第3章 ActionScript 3.0程序控制

众所周知，Flash中动画依靠的是时间轴，在没有脚本的情况下，动画会依照时间轴从第一帧不停地播放到最后一帧，然后重复播放或者停止。为了能更好地控制动画，就必须使用脚本语句。而要想使动画具有逻辑判断的功能，就要使用流程控制语句了。

一般情况下，Flash执行动作脚本从第一条语句开始，然后按顺序执行，直至达到最后的语句。这种按照语句排列方式逐句执行的方式，称为顺序结构。顺序结构是程序中使用最多的程序结构，但用顺序结构只能编写一些简单的动作脚本，解决一些简单的问题。

在实际应用中，往往有一些需要根据条件来判断结果的问题，条件成立是一种结果，条件不成立又是一种结果。像这样复杂问题的解决就必须用程序的控制结构，控制结构在程序设计中占有相当重要的地位，通过控制结构可以控制动作脚本的流向，完成不同的任务。

3.1 选择结构

选择结构在程序中以条件判断来表现，根据条件判断结果执行不同的动作。ActionScript 3.0有3个可用来控制程序流的基本条件语句。其分别为if条件语句、if-else条件语句、switch条件语句。本节将详细讲解这三种不同的选择程序结构。

3.1.1 if条件语句

```
if (条件表达式){
//条件成立的情况下,执行{}中的语句,否则跳过{}执行后面的语句
}
```

例如：

```
var a:unit = 12;
if (a % 2 == 0)
  trace(a,"是偶数.");
```

在条件表达式中，取模运算符“%”的优先级高于“==”，所以先计算“a%2”，然后判断它的值是否等于0。当条件为真时执行trace()语句输出相关信息，由于需要执行的代码只有一行，所以可以不用大括号，如果需要执行的代码只有一个语句段，必须要把所有需要执行的语句都放在一对大括号内。

3.1.2 if-else 语句

if- else 语句在简单 if 语句的基础上增加了一个程序分支，这种语句的一般形式为：

```
if (条件表达式) {
① …
}//条件成立,执行①内的语句
else {
②      …
}//条件不成立,执行②内的语句
```

例如：

```
var a:unit = 12;
if (a % 2 == 0)
  trace(a,"是偶数.");
else
  trace(a,"是奇数.");
```

if-else 条件语句执行的操作最多只有两种选择，要是有更多的选择，那就可以使用 if-else if-else 条件语句。

```
if(表达式 1)
    语句 1;
else if (表达式 2)
    语句 2;
else if (表达式 3)
    语句 3;
⋮
else if (表达式 n)
    语句 n;
else
    语句 n + 1;
```

例如，按学生的分数 score 输出其等级：score≥90 为优，90＞score≥80 为良，80＞score≥70 为中等，70＞score≥60 为及格，score＜60 为不及格。

```
if (score >= 90)
    trace( "优" );
else if (score >= 80)
    trace( "良" );
else if (score >= 70)
    trace( "中" );
else if (score >= 60)
    trace( "及格" );
else
    trace( "不及格");
```

3.1.3 switch 语句

使用 if-else 语句可以对条件表达式的真和假两种情况分别处理，使程序产生两个不同

的分支，还可以使用 if 语句嵌套，实现多重判断。

switch 语句相当于一系列的 if-else if-else 语句，但是要比 if 语句要清晰得多。switch 语句不是对条件进行测试以获得布尔值，而是对表达式进行求值并使用计算结果来确定要执行的代码块。

switch 语句格式如下：

```
    switch (表达式) {
          case:
              程序语句 1;
            break;
          case:
              程序语句 2;
            break;
          case:
              程序语句 3;
            break;
          default:
              默认执行程序语句;
}
```

例如，用 switch 语句实现，按学生的分数 score 输出其等级的功能。

```
var  a :Number == score/10;
switch(a)
{  case  10:
  case  9:
    trace( "优" );
    break;
  case  8:
    trace( "良" );
    break;
  case  7:
    trace( "中" );
    break;
  case  6:
    trace( "及格" );
    break;
  default:
    trace( "不及格" );
}
```

3.2 循环结构

如果要多次执行相同的语句，可以利用循环语句简化程序。在 Flash ActionScript 3.0 中有 3 种循环语句：for 语句、for-in 语句和 while 语句。

3.2.1 for 语句

for 语句通常用于循环次数固定的情况，它的基本形式为：

```
for(初始表达式; 条件表达式; 递增表达式)
{
      //循环执行的程序段(循环体)
}
```

下面是一个用 for 语句编写的一个简单程序,计算 1+2+3+…+98+99+100 的值。

```
var s:int = 0;
var i:int,h:String;
for (i = 1;i <= 100;i++){
  s = s + i;
}
h = "s = " + s;
trace (h);
```

运行结果:s=5050

3.2.2 for-in 和 for each-in 语句

for-in 和 for each-in 语句都可以和数组以及对象数据类型一起使用。使用此语句可以在不知道数据里面有多少个元素或元素一直在变化的情况下遍历所有的数组元素。

```
for (变量名 in 数组名或对象数据类型)
{
  //程序段,可以用变量作为下标引用数组元素或用变量作为属性名引用属性值
}
```

例如下面的语句将数值 myArr 中的元素显示出来。

```
var myArr:Array = [1,2,3,4,5,6,7,8,9,10];
for(var i:String in myArr) {
    trace(myArr[i]);              //直接用变量 i 作为数值元素的下标
}
```

在使用 for-in 时,变量的类型必须为 String 类型,如果声明为 Number 等其他类型,将不能正确输出。

下面分别使用两种语句来访问对象中的属性,代码如下所示:

```
//定义一个对象 lzxt,并添加属性 name 和 age
var lzxt:Object = {name:"浪子啸天", age:30};
//执行遍历操作
for (var i:String in lzxt) {
     //输出属性名称和属性值
     trace("for in 语句输出: " + i + ": " + lzxt[i]);
}
//执行 for each 遍历操作
for each (var k:String in lzxt) {
     //输出属性值
     trace("for each 语句输出: " + k);
}
```

3.2.3 while 语句

while 循环在条件成立的时候，一直循环到条件不成立。

```
while(条件表达式)
{
⋮
}//条件为真时,执行{}中的语句,在循环过程中,也可以使用 break 语句跳出循环
```

例如下面的语句用来计算 1～10 范围内所有自然数的乘积。

```
var i:uint = 1,mul:uint = 1;
while (i<=10) {
    mul*=i;
    i++;
}
trace(mul);
```

3.2.4 循环的嵌套

循环的嵌套就是在一个循环的循环体中存在另一个循环体，如此重复下去直到循环结束为止，即为循环中的循环。以 for 循环为例，格式如下所示：

```
for (初始化; 循环条件; 步进语句) {
    for (初始化; 循环条件; 步进语句) {
    循环执行的语句;
    }
}
```

3.2.5 break 和 continue 语句

在 ActionScript 3.0 中可以使用 break 和 continue 来控制循环流程。break 语句的结果是直接跳出循环，不再执行后面的语句；continue 语句的结果是停止当前这一轮的循环，直接跳到下一轮的循环，而当前轮次中 continue 后面的语句也不再执行。

下面的两个例子分别执行循环变量从 0 递增到 10 的过程，如果 i 等于 3，分别执行 break 和 continue 语句，看发生的情况，代码如下所示：

```
//使用 break 控制循环
for (var i:int = 0; i<10; i++) {
    if (i==3) {
        break;                    //或 continue 语句
    }
    trace("当前数字是:" + i);
}
```

3.3 影片剪辑的控制

影片剪辑元件是 Flash 中最重要的一种元件，对影片剪辑属性的控制是 ActionScript 的最重要功能之一。从根本上说，Flash 的许多复杂动画效果和交互功能都与影片剪辑属

性控制的运用密不可分。

3.3.1 影片剪辑元件基本属性

1. 坐标

Flash 场景中的每个对象都有它的坐标，坐标值以像素为单位。Flash 场景的左上角为坐标原点，它的坐标位置为(0,0)，前一个表示水平坐标，后一个表示垂直坐标。Flash 默认的场景大小为 550×400 像素，即场景右下角的坐标为(550,400)，场景中的每一点分别用 x 和 y 表示 x 坐标值属性和 y 坐标值属性，如图 3-1 所示。

例如，要在主时间轴上表示场景中的影片剪辑 my_mc 的位置属性，可以使用下面的方法：

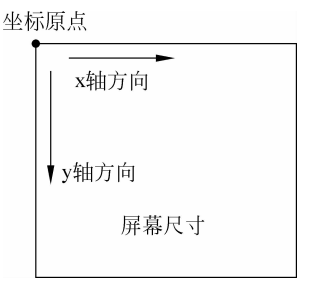

图 3-1 Flash 场景坐标

```
my_mc.x        my_mc.y
```

通过更改 x 和 y 属性可以在影片播放时改变影片剪辑的位置。例如为影片剪辑对象 my_mc 编写如下的事件处理函数，在每次 enterFrame 事件中向右移动两个像素、向下移动一个像素的位置：

```
stage.addEventListener(Event.ENTER_FRAME,moveBall);
function moveBall(event:Event){
my_mc.x += 2;
my_mc.y += 1;
}
```

影片剪辑对象的坐标中心点(对象的 x,y 坐标点)默认情况下，是在对象外接矩形的左上角。一个五角星的坐标中心点如图 3-2 所示。

坐标中心点要与变形中心点区别开来。

坐标中心点：即注册点。是对象的 x,y 坐标点，默认为对象的左上角。是元件坐标系中(0,0)原点的位置(双击元件后屏幕中小十字位置，如图 3-2 所示)，在进行“转换为元件”操作时会让你选择注册点，选左上角元件坐标系中原点的位置就在图形左上角，以此类推，如图 3-3 所示。

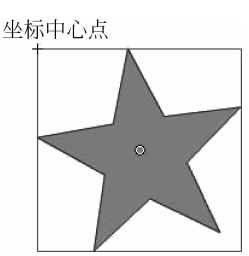

图 3-2 坐标中心点

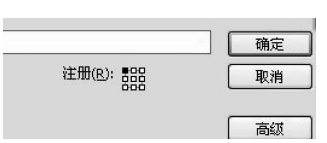

图 3-3 修改注册点位置

变形中心点：对象进行放大、缩小、旋转能变形时的参考中心点，默认为几何中心，可以调整使之位于对象的任何位置。如图 3-2 所示，在使用自由变形工具时，对象内部的一个空心圆即为其变形中心，可以用鼠标拖动这个中心点。

注意：坐标原点分为两种情况。

(1) 如果可视对象直接放置在舞台上，则以舞台的左上角作为坐标原点。

(2) 如果可视对象嵌套在某可视对象内，则以某可视对象的坐标中心点作为坐标原点。

也就是说，对象的坐标值是相对的概念。指相对于它的上层对象的中心点，它的坐标值是多少。

可视对象层次结构中的坐标具有嵌套的特点。事实上，不仅仅是坐标，像角度、透明度等可视对象属性都具有嵌套的特点。假设对象 a 中嵌套着对象 b，则 b 的属性与 a 的嵌套关系如表 3-1 所示。

表 3-1　嵌套后子对象 b 属性的计算

属　　性	嵌 套 关 系
不透明度(alpha)	全局不透明度＝a. alpha×b. alpha
高度(height)	全局高度＝a. scaleY×b. height
角度(rotation)	全局角度＝a. rotation＋b. rotation
宽度(width)	全局宽度＝a. scaleX×b. width
x 坐标(x)	全局 x 坐标＝a. x＋b. x×a. scaleX
y 坐标(y)	全局 y 坐标＝a. y＋b. y×a. scaleY

理解这一嵌套关系，对 Flash 游戏编程人员来说非常重要。

2. 尺寸控制

尺寸控制有两种方法：一种是“绝对尺寸控制”，就是设定对象的高度和宽度是多少像素，另一种是“相对尺寸控制”，就是设定对象的高度和宽度是原来的多少倍。

绝对尺寸控制一般使用属性“width”设置影片剪辑元件的绝对宽度，“height”设置影片剪辑的绝对高度，单位都是像素，并且值都不能为负。

例如，设置影片剪辑对象 my_mc 高度和宽度均为 200 像素。

```
my_mc.x = 200;
my_mc.y = 200;
```

相对尺寸控制是指在影片剪辑水平和垂直方向上进行缩放的倍数，属性“scaleX”和“scaleY”的值代表了相对于库中原影片剪辑的横向尺寸 width 和纵向尺寸 height 的百分比，而与场景中影片剪辑实例的尺寸无关。

scaleX 和 scaleY 属性值为数值，属性值 1 即缩放比例为 100%(原始大小)。影片剪辑元件缩放局部坐标系统会影响子影片剪辑元件所定义的 x 和 y 属性设置。例如，如果将父影片剪辑元件缩放 50%，则设置 x 属性会将影片剪辑中的对象移动缩放比例为 100%时影片的一半像素数目。影片剪辑元件缩放默认的注册点坐标位置是(0,0)。

例如，设置影片剪辑对象 my_mc 高度和宽度均放大 1 倍。

```
my_mc. scaleX = 2;
my_mc. scaleY = 2;
```

3. 鼠标位置

利用影片剪辑元件的属性，不但可以获得坐标位置，还可以获得鼠标位置，即鼠标光标

在影片中的坐标位置。表示鼠标光标的坐标属性的关键字是 mouseX 和 mouseY，其中，mouseX 代表光标的水平坐标位置，mouseY 代表光标的垂直坐标位置。需要说明的是，如果这两个关键字用在主时间轴中，则它们表示鼠标光标相对于主场景的坐标位置；如果这两个关键字用在影片剪辑中，则它们表示鼠标光标相对于该影片剪辑的坐标位置。mouseX 和 mouseY 属性都是从对象的坐标原点开始计算的，即在主时间轴中代表光标与左上角之间的距离；在影片剪辑中代表光标与影片剪辑中心之间的距离。Flash 不能获得超出影片播放边界的鼠标位置，这里的边界并不是指影片中设置的场景大小。如将场景大小设置为 550×400，在正常播放时能获得的鼠标位置即在(0,0)～(550,400)；如果缩放播放窗口，将视当前播放窗口的大小而定；如果进行全屏播放，则与显示器的像素尺寸有关。

4. 旋转方向

rotation 属性代表影片剪辑的旋转方向，它是一个角度值，介于－180°～180°，可以是整数和浮点数，如果将它的值设置在这个范围之外，系统会自动将其转换为这个范围之间的值。例如，将 rotation 的值设置为 181°，系统会将它转换为－179°；将 rotation 的值设置为－181°，系统会将它转换为 179°。不用担心 rotation 会超出它的范围，系统会自动将它的值转换到－180°～180°，并不会影响到影片剪辑转动的连贯性。

5. 可见性

visible 属性即可见性，使用布尔值，即为 true(1)，或者为 false(0)。为 true 表示影片剪辑可见，即显示影片剪辑；为 false 表示影片剪辑不可见，隐藏影片剪辑。

例如要隐藏影片剪辑 myMC：

```
myMC. visible = false;
```

3.3.2 控制影片剪辑元件的时间轴

影片剪辑元件时间轴的控制，与主时间轴的控制基本一致，包括播放、暂停、跳转等，例如：

```
my_mc.play();                    //播放
my_mc.stop();                    //暂停
my_mc.prevFrame();               //转到上一帧并暂停
my_mc.nextFrame();               //转到下一帧并暂停
my_mc.gotoAndPlay(n);            //跳转到第 n 帧,继续播放
my_mc.gotoAndStop(n);            //跳转到第 n 帧并暂停
```

对于影片剪辑元件，还有以下三个用来监视时间轴进程的只读属性。

1. currentFrame

通过 currentFrame 属性可取得在影片剪辑时间轴上播放头所在的帧编号，在做相对跳转的时候很有用，例如使用“快进”按钮的时候，希望每次单击按钮，可以让影片剪辑的播放头向前跳 20 帧，可以使用以下语句：

```
my_mc.gotoAndPlay(my_mc.currentFrame + 20);
```

2. framesLoaded

framesLoaded 属性会返回从 SWF 文件中加载的帧数目，该属性十分有用，在还没有完成某个 SWF 文件中指定帧的加载动作之前，可以显示一个消息，告诉用户该 SWF 文件正在加载中。例如：

```
totalFra = this. framesLoaded;
```

3. totalFrames

totalFrames 属性会返回在影片剪辑实体对象中的帧总数，该属性常与 framesLoaded 属性结合使用，可以告诉用户目前 SWF 文件的加载进度百分值(framesLoaded/totalFrames×100)。

假如设计“快进”按钮，如果在加载了 100 帧的时候播放到了第 90 帧，那么向前跳 20 帧，就要跳到 110 帧，由于此时第 110 帧还不存在，所以执行将失败，可以将代码完善如下：

```
 var t =   my_mc.currentFrame + 20;
 if ( t < my_mc.framesLoaded ) {
 my_mc.gotoAndPlay(t);
 }
else  {
    My_mc.gotoAndplay(my_mc.framesLoaded);
 }
```

3.3.3 复制与删除影片剪辑

1. 复制影片剪辑

在 ActionScript 2.0 中使用 duplicateMovieClip 方法来复制影片剪辑实例，在 ActionScript 3.0 中已取消这个方法，也就是无法直接附加对象到动画片段中，而必须先构造要复制对象的实例，然后再使用 addChild 或 addChildAt 复制该对象的实例。

(1) addChild()命令的语法结构为：

```
影片剪辑对象. addChild(对象)
```

addChild 可以将指定的对象加入到影片剪辑对象中，不限定加入的是何种对象。其中(对象)为对象变量名称，也就是使用 new 函数所创建的实体对象。

(2) addChildAt()命令的语法结构为：

```
影片剪辑对象. addChildAt(对象,叠放次序)
```

addChildAt 可以将指定的对象加入到影片剪辑对象中，并且可指定加入对象的叠放次序。其中(叠放次序)为整数，为附加对象时对象所要放置的层次。

```
myObj = new Object();
my_mc.addChildAt(myObj,3);
//新建对象"myObj"并复制影片剪辑"my_mc"到第 3 层中
```

2. 删除影片剪辑

使用 removeChild 或 removeChildAt 可以删除影片剪辑对象。

(1) removeChild()命令的语法结构为：

```
影片剪辑对象. removeChild(对象)
```

removeChild 可将已复制到影片剪辑中的对象删除，例如：

```
my_mc. removeChild(myObj);
//删除影片剪辑"my_mc"中的子对象"myObj"
```

(2) removeChildAt()命令的语法结构为：

```
影片剪辑对象. removeChildAt(叠放次序)
```

removeChildAt 可将指定叠放次序的复制影片剪辑对象删除，例如：

```
my_mc. removeChildAt(3);
//删除影片剪辑"my_mc" 中的迭放次序为 3 的子对象
```

3.3.4 拖曳影片剪辑

1. startDrag()

startDrag()命令的语法结构为：

```
影片剪辑对象. startDrag(锁定中心,拖曳区域)
```

startDrag 方法可以让用户拖曳指定的影片剪辑对象，直到 stopDrag()被调用后，或是直到其他影片剪辑对象可以拖曳为止，注意一次只能有一个影片剪辑对象为可拖曳状态。锁定中心为布尔值，指定可拖曳的影片剪辑对象要锁定于鼠标指针的中央(true)或是锁定在用户第一次按下影片剪辑对象的位置(false)，拖曳区域为矩形区域，指定影片剪辑对象拖曳的限制矩形区域。

例如：

```
my_mc.startDrag();
```

将影片剪辑对象“my_mc”进行拖曳动作。

```
this.startDrag(true);
```

将当前影片剪辑对象锁定中心点，并开始拖曳。

2. stopDrag()

startDrag()命令无参数，它可结束 startDrag 方法的调用，让用户拖曳的指定影片剪辑对象停止拖曳状态。例如停止影片剪辑对象“my_mc”的拖曳动作：

```
my_mc. startDrag();
```

3. dropTarget

dropTarget 命令无参数，是返回影片剪辑对象停止拖曳时位于其下方的影片剪辑对象，或是拖曳过程中位于其下方最后一个接触到的影片剪辑对象，例如：

```
myobject = my_mc.dropTarget;
```

取得当前影片剪辑对象“my_mc”在拖曳过程中位于其下方的影片剪辑对象，并指定给变量“myobject”。

3.4 鼠标键盘和声音的控制

3.4.1 鼠标的控制

利用“hide()”和“show()”来隐藏和显示 SWF 文件中的鼠标指针。默认情况下鼠标指针是可见的，但是可以将其隐藏。例如：

```
Mouse.show();
```

在 SWF 动画影片中将隐藏的系统鼠标指针显示出来。

```
my_mc.addEventListener(MouseEvent.MOUSE_OVER,chgMouse);
function chgMouse(me:MouseEvent){
Mouse.hide();
}
```

为对象“my_mc”建立鼠标事件侦听器，当鼠标指针进入对象范围内时（发生 MOUSE_OVER 事件），调用执行函数“chgMouse()”。在函数“chgMouse()”中使用 hide 方法隐藏系统默认的鼠标指针。

利用 startDrag 制作鼠标效果，配合鼠标对象的隐藏方法，便可以制造个性化的鼠标光标，替换默认的鼠标指针。

3.4.2 键盘的控制

在第 2 章学习了键盘事件的处理，任何对象都可以通过侦听器的设置来监控键盘操作，与键盘相关的操作事件都属于 KeyboardEvent 类。

对于键盘输入，经常要获取按键编码，可以使用以下两个属性。

（1）keyboardEvent.charCode；

（2）keyboardEvent.keyCode；

第一个属性记录了按下的键的字符编号，也就是 ASCII。ASCII（American Standard Code for Information Interchange，美国标准信息交换码）是目前计算机中使用得最广泛的字符集及其编码，它已被国际标准化组织（ISO）定为国家标准，每个字符都对应一个 ASCII，例如空格键对应的是 32，大写的“A”对应的是 65 等。

第二个属性记录了按下的键控代码值。这两个属性的区别在于，后者检查的是键盘上

按下的键，而前者检查的是实际输入的字符。例如，输入“X”和“x”，由于两个字符不同，所以“keyboardEvent. charCode”的结果不同，但是由于它们用的是同一个按键，所以它们的“keyboardEvent. keyCode”结果相同。

功能键也有按键值，但是不便于记忆。例如取消键(Esc 键)的按键值是 27，但是不必去记忆，这是因为 flash. ui. 包中“Keyboard”类有内置的常数来记录按键值，表 3-2 下面列出了 Keyboard 类中常用的一些常数。

表 3-2 Keyboard 类中常用的常数

常 数	说 明
Keyboard. ESCAPE	与 Esc 的键控代码值（27）关联的常数
Keyboard. DOWN	与向下箭头键的键控代码值（40）关联的常数
Keyboard. DELETE	与 Delete 的键控代码值（46）关联的常数
Keyboard. BACKSPACE	与 BackSpace 的键控代码值（8）关联的常数
Keyboard. CONTROL	与 Ctrl 的键控代码值（17）关联的常数
Keyboard. END	与 End 的键控代码值（35）关联的常数
Keyboard. ENTER	与 Enter 的键控代码值（13）关联的常数
Keyboard. HOME	与 Home 的键控代码值（36）关联的常数
Keyboard. LEFT	与向左箭头键的键控代码值（37）关联的常数
Keyboard. RIGHT	与向右箭头键的键控代码值（39）关联的常数
Keyboard. SHIFT	与 Shift 的键控代码值（16）关联的常数

例如 Keyboard. SHIFT 等于数值 16，因此你可检测 keyCode 是否等于 Keyboard. SHIFT 来发现 Shift 键是否被按下。检查键盘上的 Shift 键是不是被按下了，代码如下：

```
//侦听"keyDown"事件
 stage.addEventListener(KeyboardEvent.KEY_DOWN,showKey);
 function showKey(event:keyboardEvent){   //定义响应函数"showKey"
 if (event.keyCode == keyboard. SHIFT){   //查看按下的按键是不是 Shift 键
      trace("Shift 键被按下了!");         //输出"Shift 键被按下了!"
  }
}
```

按 Ctrl＋Enter 键测试影片，可以看到当按下 Shift 键的时候，在输出面板中打印“Shift 键被按下了!”，而按下其他键，则没有任何反应。

下面示例是一个获取方向键信息后控制圆形的移动。

```
package {
    import flash.display.Sprite;
    import flash.events.KeyboardEvent;
    import flash.ui.Keyboard;
    public class KeyCodes extends Sprite {
        private var ball:Sprite;
        public function KeyCodes() {
            init();
        }         private function init():void {
            ball   =   new   Sprite();
            addChild(ball);
```

```
            ball.graphics.beginFill(0xff0000);
            ball.graphics.drawCircle(0,  0,  40);
            ball.graphics.endFill();
            ball.x = stage.stageWidth/2;
            ball.y = stage.stageHeight/2;
            stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyEvent);
        }
        public function onKeyEvent(event:KeyboardEvent):void {
            switch (event.keyCode) {
                case Keyboard.UP :     //上方向键
                    ball.y -= 10;
                    break;
                case Keyboard.DOWN :   //下方向键
                    ball.y += 10;
                    break;
                case Keyboard.LEFT :   //左方向键
                    ball.x -= 10;
                    break;
                case Keyboard.RIGHT :  //右方向键
                    ball.x += 10;
                    break;
                default :
                    break;
            }
        }
    }
}
```

还有一点需要知道的是，当在 Flash IDE 环境中测试影片时，Flash IDE 会侦听一些如 Tab 键、所有的功能键和一些指定了快捷键的菜单，这些键不会在测试影片中被接收。可以按 Ctrl＋Enter 键预览影片的窗口，通过选择“控制”→“禁用快捷键”命令来解决。这样测试时就像真正工作在浏览器中一样。

3.4.3 声音的控制

1. 构造声音对象

语法：Sound 对象名称:Sound = new Sound(声音文件)；

Sound 类为 Media 组件中的类，Sound 类可以控制影片中的声音，在调用 Sound 类的方法以前，必须使用构造函数 new Sound 来建立 Sound 对象，其中声音文件为 URLRequest 类型参数，必须使用 URLRequest 类对象来转换字符串成为加载文件存储器的目标路径。

例如：

```
my_sound:Sound = new Sound( new URLRequest("test.mp3"));
```

构造“my_sound”声音对象并加载外部声音文件 test.mp3。

其他声音控制属性属于 Media 组件的特定类，主要为 SoundChannel 和 SoundTransform 这两个类，用于控制向计算机扬声器输出声音音量等属性，在使用前也必须先构造，例如：

```
my_soundChannel:SoundChannel = new SoundChannel ();
```

构造一个“my_soundChannel”SoundChannel 对象。

```
my_soundTransform:SoundTransform = new SoundTransform ();
```

构造一个“my_soundTransforml”SoundTransform 对象。

2. 加载声音文件

语法：Sound 对象名称.load(声音文件)；

load 方法可将指定的声音文件加载到 Sound 对象中，load 方法针对的是外部的声音文件，所以必须使用 URLRequest 类对象来转换文件来源字符串成为加载文件的目标路径。例如：

```
my_sound.load(new URLRequest("test.mp3"));
```

将声音文件“test.mp3”加载到“my_sound”对象中，声音文件“test.mp3”与 SWF 文件的所在位置相同(相同路径下)。

```
my_sound.load(new URLRequest ("http://www.xyz.com/test.mp3"));
```

将声音文件“test.mp3”加载到“my_sound”对象中，声音文件“test.mp3”与 SWF 文件的所在位置不同，位于 Internet 网络中。要取得加载的声音文件相关信息，例如声音文件的播放时间长度(length 属性)，必须在声音对象加载完成并没有错误发生后才能取消，也就是当声音对象的“complete”事件被触发之后。如果声音文件已经位于 SWF 文件的库中，同时在“连接属性”对话框中已设定成导出，此时只要以构造对象的方式就可以对该声音文件进行控制，并不需要再使用 load 方法。

3. 声音文件的播放与关闭

语法：

Sound 对象名称.play(开始播放时间位置，重复播放次数，SoundTransform 对象)；

Sound 对象名称.close( )；

当使用 load 方法让声音对象完成加载声音文件且没有错误发生后，可调用 play 方法开始播放声音文件，调用 play 方法但没有给定任何参数时，声音文件将从头开始播放一次，调用 play 方法也可给定参数来指定从何处开始播放、播放的次数。调用 play 方法后会返回一个 SoundChannel 对象，提供其他可以对声音文件进行的控制方法或属性设置，调用 close 方法会关闭音频数据流，所有已加载的数据将全部清除，此时声音对象将成为一个空对象。例如：

```
my_sound.close( );
```

关闭音频数据流，清除所有已加载的数据。

4. 取得声音已播放的时间

语法：SoundChannel 对象.position；

Sound 对象名称.length

position 属性值为声音已播放的经过时间，属性值的单位为毫秒。当声音被回放，则 position 属性值会在每次回放的开头被重设为 0。如果要得知声音的可播放长度(声音的持续时间)，可通过 Sound 声音对象的 length 属性知道。

```
NT = Math.floor(my_soundchannel.position/100)/10;
```

取得声音已播放的经过时间换算成含小数点一位数的秒数并存入变量“NT”，“my_soundChannel”为一个 SoundChannel 对象。

```
NT = Math.floor(my_channel.position/1000);
myM = Math.floor(NT / 60);
myS =  Math.floor(NT % 60);
trace("声音已播放: " + myM + "分" + myS + "秒";)
```

取得声音已播放的经过时间换算成分、秒并显示。

```
NT = Math.floor(my_sound.length/1000);
```

取得声音可播放总长度时间换算成秒数变存入变量“NL”，“my_sound”为一个 Sound 对象。

5. 设置/取得左右声音的声音变化(平衡)

语法：SoundTransform 对象.leftToLeft
SoundTransform 对象.leftToRight
SoundTransform 对象.rightToLeft
SoundTransform 对象.rightToRight

leftToLeft 属性是设置或取得左声道输出分配多少给左边的喇叭，leftToRight 属性是设置或取得左声道输出分配多少给右边的喇叭，rightToLeft 属性是设置或取得右声道输出分配多少给左边的喇叭，rightToRight 属性是设置或取得右声道输出分配多少给右边的喇叭，以下 4 个属性适用范围为控制“立体声”的左右声道，属性值都是介于 0～1 的数值。例如：

```
mySoundTransform. leftToLeft = 0;
```

设置左声道分配 0 输出给左边的喇叭。

```
myOut = mySoundTransform. leftToRight;
```

取得左声道分配给右边喇叭的输出值并存放变量“myOut”。

6. 设置/取得播放声音文件的音量

语法：SoundTransform 对象.volume

volume 属性用于设置/取得播放文件的音量，其属性值代表音量准位的数字(0～1)，1 是最大音量，0 则是静音。

```
myNum = mySoundTransform.volume;
```

取得目前播放声音文件的音量值并存入变量“myNum”。

```
mySoundTransform.volume = 1;
```

设置目前播放声音文件的音量值为1，最大音量。

7. 设置/取得左右声道的音量平衡

语法：SoundTransform 对象.pan

pan 属性控制了 SWF 文件当前和之后的声音的左右平衡，它决定声音在左右声道（喇叭）的播放方式或是哪一个喇叭（左或右）要播放声音，主要适用于单声道。其属性值域用于指定声音左右平衡，有效范围介于－1～1，其中－1 表示只使用左声道，1 表示只使用右声道，而 0 则表示平衡两个声道间的声音。

```
myNum = mySoundTransform.pan;
```

取得目前播放声音文件的音量平衡值并存入变量“myNum”。

```
mySoundTransform.pan = 1;
```

设置目前播放声音文件只有右声道有声音。

```
mySoundTransform.pan = myNum;
```

设置目前播放声音文件的左右声道平衡由变量“myNum”的值来决定。

3.5　Flash 的文本交互

Flash 的“交互”实际上是指人和程序之间的数据输入和输出过程：输入的内容，可以是触发事件，例如单击按钮，也可以是各种数据，例如输入字符串、数字等，这就需要用到“输入文本”；而输出的数据，则可以使用“动态文本”进行显示。下面学习动态文本与输入文本的相关知识。

3.5.1　文本类型

通过 Flash CS5 提供的文本工具，制作者可以方便地输入相应的文字。输入文字后，还可通过“属性”面板对其内容和样式进行编辑和设置。Flash 文本有以下 3 种类型。

图 3-4　设置为动态文本

1. 动态文本

所谓“动态文本”，指的是其中的文字内容可以被后台程序更新的对象。文本可以在动画播放过程中，根据用户的动作或根据当前的数据而改变。动态文本可以用于显示一些经常变化的信息，如比赛分数、股市行情和天气预报等。在属性面板中设置如图 3-4 所示。此种文本

类型使用 TextFiled 类来管理，动态文本的内容可以使用脚本语言利用 TextFled 类的 text 属性来实现设定。

2. 输入文本

所谓“输入文本”，指的是可以在其中由用户输入文字并提示的文本对象。“输入文本”对象的作用与 HTML 网页中的文本域表单作用一样。不过在 HTML 中，数据的输入和提交是在 Web 页面上完成的，而在 Flash 中，数据的输入和提交可以在动画中完成。输入文本也可以通过 TextField 类的实例来访问。

注意：“动态文本”的工作目的，主要用于显示变化文本，而“输入文本”的工作目的，主要是用于接收用户输入的文本，这两种文本对象的目的是不同的。

3. 静态文本

静态文本是一些不会改变的文字。此种文本类型只能通过 Flash 创作工具来创建，而不能使用 ActionScript 3.0 创建静态文本实例。此类型文本广泛用于 Flash 创作，用于在 Flash 中显示不变的文本。这些文字在发布为 SWF 格式文件后，显示出来的文字不再改变，不能够再更改。

本节着重学习动态文本与输入文本的使用。

3.5.2 文本实例名称

可以将动态文本与输入文本看成特殊的元件，在舞台上创建出来的每个动态文本，都是元件的实例。正如“人”是一类元件，那么“孔子”、“李世民”、“林则徐”都是元件的实例。如果要和某个人联系，就必须先确定名称，动态文本和输入文本一样，如果希望控制它的完整属性，那么就必须设置它的名称。

舞台上创建一个动态文本或输入文本，选中该文本，然后在“属性”面板中填写实例名称“my_txt”，其中后缀“_txt”可以触发文本类的相关代码提示。

单击选中主时间轴中的第一帧，选择菜单栏中的“窗口”→“动作”选项，打开“动作”面板，输入“my_txt.”，注意后面有“.”，即可触发提示代码列表，如图 3-5(a)所示。

在其中可以选择文本类的相关属性和动作，添加以下代码，效果如图 3-5(b)所示。

```
my_txt.text = "I'm a text!";
my_txt.textColor = 0x6600ff;
my_txt.alpha = 0.9;
my_txt.scaleX = 1;
my_txt.scaleY = 1.5;
```

【案例 3-1】 小学生算数游戏。计算机连续地随机给出两位数的加减法算术题，要求学生回答，答对的打“√”，答错的打“×”。将做过的题目存放在多行文本框中备查，并随时给出答题的正确率，运行界面如图 3-6 所示。

(1) 打开 Flash CS5 软件后，选择菜单“文件”→“新建”选项，系统将弹出“新建文档”窗口。在窗口中选择 ActionScript 3.0 选项。

(2) 选择菜单“修改”→“文档”选项，打开“文档属性”对话框。设置场景的尺寸为 300×

(a)　　(b)

图 3-5　设置为动态文本

450 像素，然后单击“确定”按钮。

从主菜单上选择“窗口”→“属性”选项，打开“属性”面板（见图 3-7），在属性面板设置文档类为 SuanShu。

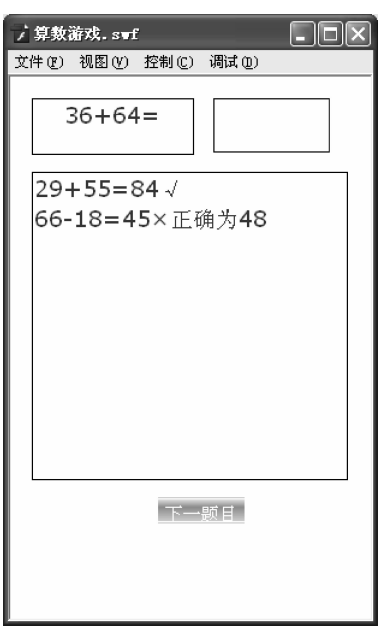

图 3-6　算数游戏界面

图 3-7　设置 Flash 文件文档类及相关属性

(3) 在工具面板中选择“文本工具”，在场景舞台上添加两个文本，并在属性面板中将文本设置为“动态文本”，并将实例分别命名为 ti_mu 和 all_mu，其中 ti_mu 是显示题目，all_mu 是多行文本用来显示做过题目的对错。

在场景舞台上再添加一个文本，并在“属性”面板中将文本设置为“输入文本”，并将实例命名为 input_txt，用来输入玩家答案。

(4) 从主菜单上选择“窗口”→“公用库”选项,再选择“按钮”选项,打开“按钮”库面板(见图 3-8),从中选取如 bar blue 按钮拖动到舞台上。双击舞台上 bar blue 按钮元件实例,进入元件编辑状态,选中此元件的 text 层,修改文字为“下一题目”。

(5) 单击“场景 1”从元件编辑状态回到舞台,选中“下一题目”元件,在“属性”面板中将实例名改为“next_btn”。

(6) 选择菜单“文件”→“新建”选项,将弹出“新建文档”窗口。在窗口中选择“ActionScript 3.0 类”选项。这样在 Flash 中新建一个 ActionScript 类文件,将其保存为 SuanShu.as,并保存到刚刚创建 Flash 文件所在的文件夹中。SuanShu 类的具体代码如下。

图 3-8 “按钮”库面板

```
package {
    import flash.text.TextField;
    import flash.display.*;
    import flash.events.*;
    public class SuanShu extends MovieClip {
        private var arr:Array = new Array();
        private var arr2:Array = new Array();
        private var m1:Number;
        private var m2:Number;
        private var m:uint = 90;
        private var op:uint;
        private var ans:Number;
        public function SuanShu():void {
            chu_ti();                    //出题
            next_btn.addEventListener(MouseEvent.CLICK,clickNext);
        }
```

由于需要产生多道题目,所以使用一个函数 chu_ti()完成出题功能。每道题用 Math.random()*m+10 产生范围在 10～99 的两个随机整数作为操作数,同时加减运算也是随机的,用随机数方法 Math.random()*2 返回一个[0,1]之间的整数,1 代表加法,0 代表减法。若为减法,应将大数作为被减数。

```
public function chu_ti() {
    var n1:uint;
    var n2:uint;
    n1 = Math.random() * m + 10;
    n2 = Math.random() * m + 10;
    op = Math.random() * 2;
    m1 = n1;
    m2 = n2;
    if(op == 1){          //1 代表加法
        ti_mu.text = String(m1) + " + " + String(m2) + " = ";
        ans = m1 + m2;
    }
    else{                 //0 代表减法
        if(m1 >= m2){     //若为减法,应将大数作为被减数
            ti_mu.text = String(m1) + " - " + String(m2) + " = ";
```

```
            ans = m1 - m2;
        }
        else{
            ti_mu.text = String(m2) + " - " + String(m1) + " = ";
            ans = m2 - m1;
        }
    }
}
```

判断正误是在小学生输入答案并单击“下一题目”按钮后进行的，因此相关的代码应放在“下一题目”按钮的单击时间事件中完成的按键事件 clickNext()中完成。

```
public function clickNext(event:MouseEvent) {
    //处理本题是否答对
    if(int(input_txt.text) == ans)
        //all_mu.appendText(ti_mu.text + input_txt.text + "√" + "\n");
        all_mu.text = ti_mu.text + input_txt.text + "√" + "\n" + all_mu.text;
    else
        all_mu.text = ti_mu.text + input_txt.text + "×" +
                "正确为" + String(ans) + "\n" + all_mu.text;
    chu_ti();              //出下一题
    input_txt.text = "";
}
```

(7) 按 Ctrl+Enter 键测试并预览动画显示效果。

【案例 3-2】 倒计时程序。游戏默认玩 10 秒钟，每秒钟刷新一次剩余时间。运行界面如图 3-9 所示。

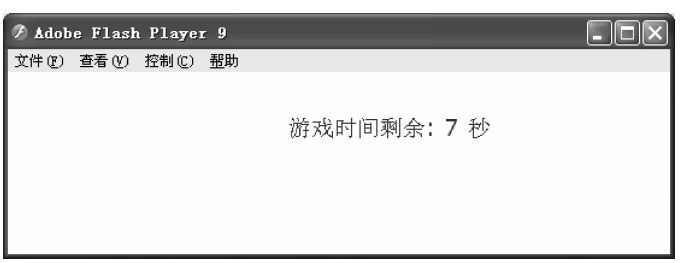

图 3-9　倒计时程序

(1) 打开 Flash CS5 软件后，选择菜单“文件”→“新建”选项，系统将弹出“新建文档”窗口。在窗口中选择 ActionScript 3.0 选项。

(2) 在工具面板中选择“文本工具”，在场景舞台上添加一个文本，并在“属性”面板中将文本设置为“动态文本”，并将实例命名为 hint_txt(显示剩余时间)。

(3) 单击选中主时间轴中的第一帧，选择菜单栏中的“窗口”→“动作”选项，打开“动作”面板，输入以下代码：

```
var tempcount:int = 0;                          //临时计数变量
var totaltime:int = 10;                         //游戏默认玩 10 秒钟
var gameTimer:Timer = new Timer(1000);          //1 秒钟刷新一次
gameTimer.addEventListener(TimerEvent.TIMER,gameTimerHandler);
```

```
gameTimer.start();                              //启动定时器
function gameTimerHandler(event:TimerEvent){
    tempcount++;
    if(tempcount > totaltime - 1){
        hint_txt.text = "游戏时间已经结束!";
        tempcount = 0;
        gameTimer.stop();
    }else{
        hint_txt.text = "游戏时间剩余: " + (totaltime - tempcount) + " 秒";
    }
}
```

下面制作具有动画效果的倒计时。这里先制作一个倒计时的影片剪辑，这个影片剪辑的长度为 50 帧，用形状补间动画把里面的填充色块逐帧扩大。

(1) 选择菜单中的“插入”→“新建元件”选项，在“创建新元件”对话框中，输入元件名称(“倒计时条”)及元件类型(勾选“影片剪辑”)，然后单击“确定”按钮。在图层 1 第 1 帧上用矩形工具，画一个矩形，宽 300 像素，高 20 像素，边框设为绿色，填充色为红色。左边边缘对齐舞台的中心点，如图 3-10 所示。

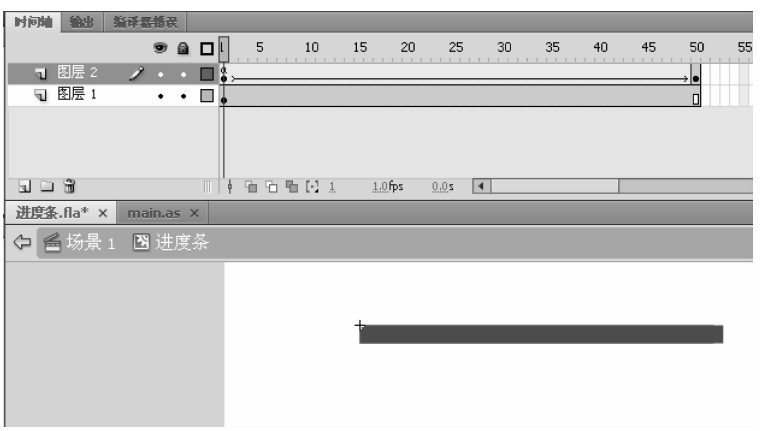

图 3-10　倒计时程序

(2) 用箭头工具选中中间的填充色块红色部分，按 Ctrl＋X 键，剪切矩形填充色块。选中图层 2 第 1 帧上，选择菜单“编辑”→“粘贴到当前位置”选项。

(3) 在图层 2 第 50 帧插入关键帧，把色块宽度调整到 1 像素(提示：可以在“属性”面板修改数据，来缩小色块的宽度)。在图层 1 上第 50 帧按 F5 键插入普通帧。按 F9 键插入如下脚本：

```
stop();
```

(4) 选择“文件”→“新建”选项，系统将弹出“新建文档”窗口。在窗口中选择“ActionScript 3.0 类”选项。这样在 Flash 中新建一个 ActionScript 文件，将其命名为 main.as，输入下面代码，并保存到创建 Flash 文件.fla 所在的文件夹中。

```
package {
    import flash.text.TextField;
```

```
import flash.display.*;
import flash.events.*;
import flash.utils.Timer;
public class main extends MovieClip {
    var tempcount:int = 0;                          //临时计数变量
    var totaltime:int = 50;                         //游戏默认玩 50 秒钟
    var gameTimer:Timer = new Timer(1000,50);       //1 秒钟一次,重复 50 次
    //构造函数
    public function main() {
        hint_txt.text = "剩余: " + (totaltime - tempcount) + " 秒";
        gameTimer.addEventListener(TimerEvent.TIMER,gameTimerHandler);
        gameTimer.addEventListener(TimerEvent.TIMER_COMPLETE
                                   ,COMPLETEHandler);
        gameTimer.start();                          //启动定时器
    }
    function gameTimerHandler(event:TimerEvent) {
        tempcount++;
        clock_mc.gotoAndStop(tempcount);
        hint_txt.text = "剩余: " + (totaltime - tempcount) + " 秒";
    }
    function COMPLETEHandler(event:TimerEvent) {
        hint_txt.text = "游戏时间已经结束!";
        gameTimer.removeEventListener(TimerEvent.TIMER_COMPLETE
                                   ,COMPLETEHandler);
        gameTimer.removeEventListener(TimerEvent.TIMER,gameTimerHandler);
    }
}
}
```

(5) 选择菜单“文件”→“新建”选项,系统将弹出“新建文档”窗口。在窗口中选择 ActionScript 3.0 选项。在创建的 Flash 文件中加入“倒计时条”影片剪辑,实例名为 clock_mc。在工具面板中选择“文本工具”,在场景舞台上添加一个文本,并在“属性”面板中将文本设置为“动态文本”,并将实例命名为 hint_txt(显示剩余时间)。

(6) 按 Ctrl+Enter 键测试并预览动画显示效果,如图 3-11 所示。

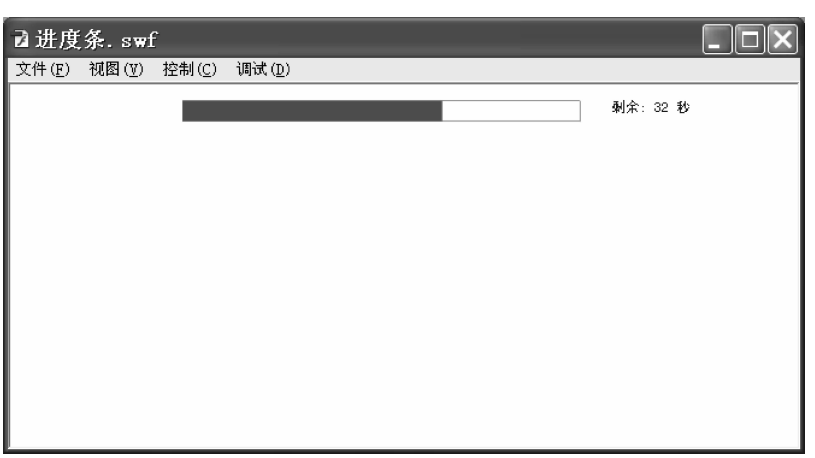

图 3-11　倒计时条效果图