

Transformationen



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Visual Computing

Winter Semester 2023-2024

Prof. Dr. A. Kuijper

Mathematical and Applied Visual Computing (MAVC)

Graphisch-Interaktive Systeme (GRIS)

Fraunhofer IGD

Fraunhoferstrasse 5

D - 64283 Darmstadt

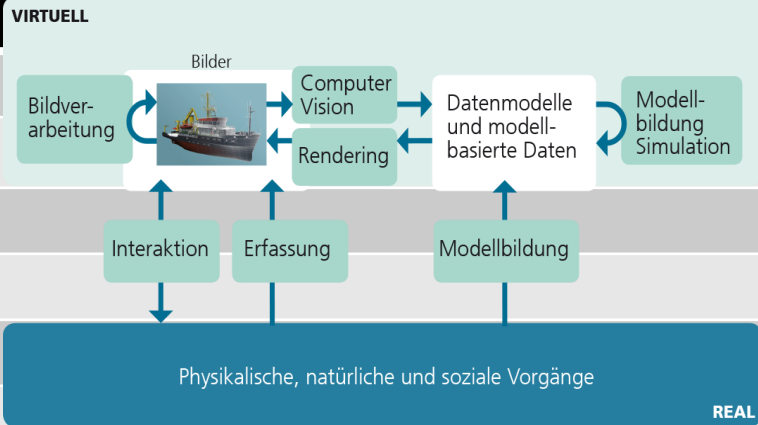
E-Mail: office@gris.tu-darmstadt.de

<http://www.gris.tu-darmstadt.de>

<https://www.mavc.tu-darmstadt.de>

$$342 < 529 < 552$$

Semesterplan

Datum		<div> <div>VIRTUELL</div>  <div>REAL</div> </div>
20. Okt	Einführung + Visual Computing	
27. Okt	Wahrnehmung	
03. Nov	Objekterkennung und Bayes	
10. Nov	Fourier Theorie	
17. Nov	Bilder	
24. Nov	Bildverarbeitung	
01. Dez	Grafikpipeline & Eingabemodalitäten & VR+AR	
08. Dez	Transformationen & 2D/3D Ausgabe	
15. Dez	3D-Visualisierung	
12. Jan	X3D – 3D in HTML	
19. Jan	Informationsvisualisierung	
26. Jan	User Interfaces + Multimedia Retrieval	
02. Feb	Farbe	
11. Feb	Biometrie (?)	

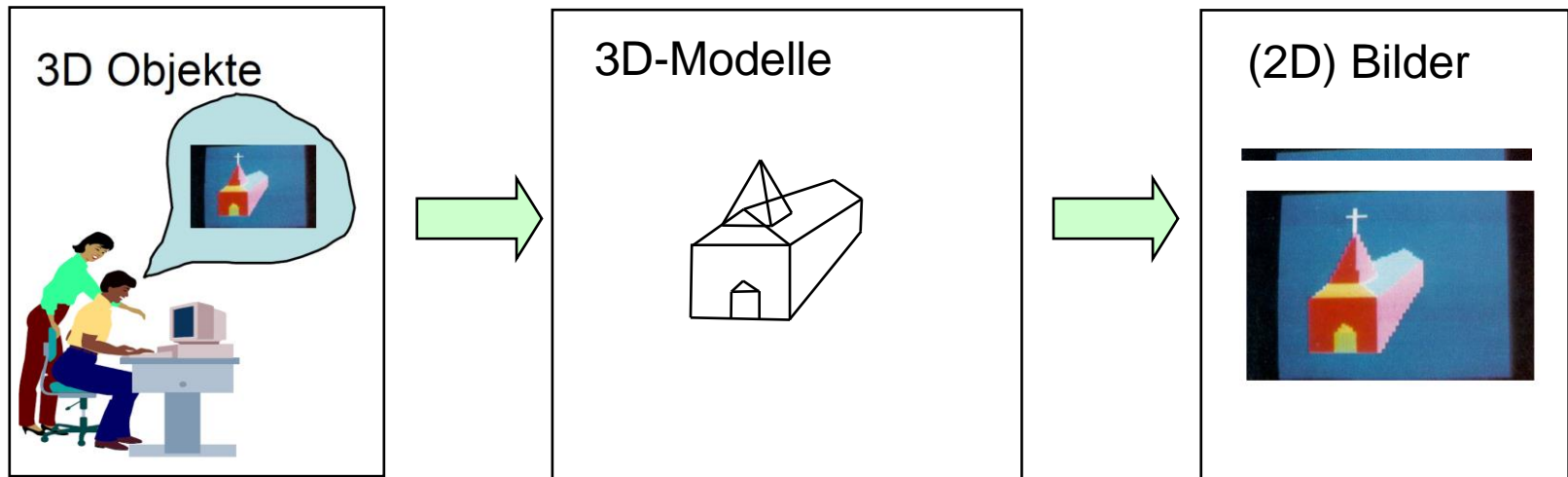


- **Wiederholung: Grafikpipeline**
- Transformationen und affine Abbildungen
- Skalierung, Scherung, Rotation
- Projektion
- 3D-Interaktion

3D-Grafik-Pipeline - Schematisch

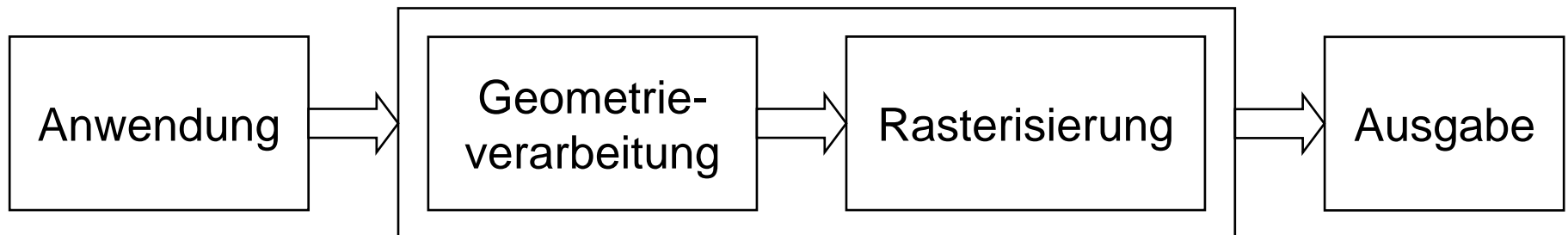
Modellierung

Darstellung



CPU

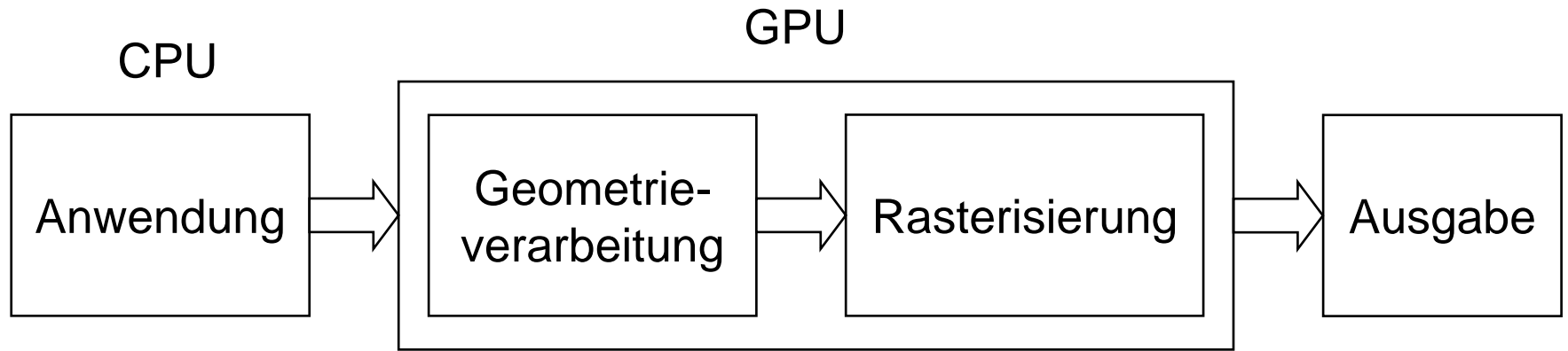
GPU



3D-Grafikpipeline

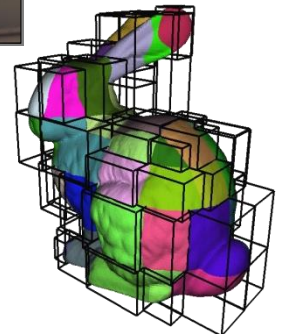
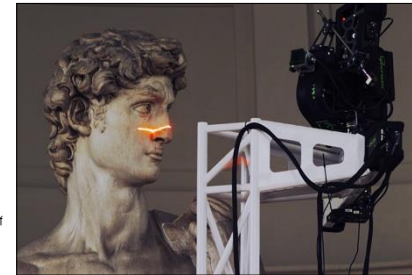
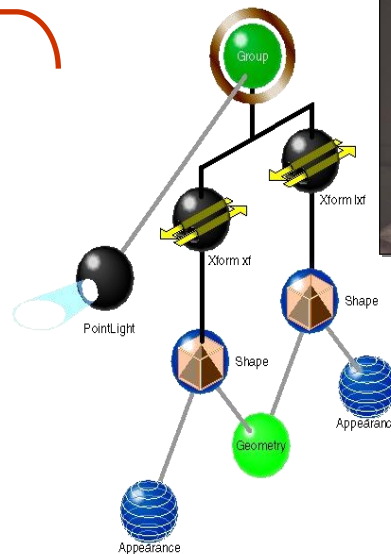


TECHNISCHE
UNIVERSITÄT
DARMSTADT




Eingabe graphischer Daten
Repräsentation von 3D Daten

- Grafische Primitive
- Transformationen
- Räumliche Datenstrukturen





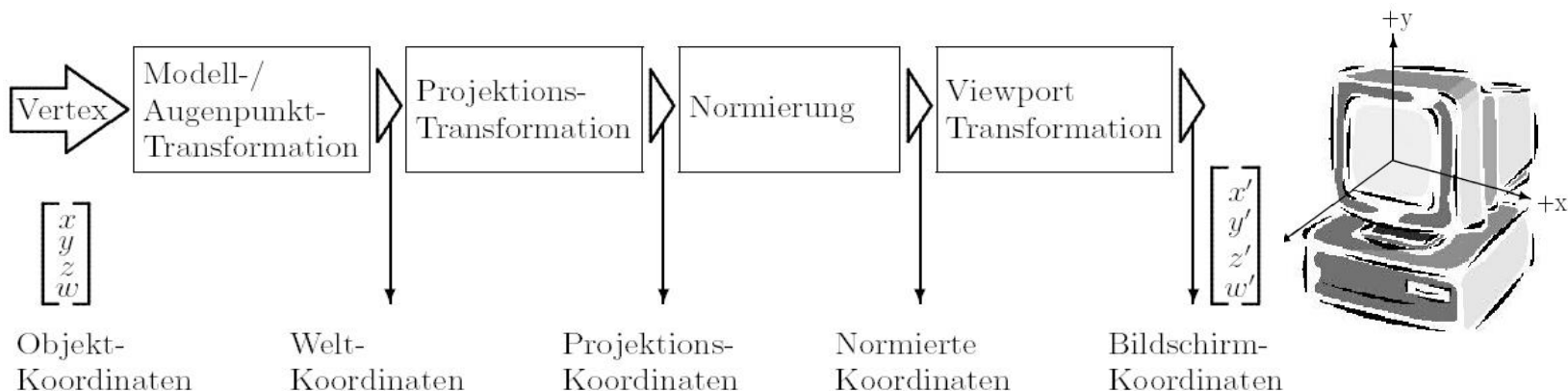
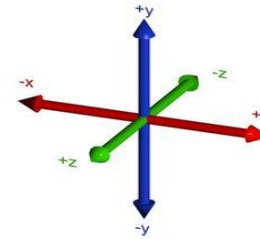
- Wiederholung: Grafikpipeline
- **Transformationen und affine Abbildungen** 
- Skalierung, Scherung, Rotation
- Projektion
- 3D-Interaktion

Koordinaten aus Sicht der Grafikpipeline



TECHNISCHE
UNIVERSITÄT
DARMSTADT

1. *Objektkoordinaten* (object coordinates): legen Lage von 3D-Objekten lokal fest
2. *Weltkoordinaten* (eye coordinates): beschreiben die gesamte Szene in 3D
3. *Projektionskoordinaten* (clip coordinates): erhält man nach Anwendung der Projektionstransformation (parallel oder perspektivisch)
4. *Normierte Koordinaten* (normalized device coordinates)
5. *Bildschirmkoordinaten* (window coordinates): stellen Szene in Fenster einer gewählten Größe und Position dar



Transformationen in der Grafikpipeline

Überblick und Vergleich mit „Fotografieren“



TECHNISCHE
UNIVERSITÄT
DARMSTADT

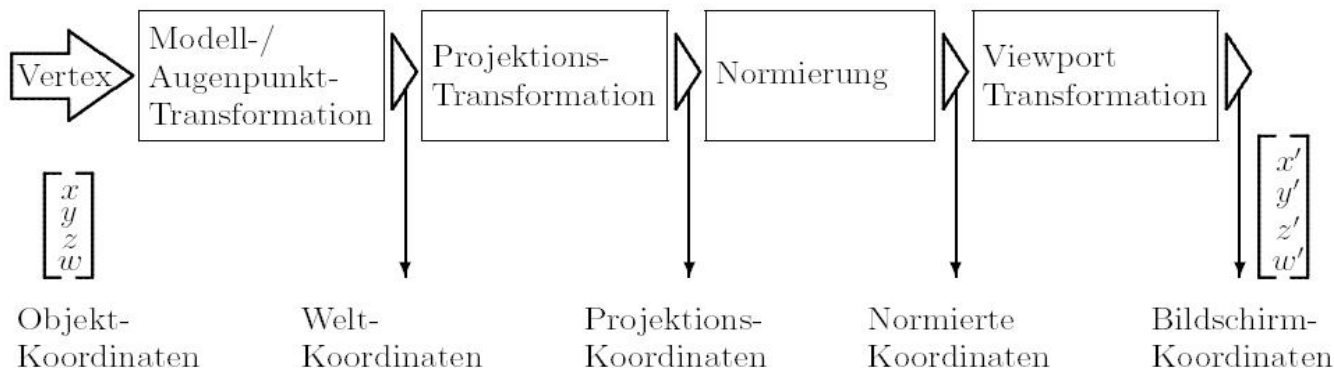
1. *Modelling* Transformations: ordne 3D-Objekte (Modelle) im Raum an und positioniere diese
Viewing Transformations: wähle Betrachterstandpunkt und positioniere diesen (default: Position: Ursprung & Blickrichtung: negative z-Achse)
2. *Projection* Transformations: projiziere Viewing Volume (sichtbarer Ausschnitt der Szene) in 2D
3. Normalizing Transformations: Alles zwischen 0 und 1
4. *Viewport* Transformations: wandle in Bildschirmkoordinaten um

Bereite Foto vor: ordne Personen und Objekte an

Positioniere die Kamera und justiere diese auf einem Stativ

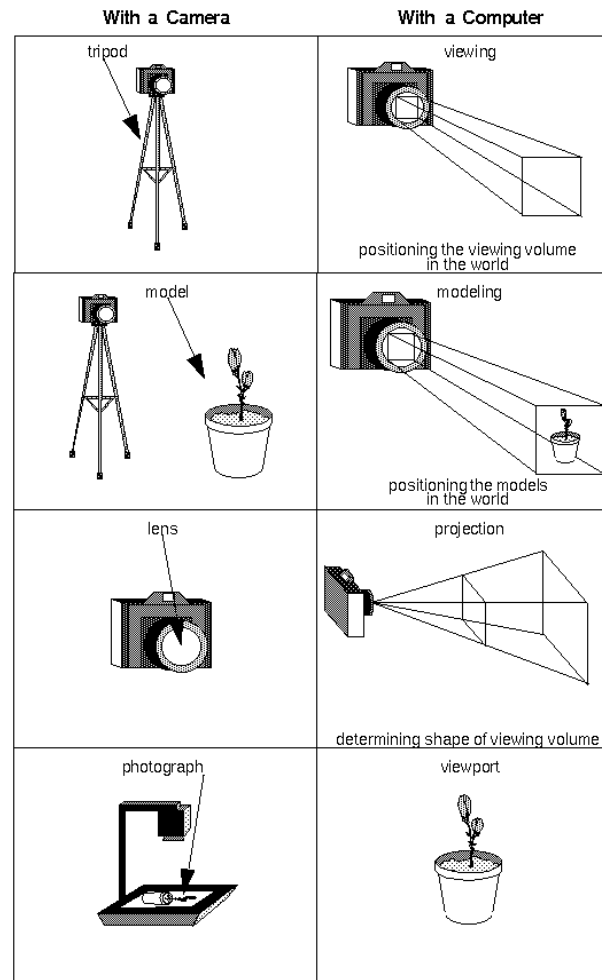
Zoom, um den gewünschten Bildausschnitt festzulegen

Bestimme das Format in dem das Bild ausgedruckt werden soll



Transformationen in der Grafikpipeline

Vergleich mit „Fotografieren“



Transformationen

Positionierung von Objekten und Primitiven

Absolute Koordinaten?



Transformationen

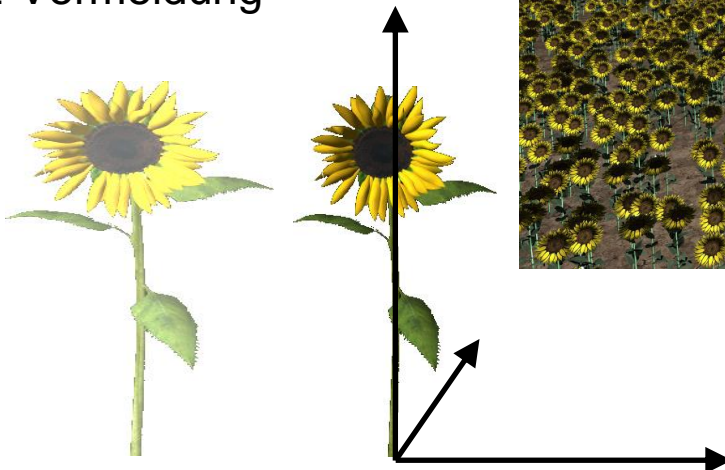
Positionierung von Objekten und Primitiven

Absolute Koordinaten?

- 28.000 Sonnenblumen
- 11 verschiedene Modelle
- Je 35.000 Dreiecke

Transformation von 3D-Objekten

- Flexibilität
- verschiedene Instanzen eines 3D-Objekts
- Redundanz-Vermeidung

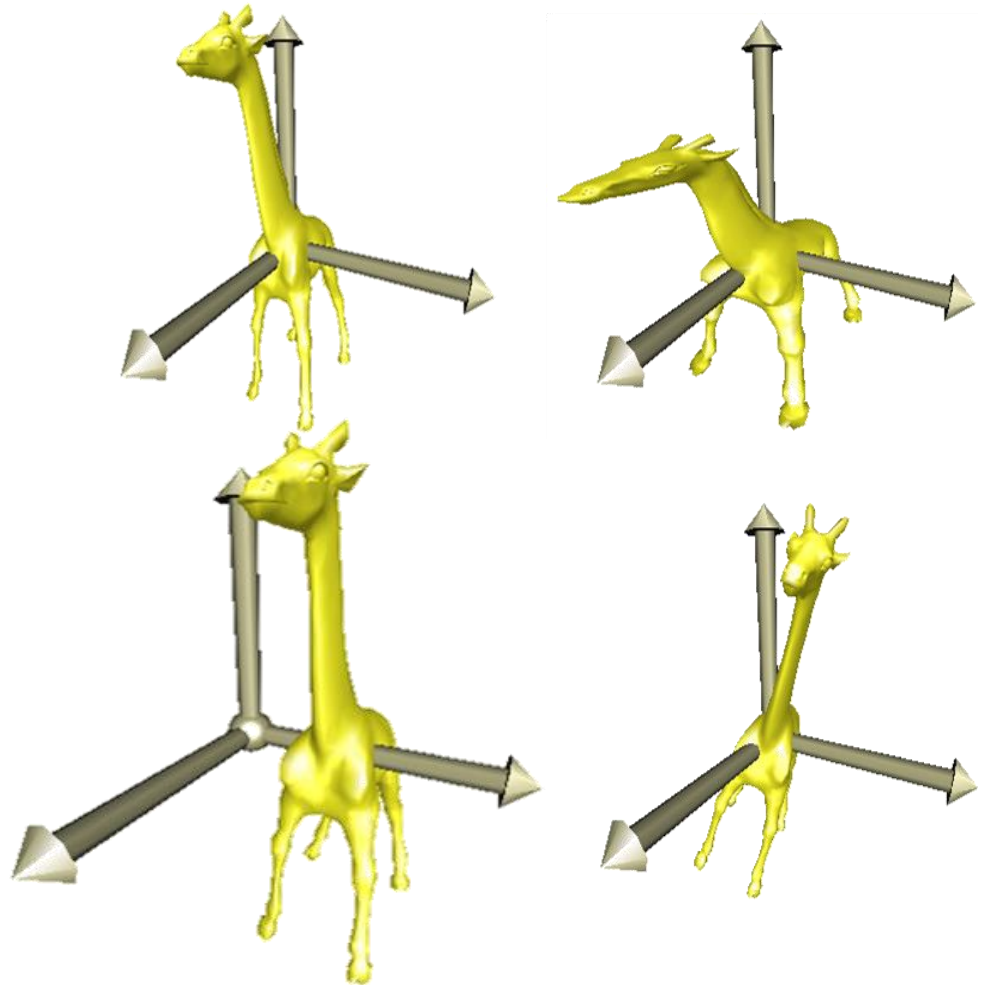


Transformationen

Positionierung von Objekten und Primitiven

Affine Transformation/Abbildung

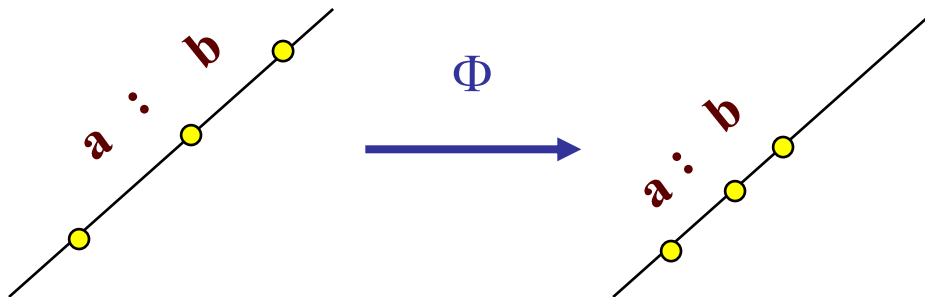
- Translation
- Rotation
- Skalierung
- Scherung



Affine Abbildungen

Eigenschaften

1. Bilden Geraden auf Geraden ab.
2. Beschränkte Objekte bleiben beschränkt.
3. Verhältnisse von Längen, Flächen, Volumen bleiben erhalten.
4. Parallele Objekte (Geraden, Ebenen, ...) bleiben parallel



Affine Abbildungen

...als lineare Abbildung

Eine Abbildung $\Phi: \mathbb{R}^n \rightarrow \mathbb{R}^n$ heißt affine Abbildung,

wenn Φ in der Form

$$\Phi(\mathbf{v}) = A(\mathbf{v}) + I(\mathbf{b})$$

darstellbar ist, wobei A, I lineare Abbildungen, I die Identitätsabbildung und $\mathbf{v}, \mathbf{b} \in \mathbb{R}^n$ sind.

A heißt lineare Abbildung, wenn gilt:

$$A(\lambda \mathbf{u} + \mu \mathbf{v}) = \lambda A(\mathbf{u}) + \mu A(\mathbf{v})$$

für alle $\mathbf{u}, \mathbf{v} \in \mathbb{R}^n$ und für alle $\lambda, \mu \in \mathbb{R}$

Affine Abbildungen

...als lineare Abbildung

Affine Abbildungen setzen sich aus einer allgemeinen linearen Abbildung (dem multiplikativen Teil) und einer Translation (dem additiven Teil) zusammen.

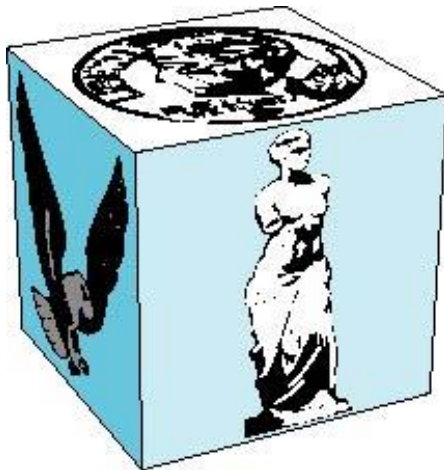
$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

Lineare Abb.

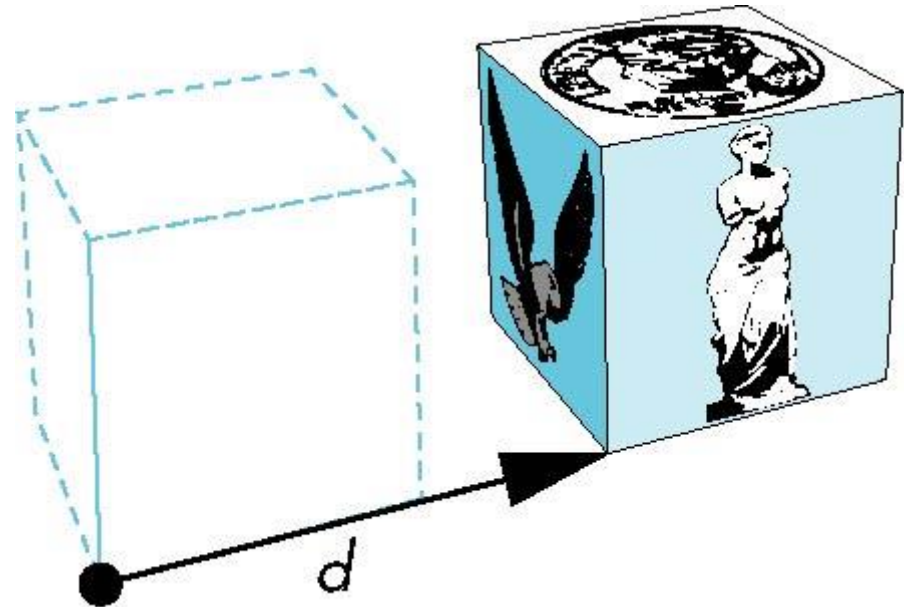
Translation

Die Translation ergibt sich aus der Multiplikation des Vektors (x_0, y_0, z_0) mit der Einheitsmatrix.

Translation



Objekt



Translation: jeder Punkt wird um den gleichen Vektor \mathbf{d} verschoben

Transformation mit Translationsanteil $(x_0 \ y_0 \ z_0)^T$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \\ a_{20} & a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix}$$

→ Repräsentation als 3x3-Matrix mit $a_{ij} = \begin{cases} 1, & i = j \\ 0, & i \neq j \end{cases}$
(Identität)
und Verschiebungsvektor

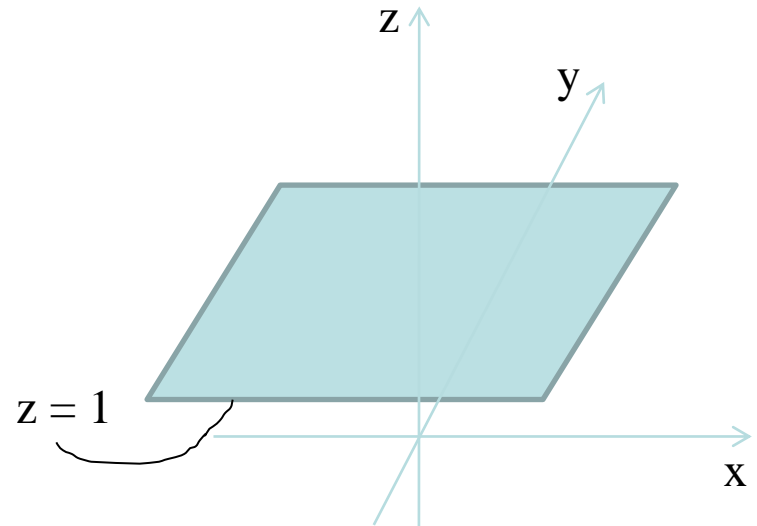
Kompaktere Darstellung möglich?

Homogene Koordinaten

n -dimensionale (inhomogene) Koordinaten
werden zu
 $(n+1)$ -dimensionalen homogenen Koordinaten

z.B. 2D \rightarrow 3D:

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Homogene Koordinaten

definiere Äquivalenzklasse:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \cong \begin{pmatrix} sx \\ sy \\ sz \\ s \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x / w \\ y / w \\ z / w \end{pmatrix}$$

Skalierungsfaktor s bzw. Gewicht w ungleich 0

Translation als Matrix-Multiplikation in homogenen Koordinaten

- Der lineare Teil einer Translation T ist die Identität
- Bsp. Translation um den Vektor $(x_0 \ y_0 \ z_0)^T$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation als Matrix-Multiplikation in homogenen Koordinaten

- Der lineare Teil einer Translation T ist die Identität
- Bsp. Translation um den Vektor $(x_0 \ y_0 \ z_0)^T$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation als Matrix-Multiplikation in homogenen Koordinaten

- Der lineare Teil einer Translation T ist die Identität
- Bsp. Translation um den Vektor $(x_0 \ y_0 \ z_0)^T$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation als Matrix-Multiplikation in homogenen Koordinaten

- Der lineare Teil einer Translation T ist die Identität
- Bsp. Translation um den Vektor $(x_0 \ y_0 \ z_0)^T$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

Translation als Matrix-Multiplikation in homogenen Koordinaten

- Der lineare Teil einer Translation T ist die Identität
- Bsp. Translation um den Vektor $(x_0 \ y_0 \ z_0)^T$

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x + x_0 \\ y + y_0 \\ z + z_0 \\ 1 \end{bmatrix}$$

Homogene Koordinaten sind einfach deutbar als ein erweitertes Rechenschema!

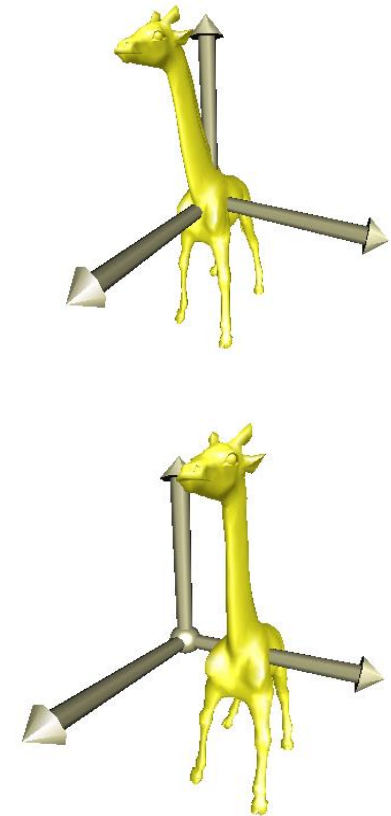
Affine Abbildungen

Matrizenschreibweise

Allgemein lässt sich jede 3D affine Abbildung durch eine 4x4-Matrix ausdrücken

- Linke obere 3x3-Submatrix: Rotation, Skalierung etc.
- Rechte Spalte: Translation

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & x_0 \\ a_{21} & a_{22} & a_{23} & y_0 \\ a_{31} & a_{32} & a_{33} & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



3D affine Abbildungen werden durch homogene (erweiterte)
4x4-Matrizen beschrieben

- Einheitliche Darstellung → einfache Implementierung
- Hintereinanderausführung verschiedener Transformationen → nur Multiplikation der Matrizen
- **Immer im Ursprung zentriert**



$$p' = A_1 \cdot p$$

$$p'' = A_2 \cdot p'$$

⋮

⋮

⋮

$$p^n = A_n \cdot p^{n-1}$$



$$p^n = A_n \cdot \dots \cdot A_2 \cdot A_1 \cdot p$$

- Wiederholung: Grafikpipeline
- Transformationen und affine Abbildungen
- **Skalierung, Scherung, Rotation**
- Projektion

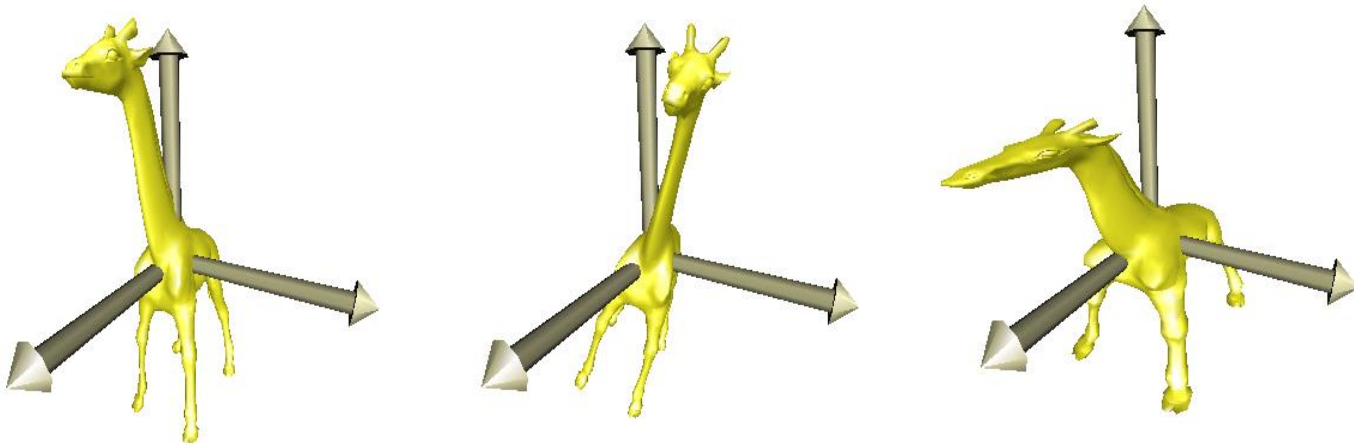
Skalierung, Scherung und Rotation

Die affinen Abbildungen *Skalierung*, *Scherung*, *Rotation* lassen den **Ursprung** invariant

Sie besitzen keinen Translationsanteil

Es sind genau die linearen Transformationen

3x3-Matrizen wären ausreichend



Skalierung, Scherung und Rotation



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Homogene Form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dabei bestimmen die Bilder der Basisvektoren
 $(1\ 0\ 0)^T$, $(0\ 1\ 0)^T$, $(0\ 0\ 1)^T$ eine lineare Abbildung

Skalierung, Scherung und Rotation

Homogene Form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dabei bestimmen die Bilder der Basisvektoren
 $(1\ 0\ 0)^T$, $(0\ 1\ 0)^T$, $(0\ 0\ 1)^T$ eine lineare Abbildung

Skalierung, Scherung und Rotation



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Homogene Form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ a_{21} & a_{22} & a_{23} & 0 \\ a_{31} & a_{32} & a_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Dabei bestimmen die Bilder der Basisvektoren
 $(1 \ 0 \ 0)^T$, $(0 \ 1 \ 0)^T$, $(0 \ 0 \ 1)^T$ eine lineare Abbildung

Skalierung, Scherung und Rotation

Multipliziert man die Ortsvektoren der Punkte von rechts, so stehen die Bilder der *Basisvektoren* der linearen Abbildung A in den Spalten der A beschreibenden Matrix.

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \\ a_{31} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a_{13} \\ a_{23} \\ a_{33} \end{bmatrix}$$

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a_{12} \\ a_{22} \\ a_{32} \end{bmatrix}$$

Skalierung, Scherung und Rotation

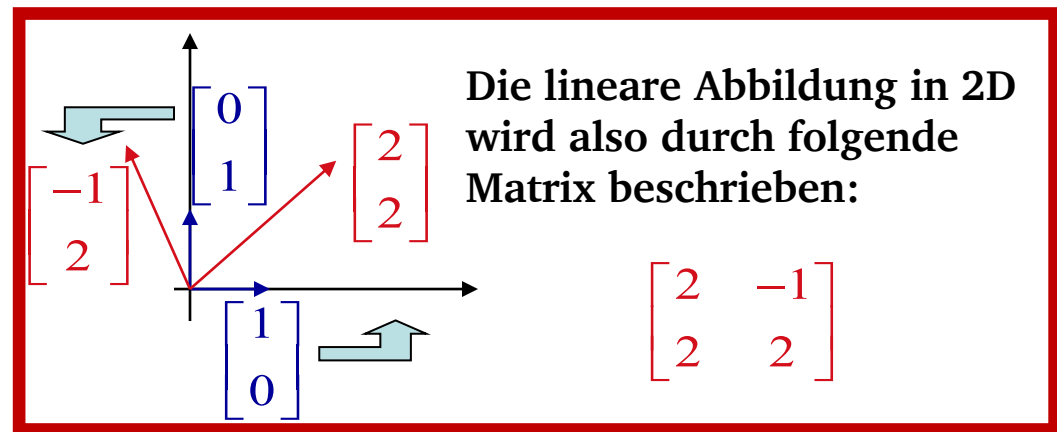
Multipliziert man die Ortsvektoren der Punkte von rechts, so stehen die Bilder der *Basisvektoren* der linearen Abbildung A in den Spalten der A beschreibenden Matrix.

2D Beispiel:

$$A \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$A \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$A = \begin{bmatrix} 2 & -1 \\ 2 & 2 \end{bmatrix}$$

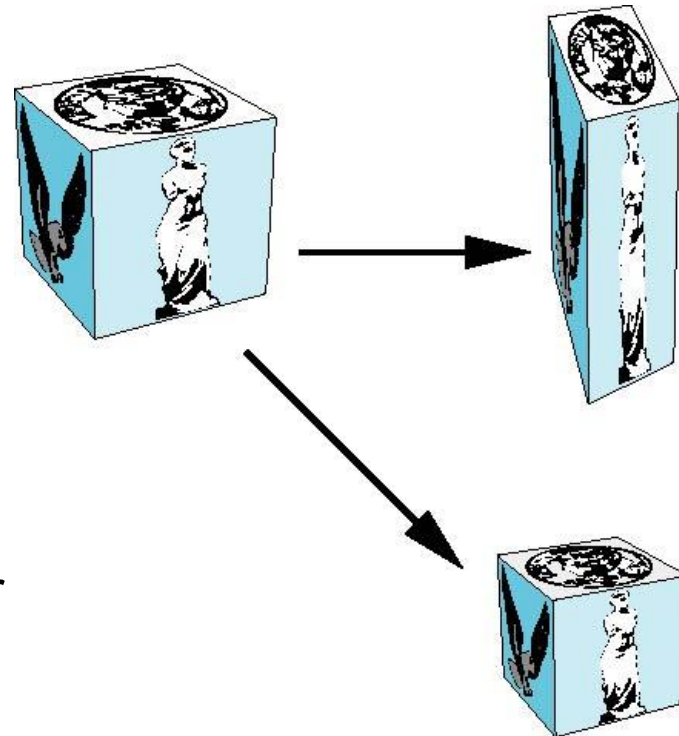


Eine Skalierung S ergibt für die affinen Basisvektoren folgende Beziehung:

- $S((1\ 0\ 0)^T) = (s_1\ 0\ 0)^T$
- $S((0\ 1\ 0)^T) = (0\ s_2\ 0)^T$
- $S((0\ 0\ 1)^T) = (0\ 0\ s_3)^T$.

Skalierung wird durch eine Diagonalmatrix beschrieben!

Skalierung kann unterschiedlich sein für die einzelnen Komponenten



Die zugehörige 3 x 3 Matrix ergibt sich daher zu

$$\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & s_3 \end{pmatrix}$$

In homogenen Koordinaten

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

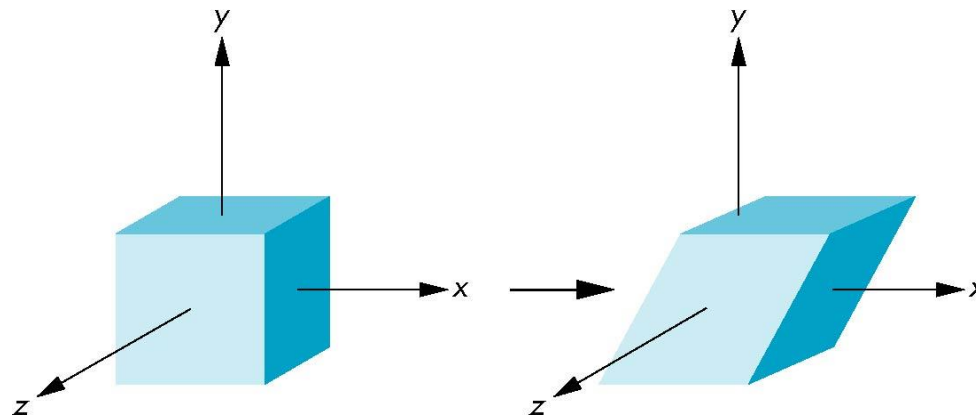
Der Sonderfall $s_1 = s_2 = s_3 = s$ bedeutet die gleiche Skalierung für alle Koordinaten

Die zugehörige homogene Matrix hat dann die Form

$$\begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \\ 0 & 0 & s & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{s} \end{bmatrix}$$

Eine Scherung SH ergibt für die affinen Basisvektoren folgende Beziehung

- $SH((1\ 0\ 0)^T) = (1\ s_1\ s_3)^T$
- $SH((0\ 1\ 0)^T) = (s_2\ 1\ s_4)^T$
- $SH((0\ 0\ 1)^T) = (s_5\ s_6\ 1)^T$



Die dazugehörige 3 x 3 Matrix wird daher zu

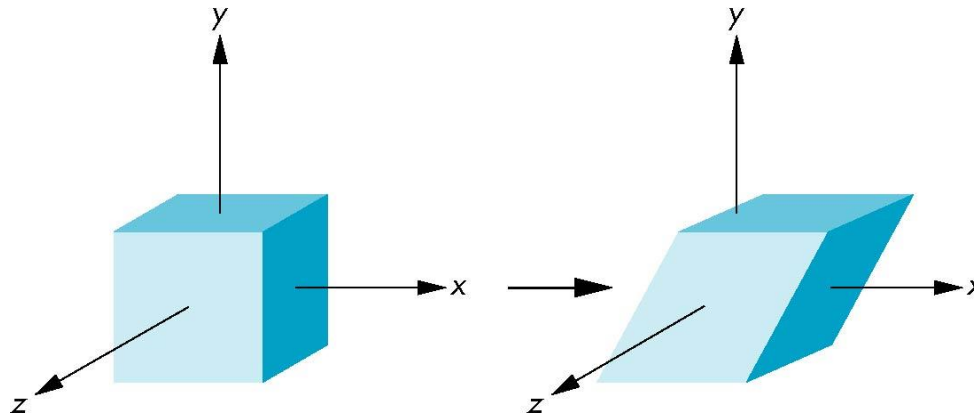
$$\begin{pmatrix} 1 & s_2 & s_5 \\ s_1 & 1 & s_6 \\ s_3 & s_4 & 1 \end{pmatrix}$$

In homogenen Koordinaten folgt

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & s_2 & s_5 & 0 \\ s_1 & 1 & s_6 & 0 \\ s_3 & s_4 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Beispiel: Diese Scherung SH ergibt für die affinen Basisvektoren folgende Beziehung

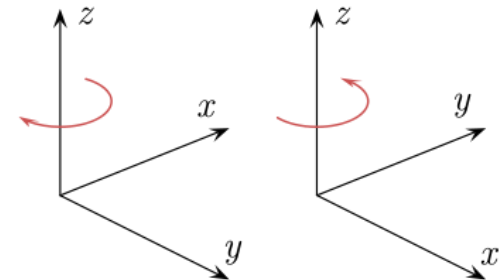
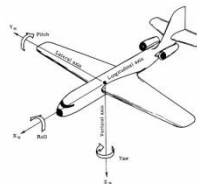
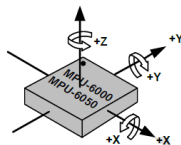
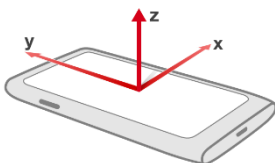
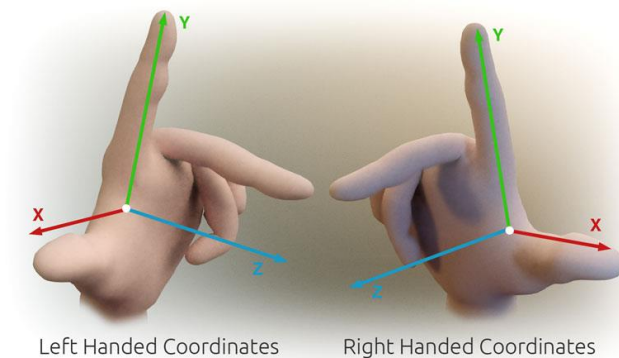
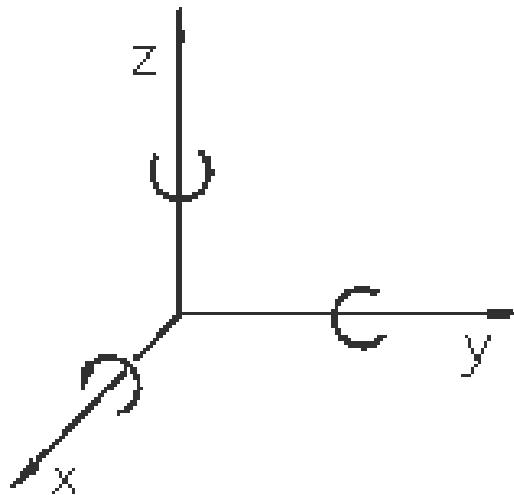
- x-Achse: $SH((1\ 0\ 0)^T) = (1\ 0\ 0)^T$
- y-Achse: $SH((0\ 1\ 0)^T) = (1\ 1\ 0)^T$
- z-Achse: $SH((0\ 0\ 1)^T) = (0\ 0\ 1)^T$



Rotation

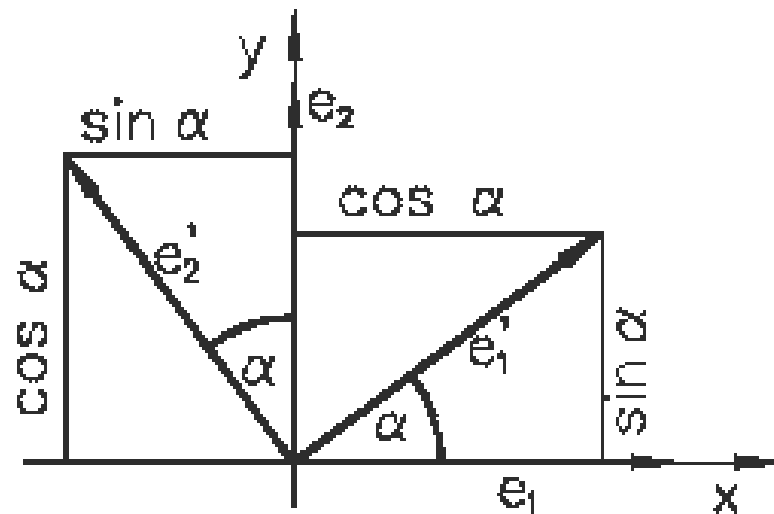
Die Drehwinkel sind in dem **Rechtssystem** x, y, z immer positiv:

(https://en.wikipedia.org/wiki/Cartesian_coordinate_system)



Eine Rotation R_α um den Winkel α um die z -Achse in mathematisch positive Richtung ergibt für die Basisvektoren folgende Beziehung:

- $R_\alpha((1\ 0\ 0)^T) = (\cos \alpha, \sin \alpha, 0)$
- $R_\alpha((0\ 1\ 0)^T) = (-\sin \alpha, \cos \alpha, 0)$
- $R_\alpha((0\ 0\ 1)^T) = (0, 0, 1)$



Die zugehörige 3 x 3 Matrix ergibt sich daher zu

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

In homogenen Koordinaten folgt für die Rotation R_α um die z -Achse

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Bei Rotation R_α um die x -bzw. y -Achse ergeben sich analog folgende homogene Darstellungen.

- Drehung mit dem Winkel α um die x -Achse

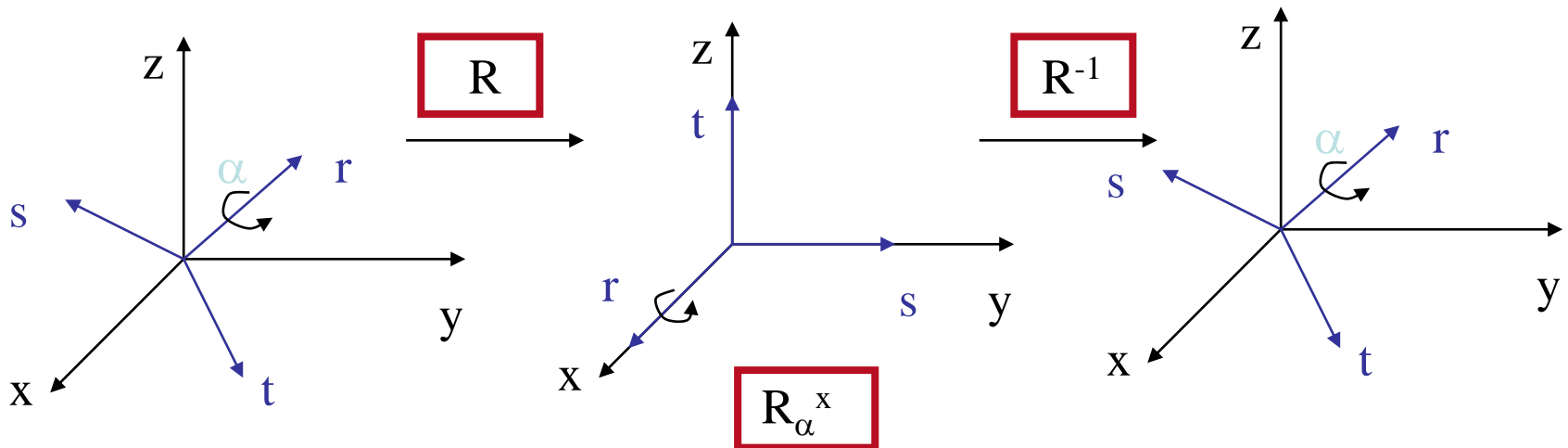
$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos \alpha & -\sin \alpha & 0 \\ 0 & \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Drehung mit dem Winkel α um die y -Achse

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin \alpha & 0 & \cos \alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Rotation um beliebige Achse

Drehung $R_\alpha(x,y,z)$ um beliebige Achse in Richtung des normierten Vektors $\mathbf{r} = (x \ y \ z)^T$ um den Winkel α



$$R_{(x,y,z)} = R^{-1} R_\alpha^x R$$

Rotation um beliebige Achse

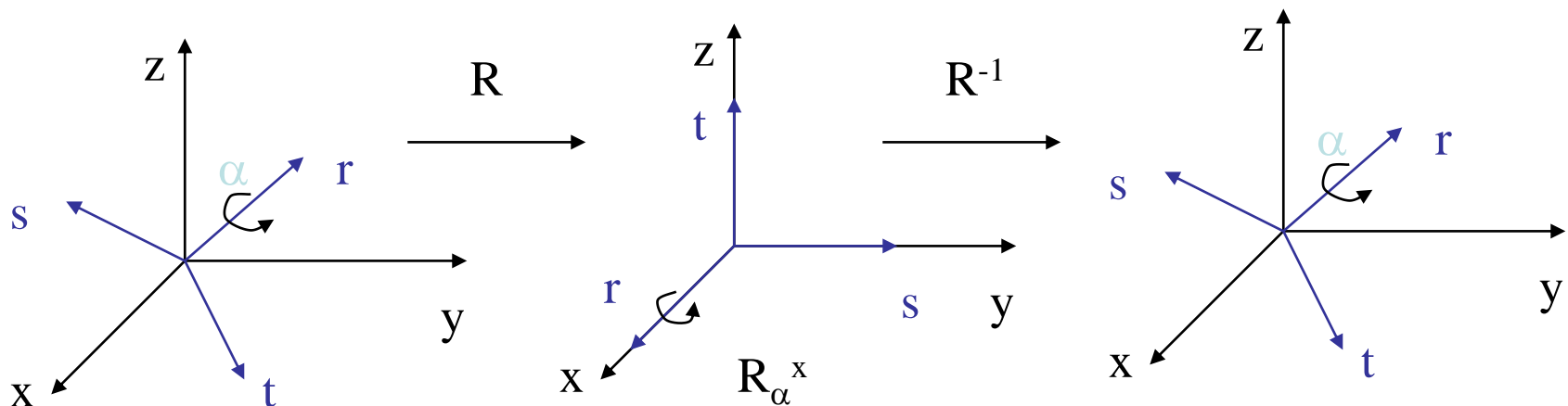
Berechnung von R

Orthonormale Basis $(\mathbf{r}, \mathbf{s}, \mathbf{t})$ bestimmen

- erster Basisvektor ist \mathbf{r}
- zweiter Basisvektor \mathbf{s} soll senkrecht auf \mathbf{r} stehen; [Kreuzprodukt](#):

$$\mathbf{s} = \frac{\mathbf{r} \times \mathbf{e}_x}{\|\mathbf{r} \times \mathbf{e}_x\|} \quad \text{oder (falls } \mathbf{r} \parallel \mathbf{e}_x) \quad \mathbf{s} = \frac{\mathbf{r} \times \mathbf{e}_y}{\|\mathbf{r} \times \mathbf{e}_y\|}$$

- dritter Basisvektor $\mathbf{t} = \mathbf{r} \times \mathbf{s}$



Rotation um beliebige Achse

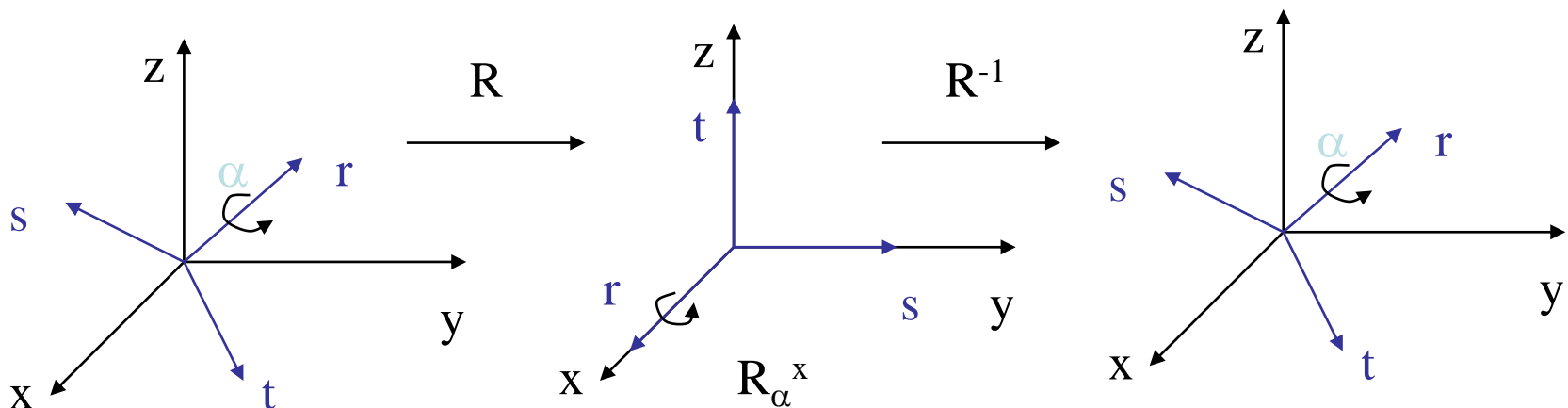
Berechnung von R

Vektoren ($\mathbf{r}, \mathbf{s}, \mathbf{t}$) werden in die *Spalten* der Transformationsmatrix geschrieben (Sehe Folie 32!)

Die Transformationsmatrix Matrix ist orthonormal (d.h.: orthogonal und normiert)

- Sie transformiert $\mathbf{e}_x \rightarrow \mathbf{r}$, $\mathbf{e}_y \rightarrow \mathbf{s}$, $\mathbf{e}_z \rightarrow \mathbf{t}$. (das ist R^{-1})
- Für orthonormierte Matrizen R gilt stets $R^{-1} = R^t$.

Also: R ergibt sich, indem man die Vektoren ($\mathbf{r}, \mathbf{s}, \mathbf{t}$) in die *Zeilen* schreibt

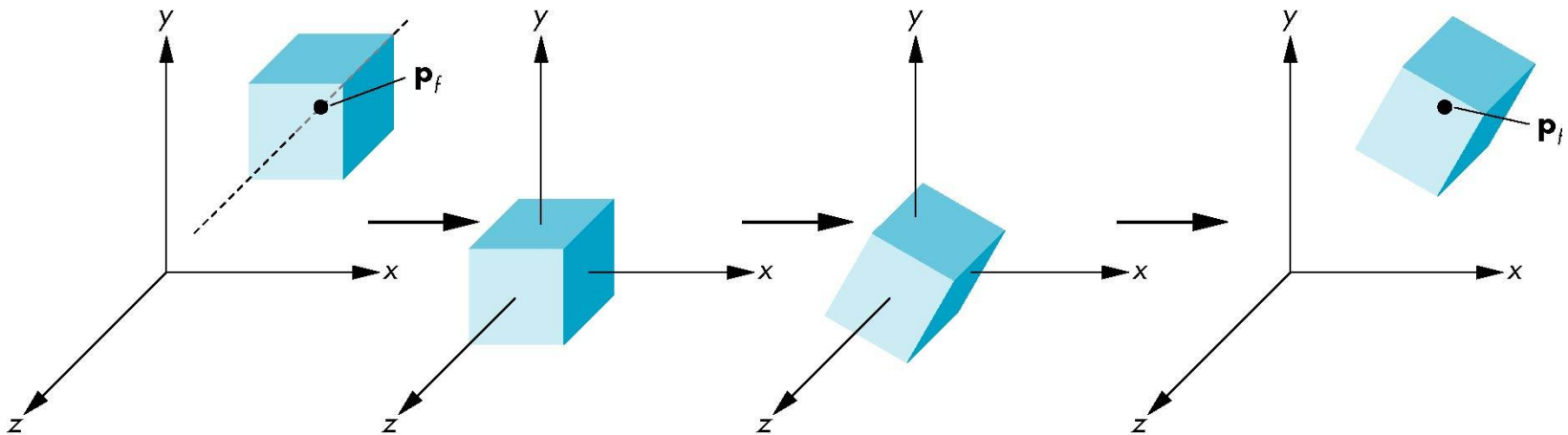


Rotation um beliebige Raumachse

Die diskutierten Rotationen lassen den **Ursprung** fest!

Rotationsachse durch eine beliebige Achse im Raum:

1. Verschiebung des Rotationszentrums in den Ursprung
2. anschließende Rotation und
3. Zurückverschiebung in das Rotationszentrum



Rotation um beliebige Raumachse

Beispiel

- Rotation in positiver Richtung um eine Achse durch den Punkt (x_0, y_0, z_0) und um den Winkel α
- Die Richtung der Rotationsachse sei die z-Richtung.

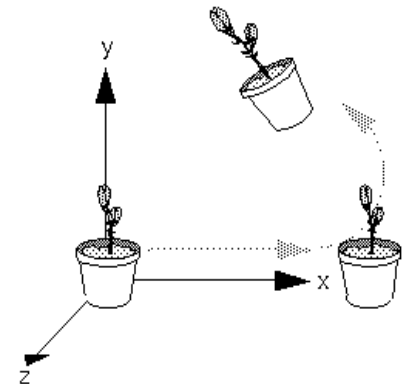
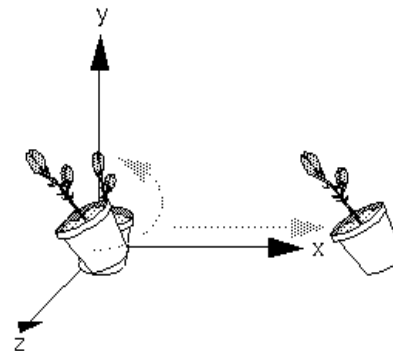
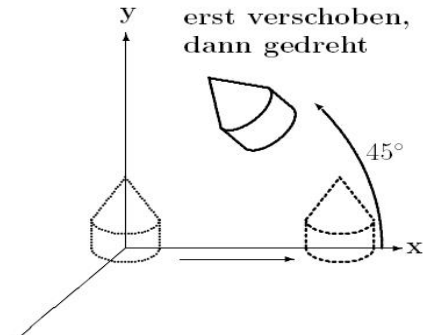
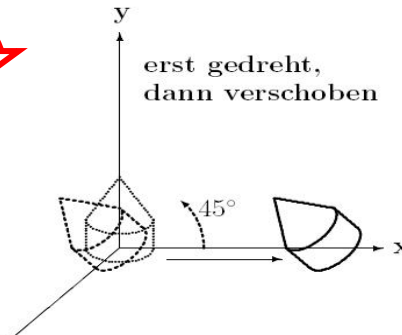
$$p' = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos \alpha & -\sin \alpha & 0 & 0 \\ \sin \alpha & \cos \alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -x_0 \\ 0 & 1 & 0 & -y_0 \\ 0 & 0 & 1 & -z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot p$$

Transformationen - Nicht-Kommutativität

Reihenfolge der Transformationen darf i. A. nicht vertauscht werden

Grund: Matrizenmultiplikation ist nicht kommutativ, $AB \neq BA$ gilt nicht für alle Matrizen A und B

Beispiel: TR versus RT



Transformationen

Bemerkung zum Aufwand

Für viele Eckpunkte (typisch in GDV) ist es günstiger einmalig die gesamte Transformationskette zu berechnen

Berechne einmalig das Produkt $A_n \dots A_2 A_1$ und wende die Eckpunkte darauf an

$$\begin{aligned} p' &= A_1 \cdot p \\ p'' &= A_2 \cdot p' \end{aligned}$$

.

.

.

$$p^n = A_n \cdot p^{n-1}$$


$$p^n = A_n \cdot \dots \cdot A_2 \cdot A_1 \cdot p$$

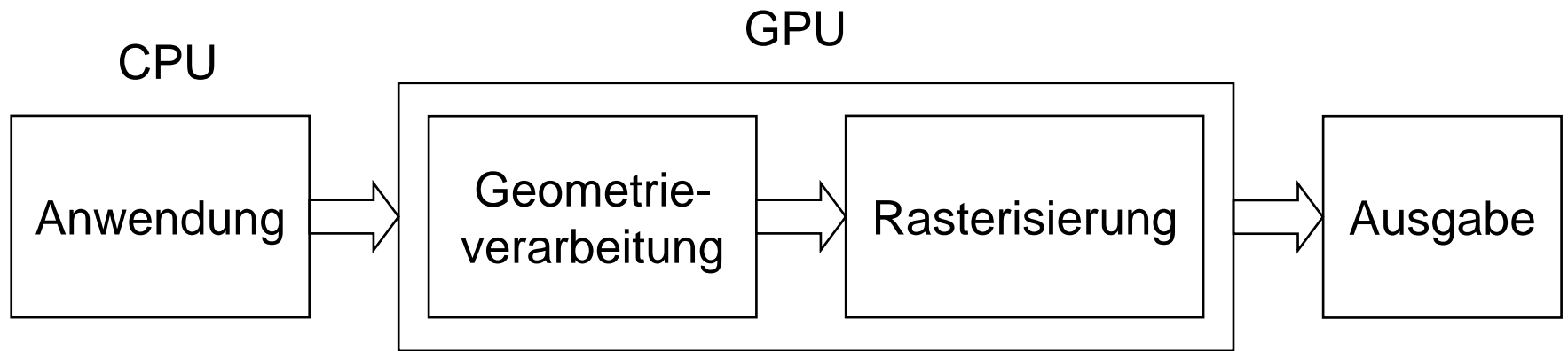

$$p^n = A_n \cdot p^{n-1}$$

No!



- Wiederholung: Grafikpipeline
- Transformationen und affine Abbildungen
- Homogene Koordinaten
- Skalierung, Scherung, Rotation
- **Projektion**
- 3D-Interaktion

3D Grafik-Pipeline



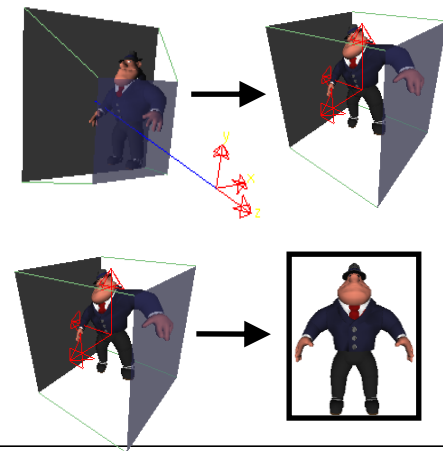
Simulation der Beleuchtung

Perspektivische Transformation ←

Clipping

Culling

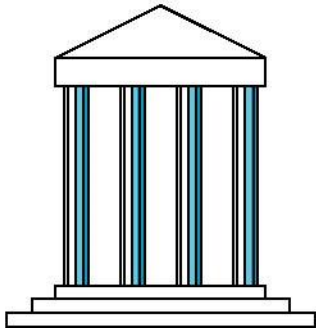
Projektion ←



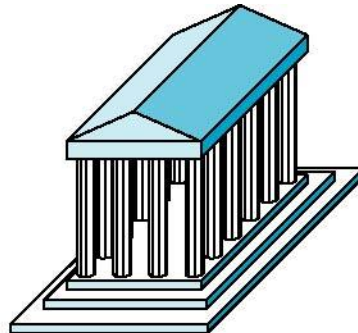
Klassische Projektionen



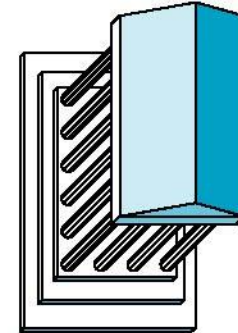
TECHNISCHE
UNIVERSITÄT
DARMSTADT



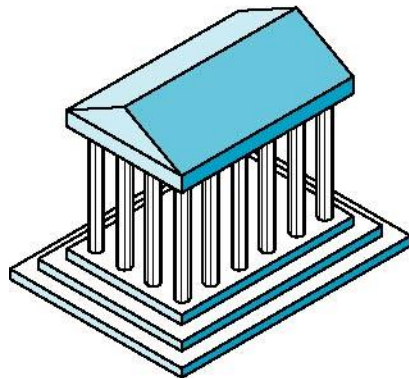
Aufriss (Frontansicht)



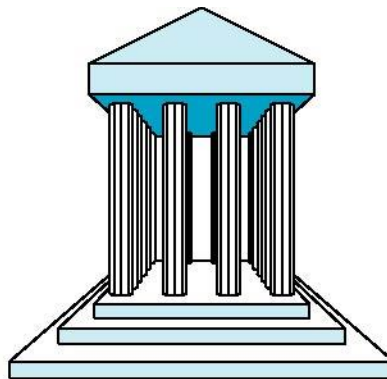
Kabinett-/
Kavalliersperspektive



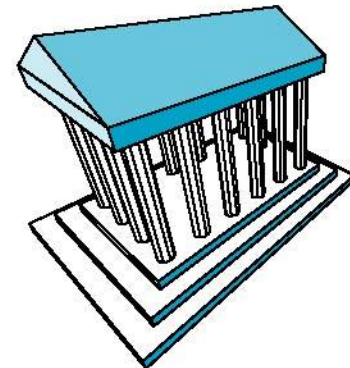
Allgemeine
Parallelprojektion



Isometrische
Perspektive



Zentralperspektive



Vogelperspektive

können durch homogene 4×4 -Matrizen beschrieben werden

1. Geraden werden auf Geraden abgebildet
2. Schnitte von Geraden bleiben erhalten
3. Flächen werden auf Flächen abgebildet
4. Reihenfolge von Punkten auf projektiven Geraden bleiben erhalten

Winkel werden verändert

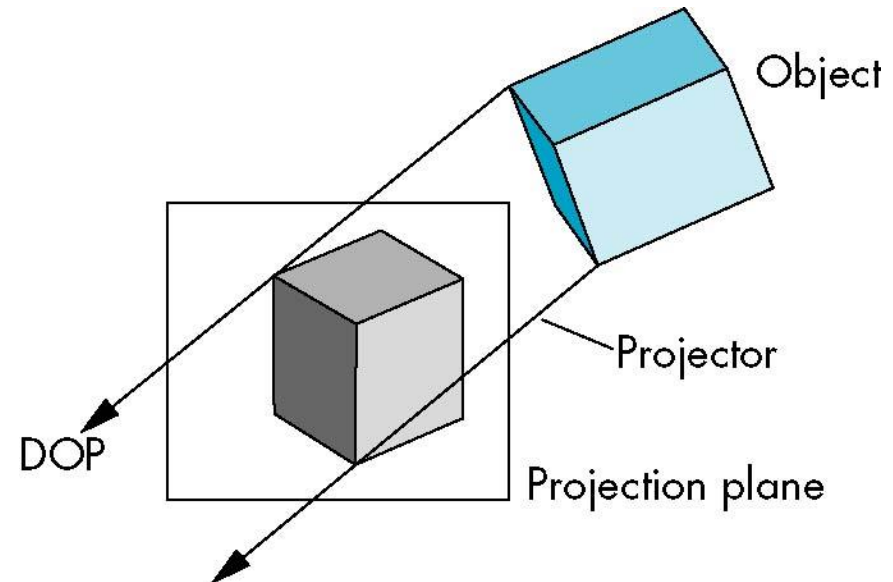
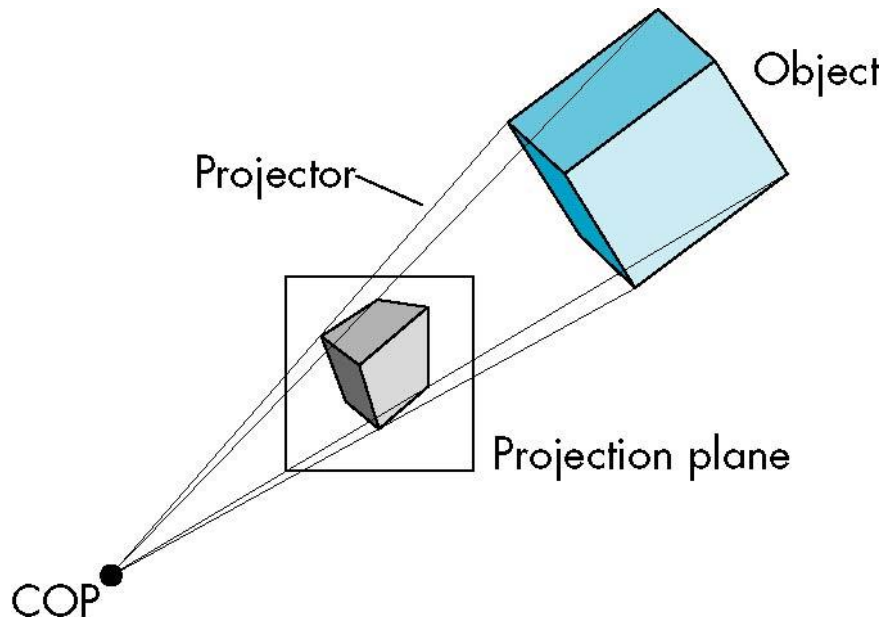
Parallelität geht oft verloren → Parallelen schneiden sich in Fluchtpunkten

Rechtecke werden auf Vierecke transformiert

Perspektivische und parallele Projektionen

Bei perspektivischer Projektion treffen sich die Strahlen im Augpunkt (COP, Center of Projection, Projektionszentrum)

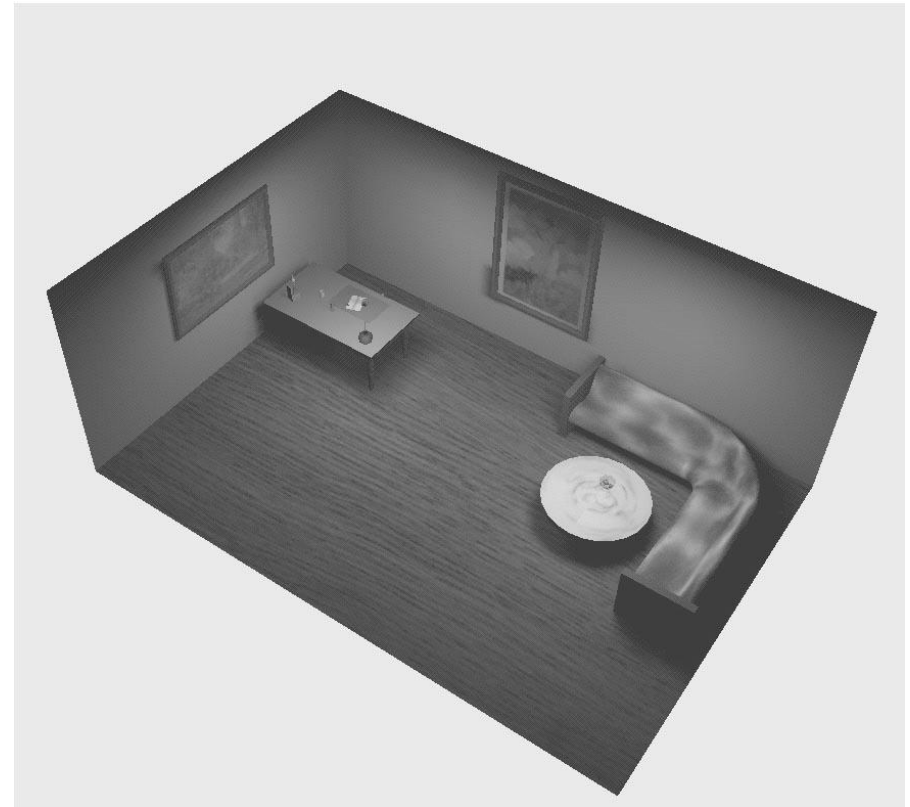
Bei parallelen Projektionen sind die Projektionsstrahlen parallel (DOP, Direction of Projection).



Vergleich: perspektivische und...

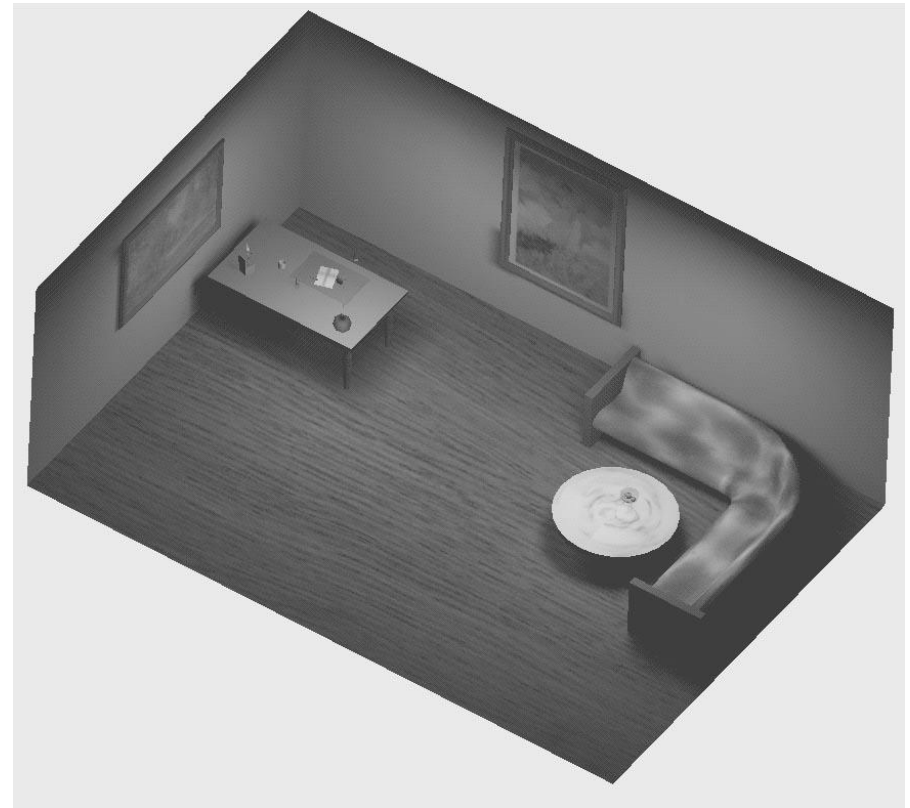
perspektivische Projektion
(auch: Zentral-Projektion)

- vergleichbar dem fotografischen System, entspricht natürlicher Wahrnehmung des Menschen
- Abstand zwischen Objekten und Projektionsebene geht ein
- Längenverhältnisse **ändern** sich
- Winkel **ändern** sich
- parallele Geraden bleiben **nicht** parallel

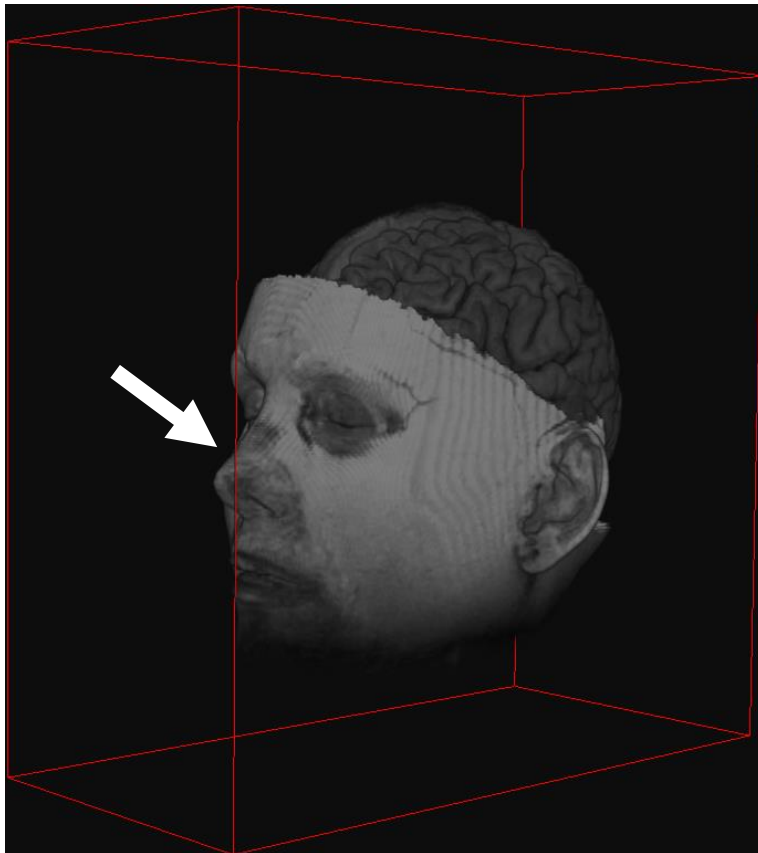


...parallele Projektion

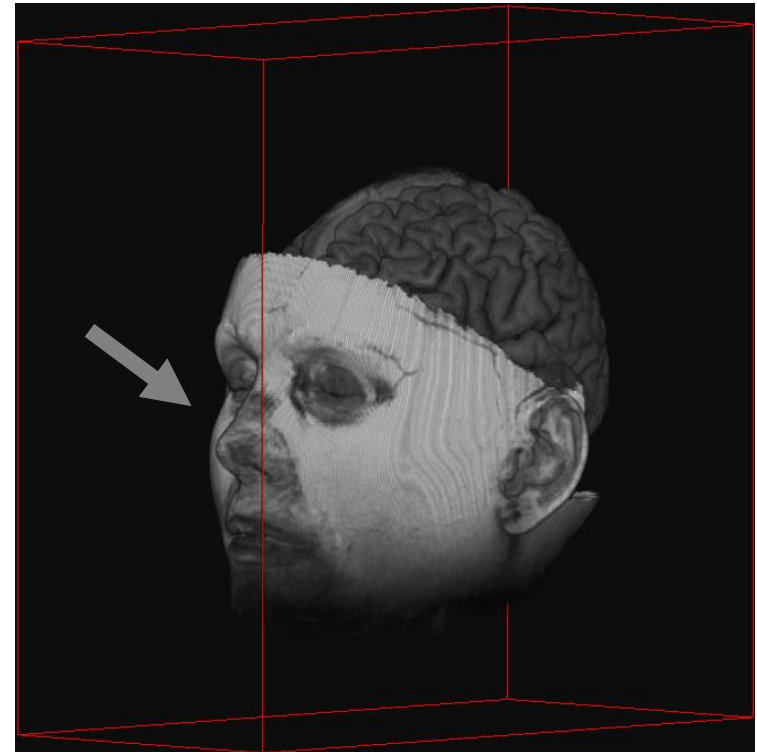
- parallele Projektion
(auch: orthografische Projektion)
- „weniger Realismus“ in der Darstellung
 - Winkel ändern sich i.A. nicht
 - parallele Geraden bleiben parallel



Medizin: parallele Projektion oft bevorzugt



Perspektivische Verzerrung



Realismus = unverzerrte Bilder

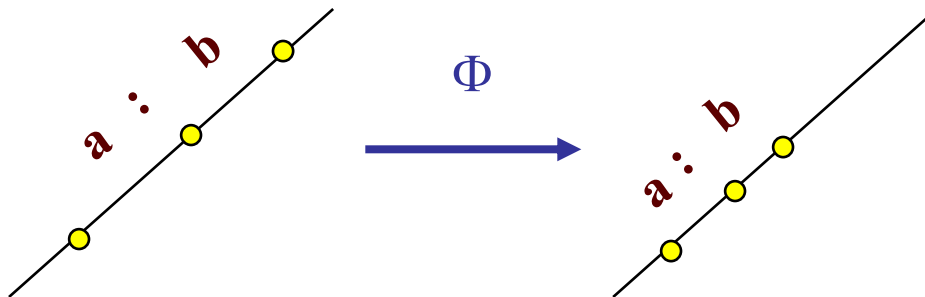
→ Längen und Abstände sind relevant!



Erinnerung

Affine Abbildungen – Eigenschaften

1. Bilden Geraden auf Geraden ab.
2. Beschränkte Objekte bleiben beschränkt.
3. Verhältnisse von Längen, Flächen, Volumen bleiben erhalten.
4. Parallele Objekte (Geraden, Ebenen, ...) bleiben parallel

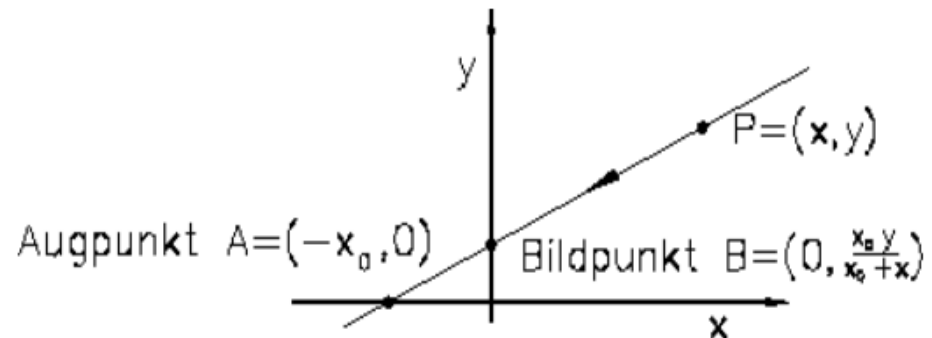


Perspektivische Abbildungen

Perspektivische Abbildungen sind **keine** affinen Abbildungen

- Längenverhältnisse sind nicht invariant
- Parallele Objekte bleiben nicht parallel

Vom Blickpunkt (Augpunkt, Beobachtungspunkt) weit entfernte Objekte werden kleiner dargestellt als nahe am Blickpunkt befindliche Objekte.



Sind der zu projizierende Punkt $P=(x, y)$ und der Augpunkt $A=(-x_0, 0)$ gegeben, so gilt nach dem Strahlensatz für den Bildpunkt $B=(0, y_0)$

$$\frac{y_0}{y} = \frac{x_0}{x + x_0}$$

Perspektivische Abbildungen

Allgemein lautet mit

$$y_0 = y \cdot \frac{x_0}{x_0 + x}$$

die Abbildung wie folgt:

$$\begin{pmatrix} x \\ y \end{pmatrix} \mapsto \begin{pmatrix} 0 \\ \frac{y \cdot x_0}{x_0 + x} \end{pmatrix}$$

Daraus ergibt sich die homogene 3×3 -Matrix (2D-Geometrie!)

$$\begin{bmatrix} 0 \\ \frac{x_0}{x + x_0} \cdot y \\ 1 \end{bmatrix} \cong \begin{bmatrix} 0 \\ x_0 \cdot y \\ x + x_0 \end{bmatrix}$$

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & x_0 & 0 \\ 1 & 0 & x_0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \cong \frac{1}{x_0} \cdot \begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{x_0} & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



Die *perspektivische Projektion* wird in zwei Abbildungen zerlegt:

1. die *perspektivische Transformation* und
2. eine anschließende *Parallelprojektion*

Nach der perspektivischen *Transformation* ist das Sichtvolumen ein Würfel!

Durch Rotation (dh. der Augpunkt $A=(-x_0, 0)$) erreicht man, dass der Würfel achsenparallel wird.

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{x_0} & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{x_0} & 0 & 1 \end{bmatrix}$$

**Perspektivische
Projektion**

**Parallel-
projektion**

**Perspektivische
Transformation**

Kanonisches Sichtvolumen

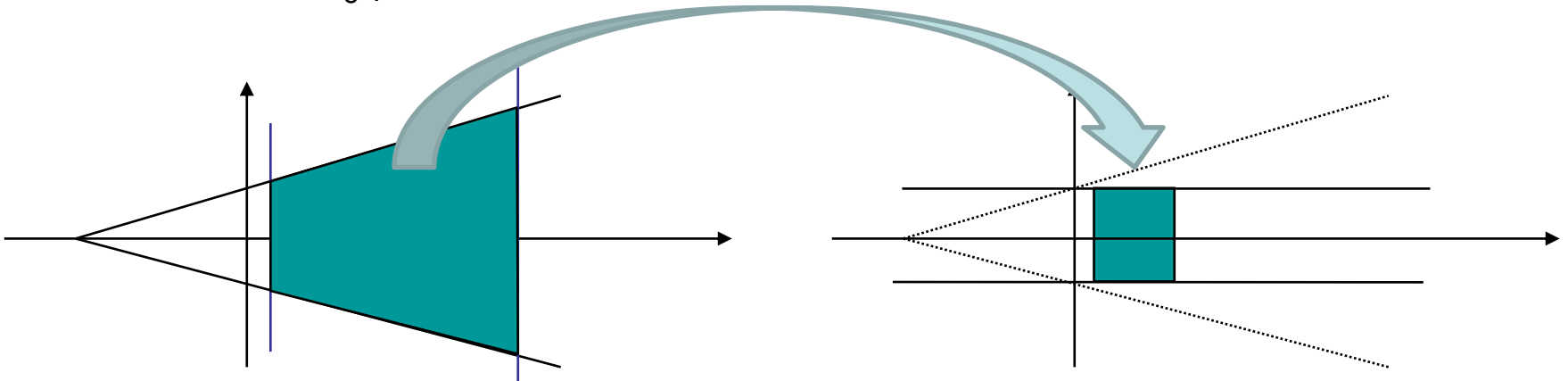
Parallelprojektion (Kamera im Unendlichen, DOP):

Sichtvolumen = Einheitswürfel

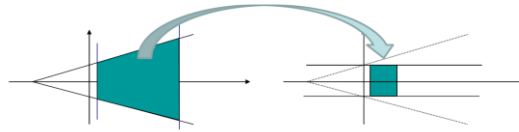
Perspektive: Sichtvolumen = Pyramide (COP)

Nach der perspektivischen *Transformation* ist das Sichtvolumen ein Würfel!

(NB: Gewicht $x+x_0$!)



2D Beispiel



Sichtvolume: Eckpunkte in $(1, 3/2)^T$, $(2, 2)^T$, $(1, -3/2)^T$, $(2, -2)^T$ [in Pink-Rot]. $X_0=2$

Dann liefert die *perspektivische Transformation* die Eckpunkte

$(1, 3/2, 3/2)^T$, $(2, 2, 2)^T$, $(1, -3/2, 3/2)^T$, $(2, -2, 2)^T$.

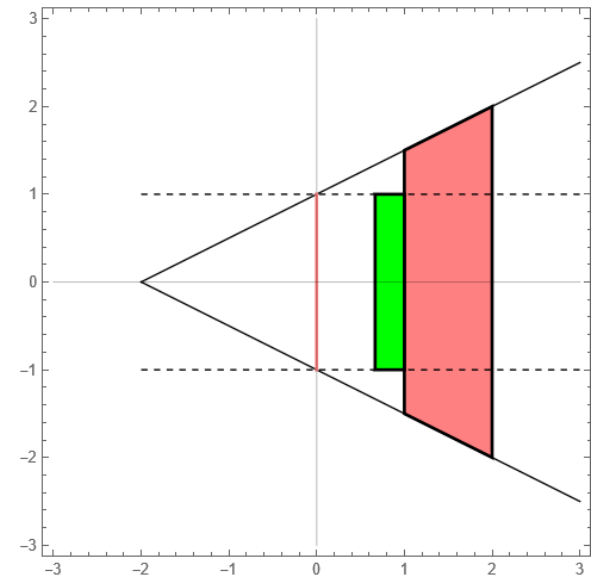
$w \rightarrow 1$ gibt die Eckpunkte

$(2/3, 1, 1)^T$, $(1, 1, 1)^T$, $(2/3, -1, 1)^T$, $(1, -1, 1)^T$

[in Grün].

Parallelprojektion gibt dann

$(0, 1, 1)^T$, $(0, 1, 1)^T$, $(0, -1, 1)^T$, $(0, -1, 1)^T$.



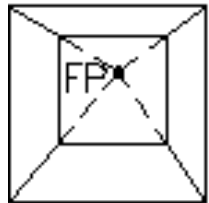
Allgemeine perspektivische Transformation zu einem
Fluchtpunkt in (x_0, y_0, z_0) :

$$T_p \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{x_0} & \frac{1}{y_0} & \frac{1}{z_0} & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = \begin{bmatrix} x \\ y \\ z \\ \frac{x}{x_0} + \frac{y}{y_0} + \frac{z}{z_0} + w \end{bmatrix}$$

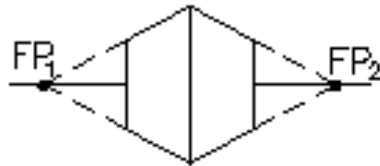
Die Richtungen von Parallelen zu den Koordinatenachsen werden auf die
Achsen-Fluchtpunkte $[x_0, 0, 0, 0]^T$, $[0, y_0, 0, 0]^T$, $[0, 0, z_0, 0]^T$ abgebildet.

Ein-, Zwei- und Dreipunktperspektive

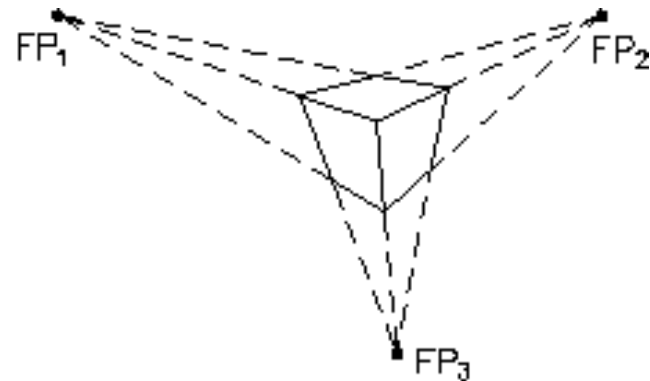
Manchmal braucht man mehrere Fluchtpunkte, entscheide dann welche Sichtfläche von welcher Punkt gesehen wird.



1-Punkt



2-Punkt

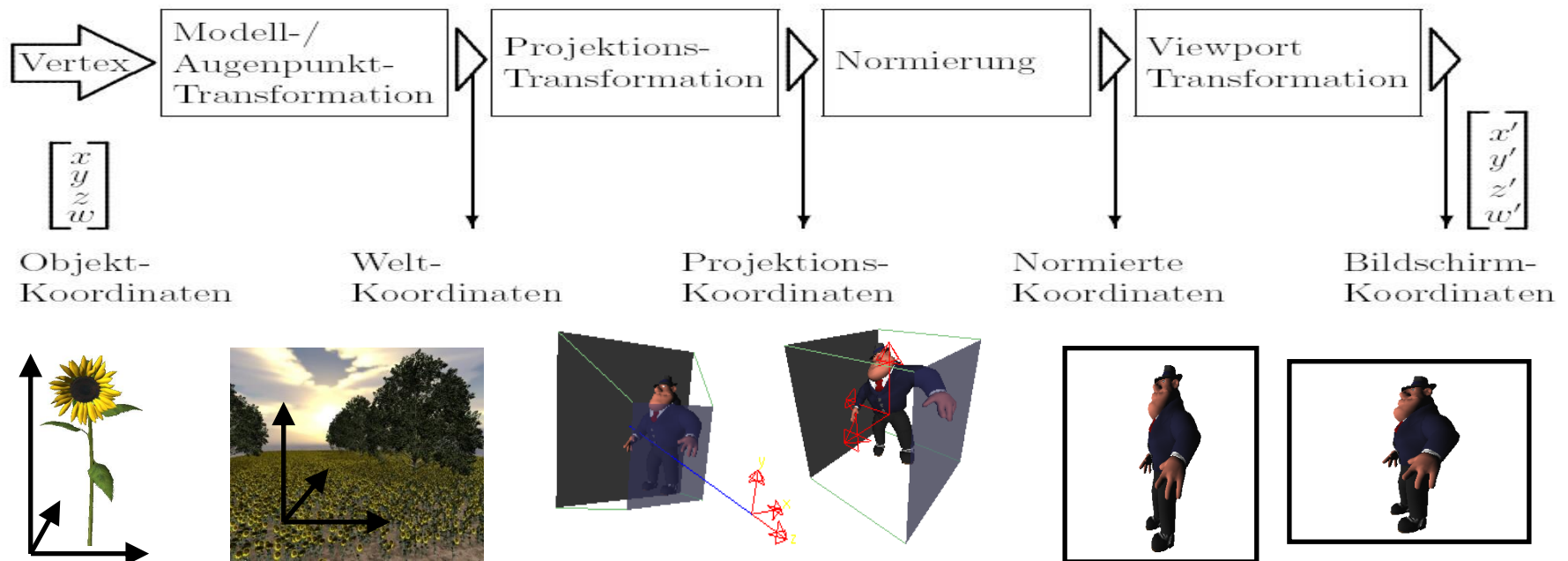


3-Punkt



Transformationen in der Grafik-Pipeline

1. *Modelling* Transformations: ordne 3D-Objekte (Modelle) im Raum an und positioniere diese
2. *Viewing* Transformations: wähle Betrachterstandpunkt und positioniere diesen (default: Position: Ursprung & Blickrichtung: negative z-Achse)
3. *Projection* Transformations: projiziere Viewing Volume in 2D
4. *Viewport* Transformations: wandle in Bildschirmkoordinaten um



Zum Spielen

Zwei Sachen zu Transformationen:

http://apike.ca/prog_svg_transform_demo.html

Hier kann man die Transformationsmatrix direkt einstellen wie man möchte.

Ein kleines Windows Programm offline:

http://www.songho.ca/opengl/gl_transform.html

(etwas scrollen oder Direktlink

<http://www.songho.ca/opengl/files/matrixModelView.zip>)

Zeigt zwar eigentlich die OpenGL Transformationen, aber man sieht auch die entsprechende Matrix.



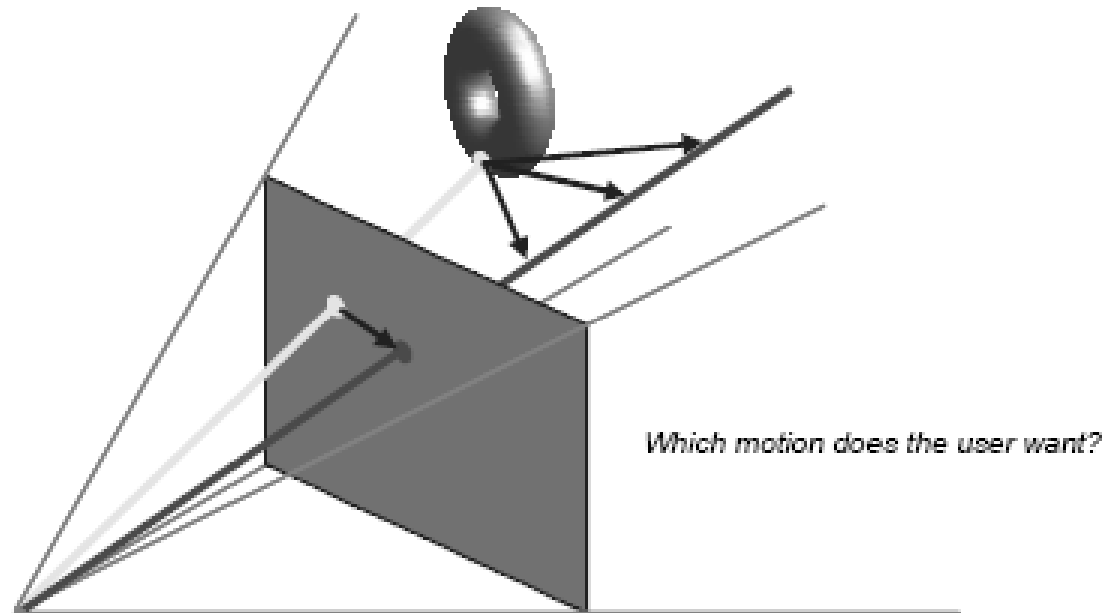
- Wiederholung: Grafikpipeline
- Transformationen und affine Abbildungen
- Homogene Koordinaten
- Skalierung, Scherung, Rotation
- Projektion
- **3D-Interaktion**

3D-Interaktion mit 2D-Eingabegeräten

Problem:

Welche Art der Bewegung möchte der Benutzer ausgeführt haben?

➤ Mehrdeutigkeit



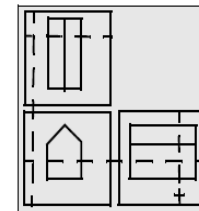
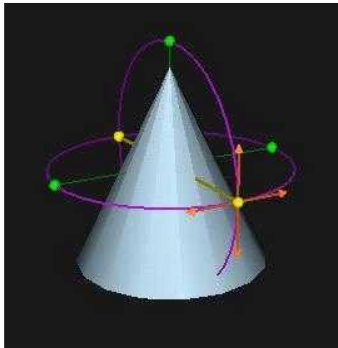
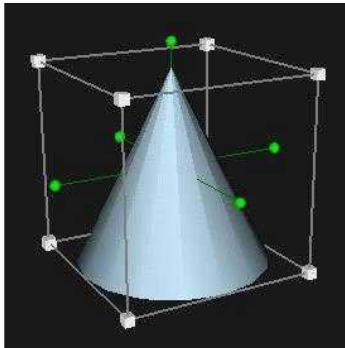
3D-Interaktion mit 2D-Eingabegeräten

Ansätze:

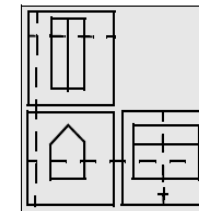
Multi-Window (Mehrfachauswahl)

Direktes 2D-Maus-Mapping

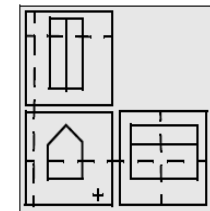
Manipulatoren



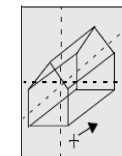
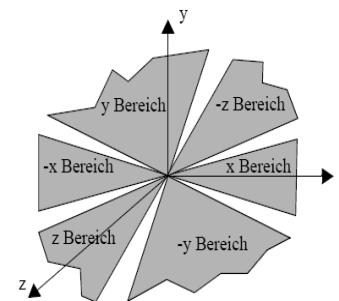
Um den gestrichelten 3D Cursor zu bewegen, wird er mit dem 2D-Mauszeiger gepickt



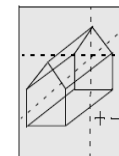
Bewegungen des Mauszeigers führen nun zur Translation des 3D Cursors in der Ebene.



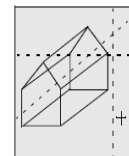
Nach Loslassen des Mausknopfs kann der 2D Cursor wieder frei bewegt werden, ohne daß der 3D Cursor folgt.



Ausgangssituation:
3D Cursor gestrichelt, 2D Cursor normal



Bewegung des 2D Cursors im z-Bereich, der 3D-Cursor bewegt sich in Richtung z



Bewegung des 2D Cursors horizontal in x-Richtung, der 3D-Cursor bewegt sich auch in Richtung x

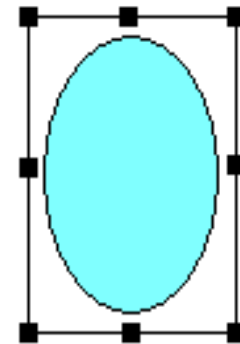
Abbildung 3: 3D-Cursorsteuerung mit 2D-Maus nach Nielson und Olson [107]

Häufig im Zweidimensionalen verwendet

- Kästen in Grafikprogrammen, mit denen skaliert, rotiert und verschoben werden kann
- Drag-and-Drop-Operationen

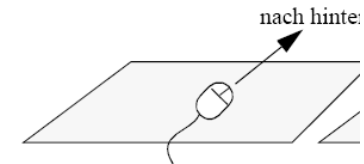
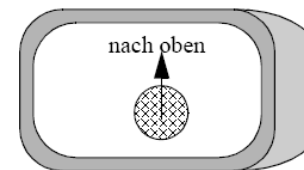
Immer häufiger auch im Dreidimensionalen

- Manipulatoren für Transformationen
- Navigation der Kamera



In 2D einfach zu verwenden (und zu implementieren)

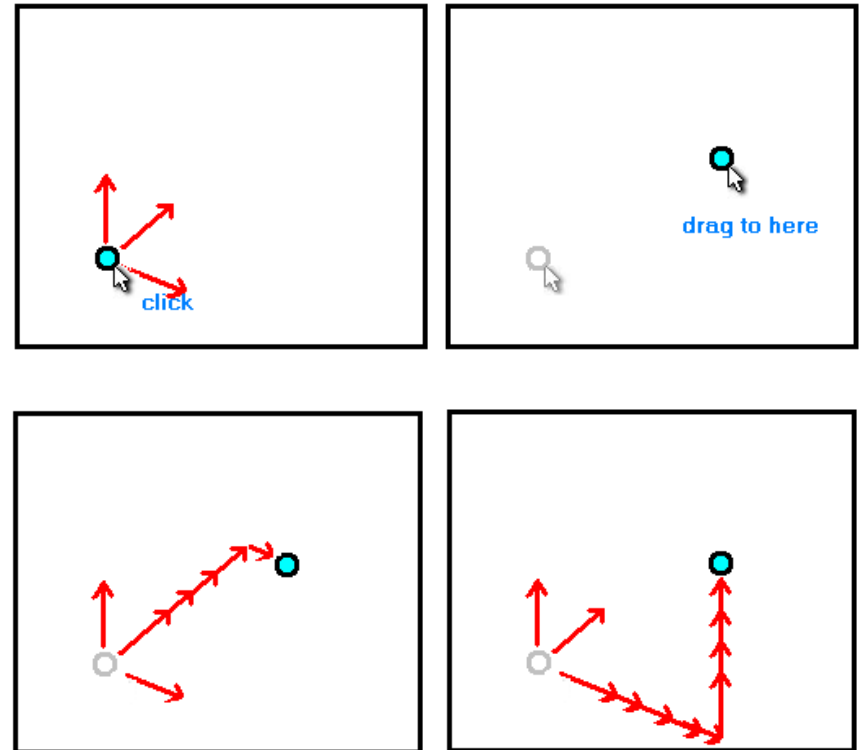
- 1:1 Abbildung zwischen der Mauszeigerposition und dem „Knauf“ im 2D-Raum (räumliche Domäne)
- Schnitttests sind leicht zu implementieren
- Interpretation der Bewegung ist einfach
- Keine Probleme mit der perspektivischen Abbildung



Was verändert sich in 3D?

Warum wird es schwieriger?

- Eine virtuelle 3D-Szene wird 2D angezeigt
- Mehrdeutigkeiten: Unendlich viele Möglichkeiten, die Cursorposition auf eine gerade Linie im 3D-Raum abzubilden
- Noch schwieriger wird es, wenn der 2D-Cursor bewegt wird



3D-Interaktion mit 3D-Widgets

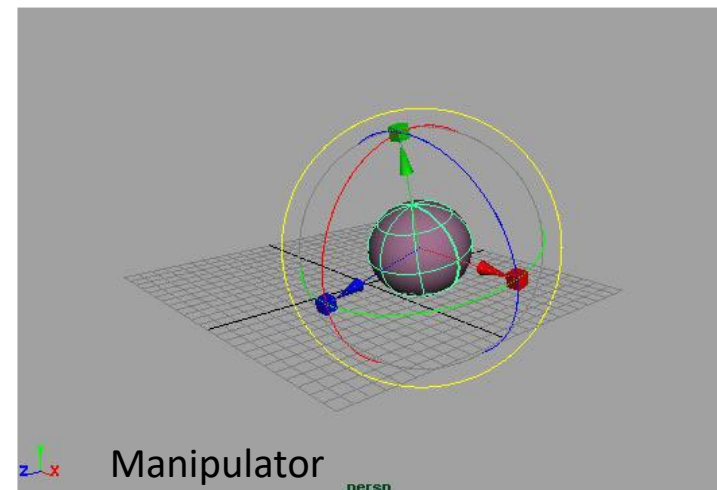
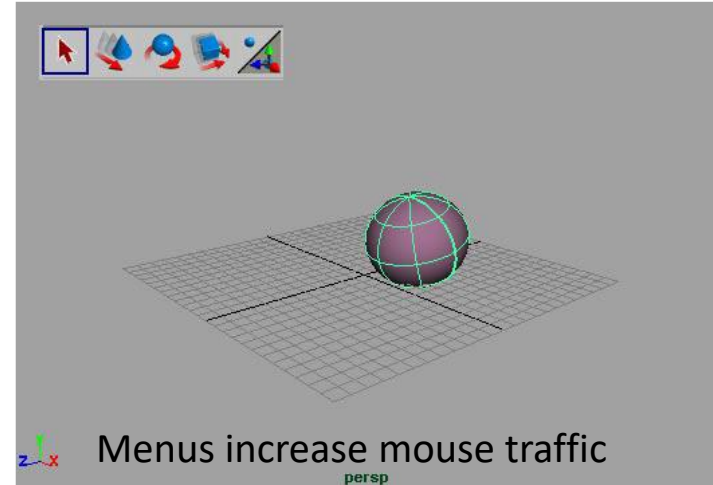
Was ist ein Manipulator?

Eine visuelle grafische Repräsentation einer Operation oder der Status eines Objekts, der zusammen mit dem Objekt selbst angezeigt wird.

Der Status bzw. Die Operation kann durch Klicken und Bewegen (Dragging) der grafischen Elements (Handle) des Manipulators kontrolliert werden

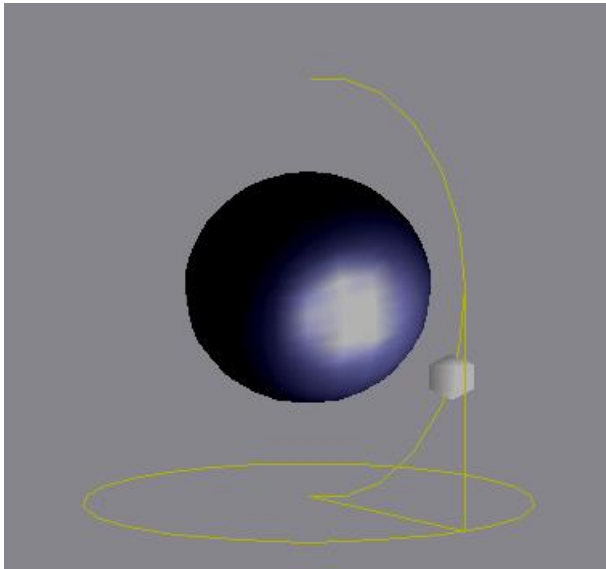
→ Der Zeiger bleibt innerhalb der Szene

→ Reduziert Mausbewegungen

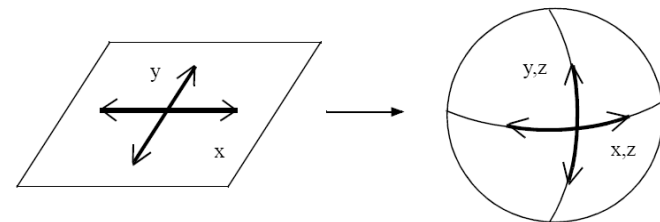
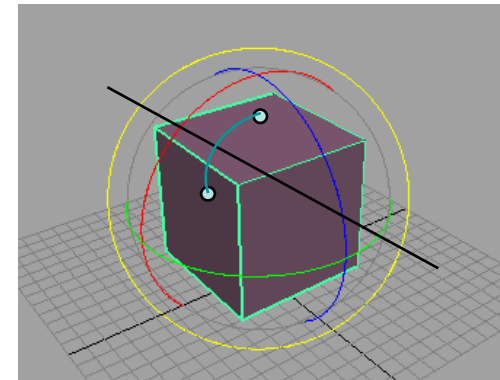


3D-Widgets-Beispiel

Rotation um Achsen



Freie Rotation



Vielen Dank für die Aufmerksamkeit