FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

Lecture #6

Affine 3D Transformations – 3D Rotations

Computer Graphics

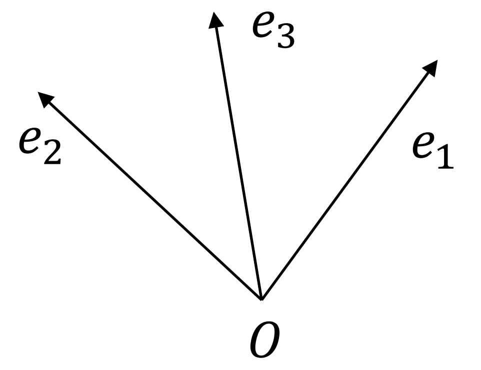
Winter Term 2016/17

Marc Stamminger / Roberto Grosso

Affine Transformations in 3D

- very similar to 2D
- three unit vectors:

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots \\ e_1 & e_2 & e_3 \\ \vdots & \vdots & \vdots \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}$$



- In homogeneous coordinates:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w' \end{pmatrix} = \begin{pmatrix} \vdots & \vdots & \vdots & t_1 \\ e_1 & e_2 & e_3 & t_2 \\ \vdots & \vdots & \vdots & t_3 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

Affine Transformations in 3D

- Basic 3D transformations
  - Translation: $translate(d_x, d_y, d_z) = \begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$
  - Scaling: $scale(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
  - Z-Shear: $shear_z(d_x, d_y) = \begin{pmatrix} 1 & 0 & d_x & 0 \\ 0 & 1 & d_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Affine Transformations in 3D

- Rotation around the x-, y- and z-axis

  - $Rot_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

  - $Rot_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi\; 0 & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

  - $Rot_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- But how do we describe arbitrary rotations ?

Rotations in 3D

- For now, we only rotate around the origin
  → a 3x3 matrix is sufficient
- The columns of a rotation matrix are the unit vectors after rotation:

$$R\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix}\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

- Here u, v, w are the main axes after the rotation

- For rotation matrices, the inverse is simply the transpose:

$$R^{-1} = R^T$$

Rotations in 3D

- The description of 3D rotations is a core problem in computer graphics
  - Positioning objects in the world
  - Animating objects (= interpolating rotations)
  - Modeling camera animations
  - …
- Two important questions:
  - how to describe a rotation ?
  - how to interpolate rotations ?
    → Some representations result in awkward interpolation

Rotations in 3D

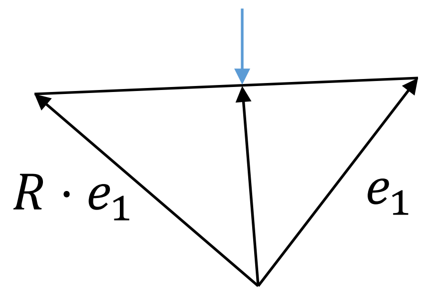
- How to specify rotations in 3D
  - Orthogonal matrices
  - 3 Euler rotations, e.g.
    - Rotz $\rightarrow$ Rotx $\rightarrow$ Rotz
    - Rotz $\rightarrow$ Roty $\rightarrow$ Rotz
    - Rotx $\rightarrow$ Roty $\rightarrow$ Rotz
  - Axis of rotation and angle
  - Quaternions
- Etc, e.g. 2 (planar) reflections

Rotations in 3D

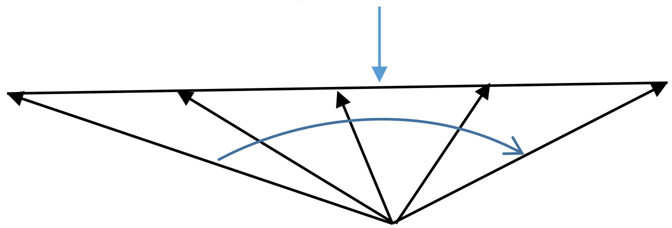
FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- Orthonormal matrices
  - 9 degrees of freedom for matrix, 6 of which are fixed by constraints
  - Not very intuitive (user interface?)
  - Interpolation
    - Linear interpolation
      → interpolation of unit vectors
    - Requires renormalization
    - Non-uniform animation
      → see later: slerp-interpolation
    - Impossible for 180° rotations

$$\alpha R \cdot e_1 + (1-\alpha) \cdot e_1$$



linear interpolation on this line



non-uniform in angular space

Rotations in 3D

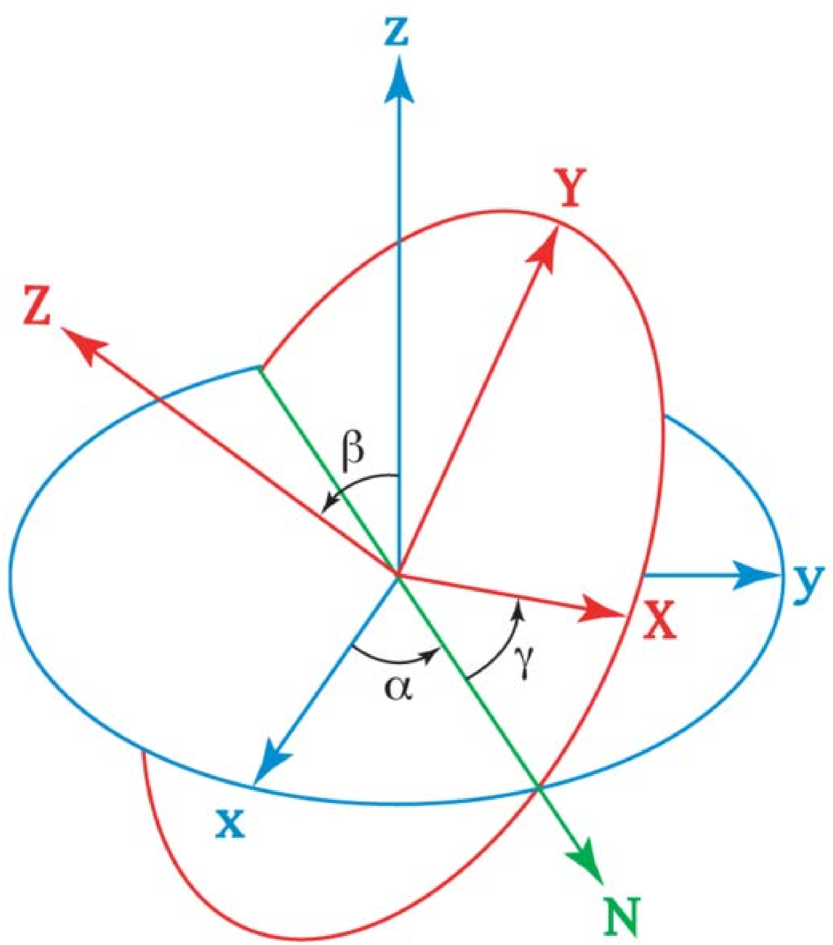
- Euler angles
  - Any rotation can be given by three rotations about the main axes, e.g. X, Y, and Z (Leonhard Euler 1707 – 1783)
  - If the rotations angles about Z, Y, and Z are ψ, θ, and ϕ respectively, then the rotation matrix is:

$$R = R_z(\phi)R_y(\theta)R_z(\psi)$$

$$= \begin{pmatrix} \cos\theta\cos\phi & \sin\psi\sin\theta\cos\phi - \cos\psi\sin\phi & \cos\psi\sin\theta\cos\phi + \sin\psi\sin\phi \\ \cos\theta\sin\phi & \sin\psi\sin\theta\sin\phi + \cos\psi\cos\phi & \cos\psi\sin\theta\sin\phi - \sin\psi\cos\phi \\ -\sin\theta & \sin\psi\cos\theta & \cos\psi\cos\theta \end{pmatrix}$$

Rotations in 3D

- Euler angles for the Euler rotation $z - x - z$ with angles $\alpha - \beta - \gamma$
- For given angles, the matrix can be computed as $R_z(\gamma)R_x(\beta)R_z(\alpha)$



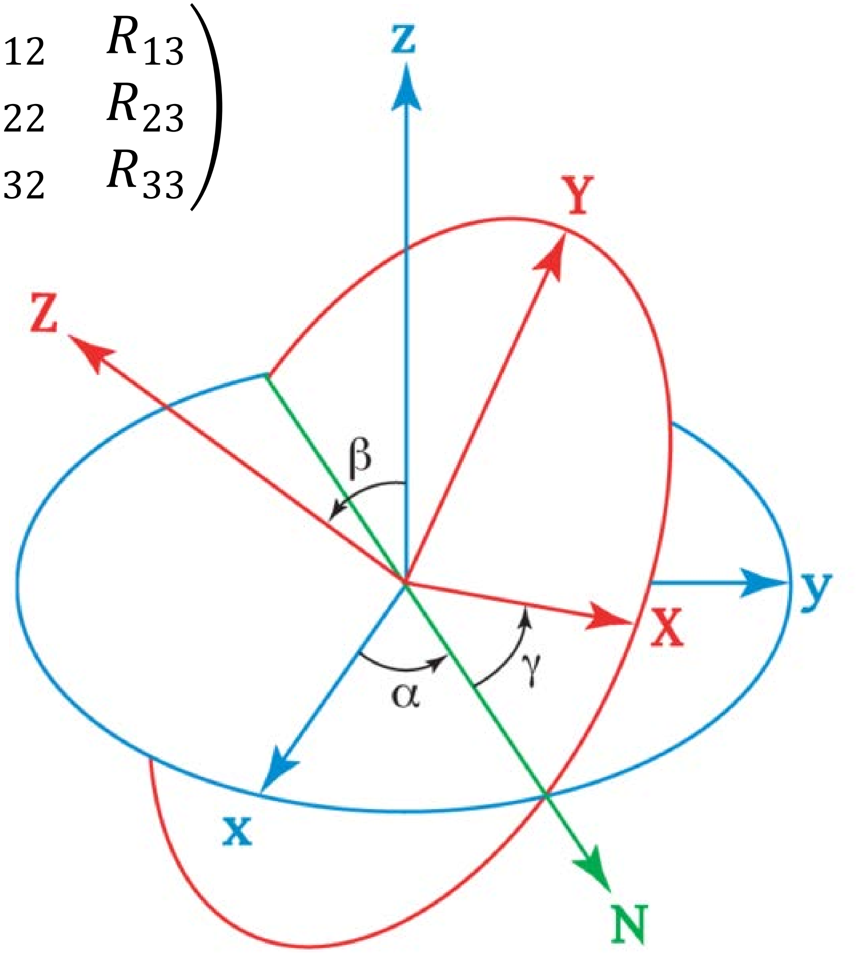
Rotations in 3D

- If the rotation matrix is given as:

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

  the angles can be determined accordingly:
  - $-\sin\beta = R_{31}$
  - $\tan\alpha = R_{32}/R_{33}$
  - $\tan\gamma = R_{21}/R_{11}$

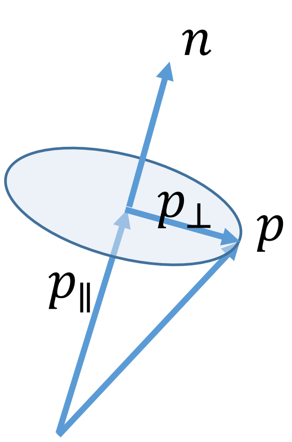
- for $R = R_z(\gamma)R_x(\beta)R_z(\alpha)$



Rotations in 3D

- Specify rotation by an axis n, $\|n\| = 1$, and a rotation angle ω

- Derivation: transform a point p
  - decompose p into parallel (to n) and orthogonal components:
    $p = p_{\parallel} + p_{\perp}$, where $p_{\parallel} = (n \circ p)n$ and $p_{\perp} = p - p_{\parallel}$
  - Create local coordinate system $\{p_{\perp}, n \times p, n\}$, where $n \times p = n \times p_{\perp}$
  - Rotate $p_{\perp}$ about n
    $$rot(p_{\perp}) = p_{\perp} \cos\omega + (n \times p)\sin\omega$$
  - add the parallel component
    $$rot(p) = p_{\perp} \cos\omega + (n \times p)\sin\omega + p_{\parallel}$$



Rotations in 3D

- Can we express this in a matrix?
- Rodrigues formula
  - $p_{\parallel} = (p \circ n)n = \cdots = \begin{pmatrix} n_x^2 & n_x n_y & n_x n_z \\ n_y n_x & n_y^2 & n_y n_z \\ n_z n_x & n_z n_y & n_z^2 \end{pmatrix} p = (n \cdot n^T)p = Pp$
    - $n \cdot n^T$ is called “outer product”
  - $n \times p$ can also be written as matrix:

$$n \times p = \begin{pmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{pmatrix} p = Qp$$

    - Q is called “skew symmetric” form of n

Rotations in 3D

- $rot(p) = p_{\perp} \cos\omega + (n \times p) \sin\omega + p_{\parallel}$
$$= \left(p - p_{\parallel}\right) cos\omega + \mathrm{Q\ p\ sin}\omega + p_{\parallel}$$
$$= (I - P) \cos\omega \cdot p + Q \sin\omega \cdot p + P \cdot p$$

- Then
$$R(\omega, n) = P + \cos\omega(1 - P) + \sin\omega\, Q$$

- Often used definition: scale rotation axis by rotation angle
  - Define rotation with an arbitrary vector w
  - Length of w is rotation angle, normalized w is axis

Rotations in 3D

- Reverse problem:
  Given an orthonormal matrix O, find rotation axis and angle.
- Points on the axis are not transformed, thus the axis is given by the eigenvector with eigenvalue 1 of O
- The rotation angle can be computed as:

$$trace(O) = \lambda_1 + \lambda_2 + \lambda_3 = 1 + 2\cos\omega$$

$$\rightarrow \omega = arc\cos\left(\frac{trace(O) - 1}{2}\right)$$

Rotations in 3D: Quaternions

- Remember: complex numbers add further, imaginary component to real number:

$$(\mathrm{x}, \mathrm{y}) = x + iy$$

- Addition, multiplication etc. can be defined on these such that they form a field (Körper), and we can use them almost like real numbers
- Multiplication with a unit length complex number $(\cos\omega, \sin\omega)$ is equivalent to a rotation by ω

Rotations in 3D: Quaternions

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- Quaternions carry this idea further and add three imaginary components:
$$(\mathrm{x}, \mathrm{y}) = x + i_1 y_1 + i_2 y_2 + i_3 y_3$$
  - Note: y is a 3D-vector
- Computing with quaternions:
$$(a, b) + (c, d) = (a + c, b + d)$$
$$(a, b) \cdot (c, d) = (ac - b \circ d, ad + bc + b \times d)$$
- Further operations:
  - Conjugate of a quaternion: $(a, b)^* = (a, -b)$
  - Norm of a quaternion $q = \sqrt{qq^*}$
  - Inverse $q^{-1} = \frac{\mathrm{q}^*}{\|q\|^2}$

Rotations in 3D: Quaternions

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG
TECHNISCHE FAKULTÄT

- Quaternions can be used to describe rotations in 3D, like complex numbers describe rotations in 2D:
  - Consider a unit length quaternion q
  - A rotation can be applied to a vector $v \in \mathbb{R}$^3
    - Transform v to a quaternion $v \rightarrow (0, v)$
    - Rotate using $q \cdot (0, v) \cdot q^{-1}$
    - Result will have real part 0, imaginary part is rotated vector!

$$v \rightarrow (0, v) \rightarrow q \cdot v \cdot q^{-1} \rightarrow rot(v)$$

    - Every rotation can be described by a quaternion and vice versa !

Rotations in 3D: Quaternions

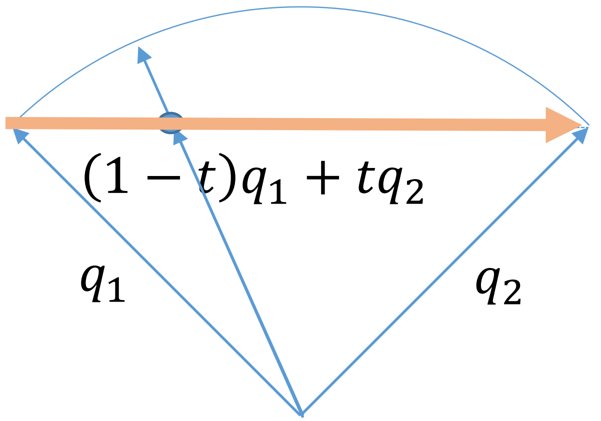
- Rotation about axis n by angle ω is expressed by the quaternion $q = (\cos\frac{\omega}{2}, \mathrm{n} \cdot \sin\frac{\omega}{2})$
- q and $-q$ describe the same rotation, otherwise the mapping is unique
  - Every rotation is represented by exactly two unit quaternions
  - Every unit quaternion describes a rotation

Rotations in 3D: Quaternions

- Application: Interpolation of rotations
  - Assume you have an object under rotation R_1 and want to animate it to a new rotation R_2
  - Interpolating matrices fails
  - Interpolating Euler angles will result in very weird movements
  - Interpolating quaternions works better…
  - … when using **spherical interpolation**

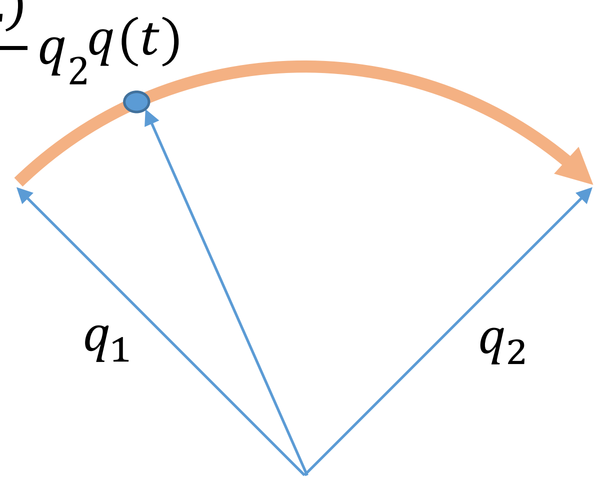
Rotations in 3D: Quaternions

- Linear interpolation of unit vectors requires renormalization
- Velocity is not uniform



- Instead, we should directly interpolate on the sphere
  -> spherical interpolation:

$$q(t) = \frac{\sin\left((1-t)\Omega\right)}{\sin\Omega}\mathrm{q}_1 + \frac{\sin(\mathrm{t}\Omega)}{\sin\Omega}q_2$$



often called “slerp”

Next Lecture

- Viewing and Perspective in 3D