

Computergrafik

Vorlesung im Wintersemester 2014/15
Kapitel 3: Transformationen und
homogene Koordinaten

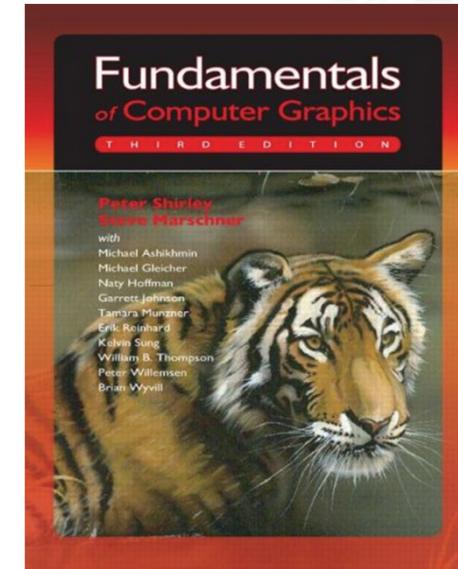
Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie



Literatur

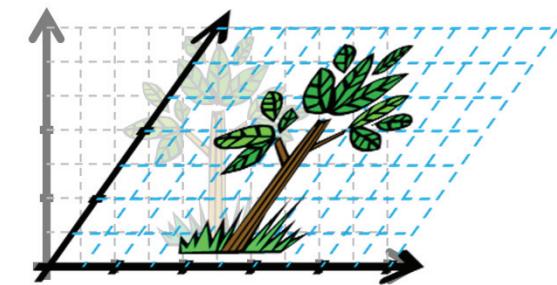
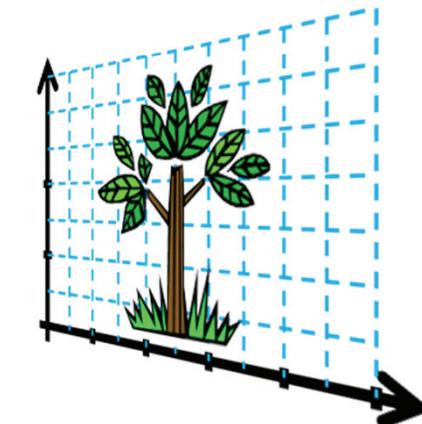
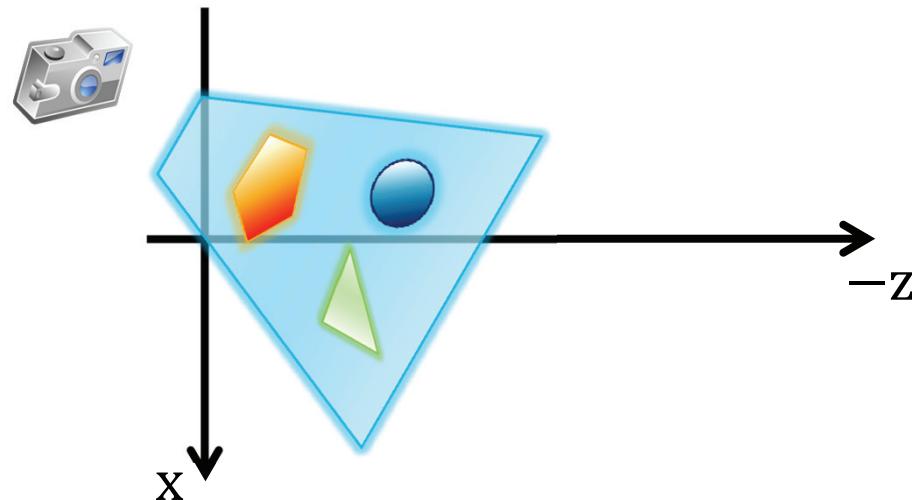


- ▶ **Fundamentals of Computer Graphics,**
P. Shirley, S. Marschner, 3rd Edition, AK Peters
→ Kapitel 5, 6 und 7



Inhalt

- ▶ 2D Transformationen
- ▶ 3D Transformationen
- ▶ Affine Abbildungen
- ▶ Homogene Koordinaten
- ▶ Transformation von Normalenvektoren
- ▶ Koordinatensysteme in der CG und hierarchische Modellierung

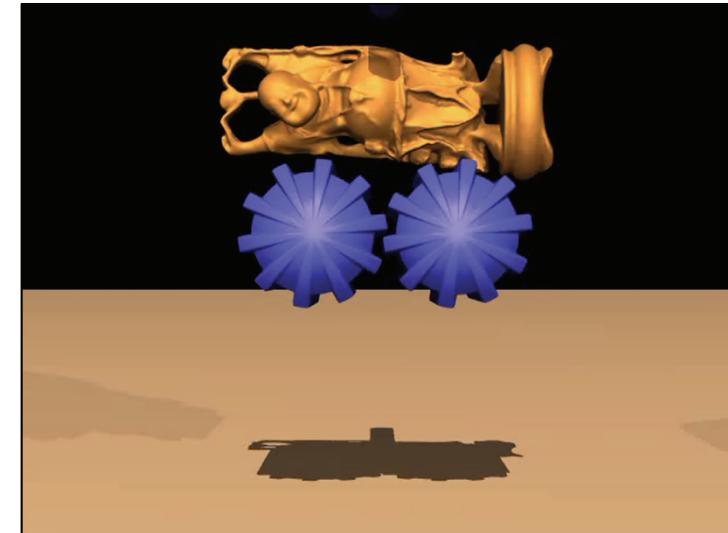


$$\begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix}$$

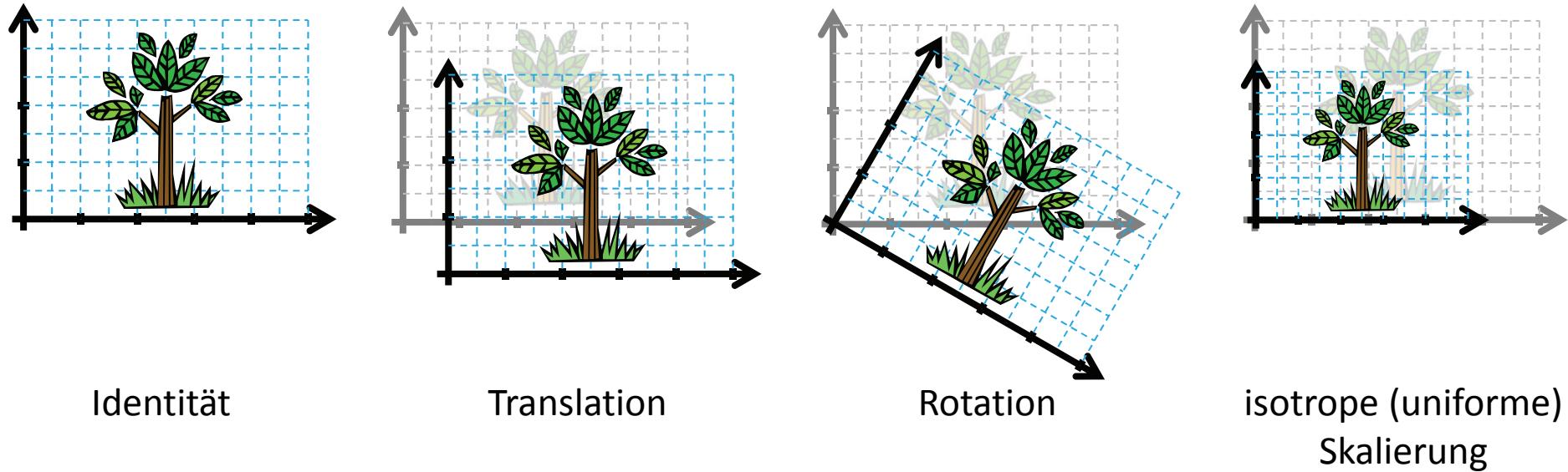
Was ist eine Transformation



- ▶ was ist eine Transformation?
 - ▶ eine Transformation bildet einen Punkt \mathbf{x} auf einen Punkt \mathbf{x}' ab
- ▶ eine lineare Transformation T ist eine Abbildung, die einen Punkt $\mathbf{x} \in \mathbb{R}^n$ auf einen Punkt $\mathbf{x}' \in \mathbb{R}^m$ abbildet: $T(\mathbf{x}) = \mathbf{A}\mathbf{x}$
 - ▶ $\mathbf{A} \in \mathbb{R}^{m \times n}$ ist die Transformationsmatrix von T
 - ▶ \mathbf{A} hat m Zeilen und n Spalten
- ▶ Beispiele
 - ▶ Platzierung von Objekten in einer Szene
 - ▶ Animation
 - ▶ Deformation
 - ▶ Kameratransformation und Projektion
 - ▶ aber auch:
 - ▶ Echtzeit-Schattenverfahren
 - ▶ Spiegelungen, usw.



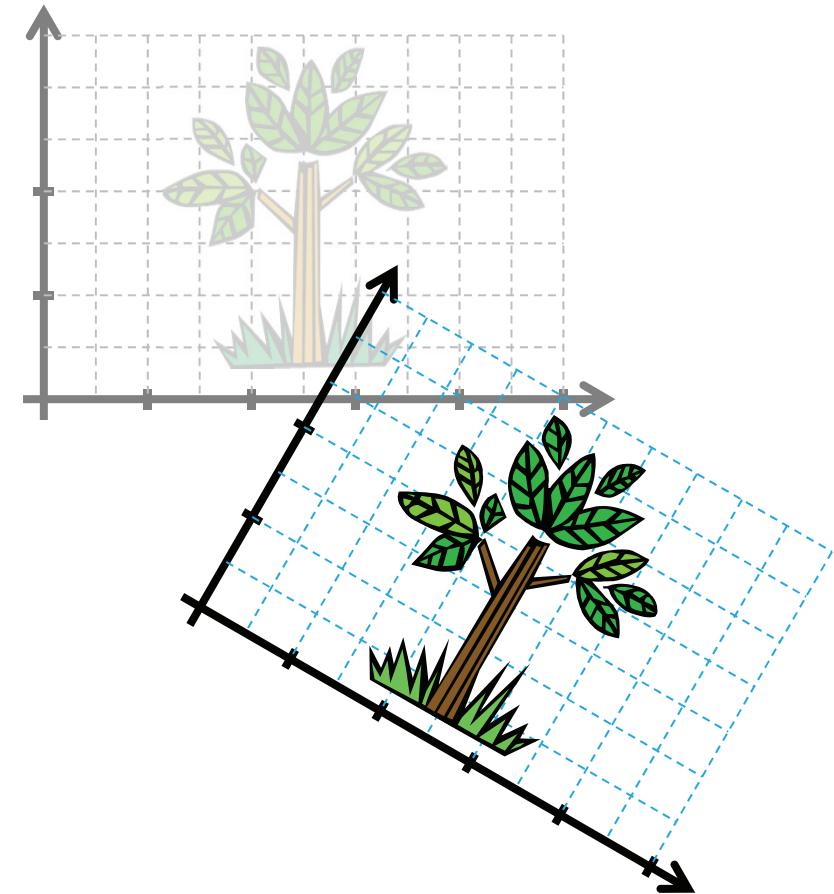
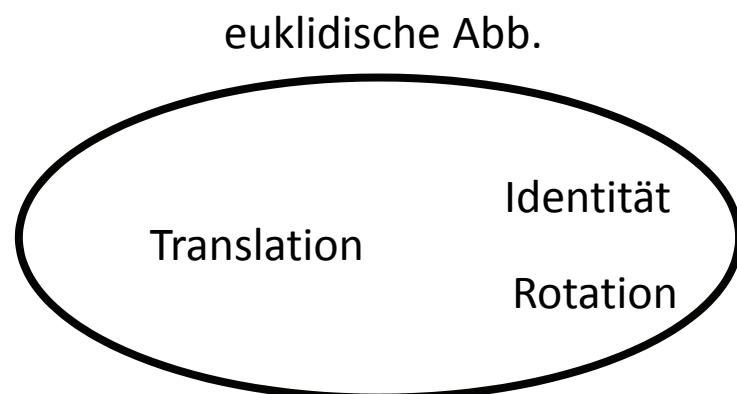
Grundlegende Transformationen



- ▶ können beliebig kombiniert werden
- ▶ sind umkehrbar/invertierbar
- ▶ natürlich abgesehen von einer Skalierung mit 0

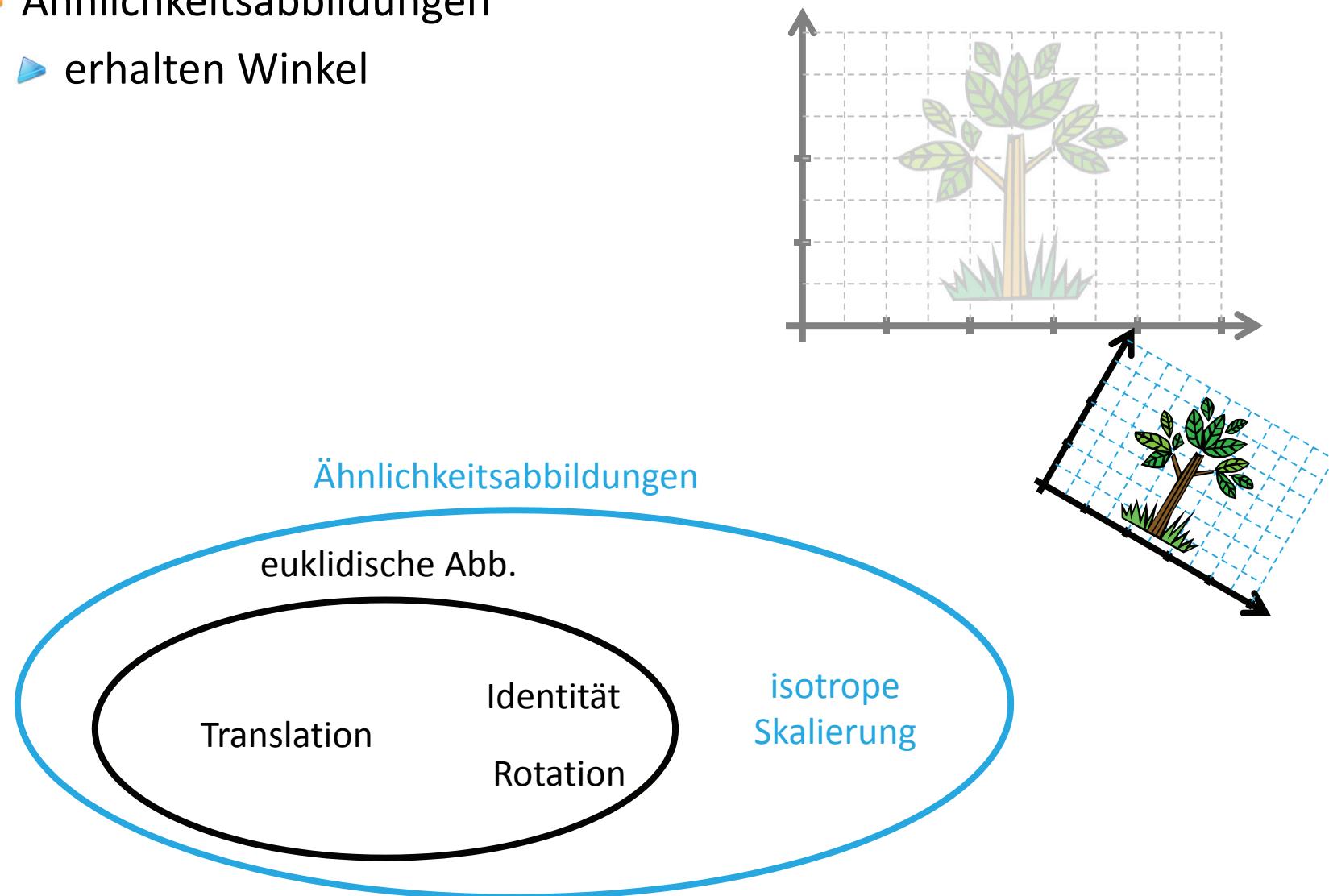
Transformationsgruppen

- ▶ euklidische/rigide Transformationen
 - ▶ erhalten Abstände und Inhaltsgrößen (Volumina, ...)
 - ▶ erhalten Winkel



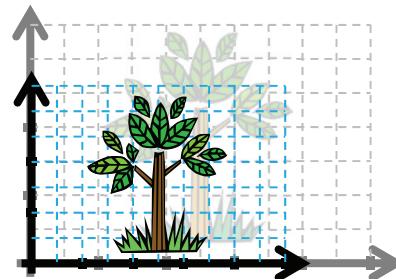
Transformationsgruppen

- ▶ Ähnlichkeitsabbildungen
 - ▶ erhalten Winkel

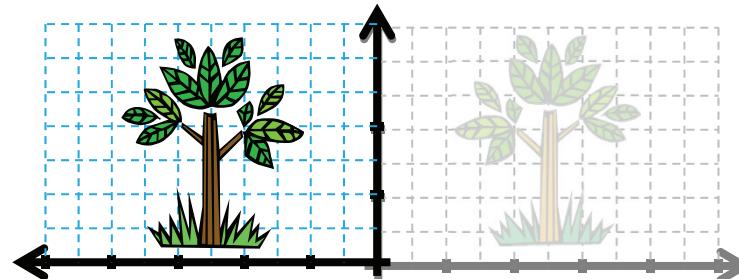


Transformationsgruppen

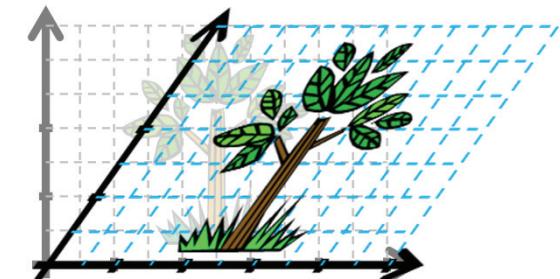
► lineare Abbildungen



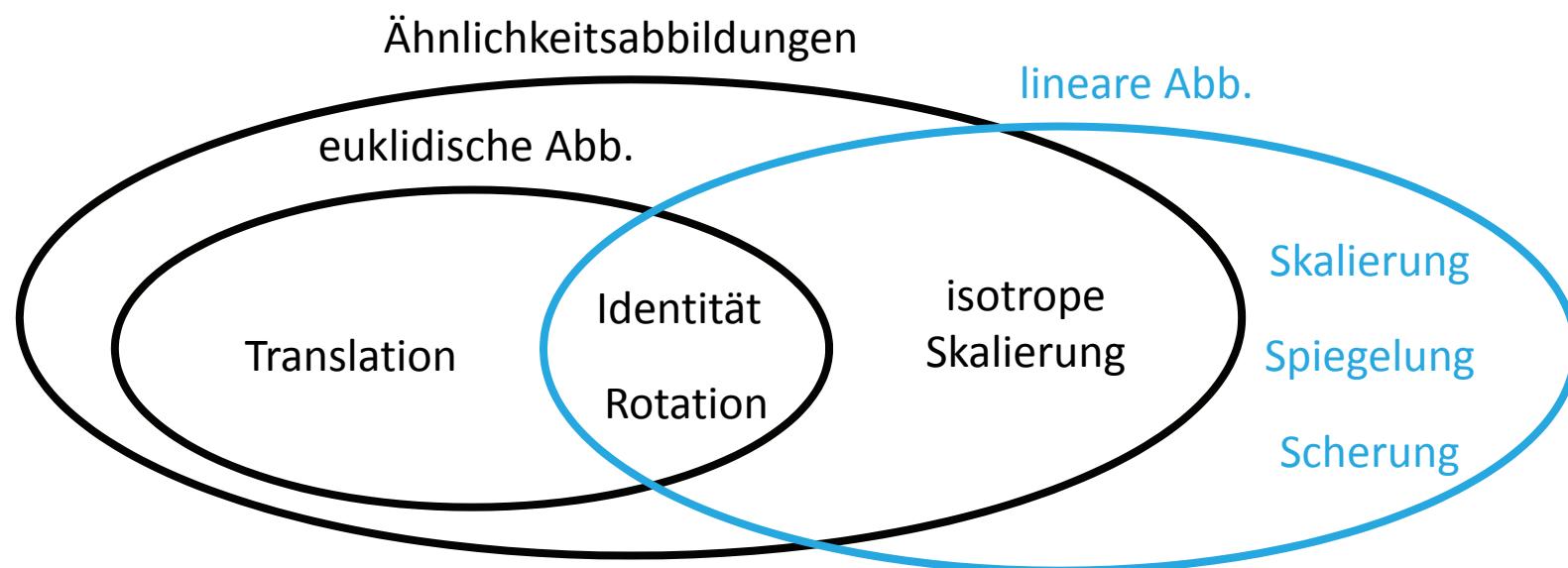
Skalierung



Spiegelung

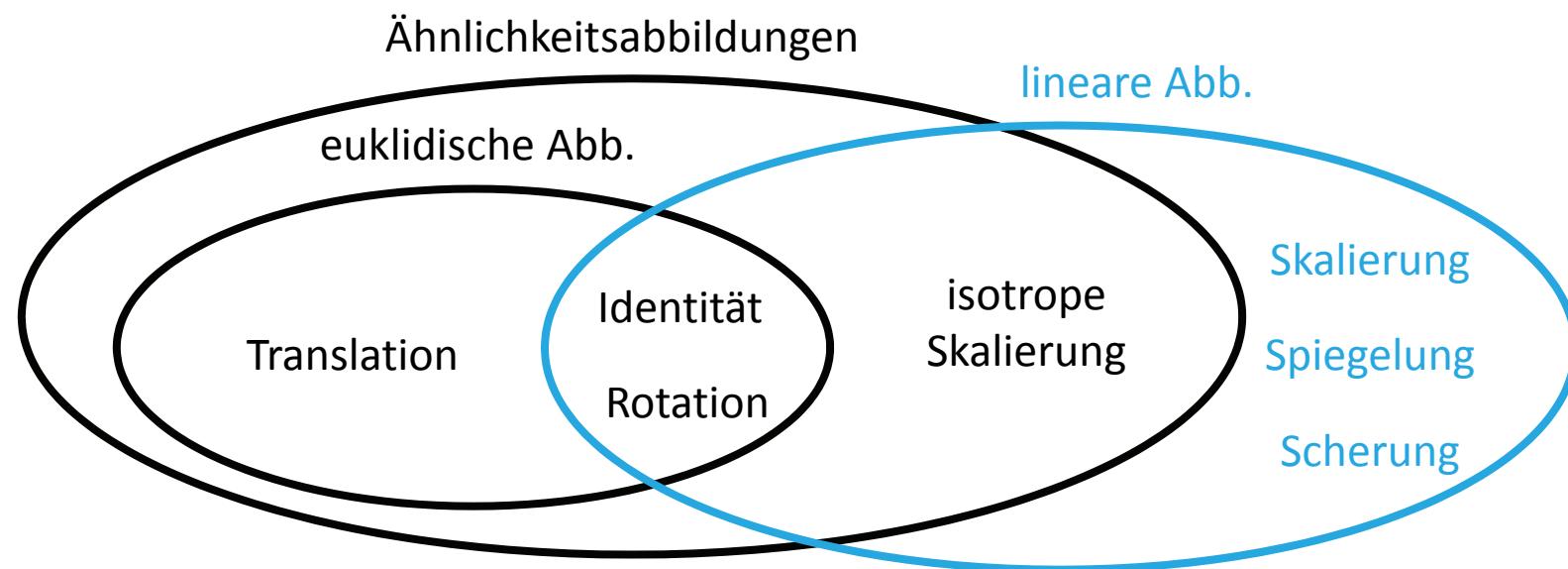
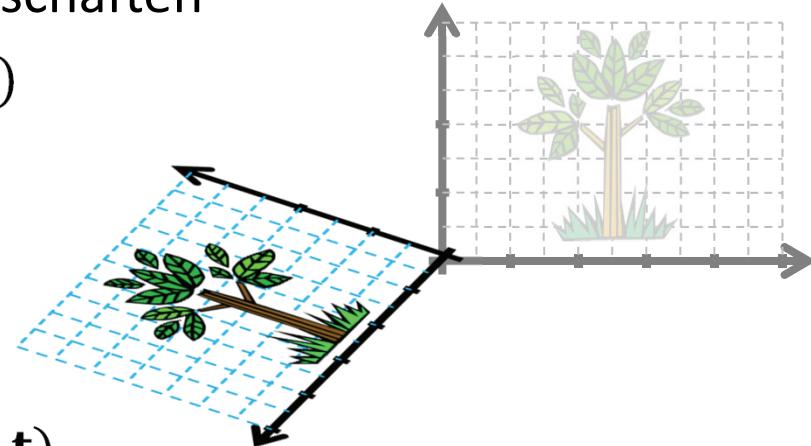


Scherung



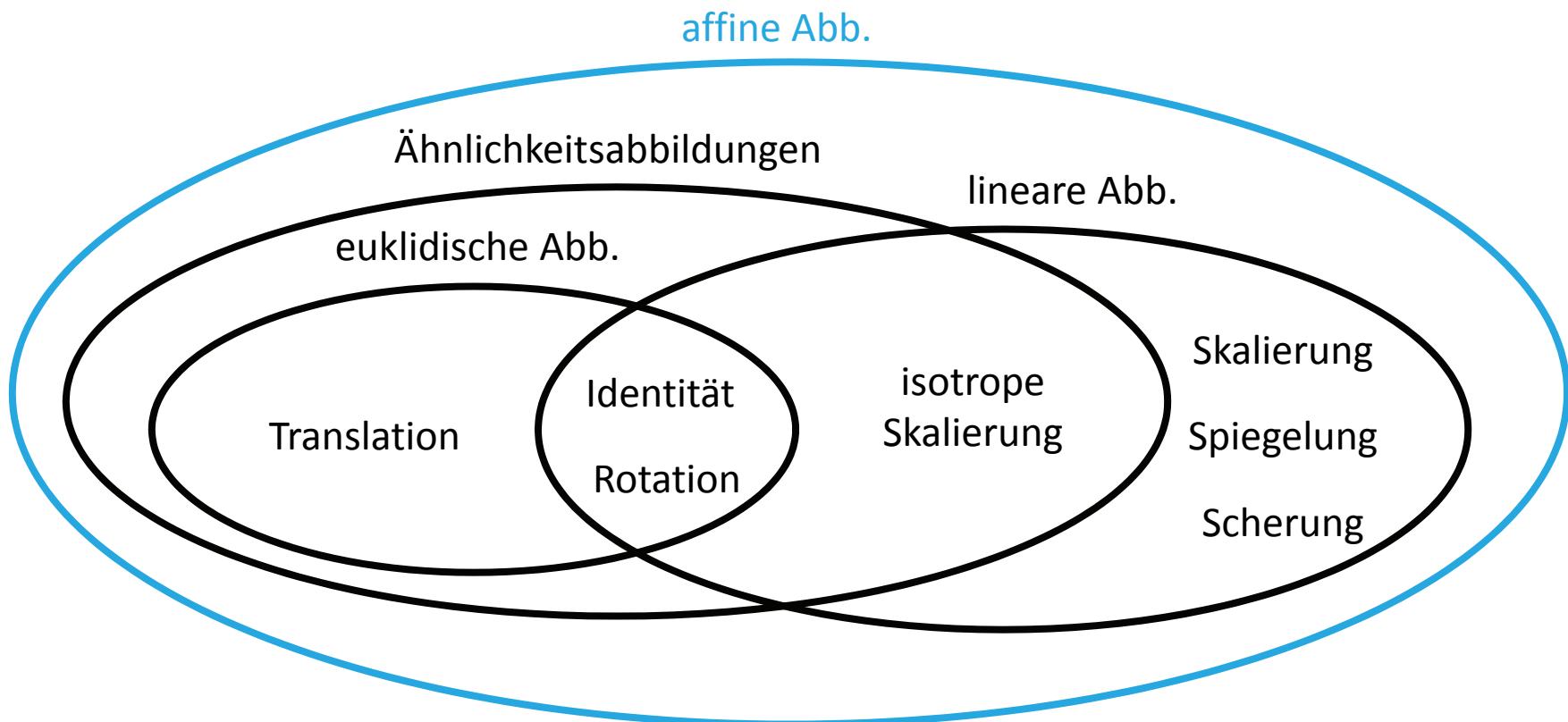
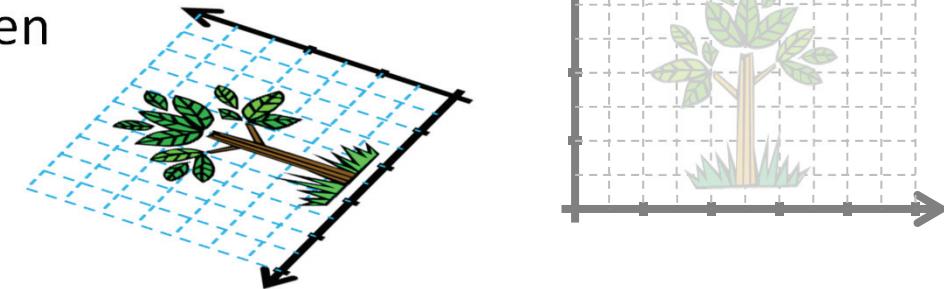
Transformationsgruppen

- ▶ lineare Abbildungen haben zwei Eigenschaften
 - ▶ additiv: $T(\mathbf{p} + \mathbf{q}) = T(\mathbf{p}) + T(\mathbf{q})$
 - ▶ homogen: $T(a\mathbf{p}) = aT(\mathbf{p})$
- ▶ eine Translation ist nicht linear:
 - ▶ sei $T(\mathbf{p}) = \mathbf{p} + \mathbf{t}$
 - ▶ $T(a\mathbf{p}) = a\mathbf{p} + \mathbf{t} \neq aT(\mathbf{p}) = a(\mathbf{p} + \mathbf{t})$



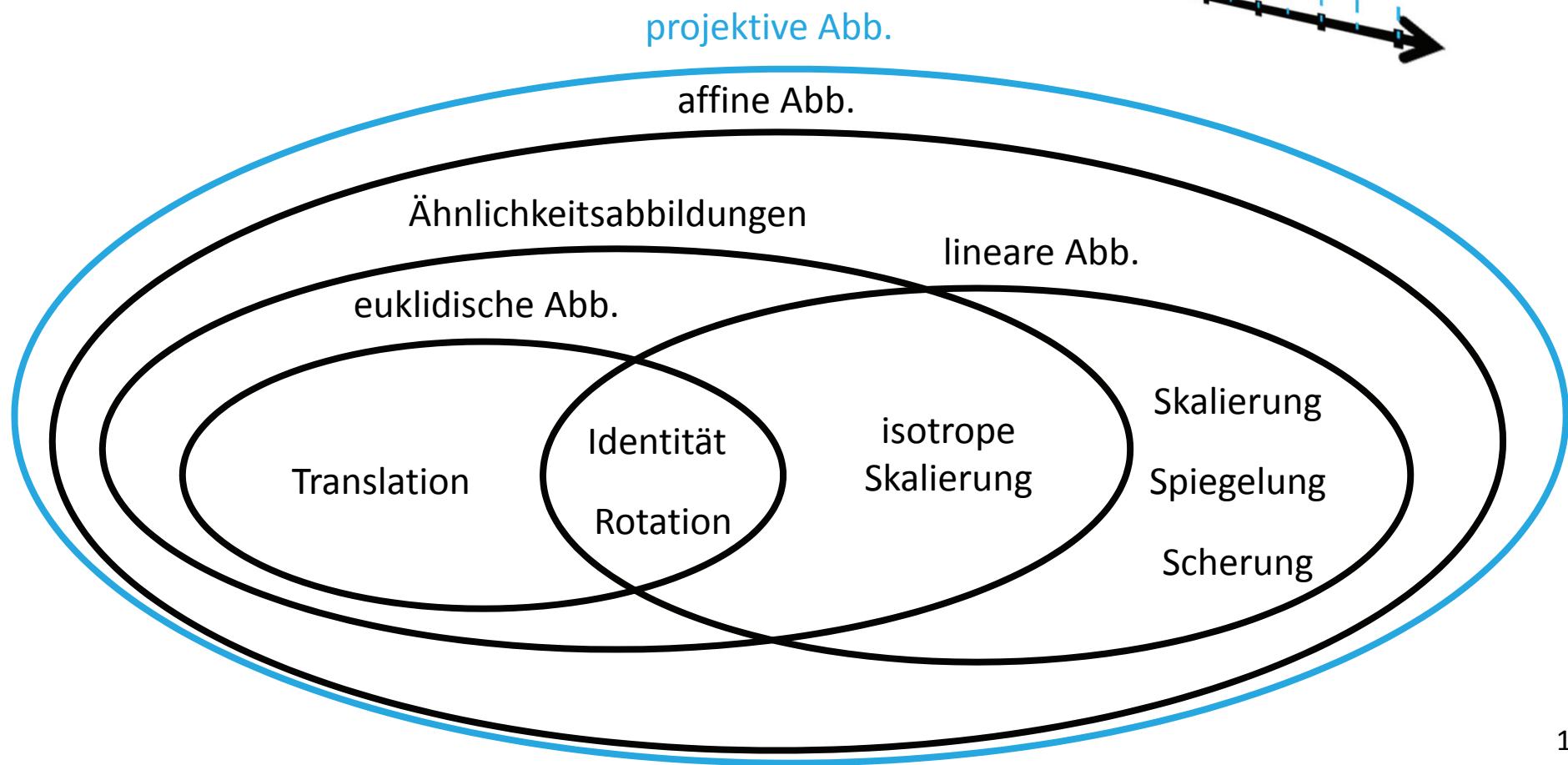
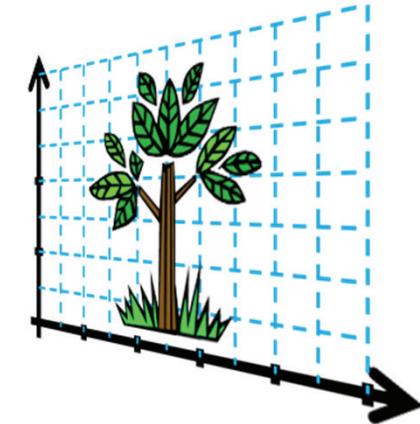
Transformationsgruppen

- ▶ affine Abbildungen
 - ▶ parallele Linien werden erhalten



Transformationsgruppen

- ▶ projektive Abbildungen
 - ▶ Geraden werden auf Geraden abgebildet



2D Transformationen



Grundlegende (lineare) 2D Transformationen

- ▶ Beschreibung mittels Vektor-Matrix-Multiplikationen
(eine Sichtweise: Wechsel der Basisvektoren)
 - ▶ Skalierung
 - ▶ Scherung
 - ▶ Spiegelung
 - ▶ Rotation
- ▶ Beispiel: Matrizen für diese Transformationen haben die Form

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \end{pmatrix}$$

2D Transformationen

Skalierung

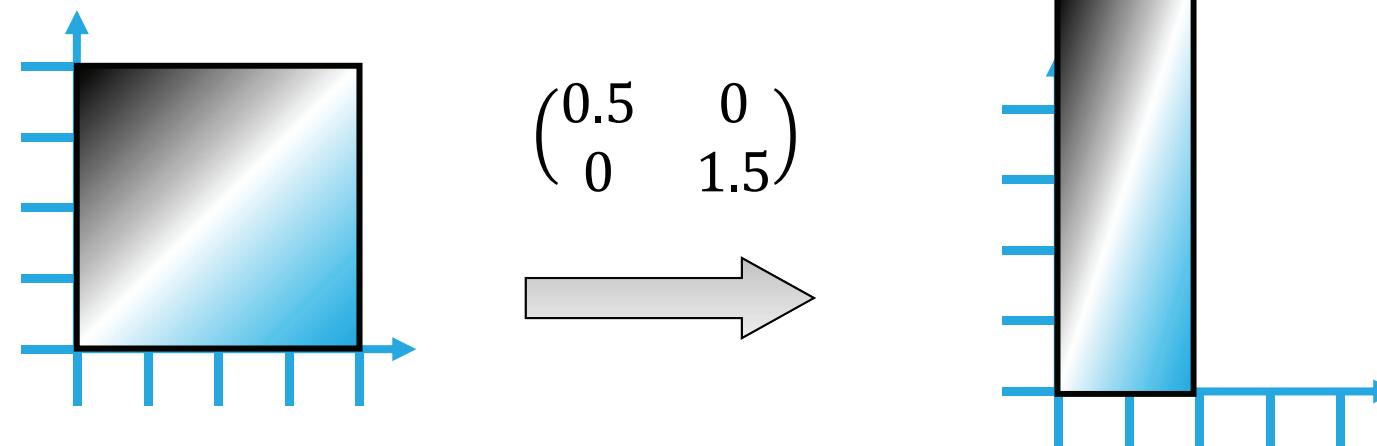
- ändert Längen, Winkel bei nicht-uniformer Skalierung (wenn $s_x \neq s_y$)

$$scale(s_x, s_y) = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix}$$

- Transformation eines Vektors

$$\begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} s_x x \\ s_y y \end{pmatrix}$$

- Beispiel:

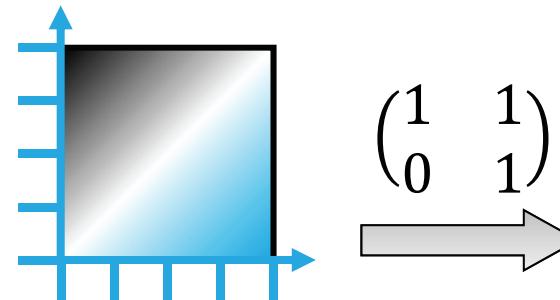


2D Transformationen

Scherung (Transvektion)

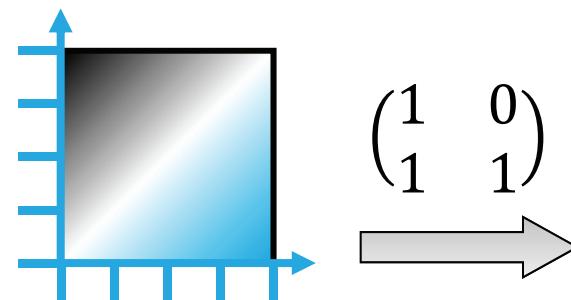
- ▶ Verschiebung parallel zu einer Achse (Flächeninhalt bleibt erhalten)
- ▶ Länge des Verschiebungsvektors proportional zum Abstand von der Achse
- ▶ Beispiele
 - ▶ horizontale Scherung (y -Koordinate bleibt unverändert)

$$shear_x(s) = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix}$$



- ▶ vertikale Scherung (x -Koordinate unverändert)

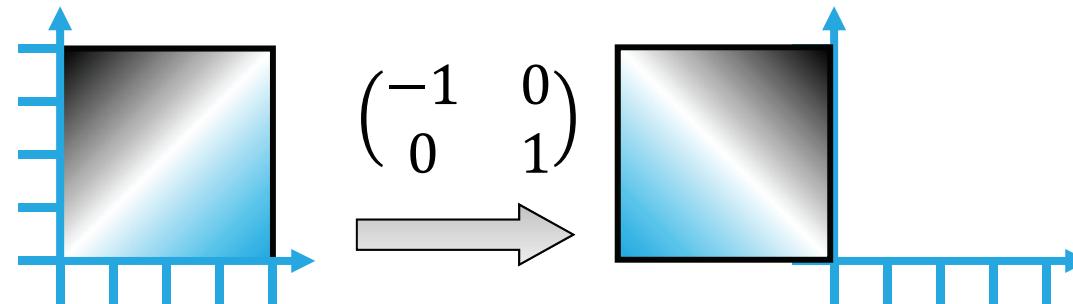
$$shear_y(s) = \begin{pmatrix} 1 & 0 \\ s & 1 \end{pmatrix}$$



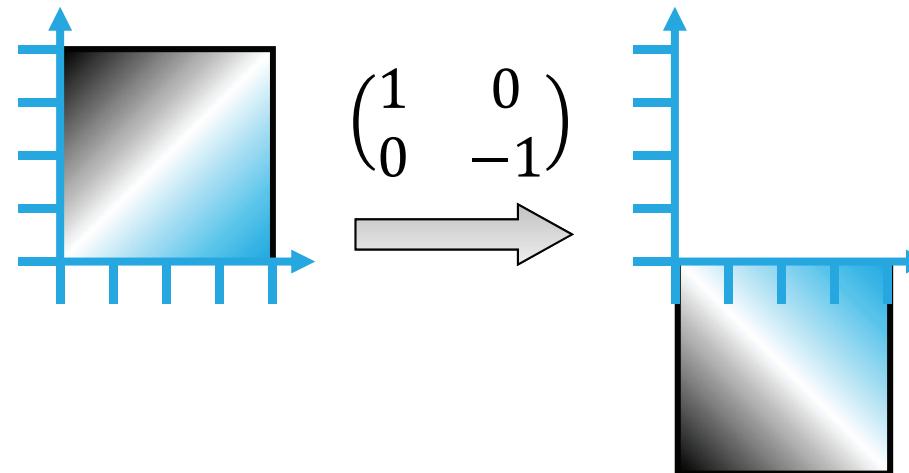
2D Transformationen

Spiegelung

- ▶ Spiegeln eines Vektors an einer der Koordinatenachsen
 - ▶ Spiegelung ist eine negative Skalierung
 - ▶ Spiegelung an der y -Achse (Multiplikation der x -Koordinate mit -1)



- ▶ ... an der x -Achse (Multiplizieren der y -Koordinate mit -1)



2D Transformationen

Rotation

► Beispiel

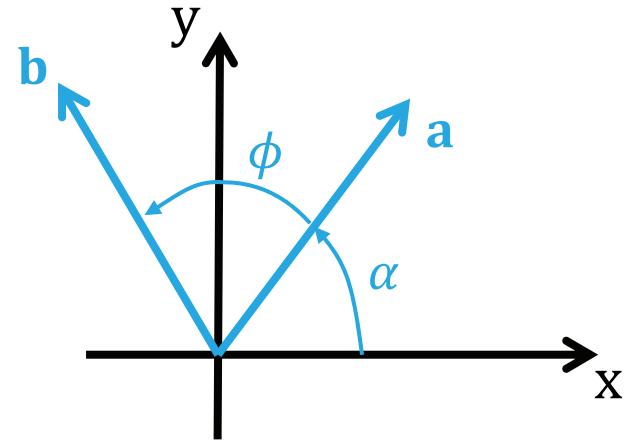
- geg. ein Vektor $\mathbf{a} = (a_x, a_y)$, mit Winkel α zur x -Achse

- Länge $r := |\mathbf{a}| = \sqrt{a_x^2 + a_y^2}$

- Polarkoordinaten

$$a_x = r \cdot \cos \alpha$$

$$a_y = r \cdot \sin \alpha$$



- Rotation um den Winkel ϕ (gegen den Uhrzeigersinn)

$$b_x = r \cos(\alpha + \phi) = r \cos \alpha \cos \phi - r \sin \alpha \sin \phi$$

$$b_y = r \sin(\alpha + \phi) = r \sin \alpha \cos \phi + r \cos \alpha \sin \phi$$

(Additionstheoreme)

2D Transformationen



Rotation (Beispiel)

- Rotation um den Winkel ϕ (gegen den Uhrzeigersinn)

$$b_x = r \cos(\alpha + \phi) = r \cos \alpha \cos \phi - r \sin \alpha \sin \phi$$

$$b_y = r \sin(\alpha + \phi) = r \sin \alpha \cos \phi + r \cos \alpha \sin \phi$$

- Substitution ($a_x = r \cdot \cos \alpha$ und $a_y = r \cdot \sin \alpha$)

$$b_x = a_x \cos \phi - a_y \sin \phi$$

$$b_y = a_y \cos \phi + a_x \sin \phi$$

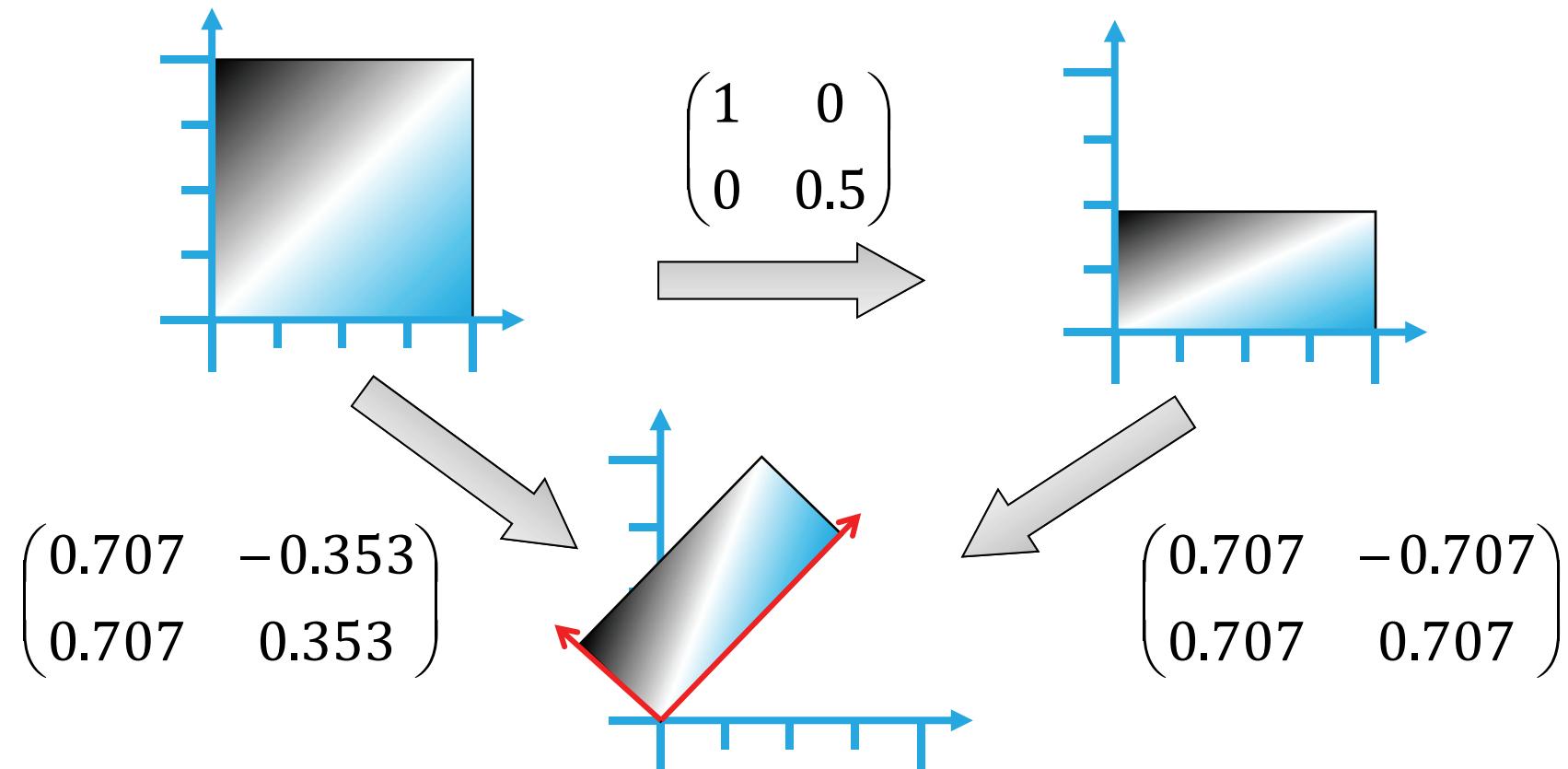
- Matrix (Abb. von **a** auf **b** bzw. Rotation um ϕ gegen den Uhrzeigersinn)

$$\text{rotate}(\phi) = \mathbf{R}(\phi) = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix}$$

Zusammengesetzte 2D Transformationen

Hintereinanderausführung von Transformationen

- ▶ Hintereinanderausführung = Matrixmultiplikation
 - ▶ erste Transformation: $v_2 = Sv_1$
 - ▶ zweite Transformation: $v_3 = Rv_2$ bzw. $v_3 = R(Sv_1) = (RS)v_1$
- ▶ kommt in der Computergrafik häufig vor



Zusammengesetzte 2D Transformationen



Hintereinanderausführung von Transformationen

- ▶ Effekt zweier (allg. mehrerer) Transformationen durch eine Matrix

$$\mathbf{v}_3 = (\mathbf{RS})\mathbf{v}_1 = \mathbf{M}\mathbf{v}_1$$

- ▶ Transformationen werden von rechts angewendet
- ▶ hier: erst \mathbf{S} , dann \mathbf{R}
- ▶ Matrix-Multiplikation
 - ▶ $\mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{B} \in \mathbb{R}^{n \times p}, \mathbf{C} = \mathbf{AB} (\in \mathbb{R}^{m \times p})$ mit $c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$
 - ▶ Eigenschaften
 - ▶ assoziativ $(\mathbf{RS})\mathbf{T} = \mathbf{R}(\mathbf{ST})$
 - ▶ **nicht** kommutativ $\mathbf{RS} \neq \mathbf{SR}$
→ die Reihenfolge der Transformationen ist entscheidend

Matrixmultiplikation



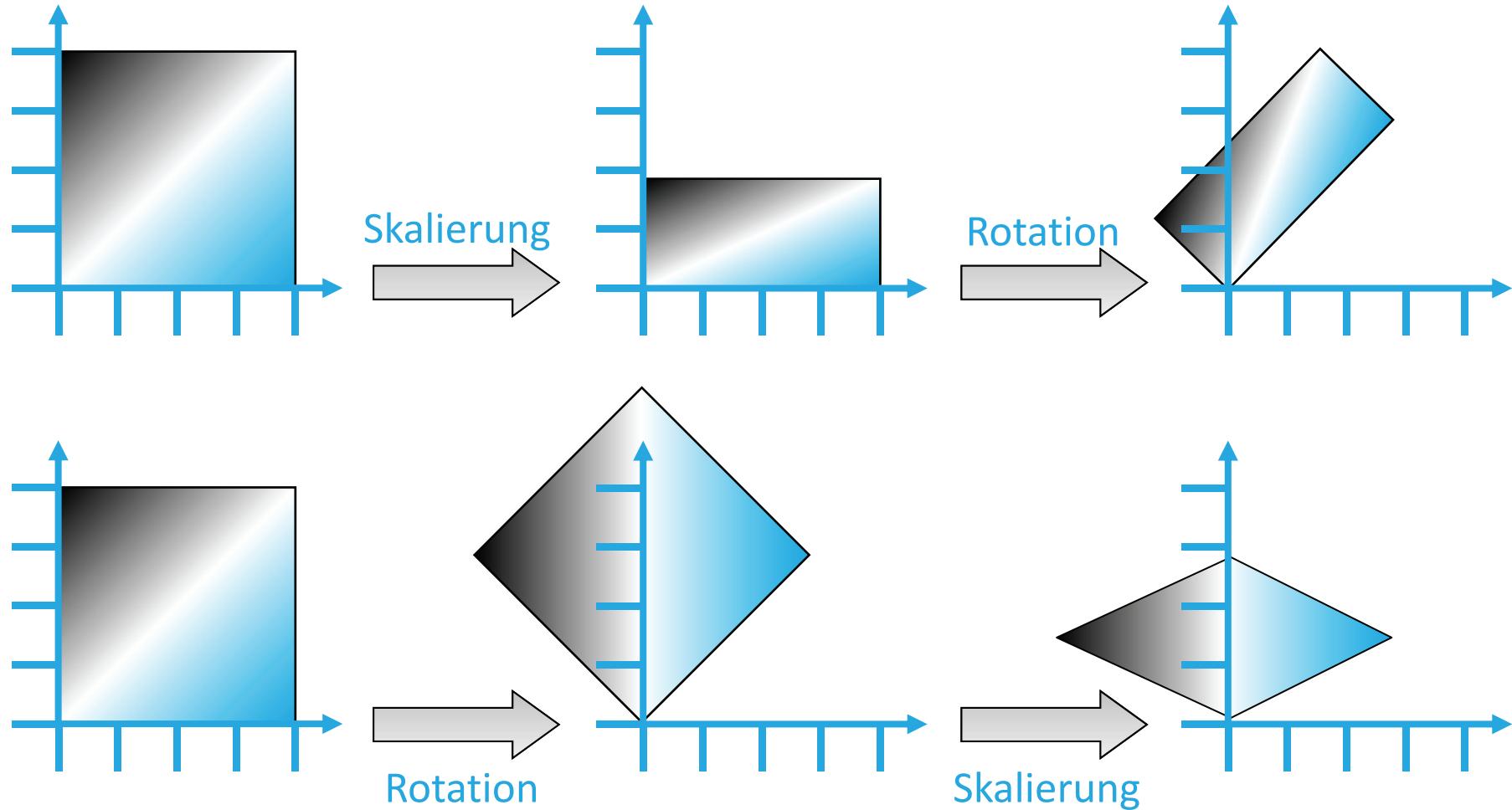
► Beispiel: Falksches Schema

- $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{B} \in \mathbb{R}^{n \times p}$, $\mathbf{C} = \mathbf{AB}$ ($\in \mathbb{R}^{m \times p}$) mit $c_{ij} = \sum_{k=1}^n a_{ik} b_{kj}$
 - Element c_{32} entsteht aus dem Skalarprodukt der 3. Zeile von \mathbf{A} und der 2. Spalte von \mathbf{B}

$$\begin{array}{c}
 \begin{array}{c|c|c}
 & 2 & 2 & 3 \\
 & -2 & -1 & 2 \\
 & 3 & 4 & 1 \\
 & -3 & 4 & 0 \\
 & 1 & -3 & 4 \\
 \hline
 \end{array} & \text{B} \\
 \\[10pt]
 \begin{array}{c|c|c|c|c|c|c|c}
 2 & -1 & 0 & 3 & -4 & -7 & 29 & -12 \\
 3 & -2 & 2 & 1 & 0 & 13 & 20 & 7 \\
 -1 & 3 & -1 & -2 & -4 & -9 & -5 & -14 \\
 3 & 3 & -4 & 2 & -2 & -20 & 1 & 3
 \end{array} & \text{A} \\
 \\[10pt]
 \begin{array}{c|c|c|c|c|c|c|c}
 & 2 & 2 & 3 \\
 & -2 & -1 & 2 \\
 & 3 & 4 & 1 \\
 & -3 & 4 & 0 \\
 & 1 & -3 & 4 \\
 \hline
 \end{array} & \text{C} = \text{AB}
 \end{array}$$

Zusammengesetzte 2D Transformationen

Reihenfolge der Transformationen ist entscheidend



3D Transformationen

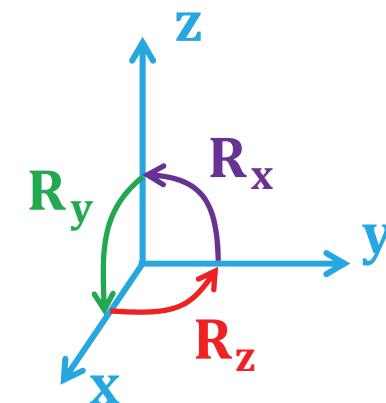
Rotation um die x -, y - bzw. z -Achse

- R_x dreht die y -Achse in Richtung z -Achse,
 R_y dreht die z -Achse in Richtung x -Achse und
 R_z dreht die x -Achse in Richtung y -Achse

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

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

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



3D Transformationen: Rotationen



3D Rotationen allgemein

- ▶ repräsentiert durch orthogonale Matrizen (orientierungserhaltend, Zeilen- und Spaltenvektoren paarweise orthonormal)
- ▶ eine quadratische, reelle Matrix ist orthogonal, wenn gilt:

$$\mathbf{M}^T \cdot \mathbf{M} = \mathbf{M} \cdot \mathbf{M}^T = I_{n \times n} \text{ und } \det(\mathbf{M}) = 1$$

- ▶ es gilt also: $\mathbf{M}^{-1} = \mathbf{M}^T$
- ▶ wir ahnen es schon, die algebraische Inverse ist auch die geometrische Inverse: auch \mathbf{M}^{-1} bzw. \mathbf{M}^T sind Rotationsmatrizen

3D Transformationen: Rotationen



Rotation des Koordinatensystems u, v, w auf das kartesische KoSys

- ▶ geg. drei Vektoren u, v, w die ein orthonormales System bilden
- ▶ x, y, z sind die kartesischen Einheitsvektoren

$$u = u_x \mathbf{x} + u_y \mathbf{y} + u_z \mathbf{z}$$

$$v = v_x \mathbf{x} + v_y \mathbf{y} + v_z \mathbf{z}$$

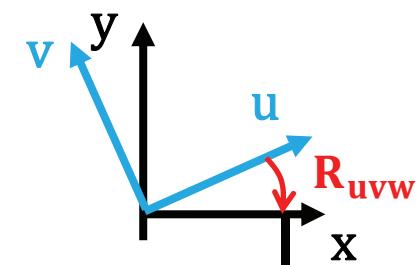
$$w = w_x \mathbf{x} + w_y \mathbf{y} + w_z \mathbf{z}$$

$$\mathbf{u} \cdot \mathbf{u} = \mathbf{v} \cdot \mathbf{v} = \mathbf{w} \cdot \mathbf{w} = 1$$

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{v} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{u} = 0$$

- ▶ Rotationsmatrix

$$\mathbf{R}_{uvw} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix}$$



3D Transformationen: Rotationen

Rotation des Koordinatensystems $\mathbf{u}, \mathbf{v}, \mathbf{w}$ auf das kartesische KoSys

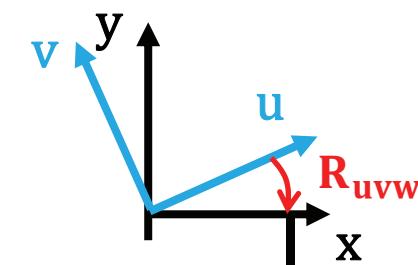
► Rotationsmatrix $\mathbf{R}_{uvw} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix}$

► \mathbf{R}_{uvw} bildet die Basisvektoren $\mathbf{u}, \mathbf{v}, \mathbf{w}$ durch Rotation auf die kartesischen Achsen ab:

► $\mathbf{R}_{uvw}\mathbf{u} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} \begin{pmatrix} u_x \\ u_y \\ u_z \end{pmatrix} = \begin{pmatrix} \mathbf{u} \cdot \mathbf{u} \\ \mathbf{v} \cdot \mathbf{u} \\ \mathbf{w} \cdot \mathbf{u} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \mathbf{x}$

► $\mathbf{R}_{uvw}\mathbf{v} = \mathbf{y}$

► $\mathbf{R}_{uvw}\mathbf{w} = \mathbf{z}$



3D Transformationen: Rotationen



Rotation des kartesischen Koordinatensystems auf das KoSys $\mathbf{u}, \mathbf{v}, \mathbf{w}$

- wenn \mathbf{R}_{uvw} eine Rotationsmatrix ist, dann ist auch \mathbf{R}_{uvw}^T eine Rotationsmatrix und es gilt: $\mathbf{R}_{uvw}^T = \mathbf{R}_{uvw}^{-1}$

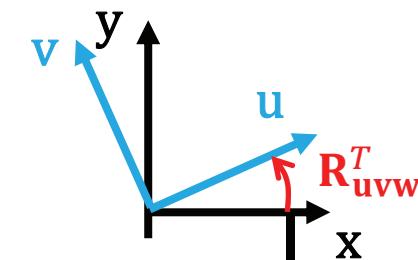
- Rotationsmatrizen

$$\mathbf{R}_{uvw} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} \text{ und } \mathbf{R}_{uvw}^T = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix}$$

- \mathbf{R}_{uvw}^T bildet die kartesischen Einheitsvektoren auf $\mathbf{u}, \mathbf{v}, \mathbf{w}$ ab

$$\mathbf{R}_{uvw}^T \mathbf{x} = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix} \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = \mathbf{u}$$

$$\mathbf{R}_{uvw}^T \mathbf{y} = \mathbf{v} \text{ bzw. } \mathbf{R}_{uvw}^T \mathbf{z} = \mathbf{w}$$



3D Transformationen: Rotationen

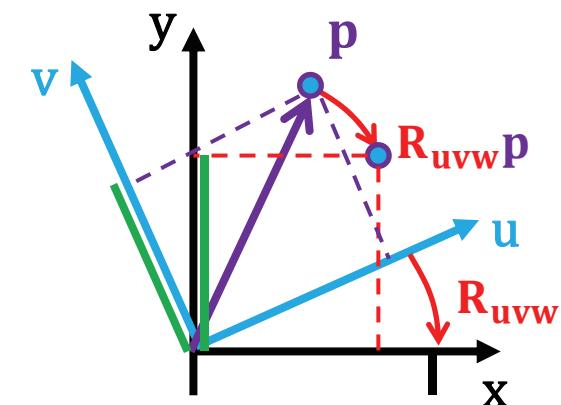


Transformation von Punkten zwischen den Koordinatensystemen

- ist der Punkt \mathbf{p} definiert in kartesischen Koordinaten

$$\triangleright \mathbf{R}_{uvw} \cdot \mathbf{p} = \begin{pmatrix} \mathbf{u} \cdot \mathbf{p} \\ \mathbf{v} \cdot \mathbf{p} \\ \mathbf{w} \cdot \mathbf{p} \end{pmatrix} \text{ berechnet lokale Koordinaten bzgl. } \mathbf{u}, \mathbf{v}, \mathbf{w}$$

- \mathbf{R}_{uvw} bildet $\mathbf{u}, \mathbf{v}, \mathbf{w}$ auf die kartesischen Achsen ab
- $\mathbf{u} \cdot \mathbf{p}$ ist die Projektion von \mathbf{p} auf \mathbf{u} , $\mathbf{v} \cdot \mathbf{p}$...
→ \mathbf{R}_{uvw} transformiert \mathbf{p} „in“ das KoSys $\mathbf{u}, \mathbf{v}, \mathbf{w}$
(berechnet lokale Koordinaten bzgl. $\mathbf{u}, \mathbf{v}, \mathbf{w}$)



3D Transformationen: Rotationen



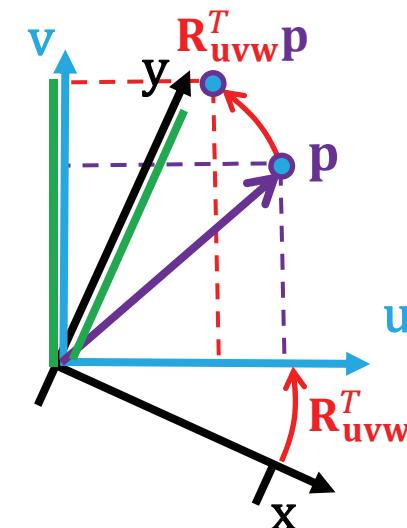
Transformation von Punkten zwischen den Koordinatensystemen

- sind die Koordinaten von Punkt p geg. bzgl. KoSys **u, v, w** dann

$$\blacktriangleright \mathbf{R}_{uvw}^T \cdot \mathbf{p} = \begin{pmatrix} u_x p_x & v_x p_y & w_x p_z \\ u_y p_x & v_y p_y & w_y p_z \\ u_z p_x & v_z p_y & w_z p_z \end{pmatrix} = \mathbf{u} \cdot p_x + \mathbf{v} \cdot p_y + \mathbf{w} \cdot p_z$$

berechnet kartesische Koordinaten von **p** im globalen KoSys

- \mathbf{R}_{uvw}^T bildet die kartesischen Einheitsvektoren auf **u, v, w** ab
- \mathbf{R}_{uvw}^T transformiert **p** „aus“ dem KoSys **u, v, w**



3D Transformationen: Rotationen



Rotation, Transponierte und Inverse

- ▶ für orthogonale Matrizen gilt: die Transponierte ist auch Inverse
- ▶ für Transformationsmatrizen gilt:
algebraische Inverse = geometrische Inverse

$$\begin{aligned}\mathbf{R}_{uvw}^T \cdot \mathbf{R}_{uvw} \mathbf{u} &= \mathbf{R}_{uvw}^T \cdot (\mathbf{R}_{uvw} \mathbf{u}) \\ &= \mathbf{R}_{uvw}^T \mathbf{x} \\ &= \mathbf{u}\end{aligned}$$

$$\begin{aligned}\mathbf{R}_{uvw} \cdot \mathbf{R}_{uvw}^T \mathbf{x} &= \mathbf{R}_{uvw} \cdot (\mathbf{R}_{uvw}^T \mathbf{x}) \\ &= \mathbf{R}_{uvw} \mathbf{u} \\ &= \mathbf{x}\end{aligned}$$

$$\mathbf{R}_{uvw}^T \cdot \mathbf{R}_{uvw} = \mathbf{R}_{uvw} \cdot \mathbf{R}_{uvw}^T = \mathbf{I}$$

Repräsentationen von Rotationen



Verschiedene Darstellungen/Beschreibungen von Rotationen in 3D

- ▶ orthogonale Matrizen
- ▶ Euler Rotationen:
 - ▶ $R_z \rightarrow R_x \rightarrow R_z$
 - ▶ $R_z \rightarrow R_y \rightarrow R_z$
 - ▶ $R_x \rightarrow R_y \rightarrow R_z$
- ▶ Rotationsachse und -winkel
- ▶ Quaternionen (siehe Vorlesung Interaktive Computergrafik, SoSe)
- ▶ 2 planare Spiegelungen
- ▶ ...

Euler Rotationen



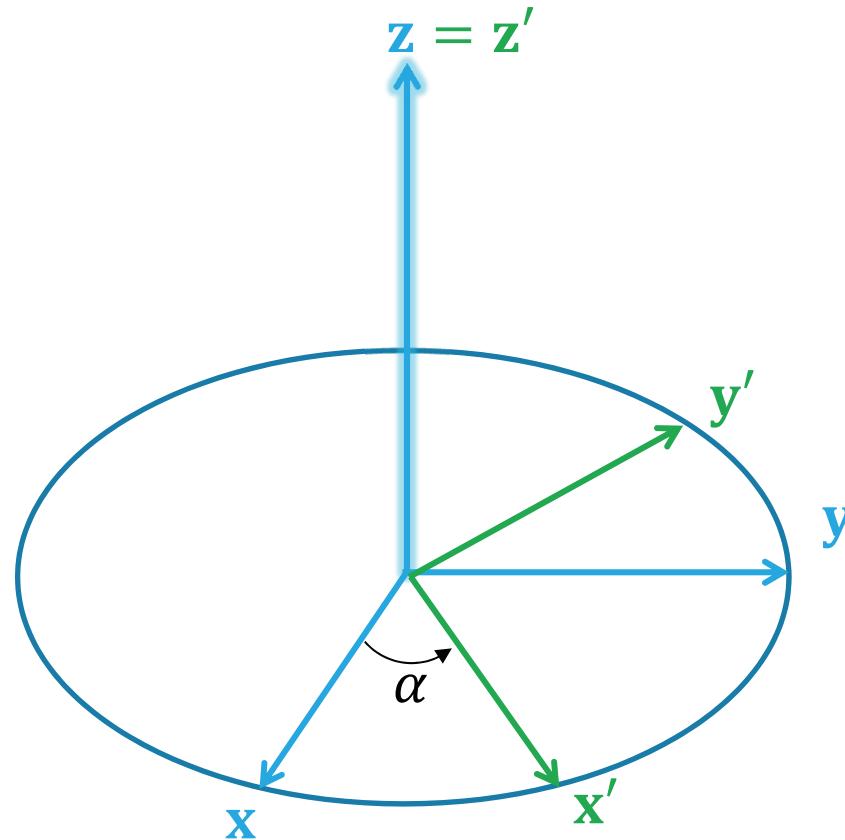
- ▶ jede Rotation kann durch 3 Rotationen um die Hauptachsen (x, y, z) ausgedrückt werden (Leonhard Euler 1707 – 1783)
 - ▶ man kann auch andere Achsen und Reihenfolgen festlegen, z.B. Luftfahrttnorm (DIN 9300) (Yaw-Pitch-Roll z, y', x'')
- ▶ sind die Rotationen um die x -, y - bzw. z -Achse ψ, θ und ϕ dann ist die Rotationsmatrix:

$$\mathbf{R} = \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_x(\psi) = \begin{pmatrix} \cos \theta \cos \phi & \sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi & \cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi \\ \cos \theta \sin \phi & \sin \psi \sin \theta \sin \phi + \cos \psi \cos \phi & \cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi \\ -\sin \theta & \sin \psi \cos \theta & \cos \psi \cos \theta \end{pmatrix}$$

- ▶ die Winkel ψ, θ und ϕ heißen Eulerwinkel
 - ▶ ... und beschreiben die Orientierung eines Objektes
 - ▶ ... zusammen mit der Festlegung der Achsen und der Reihenfolge

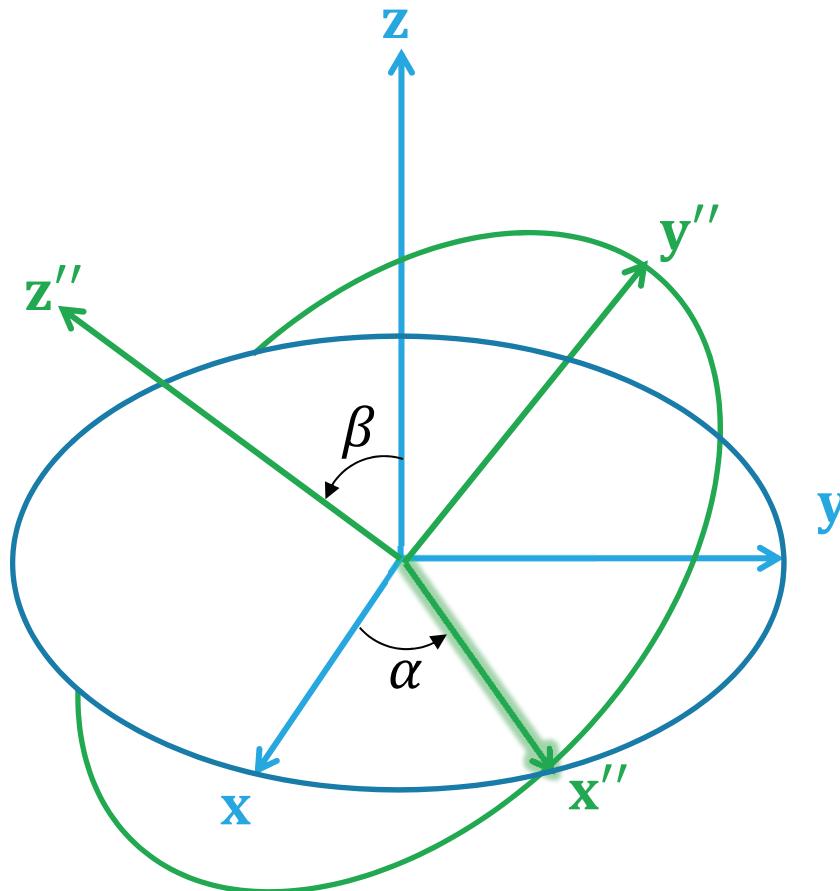
Euler Rotationen

- Rotation kann ebenfalls durch Eulerrotation $\mathbf{R} = \mathbf{R}_z(\gamma)\mathbf{R}_x(\beta)\mathbf{R}_z(\alpha)$ ausgedrückt werden, also Eulerwinkel α, β, γ für die Rotation um z-x-z



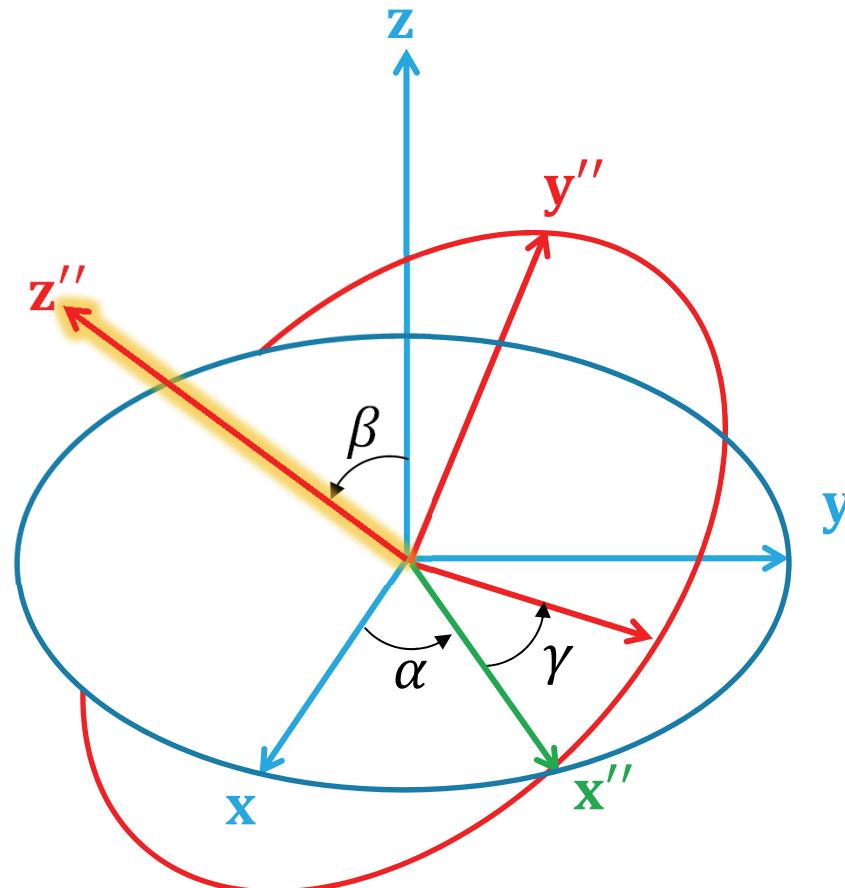
Euler Rotationen

- Eulerwinkel α, β, γ für die Rotation um z-x-z



Euler Rotationen

- Eulerwinkel α, β, γ für die Rotation um z-x-z



Euler Rotationen

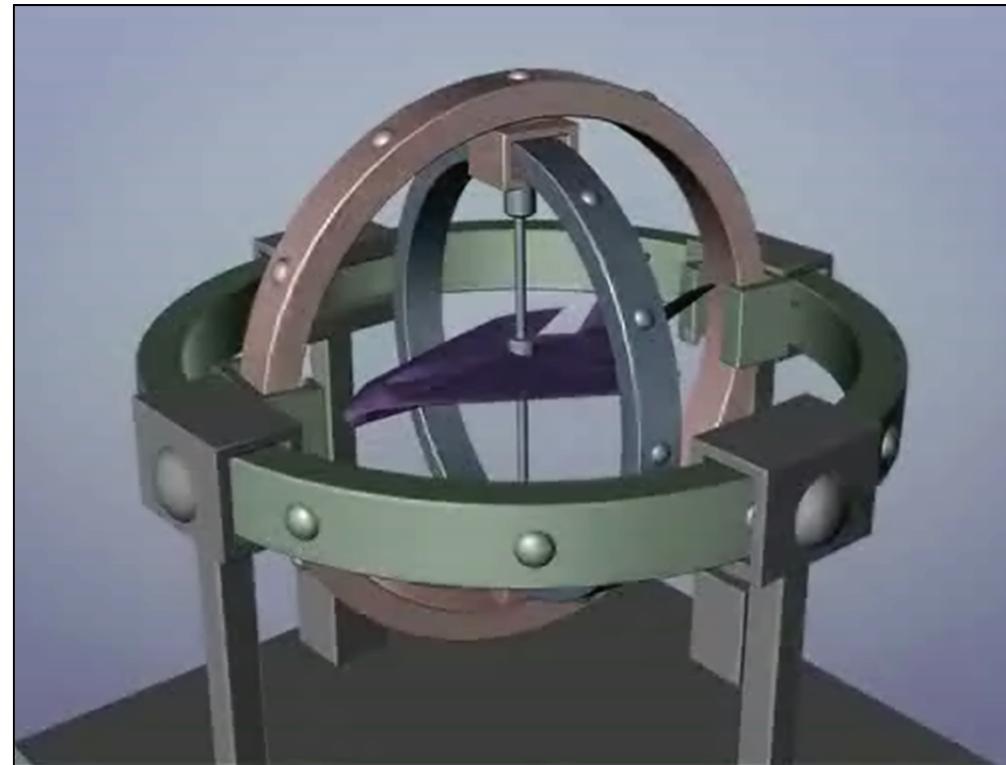


- ▶ wenn eine Rotationsmatrix \mathbf{R} und eine Achsenkonvention vorgegeben ist, können die Eulerwinkel abgelesen werden
- ▶ $\mathbf{R}_{Euler} = \mathbf{R}_z(\phi)\mathbf{R}_y(\theta)\mathbf{R}_x(\psi) =$
$$\begin{pmatrix} \cos \theta \cos \phi & \sin \psi \sin \theta \cos \phi - \cos \psi \sin \phi & \cos \psi \sin \theta \cos \phi + \sin \psi \sin \phi \\ \cos \theta \sin \phi & \sin \psi \sin \theta \sin \phi + \cos \psi \cos \phi & \cos \psi \sin \theta \sin \phi - \sin \psi \cos \phi \\ -\sin \theta & \sin \psi \cos \theta & \cos \psi \cos \theta \end{pmatrix}$$
- ▶ $\mathbf{R} = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \Rightarrow R_{31} = -\sin \theta, \frac{R_{32}}{R_{33}} = \tan \psi, \frac{R_{21}}{R_{11}} = \tan \phi$
- ▶ Bemerkung:
 - ▶ die Lösung ist nicht eindeutig
 - ▶ es gibt Spezialfälle, z.B. $\cos \theta = 0$

Euler Rotationen

Kardanische Blockade (engl. Gimbal Lock)

- ▶ Problem bei der Eulerrotation: die 1. und 3. Rotationsachse können zusammenfallen, d.h. nur Summe aus 1. und 3. Winkel ist relevant
- ▶ Anm. die Apparatur im Video nennt sich kardanische Aufhängung



<http://www.youtube.com/watch?v=zc8b2Jo7mno>

Rotation um eine beliebige Achse



Rotation um eine Achse \mathbf{d} um den Winkel ϕ

- ▶ bestimme (irgendein) orthonormales KoSys mit \mathbf{d} als eine der Achsen
 - ▶ sei $\mathbf{d} = (d_x, d_y, d_z)$ mit $|\mathbf{d}| = 1$
 - ▶ wähle z.B. $\mathbf{e} = \frac{1}{\sqrt{d_y^2 + d_z^2}}(0, -d_z, d_y)$ und $\mathbf{f} = \mathbf{d} \times \mathbf{e}$
 - ▶ Abbildung von \mathbf{d} auf die x -Achse des kartesischen KoSys (analog \mathbf{e} auf y und \mathbf{f} auf z): transformiert einen Punkt in das KoSys $\mathbf{d}, \mathbf{e}, \mathbf{f}$

$$\mathbf{M} = \begin{pmatrix} \mathbf{d}^T \\ \mathbf{e}^T \\ \mathbf{f}^T \end{pmatrix} = \begin{pmatrix} d_x & d_y & d_z \\ e_x & e_y & e_z \\ f_x & f_y & f_z \end{pmatrix}$$

- ▶ Rotation um die x -Achse: $\mathbf{R}_x(\phi)$ im KoSys $\mathbf{d}, \mathbf{e}, \mathbf{f}$
- ▶ Transformation zurück in das ursprüngliche KoSys: $\mathbf{M}^{-1} = \mathbf{M}^T$
- ▶ vollständige Matrix

$$\mathbf{R}_{\mathbf{d}, \phi} = \mathbf{M}^{-1} \mathbf{R}_x(\phi) \mathbf{M}$$

Inverse Transformationen



Matrixinversion

- ▶ inverse Matrix → inverse geometrische Transformation
- ▶ Matrixinversion im Allgemeinen kein einfaches Problem
- ▶ Ausnutzen der Matrizeigenschaften wann immer möglich

- ▶ $\mathbf{S}^{-1}(x, y, z) = \mathbf{S} \left(\frac{1}{x}, \frac{1}{y}, \frac{1}{z} \right)$
- ▶ $\mathbf{R}^{-1}(\phi) = \mathbf{R}(-\phi)$ oder
Orthonormalität ausnutzen: $\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I} \Rightarrow \mathbf{R}^{-1} = \mathbf{R}^T$
- ▶ $\mathbf{T}^{-1}(x, y, z) = \mathbf{T}(-x, -y, -z)$ ([Translation mit Matrix? gleich mehr!](#))
- ▶ zusammengesetzte Transformationen: $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$
- ▶ nicht alle Transformationen sind invertierbar (z.B. Projektionen, $\mathbf{S}(0,0,0)$)
- ▶ numerische Inversion
 - ▶ Adjungierte, Cramersche Regel, LU Zerlegung, Gauß-Elimination
 - ▶ ineffizient (im Vergleich zum Ansatz oben)
 - ▶ u.U. numerische Probleme

Affine Abbildungen

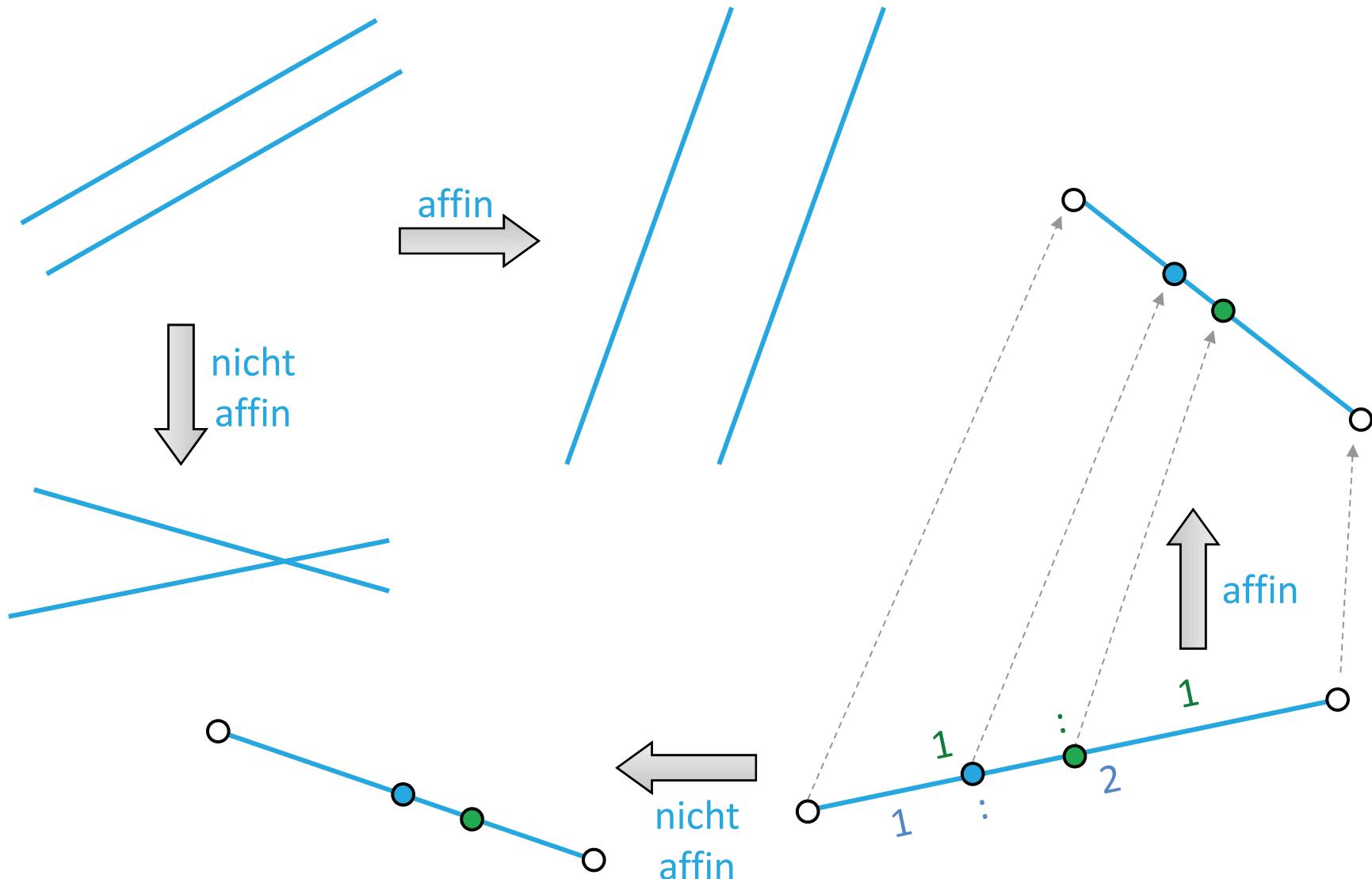


- affine Abbildung:
Kombination aus linearer Abbildung und Translation (hier in 2D)

$$\mathbf{x} \mapsto \mathbf{Ax} + \mathbf{b} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

- Eigenschaften
 - Linien werden auf Linien abgebildet
 - parallele Linien bleiben parallel
 - teilverhältnistreu
 - nicht winkelerhaltend
 - Beispiele: Rotationen, Translationen, Skalierung, Scherung

Affine Abbildungen

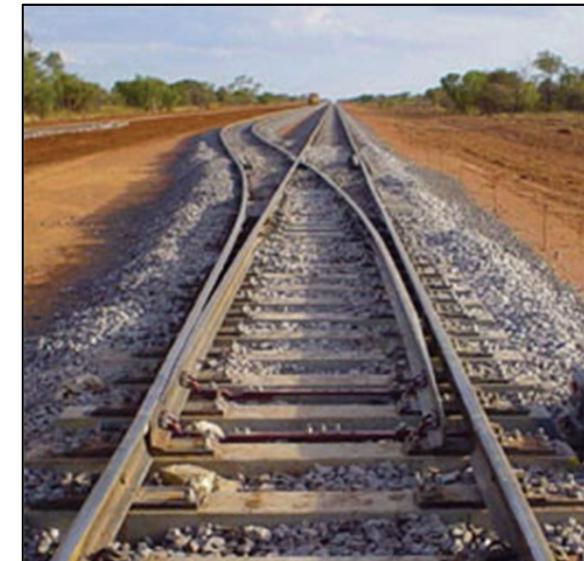


Homogene Koordinaten



Projektiver Raum

- ▶ wir ergänzen den euklidischen/affinen Raum (z.B. den \mathbb{R}^3 wie wir ihn uns vorstellen) um sogenannte Fernpunkte
- ▶ parallele Geraden im affinen Raum schneiden sich nicht – in einer Projektion allerdings schon
 - ▶ Geraden besitzen aber eine Richtung
 - ▶ Idee: ergänze den affinen Raum um diese Richtungen (= Fernpunkte oder uneigentliche Punkte)
- ▶ sogenannte „homogene Koordinaten“ werden verwendet um
 - ▶ affine Punkte und Richtungen zu beschreiben
 - ▶ Translationen durch Matrizen auszudrücken



Homogene Koordinaten

Homogene Koordinaten und projektive Räume

- die Menge aller Geraden durch den Ursprung im \mathbb{R}^3 nennt man den reellen projektiven Raum $P(\mathbb{R}^3)$

- in diesem Fall ist es eine „projektive Ebene“

- ein Vektor $\mathbf{v} \in \mathbb{R}^3$ definiert eine Gerade

- $\lambda\mathbf{v}$, $\lambda \in \mathbb{R}$ definieren dieselbe Gerade

- Dimension $\dim(P(\mathbb{R}^3)) = 2$

- Repräsentation von Punkten

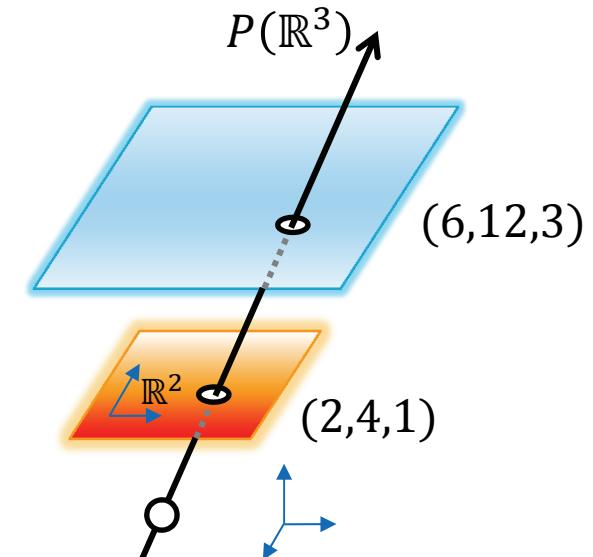
- Einbettung des \mathbb{R}^2 in den $P(\mathbb{R}^3)$

- $(x, y)_{2D} \rightarrow (x, y, 1)_h \equiv \left\{ (x', y', w) \mid \left(\frac{x'}{w}, \frac{y'}{w} \right) = (x, y) \right\}$

- Repräsentation von Richtungen

- $(x, y)_{2D} \rightarrow (x, y, 0)_h$

- $P(\mathbb{R}^3)$ enthält also affine Punkte und Richtungen des \mathbb{R}^2



Homogene Koordinaten

- ▶ Ziel: Beschreibung von affinen Punkten und Richtungen, sowie von affinen Abbildungen mit Matrizen
 - ▶ Vorteil: Zusammengesetzte affine Transformationen
- ▶ einfach: füge „1“ als 3. Koordinate (**homogene Koordinate**) hinzu:

$$(x, y)_{2D} \rightarrow (x, y, 1)_h$$

- ▶ homogene Koordinaten (etwas allgemeiner)
 - ▶ $(3x, 3y, 3)_h, \left(-\frac{x}{2}, -\frac{y}{2}, -\frac{1}{2}\right)_h, (ax, ay, a)_h$ mit $a \neq 0$ repräsentieren denselben Punkt im \mathbb{R}^2
 - ▶ „Dehomogenisierung“: $(x, y, w)_h \rightarrow \left(\frac{x}{w}, \frac{y}{w}\right)_{2D}$
- ▶ mathematische Grundlagen: projektive Geometrie

Affine Abbildungen

Affine Abbildung bisher

$$\mathbf{x} \mapsto \mathbf{Ax} + \mathbf{b} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$

Affine Abbildung mit homogenen Koordinaten

$$\begin{aligned}
 \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}_h &\mapsto \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}_h \mapsto \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}_h \\
 &= \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + b_1 \\ a_{21}x_1 + a_{22}x_2 + b_2 \\ 1 \end{pmatrix}_h \mapsto \begin{pmatrix} a_{11}x_1 + a_{12}x_2 + b_1 \\ a_{21}x_1 + a_{22}x_2 + b_2 \\ 1 \end{pmatrix} = \\
 &= \begin{pmatrix} a_{11}x_1 + a_{12}x_2 \\ a_{21}x_1 + a_{22}x_2 \\ 1 \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \\ 1 \end{pmatrix} = \mathbf{Ax} + \mathbf{b}
 \end{aligned}$$

Affine Abbildungen



- Anatomie einer affinen Transformation mit homogenen Koordinaten:
linearer Teil und Translation

$$\mathbf{M} = \begin{pmatrix} a_{11} & a_{12} & b_1 \\ a_{21} & a_{22} & b_2 \\ 0 & 0 & 1 \end{pmatrix}$$

- zusammengesetzte Transformationen

$$\mathbf{x} \xrightarrow{\mathbf{T}} \mathbf{Tx} = \mathbf{y} \xrightarrow{\mathbf{S}} \mathbf{Sy} = \mathbf{z} \quad \equiv \quad \mathbf{z} = \mathbf{S} \cdot \mathbf{T} \cdot \mathbf{x}$$

- Zusammensetzung \equiv Multiplikation
- inverse Transformation \equiv inverse Matrix
- **wichtig:** \mathbf{M} und $\lambda\mathbf{M}$ ($\lambda \neq 0$) beschreiben dieselbe Abbildung bei homogenen Koordinaten

2D Transformationen und homogene Koordinaten



Grundlegende 2D Transformationen

- ▶ Translation eines (Orts-)Vektors

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}_h = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}_h = \begin{pmatrix} x + \Delta x \\ y + \Delta y \\ 1 \end{pmatrix}_h$$

- ▶ Translation verändert Richtungsvektoren nicht

$$\begin{pmatrix} x' \\ y' \\ 0 \end{pmatrix}_h = \begin{pmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}_h = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix}_h$$

- ▶ Achtung: Transformation von Normalen erfordert spezielle Behandlung
(gleich: Normalen transformiert man mit der Inverstransponierten)
- ▶ im Folgenden lassen wir das „ h “ weg – es ist i.d.R. aus dem Kontext klar, wann es sich um homogene Koordinaten handelt

Affine Abbildungen



Beispiel

- ▶ Rotation in 2D um z-Achse durch den Punkt $\mathbf{c} = (c_x, c_y)$ und Winkel ϕ
- ▶ Schritte: $\mathbf{T}(-\mathbf{c}) \rightarrow \mathbf{R}_z(\phi) \rightarrow \mathbf{T}(\mathbf{c})$

$$\begin{pmatrix} 1 & 0 & c_x \\ 0 & 1 & c_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos \phi & -\sin \phi & 0 \\ \sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -c_x \\ 0 & 1 & -c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Grundlegende 3D Transformationen

- ▶ Skalierung und Scherung (z unverändert)

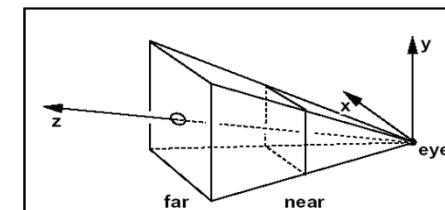
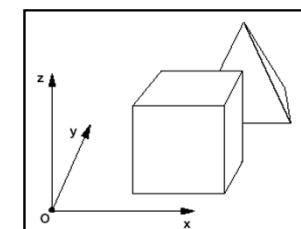
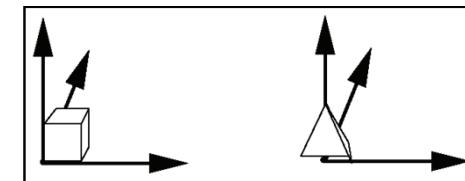
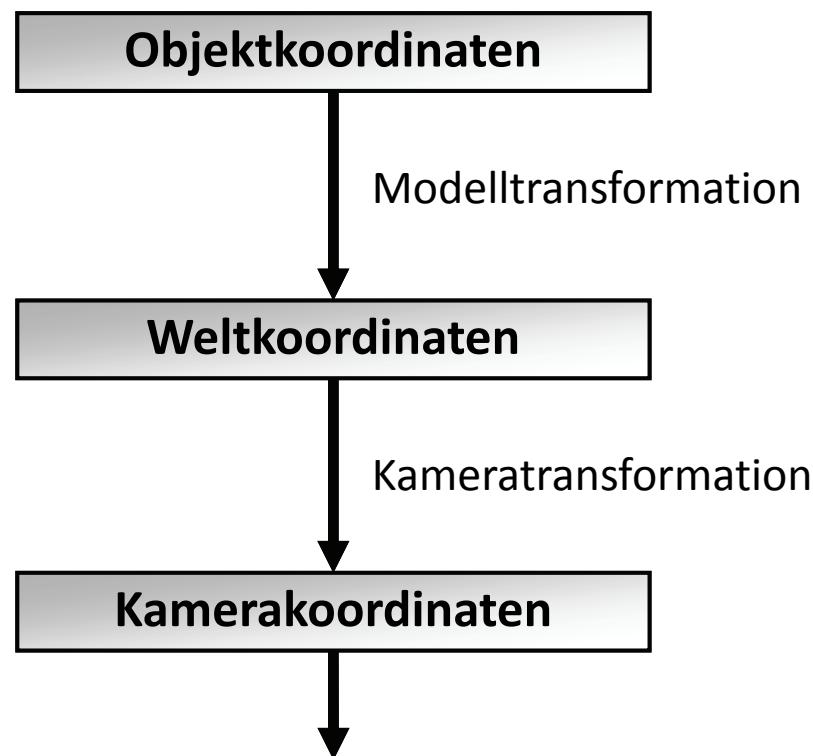
$$scale(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad shear_z(d_x, d_y) = \begin{pmatrix} 1 & 0 & d_x & 0 \\ 0 & 1 & d_y & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- ▶ Rotation um x -, y - und z -Achse

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad R_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Koordinatensysteme in der CG

- ▶ Objekte in einer Szene werden zur Modellierung (Beschreibung) in ihrem eigenen **Objekt- oder Modell-Koordinatensystem** angegeben
- ▶ die Platzierung der Objekte im **Weltkoordinatensystem** erfolgt dann durch Translation, Rotation, Skalierung etc.
- ▶ dann erfolgt – beim Ray Tracing nicht zwingend notwendig – die Transformation in das **Kamerakoordinatensystem**



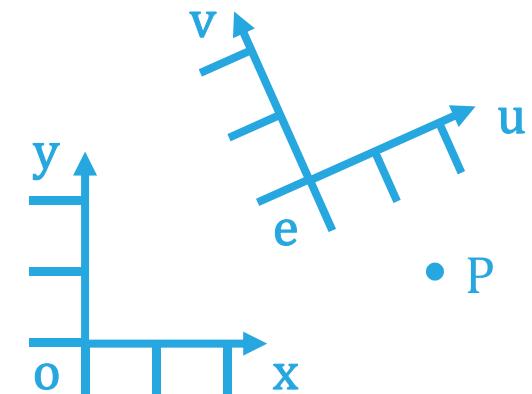
Wechsel zwischen Koordinatensystemen



- ▶ globales KoSys: Ursprung **o**, Basisvektoren **x**, **y** (nicht explizit gespeichert) ist Bezugssystem für lokale Koordinatensysteme
- ▶ lokales KoSys: Ursprung **e** und Basisvektoren **u**, **v**
 - ▶ z.B. Position eines Autos in einer virtuellen Szene
- ▶ Lage des Punktes P: $(p_x, p_y) = (5,1)$ bzw. $(p_u, p_v) = (1, -2)$
- ▶ ausgedrückt durch die Basisvektoren:
 $P = (p_x, p_y) = e + p_u u + p_v v$
- ▶ Wechsel der Koordinatensysteme

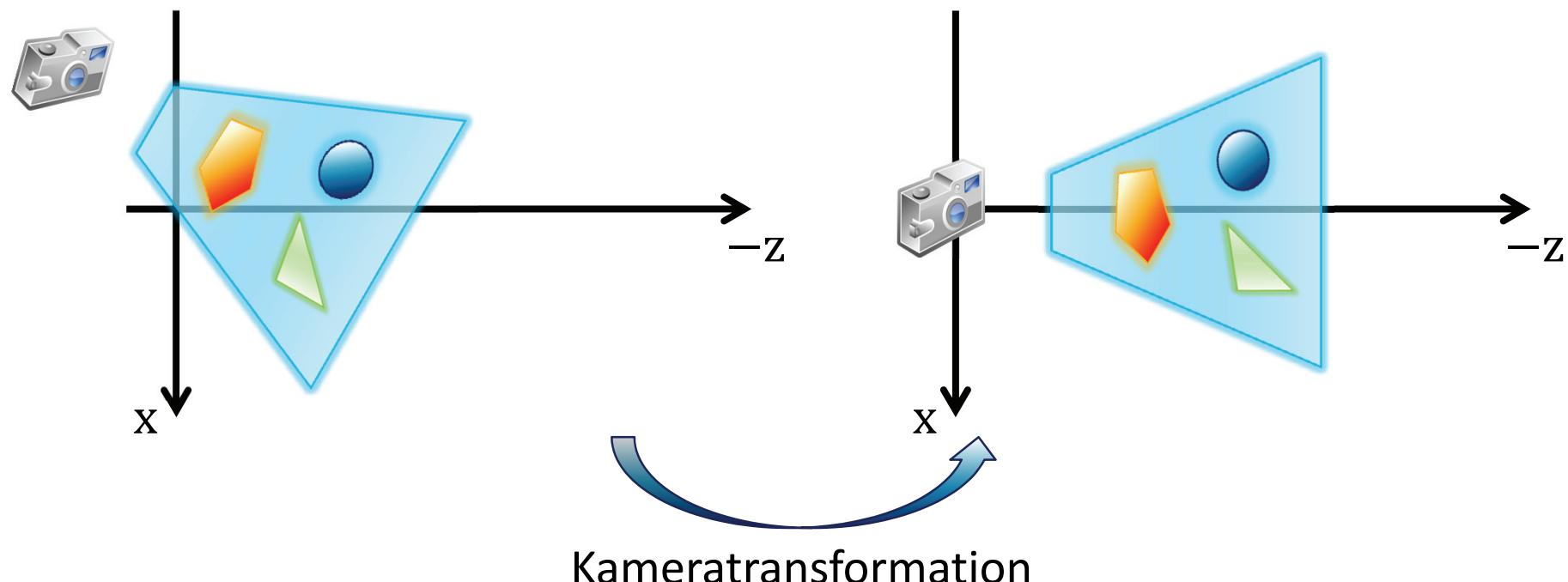
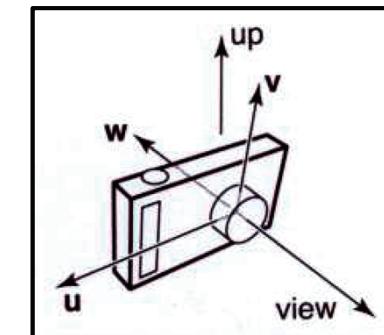
▶
$$\begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & e_x \\ 0 & 1 & e_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_x & v_x & 0 \\ u_y & v_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_u \\ p_v \\ 1 \end{pmatrix}$$

▶
$$\begin{pmatrix} p_u \\ p_v \\ 1 \end{pmatrix} = \begin{pmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & -e_x \\ 0 & 1 & -e_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} p_x \\ p_y \\ 1 \end{pmatrix}$$



Kameratransformation

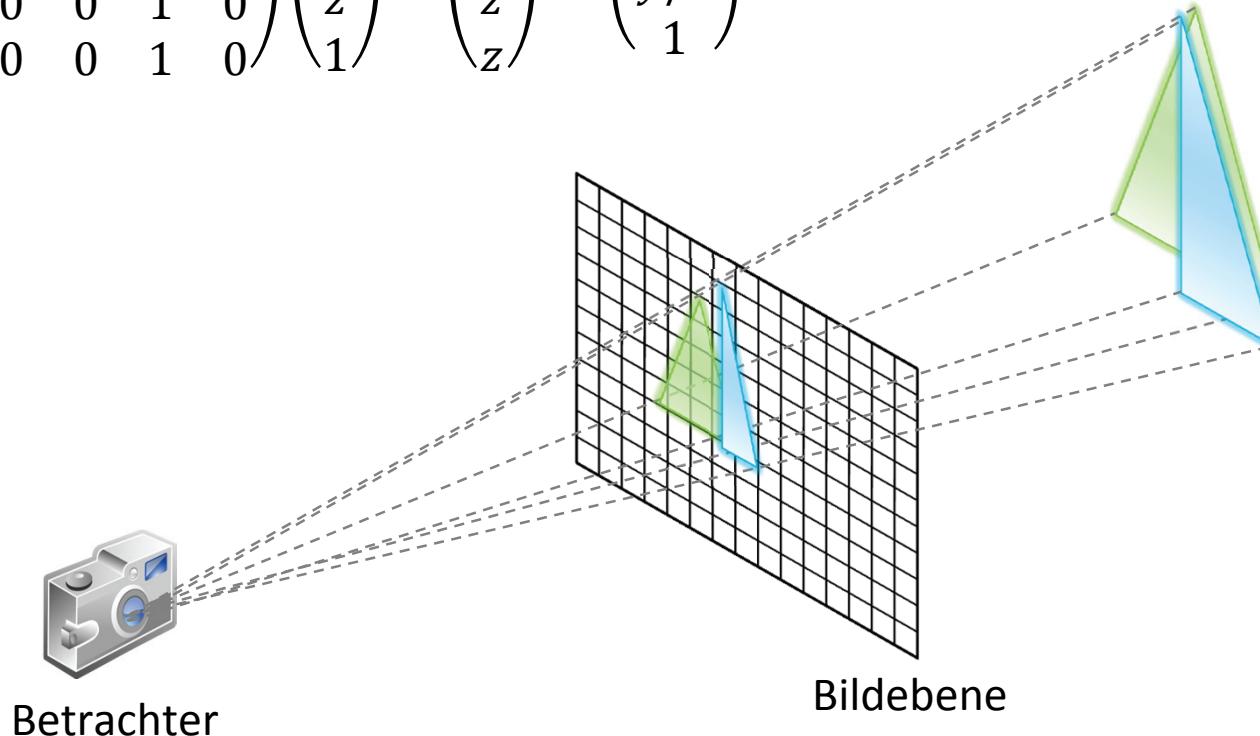
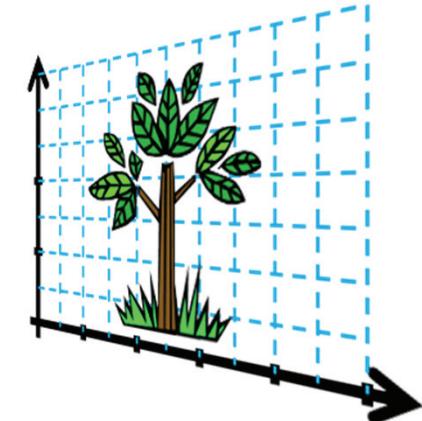
- ▶ virtuelle Kamera definiert durch
 - ▶ Position e und (negative) Blickrichtung w
 - ▶ „Up-Vektor“ up
 - ▶ $\Rightarrow u = up \times w$ und $v = w \times u$
 - ▶ u, v, w bilden Basis des Kamera-Koordinatensystems
- ▶ Transformation in dieses Koordinatensystem erleichtert bestimmte nachfolgende Schritte...



Projektive Abbildungen

- ... wie beispielsweise die Projektion auf die Bildebene
- Projektive Abbildungen besprechen wir im Kontext von Rasterisierung mit Grafik-Hardware
- Bsp. Projektion auf $z = 1$ Ebene:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ z \end{pmatrix} \mapsto \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix}$$



Hierarchisches Modellieren

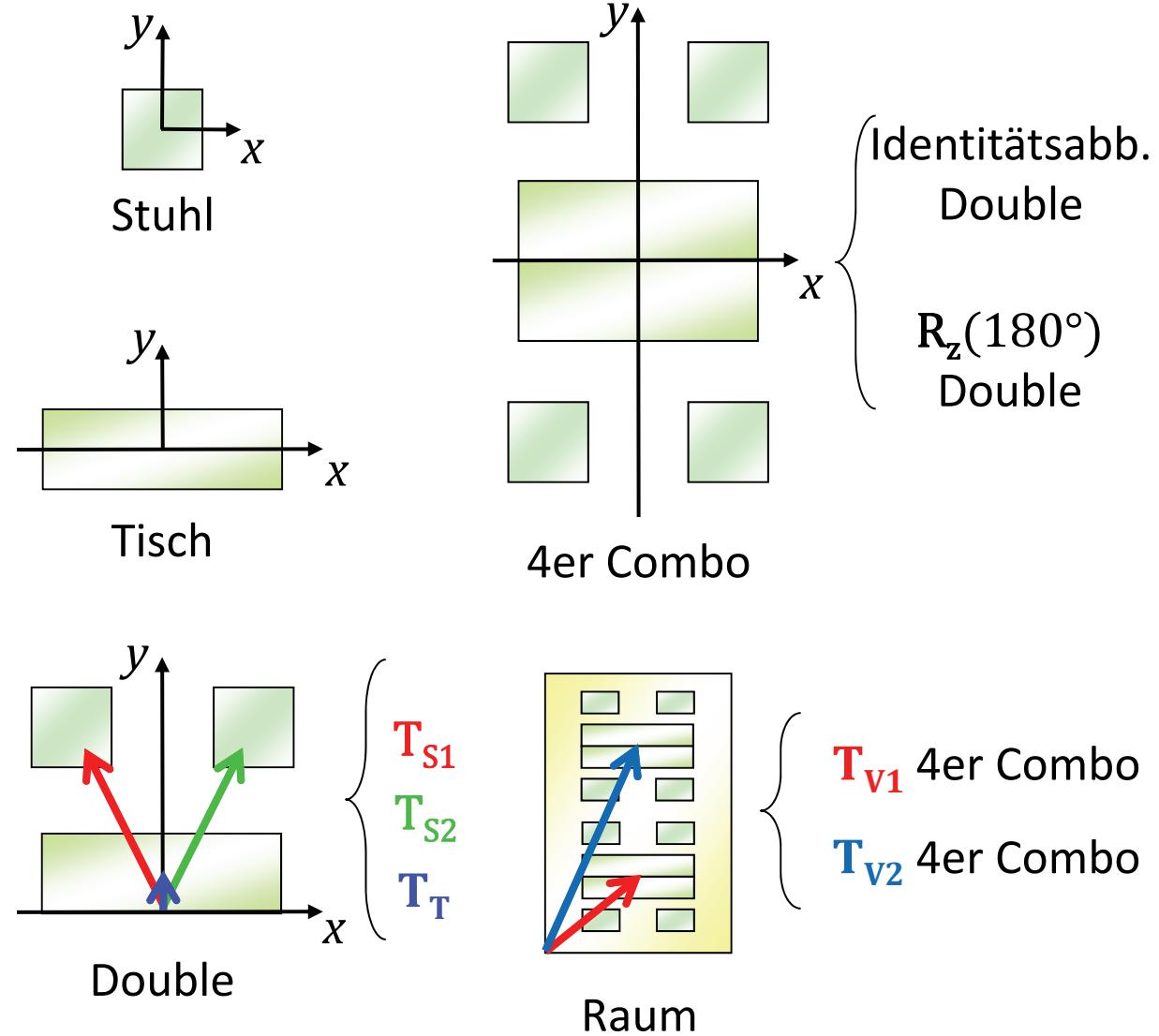
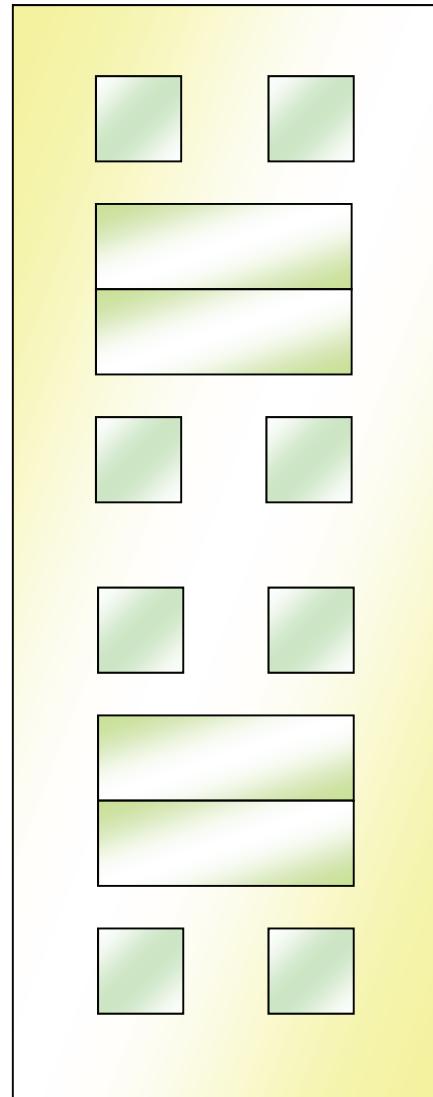


- ▶ Modelltransformationen sind typischerweise zusammengesetzte Transformationen (Rotation, Skalierung, Translation)
- ▶ aber: es macht die Modellierung komplexer Szenen einfacher, wenn man
 - ▶ mehrfache Kopien („Instanzen“) von Objekten erstellen kann
 - ▶ Objekte zu Gruppen zusammenfasst
 - ▶ Gruppen anordnet
- ▶ Beispiel: Modell eines Autos
 - ▶ ein Auto besteht aus Karosserie und 4 Rädern
 - ▶ jedes Rad besteht aus Reifen, Felge und 5 Schrauben
 - ▶ wollen Sie die Schraube 20 mal modellieren? Das Rad 4 mal? Oder die Geometrie mehrfach speichern?



Hierarchisches Modellieren

Beispiel: Platzierung von Stühlen und Tischen in einem Raum

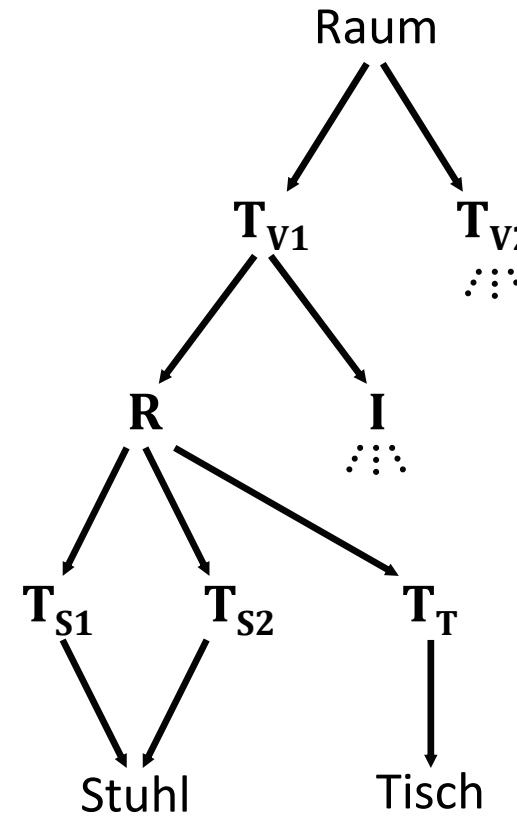
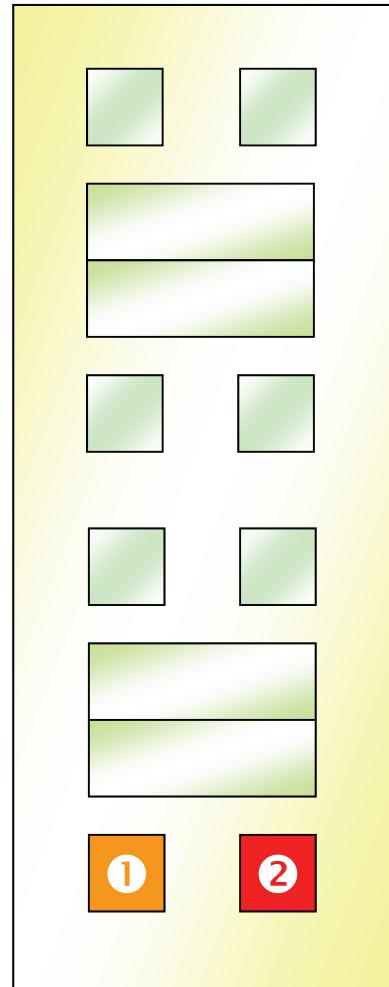


Szenengraphen: Hierarchisches Modellieren



Beispiel: Platzierung von Stühlen und Tischen in einem Raum

- es entsteht ein **Szenengraph** (gerichteter azyklischer Graph)



Transformationen
des Stuhls ① in
Weltkoordinaten:

$$(T_{V1} R_{180} T_{S2})$$

Transformationen
des Stuhls ② in
Weltkoordinaten:

$$(T_{V1} R_{180} T_{S1})$$

Szenengraphen: Matrix Stacks

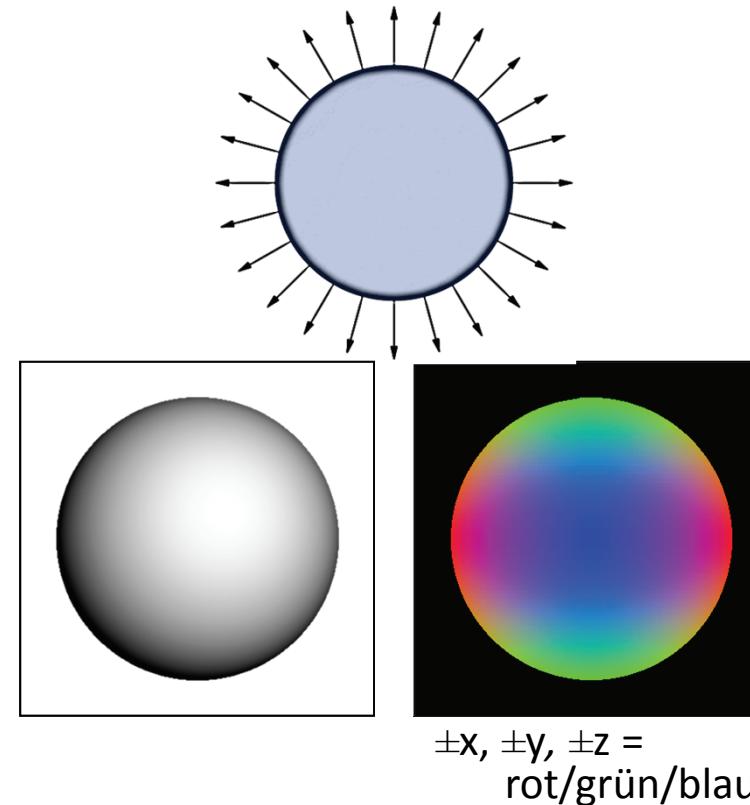
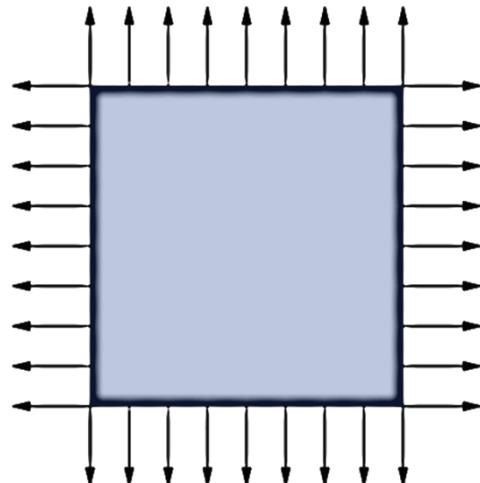


- ▶ Szenengraphen erlauben effiziente Operationen in komplexen Szenen
 - ▶ u.a. die Wiederverwendung von Matrizen mittels **Matrix Stacks**
 - ▶ die jeweils oberste Matrix definiert die aktuelle Transformation

Load(I)	Push	Mult(T_{V1})	Push	Mult(R₁₈₀)	Push	Mult(T_{S1})	Stuhl	Pop	...
I	I	I T_{V1}	I T_{V1}	I T_{V1} R	I T_{V1} R	I T_{V1} R T_{S1}		I T_{V1} R	
	I	I	I T_{V1}	I T_{V1}	I T_{V1} R	I T_{V1} R		I T_{V1}	
			I	I	I T_{V1}	I T_{V1}		I	
					I	I			

Transformation von Normalen

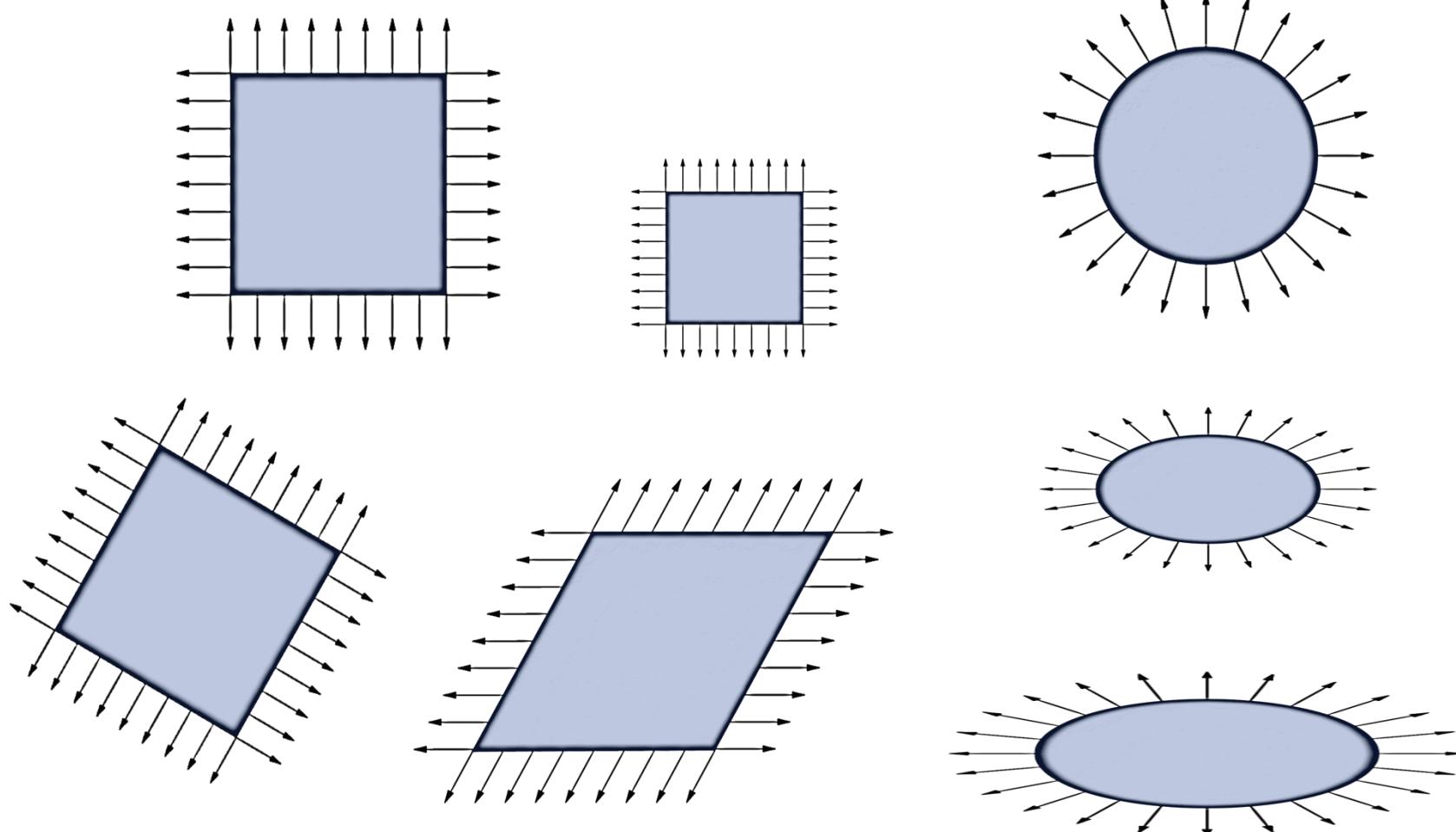
- Oberflächennormale/Normalenvektor:
Einheitsvektor lokal-senkrecht zur Oberfläche
 - ▶ wichtig für Schattierung/Beleuchtung
 - ▶ hier: verschiedene Visualisierungen



Transformation der Normalen wie Objekte?

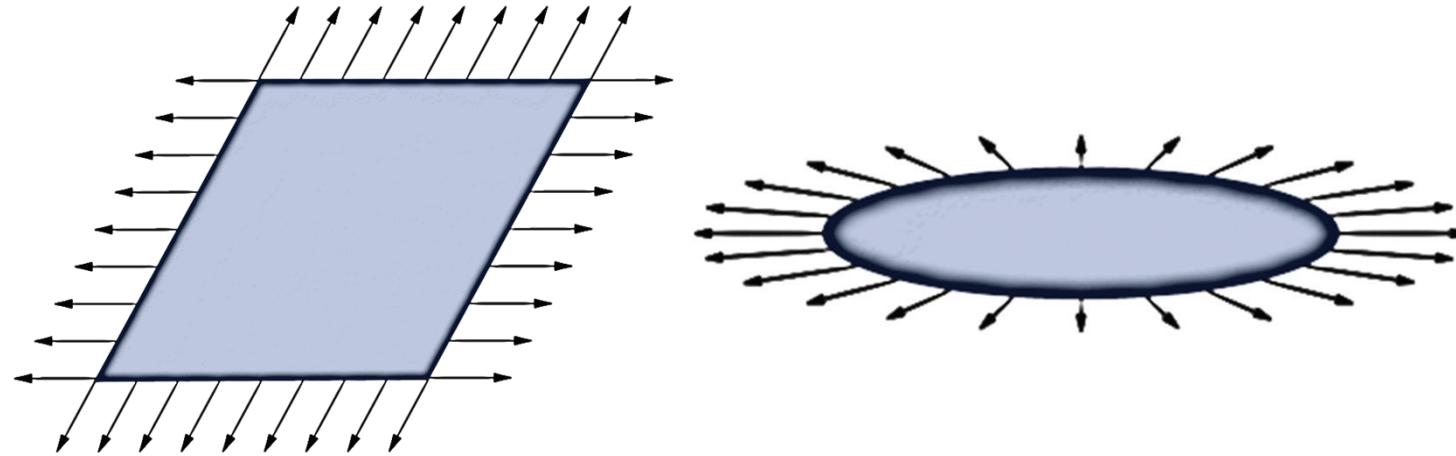


- ▶ Translation, Rotation, (isotr.) Skalierung, Spiegelung, Scherung, ...?

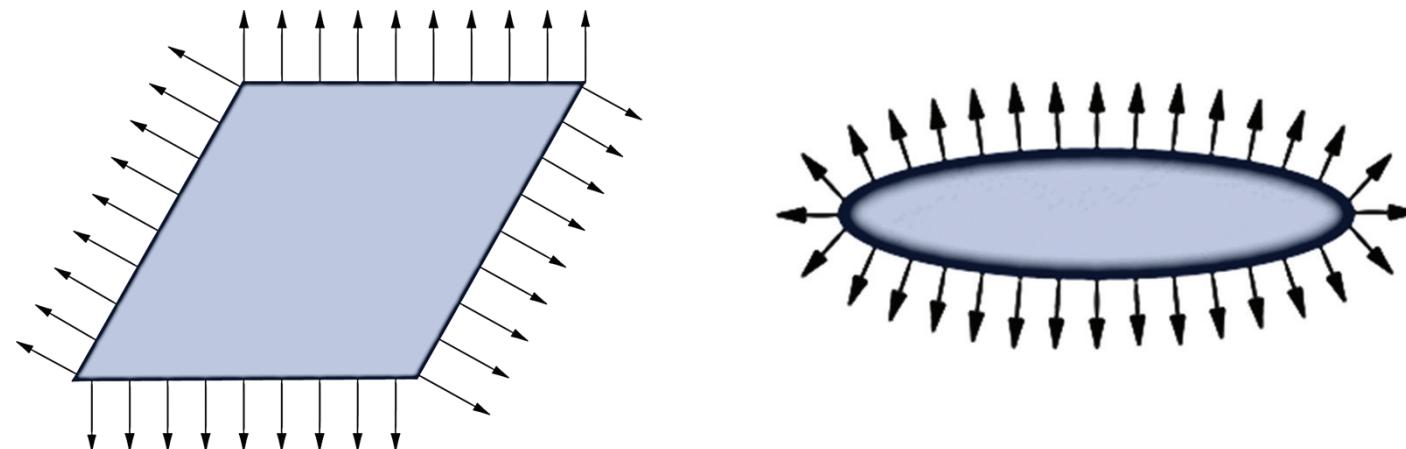


Transformationen für Scherung und Skalierung

- ▶ falsche Transformation der Normalenvektoren

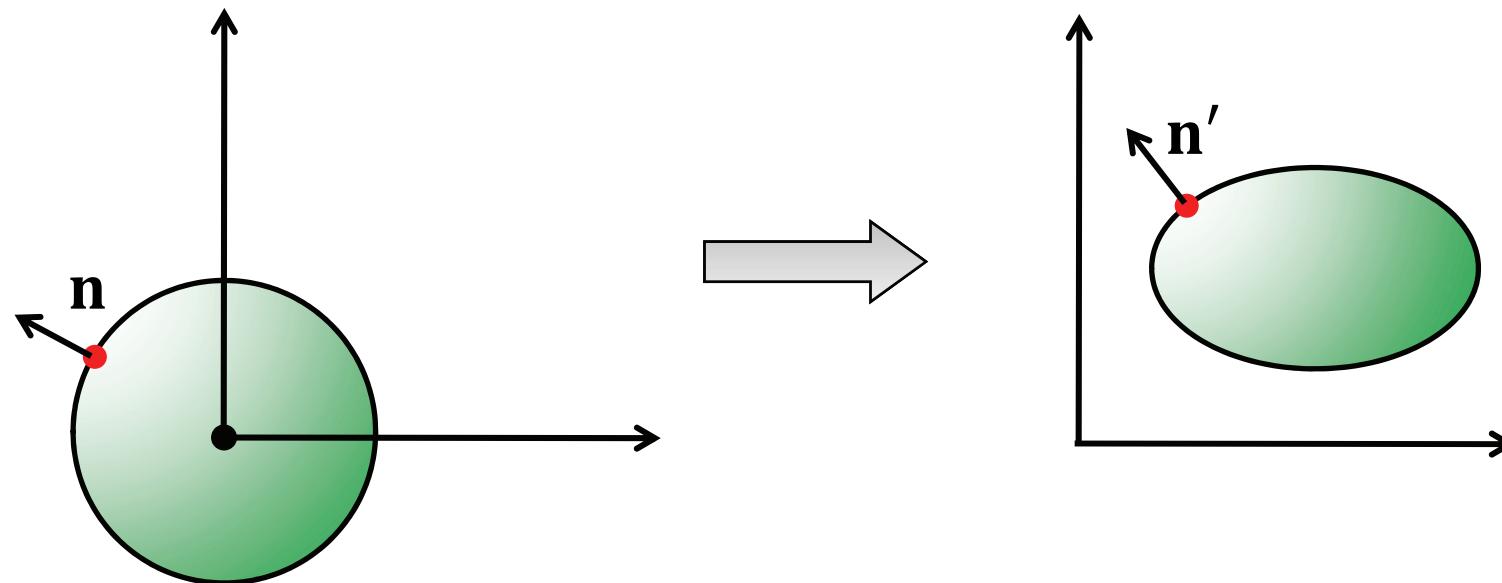


- ▶ korrekte Transformation der Normalenvektoren



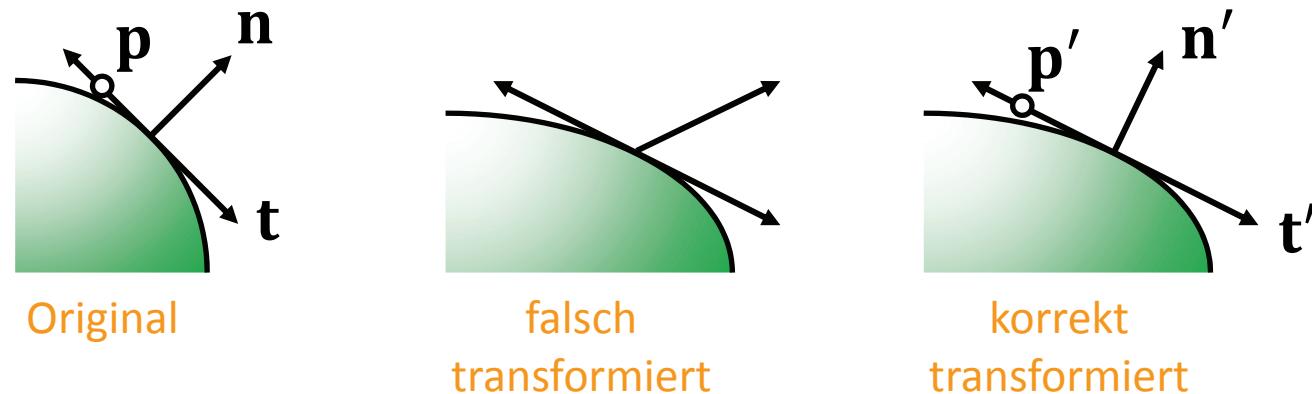
Transformation von Normalen

- ▶ lineare und affine Transformationen sind i.A. nicht winkeltreu
→ Normalen können nicht einfach mit-transformiert werden
- ▶ wie macht man es also richtig?



Transformation von Normalen

- wir verwenden homogene Koordinaten und transformieren die Tangentenebene zur Normale, nicht den Normalenvektor selbst
(Normalenvektoren sind so definiert, nicht als Differenz zweier Punkte)



- Tangentialebene E (in HNF): $n_x x + n_y y + n_z z + d = 0$
 - „Normalenvektor“ $\mathbf{n} = (n_x, n_y, n_z, d)$
 - Punkt $\mathbf{p} \in E$ in der Ebene $\mathbf{p} = (x, y, z, 1)$
 - $\mathbf{p} \in E \Leftrightarrow \mathbf{n}^T \mathbf{p} = 0$ (Skalarprodukt mit homogenen Koord.)
 - der Punkt $\mathbf{p}' = \mathbf{M}\mathbf{p}$ liegt auf der transformierten Tangentialebene und die transformierte Normale \mathbf{n}' soll darauf senkrecht stehen

Transformation von Normalen



Transformation von Normalen



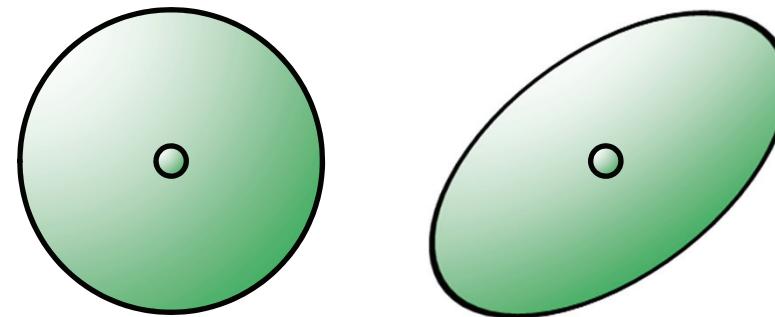
- ▶ Transformation des Normalenvektors: $\mathbf{n}' = (\mathbf{M}^{-1})^T \mathbf{n}$
 - ▶ beachte: es ist möglich das $\|\mathbf{n}'\| \neq 1$, z.B. bei nicht-uniformen Skalierungen oder Scherungen (d.h. nur Richtung von \mathbf{n}' ist korrekt)
- ▶ warum „funktioniert“ $\mathbf{n}' = \mathbf{M}\mathbf{n}$ für Ähnlichkeitsabbildungen (= euklidische Transformation plus Skalierung)?
 - ▶ es gilt in diesem Fall: $(\mathbf{M}^{-1})^T = \lambda \mathbf{M}$
(erhält zumindest die Richtung)
- ▶ bei einer Matrix mit orthonormaler Basis gilt $(\mathbf{M}^{-1})^T = \mathbf{M}$

$$\mathbf{M} = \begin{pmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} \quad \mathbf{M}^{-1} = \begin{pmatrix} u_x & v_x & w_x \\ u_y & v_y & w_y \\ u_z & v_z & w_z \end{pmatrix}$$

Transformationen und Schnitttests



- ▶ Modelltransformationen sind typischerweise zusammengesetzte Transformationen – aber: wie berechnen wir Strahl-Objekt Schnitte?
- ▶ Rotation, isotrope Skalierung, Translation sind einfach
 - ▶ Dreieck: Transformation der 3 Eckpunkte
 - ▶ Kugel: Veränderung des Kugelmittelpunkts und -radius
- ▶ Scherung, allgemeine Skalierung
 - ▶ Dreieck: kein Problem, Transformation der Eckpunkte
 - ▶ Kugel: implizite Beschreibung einer gescherten Kugel?

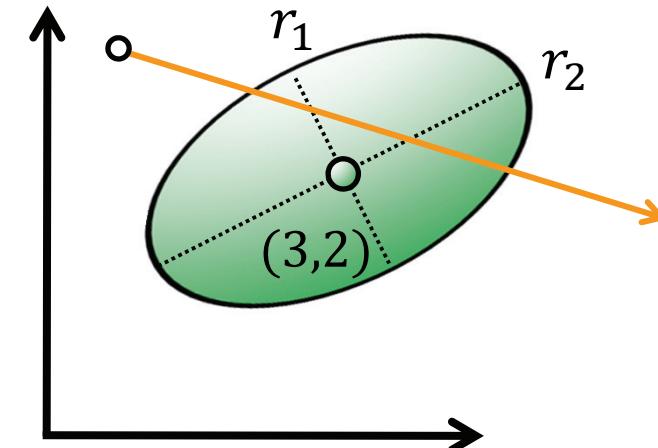


- ▶ bessere Alternative: **Schnittpunktberechnung in Modellkoordinaten**

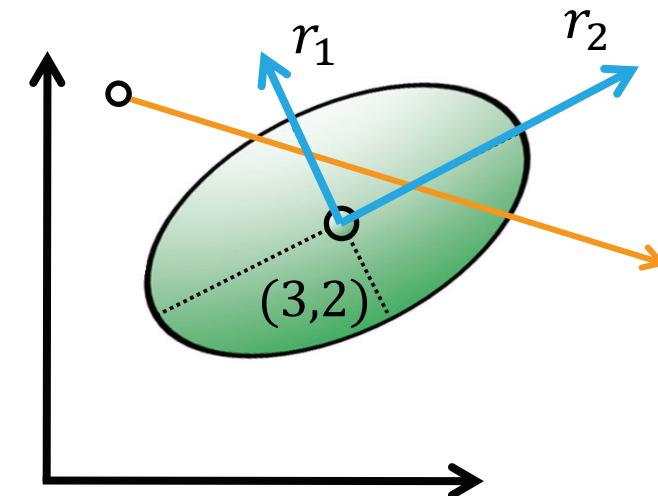
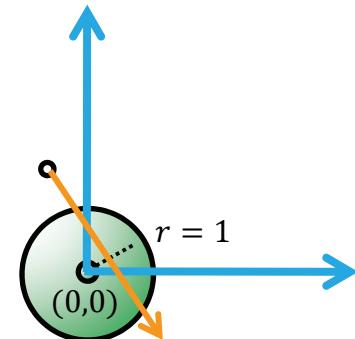
Transformationen und Schnitttests

Zwei Möglichkeiten

- Schnittpunktberechnung in Weltkoordinaten:
jedes Primitiv „handhabt“ Transformationen selbst
(oft kompliziert)

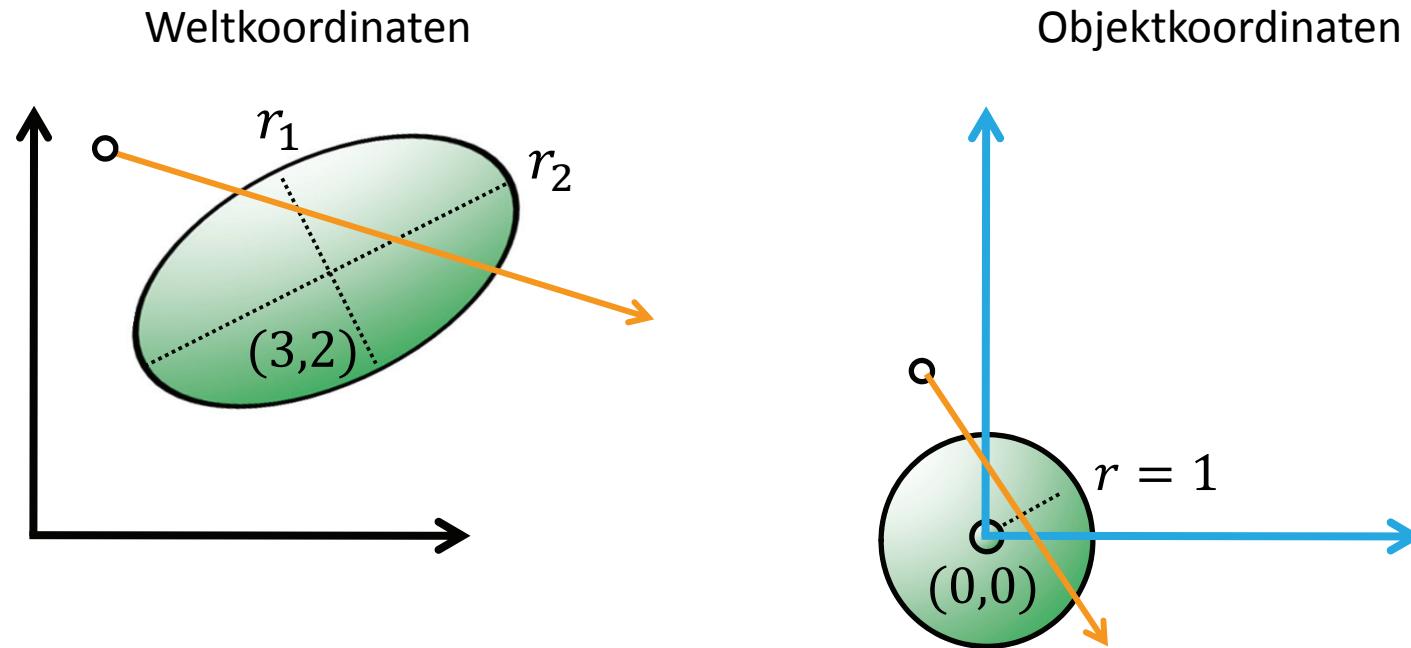


- Schnittpunktberechnung in Modellkoordinaten



Transformation des Strahls

- ▶ transformiere den Strahl von Welt- in Objektkoordinaten



- ▶ involvierte Transformationen
 - ▶ in Weltkoordinaten (world space): $\mathbf{p}_{ws} = \mathbf{M}\mathbf{p}_{os}$
 - ▶ in Objektkoordinaten (object space): $\mathbf{p}_{os} = \mathbf{M}^{-1}\mathbf{p}_{ws}$

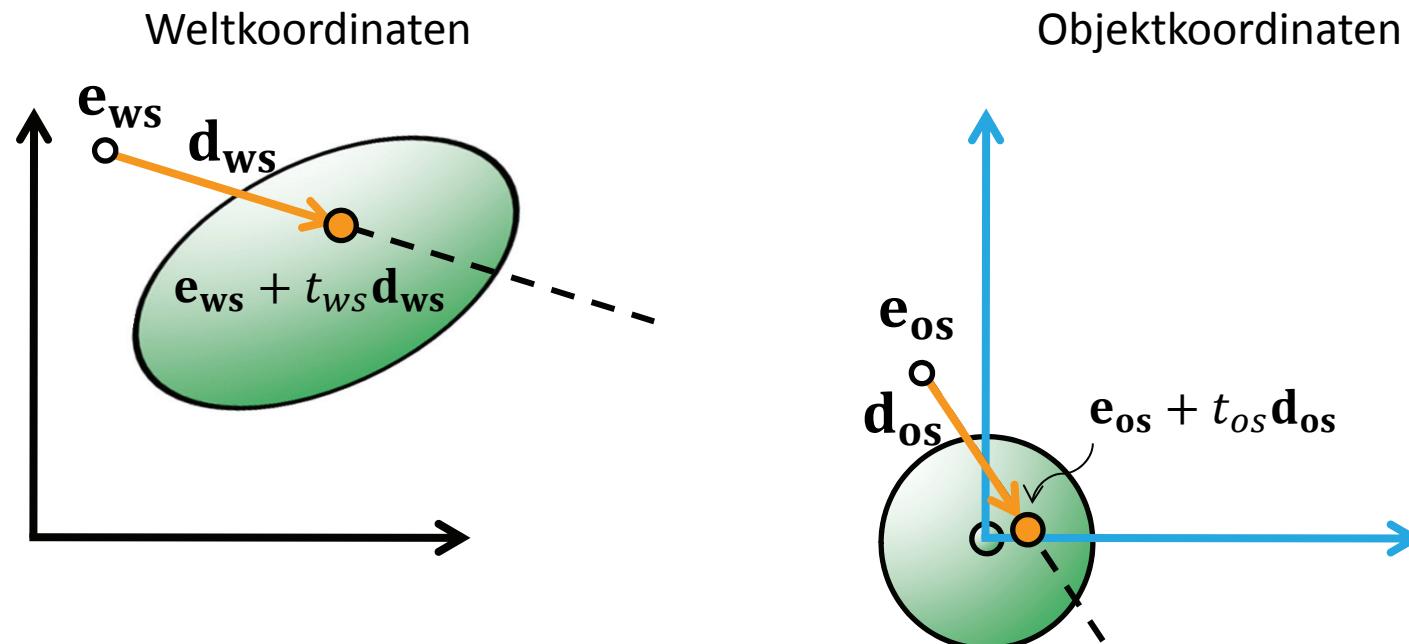
Transformation des Strahls

- neuer Strahlursprung

$$\mathbf{e}_{os} = \mathbf{M}^{-1} \mathbf{e}_{ws}$$

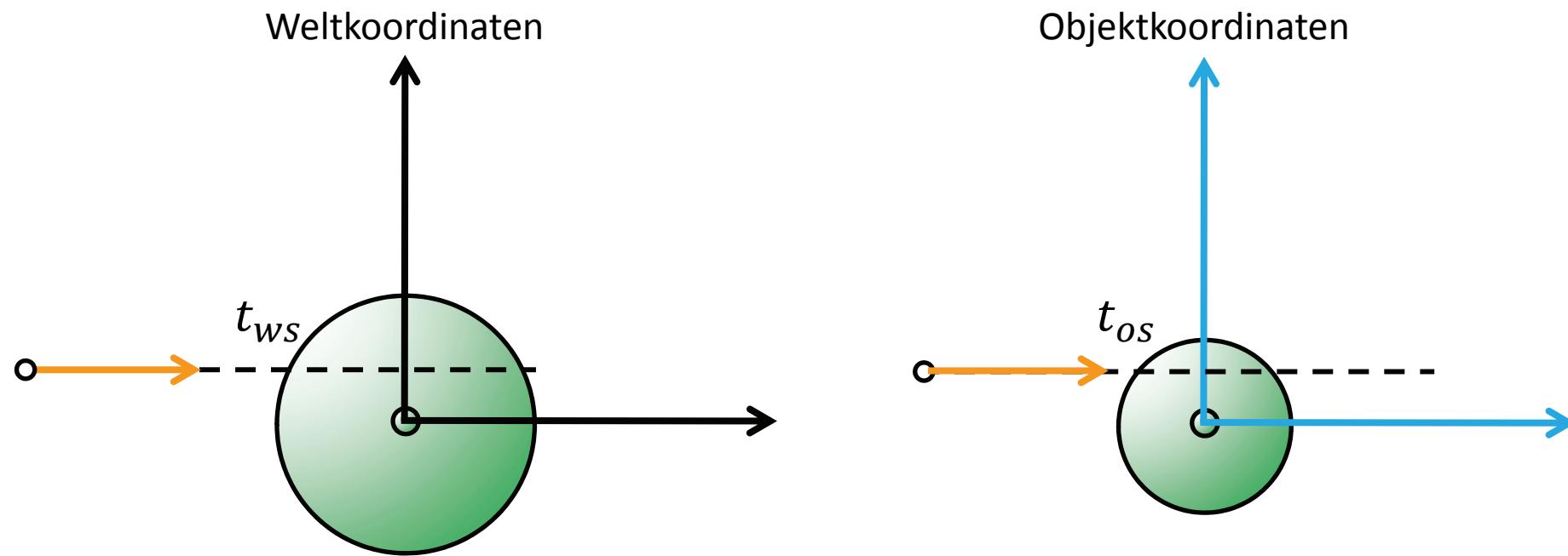
- neue Strahlrichtung (Richtung = Differenz zweier Punkte, daher dürfen wir hier direkt die inverse Matrix verwenden)

$$\begin{aligned}\mathbf{d}_{os} &= \mathbf{M}^{-1}(\mathbf{e}_{ws} + \mathbf{d}_{ws}) - \mathbf{M}^{-1} \mathbf{e}_{ws} \\ \mathbf{d}_{os} &= \mathbf{M}^{-1} \mathbf{d}_{ws}\end{aligned}$$



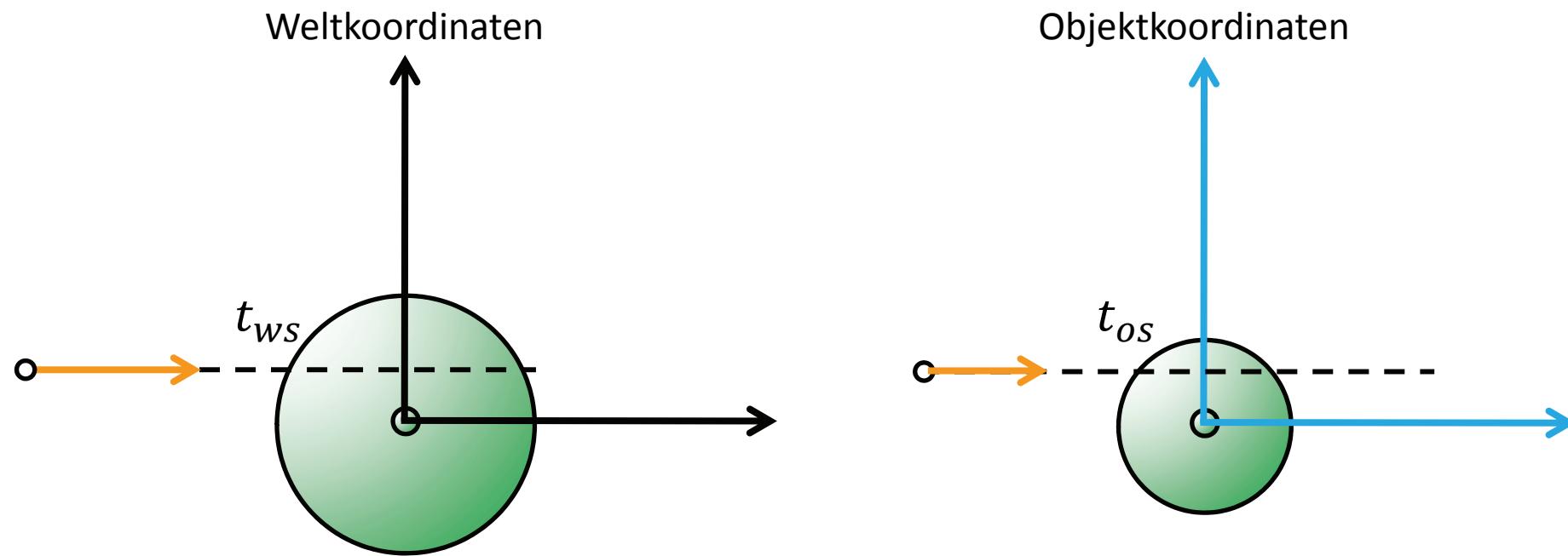
Transformation des Strahls

- ▶ wenn \mathbf{M} z.B. eine Skalierung enthält, dann ist \mathbf{d}_{os} nicht normalisiert
- ▶ wieder zwei Optionen
 - ▶ normalisiere \mathbf{d}_{os} , also verwende $\mathbf{d}_{os}/|\mathbf{d}_{os}|$
 - ▶ normalisiere nicht und verwende \mathbf{d}_{os} für die Schnittberechnung
- ▶ Fall 1: verwende $\mathbf{d}_{os}/|\mathbf{d}_{os}|$
 - ▶ $t_{ws} \neq t_{os}$ (Skalierung nach der Schnittberechnung notwendig)



Transformation des Strahls

- ▶ wenn \mathbf{M} eine Skalierung enthält, dann ist \mathbf{d}_{os} nicht normalisiert
- ▶ wieder zwei Optionen
 - ▶ normalisiere \mathbf{d}_{os} , also verwende $\mathbf{d}_{os}/|\mathbf{d}_{os}|$
 - ▶ normalisiere nicht und verwende \mathbf{d}_{os} für die Schnittberechnung
- ▶ Fall 2: verwende \mathbf{d}_{os} (wenn Schnittberechnung es zulässt)
 - ▶ $t_{ws} = t_{os}$
 - ▶ aber: t_{os} ist nicht der euklidische Abstand



Fazit

- ▶ ist (implizite) Beschreibung des mit \mathbf{M} transformierten Objekts schwierig
→ transformiere den Strahl mit \mathbf{M}^{-1} und führe Schnitttest mit der ursprünglichen Beschreibung durch
- ▶ meist günstiger als die Objekte zu transformieren
(auch wenn es bei Dreiecksnetzen einfach möglich ist)
- ▶ einfacher/effizienter, wenn Instanziierung verwendet wird
- ▶ Kosten der Matrixinversion und Strahltransformation?
 - ▶ trotzdem nicht vernachlässigbar → also nur, wenn notwendig
 - ▶ Ausnutzen der Modellhierarchie/des Szenengraphen
 - ▶ einzelne Transformationen sind einfach umkehrbar
 - ▶ Wiederverwenden von Matrizen
 - ▶ einmalige Strahltransformation für viele Objekte/Objektgruppen, z.B. ein Dreiecksnetz

Transformationen und Schnitttests

- ▶ Strahlerzeugung: typischerweise in Welt- oder Kamerakoordinaten
- ▶ Schnittpunktberechnung kann in jedem Koordinatensystem stattfinden

