# Aerospace, Android and Windows 8 Coordinate Systems

# (orientation.c)

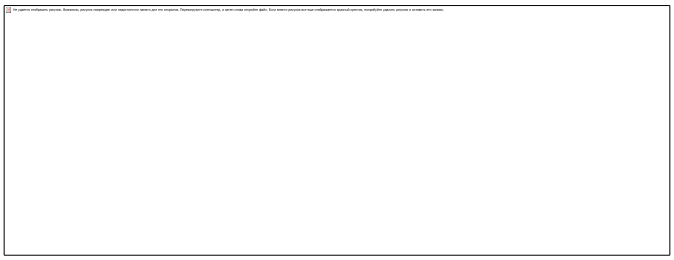
## Technical Note

Document XXX

Version: Draft

Authors: Mark Pedley and Michael Stanley

Date: xx Aug 2014



# Table of Contents

## Glossary

| | |
|---|---|
| $q$ | General quaternion $q = (q_0, q_1, q_2, q_3)$ |
| $\boldsymbol{R}$ | Rotation matrix |
| $\boldsymbol{R}_{NED}$ | Aerospace (NED) rotation matrix |
| $\boldsymbol{R}_{Android}$ | Android rotation matrix |
| $\boldsymbol{R}_{Win8}$ | Windows 8 rotation matrix |
| $\boldsymbol{R}_x$ | Rotation matrix about x axis |
| $\boldsymbol{R}_y$ | Rotation matrix about y axis |
| $\boldsymbol{R}_z$ | Rotation matrix about z axis |
| $\theta$ | Pitch angle |
| $\rho$ | Compass heading angle |
| $\phi$ | Roll angle |
| $\chi$ | Tilt angle |
| $\psi$ | Yaw angle |

Introduction

Summary

This Application Note documents the three coordinate systems used in Freescale's Sensor Fusion software:

- Aerospace (a x=North, y=East, z=Down or NED standard)
- Android (a x=East, y=North, z=Up or ENU standard)
- Windows 8 (another x=East, y=North, z=Up or ENU standard)

If developing for Android or Windows 8 systems then your choice is made for you. If not, the Aerospace coordinate system is probably the best choice since its sequence of rotations and ranges of Euler angles lead to a yaw/compass heading estimate that is more ergonomic than those computed from the other coordinate systems. Specifically, a $180^{o}$ roll while the system is pointed northwards (but inverted) results in an unchanged northwards yaw/compass heading and a $180^{o}$ pitch rotation (so the system is pointing south but inverted) leads to a southwards yaw/compass heading in the Aerospace coordinate system. The reverse occurs in the Android and Windows 8 coordinate systems.

Axes Definitions

Sections 2.1, 3.1 and 4.1 define the axes, the sign standard for the accelerometer and the definition of a positive rotation about any axis for the Aerospace, Android and Windows 8 coordinate systems.

Accelerometers measure linear acceleration minus the gravity component in each axis. This is a consequence of basic Physics which states the equivalence between the forces experienced when at rest in a gravitational field and when accelerating in the opposite direction. The author is writing this document sitting on a chair at rest in a 1g downwards pointing gravitational field but would experience exactly the same forces sitting in a spaceship with a 1g rocket engine providing a 1g acceleration towards the cabin ceiling.

The Aerospace and Windows 8 coordinate systems are gravity positive meaning that an accelerometer axis measures +1g when pointed downwards and aligned with gravity. The Android coordinate system is acceleration positive meaning that an acceleration axis measures +1g when pointed upwards and aligned with the equivalent 1g acceleration in the opposite direction to gravity.

The Aerospace and Windows 8 coordinate systems define rotations in the usual mathematical sense that a rotation is positive when it has a clockwise sense when viewed in the increasing direction of an axis. Rotations in the Android coordinate system have the opposite sign probably as a result of a bug in an early Android specification which became too expensive to fix later.

| **Item** | **Aerospace (NED)** | **Android (ENU)** | **Windows 8 (ENU)** |
|---|---|---|---|
| Axes alignment | XYZ=NED | XYZ=ENU | XY=ENU |
| Accelerometer sign | Gravity-Acceleration | Acceleration-Gravity | Gravity-Acceleration |
| | +1g when axis down | +1g when axis up | +1g when axis down |
| | -1g when axis is up | -1g when axis is down | -1g when axis is up |
| Accelerometer reading when | G[Z]=+1g | G[Z]=+1g | G[Z]=-1g |

flat

| | | | |
|---|---|---|---|
| Direction of positive rotations | Clockwise sense | Anti-clockwise sense | Clockwise sense |
| Compass heading $\rho$ | $\rho=\psi$ | $\rho=\psi$ | $\rho=360^{o}-\psi$ |

Euler Angle Orientation Matrix

Orientation matrices and quaternions are independent of assumptions about the Euler angles (roll, pitch and yaw) and the order in which the Euler angle rotations are applied. Euler angles have poor mathematical properties and are only used in Freescale's Sensor Fusion software as an output for human consumption. Euler angles have many problems of which one is that rotation matrices do not commute and order in which individual Euler angle rotations are applied is important. With different assumptions about the order of individual Euler angle rotations the same orientation matrix will give different roll, pitch and yaw angles.

Since the three standards require the Euler angles to be computed, the table below lists the decomposition of the orientation matrix into three Euler angle rotation matrices in the three standards. As mentioned in section 1.2, pitching the far end of a product upwards in the Aerospace and Windows 8 coordinate systems gives a positive pitch and a negative pitch in the Android standard.

| **Item** | **Aerospace (NED)** | **Android (ENU)** | **Windows 8 (ENU)** |
|---|---|---|---|
| Sequence of Euler angle rotations | $\boldsymbol{R} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$ | $\boldsymbol{R} = \boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi)$ | $\boldsymbol{R} = \boldsymbol{R}_y(\phi)\boldsymbol{R}_x(\theta)\boldsymbol{R}_z(\psi)$ |

Gimbal Lock

Gimbal lock occurs in all three coordinate systems when the second rotation (pitch in Aerospace and Windows 8, roll in Android) aligns the axes of the first and third Euler angle rotations. The number of degrees of freedom reduces from 3 Euler angles to just 2 Euler angles needed to define the orientation. Any change in the first Euler angle can be offset by a cancelling change in the third Euler angle and the two oscillate unstably together.

Gimbal lock in software strapdown navigation systems is therefore an avoidable mathematical instability that results purely from the decomposition of the orientation matrix or quaternion into three individual Euler angle rotations. The orientation matrix or quaternion itself is completely stable at gimbal lock orientations.

Freescale's Sensor Fusion software uses orientation matrix and quaternion algebra throughout and only computes Euler angles for human use as an output from the algorithms. The computed orientation is therefore just as stable at gimbal lock orientations as at other orientations. Gimbal lock is not a problem with Freescale's software. Any software which is based on Euler angle rotations in preference to rotation matrix or quaternion will suffer gimbal lock instability and other discontinuities discussed next.

| **Item** | **Aerospace (NED)** | **Android (ENU)** | **Windows 8 (ENU)** |
|---|---|---|---|
| Gimbal lock | +/- $90^{o}$ pitch (y axis) | +/- $90^{o}$ roll (y axis) | +/- $90^{o}$ pitch (x axis) |

Euler Angle Discontinuities

Another problem with Euler angle representation of orientation is that in addition to their being an infinite number of Euler angle solutions at gimbal lock orientations, there are two Euler angle solutions for all other orientations. This forces one of the Euler angles to be constrained with a reduced range of just -90° to +90° to eliminate one of the two solutions.

The presence of two solutions is easily demonstrated with the starting position of the PCB flat on the table and then applying these two rotation sequences: i) a rotation of 180° in pitch and ii) a rotation of 180° in yaw followed by a rotation of 180° in pitch. Both result in the same orientation. Given that orientation, both Euler angle representations give the same orientation matrix and quaternion.

The Aerospace coordinate system restricts the range of the pitch angle from -90° to +90°. The Android and Windows 9 coordinate systems restrict the range of roll angle from -90° to +90°. Although this restriction results in just one Euler angle solution at all orientations except at gimbal lock, it creates the unfortunate mathematical side-effect of introducing additional discontinuities in the Euler angles at -90° and +90° in the Aerospace coordinate system and at -90° and +90° roll angles in the Android and Windows 8 coordinate systems. Although this occurs at the same orientation as gimbal lock in the Aerospace and Android coordinate systems, it is a different phenomenon as shown by it occurring at a different orientation (-90° and +90° roll) compared to gimbal lock (-90° and +90° pitch) in the Windows 8 coordinate system.

At the further risk of repetition, Euler angles are mathematically troublesome and must be avoided in the inner workings of strapdown sensor fusion software. Their only role is providing output to humans as specified by the Android, Windows 8 or other standard.

The table below summarizes the Euler angle discontinuities in each coordinate system. The meanings of the various entries will become clearer after reading sections 2 through 4.

| **Item** | **Aerospace (NED)** | **Android (ENU)** | **Windows 8 (ENU)** |
|---|---|---|---|
| Ranges of Euler angles | 0° ≤ Yaw $\psi(z)$ < 360° | 0° ≤ Yaw $\psi(z)$ < 360° | 0° ≤ Yaw $\psi(z)$ < 360° |
| | -90° ≤ Pitch $\theta(y)$ < 90° | -90° ≤ Roll $\phi(y)$ < 90° | -180° ≤ Pitch $\theta(x)$ < 180° |
| | -180° ≤ Roll $\phi(x)$ < 180° | -180° ≤ Pitch $\theta(x)$ < 180° | -90° ≤ Roll $\phi(y)$ < 90° |
| Equivalent matrix | $\boldsymbol{R}_x(\phi+\pi)\boldsymbol{R}_y(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$ | $\boldsymbol{R}_x(\theta+\pi)\boldsymbol{R}_y(\pi-\phi)\boldsymbol{R}_z(\psi+\pi)$ | $\boldsymbol{R}_y(\phi-\pi)\boldsymbol{R}_x(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$ |
| Behavior during +/- 180 deg roll rotation | Roll is continuous in range -180° to 180°.<br>No change in yaw or compass angle. | Roll is continuous increasing to 90° and then decreasing or decreasing to -90° and then increasing.<br>Pitch and yaw have 180° discontinuities at +/-90° roll. | 180° jump in roll, pitch, yaw and compass as the roll angle passes 90° and -90°. |
| Behavior during +/- 180 deg pitch rotation | Pitch is continuous increasing to 90° and then decreasing or decreasing to -90° and then increasing. | Smooth changes in pitch. No change in roll, yaw or compass. | Smooth changes in pitch. No change in roll, yaw or compass. |

Roll and yaw have 180$^{o}$
discontinuity at +/-90$^{o}$ pitch.

Functions

void fNEDAnglesDegFromRotationMatrix(float R[][3], float *pfPhiDeg, float *pfTheDeg, float *pfPsiDeg,

float *pfRhoDeg, float *pfChiDeg);

Computes the roll, pitch, yaw, compass heading and tilt angles from an Aerospace/NED orientation matrix.

See section 2.1.

void fAndroidAnglesDegFromRotationMatrix(float R[][3], float *pfPhiDeg, float *pfTheDeg, float *pfPsiDeg,

float *pfRhoDeg, float *pfChiDeg);

Computes the roll, pitch, yaw, compass heading and tilt angles from an Aerospace/NED orientation matrix.

See section 2.2.

void fWin8AnglesDegFromRotationMatrix(float R[][3], float *pfPhiDeg, float *pfTheDeg, float *pfPsiDeg,

float *pfRhoDeg, float *pfChiDeg);

Computes the roll, pitch, yaw, compass heading and tilt angles from an Aerospace/NED orientation matrix.

See section 2.3.

Aerospace (NED) Coordinate System

Axis Definitions

The Aerospace coordinate system is shown in Figure 4-1. It is an x=North, y=East, z=Down (termed NED) coordinate system meaning that when the product is in its default orientation lying flat on the table and pointed northwards, the x axis points north, the y axis points east and the z axis points downwards.

The sign of rotations about the x, y and z axes is defined in the normal mathematical sense that a clockwise rotation is deemed positive when looking along the increasing direction of the axis. Roll, pitch and yaw rotations have their normal meaning and therefore correspond to rotations about the x, y and z axes respectively. The yaw angle $\psi$ equals the compass angle $\rho$.

The Aerospace coordinate system is gravity (not acceleration) positive meaning that the output of any accelerometer channel is positive when pointing downwards and aligned with gravity. The accelerometer *z* axis reading is therefore +1g when the phone is flat and upright.

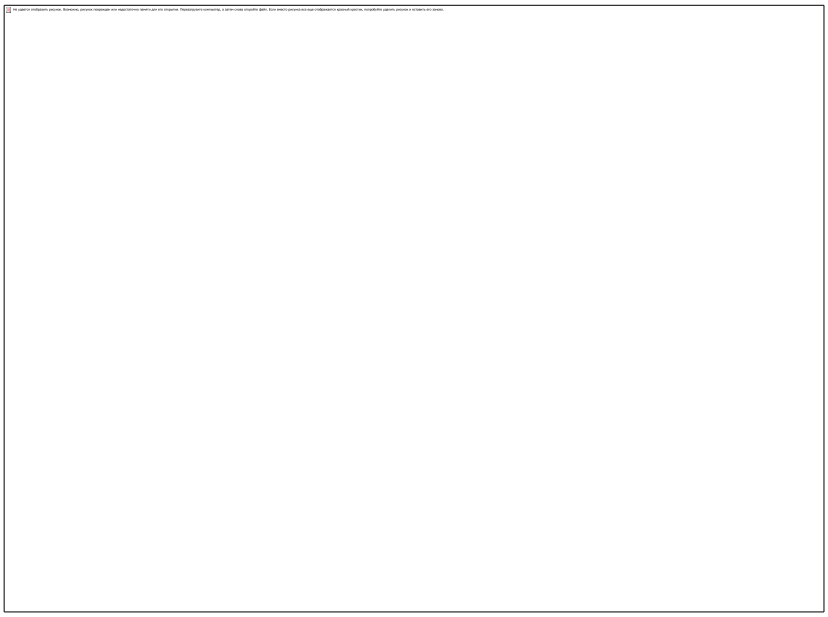


Figure 2-1: The Aerospace (NED) Coordinate System

Orientation Matrix in terms of Euler Angles

The sequence of Euler angle rotations in the Aerospace coordinate system is i) first yaw ii) secondly pitch and iii) finally roll. The roll rotation $\phi$ is about the x axis, the pitch rotation $\theta$ about the y axis and the yaw rotation $\psi$ about the z axis. The individual rotation matrices are:

$$\boldsymbol{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{pmatrix} \quad \text{Eq 2.2.1}$$

$$\boldsymbol{R}_y(\theta) = \begin{pmatrix} cos\,\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{pmatrix} \quad \text{Eq 2.2.2}$$

$$\boldsymbol{R}_z(\psi) = \begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 2.2.3}$$

The composite Aerospace rotation matrix using the specified rotation sequence is then:

$$\boldsymbol{R}_{NED} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\phi & sin\phi \\ 0 & -sin\phi & cos\phi \end{pmatrix}\begin{pmatrix} cos\,\theta & 0 & -sin\theta \\ 0 & 1 & 0 \\ sin\theta & 0 & cos\theta \end{pmatrix}\begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 2.2.4}$$

$$\Rightarrow \boldsymbol{R}_{NED} = \begin{pmatrix} cos\,\theta\,cos\,\psi & cos\,\theta\,sin\psi & -sin\theta \\ cos\,\psi\,sin\theta sin\phi - cos\,\phi\,sin\psi & cos\,\phi\,cos\,\psi + sin\theta sin\phi sin\psi & cos\,\theta\,sin\phi \\ cos\phi\,cos\,\psi\,sin\theta + sin\phi sin\psi & cos\,\phi\,sin\theta sin\psi - cos\,\psi\,sin\phi & cos\,\theta\,cos\phi \end{pmatrix} \quad \text{Eq 2.2.5}$$

Orientation Quaternion in terms of Euler Angles

The individual Euler angle rotation quaternions in the Aerospace coordinate system are:

$$q_x(\phi) = cos\left(\frac{\phi}{2}\right) + \boldsymbol{i}sin\left(\frac{\phi}{2}\right)$$

Eq 2.3.1

$$q_y(\theta) = cos\left(\frac{\theta}{2}\right) + \boldsymbol{j}sin\left(\frac{\theta}{2}\right)$$

Eq 2.3.2

$$q_z(\psi) = cos\left(\frac{\psi}{2}\right) + \boldsymbol{k}sin\left(\frac{\psi}{2}\right)$$

Eq 2.3.3

The Aerospace rotation quaternion $q_{zyx} = q_z(\psi)q_y(\theta)q_x(\phi)$ evaluates to:

$$q_{zyx} = q_z(\psi)q_y(\theta)q_x(\phi) = \left\{cos\left(\frac{\psi}{2}\right) + \boldsymbol{k}sin\left(\frac{\psi}{2}\right)\right\}\left\{cos\left(\frac{\theta}{2}\right) + \boldsymbol{j}sin\left(\frac{\theta}{2}\right)\right\}\left\{cos\left(\frac{\phi}{2}\right) + \boldsymbol{i}sin\left(\frac{\phi}{2}\right)\right\}$$

Eq 2.3.4

$$= \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\} + \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)\right\}\boldsymbol{i}$$
$$+ \left\{cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{j} + \left\{sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{k}$$

Eq 2.3.5

The elements of the Aerospace quaternion $q_{zyx}$ are therefore:

$$q_0 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 2.3.6

$$q_1 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)$$

Eq 2.3.7

$$q_2 = cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 2.3.8

$$q_3 = sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 2.3.9

Gimbal Lock

Gimbal lock occurs in the Aerospace coordinate system when the second pitch rotation aligns the first yaw and third roll rotations. This occurs at 90° pitch upwards and 90° pitch downwards. The number of degrees of freedom in the Euler angle description of the orientation therefore reduces from 3 to 2 and the roll and yaw angle oscillate with only their sum and difference defined.

At gimbal lock $\theta = 90deg$ and $\theta = -90deg$, equation 2.2.5 simplifies to:

$$\boldsymbol{R}_{NED} = \begin{pmatrix} 0 & 0 & -1 \\ -sin(\psi-\phi) & cos(\psi-\phi) & 0 \\ cos(\psi-\phi) & sin(\psi-\phi) & 0 \end{pmatrix} for\ \theta = 90deg \quad \text{Eq 2.4.1}$$

$$\boldsymbol{R}_{NED} = \begin{pmatrix} 0 & 0 & 1 \\ -sin(\psi+\phi) & cos(\psi+\phi) & 0 \\ -cos(\psi+\phi) & -sin(\psi+\phi) & 0 \end{pmatrix} for\ \theta = -90deg \quad \text{Eq 2.4.2}$$

The orientation matrix is stable but the differenced angle $\psi - \phi$ or the summed angle $\psi + \phi$ can be determined at the two gimbal lock orientations.

Euler Angle Discontinuities

The Aerospace rotation matrix $\boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi)$ is equal to the matrix $\boldsymbol{R}_x(\phi+\pi)\boldsymbol{R}_y(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$ where $\pi$ radians corresponds to 180 degrees. By direct evaluation:

$$\boldsymbol{R}_x(\phi+\pi)\boldsymbol{R}_y(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$$

$$= \begin{pmatrix} -cos(\pi-\theta)\,cos\,\psi & -cos(\pi-\theta)\,sin\psi & -sin(\pi-\theta) \\ cos\,\psi\,sin(\pi-\theta)sin\phi - cos\,\phi\,sin\psi & cos\,\phi\,cos\,\psi + sin(\pi-\theta)sin\phi sin\psi & -cos(\pi-\theta)\,sin\phi \\ cos\phi\,cos\,\psi\,sin(\pi-\theta) + sin\phi sin\psi & cos\,\phi\,sin(\pi-\theta)sin\psi - cos\,\psi\,sin\phi & -cos(\pi-\theta)\,cos\phi \end{pmatrix} \quad \text{Eq 2.5.1}$$

$$= \begin{pmatrix} cos\,\theta\,cos\,\psi & cos\,\theta\,sin\psi & -sin\theta \\ cos\,\psi\,sin\theta sin\phi - cos\,\phi\,sin\psi & cos\,\phi\,cos\,\psi + sin\theta sin\phi sin\psi & cos\,\theta\,sin\phi \\ cos\phi\,cos\,\psi\,sin\theta + sin\phi sin\psi & cos\,\phi\,sin\theta sin\psi - cos\,\psi\,sin\phi & cos\,\theta\,cos\phi \end{pmatrix} = \boldsymbol{R}_x(\phi)\boldsymbol{R}_y(\theta)\boldsymbol{R}_z(\psi) \quad \text{Eq 2.5.2}$$

This is the mathematical identity equivalent to the statement made in section 1 that there are two Euler angle solutions to each orientation. In words: "As an alternative to a rotation of $\psi$ in yaw then $\theta$ in pitch and finally by $\phi$ in roll, I can get to the same orientation by rotating 180° plus $\psi$ in yaw, then 180° minus $\theta$ in pitch and finally by 180° plus $\phi$ in roll."

The Aerospace coordinate system removes one of the two solutions by restricting the pitch angle $\theta$ to the range -90° to +90°. Equations 2.5.1 and 2.5.2 then state that whenever the pitch angle is above 90° or less than -90° then the roll angle should increase by 180° (the $\boldsymbol{R}_x(\phi+\pi)$ term), the pitch angle should be negated and 180° added (the $\boldsymbol{R}_y(\pi-\theta)$ term) and the yaw/compass heading should increase by 180° (the $\boldsymbol{R}_z(\psi+\pi)$ term). These discontinuities are shown in Figure 2-2 to 2-4.

Figure 2-2 shows the Euler angles as the PCB is rotated through 360° in roll from the default starting position of being flat and pointed northwards. Neither the pitch nor yaw angles change and the roll discontinuity from -180° to +180° is simply a consequence of modulo 360° arithmetic and is quite acceptable. The average user of a smartphone eCompass will agree that if a smartphone is pointed northwards and then given a roll rotation of 180° then it is still pointed northwards.

Figure 2-3 shows the Euler angles as the PCB is rotated 360° in pitch from the same starting position. The pitch angle reaches 90° and then starts smoothly decreasing without a discontinuity until it reaches -90° and then starts increasing smoothly. The yaw/compass angle flips by 180° at the 90° and -90° pitch angles but this is reasonable ergonomic behavior. For the same reasons discussed above, the average smartphone user will consider a 180° pitch rotation to flip the yaw/compass heading 180°.

Figure 2-4 shows the yaw/compass angle behavior as the PCB is rotated 360° in yaw while remaining flat. The yaw/compass heading increases smoothly with the acceptable modulo 360° discontinuity when pointed northwards.

This relatively benign behavior is the justification for the comment made in Section 1 that the Aerospace coordinate system is the most ergonomically reasonable and should be used in preference unless specifically developing for the Android and Windows 8 standards.

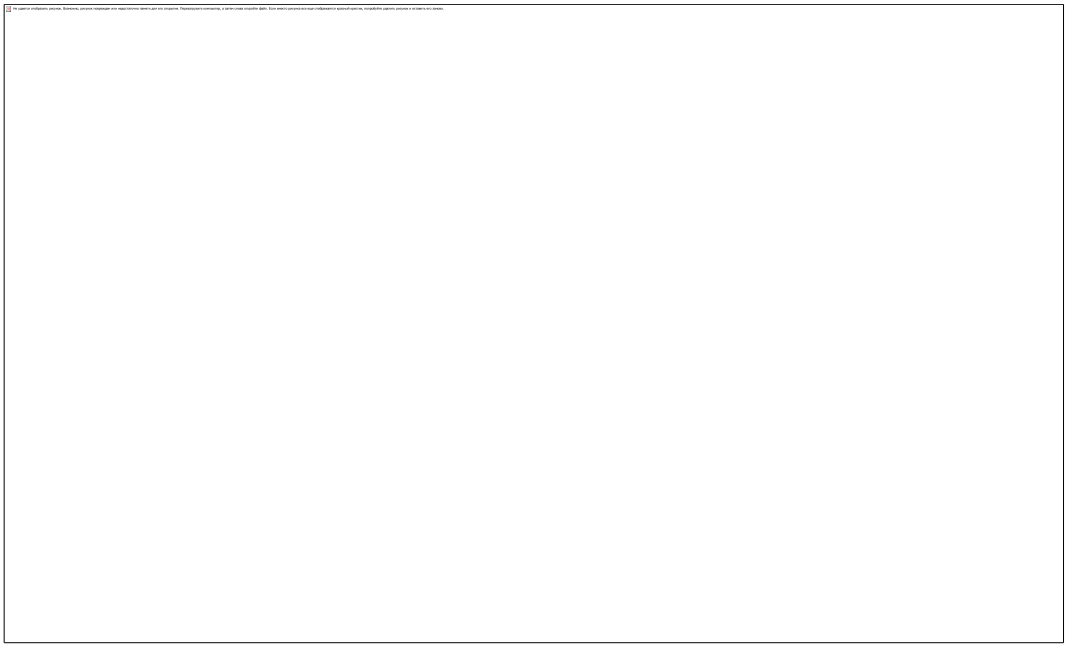


Figure 2-2: 360 degree roll rotation at zero pitch and yaw in the NED coordinate system

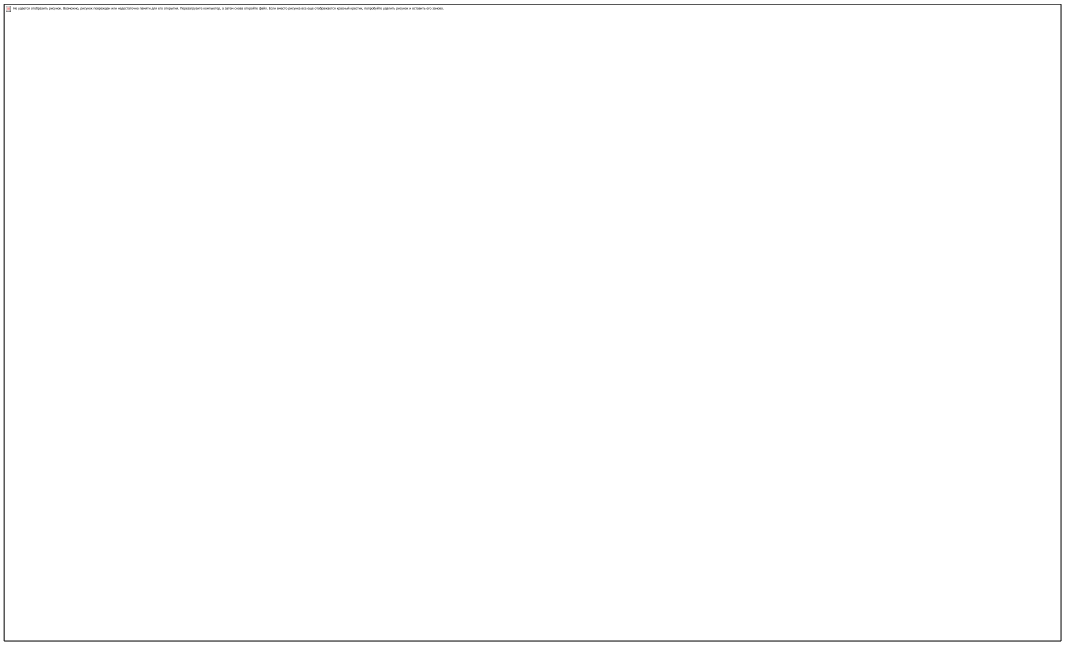


Figure 2-3: 360 degree pitch rotation at zero roll and yaw in the NED coordinate system

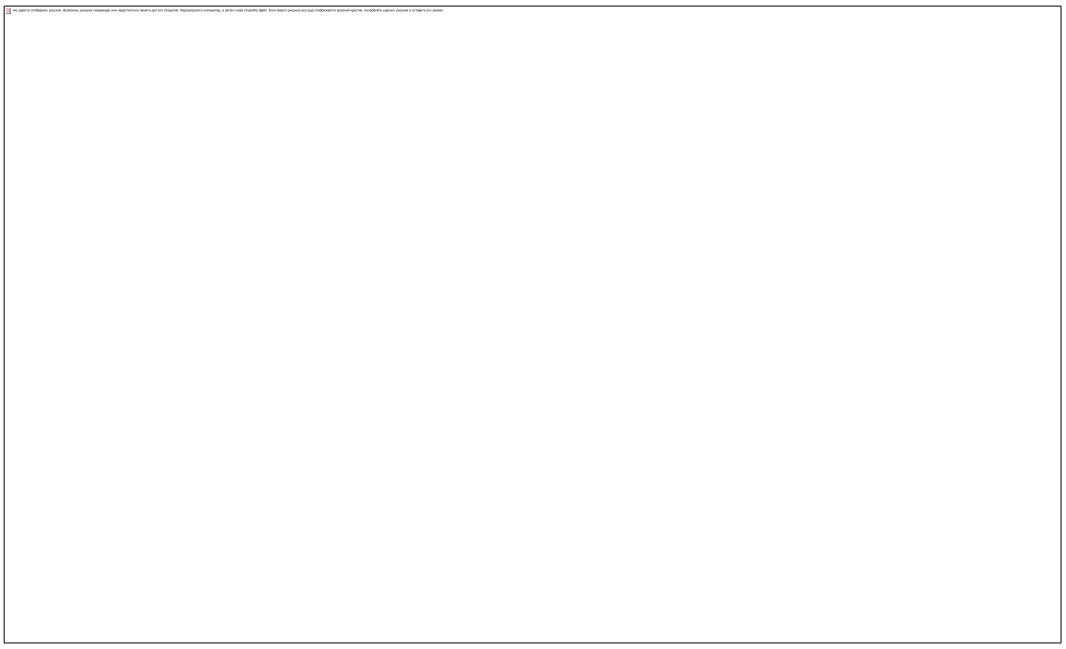


Figure 2-4: 360 degree yaw rotation at zero roll and pitch in the NED coordinate system

Calculation of Euler Angles From Rotation Matrix

This section documents function fNEDAnglesDegFromRotationMatrix which computes the Euler angles from the Aerospace rotation matrix as defined in equation 2.2.5:

The solution for the three Aerospace Euler angles is:

$$\phi = tan^{-1}\left(\frac{R_{yz}}{R_{zz}}\right), -180 \leq \phi < 180\ deg$$

Eq 2.6.1

$$\theta = \sin^{-1}(-R_{xz}), -90 \leq \theta \leq 90\ deg$$ Eq 2.6.2

$$\psi = tan^{-1}\left(\frac{R_{xy}}{R_{xx}}\right), 0 \leq \psi < 360\ deg$$

Eq 2.6.3

The Aerospace compass heading angle $\rho$ always equals the yaw angle $\psi$:

$$\rho = \psi$$ Eq 2.6.4

At gimbal lock, equations 2.4.1 and 2.4.2 give:

$$tan(\psi - \phi) = \left(\frac{R_{zy}}{R_{yy}}\right) for\ \theta = 90deg$$

Eq 2.6.5

$$tan(\psi + \phi) = \left(\frac{-R_{zy}}{R_{yy}}\right) for\ \theta = -90deg$$

Eq 2.6.6

The tilt angle from vertical $\chi$ can be determined from the scalar product of the rotated gravity vector and the downwards z axis giving:

$$cos\chi = \left\{\boldsymbol{R}_{NED}\begin{pmatrix}0\\0\\1\end{pmatrix}\right\} \cdot \begin{pmatrix}0\\0\\1\end{pmatrix} = \begin{pmatrix}R_{xz}\\R_{yz}\\R_{zz}\end{pmatrix} \cdot \begin{pmatrix}0\\0\\1\end{pmatrix} = R_{zz} = cos\,\theta\, cos\phi$$ Eq 2.6.7

Android Coordinate System

Axes

The reference for this section is the Android specification available at:

http://developer.android.com/reference/android/hardware/SensorEvent.html

*Android: "The coordinate space is defined relative to the screen of the phone in its default orientation. The axes are not swapped when the device's screen orientation changes. The OpenGL ES coordinate system is used. The origin is in the lower-left corner with respect to the screen, with the X axis horizontal and pointing right, the Y axis vertical and pointing up and the Z axis pointing outside the front face of the screen. In this system, coordinates behind the screen have negative Z values."*

The Android sensor coordinate system is shown in Figure 5-1. It is an x=East, y=North, z=Up (termed ENU) coordinate system meaning that when the product is in its default orientation lying flat on the table and pointed northwards, the x axis points east, the y axis points north and the z axis points upwards.

The sign of rotations about the x, y and z axes is defined as the opposite to normal mathematical sense. An anti-clockwise rotation is deemed positive when looking along the increasing direction of the axis. Roll, pitch and yaw rotations have their normal meaning and therefore correspond to rotations about the y, x and z axes respectively. The yaw angle $\psi$ equals the compass angle $\rho$.

The Android coordinate system is acceleration (not gravity) positive meaning that the output of any accelerometer channel is positive when pointing upwards and aligned against gravity. The accelerometer *z* axis reading is therefore +1g when the product is flat and upright. This is the same as in the Aerospace coordinate system as a result of two cancelling sign changes: i) the z axes point in opposite directions and ii) Aerospace is gravity positive and Android is acceleration positive.

*Android: "When the device lies flat on a table, the acceleration value is +9.81, which correspond to the acceleration of the device (0 m/s^2) minus the force of gravity"*

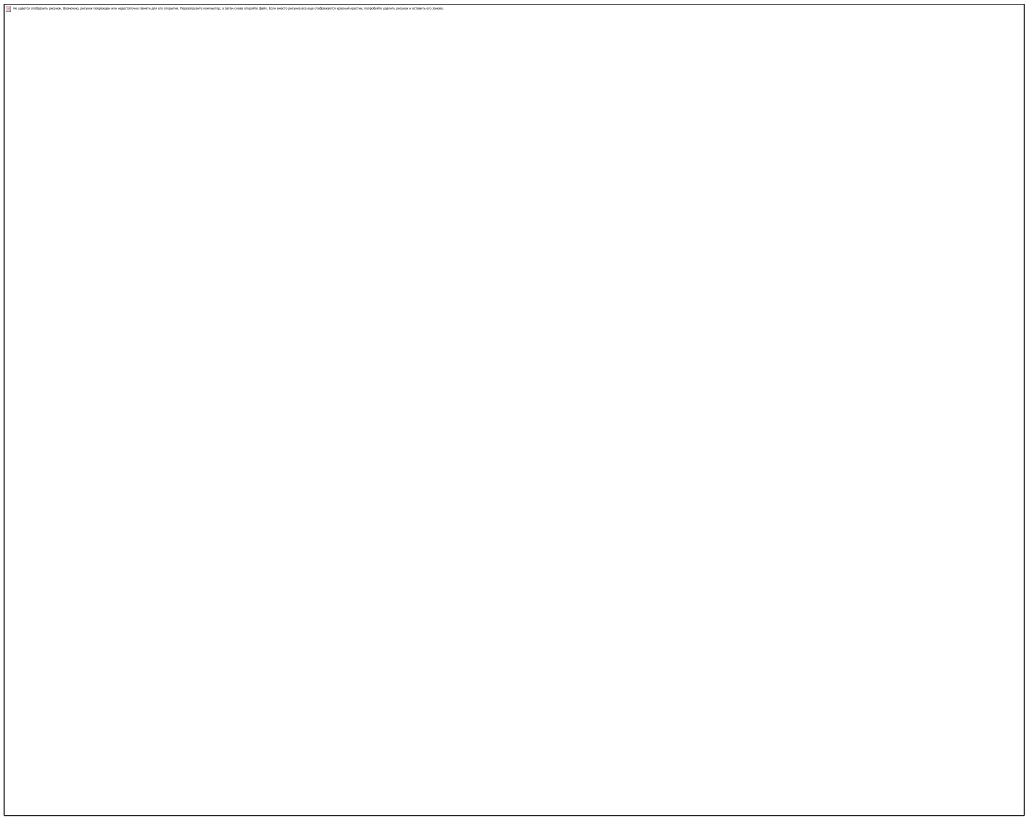


Figure 3-1: The Android Coordinate System

Orientation Matrix in terms of Euler Angles

The sequence of Euler angle rotations in the Android coordinate system does not appear to be documented in the standard. Investigation of the behavior of Android tablets and phones indicates that most use i) first yaw ii) secondly roll and iii) finally pitch. The roll rotation $\phi$ is about the y axis, the pitch rotation $\theta$ about the x axis and the yaw rotation $\psi$ about the z axis. The individual rotation matrices are:

$$\boldsymbol{R}_x(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix} \quad \text{Eq 3.2.1}$$

$$\boldsymbol{R}_y(\phi) = \begin{pmatrix} cos\,\phi & 0 & sin\phi \\ 0 & 1 & 0 \\ -sin\phi & 0 & cos\phi \end{pmatrix} \quad \text{Eq 3.2.2}$$

$$\boldsymbol{R}_z(\psi) = \begin{pmatrix} cos\,\psi & -sin\psi & 0 \\ sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 3.2.3}$$

The composite Android rotation matrix using the specified rotation sequence is:

$$\boldsymbol{R}_{Android} = \boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & -sin\theta \\ 0 & sin\theta & cos\theta \end{pmatrix}\begin{pmatrix} cos\,\phi & 0 & sin\phi \\ 0 & 1 & 0 \\ -sin\phi & 0 & cos\phi \end{pmatrix}\begin{pmatrix} cos\,\psi & -sin\psi & 0 \\ sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 3.2.4}$$

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi & -cos\phi sin\psi & sin\phi \\ cos\,\theta\,sin\psi + cos\,\psi\,sin\phi sin\theta & cos\,\psi\,cos\,\theta - sin\phi sin\psi sin\theta & -cos\,\phi\,sin\theta \\ -cos\,\psi\,cos\,\theta\,sin\phi + sin\psi sin\theta & cos\,\theta\,sin\phi sin\psi + cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} \quad \text{Eq 3.2.5}$$

Orientation Quaternion in terms of Euler Angles

The individual Euler angle rotation quaternions in the Android coordinate system are:

$$q_x(\theta) = cos\left(\frac{\theta}{2}\right) - \boldsymbol{i}sin\left(\frac{\theta}{2}\right)$$

Eq 3.3.1

$$q_y(\phi) = cos\left(\frac{\phi}{2}\right) - \boldsymbol{j}sin\left(\frac{\phi}{2}\right)$$

Eq 3.3.2

$$q_z(\psi) = cos\left(\frac{\psi}{2}\right) - \boldsymbol{k}sin\left(\frac{\psi}{2}\right)$$

Eq 3.3.3

The Android rotation quaternion $q_{zyx} = q_z(\psi)q_y(\phi)q_x(\theta)$ evaluates to:

$$q_{zyx} = q_z(\psi)q_y(\phi)q_x(\theta) = \left\{cos\left(\frac{\psi}{2}\right) - \boldsymbol{k}sin\left(\frac{\psi}{2}\right)\right\}\left\{cos\left(\frac{\phi}{2}\right) - \boldsymbol{j}sin\left(\frac{\phi}{2}\right)\right\}\left\{cos\left(\frac{\theta}{2}\right) - \boldsymbol{i}sin\left(\frac{\theta}{2}\right)\right\}$$

Eq 3.3.4

$$= \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\} - \left\{cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{i}$$
$$- \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)\right\}\boldsymbol{j} - \left\{sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{k}$$

Eq 3.3.5

The elements of the Android quaternion $q_{zyx}$ are therefore:

$$q_0 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 3.3.6

$$q_1 = -cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 3.3.7

$$q_2 = -cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)$$

Eq 3.3.8

$$q_3 = -sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 3.3.9

Gimbal Lock

Gimbal lock occurs in the Android coordinate system when the second roll rotation aligns the first yaw and third pitch rotations. This occurs at +90° roll and -90° roll. The number of degrees of freedom in the Euler angle description of the

orientation therefore reduces from 3 to 2 and the pitch and yaw angles oscillate with only their sum and difference defined.

At gimbal lock $\phi = +90deg$, equation 3.2.5 simplifies to:

$$\boldsymbol{R}_{Android} = \begin{pmatrix} 0 & 0 & 1 \\ sin(\psi+\theta) & cos(\psi+\theta) & 0 \\ -cos(\psi+\theta) & sin(\psi+\theta) & 0 \end{pmatrix} for\ \phi = 90deg \qquad \text{Eq 3.4.1}$$

At gimbal lock $\phi = -90deg$, equation 3.2.5 simplifies to:

$$\boldsymbol{R}_{Android} = \begin{pmatrix} 0 & 0 & -1 \\ sin(\psi-\theta) & cos(\psi-\theta) & 0 \\ cos(\psi-\theta) & -sin(\psi-\theta) & 0 \end{pmatrix} for\ \phi = -90deg \qquad \text{Eq 3.4,2}$$

The orientation matrix is stable but the summed angle $\psi + \theta$ or the differences angle $\psi - \theta$ can be determined at the two gimbal lock orientations.

Euler Angle Discontinuities

The Android rotation matrix $\boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi)$ is equivalent to the matrix $\boldsymbol{R}_x(\theta+\pi)\boldsymbol{R}_y(\pi-\phi)\boldsymbol{R}_z(\psi+\pi)$. By direct evaluation:

$$\boldsymbol{R}_x(\theta+\pi)\boldsymbol{R}_y(\pi-\phi)\boldsymbol{R}_z(\psi+\pi)$$

$$= \begin{pmatrix} -cos(\pi-\phi)\,cos\,\psi & cos(\pi-\phi)sin\psi & sin(\pi-\phi) \\ cos\,\theta\,sin\psi + cos\,\psi\,sin(\pi-\phi)sin\theta & cos\,\psi\,cos\,\theta - sin(\pi-\phi)sin\psi sin\theta & cos(\pi-\phi)\,sin\theta \\ -cos\,\psi\,cos\,\theta\,sin(\pi-\phi) + sin\psi sin\theta & cos\,\theta\,sin(\pi-\phi)sin\psi + cos\,\psi\,sin\theta & -cos(\pi-\phi)\,cos\,\theta \end{pmatrix} \qquad \text{Eq 3.5.1}$$

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi & -cos\phi sin\psi & sin\phi \\ cos\,\theta\,sin\psi + cos\,\psi\,sin\phi sin\theta & cos\,\psi\,cos\,\theta - sin\phi sin\psi sin\theta & -cos\,\phi\,sin\theta \\ -cos\,\psi\,cos\,\theta\,sin\phi + sin\psi sin\theta & cos\,\theta\,sin\phi sin\psi + cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} = \boldsymbol{R}_x(\theta)\boldsymbol{R}_y(\phi)\boldsymbol{R}_z(\psi)$$

Eq 3.5.2

Equation 3.5.1 and 3.5.2 show that there are two solutions for the Android Euler angles for a given rotation matrix. If $\phi, \theta$ and $\psi$ are one solution for the roll, pitch and yaw angles for a given orientation matrix, then so is the solution given by roll angle 180° minus $\phi$, pitch angle 180° plus $\theta$ and yaw angle 180° plus $\psi$.

The Android specification removes of the two solutions by restricting the roll angle $\theta$ to the range -90° to +90°.

***"Sensor.TYPE_ORIENTATION:***

*"All values are angles in degrees.*

*values[0]: Azimuth, angle between the magnetic north direction and the Y axis, around the Z axis (0 to 359). 0=North, 90=East, 180=South, 270=West*

*values[1]: Pitch, rotation around X axis (-180 to 180), with positive values when the z-axis moves toward the y-axis.*

*values[2]: Roll, rotation around Y axis (-90 to 90), with positive values when the x-axis moves toward the z-axis."*

Equations 3.5.1 and 3.5.2 then state that whenever the roll angle is above 90° or less than -90° then the pitch angle should increase by 180° (the $\boldsymbol{R}_x(\theta + \pi)$ term), the roll angle should be negated and 180° added (the $\boldsymbol{R}_y(\pi - \phi)$ term) and the yaw/compass heading should increase by 180° (the $\boldsymbol{R}_z(\psi + \pi)$ term). These discontinuities are shown in Figure 3-2 to 3-4.

Figure 3-2 shows the Euler angles as the PCB is rotated through 360° in roll from the default starting position of being flat and pointed northwards. The roll angle reaches 90° and then starts smoothly decreasing without a discontinuity until it reaches -90° and then starts increasing smoothly. The yaw/compass angle flips by 180° at the 90° and -90° roll angles.

Figure 3-3 shows the Euler angles as the PCB is rotated 360° in pitch from the same starting position. The pitch angle increases smoothly with the acceptable modulo 360° discontinuity between the equivalent angles of -180° and +180°.

Figure 3-4 shows the yaw/compass angle behavior as the PCB is rotated 360° in yaw while remaining flat. The yaw/compass heading increases smoothly with the acceptable modulo 360° discontinuity when pointed northwards.

The 180° compass heading change when the Android roll angle passes through -90° or +90° is arguably not ideal from an ergonomic standpoint. The average user would probably say that given an initial starting point pointing northwards then i) after a 180° roll rotation it is still pointing northwards (Android says it now points southwards) and ii) after a 180° pitch rotation it is pointing southwards (Android says it is still pointing northwards).

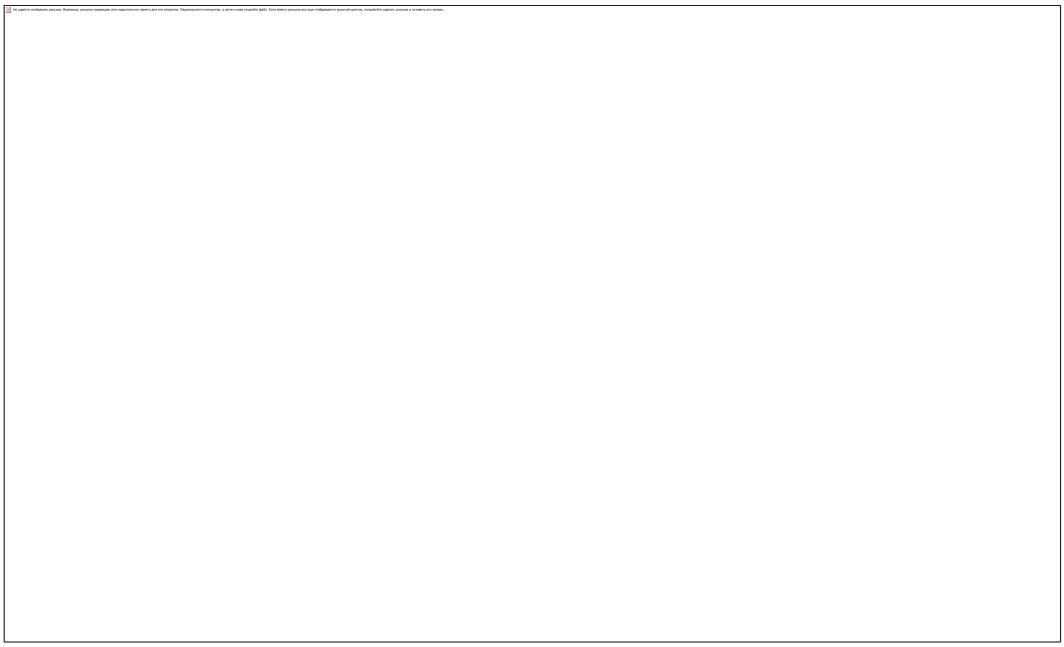


Figure 3-2: 360 degree roll rotation at zero pitch and yaw in the Android coordinate system

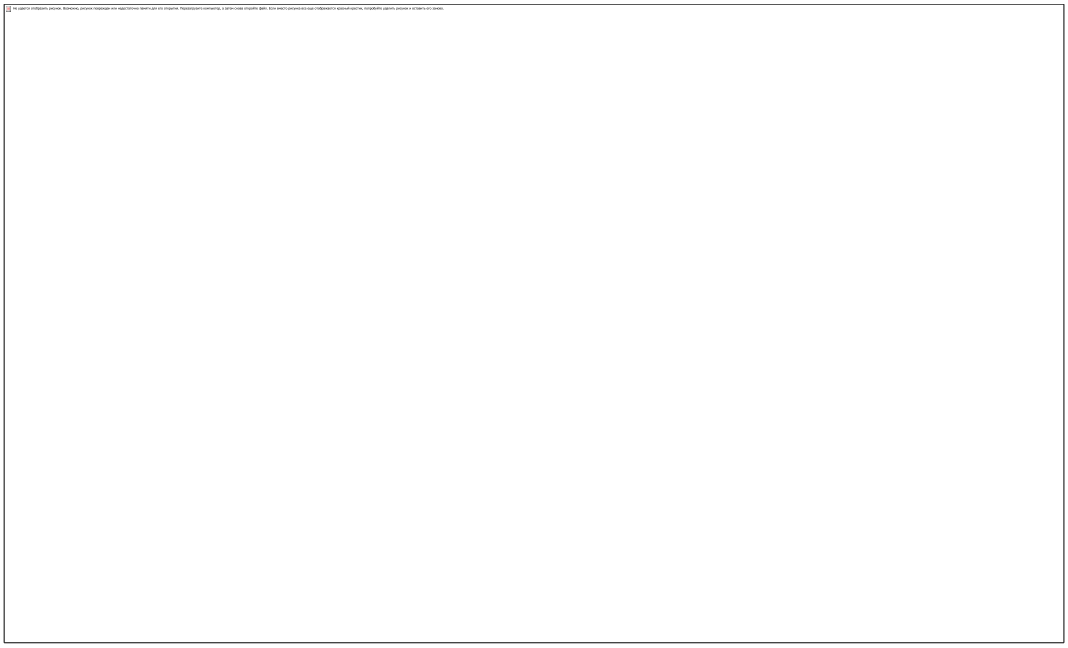


Figure 3-3: 360 degree pitch rotation at zero roll and yaw in the Android coordinate system

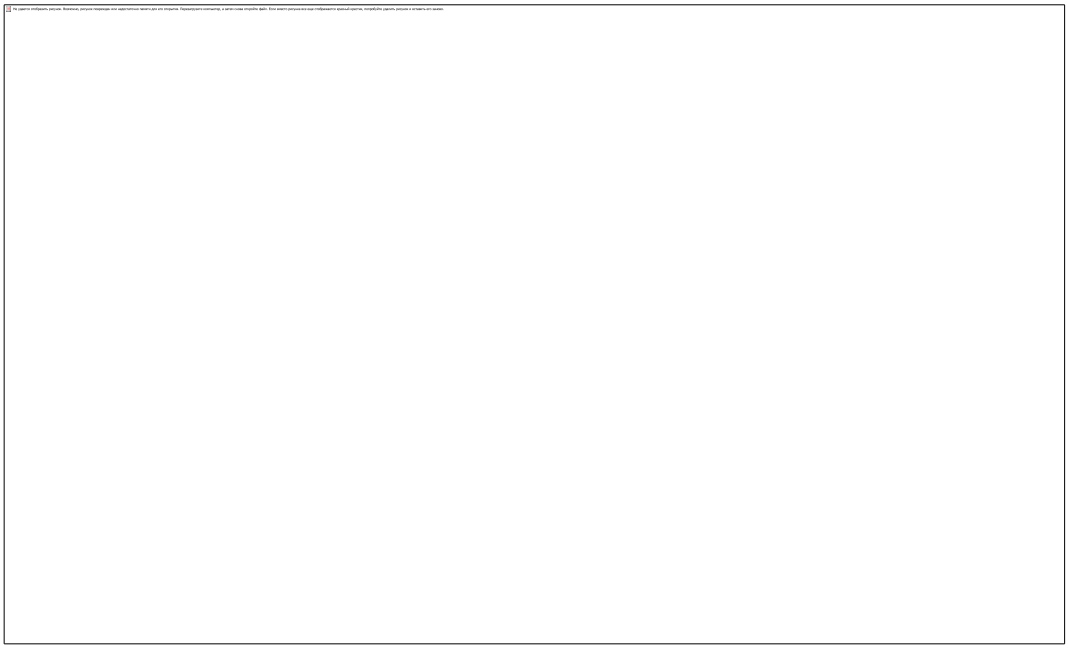


Figure 3-4: 360 degree yaw rotation at zero roll and pitch in the Android coordinate system

Calculation of Euler Angles From Rotation Matrix

This section documents function fAndroidAnglesDegFromRotationMatrix which computes the Euler angles from the Android rotation matrix as defined in equation 3.2.5:

The solution for the three Android Euler angles is:

$\phi = sin^{-1}(R_{xz}), -90 \leq \phi < 90\ deg$ Eq 3.6.1

$$\theta = tan^{-1}\left(\frac{-R_{yz}}{R_{zz}}\right), -180 \leq \theta < 180\ deg$$

Eq 3.6.2

$$\psi = tan^{-1}\left(\frac{-R_{xy}}{R_{xx}}\right), 0 \leq \psi < 360\ deg$$

Eq 3.6.3

The Android compass heading angle $\rho$ always equals the yaw angle $\psi$:

$$\rho = \psi$$

Eq 3.6.4

At gimbal lock equations 3.4.1 and 3.4.2 give:

$$tan(\psi + \theta) = \left(\frac{R_{yx}}{R_{yy}}\right)\ for\ \phi = 90deg$$

Eq 3.6.5

$$tan(\psi - \theta) = \left(\frac{R_{yx}}{R_{yy}}\right)\ for\ \phi = -90deg$$

Eq 3.6.6

The tilt angle from vertical $\chi$ can be determined from the scalar product of the rotated gravity vector and the downwards z axis giving:

$$cos\chi = \left\{\boldsymbol{R}_{Android}\begin{pmatrix}0\\0\\1\end{pmatrix}\right\}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix} = \begin{pmatrix}R_{xz}\\R_{yz}\\R_{zz}\end{pmatrix}\cdot\begin{pmatrix}0\\0\\1\end{pmatrix} = R_{zz} = cos\,\theta\,cos\phi$$

Eq 3.6.7

Windows 8 Coordinate System

Axes

The reference for this section is the Microsoft specification:

"Integrating Motion and Orientation Sensors"

available for download at:

http://msdn.microsoft.com/en-us/library/windows/hardware/dn642102(v=vs.85).aspx

The Windows 8 coordinate system is shown in Figure 4-1. It is an x=East, y=North, z=Up (termed ENU) coordinate system like Android.

The sign of rotations about the x, y, and z axes is opposite to Android and therefore compatible with normal mathematical usage. A consequence is that the Windows 8 compass heading angle $\rho$ has the opposite sense to the Windows 8 yaw angle $\psi$ and $\rho = 360deg - \psi$.

The Windows 8 coordinate system is gravity (not acceleration) positive meaning that the output of any accelerometer channel is negative when pointing upwards and aligned against gravity. The accelerometer z axis reading is therefore -1g when the product is flat and upright.

*Microsoft: "a. Lay the device on a flat surface with the screen pointing up – acceleration values should be the following:*

*X = ~0.0, Y = ~0.0, Z = ~-1.0*

*b. Hold the device with the right hand side pointing upwards up – acceleration values should be the following:*

*X = ~-1.0, Y = ~0.0, Z = ~0.0*

*c. Hold the device with the top of the screen pointing upwards up – acceleration values should be the following:*

*X = ~0.0, Y = ~-1.0, Z = ~0.0"*

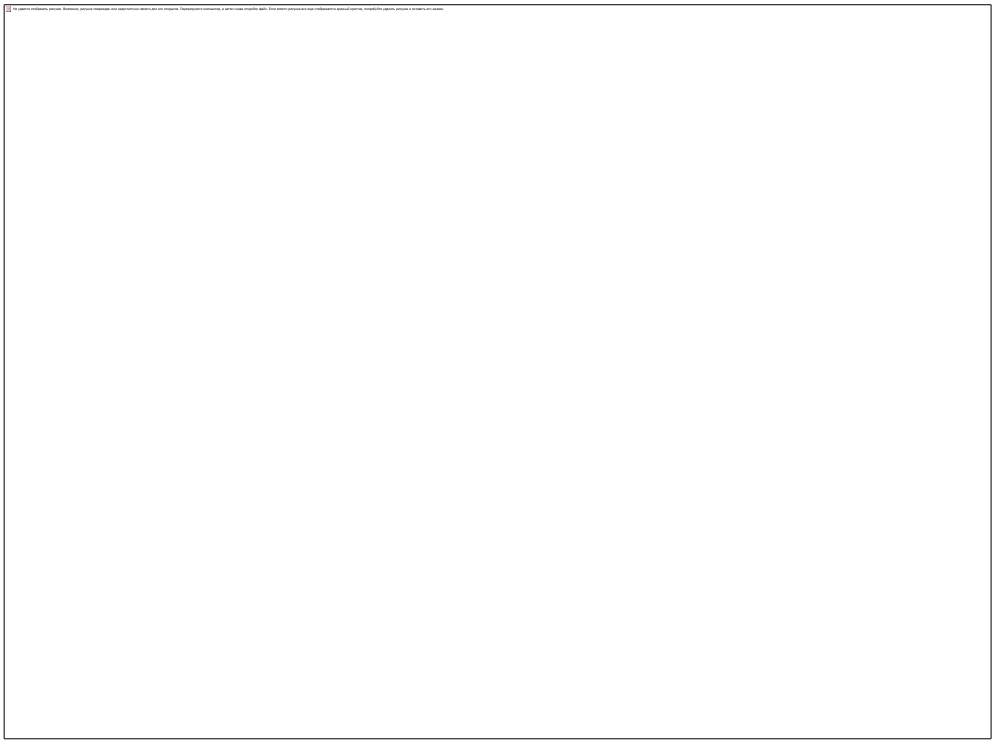


Figure 6-1: Windows 8 Coordinate System

Orientation Matrix in terms of Euler Angles

The order of the rotations in Windows 8 matches the Aerospace standard but not Android. The yaw rotation is first, followed by pitch rotation and then followed by a roll rotation.

*Microsoft: "In order to properly describe the orientation of the device in 3D-space, the Euler angles are applied in the following order, starting with the device lying on a flat surface and the +Y axis pointing due North (towards the North Pole):*

*1. The device is rotated by the Yaw angle about its Z-axis (opposite rotation direction about Z-axis compared to heading).*

*2. The device is rotated by the Pitch angle about its X-axis.*

*3. The device is rotated by the Roll angle about its Y-axis."*

The individual Windows rotation matrices are:

$$\boldsymbol{R_x}(\theta) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{pmatrix} \quad \text{Eq 4.2.1}$$

$$\boldsymbol{R_y}(\phi) = \begin{pmatrix} cos\,\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{pmatrix} \quad \text{Eq 4.2.2}$$

$$\boldsymbol{R_z}(\psi) = \begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 4.2.3}$$

The composite Windows 8 rotation matrix using the Microsoft rotation sequence is:

$$\boldsymbol{R} = \boldsymbol{R_y}(\phi)\boldsymbol{R_x}(\theta)\boldsymbol{R_z}(\psi) = \begin{pmatrix} cos\,\phi & 0 & -sin\phi \\ 0 & 1 & 0 \\ sin\phi & 0 & cos\phi \end{pmatrix}\begin{pmatrix} 1 & 0 & 0 \\ 0 & cos\,\theta & sin\theta \\ 0 & -sin\theta & cos\theta \end{pmatrix}\begin{pmatrix} cos\,\psi & sin\psi & 0 \\ -sin\psi & cos\,\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Eq 4.2.4}$$

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi - sin\phi sin\theta sin\psi & cos\phi sin\psi + cos\,\psi\,sin\phi sin\theta & -sin\phi\,cos\,\theta \\ -cos\,\theta\,sin\psi & cos\,\theta\,cos\,\psi & sin\theta \\ cos\,\psi\,sin\phi + cos\,\phi\,sin\psi sin\theta & sin\phi sin\psi - cos\,\phi\,cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} \quad \text{Eq 4.2.5}$$

Orientation Quaternion in terms of Euler Angles

The individual Euler angle rotation quaternions in the Windows 8 coordinate system are:

$$q_x(\theta) = cos\left(\frac{\theta}{2}\right) + \boldsymbol{i}sin\left(\frac{\theta}{2}\right)$$

Eq 4.3.1

$$q_y(\phi) = cos\left(\frac{\phi}{2}\right) + \boldsymbol{j}sin\left(\frac{\phi}{2}\right)$$

Eq 4.3.2

$$q_z(\psi) = cos\left(\frac{\psi}{2}\right) + \boldsymbol{k} sin\left(\frac{\psi}{2}\right)$$

Eq 4.3.3

The Windows 8 rotation quaternion $q_{zxy} = q_z(\psi)q_x(\theta)q_y(\phi)$ evaluates to:

$$q_{zxy} = q_z(\psi)q_x(\theta)q_y(\phi) = \left\{cos\left(\frac{\psi}{2}\right) + \boldsymbol{k} sin\left(\frac{\psi}{2}\right)\right\}\left\{cos\left(\frac{\theta}{2}\right) + \boldsymbol{i} sin\left(\frac{\theta}{2}\right)\right\}\left\{cos\left(\frac{\phi}{2}\right) + \boldsymbol{j} sin\left(\frac{\phi}{2}\right)\right\}$$

Eq 4.3.4

$$= \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\} + \left\{cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{i}$$
$$+ \left\{cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)\right\}\boldsymbol{j} + \left\{sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)\right\}\boldsymbol{k}$$

Eq 4.3.5

The elements of the Windows 8 quaternion $q_{zxy}$ are therefore:

$$q_0 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 4.3.6

$$q_1 = cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) - sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 4.3.7

$$q_2 = cos\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right) + sin\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right)$$

Eq 4.3.8

$$q_3 = sin\left(\frac{\psi}{2}\right)cos\left(\frac{\theta}{2}\right)cos\left(\frac{\phi}{2}\right) + cos\left(\frac{\psi}{2}\right)sin\left(\frac{\theta}{2}\right)sin\left(\frac{\phi}{2}\right)$$

Eq 4.3.9

Gimbal Lock

Gimbal lock occurs in the Windows 8 coordinate system when the second pitch rotation aligns the first yaw and third roll rotations. This occurs at +90° pitch and -90° pitch angles. The number of degrees of freedom in the Euler angle description of the orientation therefore reduces from 3 to 2 and the roll and yaw angles oscillate with only their sum and difference defined.

At gimbal lock $\theta = 90deg$, equation 4.2.5 simplifies to:

$$\boldsymbol{R}_{Win8} = \begin{pmatrix} cos(\psi+\phi) & sin(\psi+\phi) & 0 \\ 0 & 0 & 1 \\ sin(\psi+\phi) & -cos(\psi+\phi) & 0 \end{pmatrix} for\ \theta = 90deg$$

Eq 4.4.1

At gimbal lock $\theta = -90deg$, equation 4.2.5 simplifies to:

$$\boldsymbol{R}_{Win8} = \begin{pmatrix} cos(\psi-\phi) & sin(\psi-\phi) & 0 \\ 0 & 0 & -1 \\ -sin(\psi-\phi) & cos(\psi-\phi) & 0 \end{pmatrix} for\ \theta = -90deg$$ Eq 4.4.2

The orientation matrix is stable but only the summed angle $\psi + \phi$ or the differenced angle $\psi - \phi$ can be determined at the two gimbal lock orientations.

Euler Angle Discontinuities

The Windows 8 rotation matrix $\boldsymbol{R}_y(\phi)\boldsymbol{R}_x(\theta)\boldsymbol{R}_z(\psi)$ is equal to the matrix $\boldsymbol{R}_y(\phi+\pi)\boldsymbol{R}_x(\pi-\theta)\boldsymbol{R}_z(\psi+\pi)$. By direct evaluation:

$$\boldsymbol{R}_y(\phi+\pi)\boldsymbol{R}_x(\pi-\theta)\boldsymbol{R}_z(\psi+\pi) = \begin{pmatrix} cos\,\phi\,cos\,\psi - sin\phi sin\theta sin\psi & cos\phi sin\psi + cos\,\psi\,sin\phi sin\theta & -cos\,\theta\,sin\phi \\ -cos\,\theta\,sin\psi & cos\,\psi\,cos\,\theta & sin\theta \\ cos\,\psi\,sin\phi + cos\,\phi\,sin\psi sin\theta & sin\phi sin\psi - cos\,\phi\,cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix}$$ Eq 4.5.1

$$= \begin{pmatrix} cos\,\phi\,cos\,\psi - sin\phi sin\theta sin\psi & cos\phi sin\psi + cos\,\psi\,sin\phi sin\theta & -cos\,\theta\,sin\phi \\ -cos\,\theta\,sin\psi & cos\,\psi\,cos\,\theta & sin\theta \\ cos\,\psi\,sin\phi + cos\,\phi\,sin\psi sin\theta & sin\phi sin\psi - cos\,\phi\,cos\,\psi\,sin\theta & cos\,\phi\,cos\,\theta \end{pmatrix} = \boldsymbol{R}_y(\phi)\boldsymbol{R}_x(\theta)\boldsymbol{R}_z(\psi)$$ Eq 4.5.2

Equation 4.5.1 and 4.5.2 show that there are two solutions for the Windows 8 Euler angles for a given rotation matrix. If $\phi, \theta$ and $\psi$ are one solution for the roll, pitch and yaw angles for a given orientation matrix, then so is the solution given by roll angle 180° plus $\phi$, pitch angle 180° minus $\theta$ and yaw angle 180° plus $\psi$.

The Windows 8 specification removes of the two solutions by restricting the roll angle $\theta$ to the range -90° to +90°.

*Microsoft: "The following ranges are used for the representation of Yaw, Pitch, and Roll:*

- *0.0° ≤ Yaw < 360.0°*
- *-180.0° ≤ Pitch < 180.0°*
- *-90.0° ≤ Roll < 90.0°"*

Equations 4.5.1 and 4.5.2 then state that whenever the roll angle is above 90° or less than -90° then the roll angle should increase by 180° (the $\boldsymbol{R}_y(\phi+\pi)$ term), the pitch angle should be negated and 180° added (the $\boldsymbol{R}_x(\pi-\theta)$ term) and the yaw/compass heading should increase by 180° (the $\boldsymbol{R}_z(\psi+\pi)$ term). These discontinuities are shown in Figure 4-2 to 4-4 and should be compared with Figure 21 in the Microsoft specification "Integrating Motion and Orientation Sensors".

Figure 3-2 shows the Euler angles as the PCB is rotated through 360° in roll from the default starting position of being flat and pointed northwards. The roll angle reaches 90°, has a discontinuity to -90°, increases to 90° and then has another discontinuity to -90°. The pitch and yaw/compass heading angles suffer 180° discontinuities at 90° and -90° roll angles.

Figure 3-3 shows the Euler angles as the PCB is rotated 360° in pitch from the same starting position. The pitch angle increases smoothly with the acceptable modulo 360° discontinuity between the equivalent angles of -180° and +180°.

Figure 3-4 shows the yaw/compass angle behavior as the PCB is rotated 360° in yaw while remaining flat. The yaw/compass heading increases smoothly with the acceptable modulo 360° discontinuity when pointed northwards.

The 180° compass heading change when the Windows 8 roll angle passes through -90° or +90° is, like Android, arguably not ideal from an ergonomic standpoint. But this behavior is forced mathematically onto the Microsoft specification by the unfortunate properties of Euler angles and the requirement to limit the roll angle range from -90° to +90°.

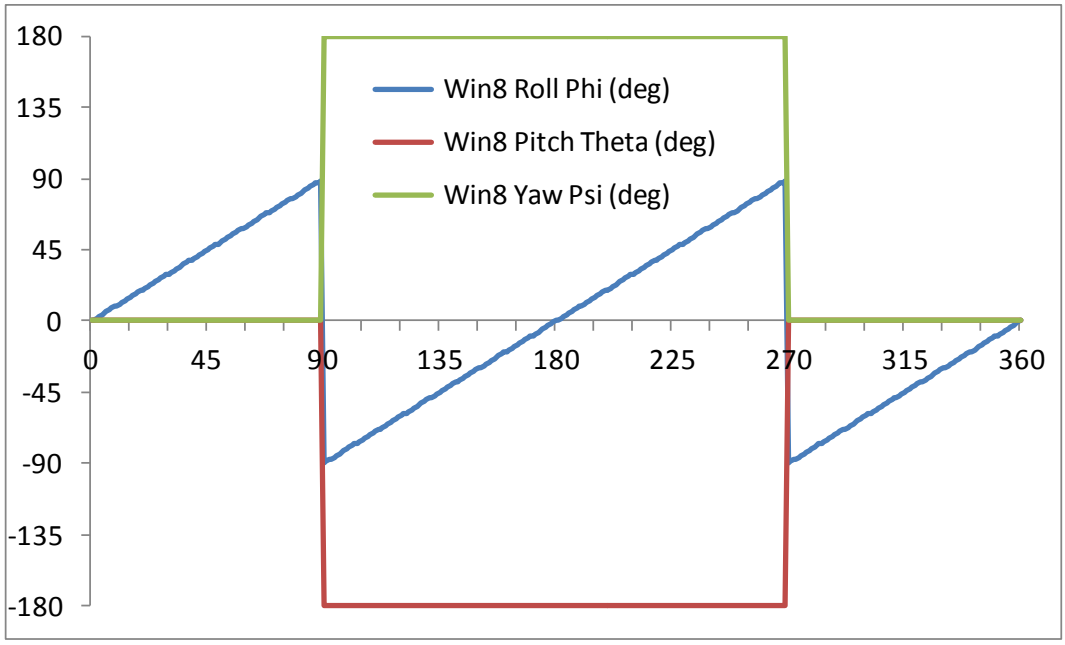



Figure 4-2: 360 degree roll rotation at zero pitch and yaw in the Windows 8 coordinate system

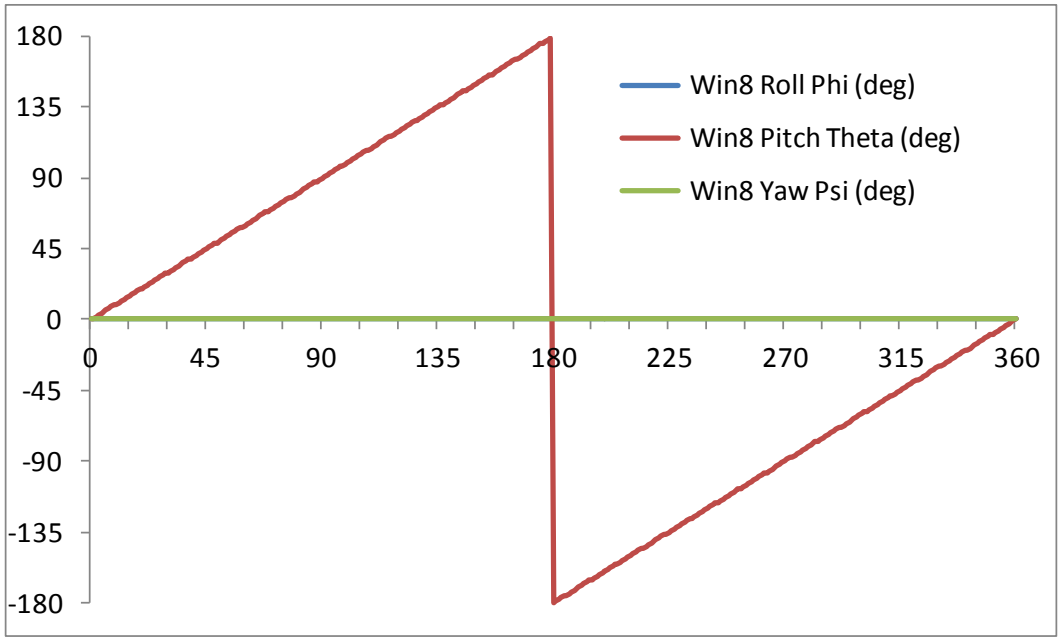


Figure 4-3: 360 degree pitch rotation at zero roll and yaw in the Windows 8 coordinate system

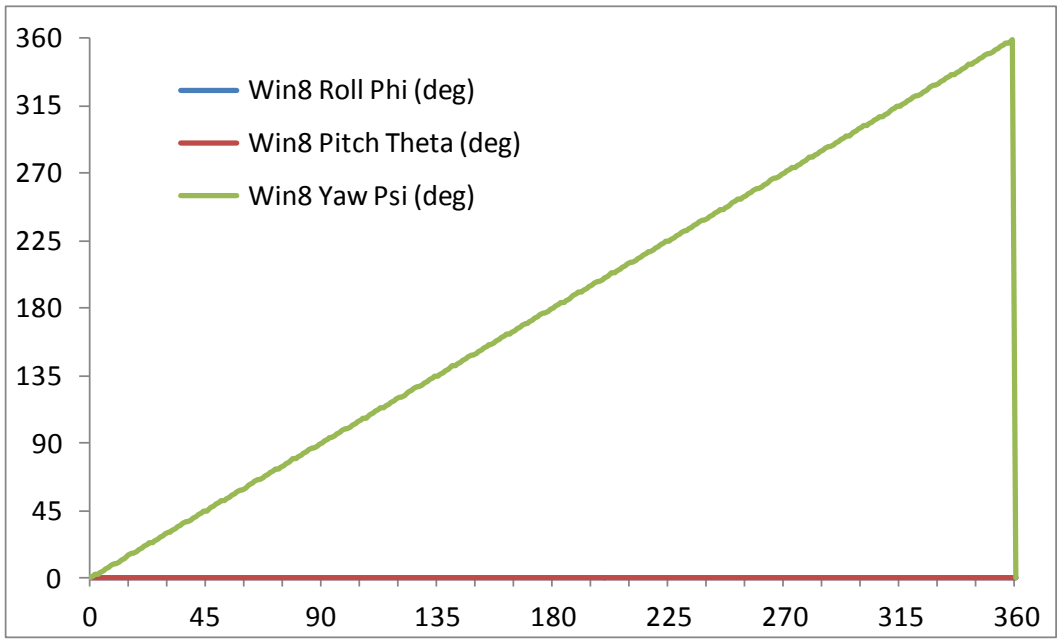



Figure 4-4: 360 degree yaw rotation at zero roll and pitch in the Windows 8 coordinate system

Calculation of Euler Angles From Rotation Matrix

This section documents function fWin8AnglesDegFromRotationMatrix which computes the Euler angles from the Windows 8 rotation matrix as defined in equation 4.2.5:

The solution for the roll angle $\phi$ is:

$$\phi = tan^{-1}\left(\frac{-R_{xz}}{R_{zz}}\right), -90 \leq \phi < 90\ deg$$

Eq 4.6.1

The pitch angle $\theta$ has range $-180 \leq \theta < 180\ deg$ but is first computed in the range -90° to 90° using:

$$\theta = sin^{-1}\left(R_{yz}\right), -90 \leq \theta < 90\ deg$$

Eq 4.6.2

Since $\phi$ is in the range -90° to 90°, it follows that $cos\phi$ is non-negative and that $R_{zz} = cos\ \phi\ cos\ \theta$ has the same sign as $cos\ \theta$ and can be used to correct $\theta$ into the range $-180 \leq \theta < 180\ deg$:

$$\theta \leftarrow \pi - \theta\ if\ R_{zz} < 0\ and\ \theta > 0$$

Eq 4.6.3

$$\theta \leftarrow -\pi - \theta\ if\ R_{zz} < 0\ and\ \theta \leq 0$$

Eq 4.6.4

The general solution for the yaw angle $\psi$ for non-zero $cos\ \theta$ is given by:

$$\psi = tan^{-1}\left(\frac{-\cos\theta\, R_{yx}}{\cos\theta\, R_{yy}}\right), 0 \le \psi < 360\, deg\,, \theta \neq -90deg, \theta \neq 90deg$$

Eq 4.6.5

At gimbal lock, equations 4.4.1 and 4.4.2 give:

$$tan(\psi + \phi) = \left(\frac{R_{xy}}{R_{xx}}\right)\ for\ \theta = 90deg$$

Eq 4.6.6

$$tan(\psi - \phi) = \left(\frac{R_{xy}}{R_{xx}}\right)\ for\ \theta = -90deg$$

Eq 4.6.7

The Windows 8 compass heading angle $\rho$ is the negative (modulo 360°) of the yaw angle $\psi$:

$$\rho = -\psi$$ Eq 4.6.8

The tilt angle from vertical $\chi$ can be determined from the scalar product of the rotated gravity vector and the downwards z axis giving:

$$cos\chi = \left\{\boldsymbol{R}_{Win8}\begin{pmatrix}0\\0\\-1\end{pmatrix}\right\}\cdot\begin{pmatrix}0\\0\\-1\end{pmatrix} = -\begin{pmatrix}R_{xz}\\R_{yz}\\R_{zz}\end{pmatrix}\cdot\begin{pmatrix}0\\0\\-1\end{pmatrix} = R_{zz} = cos\,\theta\, cos\phi$$ Eq 4.6.9

Direction of Windows 8 Rotation Matrix

The Freescale sensor fusion software defines the orientation matrix as transforming a vector from the global to the sensor coordinate frame for all coordinate systems. The rotation matrices and quaternions listed on page 82 of the Microsoft specification are the transpose and conjugate of the Freescale standard and therefore refer to transformation from the sensor frame to the global frame. To convert from one to the other, simply take the conjugate of the quaternion or transpose the matrix.