7. Visibilität

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 Sicht-volumen* (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

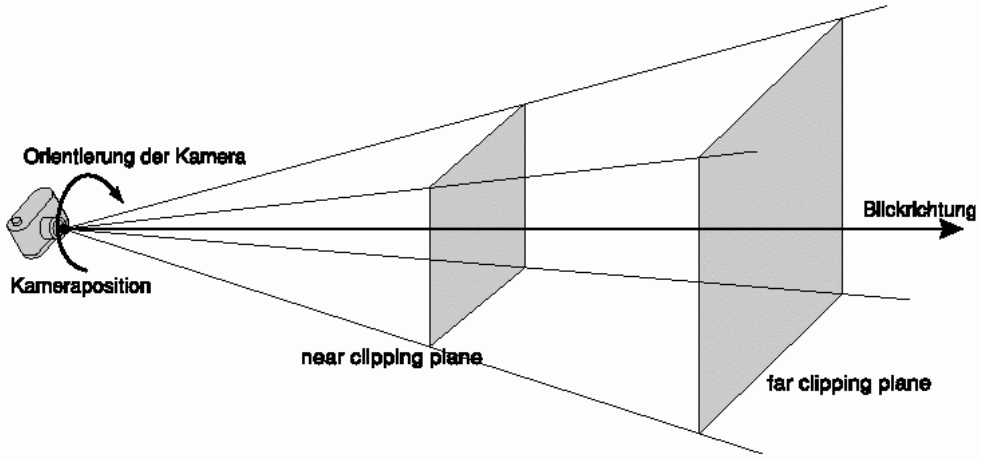
- engl.: view volume
- Sichtkörper enthält alles, was vom COP (oder DOP) aus sichtbar ist (=was die Kamera sieht)
- normalerweise ein Kreiskegel → teure Berechnungen, insbesondere beim Clipping von Objekten gegen eine Kegelfläche
- Approximation durch Pyramide(nstumpf)
  - die Ausgabe ist sowieso rechteckig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

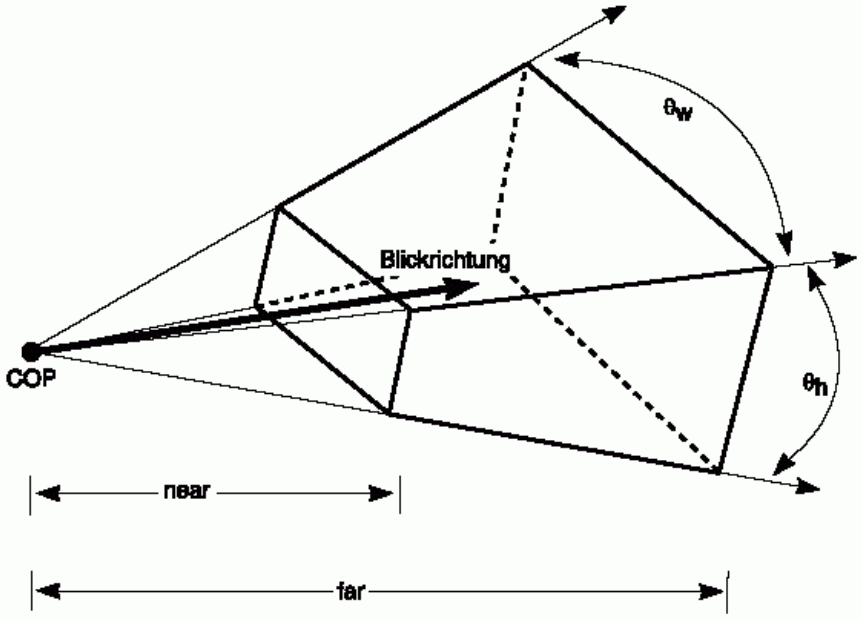
- . . . kann Sichtkörper für folgende Projektionen erstellen:
  - perspektivische Projektion mit positivem Öffnungswinkel
  - Parallelprojektion (Öffnungswinkel ist Null)
- . . . kann nicht für schräge Projektionen verwendet werden

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

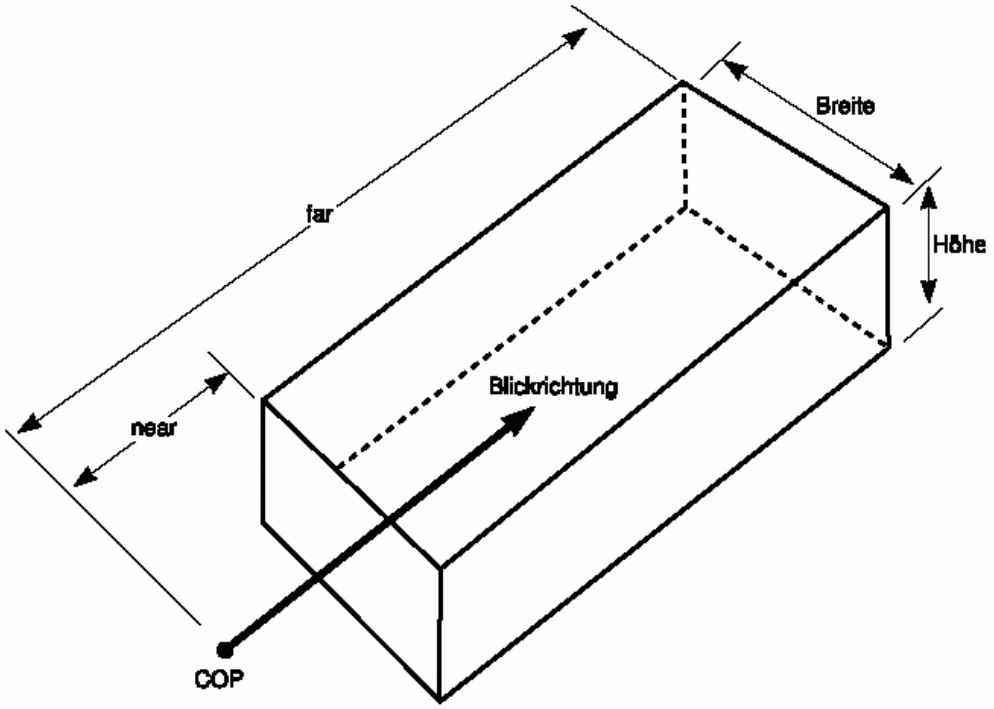
Ziel: Herleitung einer Transformationsmatrix (für Objektdarstellung in homogenen Koordinaten)

- 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

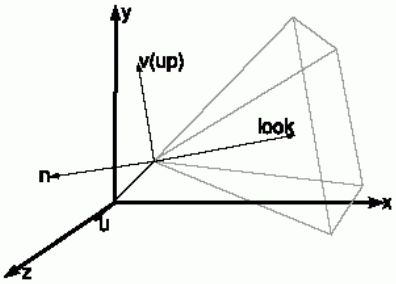


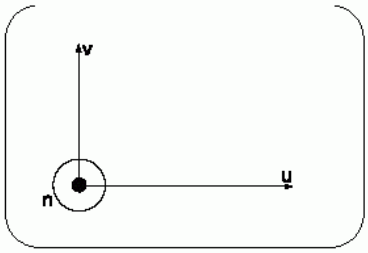
Spezfifkation einer beliebigen 3D-Ansicht

- Plazierung des Sichtkörpers spezifiziert durch:
  - Position (COP)
  - Orientierung (Blickrichtung, up vector)
- Form des Sichtkörpers spezifiziert durch
  - Öffnungswinkel
  - near clipping plane, far clipping plane
- perspektivische Projektion: Projektionsstrahlen schneiden sich im COP
- Parallelprojektion: Projektionsstrahlen parallel zur DOP, schneiden sich nie

Koordinatensysteme

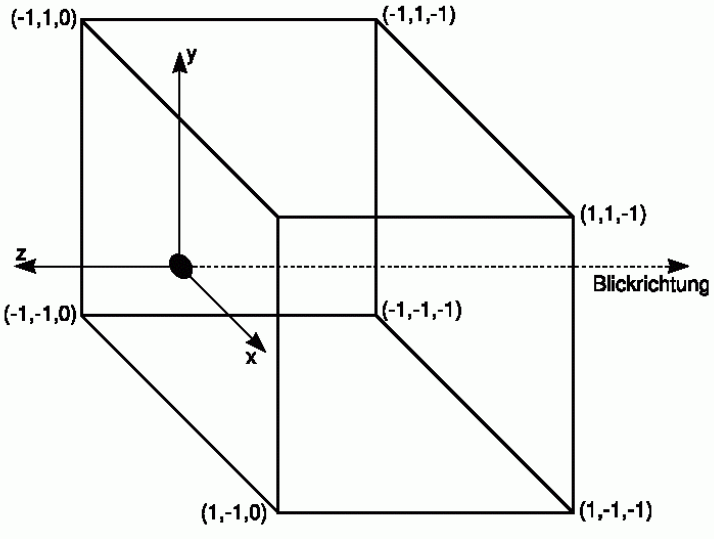
- Weltkoordinaten – Definition des Modells, der Kameraparameter
- View-Referenz-Koordinaten (Kamerakoordinaten) – Ursprung im COP, Achsen rotiert je nach Orientierung der Kamera





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:

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

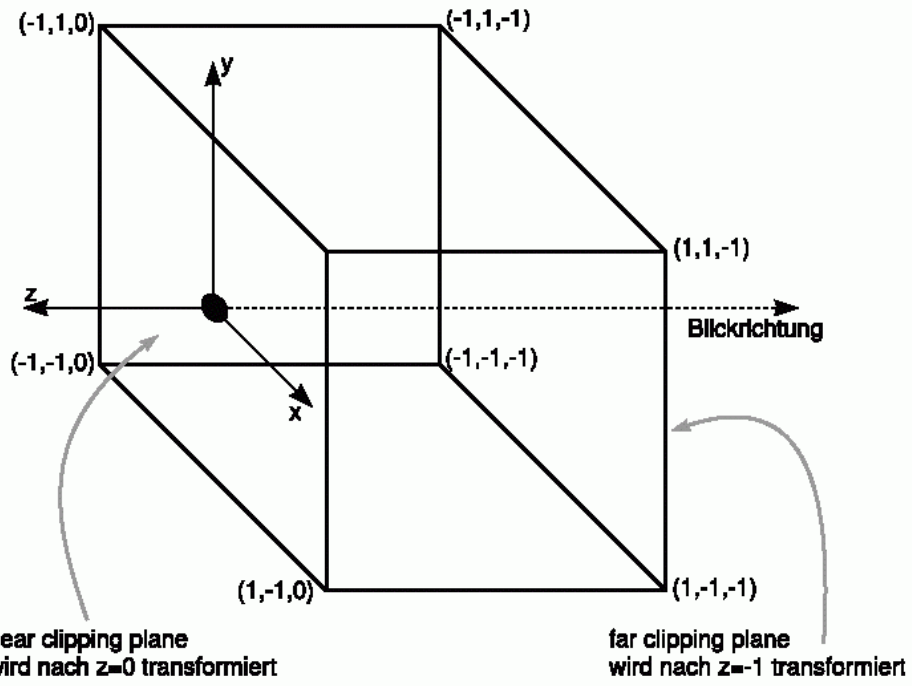
Damit sind die Komponenten der Viewing-Transformation vollständig bekannt.
Dazu inverse Transformation: Normalisierungstransformation, bildet den gegebenen Sichtkörper auf den kanonischen ab.

Transformation in den kanonischen Sichtkörper

- das Problem für Parallelprojektionen:
  Gegeben sei die Spezifikation einer Parallelprojektion und eine Menge von Objektpunkten. Wir nutzen die Normalisierungstransformation, um den Sichtkörper zu transformieren, dann erfolgt das Clipping und dann die Projektion (ignorieren der z-Koordinate).

- das Problem für perspektivische Projektionen:
  Transformation in eine Standard-Spezifikation für eine Parallelprojektion und dann ein zusätzlicher Schritt: Transformation des perspektivischen Sichtkörpers in einen parallelen, dann Clipping und Projektion

Der Standard-Sichtkörper



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: $(t_x, t_y, t_z) = (-x_{\mathrm{COP}}, -y_{\mathrm{COP}}, -z_{\mathrm{COP}})$

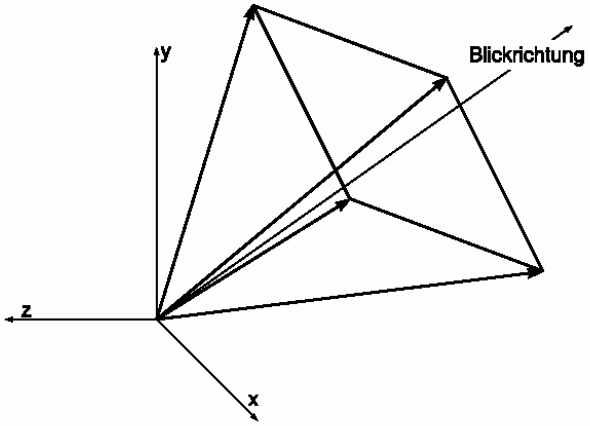
$$T(-\mathrm{COP}) = \begin{pmatrix} 1 & 0 & 0 & -x_{\mathrm{COP}} \\ 0 & 1 & 0 & -y_{\mathrm{COP}} \\ 0 & 0 & 1 & -z_{\mathrm{COP}} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

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

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

Momentane Situation:

COP liegt jetzt im Ursprung



Schritt 2

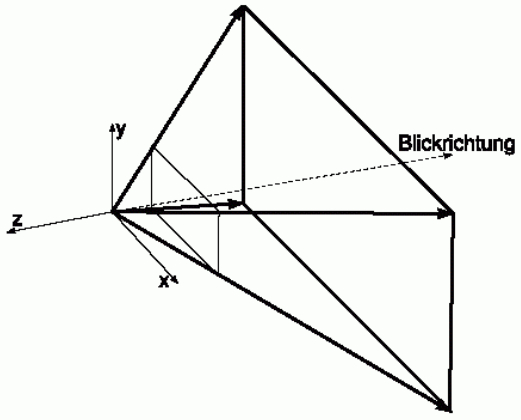
Rotation und Angleichung an das Weltkoordinatensystem:

- 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

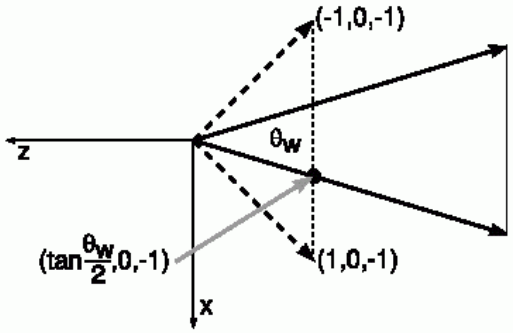


Schritt 3

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: $\theta_w = 1/\left(\tan\frac{\theta_w}{2}\right) = \cot\frac{\theta_w}{2}$
- 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* nocht nicht auf *z* = –1 normiert.
Abstand vom Projektionszentrum (COP) zur *far clipping plane* ist immer noch **`far`**.

- uniforme Skalierung mit folgender Matrix:

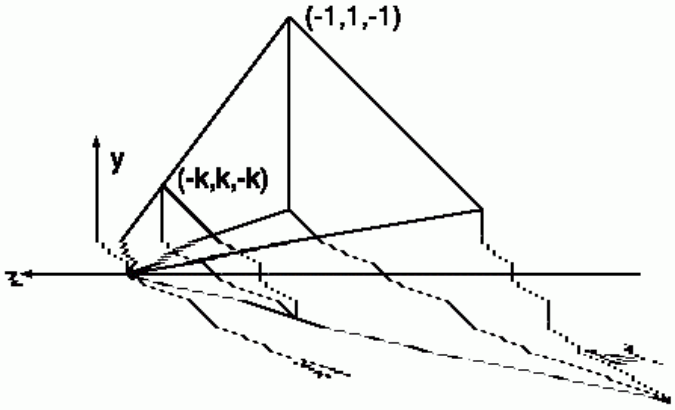
$$S_{xyz}\left(\frac{1}{\mathrm{far}}\right) = \begin{pmatrix} \frac{1}{\mathrm{far}} & 0 & 0 & 0 \\ 0 & \frac{1}{\mathrm{far}} & 0 & 0 \\ 0 & 0 & \frac{1}{\mathrm{far}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

ntane Situation:

Mome

clipping plane bei $z = -1$

- far c



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

Ergebnis

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

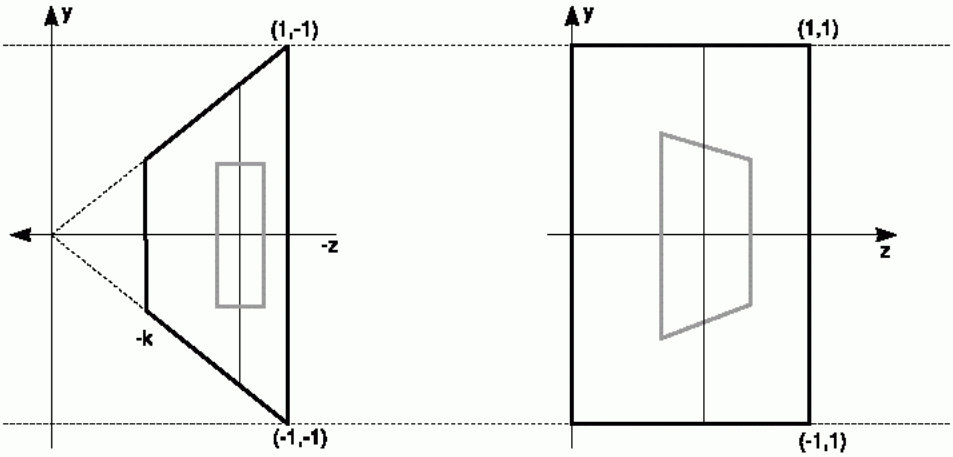
$$S_{xyz}\left(\frac{1}{\text{far}}\right) S_{xy} M^T T(-\text{COP})$$

- 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

es kann eine Variante des Cohen-Sutherland-Algorithmus angewandt werden:

6-Bit-Binärcode anstatt 4-Bit im planaren Fall

Bit 5: gesetzt für Punkte (*x*, *y*, *z*) oberhalb des Sichtvolumens
Bit 4: unterhalb
Bit 3: rechts
Bit 2: links
Bit 1: hinter dem Sichtvolumen
Bit 0: vor dem Sichtvolumen

damit dann analoge Oder- und Und-Abfrage wie im planaren Cohen-Sutherland-Algorithmus. (Vgl. Rauber 1993, S. 147 f.)

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:

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

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