

「Multitouch and gesture support on the Flash Platform」の日本語訳

本ドキュメントは、Adobe サイトで公開されている記事「Multitouch and gesture support on the Flash Platform」をヒム・カンパニー 永井勝則が自主的に日本語に訳したものです。

<http://www.himco.jp/>

[knagai@himco.jp](mailto:knagai@himco.jp)

(2010/10)

本ドキュメントの原文は

[http://www.adobe.com/devnet/flash/articles/multitouch\\_gestures.html](http://www.adobe.com/devnet/flash/articles/multitouch_gestures.html)

です。

## Flash Platform におけるマルチタッチとジェスチャのサポート

Christian Cantrell

2009/11/17

求められる予備知識

本記事は ActionScript 3.0 に習熟した開発者向けです。

必要な製品

Flash Player

Adobe AIR

サンプルファイル

multitouch\_gesture\_examples.zip

ユーザーレベル

中級

本記事は、Adobe Flash Player 10.1 beta と Adobe AIR 2 beta で使用できるようになった新しいマルチタッチ API に関する記事です。今後、マルチタッチが可能なプラットフォームが増え、タッチ操作によるデバイスとの相互作用の体験を求めるユーザーが増加していくこととなりますが、Flash Platform は開発者のみなさんに対し、ユビキタスなタッチ体験を可能にする、非常に簡単で効率的な方法を提供していきます。

### 1. マルチタッチとジェスチャの定義

マルチタッチという用語は、画面上の物理的な接触や移動を判定できるだけでなく、複数の同時的な接触や移動を判定し追跡できる機能を言います。タッチイベントはマウスイベントと似ていますが、同時に複数のタッチイベントを受け取り追跡できる点が異なります。したがってタッチイベントでは、ホバリングなどマウス特有の概念がサポートされません。

ジェスチャは複数のタッチイベントを1つのイベントに統合したものです。例としては、画像を拡大縮小する“ピンチ”やリストから項目を消去する“スワイプ”があります。プラットフォームの中には、ジェスチャの概念が明示的にサポートされ、それにより開発者がジェスチャを判定し応答するために必要になる作業を減らすことができるものもありますが、開発者自身で複数のタッチイベントを捕捉し、それをジェスチャに統合する必

要のあるものもあります。Flash Platform は異なるプラットフォームでほぼ共通するジェスチャを自動的に統合しますが、同時にその統合に必要な API も提供しています。

マルチタッチの技術はこれまでも存在していましたが、その概念が実際に広まったのは Apple の iPhone の登場からです。デバイスと、ボタンやスタイラスを通してでなく、タッチで直接相互作用できるというメリットが明白になったので、今やタッチ動作はデスクトップコンピューティングの分野にも進出しています。Windows 7 はマルチタッチを最初からサポートしているので、ヒューレットパッカーは 2007 年からタッチ可能な TouchSmart コンピュータの販売を開始し、Microsoft は 2008 年にジェスチャとタッチ中心の Microsoft Surface を世に送り出しました。さらに Apple は MacBook Air でマルチタッチのトラックパッドを登場させ、以来その技術をラップトップの製品群にも組み込んでいます。Apple の最新のマウス、Magic Mouse では限定的ながらもジェスチャがサポートされています。タッチやマルチタッチ、ジェスチャや触角にもとづくコンピューティングは今や非常に重要になりつつあり、ほとんどすべてのハイエンドな携帯デバイスでは、複数の相互作用モデルがサポートされています。

## 2. マルチタッチとジェスチャのサポート

Flash Platform は現在、ブラウザ内の Flash Player 10.1、iPhone と iPod touch 用にパブリッシュされた SWF、AIR 2 においてマルチタッチとジェスチャをサポートしていますが、マルチタッチのサポートはハードウェアの機能とターゲットとするプラットフォームとの組み合わせに依存します。

以下のリストは、本記事執筆時点でマルチタッチとジェスチャがサポートされる環境を示しています。

### マルチタッチ

- Window 7 とそれ以降(タッチ可能なハードウェアとの組み合わせによる)での、ブラウザベースの Flash Player 10.1 の SWF コンテンツと AIR 2 アプリケーション
- iPhone OS 3.0 とそれ以降

### ジェスチャ

- Window 7 とそれ以降(タッチ可能なハードウェアとの組み合わせによる場合)での、ブラウザベースの Flash Player 10.1 の SWF コンテンツと AIR 2 アプリケーション
- Mac OS X 10.5.3 かそれ以降が動作する Mac(マルチタッチトラックパッドとの組み合わせによる)
- iPhone OS 3.0 とそれ以降

以下は、ジェスチャイベントをサポートするプラットフォームとイベントの種類のリストです。

### Windows 7

- TransformGestureEvent.GESTURE\_PAN
- TransformGestureEvent.GESTURE\_ROTATE
- TransformGestureEvent.GESTURE\_ZOOM
- GestureEvent.GESTURE\_TWO\_FINGER\_TAP
- PressAndTapGestureEvent.GESTURE\_PRESS\_AND\_TAP

#### Mac OS X 10.5.3 とそれ以降

- TransformGestureEvent.GESTURE\_PAN
- TransformGestureEvent.GESTURE\_ROTATE
- TransformGestureEvent.GESTURE\_SWIPE
- TransformGestureEvent.GESTURE\_ZOOM

#### iPhone と iPod touch

- TransformGestureEvent.GESTURE\_PAN
- TransformGestureEvent.GESTURE\_ROTATE
- TransformGestureEvent.GESTURE\_SWIPE
- TransformGestureEvent.GESTURE\_ZOOM

#### Windows Mobile 6.5

- TransformGestureEvent.GESTURE\_PAN

### 3. Multitouch クラス

マルチタッチとジェスチャはすべて、Multitouch クラスから始まります。このクラスには、マルチタッチやジェスチャを可能にするアプリケーションのオーサリングで必要な、重要なプロパティが含まれています。

#### サポートの判定

マルチタッチイベントにせよジェスチャイベントにせよ、その登録を行う前に、Multitouch.supportsGestureEvents や Multitouch.supportsTouchEvent プロパティを使って、コンテンツの動作するデバイスがアプリケーションに必要なイベントのタイプをサポートするかどうかを調べるのはよい考えです。特に iPhone 用のアプリケーションを記述するときには、これらのプロパティを使う必要はないかもしれませんが、複数の場所で動作させたいコンテンツを記述する場合には、これらのプロパティが非常に有効になります。

#### 入力モードの設定

Multitouch.inputMode プロパティの設定は、ランタイムに対し、みなさんが受け取りたいイベントは何か

を伝えるために必要です。これには次の3つの選択肢があります。

- ・MultitouchInputMode.GESTURE: このモードは、マルチタッチイベントをジェスチャイベントに統合したい場合に使用します。タップのような単純なイベントはマウスイベントと解釈されます。
- ・MultitouchInputMode.TOUCH\_POINT: このモードは、関心がタッチイベントにのみあり、マウスやジェスチャイベントにない場合に使用します。このモードを使用すると、ランタイムでサポートされないジェスチャをサポートしたいとき、またはマルチタッチとジェスチャ両方をサポートする必要のある場合、自作のジェスチャに統合することができます。
- ・MultitouchInputMode.NONE: このモードは、すべてのタッチをマウスイベントとして処理したい場合に使用します。これは、タッチが有効なデバイスと有効でないデバイスで動作させたいコンテンツに適しています。コードを分けて記述する必要がなくなります。

Note: Multitouch.inputMode はまた、現在の相互作用モードの検出を行うとき、いつでも使用できます。

#### サポートされるジェスチャの検出

Multitouch.inputMode プロパティを GESTURE に設定するときには、Multitouch.supportedGestures プロパティを調べて、動作させる特定のデバイスでどのジェスチャがサポートされるのか調べるのはよい考えです。このプロパティは、GestureEvent、PressAndTapGestureEvent、TransformGestureEvent のイベントタイプ定数から成る、ストリングの Vector を返します。これはすなわち、この Vector 内のストリングを、サポートされるジェスチャイベントの登録にそのまま使用できる、ということです。

#### サポートされるタッチポイント数(接触点の数)の検出

Multitouch.inputMode プロパティを TOUCH\_POINT に設定するときには、現在動作しているデバイスでサポートされるタッチポイント数を教えてくれる Multitouch.maxTouchPoints を調べるのはよい考えです。デバイスがサポートするよりも多いタッチポイント数を処理しようとする、意図しない結果を招く場合があります(すべての有効なタッチポイントが失われるなど)。

#### タッチイベントとジェスチャイベントの登録

タッチイベントとジェスチャイベントは InteractiveObject によって送出されます。これはつまり、SimpleButton や TextField、Loader、Sprite、Stage など、InteractiveObject を継承するオブジェクトならどのオブジェクトにもタッチやジェスチャのイベントが登録できるということです。

## 4. マルチタッチイベントの処理

InteractiveObject や InteractiveObject を拡張するすべてのクラスには、8つの異なるタッチイベントが登録できます。

TOUCH_BEGIN	タッチイベントの開始を示します。
TOUCH_END	タッチイベントの終了を示します。
TOUCH_MOVE	タッチポイントが移動していることを示します（言い方を変えると、ユーザーは画面上で自分の指を動かしているということです）。
TOUCH_OVER	タッチポイントが InteractiveObject に入ったことを示します。これは、TOUCH_ROLL_OVER と異なり、InteractiveObject の各子にも発射されます。
TOUCH_ROLL_OVER	タッチポイントが InteractiveObject に入ったことを示します。これは、TOUCH_OVER と異なり、InteractiveObject の各子には発射されません。つまりこれを知るのはコンテナです。
TOUCH_OUT	タッチポイントが InteractiveObject から離れたことを示します。これは、TOUCH_ROLL_OUT と異なり、InteractiveObject の各子にも発射されます。
TOUCH_ROLL_OUT	タッチポイントが InteractiveObject から離れたことを示します。これは、TOUCH_OUT と異なり、InteractiveObject の各子には発射されません。つまりこれを知るのはコンテナです。
TOUCH_TAP	タップ(素早い TOUCH_BEGIN と TOUCH_END)が発生したことを示します。

#### タッチイベントのプロパティ

タッチイベントにはマウスイベントと同じプロパティがありますが、タッチの相互作用特有の追加的なプロパティもあります。

・isPrimaryTouchPoint: 最初の接触点がマウスイベントにマッピング(位置づけ)されるかどうかを示します。これを知ることは、マウスとタッチのイベント両方を処理するアプリケーションのインスタンスで重要になります。

・pressure: 0.0 から 1.0 の間の値で、デバイスへの接触圧を示します。これは特定のデバイスやプラットフォームでサポートされるプロパティで、マルチタッチが可能なすべてのデバイスでサポートされるわけではありません。

・sizeX: 接触領域の幅(たとえばタッチしている指の横方向の長さ)。マルチタッチが可能なすべてのデバイスでサポートされるわけではありません。

・sizeY: 接触領域の高さ(たとえばタッチしている指の縦方向の長さ)。マルチタッチが可能なすべてのデバイスでサポートされるわけではありません。

・touchPointID: タッチポイントに割り当てられる一意の識別番号。これは、複数のタッチポイントを同時に追跡するときに有効です。

### Sprite の新しいタッチ機能

タッチに関係する新しいクラスのほか、Sprite にもタッチに関係する新しい機能が2つ加わりました。

- ・startTouchDrag: タッチ可能なデバイスで、ユーザーが特定のSpriteをドラッグできます。
- ・stopTouchDrag: startTouchDrag()メソッドを終了します。

### ここまでのまとめ

以下はサンプルの MultitouchExample のコードの全文です。ここではこれまで取り上げてきたいいくつかの API を使って、タッチイベントの登録と TOUCH\_BEGIN イベントの処理の方法を示しています。dot という名前のSpriteを、this.dots というクラスメンバーの Object 変数に、イベントの touchPointID をキーとして保持している点に注目してください。この方法によって onTouchEnd()関数でその touchPointID が調べられるので、表示リストからの削除が可能になります。

### MultitouchExample.as

```
package
{
    import flash.display.Sprite;
    import flash.events.TouchEvent;
    import flash.text.AntiAliasType;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.ui.Multitouch;
    import flash.ui.MultitouchInputMode;

    [SWF(width=320, height=460, frameRate=24,
backgroundcolor=0xEB7F00)]
    public class MultitouchExample extends Sprite
    {
        private var dots:Object;
        private var labels:Object;
        private var labelFormat:TextFormat;
        private var dotCount:uint;
        private var dotsLeft:TextField;
```

```

private static const LABEL_SPACING:uint = 15;

public function MultitouchExample()
{
    super();

    this.labelFormat = new TextFormat();
    labelFormat.color = 0xACF0F2;
    labelFormat.font = "Helvetica";
    labelFormat.size = 11;

    this.dotCount = 0;

    this.dotsLeft = new TextField();
    this.dotsLeft.width = 300;
    this.dotsLeft.defaultTextFormat = this.labelFormat;
    this.dotsLeft.x = 3;
    this.dotsLeft.y = 0;
    this.stage.addChild(this.dotsLeft);
    this.updateDotsLeft();

    this.dots = new Object();
    this.labels = new Object();

    Multitouch.inputMode =
MultitouchInputMode.TOUCH_POINT;

    this.stage.addEventListener(TouchEvent.TOUCH_BEGIN,
onTouchBegin);

    this.stage.addEventListener(TouchEvent.TOUCH_MOVE,
onTouchMove);

    this.stage.addEventListener(TouchEvent.TOUCH_END,
onTouchEnd);
}

```

```

private function onTouchBegin(e:TouchEvent):void
{
    if (this.dotCount == Multitouch.maxTouchPoints)
return:
        var dot:Sprite = this.getCircle();
        dot.x = e.stageX;
        dot.y = e.stageY;
        this.stage.addChild(dot);
        dot.startTouchDrag(e.touchPointID, true);
        this.dots[e.touchPointID] = dot;

        ++this.dotCount;

        var label:TextField = this.getLabel(e.stageX + ", " +
e.stageY);

        label.x = 3;
        label.y = this.dotCount * LABEL_SPACING;
        this.stage.addChild(label);
        this.labels[e.touchPointID] = label;

        this.updateDotsLeft();
}

private function onTouchMove(e:TouchEvent):void
{
    var label:TextField = this.labels[e.touchPointID];
    label.text = (e.stageX + ", " + e.stageY);
}

private function onTouchEnd(e:TouchEvent):void
{
    var dot:Sprite = this.dots[e.touchPointID];
    var label:TextField = this.labels[e.touchPointID];

    this.stage.removeChild(dot);
}

```

```

        this.stage.removeChild(label);

        delete this.dots[e.touchPointID];
        delete this.labels[e.touchPointID];

        --this.dotCount;

        this.updateDotsLeft();
    }

    private function getCircle(circumference:uint = 40):Sprite
    {
        var circle:Sprite = new Sprite();
        circle.graphics.beginFill(0x1695A3);
        circle.graphics.drawCircle(0, 0, circumference);
        return circle;
    }

    private function getLabel(initialText:String):TextField
    {
        var label:TextField = new TextField();
        label.defaultTextFormat = this.labelFormat;
        label.selectable = false;
        label.antiAliasType = AntiAliasType.ADVANCED;
        label.text = initialText;
        return label;
    }

    private function updateDotsLeft():void
    {
        this.dotsLeft.text = "Touches Remaining: " +
(Multitouch.maxTouchPoints - this.dotCount);
    }
}

```

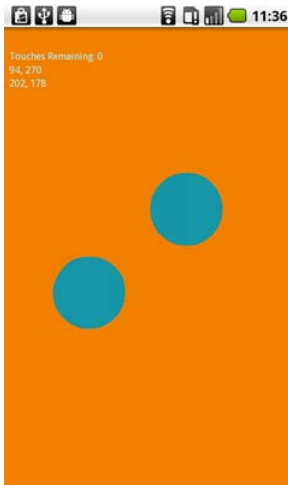


図1: MultitouchExample の実行結果: Nexus One では Multitouch.maxTouchPoints が 2 なので、2つまでのタッチが有効になる。指で画面に触れると青い円が描画され、離すとそれが消える。

## 5. ジェスチャイベントの処理

ジェスチャイベントには GestureEvent、PressAndTapGestureEvent、TransformGestureEvent の3つがあります (PressAndTapGestureEvent と TransformGestureEvent は GestureEvent を継承します)。以下は各ジェスチャイベントでサポートされるイベントのタイプです。

GestureEvent. GESTURE_TWO_FINGER_TAP	2本の指によるタップで定義されるジェスチャを示します。
PressAndTapGestureEvent. GESTURE_PRESS_AND_TAP	画面にまず1本の指で触れた後、別の指でタップすることで定義されるジェスチャを示します。これはコンテキストメニューの呼び出しに使用される Windows の方法です。
TransformGestureEvent. GESTURE_PAN	小さな画面では大きすぎて収まらないようなコンテンツをパンするジェスチャを示します。
TransformGestureEvent. GESTURE_ROTATE	コンテンツを回転させるために、2つのタッチポイントを互いに回転させることで定義されるジェスチャを示します。
TransformGestureEvent. GESTURE_SWIPE	リストをスクロールしたりリストから項目を消去するためなどに、タッチポイントを素早く移動させることで定義されるジェスチャを示します。
TransformGestureEvent. GESTURE_ZOOM	コンテンツのズームインやズームアウトを行うため、2つのタッチポイントを遠ざけるか近づけることで定義されるジェスチャを示します。

## ジェスチャイベントのプロパティ

GestureEvent クラスには MouseEvent クラスと同じプロパティが多くありますが、PressAndTapGestureEventとTransformGestureEvent クラスにはジェスチャのタイプ特有のプロパティが追加されています。

PressAndTapGestureEvent クラスには次のプロパティがあります。

- tapLocalX と tapLocalY は、発生したイベントの、インタラクティブオブジェクトを基準にした水平方向と垂直方向の座標を示します。
- tapStageX と tapStageY は、発生したタップタッチの、グローバルな Stage 座標での水平方向と垂直方向の座標を示します。

TransformGestureEvent クラスには次のプロパティがあります。

- offsetX と offsetY は、前のジェスチャイベントが発生してからの、表示オブジェクトの水平方向と垂直方向の移動量を示します。
- scaleX と scaleY は、前のジェスチャイベントが発生してからの、表示オブジェクトの水平方向と垂直方向の拡大縮小率を示します。
- rotation は、前のジェスチャイベントが発生してからの、表示オブジェクトの現在の回転角度を度単位で示します。

## ここまでのまとめ

以下はサンプルの GestureExample.as の全コードです。ここでは TransformGestureEvent.GESTURE\_ZOOM と TransformGestureEvent.GESTURE\_ROTATE イベントに関して登録し、TransformGestureEvent のデータを使ってゾウの画像を操作しています。

## GestureExample.as

```
package
{
    import flash.display.Bitmap;
    import flash.display.Sprite;
    import flash.events.TransformGestureEvent;
    import flash.text.TextField;
    import flash.text.TextFormat;
    import flash.ui.Multitouch;
```

```
import flash.ui.MultitouchInputMode;

[SWF(width=320, height=460, frameRate=24, backgroundColor=0x000000)]
public class GestureExample extends Sprite
{
    [Embed(source="african_elephant.jpg")]
    public var ElephantImage:Class;
    public var scaleDebug:TextField;
    public var rotateDebug:TextField;

    public function GestureExample()
    {
        // Debug
        var tf:TextFormat = new TextFormat();
        tf.color = 0xffffffff;
        tf.font = "Helvetica";
        tf.size = 11;
        this.scaleDebug = new TextField();
        this.scaleDebug.width = 310;
        this.scaleDebug.defaultTextFormat = tf;
        this.scaleDebug.x = 2;
        this.scaleDebug.y = 2;
        this.stage.addChild(this.scaleDebug);
        this.rotateDebug = new TextField();
        this.rotateDebug.width = 310;
        this.rotateDebug.defaultTextFormat = tf;
        this.rotateDebug.x = 2;
        this.rotateDebug.y = 15;
        this.stage.addChild(this.rotateDebug);

        var elephantBitmap:Bitmap = new ElephantImage();
        var elephant:Sprite = new Sprite();

        elephant.addChild(elephantBitmap);

        elephant.x = 160;
```

```

        elephant.y = 230;

        elephantBitmap.x          =          (300          -
(elephantBitmap.bitmapData.width / 2)) * -1;
        elephantBitmap.y          =          (400          -
(elephantBitmap.bitmapData.height / 2)) * -1;

        this.addChild(elephant);

        Multitouch.inputMode = MultitouchInputMode.GESTURE;

        elephant.addEventListener(TransformGestureEvent.GESTURE_ZOO
M, onZoom);

        elephant.addEventListener(TransformGestureEvent.GESTURE_ROT
ATE, onRotate);
    }

    private function onZoom(e:TransformGestureEvent):void
    {
        this.scaleDebug.text = (e.scaleX + ", " + e.scaleY);
        var elephant:Sprite = e.target as Sprite;
        elephant.scaleX *= e.scaleX;
        elephant.scaleY *= e.scaleY;
    }

    private function onRotate(e:TransformGestureEvent):void
    {
        var elephant:Sprite = e.target as Sprite;
        this.rotateDebug.text = String(e.rotation);
        elephant.rotation += e.rotation;
    }
}
}

```



図2: MultitouchExample の実行結果:ゾウの画像がジェスチャで回転し、ズームイン、アウトする。