ORIENTATION

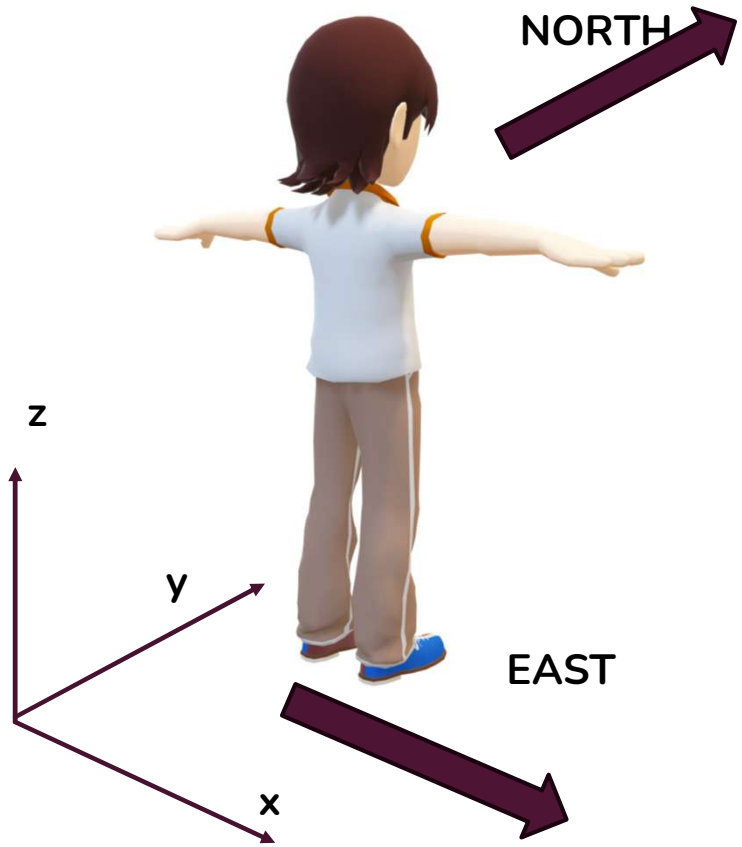
ORIENTATION (INFORMAL MEANING)

- **Orientation** is the ability to identify **reference directions** in space.
- In everyday life, orientation means answering questions such as:
- Where is **North**?
- Which direction am I facing?
- What is in front of me, behind me, on my left and right?
- Traditionally, orientation is defined using the **cardinal directions**:
  North, South, East, West.

ORIENTATION (INFORMAL MEANING)

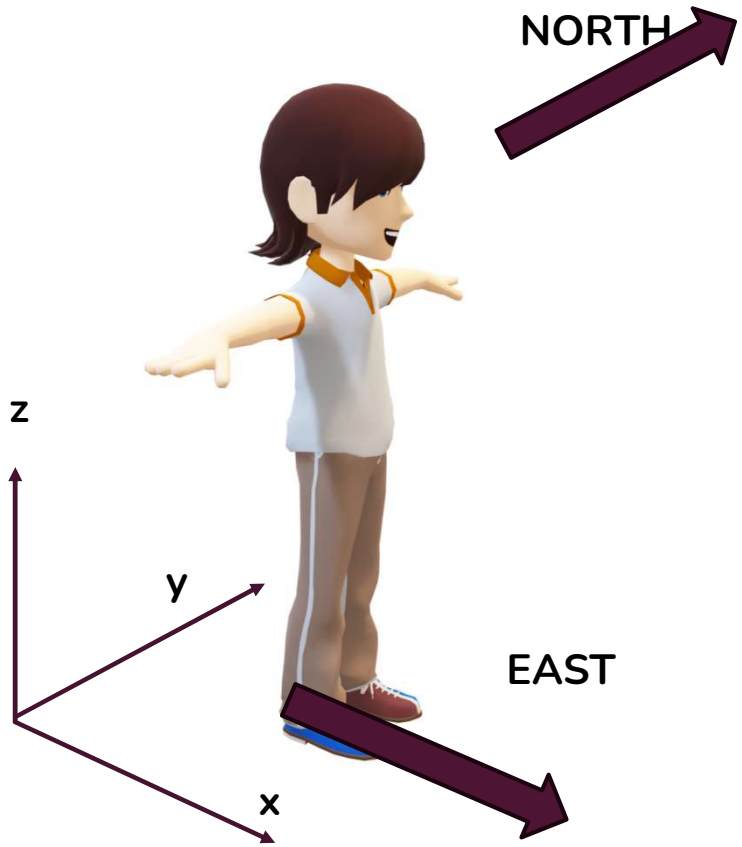
- The East-North-Up (ENU) reference frame is a right-hand frame with the axis as following:
  - Z points up to the sky, opposite the plumb line – towards the zenith at that point
  - Y points towards the true North
  - *X* points points towards East

- Consider a **person standing upright** on the ground.
- The orientation of the person is about **which direction the person is facing**.
- The ENU frame:
- is fixed to the Earth locally
- does not rotate with the person
- provides a common reference for orientation



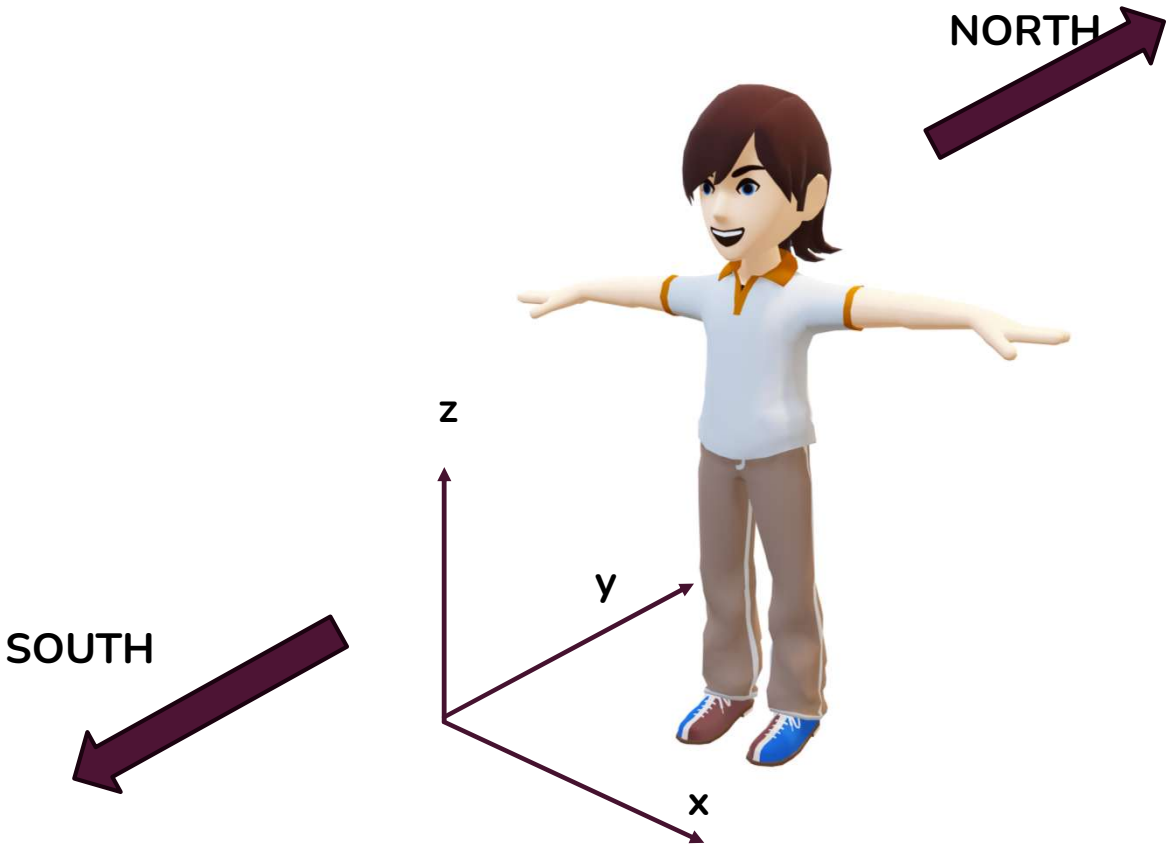
HEADING ANGLE

- For a standing person, orientation can be described by a single angle, called the **heading angle**
- The heading angle:
  - indicates the direction the person is facing
  - is measured in the **horizontal plane**
  - is defined with respect to **North**
- Examples:
  - heading = 0° → facing North
  - heading = 90° → facing East
  - heading = 180° → facing South
  - heading = 270° → facing West



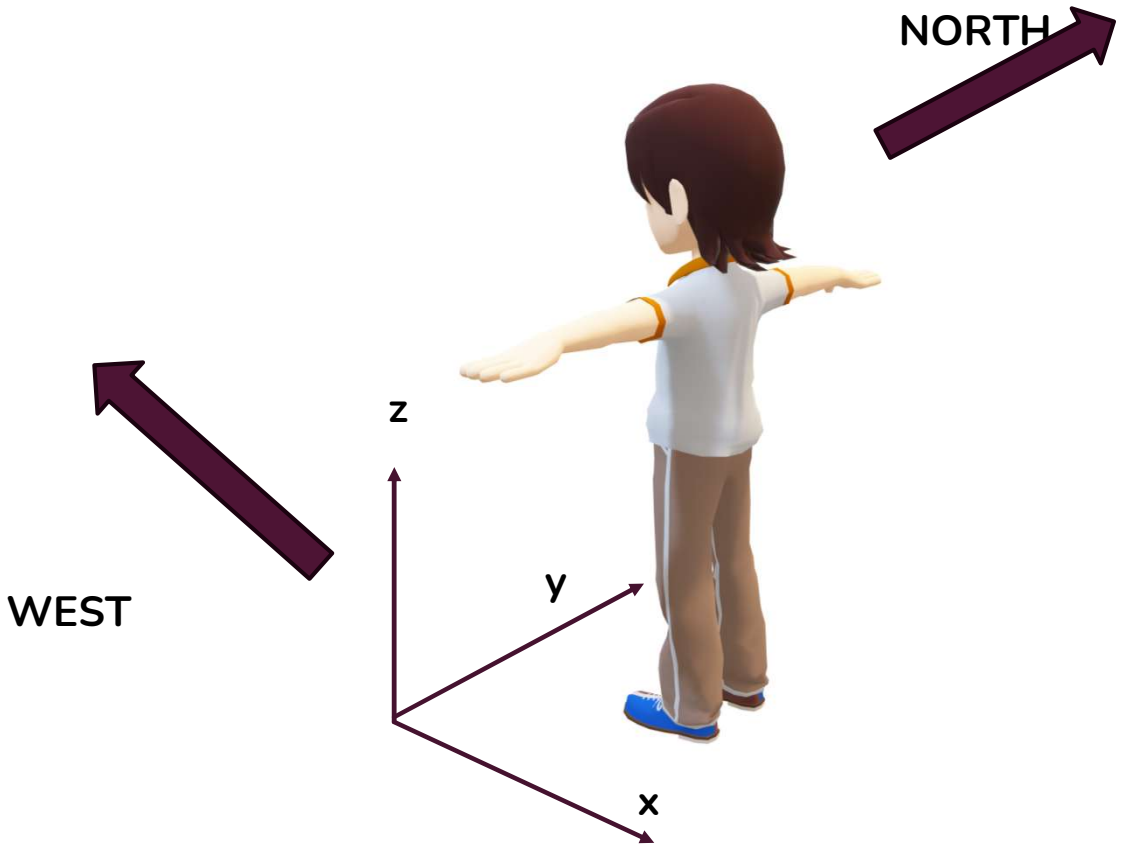
HEADING ANGLE

- Examples:
  - heading = 0° → facing North
  - heading = 90° → facing East
  - heading = 180° → facing South
  - heading = 270° → facing West



HEADING ANGLE

- Examples:
  - heading = 0° → facing North
  - heading = 90° → facing East
  - heading = 180° → facing South
  - heading = 270° → facing West

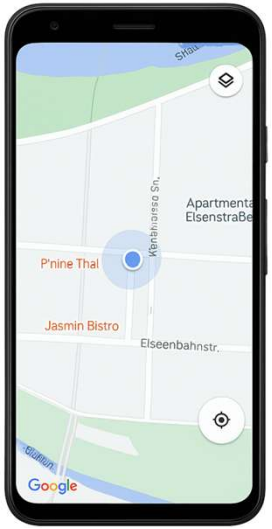


DEVICE REFERENCE FRAME

- The Device Reference Frame DRF is a right-handed orthonormal frame with the origin at the middle of the screen with
  - the Y axis pointing towards the top of the screen,
  - X to the right,
  - and Z out of the screen
- The readings from the sensors are given in the same reference frame
- When a smartphone is placed horizontally with the y axis pointing towards the north, the smartphone is aligned with the ENU frame
- The orientation of the smartphone is provided w.r.t. this rest orientation

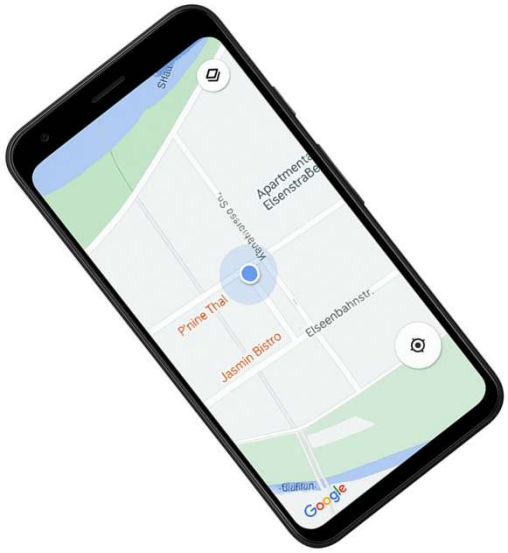
EXAMPLE OF ORIENTATION USAGE

- Google Maps rotates the map bitmap around the screen center by an angle equal to the user's heading.

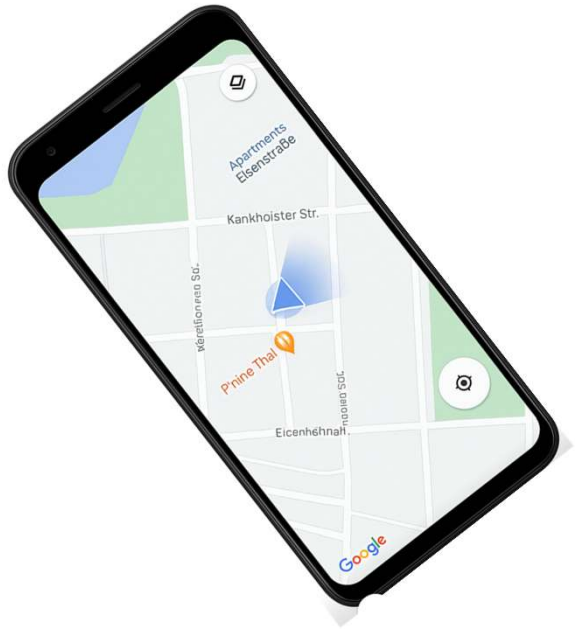


the smartphone points towards north (heading=0),

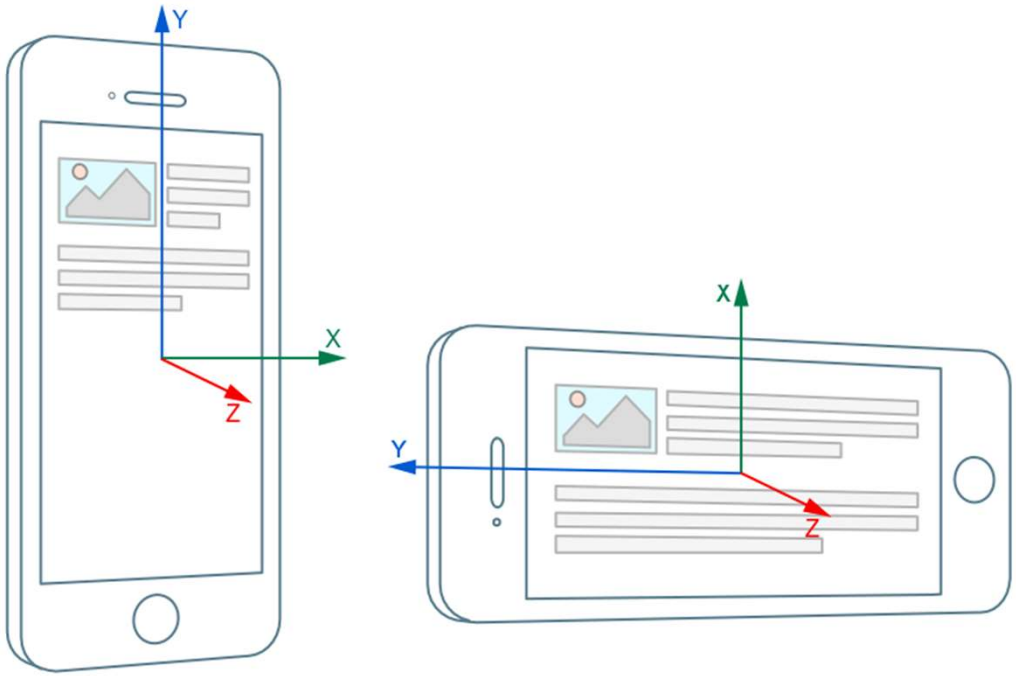
smartphone rotated of 30 degrees (heading=30)
Map not rotated



smartphone rotated of 30 degrees
Map rotated of -30



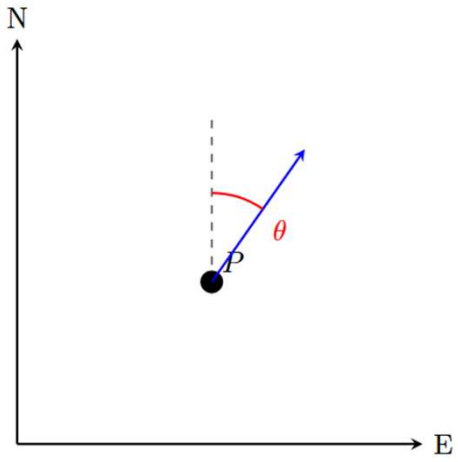
DEVICE REFERENCE FRAME



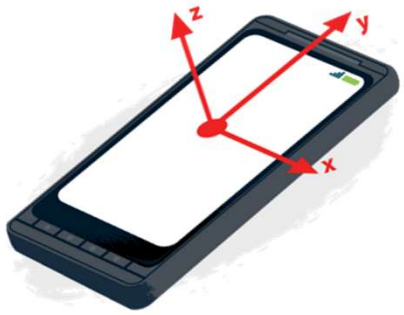
- The DFR doesn't change upon rotation from landscape to portrait, although it is possible to rename (remap) the axis

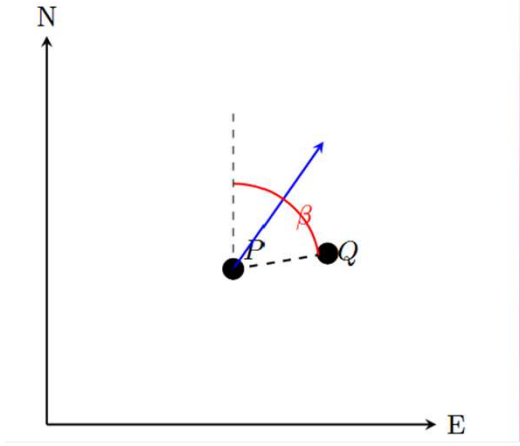
BEARING ANGLE

- Bearing angle = angle between the North direction and the line-of-sight from an observer to a target point (point Q in the figure).



Heading angle θ





Bearing angle β

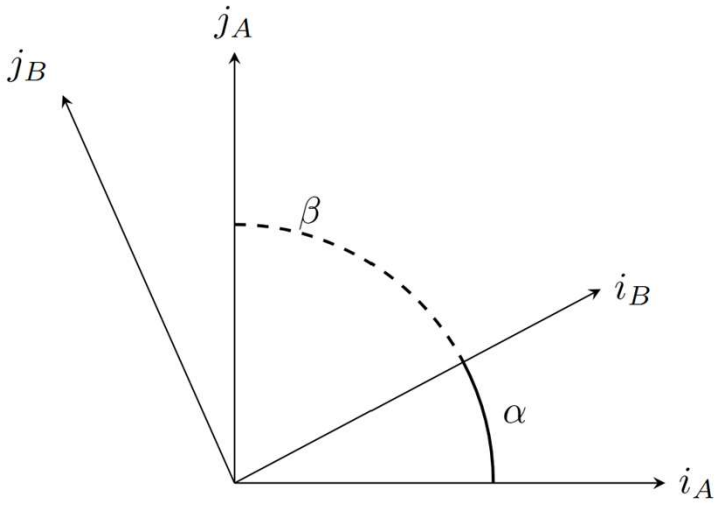
SMARTPHONE ORIENTATION

- The orientation of the device wrt to the ENU frame is not limited to the heading angle
- Orientation of the smartphone in the 3D space requires a more powerful representation
- Formally the problem consists in how to represent the orientation of the DRF in terms of the ENU frame

1. Orientation matrix (rotation matrix or Direction Cosine Matrix)
2. Orientation angles (Euler angles)
3. Quaternions

DIRECTION COSINE MATRIX

- From basic algebra we know how to change the coordinates of a vector **v** from one reference system A to another reference frame B using a change-of-basis matrix M

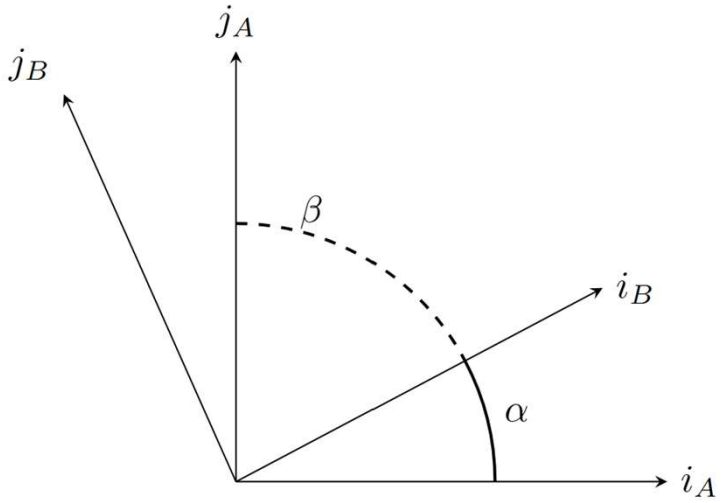


$$ {}^{B}M_{A} = \begin{bmatrix} i_A \cdot i_B & j_A \cdot i_B \\ i_A \cdot j_B & j_A \cdot j_B \end{bmatrix} \qquad \text{A}\rightarrow\text{B} $$

$$ {}^{A}M_{B} = \begin{bmatrix} i_B \cdot i_A & j_B \cdot i_A \\ i_B \cdot j_A & j_B \cdot j_A \end{bmatrix} \qquad \text{B}\rightarrow\text{A} $$

DIRECTION COSINE MATRIX

- Each column represents the coordinates of a base unit vector into the other reference frame
- For example, the coordinate of the unit vector i_B=(1,0) into the reference frame A are given below



$$ {}^{A}M_B \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} (i_B \cdot i_A)\,1 + (j_B \cdot i_A)\,0 \\ (i_B \cdot j_A)\,1 + (j_B \cdot j_A)\,0 \end{bmatrix} = \begin{bmatrix} i_B \cdot i_A \\ i_B \cdot j_A \end{bmatrix} $$

- Each element of the matrix represents the cosine of the angle between a unit vector of the original reference frame and a unit vector of the new reference frame.
- In this example, the versor i_B has cosines and with respect to the axes i_A and j_A.
- $i_B \cdot i_A = \cos\alpha$, $i_B \cdot j_A = \cos\beta$

DIRECTION COSINE MATRIX

- The change-of-basis matrix from B to A is the Direction Cosine Matrix of B relative to A

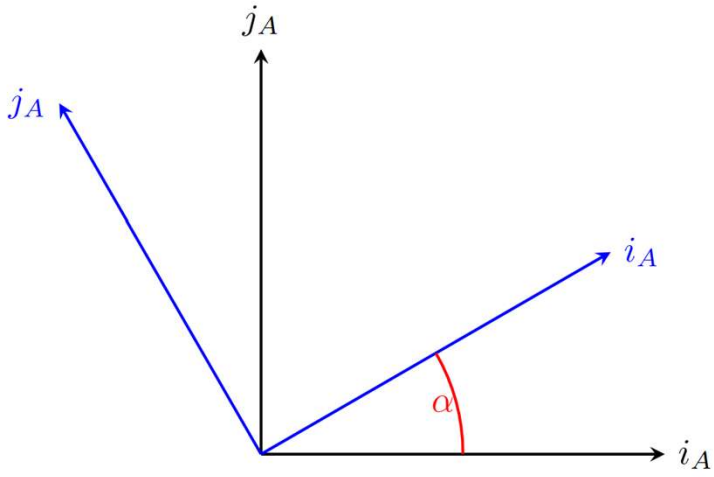
$$ {}^{A}M_{B} = {}^{A}DCM_{B} = \begin{bmatrix} i_B \cdot i_A & j_B \cdot i_A \\ i_B \cdot j_A & j_B \cdot j_A \end{bmatrix} $$

- **Property**: The inverse of a DCM matrix is the transpose of the matrix

ROTATIONS

- The DCM has another possible interpretation
- Because the axis of frames A and B are orthogonal, β=90-α and the same matrix is a function of a single angle α
- The matrix rotates the initial frame A of the angle α

$$R_A(\alpha) = {}^{A}DCM_B = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} = \begin{bmatrix} c_\alpha & -s_\alpha \\ s_\alpha & c_\alpha \end{bmatrix}$$



- Observation: in 2D the orientation depends on a single angle (with relative sign convention, either CW+ or CCW+)

ROTATIONS

$$\cos(\alpha+\beta) = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$

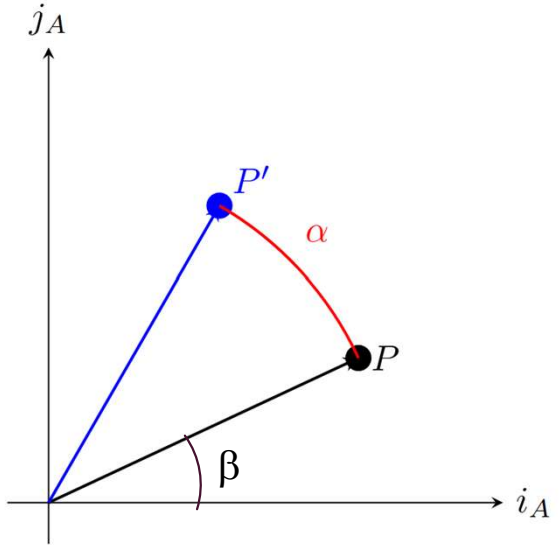
$$\sin(\alpha+\beta) = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$

- The same matrix can be interpreted as the operation ‘rotation’
- For example, the matrix R(α) rotates a point (x,y) of angle α

$$P' = R(\alpha)P = \begin{bmatrix}\cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha\end{bmatrix}\begin{bmatrix}\cos\beta \\ \sin\beta\end{bmatrix} = \begin{bmatrix}\cos(\alpha+\beta) \\ \sin(\alpha+\beta)\end{bmatrix}$$

$$x' = \cos\alpha\cos\beta - \sin\alpha\sin\beta$$

$$y' = \sin\alpha\cos\beta + \cos\alpha\sin\beta$$



$$P' = R(\alpha)\,P$$

3D ROTATION MATRIX IN ANDROID

getRotationMatrix

Added in API level 3

```
public static boolean getRotationMatrix (float[] R,
                float[] I,
                float[] gravity,
                float[] geomagnetic)
```

ENU frame

$^{\text{ENU}}R_{\text{DFR}}$

Computes the inclination matrix **I** as well as the rotation matrix **R** transforming a vector from the device coordinate system to the world's coordinate system which is defined as a direct orthonormal basis, where:

- X is defined as the vector product **Y.Z** (It is tangential to the ground at the device's current location and roughly points East).
- Y is tangential to the ground at the device's current location and points towards the magnetic North Pole.
- Z points towards the sky and is perpendicular to the ground.

Orientation matrix of the device respect to ENU

MEANING OF THE ROTATION MATRIX

- From the documentation the rotation matrix is change-of-basis matrix from the Device Reference Frame to ENU
- $R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$

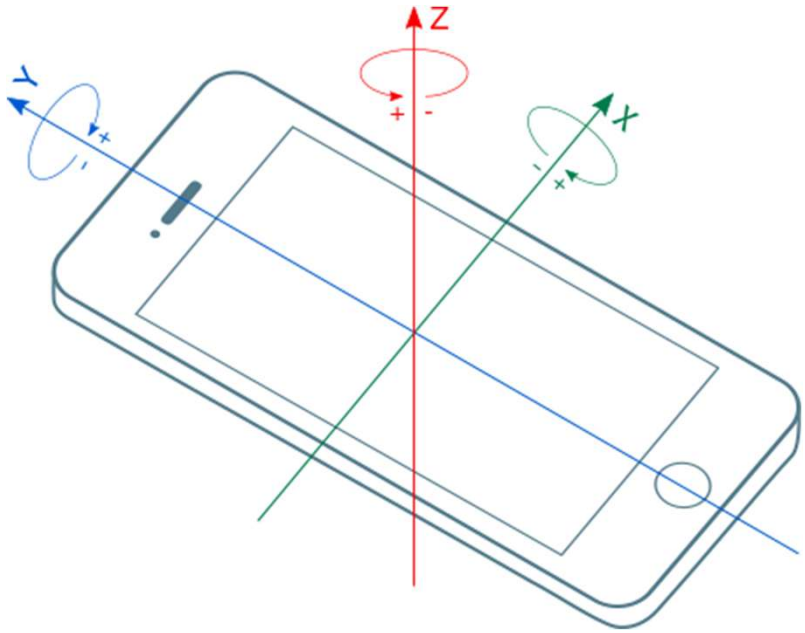
- Columns are the coordinates of the DRF axis in the ENU frame

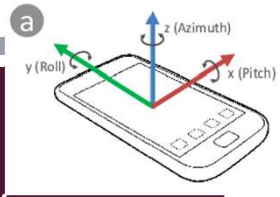
EXAMPLE OF USAGE OF ROTATION MATRIX

- The readings of vector values from the sensors are given using the DRF
- To compute the same reading in the ENU frame we can use the rotation matrix
- $\mathbf{m}_{ENU} = {}^{ENU}\mathbf{R}_{DRF}\,\mathbf{m}_{DRF}$

ORIENTATION ANGLE (METHOD 2)

- In many cases, it is worth to provide a clear physical meaning to the orientation.
- The orientation can be described by three angles commonly known as **yaw**, **pitch**, and **roll**.





ANGLE DEFINITION IN ANDROID (FROM DOCUMENTATION)

- **Azimuth (degrees of rotation about the -z axis).** This is the angle between the device's current compass direction and magnetic north. If the top edge of the device faces magnetic north, the azimuth is 0 degrees; if the top edge faces south, the azimuth is 180 degrees. Similarly, if the top edge faces east, the azimuth is 90 degrees, and if the top edge faces west, the azimuth is 270 degrees.

- **Pitch (degrees of rotation about the x axis).** This is the angle between a plane parallel to the device's screen and a plane parallel to the ground. If you hold the device parallel to the ground with the bottom edge closest to you and tilt the top edge of the device toward the ground, the pitch angle becomes positive. Tilting in the opposite direction— moving the top edge of the device away from the ground—causes the pitch angle to become negative. The range of values is -90 degrees to 90 degrees.

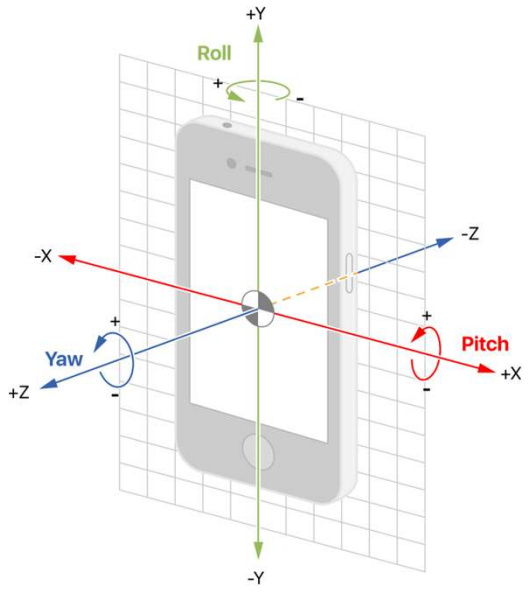
- **Roll (degrees of rotation about the y axis).** This is the angle between a plane perpendicular to the device's screen and a plane perpendicular to the ground. If you hold the device parallel to the ground with the bottom edge closest to you and tilt the left edge of the device toward the ground, the roll angle becomes positive. Tilting in the opposite direction—moving the right edge of the device toward the ground— causes the roll angle to become negative. The range of values is -180 degrees to 180 degrees.

RELASHIONSHIP BETWEEN ANGLES AND ROTATION MATRIX

$$R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix}$$ PITCH

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$ ROLL

$$R_z(\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$ YAW

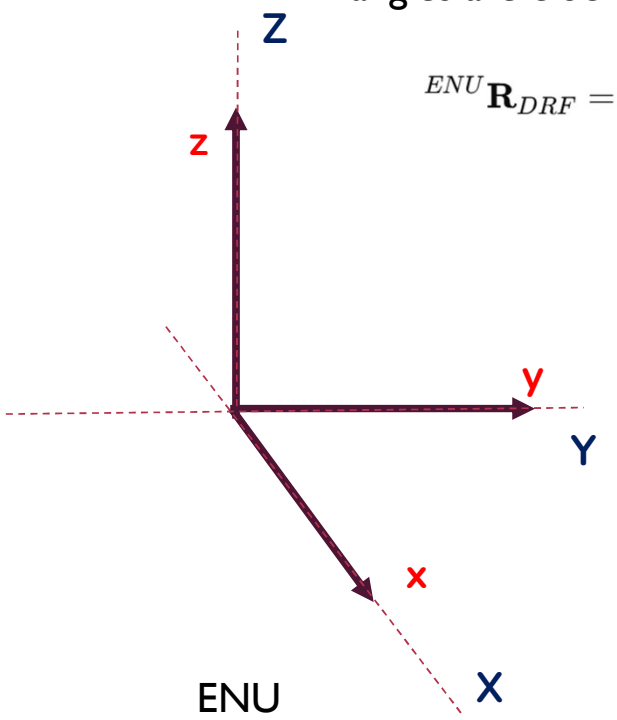


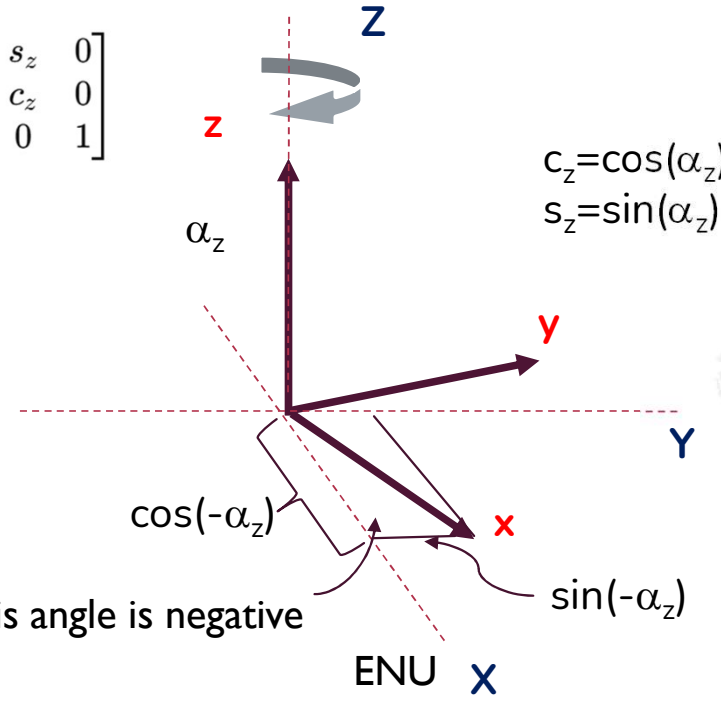
Each orientation angle describes a physical rotation of the device

EXAMPLE: YAW ORIENTATION ANGLE

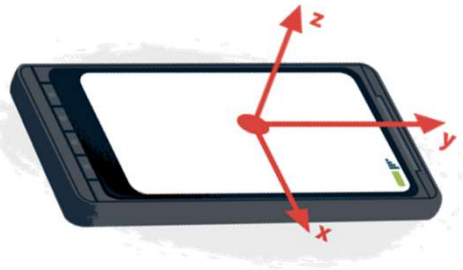
angles are clockwise positive CW+

$$^{ENU}\mathbf{R}_{DRF} = R_z = \begin{bmatrix} c_z & s_z & 0 \\ -s_z & c_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$





$c_z=\cos(\alpha_z)$
$s_z=\sin(\alpha_z)$

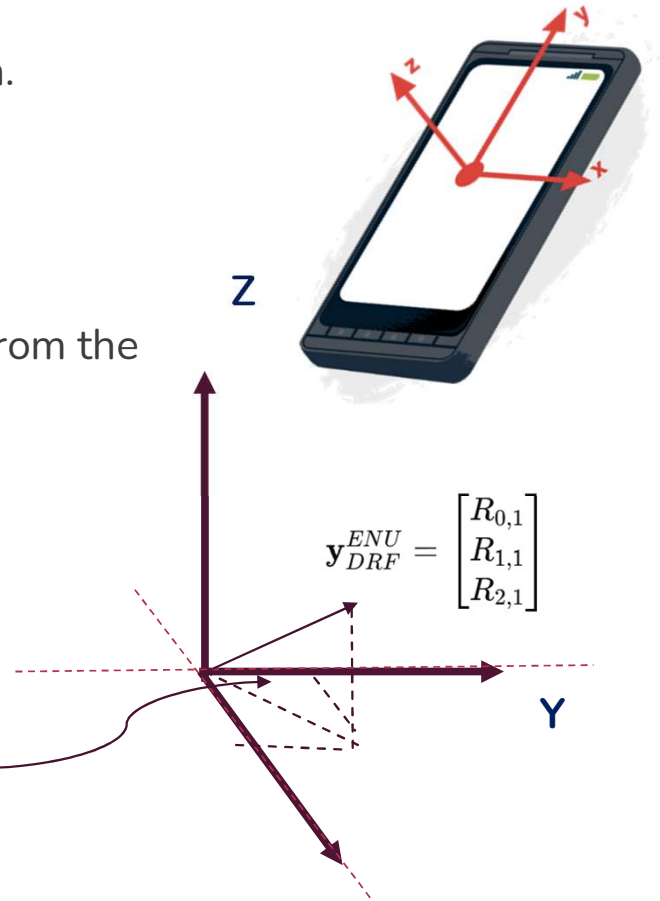


TILTED COMPENSATED COMPASS

- Heading angle is angle between the device's forward axis (y) and North.
- If the device is not tilted, then the bearing angle = yaw angle
- If the smartphone is tilted the heading angle is the angle between the projection of the y axis on the horizontal plane and the north direction
- From the documentation the rotation matrix is change-of-basis matrix from the Device Reference Frame to ENU

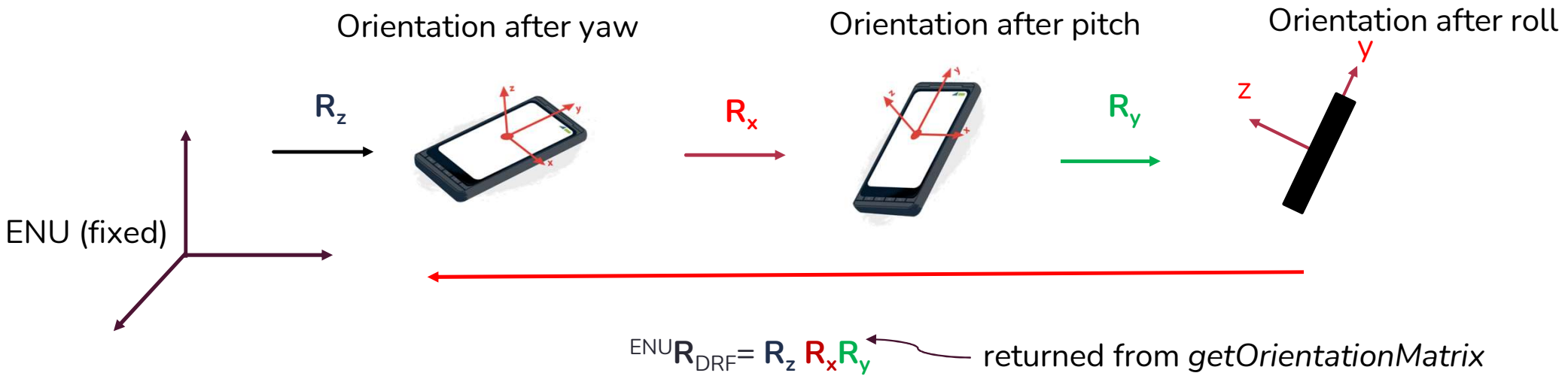
$$R = \begin{bmatrix} | & | & | \\ \mathbf{x}_{DRF}^{ENU} & \mathbf{y}_{DRF}^{ENU} & \mathbf{z}_{DRF}^{ENU} \\ | & | & | \end{bmatrix}$$

- Columns are the coordinates of the DRF axis in the ENU frame



RELASHIONSHIP BETWEEN ANGLES AND ROTATION MATRIX

- The general relationship between an orientation matrix and the rotation angles can be computed by exploiting the following result:
- Any 3D orientation can be generated through three successive rotations about three distinct axes.
  - The final result. depends on the order the rotations are applied. Android uses the convention yaw-pitch-roll
  - Given a rotation matrix, it is always possible to factorize the matrix as product of three elemental rotations (unique representations unless pith = +90 or -90)



ROTATIONS AND COMPLEX NUMBERS

$$\cos(\alpha+\beta)=\cos\alpha\cos\beta-\sin\alpha\sin\beta$$

$$\sin(\alpha+\beta)=\sin\alpha\cos\beta+\cos\alpha\sin\beta$$

- A 2D point $P=(x,y)$ can be represented as a complex number: z=x+iy
- Claim: A rotation of angle α corresponds to the multiplication by $e^{i\alpha}=\cos\alpha+i\sin\alpha$
- $z'=e^{i\alpha}z$
- $z'=(\cos\alpha+i\sin\alpha)\,(\cos\beta+i\sin\beta)=(\cos\beta\cos\alpha-\sin\beta\sin\alpha)+i(\cos\beta\sin\alpha+\sin\beta\cos\alpha)$
- An insight: The equivalence works because of how the multiplication among complex numbers is defined (in particular, i_2=-1)

INTERPOLATION (2D CASE)

- Let's suppose we want to rotate a point smoothly from α_1 to α_2
- One can think that a linear interpolation works fine: $\alpha(t)= (1-t)\ \alpha_1 + t\ \alpha_2$
- But… what happens when interpolating angles close two points α_1 =1° and α_2 =359° ?
- The intermediate angles depend on the order of the angles
- Exploit complex numbers provide a more numerically stable interpolation:
- $z(t) = (1-t)z_1 + tz_2$
- The result is normalized to encode a rotation
- This is called SLERP (**S**pherical **L**inear **E**nterpolation of **R**otational **P**oses)
- SLERP(z1,z2,t)=$\hat{z}(t) = \frac{(1-t)z_1+tz_2}{\|(1-t)z_1+tz_2\|}$.

INTERPOLATION (3D)

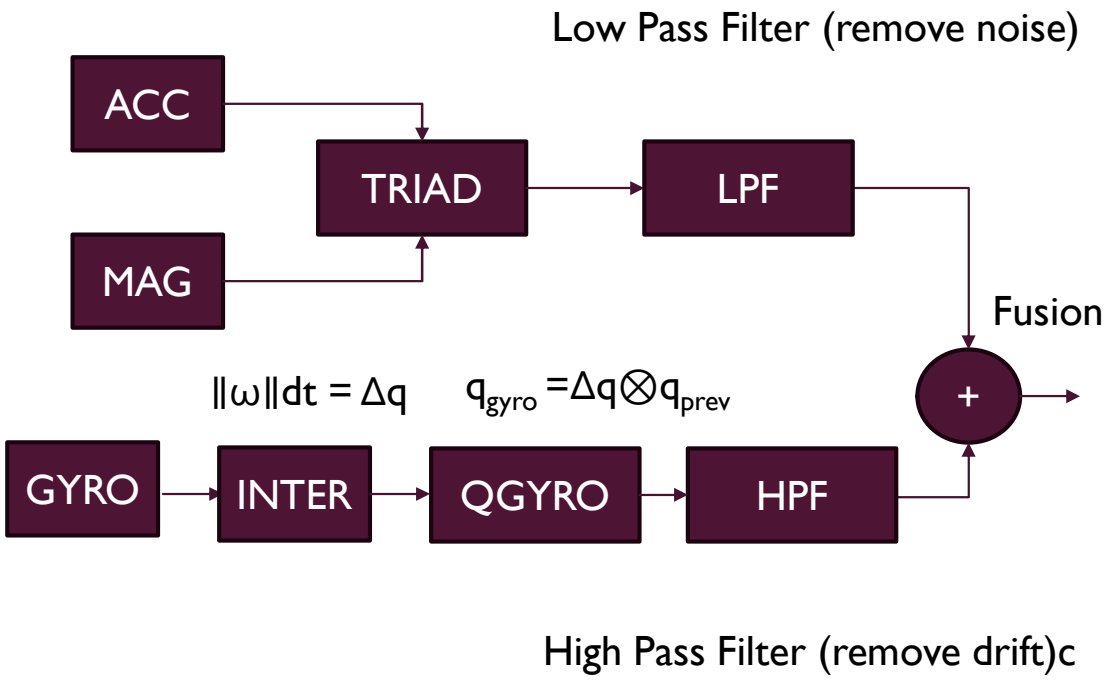
- It would be very useful to a similar techniques can be applied to 3D orientation,
- The new numbers are the unit quaternions
- In particular, when smartphone orientation changes overtime, the new orientation has a weight on the new value

$$q_{new}=SLERP(q_{old}, q_{measured},\alpha)$$

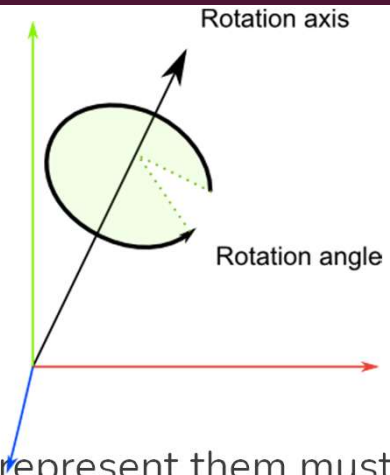
FUSION (3D) – MAIN IDEA

Orientation = LPF(acc+mag) + HPF (gyro)

- If two (o more) estimations of the same orientation are available, then they can be fused (merged) together
- For example, the gyroscope provides the current angular velocity of the smartphone.
- A simple fusion algorithm
  - Over a short time, interval the incremental rotations done can be estimated as $\theta=\|\omega\|dt \rightarrow \Delta q$
  - The new orientation is computed as $\Delta q \otimes q_{prev}$, where $\otimes$=Hamilton product
  - $q_{new}=SLERP(q_{gyro}, q_{acc}, \alpha)$



ROTATION IN 3 D



- In 3D, a rotation is defined by:
  - A rotation axis, **unit axis $\mathbf{u}$**
  - a **rotation angle** θ
- How to encode these 4 values as a single 'number'?
- Which kind of number do we need?
- Rotations in 3D are non-commutative, therefore any algebraic structure used to represent them must also be non-commutative.
- This was a major obstacle in defining a new "number system" for 3D rotations.
- Breakthrough: Hamilton product (the first non-commutative extension of complex numbers)
  - These numbers are the quaternions (with unit length): $q = \cos\frac{\theta}{2} + \mathbf{u}\ \sin\frac{\theta}{2}$ *with* $\| q \| = 1$
  - The reason why they work is in the way the multiplication is defined

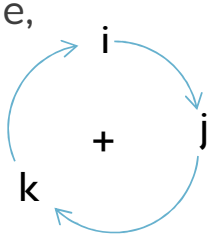
QUATERNIONS

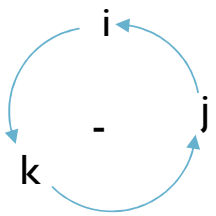
- Quaternions can be seen as **pairs of complex numbers** (and 'trions' cannot exit)
- A quaternion is: $\mathbf{q}=\mathbf{z}_1+\mathbf{j}\mathbf{z}_2$, $\mathbf{z}_1=(a_1+\mathbf{i}b_1)$, $\mathbf{z}_2=(a_2+\mathbf{i}b_2)$, where **j** is a new imaginary unit
- But how to define the product among **i** and **j** ?

FORMAL DEFINITION

- The mathematician Hamilton 'discovered' that the product among **i** and **j** defines another imaginary unit **k**:
- **ij=k**, **jk=i**,**ki=j**
- **ji=-k**, **kj=-i**,**ik=-j** (multiplication is not commutative, (revolutionary idea for that time, 1843)
- $\mathbf{i}^2=\mathbf{j}^2=\mathbf{k}^2=\mathbf{ijk}=-1$
- With these rules, **q=** w+**i**x+**j**y+**k**z
- **q** = (0,1,0,0) = **q=i** → 180 rotation around x
- **q** = (0,0,1,0) = **q=j** → 180 rotation around y
- **q** = (0,0,1,0) = **q=k** → 180 rotation around x

×	1	**i**	**j**	**k**
1	1	**i**	**j**	**k**
i	**i**	−1	**k**	−**j**
j	**j**	−**k**	−1	**i**
k	**k**	**j**	−**i**	−1





«Here as he walked by
on the 16th of October 1843
Sir William Rowan Hamilton
in a flash of genius discovered
the fundamental formula for
quaternion multiplication
$i^2 = j^2 = k^2 = i\,j\,k = -1$
and cut it on a stone of this bridge.»

QUATERNION AS GENERALIZATION

- For x=y=z=0, **q** is as a real number…
- For y=z=0, **q** is a complex number..
- … and for w=0, **q** embeds a 3D vector **v**=(x,y,z)!
- That is a ‘pure’ quaternion represents a 3D point

OPERATIONS

- The set of quaternions $\mathbb{H}$ has two operators + and · (not commutative group)
- Scalar-vector notation: $\mathbf{q}=w+\mathbf{v}$ or $\mathbf{q}=(w,\mathbf{v})$

- Given $\mathbf{q} = (w,\mathbf{v})$, $\mathbf{q}_1 = (w_1,\mathbf{v}_1)$ and $\mathbf{q}_2 = (w_2,\mathbf{v}_2)$

- **Addition**: $\mathbf{q}_1 + \mathbf{q}_2 = (w_1+w_2,\mathbf{v}_1+\mathbf{v}_2)$
- **Multiplication**: $\mathbf{q}_1 \otimes \mathbf{q}_2 = (w_1w_2 - \mathbf{v}_1 \mathbf{v}_2, w_1\mathbf{v}_2 + w_2\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2)$ [Hamilton product] $(\mathbf{q}_1 \otimes \mathbf{q}_2 \neq \mathbf{q}_2 \otimes \mathbf{q}_1)$
- **Conjugate**: $\mathbf{q} = (w,-\mathbf{v})$
- **Inverse**: $\mathbf{q}^{-1} = 1/|\mathbf{q}|^2 (w,-\mathbf{v})$
- **Length**: $|\mathbf{q}|^2=w^2+x^2+y^2+z^2$

UNIT QUATERNIONS REPRESENT ROTATION

- Let **v** be a unit vector representing the axis, and θ the rotation angle.
- The unit rotation quaternion is representing the rotation is:

$$r = \left(\cos\frac{\theta}{2},\ \sin\frac{\theta}{2}\mathbf{v}\right)$$

  - For example:
  - Pitch: **q**=cos (θ/2) + **i** sin (θ/2)
  - Roll: **q**=cos (θ/2) + **j** sin (θ/2)
  - Yaw: **q**=cos (θ/2) + **k** sin (θ/2)

UNIT QUATERNIONS REPRESENT ROTATION

- Given a unit quaternion **q** = (w,**v**)
- the rotation angle is $\theta = 2\cos^{-1}(w)$,
- The rotation axis is $\mathbf{u} = \mathbf{v}/|\mathbf{v}| = \frac{1}{\sqrt{1-w^2}}\mathbf{v}$

ROTATION BY QUATERNIONS

- Let **v** be a unit vector representing the axis, and θ the rotation angle.
- The unit rotation quaternion is:

$$r = \left(\cos\frac{\theta}{2},\ \sin\frac{\theta}{2}\mathbf{v}\right)$$

  - The inverse is

$$r^{-1} = \left(\cos\frac{\theta}{2},\ -\sin\frac{\theta}{2}\mathbf{v}\right)$$

  - Rotation of point p

$$\boxed{p' = r\,p\,r^{-1}}$$

EXAMPLE

- Axis:**v** = (0, 0, 1)
- Rotation of point **p** = (x, y , z) of θ

$$r = \cos\frac{\theta}{2} + k\sin\frac{\theta}{2}$$

- Result

$$p' = rpr^{-1} = (0, \mathbf{p}')$$

$$\mathbf{p}' = (x\cos\theta - y\sin\theta,\ x\sin\theta + y\cos\theta,\ z)$$

- It is possible to provide a geometric proof why it works, or a symbolic proof

ROTATION COMPOSITION

- Given two rotations, $\mathbf{R}_1$ and $\mathbf{R}_2$, the rotation $\mathbf{R}=\mathbf{R}_1\mathbf{R}_2$ is represented by the quaternion $\mathbf{q}_1\mathbf{q}_2$, where $\mathbf{q}_1$ represents $\mathbf{R}_1$ and $\mathbf{q}_2$ $\mathbf{R}_2$
- To compute any element of the product of two matrixes we need 2 additions and 3 multiplications (ab+a'b'+a''b''), so in total the computation cost is:
- 18 additions (9x2)
- 27 multiplications (9x3)
- The complexity of quaternion multiplication is 12 additions and 16 multiplications, that is 44% less multiplications and 33% less additions

$$
\begin{aligned}
a_3 &= a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\
b_3 &= a_1b_2 + a_2b_1 + c_1d_2 - c_2d_1 \\
c_3 &= a_1c_2 + a_2c_1 + b_2d_1 - b_1d_2 \\
d_3 &= a_1d_2 + a_2d_1 + b_1c_2 - b_2c_1.
\end{aligned}
$$

ROTATION VECTOR AND GAME ROTATION VECTOR

- Compact representation of quaternion as the vector part of a unit quaternion
- The **rotation vector** is a virtual sensor that returns the vector **v**=**e** $\sin(\theta/2)$
  - **e** =(x,y,z) is the rotation axis, to mean **q**=$\cos \theta/2$+ **e** $\sin \theta/2$
  - the angle $\theta/2$ can is encoded in the length |**v**|
- The rotation vector is estimated using accelerometer, magnetometer and gyroscope
- The **game rotation vector** sensor is identical to the rotation vector, except is not calibrated (it is used to retrieve relative orientations)

SOME USEFUL METHOS IN ANDROID

- **getRotationMatrixFromVector(float[] R, float[] rotationVector):**
- Converts a rotation vector to a rotation matrix.
- **getQuaternionFromVector(float[] Q, float[] rotationVector)**Converts a rotation vector into a quaternion.
- **getRotationMatrix(float[] R, float[] I, float[] gravity, float[] geomagnetic)**
- Computes the rotation matrix from accelerometer (gravity) and magnetometer (geomagnetic) data.
- **getOrientation(float[] R, float[] values)**
- Converts a rotation matrix to Euler angles (yaw, pitch, and roll).
- **remapCoordinateSystem(float[] inR, int X, int Y, float[] outR)**
- Adjusts the rotation matrix to be aligned with a different axis configuration.

SOME USEFUL METHOS IN ANDROID

getRotationMatrix (from sensors)

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}$$

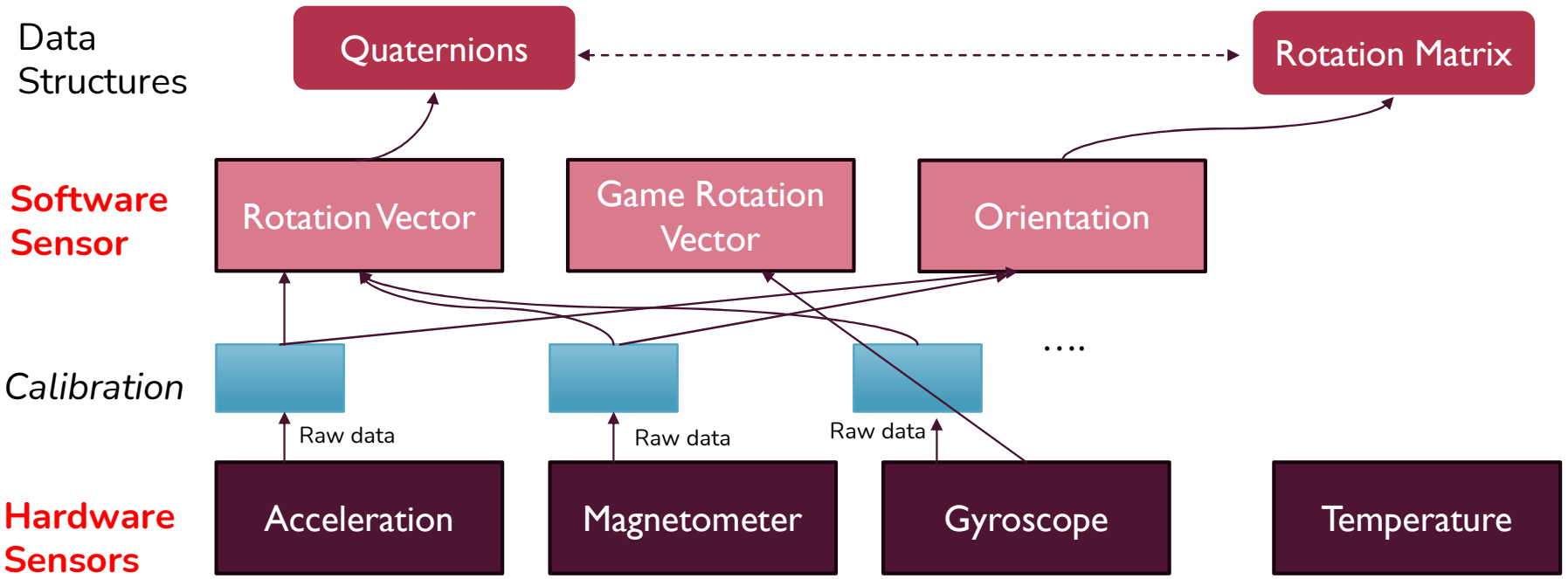
Rotation Matrix —*getOrientation*→ **Euler Angles** ψ, θ, ϕ

getRotationMatrixFromVector (**Rotation Matrix** → **Rotation Vector**)

getRotationVector (from sensor fusion)

Rotation Vector —*getQuaternionFromVector*→ **Quaternions**

SENSOR IN ANDROID

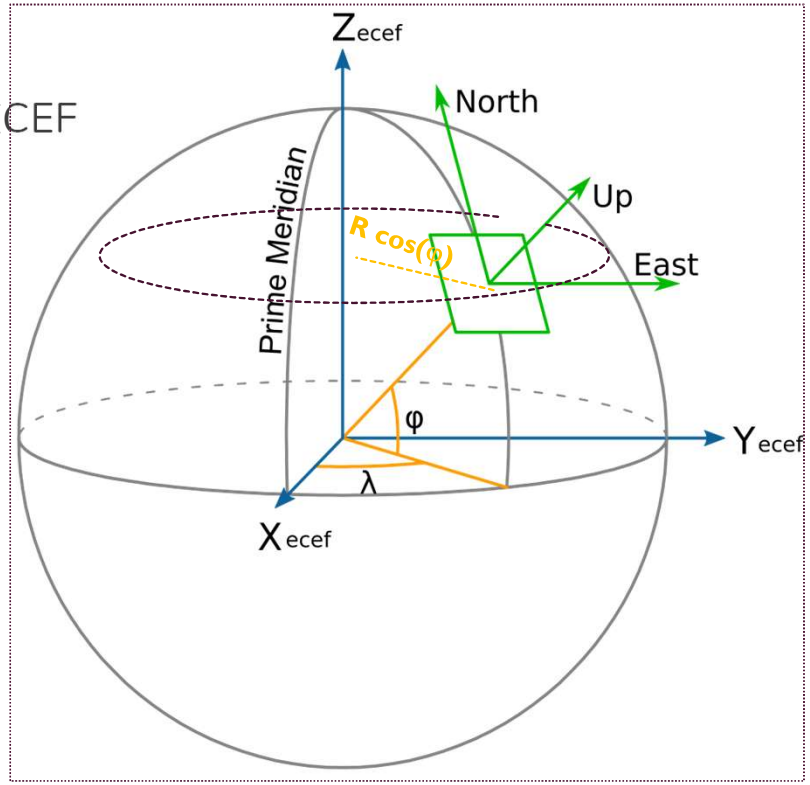


EXAMPLE: COMPUTE THE DCM OF THE ECEF FRAMEWORK RESPECT TO THE ENU FRAME

- Assume the two frames have the same origin
- It is easier to compute the direction cosines of the ENU frame in the ECEF frame
  - Direction cosines of the **UP =** (cos(φ)cos(λ),cos(φ)sin(λ),sin(φ))
  - Direction cosines of the **EAST**=(-sin(λ),cos(λ),0) – no z component
  - Direction cosines of **NORTH = UP x EAST**

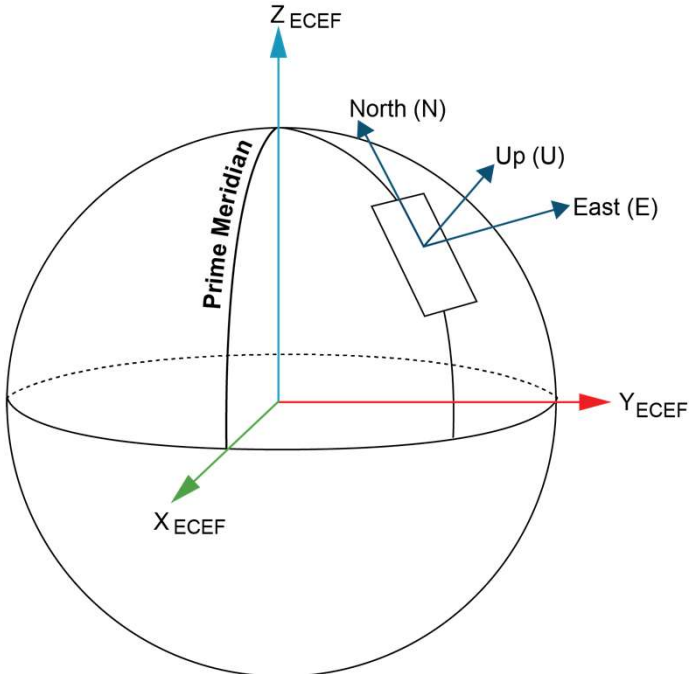
$$\mathbf{E} = (-\sin\lambda,\ \cos\lambda,\ 0)$$
$$\mathbf{N} = (-\sin\varphi\cos\lambda,\ -\sin\varphi\sin\lambda,\ \cos\varphi)$$
$$\mathbf{U} = (\cos\varphi\cos\lambda,\ \cos\varphi\sin\lambda,\ \sin\varphi)$$

$$^{\mathrm{ENU}}\mathrm{DCM}_{\mathrm{ECEF}} = \begin{bmatrix} -\sin\lambda & \cos\lambda & 0 \\ -\sin\varphi\cos\lambda & -\sin\varphi\sin\lambda & \cos\varphi \\ \cos\varphi\cos\lambda & \cos\varphi\sin\lambda & \sin\varphi \end{bmatrix}$$



EXAMPLE

- Now that we have this matrix, we can express the coordinate of any object in the ECEF frame in the ENU frame
- In case of 'very far' objects, the matrix is still useful to derive the coordinate of these points (see later)



$$\begin{bmatrix} E \\ N \\ U \end{bmatrix} = {}^{\mathbf{ENU}}M_{ECEF} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}_{ECEF}$$

EXAMPLE

$${}^{ECI}\mathbf{r}_{sat} = \begin{bmatrix} x_{sat} \\ y_{sat} \\ z_{sat} \end{bmatrix}$$

$${}^{ECEF}M_{ECI} = R_z(-\theta) = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$= GMST$ (Greenwich Mean Sidereal Time).

$$\boxed{{}^{NEU}\boldsymbol{\rho} = {}^{NEU}M_{ECEF}\ {}^{ECEF}M_{ECI}\ {}^{ECI}\mathbf{r}_{sat} - {}^{NEU}M_{ECEF}\ {}^{ECEF}\mathbf{r}_{obs}}$$

depends on the current time

depends on latitude and longitue

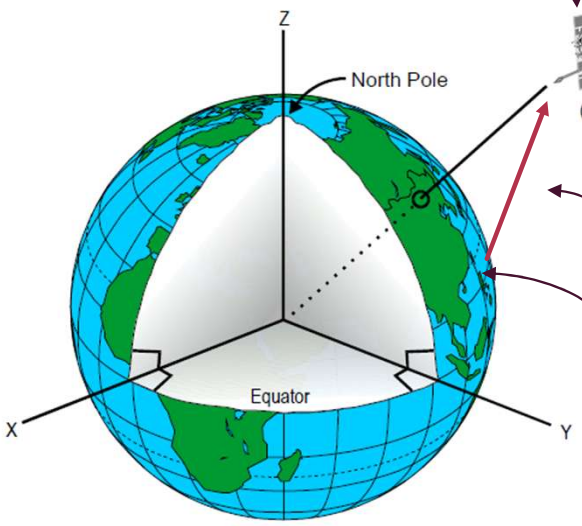
$${}^{\mathbf{ENU}}\boldsymbol{\rho} = {}^{\mathbf{ENU}}M_{ECEF}\ {}^{ECEF}\boldsymbol{\rho}$$

$${}^{ECEF}\boldsymbol{\rho} = {}^{ECEF}\mathbf{r}_{sat} - {}^{ECEF}\mathbf{r}_{obs}$$

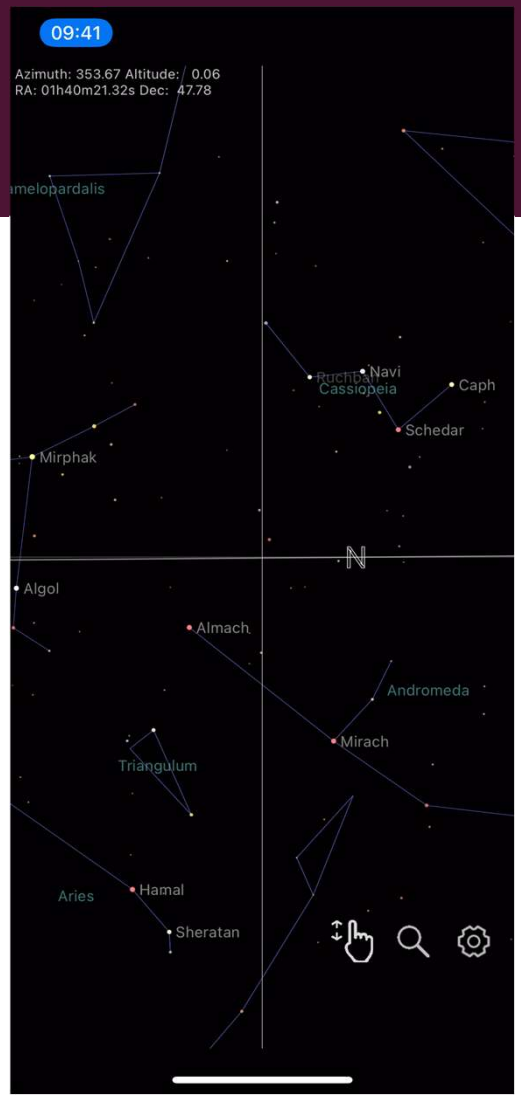
$${}^{ECEF}\mathbf{r}_{sat} = {}^{ECEF}M_{ECI}\ {}^{ECI}\mathbf{r}_{sat}$$

$${}^{\mathbf{ENU}}\boldsymbol{\rho} = {}^{\mathbf{ENU}}M_{ECEF}\ {}^{ECEF}\boldsymbol{\rho}$$

$${}^{ECEF}\mathbf{r}_{obs} = \begin{bmatrix} x_{obs} \\ y_{obs} \\ z_{obs} \end{bmatrix}$$



SKYMAP


09:41
Azimuth: 353.67 Altitude: 0.06
RA: 01h40m21.32s Dec: 47.78
Navi
Caph
Schedar
Mirphak
N
Algol
Almach
Andromeda
Mirach
Triangulum
Aries
Hamal
Sheratan