Einführung in die Computergrafik

Johannes Riesterer

13. April 2016

Vorwort

Dieses Skript entstand aus einer Reihe von Vorlesung über Computergrafik, die ich sowohl an der dualen Hochschule in Stuttgart als auch an der Hochschule für Kunst Design und populäre Musik Freiburg gehalten habe. Für die Mitarbeit an diesem Skript möchte ich Alexander Berg danken.

Inhaltsverzeichnis

1 Mathematische Werkzeuge

1.1 Lineare Algebra

1.1.1 Vektoren und Matrizen

Wir wollen zunächst den Vektorraum $\mathbb{R}^n$ einführen. Hierbei ist n eigentlich immer $2, 3$ oder 4. Zunächst ist der $\mathbb{R}^n$ eine Menge, nämlich die Menge der n-dimensionalen Vektoren

$$\mathbb{R}^n := \left\{ \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} \middle| \, x_1, x_2, \ldots, x_n \in \mathbb{R} \right\} .$$

Auf dieser Menge der Vektoren definiert man die Addition

$$+ : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^n$$
$$\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} + \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} := \begin{pmatrix} x_1 + y_1 \\ x_2 + y_2 \\ \vdots \\ x_n + y_n \end{pmatrix}$$

und die sogenannte Skalarmultiplikation

$$\cdot : \mathbb{R} \times \mathbb{R}^n \to \mathbb{R}^n$$
$$\lambda \cdot \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} := \begin{pmatrix} \lambda \cdot x_1 \\ \lambda \cdot x_2 \\ \vdots \\ \lambda \cdot x_n \end{pmatrix} .$$

Das Element $\lambda \in \mathbb{R}$ nennt man auch Skalar.

Definition 1. *Der Vektorraum $\mathbb{R}^n$ ist das Tripel* $(\mathbb{R}^n, +, \cdot)$.

Beispiel 1.

$$\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \\ 4 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} + \begin{pmatrix} -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$\begin{pmatrix} 1 \\ -1 \\ -\frac{1}{2} \\ 2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ -\frac{1}{2} \\ 2 \end{pmatrix}$$

$$-1 \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ -1 \end{pmatrix}$$

$$\pi \cdot \begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} = \begin{pmatrix} \pi \\ 2\pi \\ 3\pi \\ 4\pi \end{pmatrix}$$

Bemerkung 1. *Für alle Skalare $\lambda, \mu \in \mathbb{R}$ und Vektoren $u, v \in \mathbb{R}^n$ gelten die Rechenregeln*

$$\begin{aligned} \lambda \cdot (u + v) &= \lambda \cdot u + \lambda \cdot v \\ (\lambda + \mu) \cdot u &= \lambda \cdot u + \mu \cdot u\ . \end{aligned}$$

Definition 2. *Für ein $u, v \in \mathbb{R}^n$ sind die folgenden Kurz-Notationen üblich*

$$\begin{aligned} -u &:= -1 \cdot u \\ u - v := u + (-v) &:= u + (-1 \cdot v) \end{aligned}$$

Definition 3. *Eine $n \times m$ Matrix ist ein Objekt der Form*

$$(a_{ij})_{ij} := \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & a_{2,3} & \cdots & a_{2,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{i,1} & a_{i,2} & a_{i,3} & \cdots & a_{i,m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n,1} & a_{n,2} & a_{n,3} & \cdots & a_{n,m} \end{pmatrix}$$

mit $a_{i,j} \in \mathbb{R}$ für alle $i = 1, \ldots, n$ und $j = 1, \ldots m$.

Bemerkung 2. *Ein Vektor der dimension n ist eine $n \times 1$-Matrix.*

Definition 4. *Ist $A = (a_{ij})_{ij}$ eine $n \times m$ und $B = (b_{kl})_{kl}$ eine $m \times p$ Matrix so ist das Matrizenprodukt definiert als die $n \times p$ Matrix*

$$A \cdot B := \left(\sum_{j=1}^{m} a_{ij} \cdot b_{jl} \right)_{il} .$$

Sind A und B zwei $n \times m$-Matrizen so ist ihre Summer definiert durch

$$A + B := \left(a_{ij} + b_{ij} \right)_{ij} .$$

Für ein $\lambda \in \mathbb{R}$ definieren wir die Skalarmultiplikation

$$\lambda \cdot A := \left(\lambda \cdot a_{ij} \right)_{ij} .$$

Definition 5. *Die n-dimensionale Einheitsmatrix ist definiert durch*

$$I_n := \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & 0 & \cdots & 0 & 0 \\ \vdots & & \ddots & & \vdots & \vdots \\ \vdots & & & \ddots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 \end{pmatrix} .$$

Bemerkung 3. *Sind A und B beides $n \times n$-Matrizen, so ist im Allgemeinen*

$$A \cdot B \neq B \cdot A .$$

Für die n-te Einheitsmatrix I_n gilt jedoch immer

$$A \cdot I_n = I_n \cdot A = A .$$

Definition 6. *Für eine 2×2-Matrix definieren wir die Determinante*

$$\det : M^{n \times n} \to \mathbb{R}$$
$$\det\left(\begin{pmatrix} a & b \\ c & d \end{pmatrix}\right) := ad - bc$$

Satz 1. *Für eine $n \times n$ Matrix $A = (a_{ij})_{ij}$ definieren wir die Determinante durch die Rekursionsformel*

$$det(A) = \sum_{j=1}^{n} (-1)^{i+j} a_{ij} det(A_{ij})$$

und $det(a) = a$ für eine 1×1-Matrix a. wobei A_{ij} die Matrix ist, die aus A durch Streichen der i-ten Zeile und der j-ten Spalte entsteht. Diese Definition ist unabhängig von der Wahl von i. (Entwickeln nach der $i - tenZeile$).

Satz 2. *Für alle $n \times n$-Matrizen A, B gilt*

$$\det(A \cdot B) = \det(A) \cdot \det(B)$$

.

Satz 3. *Sei A eine $n \times n$ Matrix. Dann existiert genau dann eine Matrix A^{-1} mit*

$$A \cdot A^{-1} = A^{-1} \cdot A = I_n ,$$

wenn $\det(A) \neq 0$. A^{-1} ist eindeutig bestimmt.

Bemerkung 4. *Ist v ein n-dimensionaler Vektor und A eine $m \times n$-Matrix, so ist $A \cdot v$ ein m-dimensionaler Vektor.*

Definition 7. *Ist $A := (a_{ij}))_{ij}$ eine $n \times m$-Matrix, so heißt die $m \times n$-Matrix $A^t := (a_{ji}))_{ij}$ die transponierte Matrix.*

Definition 8. *Ist insbesondere $v = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$ ein k-dimensionaler Vektor, so heisst $v^t = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix}$ der transponierte Vektor, welcher auch eine $1 \times n$-Matrix ist.*

Satz 4. *Für alle $n \times m$-Matrizen A und m-dimensionale Vektoren u, v gilt*

$$A \cdot (\lambda \cdot u + \mu \cdot v) = \lambda \cdot A \cdot u + \mu \cdot A \cdot v\ .$$

Definition 9. *Vektoren $v_1, \dots, v_k \in \mathbb{R}^n$ heißen linear unabhängig, falls für $\lambda_i \in \mathbb{R},\ i = 1, \dots, k$ mit*

$$\sum_{i=1}^{k} \lambda_i \cdot v_i = 0$$

stets $\lambda_i = 0$ folgt für alle $i = 1, \dots, k$.

Satz 5. *Die Vektoren $v_1, \dots, v_k \in \mathbb{R}^n$ sind genau dann linear abhängig, wenn man mit Hilfe des Gaussalgorithmus in der Matrix*

$$\begin{pmatrix} v_1^t \\ \text{---} \\ \vdots \\ \text{---} \\ v_k^t \end{pmatrix}$$

eine Nullzeile erzeugen kann.

Bemerkung 5. *Für $k > n$ sind $v_1, \dots, v_k \in \mathbb{R}^n$ stets linear abhängig.*

Definition 10. *Für Vektoren $v_1, \dots, v_k \in \mathbb{R}^n$ heißt die Menge*

$$span(v_1, \dots v_k) := \left\{ \sum_{i=1}^{k} \lambda_i \cdot v_i \mid \lambda_i \in \mathbb{R} \right\} \subseteq \mathbb{R}^n$$

der von ihnen aufgespannte lineare Unterraum. Diese Definition ist offensichtlich unabhängig von der Reihenfolge. Eine Menge von Vektoren $\{w_1, \dots, w_l\}$ heißt Basis von $span(v_1, \dots v_k)$, falls $w_1, \dots, w_l$ linear unabhängig sind und $span(w_1, \dots w_l) = span(v_1, \dots v_k)$ gilt. l heißt dann auch die Dimension von $span(v_1, \dots v_k)$.

Satz 6. *Ist $B := \{b_1, \dots, b_n\}$ eine Menge linear unabhängiger n-dimensionaler Vektoren, so ist*

$$span(b_1, \dots, b_n) = \mathbb{R}^n\ .$$

Wir nennen dann die geordnete *Menge B eine Basis des $\mathbb{R}^n$.*

Definition 11. *Wir bezeichnen die Basis* $E := \{e_1, \dots, e_n\}$ *des* $\mathbb{R}^n$ *mit*

$$e_i := \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow \textit{i-te Stelle}$$

als Standardbasis des $\mathbb{R}^n$.

Definition 12. *Sei* $B := \{b_1, \dots, b_n\}$ *eine Basis des* $\mathbb{R}^n$ *und* $v \in \mathbb{R}^n$. *Dann gibt es nach dem letzten Satz Skalare* $\lambda_1, \dots, \lambda_n \in \mathbb{R}$, *so dass sich* v *als Linearkombination*

$$v = \sum_{i=1}^{n} \lambda_i \cdot b_i$$

ausdrücken lässt. Schreibt man diese λ_i *wieder in einen Vektor, so erhalten wir eine Abbildung*

$$\begin{aligned} \theta_B : \mathbb{R}^n &\to \mathbb{R}^n \\ \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} &\mapsto \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix} \quad . \end{aligned}$$

Man nennt $\theta_B(v)$ *die Darstellung von* v *zur Basis* B.

Bemerkung 6. *Für die Standardbasis* E *des* $\mathbb{R}^n$ *ist* $\theta_E(v) = v$ *für alle* $v \in \mathbb{R}^n$, *also* $\theta_S = id$.

Definition 13. *Sei* $B := \{b_1, \dots, b_n\}$ *eine Basis des* $\mathbb{R}^n$ *und* $v \in \mathbb{R}^n$. *Dann definieren wir*

$$M_B = \left(b_1 \,\middle|\, b_2 \,\middle|\, \cdots \,\middle|\, b_n \right)^{-1} .$$

Satz 7. *Sei* $B := \{b_1, \dots, b_n\}$ *eine Basis des* $\mathbb{R}^n$ *und* $v \in \mathbb{R}^n$. *Dann ist*

$$\theta_B(v) = M_B \cdot v$$

Definition 14. *Seien* $B := \{b_1, \dots, b_n\}$ *und* $B' := \{b'_1, \dots, b'_n\}$ *zwei Basen des* $\mathbb{R}^n$. *Dann heißt* $M_B^{B'} := M_{B'} \cdot M_B^{-1}$ *die Basiswechselmatrix von* B *nach* B'. *Wir haben also folgende Situation:*

$$\begin{tikzcd} \mathbb{R}^n \arrow[d, "I_n"] & \mathbb{R}^n \arrow[l, "M_B^{-1}"'] \arrow[d, "M_B^{B'}"] \\ \mathbb{R}^n \arrow[r, "M_{B'}"] & \mathbb{R}^n \end{tikzcd}$$

Definition 15. *Die Abbildung*

$$\begin{aligned}
&< \cdot\, , \cdot >: \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}\\
&\left\langle \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}, \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} \right\rangle := \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}^t \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} x_1 & \cdots & x_n \end{pmatrix} \cdot \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix} = \sum_{i=1}^{n} x_i \cdot y_i
\end{aligned}$$

heißt Skalarprodukt.

Satz 8. *Für alle $u, v, w, l \in \mathbb{R}^n$ und $\lambda, \mu, \tau, \nu \mathbb{R}$ gilt*

$$\begin{aligned}
< \lambda u + \mu v, \tau w > &= \lambda\tau < u, w > + \mu\tau < v, w >\\
< \lambda u, \tau w + \nu l > &= \lambda\tau < u, w > + \lambda\nu < u, l >
\end{aligned}$$

Definition 16. *Die Abbildung*

$$\begin{aligned}
&|| \cdot || : \mathbb{R}^n \to \mathbb{R}\\
&||v|| := \sqrt{< v, v >}
\end{aligned}$$

heißt Norm.

Definition 17. *Zwei vom Nullvektor verschiedene Vektoren $u, v \in \mathbb{R}^n$ heißen orthogonal, falls $< u, v >= 0$ ist. Man sagt auch sie stehen senkrecht aufeinander und benutzt auch die Bezeichnung $u \perp v$.*

Satz 9. *Der Kosinus des Innenwinkel φ zweier Vektoren u, v lässt sich durch*

$$\cos(\varphi) = \frac{< u, v >}{||u|| \cdot ||v||}$$

Definition 18. *Ein Vektor $v \in \mathbb{R}$ heißt normal, falls $||v|| = 1$ ist. Ist $w \in \mathbb{R}^n$ ein beliebiger Vektor, so heißt $\frac{1}{||w||} w$ die Normalisierung von w.*

Definition 19. *Eine Basis $B := \{b_1, \ldots, b_n\}$ heißt Orthonormalbasis (kurz ONB), falls*

$$< b_i, b_j >= \begin{cases} 1 \text{ falls } i = j \\ 0 \text{ sonst} \end{cases}$$

gilt. Insbesondere sind alle b_i normal.

Algorithmus 1. *Seien $v_1, \ldots v_n$ linear unabhängige Vektoren. Dann lässt sich daraus durch folgenden Algorithmus eine ONB generieren:*

$$\begin{aligned}
b'_i &:= v_i - \sum_{j=1}^{i-1} < v_i, b_j > b_j\\
b_i &:= \frac{1}{||b'_i||} b'_i
\end{aligned}$$

und Rekursionsanfang $b'_1 = v_1$.

Im $\mathbb{R}^3$ gibt es eine besonders einfache Methode aus zwei Vektoren einen Vektor zu generieren, der auf den Ausgangs-Vektoren senkrecht steht.

Definition 20. *Für* $u = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \end{pmatrix}$ *und* $v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$ *heißt*

$$u \times v := \begin{pmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{pmatrix}$$

das Kreuzprodukt von u *und* v*.*

Bemerkung 7. *Es gilt*

- $< u \times v, u >=< u \times v, v >= 0$
- $u \times v = -(v \times u)$
- $u \times v = 0$ *genau dann, wenn* u *und* v *linear abhängig sind.*

Bemerkung 8. *Ist* $B := \{b_1, \ldots, b_n\}$ *eine ONB, so gilt*

$$M_B^{-1} = M_B^t$$

Definition 21. *Eine Matrix* $O \in \mathbb{M}^{n \times n}$ *heißt orthogonal, falls* $O^{-1} = O^t$ *ist.*

Satz 10. *Eine Matrix* $O \in \mathbb{M}^{n \times n}$ *ist genau dann orthogonal, falls*

$$\det(O) \in \{-1, 1\} \; .$$

Ist $\det(O) = 1$*, so nennen wir* O *eine Drehung und* $SO(n) := \{O \in \mathbb{M}^{n \times n} | \det(O) = 1\}$ *die Drehgruppe (oder auch spezielle orthogonale Gruppe).*

Satz 11. *Sei* $O \in \mathbb{M}^{n \times n}$ *eine orthogonale Matrix, dann gilt für alles* $v, w \in \mathbb{R}^n$

$$< O \cdot v \; , \; O \cdot w >=< v \; , \; w >$$

und somit insbesondere

$$||O \cdot v|| = ||v|| \; .$$

Definition 22. *Eine* 2×2*-Drehmatrix ist eine Matrix der Form*

$$\begin{pmatrix} \cos(\varphi) & \pm \sin(\varphi) \\ \mp \sin(\varphi) & \cos(\varphi) \end{pmatrix}$$

für ein $\varphi \in [0, 2\pi]$*.*

Bemerkung 9. *Eine* 2×2*-Drehmatrix ist eine orthogonale Matrix.*

Definition 23. *Eine elementare* 3×3*-Drehmatrix ist eine Matrix der Form*

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\varphi) & \pm\sin(\varphi) \\ 0 & \mp\sin(\varphi) & \cos(\varphi) \end{pmatrix}, \quad \begin{pmatrix} \cos(\varphi) & 0 & \pm\sin(\varphi) \\ 0 & 1 & 0 \\ \mp\sin(\varphi) & 0 & \cos(\varphi) \end{pmatrix}, \quad \begin{pmatrix} \cos(\varphi) & \pm\sin(\varphi) & 0 \\ \mp\sin(\varphi) & \cos(\varphi) & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

für ein $\varphi \in [0, 2\pi]$.

Satz 12. *Jede Drehung* $O \in SO(3)$ *lässt sich zerlegen in ein Produkt*

$$O = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\phi) & \pm\sin(\phi) \\ 0 & \mp\sin(\phi) & \cos(\phi) \end{pmatrix} \cdot \begin{pmatrix} \cos(\psi) & 0 & \sin(\psi) \\ 0 & 1 & 0 \\ -\sin(\psi) & 0 & \cos(\psi) \end{pmatrix} \cdot \begin{pmatrix} \cos(\xi) & \sin(\xi) & 0 \\ -\sin(\xi) & \cos(\xi) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Die Winkel ϕ, ψ, ξ *heißen Eulerwinkel.*

Bemerkung 10. *Die Zerlegung* $O \in SO(3)$ *einer Drehung in obiges Produkt ist nicht eindeutig. Ein anschauliches Beispiel dafür liefert der sogenannte "Gimbal lock".*

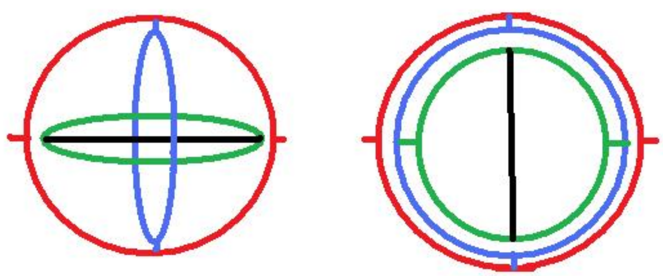


Abbildung 1: Kardansche Aufhängung und Gimbal lock

Bemerkung 11. *Man kann bei der Produktzerlegung auch andere elementare Drehmatratzen (elementare Drehachsen) wählen, wobei eine spezielle Wahl zu den in der Luft und Raumfahrt verwendeten "Roll, Nick, Gier"Winkeln führen.*

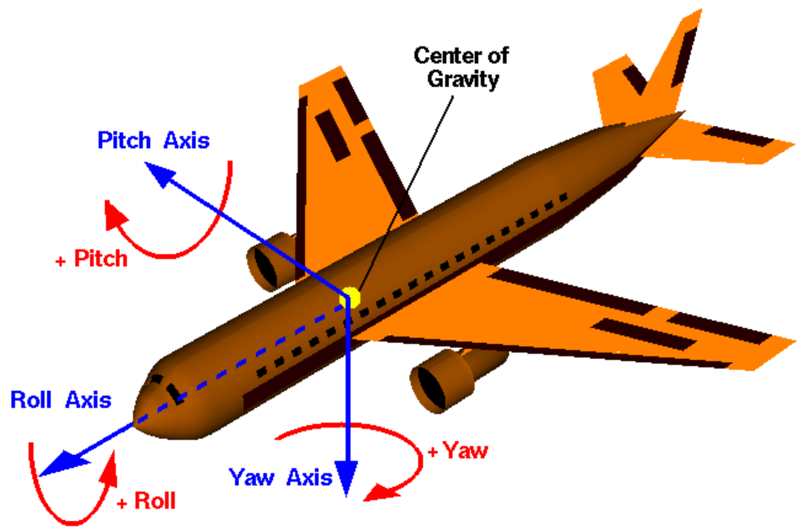


Abbildung 2: Roll, Nick und Gier Winkel

1.2 Affiner Raum und affine Abbildungen

Der Affine Raum $\mathbb{A}^n$ ist ein Tupel

$$\big(\mathbb{R}^n, (\mathbb{R}^n, +, \cdot)\big)$$

zusammen mit den Abbildung

$$\begin{aligned} \overline{} : \mathbb{R}^n \times \mathbb{R}^n &\to (\mathbb{R}^n, +, \cdot) \\ \overline{PQ} &:= Q - P \end{aligned}$$

und

$$\begin{aligned} + : \mathbb{R}^n \times (\mathbb{R}^n, +, \cdot) &\to \mathbb{R}^n \\ \begin{pmatrix} P_1 \\ \vdots \\ P_n \end{pmatrix} + \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} &:= \begin{pmatrix} P_1 + v_1 \\ \vdots \\ P_n + v_n \end{pmatrix} . \end{aligned}$$

Die Elemente (Vektoren) aus $\mathbb{R}^n$ nennt man auch Punkte in Abgrenzung zu den Vektoren aus $(\mathbb{R}^n, +, \cdot)$. Für Punkte $P, Q \in \mathbb{R}^n$ ist also $\overline{PQ}$ ein Vektor, auch Verbindungsvektor genannt.

Definition 24. *Ist $B := \{b_1, \ldots, b_n\}$ eine Basis des Vektorraums $(\mathbb{R}^n, +, \cdot)$ und $P \in \mathbb{A}$ ein Punkt, so nennen wir das Tupel (P, B) eine affine Basis. Für jeden Punkt Q gibt es dann also Skalare $\lambda_1, \ldots, \lambda_n$ mit*

$$Q = P + \sum_{i=1}^{n} \lambda_i \cdot b_i \, .$$

Der Punkt $\begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix}$ *heißt die Darstellung von* Q *bezüglich der affinen Basis* (P, B)*.*

Definition 25. *Abbildungen der Form*

$$\begin{aligned} \phi : \mathbb{A}^n &\to \mathbb{A}^n \\ \phi(P) &:= A \cdot P + t \end{aligned}$$

mit $A \in M^{n \times n}$ *und* $t \in (\mathbb{R}^n, +, \cdot)$ *heißen affine Abbildungen. Insbesondere heißt eine affine Abbildung mit* $A = I_n$ *und* $t \neq 0$ *Translation.*

Bemerkung 12. *Eine Affine Abbildung*

$$\begin{aligned} \phi : \mathbb{A}^n &\to \mathbb{A}^n \\ \phi(P) &:= A \cdot P + t \end{aligned}$$

ist genau dann invertierbar, falls $\det(A) \neq 0$ *ist und die Inverse Abbildung ist dann*

$$\begin{aligned} \phi^{-1} : \mathbb{A}^n &\to \mathbb{A}^n \\ \phi^{-1}(P) &:= A^{-1} \cdot P - A^{-1} \cdot t \ . \end{aligned}$$

Definition 26. *Sind* $(P, B := \{b_1, \ldots, b_n\})$ *und* $(P', B' := \{b'_1, \ldots, b'_n\})$ *zwei affine Basen und definieren wir die Abbildung*

$$\begin{aligned} \theta_{(P,B)} : \mathbb{A}^n &\to \mathbb{A}^n \\ \theta_{(P,B)}(Q) &:= M_B \cdot Q - M_B \cdot P \ , \end{aligned}$$

so erhalten wir analog zu der Situation in Vektorräumen

$$\begin{tikzcd} \mathbb{A}^n \arrow[d, "id"'] & \mathbb{A}^n \arrow[l, "\theta^{-1}_{(P,B)}"'] \arrow[d, "\theta^{(P',B')}_{(P,B)}"] \\ \mathbb{A}^n \arrow[r, "\theta_{(P',B')}"] & \mathbb{A}^n \end{tikzcd}$$

mit $\theta^{(P',B')}_{(P,B)}(Q) := \theta_{(P',B')}\left(\theta^{-1}_{(P,B)}(Q)\right)$*.*

Definition 27. *Der Abstand von* $P, Q \in \mathbb{A}$ *ist definiert durch*

$$\begin{aligned} d : \mathbb{A}^n \times \mathbb{A}^n &\to \mathbb{R}^n \\ d(P, Q) &:= ||\overline{PQ}|| \ . \end{aligned}$$

1.2.1 Homogene Koordinaten und Projektionen

Definition 28. *Der projektive Raum ist definiert als*

$$\mathbb{P}^n := \mathbb{R}^{n+1} - \{0\}/\sim$$
$$v \sim w \Leftrightarrow v = \lambda w \text{ für ein } \lambda \neq 0 \in \mathbb{R}\,.$$

Wir haben die Abbildung

$$\mathbb{A}^n \to \mathbb{P}^n$$
$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \end{pmatrix} \mapsto \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \\ 1 \end{pmatrix}$$

und nennen das Bild eines Punktes unter dieser Abbildung die homogenen Koordinaten. Auf der Menge der homogenen Koordinaten haben wir die Umkehrabbildung

$$\mathbb{P}^n - \left\{ \begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \\ p_{n+1} \end{pmatrix} \middle| \, p_{n+1} = 0 \right\} \to \mathbb{A}^n$$
$$\begin{pmatrix} p_1 \\ p_2 \\ \vdots \\ p_n \\ p_{n+1} \end{pmatrix} \mapsto \begin{pmatrix} \frac{p_1}{p_{n+1}} \\ \frac{p_2}{p_{n+1}} \\ \vdots \\ \frac{p_n}{p_{n+1}} \end{pmatrix}\,.$$

Die Matrizenmultiplikation

$$\mathbb{R}^{n+1} \to \mathbb{R}^{n+1}$$
$$v \mapsto A \cdot v$$

setzt sich wegen der Eigenschaft $A \cdot (\lambda v) = \lambda A \cdot v$ zu einer Abbildung

$$\mathbb{P}^n \to \mathbb{P}^n$$
$$p \mapsto A \cdot p$$

fort. Wir können damit und mit der Definition der Matrix-Vektor-Multiplikation eine affine Abbildung

$$\phi : \mathbb{R}^n \to \mathbb{R}^n$$
$$\phi(v) := A \cdot v + t$$

in homogenen Koordinate ausdrücken durch eine Matrizenmultiplikation

$$\begin{pmatrix} v \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v \\ 1 \end{pmatrix} .$$

Definition 29. *Die Abbildung*

$$persp_{xy} : \mathbb{A}^3 \to \mathbb{A}^2$$
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} \frac{X}{\frac{Z}{d}+1} \\ \frac{Y}{\frac{Z}{d}+1} \end{pmatrix}$$

die Zentralprojektion auf die $X - Y$-Ebene mit Augendistanz d.

Bemerkung 13. *Die Zentralprojektion auf die $X - Y$-Ebene mit Augendistanz d lässt sich nicht durch Multiplikation mit einer 2×3-Matrix realisieren.*

Definieren wir die Matrix

$$K_{persp_{xy}} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{d} & 1 \end{pmatrix}$$

und

$$K_{orth_{xy}} := \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} ,$$

so können wir die Zentralprojektion auf die $X - Y$-Ebene mit Augendistanz d durch die Hintereinanderausführung folgender Abbildungen darstellen:

$$persp_{xy} : \mathbb{R}^3 \to \mathbb{A}^3 \to \mathbb{A}^3 \to \mathbb{A}^2 \to \mathbb{R}^2$$
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \mapsto K_{persp_{xy}} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ z \\ \frac{z}{d}+1 \end{pmatrix}$$
$$\mapsto K_{orth_{xy}} \cdot \begin{pmatrix} x \\ y \\ z \\ \frac{z}{d}+1 \end{pmatrix} = \begin{pmatrix} x \\ y \\ \frac{z}{d}+1 \end{pmatrix} \mapsto \begin{pmatrix} \frac{x}{\frac{z}{d}+1} \\ \frac{y}{\frac{z}{d}+1} \end{pmatrix}$$

2 Kurven, Flächen und Netze

Die rechnergestützte Beschreibung von Kurven und Flächen ist ein wichtiges Gebiet der Computergrafik. Es ermöglicht die Berechnung geometrischer und physikalischer Eigenschaften von Körpern so wie deren graphische Darstellung.

Eine zentrale Rolle spielen hierbei geometrische Objekte, die durch Punkte, Kanten und einfache Flächen, wie zum Beispiel Dreiecke oder Vierecke, beschrieben werden können. Man spricht dann auch entsprechend von Dreicks- bzw. Vierecksnetzen. Häufig sind dabei nicht alle möglichen Konstruktionen zulässig und es hat sich ein Struktur herausgebildet, die für die Computergrafik besonders geeignet ist. Zum einen, weil seine grafische Darstellung besonders schnell und einfach möglich ist und zum anderen, weil viele Berechnungen bestimmte Voraussetzungen benötigen, man denke da zum Beispiel an die Oberfläche oder das Volumen eines Körpers. Von Bedeutung sind in diesem Kontext auch geeignete Datenstrukturen, mit deren Hilfe sich diese Strukturen und gängige Berechnungen effizient verarbeiten lassen. Ein weiterer wichtiger Aspekt ist das generieren und modellieren solcher Strukturen. Hierbei treten häufig sogenannte Interpolationsprobleme auf. Sie entstehen aus dem Wunsch heraus, Kurven und Flächen nur durch die Angabe von Punkten zu generieren.

2.1 Polygone, Netze und Elemente der diskreten Geometrie

Definition 30. *Ein geschlossenes Polygon ist ein Paar* $P := (E_P, K_P)$*, wobei*

$$E_P := \{e_0, \dots, e_k\} \subset \mathbb{A}^3,$$

eine geordnete Menge von paarweise verschiedenen Punkten, die auch Ecken genannt werden, und

$$K_P := \Big\{(e_0, e_1) \dots, (e_{k-1}, e_k), (e_k, e_0)\Big\} \subset \mathbb{A}^3 \times \mathbb{A}^3$$

die zugehörige Menge der gerichteten Kanten ist. Ist $(e_{l-1}, e_l) \in K_p$ *eine gerichtete Kante, so bezeichnen wir mit*

$$|(e_{l-1}, e_l)| := \big\{e_{l-1} + t \cdot \overline{e_{l-1}e_l} \;\big|\; t \in [0,1]\big\}$$

ihre geometrische Realisierung oder einfach Kante und mit

$$|P| := \bigcup_{(e_{l-1}, e_l) \in K_P} |(e_{l-1}, e_l)|$$

die geometrische Realisierung des Polygons oder geometrisches Polygon.

Ein (geschlossenes) Polygon heißt ***einfach****, falls der Schnitt zweier geometrischer Kanten entweder leer oder genau ein Punkt aus E ist. Es heißt* ***planar****, falls alle Punkte* $e \in E_P$ *in einer Ebene liegen.*

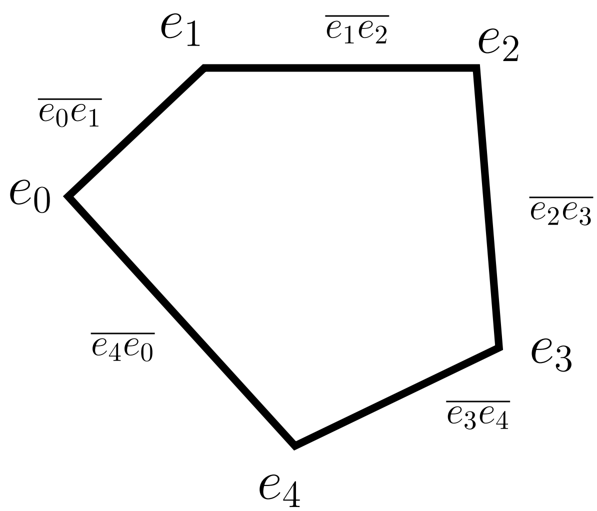


Abbildung 3: Geschlossenes Polygon

Definition 31. *Wir nennen ein geschlossenes Polygon P' eine* ***Nummerrierung*** *von P, falls $E_{P'} = E_P$ (als Mengen) und $|P'| = |P|$ ist.*

Wie wir gleich sehen werden, definiert die Reihenfolge der Punkte eines einfachen, geschlossenen, planaren Polygons eine Orientierung. Um diesen Sachverhalt präzise definieren zu können, benötigen wir einen Satz, der auf den ersten Blick als selbstverständlich erscheint aber rein mathematisch betrachtet tatsächlich schwer zu beweisen ist. Es handelt sich um den Jordanschen Kurvensatz:

Satz 13. *Sei P ein einfaches, geschlossenes, planares Polygon und U die Ebene, in der alle seine Punkte liegen. Dann unterteilt die geometrische Realisierung $|P|$ die Ebene U in genau zwei Gebiete, ein beschränktes, das das Innere des Polygons genannt wird, und ein unbeschränkte, das das Äußere des Polygons genannt wird.*

Definition 32. *Wir bezeichnen das Innere eines Polygons P mit $|\mathring{P}|$.*

Mit Hilfe des Jordanschen Kurvensatzes können wir nun sagen, was die Durchlaufrichtung einer Kante ist und schließlich ob ein Polygon mit oder gegen den Uhrzeigersinn orientiert ist.

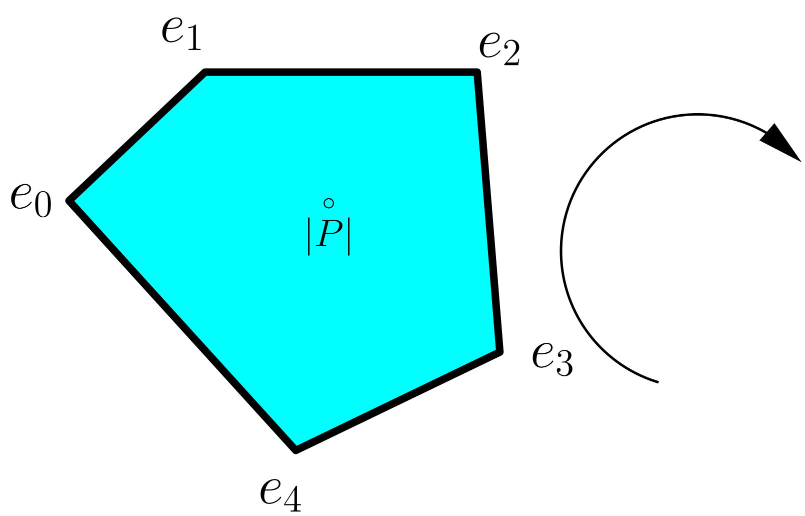
Definition 33. *Sei P ein einfaches, geschlossenes, planares Polygon. Eine gerichtete Kante $(e_{l-1}, e_l) \in E_P$ hat* ***positive Durchlaufrichtung****, falls beim Durchlaufen der Kante $|(e_{l-1}, e_l)|$ von e_{l-1} nach e_l das innere des Polygons stets rechts von der Kante liegt und* ***negative Durchlaufrichtung****, falls es stets links von ihr liegt.*

Definition 34. *Ein einfaches, geschlossenes, planares Polygon P heißt im Uhrzeigersinn orientiert, falls eine und damit alle gerichteten Kanten positive*

Durchlaufrichtung haben und entsprechend gegen den Uhrzeigersinn orientiert bei negativer Durchlaufrichtung.

$$P = \{e_0, e_1, e_2, e_3, e_4\}$$

$$K_P = \{(e_0, e_1), (e_1, e_2), (e_2, e_3), (e_3, e_4), (e_4, e_0)\}$$



$$P' = \{e_0, e_4, e_3, e_2, e_1\}$$

$$K_{P'} = \{(e_0, e_4), (e_4, e_3), (e_3, e_2), (e_2, e_1), (e_1, e_0)\}$$

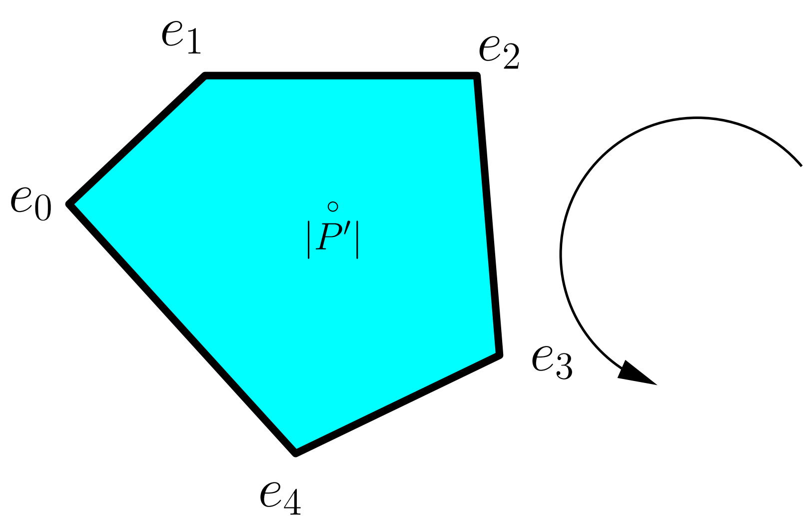


Abbildung 4: Gerichtete Polygone

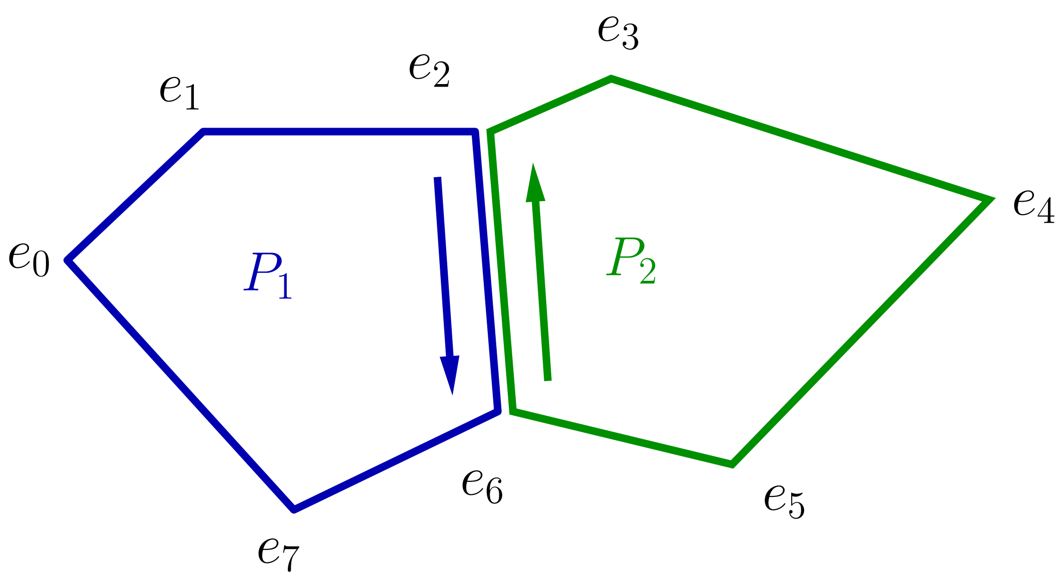
Netze bestehen nun im wesentlichen aus an den Kanten zusammengeklebten, einfachen, geschlossenen, planaren Polygonen.

Definition 35. *Sei $N := \bigcup P_i$ die Vereinigung endlich vieler, einfacher, geschlossener, planarer Polygone, $E_N := \bigcup E_{P_i}$ und $K_N := \bigcup K_{P_i}$ die Vereinigung der Ecken beziehungsweise der Kantenmengen. Analog zu der Definition eines Polygons definieren wir die geometrischen Realisierungen $|K_N| := \bigcup |P_i|$ und $|N| := \bigcup |\overset{\circ}{P}_i| \bigcup |K_N|$*

- *Dann heißt N **Netz**, falls Der Schnitt zweier Polygone entweder leer, ein Punkt in E_N oder eine Kante in K_N ist.*
- *Die Menge aller Kanten, die nur zu einem Polygon gehören, heißt Rand von N und wird mit ∂N bezeichnet.*
- *Ein Netz heißt **geschlossen**, falls es keinen Rand gibt, in Zeichen $\partial N = \emptyset$.*
- *Die Polygone des Netzes werden auch als Facetten bezeichnet.*

Durch die Orientierung von Polygonen können wir nun den Begriff der Orientierung und der Orientierbarkeit eines Netzes einführen.

Definition 36. *Ein Netz N heißt orientierbar, falls man alle seine Polygone so nummerieren kann, dass jede gemeinsame Kante zweier Polygone jeweils entgegengesetzt gerichtet ist.*



$$P_1 = \{e_0, e_1, e_2, e_6, e_7\}$$

$$P_2 = \{e_2, e_3, e_4, e_5, e_6\}$$

Abbildung 5: Orientierbare Fläche

Bemerkung 14. *Ist eine Fläche orientierbar und nummeriert man die Polygone so, dass benachbarte Kanten entsprechend der Definition immer entgegengesetzt gerichtet sind, so sind entweder alle seine Polygone im oder alle seine Polygone gegen den Uhrzeigersinn orientiert.*

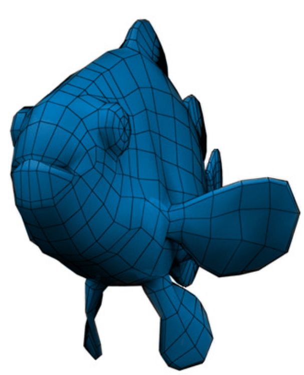


Abbildung 6: Ein geschlossenes, orientierbares Netz. Quelle:CGAL

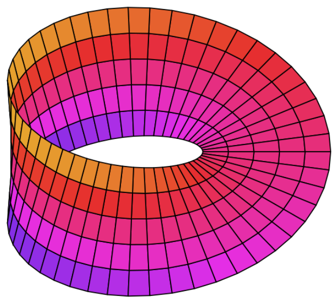


Abbildung 7: Das sogenannte Möbiusband ist hingegen nicht orientierbar. Es ist nicht geschlossen, sondern hat einen zusammenhängenden Rand. Quelle:Wikipedia

Definition 37. *Eine Orientierung ist die Wahl einer Klasse von Nummerierungen, so dass alle Polygone entweder im oder gegen den Uhrzeigersinn orientiert sind.*

Bemerkung 15. *Ein Netz hat entweder genau zwei oder keine Orientierung.*

Definition 38. *Sei N ein orientierbares Netz mit einer Orientierung so gewählt, dass alle Polygone gegen den Uhrzeigersinn orientiert sind. Wählt man für jedes Polygon $P_i \in N$ drei benachbarte Ecken $e_{j-1}^{P_i}, e_j^{P_i}$ und $e_{j+1}^{P_i}$, so erhalten wir mit $n_{P_i} = e_j^{P_i} e_{j+1}^{P_i} \times e_j^{P_i} e_{j-1}^{P_i}$ einen Vektor, der Senkrecht auf der Ebene steht in der das Polygon liegt. Wir nennen die Menge $\{P_0, \dots, P_n\}$ das äußere Normalenfeld des Netzes.*

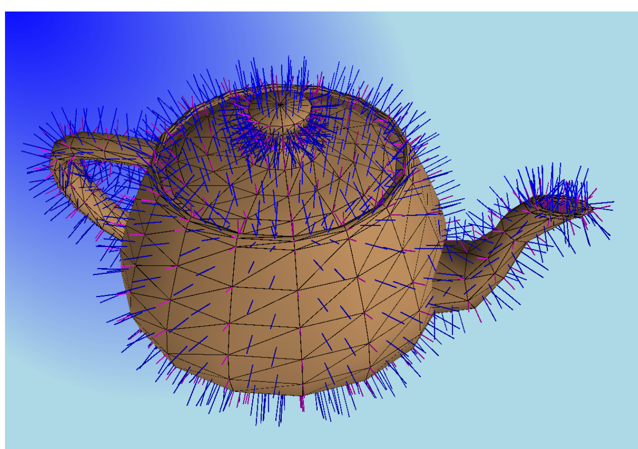


Abbildung 8: Das äußere Normalenfeld

Eine wichtige Größe für ein Netz ist die sogenannte Eulercharakteristik, welche in direkter Beziehung zum sogenannten Geschlecht eines Netzes steht.

Definition 39. *$N := \bigcup P_i$ ein Netz. Dann ist die Eulercharakteristik definiert als*

$$\chi(N) := \#(E_N) - \#(|K_N|) + \#(N)$$

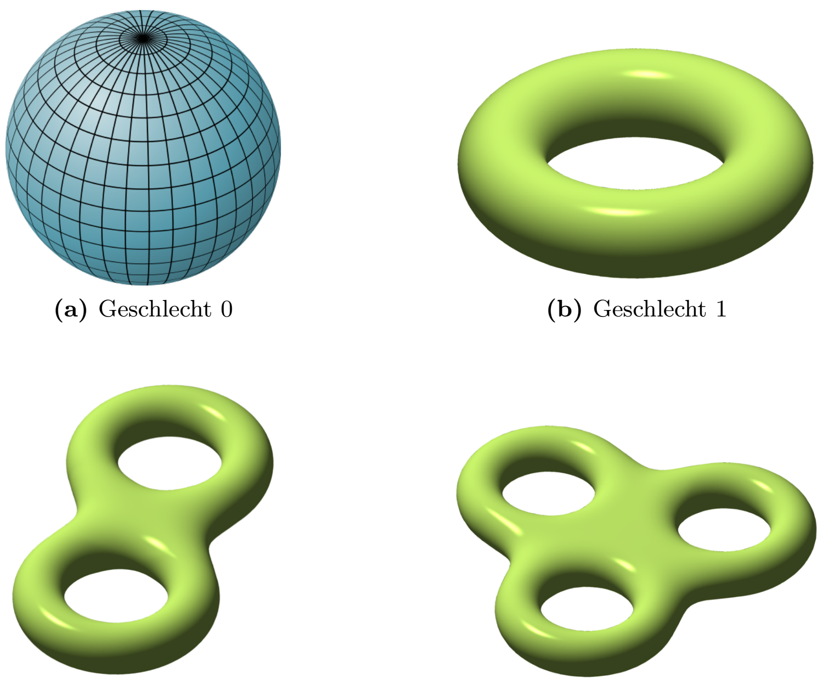
Sie ist also die Anzahl der Punkte, minus die Anzahl der geometrischen Kanten, plus die Anzahl der Facetten.

Das Geschlecht eines geschlossenen Netzes ist anschaulich gesprochen die Anzahl an Löchern, die durch das geometrische Netz hindurch gehen. Dies lässt sich mit einigem Aufwand mathematisch präzise definieren, jedoch für unsere Zwecke reicht eine Definition durch Beispiele.

Definition 40. *Sei N ein geschlossenes Netz. Dann bezeichnen wir mit*

$$g(N) := \textbf{\textit{Anzahl der Löcher in }} |N|$$

Beispiele:



(a) Geschlecht 0

(b) Geschlecht 1

(c) Geschlecht 2

(d) Geschlecht 3

Abbildung 9: Geschlossene Netze der Geschlechter 0 bis 3. Quelle:Wikipedia

Der Zusammenhang zwischen der Eulercharacteristik und dem Geschlecht wurde im allgemeinen Fall von dem Mathematiker Henri Poincare und vorher in einem Spezialfall von Leonhard Euler bewiesen.

Satz 14. *Für ein geschlossenes Netz N gilt $\chi(N) = 2 - 2g(N)$.*

Bemerkung 16. *Nach dem Satz lässt sich das Geschlecht und somit die Anzahl der Löcher via $g(N) = \frac{2-\chi(N)}{2}$ berechnen.*

2.2 Datenstrukturen

Definition 41. *Eine Liste ist eine Datenstruktur, in der man Objekte unter Beachtung der Reihenfolge speichern kann. Für eine Liste von Objekten $O_1, \ \ldots, \ O_n$ verwenden wir die Notation*

$$E = (O_1, \ \ldots, \ O_n)$$

Jedes Element kennt seinen Vorgänger und Nachfolger und man kann direkt auf das i-te Element zugreifen:

$$\begin{array}{ccc} E[i] & := & O_i \\ prev(E[i]) & := & O_{i-1} \\ next(E[i]) & := & O_{i+1} \end{array}$$

Dabei bezeichnet $prev(E[i])$ den Vorgänger und $next(E[i])$ den Nachfolger, wobei

$$\forall l \in \mathbb{N} : \{E[l] = NULL \mid (l > n) \vee (l < 0)\}$$

i wird auch als Index des i-ten Listenelements bezeichnet.

Definition 42 (Eckenliste)**.** *In einer Liste $E = (e_1, e_2, \ldots, e_n)$ werden Referenzen auf die Ecken in Form von Indizes der Eckenliste gespeichert. Eine Referenz im $\mathbb{R}^3$ könnte so aussehen:*

$$e_k = \begin{pmatrix} x_k \\ y_k \\ z_k \end{pmatrix}$$

Die i-te Facette wird als Liste $F_i = \big(i_1, \ \ldots, \ i_l\big)$ von Referenzen auf die Eckenliste gespeichert. Die k-te Ecke der i-ten Facette kann so z.B. mit $E[F_i[k]]$ referenziert werden.

Vor- und Nachteile:

+ Einfach zu implementieren
+ Orientierung kann durch Konvention gespeichert werden
- Nachbarschaftsbeziehungen nicht enthalten / schwierig zu berechnen
- Die Zugehörigkeit einer Ecke zu einer Kante muss immer berechnet werden
- Point-in-Mesh, Abstandsberechnungen, Schnittprobleme und ähnliches fast unmöglich zu berechnen

Definition 43 (Kantenliste)**.** *In einer Liste $E = (e_1,\ e_2,\ \ldots,\ e_n)$ werden Referenzen auf die Ecken gespeichert. In einer Liste K werden die Kanten in einer zwei Elementen Liste von Indizes auf die Eckenliste abgespeichert:*

$$K = \big((i_{e_1}, i_{e_n}),\ \ldots,\ (i_{e_1}, i_{e_m})\big)$$

Die k-te Facette wird als Liste von Indizes auf die Kantenliste gespeichert:

$$F_k = \big(j_1, \ldots, j_l\big)$$

In einer Liste M werden die Indizes auf Facetten in einer zwei Elementen Liste abgespeichert, die die entsprechende Kante in der Kantenliste als Kante haben:

$$M = \big((i_1, i_2),\ \ldots,\ (l_1, l_2)\big)$$

$M[i] \in M$ sind also zwei Facetten, die $K[i] \in K$ als Kante haben. Der erste Eintrag der Facette befindet sich links und der zweite rechts von der Kante. Ist die Kante eine Randkante, so wird der andere Wert auf -1 gesetzt.

Vor- und Nachteile:

+ Nachbarschaftsbeziehungen der Polygone werden gespeichert

- Orientierung der Kanten geht verloren (schwierig zu speichern)

Definition 44 (Halfedge)**.** *In einer Liste $E = (e_1,\ e_2,\ \ldots,\ e_n)$ werden Referenzen auf die Ecken gespeichert. Es wird eine Datenstruktur namens Halfedge eingeführt. Eine Halfedge e besitzt*[1]*:*

- *Indizes auf den Anfangs- und Endpunkt (i_b und i_e)*
- *Referenzen auf die gegenüberliegende Halfedge $twin(e)$*
  (Bei dieser sind die Anfangs- und Endpunkte vertauscht.)
- *Referenzen auf die vorangehende $prev(e)$ und die folgende $next(e)$ Halfedge*
- *Referenz auf die Facette, die sie berandet*

Die unendliche oder äußere Zelle wird wieder mit -1 bezeichnet. Eine Facette ist dann eine Liste mit Referenzen von Halfedges, beziehungsweise reicht auch die Referenz auf eine Halfedge.

Vor- und Nachteile:

+ Alle kombinatorischen Daten werden effizient gespeichert

+ Auch die Orientierung der Kanten bleibt erhalten

- Die Implementierung ist komplex

- Overhead durch große Datenmengen

- Nur für orientierbare Netze nutz-

[1]Siehe auch Abbildung 10 auf der nächsten Seite

bar

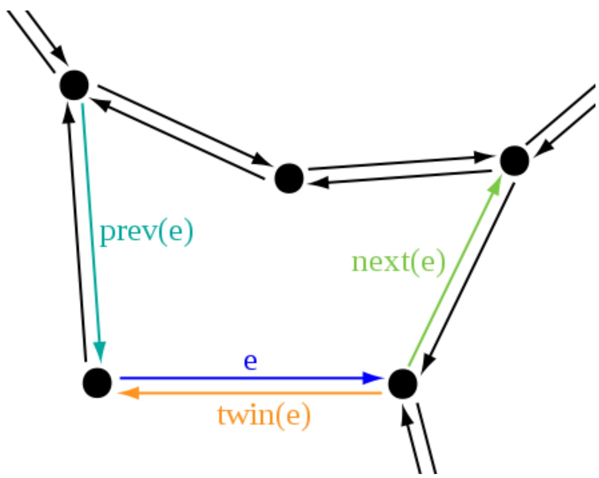


Abbildung 10: Halfedge. Quelle:Wikipedia

2.2.1 Subdivision

2.3 Modellierung

2.3.1 Parameterdarstellungen

Definition 45. *Eine Kurve ist eine Abbildung*

$$c : I \subset \mathbb{R} \to \mathbb{R}^3$$

$$c(t) := \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$$

bei der die Funktionen $x, y, z : I \to \mathbb{R}$ stetig sind. Sie heisst differenzierbar, falls x, y, z differenzierbar sind und die Ableitung ist dann definiert als

$$c' : I \subset \mathbb{R} \to \mathbb{R}^3$$

$$c'(t) := \begin{pmatrix} x'(t) \\ y'(t) \\ z'(t) \end{pmatrix} .$$

Beispiel 2. *Die Kurve $c : [0, 2\pi] \to \mathbb{R}^3$, $c(t) := \begin{pmatrix} r\cos(t) \\ r\sin(t) \\ 0 \end{pmatrix}$ beschreibt einen Kreis mit Radius r, der in der XY-Ebene liegt. Die Ableitung ist*

$$c'(t) = \begin{pmatrix} (r\cos(t))' \\ (r\sin(t))' \\ 0 \end{pmatrix} = \begin{pmatrix} -r\sin(t) \\ r\cos(t) \\ 0 \end{pmatrix} .$$

Beispiel 3. *Die Kurve* $c : \mathbb{R} \to \mathbb{R}^3$, $c(t) := \begin{pmatrix} \cos(t) \\ \sin(t) \\ t \end{pmatrix}$ *beschreibt eine sogenannte Helix. Die Ableitung ist*

$$c'(t) = \begin{pmatrix} \cos'(t) \\ \sin'(t) \\ t' \end{pmatrix} = \begin{pmatrix} -\sin(t) \\ \cos(t) \\ 1 \end{pmatrix} .$$

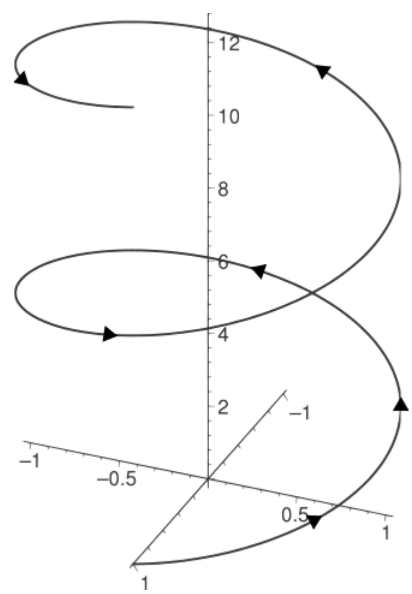


Abbildung 11: Eine Helix. Quelle:Wikipedia

Definition 46. *Ist* $c : I \subset \mathbb{R} \to \mathbb{R}^3$ *eine stückweise differenzierbare Kurve, so heißt*

$$l(c) := \int_I ||c'(t)||dt$$

ihre Länge.

Definition 47. *Ein Fläche ist eine Abbildung*

$$s : U \subset \mathbb{R}^2 \to \mathbb{R}^3$$
$$s(u,v) := \begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix}$$

bei der die Abbildungen $x, y, z : U \subset \mathbb{R}^2 \to \mathbb{R}^3$ stetig sind. Sie heißt differenzierbar, falls die partiellen Ableitungen

$$\frac{\partial}{\partial u} s(u,v) = \begin{pmatrix} \frac{\partial}{\partial u} x(u,v) \\ \frac{\partial}{\partial u} y(u,v) \\ \frac{\partial}{\partial u} z(u,v) \end{pmatrix}$$

und

$$\frac{\partial}{\partial v} s(u,v) = \begin{pmatrix} \frac{\partial}{\partial v} x(u,v) \\ \frac{\partial}{\partial v} y(u,v) \\ \frac{\partial}{\partial v} z(u,v) \end{pmatrix}$$

existieren. Die Ebene

$$T_s(u,v) := \{s(u,v) + \lambda \cdot \frac{\partial}{\partial u} s(u,v) + \mu \cdot \frac{\partial}{\partial v} \mid \lambda, \mu \in \mathbb{R}\}$$

heißt Tangentialebene am Punkt (u,v) und der Vektor

$$n(u,v) := \frac{\partial}{\partial u} s(u,v) \times \frac{\partial}{\partial v} s(u,v) \ ,$$

welcher Senkrecht auf dieser Ebene steht, die Normale.

2.3.2 Bezier Kurven und Flächen

Definition 48. *Die Bernsteinpolynome vom Grad n sind definiert als*

$$B_i^n(t) := \binom{n}{i} (1-t)^{n-i} t^i$$

mit $i = 0, \dots n$, $t \in [0,1]$ und dem Binomialkoeffizient

$$\binom{n}{i} := \frac{n!}{i!(n-i)!} = \frac{n(n-1)\cdots 1}{i(i-1)\cdots 1(n-i)(n-i-1)\cdots 1} \ .$$

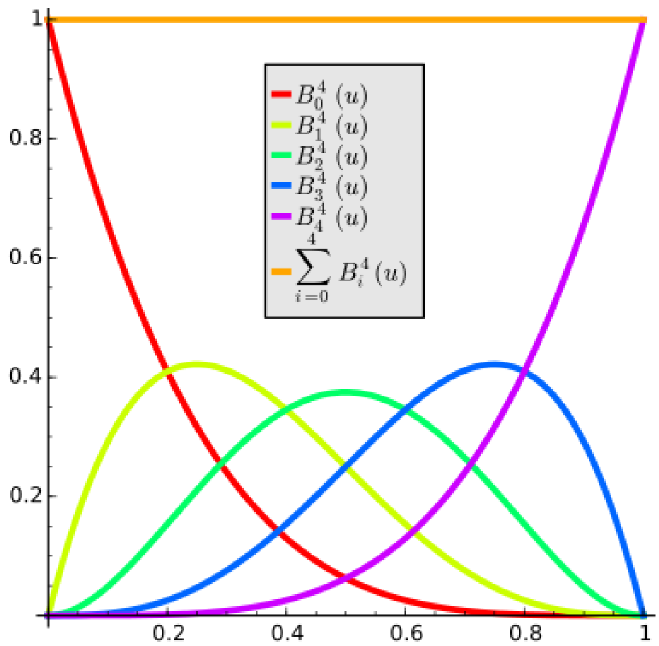


Abbildung 12: Die Bernsteinpolynome B_i^4 und deren Summe. Quelle:Wikipedia

Bemerkung 17. *Die Bernsteinpolynome vom Grad n bilden eine Basis des Vektorraums der Polynome vom Grad n im Intervall* $[0, 1]$.

Satz 15. *Es gilt die Rekursionsformel*

$$B_i^n(t) = (1-t) \cdot B_i^{n-1}(t) + t \cdot B_{i-1}^{n-1}(t)$$

mit $B_0^0(t) = 1$ *und* $B_n^i(t) = 0$ *für* $i < 0$ *oder* $i > n$.

Beweis. Folgt fast direkt aus der Rekursionsformel des Binomialkoeffizienten

$$\binom{n}{i} = \binom{n-1}{i} + \binom{n-1}{i-1}$$

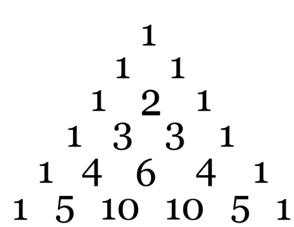


Abbildung 13: Pascalsches Dreieck

□

Definition 49. *Seien $b_0, \dots b_n \in \mathbb{R}^3$. Dann heißt die Kurve*

$$B^n(t) := \sum_{i=0}^{n} B_i^n(t) \cdot b_i \ , t \in [0,1]$$

eine Bezierkurve vom Grad n. Die b_i werden auch Kontrollpunkte genannt. Für ein beliebiges Intervall $[a,b]$ definieren wir

$$B_{[a,b]}^n(t) := B^n\left(\frac{t-a}{a-b}\right) , t \in [a,b] \ .$$

Satz 16. *Eine Bezierkurve hat die Ableitung*

- $(B^n)'(t) = n \cdot \sum_{j=0}^{n-1} B_j^{n-1}(t) \cdot (b_{j+1} - b_j)$, *und nach der Kettenregel*
- $(B_{[a,b]}^n)'(t) = \frac{1}{b-a}(B^n)'\left(\frac{t-a}{b-a}\right)$ *für ein beliebiges Parameterintervall.*

Satz 17 (Algorithmus von de Casteljau)**.** *Sei $B^n(t) := \sum_{i=0}^{n} B_i^n(t) \cdot b_i$ eine Bezierkurve. Für ein $t_0 \in [0,1]$ definieren wir rekursiv*

$$b_i^k := \begin{cases} b_i & i = 0, \dots, n \\ (1-t_0) \cdot b_{i-1}^{k-1} + t_0 \cdot b_i^{k-1} & i = 1, \dots, n \ \ k = 1, \dots, i \end{cases}$$

was sich schematisch folgendermaßen darstellen lässt:

$$\begin{tikzcd}
b_0 & = & b_0^0 \arrow[dr, "\cdot(1-t_0)"] & & & & \\
b_1 & = & b_1^0 \arrow[r, "\cdot t_0"] \arrow[dr, "\cdot(1-t_0)"] & b_1^1 \arrow[dr, "\cdot(1-t_0)"] & & & \\
b_2 \arrow[d, dotted, no head] & = & b_2^0 \arrow[r, "\cdot t_0"] \arrow[d, dotted, no head] & b_2^1 \arrow[r, "\cdot t_0"] \arrow[d, dotted, no head] & b_2^2 \arrow[dr, dotted, no head] & & \\
b_{n-1} & = & b_{n-1}^0 \arrow[r, "\cdot t_0"] \arrow[dr, "\cdot(1-t_0)"] & b_{n-1}^1 \arrow[r, "\cdot t_0"] \arrow[dr, "\cdot(1-t_0)"] & b_{n-1}^2 \arrow[r, dotted, no head] & b_{n-1}^{n-1} \arrow[dr, "\cdot(1-t_0)"] & \\
b_n & = & b_n^0 \arrow[r, "\cdot t_0"] & b_n^1 \arrow[r, "\cdot t_0"] & b_n^2 \arrow[r, dotted, no head] & b_n^{n-1} \arrow[r, "\cdot t_0"] & b_n^n
\end{tikzcd}$$

Dann gilt $b_n^n = B^n(t_0)$.

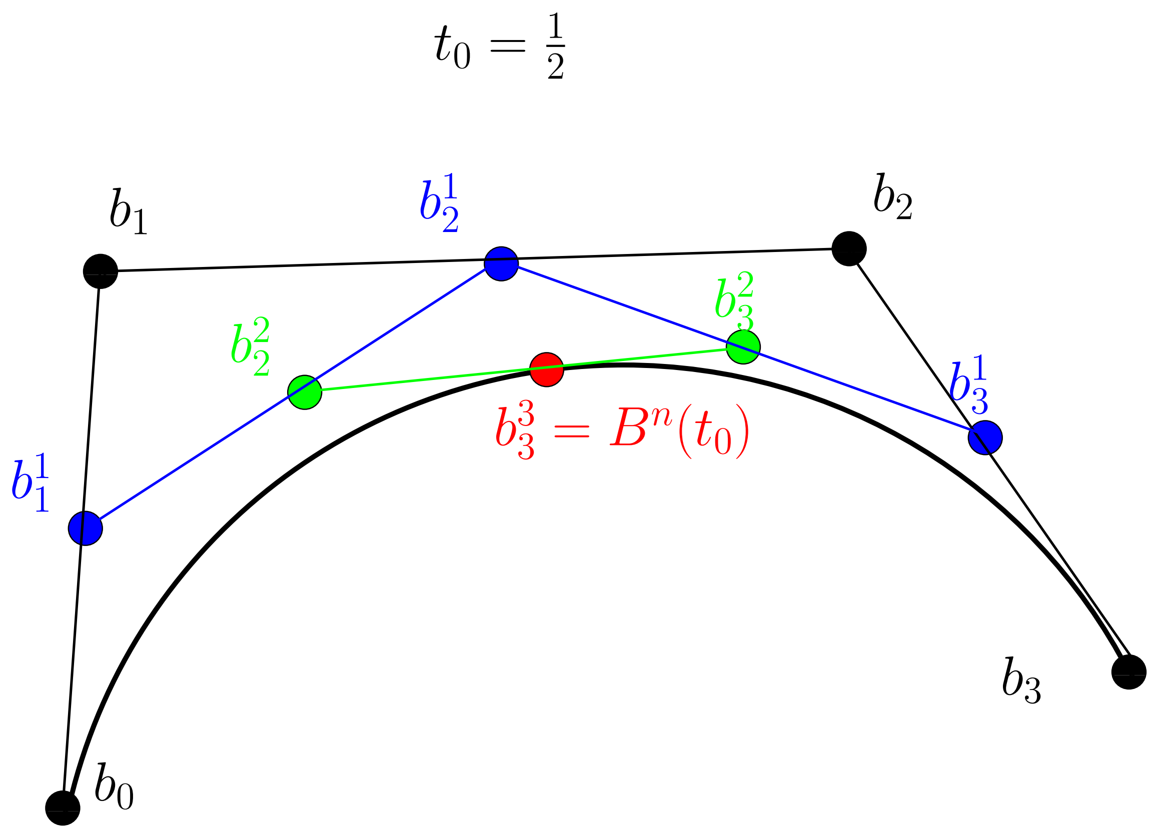


Abbildung 14: Auswertung einer Beziekurve zum Zeitpunkt t_0 durch itterative Streckenteilung nach de Casteljau

Definition 50 (Patching)**.** *Seien $B^n(t)$ und $(B^*)^m(t)$ Bezierkurven. Man spricht von einem C^0-Patching (an der Stelle $B^n(1)$), falls $B^n(1) = (B^*)^m(0)$ gilt. Stimmen auch die Ableitungen überein, also $(B^n)'(1) = ((B^*)^m)'(0)$, dann spricht man von einem C^1-Patching und stimmen noch für $k > 1$ auch die höheren Ableitungen $(B^n)^k(1) = ((B^*)^m)k(0)$ überein, so spricht man von einem C^k-Patching .*

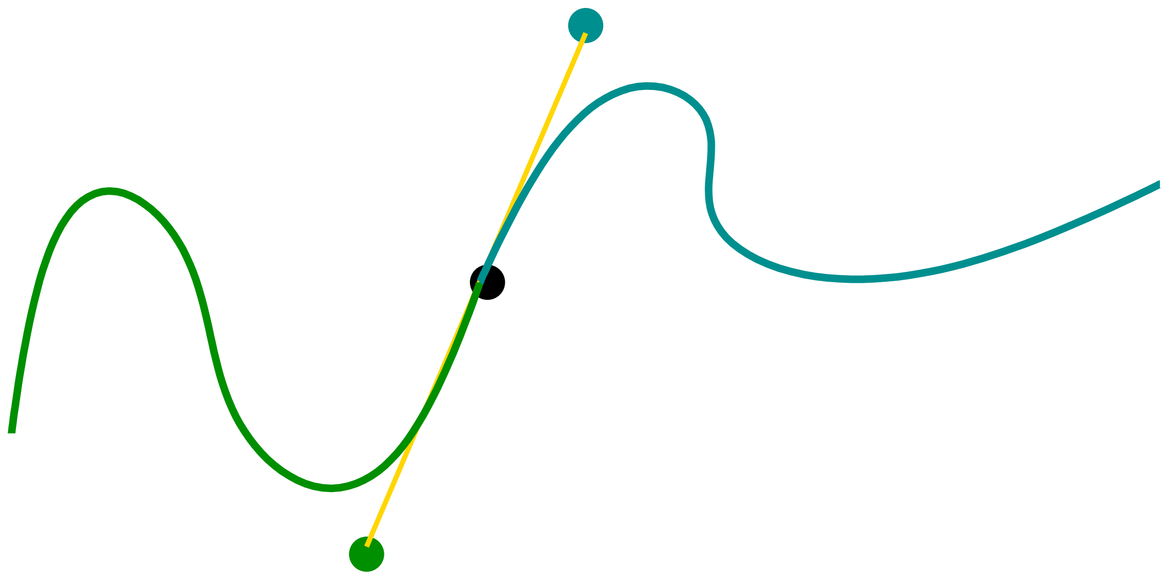


Abbildung 15: Bezier Patch

Bemerkung 18. *Zwei Bezierkurven* $B^n(t) := \sum_{i=0}^n B_i^n(t) \cdot b_i$ *und* $(B^*)^m(t) := \sum_{i=0}^m B_i^n(t) \cdot b_i^*$ *bilden genau dann einen* C^1*-Patch (an der Stelle* $B^n(1)$*), wenn* $B^n(1) = (B^*)^m(0)$ *und* $\Big(B^n(1) - b_{n-1}\Big) = -\frac{m}{n}\Big((B^*)^m(0) - (b^*)_1\Big)$ *gilt.*

Definition 51. *Ist* $B^n(t)$ *eine Bezierkurve, so bilden die Bezierkurven* $(B^*)^n(t) := \sum_{i=0}^n B_i^n(t/2) \cdot b_i^i$ *und* $(B^{**})^n(t) := \sum_{i=0}^n B_i^n(\frac{t+1}{2}) \cdot b_n^{n-i}$ *einen* C^1*-Patch.*

Definition 52. *Ersetzt man in einer Bezierkurve*

$$B^m(v) := \sum_{j=0}^m B_j^m(v) \cdot b_j$$

vom Grad m *die Kontrollpunkte* b_j *durch* $m+1$ *Bezierkurven*

$$b_j(u) = \sum_{i=0}^n B_i^n(u) \cdot b_{ij}$$

vom Grad n *mit jeweils* $n+1$ *Kontrollpunkten* b_{ij} *für* $i = 0, \cdots, n$*, so erhält man eine Fläche*

$$F(u,v) = \sum_{i=0}^n \sum_{j=0}^m B_i^n(u) B_j^m(v) \cdot b_{ij} \ ,$$

welche auch Tenorprodukt-Fläche der Bezierkurven oder einfach Bezierfläche genannt wird.

Abbildung 16: Ein Rendering der Utah Teekanne, eines der weit verbreitetsten 3D-Modelle in der Computergrafik. Sie wurde mit Bezierflächen modelliert.

3 Echtzeitvisualisierung und OpenGL

3.1 Geschichte

OpenGL entstand ursprünglich aus dem von Silicon Graphics (SGI) entwickelten IRIS GL. Im sogenannten Fahrenheit-Projekt versuchten Microsoft und SGI ihre 3D-Standards zu vereinheitlichen, das Projekt wurde jedoch wegen finanzieller Schwierigkeiten auf Seiten von SGI abgebrochen.

Der OpenGL-Standard wird vom OpenGL ARB (Architecture Review Board) festgelegt. Das ARB existiert seit 1992 und besteht aus einer Reihe von Firmen. Stimmberechtigte Mitglieder sind die Firmen 3DLabs, Apple, AMD/ATI, Dell, IBM, Intel, Nvidia, SGI und Sun (Stand Nov. 2004). Weiter mitwirkende Firmen sind Evans and Sutherland, Imagination Technologies, Matrox, Quantum3D, S3 Graphics, Spinor GmbH, Tungsten Graphics, und Xi Graphics. Microsoft, eines der Gründungsmitglieder, hat das ARB im März 2003 verlassen.

Neue Funktionen in OpenGL werden meist zuerst als herstellerspezifische Erweiterungen eingeführt und gehen dann den Weg über herstellerübergreifende Erweiterungen und ARB-Erweiterungen zu Kernfunktionalität. Dies erlaubt es, neueste Möglichkeiten der Grafikhardware zu nutzen und dennoch OpenGL abstrakt genug zu halten.

Quelle: https://de.wikipedia.org/wiki/OpenGL

3.2 GL-Pipeline

Eine Computergrafik-Pipeline besteht im Wesentlichen aus den folgenden Schritten:

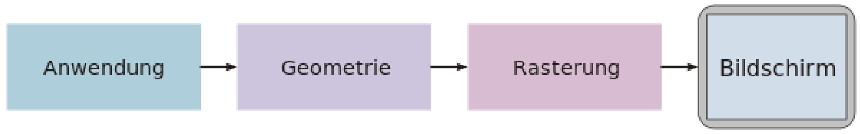


Abbildung 17: Computergrafik-Pipeline

Eine solche Pipeline wird auch von OpenGL implementiert. Es besteht jedoch die Besonderheit, dass die Geometrie und die Rasterung in Hardware realisiert ist und man durch kleine Programme, sogenannte Shader, auf diese Hardware zugreifen und Manipulationen vornehmen kann beziehungsweise seit OpenGL 2.0 sogar muss.

3.2.1 Geometrie

Die Geometrieverarbeitung besteht aus der Hintereinanderausführung der folgenden affinen/homogenen Abbildungen und Algorithmen:

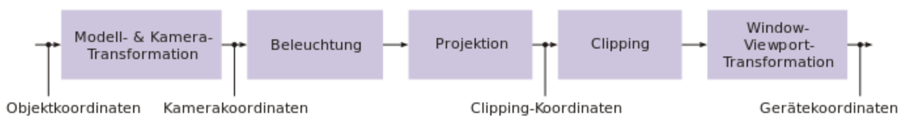


Abbildung 18: Geometrie-Pipeline

Modell und Kameratransformationen

Die Transformationen von Objektkoordinaten in das Kamerakoordinatensystem werden durch affine Transformationen beschrieben, welche durch Multiplikation durch homogene 4×4-Matrizen realisiert werden.

Projektion

Die Projektion wird durch Matrizen ähnlich der Projektionsmatrix $K_{persp_{xy}}$ und $K_{orth_{xy}}$ aus Abschnitt 1.2.1 realisiert. Es werden jedoch noch Translationen, Rotationen und Stauchungen dazwischen geschaltet, die ebenfalls als 4×4-Matrizen realisiert werden können und mit denen diese Matrix multipliziert werden. Man erreicht damit, dass der Kegelstumpf zwischen der vorderen (*nearplane*) und der hinteren (*farplane*) Projektionsebene in den $[-1, 1] \times [-1, 1] \times [-1, 1]$ Würfel abgebildet wird, welcher auch Sichtvolumen genannt wird.

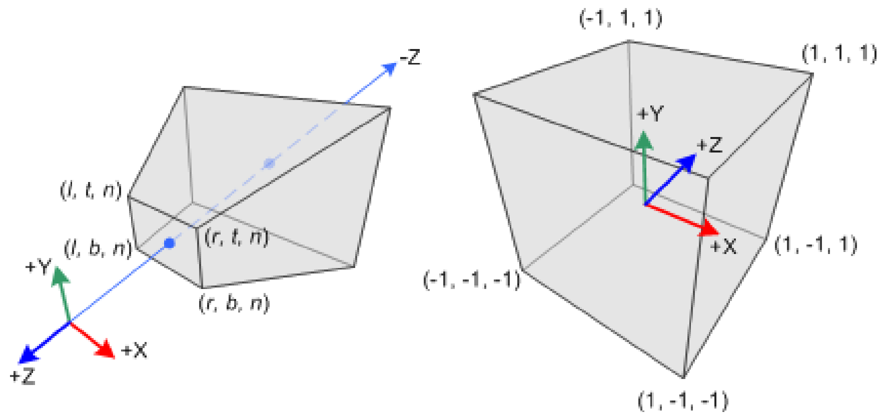
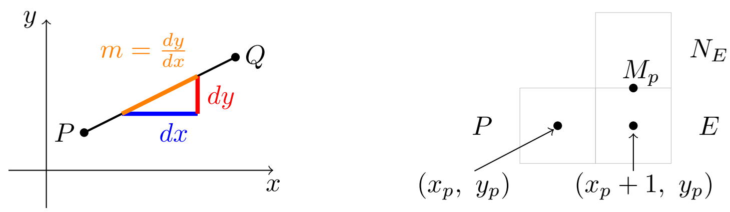


Abbildung 19: Projektion und Sichtvolumen

Die Multiplikation all diese Matrizen wird auch die MODEL-VIEW-PROJECTION-MATRIX genannt.

Windows-Viewport-Transformation

Zuletzt müssen die zweidimensionalen Punkte noch mit Hilfe einer zweidimensionalen affinen Transformation in das Koordinatensystems des Anzeigenfensters auf dem Ausgabegerät transformiert werden. Diese Transformation wird auch Windows-Viewport-Transformation genannt.



(a) Steigungsdreieck **(b)** Pixelraster

Abbildung 21: Schrittbetrachtung Bresenham Algorithmus

3.2.2 Rasterung

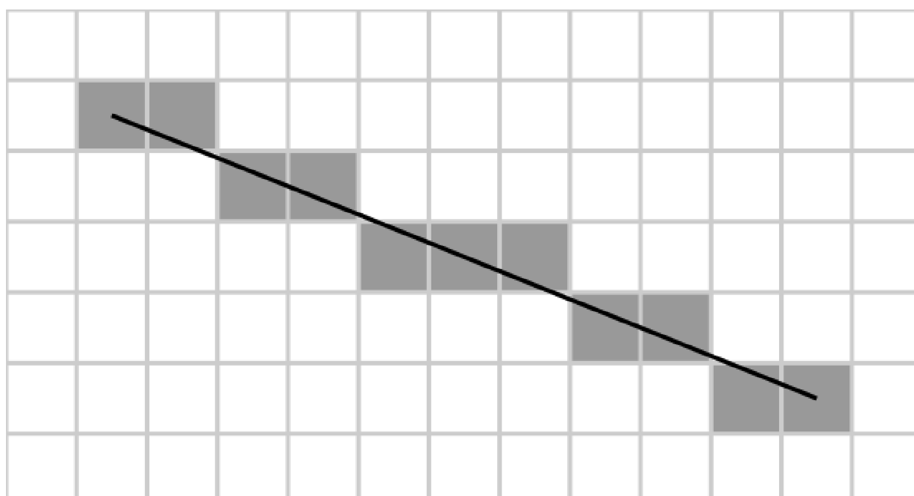


Abbildung 20: Rasterung nach Bresenham Algorithmus

Als Rasterung bezeichnet man das Transformieren kontinuierlicher, zweidimensionaler Daten auf diskrete Pixel. Anhand der gegebenen Daten muss also entschieden werden, welche Farbe ein Pixel des Ausgabegerätes erhält. Wir haben also eine Funktion $Frame{-}Buffer : \mathbb{N}\times\mathbb{N} \rightarrow Farbe$. Im Bereich Echtzeitvisualisierung sind alle gängigen Rasterverfahren für Dreiecke und andere primitiven im wesentlichen Abwandlungen und Weiterentwicklungen von Algorithmen, die von Bresenham eingeführt wurden. Wir wollen uns den Bresenham Algorithmus für Linien/Strecken dazu exemplarisch anschauen.

Algorithmus 2 (Bresenham für Strecken mit positiver Steigung)**.** *Die Strecke $S = \overline{PQ} \in \mathbb{R}^n$ soll auf ein (n-dimensionales Pixel-) Raster abgebildet werden (Siehe Abbildung 20). Wir beschränken uns hier auf die Berechnung im $\mathbb{R}^2$. P*

und Q *seien gegeben mit*

$$P = \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \quad Q = \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} \quad \forall x_i, y_i \in \mathbb{R}$$

Die Geradengleichung lässt sich einfach bestimmen (Abbildung 21a):

$$\begin{array}{ccc} y & = & mx + b \\ & mit & \\ m & = & \frac{dy}{dx} \\ dx & = & x_1 - x_0 \\ dy & = & y_1 - y_0 \end{array}$$

Die Abbildung auf ein Raster ist für Steigungen $m \in \{0,\ 1\}$ *trivial. Anders für* $0 < m < 1$*. Dafür betrachten wir in einem Schritt benachbarte Pixel (Abbildung 21b auf der vorherigen Seite):*

- *Ausgangspixel* $P\,(x_p,\ y_p)$
- *Nachbar* $E\,(x_p + 1,\ y_p)$
- *„Nachfolgender" Nachbar* $N_E\,(x_p + 1,\ y_p + 1)$

Der Mittelpunkt zwischen N *und* E *wird festgelegt durch*

$$M_p = \left(x_p + 1,\ y_p + \frac{1}{2}\right)$$

Damit kann entschieden werden welches benachbarte Pixel zur Strecke gehört. Liegt nun

$$M_p \begin{cases} \textit{oberhalb } \overline{PQ} & \Rightarrow \quad \textit{wähle } E \\ \textit{unterhalb } \overline{PQ} & \Rightarrow \quad \textit{wähle } N_E \end{cases}$$

Bresenham hat eine effiziente Lösung dieses Problems aufgezeigt. Zunächst lässt sich die Geradengleichung in eine Funktion F *umformen, die von zwei Variablen abhängt:*

$$\begin{array}{ccc} y & = & \frac{dy}{dx}x + b \\ dx \cdot y & = & dy \cdot x + dx \cdot b \\ 0 & = & dy \cdot x - dx \cdot y + dx \cdot b \\ F\,(x,\ y) & := & dy \cdot x - dx \cdot y + dx \cdot b \end{array}$$

Das arithmetische Entscheidungskriterium lautet also

$$\begin{array}{ccc} (x,\ y) \in y = mx + b & \Leftrightarrow & F\,(x,\ y) = 0 \\ & bzw. & \\ F\,(x,\ y) & & \begin{cases} = 0 & (x,\ y) \textit{ liegt auf } S \\ > 0 & (x,\ y) \textit{ liegt unterhalb von } S \\ < 0 & (x,\ y) \textit{ liegt oberhalb von } S \end{cases} \end{array}$$

Damit kann die Entscheidungsfunktion D_p definiert werden

$$\begin{aligned} D_p &:= 2 \cdot F\left(M_p\right) \\ &= 2 \cdot F\left(x_p + 1,\ y_p + \tfrac{1}{2}\right) \\ &= dy\left(2x_p + 2\right) - dx\left(2y_p + 1\right) + dx \cdot 2b \end{aligned}$$

1. Fall: $D_p < 0 \;\Rightarrow\;$ *Nachfolgepixel ist* E

$$\begin{aligned} D_E &= 2 \cdot F\left(M_E\right) \\ &= 2 \cdot F\left(x_p + 2,\ y_p + \tfrac{1}{2}\right) \\ &= dy \cdot \left(2x_p + 4\right) - dx \cdot \left(2y_p + 1\right) + 2 \cdot b \cdot dx \\ &= D_p + \triangle_E \\ \mathit{mit} \quad & \triangle_E := 2 \cdot dy \end{aligned}$$

2. Fall: $D_p \geq 0 \;\Rightarrow\;$ *Nachfolgepixel ist* NE

$$\begin{aligned} D_{NE} &= 2 \cdot F\left(M_{NE}\right) \\ &= D_p + \triangle_{NE} \\ \mathit{mit} \quad & \triangle_{NE} := 2dy - 2dx \end{aligned}$$

Eine Implementierung könnte wie folgt aussehen:

```
bresenham(x0, y0, x1, y1){
    dx = x1 - x0;
    dy = y1 - y0;
    d = 2*dy - dx; /*F(x0 + 1,y0 + 1/2) | F(x0,y0)=0*/
    deltaE = 2 * dy;
    deltaNE = 2 * (dy - dx);
    x = x0;
    y = y0;
    writepixel(x0, y0);
    while (x < x1) {
        if (d < 0) {
            d += deltaE;
            x++;
        } else {
            d += deltaNE;
            x++; y++;
        }
        writepixel(x, y);
    }
}
```

Das Rastern erzeugt oft harte, kantige und ausgefranste Übergange. Diese Effekte bezeichnet man auch als Aliasing. Algorithmen, die diesen Aliasing-Effekten entgegenwirken nennt man auch Antialiasing.

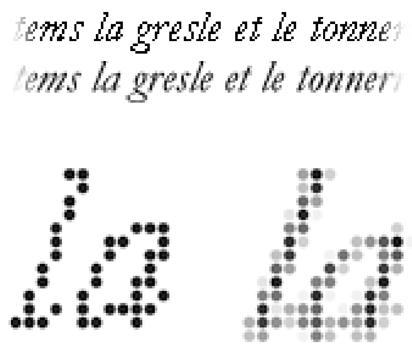


Abbildung 22: Eine Schrift mit und ohne Antialiasing

Sichtbarkeitsprobleme

Clipping

Beim 3D-Clipping werden alle Polygone verworfen, die nach der Transformation in Clipping-Koordinaten vollständig ausserhalb des Sichtvolumens $[-1,1] \times [-1,1] \times [-1,1]$ liegen. Das 2D-Clipping findet nach der Projektion der Polygone statt. Polygone, die über das Viewportfenster hinaus ragen, werden an den Fensterkanten abgeschnitten. Der wichtigste Schritt ist hierbei das Abschneiden von Strecken an diesen Kanten. Wir betrachten dazu exemplarisch den Cohen-Sutherland-Algorithmus:

Algorithmus 3 (Cohen-Sutherland)**.**

Culling

Als Culling bezeichnet man das Entfernen von Polygonen anhand Ihrer Orientierung. Man definiert, welche Orientierung eine Rückseite darstellt und verwirft dann Polygone mit dieser Orientierung.

z-Buffer

Der Z-Buffer enthält für jedes Pixel (u, v) nach der Rasterung einen Wert zwischen -1 und 1. Dieser Wert ist der Abstand zum nächsten Punkt eines Polygons in Clipping-Koordinaten von diesem Pixel auf der Projektionsebene. Wir haben somit eine Funktion

$$Z - Buffer : \mathbb{N} \times \mathbb{N} \to [-1, 1] \; .$$

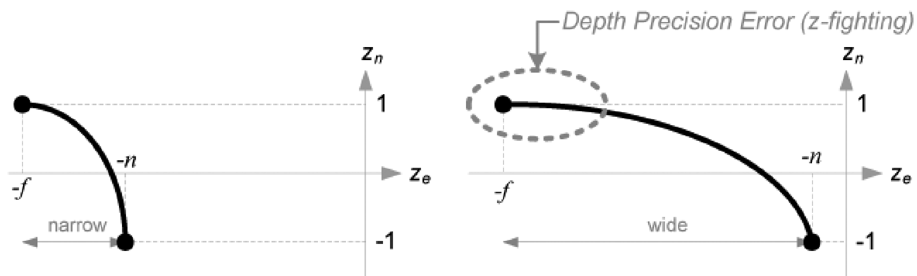


Abbildung 23: Z-Buffer-War

Algorithmus 4 (z-Buffer Algorithmus)**.** *Setze Alle Einträge im Z-Buffer auf 1 (Hintergrund). Setze alle Einträge im Framebuffer auf die Hintergrundfarbe.*
$\forall$ *Polygone P:*
begin:
Transformiere das Polygon in Clipping-Koordinaten.
Transformiere das Polygon in Rasterkoordinaten.
$\forall$ *erhaltenen Pixel (u,v):*
begin:
pz :=Z-Koordinate des Polygons in Clipping-Koordinaten zum entsprechenden Pixel
zz := *Eintrag im Z-Buffer für das Pixel* (u, v)
if $pz < zz$*:*
begin:
$Z - Buffer(u, v) := pz$
$Frame - Buffer(u, v) := Farbe(P, (u, v))$
end
end
end

3.3 Lokale Beleuchtungsmodelle

3.3.1 Ideale und diffuse Reflexionen

3.3.2 Lambert Modell

Das Lambertsche Modell beschreibt, wie durch den perspektivischen Effekt die Strahlungsstärke mit flacher werdendem Abstrahlwinkel abnimmt. Sei nun L der Punkt, an dem sich eine punktförmige Lichtquelle befindet und $P \in N$ ein Punkt eines Netzes N mit Normale n (also insbesondere gilt $||n|| = 1$). Dann berechnet sich die Lichtintensität I_d an diesem Punkt durch

$$I_d := I_l \cdot I_m \cdot max\left(0, \left\langle n, \frac{1}{||\overline{PL}||} \cdot \overline{PL} \right\rangle\right)$$

wobei $I_l \in [0, 1]$ die Lichtintensität der Lichtquelle, $I_m \in [0, 1]$ eine Materialkonstante und $max(a, b)$ das maximum der Zahlen a und b ist. Um die gesamte

Lichtintensität zu berechnen wird noch ein konstanter, sogenannter ambienter Lichtanteil I_a hinzuaddiert

$$I := I_d + I_a \ . \tag{1}$$

I_a ist wie gesagt eine Konstante. Der Eindruck von ambienten Licht, also Licht, das aus keiner erkennbaren Richtung kommt und alles gleichmässig erhellt, entsteht durch Lichtstrahlen, die sehr oft an Gegenständen reflektiert werden. Der Wert I muss gegebenenfalls auf das Intervall $[0, 1]$ beschränkt werden.

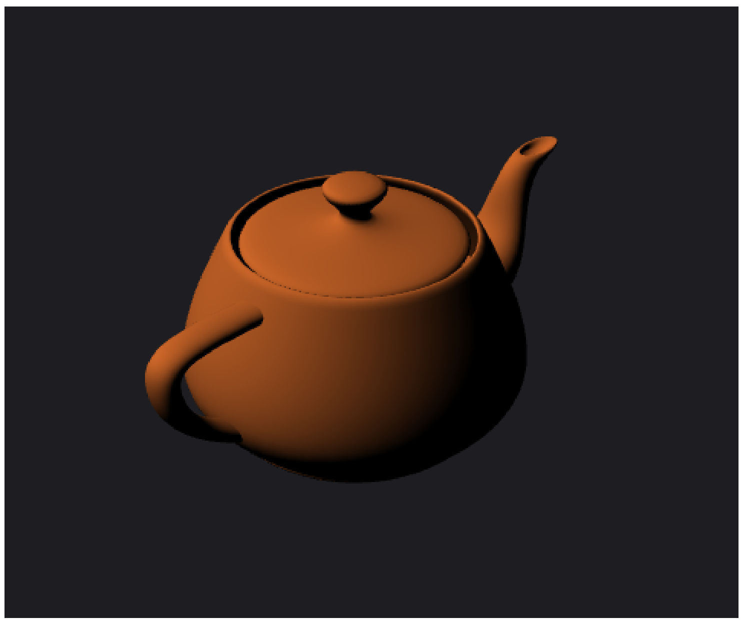


Abbildung 24: Lambert Modell diffuser Reflektion

3.3.3 Phong Modell

Das Phong-Modell erweitert das Lambert-Modell um den Effekt von spiegelnden Reflektionen. Die Idee ist, dass die Helligkeit zusätzlich zunimmt, je mehr die Betrachtungsrichtung dem perfekt reflektiertem Lichtstrahl entspricht.

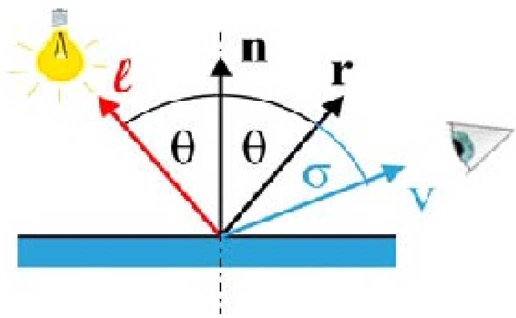


Abbildung 25: Die einzelnen Komponenten des Phong-Modells

Ist v der normierte Vektor in Richtung Kamera und l der normierte Vektor in Richtung Lichtquelle, dann definiert man die reflektierende Lichtintensität durch

$$I_s := I_l \cdot I_m \cdot s \cdot < r, v >^h \quad . \tag{2}$$

I_l ist hierbei wieder die Intensität der Lichtquelle und I_m eine Materialkonstante. Beide liegen in einen Wertebereich von $[0, 1]$ h ist eine ganze Zahl im Wertebereich $[0, \infty)$ und bestimmt die "härte"der Reflexion. s ist eine Fließkommazahl im Wertebereich $(0, 1]$ und beeinflusst den Glanz der Oberfläche. Der Vektor r lässt sich berechnen durch

$$r = -l + 2 \cdot < n, l > \cdot n \tag{3}$$

wobei n die Normale an dem betrachteten Oberflächenpunkt ist.

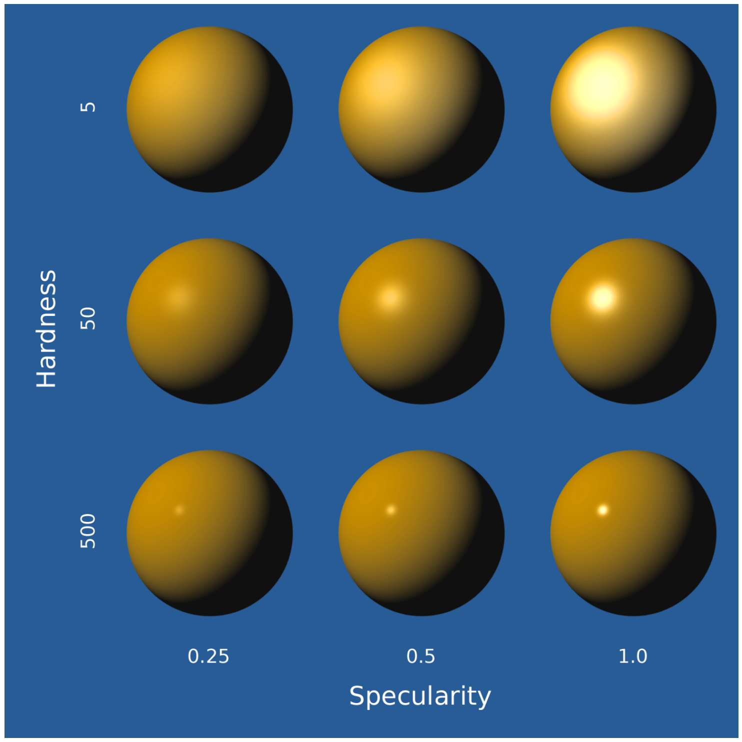


Abbildung 26: Phong Modell spiegelnder Reflektion

Im Phong-Modell wird nun endgültige Lichtentität wie folgt zusammengesetzt:

$$I := I_s + I_d + I_a \tag{4}$$

wobei I_d die diffuse Lichtintensität aus dem Lambert-Modell und I_a den konstanten, ambienten Anteil bedeutet. Gegebenenfalls muss I wieder auf das Intervall $[0, 1]$ beschränkt werden.

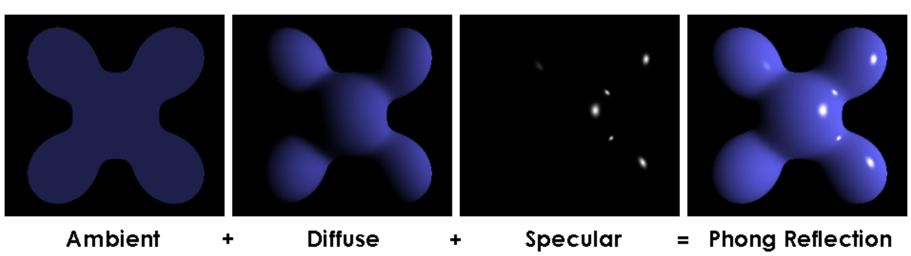


Abbildung 27: Die einzelnen Komponenten des Phong-Modells

Die Möglichkeiten, ein bestimmtes Material zu beschreiben, sind beim Phong-Modell beschränkt. Es gibt komplexere lokale Beleuchtungsmodelle, wie zum Beispiel das Cook-Torrente-Modell, die eine detaillierte Materialbeschreibung zulassen.

3.4 Shader und standard Algorithmen

3.4.1 Flat-Shading

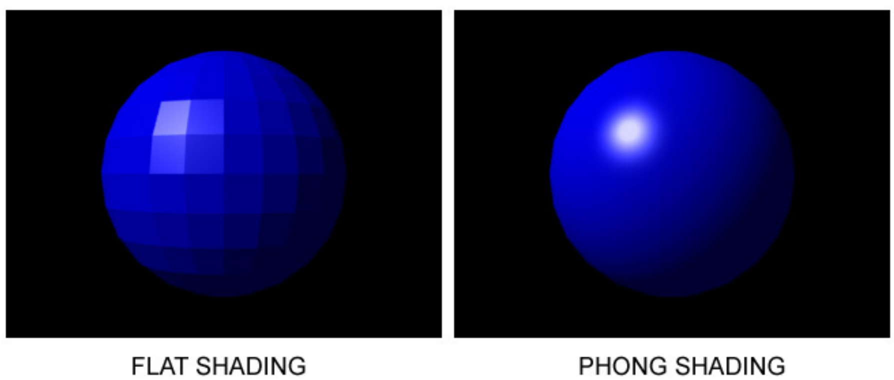


Abbildung 28: Flat- und Phong-Shading-Modelle

3.4.2 Gouraud und Phong Shading

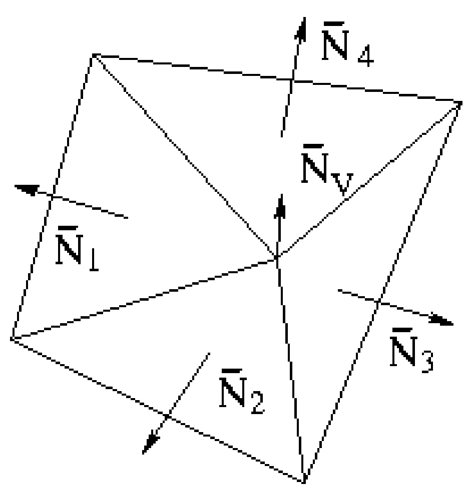


Abbildung 29: Verschiedene Normalen von angrenzenden Flächen

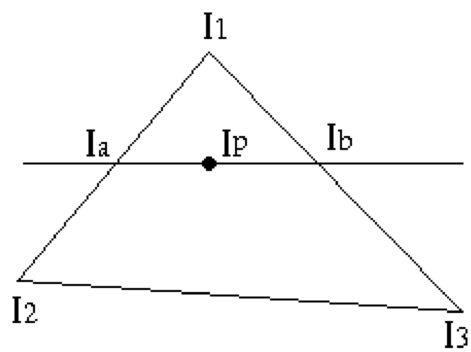


Abbildung 30: Scanline-Verfahren beim Gouraud-Shading

3.4.3 Texturen und UV-Mapping

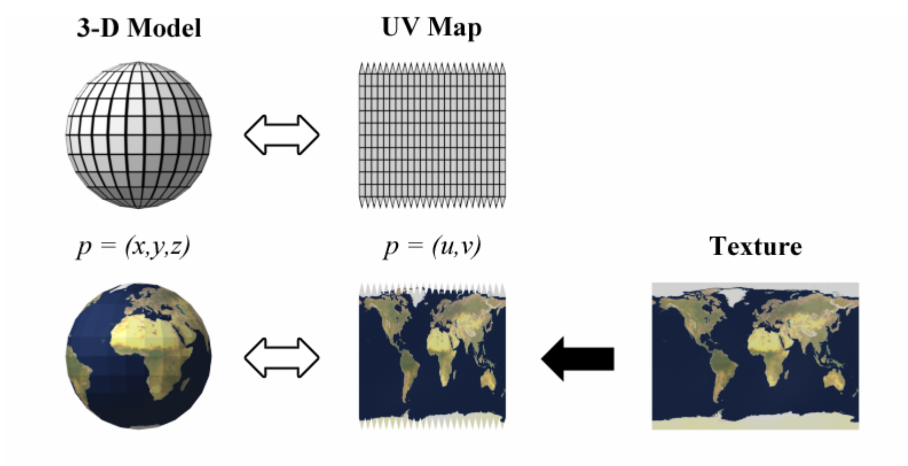


Abbildung 31: uv mapping

Abbildung 32: uv mapping

3.4.4 Bumpmapping

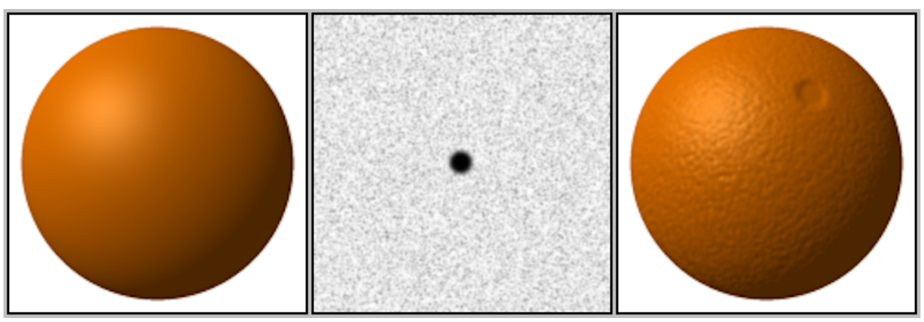


Abbildung 33: uv mapping

3.4.5 Displacementmapping

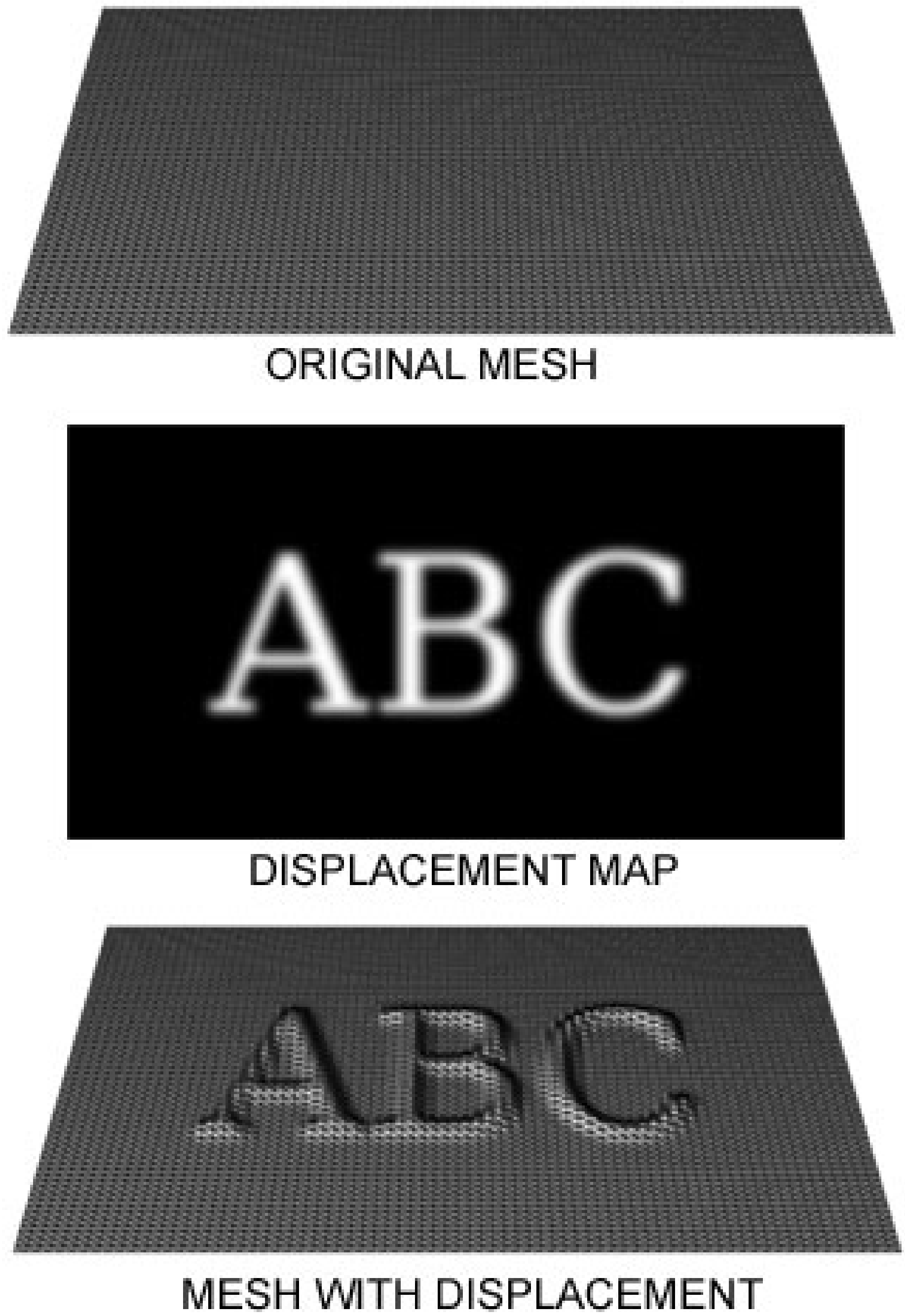


Abbildung 34: Displacementmapping

3.4.6 Shadowmap

3.4.7 Bumpmapping

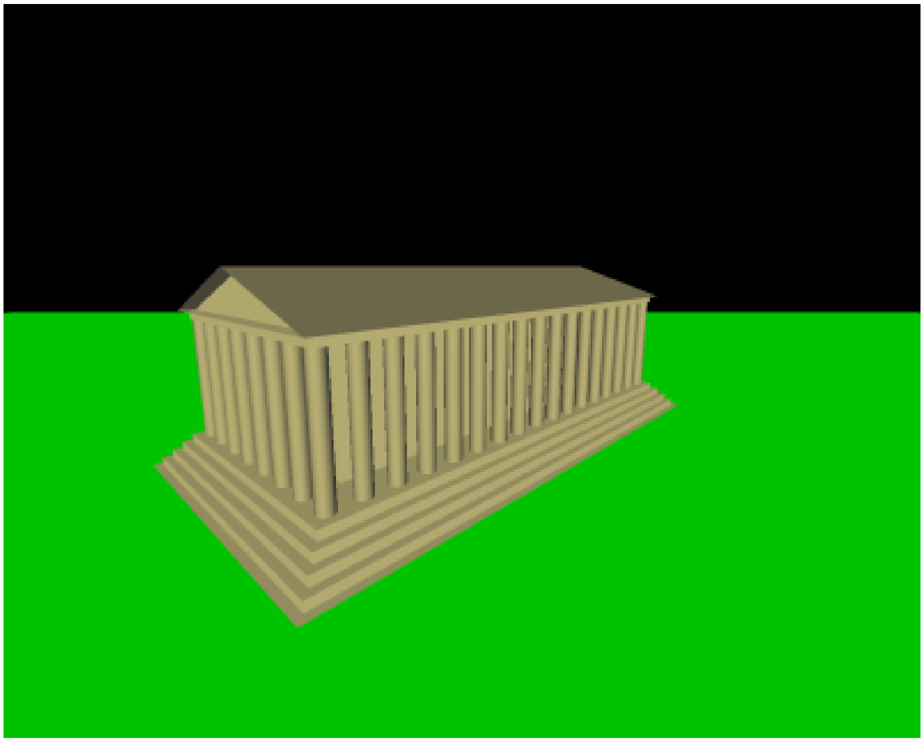


Abbildung 35: shadowmap

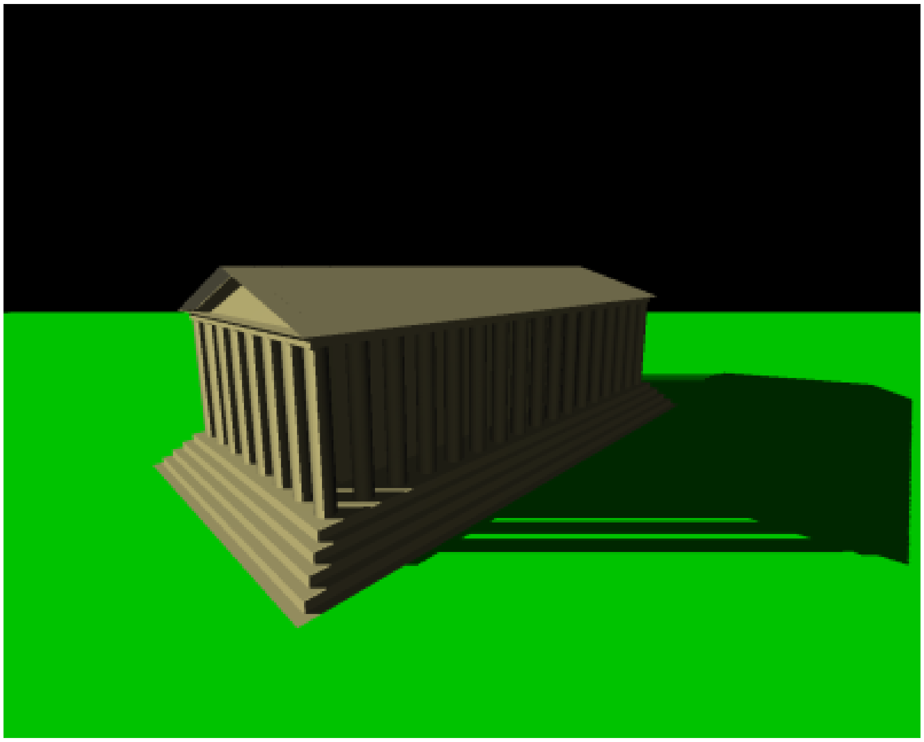


Abbildung 36: shadowmap

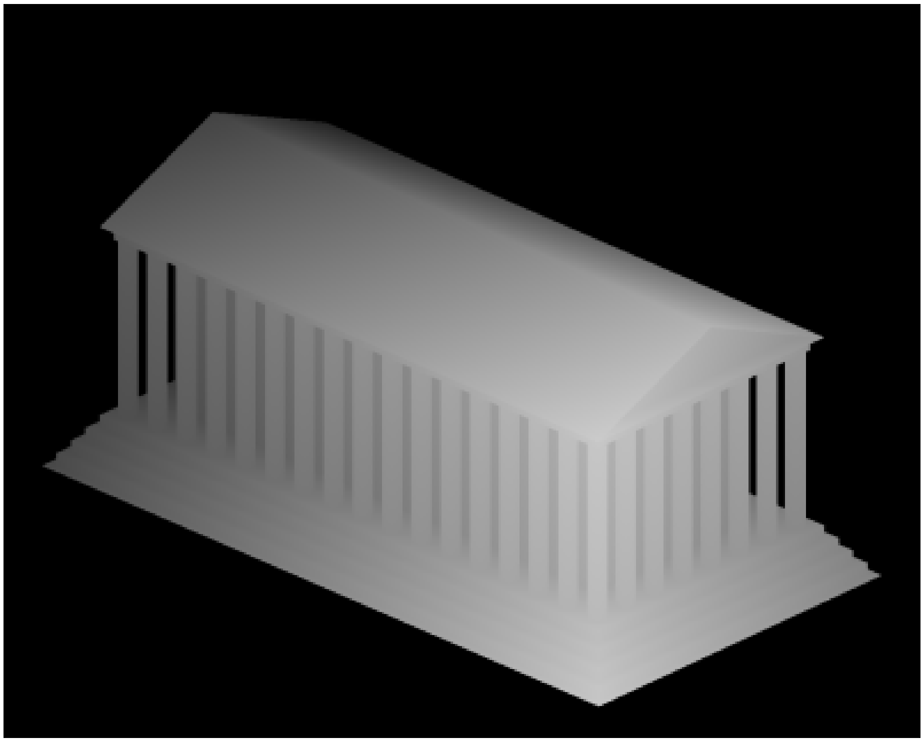


Abbildung 37: shadowmap

3.4.8 Forward shading, Blending und Transparenz

3.4.9 Deffered shading

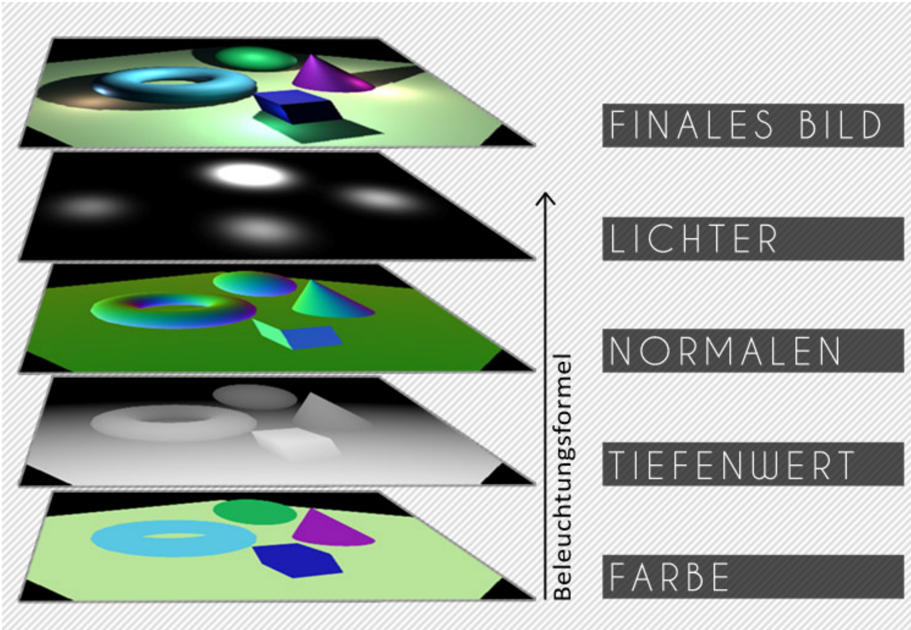


Abbildung 38: deffered shading

3.5 GLSL via WebGL

4 Raytracing

4.1 Farbwahrnehmung und Farbmodelle

4.2 Globale Beleuchtungsmodelle und Rendergleichung

4.2.1 Photometrie

Die Radiantenergie Q ist die Lichtenergie. Sie wird durch einen Strom von Photonen erzeugt. Die Energie eines Photons ist durch $E = h \cdot f$ geben, wobei h das konstante Plancksche Wirkungsquantum und f die Frequenz der Welle ist (Welle-Teilchen Dualismus). Die Ableitung nach der Zeit

$$\phi := \frac{\partial Q}{\partial t} \tag{5}$$

wird als Radiant Flux oder Strahlungsleistung bezeichnet. Diese beschreibt den Energiefluss, beziehungsweise den Energie-Eintritt und Energie-Austritt.

Der Radiant Flux in einem infinitesimal dünnen Strahl vom Punkt x in Richtung ω wird als Radiance $L(x, \omega)$ bezeichnet. Formal lässt sich dies durch

$$L(x, \omega) := \frac{d^2\phi}{\cos(\theta) dA \cdot d\omega} \tag{6}$$

ausdrücken. Die Radiance ist also der Radiant Flux pro infinitesimale Flächeneinheit $\cos(\theta)dA$ und pro Raumwinkel $d\omega$.

4.2.2 BRDF und Reflectancegleichung

Die sogenannte bidirektionale Reflektanzverteilungsfunktion (engl. Bidirectional Reflectance Distribution Function, BRDF) ist eine Funktion $f_r(x, \omega_i, \omega_r)$, die das Reflexionsverhalten der Oberfläche eines Materials beschreibt. Sie hat als Eingabe die ausgehende Richtung ω_r und die eingehende Richtung ω_i am Punkt x. Sie liefert den Quotienten aus Strahlungsdichte und Bestrahlungsstärke für die ausgehende Richtung ω_r und die eingehende Richtung ω_i am Punkt x. Sie gibt somit die Abhängigkeit des reflektierten Lichts von der einfallenden Lichtstärke an:

$$dL_r = f_r(x, \omega_i, \omega_r) dE = f_r(x, \omega_i, \omega_r) \cdot L_i(x, \omega_i) \cdot \cos(\theta_i) d\omega_i \tag{7}$$

$$L_r(x, \omega_r) = \int_{S^2} dL_r = \int_{S^2} f_r(x, \omega_i, \omega_r) \cdot L_i(x, \omega_i) \cdot \cos(\theta_i) d\omega_i \tag{8}$$

Letztere Gleichung wird auch Reflectance-Gleichung genannt.

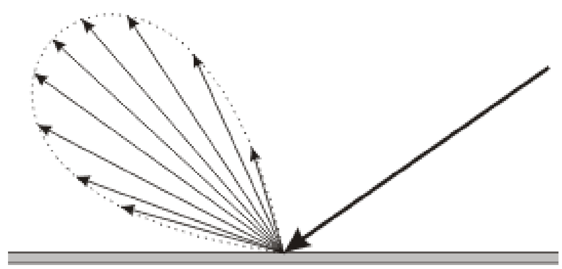


Abbildung 39: BRDF Funktion

4.2.3 Das photometrische Grundgesetz

4.2.4 Die Rendergleichung (Erste und zweite Form)

Durch Hinzunahme des von dem Oberflächenpunkt emittierten Lichtes $L_e(x, \omega_o)$ zu dem an diesem Punkt reflektiert Lichtes erhalten wir die sogenannte Renderingleichung

$$L_o(x, \omega_o) = L_e(x, \omega_o) + \int_{S^2} f_r(x, \omega_i, \omega_0) \cdot L_i(x, \omega_i) \cdot \cos(\theta_i) d\omega_i \ . \tag{9}$$

Dieses Integral kann durch eine Integraltransformation in eine handlichere Form gebracht werden. Insbesondere die Berechnung der Einfallenden Irradiance $L_i(x, \omega_i)$ ist in keinster weise offensichtlich.

Durch die Beziehung $d_o mega_i =$

4.3 Raycasting

4.3.1 "Klassisches"Raytracing

4.3.2 Radiocity Verfahren

4.3.3 Monte Carlo Integration und Pathtracing

4.3.4 Raymarching

4.4 Klassifikation der Verfahren

4.5 Datenstrukturen für Bereichsabfragen

4.6 Labor

4.6.1 Blender

4.6.2 Echtzeitfähiges Raymarching in WebGL

5 Animation und Simulation

5.1 Keyframe Animation

5.2 Partikelsysteme

5.3 Elemente der Kollisionserkennung

5.4 Labor

Tabellenverzeichnis

Abbildungsverzeichnis