Praktikum Computergrafik

Winfried Kurth

Lehrstuhl für Computergrafik und ökologische Informatik

Abteilung Ökoinformatik, Biometrie und Waldwachstum, Büsgenweg 4 (Fakultät für Forstwissenschaften und Waldökologie); **wk<at>informatik.uni-goettingen.de**

Wintersemester 2018/19

Zeitplan und Themenübersicht des theoretischen Teils
(donnerstags, 16:15-18:00, Büsgenweg 4, FSR4.3)

18. 10. 2018	Übersicht. Affine Abbildungen, homogene Koordinaten, Objekttransformationen. Sichttransformation.
25. 10.	Projektionsarten und Perspektive. Techniken der 2D-Rastergrafik.
1. 11.	Halbtontechniken, Antialiasing. Visible Surface Determination, z-Buffer.
8. 11.	Grundlagen zur visuellen Wahrnehmung und zur Farbe. Grafische Hardware.
15. 11.	Lichtquellen und Reflexion, lokale Beleuchtungsrechnung.
22. 11.	Gouraud- und Phong-Shading, Transparenz und Schatten. Kurvenmodelle (Splines, Bézier-Kurven).
29. 11.	Kurvenmodelle (2.), Modelle für gekrümmte Flächen.
6. 12.	Texturierung.
13. 12.	Raytracing.
20. 12.	Darstellungsarten für 3D-Objekte und Szenen: Boundary representation, Voxelmodelle, Constructive solid geometry.
10. 1. 19	Grundzüge von VRML.
17. 1.	Volumen-Rendering.
24. 1.	Fraktale Modelle.
31. 1.	Animation.

Skript des Theorie-Teils:
http://www.uni-forst.gwdg.de/~wkurth/cg18_skript.htm

Quellenangaben und Literaturhinweise:
http://www.uni-forst.gwdg.de/~wkurth/cg08_lit.htm

1. Einleitung: Einige Begriffe

"Grafische Datenverarbeitung" als Oberbegriff.

ISO-Definition von 1982: "Methods and techniques for converting data to and from graphics displays via computer".

Untergliederung der grafischen DV nach Rosenfeld:

		Ausgabe	
		Bild	Beschreibung
Eingabe	Bild	Bildverarbeitung	Bildanalyse, Mustererkennung
	Beschreibung	generative Computergrafik	(andere DV-Disziplinen)

Hier: Schwerpunktthema "generative Computergrafik", kurz ***CG***.

Beziehungen zwischen den 3 Teilgebieten:

Weitere Begriffe:

Visualisierung:

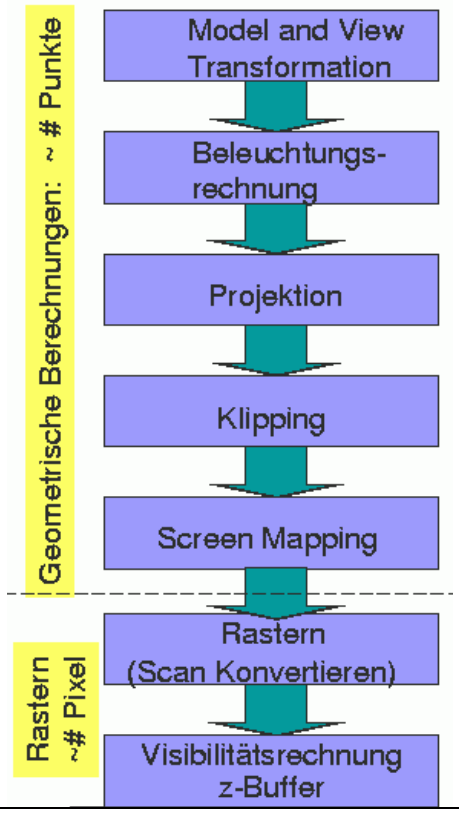
Verwendung von computergestützten Techniken der grafischen Datenverarbeitung, um Daten besser verstehbar zu machen oder aus den Resultaten von Messungen, Simulationen oder Berechnungen Wissen zu extrahieren.
Verwendung von optischen *Wahrnehmungsprimitiven*, um Informationen abzubilden (Form, Farbe, Textur, Zeit).

Rendering (keine adäquate deutsche Übersetzung):
Bilderzeugung am Computer aus geometrischen Modellen.
Häufiges Ziel: fotorealistisches Rendering;
Super-Fotorealismus (!)
interessanter, jüngerer Zweig, bes. in der Computerkunst:
Nicht-fotorealistisches Rendering (NPR).

"Rendering Pipeline"

- geometrische Transformationen (Rotation, Skalierung, Translation)
- Betrachtungstransformationen / Viewing: Projektion der 3D-Geometrie auf die 2D-Darstellungsfläche
- *visible surface determination* / *hidden surface removal*: Entfernen von Teilen des Modells, die zur Darstellung nichts beitragen oder von anderen Teilen verdeckt sind
- Beleuchtung und Schattierung
  Beschreibung von Licht und Farbe und der Interaktion des Lichtes mit den geometrischen Objekten; Verfahren zur Berechnung der Farbe der dargestellten Bildpunkte
- Texturen und Schatten
  realistischere Darstellung durch Simulation verschiedener Materialoberflächen und durch Schatten

(vgl. Schlechtweg 2001)



Beispiel für eine *rendering pipeline* (aus Krömker 2001), verschiedene Varianten (Kombinationsmöglichkeiten der Einzelschritte) sind denkbar.

Alternatives Paradigma zur Geometrie-basierten Grafik:

Sample-basierte Grafik.
Kein geometrisches Modell als Zwischenschritt – direkte Verwendung "gesampelter" optischer Informationen (z.B. von Digitizern, digitalen Kameras).
Einsatz:
2D-Bildmanipulation

Beispiel:

Michele Turre: Die Künstlerin, ihre Tochter und ihre Mutter, alle im Alter von 3 Jahren (aus van Dam 2001).

Nachteil der sample-based graphics:
WYSIAYG (What You See Is All You Get): Keine Zusatzinformationen, insbes. keine Tiefeninformation, kein *Walkthrough* möglich.
In neuerer Zeit dennoch großes Interesse: Neue Bilder werden aus alten konstruiert durch Interpolation (*Morphing*), Komposition, Verzerrung usw.

2. Affine Abbildungen, homogene Koordinaten, Transformationen

Für die Grafik sind *Abbildungen* (*Transformationen*) von 2D- und 3D-Objekten für verschiedene Zwecke wichtig:

- Platzierung von (Standard-) Objekten in einer Szene
- Animation von Objekten (kontinuierliche Veränderung der Lage und Größe)
- interaktives Editieren (Drehen, Ändern der Größe, Spiegeln, Verzerren...)

Behandlung von Transformationen mittels vektorieller Geometrie (lineare Algebra)

$$\vec{v} \in \mathrm{R}^n \;\Leftrightarrow\; \vec{v} = \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-1} \end{pmatrix} \quad \text{mit } v_i \in \mathrm{R} \quad (i = 0; 1; ...; n-1)$$

Viele wichtige Transformationen sind mathematisch *lineare Abbildungen*:

Eigenschaft: $f(cx+y) = c\,f(x) + f(y)$

Eine lineare Abbildung ist vollständig bekannt, wenn die Bilder der Vektoren einer *Basis* bekannt sind (z.B. die Bilder der Standard-Basisvektoren $\vec{e}_1 = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}, \vec{e}_2 = \begin{pmatrix} 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix}, \ldots, \vec{e}_n = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix}$).

Darstellung der linearen Abb. *f* durch ihre zugehörige *Matrix* M_f: in den Spalten von M_f stehen die Bilder der Standard-Basisvektoren unter *f*.

Allgemein besteht eine Matrix aus $p \times q$ Skalaren m_{ij} .

$$M = \begin{pmatrix} m_{00} & m_{01} & \cdots & m_{0,q-1} \\ m_{10} & m_{11} & \cdots & m_{1.q-1} \\ \vdots & \vdots & \ddots & \vdots \\ m_{p-1,0} & m_{p-1,1} & \cdots & m_{p-1,q-1} \end{pmatrix} = (m_{ij})$$

Anwendung einer linearen Abbildung *f* auf einen Vektor $\vec{v}$ durch *Multiplikation* der Matrix M_f mit dem Vektor (dabei Schreibweise des Vektors als Spaltenvektor und Multiplikation der Matrix *von links* an den Vektor):

$$\vec{w} = f(\vec{v}) = M \cdot \vec{v} = \begin{pmatrix} m_{00} & \cdots & m_{0,q-1} \\ \vdots & \ddots & \vdots \\ m_{p-1,0} & \cdots & m_{p-1,q-1} \end{pmatrix} \cdot \begin{pmatrix} v_0 \\ \vdots \\ v_{q-1} \end{pmatrix} = \begin{pmatrix} \sum_{k=0}^{q-1} m_{0,k} \cdot v_k \\ \vdots \\ \sum_{k=0}^{q-1} m_{p-1,k} \cdot v_k \end{pmatrix} = \begin{pmatrix} w_0 \\ \vdots \\ w_{p-1} \end{pmatrix}$$

Nacheinander-Anwendung (*Komposition*) zweier linearer Abbildungen *f* ° *g*: wende erst *g* an, dann *f*.

$$f \circ g\ (x) = f(g(x))$$

Die Komposition wird beschrieben durch das *Produkt* der zugehörigen Matrizen:

$$\mathrm{M_{f \circ g}} = \mathrm{M_f} \cdot \mathrm{M_g}$$

mit

$$M \cdot N = \begin{pmatrix} m_{00} & \cdots & m_{0,q-1} \\ \vdots & \ddots & \vdots \\ m_{p-1,0} & \cdots & m_{p-1,q-1} \end{pmatrix} \cdot \begin{pmatrix} n_{00} & \cdots & n_{0,r-1} \\ \vdots & \ddots & \vdots \\ n_{q-1,0} & \cdots & n_{q-1,r-1} \end{pmatrix} = \begin{pmatrix} \sum_{k=0}^{q-1} m_{0,k} n_{k,0} & \cdots & \sum_{k=0}^{q-1} m_{0,k} n_{k,r-1} \\ \vdots & \ddots & \vdots \\ \sum_{k=0}^{q-1} m_{p-1,k} n_{k,0} & \cdots & \sum_{k=0}^{q-1} m_{p-1,k} n_{k,r-1} \end{pmatrix}$$

Beachte: Das Matrizenprodukt ist nichtkommutativ (d.h. i. allg. ist $N \cdot M \neq M \cdot N$).

Lineare Abbildungen eines Vektorraums (z.B. des $\mathbb{R}^2$) in sich selbst heißen auch *Endomorphismen*. Darstellung durch $n \times n$-Matrizen (n = Dimension des Vektorraumes).

Eindeutig umkehrbare (bijektive) Endomorphismen heißen *Automorphismen*. Ihnen entsprechen die *regulären* Matrizen:

$$f \text{ bijektiv} \Leftrightarrow M_f \text{ regulär} \Leftrightarrow \det(M_f) \neq 0$$

Die regulären $n \times n$-Matrizen (bzw. die Automorphismen des $\mathbb{R}^n$) bilden eine Gruppe, die *allgemeine lineare Gruppe* GL(n).

(math. Begriff der *Gruppe*:

- *Abgeschlossenheit gegenüber der Verknüpfungsoperation*
- Assoziativgesetz $(MN)P = M(NP)$
- Existenz eines neutralen Elements E: $EM = ME = M$

  (hier die Einheitsmatrix $E = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$)
- zu jedem Element M Existenz eines Inversen M^{-1}.)

Beispiele für bijektive lin. Abbildungen:

Drehungen um den Nullpunkt, zentrische Streckungen mit einem Faktor $\neq 0$ vom Nullpunkt aus.

Beispiel für eine nicht-bijektive lin. Abb.: Projektion entlang der x-Achse auf die y-Achse.

Wichtiger Spezialfall der bijektiven lin. Abb.:
orthogonale Abbildungen.
Entsprechend: orthogonale Matrizen.

M orthogonal $\Leftrightarrow$ $M^{\mathsf{T}} \cdot M = E$ (d.h. die Transponierte ist die Inverse)

$\Leftrightarrow$ die Spaltenvektoren von M bilden eine Orthonormalbasis (paarw. senkrecht u. Länge 1)

$\Leftrightarrow$ $\forall \vec{x}, \vec{y} \in R^n : \langle M\vec{x}, M\vec{y} \rangle = \langle \vec{x}, \vec{y} \rangle$ (Skalarprodukt-Invarianz)

$\Leftrightarrow$ $\forall \vec{x} \in R^n : \|M\vec{x}\| = \|\vec{x}\|$ (Längen-Invarianz)

Für alle orthogonalen Matrizen hat ferner die Determinante den Betrag 1.
(Die Umkehrung gilt nicht: Beispiel Scherungsmatrizen.)

Die orthogonalen $n \times n$-Matrizen bilden eine Untergruppe von GL(n), die *orthogonale Gruppe* O(n).

Alle Elemente von O(n) mit Determinante +1 bilden wiederum eine Untergruppe von O(n), die *spezielle orthogonale Gruppe* SO(n).

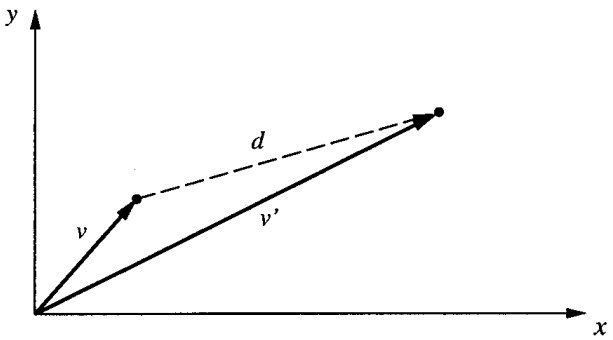
Es gilt:
$M \in$ SO(n) $\Leftrightarrow$ M ist Produkt einer *geraden Anzahl* von Spiegelungen (an Hyperebenen durch den Nullpunkt).

Beispiele:

- Drehungen um den Nullpunkt gehören zu SO(n).
- Spiegelungen an Geraden durch den Nullpunkt (im R^2) bzw. an Ebenen durch den Nullpunkt (im R^3) sind orthogonal, aber gehören nicht zu SO(n).
- Streckungen mit Streckfaktor $c \neq \pm 1$ sind nicht orthogonal.

Einschränkung: für jede lineare Abbildung f gilt $f(\vec{0}) = \vec{0}$ (Nullpunkt ist Fixpunkt).

wichtige Transformationen, die nicht zu den linearen Abbildungen gehören:
Translationen (Verschiebungen).



Verschiebungen rechnerisch:
$v' = t(v) = v + d_t$,
d.h. Addition eines konstanten Vektors d_t.

Affine Abbildungen:

alle Abbildungen a, die sich darstellen lassen als

$a(v) = f(v) + d$ mit einer linearen Abbildung f,

d.h. als Komposition einer linearen Abb. und einer Translation.

Es gilt: $d = a(0)$, $f(v) = a(v) - a(0)$
$\Rightarrow$ f und d sind eindeutig bestimmt.

Beispiele:

- Drehungen um beliebige Punkte
- Streckungen mit beliebigen Streckzentren
- Schraubungen im R^3
- Spiegelungen an beliebigen Geraden im R^2 oder an beliebigen Ebenen im R^3
- Parallelprojektionen auf beliebige Ebenen im R^3

usw.
(lineare Abbildungen und Translationen als Spezialfälle mit enthalten!)

Die bijektiven affinen Abbildungen des R^n in sich bilden eine Gruppe, die *affine Gruppe* Aff(n).

Eine affine Abb. a heißt *Bewegung* oder *Kongruenzabbildung*, wenn ihre zugehörige lineare Abbildung f orthogonal ist,
und *eigentliche Bewegung*, wenn f sogar in SO(n) liegt.

Beispiele:
Drehungen, Schraubungen, Translationen:
eigentliche Bewegungen
Spiegelungen, Drehspiegelungen:
uneigentliche Bewegungen
Streckungen: keine Bewegungen,
da $f(v) = c \cdot v$ nicht orthogonal (wenn $|c| \neq 1$).

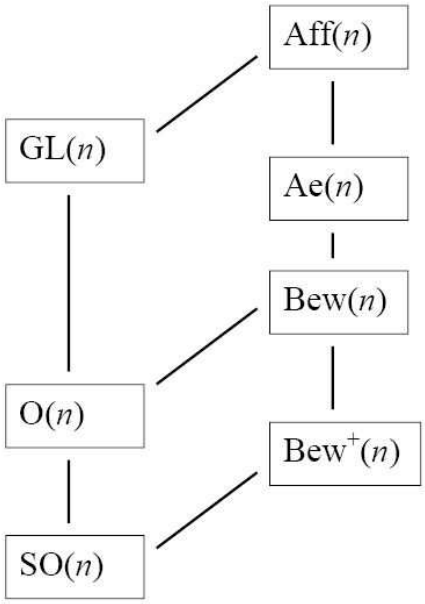
Bew(n): Gruppe der Bewegungen des R^n (= Untergruppe von Aff(n))
$Bew^+(n)$: Untergruppe der eigentlichen Bewegungen.

Eine *Ähnlichkeitsabbildung* ist eine affine Abb. a, die Abstände stets um einen festen Faktor $c > 0$ verändert:
$d(a(u), a(v)) = c \cdot d(u, v)$.

Ähnlichkeitsabbildungen liegen "zwischen" den affinen Abbildungen und den Kongruenzabbildungen (Bewegungen) (für Bewegungen ist $c = 1$).

Jede Ähnlichkeitsabb. hat die Form $a(v) = c \cdot M \cdot v + d$ mit einer orthogonalen Matrix M und $c > 0$.
Ae(n): Gruppe der Ähnlichkeitsabb. des R^n.
Beispiele: Streckungen, Streckspiegelungen, Drehstreckungen.

Untergruppen-Diagramm:



Man kann affine Abbildungen aus bestimmten Grundoperationen zusammensetzen

Beispiel: Drehung um einen beliebigen, vorgegebenen Punkt *P* im R^2

- verschiebe den Drehpunkt in den Nullpunkt
- wende Drehung um den Nullpunkt an
- mache Verschiebung wieder rückgängig

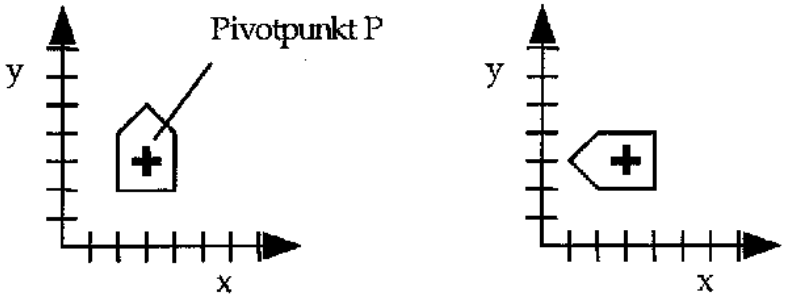
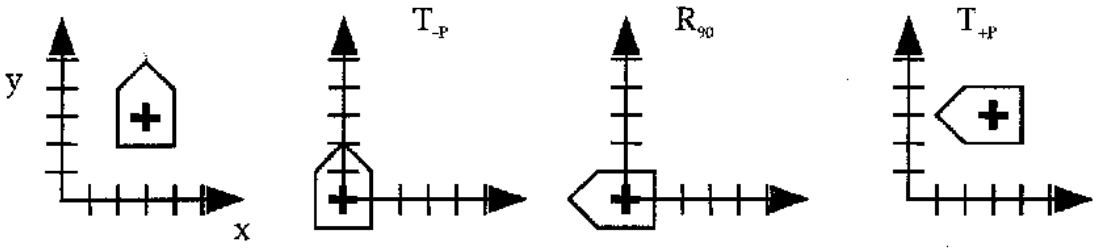


Abb. 2-20 Drehung um Fixpunkt



Man benötigt für Bewegungen im R^2:

- alle Translationen
- Drehungen um den Nullpunkt um alle Winkel
- eine Spiegelung

im R^3:

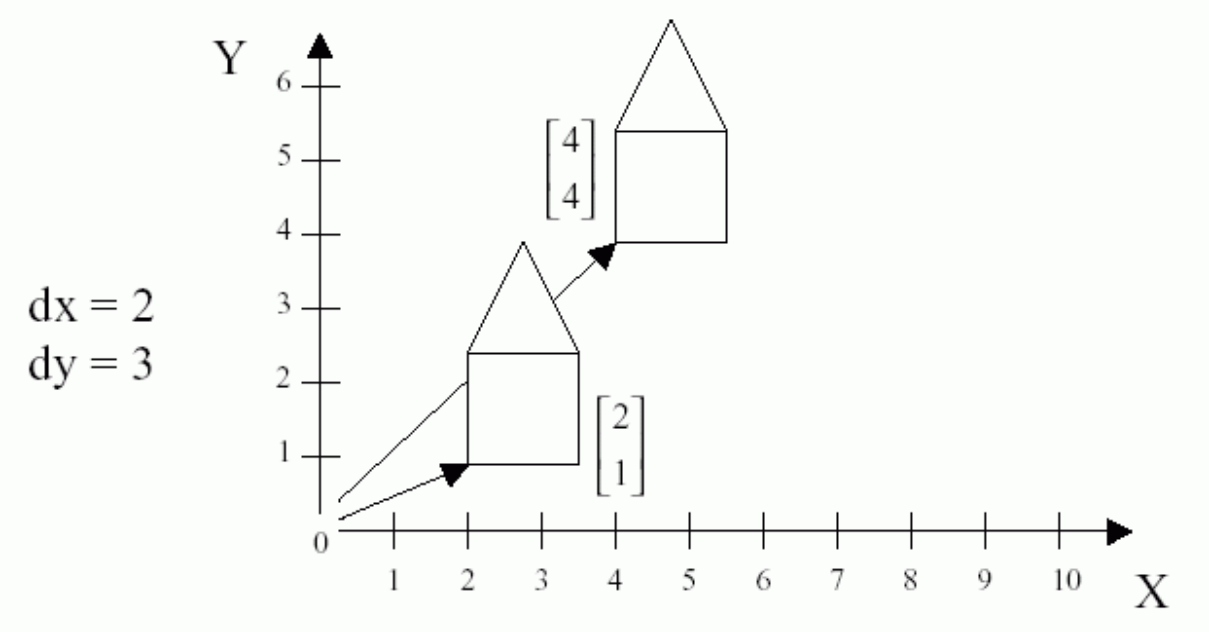
- alle Translationen
- Drehungen um die 3 Koordinatenachsen (alle Winkel)
- eine Spiegelung an einer Ebene

Die benötigten Winkel für die Drehungen um die Koordinatenachsen (im R^3) heißen auch die *Euler-Winkel* der Transformation.

Wichtige Grundoperationen im R^2 sind somit:

(a) Translationen

$v' = v + d$



(b) Drehungen (Rotationen)

$$v' = R_\theta v \text{ where } \quad v = \begin{bmatrix} x \\ y \end{bmatrix}, \quad v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

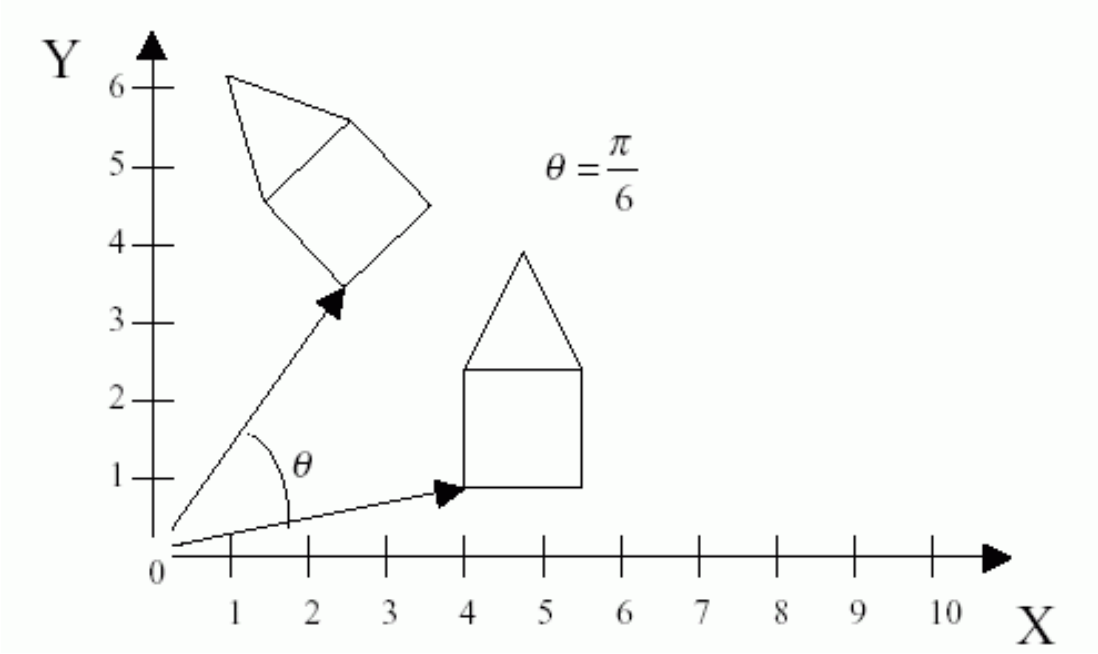
$$R_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

and

$$x' = x\cos\theta - y\sin\theta$$

$$y' = x\sin\theta + y\cos\theta$$

θ ist der Drehwinkel
Orientierung im math. positiven Sinne (gegen den Uhrzeigersinn)
Angabe manchmal auch in Bogenmaß:
Winkel [Bogenmaß] = Winkel [Grad] $\cdot \pi$ / 180.



(c) Spiegelungen

Spiegelung an der *x*-Achse:
Vorzeichenwechsel in der *y*-Koordinate

also $v' = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot v$

Spiegelung an der *y*-Achse:

analog, $v' = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \cdot v$

(d) Skalierungsoperationen

oft mit unterschiedlichen Skalierungsfaktoren in *x*- und in *y*-Richtung gebraucht (keine Kongruenzabb.!):

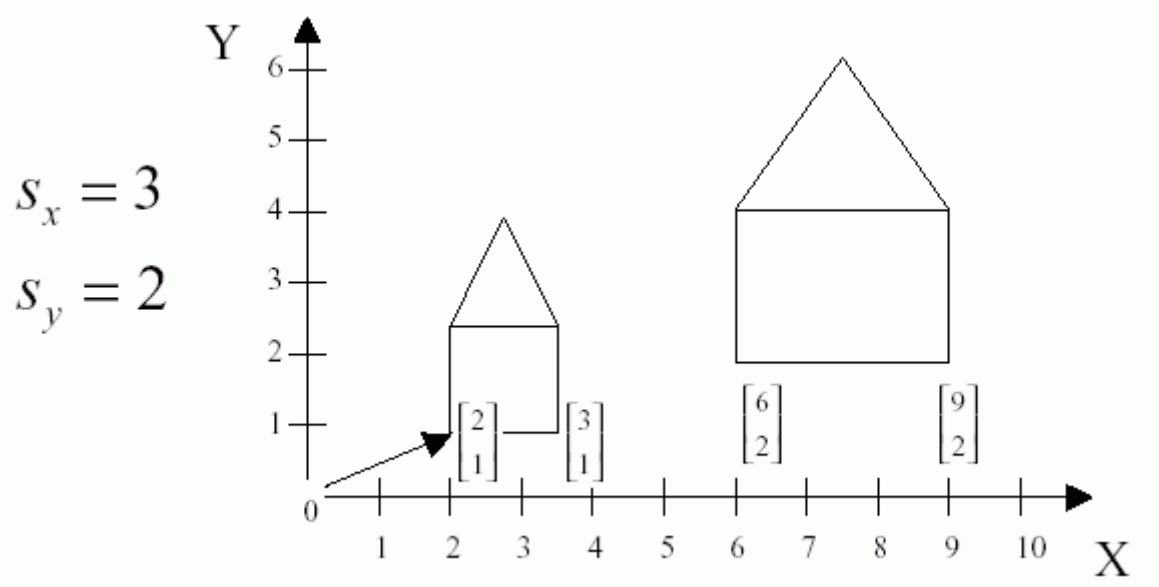
$v' = Sv$ where $v = \begin{bmatrix} x \\ y \end{bmatrix}, \quad v' = \begin{bmatrix} x' \\ y' \end{bmatrix}$

and $S = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix}$

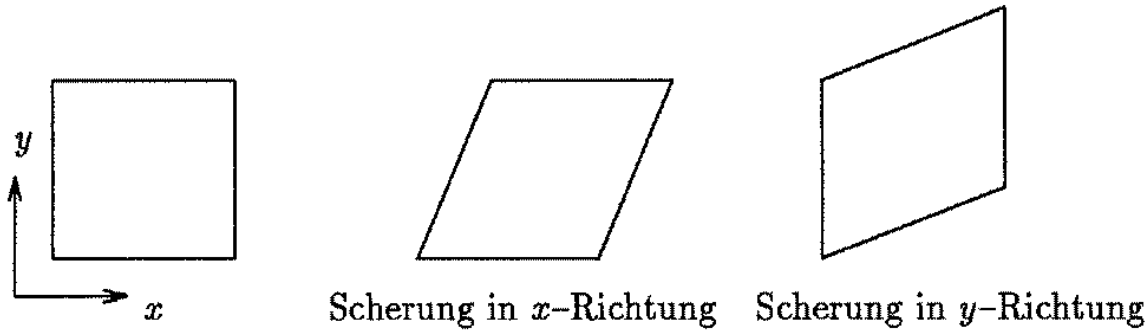
$x' = s_x x$

$y' = s_y y$

Proportionen und Winkel bleiben nur erhalten, wenn $s_x = s_y$
(dann liegt eine Ähnlichkeitsabb. vor)
andernfalls Verzerrung des abgebildeten Objekts:



Manchmal benötigt man auch
(e) Scherungen (engl.: *shear*)



Scherungen sind flächenerhaltend, aber nicht orthogonal
(Figuren werden verzerrt).

Matrix einer Scherung in *x*-Richtung:

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

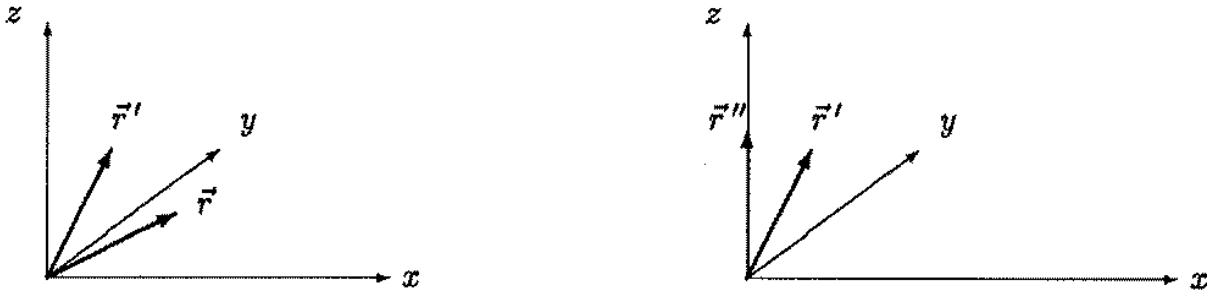
für die *y*-Richtung analog.

Im R^3 analoge Grundoperationen;

- anstelle der Spiegelung an Koordinatenachsen: Sp. an Koordinatenebenen (*xy*, *xz*, *yz*)
- anstelle der Drehung um den Nullpunkt: Drehungen um die Koordinatenachsen $R_{x,\alpha}$, $R_{y,\alpha}$, $R_{z,\alpha}$

Rückführung einer Drehung um eine beliebige Achse *AB* mit Winkel θ auf diese Grund-Drehungen:

1.	Translation der Achse um $-A$ zum Nullpunkt	T_{-A}
2.	Rotation der Drehachse *r* in die *yz*-Ebene (drehe um die *y*-Achse mit Winkel $-\alpha$)	$R_{y,-\alpha}$
3.	Rotiere die Drehachse auf die z-Achse (drehe um die *x*-Achse mit Winkel β)	$R_{x,\beta}$
4.	Rotiere um die z-Achse mit Winkel θ	$R_{z,\theta}$
5.	Invertiere Transformation von Schritt 3	$R_{x,-\beta}$
6.	Invertiere Transformation von Schritt 2	$R_{y,\alpha}$
7.	Invertiere Transformation von Schritt 1	T_A



Veranschaulichung der Schritte 2 und 3. *r* ist der Richtungsvektor der Rotationsachse, um die die Rotation erfolgen soll. Links: Drehung um die *y*-Achse, die *r* in die *yz*-Ebene dreht (Ergebnis *r*'). Rechts: Drehung von *r*' um die *x*-Achse, so dass er in Richtung der positiven *z*-Achse zeigt.

Die Winkel α und β lassen sich durch trigonometrische Beziehungen aus den Koordinaten von *A* und *B* berechnen.

Koordinatentransformationen

Oft muss man von einem Koordinatensystem in ein anderes umrechnen
Beispiel:
Weltkoordinaten → normierte Koordinaten → Gerätekoord.

Koordinatentransformation: komplementär zur Transformation von Objekten

Objekttransformation:	Objekt ändert seine Lage (und evtl. seine Form) Koordinatensystem bleibt fest
Koordinatentransformation:	Objekt bleibt fest Koordinatensystem ändert seine Lage (u. evtl. seine Form)

rechnerisch selber Formalismus:
Multiplikation des Koordinatenvektors mit Transformationsmatrix, ggf. + Translationsanteil (wenn Ursprünge der Koordinatensysteme unterschiedlich)

wenn Translationsanteil entfällt:
Spalten der Matrix = Basisvektoren des alten Koordinaten-systems, ausgedrückt durch Koordinaten im neuen System

Unterschiede zu Objekttransformationen:

- Reihenfolge von Objektkonstruktion und Transformation vertauscht
- Änderung des Koordinatensystems beeinflusst *alle* nachfolgenden Konstruktionen

Homogene Koordinaten

Darstellung affiner Abbildungen und Koordinatentransformationen in der Form $M \cdot v + d$ macht Berechnungen schwerfällig.

Idee: alles in der Form $M \cdot v$ darstellen
$\Rightarrow$ bei Komposition von Transformationen nur noch Matrixmultiplikationen,
einheitliche Form aller affinen Abb.

dies wird erreicht durch *Hinzunahme einer weiteren Koordinate*

- aus (x, y) wird (x, y, 1)
- aus (x, y, z) wird (x, y, z, 1)

dafür werden Vektoren, die sich nur um einen festen Faktor $w \neq 0$ unterscheiden, miteinander identifiziert (Klassenbildung).

"*homogene Koordinaten*"

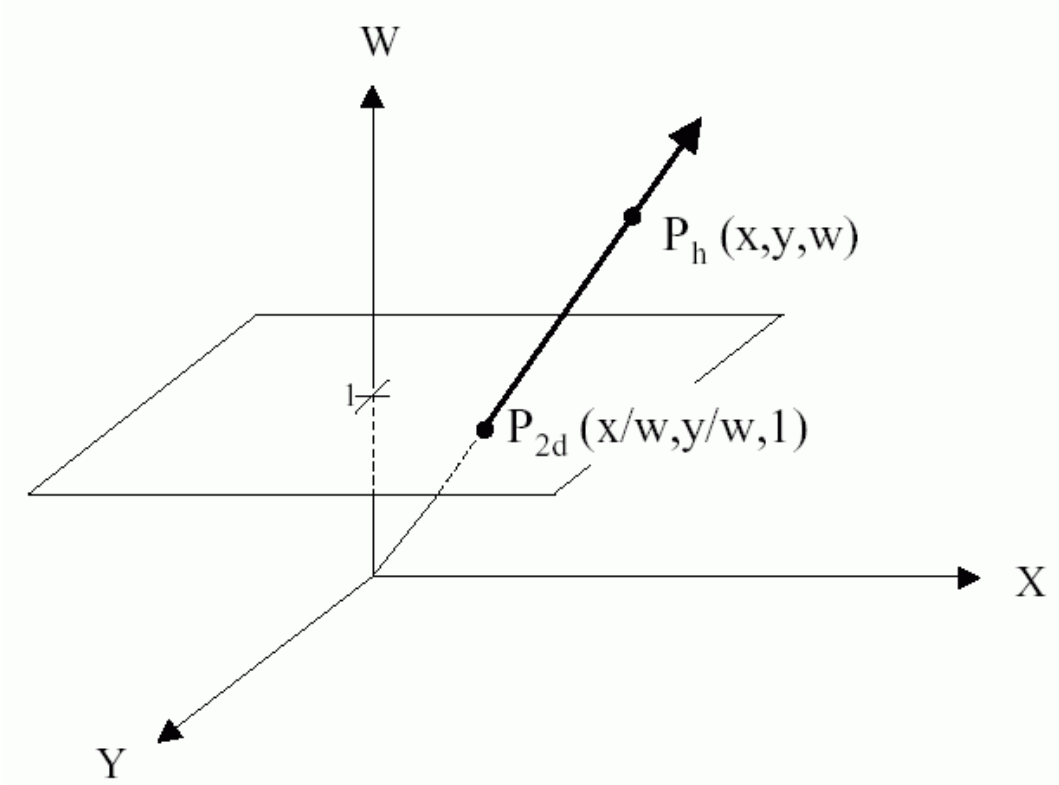
dem homogenen Vektor (x, y, w) entspricht der kartesische Vektor (x/w, y/w).

$$P_{2d}(x, y) \rightarrow P_h(wx, wy, w), \quad w \neq 0$$

$$P_h(x', y', w), \quad w \neq 0$$

$$P_{2d}(x, y) = P_{2d}\left(\frac{x'}{w}, \frac{y'}{w}\right)$$

Jeder Punkt auf der Geraden durch den Nullpunkt und durch (*x*/*w*, *y*/*w*, 1) repräsentiert im homogenen Koordinatensystem denselben kartesischen Punkt (*x*, *y*) aus dem gewöhnlichen 2D-Koordinatensystem:



In der Tat lassen sich nun die Translationen gleichwertig mit allen anderen affinen Abb. als Produkte "Matrix mal Vektor" berechnen:

$$T(dx,dy)=\begin{bmatrix}1 & 0 & dx\\ 0 & 1 & dy\\ 0 & 0 & 1\end{bmatrix};\quad v'=T(dx,dy)v$$

$$S(s_x,s_y)=\begin{bmatrix}s_x & 0 & 0\\ 0 & s_y & 0\\ 0 & 0 & 1\end{bmatrix};\quad v'=S(s_x,s_y)v$$

$$R(\phi)=\begin{bmatrix}\cos\phi & -\sin\phi & 0\\ \sin\phi & \cos\phi & 0\\ 0 & 0 & 1\end{bmatrix};\quad v'=R(\phi)v$$

Beispiel:
Verschiebe mittels homogener Koordinaten den Punkt (2; 3) um den Vektor (7; 5) und drehe anschließend das Ergebnis um 90° um den Ursprung.

$T(7; 5)$: $\begin{pmatrix} 1 & 0 & 7 \\ 0 & 1 & 5 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 3 \\ 1 \end{pmatrix} = \begin{pmatrix} 9 \\ 8 \\ 1 \end{pmatrix}$, entspr. (9; 8)

$R(90°)$: sin 90° = 1, cos 90° = 0, somit

$$\begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 9 \\ 8 \\ 1 \end{pmatrix} = \begin{pmatrix} -8 \\ 9 \\ 1 \end{pmatrix} = v'$$, entspr. (–8, 9)

zusammen: $v' = R(90°) \cdot T(7;9) \cdot v$.

Analog im dreidimensionalen Fall:
vierte, homogenisierende Koordinate,
Darstellung der affinen Transformationen durch 4×4-Matrizen.

Grund-Transformationen:

- Translation $\begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Scaling $\begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$

- Rotation about X-axis

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation about Y-axis

$$\begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Rotation about Z-axis

$$\begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

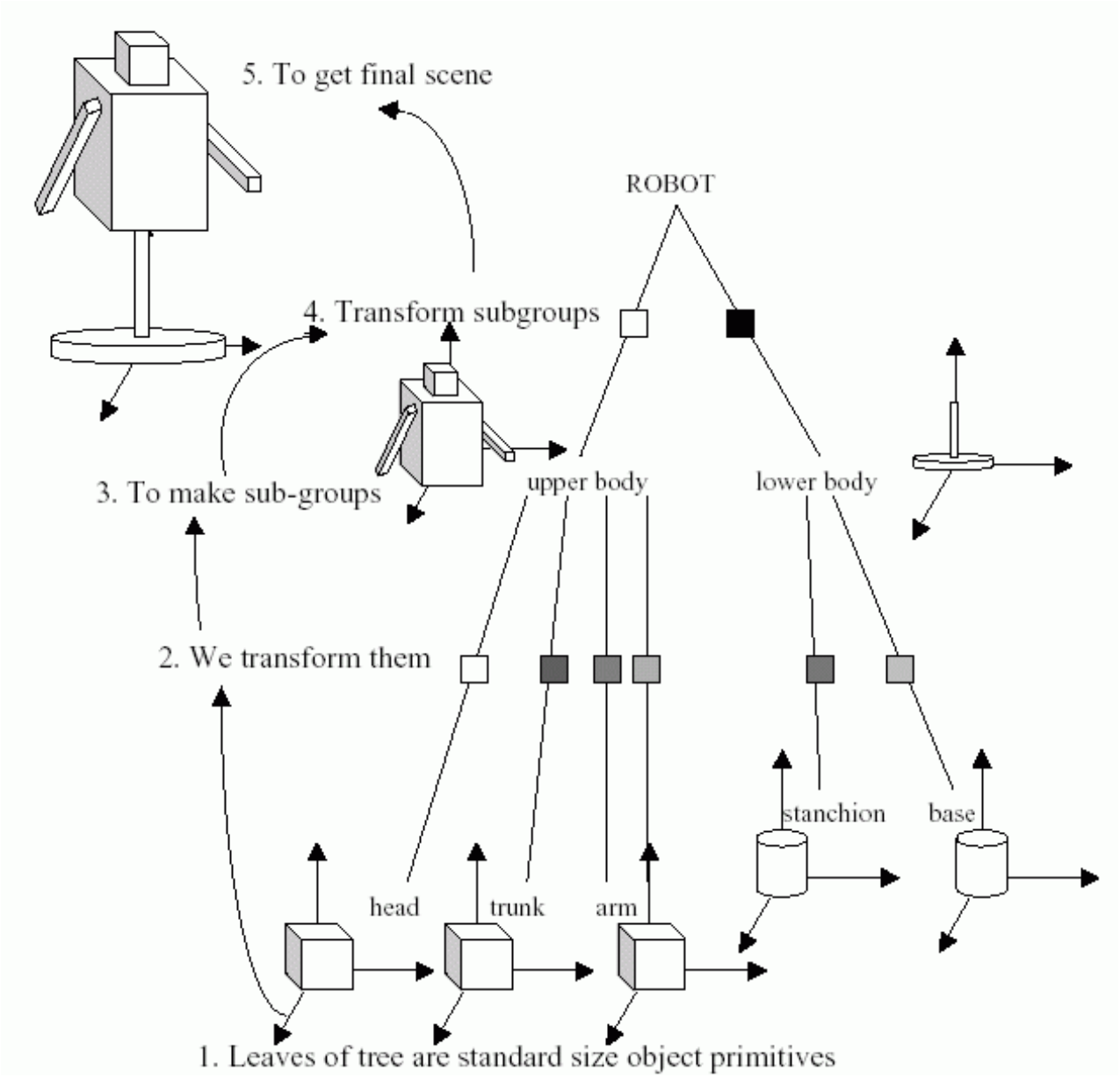
Anwendung dieser Transformationen in Szenengraphen

3D-Szenen werden oft in gerichteten, azyklischen Graphen abgelegt, den sog. *Szenengraphen*, die z.B. durch Aufrufe einer Spezialbibliothek erzeugt werden.

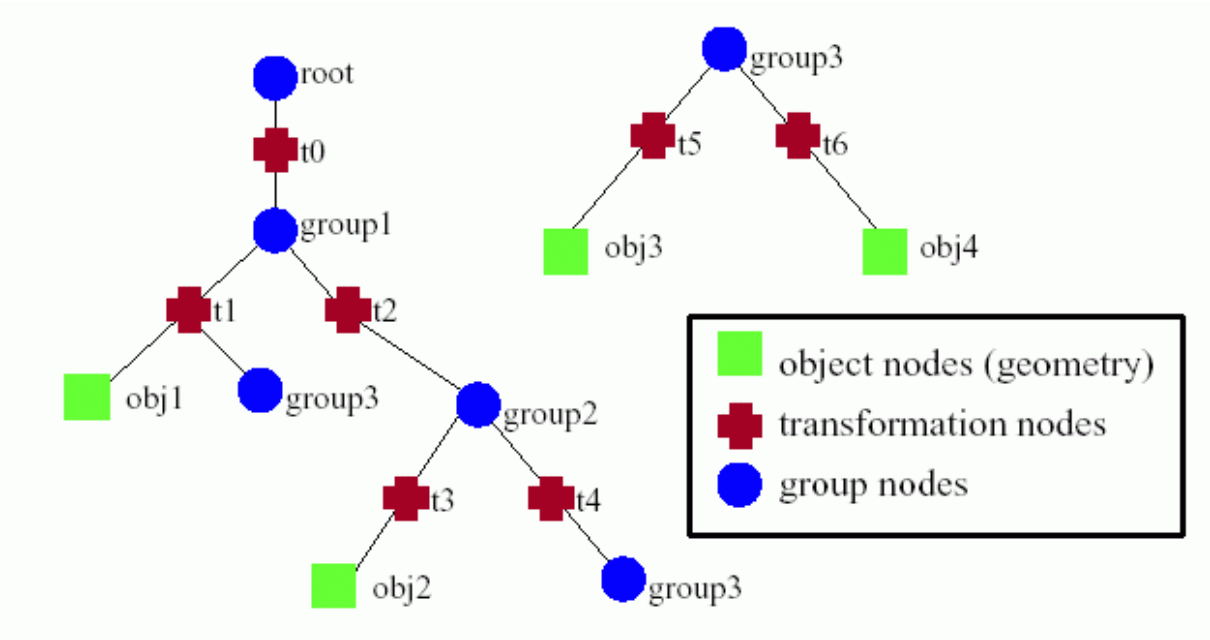
Typisches Szenengraphen-Format (es gibt zahlreiche Varianten!):

- Graphische Primitive (grundlegende Objekte: Würfel, Kugel, Kegel etc.) mit *default*-Eigenschaften (Position am Koordinatenursprung, Einheitsvolumen...) bilden Blattknoten des Graphen
- Attribute (Farbe, Textur etc.) und Transformationen sind ebenfalls Knoten des Graphen und werden auf Objektknoten je nach deren Lage im Graphen angewandt.

Beispiel (aus vanDam 2001):



im folgenden Szenengraphen beeinflusst Transformation t0 alle Objekte,
aber t2 beeinflusst nur obj2 und eine Instanz von group3 (welche je eine Instanz von obj3 und obj4 enthält).
t2 wirkt nicht auf obj1 und auf die andere Instanz von group3.



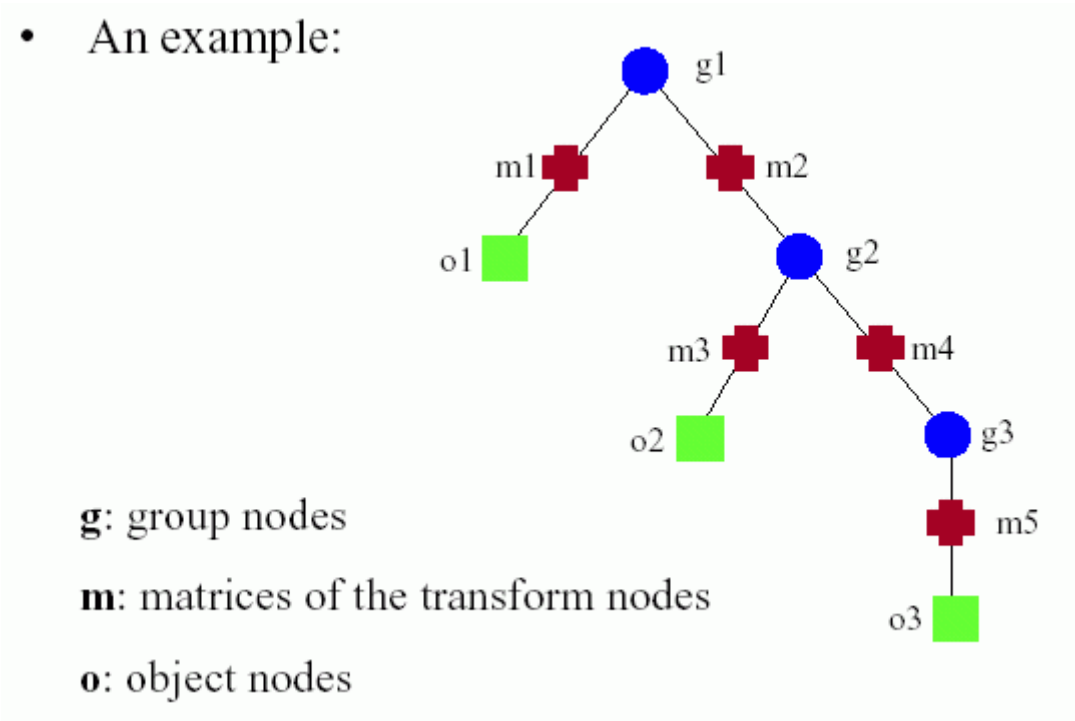
Transformationsknoten enthalten normalerweise mindestens eine Matrix (für homogene Koordinaten-darstellung), die die Transformation beschreibt; evtl. noch weitere Parameter.

Bestimmung der Komposition der Transformationsmatrizen (CTM: *composite transformation matrix*), die auf einen Objektknoten anzuwenden ist:

Multiplikation der Matrizen in den Transformationsknoten bei top-down-Durchlauf des Baumes

(Details abhängig vom Grafikpaket).

Beispiel (aus vanDam 2001):



CTM(o1) = m1
CTM(o2) = m2 · m3
CTM(o3) = m2 · m4 · m5

für einen Punkt *v* im Objekt o3 errechnet sich seine Position im Weltkoordinatensystem (gültig an der Wurzel des Baumes) als:

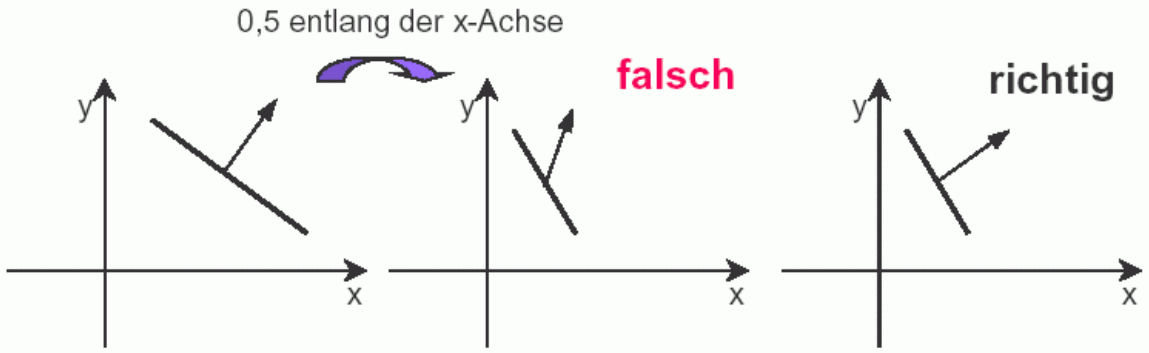
CTM·*v* = (m2 · m4 · m5) · *v*.

Flächen werden in der Computergrafik oft zusammen mit Normalenvektoren abgespeichert (orthogonal zur Fläche stehender Vektor – Anwendung z.B. in der Beleuchtungsrechnung).

Vorsicht bei der Anwendung von Transformationen auf solchen Normalenvektoren!

Transformation von Normalen

- Matrizen können Punkte, Linien, Polygone, ... transformieren
- Vorsicht bei **Normalen!**



Wird eine Matrix **M** benutzt um Geometrie zu transformieren, dann müssen zugehörige Flächennormalen mit

$$\mathbf{N} = \left(\mathbf{M}^{-1}\right)^{T}$$

transformiert werden.

Hier ohne Beweis angegeben, siehe z.B. Turkowski´s Beitrag in *Graphics Gems, A. Glassner (Edt.)*

(Krömker 2001)

Betrachtungstransformationen (*Viewing transformations*)

In der Regel wird bei der Implementierung der Projektion in die Bildschirmebene auch das Clipping vorbereitet.

Anstelle der Projektion auf eine Ebene und einem Clippen in dieser Ebene transformiert man in ein *kanonisches Sichtvolumen* (Sichtkörper, i.allg. ein Quader) und clippt an dessen Seitenflächen.
Dann: Projektion in Ebene auf einheitliche Weise.

Im Folgenden Darstellung nach van Dam (2001) und Schlechtweg (2001).

Zugrundegelegtes Modell bei der Berechnung des sichtbaren Bereichs:

Die synthetische Kamera

- Referenzmodell zur Spezifikation von Betrachtungstransformationen
- allgemeines Modell:
  - Position der Kamera
  - Orientierung
  - field of view (Weitwinkel, Normal, Tele)
  - depth of field (Fokus)
  - Brennweite
  - Neigung der Projektionsebene
  - Projektionsart
- aber: hier einfacheres, etwas weniger mächtiges Modell

Der Kamera wird (in einer bestimmten Position und Blickrichtung) ein "Sichtkörper" zugeordnet.

Der Sichtkörper

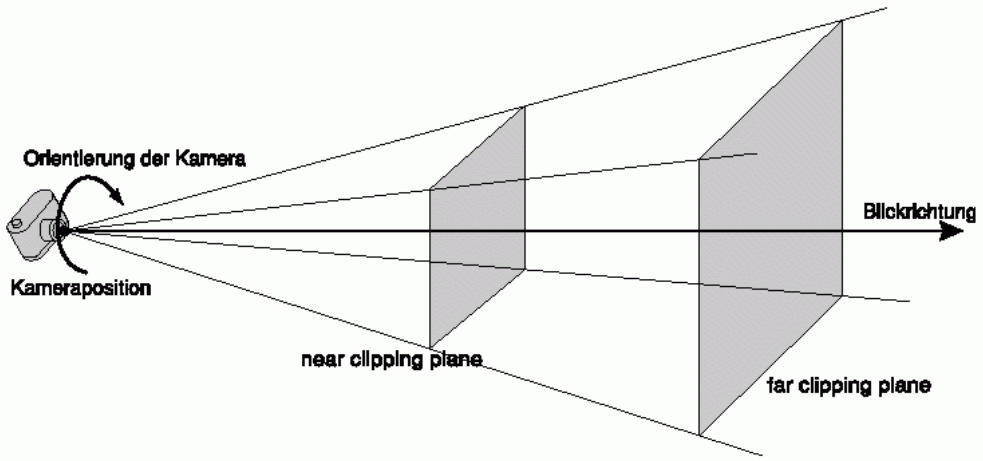
- engl.: view volume
- Sichtkörper enthält alles, was vom COP (oder DOP) aus sichtbar ist (=was die Kamera sieht)
- normalerweise ein Kreiskegel → teure Berechnungen, insbesondere beim Clipping von Objekten gegen eine Kegelfläche
- Approximation durch Pyramide(nstumpf)
  - die Ausgabe ist sowieso rechteckig
  - Clipping durch Lösen eines linearen Gleichungssystems
- Pyramide(nstumpf) = view frustum

Vereinfachtes Kameramodell

Sechs Parameter zur Spezifikation der Kamera

1. Position der Kamera
2. (Blick-)Richtung der Kamera
3. Orientierung der Kamera
4. Seitenverhältnis des zu erzeugenden Bildes
5. Öffnungswinkel
6. near clipping plane und far clipping plane
7. optional: Brennweite

Sichtkörper für dieses Modell



Kameraposition

- analog zur Entscheidung eines Photographen, wo die Kamera aufgestellt wird
- Spezifikation durch einen Punkt in 3D

Orientierung der Kamera

Spezifikation durch zwei Angaben

- Blickrichtung
  - Angabe eines Zielpunktes,
    Berechnung der Richtung aus Zielpunkt – Position
  - Richtung, in die die Kamera „schaut“
  - Spezifikation durch einen Vektor in 3D
- up vector
  - bestimmt die Rotation der Kamera um die Blickrichtung
  - Projektion des up vectors muß in der Ebene senkrecht zur Blickrichtung liegen, d. h. Blickrichtung und up vector dürfen nicht kollinear sein

Seitenverhältnis

- engl.: aspect ratio
- analog zum Seitenverhältnis des Films in der Photographie
- Verhältnis von Breite zu Höhe
- für quadratische Bilder: $1:1$
- Kinofilme: $2:1$
- Fernsehen: $4:3$
- HDTV: $16:9$

Öffnungswinkel

- analog zur Auswahl des Objektivs beim Photographieren
- bestimmt die Stärke der perspektivischen Verzerrung
  - keine Verzerrung – Paralleprojektion
  - sehr starke Verzerrung – Weitwinkellinsen
- eigentlich zwei Öffnungswinkel im view frustum: horizontal und vertikal
- sollten übereinstimmen (Spezifikation eines Winkels, der andere kann aus Seitenverhältnis berechnet werden)

Clipping planes

- Raum zwischen near clipping plane und far clipping plane bestimmt, was die Kamera sieht
- Position der Ebenen durch Entfernungen entlang der Blickrichtung festgelegt
- Objekte außerhalb dieser Ebenen werden nicht gezeichnet
- diese Ebenen schneidende Objekte müssen geclippt werden

Gründe für die Verwendung der Clipping planes

- near clipping plane
  - Objekte zu nah an der Kamera sollen nicht dargestellt werden, da:
    - Sicht auf Rest der Szene blockiert würde
    - zu starke Verzerrung
  - Objekte hinter der Kamera sollen nicht dargestellt werden
- far clipping plane
  - Objekte zu weit von der Kamera entfernt sollen nicht dargestellt werden, da sie visuell nicht signifikant (kaum erkennbar) sind, aber in die Berechnungen einbezogen werden müssen

Kameramodell . . .

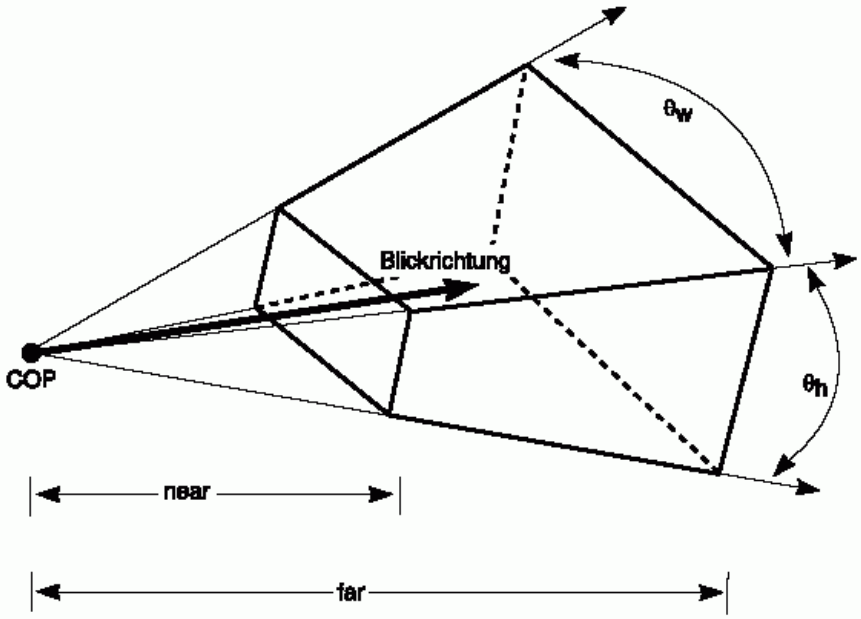
- . . . kann Sichtkörper für folgende Projektionen erstellen:
  - perspektivische Projektion mit positivem Öffnungswinkel
  - Parallelprojektion (Öffnungswinkel ist Null)
- . . . kann nicht für schräge Projektionen verwendet werden

Welche Transformation ist rechnerisch anzuwenden, um die Betrachtungstransformation (Abbildung in den Bildschirmausschnitt) durchzuführen?

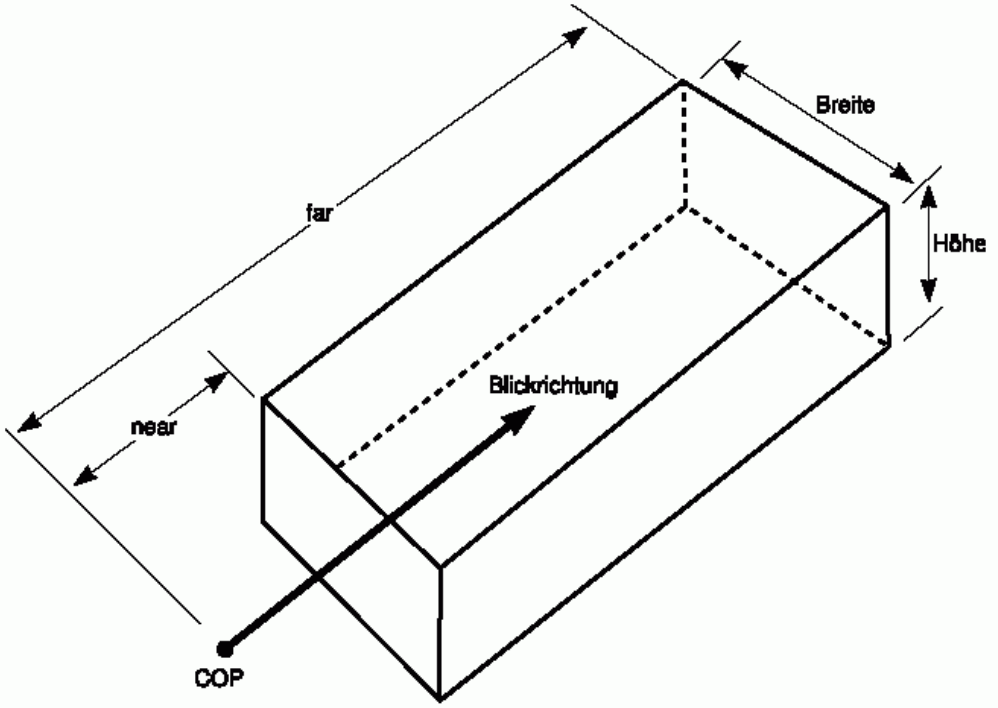
Ziel: Herleitung einer Transformationsmatrix (für Objektdarstellung in homogenen Koordinaten)

- Vorgehensweise:
  - Der kanonische Sichtkörper
  - Spezifikation des Sichtkörpers (bereits behandelt)
  - Transformation des spezifizierten Sichtkörpers in den kanonischen Sichtkörper
  - Projektion und Clipping an diesem kanonischen Sichtkörper

Sichtkörper für perspektivische Projektionen

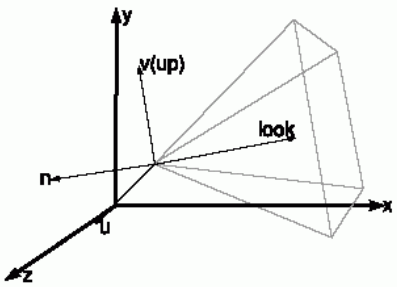


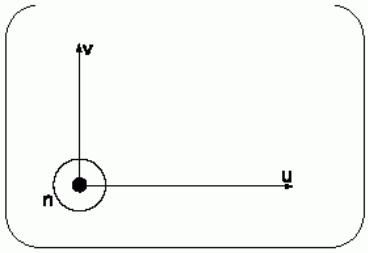
Sichtkörper für Parallelprojektionen



Koordinatensysteme

- Weltkoordinaten – Definition des Modells, der Kameraparameter
- View-Referenz-Koordinaten (Kamerakoordinaten) – Ursprung im COP, Achsen rotiert je nach Orientierung der Kamera

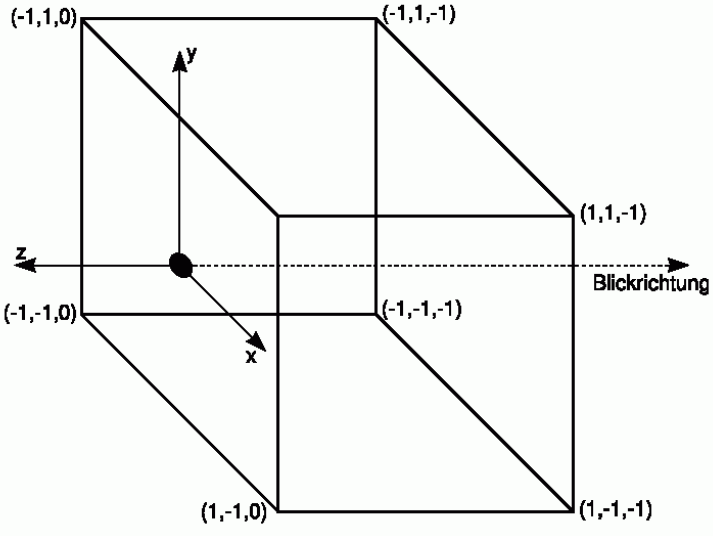




Der kanonische Sichtkörper

- beliebiger Sichtkörper zu komplex (Clipping und Projektion)
- Bilderzeugung einfacher vom kanonischen Sichtkörper aus:
  - Parallelprojektion
  - Kamera im Ursprung (Position: $(0, 0, 0)$)
  - Sicht entlang der negativen z-Achse (look vector: $(0, 0, -1)$)
  - aufrecht orientiert (up vector: $(0, 1, 0)$)
  - Ausdehnung der Bildebene zwischen -1 und 1 in x und y
- für perspektivische Projektion ein zusätzlicher Schritt: Unwandlung des perspektivischen Sichtkörpers in den der Parallelprojektion

Normalisierungstransformation



Ziel: Transformation eines beliebigen Sichtkörpers in den kanonischen Sichtkörper

- für Parallelprojektion: Ähnlichkeitstransformation (aus Translation, Rotation und Skalierungen zusammengesetzt)
- für perspektivische Projektion: zusätzlich eine Transformation, die keine Ähnlichkeitsabb. ist (Verzerrung des Sichtkörpers Pyramidenstumpf $\rightarrow$ Quader)
- gesamte Transformation kann durch eine 4×4-Matrix (für homogene Koordinaten) ausgedrückt werden
- Clipping wird danach einfach (Clipping-Ebenen sind achsenparallel)
- Projektion wird einfach (nur noch Wegfall der *z*-Koordinate).

Normalisierung

- Problem der Berechung der Projektion wurde reduziert auf das Problem der Bestimmung einer korrekten Normalisierungstransformation
- Schwierigkeit: Bestimmen der Rotationskomponente der Normalisierungstransformation – *Aber:* einfacher, die Inverse dieser Rotation zu finden
- Umweg: Suche nach der Inversen der Rotationskomponente der Normalisierungstransformation $\rightarrow$ *viewing transformation* (transformiert den kanonischen Sichtkörper in einen beliebigen Sichtkörper)

Viewing Transformation

- wir kennen Position (COP), Blickrichtung und up vector
- Herleitung einer affinen Transformation, die den kanonischen Sichtkörper so verschiebt und rotiert, daß er mit dem gegebenen Sichtkörper übereinstimmt (zunächst ohne Skalierung)
- Translation einfach: Verschiebung des Ursprungs zum COP:

$$T_{\mathrm{COP}} = \begin{pmatrix} 1 & 0 & 0 & x_{\mathrm{COP}} \\ 0 & 1 & 0 & y_{\mathrm{COP}} \\ 0 & 0 & 1 & z_{\mathrm{COP}} \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (1)$$

nächster Schritt: Rotation

Die gesuchte Rotationsmatrix *M* soll die Basisvektoren der Weltkoordinaten *x*, *y*, *z* in die Basis des Kamera-Koordinatensystems überführen

die inverse Matrix ist einfach zu beschreiben: sie enthält die Basisvektoren des Kamera-Koordinatensystems (dargestellt in Weltkoordinaten) als Spaltenvektoren! (siehe Koordinatentransformationen im letzten Kapitel).

wir nutzen nun aus: *M* ist orthogonal (als Matrix einer Bewegung) $\Rightarrow M^{-1} = M^{T}$, die Inverse stimmt mit der Transponierten überein (und ist damit ganz einfach zu bestimmen)!

- Rotationsmatrizen transformieren die paarweise senkrecht zueinander stehenden Einheitsvektoren e_1, e_2 und e_3 in neue paarweise senkrecht zueinander stehenden Einheitsvektoren v_1, v_2 und v_3.
- Weltkoordinatenachsen x, y und z: e_1, e_2 und e_3
- Kamarakoordinatenachsen u, v und n: v_1, v_2 und v_3
- gesucht: Matrix für die Transformation $(u, v, n) \rightarrow (x, y, z)$
- gefunden: Matrix M für Transformation $(x, y, z) \rightarrow (u, v, n)$
- $\rightarrow$ Matrix M^T transformiert (u, v, n) nach (x, y, z), Zeilen dieser Matrix sind u, v und n
- $\rightarrow$ neues Problem: Bestimmen von u, v und n aus COP, Blickrichtung und up vector

Bestimmen von u, v und n

- Vom kanonischen Sichtkörper bekannt:
  - COP liegt im Ursprung $(0, 0, 0)$
  - Blickrichtung zeigt entlang der negativen z-Achse
  - up vector zeigt entlang der y-Achse
- viewing transformation transformiert die (x, y, z)-Achsen in die (u, v, n)-Achsen des gegebenen Sichtkörpers
- daher folgende Eigenschaften der (u, v, n)-Achsen:
  - gegebene Blickrichtung zeigt entlang der negativen n-Achse
  - Projektion des up vectors in die Ebene senkrecht zur n-Achse zeigt entlang der v-Achse

Bestimmen von n

Die Berechnung von n ist relativ einfach.

- Blickrichtung liegt im kanonischen Sichtkörper entlang der z-Achse
- z wird auf n abgebildet
- n ist ein normalisierter Vektor, der in die entgegengesetzte Richtung der gegebenen Blickrichtung zeigt

$$n = \frac{-\overrightarrow{\text{Look}}}{\|\overrightarrow{\text{Look}}\|} \tag{8}$$

Bestimmen von v

- auf den ersten Blick: v sollte mit up vector übereinstimmen
- im kanonischen Sichtkörper: up vector entlang der y-Achse rechtwinklig zur Blickrichtung
- *Aber:* im gegebenen Sichtkörper up vector und Blickrichtung müssen nicht senkrecht zueinander sein
- $\rightarrow v$ muß rechtwinklig zu n „gemacht werden“
- Subtraktion des Anteils vom up vector, der entlang der Blickrichtung liegt.
- dieser Anteil entspricht dem Skalarprodukt zwischen up vector und Blickrichtung

$$v = \frac{\overrightarrow{\text{Up}} - \overrightarrow{\text{Look}}(\overrightarrow{\text{Look}} \cdot \overrightarrow{\text{Up}})}{\|\overrightarrow{\text{Up}} - \overrightarrow{\text{Look}}(\overrightarrow{\text{Look}} \cdot \overrightarrow{\text{Up}})\|} \tag{9}$$

Bestimmen von u

- Bestimmung von u jetzt einfach
- u muß senkrecht zu v und n stehen, da alle drei zusammen die Basis des Koordinatensystems bilden
- für rechtshändige Koordinatensysteme: Kreuzprodukt zwischen v und n

$$u = \frac{v \times n}{\|v \times n\|} \tag{10}$$

Bestimmen von u, v und n

Zusammenfassung:

$$\begin{aligned} n &= \frac{-\overrightarrow{\text{Look}}}{\|Look\|} \\ v &= \frac{\overrightarrow{\text{Up}} - \overrightarrow{\text{Look}}(\overrightarrow{\text{Look}} \cdot \overrightarrow{\text{Up}})}{\|\overrightarrow{\text{Up}} - \overrightarrow{\text{Look}}(\overrightarrow{\text{Look}} \cdot \overrightarrow{\text{Up}})\|} \\ u &= \frac{v \times n}{\|v \times n\|} \end{aligned}$$

Damit sind die Komponenten der Viewing-Transformation vollständig bekannt.
Dazu inverse Transformation: Normalisierungstransformation, bildet den gegebenen Sichtkörper auf den kanonischen ab.

Schrittweises Vorgehen

Jeder Schritt wird durch eine Transformationsmatrix repräsentiert → Komposition dieser Matrizen beschreibt die gesamte Transformation

- Parallelprojektion:
  - Verschiebe Kamera in den Ursprung
  - Transformiere Ansicht, daß (u, v, n) mit (x, y, z) übereinstimmt
  - Skalierung, daß Sichtkörper zwischen -1 und 1 in x- und y-Richtung, far clipping plane bei $z = -1$ und near clipping plane bei $z = 0$ liegen
- perspektivische Projektion
  - gleiche Schritte wie oben, zusätzlich
  - Verzerre Pyramidenstumpf zum Quader, um perspektivische Verzerrung zu erreichen

Schritt 1

Transformation der Kamera in den Ursprung:

- gesucht: Matrix, die $(x_{\mathrm{COP}}, y_{\mathrm{COP}}, z_{\mathrm{COP}})$ nach $(0, 0, 0)$ verschiebt
- Lösung: $(t_x, t_y, t_z) = (-x_{\mathrm{COP}}, -y_{\mathrm{COP}}, -z_{\mathrm{COP}})$

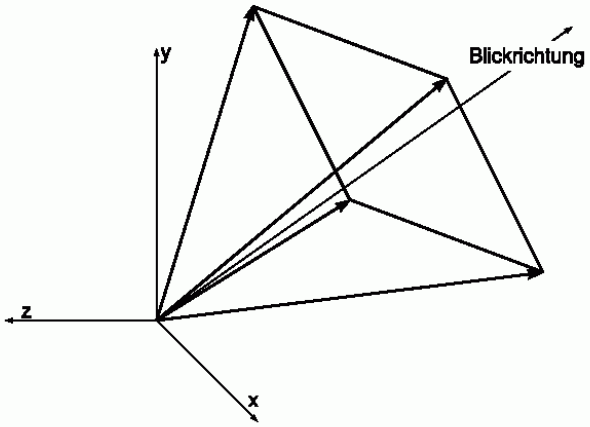
$$T(-\mathrm{COP}) = \begin{pmatrix} 1 & 0 & 0 & -x_{\mathrm{COP}} \\ 0 & 1 & 0 & -y_{\mathrm{COP}} \\ 0 & 0 & 1 & -z_{\mathrm{COP}} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Multiplikation aller Szenenkoordinaten p mit $T(-\mathrm{COP})$

$$p' = T(-\mathrm{COP})p$$

Momentane Situation:

COP liegt jetzt im Ursprung



Schritt 2

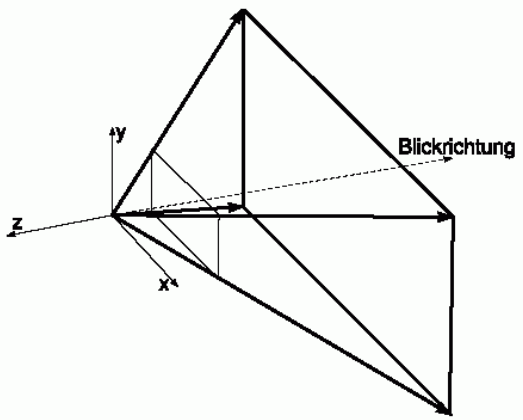
Rotation und Angleichung an das Weltkoordinatensystem:

- vorher besprochen: view transformation matrix mit den Spalten u, v und n rotiert die x, y und z-Achsen in die u, v und n-Achsen
- Anwendung der inversen (transponierten) Matrix auf die Szene, d. h. eine Matrix mit den *Zeilen* u, v und n rotiert die Achsen u, v und n in die Achsen x, y und z
- Multiplikation aller Szenenkoordinaten p mit der zusammengesetzten Transformationsmatrix $M^T T(-\mathrm{COP})$:

$$p' = M^T T(-\mathrm{COP})p$$

Momentane Situation:

COP liegt im Ursprung, (u, v, n) an (x, y, z) ausgerichtet

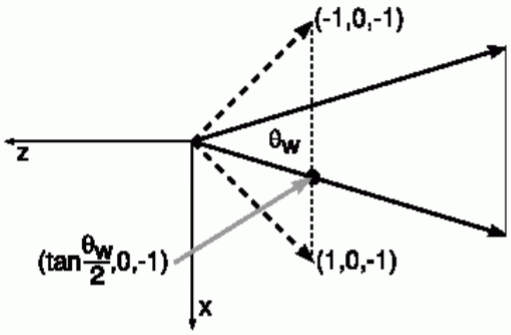


Schritt 3

Skalierung der Clipping planes

- Proportionen des Sichtkörpers müssen normalisiert werden: Skalierung
- Eckpunkte der far clipping plane $\rightarrow (\pm 1, \pm 1, -1)$
- Vektoren vom Ursprung zu den Eckpunkten der far clipping plane müssen einen Winkel von 45° mit der x- und y-Achse einschließen
- Erreicht durch Skalierung in x und y

- Ansicht von oben (entlang der negativen y-Achse):



- Skalierung gesucht, so daß $\theta_w = 90^\circ$
- Skalierungsfaktor: $1/\left(\tan\frac{\theta_w}{2}\right) = \cot\frac{\theta_w}{2}$
- genauso für y (dann mit Winkel θ_h)

- Transformationsmatrix:

$$S_{xy} = \begin{pmatrix} \cot\frac{\theta_w}{2} & 0 & 0 & 0 \\ 0 & \cot\frac{\theta_h}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Multiplikation aller Szenenkoordinaten p mit der zusammengesetzten Transformationsmatrix $S_{xy}M^T T(-\mathrm{COP})$:

$$p' = S_{xy}M^T T(-\mathrm{COP})p$$

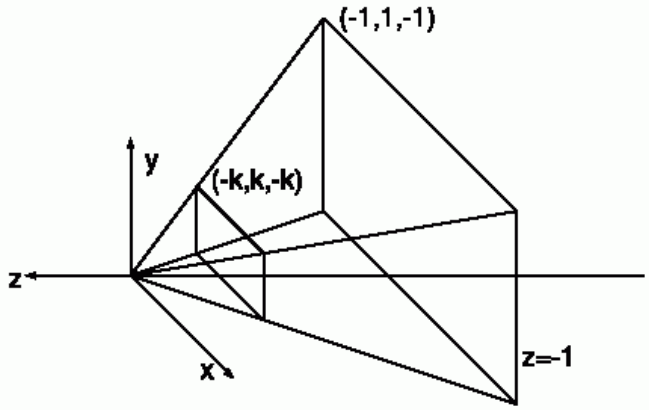
Die relativen Proportionen des Sichtkörpers sind jetzt korrekt.
Aber: *far clipping plane* noch nicht auf *z* = –1 normiert.
Abstand vom Projektionszentrum (COP) zur *far clipping plane* ist immer noch **`far`**.

- uniforme Skalierung mit folgender Matrix:

$$S_{xyz}\left(\frac{1}{\mathrm{far}}\right) = \begin{pmatrix} \frac{1}{\mathrm{far}} & 0 & 0 & 0 \\ 0 & \frac{1}{\mathrm{far}} & 0 & 0 \\ 0 & 0 & \frac{1}{\mathrm{far}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Momentane Situation:

- far clipping plane bei $z = -1$



- near clpping plane bei $z = -k$ mit $k > 0$

Ergebnis

- Normalisierungstransformation für den kanonischen perspektivischen Sichtkörper ist die zusammengesetzte Matrix:

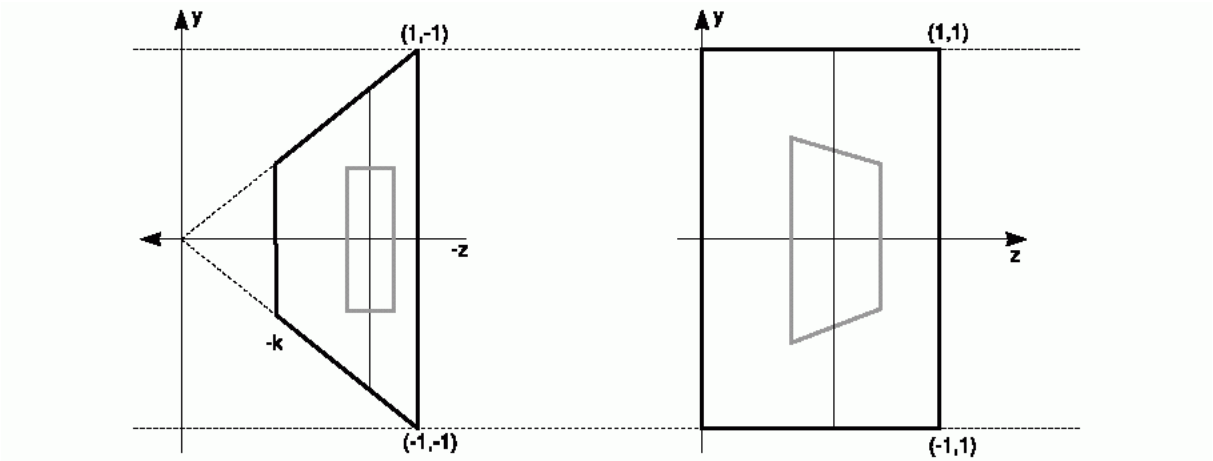
$$S_{xyz}\left(\frac{1}{\text{far}}\right) S_{xy} M^T T(-\text{COP})$$

- exakt gleiche Vorgehensweise zur Transformation des parallelen Sichtkörpers in den kanonischen

- Jetzt: Transformation der Punkte im Standard-Sichtkörper für perspektivische Projektionen zwischen $-k$ und -1 in den Standard-Sichtkörper für Parallelprojektionen
- für den z-Buffer-Algorithmus noch Transformation der Szene nach $0 \leq z \leq 1$
- Transformationsmatrix $(0 < k < 1)$:

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & \frac{1}{k-1} & \frac{k}{k-1} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

darin ist *k* = near / far.



Die perspektivische Transformation erzeugt die perspektivische Verzerrung *bevor* der eigentlichen (Parallel-)Projektion. Der Unterschied zwischen der perspektivischen *Transformation* und einer perspektivischen *Projektion* besteht darin, daß erstere auch die z-Koordinate transformiert und *keine* Projektion durchführt. D. h. die z-Ordnung belibt erhalten – wichtig für HSR.

(HSR = hidden surface removal, Entfernen verdeckter Kanten – siehe später in der Vorlesung)

Endergebnis

- endgültige Transformation:

$$p' = DS_{xyz}\left(\frac{1}{\text{far}}\right) S_{xy} M^T T(-\text{COP}) p$$

- Die Matrizen $D, S_{xyz}\left(\frac{1}{\text{far}}\right), S_{xy}, M^T, T(-\text{COP})$ können alle berechnet werden, wenn die Kameraparameter bekannt sind.

- Erzeugen einer zusammengesetzten Transformationsmatrix, mit der alle Szenenkoordinaten multipliziert werden, um diese von Weltkoordinaten in den Standard-Sichtkörper für Parallelprojektionen zu überführen.

Clipping

- Transformierte Szene paßt in einen Quader nahe des Koordinatenursprungs
- noch notwendig: Clipping der Szene gegen die Seiten des Sichtkörpers
- normalisierter Sichtkörper mit folgender Ausdehnung:

$$\begin{aligned} -1 \leq\ & x\ \leq 1 \\ -1 \leq\ & y\ \leq 1 \\ 0 \leq\ & z\ \leq 1 \end{aligned}$$

- Clipping ist jetzt sehr einfach!!! x- und y-Komponenten gegen ± 1 testen, z-Komponenten gegen 0 und 1

Projektion

- Projektion in die xy-Ebene durch „Weglassen“ der z-Koordinate
- dabei Mapping auf (Pixel-)Koordinaten des Bildschirms
- Seien (x, y, z) Koordinaten nach Normalisierungstransformation und Clipping, dann ergeben sich die Screen-Koordinaten (x', y') aus:

$$\begin{aligned} x' &= \left\lfloor \frac{W}{2}(x+1) \right\rfloor \\ y' &= \left\lfloor \frac{H}{2}(y+1) \right\rfloor \end{aligned}$$

  mit W als Bildschirmbreite und H als Bildschirmhöhe

Zusammenfassung

- Reduktion des gesamten Problems auf Multiplikation der Koordinaten mit einer zusammengesetzten Matrix, die aus der Spezifikation der Kameraparameter gewonnen werden kann
- Zusätzlich dazu noch die Modelltransformationen
- $\rightarrow$ Transformation lokaler Koordinaten des Modells p in Bildschirmkoordinaten p' ist Matrixmultiplikation:

$$p' = NMp$$

  mit der Normalisierungstransformation N und der zusammengesetzten Modell-Trannsformationsmatrix M