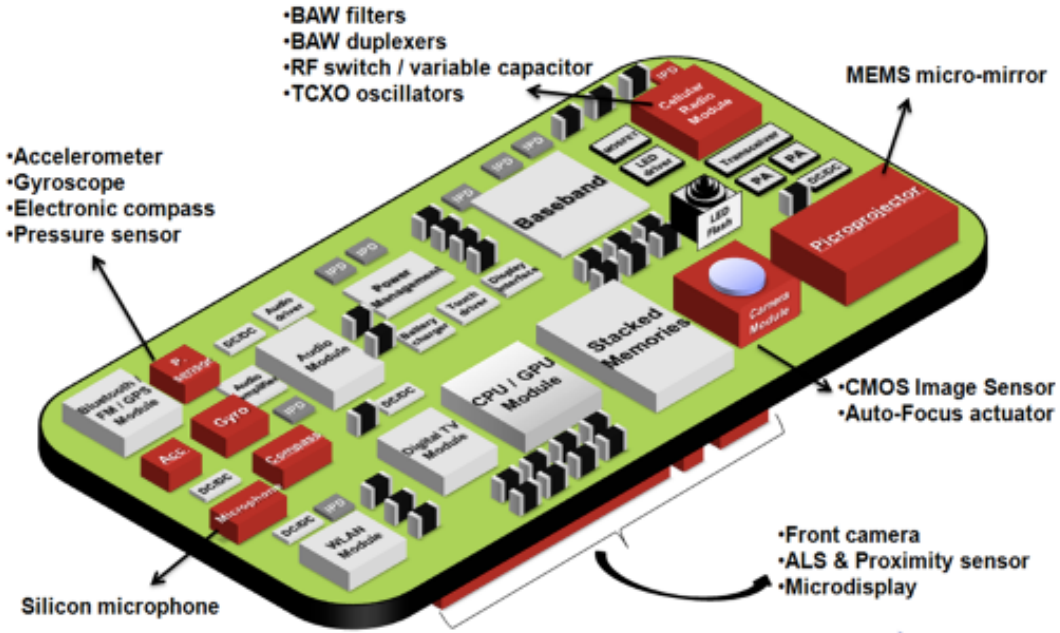
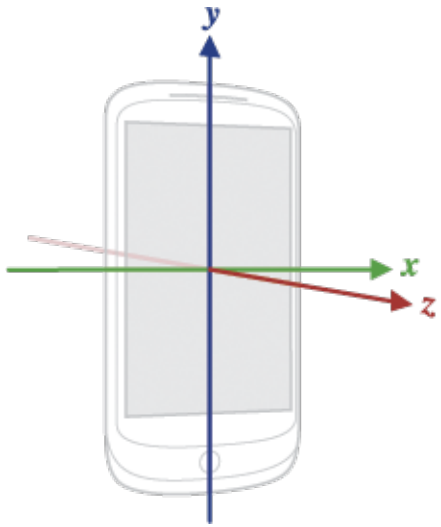
SENSORS AND ORIENTATION

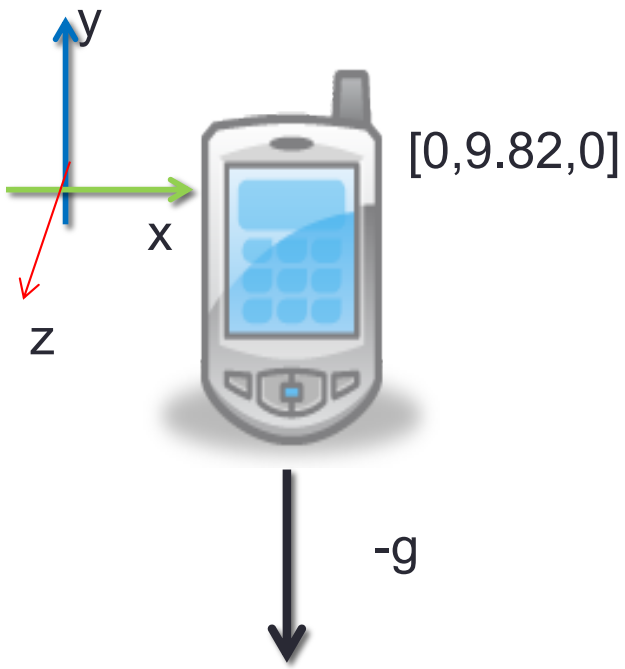
Smartphone sensors

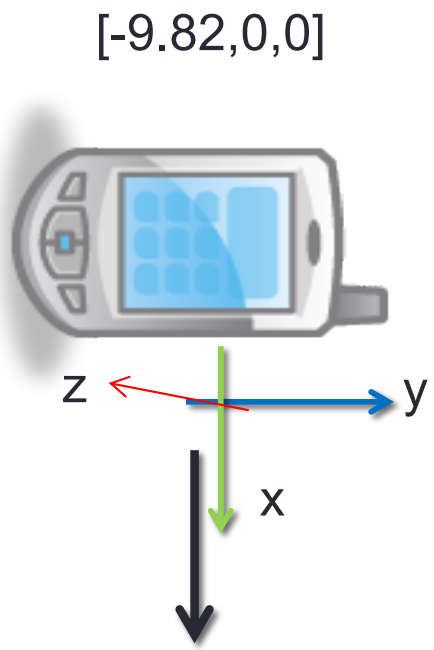


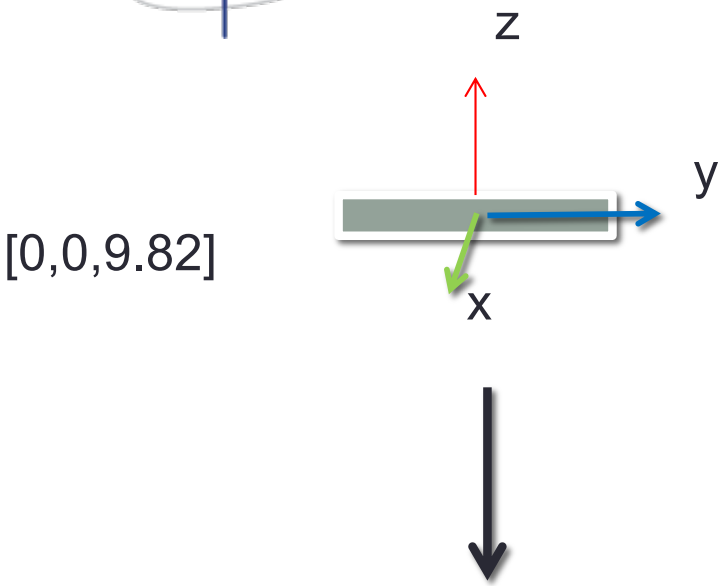
All these sensors allow to be aware of the environment:
(where we are, how we are moving, pressure, .. etc)
These info enlarge the way a user interacts with the smartphone
An application has to take awareness into account

Example

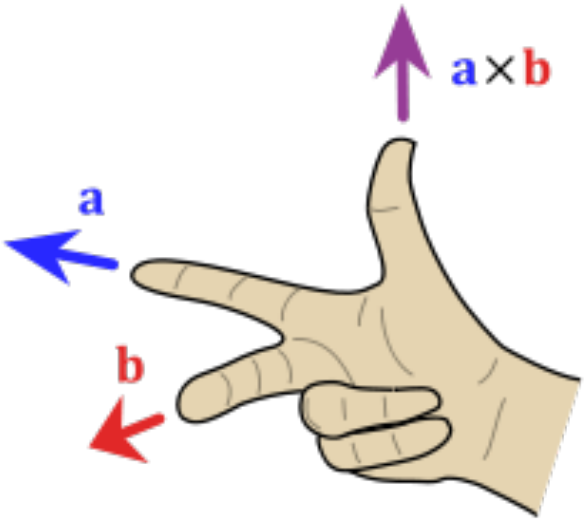






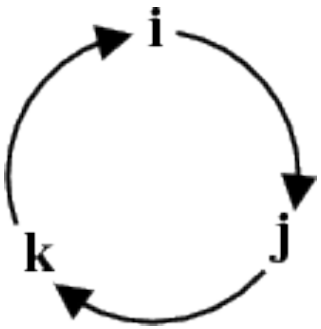


Right-hand frames and right-hand rule

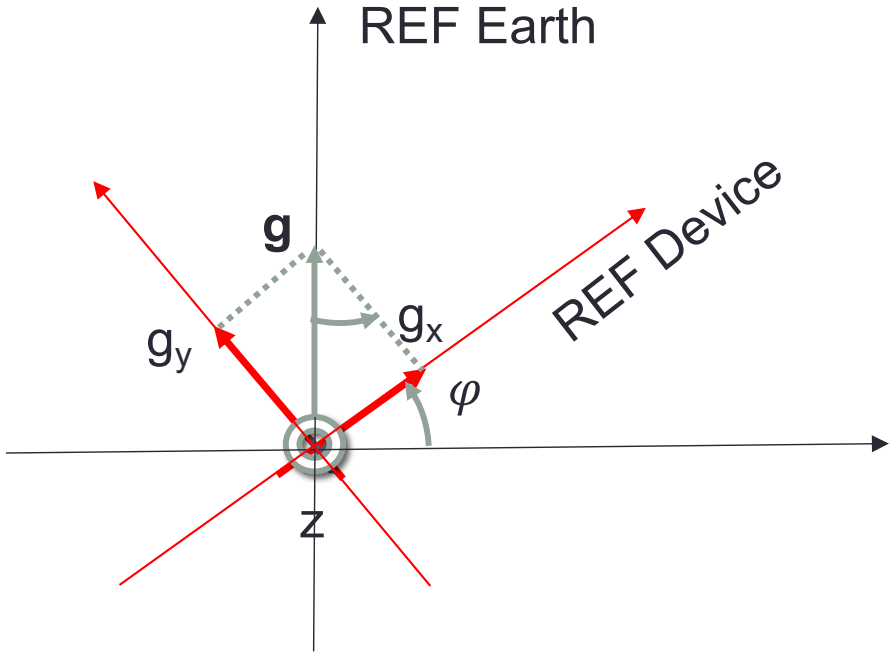


Points the forefinger of the right hand in the direction of **a** and the middle finger in the direction of **b**. Then, the vector **n** is coming out of the thumb

Right-hand coordinate system



Orientation simple 2D case



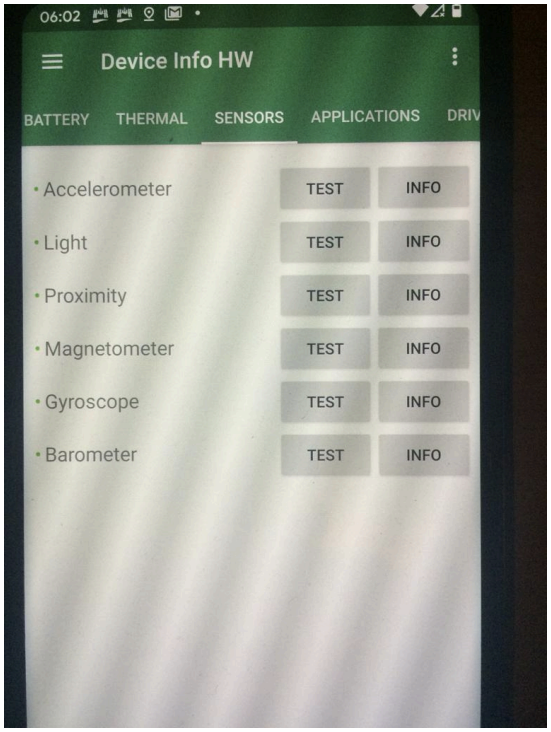
- $\mathbf{g}_E=(0,g,0)$
- $\mathbf{g}_D=(g_x,g_y,0)$
- $\mathbf{g}_D=(g\,\mathrm{sen}\,\phi\,,\,g\cos\phi,0)$

$$\varphi = \mathrm{atan}(\frac{g_x}{g_y})$$

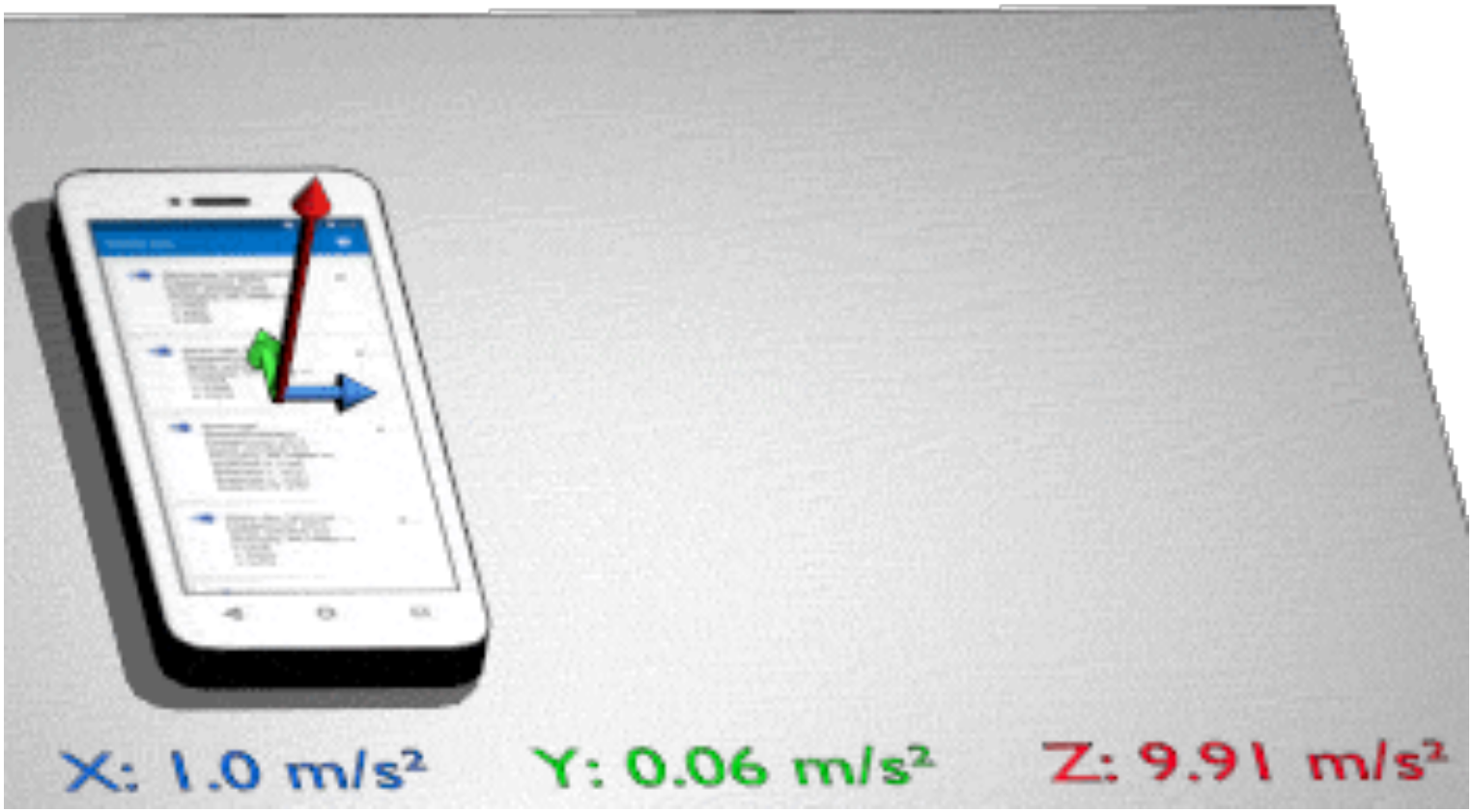
- A gravity sensor is needed to compute the angle φ , i.e., the device orientation

Sensor implementation

- They are Micro-Electro-Mechanical Systems (MEMS)
- Used to measure some **physical** quantity.
- Two parts:
  - Mechanical part that exploits some physical law
  - Electrical part: used to transduce the mechanical quantity into an electrical readable value (e.g., acceleration → volts)



Accelerometer



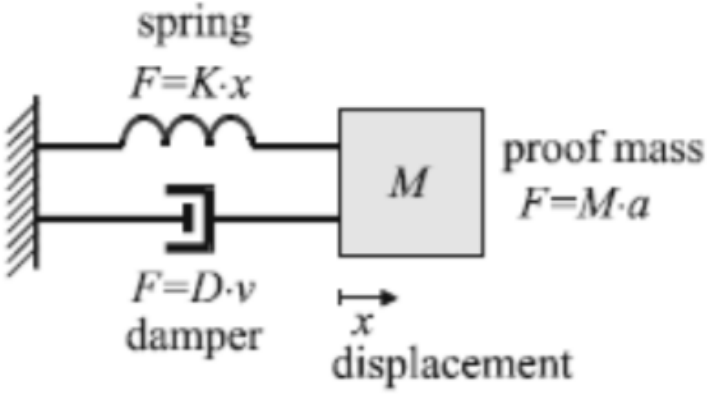
Credits: https://developers.google.com/web/updates/2017/09/sensors-for-the-web

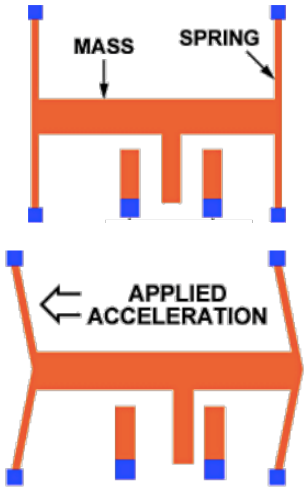
Inertial Sensors (Inertial Motion Unit)

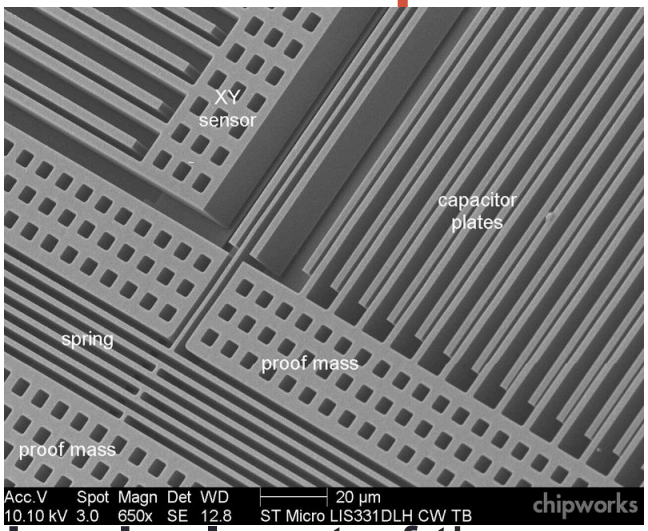
- Exploit the concept of inertia and the classical Newton's Law

$$\mathbf{F} = m\mathbf{a}$$

Acceleration sensor: mechanical part



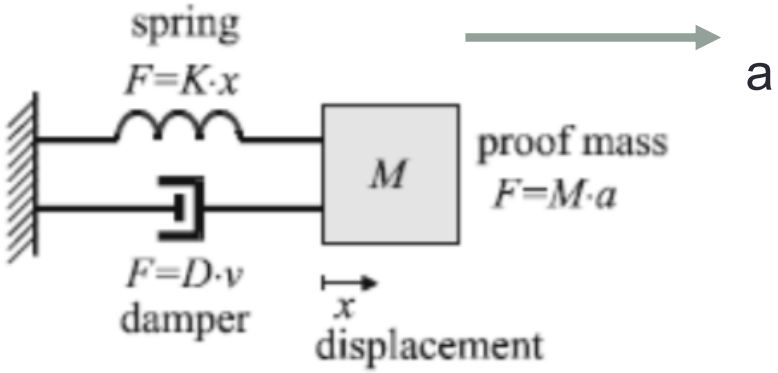




- The figure represents a model of the **mechanical** part of the sensor
- Suppose the system that hosts the sensor is accelerated by a
- The proof mass sees three forces:

1. Force due to its inertia proportional to the external acceleration: F=Ma
2. Damper force proportional to the speed of the mass: F= D v = D $\dot{x}$
3. Spring force proportional to the displacement of the mass: F=K x

Acceleration sensor



- The total force on on the mass is then

$$\boldsymbol{F = Ma - D\,\dot{x} - K\,x}$$

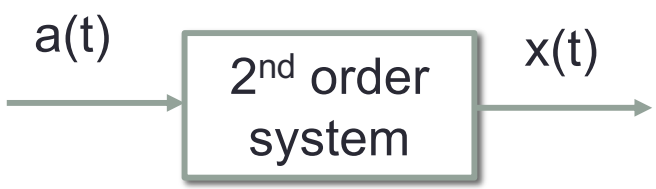
- Applying the Newton's second law to the mass ($F = M\,\ddot{x}$) one gets
  - $Ma - D\,\dot{x} - K\,x = M\,\ddot{x}$
- Note: the external acceleration (a) is different from the acceleration of the proof mass ($\ddot{x}$)

$$a = \ddot{x} + \frac{D}{M}\,\dot{x} + \frac{K}{M}\,x$$

We need to express x as a function of a

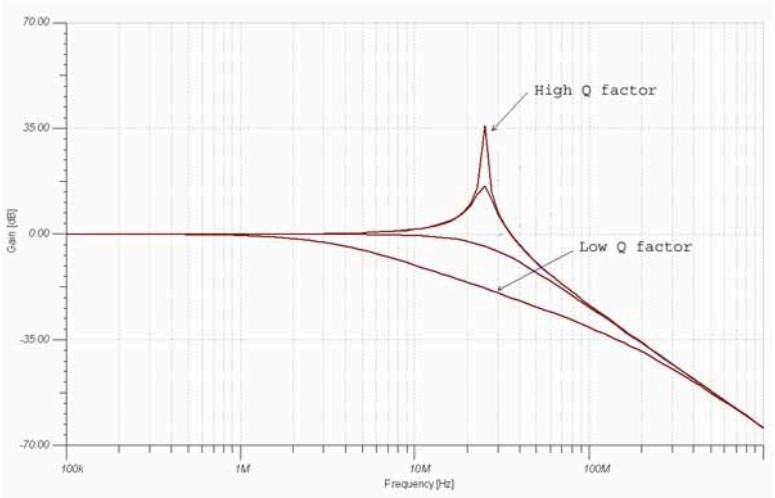
Acceleration sensor

- $a = \ddot{x} + \frac{D}{M}\dot{x} + \frac{K}{M}x \rightarrow a(s) = s^2x(s) + \frac{D}{M}s\,x(s) + \frac{K}{M}x(s)$
- $\omega_n = \sqrt{\frac{K}{M}}, Q = \frac{M\omega_n}{D} \rightarrow a(s) = s^2x(s) + \frac{\omega_n}{Q}s\,x(s) + \omega_n^2\,x(s)$
  - $x(s) = \frac{a(s)}{s^2 + \frac{\omega_n}{Q}s + \omega_n^2}$



Well known system

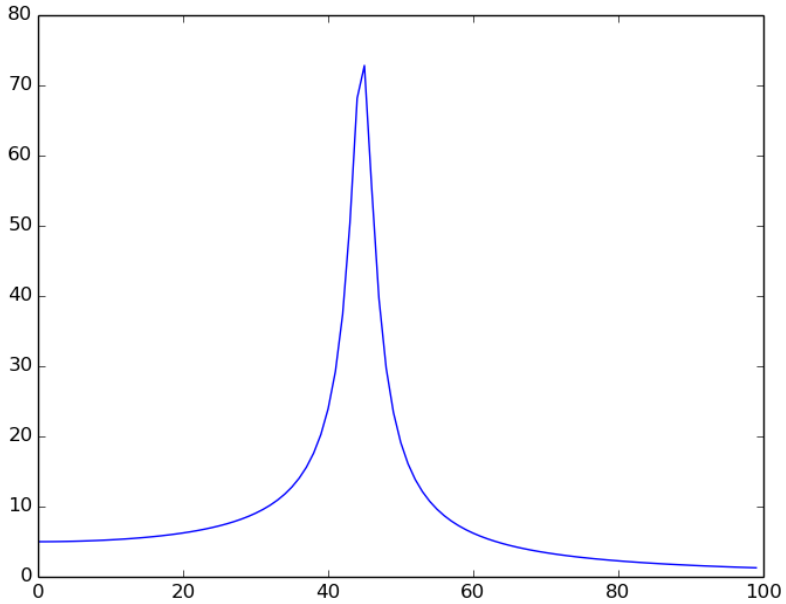
Acceleration sensor



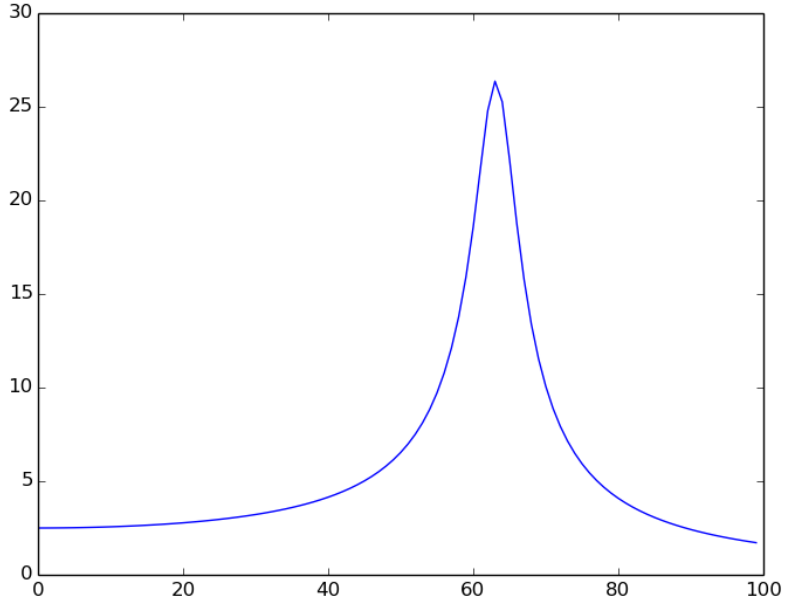
- For s=0 (at steady state):
- $x = \frac{M}{K}a$
- $\frac{M}{K}$ is the **sensitivity** of the sensor
- High values are better; can be achieved with higher M or lower K but..
- The **natural frequency** is $\omega_n = \sqrt{\frac{K}{M}}$ decreases with the sensitivity
- To keep the a-x relationship linear the range of linear response

Numerical Example

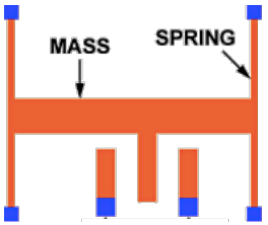
K=20.0
D=3.0
M=100.0
ω=44.72 ($x10^{-2}$)

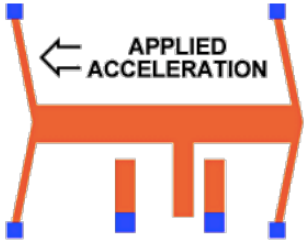


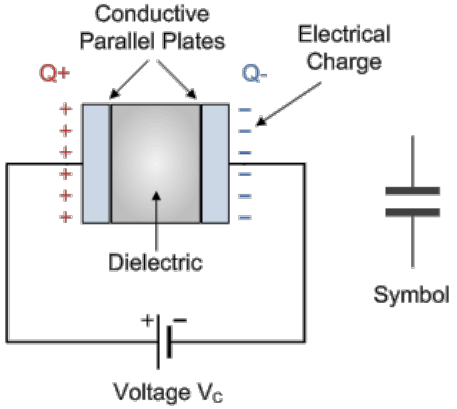
K=20.0
D=3.0
M=50.0
ω=63.24 ($x10^{-2}$)



Accelerometer (the electrical side)







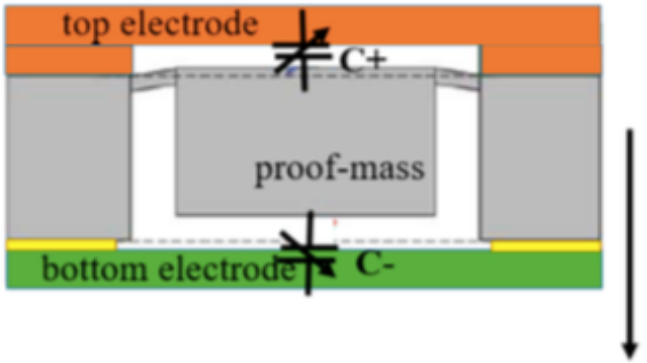
- Two plates of the device are a capacitor
- Simple electrical model of the capacitor is the parallel plates capacitor
- Recall that the capacity is

$$C = \varepsilon \frac{A}{x}$$

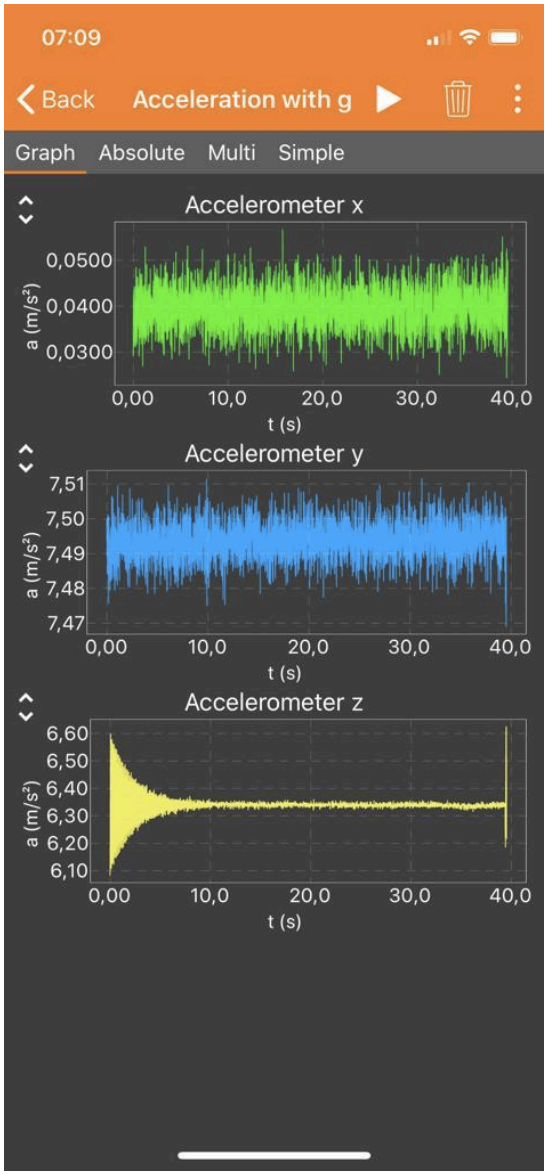
- Hence ($\dot{C}$ is the derivative wrt x)
  - $\dot{C} = -\varepsilon \frac{A}{x^2} \rightarrow \Delta C = -\varepsilon \frac{A}{x^2} \Delta x$
- An acceleration eventually modifies the capacity; the variation can be read as a current
  - $C = \frac{Q}{V} \rightarrow \dot{C} = \frac{1}{V}\dot{Q}$ (Q=electrical charge, V=potential difference)

Accelerometer

- This was a simplified model:
- Coulomb force is to be taken into the mechanical model into account
- Parallel Capacitors are used to make the C-x relationship linear



Example: touch the screen



removed via *calibration* (filtering)

vibrations from the environment

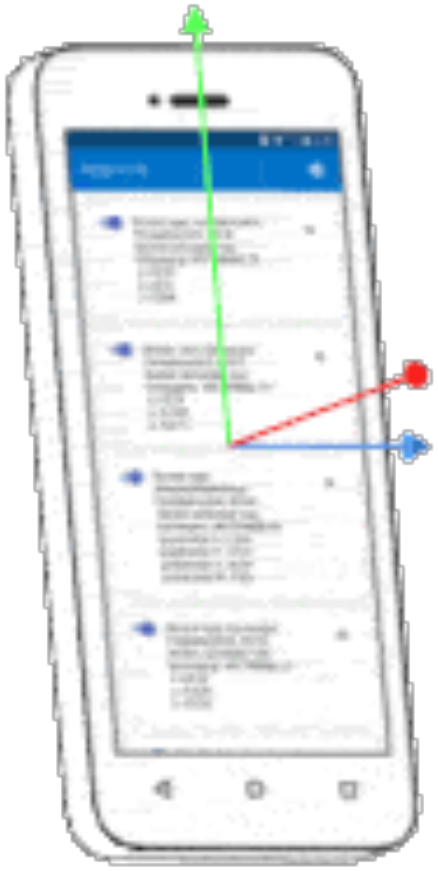
Vibrations due to screen touch (to start recording)

Gyroscope

X: 0.0 rad/s

Y: 0.0 rad/s

Z: 0.0 rad/s



Gyroscope sensor exploits Coriolis force

- This force is due to momentum conservation
- Suppose a proof mass is rotating and moves at constant radial speed v
- Since no real force is applied to the rotating mass, the angular momentum of the mass should remain the same
- This implies that if the mass moves outwards, v decreases (from v_1 to v_2), i.e. the mass sees an apparent force called **Coriolis force**

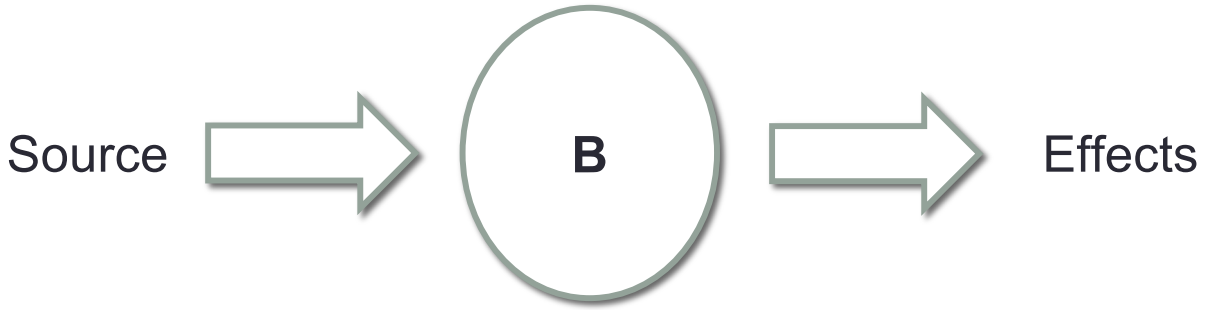
Gyroscope sensor

- Recall that the momentum of a moving mass is: $\mathbf{p}=m\mathbf{v}$
- The angular moment wrt to a point at distance r is: $\mathbf{L}=\mathbf{r} \times \mathbf{p}$
- The angular moment when rotating: $L=rmv=rmr\omega=m\omega r^2$
- The variation of its momentum, dL/dt, is always due to a torque $\tau=r\ F$
- Hence, if the mass is moving radially, $dL/dt = 2\ r\ m\omega\ dr/dt$
- $dL/dt = \tau \rightarrow 2r\ m\omega\ v_r = \tau = r\ F$
- $a_c = 2\ \omega\ v_r$
- When the smart phone rotates the force arises
- The force is proportional to the angular speed

Magnetism

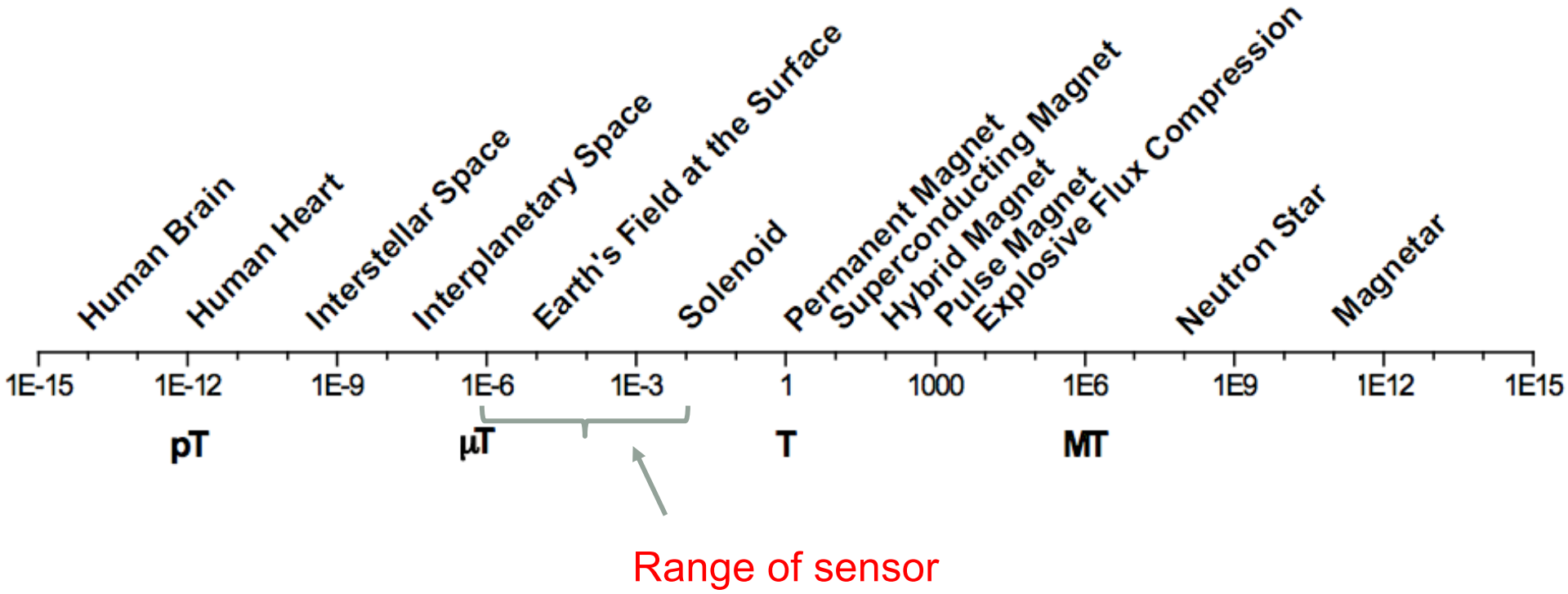
- Static Magnetic field sources:
  - Property of the matter
  - Charged particles moving at constant speed
  - [Source are always dipole]

- Effect of a magnetic field:
  - Force on single moving charged particles
  - Torque force on magnetic dipoles



Magnetic Field

Magnitude of magnetic fields



Effect of a magnetic field

- A magnetic field generates a force (Lorentz) on any moving charge*:

$$\boldsymbol{F} = q\boldsymbol{v} \times \boldsymbol{B}$$

- and a torque on any magnetic dipole, characterized by its *magnetic moment* **m**

$$\tau = \mathbf{m} \times \mathbf{B}$$

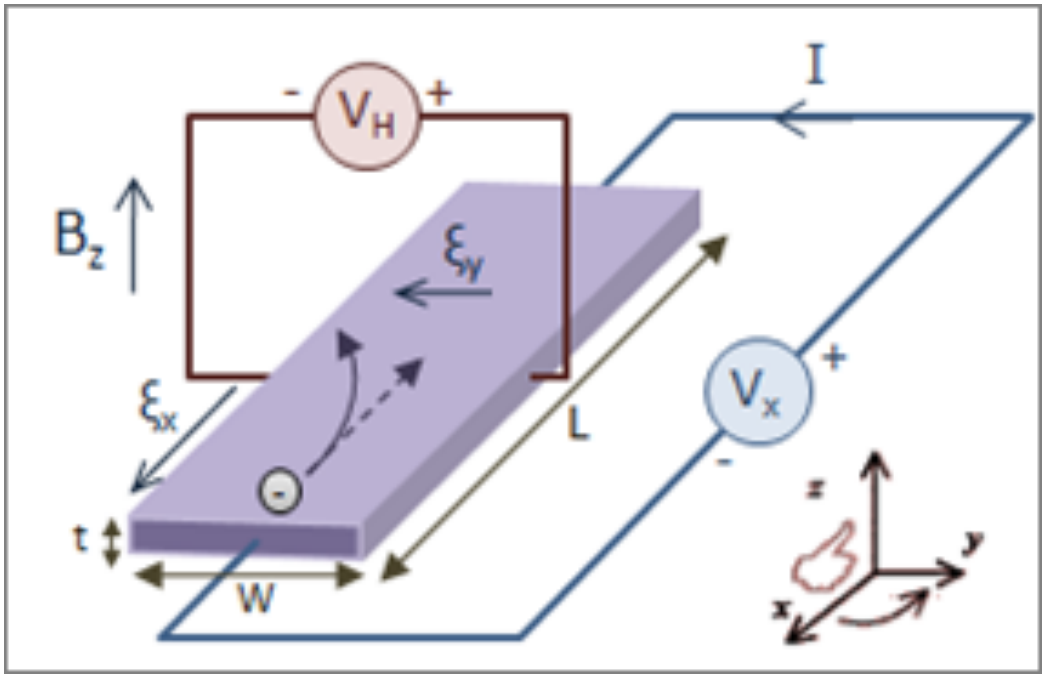
- The magnetic moment m depends on the shape of the object and its circulating current

- The torque orients a compass

*not relativistic formula

Magnetic sensor

- Can exploit the Lorentz Force (Hall effect).
- Hall effect is the trasduction principle to detect magnetic field

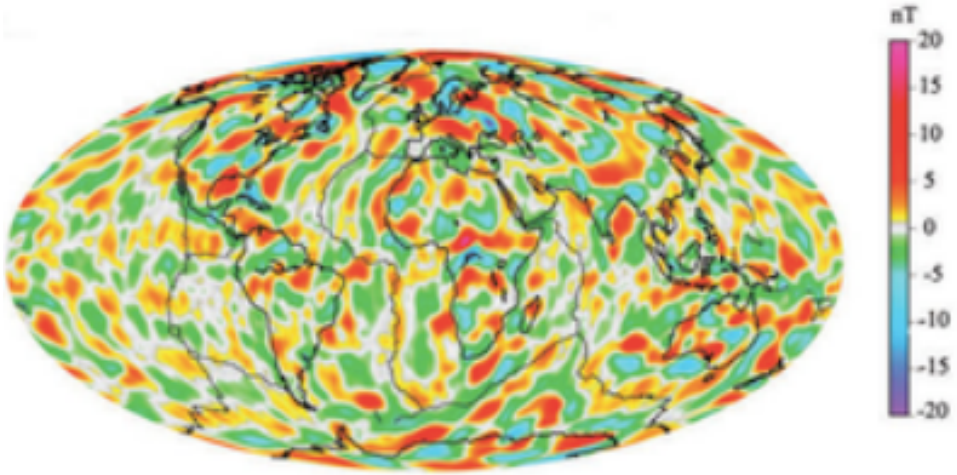


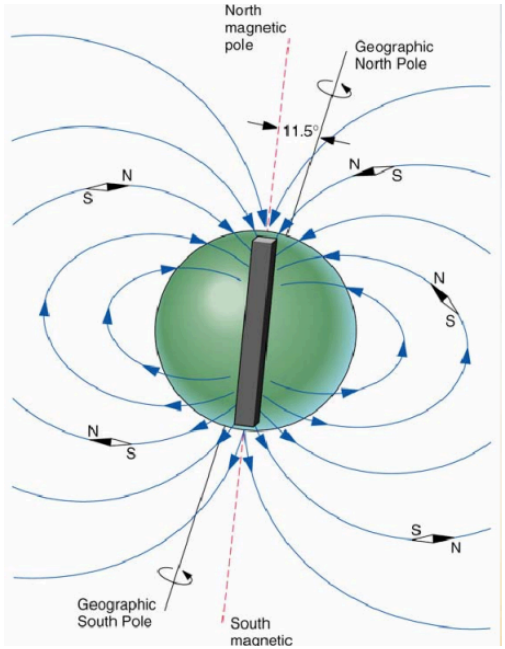
$$V_H = K B_z$$

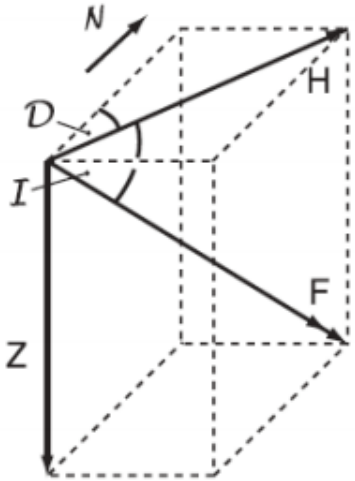
Earth’s magnetic field

- The Earth’s magnetic field can be modeled by a dipole and follows basic laws of magnetic fields.
- At any location, the Earth’s magnetic field can be represented by a three-dimensional vector and its intensity varies from 25μT to 65μT.
- The National Geospatial-Intelligence Agency (NGA) and the United Kingdoms Defence Geographic Centre (DGC) provide a World Magnetic Model (WMM)

Earth's magnetic field

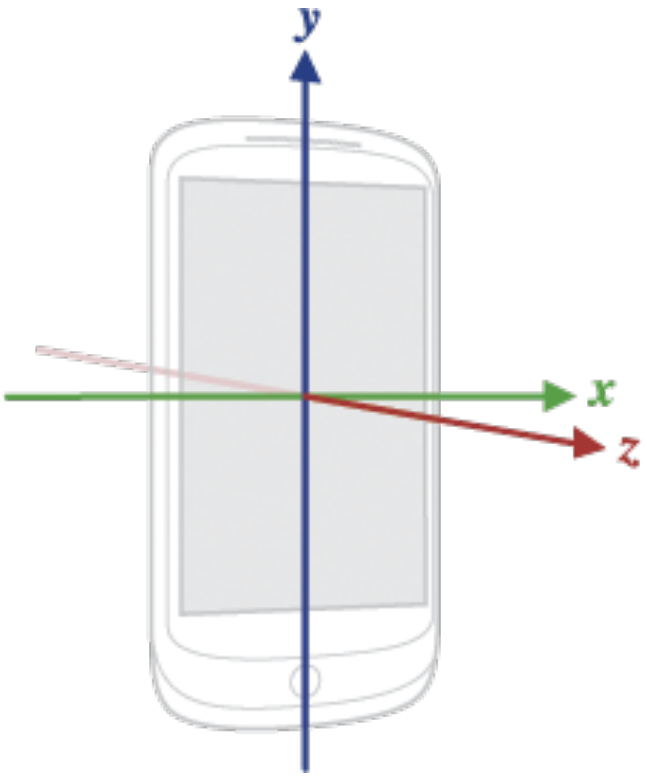


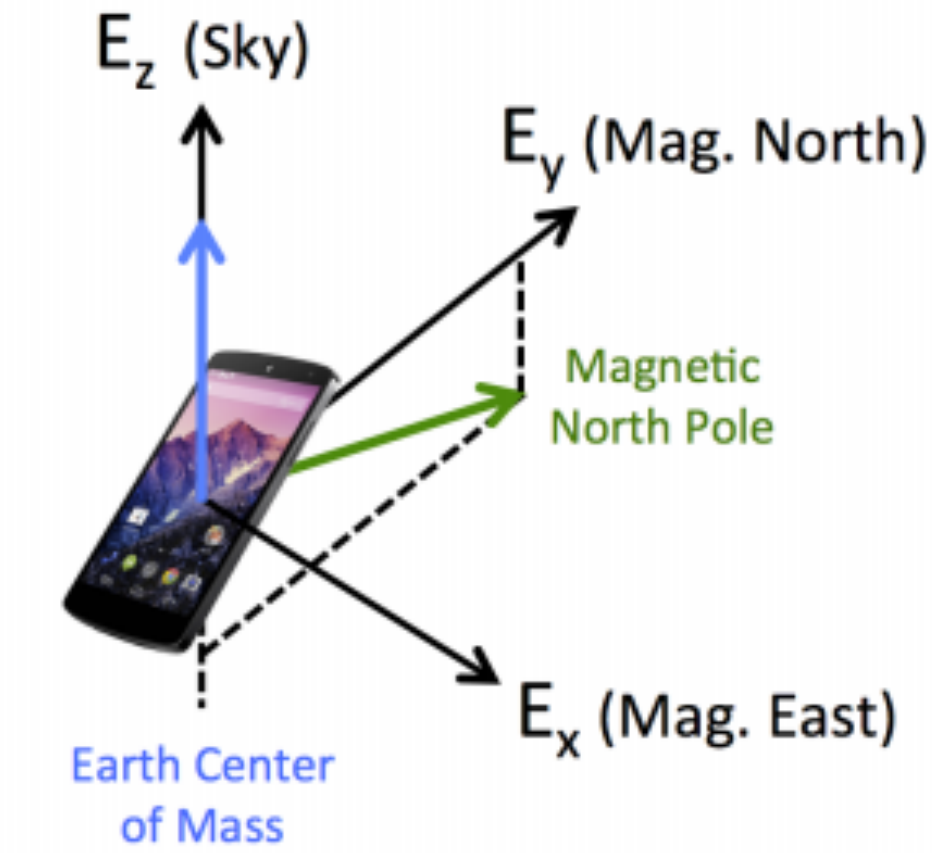




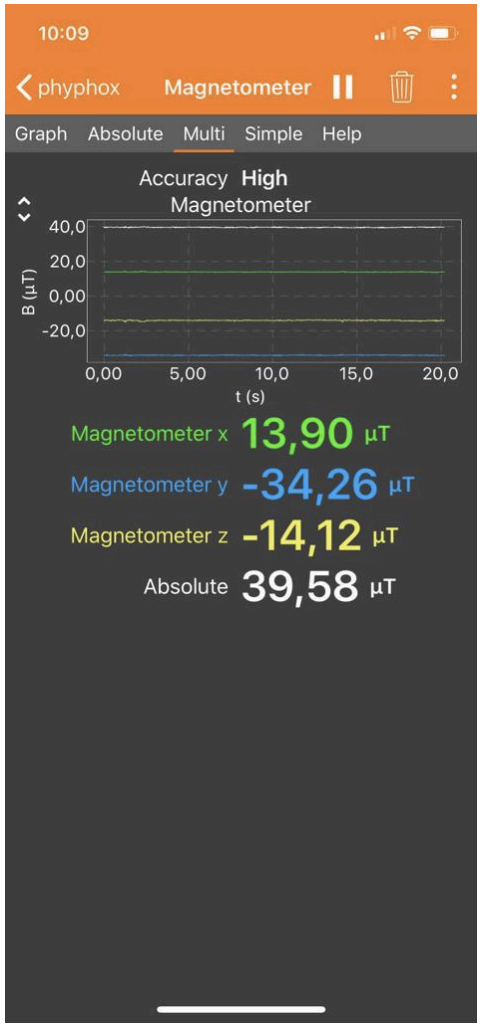
Horizontal and vertical components H, Z of the Earth's field, which has magnitude F, and direction defined by the declination (variation) D and inclination (dip) I.

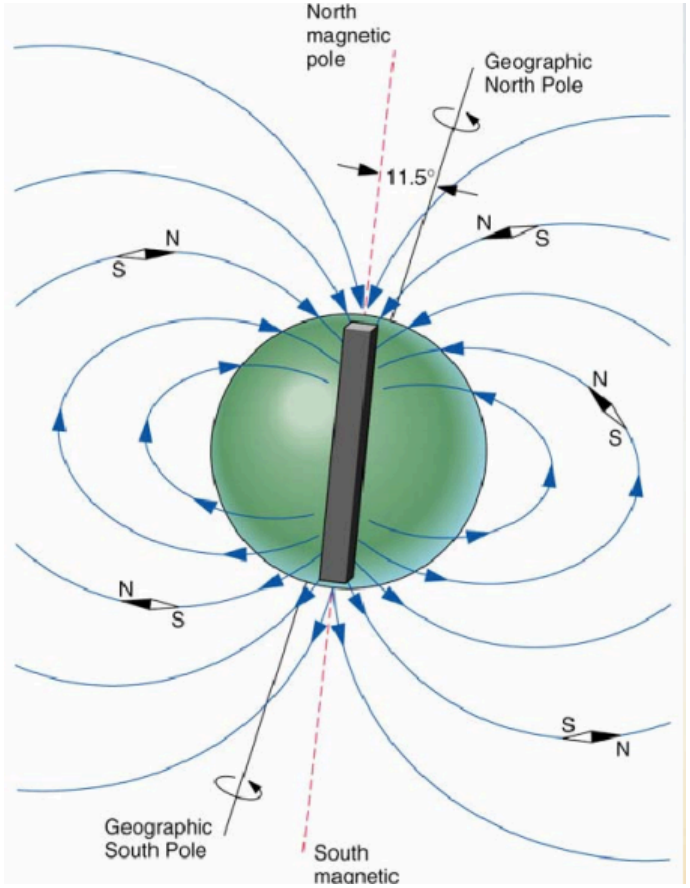
Reading with the smartphone

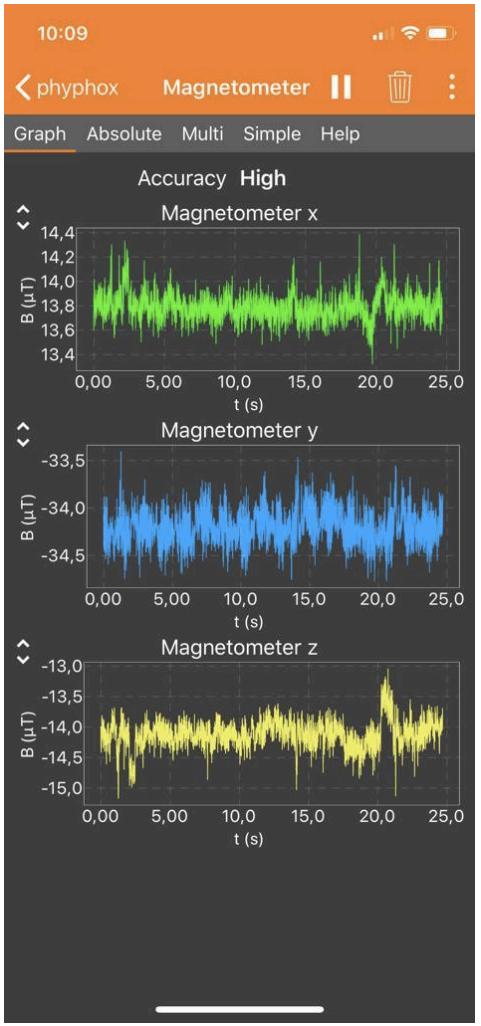




Example of magnetic field readings







Magnetic field reading*

- Unfortunately, the magnetometer in a smarthpne does not measure only the Earth's magnetic field.

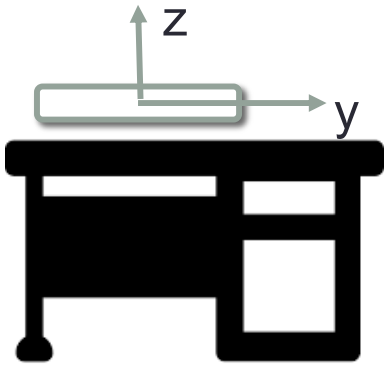
- Most of the time in indoor environments, we are in presence of magnetic dipoles which perturb the measure of Earth's magnetic field.
  - For example a refrigerator may produce up to 10^{-3} mT

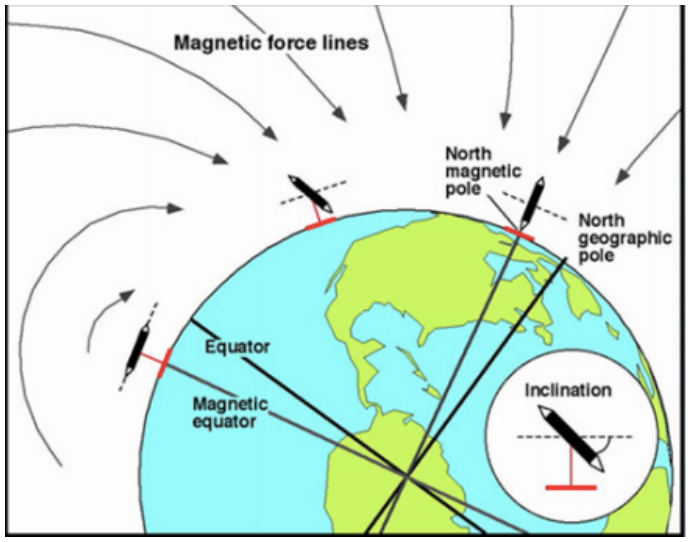
- These perturbations can be caused by electromagnetic devices (speakers, magnets), manmade structures (walls, floors) or other ferromagnetic objects like belts, keys, etc.

*Source: HAL Id: hal-01376745 https://hal.inria.fr/hal-01376745v1

Measuring magnetic earth's inclination

- Put the smartphone on a table that is parallel to the ground
- Orient the smartphone till $B_x=0$, so that you read $B=(0,B_y,B_z)$
- Inclination is *atan2*$(|B_z|,B_y)$





Example

Our department

https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml

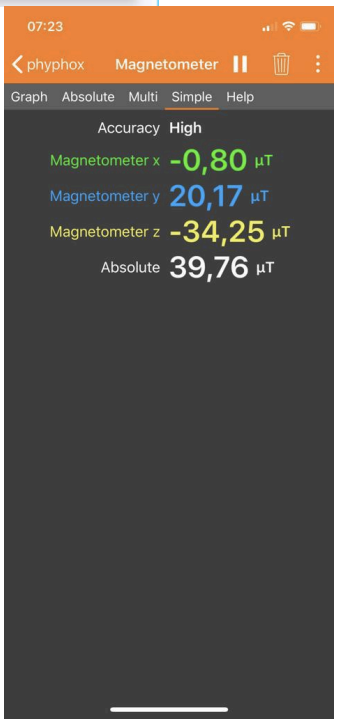
Magnetic Field

Model Used:	WMM2015V2
Latitude:	41° 53' 27" N
Longitude:	12° 30' 13" E
Elevation:	0.0 km Mean Sea Level

Date	Declination (+ E | - W)	Inclination (+ D | - U)	Horizontal Intensity	North Comp (+ N | - S)	East Comp (+ E | - W)	Vertical Comp (+ D | - U)	Total Field
2019-09-07	3° 20' 48"	58° 9' 7"	24,580.8 nT	24,538.9 nT	1,435.0 nT	39,570.7 nT	46,583.8 nT
Change/year	0° 8' 4"/yr	0° 0' 55"/yr	14.7 nT/yr	11.3 nT/yr	58.4 nT/yr	47.2 nT/yr	47.9 nT/yr
Uncertainty	0° 20'	0° 13'	133 nT	138 nT	89 nT	165 nT	152 nT

- $|B_z|$=39.570 μT
- B_y=24.538 μT

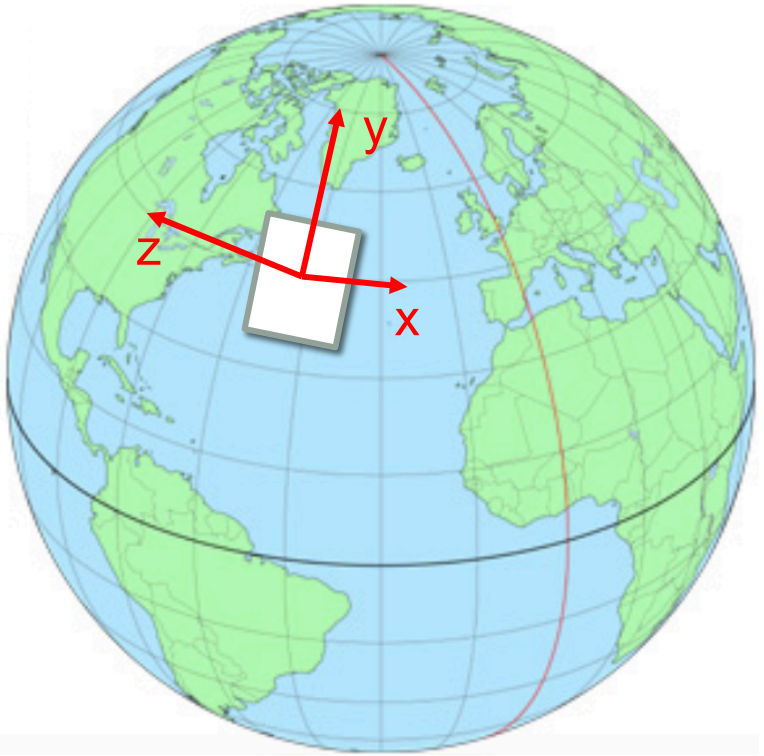
```
>>> math.atan2(39.5707,24.5389)*360/(2*math.pi)
58.1957889849151
```

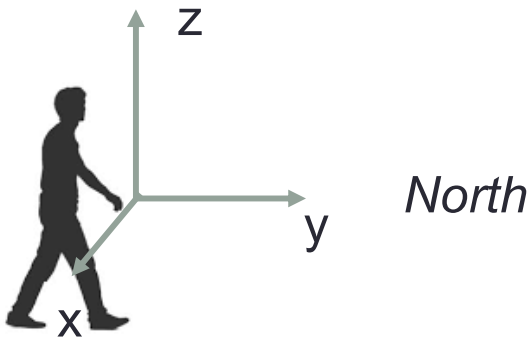


A real measurement

Earth's reference system

- *3D system*
- *z* extends up into space
- *y* points to magnetic north
- *x* is 90 degrees from *y*, pointing approximately east.



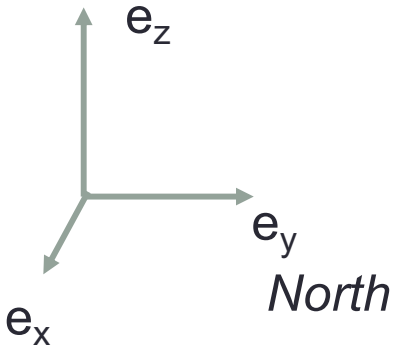


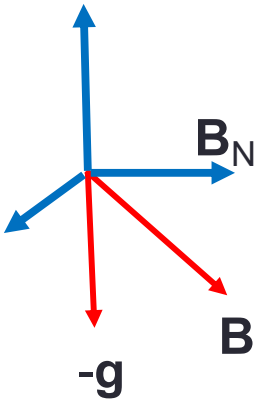
North

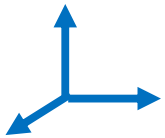
Earth's reference system

- How to 'define' the reference system?
- Exploit two vectors: gravity (**g**) and magnetic field (**B**)
- Vector **-g** points the center of the earth
- Vector **B** has a component pointing north (B_N) and the other the center of the earth

$\mathbf{e}_z \parallel \mathbf{g}$
$\mathbf{e}_y \parallel \mathbf{B}_N$
$\mathbf{e}_x = \mathbf{e}_y \times \mathbf{e}_z$





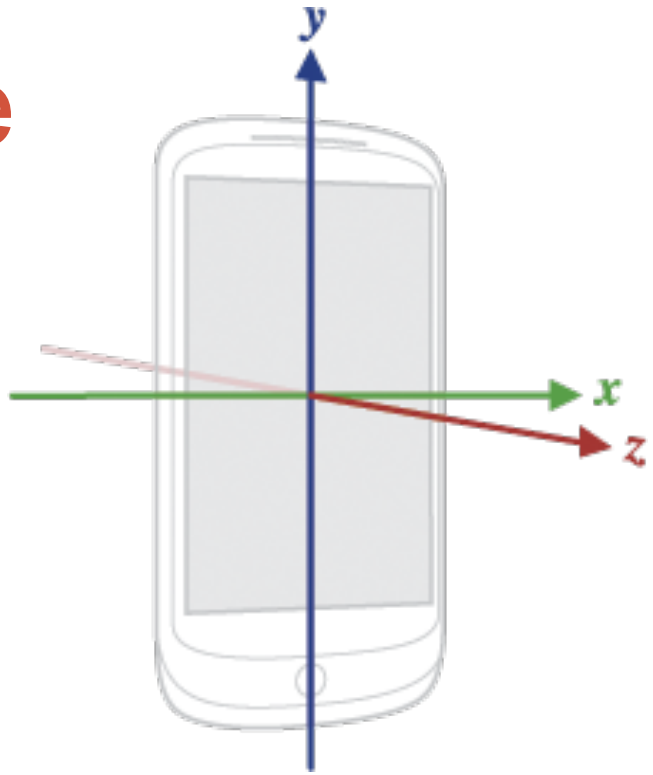


$$\mathbf{e}_z = \frac{\boldsymbol{g}}{||\boldsymbol{g}||}$$

$$\mathbf{e}_y = \frac{\boldsymbol{B}_N}{||\boldsymbol{B}_N||}$$

$$\mathbf{e}_x = \mathbf{e}_y \times \mathbf{e}_z$$

Device's reference frame



- Readings from sensors are measured using the device framework
- This coordinate system changes as the device rotates (MEMS are fixed into the device)

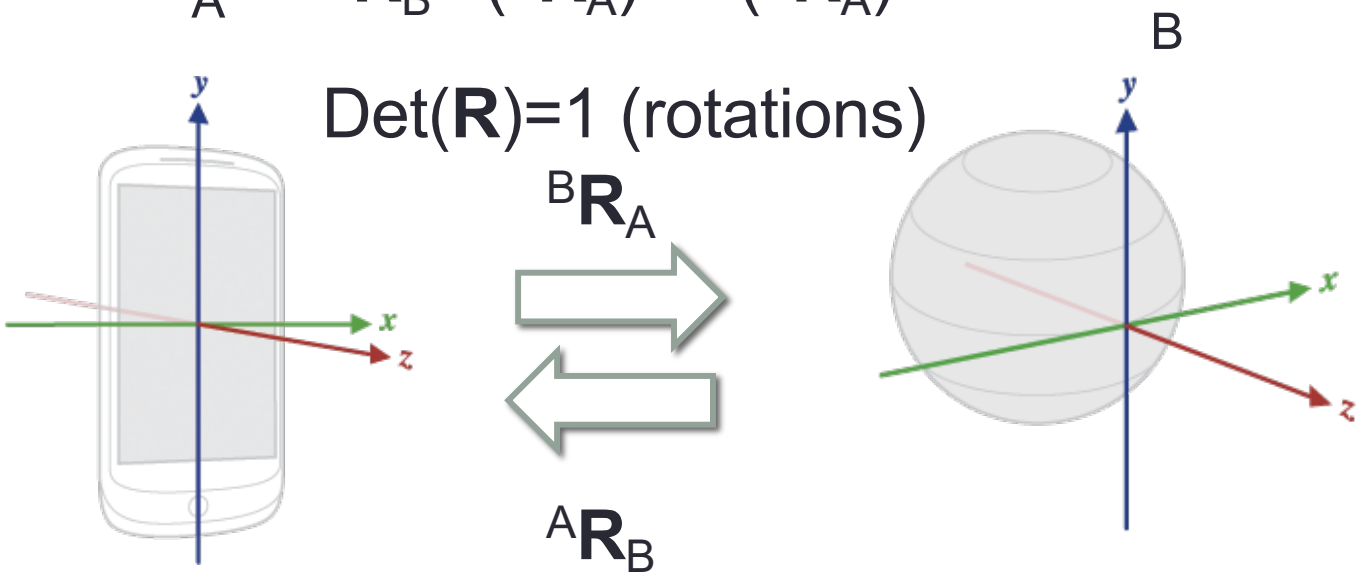
Rotation matrix

- We are given two reference systems A and B, and a vector whose coordinate $v=(x_A,y_A,z_A)$ are expressed in A. We want to calculate the coordinate $v' =(x_B,y_B,z_B)$ of the same vector expressed in B

- The 'rotation matrix' is a compact way to calculate the new coordinates: ${}^{B}\mathbf{R}_{A}$ (it changes the coordinate from A to B)

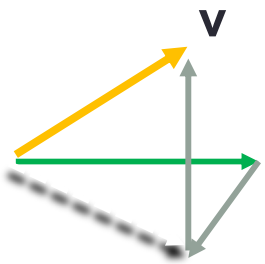
$${}^{A}\mathbf{R}_{B} =({}^{B}\mathbf{R}_{A})^{-1} =({}^{B}\mathbf{R}_{A})^{T}$$

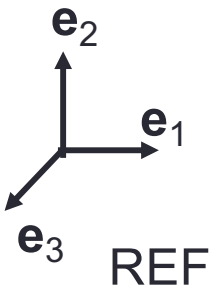
$$\text{Det}(\mathbf{R})=1 \text{ (rotations)}$$



Rotation matrix

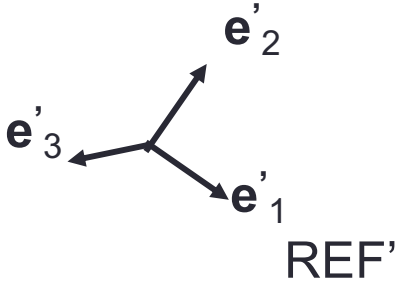
- $\mathbf{v} = v_x\,\mathbf{e}_1 + v_y\,\mathbf{e}_2 + v_z\,\mathbf{e}_3$

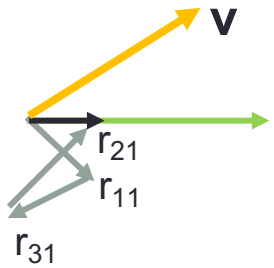




Coordinate of **v** in REF $\mathbf{v} \equiv (v_x, v_y, v_z)$

Coordinate of **v** in REF' $\mathbf{v} \equiv (v'_x, v'_y, v'_z)$





- $\mathbf{v} = v_x\,[r_{11}\mathbf{e}'_1 + r_{21}\mathbf{e}'_2\, r_{31}\mathbf{e}'_3] + v_y\,[r_{12}\mathbf{e}'_1 + r_{22}\mathbf{e}'_2\, r_{32}\,\mathbf{e}'_3] + v_Z\,[r_{13}\mathbf{e}'_1 + r_{23}\mathbf{e}'_2\, r_{33}\,\mathbf{e}'_3]$

($\mathbf{e}_1$, $\mathbf{e}_2$, $\mathbf{e}_3$ mark the three bracketed groups)

r_{ij}= i-th coordinate in REF' of the j-th component of REF

Rotation matrix

- $\mathbf{v} = v_x \overbrace{[r_{11}\mathbf{e}'_1 + r_{21}\mathbf{e}'_2\, r_{31}\mathbf{e}'_3]}^{\mathbf{e}_1} + v_y \overbrace{[r_{12}\mathbf{e}'_1 + r_{22}\mathbf{e}'_2\, r_{32}\,\mathbf{e}'_3]}^{\mathbf{e}_2} + v_Z \overbrace{[r_{13}\mathbf{e}'_1 + r_{23}\mathbf{e}'_2\, r_{33}\,\mathbf{e}'_3]}^{\mathbf{e}_3}$

- $\mathbf{v} = (v_x\, r_{11}+v_y\, r_{12} +v_z\, r_{13})\, \mathbf{e}'_1 + (v_x\, r_{21}+ v_y\, r_{22} +v_z\, r_{23})\, \mathbf{e}'_2 + (v_x\, r_{31}+v_y\, r_{32} +v_z\, r_{33})\, \mathbf{e}'_3$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \qquad \mathbf{v} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} \qquad \mathbf{v}' = \begin{bmatrix} v'_x \\ v'_y \\ v'_z \end{bmatrix}$$

coordinate in RIF'

$\mathbf{v}' = \mathbf{R}\,\mathbf{v}$

coordinate of in REF

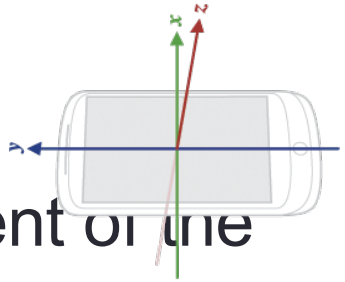
First meaning of the Rotation matrix

- The rotation matrix can be used to **change** the measures from the device framework to the fixed framework:

- Suppose that $\mathbf{v}_D$ is a reading (e.g., the magnetic field) in the Device's reference framework, then $\mathbf{v}_E$ (the same vector expressed in the earth's REF) is given by:

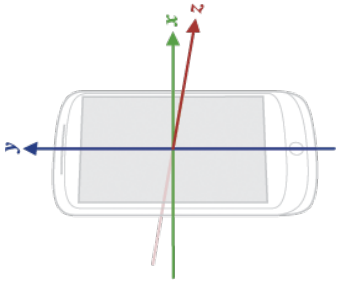
  - $\mathbf{v}_E = {}^{E}\mathbf{R}_D \, \mathbf{v}_D$

How to determine the Rotation matrix? The TRIAD* algorithm

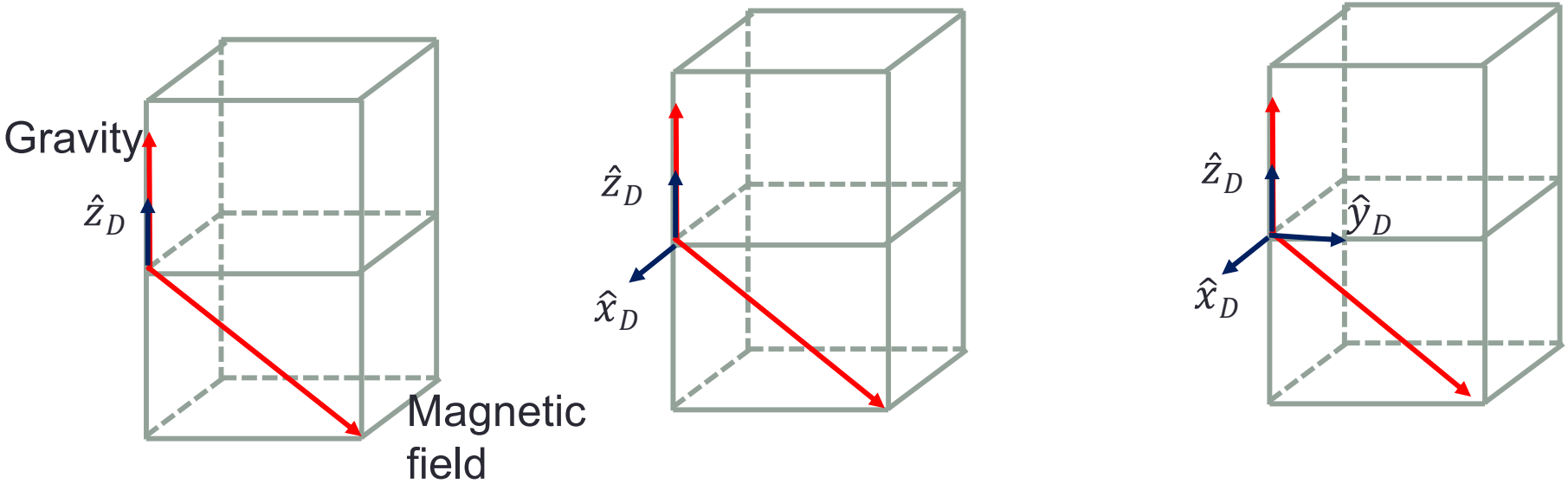


- Simple algebraic method to determine the coefficient of the matrix
- Idea:
- Define a three orthonormal unit vectors (a base) starting from the magnetic field and the gravity
- Map the base to base of the earth reference frame

How to determine the Rotation matrix? The TRIAD* algorithm



- $\hat{\boldsymbol{z}}_D = \frac{\mathbf{g}_D}{|\mathbf{g}_D|}, \hat{\boldsymbol{x}}_D = \frac{\mathbf{B}_D \times \mathbf{g}_D}{|\mathbf{B}_D \times \mathbf{g}_D|}, \hat{\boldsymbol{y}}_D = \hat{\boldsymbol{z}}_D \times \hat{\boldsymbol{x}}_D$



H. D. Black, “A passive system for determining the attitude of a satellite,” AIAA journal, vol. 2, no. 7, pp. 1350–1351, 1964.

How to determine the Rotation matrix? The TRIAD algorithm

- ${}^{E}\mathbf{R}_{D}$ is such that:

- $\hat{\boldsymbol{x}}_D$ is mapped to $\hat{\boldsymbol{x}}_E$ = $[1,0,0]^T$
- $\hat{\boldsymbol{y}}_D$ is mapped to $\hat{\boldsymbol{y}}_E$ = $[0,1,0]^T$
- $\hat{\boldsymbol{z}}_D$ is mapped to $\hat{\boldsymbol{z}}_E$ = $[0,0,1]^T$

- That is:

- ${}^{E}\mathbf{R}_{D}\ \hat{\boldsymbol{x}}_D = [1,0,0]^T$
- ${}^{E}\mathbf{R}_{D}\ \hat{\boldsymbol{y}}_D = [0,1,0]^T$
- ${}^{E}\mathbf{R}_{D}\ \hat{\boldsymbol{z}}_D = [0,0,1]^T$

How to determine the Rotation matrix? The TRIAD algorithm

- Let for convenience define the matrix

- $\mathbf{M}_D = [\,\hat{\boldsymbol{x}}_D \mid \hat{\boldsymbol{z}}_D \times \hat{\boldsymbol{x}}_D \mid \hat{\boldsymbol{z}}_D]$
- $\mathbf{I} = [(1,0,0) \mid (0,1,0) \mid (0,0,1)]$

- The previous equations are written as

- ${}^{E}\mathbf{R}_D\, \mathbf{M}_D = \mathbf{I}$

- So that ${}^{E}\mathbf{R}_D = (\mathbf{M}_D)^{-1}$
- Since $\mathbf{M}_D$ is an orthonormal matrix, its inverse is equal to its transpose

$$ {}^{E}\mathbf{R}_D = (\mathbf{M}_D)^{T} $$

Experiment 1: Device aligned with the earth

Experiment 2: Device pointing north, tilted. Compute tilt angle and Inclination

Experiment 2bis: Device pointing north, tilted. Compute the rotation matrix

The TRIAD algorithm in python

```
import numpy as np

#Readings from sensors
G=np.array([0.045217514, 0.008384705, 9.843344])
B=np.array([0.18692017, 18.654633, -35.813904])


z=G/np.sqrt(G.dot(G)) #Normalize g
x=np.cross(B,G) # Find a vector pointing towards x
x=x/np.sqrt(x.dot(x)) #Normalize
y=np.cross(z,x) #Find y
R=np.array([x,y,z]) #Rotation matrix

print 'G:',G,'\nB:',B
print 'ROTATION MATRIX:\n',R
```

A real numeric example

- Input (readings from sensors):
- B=[0.1869 18.654 -35.813]
- G=[0.045217 0.008384 9.843] →(|G|=9.8434)

- `z=G/np.sqrt(G.dot(G)) #Normalize g`
- `#z = G/|G| = [0.9998  -0.0188 -0.00457]`

- `x=np.cross(B,G) # Find a vector pointing towards x`
- `#x=[183.92425883  -3.45933524  -0.84194886]`
- `x=x/np.sqrt(x.dot(x)) #Normalize`
- `#0.9998127  -0.01880495 -0.00457684`

- `y=np.cross(z,x) #Find y`
- `#[1.88008505e-02  9.99822808e-01  -9.38029497e-04]`

A real numeric example: Rotation Matrix

$$\begin{bmatrix} 0.99981266 & -0.018804953 & -0.004576836 \\ 0.018800847 & 0.99982274 & -9.3802944E\text{-}4 \\ 0.0045936643 & 8.518053E\text{-}4 & 0.99998903 \end{bmatrix}$$

Browsing the source code

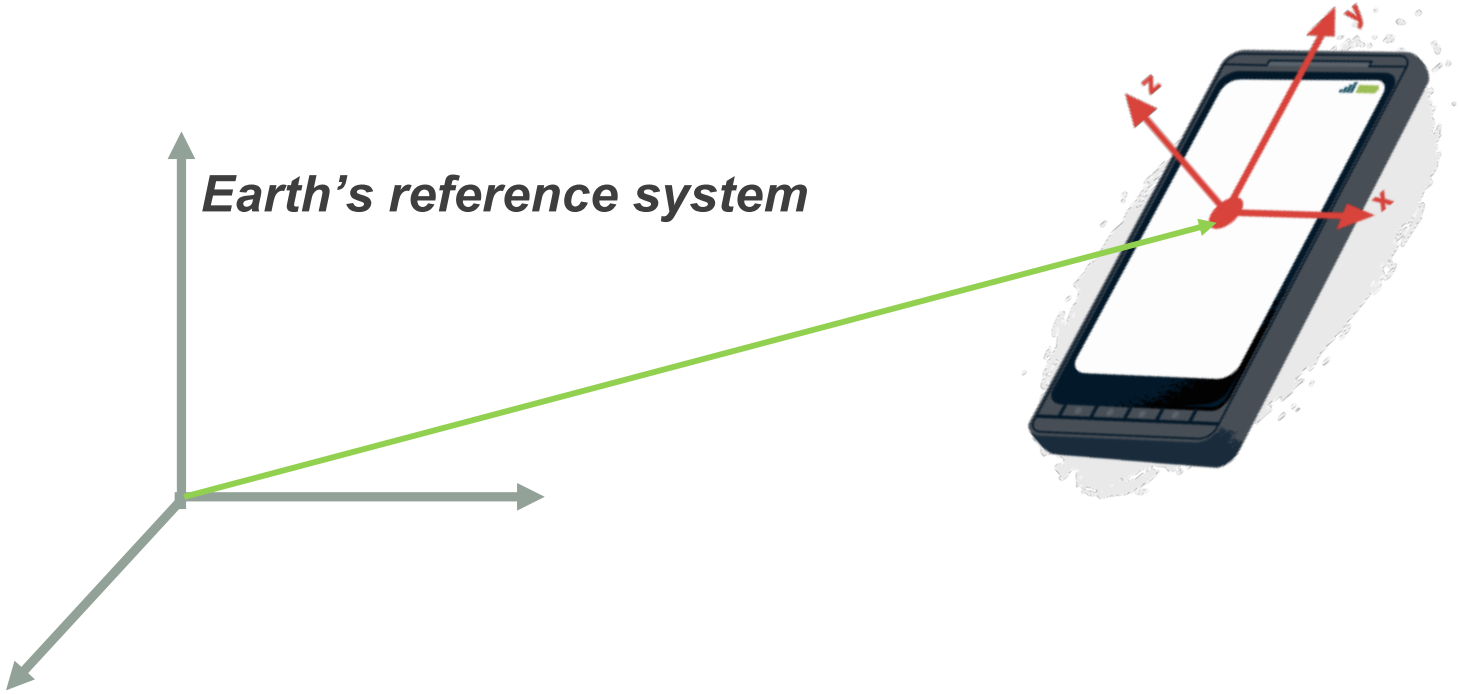
```
public static boolean getRotationMatrix(float[] R, float[] I,
        float[] gravity, float[] geomagnetic) {
    // TODO: move this to native code for efficiency
    float Ax = gravity[0];
    float Ay = gravity[1];
    float Az = gravity[2];

    final float normsqA = (Ax * Ax + Ay * Ay + Az * Az);
    final float g = 9.81f;
    final float freeFallGravitySquared = 0.01f * g * g;
    if (normsqA < freeFallGravitySquared) {
        // gravity less than 10% of normal value
        return false;
    }

    final float Ex = geomagnetic[0];
    final float Ey = geomagnetic[1];
    final float Ez = geomagnetic[2];
    float Hx = Ey * Az - Ez * Ay;
    float Hy = Ez * Ax - Ex * Az;
    float Hz = Ex * Ay - Ey * Ax;
    final float normH = (float) Math.sqrt(Hx * Hx + Hy * Hy + Hz * Hz);

    if (normH < 0.1f) {
        // device is close to free fall (or in space?), or close to
        // magnetic north pole. Typical values are  > 100.
        return false;
    }
    final float invH = 1.0f / normH;
    Hx *= invH;
    Hy *= invH;
    Hz *= invH;
    final float invA = 1.0f / (float) Math.sqrt(Ax * Ax + Ay * Ay + Az * Az);
    Ax *= invA;
    Ay *= invA;
    Az *= invA;
    final float Mx = Ay * Hz - Az * Hy;
    final float My = Az * Hx - Ax * Hz;
    final float Mz = Ax * Hy - Ay * Hx;
    if (R != null) {
        if (R.length == 9) {
            R[0] = Hx;     R[1] = Hy;     R[2] = Hz;
            R[3] = Mx;     R[4] = My;     R[5] = Mz;
            R[6] = Ax;     R[7] = Ay;     R[8] = Az;
        } else if (R.length == 16) {
            R[0]  = Hx;    R[1]  = Hy;    R[2]  = Hz;   R[3]  = 0;
            R[4]  = Mx;    R[5]  = My;    R[6]  = Mz;   R[7]  = 0;
            R[8]  = Ax;    R[9]  = Ay;    R[10] = Az;   R[11] = 0;
            R[12] = 0;     R[13] = 0;     R[14] = 0;    R[15] = 1;
        }
    }
```

Orientation of the mobile phone



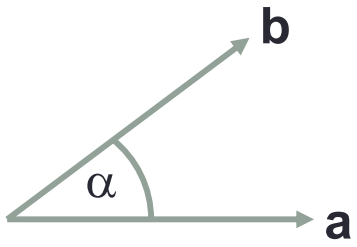
Given a given tilted smartphone, the orientation says how to rotate the smartphone so that starting from the position aligned to the earth, it will reach that tilted configutarion

How to express the orientation?

- Direction cosine → Rotation Matrix
- Euler angles
- Quaternion

Angle among two vectors

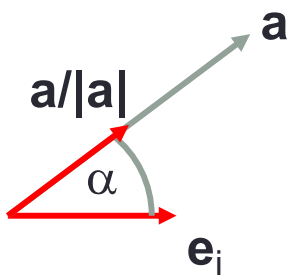
- The (cosine of) the angle among two vectors, **a b** is by definition



$$\cos\alpha = \frac{a \cdot b}{|a||b|}$$

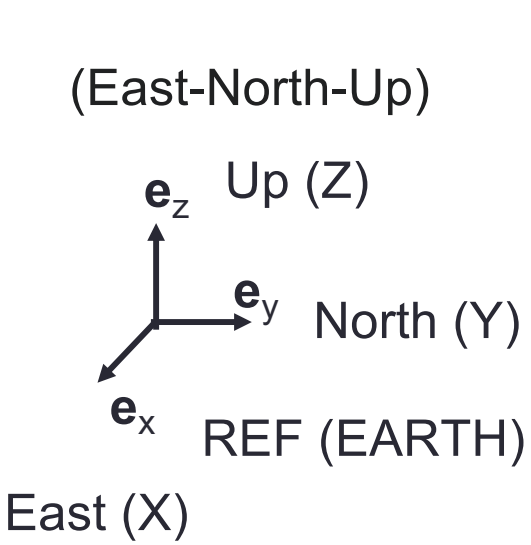
Direction cosine of a vector wrt a reference frame

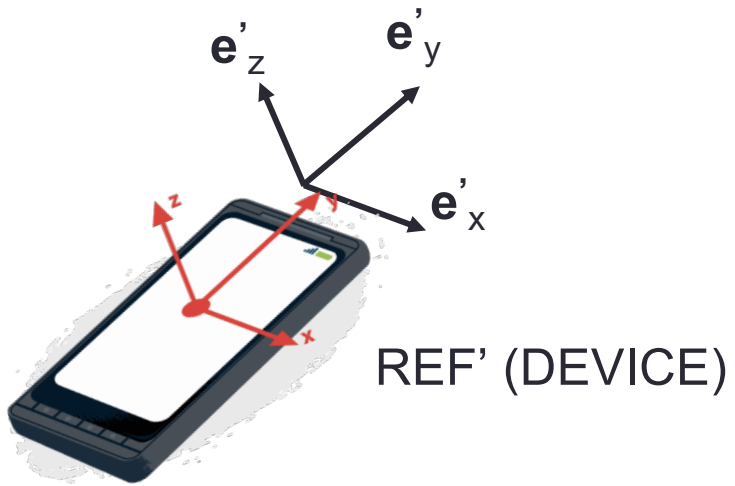
- The direction cosine of a vector **a** wrt to a unit vector $\mathbf{e}_i$ of the base of reference system, is the cosine of the angle of the **a**/|**a**| wrt to $\mathbf{e}_i$
- The cosine basically corresponds to the projection of **a**/|**a**| on $\mathbf{e}_i$, and hence it is the coordinate wrt to $\mathbf{e}_i$



Direction cosines of the smartphone

- They are the direction cosines of the basis of the device reference system, wrt to reference frame of the earth





Orientation and rotation matrix

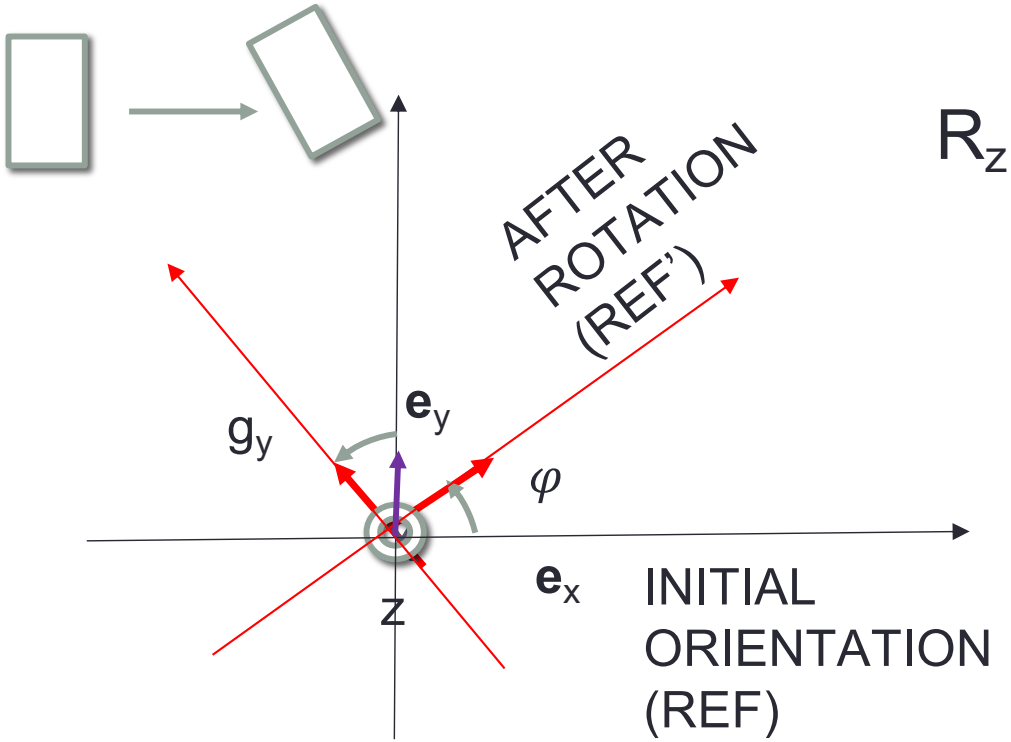
- The rotation matrix contains all information to determine the orientation
- To get the coordinate of $\mathbf{e}'_i$ wrt $\{\mathbf{e}_x,\mathbf{e}_y,\mathbf{e}_z\}$, we have to see how it is mapped to the earth-frame

$$\begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} r_{11} \\ r_{21} \\ r_{31} \end{bmatrix} \quad \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} r_{12} \\ r_{22} \\ r_{32} \end{bmatrix} \quad \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} r_{13} \\ r_{23} \\ r_{33} \end{bmatrix}$$

- The **columns** of ${}^{E}\mathbf{R}_{D}$ are the direction cosines of the device reference frame

Pure rotation around z

- To define a rotation, we will again use a rotation matrix
- The device is a ‘rigid body’
- This matrix is used to calculate the coordinates of device reference frame after the rotation (given in the initial reference frame)



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

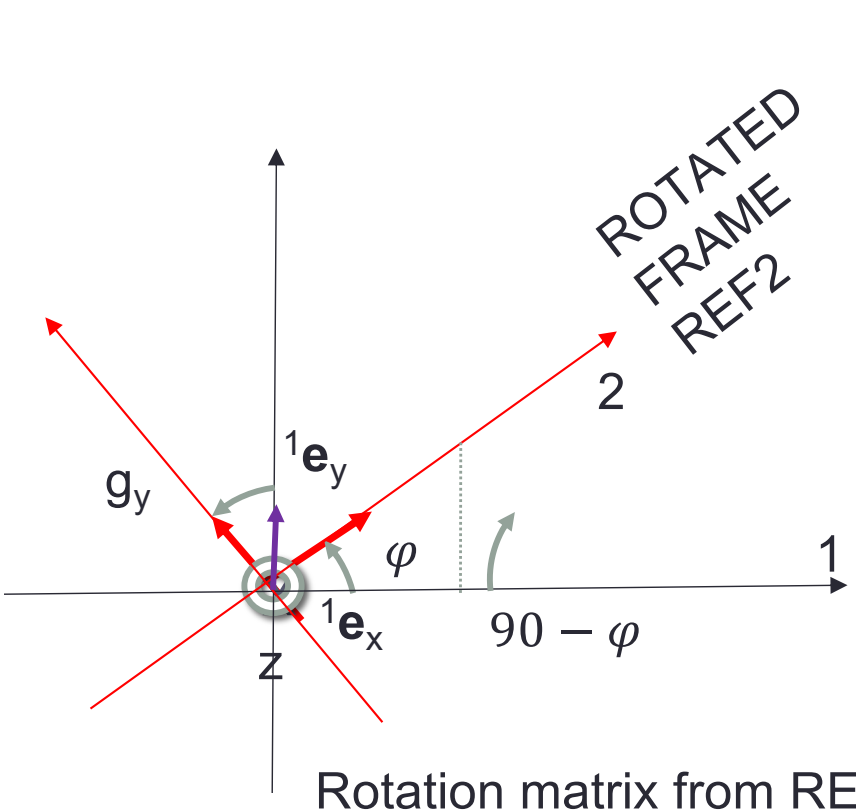
Coordinate of $\mathbf{e}_x$ after the rotation (Direction cosine)

Coordinate of $\mathbf{e}_y$ after the rotation

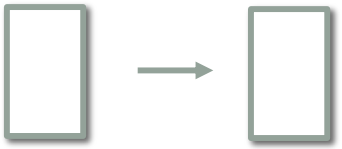
NB: In this example we are assuming that counter-clockwise rotations are positive

Alternative interpretation

- The device is still, but its reference frame rotates
- We want to know how the coordinates change in this new frame



INITIAL
FRAME, REF1



Coordinates of $\mathbf{e}_x$ in REF1

$^2R_1\ ^1\mathbf{e}_x = {}^2\mathbf{e}_x$

Coordinates of $^2\mathbf{e}_x$ in REF2

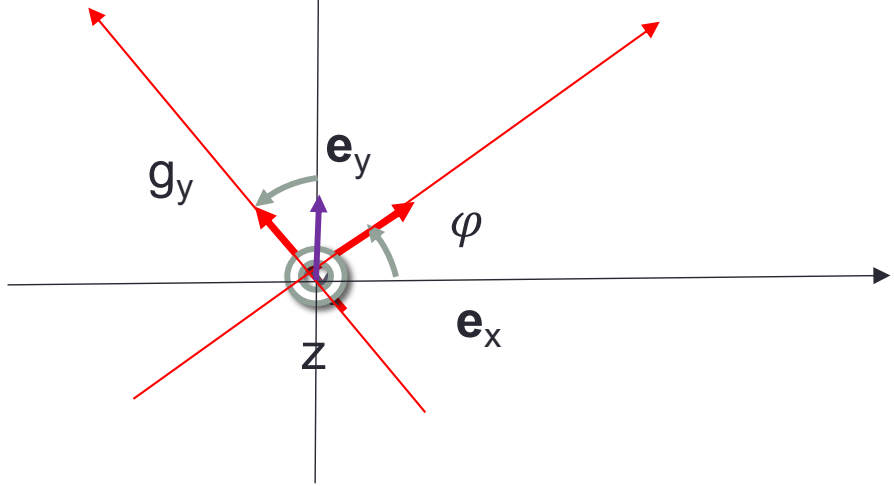
Rotation matrix from REF1 to REF2:
Used to calculate how the coordinates of a vector expressed in REF1
Change when expressed in REF2

Alternative interpretation

- ${}^{2}R_{1}\ {}^{1}\mathbf{e}_x = {}^{2}\mathbf{e}_x$
- ${}^{1}R_{2}\ {}^{2}\mathbf{e}_x = {}^{1}\mathbf{e}_x$
- ${}^{1}R_{2}\ {}^{2}R_{1}\ {}^{1}\mathbf{e}_x = {}^{1}\mathbf{e}_x \rightarrow {}^{1}R_{2}\ {}^{2}R_{1} = I \rightarrow {}^{2}R_{1} = ({}^{1}R_{2})^{-T}$

Because the matrix is orthogonal
the inverse and transpose are equal

R_z maps (1,0,0) to (cos(φ),sin(φ),0)
→ ${}^{1}R_{2} = R_Z$



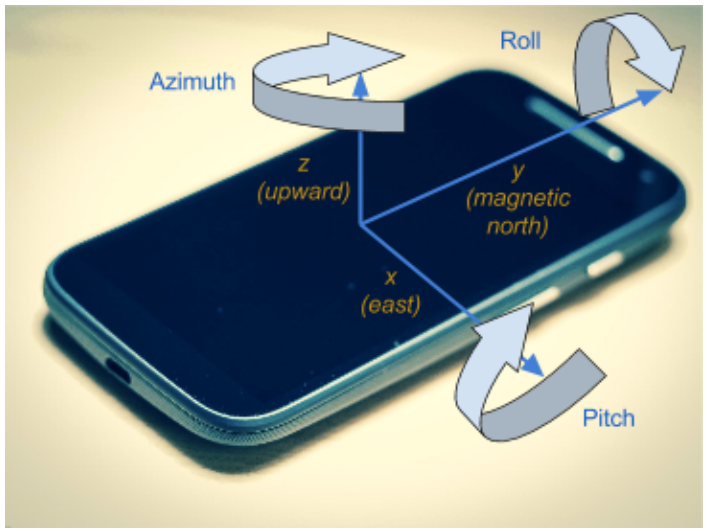
$$\mathbf{{}^{2}R_{1} = (R_Z)^{-1} = (R_Z)^{-T}}$$

Rotation matrix

- R_z describes a physical rotation of the device around z
- $(R_z)^T$ describes the frame rotation around z

Elemental rotations

- R_z → Yaw (Azimuth): degrees of rotation about the z axis
- R_x → Pitch: degrees of rotation about the x axis
- R_y → Roll: degrees of rotation about the y axis



Given an orientation, we want to know how to rotate the smartphone so that starting from the position aligned to the earth, that position is reached…

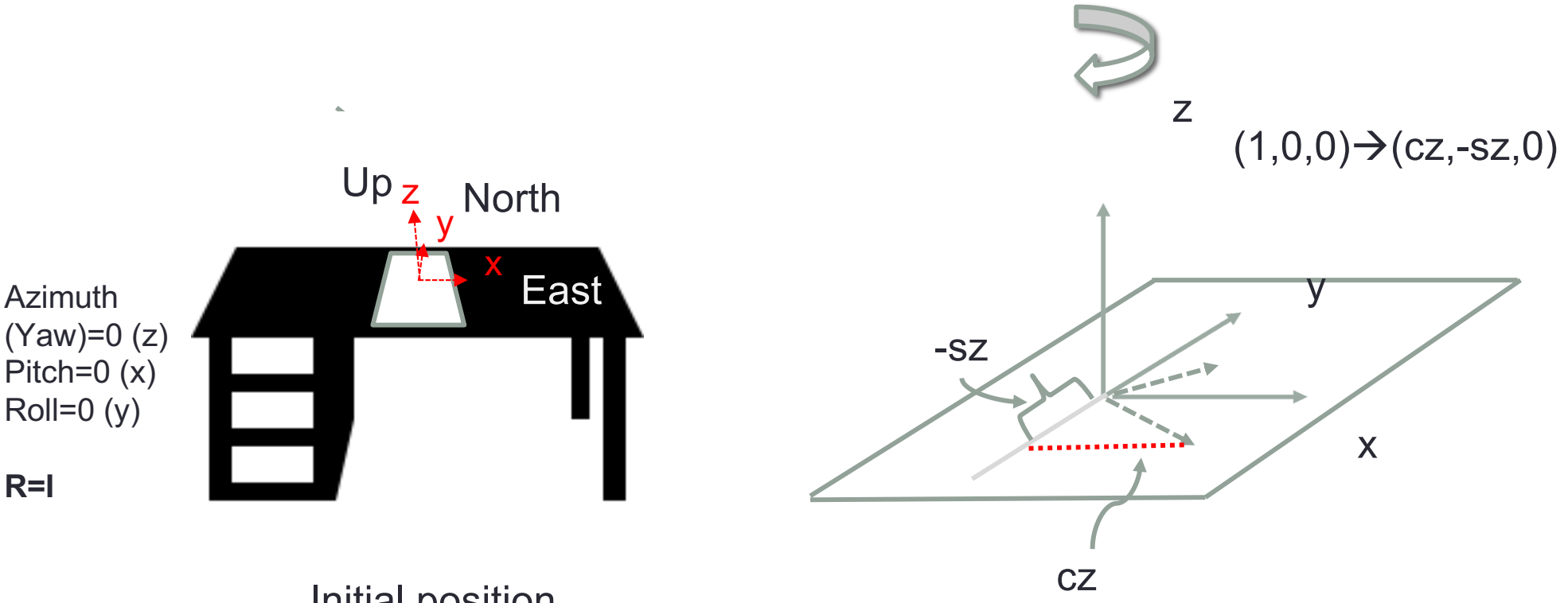
"observer" looking at the head of the arrow

Elemental rotation around z (yaw)

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

$\sin(-\alpha) = -\sin(\alpha)$

Android convention:
clockwise rotations around the z axis are positive, rotation towards east is positive



Initial position

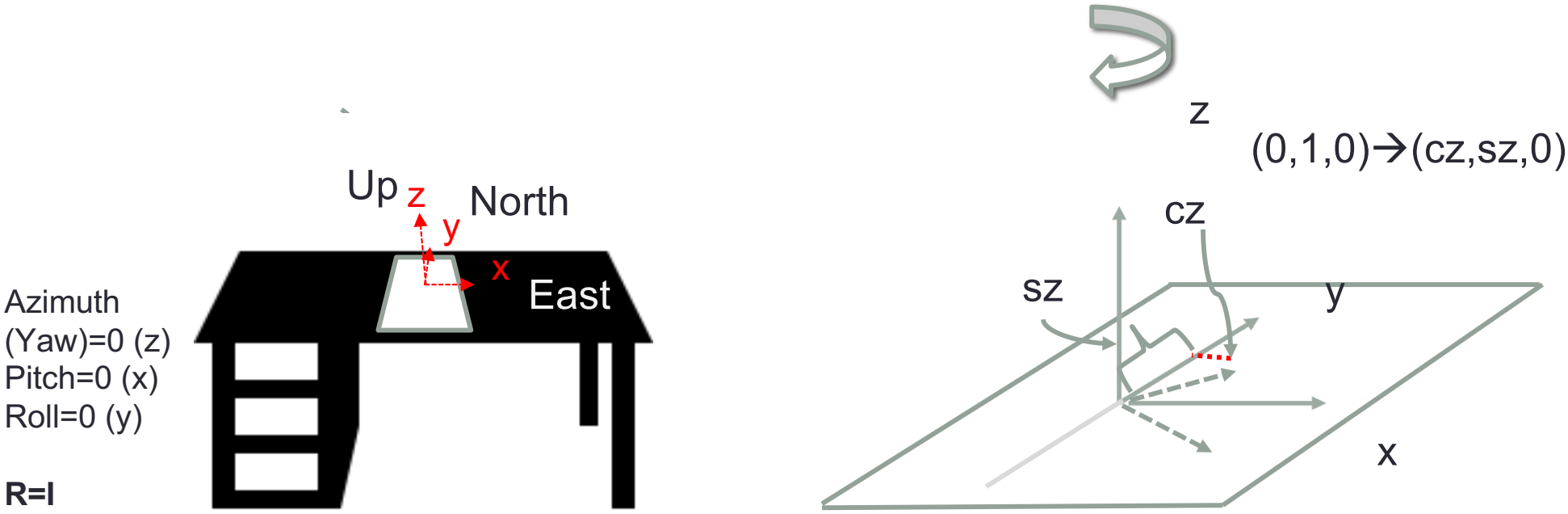
"observer" looking at the head of the arrow

Elemental rotation around z (yaw)

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

$\sin(-\alpha) = -\sin(\alpha)$

Android convention:
clockwise rotations around the z axis are positive, rotation towards east is positive



Azimuth (Yaw)=0 (z)
Pitch=0 (x)
Roll=0 (y)

R=I

Initial position

"observer" looking at the head of the arrow

Elemental rotation around z (yaw)

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

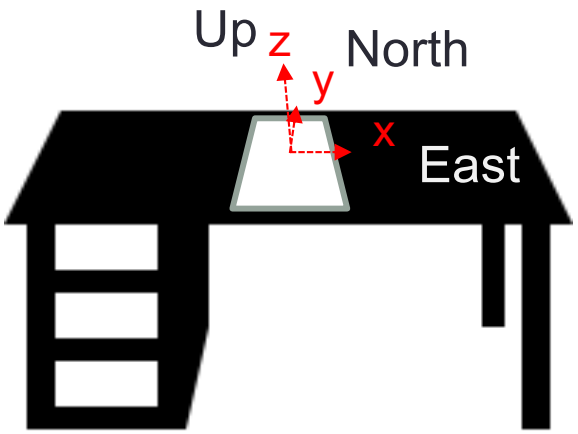
$\sin(-\alpha) = -\sin(\alpha)$

Android convention:
clockwise rotations around the z axis are positive, rotation towards east is positive

$$R_z = \begin{bmatrix} c_z & s_z & 0 \\ -s_z & c_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Azimuth (Yaw)=0 (z)
Pitch=0 (x)
Roll=0 (y)

R=I



Initial position

"observer" looking at the head of the arrow

Elemental rotation around z (yaw)

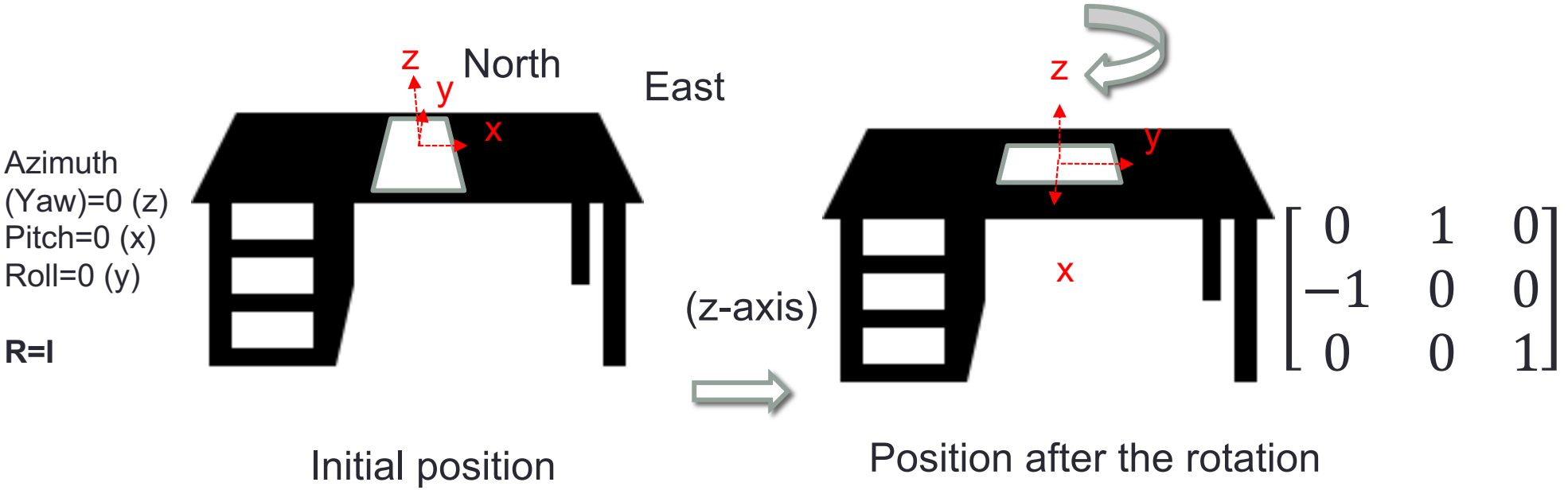
$c_z = \cos(\phi_z)$
$s_z = \sin(\phi_z)$

$\sin(-\alpha) = -\sin(\alpha)$

Android convention:
clockwise rotations around the z axis are positive, rotation towards east is positive

$$R_z = \begin{bmatrix} c_z & s_z & 0 \\ -s_z & c_z & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

This is a positive rotation of 90 degree (y moved towards east)



Initial position

Position after the rotation

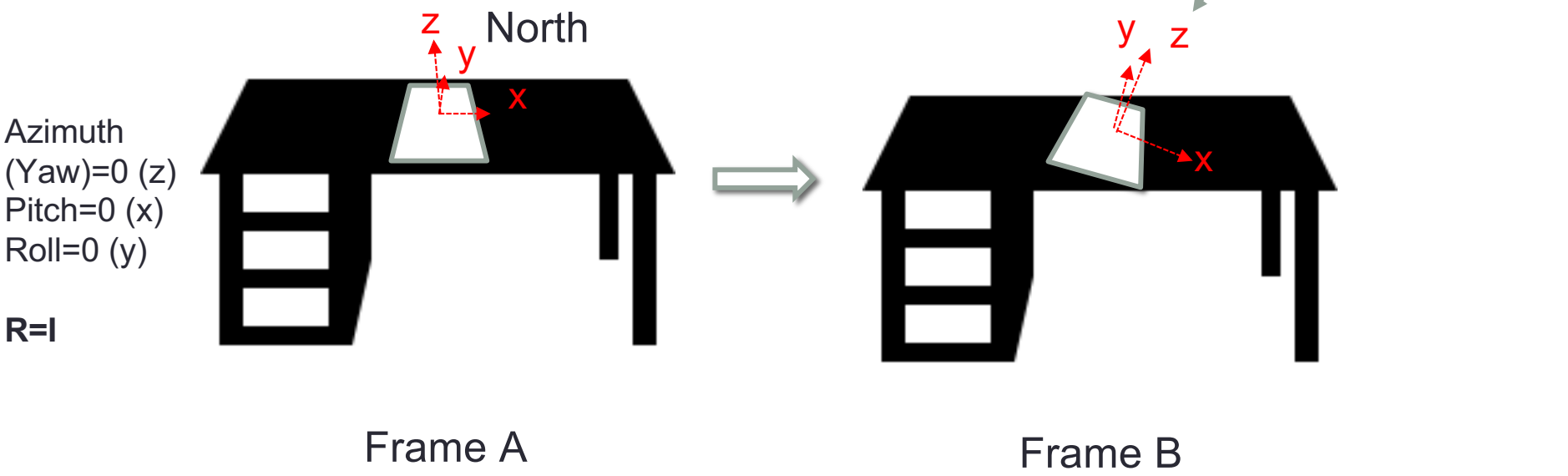
Elemental rotation around y (roll)

$c_y = \cos(\phi_y)$
$s_y = \sin(\phi_y)$

Android convention

$$R_y = \begin{bmatrix} c_y & 0 & s_y \\ 0 & 1 & 0 \\ -s_y & 0 & c_y \end{bmatrix}$$

This is a positive rotation



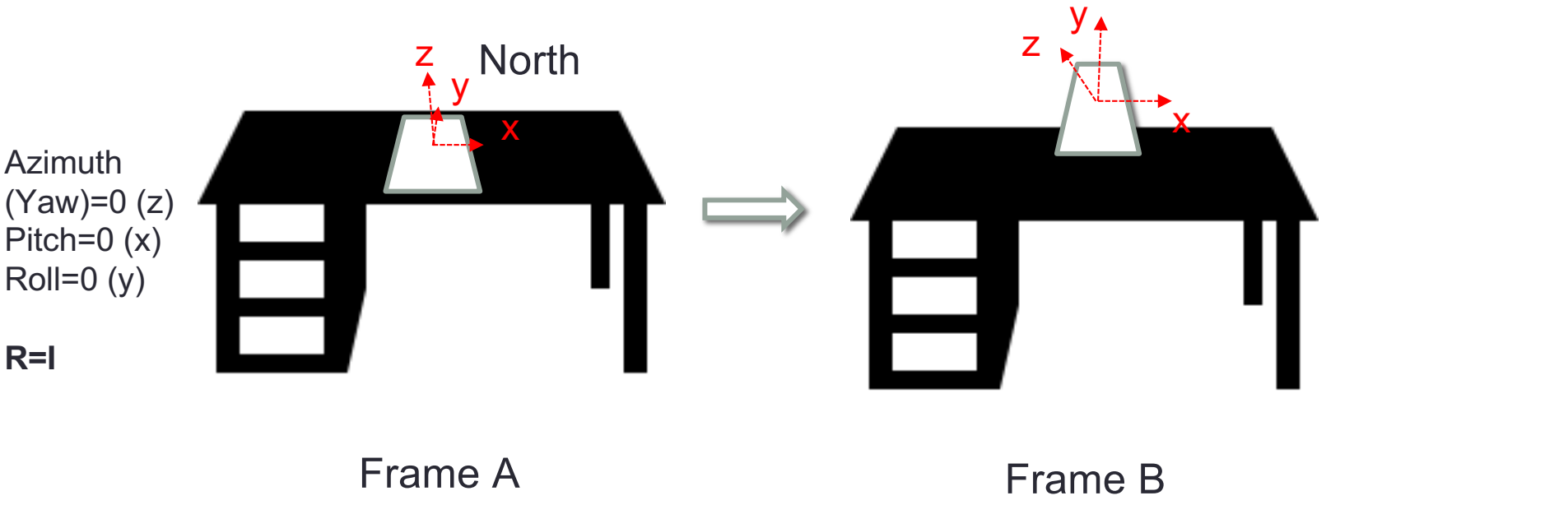
Elemental rotation around x (pitch)

$c_x = \cos(\phi_x)$
$s_x = \sin(\phi_x)$

Android convention

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_x & s_x \\ 0 & -s_x & c_x \end{bmatrix}$$

This is a negative rotation
(top edge moved up)



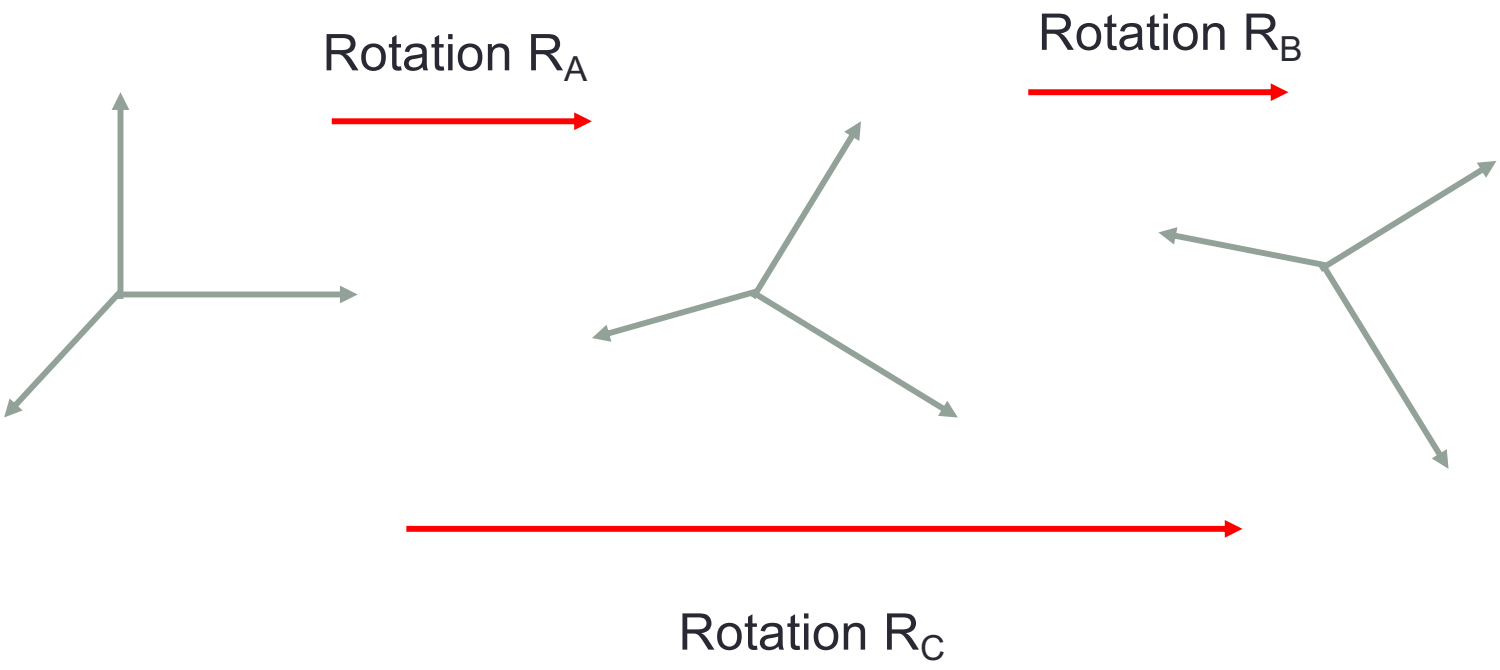
Rotation composition

- Rotations can be applied in sequence
- The order of rotations affects the final position

- Extrinsic rotations: rotations around the fixed frame
  - Multiplication follows the same order of rotation sequence
  - First R_Z and then R_Y → $R = R_Y R_Z$
  - (R_Z is pre-multiplied by R_Y)

- Intrinsic rotations: rotations around the rotated frame
  - Multiplication follows the the inverse order of rotation sequence
  - First R_Z and then R_Y → $R = R_Z R_Y$
  - (R_Z is post-multiplied by R_Y)

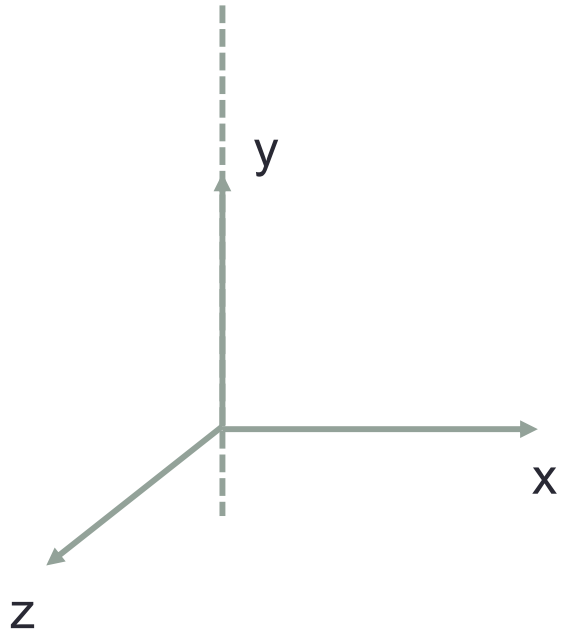
Rotation composition

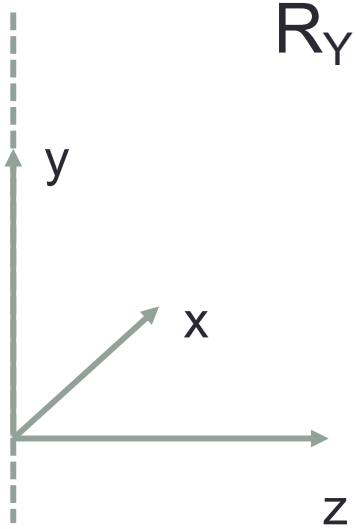


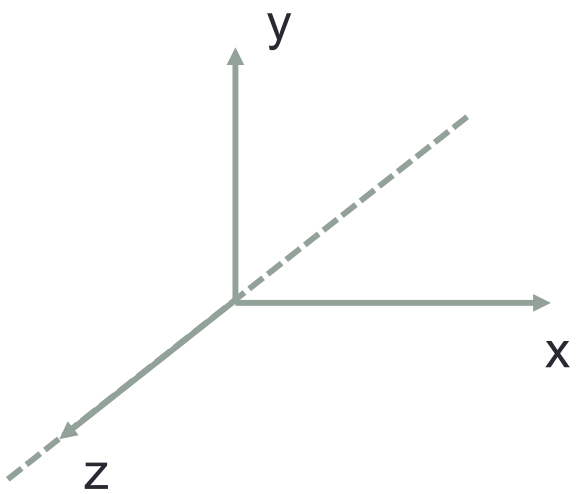
$R_C = R_A R_B$

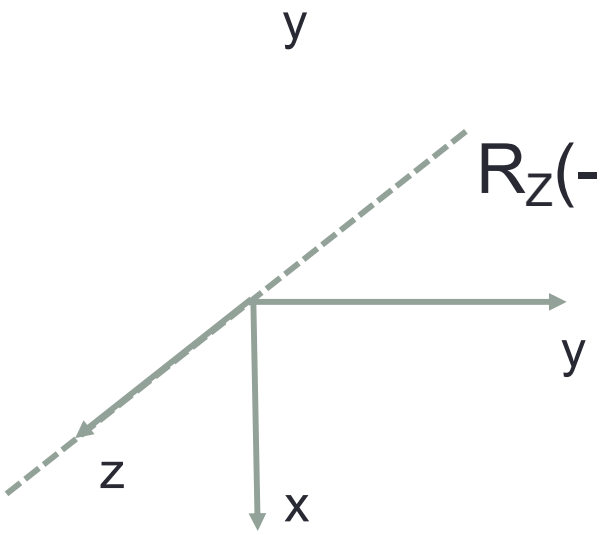
Example: elementary rotations

$$R_Y(90) = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$





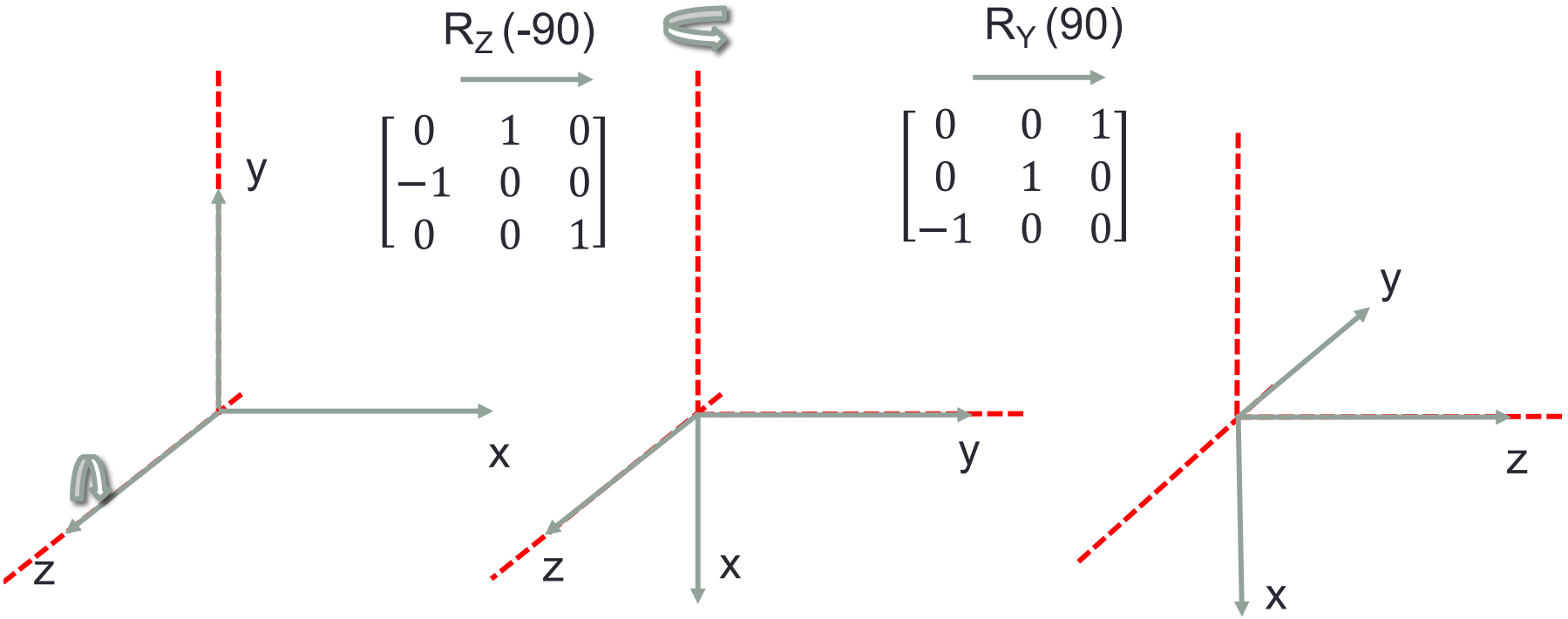




$$R_Z(-90) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Composition of extrinsic rotations

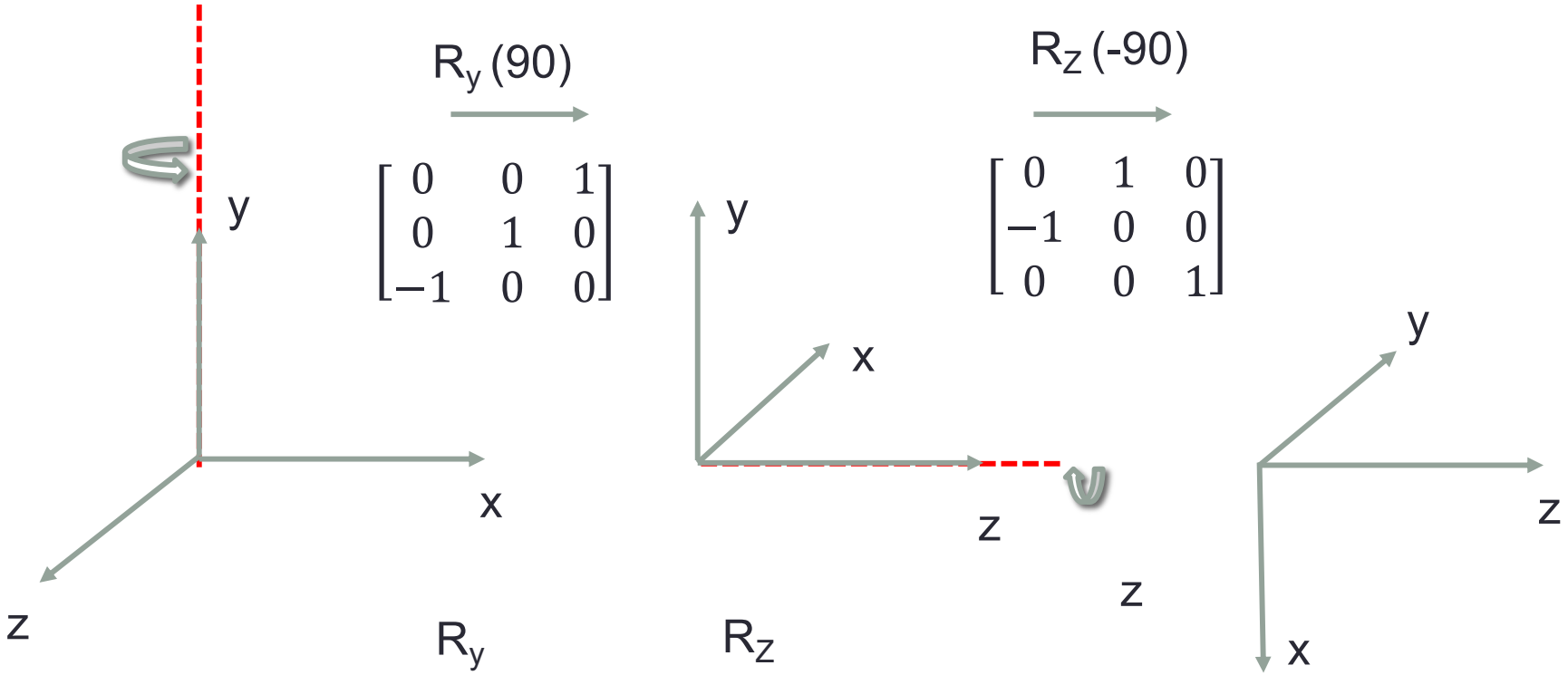
- The multiplication $R_Y R_Z$ can be interpreted as to first apply the rotation R_Z and then R_Y , both around the **fixed axis** (dashed line)



$$R_Y R_Z = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Composition of intrinsic rotations

- The multiplication $R_Y R_Z$ is interpreted as to first apply the rotation R_Y and then R_Z around the **mobile axis** (dashed line)

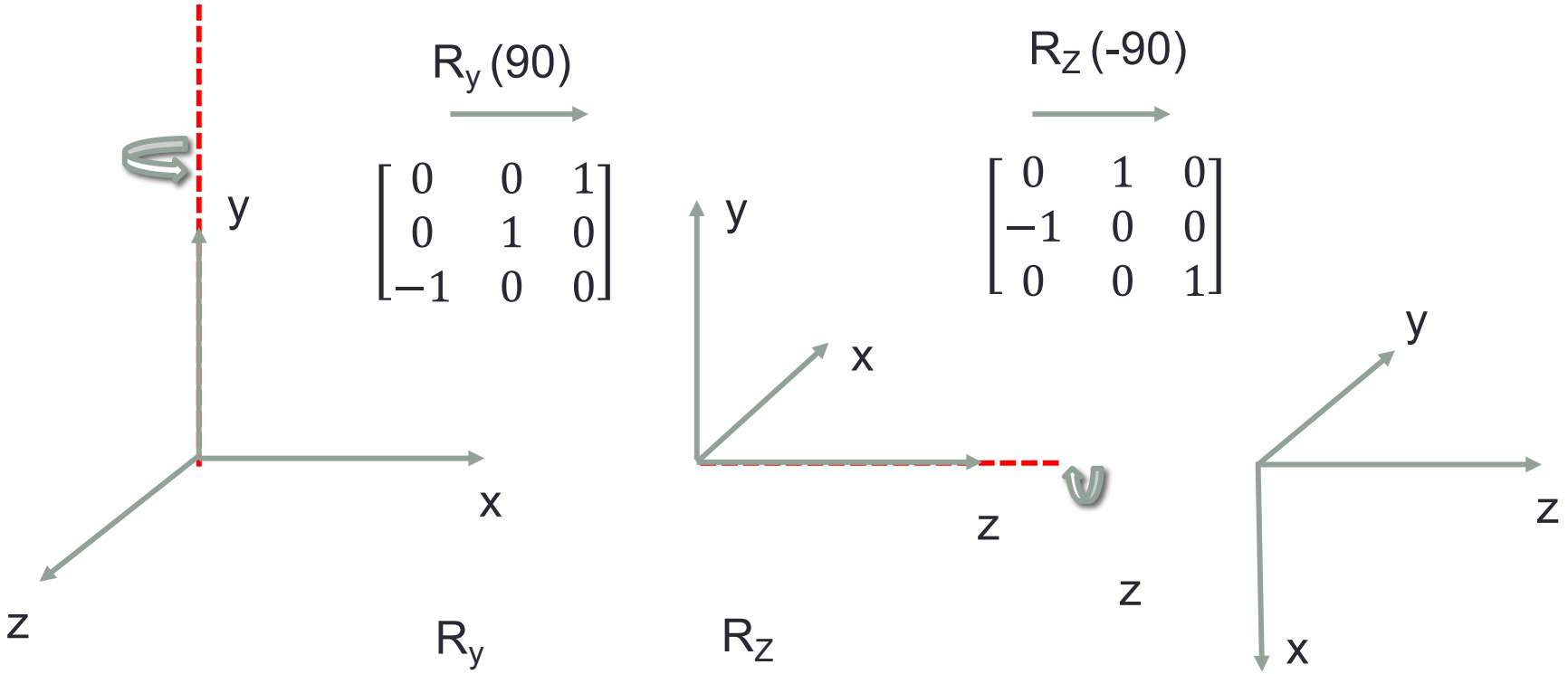


$$R_Y R_Z = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

(R_Y and R_Z above the first two matrices)

Composition of intrinsic rotations

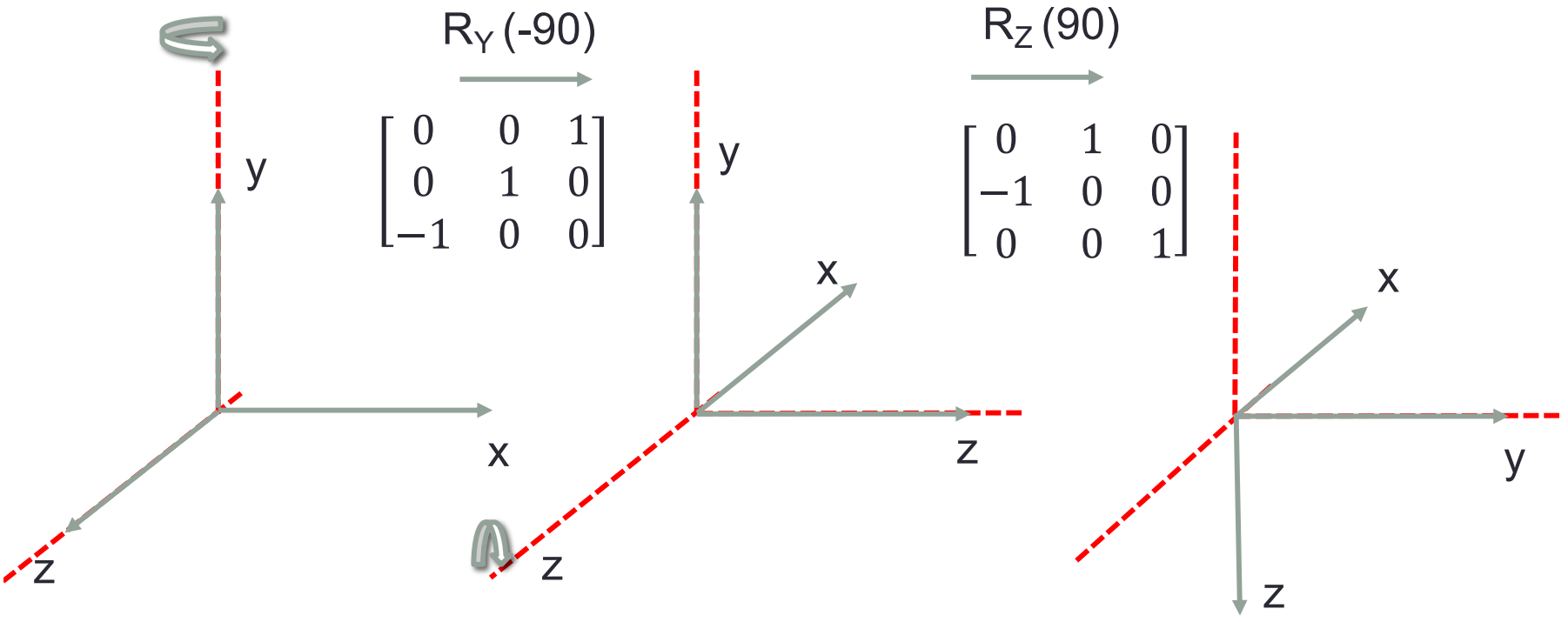
- The multiplication $R_Y R_Z$ is interpreted as to first apply the rotation R_Y and then R_Z around the **mobile axis** (dashed line)



$$\begin{matrix} R_y \\ \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} \end{matrix} \begin{matrix} R_Z \\ \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{matrix} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$$

Composition of extrinsic rotations

- The multiplication $R_Z R_Y$ can be interpreted as to first apply the rotation R_Y and then R_Z , both around the **fixed axis** (dashed line)



$$R_Z R_Y = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ -1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{bmatrix}$$

Smartphone orientation

- Any orientation can be expressed by three different elemental rotations of a given amount around linearly independent axis (Euler's Theorem)
- These angles are called the Euler angles (the values depend on the order of the rotations and if rotations are extrinsic or intrinsic)
- Tiat-Bryan angles are rotations along three **different** axis
- Orientation in smartphone are Tait-Bryan angles, z-x-y
- The angles are the yaw, pitch, roll and can be interpreted as intrinsic rotations (first yaw, followed by pitch followed by roll)

From rotation matrix to orientation angles (android)

Intrinsic rotations: yaw(Z)→pitch(X)→roll(Y)
or extrinsic (roll→pitch→yaw)

$$\mathbf{R_z R_x R_Y} = \begin{bmatrix} c_z & s_z & 0 \\ -s_z & c_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_x & s_x \\ 0 & -s_x & c_x \end{bmatrix} \begin{bmatrix} c_y & 0 & s_y \\ 0 & 1 & 0 \\ -s_y & 0 & c_y \end{bmatrix} = \begin{bmatrix} * & c_x s_z & * \\ * & c_x c_z & * \\ -c_x s_y & -s_x & c_x c_y \end{bmatrix}$$

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} = \begin{bmatrix} * & c_x s_z & * \\ * & c_x c_z & * \\ -c_x s_y & -s_x & c_x c_y \end{bmatrix}$$

$$\begin{pmatrix} cy\ cz + sx\ sy\ sz & cx\ sz & -cz\ sy + cy\ sx\ sz \\ cz\ sx\ sy - cy\ sz & cx\ cz & cy\ cz\ sx + sy\ sz \\ cx\ sy & -sx & cx\ cy \end{pmatrix}$$

$$azimuth(Z) = tg^{-1}\left(\frac{r_{12}}{r_{22}}\right)$$

←for r_{22}=0 roll is undefined

$$pitch\ (X) = sin^{-1}\left(-r_{32}\right)$$

$$roll(Y) = tg^{-1}\left(\frac{-r_{31}}{r_{33}}\right)$$

←for r_{33}=0 roll is undefined

atan2(y,x) is used

Browsing the source code..

https://developer.android.com/guide/topics/sensors/sensors_position

```
 */
public static float[] getOrientation(float[] R, float[] values) {
    /*
     * 4x4 (length=16) case:
     *   /  R[ 0]   R[ 1]   R[ 2]   0  \
     *   |  R[ 4]   R[ 5]   R[ 6]   0  |
     *   |  R[ 8]   R[ 9]   R[10]   0  |
     *   \      0       0       0   1  /
     *
     * 3x3 (length=9) case:
     *   /  R[ 0]   R[ 1]   R[ 2]  \
     *   |  R[ 3]   R[ 4]   R[ 5]  |
     *   \  R[ 6]   R[ 7]   R[ 8]  /
     *
     */
    if (R.length == 9) {
        values[0] = (float) Math.atan2(R[1], R[4]);
        values[1] = (float) Math.asin(-R[7]);
        values[2] = (float) Math.atan2(-R[6], R[8]);
    } else {
        values[0] = (float) Math.atan2(R[1], R[5]);
        values[1] = (float) Math.asin(-R[9]);
        values[2] = (float) Math.atan2(-R[8], R[10]);
    }

    return values;
}
```

A numerical example:

- G=[0.188,0.940,9.78]
- B= [14.224, -21.589, -32.692]

- MATRIX
- -0.77668375 -0.6254023 0.075061865
- 0.6295999 -0.7744137 0.06234753
- 0.019136654 0.09568327 0.9952279

- az,ax,ay = -2.462246 -0.095829874 0.019226

Summary

- The rotation matrix is used to map readings from sensors (measured wrt to the device frame) to the Earth reference system

- The rotation matrix is used to determine the orientation of the device

- QUESTIONS?