Vorlesung
Computergrafik

Media Informatics
Graphics & Geometric Computing
Prof. Dr. M. Campen

Übersicht

Tricks & Effekte
Texturing
Environment Mapping
Displacement Mapping
Anti-Aliasing

Repräsentation
Polygonnetze
Bézier-Flächen
Splines/NURBS
Volumendaten

Globale Beleuchtung
Raytracing
Radiosity

3D Daten → **Positionieren Anordnen** → **Projizieren** → **Beleuchten** → **Sichtbarkeit Schatten** → **2D Bild**

GPU ausnutzen
(OpenGL, WebGL)

Projektionen (3D nach 2D)

- Parallelprojektion, orthogonal auf z = 0 Ebene:

$$P_{\text{para-z}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- Perspektivische Projektion, auf z = -1 Ebene, Proj.-Zentrum bei (0,0,0):

$$p' = \frac{1}{-p_z} p$$

$$(p'_x, p'_y, p'_z) = \left(\frac{p_x}{-p_z}, \frac{p_y}{-p_z}, \frac{p_z}{-p_z} \right) = \left(\frac{p_x}{-p_z}, \frac{p_y}{-p_z}, -1 \right)$$

  - Problem: Division durch $-p_z$ lässt sich durch Matrix-Vektor-Produkt nicht darstellen.

Homogene Koordinaten

- (x,y,z,w) mit w $\neq$ 0 meint den 3D Punkt (x/w, y/w, z/w)
- (x,y,z,0) meint den 3D Vektor (x,y,z)

- Das heißt insbesondere: (x/a, y/a, z/a, 1) und (x, y, z, a) meinen den selben Punkt. Daher: Division durch a lässt sich indirekt ohne Division ausdrücken.
- Die perspektivische **Standardprojektion** nach der Regel $p' = \frac{1}{-p_z} p$ lässt sich schreiben als:

$$P_{\text{std}} \cdot p = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix}$$

Allgemeinere perspektivische Projektion

- Perspektivische Projektion, Proj.-Zentrum bei (0,0,0), auf Ebene senkrecht zu Vektor $\vec{n}$ mit Abstand δ vom Proj.-Zentrum

$$p' = \frac{\delta}{\vec{n}^{\mathsf{T}} p} \cdot p$$

$$P_{\vec{n},\delta} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \frac{n_x}{\delta} & \frac{n_y}{\delta} & \frac{n_z}{\delta} & 0 \end{bmatrix}$$

- Spezialfall:

$$P_{\text{std}} = P_{(0,0,-1),1}$$

LookAt-Transformation

- Es genügt, die Standard-Projektion P_{std} zu implementieren, da wir beliebige andere Projektionszentren und Blickrichtungen (d.h. Bildebenenausrichtungen) durch Koordinatensystemwechsel realisieren können.

- Denn: wendet man die LookAt-Transformation auf "die Kamera" an, so wird diese auf den Ursprung, in Richtung -z blickend, abgebildet. Dies zeigt, dass wir es anschließend einfach nur noch mit P_{std} zu tun haben.

Screen-Koordinaten

- Nach der Projektion befinden sich alle Punkte in der Bildebene. Sie sind jedoch immer noch durch 3D-Koordinaten repräsentiert.

- Daher nun: Konvertieren in 2D-Koordinatensystem, welches die Bildebene aufspannt.

- 2D-System in der Bildebene: Ursprung c, Achse u, Achse v
    - Wir nehmen an, dass das System orthonormal ist, d.h.
        - u und v sind senkrecht zueinander (“*ortho*”)
        - u und v haben die Länge 1 (“*normal*”)
    - Dann: $$x' = u^\top (p - c) = u^\top p - u^\top c$$ $$y' = v^\top (p - c) = v^\top p - v^\top c$$
    - Also: $$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} u_x & u_y & u_z & -u^\top c \\ v_x & v_y & v_z & -v^\top c \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$ nachdem der Punkt rechts *dehomogenisiert* wurde (“*perspektivische Division*” durch die 4. Komponente)

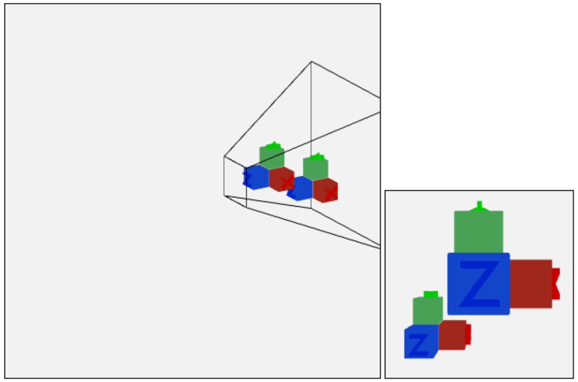
Frustum-Transformation

- Nach der (perspektivischen) Projektion liegen alle Punkte in einer Ebene; ihre ursprüngliche "Tiefe" (Entfernung von der Bildebene) ist nicht mehr ablesbar. Wir benötigen diese Information jedoch später noch für Sichtbarkeitsentscheidungen.

- Lösung: Perspektivische Projektion in zwei Teile aufspalten:

  - Frustum-Transformation gefolgt von Parallelprojektion

$$P_{\text{std}} = P_{\text{para-z}} \cdot M_{\text{Frustrum}}$$



[Interactive Demo]

Frustum-Transformation

- Nach der (perspektivischen) Projektion liegen alle Punkte in einer Ebene; ihre ursprüngliche "Tiefe" (Entfernung von der Bildebene) ist nicht mehr ablesbar. Wir benötigen diese Information jedoch später noch für Sichtbarkeitsentscheidungen.

- Lösung: Perspektivische Projektion in zwei Teile aufspalten:

  - Frustum-Transformation gefolgt von Parallelprojektion

$$P_{\text{std}} = P_{\text{para-z}} \cdot M_{\text{Frustrum}}$$

$$M_{\text{Frustrum}} = \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{-f-n}{f-n} & \frac{-2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}$$

n, f Distanz der near/far-Ebenen vom Proj.-Zentrum

t, b, l, r Position der top, bottom, left, right Ränder des sichtbaren Bereichs relativ zur Blickachse

Viewport-Transformation

- Nach der Frustum-Transformation befindet sich die Region der 3D-Szene, die zwischen den top, bottom, left, right, near, far Ebenen liegt, im Bereich [-1,1]x[-1,1]x[-1,1]. Die folgende Parallelprojektion lässt im Grunde lediglich die z-Koordinate wegfallen. Das Ergebis liegt also in [-1,1]x[-1,1]. Dieser Bereich muss dann lediglich noch auf einen gewünschten rechteckigen Bereich (den *"Viewport"*) innerhalb des Bildschirmkoordinatensystems abgebildet werden (durch Translation und Skalierung):

$$M_{\text{Viewport}} = \begin{bmatrix} \frac{w}{2} & 0 & 0 & l + \frac{w}{2} \\ 0 & \frac{h}{2} & 0 & b + \frac{h}{2} \end{bmatrix}$$

  wobei die linke untere Ecke des Viewport bei (l, b) (*"left, bottom"*) liegt und der Viewport (w, h) (*"width, height"*) groß ist.

- (Spezialfall der Screen-Koordinaten-Transformation für die Standardproj.)

Zusammenfassung:

- Punkte durchlaufen, durch Anwendung verschiedener Transformationen, eine Reihe von Koordinatensystemen — beginnend bei Modell-Koordinaten, ended bei Bild(schirm)-Koordinaten:

4D (homog.)	4D (homog.)	4D (homog.)	3D (dehomog.)	2D
Model	**World**	**Eye/Camera**	**NDC**	**Screen**

Translation
Rotation
Skalierung
…

LookAt
…

Frustum
+ Dehomogenisierung
("*perspective division*")

"ModelView"

"Projection"

"Viewport"

Bis zum nächsten Mal!

Media Informatics
Graphics & Geometric Computing
Prof. Dr. M. Campen