第 29 章：触摸、多点触控和动作输入

Flash Player 10.1 和更高版本， **Adobe AIR 2** 和更高版本

Flash Platform 的触摸事件处理功能包括从启用触摸的设备上一个或多个接触点的输入。此外， Flash 运行时处理将多个触摸点与移动结合以创建动作的事件。换句话说， Flash 运行时解释两种输入类型：

触摸 使用启用触摸的设备中的单点设备 （例如手指、笔针或其他工具）输入。某些设备支持多个同步接触点 （使用手指或笔针）。

多点触控 使用多个同步接触点输入。

动作 由设备或操作系统解释以响应一个或多个触摸事件的输入。例如，用户同时旋转两个手指，设备或操作系统会将触摸输入解释为旋转动作。有些动作使用一个手指或触摸点执行，而有些动作需要多个触摸点。设备或操作系统确定要分配给输入的动作类型。

触摸和动作输入都可以是多点触控输入，具体取决于用户的设备。 ActionScript 提供了相应 API，用于处理触摸事件、动作事件以及针对多点触控输入单独跟踪的触摸事件。

更多帮助主题

flash.ui.Multitouch

flash.events.TouchEvent

flash.events.GestureEvent

flash.events.TransformGestureEvent

flash.events.GesturePhase

flash.events.PressAndTapGestureEvent

触摸输入的基础知识

Flash Player 10.1 和更高版本， **Adobe AIR 2** 和更高版本

当 Flash Platform 在支持触摸输入的环境中运行时， InteractiveObject 实例可以侦听触摸事件并调用处理函数。通常，可以像处理 ActionScript 中的其他事件一样处理触摸、多点触控和动作事件 （有关使用 ActionScript 进行事件处理的基本信息，请参阅第 100 页的 “ 处理事件 ”）。

但是，由于 Flash 运行时解释触摸或动作，所以运行时必须在支持触摸或多点触控输入的硬件和软件环境中运行。有关比较不同触摸屏类型的图表，请参阅第 431 页的 “ 了解输入类型 ”。此外，如果运行时在容器应用程序 （例如浏览器）内运行，则随后该容器会将输入传递到运行时。在某些情况下，虽然当前硬件和操作系统环境支持多点触控，但是包含 Flash 运行时的浏览器只解释输入却不将其传递到运行时。或者，完全忽略输入。

下图显示了从用户到运行时的输入流：

从用户到 Flash Platform 运行时的输入流

幸运的是，用于开发触摸应用程序的 ActionScript API 包括类、方法和属性，来确定运行时环境中是否支持触摸或多点触控输入。用于确定是否支持触摸输入的 API 是用于触摸事件处理的“发现 API”。

重要概念和术语

以下参考列表包含与编写触摸事件处理应用程序相关的重要术语：

发现 API 用于测试运行时环境是否支持触摸事件和其他输入模式的方法和属性。

触摸事件 使用单个接触点在启用触摸的设备上执行的输入动作。

触摸点 单个触摸事件的接触点。即使设备不支持动作输入，也可以支持多个同步触摸点。

触摸序列 表示单个触摸的生命期的一系列事件。这些事件包括一个开始事件、零个或多个移动事件和一个结束事件。

多点触控事件 使用多个接触点（例如多个手指）在启用触摸的设备上执行的输入动作。

动作事件 在启用触摸的设备上执行的输入动作（跟踪某些复杂的移动）。例如，某个动作使用两个手指触摸屏幕并同时沿抽象圆的周长移动以指示旋转。

阶段 事件流中不同的时间点（例如开始和结束）。

笔针 与启用触摸的屏幕进行交互的工具。笔针比人类的手指更精确。某些设备仅识别来自特定类型的笔针的输入。识别笔针输入的设备可能不识别多个同步接触点或手指接触。

按住并点击 特定类型的多点触控输入动作，用户用一个手指按下启用触摸的设备，然后使用另一个手指或指针设备点击。此动作通常用于在多点触控应用程序中模拟鼠标右键单击。

触摸输入 API 结构

ActionScript 触摸输入 API 旨在面向触摸输入处理取决于 Flash 运行时的硬件和软件环境这一事实。触摸输入 API 主要面向三种触摸应用程序开发的需要：发现、事件和阶段。配合使用这些 API 可以为用户生成一个可预知和可以响应的体验；即使在您开发应用程序时目标设备是未知的也是如此。

发现

发现 API 提供了在运行时测试硬件和软件环境的功能。由运行时填充的值决定在当前上下文中，触摸输入是否可用于 Flash 运行时。此外，使用发现属性和方法的集合可以将应用程序设置为响应鼠标事件（如果环境不支持某些触摸输入，则代替触摸事件）。有关详细信息，请参阅第 447 页的“触摸支持发现”。

事件

ActionScript 使用事件侦听器和事件处理函数管理触摸屏输入事件，与它对其他事件的管理方式一样。但是，还必须考虑触摸屏输入事件处理：

- 设备或操作系统可以多种方式解释触摸，可以解释为一系列触摸，也可以笼统地解释为一个动作。
- 对启用触摸的设备的单个触摸（通过手指、笔针或指针设备）也会始终调度鼠标事件。您可以处理具有 MouseEvent 类中事件类型的鼠标事件。或者，您可以将应用程序设计为只响应触摸事件。或者，您可以设计一个响应这两种类型事件的应用程序。
- 应用程序可以响应多个同步触摸事件并单独处理每个事件。

通常，使用发现 API 可以有条件地处理您的应用程序处理的事件以及处理方式。应用程序熟悉运行时环境后，在用户与应用程序交互时，它可以调用适当的处理函数或确定正确的事件对象。或者，应用程序可以指示在当前环境中无法处理特定输入，并为用户提供替代方法或信息。有关详细信息，请参阅第 448 页的“Touch 事件处理”和第 452 页的“Gesture 事件处理”。

阶段

对于触摸和多点触控应用程序，触摸事件对象包含用于跟踪用户交互阶段的属性。编写 ActionScript 以处理阶段（如用户输入的开始、更新或结束阶段），从而为用户提供反馈。响应事件阶段，以便可视对象随用户触摸和移动在屏幕上的触摸点而改变。或者，使用此阶段跟踪特定的动作属性，如动作演变。

对于触摸点事件，跟踪用户在特定的交互式对象上停留的时间。应用程序可分别跟踪多个同步触摸点阶段，并相应地对其进行处理。

对于动作，当动作发生时解释有关动作转换的特定信息。当接触点（或多个接触点）沿屏幕移动时，跟踪其坐标。

触摸支持发现

Flash Player 10.1 和更高版本， **Adobe AIR 2** 和更高版本

使用多点触控类属性设置您的应用程序处理的触摸输入的范围。然后，测试此环境以确保支持您的 ActionScript 处理的事件。具体来说，首先确定应用程序的触摸输入的类型。选项是：触摸点、动作或无（将所有触摸输入解释为鼠标单击且仅使用鼠标事件处理函数）然后，使用 Multitouch 类的属性和方法确保运行时位于支持您的应用程序需要的触摸输入的环境中。测试运行时环境是否支持这些类型的触摸输入（例如，是否能够解释动作）并相应地做出响应。

注：Multitouch 类属性是静态属性，且不属于任何类的实例。将其与语法 Multitouch.property 一起使用，例如：

```
var touchSupport:Boolean = Multitouch.supportsTouchEvents;
```

设置输入类型

Flash 运行时必须知道要解释的触摸输入的类型，因为触摸事件可能包含多个元素或阶段。如果手指仅触摸启用触摸的屏幕，则运行时是否会调度触摸事件？或者，运行时是否会等待动作？或者，运行时是否将触摸作为鼠标按下事件进行跟踪？支持触摸输入的应用程序必须确定其针对 Flash 运行时处理的触摸事件的类型。使用 Multitouch.inputMode 属性为运行时确定触摸输入类型。输入模式可以是以下三个选项之一：

无 对触摸事件不提供特殊处理。设置：Multitouch.inputMode=MultitouchInputMode.NONE 并使用 MouseEvent 类处理输入。

单个触摸点 分别解释所有触摸输入，并跟踪和处理所有触摸点。设置：Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT 并使用 TouchEvent 类处理输入。

动作输入 设备或操作系统将输入解释为手指沿屏幕移动的一种复杂形式。设备或操作系统将移动集体分配给单个动作输入事件。设置：Multitouch.inputMode=MultitouchInputMode.GESTURE 并使用 TransformGestureEvent、PressAndTapGestureEvent 或 GestureEvent 类处理输入。

有关在处理触摸事件之前使用 Multitouch.inputMode 属性设置输入类型的示例，请参阅第 448 页的 “Touch 事件处理 ”。

测试是否支持触摸输入

Multitouch 类的其他属性提供了相应的值，用于微调您的应用程序以支持当前环境中的触摸。Flash 运行时填充允许同时进行的触摸点的数量值或可用的动作的数量值。如果运行时位于不支持您的应用程序所需的触摸事件处理的环境中，则为用户提供其他处理。例如，提供鼠标事件处理或有关在当前环境中可用或不可用的功能的信息。

还可以将 API 用于键盘、触摸和鼠标支持，请参阅第 431 页的 “ 了解输入类型 ”。

有关兼容性测试的详细信息，请参阅第 455 页的 “ 疑难解答 ”。

Touch 事件处理

Flash Player 10.1 和更高版本， **Adobe AIR 2** 和更高版本

在 ActionScript 中，基本触摸事件与其他事件 （如鼠标事件）的处理方式相同。您可以侦听由 TouchEvent 类中的事件类型定义的一系列触摸事件。

注：对于多个触摸点输入 （例如，使用多个手指触摸设备），第一个接触点将调度鼠标事件和触摸事件。

处理基本触摸事件：

1. 通过将 flash.ui.Multitouch.inputMode 属性设置为 MultitouchInputMode.TOUCH_POINT，可以将您的应用程序设置为处理触摸事件。
2. 将事件侦听器附加到从 InteractiveObject 类继承属性的类实例，如 Sprite 或 TextField。
3. 指定要处理的触摸事件的类型。
4. 调用事件处理函数以执行某些操作，从而响应事件。

例如，当在启用触摸的屏幕上点击在 mySprite 上绘制的正方形时，以下代码显示一则消息：

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taphandler);

function taphandler(evt:TouchEvent): void {
    myTextField.text = "I've been tapped";
    myTextField.y = 50;
    addChild(myTextField);
}
```

Touch 事件属性

发生某个事件时，将创建一个事件对象。 TouchEvent 对象包含有关触摸事件的位置和条件的信息。您可以使用事件对象的属性检索该信息。

例如，以下代码创建 TouchEvent 对象 evt，然后在文本字段中显示事件对象的 stageX 属性（发生触摸的舞台空间中该点的 x 坐标）：

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taphandler);

function taphandler(evt:TouchEvent): void {
myTextField.text = evt.stageX.toString;
myTextField.y = 50;
addChild(myTextField);
}
```

请参阅通过事件对象提供的属性的 TouchEvent 类。

注：并非所有运行时环境都支持所有 TouchEvent 属性。例如，并非所有启用触摸的设备都能够检测用户应用到触摸屏的压力。因此，这些设备不支持 TouchEvent.pressure 属性。尝试测试是否支持特定属性以确保您的应用程序能够正常工作，有关详细信息，请参阅第 455 页的 “ 疑难解答 ”。

触摸事件阶段

跟踪 InteractiveObject 内外的各种舞台中的触摸事件，就像您跟踪鼠标事件一样。并且，跟踪触摸交互开头、中间和结尾中的触摸事件。 TouchEvent 类提供了用于处理 touchBegin、 touchMove 和 touchEnd 事件的值。

例如，您可以使用 touchBegin、 touchMove 和 touchEnd 事件在用户触摸和移动显示对象时为其提供可视反馈：

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);
var myTextField:TextField = new TextField();
myTextField.width = 200;
myTextField.height = 20;
addChild(myTextField);

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
stage.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
stage.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);
function onTouchBegin(event:TouchEvent) {
    myTextField.text = "touch begin" + event.touchPointID;
}
function onTouchMove(event:TouchEvent) {
    myTextField.text = "touch move" + event.touchPointID;
}
function onTouchEnd(event:TouchEvent) {
    myTextField.text = "touch end" + event.touchPointID;
}
```

注：将初始触摸侦听器附加到 mySprite，但不要附加用于移动和结束触摸事件的侦听器。如果用户的手指或指针设备先于显示对象移动，则舞台将继续侦听触摸事件。

触摸点 ID

编写用于响应触摸输入的应用程序需要 TouchEvent.touchPointID 属性。 Flash 运行时为每个触摸点分配一个唯一的 touchPointID 值。当应用程序响应触摸输入阶段或触摸输入的移动时，请先检查 touchPointID，然后再处理该事件。 Sprite 类的触摸输入拖动方法将 touchPointID 属性用作参数，以便处理正确的输入实例。touchPointID 属性确保事件处理函数响应正确的触摸点。否则，事件处理函数将响应设备上触摸事件类型的任何实例 （例如，所有 touchMove 事件），从而产生不可预测的行为。此属性在用户拖动对象时特别重要。

使用 touchPointID 属性可以管理整个触摸序列。触摸序列包含一个 touchBegin 事件、 0 个或多个 touchMove 事件和一个 touchEnd 事件，所有这些事件都具有相同的 touchPointID 值。

以下示例建立一个变量 touchMoveID，以在响应触摸移动事件之前测试 touchPointID 值是否正确。否则，其他触摸输入也会触发该事件处理函数。请注意，移动和结束阶段的侦听器位于舞台 （而不是显示对象）上。舞台将侦听移动或结束阶段，以防用户的触摸移动到显示对象边界以外。

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);
var myTextField:TextField = new TextField();
addChild(myTextField);
myTextField.width = 200;
myTextField.height = 20;
var touchMoveID:int = 0;

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
function onTouchBegin(event:TouchEvent) {
    if(touchMoveID != 0) {
        myTextField.text = "already moving. ignoring new touch";
        return;
    }
    touchMoveID = event.touchPointID;

    myTextField.text = "touch begin" + event.touchPointID;
    stage.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
    stage.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);
}
function onTouchMove(event:TouchEvent) {
    if(event.touchPointID != touchMoveID) {
        myTextField.text = "ignoring unrelated touch";
        return;
    }
    mySprite.x = event.stageX;
    mySprite.y = event.stageY;
    myTextField.text = "touch move" + event.touchPointID;
}
function onTouchEnd(event:TouchEvent) {
    if(event.touchPointID != touchMoveID) {
        myTextField.text = "ignoring unrelated touch end";
        return;
    }
    touchMoveID = 0;
    stage.removeEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
    stage.removeEventListener(TouchEvent.TOUCH_END, onTouchEnd);
    myTextField.text = "touch end" + event.touchPointID;
}
```

触摸和拖动

Flash Player 10.1 和更高版本， **Adobe AIR 2** 和更高版本

以下两个方法已添加到 Sprite 类，以便为支持触摸点输入且启用触摸的应用程序提供更多支持：Sprite.startTouchDrag() 和 Sprite.stopTouchDrag()。这些方法的作用与针对鼠标事件的 Sprite.startDrag() 和 Sprite.stopDrag() 的作用相同。然而，请注意，Sprite.startTouchDrag() 和 Sprite.stopTouchDrag() 方法都将 touchPointID 值用作参数。

运行时将 touchPointID 值分配给触摸事件的事件对象。当环境支持多个同步触摸点时 （即使不处理动作），使用此值响应特定的触摸点。有关 touchPointID 属性的详细信息，请参阅第 450 页的 “ 触摸点 ID”。

以下代码显示触摸事件的简单开始拖动事件处理函数和停止拖动事件处理函数。变量 bg 是一个包含 mySprite 的显示对象：

```
mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
mySprite.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);

function onTouchBegin(e:TouchEvent) {
    e.target.startTouchDrag(e.touchPointID, false, bg.getRect(this));
    trace("touch begin");
}

function onTouchEnd(e:TouchEvent) {
    e.target.stopTouchDrag(e.touchPointID);
    trace("touch end");
}
```

并且，以下代码显示更高级的示例，该示例将拖动与触摸事件阶段组合在一起：

```
Multitouch.inputMode = MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_BEGIN, onTouchBegin);
mySprite.addEventListener(TouchEvent.TOUCH_MOVE, onTouchMove);
mySprite.addEventListener(TouchEvent.TOUCH_END, onTouchEnd);

function onTouchBegin(evt:TouchEvent) {
    evt.target.startTouchDrag(evt.touchPointID);
    evt.target.scaleX *= 1.5;
    evt.target.scaleY *= 1.5;
}

function onTouchMove(evt:TouchEvent) {
    evt.target.alpha = 0.5;
}

function onTouchEnd(evt:TouchEvent) {
    evt.target.stopTouchDrag(evt.touchPointID);
    evt.target.width = 40;
    evt.target.height = 40;
    evt.target.alpha = 1;
}
```

Gesture 事件处理

Flash Player 10.1 和更高版本， **Adobe AIR 2** 和更高版本

处理动作事件的方式与处理基本触摸事件相同。您可以侦听由 TransformGestureEvent 类、 GestureEvent 类和 PressAndTapGestureEvent 类中的事件类型常数定义的一系列动作事件。

要处理动作触摸事件，请执行下列操作：

1 通过将 flash.ui.Multitouch.inputMode 属性设置为 MultitouchInputMode.GESTURE 将您的应用程序设置为处理动作输入。

2 将事件侦听器附加到从 InteractiveObject 类继承属性的类实例，如 Sprite 或 TextField。

3 指定要处理的动作事件的类型。

4 调用事件处理函数以执行某些操作，从而响应事件。

例如，当在启用触摸的屏幕上点击在 mySprite 上绘制的正方形时，以下代码显示一则消息：

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TransformGestureEvent.GESTURE_SWIPE, swipehandler);

function swipehandler(evt:TransformGestureEvent): void {
myTextField.text = "I've been swiped";
myTextField.y = 50;
addChild(myTextField);
}
```

以相同的方式处理二指点击事件，但要使用 GestureEvent 类：

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(GestureEvent.GESTURE_TWO_FINGER_TAP, taphandler);

function taphandler(evt:GestureEvent): void {
myTextField.text = "I've been two-finger tapped";
myTextField.y = 50;
addChild(myTextField);
}
```

也以相同的方式处理按住轻敲事件，但要使用 PressAndTapGestureEvent 类：

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField();

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(PressAndTapGestureEvent.GESTURE_PRESS_AND_TAP, taphandler);

function taphandler(evt:PressAndTapGestureEvent): void {
myTextField.text = "I've been press-and-tapped";
myTextField.y = 50;
addChild(myTextField);
}
```

注：所有运行时环境中并非支持所有 GestureEvent、 TransformGestureEvent 和 PressAndTapGestureEvent 事件类型。例如，并非所有启用触摸的设备都能够检测多个手指滑动。因此，这些设备不支持 InteractiveObject gestureSwipe 事件。尝试测试是否支持特定事件以确保您的应用程序能够正常工作，有关详细信息，请参阅第 455 页的 “ 疑难解答 ”。

Gesture 事件属性

动作事件的事件属性范围比基本触摸事件小。您可以通过事件处理函数中的事件对象以相同的方式访问这些属性。

例如，以下代码在用户对 mySprite 执行旋转动作时会旋转该对象。该文本字段显示自最后一个动作之后的旋转量 （测试此代码时，将其多旋转几次以查看值更改）：

```
Multitouch.inputMode=MultitouchInputMode.GESTURE;

var mySprite:Sprite = new Sprite();
var mySpriteCon:Sprite = new Sprite();
var myTextField:TextField = new TextField();
myTextField.y = 50;
addChild(myTextField);

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(-20,-20,40,40);
mySpriteCon.addChild(mySprite);
mySprite.x = 20;
mySprite.y = 20;
addChild(mySpriteCon);

mySprite.addEventListener(TransformGestureEvent.ROTATE, rothandler);

function rothandler(evt:TransformGestureEvent): void {
evt.target.parent.rotationZ += evt.target.rotation;
myTextField.text = evt.target.parent.rotation.toString();
}
```

注：所有运行时环境中并非支持所有 TransformGestureEvent 属性。例如，并非所有启用触摸的设备都能够检测屏幕上的动作旋转。因此，这些设备不支持 TransformGestureEvent.rotation 属性。尝试测试是否支持特定属性以确保您的应用程序能够正常工作，有关详细信息，请参阅第 455 页的 “ 疑难解答 ”。

Gesture 阶段

另外，可以跟踪阶段中的动作事件，因此您可以在发生动作时跟踪相应的属性。例如，您可以在使用滑动动作移动对象时跟踪 X 坐标。当滑动完成后，使用这些值绘制一条在其路径中经过所有点的直线。或者，在使用全景动作沿屏幕拖动显示对象时以可视方式更改该对象。当全景动作完成后，再次更改该对象。

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;
var mySprite = new Sprite();
mySprite.addEventListener(TransformGestureEvent.GESTURE_PAN , onPan);
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0, 0, 40, 40);
var myTextField = new TextField();
myTextField.y = 200;
addChild(mySprite);
addChild(myTextField);

function onPan(evt:TransformGestureEvent):void {

    evt.target.localX++;

if (evt.phase==GesturePhase.BEGIN) {
    myTextField.text = "Begin";
    evt.target.scaleX *= 1.5;
    evt.target.scaleY *= 1.5;
}
if (evt.phase==GesturePhase.UPDATE) {
    myTextField.text = "Update";
    evt.target.alpha = 0.5;
}
if (evt.phase==GesturePhase.END) {
    myTextField.text = "End";
    eEnd.target.width = 40;
    eEnd.target.height = 40;
    eEnd.target.alpha = 1;
}
}
```

注：更新阶段的频率取决于运行时的环境。某些操作系统和硬件组合根本不会传递更新。

动作阶段对于简单的动作事件为 “all”

某些动作事件对象不跟踪单个动作事件阶段，而是使用值 all 填充事件对象的 phase 属性。简单的动作滑动和二指点击不跟踪包含多个阶段的事件。侦听 gestureSwipe 或 gestureTwoFingerTap 事件的 InteractiveObject 的事件对象的 phase 属性在调度事件后始终为 all：

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;
var mySprite = new Sprite();
mySprite.addEventListener(TransformGestureEvent.GESTURE_SWIPE, onSwipe);
mySprite.addEventListener(GestureEvent.GESTURE_TWO_FINGER_TAP, onTwoTap);
mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0, 0, 40, 40);
var myTextField = new TextField();
myTextField.y = 200;
addChild(mySprite);
addChild(myTextField);

function onSwipe(swipeEvt:TransformGestureEvent):void {
    myTextField.text = swipeEvt.phase // Output is "all"
}
function onTwoTap(tapEvt:GestureEvent):void {
    myTextField.text = tapEvt.phase // Output is "all"
}
```

疑难解答

Flash Player 10.1 和更高版本， **Adobe AIR 2** 和更高版本

支持触摸输入的硬件和软件的变化速度很快。此参考不能维护支持多点触控的操作系统和软件组合上的每个设备的列表。但是，它提供了有关如何使用发现 API 来确定是否可以在支持多点触控的设备上部署您的应用程序的指导，还提供了用于疑难解答 ActionScript 代码的技巧。

Flash 运行时基于传递给运行时的设备、操作系统或包含的软件 （如浏览器）的信息来响应触摸事件。对软件环境的这种依赖关系使文档多点触控兼容变得很复杂。某些设备对动作或触摸动作的解释不同于其他设备。旋转是由两个手指同时旋转定义的吗？旋转是指使用一个手指在屏幕上绘制圆形吗？根据硬件和软件环境，旋转动作可能会完全不同。因此，当用户输入时，设备会告知操作系统，然后操作系统将该信息传递给运行时。如果运行时位于浏览器之内，则有时浏览器软件会解释动作或触摸事件，但不会将输入传递给运行时。此行为类似于 “ 热键 ” 行为：尝试使用特定的键组合使 Flash Player 在浏览器内部执行某些操作且浏览器始终打开一个菜单。

如果单独的 API 和类与特定的操作系统不兼容，会显示出来。您可以从以下位置了解各个 API 条目，首先了解 Multitouch 类：http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/ui/Multitouch.html。

以下是一些常用的动作和触摸说明：

平移 从左到右或从右到左移动手指。某些设备要求两个手指平移。

旋转 按下两个手指，然后绕一个圆移动 （就像它们同时在跟踪平面上一个虚构的圆一样）。将轴点设置为这两个手指触摸点之间的中点。

滑动 从左到右或从右到左、从上到下或从下到上快速移动三个手指。

缩放 按下两个手指，然后使其互相远离进行放大，互相靠近进行缩小。

按住并点击 移动或按下一个手指，然后使用另一个手指点击平面。

每个设备都有自己的文档，其中介绍有关该设备支持的动作以及如何在该设备上执行各个动作的信息。通常，在动作之间，用户必须移开与设备接触的所有手指，具体取决于操作系统。

如果您发现您的应用程序无法响应触摸事件或动作，请执行以下测试：

1. 您是否将触摸或动作事件的事件侦听器附加到从 InteractiveObject 类继承的 Object 类？只有 InteractiveObject 实例可以侦听触摸和动作事件
2. 是否正在 Flash Professional CS5 中测试您的应用程序？如果是，尝试发布和测试该应用程序，因为 Flash Professional 可截获交互。
3. 首先启动简单的事件并查看哪些可以正常工作 （以下代码示例来自 Multitouch.inputMode 的 API 条目：

```
Multitouch.inputMode=MultitouchInputMode.TOUCH_POINT;
var mySprite:Sprite = new Sprite();
var myTextField:TextField = new TextField()

mySprite.graphics.beginFill(0x336699);
mySprite.graphics.drawRect(0,0,40,40);
addChild(mySprite);

mySprite.addEventListener(TouchEvent.TOUCH_TAP, taplistener);

function taplistener(e:TouchEvent): void {
    myTextField.text = "I've been tapped";
    myTextField.y = 50;
    addChild(myTextField);
}
```

点击此矩形。如果此示例可以正常运行，即可了解您的环境支持简单的轻敲。然后您可以尝试执行更复杂的处理。

测试动作支持更复杂。单个设备或操作系统支持任何动作输入组合或不支持任何动作输入。

以下是一个简单的缩放动作测试：

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;

stage.addEventListener(TransformGestureEvent.GESTURE_ZOOM , onZoom);
var myTextField = new TextField();
myTextField.y = 200;
myTextField.text = "Perform a zoom gesture";
addChild(myTextField);

function onZoom(evt:TransformGestureEvent):void {
    myTextField.text = "Zoom is supported";
}
```

对设备执行缩放动作，并查看该文本字段是否填充了支持缩放消息。事件侦听器已添加到舞台中，因此您可以对测试应用程序的任何部分执行动作。

以下是一个简单的全景动作测试：

```
Multitouch.inputMode = MultitouchInputMode.GESTURE;

stage.addEventListener(TransformGestureEvent.GESTURE_PAN , onPan);
var myTextField = new TextField();
myTextField.y = 200;
myTextField.text = "Perform a pan gesture";
addChild(myTextField);

function onPan(evt:TransformGestureEvent):void {
    myTextField.text = "Pan is supported";
}
```

对设备执行全景动作，并查看该文本字段是否填充了支持全景消息。事件侦听器已添加到舞台中，因此您可以对测试应用程序的任何部分执行动作。

某些操作系统和设备组合同时支持这两种动作，某些只支持一种，某些这两种都不支持。测试您的应用程序的部署环境以了解情况。

已知问题

以下是与触摸屏输入相关的已知问题：

1 Windows Mobile 操作系统上的 Mobile Internet Explorer 会自动缩放 SWF 文件内容：

通过将下列代码添加到承载 SWF 文件的 HTML 页面来覆盖此 Internet Explorer 缩放行为：

```
<head>
<meta name="viewport" content="width=device-width, height=device-height, initial-scale=1.0">
</head>
```

2 对于 Windows 7 （也可能是其他操作系统），用户必须在动作之间将指针设备 （或手指）从屏幕上移开。例如：旋转和缩放图像：

- 执行旋转动作。
- 将您的手指从屏幕上移开。
- 将您的手指放回到屏幕上并执行缩放动作。

3 对于 Windows 7 （可能是其他操作系统），如果用户非常快速地执行手势，则旋转和缩放手势并不始终生成 “update” 相位。

4 Windows 7 Starter Edition 不支持多点触控。有关详细信息，请参阅 AIR Labs 论坛：
http://forums.adobe.com/thread/579180?tstart=0

5 对于 Mac OS 10.5.3 及更高版本， Multitouch.supportsGestureEvents 值始终为 true，即使硬件不支持动作事件也是如此。