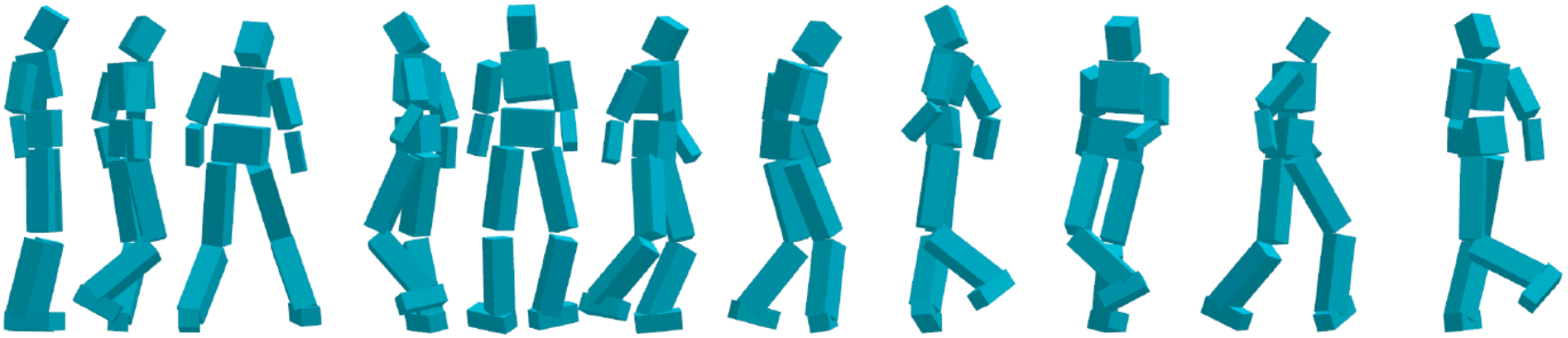
Movement and Artificial Intelligence

Skeleton Representation

Sylvie Gibet

Programming with Motion Capture

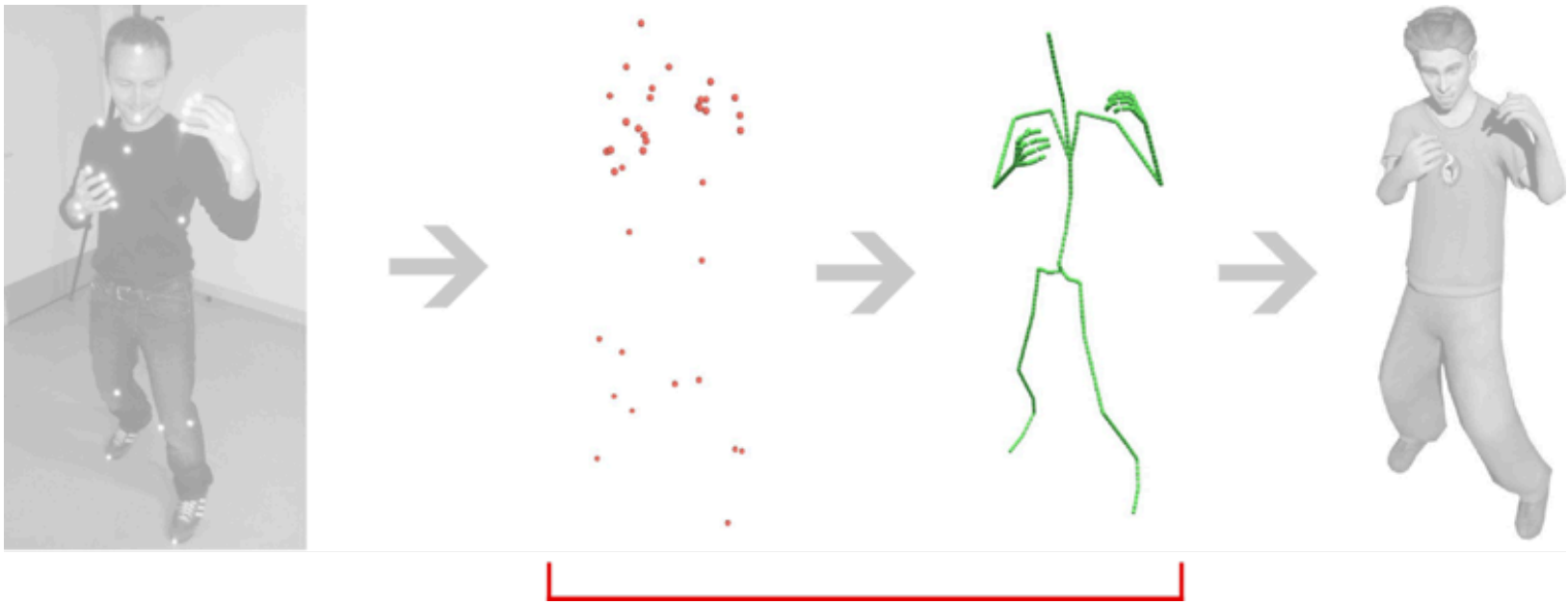
- Why is it difficult?
  - Encompass a lot of heterogeneous information
  - Joint angles
  - Position/orientation of a skeletal root
  - Their temporal trajectories
  - A number of local/global coordinate systems



Traditional Process for Animating Virtual Characters

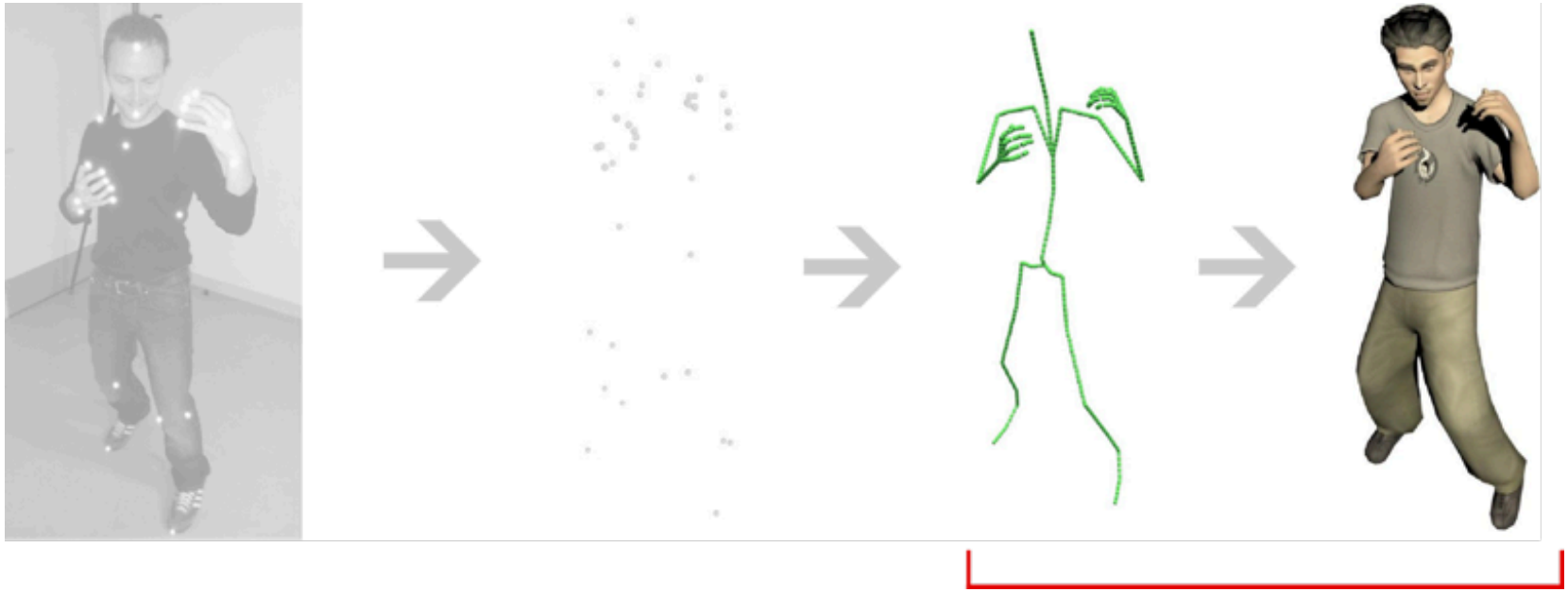
- **Data acquisition**
  - - Markers: set of 3D positions along time
  - - Processing
    - *markers labeling*
    - *processing markers inversion*
    - *dealing with occlusions (filling the gaps)*
    - *filtering*

Traditional Process for Animating Virtual Characters



- **Skeleton reconstruction**
  - Skeleton: hierarchical joints (position, rotations)
  - Processing
    - *calculation of joint centers*
    - *reconstruction of the skeleton along time*

Traditional Process for Animating Virtual Characters



- **Towards the animated mesh**
  - Animated mesh: structure composed of geometric and topological information
  - Processing
    - *coupling skeleton / mesh (rigging)*
    - *dealing with the skin deformation (skinning)*
    - *morphological adaptation (retargeting)*

Course overview

- Skeleton Modeling
  - Joints, Skeleton
- Position and Orientation Representations
  - Programming with positionsand translations
  - Programming with orientations and rotations
- Data-driven Animation with Motion Capture
  - Motion representation
  - Parsing mocap data and playing the motion
  - Pratical examples: motion exaggeration, style transfer, transition graph, connecting motion segments, warping, blending, time scaling, etc.

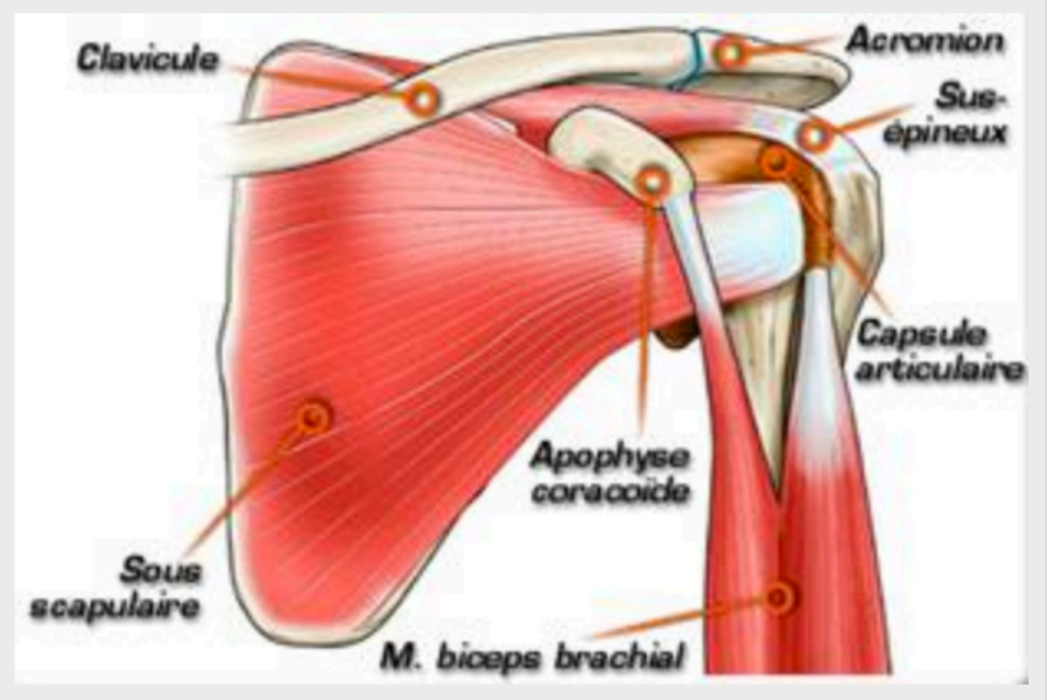
Course overview

- Skeleton Modeling
  - Joints, Skeleton
- Position and Orientation Representations
  - Programming with positionsand translations
  - Programming with orientations and rotations
- Data-driven Animation with Motion Capture
  - Motion representation
  - Parsing mocap data and playing the motion
  - Pratical examples: motion exaggeration, style transfer, transition graph, connecting motion segments, warping, blending, time scaling, etc.

Skeleton Modeling

- **Anatomy**
  - Joints
  - Bones
  - Muscles (create movement)
  - Tendons, ligaments, other tissues (limit the movement range)

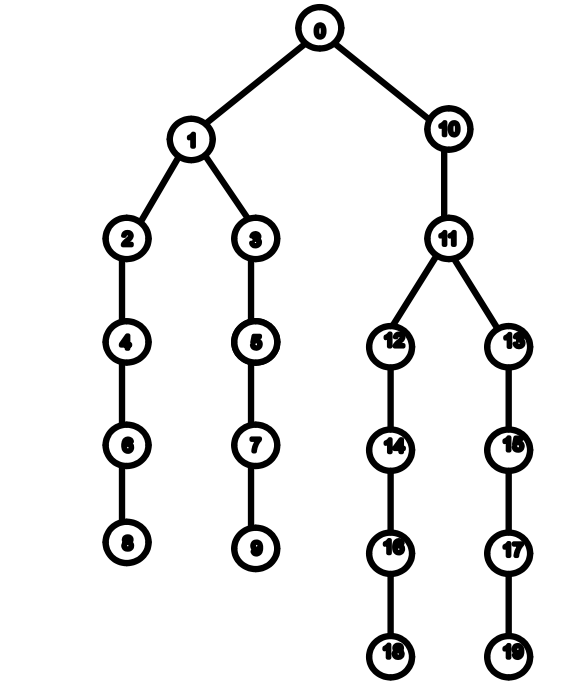
Shoulder joint



Skeleton Modeling

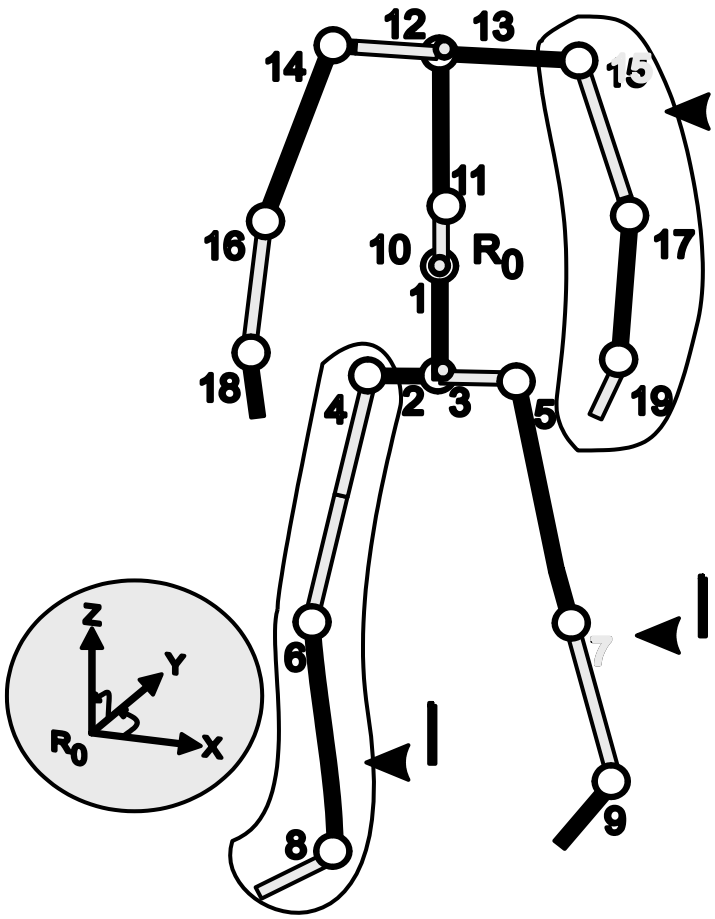
- **Joint**: one point
  - A joint allows relative movement within the skeleton. Joints are essentially 4x4 matrix transformations.
  - rotational, translational, or more specific
- **Bone**: a segment
  - Bone refers to the segment joining two successive joints: for example upper arm bone (humerus)
- **Skeleton**: a set of of joints arranged in a tree structure.
  - Local coordinates of the joints
  - Each joint has a parent and children (or leaf)
  - Root: the hip, or a point around the center of mass

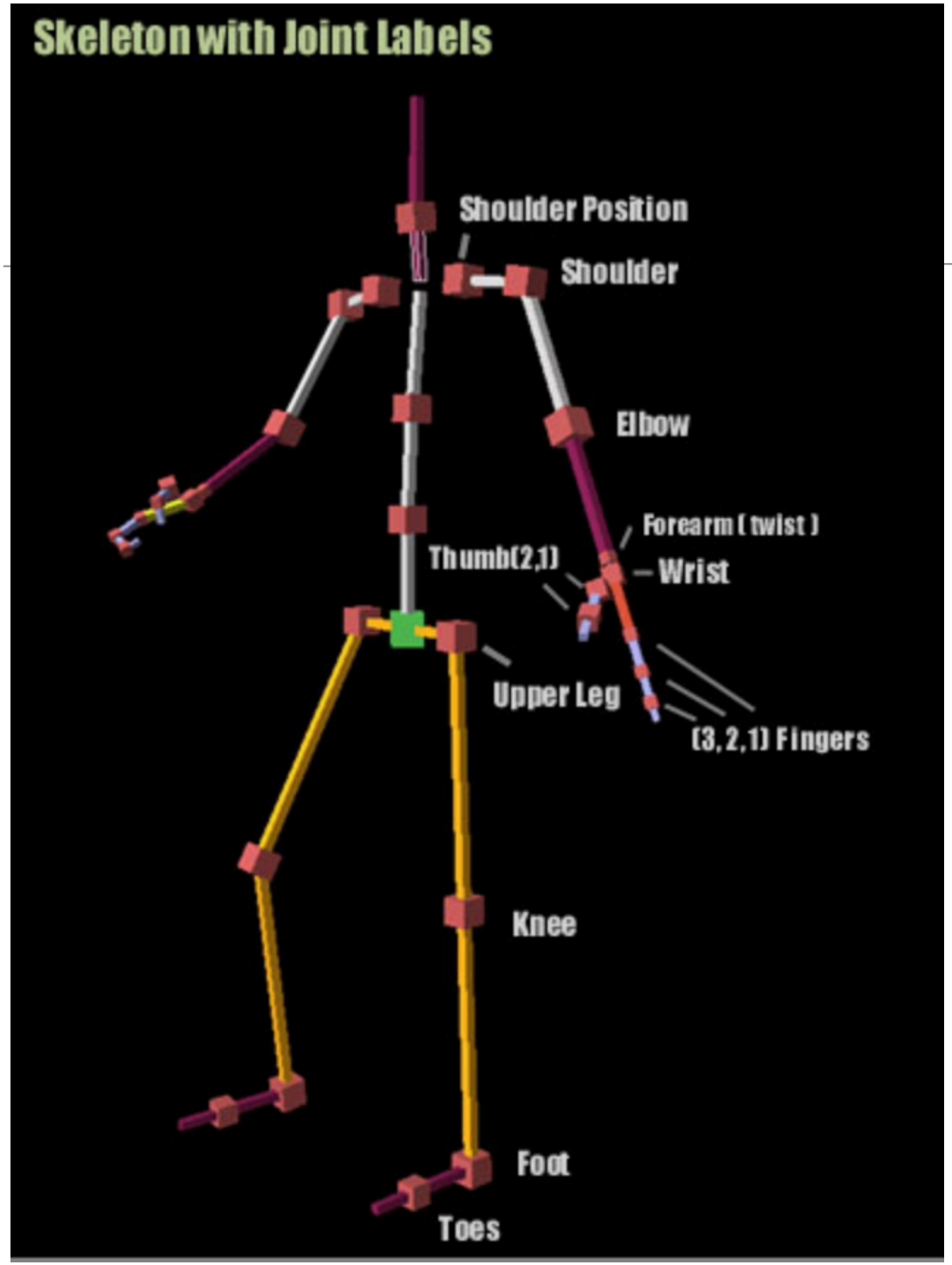
Squelette



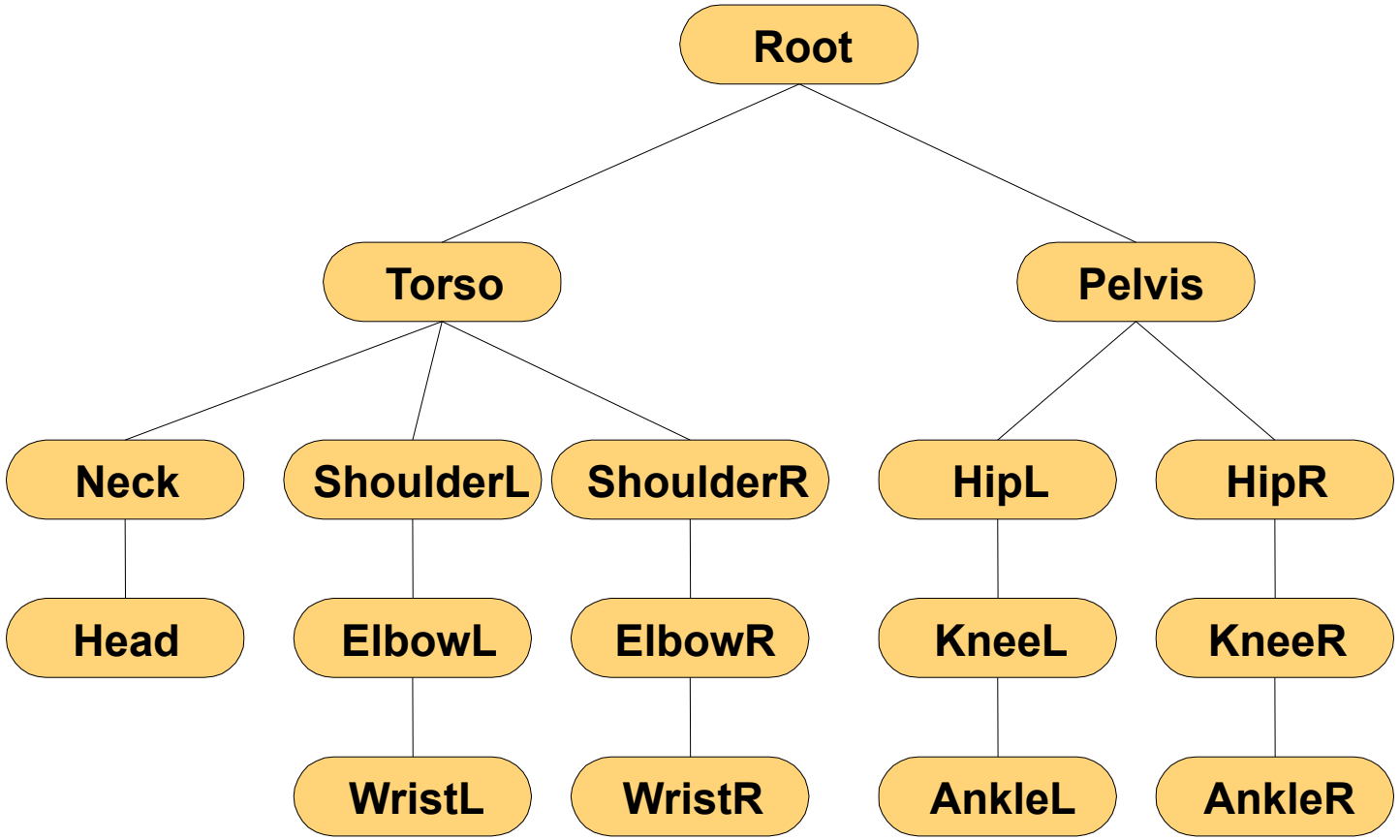
Node l (l=0..19)

L	ψ			θ			α		
	min	val	max	min	val	max	min	val	max




Skeleton with Joint Labels
Shoulder Position
Shoulder
Elbow
Forearm (twist)
Thumb(2,1)
Wrist
Upper Leg
(3, 2, 1) Fingers
Knee
Foot
Toes

Skeleton Modeling



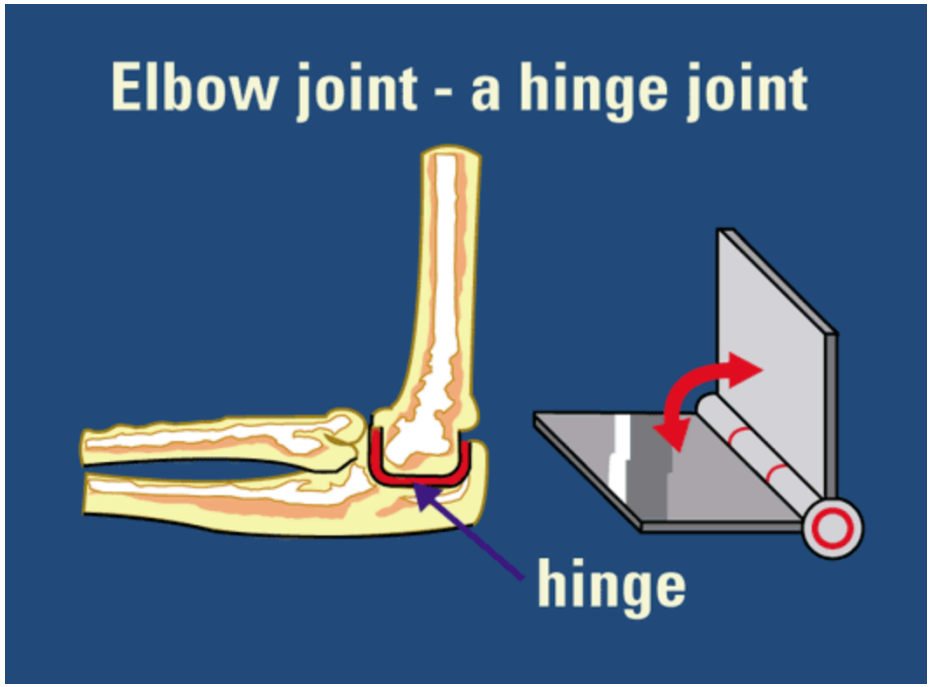
Degree of Freedom (DOF)

- **Degree of Freedom** (DOF): A variable describing a particular axis or dimension of movement within a joint
  - Rotation θ_x, θ_y, θ_z
  - Translation x, y, z

- Joints typically have 1- 6 DOFs

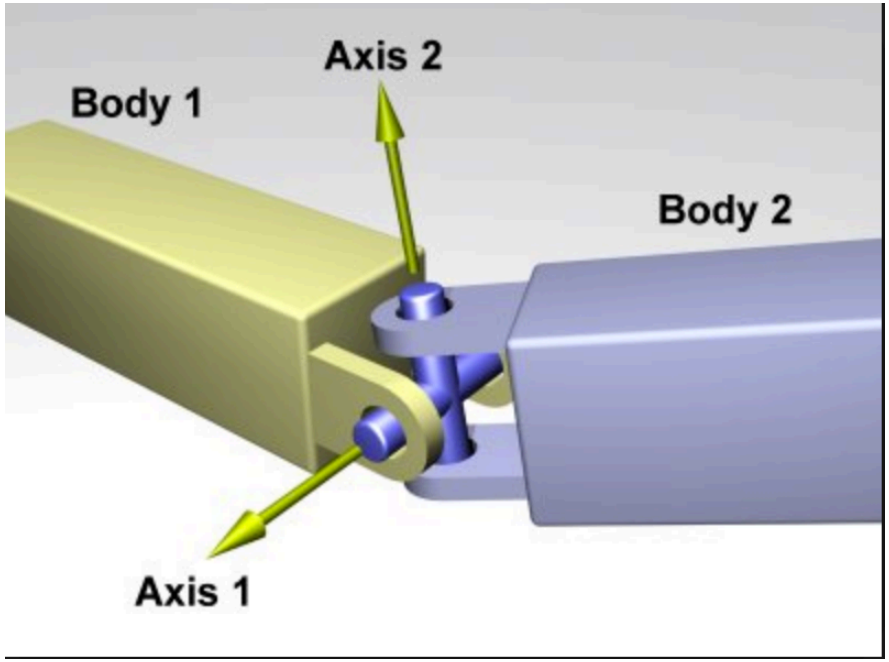
Joint Types

- **Rotational**
  - **Hinge: 1-DOF**
  - Universal: 2-DOF
  - Ball & Socket: 3-DOF
    - Euler Angles
    - Quaternions
- Translational
  - Prismatic: 1-DOF
  - Translational: 3-DOF (or any number)



Joint Types

- **Rotational**
  - Hinge: 1-DOF
  - **Universal: 2-DOF**
  - Ball & Socket: 3-DOF
    - Euler Angles
    - Quaternions
- Translational
  - Prismatic: 1-DOF
  - Translational: 3-DOF (or any number)

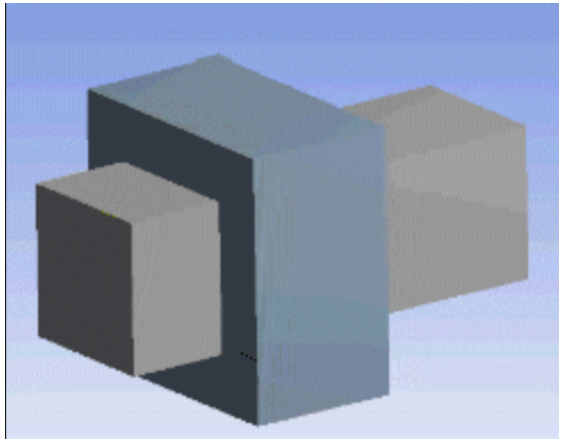


Joint Types

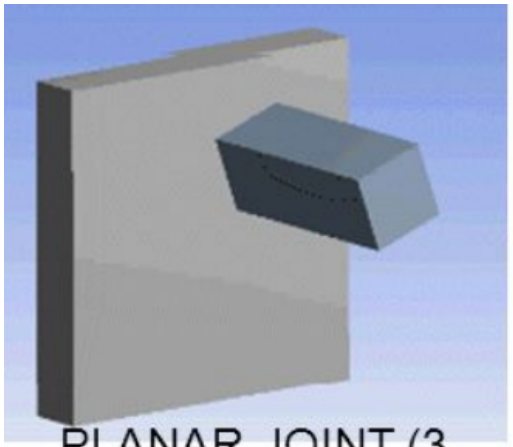
- **Rotational**
  - Hinge: 1-DOF
  - Universal: 2-DOF
  - **Ball & Socket: 3-DOF**
    - Euler Angles
    - Quaternions
- Translational
  - Prismatic: 1-DOF
  - Translational: 3-DOF (or any number)

Joint Types

- Rotational
  - Hinge: 1-DOF
  - Universal: 2-DOF
  - Ball & Socket: 3-DOF
    - Euler Angles
    - Quaternions
- Translational
  - Prismatic: 1-DOF
  - Translational: 3-DOF (planar joint)



1 DOF)



PLANAR JOINT (3 DOF)

DOF Limits and Variations along Time

- Each DOF should be limited to some range
  - for example, the elbow could be limited from 0º to 150º
- Usually, in a realistic character, all DOFs will be limited except the ones controlling the root.
- Changing the DOF values over time results in the animation of the skeleton (see movement representation)

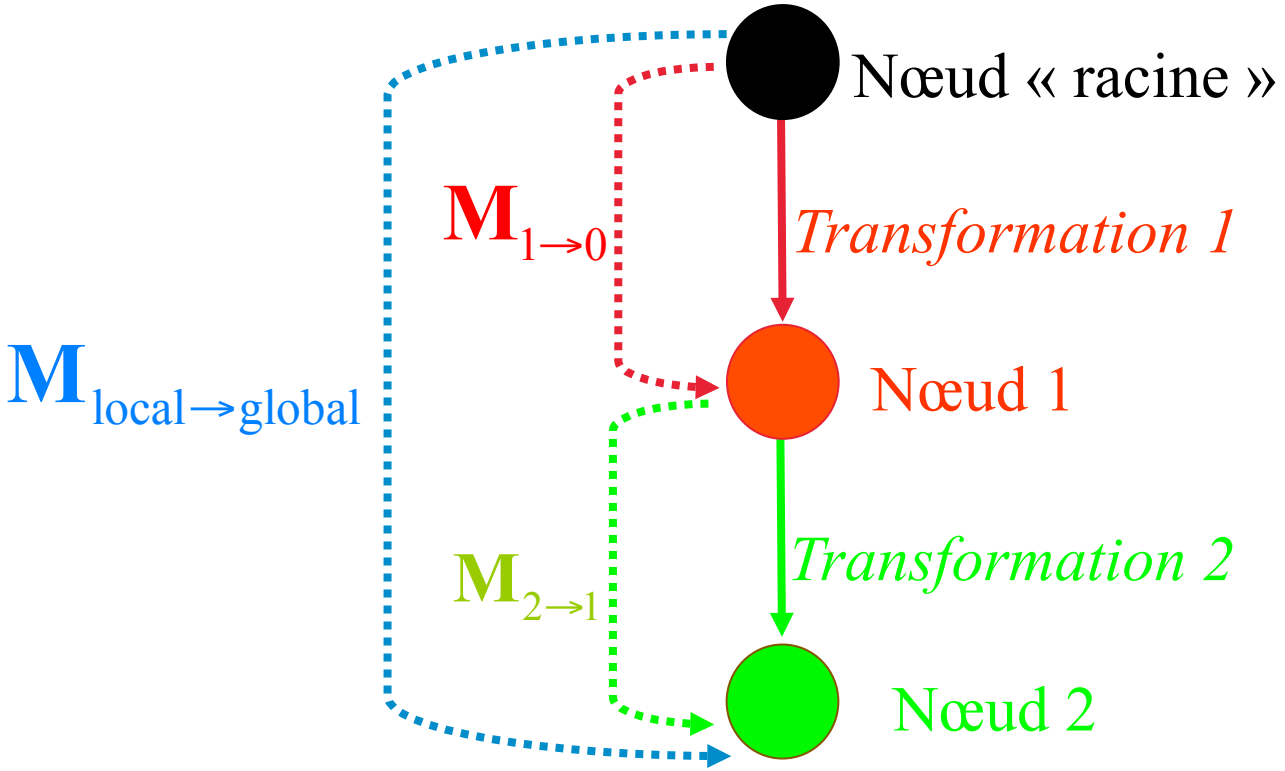
Skeleton Data

- When going from each joint to its child joint:
  - **Rotations** (from 1 to 3 floats): **R**
  - **Translation** (from 1 to 3 floats): **T**
- Adding **constraints** on each joint
  - Angular **limits** (min & max value per DOF)
  - Maximum force applied
- The skeleton data can be represented by Local matrices **M** or world matrices (computed from local ones)

Course overview

- Skeleton Modeling
  - Joints, Skeleton
- Position and Orientation Representations
  - Programming with positionsand translations
  - Programming with orientations and rotations
- Data-driven Animation with Motion Capture
  - Motion representation
  - Parsing mocap data and playing the motion
  - Pratical examples: motion exaggeration, style transfer, transition graph, connecting motion segments, warping, blending, time scaling, etc.

Hierarchy in the Skeleton Joints



$$\mathbf{M}_{\text{local}\to\text{global}} = \mathbf{M}_{1\to0} \cdot \mathbf{M}_{2\to1}$$

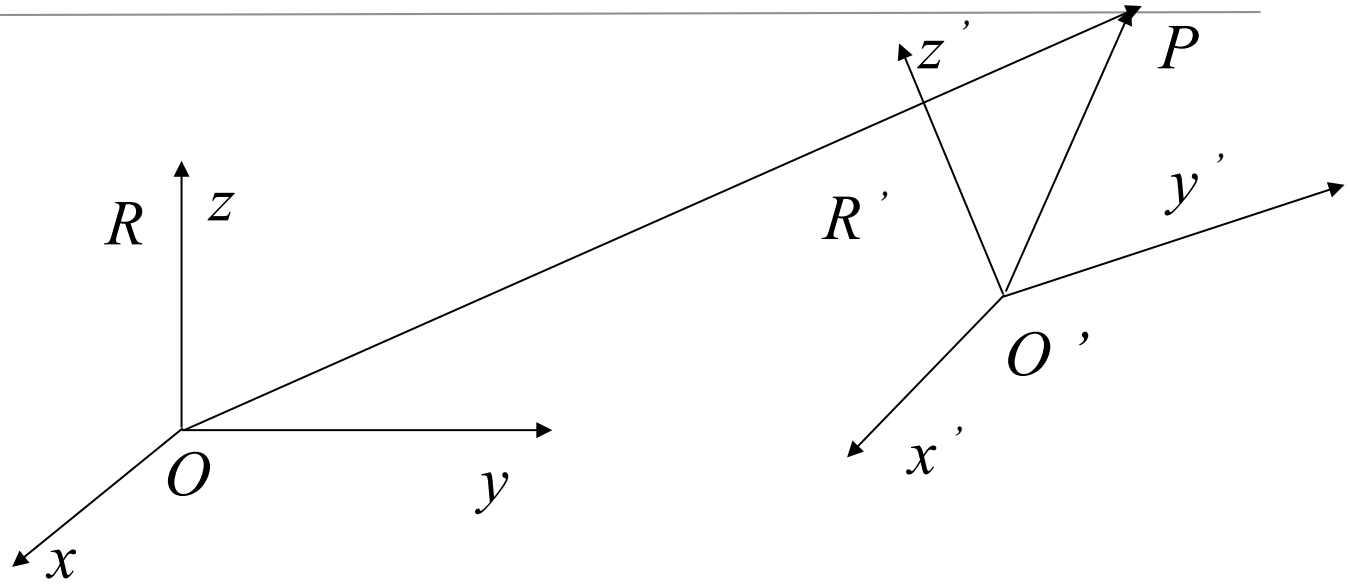
Homogeneous Matrix

- Definition: transforming the reference coordinate system (O,x,y,z) to (O,x,y,z') can be defined by the homogeneous matrix M:

$$M = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

  - R: Rotation matrix
  - T: Translation matrix

Homogeneous Matrix



- $OP = OO' + OP$
- Interest : gathering the information on the point P and on the orientation R

$$\begin{pmatrix} V \\ 0 \end{pmatrix}_{/R} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{pmatrix} V \\ 0 \end{pmatrix}_{/R'}$$

$$\begin{pmatrix} P \\ 1 \end{pmatrix}_{/R} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{pmatrix} P \\ 1 \end{pmatrix}_{/R'}$$

Different Ways to Describe positions and Orientations

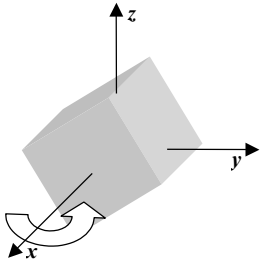
- R and T depend on the reference coordinate system

- Position : Cartesian coordinates
  Cylindar coordinates
  Spherical coordinates
- Orientation : Euler angles
  Quaternions (animation)
  Denavit Hartenberg (robotic)
  row, pitch, yaw (avionic)
- Which choice?
  - Depends on the system you want to model
  - Geometrical articulated system: number of DOF, animation method, ...

Hinge Joint: Rotation around one-axis

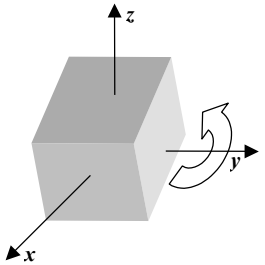
Rotation Matrix

- Rotation θ around x-axis



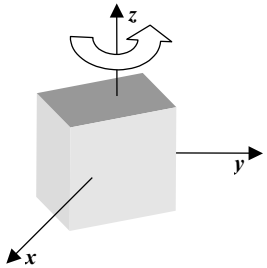
$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{pmatrix}$$

- Rotation θ around y-axis



$$\begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix}$$

- Rotation θ around z-axis



$$\begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Ball & Socket (3-DOF)

- Composition of 3 rotations around the 3 axis Ox, Oy, Oz:

$$R(\theta_1,\theta_2,\theta_3) = R_x(\theta_1)\mathrm{o}R_y(\theta_2)\mathrm{o}R_z(\theta_3)$$

$$\begin{pmatrix} c_2c_3 & c_2s_3 & -s_2 \\ s_1s_2c_3 - c_1s_3 & s_1s_2s_3 + c_1c_3 & s_1c_2 \\ c_1s_2c_3 + s_1s_3 & c_1s_2s_3 - s_1c_3 & c_1c_2 \end{pmatrix}$$

$$s_i = \sin\theta_i \quad \text{et} \quad c_i = \cos\theta_i \quad \text{pour } i = 1,2,3$$

Euler Angles

$$\left(\begin{array}{ccc|c} c_2c_3 & c_2s_3 & -s_2 & 0 \\ s_1s_2c_3 - c_1s_3 & s_1s_2s_3 + c_1c_3 & s_1c_2 & 0 \\ c_1s_2c_3 + s_1s_3 & c_1s_2s_3 - s_1c_3 & c_1c_2 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right) =$$

$$\left(\begin{array}{ccc|c} c_3 & s_3 & 0 & 0 \\ -s_3 & c_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\left(\begin{array}{ccc|c} c_2 & 0 & -s_2 & 0 \\ 0 & 1 & 0 & 0 \\ s_2 & 0 & c_2 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)\left(\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & c_1 & s_1 & 0 \\ 0 & -s_1 & c_1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right)$$

Problems with Euler Representation

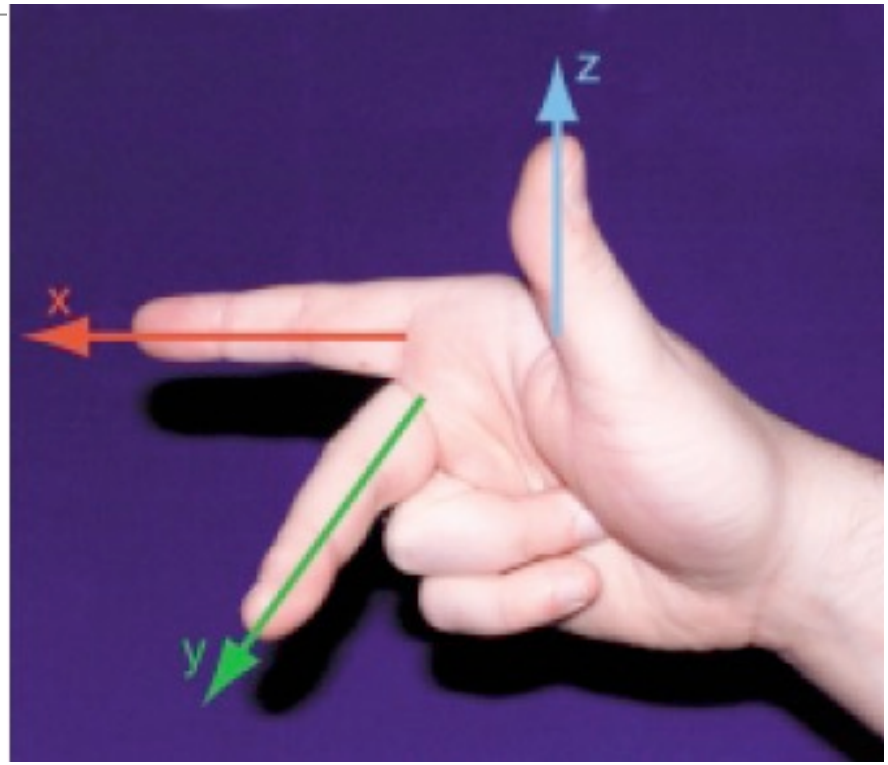
- *«**Gimbal lock**» problem:*
  - Case of a solid represented by Euler angles
  - **Non-unicity of the orientation representation!**
  - We may get a «degenerated» matrix when $\theta_2 = \pi/2$
  - Potential problems when interpolating between orientations!

$$R(\theta 1, \frac{\pi}{2}, \theta 3) = \begin{pmatrix} 0 & 0 & -1 & 0 \\ \sin(\theta 1 - \theta 3) & \cos(\theta 1 - \theta 3) & 0 & 0 \\ \cos(\theta 1 - \theta 3) & -\sin(\theta 1 - \theta 3) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$\Rightarrow$ *mathématiquement* : perte d'un DDL!

Problems with Euler Representation

Yaw: rotation / z (thumb)
Pitch: rotation / y (middle)
Roll: rotation / x (index)



If rotation of $\pi/2$ around z
then rotation of $\pi/2$ around y,
then x is aligned with z! (the index has the same direction than the thumb)

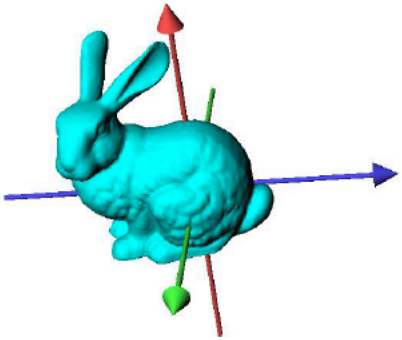
Each orientation that could be obtained by adding a rotation Rx (roll) in this new orientation could have been obtained by turning around z instead!

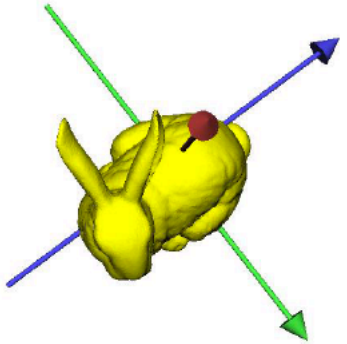
Orientation and Rotation

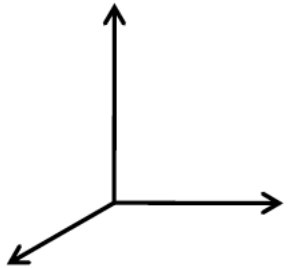
- ***Rotation***
  - Circular movement
- ***Orientation***
  - The state of being oriented
  - Given a coordinate system, the orientation of an object can be represented as a rotation from a reference pose

Analogy

- (point: vector) is similar to (orientation: rotation)
- Both represent a sort o (state: movement)



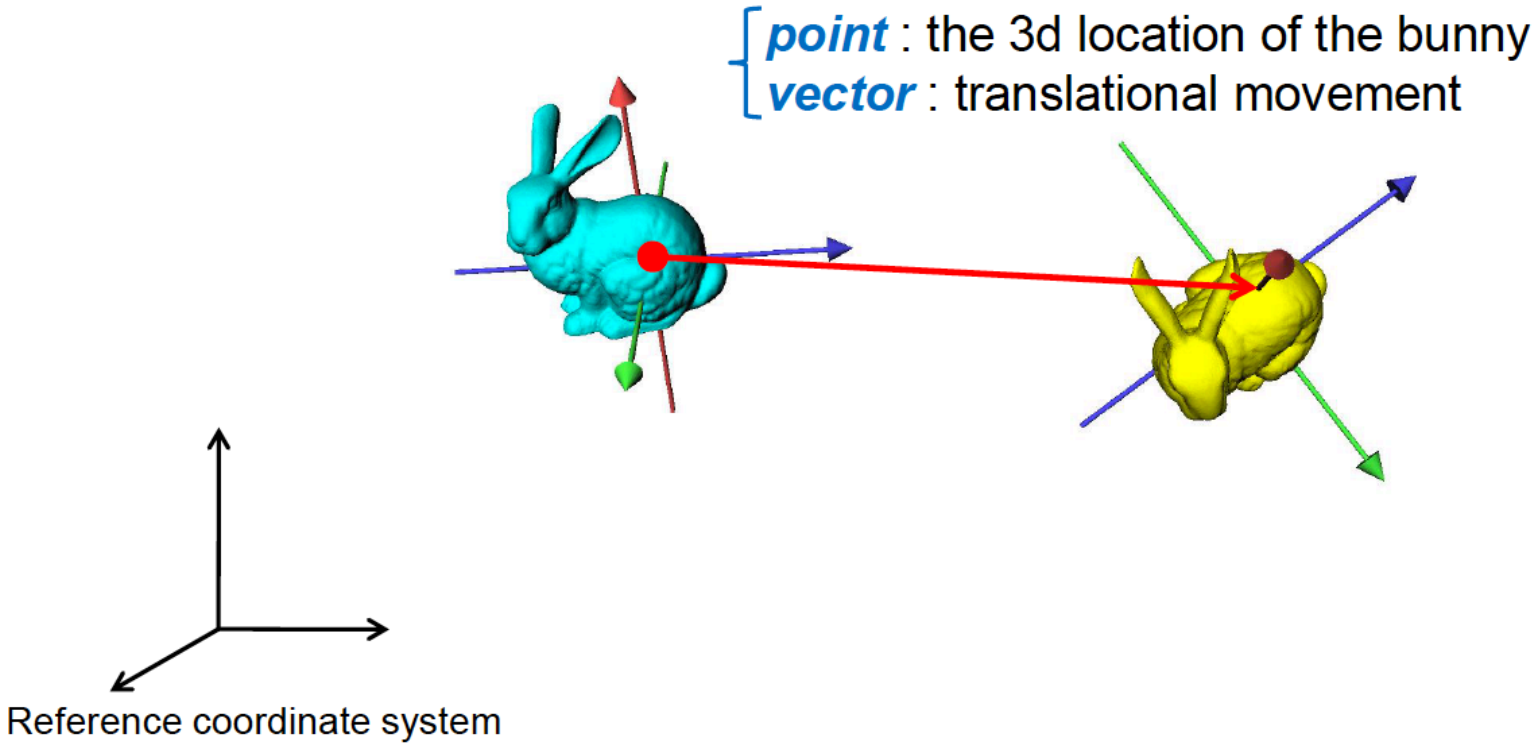




Reference coordinate system

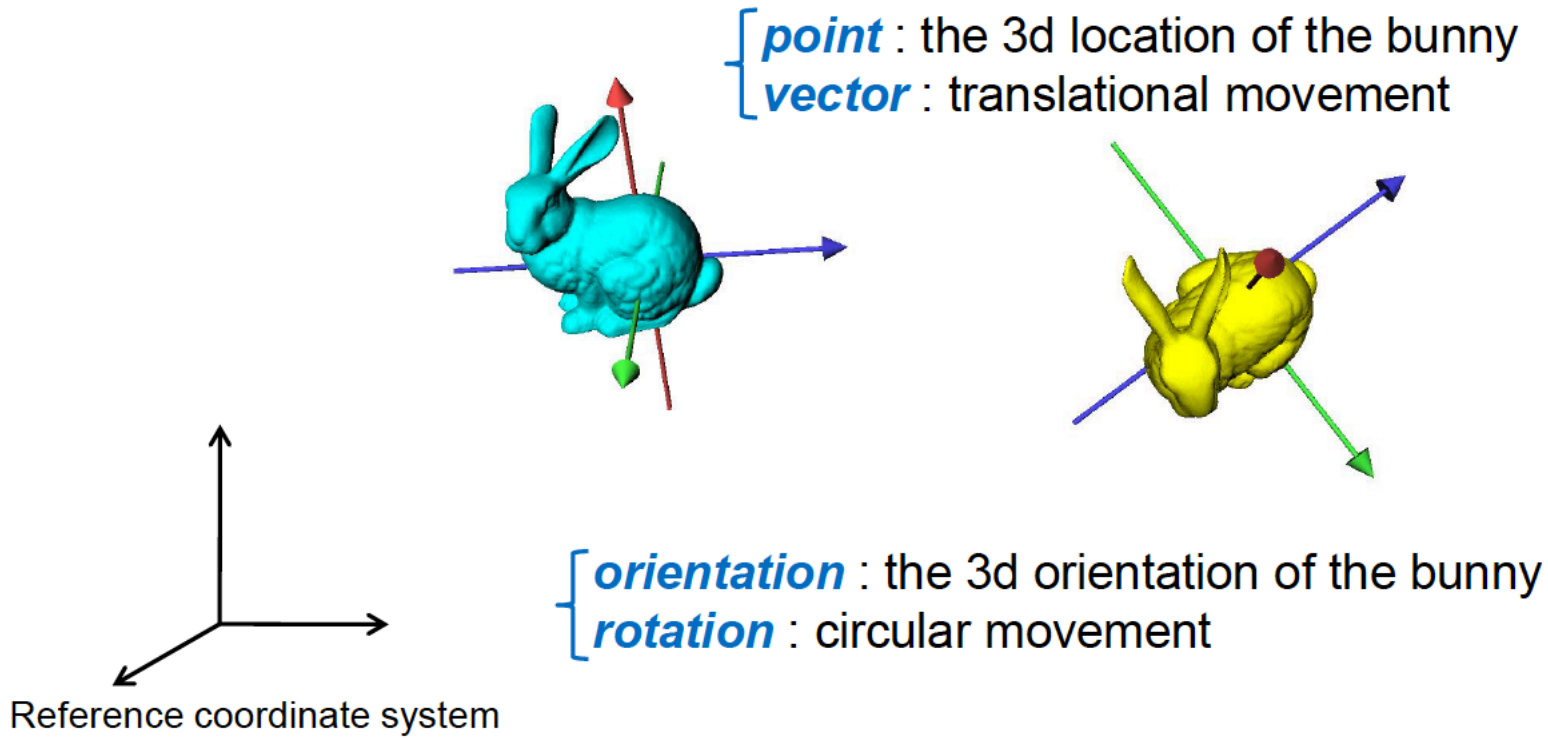
Analogy

- (point: vector) is similar to (orientation: rotation)
- Both represent a sort o (state: movement)



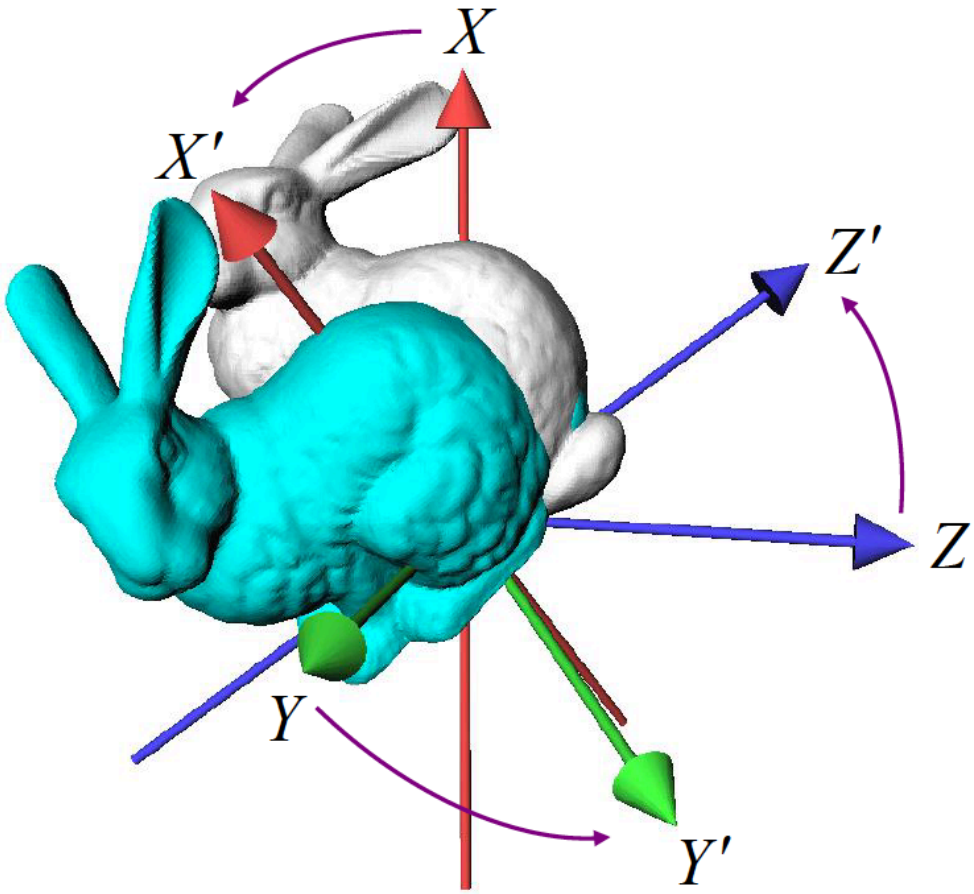
Analogy

- (point: vector) is similar to (orientation: rotation)
- Both represent a sort o (state: movement)



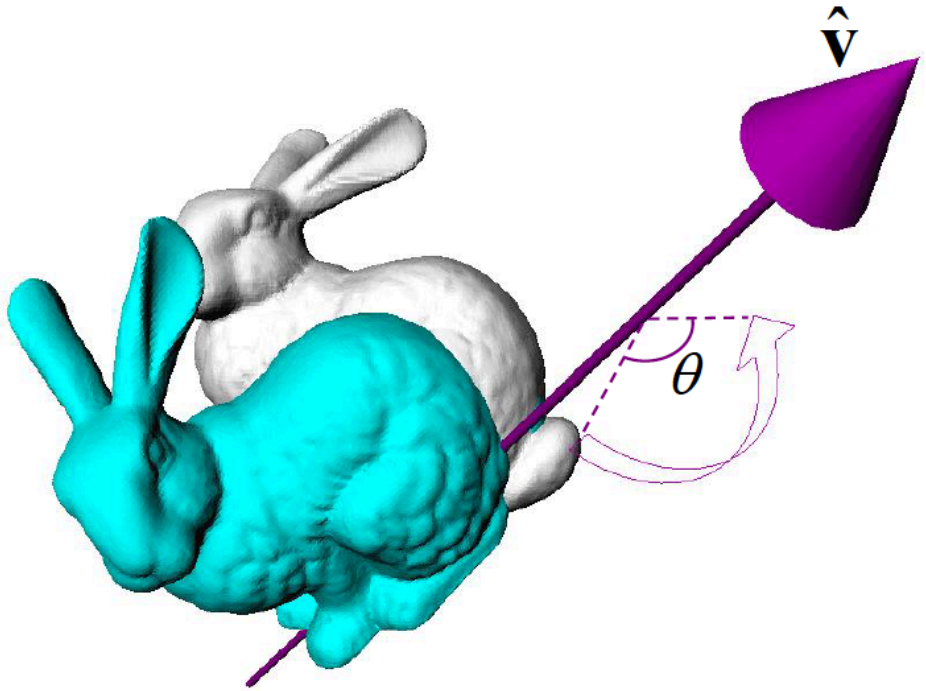
3D Orientation and Rotation

- Given two arbitrary orientations of a rigid object,

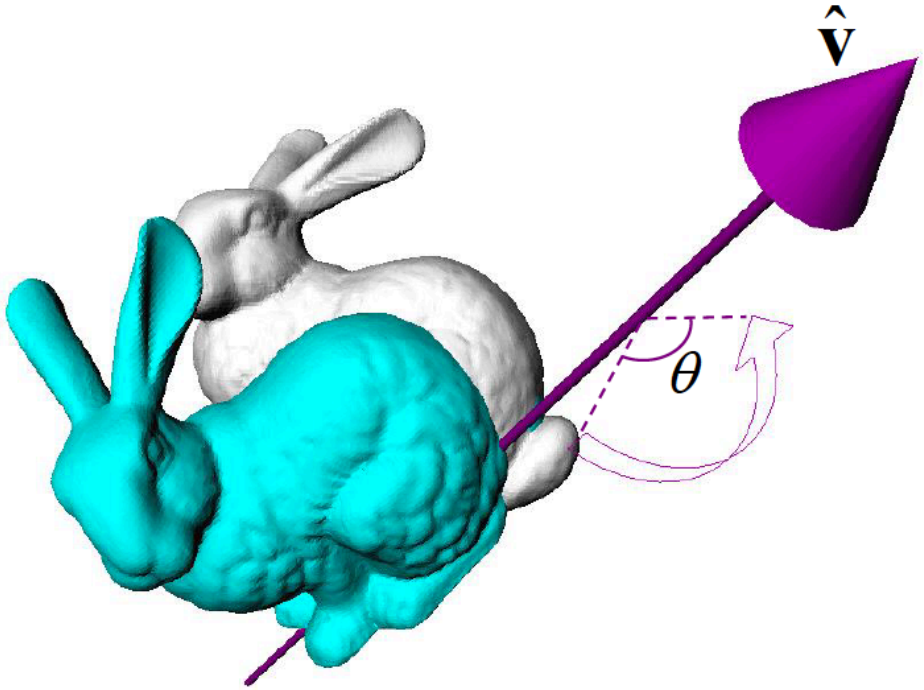


3D Orientation and Rotation

- **Euler's fundamental theorem**
  - We can always find a fixed axis of rotation and an angle about the axis



Rotation Vector



$\hat{\mathbf{v}}$: unit vector

θ : scalar angle

- Rotation vector (3 parameters) $\mathbf{v} = \theta\,\hat{\mathbf{v}} = (x, y, z)$
- Axis-Angle (2+1 parameters) $(\theta, \hat{\mathbf{v}})$

Representing Orientations with Quaternions

- Definition of quaternions
  - The notion of complex space in 2D is extended in 3D (Hamilton, 1943)

$$\mathbf{q} = w + x\,\mathbf{i} + y\,\mathbf{j} + z\,\mathbf{k} = (w, \mathbf{v})$$

    - w : scalar part of the quaternion
    - **v** : vectorial part of the quaternion

Quaternions

- **Interest of using quaternions**
  - Unicity of the representation

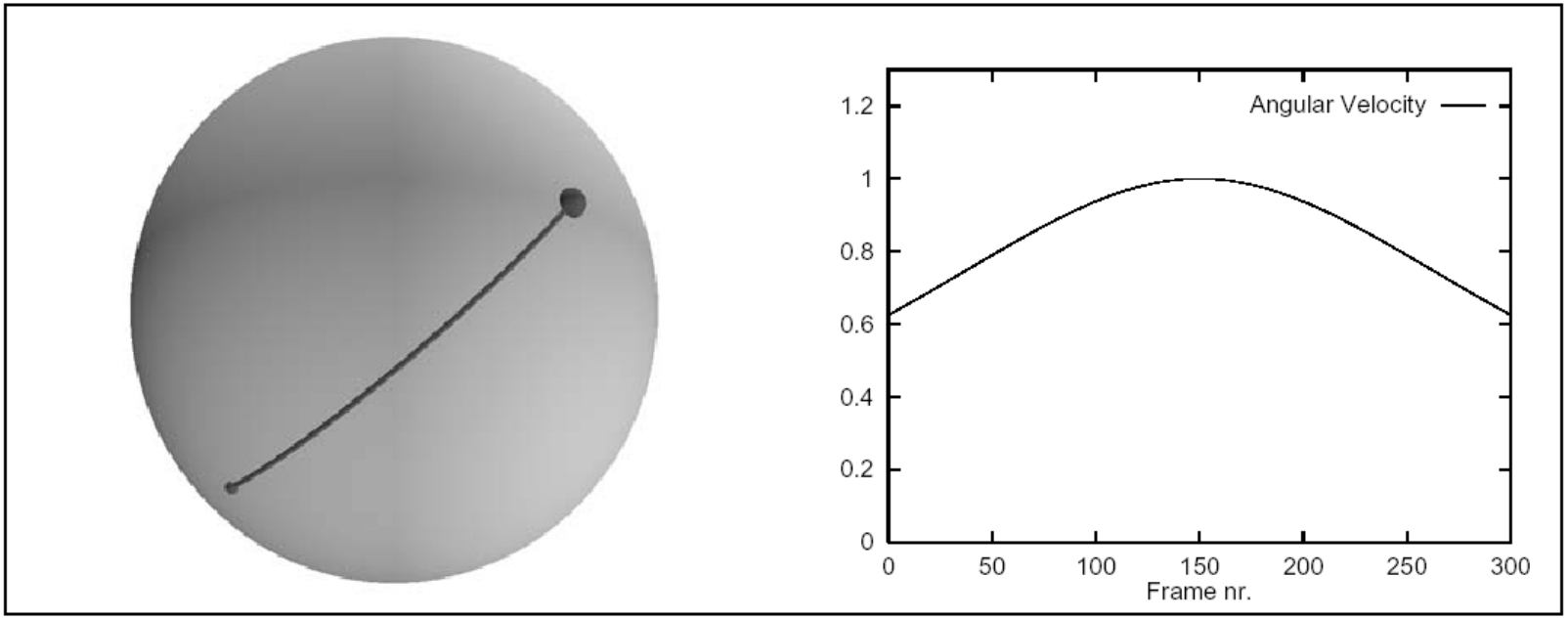
$$\mathbf{q} = \left[\cos\frac{\theta}{2}, \quad \sin\frac{\theta}{2}\mathbf{n}_x, \quad \sin\frac{\theta}{2}\mathbf{n}_y, \quad \sin\frac{\theta}{2}\mathbf{n}_z\right], \qquad \|\mathbf{q}\| = 1$$

  - Avoid the « **gimbal lock** » problem

  - Facilitate the interpolation between orientations

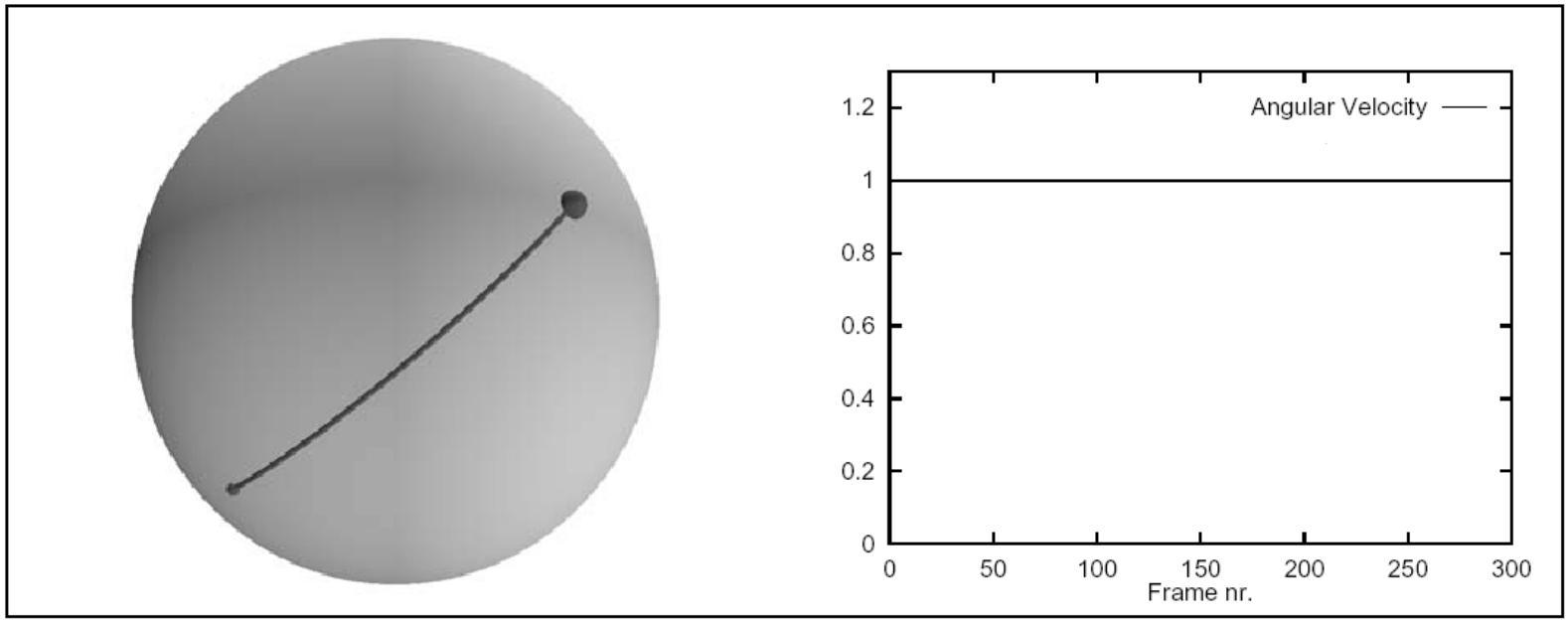
Linar Interpolation of orientations

$$\mathrm{lerp}(\mathbf{q}^1, \mathbf{q}^2, u) = (1-u)\mathbf{q}^1 + u\mathbf{q}^2$$



LinearInterpolation of Orientations

$$\mathrm{slerp}\left(\mathbf{q}^1,\mathbf{q}^2,u\right)=\frac{\sin\left(\left(1-u\right)\omega\right)\mathbf{q}^1+\sin\left(u\omega\right)\mathbf{q}^2}{\sin\omega}$$



Quaternions

- Going from Rotation matrix to quaternions is easy for the animator, and vice versa!