Device Orientation ‘alpha’ Calibration

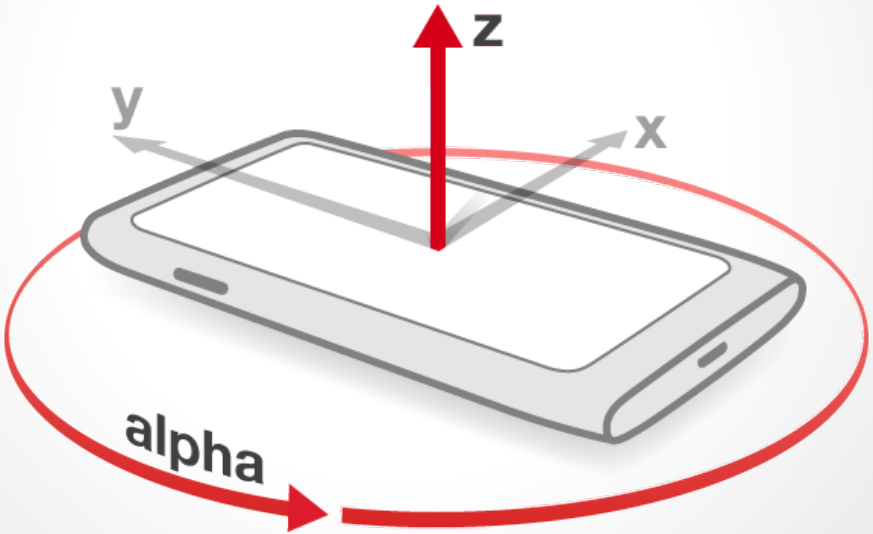
Implementation Status and Challenges

W3C TPAC 2014 - Geolocation Working Group Meeting

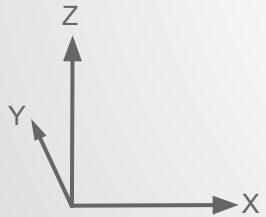
richt at opera dot com

Current Definition (1/2)

`DeviceOrientationEvent.alpha` = The amount of rotation, in a counter-clockwise direction, around a reference **Z** axis denoted by **z**.



World Frame:



`DeviceOrientationEvent.alpha` is of type ‘double’ in the range [0, 360). i.e. 0 <= `event.alpha` < 360.

Current Definition (2/2)

The **`DeviceOrientationEvent.absolute`** property indicates in which direction the corresponding **`alpha`** value ‘points’.

- When `` `event.absolute === true` `` then when `` `event.alpha === 0` `` the device faces due north by the compass.
  - a.k.a. *World*-based Calibration

- When `` `event.absolute === false` `` then when `` `event.alpha === 0` `` the device is facing the direction that the device faced at deviceorientation event initialization.
  - a.k.a. *Game*-based Calibration

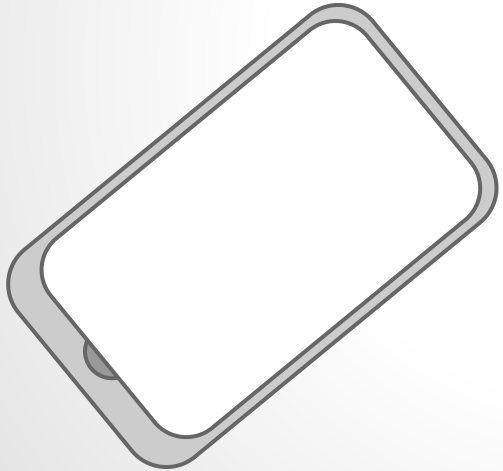
Implementation Status

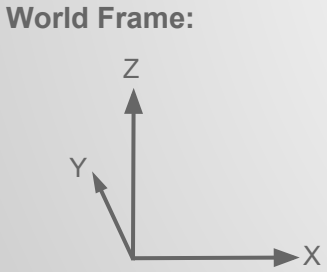
Browser Name	Platform	Is Device Orientation supported?	evt.absolute
Chrome	Android	Yes	true
Opera	Android	Yes	true
Firefox	Android	Yes	true
Safari	iOS	Yes	false
Chrome	iOS	Yes	false
Opera	iOS	Yes	false

- In Android browsers, `DeviceOrientationEvent.alpha` is *world*-based.
- In iOS browsers, `DeviceOrientationEvent.alpha` is *game*-based.

Typical Situation

Initializing Device Orientation Events with the device pointing in an arbitrary direction will give us different `event.alpha` values in different browsers:





Android [*]:

```
event.alpha === 268.342
```

iOS:

```
event.alpha === 0
```

[*] event.alpha === (360 - current *compass* heading)

Game-based calibration on Android

```
var initialOffset = null;

window.addEventListener('deviceorientation', function(evt) {

    if(initialOffset === null) {
        initialOffset = evt.alpha;
    }

    var alpha = evt.alpha - initialOffset;
    if(alpha < 0) {
        alpha += 360;
    }

    // Now use our derived game-based `alpha` instead of raw `evt.alpha` value

}, false);
```

World-based calibration on iOS

```
var initialOffset = null;

window.addEventListener('deviceorientation', function(evt) {

    if(initialOffset === null && evt.absolute !== true
            && +evt.webkitCompassAccuracy > 0 && +evt.webkitCompassAccuracy < 50) {
        initialOffset = evt.webkitCompassHeading || 0;
    }

    var alpha = evt.alpha - initialOffset;
    if(alpha < 0) {
        alpha += 360;
    }

    // Now use our derived world-based `alpha` instead of raw `evt.alpha` value

}, false);
```

The good

It is currently *possible* to derive ‘world’ or ‘game’ based calibration frames for web apps across implementations...

The bad

...but the process of deriving ‘world’ or ‘game’ based calibration frames for web apps is currently *complex*, *non-intuitive* and *non-trivial*.

Additional Issues

- Inconsistencies exist between different implementations.
  - `webkitCompassAccuracy`/`webkitCompassHeading` on iOS
  - `absolute === undefined` on iOS

- Different ‘warm-up periods’ for obtaining compass readings exist on different platforms.

- `alpha` values often ‘drift’ over time.

- Local magnetic interference can affect provided `alpha` values on Android (e.g. from the Google Cardboard Magnetic Button).

Proposal

DeviceOrientation Events should either always provide a game-based reference frame or always provide a world-based reference frame. Not *both* depending on the platform used.

Converging on a Standard (1/3)

Current Android browsers:

- DeviceOrientation = Accelerometer + Gyroscope + *Magnetometer*

Current iOS browsers:

- DeviceOrientation = Accelerometer + Gyroscope

Converging on a Standard (2/3)

Future Android browsers:

- DeviceOrientation = Accelerometer + Gyroscope ~~+ Magnetometer~~

Future iOS browsers:

- DeviceOrientation = Accelerometer + Gyroscope

Converging on a Standard (3/3)

All future browsers:

- DeviceOrientation = Accelerometer + Gyroscope
- i.e. `DeviceOrientationEvent.absolute === false`

Then provide additional tooling to derive world-based orientation frames:

\+ Magnetometer API ?

or

\+ DeviceOrientationEvent.worldAlphaOffset ?

Benefits

- All browsers use the same game-based reference frame for `DeviceOrientationEvent.alpha`.
- Very few current uses of the DeviceOrientation data rely on having a *world*-based reference frame.
  - Removing Magnetometer from DeviceOrientationEvent.alpha derivation provides the least disruptive way to fix current differences without affecting existing web applications.
- If/when web developers need a world-based reference frame for e.g. Virtual Reality use cases they should be able to mix additional magnetometer-derived data in to game-based DeviceOrientation reference frames.

Use Cases for a Magnetometer API

- Virtual Reality in the browser:
  - derived from Magnetometer + Accelerometer-derived Gravity
  - and/or derived from Magnetometer + DeviceOrientation data
    - Enable Sensor Fusion: more accurate world-orientation derivation with improved drift compensation

- Respond to local magnetic field events:
  - Detect Google Cardboard Magnetic Button ‘clicks’ http://youtu.be/DFog2gMnm44?t=18m36s
  - Use magnets to control web apps.
    - e.g. MagiTact http://magitact.com/

Q&A