# 第 3 章　动画编程

## 3.1　ActionScript 3.0 编程

### 3.1.1　按钮编程——声音的产生与传播

**任务布置**

制作声音的产生与传播课件，利用按钮实现单击响应播放声音，如图 3-1-1 所示。要求该小型课件具有交互性，能多次播放，讲述声音的产生原理；通过单击按钮听到相应乐器发出的声音。

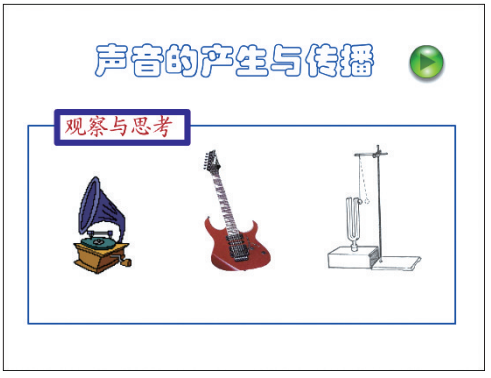


图 3-1-1　声音的产生与传播效果

**知识讲解**

**1. 按钮元件的创建**

按钮元件有 4 个帧，对应不同状态。

第 1 帧（弹起）：按钮的常规状态。

第 2 帧（指针经过）：鼠标指针经过按钮时显示的状态。

第 3 帧（按下）：鼠标按下去时显示的状态。

第 4 帧（单击）：按钮的热区，也就是感应范围。

**2. 声音同步属性的设置**

Flash 中声音的使用类型有两种：流式声音和事件激发声音，如图 3-1-2 所示。

流式播放可以使声音独立于时间线，自由、连续播放，如给作品添加背景音乐，也可


图 3-1-2　声音同步属性的设置

以和动画同步。

事件激发播放允许将声音文件附着在按钮上，可以使按钮更易体现交互性。

“事件”选项会将声音和一个事件的发生过程全部同步起来。如果触发了播放声音的事件，它会自动播放直至结束，在这个过程中声音的停止不受动画本身的制约。例如，在 Flash 中制作了一个声音播放按钮，如果事件声音正在播放，再次单击，则第一个实例继续播放，另一个声音实例同时开始播放。

“开始”选项和“事件”选项一样，只是如果声音正在播放，就不会播放新的声音实例。“停止”选项可以使指定的声音静音。向影片第 1 帧导入声音，在第 50 帧处创建关键帧，选择要停止的声音，在“同步”下拉列表中选择“停止”选项，则声音播放到第 50 帧时停止播放。

“数据流”用于在因特网上同步播放声音。Flash 会协调动画与声音流，使动画与声音同步。如果 Flash 显示动画帧的速度不够快，Flash 会自动跳过一些帧。与事件声音不同的是，如果声音过长，而动画过短，声音流将随着动画的结束而停止播放。播放影片时，声音流是混合在一起播放的。

**创意设计**

步骤 1：新建一个 Flash 文件，设置舞台的尺寸为 550px×400px，然后将文件以“声音的产生”命名并保存到指定的目录。

步骤 2：输入文字“声音的产生与传播”，文字颜色为“#0030CE”，大小为 40，字体任选。

步骤 3：选择“矩形工具”绘制一个大的蓝色矩形边框，笔触值为 2，并组合这个矩形框。再绘制一个小的紫色矩形边框，笔触值为 5，填充色为白色，并组合这个矩形框。在小矩形框上输入文字“观察与思考”，如图 3-1-3 所示。

步骤 4：选择“插入”→“新建元件”命令，弹出如图 3-1-4 所示的“创建新元件”对话框，创建一个按钮元件“留声机”。

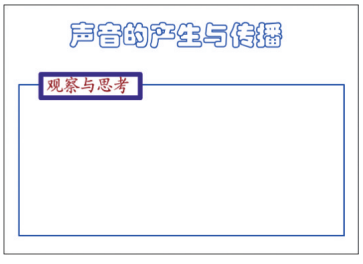


图 3-1-3　绘制边框

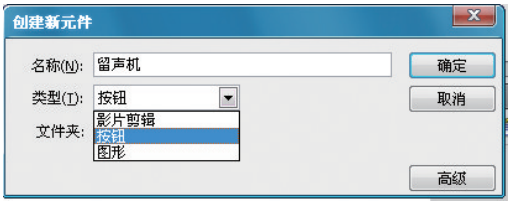


图 3-1-4　插入按钮元件

步骤 5：将素材全部导入库，进入按钮“留声机”的编辑区。

步骤 6：选择时间轴上的“弹起”，拖曳库中留声机的图片至舞台。在时间轴上的“指

针经过”处插入关键帧，在图片下方输入文字“单击留声机播放声音”。在时间轴上的“按下”处插入关键帧，删除文字，拖曳库中留声机的声音至舞台，如图 3-1-5 所示。

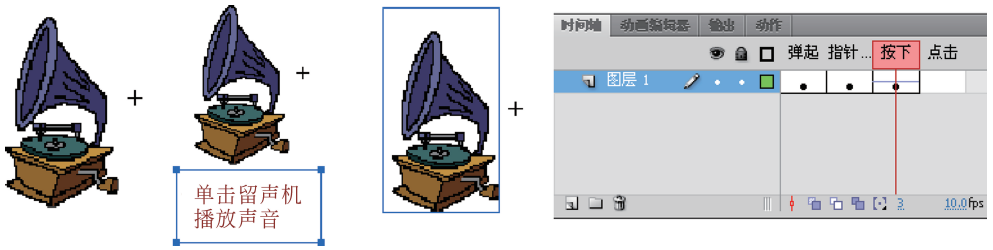


图 3-1-5　在“弹起”“指针经过”“按下”时的状态

步骤 7：返回到主场景，将库中的按钮“留声机”拖入舞台的大矩形框中，并调整大小和位置。

步骤 8：参照按钮“留声机”的制作方法，继续制作按钮“吉他”和“音叉”。将制作好的按钮拖入舞台的大矩形框中，并调整大小和位置。

步骤 9：选择“窗口”→“公用库－按钮”命令，在弹出的窗口中选择文件夹 Playback 内的按钮，如图 3-1-6 所示。

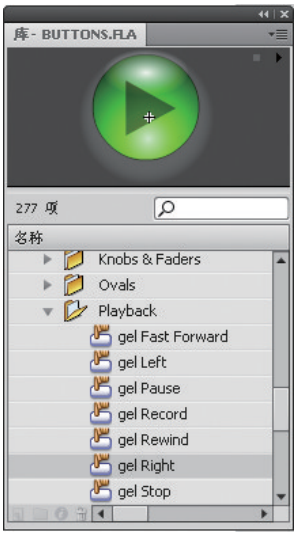


图 3-1-6　插入按钮 1

步骤 10：右击刚才插入的按钮，在弹出的快捷菜单中选择“动作”命令。在“动作”面板中输入以下代码：

```
on(press){
    gotoAndPlay(2);
}
```

步骤 11：右击“图层 1“的第 1 帧，在弹出的快捷菜单中选择“动作”命令。在“动作”面板中输入代码：

```
stop();
```

步骤 12：选择“图层 1”，并在第 2 帧处插入空白关键帧。接着输入文字“声音是怎样产生的?”，文字颜色为“＃0099CC“，大小为 50，字体任选。再用“直线工具”绘制一条墨绿的直线，笔触值为 3，如图 3-1-7 所示。

步骤 13：选择“插入”→“新建元件”命令，在弹出的对话框中创建一个透明按钮元件。

步骤 14：进入该按钮的编辑区，在图层的“弹起”“指针经过”处插入空白关键帧，在“按下”处绘制一个矩形，填充色任意。返回到场景，将刚刚创建的按钮拖曳到直线上方，如图 3-1-8 所示。

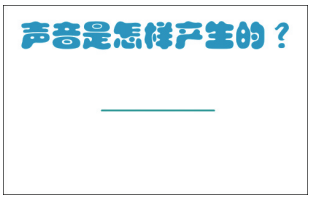


图 3-1-7　插入文字和线条

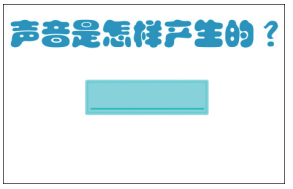


图 3-1-8　插入按钮

步骤 15：右击刚才插入的按钮，在弹出的快捷菜单中选择“动作”命令。在“动作”面板中输入以下代码：

```
on(press){
    gotoAndPlay(3);
}
```

步骤 16：右击图层 1 的第 2 帧，在弹出的快捷菜单中选择“动作”命令。在“动作”面板中输入以下代码：

```
stop();
```

步骤 17：在第 3 帧处插入关键帧，删除按钮，并在直线的上方插入文字“空气的震动”，设置文字颜色为“＃DB2D02”，大小为 35，字体任选。

步骤 18：选择“窗口”→“公用库”→“按钮”命令，在弹出的窗口中选择文件夹 Playback 内的按钮，如图 3-1-9 所示。

步骤 19：右击刚才插入的按钮，在弹出的快捷菜单中选择“动作”命令。在“动作”面板中输入以下代码：

```
on(press){
    gotoAndPlay(1);
}
```

步骤 20：调整按钮和文字的位置，如图 3-1-10 所示。

步骤 21：右击图层 1 的第 3 帧，在弹出的菜单中选择“动作”命令。在“动作”面板中输入以下代码：

```
stop();
```

步骤 22：保存文件，按 Ctrl＋Enter 组合键测试影片。

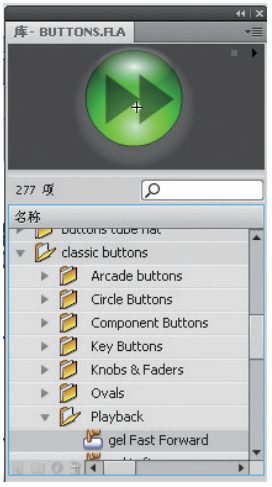


图 3-1-9　插入按钮 2


图 3-1-10　调整按钮和文字的位置

**创意拓展**

### 1. 提取声音的某个部分

在时间轴上插入声音后，在声音的“属性”对话框中单击“效果”后的“编辑”按钮，打开“编辑封套”对话框，选择自定义效果。需要声音的某个部分时，音量控制点的位置如图 3-1-11 所示，“几”字形中间的区域就是需要播放的部分。开始播放音乐时，音量控制点

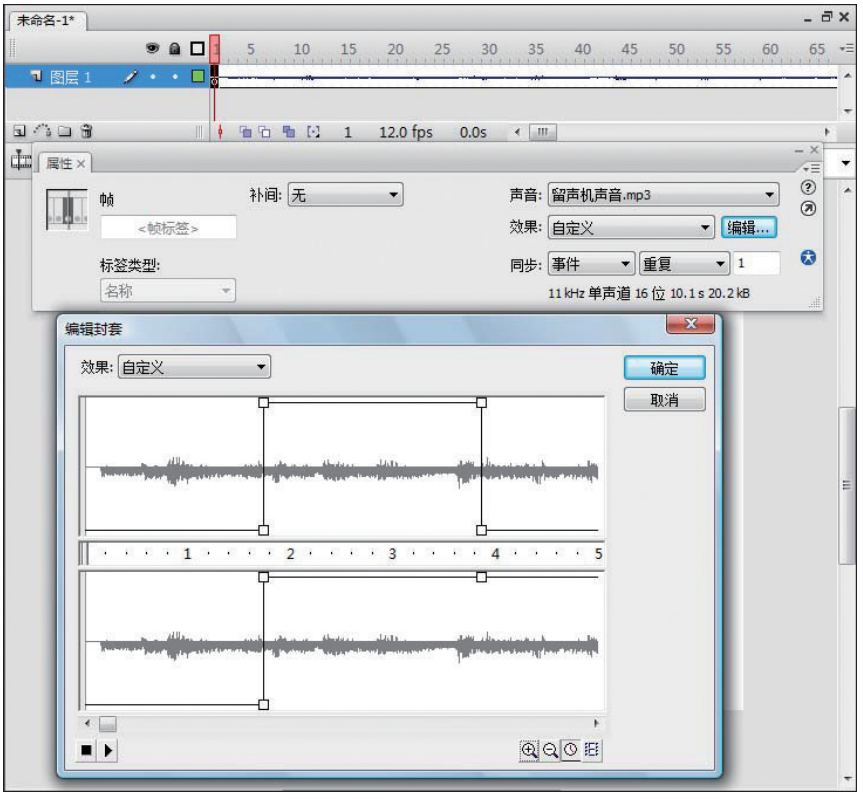


图 3-1-11　提取声音的某个部分

一直处于最低的位置，表示没有声音；到指定区域后，音量控制点垂直进入最高的位置，表示此时的声音为最高的音量。音量控制线在两个音量控制点之间是垂直的。

### 2. 声音的淡入淡出效果

打开“编辑封套”对话框，选择“自定义”效果，设置音量控制线，如图 3-1-12 所示。音量控制线在两个音量控制点之间呈“/”“\”倾斜，淡入淡出的快慢取决于音量控制线的角度，角度越大，淡入淡出的时间越长；角度越小，淡入淡出的时间越短。当角度为 90°时，则没有淡入淡出效果。

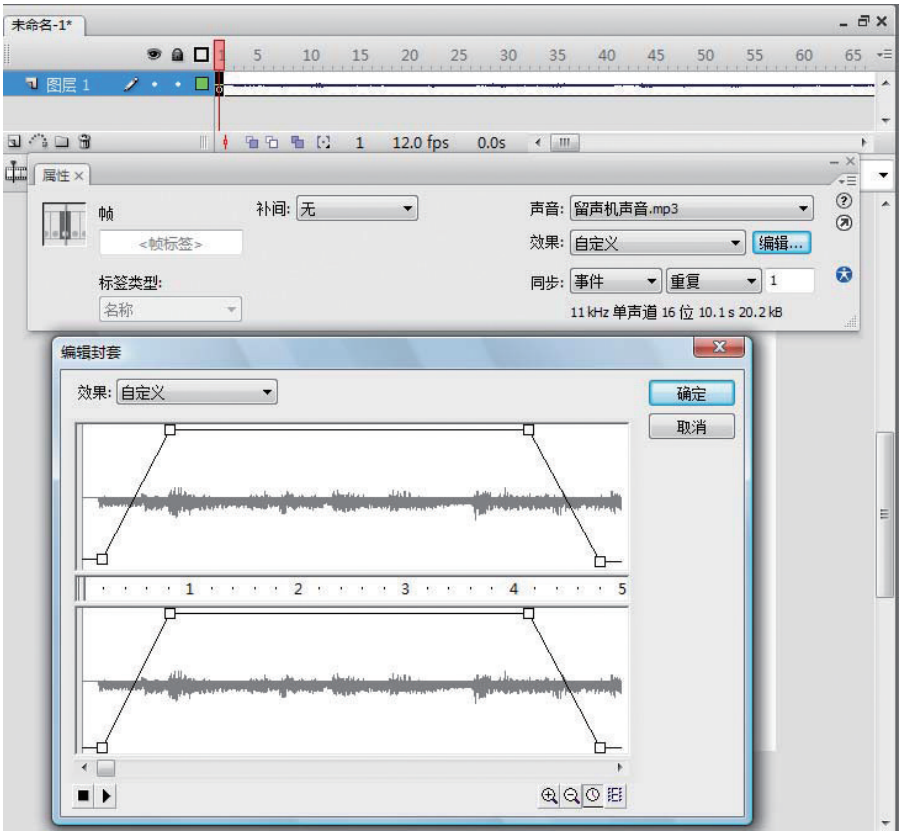


图 3-1-12　声音的淡入淡出

### 3. 实现代码

Animate 不支持 ActionScript 2.0 脚本编程，如需实现案例的以上功能，需要使用 ActionScript 3.0 脚本编程。具体方法是：单击“下一页”按钮的实例并命名为 bt1，为该按钮添加侦听器，侦听鼠标单击事件的发声，然后执行对应的自定义函数，代码如下：

```
bt1.addEventListener(MouseEvent.CLICK, fl_fun);
function fl fun(event:MouseEvent):void
{
    gotoAndPlay(2);
}
```

**创意分享**

使用 ActionScript 3.0 交互技术，创意设计老年手机按键拨号动画。建议有按键动

作和按键声音。将完成后的作品发布到学习平台供同学们交流与学习。

### 3.1.2 按钮编程——场景切换

**任务布置**

制作一个动画演示系统，将前面学习过的典型动画案例集结在一个文件，放在不同场景中。通过菜单选项进行场景切换，选择不同的菜单项，执行不同的动画场景，每个动画场景结束后返回主菜单，重新选择菜单项。多场景动画菜单如图 3-1-13 所示。


图 3-1-13 多场景动画菜单

**知识讲解**

(1) 场景面板的使用。Flash 中，一个文件里可以包括多个场景，一个场景可以包含一个舞台，一个舞台可以包含多个关键帧，所有场景共用一个库。新建 Flash 文件默认只有“场景 1”，根据作品需要可以创建多个场景，便于规划作品内容，也便于分场景调试作品。选择“窗口”→“其他面板”→“场景”命令(或者按 Shift＋F2 组合键)打开“场景”面板，如图 3-1-14 所示。面板的左下角有 3 个按钮，分别是添加场景、复制场景、删除场景，可添加新的场景、复制场景以及删除选中的场景。在场景名称上双击，可重命名该场景，拖动场景可上下调整场景顺序。一般情况下，本场景会按照“场景”面板中的排序执行，如果遇到 ActionScript 脚本控制语句，则可能改变场景的执行顺序。

(2) 制作透明按钮。有一类按钮，其形状相同，大小、位置、内容不同，可以制作为一个透明的通用按钮，它具备按钮的一般属性，而且简单、实用。下面制作适用于菜单选择的透明按钮。

首先，创建影片剪辑元件，绘制一矩形，宽度、高度与菜单选项的宽度、高度相同。

其次，创建按钮元件，将上面制作的影片剪辑元件拖入按钮元件第 1 帧，选中矩形元件，修改其属性，选择“色彩效果”→“样式”→Alpha，设置 Alpha 属性值为 0%，使其完全透明。

最后，返回场景，制作菜单，将按钮元件拖动到舞台中的每个菜单项，测试影片，在按钮覆盖区域鼠标指针变为形状，如图 3-1-16 所示。

**创意设计**

步骤 1：新建 ActionScript 2.0 类型文件，参照图 3-1-15 所示内容输入菜单选项。打开“场景”面板，将“场景 1”重命名为“菜单”，另外新建 3 个场景，命名为“转动动画”“变形动画”“骨骼动画”，分别在这 3 个场景中创建不同的动画演示片段。

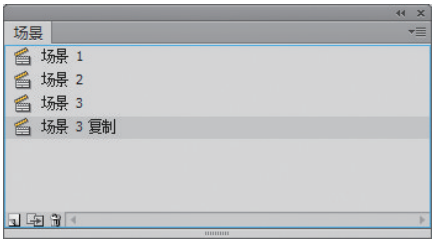


图 3-1-14　“场景”面板

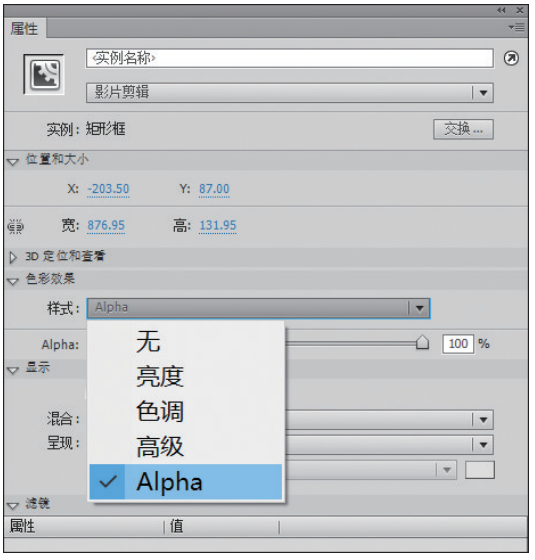


图 3-1-15　设置元件的 Alpha 属性

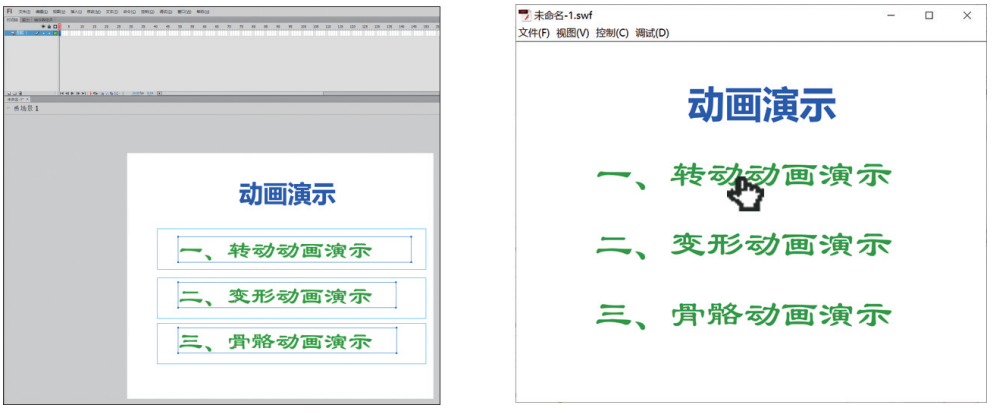


图 3-1-16　透明按钮的应用

步骤 2：返回“菜单”场景，新建矩形透明按钮，将按钮拖动到舞台中的 3 个菜单选项上。在第一帧输入控制语句：

```
stop();
```

使文件执行停止在第 1 帧，等待用户选择菜单项。在第一个按钮上右击，打开动作面板，输入：

```
on(release){
    gotoAndPlay("转动动画",1);
}
```

步骤 3：进入“转动动画”场景，在动画的最后一帧输入控制语句：

```
gotoAndPlay("菜单",1);,
```

使动画结束后返回“菜单”场景。

步骤 4：返回“菜单”场景，测试影片，影片可以在“菜单”场景和“转动动画”场景之间

进行切换。

步骤 5：按照以上步骤实现“变形动画”场景、“骨骼动画”场景与“菜单”场景的切换。

**创意拓展**

**1. 透明按钮**

透明按钮是一种特殊的按钮。将按钮设置为透明后，它不会再显示任何视觉元素，但仍然可以被用户单击。透明按钮在游戏中常用于创建互动元素，如菜单按钮、交互式地图中的物体或角色等，通过使用透明按钮，可以轻松实现用户与场景间的交互。透明按钮在网页中常用于各种菜单、网页广告、动画中，在浏览器中可以随时单击相关区域，进入相关网页。

透明按钮通常在需要用户交互但又希望保持设计简洁或者不妨碍底层内容的场景下使用。这种按钮让用户知道它们可以被点击，但其由于透明性，不会过于突出或干扰视觉体验，如界面菜单、弹窗或浮动窗口、图片或产品展示等。

**2. 多场景动画**

多场景动画技术的使用可以为提升作品制作质量和效率助力。

在故事叙述中，多场景动画可用于讲述连续的故事情节。通过在不同的场景中展示相应的动画效果，可以吸引观众的注意力并传达你想表达的故事。

在游戏开发与游戏设计中，多场景动画非常重要。每个游戏关卡或场景都可以被设计成独立的场景动画，使玩家在游戏过程中体验不同的环境和挑战。

在广告制作中，多场景动画也很常见。通过跳转不同的场景，可以展示产品或服务在不同情境下的使用场景，吸引潜在客户的兴趣，传递产品或品牌信息。

在教育和培训中，多场景动画可用于教育和培训领域，比如创建交互式教学动画和模拟场景。它可以更好地呈现复杂概念和流程，帮助学生或员工理解和记忆知识。

在视觉艺术创作中，多场景动画也可以被视为一种艺术形式，可用于短片、艺术展览或视觉转场效果等创作。

3. Animate 软件中不支持 ActionScript 2.0 脚本功能，要实现场景跳转，需使用 ActionScript 3.0 事件侦听功能。参考代码如下：

```
bt1.addEventListener(MouseEvent.CLICK,fun_1);
function fun_1(event:MouseEvent): void
gotoAndPlay(1,"平动动画");
```

**创意分享**

在 Animate 中使用 ActionScript 3.0 实现菜单设计，实现 3 个以上案例交互展示。将完成后的作品发布到学习平台供同学们交流与学习。

## 3.2 ActionScript 3.0 基础编程

### 3.2.1 鼠标事件——媒体播放器

**任务布置**

设计一个媒体播放器，控制影片的播放、暂停和停止，效果如图 3-2-1 所示。

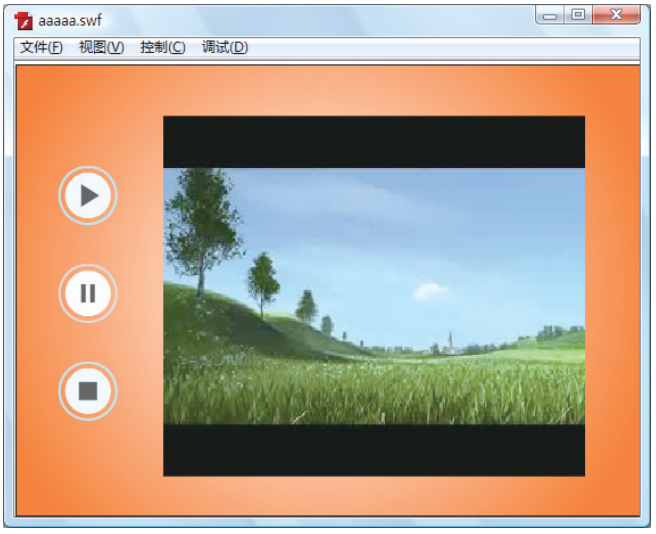


图 3-2-1　媒体播放器

**知识讲解**

### 1. ActionScript 概述

ActionScript 是针对 Flash Player 运行环境的脚本编程语言，它使 Flash 应用程序实现了交互性、数据处理以及其他许多功能。

ActionScript 包括两部分：核心语言和 Flash Player API。核心语言用于定义编程语言的结构，如声明、表示、条件、循环和类型。Flash Player API 是提供给 ActionScript 语言使用的一组类和函数，用于使用 Flash Player 的功能。通过这些类和函数，开发人员可以使用 ActionScript 3.0 代码访问 Flash Player 运行时的许多功能。Flash Player API 是对核心语言的一个非常重要的补充。ActionScript 代码通常被 Flash 提供的编译器编译成“字节码”格式，字节码嵌入在.swf 文件中，由运行时环境 Flash Player 执行（由 Flash Player 内置的 ActionScript 虚拟机执行）。Flash Player 由两部分组成：ActionScript 虚拟机（AVM）和渲染引擎。要显示的内容先由 AVM 创建显示对象，再由渲染引擎将其显示在屏幕上。Flash Player 的运行平台结构如图 3-2-2 所示。

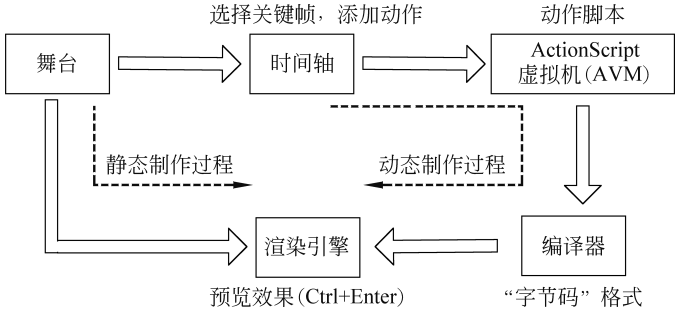


图 3-2-2　Flash Player 的运行平台结构

### 2. 数据类型

Flash 中包括两种数据类型，即原始数据类型和引用数据类型。原始数据类型包括

字符串、数字和布尔值，都有一个常数值，因此可以包含它们所代表的元素的实际值；引用数据类型是指影片剪辑和对象，值可能更改，因此它们包含对该元素实际值的引用。

### 3. 处理对象

ActionScript 3.0 是一种面向对象编程（OOP）的语言，面向对象的编程仅是一种编程方法，它与使用对象组织程序中的代码的方法并没有差别。

1）属性

属性是对象的基本特性，如影片剪辑元件的位置、大小和透明度等。它表示某个对象中绑定在一起的若干数据块的一个。

2）方法

方法是指可以由对象执行的操作。如果在 Flash 中使用时间轴上的几个关键帧和基本动画制作了一个影片剪辑元件，则可以播放或停止该影片剪辑，或者指示它将播放头移动到特定的帧。

3）事件

事件是确定执行哪些指令以及何时执行的机制。事实上，事件是指所发生的、ActionScript 能识别并可响应的事情。许多事件与用户交互动作有关，如用户单击按钮或按下键盘上的键等操作。

4）创建对象实例

在 ActionScript 中使用对象之前，必须确保该对象存在。创建对象的一个步骤就是声明变量。但仅声明变量，只表示在计算机内创建了一个空位置，所以需要为变量赋一个实际的值，这样整个过程就称为对象的“实例化”。除了在 ActionScript 中声明变量时赋值外，用户也可以在“属性”面板中为对象指定对象实例名。

### 4. ActionScript 3.0 的使用

在 ActionScript 3.0 环境下，按钮或影片剪辑不可以被直接添加代码，用户只能将代码输入在时间轴上，或者将代码输入在外部类文件（ActionScript 文件）中。

- 在时间轴上输入代码：在 Flash CS6 中，用户可以在时间轴上的任何一帧中添加代码，包括主时间轴和影片剪辑的时间轴中的任何帧。输入时间轴的代码将在播放头进入该帧时被执行。
- 在外部 ActionScript 文件中添加代码：当用户需要创建较大的应用程序或者包括重要的代码时，就可以创建单独的外部 ActionScript 类文件并在其中组织代码，如图 3-2-3 所示。

### 5. ActionScript 3.0 中的事件和事件处理

ActionScript 3.0 中的事件指由鼠标、键盘、文本输入、加载数据、远程连接及与.swf 文件进行的交互操作。事件用事件对象表示，事件对象是 Event（事件）类或 Event 类子类的实例。

事件侦听器是用户编写的用于响应事件的函数或方法，并添加到事件目标/显示对象列表。事件侦听器的创建格式：

```
事件目标.addEventListener(事件对象的事件名称,事件侦听函数);
```

事件侦听器的删除格式：

```
事件目标. removeEventListener(事件对象的事件名称,事件侦听函数);
```

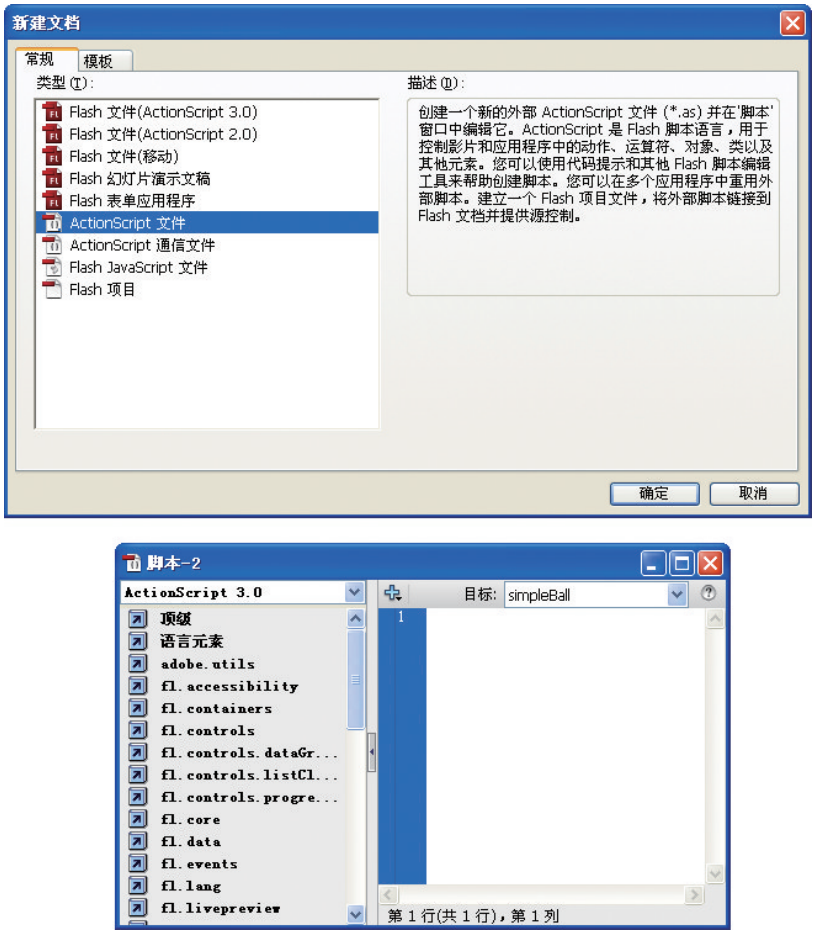


图 3-2-3　在外部 ActionScript 文件中添加代码

事件侦听器的格式说明如下。

(1) 事件侦听函数：响应事件要执行的动作或方法。

(2) 事件对象：事件对象指定的类名称。

(3) 事件目标：被侦听的对象名称。

(4) 事件对象的事件名称：事件常量。

事件侦听函数的格式如下。

```
function 事件侦听函数(事件对象:事件类型):void {
    //为响应事件而执行的动作
}
```

### 6. 鼠标事件处理方法

鼠标事件是指与鼠标操作有关的事件，如单击事件、双击事件、鼠标按下事件、鼠标滑出事件等。

MouseEvent 类定义了以下 10 种鼠标事件。

CLICK：鼠标单击事件。

DOUBLE_CLICK：鼠标双击事件。

MOUSE_DOWN：鼠标按下事件。
MOUSE_MOVE：鼠标移动事件。
MOUSE_OUT：鼠标移出事件。
MOUSE_OVER：鼠标移过事件。
MOUSE_UP：鼠标释放事件。
MOUSE_WHEEL：鼠标滚轮滚动事件。
ROLL_OUT：滑入事件。
ROLL_OVER：滑出事件。

（1）鼠标单击事件：

```
btn.addEventListener(MouseEvent.CLICK,functionbtn);
//鼠标单击事件,btn 为按钮元件实例名称
function functionbtn(event:MouseEvent):void//事件处理方法
{
this.play();//执行代码
}
```

（2）鼠标双击事件：

```
btn.addEventListener(MouseEvent. DOUBLE_CLICK ,functionbtn);
//鼠标双击事件,btn 为按钮元件实例名称
function functionbtn(event:MouseEvent):void//事件处理方法
{
This.play();//执行代码
}
```

（3）鼠标移动事件：

```
Ball_mc.addEventListener(MouseEvent.MOUSE_DOWN ,mouseDownListener);
Ball_mc.addEventListener(MouseEvent.MOUSE_UP ,mouseUpListener);
//鼠标拖动事件需要加入两个事件侦听,Ball_mc 为影片剪辑元件实例名称
functionmouseDownListener (event:MouseEvent):void
{
Ball_mc.startDrag();//可以拖动
}
functionmouseUpListener (event:MouseEvent):void
{
Ball_mc.stopDrag();//停止拖动
}
```

**创意设计**

步骤 1：新建文件，导入外部视频 movie.flv，选择在.swf 文件中嵌入视频并在时间轴上播放的视频部署方式，符号类型选影片剪辑。

步骤 2：新建背景图层，绘制矩形框，设置背景颜色为白色—橙色放射性填充。

步骤 3：新建影片图层，拖动视频影片剪辑放置在舞台适当位置，影片剪辑实例命名为 movie。

步骤 4：新建按钮图层，在公共按钮库中选择播放、暂停和停止 3 个按钮，分别命名为 playBt、pauseBt 和 stopBt。

步骤 5：添加代码。

```
playBt.addEventListener(MouseEvent.CLICK,playFun);
stopBt.addEventListener(MouseEvent.CLICK,stopFun);
pauseBt.addEventListener(MouseEvent.CLICK,pauseFun);
function playFun(event:MouseEvent):void{
    movie.play();
    }
function pauseFun(event:MouseEvent):void{
    movie.stop();
    }
function stopFun(event:MouseEvent):void{
    movie.gotoAndStop(1);
    }
```

**创意拓展**

**1. 扩展媒体播放器的按钮功能**

如增加上 1 帧、下 1 帧按钮。

跳转到上 1 帧的控制语句为

```
prevFrame();
```

跳转到下 1 帧的控制语句为

```
nextFrame();
```

**2. 控制音乐播放**

如果要制作音乐播放器，该如何控制声音文件的播放和暂停呢？将声音文件“远方的寂寞.mp3”导入库。创建影片剪辑，放置音效。在场景中将影片剪辑拖入舞台，命名为 mc；创建按钮，给按钮命名，添加侦听器侦听是否有按钮按下。

控制音乐播放的语句为

```
mc.play();
```

控制音乐暂停的语句为

```
mc.stop();
```

**创意分享**

使用 ActionScript 3.0 交互技术，创意设计实现媒体播放器，能加载多个外部视频文件并实现播放控制。将完成后的作品发布到学习平台供同学们交流与学习。

### 3.2.2　键盘事件——走迷宫

**任务布置**

利用键盘上的方向键完成舞台对象控制，实现走迷宫游戏如图 3-2-4 所示。

**知识讲解**

(1) 在 ActionScript 3.0 中，使用 KeyboardEvent 类处理键盘操作事件，有以下两种类型事件。

KeyboardEvent.KEY_DOWN：按下键盘。

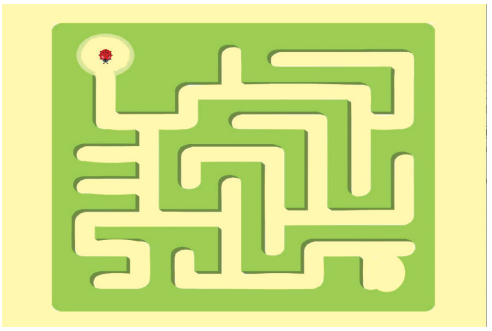

图 3-2-4　走迷宫效果图

KeyboardEvent.KEY_UP：释放键盘。

（2）键盘事件处理。首先需要添加侦听器侦听键盘事件的发生，然后需要判断哪一个键被“按下”或“弹起”，代码如下。

```
stage.addEventListener(KeyboardEvent.KEY_DOWN, xKeyDown);
stage.addEventListener(KeyboardEvent.KEY_UP, xKeyUp);

function xKeyDown(event:KeyboardEvent):void{
kb.text ="按下"+event.keyCode;
};

function xKeyUp(event:KeyboardEvent):void{
kb.text ="弹起"+event.keyCode;
};
```

（3）按键状态的确认。

```
var space_flag:Boolean =false;
stage.addEventListener(KeyboardEvent.KEY_DOWN,xKeyDown);
stage.addEventListener(KeyboardEvent.KEY_UP,xKeyUp);
stage.addEventListener(Event.ENTER_FRAME,xEnterFrame);

function xKeyDown(evt:KeyboardEvent):void{
if(evt.keyCode ==Keyboard.SPACE){
space_flag =true;
}
}

function xKeyUp(evt:KeyboardEvent):void{
if(evt.keyCode ==Keyboard.SPACE){
space_flag =false;
}
}

function xEnterFrame(evt:Event):void{
if(space_flag){
mc.gotoAndStop(2);
}else{
mc.gotoAndStop(1);
}
}
```

（4）键盘的 KeyCode 表示。舞台上有一五角星形状的影片剪辑元件，实例名称为 star，用上、下、左、右方向键控制其在舞台上移动，代码如下。

```
stage.addEventListener(KeyboardEvent.KEY_DOWN, fl);
function fl(e:KeyboardEvent):void
{
    if(e.keyCode==37)
      {star.x-=10;
        if(star.x<0) star.x=0;}
    if(e.keyCode==39)
      {star.x+=10;
       if(star.x>550) star.x=550;
      }
    if(e.keyCode==38)
      {star.y-=10;
       if(star.y<0) star.y=0;
      }
    if(e.keyCode==40)
      {star.y+=10;
       if(star.y>400) star.y=400;
      }
}
```

键盘的 KeyCode 一览表见表 3-2-1。

表 3-2-1　键盘的 KeyCode 一览表

| 大键盘的 KeyCode | | | | 小键盘的 KeyCode | | F1～F15 键 KeyCode | | 其他键的 KeyCode | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 键 | KeyCode | 键 | KeyCode | 键 | KeyCode | 键 | KeyCode | 键 | KeyCode | 键 | KeyCode |
| 0 | 48 | I | 73 | 0 | 96 | F1 | 112 | BackSpace | 8 | Insert | 45 |
| 1 | 49 | J | 74 | 1 | 97 | F2 | 113 | Tab | 9 | Delete | 46 |
| 2 | 50 | K | 75 | 2 | 98 | F3 | 114 | Clear | 12 | Help | 47 |
| 3 | 51 | L | 76 | 3 | 99 | F4 | 115 | Enter | 13 | NumLock | 144 |
| 4 | 52 | M | 77 | 4 | 100 | F5 | 116 | Shift | 16 | ： * | 186 |
| 5 | 53 | N | 78 | 5 | 101 | F6 | 117 | Ctrl | 17 | ； ＋ | 187 |
| 6 | 54 | O | 79 | 6 | 102 | F7 | 118 | Alt | 18 | －＝ | 189 |
| 7 | 55 | P | 80 | 7 | 103 | F8 | 119 | CapsLock | 20 | / ？ | 191 |
| 8 | 56 | Q | 81 | 8 | 104 | F9 | 120 | Esc | 27 | @ ` | 192 |
| 9 | 57 | R | 82 | 9 | 105 | F10 | 121 | Space | 32 | [ { | 219 |
| A | 65 | S | 83 | * | 106 | F11 | 122 | PageUp | 33 | / \| | 220 |
| B | 66 | T | 84 | ＋ | 107 | F12 | 123 | PageDown | 34 | ] } | 221 |
| C | 67 | U | 85 | Enter | 108 | F13 | 124 | End | 35 | " ' | 222 |
| D | 68 | V | 86 | － | 109 | F14 | 125 | Home | 36 | | |
| E | 69 | W | 87 | . | 110 | F15 | 126 | ← （左） | 37 | | |
| F | 70 | X | 88 | / | 111 | | | ↑ （上） | 38 | | |

续表

| 大键盘的 KeyCode | | | | 小键盘的 KeyCode | | F1～F15 键 KeyCode | | 其他键的 KeyCode | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 键 | KeyCode | 键 | KeyCode | 键 | KeyCode | 键 | KeyCode | 键 | KeyCode | 键 | KeyCode |
| G | 71 | Y | 89 | | | | | →（右） | 39 | | |
| H | 72 | Z | 90 | | | | | ↓（下） | 40 | | |

（5）键盘的定数表示。用键盘箭头移动指定的元件实例 star_1，希望每次按键时元件实例移动的像素数为 5，代码如下。

```
//定义变量,记录按键状态
var upPressed:Boolean =false;
var downPressed:Boolean =false;
var leftPressed:Boolean =false;
var rightPressed:Boolean =false;
star_1.addEventListener(Event.ENTER_FRAME, fl_MoveInDirectionOfKey_2);
stage.addEventListener(KeyboardEvent.KEY_DOWN, fl_SetKeyPressed_2);
stage.addEventListener(KeyboardEvent.KEY_UP, fl_UnsetKeyPressed_2);

function fl_MoveInDirectionOfKey_2(event:Event)
{
    if (upPressed)
    {
        star_1.y -=5;
    }
    if (downPressed)
    {
        star_1.y +=5;
    }
    if (leftPressed)
    {
        star_1.x -=5;
    }
    if (rightPressed)
    {
        star_1.x +=5;
    }
}

function fl_SetKeyPressed_2(event:KeyboardEvent):void
{
    switch (event.keyCode)
    {
        case Keyboard.UP:
        {
            upPressed =true;
            break;
        }
```

```
            case Keyboard.DOWN:
            {
                downPressed = true;
                break;
            }
            case Keyboard.LEFT:
            {
                leftPressed = true;
                break;
            }
            case Keyboard.RIGHT:
            {
                rightPressed = true;
                break;
            }
        }
    }

    function fl_UnsetKeyPressed_2(event:KeyboardEvent):void
    {
        switch (event.keyCode)
        {
            case Keyboard.UP:
            {
                upPressed = false;
                break;
            }
            case Keyboard.DOWN:
            {
                downPressed = false;
                break;
            }
            case Keyboard.LEFT:
            {
                leftPressed = false;
                break;
            }
            case Keyboard.RIGHT:
            {
                rightPressed = false;
                break;
            }
        }
    }
```

KeyCode 定数一览表见表 3-2-2。

表 3-2-2　KeyCode 定数一览表

| 键 | 定数 | 键 | 定数 | 键 | 定数 |
|---|---|---|---|---|---|
| BackSpace | Keyboard.BACKSPACE | F11 | Keyboard.F11 | NUMPAD 9 | Keyboard.NUMPAD_9 |
| CapsLock | Keyboard.CAPS_LOCK | F12 | Keyboard.F12 | NUMPAD+ | Keyboard.NUMPAD_ADD |
| Control | Keyboard.CONTROL | F13 | Keyboard.F13 | NUMPAD . | Keyboard.NUMPAD_DECIMAL |
| Delete | Keyboard.DELETE | F14 | Keyboard.F14 | NUMPAD / | Keyboard.NUMPAD_DIVIDE |
| End | Keyboard.END | F15 | Keyboard.F15 | NUMPAD Enter | Keyboard.NUMPAD_ENTER |
| Enter | Keyboard.ENTER | Home | Keyboard.HOME | NUMPAD * | Keyboard.NUMPAD_MULTIPLY |
| Escape | Keyboard.ESCAPE | Insert | Keyboard.INSERT | NUMPAD - | Keyboard.NUMPAD_SUBTRACT |
| F1 | Keyboard.F1 | NUMPAD 0 | Keyboard.NUMPAD_0 | Page Up | Keyboard.PAGE_UP |
| F2 | Keyboard.F2 | NUMPAD 1 | Keyboard.NUMPAD_1 | Page Down | Keyboard.PAGE_DOWN |
| F3 | Keyboard.F3 | NUMPAD 2 | Keyboard.NUMPAD_2 | Arrow Up | Keyboard.UP |
| F4 | Keyboard.F4 | NUMPAD 3 | Keyboard.NUMPAD_3 | Arrow Down | Keyboard.DOWN |
| F5 | Keyboard.F5 | NUMPAD 4 | Keyboard.NUMPAD_4 | Arrow Left | Keyboard.LEFT |
| F6 | Keyboard.F6 | NUMPAD 5 | Keyboard.NUMPAD_5 | Arrow Right | Keyboard.RIGHT |
| F7 | Keyboard.F7 | NUMPAD 6 | Keyboard.NUMPAD_6 | Shift | Keyboard.SHIFT |
| F8 | Keyboard.F8 | NUMPAD 7 | Keyboard.NUMPAD_7 | Space | Keyboard.SPACE |
| F9 | Keyboard.F9 | NUMPAD 8 | Keyboard.NUMPAD_8 | Tab | Keyboard.TAB |
| F10 | Keyboard.F10 | | | | |

**创意设计**

步骤 1：新建.fla 文件，修改文件大小为 1200px×800px，背景为白色。导入迷宫背景图片到舞台。

步骤 2：新建影片剪辑元件，导入瓢虫图片。将该影片剪辑拖动到舞台，修改其大小与迷宫道路相匹配，实例名称命名为 piaochong。

步骤 3：在时间轴第 1 帧关键帧处右击，打开“动作”面板，输入以下代码：

```
var upPressed:Boolean =false;//记录键盘状态
var downPressed:Boolean =false; //记录键盘状态
var leftPressed:Boolean =false; //记录键盘状态
var rightPressed:Boolean =false; //记录键盘状态
piaochong.addEventListener(Event.ENTER_FRAME, fl_MoveInDirectionOfKey_2);
//侦听 piaochong 事件
```

```
stage.addEventListener(KeyboardEvent.KEY_DOWN, fl_SetKeyPressed_2);
//侦听舞台键盘事件
stage.addEventListener(KeyboardEvent.KEY_UP, fl_UnsetKeyPressed_2);
//侦听舞台键盘事件
function fl_MoveInDirectionOfKey_2(event:Event)
{
if (upPressed)
{
piaochong.rotation=-180; //修改 piaochong 朝向向上
piaochong.y -=5; // piaochong 向上移动 5 个像素的位置
}
if (downPressed)
{ piaochong.rotation=0; //修改 piaochong 朝向向下
piaochong.y +=5;//piaochong 向下移动 5 个像素的位置
}
if (leftPressed)
{
piaochong.rotation=90; //修改 piaochong 朝向向左
piaochong.x -=5;//piaochong 向左移动 5 个像素的位置
}
if (rightPressed)
{ piaochong.rotation=-90; //修改 piaochong 朝向向右
piaochong.x +=5;//piaochong 向右移动 5 个像素的位置
}
}
function fl_SetKeyPressed_2(event:KeyboardEvent):void
{
switch (event.keyCode)
{
case Keyboard.UP:
{
upPressed =true;//记录向上的方向键按下状态
break;
}
case Keyboard.DOWN:
{
downPressed =true; //记录向下的方向键按下状态
break;
}
case Keyboard.LEFT:
{
leftPressed =true; //记录向左的方向键按下状态
break;
}
case Keyboard.RIGHT:
{
rightPressed =true; //记录向右的方向键按下状态
break;
}
}
```