Interaktive Computergrafik

Prof. Dr. Frank Steinicke
Human-Computer Interaction
Fachbereich Informatik
Universität Hamburg

Interaktive Computergrafik

Kapitel Transformationen

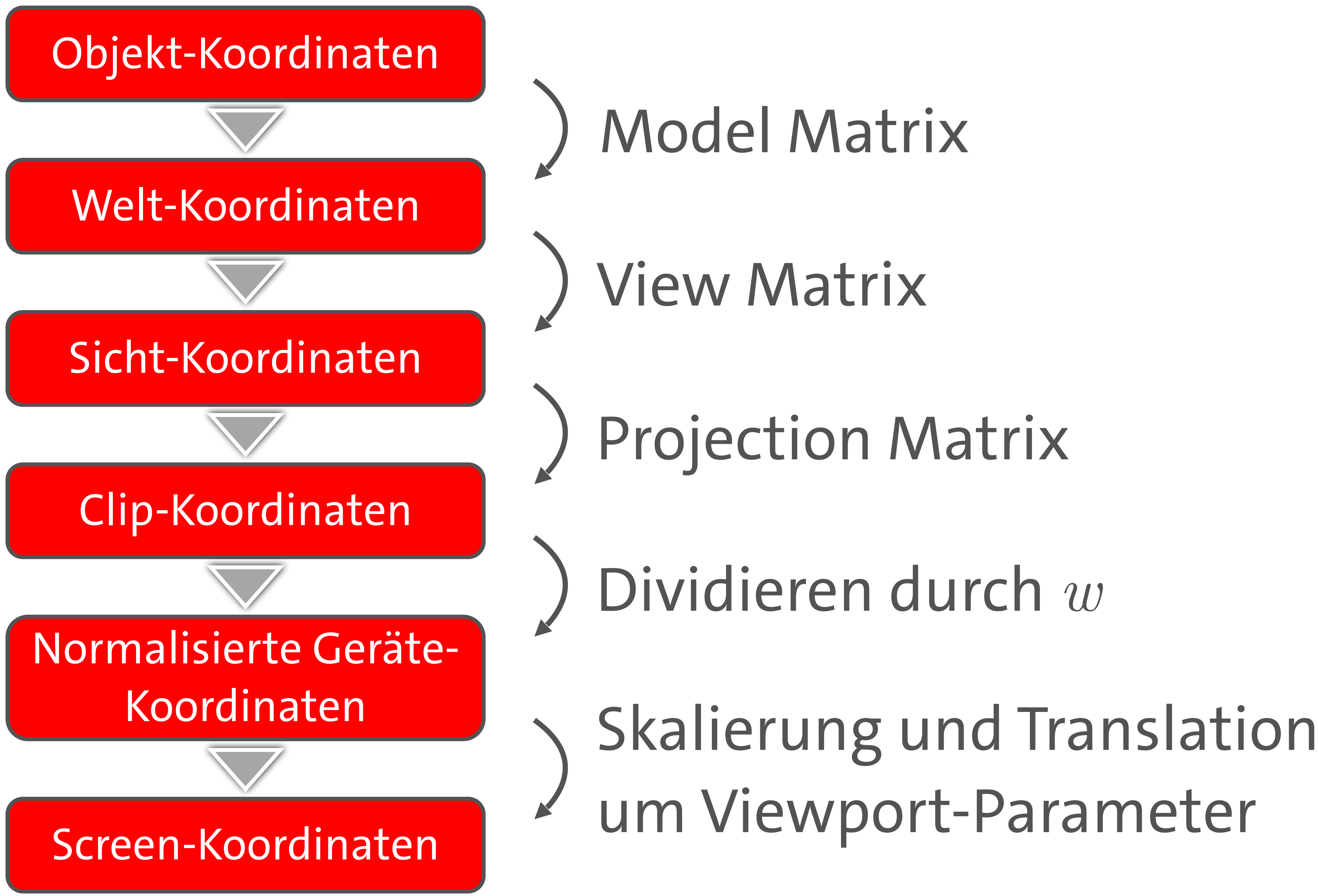
Prof. Dr. Frank Steinicke
Human-Computer Interaction, Universität Hamburg

Interaktive Computergrafik

Kapitel Transformationen

Rendering Pipeline

Rendering Pipeline
Koordinatentransformationen



Rendering Pipeline

Koordinatentransformationen

- alle Primitive werden am Ende durch Menge von Pixel beschrieben
- Achtung: direkte Eingabe der Bildschirm-koordinaten hat Nachteile, u.a.
  - Transformationen schwer umzusetzen
  - Animationen schwer zu spezifizieren
  - kaum hierarchische Gruppierungen
  - ...

Rendering Pipeline

Koordinatentransformationen

Objekt-Koordinaten

↓ Model Matrix

Welt-Koordinaten

↓ View Matrix

Sicht-Koordinaten

↓ Projection Matrix

Clip-Koordinaten

↓ Dividieren durch w

Normalisierte Geräte-Koordinaten

↓ Skalierung und Translation um Viewport-Parameter

Screen-Koordinaten

Rendering Pipeline
Objektkoordinaten

- Modellierung der Objekte erfolgt in **Objektkoordinaten** (**lokale Koordinaten**)
- jedes Objekt hat „eigenes“ Koordinatensystem
- jedes Objekt wird in lokalen Koordinaten definiert

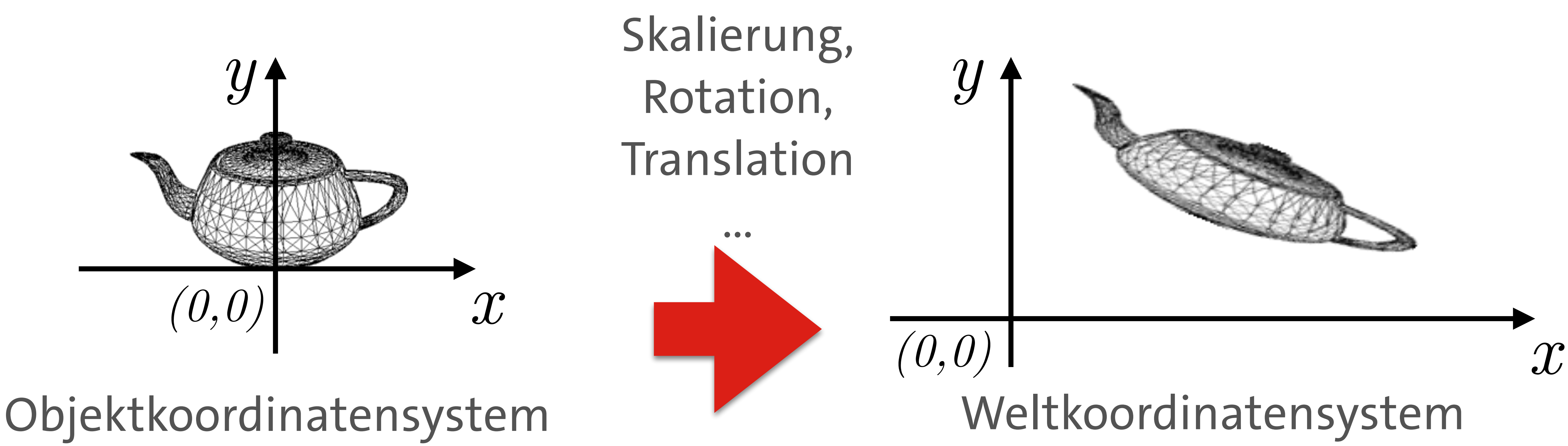
Rendering Pipeline
Weltkoordinaten

- **Weltkoordinaten** sind Bezugskoordinatensystem der Szene
- **Weltkoordinatensystem** ist einheitliches Koordinatensystem, in dem alle Objekte der Szene (sowie Kamera und Lichter in 3D definiert sind)

➡ Bezüge zwischen Objekten können hergestellt werden

2D-Rendering-Pipeline
1. Schritt: Objekt → Welt

- Primitive (bestehend aus Punkten, Linien, Flächen, ...) werden in Objektkoordinaten beschrieben und ins Weltkoordinaten-system transformiert



Objektkoordinaten
Motivation

- effektiv zur Modellierung
- einfache Änderung von Position, Größe und Ausrichtung einzelner Objekte
- Animationen einfacher zu spezifizieren
- hierarchische Gruppierungsmöglichkeiten
- ...

Hierarchische Modelle

- **Hierarchische Modellierung** erlaubt Mehrfachverwendung von Geometrie sowie einfache Transformationen
- Darstellung eines Szenengraphen als **gerichteten azyklischen Graph**

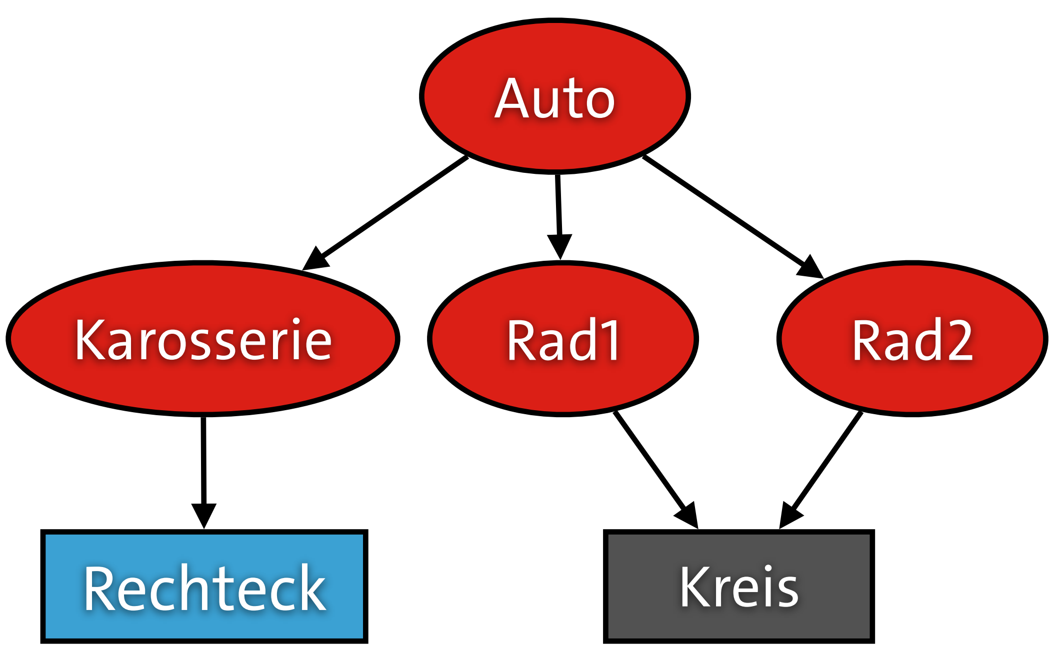
Transformation (z.B. Skalierung, Rotation, Translation)

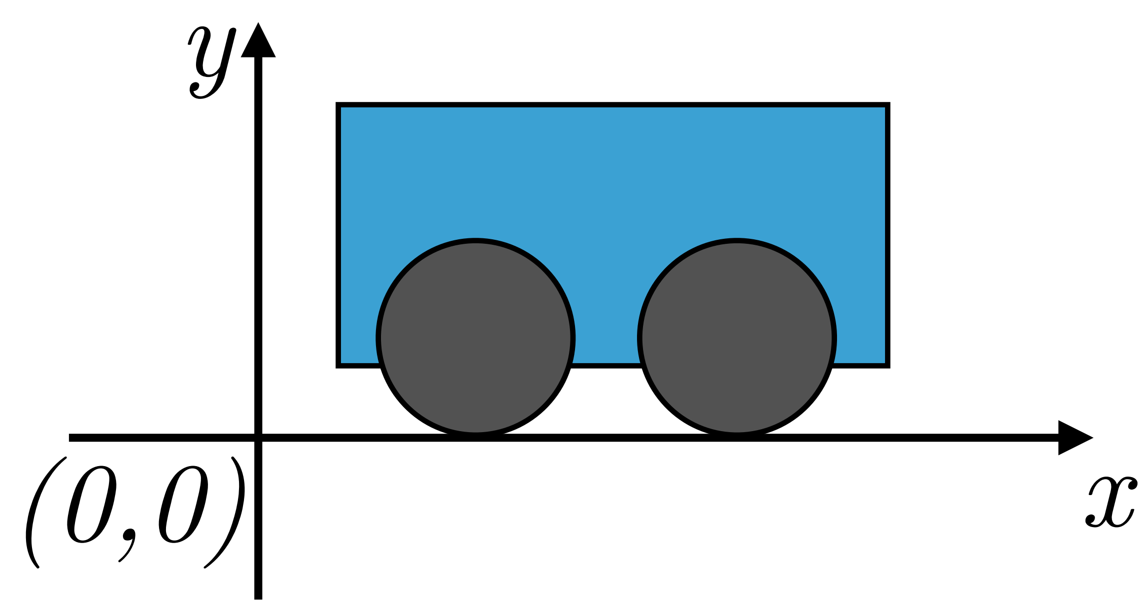
Geometrie (z.B. Primitive), Attribute (z.B. Farbe)

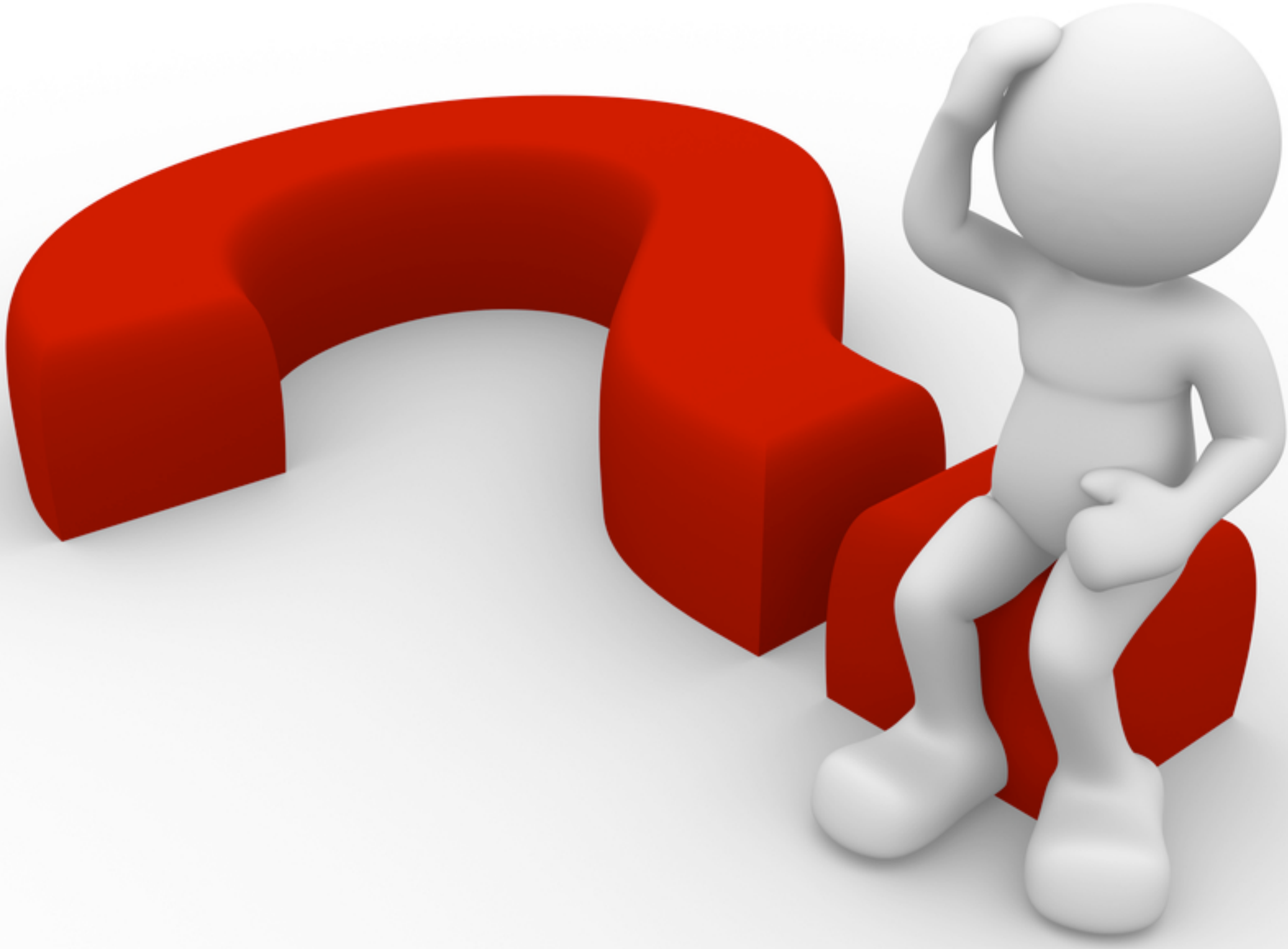
Hierarchische Modelle

Beispiel: Szenengraph

- "Auto" besteht aus Objekt "Karosserie" und zwei Objekten "Rad1" und "Rad2"
- Geometrie "Kreis" wird in Objektkoordinaten beschrieben und zweimal verwendet







Interaktive Computergrafik

Kapitel Transformationen

2D-Transformationen

Transformationen

- **Eckpunkte** (engl. **Vertices**) werden durch Vektoren beschrieben
- alle geometrischen Objekte, welche über Vertices beschrieben werden, können transformiert werden
- *Achtung: Wir betrachten zunächst nur* **affine Transformationen**, *z.B. Translationen, Rotationen, Skalierungen*

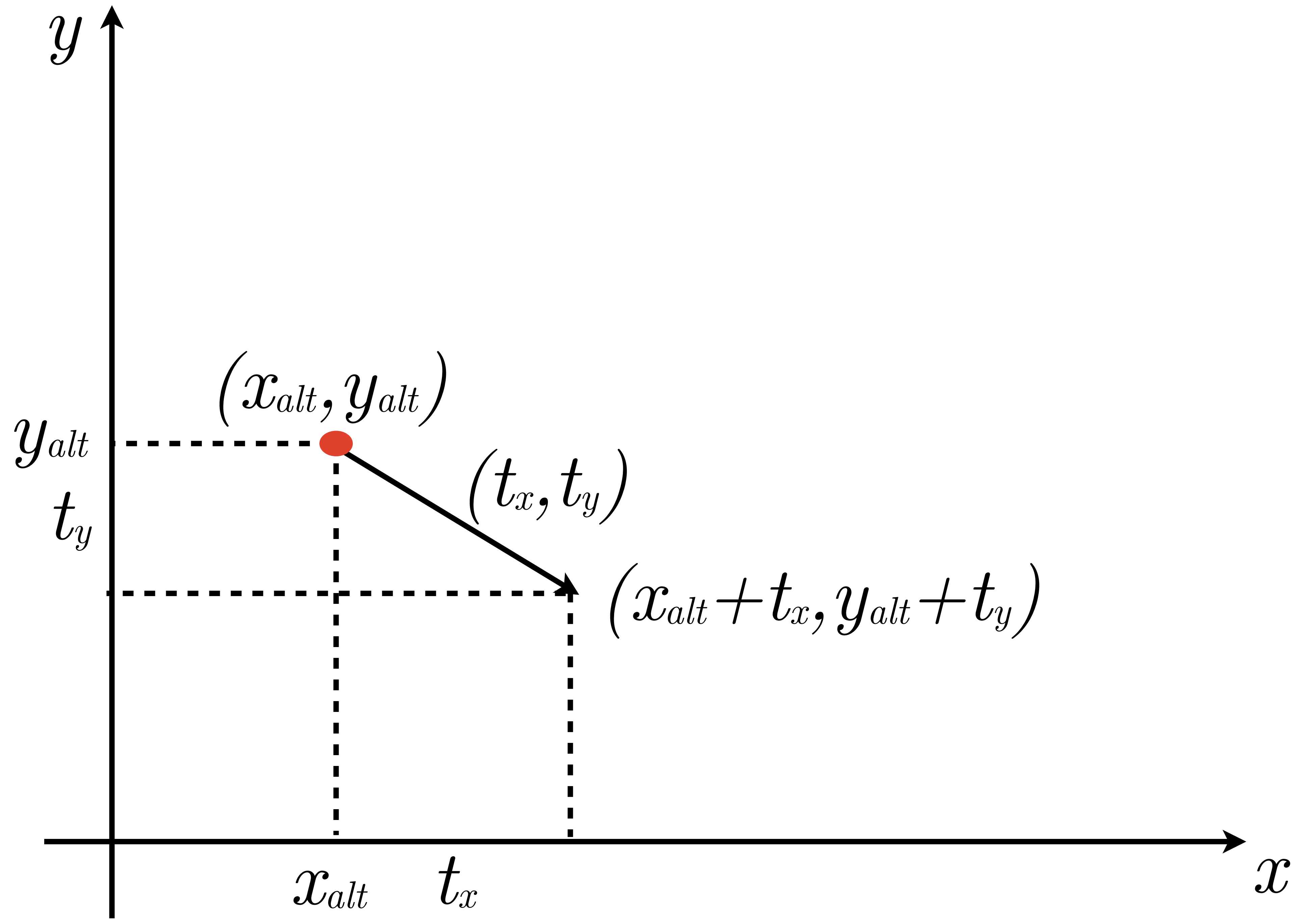
Transformation
2D-Translation

- **Translationen** verschieben jeden Eckpunkt um gleichen Vektor (t_x, t_y)

$$\begin{pmatrix} x_{\text{neu}} \\ y_{\text{neu}} \end{pmatrix} = \begin{pmatrix} x_{\text{alt}} \\ y_{\text{alt}} \end{pmatrix} + \begin{pmatrix} t_x \\ t_y \end{pmatrix} = \begin{pmatrix} x_{\text{alt}} + t_x \\ y_{\text{alt}} + t_y \end{pmatrix}$$

Transformation

2D-Translation



Transformation

2D-Skalierung

- **Skalierung (Größenänderung)** eines Vektors (x_{alt}, y_{alt}) mit **Skalaren** s_x und s_y
- **Skalierung** heißt **uniform**, wenn $s_x = s_y$, ansonsten **nicht-uniform**

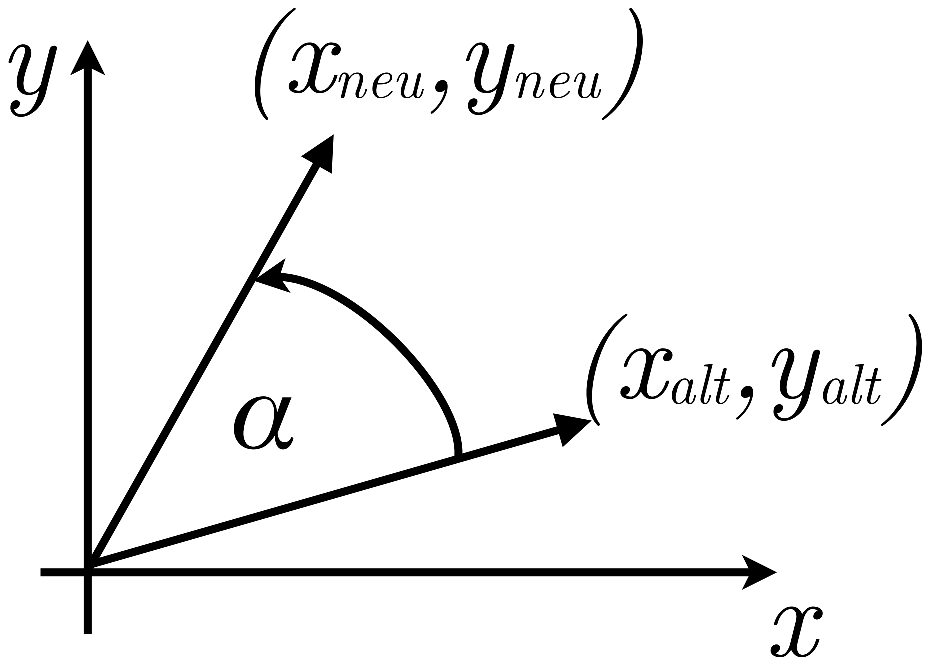
$$\begin{pmatrix} x_{neu} \\ y_{neu} \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x_{alt} \\ y_{alt} \end{pmatrix} = \begin{pmatrix} s_x \cdot x_{alt} \\ s_y \cdot y_{alt} \end{pmatrix}$$

Transformation
2D-Rotation

- **Rotation** um Winkel α

$$\begin{pmatrix} x_{neu} \\ y_{neu} \end{pmatrix} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} x_{alt} \\ y_{alt} \end{pmatrix}$$

- **positive** Winkel sind **linksdrehend**



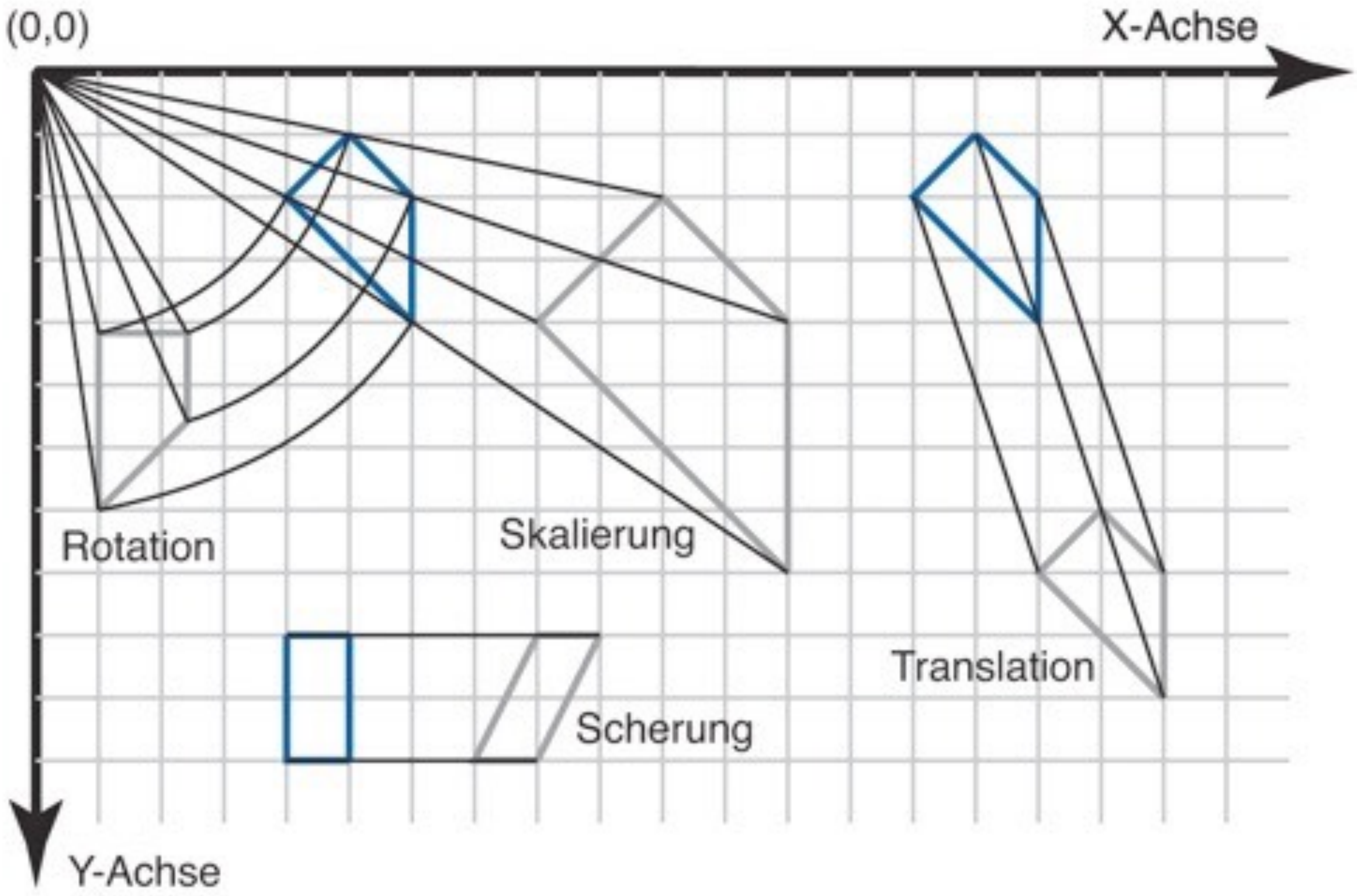
Transformation
Scherung

- **Scherung** entlang der x-Achse verändert x-Koordinate in Abhängigkeit der y -Koordinate

$$\begin{pmatrix} x_{\text{neu}} \\ y_{\text{neu}} \end{pmatrix} = \begin{pmatrix} 1 & m \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{\text{alt}} \\ y_{\text{alt}} \end{pmatrix} = \begin{pmatrix} x_{\text{alt}} + m \cdot y_{\text{alt}} \\ y_{\text{alt}} \end{pmatrix}$$

Transformation

Beispiele



Transformation

Komposition

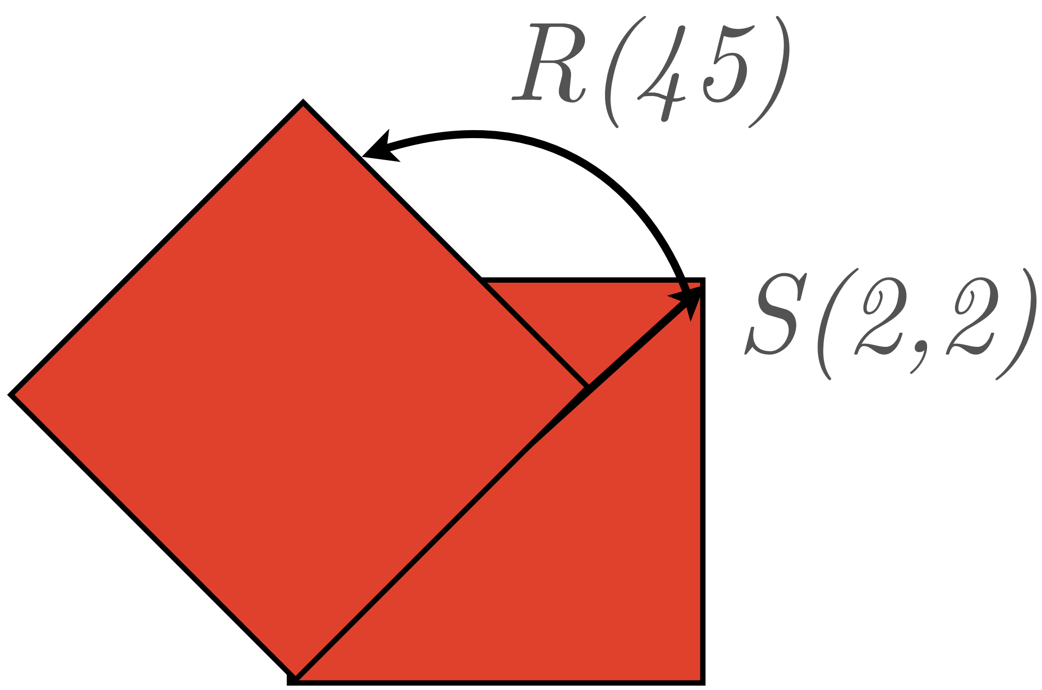
- **Komposition** bedeutet **Hintereinanderausführung** von Transformationen
- **Achtung:** Skalierung und Rotation verhalten sich unterschiedlich zu Translationen

Komposition

Beispiel

- (i) Komposition von (ii) Skalierung und (iii) Rotation

$$\underbrace{\begin{pmatrix} cos(45) & -sin(45) \\ sin(45) & cos(45) \end{pmatrix}}_{\text{(iii)}} \cdot \underbrace{\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix}}_{\text{(ii)}} = \underbrace{\begin{pmatrix} 2 \cdot cos(45) & -2 \cdot sin(45) \\ 2 \cdot sin(45) & 2 \cdot cos(45) \end{pmatrix}}_{\text{(i)}}$$

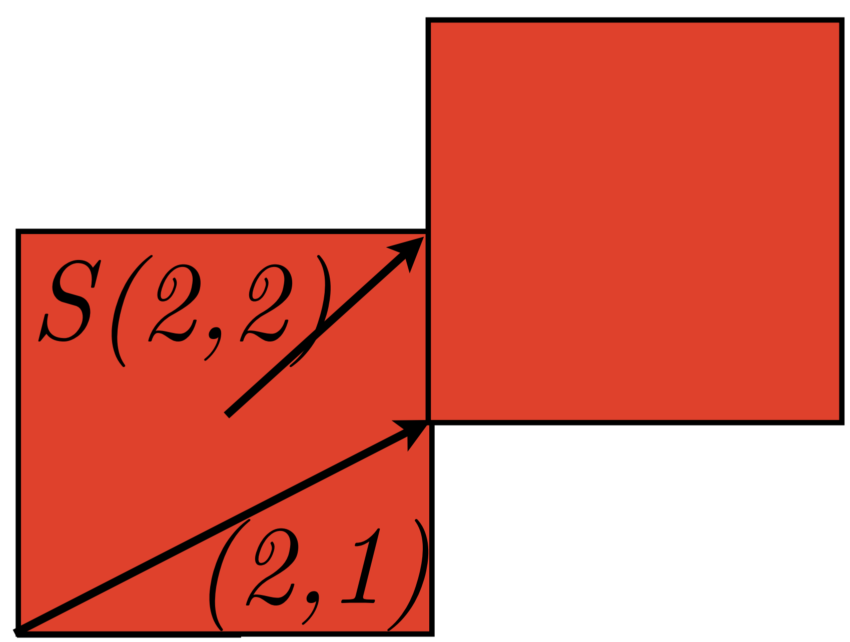


Komposition

Beispiel

- Komposition von (i) Skalierung und (ii) Verschiebung

$$\underset{\text{(ii)}}{\boxed{\begin{pmatrix}2\\1\end{pmatrix}}} + \underset{\text{(i)}}{\boxed{\begin{pmatrix}2 & 0\\0 & 2\end{pmatrix}}}$$



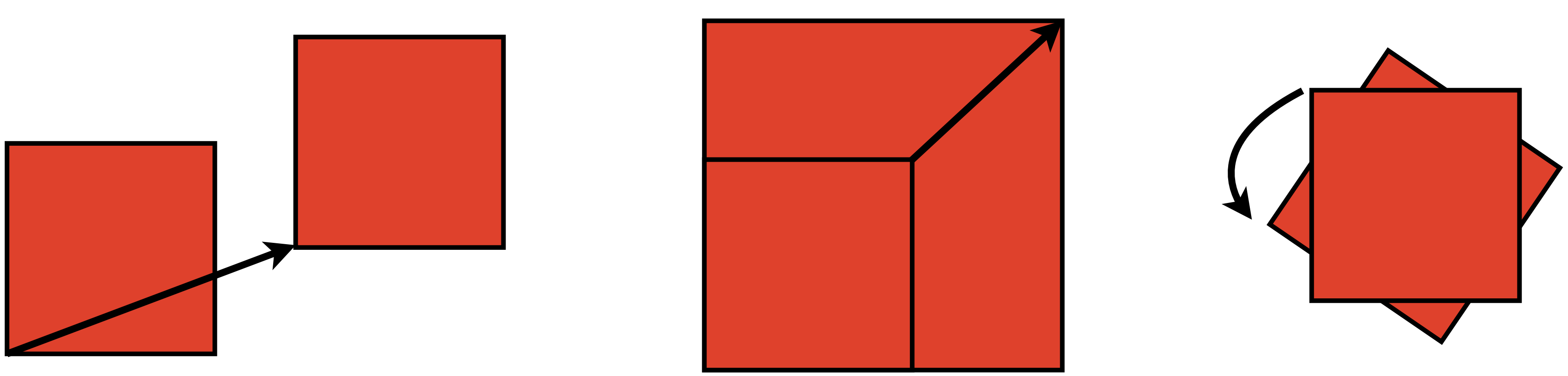
Homogene Koordinaten

- **Skalierung** und **Rotation** können durch **Matrixmultiplikation** umgesetzt werden
- **Translationen** werden über Vektoraddition umgesetzt

$$\boxed{P_{neu} = t + P_{alt}}$$

$$P_{neu} = S \cdot P_{alt}$$

$$P_{neu} = R \cdot P_{alt}$$



Homogene Koordinaten

- mit **homogenen Koordinaten** wird jeder 2D-Punkt (x,y) durch $(x,y,1)$ repräsentiert
- $(x,y,1)$ und (x,y,W) repräsentieren gleichen Punkt genau dann wenn $W \neq 0$
- $(x,y,0)$ repräsentiert 2D-Vektor (x,y) in **homogener Darstellung**

Translation
Wiederaufgegriffen

- **Translation** verschiebt jeder Vertex um gleichen Vektor (t_x, t_y)

$$\begin{pmatrix} x_{\text{neu}} \\ y_{\text{neu}} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_{\text{alt}} \\ y_{\text{alt}} \\ 1 \end{pmatrix} = \begin{pmatrix} x_{\text{alt}} + t_x \\ y_{\text{alt}} + t_y \\ 1 \end{pmatrix}$$

Transformation
Homogenisierung

- alle Transformationen können durch **Homogenisierung** umgesetzt werden

$$\begin{pmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{pmatrix} \longrightarrow \begin{pmatrix} m_{11} & m_{12} & 0 \\ m_{21} & m_{22} & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

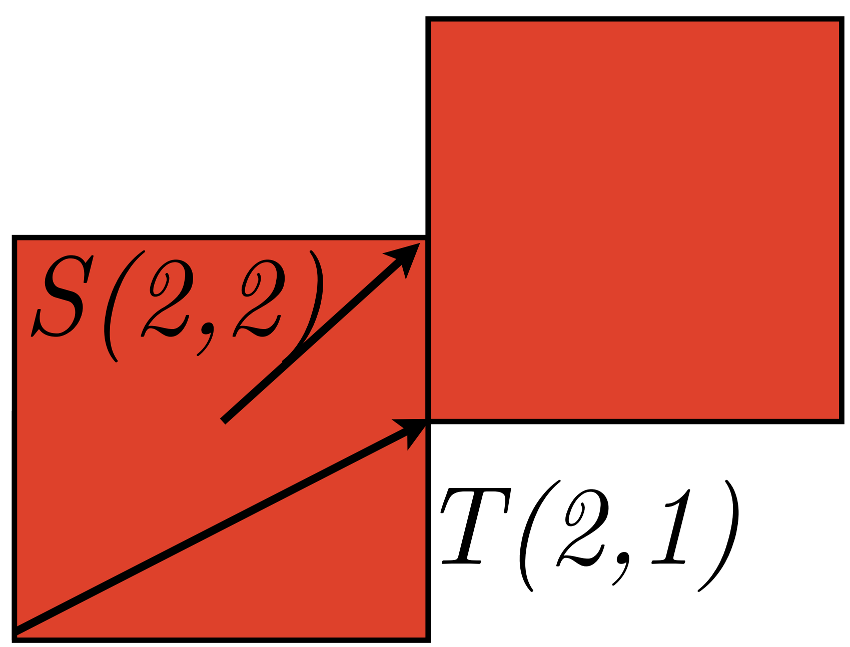
$$\begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \longrightarrow \begin{cases} \begin{pmatrix} v_1 \\ v_2 \\ 1 \end{pmatrix}, \text{falls } \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \text{ Punkt} \\ \begin{pmatrix} v_1 \\ v_2 \\ 0 \end{pmatrix}, \text{falls } \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \text{ Vektor} \end{cases}$$

Komposition

Bsp: Homogene Transformation

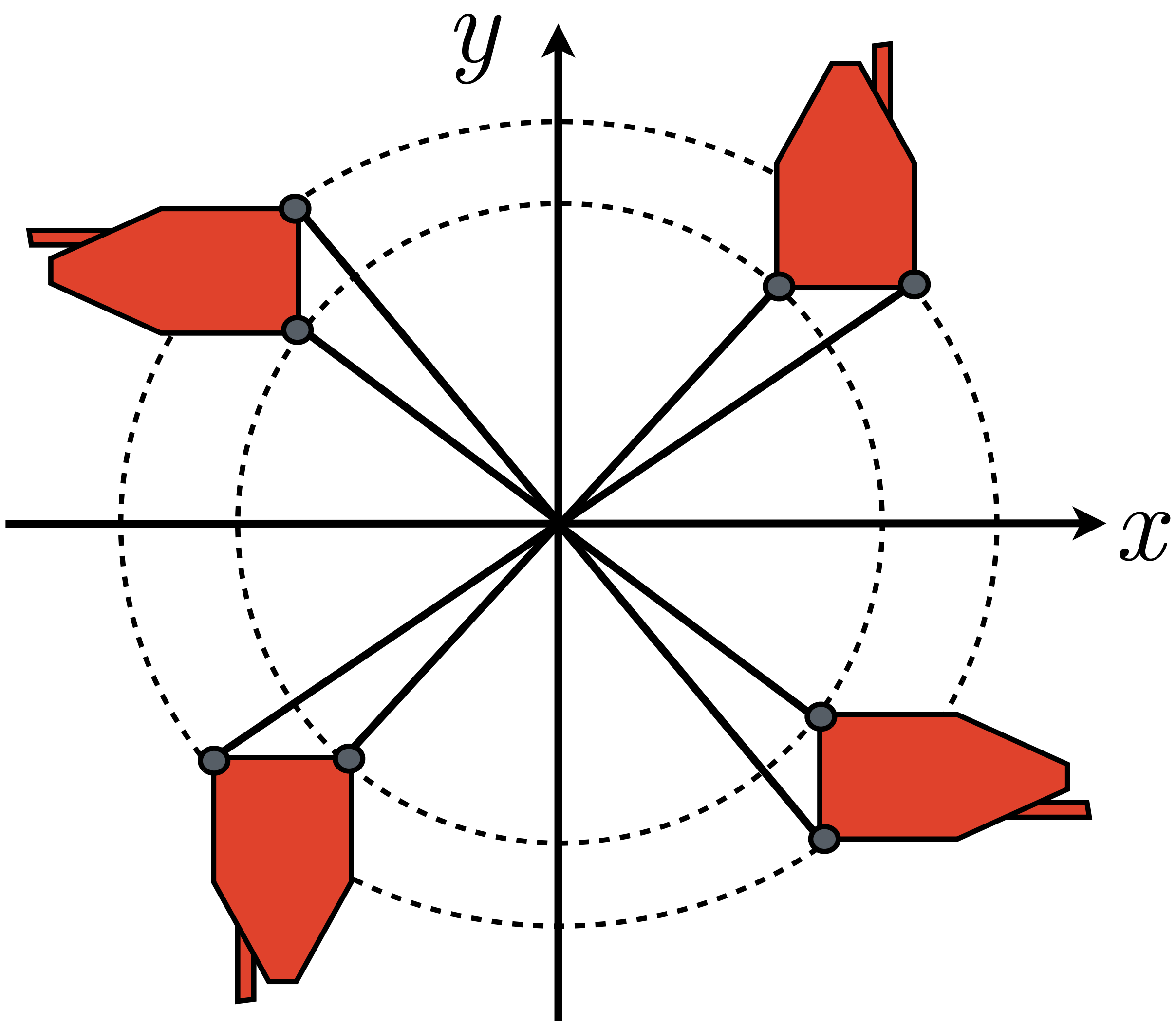
- (i) Komposition von (ii) Skalierung und (iii) Verschiebung

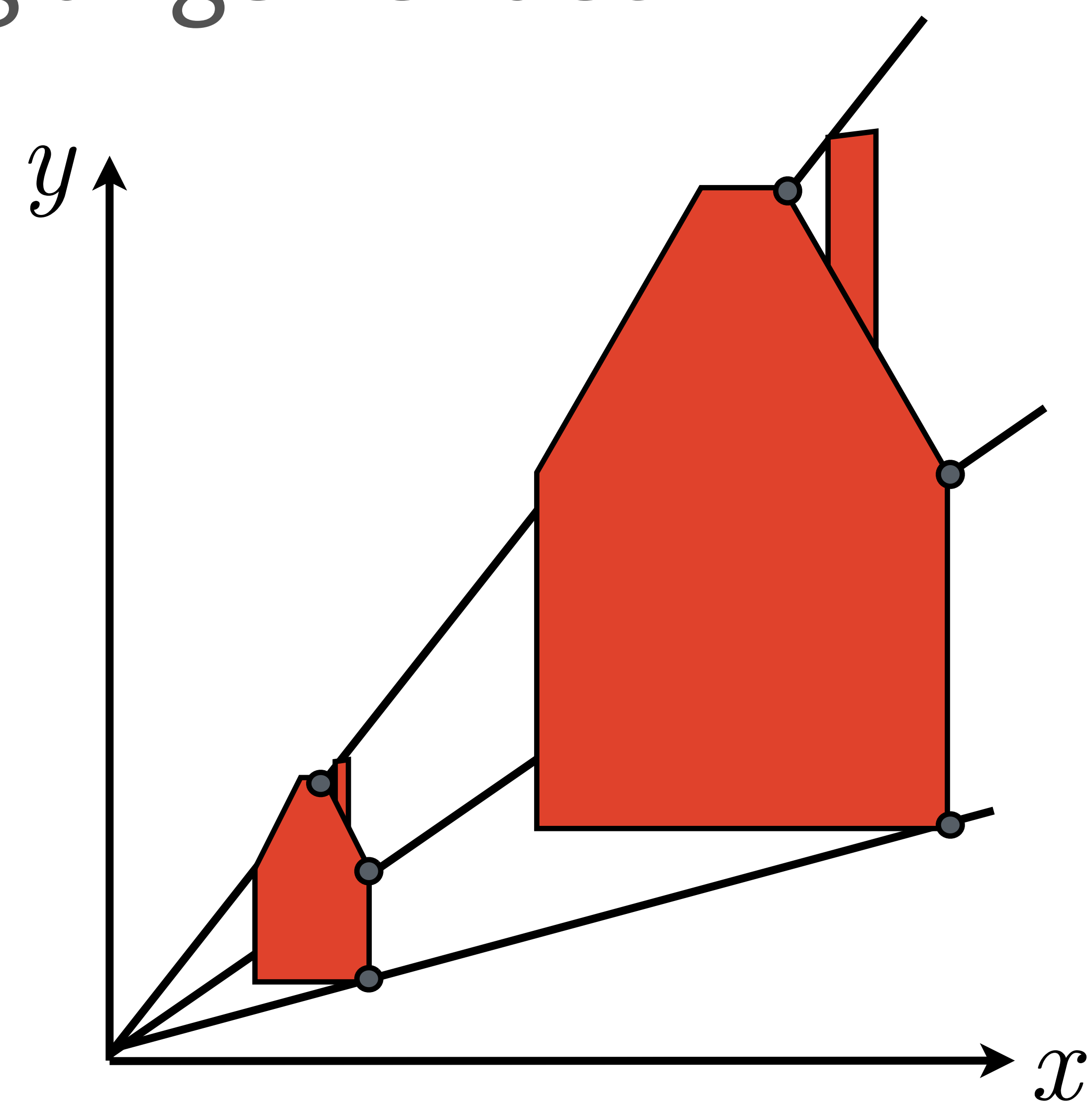
$$\underset{\text{(iii)}}{\begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}} \cdot \underset{\text{(ii)}}{\begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix}} = \underset{\text{(i)}}{\begin{pmatrix} 2 & 0 & 2 \\ 0 & 2 & 1 \\ 0 & 0 & 1 \end{pmatrix}}$$



Komposition

- Rotationen und Skalierungen werden relativ zum Ursprung angewendet





Komposition

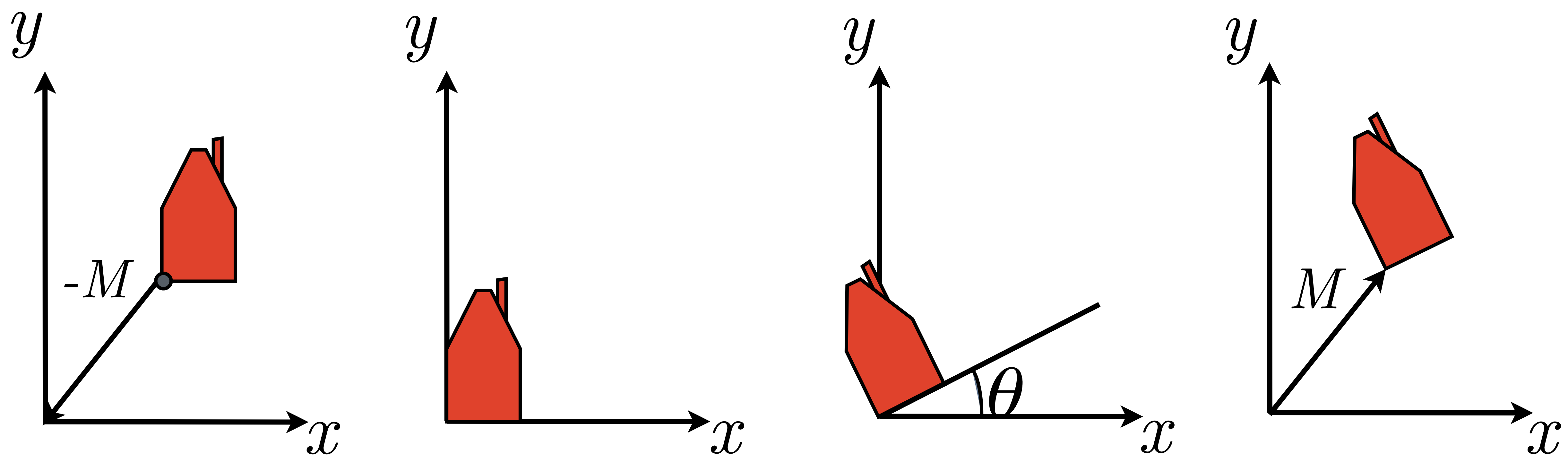
Beispiel: Rotation

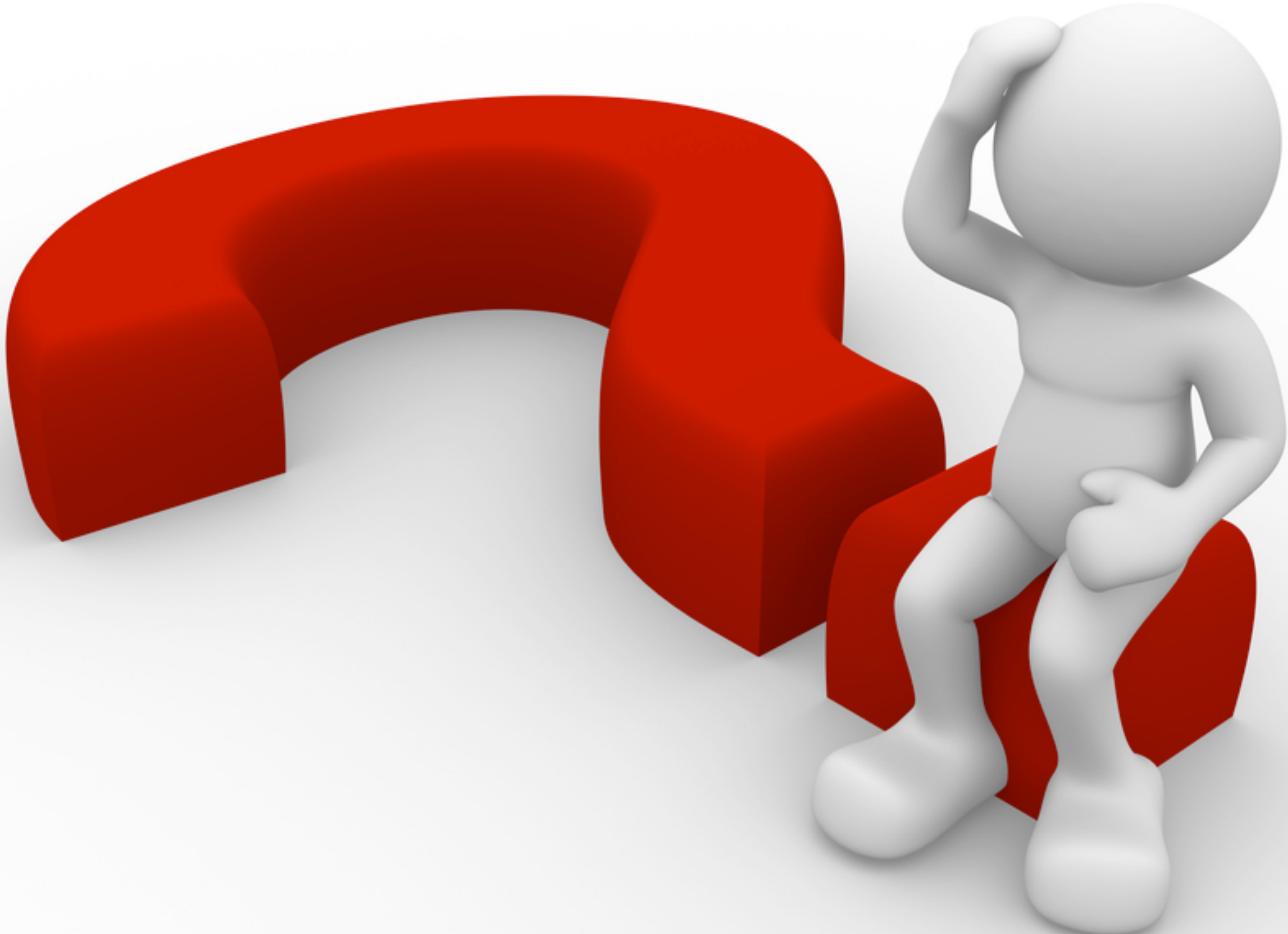
- Um Objekte um beliebigen **Bezugspunkt** zu rotieren oder zu skalieren, muss **Transformationssequenz** angewendet werden
- Beispiel: $\theta°$-Rotation um $M = (x_M, y_M)$:

1. Verschiebe M zum Ursprung O
2. Rotiere um $\theta°$
3. Verschiebe zurück zu M

Komposition
Beispiel: Rotation

$$T(x_M, y_M) \cdot R_z(\theta) \cdot T(-x_M, -y_M)$$



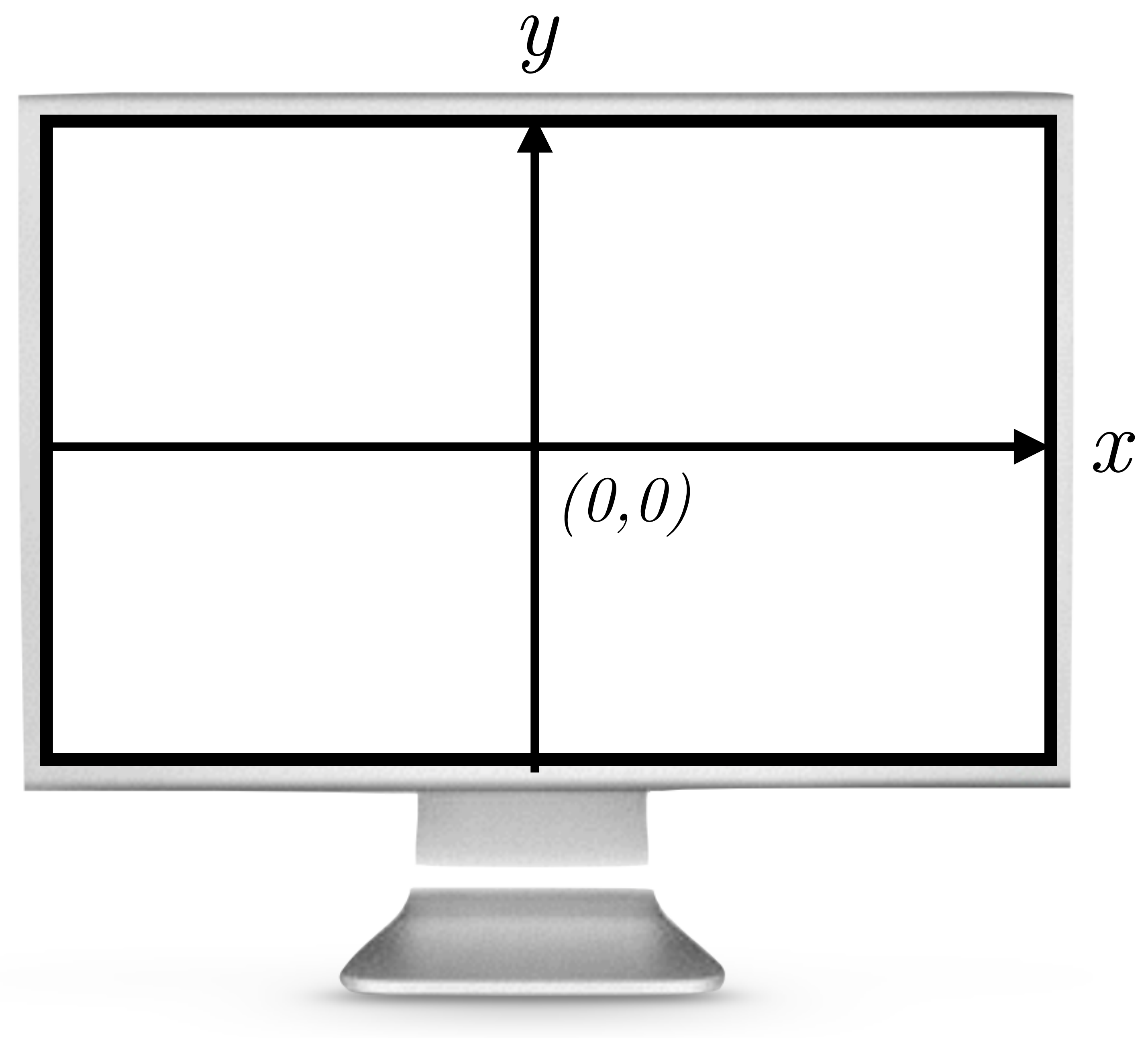


Interaktive Computergrafik

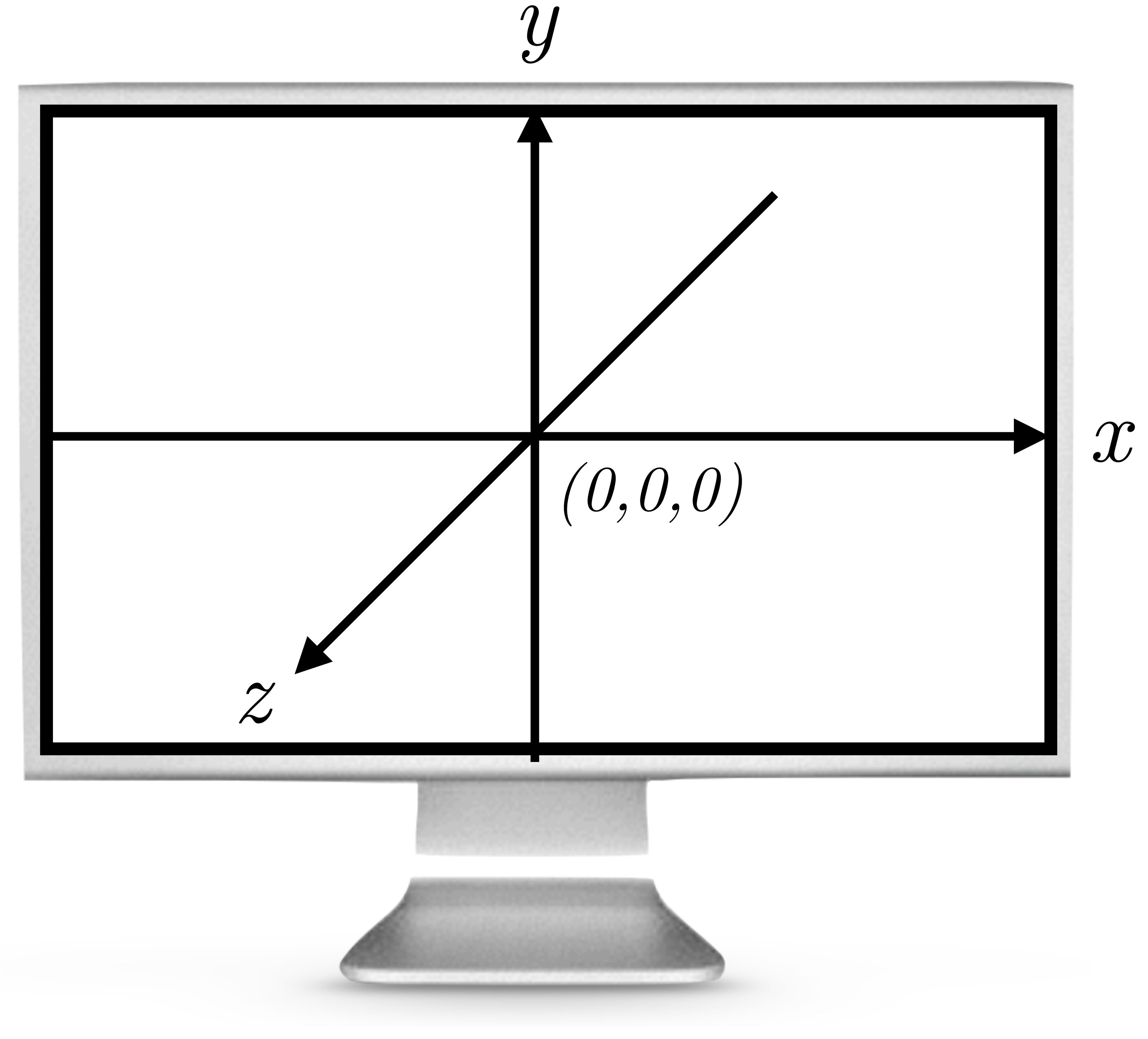
Kapitel Transformationen

Die 3te Dimension

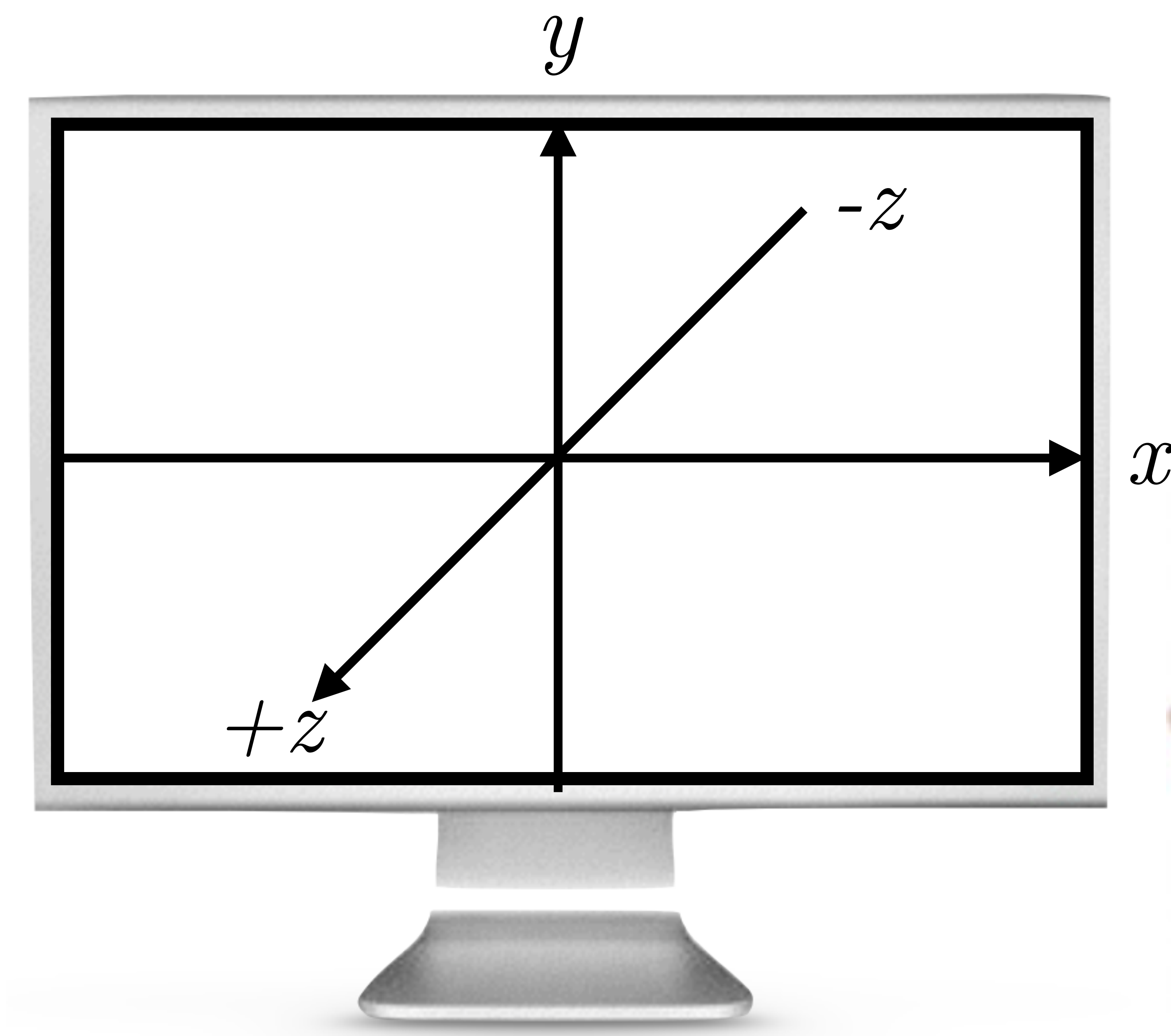
2D-Bildebene

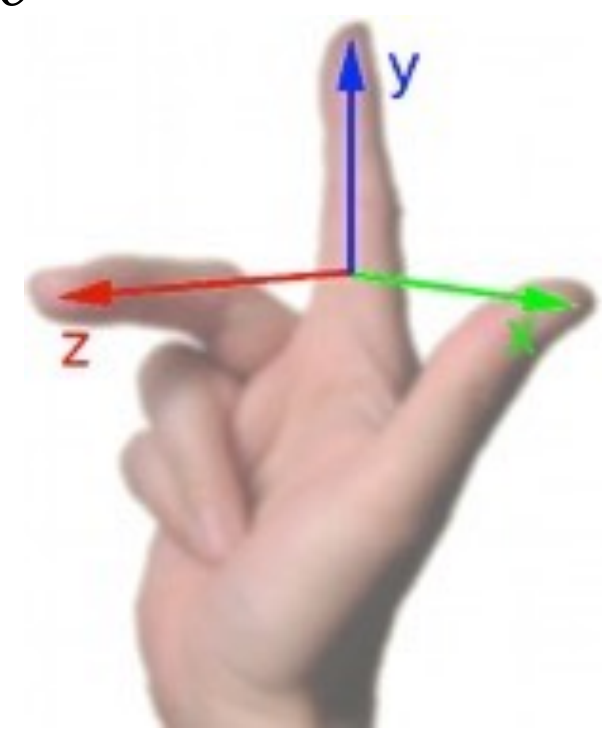


Die 3te Dimension

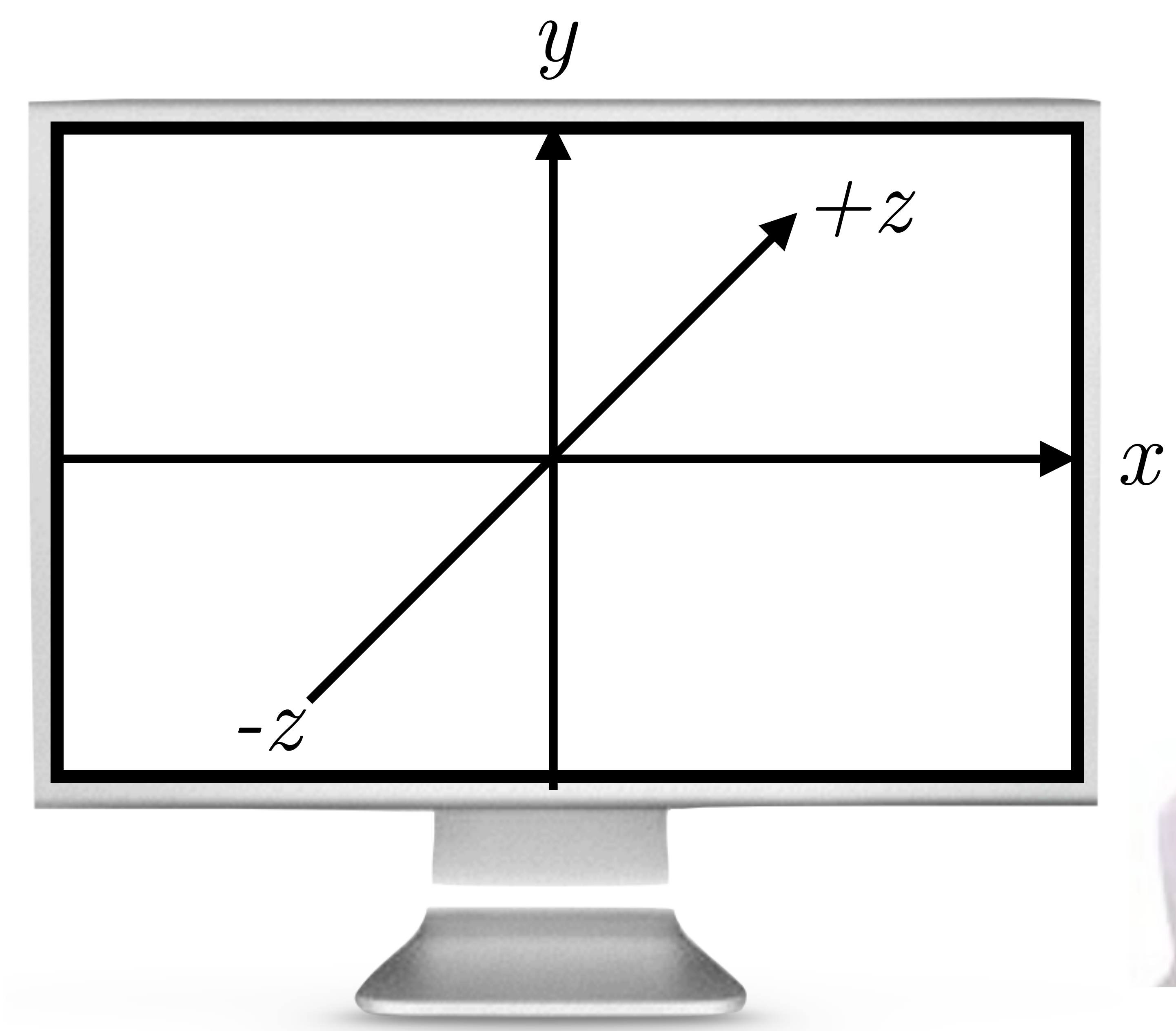


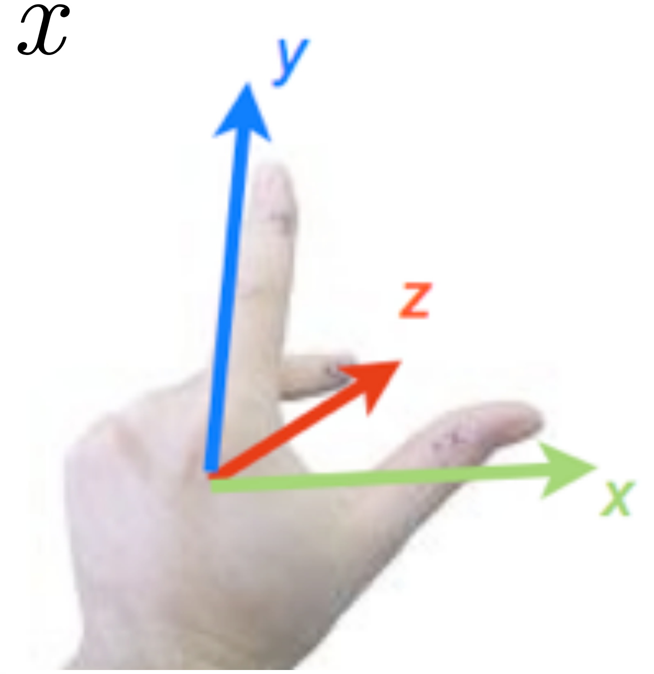
Koordinatensystem
Rechtshändig





Koordinatensystem
Linkshändig

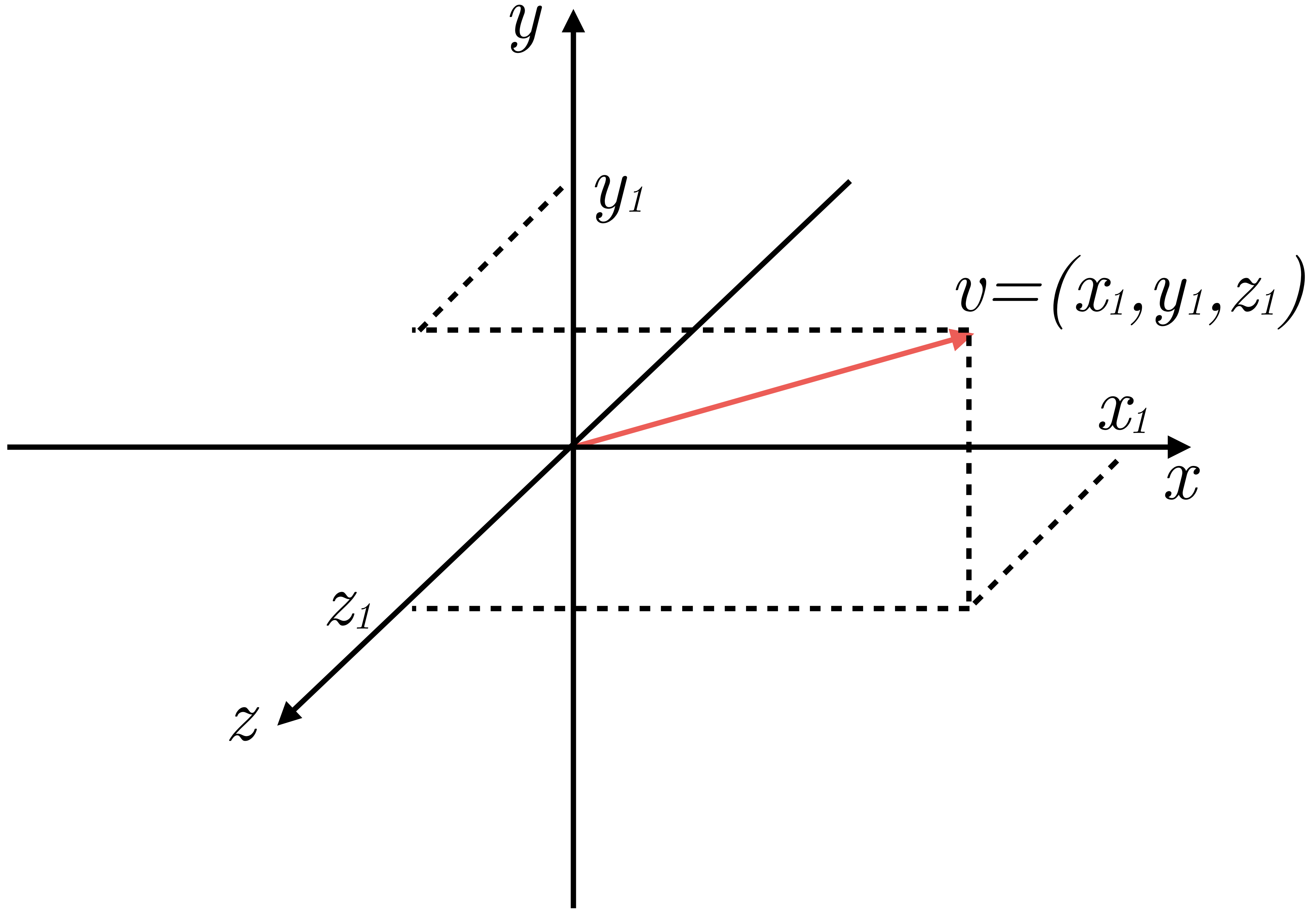




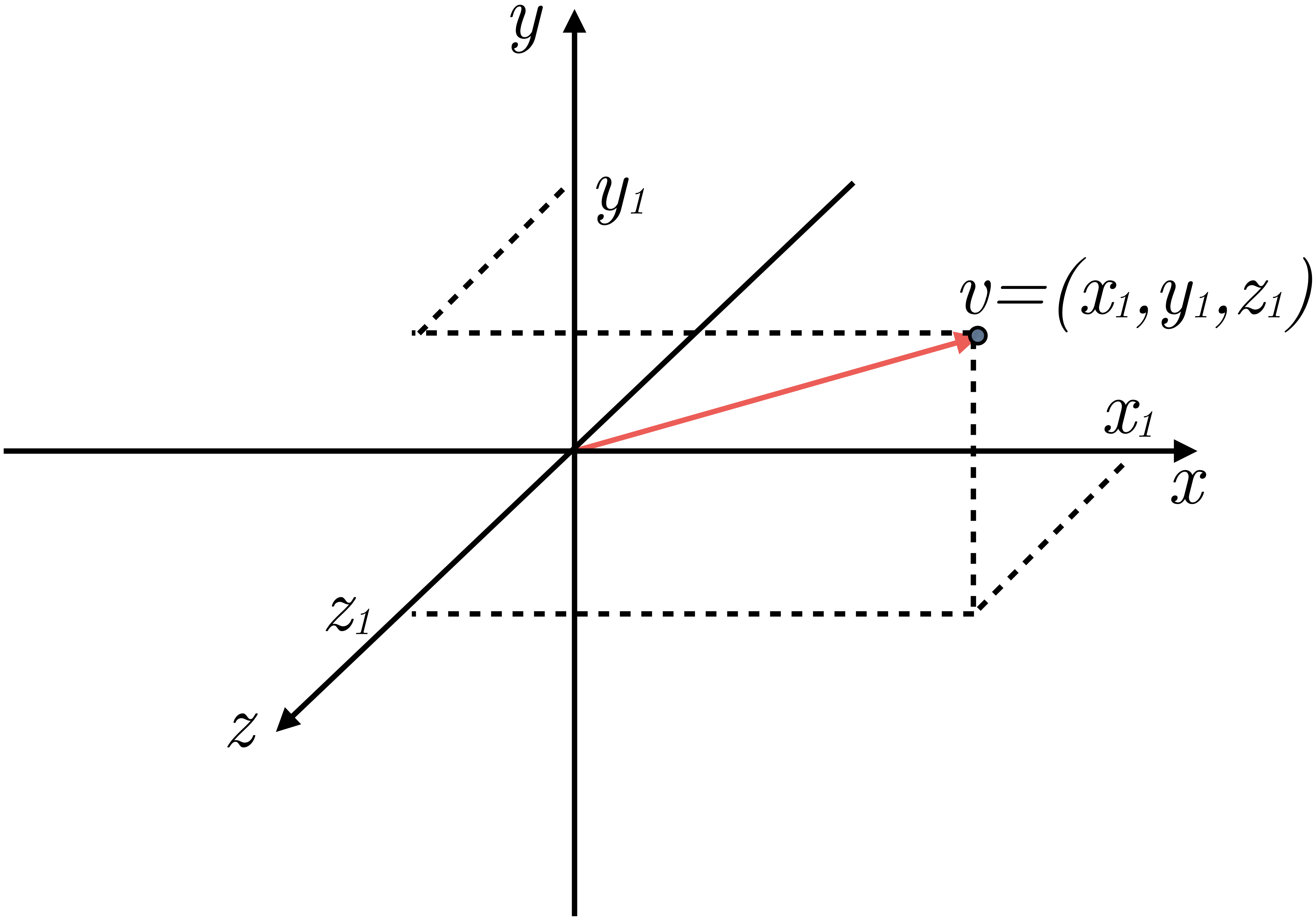
Koordinatensysteme

- **kein Standard** bzgl. links- oder rechtshändiger Koordinatensysteme in der Computergrafik
  - *Beispiele: OpenGL (rechtshändig), DirectX (linkshändig)*

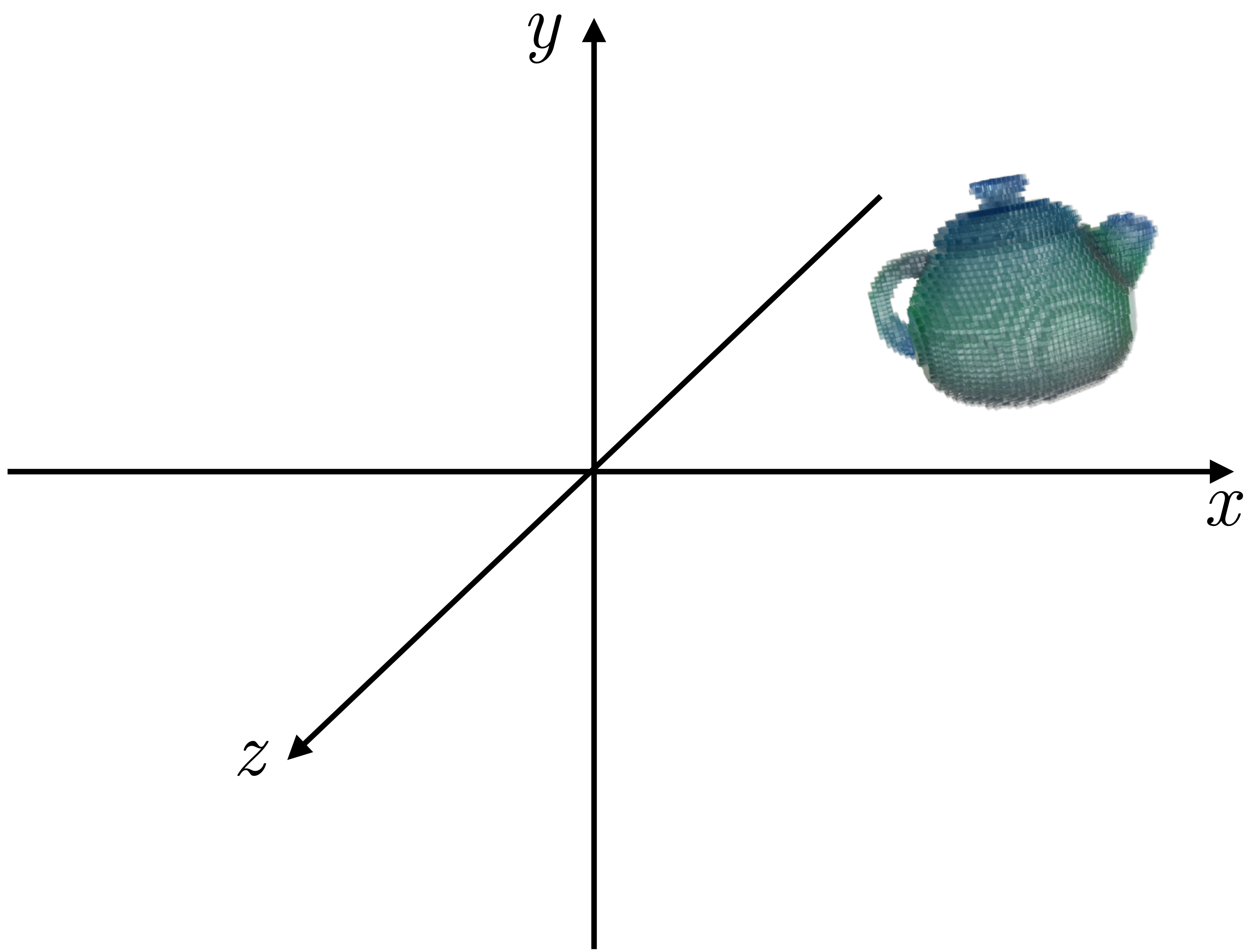
3D-Vektoren



3D-Punkte



3D-Punkte

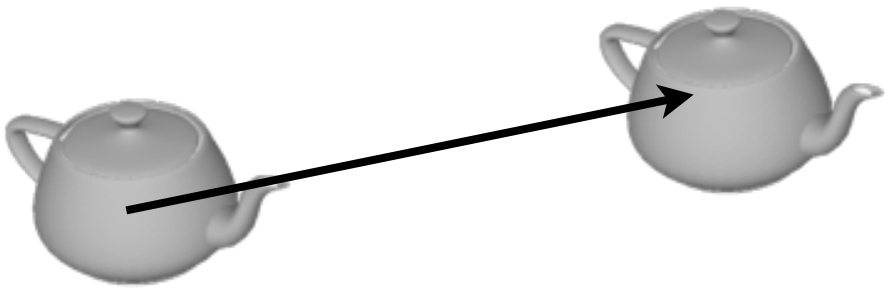


3D-Vektoren/-Punkte

- In **homogenen Koordinaten** wird jeder…
  - **3D-Punkt** (x,y,z) durch $(x,y,z,1)$ repräsentiert
  - **Richtungsvektor** (x,y,z) durch $(x,y,z,0)$ repräsentiert

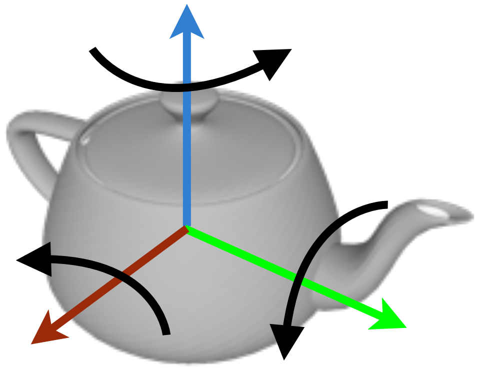
Grundtransformationen

- Translation



- Skalierung

- Rotation



3D-Translationen

$$T(d_x, d_y, d_z) \cdot P = \underbrace{\begin{pmatrix} 1 & 0 & 0 & d_x \\ 0 & 1 & 0 & d_y \\ 0 & 0 & 1 & d_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Translations-matrix}} \cdot \underbrace{\begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}}_{\text{3D-Punkt}} = \begin{pmatrix} P_x + d_x \\ P_y + d_y \\ P_z + d_z \\ 1 \end{pmatrix}$$

Translations-matrix

3D-Punkt

3D-Translationen

- erhalten **Längen** und **Winkel**
- **Identität**
  - $T(0,0,0) = E$ (Einheitsmatrix)
- **Inverse**
  - $T^{-1}(d_x,\ d_y,\ d_z) = T(-d_x,\ -d_y,\ -d_z)$

3D-Translationen

- **Komposition**
  - $T(d1_x,\ d1_y,\ d1_z) \cdot T(d2_x,\ d2_y,\ d2_z) = T(d1_x + d2_x,\ d1_y + d2_y,\ d1_z + d2_z)$
- **Kommutativität**
  - $T1 \cdot T2 = T2 \cdot T1$

3D-Skalierungen

$$S(s_x, s_y, s_z) \cdot P = \underbrace{\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{\text{Skalierungs-matrix}} \cdot \underbrace{\begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix}}_{\text{3D-Punkt}} = \begin{pmatrix} s_x \cdot P_x \\ s_y \cdot P_y \\ s_z \cdot P_z \\ 1 \end{pmatrix}$$

3D-Skalierungen

- **winkelerhaltend** nur bei **uniformer** Skalierung ($s_x = s_y = s_z$)
- **Identität**
  - $S(1,1,1) = E$ (*Einheitsmatrix*)
- **Inverse**
  - $S^{-1}(s_x,\ s_y,\ s_z) = S(1/s_x,\ 1/s_y,\ 1/s_z)$

3D-Skalierungen

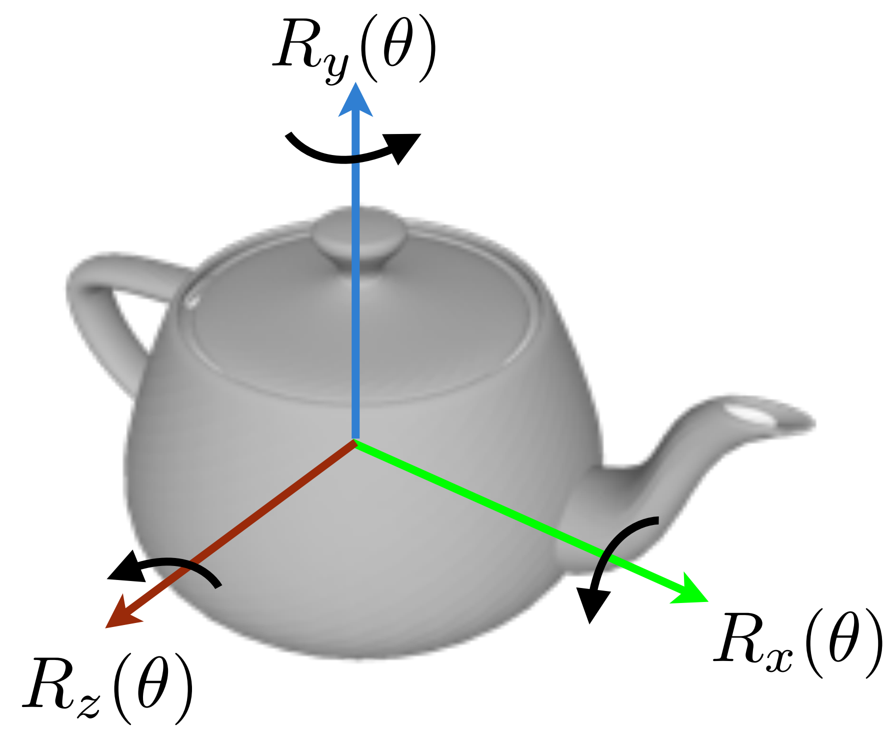
- **Komposition**
  - $S(s1_x, s1_y, s1_z) \cdot S(s2_x, s2_y, s2_z) = S(s1_x \cdot s2_x, s1_y \cdot s2_y, s1_z \cdot s2_z)$
- **Kommutativität**
  - $S1 \cdot S2 = S2 \cdot S1$

3D-Rotationen

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

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

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

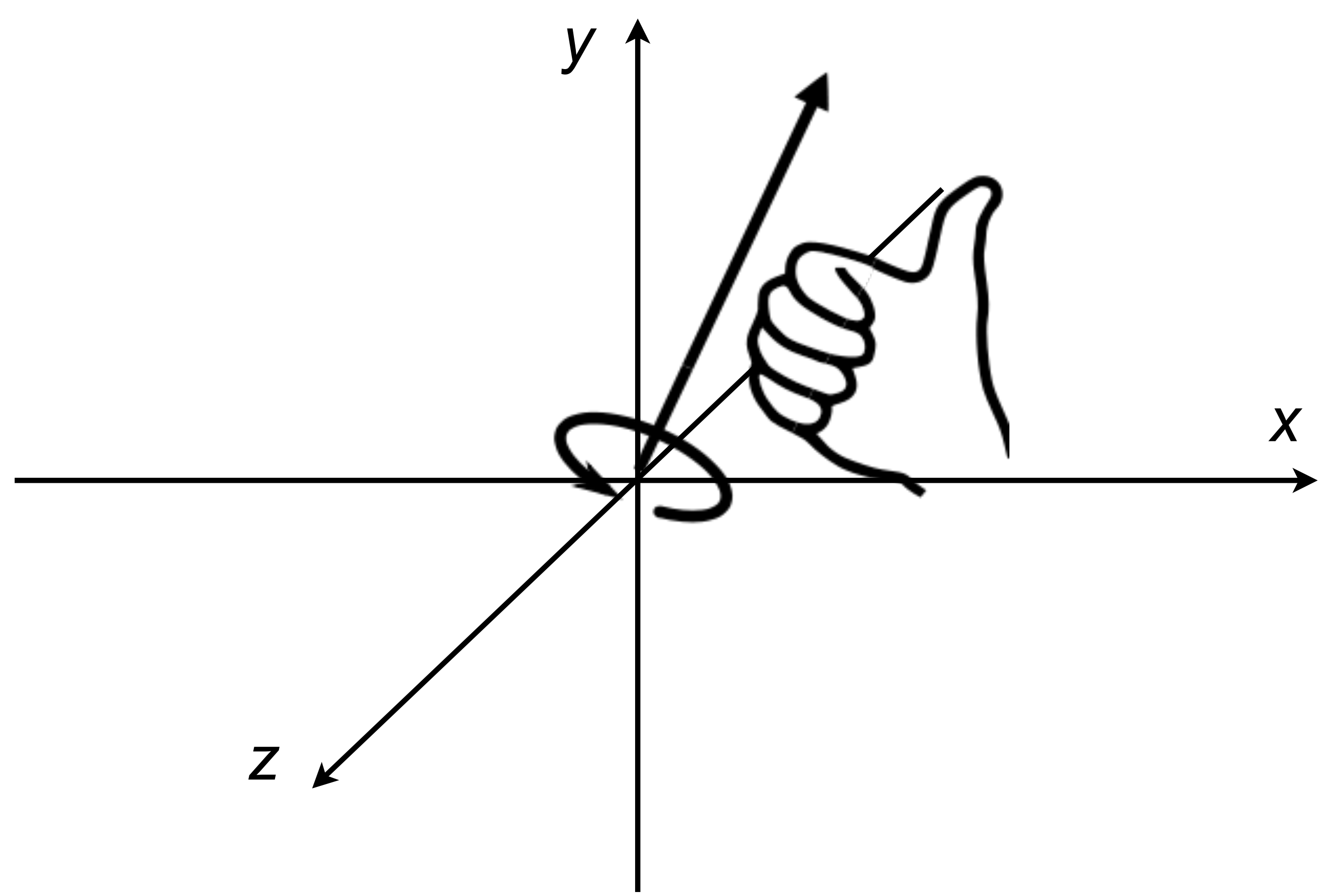


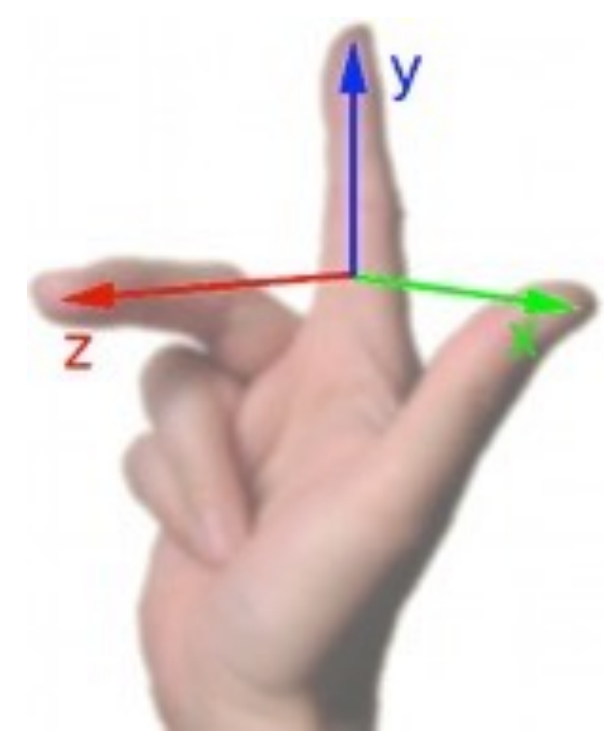
Rotationsmatrizen

3D-Rotationen

Rechte-Hand-Regel

- **positiver** Winkel rotiert **gegen den Uhrzeigersinn** in rechtshändigen Koordinatensystemen





3D-Rotationen

- **längen**- und **winkelerhaltend**
- **Identität**
  - $R_x(0) = R_y(0) = R_z(0) = E$ **(Einheitsmatrix)**
- **Inverse** (für beliebige Achse a)
  - $R_a^{-1}(\theta) = R_a(-\theta)$

3D-Rotationen

- **Komposition** (nur um gleiche Achse *a*)
  - $R_a(\phi) \cdot R_a(\theta) = R_a(\phi + \theta)$
- **Kommutativität** (nur um gleiche Achse *a*)
  - $R_a(\phi) \cdot R_a(\theta) = R_a(\theta) \cdot R_a(\phi)$

Komposition

- **Komposition** mehrerer nacheinander angewendeter Transformationen ergibt **Transformationssequenz**
- Leserichtung von **rechts nach links**
- Beispiel:

$$P_{neu} = T(d_x, d_y, d_z) \cdot R_y(\alpha_3) \cdot R_z(\alpha_2) \cdot R_x(\alpha_1) \cdot P_{alt}$$

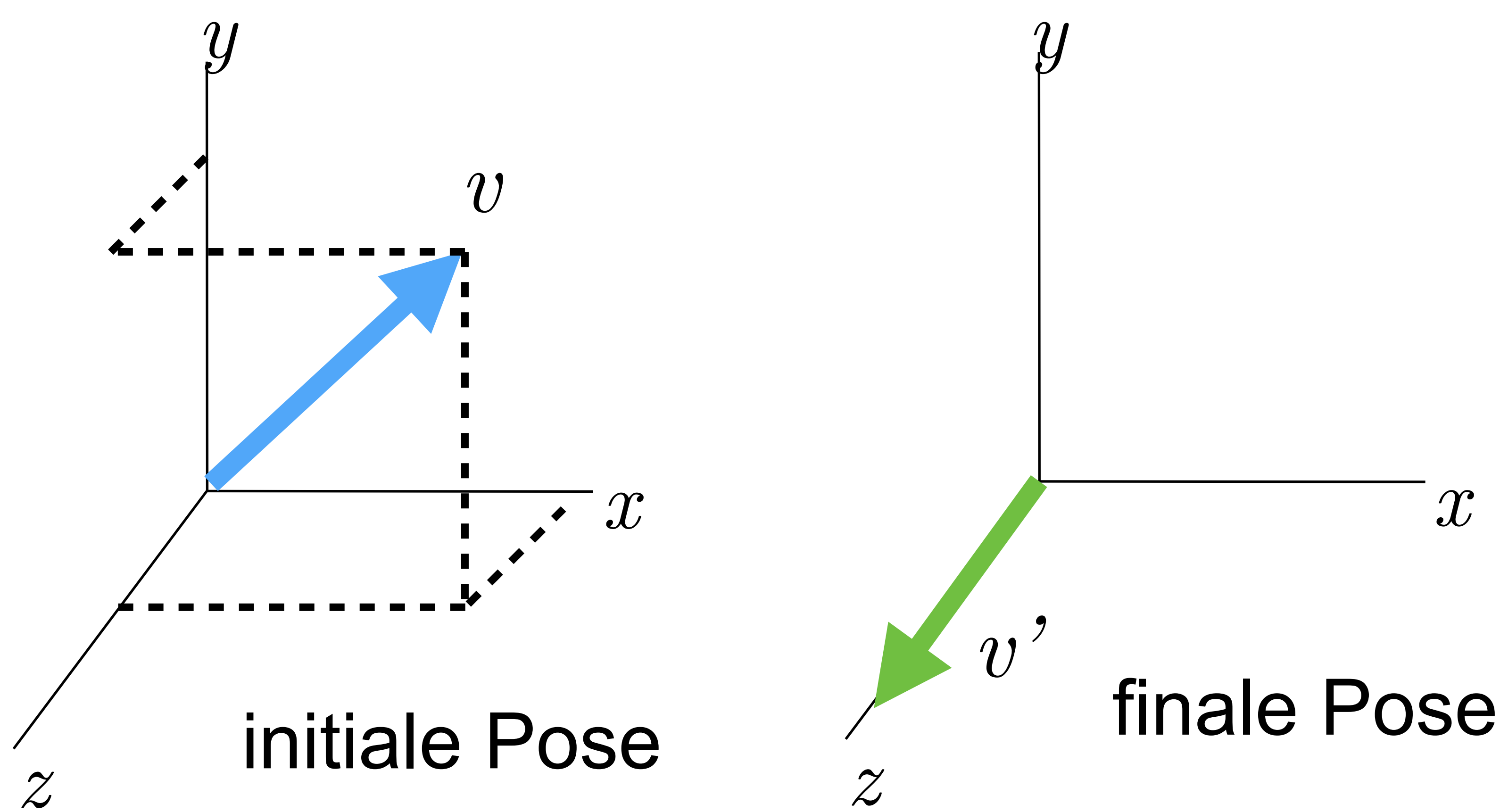
Komposition durch Matrixmultiplikation der 4x4-Matrizen

Komposition

- **Kompositionsmatrix** $M_{composite}$ ergibt sich durch Matrixmultiplikation der Transformationsmatrizen $M_1, \ldots, M_n$

$$
\begin{aligned}
& M_1 \cdot M_2 \cdot \ldots \cdot M_n \cdot P \\
= & \ (M_1 \cdot (M_2 \cdot \ldots \cdot (M_n \cdot P))) \\
= & \ (M_1 \cdot M_2 \cdot \ldots \cdot M_n) \cdot P \\
= & \ M_{composite} \cdot P
\end{aligned}
$$

Diskussion



Spezifiziere Transformationssequenz um blauen Vektor auf grünen Vektor zu rotieren!

Mögliche Lösung
Rotationssequenz

1. z-Rotation um Vektor in xz-Ebene zu bringen
2. y-Rotation um Vektor auf z-Achse zu bringen