§3 Transformationen und Projektionen

3.1 Koordinatentransformationen

- **Beispiel:** Koordinatensysteme der Objekte und des Ausgabegerätes unterscheiden sich:
  - Objektsystem: 3D-Koordinatensystem des Objektes ist über geometrische Eigenschaften des Objektes festgelegt, z.B. ausgezeichnete Richtungen, Symmetrien.
  - Gerätesystem: 2D-Koordinatensystem des Bildschirms oder die Größe der Bildfenster ist durch das Gerät selbst definiert, z.B. Nullpunkt in der linken, oberen Ecke, x- und y-Achsen parallel zu den Bildrändern.
  - Koordinatentransformationen im $\mathbb{R}^2$ oder $\mathbb{R}^3$ transformieren das Objektsystem in das Gerätesystem: Translationen, Skalierungen, Rotationen, Projektionen.

3.1 Koordinatentransformationen

Motivation

Objektdefinition, Szenenmodellierung, Sichtdefinition, Animation, Rendering, Ausgabe, ...

→ Einsatz verschiedenster Koordinatensysteme

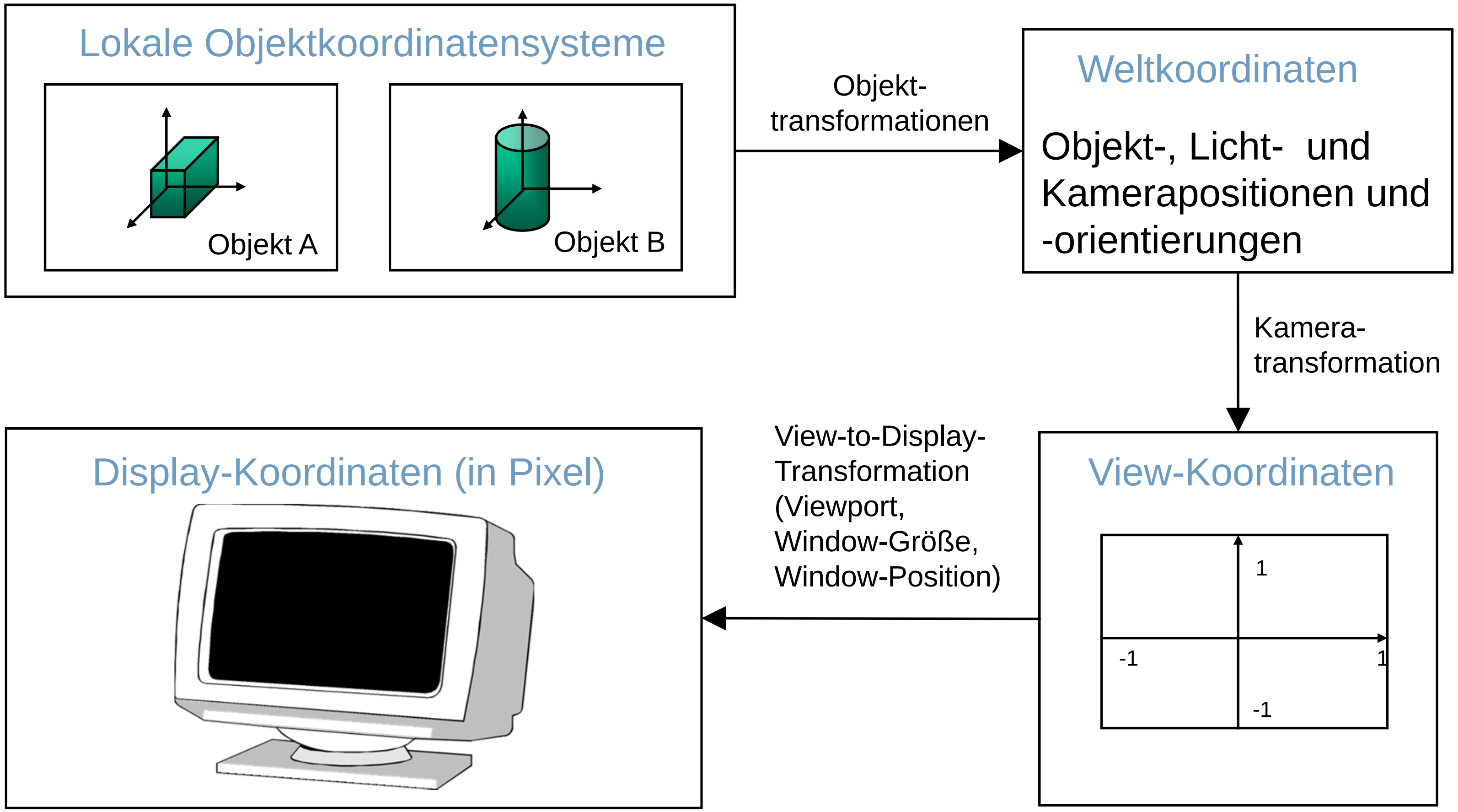
→ Koordinatentransformationen im 2D-, 3D-Raum
(Verschiebungen, Skalierungen, Drehungen,...)
zur Definition von Position und Orientierung!

→ Matrixoperationen

→ Hard- und softwaregestützte Umrechnung / Transformation zwischen Koordinatensystemen
(i.d.R. bis auf Translationen Basistransformationen im $\mathbb{R}^3$)

3.1 Koordinatentransformationen

Überblick / Zusammenspiel:



3.1 Koordinatentransformationen

3.1.1 Objektkoordinatensystem (local / object / modeling coordinate system)

- Eigenes **3D**-Koordinatensystem für jedes Objekt sinnvoll.
- Erleichtert Modellierung und Animation.
- Oft über
  - geometrische Eigenschaften des Objekte, z.B. Symmetrien oder ausgezeichnete Richtungen, oder
  - gewünschte Aktionen des Objektes, z.B. Flug entlang Kurve mit gleichzeitiger Drehung,

  festgelegt.
- Auch für Lichtquellen möglich.

3.1 Koordinatentransformationen

3.1.2 Weltkoordinatensystem (world / global / scene coordinate system)

- Hier „lebt“ die gesamte Szene im **3D**.
- Objekte (Objektkoordinatensysteme) werden mittels Transformationen platziert.
- Zum Rendern: Umrechnung lokaler Objektkoordinaten in globale Objektkoordinaten mit den **Objekttransformationen**.
  - Bei Animationen d.h. animiertes Objekt: dynamisch in jedem Frame.
- Hier ebenso: Beleuchtung der Szene.

3.1 Koordinatentransformationen

3.1.3 Sichtkoordinatensystem (eye / camera coordinate system)

- Konzept: virtuelle Kamera im **3D** mit entsprechenden Parametern.
- Im Weltsystem (mittels Transformationen) verankert.
- So soll es der Beobachter sehen, d.h. hier wird auch der Bildausschnitt (Sichtfenster siehe Seite §3-8) definiert!
- Zum Rendering: i.d.R. Umrechnung globaler Koordinaten in Sichtkoordinaten („Szene vor die Kamera drehen").
  - Bei Animationen d.h. animierter Kamera: dynamisch in jedem Frame!
- Virtuelle Kamera bestimmt auch Art der Projektion.

3.1 Koordinatentransformationen

3.1.4 View Coordinate System / View Plane Coordinate System

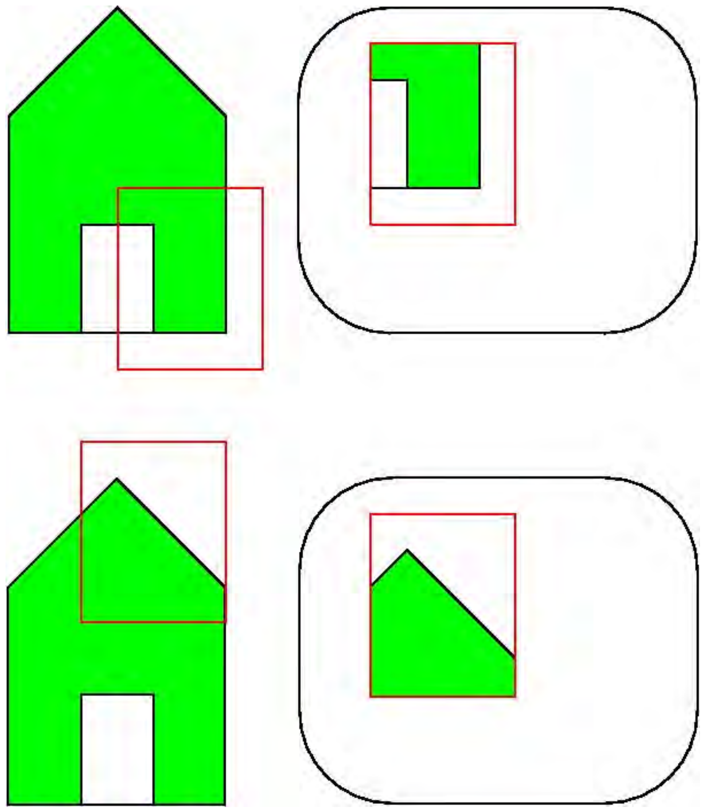
- Koordinatensystem in der **2D**-Bildebene.
- Durch Projektion (**Kameratransformation**) werden den 3D-Koordinaten 2D-Koordinaten der Bildebene zugeordnet.
- Ein **Window** definiert ein Sichtfenster in der Bildebene.

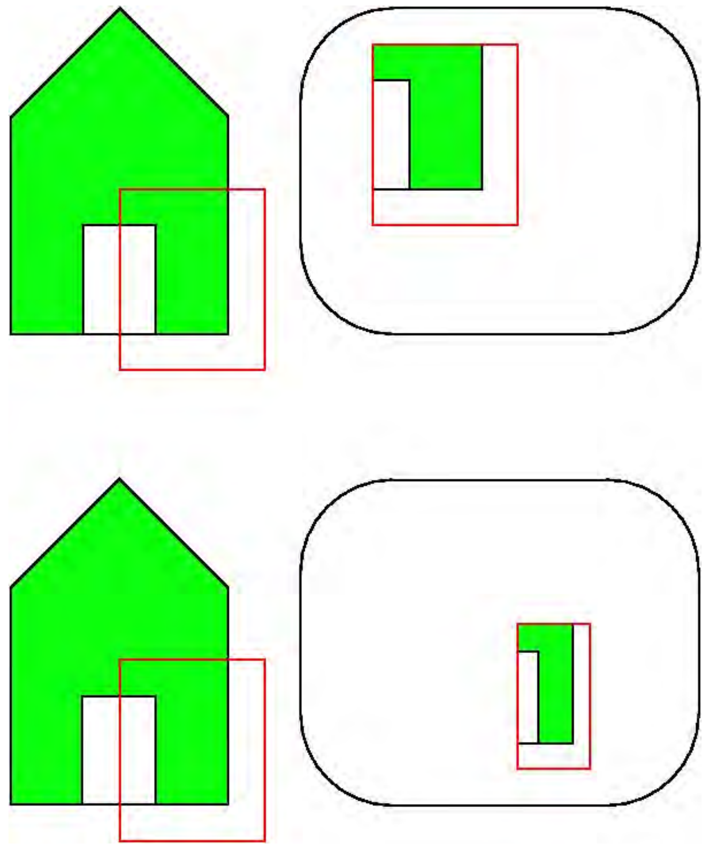
3.1 Koordinatentransformationen

3.1.5 Display Coordinate System / Device Coordinate System

- Definiert das **2D**-Koordinatensystem des Bildschirms (Display) oder Ausgabegerätes (Device).
- Ein **Viewport** definiert einen Bereich des Bildschirms, in dem der Inhalt eines Windows dargestellt werden soll.
- Window-Viewport-Transformationen (**Windowing**) sind 2D-2D-Transformationen.

3.1 Koordinatentransformationen



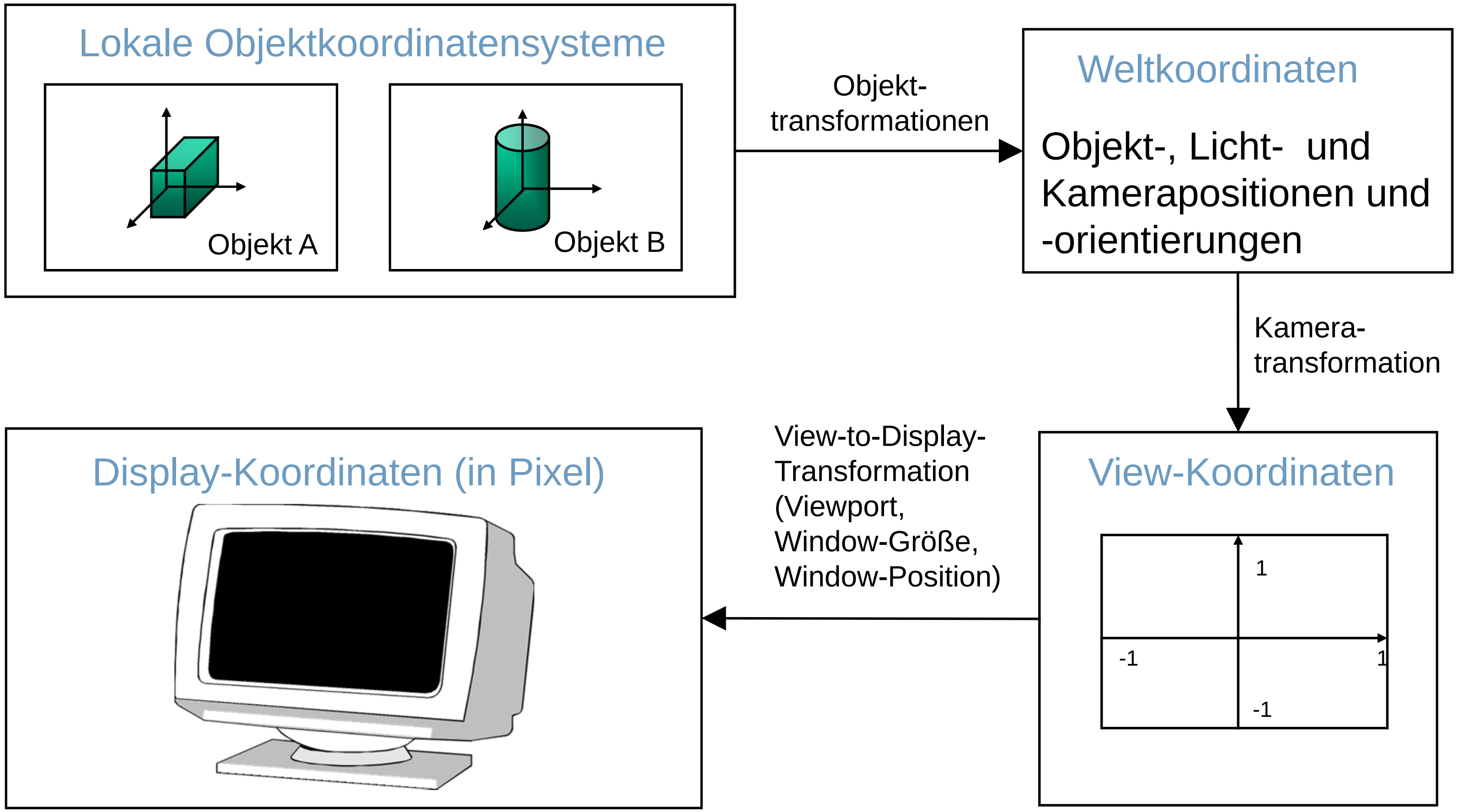


Verschiedene Fenster, dieselben Viewports

Dieselben Fenster, verschiedene Viewports

3.1 Koordinatentransformationen

Überblick / Zusammenspiel:



3.1 Koordinatentransformationen

- Generell wird ein
  - orthonormiertes: paarweise senkrechte und normierte Basisvektoren,
  - kartesisches: orthonormal + Rechtssystem,

  Koordinatensystem vorausgesetzt.

- Aber **Vorsicht vor Verwirrungen!**

Für Transformationen existieren mehrere **äquivalente Sichtweisen!**

3.1 Koordinatentransformationen

3.1.6 Sichtweisen von Transformationen

a) „Transformation der Punkte“

- Koordinatensystem bleibt fest.
- Punkt P ändert Ort und damit Koordinaten im gleichen System, d.h. Punkttransformation.

➡ Im (globalen) Koordinatensystem S wirkt die Transformation auf die Koordinaten von P.

- Beteiligte Koordinatensysteme: eins.

3.1 Koordinatentransformationen

b) „Transformation des Koordinatensystems“

- Koordinatensystem wird der inversen Transformation der Punkttransformation unterworfen.

➡ Koordinatensystem bewegt sich.

➡ Punkt bleibt fest, ändert allerdings die Koordinaten, da in neuem System beschrieben.

- Beteiligte Koordinatensysteme: eigentlich nur eins
  (aber: vorher, nachher)

➡ „Transformation zwischen Koordinatensystemen.“

➡ Basiswechsel

3.1 Koordinatentransformationen

c) „Transformation mittels Koordinatensystem“

- Ein dem Punkt P lokales Koordinatensystem S wird mittels der Punkttransformation im globalen Koordinatensystem S' bewegt.

➡ Lokales Koordinatensystem S bewegt sich.

- Anschließend wird der Punkt P (mit „alten“ Koordinaten bzgl. S) in bewegtes lokales Koordinatensystem eingetragen.

➡ Punkt ändert Ort.

- Beteiligte Koordinatensysteme: zwei (lokal, global).
- Geeignet für Modellierung und Animation.

3.1 Koordinatentransformationen

Wir verwenden zunächst parallel die Sichtweisen

„Transformation des Koordinatensystems“ (Sicht b)) und

„Transformation der Punkte“ (Sicht a)).

Gegeben:

- Koordinatensystem S' (z.B. Weltsystem) gegeben durch

  S': $(O'; x'_1, x'_2)$, und
- Koordinatensystem S (z.B. Objektsystem) gegeben durch

  S: $(O; x_1, x_2)$.

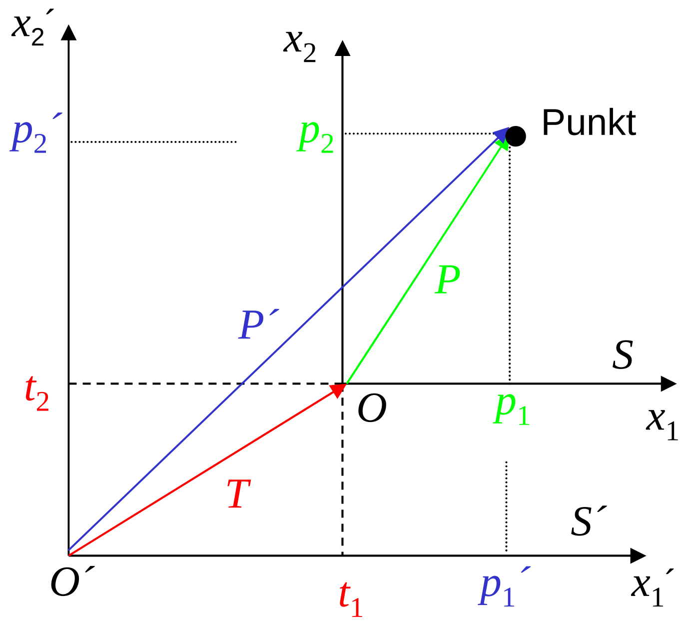
Gesucht:

- Transformation von S nach S' (Sicht b)) bzw.
- Änderung der Punktkoordinaten von System S nach S' (Sicht a)).

3.2 Transformationen in der Ebene

3.2.1 Translation (Verschiebung)

Voraussetzung: Koordinatenachsen sind zueinander jeweils parallel.



Sicht b): Es gilt für das System $S´$.

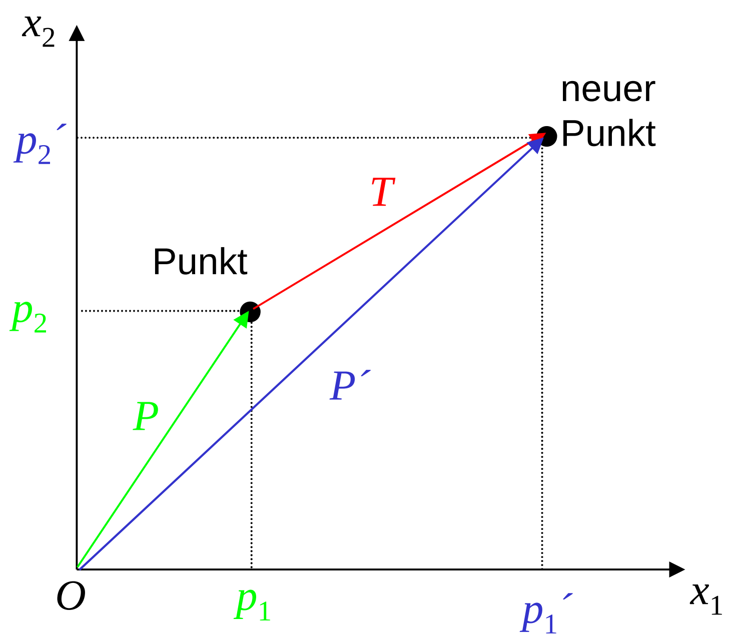
- $S´$ ergibt sich aus S durch Verschiebung um $-T$, wobei $T=(t_1, t_2)^T$ die Koordinaten des Ursprungs O von S im System $S´$ sind.
- Der Punkt hat also
  - in S die Koordinaten $P = (p_1,p_2)^T$,
  - in $S´$ die Koordinaten $P´ = T + P = (t_1,t_2)^T + (p_1,p_2)^T$.

3.2 Transformationen in der Ebene

Sicht a): Es gilt für den Punkt.

- Punkt mit Koordinaten P wird um T verschoben

  und es entsteht ein neuer Punkt mit Koordinaten $P´$ mit $P´=T+P$.



- Für beide Sichtweisen gilt in Vektor-Schreibweise:

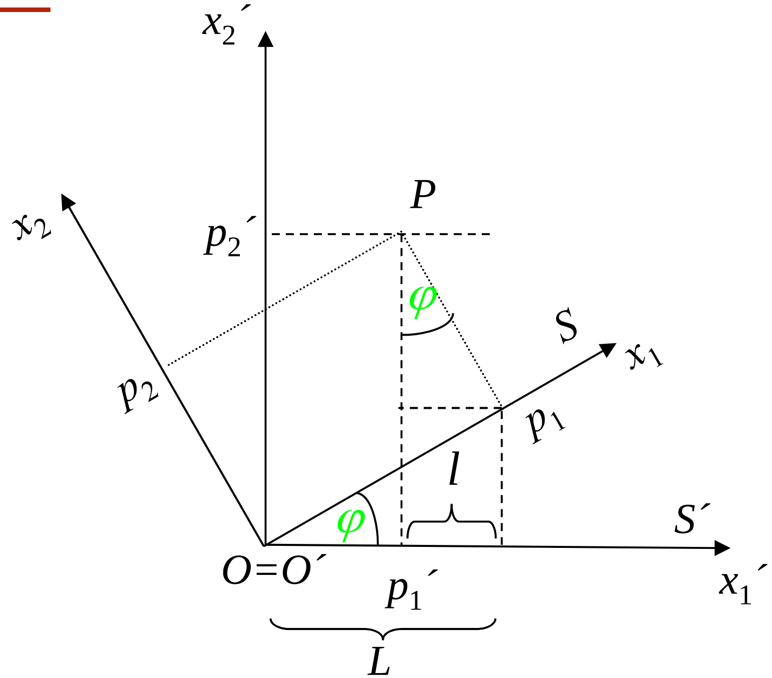
$$\begin{pmatrix} p_1' \\ p_2' \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} + \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$

- Oder kurz: $P´=T+P$.

3.2 Transformationen in der Ebene

3.2.2 Rotation (Drehung)

Voraussetzung: Koordinatensysteme haben gleichen Ursprung $O=O'$.



Sicht b): Drehung des Systems S gegenüber S' um den Winkel φ um O:

- Das System S' ergibt sich aus S durch Drehung um $-\varphi$.
- Es gilt: $l/p_2 = \sin\varphi$ und $L/p_1 = \cos\varphi$

also $p_1' = L - l = p_1 \cdot \cos\varphi - p_2 \cdot \sin\varphi$

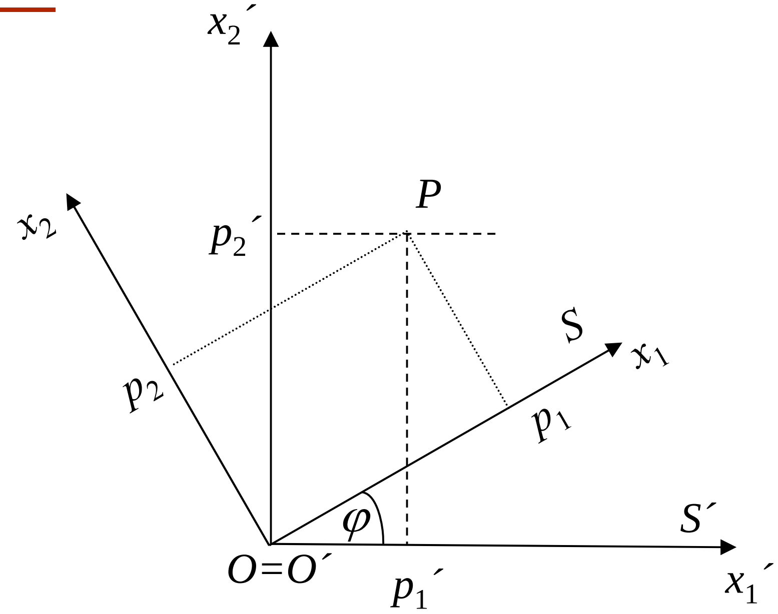
analog $p_2' = p_1 \cdot \sin\varphi + p_2 \cdot \cos\varphi$

3.2 Transformationen in der Ebene

- Kurz:

$$P' = \begin{pmatrix} p_1' \\ p_2' \end{pmatrix} = \begin{pmatrix} p_1 \cdot \cos\varphi - p_2 \cdot \sin\varphi \\ p_1 \cdot \sin\varphi + p_2 \cdot \cos\varphi \end{pmatrix}$$

$$= \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$



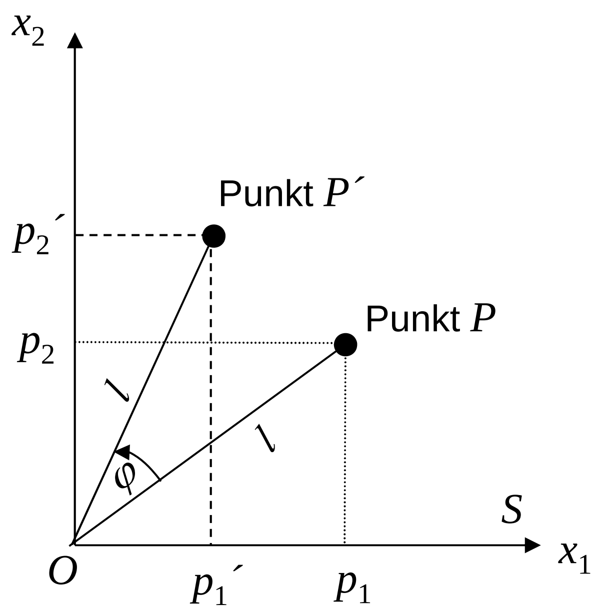
- In Vektor-Matrix-Schreibweise: $P' = R(\varphi) \cdot P$,
  mit der (orthonormalen) Rotationsmatrix: $R(\varphi) = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$

  Bem.: orthonormal gdw. $R^{-1} = R^T$.

3.2 Transformationen in der Ebene

Sicht a)

Die Matrix $R(\varphi)$ dreht Punkte um den Winkel φ in positiver Richtung um den Ursprung O eines festen Koordinatensystem.



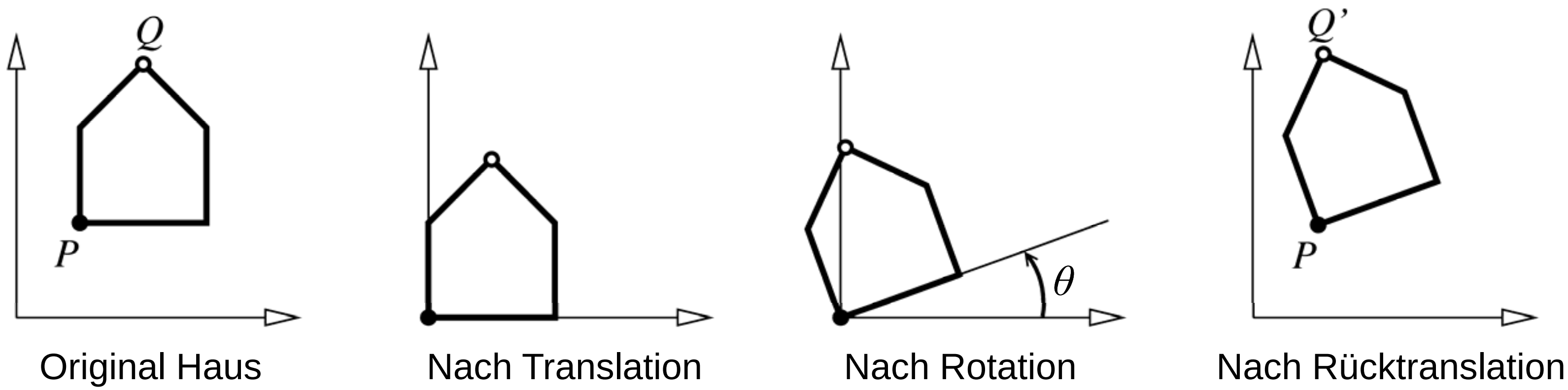
3.2 Transformationen in der Ebene

Rotation um einen beliebigen Punkt P:

(1) Translation von P in den Ursprung,

(2) Rotation um den Ursprung,

(3) Translation von P in die ursprüngliche Position.



Bemerkung: Die Matrizenmultiplikation ist nicht kommutativ, d.h. die Reihenfolge der Matrizen muss der Reihenfolge der Rotationen entsprechen.

3.2 Transformationen in der Ebene

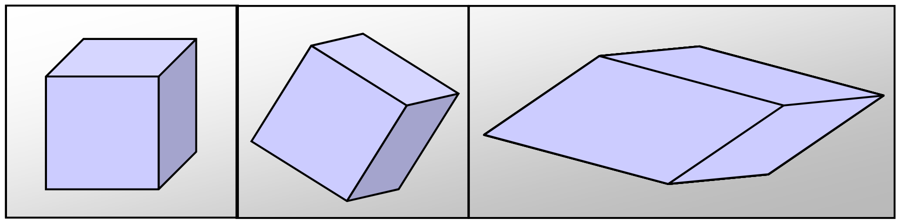
3.2.3 Skalierung (Scaling, Größenänderung)

- Soll das System S „vergrößert“ oder „verkleinert“ werden, so muß eine Skalierung durchgeführt werden:

$$p_1' = \lambda_1 \cdot p_1,$$
$$p_2' = \lambda_2 \cdot p_2.$$

In Vektor-Matrix-Schreibweise: $P' = S \cdot P$ mit $S = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}$

- **Vorsicht!** Skalierungen können Längen und Winkel ändern, d.h. Orthogonalität, Orthonormalität, Rechtssystem werden zerstören!



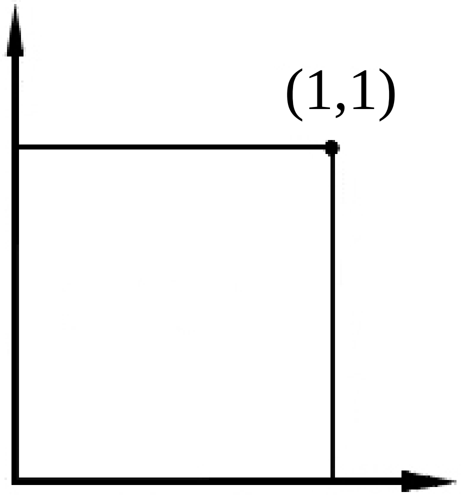
3.2 Transformationen in der Ebene

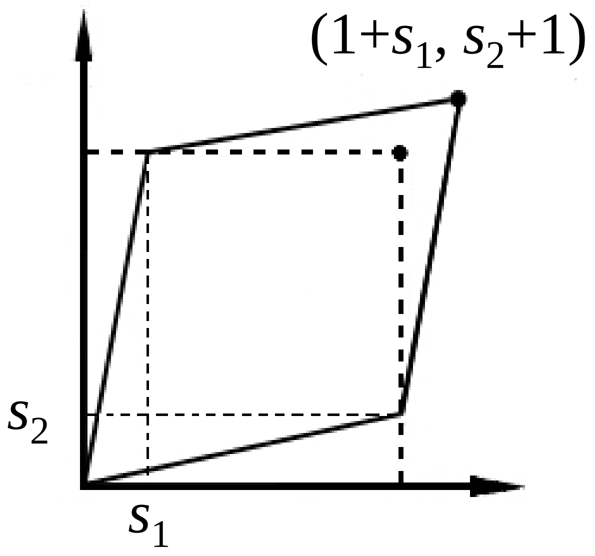
3.2.4 Scherung (Shear)

- Eine Scherung ergibt sich, wenn Abhängigkeiten folgender Form bestehen:

$$p_1' = p_1 + s_1 \cdot p_2,$$
$$p_2' = s_2 \cdot p_1 + p_2.$$

- In Vektor-Matrix-Schreibweise: $P' = S \cdot P$ mit $S = \begin{pmatrix} 1 & s_1 \\ s_2 & 1 \end{pmatrix}$





3.2 Transformationen in der Ebene

3.2.5 Affine Transformationen

- Affine Transformationen lassen sich als Kombination einer linearen Abbildung A und einer Translation T schreiben:

$$P' = A \cdot P + T.$$

- Die bisher genannten Transformationen (Translation, Rotation, Skalierung, Scherung) sind affine Transformationen.

3.2 Transformationen in der Ebene

Affine Invarianz von Teilungsverhältnissen:

Für eine affine Transformation F und die Punkte P und Q gilt immer:

$$F(\lambda P+(1-\lambda)Q) = \lambda F(P)+(1-\lambda) F(Q) \quad \text{für} \quad 0 \leq \lambda \leq 1.$$

bzw. allgemein für Punkte P_i

$$F\left(\sum_{i=0}^{n} \alpha_i P_i\right) = \sum_{i=0}^{n} \alpha_i F(P_i) \quad \text{für} \quad \sum_{i=0}^{n} \alpha_i = 1.$$

3.2 Transformationen in der Ebene

Eigenschaften affiner Abbildungen:

- Das Bild einer Strecke von Q nach P unter einer affinen Abbildung F ist wieder eine Strecke.
- Eine affine Abbildung F ändert Teilverhältnisse $\lambda : (1 - \lambda)$ nicht.
- Es genügt Endpunkte Q und P einer Strecke abzubilden; Zwischenpunkte erhält man durch Interpolation von $F(Q)$ und $F(P)$.
- Unter affinen Abbildungen bleiben parallele Linien parallel.

3.2 Transformationen in der Ebene

Weitere affine Transformationen:

- Reflexion an der Gerade $y=x$: $A=\begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$
- Reflexion an der Gerade $y=-x$: $A=\begin{pmatrix}0 & -1\\ -1 & 0\end{pmatrix}$
- Reflexion an der x-Achse: $A=\begin{pmatrix}-1 & 0\\ 0 & 1\end{pmatrix}$
- Reflexion an der y-Achse: $A=\begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$
- Reflexion am Ursprung: $A=\begin{pmatrix}-1 & 0\\ 0 & -1\end{pmatrix}$

3.2 Transformationen in der Ebene

3.2.6 Homogene Koordinaten

- Die Hintereinanderschaltung von Rotation, Translation und Skalierung führt auf die Abbildungsgleichung

$$P' = S \cdot (T + R \cdot P).$$

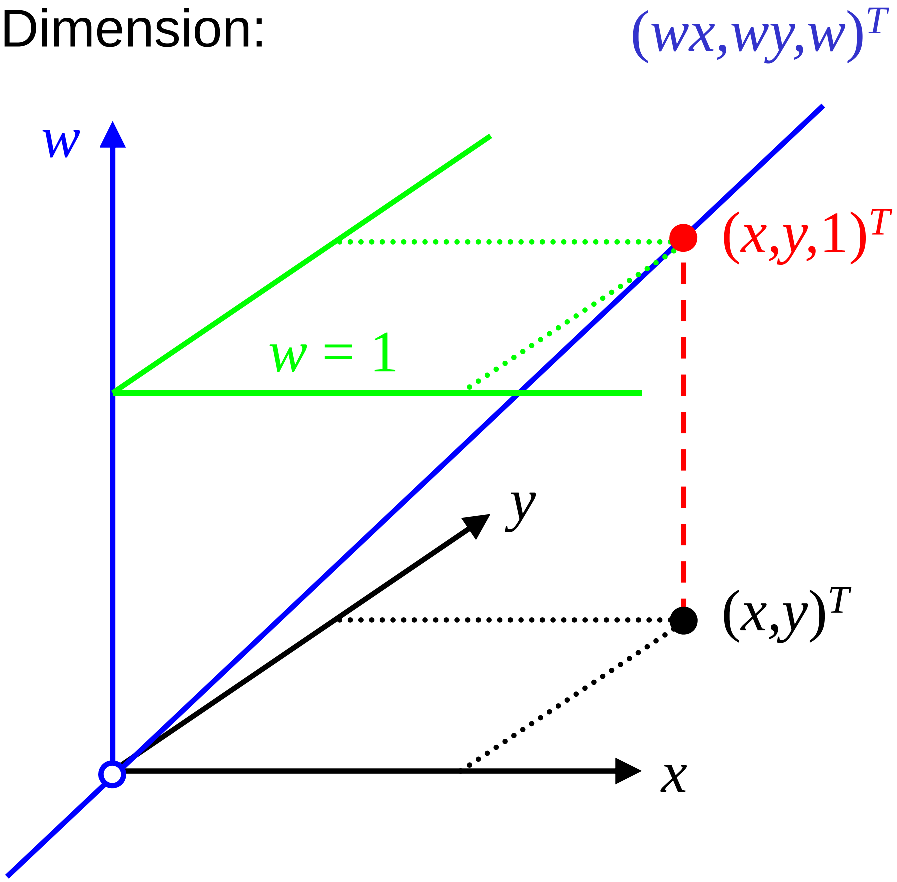
- Müssen jedoch mehrere solcher Transformationen hintereinander ausgeführt werden, so stört die Addition in der obigen Gleichung.
- Da heutige Grafik-Hardware insbesondere auch Matrixmultiplikationen unterstützt, ist es günstig Transformationen ausschließlich mittels Matrixmultiplikationen auszuführen

$$P' = M_n \cdots M_3 \cdot M_2 \cdot M_1 \cdot P.$$

3.2 Transformationen in der Ebene

Übergang auf die nächst höhere Dimension:

- Das Tripel $(wx, wy, w)^T$, $w \neq 0$, stellt die homogenen Koordinaten des Punktes $(x, y)^T \in \mathbb{R}^2$ dar.
- Es gibt unendlich viele solcher Darstellungen desselben Punktes.
- Verwende die so genannte Standarddarstellung $w = 1$.
- Also besitzt ein Punkt $P = (x, y)^T \in \mathbb{R}^2$ die homogenen Koordinaten $(x, y, 1)^T$.



- Für Punkt im $\mathbb{R}^3$ gilt Analoges.

3.2 Transformationen in der Ebene

Darstellung affiner Transformationen in homogenen Koordinaten:

- **Translation** des Punktes $(x, y)^T$ um den Vektor $(t_1,t_2)^T$:

$$\begin{pmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x+t_1 \\ y+t_2 \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

- **Rotation** des Punktes $(x, y)^T$ um den Winkel φ um den Ursprung:

$$\begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x\cdot\cos\varphi - y\cdot\sin\varphi \\ x\cdot\sin\varphi + y\cdot\cos\varphi \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

3.2 Transformationen in der Ebene

- **Skalierung** des Punktes $(x, y)^T$ mit den Faktoren λ_1 und λ_2:

$$\begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda_1 \cdot x \\ \lambda_2 \cdot y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

- **Rotation** des Punktes $(x, y)^T$ **um einen Punkt** $P = (p_x, p_y)^T$ um den Winkel φ:

$$\begin{pmatrix} 1 & 0 & p_x \\ 0 & 1 & p_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\varphi & -\sin\varphi & 0 \\ \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -p_x \\ 0 & 1 & -p_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$

3.3 Transformationen im Raum

3.3.1 Translation

Die Verschiebung eines Punktes $(x, y, z)^T$ um den Translationsvektor $(t_x, t_y, t_z)^T$ ergibt den Punkt $(x´, y´, z´)^T$ mit

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{=T(t_x,t_y,t_z)} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+t_x \\ y+t_y \\ z+t_z \\ 1 \end{pmatrix} = \begin{pmatrix} x´ \\ y´ \\ z´ \\ 1 \end{pmatrix}$$

3.3 Transformationen im Raum

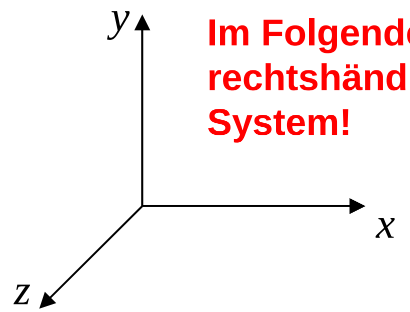
3.3.2 Skalierung

Eine Skalierung mit den Faktoren λ_1, λ_2 und λ_3 in den drei Achsenrichtungen hat die Form:

$$\begin{pmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \lambda_1 \cdot x \\ \lambda_2 \cdot y \\ \lambda_3 \cdot z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

3.3 Transformationen im Raum

3.3.3 Rotation



Im Folgenden: rechtshändiges System!

- Alle Rotationen erfolgen im mathematisch positiven Sinn (d.h. gegen den Uhrzeigersinn).
- Der Betrachter „sitzt" dabei auf der Rotationsachse und schaut in Richtung Ursprung des Koordinatensystems.
- Wir betrachten zuerst die Rotationen um die einzelnen Koordinatenachsen um den Winkel φ.

➡ Transformationsmatrizen $R_x(\varphi), R_y(\varphi), R_z(\varphi)$.

- Wir verwenden die Sicht a):

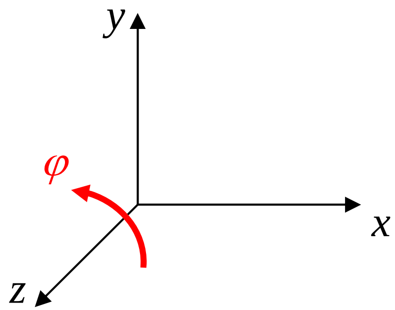
„globales festes Koordinatensystem;

Punkt wird transformiert (gedreht)"

3.3 Transformationen im Raum

Rotation um die z-Achse

Die Rotation eines Punktes $(x, y, z)^T$ um die z-Achse um den Winkel φ ergibt den Punkt $(x', y', z')^T$ mit



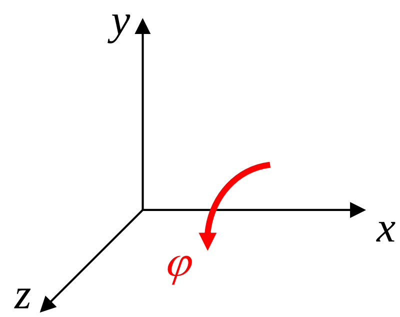
$$\underbrace{\begin{pmatrix} \cos\varphi & -\sin\varphi & 0 & 0 \\ \sin\varphi & \cos\varphi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{=R_z(\varphi)} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \cdot \cos\varphi - y \cdot \sin\varphi \\ x \cdot \sin\varphi + y \cdot \cos\varphi \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

Beachte: Drehung um den Winkel φ um die z-Achse entspricht dem 2D-Fall, wobei die z-Koordinate konstant bleibt!

3.3 Transformationen im Raum

Rotation um die x-Achse

Die Rotation eines Punktes $(x, y, z)^T$ um die x-Achse um den Winkel φ ergibt den Punkt $(x´, y´, z´)^T$ mit



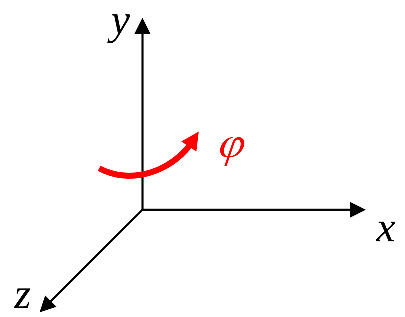
$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\varphi & -\sin\varphi & 0 \\ 0 & \sin\varphi & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{=R_x(\varphi)} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y\cdot\cos\varphi - z\cdot\sin\varphi \\ y\cdot\sin\varphi + z\cdot\cos\varphi \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

Beachte: Drehung um den Winkel φ um die x-Achse entspricht dem 2D-Fall, wobei die x-Koordinate konstant bleibt!

3.3 Transformationen im Raum

Rotation um die y-Achse

Die Rotation eines Punktes $(x, y, z)^T$ um die y-Achse um den Winkel φ ergibt den Punkt $(x', y', z')^T$ mit



$$\underbrace{\begin{pmatrix} \cos\varphi & 0 & \sin\varphi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\varphi & 0 & \cos\varphi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{=R_y(\varphi)} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x\cdot\cos\varphi + z\cdot\sin\varphi \\ y \\ -x\cdot\sin\varphi + z\cdot\cos\varphi \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$

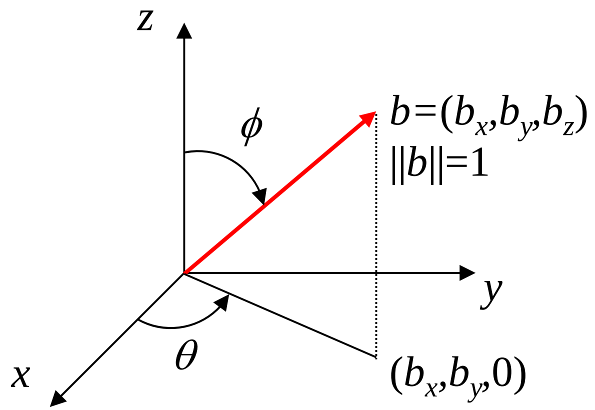
Beachte: Drehung um den Winkel φ um die y-Achse entspricht dem 2D-Fall, wobei die y-Koordinate konstant bleibt!

3.3 Transformationen im Raum

Bem.: Immer noch rechtshändiges System, hier nur gedreht.

Rotation um eine beliebige Achse

- Jede Rotation um eine beliebige Achse kann aus **drei** Rotationen um die einzelnen Koordinatenachsen zusammengesetzt werden ($\rightarrow$ Euler).
- **Ziel**: Rotation $R_G(\alpha)$ eines Punktes P um eine beliebig orientierte Achse G im Raum um einen Winkel α.



$$b_x = \sin\phi \ \cos\theta$$
$$b_y = \sin\phi \ \sin\theta$$
$$b_z = \cos\phi$$

- **Zunächst Sonderfall:** Drehachse G geht durch den Ursprung und wird von einem Vektor $b=(b_x,b_y,b_z)^T$ mit $||b||=1$ generiert:

$$G=\{ \lambda \cdot b : \lambda \in \mathbb{R}\}.$$

3.3 Transformationen im Raum

Gesucht: Koordinaten eines Punktes P nach einer Drehung um die Achse G um den Winkel α.

- Vorgehensweise:
  - Der Punkt P wird so transformiert, dass die Drehachse mit der z-Achse zusammenfällt.
  - Anschließend wird für die Drehung um α die Rotationsmatrix $R_z(\alpha)$ verwendet.
  - Hinterher werden die „Hilfstransformationen“ wieder rückgängig gemacht.
  - Ist G mit der z-Achse identisch, entfallen die Hilfstransformationen.

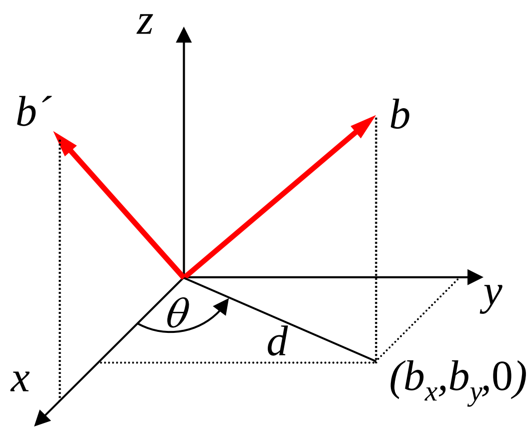
3.3 Transformationen im Raum

Schritt 1:

- Drehe den Vektor b in die (z,x)-Ebene: $b \longrightarrow b'$.
- P wird auf P' mit $P' = R_z(-\theta)\, P$ abgebildet, mit

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

$$= \frac{1}{d} \begin{pmatrix} b_x & b_y & 0 & 0 \\ -b_y & b_x & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & d \end{pmatrix}$$



$$d^2 = b_x^2 + b_y^2$$

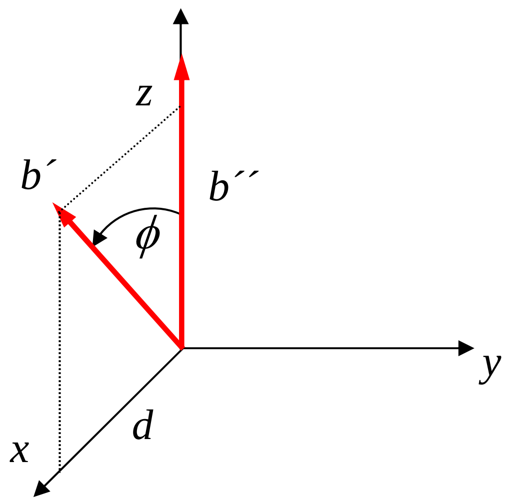
3.3 Transformationen im Raum

Schritt 2:

- Drehe den Vektor $b´$ auf die z-Achse: $b´ \longrightarrow b´´$.
- $P´$ wird auf $P´´$ mit $P´´ = R_y(-\phi)\, P´$ abgebildet, mit

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

$$= \begin{pmatrix} b_z & 0 & -d & 0 \\ 0 & 1 & 0 & 0 \\ d & 0 & b_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

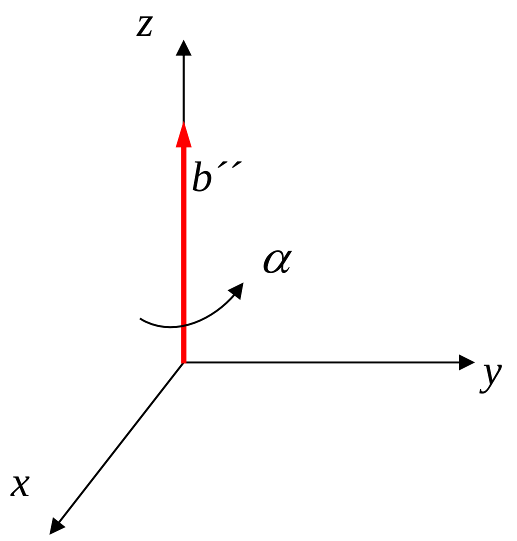


3.3 Transformationen im Raum

Schritt 3:

- Drehe um den Winkel α um die z-Achse.
- P'' wird auf P''' mit $P''' = R_z(\alpha)\, P''$ abgebildet, mit

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



3.3 Transformationen im Raum

Schritte 4 und 5:

- Die Rotationen aus den Schritten 1 und 2 werden in umgekehrter Reihenfolge rückgängig gemacht.
- P''' wird auf den gewünschten gedrehten Punkt Q des ursprünglichen Punktes P mittels $Q = R_z(\theta)\, R_y(\phi)\, P'''$ abgebildet, mit

$$R_y(\phi) = \begin{pmatrix} b_z & 0 & d & 0 \\ 0 & 1 & 0 & 0 \\ -d & 0 & b_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad R_z(\theta) = \frac{1}{d}\begin{pmatrix} b_x & -b_y & 0 & 0 \\ b_y & b_x & 0 & 0 \\ 0 & 0 & d & 0 \\ 0 & 0 & 0 & d \end{pmatrix}$$

3.3 Transformationen im Raum

Ergebnis:

Die Gesamttransformation läßt sich in einem Schritt durch die Verknüpfung aller Transformationen realisieren

$$M_b(\alpha) = R_z(\theta)\, R_y(\phi)\, R_z(\alpha)\, R_y(-\phi)\, R_z(-\theta).$$

Allgemeiner Fall:

Ist die Drehachse eine allgemeine Gerade

$$G = \{\, a + \lambda \cdot b : \ \lambda \in \mathbb{R},\ a = (a_x, a_y, a_z)^T,\ ||b|| = 1 \,\}\ ,$$

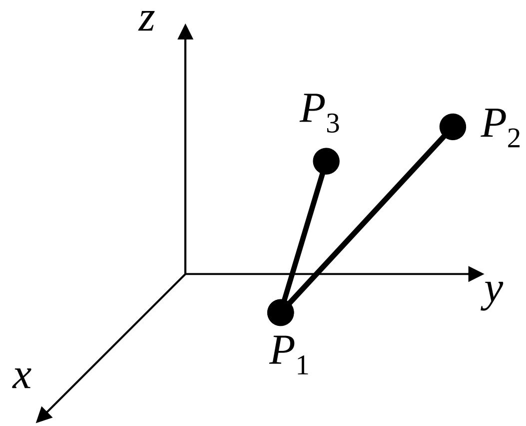
so ist vor Schritt 1 und nach Schritt 5 eine entsprechende Translation einzuschieben, also

$$M_b(\alpha) = T(a_x, a_y, a_z)\, R_z(\theta)\, R_y(\phi)\, R_z(\alpha)\, R_y(-\phi)\, R_z(-\theta)\, T(-a_x, -a_y, -a_z).$$

3.3 Transformationen im Raum

3.3.4 Transformation von Koordinatensystemen (Orientierung)

- Zur Definition eines Koordinatensystems im 3D genügen drei (nicht-kollineare) Punkte P_1, P_2, P_3.
- Wie berechnet man die Transformationsmatrix, die
  - P_1 in den Ursprung,
  - P_1P_2 auf die z-Achse und
  - P_2P_3 in die (y,z)-Ebene mit positiver y-Koordinate



abbildet?

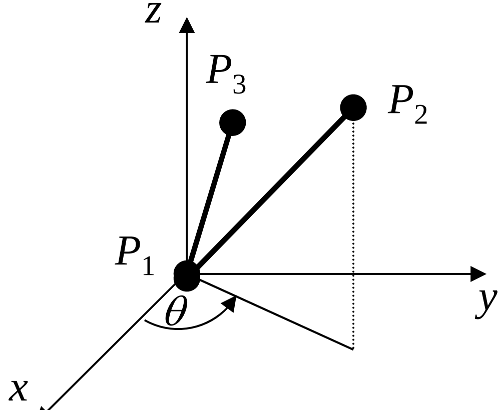
3.3 Transformationen im Raum

Schritt 0: Translation mit $T(-P_1)$.

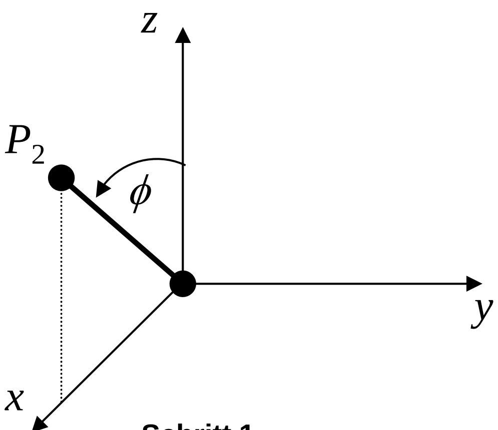
Schritt 1: Rotation mit $R_z(-\theta)$.

Schritt 2: Rotation mit $R_y(-\phi)$.

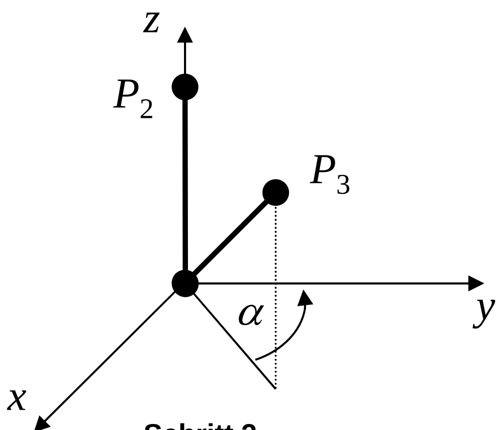
Schritt 3: Rotation mit $R_z(\alpha)$.



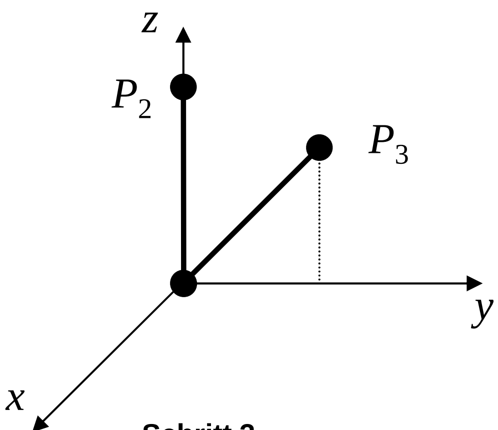
Schritt 0



Schritt 1



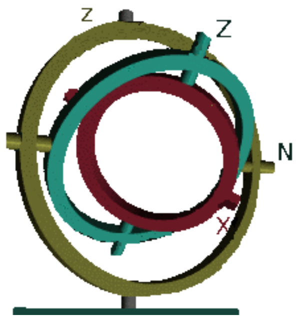
Schritt 2



Schritt 3

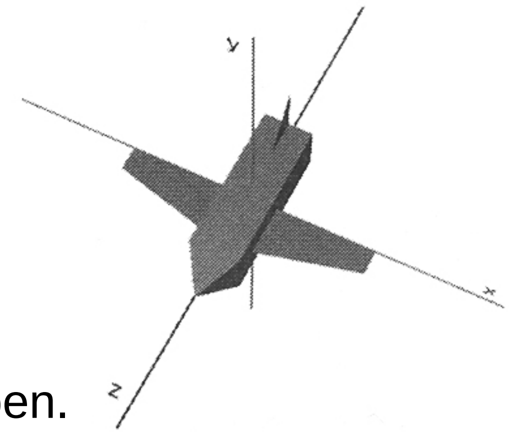
3.3 Transformationen im Raum

Achtung: Sind P_1P_2 und die z-Achse parallel, sind die Winkel θ und α nicht wohldefiniert.



wikipedia

- In diesem Fall ist nur $\theta + \alpha$ wohldefiniert.
- Dieser Effekt ist eine Ausprägung des sog. **Gimbal-Lock** (gimbal = Kardanaufhängung):
  - Er ist bei einer anderen Rotationsreihenfolge (pilotview) ausgeprägter:
    - x-Rotation um Neigungswinkel (pitch) φ_x,
    - y-Rotation um Gierungswinkel (yaw) φ_y,
    - z-Rotation um Rollwinkel (roll) φ_z.
  - Die Orientierung eines Koordinatensystem wird dann durch das Tripel $(\varphi_x, \varphi_y, \varphi_z)$ beschrieben.

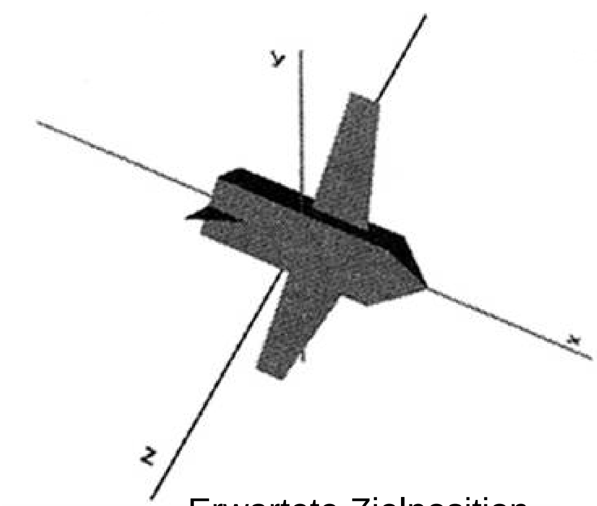


Bender/Brill

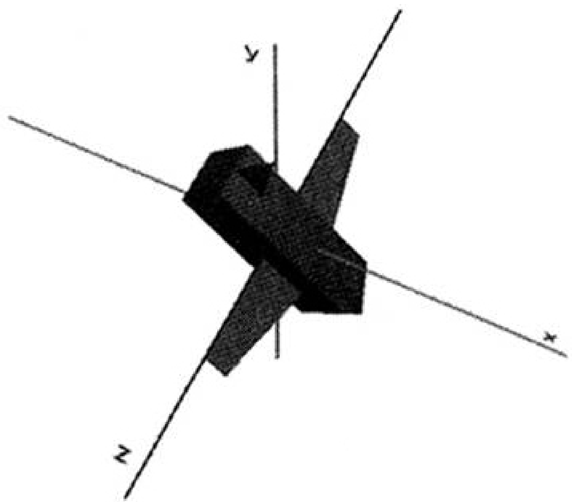
3.3 Transformationen im Raum

– Eine Änderung der Orientierung von $(0°, 0°, 0°)$ nach $(0°, 90°, 45°)$ liefert nicht das gewünschte Ergebnis:

  - Nach der Drehung um die y-Achse ist die dritte Rotationsachse auf die x-Achse gedreht.



Erwartete Zielposition.



Erreichte Zielposition.

Bender/Brill

– Ein Freiheitsgrad geht verloren!

3.3 Transformationen im Raum

Die zusammengesetzte Transformationsmatrix hat die Form

$$R_z(\alpha)\cdot R_y(-\phi)\cdot R_z(-\theta)\cdot T(-P_1)=\begin{pmatrix} R & -P_1 \\ 0 & 1 \end{pmatrix}$$

mit einer orthonormalen 3×3 Matrix R.

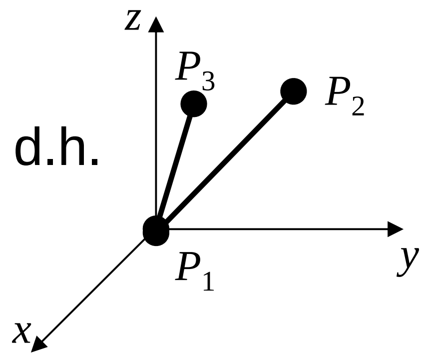
- Die Zeilen R_1, R_2, R_3 von R bilden eine Orthonormalbasis

$$R\cdot R_1{}^T=(1,0,0)^T,$$
$$R\cdot R_2{}^T=(0,1,0)^T,$$
$$R\cdot R_3{}^T=(0,0,1)^T,$$

d.h. sie werden durch R auf die Achsen des Koordinatensystems (x,y,z) abgebildet.

3.3 Transformationen im Raum

Daraus lassen sich die Zeilen von R bestimmen:



Zeile 3: P_1P_2 wird durch R auf die z-Achse abgebildet, d.h.

$$R_3^T = (P_2 - P_1) \,/\, \|P_2 - P_1\|.$$

Zeile 1: Der (normierte) Normalenvektor der Ebene $P_1P_2P_3$ wird durch R auf die positive x-Achse abgebildet, d.h.

$$R_1^T = (P_3 - P_1) \times (P_2 - P_1) \,/\, \|(P_3 - P_1) \times (P_2 - P_1)\|.$$

Zeile 2: Die zweite Zeile ist orthonormal zu R_1, R_3, d.h.

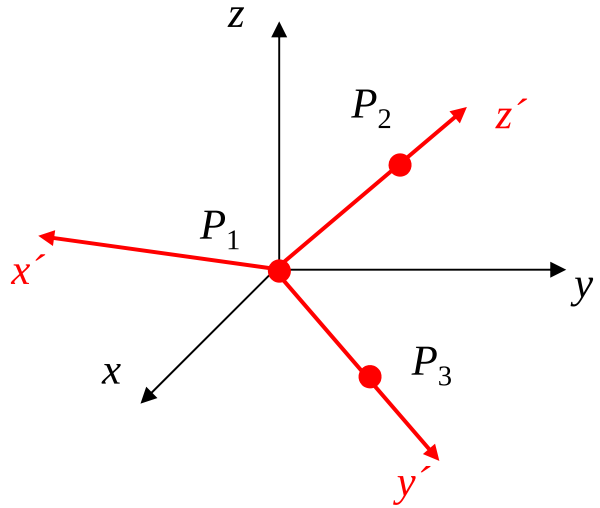
$$R_2^T = R_3^T \times R_1^T.$$

3.3 Transformationen im Raum

Sonderfall: Ist das von P_1, P_2, P_3 aufgespannte Koordinatensystem kartesisch, ist R gegeben durch

$$R = \begin{pmatrix} x'^T \\ y'^T \\ z'^T \end{pmatrix}$$

in Koordinaten bzgl. dem Koordinatensystem (x,y,z).



3.3 Transformationen im Raum

3.3.5 Quaternionen

- Vermeidung des Gimbal-Locks bei Euler-Winkeln.
- Interpolation von Rotationen bei der Animation.

Idee: Koordinatensystemfreie Beschreibung einer Rotation!

- Drehachse G geht durch den Ursprung und wird von einem Vektor $\boldsymbol{b}=(b_x,b_y,b_z)^T$ mit $||\boldsymbol{b}||=1$ generiert:

$$G=\{\ \lambda\cdot\boldsymbol{b} : \lambda\in\mathbb{R}\}.$$

- Drehe einen Punkt P um die orientierte Achse G im Raum um einen Winkel α.

3.3 Transformationen im Raum

- Wähle Ebene E durch P senkrecht zu $\boldsymbol{b}$.

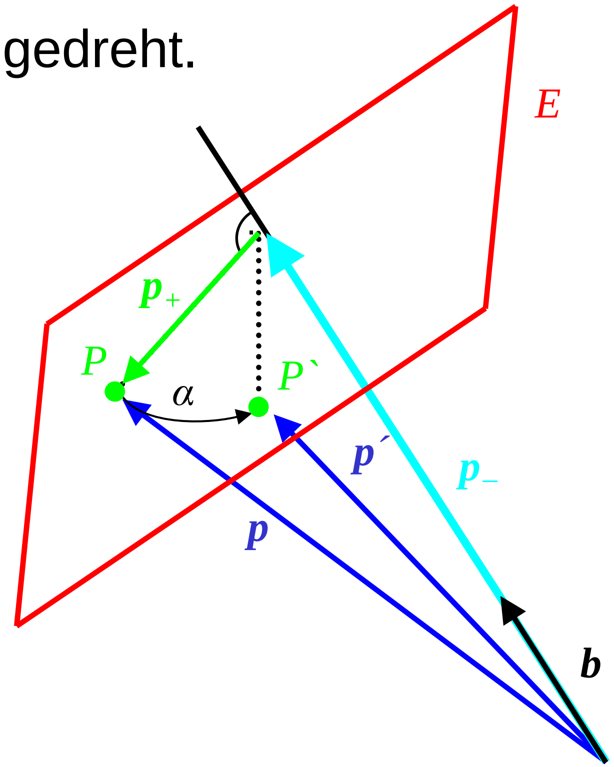
➡ P wird in der Ebene E auf P' gedreht.

➡ Der Vektor $\boldsymbol{p}$ wird auf den Vektor $\boldsymbol{p}'$ gedreht.

➡ Diese Rotation hat eine Anteil

- $\boldsymbol{p}_{-} = \boldsymbol{b}\,(\boldsymbol{p}\,\boldsymbol{b})$ parallel und
- $\boldsymbol{p}_{+} = \boldsymbol{p} - \boldsymbol{b}\,(\boldsymbol{p}\,\boldsymbol{b})$ senkrecht

zur Ebenennormale $\boldsymbol{b}$.



3.3 Transformationen im Raum

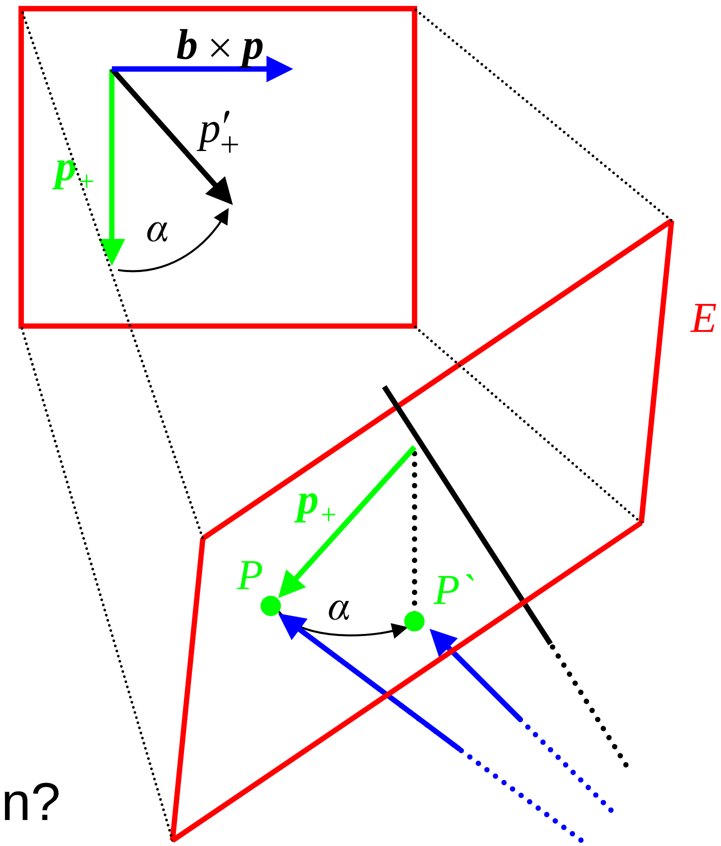
- In der Ebene E wird $\boldsymbol{p}_+$ auf

  $$\boldsymbol{p}'_+ = \cos\alpha \cdot \boldsymbol{p}_+ + \sin\alpha \cdot \boldsymbol{b} \times \boldsymbol{p}$$

  gedreht.

- Die gesamte Rotation wird also beschrieben durch

  $$\begin{aligned} \boldsymbol{p}' = & \quad\cos\alpha \cdot \boldsymbol{p} \\ & + (1 - \cos\alpha) \cdot \boldsymbol{b} \cdot (\boldsymbol{b} \cdot \boldsymbol{p}) \\ & + \sin\alpha \cdot \boldsymbol{b} \times \boldsymbol{p}. \end{aligned}$$



- Wie lässt sich das elegant formulieren?

3.3 Transformationen im Raum

... mit Quaternionen:

Definition

Ein **Quaternion** q ist gegeben durch vier reelle Zahlen (s,a,b,c) und

$$q = s + a \cdot \boldsymbol{i} + b \cdot \boldsymbol{j} + c \cdot \boldsymbol{k} \quad \text{mit} \quad \boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{k}^2 = -1.$$

Für die **imaginären Einheiten** $\boldsymbol{i}, \boldsymbol{j}, \boldsymbol{k}$ gilt darüber hinaus

$$\boldsymbol{i} \cdot \boldsymbol{j} = -\boldsymbol{j} \cdot \boldsymbol{i} = \boldsymbol{k}; \quad \boldsymbol{j} \cdot \boldsymbol{k} = -\boldsymbol{k} \cdot \boldsymbol{j} = \boldsymbol{i}; \quad \boldsymbol{k} \cdot \boldsymbol{i} = -\boldsymbol{i} \cdot \boldsymbol{k} = \boldsymbol{j}.$$

3.3 Transformationen im Raum

Addition und Multiplikation sind definiert als

$$(s_1,a_1,b_1,c_1) + (s_2,a_2,b_2,c_2) = (s_1+ s_2,\ a_1+ a_2,\ b_1+ b_2,\ c_1+ c_2)$$

$$= s_1+s_2+(a_1+a_2)\boldsymbol{i} +(b_1+b_2)\boldsymbol{j} +(c_1+c_2)\boldsymbol{k},$$

$$\begin{aligned}(s_1,a_1,b_1,c_1) \cdot (s_2,a_2,b_2,c_2) = \quad & s_1s_2 - a_1a_2 - b_1b_2 - c_1c_2 \\ & + (s_1a_2 + s_2a_1 +b_1c_2 - b_2c_1)\,\boldsymbol{i} \\ & + (s_1b_2 + s_2b_1 +c_1a_2 - c_2a_1)\,\boldsymbol{j} \\ & + (s_1c_2 + s_2c_1 +a_1b_2 - a_2b_1)\,\boldsymbol{k}\end{aligned}$$

Ein Quaternion q kann auch geschrieben werden als $q = (s,\boldsymbol{x})$ mit $s \in \mathbb{R}$ und $\boldsymbol{x} \in \mathbb{R}^3$. Dann gilt:

$$(s_1,\boldsymbol{x}_1) \cdot (s_2,\boldsymbol{x}_2) = (s_1s_2 - \boldsymbol{x}_1 \cdot \boldsymbol{x}_2,\ s_1\,\boldsymbol{x}_2 + s_2\,\boldsymbol{x}_1 + \boldsymbol{x}_1 \times \boldsymbol{x}_2).$$

3.3 Transformationen im Raum

- Auf Grund der Multiplikation für Quaternionen gibt es auch eine Norm für Quaternionen:

$$(q \cdot q')^{1/2} = \| q \| = (s^2 + \| \boldsymbol{x} \|^2)^{1/2} \quad \text{mit} \quad q' = (s, -\boldsymbol{x}).$$

  Dabei heißt q' das **konjugierte Quaternion**.

→ Für ein **normiertes Quaternion** q, d.h. $\| q \| = 1$, ist das konjugierte Quaternion das Inverse q^{-1}, d.h. $q \cdot q' = (1,0,0,0)$.

Theorem

Jedes normierte Quaternion q kann geschrieben werden als

$$q = (\cos(\theta/2), \sin(\theta/2)\, \boldsymbol{n})$$

für ein $\theta \in [0,\pi]$ und $\boldsymbol{n} \in \mathbb{R}^3$ mit $\|\boldsymbol{n}\|=1$.

3.3 Transformationen im Raum

- Identifiziert man den Vektor $\boldsymbol{p}$ von Seite §3-54 mit dem Quaternion $p = (0, \boldsymbol{p})$, liefert Multiplikation mit einem normierten Quaternion q von links und dem konjugierten Quaternion q' von rechts:

$$q \cdot p \cdot q' = (\cos(\theta/2), \sin(\theta/2)\, \boldsymbol{n}) \cdot (0, \boldsymbol{p}) \cdot (\cos(\theta/2), -\sin(\theta/2)\, \boldsymbol{n})$$
$$= (0, \cos\theta \cdot \boldsymbol{p} + (1-\cos\theta) \cdot \boldsymbol{n} \cdot (\boldsymbol{n} \cdot \boldsymbol{p}) + \sin\theta \cdot \boldsymbol{n} \times \boldsymbol{p}).$$

→ Das ist exakt die selbe Formel wie auf Seite §3-55 für $\boldsymbol{p}'$ mit $\boldsymbol{b}=\boldsymbol{n}$ und $\alpha = \theta$.

→ Die Multiplikation eines Quaternions mit einem normierten Quaternion q von links und mit dem konjugierten Quaternion q' von rechts ist eine Rotation um die Achse $\boldsymbol{n}$ mit den Winkel θ.

3.3 Transformationen im Raum

Rotation mit Quaternionen von einem Vektor $\boldsymbol{p}$ um die Achse $\boldsymbol{n}$ mit dem Winkel θ:

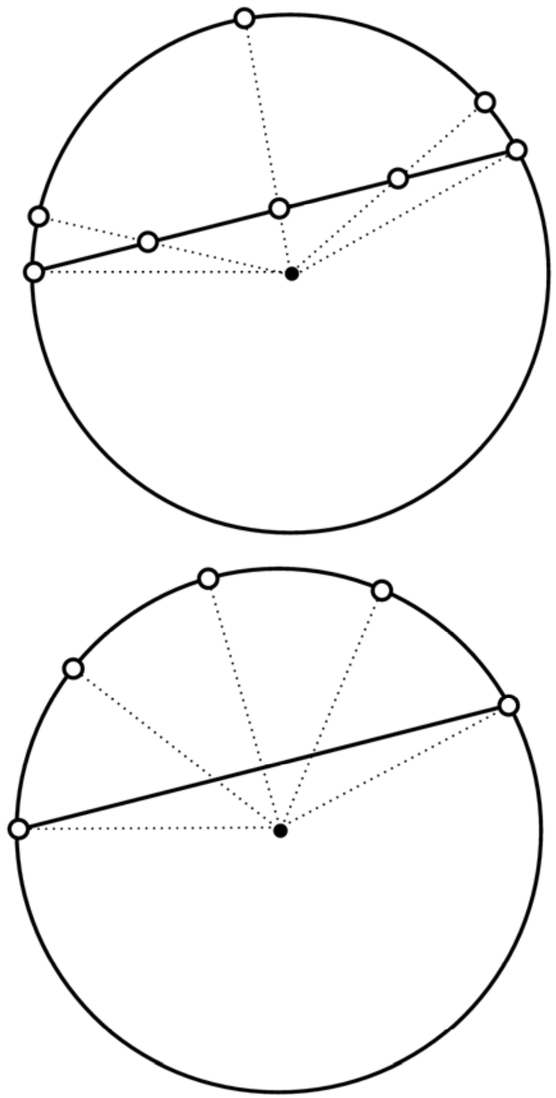
1. Bilde das Quaternion $q = (\cos(\theta/2), \sin(\theta/2)\, \boldsymbol{n})$.
2. Berechne $q \cdot (0, \boldsymbol{p}) \cdot q'$.
3. Transformiere das Ergebnis zurück zu einem Vektor im $\mathbb{R}^3$.

- Weil das Produkt zweier normierte Quaternionen wieder ein normiertes Quaternion ist, können mehrere Rotationen um verschiedene Achsen durch Multiplikation hintereinander ausgeführt werden.
- **Achtung:** Die Multiplikation ist nicht kommutativ!

3.3 Transformationen im Raum

Eigenschaften:

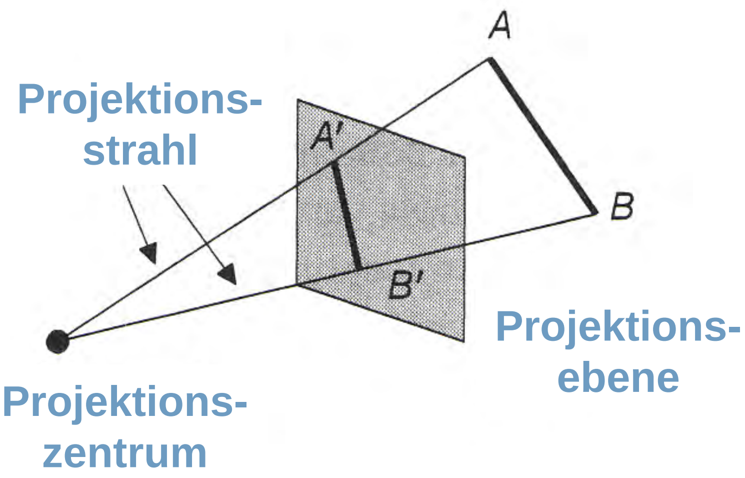
- Die Multiplikation ist nicht kommutativ!
- Es gibt keinen Gimbal-Lock, weil die Reihenfolge von Koordinatenachsen unerheblich ist.
- Interpolation:
  - Lineare Interpolation liefert variierende Winkelgeschwindigkeiten.
  - Spherical LinEaR interPolation (SLERP) [Shoemeker85].



3.4 Projektionen

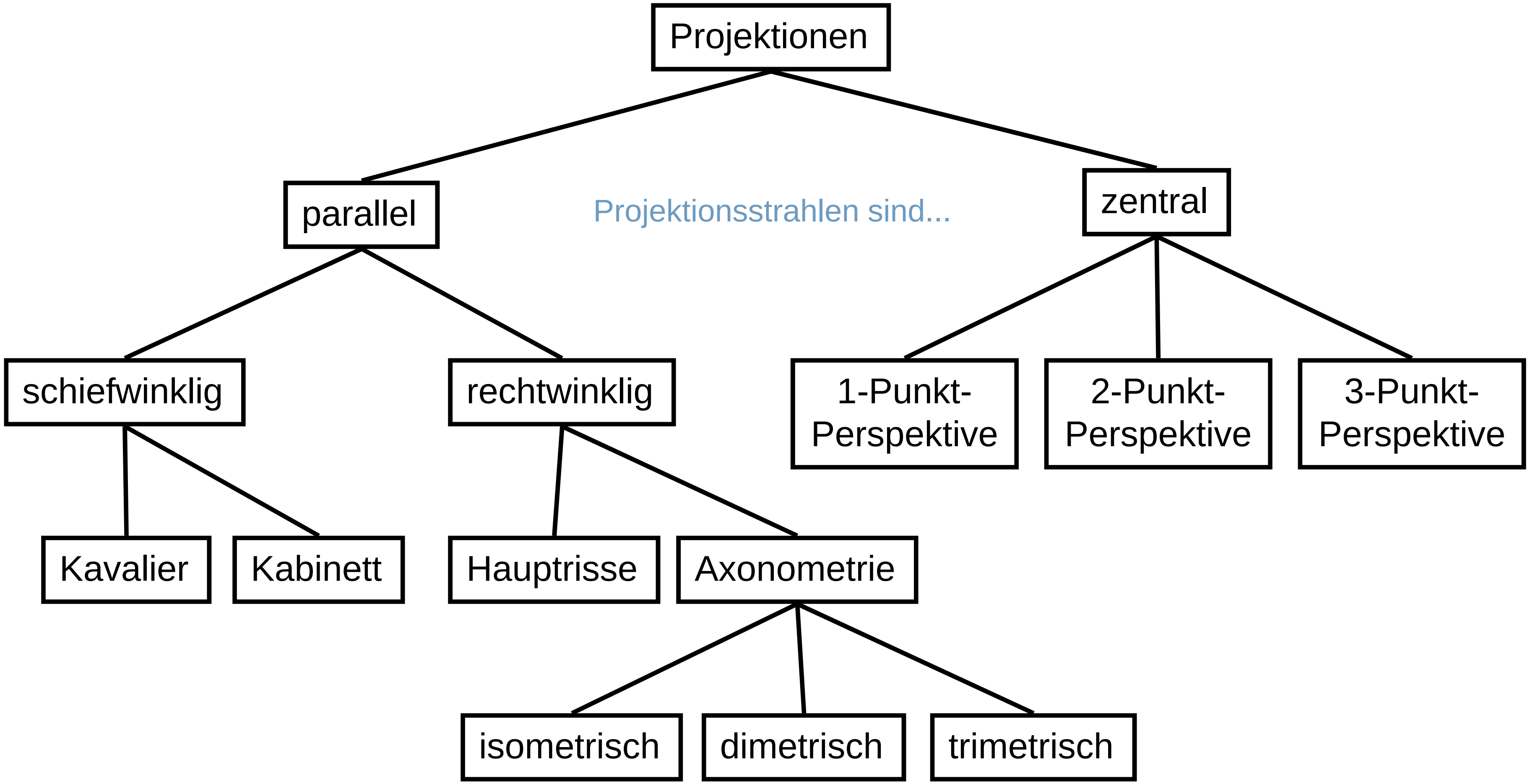
Definition: Eine **Projektion** ist eine Abbildung, die einen Raum der Dimension n auf einen Raum mit einer Dimension $< n$ abbildet.

- Da ein Bildschirm ein zweidimensionales Ausgabemedium ist, müssen dreidimensionale Objekte in zweidimensionalen Ansichten dargestellt werden.
  - Hierzu wird ein Raumpunkt entlang eines **Projektionsstrahls** (projector) auf eine vorgegebene **Projektionsebene** (projection plane) abgebildet.
- Der Projektionsstrahl wird durch das **Projektionszentrum** und den Raumpunkt A festgelegt.
- Der Schnittpunkt des Projektionsstrahls mit der Projektionsebene bestimmt den **projizierten Raumpunkt** $A´$.



3.4 Projektionen

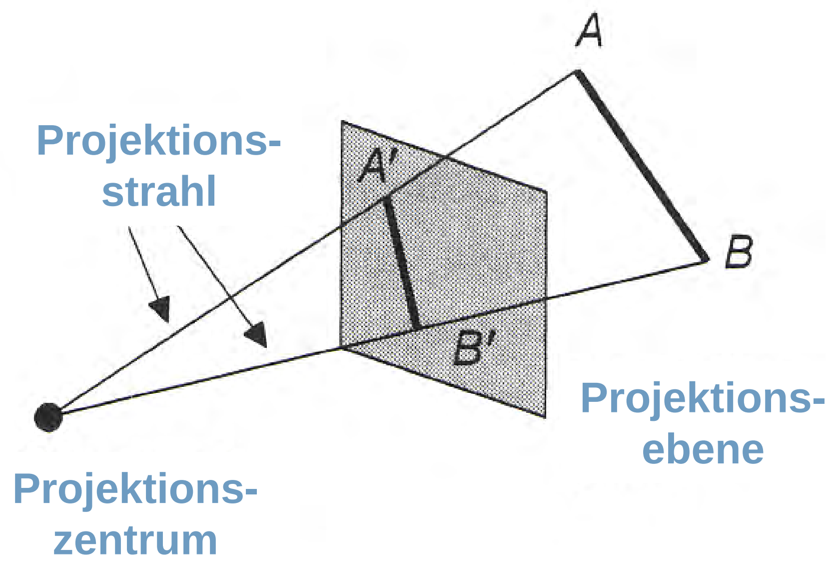
Klassifikation von Projektionen



3.4 Projektionen

3.5.1 Perspektivische Projektionen / Zentralprojektion

- Alle **Projektionsstrahlen** gehen durch das **Projektionszentrum**, das mit dem Auge des Beobachters zusammenfällt.
- Das Verfahren erzeugt eine optische Tiefenwirkung und geht in seinen Anfängen bis in die Malerei der Antike zurück.

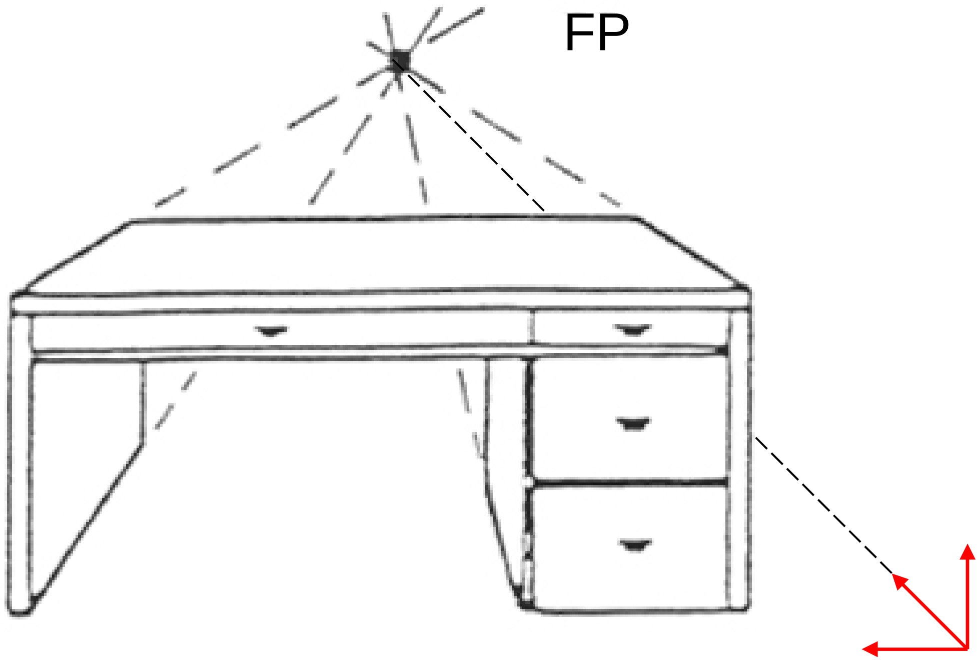


3.4 Projektionen

- Eigenschaften:
  - Die Bildgeraden je zwei parallele Geraden im Raum, die nicht parallel zur Projektionsebene sind, treffen sich in einem Punkt, dem Fluchtpunkt.
    - Diese Geraden schneiden die Projektionsebene!
  - Es gibt unendlich viele Fluchtpunkte, je einen pro Richtung, die nicht parallel zur Projektionsebene ist.
  - Hervorgehoben werden die Fluchtpunkte der Hauptachsen.
    - Z.B. Geraden, die parallel zur x-Achse (des Weltsystems) verlaufen, treffen sich im x-Fluchtpunkt, etc.
- Perspektivische Projektionen werden nach der Anzahl der Hauptachsen, die von der Projektionsebene geschnitten werden, klassifiziert.
  - So entstehen 1-Punkt-, 2-Punkt- und 3-Punkt-Perspektiven.

3.4 Projektionen

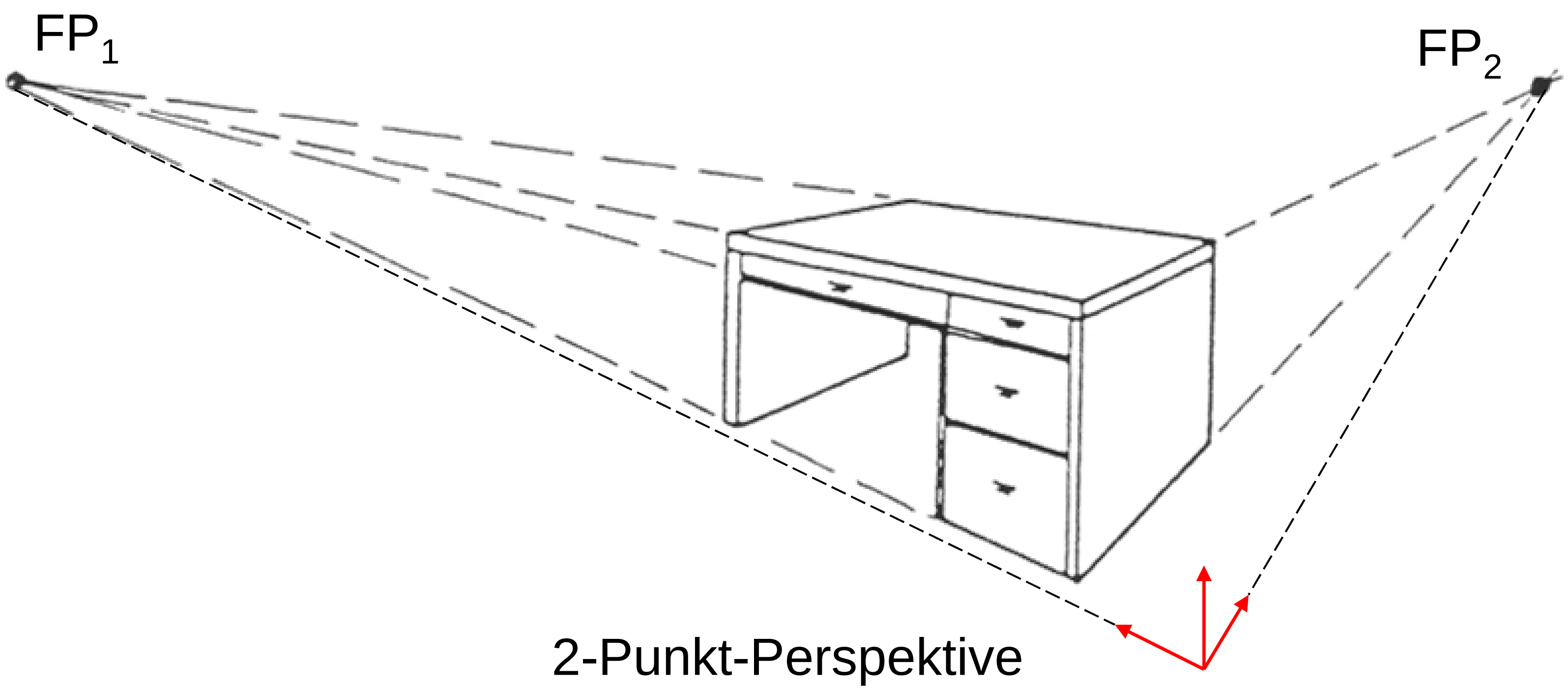
Beispiel: 1-Punkt-Perspektive



1-Punkt-Perspektive

3.4 Projektionen

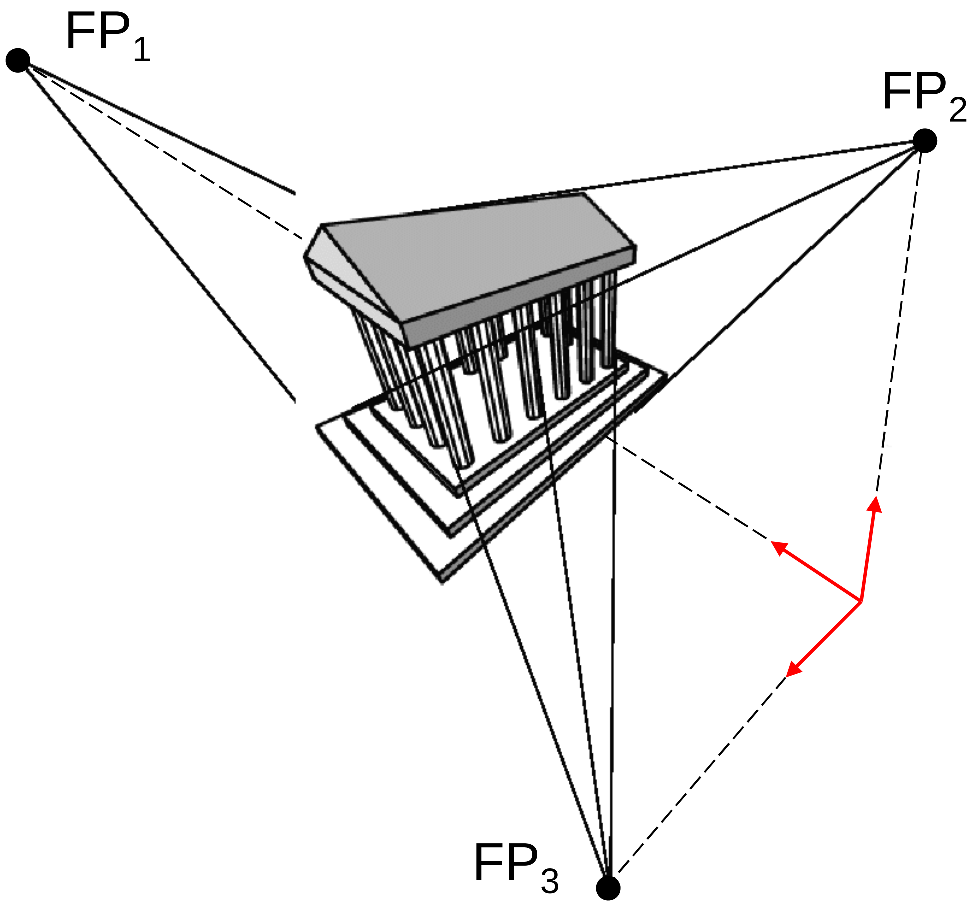
Beispiel: 2-Punkt-Perspektive



2-Punkt-Perspektive

3.4 Projektionen

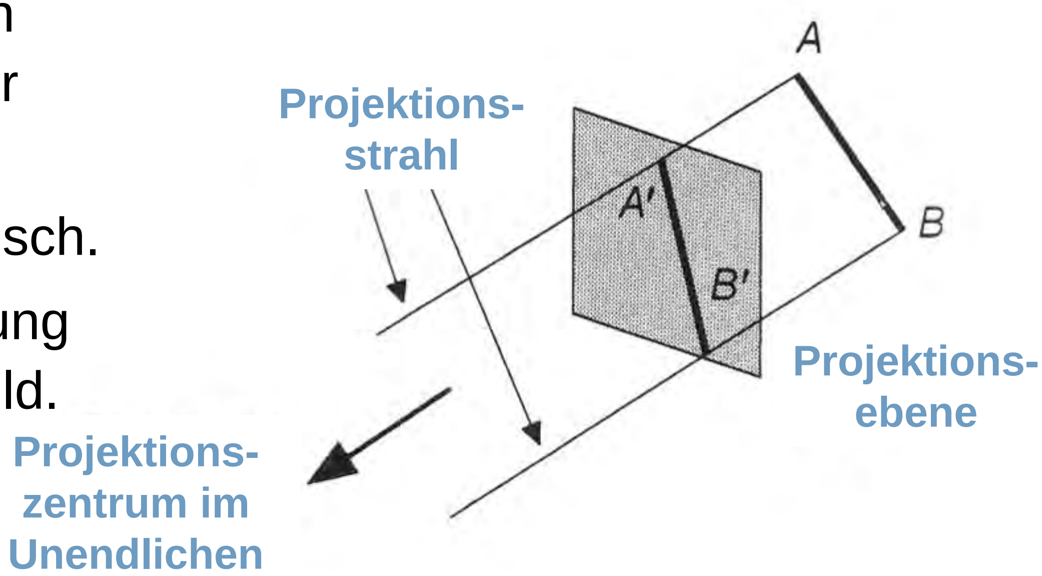
Beispiel: 3-Punkt-Perspektive



3.4 Projektionen

3.4.2 Parallelprojektionen

- Alle **Projektionsstrahlen** verlaufen parallel in eine Richtung.
- Das **Projektionszentrum** liegt in einem unendlich fernen Punkt.
- In der projektiven Geometrie stellt die Parallelprojektion somit einen Spezialfall der Zentralprojektion dar.
- **Nachteil:** Weniger realistisch.
- **Vorteil:** Erlaubt Bestimmung exakter Maße aus dem Bild.



3.4 Projektionen

- Die Projektionsstrahlen können bei Parallelprojektionen gegen die Projektionsebene
  - schief (**schiefewinklige Projektion**) oder
  - senkrecht (**orthographische/rechtwinklige Projektion**) stehen.

3.4.3 Rechtwinklige / senkrechte / orthographische Projektion

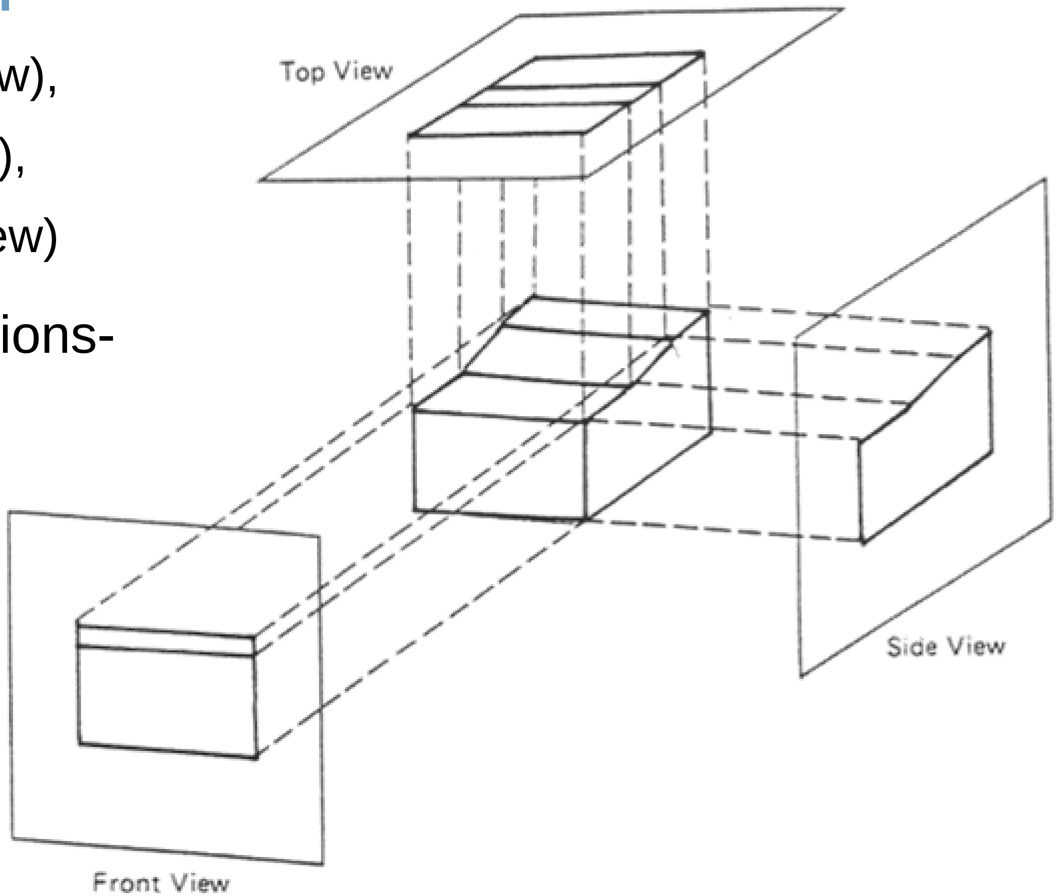
- Die Projektionsrichtung fällt mit der Normalen der Projektionsebene zusammen.
- Man unterscheidet **Hauptrisse** und **Axonometrie**.

3.4 Projektionen

- Bei den **Hauptrissen**
  - Grundriss (top view),
  - Aufriss (front view),
  - Kreuzriss (side view)

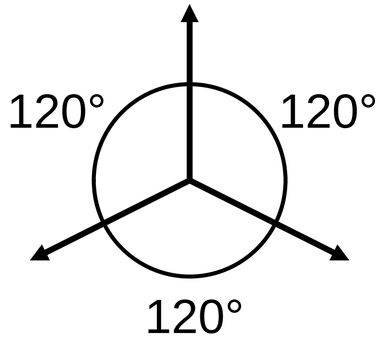
  schneidet die Projektions-ebene nur eine Hauptachse.

- Die Normale der Projektionsebene ist also parallel zu einer der Hauptachsen.



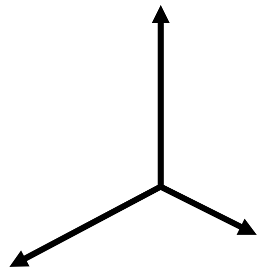
3.4 Projektionen

- Bei der **Axonometrie** ist die Projektionsebene nicht orthogonal zu einer der Koordinatenachsen (des Weltsystems).
  - Parallele Linien werden auf parallele Linien abgebildet.
  - Winkel bleiben nicht erhalten.
  - Abstände können längs der Hauptachsen gemessen werden, allerdings i.a. in jeweils einem anderen Maßstab.
- Bei der **isometrischen Axonometrie** bildet die Projektionsebene mit allen Hauptachsen den gleichen Winkel. Hier hat man eine gleichmäßige Verkürzung aller Koordinatenachsen.
- Es gibt nur acht mögliche isometrische Projektionen.

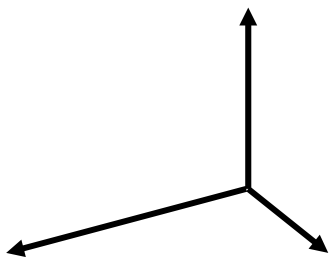


3.4 Projektionen

- Bei der **dimetrischen** Projektion bildet die Projektionsebene mit zwei Hauptachsen den gleichen Winkel, die Skalierung ist in zwei Achsenrichtungen gleich.



- Bei der **trimetrischen** Projektion bildet dieProjektionsebene mit jeder Achse einen anderen Winkel, die Skalierungen sind in den drei Achsenrichtungen verschieden.



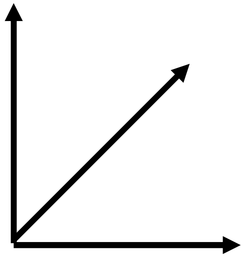
3.4 Projektionen

3.4.4 Schiefwinklige Parallelprojektionen

- Entstehen, wenn die Projektionsrichtung sich von der Normalen der Projektionsebene unterscheidet.
- Die beiden gebräuchlichsten schiefen Parallelprojektionen sind die sogenannte **Kavalier-** und **Militärprojektion**.

Kavalierprojektion

- Der Winkel zwischen Projektionsrichtung und Bildebene beträgt 45°. Hier bleibt die Länge der Projektion einer Linie, die senkrecht zur Bildebene steht, unverändert.
- Es gibt unendlich viele Kavalierprojektionen, eine für jede Richtung in der Bildebene.

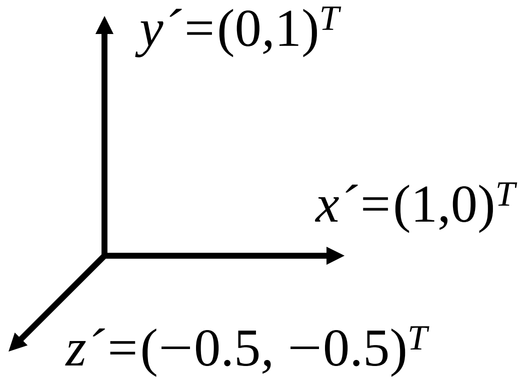


3.4 Projektionen

Militärprojektion / Kabinettprojektion

- Hier soll die Länge der Projektion einer zur Projektionsebene senkrechten Linie halbiert werden.
- Der Winkel zwischen der Projektionsrichtung und der Bildebene beträgt somit arctan $2 = 63.4°$.

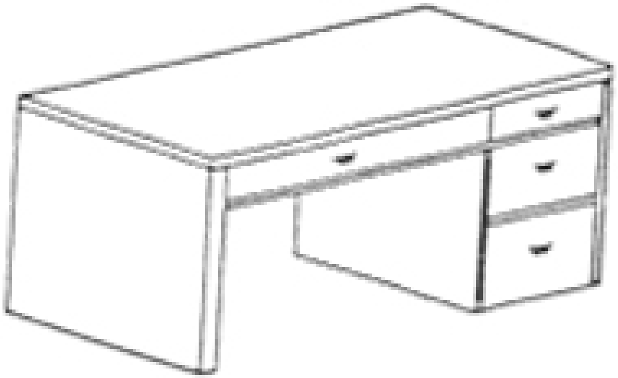
Beispiel:



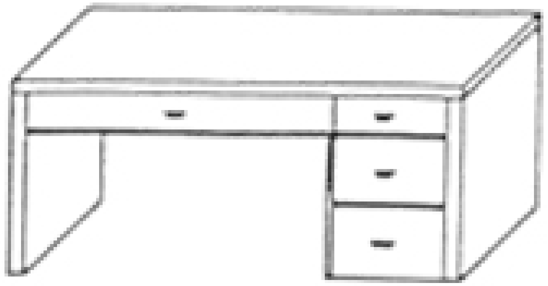
projizierte Einheitsvektoren

3.4 Projektionen

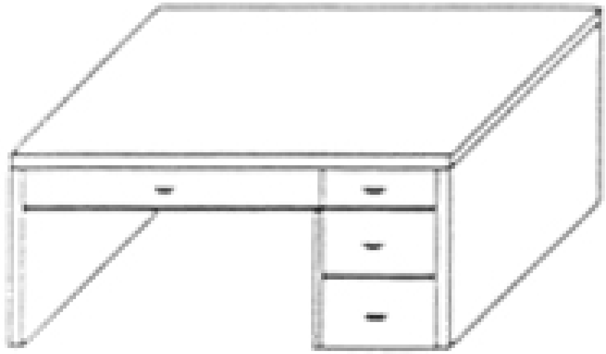
Beispiel:



Isometrische 1:1:1



Kabinettprojektion

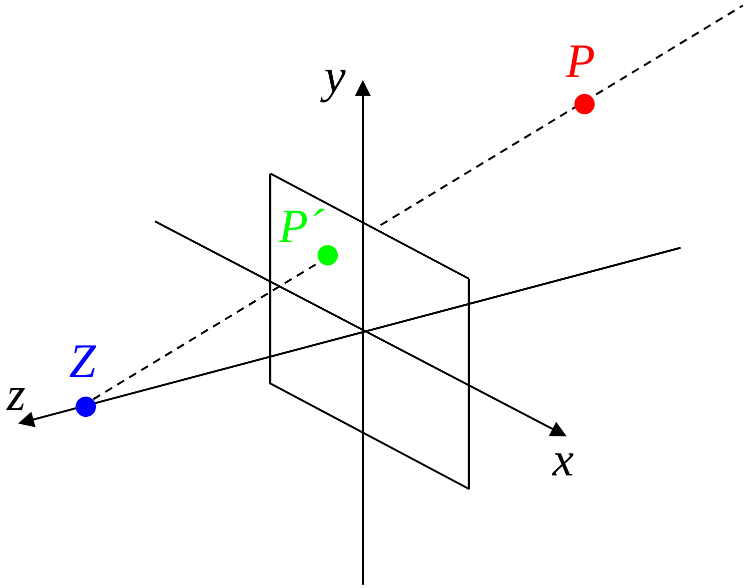


Kavalierprojektion

3.4 Projektionen

3.4.5 Umsetzung der Zentralprojektion

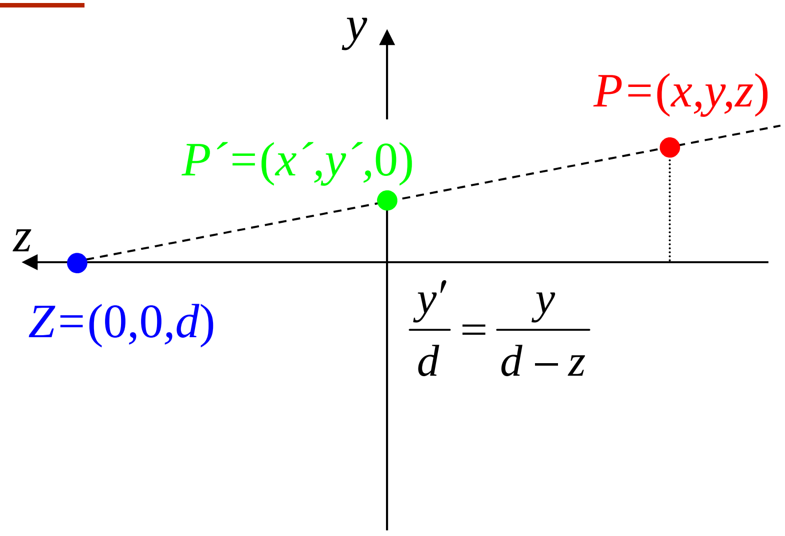
- Die praktische Umsetzung der perspektivischen Projektion erfolgt je nach Anwendung in unterschiedlichen Konfigurationen, die mittels geeigneter Transformationen des Koordinatensystems erreicht werden können.
- Exemplarisches Setup:
  - Projektionszentrum Z und Augpunkt fallen zusammen, liegen auf der positiven z-Achse mit Abstand $d>0$ zum Ursprung, also $Z=(0,0,d)$.
  - Blickrichtung ist die negative z-Achse.
  - Bildebene liegt in der (x,y)-Ebene.



3.4 Projektionen

Aus dem Strahlensatz folgt:

$$\frac{y'}{d} = \frac{y}{d-z} \quad \text{und} \quad \frac{x'}{d} = \frac{x}{d-z}$$



Folglich:

$$y' = y \cdot \left(\frac{d}{d-z}\right) = y \cdot \left(1 - \frac{z}{d}\right)^{-1}$$

$$x' = x \cdot \left(\frac{d}{d-z}\right) = x \cdot \left(1 - \frac{z}{d}\right)^{-1}$$

Die Zentralprojektion wird in diesem Setup somit durch folgende Matrix beschrieben:

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -d^{-1} & 1 \end{pmatrix}$$

3.4 Projektionen

- Erweitertes Setup:
  - In der Bildebene wird ein Sichtfenster (**view window**) spezifiziert durch
    - Breite b, Höhe h, Verhältnis Breite zu Höhe.
    - Es ist symmetrisch um den Ursprung angeordnet.
  - Die Projektoren durch die Ecken des Sichtfensters definieren das sogenannte Sichtvolumen (**viewing frustum**).
  - Zusätzlich begrenzen zwei zur Bildebene parallele Ebenen (**front und back clipping plane**) das Sichtvolumen in z-Richtung.
  - Das Sichtvolumen begrenzt den Teil des Raums, der dargestellt werden soll (siehe 3.6 Clipping).

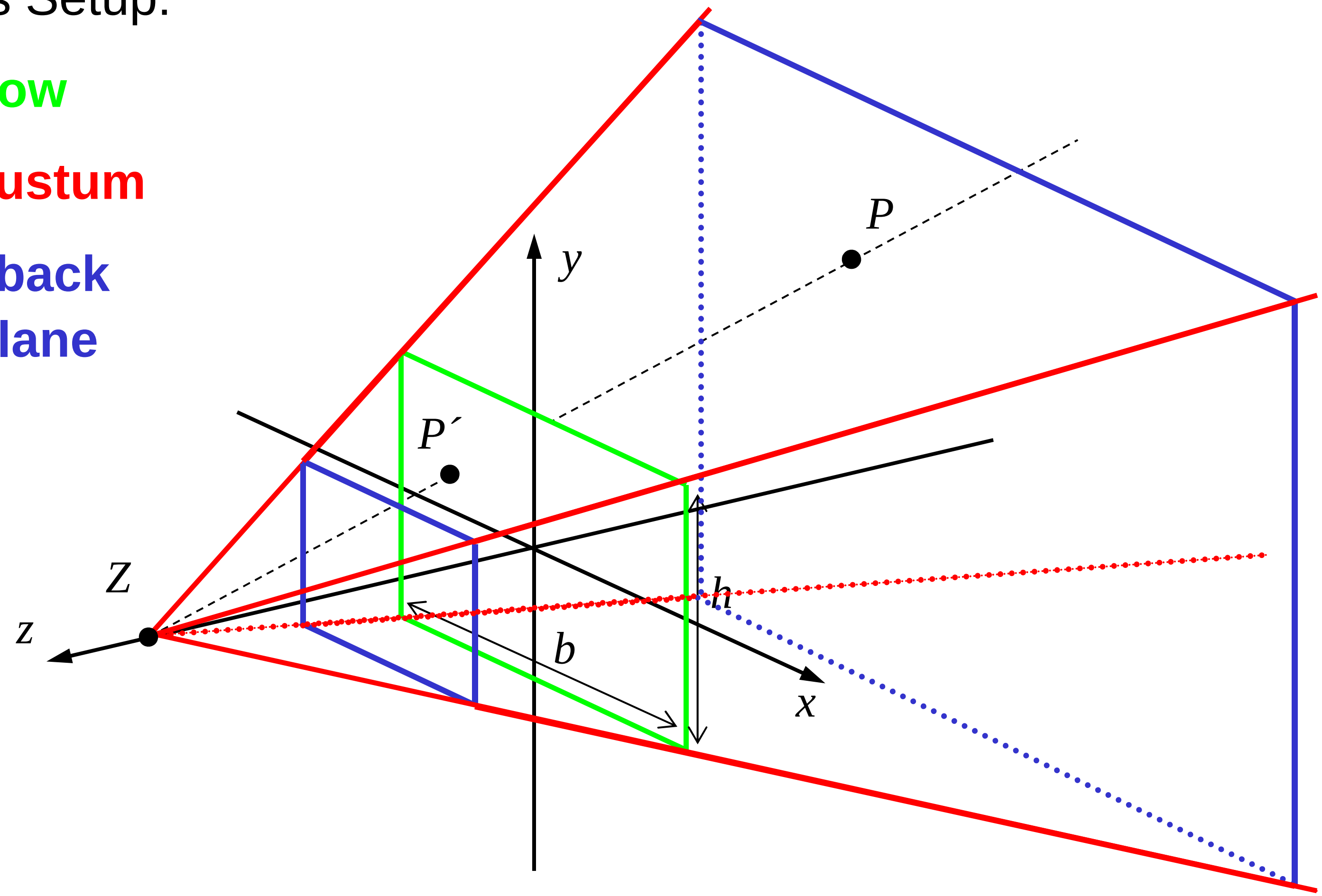
3.4 Projektionen

Erweitertes Setup:

view window

viewing frustum

front und back
clipping plane



3.5 Windowing

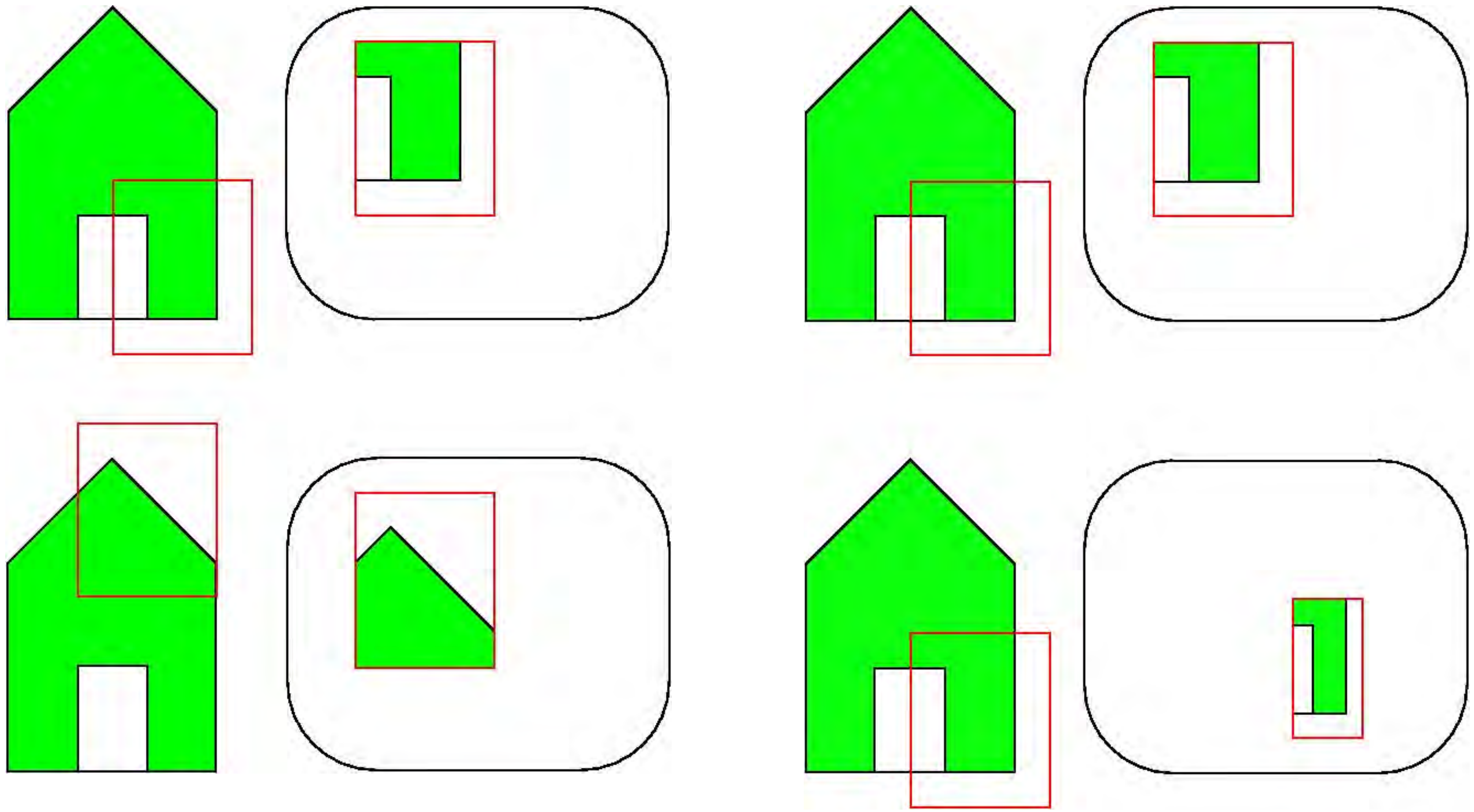
Window:
- Auch „view window" (→ erweitertes Setup Zentralprojektion)
- Definiert Sichtfenster in der Bildebene.
- Definiert, welcher Teilbereich der Szene abgebildet werden soll.

Viewport:
- Definiert Bildschirmbereich, in dem der Inhalt eines Windows dargestellt werden soll.

- In der Regel sind sowohl Window als auch Viewport an den Koordinatenachsen ausgerichtete rechteckige Gebiete.

- Die Windowing-Operation (Window-Viewport-Transformation) setzt sich aus elementaren Translationen und Skalierungen zusammen.

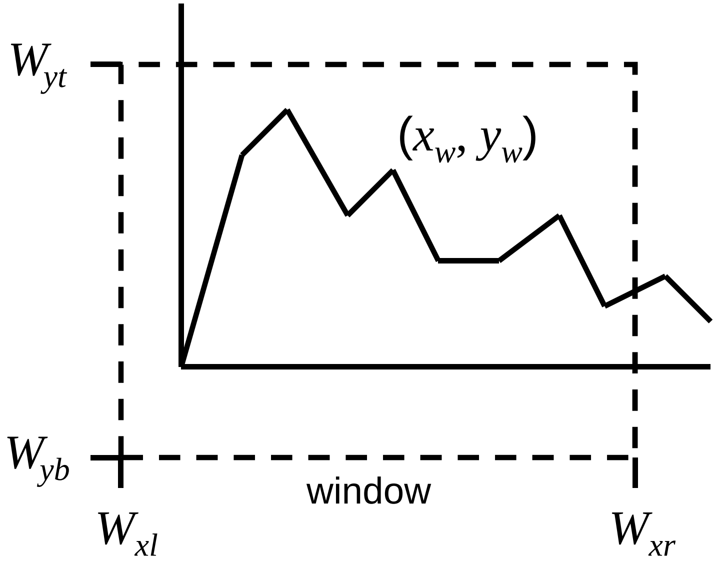
3.1 Koordinatentransformationen

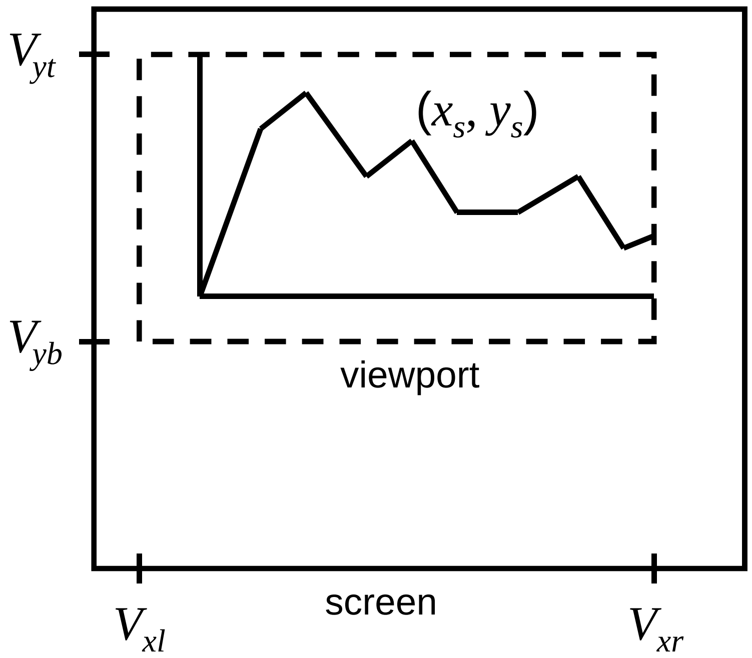


Verschiedene Fenster, dieselben Viewports

Dieselben Fenster, verschiedene Viewports

3.5 Windowing





x_w, y_w	Punktkoordinaten im Window
x_s, y_s	Punktkoordinaten auf dem Bildschirm
$W_{xl}, W_{xr}, W_{yb}, W_{yt}$	Koordinaten des Windows
$V_{xl}, V_{xr}, V_{yb}, V_{yt}$	Koordinaten des Viewports im Bildschirmkoordinatensystem

3.5 Windowing

Transformation in 3 Schritten

1) Translation in den Koordinatenursprung

$$x' = x_w - W_{xl},$$
$$y' = y_w - W_{yb}.$$

2) Skalierung auf gewünschte Größe

$$x'' = (V_{xr} - V_{xl}) / (W_{xr} - W_{xl}) \cdot x',$$
$$y'' = (V_{yt} - V_{yb}) / (W_{yt} - W_{yl}) \cdot y'.$$

3) Translation an die gewünschte Stelle

$$x_s = x'' + V_{xl},$$
$$y_s = y'' + V_{yb}.$$

3.5 Windowing

Zusammengefasst

$$x_s = a \cdot x_w + b,$$

$$y_s = b \cdot x_w + d,$$

mit $a = (V_{xr} - V_{xl}) \,/\, (W_{xr} - W_{xl}),\qquad b = V_{xl} - a \cdot W_{xl},$

$c = (V_{yt} - V_{yb}) \,/\, (W_{yt} - W_{yb}),\qquad d = V_{yb} - c \cdot W_{yb},$

→ Punkttransformation durch zwei Multiplikationen und zwei Additionen.

3.6 Clipping

Sollen Objekte in der Bildebene innerhalb eines Fensters dargestellt werden, so wird ein Verfahren benötigt, um alle außerhalb des Fensters liegenden Objektteile abzuschneiden:

Clipping am Fensterrand

3.6 Clipping

3.7.1 Clipping von Linien

Clipping an einem rechteckigen, achsenparallelen Fenster.

Offensichtlich gibt es drei Fälle, von denen zwei schnell gelöst sind:

1. Beide Endpunkte der Linie liegen innerhalb des Fensters:

   Linie zeichnen.

2. Beide Endpunkte der Linie liegen oberhalb, unterhalb, links oder rechts des Fensters:

   Linie nicht zeichnen.

3. Sonst: Schnittpunkte der Linie mit dem Fensterrand berechnen und daraus die sichtbare Strecke bestimmen.

3.6 Clipping

3.7.2 Cohen-Sutherland Line-Clipping Algorithmus

Idee: Schnelles Verfahren zur Klassifizierung der Linien als innerhalb, außerhalb, schneidend.

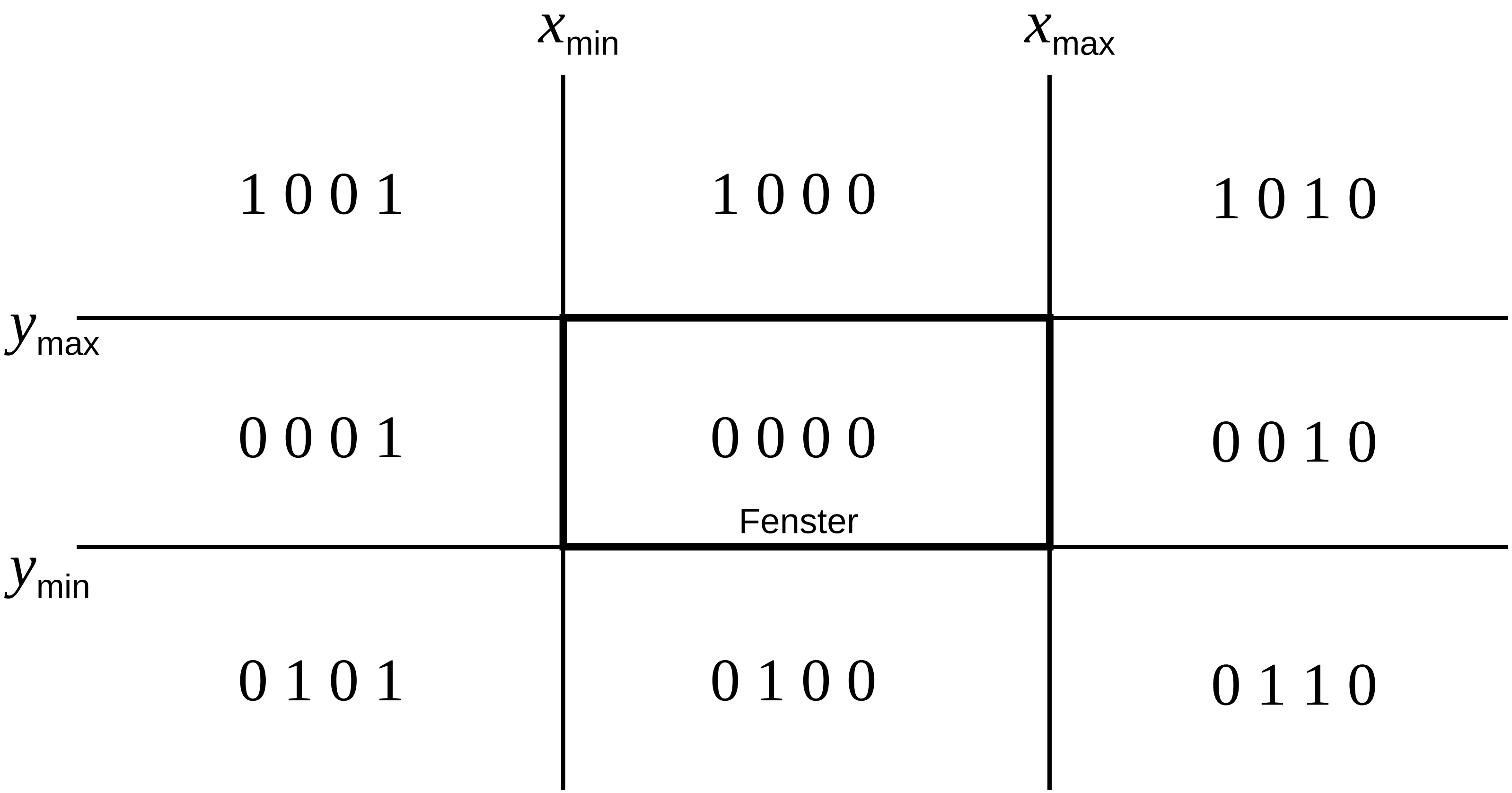
Gegeben: Fenster (x_{min}, y_{min}, x_{max}, y_{max})

- Die begrenzenden Geraden zerlegen die Bildebene in neun Regionen. Ein 4-Bit-Code gibt Auskunft über die Lage in Bezug auf das Fenster:

	... gesetzt, falls Region ...	
Bit 0	... links des Fensters	$x < x_{\text{min}}$
Bit 1	... rechts des Fensters	$x > x_{\text{max}}$
Bit 2	... unterhalb des Fensters	$y < y_{\text{min}}$
Bit 3	... oberhalb des Fensters	$y > y_{\text{max}}$

3.6 Clipping

Codes für Fenster und umgebende Regionen



3.6 Clipping

- Für die Endpunkte einer Linie bestimme die 4-Bit-Codes:
  - Die Linie liegt **komplett im Fenster**, wenn beide Endpunkte den 4-Bit-Code 0000 besitzen (OR-Verknüpfung ist Null).
  - Die Linie liegt **vollständig außerhalb des Fensters**, wenn der Durchschnitt (AND-Verknüpfung) der 4-Bit-Codes beider Endpunkte von Null verschieden ist.
  - Sonst:
    - Schneide alle Linie nacheinander mit den das Fenster begrenzenden Geraden.
    - Zerlege jede Linie in zwei Teile und bestimme die 4-Bit-Codes der neuen Endpunkte.
    - Der außen liegende Teil wird sofort eliminiert.

3.6 Clipping

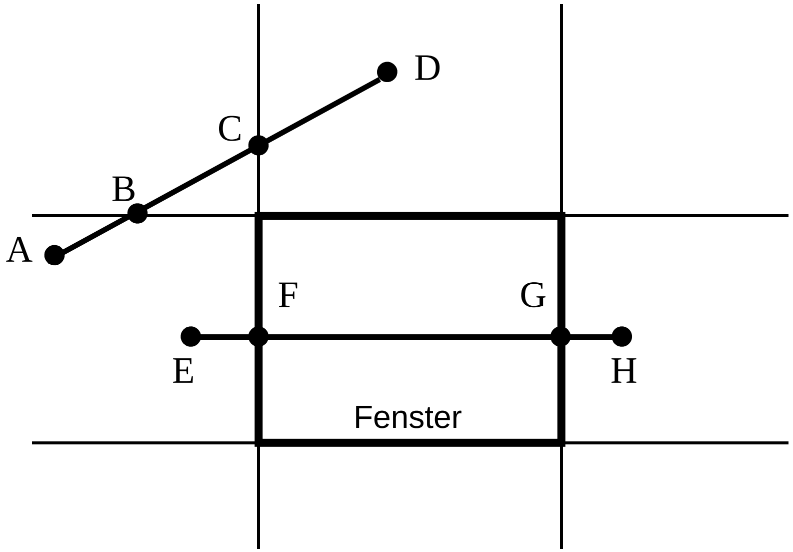
Beispiele:

- **Linie AD**:

  Codes 0001 und 1000 → Schnittberechnung

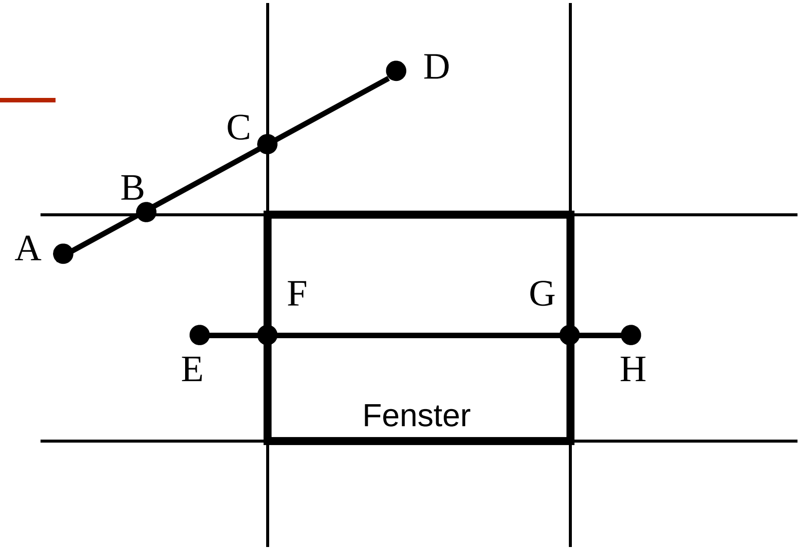
  Schnitt mit linker Fenstergrenze liefert C → eliminiere AC

  C und D liegen oberhalb des Fensters → eliminiere CD



3.6 Clipping

Beispiele:



- **Linie EH:**

  Codes 0001 und 0010 → Schnittberechnung

  Schnitt mit linker Fenstergrenze liefert F → eliminiere EF

- **Linie FH:**

  Codes 0000 und 0010 → Schnittberechnung

  Schnitt mit rechter Fenstergrenze liefert G → eliminiere GH

- **Linie FG:**

  Codes 0000 und 0000 → FG wird gezeichnet

3.6 Clipping

Spezialfälle und Beschleunigungen

- Senkrechte/waagrechte Linien: nur y-/x-Grenzen testen und schneiden.
- Genau ein Endpunkt im Fenster: nur ein Schnitt mit dem Fensterrand.
- Überflüssige Schnittoperationen aus den Bitcodes ermitteln:
  - Jedes Bit korrespondiert genau zu einem Fensterrand.

  → Nur die Fensterränder betrachten, deren zugehörige Bits in den Endpunkt-Codes verschieden sind.

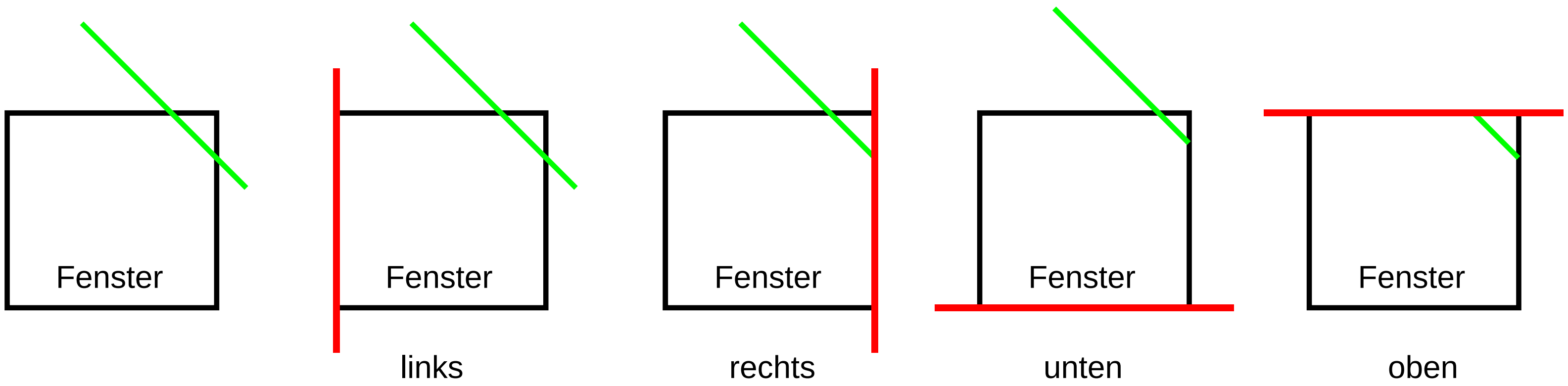
3.6 Clipping

- Schnittpunktberechnung durch Bisektionsmethode vermeiden:
  - Linien, die weder ganz außerhalb, noch ganz innerhalb des Fensters liegen, werden so lange unterteilt, bis ihre Länge kleiner als ein Pixel ist.
  - Bei $2^{10}=1024$ Pixeln in einer Zeile bzw. Spalte erfordert dies maximal 10 Unterteilungen.
  - In Hardware ist diese Variante wegen der schnellen Division durch 2 und ihrer Parallelisierbarkeit schneller als eine direkte Schnittberechnung.

3.6 Clipping

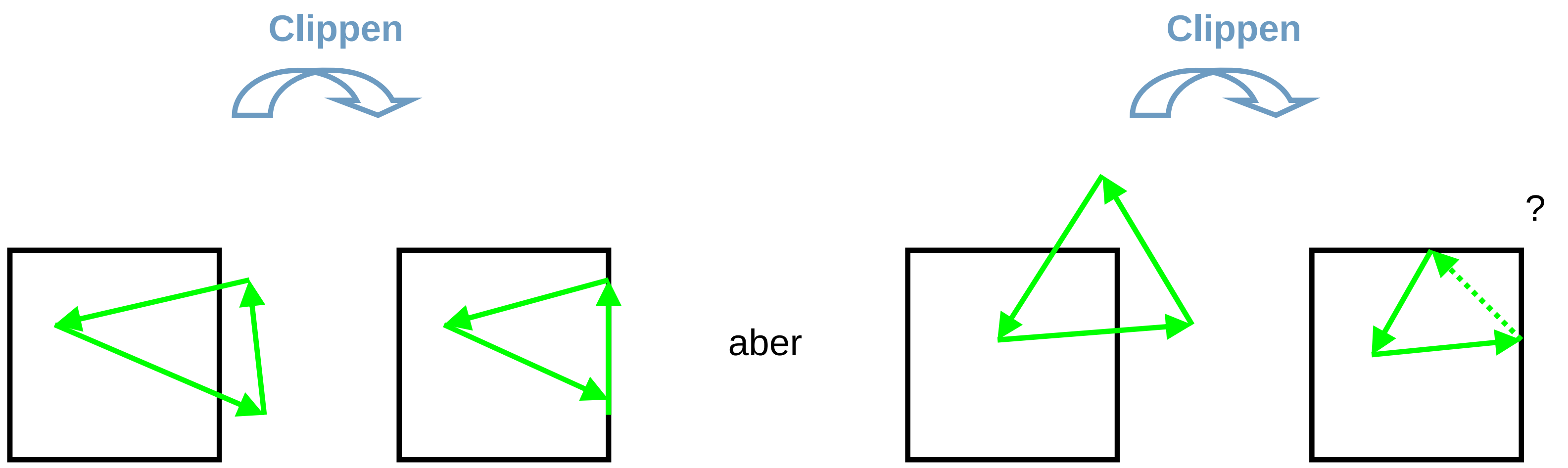
3.7.3 Clipping von Polygonen

- Polygone als Begrenzung von Flächen wichtig.
- Naiver Ansatz: jede Seite gegen die Fenster clippen.



3.6 Clipping

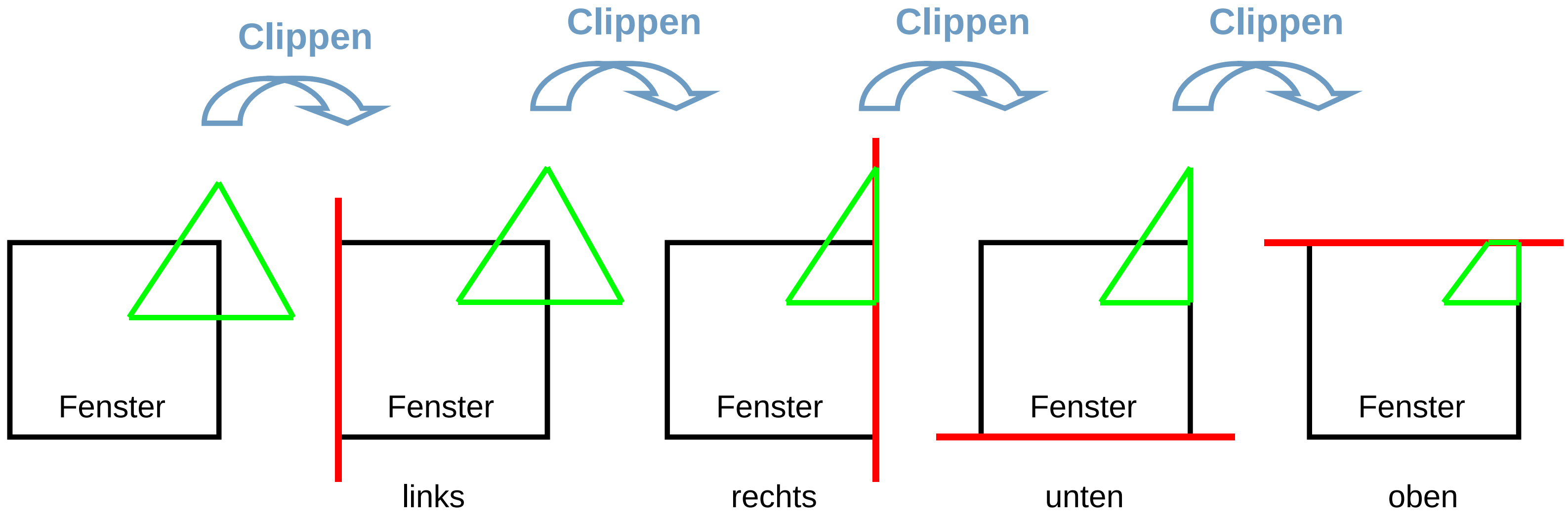
- Problem: Polygon-Clipping muss wieder geschlossene Polygone liefern, also ggf. Teile des Fensterrandes mit zurückgeben.



3.6 Clipping

3.7.4 Sutherland-Hodgman Polygon-Clipping Algorithmus

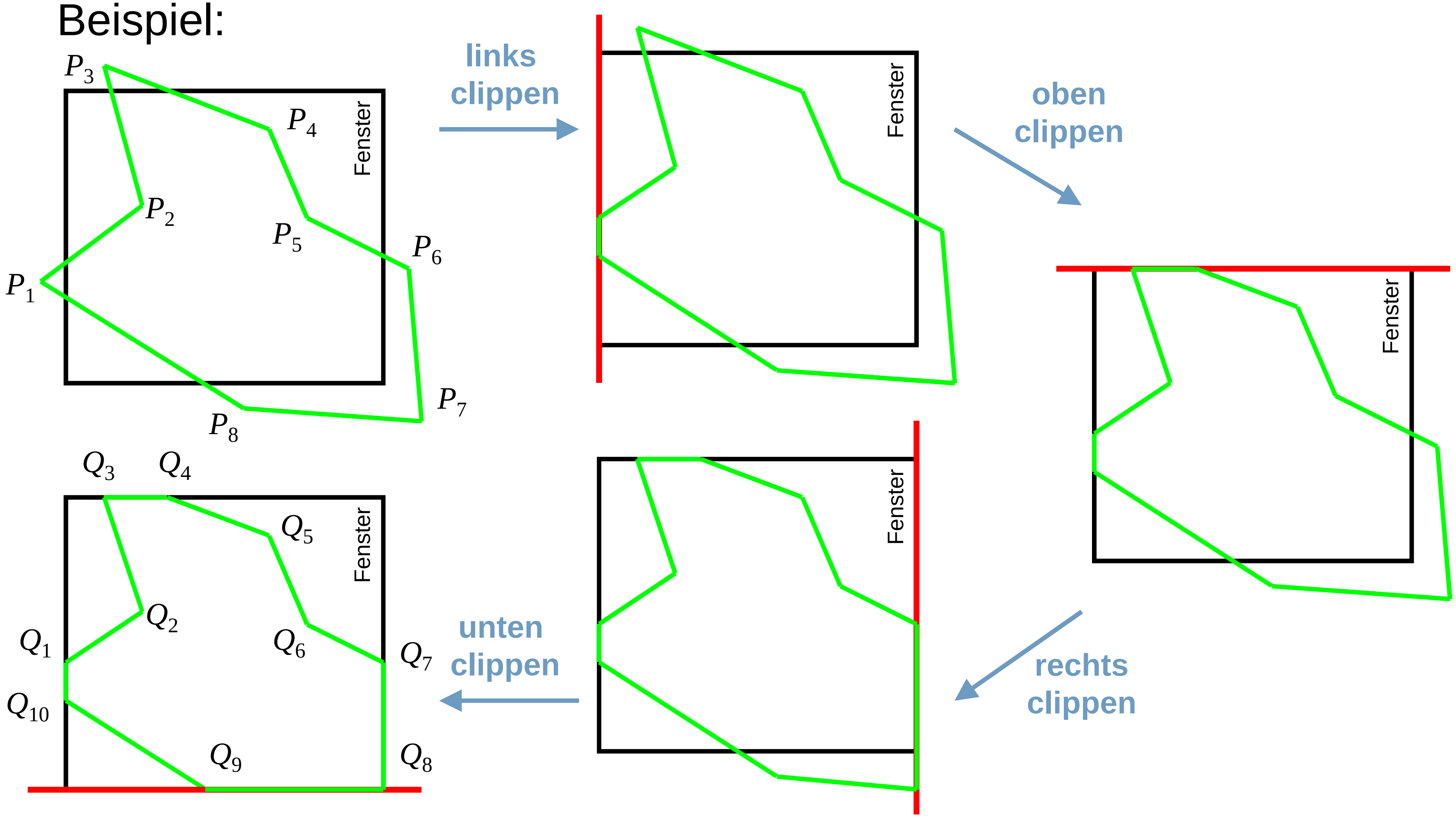
- Vollständige Clippen des Polygons gegen eine Fensterseite.



- Die Zwischenergebnisse müssen gespeichert werden.

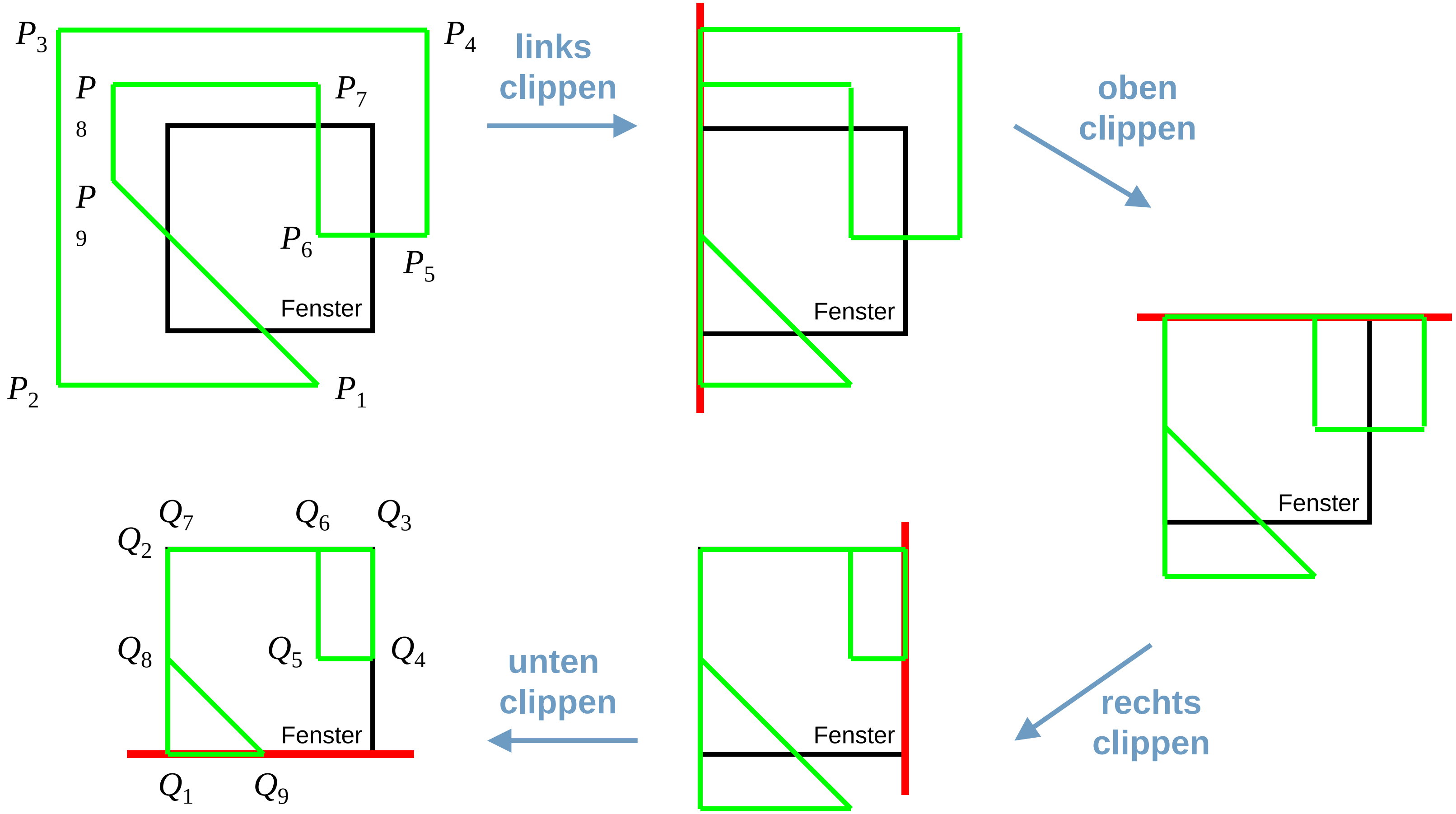
3.6 Clipping

Beispiel:



3.6 Clipping

Beispiel:



3.6 Clipping

3.7.5 Clipping im Raum

- Statt nach der Projektion in die Bildebene kann auch bereits im dreidimensionalen Objektraum geclippt werden.
  - Vorteil: Nur die sichtbaren Objekte müssen transformiert werden.
- Mit den Ebenengleichungen der Randflächen des Frustums kann bestimmt werden, ob ein gegebener Punkt (Linie, Objekt, etc.) außerhalb oder innerhalb liegt.
- Die vorgestellten zweidimensionalen Clipping-Verfahren können analog auf den dreidimensionalen Fall übertragen werden.

Lernziele

- Was ist der Unterschied zwischen Objekt-, Welt-, und Sichtkoordinaten?
- Wie können Translationen, Rotationen und Skalierungen im 2D und 3D beschrieben werden?
- Was sind affine Transformationen?
- Was sind homogenen Koordinaten?
- Was ist eine perspektivische Projektion?
- Was ist eine Parallelprojektion?
- Was ist das Frustum?
- Was ist der Unterschied zwischen Window und Viewport?
- Was ist Windowing?
- Wie arbeiten Clipping-Algorithmen für Linien und Polygone?