标或帧事件：
Mother：目标应为“ ../../ ”。
Lucy或Bob：目标路径应为“ ../../Lucy ”或“ ../../Bob ”。
Joe：目标路径应为“ ../ ”。
Junior：目标路径应为“ ../Junior ”。
最后我们需了解的目标路径是当用Load/Unload Movie动作加载电影时所创建的目标路径。

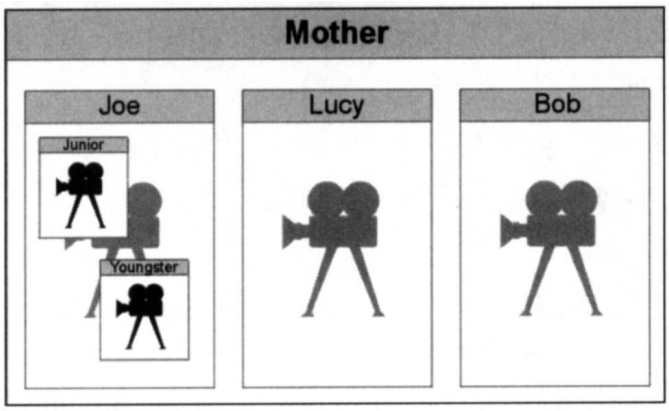

图11-41 目标不同的电影层次结构

无论何时将新电影(除了已存在的主电影)加载到Flash电影窗口，该电影都被赋予一个级层编号。原电影自动分配级层0。可将任何加载到某一级层的新电影看作另一个包含子电影的父电影。

假想级层5上有这样一个电影结构：

```
Mother (_level5)
       /Kathy
                /Ashlie
                /Carla
       /Jack
       /Liz
```

你可以看到，除了级层0上原来的“家庭”，级层5上还存在一个新家族(见图11-42)。

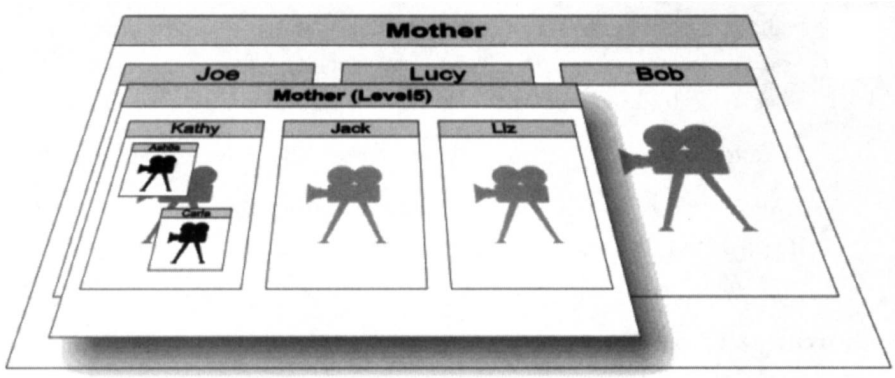

图11-42 在Flash Player中加载另一个电影时将出现一个新的分层结构

对于任何加载到Flash Player窗口的其他电影，若要以级层5中电影的主时间线为目标，应如下设置脚本：

```
On (Release)
  Tell Target ("_level5")
    Go To and Stop (10)
  End Tell Target
End On
```

对于任何加载到Flash Player窗口中的其他电影，若要以名为 Ashlie的电影剪辑(它包含在级层5的电影中)为目标，则应如下设置脚本：

```
On (Release)
  Tell Target ("_level5/Kathy/Ashlie")
    Go To and Stop ("Bed")
  End Tell Target
End On
```

提示 无论何时以另一级层中的电影或电影剪辑为目标，都必须在目标路径中包括“ _level ”和级层编号。

5. 目标编辑器

Flash包括一个名为Target Editor的实用程序以帮助你定义目标路径。 Target Editor是一个可视工具，它展示了电影的分层结构。可以通过以下命令和动作来获得它： Tell Target、Load/Unload Movie(当加载到某个目标时)、Set Property、Duplicate/Remove Movie Clip和Drag Movie Clip。

要使用Target Editor，应如下操作：

1) 选择Tell Target命令，或选择一个必须为它定义目标的动作。

2) 单击Target参数框旁边的“ abc ”按钮，然后选择 Target Editor(见图11-43)。

出现一个对话框，显示可用的电影剪辑目标的目录。如果某个电影剪辑实例名称旁边出现一个加号 (“ + ”)，则该电影具有相关的子电影剪辑。单击此加号展开目录。

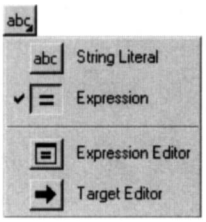

图11-43 任何要定义目标的地方都可获得Target Editor

3) 双击某个目标的名称以将它自动作为目标输入。

此框中只出现当前电影剪辑的子电影剪辑实例。如果你想以一个父电影剪辑或另一子电影剪辑为目标，则必须通过手动输入它的路径。

6. 设置和获得属性

电影的属性具有两个基本操作：设置和获得属性。要执行其中任何一个任务，都需要电影的名称和路径。

设置属性需要在某个事件触发时，以某种方式改变电影的属性。以下脚本展示了 Mother中的一个帧事件，它将电影剪辑 Junior的透明度设为50：

```
Set Property ("/Joe/Junior", Alpha) = 50
```

或者是这种脚本：

```
Tell Target ("/Joe/Junior")
  Set Property ("" ,Alpha) = 50
End Tell Target
```

请注意以上脚本中，使用了你以前所学的名称和路径习惯，以便你可以将某部电影作为目标来处理它的属性。

获得某部电影的属性则允许你估计它当前的状态，并采取相应的动作。以下脚本继续使用前面的例子，并展示了主电影 (Mother)中的一个鼠标事件，该事件查看 Junior的高度是否大于200。如果是，他父亲Joe的宽度应设为20；如果不是，Joe的宽度则应设为50：

```
On (Roll Over)
  If (GetProperty("/Joe/Junior",_height) > 200)
     Set Property ("/Joe",_width) = 20
   Else
     Set Property ("/Joe",_width) = 50
  End If
End On
```

无需使用Get Property功能来获得当前电影的属性。只需在某部电影中的某个事件触发此动作以获取另一电影的属性时使用此功能。如果你想估计当前电影的状态 (参见本章前面11.1.2节“目标”)，则应设置以下脚本：

```
On (Roll Out)
  If (_alpha > 50)
    Set Property ("", Alpha) = 100
  End If
End On
```

以上脚本检查当前电影的透明度，如果大于 50%，Set Property动作便将该数字设置为100%。你无需在If语句中标识当前电影，因为如果未指定目标，脚本的默认值便以当前电影为目标。

7. 识别帧标记

当使用Call动作时，你需定义要调用的帧标记。你可以调用所显示的任意时间线中的某个帧标记。要定义一个帧标记，首先需要定义它所在的时间线的名称和路径，然后加上冒号和帧标记本身。以下脚本展示了主电影中的一个 Call动作，它以电影剪辑 Junior中的帧标记HisRoom为目标(继续使用前面的例子)：

```
Call ("/Joe/Junior:HisRoom")
```

若要以当前电影中的某一帧标记为目标，则只需要帧标记：

```
Call ("MyFrame")
```

11.3 ActionScript

欢迎学习Flash ActionScript！可以将ActionScript当作基本程序设计语言，从而设计执行各种任务的Flash电影。同大多数程序设计语言一样， ActionScript中的术语(或单词)用来传达意思，指令可提供结构，而标点符号则构成上下文。现在你可能对此还无法了解，那么暂且先将它搁在一边，很快你就会有所感觉。

11.3.1 变量

变量是任何程序设计脚本的基本和重要组成部分，实际上，变量是组成动态软件的关键内容。同样，用ActionScript创建交互作用时也用到变量。

变量是值的容器。虽然你不一定总是能意识到到这一点，但实际上你的生活就是由变量组成的。例如，请看以下的例子：

```
X = 25
Name = Derek
Age = 29
Income = 500
Best Band = The Beatles
IQ = 47
```

X、Name、Age、Income、BestBand和IQ均为变量名，而等号后面的信息则是该变量的值。ActionScript中同样也是如此。当使用Set Variable动作时，你可能会创建如下几个变量：

```
Set Variable: "Name" = "Derek"
Set Variable: "Age" = 29
Set Variable: "Income" = 500
```

在Flash中创建变量并为变量命名时，需注意以下几点：

- 所有变量名必须以字符开头。该字符后面可以是字母、数字或下划线。而且，变量名不区分大小写；因此， MyVariable等同于myvariable。变量名不能含空格。只有在用 Set Variable动作分配变量时， Flash才自动为变量加上引号；而表达式中的变量则不加引号(参见本章后面11.3.2节“ 使用表达式 ”)。
- 每部电影或电影剪辑都有一组唯一的变量。只要时间线显示，则它里面的变量继续存在，且可设置或检索这些变量的值。
- 为可编辑的文本框分配变量名的方式与其他所有变量均不相同。尽管你可以用 Set Variable动作分配大多数变量名，但是文本字段将名称作为它的一种属性来分配。文本字段的值(也就是它所显示的文本)由电影放映时用户所输入的文本决定；或者也可以用Set Variable动作动态产生。详细信息，请参见本节后面“ 处理文本字段 ”小节。
- 虽然变量的值可以改变，但是名称保持不变。例如，某一时刻电影中的 x值等于25，稍后，它的值可能变成720。

变量名应具有一定意义。如果变量所表示的值是用户单击鼠标按钮的次数，那么你应将该变量命名为MouseClick或类似的名称。

1. 值

在Flash中，变量值可以有以下形式：

数字：数字值指的是从 0到999,999+之间的任意值。变量 Age的值可能为 20，那么在ActionScript中应为：

```
Set Variable: "Age" = 20
```

字符串：程序设计语言中字符串通常表示文本值。典型的字符串值可以是一个字母“ a”，也可以是几个句子，如“ Hello, what ’ s your name? Does your dog bite, or is it a nice dog? ”字符串值可以包含任意多字母(在合理的范围之内)，并且可包括文本、空格、标点符号甚至数字。甚至可以把“ 345 ”当作一个字符串值。含数字的字符串值用引号与实际数字区分开来，也就是说， ActionScript用引号表示字符串。这样， 1996是数字，而“ 1996 ”则是一个字符串值。例如，以下几个变量便包含字符串值：

```
Set Variable: "PhraseThatPays" ="Flash 4 Rocks!"
Set Variable: "FavoriteWife"="Kathy"
Set Variable: "FavoriteWifesAge"="Always 29"
```

逻辑值：逻辑值用来判断某个条件是否存在。逻辑值有两种， True或False。在Flash中，用0表示False，任意非0值表示True。为变量分配逻辑值如下所示：

```
Set Variable: "MacromediaRocks"= True
Set Variable: "MacromediaRocks"= 1
```

无：虽然它不是一个真正的值，但可表示某个字符串值的不存在。例如，如果你记不起某人的姓名，那么应以如下方式表示你的记忆变量：

```
Set Variable:"Memory" = " "
```

2. 分配值

为变量分配值有两种形式，文字形式或表达式形式。以下是文字分配形式：

```
Set Variable:"Cost"= 25.00
```

或者

```
Set Variable:"Name" ="John Doe"
```

请注意分配给此变量的值并不是动态的。例如，如果你希望将 Cost的值用在ActionScript中其他的地方，那么该值将总是 25。同样， Name的值也总是John Doe。显然，这种结果并不让人感到振奋。要创建一个可动态分配的值，应使用表达式。如：

```
Set Variable:"Product"= 20.00
Set Variable:"Tax"= 5.00
Set Variable:"Cost"= Product + Tax
```

上面的脚本中， Cost变量的值是反身变量Product + Tax。要将Cost分成最基本的部分，则应为“ Cost ” = 20.00 + 5.00，总量为25.00。但是，如果变量Product的值改为22.00，而Tax的值变为6.00，那么Cost变量的值将自动变为 28.00，这是因为它是基于 Product + Tax的值。文字是明确分配给某个变量的值，而表达式则是基于某个运算短语的值 (参见本章后面 11.3.2节“ 使用表达式 ”)。以下脚本中，变量“ Name ”的值也是基于表达式的：

```
Set Variable:"FirstName"= "John"
Set Variable:"LastName"= "Doe"
Set Variable:"Name"= FirstName & LastName
```

要创建一个变量并为它分配值，应如下操作：

1) 双击舞台或时间线上某一关键帧上的按钮实例。

Instance或Frame Property框出现。

2) 单击Action选项卡。

3) 单击加号(“ + ”)按钮，然后从出现的菜单中选择 Set Variable。

Set Variable参数出现在Action对话框的右侧。

4) 在Variable框中输入一个描述该变量的名称。

不用加引号；Flash会自动完成。

5) 需要在Value框中进行一项选择：

如果该变量的值是一个文本字符串，则只需将它 (不带引号)输入到此框中。此框右边的按钮显示“ abc ”，这意味着无论你输入什么，都将输入的内容当作字符串值 (即便你输入465这样的值也如此)。

如果希望这个值被当作数字、逻辑值或者表达式，则单击“ abc ”按钮，然后选择“ = ” (见图11-44)。如果在此过程中留意Action面板，则会发现值部分的引号被去掉(在将数字、逻辑值或表达式作为值分配的时候，这是有必要的)。

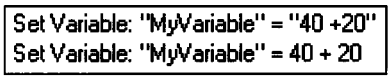

图11-44 若将变量值的引号删除，则它的值将为数字、逻辑值或表达式，而不是文本字符串

6) 单击OK。

当电影放映时，若单击你在步骤 1中设置的按钮，或到达在该步骤中设置的关键帧，则此变量将被创建出来，并成为当前时间线的一部分。

3. 设置或更新不同时间线的值

可以使用Set Variable动作创建或更新Flash Player窗口中任何电影(包括电影剪辑和加载的电影)的变量。只需在输入的变量名的前面加上时间线的路径，例如：

```
/MyMovieClip:MyVariable
```

如果希望在子电影中处理父电影中的变量，则应为：

```
../:MyVariable
```

当在表达式中使用不同时间线的变量时，也是同样的。如果希望用某一时间线中的变量值创建另一时间线中的表达式，只需使用与上例相同的句法。请看以下例子：

```
If (/MyMovieClip:MyVariable + 50 = 300)
   Go To and Play (20)
End If
```

电影(或时间线)之间变量值的传递是Call动作的使用过程中一个必不可少的部分。此动作有时允许某个时间线中的一组动作运算并处理另一时间线中的变量值，然后将处理和运算得来的变量值发送回原来的电影。当使用此动作时，时间线之间变量值的传输并不是自动的；必须使用Set Variable动作和适当的句法来回发送变量数据。

提示 有关创建和使用表达式的详细信息，请参见本章11.3.2节“ 使用表达式 ”。

4. 更新变量值

一旦已创建出某个变量，那么，只要它所在的时间线继续显示，就可使用它的值。假设某个变量存在于某个你永远也看不到的位置，那么无论时间线移动与否，该变量都在那里。

有些时候，你希望更新某个变量的值。例如，你希望将它的文字值从 Joe改为Fred，或者

希望更新一个表达式的值，以跟踪按钮单击的次数。

假设某个事件触发创建某个变量。当你创建此变量时，将它命名为 MyVariable，并赋予值Hello。如果你希望在电影播放期间的某个时间点，将它的值改为 Goodbye，那么只需在某个鼠标或帧事件上再附上一个 Set Variable动作。该动作将 MyVariable标识为变量，而将Goodbye标识为该变量将更新的值。使用 Set Variable动作时需记住的一点是，Flash查看你所分配的变量名是否已经存在。如果不是，将创建一个；如果是，则对它进行更新。

可以通过鼠标或帧事件来更新基于表达式的变量值，而无需使用另一个 Set Variable动作。请看以下脚本：

```
On (Release)
  Set Variable:"Count"= Count + 1
End On
```

在上面的脚本中，鼠标事件每发生一次，Count的值就加1。这是因为Count的值是一个表达式，而表达式后面的逻辑则为：

- 当鼠标事件发生时，Count的值等于Count的当前值加1。
- 要计算Count的值，首先应获得它的当前值。
- 假设它的当前值是15。
- 那么，15加1就是16。

下次鼠标事件发生时，Flash将检查Count的当前值，现在它的当前值应为 16，然后16加1就是17。

文本字段的字符串值总是显示在字段中，并经常更新。因此，如果一个文本字段显示单词Mushroom，那么它是该字段的值；但是，如果在该字段中键入 tomato，它将自动成为新的值。

5. 处理文本字段

第4章“文本”中已经介绍过，文本字段是由变量名标识的动态文本块。你可以在ActionScript中使用此变量名来评估在文本字段中键入的文本。还可以用变量名动态地产生显示在字段本身中的文本。

例如，假设舞台上某个文本字段的变量名为 Food。有人已在该字段中输入单词 Burger。按某个按钮，执行以下脚本：

```
On (Release)
  If (Food eq "Burger")
    Go To and Play ("Diet")
    Else If (Food eq "Salad")
      Go To and Play ("EatUp")
  End If
End On
```

如果此脚本求出Food等于Burger，便使时间线跳转到标记为Diet的帧。

如果你想动态地产生显示在文本字段 Message中的信息，应使用以下 ActionScript(它用鼠标事件触发)。此ActionScript将产生以Name文本字段中所输入的文本为基础的信息。本例中，我们在此字段中输入Ashile：

```
On (Release)
  Set Variable: "Message"="Hello,"& Name & ",
  is your homework finished?"
```

```
End On
```

当鼠标事件发生时，此脚本将在文本字段 Message中产生以下信息：

```
Hello, Ashlie, is your homework finished?
```

11.3.2 使用表达式

表达式是使Flash电影变成真正动态和可交互的核心，它使得每个用户对于电影的体验都独一无二。

在Flash中，表达式是一个短语或变量、数字、文本和操作符的集合，它所等于的值可以是字符串、数字或逻辑值。可通过计算表达式来执行多项任务，包括设置变量的值、定义将影响的目标、确定将跳转到的帧编号以及拖动。请看以下脚本，它使用表达式计算将跳转到的帧编号：

```
On (Release)
  Go To and Play (24 + 26)
End On
```

上面脚本中的表达式，也就是 24 + 26的值将使得时间线跳转到第 50帧。显然，这十分简单。你甚至可以创建这样一个脚本，它使用表达式以不同的方式完成相同的任务：

```
On (Release)
  Set Variable: "FavoriteNumber"= 24
  Set Variable: "SecondFavNumber"= 26
  Go To and Play (FavoriteNumber + SecondFavNumber)
End On
```

首要事项

书写表达式时，需确定的第一件事情就是返回值的类型。它可能是一个字符串，例如“ Hello, there ”，也可能是一个数字，例如 560，或是一个逻辑值，如 True。根据你要完成的任务来选择值类型。

你可能已意识到，许多动作参数可以使用表达式来设置值。如果你打算这样做，那么需注意此参数通常所使用的值类型，这样，就可以相应地创建自己的表达式。例如，当使用 Get URL动作时，URL参数需要一个字符串值，如 http://www.mydomain.com/mymovie.swf。如果你用表达式来产生此参数的值，那么它必须等于一个字符串。相反，如果你想使用表达式来设置电影的透明属性 alpha，那么，它应等于一个数值。如果书写一个等于 telephone的表达式则没有意义，也无法正确设置 alpha。

11.3.3 数字运算符

当进行数值(或值为数字的变量)运算时，数字运算符用来进行加和减这样的运算。数字运算符有两种：算术运算符和比较运算符。表 11-2提供了一列算术数字运算符以及它们的功能简介。

表11-2 算术数字运算符

运 算 符	描 述
+	将一个数值加到另一个数值
-	从一个数值中减去另一个数值
*	用一个数值乘以另一个数值
/	用一个数值除以另一个数值

1. 脚本范例

```
On (Release)
```

```
  Set Variable:"Season"= 4
  Set Variable:"Hrs"= 24
  Set Variable:"Mins"= 60
  Set Variable:"MinsASeason"=(365 / Season)*(Hrs * Mins)
End On
```

此脚本创建了4个变量，并为它们分配数值。其中的三个变量分配的是文字值，而第 4个变量分配的是表达式。请看以下使用了实际数值的表达式：

```
(365 / 4)*(24*60)
```

第一步运算得到：

```
91.25 * 1440
```

进一步得到：

```
131400
```

因此，MinsASeason的值为131400。

要了解表达式的动态程度，请看以下脚本。它与前一个脚本的功能相同，但是使用了两个基于表达式的变量：

```
On (Release)
   Set Variable:"Season"= 4
   Set Variable:"Hrs"= Season + 20
   Set Variable:"Mins"= 60
   Set Variable:"MinsASeason" =(365 / Season)*(Hrs * Mins)
End On
```

Hrs的值以表达式的值 24为基础。该值在脚本的第二个表达式中用于和 Mins的值相乘。

只能对数字值进行加减处理。还可以使用比较运算符来比较变量的数字值，尤其是在使用If and Loop While命令的时候。

表11-3提供了一列比较数字运算符及其功能。

表11-3 比较数字运算符

运 算 符	描 述
=	等于
<>	大于或小于
<	小于
>	大于
<=	小于或等于
>=	大于或等于

2. 脚本范例

```
On (Release)
  Set Variable: "Paycheck"= 200
  Set Variable:"Savings"= 5
  Set Variable:"Bills"= 500
  If (Paycheck + Savings >= Bills)
     Go To and Stop ("Happiness")
  Else
    Go To and Stop ("NotSoHappy")
  End If
End On
```

此脚本创建了三个变量，并为它们分配数值。 If命令检查Paycheck和Savings的和是否大于或等于Bills的值。如果是，时间线将跳转到标记为 Happiness的帧；如果不是，则跳转到标记为NotSoHappy的帧。

使用数字运算符时需注意的几个事项是：

- 表达式的运算按顺序执行，即按先后顺序执行。括号中的所有内容最先计算，然后进行乘除运算，最后才进行加减运算。请记住：运算的顺序不同，最后的结果也不一样。
- 如果你试图用数字运算符来计算字符串，那么 Flash会将字符串转换为数值。数值以字符串中的字符数为基础。例如， 10 + Hello = 15，因为Hello包含5个字符。

提示 有关文本处理的详细信息，请参见本章 11.3.6节“ 使用函数 ”中所讨论的 Int和Random函数。

11.3.4 字符串运算符

当计算字符串或任何值为文本的变量时，字符串运算符所执行的任务都是连接和比较字符串的值。

表11-4提供一列字符串运算符及其功能。

表11-4 字符串运算符

运 算 符	描 述
“ ”	字符串
&	连接符
eq	等于
ne	不等于
lt	小于
gt	大于
le	小于或等于
ge	大于或等于

1. 脚本范例

想象舞台上有 4个文本字段，变量名分别为First、Last、Age和Message。用户分别在前3个字段中键入John、Doe和30。当用户按某个按钮时，将执行以下脚本：

```
On (Release)
  Set Variable:"Message"="Hello,"& First & Last &".
  You appear to be "& Age & "year old."
End On
```

当执行此脚本时，变量Message显示以下信息：

```
"Hello,John Doe.You appear to be 30 years old."
```

在表达式中，变量First、Last和Age的值与带引号的文本值相连。

现在看另一个脚本。

想象舞台上有一个变量名为Password的文本字段。某人已在该字段中输入文本Boom Bam。当该用户按某个按钮时，将执行以下脚本：

```
On (Release)
  If (Password eq "Boom Bam")
    Go To and Stop ("Accepted")
  Else
    Go To and Stop ("AccessDenied")
  End If
End On
```

当执行此脚本时 (以Password文本框中所输入的信息为基础)，时间线将跳转至标记为Accepted的帧，并停在该帧。

请注意字符串运算符eq，它不同于等号(等号是一个数字运算符)，而是判断某个字符串是否等于另一个字符串。如果使用等号，脚本将无法正确判断。

用于文本的比较运算符，如lt、gt、le和ge，可根据第一个字符确定单词的字母顺序。小写字符(a到z)的值大于大写字符(A到Z)的值。例如， derek大于Brooks，而Derek则小于Brooks。

使用字符串运算符时，需注意以下事项：

- 字符串的值区分大小写：kathy不等于Kathy。
- 在字符串表达式中使用数值会使数值自动转换为字符串。例如，表达式“ I love to eat ”& 10 + 5 &“ donuts a day! ”会转换为“ I love to eat 15 donuts a day! ”。

提示 有关文本处理的详细信息，请参见本章 11.3.6节“使用函数”中所讨论的Substring、Length、Chr和Ord函数。

11.3.5 逻辑运算符

在表达式中使用逻辑运算符来判断某个条件是否存在。逻辑运算符主要用在If and Loop While动作中。

表11-5提供一列逻辑运算符及其功能。

表11-5 逻辑运算符

运 算 符	描 述
and	逻辑与
or	逻辑或
not	逻辑否

1. 脚本范例

```
On (Release)
  Set Variable:"Paycheck"= 1000
  Set Variable:"Decision"= "Buy"
  If (Paycheck >= 1000 and Decision eq "Buy")
     Go To and Stop ("NewComputer")
  Else
    Go To and Stop ("Cry")
  End If
End On
```

此脚本中的表达式检查Paycheck的数值是否等于或大于1000，以及Decision的字符串值是否等于Buy。如果是，便购买一台新的计算机；否则，便只好哭泣了！在此脚本中， If判断为True，所以该是买一台新计算机的时候了！

你会发现，如果在表达式中使用逻辑运算符，那么你可以同时判断数值和字符串值，并采取相应的动作。

11.3.6 使用函数

函数内置在ActionScripting功能中，可从Expression Editor获得，用来创建和检索表达式中的动态数据。我们将了解每个函数及其脚本范例。

下面的脚本绝不能说明函数的强大功能，仅仅举例说明函数的使用方法。

1. Eval

语法：Eval(变量名)

Eval函数可用来确定本身是表达式的某个变量的值。它允许在电影放映期间确定所求出的变量名。此函数的概念可能有些难懂，现在让我们看看下面的脚本。

```
On (Release)
  If (Eval ("GamePiece"& Number) = 50
  Actions.....
End On
```

此脚本与下面的脚本具有同样的功能：

```
On (Release)
```

```
  If (Gamepiece7) = 50
  Actions.....
End On
```

或者也可以是这样的脚本：

```
On (Release)
  If (Gamepiece2) = 50
  Actions.....
End On
```

它们之间的主要不同在于，第一个脚本中的变量 Gamepiece是动态的，也就是说所检查的游戏段(game piece)是基于变量Number的当前值；如果该值改变，那么所计算出来的游戏段也相应改变。

2. True

语法：True

此函数将逻辑值True分配给某个变量。

将以下脚本放置在某个按钮上，以便将它设置成已单击的情况：

```
On (Release)
  Set Variable:"Answer1Value"= True
End On
```

然后在另一个按钮上检查这个值是 True还是False，从而采取相应的动作，如以下脚本所示：

```
On (Release)
  If (Answer1Value)
    Go To and Play ("Correct")
  Else
    Go To and Play ("Wrong")
  End If
End On
```

提示 以上脚本中的If语句虽然为“ If(Answer1Value) ”，但实际是“ If (Answer1Value = True) ”的简写形式。

3. False

语法：False

此函数将逻辑值False分配给某个变量。具体例子，请看上面 True函数的脚本。

4. Newline

语法：Newline

此函数表达一个新的行。

想象舞台上有一个变量名为 Name的文本字段，用户在该字段中输入单词 Jim。当用户单击某个按钮时，将执行以下脚本：

```
On (Release)
  Set Variable:"Greeting"="Hello there,"
  Set Variable:"Phrase"= Greeting & Newline & Name &"."& Newline
  & "How are you today?"
End On
```

此脚本以Name文本字段中所输入的信息为基础，执行后，另一个文本字段 Phrase将显示

以下信息：

```
"Hello there,
Jim.
How are you today?"
```

5. GetTimer

语法：GetTimer

GetTimer函数用于确定电影已放映的时间 (以毫秒计)。该值以计算机上的系统时钟为基础，并且如果电影的放映速率 (每秒帧)减慢，该值也不受影响。同时在 Flash电影窗口放映的各部电影不具有单独的计时器；它是一个全局计时器。

使用GetTimer函数跟踪电影事件之间的时间间隔。

以下脚本创建了一个按钮的双击动作：

```
On (Release)
  If (GetTimer - LastClick < 500)
     Go to and Stop (10)
  End If
  Set Variable:"LastClick"= GetTimer
End On
```

6. Int

语法：Int(数字)

此函数提取数值的整数部分。例如， Int(43.364)的值为43。如果某个变量的值是数字，你就可以用变量名，而不用数字，例如 Int(VariableName)。

使用Int函数消除ActionScript中的小数部分。

以下脚本中，Total等于38：

```
On (Release)
  Set Variable:"FirstNumber"= 19.35
  Set Variable:"SecondNumber" = 2
  Set Variable:"Total"= Int (FirstNumber) * SecondNumber
End On
```

7. Random

语法：Random (数字)

此函数产生某个指定的范围之内的随机数值。例如， Random(300)将产生一个0到299之间的随机数。

使用Random函数创建不可预测的动态行为。如果某个变量的值是数字，你就可以用此变量名，而不用数字，如Random(VariableName)。

在以下脚本中，电影剪辑实例 Dice将根据随机产生的数值，跳转至 0和5之间的某一帧。

```
On (Release)
  Set Variable:"DiceRoll"= Random(6)
  Tell Target ("/Dice")
    Go To and Stop (DiceRoll)
  End Tell Target
End On
```

8. Substring

语法：Substring(字符串，索引，要包括的字符)

此函数的功能是提取字符串的一部分。字符串参数表示要从中提取的字符串，索引表示提取的第一个字符的位置(从字符串左边算起)。要包括的字符表示提取的字符数。

例如，Substring (“Macromedia”, 4, 4)的值等于rome。Macromedia是字符串；第4个字符是r，从索引开始的4个字符是rome。如果忽略第三个参数，则包括索引之后的所有字符。因此，Substring (“Macromedia”, 4)应等于romedia。

如果某个变量的值是字符串，那么你可以使用变量名，而无需使用字符串，例如Substring(变量名，索引，要包括的字符串)。

你可以用Substring函数独立出字符串的某部分，以便单独考虑这一部分。

想象舞台上有一个变量名为Title的文本字段。用户在该字段中输入文本Dr.Frankenstein，然后按某个按钮，将执行以下脚本：

```
On (Release)
  If (Substring(Title, 1, 3) eq"Ms.")
     Go To and Stop ("DivorceCourt")
  Else If (Substring(Title, 1, 3) eq"Dr.")
    Go To and Stop ("TheLab")
  Else
    Go To and Stop ("HelloJunior")
End On
```

当根据文本字段Title中所输入的信息为基础，执行此脚本时，电影将跳转至标记为Lab的帧。

9. Length

语法：Length (字符串)

Length函数创建一个基于字符串中字符数的数值。字符串参数标识要求算的字符串。例如，Length(“Flash”)等于数值5，因为单词Flash有5个字符。如果某个变量的值是字符串，则可以使用变量名，而不用字符串，例如Length(变量名)。

使用Length函数可以轻易地求得字符串的长度。例如，检验某个数据是否具有指定的位数，如ZIP编码和电话号码。

想象舞台上有某个变量名为ZIPCode的文本字段，用户已在该字段中输入文本46293。当用户按某个按钮时，将执行以下脚本：

```
On (Release)
  If (Length(ZIPCode) = 5
     Set Variable:"Message"="That is a valid ZIP Code"
  Else
    Set Variable:"Message"="Please enter a valid ZIP Code."
  End If
End On
```

执行完此脚本后，根据ZIPCode文本字段中所输入的信息，另一个文本字段Message将显示字符串“That is a valid ZIP Code.”

10. Chr

语法：Chr (数字)

此函数将数值转换为ASCII字符。例如，Chr(90)等于Z。如果某个变量的值为数字，则可以使用变量名，而不用数字，例如Chr(变量名)。

使用Chr函数可将数字分配给字符串值。

11. Ord

语法：Ord(字符)

此函数将ASCII字符转换为数字。例如，Ord(“D”)等于68。如果某个变量的值为字符串，则可以使用变量名，而不用字符串，例如 Ord(变量名)。

使用Ord函数可将数值分配给字符串。

12. Properties

Flash Player窗口中的每部电影都有一组独有的属性，它们被不断更新和运算，以使得ActionScript可以根据电影属性的当前值做出动态决定。

属性可从Expression Editor获得。

13. GetProperty

语法：GetProperty(目标, 属性)

GetProperty函数提供电影属性的当前值。仅使用此函数检索非当前电影的属性值。例如，GetProperty (“/MyMovieClip”, _alpha)将返回电影剪辑MyMovieClip的透明度。如果变量的值为字符串，则使用变量名而不是字符串文字来设置要计算的目标或属性，例如GetProperty(变量名，变量名2)。

GetProperty函数可计算任何电影剪辑的当前属性，以使电影采取相应的动作。

以下脚本计算MyMovieClip的高度，并采取相应的动作。

```
On (Release)
  If (GetProperty ("/MyMovieClip", _height)<300)
     Set Variable:"Message"="Thata pretty small movie clip"
  Else
    Set Variable:"Message"="That movie clip is way too big!"
  End If
End On
```

执行完此脚本后，且假若计算出来的 MyMovieClip的高度大于300，文本字段Message将显示字符串“That movie clip is way too big!”

14. _x

语法：_x，或者GetProperty,(目标，_x)

x属性提供电影剪辑实例的当前水平位置。值以对象监控板中的像素为基础，是电影剪辑实例的左上角相对于父电影剪辑实例左上角的位置 (参见本章前面的文字框“父子关系”)。

选择某个电影剪辑，然后在对象监控板中选择 Use Center Point，此时电影剪辑的x和y位置均是以电影剪辑中心到父电影剪辑中心的距离为基础的相对值 (参见本章前面所讨论的 Set Property动作)。

以下脚本计算当前电影的当前水平位置，并采取相应的动作：

```
On (Release)
  If (_x < 200)
    Set Variable: "Message"="Ion the left."
  Else If (_x > 200)
    Set Variable:"Message" ="Ion the right."
  Else
    Set Variable:"Message"="Istuck in the middle somewhere."
```

```
    End If
End On
```

当执行此脚本时，当前电影的水平位置被计算出恰好等于 200；因此，变量名为Message的文本字段将显示字符串“ I ’m stuck in the middle somewhere. ”。

15. _y

语法：_y，或者GetProperty,(目标， _y)

y属性提供某部电影的当前垂直位置。值用像素表示，并以父电影剪辑实例的中心为参照点(请参见文字框“ 父子关系 ”)。

选择某个电影剪辑，然后选择对象监控板中的 Use Center Point，此时电影剪辑的x和y位置均是以电影剪辑中心到父电影剪辑中心的距离为基础的相对值。具体例子，请参见 _x函数中的脚本。

16. _width

语法：_width，或者GetProperty,(目标， _width)

宽度属性提供电影的当前宽度，单位是像素。以下脚本计算当前电影的当前宽度，并采取相应的动作。

```
On (Release)
Set Variable:"NextMeal"= 50
  If (_width + NextMeal >= 400)
    Set Variable:"Message"=‘I too fat"
  Else If (_width + NextMeal <= 100)
    Set Variable:"Message"=’‘Itoo shinny."
  Else
    Set Variable:"Message"=‘I just right."
  End If
End On
```

执行过程中，此脚本创建一个变量 NextMeal，并为它赋值50。确定当前电影的宽度，并加上NextMeal的值。然后计算两个值的和，以确定是大于或等于 400，小于或等于100，还是两者之间的值。本例中，电影的宽度被确定为 230，加上50即为280。因此，变量名为Message的文本字段将显示字符串“ I'm just right ”。

17. _height

语法：_height，或者GetProperty,(目标， _height)

高度属性提供电影的当前高度，单位是像素。具体例子，请参见宽度属性的脚本范例。

18. _rotation

语法：_rotation，或者GetProperty,(目标， _rotation)

旋转属性提供电影的旋转程度。单位是度数，是相对于父电影的旋转程度。

以下脚本由On (Release)鼠标事件触发。触发时，为变量Spin产生一个0到359之间的随机数值。该值用来设置电影剪辑MyMovieClip旋转的度数。同时，判断Spin的值：如果在0到45之间，则在变量名为Message的文本字段中显示“ You spun a 1,you win! ”。否则，显示“ Try Again ”。

```
On (Release)
  Set Variable"Spin"= Random (360)
  Set Property ("/MyMovieClip", Rotation) = Spin
  If (Spin >= 0 and < 45)
```

```
    Set Variable:"Message"="You spun a 1, you win!"
  Else
    Set Variable:"Message"="Try again."
  End If
End On
```

19. _target

语法：_target，或者GetProperty,(目标，_target)

目标属性以字符串值的形式提供电影剪辑的目标名称和完整路径，如/MainClip/MyMovieClip。

以下脚本的功能是，当On (Press)事件发生时，可拖动当前电影，而当On (Release)事件发生时，则禁止拖动当前电影。

```
On (Press)
  Set Variable: "MovieToDrag"= _target
  Start Drag (MovieToDrag)
End On
On (Release)
  Stop Drag
End On
```

提示 可将此脚本放置在某个电影剪辑实例的按钮中，以使得当按下此按钮时，该电影剪辑实例可拖动。

20. _name

语法：_name，或者GetProperty,(目标，_name)

名称属性以字符串值的形式提供电影剪辑的名称，如 MyMovieClip。它与目标属性类似，但不包括完整的路径。

以下脚本求得当On (Release)鼠标事件发生时当前电影剪辑的名称，然后将该名称输出到变量名为MovieName的文本字段：

```
On (Release)
  Set Variable"MovieName"= _name
End On
```

脚本运行的结果是一个显示电影名称的文本字段 (例如，MyMovieClip)。

21. _url

语法：_url，或者GetProperty,(目标，_url)

此属性为.swf或它的任何子电影剪辑提供完整的 URL。它通常与已用Load/Unload Movie动作加载到Flash Player窗口的.swf结合使用。例如，如果一个.swf从URL http://www.mydomain.com/secondmovie.swf加载到Flash Player窗口，选择此电影的url属性将返回一个字符串值http://www.mydomain.com/secondmovie.swf。

使用此属性可确保没有其他人“借用”你的工作。可以将一个脚本放置在某部电影的第 1帧，以检查它的url属性，并且如果它是从“正确的”URL加载，将执行某个动作，而如果它是从“错误的”URL加载，则执行另一个动作。通过这种办法，可以防止其他人从他们的高速缓存器偷窃.swf文件并将它用作自己的.swf。以下脚本放置在电影的第 1帧，它判断当前电影的url属性。如果URL正确，则继续放映；否则，将停止放映并跳转至 (并停在)标记为Denied的帧：

```
If (Substring(_url, 1, 23) eq "http://www.myserver.com")
  Play
Else
  Go To and Stop ("Denied")
End If
```

在以上脚本中，我们只需了解 URL http://www.myserver.com的第一部分是否正确。这就是为什么使用Substring函数。它提取返回的头23个字符(URL长度的计算可能有些困难)，并查看它是否等于应等于的字符串。如果是，将放映电影。如果不是，将跳转至某一具有诸如Access Denied信息的帧，并停止放映该帧。

22. _xscale

语法：_xscale，或者GetProperty，(目标，_xscale)

此属性提供电影或电影剪辑在水平方向上相对于原大小的缩放比，原大小是在前面的 Set Property动作中通过X Scale属性设定的。

当On (Release)鼠标事件发生时，以下脚本判断电影剪辑 /MyMovieClip相对于原大小的缩放比例。如果大于100%，便重新设置为100%；否则便保持它当前的大小。

```
On (Release)
  If (GetProperty ("/MyMovieClip",_xscale) > 100)
    Set Property ("/MyMovieClip", X Scale) = 100
  End If
End On
```

23. _yscale

语法：_yscale，或者GetProperty，(目标，_yscale)

此属性提供电影或电影剪辑在垂直方向上相对于原大小的缩放比，原大小是在前面的 Set Property动作中通过yscale属性设定的。

具体例子，请参见xscale的脚本范例。

24. _currentframe

语法：_currentframe，或者GetProperty，(目标，_currentframe)

currentframe属性为电影或电影剪辑提供时间线上的当前帧编号位置。

当On (Release)鼠标事件发生时，以下脚本将当前电影的时间线从它的当前位置向前发送20帧。

```
On (Release)
  Go To and Stop (_currentframe + 20)
End On
```

25. _totalframes

语法：_totalframes，或者GetProperty，(目标，_totalframes)

totalframes属性提供电影或电影剪辑中帧的数量。

当On (Release)鼠标事件发生时，以下脚本将创建变量 TimeToPlay。该变量的值是用电影的帧数除以放映速率 (帧/每秒)所得来的。 Int函数用来删除计算中的所有小数位。下一个变量Message是一个文本字段，它显示以TimeToPlay的值为基础的信息：

```
On (Release)
  Set Variable:"TimeToPlay"= Int(_totalframes / 12)
  Set Variable:"Message"="This movie will take" & TimeToPlay" &"Seconds to play.
```

```
End On
```

如果电影有240帧，文本字段Message将显示以下信息：

```
"This movie will take 20 seconds to play."
```

26. _framesloaded

语法：_framesloaded，或者GetProperty，(目标，_framesloaded)

framesloaded属性提供电影已加载的帧数。此属性与 If Frame is Loaded命令类似；但是，它允许返回的数字用在表达式中。

以下脚本放置在时间线上的第 2帧。如果加载的帧数超过 200，时间线将跳转至场景 2的第 1帧，并从该处开始放映。否则，时间线将跳转至当前场景的第 1帧，并从该处开始放映。直到加载的帧数超过200，循环才被打破。

```
If (_framesloaded > 200)
  Go to and Play (Scene 2, 1)
Else
  Go to and Play (1)
End If
```

27. _alpha

语法：_alpha，或者GetProperty，(目标，_alpha)

alpha属性提供电影或电影剪辑的透明度 (用百分比表示)。以下脚本判断电影剪辑“ dress ”的透明度。如果超过 50%，便在跳转至DanceParty之前将WearSlip?设置为True。否则，直接跳转至DanceParty。

```
If (GetProperty ("/Dress",_alpha) > 50)
  Set Variable:"/Dress:WearSlip?"= True)
  Go To and Play ("DanceParty")
Else
  Go To and Play ("DanceParty")
End If
```

28. _visible

语法：_visible，或者GetProperty，(目标，_visible)

visible属性返回逻辑值True或False：如果可见，便为True，如果不可见，则为False。

以下脚本由帧事件触发，它检查电影剪辑 Teacher的可见性。如果不可见，电影剪辑 Kids便被发送至标记为“ Recess ”的帧；否则， Kids便被发送至“ Desk ”：

```
If (GetProperty ("/Teacher",_visible) = False)
  Tell Target ("/kids")
    Go To and Play ("Recess")
  End Tell Target
Else
  Tell Target ("/kids")
    Go To and Stop ("Desk")
  End Tell Target
End If
```

29. _droptarget

语法：_droptarget，或者GetProperty，(目标，_droptarget)

droptarget属性返回某个电影的目标路径，该电影在当前拖动的电影的下面。此属性允许

你模拟拖放行为。参见图11-45。

以下脚本模拟拖放行为。当鼠标事件On (Press)发生时，电影剪辑MyMovieClip拖动，并以鼠标的位置为中心。当On (Release Outside)鼠标事件发生时，拖动停止，并用表达式计算电影剪辑的位置。拖动的电影剪辑下面的电影剪辑就是droptarget。在该脚本中，当拖动操作停止时，标识出droptarget。如果它等于MyTarget，MyMovieClip将不可见；否则，便什么也不会发生。

```
On (Press)
  Start Drag ("/MyMovieClip", lockcenter)
End On
  On (Release Outside)
  Stop Drag
    If (GetProperty ("/MyMovieClip", _droptarget) eq "/MyTarget")
    Set Property ("/MyMovieClip", Visibility) = False
  End If
End On
```

30. _highquality

语法：_highquality

highquality属性根据当前重放的质量设置，返回数值0、1或2。它是一个全局属性，因此，适用于当前在Flash Player窗口中放映的所有电影。

```
If (_highquality = 2)
  Actions...
Else
  Actions...
```

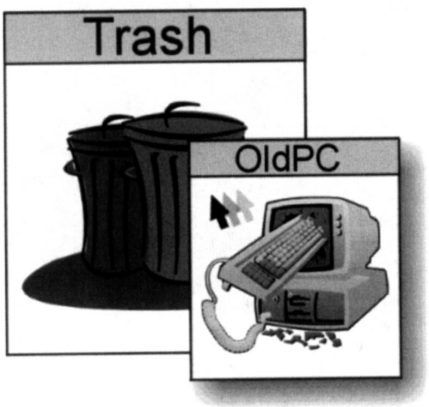

图11-45 droptarget属性是当前在被拖动电影下面的电影的目标路径。本图中，/oldPC的droptarget是/Trash。该属性被不断更新

31. _focusrect

语法：_focusrect

focusrect属性根据矩形焦点是打开还是关闭(参见本章前面的Set Property动作)，返回逻辑值True或False。它是一个全局属性，适用于当前在Flash Player窗口中放映的所有电影。

```
If (_focusrect = True)
  Actions...
Else
  Actions...
```

32. _soundbuftime

语法：_soundbuftime

soundbuftime属性返回当前的声音缓冲时间设置值。默认设置是 5(5秒)。它是一个全局属性，适用于当前在Flash Player窗口中放映的所有电影。

```
If (_soundbuftime > 15)
  Actions...
Else
  Actions...
```

33. 特殊文本属性

因为文本字段可以包含多行文本，所以可以在其他的事物中用两个独特属性为它们创建

滚动栏。

34. Scroll

语法：变量名.scroll

scroll属性是一个数值，它表示当前显示在文本字段中的最上面的一个可见行的行编号。因此，如果文本字段有10个文本行，且用户已滚动到当前最上面的可见行第 4行，那么，此文本字段的scroll值应为4。电影放映期间，该值不断更新。它可以用表达式计算，或者你也可以创建按钮，以使最上一行跳至你需要的位置 (请看下面的脚本)。

以下脚本展示了一个帧事件，它用表达式判断文本字段 MyTextBox的当前Scroll属性。如果该值大于3，电影将停止放映。

```
If (MyTextBox.scroll > 3)
  Stop
End If
```

以下脚本展示了一个鼠标事件，它使得文本字段 MyTextBox中的第6行成为最上面的可见行。

```
On (Release)
  Set Variable:"MyTextBox.scroll"= 6
End On
```

35. maxscroll

语法：变量名.maxscroll

maxscroll属性是一个数值，它表示文本字段中最上面的可滚动行的行编号。如果你的某个文本字段的高度只能显示两行文本，尽管它实际包含 5行，但maxscroll的值也只能是4。这是因为在它的最高滚动点，第 4行是最上面的一个可见行。如果一个文本字段包含 10个文本行，但是一次只能显示 4行，那么此字段的maxscroll值将为7。这是因为在它的最高滚动点，第 7行是最上面的可见行(见图11-46)。

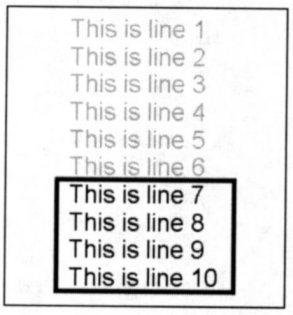

图11-46 一次只能显示4行的10行文本字段的maxscroll值为7

显然，maxscroll值不是你设置的。它由文本字段的文本行数和它一次所能显示的行数所决定。该值可以用在按钮实例的表达式中以创建循环滚动。也就是说，当单击按钮时，文本字段中的文本将滚动，但是当文本到达字段的最底部时，滚动将自动回到顶部(请参见下面的脚本)。

以下脚本使文本字段MyTextBox跳转至以Count的值为编号的行，每单击按钮一次，该值便更新一次，且将 Count的值与MyTextBox.maxscroll的值进行比较。一旦 Count的值等于MyTextBox.maxscroll的值,便到达滚动的末尾，同时重新设置 Count的值，以便当再次单击按钮时，该文本字段将再次显示第 1行，并重新开始整个过程：

```
On (Release)
  Set Variable:"Count"= Count + 1
  Set Variable:"MyTextBox.scroll"= Count
  If (Count = MyTextBox.maxscroll)
    Set Variable:"Count"= 0
  End If
End On
```

11.3.7 Expression Editor

Expression Editor(表达式编辑器)提供一种将用在ActionScript中的表达式快速组合在一起的方法。用它可以访问前面所讨论的所有函数、属性和运算符。了解这些组合方式会让你受益非浅，而Expression Editor则将会为你提供向导。

要使用Expression Editor，应如下操作：

1) 单击参数字段的按钮，然后从出现的菜单中选择 Expression Editor(见图11-47)。

Expression Editor出现。

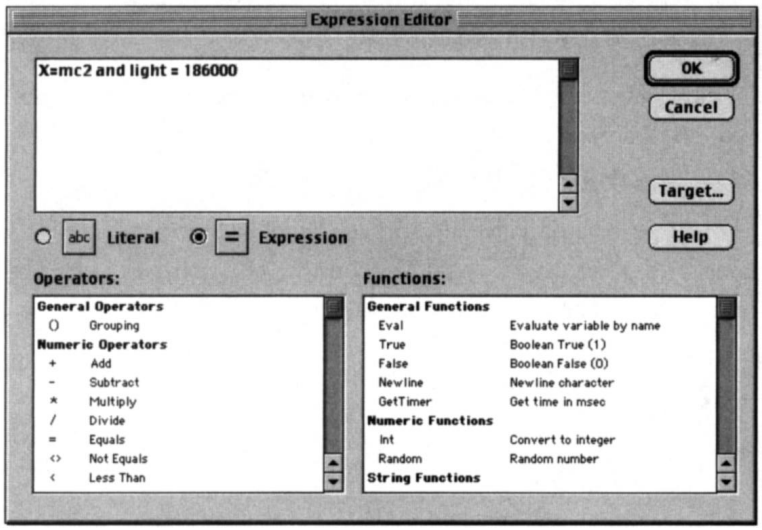

图11-47 Expression Editor可以快速轻松地创建表达式

2) 双击任何单独运算符或函数，将它们添加到表达式。

双击某些函数，例如Eval或GetProperty，将常用术语(如target、property或variable)放置在表达式中，以标识某些信息将放置的位置。

3) 如果你希望设置GetProperty函数的目标，选择“ target ”，然后单击Target按钮。

4) 双击列表中某个目标的名称以选择它。如果此名称未出现在列表中，则必须输入它。

5) 单击OK选择Target Editor。

6) 如果你希望设置GetProperty函数的属性，只需突出显示术语 property，然后从列表中选择属性。

7) 使用完Expression Editor后，单击OK。

提示 右击(Windows)表达式的文本区域，将在Expression Editor中出现标准编辑函数。

11.4 交互性教程

为帮助你综合学习这些概念，我们已编排了 4个交互性教程。你可以在光盘中找到这些交互性教程以及它们的源文件：

Emulating Drag-and-Drop Behavior(模拟拖放行为)：此教程向你展示了如何在 Flash电影

中模拟拖放行为。

Creating a Mouse Tracker(创建鼠标跟踪器)：此教程向你展示了如何创建包含鼠标跟踪器的电影，鼠标跟踪器是一个简单的设备，可不断显示用户光标在几个文本字段中的当前 x和y位置。

Whack a Mole：没有与此Flash项目相关的视频教程。设置此教程只是为了帮助你了解在Flash中组合交互性游戏背后的一些深层原理，它使用你在本章中已学过的技术。