

Vorlesungen

„Einführung in die Datenvisualisierung“

„Angewandte Datenvisualisierung für
Medizinphysiker“

Kapitel B. Bilderzeugung

Dr. Frank Weichert
Informatik VII
Technische Universität Dortmund
<http://ls7-www.cs.uni-dortmund.de>

Nur zum persönlichen Gebrauch durch Mitglieder der Universität Dortmund

B.1. Einleitung

- B.1.1. Ablauf der Datenvisualisierung
- B.1.2. Bilderzeugungs-Pipeline
- B.1.3. Graphikelemente
- B.1.4. Farbmodelle

B.2. Zweidimensionale Bilderzeugung

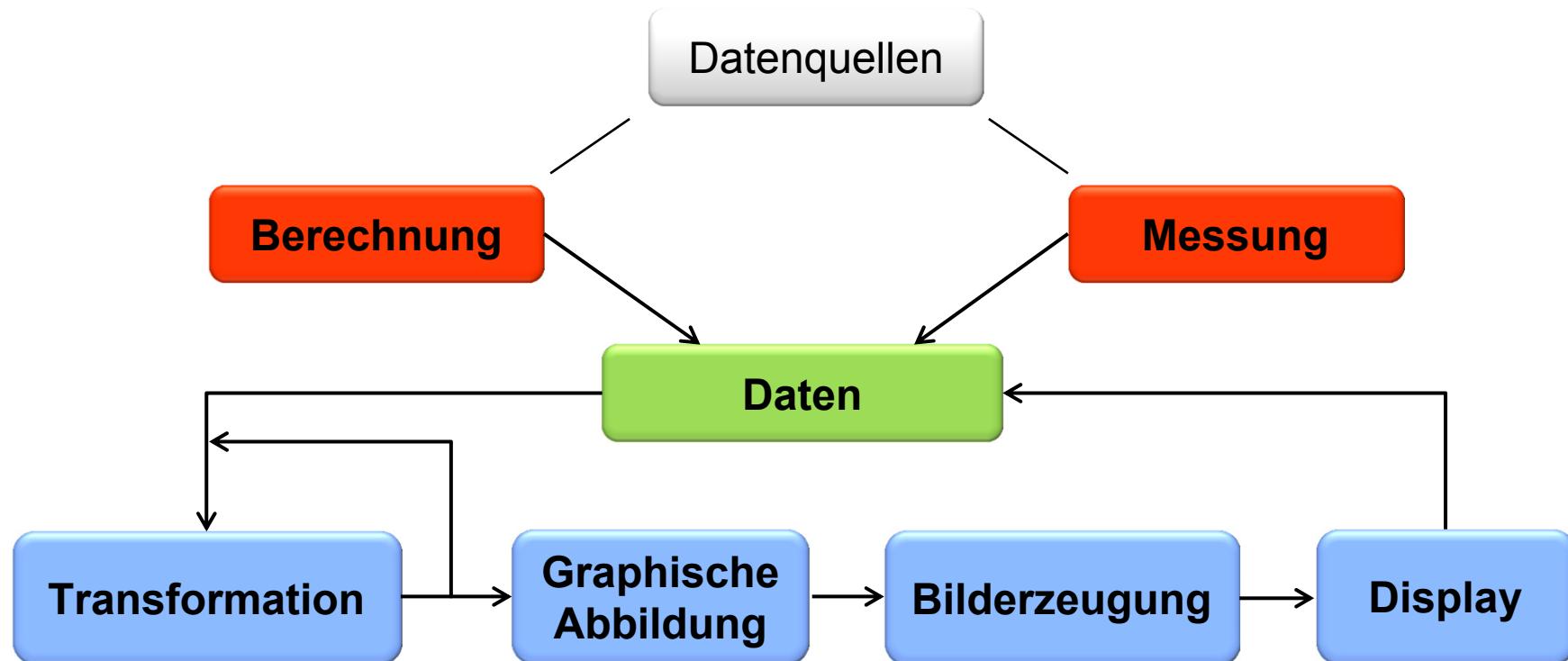
- B.2.1. Transformation
- B.2.2. Clipping
- B.2.3. Verrasterung

B.3. Dreidimensionale Bilderzeugung

- B.3.1. Transformation
- B.3.2. Beleuchtung
- B.3.3. Projektion
- B.3.4. Sichtbarkeitsberechnung

B.4. Beleuchtungssimulation

- B.4.1. Strahlverfolgung (Raytracing)
 - B.4.2. Strahlungsverfahren (Radiosity)
-



Bilderzeugung:

überführt eine Szenenbeschreibung in eine bildliche Darstellung

Fallunterscheidung:

- zweidimensionale Bilderzeugung
- dreidimensionale Bilderzeugung

Graphik-/Bilderzeugungs-Pipelines:

- Transformieren einer Bildbeschreibung (Szene) in ein Bild
- Grundprinzip heutiger hardware- und software-basierter Graphiksysteme

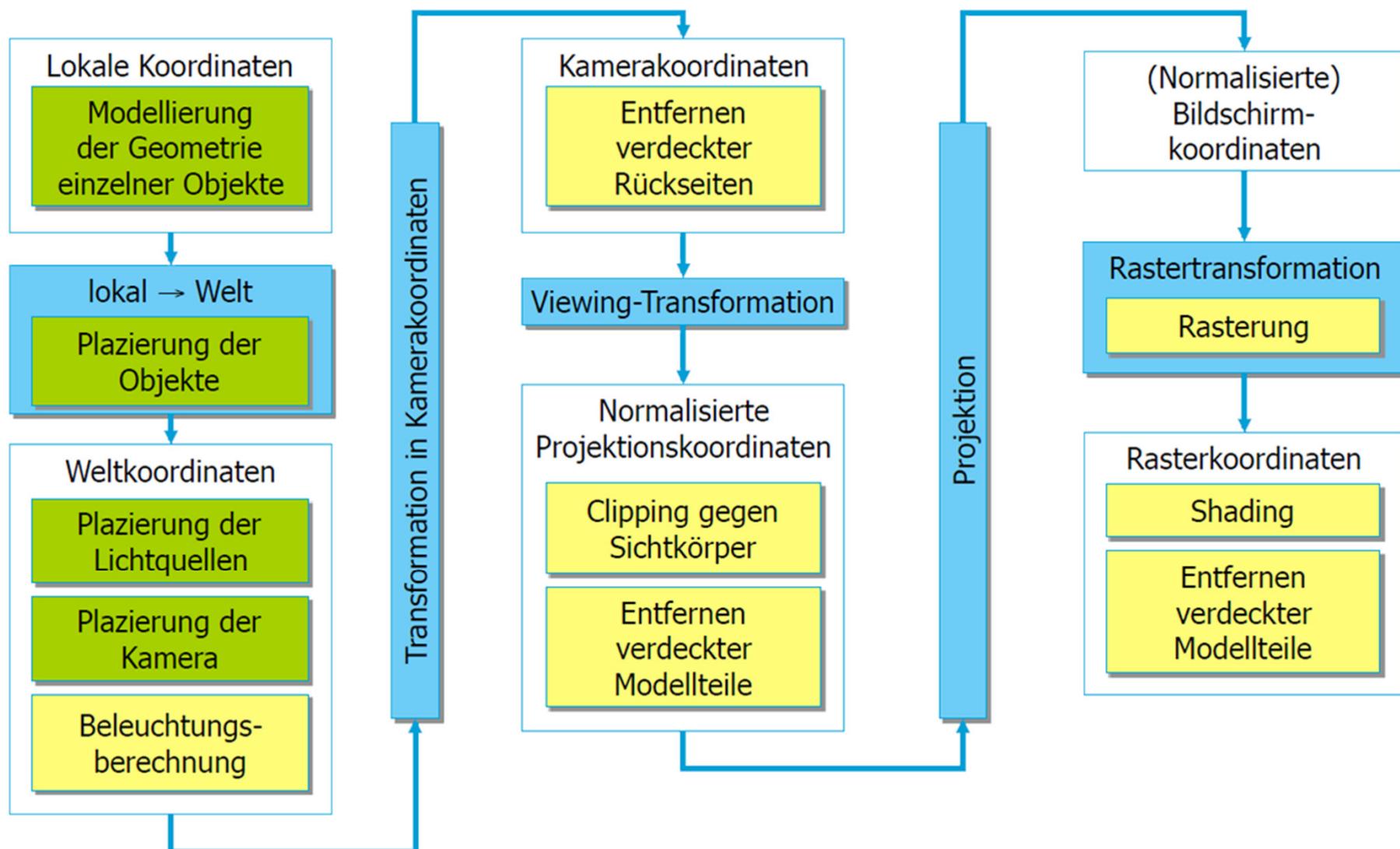
Im Folgenden:

Kurzeinführung

Details:

- Nachfolgende Kapitel
- Vorlesungen
 - Mensch-Maschine-Interaktion
 - Graphische Datenverarbeitung
- Lehrbücher zur (dreidimensionalen) Computergraphik

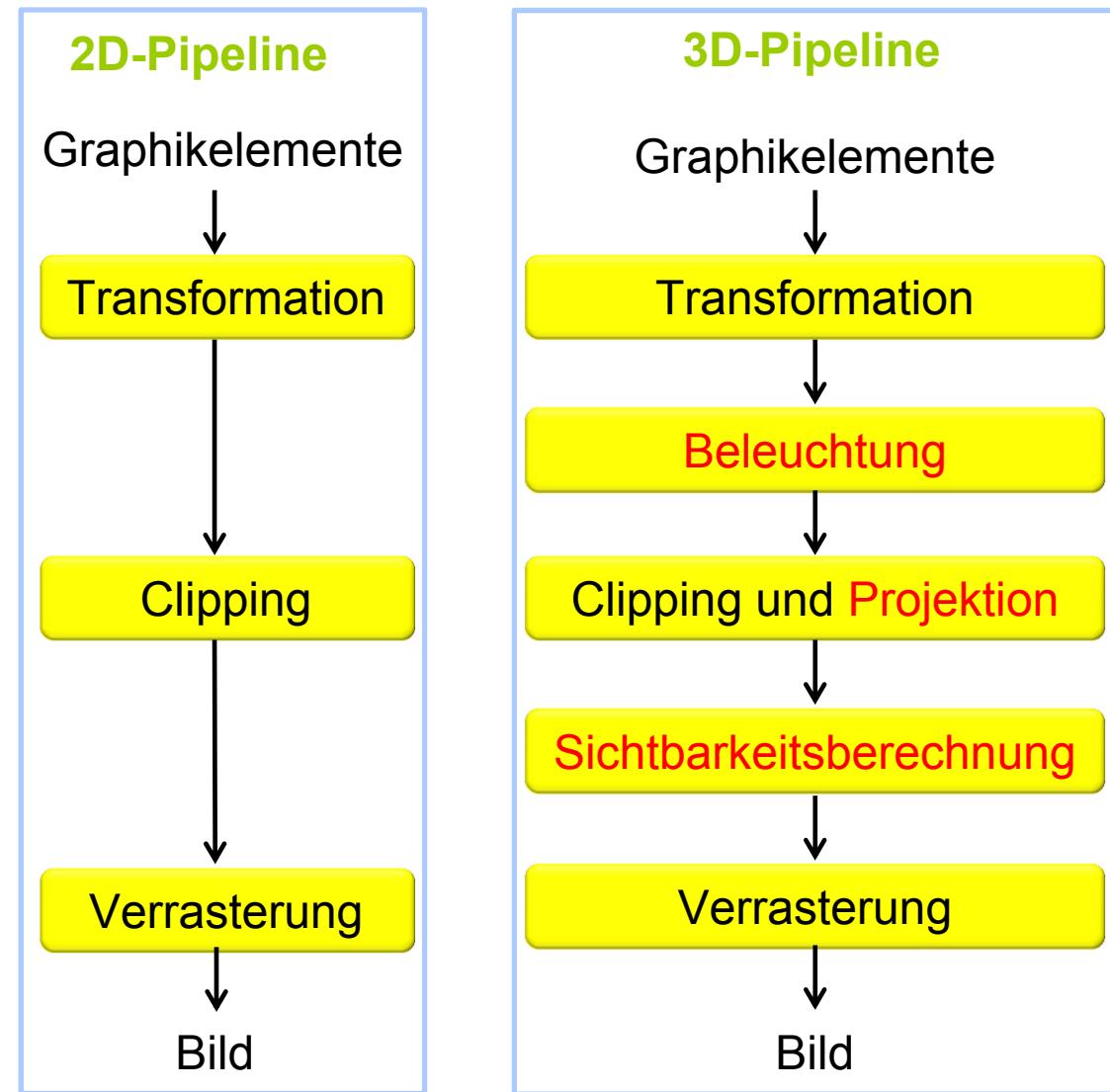
Zusammensetzen der Pipeline



B.1. Einleitung

B.1.2. Bilderzeugungs-Pipelines

Vergleich zwischen den
2D- und 3D-
Bilderzeugungs-Pipelines



Teilschritte der Bilderzeugungspipelines

1. Transformation:

Anwendung einer affinen Abbildung auf die Szene

Bsp.: Szene aus Dreiecken, Anwendung der affinen Abbildung, z.B. in homogener Darstellung, auf die Eckpunkte der Dreiecke

2. Beleuchtung:

Anwendung einer bidirektionalen Shading-Funktion (BSF) an Stützstellen der Szene, aus denen dann bei der Verrasterung durch Interpolation die Farbe berechnet wird.

Bsp.: Szene aus Dreiecken, Auswertung der BSF an den Eckenpunkten, Interpolation durch Gouraud-Shading

3. Clipping:

Festlegung einer Sehpyramide, deren Inneres den Teil der Szene definiert, der im Bild dargestellt wird.



Teilschritte der Bilderzeugungspipelines

4. Projektion:

Projektion der dreidimensionalen Szene auf eine zweidimensionale Bildebene

Bsp.: perspektivische Projektion, Parallelprojektion

5. Sichtbarkeitsberechnung:

Entfernung der nicht sichtbaren Teile der Szene bezüglich eines Augenpunktes oder einer Blickrichtung

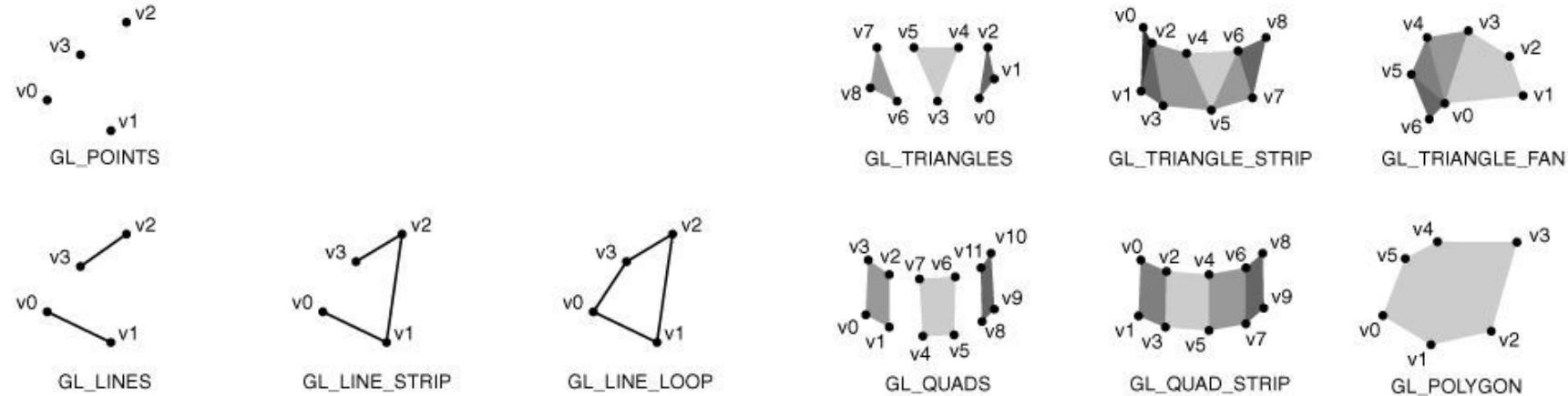
Bsp.: Sichtbarkeitsberechnung mit dem Tiefenpufferalgorithmus (z-buffer)

6. Verrasterung:

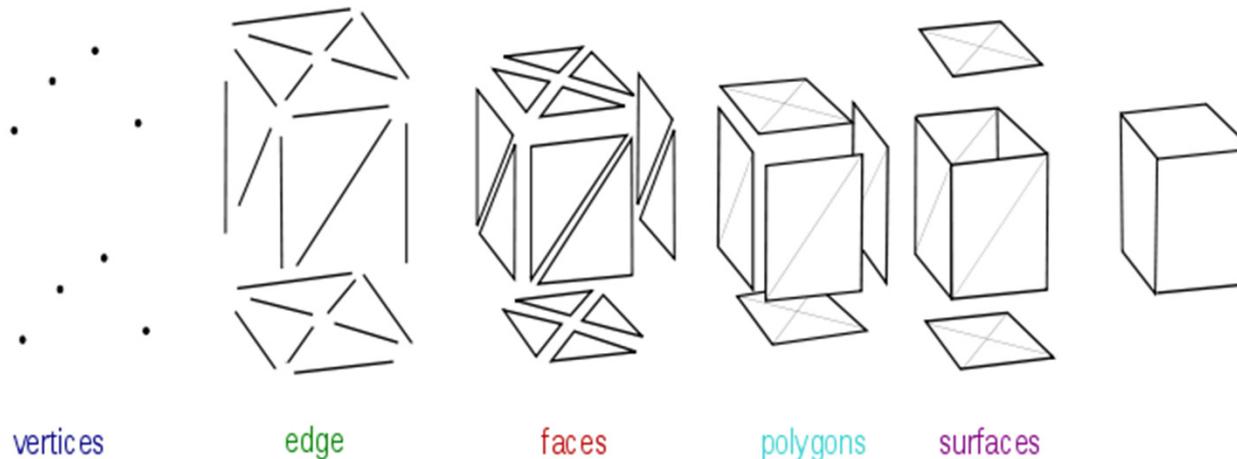
Umwandlung der gegebenen geometrischen Repräsentation der Szene in eine Rasterdarstellung. Manchmal wird dies schon zusammen mit der Sichtbarkeitsberechnung erledigt, z.B. beim Tiefenpufferalgorithmus.



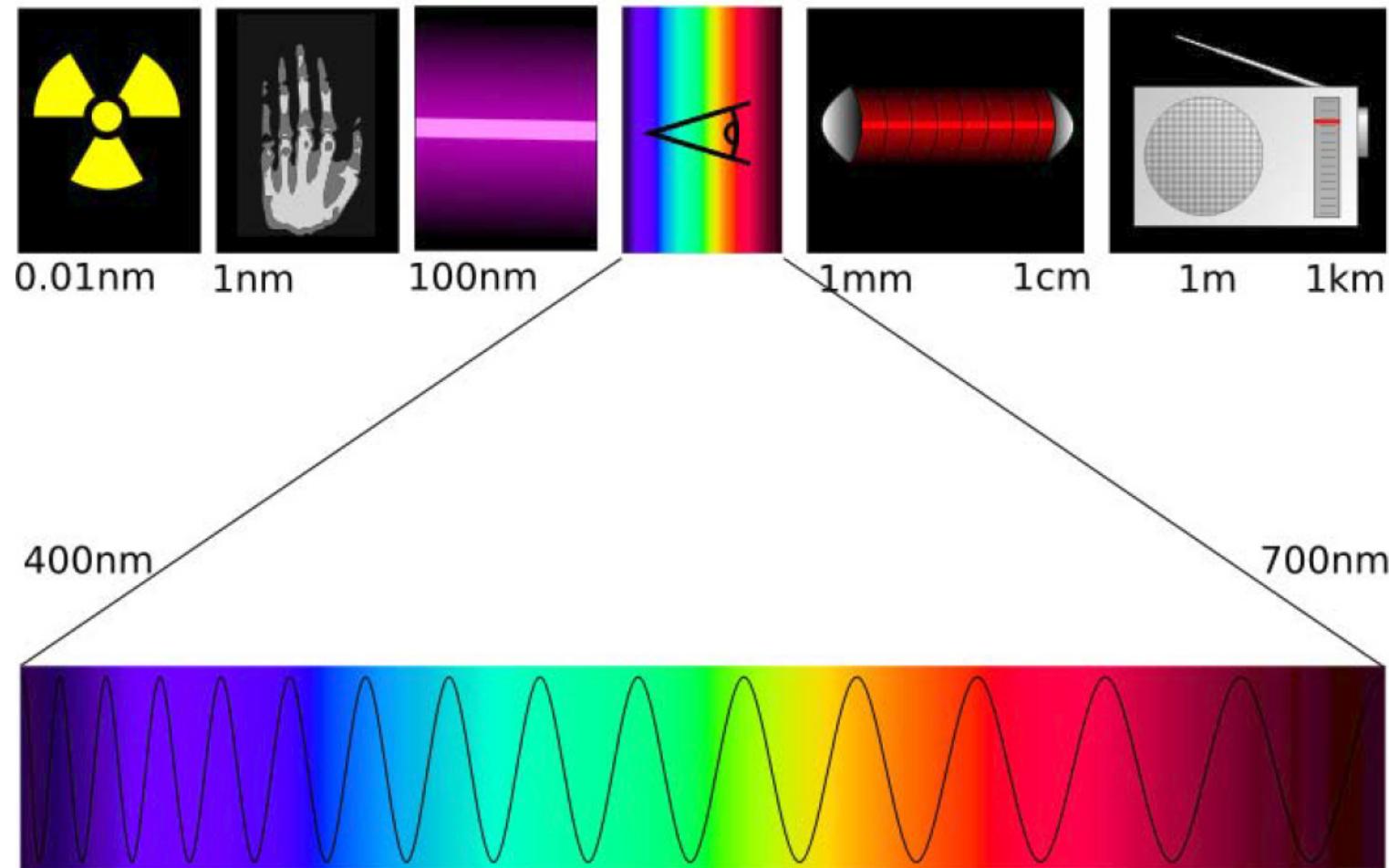
2D-Graphikelemente (Bsp.: OpenGL)



3D-Graphikelemente (Bsp.: Meshes)

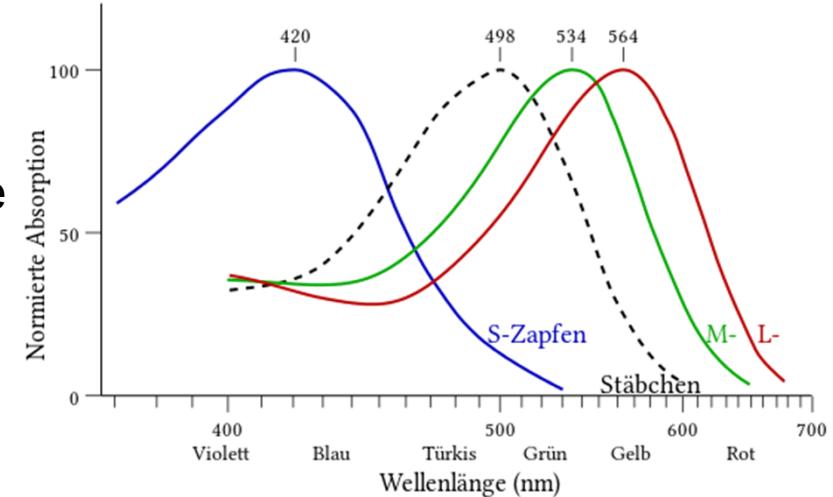


Wellenlängen



Dreifarbenlehre

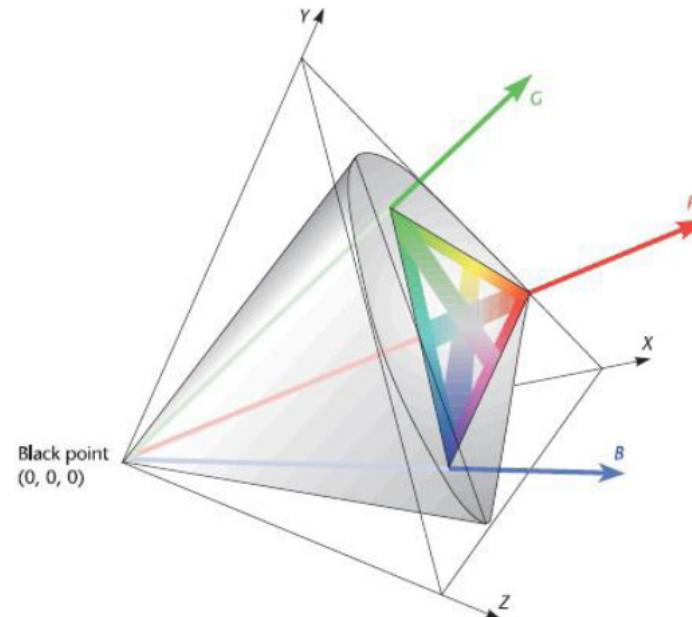
- Mensch verfügt über drei Typen von „Seheinheiten“ (Zapfen), deren maximale Sensitivität bei einer anderen Wellenlänge liegt ($S = \text{blau}$, $M = \text{grün}$ und $L = \text{rot}$).



- Ein Farbsystem, bestehend aus drei Grundtönen, um alle vom Menschen wahrnehmbaren Farben darzustellen (→ **Dreifarbenlehre**).
- Unterschiedliche Sensitivität der Zapfen.
- Farben werden können in einem (3D-)Farbraum dargestellt werden

CIE Standard für Farben

- Commission Internationale de l'Eclairage (CIE) nutzt drei abstrakte Primärfarben X , Y , Z . Y entspricht der Leuchtdichte (Helligkeit von Lichtflächen)
- Wahrnehmbare Farben sind als graues Volumen dargestellt.
- Farben, welche durch drei farbige Lichter in rot, grün und blau erzeugt werden können, entsprechen der einbeschriebenen Pyramide.



CIE Standard für Farben

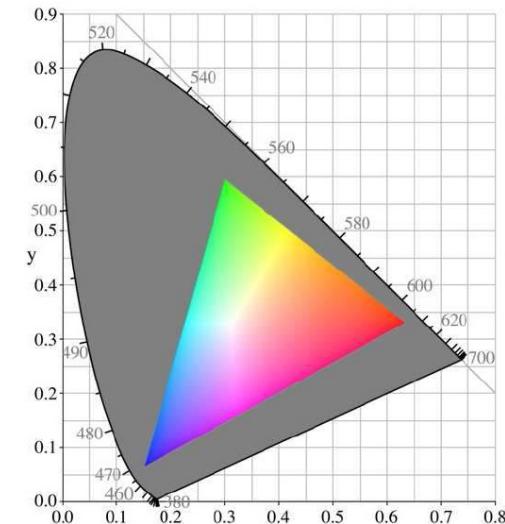
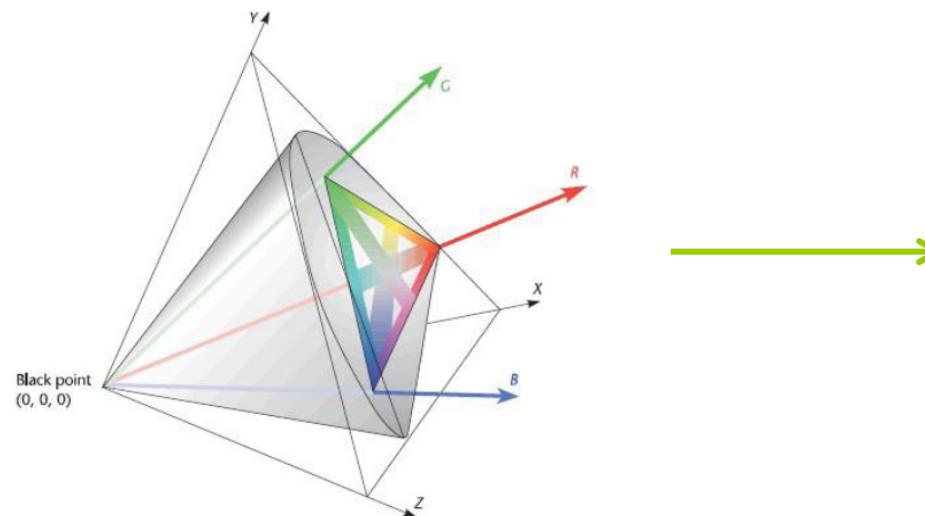
- Farben in XYZ Koordinaten sind nur schwer zu verstehen. Leichter verständlich ist die Repräsentation in Farbwertanteilen (chromaticity coordinates):

$$x = X / (X+Y+Z)$$

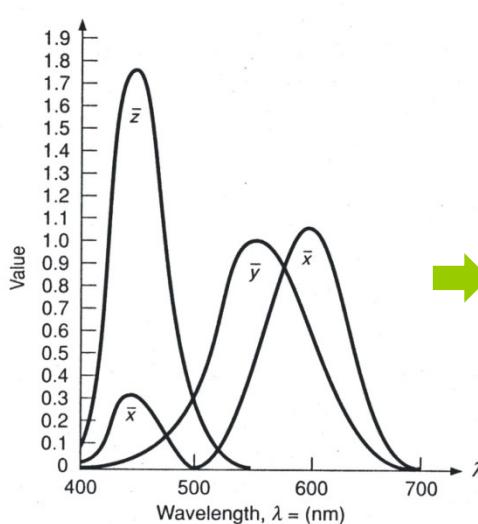
$$y = Y / (X+Y+Z)$$

$$z = Z / (X+Y+Z)$$

- Wegen $x + y + z = 1$, reicht es die x und y Werte anzugeben. Üblicherweise werden Farben in Form von (x, y, Y) spezifiziert, wobei Y die Helligkeit angibt.



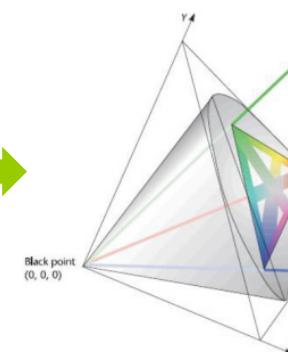
CIE Standard für Farben



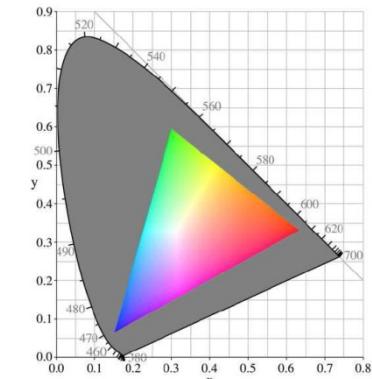
Spektrum $P(\lambda)$

$$\begin{aligned} X &= k \int P(\lambda) \bar{x}(\lambda) d\lambda \\ Y &= k \int P(\lambda) \bar{y}(\lambda) d\lambda \\ Z &= k \int P(\lambda) \bar{z}(\lambda) d\lambda \end{aligned}$$

Grundfarben



Farbraum:
Koordinaten X, Y, Z



normierter Schnitt
durch Farbraum:
Koordinaten x, y

Inverse Transformation:

$$\begin{aligned} X &= Y (x / y) \\ Y &= Y \\ Z &= (1 - x - y) Y / y \end{aligned}$$

Transformation (Zuordnung):

$$x = \frac{X}{(X + Y + Z)} \quad y = \frac{Y}{(X + Y + Z)} \quad z = \frac{Z}{(X + Y + Z)}$$

Bemerkungen:

1. Messinstrumente zur Bestimmung der C.I.E-Werte (des Phosphors):

- **Colorimeter**: misst X , Y und Z
- **Spektroradiometer**: misst X , Y und Z und das Spektrum
- **Fotometer**: misst die Intensität Y .

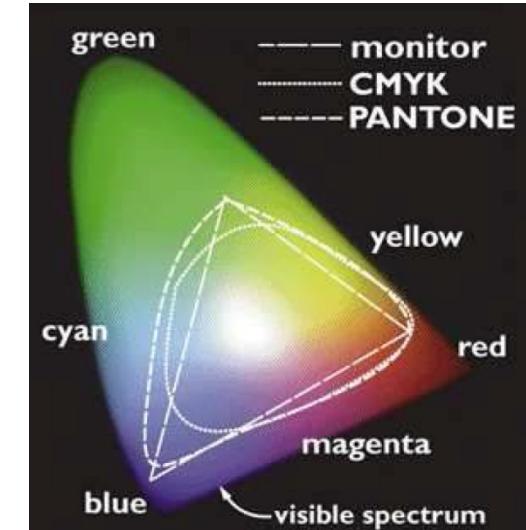
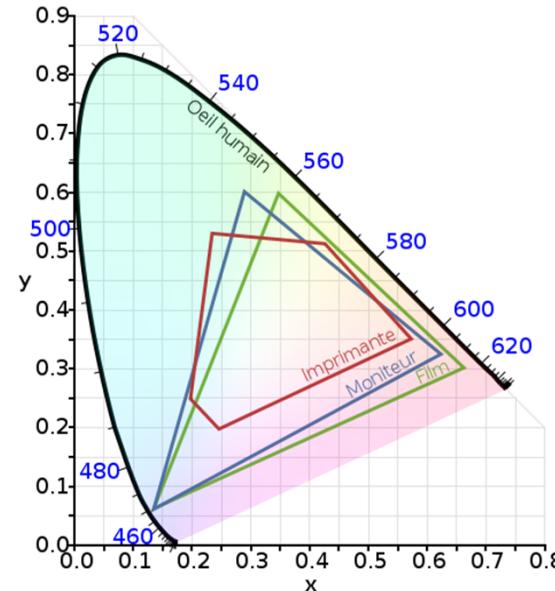
2. In Produktbeschreibungen von Monitoren werden häufig die x -, y -Werte angegeben. Typische Werte sind

	Rot	Grün	Blau
x	0.61	0.29	0.15
y	0.35	0.59	0.063

3. Aus Y und x , y und z lassen sich X , Y und Z berechnen

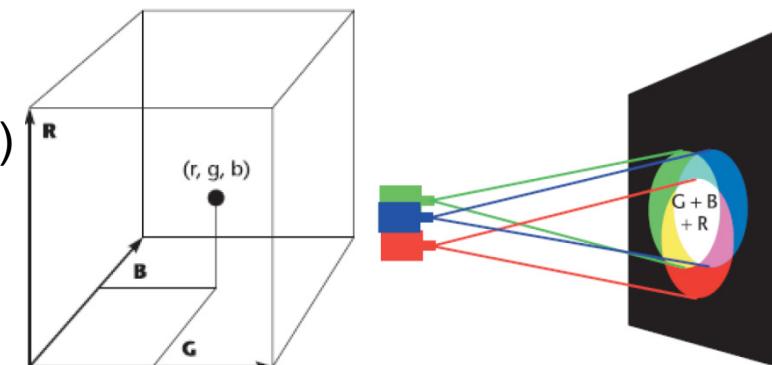
Gamut

- „Hufeisen“: alle wahrnehmbaren Farbe
- Farbpalette (Regionen innerhalb des Hufeisens) bzw. Farben des Farbraums, welches ein Gerät (Sender und Empfänger) kann
- CIE-System



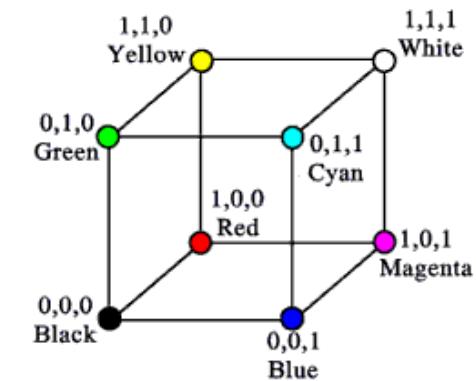
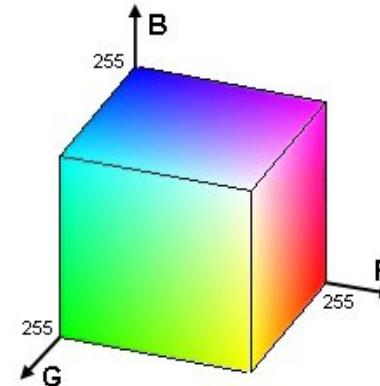
Farbmetrikt

- Eine (Licht-)Farbe C kann mittels der Grundfarben Rot (R), Grün (G) und Blau (B) ausgedrückt werden: $C=rR+gG+bB$
- RGB-Farbmodell



RGB-Farbmodell

- Additive Darstellung von Farben als gewichtete Summe von drei Grundfarben Rot, Grün, Blau



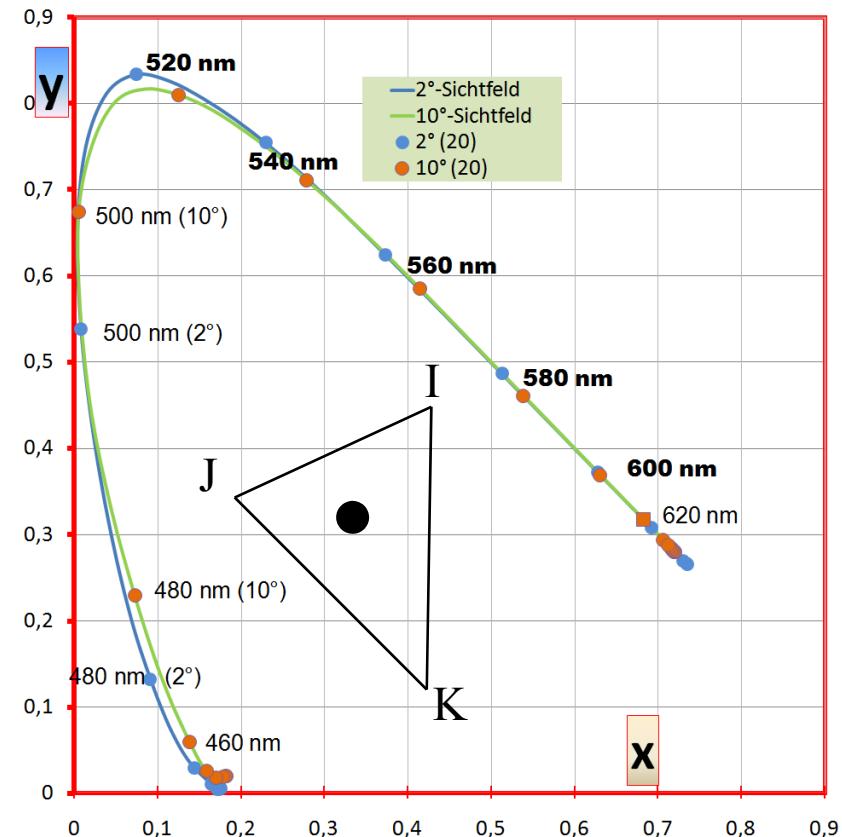
- Meistverwendetes Modell für aktiv lichterzeugende Ausgabemedien (z.B. Displays)
- RGB-Darstellungen in der Praxis unterschiedlich (Kalibrierung)



- Bestimmte Farben nicht im RGB-Modell darstellbar.

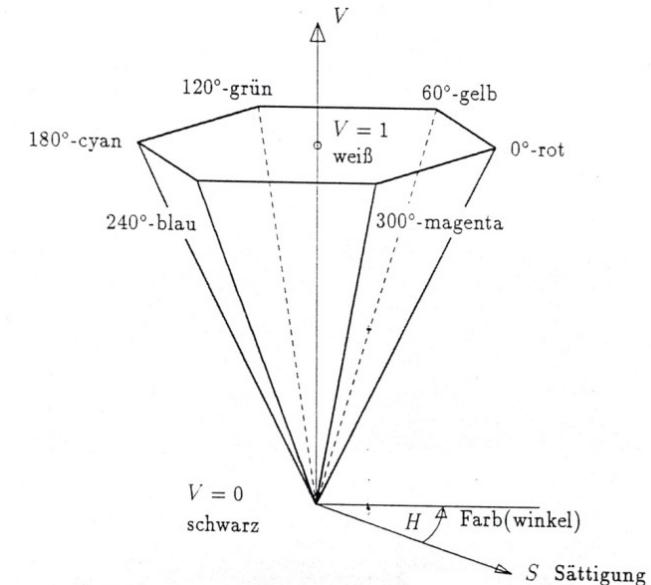
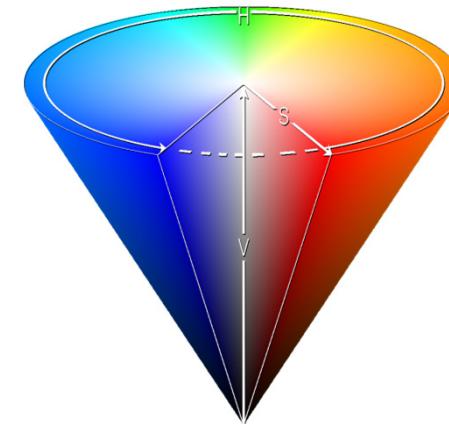
(Additive) Farbmischung

- Jede Farbe auf der Linie (I,J) kann aus I und J gemischt werden.
- Insbesondere können alle Farben aus zwei Spektralfarben gemischt werden.
- Alle Farben innerhalb des Dreiecks können aus I, J, K gemischt werden.
- Daraus folgt eine Einschränkung:
Bei additiver Farbmischung können nicht alle Farben aus drei Farben erzeugt werden.



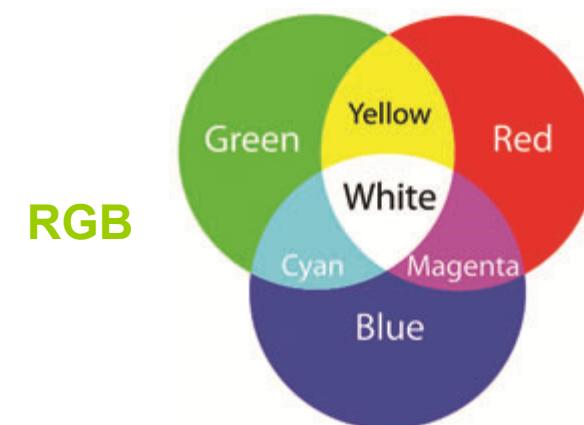
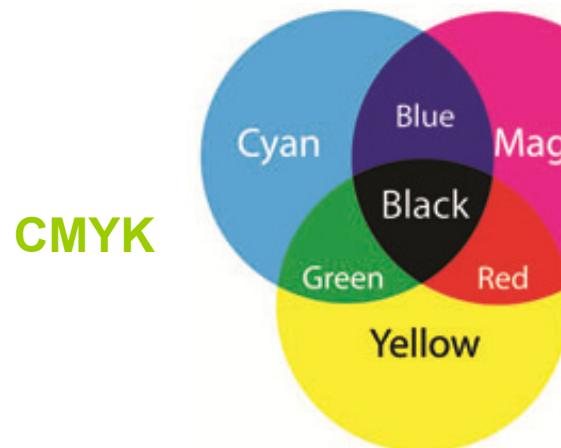
HSV-Farbmodell

- **H=hue=Farbton:** reiner Grundfarbton
Wertebereich: [0,360]
- **S=saturation=Sättigung:**
Weißanteil, der den Reinheitsgrad beeinflusst:
Wertebereich: [0,1]
- **V=value=Wert:** Schwarzanteil, der die Helligkeit beeinflusst
Wertebereich: [0,1]



CMY(K)-Farbmodell

- Subtraktives Modell der Farbe Cyan, Magenta, Yellow und der vierten Komponente „schwarz“ (blacK)
- Meistverwendetes Modell zur Ausgabe auf reflektierenden Ausgabemedien (z.B. Farldrucker)
- Da schwarz aus CMY nur unvollständig und mit viel Farbauftrag erzeugbar zusätzliche Schwarzkomponente
- Anschaulich: Farbfilter subtrahieren Farbwerte



B.1. Einleitung

- B.1.1. Ablauf der Datenvisualisierung
- B.1.2. Bilderzeugungs-Pipeline
- B.1.3. Graphikelemente
- B.1.4. Farbmodelle

B.2. Zweidimensionale Bilderzeugung

- B.2.1. Transformation
- B.2.2. Clipping
- B.2.3. Verrasterung

B.3. Dreidimensionale Bilderzeugung

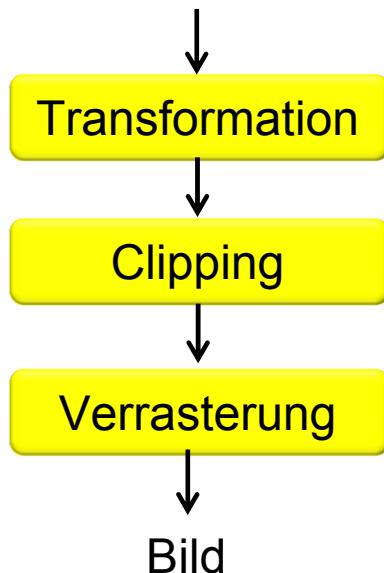
- B.3.1. Transformation
- B.3.2. Beleuchtung
- B.3.3. Projektion
- B.3.4. Sichtbarkeitsberechnung

B.4. Beleuchtungssimulation

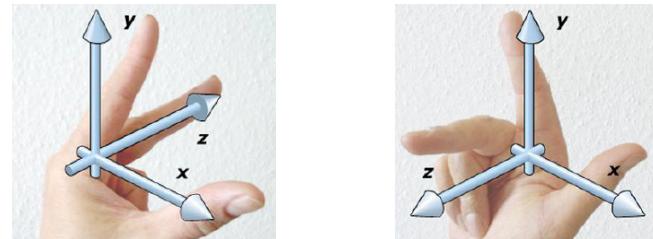
- B.4.1. Strahlverfolgung (Raytracing)
- B.4.2. Strahlungsverfahren (Radiosity)

Aufbau der 2D-Graphik-Pipeline

Graphikelemente
(z.B. Strecken, Dreiecke,
Polygonzüge, Polygone)



Koordinatensysteme:



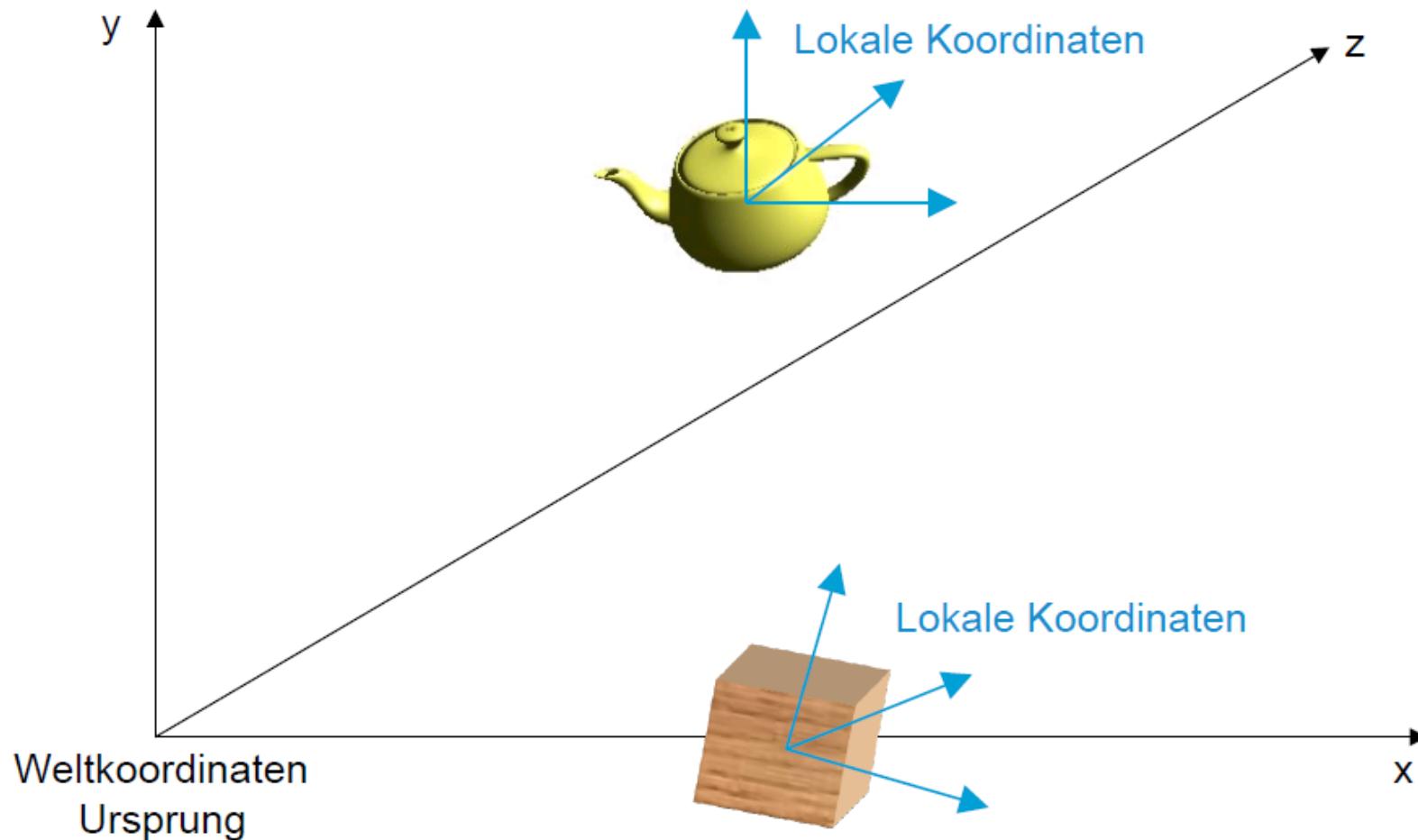
(affine) Transformation: Translation, Rotation, Skalierung, Projektion

Clipping: Abschneiden von Teilszenen, die außerhalb eines gegebenen Bereichs liegen

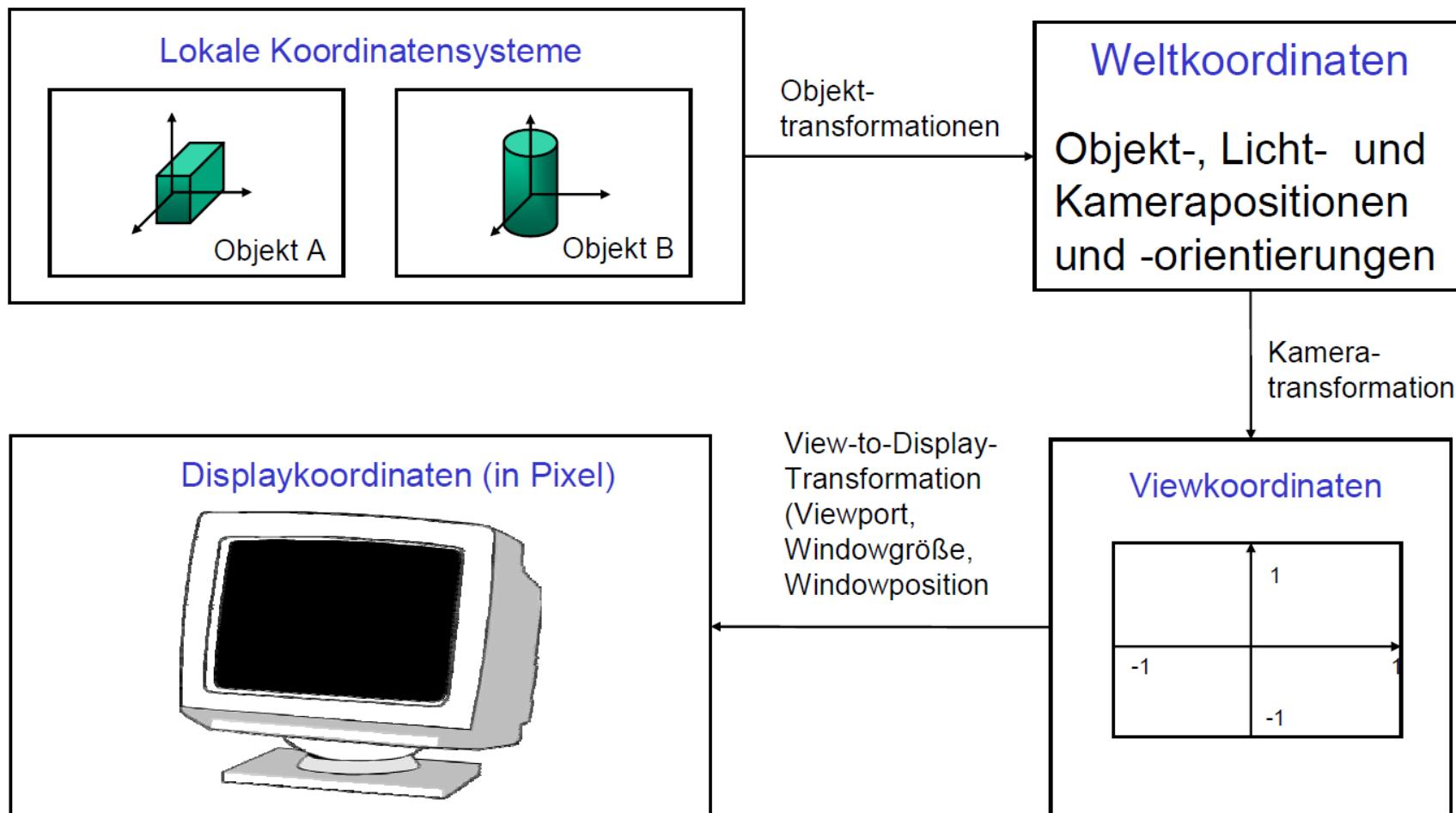
Verrasterung: Erzeugung eines digitalen Bildes aus einer Bildbeschreibung

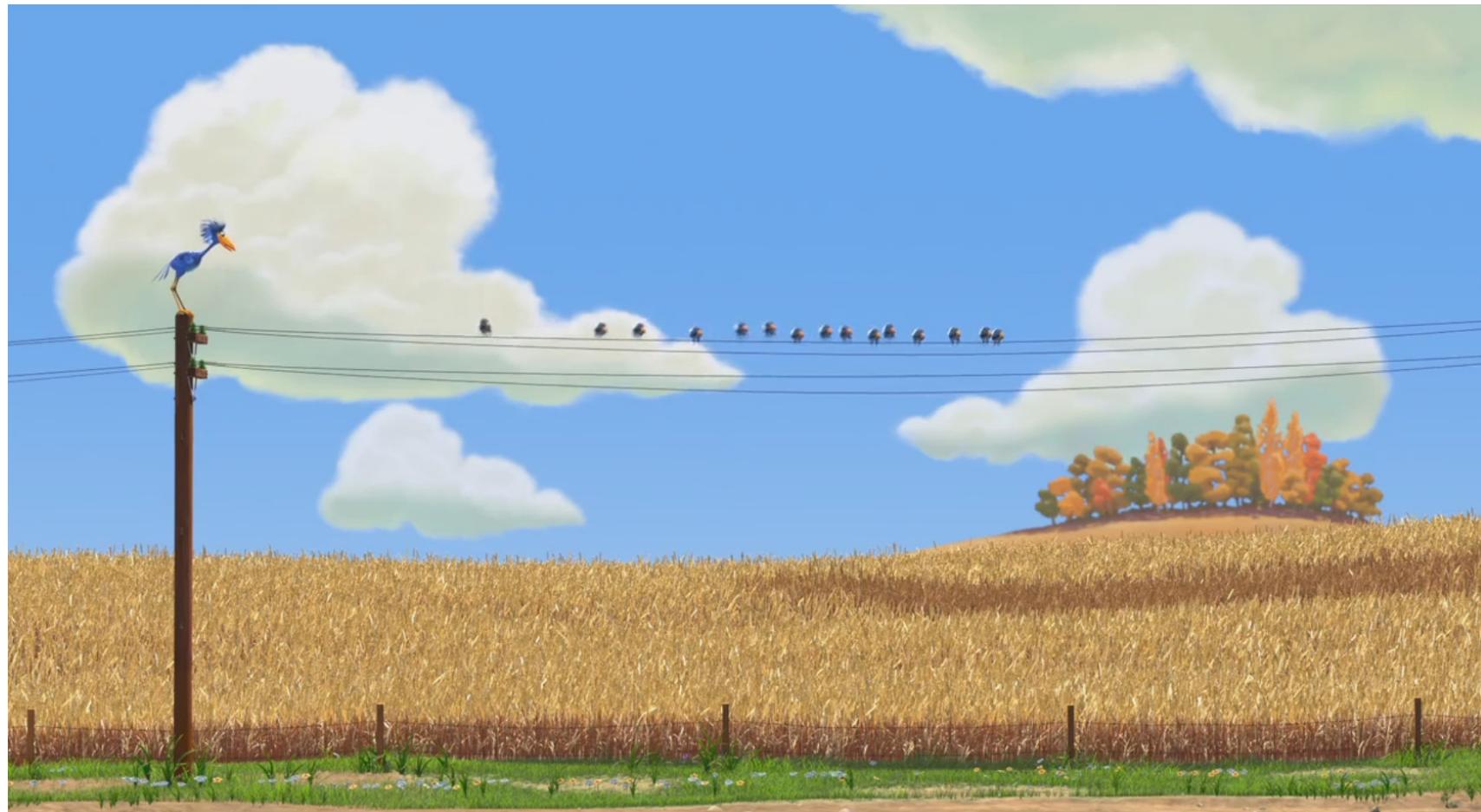
Übersicht zu den Koordinatensystemen

Weltkoordinaten



Übersicht zu den Koordinatensystemen





B.2.1. Transformation

B.2.1.1. Affine Abbildungen

Transformation: üblicherweise durch affine Abbildungen

Affine Abbildung (in der Ebene): $\bar{p} = A \cdot p + t$

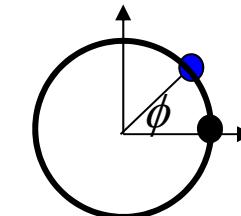
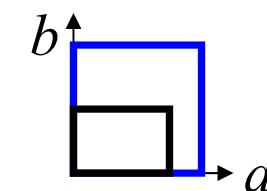
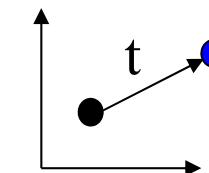
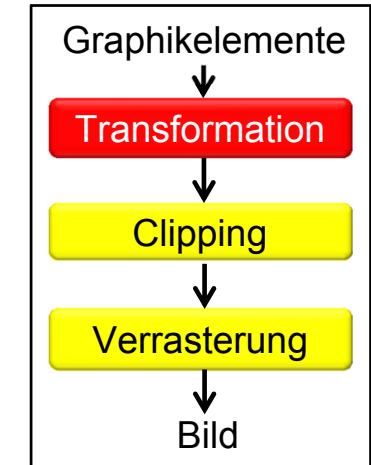
$$\text{mit } A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}, \quad a_{ij} \in \mathbb{R}, \quad t = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}, \quad t_i \in \mathbb{R}$$

Spezialfälle für affine Abbildungen:

- **Translation:** $\bar{p} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot p + t.$

- **Skalierung:** $\bar{p} = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \cdot p, \quad a, b \geq 0.$

- **Rotation:** $\bar{p} = \begin{pmatrix} \cos \phi & -\sin \phi \\ \sin \phi & \cos \phi \end{pmatrix} \cdot p$



B.2.1. Transformation

B.2.1.1. Affine Abbildungen

- Spiegelung

an der waagrechten Achse:

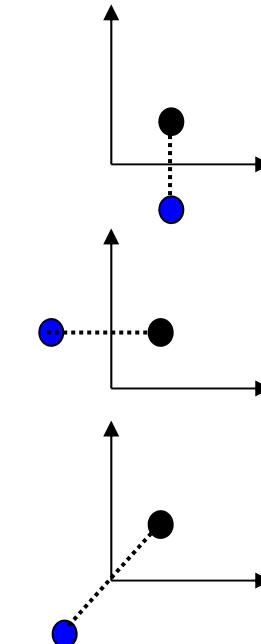
$$\bar{p} = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \cdot p$$

an der senkrechten Achse:

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

am Ursprung:

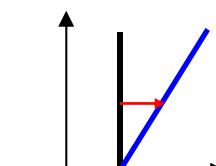
$$\bar{p} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix} \cdot p$$



- Scherung

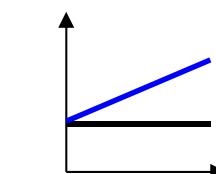
in Richtung der waagrechten Achse:

$$\bar{p} = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \cdot p$$



in Richtung der senkrechten Achse:

$$\bar{p} = \begin{pmatrix} 1 & 0 \\ s & 1 \end{pmatrix} \cdot p$$



Homogene Darstellung affiner Abbildungen:

$$\bar{p}^* = A^* \cdot p^*$$

mit $\bar{p}^* := \begin{pmatrix} \bar{p} \\ 1 \end{pmatrix}$, $p^* := \begin{pmatrix} p \\ 1 \end{pmatrix}$, $A^* := \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix}$,

wobei $\bar{p} = A \cdot p + t$ eine gegebene affine Abbildung ist.

Vorteil der homogenen Darstellung:

Hintereinanderausführung von Abbildungen durch einfache
Matrizenmultiplikation

Hintereinanderausführung von Abbildungen

in *homogener Darstellung* durch einfache Matrizenmultiplikation

$$T_1 : \bar{p} = A_1 \cdot p + t_1,$$

$$T_2 : \bar{p} = A_2 \cdot p + t_2$$

$$\begin{aligned} \rightarrow T_3 := T_2 \circ T_1 : \bar{p} &= A_2 \cdot (A_1 \cdot p + t_1) + t_2 \\ &= A_2 \cdot A_1 \cdot p + A_2 \cdot t_1 + t_2 \\ &= A_3 \cdot p + t_3 \end{aligned}$$

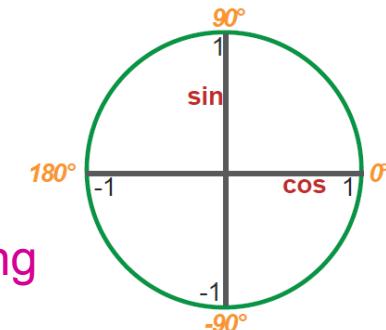
$$\begin{aligned} \text{andererseits: } A_2^* \cdot A_1^* \cdot p^* &= \begin{pmatrix} A_2 \cdot A_1 & A_2 \cdot t_1 + t_2 \\ 0 & 1 \end{pmatrix} \cdot p^* \\ &= \begin{pmatrix} A_3 & t_3 \\ 0 & 1 \end{pmatrix} \cdot p^* \end{aligned}$$

B.2.1. Transformation

B.2.1.3. Transformationsmatrizen

Transformationsmatrix \mathbf{T} =
$$\begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix}$$

Rotation
Skalierung
Translation
Homogene Erweiterung



Rotation:

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

Skalierung:

$$\begin{pmatrix} s1 & 0 & 0 \\ 0 & s2 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Translation:

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

Reflexion an der X-Achse:

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

Reflexion an der Y-Achse:

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

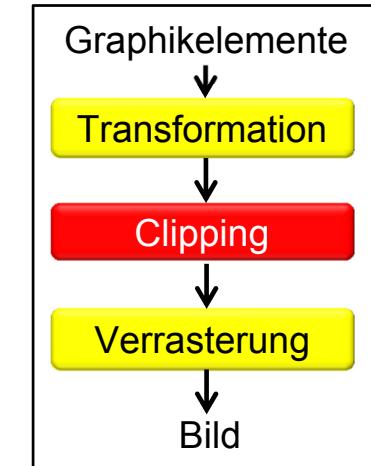
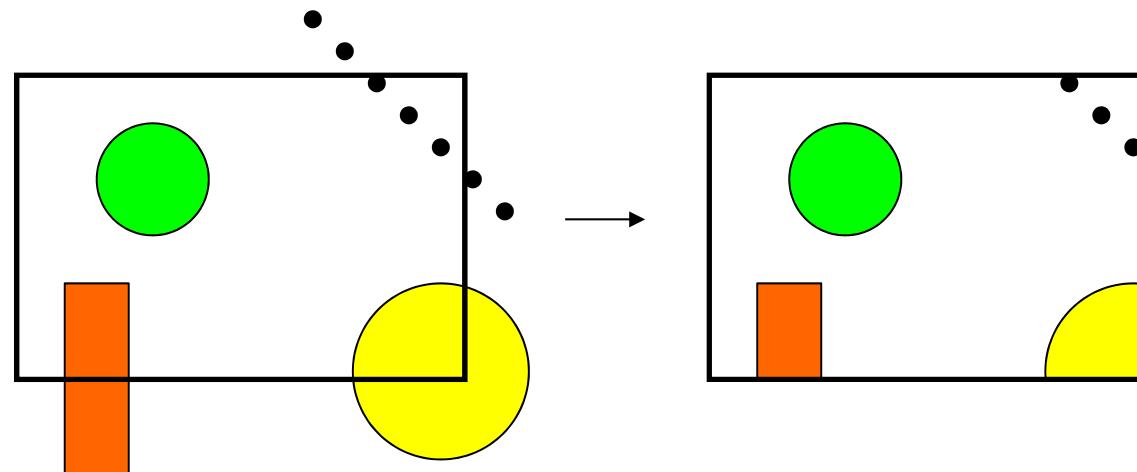
Reflexion an X- und Y-Achse

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

**Hintereinanderausführung von Abbildungen durch
Transformationsmatrizen $\mathbf{T}_1, \mathbf{T}_2, \mathbf{K}, \mathbf{T}_N$:** $\mathbf{p}' = \mathbf{T}_N \cdot \mathbf{K} \cdot \mathbf{T}_2 \cdot \mathbf{T}_1 \cdot \mathbf{p}$

Clipping:

Abschneiden von Teilen einer Szene, die außerhalb eines gegebenen Bereichs liegen.



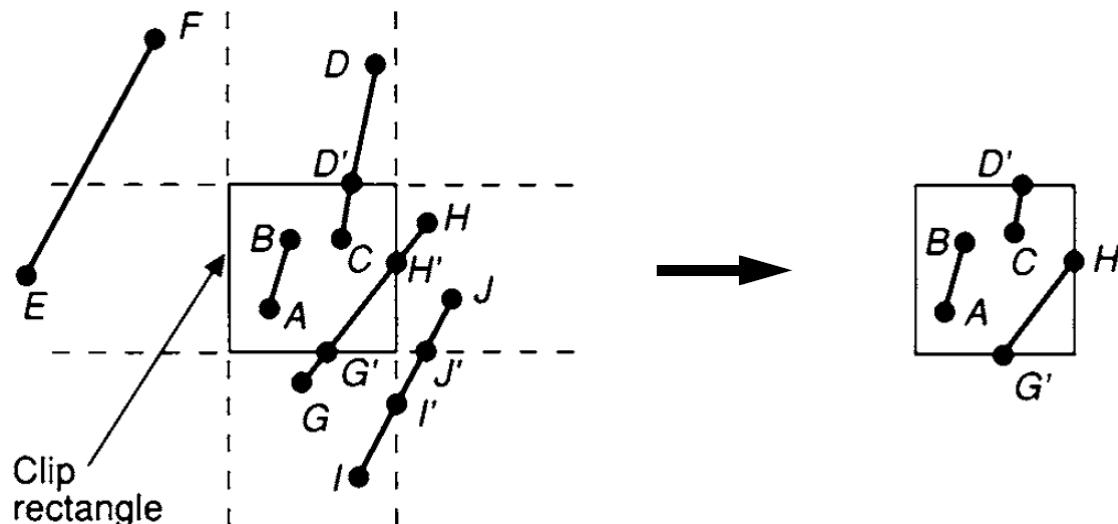
Beispiele:

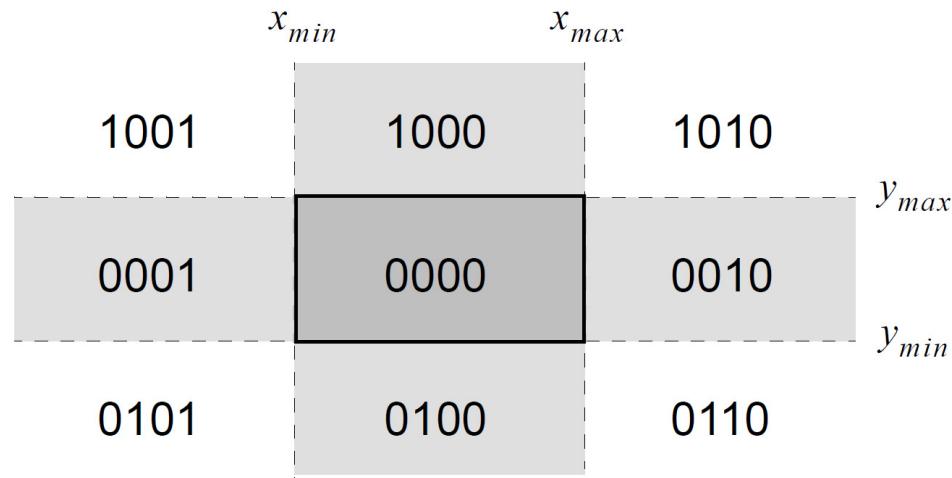
- Punkt-Clipping
- **Strecken-Clipping (z.B. Cohen-Sutherland)**
- Polygon-Clipping (z.B. Sutherland-Hodgman)
- Kreis-Clipping

Strecken-Clipping (z.B. Cohen-Sutherland)

Idee

1. Schritt: Zur Vermeidung der rechenaufwendigen Schnittpunktberechnungen wird versucht, möglichst viele Linien durch einfache Vergleiche zu akzeptieren oder zu eliminieren.
2. Schritt: Sofern Schritt 1 nicht gelingt, wird die Linie in zwei Segmente unterteilt, wobei eines jeweils einfach behandelt werden kann.



Strecken-Clipping (z.B. Cohen-Sutherland):**1. Schritt: Sektionierung der Ebene über 4-Bit-Code mit Vorzeichen****Bedeutung der Bits:**

1. Bit: In der Halbebene über der oberen Kante $y > y_{max}$
2. Bit: In der Halbebene unter der unteren Kante $y < y_{min}$
3. Bit: In der Halbebene rechts der rechten Kante $x > x_{max}$
4. Bit: In der Halbebene links der linken Kante $x < x_{min}$

Strecken-Clipping (z.B. Cohen-Sutherland):

Vorzeichenberechnung der Bits:

1. Bit: $y_{max} - y$
2. Bit: $y - y_{min}$
3. Bit: $x_{max} - x$
4. Bit: $x - x_{min}$

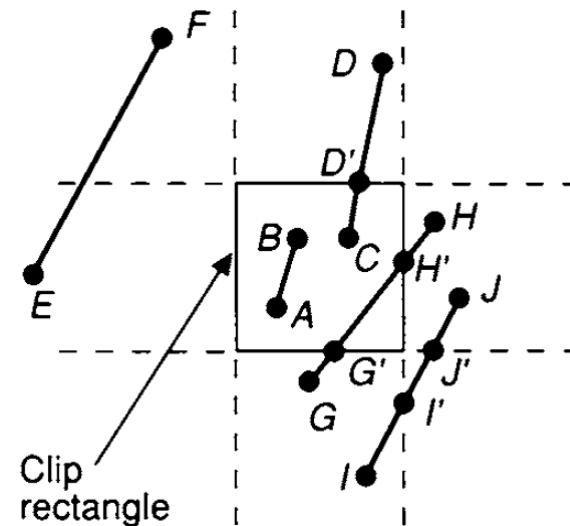
Codeberechnung:

- Für jeden Linienendpunkt Code berechnen
- Anschließend werden Codes beider Linienendpunkte binär **AND**-verknüpft
- Fallunterscheidung:
 - a) Falls $code_1 \wedge code_2 \neq 0$ kann die Linie entfernt werden, weil kein Schnitt mit dem Clipping-Rechteck besteht.
 - b) Für $code_1 \vee code_2 = 0$ liegt die Linie ganz innerhalb des Clipping-Rechtecks.

Strecken-Clipping (z.B. Cohen-Sutherland)

Beispiel zum 1. Schritt:

Ausgangslage

Kodierung nach
dem 1. Schritt

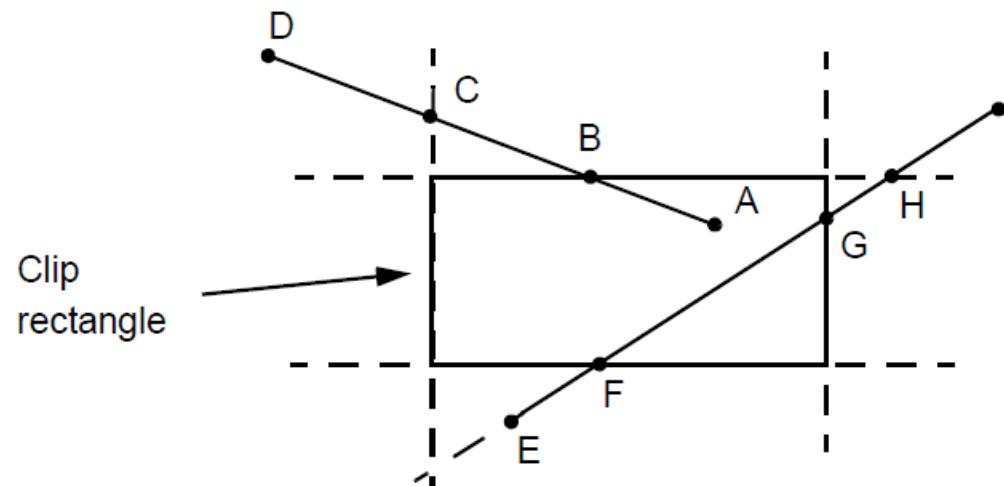
Linie	code ₁	code ₂	code ₁ \wedge code ₂
A → B	0000	0000	0000
C → D	0000	1000	0000
E → F	0001	1001	0001
G → H	0100	0010	0000
I → J	0100	0010	0000

Strecken-Clipping (z.B. Cohen-Sutherland):

2. Schritt: Iterative Subdivision

Anhand der Codes der Linien-Endpunkte und der im 1. Schritt durchgeföhrten Tests Schnitte zwischen den Kanten und dem Clipping-Rechteck bestimmen.

Beispiel zum 2. Schritt:



Beispiel:

Da Punkt *D* durch 1001 kodiert, wird obere und linke Kante geschnitten

Strecken-Clipping (z.B. Cohen-Sutherland):

Verbleibende Linien werden gemäß ihrer Schnittpunkte mit den Rechteckkanten über eine feste Ordnung bei der Interpretation der Bits im Code (z. B. oben → unten → rechts → links) unterteilt.

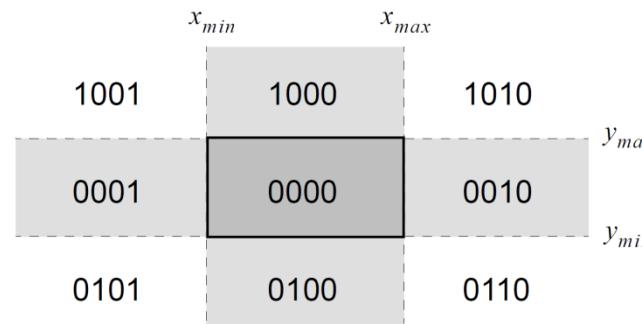
Iteratives Vorgehen, bis die Linie trivial akzeptiert wird:

- 1) Auswahl eines der beiden Punkte der Linie. Der Punkt muss in der äußeren Halbebene einer Rechteckkante liegen.
- 2) Unterteilen der Linie in zwei Segmente an ihrem Schnittpunkt mit der Rechteckkante höchster Priorität
- 3) Eliminieren des Segments Punkt→Schnittpunkt
- 4) Berechnung des Codes für den Schnittpunkt

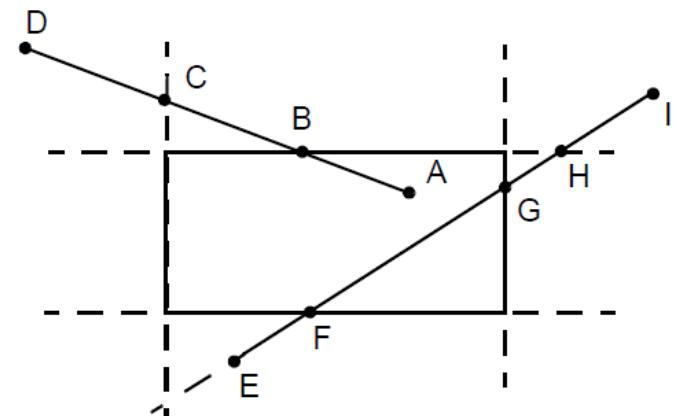
Strecken-Clipping (z.B. Cohen-Sutherland):

Beispiel:

- Punkt D als Startpunkt wählen und wegen der Prioritätskonvention Schnittpunkt B berechnet. $D \rightarrow B$ wird eliminiert und $B \rightarrow A$ akzeptiert.
- Für Linie $E \rightarrow I$ wird zunächst E (0100) ausgewählt und Schnittpunkt $H(0010)$ Restsegment $E \rightarrow H$ berechnet. Für dieses wird E als Startpunkt gewählt und aufgrund der Prioritäten zunächst F berechnet. Schließlich erfolgt die Berechnung der geclippten Linien aus $F \rightarrow H$ durch Teilung in G .



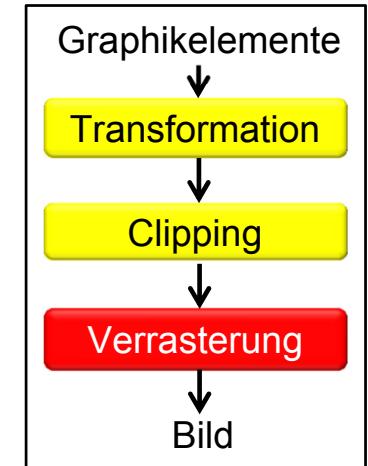
Linie	code_1	code_2	$\text{code}_1 \wedge \text{code}_2$	$\text{code}_1 \vee \text{code}_2$
$D \rightarrow B$	1001	0000	0000	1001
$B \rightarrow A$	0000	0000	0000	0000



Konzept der Verrasterung:

Gegeben: ein kontinuierliches geometrisches Objekt O .

Gesucht: eine Approximation von O durch Rasterpunkte, die bei hinreichend feiner Rasterauflösung wie O aussieht.



Lösungsmöglichkeit

Zwei Schritte:

1. Approximation von O durch Polygonzüge oder Polygone (Dreiecke)
2. Verrasterung von Strecken beziehungsweise Polygonen (Dreiecke)

Verrasterung von Strecken in impliziter Darstellung

Algorithmus

Gegeben: eine Strecke mit Steigung zwischen 0 und 1 sowie mit ganzzahligen Endpunkten \mathbf{p} und \mathbf{q} auf dem Rastergitter (\mathbf{p} links von \mathbf{q}).

Gesucht: Eine Folge von ganzzahligen Punkten, die die Strecke approximieren.

Ablauf:

$$a := q_y - p_y, \quad b := -(q_x - p_x), \quad c := -b p_y - a p_x,$$

$$F(x, y) := a \cdot x + b \cdot y + c.$$

Zeichne (p_x, p_y) ; $j := p_y$;

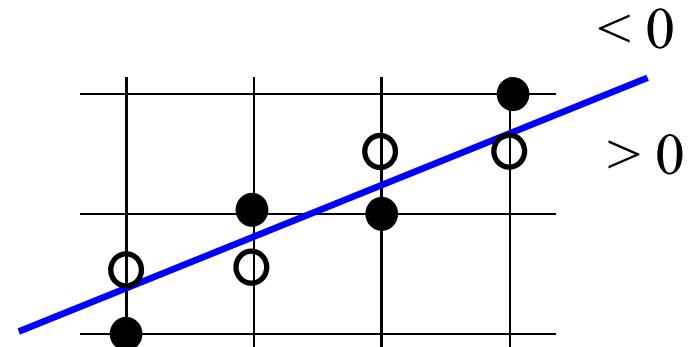
FOR $i := p_x + 1$ **TO** q_x **DO**

BEGIN

IF $F(i, j + 0.5) > 0$ **THEN** $j := j + 1$;

zeichne (i, j)

END.



Bemerkung: Die Testgröße $F_d(i, j) := F(i, j + 0.5)$ kann inkrementell berechnet werden - entsprechende Realisierung des Algorithmus ist unter dem Namen „*Bresenham-Algorithmus*“ bekannt.

Verrasterung von Polygonen

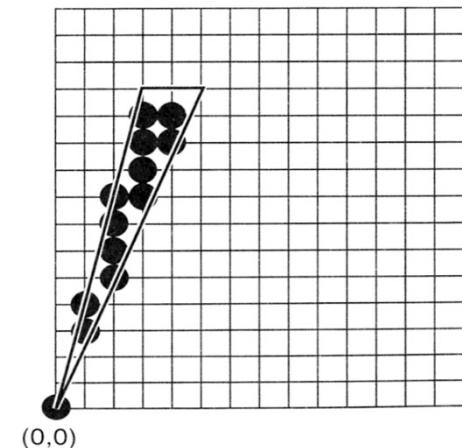
Gegeben: ein Polygon P , ein Punktraster, das P vollständig umfasst.
Die Polygoneckpunkte werden auf dem Raster angenommen.

Gesucht: die Punkte des Rasters, die in P oder auf einer linken oder unteren Kante liegen.

Schwachstelle dieser Definition: Problem bei Spitzen (Alias-Problem).

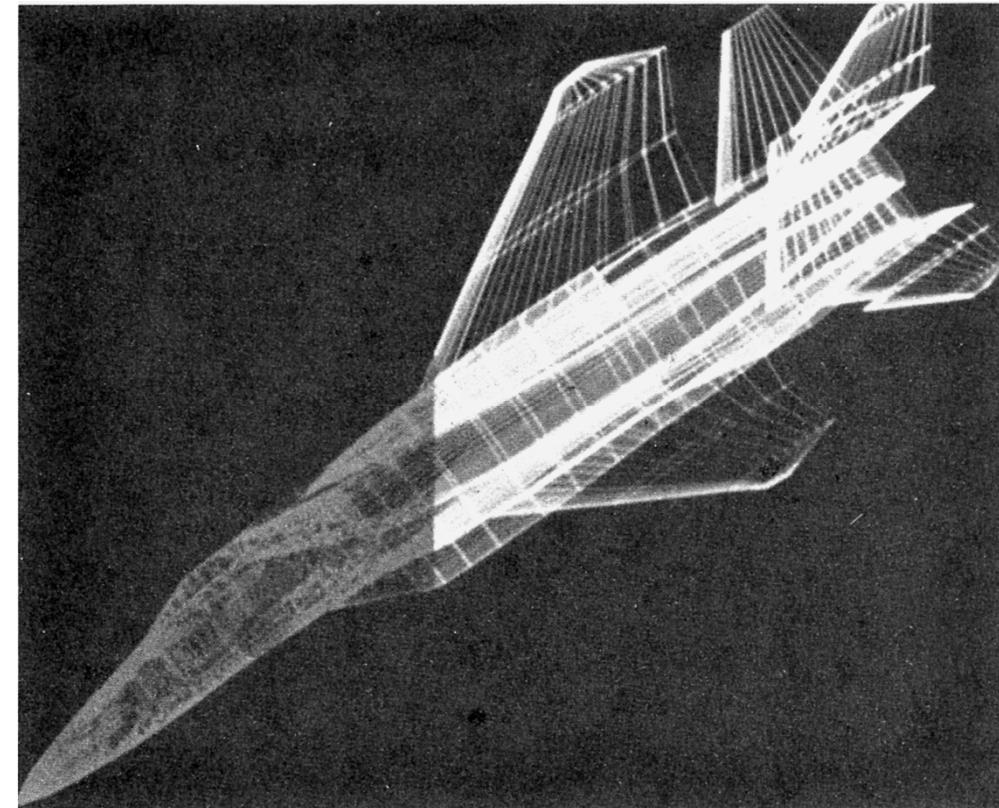
Verfahren zur Polygonverrasterung:

- Scan-Line-Verfahren
(Foley et al., Kap. 3.6, Fellner, Kap. 6.4)
- randbezogenes Füllen
(Fellner, Kap. 6.4)



Alias-Problem: stufiges Aussehen und „Zerfallen“ von verrasterten Strecken;
Grund: nicht korrekte Abtastung

Lösung: bei Rasterdisplays mit mehreren Intensitätsstufen Verwendung der
Intensitätsstufen zur Glättung (**Anti-Aliasing**)



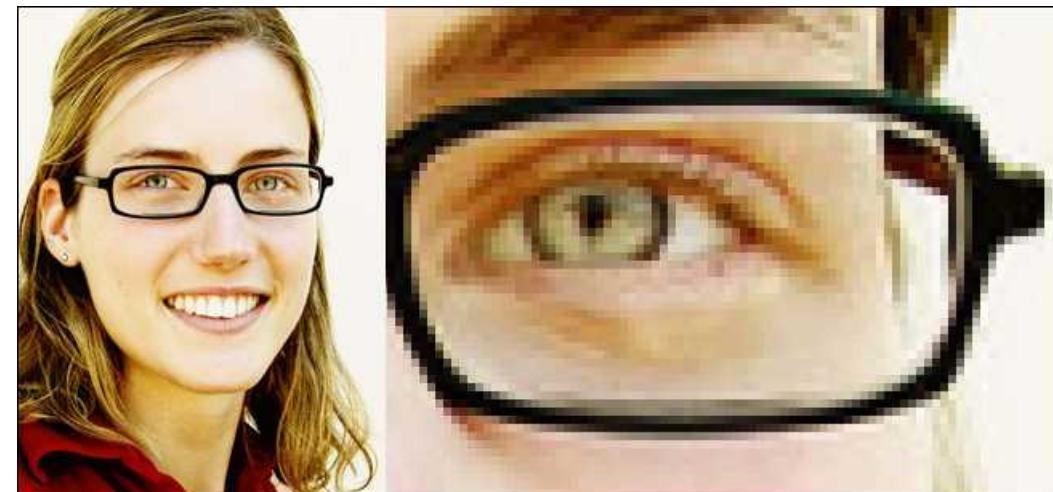
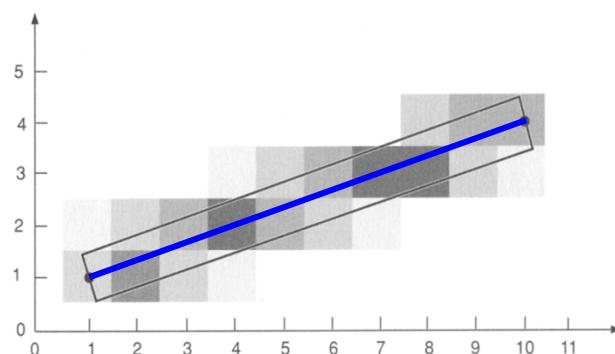
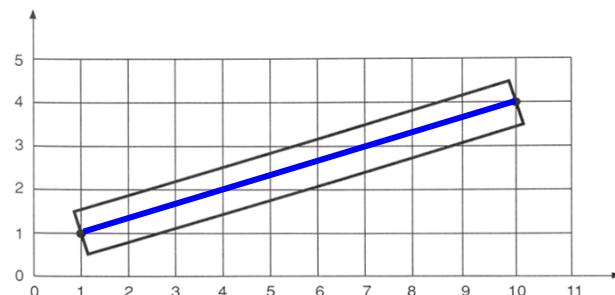
Beispiel:

Geglättete Linienzeichnung: der
vordere Bereich ist nicht geglättet,
der hintere Bereich ist geglättet.

Beispiel für Anti-Aliasing

Glättung durch Verwendung von Graustufen - Helligkeit wächst mit dem Abstand des Pixels von der Strecke. In diesem Beispiel ist die Helligkeit eines Pixels proportional zur Fläche des Schnitts der zum Rechteck vergrößerten Strecke mit dem quadratischen Pixelgebiet.

Entsprechend kann bei Farbdarstellung durch Überblendung von Strecken- und Hintergrundfarbe verfahren werden.



B.1. Einleitung

- B.1.1. Ablauf der Datenvisualisierung
- B.1.2. Bilderzeugungs-Pipeline
- B.1.3. Graphikelemente
- B.1.4. Farbmodelle

B.2. Zweidimensionale Bilderzeugung

- B.2.1. Transformation
- B.2.2. Clipping
- B.2.3. Verrasterung

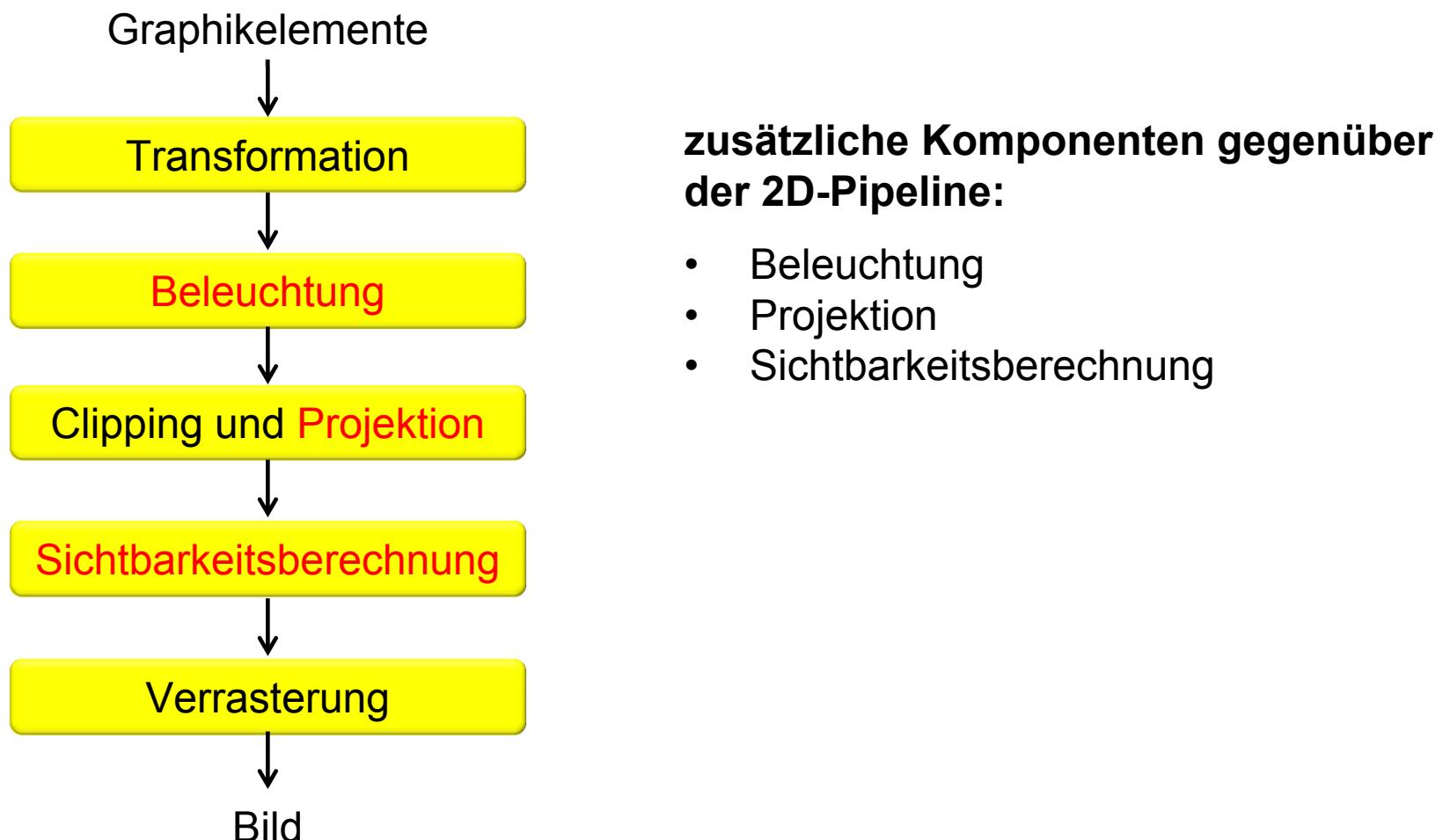
B.3. Dreidimensionale Bilderzeugung

- B.3.1. Transformation
- B.3.2. Beleuchtung
- B.3.3. Projektion
- B.3.4. Sichtbarkeitsberechnung

B.4. Beleuchtungssimulation

- B.4.1. Strahlverfolgung (Raytracing)
 - B.4.2. Strahlungsverfahren (Radiosity)
-

3D-Bilderzeugungs-Pipeline



Affine Abbildung im Raum: $\bar{p} = A \cdot p + t$

$$\text{mit } A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}, \quad a_{ij} \in \mathbb{R}, \quad t = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix}, \quad t_i \in \mathbb{R}$$

Homogene Darstellung affiner Abbildungen: $\bar{p}^* = A^* \cdot p^*$

$$\text{mit } \bar{p}^* := \begin{pmatrix} \bar{p} \\ 1 \end{pmatrix}, \quad p^* := \begin{pmatrix} p \\ 1 \end{pmatrix}, \quad A^* := \begin{pmatrix} A & t \\ 0 & 1 \end{pmatrix},$$

wobei $\bar{p} = A \cdot p + t$ eine gegebene affine Abbildung ist.

Transformationsmatrix $T = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ 0 & 0 & 0 & 1 \end{pmatrix}$

- a b c d Rotation
- e f g h Skalierung
- i j k l Translation
- 0 0 0 1 Homogene Erweiterung

Beleuchtungsmodell = Lichtquellenmodell
 + Reflexions- und Transmissionsmodell (Material)
 + Lichttransportmodell

Beleuchtung: geschieht durch lokale Reflexionsberchnung

Lichtquellen:

- punktförmig
- parallel einfallendes Licht (unendlich ferne Lichtquelle)
- ambientes Licht (ungerichtet)

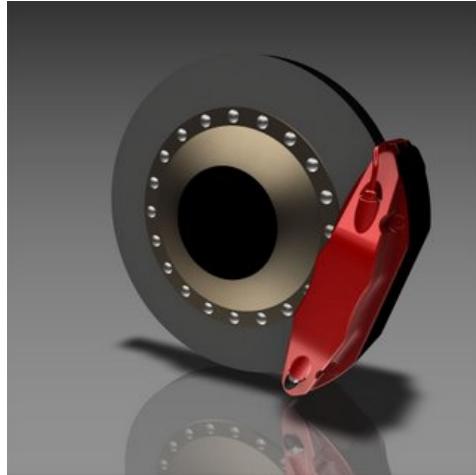
Reflexion: festgelegt durch eine bidirektionale Shading-Funktion (BSF)

Lichttransport:

- Berücksichtigung von Verdeckungen bezüglich des Augenpunkts
- keine Berücksichtigung der Verdeckung von Lichtquellen (Schatten)
- nur Berücksichtigung der einmaligen Reflexion des von einer Lichtquelle einfallenden Lichts am Augenpunkt

B.3.2. Beleuchtung

Einleitung



*Alias Studio
(joshua.maruskadesign)*

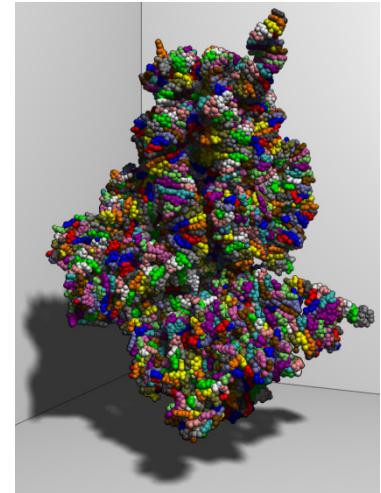


Visual Nature Studio



Open Inventor

GPU-Based Ray-Casting of
Quadratic Surfaces, Symp. on
Point-Based Graphics 06



Beleuchtungsmodelle:

beschreiben die Faktoren, welche die Farbe eines Objektes an einem bestimmten Punkt bestimmen

Lokale Beleuchtungsberechnung:

- Einschränkung: nur Interaktion Lichtquelle mit einem Objekt,
- integriert in das Pipeline-Konzept der Graphikhardware,
- oft heuristisch, approximierend
- einfacher als globale Beleuchtungsberechnung
- **Methoden:** Beleuchtungsmodell nach Phong und polygonales Shading

Globale Beleuchtungsberechnung:

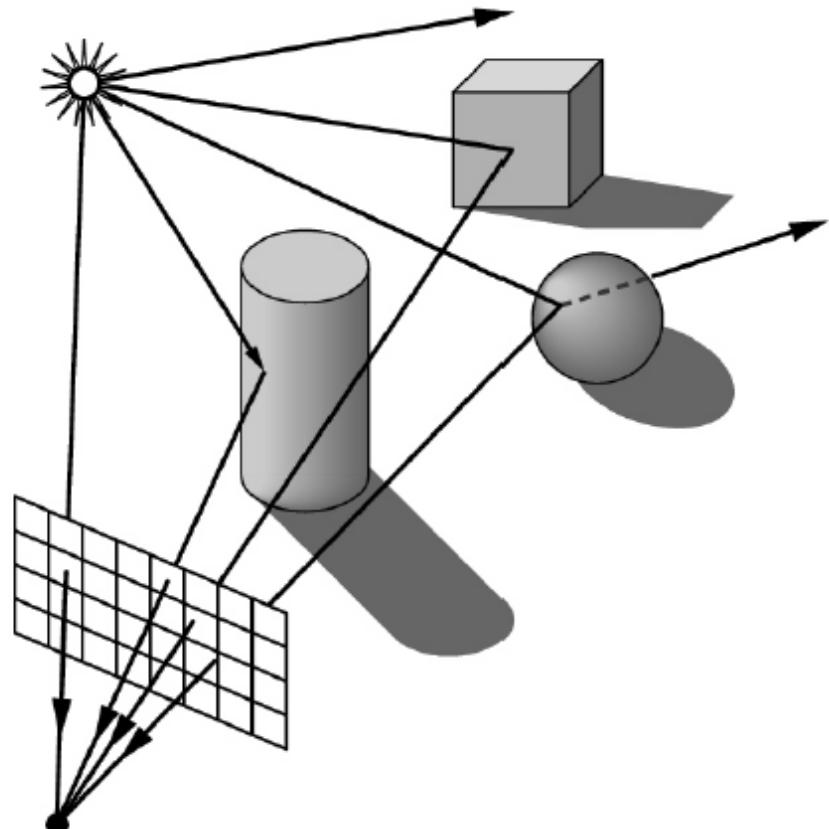
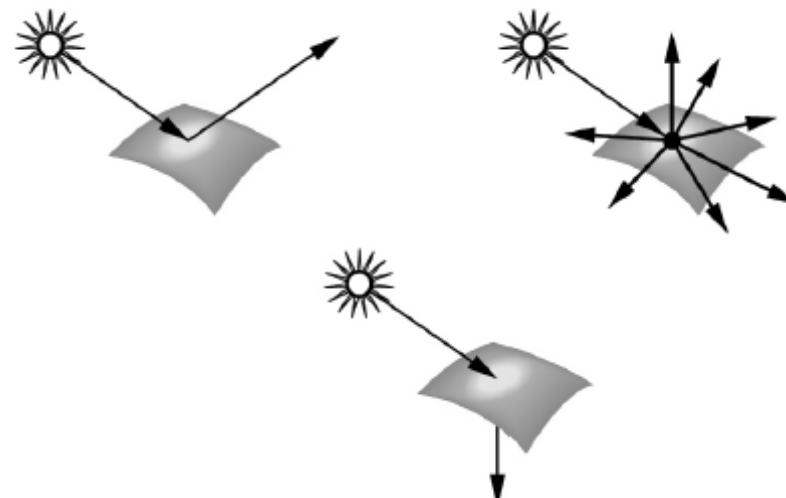
- Einbeziehung aller Objekte einer Szene, um die Farbe an einem Punkt zu bestimmen,
- Interaktion Lichtquelle-Objekt und auch Interaktion Objekt-Objekt,
- oft physikalisch basiert (Erhaltungssätze)
- **Methoden:** Raytracing, Radiosity

Ziel der Beleuchtungsmodelle:

Wechselwirkungen zwischen Licht und Oberflächen beschreiben, um Beleuchtungseffekte in das Rendering zu integrieren.

Auftretende Effekte:

- Lichtstrahlen (gerichtet) reflektiert
- Lichtstrahlen gebrochen
- In Teilen der Szene der Weg zur Lichtquelle blockiert → Schatten



Beleuchtungsmodell von Phong: $I(v) = I_a + I_d + I_g(v)$

I_a : Ambienter Anteil

- Grundhelligkeit in der Szene
- Simuliert Streuung des Lichtes durch Oberflächen (indirekte Beleuchtung)
- Unabhängig von Betrachterstandpunkt und vom einfallendem Licht

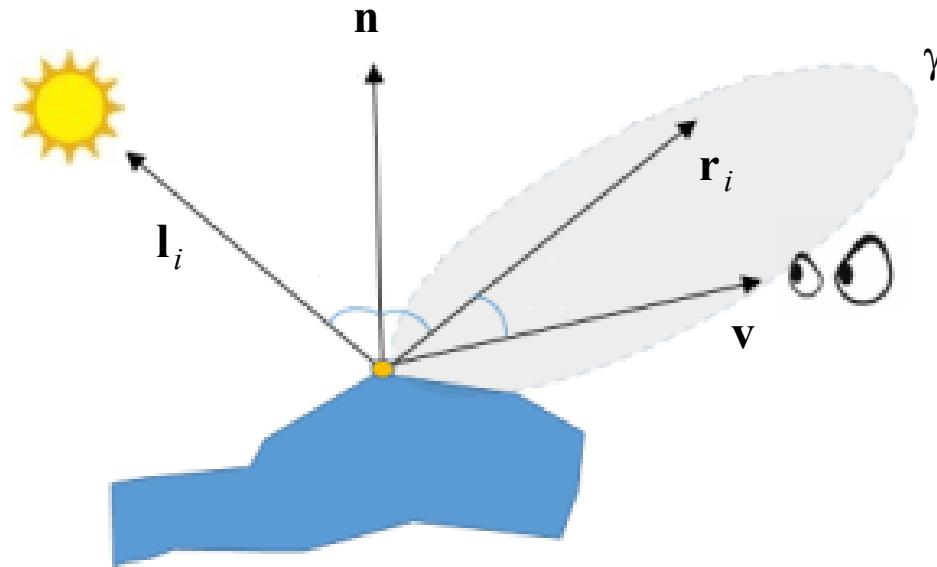
I_d : Diffuser Anteil

- Reflexion an matten Oberflächen
- Gleichmäßig in alle Richtungen
- Unabhängig vom Betrachterstandpunkt

I_g : Glänzender (spekularer) Anteil

- Gerichtete Reflexion an spiegelnden Oberflächen
- Abhängig vom Betrachterstandpunkt
- Erzeugt Glanzpunkte

Beleuchtungsmodell von Phong: $I(v) = I_a + I_d + I_g(v)$



Geometrische Parameter:

- v normierte Blickrichtung
- n auf Länge 1 normierte Oberflächennormale
- l_i auf Länge 1 normierte Richtungsvektor zur i -ten Lichtquelle
- r_i auf Länge 1 normierte Reflexionsvektor zu l_i nach dem Reflexionsgesetz
- $\gamma > 0$ Ausdehnung des Glanzlichtes

B.3.2. Beleuchtung

B.3.2.1. Beleuchtungsmodell

- **ambiente Reflexion**

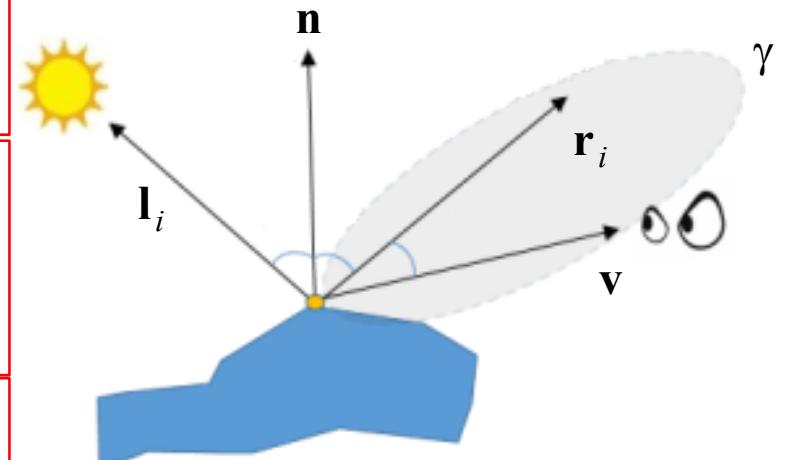
$$\mathbf{I}_a := \mathbf{k}_a \cdot \mathbf{I}_{amb},$$

- **diffuse Reflexion**

$$\mathbf{I}_d := \mathbf{k}_d \sum_{i=1}^l \mathbf{n} * \mathbf{l}_i \cdot \mathbf{I}_i / f(d_i)$$

- **glänzende Reflexion**

$$\mathbf{I}_g := \mathbf{k}_g \sum_{i=1}^l (\mathbf{v} * \mathbf{r}_i)^\gamma \cdot \mathbf{I}_i / f(d_i).$$



$$\mathbf{k}_a, \mathbf{k}_d, \mathbf{k}_g, \gamma > 0$$

Bemerkung:

- \mathbf{I}_i : Intensität der i -ten Lichtquelle.
- „*“: Transponierung des Vektor; Ausdrücke in den Formeln somit Skalarprodukte, ergeben den Kosinus des Winkels zw. jeweils multiplizierten normierten Vektoren.
- Intensitäten/Koeffizienten haben drei Komponenten (*Rot, Grün und Blau*)
- d_i : Entfernung zur i -ten Lichtquelle, $f(\cdot)$: Funktion des Abstands

Beleuchtungsmodell von Phong: $\mathbf{I}(\mathbf{v}) = \mathbf{I}_a + \mathbf{I}_d + \mathbf{I}_g(\mathbf{v})$

$$\mathbf{I}(\mathbf{v}) = \mathbf{k}_a \cdot \mathbf{I}_{amb} \sum_{i=1}^l \mathbf{I}_i / f(d_i) \cdot [\mathbf{k}_d (\mathbf{n} * \mathbf{l}_i) + \mathbf{k}_g (\mathbf{v} * \mathbf{r}_i)^\gamma]$$

Variation des Termint γ zum Glanzlicht



$\gamma = 2$

$\gamma = 5$

$\gamma = 25$

$\gamma = 125$

$\gamma = 625$

Schattenberechnung

Schattenterm S_i in Beleuchtungsformel $\mathbf{I}(\mathbf{v})$ einsetzen:

$$\mathbf{I}(\mathbf{v}) = \mathbf{k}_a \cdot \mathbf{I}_{amb} + \sum_{i=1}^l S_i \cdot \mathbf{l}_i / f(d_i) \cdot [\mathbf{k}_d(\mathbf{n} * \mathbf{l}_i) + \mathbf{k}_g(\mathbf{v} * \mathbf{r}_i)^\gamma]$$

mit $S_i = \begin{cases} 0 & \text{wenn Licht } i \text{ an diesem Punkt blockiert} \\ 1 & \text{sonst} \end{cases}$

Realisierung:

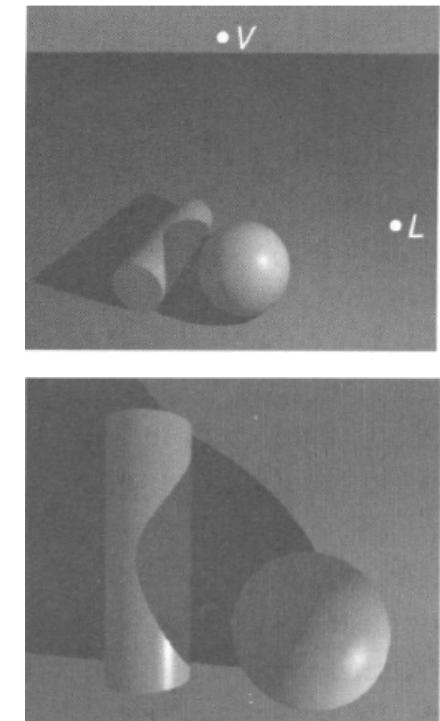
1. Verwendet analytischen Sichtbarkeitsalgorithmus
2. Pixelbasiert (Shadow-Map)
 - Berechnet z-Buffer von Betrachter und Lichtquelle aus

Algorithmus:

Für alle Pixel des Bildes

Bestimme Koordinaten in der Lichtquelle (x,y,z)

Wenn Tiefe größer als Wert im z-Buffer der Lichtquelle
blockiere den Punkt (Schatten)



Beleuchtungsmodell = Lichtquellenmodell
 + Reflexions- und Transmissionsmodell (Material)
 + Lichttransportmodell

Beleuchtungsberechnung für Polygone/Polygonszenen

Ziel: Zuordnung von Farben zu den Pixeln

- Betrachten polygonale 3D-Modelle.
- Verfahren unterscheiden sich darin, an welchen Stellen das Beleuchtungsmodell ausgewertet wird.

Einfache Techniken: interpolative Shading-Verfahren

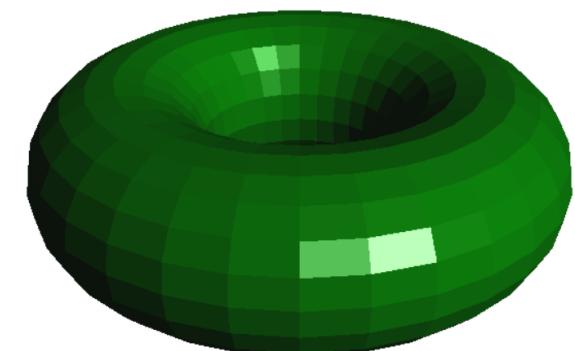
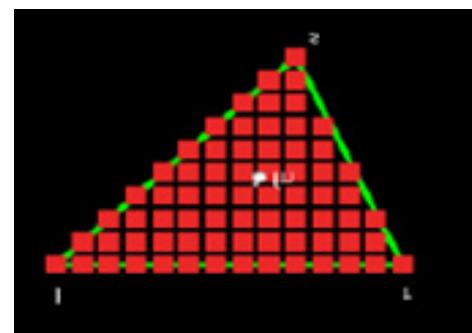
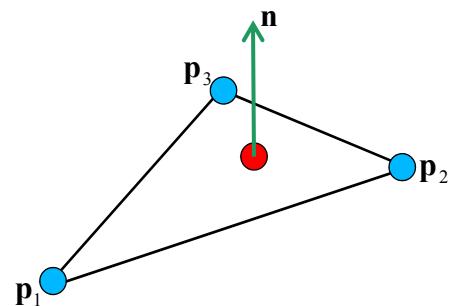
- Über das Beleuchtungsmodell werden Farbwerte an den Eckpunkten berechnet.
- Farbwerte aller Punkte eines Polygons ergeben sich durch lineare Interpolation der Farbwerte an den Eckpunkten.

Verfahren

- Flat-Shading
- Gouraud-Shading
- Phong-Shading

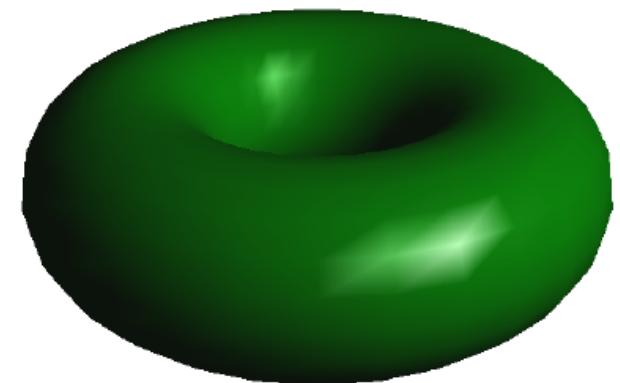
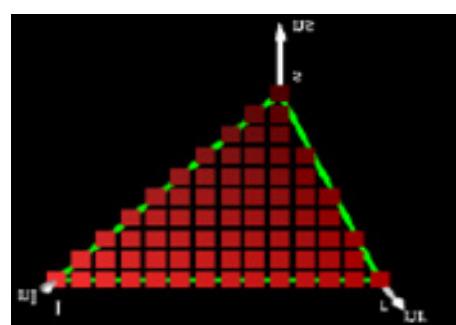
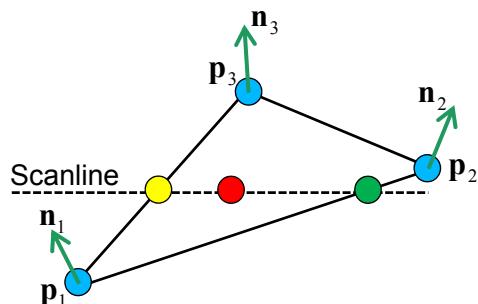
Flat Shading:

- Keine Interpolation, bestimme einen Farbwert pro Polygon
 - Variante 1: Mittelwert aus den Farbwerten an den Eckpunkten, nach dem Beleuchtungsmodell
 - Variante 2: Auswahl eines Punktes und Berechnung der Farbe an dem Punkt
- Die vom Polygon überdeckten sichtbaren Pixel werden mit dem berechneten Farbwert gefärbt.
- Einzelne Polygone deutlich sichtbar
- Einfache Implementierung, sehr schnell, aber geringe Qualität



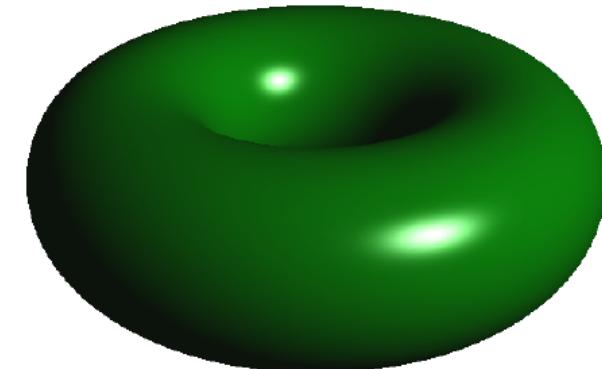
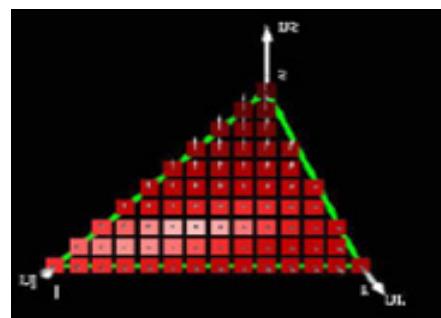
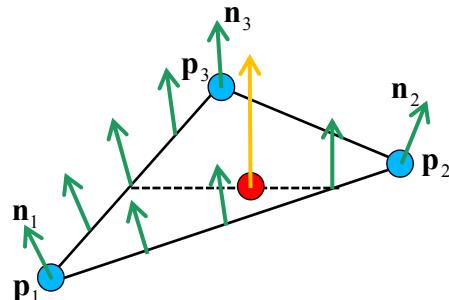
Gouraud-Shading („Smooth Shading“):

- Berechnen der Farbwerte an den Ecken des Polygons nach dem Beleuchtungsmodell
- Lineare Interpolation der Farbwerte von den Ecken in das Innere des Polygons
- Aufwändiger als Flat-Shading, aber qualitativ besser
- Kanten zwischen Polygonen nicht mehr sichtbar
- Problem: Glanzpunkt in der Mitte eines Polygons



Phong-Shading:

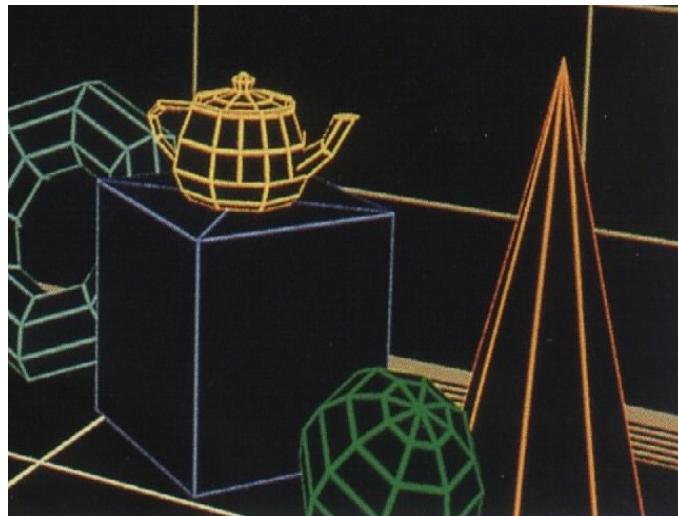
- Material- und Normalenattribute für die Polygone-Eckpunkte
- Interpoliere die Normalenvektoren der Eckpunkte des Polygons über die Fläche des Polygons oder Schätzungen davon
- Berechne an jedem Punkt (Pixel) im Inneren des Polygons einen Farbwert aus der interpolierten Normalen
- Aufwendiger als Gouraud-Shading, aber die besten Ergebnisse
- Glanzpunkte im Inneren eines Polygons werden korrekt dargestellt
- Aber: auch einige Fälle, die nicht dargestellt werden können



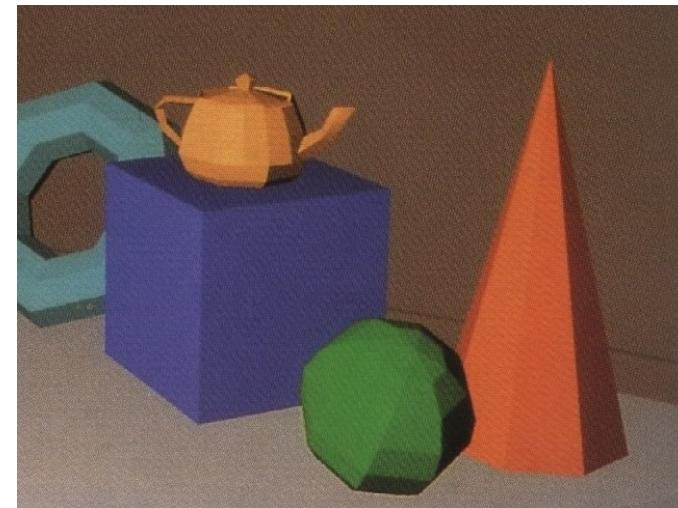
B.3.2. Beleuchtung

B.3.2.2. Shading-Modelle

Beispiele:



Originalszene aus glatten Objekten



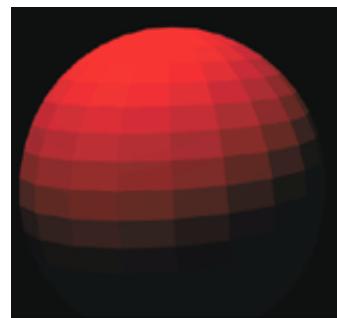
Flat Shading der polygonapprox. Objekte



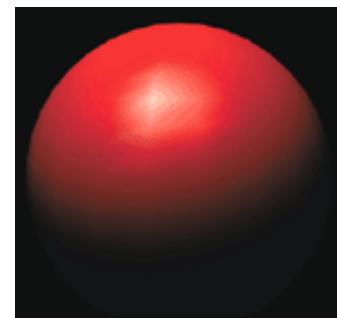
diffuses Smooth Shading der Polygonappr. glänzendes Smooth Shading der Polygonappr.



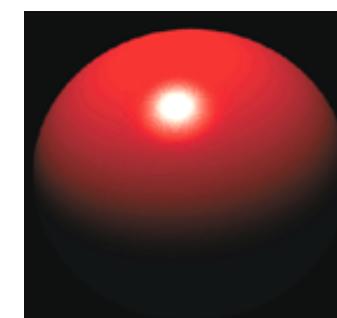
Vergleich zwischen Flat-, Gouraud- und Phong-Shading



Flat

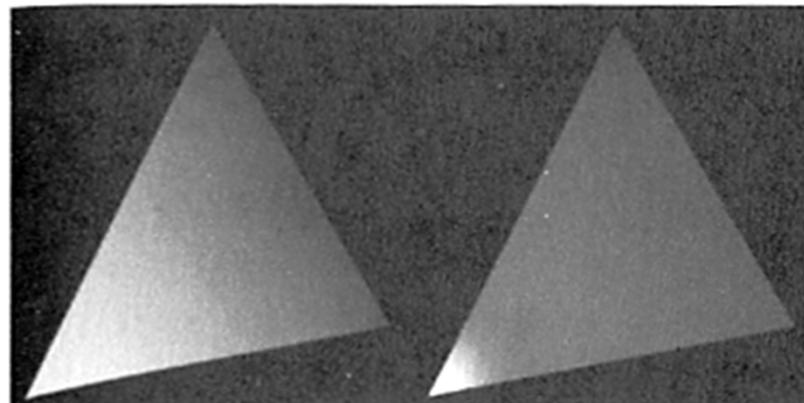


Gouraud



Phong

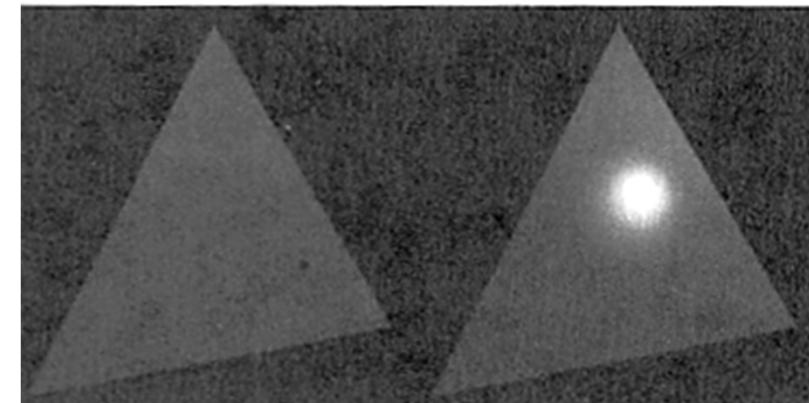
Glanzreflektion mit Gouraud-Shading und Phong-Shading



Gouraud-Shading

Phong-Shading

Glanzlicht am linken Knoten



Gouraud-Shading

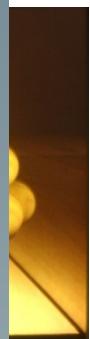
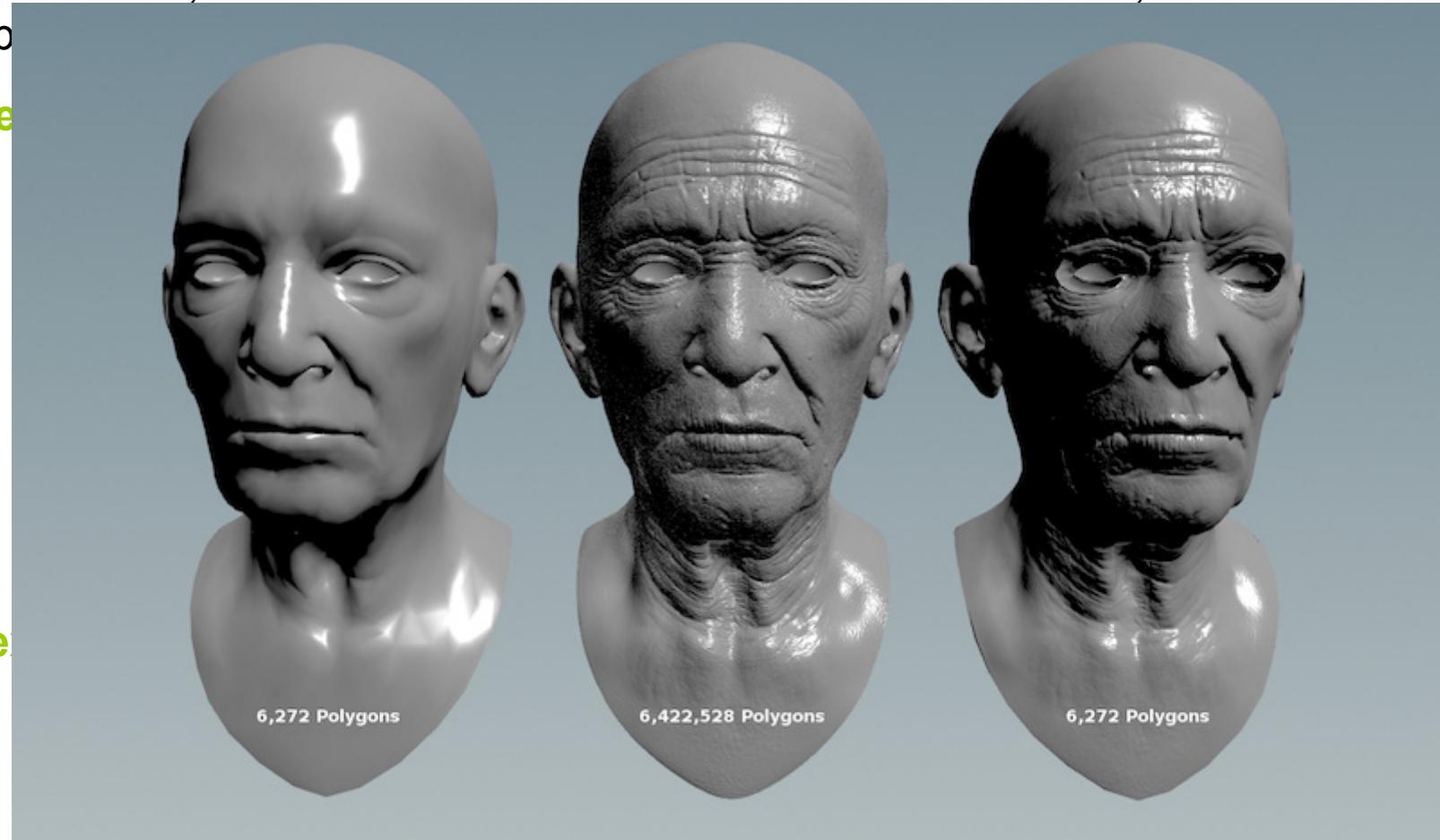
Phong-Shading

Glanzlicht im Inneren des Dreiecks

Textur in der Computergraphik:

Feinstruktur, die durch Funktionen beschrieben wird, welche die Ab-

Be



Farbe



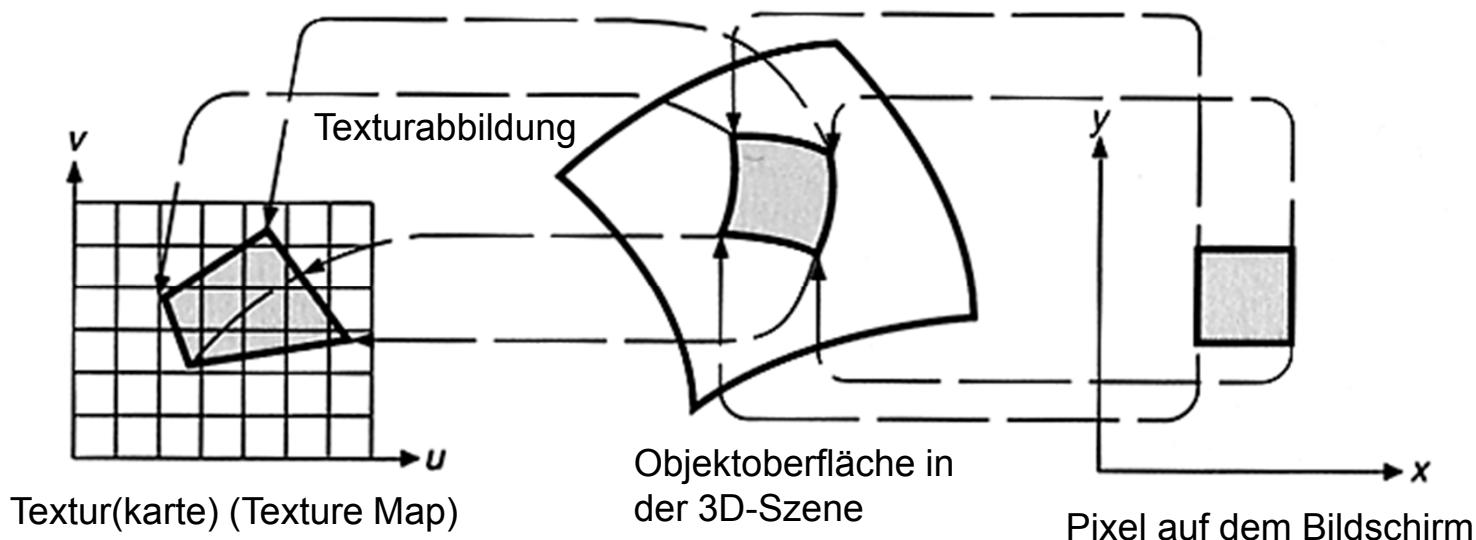
Vektors

Te

-
-
-

Voraussetzung: Raumkoordinatensystem

Texture Mapping (abgebildete Textur)



Die Textur wird als Rastermatrix (Texturkarte) vorgegeben, die z.B. durch Digitalisieren natürlicher Texturen gewonnen wurde.

Vorteil: natürliches Aussehen

Nachteil: Anti-Alias-Maßnahmen sind erforderlich; Bsp.: Mip-Mapping

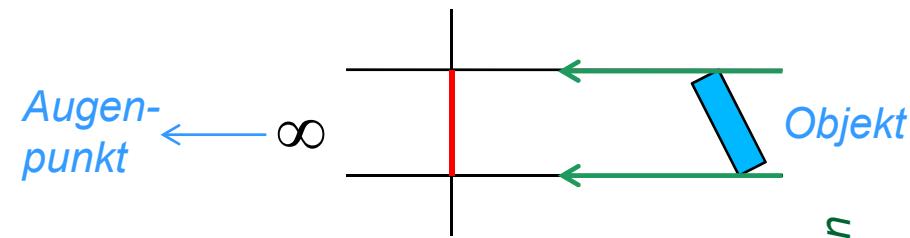
3D-Bilderzeugungs-Pipeline



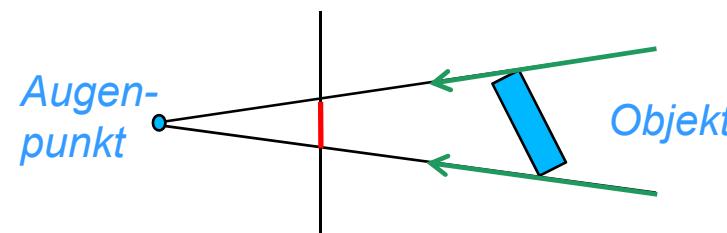
Projektion:

Abbildung aus einem Raum der Dimension n (hier: 3D) in einen Raum der Dimension $m < n$ (hier: 2D):

- **Parallelprojektion**



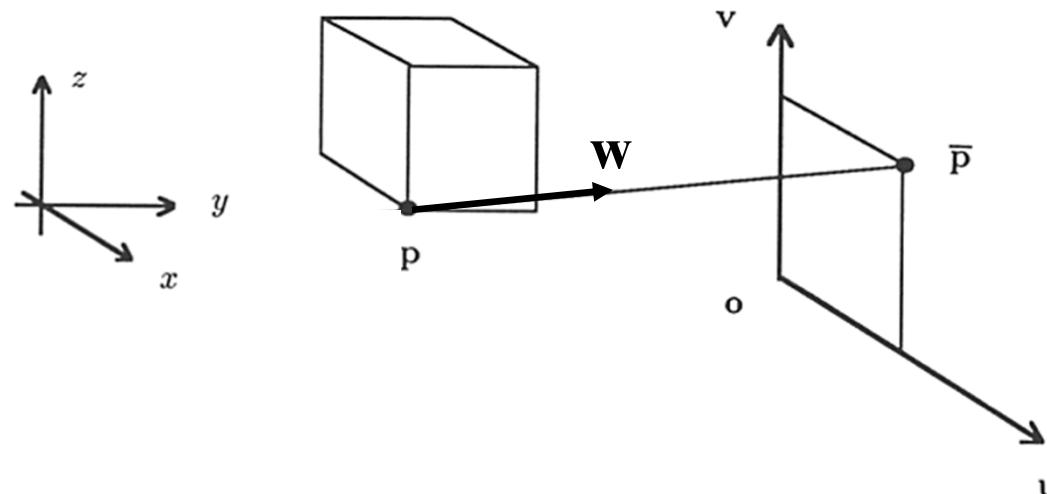
- **perspektivische Projektion**



Gegeben: eine Bildebene, spezifiziert durch einen Ursprung $\mathbf{o} \in \mathbb{R}^3$ und zwei Koordinatenvektoren \mathbf{u} und \mathbf{v} .

Parallelprojektion $\bar{\mathbf{p}} = (\bar{p}_x, \bar{p}_y)$ eines Punktes \mathbf{p} bezüglich einer Richtung $\mathbf{w} \in \mathbb{R}^3$:

$$\bar{p}_x = \frac{|\mathbf{p} - \mathbf{o} \quad \mathbf{v} \quad \mathbf{w}|}{|\mathbf{u} \quad \mathbf{v} \quad \mathbf{w}|}, \quad \bar{p}_y = \frac{|\mathbf{u} \quad \mathbf{p} - \mathbf{o} \quad \mathbf{w}|}{|\mathbf{u} \quad \mathbf{v} \quad \mathbf{w}|}.$$



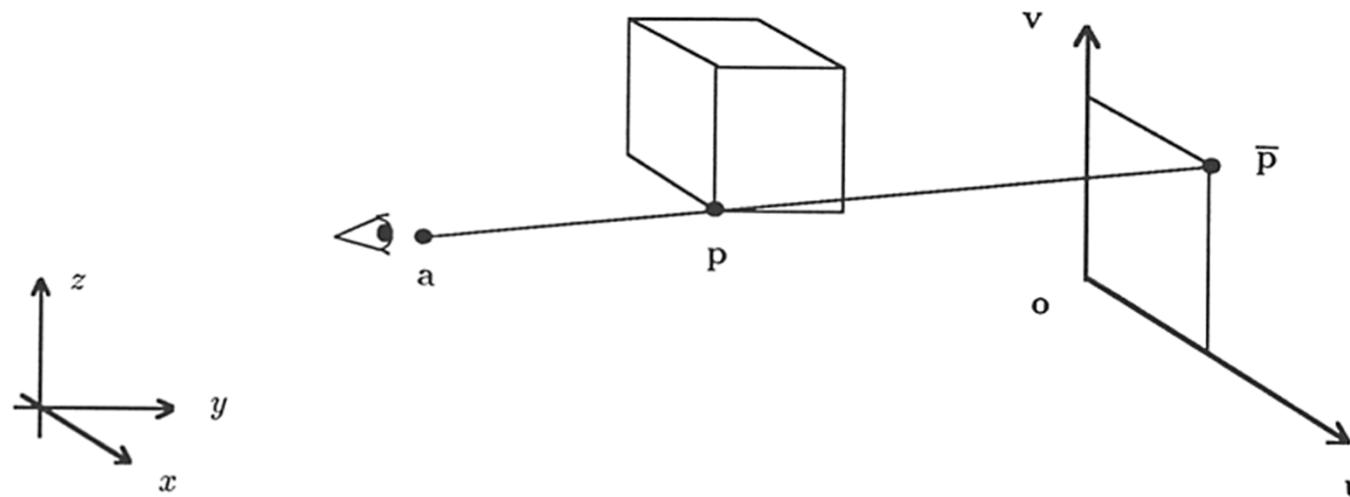
Bemerkung:

| | bezeichnet die Determinante.

perspektivische Projektion $\bar{p} = (\bar{p}_x, \bar{p}_y)$ eines Punktes p bezüglich eines Augenpunktes $a \in \mathbb{R}^3$:

$$\bar{p}_x = \frac{|\begin{matrix} a - o & v & a - p \end{matrix}|}{|\begin{matrix} u & v & a - p \end{matrix}|},$$

$$\bar{p}_y = \frac{|\begin{matrix} u & a - o & a - p \end{matrix}|}{|\begin{matrix} u & v & a - p \end{matrix}|}.$$



Herleitung der perspektivischen Transformation:

1. Gleichung für den Schnitt des Strahls vom Augenpunkt \mathbf{a} durch den zu projizierenden Punkt \mathbf{p} mit der Bildebene, die durch den Ursprungspunkt \mathbf{o} und zwei Koordinatenvektoren \mathbf{u} und \mathbf{v} gegeben ist:

$$\mathbf{a} + \bar{p}_z(\mathbf{p} - \mathbf{a}) = \mathbf{o} + \bar{p}_x\mathbf{u} + \bar{p}_y\mathbf{v}.$$

2. Umordnen liefert:

$$\bar{p}_x\mathbf{u} + \bar{p}_y\mathbf{v} + \bar{p}_z(\mathbf{a} - \mathbf{p}) = \mathbf{a} - \mathbf{o}.$$

3. Dies sind drei Gleichungen mit drei Unbekannten. Durch Anwenden der Cramerschen Regel ergibt sich folgende Lösung:

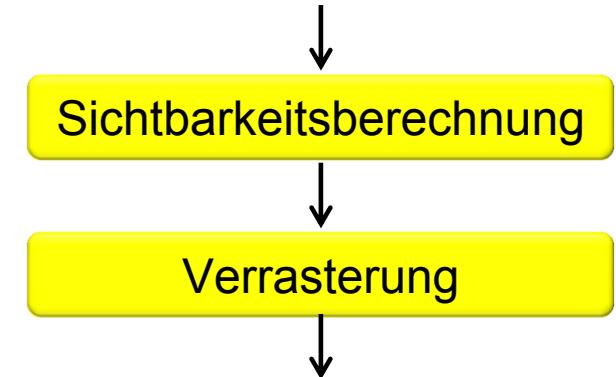
$$\bar{p}_x = \frac{|\mathbf{a} - \mathbf{o} \quad \mathbf{v} \quad \mathbf{a} - \mathbf{p}|}{|\mathbf{u} \quad \mathbf{v} \quad \mathbf{a} - \mathbf{p}|}, \quad \bar{p}_y = \frac{|\mathbf{u} \quad \mathbf{a} - \mathbf{o} \quad \mathbf{a} - \mathbf{p}|}{|\mathbf{u} \quad \mathbf{v} \quad \mathbf{a} - \mathbf{p}|}, \quad \bar{p}_z = \frac{|\mathbf{u} \quad \mathbf{v} \quad \mathbf{a} - \mathbf{o}|}{|\mathbf{u} \quad \mathbf{v} \quad \mathbf{a} - \mathbf{p}|}.$$

3D-Bilderzeugungs-Pipeline



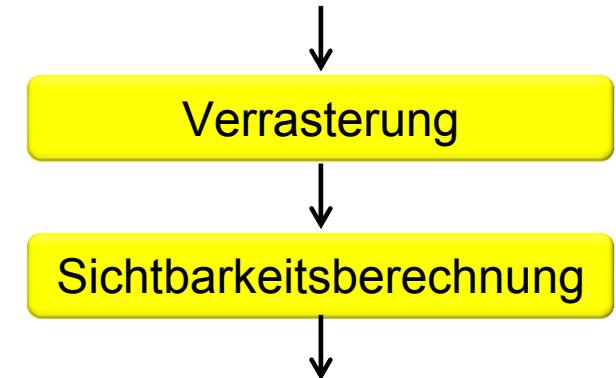
Szenenbasierte Lösungsverfahren:

- Painter's Algorithm
 - Binäre Raumzerlegung
(Binary Space Partitioning)
- } Prioritätslistenverfahren



Rasterbildbasierte Lösungsverfahren:

- Tiefenpufferverfahren (z-Buffer)
- Scan-Line-Verfahren
- Bereichsunterteilungsverfahren

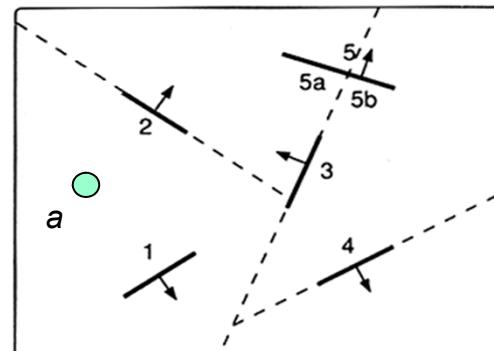


Binäre Raumzerlegung

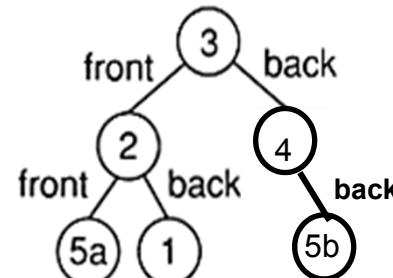
Aufbau des BSP-Suchbaums: Nimm ein Polygon der Szene und teile die Szene durch seine Ebene in die beiden Teilszenen, die aus den Polygonen in den beiden durch die Ebene induzierten Teilräume liegen. Polygone, welche die Ebene schneiden, werden durch die Ebene in Teilpolygone zerlegt. Das ausgewählte Polygon bildet die Wurzel des Baums. Sie besitzt zwei Teilbäume, deren Wurzeln entsprechend aus den beiden Teilszenen bestimmt werden.

Sichtbarkeitsberechnungsphase: Durchlaufe den BSP-Baum

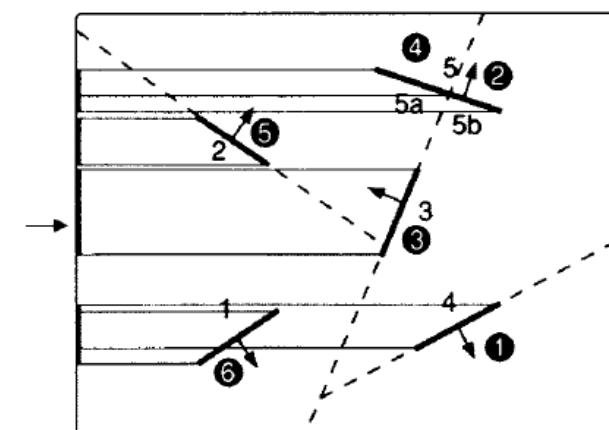
Beispiel:



Aufsicht der Szene mit
Polygon 3 als Root-Polygon



BSP-Baum

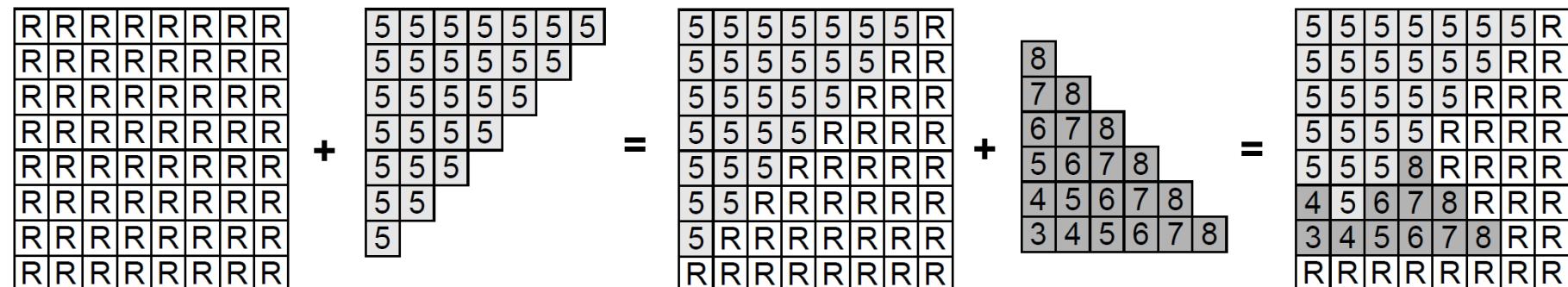


BSP-Baumtraversierung

Tiefenpufferverfahren

Die Polygone der Szene werden nacheinander auf die Bildebene projiziert und verrastert. Für jeden Bildpunkt eines Polygons wird seine Entfernung zum Augenpunkt bestimmt (die Tiefe) und mit einem für den Bildpunkt bereits gespeicherten Tiefenwert verglichen. Ist seine Tiefe geringer, wird diese dem Bildpunkt zugewiesen und dieses Polygon als an diesem Bildpunkt sichtbar registriert. Die Sichtbarkeitsinformation bzw. die Tiefeninformation werden in einem Feld (Array) mit der Bildauflösung entsprechenden Größe abgelegt.

Beispiel:



B.1. Einleitung

- B.1.1. Ablauf der Datenvisualisierung
- B.1.2. Bilderzeugungs-Pipeline
- B.1.3. Graphikelemente
- B.1.4. Farbmodelle

B.2. Zweidimensionale Bilderzeugung

- B.2.1. Transformation
- B.2.2. Clipping
- B.2.3. Verrasterung

B.3. Dreidimensionale Bilderzeugung

- B.3.1. Transformation
- B.3.2. Beleuchtung
- B.3.3. Projektion
- B.3.4. Sichtbarkeitsberechnung

B.4. Beleuchtungssimulation

- B.4.1. Strahlverfolgung (Raytracing)
 - B.4.2. Strahlungsverfahren (Radiosity)
-

Lokale Beleuchtungsmodelle

- Empirisch
- Diffuses und ambientes Licht
- Glanzeffekte

→ Helligkeit an einer Stelle eines Objektes war abhängig von dem *direkt einfallenden* Licht an dieser Stelle und den Materialeigenschaften

Realität aber komplexer: Interaktion zwischen Objektoberflächen und Licht

- indirekte Beleuchtung
- mehrfache Reflexion
- Brechung, Transmission

Simulation durch Globale Beleuchtungsmodelle

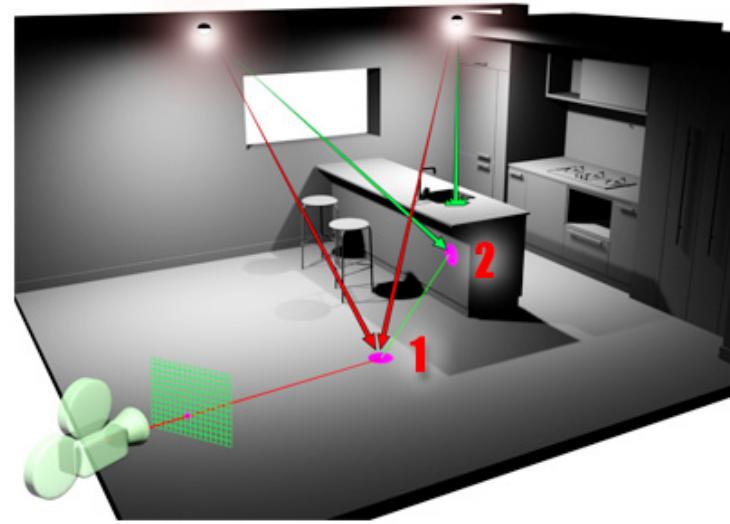
- Raytracing
- Radiosity

Globale Beleuchtungsmodelle

- **Ray Tracing (Strahlverfolgung)**

„From Eye to Light“

- Reflexionen
- Spiegelungen
- Schatten (implizit)

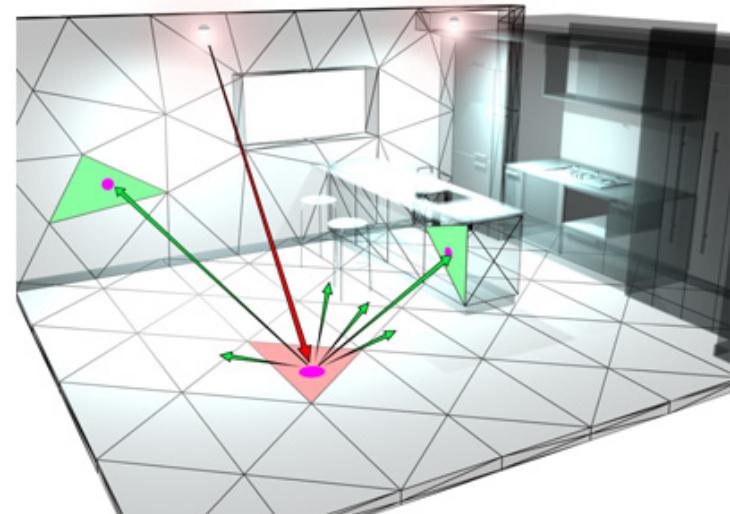


Prinzip der Strahlverfolgung

- **Radiosity (Strahlungsverfahren)**

„From Light to Surface“

- Indirektes Licht
- Licht über mehrere diffuse Flächen
- Photon Mapping
- Kaustiken

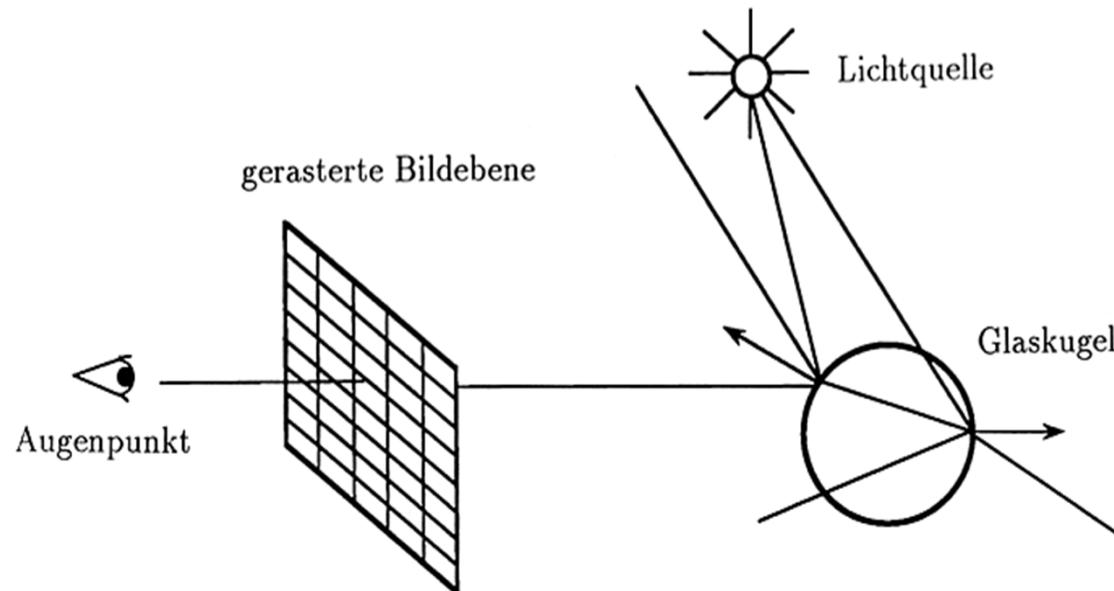


Prinzip des Strahlungsverfahrens

Quelle: Autodesk 3ds Max 2012

Basisalgorithmus

- schicke einen Strahl vom Betrachterstandpunkt durch jedes Pixel der Bildebene
- bestimme den Schnittpunkt des Strahls mit einem Objekt der Szene und am Schnittpunkt den Beitrag ① des lokalen Beleuchtungsmodells (Phong)
- berechne Strahl in Reflexionsrichtung und evtl. in Brechungsrichtung
- Berechne den Beitrag von Reflexion ② und Brechung ③ rekursiv durch Verfolgen dieser beiden Strahlen
- Beleuchtung am Schnittpunkt (und damit Farbe des Pixels) ergibt sich aus gewichteter Summe dieser drei Beiträge



B.4.1. Strahlverfolgung (Raytracing)

B.4.1.1. Algorithmus

Eingabe:

- 3D-Szene S aus Flächen,
- l Punktlichtquellen L_i , $i = 1, \dots, l$,
- Oberflächenattribute,
- Augenpunkt a ,
- Projektionsebene
- Bildauflösung $m \times n$.

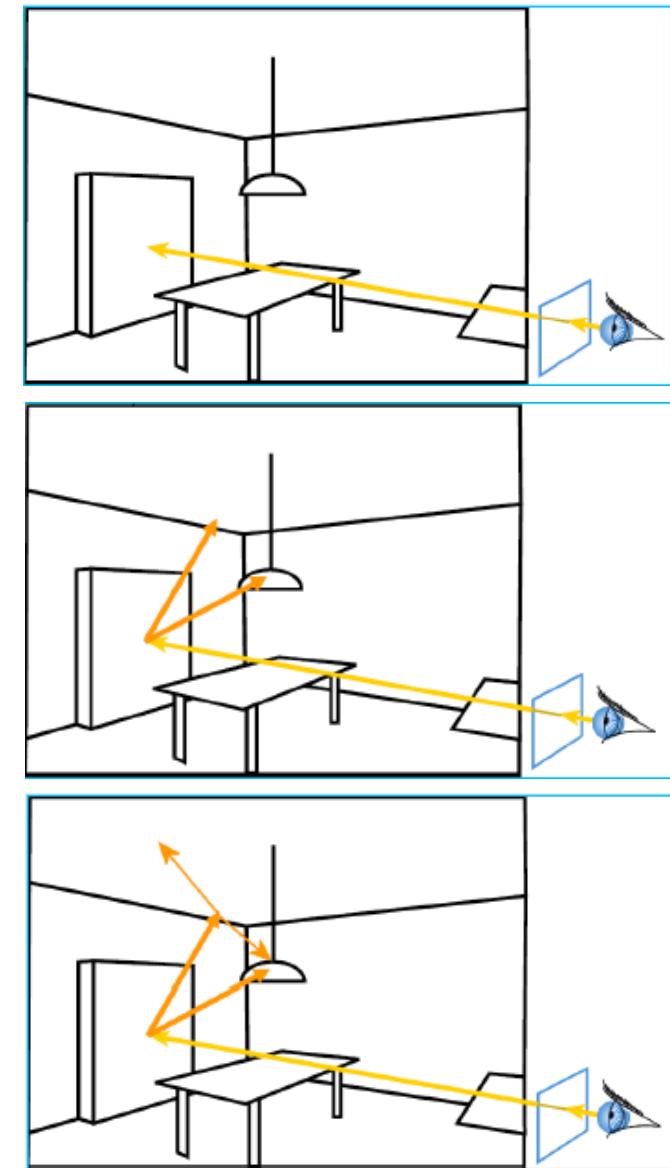
Ausgabe: $m \times n$ -Rasterbild

Algorithmus:

- ein Strahl pro Pixel
- Reflexionsrichtung verfolgen, Strahlen zu den Lichtquellen
- Rekursion

Bedingungen der Rekursion:

- vorgegebene maximale Rekursionstiefe erreicht
- Dämpfung auf dem Sehweg vom Augenpunkt zum aktuellen Punkt stärker als gegebene Schranke.

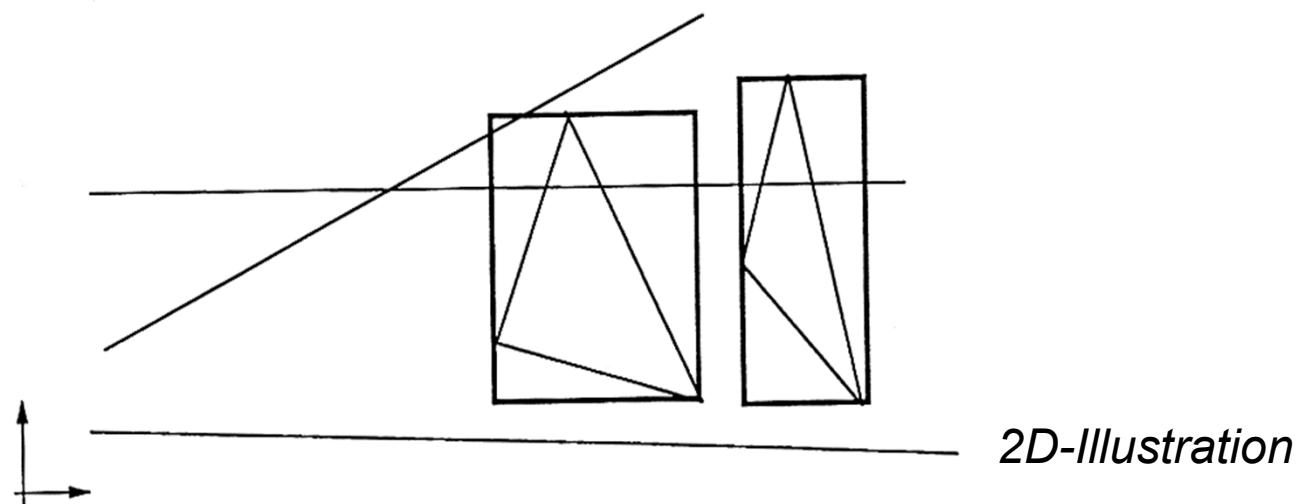


Effiziente Strahlverfolgung (Reduktion des Rechenaufwandes für Schnitttests)

1. Effiziente Ausführung des Tests auf Schnitt zwischen Strahl und Szenenelement
 - Hüllquader
2. Reduzierung der Anzahl der auszuführenden Schnitttests durch geeignete Suchdatenstrukturen
 - geschachtelte Hüllquader
 - reguläres, hierarchische oder geschachtelte Gitter
 - kd-Baum
3. Strahlverfolgung in Echtzeit (Real Time Ray Tracing)
 - Ausnutzen interner Parallelverarbeitung von PC-Prozessoren und Graphikprozessoren
 - Parallelverarbeitung auf einem schnellen Netzwerk von PCs

Hüllquader:

Anstelle des Schnitttests mit einem gegebenen Szenenelement wird zunächst ein dieses umgebender Quader mit dem Strahl auf Schnitt getestet. Erst wenn dies einen Schnitt ergibt, wird der Schnitttest mit dem eigentlichen Element ausgeführt.

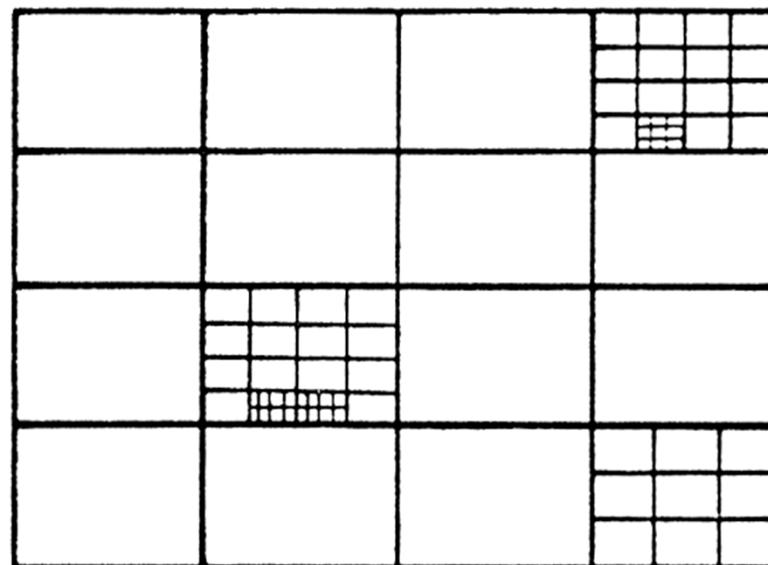


Vorteil: Falls häufig kein Schnitt gefunden wird und der Schnitttest für das gegebene Element relativ aufwendig ist, kann durch den schnell ausführbaren Schnitttest Strahl/Quader signifikant Rechenzeit gespart werden.

Nachteil: Die Hüllquader werden vorberechnet und gespeichert. Dadurch wird zusätzlicher Speicherplatz für die Quader benötigt.

Hierarchische Gitter (Gitterbaum):

Gitterzellen, in denen sich bei der Verteilung nach der Strategie des regulären Gitters Szenenelemente häufen, werden ihrerseits wieder durch ein reguläres Gitter zerlegt. Bei erneuter Häufung wird die Zerlegung in den betroffenen Zellen iteriert.



2D-Illustration

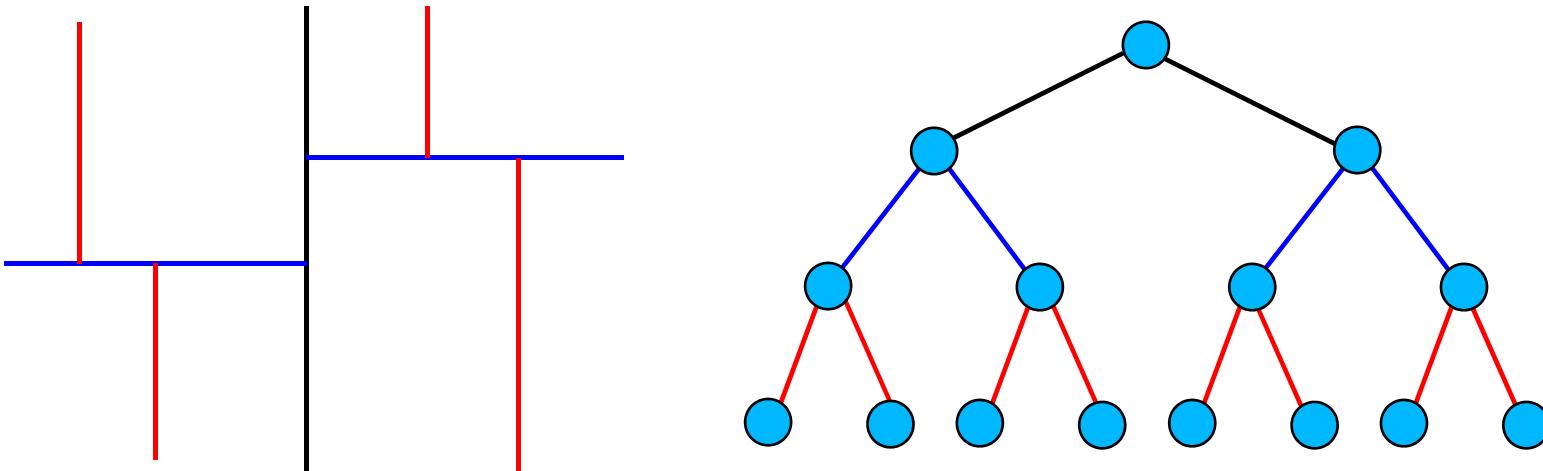
Spezialfall: Ein $2 \times 2 \times 2$ -Gitter führt auf die sogenannte Octree-Zerlegung.

kd-Baum:

Binärbaum, der den Szenenraum iterativ durch achsenparallele Ebenen in zwei Teile aufteilt.

Dabei geschieht die Teilung alternierend in den Koordinatenrichtungen.

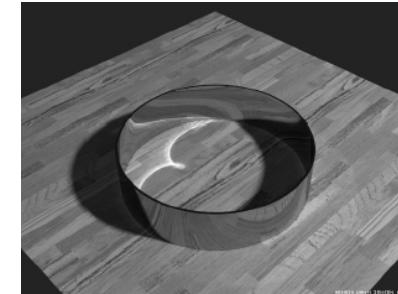
Beispiel: Illustration eines 2-D-Baums (in der Anwendung ist $k=3$)



B.4.1. Strahlverfolgung (Raytracing)

B.4.1.3. Bemerkungen

1. Durch Strahlverfolgung lassen sich auch weitergehende Lichteffekte berechnen. Damit kann etwa die allgemeine Bilderzeugungsgleichung gelöst werden (siehe Vorlesung „Graphische Datenverarbeitung“).



2. Die Strahlverfolgung erlaubt die Erzeugung von Bildern aus praktisch beliebigen Typen von Szenenbeschreibungen. Die wird im Kapitel J der Vorlesung beispielsweise für Voxelvolumendatensätze genutzt.



3. Eine Alternative zur realitätsnahen Bilderzeugung ist die Erzeugung nicht fotorealistisch wirkender Bilder.



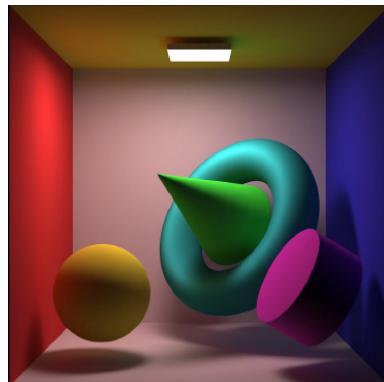
Nachteile des Strahlverfolgungsverfahrens

- keine diffuse Interreflexion zwischen Szenenelementen
- keine ausgedehnten Lichtquellen

Lösung: Strahlungsverfahren (Radiosity Approach)

Idee:

- jede Fläche kann Licht abstrahlen
- alle Lichtquellen mit inhärenten Flächeninhalt modelliert
- Szenenerzeugung in diskrete Flächenelemente (*Patches*)
- Jeder Patch hat endliche Größe, strahlt über die gesamte Fläche gleichmäßig Licht ab und reflektiert Licht.



Diffuse Reflexion



Diffuse und spekulare Reflektion



Diffuse, spekulare und transparente Reflexion

Vorgehensweise beim Strahlungsverfahren:

Simulation des Lichttransports zwischen den Oberflächen der Szene.

Grundlagen:

- (alternative) Strahlungsgleichung (ohne Fläche A_i des Flächenelements i)
- Energieerhaltungssatz

Finite Elemente Verfahren: Unterteilen der Szene in „Patches“

Prinzip:

Berechnung des Anteils an Licht, das ausgehend von einem Patch ein anderes trifft

- nur diffuse Reflexion betrachtet
- Anteil des Lichts nur abhängig von Geometrie – aufwendig zu berechnen

B.4.2. Strahlungsverfahren (Radiosity)

B.4.2.1. Strahlungsgleichung

Strahlungsgleichung: $B_i = E_i + \rho_i \sum_{j=1}^n B_j F_{i,j}$

mit

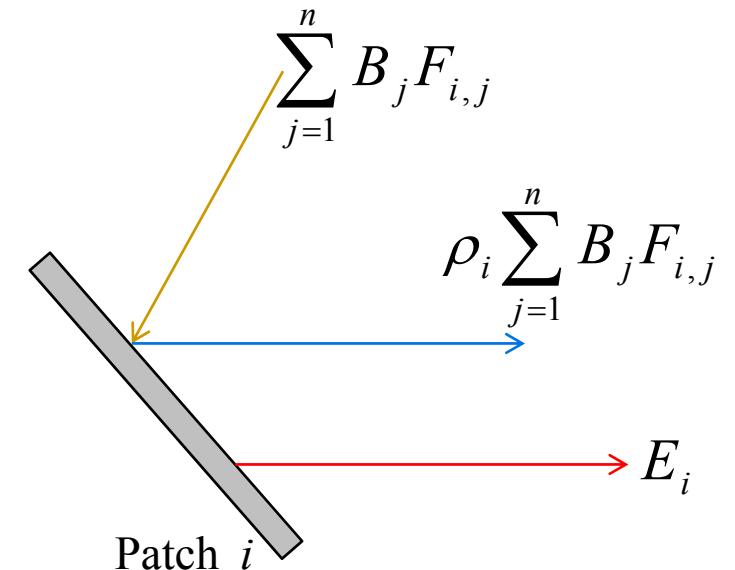
B_i : Strahlung des Patches i

E_i : Eigenemission des Patches i

ρ_i : diffuser Reflektionskoeffizient des Patches i

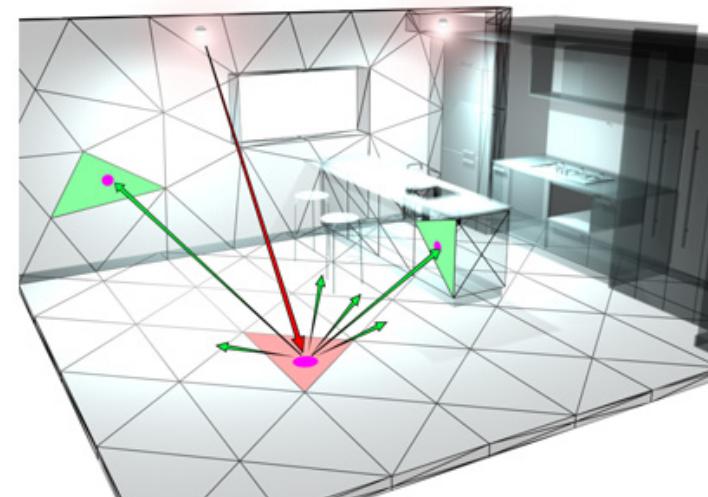
$F_{j,i}$: Formfaktor von Patch j nach Patch i

n : Zahl der Patches



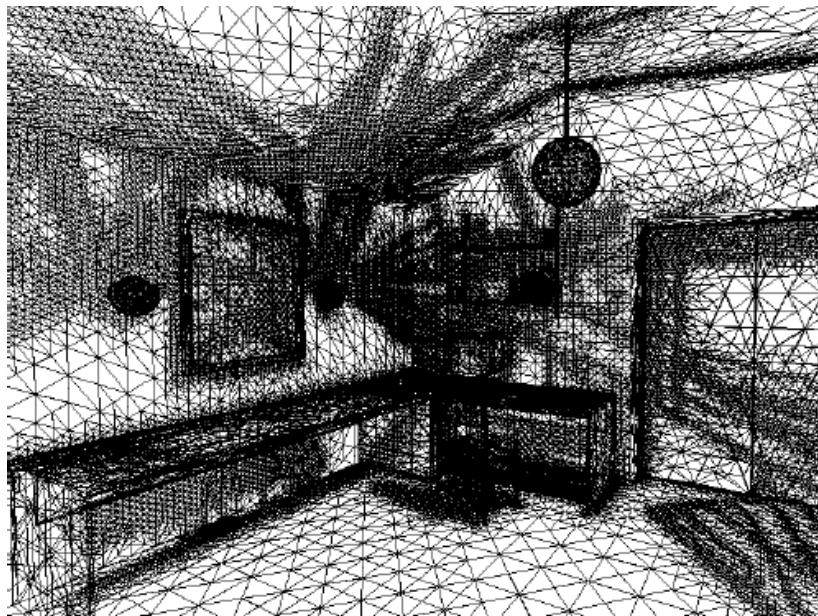
Algorithmus

- starte mit den Patches, die Licht aussenden (Lichtquellen)
- sende von diesen Licht in die Szene
- berechne für alle anderen Patches ankommende Lichtmenge
- iteriere bis Energiegleichgewicht



B.4.2. Strahlungsverfahren (Radiosity)

B.4.2.2. Beispiel



Szene mit Patches



Beleuchtungssimulation