Orientation, Rotation, Euler Angles & Quaternion

ESIEE [it]
L'école de l'expertise numérique

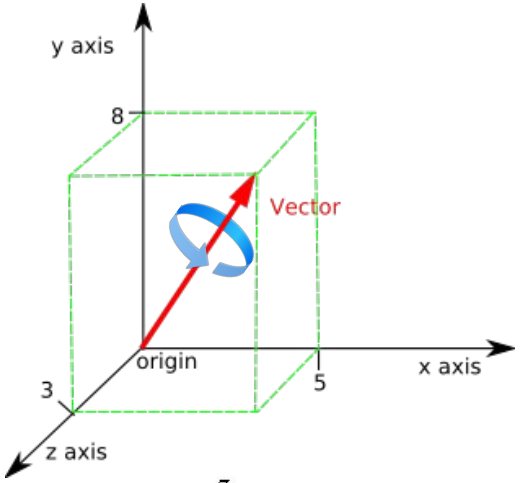
What is Orientation ?

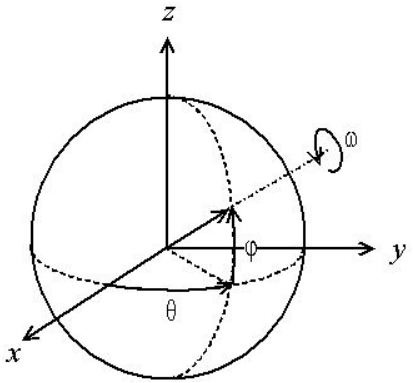
- Notions not to be confused:
  - direction
  - orientation
  - rotation
  - angular displacement

- A vector has a direction, but not an orientation: the vector can rotate on itself longitudinally, the vector does not undergo a real change. Why? The vector has no thickness and its only dimension is its length.
- A direction can be defined in space with two parameters (θ,φ)

- An object that has a thickness and that is rotated along an axis undergoes a change in orientation.
- An orientation is defined with 3 angular parameters, the Euler angles: heading, pitch, roll. (next slide)
- Like position, an orientation cannot be defined in absolute terms, but always in relation to a given reference frame. The 3 angles define in fact the rotation "from the null orientation to the considered orientation".
- The amount of rotation is called angular displacement.

- The orientation defines a state (static), the rotation a difference of states (dynamic).
- (analogy with position/translation... position is static, while translation is dynamic)





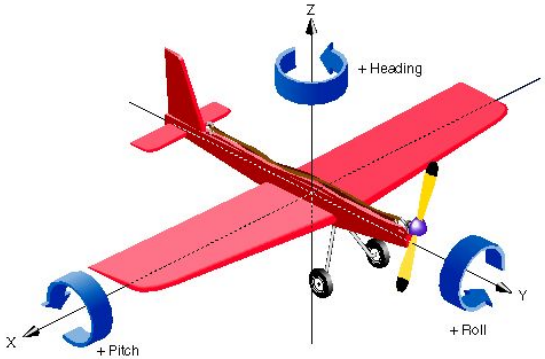
Euler Angles

- Leonhard Euler (1707-1783), Swiss mathematician

- Euler's angles define an orientation as a succession of 3 rotations around 3 perpendicular axes (two by two). Which axes? Any triplet of perpendicular axes, but for simplicity we will choose the axes (Ox), (Oy), (Oz).

- We will use the "Heading-Pitch-Bank" convention for the Euler angles. From a reference orientation, we apply the 3 angular displacements in this order:
  - Step 1: Heading, rotation around the local axis (Oy)
  - Step 2 : Pitch, rotation around the local axis (Ox)
  - Step 3: Bank, rotation around the local axis (Oz)

- Note the similarity of steps 1 & 2 with the spherical coordinate system
  - Heading & Pitch define the direction
- Sometimes other terminology is used:
  - Roll = Tilt = Twist = Bank
  - Yaw ≈ Heading, Azimuth = Heading
  - Attitude = Elevation = Pitch



Euler Angles …. In the computer

- In computer science, the calculations performed by the computer actually use the inverse convention "Roll-Pitch-Yaw" (Bank-Pitch-Heading).

- The three rotations are applied in reverse order, and according to the axes of the parent frame of reference, and no longer local:
  - Step 1: Bank/Roll, rotation around the parent frame axis (Oz)
  - Step 2: Pitch, rotation around the parent frame axis (Ox)
  - Step 3: Heading/Yaw, rotation around the axis of the parent frame (Oy)

- The result is identical!

- We speak of the extrinsic convention (Roll-Pitch-Yaw), opposed to the intrinsic convention (Heading-Pitch-Bank).
- The extrinsic convention has the advantage of being based on fixed reference axes, which greatly facilitates the calculations.

- The difference between Yaw & Heading ... because there is one often swept away...
  - Yaw: rotation around the local axis (Oy)
  - Heading: rotation around the parent referential axis (Oy)

- Euler & Tait-Bryan angles: Euler uses only 2 axes to carry out 3 rotations, Tait-Bryan introduced the third axis

Euler Angles … pros & cons

- Advantages:
  - Euler angles are intuitive, they stick quite well to our natural perception of orientation.
  - Memory footprint minimum: three numerical values easily compressible and evenly distributed over the interval [0;2π] (Fixed-point numbers possible).
  - All angle values are mathematically valid.

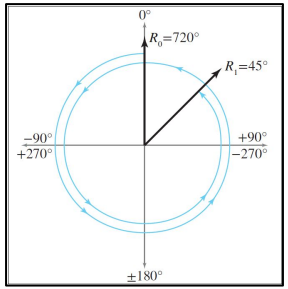
- Disadvantages:
  - 1st Aliasing problem → there is not a unique triplet of angles for a given orientation (non-bijectivity)
    - example: (pitching down 135°) = (heading 180°, pitching down 45°, banking 180°)
    - the solution: to ensure the bijection between Euler angles & orientation, we will limit the angles to the canonical Euler angles:

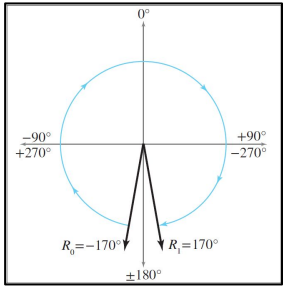
      -180° ≤ heading < 180° -180° ≤ bank < 180° -90° ≤ pitch ≤ 90°

      if pitch = +90° or -90°, then bank = 0 (Gimbal Lock situation, see below)
  - 2nd aliasing problem: "Gimbal Lock". when pitch is +90° or -90°, we are restricted to turn around the vertical axis according to (Oy) and (Oz), we lose one degree of rotation.
    By convention, we will set bank to 0, and the rotation around the vertical axis (Oy) will be assured by heading
    Gimbal Lock! is an unsolvable problem
    https://www.youtube.com/watch?v=zc8b2Jo7mno

  - Interpolation between two orientations is problematic, very problematic!
    This is due to the cyclic nature of angles & Gimbal Lock!
  - https://www.youtube.com/watch?v=ztIXMVLhUYA (illustration of the cyclic nature, not Gimbal Lock as announced by the video title!)
  - Angle Wrapping algorithms solve the problem induced by the cyclic nature of angles

    Gimbal Lock remains a major problem ... which definitely disqualifies Euler angles to ensure rotations






```
float wrapPi(float theta) {

    // Check if already in range.  This is not strictly necessary,
    // but it will be a very common situation.  We don't want to
    // incur a speed hit and perhaps floating precision loss if
    // it's not necessary
    if (fabs(theta) <= PI) {

        // One revolution is 2 PI.
        const float TWOPPI = 2.0f*PI;

        // Out of range.  Determine how many "revolutions"
        // we need to add.
        float revolutions = floor((theta + PI) * (1.0f/TWOPPI));

        // Subtract it off
        theta -= revolutions*TWOPPI;
    }

    return theta;
}
```

Quaternions to represent an orientation

- A quaternion, noted **q**, is a mathematical object which contains 4 values:
  - A vector (x,y,z)
  - A scalar value w

$$\begin{bmatrix} w & \begin{pmatrix} x & y & z \end{pmatrix} \end{bmatrix}$$

- A quaternion can represent a rotation of a quantity θ about a unit vector n.
  - Be careful, x, y and z do not correspond directly to the coordinates of the vector n

$$\begin{bmatrix} w & \begin{pmatrix} x & y & z \end{pmatrix} \end{bmatrix} = \begin{bmatrix} \cos(\theta/2) & \begin{pmatrix} \sin(\theta/2)n_x & \sin(\theta/2)n_y & \sin(\theta/2)n_z \end{pmatrix} \end{bmatrix}$$

- ***q*** *and* ***-q*** *represent exactly the same rotation*
- ***q*** *and* ***q****, *its conjugate,* $\begin{bmatrix} w & \begin{pmatrix} -x & -y & -z \end{pmatrix} \end{bmatrix}$ *represent the opposite rotations*
- *Null rotation, the identity quaternion [1 (0 0 0)] or [-1 (0 0 0)]*
- *A quaternion representing an orientation is called a unit quaternion, because its norm is 1.*

$$\begin{aligned} \|\mathbf{q}\| &= \left\| \begin{bmatrix} w & \mathbf{v} \end{bmatrix} \right\| = \sqrt{w^2 + \|\mathbf{v}\|^2} \\ &= \sqrt{\cos^2(\theta/2) + (\sin(\theta/2)\|\hat{\mathbf{n}}\|)^2} \\ &= \sqrt{\cos^2(\theta/2) + \sin^2(\theta/2)\|\hat{\mathbf{n}}\|^2} \\ &= \sqrt{\cos^2(\theta/2) + \sin^2(\theta/2)(1)} \\ &= \sqrt{1} \\ &= 1. \end{aligned}$$

- Quaternions support many basic operations:
  - conjugate, inverse, "difference", dot product, log, exp, multiplication by a scalar, exponentiation,
  - multiplication: $\mathbf{q_2}*\mathbf{q_1}$ leads to the application in sequence of the rotations represented by $\mathbf{q_1}$ then $\mathbf{q_2}$
  - Linear interpolation and spherical interpolation

Quaternions … cons & pros

- Advantages:
  - No aliasing problems...there is bijection between orientations in space and quaternion value quadruplets
  - Interpolation is made easier.
  - The composition of rotations (quaternion multiplication) and the inversion of a rotation (quaternion conjugate) are fast
  - Quaternion <-> Matrix conversion a bit faster than for Euler angles
  - Takes less space in memory than a matrix (9 values)

- Disadvantages:
  - Takes up more memory space than Euler angles
  - Difficult to compress into fixed-point numbers
  - Not all quaternion values are valid rotations (because of successive rounding for example) ... just normalize the quaternion but this comes with a loss of precision
  - Not very intuitive