Computergrafik

– für Bachelor-Studierende –

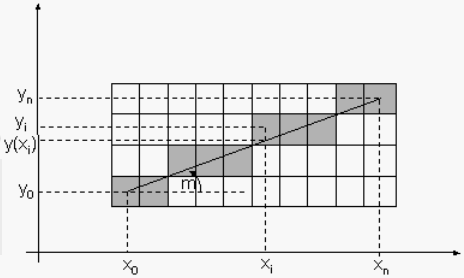
Prof. Dr. Aris Christidis

SS 2011

– Auszug –

Gerasterte Linien

(n+1) Punkte einer gerasterten Geraden zw. (x_0,y_0) u. (x_n,y_n)
– hier verwendete Notation:



x_i: Abszisse des i-ten Rasterpunktes ($x_i \in \mathbf{N}_0$)
$x_{i+1} = x_i + 1 \Rightarrow x_{i+j} = x_i + j$

$y(x_i)$: zu x_i, gehörige, mathem. exakte Ordinate ($y(x_i) \in \mathbf{R}$)

y_i: zu x_i gewählte („optimale“) Raster-Ordinate ($y_i \in \mathbf{N}_0$)

Annahme: $y(x_0) = y_0$; $y(x_n) = y_n$

$\Delta x = x_n - x_0$; $\Delta y = y(x_n) - y(x_0) = y_n - y_0$

Geradensteigung: $m = \Delta y / \Delta x = [\, y(x_i) - y_0 \,] / [\, x_i - x_0 \,]$ i=1,..,n

Berechnung der Ordinaten ($\in \mathbf{R}$) aus Pixelgrößen ($\in \mathbf{N}_0$):

$$y(x_i) = m \cdot (x_i - x_0) + y_0$$

Folgepixel: $y(x_{i+1}) = y(x_i+1) = m \cdot (x_i + 1 - x_0) + y_0$

Gerasterte Linien

$$y(x_{i+1}) - y(x_i) = m\,(x_i + 1 - x_0 - (x_i - x_0)) + y_0 - y_0$$

$$= m$$

⇨ $$y(x_{i+1}) = y(x_i) + m$$

$$= y_0 + (i+1) \cdot m \qquad \text{(DDA)}$$

Digital Differential Analyzer

```
void BruteForceLine (int x0,int y0,int xn,int yn)

{ int   x=x0;
  float y=(float)y0, m=(float)(yn-y0)/(xn-x0);

  for (x=x0; x<=xn; x++,y+=m)
     setPixel (x, (int)y);

  return;
}
```

t nsec / floatOp * *f* floatOps / Umrißpixel
* *p* Umrißpixel / Dreieck * *d* Dreiecke / Szene

ergeben, falls: *t*=10; *f*=1; *p*=100; *d*=100.000

eine (unnötige) Wartezeit von 0,1 sec / Szene **!**

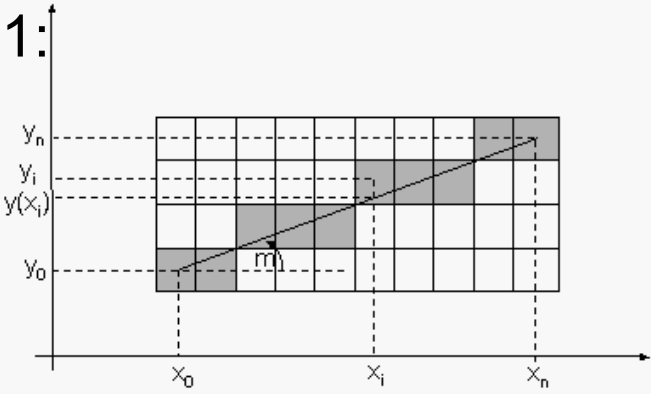
Wunsch: Von Pixel-Koordinaten (gerundet, $\in \mathbf{N}_0$) mit Ganzzahl-Arithmetik auf Folgepixel schließen

Gerasterte Linien

Hinzunahme von Folgepixel (i+1), i=1,...,n-1:

(DDA) ⇨ $y(x_{i+1}) = y(x_0+i+1)$

$= y_0 + (i+1) \cdot \Delta y / \Delta x$



Weiter nach Osten ($y_{i+1} = y_i$),

wenn: $y_0 + (i+1) \cdot \Delta y / \Delta x < y_i + ½$

– sonst weiter nach Nordosten ($y_{i+1} = y_i+1$)

[y_i ist (ganzzahlig) nicht berechenbar!]

Ost, wenn: $(i+1) \cdot \Delta y / \Delta x < y_i - y_0 + ½$

N := Anzahl bisheriger Nordost-Entscheidungen

$= y_i - y_0$

E := Anzahl bisheriger Ost-Entscheidungen

[N, E: nicht berechenbar – aber zählbar !]

$N + E = i$

Gerasterte Linien

(Wdhlg.:) Ost, d.h. $y_{i+1} = y_i$, wenn

$$(i+1) \cdot \Delta y / \Delta x < y_i - y_0 + ½$$

$$(N+E+1) \cdot \Delta y / \Delta x < N + ½$$

$$2 \cdot N \cdot \Delta y + 2 \cdot E \cdot \Delta y + 2 \cdot \Delta y < 2 \cdot N \cdot \Delta x + \Delta x$$

$$\underbrace{N \cdot 2 \cdot (\Delta y - \Delta x) + E \cdot 2 \cdot \Delta y + 2 \cdot \Delta y - \Delta x}_{d} < 0$$

Zur Erinnerung: Bei der ersten Entscheidung $\Leftrightarrow$ i = N = E = 0
„Ost-Bedingung“: $2\Delta y - \Delta x < 0$

Bresenham‘s Strategie: Einrichtung einer „decision variable“ **d**, die mit $2 \cdot \Delta y - \Delta x$ initialisiert und nach jeder Folge-Entscheidung um $2\Delta y$ (Ost) bzw. $2 \cdot (\Delta y - \Delta x)$ (Nordost) erhöht wird.

Gerasterte Linien

Besonderheiten / Vorzüge des Bresenham-Algorithmus („Bresenham‘s algorithm for scan-converting straight lines“):

- Nutzung der Tatsache, daß Pixel diskret sind:
  $x_{i+1} = x_i + 1 \Rightarrow x_{i+j} = x_i + j \Rightarrow y(x_j) = y_0 + j \cdot m$
- Verknüpfung algorithmischer Vorgänge (Entscheidungen) mit Variablen-Werten:
  $N = y_i - y_0 \; ; \; N + E = i$
- Entscheidung über Hinzunahme von Folgepixel anhand des zuletzt gesetzten Pixels (ohne Geraden-Steigung m)
- Ganzzahlige Addition und Subtraktion
- Je Linie nur 3 ganzzahlige Multiplikationen mit 2 (Hw!)
- Erweiterbar auf Kreise und Ellipsen (in obiger Form)
- Repetierbar u. (grundsätzlich) in der Richtung umkehrbar (Löschung mit Hintergrund-Farbe)
- Läßt in obiger Version den letzten Punkt aus (Polygone: letzter Punkt = erster ⇨ Flimmern, Löcher im Papier)

2D-Punkt-Transformationen

Zur Erinnerung – Drehung eines beliebigen Punktes B‘ um den Winkel θ um den Koordinaten-Ursprung zum Punkt B“:

$x_{B‘} = r \cdot \cos\alpha$

$y_{B‘} = r \cdot \sin\alpha$ [r, α: Hilfsgrößen]

$x_{B“} = r \cdot \cos(\alpha+\theta)$

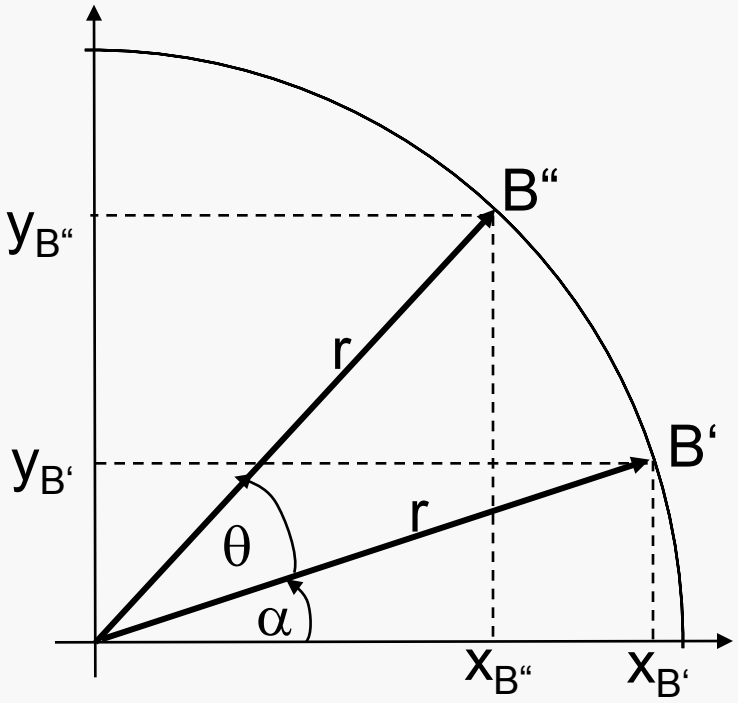
$= r \cdot (\cos\alpha \cos\theta - \sin\alpha \sin\theta)$

$= x_{B‘} \cdot \cos\theta - y_{B‘} \cdot \sin\theta$

$y_{B“} = r \cdot \sin(\alpha+\theta)$

$= r \cdot (\sin\alpha \cos\theta + \cos\alpha \sin\theta)$

$= x_{B‘} \cdot \sin\theta + y_{B‘} \cdot \cos\theta$



In Matrizen-Schreibweise:

$$\begin{pmatrix} x_{B“} \\ y_{B“} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x_{B‘} \\ y_{B‘} \end{pmatrix}$$

$\sin(\alpha+\beta) = \sin\alpha \cos\beta + \cos\alpha \sin\beta$

$\cos(\alpha+\beta) = \cos\alpha \cos\beta - \sin\alpha \sin\beta$

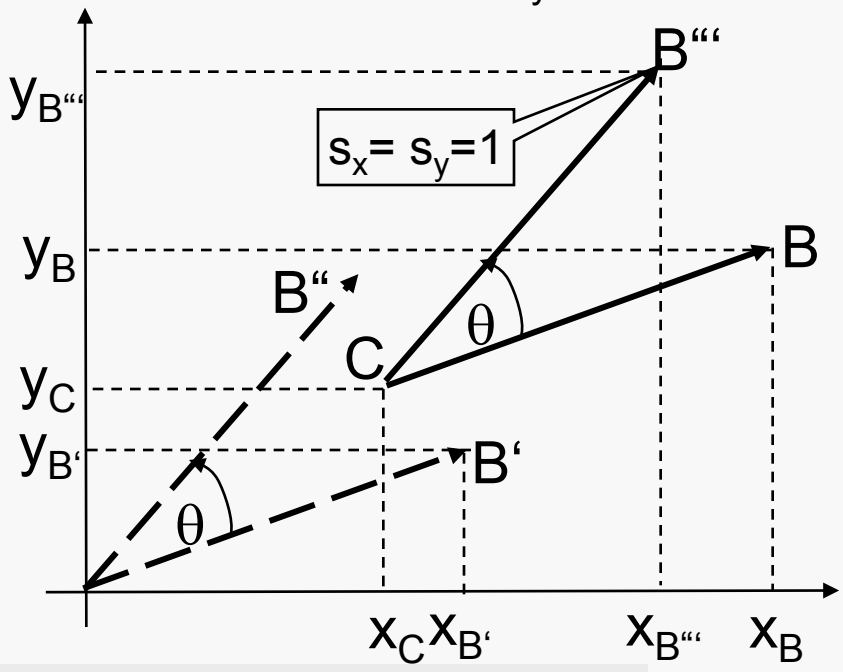
2D-Punkt-Transformationen

Verallgemeinerung: Rotation eines beliebigen Punktes B um den Winkel θ um einen beliebigen Punkt C zum Punkt B‘‘‘. Skalierung des Ergebnisses mit den Faktoren s_x und s_y.

$$x_{B'} = x_B - x_C$$
$$y_{B'} = y_B - y_C$$

$$x_{B''} = x_{B'} \cdot \cos\theta - y_{B'} \cdot \sin\theta$$
$$y_{B''} = x_{B'} \cdot \sin\theta + y_{B'} \cdot \cos\theta$$

$$x_{B'''} = s_x \cdot (x_{B''} + x_C)$$
$$y_{B'''} = s_y \cdot (y_{B''} + y_C)$$



In Matrizen-Schreibweise:

$$\begin{pmatrix} x_{B'''} \\ y_{B'''} \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \left\{ \begin{pmatrix} x_C \\ y_C \end{pmatrix} + \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \cdot \left[\begin{pmatrix} x_B \\ y_B \end{pmatrix} - \begin{pmatrix} x_C \\ y_C \end{pmatrix} \right] \right\}$$

2D-Punkt-Transformationen

Erweiterung um eine Dimension ermöglicht die Darstellung der Translation (Verschiebung) als Matrizen-Produkt:

$$x_{B'} = x_B - x_C$$
$$y_{B'} = y_B - y_C$$

„homogene Koordinaten“,
„homogene Punkt-Darstellung“

$$\begin{pmatrix} x_{B'} \\ y_{B'} \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -x_C \\ 0 & 1 & -y_C \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}$$

Rotation eines Punktes B um θ um bel. Punkt C nach B‘‘‘ und Skalierung mit s_x , s_y in homogenen Koordinaten:

$$\begin{pmatrix} x_{B'''} \\ y_{B'''} \\ 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & x_C \\ 0 & 1 & y_C \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & -x_C \\ 0 & 1 & -y_C \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix}$$

„Grafik-Pipeline“: Verarbeitungsschritte von den Objekt-Daten bis zur Bild-Erzeugung

2D-Punkt-Transformationen

- Zur Erinnerung – einige Rechenregeln d. Matrizenrechnung*:

  $\underline{A} + \underline{B} = \underline{B} + \underline{A}$ [Kommutativgesetz d. Matrizenaddition]

  $\underline{A} \bullet (\underline{B} + \underline{C}) = \underline{A} \bullet \underline{B} + \underline{A} \bullet \underline{C}$ [Distributivgesetz]

  $(\underline{A} \bullet \underline{B}) \bullet \underline{C} = \underline{A} \bullet (\underline{B} \bullet \underline{C})$ [Assoziativgesetz]

  $\underline{A} \bullet \underline{B} \neq \underline{B} \bullet \underline{A}$ [Matrizenprodukt ist nicht kommutativ!]

  $\underline{A} \bullet \underline{B} = \underline{0} \iff \underline{B} = \underline{0} \vee \underline{A} = \underline{0}$!

  $\underline{A} \bullet \mathbf{\underline{I}} = \mathbf{\underline{I}} \bullet \underline{A} = \underline{A}$ [$\mathbf{\underline{I}}$:„neutrales Element“ d.M.-Multiplikation]

  $\underline{A} \bullet \underline{A}^{-1} = \underline{A}^{-1} \bullet \underline{A} = \mathbf{\underline{I}}$ (falls $\underline{A}$ invertierbar!)

  $\underline{A} \bullet \underline{x} = \underline{b} \Rightarrow \underline{A}^{-1} \bullet \underline{b} = \underline{A}^{-1} \bullet \underline{A} \bullet \underline{x} = \underline{x}$ (falls $\underline{A}$ invertierbar!)

  $(\underline{A} \bullet \underline{B})^{-1} = \underline{B}^{-1} \bullet \underline{A}^{-1}$ (falls $\underline{A}$, $\underline{B}$ invertierbar!)

  $(\underline{A} \bullet \underline{B})^{T} = \underline{B}^{T} \bullet \underline{A}^{T}$

  Wichtiger Spezialfall: Rotationsmatrizen sind orthogonal, d.h.

  $\underline{O}^{-1} = \underline{O}^{T}$

(*) unter d. Voraussetzung geeigneter Dimensionierung beteiligter Matrizen

2D-Punkt-Transformationen

Verkettung von Punkt-Transformationen und ihren Inversen:

- Translation (Verschiebung):

$$\begin{pmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & x_c - x_c \\ 0 & 1 & y_c - y_c \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Skalierung:

$$\begin{pmatrix} 1/s_x & 0 & 0 \\ 0 & 1/s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Spiegelung ist ein Spezialfall der Skalierung!

- Rotation (Drehung):

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

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

2D-Punkt-Transformationen

Beobachtungen an der Rotationsmatrix:
(vor allem an der o./li. 2x2-Untermatrix)

$$\underline{\mathbf{R}} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

- Jede Zeile u. jede Spalte ist ein Einheitsvektor (Länge=1)
- Sowohl die Zeilen- als auch die Spaltenvektoren stehen jeweils senkrecht aufeinander (Skalarprodukte =0)

➡ $\underline{\mathbf{R}}$ ist orthogonal: $\underline{\mathbf{R}} \cdot \underline{\mathbf{R}}^T = \underline{\mathbf{R}}^T \cdot \underline{\mathbf{R}} = \underline{\mathbf{I}}$

(^{T}:Transposition; $\underline{\mathbf{I}}$: Einheitsmatrix)

- Werden die Einheitsvektoren der Hauptachsen (x,y) mit $\underline{\mathbf{R}}$ transformiert, so ergeben sie die Spaltenvektoren von $\underline{\mathbf{R}}$.
- Werden die Zeilenvektoren von $\underline{\mathbf{R}}$ mit $\underline{\mathbf{R}}$ transformiert, so ergeben sie die Einheitsvektoren der Hauptachsen (x,y).

➡ Möglichkeit, aus dem Rotationsergebnis auf die Rotationsmatrix zu schließen!

2D-Punkt-Transformationen

Weitere 2D-Transformation: **Scherung** (engl. *shear*)

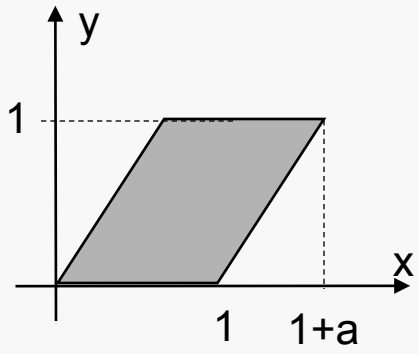
Scherung entlang einer Achse verändert dazugehörige Punkt-Koordinaten proportional zur jeweils anderen Koordinate:

- Scherung eines Einheitsquadrats entlang der x-Achse verändert jeden Punkt $[x, y, 1]^T$ zu $[x+ay, y, 1]^T$ mit:

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

Inverse:

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

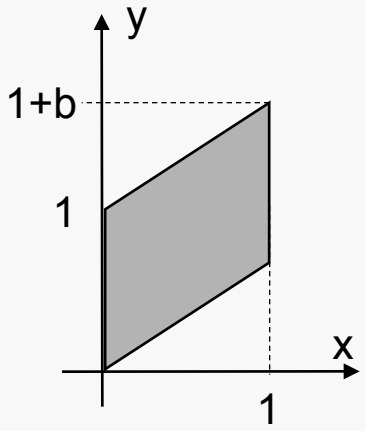


- Scherung des Einheitsquadrats entlang der y-Achse verändert jeden Punkt $[x, y, 1]^T$ zu $[x, bx+y, 1]^T$ mit:

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

Inverse:

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



x-y-Kombination als Verkettung!

Window-Viewport-Transformation

Alte Begriffe der Computergrafik:

- (World-Coordinate-) **Window** heißt ein rechteckiger (Szene-, Welt-) Ausschnitt, der abgebildet werden soll; darf nicht mit dem (Window-Manager-) Window des Sw-Fenstersystems verwechselt werden.
- **Viewport** nennt man den rechteckigen Ausschnitt der Geräte-Ausgabefläche (Bildschirm, Plotter), in dem die Ausgabe erfolgen soll.

Im allgemeinen Fall besteht die Window-Viewport-Transform. aus

- einer Translation des Window an den Ursprung des Welt-Koordinatensystems,
- einer Skalierung auf die Größe des Viewport und
- einer Translation an die Lage des Viewport.

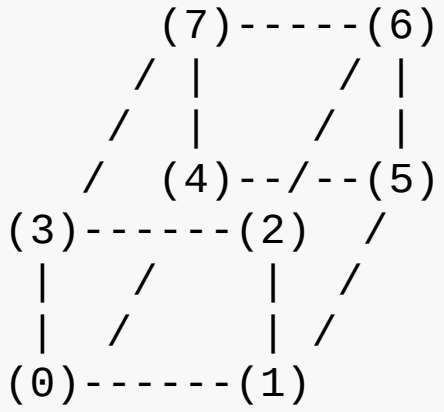
Codierung von Grafik-Modellen

Speicherung eines Grafik-Objekts (z.B. eines Würfels):

```
cube.cgf ; Objekt-Name - CGF Version 0.0
8 ; Anzahl Punkte
-1., -1.,  1. ; Koord. Pkt[0] etc.
 1., -1.,  1.
 1.,  1.,  1.
-1.,  1.,  1.
-1., -1., -1.
 1., -1., -1.
 1.,  1., -1.
-1.,  1., -1.
6 ; Anzahl Flaechen
4, 4, 4, 4, 4, 4 ; Pkte je Flaeche
0, 3, 7, 4 ; Pkt-Id ab Flaeche[0]
1, 0, 4, 5
2, 3, 0, 1
3, 2, 6, 7
4, 7, 6, 5
5, 6, 2, 1
```

Geometrie der Objektpunkte (ihre räumliche Lage)

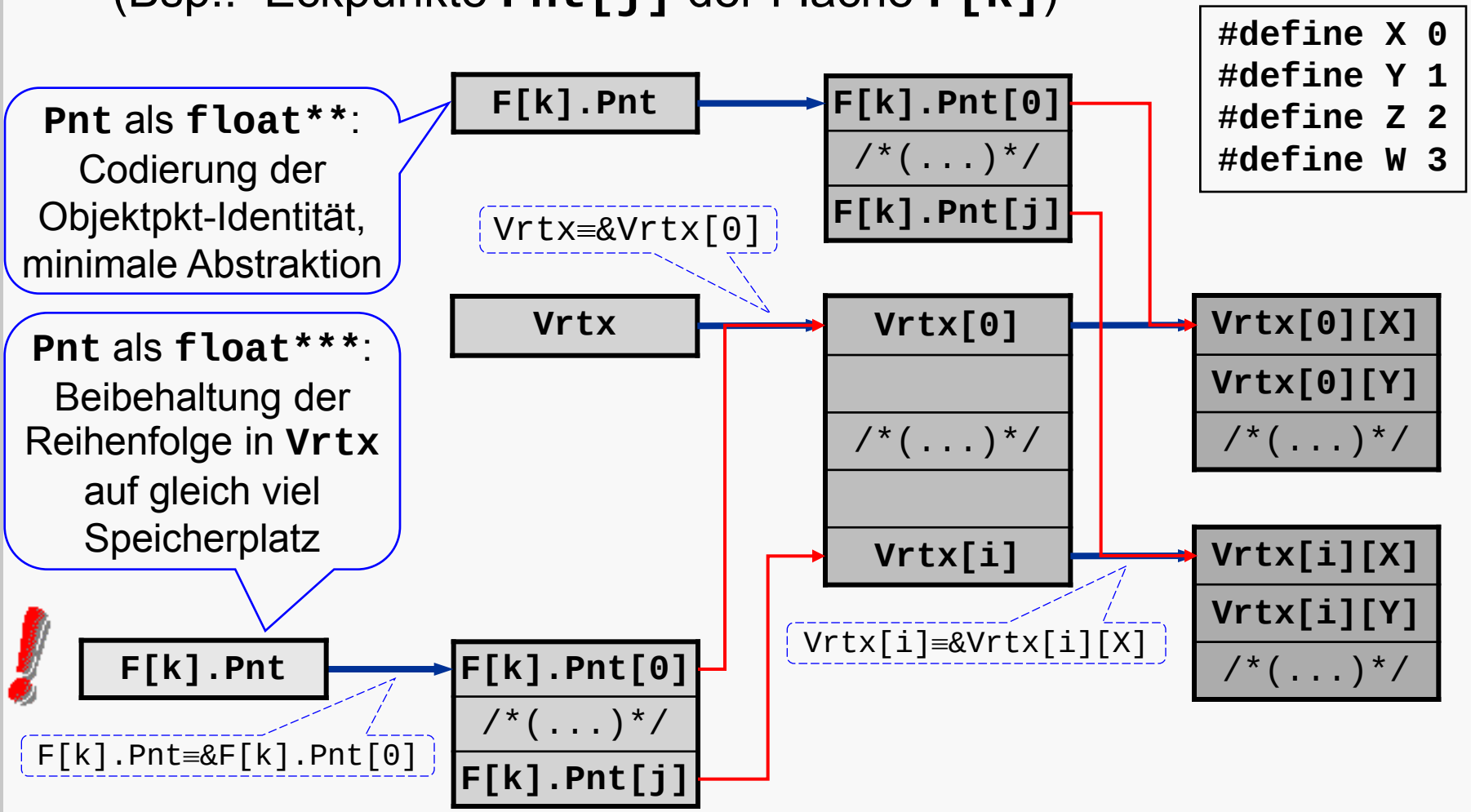
Topologie der Objektpunkte (Beziehungen zwischen ihnen)



(Xface?)

Codierung von Grafik-Modellen

Codierung d. Objekt-/Eckpkt-Zuordnung prinzipiell wählbar:
(Bsp.: Eckpunkte **`Pnt[j]`** der Fläche **`F[k]`**)



Codierung von Grafik-Modellen

Allgemeine programm-interne Darstellg. eines 3D-Objektes:

```
typedef struct
{ int      nPnt;  //Anzahl Eckpunkte
  float ***Pnt;   //Punkt-Indizes
  char     symbol;//Zeichn f.Konsole
} Polygon;
```

```
typedef char  String[LENGTH];
```

stellv. für Flächen-Eigensch.: Farbe, Textur, Rauhigkeit,...

```
#define DIM      4
#define LENGTH  80
```

Als Zeiger: Größen, die zur Compi-lierungszeit nicht bekannt sein können

```
typedef struct
{ String   Name;              //Obj.-Name, Header
  int      nVrtx;             //Anzahl Objektpkte
  int      nFace;             //Anzahl Flaechen
  float  **Vrtx;              //Objektpunktkoord.
  Polygon *Face;              //Flaechenliste
  float    posMat[DIM*DIM];//Positionsmatrix
} CGFobject;
```

Codierung von Grafik-Modellen

Typische Vorgänge beim Laden eines grafischen Objektes:

- Bereitstellung von (Struktur-)Variablen für bekannte Merkmale
- Belegung der (Struktur-)Variablen mit Werten aus Datei
- Sukzessive Speicherplatzreservierg. für restliche Merkmale – z.B. (in Funktion mit CGF-Objekt **`obj`**):

```
obj->Vrtx=(float**)calloc(obj->nVrtx, sizeof(float *));
```

- Berechnung (i.d.R. auch Speicherung) wiederholt benötigter, abhängiger Merkmale – z.B. (vgl. Übungen)
  - **Bounding Box** (umhüllender Quader, zur schnellen Überprüfg. von Sichtbarkeit im Sichtvolumen, Verdeckung, Kollision etc.)
  - **Flächennormalen** (zur Ermittlung der Lage gegenüber Lichtquellen oder dem Augenpunkt, z.B. zur Eliminierung abgewandter Objektflächen – engl. *back face culling*)

3D-Punkte, 3D-Vektoren

In den Naturwissenschaften häufig: Rechnen mit 3D-Vektoren und -Punkten –z.B.: Stärke, Richtung d. Gravitation g(x,y,z) an einem bestimmten Punkt im Weltraum p(x,y,z)

Darstellung von 3D-Vektoren und -Punkten meist identisch: Zahlentripel (x, y, z) – aber:

- Vektoren haben Betrag und Richtung, aber keine Position.
- Punkte haben Position, aber weder Betrag noch Richtung.

Typische Umgehung der Unwegsamkeit: Darstellung eines Punktes über seinen Ortsvektor, d.h. über seinen *Versatz* gegenüber d.Koordinaten-Ursprung (*V*.-Betrag, *V*.-Richtung!)

Verbleibendes Problem: Darstellung bei Verwendung mehrerer Koordinatensysteme.

⇨ Erweiterung des Begriffs des (3D-)Koordinatensystems:

3D-Punkte, 3D-Vektoren

Ein *Coordinate Frame* (*CF*, Koord.rahmen,-netz) besteht aus

- 3 senkrecht zueinander stehenden Einheitsvektoren $\underline{\mathbf{i}},\underline{\mathbf{j}},\underline{\mathbf{k}}$
- einem besonderen Punkt φ, dem Ursprung (engl. *origin*).

($\underline{\mathbf{i}},\underline{\mathbf{j}},\underline{\mathbf{k}}$,φ können nur über andere CF spezifiziert werden!)

„Homogene“ Vektor- u. Punkt-Darstellung (4 Komponenten):

Vektor $\underline{v} = v_1\,\underline{\mathbf{i}} + v_2\,\underline{\mathbf{j}} + v_3\,\underline{\mathbf{k}} \qquad = [\,v_1, v_2, v_3, 0\,]^T$

Punkt $\underline{p} = p_1\,\underline{\mathbf{i}} + p_2\,\underline{\mathbf{j}} + p_3\,\underline{\mathbf{k}} + \varphi \quad = [\,p_1, p_2, p_3, 1\,]^T$

Übereinstimmend mit bisherigen Feststellungen (vgl. Ortsv.):

- Die Differenz zweier Punkte ist ein Vektor.
- Die Summe eines Punktes und eines Vektors ist ein Punkt.
- Die Summe zweier Vektoren ist ein Vektor.
- Skalierung eines Vektors ist sinnvoll.
- Addition von Punkten ist nicht sinnvoll / nicht zulässig.
- Jede Linearkombination von Vektoren ergibt einen Vektor.

3D-Punkte, 3D-Vektoren

Zur Erinnerung:

- **Linearkombination** von Vektoren = Summe skalierter Vektoren:

  $\underline{v} = \alpha_1 \underline{v}_1 + \alpha_2 \underline{v}_2 + \ldots (\alpha_i \in \mathbf{R})$

- **Affine Kombination** von Vektoren = Summe skalierter Vektoren mit Summe der Skalierungsfaktoren =1:

  $\underline{v} = \alpha_1 \underline{v}_1 + \alpha_2 \underline{v}_2 + \ldots (\alpha_i \in \mathbf{R},\ \Sigma\alpha_i = 1)$

- **Konvexe Kombination** von Vektoren = Summe skalierter Vektoren mit Summe der Skalierungsfaktoren =1 und mit nichtnegativen Skalierungsfaktoren :

  $\underline{v} = \alpha_1 \underline{v}_1 + \alpha_2 \underline{v}_2 + \ldots (\alpha_i \in \mathbf{R},\ \Sigma\alpha_i = 1,\ \alpha_i \geq 0)$

Affine u. konvexe Kombinationen von Punkten sind zulässig!

3D-Punkt-Transformationen

3D-Punkt-Transformationen und ihre Inversen:

- Translation:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & -t_x \\ 0 & 1 & 0 & -t_y \\ 0 & 0 & 1 & -t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Skalierung:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1/s_x & 0 & 0 & 0 \\ 0 & 1/s_y & 0 & 0 \\ 0 & 0 & 1/s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Scherung: (entlang x)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & -q & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & q & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

evtl.beteiligte Elemente nach Verkettung mehrerer eindimens. Scherungen

3D-Punkt-Transformationen

Reihenfolge der Transformationen entscheidend – z.B.:

- Skalierung mit anschließender Translation:

$$\begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & t_x \\ 0 & s_y & 0 & t_y \\ 0 & 0 & s_z & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

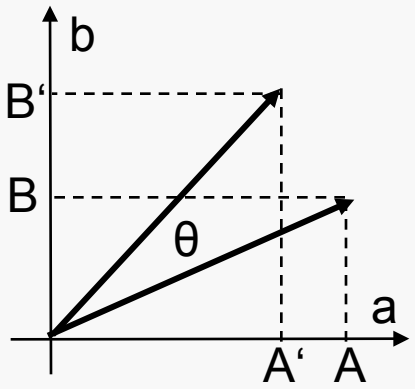
- Translation mit anschließender Skalierung:

$$\begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & 0 & s_x t_x \\ 0 & s_y & 0 & s_y t_y \\ 0 & 0 & s_z & s_z t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

[s.o.: Das Matrizenprodukt ist nicht kommutativ!]

3D-Punkt-Transformationen

- Verwendung eines Rechts-Systems (in der CG wählbar): Rotation positiv gegen d. Uhrzeigersinn bei Blickrichtung vom positiven Teil der Rotatiosachse zum Koordinaten-Ursprung
- Herleitung der Rotationsmatrizen (s.o.):
  $A' = A \cdot \cos\theta - B \cdot \sin\theta$
  $B' = A \cdot \sin\theta + B \cdot \cos\theta$



- Rotation um die x-Achse:
  (a ⇨ y; b ⇨ z)

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta_x & -\sin\theta_x & 0 \\ 0 & \sin\theta_x & \cos\theta_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- Rotation um die y-Achse:
  (a ⇨ z; b ⇨ x)

$$\begin{pmatrix} \cos\theta_y & 0 & \sin\theta_y & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta_y & 0 & \cos\theta_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

„Rechte-Hand-Regel“

3D-Punkt-Transformationen

- Rotation um die z-Achse (vgl. 2D):
  (a ⇨ x bzw. b ⇨ y)

$$\begin{pmatrix} \cos\theta_z & -\sin\theta_z & 0 & 0 \\ \sin\theta_z & \cos\theta_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Beobachtungen an den Rotationsmatrizen:

- Die letzte Spalte ist gleich jener einer Einheitsmatrix (Platz für Translationswerte).
- Zeile und Spalte, die der jeweiligen Drehachse entsprechen, sind identisch mit der Zeile u. der Spalte einer Einheitsmatrix
  (d.h.: keine Veränderung für diese Koordinate).
- Besonders interessanter Spezialfall:
  Rotation um Achse durch den Koordinaten-Ursprung und einen beliebigen Punkt:

3D-Punkt-Transformationen

- Affine Transformationen (ATn) sind Punkt-Transformationen, die Linearkombinationen der kartesischen Koordinaten von Punkten bilden; sie sind die am häufigsten angewandten Transformationen in der CG.
- Translation, Rotation, Skalierung und Scherung sind ATn.
- ATn erhalten affine (und konvexe) Punkt-Kombinationen.
- ATn erhalten gerade Linien und Ebenen.
- ATn erhalten Parallelität von Linien und Ebenen; Translation und Rotation erhalten darüber hinaus auch die Längen von Linien und die Winkel zwischen ihnen.
- ATn erhalten Teilungsverhältnisse von Strecken. (Spezialfall: mittiger Schnittpunkt von Quadrat-/ Würfel-Diagonalen bildet sich ab auf mittigen Schnittpunkt der Parallelogramm-/Parallelepiped-Diagonalen)

3D-Punkt-Transformationen

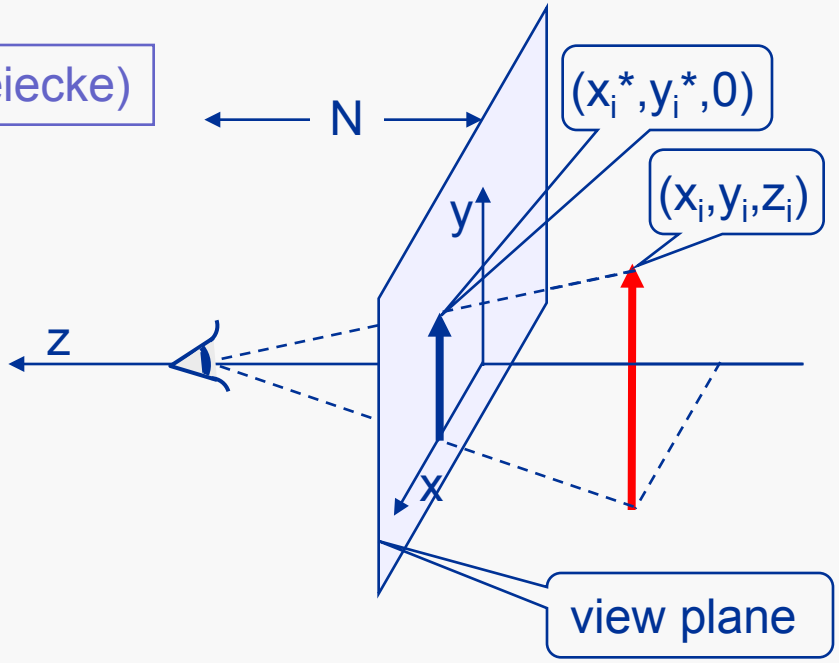
- 2D-ATn multiplizieren den Flächeninhalt transformierter 2D-Objekte mit der Determinanten ihrer Matrix; 3D-ATn verändern entsprechend das Objekt-Volumen mit |det M|
- Multiplikation einer bel. Anzahl der Matrizen mehrerer ATn ergibt die Matrix einer wirkungsgleichen AT.
- Die Transformation, die eine AT rückgängig macht, ist ebenfalls eine AT; ihre Matrix-Darstellung ist die inverse Matrix der ursprünglichen AT.
- Jede AT läßt sich aus einer (bel.) vorgegebenen Folge v. Elementar-ATn zusammensetzen –z.B.: (⇦Leserichtung!)
  M = (Scherung) · (Skalierung) · (Rotation) · (Translation)
- Die Spalten der AT-Matrix enthalten das transformierte CF (≡ transformierte Einheitsmatrix!).
- ATn lassen sich immer in Matrixform darstellen; ihre letzte Zeile hat immer die Form: [0 ... 0 1].

3D-Sicht, Projektionen

Transformationen, deren Matrix als letzte Zeile nicht die Form: [0 ... 0 1] hat, gehören zur allgemeineren Klasse der **perspektivischen Transformationen**.

Perspektivische Projektion von Punkten (x_i,y_i,z_i) auf $(x_i^*,y_i^*,0)$ in der Projektionsebene z=0 mit Proj.zentrum („Augenpunkt“) bei z=N (N>0) in einem Rechts(koordinaten)system:

$x_i^*/x_i = y_i^*/y_i = N/(N-z_i)$ (ähnliche Dreiecke)



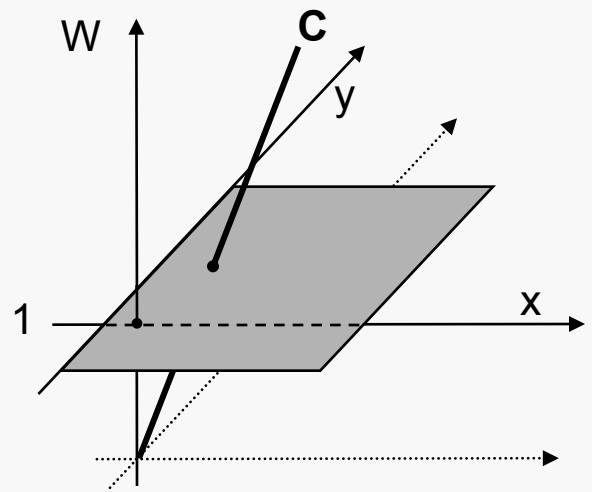
Versuch der Bildung eines Matrizenprodukts:

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & N \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{pmatrix} Nx_i \\ Ny_i \\ 0 \\ N-z_i \end{pmatrix} = \begin{pmatrix} (N-z_i)\cdot x_i^* \\ (N-z_i)\cdot y_i^* \\ (N-z_i)\cdot 0 \\ (N-z_i)\cdot 1 \end{pmatrix}$$

3D-Sicht, Projektionen

Konzept-Erweiterung: Homogene Koordinaten $[x_i,y_i,z_i,1]^T$ als Darstellung v. „Punkt-Familien“ $[w\cdot x_i,w\cdot y_i,w\cdot z_i,w]^T$, $w\neq 0$:

- Wechsel von kartesischen zu homogenen Koordinaten durch Anhängen einer 1 (oder einer anderen Zahl $\neq 0$, mit der zuvor alle Punkt-Koordinaten multipliziert wurden).
- Wechsel von homogenen zu kartesischen Koordinaten durch Division durch die letzte Komponente („perspektivische Division“, auch: „Homogenisieren“, engl. *homogenize*) und Weglassen dieser letzten Komponente.



Geometrische Deutung 2D-Fall: Punkt C ∈ (x, y) wird in homog. Koord. als Gerade C ∈ (x_h, y_h, w_h) mit $x_h=w_h\cdot x$, $y_h=w_h\cdot y$ dargestellt.

3D-Sicht, Projektionen

Begriffliche und rechnerische Trennung:

Die (hier: perspektivische) Projektion reduziert die Anzahl von Objekt-Dimensionen; die perspektivische Transformation verformt Objekte zur Vorbereitung einer Projektion (s.u.).

Proj. Trf.

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&0&1\end{pmatrix} \cdot \begin{pmatrix}N&0&0&0\\0&N&0&0\\0&0&N&0\\0&0&-1&N\end{pmatrix} \cdot \begin{pmatrix}x_i\\y_i\\z_i\\1\end{pmatrix} = \begin{pmatrix}Nx_i\\Ny_i\\0\\N-z_i\end{pmatrix} \xrightarrow{\text{perspekt. Division}} \begin{pmatrix}Nx_i/(N-z_i)\\Ny_i/(N-z_i)\\0\\1\end{pmatrix} = \begin{pmatrix}x_i^*\\y_i^*\\0\\1\end{pmatrix}$$

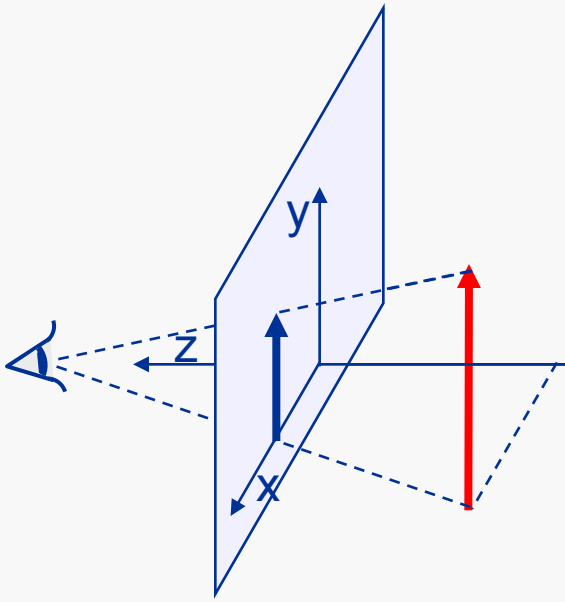
Namensgebung: Koordinaten „homogen“, denn sie ermöglichen auch Perspektive als Matrizen-Multiplikation i.d. Grafik-Pipeline.

$$\begin{pmatrix}1&0&0&0\\0&1&0&0\\0&0&0&0\\0&0&-1/N&1\end{pmatrix} \cdot \begin{pmatrix}x_i\\y_i\\z_i\\1\end{pmatrix} = \begin{pmatrix}x_i\\y_i\\0\\1-z_i/N\end{pmatrix} \xrightarrow{\text{perspektivische Division}} \begin{pmatrix}Nx_i/(N-z_i)\\Ny_i/(N-z_i)\\0\\1\end{pmatrix} = \begin{pmatrix}x_i^*\\y_i^*\\0\\1\end{pmatrix}$$

3D-Sicht, Projektionen

Anmerkungen zur Perspektive:

- Die perspektivische Division bewirkt, daß weiter entfernte Objekte (z_i groß) in der Projektion kleiner erscheinen.
- Verschiebung d. Projektionszentrums (N) verändert die Abbildungs-Unterschiede zwischen nah und fern.
- Verschiebung der Proj.ebene entlang der z-Achse (auf $z \neq 0$) verändert nur den Abb.-Maßstab. (Dreiecke bleiben ähnlich.)
- Auslassen der perspektiv. Transformation erzeugt Parallelprojektion (orthograph. P., engl. *orthographic p.*); das entspricht einer Abb. mit Proj.zentr. im Unendlichen.

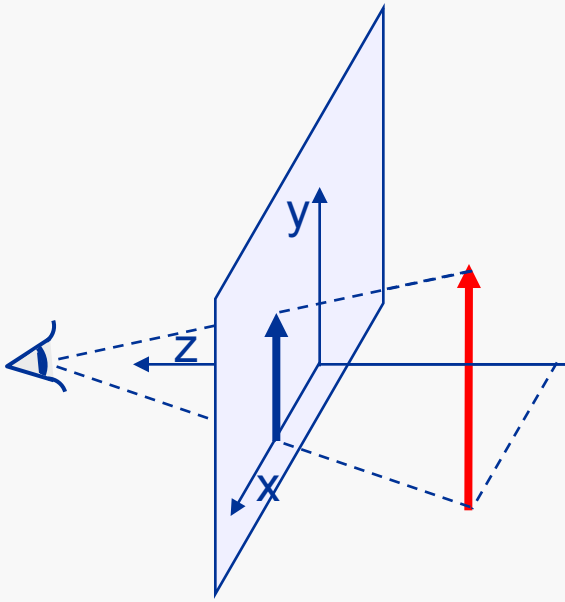


$$\begin{pmatrix} Nx_i/(N-z_i) \\ Ny_i/(N-z_i) \\ 1 \end{pmatrix} = \begin{pmatrix} x_i^* \\ y_i^* \\ 1 \end{pmatrix}$$

3D-Sicht, Projektionen

Weitere Anmerkungen zur Perspektive:

- Gerade Linien und ebene Flächen werden als solche abgebildet: Punkt-Kollinearität und -Komplanarität bleiben erhalten; Teilungsverhältnisse von Strecken und Flächen bleiben dagegen nicht erhalten.
- Parallele Linien, die auch parallel zur Projektionsebene liegen, werden als Parallelen abgebildet; sonst laufen sie in einem Punkt zusammen, dem jeweil. Fluchtpunkt (engl. *vanishing point*).
- Geraden, die das Projektionszentrum enthalten, werden auf Punkte projiziert.
- Ebenen, die d. Proj.zentrum enthalten, werden auf Geraden projiziert.



3D-Sicht, Projektionen

In Praxis und Literatur meist verbreitetes Paradigma:

Perspektivische Projektion v. Punkten (x_i,y_i,z_i) auf $(x_i^*,y_i^*,-N)$ i.d. Projektionsebene $z= -N$ $(N>0)$ mit Projektionszentrum am Koordinaten-Ursprung eines Rechts(koordinaten)systems:

$x_i^*/ x_i = y_i^*/ y_i = N / (-z_i)$

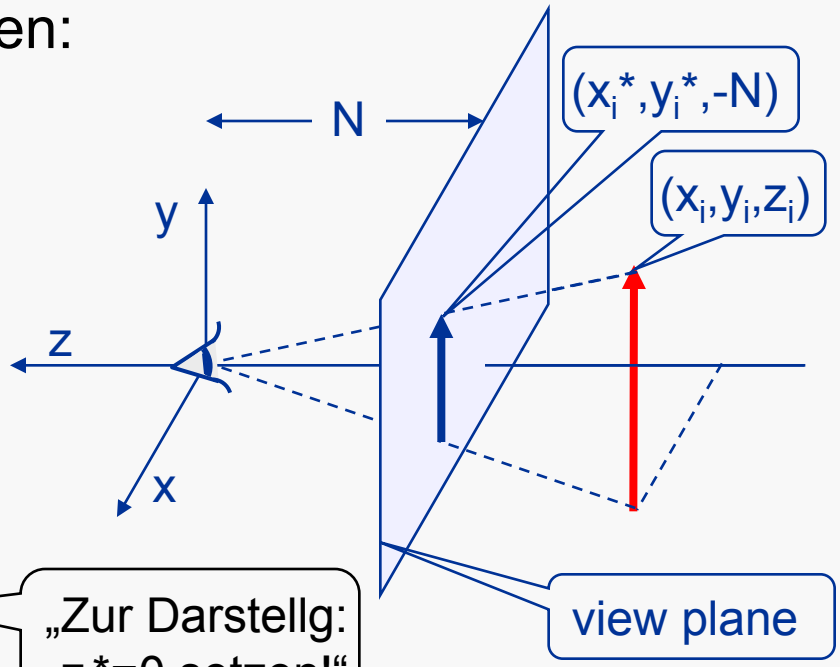
Verwendung homogener Koordinaten:

$$\begin{pmatrix} N & 0 & 0 & 0 \\ 0 & N & 0 & 0 \\ 0 & 0 & N & 0 \\ 0 & 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \\ z_i \\ 1 \end{pmatrix} = \begin{pmatrix} Nx_i \\ Ny_i \\ Nz_i \\ -z_i \end{pmatrix}$$

perspektivische Division

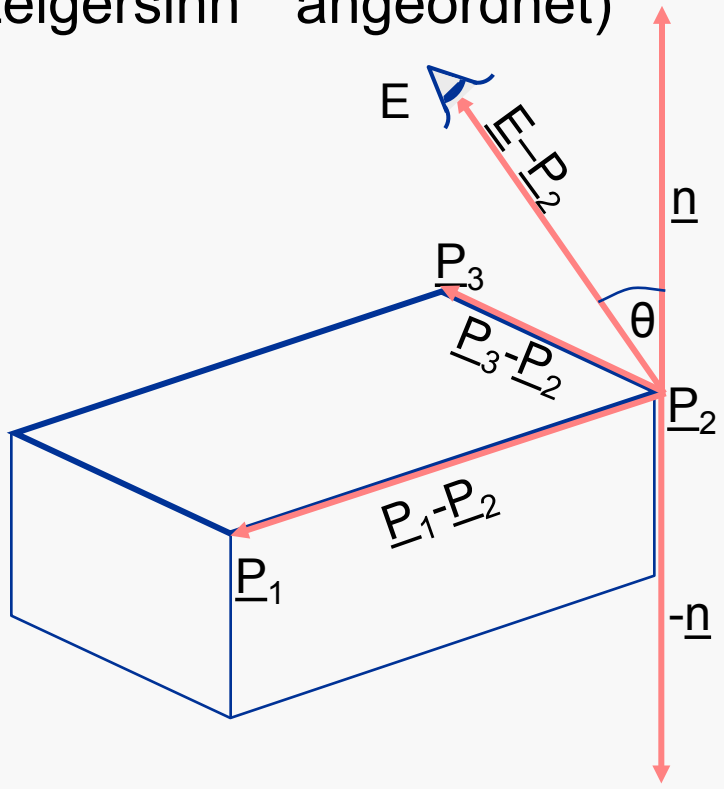
$$\begin{pmatrix} Nx_i/(-z_i) \\ Ny_i/(-z_i) \\ Nz_i/(-z_i) \\ -z_i/(-z_i) \end{pmatrix} = \begin{pmatrix} x_i^* \\ y_i^* \\ -N \\ 1 \end{pmatrix}$$

„Zur Darstellg: $z_i^*=0$ setzen!“



3D-Sicht, Projektionen

Orientierung einer Objektfläche mit den Eckpunkten $\underline{P}_1,\underline{P}_2,\underline{P}_3$ (bei Draufsicht: gegen den Uhrzeigersinn angeordnet) gegenüber dem Augenpunkt E:



Nach außen gerichtete Normale $\underline{n}$:

$$\underline{n} = (\underline{P}_3 - \underline{P}_2) \times (\underline{P}_1 - \underline{P}_2)$$

Winkel zwischen der Normalen und dem Verbindungsvektor vom Eckpunkt P_2 zum Augenpunkt E:

$$\cos\theta = \underline{n} \cdot (\underline{E} - \underline{P}_2) / (|\underline{n}| \cdot |\underline{E} - \underline{P}_2|)$$

$\underline{n} \cdot (\underline{E} - \underline{P}_2) \geq 0 \Leftrightarrow -90° \leq \theta \leq 90° \Leftrightarrow$ sichtbare Fläche

$\underline{n} \cdot (\underline{E} - \underline{P}_2) < 0 \Leftrightarrow 90° < \theta < 270° \Leftrightarrow$ nicht sichtbare (Rück-)Fläche