Archived:Zoom and Rotate Gestures in FlashLite for touch-enabled devices

Archived: This article is **archived** because it is not considered relevant for third-party developers creating commercial solutions today. If you think this article is still relevant, let us know by adding the template {{ReviewForRemovalFromArchive|user=~~~~|*write your reason here*}}.

We do not recommend Flash Lite development on current Nokia devices, and all Flash Lite articles on this wiki have been archived. Flash Lite has been removed from all Nokia Asha and recent Series 40 devices and has limited support on Symbian. Specific information for Nokia Belle is available in Flash Lite on Nokia Browser for Symbian. Specific information for OLD Series 40 and Symbian devices is available in the Flash Lite Developers Library.

Introduction

22 Nov 2009

Flash Lite technology is supported across all Nokia platforms and is widely integrated with our mobile WebKit browser. The S60 5th Edition platform comes up with the exciting touch UI concept. S60 5th Edition with touch UI offers a promising future for the mobile developers. It has opened new doors to developers not only to survive but also to dominate market with touch UI-enabled applications.

Till now, we have been using the basic touch gestures in Flash Lite for Nokia devices. The basic gestures are vertical and horizontal gestures.

The following article will explain how to implement the interesting zoom, rotate and panning gestures on touch enabled devices like Nokia 5800 XpressMusic using Flash Lite.

Core concept

The traditional zoom gestures are normally implemented for multi-touch screens. Multi-touch (e.g. Microsoft Surface) consists of a touch screen (screen, table, wall, etc.) or touchpad, as well as software that recognizes multiple simultaneous touch points, as opposed to the standard touch-screen (e.g. ATM), which recognizes only one touch point. In case of those multi-touch screens, the zoom gesture is assumed if we drag the fingers in opposite directions. If the fingers are dragged toward each other, it is considered as Zoom out gesture and in case the fingers are dragged away from each other, it is considered as Zoom in gesture.

To define the counter zoom gesture in case of single touch screen, we need to assume a fixed reference point. The fixed reference point can be the center of the image. In case the user drags the finger away from the fixed reference point, it will be considered as Zoom In gesture and in case the user drags the finger towards the fixed reference point, it will be considered as Zoom Out gesture. Depending upon the gesture, the image will be resized at the center and aligned accordingly.

In order to check if the gesture is towards/away from the reference point, we need to check, if the line formed by the start and end points coincides with the reference point. As the probability of the line crossing a single point is less, we will assume a fixed reference area that is, circle. We will check if the line formed by the start and end point intersects the reference circle. In case the line intersects the circle, it will be considered as Zoom gesture and rotate gesture otherwise.

Source Code

Create Zoom Movie

Create a base movie clip for holding the image. Add the graphics of the image inside the movie clip. This is the movie clip which is able to detect and react to zoom and rotate gesture.

Make sure that the registration point of the above movie clip is center.

Add the following code on the above created movie clip and name it as ‘mcImage’.

```
onClipEvent (mouseDown) {
 if (_parent.bZoomRotateMode)
 {
  if (this.bRollingOver)
  {
```


(C) Copyright Nokia 2014. All rights reserved.

```
    this.bDragging = true;
    this.nStartX = _parent._xmouse;
    this.nStartY = _parent._ymouse;
    nAngleOnMousePress = (180 * Math.atan2 (_root._ymouse - this._y, _root._xmouse -
this._x)) / Math.PI + 90;
   }
  }
 }
onClipEvent (mouseMove) {
 if (_parent.bZoomRotateMode)
 {
  if (this.bRollingOver && this.bDragging)
  {
   _parent.fnCheckIfToZoom (this.nStartX,this.nStartY,_parent._xmouse,_parent._ymouse);
  }
 }
}
onClipEvent (mouseUp) {
 if (_parent.bZoomRotateMode)
 {
  if (this.bRollingOver)
  {
   this.bDragging = false;
   nInitialWidth = mcLoadedImage._width;
   nInitialHeight = mcLoadedImage._height;
   nInitialWidth = mcLoadedImage._width;
   nInitialHeight = mcLoadedImage._height;
   nInitialAngle = this._rotation;
  }
 }
}
onClipEvent (load) {
 nInitialOnloadX = this._x;
 nInitialOnloadY = this._y;
}
```

Define the reference circle

As per the logic explained above, we need to define the area for the reference circle. In case the user intersects with this circle, it will be considered as zoom gesture. Create a reference circle of suitable width and height inside the above image movie clip and name it as ‘mcRestrict’.

Add the following code on the first layer inside the zoom movie clip for storing the initial width and height of the image.

```
var nInitialWidth:Number = mcLoadedImage._width;
var nInitialHeight:Number = mcLoadedImage._height;
var nInitialWidthForXY:Number = mcLoadedImage._width;
var nInitialHeightForXY:Number = mcLoadedImage._height;
var nAspectRatio:Number = nInitialWidth / nInitialHeight;
var nInitialAngle:Number = this._rotation;
```

Define the Spanning Constraints

As the main image is zoomed, it will increase in width and height and will not accommodate in the actual mobile space, we need to provide the spanning for the image. To define the spanning constraints create a movie clip. When the image will start spanning, it will always span within this movie clip boundary. Name the movie clip as ‘mcSpanningRestrictions’. This movie clip will be on the same timeline as that of zoom image ‘mcImage’.

 (C) Copyright Nokia 2014. All rights reserved.

Define the Movie clip Dragger

Inside the main image movie clip and above the reference circle layer create a button of same width and height as that of the image and name it as 'mcLoadedImage'. This button will be used to add the rollover and roll-out listener for the image and also to define the image dragging in case of spanning.

Add the following code on the button.

```
on (rollOver) {
 this.bRollingOver = true;
}
on (rollOut) {
 this.bRollingOver = false;
}
on (press) {
 if (_parent.bZoomRotateMode == false)
 {
  if (this._width > _parent.mcSpanningRestrictions._width || this._height >
_parent.mcSpanningRestrictions._height)
  {
   this.startDrag (false,(_parent.mcSpanningRestrictions._x -
(_parent.mcSpanningRestrictions._width / 2 - this._width /
2)),(_parent.mcSpanningRestrictions._y - (_parent.mcSpanningRestrictions._height / 2 -
this._height / 2)),(_parent.mcSpanningRestrictions._x +
(_parent.mcSpanningRestrictions._width / 2 - this._width /
2)),(_parent.mcSpanningRestrictions._y + (_parent.mcSpanningRestrictions._height / 2 -
this._height / 2)));
  }
 }
}
on (release, releaseOutside) {
 this.stopDrag ();
}
```

The above code starts the image dragging in case the current mode is 'Spanning'. The dragging is confined with-in the boundary of the above 'mcSpanningRestrictions' movie clip.

Zoom Logic Implementation

The main image movie clip has a Mouse-move listener. This listener checks the current mode, if the current mode is that of Zoom, it calls the main timeline function 'fnCheckIfToZoom'. This is the core function which checks if the gesture is zoom gesture and the amount by which the image should be zoomed. The function is called with the following parameters passed:

- X-Mouse location when the user press mouse on the image to zoom
- Y-Mouse location when the user press mouse on the image to zoom
- Current X-Mouse location
- Current Y-Mouse location

Create the 'Functions' layer on the main timeline and add the following code to the layer.

```
var bZoomRotateMode:Boolean = true;
var nMinimumWidth:Number = 75;
var nMaximumWidth:Number = 450;
function fnFindShortestDistance (point_x:Number, point_y:Number, line_x1:Number,
line_y1:Number, line_x2:Number, line_y2:Number):Number
{
 var A:Number = point_x - line_x1;
```

 (C) Copyright Nokia 2014. All rights reserved.

```
 var B:Number = point_y - line_y1;
 var C:Number = line_x2 - line_x1;
 var D:Number = line_y2 - line_y1;
 var nDiastance = Math.abs ((A * D) - (C * B)) / Math.sqrt ((C * C) + (D * D));
 return nDiastance;
}
function fnFindDistanceInPoints (oPoint_1, oPoint_2):Number
{
 var nLineLength:Number = Math.sqrt (Math.pow ((oPoint_1.x - oPoint_2.x), 2) + Math.pow
((oPoint_1.y - oPoint_2.y), 2));
 return nLineLength;
}
function fnCheckIfToZoom (nStartX, nStartY, nEndX, nEndY)
{
 var nSlopeOfLine:Number = (nEndY - nStartY) / (nEndX - nStartX);
 var nPreviousY:Number = ((nSlopeOfLine) * (nStartX - 800 - nEndX)) + nEndY;
 var nNextY:Number = ((nSlopeOfLine) * (nEndX + 800 - nEndX)) + nEndY;
 var myPoint:Object = {x:mcImage.mcRestrict._x, y:mcImage.mcRestrict._y};
 mcImage.localToGlobal (myPoint);
 var nCirCentX:Number = myPoint.x;
 var nCirCentY:Number = myPoint.y;
 var nShortestDistance:Number = fnFindShortestDistance (nCirCentX, nCirCentY, nStartX,
nStartY, nEndX, nEndY);
 if (nShortestDistance <= (mcImage.mcRestrict._width / 2))
 {
  var oPoint_1:Object = new Object ();
  var oPoint_2:Object = new Object ();
  oPoint_1.x = nStartX;
  oPoint_1.y = nStartY;
  oPoint_2.x = nEndX;
  oPoint_2.y = nEndY;
  var nDisplacement:Number = fnFindDistanceInPoints (oPoint_1, oPoint_2);
  var sZoomDirection:String = fnCheckQuadrant (nStartX, nStartY, nEndX, nEndY);
  if (sZoomDirection == "ZoomIn")
  {
   mcImage.mcLoadedImage._width = mcImage.nInitialWidth + nDisplacement;
   if (mcImage.mcLoadedImage._width > nMaximumWidth)
   {
    mcImage.mcLoadedImage._width = nMaximumWidth;
   }
  }
  else if (sZoomDirection == "ZoomOut")
  {
   mcImage.mcLoadedImage._width = mcImage.nInitialWidth - nDisplacement;
   if (mcImage.mcLoadedImage._width < nMinimumWidth)
   {
    mcImage.mcLoadedImage._width = nMinimumWidth;
   }
  }
  mcImage.mcLoadedImage._height = mcImage.mcLoadedImage._width / mcImage.nAspectRatio;
 }
 else
 {
  fnCheckIfToRotate (nEndX,nEndY);
 }
 fnCheckIfToRotate (nEndX,nEndY);
}
```

 (C) Copyright Nokia 2014. All rights reserved.

```
function fnCheckQuadrant (nStartX:Number, nStartY:Number, nEndX:Number,
nEndY:Number):String
{
 var oGlobalCoordinatesOfRestrict:Object = {x:mcImage.mcRestrict._x,
y:mcImage.mcRestrict._y};
 mcImage.localToGlobal (oGlobalCoordinatesOfRestrict);
 if (nStartX > (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartX < (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) && nStartY <
oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2)
 {
  //This is Quadrant 1
  if (nEndY < nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX >= (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) &&
nStartY <= (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 2
  if (nEndY < nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX > (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) &&
nStartY > (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2) && nStartY <
(oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 3
  if (nEndX > nStartX)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX >= (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) &&
nStartY >= (oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 4
  if (nEndX > nStartX)
  {
   return "ZoomIn";
  }
  else
```

 (C) Copyright Nokia 2014. All rights reserved.

```
  {
   return "ZoomOut";
  }
 }
 else if (nStartX > (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartX < (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) && nStartY >
(oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 5
  if (nEndY > nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX <= (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartY >= (oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 6
  if (nEndY > nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX < (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartY > (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2) && nStartY <
(oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 7
  if (nEndX < nStartX)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX <= (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartY <= (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 8
  if (nEndX < nStartX)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
```

 (C) Copyright Nokia 2014. All rights reserved.

```
 }
 else
 {
  return "ZoomIn";
 }
}
function fnCheckIfToRotate (nXMouse:Number, nYMouse:Number)
{
 var nBaseAngle:Number = (180 * Math.atan2 (nYMouse - mcImage._y, nXMouse - mcImage._x))
/ Math.PI + 90;
 var nRotationChange:Number = nBaseAngle - mcImage.nAngleOnMousePress;
 mcImage._rotation = mcImage.nInitialAngle + nRotationChange;
}
function fnToggleZoomSpan ()
{
 bZoomRotateMode = !bZoomRotateMode;
 if (bZoomRotateMode)
 {
  mcZoomSpan.gotoAndStop (1);
 }
 else
 {
  mcZoomSpan.gotoAndStop (2);
 }
 mcImage._x = mcImage.nInitialOnloadX;
 mcImage._y = mcImage.nInitialOnloadY;
}
```

The function performs the following important tasks for identifying the zooming:

- Check if the user is dragging through the reference circle.
- Check if it is Zoom-out or Zoom-in gesture.
- Calculate the zoom factor and resize the image accordingly.

Check if the user is dragging through the reference circle

We need to figure out if the user is dragging the image so that the line formed goes through the circle or not. If the line drawn by the user intersects the reference circle, it means it is Zoom gesture.

We will calculate the shortest distance between the line and the center of the reference circle, if the shortest distance happens to be less than the radius of the reference circle, it means that the line is intersecting the circle and not otherwise.

As the user may drag a line of short length which does not intersect the circle but there is probability of it intersecting the circle if extended, we will extend the line in both directions before calculating the shortest distance.

We will calculate the slop of the line and extend the line in both directions about 800 pixels away.

Check if it is Zoom-out or Zoom-in gesture

Once we conclude it is Zoom gesture, we need to identify if it is Zoom-out/Zoom-in gesture. The following function takes the starting X and Y coordinates and Ending X and Y coordinates and returns the Zoom direction.

```
function fnCheckQuadrant (nStartX:Number, nStartY:Number, nEndX:Number,
nEndY:Number):String
{
 var oGlobalCoordinatesOfRestrict:Object = {x:mcImage.mcRestrict._x,
y:mcImage.mcRestrict._y};
 mcImage.localToGlobal (oGlobalCoordinatesOfRestrict);
 if (nStartX > (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartX < (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) && nStartY <
```

 (C) Copyright Nokia 2014. All rights reserved.

```
oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2)
 {
  //This is Quadrant 1
  if (nEndY < nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX >= (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) &&
nStartY <= (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 2
  if (nEndY < nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX > (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) &&
nStartY > (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2) && nStartY <
(oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 3
  if (nEndX > nStartX)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX >= (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) &&
nStartY >= (oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 4
  if (nEndX > nStartX)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX > (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartX < (oGlobalCoordinatesOfRestrict.x + mcImage.mcRestrict._width / 2) && nStartY >
(oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 5
```

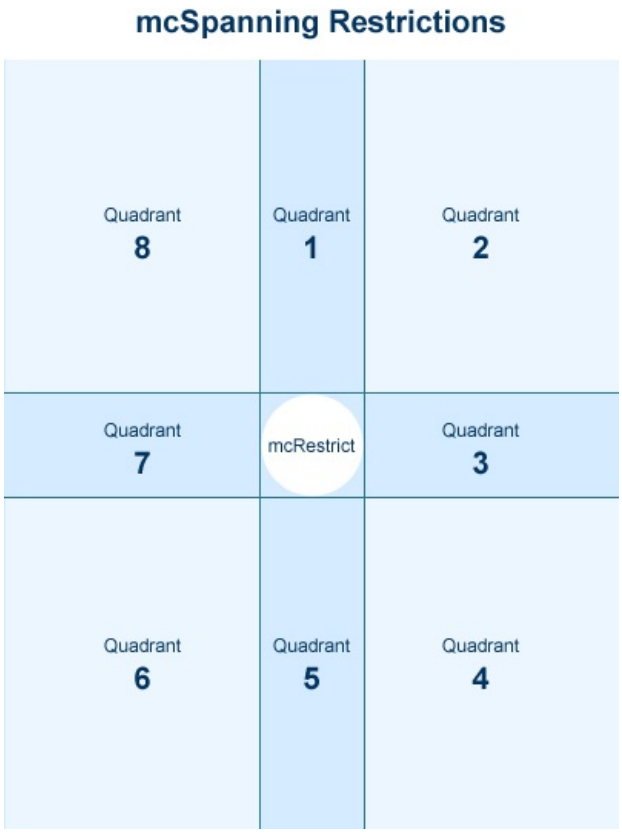
 (C) Copyright Nokia 2014. All rights reserved.

```
  if (nEndY > nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX <= (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartY >= (oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 6
  if (nEndY > nStartY)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX < (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartY > (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2) && nStartY <
(oGlobalCoordinatesOfRestrict.y + mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 7
  if (nEndX < nStartX)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else if (nStartX <= (oGlobalCoordinatesOfRestrict.x - mcImage.mcRestrict._width / 2) &&
nStartY <= (oGlobalCoordinatesOfRestrict.y - mcImage.mcRestrict._height / 2))
 {
  //This is Quadrant 8
  if (nEndX < nStartX)
  {
   return "ZoomIn";
  }
  else
  {
   return "ZoomOut";
  }
 }
 else
 {
  return "ZoomIn";
 }
}
```

We divide the stage in 8 quadrants as shown below and will check in which quadrant the user is zooming. Depending upon the

 (C) Copyright Nokia 2014. All rights reserved.

quadrant, we will decide if this is Zoom-out gesture or Zoom-in gesture.



Calculate the zoom factor and resize the image accordingly

Calculate the distance between the start and end points. This will be nothing but the length of the line formed by these two points. If the gesture is 'Zoom Out', decrease the width of the image by this length and if the gesture is 'Zoom In', increase the width of the image by this length. Now increase the height of the image with respect to the original aspect ratio.

Rotate and Spanning Gesture

Calculating the angle of the rotation is fairly simple and we will apply the change in angle to the image rotation. The following function takes the _xmouse and _ymouse as the parameter and calculates the angle formed at the center of the image. The final angle will be original angle in addition to the change in angle.

```
function fnCheckIfToRotate (nXMouse:Number, nYMouse:Number)
{
 var nBaseAngle:Number = (180 * Math.atan2 (nYMouse - mcImage._y, nXMouse - mcImage._x))
/ Math.PI + 90;
 var nRotationChange:Number = nBaseAngle - mcImage.nAngleOnMousePress;
 mcImage._rotation = mcImage.nInitialAngle + nRotationChange;
}
```

The spanning code is added on the button(mcLoadedImage) inside the Image movieclip(mcImage). This will start the dragging of the Image movieclip subject to the dragging constraint movieclip on the main timeline.

```
on (press) {
 if (_parent.bZoomRotateMode == false)
 {
  if (this._width > _parent.mcSpanningRestrictions._width || this._height >
_parent.mcSpanningRestrictions._height)
  {
   this.startDrag (false,(_parent.mcSpanningRestrictions._x -
(_parent.mcSpanningRestrictions._width / 2 - this._width /
2)),(_parent.mcSpanningRestrictions._y - (_parent.mcSpanningRestrictions._height / 2 -
```

 (C) Copyright Nokia 2014. All rights reserved.

```
this._height / 2)),(_parent.mcSpanningRestrictions._x +
(_parent.mcSpanningRestrictions._width / 2 - this._width /
2)),(_parent.mcSpanningRestrictions._y + (_parent.mcSpanningRestrictions._height / 2 -
this._height / 2)));
  }
 }
}
on (release, releaseOutside) {
 this.stopDrag ();
}
```

Download source code

The following source code is tested on Nokia 5800 XpressMusic device.

The source code is available at Zoom and Rotate Gestures in FlashLite.

 (C) Copyright Nokia 2014. All rights reserved.