Using Head Tracking

Overview

By integrating Amazon WebView (AWV) into their HTML web application, app developers can use Fire phone head tracking functionality directly from within their app's embedded HTML5 engine via custom JavaScript head tracking events.

To access the head tracking stream, register a listener on the window object for the "amazonheadtracking" custom DOM event, similar to using HTML5 Orientation events. Head tracking events give developers granular information about the relation of the user's head to the Fire phone. This information can enhance mobile web apps by enabling custom shortcuts or other unique ways of interacting with the device.

Processing Head Tracking Events

To set up your' app to process head tracking events add the following to your web app's JavaScript code to enable head tracking events:

```
window.addEventListener("amazonheadtracking",
handleAmazonHeadTracking);
```

AWV requests head tracking events at 30 fps, so your event listener (in this case, `handleAmazonHeadTracking()`) will be called MINIMUM of 30fps. If other apps are tracking the head at a higher rate, AWV may get events more frequently. Updated head tracking information that is contained in the event is passed to the listener function.

The AmazonHeadTrackingEvent has the following properties:

AmazonHeadTrackingEvent.headInclinationAngle_deg

The sideways inclination of the head relative to the device, in degrees. If the head is aligned straight with respect to the device in portrait orientation, the inclination angle is 0. It is a positive value if the head rotates clockwise, and negative if it rotates counterclockwise. The range of values is from -180 to 179 degrees, inclusive.

Note: The angle is relative to the physical device in portrait orientation, and independent of the actual screen orientation.

AmazonHeadTrackingEvent.isFaceDetected

True, if a face is detected for this head tracking event. You can use this boolean as a flag to indicate whether the device identified a face.

Note: *The head position may be valid even when no face is detected. This is because the system tracks the head separately from face detection. It is up to the client app whether to rely on the head position when no face is detected.*

AmazonHeadTrackingEvent.isTracking

True, if the service is tracking the head, and therefore the head position is valid. This boolean determines the validity of the head position.

Note*: If this is false, the head position is invalid since the system has stopped tracking the head.*

AmazonHeadTrackingEvent.timestamp

The time (in milliseconds, same base as Date.now()) when the event occurred.

AmazonHeadTrackingEvent.timestamp_nsecs

The system timestamp when the event occurred, in nanoseconds.

AmazonHeadTrackingEvent.x_mm

The X coordinate of the head position relative to the physical center of the device, in millimeters. The physical center of the device is at (0,0,0), and positive X is towards the right of the device.

AmazonHeadTrackingEvent.y_mm

The Y coordinate of the head position relative to the physical center of the device, in millimeters. The physical center of the device is at (0,0,0), and positive Y is towards the top of the device.

AmazonHeadTrackingEvent.z_mm

The Z coordinate of the head position relative to the physical center of the device, in millimeters. The physical center of the device is at (0,0,0) and positive Z is out of the device towards the user.

The value for each of the event properties represents the data derived from the underlying head tracking system. The event provides the absolute position of the head with respect to the physical center of the device.

The center of the device is at point (0,0,0). While looking at the device screen in portrait orientation, positive X is towards the right, positive Y is towards the top, and positive Z is out of the device towards the user.

The head position is a best estimate position as tracked by the service and is independent of whether a face is detected or not.

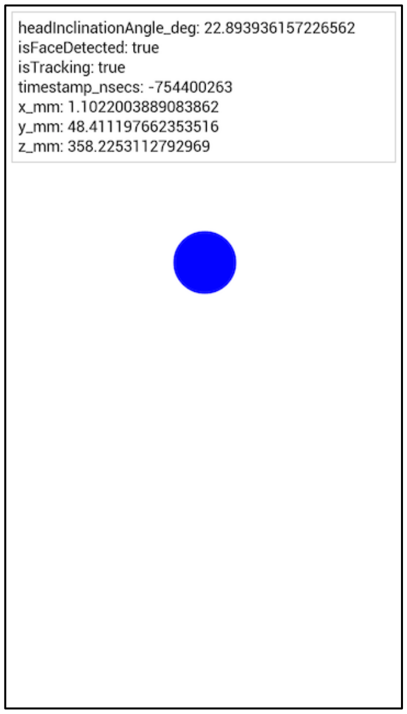
The event also provides the inclination angle of the head along with a boolean indicating whether a face was detected.

To convert the head position into angular values, use the following sample code.

```
// Convert head position to angular values

// Angle around the x-axis
double angleX = Math.atan2(event.y_mm, event.z_mm);

// Angle around the y-axis
double angleY = Math.atan2(event.x_mm, event.z_mm);

// Distance of head from the device origin
double distance = Math.sqrt((event.x_mm * event.x_mm)
                          + (event.y_mm * event.y_mm)
                          + (event.z_mm * event.z_mm));
```

Head Tracking Sample Code

The following sample code uses the head tracking events to draw a round circle on the screen via the DOM.



```
headInclinationAngle_deg: 22.893936157226562
isFaceDetected: true
isTracking: true
timestamp_nsecs: -754400263
x_mm: 1.1022003889083862
y_mm: 48.411197662353516
z_mm: 358.2253112792969
```

Note: This sample is meant to be used within a web app on the Fire.

- The ball follows the user's head as the user looks around the screen, based on the device angle and distance from the user's head.
- The closer the user's face is to the screen, the larger the circle becomes.
- If the user's head tilts beyond a 20 degrees angle, the circle turns blue.
- If the user's face goes outside the system's detection zone, the circle turns black.
- A log of raw event numbers is shown at the top.

See "samples/HeadTracking/index.html" for the complete example.

Manipulate the following DOM elements for this sample:

```
<div id="container">
    <div id="log"></div>
    <div id="ball"></div>
</div>
```

You can register an event listener for the events coming from the head tracking system, the events will fire up to a maximum rate of 30fps. While you have several options to buffer the events, the sample code below simply sets global variables to the values being passed from the head tracking event.

```


window.addEventListener('amazonheadtracking',
handleAmazonHeadTracking);


function handleAmazonHeadTracking(event){

    logText = "";
    logText += "headInclinationAngle_deg: " +
event.headInclinationAngle_deg + "<br>";
    logText += "isFaceDetected: " + event.isFaceDetected +
"<br>";
    logText += "isTracking: " + event.isTracking + "<br>";
    logText += "timestamp_nsecs: " + event.timestamp_nsecs +
"<br>";
    logText += "x_mm: " + event.x_mm + "<br>";
    logText += "y_mm: " + event.y_mm + "<br>";
    logText += "z_mm: " + event.z_mm;

    x_mm = event.x_mm;
    y_mm = event.y_mm;
    z_mm = event.z_mm;

    headAngle = event.headInclinationAngle_deg;

    faceDetected = event.isFaceDetected;

}
```

The following uses those global variables in a requestAnimationFrame() loop, so that you only modify the DOM when it is ready, smoothing out the animation. Estimate the range of values returned from the head tracking event to create a percentage for ball movement.

The code also shows how the z_mm value, which is the measure of distance the user is to the device's screen, can be used to help smooth the numbers being returned from the events into a more natural curve. The closer the user's face is to the device, the shallower the angle needed to move the ball.

```
function animate(){

    var xArc = Math.atan2(x_mm, z_mm) * 100;
    var yArc = Math.atan2(y_mm, z_mm) * 100;

    //Estimated values to get percentages

    var left = w * ((xArc + 50) / 100);
    var top = h - (h * ((yArc + 50) / 100));

    var scale = 4 * (Math.floor(z_mm/5) / 100);

    ball.style.webkitTransform = "scale(" + scale + ")";

    ball.style.top = (top - 10) + "px";
    ball.style.left = (left - 10) + "px";

    if(Math.abs(headAngle) > 10){
        ball.style.backgroundColor = "blue";
    } else {
        ball.style.backgroundColor = "red";
    }

    if(!faceDetected){
        ball.style.backgroundColor = "#000";
    }

    log.innerHTML = logText;

    requestAnimationFrame(animate);

}

animate();
```