

3.3 Geometrische Transformationen

3. Rendering

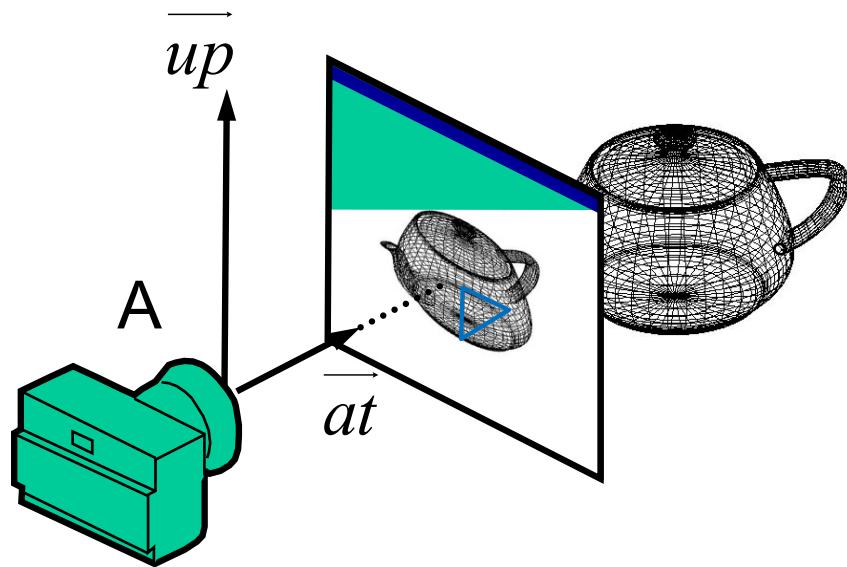


- In der Computergraphik gibt es 2 grosse Ansätze zur Bilderzeugung (rendering)
 - Projektion und Rasterisierung
 - Raytracing

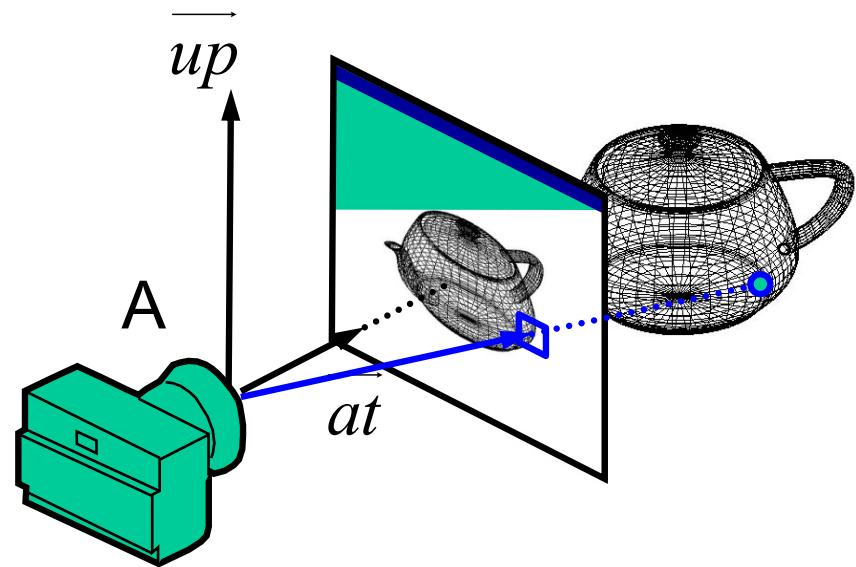
3. Rendering



Projektion/Rasterisierung



Ray Tracing



Die Polygone werden der Reihe nach rasterisiert

Das Bild wird durch Sehstrahlen abgetastet

3.3 Geometrische Transformationen



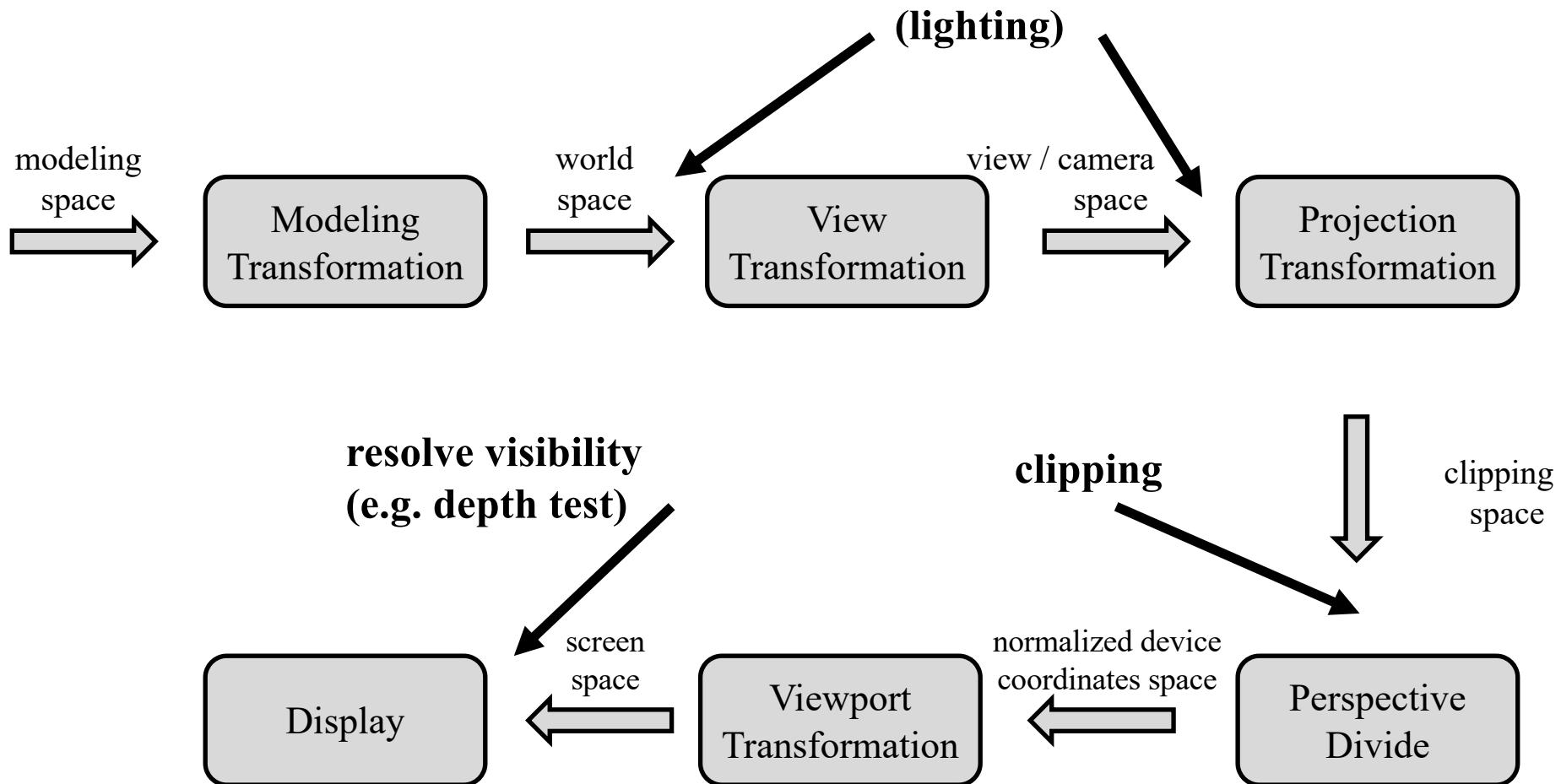
- Die Darstellungselemente durchlaufen mehrere Koordinatensysteme von ihrer Definition bis zur graphischen Ausgabe.
- Dieser Durchlauf wird als Ausgabepipeline (Viewing-Pipeline oder Grafikpipeline) bezeichnet und durch Koordinatentransformationen realisiert.
- Dazu gehört auch das Setzen der Kameraparameter.

3.3.1 Die Viewing-Pipeline

3.3.1 Die Viewing-Pipeline



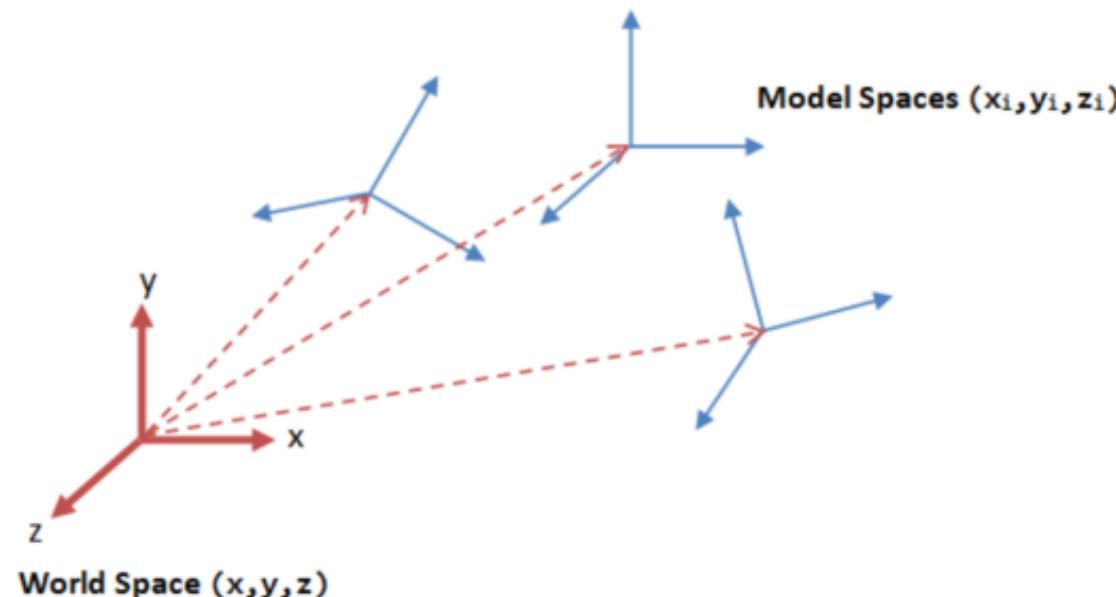
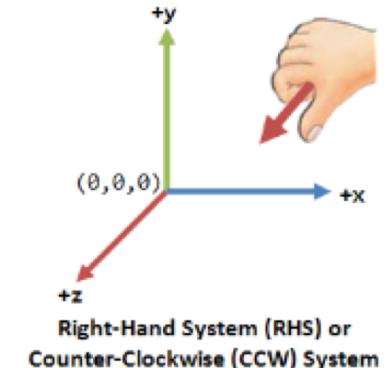
Übliche Koordinatensysteme der 3D-Computergrafik



3.3.1 Die Viewing-Pipeline



- Modeling Transformation
 - Platziere versch. Objekte in Weltkoordinatensystem
 - Szenenzusammensetzung, Komposition
 - Meist beschrieben durch Hintereinanderausführung von Verschiebungen, Drehungen, Skalierungen oder Scherungen



3.3.1 Die Viewing-Pipeline

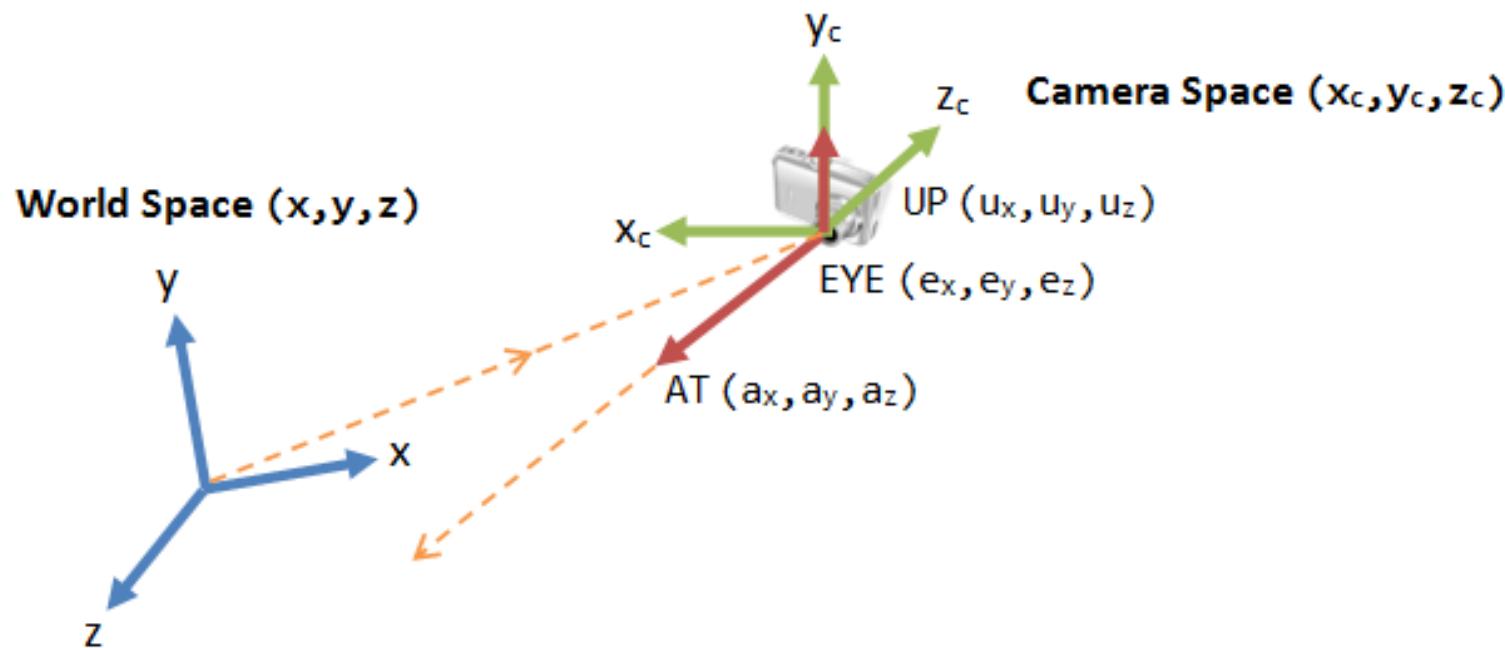


- Die Ansichtstransformation nutzt als Grundmodell das synthetische Kameramodell einer Lochkamera.
- Sie läuft in 2 Stufen ab:
 - View Transformation (Ansichtsorientierungstransformation)
überführt die Darstellungselemente in ein neues Bezugssystem, das Ansichtskoordinatensystem (camera space).
 - Projection Transformation (Ansichtsabbildungstransformation)
ist eine projektive Abbildung und überführt die Ansichtskoordinaten in normierte Projektionskoordinaten (clipping space). Anschließend kommt perspektivische Division (normalized device coordinates space).

3.3.1 Die Viewing-Pipeline



- View Transformation
- Dient der Überführung aller darzustellenden Objekte in ein gemeinsames Bezugssystem mit der Kamera im Ursprung und einer entsprechenden Orientierung (Blickrichtung ist neg. z-Achse)



3.3.1 Die Viewing-Pipeline



- View Transformation
- Es wird wie folgt festgelegt:
 - Achsen
 - \vec{Z}_C negierte Blickrichtung (kommt auf Kamera zu)
 - \vec{Y}_C „Up“-Vektor, gibt „Kopfneigung“ der Kamera an
 - \vec{X}_C zu \vec{Z}_C und \vec{Y}_C orthogonal, so dass ein Rechtssystem entsteht.

3.3.1 Die Viewing-Pipeline

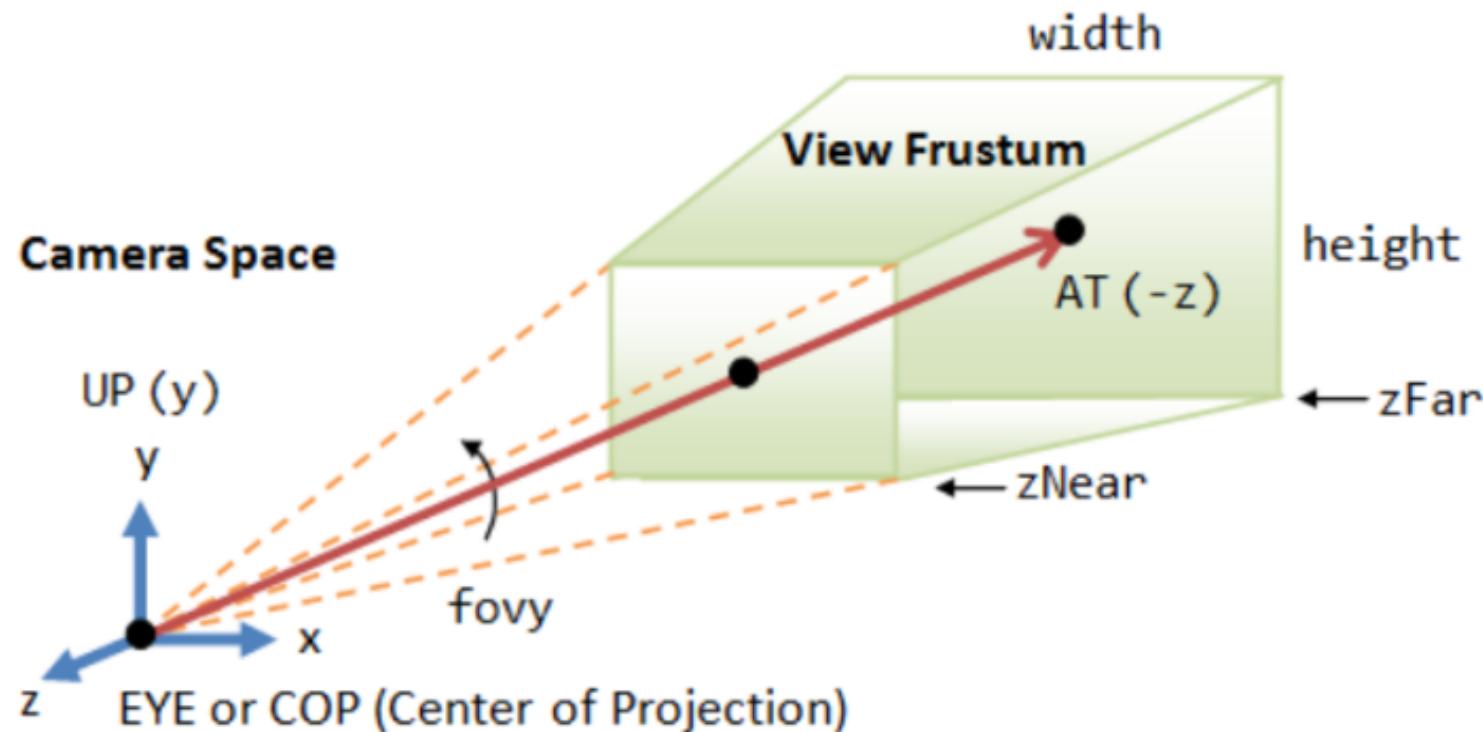


- View Transformation
- Die Festlegung durch den Nutzer erfolgt meist indirekt in intuitiver Weise durch
 - Punkt EYE: Position der Kamera
 - Punkt AT: Blickziel / Fokuspunkt der Kamera
 - Vektor UP: Zeigt nach oben
- Beispiel
 - `Matrix4x4 view = getLookAt(EYE,AT,UP)`
- “view” überführt von world space in camera space

3.3.1 Die Viewing-Pipeline



- Projection Transformation
- Sie bildet das Ansichtsvolumen in einen quaderförmigen Projektionsdarstellungsraum (View Frustum) ab.



3.3.1 Die Viewing-Pipeline

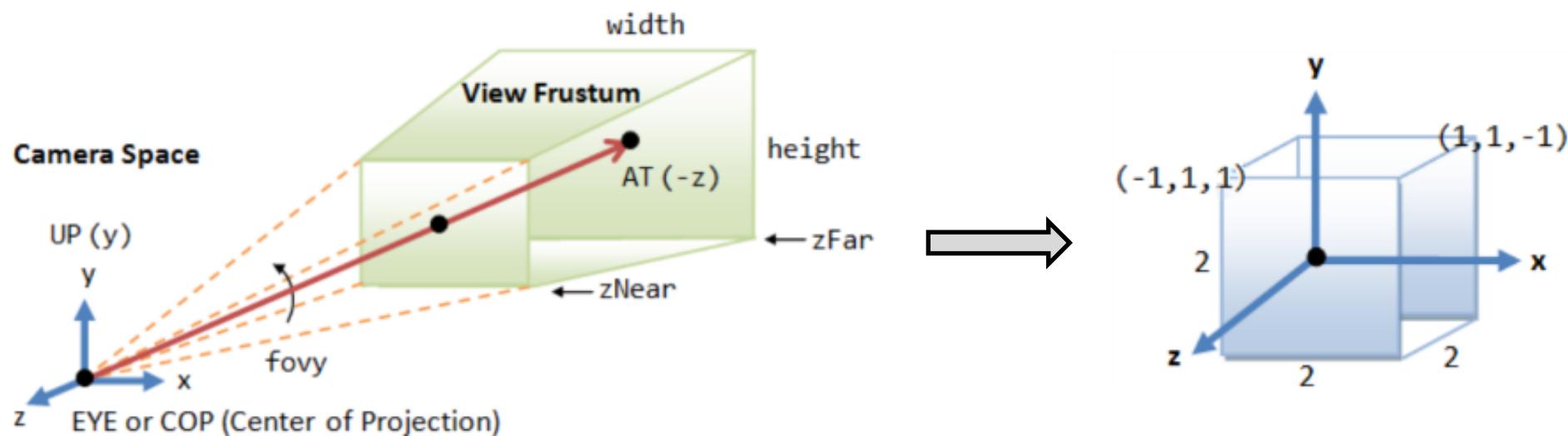


- Projection Transformation
- Für die Spezifikation der perspektivischen Projektionsparameter sind folgende zwei Vorgehensweisen üblich:
 - `Matrix4x4 proj = getPerspective(fovy, aspectRatio=width/height, zNear, zFar)`
- Oder die direkte Beschreibung des Frustums
 - `Matrix4x4 proj = getFrustum(xLeft, xRight, yBottom, yTop, zNear, zFar)`

3.3.1 Die Viewing-Pipeline



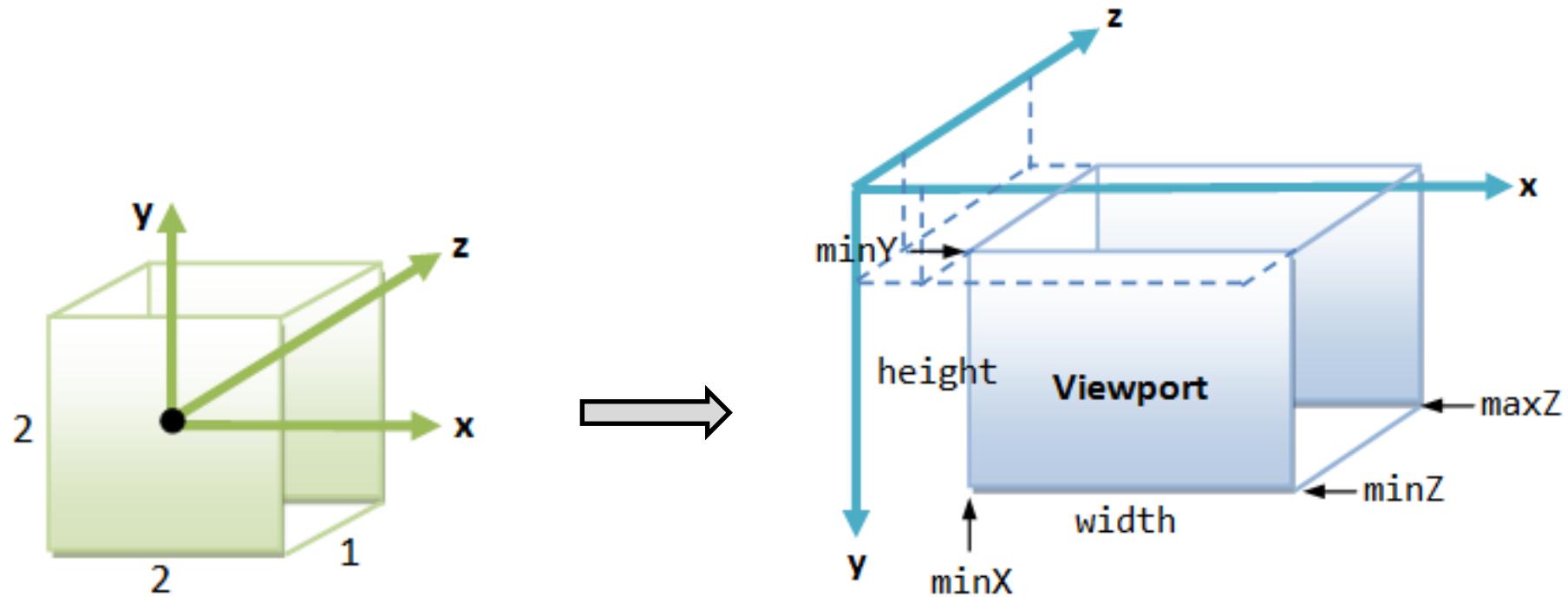
- Perspective Projection Transformation
- Nach der Perspective Division (und des Verwerfens nicht-sichtbarer Primitive) liegen normierte Projektionskoordinaten (normalized device coordinates space) vor.



3.3.1 Die Viewing-Pipeline



- Viewport Transformation
- Auswahl des Viewports auf dem Bildschirm
- Dann Skalierung und Parallelprojektion in Richtung der z-Achse



3.3.1 Die Viewing-Pipeline



- Rasterisierung



- Dazu später mehr...

3.3.2 Transformationen

3.3.2 Transformationen



- Standardtransformationen in graphischen Systemen sind:
Translation, Skalierung, Rotation und Scherung.
- **Sie sind wesentliche Voraussetzungen zur:**
 - Beschreibung und Realisierung von Objektmanipulationen,
 - Realisierung der Ausgabe und Arbeit in unterschiedlichen Koordinatensystemen,
 - Realisierung der Eingabe,
 - Durchführung von Maßstabsänderungen und
 - Beschreibung von Animationen.

3.3.2 Transformationen



- **Affine und Projektive Abbildungen**
- Geometrische Transformationen in der Computergraphik werden durch **affine** und **projektive** Abbildungen beschrieben.

3.3.2 Transformationen



- **Affine und Projektive Abbildungen**
- **Zwei Definitionen für affine Abbildungen:**
 - 1) Eine affine Abbildung f ist eine Abbildung von $\mathbb{R}^{2/3}$ in $\mathbb{R}^{2/3}$, die
 - Geraden in Geraden überführt
 - das Teilverhältnis (ratio) von beliebigen 3 Punkten auf einer beliebigen Geraden erhält.
 - 2) Eine affine Abbildung f ist eine Abbildung von $\mathbb{R}^{2/3}$ in $\mathbb{R}^{2/3}$, die aus der Hintereinanderausführung von endlich vielen Translationen, Drehungen, Skalierungen und Scherungen besteht.

3.3.2 Transformationen

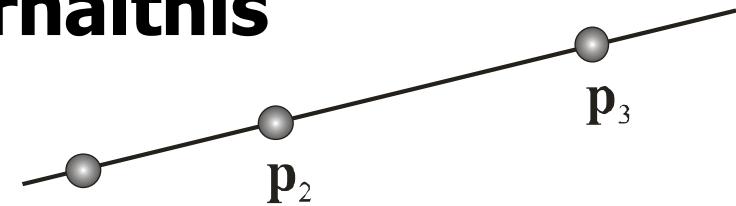


- **Affine und Projektive Abbildungen**
- **Definitionen für projektive Abbildung:**
 - 1) Eine projektive Abbildung f ist eine Abbildung von $\mathbb{R}^{2/3}$ in $\mathbb{R}^{2/3}$, die
 - Geraden in Geraden überführt (im erweiterten euklidischen Raum)
 - das Doppelverhältnis (cross ratio) von beliebigen 4 Punkten auf einer beliebigen Geraden erhält.

3.3.2 Transformationen

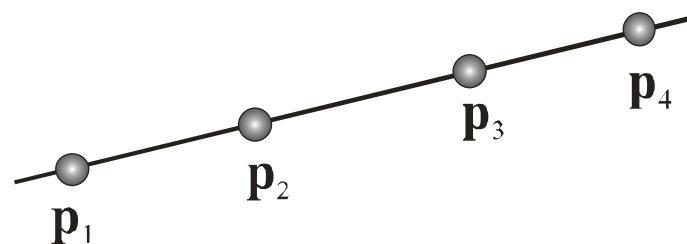


- **Teilverhältnis und Doppelverhältnis**



$$\text{ratio}(p_1, p_2, p_3) = t / (1-t) \Leftrightarrow p_2 = \frac{p_1}{(1-t)} + t p_3$$

$$\text{cr}(p_1, p_2, p_3, p_4) = \text{ratio}(p_1, p_2, p_4) / \text{ratio}(p_1, p_3, p_4)$$



3.3.2 Transformationen



- **Erweiterter euklidischer Raum**
= „euklidischer Raum + Punkte im Unendlichen“
- Punkt im Unendlichen (unendlich ferner Punkt): Klasse paralleler Geraden
- 2D: Unendlich ferne Gerade: Menge aller unendlich fernen Punkte
- 3D: unendlich ferne Ebene: Menge aller unendlich fernen Punkte

3.3.3 Beschreibung von Transformationen mit Matrizen

3.3.3 Beschreibung von Transformationen mit Matrizen



- **Voraussetzung:**

- In der Computergraphik werden Eckpunkte transformiert, die als Ortsvektoren im 2- oder 3-dim. euklidischen Raum gegeben sind.
- Wir gehen von der Spaltenschreibweise aus mit:

$$\mathbf{p} = (x, y)^T \text{ bzw. } \mathbf{p} = (x, y, z)^T .$$

- **Translation**

Hierbei wird ein Punkt \mathbf{p} um dx in x -Richtung und um dy in y -Richtung bzw. im 3D-Fall zusätzlich um dz in z -Richtung verschoben.

3.3.3 Beschreibung von Transformationen mit Matrizen



Damit ergibt sich für den 2D-Fall:

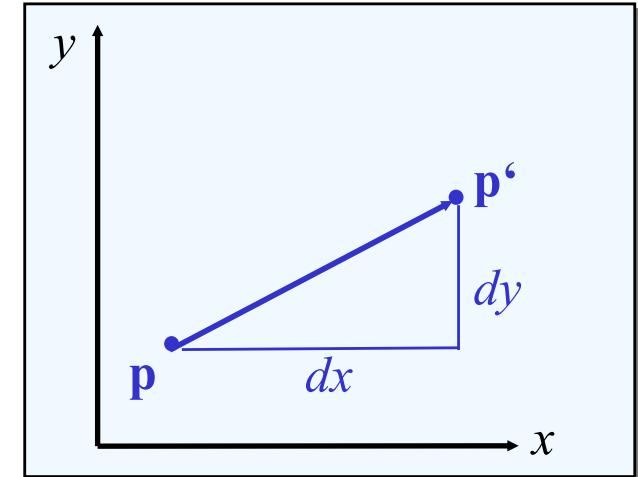
$$\mathbf{p}' = (x + dx, y + dy)^T \text{ oder}$$

$$\mathbf{p}' = \mathbf{p} + \mathbf{t} \quad \text{mit} \quad \mathbf{t} = (dx, dy)^T$$

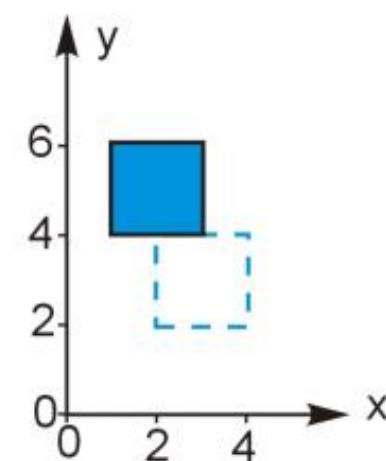
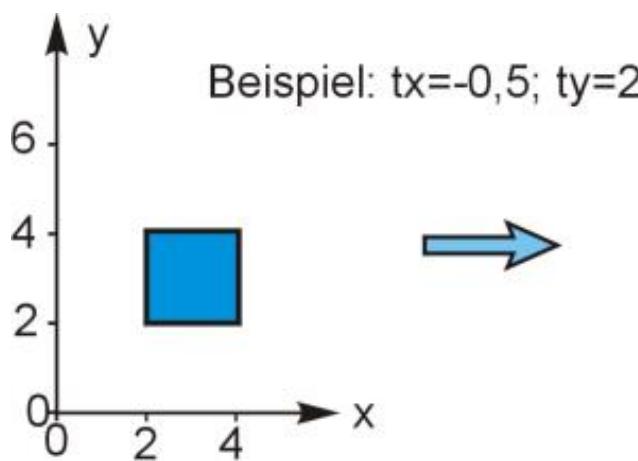
Analog zu 2D gilt für den 3D-Fall:

$$\mathbf{p}' = (x + dx, y + dy, z + dz)^T \text{ bzw.}$$

$$\mathbf{p}' = \mathbf{p} + \mathbf{t} \quad \text{mit} \quad \mathbf{t} = (dx, dy, dz)^T$$



2D-Translation von \mathbf{p} in \mathbf{p}'



3.3.3 Beschreibung von Transformationen mit Matrizen



- **Skalierung**

Hierbei wird ein Punkt \mathbf{p} bezogen auf den Koordinatenursprung in Richtung der x -Achse um den Faktor s_x und in Richtung der y -Achse um den Faktor s_y bzw. im 3D-Fall zusätzlich um den Faktor s_z in z -Richtung gestaucht oder gestreckt.

- Damit ergibt sich für den 2D-Fall:

$$\mathbf{p}' = (s_x * x, s_y * y)^T \text{ oder}$$

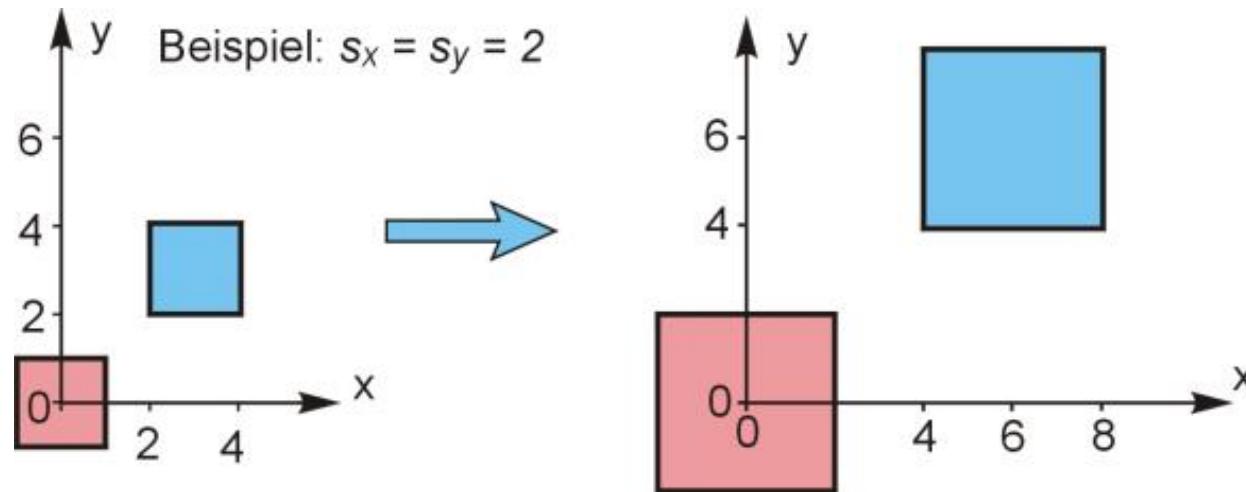
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

- Analog zu 2D gilt für den 3D-Fall:

$$\mathbf{p}' = (s_x * x, s_y * y, s_z * z)^T \text{ bzw.}$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

3.3.3 Beschreibung von Transformationen mit Matrizen



Beispiel einer Skalierung im 2D-Fall

3.3.3 Beschreibung von Transformationen mit Matrizen



- **Rotation**

2D-Fall: Ein Punkt \mathbf{p} wird mit dem Winkel ϕ um den Ursprung entgegen der Uhrzeigerrichtung gedreht.

Dann gilt:

$$x' = r * \cos(\alpha + \phi)$$

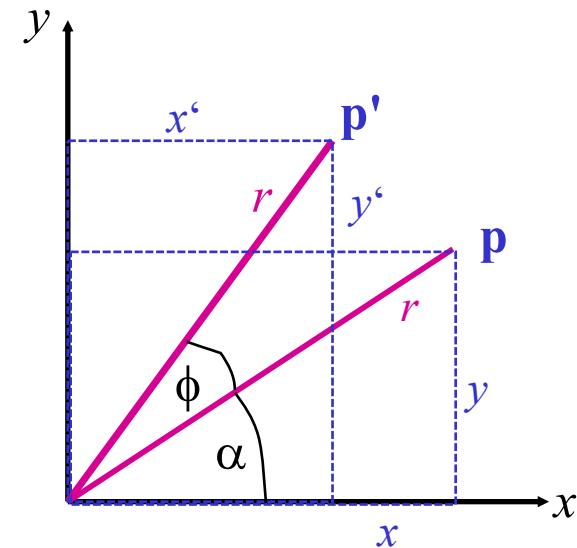
$$x' = \underline{r * \cos \alpha * \cos \phi} - \underline{r * \sin \alpha * \sin \phi}$$

$$x' = x * \cos \phi - y * \sin \phi$$

$$y' = r * \sin(\alpha + \phi)$$

$$y' = \underline{r * \cos \alpha * \sin \phi} + \underline{r * \sin \alpha * \cos \phi}$$

$$y' = x * \sin \phi + y * \cos \phi$$



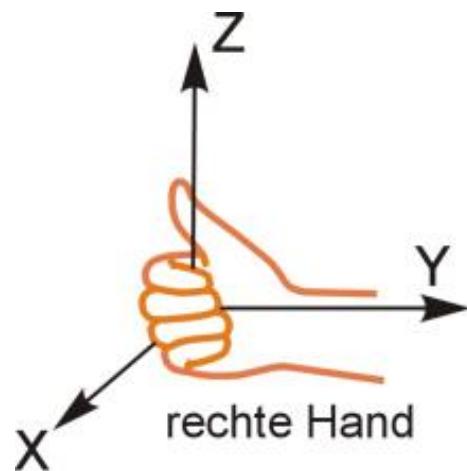
$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \bullet \begin{pmatrix} x \\ y \end{pmatrix}$$

3.3.3 Beschreibung von Transformationen mit Matrizen

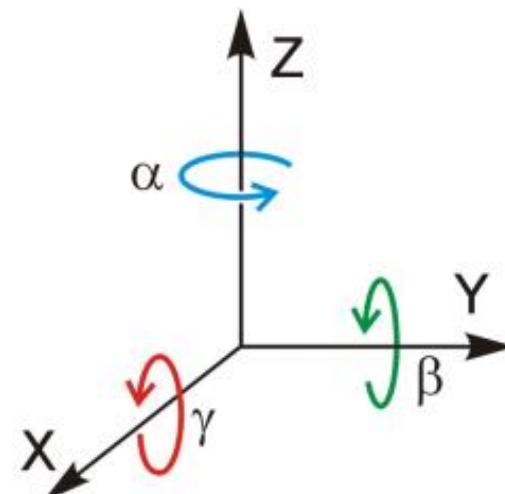


3D-Fall: Hierbei wird die Rotation durch Angabe eines Drehwinkels ϕ und einer Drehachse spezifiziert.

Ein positiver Drehwinkel bedeutet eine Rotation entgegen des Uhrzeigersinns im rechtshändigen Koordinatensystem.



rechtshändiges
Koordinatensystem



Drehrichtungen im rechts-händigen Koordinatensystem

3.3.3 Beschreibung von Transformationen mit Matrizen



Damit ergeben sich folgende Rotationsmatrizen:

$$\mathbf{R}_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{Rotation um die } z\text{-Achse}$$

$$\mathbf{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \quad \text{Rotation um die } x\text{-Achse}$$

$$\mathbf{R}_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{pmatrix} \quad \text{Rotation um die } y\text{-Achse}$$

3.3.3 Beschreibung von Transformationen mit Matrizen



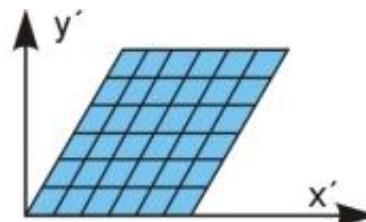
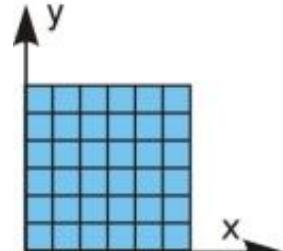
■ Scherung

Eine 2D-Scherung ergibt sich, wenn Abhängigkeiten folgender Form existieren:

$$x' = x + sy * y$$

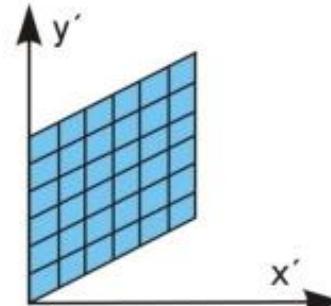
$$y' = sx * x + y$$

Allgemeine 3D-Scherung:



Scherung in x -Richtung

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



Scherung in y -Richtung

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

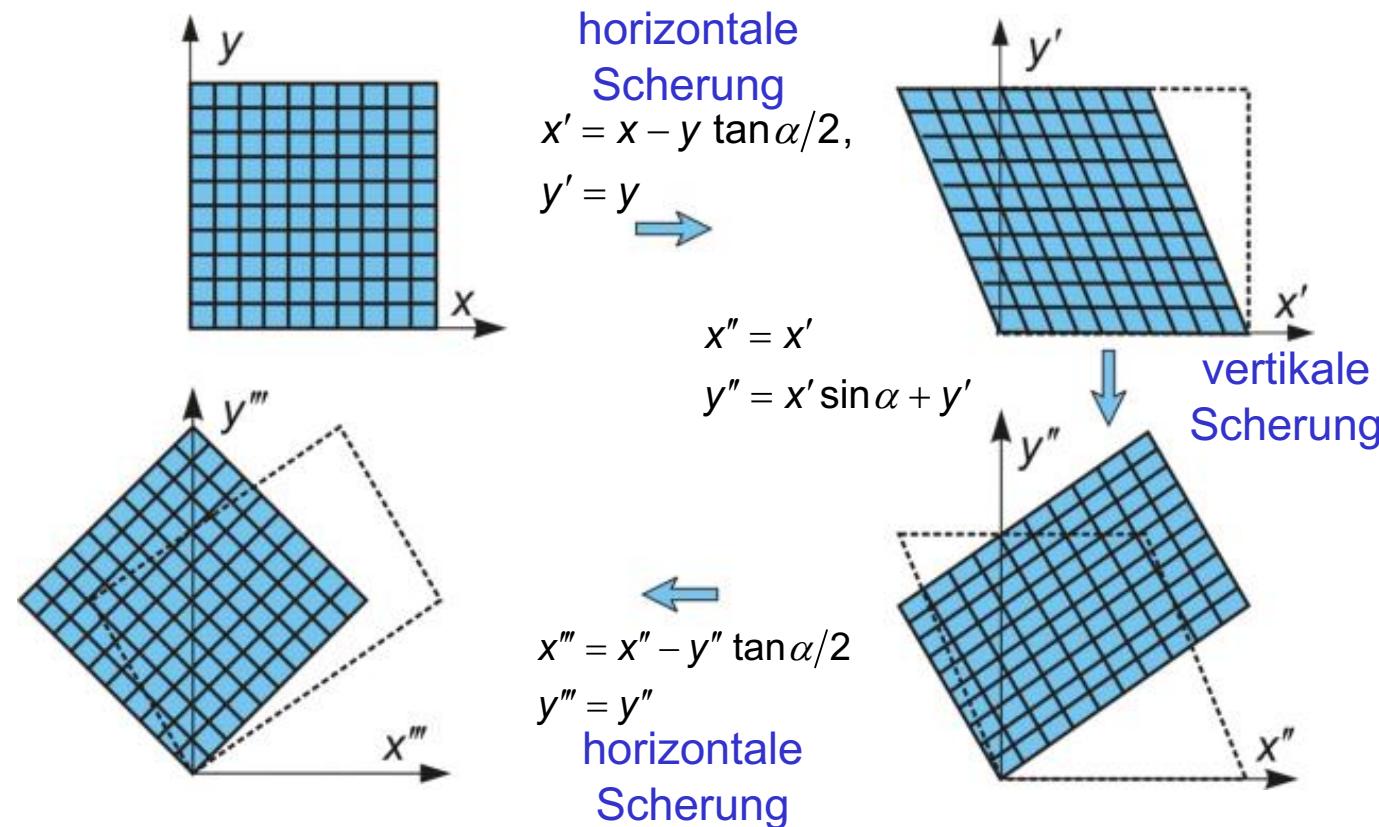
$$\boxed{\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} 1 & s1 & s2 \\ s3 & 1 & s4 \\ s5 & s6 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \end{pmatrix}}$$

3.3.3 Beschreibung von Transformationen mit Matrizen



- Mit Scherungen lassen sich auch Rotationen beschreiben, z.B. Rotation mit 3-Pass-Scherung

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & -\tan \alpha/2 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 \\ \sin \alpha & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & -\tan \alpha/2 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$



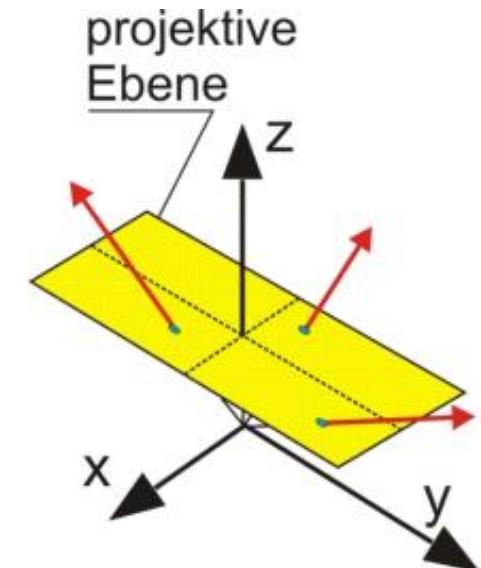
3.3.3 Beschreibung von Transformationen mit Matrizen



▪ Homogene Koordinaten

- **Motivation:** Bisher wurden die Transformationen nicht einheitlich ausgedrückt: $p' = p + t$ bzw. $p' = M \cdot p$
- Mit homogenen Koordinaten ist eine einheitliche Schreibweise und damit das Zusammenfassen von Transformationen möglich.
- Ausgangspunkt zur Definition homogener Koordinaten ist die **Projektive Ebene**:

die Ebene $z=1$ in kartesischen Koordinaten, die von Geraden durch den Ursprung des Koordinatensystems geschnitten wird. Wenn wir den Ursprung als Projektionszentrum betrachten, wird die gesamte Punktmenge einer Geraden auf genau einen Punkt der Projektiven Ebene abgebildet.



3.3.3 Beschreibung von Transformationen mit Matrizen

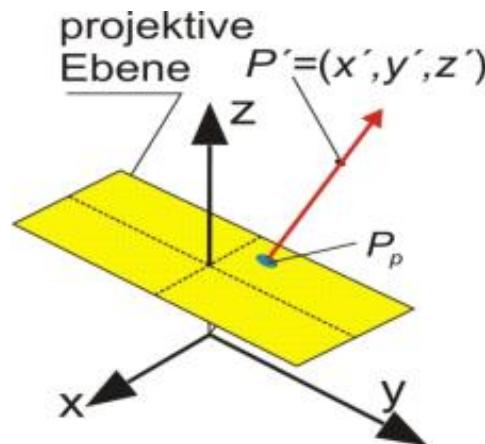


- Homogene Koordinaten von Punkten der Projektiven Ebene lassen sich wie folgt definieren:
 - Es sei ein Punkt $\mathbf{p} = (x, y, 1)^T$ auf der Projektiven Ebene und ein zweiter Punkt $\mathbf{p}' = (x', y', z')^T$ im Koordinatenraum gegeben. (x', y', z') werden als **homogene Koordinaten** von \mathbf{p} in der Projektiven Ebene bezeichnet, wenn gilt:

$$x = x' * t$$

$$y = y' * t$$

$$1 = z' * t$$



3.3.3 Beschreibung von Transformationen mit Matrizen



- **Die Beziehung zwischen homogenen und kartesischen Koordinaten ergibt sich wie folgt:**

(x', y', z') seien die homogenen Koordinaten eines Punktes der Projektiven Ebene, dann sind

$$x = x' / z' \quad \text{und} \quad y = y' / z'$$

die Koordinaten des entsprechenden kartesischen Punktes

$$\mathbf{p} = (x, y)^T.$$

- Jedem Punkt $\mathbf{p} = (x, y)^T$ auf der 2D-kartesischen Ebene ist ein Punkt auf der Projektiven Ebene mit den 3D-homogenen Koordinaten

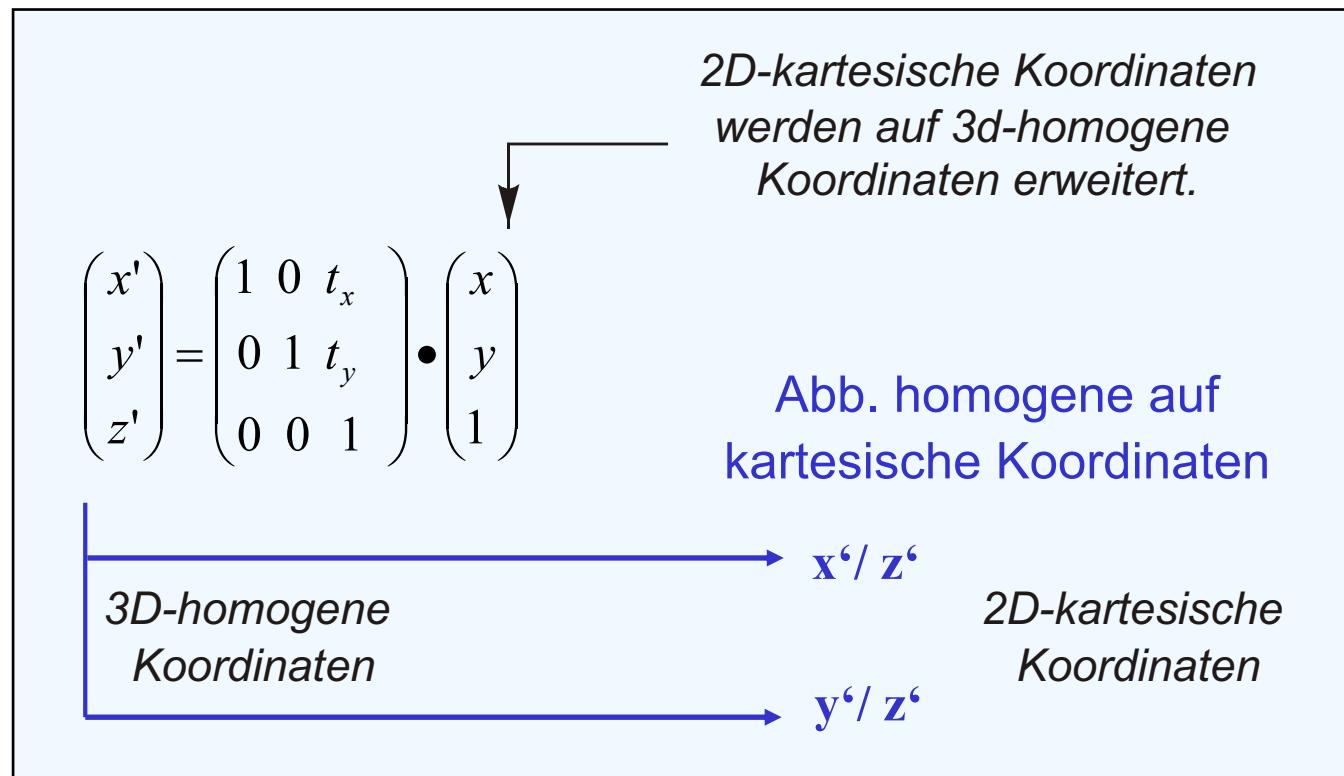
$$x' = x, \quad y' = y, \quad z' = 1$$

zugeordnet.

3.3.3 Beschreibung von Transformationen mit Matrizen



Mit homogenen Koordinaten lassen sich auch Translation und Projektion mittels Matrixmultiplikation ausführen



3.3.3 Beschreibung von Transformationen mit Matrizen



- Der 3D Fall lässt sich analog herleiten:

(x', y', z', w') seien die homogenen Koordinaten eines Punktes des 3-dim. projektiven Raumes, dann sind

$$x = x' / w' \quad y = y' / w' \quad \text{und} \quad z = z' / w' \quad (w' \neq 0)$$

die Koordinaten des zugeordneten 3D-kartesischen Punktes.

Ist $\mathbf{p} = (x, y, z)^T$ ein kartesischer Punkt, so wird er dem projektiven Punkt $\mathbf{p}' = (x, y, z, 1)^T$ zugeordnet.

3.3.3 Beschreibung von Transformationen mit Matrizen



- Affine Transformationen in homogenen Koordinaten:

2D-Fall:

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

3D-Fall:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

3.3.3 Beschreibung von Transformationen mit Matrizen



	Translation	Skalierung	Scherung
2D	$\mathbf{T}(t_x, t_y) = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$	$\mathbf{S}(s_x, s_y) = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$	$\mathbf{H}_x = \begin{pmatrix} 1 & h_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$
3D	$\mathbf{T}(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\mathbf{S}(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$	$\mathbf{H} = \begin{pmatrix} 1 & s_1 & s_2 & 0 \\ s_3 & 1 & s_4 & 0 \\ s_5 & s_6 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

Affine Transformationen mit homogenen 2D/3D-Koordinaten

3.3.3 Beschreibung von Transformationen mit Matrizen



2D-Rotation	3D-Rotation
	Rotation um x -Achse $\mathbf{R}_x(\phi) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
$\mathbf{R}(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$	Rotation um y -Achse $\mathbf{R}_y(\phi) = \begin{pmatrix} \cos\phi & 0 & \sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
	Rotation um z -Achse $\mathbf{R}_z(\phi) = \begin{pmatrix} \cos\phi & -\sin\phi & 0 & 0 \\ \sin\phi & \cos\phi & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$

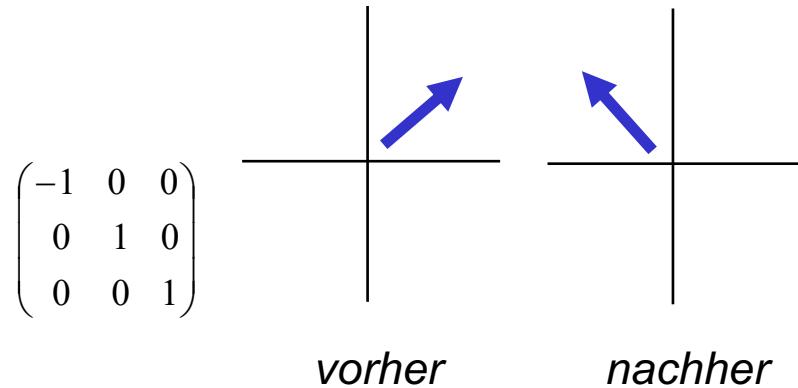
Rotation mit homogenen 2D/3D-Koordinaten

3.3.3 Beschreibung von Transformationen mit Matrizen

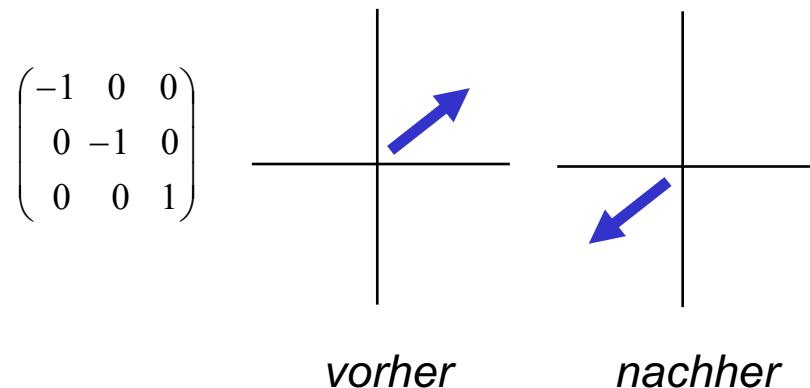


Beispiele für weitere Transformationen:

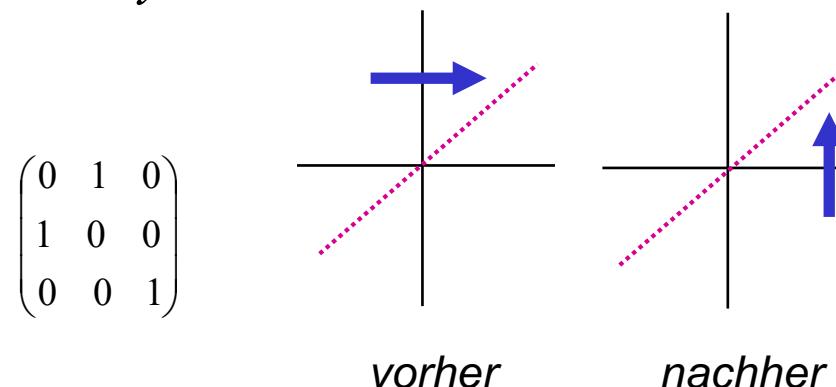
Spiegelung an der y -Achse



Spiegelung am Ursprung



Spiegelung an der Geraden $y = x$



3.3.3 Beschreibung von Transformationen mit Matrizen



▪ Komposition von Transformationen

Durch die einheitliche Schreibweise lassen sich mehrere Transformationen zusammenfassen

- **Beispiel 1:** Drehung um den Punkt $\mathbf{p}_1 = (x_1, y_1)^T$ mit Winkel α und Skalierung mit sx und sy

Hierzu müssen folgende Schritte durchgeführt werden:

- Verschieben von \mathbf{p}_1 in den Ursprung,
- Drehung um den Winkel α ,
- Skalierung,
- Rückverschiebung.

Daraus ergibt sich folgende Transformationsmatrix:

$$\mathbf{T}(x_1, y_1)) \bullet \mathbf{S}(sx, sy) \bullet \mathbf{R}(\alpha) \bullet \mathbf{T}(-x_1, -y_1) =$$

$$\begin{pmatrix} 1 & 0 & x_1 \\ 0 & 1 & y_1 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} 1 & 0 & -x_1 \\ 0 & 1 & -y_1 \\ 0 & 0 & 1 \end{pmatrix}$$

3.3.3 Beschreibung von Transformationen mit Matrizen

- Beispiel 2: Rotation um eine Achse $\overline{\mathbf{p}_1 \mathbf{p}_2}$ mit dem Winkel ϕ mit $\mathbf{p}_1 = (x_1, y_1, z_1, 1)^T$ und $\mathbf{p}_2 = (x_2, y_2, z_2, 1)^T$

- Verschiebung von \mathbf{p}_1 in den Ursprung

$$\mathbf{p}_2' = \mathbf{T}(-x_1, -y_1, -z_1) \bullet \mathbf{p}_2 = (x_2 - x_1, y_2 - y_1, z_2 - z_1, 1)^T$$

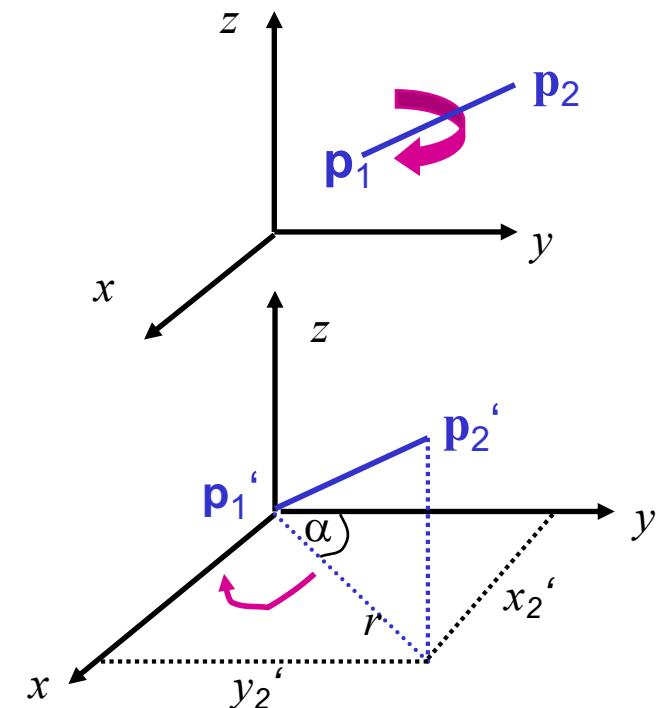
- Drehung um z-Achse bis \mathbf{p}_2' in der $x-z$ -Ebene liegt

Drehwinkel ist $-(90-\alpha)$ bzw. $(\alpha - 90)$

$$\cos(\alpha - 90) = \sin \alpha = x_2' / r$$

$$\sin(\alpha - 90) = \cos \alpha = y_2' / r$$

$$r = \sqrt{(x_2')^2 + (y_2')^2}$$



$$\mathbf{p}_2'' = \mathbf{R}_z(\alpha - 90) \bullet \mathbf{p}_2' = \mathbf{R}_z(\alpha - 90) \bullet \mathbf{T}(-x_1, -y_1, -z_1) \bullet \mathbf{p}_2$$

3.3.3 Beschreibung von Transformationen mit Matrizen



- Drehung um die y -Achse bis p_2'' in der x -Achse liegt
Drehwinkel ist β

$$\cos \beta = x_2''/R$$

$$\sin \beta = z_2''/R$$

$$R = \sqrt{(x_2'')^2 + (z_2'')^2}$$

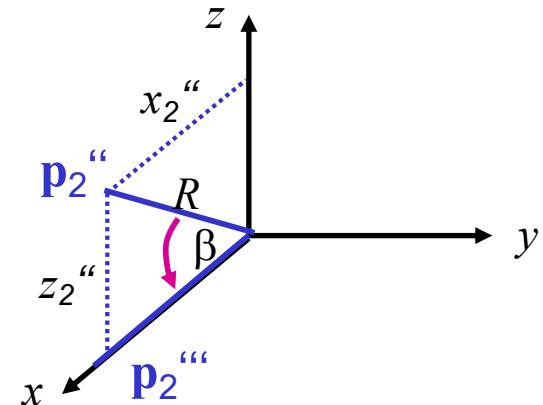
Transformationsmatrix $R_y(\beta)$

- Drehung um die x -Achse um Winkel ϕ
(eigentliche Drehung)

Transformationsmatrix $R_x(\phi)$

- Rücktransformation

$$\begin{aligned} & (R_y(\beta) \bullet R_z(\alpha - 90) \bullet T(-x_1, -y_1, -z_1))^{-1} \\ &= T(x_1, y_1, z_1) \bullet R_z(-\alpha + 90) \bullet R_y(-\beta) \\ &= T(x_1, y_1, z_1) \bullet R_z^T(\alpha - 90) \bullet R_y^T(\beta) \end{aligned}$$



3.3.3 Beschreibung von Transformationen mit Matrizen



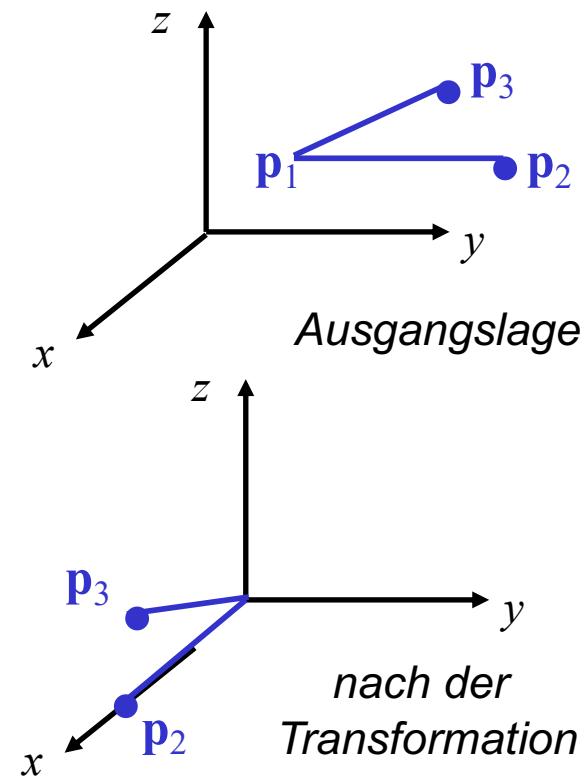
Zusammenfassung der Transformationsschritte zur Transformation eines Punktes \mathbf{p} um die Achse $\overline{\mathbf{p}_1\mathbf{p}_2}$ mit dem Winkel ϕ

$$\mathbf{p}_{\text{trans}} = \mathbf{T}(x_1, y_1, z_1) \bullet \mathbf{R}_z^T(\alpha - 90) \bullet \mathbf{R}_y^T(\beta) \bullet \mathbf{R}_x(\phi) \\ \bullet \mathbf{R}_y(\beta) \bullet \mathbf{R}_z(\alpha - 90) \bullet \mathbf{T}(-x_1, -y_1, -z_1) \bullet \mathbf{p}$$

3.3.3 Beschreibung von Transformationen mit Matrizen



- **Beispiel 3:** Rotation zweier gerichteter Liniensegmente (beschrieben durch die Punkte p_1 , p_2 und p_3), so daß die Strecke $\overline{p_1p_2}$ in der x -Achse und p_3 in der xz -Ebene liegt.
 - Translation von p_1 in den Ursprung mit $T(-x_1, -y_1, -z_1)$
 - Rotation um die z -, y - und x -Achse (vgl. Beispiel 2)
 - Rotation um die z -Achse bis p_2 in der xz -Ebene liegt
 - Rotation um die y -Achse bis p_2 auf der x -Achse liegt
 - Rotation um die x -Achse bis p_3 in der xz -Ebene liegt



3.3.3 Beschreibung von Transformationen mit Matrizen



Die Rotation um alle 3 Achsen lässt sich anhand nur einer Matrix unter Ausnutzung der Orthogonalität beschreiben mit:

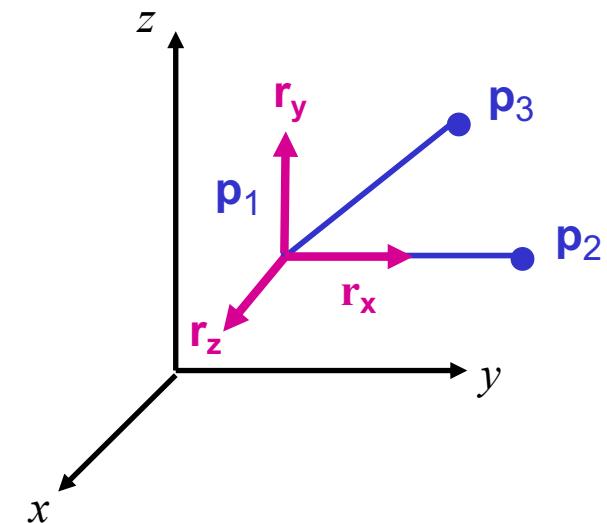
$$\mathbf{r}_x = (r_{x1}, r_{x2}, r_{x3})^T = \frac{\overrightarrow{\mathbf{p}_2 - \mathbf{p}_1}}{\|\overrightarrow{\mathbf{p}_2 - \mathbf{p}_1}\|} \text{ (Einheitsvektor in } \mathbf{p}_1\mathbf{p}_2\text{-Richtung)}$$

$$\mathbf{r}_y = (r_{y1}, r_{y2}, r_{y3})^T = \frac{(\overrightarrow{\mathbf{p}_3 - \mathbf{p}_1}) \times (\overrightarrow{\mathbf{p}_2 - \mathbf{p}_1})}{\|(\overrightarrow{\mathbf{p}_3 - \mathbf{p}_1}) \times (\overrightarrow{\mathbf{p}_2 - \mathbf{p}_1})\|} \text{ (} \mathbf{r}_y \text{ orthogonal zur } \mathbf{p}_1\mathbf{p}_2\mathbf{p}_3\text{-Ebene)}$$

$$\mathbf{r}_z = (r_{z1}, r_{z2}, r_{z3})^T = \mathbf{r}_x \times \mathbf{r}_y$$

Die Transformationsmatrix lässt sich damit wie folgt aufstellen:

$$\mathbf{R} = \begin{pmatrix} r_{x1} & r_{x2} & r_{x3} & 0 \\ r_{y1} & r_{y2} & r_{y3} & 0 \\ r_{z1} & r_{z2} & r_{z3} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



3.3.3 Beschreibung von Transformationen mit Matrizen

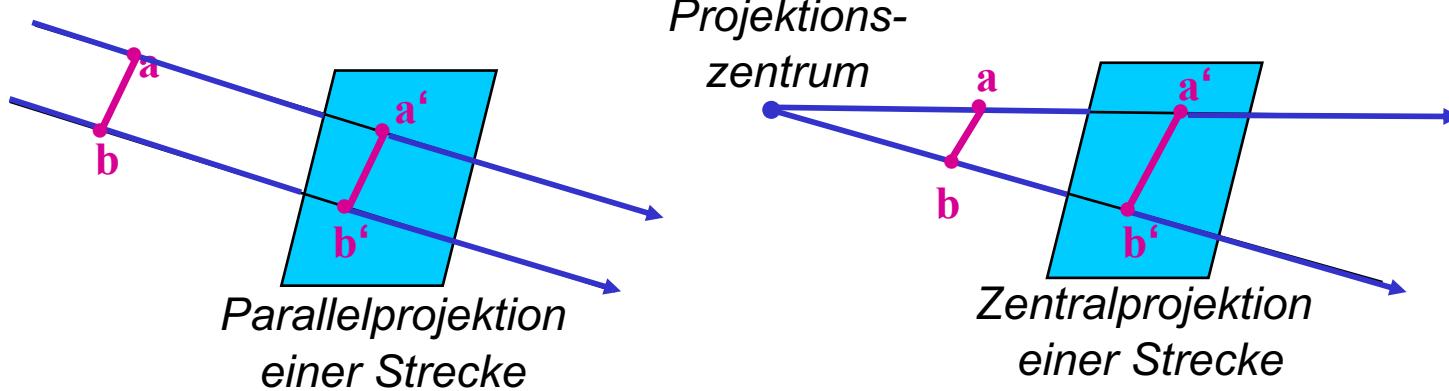


- **Projektionen**

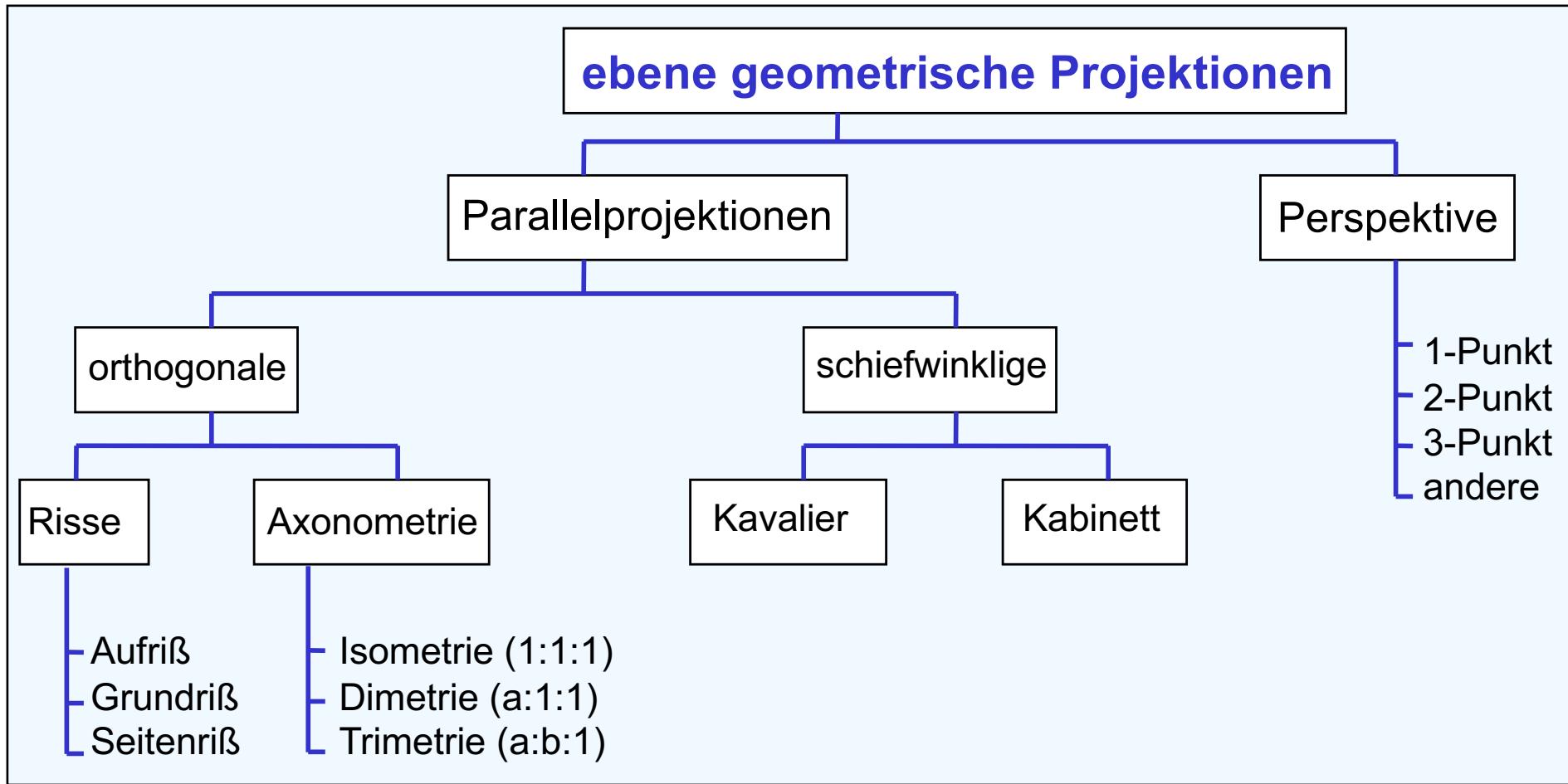
Eine Projektion ist eine spezielle Abbildung, bei der ein Punkt $\mathbf{p} = (x_1, x_2, \dots, x_n)^T$ der Dimension n in einen Punkt $\mathbf{p}' = (x_1, x_2, \dots, x_m)^T$ der Dimension m mit $m < n$ abgebildet wird.

In der Computergraphik betrachten wir Projektionen von 3-dim. Punkten auf eine 2-dim. Ebene.

Wir unterscheiden zwischen Parallelprojektion und Zentralprojektion (perspektivische Projektion).



3.3.3 Beschreibung von Transformationen mit Matrizen



3.3.3 Beschreibung von Transformationen mit Matrizen



▪ Parallelprojektionen

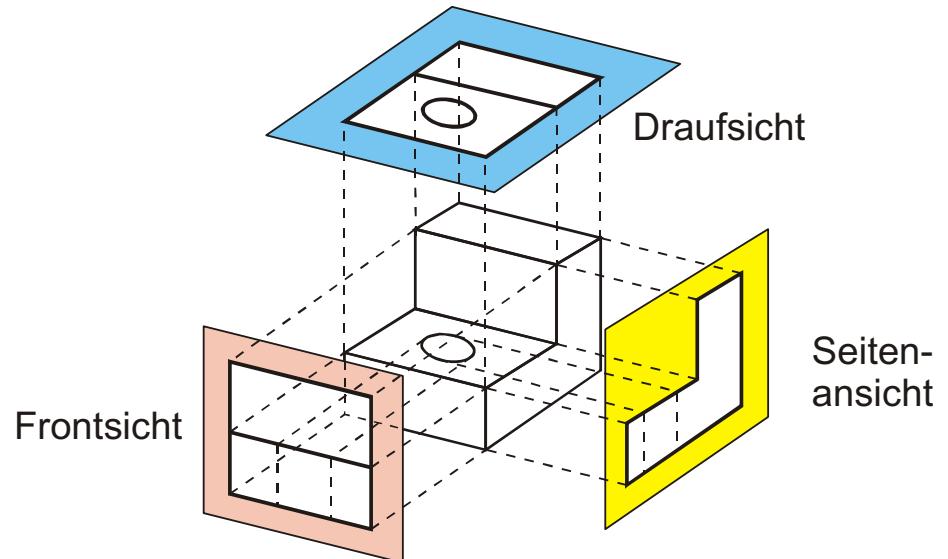
In Abhängigkeit davon, ob die Projektionsrichtung senkrecht oder schiefwinklig auf der Projektionsebene steht, unterscheidet man orthogonale und schiefwinkelige Projektionen. Zu den **orthogonalen Projektionen** gehören:

- **Risse**

Die Projektionsrichtung stimmt mit einer Koordinatenrichtung überein, z.B. Front-, Drauf- und Seitensicht.

Die Projektion auf die Ebene $z = z_0$ ist in Matrixschreibweise gegeben durch:

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & z_0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



3.3.3 Beschreibung von Transformationen mit Matrizen



- Die Berechnung der Transformationsmatrix für die parallele Projektion:

gegeben:

- Projektionsebene durch Punkt/Normale \mathbf{p} , \mathbf{n}
- Projektionsrichtung \mathbf{r}

- 1) Verschiebung dass \mathbf{p} in Koordinatenursprung
- 2) Achsenrotationen bis \mathbf{n} in Richtung der z -Achse, d.h.
Projektionsebene ist x,y -Ebene
- 3) Scherungen bis \mathbf{r} in Richtung der z -Achse
- 4) Projektion auf Ebene $z = 0$
- 5) Rücktransformation von 3), 2) und 1).

3.3.3 Beschreibung von Transformationen mit Matrizen



- **Perspektivische Projektionen**

Zentralprojektion durch einen Punkt

3.3.3 Beschreibung von Transformationen mit Matrizen



Herleitung der Transformationsmatrix für die Zentralprojektion für folgendes Beispiel :

Projektionsebene ist die Ebene $z = 0$ und das Projektionszentrum liegt auf der negativen z -Achse $(0, 0, -a)$.

Aus der Ähnlichkeit der Dreiecke

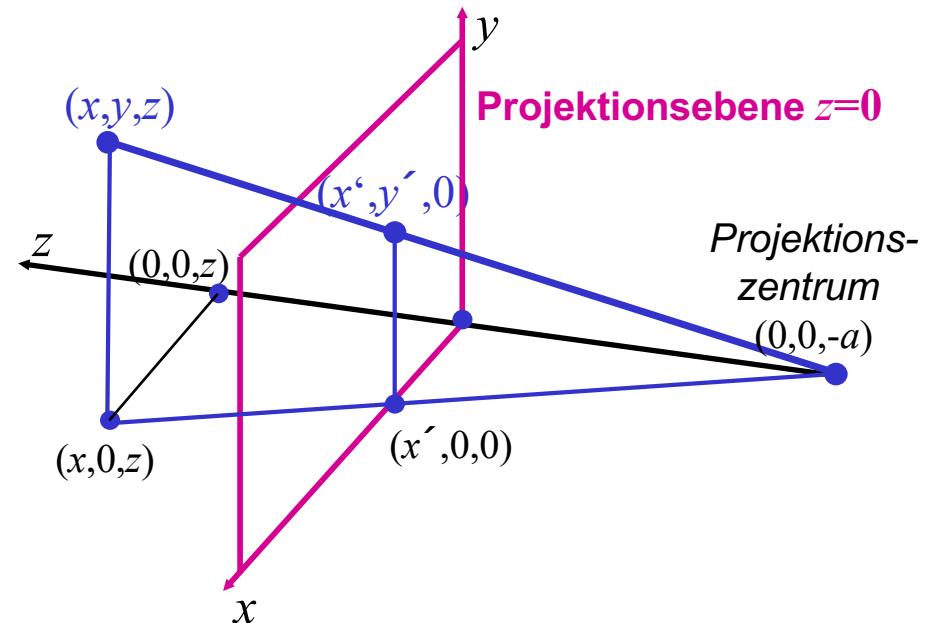
$(0,0,-a), (0,0,z), (x,0,z)$ und
 $(0,0,-a), (0,0,0), (x',0,0)$

ergibt sich für x' : $x' = \frac{ax}{(z+a)} = \frac{x}{(z/a+1)}$

und analog für y' : $y' = \frac{ay}{(z+a)} = \frac{y}{(z/a+1)}$

Und damit für die Transformation:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/a & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$



3.3.3 Beschreibung von Transformationen mit Matrizen



- Die Berechnung der Transformationsmatrix für die Zentralprojektion:
gegeben:

- Projektionsebene durch Punkt/Normale \mathbf{p} , \mathbf{n}
 - Projektionszentrum \mathbf{c} außerhalb der Projektionsebene

- 1) Verschiebung dass \mathbf{p} in Koordinatenursprung
- 2) Achsenrotationen bis \mathbf{n} in Richtung der z -Achse, d.h. Projektionsebene ist x,y -Ebene
- 3) Scherungen bis $(\mathbf{c}-\mathbf{0})$ in Richtung der z -Achse
- 4) Zentralprojektion (s. vorige Folie)
- 5) Rücktransformation von 3), 2) und 1).

3.3.3 Beschreibung von Transformationen mit Matrizen



- Affine Transformationen in homogenen Koordinaten:

2D-Fall:

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

3D-Fall:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Parallelprojektionen sind affine Transformationen

3.3.3 Beschreibung von Transformationen mit Matrizen



- Projektive Transformationen in homogenen Koordinaten:

2D-Fall:

$$\begin{pmatrix} x' \\ y' \\ w \end{pmatrix} = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

3D-Fall:

$$\begin{pmatrix} x' \\ y' \\ z' \\ w \end{pmatrix} = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & p & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

Zentralprojektionen sind projektive Transformationen

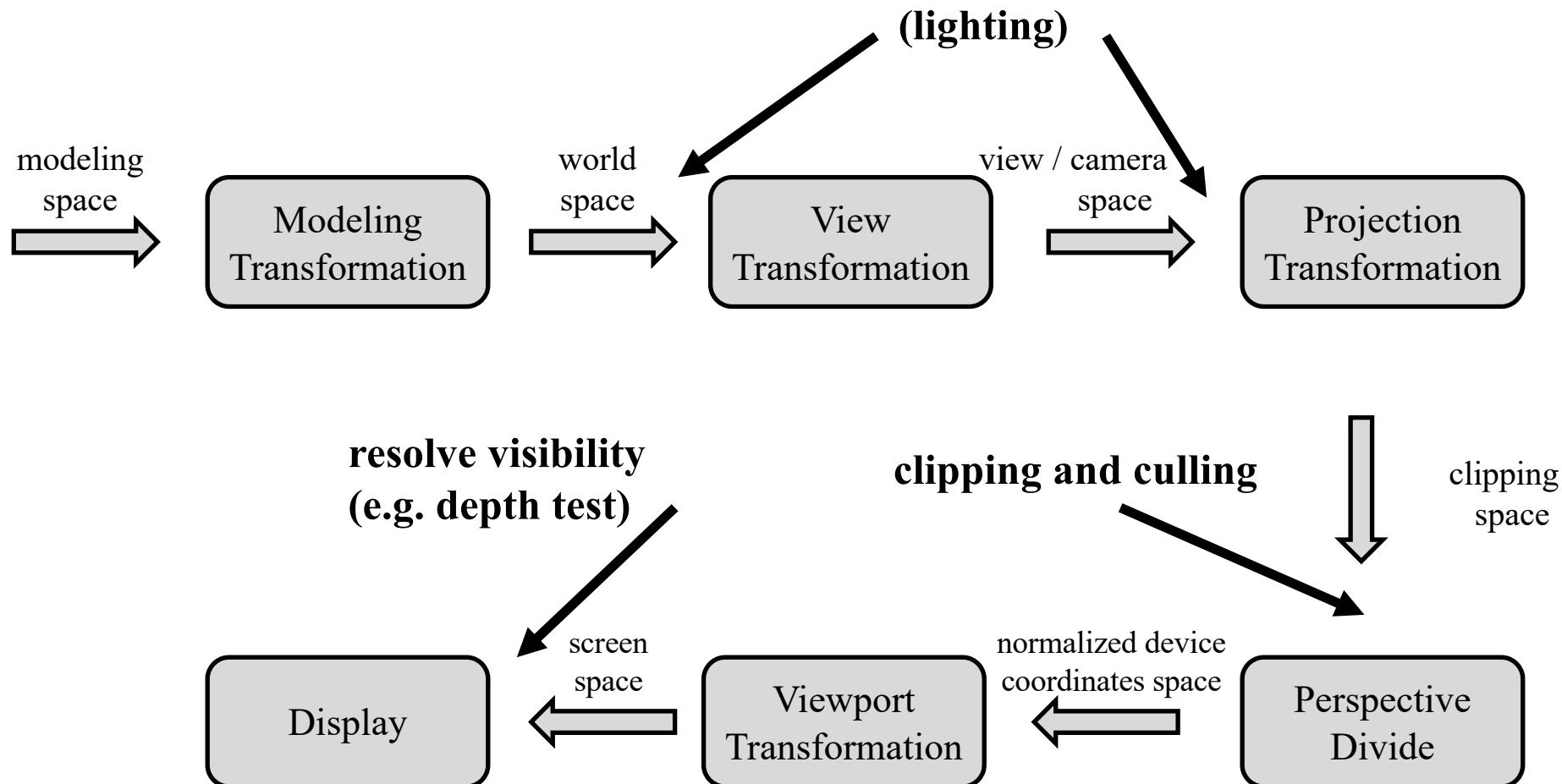


3.3.4 Realisierung der Viewing-Pipeline

3.3.4 Realisierung der Viewing-Pipeline



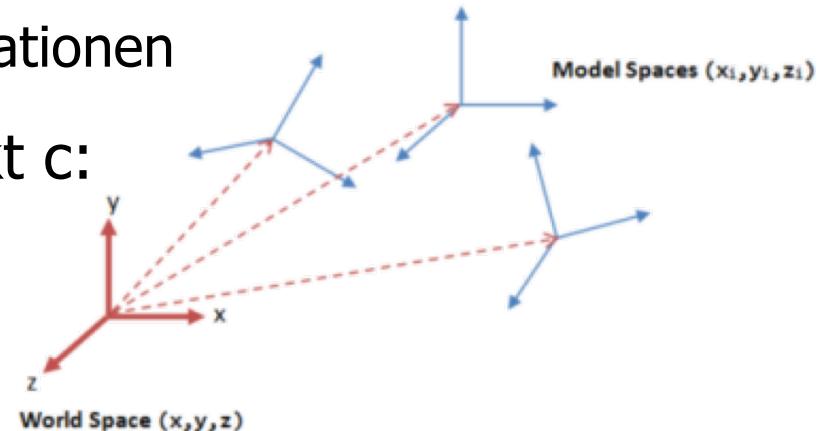
Übliche Koordinatensysteme der 3D-Computergrafik



3.3.1 Die Viewing-Pipeline



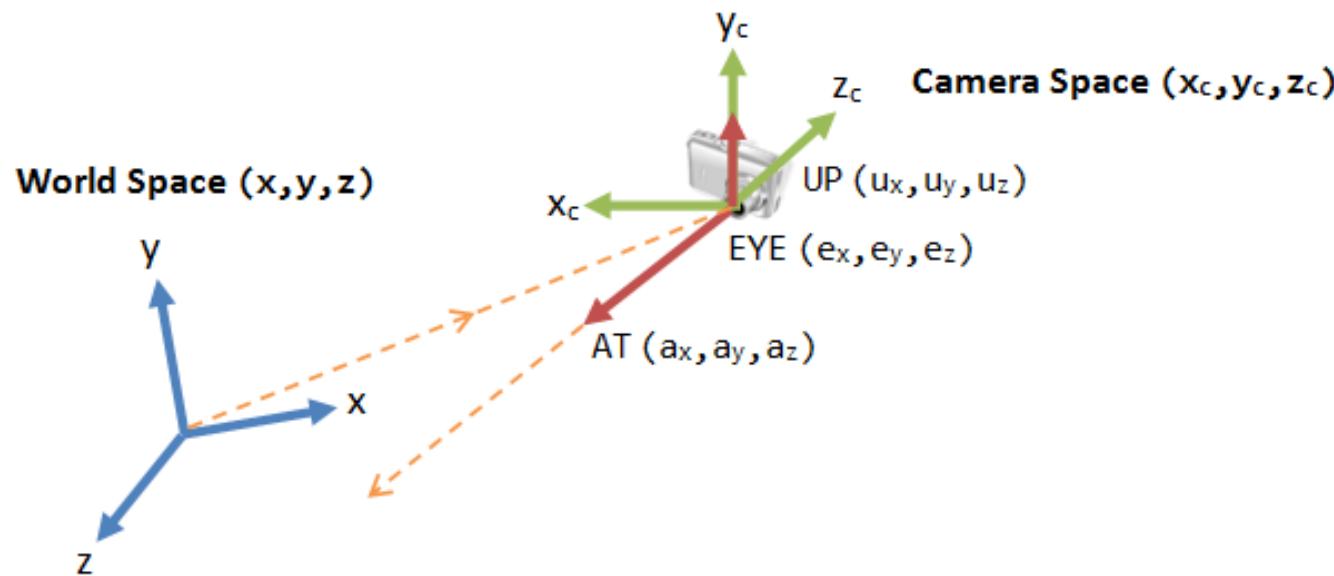
- Modeling Transformation
 - Quader-zu-Quader-Transformation , d.h. Hintereinanderausführung der objektbezogenen affinen Transformationen
- Beispiel Rotation um Objektschwerpunkt c :
 - Translation in den Ursprung $T(-c)$
 - Rotation um x -Achse $R(\phi, 0, 0, 1)$
 - Rückgängigmachen der Translation $T(c)$
 - Die Model-Matrix ist somit $M = T(c) * R(\phi, 0, 0, 1) * T(-c)$
 - Die Überführung in world space erfolgt mit $p_{\text{world}} = M * p_{\text{model}}$



3.3.1 Die Viewing-Pipeline



- View Transformation
 - Verschiebung des Weltkoordinatenursprungs zur Kameraposition
 - Rotation der Weltkoordinatenachsen auf die Achsen des Kamerakoordinatensystems (Orthonormalbasis)



3.3.1 Die Viewing-Pipeline



- View Transformation

- `Matrix4x4 view = getLookAt(EYE, AT, UP)`

$$\mathbf{z}_c = \frac{\mathbf{EYE} - \mathbf{AT}}{|\mathbf{EYE} - \mathbf{AT}|}$$

$$\mathbf{x}_c = \frac{\mathbf{UP} \times \mathbf{AT}}{|\mathbf{UP} \times \mathbf{AT}|}$$

$$\mathbf{y}_c = \mathbf{z}_c \times \mathbf{x}_c$$

- Rotation $\mathbf{R} = \begin{pmatrix} x_{cx} & x_{cy} & x_{cz} & 0 \\ y_{cx} & y_{cy} & y_{cz} & 0 \\ z_{cx} & z_{cy} & z_{cz} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$
- Translation $\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & -EYE_x \\ 0 & 1 & 0 & -EYE_y \\ 0 & 0 & 1 & -EYE_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- View-Matrix $\mathbf{V} = (\mathbf{R} * \mathbf{T}) = \begin{pmatrix} x_{cx} & x_{cy} & x_{cz} & -\mathbf{x}_c \cdot \mathbf{EYE} \\ y_{cx} & y_{cy} & y_{cz} & -\mathbf{y}_c \cdot \mathbf{EYE} \\ z_{cx} & z_{cy} & z_{cz} & -\mathbf{z}_c \cdot \mathbf{EYE} \\ 0 & 0 & 0 & 1 \end{pmatrix}$

3.3.1 Die Viewing-Pipeline

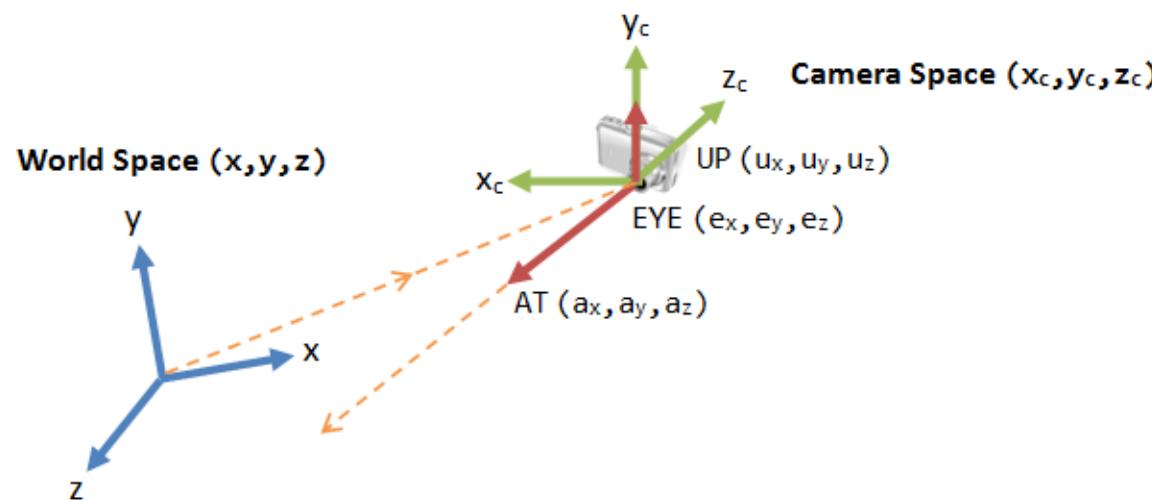


- View Transformation
- Die Überführung von world space in camera space erfolgt mit

$$\mathbf{p}_{\text{camera}} = \mathbf{V} * \mathbf{p}_{\text{world}}$$

- Oder mittels Model-View-Matrix von model space in camera space

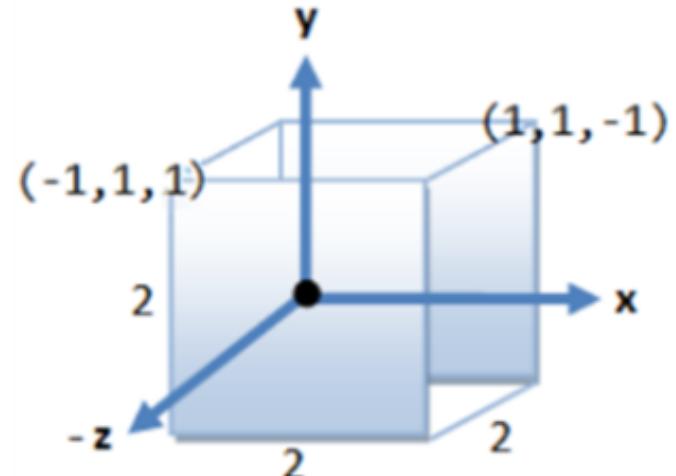
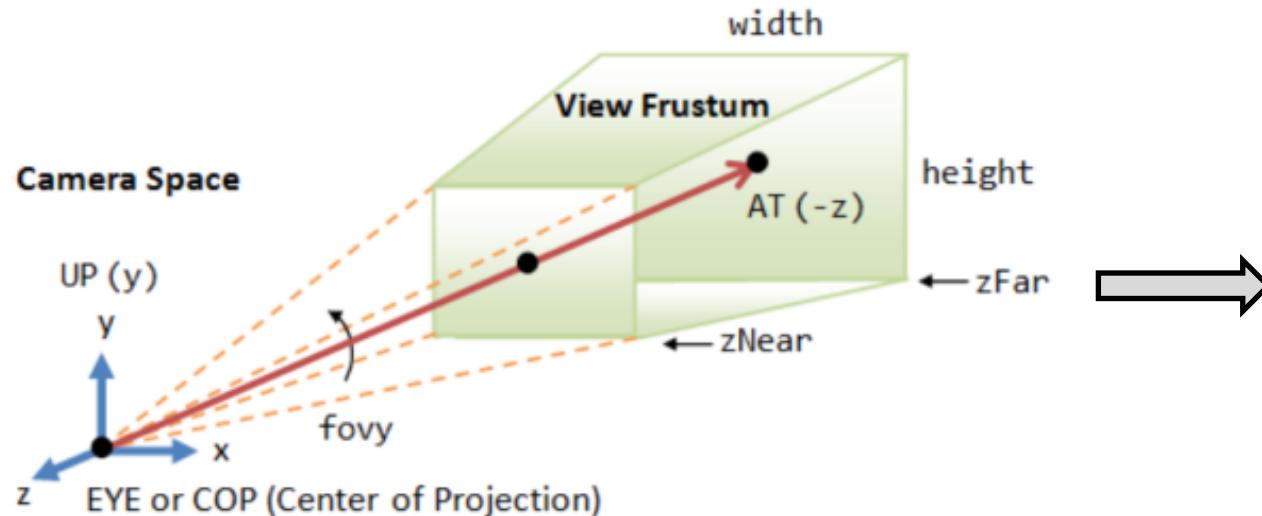
$$\mathbf{p}_{\text{camera}} = \mathbf{V} * \mathbf{M} * \mathbf{p}_{\text{model}}$$



3.3.1 Die Viewing-Pipeline



- Perspective Projection Transformation



$$P = \begin{pmatrix} \frac{\cot(fov / 2)}{aspect} & 0 & 0 & 0 \\ 0 & \cot(fov / 2) & 0 & 0 \\ 0 & 0 & -\frac{2 * z_{far}}{z_{far} - z_{near}} & \frac{2 * z_{near} * z_{far}}{z_{far} - z_{near}} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

$$aspect = \frac{width}{height}$$

3.3.1 Die Viewing-Pipeline



- Perspective Projection Transformation
- Die Überführung von camera space in clipping space erfolgt mit

$$\mathbf{p}_{\text{clipping}} = \mathbf{P} * \mathbf{p}_{\text{camera}}$$

- Oder von model space in clipping space

$$\mathbf{p}_{\text{clipping}} = \mathbf{P} * \mathbf{V} * \mathbf{M} * \mathbf{p}_{\text{model}}$$

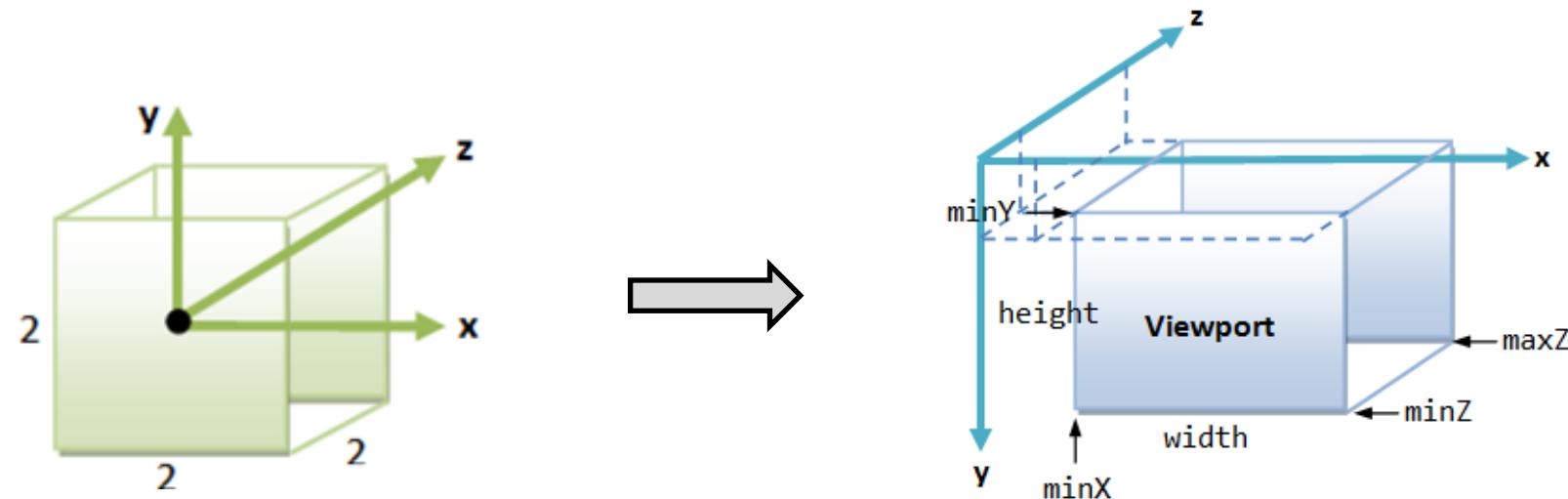
- Die Überführung in normalized device coordinates space durch Perspective Division erfolgt durch die Grafikhardware

$$\mathbf{p}_{\text{device}} = \frac{\mathbf{p}_{\text{clipping}}}{w_{\text{clipping}}}$$

3.3.1 Die Viewing-Pipeline



- Viewport Transformation (auch durch Grafikhardware)
- Abschließend erfolgt eine Skalierung entsprechend der Bildauflösung und eine Parallelprojektion in Richtung der z-Achse (durch Weglassen der z-Koordinate)
- Dies ist eine Quader-zu-Quader-Abbildung, sie überführt von normalized device coordinates space in den screen space



3.3.4 Realisierung der Viewing-Pipeline



- **Zusammenfassende Darstellung der Viewing-Pipeline:**
 - **Modeling Transformation** (Quader zu Quader-Abbildung, Translationen, Rotationen, Skalierungen),
 - **View Transformation**
 - Ausrichtung des Weltkoordinatensystems am Kamerakoordinatensystem (Verschiebung und Rotation)
 - **Projection Transformation**
 - Pyramidenstumpf-förmiges Frustum wird in das kanonische Ansichtsvolumen überführt
 - Überführung von view space in den clipping space
 - **Perspective Divide**
 - Überführt von clipping space in normalized device coordinates space durch Division mit der w-Komponente
 - **Viewport Transformation**
 - Quader zu Quader- Abbildung, 2 Translationen, 1 Skalierung