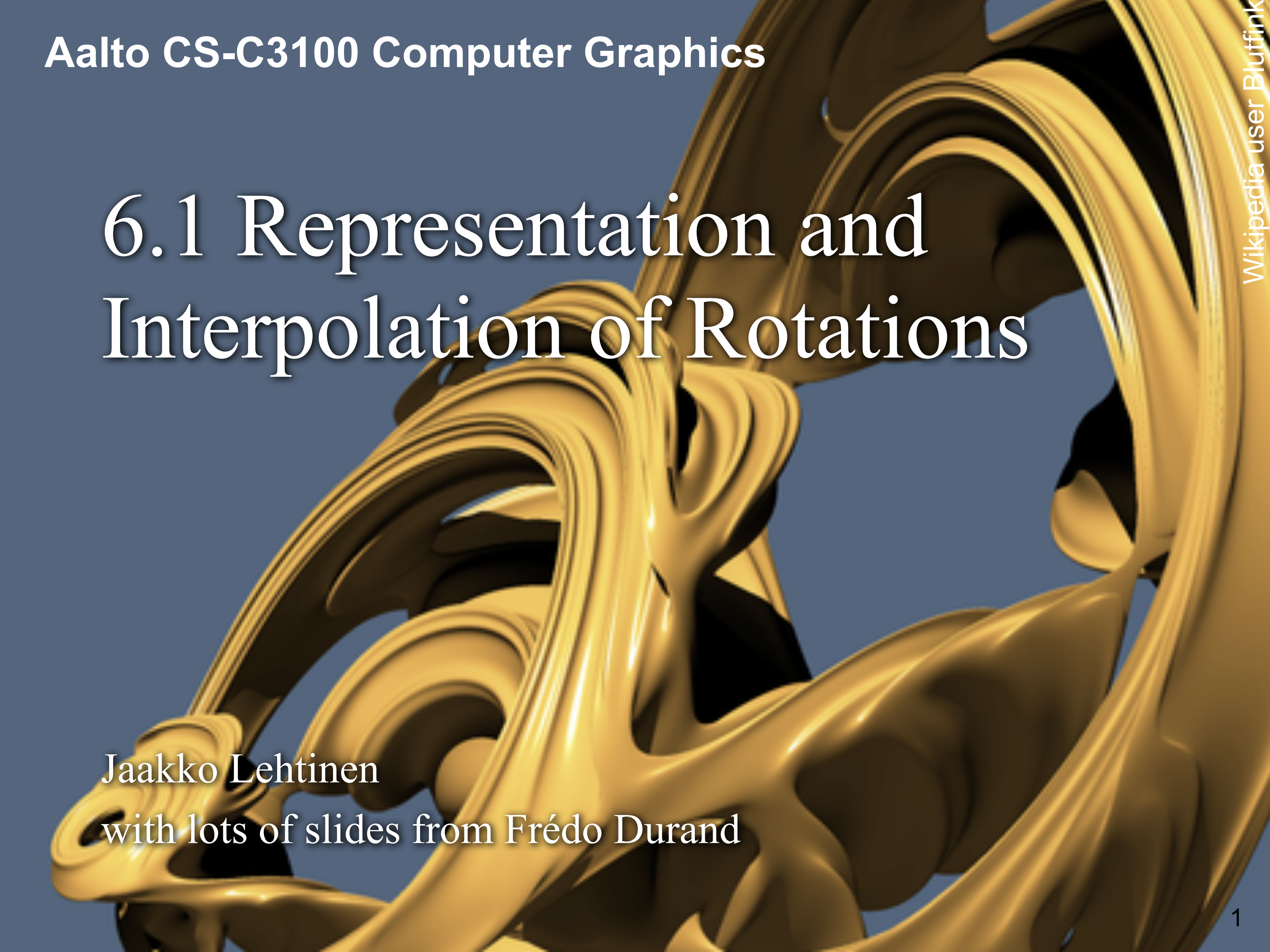
6.1 Representation and Interpolation of Rotations



Jaakko Lehtinen
with lots of slides from Frédo Durand

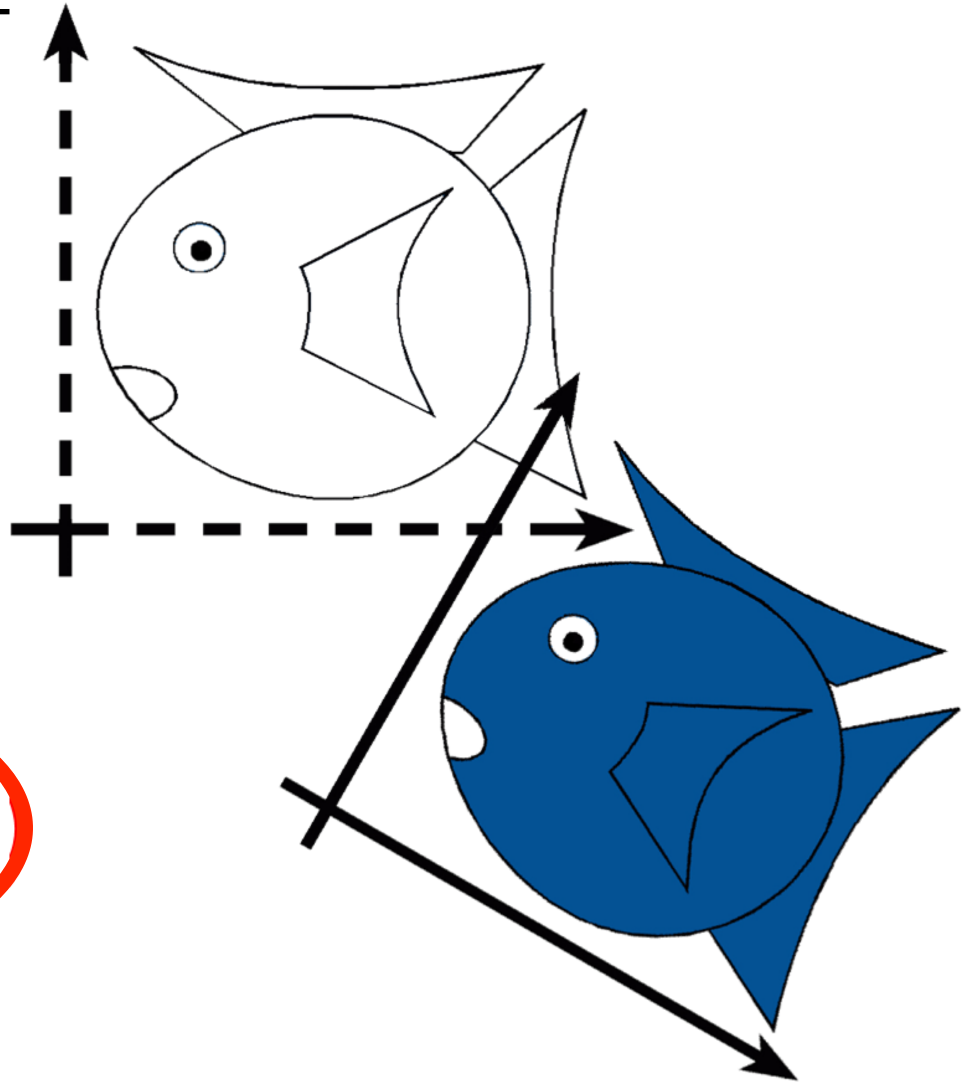
In This Video

- What is a rotation?
- Some simple rotation representations
  - 3x3 orthogonal matrix
  - axis-angle (“exponential map”)
  - Euler angles
  - Limitations

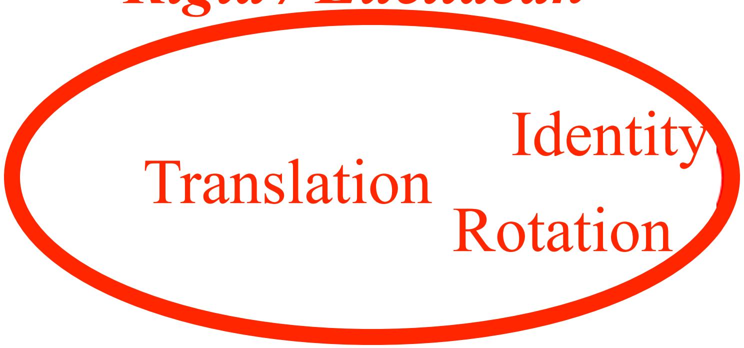
- Extra material: correct interpolation using the axis-angle representation

Orientations are Everywhere

- Euclidean transforms
  - Preserves distances
  - Preserves angles



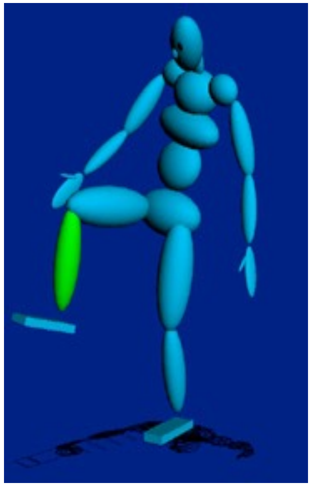
Rigid / Euclidean



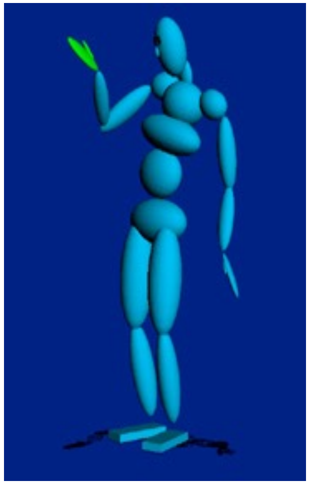
Orientations are Everywhere

- In an articulated character,
  each joint is characterized by its degrees of freedom (dof)
  - Usually rotation about one, two or three axes

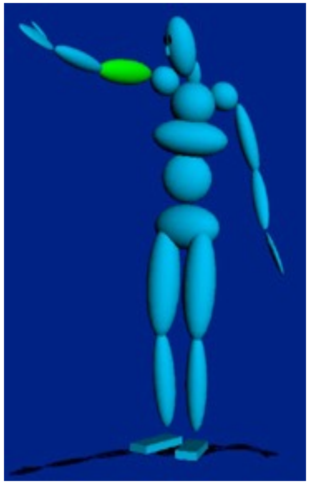
1 DOF: knee



2 DOF: wrist



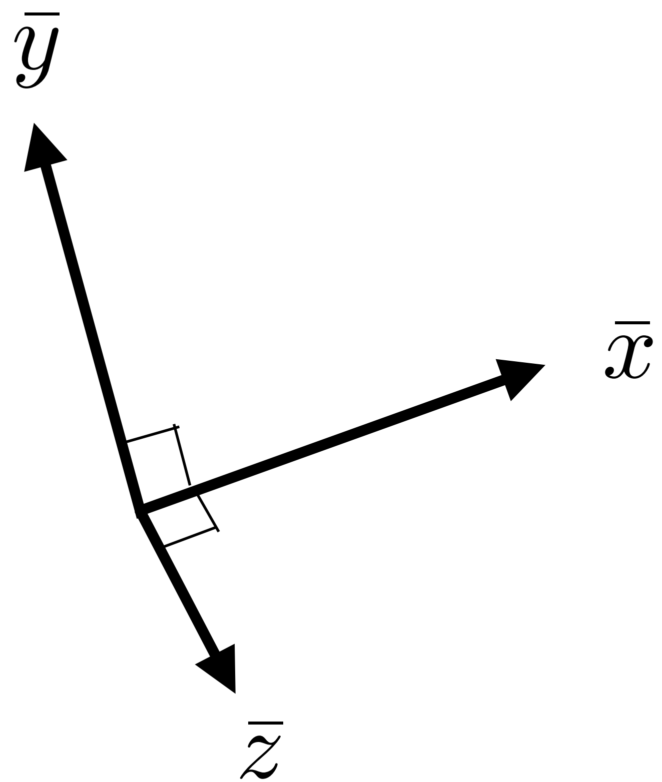
3 DOF: arm



What is an Orientation?

What is an Orientation?

- Most intuitive: orthonormal coordinate system



Orthonormality

$$\|\bar{x}\| = \|\bar{y}\| = \|\bar{z}\| = 1$$

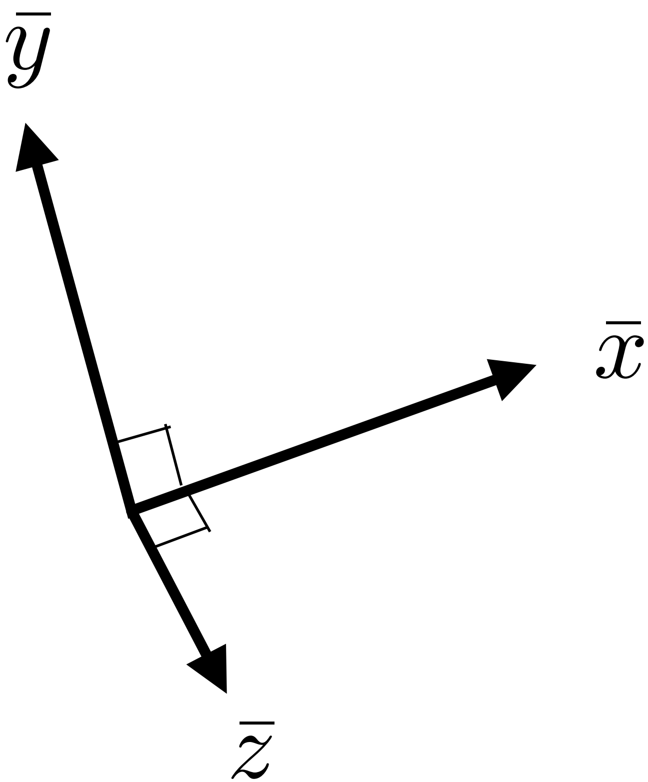
$$\bar{x} \cdot \bar{y} = 0$$

$$\bar{x} \cdot \bar{z} = 0$$

$$\bar{y} \cdot \bar{z} = 0$$

What is an Orientation?

- Most intuitive: orthonormal coordinate system



Put axes as columns in 3x3 matrix:

$$\mathbf{M} = \begin{pmatrix} \vdots & \vdots & \vdots \\ \bar{x} & \bar{y} & \bar{z} \\ \vdots & \vdots & \vdots \end{pmatrix}$$

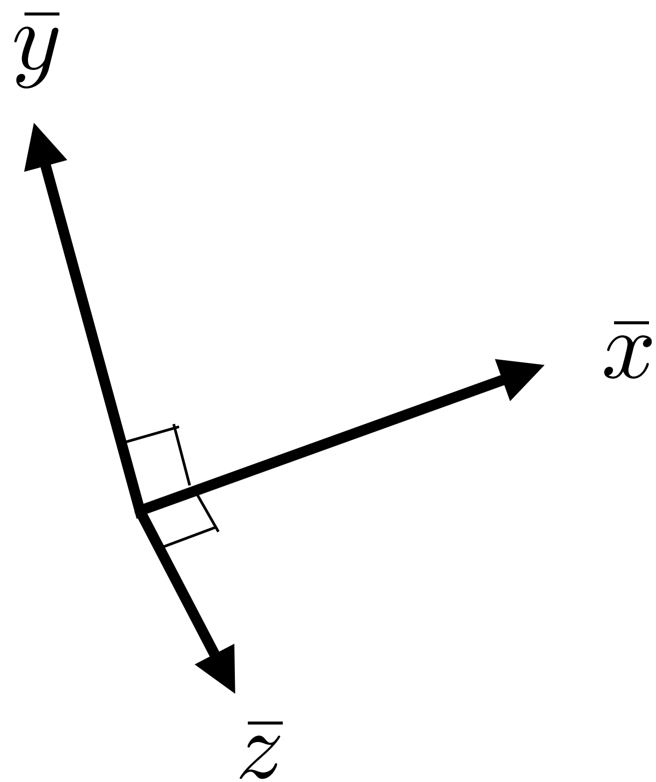
then orthogonality

$\Leftrightarrow$

....?

What is an Orientation?

- Most intuitive: orthonormal coordinate system



Put axes as columns in 3x3 matrix:

$$\mathbf{M} = \begin{pmatrix} \vdots & \vdots & \vdots \\ \bar{x} & \bar{y} & \bar{z} \\ \vdots & \vdots & \vdots \end{pmatrix}$$

then orthogonality

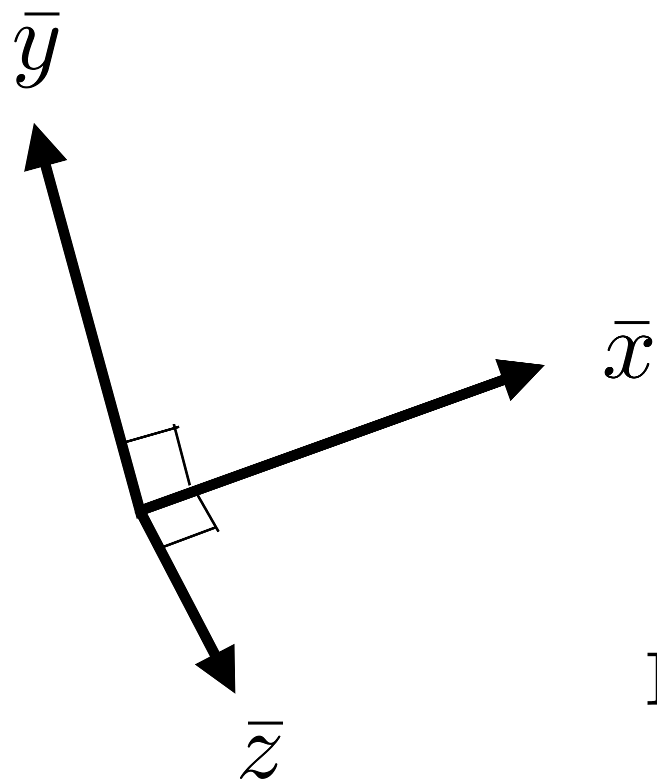
$$\Leftrightarrow$$

$$\mathbf{M}^{\mathrm{T}}\mathbf{M} = \mathbf{I}$$

Why?

What is an Orientation?

- Most intuitive: orthonormal coordinate system



Why:

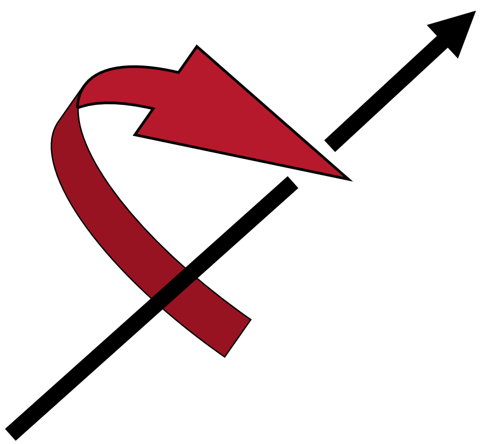
$$\mathbf{M} = \begin{pmatrix} \vdots & \vdots & \vdots \\ \bar{x} & \bar{y} & \bar{z} \\ \vdots & \vdots & \vdots \end{pmatrix}$$

$$\mathbf{M}^{\mathrm{T}}\mathbf{M} = \begin{pmatrix} \bar{x}\cdot\bar{x} & \bar{x}\cdot\bar{y} & \bar{x}\cdot\bar{z} \\ \bar{y}\cdot\bar{x} & \bar{y}\cdot\bar{y} & \bar{y}\cdot\bar{z} \\ \bar{z}\cdot\bar{x} & \bar{z}\cdot\bar{y} & \bar{z}\cdot\bar{z} \end{pmatrix}$$

But also: Orientation is Rotation

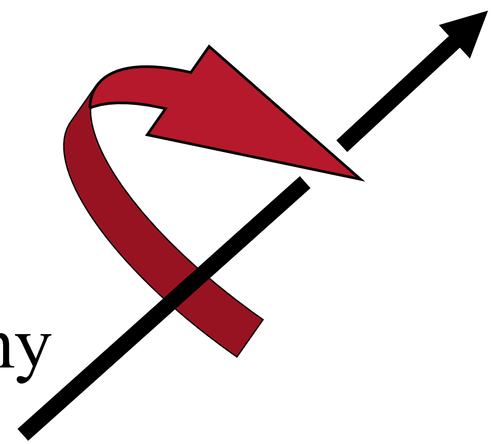
Rotation is Orientation

- Euler's Rotation Theorem: **All pairs of 3D orthogonal (Cartesian) coordinate systems that share a common origin are related through a rotation about some fixed axis.**
  - In other words, you can orient any orthogonal coordinate frame with any other using a rotation.



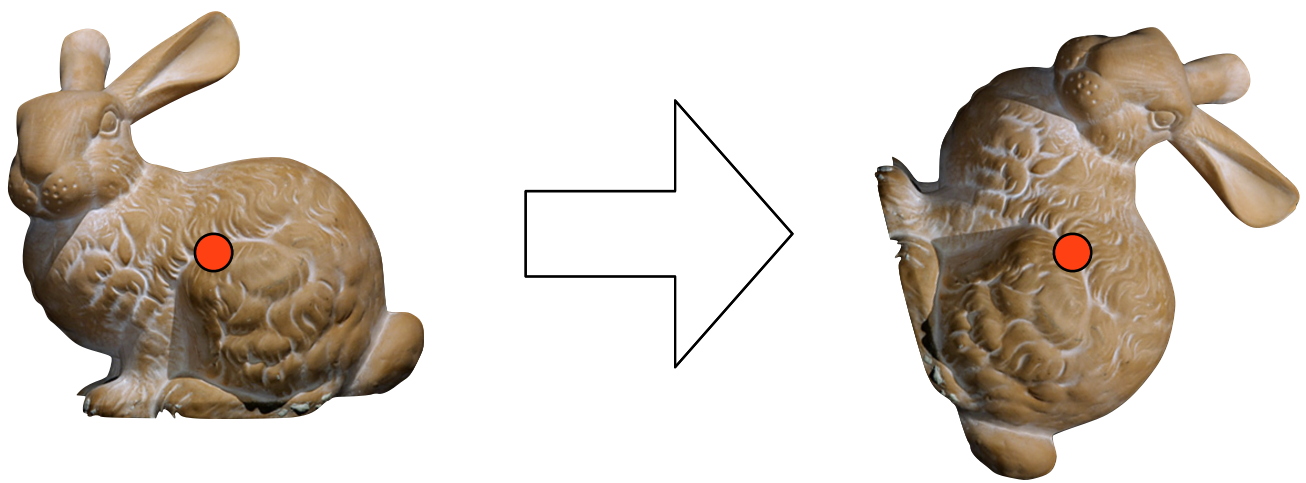
Rotation is Orientation

- Euler's Rotation Theorem: **All pairs of 3D orthogonal (Cartesian) coordinate systems that share a common origin are related through a rotation about some fixed axis.**
  - In other words, you can orient any orthogonal coordinate frame with any other using a rotation.
- Consequence:
  Orientation is really the same as rotation
  - This is because you can get to any orientation from the identity transform using a rotation.



Plane Rotations

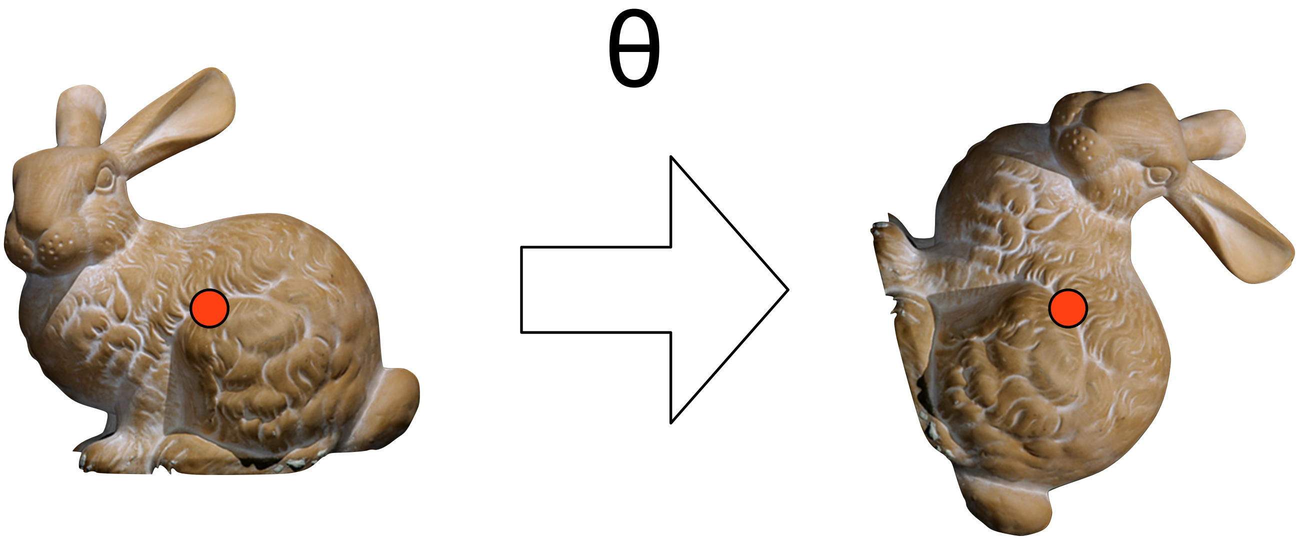
- How many degrees of freedom?



(origin really stays fixed)

2D (Plane) Rotations

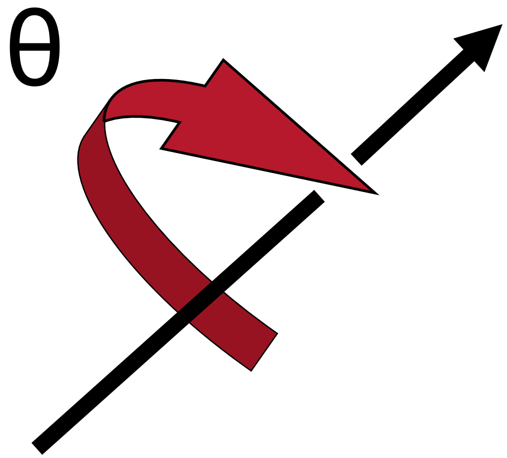
- How many degrees of freedom?



1 DOF, just one rotation angle

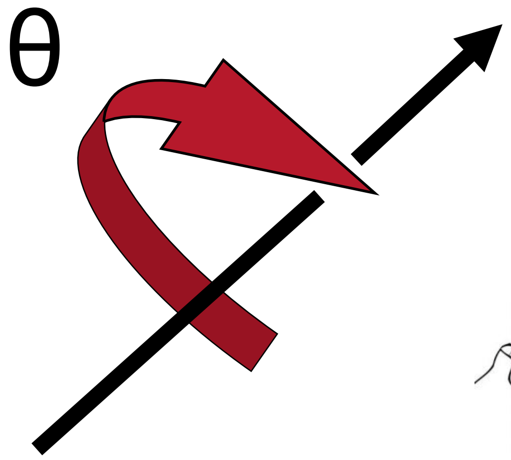
3D Rotations

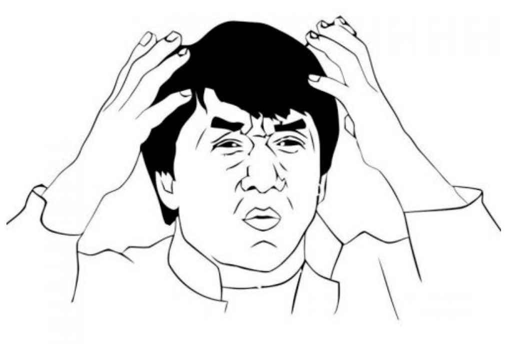
- How many degrees of freedom?



3D Rotations

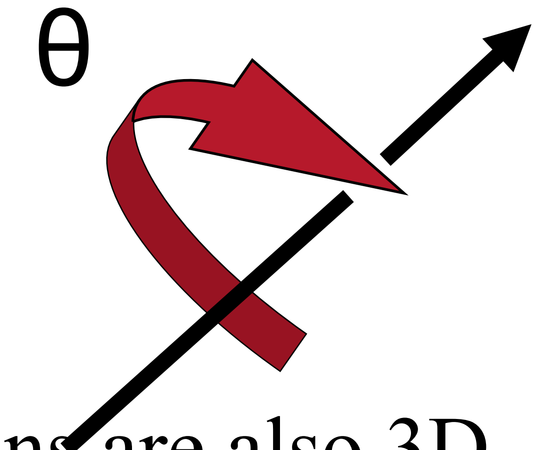
- How many degrees of freedom?
- 3 degrees of freedom (!? 2D only had 1…)
  - direction of rotation (2D) and angle (1D)
  - Only have to care for angle $0 < \theta < \pi$
    - Why? When over π, negate axis, take angle $2\pi - \theta$





3D Rotations

- How many degrees of freedom?
- 3 degrees of freedom
  - direction of rotation (2D) and angle (1D)
  - Only have to care for angle $0 < \theta < \pi$
    - Why? When over π, negate axis, take angle $2\pi - \theta$



- Because orientations and rotations are basically the same, this means orientations are also 3D

What is a Rotation?

- Axis-angle view (as above):
  Rotation about an axis **v** by angle θ
  - Can encode rotation in a 3D vector ("rotation vector")
    $\mathbf{r} = \theta\mathbf{v}$, where θ is the angle and **v** is a unit vector
    - Origin is identity, length of vector encodes angle
  - Points inside radius-π sphere are orientations
    - (Namedropping: "The exponential map")

What is a Rotation?

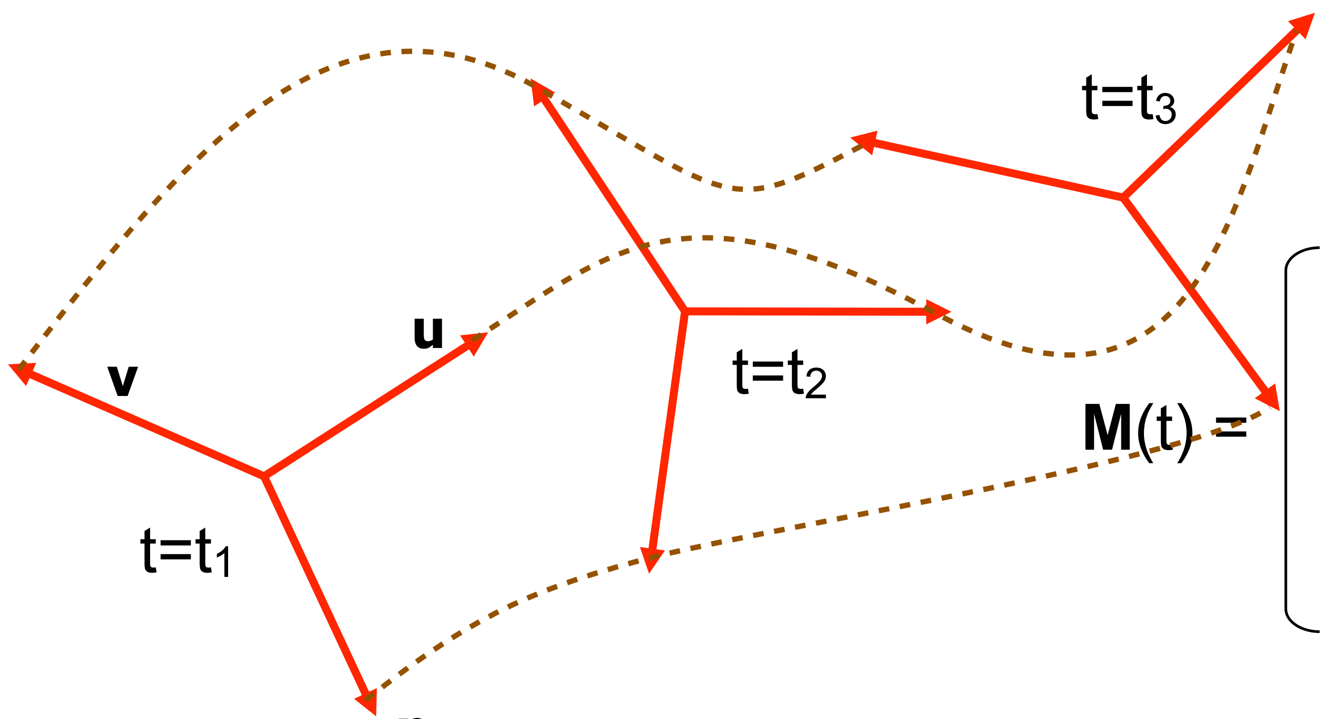
- Linear algebra view
  - Orthogonal matrix, $\mathbf{M}^T\mathbf{M} = \mathbf{I}$, $\det(\mathbf{M}) = 1$
    - Determinant condition rules out reflections
  - In other words, **M** has orthonormal columns and rows, i.e., the **columns are basis vectors at right angles**
  - Count the degrees of freedom!

What is a Rotation?

- Linear algebra view
  - Orthogonal matrix, $\mathbf{M}^T\mathbf{M} = \mathbf{I}$, $\det(\mathbf{M}) = 1$
    - Determinant condition rules out reflections
  - In other words, **M** has orthonormal columns and rows, i.e., the **columns are basis vectors at right angles**
  - **Overcomplete representation**: **M** has more than 3 entries, meaning that *not all matrices are proper rotations* (well, duh)

Interpolating Orientations in 3D

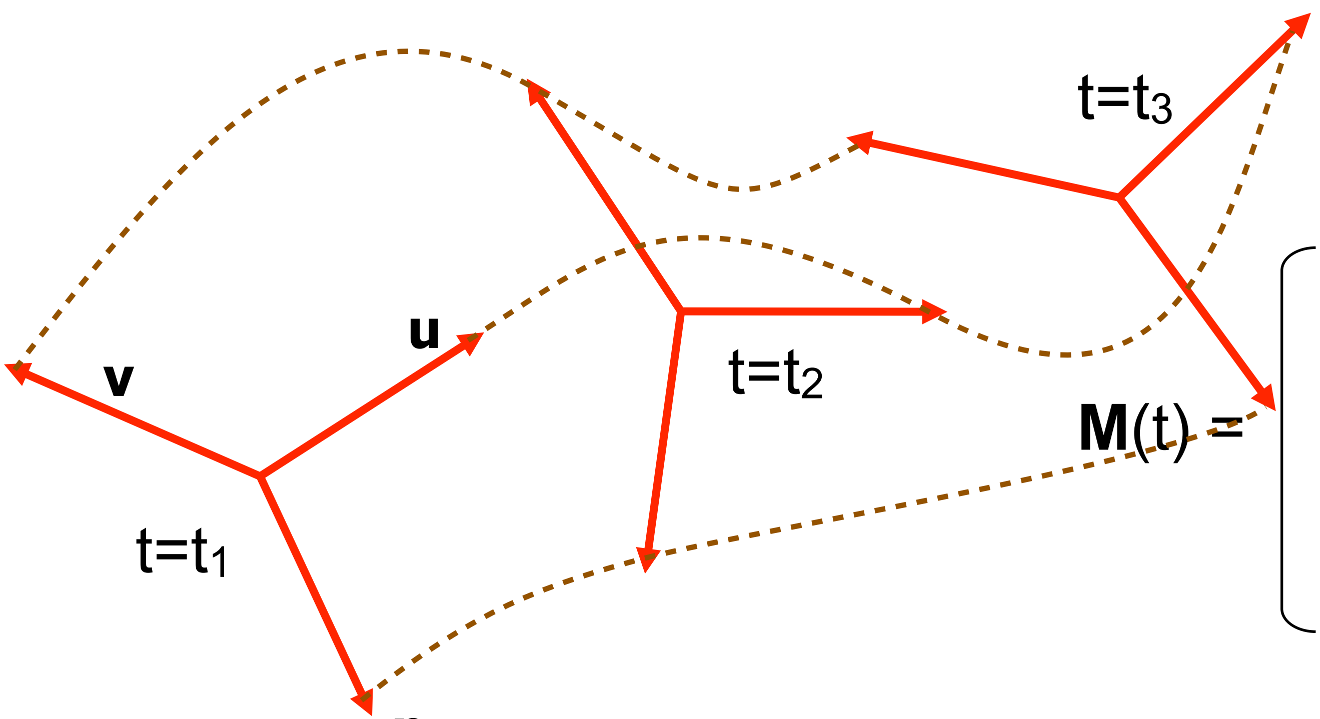
- Critical for animation: Given rotation matrices $\mathbf{M}_i$ and time t_i, find $\mathbf{M}(t)$ such that $\mathbf{M}(t_i)=\mathbf{M}_i$
- Problem reduces to question:
  "How do you morph between two rotations?"



$$\mathbf{M}(t) = \begin{pmatrix} u_x(t) & u_y(t) & u_z(t) \\ v_x(t) & v_y(t) & v_z(t) \\ n_x(t) & n_y(t) & n_z(t) \end{pmatrix}$$

Interpolating Orientations in 3D

- Critical for animation: Given rotation matrices $\mathbf{M}_i$ and time t_i, find $\mathbf{M}(t)$ such that $\mathbf{M}(t_i)=\mathbf{M}_i$
- Problem reduces to question:
  "How do you morph between two rotations?"



Interpolating positions is easy, just use splines!

$$\mathbf{M}(t) = \begin{bmatrix} u_x(t) & u_y(t) & u_z(t) \\ v_x(t) & v_y(t) & v_z(t) \\ n_x(t) & n_y(t) & n_z(t) \end{bmatrix}$$

First Try

- Interpolate each matrix entry independently
- Example: $\mathbf{M}_0$ is identity and $\mathbf{M}_1$ is 90 degrees around x-axis

$$\text{Interpolate}\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 1\\ 0 & -1 & 0\end{bmatrix}\right) = \begin{bmatrix}1 & 0 & 0\\ 0 & 0.5 & 0.5\\ 0 & -0.5 & 0.5\end{bmatrix}$$

- Is the result a rotation matrix?

First Try

- Interpolate each matrix entry independently
- Example: $\mathbf{M}_0$ is identity and $\mathbf{M}_1$ is 90 degrees around x-axis

$$\text{Interpolate}\left(\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}, \begin{bmatrix}1 & 0 & 0\\0 & 0 & 1\\0 & -1 & 0\end{bmatrix}\right) = \begin{bmatrix}1 & 0 & 0\\0 & 0.5 & 0.5\\0 & -0.5 & 0.5\end{bmatrix}$$

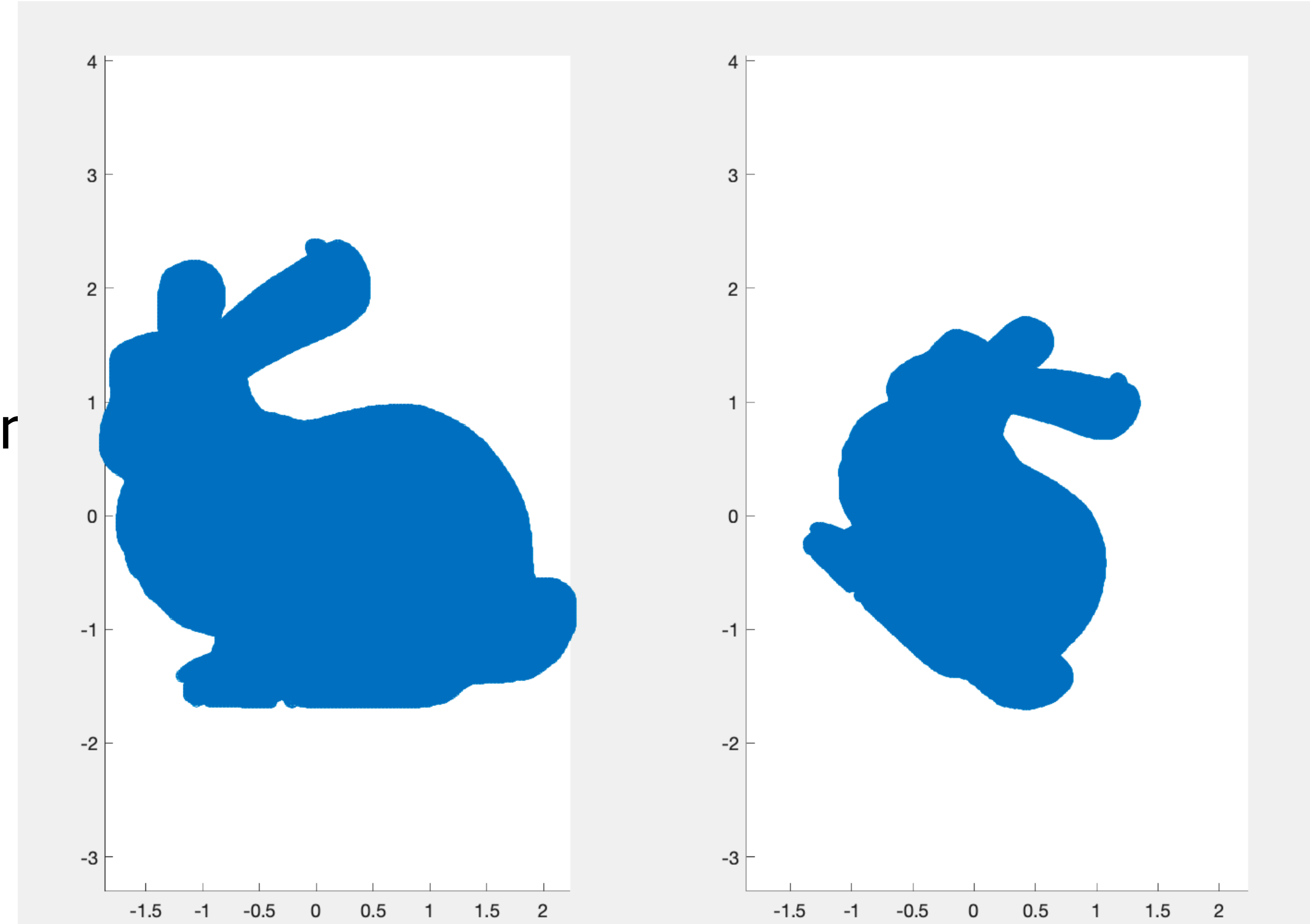
- Is the result a rotation matrix?
  - **No, it does not preserve rigidity (angles and lengths) – what *does* it do?**

(Both rotates and scales)

-
-

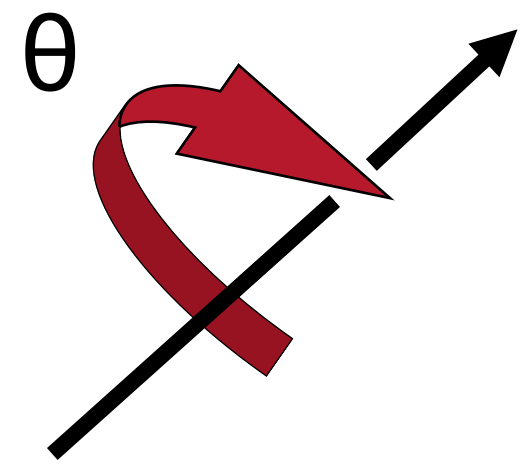
Ir

-



Are Rotations Simply Vectors Then?

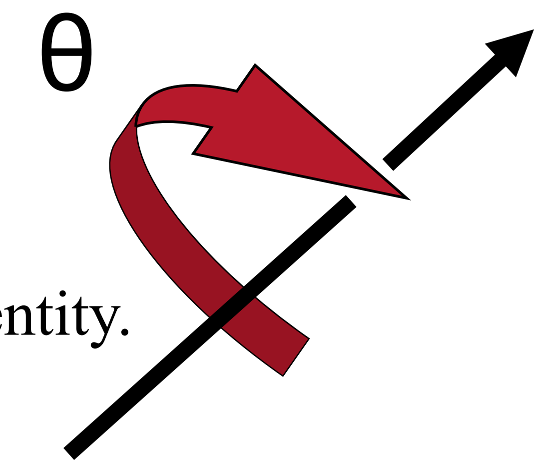
- Axis-angle: Rotation about an axis (3 DOF)
- Can encode rotation in one 3D vector
  $\mathbf{r} = \theta\mathbf{v}$, where θ is the angle and $\mathbf{v}$ is a unit vector
  - Origin is identity
- All good?



- Let's think about
  - $\theta\mathbf{v}$, $(\theta+n2\pi)\mathbf{v}$, $(2\pi-\theta)(-\mathbf{v})$?
  - $n2\pi\mathbf{v}$, in particular $r = 2\pi\mathbf{v}$?

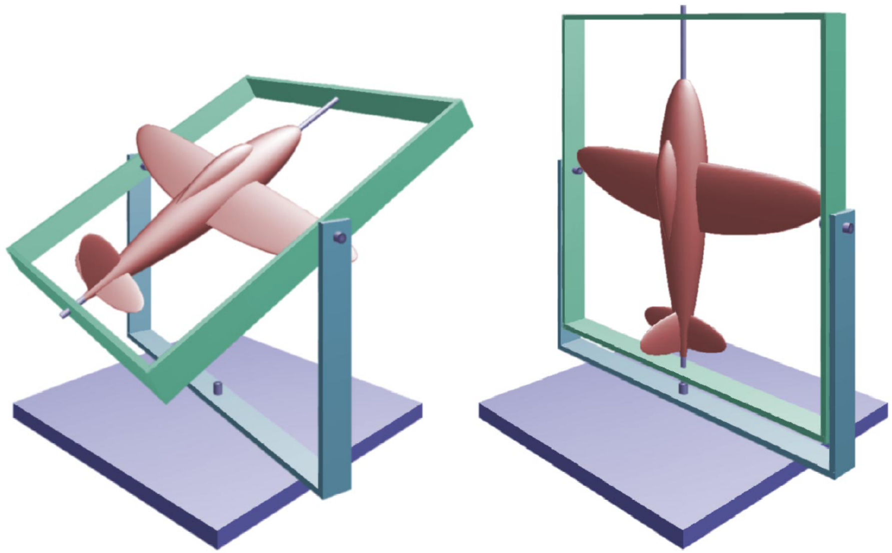
Are Rotations Simply Vectors Then?

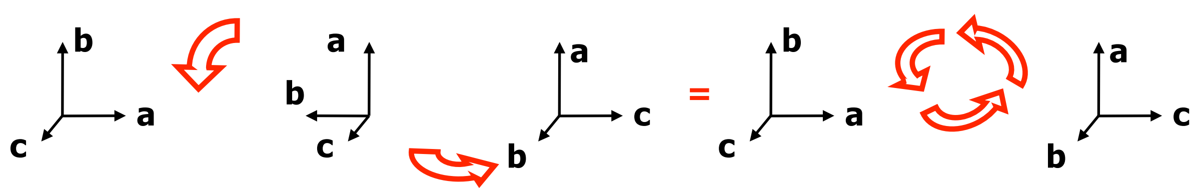
- Let's think about
  - $\theta\mathbf{v}$, $(\theta+n2\pi)\mathbf{v}$, $(2\pi-\theta)(-\mathbf{v})$?
  - $n2\pi\mathbf{v}$, in particular $r = 2\pi\mathbf{v}$?
  - **There are infinitely many 3D axis-angle vectors that correspond to the same rotation.**
    - E.g., the whole ball $|\mathbf{r}| = 2\pi$ is the identity.
  - Things are relatively OK if we stay within the sphere of radius π.



Another Try: Rotation Angles

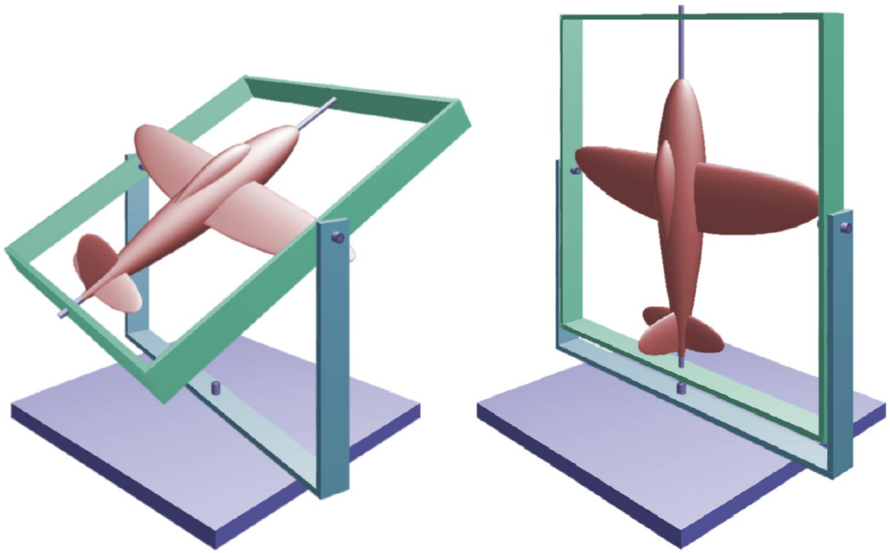
- "Euler angles" are sequential rotations about a single coordinate (e.g.) in the sequence Z-Y-Z
  - Corresponds to a gimbal
  - Such a sequence of 3 can get you to any orientation
- Can also use a sequence of rotations around X-Y-Z
  - Roll, pitch and yaw (perfect for flight simulation)



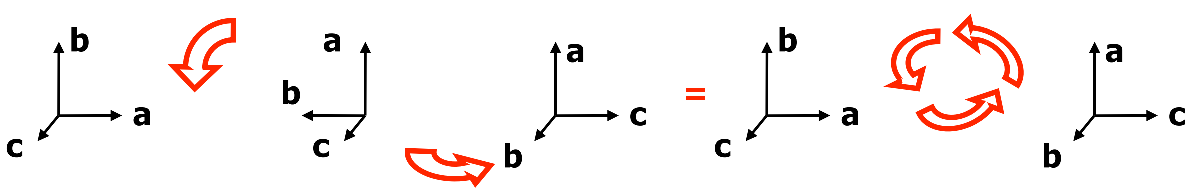


Another Try: Rotation Angles

- Problems:
  - Bad interpolation
  - Gimbal lock

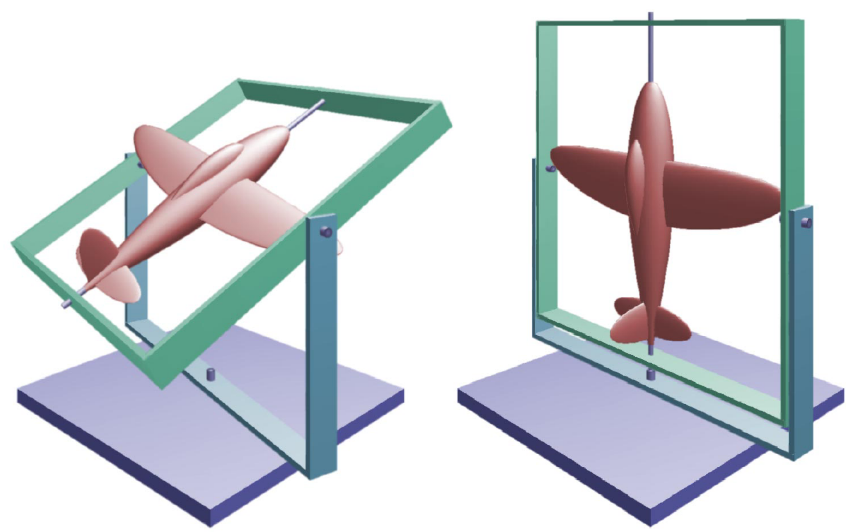


http://www.fho-emden.de/~hoffmann/gimbal09082002.pdf

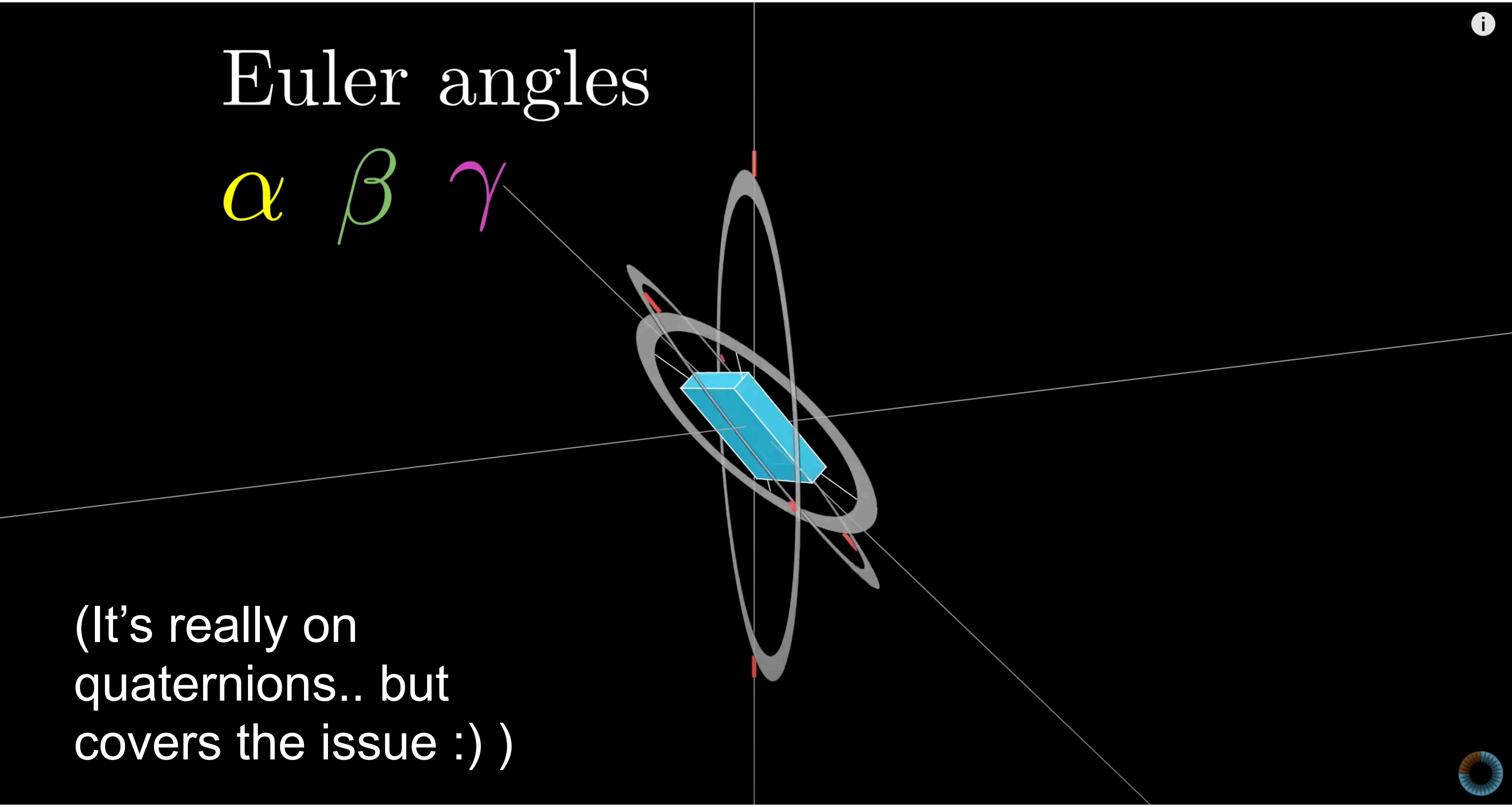


Gimbal Lock

- Two or more axis align resulting in a loss of rotation degrees of freedom.
  - http://en.wikipedia.org/wiki/Euler_angles
  - http://en.wikipedia.org/wiki/Gimbal_lock



(See Great 3Blue1Brown video)



Fundamental Problem

- The space of rotations ("the rotation group") is not Euclidean
  - Increasing rotation angle ends up where you started
  - (Buzzword: "The topology of the rotation group is that of the projective space RP^3")

Fundamental Problem

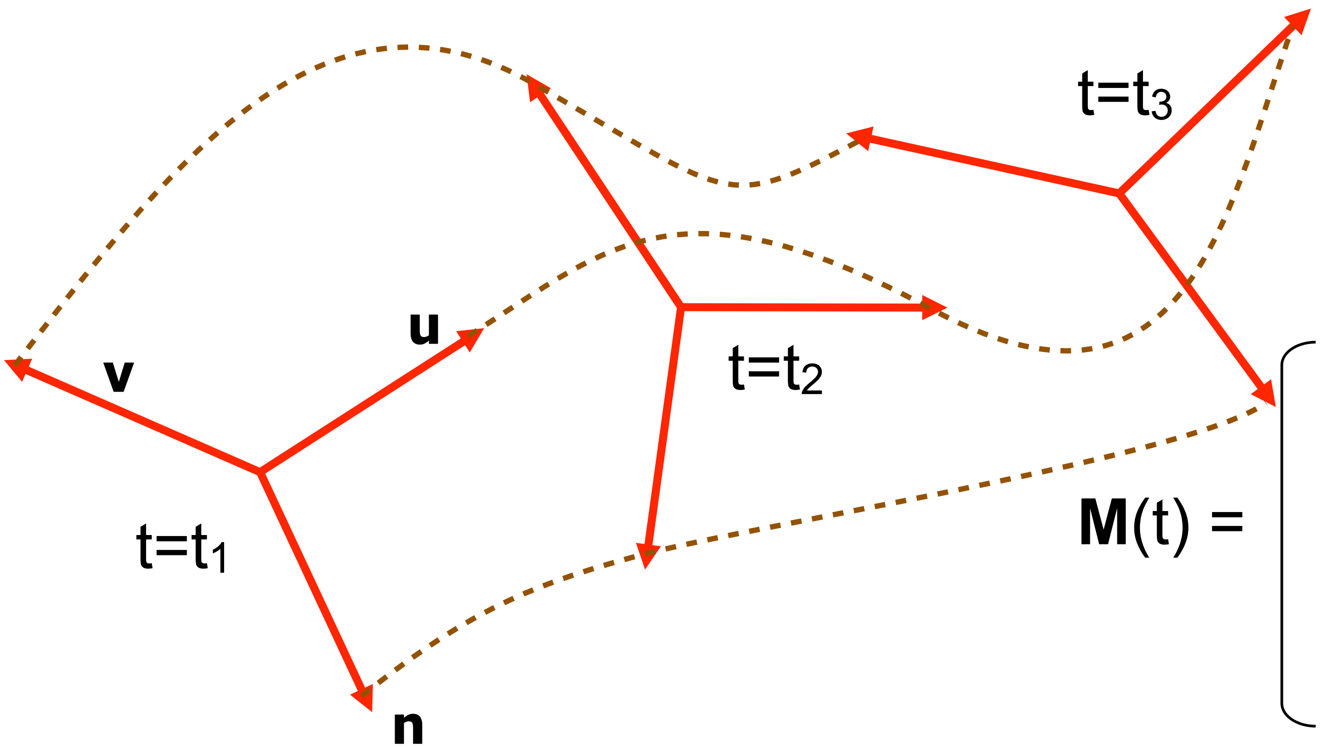
- Even though the space is 3D, rotations cannot be represented in R^3 without kinks or multiple-valuedness
  - **Euler angles are really really nasty** (gimbal lock, interpolation, no easy composition)
  - Axis-angle is multiple-valued, doesn't interpolate nicely without special considerations, cannot be easily composed
  - 3x3 matrices are redundant and don't interpolate nicely, but can be composed easily (matrix multiplication)

What to Do About It?

- Next video: quaternions!

Extra: Correct Interpolation Using Axis-Angle

- Problem reduces to question:
  "How do you morph between two rotations?"



$$\mathbf{M}(t) = \begin{pmatrix} u_x(t) & u_y(t) & u_z(t) \\ v_x(t) & v_y(t) & v_z(t) \\ n_x(t) & n_y(t) & n_z(t) \end{pmatrix}$$

Desirable Properties for Interpolation

- Remember Euler's theorem:
  All orientations **A**, **B** are related by a rotation **R** around an axis **v** by an angle θ: $\mathbf{B} = \mathbf{AR}$.
  - **R** can be represented as $\mathbf{r} = \theta\mathbf{v}$

- Interpolated orientation should rotate around **v** with constant speed, starting from zero angle.

Recipe for Axis-Angle Interpolation

- Axis-angle interpolation how-to:
  - Compute **R** as $\mathbf{A}^{-1}\mathbf{B}$ (then, clearly: $\mathbf{B}=\mathbf{AR}$)
  - Get axis-angle representation from **R** (how to: soon)
  - Interpolate linearly between zero and **r** to yield **r**(t), i.e., $\mathbf{r}(t) = t\mathbf{r}$
  - Convert **r**(t) back to matrix **R**(t)
  - Get final orientation by computing **AR**(t)

Interpolation Using Axis-Angle

- This works because the interpolated rotation vector starts at zero (identity) and always points to the same direction: $\mathbf{r}(t) = t\mathbf{r}$, with t a scalar
  - equivalently, the axis stays fixed and the angle changes linearly
  - that is, in the local coordinate system of **A**, the rotation axis direction stays fixed, only angle changes.
- However if you represent them as vectors rA and rB using axis-angle, and interpolate the corresponding 3D vectors linearly, this does NOT yield the correct result.

Interpolation Using Axis-Angle

- Careful: if you represent rotations **A** and **B** as rotation vectors **r**a and **r**b and interpolate them linearly, you do **not** get the correct result.
  - It interpolates the orientations, but not at unit speed.
  - (Why not? It's kind of deep. Let's not go there.)

- What our recipe does instead: axis-angle interpolation of the **relative rotation** between A and B

Axis-angle to Matrix

- Given unit axis **v**, angle θ, compute rotation matrix **R**

$$\boldsymbol{v}_+ = \begin{pmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{pmatrix}$$

This matrix corresponds to cross product with **v**.

$$\boxed{\boldsymbol{R} = \boldsymbol{I}\,\cos\theta + (1-\cos\theta)\boldsymbol{v}\boldsymbol{v}^{\mathrm{T}} - \boldsymbol{v}_+ \sin\theta}$$

Outer product

If represented as **r**=θ**v**, must normalize and compute length first, and watch out for zeros

Matrix to Axis-Angle

- Given rotation matrix **R**, compute axis **v** and angle θ

$$
\begin{aligned}
\theta &= \cos^{-1}((R_{11} + R_{22} + R_{33} - 1)/2) \\
v_1 &= (R_{32} - R_{23})/(2\sin\theta) \\
v_2 &= (R_{13} - R_{31})/(2\sin\theta) \\
v_3 &= (R_{21} - R_{12})/(2\sin\theta)
\end{aligned}
$$

Recap

- You can't identify orientations/rotations with 3D points and have all of
  - Nice interpolation
  - Smoothness (no gimbal lock)
  - Simple composition of rotations
  - No complicated multiple-valuedness

  - (You can, however, interpolate **relative** changes in orientation using the axis-angle representation.)
- **Next up**: **Quaternions**, a 4D construction that does exactly what we want