Seminar 3D-Grafik

Mathematische Grundlagen, Räume, Koordinatensysteme, Projektionen

Hermann Schwarz und Marko Pilop
{hschwarz|pilop}@informatik.hu-berlin.de
http://www.informatik.hu-berlin.de/~pilop/3D-basics_A.pdf

WS 2002/2003
2004-06-21

Humboldt-Universität zu Berlin
Institut für Informatik

Inhaltsverzeichnis

1 Vektoren und Vektorräume

Da die Objekte in der Computergrafik als Punktmengen im zwei- oder dreidimensionalen Zahlenraum betrachtet werden, für den die Gesetze der Vektor- bzw. der linearen Algebra gelten, machen wir zuerst eine kleine Einführung in die n-dimensionale Vektorräume (Euklidische Räume), Matrizen und geometrische Objekte.
Die mit der linearen Algebra eng verbundene analytische Geometrie wird uns hier mit der Berechnung der Flächeninhalte geometrischer Objekte und mit Lösungen anderer geometrischer Probleme mit Hilfe der Vektorrechnung bekanntmachen.

Ein Vektor x ist ein Element des n-dimensionalen Euklidischen Zahlenraums und wird durch reelwertige Koordinaten $x_1, ..., x_n$ repräsentiert.

Schreibweise als Spaltenvektor: $x^T = \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix}$

Schreibweise als Zeilenvektor: $x = (x_1, ..., x_n)$

Abgeschlossenheit bzgl. Multiplikation mit skalaren Größen und bzgl.Vektoraddition:
$x \in \mathbf{R^n}, \alpha \in \mathbf{R}$:
$\alpha \cdot x = (\alpha \cdot x_1, \alpha \cdot x_2, ..., \alpha \cdot x_n) \in \mathbf{R^n}$

$x, y \in \mathbf{R^n}$
$x + y = (x_1 + y_1, x_2 + y_2, ..., x_n + y_n) \in \mathbf{R^n}$

Linearkombination:
$\xi, x, y, z \in \mathbf{R^n}$, $\alpha, \beta, \gamma \in \mathbf{R}$
ξ heisst Linearkombination von x, y, z, falls $\xi = \alpha \cdot x + \beta \cdot y + \gamma \cdot z$

Lineare Unabhängigkeit:
Sind wieder ξ, x, y, z Elemente eines Vektorraumes, heisst diese Menge linear unabhängig, falls keiner von ihnen als Linearkombination der anderen Vektoren darstellbar ist. Anderenfalls sind sie linear abhängig.
Und im Falle linearer Abhängigkeit gibt es $\alpha, \beta, \gamma, \delta \in \mathbf{R}$, so dass
$\alpha \cdot x + \beta \cdot y + \gamma \cdot z + \delta \cdot \xi = 0$ - Nullvektor
Im Falle linearer Unabhängigkeit gibt es solche skalaren Koeffizienten nicht.

Skalarprodukt: $x, y \in \mathbf{R^n}$
$x \cdot y = \sum_{i=1}^{n} = x_i \cdot y_i$

Geometrische Deutung des Skalarprodiktes:
$x \cdot y =$Länge des auf den Vektor y projizierten Anteils von x, multipliziert mit der Länge von y.
$y \cdot x =$ analog.
Folgerung: $x, y \in \mathbf{R^n}$ und x, y sind orthogonal zueinander
$\Rightarrow x \cdot y = 0$

Länge oder Betrag eines Vektors:
$|x| = \sqrt{x \cdot x}$

2 Matrizen und Matrizenmultiplikation

In der 3D-Programmierung hat jedes Objekt ein eigenes (lokales) Koordinatensystem. Das Objekt ist meistenes genau zentral Um dieses Koordinatensystem positioniert. Dabei besteht das Objekt nur aus Ortsvektoren (vom Koordinatenursprung zu den Punkten des Objekts). Um dieses Objekt auf den Bildschirm zu bringen, muss es erst verschoben, rotiert oder skaliert werden. Um diese Operationen zu berechnen, verwendet man Matrizen. Also können sowohl geometrische Objekte als auch deren Transformationen durch Matrizen beschrieben werden.
Ohne Matrizen müssten wir mehrere Formeln verwenden, um das Objekt richtig zu positionieren. Das würde Rechenzeit kosten. Wir erstellen zwar mehrere Matrizen um das Objekt zu bewegen, jedoch können wir diese miteinander multiplizieren und am Schluß bleibt nur noch eine einzige Matrix.

Sei eine Matrix $A_{m,n} = \begin{pmatrix} a_{11} & a_{12} & ... & a_{1n} \\ a_{21} & a_{22} & ... & a_{2n} \\ . & ... & ... & ... \\ . & ... & ... & ... \\ . & ... & ... & ... \\ a_{m1} & ... & ... & a_{mn} \end{pmatrix}$

$\Rightarrow$

Produkt einer Matrix $A_{m,n}$ mal Spaltenvektor $x = \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix}$:

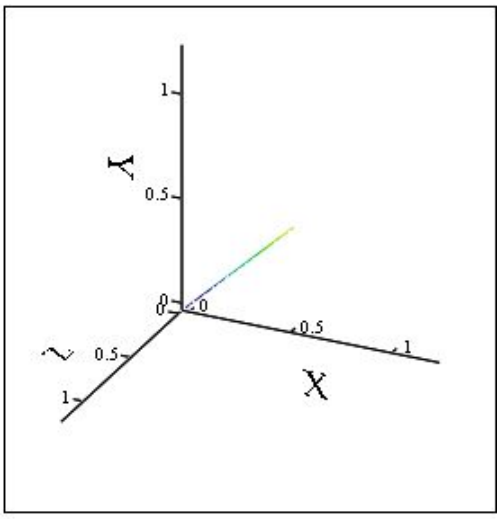
$$A_{m,n} \cdot \begin{pmatrix} x_1 \\ x_2 \\ . \\ . \\ . \\ x_n \end{pmatrix} = \begin{pmatrix} z_1 \\ z_2 \\ . \\ . \\ . \\ z_m \end{pmatrix} \text{ ,wobei } z_i = A_{m,} \cdot z = \sum_{j=1}^{n} a_{i,j} \cdot x_j$$

Produkt zweier Matrizen $A_{m,n}$ und $B_{n,k}$ (Voraussetzung: Beide Matrizen aus dem selben Vektorraum, d.h. Anzahl der Spalten der ersten Matrix muss der Anzahl der Zeilen der zweiten Matrix gleich sein):
$A_{m,n} \cdot B_{n,k} = C_{m,k}$,
wobei die Komponenten $c_{i,j}$ der Matrix $C_{m,k}$ Skalarprodukte der i-ten Zeile von A mit der j-ten Spalte von B sind:
$c_{i,j} = \sum_{l=1}^{n} (a_{i,l} \cdot b_{l,j})$

Wichtige Eigenschaften der Matrix- und Vektormultiplikationen:
Assziativität: $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
Kommutativität: $A \cdot B \neq B \cdot A$ - Matrixmultiplikation ist i.a. nicht kommutativ.

3 Objekte im 2- und 3-dimensionalen kartesischen Koordinatensystem

Ein dreidimensionales, rechtshändiges (Reihenfolge x-y-z-Achse dem Uhrzeigersinn entgegengesetzt) kartesisches Koordinatensystem wird aufgespannt durch drei orthonormalen Einheitsvektoren:



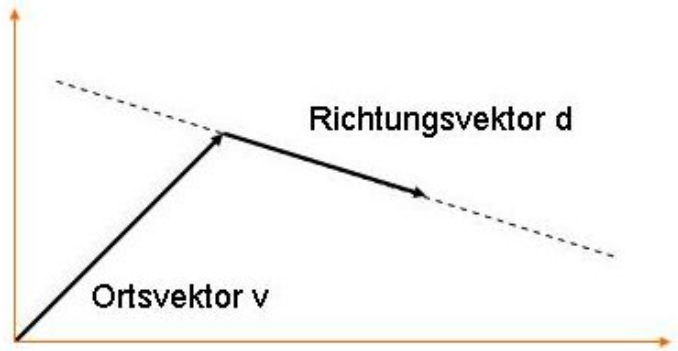
(X,Y,Z)

Die Punkte in diesem Raum werden durch Ortsvektore repräsentiert, die vom Koordinatenursprung zum jeweiligen Punkt p mit Koordinaten (p_x, p_y, p_z) laufen.

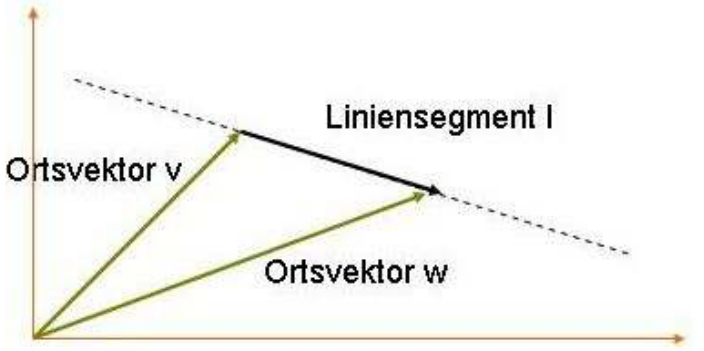
Mit Richtungsvektoren bezeichnet man die Differenzvektore zwischen zwei Ortsvektoren (Punkten).

3.1 Linien, Ebenen, deren Schnittpunkte und Schnittgeraden

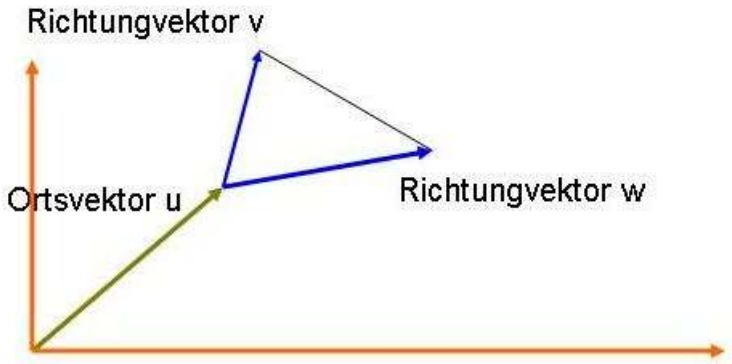
Mit Linearkombination von zwei Vektoren kann man eine unendlich ausgedehnte Linie darstellen:
Seien v (Ortsvektor) und d (Richtungsvektor) Vektore, dann wird so eine Linie durch
$l = v + \lambda \cdot d$;
$-\infty < \lambda < +\infty$ definiert



Um eine begrenzte Linie darzustellen, brauchen wir zwei Ortsvektore, die diese Linie begrenzen. Sie kann dann durch $l = v + \lambda(w - v)$;
$0 \leq \lambda \leq 1$ ausgedrückt werden.



Um eine Ebene darzustellen, braucht man zwei linear unabhängige Vektore, die diese Ebene aufspannen. Eine Ebene kann so ausgedrückt werden (explizite Form):
Seien u, v, w Vektore, wobei u ein Ortsvektor und v, w linear unabhängige Richtungsvektore sind
$p = u + \alpha \cdot v + \beta \cdot w$ ist dann die Ebene, die von Vektoren v und w aufgespannt wird. Dabei sind α und β unbegrenzt.
Ähnlich einer Ebene wird auch ein Dreieck definiert. Nur die Parameter α und β werden begrenzt, damit keine Punkte außerhalb des von zwei linear unabhängigen Vektoren aufgespannten Dreiecks ausgedrückt werden könnten:
$t = u + \alpha \cdot v + \beta \cdot w$;
$0 \leq \alpha \leq 1$; $0 \leq \beta \leq 1 - \alpha$
ist dann das Dreieck, das im Punkt u ein Eck hat und von dort aus ausgehenden Vektoren v und w aufgespannt wird.



Implizite Form für Ebenen:

Hier werden Begriffe Betrag eines Vektors und Vektorprodukt benutzt:
Betrag eines Vektors: $|v| = \sqrt{v_x + v_y + v_z}$

Vektorprodukt (Kreuzprodukt) der Vektore v und w:

$$v \times w = det \begin{pmatrix} x & y & z \\ v_x & v_y & v_z \\ w_x & w_y & w_z \end{pmatrix} = \begin{pmatrix} x & y & z \end{pmatrix} \cdot \begin{pmatrix} v_y w_z & - & v_z w_y \\ v_z w_x & - & v_x w_z \\ v_x w_y & - & v_y w_x \end{pmatrix}$$

wobei der Vektor $\begin{pmatrix} v_y w_z & - & v_z w_y \\ v_z w_x & - & v_x w_z \\ v_x w_y & - & v_y w_x \end{pmatrix}$ senkrecht auf v und w steht.

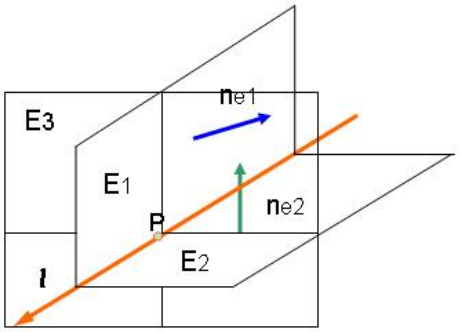
Der Betrag dieses Vektors ist gleich dem Flächeninhalt des durch die Vektoren v und w aufgespannten Parallelogramms.

Implizite Form für Ebenen wird aus der parametrischen Form $p = u+\alpha \cdot v+\beta \cdot w$
Aus dem Einschub können wir ableiten, dass das Vektorprodukt $n = v \times w$ eine Flächennormale bildet. Außerdem gilt es, dass die Differenzvektoren zwischen beliebigen Punkten $p^{'}$ der Ebene und Punkt u senkrecht zur Flächennormalen n stehen.
$\Longrightarrow (p^{'} - u) \cdot n = p^{'} \cdot n - u \cdot n = 0$
Mit $d := u \cdot n$
und Normierung der Gleichung mit $|n|$ ergibt sich die Ebenengleichung:
$\frac{p^{'} \cdot n}{|n|} - \frac{d}{|n|} = 0$
oder $p^{'} \cdot n - d = 0$
$\Longrightarrow p^{'} \cdot n = d$
und die Konstante d wird zum Abstand der Ebene zum Koordinatennullpunkt.

Linie-Ebene-Schnitt:

Die Ebenengleichung $p^{'} \cdot n = d$ und die Liniengleichung $l = v + \lambda \cdot w$
sind gegeben.
Durch Substituieren der Punkte der Linie in die Ebenengleichung
$(v + \lambda \cdot w) \cdot n = d$
erhält man nach Auflösen nach λ den Schnittpunktparameter $\lambda_s = \frac{d - n \cdot v}{n \cdot w}$
Setzt man λ_s in die gegebene Liniengleichung ein, so erhält man den Schnittpunkt p_s der gegebenen Linie mit der Ebene $p^{'} \cdot n = d$:
$p_s = v + \lambda_s \cdot w$

Ebene-Ebene-Schnitt:

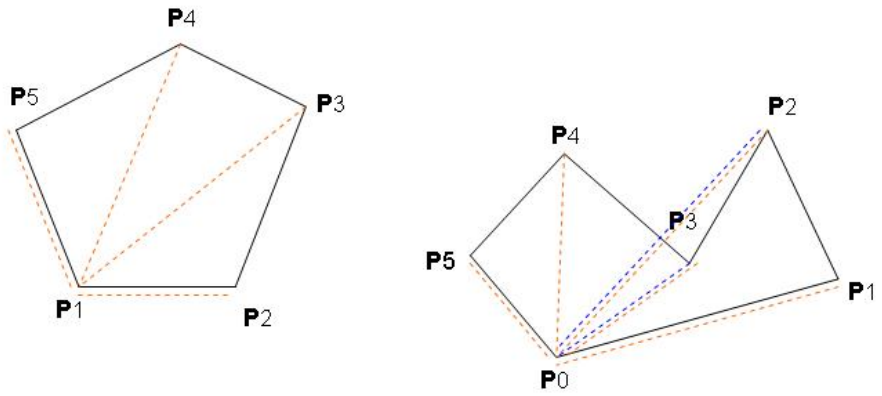


Zwei Ebenen sind durch implizite Gleichungen gegeben:
E_1: $n_{E_1} \cdot P - d_1 = 0$
E_2: $n_{E_2} \cdot P - d_2 = 0$
Der Richtungsvektor der Schnittlinie l ergibt sich aus dem Kreuzprodukt der beiden Normalen n_{E_1} und n_{E_2}, denn die Schnittlinie senkrecht zu ihnen liegt. Für die Schnittliniengleichung bleibt nur noch einen beliebigen Punkt P dieser Linie zu finden. Dazu konstruiert man eine dritte Ebene, deren Flächennormale parallel zu der Schnittlinie ist. Das erlaubt uns die Schnittlinie l als Flächennormale dritter Ebene in der Gleichung für diese dritte Ebene zu benutzen. Die dritte Ebene definieren so, dass sie durch den Ursprung geht ($d = 0$):
E_3: $n_{E_3} \cdot P - d_3 = l_x \cdot P_x + l_y \cdot P_y + l_z \cdot P_z - 0 = 0$
wobei d_3 Abstand der Ebene E_3 zum Ursprung ist, l_x, l_y, l_z Komponenten des Richtungsvektors l und P_x, P_y, P_z Koordinaten des Punkts P sind.
Da der Punkt P der Schnittpunkt aller Drei Ebenen ist, muss er die Gleichun-

gen aller Drei Ebenen erfüllen. Seine Koordinaten werden demnach durch das Lösen eines linearen Gleichnungssystem aus drei Ebenengleichungen:

$$\begin{pmatrix} l_x & l_y & l_z \\ n_{E_{1x}} & n_{E_{1y}} & n_{E_{1z}} \\ n_{E_{2x}} & n_{E_{2y}} & n_{E_{2z}} \end{pmatrix} \cdot \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = \begin{pmatrix} 0 \\ d_1 \\ d_2 \end{pmatrix}$$

3.2 Polygone und deren Flächeninhalte



Konvexes Polygon **Nicht-Konvexes Polygon**

Um Flächeninhalte von Polygonen zu berechnen unterteilt man Polygone in Dreiecke, man berechnet die Flächeninhalte von Dreiecken und addiert deren Flächeninhalte:

Fläche des Polygons $F = \frac{1}{2}|\sum_{i=1}^{n-2}(P_i - P_0) \times (P_{i+1} - P_0)|$

Diese Formel kann man sowohl für die konvexen als auch für die nicht-konvexen Polygone anwenden, denn die bei den nicht-konvexen Polygonen mehrfach gezählten oder außerhalb des Polygons liegenden Dreieckflächen werden durch die unterschiedlichen Vorzeichen des Kreuzprodukts ausgeglichen, so dass man richtiges Ergebnis erhält.

4 Geometrische Transformationen in 2D- und 3D-Koordinatensystem

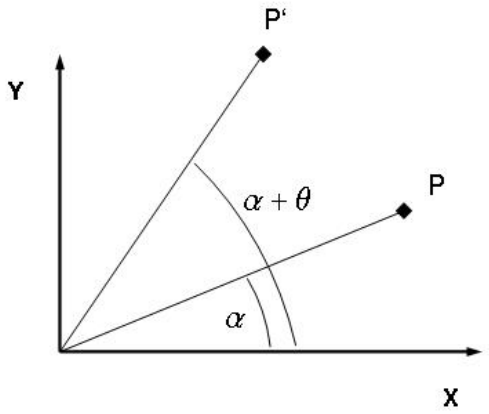
4.1 Translation

Bei einer Verschiebung eines Objekts werden die Koordinaten der Punkte dieses Objekts mit einem Richtungsvektor addiert.

4.2 Skalierung

Hier handelt es sich um eine Streckung oder Stauchung eines Objekts bezüglich des Koordinatenurprungs.
Uniforme Skalierung wird realisiert durch skalare Multiplikation der Ortsvektore des Objekts mit dem Skalierungsfaktor.
Nicht-uniforme Skalierung funktioniert wie die uniforme, nur der Skalierungsfaktor ist für jeden Komponenten des Ortsvektors unterschiedlich.

4.3 Rotation



Für die Darstellung einer Drehung eines Objekts werden Polarkoordinaten als Hilfsmittel herangezogen. Die Polarkoordinaten eines Punktes ergeben sich aus dem Abstand dieses Punktes zum Ursprung und dem Winkel zwischen x-Achse und dem Ortsvektor dieses Punktes.
Seien die Polarkoordinaten eines Punktes P des gegebenen Objekts (r, α)
Aus den Verhältnissen $r = \sqrt{x^2 + y^2}$,
$x = r \cdot cos\alpha$, $y = r \cdot sin\alpha$ (*)
folgt, dass sich die neuen Polarkoordinaten des um den Winkel θ gedrehten Punktes P sehr einfach darstellen lassen: $P^{'} = (r, \alpha + \theta)$
Um die kartesischen Koordinaten zu erhalten, setzt man in (*) den neuen Winkel ein. Der Abstand zum Koordinatenursprung bleibt natürlich unverändert:
$x^{'} = r \cdot cos(\alpha + \theta)$, $y^{'} = r \cdot sin(\alpha + \theta)$
Nach Anwenden des Satzes für *Sinus* und *Cosinus* einer Summe zweier Winkel erhält man
$x^{'} = r \cdot cos\alpha \cdot cos\theta - r \cdot sin\alpha \cdot sin\theta$,
$y^{'} = r \cdot cos\alpha \cdot sin\theta + r \cdot sin\alpha \cdot cos\theta$

Und wegen (*) erhält man
$x^{'} = x \cdot cos\theta - y \cdot sin\theta$,
$y^{'} = x \cdot sin\theta + y \cdot cos\theta$

Dies lässt sich als Matrixoperation für Spaltenvektoren darstellen:

$$\begin{pmatrix} x^{'} \\ y^{'} \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

5 Matrixdarstellung von Transformationen in homogenen Koordinaten

5.1 Homogene Koordinaten

Dass die bereits vorgestellten Transformationen nicht einheitlich dargestellt wurden (durch Vektoraddition, skalare Multiplikation oder Matrizenoperation) heißt nicht, dass es nicht einheitlich geht. Die Einführung einer zusätzlichen Dimension erlaubt es, alle Transformationen einheitlich als Matrixoperationen darzustellen. Man arbeitet dann nicht einfach im kartesischen Vektorraum, sondern in dem sogenannten homogenen Vektorraum, der den kartesischen Vektorraum als Teilraum enthält. Die zusätzliche, oder homogene Koordinate ist für die meisten Anwendungen der Computergrafik auf Eins gesetzt.

5.2 Translation und Rotation

Um die Translationsmatrix zu erhalten, löst man das Gleichungssystem aus dem Verhältnis (Translationsmatrix angewendet auf ein Ortsvektor in homogenen Koordinaten ergibt neue homogene Koordinaten):

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + \Delta x \\ y + \Delta y \\ z + \Delta z \\ 1 \end{pmatrix}$$

Aus diesem Produkt erhält man ein Gleichungssystem, dessen Lösung die Translationsmatrix ergibt:

$$\begin{pmatrix} 1 & 0 & 0 & \Delta x \\ 0 & 1 & 0 & \Delta y \\ 0 & 0 & 1 & \Delta z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Die homogene Rotationsmatrix lässt sich leicht aus der kartesischen Form ableiten, indem man vierte Dimension einführt. Da im 3-dimensionalen Raum außer der $x-y$-Ebene noch $x-z$ und $y-z$-Ebenen gibt, gibt es auch drei Rotationsmatrizen:

$x-y$-Ebene (Drehung um die z-Achse):

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

$x-z$-Ebene (Drehung um die y-Achse):

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

$y-z$-Ebenen (Drehung um die x-Achse):

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

5.3 Skalierung, Spiegelung, Projektion und Scherung

Sowohl die uniforme als auch die nicht-uniforme Skalierung wird durch folgende Matrix erreicht:

$$\begin{pmatrix} F_x & 0 & 0 & 0 \\ 0 & F_y & 0 & 0 \\ 0 & 0 & F_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Alle Faktoren sind dabei grösser Null und bei uniformer Skalierung gilt zusätzlich: $F_x = F_y = F_z$

Eine Spiegelung wird dadurch erreicht, indem man negative Skalierungsfaktoren einsetzt (-1 für eine Spiegelung ohne Skalierung).

Bei einer Scherung verändern sich Koordinaten einer Dimension in Abhängigkeit von den Koordinatengröße anderer Dimensionen:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & s & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix} = \begin{pmatrix} x \\ y + s \cdot z \\ z \\ w \end{pmatrix}$$

In diesem Fall ist die neue y-Koordinate von der z-Koordinate und Faktor s abhängig. Dabei wird der ursprüngliche Körper immer mehr in die y-RIchtung verschoben, je weiter er von der $x-y$-Ebene entfern ist ($z-Achsen-Richtung$).

5.4 Verknüpfung von Transformationen

Die meisten Transormationen von Objekten in der Computergrafik sind keine einfachen, sondern bestehen aus mehreren solchen Transormationen. Da alle Transformationen einheitlich als Matrizen dargestellt werden können und deren Anwendung nichts anderes als nacheinander ausgeführte Matrizenmultiplikation ist, kann man mehrere auszuführende Transformationen in einer Matrix darstellen. Die Matrizen werden dabei nacheinander von links multipliziert, wobei die Reihenfolge beibehalten werden muss, denn Matrizenmultiplikation i.a. nicht kommutativ ist. Das heißt, es ist nicht egal, in welcher Reihenfolge die einfachen Transformationen nacheinander ausgeführt werden. Das resultierende Produkt wendet man dann von links auf das zu transformirende Objekt an.

5.5 Inverse Transformationen

Um eine durchgeführte Transformation rückgängig zu machen, wird die inverse Matrix der Transformationsmatrix von links angewendet. Wenn dabei die Transformation aus mehreren einfachen Transformationen bestand, werden die iversen Matrizen in umgekehrter Reihenfolge von links angewendet:
Wenn eine Trasformation aus vier einfachen Transformationen besteht:
$(T_4 \cdot T_3 \cdot T_2 \cdot T_1)$, wobei die Indizes die Reihenfolge repräsentieren, die bei einem Hintereinanderausführen von Matrizen eingehalten wäre.
Um diese Transformation rückgängig zu machen, wird auf das Objekt die inverse Matrix $(T_4 \cdot T_3 \cdot T_2 \cdot T_1)^{-1} = T_1^{-1} \cdot T_2^{-1} \cdot T_3^{-1} \cdot T_4^{-1}$ angewendet.

6 Abstände

6.1 Punkt - Ebene

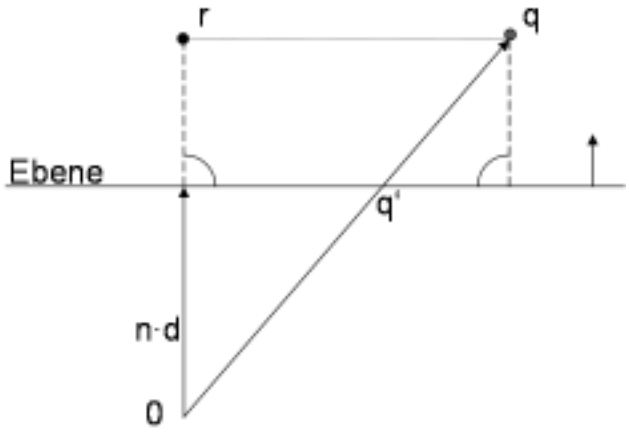


Abbildung 6.1: Abstand von einem Punkt zur Ebene

Sei n der Normalvektor, der die Ebene beschreibt, mit $|n| = 1$. Dann gilt nach der HNF: $n \cdot p = d$. Dabei ist $p \in R^3$.
Der Abstand des Punktes q von der Ebene ist demnach der die Länge des Vektors $|q'q| = n \cdot q - d$.

Mit diesem Lösungsweg ist sind dann auch die Abstände *Gerade-Ebene* und *Ebene-Ebene* analog berechenbar. Dazu wählt man sich bei *Gerade-Ebene* einen Punkt auf der Geraden bzw. bei *Ebene-Ebene* einen Punkt in einer Ebene, um das Problem auf *Punkt-Ebene* zu reduzieren.

6.2 Gerade - Ebene

Wie in Abschnitt 6.1 gezeigt, läßt sich dieser Fall auf die Berechnung des Abstandes *Punkt-Ebene* berechnen.

6.3 Gerade - Gerade

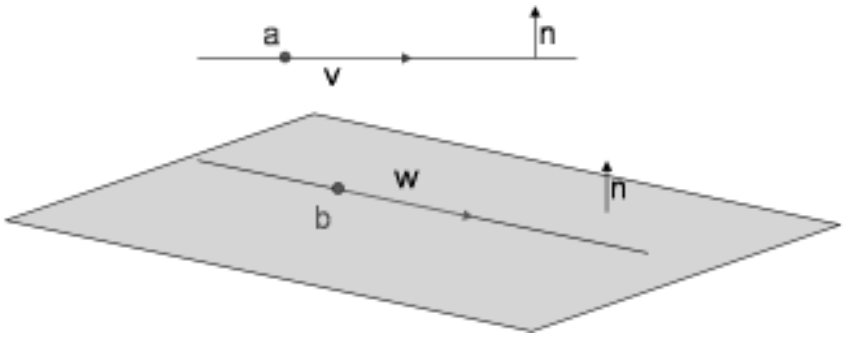


Abbildung 6.2: Abstand von einer Geraden zu einer Geraden

Seien zwei Geraden gegeben, die durch die Punkte a und b in Verbindung mit den Richtungsvektoren v und w. Um den Abstand der beiden Geraden zu bestimmen, greifen ich mir den nicht-trivialen Fall heraus, in dem die Geraden wie in Abbildung 6.2 dargestellt, windschief sind. Sollten sich die Geraden schneiden, so ergibt sich der Abstand $d = 0$.

Zuerst wird eine Fläche konstruiert, in der b liegt und deren Normalvektor n gleich dem Normalvektor der Geraden a ist. Nun ist es möglich, mit der in Abschnitt 6.1 gezeigten *Punkt-Ebene*-Methode den Abstand zu berechnen.

7 Punkt-Tests

Oftmals ist es in der 3D-Grafikprogrammierung nötig herauszufinden, wo sich ein Punkt relativ zu einer Fläche befindet. In diesem Kapitel werden einige dieser Punkt-Tests vorgestellt.

7.1 Punkt im Rechteck

Der Test, ob ein Punkt in einem Rechteck liegt, ist ein Spezialfall des *Punkt im Polygon*-Tests aus Abschnitt 7.2.

7.2 Punkt im Polygon

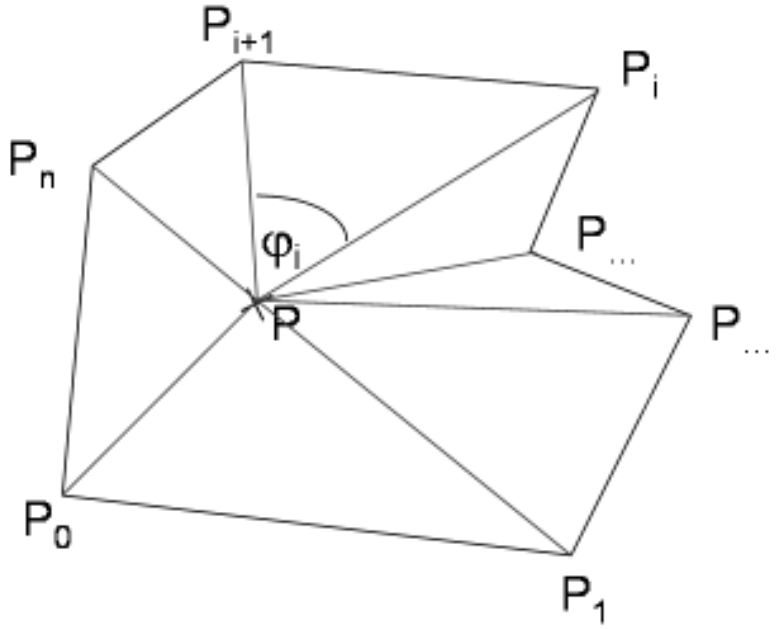


Abbildung 7.1: Befindet sich ein Punkt P im Polygon $P_0 \dots P_n$?

In der Abbildung 7.1 wird veranschaulicht, wie getestet wird, ob der Punkt P in dem Polygon mit den $n+1$ Eckpunkten $P_0 \dots P_n$ liegt, oder nicht.
Dazu wird vom Punkt P aus jeweils eine Gerade zu einem Eckpunkt konstruiert. Die Winkel ϕ_i zwischen zwei benachbarten Geraden zu den Punkten P_i und P_{i+1} werden aufsummiert.
Ist $\sum_{i=0}^{n+1} \phi_i = 2\pi$, so liegt P im Polygon.
Ist andernfalls $\sum_{i=0}^{n+1} \phi_i = 0$, so liegt P nicht im Polygon.

7.3 Punkt im Kreis

Abbildung 7.2 zeigt einen Kreis um den Mittelpunkt K mit dem Radius r. Es gilt nun zu testen, ob der Punkt P in diesem Kreis liegt, oder nicht.
Dazu werden die beiden Ortsvektoren OK zum Kreismittelpunkt K und OP zu Punkt P konstruiert. Ist der Abstand der beiden Punkte K und P größer, als der Radius r, so liegt der Punkt P offensichtlich ausserhalb des Kreises. Es gilt dann also: $r < |OP - OK|$. Umgekehrt gilt: P liegt im Kreis, wenn $r > |OP - OK|$.

7.4 Punkt in einer Kugel

Der Fall aus Abschnitt 7.3 *Punkt im Kreis* läßt sich auf den 3-Dimensionalen Raum übertragen. Das Vorgehen ist dabei Analog. Lediglich die Vektoren haben je eine Komponente mehr.

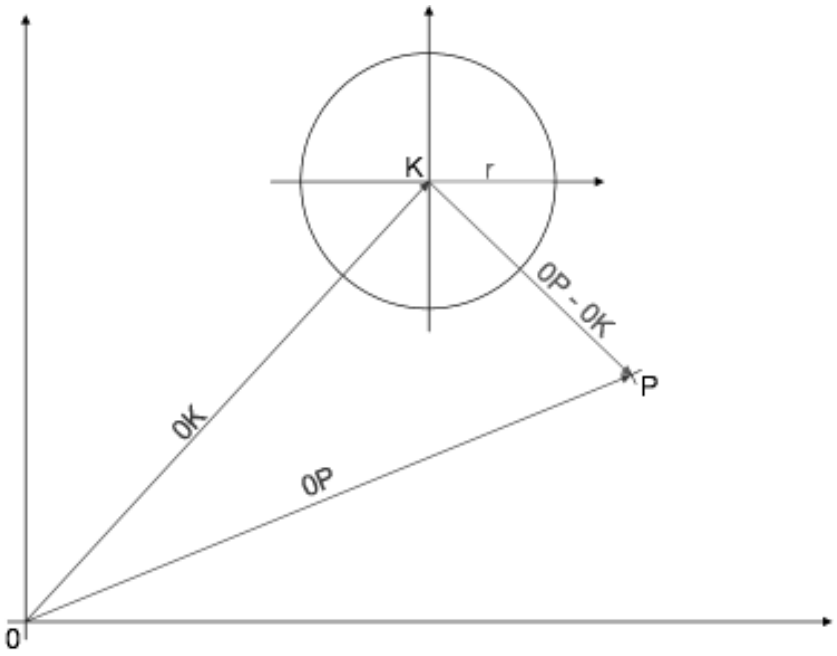


Abbildung 7.2: Befindet sich ein Punkt P im Kreis?

8 Koordinatentransformationen

Bei der Modellierung von 3-Dimensionalen Räumen existieren verschiedene Arten von Koordinaten- bzw. Bezugssystemen. Diese könne ineinander transformiert werden. In diesem Kapitel sollen die wichtigsten vorgestellt werden.

8.1 Verschiedene Arten von Koordianten- bzw. Bezugssystemen

Durch die Abeit mit verschiedenen Koordinatensystemen, wird vieles in der Verwaltung der Objekte einer Szene vereinfacht.

8.1.1 Lokale Koordinaten

Lokale Koordinaten werden auch *Objekt-*, *Modell-* bzw. *Master*-Koordinaten genannt. Sie sind *relativ* zum Objekt definiert, d.h. jedes Objekt besitzt sein eigenes Koordinatensystem. Der Ursprung liegt im Zentrum des Objektes.

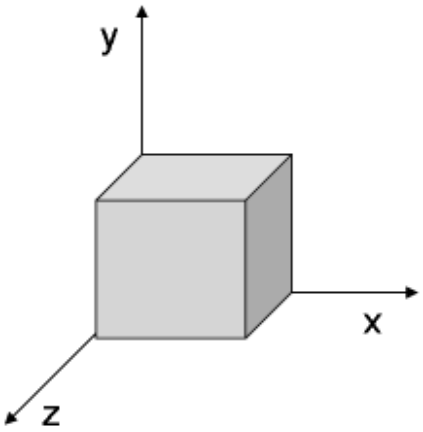


Abbildung 8.1: Beispiel für die lokalen Koordinaten eines Objektes.

8.1.2 Welt-Koordinaten

Welt-Koordinaten bilden die Wurzel für die gesamte Szenenhierarchie. Sie sind unabhängig von allen anderen und beschreiben die Anordnung und Größe der Szenenobjekte.

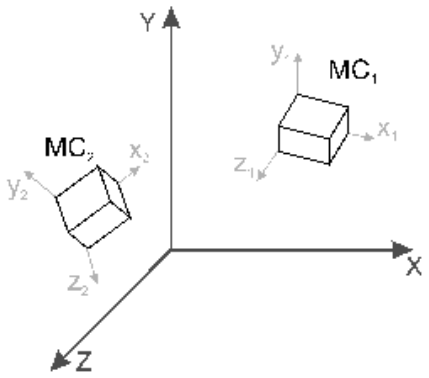


Abbildung 8.2: Beispiel für die die Komposition von zwei Objekten (mit lokalen Koordinaten) in globalen Welt-Koordinaten.

8.1.3 Kamera-Koordinaten

Kamera-Koordinaten werden auch *Augen-Koordinaten* genannt. Sie sind innerhalb der Welt definiert und stellen somit den Blickwinkel auf die zu rendernde Szene dar. Das Zentrum ist der *Augpunkt*. Die Blickrichtung fokussiert den *View Reference Point*. Dadurch wird das *View Frustum* definiert, daß auf Seite 30 in Abschnitt 8.2 beschrieben wird.

8.1.4 Bild-Koordinaten

Soll eine Szene rerendert werden, so müssen die 3D-Koordinaten auf eine 2D-Fläche projetiert werden, um sie z.B. auf einem Monitor darstellen zu können.

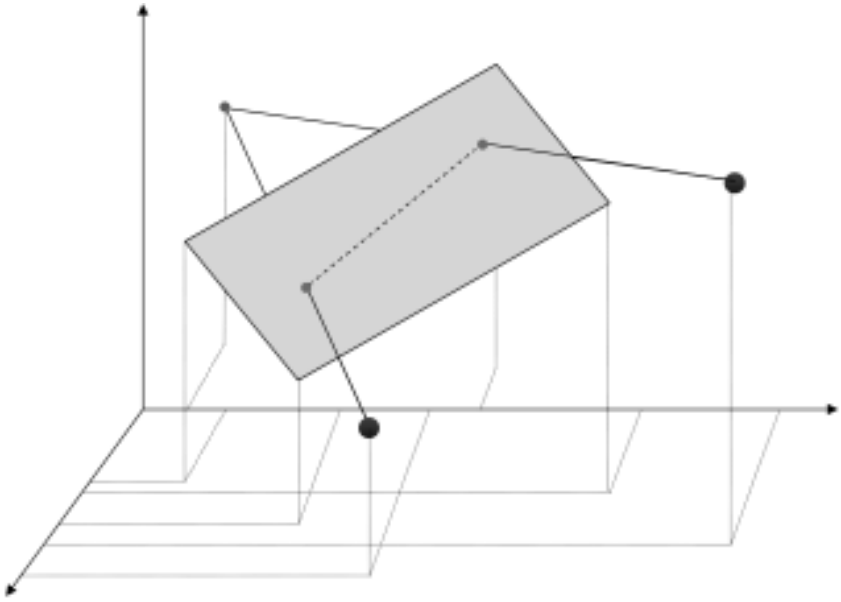


Abbildung 8.3: Beispiel für die die Projektion von zwei Punkten aus dem 3-Dimensionalen auf eine Ebene.

8.2 View Frustum

Das *View Frustum* ist der Pyramidenstumpf, der sich ergibt, wenn man ausgehend vom *view point* in die Szene sieht. Er ist in Abbildung 8.4 visualisiert. Dabei beschreibt die *view distance* den Abstand zwischen Betrachter und der *Projektionsebene*. Damit zu nahe Objekte nicht die gesamte Szene verdecken, werden alle Punkte vor der *Front-Plane* nicht mit berechnet. Sehr weit entfernte Objekte erscheinen dagegen ganz klein und können daher hinter der *Back-Plane* verworfen werden.

8.3 Koordinatentransformationen

Koordinatentransformationen verhalten sich komplementär zu den Transformationen von Objekten. Dabei bleiben die Objekte fest, aber das Koordinatensystem ändert seine Lage ung eventuell seine Form.

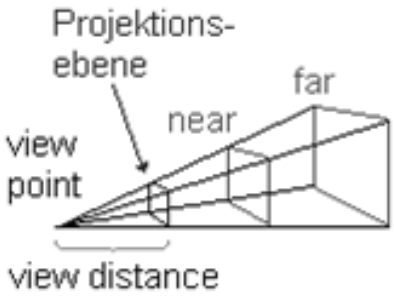


Abbildung 8.4: View Frustum mit Front-Plane (near) und Back-Plane (far).

8.3.1 Viewing-Pipeline

Die *Viewing-Pipeline* beschreibt die Schritte, die ein Punkt durchlaufen muß, der in *Lokalen Koordinaten* vorliegt, bis dieser zu *Bild-Koordinaten* transformiert wird.

Lokale Koordinaten in Welt-Koordinaten

Die Objekte liegen als Prototypen vor. Ihre Definitionspunkte sind unabhängig von späterer Größe und Position im Welt-Koordinatensystem. Durch das *Modeling* erhalten die Objekte im Welt-Koordinatensystem ihre individuelle Größe, Orientierung und Position durch Skalierung, Rotation und Translation.

1. Das Modell-Koordinatensystem liegt deckungsgleich zum Welt-Koordinatensystem.
2. Das Modell-koordinatensystem wird durch schrittweise Ausführung von Einzeltransformationen in die gewünschte rüumliche Lage gebracht.
3. Das Objekt wird im Welt-Koordinatensystem abgebildet.
4. Die Objektkoordinaten aller Objektpunnkte werden mit den Transformationsmatrizen multipliziert.

Anhand dieser Schritte wird im Beispiel in Abbildung 8.5 der Würfel gedreht.

Sieht man sich in Abbildung 8.6 die Rotation in den Einzelschritten der drei Dimensionen an, so erkennt man, daß man diese drei Einzeltransformationen $R_1 \dots R_3$ zu einer einzigen mathematisch zusammengefasst werden können.

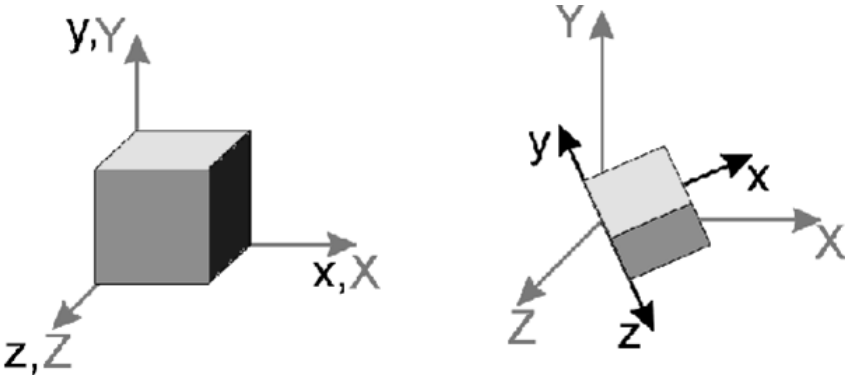


Abbildung 8.5: Beispiel für die Drehung eines Würfels (links: Ausgansposition rechts: transformierte Position)

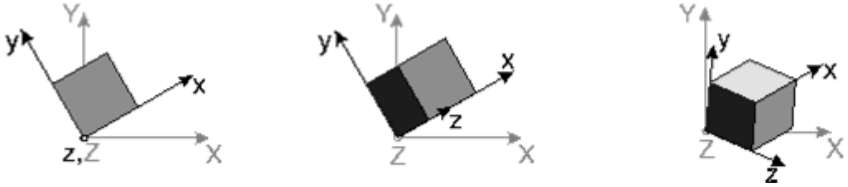


Abbildung 8.6: Beispiel für die Drehung eines Würfels entlang der einzelnen Achsen mittels der drei Einzeltransformationen $R_1 \dots R_3$.

Sei $Q(x, y, z, 1)$ der Punkt eines Objektes, so erhält man die *Welt-Koordianten* aus den *Modell-koordinaten* und den folgenden Transformationen: $(Q_{\mathrm{WC}})^T = R_3 \cdot R_2 \cdot R_1 \cdot (Q_{\mathrm{MC}})^T$.

Welt-Koordinaten in Kamera-Koordinaten

Hierbei findet ein Koordinatensystemwechsel statt, der dazu dient, daß eine Szene einmal relativ zu den *Welt-Koordinaten* definiert wird, aber trotzdem flexibel betrachtet werden kann. Die XY-Ebene ist dabei die Bildebene. Das „Auge“ liegt bei $z = VPD$.

Kamera-Koordinaten in Normalisierte Projektions-Koordinaten

Um die möglichen Persketivischen Verzerrungen bei den vorherigen Transformationen zu korrigieren, wird der Pyramidenstumpf des *View Frustum* auf einen Einheitswürfel abgebildet.

Normalisierte Projektions-Koordinaten in Bild-Koordinaten

Durch weglassen der 3. Dimensions-Komponente werden die Koordinaten auf eine Ebene abgebildet.

9 Projektionen

Bei Projektionen handelt es sich um die Abbildung eines Vektors x aus einem n-Dimensionalen Vektorraum X auf einen Vektor aus einem m-Dimensionalen Unterraum U. Im besonderen geht es uns um die Abbildungen des 3D-Raumes auf eine Ebene. Dabei handelt es sich um *planare Projektionen* (mit n=3,m=2). Abbildung 9.1 zeigt die hierarchische Gliederung der Projektionen.

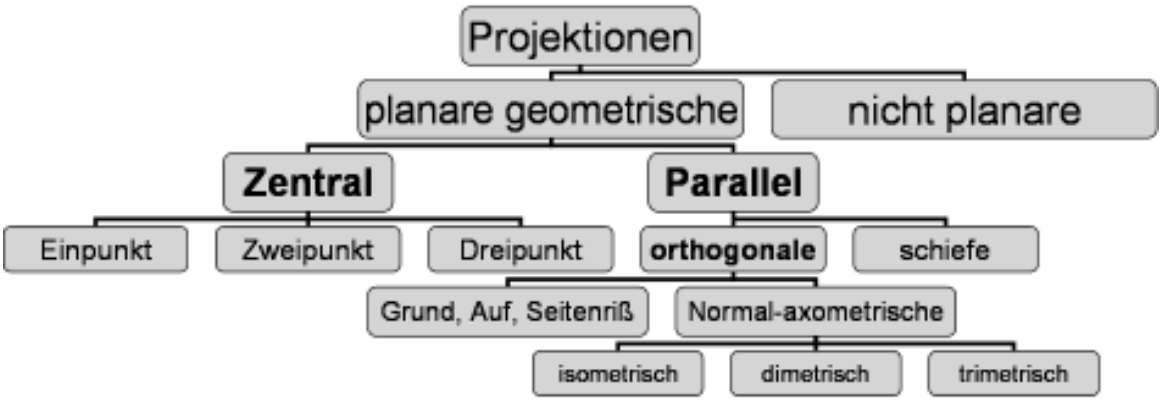


Abbildung 9.1: Hierarchie von Projektionen

9.1 Zentralprojektion

Zentralprojektionen werden oft auch *Perspektivische Projektion* genannt. Wir kennen den Augpunkt, der das Projektionszentrum Z bildet. Bei der Projektion kann es zu perspektivischen Verzerrungen kommen, da parallele Linien nicht immer parallel projeziert werden. Es entsteht ein räumlicher Eindruck durch perspektivische Verkürzung. Winkel bleiben nur erhalten, wenn die Geraden parallel zur Bildebene sind. Parallele Linien schneiden sich in der Unendlichkeit.

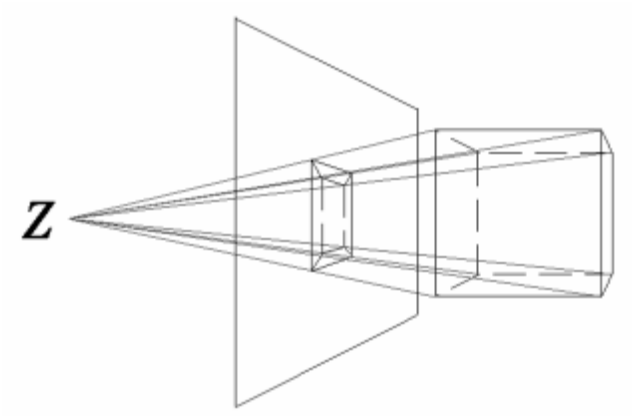
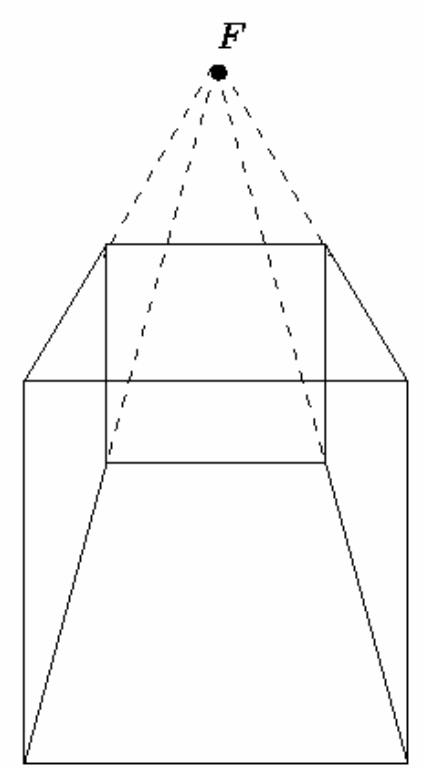


Abbildung 9.2: Beispiel für eine Zentralprojektion

9.2 Punkt-Perspektiven

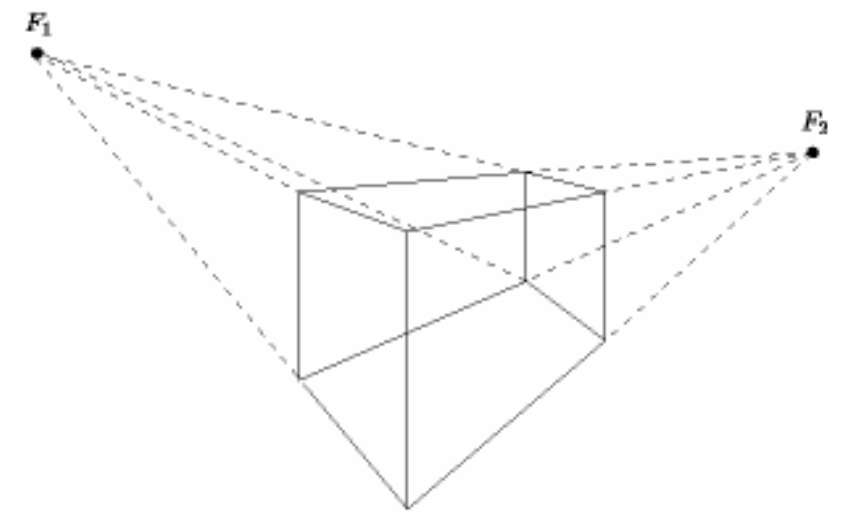
9.2.1 Ein-Punkt-Perspektive

Die *Ein-Punkt-Perspektive* besteht nur aus dem *Fluchtpunkt* F, der das Projektionszentrum darstellt.



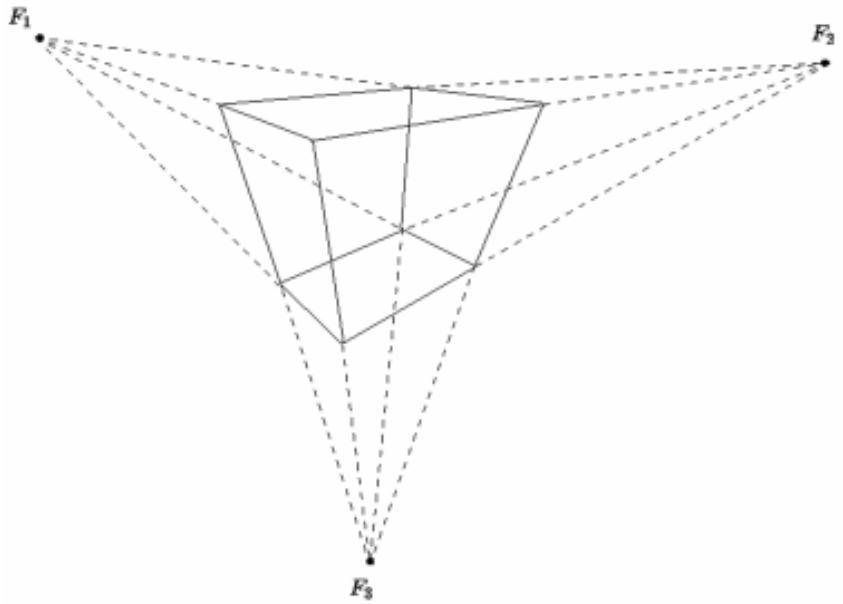
9.2.2 Zwei-Punkt-Perspektive

Die *Zwei-Punkt-Perspektive* besteht aus den beiden *Fluchtpunkten* F_1 und F_2, die in der Regel auf Augenhöhe liegen.



9.2.3 Drei-Punkt-Perspektive

Eine *Drei-Punkt-Perspektive* besteht aus den drei *Fluchtpunkten* F_1, F_2 und F_3 . Der zusätzliche Fluchtpunkt trägt nur wenig zu einer Realitätsverbesserung bei. Er wird als zusätzlicher Fluchtpunkt für die vertikalen Kanten genutzt. Dies führt zu *Frosch-* und *Vogelperspektiven*.



9.3 Parallelprojektionen

Bei *Parallelprojektionen* ist der Abstand vom *Projektionszentrum* zur *Projektionsebene* unendlich. Demzufolge können die Projektionslinien als *parallel* ange-

nommen werden. Die *Parallelprojektion* ist ein Spezialfall der *Zentralprojektion*. Sie spiegelt *exakte* Maße wider. Parallele Linien bleiben parallel. Allerdings bleiben nur Winkel erhalten, die parallel zur Projektionsebene liegen. Diese Perskektive ist aufgrund der fehlenden pespektivischen Verkürzung weniger realistisch.

9.3.1 Orthogonale Parallelprojektion

Bei der *Orthogonalen Parallelprojektion* ist die Projektionsrichtung gleich einer Normalen der Projektionsebene, d.h. die Projektionslinien stehen senkrecht zur Bildebene, wie in Abbildung 9.3 zu sehen ist.

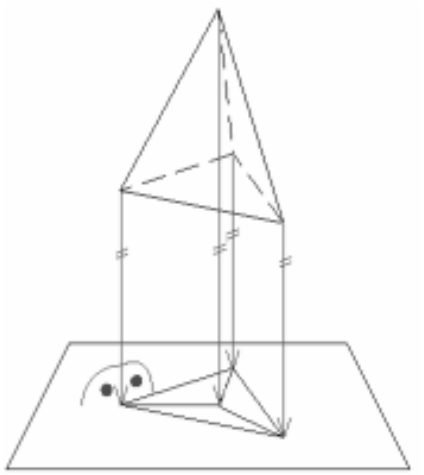


Abbildung 9.3: Beispiel für eine Orthogonale Parallelprojektion

9.3.2 Risse

Risse werden oft für technische Zeichnungen verwendet, da Entfernungen und Winkel direkt abgemesen werden können. Allerdings sind die meist schwer zu Interpretieren und nicht immer rekonstruierbar.

Aufriß

Der *Aufriß* betrachtet die XY-Ebene.

Grundriß

Der *Grundriß* betrachtet die ZY-Ebene.

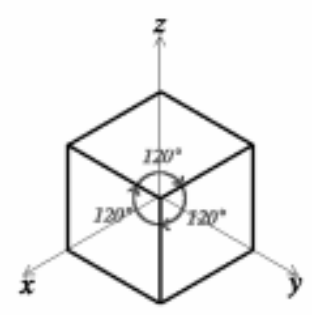
Seitenriß

Der *Seitenriß* betrachtet die XZ-Ebene.

9.3.3 Normalaxonometrische Projektionen

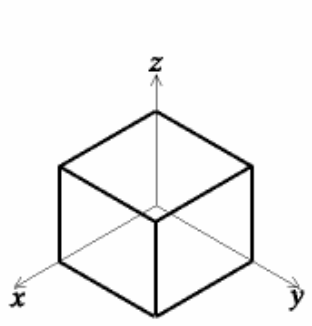
isometrisch

Bei der *Isometrischen Normalaxonometrischen Projektion* sind die Schnittpunkte der Bildebene mit den Hauptachsen alle gleich weit vom Ursprung entfernt. Alle Koordinaten werden gleich verkürzt.



dimetrisch

Bei der *Dimetrischen Normalaxonometrischen Projektion* sind *zwei* Schnittpunkte der Bildebene mit den Hauptachsen alle gleich weit vom Ursprung entfernt. Auf zwei Achsen werden die Koordinaten werden gleich verkürzt.



trimetrisch

Bei der *Trimetrischen Normalaxonometrischen Projektion* werden alle Koordinatenachsen in unterschiedlicher Entfernung vom Ursprung von der Bildebene geschnitten. Alle Koordinaten werden unterschiedlich verkürzt.

9.3.4 Schiefwinklige Parallelprojektion (Oblique)

Bei einer *Schiefwinkligen Parallelprojektion*, die auch *Oblique Projektion* genannt wird, ist die Projektionsrichtung *ungleich* der Normalen der Projektionsebene.

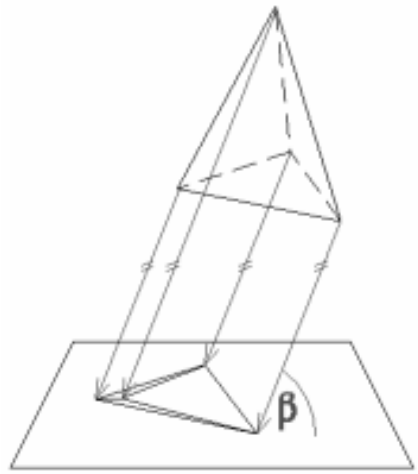


Abbildung 9.4: Beispiel einer Schiefwinkligen Parallelprojektion

Literaturverzeichnis

„Computergrafik und Geometrisches Modellieren“, Beate Brüderlin, Andreas Meier, Teubner-Verlag, 2001

„Computer Graphics - Principles and Practice“, Foley, J., van Dam, A., et al. Reading, MA, USA. Addison-Wesley Publishing Company, 1990

„Mathematische Grundlagen der Computergrafik“, P. Krug, UNI Bremen, 2003, http://info.ifi.hs-bremen.de/~krug/veranstaltungen/Mathe4/MathematischeGrundlagenderCG.pdf