

## 2.1 Motivation

Wichtige Grundlage der Bildwiedergabe auf dem Bildschirm oder anderen Ausgabegeräten sind **Koordinatensysteme** und **Koordinatentransformationen** im  $\mathbb{IR}^2$  und  $\mathbb{IR}^3$ .

Im allgemeinen unterscheidet sich das Koordinatensystem eines Objektes von dem Koordinatensystem des Ausgabegerätes:

- Das Koordinatensystem eines Objektes ist oft über geometrische Eigenschaften des Objektes festgelegt.
- Das Koordinatensystem des „Bildschirms“ und die Größe der Bildfenster sind durch das Gerät selbst definiert (z. B. Nullpunkt in der linken, oberen Ecke, x- und y-Achsen parallel zu den Bildrändern).
- Mit (mehreren) Koordinatentransformationen wird das Objektsystem in das Gerätesystem transformiert.



## 2.1 Motivation

Spezieller:

Objektdefinition, Modeling, Sichtdefinition,  
Animation, Rendering, Ausgabe, ...

- verschiedenste Koordinatensysteme
  - Koordinatentransformationen im 2D-, 3D-Raum  
(Verschiebungen, Skalierungen, Drehungen,...)  
zur Definition von Position **und Orientierung!**
  - **Matrixoperationen**
  - Hardware- und Softwareunterstützung
- aber auch: Grundlagenverständnis, Gefühl!



## 2.2 Koordinatensysteme

**local coordinate system, object coordinate system, modeling coordinate system, 3D**

- eigenes Koordinatensystem für jedes Objekt sinnvoll
- erleichtert Modellierung und Animation!
- oft über geometrische Eigenschaften des Objektes (z. B. Symmetrien oder ausgezeichnete Richtungen) oder durch gewünschte Aktionen des Objektes (z. B. Flug entlang Kurve mit gleichzeitiger Drehung) festgelegt
- **findet auch für Lichtquellen Verwendung!**



## 2.2 Koordinatensysteme

**world coordinate system, global coordinate system, scene coordinate system, 3D**

- hier „lebt“ die gesamte Szene
- Objekte (Objektkoordinatensysteme) werden mittels Transformationen platziert
- zum Rendern: Umrechnung lokaler Objektkoordinaten in globale Objektkoordinaten
  - (bei Animationen d. h. animiertem Objekt: dynamisch in jedem Frame!)
- hier ebenso: **Definition der Beleuchtung der Szene**



## 2.2 Koordinatensysteme

### eye coordinate system, camera coordinate system, 3D

- Konzept: **virtuelle Kamera** mit entsprechenden Parametern
- im Weltsystem (mittels Transformationen) verankert
- so soll es der Beobachter sehen  
→ hier wird der Bildausschnitt definiert!
- zum Rendern: i. d. R. Umrechnung globaler Koordinaten in Kamerakoordinaten („Szene vor die Kamera drehen“)  
(bei Animationen d. h. animierter Kamera:  
dynamisch in jedem Frame!)
- Bem.: virtuelle Kamera bestimmt auch Art der **Projektion**



## 2.2 Koordinatensysteme

### view coordinate system, view plane coordinate system, 2D

- durch die Projektion (Kameratransformation) werden 3D-Koordinaten 2D-Koordinaten in der **Bildebene** zugeordnet
- Koordinatensystem in der zweidimensionalen Bildebene nach der Projektion
- ein **Window** definiert ein Sichtfenster in der Bildebene

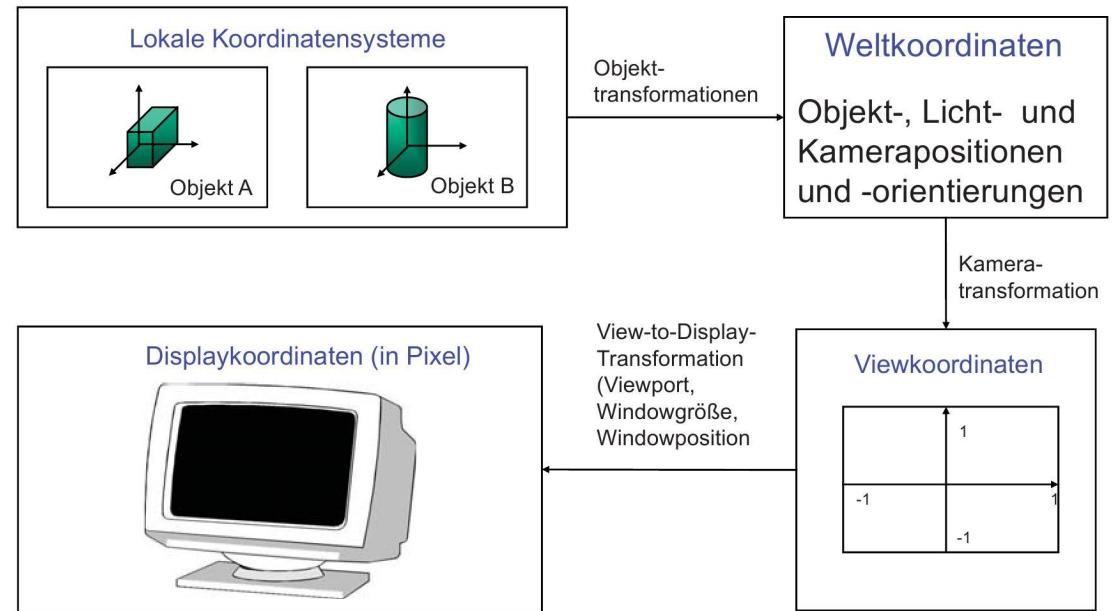
### display coordinate system, device coordinate system, 2D

- definiert das Koordinatensystem des Gerätes
- ein **Viewport** definiert einen Bereich, in dem der Inhalt eines Windows dargestellt werden soll („Windowgröße“, „-position“)
- Window-Viewport-Transformationen sind 2D-2D



## 2.2 Koordinatensysteme

Überblick / Zusammenspiel:



## 2.2 Koordinatensysteme

**Bemerkungen:** 3D-Koordinatensysteme

Wir verwenden (bis auf Widerruf)  
**orthonormierte kartesische** Koordinatensysteme!

Wiederholung:

**orthonormiert:** orthogonal, d. h. Basisvektoren paarweise senkrecht

- + Basisvektoren **normiert (Länge 1)**

**kartesisch:** orthogonal

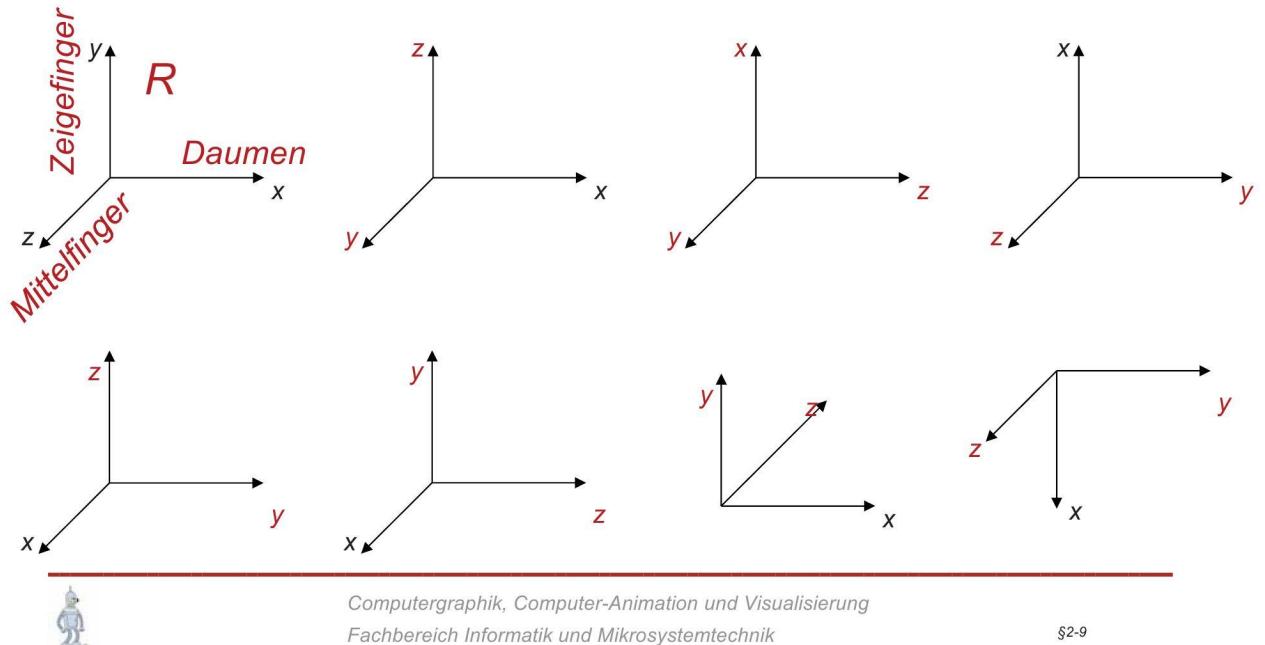
- + Basisvektoren gleiche Länge
- + Basisvektoren **bilden Rechtssystem**



## 2.2 Koordinatensysteme

**Bemerkungen:** 3D-Koordinatensysteme (cont.)

Rechts-/Linkssystem?



## 2.3 Transformationen

Bemerkung:

**Vorsicht vor Verwirrungen!**

Für Transformationen existieren mehrere  
äquivalente Sichtweisen!

- „Transformation der Punkte“ **A**
- Koordinatensystem bleibt fest
- Punkt ändert Ort und damit Koordinaten im gleichen System
- beteiligte Koordinatensysteme: **1**



## 2.3 Transformationen

Bemerkung: (cont.)

- „Transformation des Koordinatensystems“ **B**
  - Koordinatensystem wird der inversen Transformation der Punkttransformation unterworfen  
→ Koordinatensystem bewegt sich!
  - Punkt bleibt fest, ändert allerdings Koordinaten, da in neuem System beschrieben
  - beteiligte Koordinatensysteme: eigentlich nur 1  
(aber: vorher, nachher)  
→ „Transformation zwischen Koordinatensystemen“



## 2.3 Transformationen

Bemerkung: (cont.)

hervorragend geeignet  
für Modellierung und Animation!

- „Transformation mittels Koordinatensystem“ **C**
  - ein dem Punkt lokales Koordinatensystem wird mittels der Punkttransformation im globalen Koordinatensystem bewegt  
→ (lokales) Koordinatensystem **bewegt sich!**
  - anschließend wird Punkt (mit „alten“ Koordinaten) in bewegtes lokales Koordinatensystem eingetragen
  - Punkt **ändert Ort!**
  - beteiligte Koordinatensysteme: **2**



## 2.4 Transformationen in der Ebene

Es geht los:

Wir verwenden zunächst parallel die Sichtweisen  
 „Transformation der Punkte“ und  
 „Transformation zwischen Koordinatensystemen“

Gegeben:

Koordinatensystem  $S'$  (z. B. Weltsystem)  
 durch  $S':(O'; x_1', x_2')$ , sowie

Koordinatensystem  $S$  (z. B. Objektsystem)  
 durch  $S:(O; x_1, x_2)$

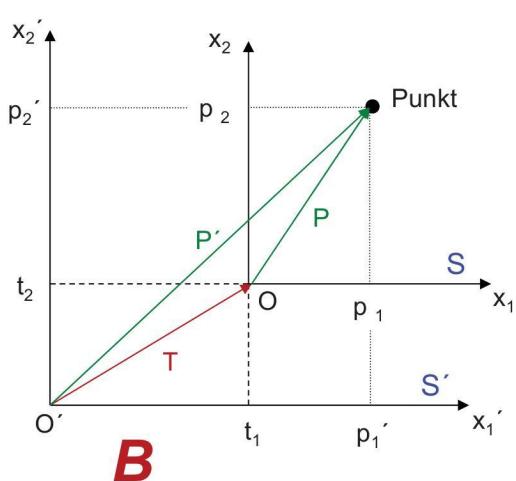
Wir transformieren  $S$  nach  $S'$ , also ändern sich  
 Punktkoordinaten von System  **$S$  nach  $S'$** .



## 2.4 Transformationen in der Ebene

### Translation / Verschiebung

Voraussetzung: beide (gerichteten) Koordinatenachsen  
 sind jeweils zueinander parallel



Es gilt für das System  $S'$ :

$S'$  ergibt sich aus  $S$   
 durch Verschiebung um  $-T$

$T=(t_1, t_2)^T$  sind die Koordinaten  
 des Ursprung von System  $S$  in System  $S'$

Es gilt für den Punkt:

hat in  $S$  die Koordinaten  $P = (p_1, p_2)^T$   
 hat in  $S'$  die Koordinaten  
 $P' = T + P = (t_1, t_2)^T + (p_1, p_2)^T$



## 2.4 Transformationen in der Ebene

### Translation / Verschiebung (cont.)

**A**

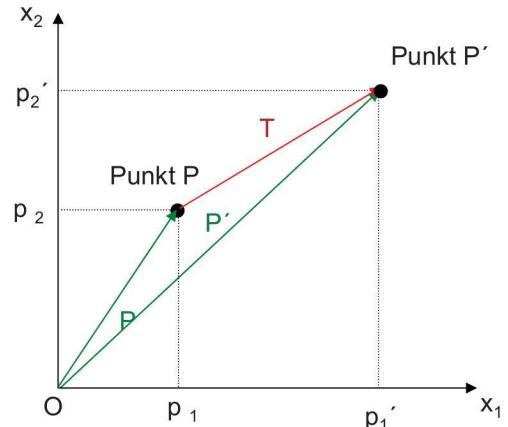
Es gilt für den Punkt:

Punkt  $P$  wird um  $T$  verschoben,  
es entsteht neuer Punkt  $P'$   
mit  $P' = T + P$

Bei beiden Sichtweisen gilt:

Vektor-Matrix-Schreibweise:

$$\begin{pmatrix} p'_1 \\ p'_2 \end{pmatrix} = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix} + \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}$$



kurz:

$$P' = T + P$$



## 2.4 Transformationen in der Ebene

### Rotation / Drehung

Drehung von  $S$  gegenüber  $S'$   
um Winkel  $\varphi$  um gemeinsamen  
Ursprung  $O=O'$

Bem.: Drehung im math. positiven Sinn  
verläuft gegen den Uhrzeigersinn!

Es gilt für das System  $S'$ :

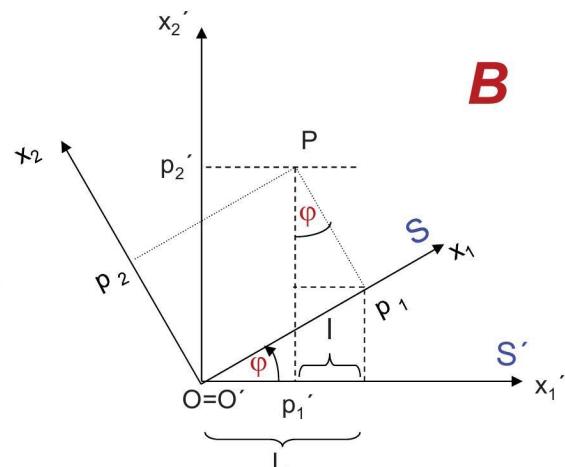
$S'$  ergibt sich aus  $S$   
durch Drehung um  $-\varphi$

Geometrie: Es ist  $1/p_2 = \sin \varphi$  und  $L/p_1 = \cos \varphi$

$$\text{also } p'_1 = L - 1 = p_1 \cdot \cos \varphi - p_2 \cdot \sin \varphi$$

$$\text{analog } p'_2 = p_1 \cdot \sin \varphi + p_2 \cdot \cos \varphi$$

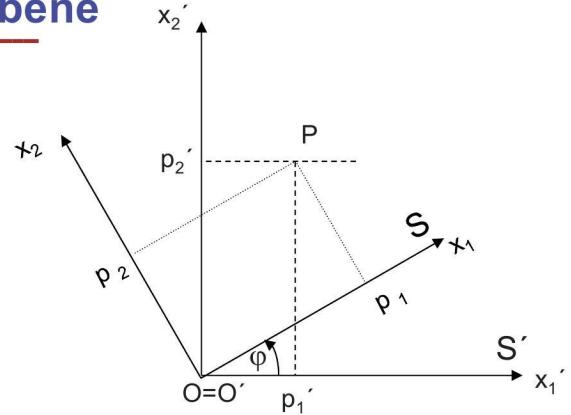
**B**



## 2.4 Transformationen in der Ebene

### Rotation / Drehung (cont.)

$$\begin{aligned} P' &= \begin{pmatrix} p'_1 \\ p'_2 \end{pmatrix} = \begin{pmatrix} p_1 \cdot \cos \varphi - p_2 \cdot \sin \varphi \\ p_1 \cdot \sin \varphi + p_2 \cdot \cos \varphi \end{pmatrix} \\ &= \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix} \cdot \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \\ &= R \cdot P \end{aligned}$$



Vektor-Matrix-Schreibweise:  $P' = R \cdot P$

mit der (orthogonalen) (Dreh-)Matrix:  $R = \begin{pmatrix} \cos \varphi & -\sin \varphi \\ \sin \varphi & \cos \varphi \end{pmatrix}$

*Spaltenvektoren bilden normiertes Orthogonalsystem,  
es ist  $|\det(R)|=1$ , es ist  $A^{-1} = A^T$*

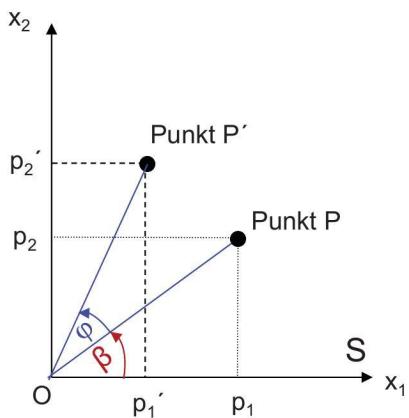


## 2.4 Transformationen in der Ebene

### Rotation / Drehung (cont.)

**A**

In dieser Sichtweise  
„Transformation der Punkte“  
dreht die Matrix  $R(\varphi)$  Punkte  
um den Winkel  $\varphi$  in positiver  
Richtung in einem  
festen Koordinatensystem

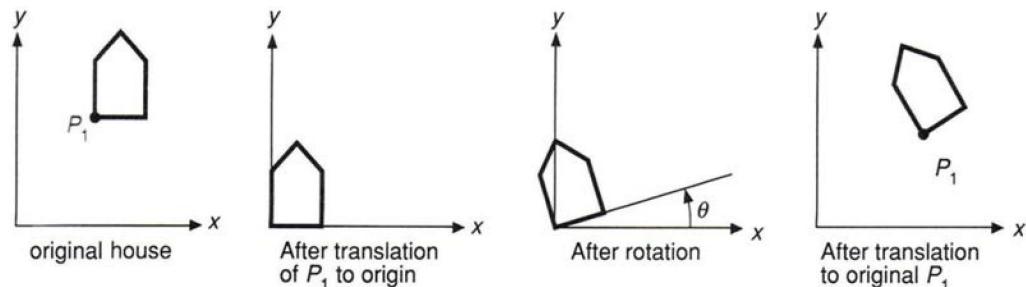


## 2.4 Transformationen in der Ebene

### Rotation / Drehung (cont.)

Bei der **Rotation um einen beliebigen Punkt  $P_1$**  müssen zusätzlich zwei Translationen gemacht werden:

- (1) Translation von  $P_1$  in den Ursprung
- (2) Rotation um den Ursprung
- (3) „Rücktranslation“ von  $P_1$  in die ursprüngliche Position



## 2.4 Transformationen in der Ebene

### Rotation / Drehung (cont.)

Bemerkung:

Die Matrixmultiplikation ist i. a. nicht kommutativ,  
d. h.  $A \cdot B \neq B \cdot A$

Bei hintereinander geschalteten Rotationen muss also darauf geachtet werden, dass die Reihenfolge der Matrizen der der Rotationen entspricht.

**Diese Aussage gilt grundsätzlich für jede Art zusammengesetzter Transformationen!**



## 2.4 Transformationen in der Ebene

### Scaling / Skalierung / Größenveränderung

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

$$p'_1 = s_1 \cdot p_1$$

$$p'_2 = s_2 \cdot p_2$$

Vektor-Matrix-Schreibweise:

$$P' = S \cdot P$$

mit der Skalierungsmatrix:

$$S = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}$$

*Vorsicht! Skalierungen nur sehr sehr sparsam und bedacht einsetzen!*

*können Orthonormalität, Orthogonalität, kartesisches System, Rechtssystem, Normalenlängen und Normalen ZERSTÖREN!*



## 2.4 Transformationen in der Ebene

### Homogene Koordinaten

- entstammen „projektiver Geometrie“
- unsere Motivation: Hintereinanderschaltung von Rotationen, Translationen und Skalierungen führt auf Zusammenhänge wie z. B.  $P' = R \cdot S \cdot (T + R \cdot P)$

Problem:

- bei der Hintereinanderausführung mehrerer Transformationen stört die **Addition von Translationen**
- heutige Computergrafikhardware unterstützt aber (zusammengesetzte) Matrixmultiplikationen, wie z. B.  $P' = M_n \cdot \dots \cdot M_3 \cdot M_2 \cdot M_1 \cdot P$



## 2.4 Transformationen in der Ebene

### Homogene Koordinaten (cont.)

→ einheitliche Matrixdarstellung aller Transformationen durch Übergang auf die nächst höhere Dimension

Das Tripel  $(x, y, w)^T$  ( $w \neq 0$ ) stellt die homogenen Koordinaten des Punktes  $(x/w, y/w)^T \in \mathbb{R}^2$  dar.

Da es unendlich viele solcher Darstellungen des selben Punktes gibt, verwendet man i. d. R. die sogenannte Standarddarstellung mit  $w=1$ .

Also besitzt ein Punkt  $(x, y)^T \in \mathbb{R}^2$   
z. B. die homogenen Koordinaten  $(x, y, 1)^T$ .

Bem.: Für Punkte im  $\mathbb{R}^3$  gilt analoges.



## 2.4 Transformationen in der Ebene

### Homogene Koordinaten (cont.)

Dies erlaubt nun eine „neue“, einheitliche Darstellung unserer Transformationen:

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

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

$\underbrace{\phantom{x'}}$   
 $T(t_1, t_2)$



## 2.4 Transformationen in der Ebene

### Homogene Koordinaten (cont.)

- **Rotation** eines Punktes  $(x,y)^T$  um den Winkel  $\varphi$ :

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

- **Skalierung** eines Punktes  $(x,y)^T$  mit den Faktoren  $s_1$  und  $s_2$ :

$$\underbrace{\begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{S(s_1, s_2)} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 \cdot x \\ s_2 \cdot y \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix}$$



## 2.4 Transformationen in der Ebene

### Homogene Koordinaten (cont.)

- **Rotation** eines Punktes  $(x,y)^T$  um einen Punkt  $P_1 = (P_{1x}, P_{1y})^T$  um den Winkel  $\varphi$ :

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

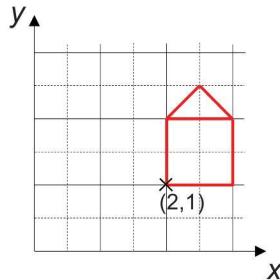
kurz:  $P' = T(P_{1x}, P_{1y}) \cdot R(\varphi) \cdot T(-P_{1x}, -P_{1y}) \cdot P$



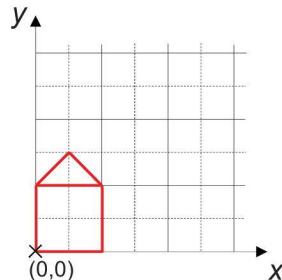
## 2.4 Transformationen in der Ebene

### Übung: zusammengesetzte Transformationen

- I) Führen Sie die Transformation  $R(45^\circ)T(3,1)$  am Haus aus mittels:
- Sichtweise A,
  - Matrixrechnung,
  - Sichtweise C.



- II) Führen Sie die Transformation  $R(120^\circ)T(2,2)R(30^\circ)T(2,1)$  am Haus aus mittels:
- Sichtweise A,
  - Sichtweise C.



- III) Ermitteln Sie allgemein die zusammengesetzten Matrizen für die Transformationen  $R(\varphi)T(a,b)$  und  $T(a,b)R(\varphi)$ .



## 2.5 Transformationen im Raum

### 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

$$\underbrace{\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{T(t_x, t_y, t_z)} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \\ z + t_z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$



## 2.5 Transformationen im Raum

### Skalierung

Eine Skalierung mit den Faktoren  $s_1$ ,  $s_2$  und  $s_3$  für die drei Achsenrichtungen hat die Gestalt

$$\underbrace{\begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{S(s_1, s_2, s_3)} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 \cdot x \\ s_2 \cdot y \\ s_3 \cdot z \\ 1 \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix}$$



## 2.5 Transformationen im Raum

### Rotation

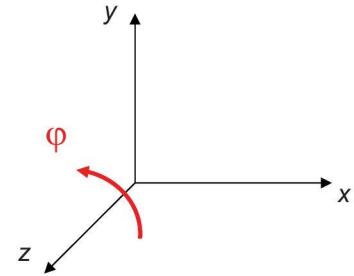
- alle Rotationen erfolgen im mathematisch positiven Sinn (d. h. gegen den Uhrzeigersinn)
- Der Betrachter „sitzt“ dabei auf der Rotationsachse und schaut in Richtung **Ursprung des Koordinatensystems.**
- Wir betrachten zuerst die Rotationen um die einzelnen Koordinatenachsen um den Winkel  $\varphi$ .  
→ Transformationsmatrizen  $R_x(\varphi)$ ,  $R_y(\varphi)$ ,  $R_z(\varphi)$
- Wir verwenden Sichtweise A



## 2.5 Transformationen im Raum

### Rotation um die z-Achse

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



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

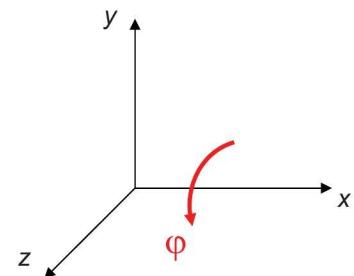
Beachte: Drehung um Winkel  $\varphi$  um z-Achse entspricht  
2D-Fall, wobei z-Koordinate konstant bleibt



## 2.5 Transformationen im Raum

### Rotation um die x-Achse

Die Rotation eines Punktes  $(x,y,z)^T$  um die x-Achse um den Winkel  $\varphi$  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: Hier bleibt x-Komponente konstant, so daß  
Einschränkung auf  $(y,z)$ -Ebene 2D-Fall entspricht

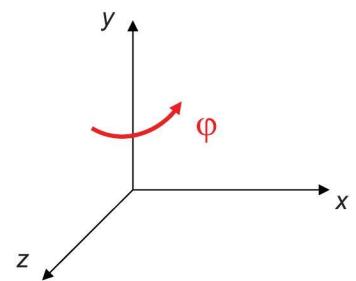


## 2.5 Transformationen im Raum

### Rotation um die y-Achse

Die Rotation eines Punktes  $(x,y,z)^T$  um die y-Achse um den Winkel  $\varphi$  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: Hier bleibt y-Komponente konstant, so daß Einschränkung auf  $(z,x)$ -Ebene 2D-Fall entspricht



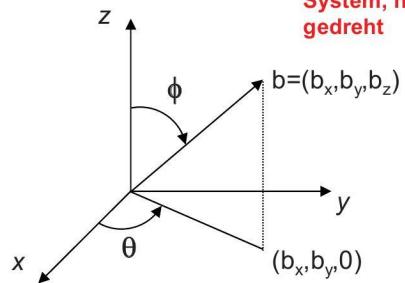
## 2.5 Transformationen im Raum

### Rotation um eine beliebige Achse

Jede Rotation um eine beliebige Achse kann aus Rotationen um die einzelnen Koordinatenachsen zusammengesetzt werden (→ Euler).

Wir entwickeln nun die Rotation  $R_G(\alpha)$  für die Drehung eines Punktes P um eine beliebig orientierte Achse G im Raum um einen Winkel  $\alpha$ .

Bem.: immer noch rechtshändiges System, hier nur gedreht



$$\begin{aligned} b_x &= \sin \phi \cos \theta \\ b_y &= \sin \phi \sin \theta \\ b_z &= \cos \phi \end{aligned}$$

Zuerst betrachten wir den Sonderfall, bei dem die Drehachse G durch den Ursprung geht und von einem Vektor b ( $b=(b_x, b_y, b_z)^T$  mit  $\|b\|=1$ ) generiert wird, also  $G: \lambda \cdot b$  ( $\lambda \in \mathbb{R}$ ).



## 2.5 Transformationen im Raum

### Rotation um eine beliebige Achse (cont.)

Gesucht sind nun die Koordinaten eines Punktes P nach einer Drehung um die Achse G um den Winkel  $\alpha$ .

Vorgehensweise:

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

(Bem.: Die Achse G soll nicht kollinear zur z-Achse verlaufen, sonst sind sowieso keine Hilfstransformationen durchzuführen)

Man geht in mehreren Teilschritten vor:



## 2.5 Transformationen im Raum

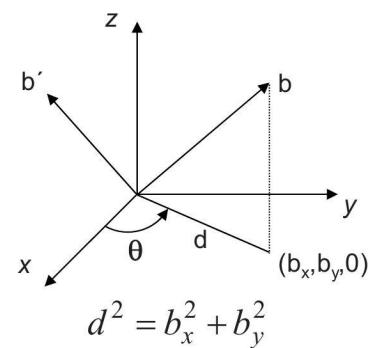
### Rotation um eine beliebige Achse (cont.)

Schritt 1:

Zunächst dreht man so, daß der Vektor b in die (z,x)-Ebene gedreht wird ( $b'$ ). Aus P entsteht dabei  $P'$  mit  $P' = R_z(-\theta) P$ , wobei

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

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



## 2.5 Transformationen im Raum

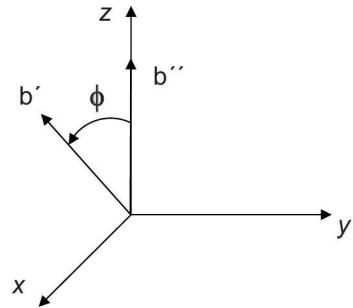
### Rotation um eine beliebige Achse (cont.)

Schritt 2:

Dann dreht man so, daß der Vektor  $b'$  mit der z-Achse zusammenfällt ( $b''$ ). Aus  $P'$  entsteht dabei  $P''$  mit  $P'' = R_y(-\phi) P'$ , wobei

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

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



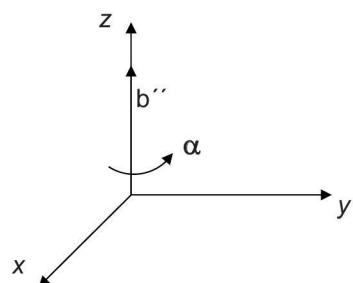
## 2.5 Transformationen im Raum

### Rotation um eine beliebige Achse (cont.)

Schritt 3:

Man dreht nun um den Winkel  $\alpha$  um die z-Achse. Aus  $P''$  entsteht dabei  $P'''$  mit  $P''' = R_z(\alpha) P''$ , wobei

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



## 2.5 Transformationen im Raum

### Rotation um eine beliebige Achse (cont.)

Schritte 4 und 5:

Zum Schluß werden die Rotationen aus den Schritten 1 und 2 in umgekehrter Reihenfolge rückgängig gemacht. Aus  $P'''$  entsteht dabei der gewünschte „gedrehte“ Punkt Q des ursprünglichen Punktes P mittels  $Q = R_z(\theta) R_y(\phi) P'''$ , wobei

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



## 2.5 Transformationen im Raum

### Rotation um eine beliebige Achse (cont.)

#### Ergebnis:

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

$$R_b(\alpha) = R_z(\theta) R_y(\phi) R_z(\alpha) R_y(-\phi) R_z(-\theta)$$

#### Allgemeiner Fall:

Ist die Drehachse eine allgemeine Gerade

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

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

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



## 2.5 Transformationen im Raum

### Einschub: Affine Transformationen

Alle bisherigen Transformationen (Translation, Rotation, Skalierung) sind Beispiele sogenannter affiner Transformationen.

Für eine affine Transformation  $f$  und zwei Punkte  $P$  und  $Q$  gilt immer:

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



## 2.5 Transformationen im Raum

### Einschub: Affine Transformationen (cont.)

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

Interpretation:

- das Bild einer Strecke (Strecke von  $Q$  nach  $P$ ) ist unter einer affinen Abbildung **wieder eine Strecke!**
- Teilverhältnisse - hier  $\lambda:(1-\lambda)$  - **bleiben unter einer affinen Abbildung erhalten!**
- Es genügt die Endpunkte  $Q$  und  $P$  auf der Strecke abzubilden. Zwischenpunkte erhält man durch **Interpolation von  $f(P)$  und  $f(Q)$ .**

*Bem.: Man beachte, daß unter affinen Abbildungen parallele Linien parallel bleiben.*



## 2.5 Transformationen im Raum

### Einschub: Affine Transformationen (cont.)

einige zusätzliche affine 2D-Transformationen:

Spiegelung an der Geraden  $y=x$ :

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Spiegelung an der x-Achse:

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Spiegelung an der y-Achse:

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Spiegelung am Ursprung:

$$\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$



## 2.6 Projektionen

Unter einer **Projektion** versteht man allgemein eine Abbildung aus einem Raum der Dimension  $n$  in einen Raum mit einer kleineren Dimension als  $n$ .

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 festgelegt. Der Schnittpunkt des Projektionsstrahls mit der Projektionsebene bestimmt den projizierten Raumpunkt.



## 2.6 Projektionen

Wir betrachten **Zentralprojektion** (**perspektivische Projektion**) und **Parallelprojektion** als Beispiele geometrisch planarer Projektionen.

Bei der Parallelprojektion liegt das Projektionszentrum in einem unendlich fernen Punkt.

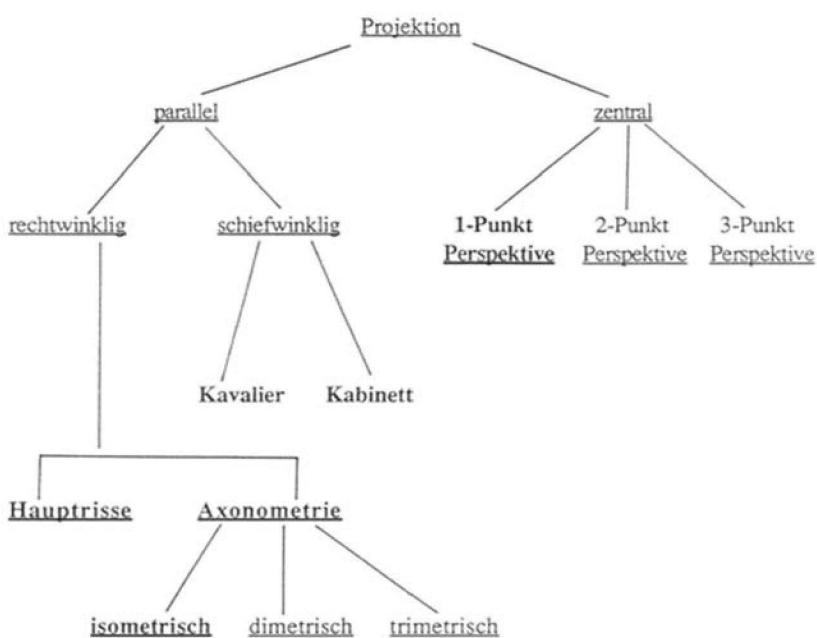
Im Rahmen der projektiven Geometrie stellt die Parallelprojektion einen Spezialfall der Zentralprojektion dar.

(Dies lässt sich z. B. bei der praktischen Umsetzung der Projektionen in Matrixschreibweise gewinnbringend anwenden!)



## 2.6 Projektionen

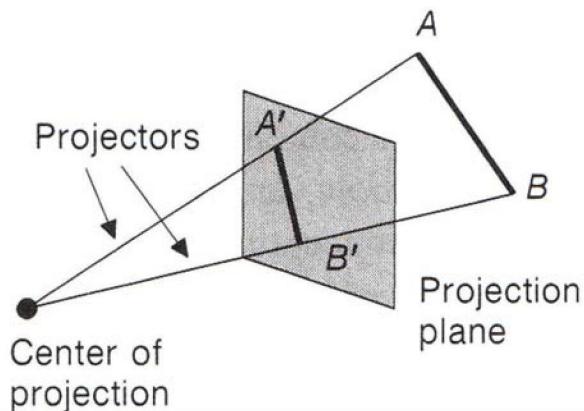
Klassifikation der gängigen Projektionsarten:



## 2.7 Perspektivische Projektion / Zentralprojektion

Bei der perspektivischen Projektion laufen alle Projektionsstrahlen 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.



## 2.7 Perspektivische Projektion / Zentralprojektion

Eigenschaften:

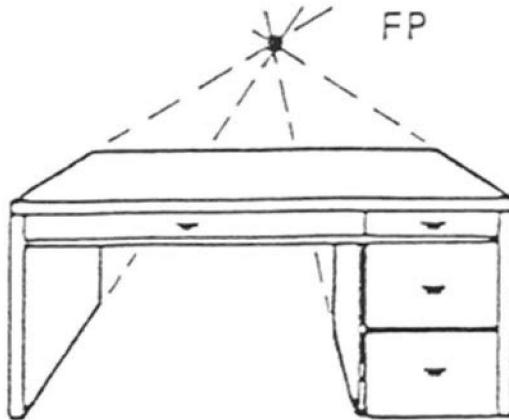
- Je zwei parallele Geraden, die nicht parallel zur Projektionsebene sind, treffen sich in einem Punkt, dem Fluchtpunkt.
- Es gibt unendlich viele Fluchtpunkte, je einen pro Richtung nicht parallel zur Projektionsebene.
- Hervorgehoben werden die Fluchtpunkte der Hauptachsen: z. B. treffen sich Geraden, die parallel zur x-Achse verlaufen, im x-Fluchtpunkt

Perspektivische Projektionen werden nach der Anzahl der Hauptachsen, welche die Projektionsebene schneiden, eingeteilt. So entstehen 1-Punkt-, 2-Punkt- und 3-Punkt-Perspektiven.



## 2.7 Perspektivische Projektion / Zentralprojektion

Beispiel:

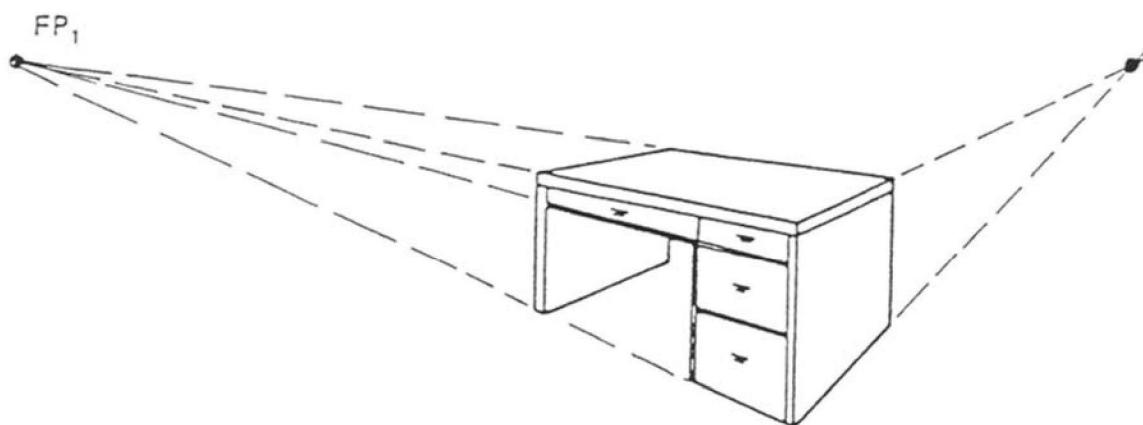


1-Punkt-Perspektive



## 2.7 Perspektivische Projektion / Zentralprojektion

Beispiel:



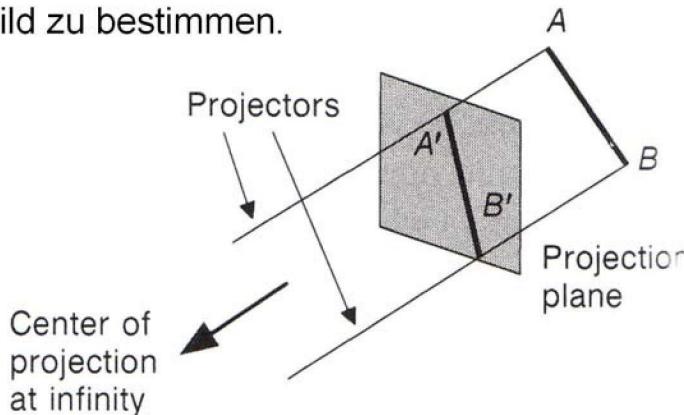
2-Punkt-Perspektive



## 2.8 Parallelprojektion

Bei der Parallelprojektion ist das Projektionszentrum im Unendlichen, alle Projektionsstrahlen verlaufen parallel in einer Richtung.

Die Parallelprojektion ist zwar weniger realistisch als die Zentralprojektion, es ist aber leichter, exakte Maße aus dem projizierten Bild zu bestimmen.



## 2.8 Parallelprojektion

Die Projektionsstrahlen können bei Parallelprojektionen gegen die Projektionsebene senkrecht ( $\rightarrow$  orthographische Projektionen) oder schiefwinklige ( $\rightarrow$  schiefe Projektionen) stehen.

### Rechtwinklige / senkrechte / orthographische Projektion

Hier fällt die Projektionsrichtung mit der Ebenennormalen zusammen. Man unterscheidet Hauptrisse und Axonometrie.

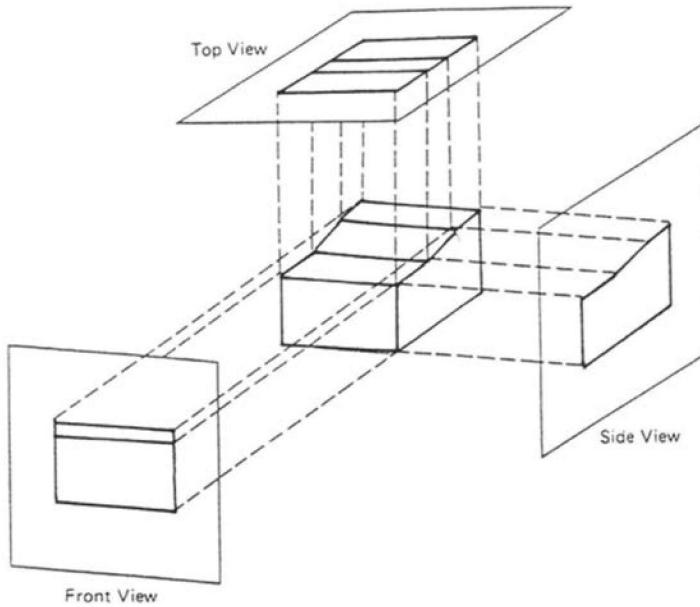
Bei den sogenannten **Hauptrissen** - Grundriss (top view), Aufriss (front view), Kreuzriß (side view) - schneidet die Projektionsebene nur eine Hauptachse. Die Normale der Projektionsebene ist also parallel zu einer der Hauptachsen.



## 2.8 Parallelprojektion

### Rechtwinklige / senkrechte / orthographische Projektion (cont.)

Beispiel:



## 2.8 Parallelprojektion

### Rechtwinklige / senkrechte / orthographische Projektion (cont.)

#### Axonometrie

Bei der **Axonometrie** ist die Normale der Projektionsebene nicht normal zu einer der Koordinatenachsen.

- Hier werden parallele Linien zwar auf parallele Linien abgebildet, Winkel bleiben allerdings nicht erhalten.
- Abstände können längs der Hauptachsen gemessen werden, allerdings i. a. in jeweils einem anderen Maßstab.

Im häufigsten Fall der **isometrischen Axonometrie** bildet die Projektionsebene mit allen Hauptachsen den gleichen Winkel. Hier hat man eine gleichmäßige Verkürzung aller Koordinatenachsen.



## 2.8 Parallelprojektion

### Rechtwinklige / senkrechte / orthographische Projektion (cont.)

#### Axonometrie (cont.)

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 die Projektionsebene mit jeder Achse einen anderen Winkel, die Skalierungen sind in den drei Achsenrichtungen verschieden.



## 2.8 Parallelprojektion

### Schiefe / Schiefwinklige Parallelprojektionen

**Schiefe Parallelprojektionen** entstehen, wenn sich die Projektionsrichtung von der Projektionsebenennormalen unterscheidet.

Die beiden gebräuchlichsten schiefen Parallelprojektionen sind die sogenannte **Kavalier-** und die sog. **Kabinettpunktprojektion**.

#### Kavalierprojektion

Der Winkel zwischen der Projektionsrichtung und der Projektionsebenennormalen beträgt  $45^\circ$ .

#### Kabinettpunktprojektion / Militärprojektion

Der Winkel zwischen der Projektionsrichtung und der Projektionsebenennormalen beträgt  $\arctan 2 = 63.4^\circ$ .



## 2.9 Zentralprojektion - Umsetzung

Die praktische Umsetzung der perspektivischen Projektion erfolgt je nach Anwendung in unterschiedlichsten Konfigurationen, die mittels geeigneter Transformationen des Koordinatensystems erreicht werden können.

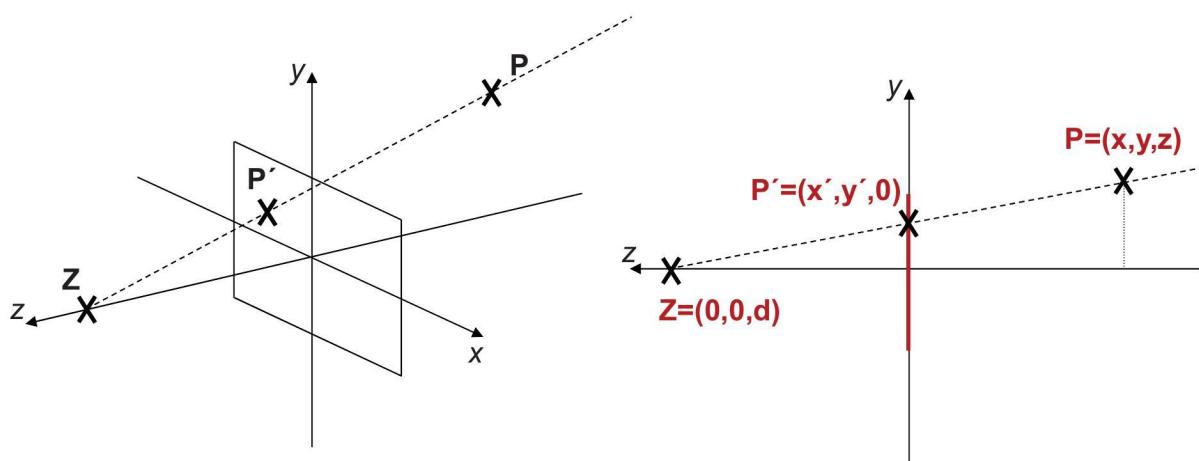
Exemplarisch wählen wir folgendes Setup:

- das Projektionszentrum  $Z$  und der Augpunkt fallen zusammen und liegen auf der positiven  $z$ -Achse mit Abstand  $d > 0$  zum Ursprung, also  $Z = (0,0,d)$
- die Blickrichtung ist die negative  $z$ -Achse
- die Bildebene liegt in der  $(x,y)$ -Ebene



## 2.9 Zentralprojektion - Umsetzung

Setup:



## 2.9 Zentralprojektion - Umsetzung

Aus dem Strahlensatz folgt:  $\frac{y'}{d} = \frac{y}{d-z}$  und  $\frac{x'}{d} = \frac{x}{d-z}$

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

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

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

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



## 2.9 Zentralprojektion - Umsetzung

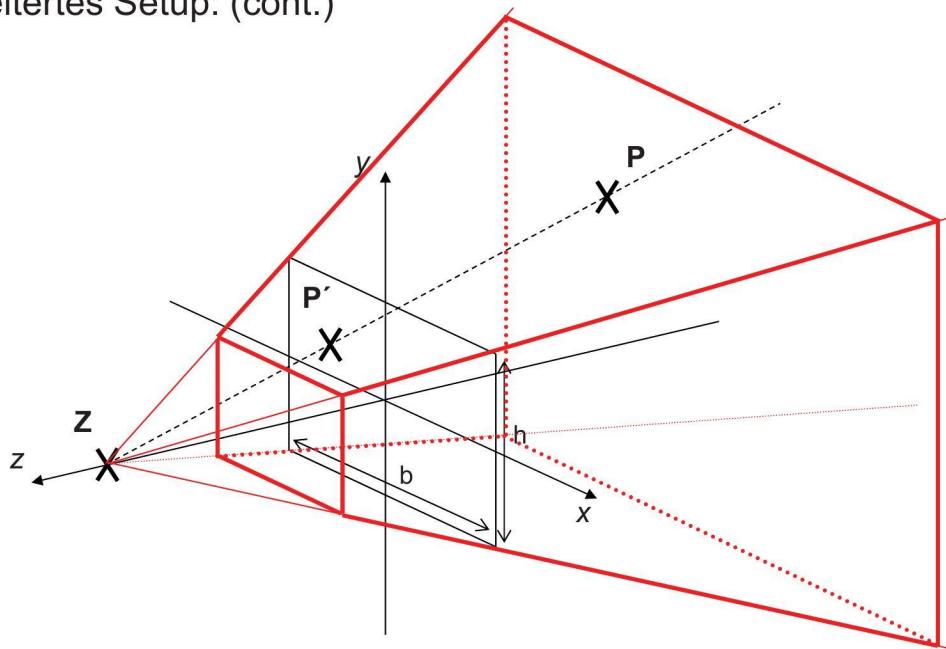
Erweitertes Setup:

- in der Bildebene wird ein Sichtfenster (view window) spezifiziert (Breite, Höhe, Verhältnis Breite zu Höhe);  
das Sichtfenster 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 (→ Clipping)



## 2.9 Zentralprojektion - Umsetzung

Erweitertes Setup: (cont.)



## 2.10 Windowing

Begriffe:

- Window:** - auch „view window“ ( $\rightarrow$  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

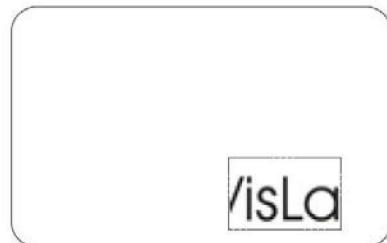
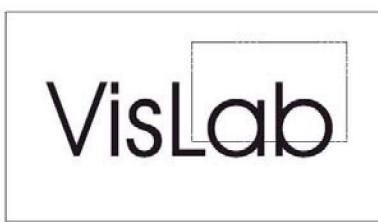
Bem.: I. d. R. sind sowohl Window als auch Viewport an den Koordinatenachsen ausgerichtete rechteckige Gebiete.

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



## 2.10 Windowing

---

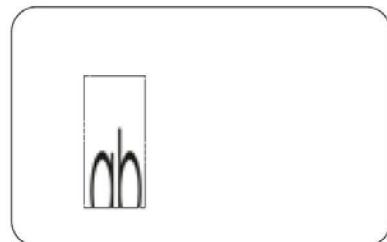


Verschiedene Fenster, dieselben Viewports



## 2.10 Windowing

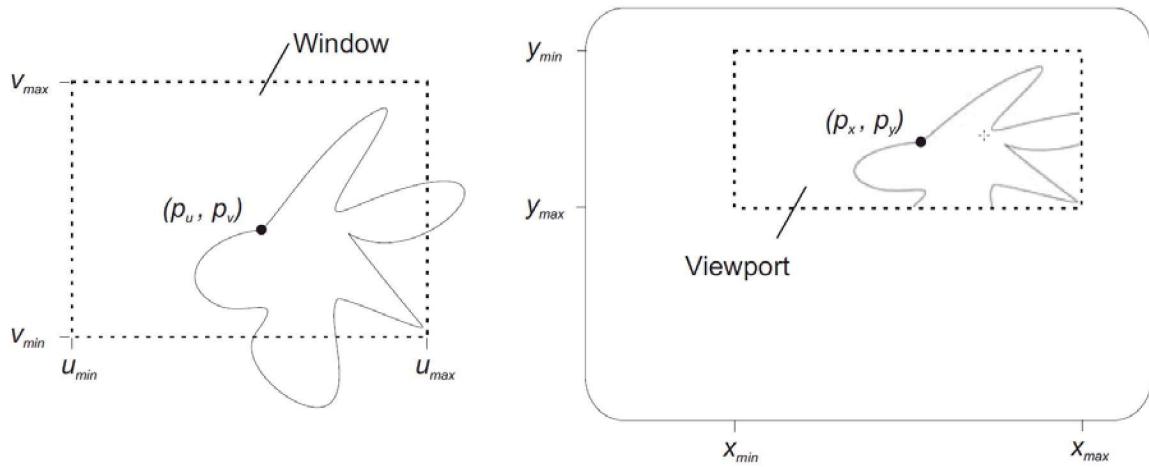
---



Dieselben Fenster, verschiedene Viewports



## 2.10 Windowing



$p_u, p_v$  Punktkoordinaten im Bild-Koordinatensystem

$p_x, p_y$  Punktkoordinaten im Geräte-Koordinatensystem

$u_{min}, v_{min}, u_{max}, v_{max}$  Koordinaten des Windows (Bild-Koordinaten)

$x_{min}, y_{min}, x_{max}, y_{max}$  Koordinaten des Viewports (Geräte-Koordinaten)



## 2.10 Windowing

Transformation in 3 Schritten:

- 1) Translation (des Fensters) in den Koordinatenursprung des Bild-Koordinatensystems

$$p'_u = p_u - u_{min}, \quad p'_v = p_v - v_{min}.$$

- 2) Skalierung des Fensterbereichs auf die Größe des Viewports

$$p''_u = \frac{x_{max} - x_{min}}{u_{max} - u_{min}} p'_u, \quad p''_v = \frac{y_{max} - y_{min}}{v_{max} - v_{min}} p'_v.$$

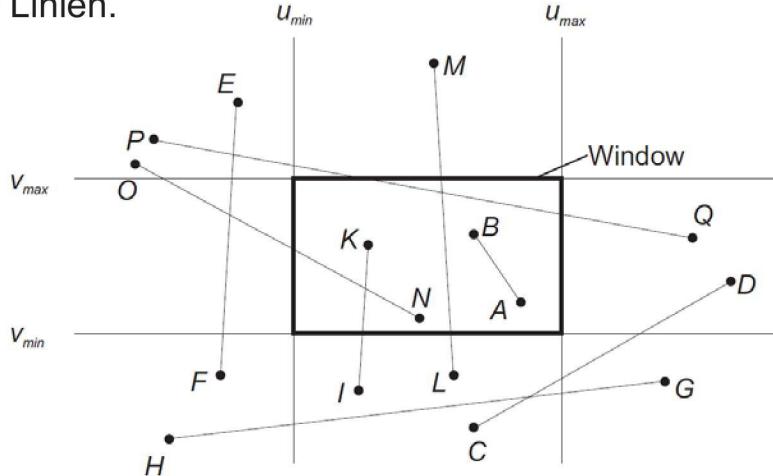
- 3) Translation des Viewports an die richtige Stelle im Geräte-Koordinatensystem

$$p_x = p''_u + x_{min}, \quad p_y = -p''_v + y_{max}.$$



## 2.11 Clipping von Linien

Sollen Objekte in der Bildebene innerhalb eines Fensters dargestellt werden, so wird ein Verfahren benötigt, mit dem alle außerhalb des Fensters liegenden Objektteile abgeschnitten werden können (Clipping am Fensterrand). Wir betrachten zuerst das Clipping von Linien.



## 2.11 Clipping von Linien

Offensichtlich gibt es 3 Fälle, von denen zwei trivial sind:

- beide Endpunkte innerhalb des Fensters  
→ Linie **zeichnen**
- beide Endpunkte oberhalb oder unterhalb oder links oder rechts des Fensters  
→ Linie **nicht zeichnen**
- Ansonsten müssen die Schnittpunkte der Linie mit dem Fensterrand anhand der Geradengleichungen berechnet und daraus die sichtbare Strecke bestimmt werden.



## 2.11 Clipping von Linien

### Cohen-Sutherland Line-Clipping Algorithmus

Der Algorithmus basiert auf einem leistungsfähigen Verfahren zur Bestimmung der Kategorie einer Linie (innerhalb, außerhalb, schneidend).

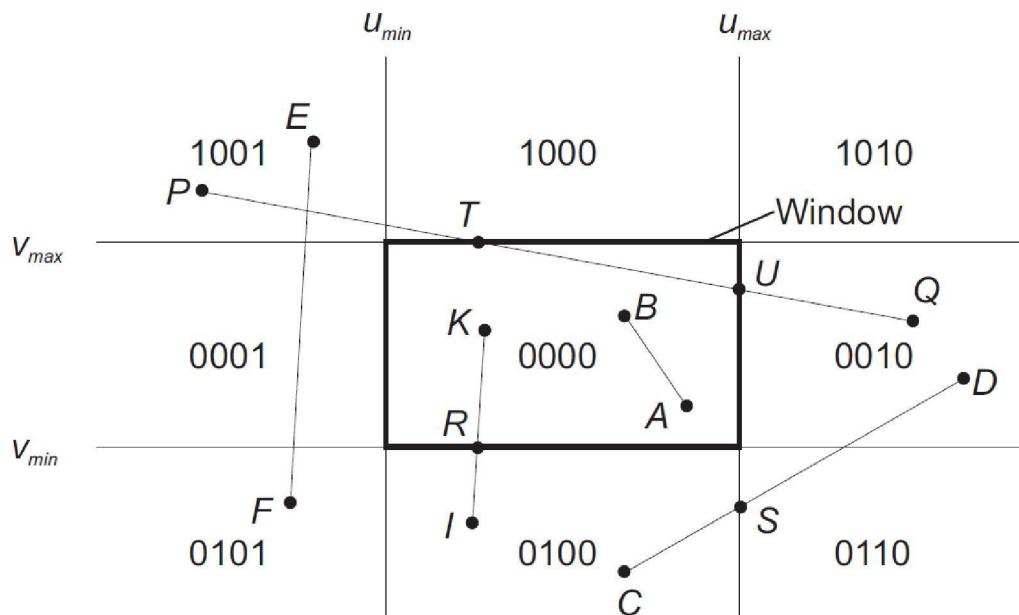
Es sei ein Fenster  $(u_{\min}, v_{\min}, u_{\max}, v_{\max})$  gegeben, dessen begrenzende Geraden die Bildebene in neun Regionen unterteilen. Jeder Region ist ein eindeutiger 4-Bit-Code zugeordnet, der Auskunft über deren Lage in Bezug auf das Fenster gibt:

$u < u_{\min}$	Bit 0 gesetzt	links des Fensters
$u > u_{\max}$	Bit 1 gesetzt	rechts des Fensters
$v < v_{\min}$	Bit 2 gesetzt	unterhalb des Fensters
$v > v_{\max}$	Bit 3 gesetzt	oberhalb des Fensters



## 2.11 Clipping von Linien

### Cohen-Sutherland Line-Clipping Algorithmus (cont.)



## 2.11 Clipping von Linien

### Cohen-Sutherland Line-Clipping Algorithmus (cont.)

Für die Endpunkte einer Linie bestimmt man nun die 4-Bit-Codes der Regionen, in denen sie sich befinden. Dann gilt:

- Die Linie liegt **vollständig außerhalb des Fensters**, falls der Durchschnitt (**AND**-Verknüpfung) der Codes beider Endpunkte von Null verschieden ist.
- Die Linie liegt **komplett im Fenster**, wenn beide Endpunkte den 4-Bit-Code 0000 besitzen (**OR**-Verknüpfung ist Null).

In allen anderen Fällen wird die Linie nacheinander mit den das Fenster begrenzenden Geraden geschnitten und jeweils in zwei Teile zerlegt, die gemäß obiger Vorgehensweise kategorisiert werden. Der dabei außerhalb des Fensters liegende Teil kann sofort eliminiert werden.

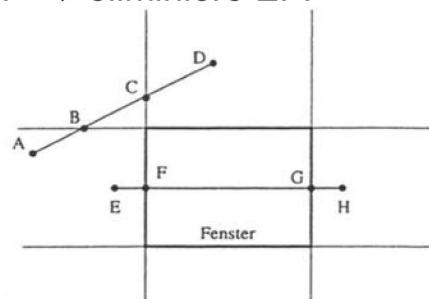


## 2.11 Clipping von Linien

### Cohen-Sutherland Line-Clipping Algorithmus (cont.)

Beispiele nichttrivialer Fälle:

- **Linie AD:** Codes 0001 und 1000 → Schnittberechnung  
Schnitt mit linker Fenstergrenze liefert C → eliminiere AC  
Punkte C und D liegen oberhalb des Fensters → eliminiere CD
- **Linie EH:** Codes 0001 und 0010 → Schnittberechnung  
Schnitt mit linker Fenstergrenze liefert F → eliminiere EF.  
Für FH ist eine Schnittberechnung mit der rechten Fenstergrenze notwendig,  
die den Punkt G liefert  
→ eliminiere GH  
Punkte F und G liegen innerhalb des Fensters → FG wird gezeichnet



## 2.12 Clipping von Polygonen

Polygone sind in der Computergrafik als Begrenzung von Flächen von Bedeutung.

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

Ein einfacher Algorithmus würde jede Polygonseite gegen das Fenster clippen.

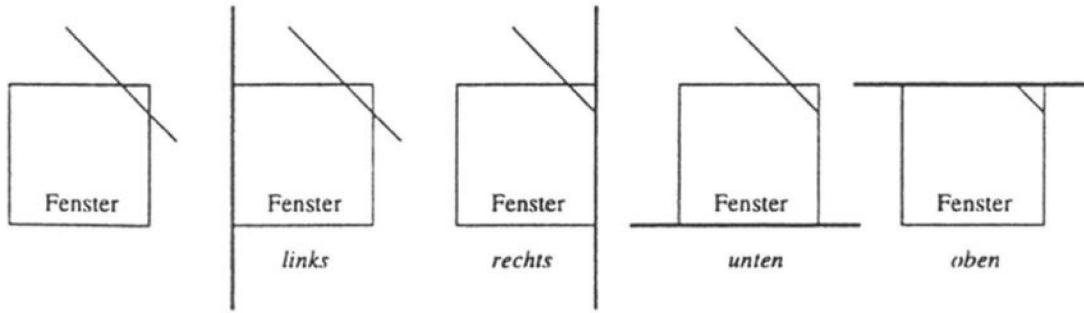


Abb.: Polygon-Clipping durch n-faches Vektor-Clipping



## 2.12 Clipping von Polygonen

Wenn eine Seite das Fenster verlässt, wird der Austrittspunkt mit dem Wiedereintritt verbunden, wobei z. B. die Ecken zu Problemen führen können.

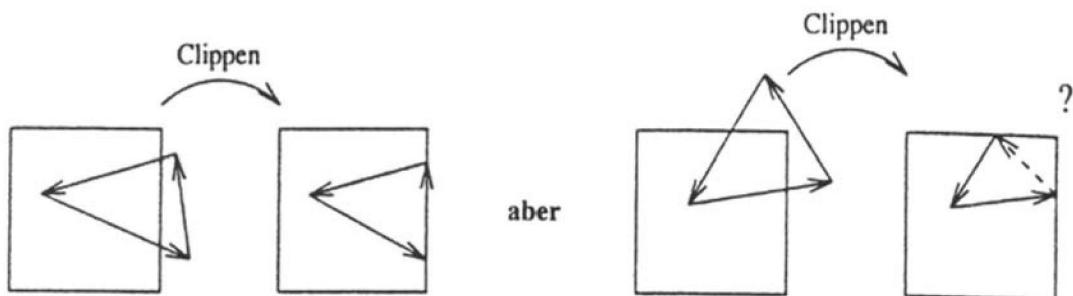


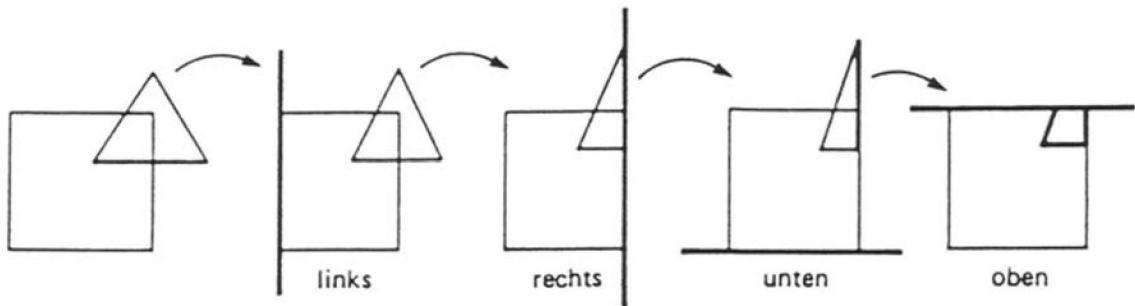
Abb.: Einfügen von Fenstergrenzen beim Polygon-Clipping



## 2.12 Clipping von Polygonen

### Sutherland-Hodgman Polygon-Clipping Algorithmus

Statt jede Polygonseite gegen alle 4 Fensterseiten zu clippen, führt das vollständige Clippen des Polygons gegen eine Fensterseite nach der anderen zum Ziel.

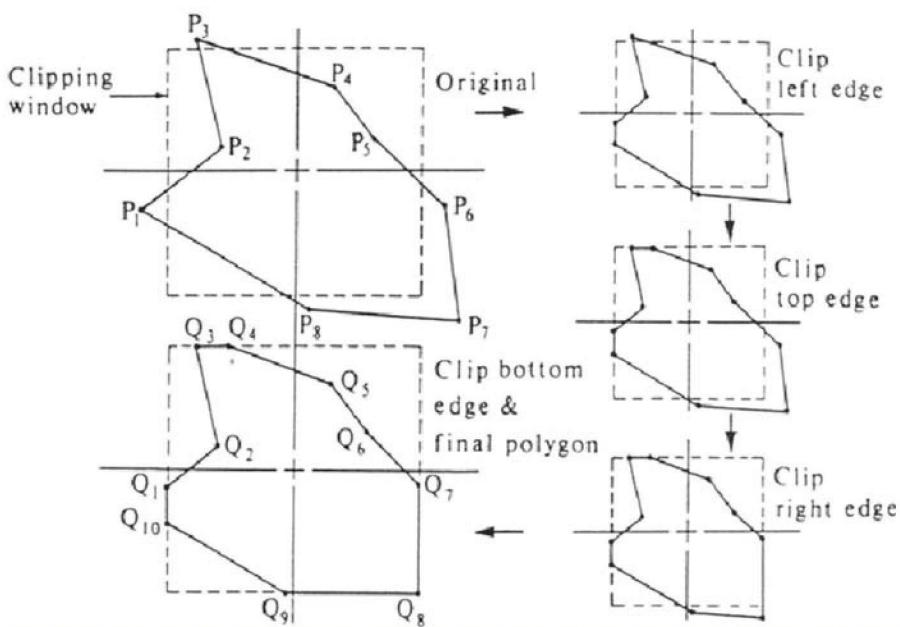


Die Zwischenergebnisse müssen gespeichert werden.



## 2.12 Clipping von Polygonen

### Sutherland-Hodgman Polygon-Clipping Algorithmus (cont.)



## 2.13 Clipping im 3D-Raum

Statt nach der Projektion in die zweidimensionale Bildebene kann das Clipping auch bereits im dreidimensionalen Objektraum durchgeführt werden. Dies hat den Vorteil, dass nur die tatsächlich sichtbaren Objekte transformiert werden müssen.

Das Sichtvolumen begrenzt den Teil des Raums, der dargestellt werden soll. Anhand der Ebenengleichungen der Randflächen des Sichtvolumens kann bestimmt werden, ob ein gegebener Punkt (Linie, Objekt ...) außerhalb oder innerhalb des Sichtvolumens liegt.

Die vorgestellten zweidimensionalen Clipping-Verfahren können auch auf den dreidimensionalen Fall übertragen werden.



## 2.13 Clipping im 3D-Raum

