Using One-Handed Shortcuts

Tilt and Peek Events

Overview

One-handed shortcuts such as tilt and peek events provide a new way for users to interact with content by enabling them to control UI features through simple movements of the Fire phone.

The purpose of this document is to detail using the MotionGestures APIs for tilt and peek in web apps. Shortcuts that are currently available are listed below, although they may not be available on all devices. There may also be additional shortcuts available in the future.

Note: This feature is specific to the Fire phone.

Tilt

A tilt shortcut can be in one of four directions (left, right, back or forward), and it is stateless (`action=default`) so you will receive one tilt event per complete tilt motion – that is, rocking device to the left and then back to center.

Peek

A peek shortcut can be in one of four directions, just like tilt (left, right, back or forward), but unlike tilt, the peek has state (`action:on` or `action:off`) and will fire each time the state changes (that is, `action:on`, `action:off`);

Continuous peek

Continuous peek is performing an action during the "off and on" state of peeking, where the gesture manager returns a value and an action is performed based on the value of that action.

Note: Peek is far more sensitive and requires less motion than the tilt gesture

Setup

For instructions on setting up your environment please refer to the *"Web Application Developer's Guide"*. If you are a Cordova developer you will need to install the Amazon MotionGestureManager plugin. Please refer to the *"Using Cordova Plugins"* for more details.

API

amazonMotionGestureManager

One-handed shortcut events are available to your web app through a single object called `amazonMotionGestureManger`. Through this object you can get a list of supported event types and register handlers for those events.

The amazonMotionGestureManager object contains a supportedEventTypes element which contains the set of supported events, e.g. [“amazonmotiongesturetilt”, “amazonmotiongesturepeek”]. There is also a supportedGestures element, which contains the set of supported gesture kinds, e.g. [“tilt”, “peek”].

Events

- amazonmotiongesturetilt
- amazonmotiongesturepeek
- amazonmotiongesturecontinuous_peek

Before using the MotionGestures API, check to ensure that the amazonMotionGestureManager object exists, in case you are on a platform that does not support this functionality. Only use the MotionGestures API after handling the amazonPlatformReady event in web applications or the deviceready event from Cordova applications.

The following code snippet checks for the existence of the amazonMotionGestureManager in web applications:

```
document.addEventListener(‘amazonPlatformReady’, function () {
     if (window.amazonMotionGestureManager){
            //run API code here
     }
});
```

The following code snippet checks for the existence of the amazonMotionGestureManager in Cordova applications:

```
document.addEventListener(‘deviceready’, function () {
     if (window.amazonMotionGestureManager){
            //run API code here
     }
});
```

Listeners

Add and remove listeners for gesture events by using the addEventListener() and removeEventListener() method on the amazonMotionGestureManager object for the desired event type.

EXAMPLE :
amazonMotionGestureManager
.addEventListener(“amazonmotiongesturetilt”, handleTiltEvent);

Event Detail Values

The `event` object that is passed back to the handler method contains a `detail` object that exposes the important details of the shortcut event. The table below describes the properties that are available in the `event.detail` object.

Property	Description	Possible values
`Event.detail.action`	The one-handed shortcut state – depends on kind	default, on or off
`Event.detail.direction`	Direction of the shortcut	left, right, back or forward
`Event.detail.kind`	Type of shortcut	tilt or peek
`Event.detail.rotation`	The rotation property refers to the device rotation state, which is a value of 0-3, depending on the Fire phone orientation. The W3C Screen Orientation API allows for a possible 4 orientation values (portrait-primary, landscape-primary, portrait-secondary, landscape-secondary)	0 = portrait-primary 1 = landscape-primary 2 = portrait-secondary 3 = landscape-secondary
`Event.detail.timestamp`	The time (in milliseconds, same base as Date.now()) when the event occurred	Timestamp
`Event.detail.timestamp.nsecs`	The high resolution time (in nanoseconds, same base as System.nanoTime()) when the event occurred	Timestamp
`Event.detail.magnitude`	The float value during continuous peeking to show the angle between the user's head and the screen during a peek event.	Float, or undefined if not a continuous peek event.

Shortcuts Example

In order to use shortcuts in your web app you must add event listeners for tilt, peek and continuous peek events by using the `amazonMotionGestureManager` object as shown below:

```
//First check to make sure the object is available
if (amazonMotionGestureManager) {

    //add tilt listener
    amazonMotionGestureManager
       .addEventListener("amazonmotiongesturetilt", handleTilt);

    //add peek listener
    amazonMotionGestureManager
       .addEventListener("amazonmotiongesturepeek", handlePeek);

}
```

To remove these listeners, simply use the `removeEventListener()` method on the object.

```
//remove tilt listener
amazonMotionGestureManager
    .addEventListener("amazonmotiongesturetilt", handleTilt);

//remove peek listener
amazonMotionGestureManager
    .addEventListener("amazonmotiongesturepeek", handlePeek);
```

Data for the event is sent to the handling method inside the `event.detail` object:

```
//handler method for tilt event
function handleTilt(event) {
    var data = event.detail;
}
```

Sample Code

This document was created in conjunction with a sample application for demonstrating the use of tilt and peek shortcut events. The source code can be found in Amazon WebView API SDK in the *“cordova/samples/MotionGestureManger”* directory.

The sample code below demonstrates the use of the tilt shortcut to show and hide a side panel that is on the left side of the application – both right and left panels are used in the sample source, but for the purposes of this document we will only focus on the left.

NOTE: For more information about side panels, see the *"Implementing Side Panels"* companion document.

- On page load, the panel is placed on the left-hand side of the app – off screen. Figure 1.1
- When the user tilts the device to the right, the panel slides to the right – into view. Figure 1.2
- When the panel is visible and the user tilts the device to the left, the panel slides back to its original position, out of view to the left.

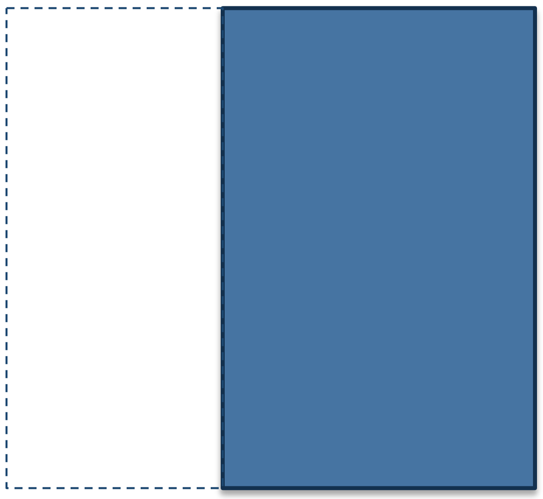
Figure 1.1

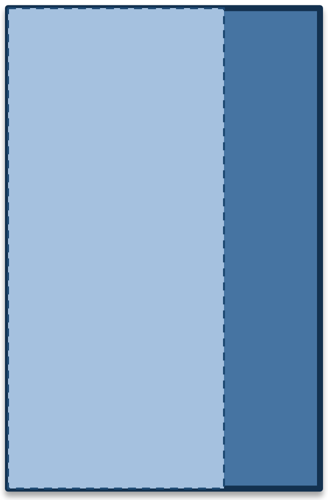
Figure 1.2

For the complete example, see `index.html` and js/`index.js`.

In the snippet of markup below, there is an element with the id "panel_left". On a tilt event, we modify the CSS style for this element to show or hide the panel.

```
<style>
  .box_panelLeft {
      position: absolute;
      top     : 0px;
      width   : 320px;
      height  : 640px;
      z-index : 20;
      left    : -320px;
      transition:all 0.5s ease-in-out;
      -webkit-transition:all 0.5s ease-in-out;
  }

  .act_slideLeft {
    -webkit-transform : translateX(320px);
  }

<body>
    <div id="panel_left" <div>
    <!–Application Body -- >
</body>
```

Now we register a listener for the tilt event.

```
//Add listener for tilt event
amazonMotionGestureManager
   .addEventListener("amazonmotiongesturetilt", handleTilt);
```

```
//flag for current visible panel
var curPanel = undefined;

/**
 * Handler method for the tilt event
 */
function handleTilt(event) {
      var dir = event.detail.direction;

        switch(dir) {
            //dir could be left, right, back or forward, but
            //right now we only care about left and right
            case "left" :
                slidePanel("right");
                break;
            case "right" :
                slidePanel("left");
                break;
            default :
                break;
        }
} //end handleTilt

/**
 * This method makes sure we have the tilt
 * in the direction we want and then toggles the
 * CSS style that will transition the panel position
 */
function slidePanel(dir) {
    if(!curPanel && dir === “right”) { //show panel
        //apply the CSS style to slide
        document.getElementById(“panel_left”)
            .classList.toggle("act_slideLeft”);

        //set our flag
        app.curPanel = dir;

    } else if(curPanel && dir === “left”) { //hide panel

        //apply the CSS style to slide
        document.getElementById(“panel_left”)
            .classList.toggle(“act_slideLeft”);
        //reset flag
        app.curPanel = undefined;
    }
} //end slidePanel
```