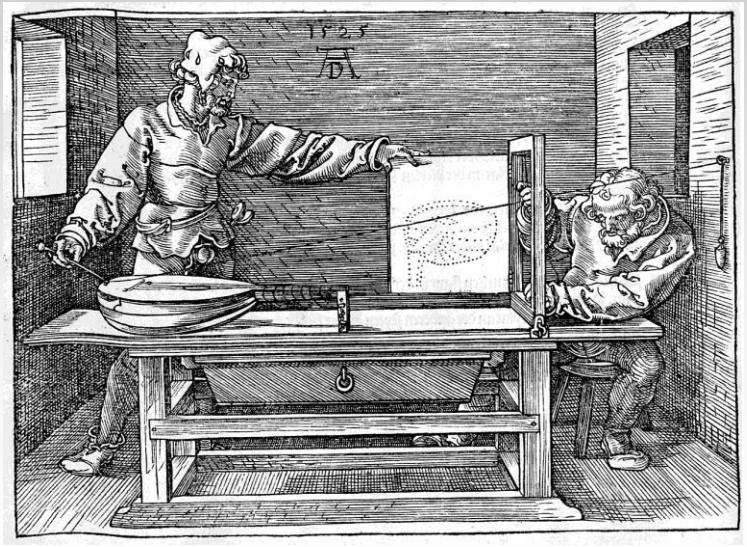
Computergrafik

Vorlesung im Wintersemester 2024/25
Kapitel 2: Raytracing

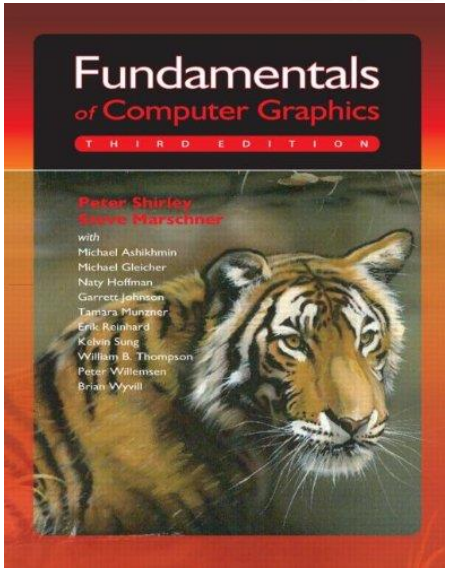


Albrecht Dürer: „Underweysung der messung mit dem zirckel und richtscheyt, in Linien ebnen unnd gantzen corporen“, 1525 [9]

Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

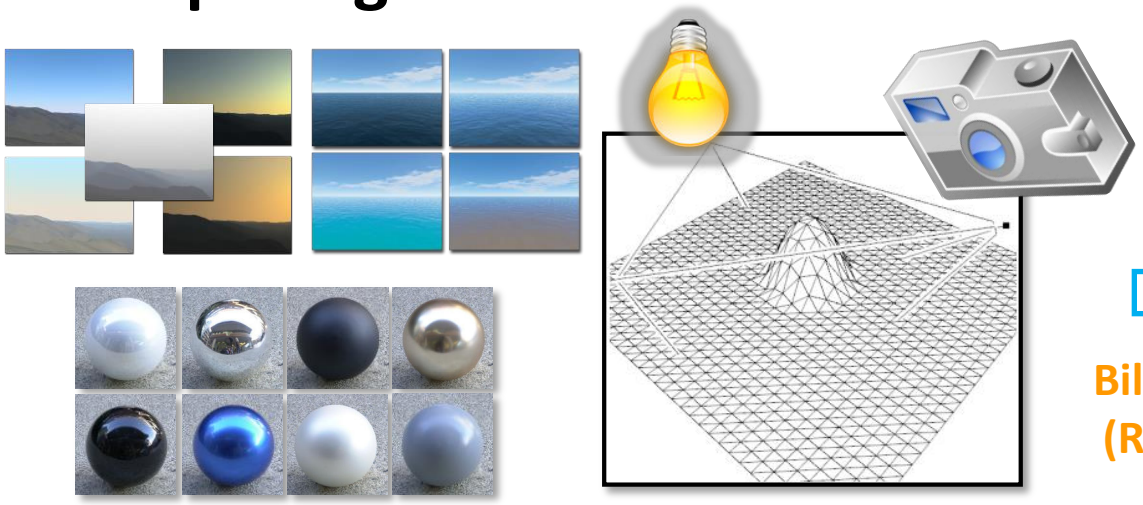
Literatur

- **Fundamentals of Computer Graphics**,
  P. Shirley, S. Marschner, 3rd Edition, AK Peters
  → Kapitel 4
- **Immersive Linear Algebra**,
  J. Ström, K. Åström, T. Akenine-Möller,
  http://immersivemath.com/ila/index.html
  → Kapitel 2-4



[1]

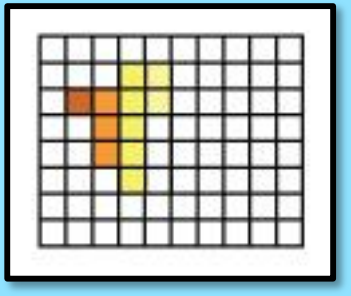
Computergrafik und Farbbilder



Virtuelle Szene
(Geometrie, Material, Kamera, Lichtquellen, …)

Bildsynthese (Rendering)

[2]



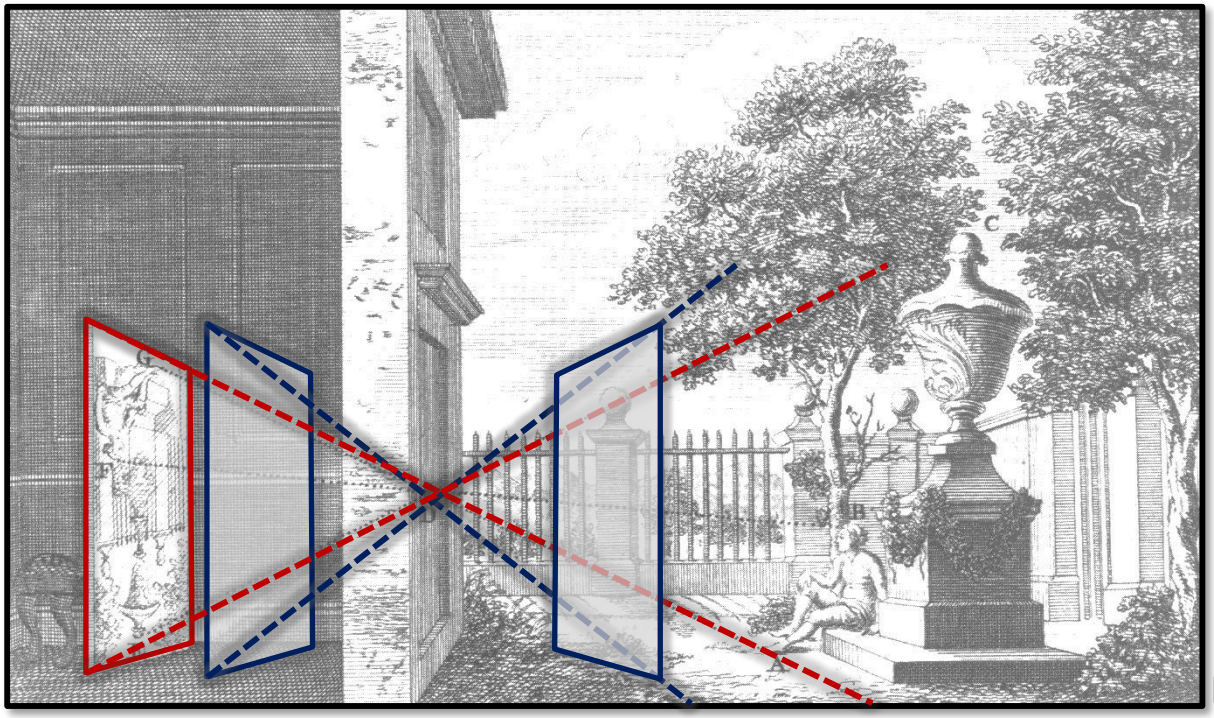
Frame Buffer

Ansteuerung des Monitors

Monitor

Lochkamera

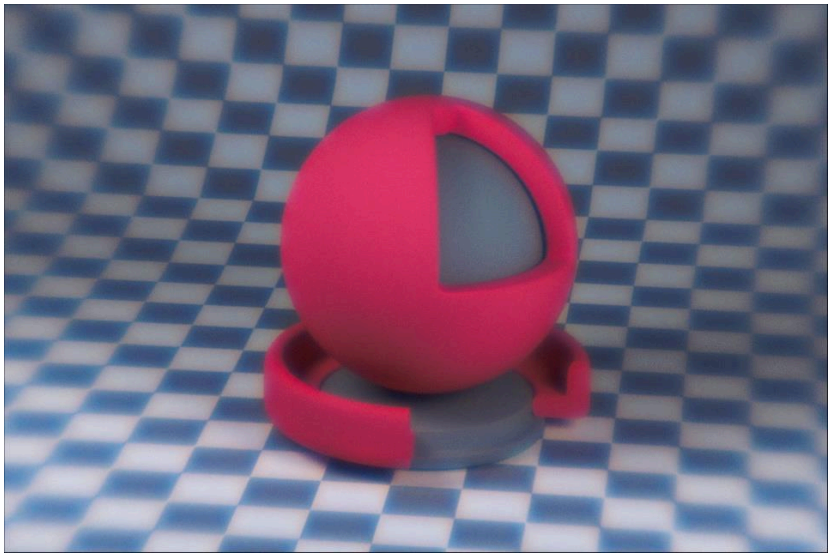
- in der CG verwendet man oft das einfache Modell der Lochkamera
  - Kammer mit kleiner Öffnung → auf der gegenüberliegenden Wand entsteht ein spiegelverkehrtes Bild

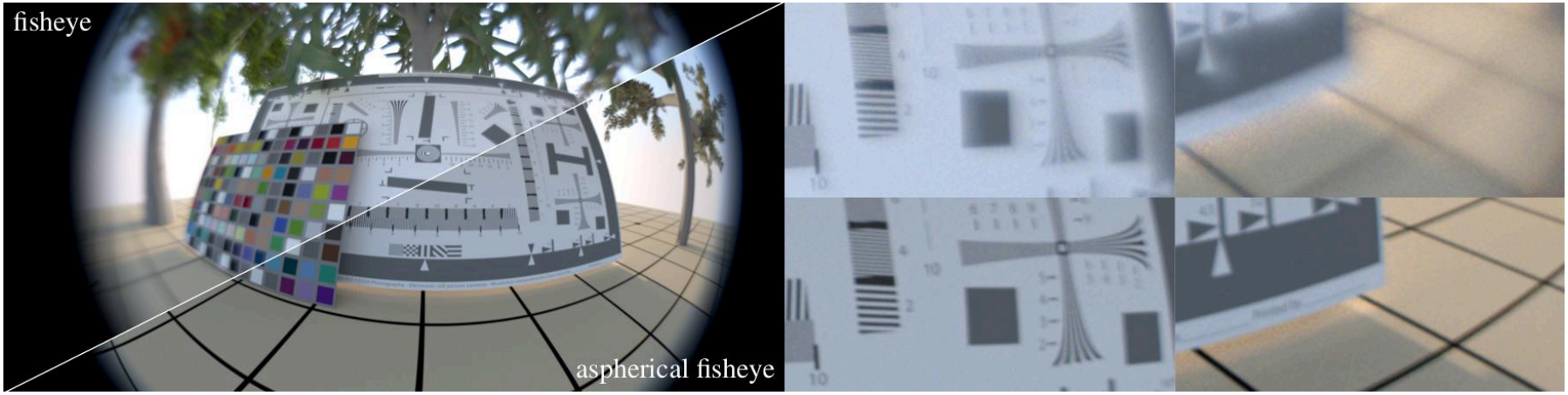


[3]

  - (im Prinzip) unbegrenzte Schärfentiefe
  - Kamera definiert durch Position der Öffnung und Bildebene

Bsp.: Abbildung durch Linsen (keine Lochkamera)

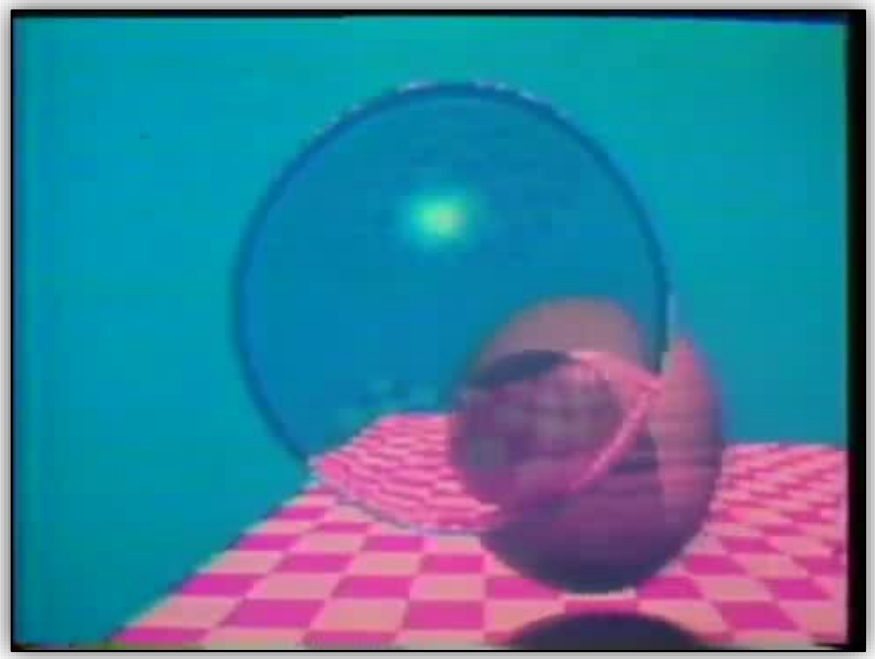




Raytracing

Grundidee (Whitted-style raytracing)

- Raytracing ist ein Verfahren zur Bildsynthese (engl. Rendering)
- geometrische Optik: Licht breitet sich entlang von Strahlen aus
- die Ausbreitung des Lichts wird – von der Kamera aus – zurückverfolgt

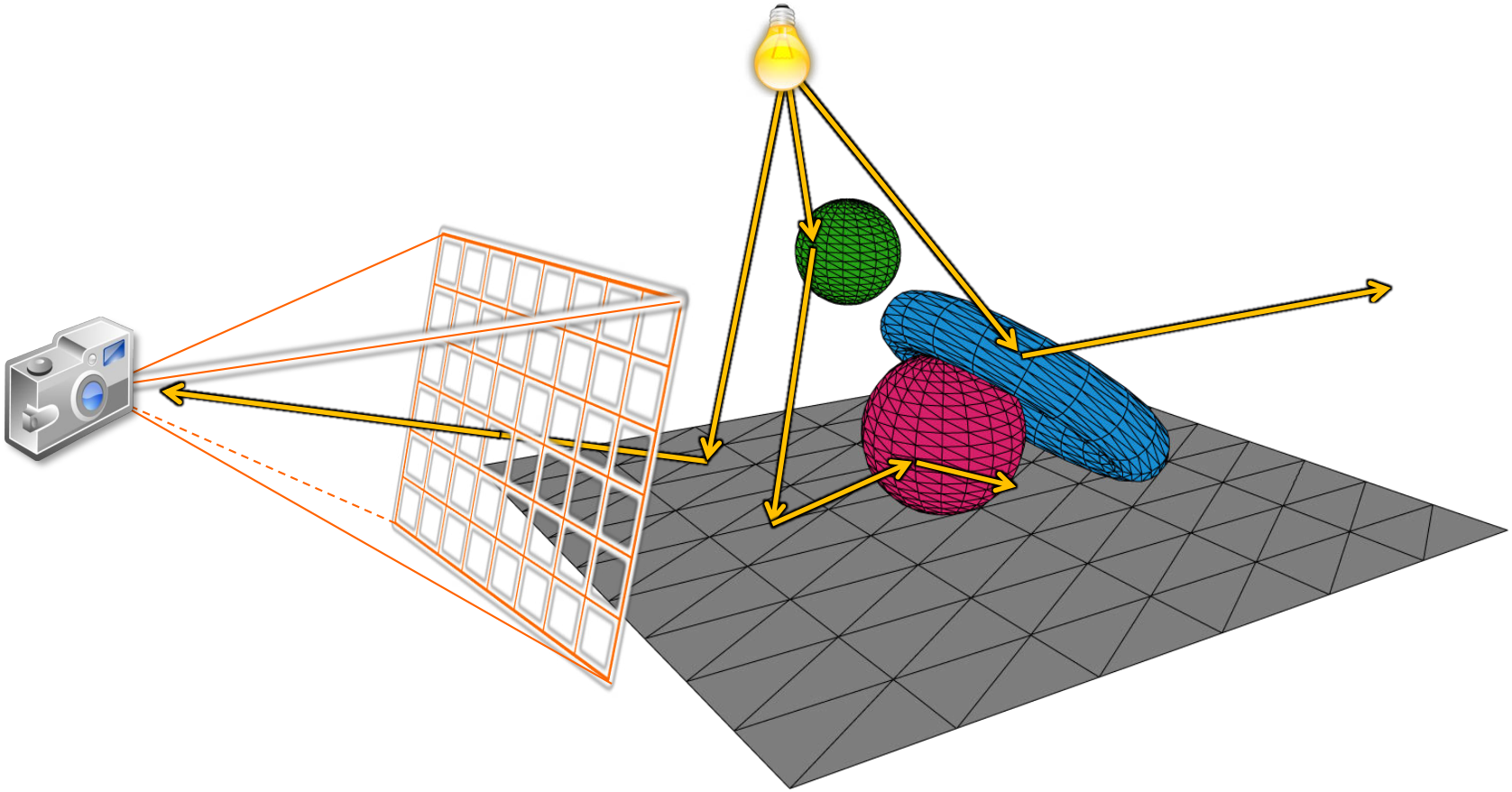


Rekursives Raytracing von Turner Whitted (1979/80)

Raytracing

Grundidee (Whitted-style raytracing)

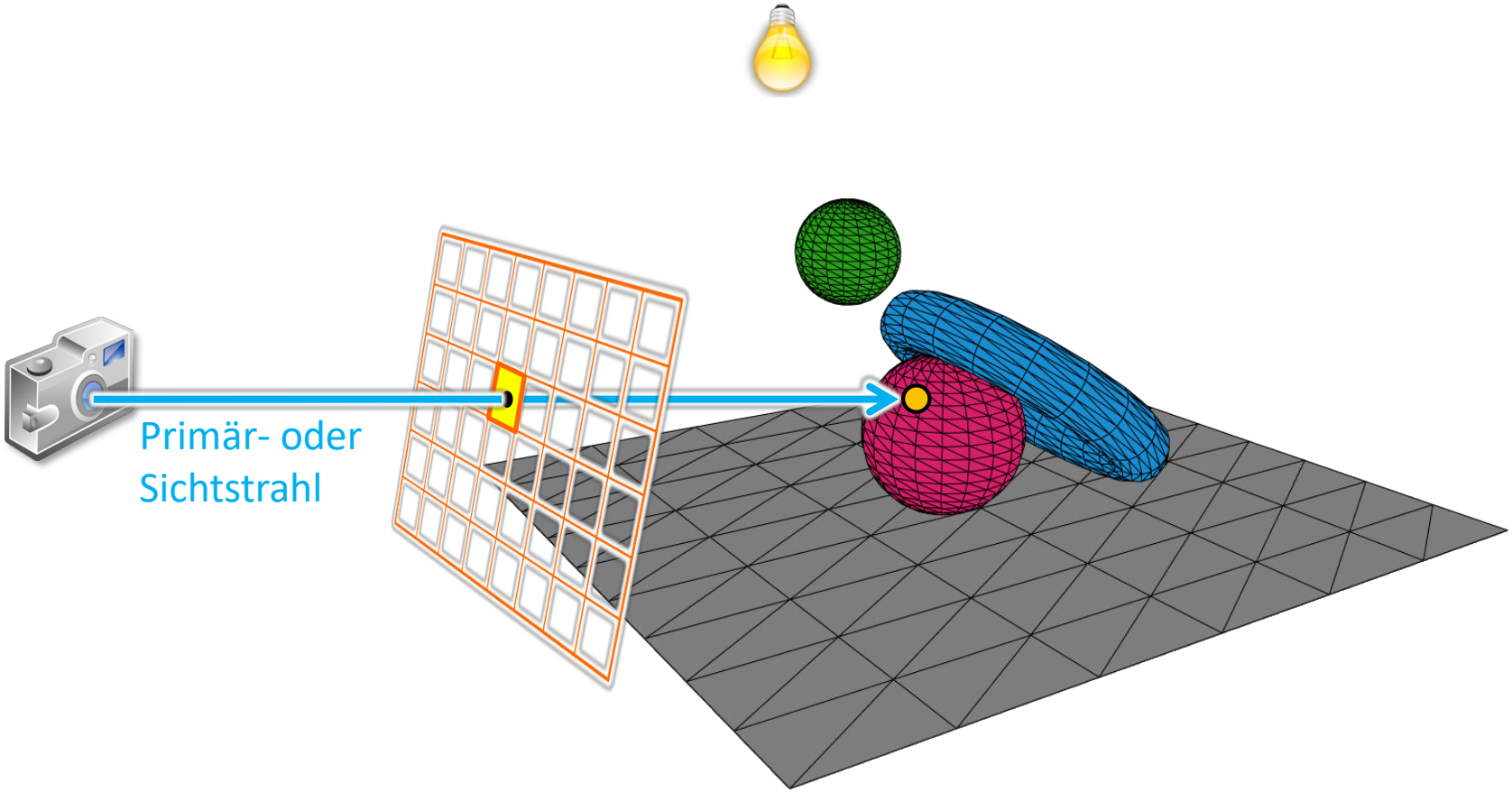
- Raytracing ist ein Verfahren zur Bildsynthese (engl. Rendering)
- geometrische Optik: Licht breitet sich entlang von Strahlen aus
- die Ausbreitung des Lichts wird – von der Kamera aus – zurückverfolgt



Raytracing

Grundidee (Whitted-style raytracing)

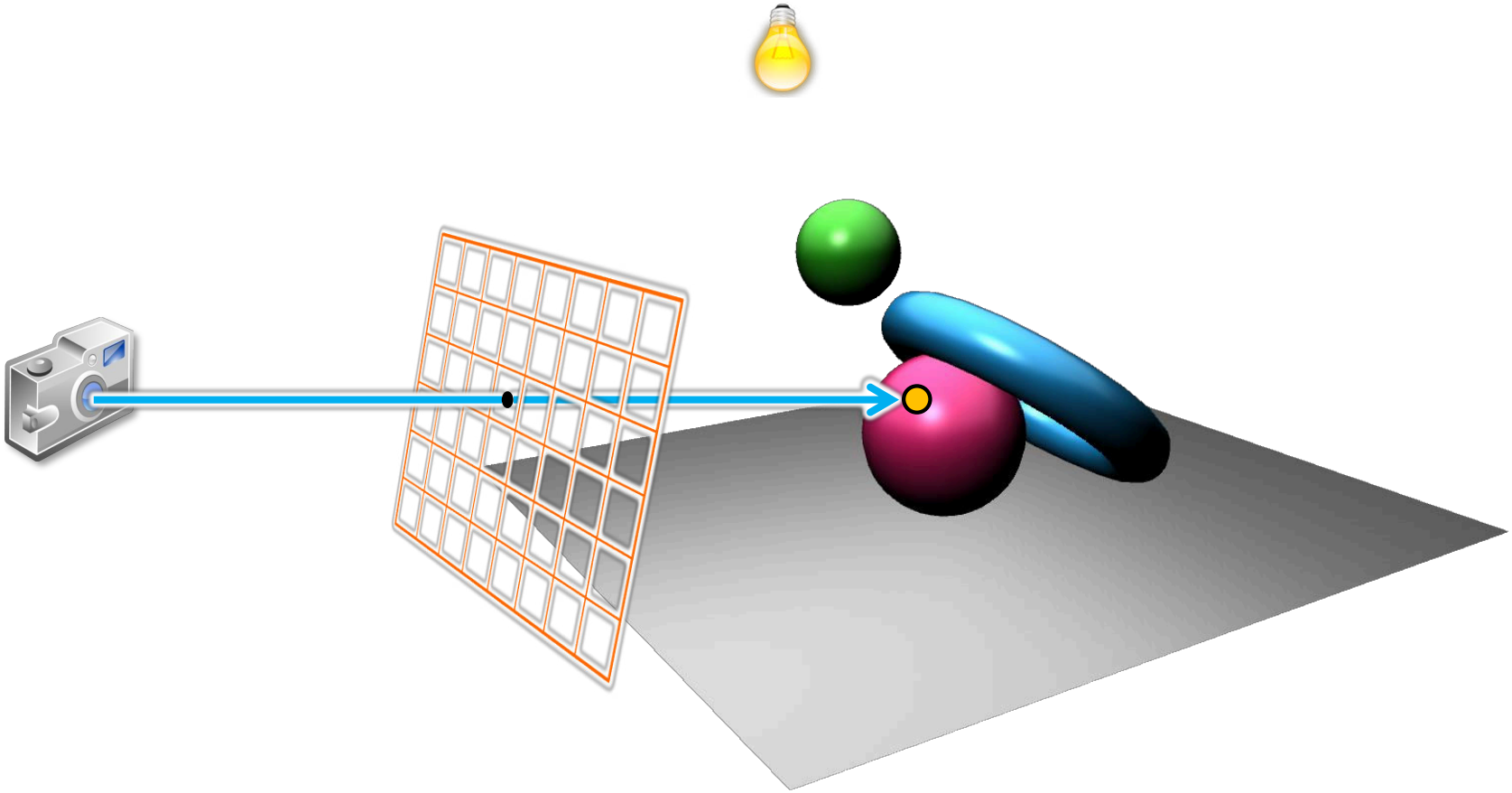
- betrachte jeden Pixel des Bildes: verfolge einen Strahl durch diesen Pixel und finde die nächste Fläche entlang des Strahls



Raytracing

Grundidee (Whitted-style raytracing)

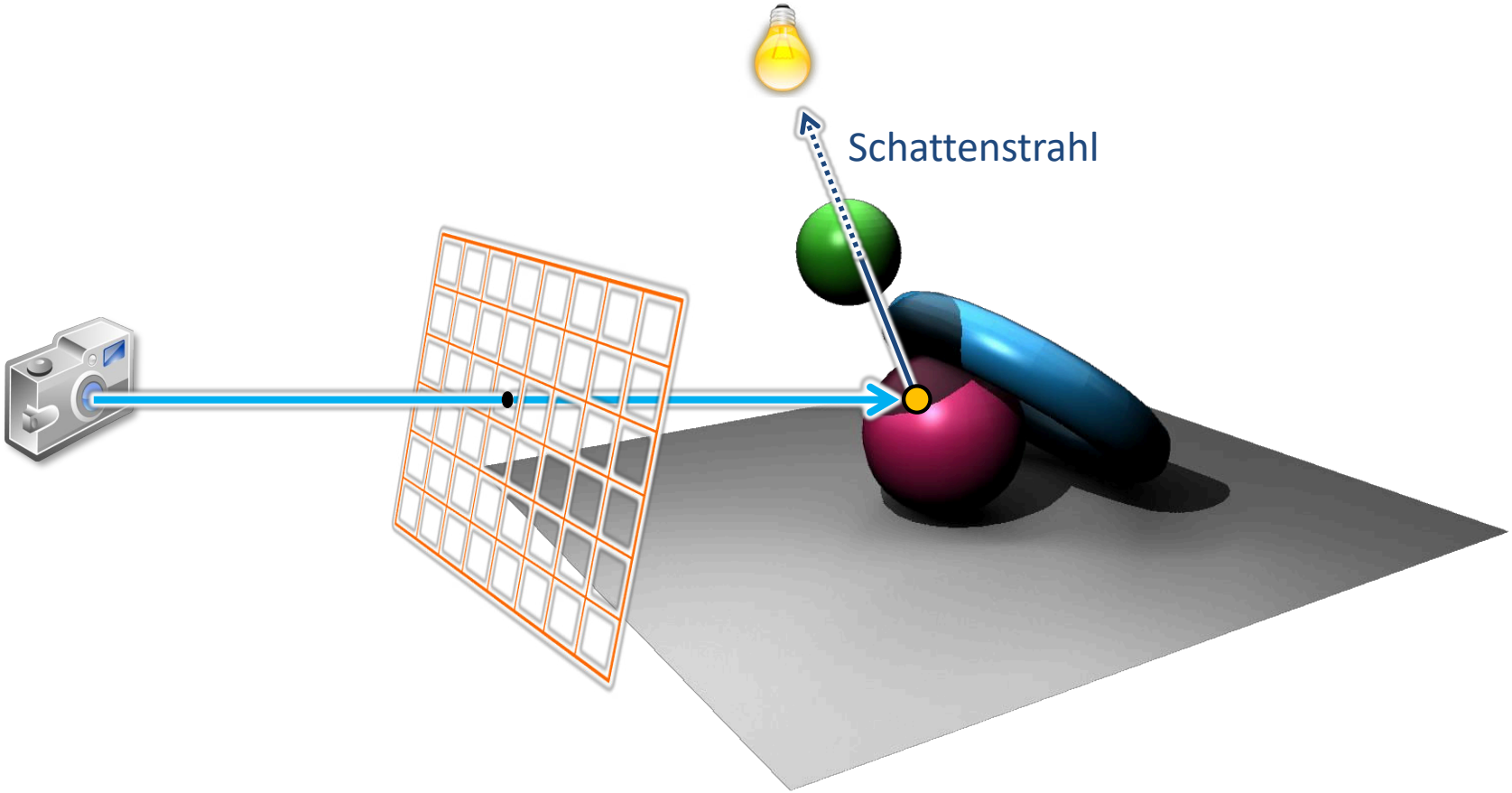
- betrachte jeden Pixel des Bildes: verfolge einen Strahl durch diesen Pixel und finde die nächste Fläche entlang des Strahls
- berechne die Schattierung der Fläche



Raytracing

Grundidee (Whitted-style raytracing)

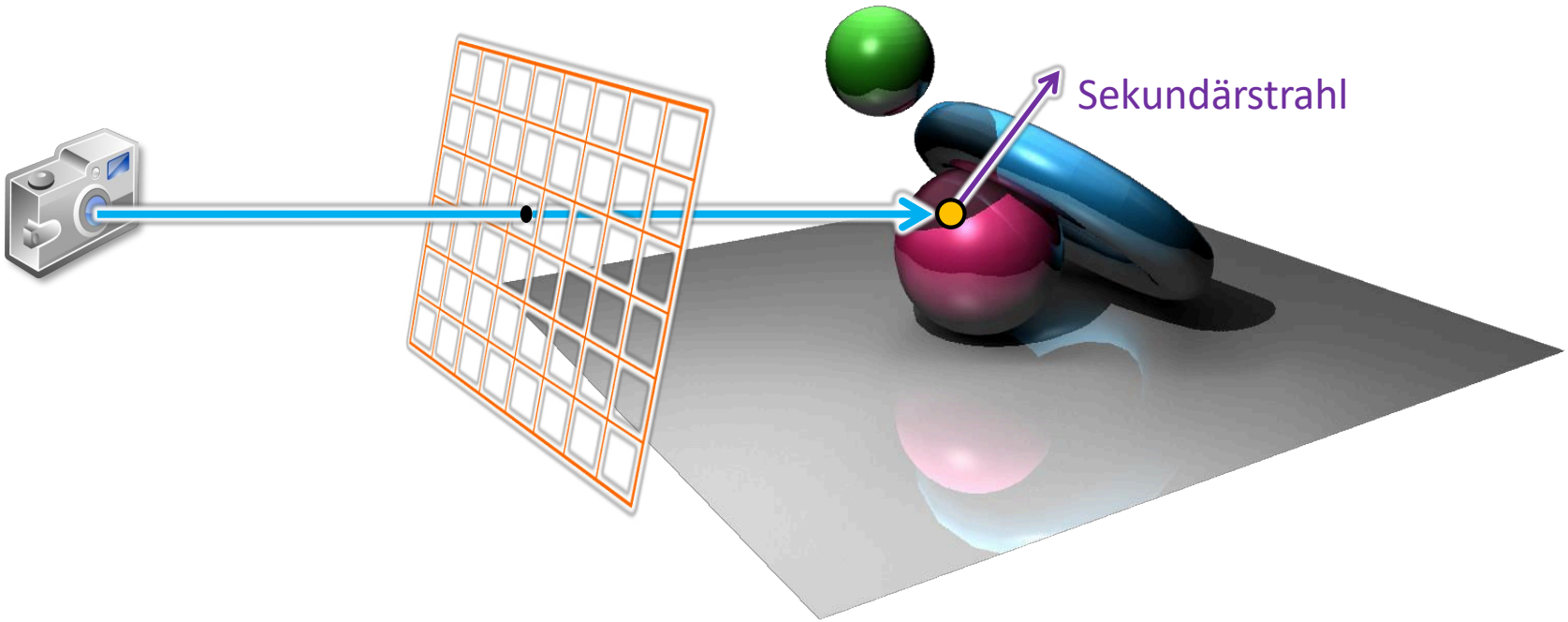
- betrachte jeden Pixel des Bildes: verfolge einen Strahl durch diesen Pixel und finde die nächste Fläche entlang des Strahls
- berechne die Schattierung der Fläche



Raytracing

Grundidee (Whitted-style raytracing)

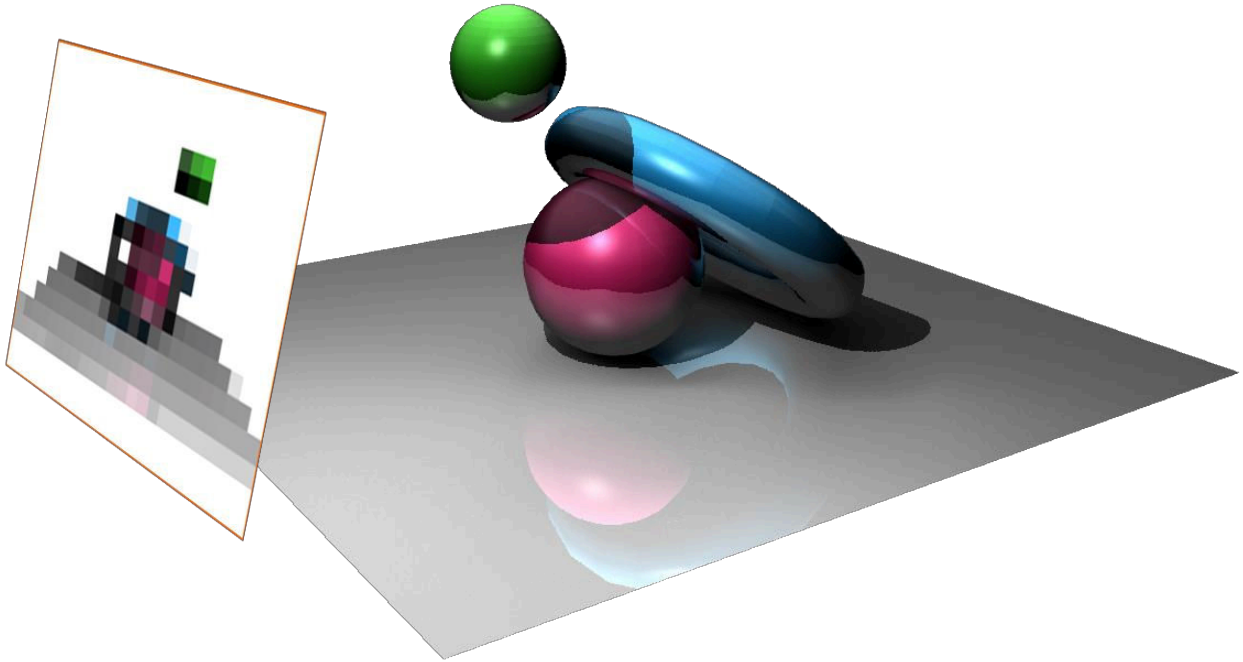
- betrachte jeden Pixel des Bildes: verfolge einen Strahl durch diesen Pixel und finde die nächste Fläche entlang des Strahls
- berechne die Schattierung der Fläche
- setze Strahlverfolgung fort, wenn Flächen spiegeln oder transmittieren



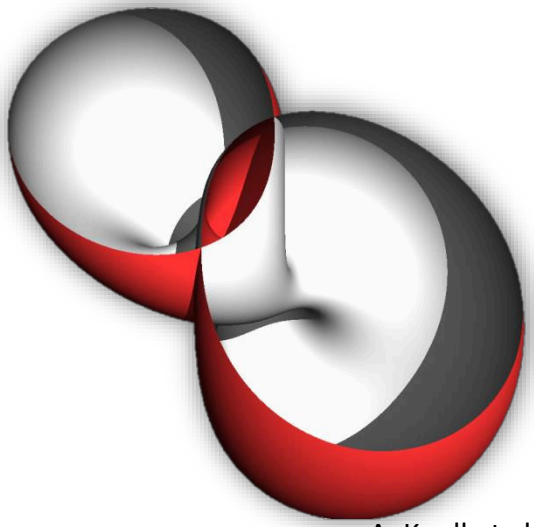
Raytracing

Grundidee (Whitted-style raytracing)

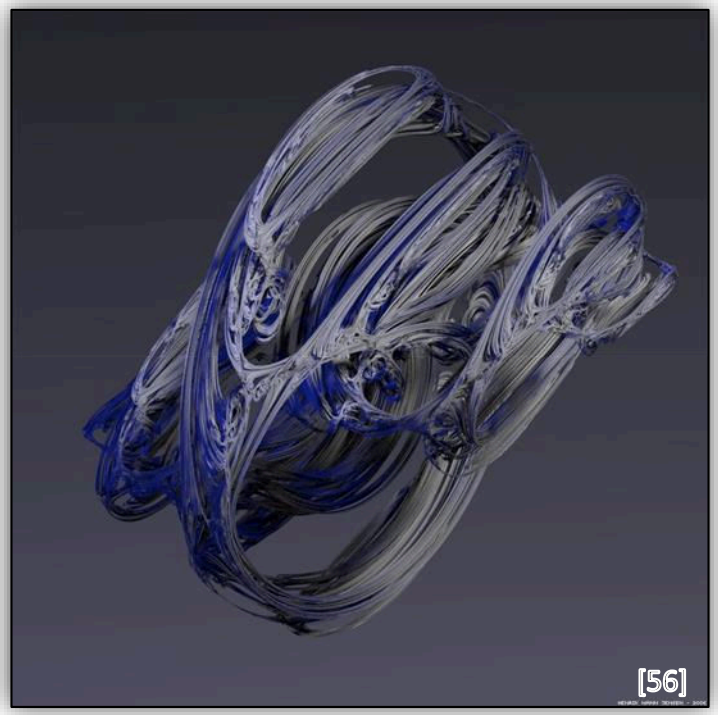
- betrachte jeden Pixel des Bildes: verfolge einen Strahl durch diesen Pixel und finde die nächste Fläche entlang des Strahls
- berechne die Schattierung der Fläche
- setze Strahlverfolgung fort, wenn Flächen spiegeln oder transmittieren



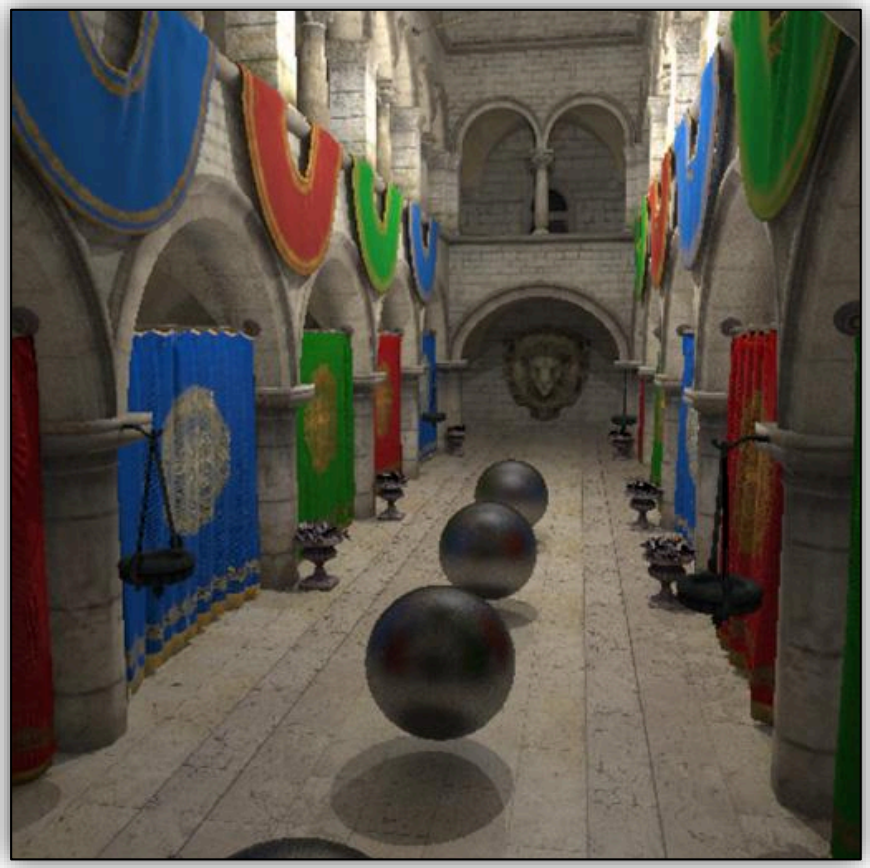
Raytracing Beispiele

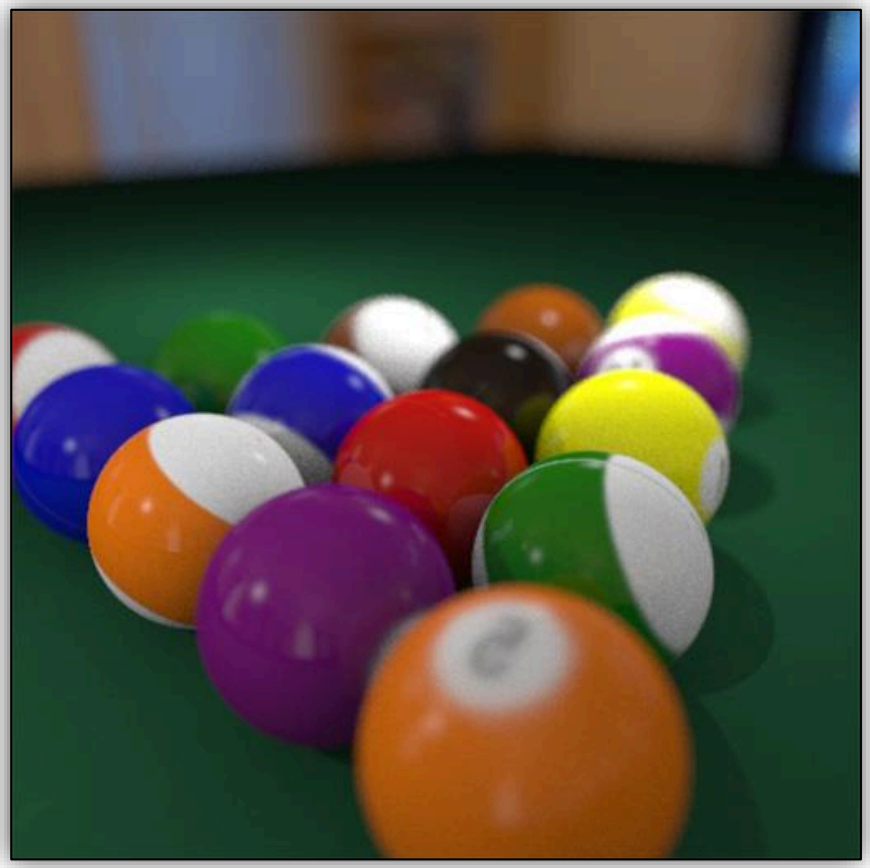


A. Knoll et al. 07 [4]



Raytracing in der Übung



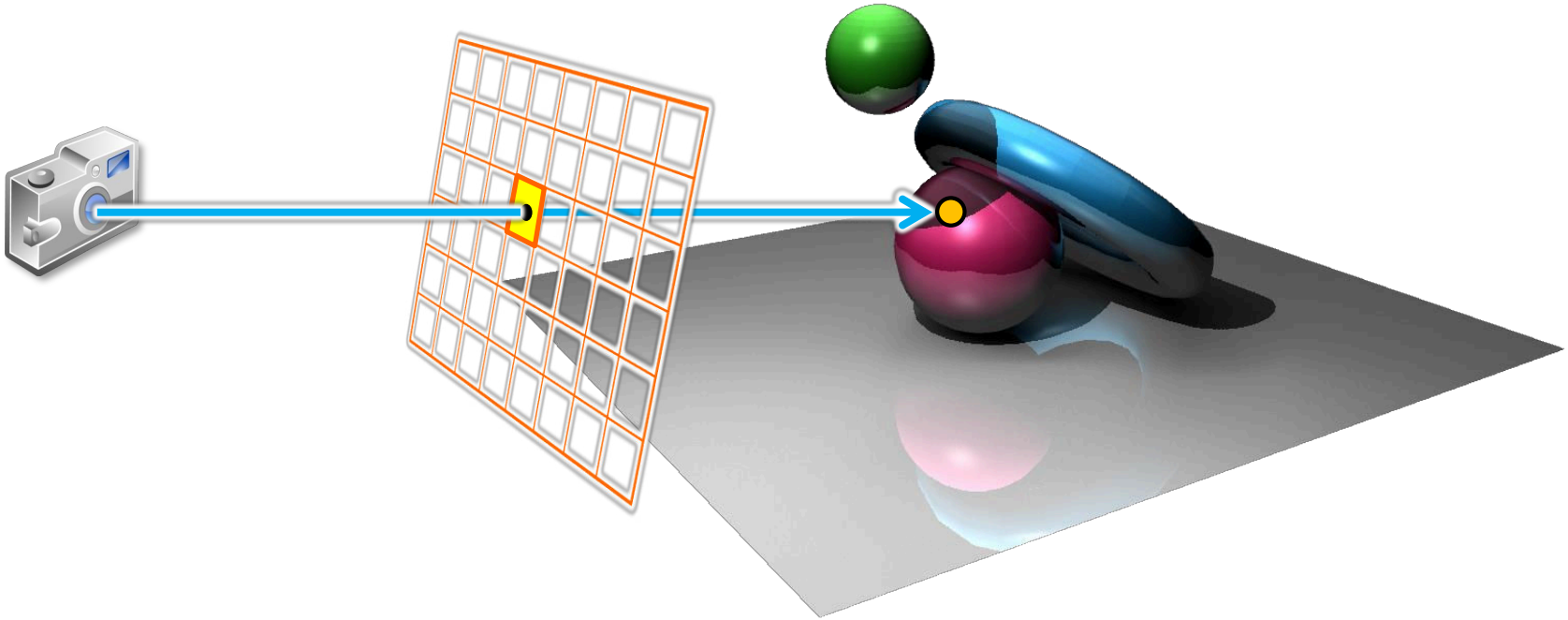


Raytracing Beispiele (siehe CG2 und FotoBS)

Raytracing

Schritte (die ein Raytracing-Programm implementieren muss)

- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting, ray intersection):
  finde Dreieck, das den Sichtstrahl am nächsten zur Kamera schneidet
- Schattierung und Beleuchtungsberechnung (shading)
- Sekundärstrahlen für Spiegelung und Transmission
- … gleich nach ein paar Vorbetrachtungen!



Analytische Geometrie


[10]

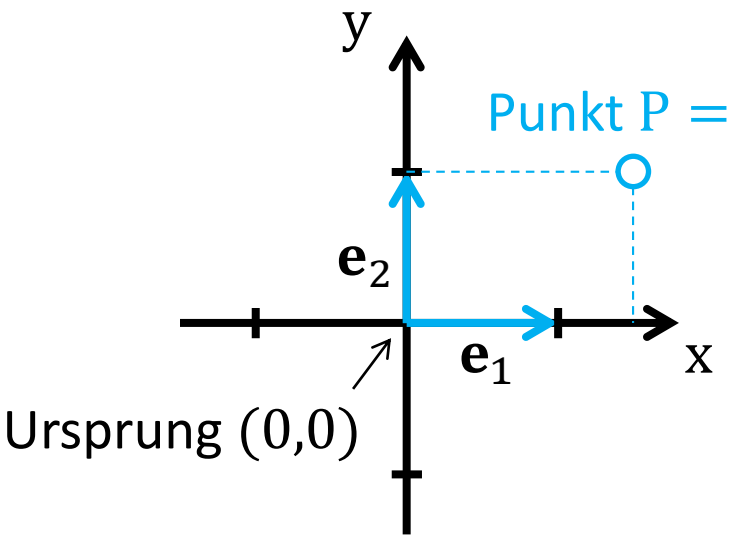
- die Geometrie ist ein Grundpfeiler der Computergrafik
  - Euklid (300 v.Chr.), René Descartes (*1596 †1650)
- Begriffe, die Sie kennen …
  - Koordinatensysteme, Vektoren und Matrizen
  - (Halb-)Geraden, Ebenen, …
  - benötigen wir bspw. für Erzeugung Primärstrahlen, Schnittberechnung, Schattierung
- Notation/Konventionen

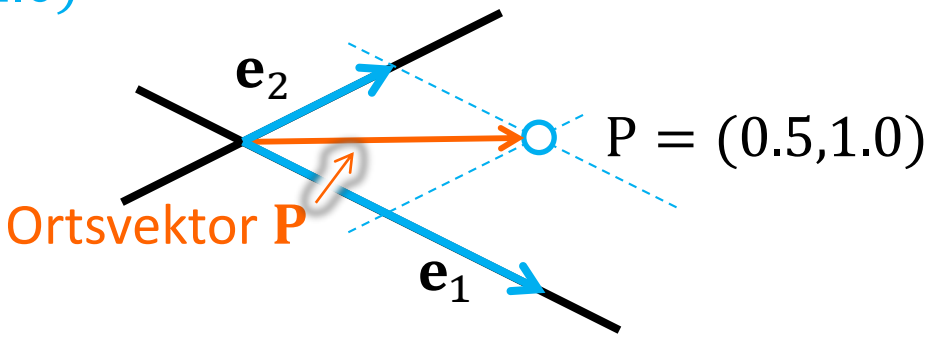
$\alpha, \beta, \gamma, \ldots$	reelle Zahlen, Skalare	$a, b, c, \ldots$	reelle Zahlen
$i, j, k, \ldots$	ganze Zahlen	$\mathbf{a}, \mathbf{b}, \mathbf{c}, \ldots$	Vektoren
$\mathrm{P}, \mathrm{Q}, \mathrm{R}, \ldots$	Punkte	$\mathbf{P}, \mathbf{Q}, \mathbf{R}, \ldots$	Ortsvektoren
$s, t, u, v, \ldots$	Parameterwerte		

Analytische Geometrie

Koordinatensystem, Punkte, Vektoren

- Vektor beschreibt anschaulich eine Länge und Richtung („Pfeil“)
- Koordinatensysteme mit Basisvektoren $\mathbf{e}_1$, $\mathbf{e}_2$, …
  - Kartesisches KoSys: Basisvektoren bilden eine Orthonormalbasis



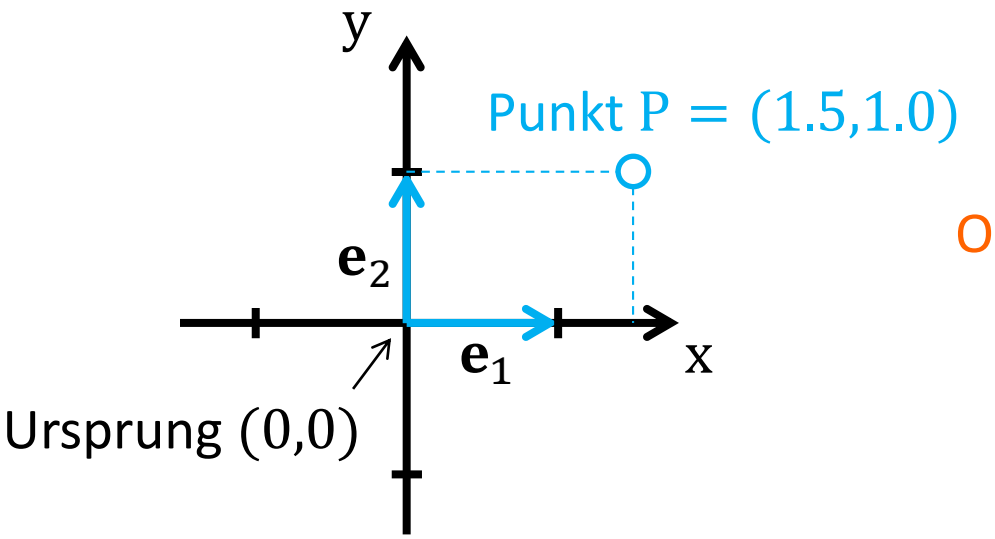


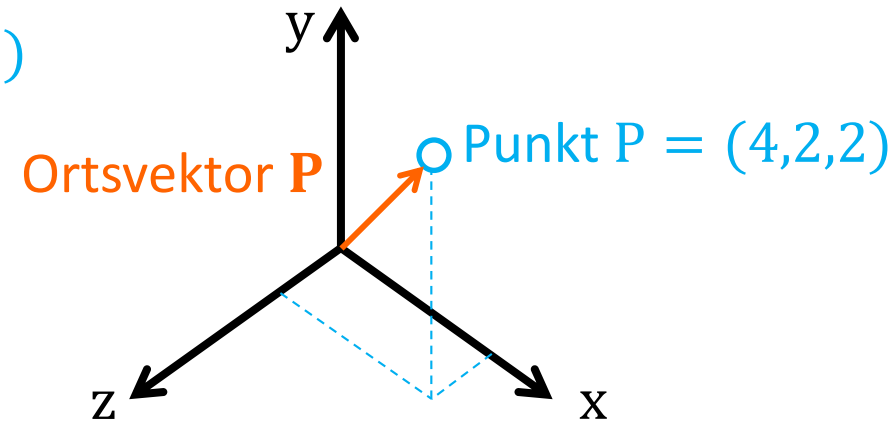
- Ortsvektor $\mathbf{P}$ bezeichnet Vektor $\overrightarrow{\mathrm{OP}}$ von einem Bezugspunkt (typ. Koordinatenursprung O) zu Aufpunkt $\mathrm{P} = (p_1, p_2, \ldots, p_n)$
  - p_i sind die Gewichtungsfaktoren der Linearkombination der aufspannenden Vektoren, d.h. in 3D: $\mathbf{P} = p_1 \cdot \mathbf{e}_1 + p_2 \cdot \mathbf{e}_2 + p_3 \cdot \mathbf{e}_3$

Analytische Geometrie

Koordinatensystem, Punkte, Vektoren

- Vektor beschreibt anschaulich eine Länge und Richtung („Pfeil“)
- Koordinatensysteme mit Basisvektoren $\mathbf{e}_1$, $\mathbf{e}_2$, …
  - Kartesisches KoSys: Basisvektoren bilden eine Orthonormalbasis





- Ortsvektor $\mathbf{P}$ bezeichnet Vektor $\overrightarrow{\mathrm{OP}}$ von einem Bezugspunkt (typ. Koordinatenursprung O) zu Aufpunkt $\mathrm{P} = (p_1, p_2, \ldots, p_n)$
  - p_i sind die Gewichtungsfaktoren der Linearkombination der aufspannenden Vektoren, d.h. in 3D: $\mathbf{P} = p_1 \cdot \mathbf{e}_1 + p_2 \cdot \mathbf{e}_2 + p_3 \cdot \mathbf{e}_3$

Vektor, Punkt und Ortsvektor

- beachte den semantischen Unterschied
  - wir addieren zwei (Richtungs-)Vektoren,
    z.B. Hintereinanderausführung von Verschiebungen
  - Ortsvektoren werden addiert/subtrahiert als Zwischenoperation,
    z.B. Berechnung eines Mittelpunkts oder des Differenzvektors
  - später: tatsächliche Unterscheidung mit homogenen Koordinaten

- Rechenoperationen
  - Länge eines Vektors: $|\mathbf{a}| = \sqrt{\sum_{i=1}^{n} {a_i}^2}$
  - Einheitsvektor gdw. $|\mathbf{a}| = 1$
  - Vektoraddition: $\mathbf{a} \pm \mathbf{b} = (a_1 \pm b_1, a_2 \pm b_2, \dots, a_n \pm b_n)$
  - Skalierung: $\lambda\mathbf{a} = (\lambda a_1, \lambda a_2, \dots, \lambda a_n)$

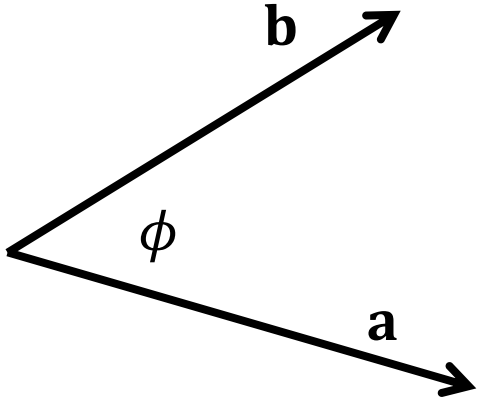
Vektoren

Skalarprodukt (inneres Produkt, engl. dot product)

- Definition
  - $\mathbf{a} \cdot \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$

- als Matrixmultiplikation
  - $\langle \mathbf{a}, \mathbf{b} \rangle = \mathbf{a}^\mathbf{T} \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots + a_n b_n$

- Eigenschaften
  - $\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos \phi$
  - $\mathbf{a} \cdot \mathbf{a} = |\mathbf{a}|^2$
  - $\mathbf{a} \cdot \mathbf{b} = \mathbf{b} \cdot \mathbf{a}$ (kommutativ)
  - $\mathbf{a} \cdot (\mathbf{b} \pm \mathbf{c}) = \mathbf{a} \cdot \mathbf{b} \pm \mathbf{a} \cdot \mathbf{c}$ (distributiv)



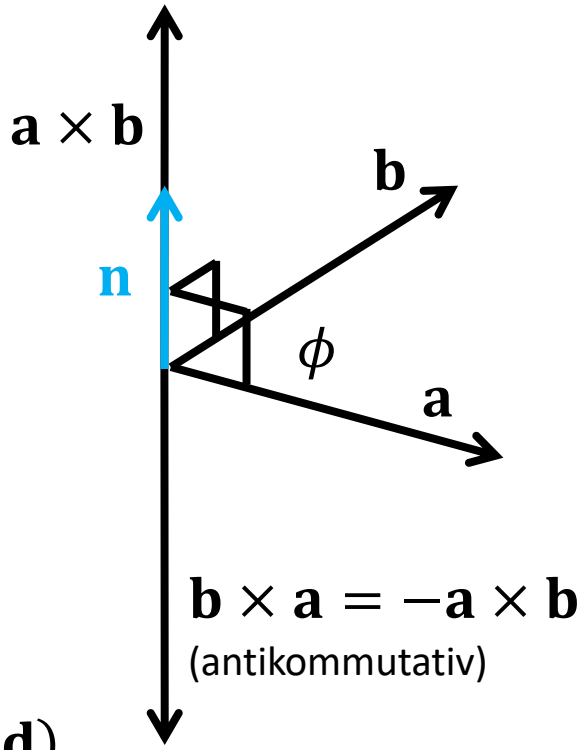
$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos \phi$$

http://immersivemath.com/ila/ch03_dotproduct/ch03.html – Abbildung 3.4

Vektoren

Kreuzprodukt (auch äußeres/vektorielles Produkt, engl. cross product)

- definiert in 3D-Vektorräumen
  - $\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix} = \mathbf{n}|\mathbf{a}||\mathbf{b}| \sin \phi$
  - $|\mathbf{n}| = 1$ und $\mathbf{n}$ steht senkrecht auf $\mathbf{a}$ und $\mathbf{b}$
- Graßmann, Lagrange, … Identitäten
  - $\mathbf{a} \cdot (\mathbf{a} \times \mathbf{b}) = \mathbf{b} \cdot (\mathbf{a} \times \mathbf{b}) = 0$
  - $\mathbf{a} \cdot (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}$
  - $\mathbf{a} \times (\mathbf{b} + \mathbf{c}) = \mathbf{a} \times \mathbf{b} + \mathbf{a} \times \mathbf{c}$
  - $\mathbf{a} \times (k\mathbf{b}) = k(\mathbf{a} \times \mathbf{b})$
  - $\mathbf{a} \times (\mathbf{b} \times \mathbf{c}) = (\mathbf{a} \cdot \mathbf{c}) \cdot \mathbf{b} - (\mathbf{a} \cdot \mathbf{b}) \cdot \mathbf{c}$
  - $(\mathbf{a} \times \mathbf{b}) \cdot (\mathbf{c} \times \mathbf{d}) = (\mathbf{a} \cdot \mathbf{c})(\mathbf{b} \cdot \mathbf{d}) - (\mathbf{b} \cdot \mathbf{c})(\mathbf{a} \cdot \mathbf{d})$

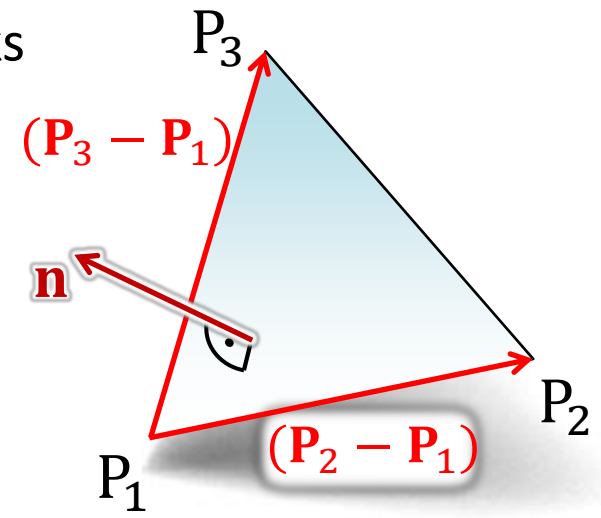


http://immersivemath.com/ila/ch04_vectorproduct/ch04.html – Abbildung 4.7 und 4.8

Vektoren

Kreuzprodukt (auch äußeres/vektorielles Produkt, engl. cross product)

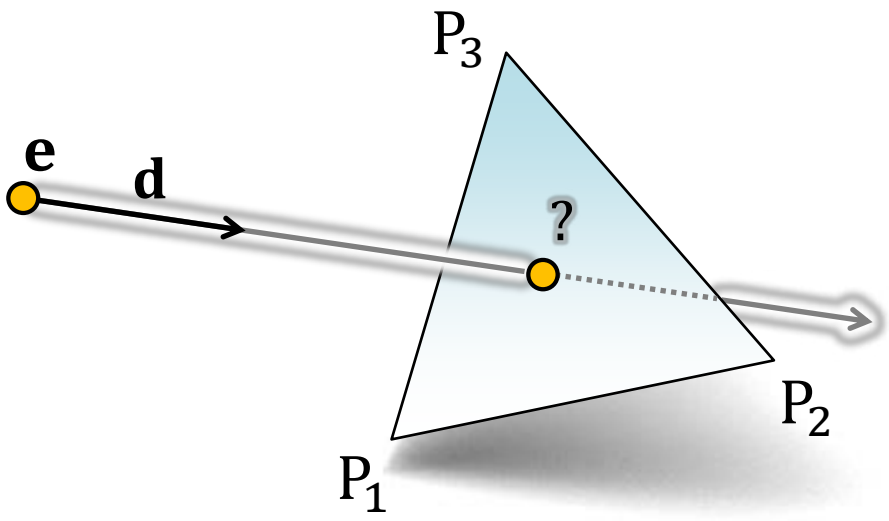
- definiert in 3D-Vektorräumen
  - $\mathbf{a} \times \mathbf{b} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} \times \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{pmatrix} = \mathbf{n}|\mathbf{a}||\mathbf{b}| \sin \phi$
  - $|\mathbf{n}| = 1$ und $\mathbf{n}$ steht senkrecht auf $\mathbf{a}$ und $\mathbf{b}$
- Anwendungsbeispiel: Berechnung der Oberflächennormale und Fläche eines Dreiecks
  - gegeben: $\Delta(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$
  - $\mathbf{n}' = (\mathbf{P}_2 - \mathbf{P}_1) \times (\mathbf{P}_3 - \mathbf{P}_1)$
  - $\mathbf{n} = \mathbf{n}'/|\mathbf{n}'|$
  - $A_\Delta = \frac{1}{2}|\mathbf{n}'|$ ← das Vorzeichen wird gleich interessant!



http://immersivemath.com/ila/ch04_vectorproduct/ch04.html – Abbildung 4.7 und 4.8

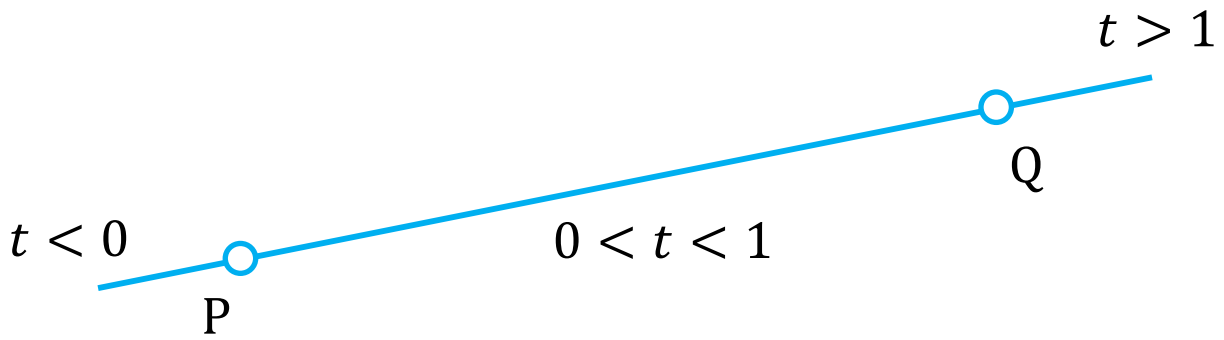
Nächste Schritte

- Schnitttest:
  liegt ein Punkt entlang einer Halbgerade (Strahl) auf dem Dreieck?



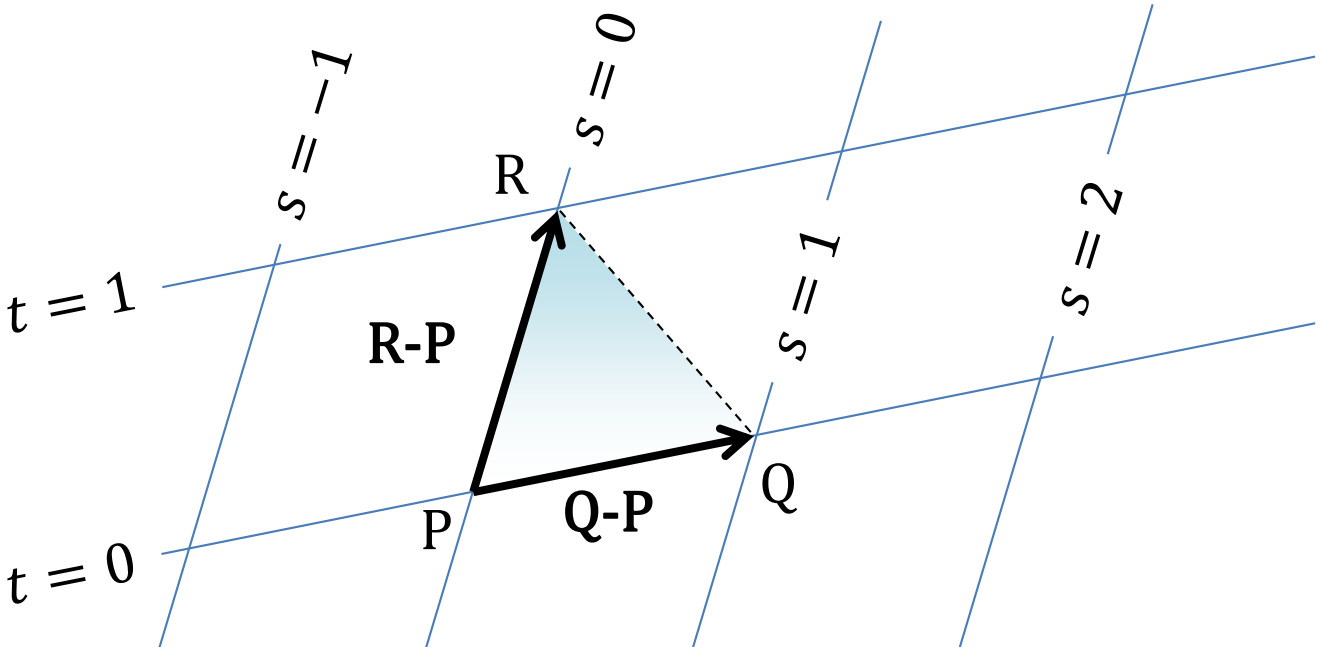
Parameterdarstellungen

- z.B. Halbgerade $g(t) = \mathbf{P} + t \cdot (\mathbf{Q} - \mathbf{P}), t \in \mathbb{R}_0^+$ (also $t \geq 0$)



Parameterdarstellungen

- z.B. Halbgerade $g(t) = \mathbf{P} + t \cdot (\mathbf{Q} - \mathbf{P}), t \in \mathbb{R}_0^+$ (also $t \geq 0$)
- Ebene: eindeutig bestimmt durch drei nicht-kollineare Punkte P, Q, R
  - aufspannenden Vektoren der Ebene sind $(\mathbf{Q} - \mathbf{P})$ und $(\mathbf{R} - \mathbf{P})$
  - Ebene $g(s,t) = \mathbf{P} + s \cdot (\mathbf{Q} - \mathbf{P}) + t \cdot (\mathbf{R} - \mathbf{P}), s, t \in \mathbb{R}$
  - Parallelogramm $g(s,t) = \mathbf{P} + s \cdot (\mathbf{Q} - \mathbf{P}) + t \cdot (\mathbf{R} - \mathbf{P}),$ $s, t \in \mathbb{R}_0^+ \wedge s, t \in [0;1]$
  - Dreieck $g(s,t) = \mathbf{P} + s \cdot (\mathbf{Q} - \mathbf{P}) + t \cdot (\mathbf{R} - \mathbf{P}),$ $s, t \in \mathbb{R}_0^+ \wedge s + t \leq 1$



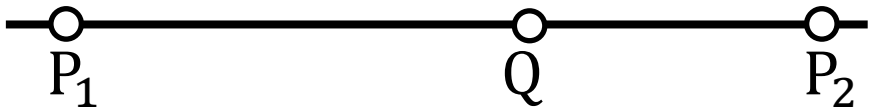
Baryzentrische Koordinaten

- Definition:
  - gegeben k Punkte $\mathrm{P}_1, \ldots, \mathrm{P}_k \in \mathbb{R}^n$ mit $k \leq n+1$
  - wenn ein Punkt Q in der Form

$$\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \cdots + \lambda_k \mathrm{P}_k$$

    mit $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 1$ (Affinkombination) dargestellt werden kann, so bezeichnet man $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ als die **baryzentrischen Koordinaten** von Q bzgl. der Basispunkte $\mathrm{P}_1, \ldots, \mathrm{P}_k$

- Beispiel $k = 2, n = 2$
  - alle Punkte $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2$ mit $\lambda_1 + \lambda_2 = 1$ liegen auf der Geraden durch P_1 und P_2
  - $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = (1 - \lambda_2)\mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = \mathrm{P}_1 + \lambda_2(\mathrm{P}_2 - \mathrm{P}_1)$, wegen $\lambda_1 = 1 - \lambda_2$



$\lambda_1 = {}^1/_3 \qquad \lambda_2 = {}^2/_3$

…wann liegt Q auf der Strecke $(\mathrm{P}_1, \mathrm{P}_2)$?

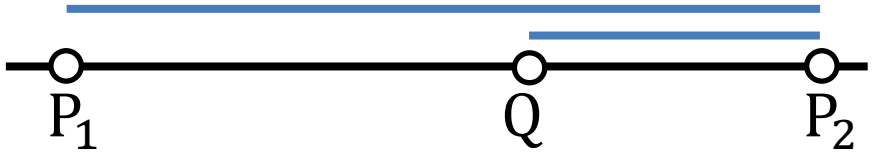
Baryzentrische Koordinaten

- Definition:
  - gegeben k Punkte $\mathrm{P}_1, \ldots, \mathrm{P}_k \in \mathbb{R}^n$ mit $k \leq n+1$
  - wenn ein Punkt Q in der Form

$$\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \cdots + \lambda_k \mathrm{P}_k$$

    mit $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 1$ (Affinkombination) dargestellt werden kann, so bezeichnet man $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ als die **baryzentrischen Koordinaten** von Q bzgl. der Basispunkte $\mathrm{P}_1, \ldots, \mathrm{P}_k$

- Beispiel $k = 2, n = 2$
  - alle Punkte $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2$ mit $\lambda_1 + \lambda_2 = 1$ liegen auf der Geraden durch P_1 und P_2
  - $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = (1-\lambda_2)\mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = \mathrm{P}_1 + \lambda_2(\mathrm{P}_2 - \mathrm{P}_1)$, wegen $\lambda_1 = 1 - \lambda_2$



$\lambda_1 = {}^1/_3 \qquad \lambda_2 = {}^2/_3$

$\lambda_1 = \overline{\mathrm{QP}_2} / \overline{\mathrm{P}_1\mathrm{P}_2}$

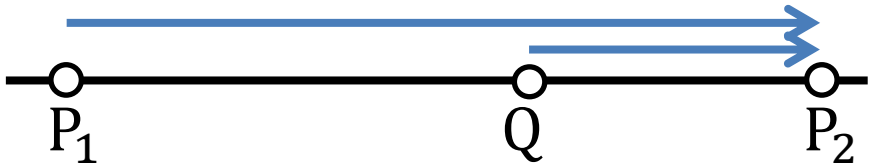
Baryzentrische Koordinaten

- Definition:
  - gegeben k Punkte $\mathrm{P}_1, \ldots, \mathrm{P}_k \in \mathbb{R}^n$ mit $k \leq n+1$
  - wenn ein Punkt Q in der Form

$$\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \cdots + \lambda_k \mathrm{P}_k$$

    mit $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 1$ (Affinkombination) dargestellt werden kann, so bezeichnet man $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ als die **baryzentrischen Koordinaten** von Q bzgl. der Basispunkte $\mathrm{P}_1, \ldots, \mathrm{P}_k$

- Beispiel $k = 2, n = 2$
  - alle Punkte $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2$ mit $\lambda_1 + \lambda_2 = 1$ liegen auf der Geraden durch P_1 und P_2
  - $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = (1-\lambda_2)\mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = \mathrm{P}_1 + \lambda_2(\mathrm{P}_2 - \mathrm{P}_1)$, wegen $\lambda_1 = 1 - \lambda_2$



$\lambda_1 = {}^1/_3 \qquad \lambda_2 = {}^2/_3$

$\lambda_1 = \overline{\mathrm{QP}_2} / \overline{\mathrm{P}_1\mathrm{P}_2}$

Baryzentrische Koordinaten

- Definition:
  - gegeben k Punkte $\mathrm{P}_1, \ldots, \mathrm{P}_k \in \mathbb{R}^n$ mit $k \leq n+1$
  - wenn ein Punkt Q in der Form

$$\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \cdots + \lambda_k \mathrm{P}_k$$

    mit $\lambda_1 + \lambda_2 + \cdots + \lambda_k = 1$ (Affinkombination) dargestellt werden kann, so bezeichnet man $(\lambda_1, \lambda_2, \ldots, \lambda_k)$ als die **baryzentrischen Koordinaten** von Q bzgl. der Basispunkte $\mathrm{P}_1, \ldots, \mathrm{P}_k$

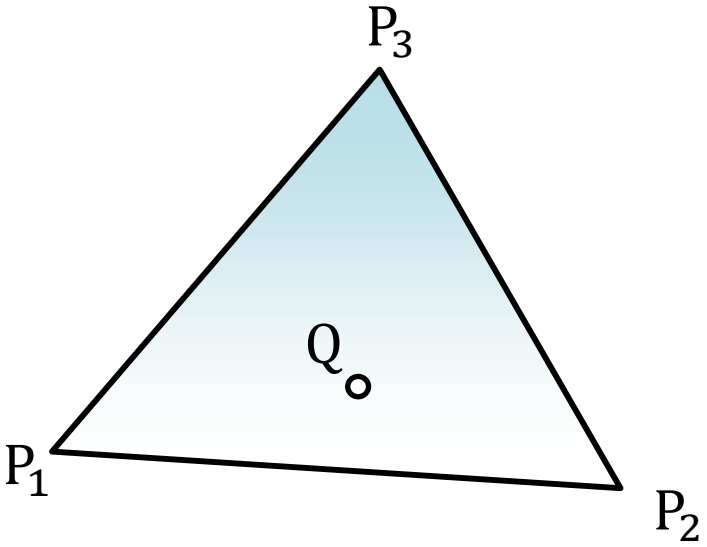
- Beispiel $k = 2, n = 2$
  - alle Punkte $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2$ mit $\lambda_1 + \lambda_2 = 1$ liegen auf der Geraden durch P_1 und P_2
  - $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = (1-\lambda_2)\mathrm{P}_1 + \lambda_2 \mathrm{P}_2 = \mathrm{P}_1 + \lambda_2(\mathrm{P}_2 - \mathrm{P}_1)$, wegen $\lambda_1 = 1 - \lambda_2$

$\lambda_1 = -{}^1/_3 \qquad \lambda_2 = {}^4/_3$

$\lambda_1 = -\overline{\mathrm{QP}_2}/\overline{\mathrm{P}_1\mathrm{P}_2}$

Baryzentrische Koordinaten

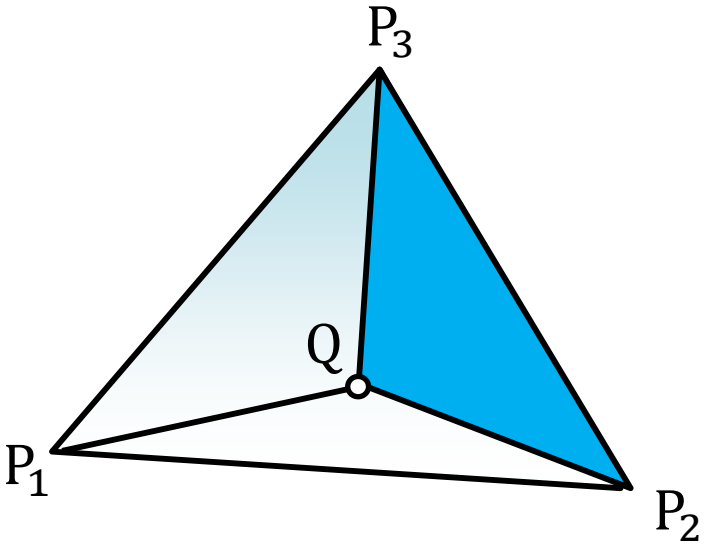
- Beispiel $k = 3, n = 2$ (Dreieck im $\mathbb{R}^2$)
  - alle Punkte Q mit $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \lambda_3 \mathrm{P}_3$ und $\lambda_1 + \lambda_2 + \lambda_3 = 1$ liegen innerhalb des Dreiecks $\triangle(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$ gdw. $\lambda_i \geq 0, i = 1..3$



- wie berechnet man $\lambda_1, \lambda_2, \lambda_3$?
  - es gilt natürlich wieder: $\lambda_1 = 1 - \lambda_2 - \lambda_3$
  - gegeben $\mathrm{Q}, \mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3 \rightarrow$ lineares Gleichungssystem mit zwei Gleichungen ($n = 2$) und zwei Unbekannten λ_2, λ_3

Baryzentrische Koordinaten

- Beispiel $k = 3, n = 2$ (Dreieck im $\mathbb{R}^2$)
  - alle Punkte Q mit $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \lambda_3 \mathrm{P}_3$ und $\lambda_1 + \lambda_2 + \lambda_3 = 1$ liegen innerhalb des Dreiecks $\triangle(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$ gdw. $\lambda_i \geq 0, i = 1..3$



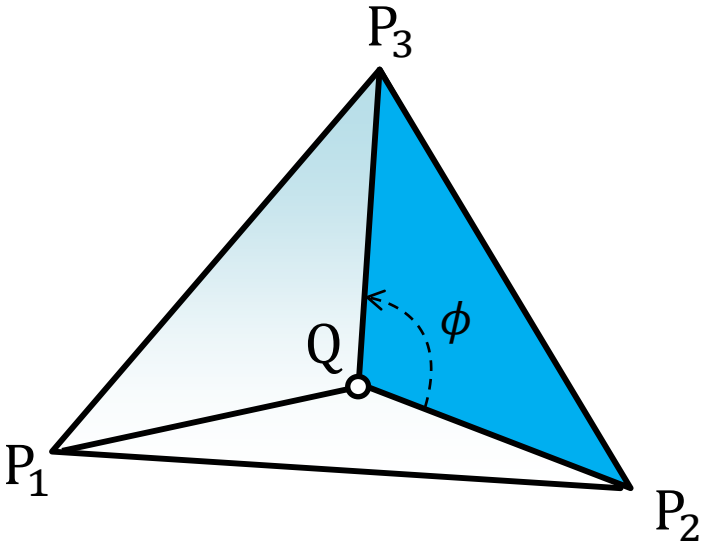
  - geometrische Interpretation der baryzentrischen Koordinaten

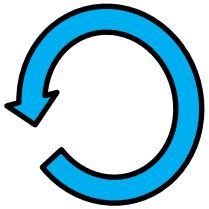
$$\lambda_1 = \frac{\mathrm{A}_\Delta(\mathrm{Q},\mathrm{P}_2,\mathrm{P}_3)}{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)} \qquad \lambda_2 = \frac{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{Q},\mathrm{P}_3)}{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)} \qquad \lambda_3 = \frac{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{Q})}{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)}$$

  - offensichtlich: $\lambda_1 + \lambda_2 + \lambda_3 = 1$

Baryzentrische Koordinaten

- Beispiel $k = 3, n = 2$ (Dreieck im $\mathbb{R}^2$)
  - alle Punkte Q mit $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \lambda_3 \mathrm{P}_3$ und $\lambda_1 + \lambda_2 + \lambda_3 = 1$ liegen innerhalb des Dreiecks $\triangle(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$ gdw. $\lambda_i \geq 0, i = 1..3$





$\mathrm{A}_\Delta(\mathrm{Q}, \mathrm{P}_2, \mathrm{P}_3) > 0$

- geometrische Interpretation der baryzentrischen Koordinaten

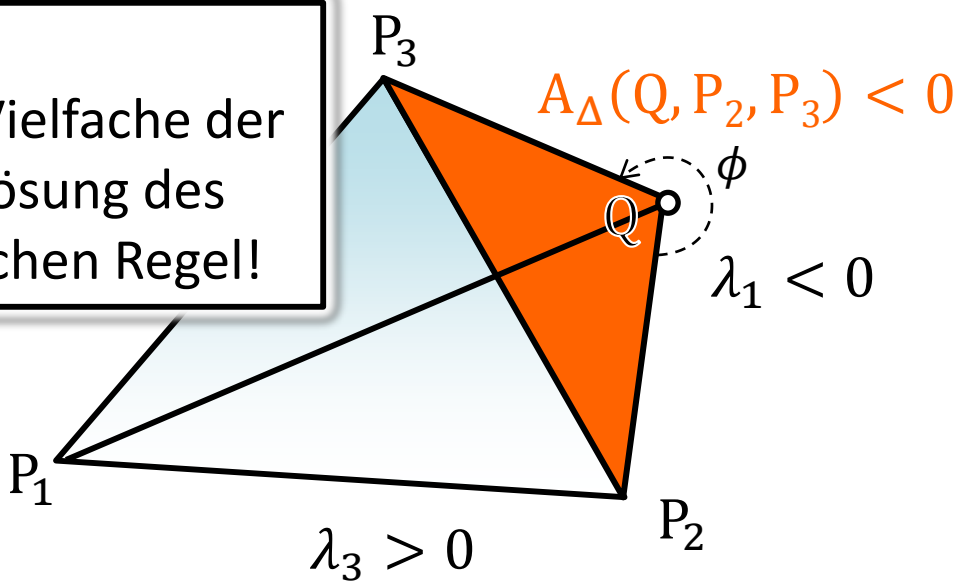
$$\lambda_1 = \frac{\mathrm{A}_\Delta(\mathrm{Q},\mathrm{P}_2,\mathrm{P}_3)}{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)} \qquad \lambda_2 = \frac{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{Q},\mathrm{P}_3)}{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)} \qquad \lambda_3 = \frac{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{Q})}{\mathrm{A}_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)}$$

- $\mathrm{A}_\Delta(\mathrm{Q}, \mathrm{P}_2, \mathrm{P}_3) = \frac{1}{2} |\mathbf{P_2} - \mathbf{Q}| |\mathbf{P_3} - \mathbf{Q}| \sin \phi$

Baryzentrische Koordinaten

- Beispiel $k = 3, n = 2$ (Dreieck im $\mathbb{R}^2$)
  - alle Punkte Q mit $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \lambda_3 \mathrm{P}_3$ und $\lambda_1 + \lambda_2 + \lambda_3 = 1$ liegen innerhalb des Dreiecks $\triangle(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$ gdw. $\lambda_i \geq 0, i = 1..3$

Bemerkung:
Flächeninhalte sind Vielfache der Determinanten bei Lösung des LGS mit der Cramerschen Regel!



- geometrische Interpretation der baryzentrischen Koordinaten

$$\lambda_1 = \frac{A_\Delta(Q,P_2,P_3)}{A_\Delta(P_1,P_2,P_3)} \qquad \lambda_2 = \frac{A_\Delta(P_1,Q,P_3)}{A_\Delta(P_1,P_2,P_3)} \qquad \lambda_3 = \frac{A_\Delta(P_1,P_2,Q)}{A_\Delta(P_1,P_2,P_3)}$$

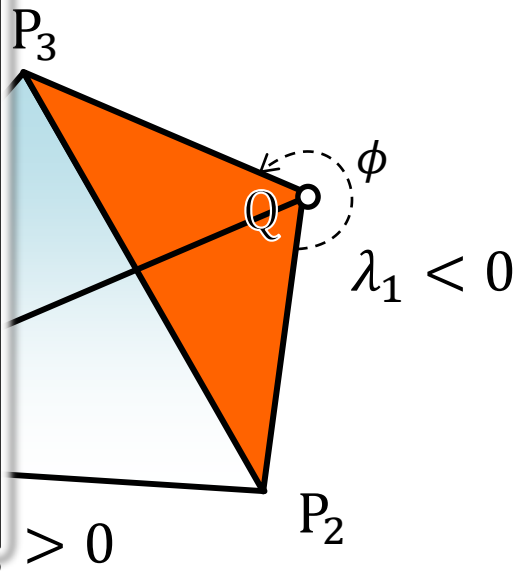
- $A_\Delta(Q, P_2, P_3) = \frac{1}{2}|\mathbf{P_2} - \mathbf{Q}||\mathbf{P_3} - \mathbf{Q}| \sin\phi < 0$ und daher $\lambda_1 + \lambda_2 + \lambda_3 = 1$

Baryzentrische Koordinaten

- Beispiel $k = 3, n = 2$ (Dreieck im $\mathbb{R}^2$)
  - alle Punkte Q mit $\mathrm{Q} = \lambda_1 \mathrm{P}_1 + \lambda_2 \mathrm{P}_2 + \lambda_3 \mathrm{P}_3$ und $\lambda_1 + \lambda_2 + \lambda_3 = 1$ liegen innerhalb des Dreiecks $\triangle(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$ gdw. $\lambda_i \geq 0, i = 1..3$

Hier gemäß unserer Definition:
$A_\Delta(\mathrm{Q}, \mathrm{P}_2, \mathrm{P}_3) < 0$ weil $\phi > \pi$.

Häufig in der CG: Anordnung der Eckpunkte (im  bzw. gegen  den Uhrzeigersinn im Bild) legt Orientierung eines Dreiecks fest oder definiert Vorder-/Rückseite



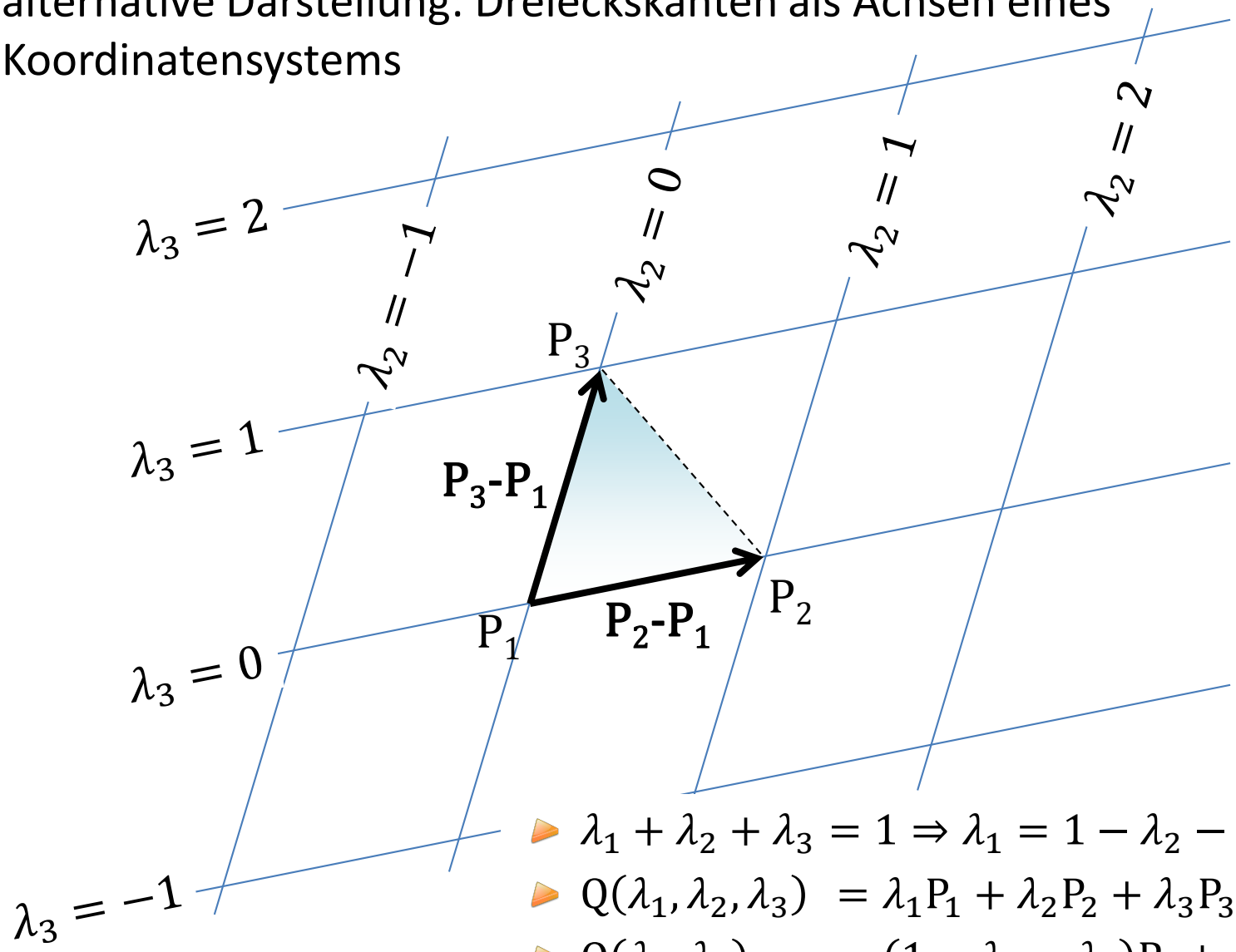
- geometrische Interpretation der baryzentrischen Koordinaten

$$\lambda_1 = \frac{A_\Delta(\mathrm{Q},\mathrm{P}_2,\mathrm{P}_3)}{A_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)} \qquad \lambda_2 = \frac{A_\Delta(\mathrm{P}_1,\mathrm{Q},\mathrm{P}_3)}{A_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)} \qquad \lambda_3 = \frac{A_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{Q})}{A_\Delta(\mathrm{P}_1,\mathrm{P}_2,\mathrm{P}_3)}$$

- $A_\Delta(\mathrm{Q},\mathrm{P}_2,\mathrm{P}_3) = \frac{1}{2}|\mathbf{P_2} - \mathbf{Q}||\mathbf{P_3} - \mathbf{Q}| \sin\phi < 0$ und daher $\lambda_1 + \lambda_2 + \lambda_3 = 1$

Baryzentrische Koordinaten

- alternative Darstellung: Dreieckskanten als Achsen eines Koordinatensystems



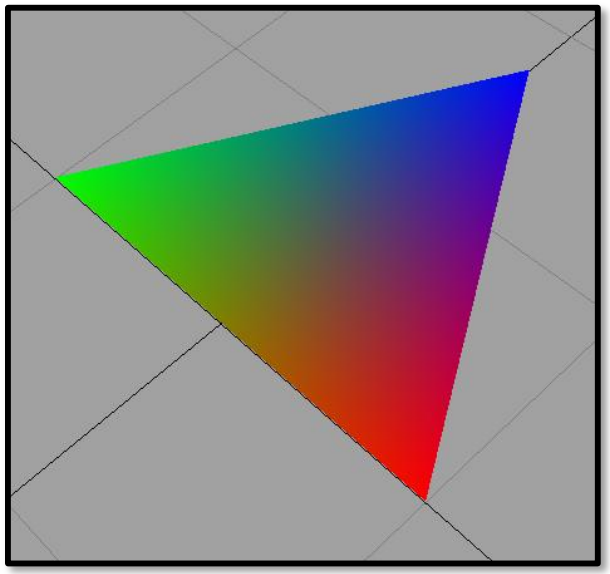
- $\lambda_1 + \lambda_2 + \lambda_3 = 1 \Rightarrow \lambda_1 = 1 - \lambda_2 - \lambda_3$
- $Q(\lambda_1, \lambda_2, \lambda_3) = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3$
- $Q(\lambda_2, \lambda_3) = (1 - \lambda_2 - \lambda_3) P_1 + \lambda_2 P_2 + \lambda_3 P_3$
  $= P_1 + \lambda_2 (P_2 - P_1) + \lambda_3 (P_3 - P_1)$

Anwendungen: Baryzentrische Koordinaten

- Beispiel: teste, ob ein Punkt Q innerhalb eines Dreiecks $\Delta(P_1, P_2, P_3)$ liegt
  - Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$, z.B. über Teilflächen und Dreiecksfläche
  - Punkt $Q \in \Delta(P_1, P_2, P_3)$ gdw. $\lambda_1, \lambda_2, \lambda_3 \geq 0$
  - im $\mathbb{R}^3$: nur wenn Q in der Ebene des Dreiecks liegt

- Beispiel: lineare Interpolation von (Farb-)Werten
  - gegeben: eine Farbe $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$ (als RGB-Tripel) zu jedem Eckpunkt eines Dreiecks $\Delta(P_1, P_2, P_3)$
  - gesucht: interpolierte Farbe $\mathbf{c_Q}$ an einem Punkt Q auf dem Dreieck
  - Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$, dann ist $\mathbf{c_Q} = \lambda_1 \mathbf{c_1} + \lambda_2 \mathbf{c_2} + \lambda_3 \mathbf{c_3}$

- **Interpolation mit baryzentrischen Koordinaten = lineare Interpolation**

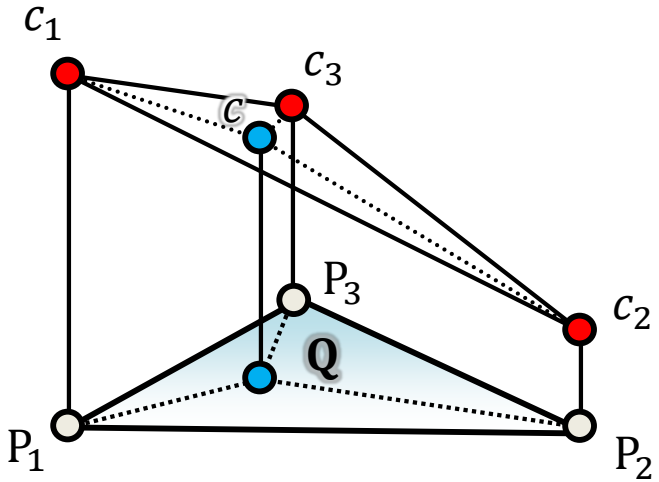


Anwendungen: Baryzentrische Koordinaten

- Beispiel: teste, ob ein Punkt Q innerhalb eines Dreiecks $\Delta(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$ liegt
  - Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$, z.B. über Teilflächen und Dreiecksfläche
  - Punkt $\mathrm{Q} \in \Delta(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$ gdw. $\lambda_1, \lambda_2, \lambda_3 \geq 0$
  - im $\mathbb{R}^3$: nur wenn Q in der Ebene des Dreiecks liegt

- Beispiel: lineare Interpolation von (Farb-)Werten
  - gegeben: eine Farbe $\mathbf{c_1}, \mathbf{c_2}, \mathbf{c_3}$ (als RGB-Tripel) zu jedem Eckpunkt eines Dreiecks $\Delta(\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3)$
  - gesucht: interpolierte Farbe $\mathbf{c_Q}$ an einem Punkt Q auf dem Dreieck
  - Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$, dann ist $\mathbf{c_Q} = \lambda_1 \mathbf{c_1} + \lambda_2 \mathbf{c_2} + \lambda_3 \mathbf{c_3}$

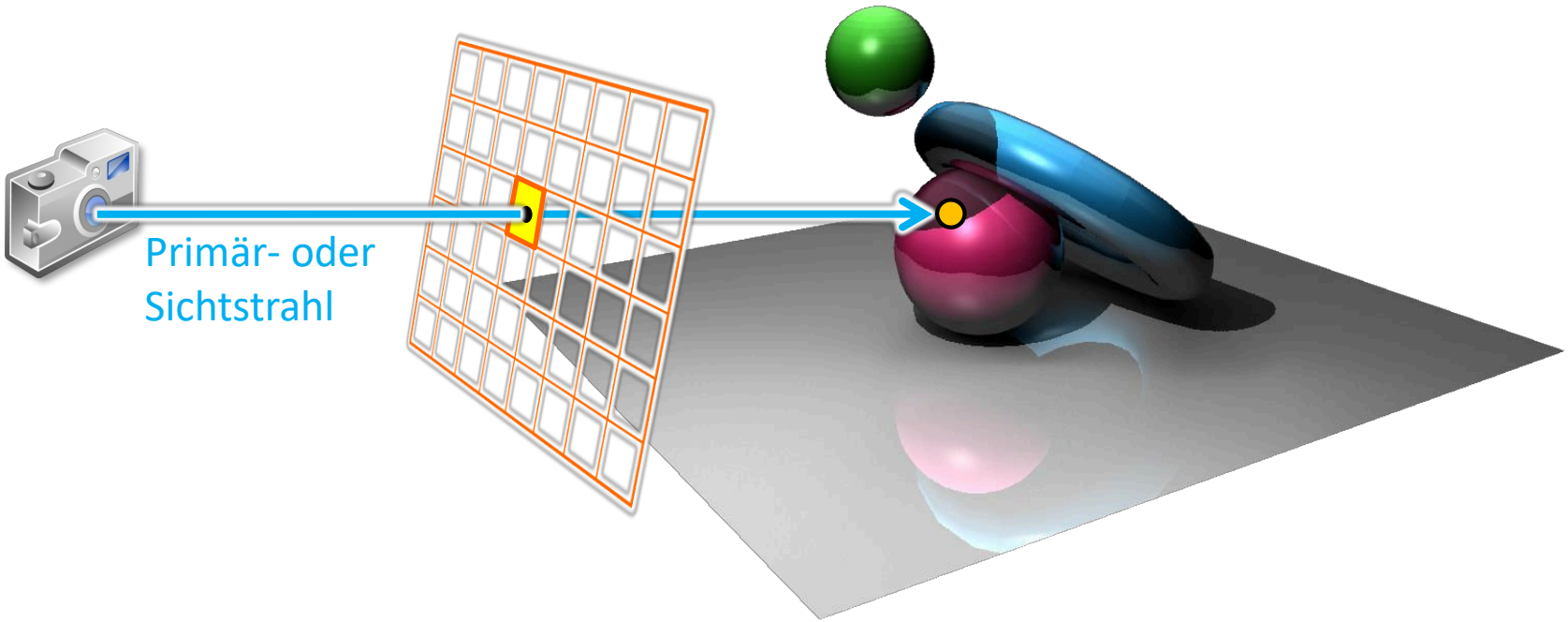
- **Interpolation mit baryzentrischen Koordinaten = lineare Interpolation**



Raytracing

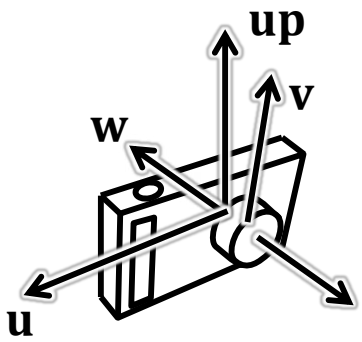
Schritte

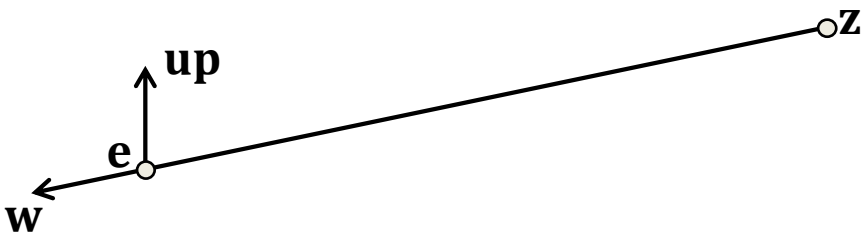
- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting):
  finde Dreieck, das den Sichtstrahl am nächsten zur Kamera schneidet
- Schattierung und Beleuchtungsberechnung (shading)
- Sekundärstrahlen für Spiegelung und Transmission



Erzeugung von Sichtstrahlen (ray generation)

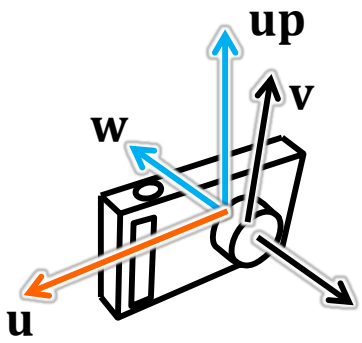
- virtuelle Kamera (Lochkamera-Modell) definiert durch
  - Position (= Projektionszentrum, „eye“) $\mathbf{e}$
  - Zielpunkt $\mathbf{z}$
  - negative Blickrichtung $\mathbf{w} = (\mathbf{e} - \mathbf{z})/|\mathbf{e} - \mathbf{z}|$
  - „Up-Vektor“ $\mathbf{up}$ ($|\mathbf{up}| = 1$)
  - $\Rightarrow \mathbf{u} = \mathbf{up} \times \mathbf{w}$ und $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
  - Normalisiere die Vektoren $\mathbf{u}$ und $\mathbf{v}$

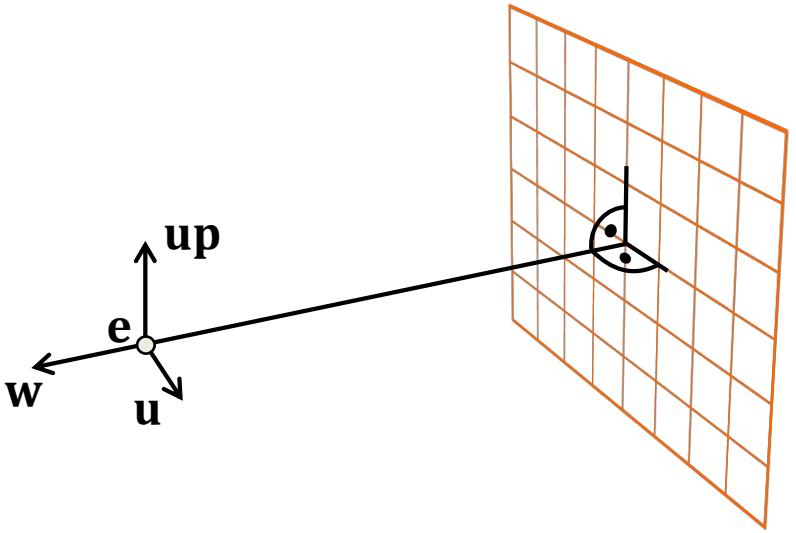




Erzeugung von Sichtstrahlen (ray generation)

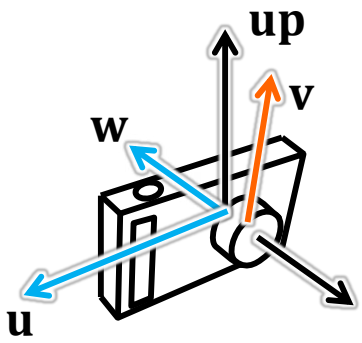
- virtuelle Kamera (Lochkamera-Modell) definiert durch
  - Position (= Projektionszentrum, „eye“) $\mathbf{e}$
  - Zielpunkt $\mathbf{z}$
  - negative Blickrichtung $\mathbf{w} = (\mathbf{e} - \mathbf{z})/|\mathbf{e} - \mathbf{z}|$
  - „Up-Vektor“ $\mathbf{up}$ ($|\mathbf{up}| = 1$)
  - $\Rightarrow \mathbf{u} = \mathbf{up} \times \mathbf{w}$ und $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
  - Normalisiere die Vektoren $\mathbf{u}$ und $\mathbf{v}$

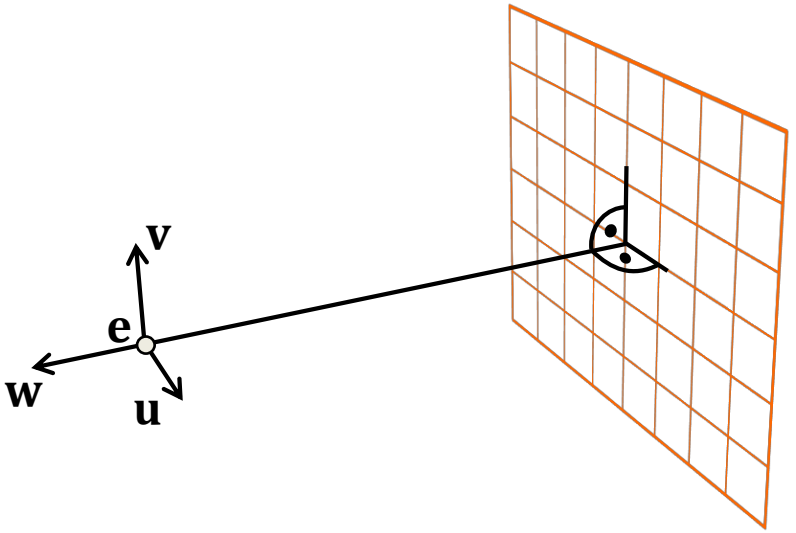




Erzeugung von Sichtstrahlen (ray generation)

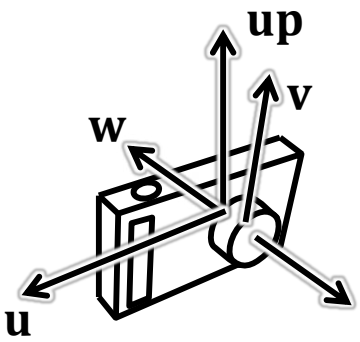
- virtuelle Kamera (Lochkamera-Modell) definiert durch
  - Position (= Projektionszentrum, „eye“) $\mathbf{e}$
  - Zielpunkt $\mathbf{z}$
  - negative Blickrichtung $\mathbf{w} = (\mathbf{e} - \mathbf{z})/|\mathbf{e} - \mathbf{z}|$
  - „Up-Vektor“ $\mathbf{up}$ ($|\mathbf{up}| = 1$)
  - $\Rightarrow \mathbf{u} = \mathbf{up} \times \mathbf{w}$ und $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
  - Normalisiere die Vektoren $\mathbf{u}$ und $\mathbf{v}$



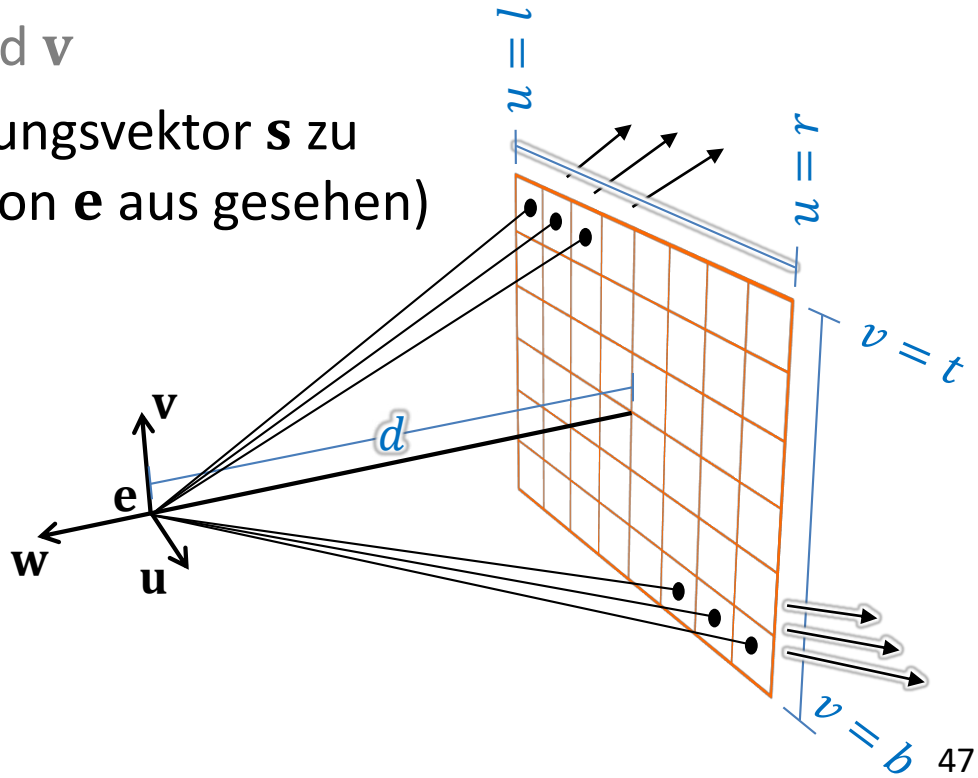


Erzeugung von Sichtstrahlen (ray generation)

- virtuelle Kamera (Lochkamera-Modell) definiert durch
  - Position (= Projektionszentrum, „eye“) $\mathbf{e}$
  - Zielpunkt $\mathbf{z}$
  - negative Blickrichtung $\mathbf{w} = (\mathbf{e} - \mathbf{z})/|\mathbf{e} - \mathbf{z}|$
  - „Up-Vektor“ $\mathbf{up}$ ($|\mathbf{up}| = 1$)
  - $\Rightarrow \mathbf{u} = \mathbf{up} \times \mathbf{w}$ und $\mathbf{v} = \mathbf{w} \times \mathbf{u}$
  - Normalisiere die Vektoren $\mathbf{u}$ und $\mathbf{v}$

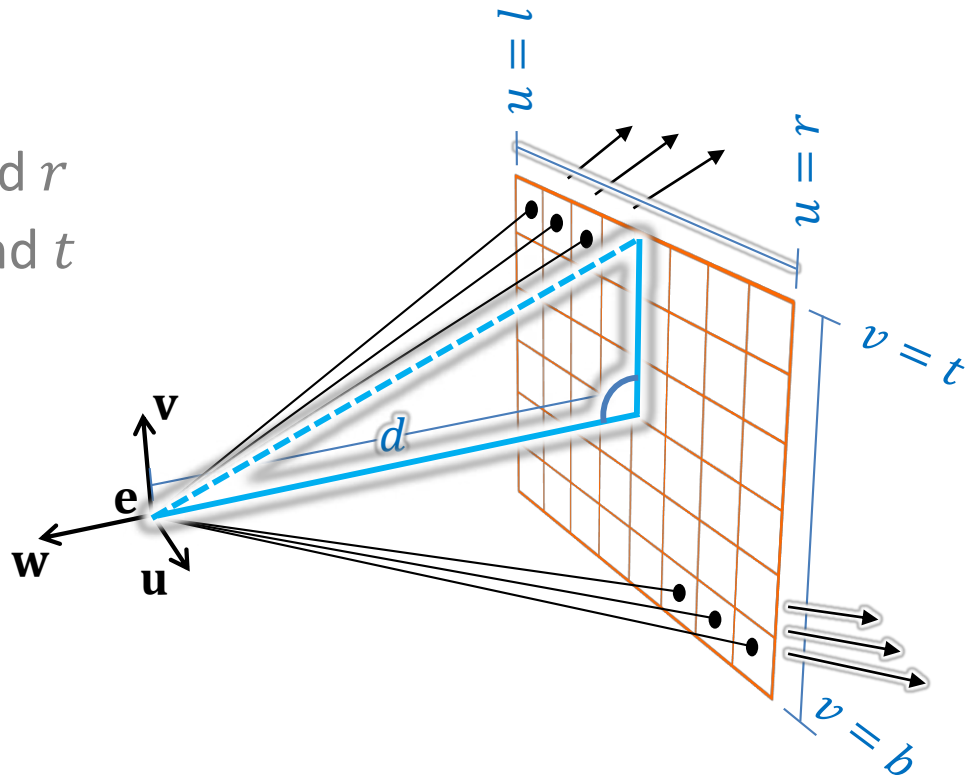


- Erzeugen von Sichtstrahlen: Richtungsvektor $\mathbf{s}$ zu Punkten auf der Bildebene (also von $\mathbf{e}$ aus gesehen)
- Bildebene gegeben durch
  Abstand zur Kamera: d
  linker/rechter Rand: l und r
  unterer/oberer Rand: b und t
  - $\mathbf{s} = u \cdot \mathbf{u} + v \cdot \mathbf{v} - d \cdot \mathbf{w}$,
    mit $u \in [l, r]$ und $v \in [b, t]$



Erzeugung von Sichtstrahlen (ray generation)

- Bildebene gegeben durch
  - Abstand zur Kamera: d
  - linker/rechter Rand: l und r
  - unterer/oberer Rand: b und t

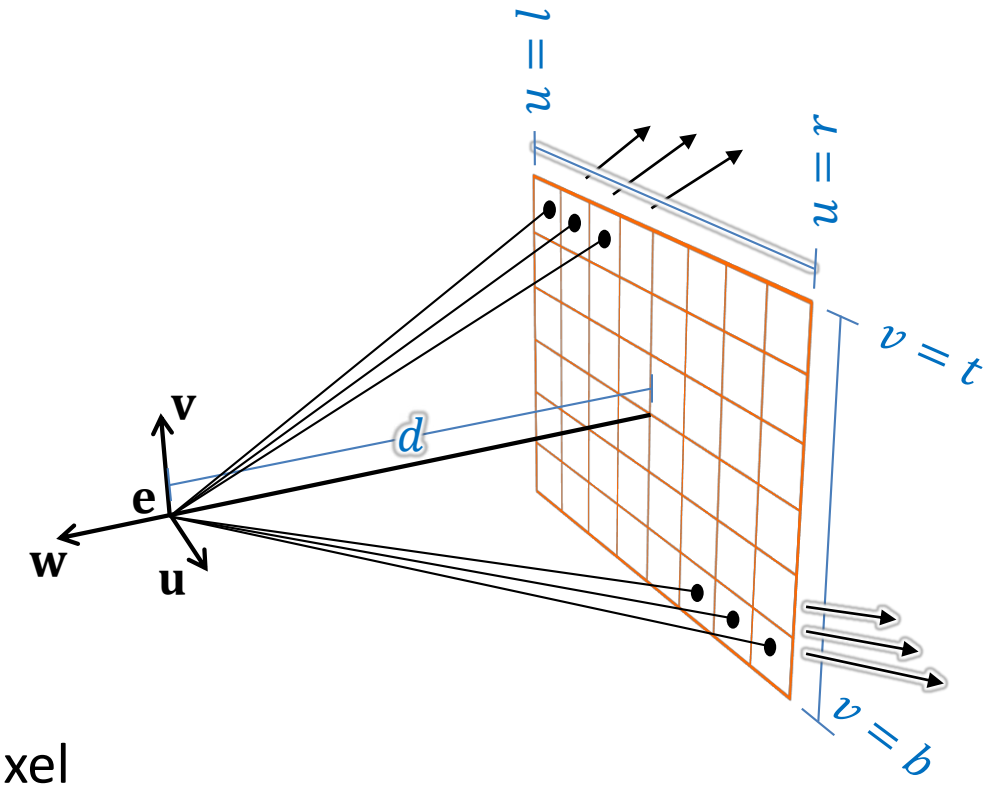


- Richtungsvektor $\mathbf{s}$ zu Zielpunkten auf der Bildebene
  - $\mathbf{s} = u \cdot \mathbf{u} + v \cdot \mathbf{v} - d \cdot \mathbf{w}$, mit $u \in [l, r]$ und $v \in [b, t]$

- Bsp. typisches Sichtfeld ist 90° und symmetrisch, d.h. $l = -r \ \wedge \ b = -t$
  - $\tan 90°/2 = t/d$ (d wird festgelegt) und
  - $(r - l)/(t - b) = r/t$ „Aspect Ratio“
  - Aspect Ratio = Verhältnis Breite zu Höhe des Bildschirms (typische Werte: 4:3, 16:9, 16:10, 21:9, …)

Erzeugung von Sichtstrahlen (ray generation)

- Richtungsvektor $\mathbf{s}$
  - $\mathbf{s} = u \cdot \mathbf{u} + v \cdot \mathbf{v} - d \cdot \mathbf{w}$,
    mit $u \in [l, r]$ und $v \in [b, t]$
- Strahlgleichung $\mathbf{r}(t) = \mathbf{e} + t\mathbf{s}$
  - $t = 0 \rightarrow \mathbf{e}$ (Kameraposition)
  - $t = 1 \rightarrow \mathbf{e} + \mathbf{s}$ (Pixelmitte)
  - i.d.R. $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}$,
    mit $\mathbf{d} = \mathbf{s}/|\mathbf{s}|$

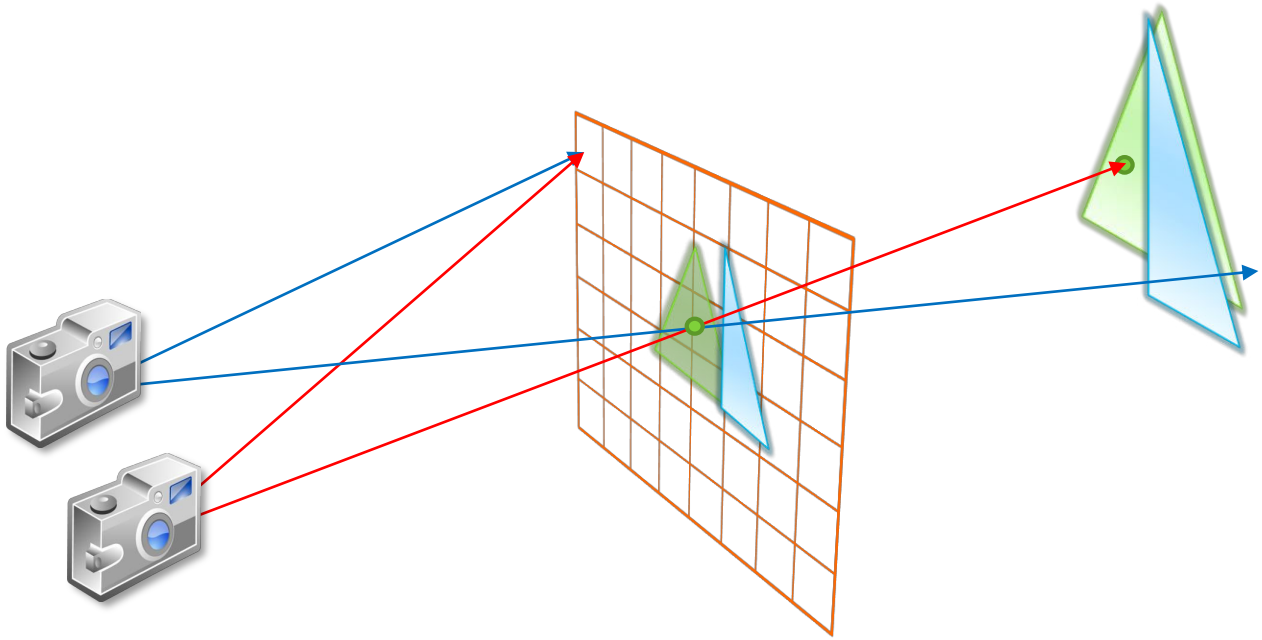


- Pseudocode: ein Zielpunkt pro Pixel

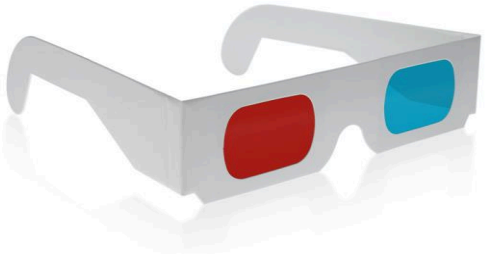
```
for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    ...
} }
```

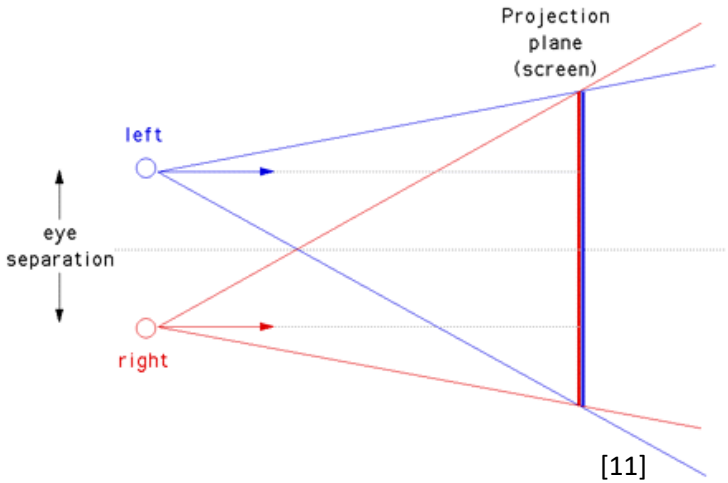
Raytracing, Stereo-Rendering

- Grundidee: für jedes Auge ein Bild



Bildebene



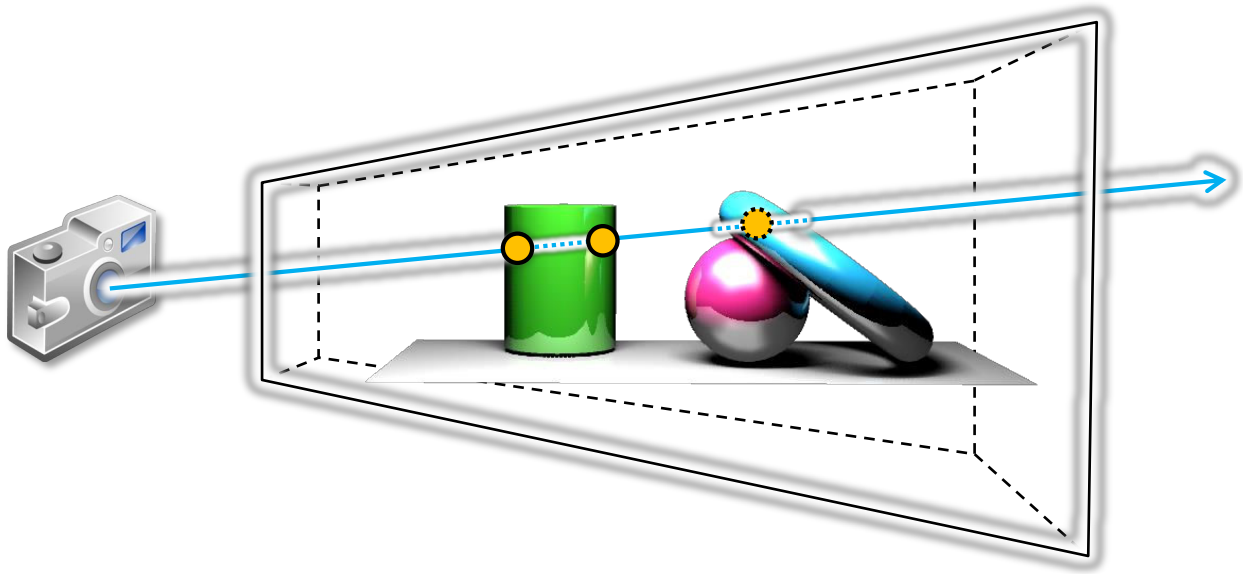


[11]

Raytracing

Schritte

- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting):
  finde geometrisches Primitiv/Objekt (und Schnittpunkt damit),
  das den Sichtstrahl am nächsten zur Kamera schneidet
- Schattierung und Beleuchtungsberechnung (shading)
- Sekundärstrahlen für Spiegelung und Transmission



Raytracing Pseudocode

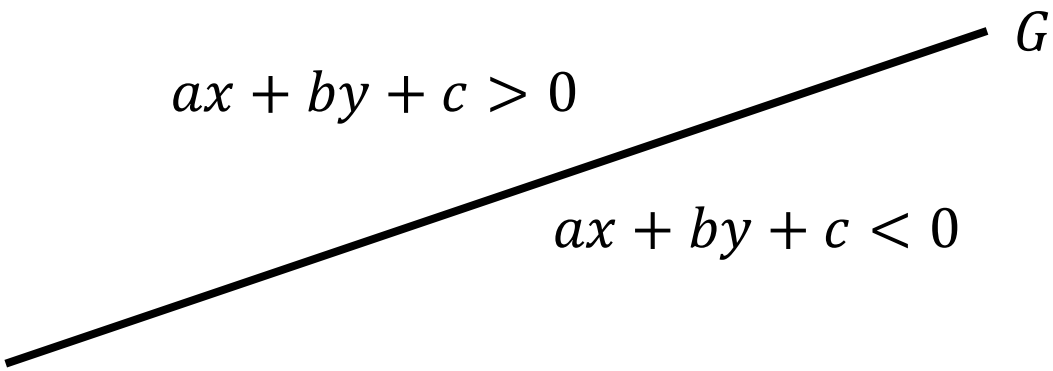
```
for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t' = intersect( object, e, d );
      if ( t' > 0 && t' < t ) {
        intersection = object;
        t = t';
      }
    }
    ...
} }
```

- Achtung: uns interessieren nur Schnittpunkte vor der Kamera (`t'>0`), und von diesen der nächste in Strahlrichtung (`t'<t`)

Parameter-, explizite und implizite Darstellung

- **Parameterdarstellung**
  alle Punkte eines geometrischen Gebildes (Linie, Ebene etc.) kann man durch Einsetzen aller zulässigen Parameterwerte gewinnen
  - Bsp.: Kreis in 2D: $x(t) = r \cdot \cos t, y(t) = r \cdot \sin t, t \in [0..2\pi)$
- **Explizite Darstellung**
  - Bsp.: Kurve in 2D: $y = f(x)$, Fläche in 3D: $z = f(x, y)$
- **Implizite Darstellung**
  die Punkte die zu einem geometrischen Gebilde gehören bilden die Lösungsgesamtheit eines Systems von Gleichungen
  - Bsp.: Gerade in 2D

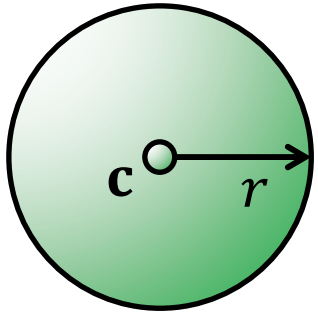
$$G = \{(x, y) | ax + by + c = 0 \wedge a, b, c \in \mathbb{R}\}$$



Implizite Darstellung von Kugel und Torus

- Kugel: alle Punkte auf der Kugeloberfläche haben den Abstand r vom Mittelpunkt $\mathbf{c} = (c_x, c_y, c_z)$

$$\begin{aligned} \mathrm{K} &= \{(x,y,z) \mid |(x-c_x, y-c_y, z-c_z)| = r\} \\ &= \left\{(x,y,z) \,\middle|\, |(x-c_x, y-c_y, z-c_z)|^2 = r^2\right\} \end{aligned}$$

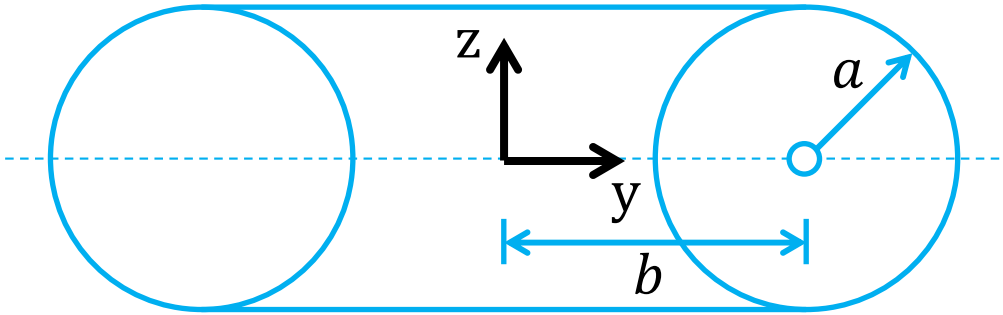


- Torus (nur damit Sie es mal gesehen haben):
  - implizit:

$$(x^2 + y^2 + z^2 + b^2 - a^2)^2 = 4b^2(x^2 + y^2)$$

  - parametrisch:

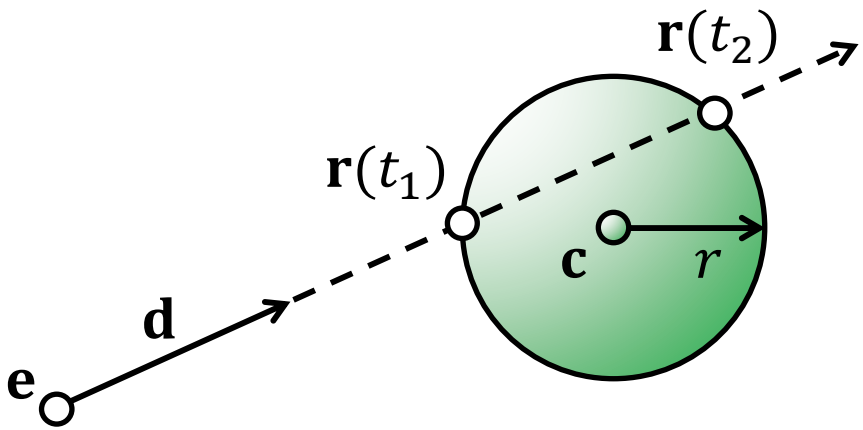
$$\begin{aligned} x &= -\sin(u)\,(b + \mathrm{a} \cdot \cos(v)) \\ y &= \cos(u)\,(b + \mathrm{a} \cdot \cos(v)) \\ z &= \mathrm{a} \cdot \sin(v) \\ u, v &\in [0..2\pi) \end{aligned}$$



Schnittpunktberechnung

Strahl-Kugel-Schnitt

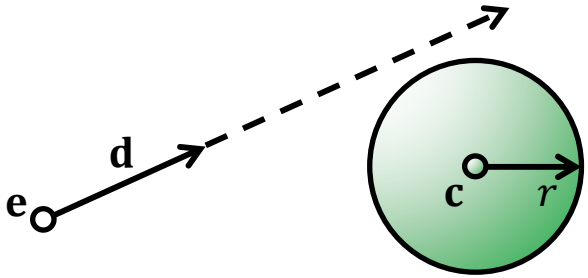
- Strahlgleichung (parametrisch) $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}$
- Kugel (implizite Darstellung) $|\mathbf{x} - \mathbf{c}|^2 - r^2 = 0$
- Einsetzen:
  - $|\mathbf{r}(t) - \mathbf{c}|^2 - r^2 = 0$
  - $|\mathbf{e} + t\mathbf{d} - \mathbf{c}|^2 - r^2 = 0 \quad (|\mathbf{x}|^2 = \mathbf{x} \cdot \mathbf{x})$
  - $(\mathbf{e} + t\mathbf{d} - \mathbf{c}) \cdot (\mathbf{e} + t\mathbf{d} - \mathbf{c}) - r^2 = 0$
  - $\underbrace{(\mathbf{e} - \mathbf{c}) \cdot (\mathbf{e} - \mathbf{c}) - r^2}_{const.} + \underbrace{2(t\mathbf{d} \cdot (\mathbf{e} - \mathbf{c}))}_{t(2\mathbf{d} \cdot (\mathbf{e} - \mathbf{c}))} + \underbrace{(t\mathbf{d}) \cdot (t\mathbf{d})}_{t^2(\mathbf{d} \cdot \mathbf{d})} = 0$



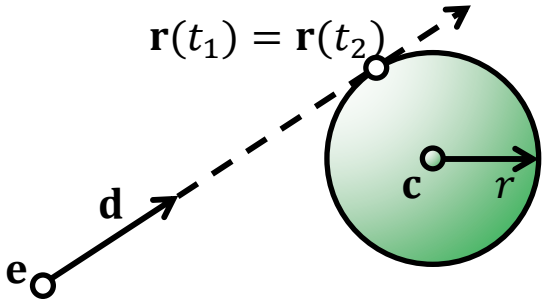
Schnittpunktberechnung

Strahl-Kugel-Schnitt

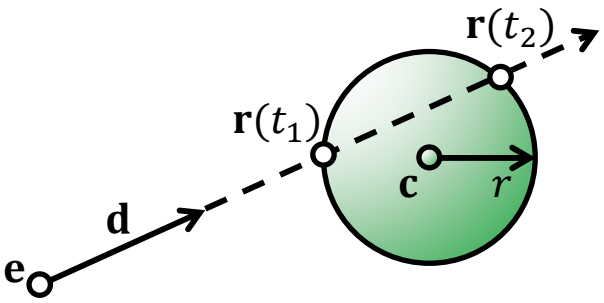
- $\underbrace{(\mathbf{e}-\mathbf{c})\cdot(\mathbf{e}-\mathbf{c})-r^2}_{const.} + \underbrace{2(t\mathbf{d}\cdot(\mathbf{e}-\mathbf{c}))}_{t(2\mathbf{d}\cdot(\mathbf{e}-\mathbf{c}))} + \underbrace{(t\mathbf{d})\cdot(t\mathbf{d})}_{t^2(\mathbf{d}\cdot\mathbf{d})} = 0$
- quadratische Gleichung: Mitternachtsformel
  - $t_{1,2} = \frac{-b \pm \sqrt{b^2-4ac}}{2a}$
    mit $a = \mathbf{d}\cdot\mathbf{d}$, $b = 2\mathbf{d}\cdot(\mathbf{e}-\mathbf{c})$, $c = (\mathbf{e}-\mathbf{c})\cdot(\mathbf{e}-\mathbf{c})-r^2$
  - Diskriminante b^2-4ac (Achtung: bei 2 Lösungen kann $t_1 < 0$ sein)



Diskriminante < 0
→ keine Lösung



Diskriminante $= 0$
→ eine Lösung



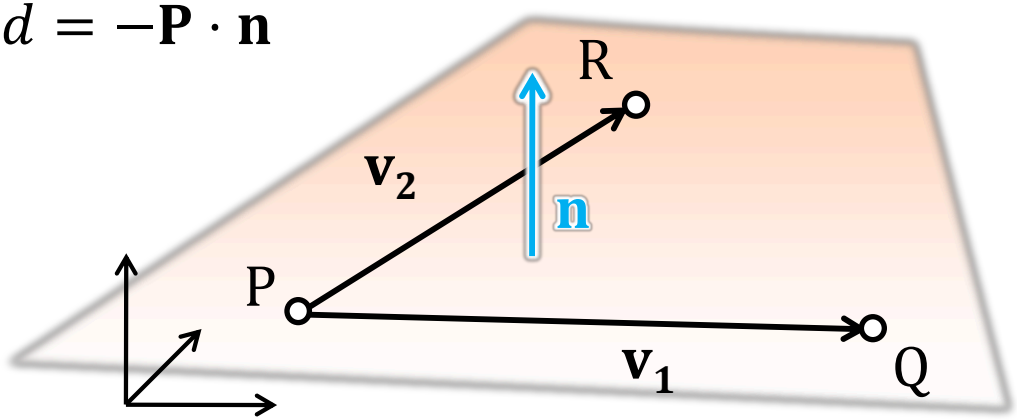
Diskriminante > 0
→ zwei Lösungen

Implizite und Parameter-Darstellung einer Ebene

- eine Ebene E im $\mathbb{R}^3$ hat die implizite Form

$$\mathrm{E} = \{(x, y, z) | ax + by + cz + d = 0 \wedge a, b, c, d \in \mathbb{R}, \neg a, b, c = 0\}$$

- Parameterdarstellung → implizite Darstellung
  - geg. Ebene durch drei nicht-kollineare Punkte P, Q, R (alternativ: Punkt P und Normale $\mathbf{n}$)
  - ges. Werte a, b, c, d der Ebenengleichung
  - aufspannende Vektoren: $\mathbf{v_1} = \mathbf{Q} - \mathbf{P}$ und $\mathbf{v_2} = \mathbf{R} - \mathbf{P}$
  - Normale der Ebene: $\mathbf{n} = \mathbf{v_1} \times \mathbf{v_2}$
  - $a = n_x$, $b = n_y$, $c = n_z$ und $d = -\mathbf{P} \cdot \mathbf{n}$

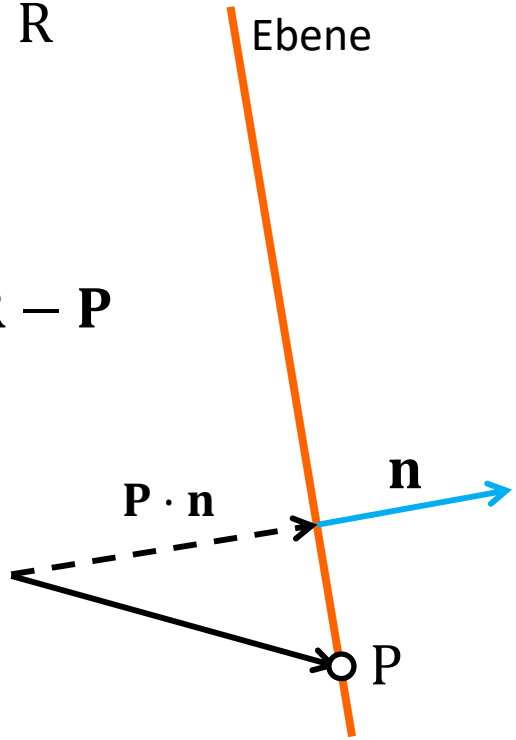


Implizite und Parameter-Darstellung einer Ebene

- eine Ebene E im $\mathbb{R}^3$ hat die implizite Form

$$\mathrm{E} = \{(x, y, z) | ax + by + cz + d = 0 \wedge a, b, c, d \in \mathbb{R}, \neg a, b, c = 0\}$$

- Parameterdarstellung → implizite Darstellung
  - geg. Ebene durch drei nicht-kollineare Punkte P, Q, R (alternativ: Punkt P und Normale $\mathbf{n}$)
  - ges. Werte a, b, c, d der Ebenengleichung
  - aufspannende Vektoren: $\mathbf{v_1} = \mathbf{Q} - \mathbf{P}$ und $\mathbf{v_2} = \mathbf{R} - \mathbf{P}$
  - Normale der Ebene: $\mathbf{n} = \mathbf{v_1} \times \mathbf{v_2}$
  - $a = n_x$, $b = n_y$, $c = n_z$ und $d = -\mathbf{P} \cdot \mathbf{n}$
  - ist $|\mathbf{n}| = 1$, dann spricht man von der Hesse-Normalform (HNF)
  - Überführung in HNF durch teilen von a, b, c, d durch $|\mathbf{n}|$



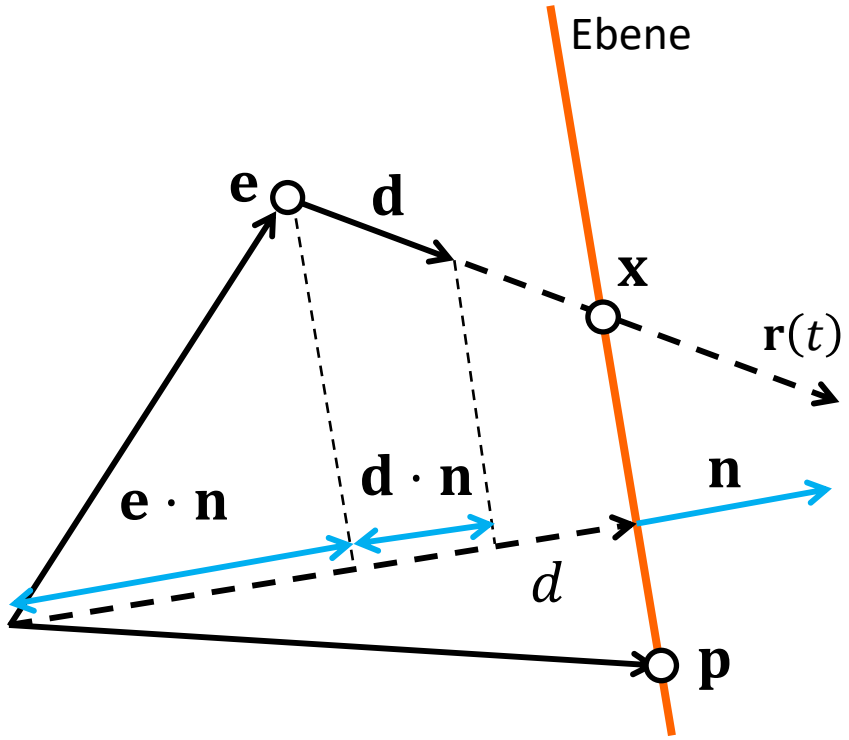
Schnittpunktberechnung mit Ebene

Strahl-Ebene-Schnitt

- Strahl (parametrische Repr.) $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}, |\mathbf{d}| = 1$
- Ebene (implizite Repr., HNF) $\mathbf{x} \cdot \mathbf{n} - d = 0, \ |\mathbf{n}| = 1$
  - Normalenvektor: $\mathbf{n}$
  - Abstand vom Ursprung: d (wg. $|\mathbf{n}| = 1$)
    (Achtung: Vorzeichen anders als auf letzter Folie, weil anschaulicher)
- Lösung durch Einsetzen
  - Einsetzen:

    $(\mathbf{e} + t\mathbf{d}) \cdot \mathbf{n} - d = 0$

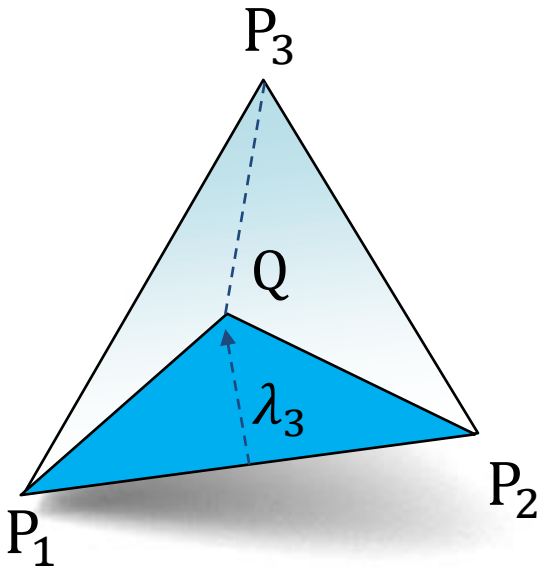
    $\mathbf{e} \cdot \mathbf{n} + t(\mathbf{d} \cdot \mathbf{n}) - d = 0$
  - Auflösen ergibt: $t = \frac{d - \mathbf{e} \cdot \mathbf{n}}{\mathbf{d} \cdot \mathbf{n}}$
  - $\mathbf{d} \cdot \mathbf{n} = 0 \Leftrightarrow$
    Strahl und Ebene sind parallel



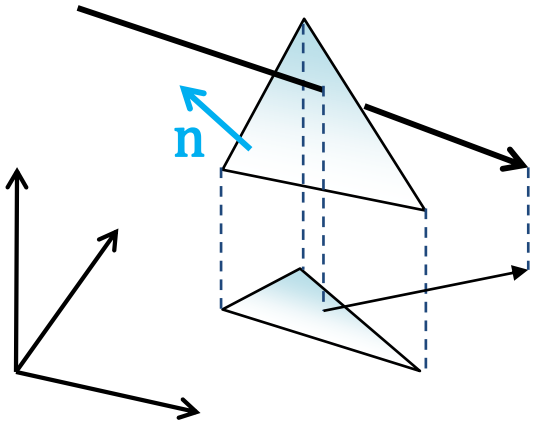
Schnittpunktberechnung mit Dreieck

Strahl-Dreieck-Schnitt (anschaulich, so macht man es aber nicht!)

- baryzentrische Koordinaten eines potentiellen Schnittpunkts
  - $Q = \lambda_1 P_1 + \lambda_2 P_2 + \lambda_3 P_3$
    mit $\lambda_1 + \lambda_2 + \lambda_3 = 1$
  - $\lambda_3 = \triangle (Q P_1 P_2) / \triangle (P_1 P_2 P_3)$, etc.
  - Punkt in Dreieck, falls alle $\lambda_i \geq 0$



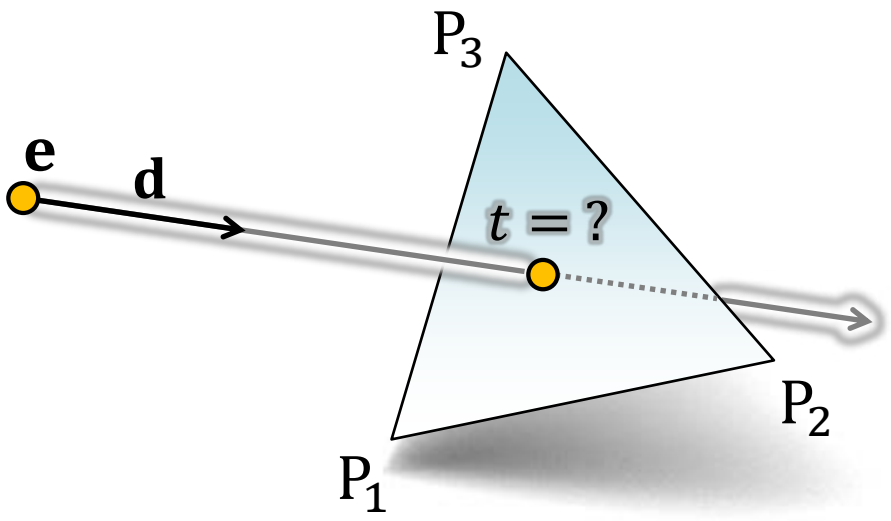
- möglicher Schnitttest:
  - Ebene durch Dreieck legen
  - berechne Schnittpunkt mit Ebene
  - berechne baryzentrische Koordinaten
    - z.B. auch über 2D-Punkte in Ebene:
      Projektion in Hauptebene
      mit max. Normalenkoordinate
  - teste $\lambda_1, \lambda_2, \lambda_3 \geq 0$



Schnittpunktberechnung mit Dreieck

Strahl-Dreieck-Schnitt (direkt, so wird es gemacht!)

- Strahlgleichung (allg.): $\mathbf{r}(t) = \mathbf{e} + t\mathbf{d}$, $|\mathbf{d}| = 1$
- Dreieck mit Eckpunkten $\mathrm{P}_1, \mathrm{P}_2, \mathrm{P}_3$
- existiert ein Schnittpunkt? wenn ja, für welches t?

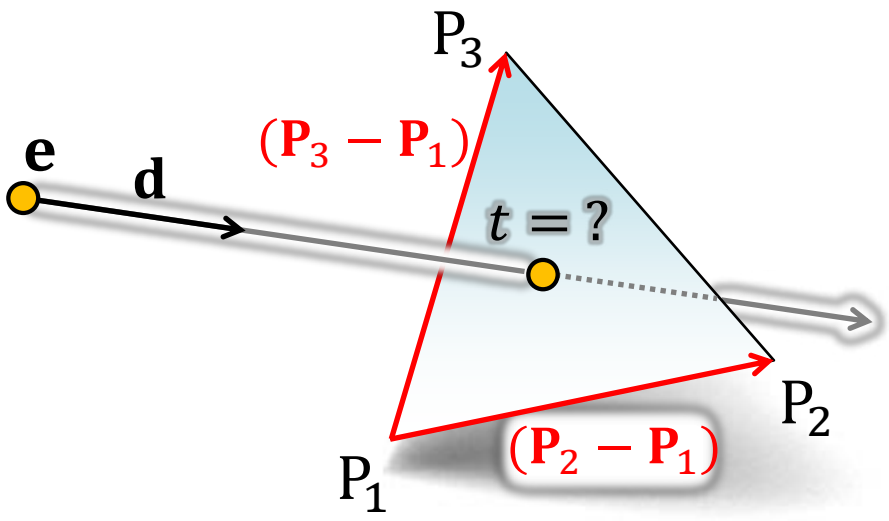


Schnittpunktberechnung

Strahl-Dreieck-Schnitt

- jeder Punkt in der Ebene des Dreiecks lässt sich beschreiben durch

$$Q(\lambda_2, \lambda_3) = P_1 + \lambda_2(P_2 - P_1) + \lambda_3(P_3 - P_1)$$



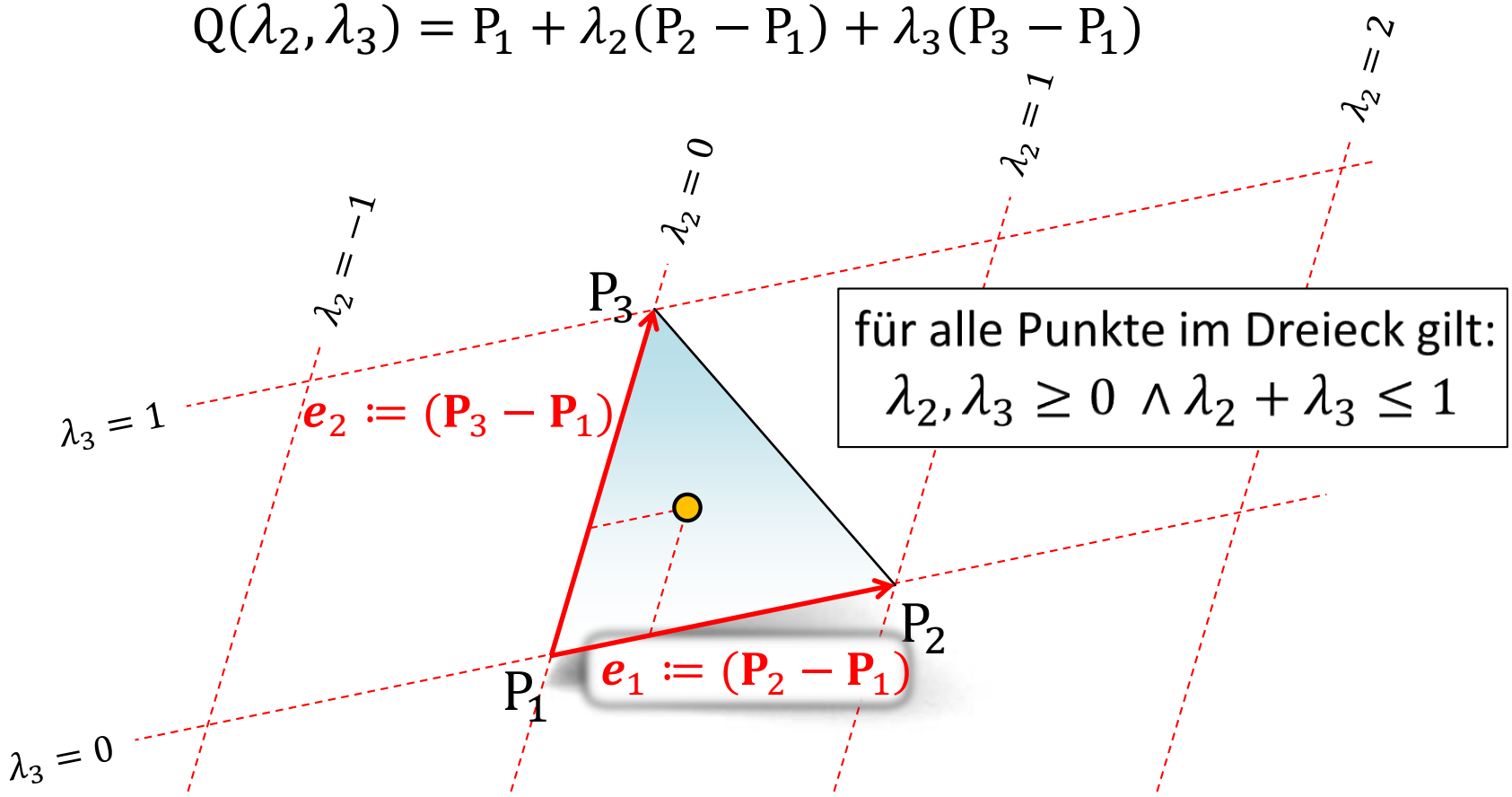
Interpretation:
Dreieckskanten als Achsen eines schiefwinkligen Koordinatensystems

Schnittpunktberechnung mit Dreieck

Strahl-Dreieck-Schnitt

- jeder Punkt in der Ebene des Dreiecks lässt sich beschreiben durch

$$Q(\lambda_2, \lambda_3) = P_1 + \lambda_2(P_2 - P_1) + \lambda_3(P_3 - P_1)$$



Interpretation:
Dreieckskanten als Achsen eines schiefwinkligen Koordinatensystems

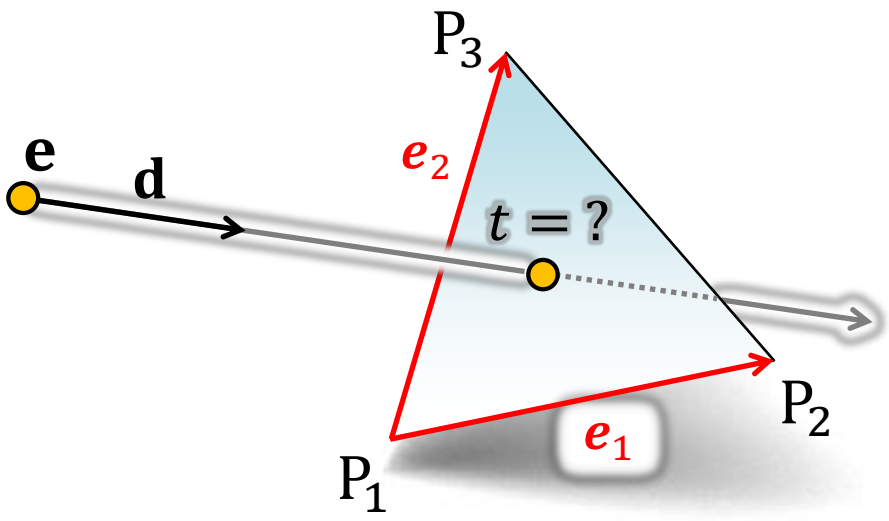
Schnittpunktberechnung mit Dreieck

Strahl-Dreieck-Schnitt

- Gleichsetzen $\mathbf{r}(t) = \mathrm{Q}(\lambda_2, \lambda_3)$

  $\mathbf{e} + t\mathbf{d} = \mathrm{P}_1 + \lambda_2 \boldsymbol{e}_1 + \lambda_3 \boldsymbol{e}_2$
- Lösung existiert ⇔ Schnittpunkt mit der Ebene
  - 3 Gleichungen, 3 Unbekannte t, λ_2, λ_3:

    $e_x + t d_x = \mathrm{P}_{1,x} + \lambda_2 \boldsymbol{e}_{1,x} + \lambda_3 \boldsymbol{e}_{2,x}$, …



- Schnittpunkt mit dem Dreieck, wenn $\lambda_2, \lambda_3 \geq 0 \ \wedge \lambda_2 + \lambda_3 \leq 1$, in Richtung des Strahls, wenn $t > 0$
- Lösung effizient mittels Cramerscher Regel (Determinanten)

Schnittpunktberechnung mit Dreieck

Lösung mit Determinanten (Cramersche Regel)

- Gleichsetzen $\mathbf{r}(t) = \mathrm{Q}(\lambda_2, \lambda_3)$

$$\mathbf{e} + t\mathbf{d} = \mathrm{P}_1 + \lambda_2 \boldsymbol{e}_1 + \lambda_3 \boldsymbol{e}_2$$

- Gleichungssystem in Matrixform:

$$\begin{bmatrix} -d_x & e_{1,x} & e_{2,x} \\ -d_y & e_{1,y} & e_{2,y} \\ -d_z & e_{1,z} & e_{2,z} \end{bmatrix} \begin{bmatrix} t \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = \begin{bmatrix} \mathrm{e}_x - \mathrm{P}_{1,x} \\ \mathrm{e}_y - \mathrm{P}_{1,y} \\ \mathrm{e}_z - \mathrm{P}_{1,z} \end{bmatrix}$$

$$[-\mathbf{d} \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2] \begin{bmatrix} t \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = [\mathbf{e} - \mathbf{P}_1]$$

Schnittpunktberechnung mit Dreieck

Lösung mit Determinanten (Cramersche Regel)

- Gleichungssystem in Matrixform:

$$[-\mathbf{d} \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2] \begin{bmatrix} t \\ \lambda_2 \\ \lambda_3 \end{bmatrix} = [\mathbf{e} - \mathrm{P}_1]$$

- Lösung:

$$t = \frac{|(\mathbf{e} - \mathrm{P}_1) \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2|}{|-\mathbf{d} \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2|},$$

$$\lambda_2 = \frac{|-\mathbf{d} \quad (\mathbf{e} - \mathrm{P}_1) \quad \boldsymbol{e}_2|}{|-\mathbf{d} \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2|}, \lambda_3 = \ \ldots$$

- letzte Umformung („Spatprodukt“):

$$|-\mathbf{d} \quad \boldsymbol{e}_1 \quad \boldsymbol{e}_2| = (-\mathbf{d} \times \boldsymbol{e}_1) \cdot \boldsymbol{e}_2$$

$$(-\mathbf{d} \times \boldsymbol{e}_1) \cdot \boldsymbol{e}_2 = (\boldsymbol{e}_1 \times \boldsymbol{e}_2) \cdot -\mathbf{d} = (\boldsymbol{e}_2 \times -\mathbf{d}) \cdot \boldsymbol{e}_1$$

Schnittpunktberechnung mit Dreieck

(Pseudo-)Code des sog. Möller-Trumbore-Algorithmus

```
vec3  edge1 = P2 - P1;
vec3  edge2 = P3 - P1;
vec3  pvec  = cross( d, edge2 );
float idet  = 1.0 / dot( edge1, pvec );

vec3  tvec = S - P1;
float v    = dot( tvec, pvec ) * idet;

if ( v < 0 || v > 1 ) return 0;            // early out 1

vec3  qvec = cross( tvec, edge1 );
float u    = dot( d, qvec ) * idet;

if ( u < 0 || u + v > 1 ) return 0;        // early out 2

float t = dot( edge2, qvec ) * idet;

// <== early out 3, only for SIMD if(any(..))
if ( t > 0 && t <= "kleinstes bisher gefundenes t" )
{
  hit = { Dreieck-ID, t, u, v };
  return 1;
}
return 0;
```

Weitere Schnitttests

	ray	plane	sphere	cylinder	cone	triangle	AABB	OBB	frustum	polyhedron	
ray	Gems p. 304; SG; TGS; RTCD p.198; SoftSurfer; RTR2 p.618; RTR3 p. 781	IRT p.90,88; SG; GTCG p.482; TGS; RTCD p.175; SoftSurfer (more)	IRT p. 39, 91; Gems p. 388; Held jgt 2(4); GTweb; 3DG p. 16; GTCG p. 501; TGS; RTCD p. 127, 177; RTR2 p. 568; RTR3 p. 738	IRT p. 91; Gems IV p. 356; Held jgt 2(4); GTweb; GTCG p. 507; TGS; RTCD p. 194	IRT p. 91; Gems V p. 227; Held jgt 2(4); GTweb; GTCG p. 512	Möller-Trumbore jgt 2(1); IRT p. 53, 102; Gems IV p. 24; Held jgt 2(4); GTweb; 3DG p. 17; Möller; GTCG p. 485; TGS; RTCD p. 153, 184; Lofstedt jgt 10(2); Chirkov jgt 10(3); Lagae jgt 10(4); SoftSurfer; RTR2 p. 578; RTR3 p. 746; Havel TVCG June 2009	IRT p. 65,104; Gems p. 395; Smits; 3DG p. 20; Terdiman (optimized Woo); Schroeder; GTCG p. 628; TGS; RTCD p. 179; Mahovsky jgt 9(1); Williams jgt 10(1); Eisemann jgt 12(4) (code); RTR2 p. 572; RTR3 p. 742	(IRT p. 104; Gems II p. 247); GTweb; Gomez; GTCG p. 630; TGS; RTCD p. 179; RTR2 p. 572; RTR3 p. 743	(IRT p. 104; Gems II p. 247)	IRT p. 104; Gems II p. 247; GTCG p. 493; Platis jgt 8(4); RTCD p. 198; SoftSurfer	**ray**
plane	IRT p.90,88; SG; GTCG p.482; TGS; RTCD p.175; SoftSurfer (more)	GTweb; SG; GTCG p.529; RTCD p. 207	distance of center to plane <= radius; GTweb; Gomez; GTCG p.548; TGS; RTCD p.160,219	GTweb; GTCG p.551; TGS;	GTCG p. 563; RTCD p.164	Check if all vertices are on one side; (Möller jgt 2(2)); GTCG p.534; SoftSurfer	Gems IV p.74; GTCG p. 634; TGS; RTCD p. 161, 222; RTR2 p. 587; RTR3 p. 755	GTweb; Gomez; GTCG p.635; TGS; RTCD p. 161; RTR2 p. 588; RTR3 p. 757			**plane**
sphere	IRT p.39,91; Gems p. 388; Held jgt 2(4); GTweb; 3DG p.16; GTCG p.501; TGS; RTCD p.127,177; RTR2 p.568; RTR3 p.738	distance of center to plane <= radius; GTweb; Gomez; GTCG p.548; TGS; RTCD p.160,219	If radii A+B >= center/center distance; Held jgt 2(4); GTweb; Gomez; GPG p. 390; GTCG p. 602; TGS; RTCD p.88,215,223; RTR3 p.763	If radii A+B >= center/axis distance; Held jgt 2(4); (GTCG p.602); TGS; RTCD p. 114	GTweb; (GTCG p. 602)	Held jgt 2(4); GTweb; Karabassi jgt 4(1); TGS; RTCD p.139,167,226	Gems p.335; Gomez; GTCG p. 644; TGS; RTCD p. 130, 165, 228; RTR2 p. 599; RTR3 p. 763	TGS; RTCD p. 132,166	GTweb; Assarsson; GPG p. 433; 3DG p. 302; RTR2 p.609; RTR3 p.774	3DG p. 462; RTCD p. 142	**sphere**
(capped) cylinder	IRT p. 91; Gems IV p. 356; Held jgt 2(4); GTweb; GTCG p.507; TGS; RTCD p. 194	GTweb; GTCG p.551; TGS;	If radii A+B >= center/axis distance; Held jgt 2(4); (GTCG p.602); TGS;	If radii A+B >= axis/axis distance; GTweb; GTCG p. 602,646; TGS; RTCD p. 114	(GTCG p. 602)	Held jgt 2(4); TGS	TGS	TGS	GPG p.380		**(capped) cylinder**
(capped) cone	IRT p. 91; Gems V p. 227; Held jgt 2(4); GTweb; GTCG p.512	GTCG p.563; RTCD p. 164	GTweb; (GTCG p. 602)	(GTCG p. 602)	(GTCG p. 602)	Held jgt 2(4); GTweb; GTCG p.583					**(capped) cone**
triangle (polygon)	Möller-Trumbore jgt 2(1); IRT p. 53, 102; Gems IV p. 24; Held jgt 2(4); GTweb; 3DG p. 17; Möller; GTCG p. 485; TGS; RTCD p. 153, 184; Lofstedt jgt 10(2); Chirkov jgt 10(3); Lagae jgt 10(4); SoftSurfer; RTR2 p. 578; RTR3 p. 746	Check if all vertices are on one side; (Möller jgt 2(2)); GTCG p.534; SoftSurfer	Held jgt 2(4); GTweb; Karabassi jgt 4(1); TGS; RTCD p. 135, 167, 226	Held jgt 2(4); TGS	Held jgt 2(4); GTweb; GTCG p.583	Möller jgt 2(2); Held jgt 2(4); GTweb; Möller; GPG p. 393; GTCG p.539; TGS; RTCD p. 155, 172; Shen jgt 8(1); Guigue jgt 8(1); SoftSurfer; RTR2 p. 590; RTR3 p. 757	Gems III p. 236; Gems V p. 375; TGS; RTCD p. 169; Möller; RTR2 p. 596; RTR3 p. 760	GTweb; TGS	homogeneous clipping; Gems p. 84	generalized clipping	**triangle (polygon)**
axis-aligned bounding box	IRT p. 65, 104; Gems p. 395; Smits; 3DG p. 20; Terdiman (optimized Woo); Schroeder; GTCG p. 626; TGS; RTCD p. 179; Mahovsky jgt 9(1); Williams jgt 10(1); Eisemann jgt 12(4); RTR2 p.572; RTR3 p. 742	Gems IV p. 74; GTCG p. 634; TGS; RTCD p. 161, 222; RTR2 p. 587; RTR3 p. 755	Gems p.335; Gomez; GTCG p. 644; TGS; RTCD p. 130, 165, 228; RTR2 p. 599; RTR3 p. 763	TGS		Gems III p. 236; Gems V p. 375; TGS; RTCD p. 169; Möller; RTR2 p. 596; RTR3 p. 760	A.LO <= B.HI && A.HI >= B.LO; Gomez; 3DG p. 445; GTCG p. 637; TGS; RTCD p. 79, 230; RTR2 p. 600; RTR3 p. 765	(Gems V p. 378; GTweb; Gomez; GTCG p. 639); TGS; RTCD p. 101; RTR2 p. 602; RTR3 p. 777	(Gems IV p. 83); (Gems V p. 378); (GTweb); Assarsson; 3DG p. 302;(GTCG p.624); RTR2 p. 612; RTR3 p. 771	Gems IV p. 74; Gems V p. 378	**axis-aligned bounding box**
oriented bounding box	(IRT p. 104; Gems II p. 247); GTweb; Gomez; GTCG p. 630; TGS; RTCD p. 179; RTR2 p. 572; RTR3 p. 743	GTweb; Gomez; GTCG p. 635; TGS; RTCD p. 161; RTR2 p. 588; RTR3 p. 757	TGS; RTCD p. 132, 166	TGS		GTweb; TGS	(Gems V p. 378; GTweb; Gomez; GTCG p. 639); TGS; RTCD p. 602; RTR3 p. 777	GTweb; Gomez; 3DG p. 449; GTCG p. 639; TGS; RTCD p. 101; RTR2 p. 602; RTR3 p. 767	GTweb; Assarsson; GTCG p. 624; RTR2 p. 612; RTR3 p. 777	(Gems IV p. 83)	**oriented bounding box**
viewing frustum	(IRT p. 104; Gems II p. 247)		GTweb; Assarsson; GPG p. 433; 3DG p. 302; RTR2 p. 609; RTR3 p. 774	GPG p. 380		homogeneous clipping; Gems p. 84	(Gems IV p. 83); (Gems V p. 378); (GTweb); Assarsson; 3DG p. 302;(GTCG p.624); RTR2 p. 612; RTR3 p. 771	GTweb; Assarsson; GTCG p. 624; RTR2 p. 612; RTR3 p. 777	(Gems IV p. 83)	(Gems IV p. 83)	**viewing frustum**
convex polyhedron	IRT p. 104; Gems II p. 247; GTCG p. 493; Platis jgt 8(4); RTCD p. 198; SoftSurfer		3DG p. 462; RTCD p. 142			generalized clipping	Gems IV p. 74; Gems V p. 378	(Gems IV p. 83)	(Gems IV p. 83)	Gems IV p. 83; 3DG p. 453; (GTweb; RTR2 p. 963; RTR3 p. 731); GTCG p. 726; Ganovelli jgt 7(2); RTCD p. 383, 399, 410	**convex polyhedron**

Raytracing Pseudocode

```
for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t' = intersect( object, e, d );
      if ( t' > 0 && t' < t ) {
        intersection = object;
        t = t';
      }
    }
    ...
} }
```

Raytracing ist „embarrassingly parallel“:

```
#pragma omp parallel for
```

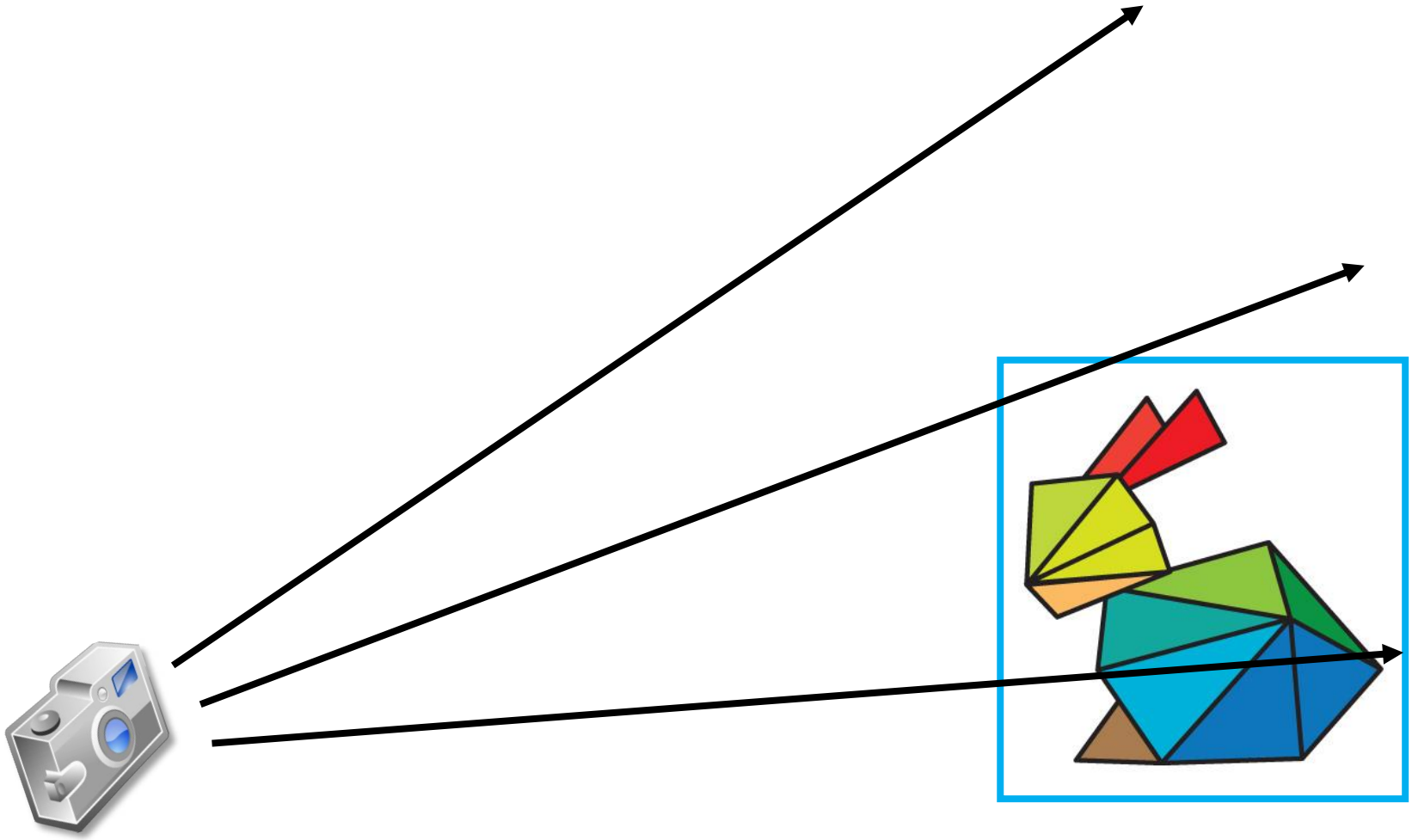
typische Parallelisierung: Multithreading über der Bildebene und SIMD/Vektorisierung innerhalb

- Achtung: uns interessieren nur Schnittpunkte vor der Kamera (`t'>0`), und von diesen der nächste in Strahlrichtung (`t'<t`)

Vorschau: Beschleunigung des Raycasting

Reduziere die Anzahl der Strahl-Primitiv-Schnitttests

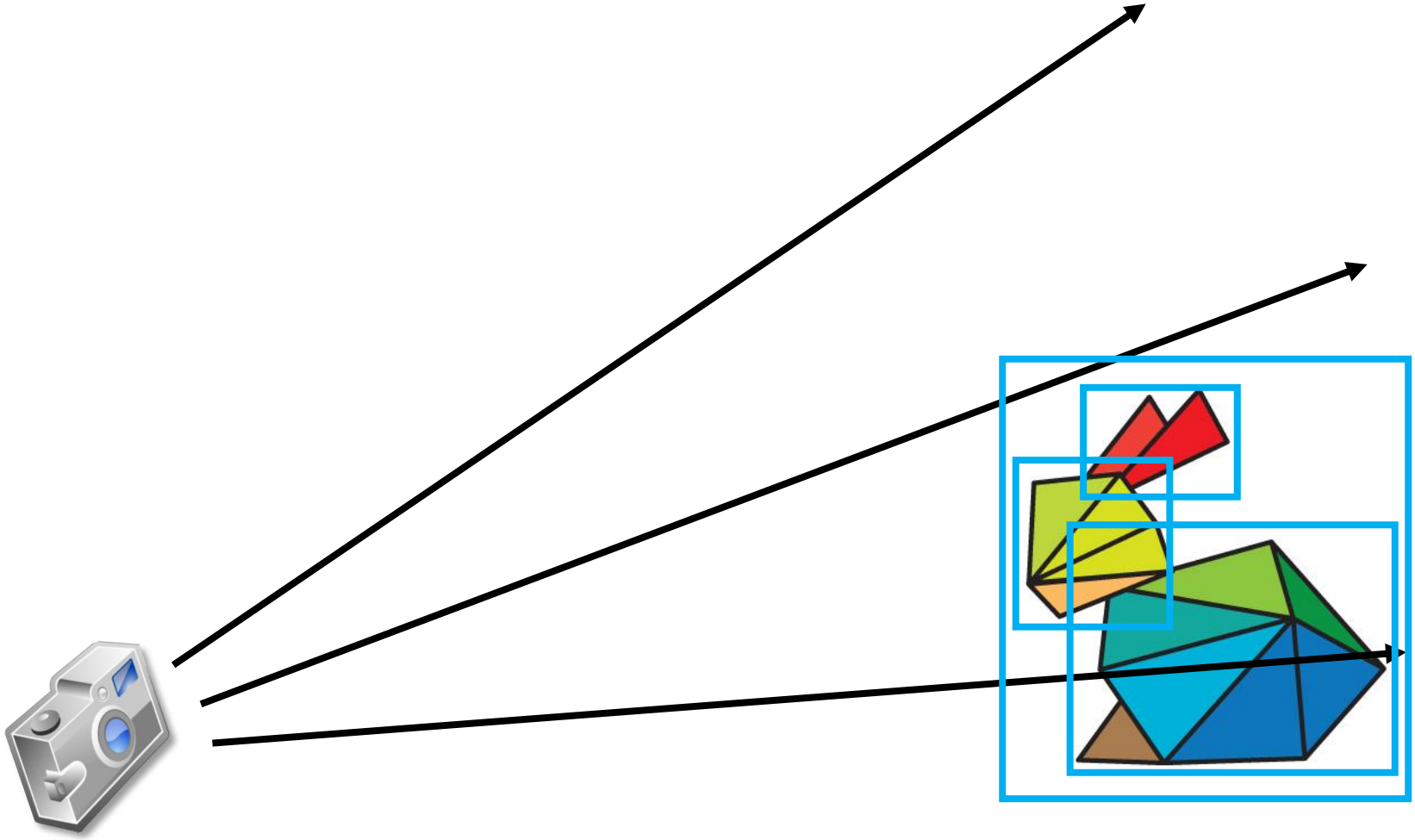
- schneller Ausschluss von Schnitttests mit Primitiven (Dreiecken)



Vorschau: Beschleunigung des Raycasting

Reduziere die Anzahl der Strahl-Primitiv-Schnitttests

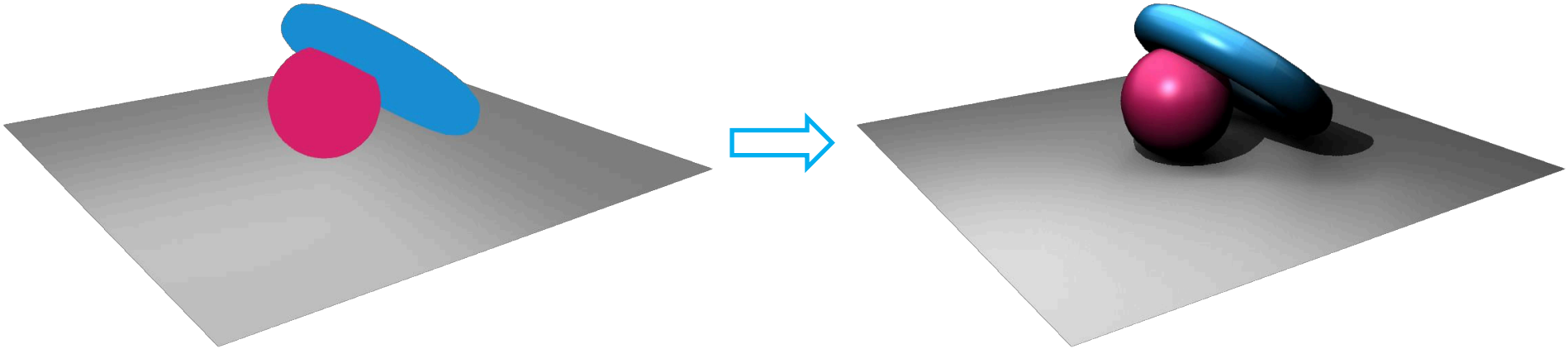
- schneller Ausschluss von Schnitttests mit Primitiven (Dreiecken)



Raytracing

Schritte

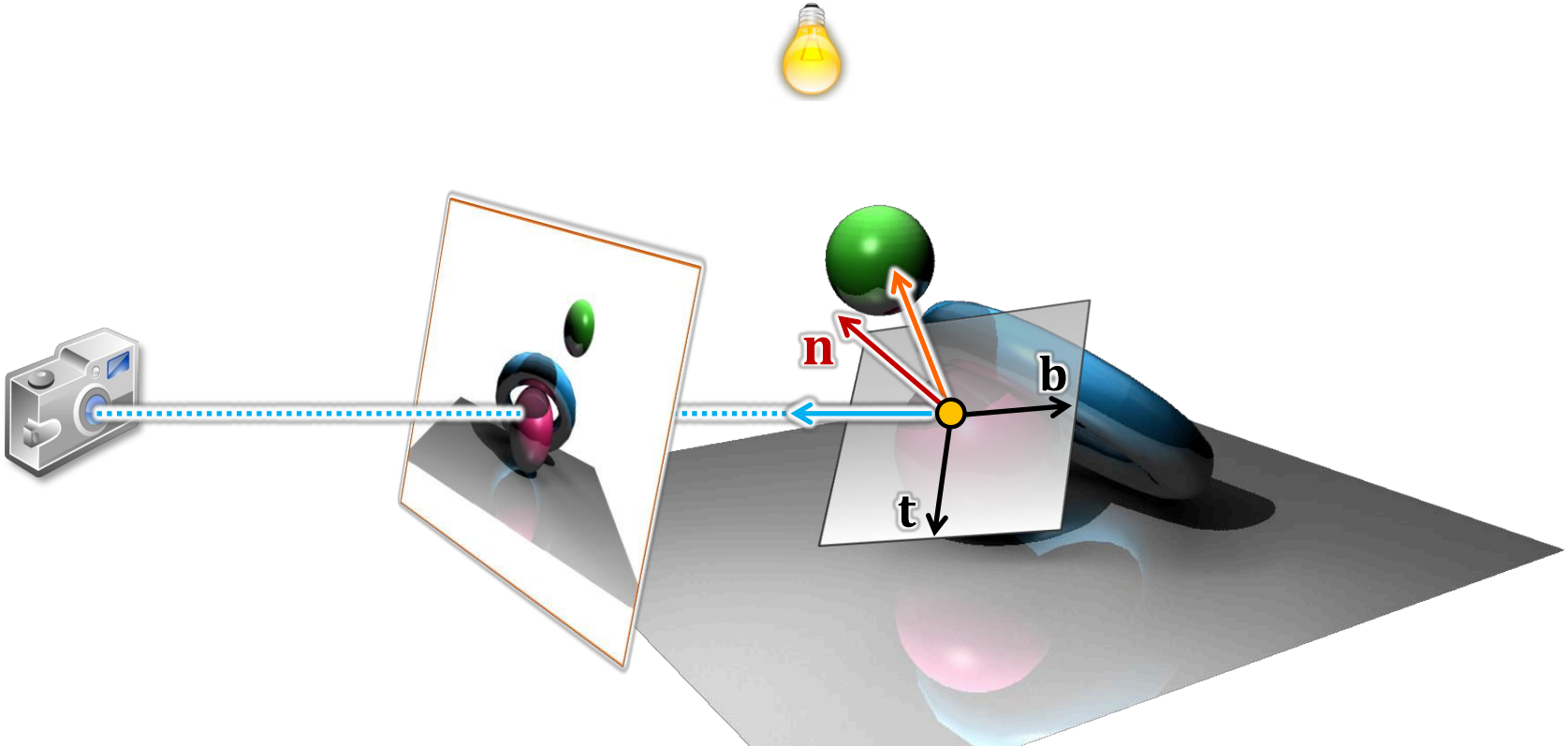
- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting):
  finde Dreieck, das den Sichtstrahl am nächsten zur Kamera schneidet
- Schattierung und Beleuchtungsberechnung (shading)
- Sekundärstrahlen für Spiegelung und Transmission



Raytracing

Schritte

- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting):
  finde Dreieck, das den Sichtstrahl am nächsten zur Kamera schneidet
- Schattierung und Beleuchtungsberechnung (shading)
- Sekundärstrahlen für Spiegelung und Transmission



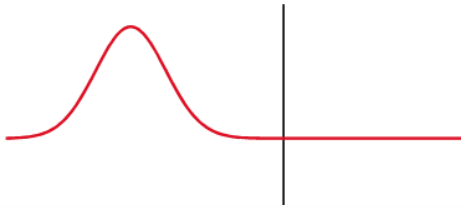
Materialeigenschaften

- woher kommt das Aussehen von Materialien?
- Schattierung ist essentiell für dreidimensionalen Eindruck und ist ein Ergebnis von
  - Emission der Lichtquellen (Intensität, Farbe, Position, …) und
  - Oberflächeneigenschaften (Material, Rauheit, Orientierung zur LQ, …)

[14]

Licht-Material-Interaktion

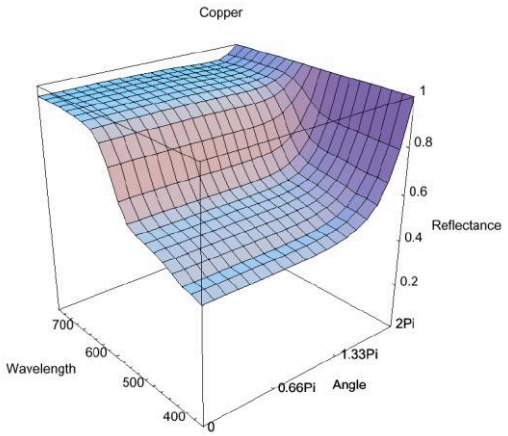
- was passiert, wenn Licht auf eine glatte Oberfläche trifft?
  - was bedeutet „glatt“?
  - ein Teil wird reflektiert – ein Teil dringt ins Material ein
  - das Verhältnis ist abhängig vom Einfallswinkel
    und von den Brechungsindizes beider Materialien
    (sog. Fresnel-Effekt)



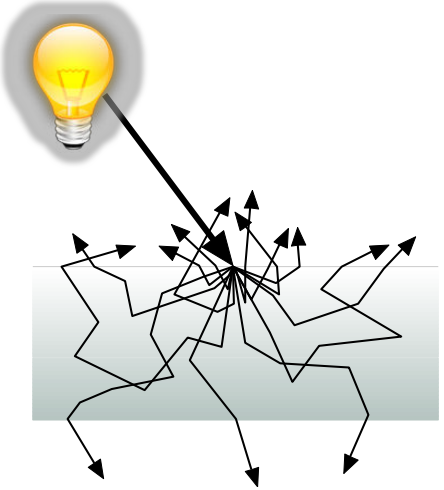
Bilder: Lafortune

Licht-Material Interaktion

- eindringendes Licht wird entweder (sehr schnell) absorbiert: Metalle



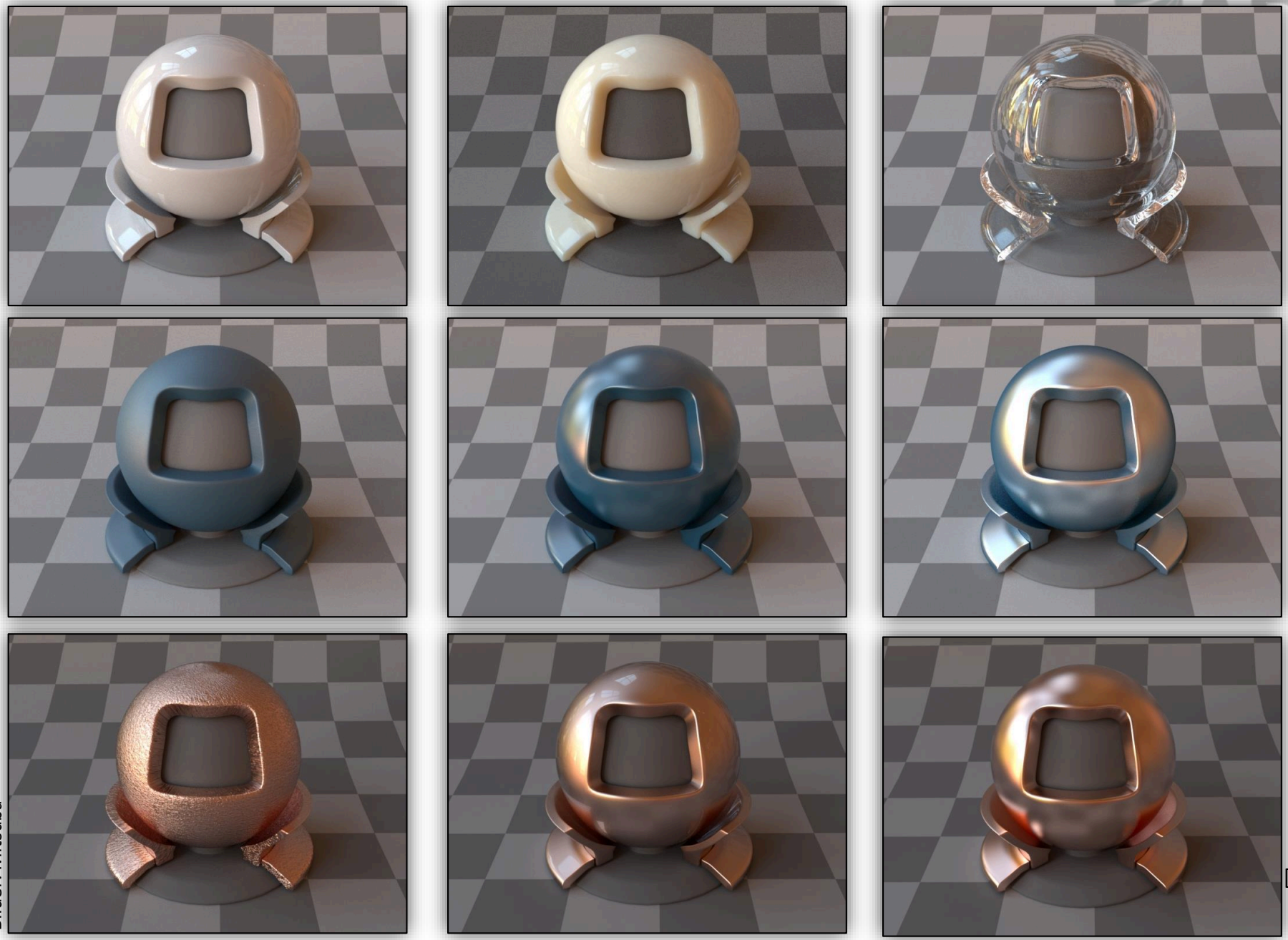
- …oder verlässt das Material an anderer Stelle wieder



[15]

- wir befassen uns im Folgenden nur mit der Reflexion an der Oberfläche!

Mehr als nur Oberflächenreflexion

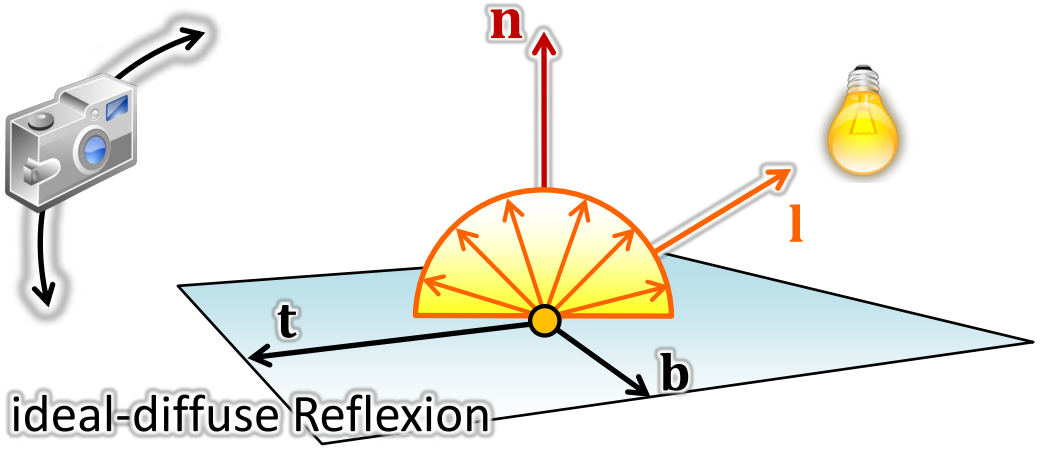


Bilder: Mitsuba

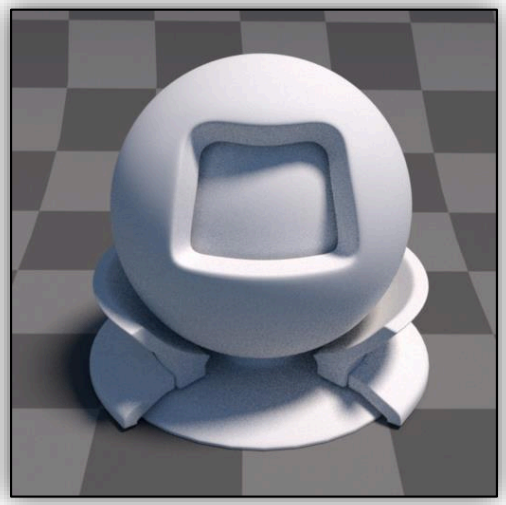
[16]

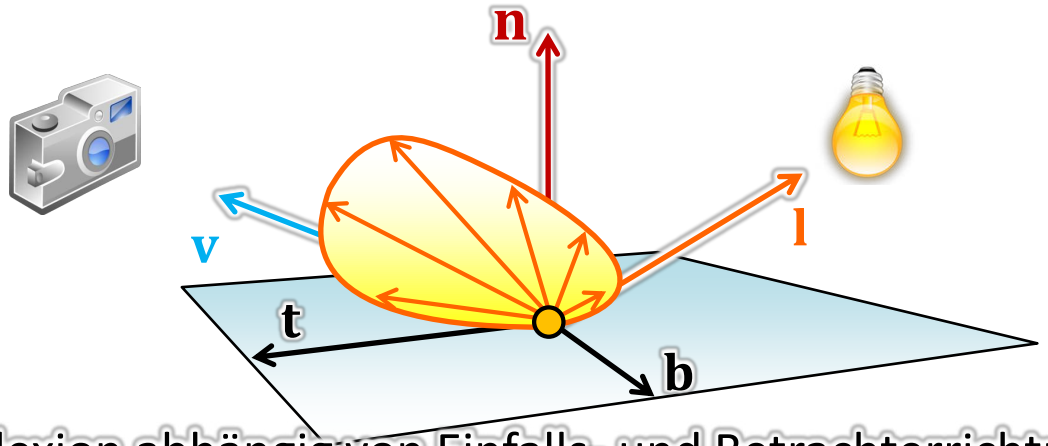
Einführung: Lokales Beleuchtungsmodell

Beispiele

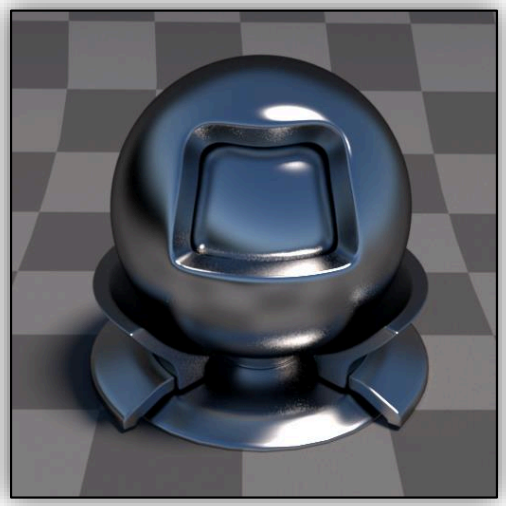


ideal-diffuse Reflexion





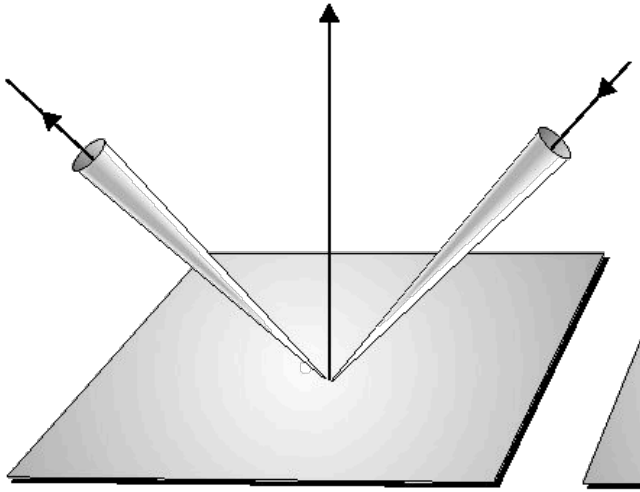
Reflexion abhängig von Einfalls- und Betrachterrichtung



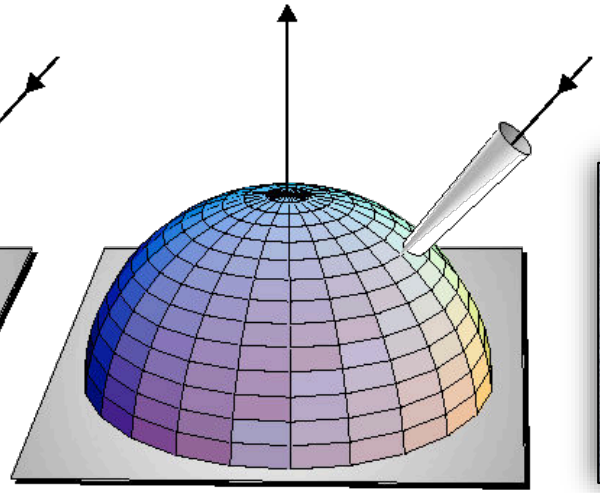
[16]

Reflexion

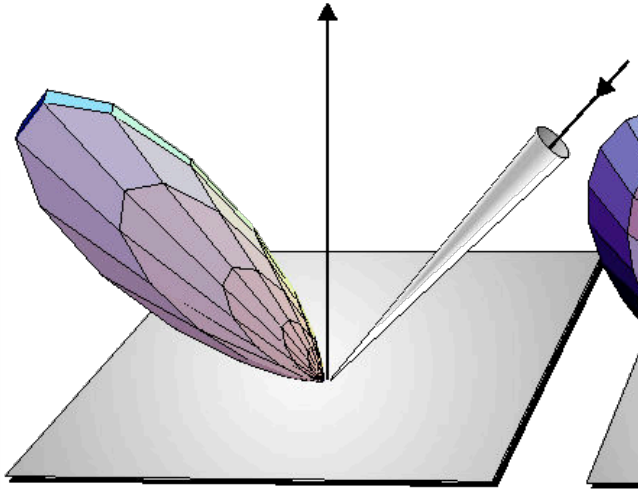
Begriffe und Darstellung



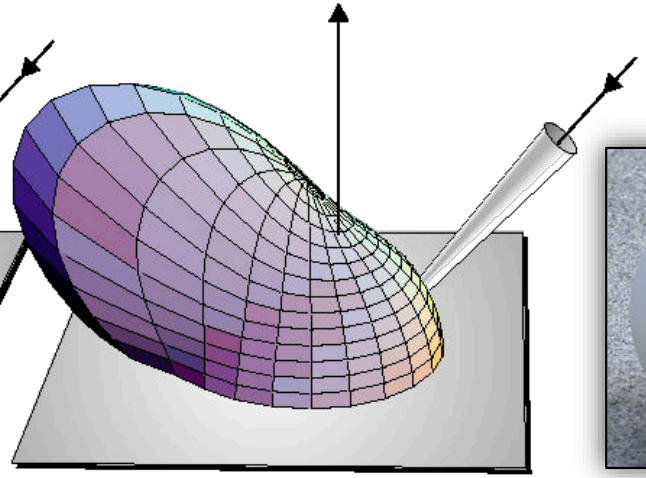
perfekt spiegelnd (engl. specular)



ideal-diffuse
(auch Lambertsche Reflexion)



glänzend, imperfekt spiegelnd
(engl. glossy, seltener rough specular)



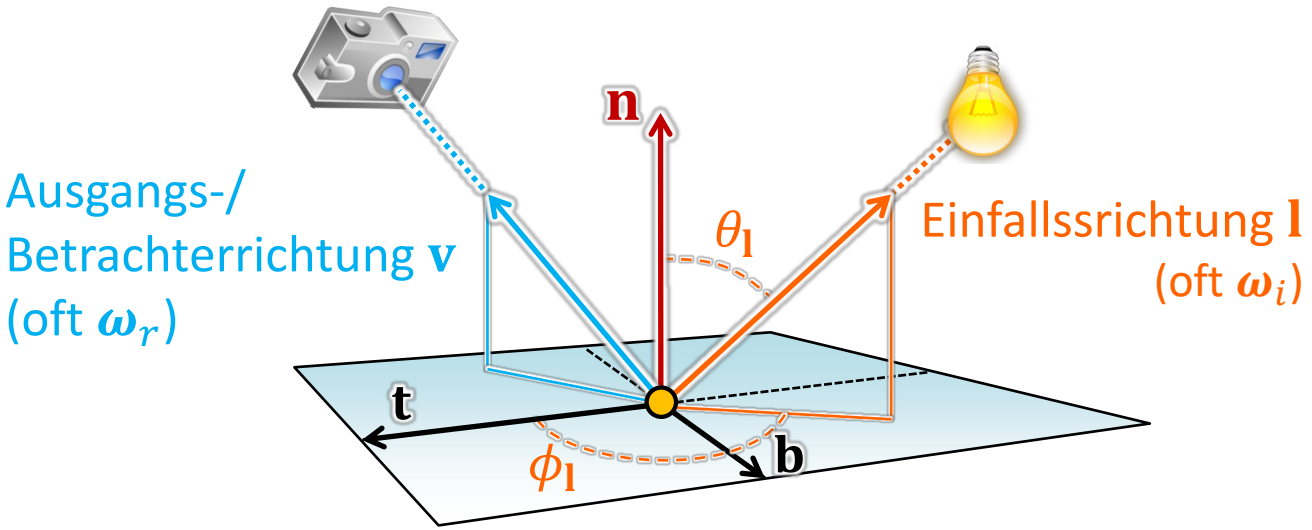
directional diffuse
(selten verwendeter Begriff)

Bilder: [17,58]

Einführung: Lokales Beleuchtungsmodell

Bidirektionale Reflektanzverteilungsfunktion

- … beschreibt Reflexionsverhalten von Oberflächen (engl. Bidirectional Reflectance Distribution Function, BRDF)
- um BRDFs unabhängig von einem bestimmten Oberflächenpunkt (und seiner Orientierung) anzugeben, verwendet man ein Referenzkoordinatensystem



Einführung: Lokales Beleuchtungsmodell

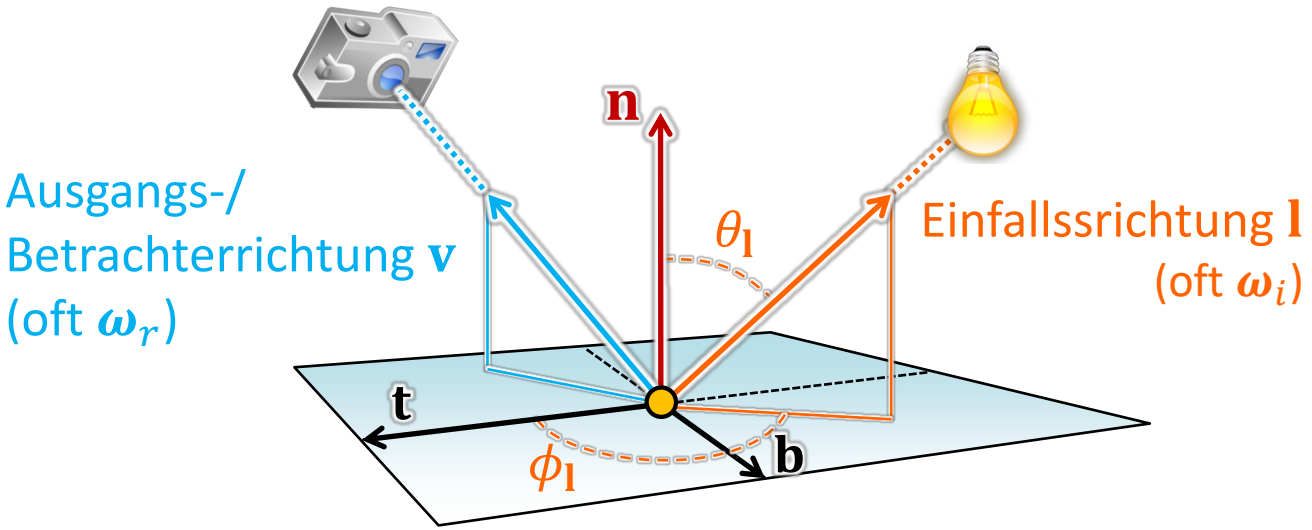
Bidirektionale Reflektanzverteilungsfunktion

- … beschreibt Reflexionsverhalten von Oberflächen
  (engl. Bidirectional Reflectance Distribution Function, BRDF)
- (Referenz-)Koordinatensystem mit den aufspannenden Vektoren Tangente $\mathbf{t}$, Bitangente $\mathbf{b}$ und Normale $\mathbf{n}$ am Oberflächenpunkt $\mathbf{x}$

$$f_r(\mathbf{l}, \mathbf{x}, \mathbf{v}) = f_r(\theta_\mathbf{l}, \phi_\mathbf{l}, \mathbf{x}, \theta_\mathbf{v}, \phi_\mathbf{v}) \quad [\mathrm{sr}^{-1}]$$

später noch etwas zur Radiometrie

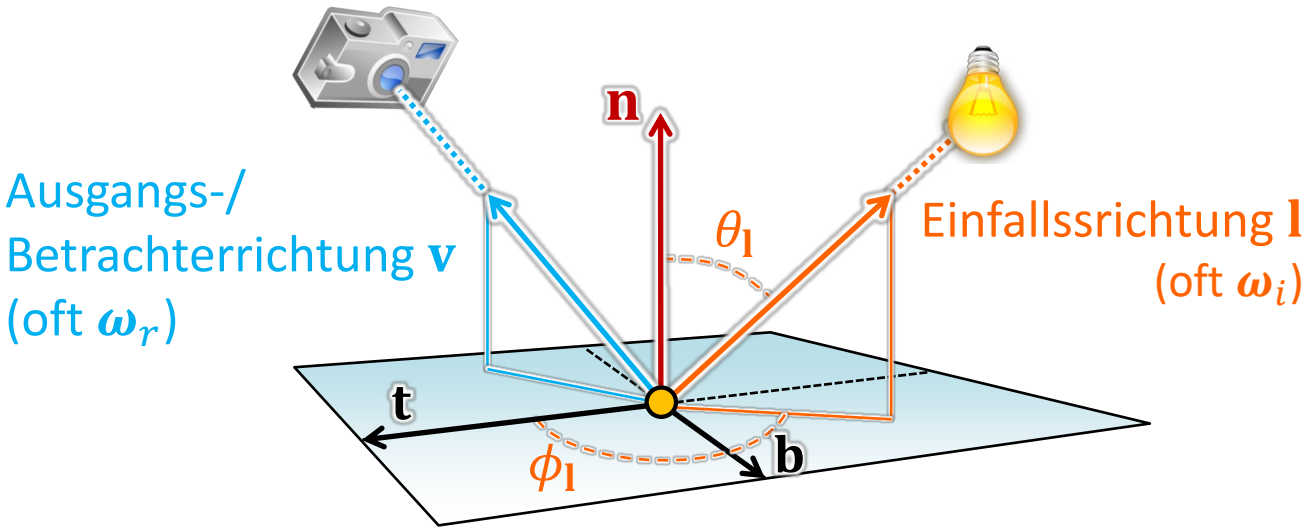
mit Polarwinkel $0 \leq \theta_\mathbf{l}, \theta_\mathbf{v} \leq {}^{\pi}/_{2}$ und Azimutalwinkel $0 \leq \phi_\mathbf{l}, \phi_\mathbf{v} < 2\pi$



Einführung: Lokales Beleuchtungsmodell

Bidirektionale Reflektanzverteilungsfunktion

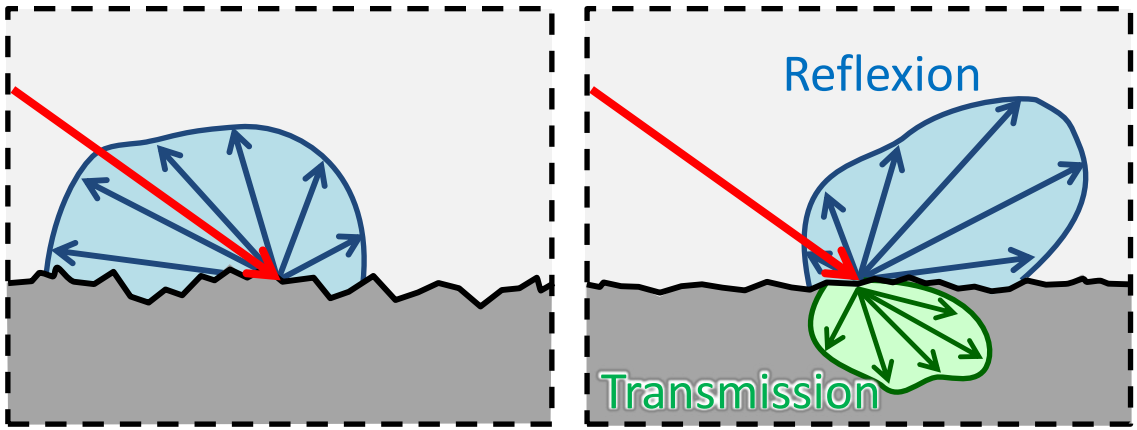
- … beschreibt Reflexionsverhalten von Oberflächen
  (engl. Bidirectional Reflectance Distribution Function, BRDF)
- BRDF basiert auf radiometrischen Größen und ist Verhältnis von…
  - Radiance (Strahldichte) $[\mathrm{W/m^2 sr}]$ die Oberfläche in Richtung **v** verlässt, zu
  - einfallender Irradiance (Flussdichte $[\mathrm{W/m^2}]$) aus Richtung **l**
  - merken Sie sich vorläufig einfach nur:
    „Verhältnis von ausgehendem zu einfallendem Licht“

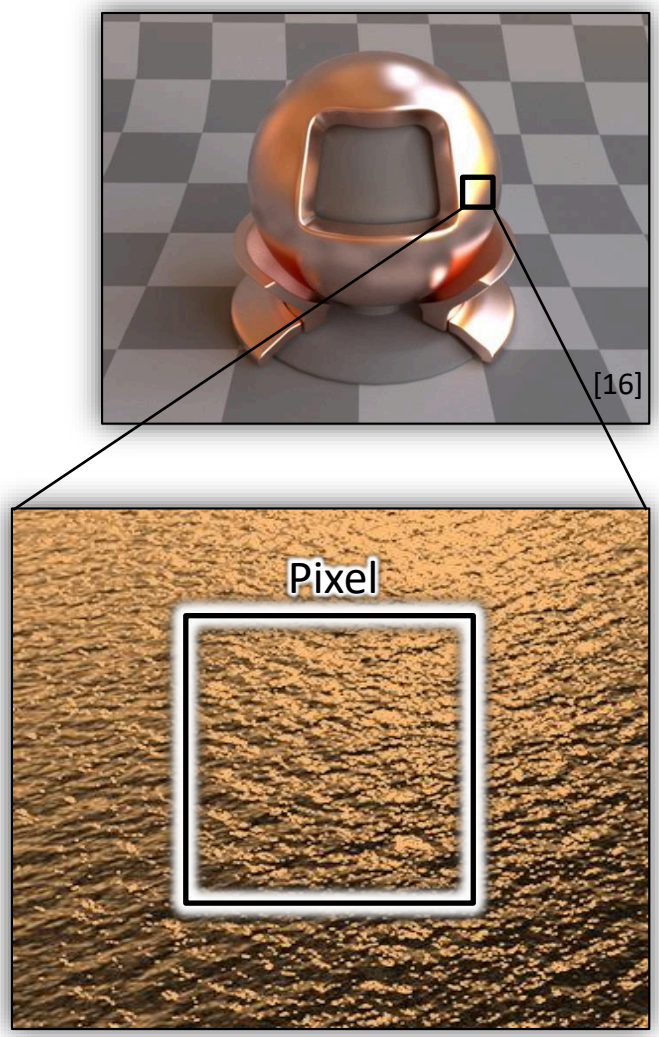


Einführung: Lokales Beleuchtungsmodell

BRDFs und Skalen

- mit einer BRDF beschreiben wir immer den Einfluss von Oberflächenstrukturen auf die Reflexion, die so klein sind, dass wir sie nicht geometrisch/anderweitig modellieren
- analog für Transmission: BTDF = Bidirectional Transmission Distribution Function
- Bidirectional Scattering Distribution Function bezeichnet BRDF und BTDF zusammen





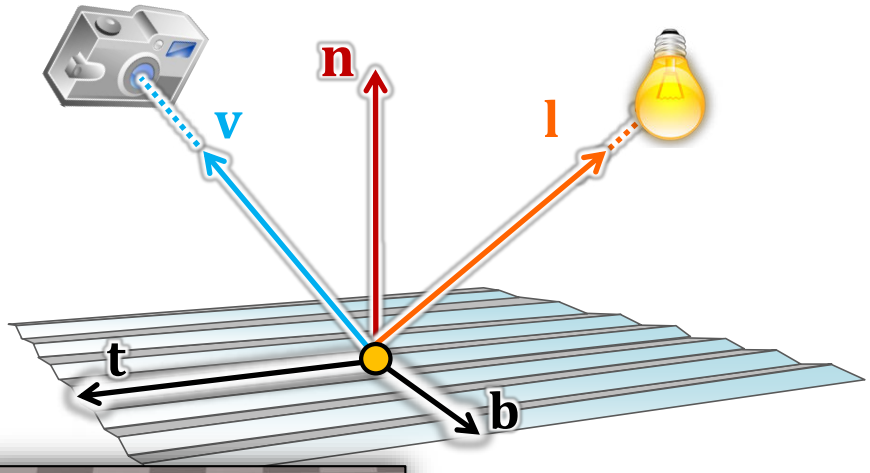
Einführung: Lokales Beleuchtungsmodell

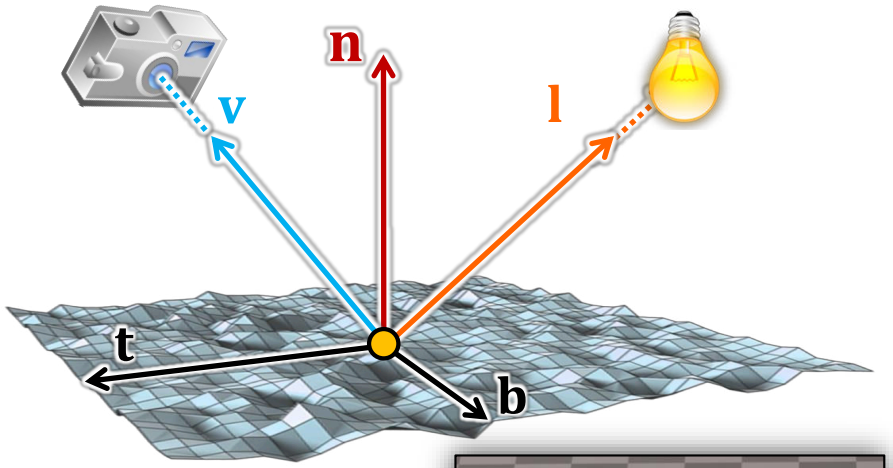
Bidirektionale Reflektanzverteilungsfunktion

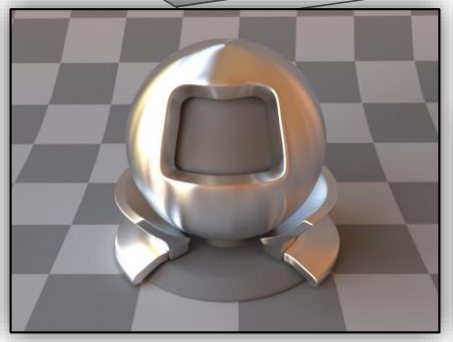
- **anisotrope BRDF (4D ohne Ort, links)**: Reflexion abhängig von der Rotation um die Normale aufgrund orientierter Mikrostrukturen
- **isotrope BRDF (3D ohne Ort, rechts):** Rotationsinvarianz um die Normale

$$f_r(\theta_\mathbf{l}, \phi_\mathbf{l}, \mathbf{x}, \theta_\mathbf{v}, \phi_\mathbf{v}) = f_r(\theta_\mathbf{l}, \phi_\mathbf{l} + \Delta\phi, \mathbf{x}, \theta_\mathbf{v}, \phi_\mathbf{v} + \Delta\phi)$$

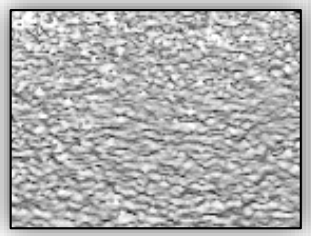
  - Achtung: die Differenz $\phi_\mathbf{l} - \phi_\mathbf{v}$ ist deswegen nicht irrelevant!

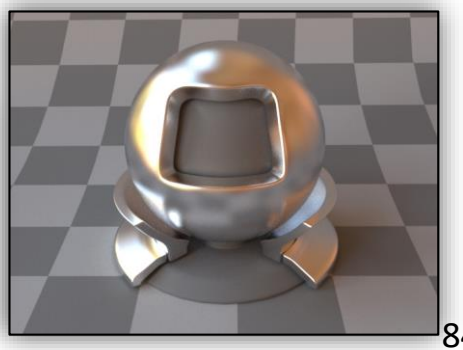






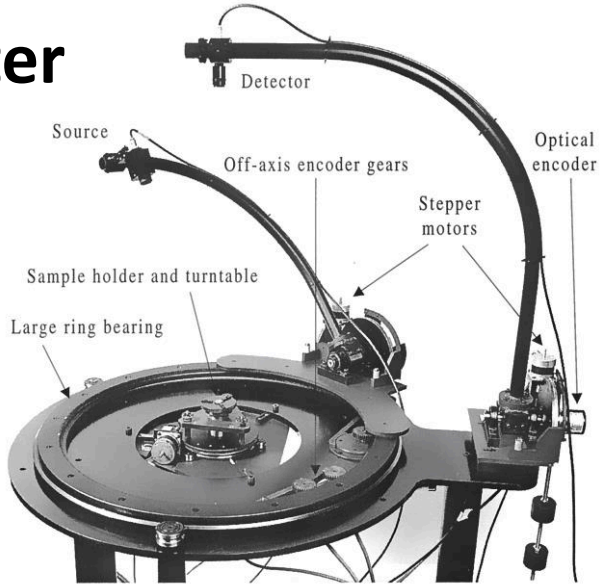
mikroskop. Aufnahmen





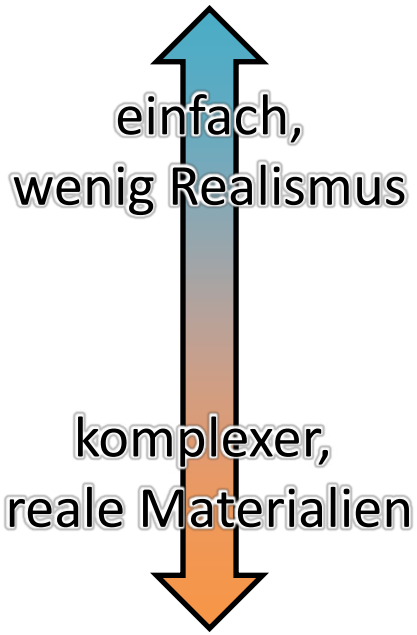
Messung von BRDFs – Gonioreflektometer

- Messung realer Materialproben ist aufwändig
- Modelle dagegen sind kompakt, rauschfrei und können an Messdaten angepasst werden



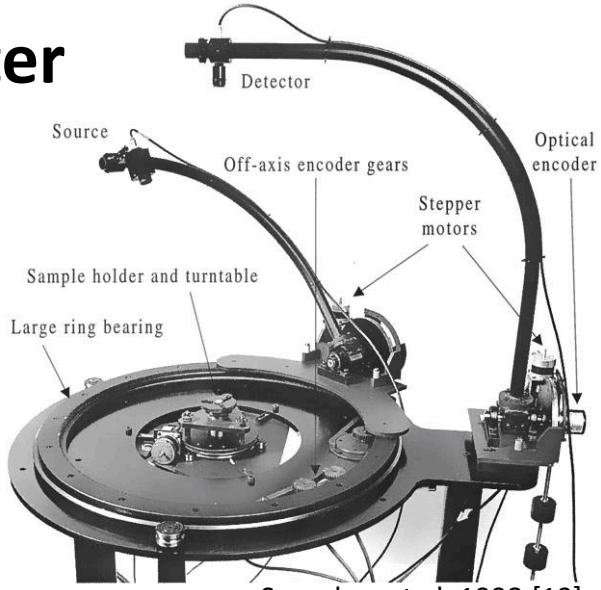
Saunders et al. 1998 [19]

- phänomenologische/empirische Modelle
  - wenige, intuitiv verständliche Parameter
  - keine strengen physikalischen Gesetzmäßigkeiten
  - oft ungeeignet Messdaten zu repräsentieren
- physikalisch-basierte Modelle
  - geeignet reale Materialien nachzubilden
  - Parameterwahl erfordert Verständnis des Modells
  - z.B. Mikrofacetten Modelle (gleich mehr)



Messung von BRDFs – Gonioreflektometer

- Messung realer Materialproben ist aufwändig
- Modelle dagegen sind kompakt, rauschfrei und können an Messdaten angepasst werden



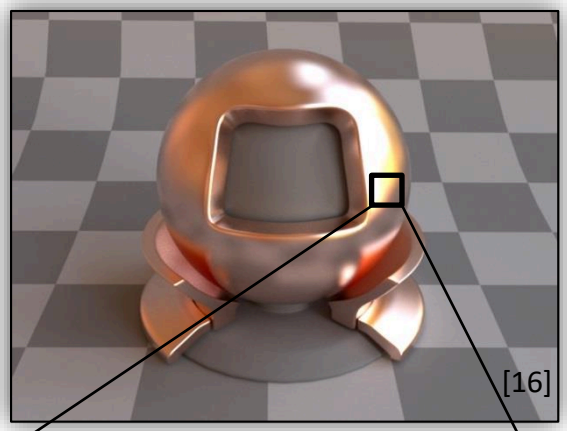
Saunders et al. 1998 [19]

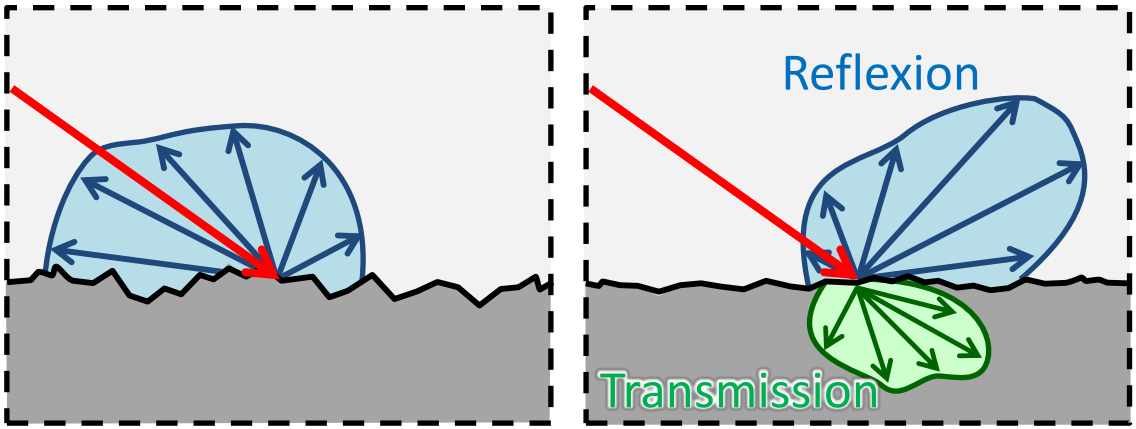
- phänomenologische/empirische Modelle
  Phong [75], Blinn-Phong [77],
  Ward [92], Lafortune et al. [97],
  Ashikhmin et al. [00], …
- physikalisch-basierte Modelle
  Mikrofacetten Modelle (Cook-Torrance [81], Walter et al. [07])
  Multiple-Scattering Microfacet BSDFs…, Heitz et al. 2016
  A Practical Extension to … Iridescence, Belcour et al. 2017
  und viele mehr!

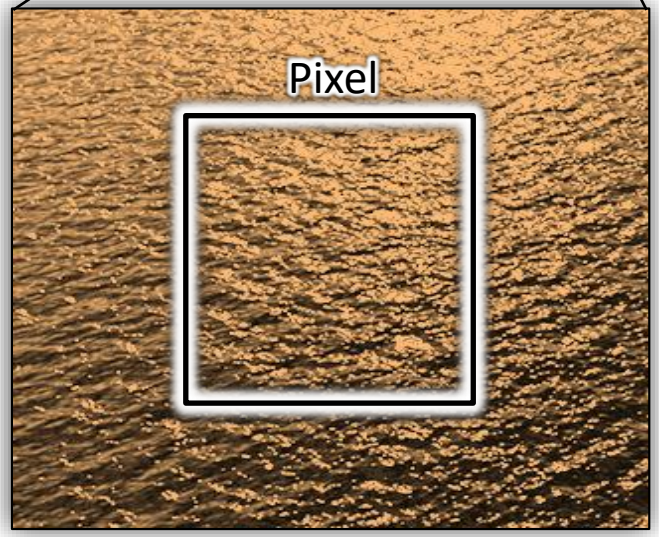
Exkurs: Modellierung realistischer Materialien

Mikrofacetten-Streumodelle für Oberflächen

- Vorstellung: Oberfläche zusammengesetzt aus vielen kleinen planaren, spiegelnden oder reflektierenden Flächen



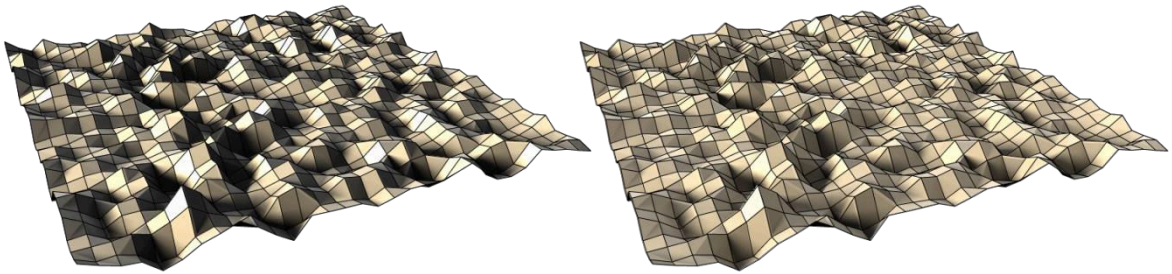


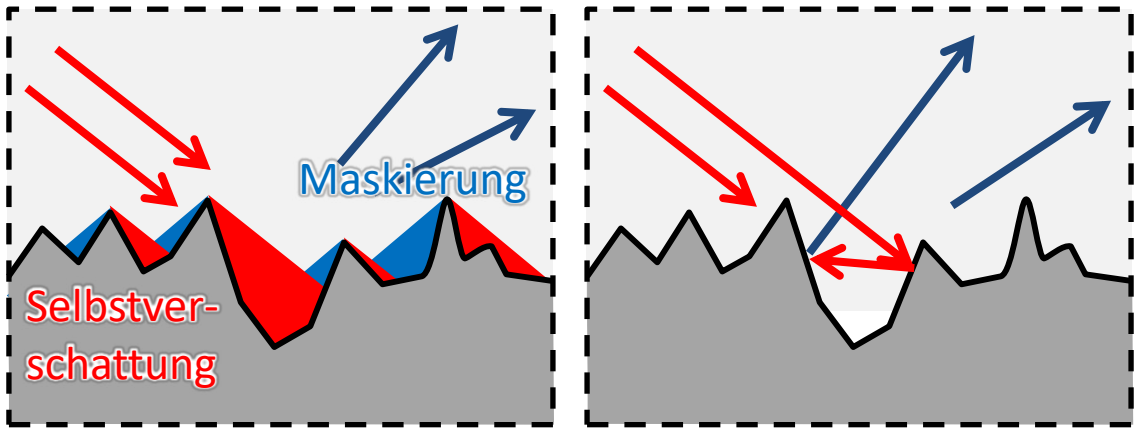


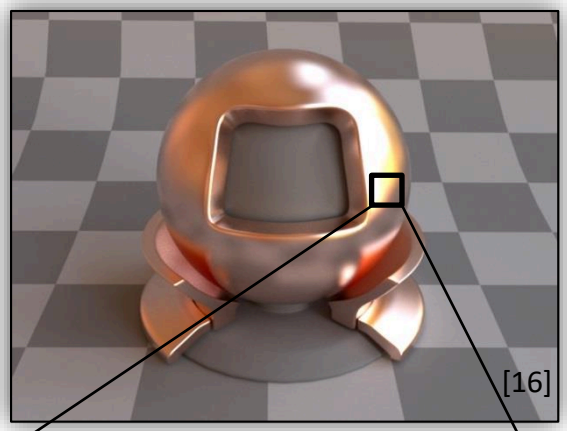
Exkurs: Modellierung realistischer Materialien

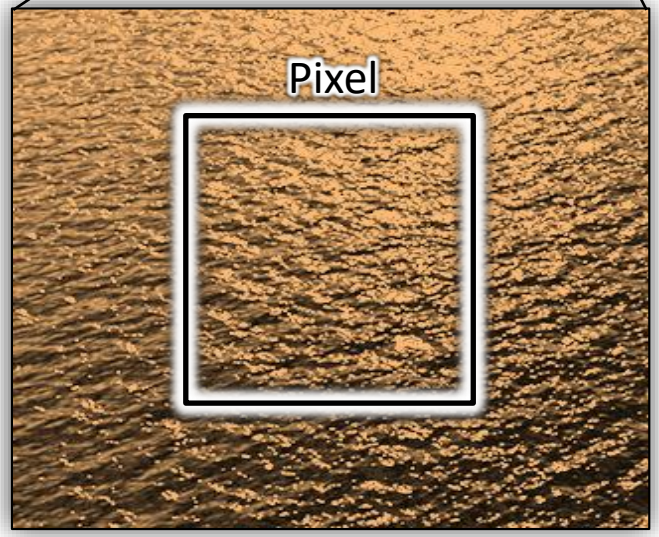
Mikrofacetten-Streumodelle für Oberflächen

- Reflexionseigenschaften maßgeblich durch Rauheit bestimmt: speichere Verteilung der Orientierungen der Mikrofacetten





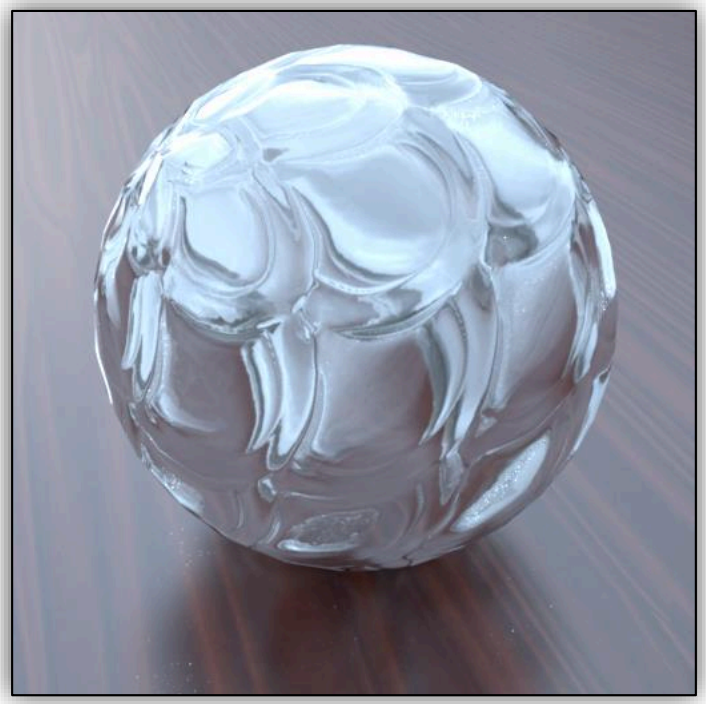


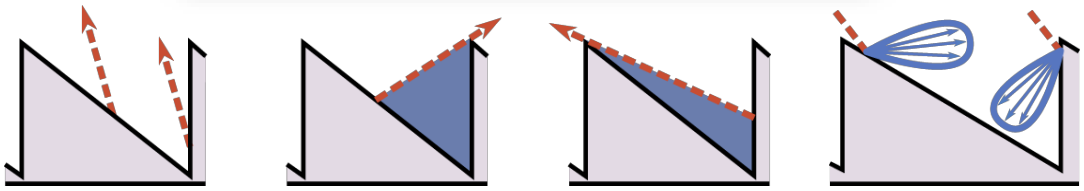


Exkurs: Modellierung realistischer Materialien

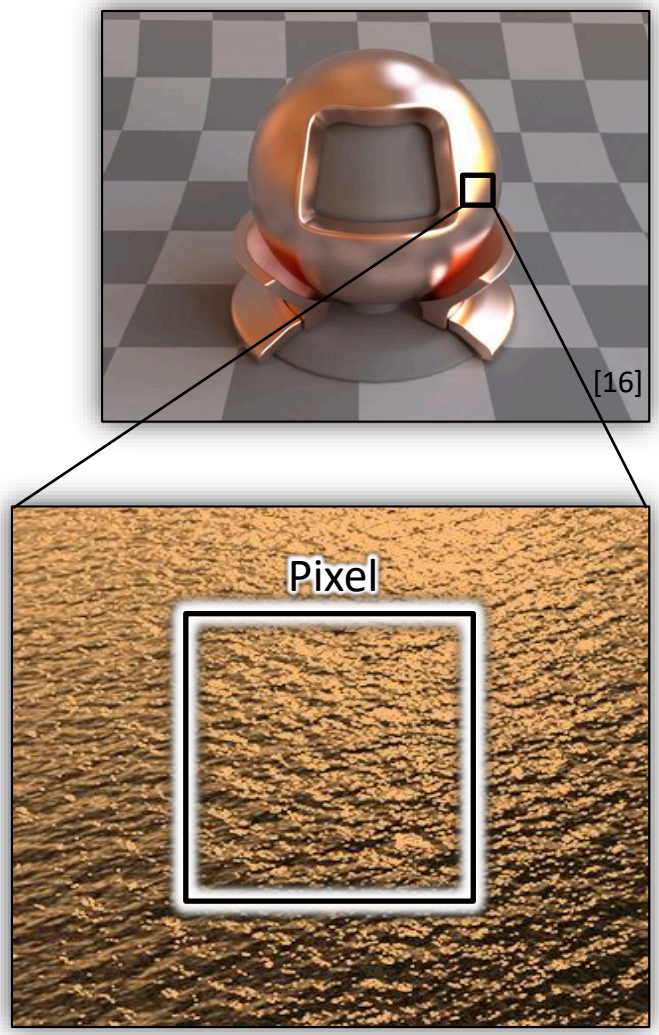
Mikrofacetten-Streumodelle für Oberflächen

Mesoskopisches Detail



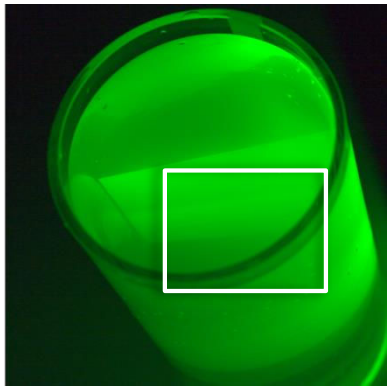


Schüssler et al. 17 [20]

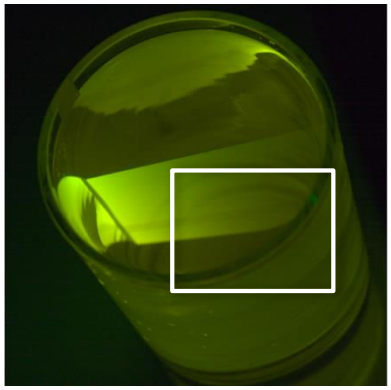


Exkurs: Motivation phänomenologische Modelle

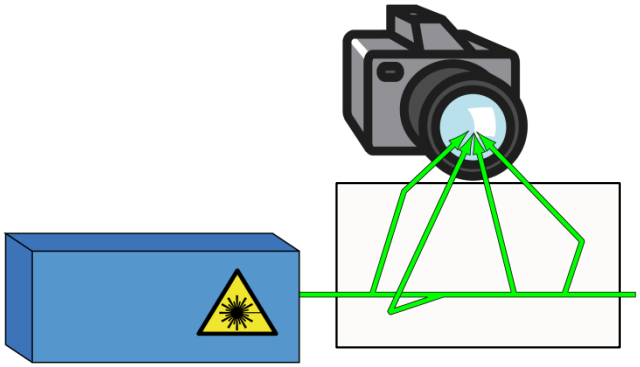
- wie kann man Lichtstrahlen sichtbar machen?

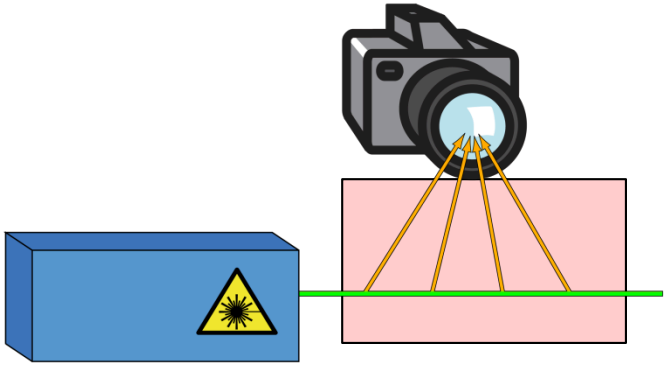


verdünnte Milch:
mehrfache Streuung



fluoreszierende Lösung:
einfache Streuung

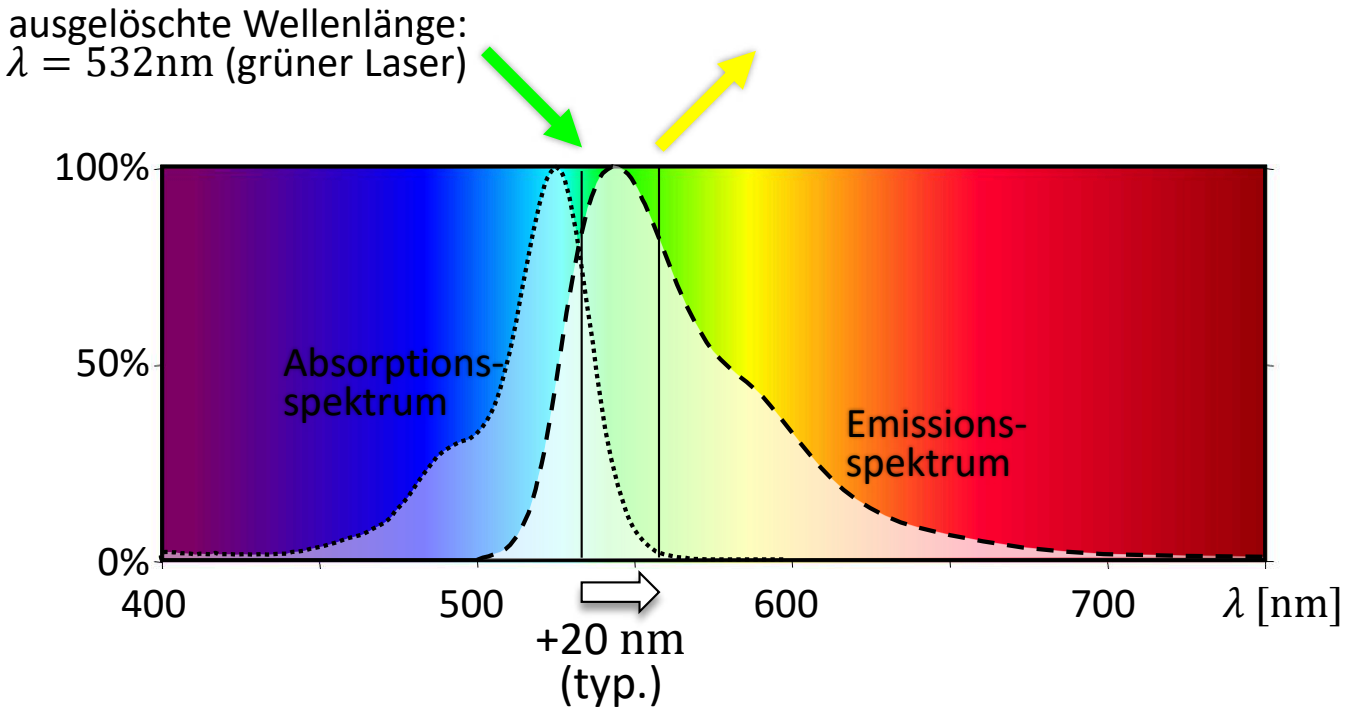




Hullin et al. 08 [21]

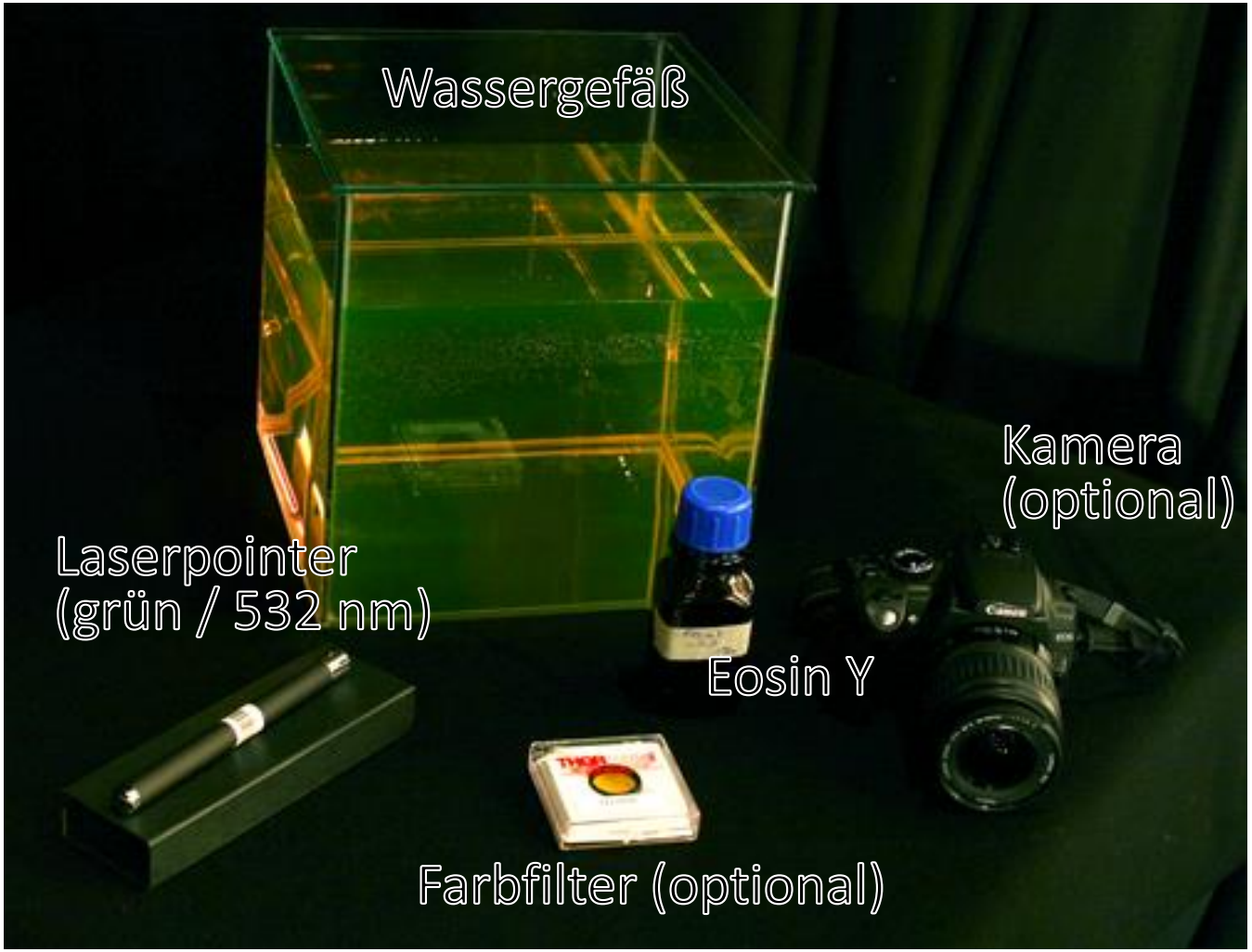
Exkurs: Fluoreszenz

- Absorptions- und Emissionsspektrum eines fluoreszierenden Farbstoffs (Eosin Y, Y=„yellowish“)



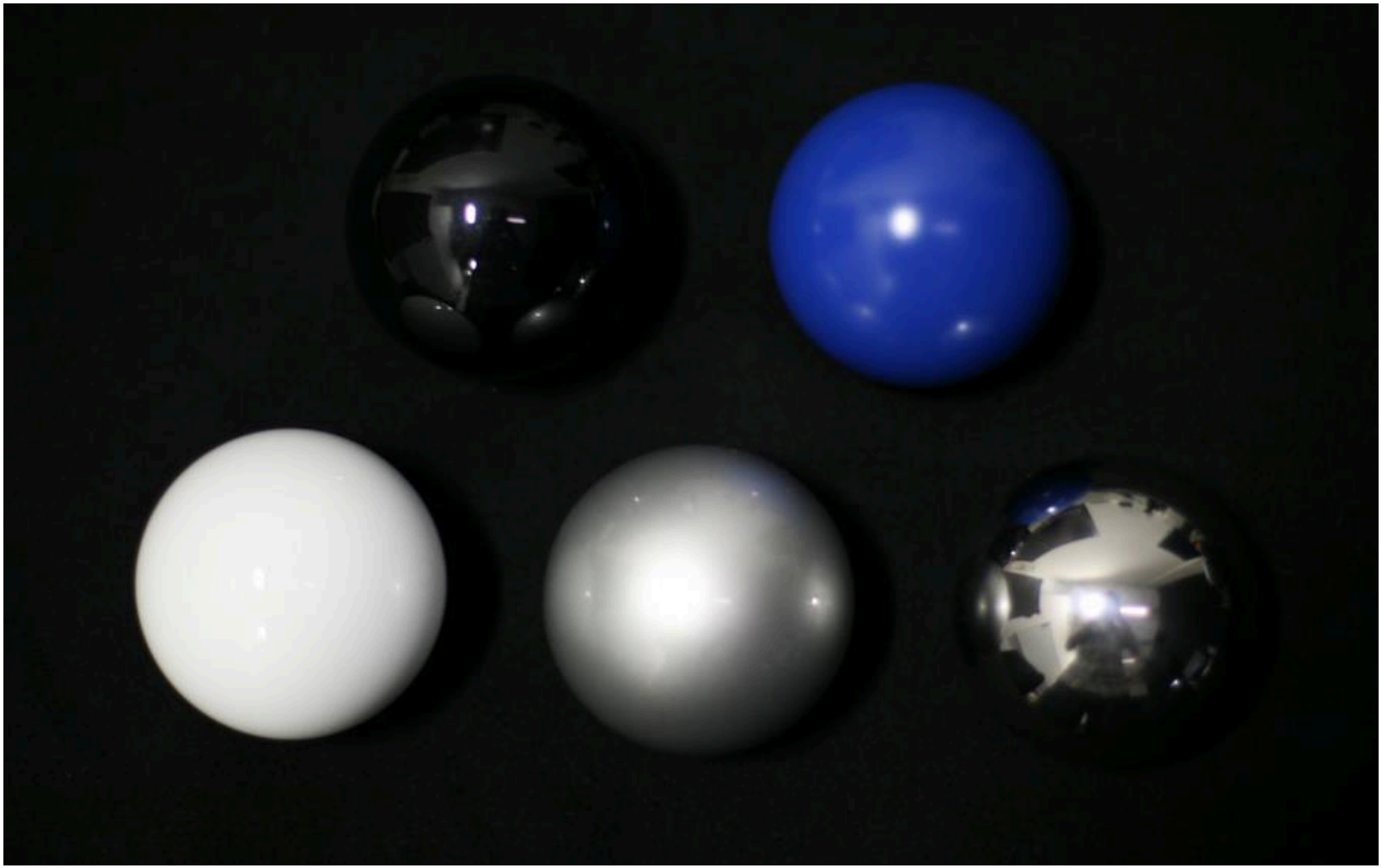
- ein Streu-/Reemissionsvorgang ändert die Wellenlänge des Photons: (praktisch) nur einfache Streuung möglich

Exkurs: Gerätschaften



[22]

Exkurs: Oberflächenreflexion



SpiegelInd

Schwarz, spekular

Diffus weiß, mit Lackschicht

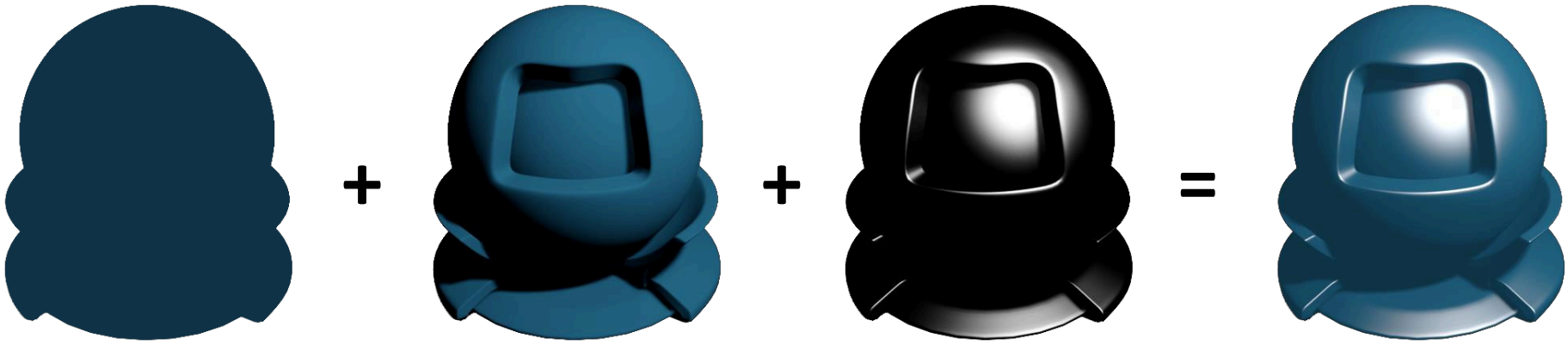
Silber, glänzend

Diffus und glänzend

[22]

Phong-Beleuchtungsmodell [1]

- … ist ein phänomenologisches Modell und modelliert die Reflexion mit drei Komponenten
  - **ambient**: indirekte Beleuchtung, Licht von anderen Oberflächen
  - **diffus**: nach dem Lambertschen Gesetz
  - **spekular**: imperfekte Spiegelung



ambient ideal-diffus spekular/glänzend

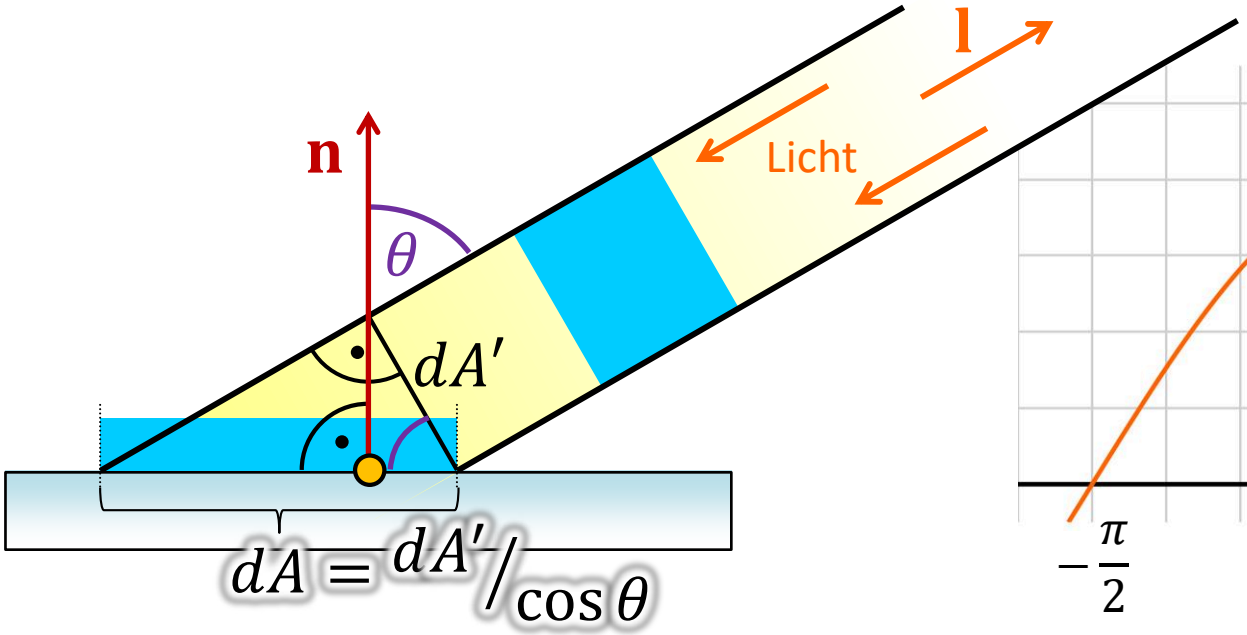
- … lässt sich natürlich als BRDF $f_r(\mathbf{l}, \mathbf{x}, \mathbf{v})$ schreiben, wir verwenden aber zunächst eine anschaulichere Schreibweise (und gehen Folgenden davon aus, dass alle Richtungsvektoren normiert sind!)

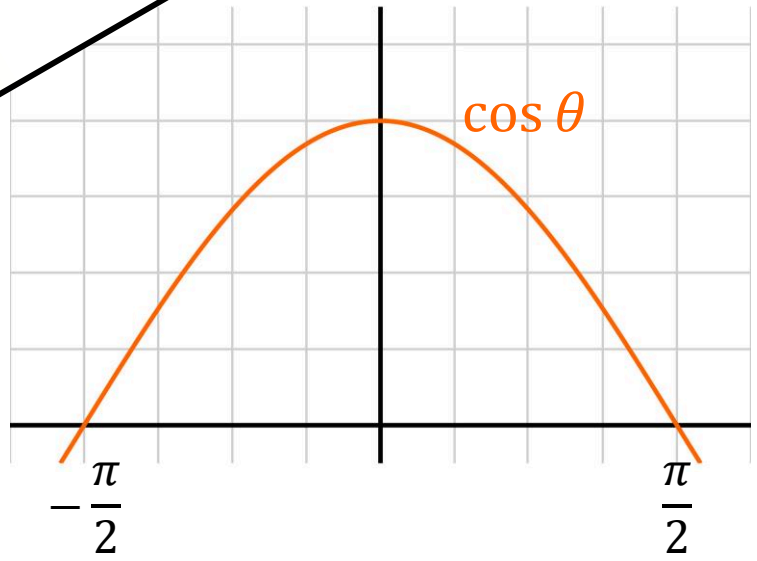
[1]Bùi Tường Phong, *Illumination for Computer Generated Pictures.* Comm. of the ACM 18, 1975

Phong-Beleuchtungsmodell

Lambertsche (ideal-diffuse) Reflexion

- Lichtstrahl aus Richtung $\mathbf{l}$ zur Lichtquelle mit Intensität I_L und Materialkoeffizient k_d (geg. für Wellenlängen oder RGB)
- Bestrahlungsstärke der Fläche: $I_L \, {dA'}/{dA} \sim \cos\theta$
- modelliere diffuse Reflexion mit $k_d \cdot I_L \cdot \cos\theta$, wenn $|\theta| \leq \frac{\pi}{2}$
  - mit Materialkoeffizient k_d und mit $\cos\theta = \mathbf{n} \cdot \mathbf{l}$
  - Erinnerung: Skalarprodukt definiert als $\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\phi$

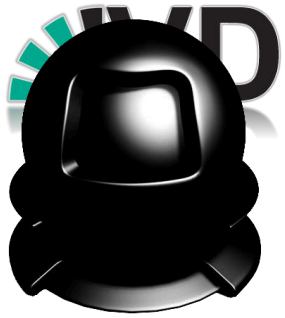




Lambertsche (ideal-diffuse) Reflexion

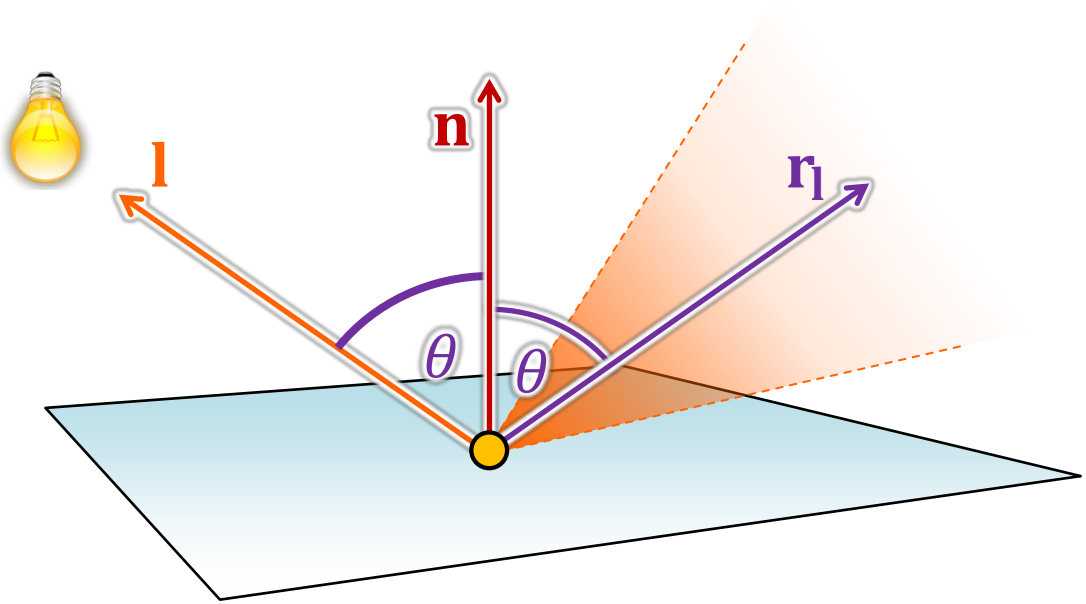
Bild: Andrea Weidlich

Phong-Beleuchtungsmodell



Spekulare Reflexion: Herleitung perfekte Spiegelung

- durch gerichtete Reflexion entstehen Glanzlichter
- zentriert um die Richtung der *perfekten* Spiegelung $\mathbf{r}_\mathbf{l}$

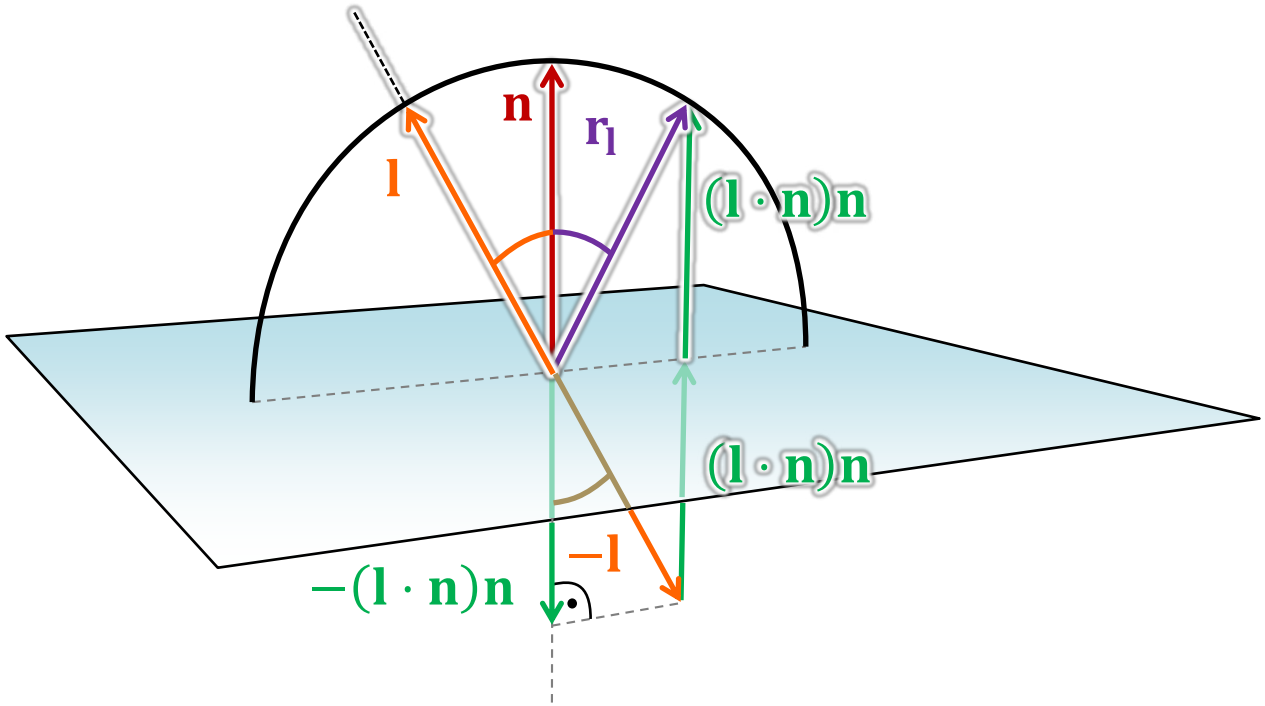


Phong-Beleuchtungsmodell

Spekulare Reflexion: Herleitung perfekte Spiegelung

- Spiegelung des Vektors $\mathbf{l}$ an der Oberfläche mit der Normale $\mathbf{n}$
- die Vektoren $\mathbf{l}$, $\mathbf{n}$, und $\mathbf{r}_\mathbf{l}$ liegen in einer Ebene

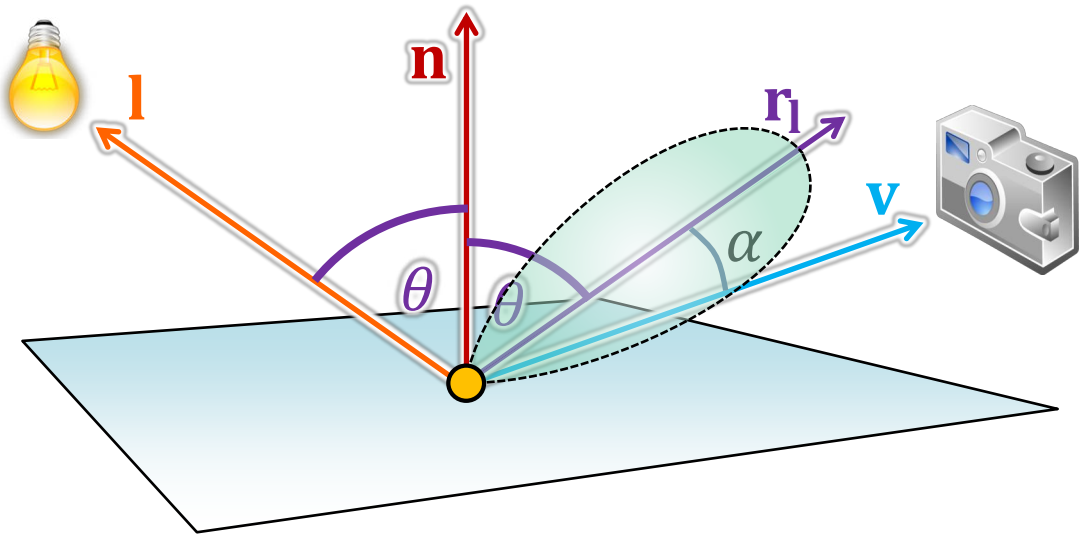
$$\mathbf{r}_\mathbf{l} = 2(\mathbf{l} \cdot \mathbf{n})\mathbf{n} - \mathbf{l}$$

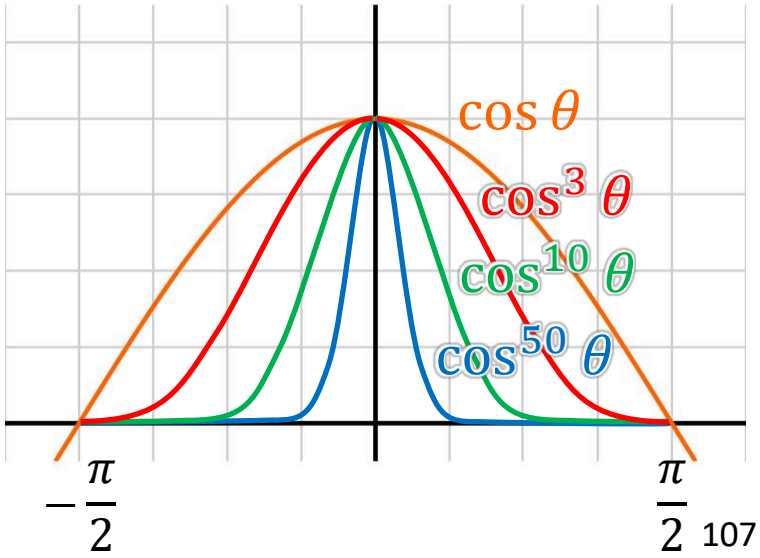


Phong-Beleuchtungsmodell

Spekulare Reflexion

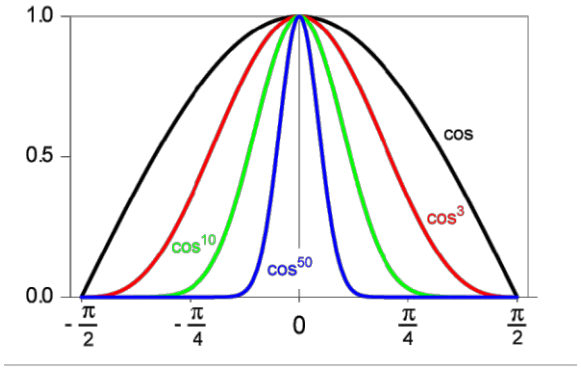
- *perfekte* Spiegelung in Richtung $\mathbf{r_l} = 2(\mathbf{l} \cdot \mathbf{n})\mathbf{n} - \mathbf{l}$
- Stärke der *imperfekten* Spiegelung fällt für von $\mathbf{r_l}$ verschiedene Richtungen ab: modelliere Abfall durch $\cos^n \alpha$
- spekulare Reflexion: $I_s = k_s \cdot I_\mathrm{L} \cdot \cos^n \alpha = k_s \cdot I_\mathrm{L} \cdot (\mathbf{r_l} \cdot \mathbf{v})^n$
  - spekularer Reflexionskoeffizient k_s und sog. Phong-Exponent n
  - n groß → kleine Glanzlichter, n klein → große Glanzlichter
  - Hinweis: $(\mathbf{r_l} \cdot \mathbf{v}) = (\mathbf{r_v} \cdot \mathbf{l})$

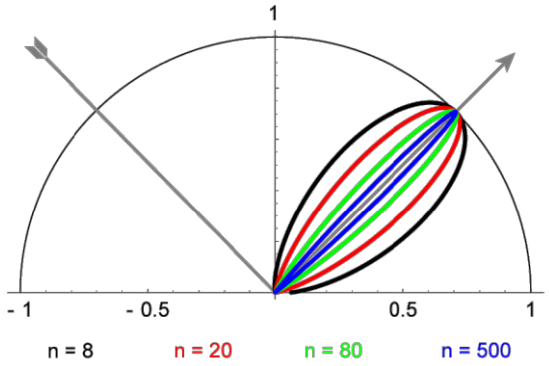


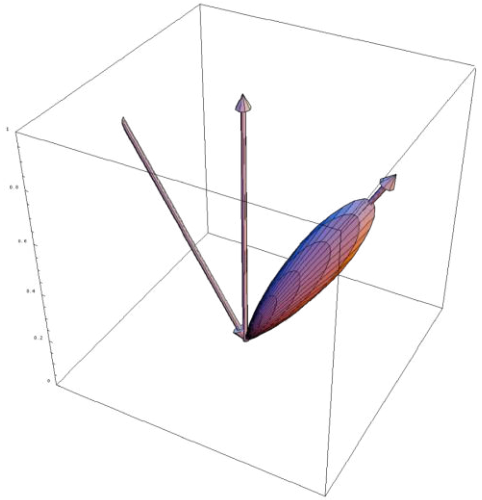


Spekulare Reflexion und Glanzlichter

- Auswirkungen des Phong-Exponenten







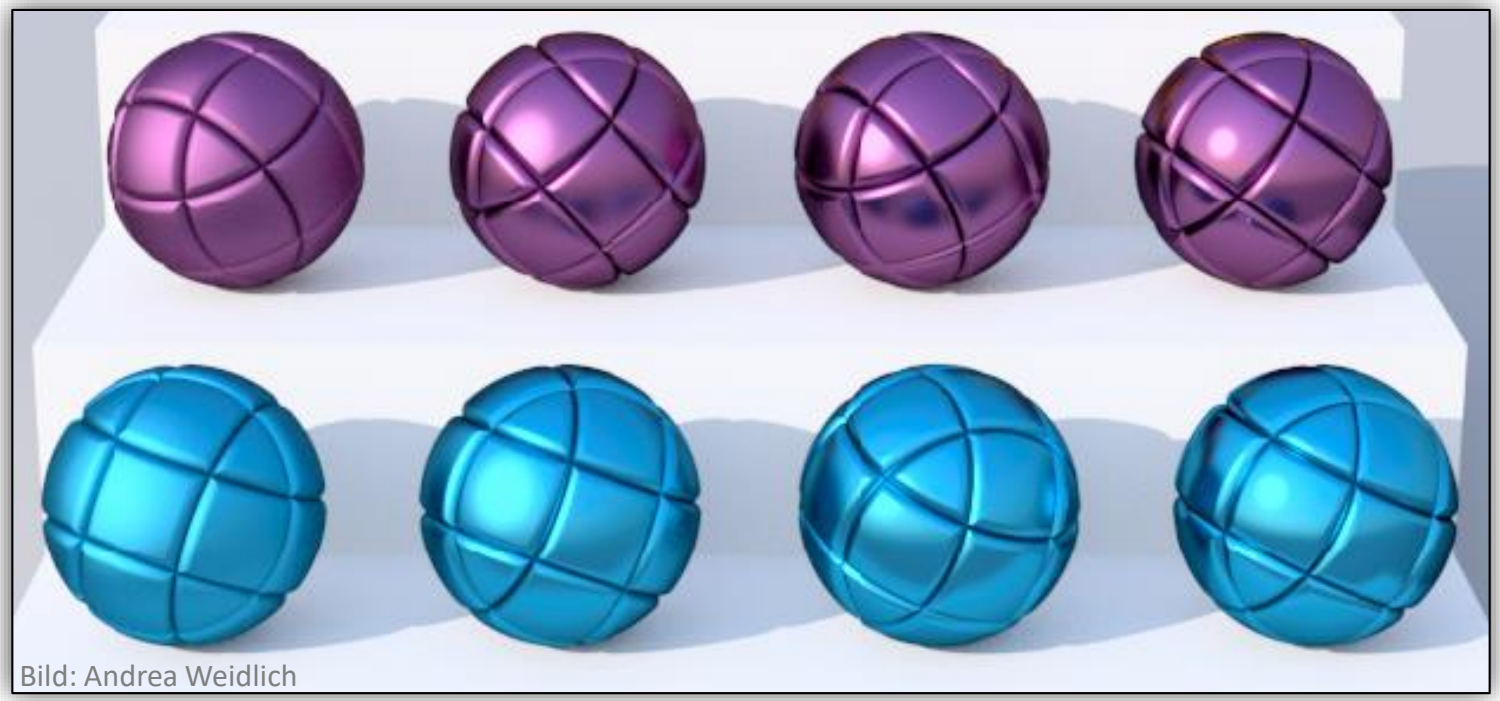


Bild: Andrea Weidlich

Spekulare Reflexion und Glanzlichter

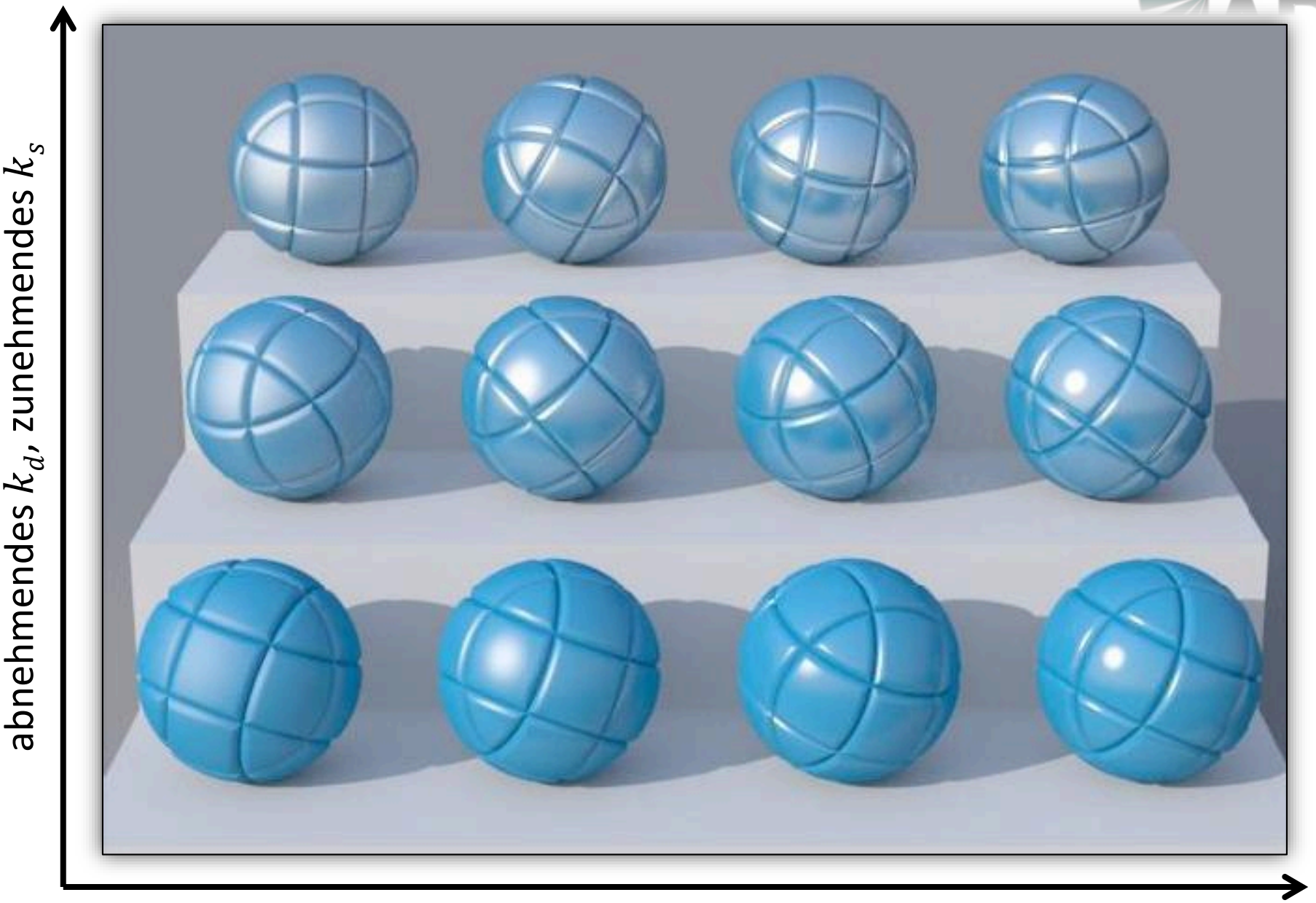


Bild: Andrea Weidlich

Phong-Beleuchtungsmodell

Zusammenfassung

- das Phong-Beleuchtungsmodell ist ein phänomenologisches Modell mit drei Komponenten ambient, diffus und spekular:

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{n} \cdot \mathbf{l}) + k_s \cdot I_L \cdot (\mathbf{r_l} \cdot \mathbf{v})^n$$

- Bemerkungen:
  - k_a, k_d, k_s und Lichtintensität I_L sind wellenlängenabhängig
  - k_a, k_d repräsentieren die Eigenfarbe der Oberfläche
  - Glanzlichter haben bei Metallen Farbe der Oberfläche ($k_s \propto k_d$), sonst meist die Farbe der Lichtquelle
  - wir sind nur an Richtungen interessiert, für die die Skalarprodukte positiv sind; daher schreibt man oft z.B. $(\mathbf{n} \cdot \mathbf{l})^+ := \max(0, (\mathbf{n} \cdot \mathbf{l}))$


[24]

- physikalisch-korrekt?
  - welche Werte dürfen k_a, k_d, k_s annehmen?

Raytracing Pseudocode

```
for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t' = intersect( object, e, d );
      if ( t' > 0 && t' < t ) {
        intersection = object;
        t = t';
      }
    }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) { ... }
} }
```

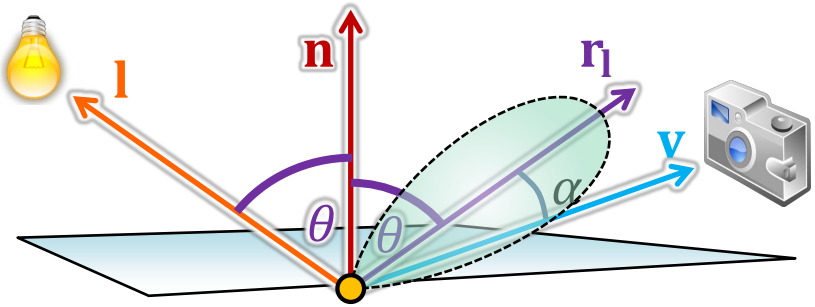
Beleuchtungsberechnung im Raytracer

```
struct vec3 {           // wir verwenden in der Übung die
  float x, y, z;        // OpenGL Mathematics (GLM) Library
  ...
};

class Material {
  vec3  ka, kd, ks;     // Reflexionsparameter (hier: RGB, oder Spektrum)
  float n;              // Phong-Exponent
  ...
};

struct Intersection {
  vec3 p, n, v;         // Position, Normale,
                        // Vektor zur Kamera/zum Startpunkt des Strahls
  Material *material; // Reflexionsparameter
};

class PointLight : public LightSource {
  vec3 pos,             // Position
       I_L;             // Intensität der LQ (hier: RGB, oder Spektrum)
  ...
};
```

Beleuchtungsberechnung im Raytracer

```
vec3 PointLight::computeDirectLight( const Intersection &i, ... )
{
  // ambienter Term
  vec3 I = i.material->ka * I_L;

  // hier: Punktlichtquelle
  vec3 L = normalize( pos - i.p );
  float  NdotL = dot( i.n, L );

  if ( NdotL > 0 ) {
    // diffuser Term
    I += i.material->kd * I_L * NdotL;

    // spekularer Term
    vec3  R = 2.0f * i.n * NdotL - L;
    float RdotV = dot( R, V );
    if ( RdotV > 0 ) {
      I += i.material->ks * I_L * powf( RdotV, i.material->n );
    }
  }
};
```

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{n} \cdot \mathbf{l}) + k_s \cdot I_L \cdot (\mathbf{r_l} \cdot \mathbf{v})^n$$



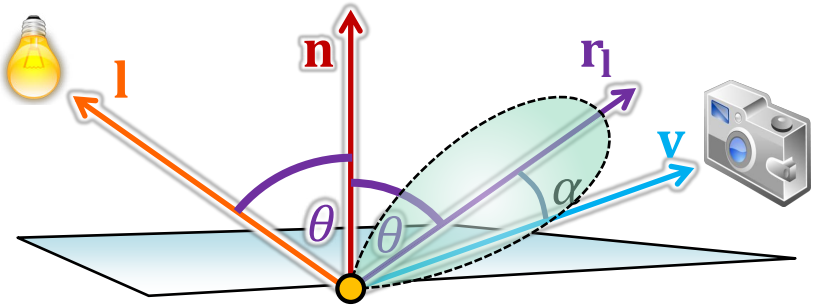
Beleuchtungsberechnung im Raytracer

```
vec3 PointLight::computeDirectLight( const Intersection &i, ... )
{
  // ambienter Term
  vec3 I = i.material->ka * I_L;

  // hier: Punktlichtquelle
  vec3 L = normalize( pos - i.p );
  float  NdotL = dot( i.n, L );
```

$$I = k_a \cdot I_L + k_d \cdot I_L \cdot (\mathbf{n} \cdot \mathbf{l}) + k_s \cdot I_L \cdot (\mathbf{r_l} \cdot \mathbf{v})^n$$

die Intensität einer „realistischen“ **Punkt**lichtquelle würde mit dem Abstandsquadrat abfallen:

```
float dist = length( pos - i.p );
...
I += i.material->kd * I_l * NdotL /
     ( dist * dist );
```



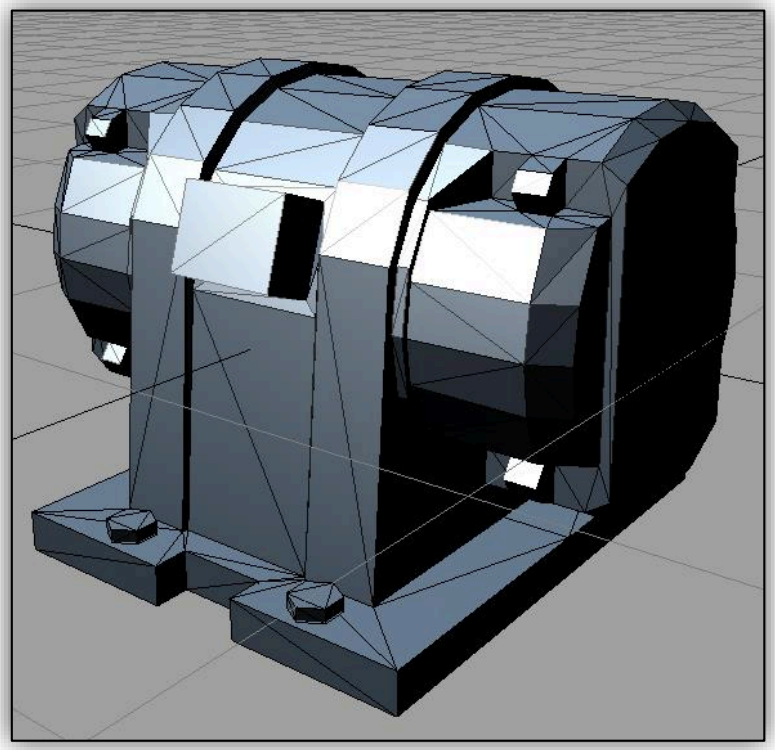

```
    float RdotV = dot( R, V );
    if ( RdotV > 0 ) {
      I += i.material->ks * I_L * powf( RdotV, i.material->n );
    }
  }
};
```

Die Beiträge mehrerer Lichtquellen werden aufsummiert (Superpositionsprinzip)

Schattierung von Dreiecksnetzen

Beleuchtungsberechnung mit welcher Normale?

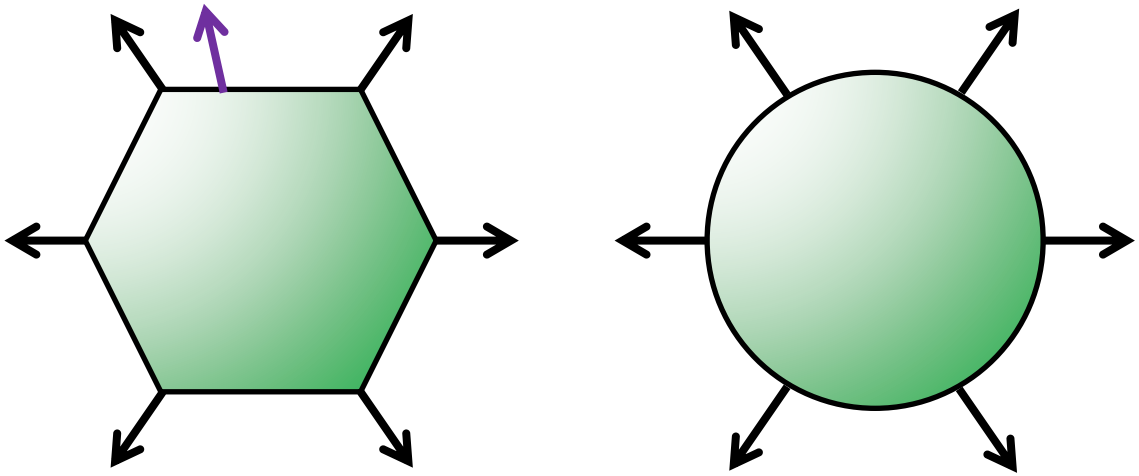
- Dreiecke: stückweise lineare Approximation einer (möglicherweise gekrümmten) Oberfläche
- **Flat Shading**: Verwendung der Dreiecksnormale für Beleuchtung



Schattierung von Dreiecksnetzen

Beleuchtungsberechnung mit welcher Normale?

- Illusion einer glatten, gekrümmten Fläche durch
  - Normalenvektoren an den Vertices/Eckpunkten („Vertex-Normalen“)
  - Interpolation der Normalen über den Dreiecken

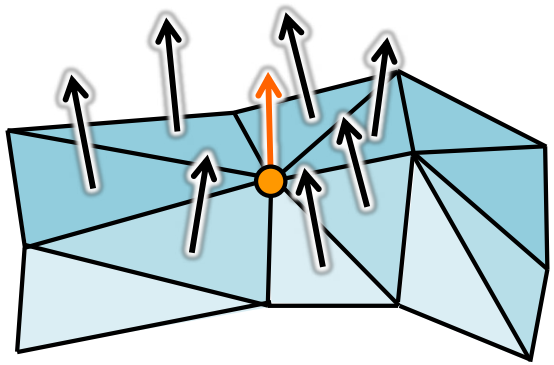


- Vertex-Normalen können entweder
  - berechnet werden (z.B. bei der Triangulierung parametrischer Flächen)
  - direkt aus einem Dreiecksnetz bestimmt werden

Schattierung von Dreiecksnetzen

Normalen für Dreiecksnetze

- Vertex-Normalen bestimmen über
  - (gewichtete) Summe der Normalen angrenzender Flächen
  - Normalen für ein Dreieck über Kreuzprodukt

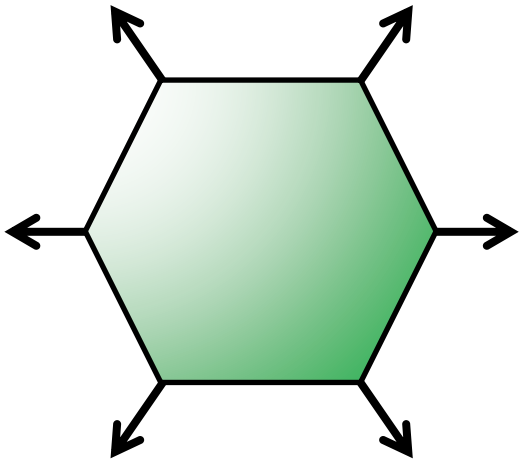


- Vorgehen
  - berechne Normale für jedes Dreieck
  - für jeden Vertex: summiere die Normalen aller angrenzenden Dreiecke
    - optional: skaliere Normalen mit der Dreiecksfläche (ohne Zutun im Kreuzprodukt)
  - **normalisiere** die Vertex-Normalen
    - NICHT durch die Anzahl der Normalen teilen!

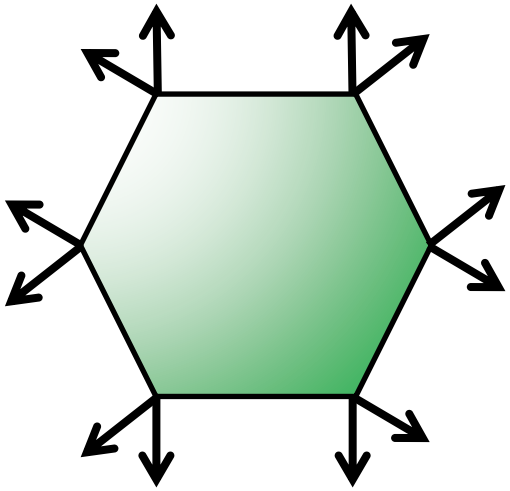
Schattierung von Dreiecksnetzen

Normalen für Dreiecksnetze

- scharfe Kanten brauchen mehrere Normalen
  - Kante, wenn der Winkel zweier benachbarter Dreiecke zu groß ist
- bei Dreiecksnetzen werden pro Dreieck typischerweise 3 Eckpunkte und 3 Normalen (und zusätzliche Attribute, z.B. Farben) gespeichert
  - bei Mittelung pro Vertex: berücksichtige nur Dreiecksnormalen, deren Winkel zur Normale des „eigenen“ Dreiecks nicht zu groß ist



glatte Schattierung



scharfe Kanten

Schattierung von Dreiecksnetzen

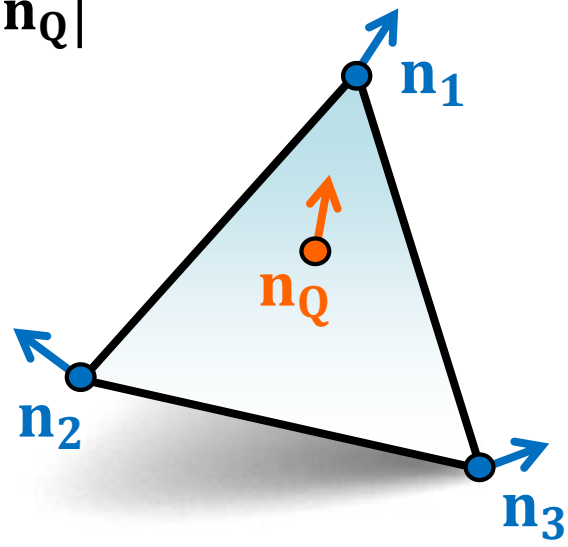
Interpolation von Normalen (z.B. für Schnittpunkt für Schattierung)

- Berechnung durch lineare komponentenweise Interpolation der Vertex-Normalen anhand der baryzentrischen Koordinaten

- Beispiel:
  - geg.: Normalen $\mathbf{n_1}, \mathbf{n_2}, \mathbf{n_3}$ eines Dreiecks $\triangle(\mathrm{P_1}, \mathrm{P_2}, \mathrm{P_3})$
  - ges.: interpolierte Normale $\mathbf{n_Q}$ an einem Punkt Q auf dem Dreieck
  - Lösung: berechne $\lambda_1, \lambda_2, \lambda_3$, dann ist $\mathbf{n_Q} = \lambda_1 \mathbf{n_1} + \lambda_2 \mathbf{n_2} + \lambda_3 \mathbf{n_3}$

- Beleuchtungsberechnung mit der Normale $\mathbf{n_Q}/|\mathbf{n_Q}|$
  - lineare komponentenweise Interpolation ist nicht längenerhaltend!

- Beleuchtungsberechnung mit interpolierten Normalen nennt man **Phong Shading** (≠ Phong Beleuchtungsmodell)



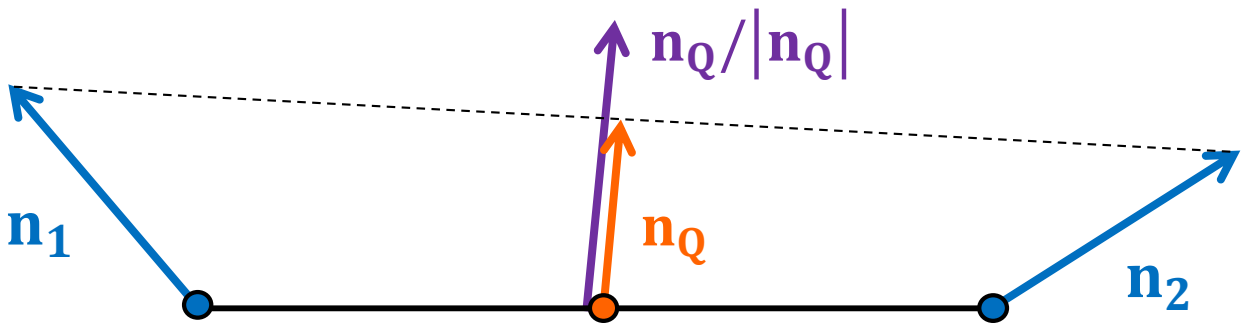
Schattierung von Dreiecksnetzen

Interpolation von Normalen (z.B. für Schnittpunkt für Schattierung)

- Berechnung durch lineare komponentenweise Interpolation der Vertex-Normalen anhand der baryzentrischen Koordinaten

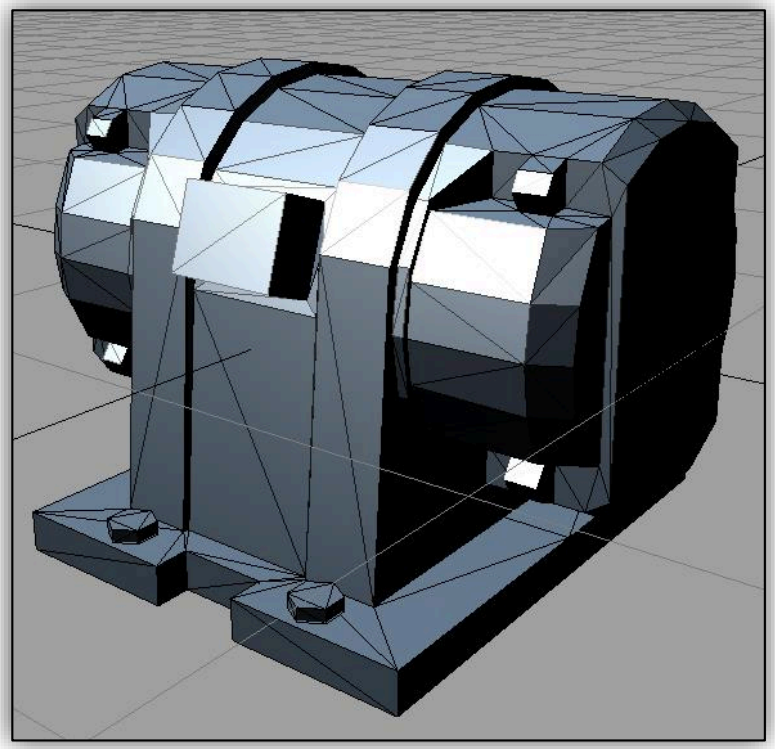
- Beleuchtungsberechnung mit der Normale $\mathbf{n_Q}/|\mathbf{n_Q}|$
  - lineare komponentenweise Interpolation ist nicht längenerhaltend!

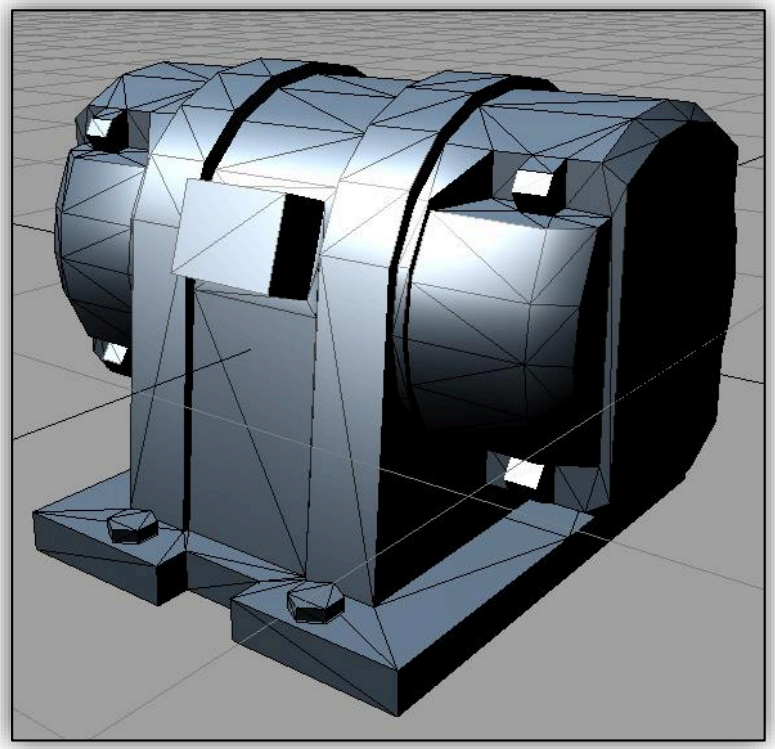
- Darstellung in 2D: Interpolation entlang einer Kante



Schattierung von Dreiecksnetzen

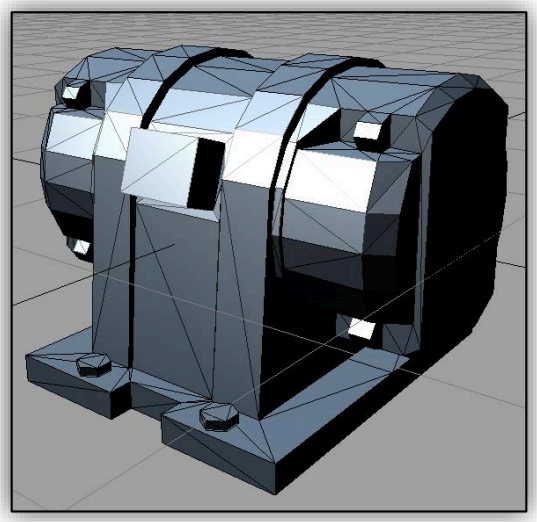
- Flat Shading (links) und Phong Shading (rechts, mit Erhalten scharfer Kanten, also 3 Normalen pro Dreieck gespeichert)

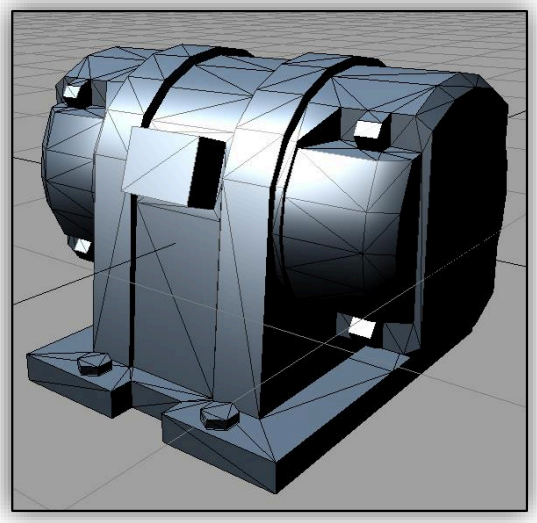


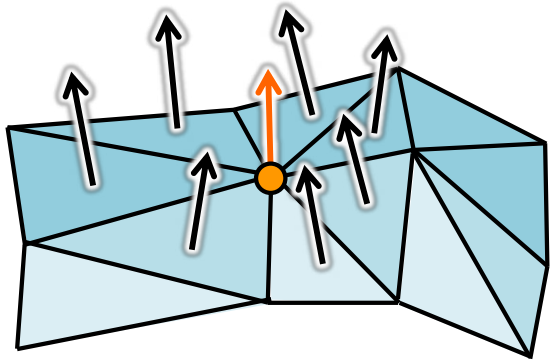


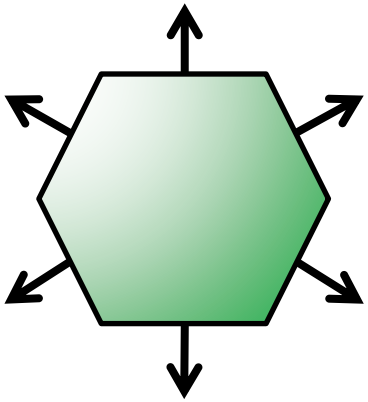
Zusammenfassung: Schattierung von Dreiecksnetzen

Normalen für Shading

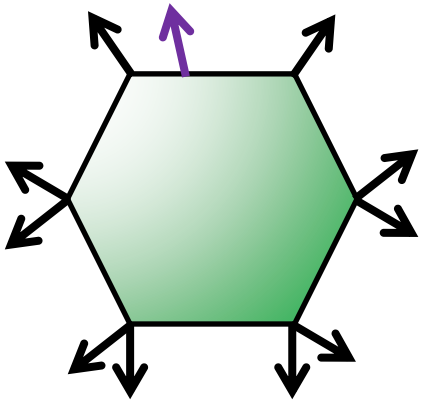




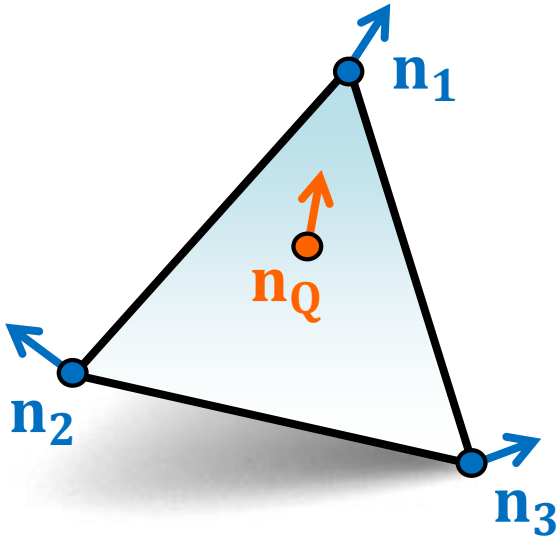




Flat Shading
(Dreiecksnormale)

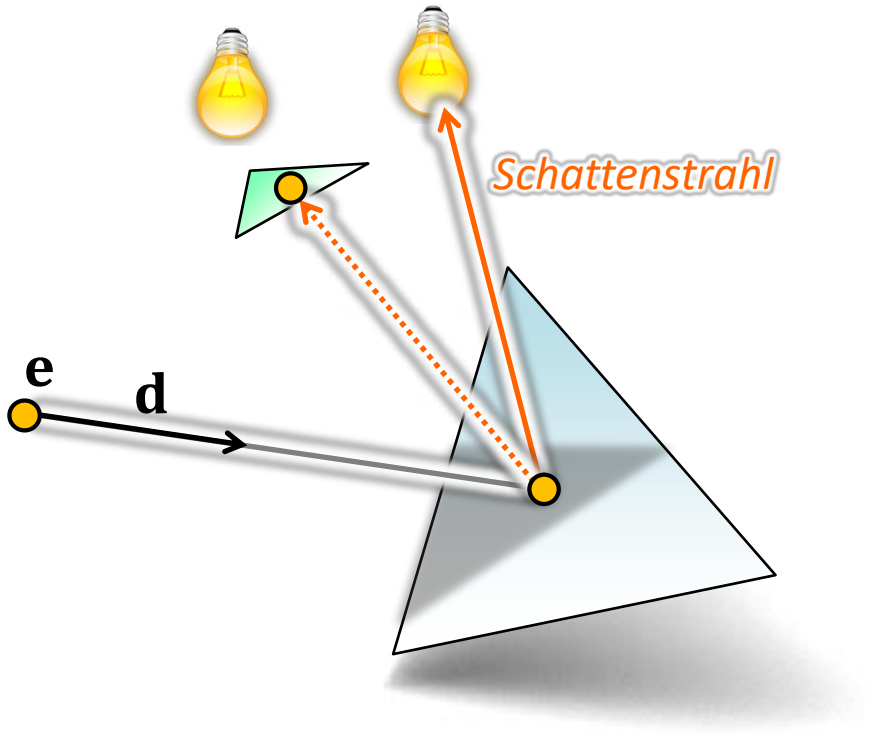


Phong Shading
(interpolierte Normale)



Licht und Schatten

- bisher: eine Oberfläche wird immer beleuchtet, wenn sie nicht von der Lichtquelle abgewandt ist
- aber: andere Objekte der Szene können Schatten werfen
- Lösung: sende einen „Schattenstrahl“ zur Lichtquelle hin
  - teste, ob dieser Strahl auf dem Weg **von der Oberfläche zur Lichtquelle** ein anderes Objekt schneidet



Schattenberechnung im Raytracer

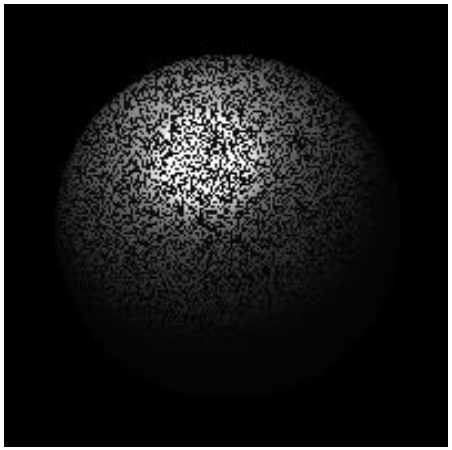
```
vec3 PointLight::computeDirectLight( const Intersection &i, ... ) {
  // ambienter Term
  vec3 I = i.material->ka * I_L;

  vec3 L = pos - i.p;

  // Schattenstrahl
  float dist2Light = length( L );

  for ( each object ) {
    t' = intersect( object, i.p, L );
    if ( t' > 0 && t' < dist2Light )
      return I;
  }
  ...
};
```

- i.d.R. zwei unterschiedliche Routinen in einem Raytracer
  - teste ob irgendein Schnittpunkt für $t \in (0; t_{max})$
  - suche den nächsten Schnittpunkt für $t > 0$
- einfache Optimierung: Schattenstrahl nur wenn **`(NdotL > 0)==true`**
- semi-transparente Flächen reduzieren I_L

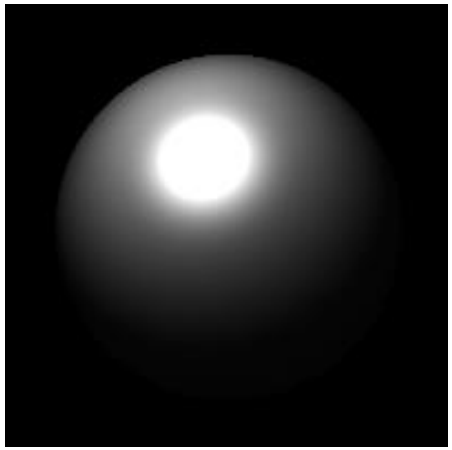
Schattenberechnung im Raytracer

- Problem: endliche Genauigkeit bei der Gleitkommadarstellung
  - wird der Schattentest so durchgeführt, wie auf der vorherigen Folie beschrieben, treten Artefakte auf:
    der Strahl schneidet u.U. die Oberfläche von der er startet nochmal!

- prinzipiell zwei Möglichkeiten
  - teste explizit, ob nochmals dasselbe Objekt geschnitten wurde:
    funktioniert nur bei konvexen Objekten (z.B. Dreiecke, Kugeln)
  - üblich: starte etwas entfernt von der Oberfläche, z.B. mit
    `t' = intersect( object, i.p + epsilon * i.n, L );`



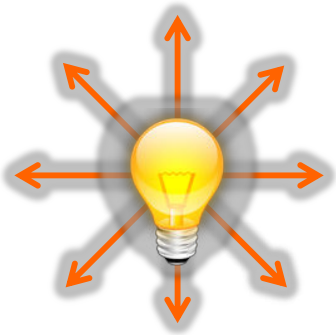
$\epsilon = 0$

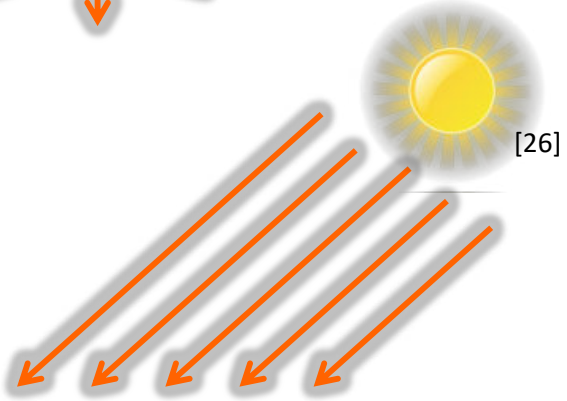


$\epsilon = 10^{-7}$

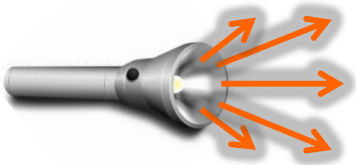
Arten von Lichtquellen

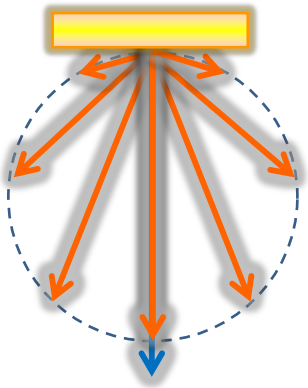
- Punktlichtquelle
  - definiert durch Position $\mathbf{p}$ und Intensität I_L $[\mathrm{W/sr}]$
  - reale Lichtquellen: Abfall der Intensität mit dem Abstandsquadrat
- paralleles Licht/direktionale Lichtquelle
  - definiert durch Richtung $\mathbf{d}$ und Flussdichte E $[\mathrm{W/m^2}]$
  - Bsp. Sonnenlicht ist annähernd parallel
- weitere Lichtquellentypen
  - Strahler (Spot-Lights): Lichtkegel, Abstrahlungscharakteristik mit $\cos^n \theta$
  - Punktlichtquellen mit Richtungscharakteristik
  - Flächenlichtquellen





[26]





Raytracing Pseudocode

```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t' = intersect( object, e, d );
      if ( t' > 0 && t' < t ) {
        intersection = object;
        t = t';
    } }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) {
      computeDirectLight ...
} } }
```

Strahlerzeugung (abh. vom Kameramodell)

```
generateRay( Ray *ray, int x, int y );
```

Strahlverfolgung und Beleuchtung

```
vec3 color =
  raytrace( Ray *ray, ... );
```

Raytracing Pseudocode

```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t' = intersect( object, e, d );
      if ( t' > 0 && t' < t ) {
        intersection = object;
        t = t';
    } }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) {
      computeDirectLight ...
} } }
```

Strahlerzeugung (abh. vom Kameramodell)

```
generateRay( Ray *ray, int x, int y );
```

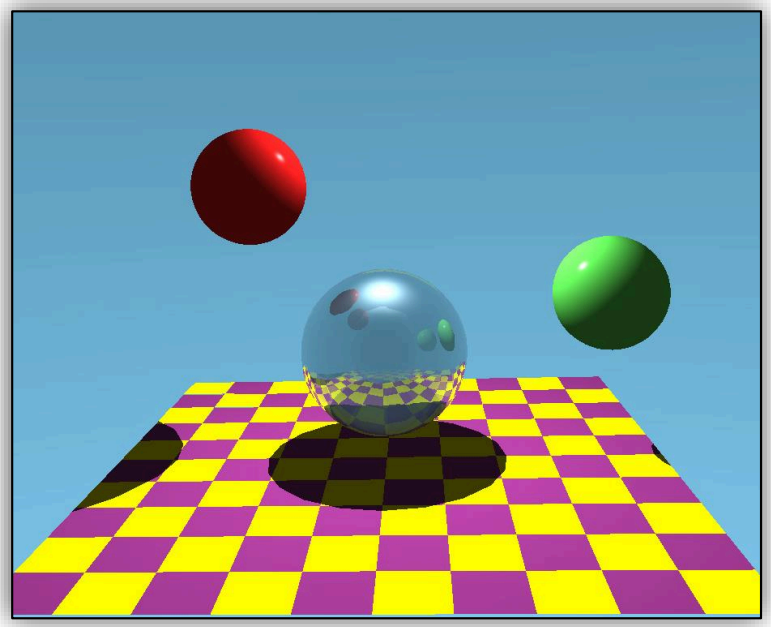
Schnittpunktberechnung

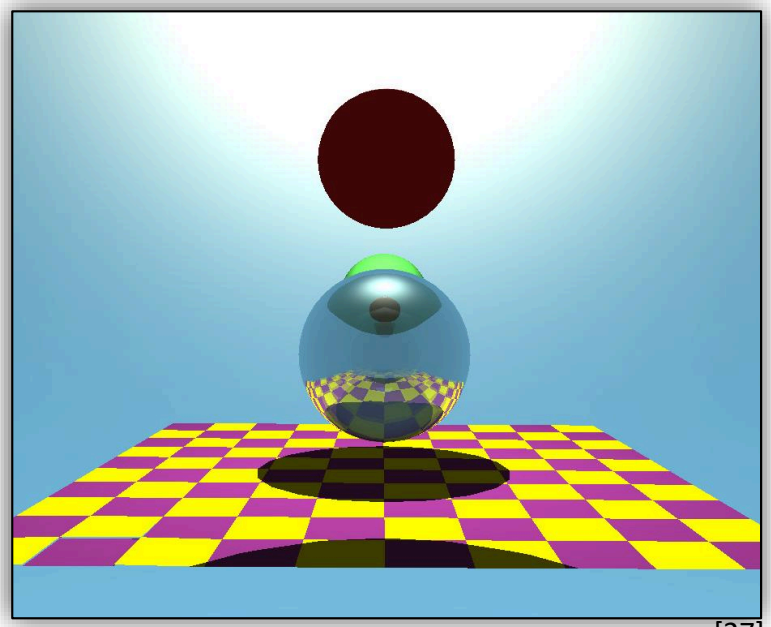
```
bool cast( Ray *ray, float maxDist );
```

Raytracing

Schritte

- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting):
  finde geometrisches Primitiv/Objekt, das den Sichtstrahl am nächsten zur Kamera schneidet
- Schattierung und Beleuchtungsberechnung (shading)
- Sekundärstrahlen für Spiegelung und Transmission



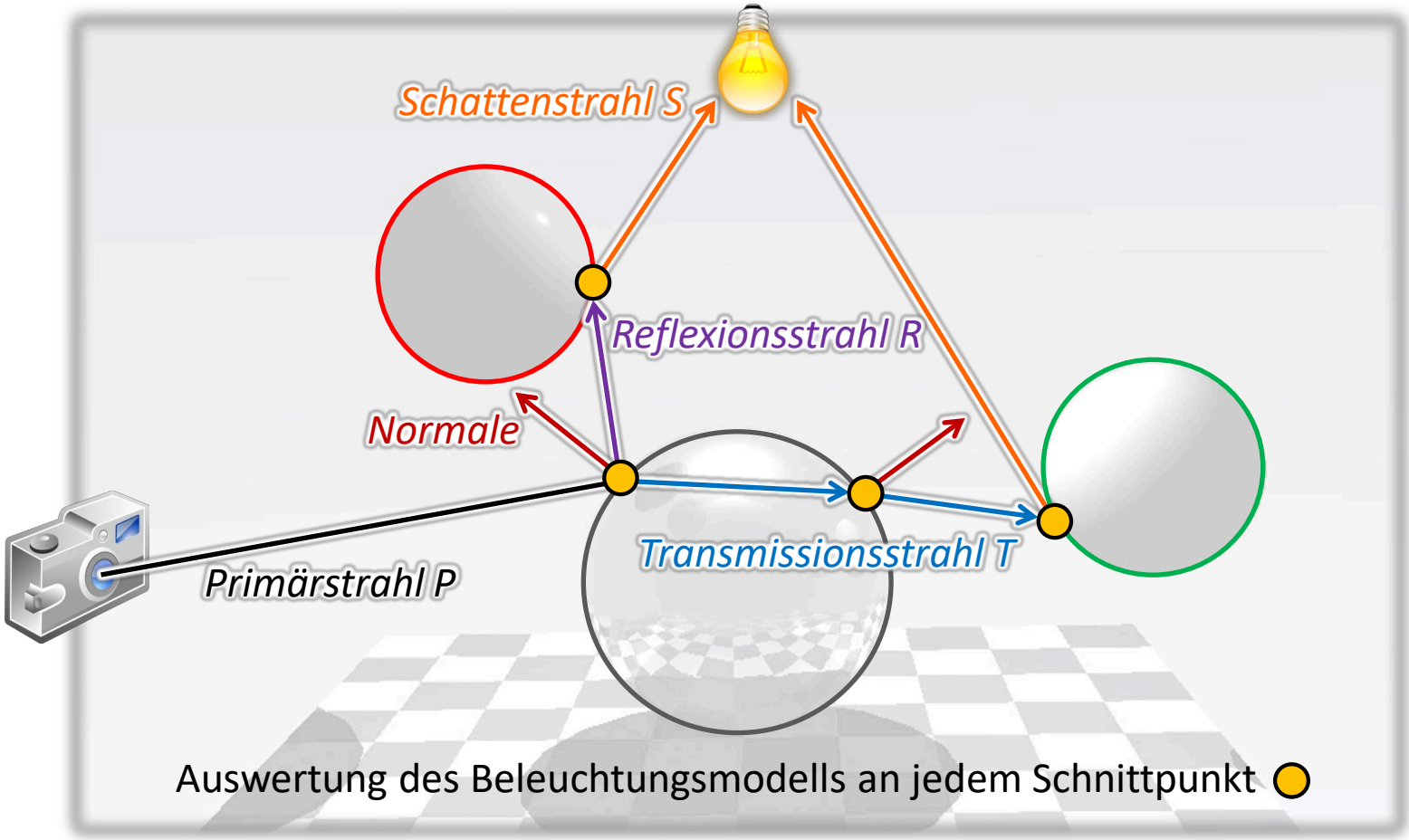


[27]

Sekundärstrahlen

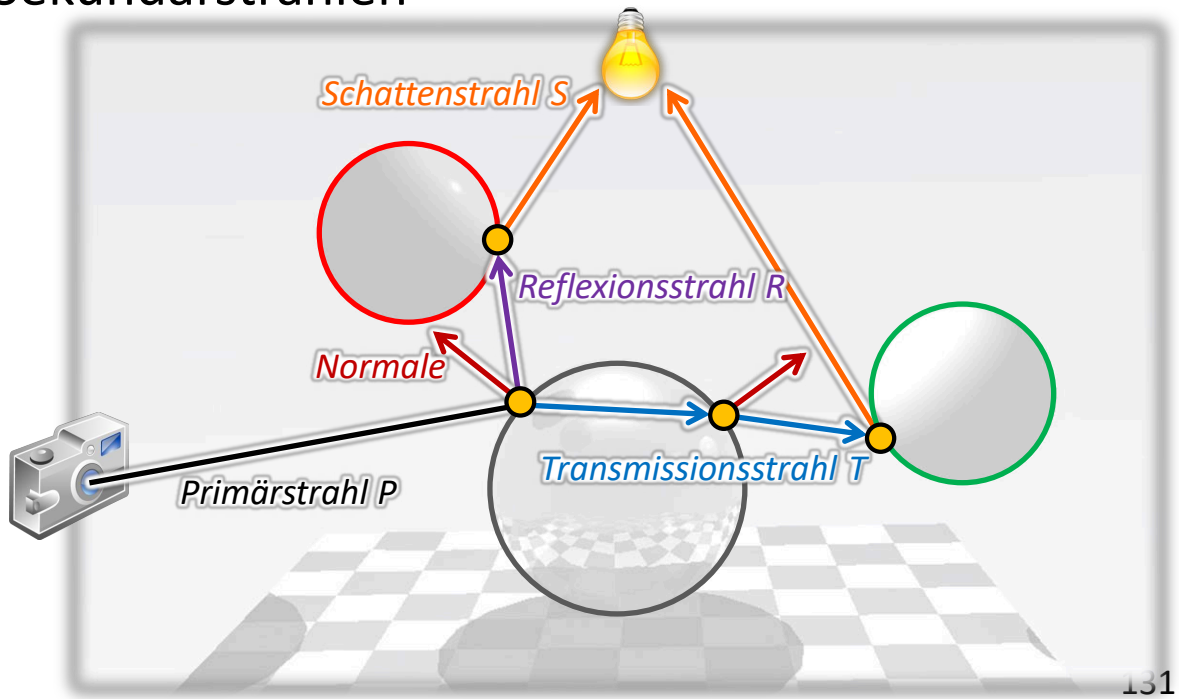
Berechnung von Spiegelungen und Transmission: Rekursives Raytracing

- bestimme Richtungen der Reflexions- und Transmissionsstrahlen (sog. Sekundärstrahlen)
- berechne Beitrag zur Farbe einer Oberfläche mit **`raytrace(...)`**



Spiegelungen, Reflexionsstrahlen

- Spiegelungen der Szene auf reflektierenden Oberflächen
  - Sichtstrahl *P* wird an der Oberfläche reflektiert → Reflexionsstrahl *R*
- behandle Reflexionsstrahl (fast so) wie einen Sichtstrahl: berechne Schnittpunkt und gebe Farbe zurück
  - Beleuchtungsberechnung für den Ursprung und „Blickrichtung“ von *R*
  - rekursive Verfolgung von (Mehrfach-)Reflexionen
  - weitere Schatten- und Sekundärstrahlen
- addiere Farbe des Reflexions- bzw. Transmissionsstrahls zur Farbe am Ausgangspunkt
  - gewichtet mit weiteren Koeffizienten k_r bzw. k_t



Raytracing Pseudocode

```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    u = l + (r-l) * (x+0.5) / width;
    v = t + (b-t) * (y+0.5) / height;
    s = ...;
    d = normalize( s );

    // finde nächsten Schnittpunkt
    intersection = NULL;
    float t = FLOAT_MAX;

    for ( each object ) {
      t' = intersect( object, e, d );
      if ( t' > 0 && t' < t ) {
        intersection = object;
        t = t';
    } }

    // Beleuchtungsberechnung
    if ( intersection != NULL ) {
      computeDirectLight ...
} } }
```

Strahlverfolgung und Beleuchtung

```
vec3 color =
  raytrace( Ray *ray, ... );
```

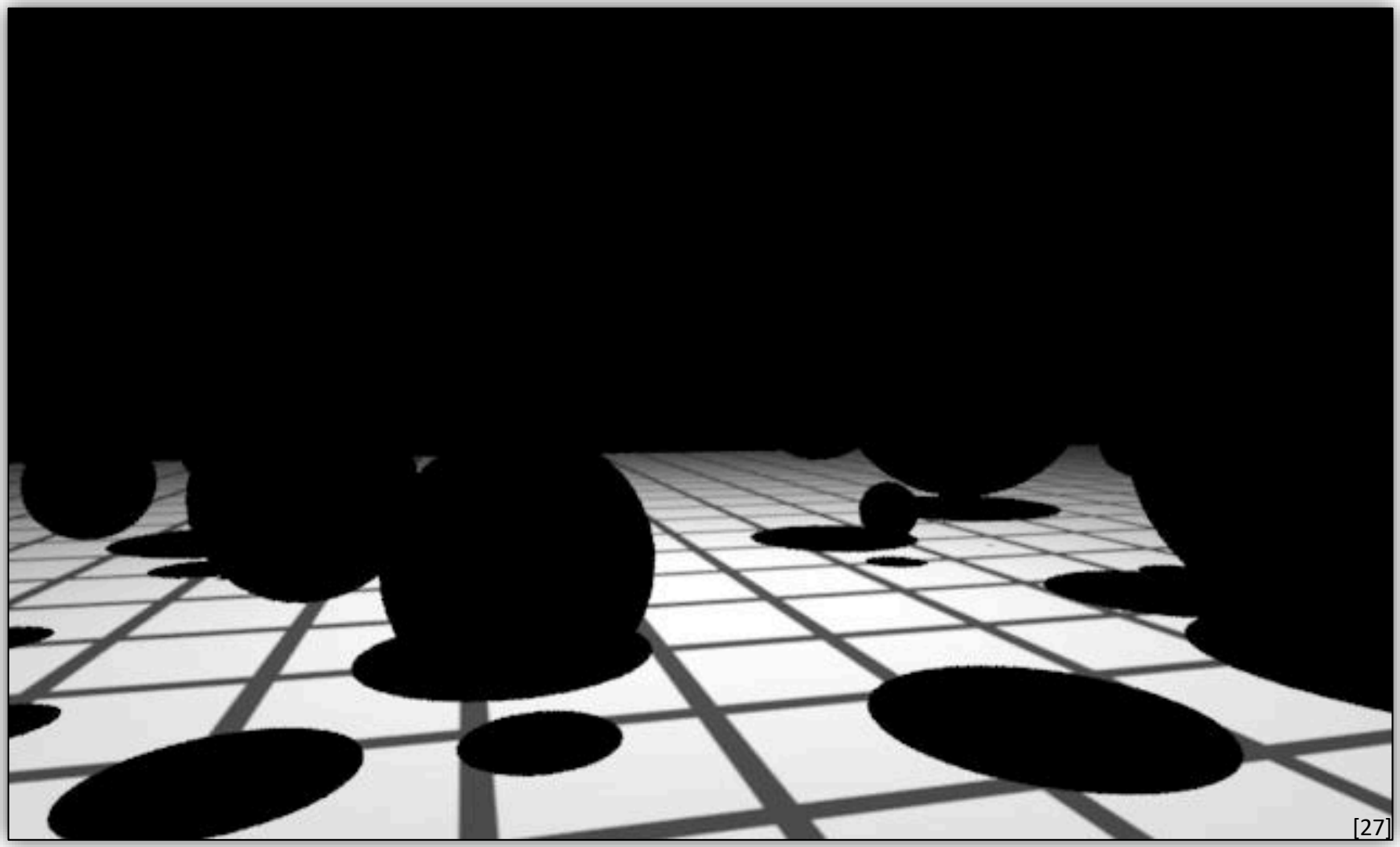
um rekursive Strahlen zu verfolgen

Raytracing Pseudocode

```
vec3 raytrace( Ray *ray, ... ) {
  // Farbe
  vec3 color = 0.0f;

  // Schnittberechnung
  Intersection i;
  if ( !cast( ray, FLOAT_MAX, &i ) )
    return color;          // optional: Umgebungslicht/Hintergrundfarbe

  for ( jede Lichtquelle )
    color += computeDirectLight( ... );

  if ( Fläche ist spiegelnd ) {
    // berechne Reflexionsstrahl
    Ray reflect = ...;
    color += i.kr * raytrace( reflect, ... );
  }

  return color;
}
```

Metallkugeln – Rekursionstiefe 0

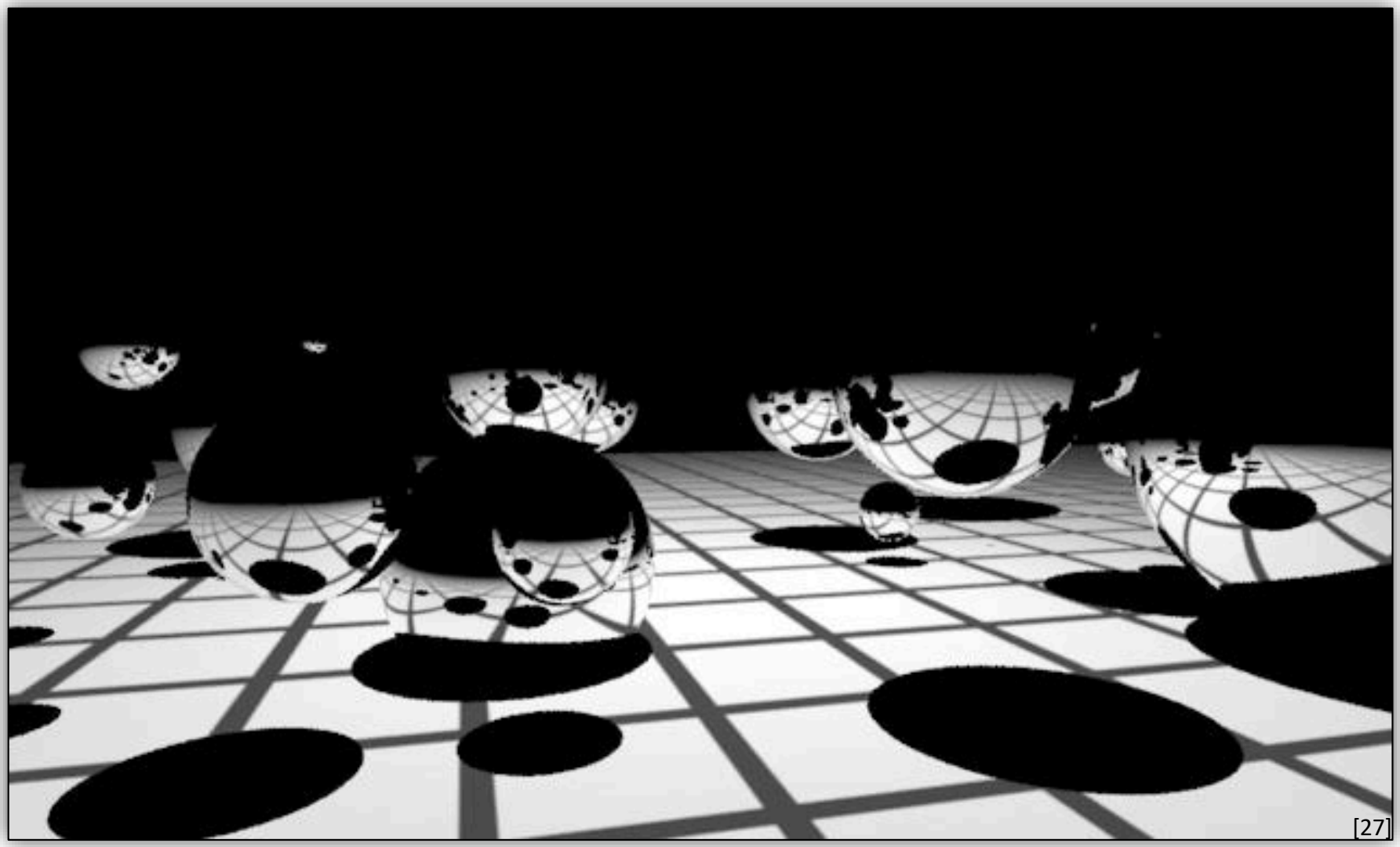
- Bild: Pat Hanrahan



Materialkoeffizienten der Metallkugeln: $k_a = k_d = k_s = k_t = 0$ und $k_r > 0$

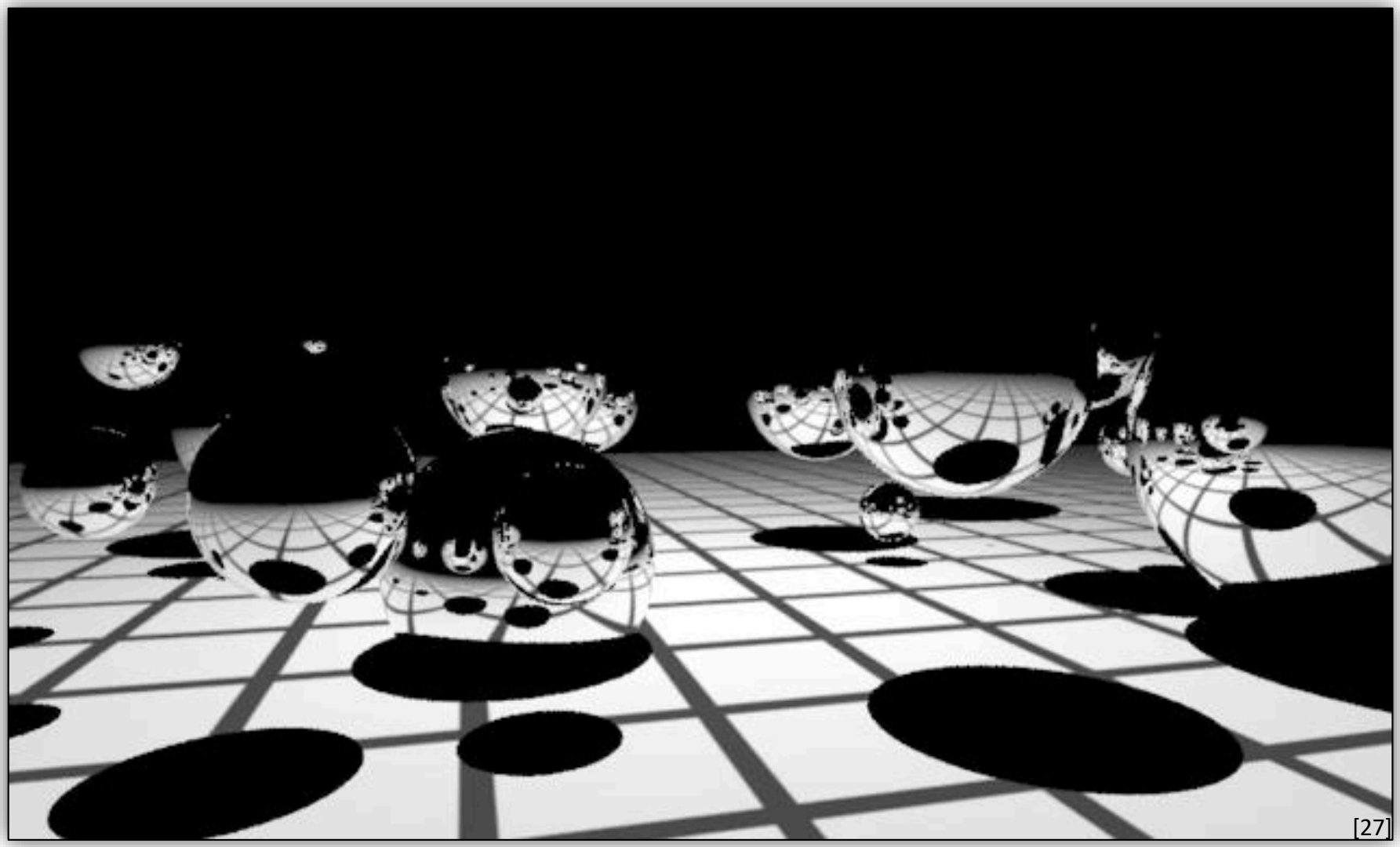
Metallkugeln – Rekursionstiefe 1

- Bild: Pat Hanrahan



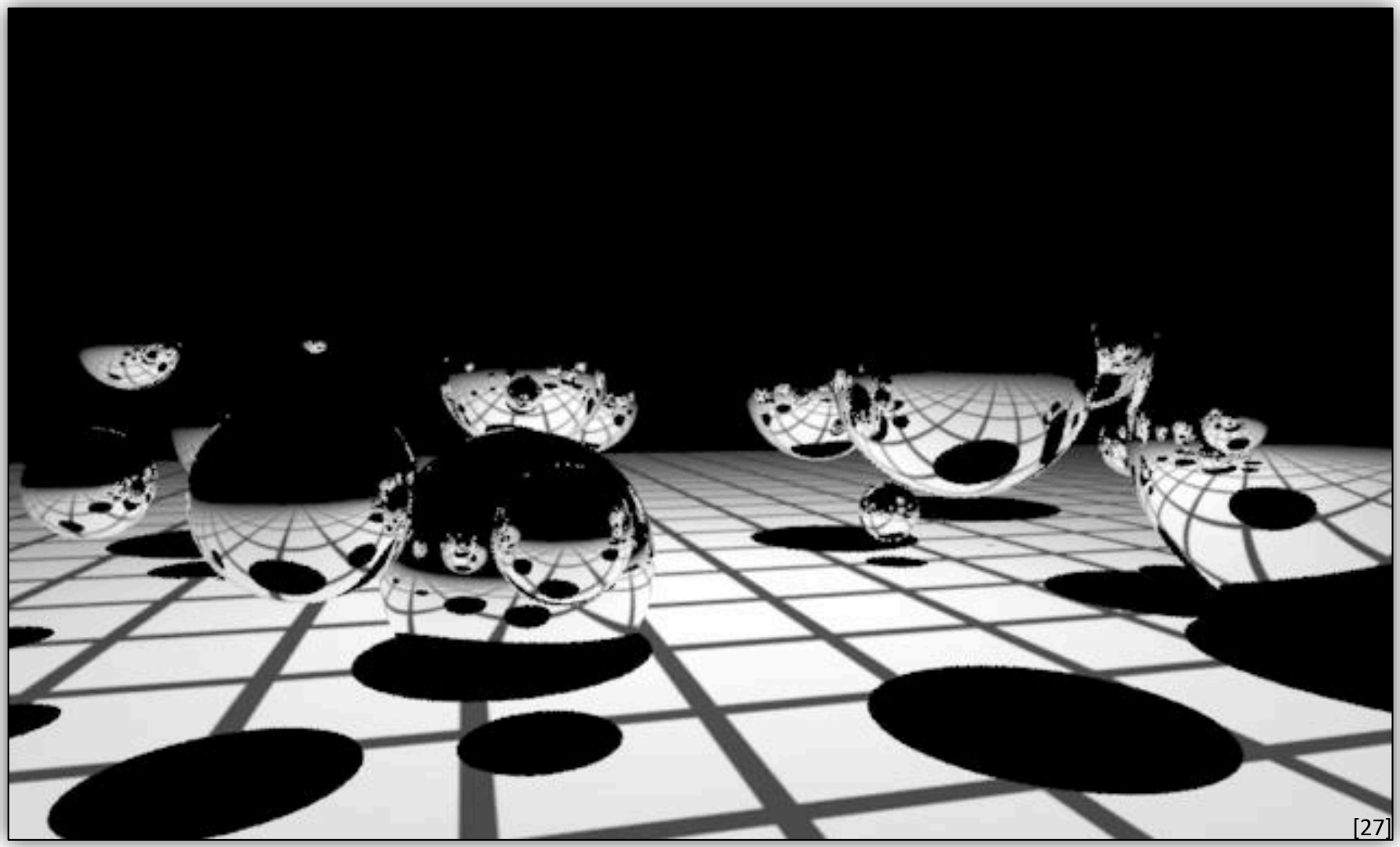
Metallkugeln – Rekursionstiefe 2

- Bild: Pat Hanrahan



Rekursionstiefe 5

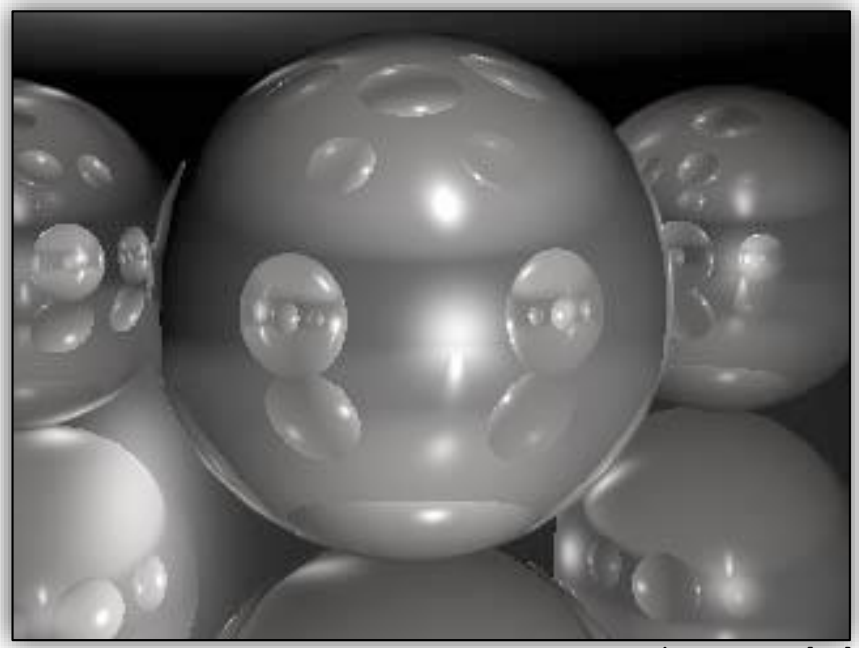
- Bild: Pat Hanrahan



Spiegelungen, Reflexionsstrahlen

Wie bestimmt man die Rekursionstiefe?

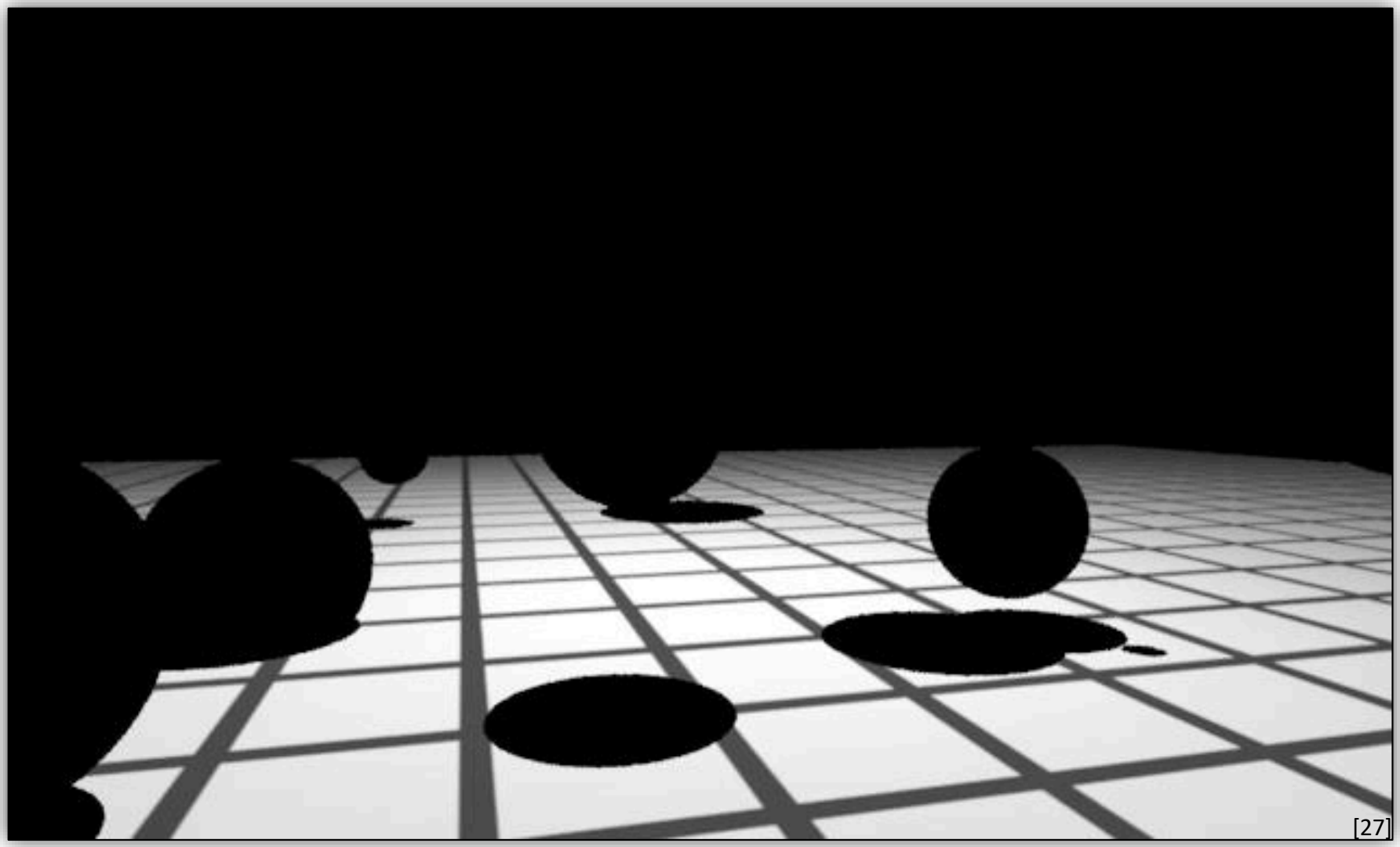
- vorgegebene maximale Anzahl von Reflexionen, oder
- Rekursion bis der Beitrag zur Farbe vernachlässigbar wird
  - Beitrag der 1. Reflexion: $k_{r,1}$
  - Beitrag der 2. Reflexion: $k_{r,1} \cdot k_{r,2}$
  - verfolge bis $\prod_i k_{r,i} < \epsilon_r$
  - Achtung: der tatsächliche Beitrag hängt auch von Intensität der LQ ab



O'Brien 2015 [28]

Glaskugeln – Rekursionstiefe 0

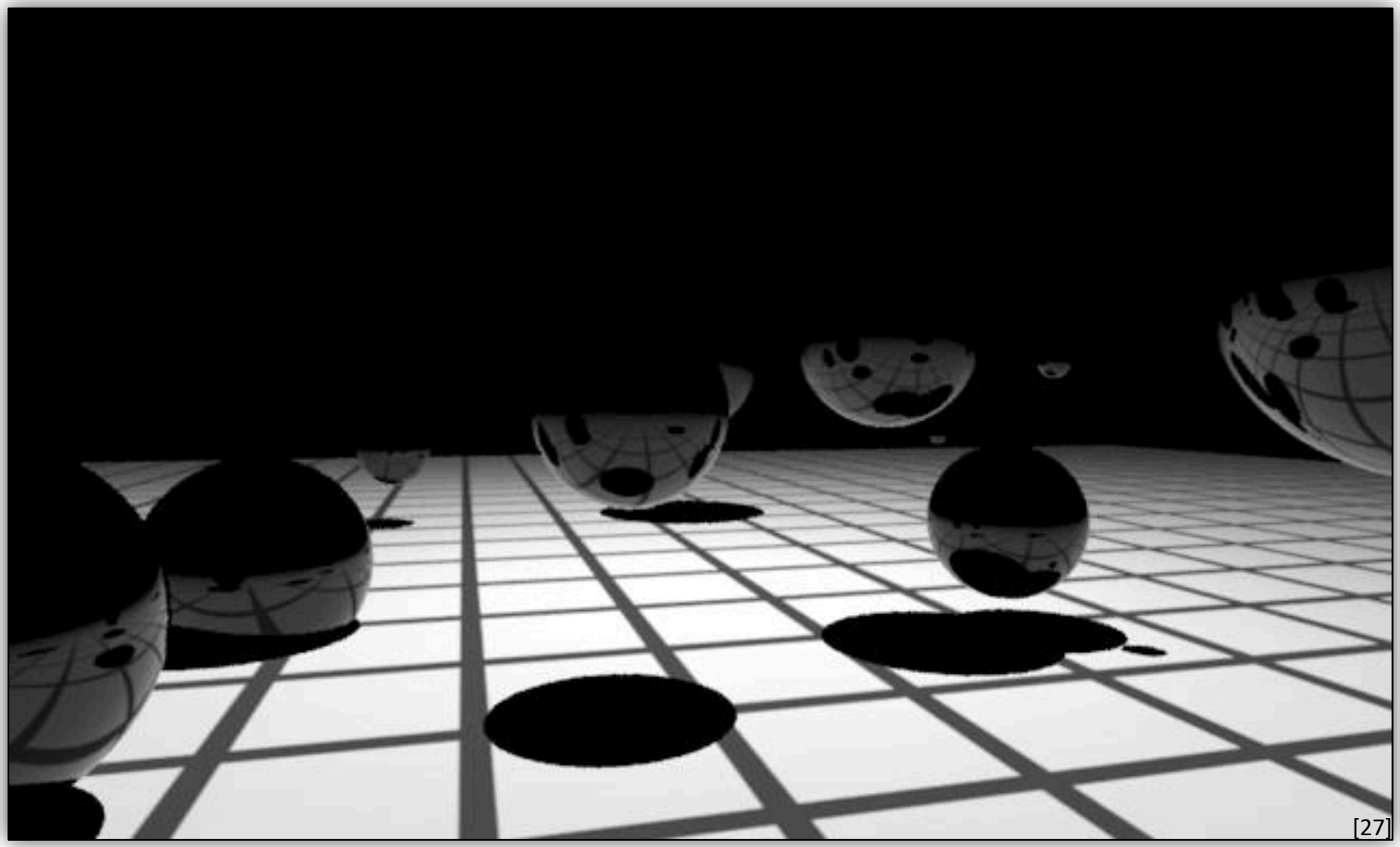
- Bild: Pat Hanrahan



Materialkoeffizienten der Glaskugeln: $k_a = k_d = k_s = 0$ und $k_r, k_t > 0$

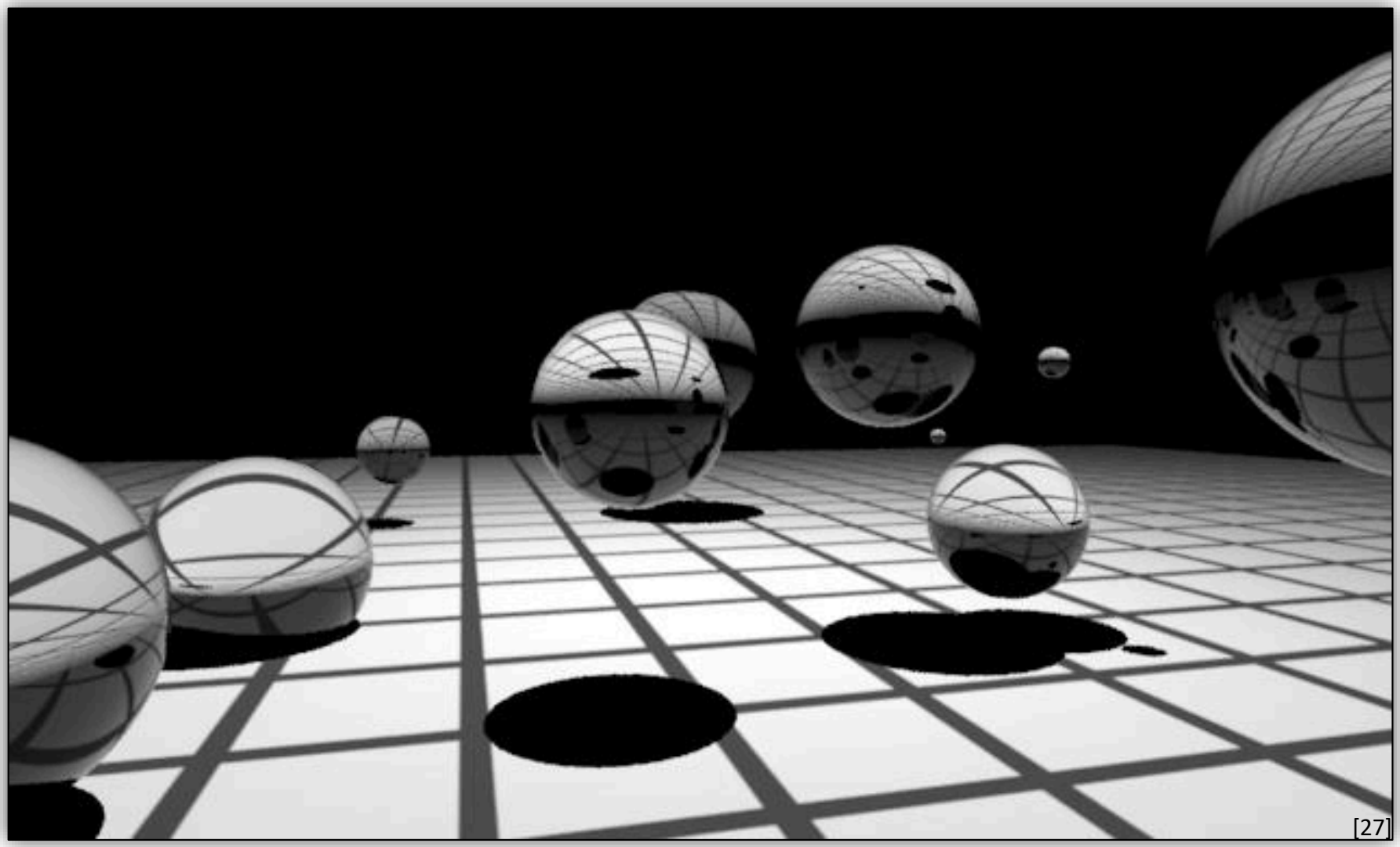
Glaskugeln – Rekursionstiefe 1

- Bild: Pat Hanrahan



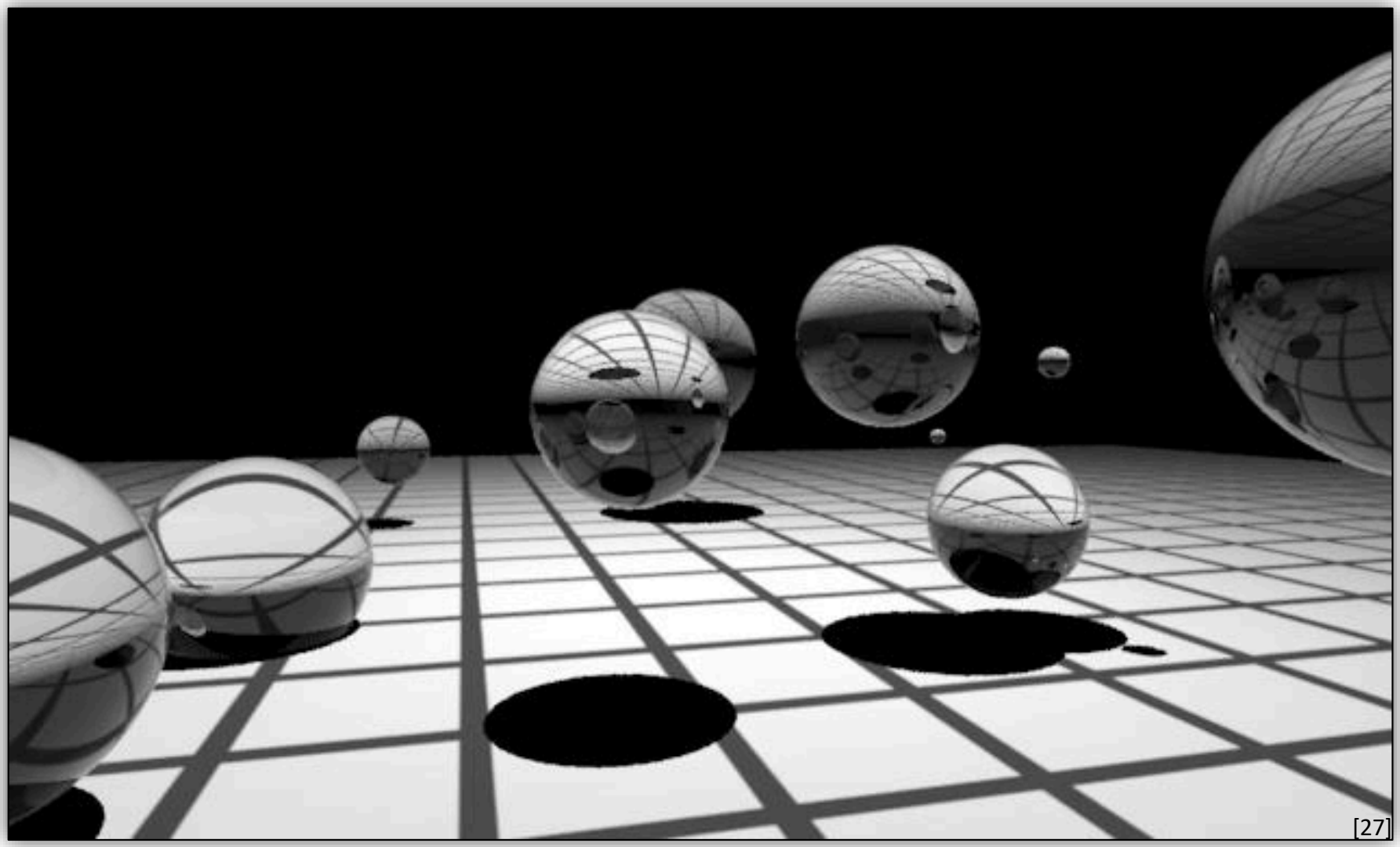
Glaskugeln – Rekursionstiefe 2

- Bild: Pat Hanrahan



Glaskugeln – Rekursionstiefe 5

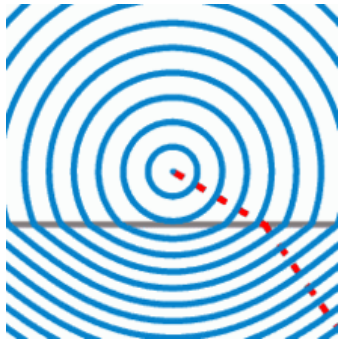
- Bild: Pat Hanrahan



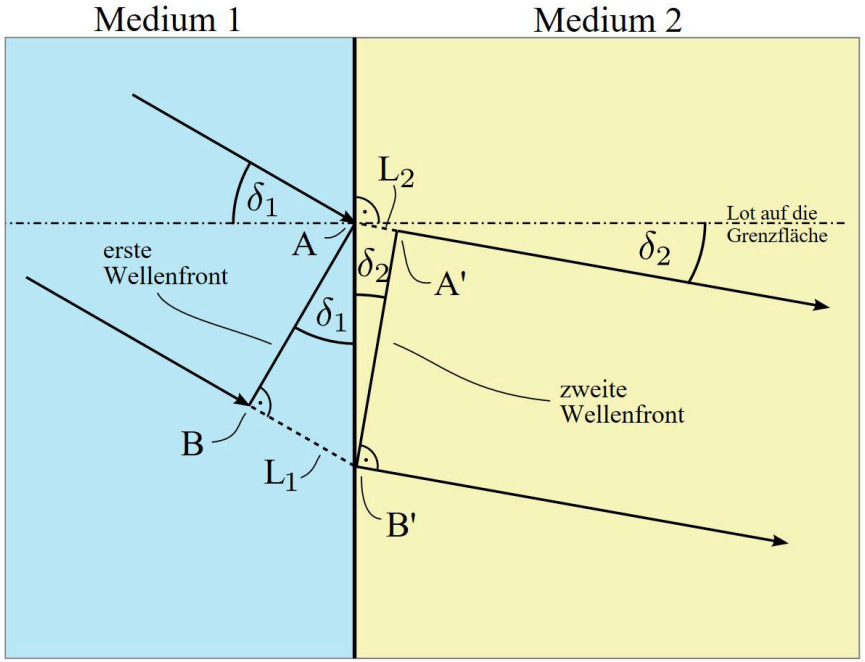
Snellsches Brechungsgesetz

- … beschreibt die Richtungsänderung einer Welle (Licht) beim Übergang von einem Medium in eines mit anderer Brechzahl

- Licht bewegt sich in unterschiedlichen Medien unterschiedlich schnell
  - (reelle) Brechzahl $\eta = {}^{c_0}/_{c_\eta}$
  - c_0, c_η Phasengeschwindigkeit im Vakuum/Medium
  - Ausbreitungsrichtung senkrecht zur Wellenfront



[30]



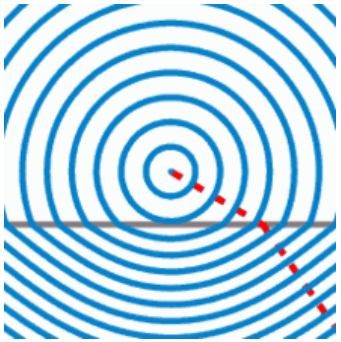
[29]

Bilder: Wikipedia u.a.

Snellsches Brechungsgesetz

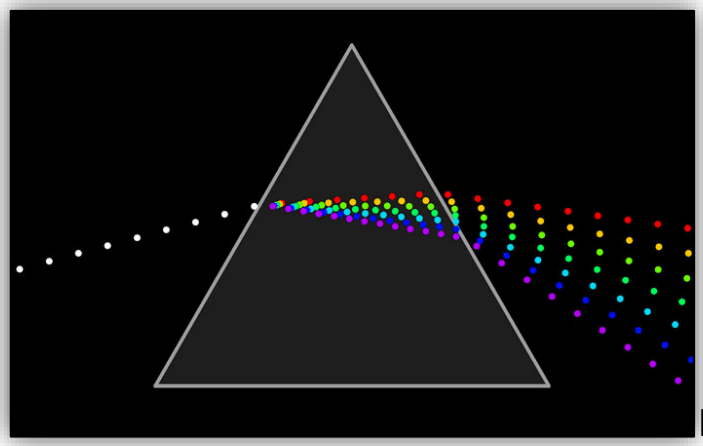
- … beschreibt die Richtungsänderung einer Welle (Licht) beim Übergang von einem Medium in eines mit anderer Brechzahl

- Licht bewegt sich in unterschiedlichen Medien unterschiedlich schnell
  - (reelle) Brechzahl $\eta = {}^{c_0}/_{c_\eta}$
  - c_0, c_η Phasengeschwindigkeit im Vakuum/Medium
  - Ausbreitungsrichtung senkrecht zur Wellenfront

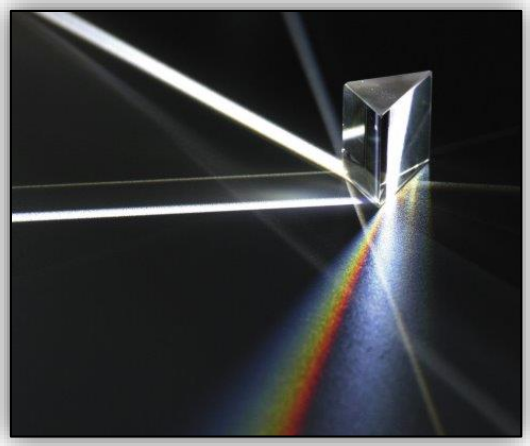


[30]

- Brechzahl ist i.d.R. wellenlängenabhängig $\eta(\lambda)$
  - wellenlängenabhängige Ausbreitungsgeschwindigkeit von Wellen (Dispersion) → Aufspaltung in unterschiedliche Wellenlängen



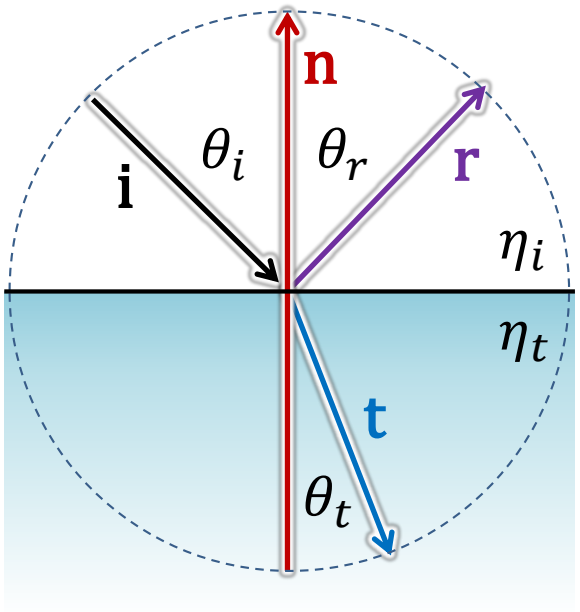
[31]



Bilder: Wikipedia u.a.

Snellsches Brechungsgesetz

- Brechungsgesetz: $\eta_i \sin\theta_i = \eta_t \sin\theta_t$
  - Brechung bei Übergang ins optisch dichtere Medium ($\eta_t > \eta_i$) zum Lot hin
  - beim Übergang ins optisch dünnere Medium vom Lot weg
  - Fresnel-Effekt: Verteilung der Strahldichte
- Grenzwinkel der Totalreflexion
  - Licht wird an der Grenzfläche von dichterem zu dünnerem Medium vollständig reflektiert: $\sin\theta_c = \eta_i/\eta_t$



- Transmissionsvektor $\mathbf{t}$ (Herleitung siehe Fund. of Computer Graphics)

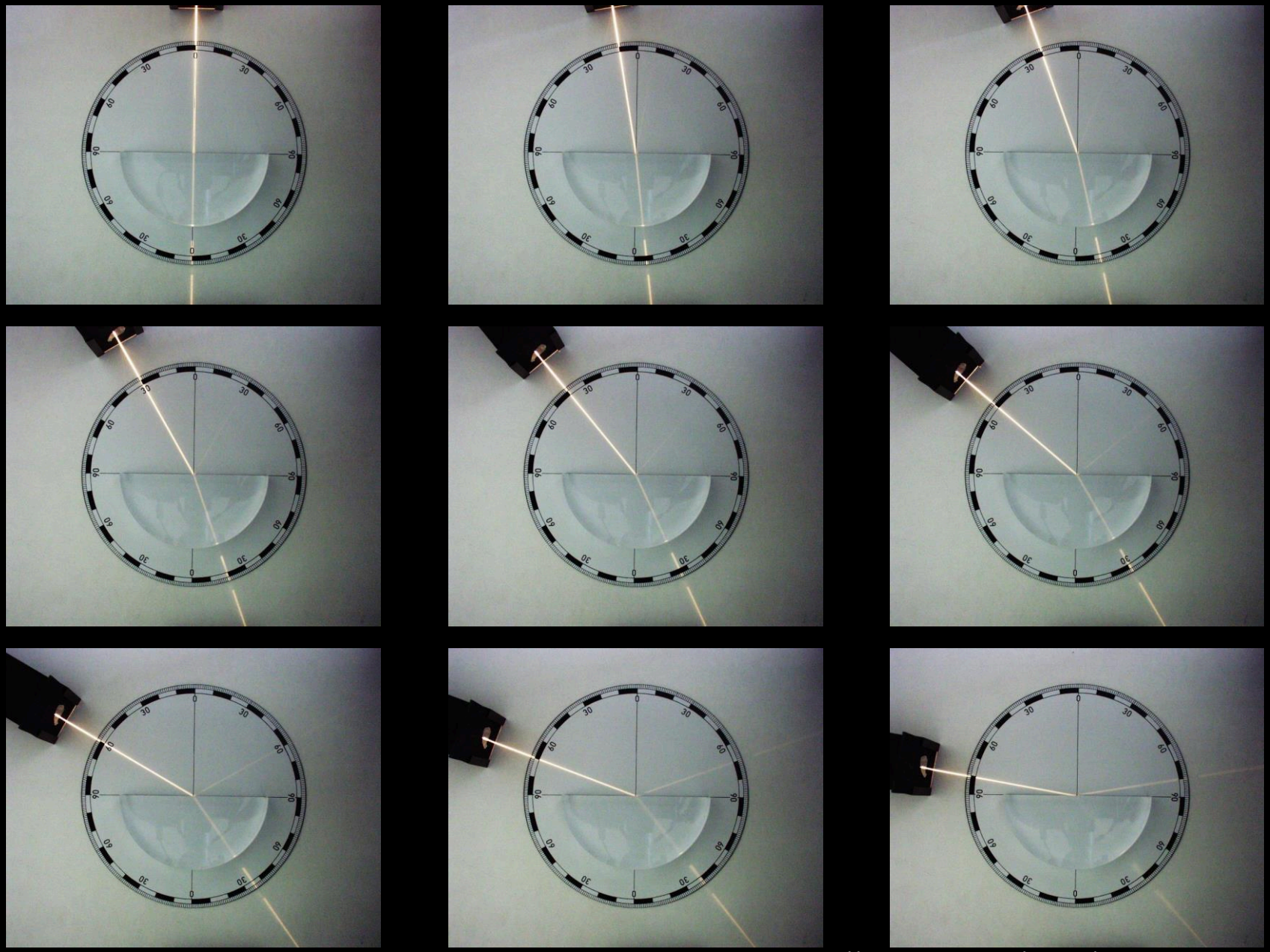
$$\mathbf{t} = -\frac{\sin\theta_t}{\sin\theta_i}(\mathbf{i} - \mathbf{n}\cos\theta_i) - \mathbf{n}\cos\theta_t$$

$$= -\frac{\eta_i}{\eta_t}\mathbf{i} + \left(\frac{\eta_i}{\eta_t}\cos\theta_i - \sqrt{1 - \left(\frac{\eta_i}{\eta_t}\right)^2 (1 - \cos^2\theta_i)}\right)\mathbf{n}$$

- $\mathbf{i}$, $\mathbf{n}$, $\mathbf{r}$ und $\mathbf{t}$ liegen in einer Ebene

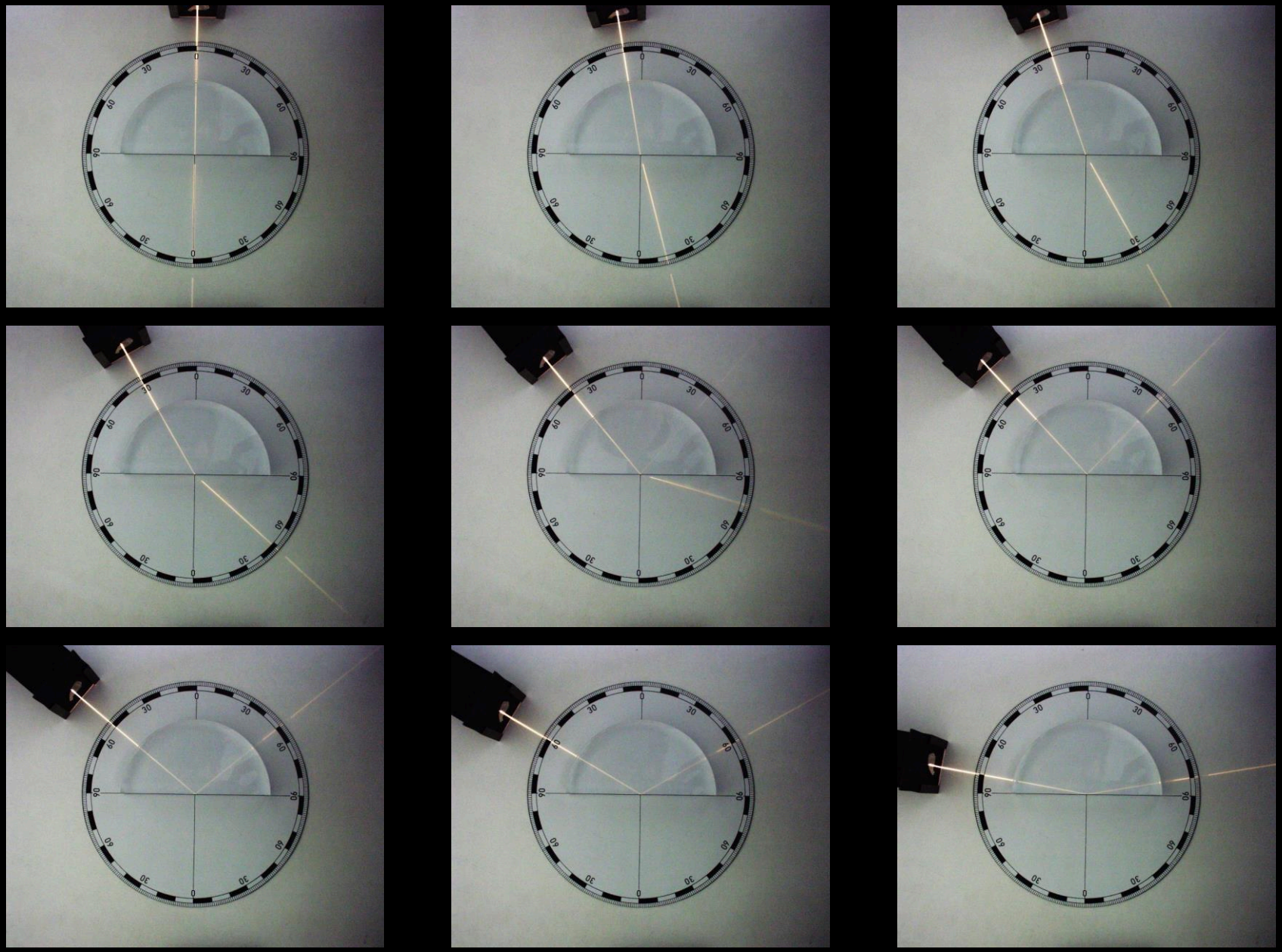
Sehr ausführliche Herleitung: http://graphics.stanford.edu/courses/cs148-10-summer/docs/2006--degreve--reflection_refraction.pdf

Fresnel-Effekt, Lichtbrechung und Reflexion



Bilder: http://www.fizkapu.hu/fizfoto/fizfoto6.html [32]

Totalreflexion



Bilder: http://www.fizkapu.hu/fizfoto/fizfoto6.html [32]

Totalreflexion und Fresneleffekt

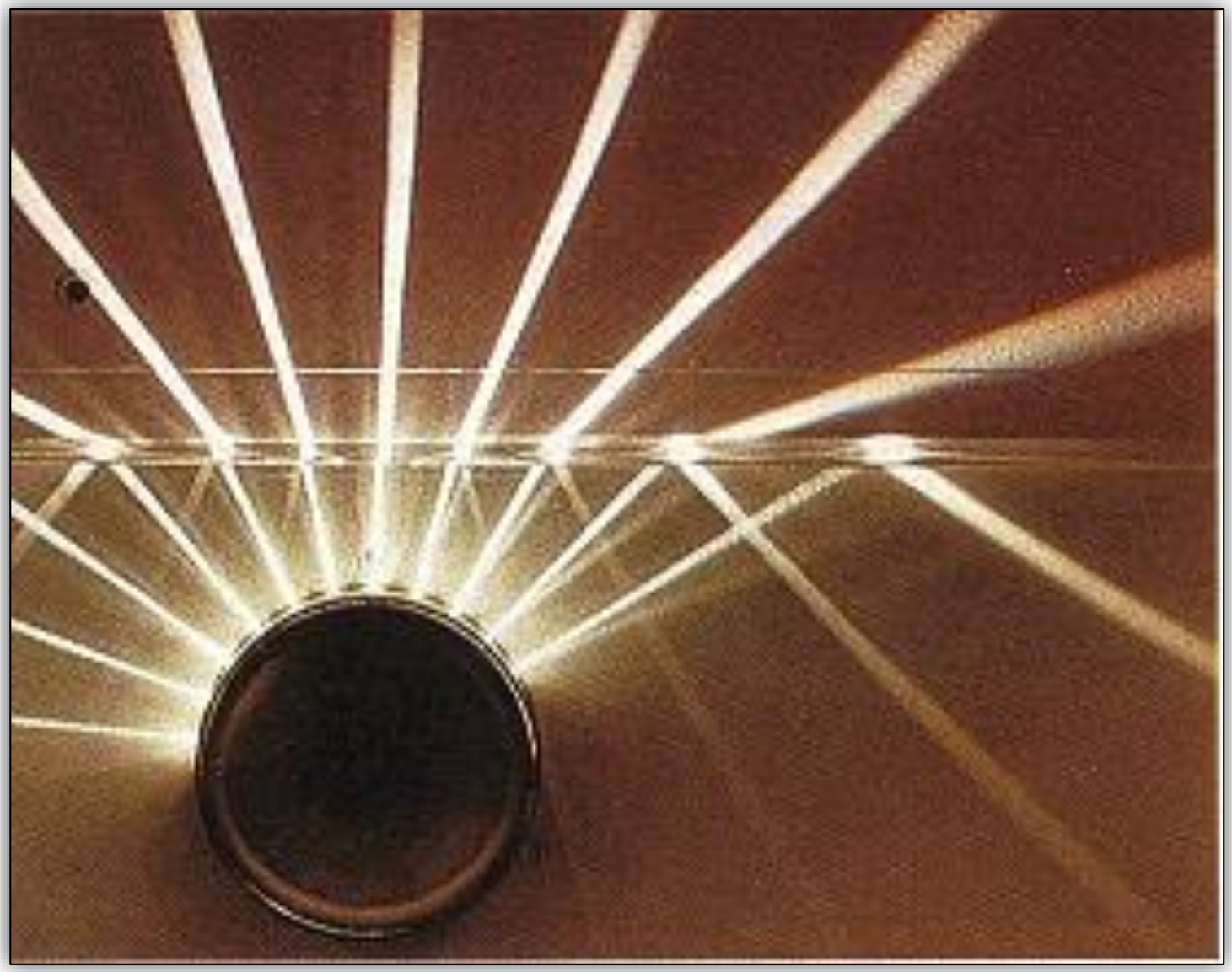


Bild: http://www.dieter-heidorn.de/Physik/VS/Optik/K04_Brechung/K04_Brechung.html [33]

Refraktion: Snell‘s Window, Optical Manhole

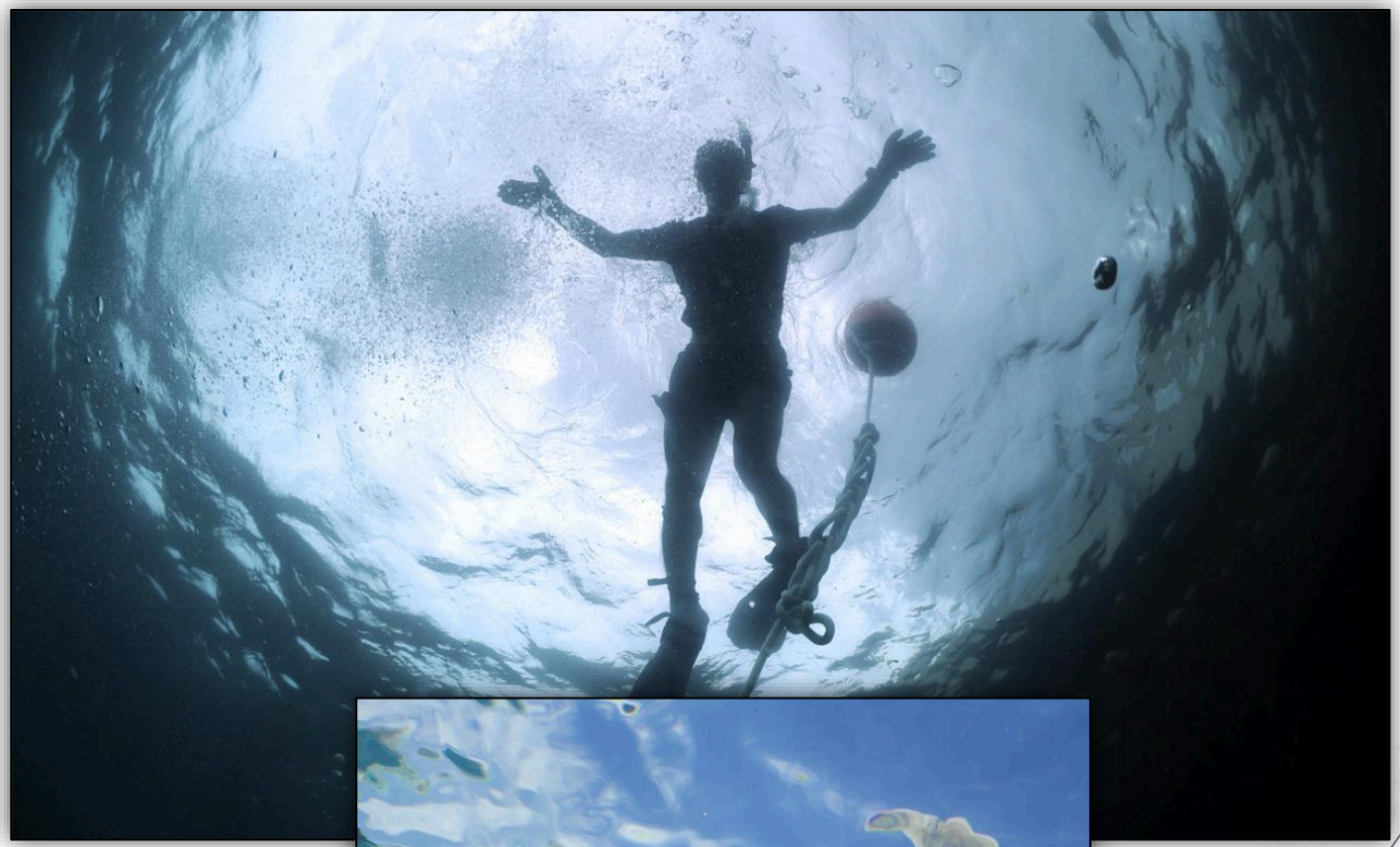


Bild: Wikipedia [34]

[34]

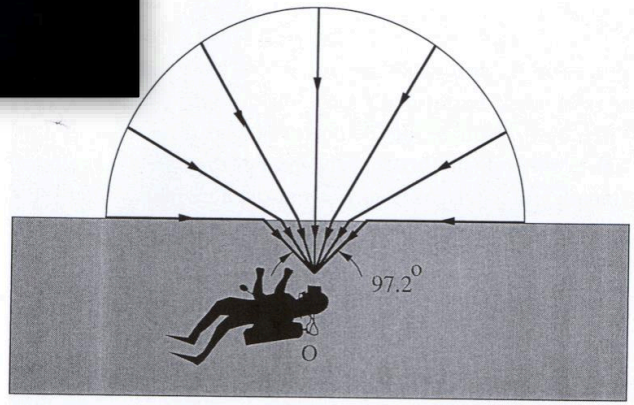


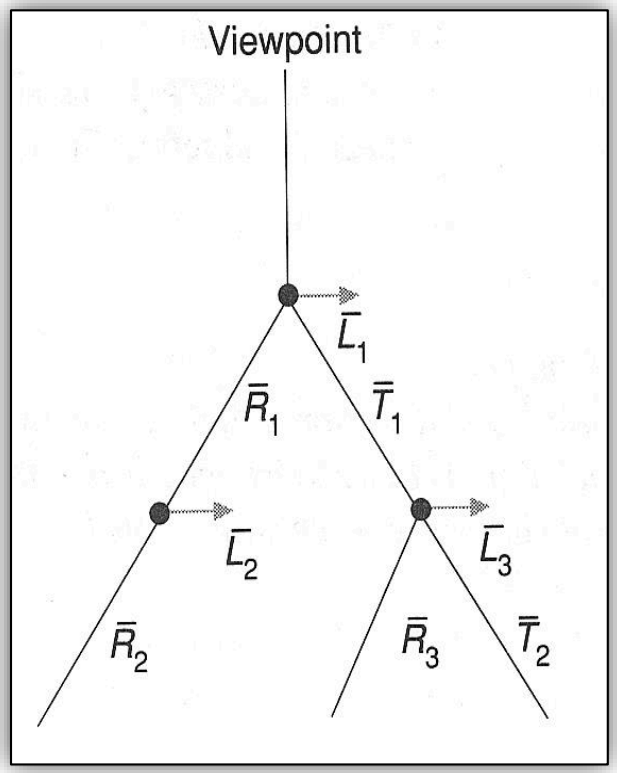
Fig. 3.7B The optical manhole. Light from the horizon (angle of incidence = 90°) is refracted downward at an angle of 48.6°. This compresses the sky into a circle with a diameter of 97.2° instead of its usual 180°.

aus “Color and Light in Nature”, Lynch and Livingstone[35]

Raytracing Pseudocode

```
vec3 raytrace( Ray *ray, ... ) {
  vec3 color = 0.0f;

  if ( !cast( ray, FLOAT_MAX ) )
    return color;

  for ( jede Lichtquelle )
    color += computeDirectLight( ... );

  if ( Fläche ist spiegelnd ) {
    // berechne Reflexionsstrahl
    Ray reflect = ...;
    color += i.kr * raytrace( reflect, ... );
  }

  if ( Fläche ist (semi-)transparent ) {
    // berechne Transmissionsstrahl
    Ray refract = ...;
    if ( Totalreflexion )
      color += i.kr * raytrace( reflect, ... );
    else
      color += i.kt * raytrace( refract, ... );
  }

  return color;
}
```

Diese rekursive Formulierung heißt Whitted-Style Raytracing (nach Turner Whitted)

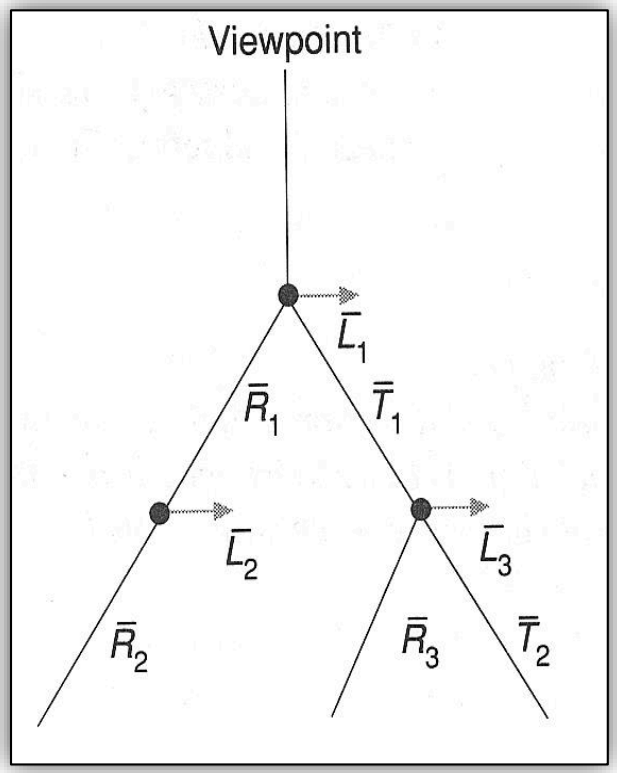


Raytracing Pseudocode

```
vec3 raytrace( Ray *ray, ... ) {
  vec3 color = 0.0f;

  if ( !cast( ray, FLOAT_MAX ) )
    return color;

  for ( jede Lichtquelle )
    color += computeDirectLight( ... );

  if ( Fläche ist spiegelnd ) {
    // berechne Reflexionsstrahl
    Ray reflect = ...;
    color += i.kr * raytrace( reflect, ... );
  }

  if ( Fläche ist (semi-)transparent ) {
    // berechne Transmissionsstrahl
    Ray refract = ...;
                                              );
                                              );
  }
  return color;
}
```

Um die Richtung zu berechnen, muss der Raytracer mitführen, in welchem Medium sich der Strahl aktuell befindet (z.B. Umgebung, Glaskugel, …)

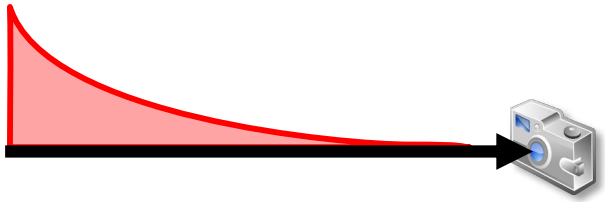
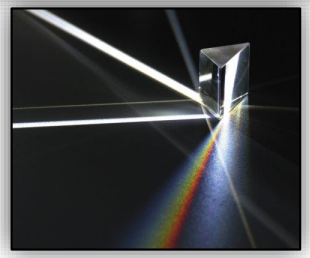
Diese rekursive Formulierung heißt Whitted-Style Raytracing (nach Turner Whitted)



Transmission

- transparentes Material mit keiner bzw. konstanter Abschwächung (links)
- exponentieller Abfall durch Absorption $e^{-\sigma_t \cdot s}$ (Beer‘s Law, Mitte)
- zus. Dispersion mit einem Transmissionsstrahl pro Wellenlänge (rechts)

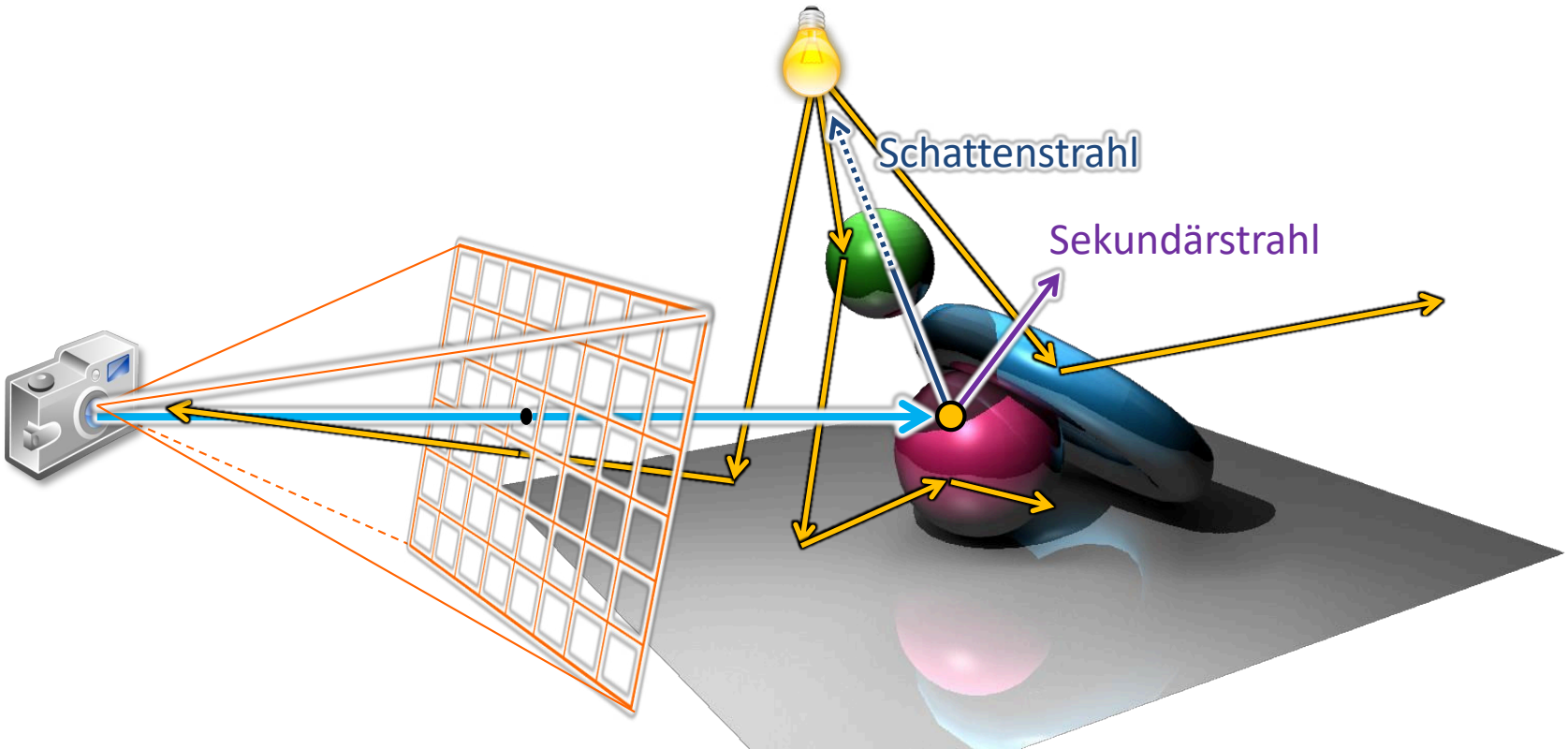


Whitted-style Raytracing

Zusammenfassung

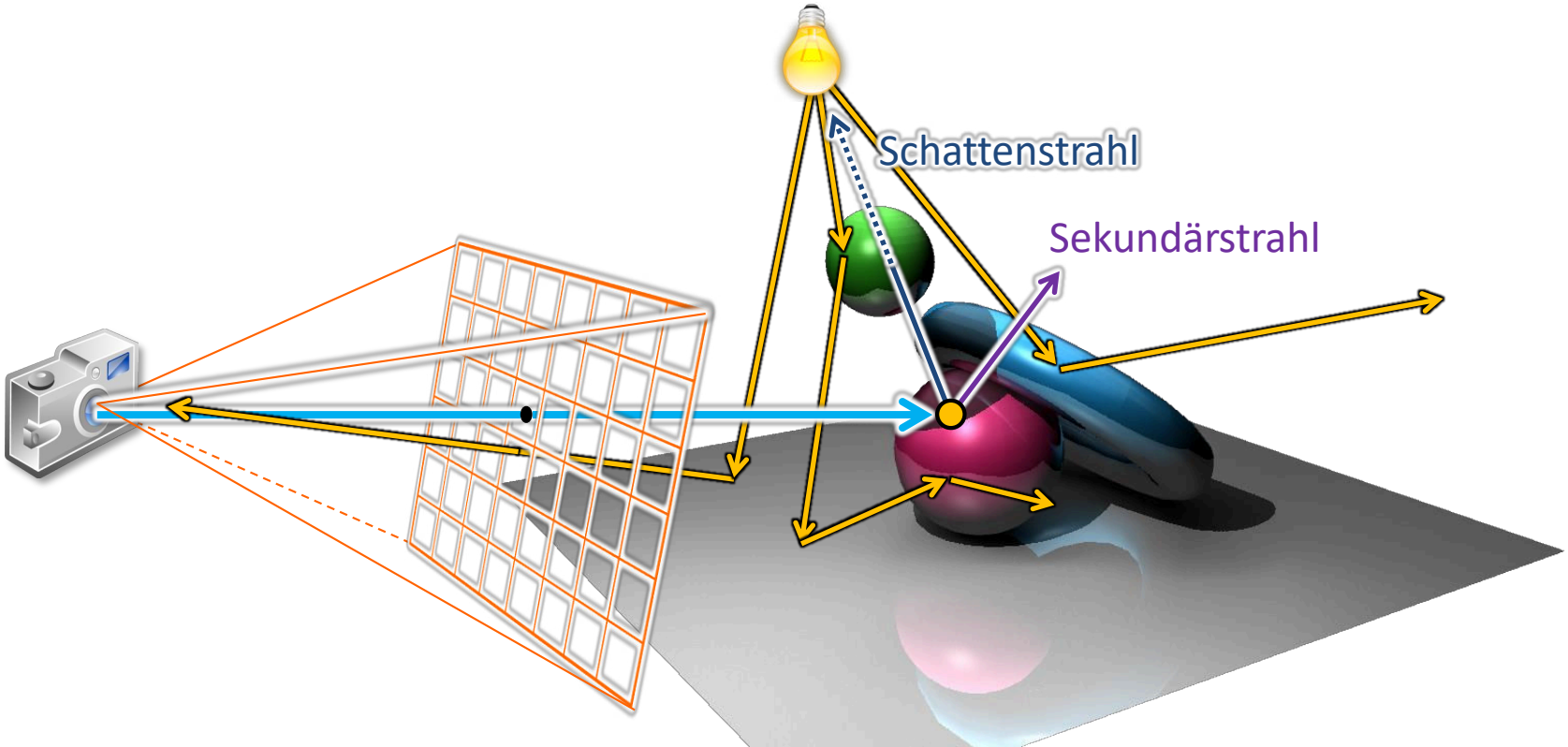
- Raytracing ist ein Verfahren zur Bildsynthese (engl. Rendering)
- geometrische Optik: Licht breitet sich entlang von Strahlen aus
- die Ausbreitung des Lichts wird – von der Kamera aus – zurückverfolgt



Whitted-style Raytracing

Zusammenfassung: Schritte

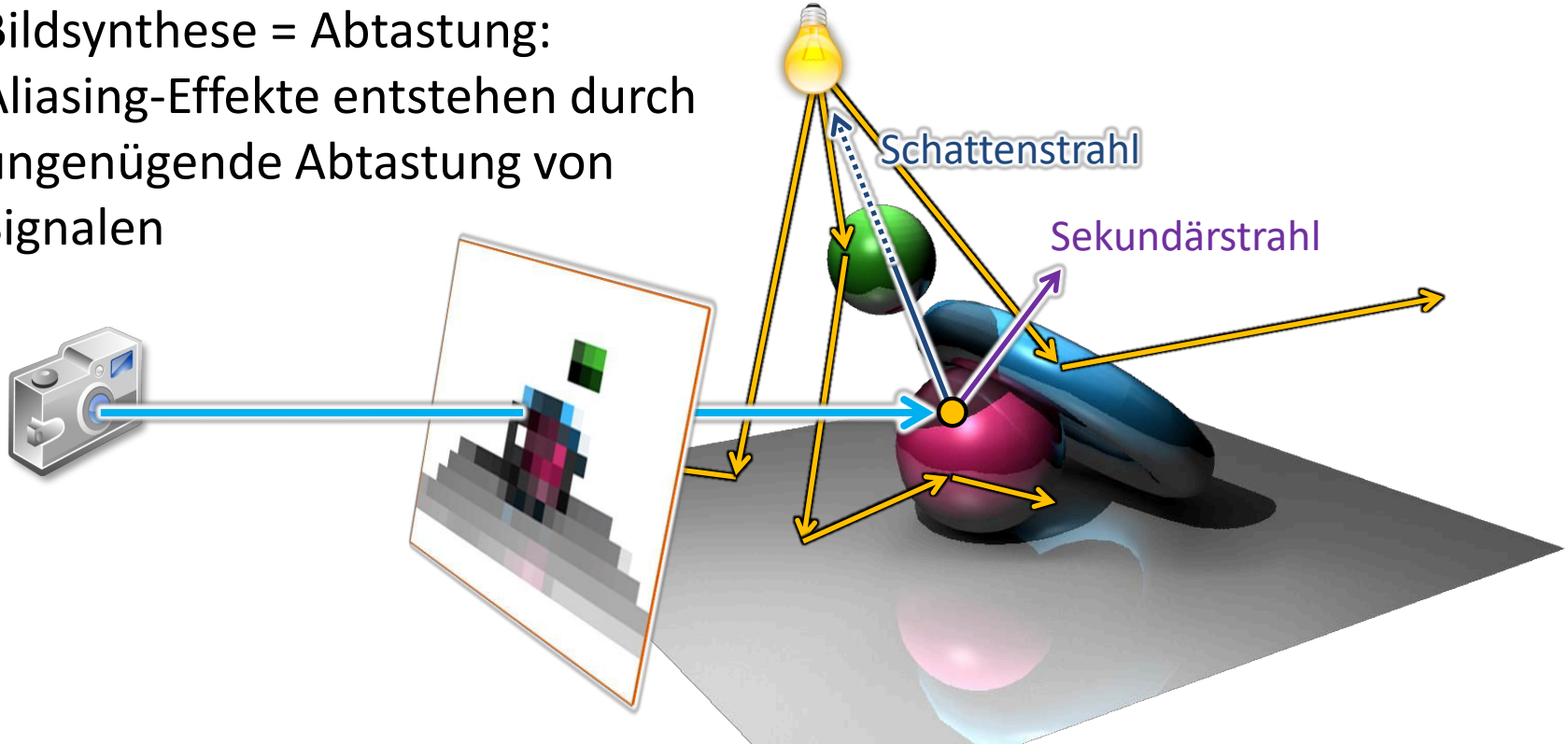
- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting)
- Schattierung und Beleuchtungsberechnung (shading):
  lokale Beleuchtungsberechnung und **globale** Schattenstrahlen
- rekursive Sekundärstrahlen für Spiegelung und Transmission



Raytracing und Aliasing

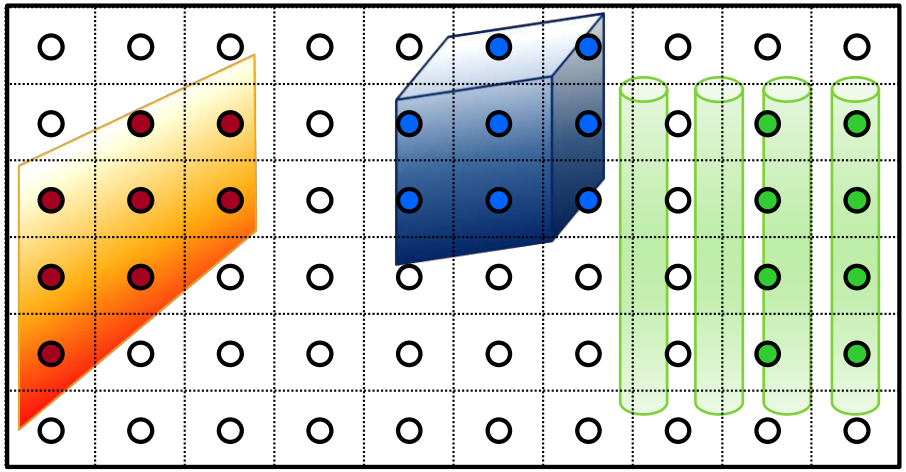
Zusammenfassung: Schritte

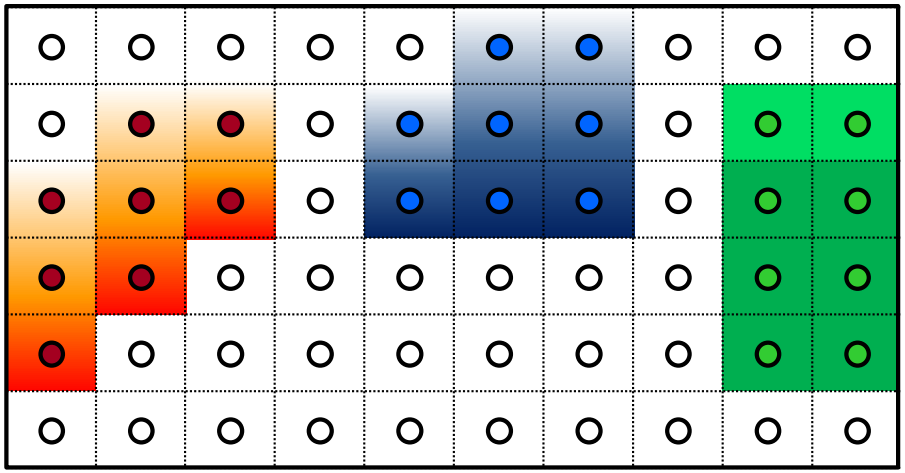
- Erzeugung der Sichtstrahlen durch jeden Pixel (ray generation)
- Schnittberechnung (ray casting)
- Schattierung und Beleuchtungsberechnung (shading):
  lokale Beleuchtungsberechnung und **globale** Schattenstrahlen
- rekursive Sekundärstrahlen für Spiegelung und Transmission
- Bildsynthese = Abtastung:
  Aliasing-Effekte entstehen durch ungenügende Abtastung von Signalen



Raytracing und Aliasing

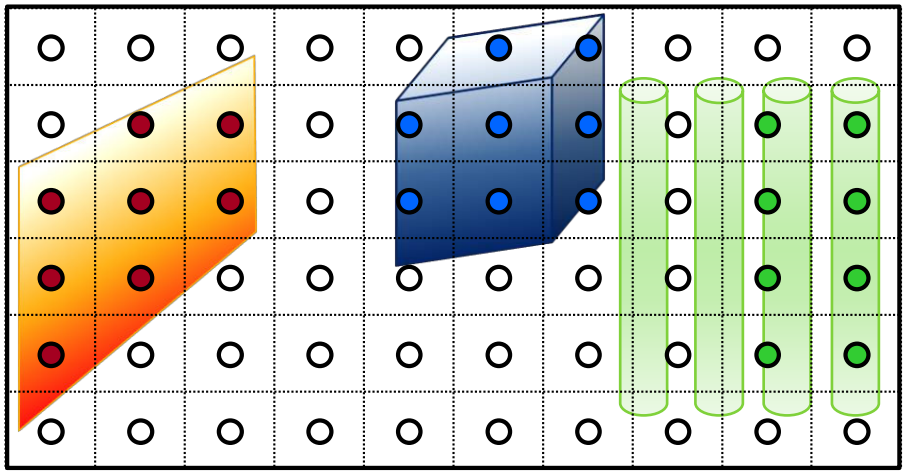
- das Bild einer 3D-Szene ist ein zweidimensionales Signal
  - äquidistante, diskrete Abtastung eines Bildsignals
  - Farbe eines Pixels bestimmt durch Strahl durch seinen Mittelpunkt, d.h. an den Pixelmittelpunkten tasten wir das Bildsignal ab (Vorstellung: stückweise konstante Funktion)
  - das Bildsignal ist nicht bandbegrenzt (gilt insb. für die Objektkanten)
  - durch quadratische Form der Pixel sehen wir zusätzlich Treppenstufen („Jaggies“) bei der Rekonstruktion des Signals

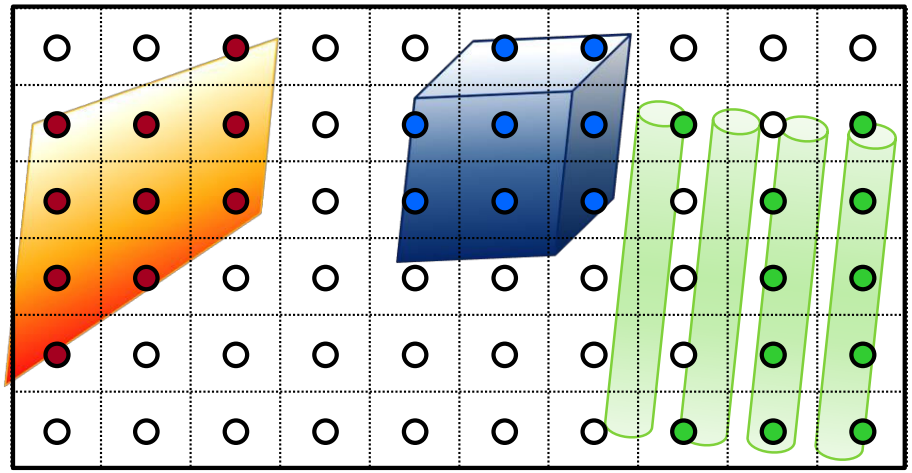




Raytracing und Aliasing

- das Bild einer 3D-Szene ist ein zweidimensionales Signal
  - äquidistante, diskrete Abtastung eines Bildsignals
  - Farbe eines Pixels bestimmt durch Strahl durch seinen Mittelpunkt, d.h. an den Pixelmittelpunkten tasten wir das Bildsignal ab (Vorstellung: stückweise konstante Funktion)
  - das Bildsignal ist nicht bandbegrenzt (gilt insb. für die Objektkanten)
  - durch quadratische Form der Pixel sehen wir zusätzlich Treppenstufen („Jaggies“) bei der Rekonstruktion des Signals

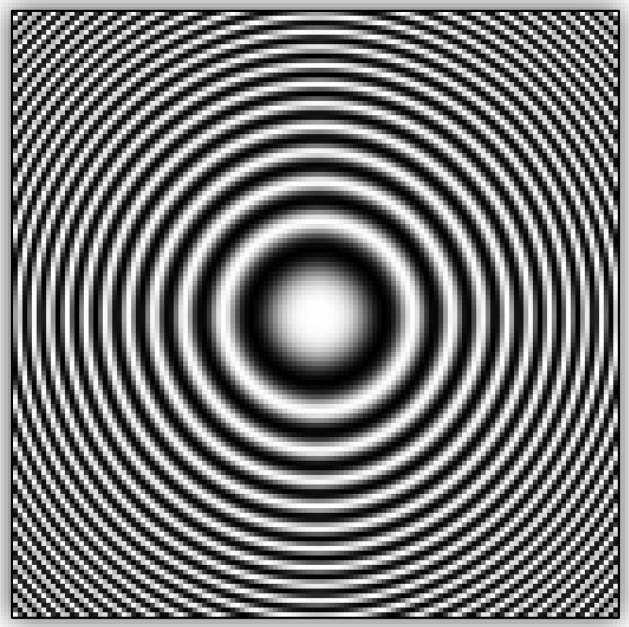




Bild, Signal und Abtastung

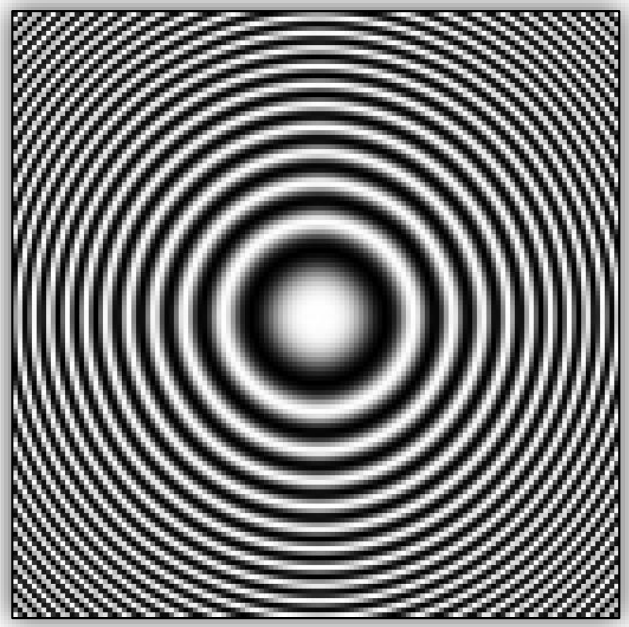
- Graustufenbild einer Funktion $f(d) = \cos ad^2$ mit $d \in [-1,1]^2$, $a \in \mathbb{R}$

```
#define WIDTH  128
#define HEIGHT 128

unsigned char img[ WIDTH * HEIGHT ];

for ( y = 0; y < HEIGHT; y++ ) {
  for ( x = 0; x < WIDTH; x++ ) {
    float dx = (float)( 2*x - WIDTH  ) / (float)WIDTH;
    float dy = (float)( 2*y - HEIGHT ) / (float)HEIGHT;
    float d2 = dx * dx + dy * dy;

    // f(d) mit a = 64
    float f = cosf( 64.0f * d2 );

    // abbilden von f(d) auf [0..255]
    img[ x + y * WIDTH ] =
        ( f + 1.0f ) * 127.5f;
}  }
```

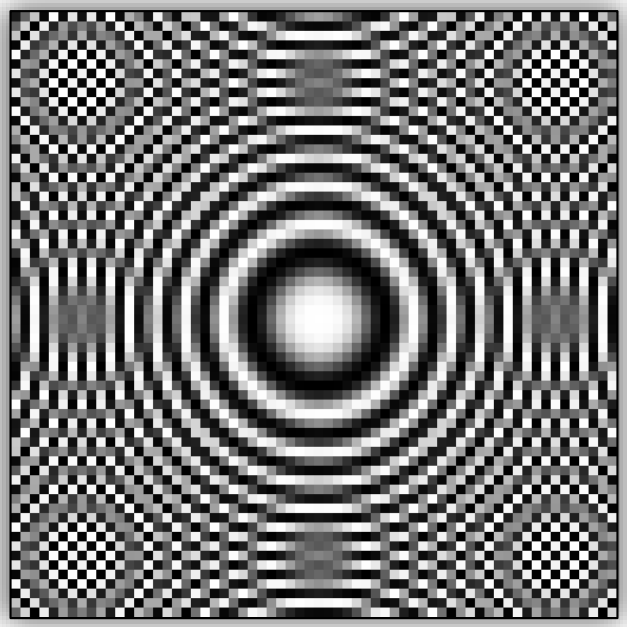


Bild, Signal und Abtastung

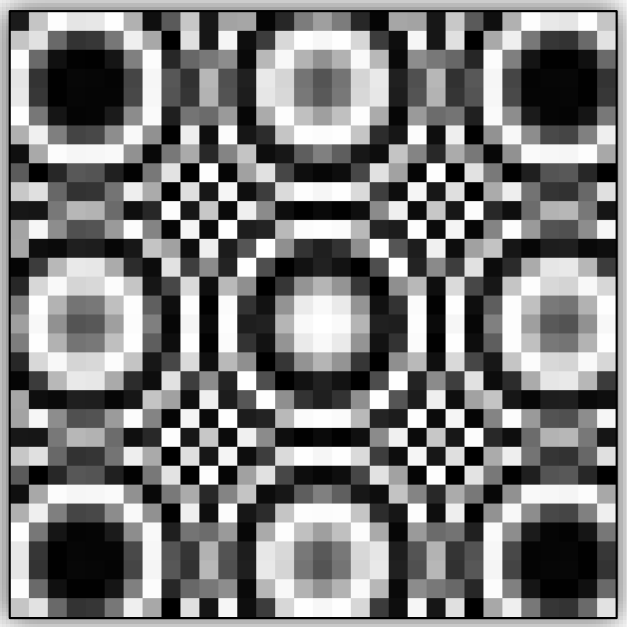
- Graustufenbild einer Funktion $f(d) = \cos ad^2$ mit $d \in [-1,1]^2$, $a \in \mathbb{R}$
- Aliasing-Effekte entstehen durch ungenügende Abtastung von Signalen



WIDTH=HEIGHT=128

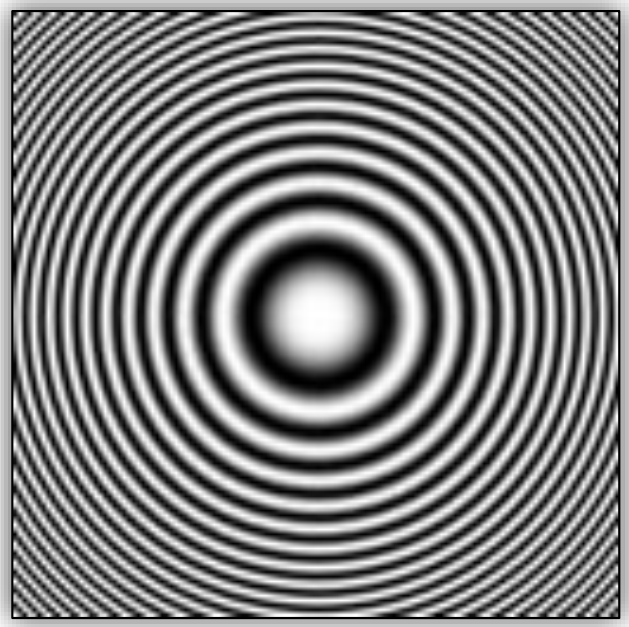


WIDTH=HEIGHT=64

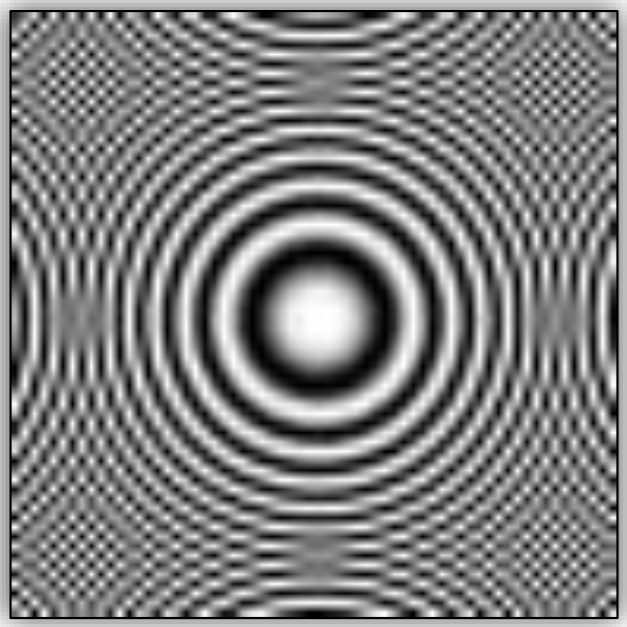


WIDTH=HEIGHT=32

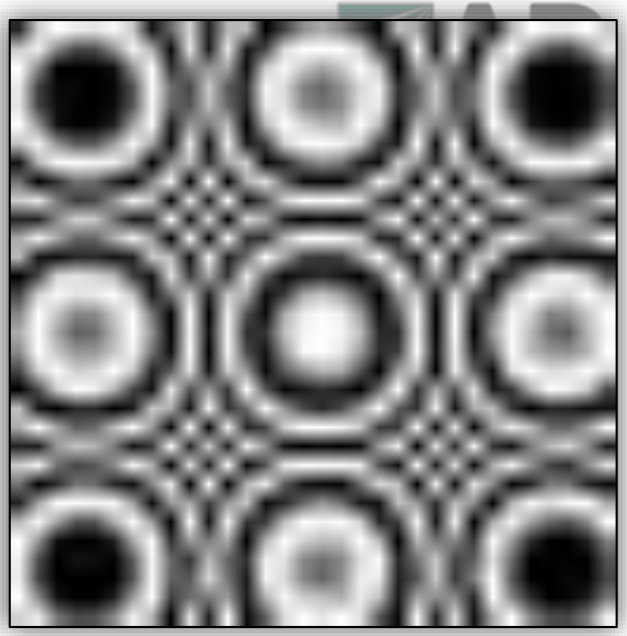
Unterschiedliche Rekonstruktionsfilter



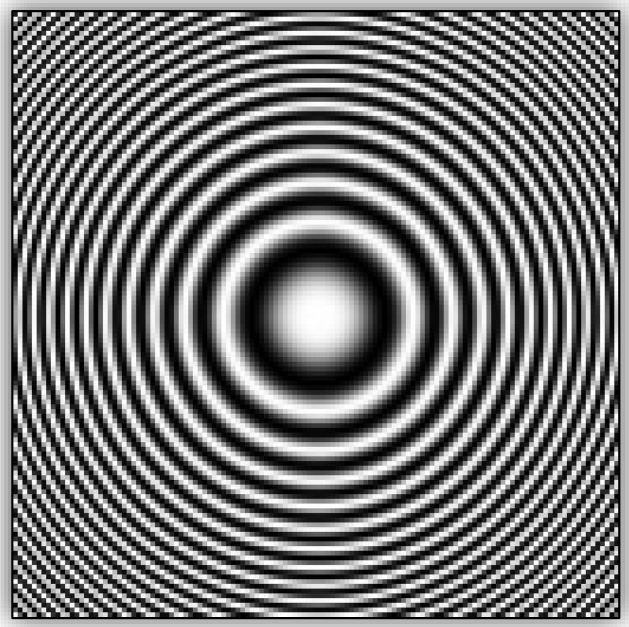
WIDTH=HEIGHT=128

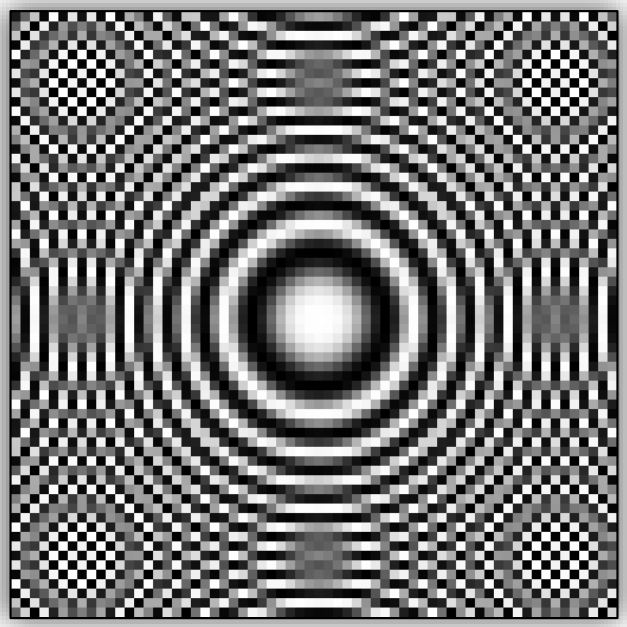


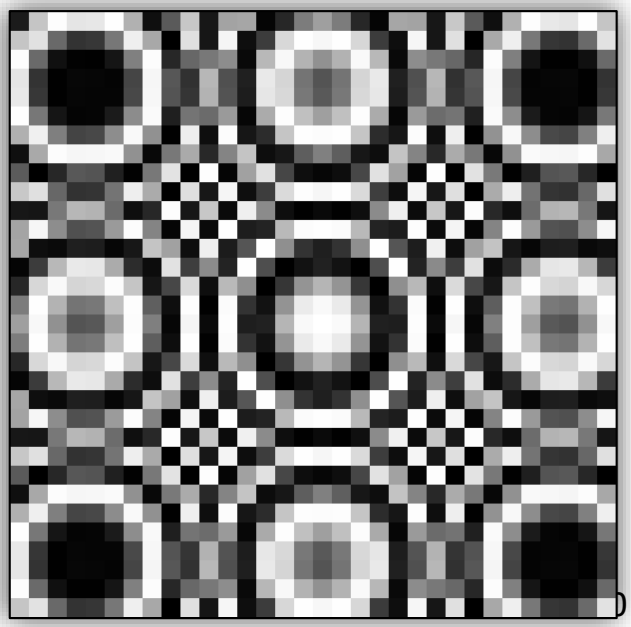
WIDTH=HEIGHT=64



WIDTH=HEIGHT=32







Bild, Signal und Abtastung

- Nyquist-Shannon-Abtasttheorem:
  ein kontinuierliches, bandbegrenztes Signal mit einer max. Frequenz f_{max} muss mit einer Frequenz größer als $2f_{max}$ abgetastet werden, damit aus dem diskreten Signal das Ursprungssignal exakt rekonstruiert werden kann
- Beispiel: $f_{abtast} > 2f_{max}$ (letzte Zeile: Grenzfall $f_{abtast} = 2f_{max}$)

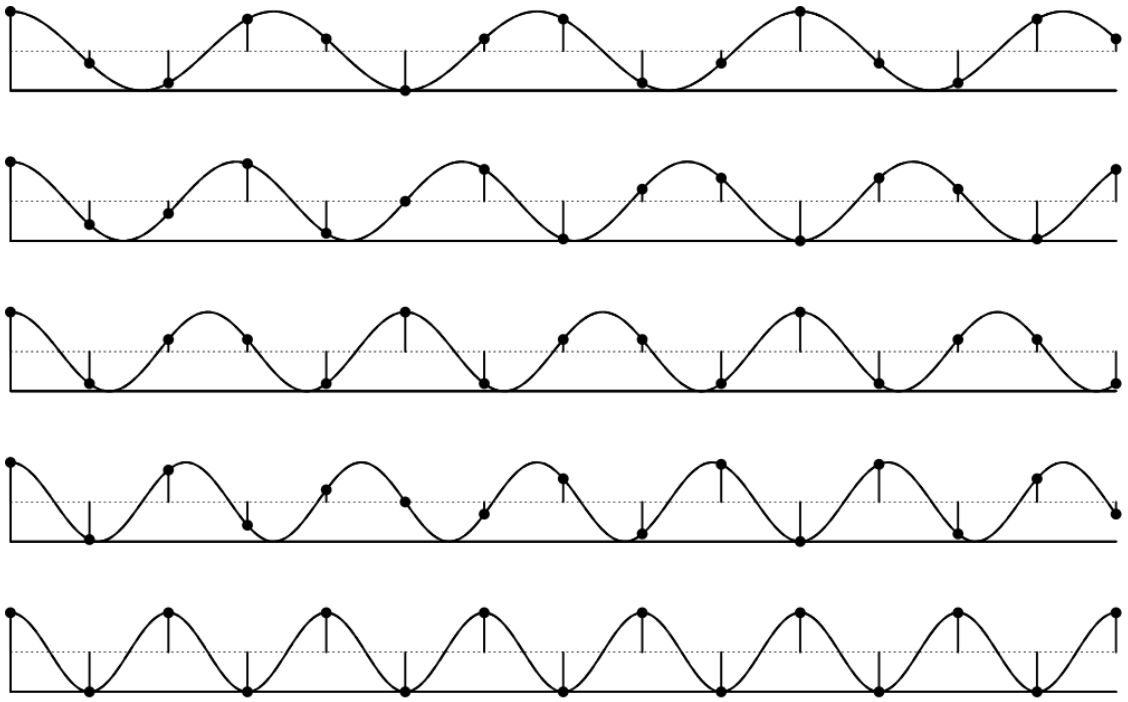


Bild: https://de.wikipedia.org/wiki/Nyquist-Shannon-Abtasttheorem

Bild, Signal und Abtastung

- Nyquist-Shannon-Abtasttheorem:
  ein kontinuierliches, bandbegrenztes Signal mit einer max. Frequenz f_{max} muss mit einer Frequenz größer als $2f_{max}$ abgetastet werden, damit aus dem diskreten Signal das Ursprungssignal exakt rekonstruiert werden kann
- Beispiel: $f_{abtast} < 2f_{max}$ —— orig. Signal ------ rekonstr. Signal

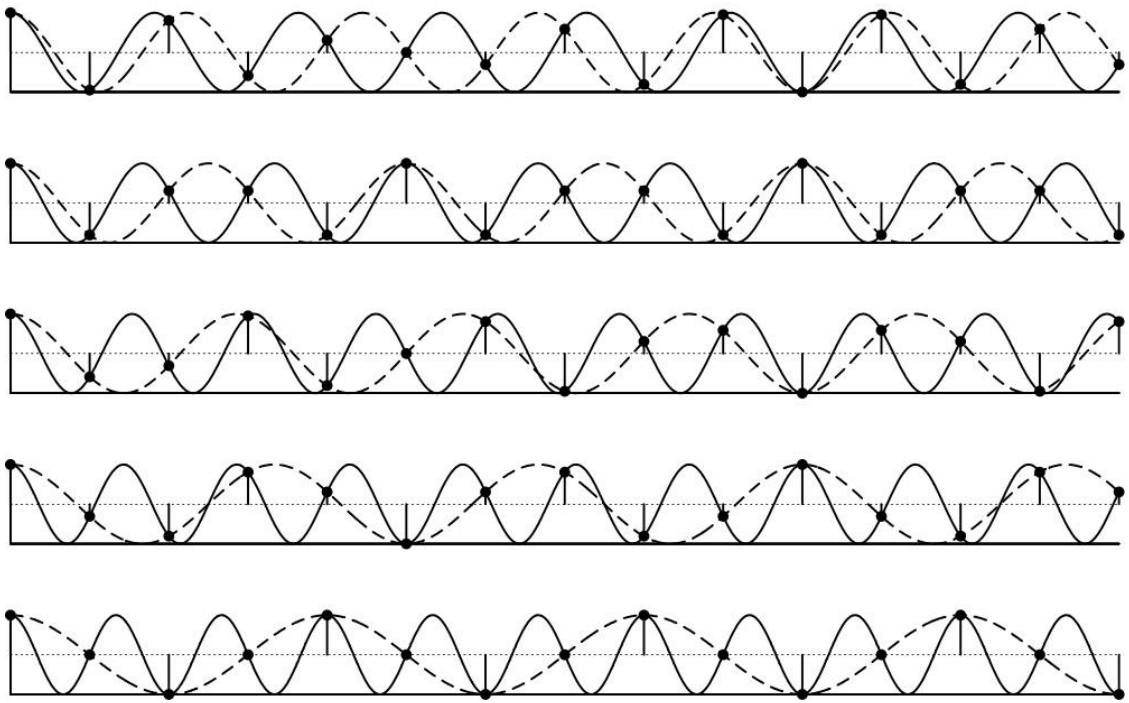
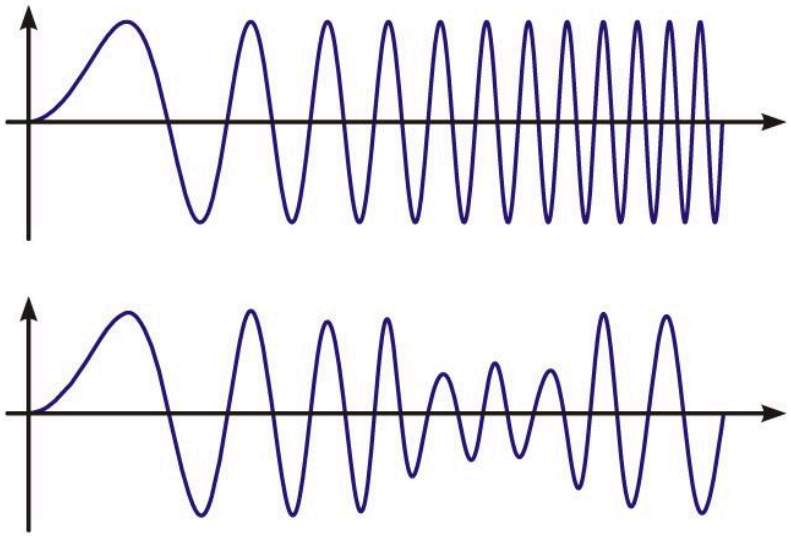


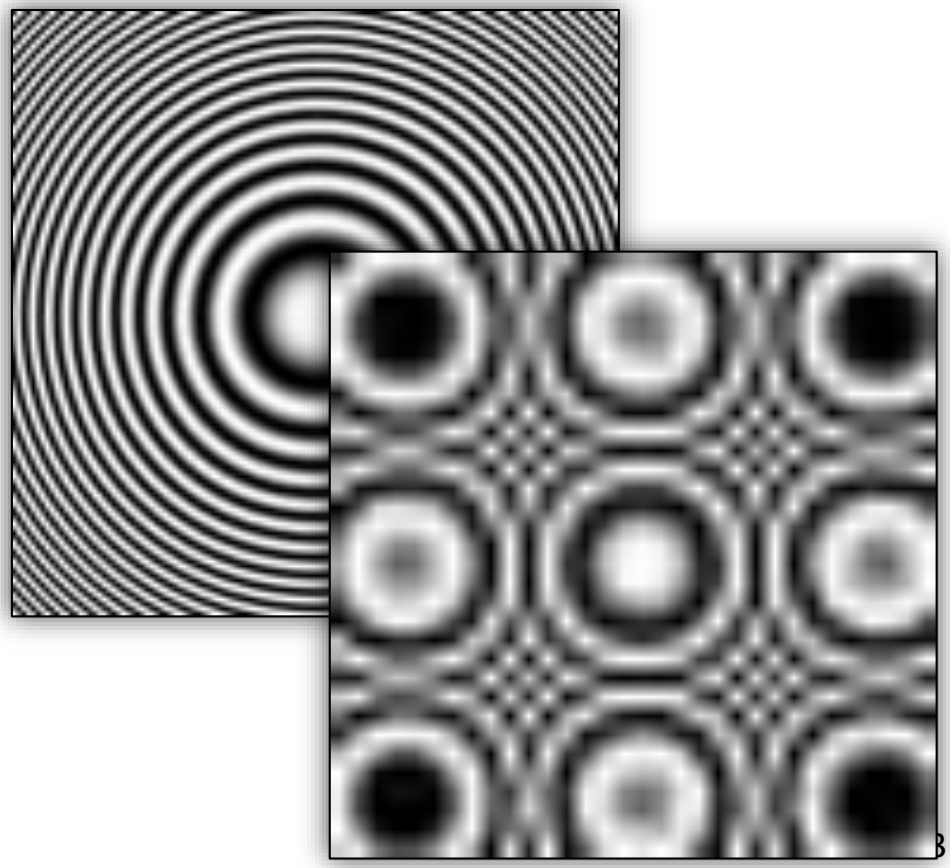
Bild: https://de.wikipedia.org/wiki/Nyquist-Shannon-Abtasttheorem

Bild, Signal und Abtastung

- Aliasing-Effekt: Fehler durch ungenügendes Abtasten von Signalen
  - im rekonstruierten Signal (z.B. Darstellung des Bildes am Monitor) treten Frequenzen auf, die im Original nicht enthalten sind

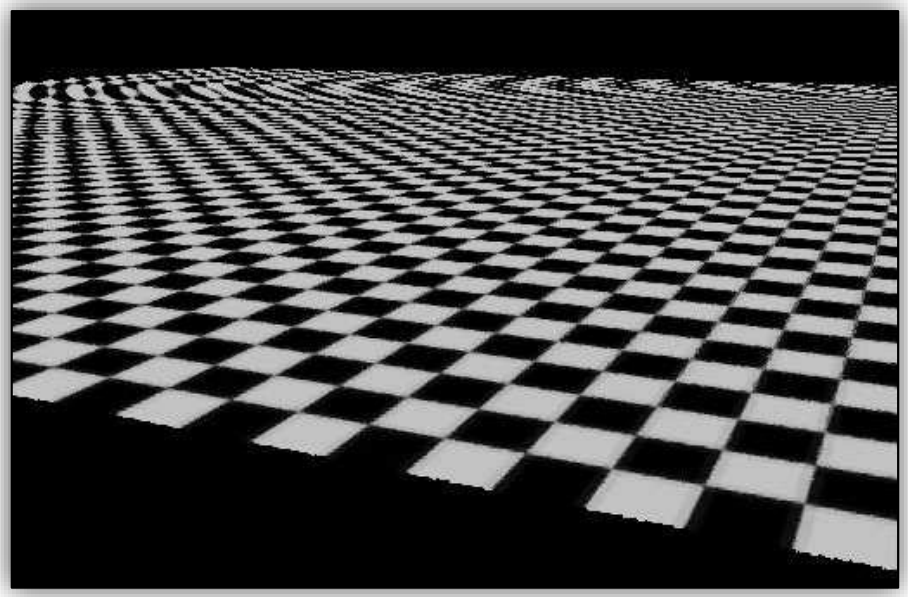
- 1D- und 2D-Version unseres Beispiels: Original und Rekonstruktion

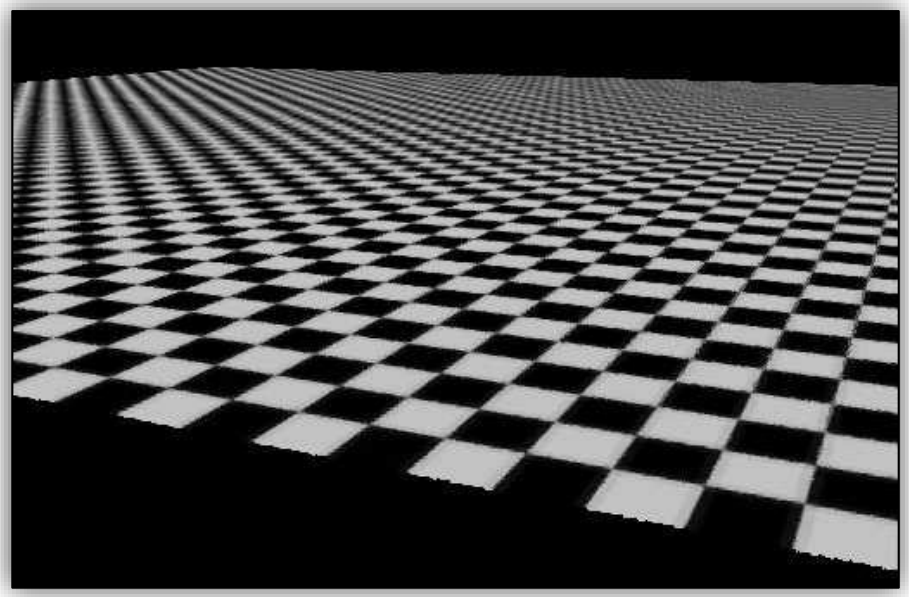




Bsp. Aliasing

- …das klassische Beispiel für Aliasing in der Computergrafik

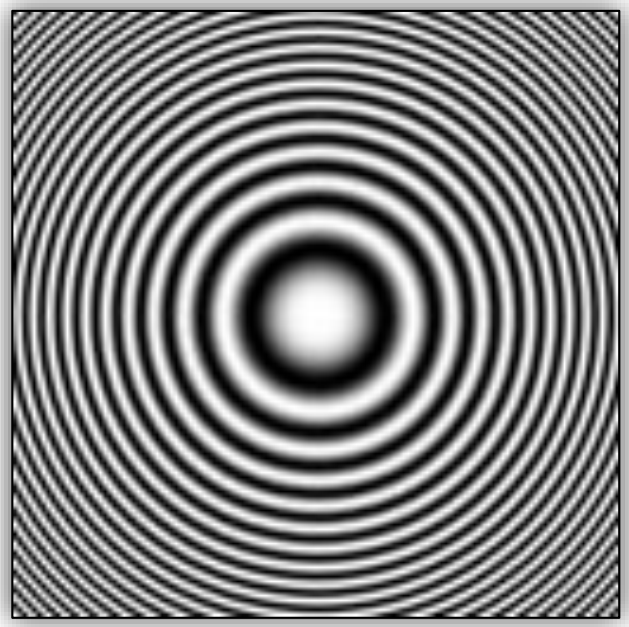




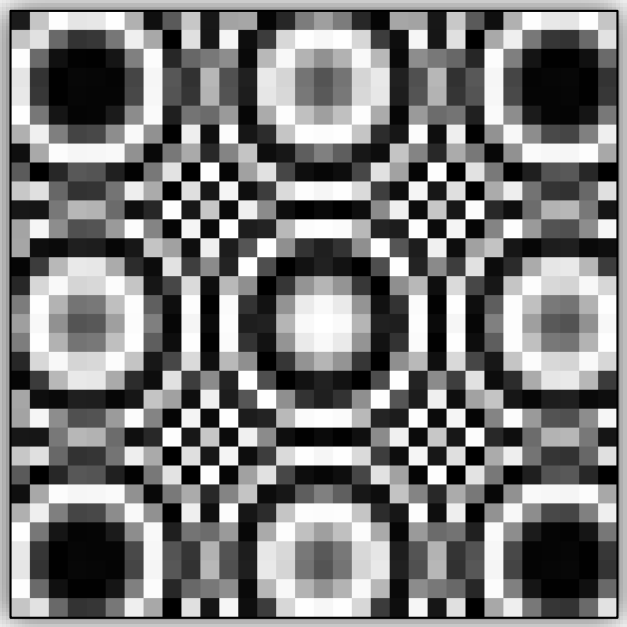
- typische Quellen von Aliasing
  - detaillierte Geometrie
  - Texturen/Textursignale (Farben, Normalen, …)
  - Shading (z.B. Glanzlichter auf gekrümmten Flächen)
  - …

Bild, Signal und Abtastung

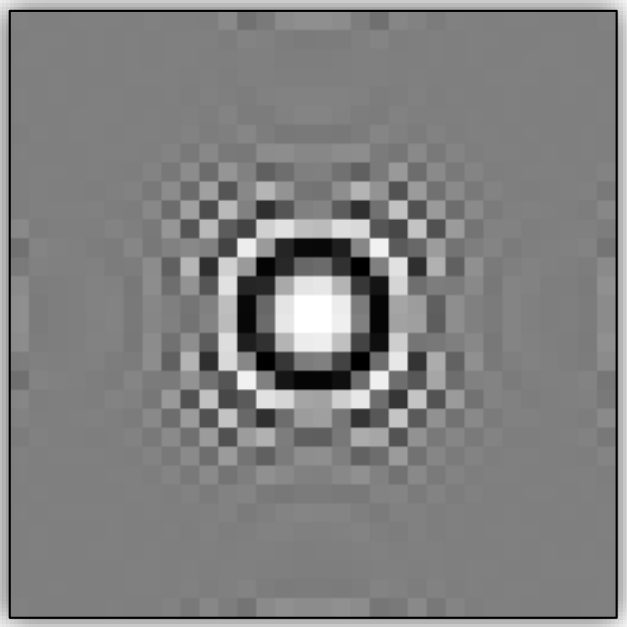
- Aliasing-Effekt: Fehler durch ungenügendes Abtasten von Signalen
  - im rekonstruierten Signal (z.B. Darstellung des Bildes am Monitor) treten Frequenzen auf, die im Original nicht enthalten sind
- Lösungsansätze
  - Vorfilterung des Signals vor der Abtastung (hohe Frequenzen „entfernen“): im allgemeinen Fall nicht möglich, bei Texturen schon!
  - höhere Abtastung des Signals, anschließend „Mitteln“ (rechtes Bild)



Abtastung ausreichend



Abtastung zu niedrig → Aliasing

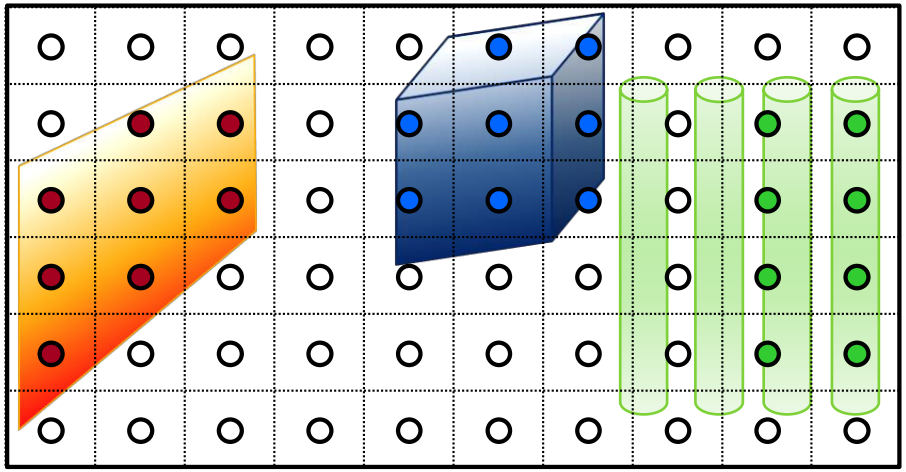


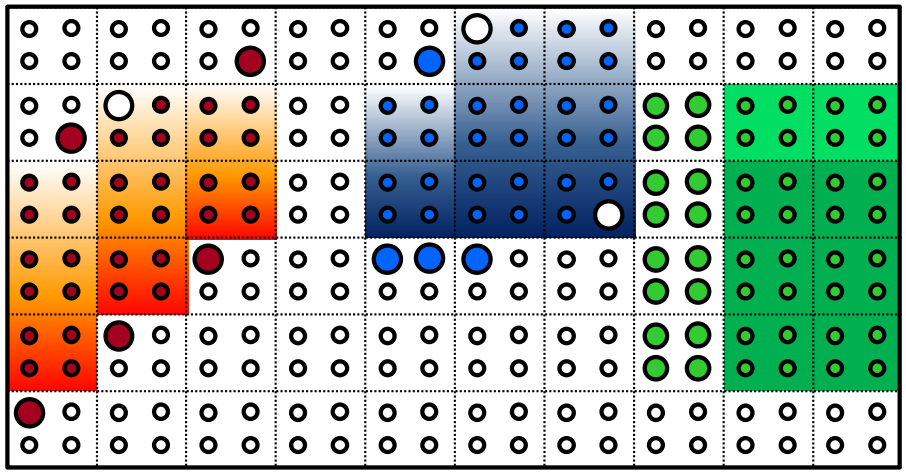
Überabtastung

Raytracing und Aliasing

Anti-Aliasing Strategien

- **Anti-Aliasing**: Versuch Aliasing zu reduzieren oder zu vermeiden
- wir können das Bildsignal als Ganzes nicht bandbegrenzen
- es bleibt daher i.d.R. – zumindest für Aliasing aufgrund von Objektkanten und geometrischem Detail – nur Überabtastung
- Überabtasten + „Mitteln“: ein Objekt bedeckt nur einen Teil eines Pixels und wir versuchen durch Überabtasten (= Supersampling) herauszufinden, welchen Beitrag die Objekte zu den Pixelfarben liefern

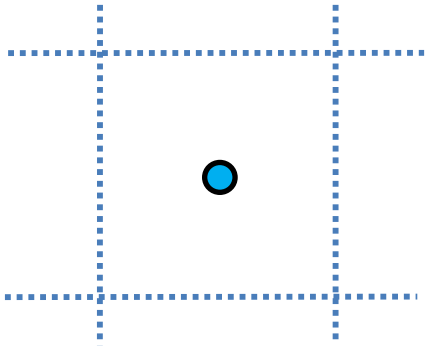




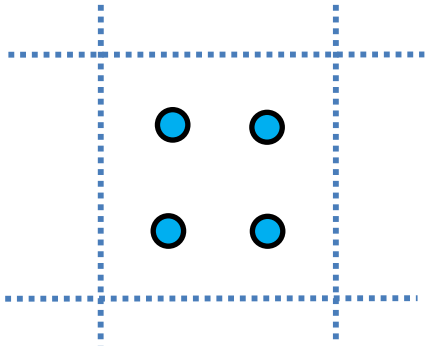
Anti-Aliasing Strategien

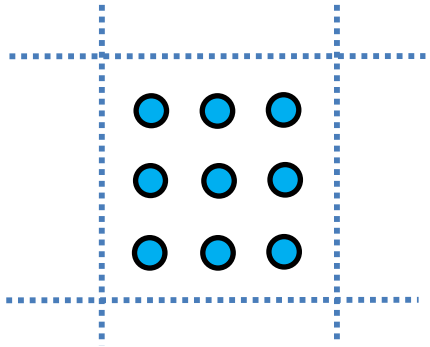
Uniformes Supersampling

- statt Abtastung eines Punktes innerhalb eines Pixels (ein Sample) tasten wir k^2-mal in **äquidistanten** Intervallen (= uniform) ab



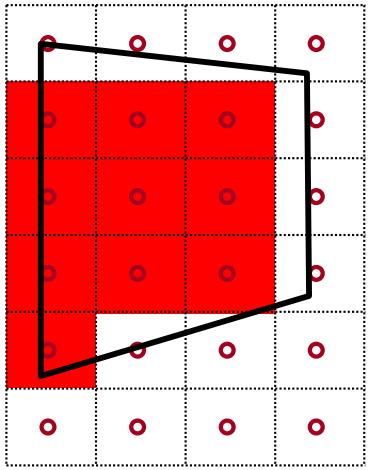
1 Pixel – 1 Sample

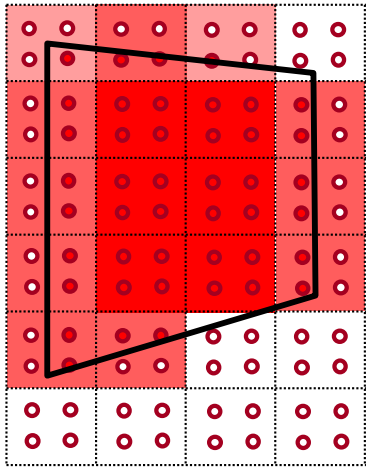




1 Pixel – mehrere Samples

- Pixel-Farbe: Mittelwert der k^2-Farben (es gibt bessere Rekonstruktions-/Tiefpaßfilter!)
- Primärstrahlen dann auch durch nicht-ganzzahlige Pixel-Positionen auf der Bildebene

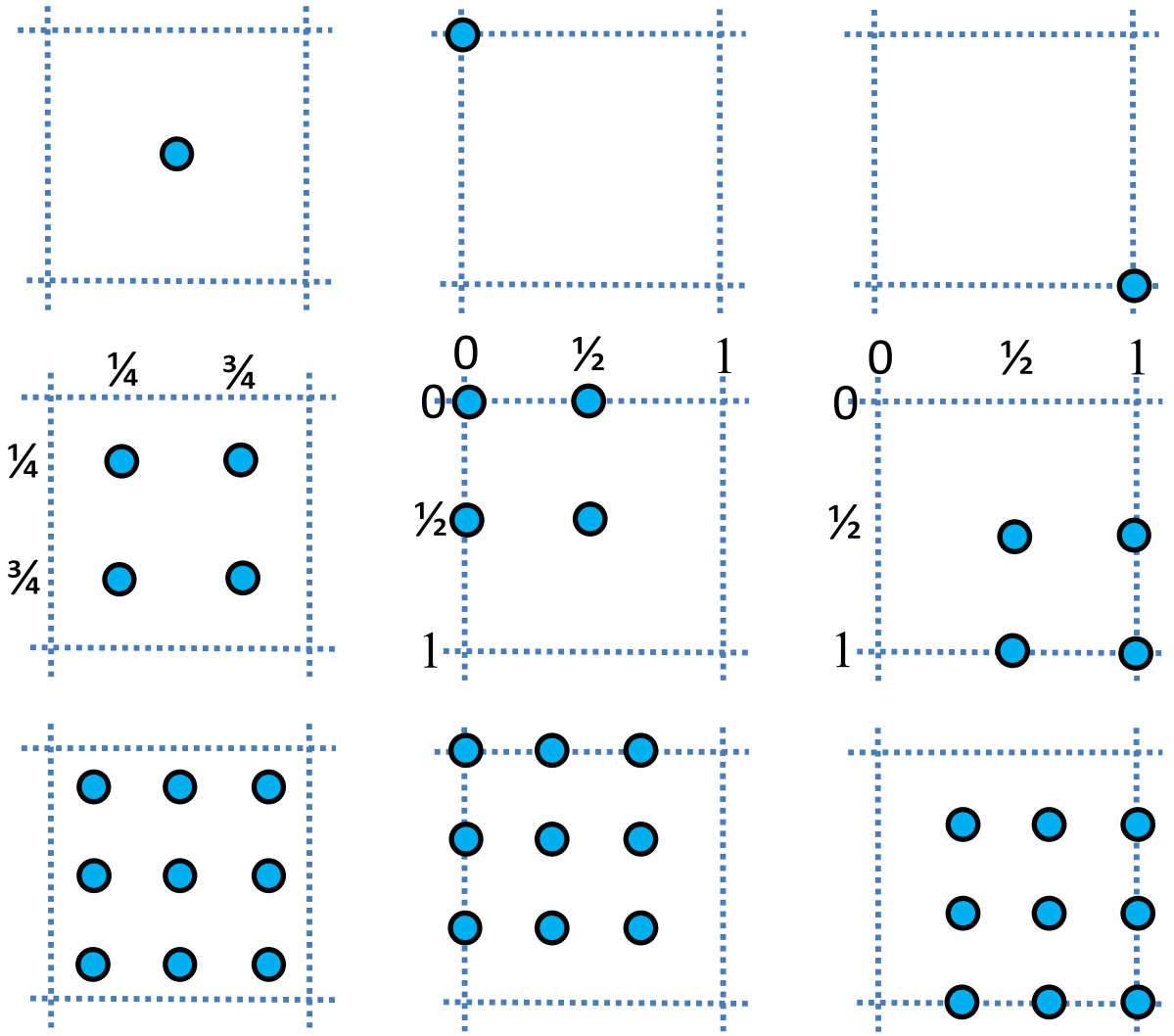




Anti-Aliasing Strategien

Uniformes Supersampling

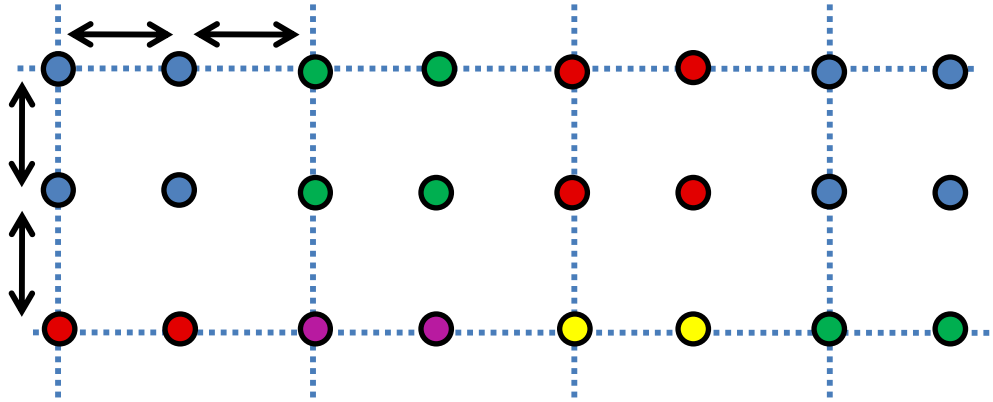
- Abtastung eines Pixels



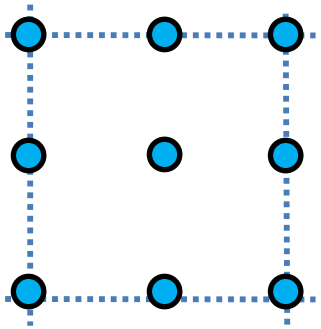
Anti-Aliasing Strategien

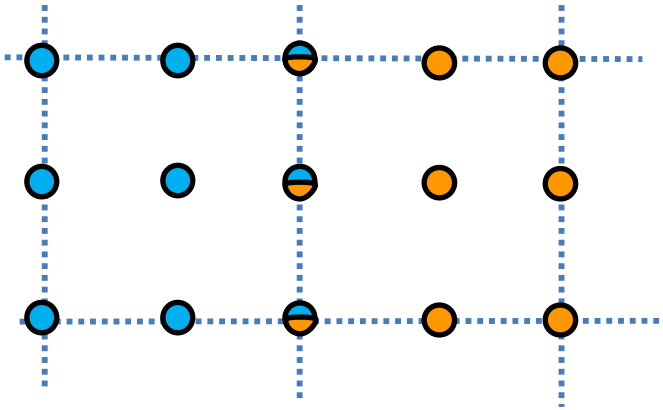
Uniformes Supersampling

- Abstand zwischen den Samples im Bild muss immer gleich sein



- so darf es nicht sein:



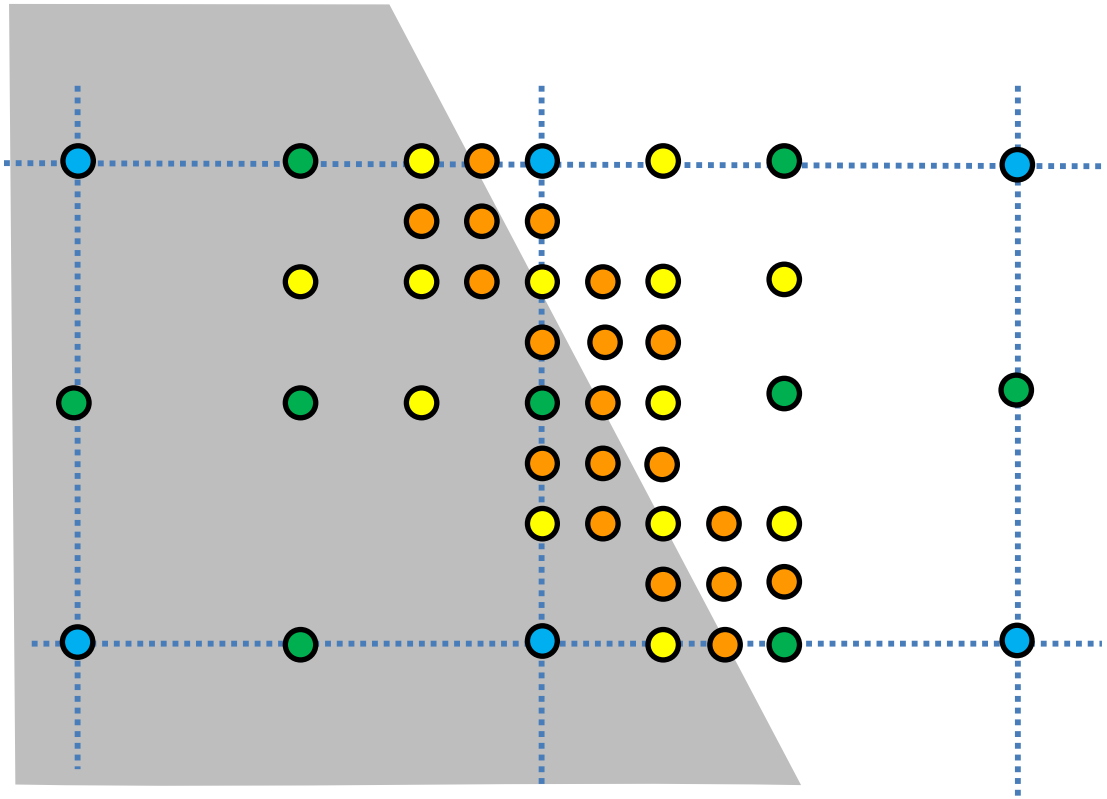


mehrere Samples an derselben Stelle

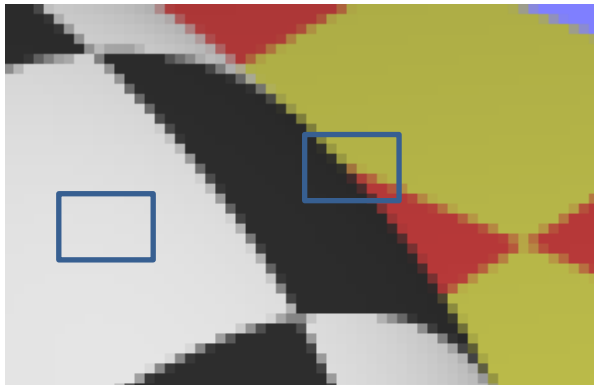
Anti-Aliasing Strategien

Adaptives Supersampling (heute für AA so nicht mehr verwendet)

- ist die (Farb-)Differenz benachbarter Samples zu groß, dann unterteile den Pixel in 4 Bereiche und verfolge weitere Strahlen
- „einfarbige“ Pixel im Schnitt 1 Sample, Kanten werden trotzdem geglättet
- Farbdifferenzen sind nicht verlässlich (potentiell immer noch Unterabt.)



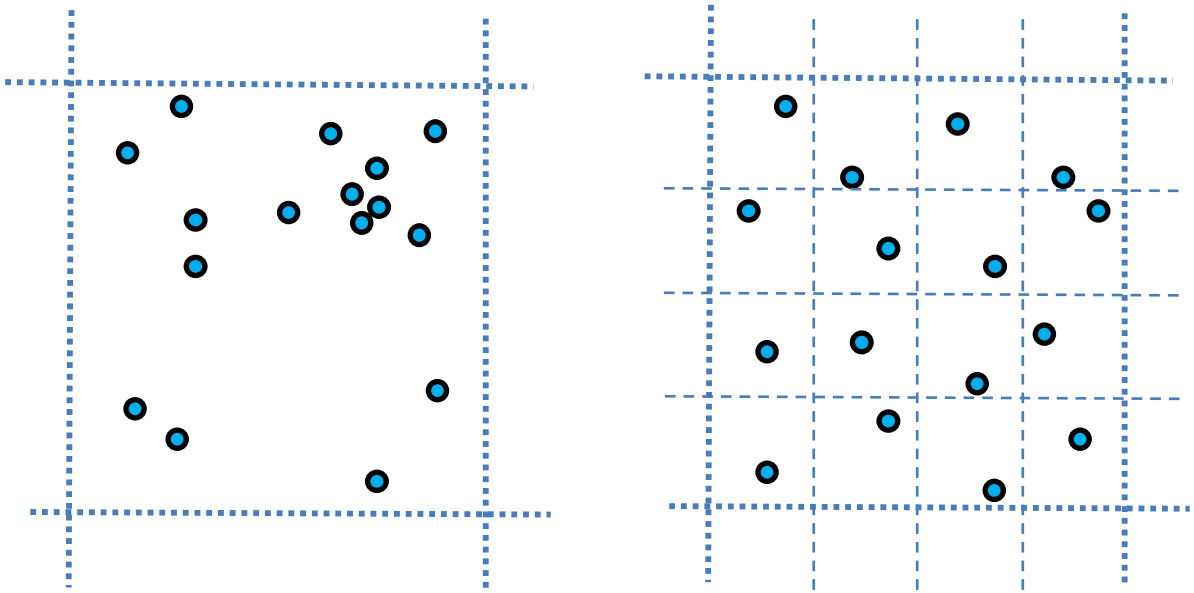
- ursprüngliche Samples
- 1. Unterteilung
- 2. Unterteilung
- 3. Unterteilung



Anti-Aliasing Strategien

Stochastisches Sampling

- uniformes Supersampling löst hohe Frequenzen oft nicht gut auf
  → stochastisch = verwende zufällige Samples in einem Pixel (links)
- noch besser: zufällige Samples mit Stratifikation (Stratified Sampling)
  - unterteile Pixel in Gitter, wähle einen Zufallspunkt pro Stratum (Zelle)
  - sehr oft verwendeter Ansatz (rechts), auch „stratified jittered“
- nicht-uniformes Abtasten: reduziert Aliasing, aber führt zu mehr Rauschen (von unserer Wahrnehmung als weniger störend empfunden)



Anti-Aliasing Strategien

Uniformes und stochastisches Überabtasten

Referenz
(256 SPP)

1 SPP
(uniform)

1 SPP
(jittered)

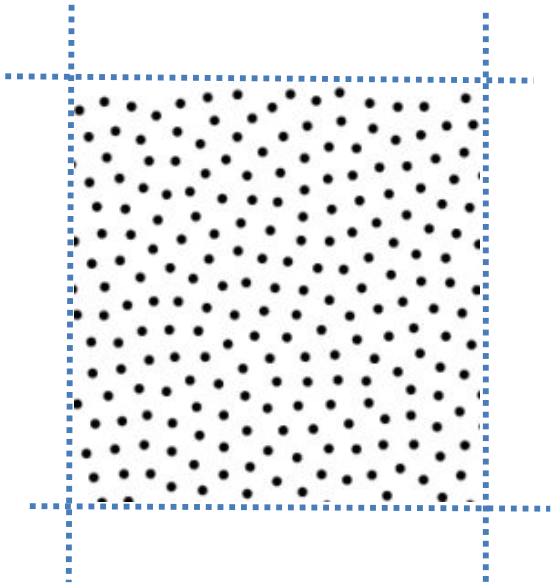
4 SPP
(jittered)

SPP = samples per pixel

Anti-Aliasing Strategien

Blue Noise Sampling (blaues Rauschen)

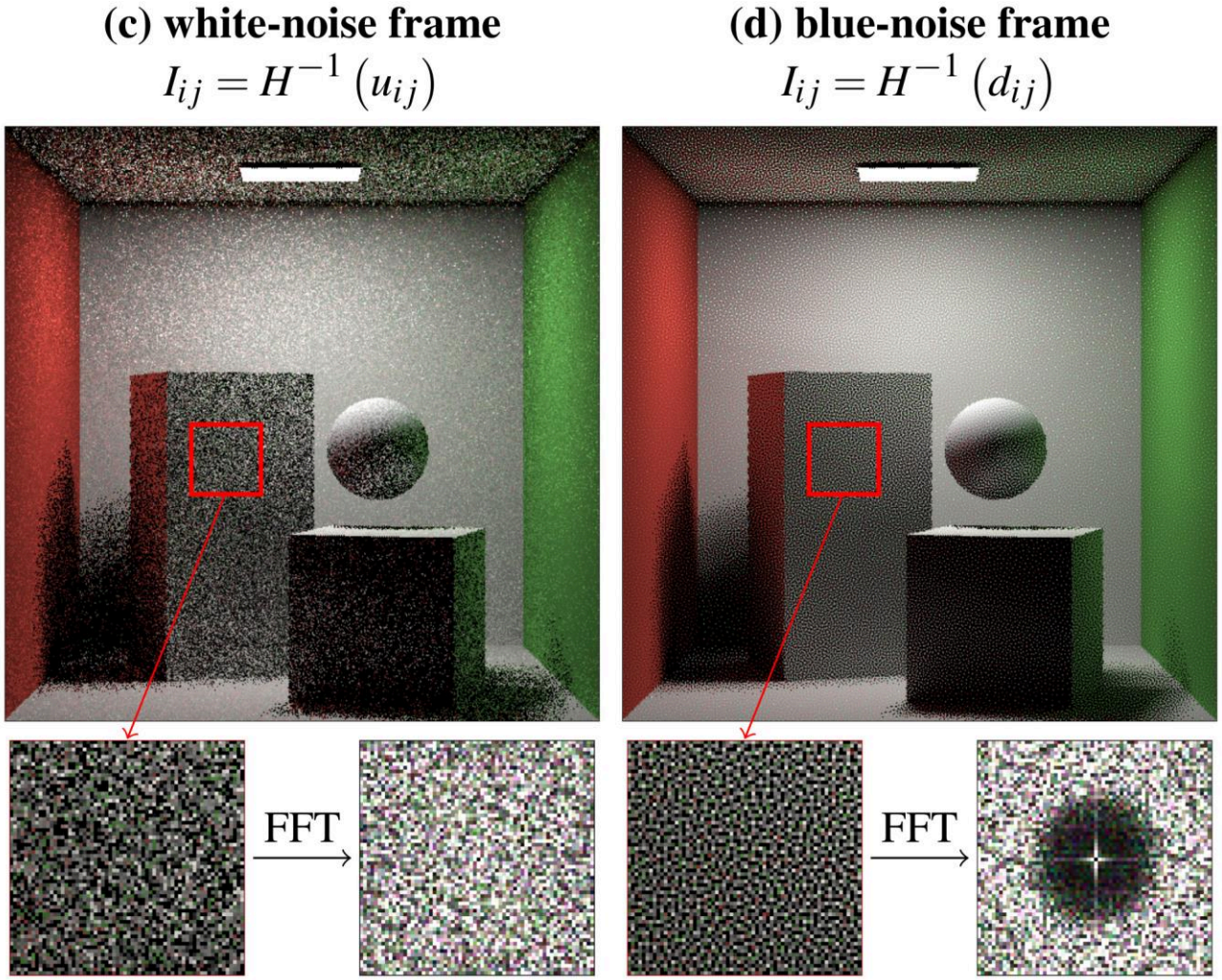
- möglichst uniformer Abstand zwischen den nahsten Punkten
- weniger/keine niederfrequenten Anteile, isotropes Frequenzspektrum
- in der Praxis: vorberechnete Abtastpunkte
- unerlässlich für Echtzeit-Raytracing mit wenigen Abtastpunkten



Anti-Aliasing Strategien

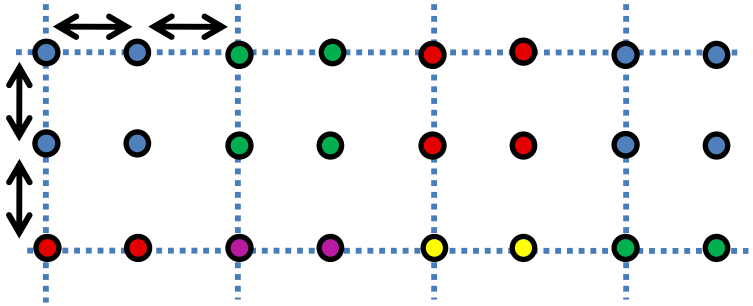
Blue Noise Sampling (blaues Rauschen)

- Bild aus „Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds Between Frames”, Heitz und Belcour, https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.13778

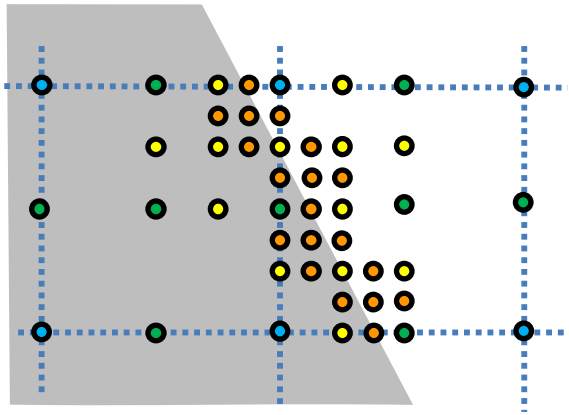


Zusammenfassung

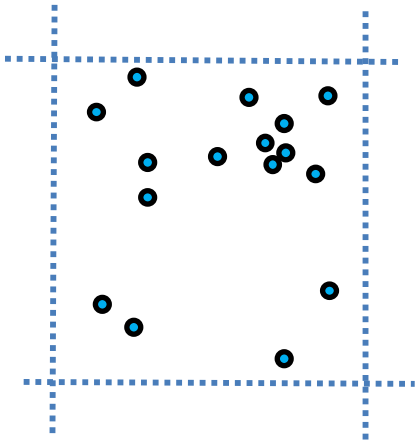
Uniformes Supersampling



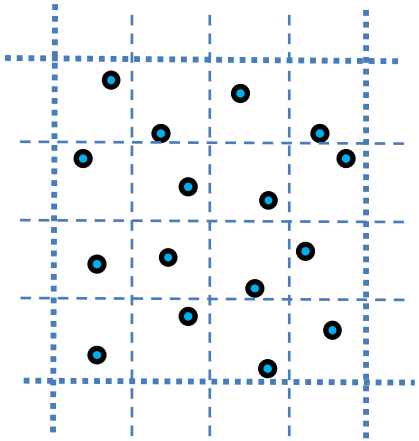
Adaptives Supersampling



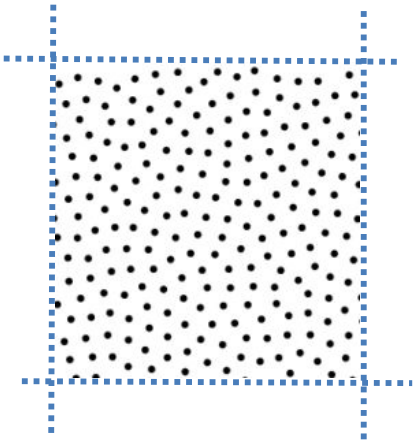
Stochastisches S.



Stratified Sampling



Blue Noise Sampling

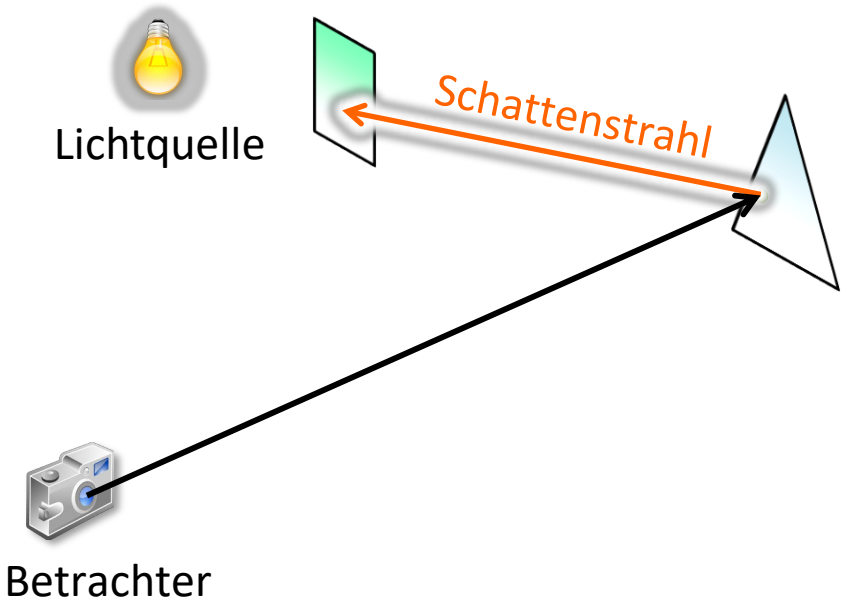


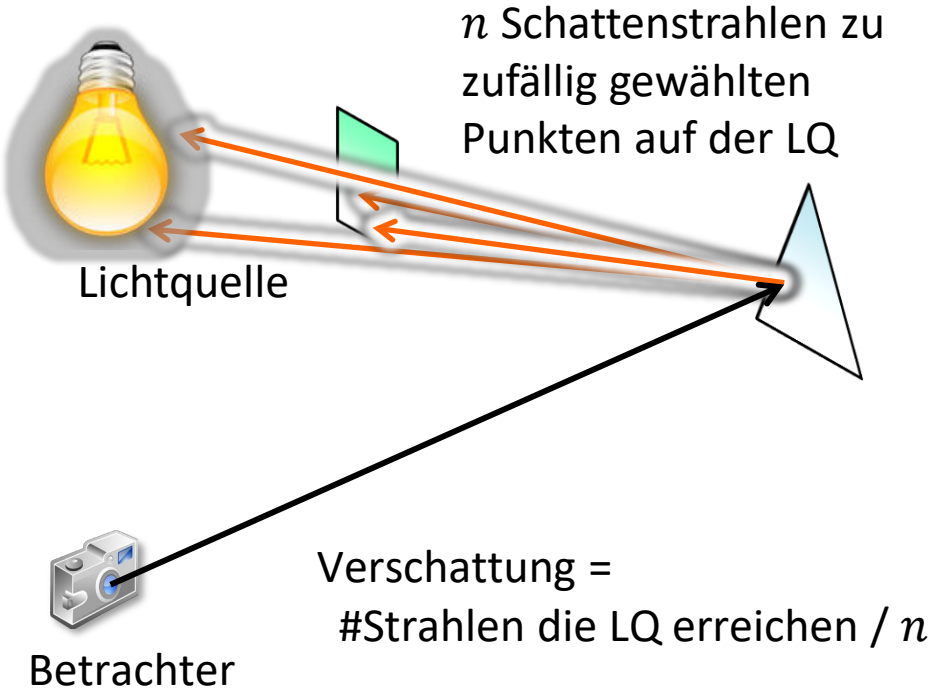
Distributed Raytracing

Probleme des Whitted-Style Raytracing-Verfahrens

- die Bilder sehen zu makellos aus:
  - perfekte Spiegelung und Transmission
  - harte Schattenkanten, unendliche Schärfentiefe etc.
  - was kann man da tun?

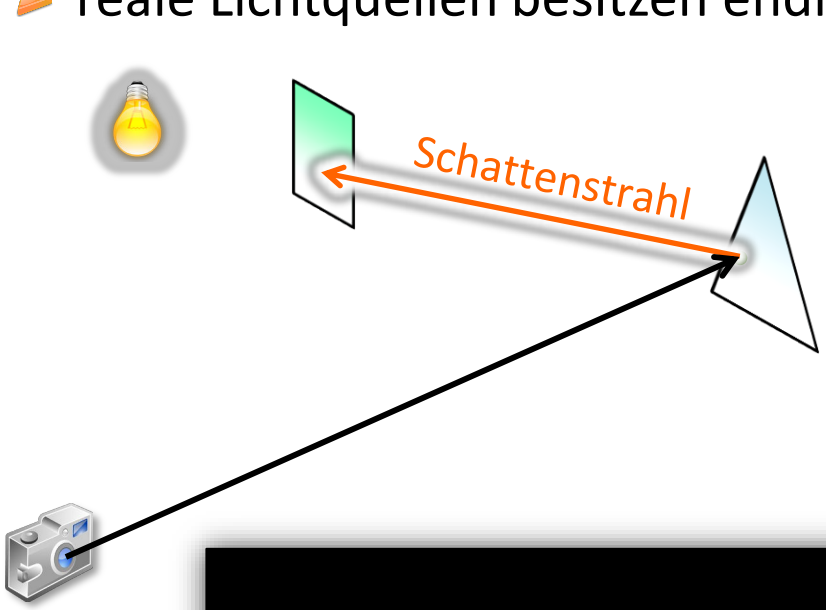
- Beispiel: Schattenstrahlen

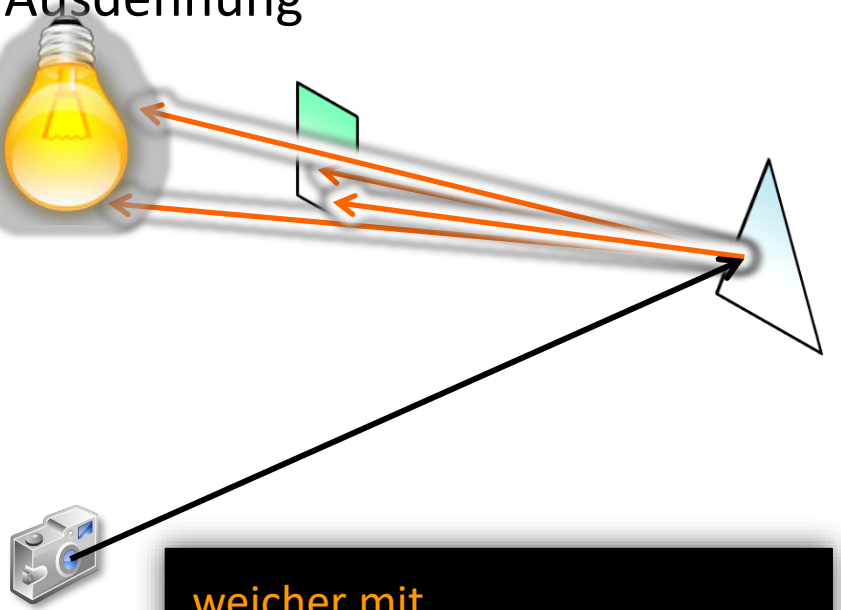




Weiche Schatten

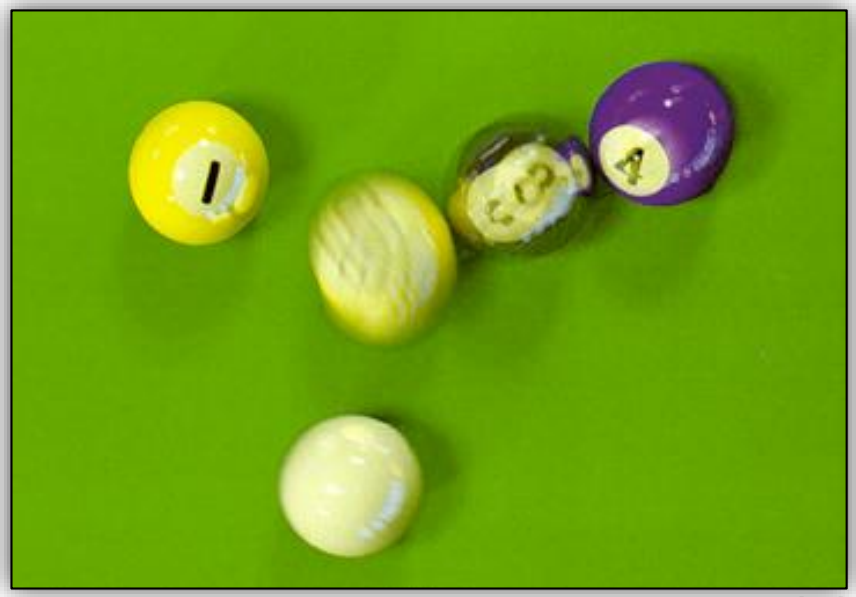
- reale Lichtquellen besitzen endliche Ausdehnung



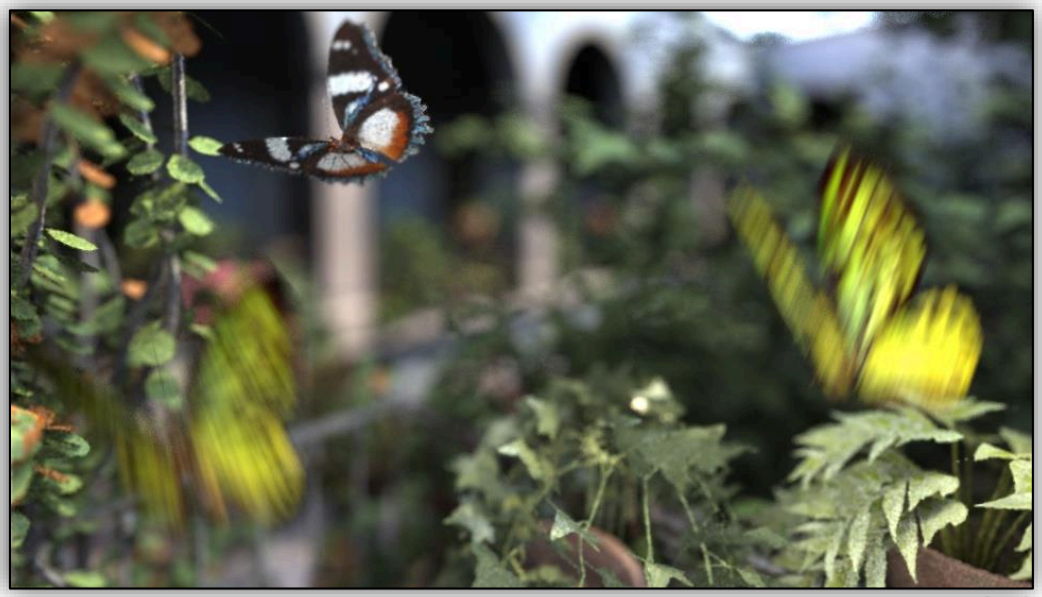


Bewegungsunschärfe (engl. motion blur)

- Verteilung der Strahlen in der Zeit
  - vgl. Belichtungszeit bei realen Kameras
  - Objekte bewegen sich in einem Zeitintervall $[t; t + \Delta t)$
  - erzeuge für jeden Pixel n Strahlen für einen Zeitpunkt $t' \in [t; t + \Delta t)$
    - führe Raytracing mit der Szene zum Zeitpunkt t' durch
    - mittle Farbwerte



[36]

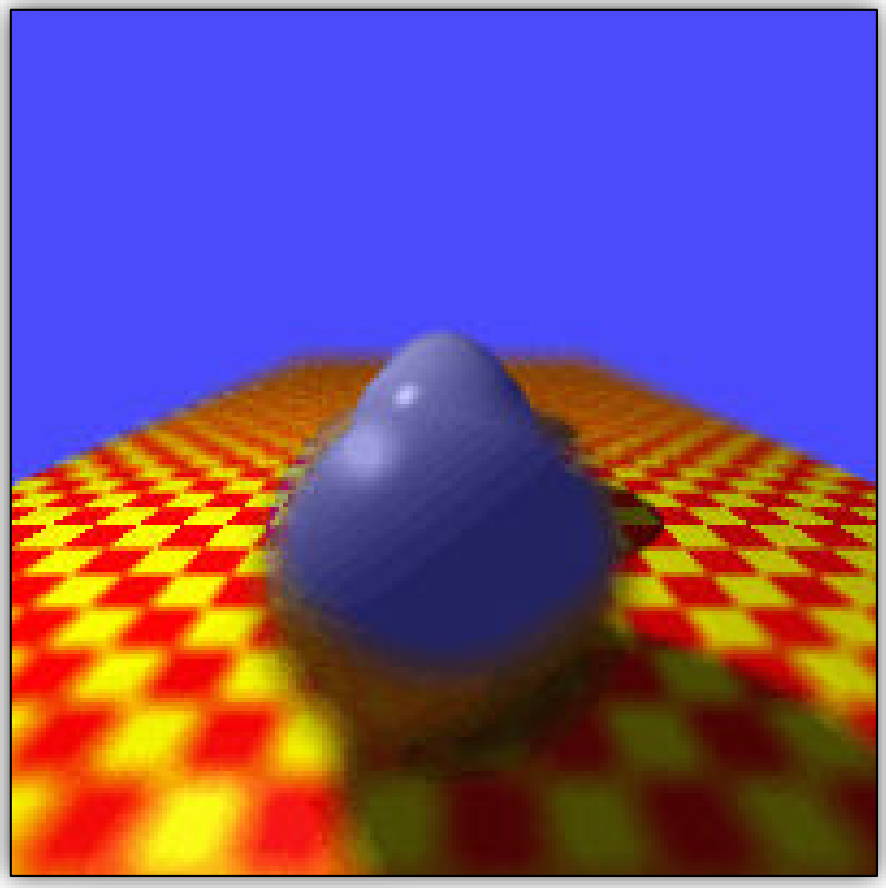


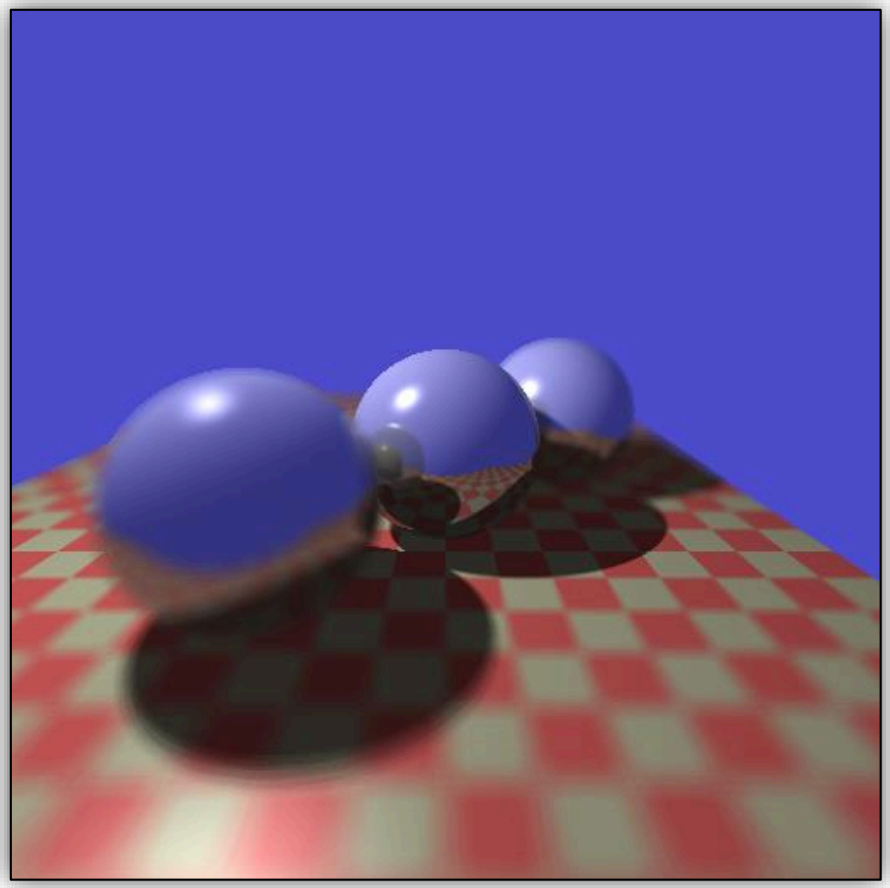
[37]

Bewegungsunschärfe

```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    vec3 c = 0.0f;
    for ( s = 0; s < nSuperSample; s++ ) {
      float x_ = ..., y_ = ...;          // Supersampling in Abhängigkeit von s
      for ( ts = 0; ts < nTimeSamples; ts++ )
      {
        float time = t + delta_t * random01();
        Ray ray;
        generateRay( &ray, x_, y_, time );      // time wg. Kamerabewegung
        c += raytrace( ray, time );             // time wg. Objektbewegung
      }
    }
    c /= (float)( nTimeSamples * nSuperSample );
} }
```

Tiefenunschärfe





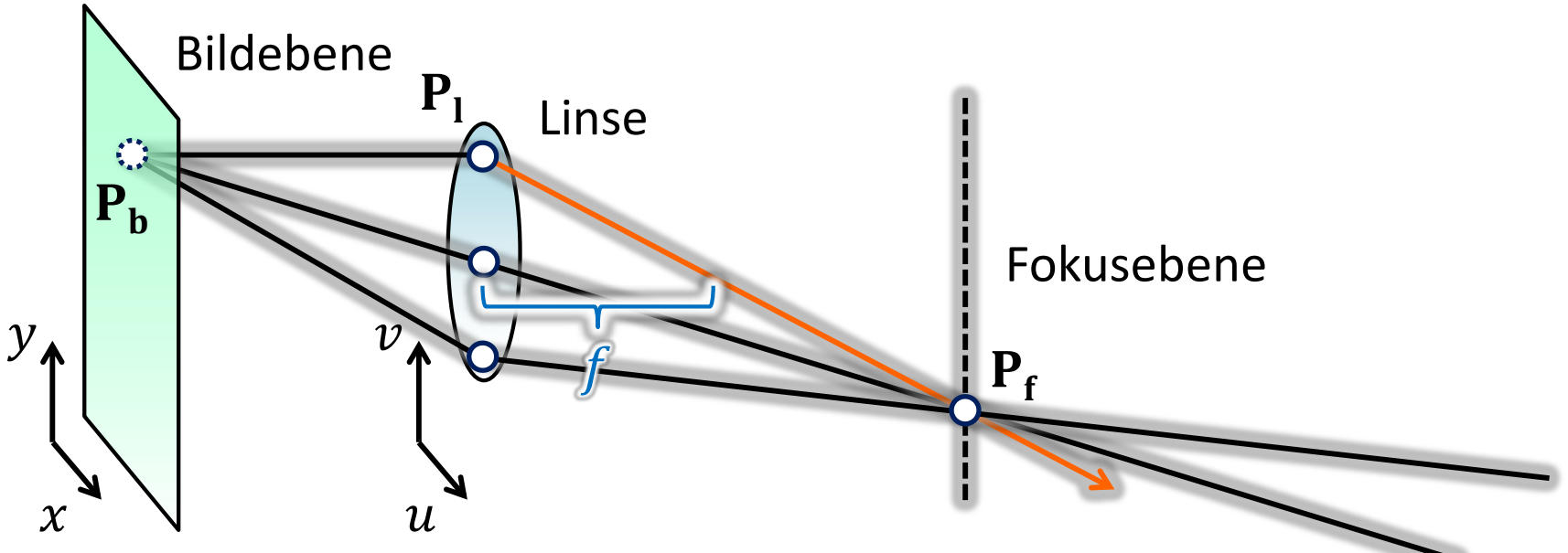
Cook et al. 1984 [38]

Rob Cook is among the first people to have been awarded an Oscar for software. He is the co-architect and primary author of Pixar's RenderMan.

Tiefenunschärfe

Modell der „dünnen konvexen Linse“

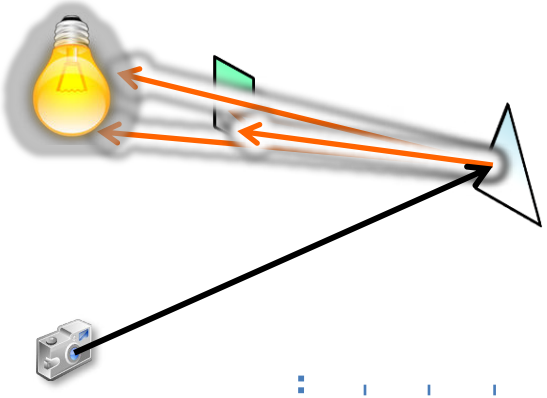
- weitere Samples für das Abtasten der Linsenfläche
  - wähle Punkt auf der Bildebene $\mathbf{P_b}$ (und somit Pixel) und der Linse $\mathbf{P_l}$
  - Strahlen durch alle Punkte auf der Linse führen zum Punkt im Fokus $\mathbf{P_f}$
    - auch der Strahl durch das Zentrum, der nicht abgelenkt wird
  - berechne daraus Punkt auf der Fokusebene $\mathbf{P_f}$ (Abstand zur Fokusebene berechnet aus Brennweite f und Sensorabstand)
  - erzeuge Primärstrahl von $\mathbf{P_l}$ durch $\mathbf{P_f}$

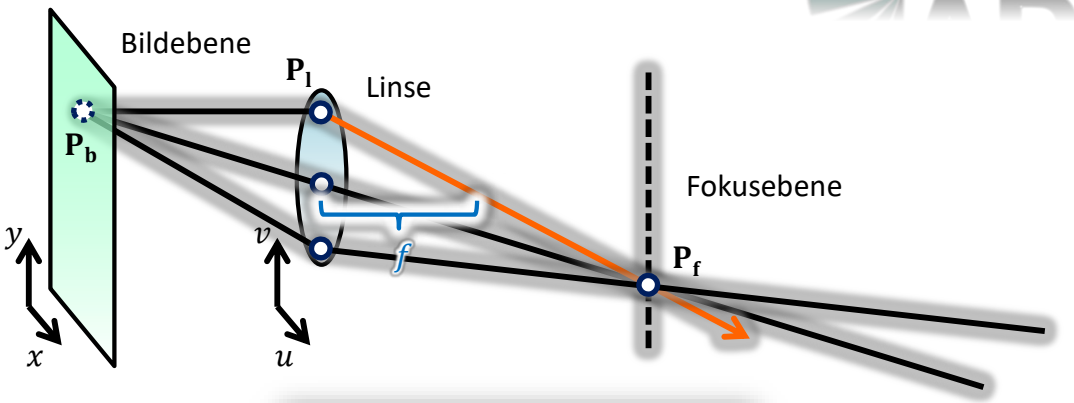


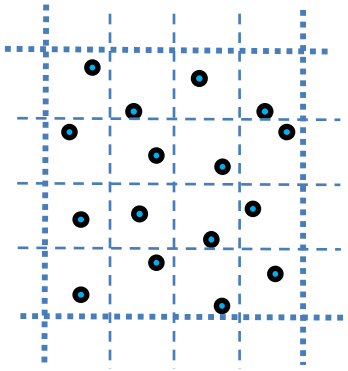
Bewegungs- und Tiefen(un)schärfe

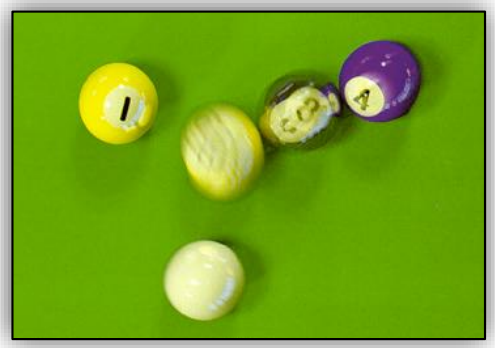
```
class Ray {
  vec3   origin, direction; // Startpunkt und Richtung des Strahls
  float  t;                 // Strahlparameter
  void * object;            // Zeiger auf evtl. getroffenes Objekt
  ...
};

for ( y = 0; y < height; y++ ) {
  for ( x = 0; x < width; x++ ) {
    vec3 c = 0.0f;
    for ( s = 0; s < nSuperSample * nLensSample; s++ ) {

      float x_ = ..., y_ = ...;                  // Supersampling
      float u = random01(), v = random01(); // verwendet, um Zufallspunkt
                                            // auf der Linse zu wählen
      for ( ts = 0; ts < nTimeSamples; ts++ )
      {
        float time = t + delta_t * random01();// Sampling in der Zeit

        Ray ray;
        generateRayThinLens( &ray, x_, y_, u, v, time );

        c += raytrace( ray, time );
      }
    }
    c /= (float)( nTimeSamples * nSuperSample * nLensSample );
} }
```

Zusammenfassung: Distributed Raytracing



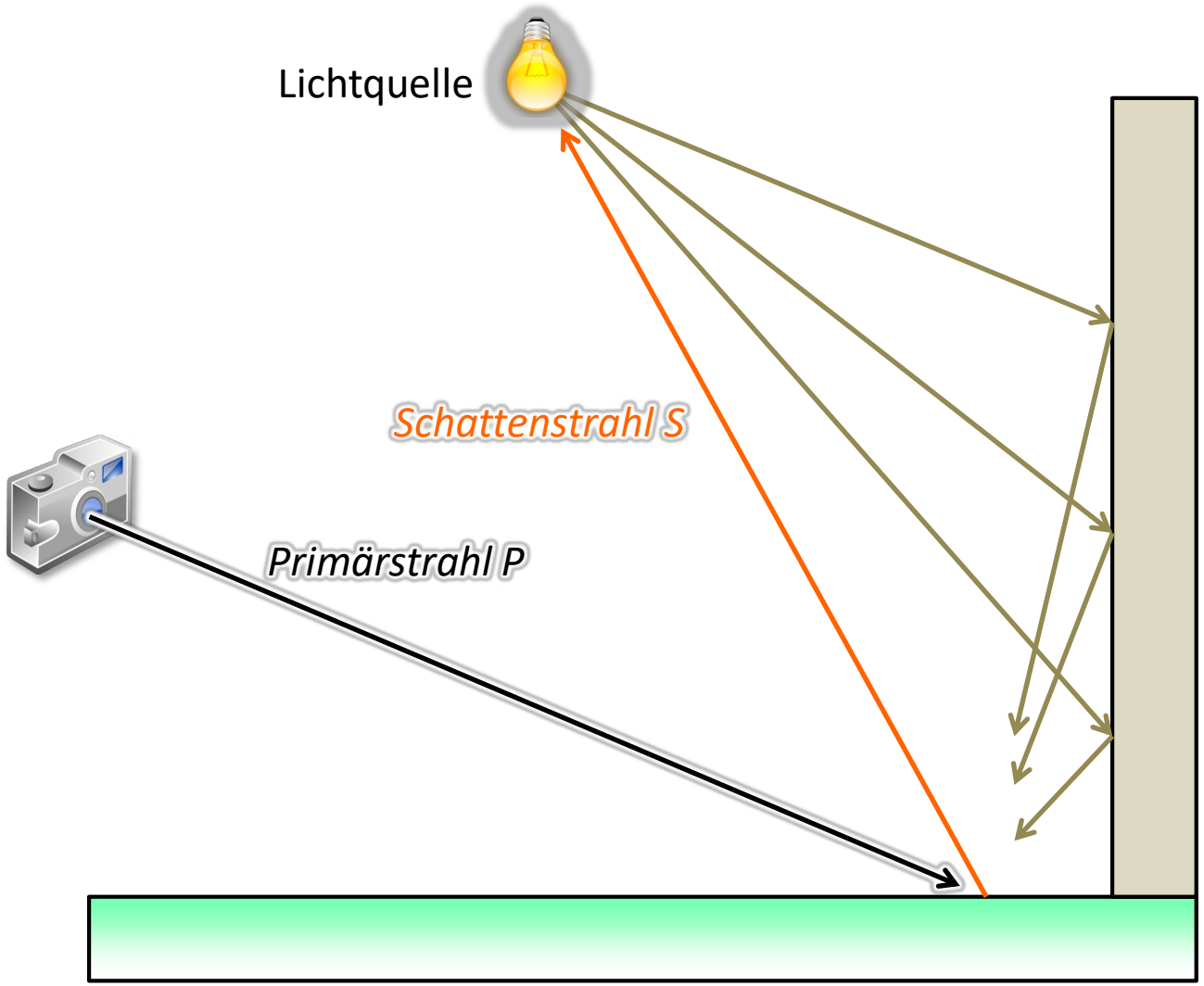






```
for ( all pixels ) {
  vec3 c = 0.0f;
  for ( s = 0; s < nSuperSample * nLensSample * nTimeSamples; s++ ) {
    float x_ = ..., y_ = ...;
    float u = random01(), v = random01();
    float time = t + delta_t * random01();

    generateRayThinLens( &ray, x_, y_, u, v, time );
    c += raytrace( ray, time );
  }
  c /= (float)( nTimeSamples * nSuperSample * nLensSample );
}
```

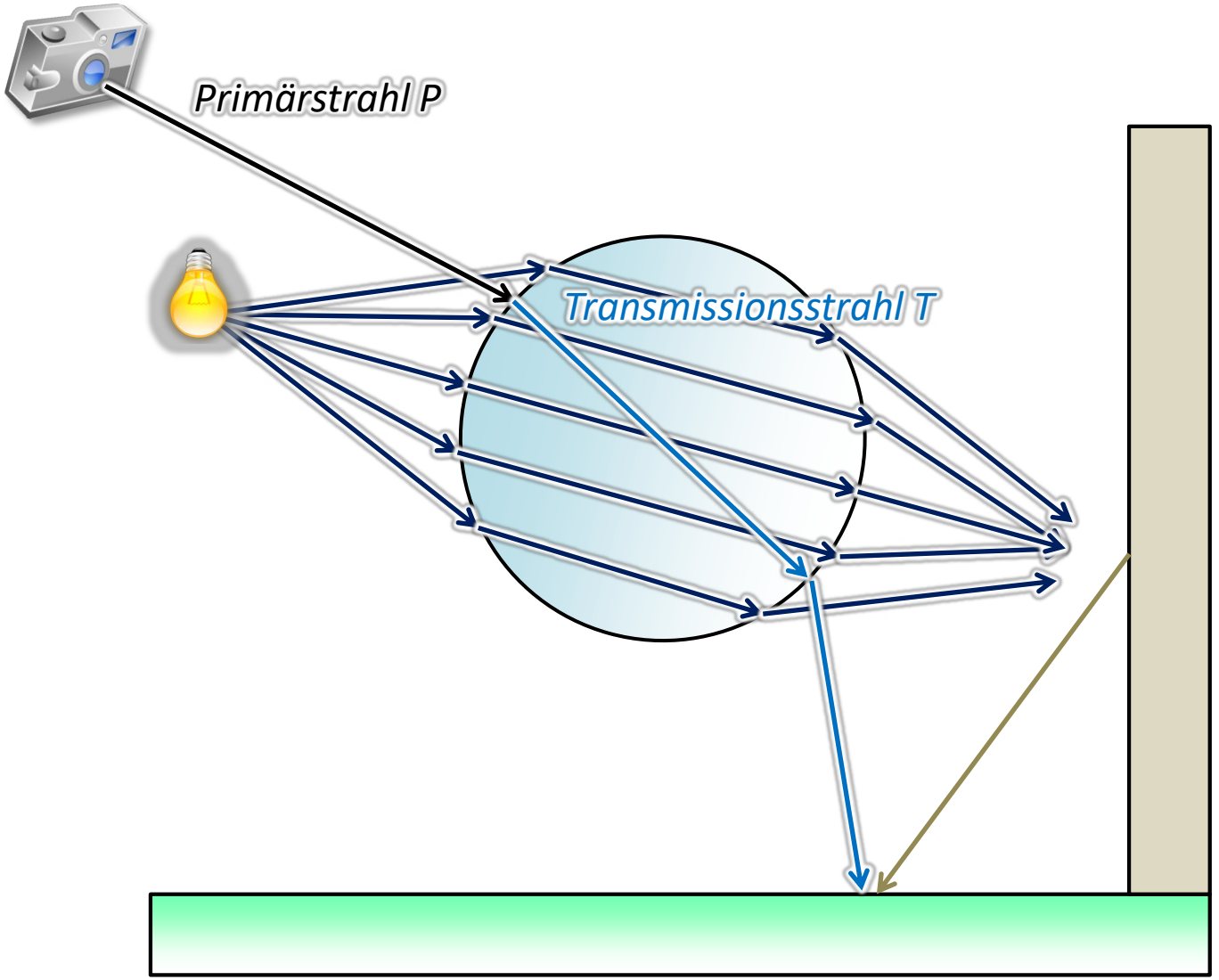
Und was ist mit …



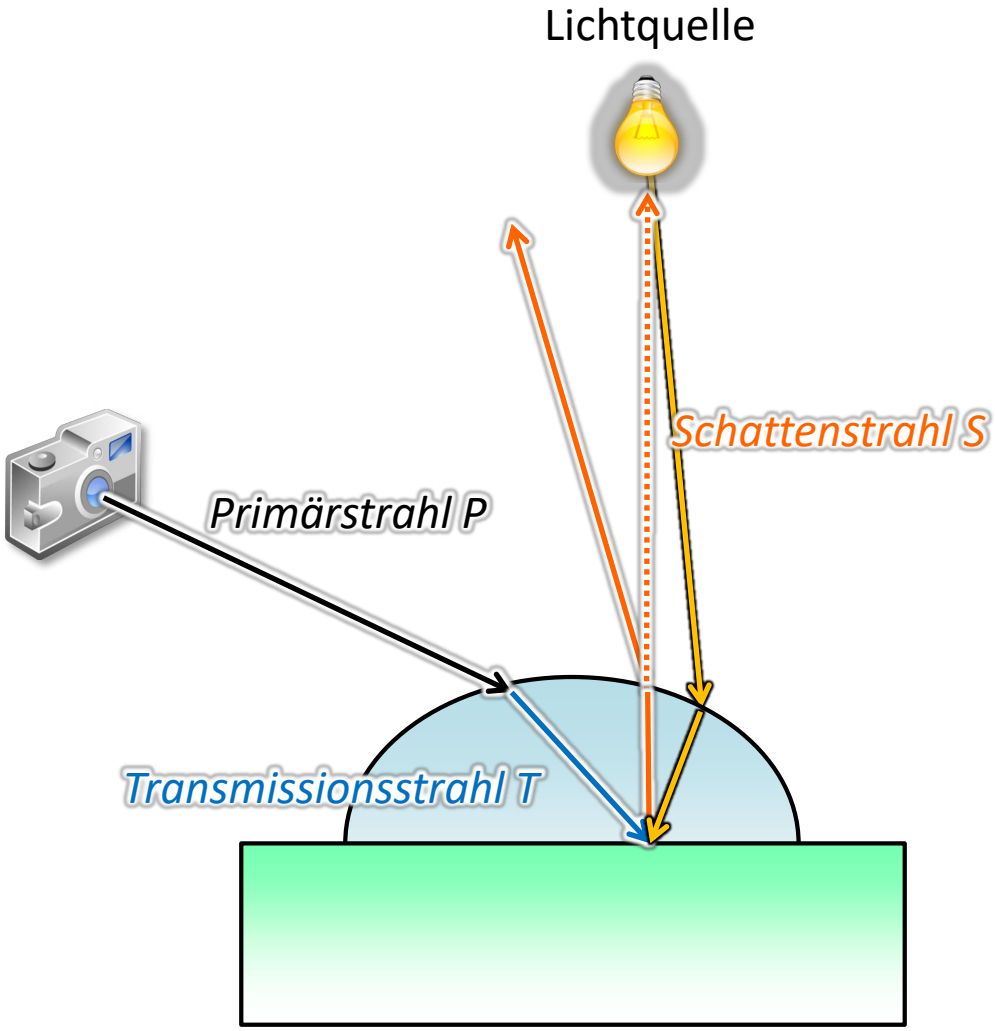
Und was ist mit …

http://nestohuis.deviantart.com/art/Caustic-Chrome-Tut-55223982[40]

Und was ist mit …



Und was ist mit …



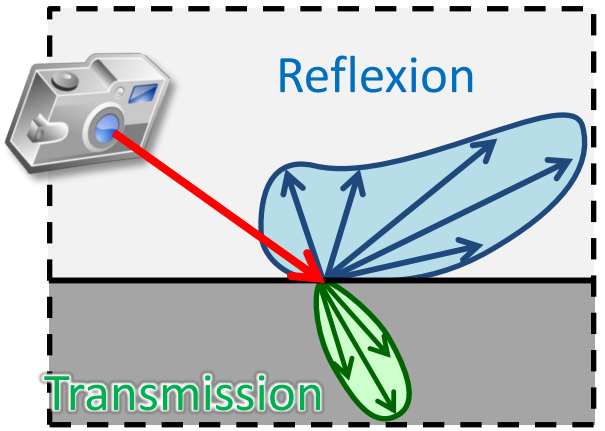
[39]

- Forschung zu diesem Thema: Johannes Hanika, Marc Droske, and Luca Fascione. Manifold next event estimation. *Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering)*, 34(4), 2015
- Granizo-Hidalgo et al., Interactive Rendering of Caustics using Dimension Reduction for Manifold Next-Event Estimation. *ACM on Computer Graphics and Interactive Techniques*, 2024

Imperfekte Spiegelung und Transmission

Grundidee: Distributed Raytracing

- um alle Lichtwege zu berücksichtigen, sind mehrere Strahlen für Reflexion und Transmission von jedem Schnittpunkt aus notwendig
  - es werden also viele Sekundärstrahlen weiterverfolgt
  - (quasi-)zufällige Richtungen
  - deren Beiträge zum reflektierten/ gebrochenen Licht werden durch die BRDF und BTDF (= BSDF) bestimmt



- wie viele Strahlen werden benötigt?
  - durch Rekursion verzweigt der „Strahlenbaum“ schon beim Whitted-Style Raytracing
  - Explosion bei mehreren Sekundärstrahlen an jedem Punkt
- physikalisch-basierter Lichttransport → Radiometrie

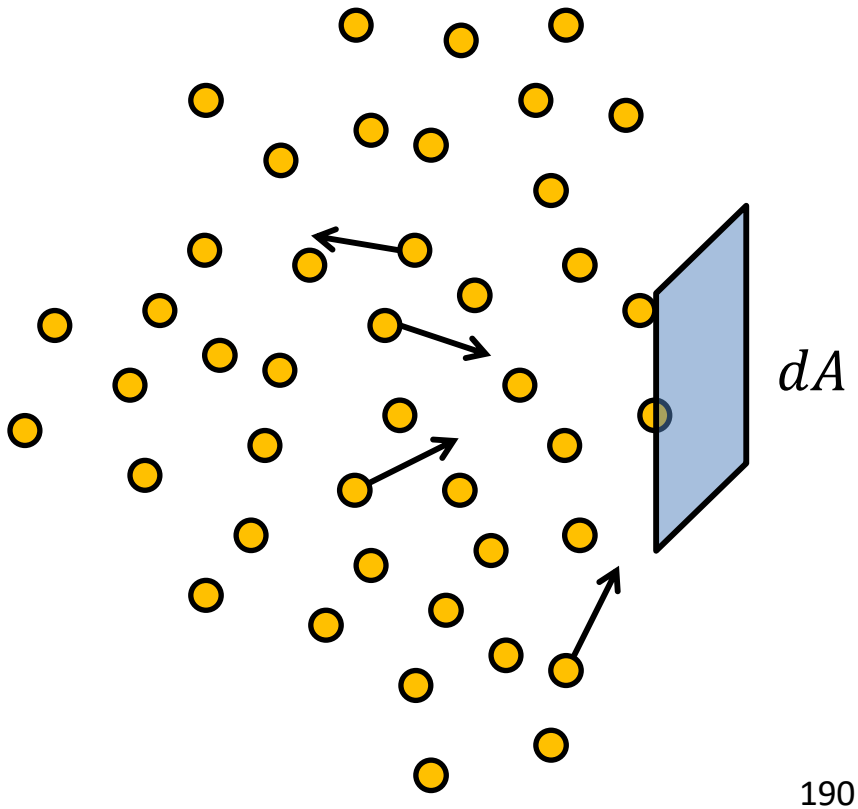
Radiometrie (zur Erinnerung)

Flussdichte (engl. Irradiance und Radiosity/Radiant Exitance) $[\mathrm{W/m^2}]$

- eine offensichtliche Fragestellung, z.B. zur Beleuchtungsberechnung: wie viel Licht erreicht eine Oberfläche?
  - $E = \frac{d\Phi}{dA} = \frac{d^2Q}{dAdt}$ Fluss/Leistung pro Fläche in $[\mathrm{W/m^2}] = [\mathrm{J/(m^2 \cdot s)}]$

- Irradiance beschreibt die Menge des Lichts, aber nicht aus welcher Richtung es kommt!

- diese Richtungsabhängigkeit benötigen wir für BRDFs!

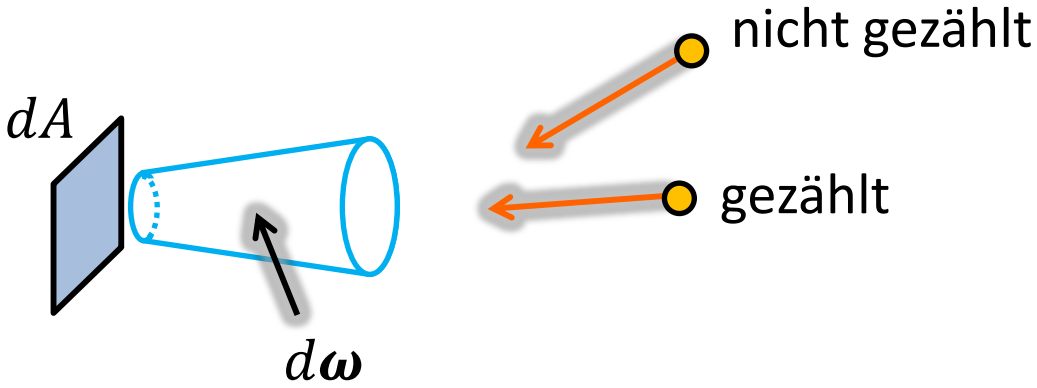


Radiometrie

Strahldichte (engl. Radiance)

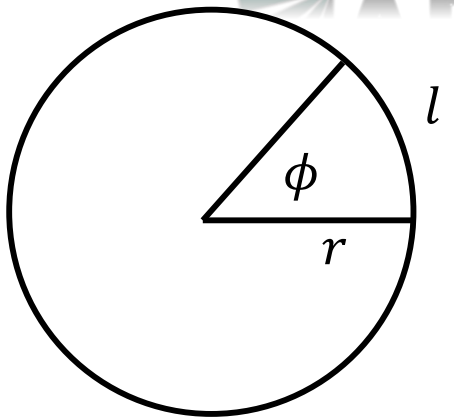
- beschreibt wie viel Licht mit bestimmten Richtungen transportiert wird
  - etwa vergleichbar mit dem, was unser Auge misst
  - **(Raster-)Bilder repräsentieren Radiance**

- Vorstellung: „Messapparat“ bestehend aus einem Irradiance-Detektor (misst Leistung pro Fläche in $[\mathrm{W}/\mathrm{m}^2]$) und kegelförmiger Trichter
  - wir brauchen jetzt noch ein Maß für die „Menge der Richtungen“, die der Trichter abdeckt



Raumwinkel

- Menge von Richtungen in der Ebene können durch einen Winkel angegeben werden
- der Winkel ist 1D auf S^1 (1-Kugel = Kreis) und im Bogenmaß definiert als:
  $\phi = l/r$ Länge des Kreisbogens / Radius



- der **Raumwinkel** in 2D auf S^2 (2-Kugel) ist das Pendant zum Winkel
  - auf Einheitskugel mit $r = 1\mathrm{m}$ umschließt ein Raumwinkel von $\Omega = \frac{A}{r^2} = 1\mathrm{sr}$ (Steradiant) eine Fläche von $A = 1\mathrm{m}^2$

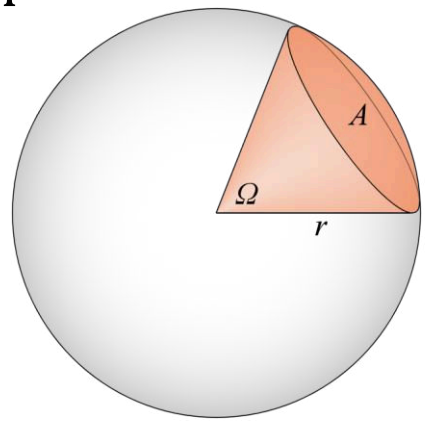
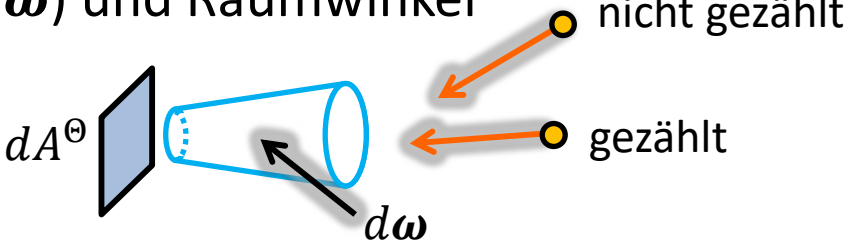


Bild: metricsystem.net/derived-units/special-names/steradian/

Strahldichte

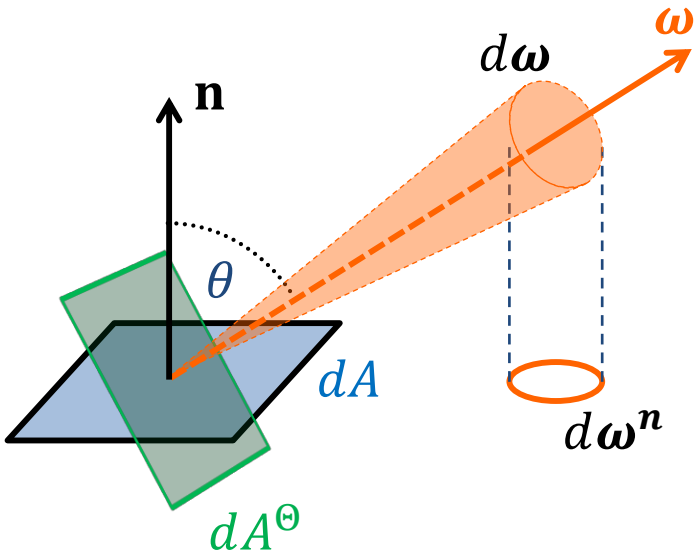
- Definition Radiance $L = \frac{dE}{d\omega} = \frac{d^2\Phi}{dA^\Theta d\boldsymbol{\omega}}$ in $\left[\frac{\mathrm{W}}{\mathrm{m}^2\mathrm{sr}}\right]$
  - Leistung Φ pro Fläche (senkrecht zu $\boldsymbol{\omega}$) und Raumwinkel



  - Leistung ist Energie pro Zeit, Radiance ist also:
    #Photonen · Energie eines Photons pro Zeit, Fläche und Raumwinkel

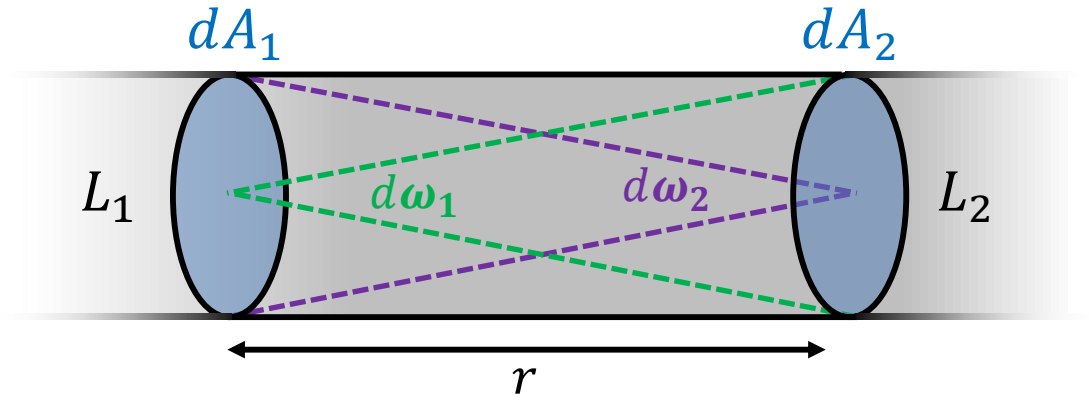
- Leistung pro Raumwinkel und pro projizierter Einheitsfläche

$$L = \frac{d^2\Phi}{dA\cos\theta\, d\boldsymbol{\omega}} \quad \left[\frac{\mathrm{W}}{\mathrm{m}^2\mathrm{sr}}\right]$$



Strahldichte als Erhaltungsgröße

- betrachte Energiefluss entlang eines Strahl (ohne Emission oder Absorption), d.h. Photonen bewegen sich entlang des Strahls im Vakuum



- wegen der Energieerhaltung ist der Fluss $[\mathrm{W}] = [\mathrm{Js}^{-1}]$ konstant:

$$L_1 dA_1 d\boldsymbol{\omega_1} = L_2 dA_2 d\boldsymbol{\omega_2} \quad \text{mit} \quad d\boldsymbol{\omega}_1 = dA_2/r^2 \text{ und } d\boldsymbol{\omega_2} = dA_1/r^2$$

$$L_1 dA_1 dA_2/r^2 = L_2 dA_2 dA_1/r^2$$

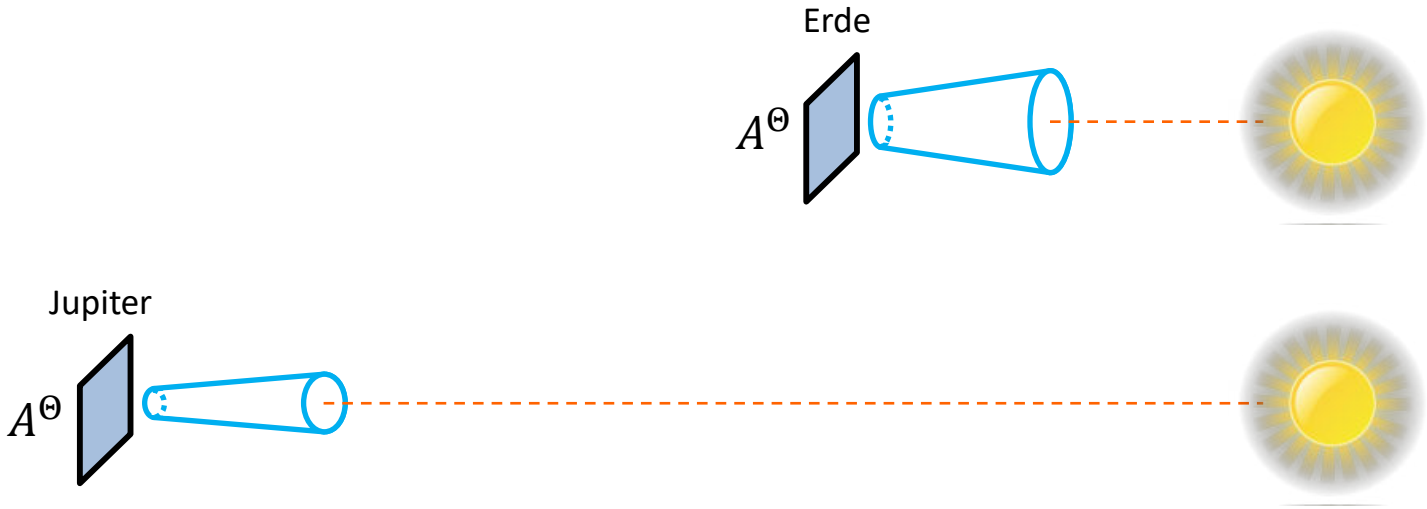
$$\Rightarrow L_1 = L_2$$

- Strahldichte entlang eines Strahls konstant
  → zu transportierende Größe beim Raytracing

Strahldichte

Beispiel: „Radiance auf der Erde und Jupiter“

- Radiance $L = \frac{dE}{d\omega} = \frac{d^2\Phi}{dA^{\Theta} d\boldsymbol{\omega}}$ in $\left[\frac{\mathrm{W}}{\mathrm{m^2 sr}}\right]$



- für den Betrachter erscheint die Sonne gleich hell: keine Absorption oder Streuung des Sonnenlichts im Vakuum
- aber: die Flussdichte ist auf dem Jupiter geringer, es kommt weniger Energie an, weil die Sonne einen kleineren Raumwinkel einnimmt

Zusammenhang radiometrischer Größen

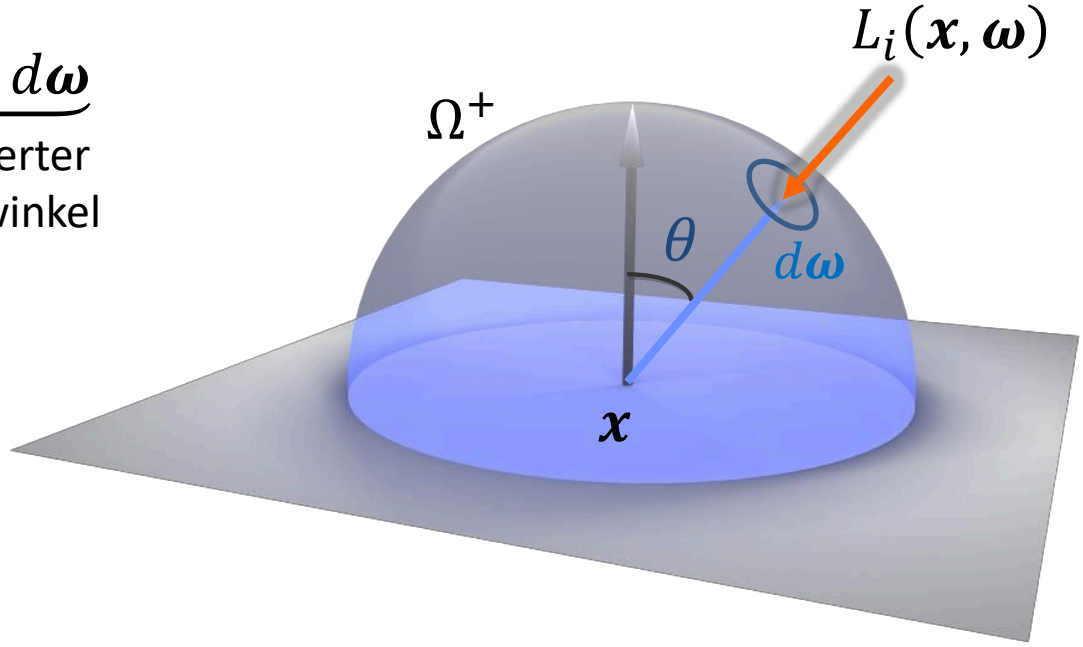
Strahldichte und Flussdichte

- Flussdichte $E = \frac{d\Phi}{dA}$ ist Leistung pro Fläche in $\left[\frac{\mathrm{W}}{\mathrm{m}^2}\right]$

- Strahldichte $L = \frac{d^2\Phi}{dA \cos\theta \, d\boldsymbol{\omega}} \Leftrightarrow L \cos\theta \, d\boldsymbol{\omega} = \frac{d^2\Phi}{dA} = dE$

- Integration über die einfallende Strahldichte L_i $[\mathrm{W}/(\mathrm{m}^2\mathrm{sr})]$

$$E(\boldsymbol{x}) = \int_{\Omega^+} L_i(\boldsymbol{x}, \boldsymbol{\omega}) \underbrace{\cos\theta \, d\boldsymbol{\omega}}_{\text{projizierter Raumwinkel}}$$

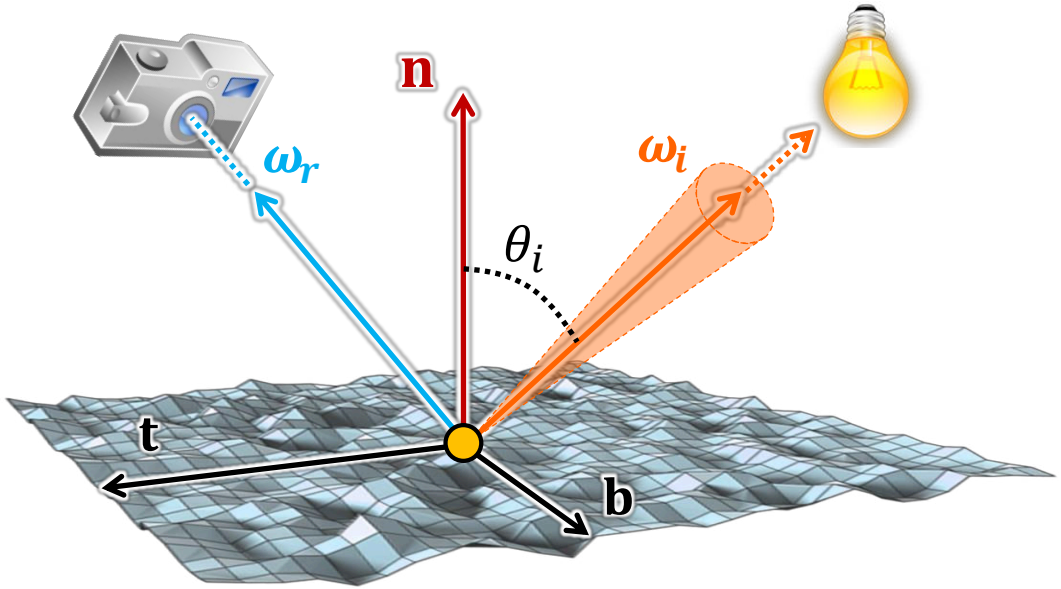


Radiometrie und BRDF

Reflexion an einer Oberfläche

- wie viel Licht wird in Richtung $\mathbf{v} = \boldsymbol{\omega_r}$ (z.B. zu einem Betrachter) reflektiert und erreicht diesen: Strahldichte $[\mathrm{W/m^2 sr}]$
- in Abhängigkeit des ankommenden Lichts aus einer Richtung $\mathbf{l} = \boldsymbol{\omega_i}$

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega_r}) = \frac{dL_r(\boldsymbol{x}, \boldsymbol{\omega_r})}{dE_i(\boldsymbol{x}, \boldsymbol{\omega_i})} = \frac{dL_r(\boldsymbol{x}, \boldsymbol{\omega_r})}{L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}}$$



Radiometrie und BRDF

Reflexion an einer Oberfläche

- Berechnung der differentiellen reflektierten Strahldichte aus der einfallenden Strahldichte mittels der BRDF

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega_r}) = \frac{dL_r(\boldsymbol{x}, \boldsymbol{\omega_r})}{L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}}$$

$$dL_{r\ aus\ \boldsymbol{\omega_i}}(\boldsymbol{x}, \boldsymbol{\omega_r}) = f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega_r}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

- Reflexionsintegral: berechne reflektierte Strahldichte in Richtung $\boldsymbol{\omega_r}$ durch Integration über alle einfallende Strahldichte

$$L_r(\boldsymbol{x}, \boldsymbol{\omega_r}) = \int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega_r}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

Radiometrie und BRDF

Eigenschaften einer physikalisch-plausiblen BRDF

- Wertebereich ist nicht-negativ, d.h.

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega_r}) \in [0; \infty)$$

  - 0 (vollständige Absorption), ∞ (perfekte Spiegelung, Deltafunktion)

- Helmholtz-Reziprozität (Umkehrbarkeit des Lichtwegs) gilt:

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega_r}) = f_r(\boldsymbol{\omega_r}, \boldsymbol{x}, \boldsymbol{\omega_i})$$

  - ergibt sich aus den Maxwellschen Gleichungen (gilt aber eigentlich nur unter speziellen Voraussetzungen, keine Details!)

- Energieerhaltung: nicht mehr Energie reflektieren als eintrifft

$$\int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega_r}) \cos\theta_r \, d\boldsymbol{\omega_r} \leq 1 \quad \forall \boldsymbol{x}, \boldsymbol{\omega_i}$$

Die Rendering Gleichung

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

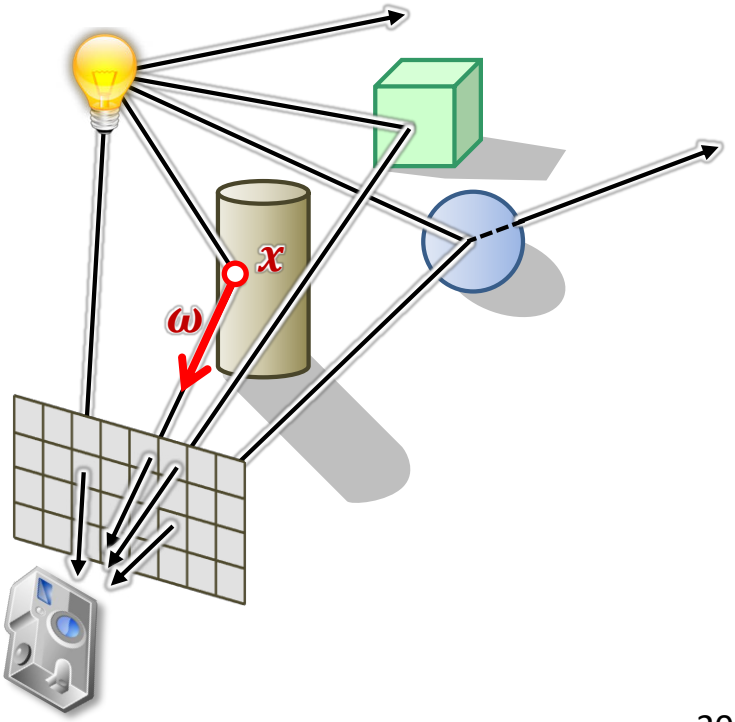
Demo: https://wwwtyro.net/2018/02/25/caffeine.html [60]

Rendering Gleichung

Beschreibt die Lichtverteilung in einer Szene [Kajiya86]

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

- L Strahldichte $[\mathrm{W/m^2 sr}]$
- L_e Emissionsterm $[\mathrm{W/m^2 sr}]$
- Abhängigkeit von der Wellenlänge (hier nicht dargestellt)
- Integrationsbereich:
  Ω^+ positive Hemisphäre (nur Reflexion)
  Ω alle Richtungen (inkl. Transmission)

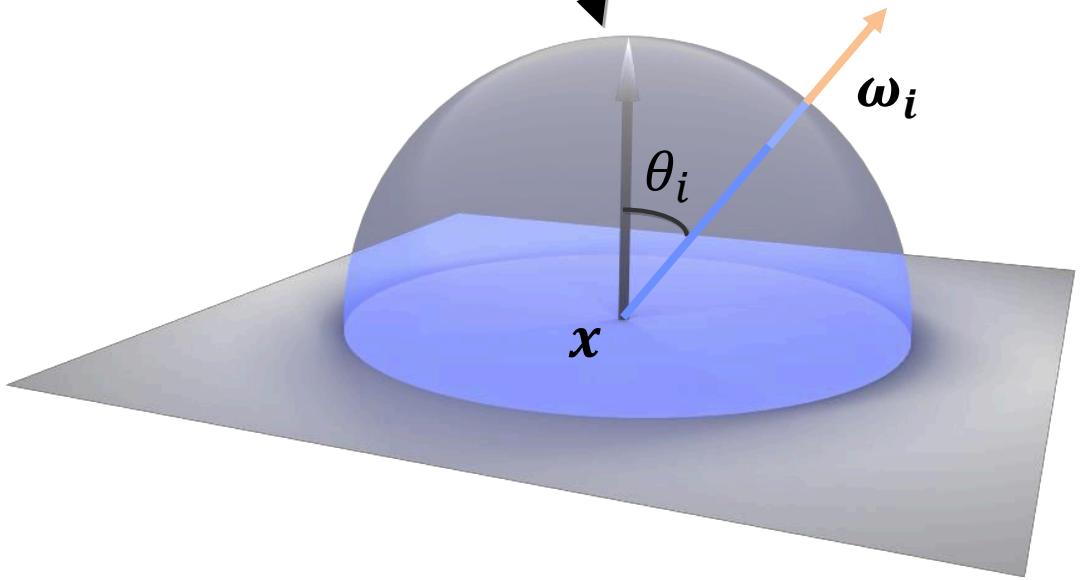


Rendering Gleichung

Einfallsrichtung

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

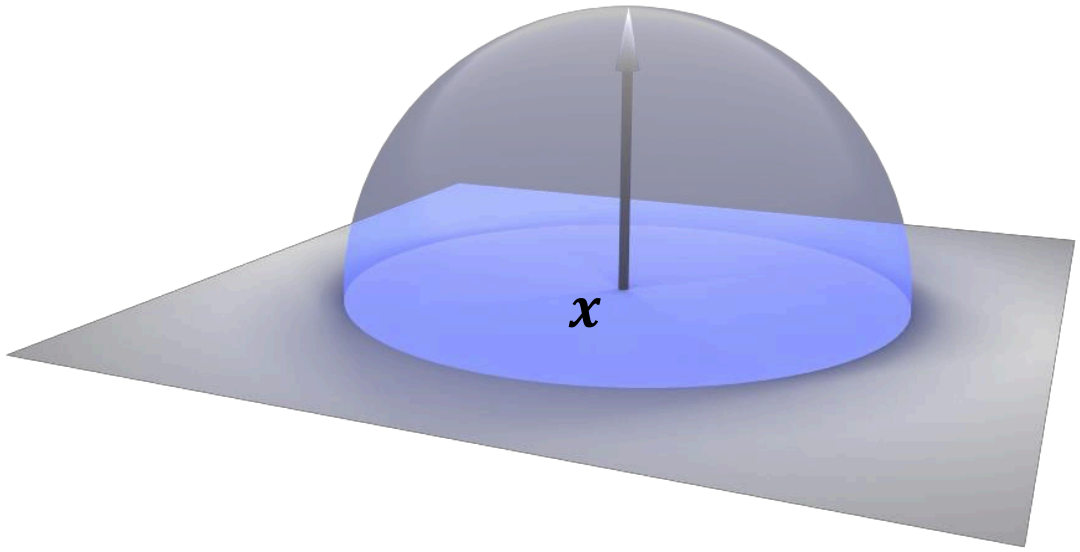
Integration über die positive Hemisphäre:
alle Richtungen aus denen Licht einfällt



Rendering Gleichung

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

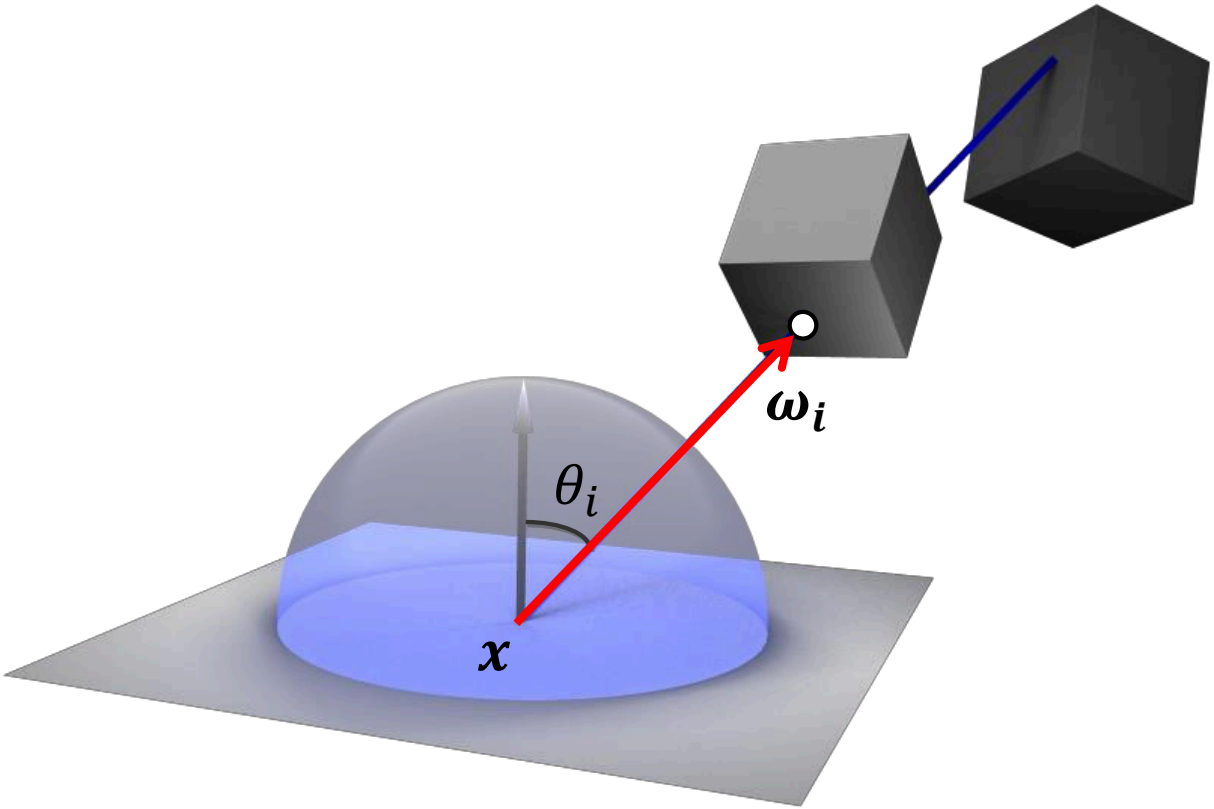
Reflektanzverteilungsfunktion:
Wie viel Licht aus Richtung $\boldsymbol{\omega_i}$
wird in Richtung $\boldsymbol{\omega}$ reflektiert



Rendering Gleichung

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

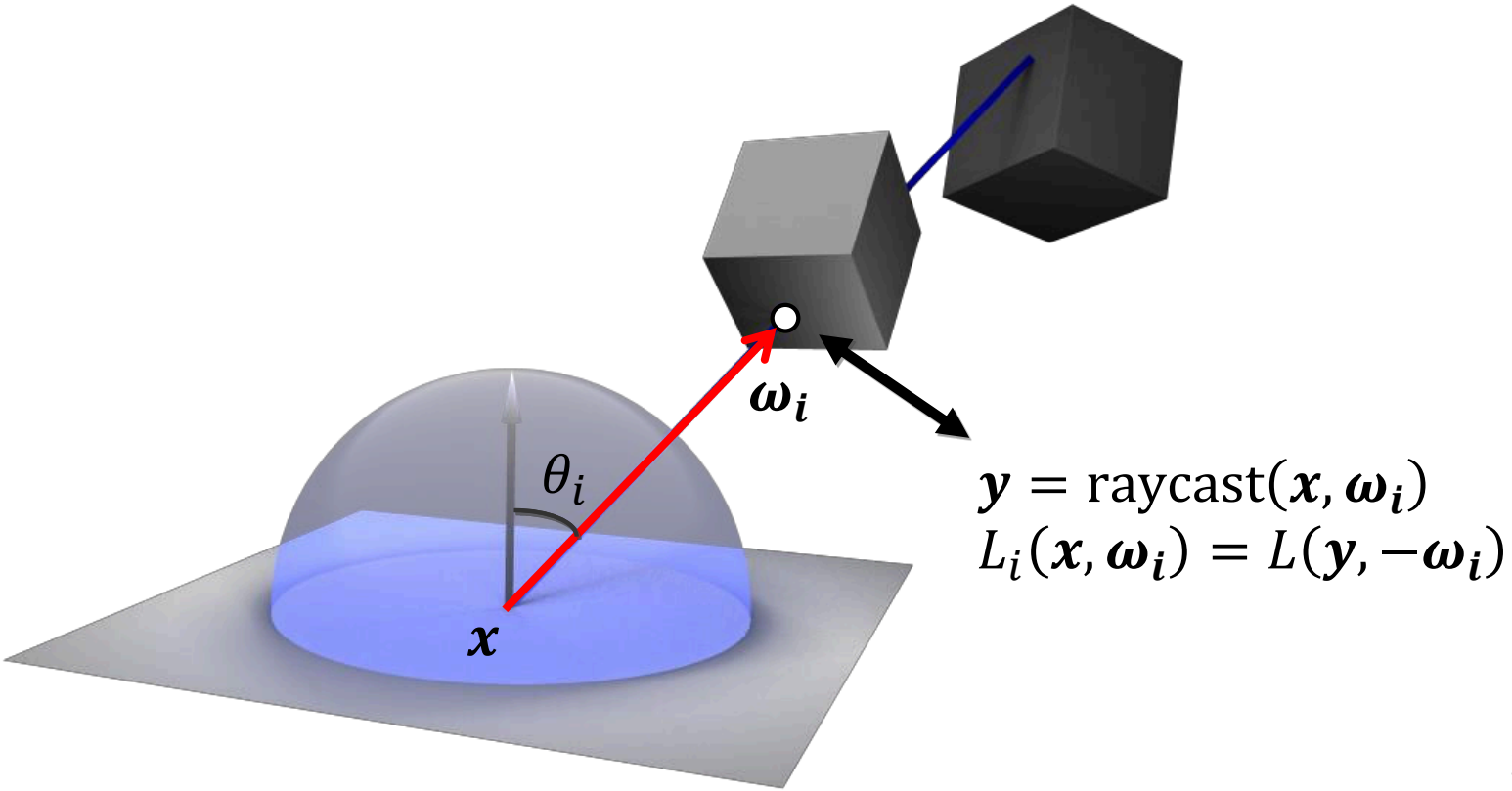
einfallendes Licht



Rendering Gleichung

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) \boxed{L(\boldsymbol{y}, -\boldsymbol{\omega_i})} \cos\theta_i \, d\boldsymbol{\omega_i}$$

ausgehendes Licht



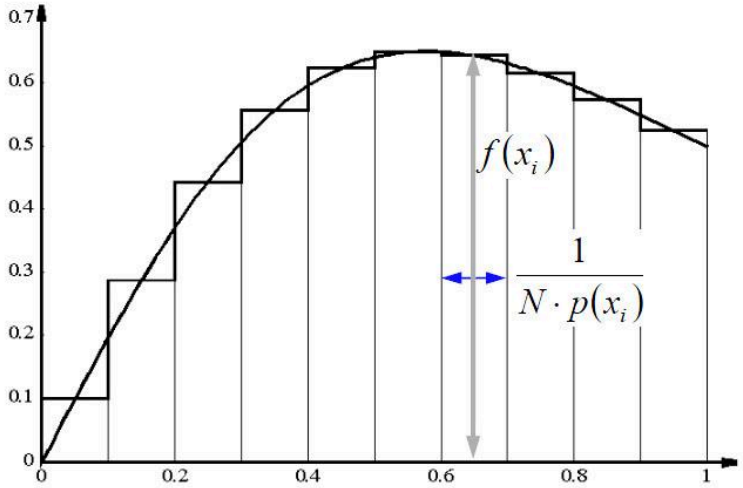
Monte Carlo-Integration

- Grundidee der Monte Carlo-Integration (vereinfacht!):
  wir können den Wert eines Integrals näherungsweise berechnen indem wir den Integranden an N zufälligen uniform-verteilten Stellen $x_i \in [a; b]$ auswerten

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i)$$

  - die Näherung/Schätzung wird besser, wenn N größer wird

- vergleiche Riemann-Integral:

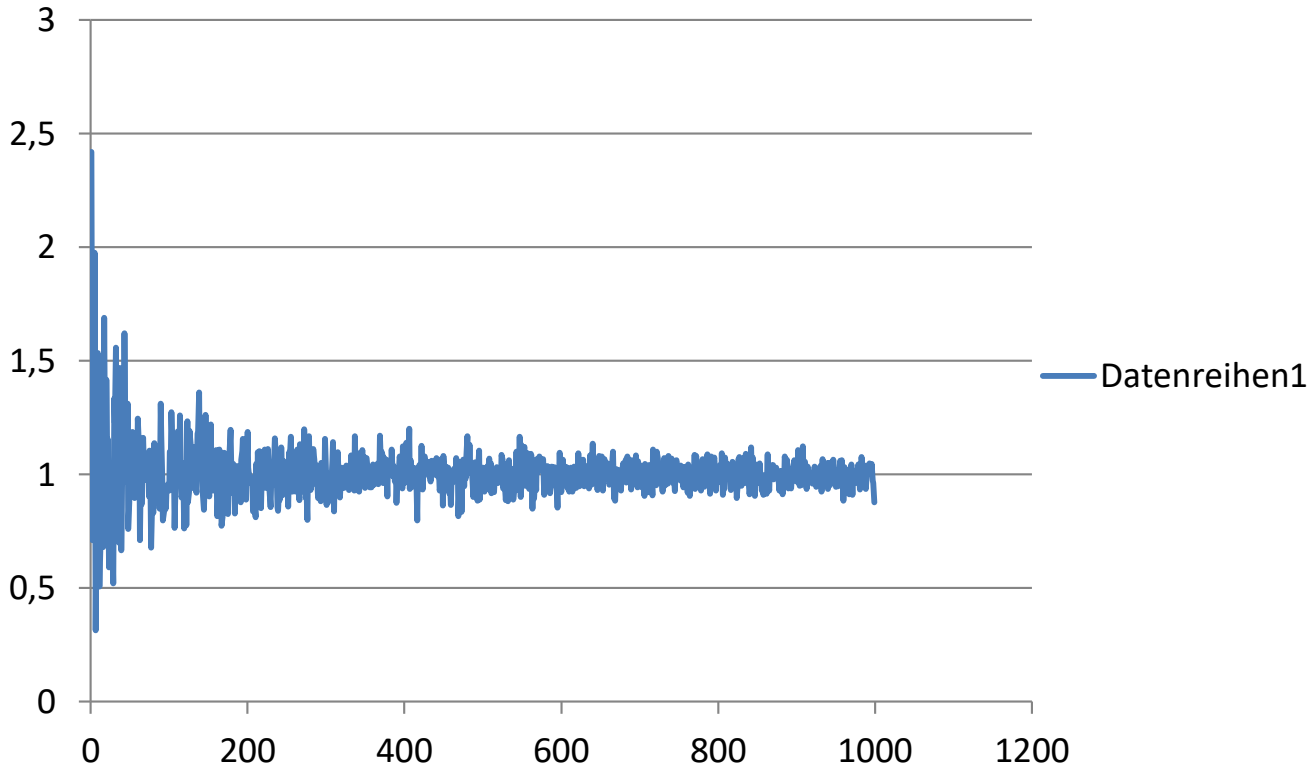


Monte Carlo-Integration

Beispiel

- berechne $I = \int_0^1 5x^4 dx = 1$

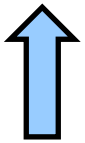
- naïve Monte Carlo-Integration mit $N = 1 \ldots 1000$



Monte Carlo-Rendering

- um den Wert des Integrals zu berechnen verwendet Distributed Raytracing (und verwandte Verfahren) die Monte Carlo-Integration
- im einfachsten Fall (wie eben gezeigt) bedeutet das: berechne den Wert des Integranden für (viele) zufällige Richtungen $\boldsymbol{\omega_i} \in \Omega^+$ und mittle

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \frac{2\pi}{N} \sum_{i=1}^{N} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i$$



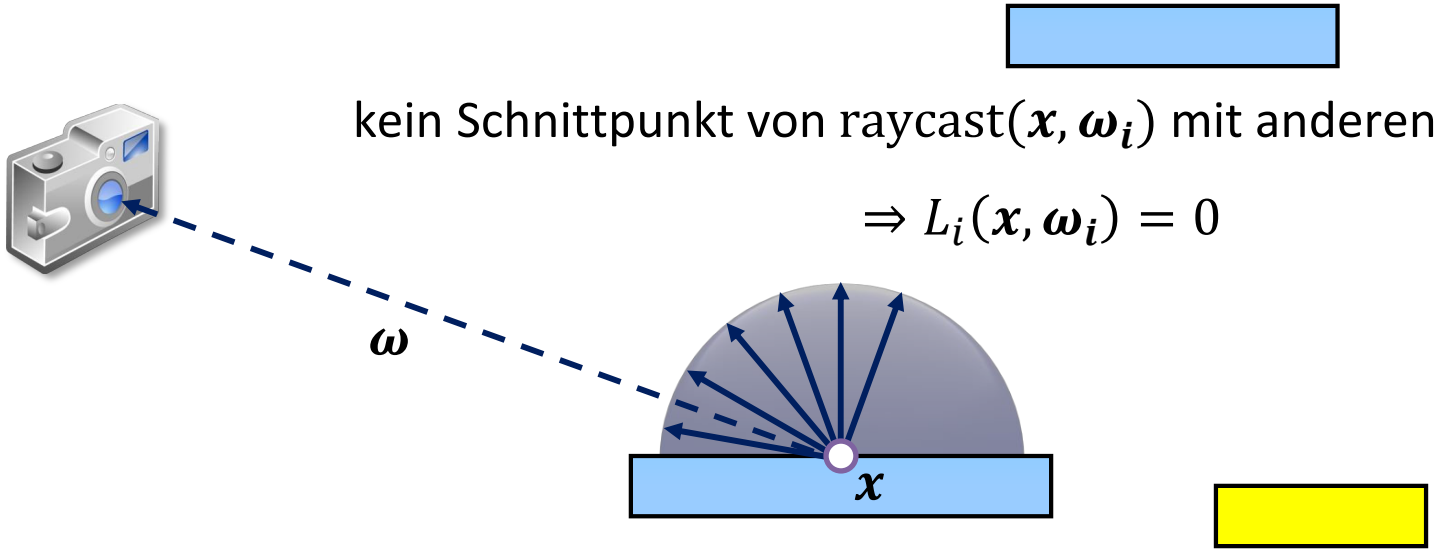
$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \int_{\Omega^+} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i \, d\boldsymbol{\omega_i}$$

- der Faktor 2π entspricht dem Raumwinkel der Hemisphäre

Distributed Raytracing

- d.h. wie Whitted-style Raytracing findet auch hier Rekursion statt, aber für jeden Schritt berechnen wir das (hemi-)sphärische Integral und nicht nur einzelne *ausgewählte* Sekundärstrahlen

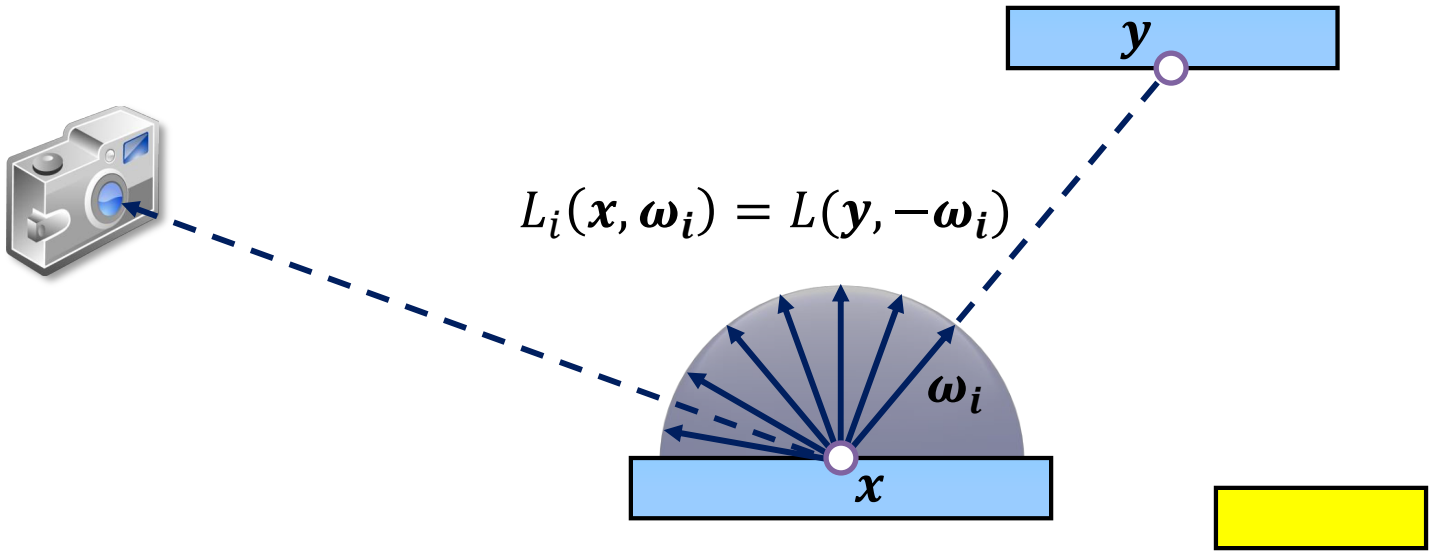
$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \frac{2\pi}{N} \sum_{i=1}^{N} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i$$



Distributed Raytracing

- d.h. wie Whitted-style Raytracing findet auch hier Rekursion statt, aber für jeden Schritt berechnen wir das (hemi-)sphärische Integral und nicht nur einzelne *ausgewählte* Sekundärstrahlen

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \frac{2\pi}{N} \sum_{i=1}^{N} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i$$

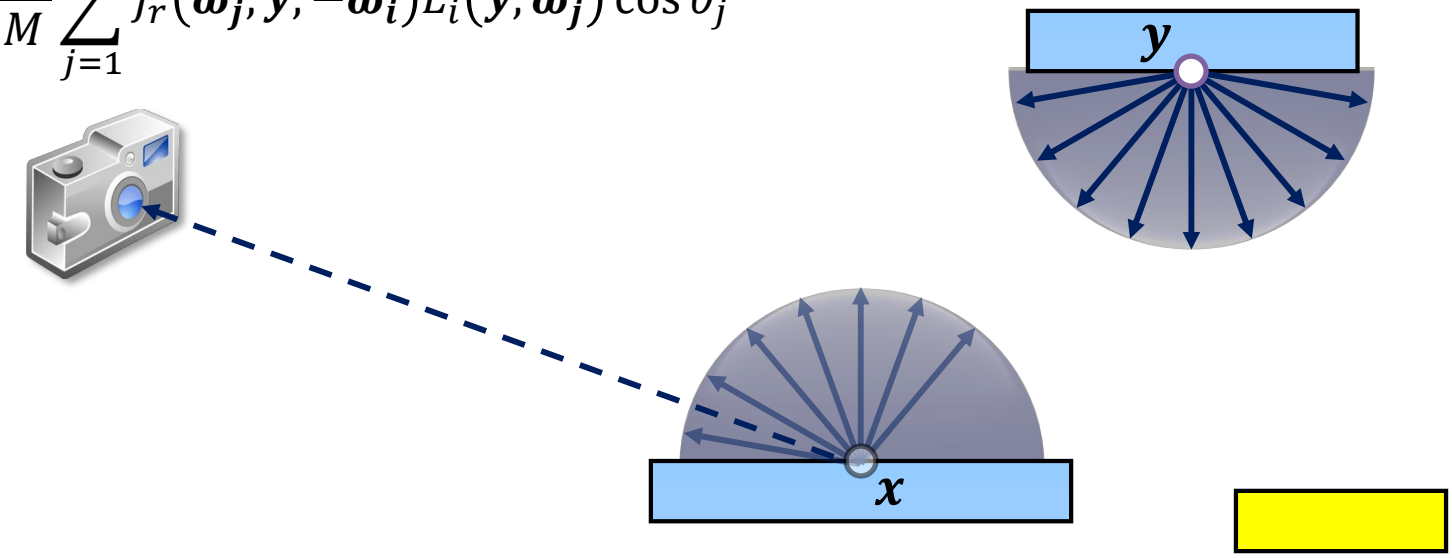


Distributed Raytracing

- d.h. wie Whitted-style Raytracing findet auch hier Rekursion statt, aber für jeden Schritt berechnen wir das (hemi-)sphärische Integral und nicht nur einzelne *ausgewählte* Sekundärstrahlen

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \frac{2\pi}{N} \sum_{i=1}^{N} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i$$

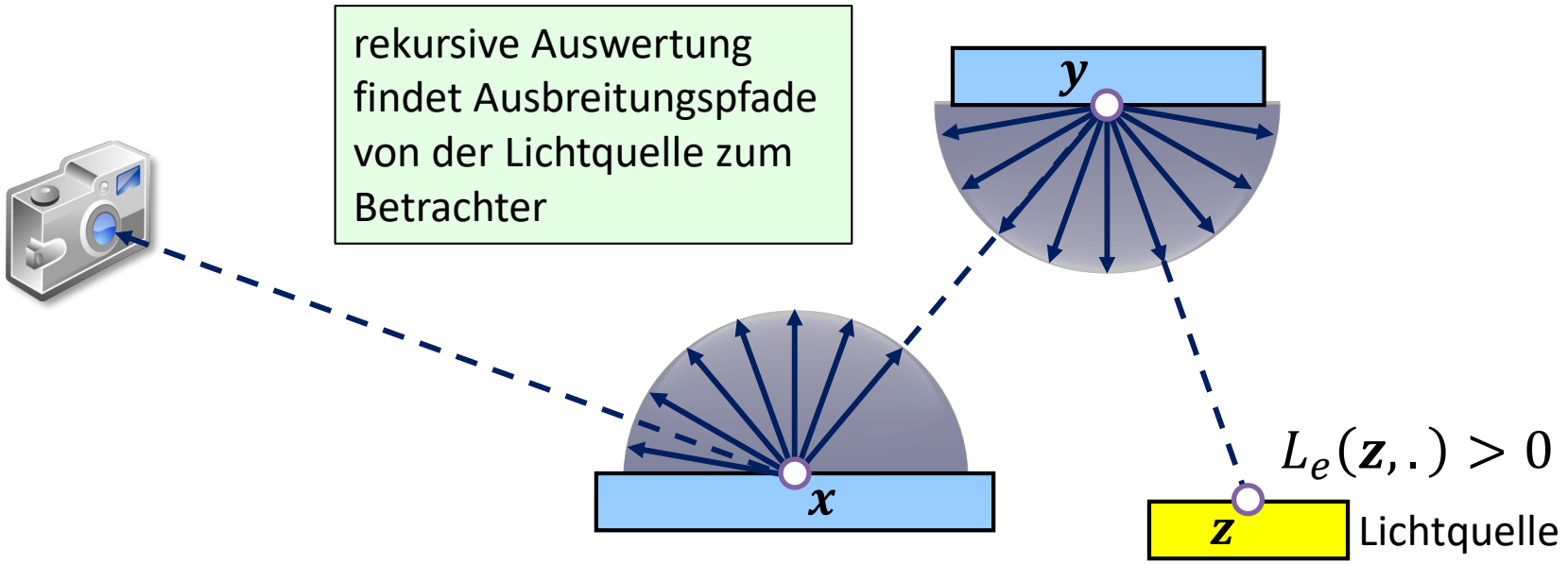
$$L(\boldsymbol{y}, -\boldsymbol{\omega_i}) = L_e(\boldsymbol{y}, -\boldsymbol{\omega_i}) +$$
$$+ \frac{2\pi}{M} \sum_{j=1}^{M} f_r(\boldsymbol{\omega_j}, \boldsymbol{y}, -\boldsymbol{\omega_i}) L_i(\boldsymbol{y}, \boldsymbol{\omega_j}) \cos\theta_j$$



Distributed Raytracing

- d.h. wie Whitted-style Raytracing findet auch hier Rekursion statt, aber für jeden Schritt berechnen wir das (hemi-)sphärische Integral und nicht nur einzelne *ausgewählte* Sekundärstrahlen

$$L(\boldsymbol{x}, \boldsymbol{\omega}) = L_e(\boldsymbol{x}, \boldsymbol{\omega}) + \frac{2\pi}{N} \sum_{i=1}^{N} f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) L_i(\boldsymbol{x}, \boldsymbol{\omega_i}) \cos\theta_i$$



Distributed Raytracing

Weitere Schritte / Umformulierungen (Kurzfassung)

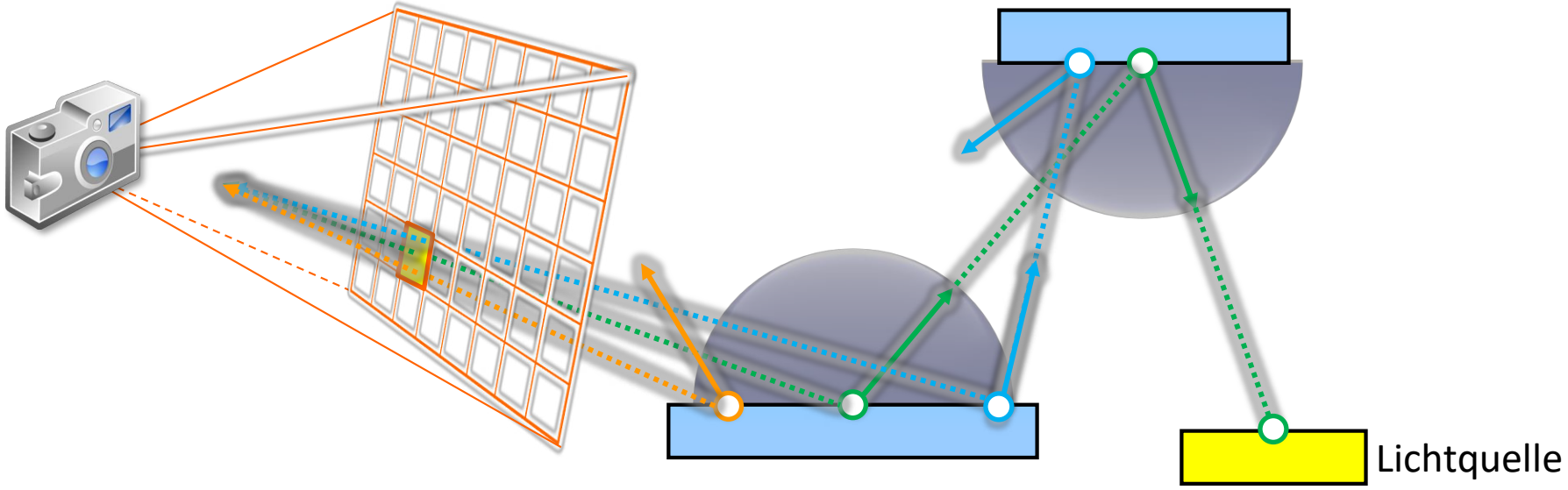
- Phong-Beleuchtungsmodell als BRDF (optional mit Energieerhaltung):

$$f_r(\boldsymbol{\omega_i}, \boldsymbol{x}, \boldsymbol{\omega}) = k_d + k_s \cdot \frac{n+2}{2\pi} \cdot \left(\left((2\mathbf{N}(\mathbf{N} \cdot \boldsymbol{\omega_i}) - \boldsymbol{\omega_i}) \cdot \boldsymbol{\omega} \right)^+ \right)^n / (\mathbf{N} \cdot \boldsymbol{\omega_i})^+$$

- zufällige Richtungen erzeugen (in Kugelkoordinaten)
  - $\theta = \arccos(1 - 2\xi_1)$ und $\phi = 2\pi\xi_2$ mit zwei gleichverteilten Zufallszahlen $\xi_1, \xi_2 \in [0; 1)$
  - $\boldsymbol{\omega_i} = (\sin\theta\cos\phi\,, \sin\theta\sin\phi\,, \cos\theta)$
- damit rekursiv berechnen, z.B. bis zu einer bestimmten Rekursionstiefe
  - z.B. $N = 64$ (für größere Rekursionstiefe: kleinere N bzw. M wählen)
  - „Russisches Roulette“ mit Pfaden

Path Tracing

- verwandt mit Distributed Raytracing und praxisrelevanter:
  nur jeweils ein weiterer Strahl, dafür mehrere Pfade pro Pixel
- führt zu mehr abgetasteten Richtungen bei Flächen nahe der Kamera,
  und geringerer Abtastung bei weiter entfernten (meist sinnvoll!)



Path Tracing und andere Ansätze

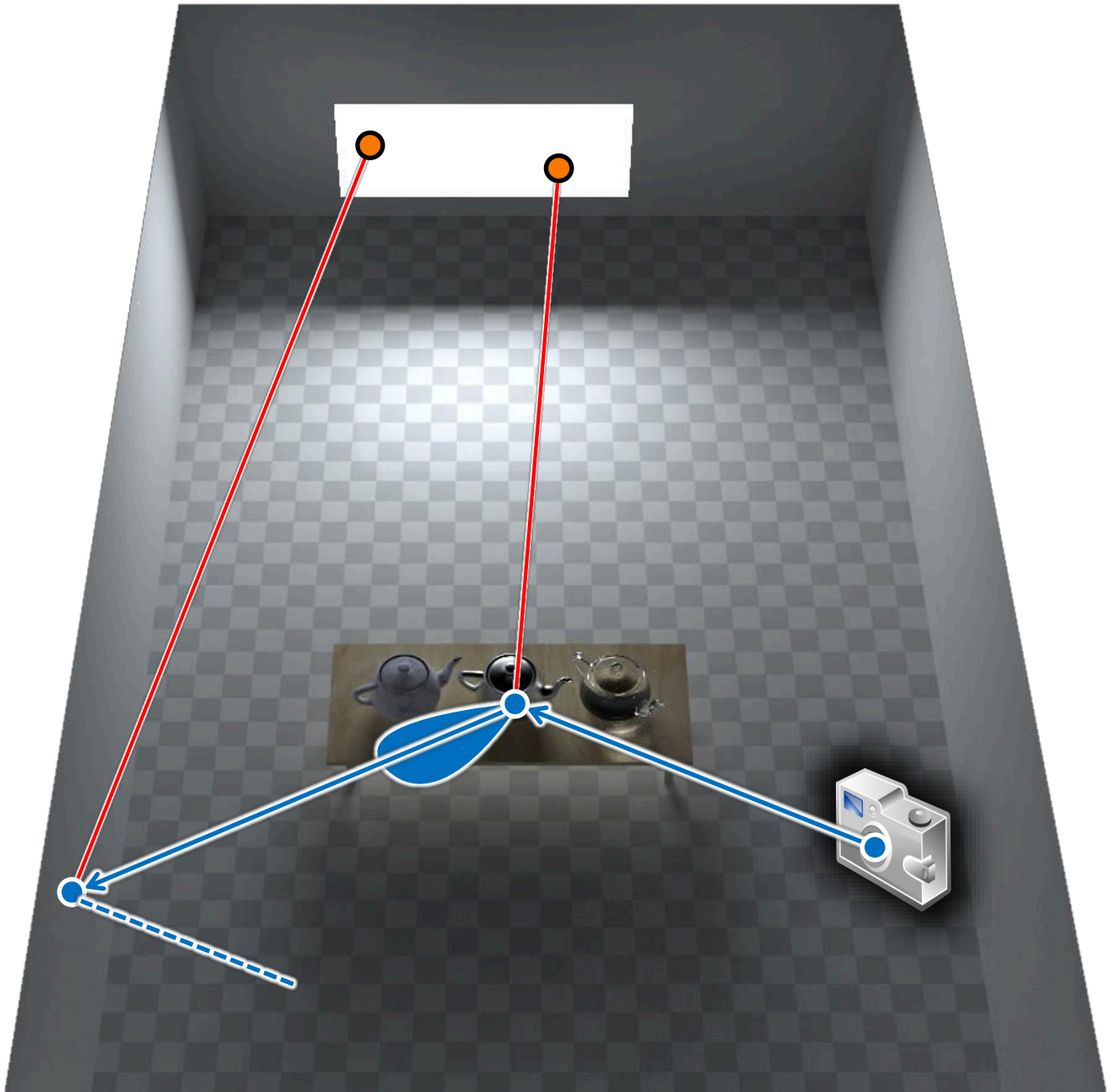
Konvergenz und schwierige Fälle

Pfade pro Pixel
$N = 4$

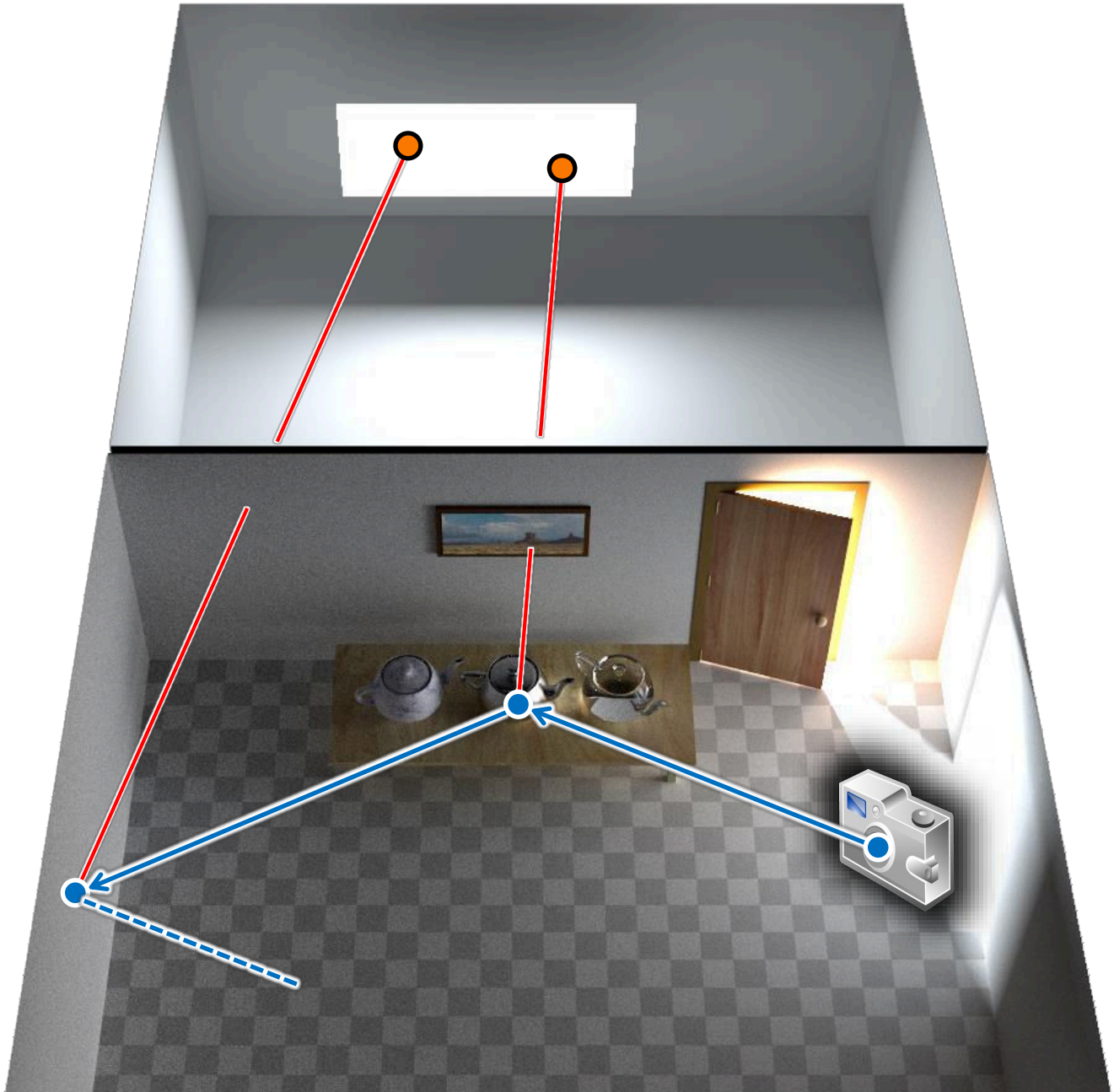
Fehler $\propto {}^{1}/_{\sqrt{N}}$

Pfadgenerierung: Path Tracing

[Kajiya86]



Pfadgenerierung: Problem - Verdeckung



Forschung: Pfadkonstruktionsstrategien

Herausforderungen bei der Pfadkonstruktion, Ziel: $p(\boldsymbol{X}) \propto f(\boldsymbol{X})$

Streuung $f_s(\boldsymbol{x}_{i-1} \rightarrow \boldsymbol{x}_i \leftrightarrow \boldsymbol{x}_{i+1}) = 0$

Verdeckung $G(\boldsymbol{x}_{i-1} \leftrightarrow \boldsymbol{x}_i) = 0$

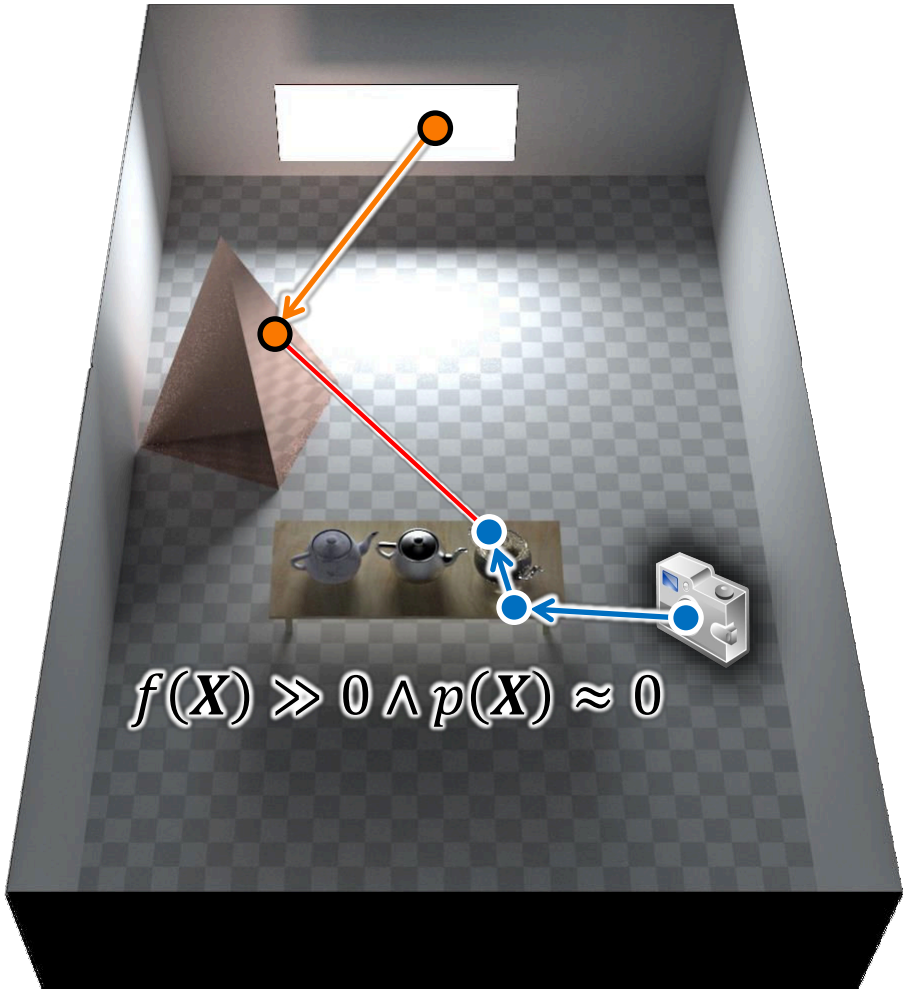
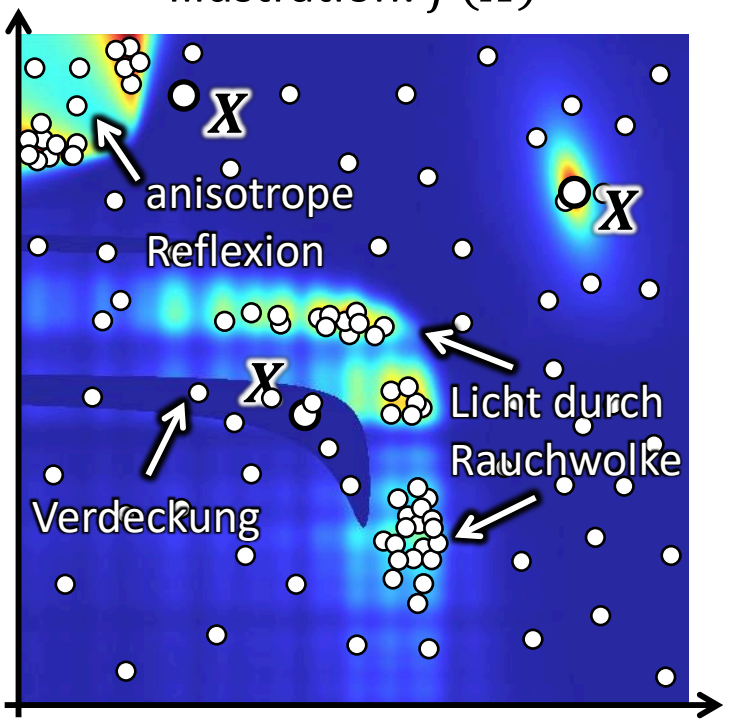
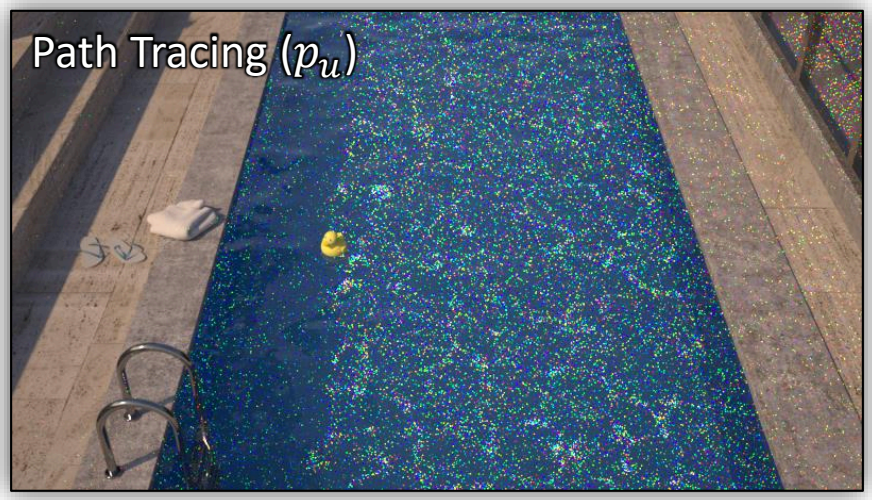


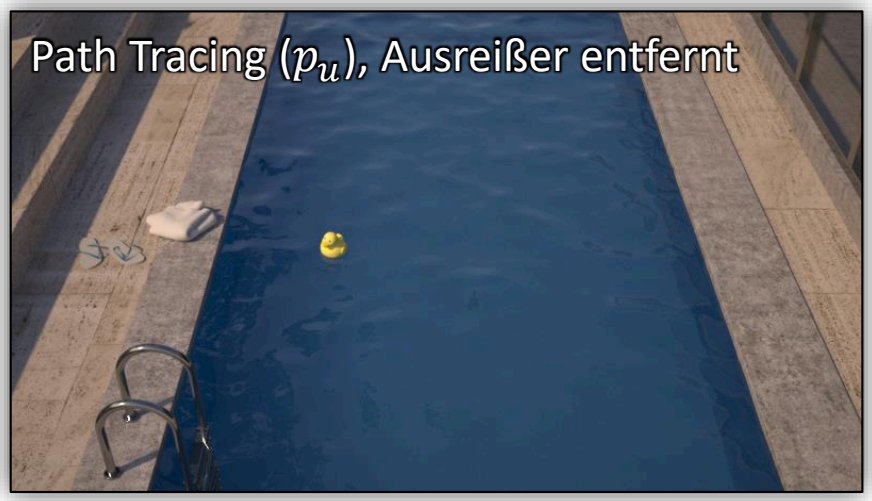
Illustration: $f(\boldsymbol{X})$



Selective Guided Sampling

Beispiel: 30 Minuten Rechenzeit, 104.000 Lernpfade (ca. 310 MB)



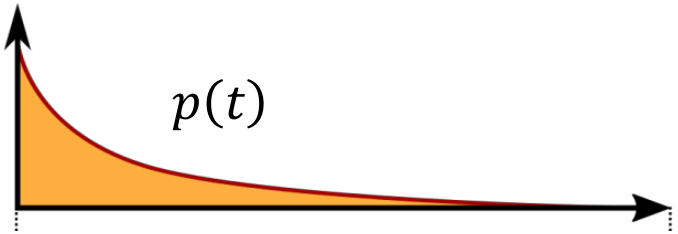


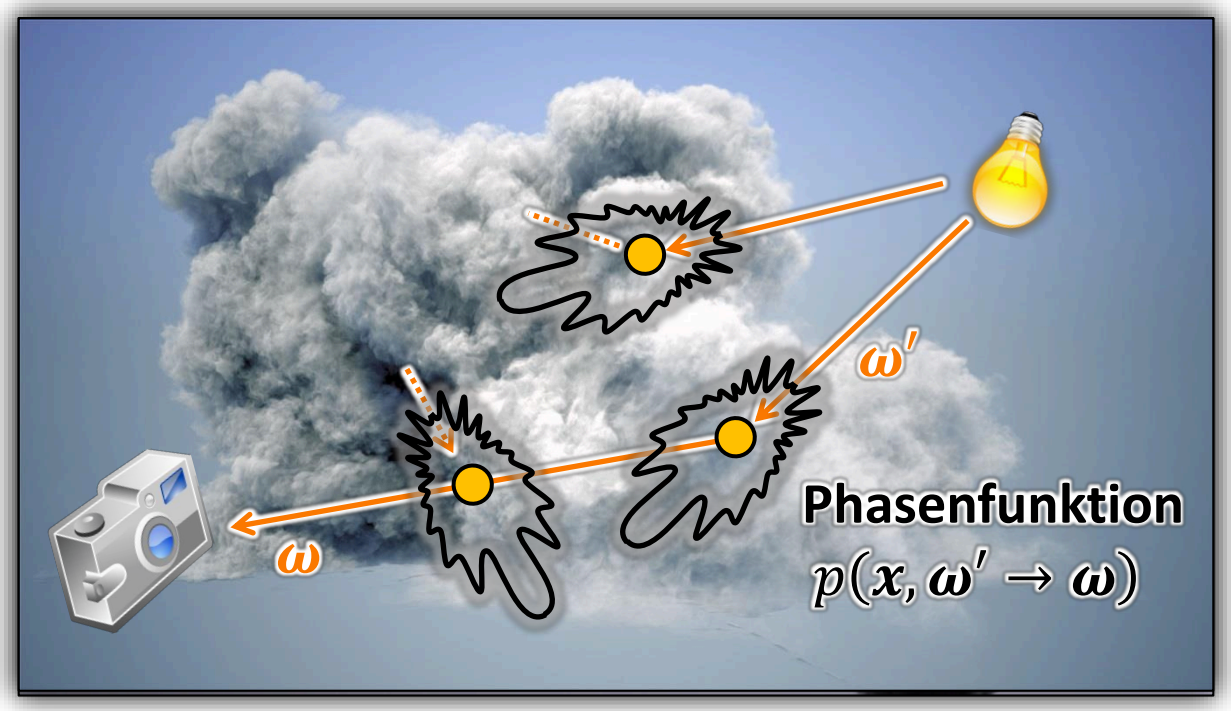
Selective Guided Sampling with Complete Light Transport Paths
Reibold, Hanika, Jung, Dachsbacher, *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 2018 [43]

Forschung: Modellierung realistischer Materialien

Partizipierende Medien

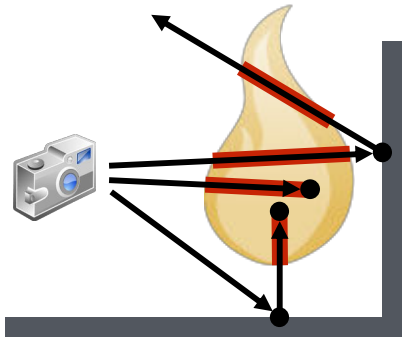
- Streu- und Absorptions-koeffizienten $\sigma_s(\boldsymbol{x})$, $\sigma_a(\boldsymbol{x})$ $[\mathrm{m}^{-1}]$
- erfordert Erweiterung der Rendering Gleichung

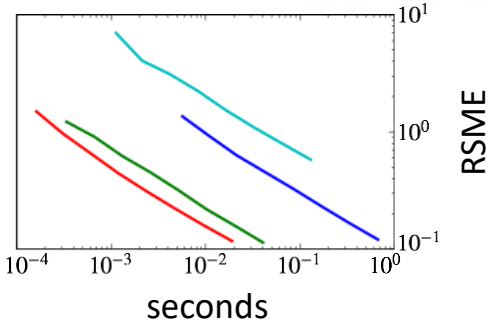




Forschung: Rendering, komplexer Lichttransport

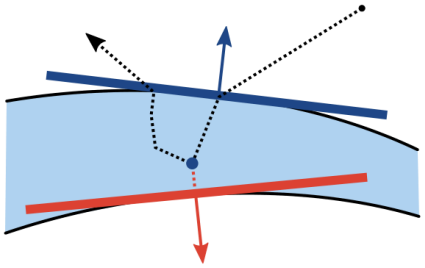
War for the Planet of the Apes [44]

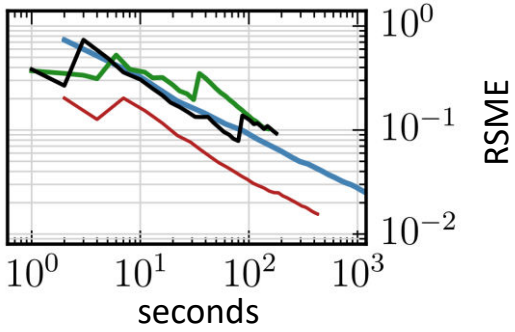




Line Integration for Rendering Heterogeneous Emissive Volumes
Simon, Hanika, Zirr, Dachsbacher,
Computer Graphics Forum (Proc. EG Symp. on Rendering), 2017

Alita: Battle Angel [45]

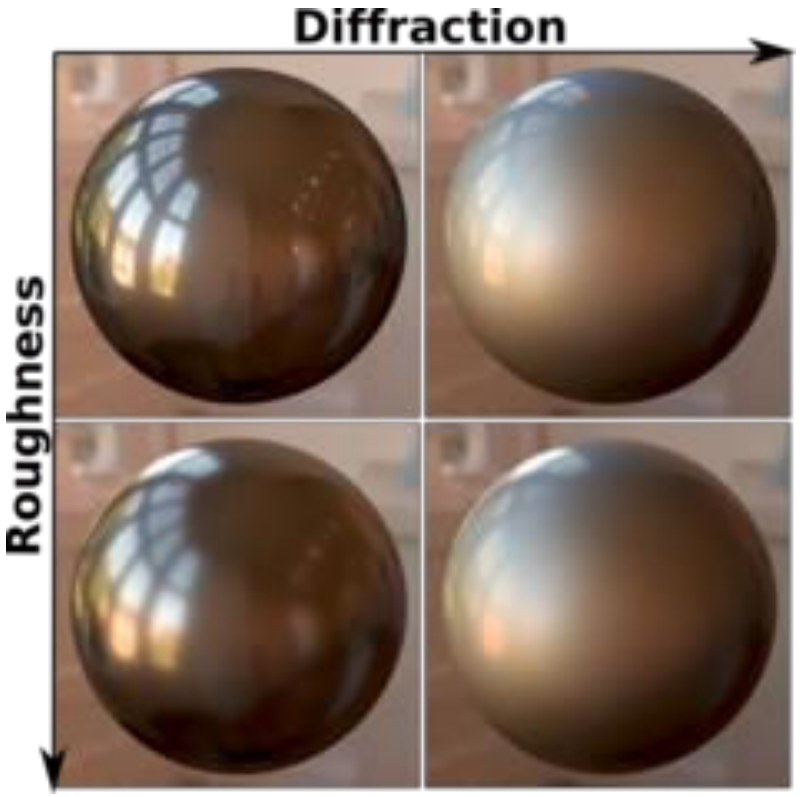


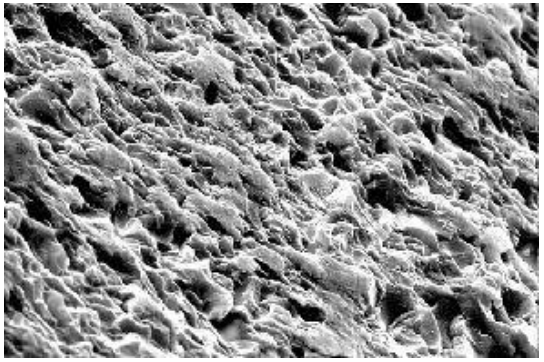


Improving the Dwivedi Sampling Scheme
Meng, Hanika, Dachsbacher
Computer Graphics Forum (Proc. EG Symp. on Rendering), 2016

Forschung: Modellierung realistischer Materialien

Beugungseffekte



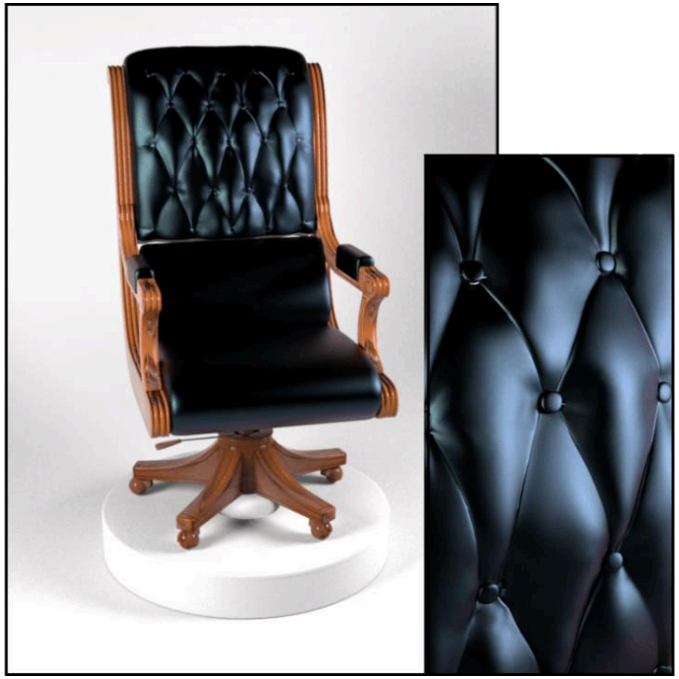


A Two-Scale Microfacet Reflectance Model Combining Reflection And Diffraction
Holzschuch, Pacanowski
ACM Transactions on Graphics (Proc. of SIGGRAPH), 2017 [47]

Forschung: Modellierung realistischer Materialien

- Interferenzen an dünnen Filmen auf der Oberfläche der Mikrofacetten

Mikrofacetten

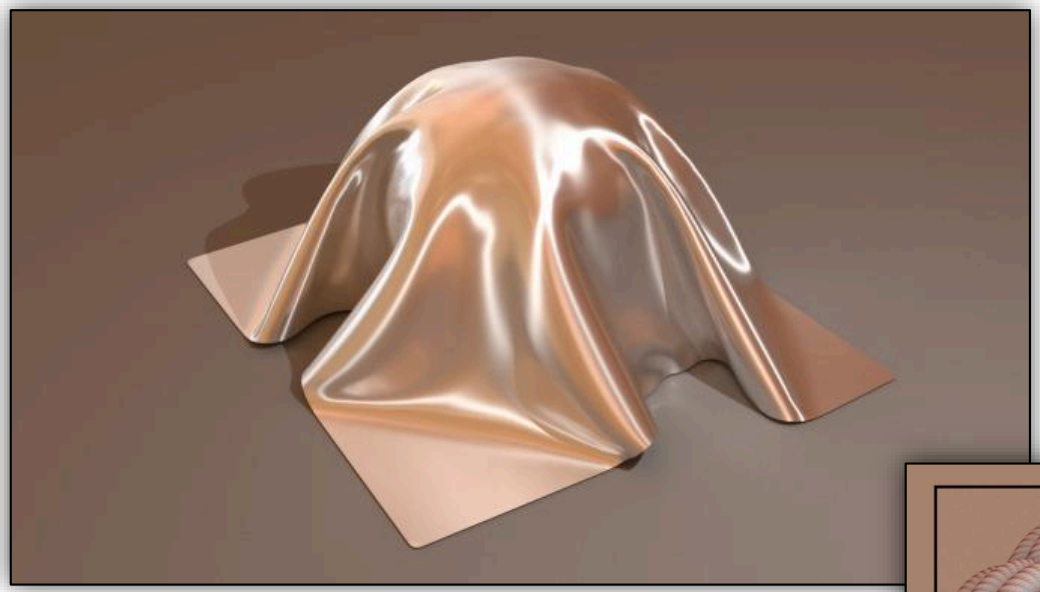


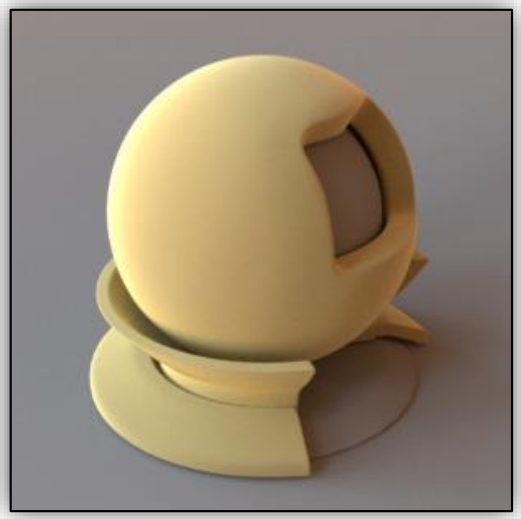
irisierende Mikrofacetten

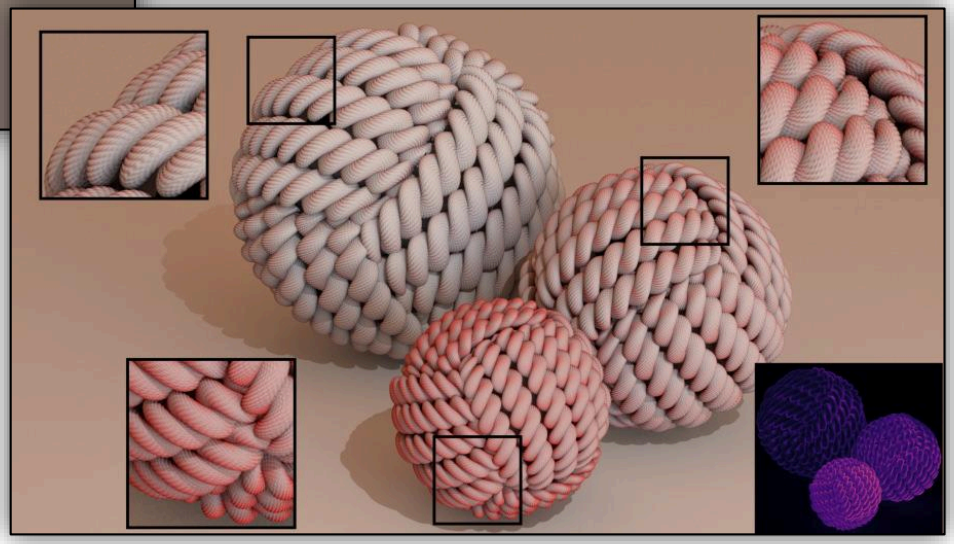
A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence
Belcour, Barla
ACM Transactions on Graphics (Proc. of SIGGRAPH), 2017 [48]

Forschung: Modellierung realistischer Materialien

- Anisotrope poröse Schichten (Micrograin BSDF-Modell)



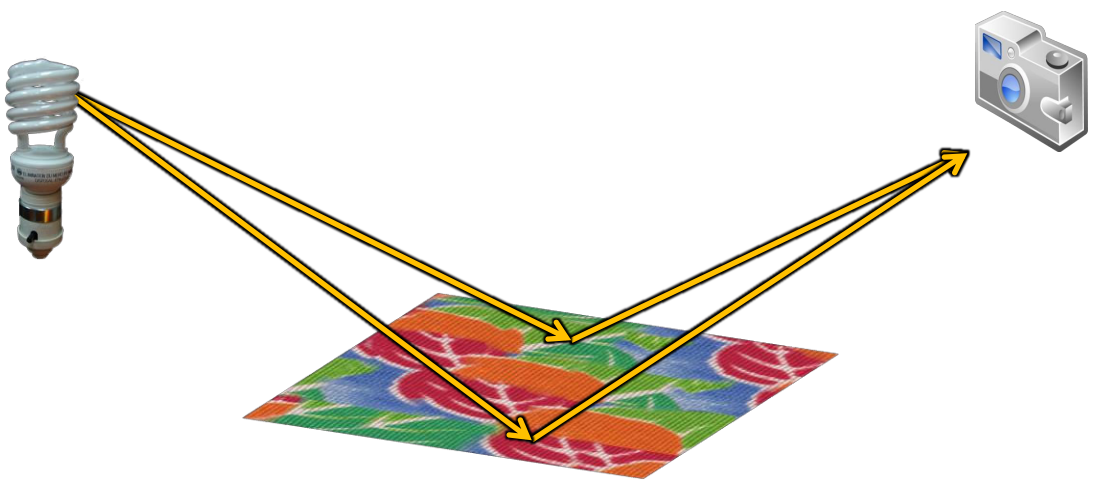


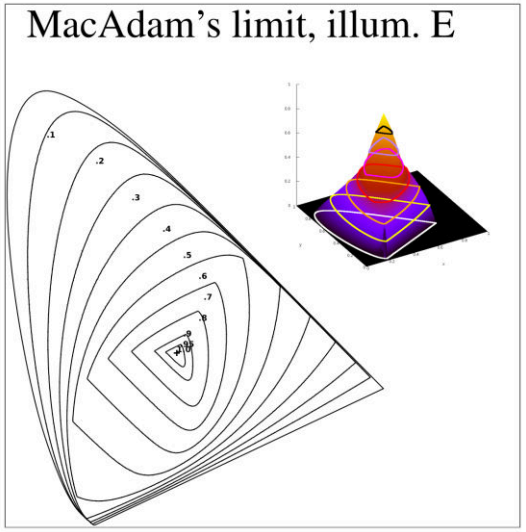


A Fully-correlated Anisotropic Micrograin BSDF Model, Lucas et al. *ACM Transactions on Graphics*, 2024

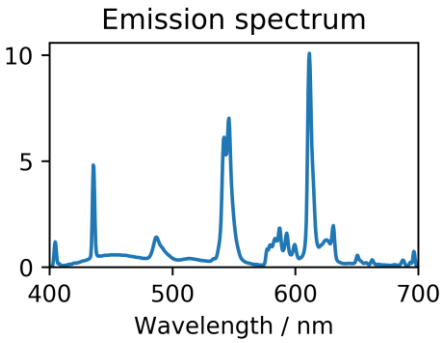
Spektrales Rendering

- verwende aufgelöste Spektren (z.B. 5nm Schritte) statt RGB-Vektoren
- Beobachtung: Reflektanzen sind entweder sehr gesättigt oder sehr hell

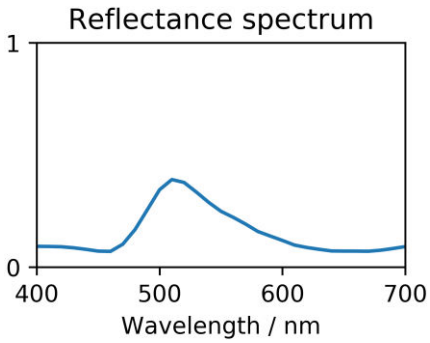




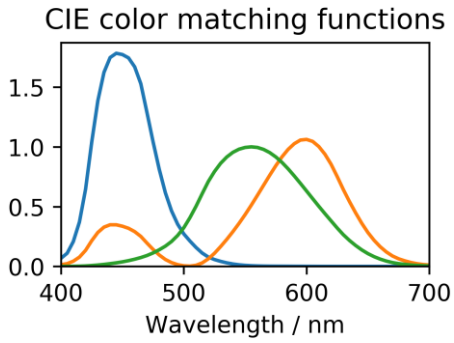
$$\int$$



$\cdot$



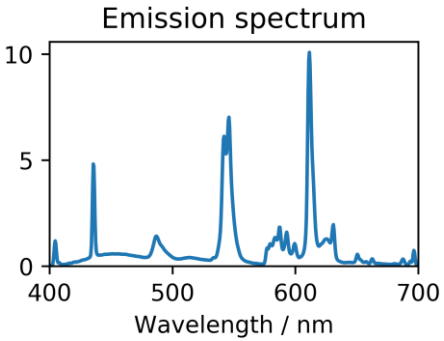
$\cdot$

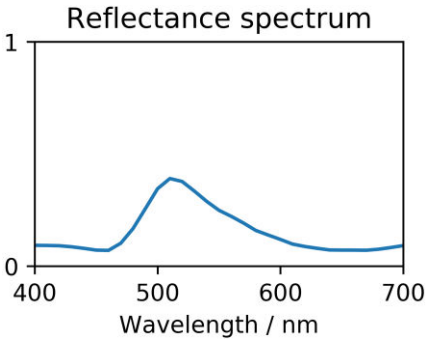


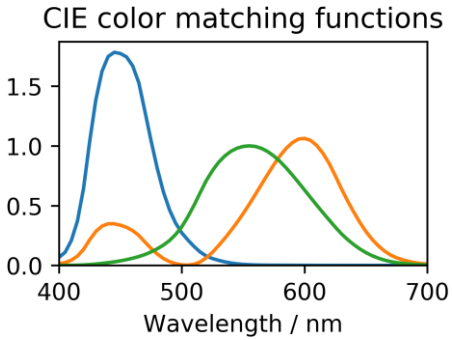
$$d\lambda$$

Spektrales Rendering

- warum spektrales Rendering?
  - exakte Reproduktion von Emission · Reflektanz, Dispersion, …
- was spricht gegen spektrales Rendering?
  - mehr Aufwand bei Licht-Material-Interaktionen
  - signifikant mehr Speicherbedarf von Texturen
- Spectral Upsampling-Methoden erzeugen ein Spektrum aus RGB-Werten
  - nützlich: bei Reflektanzspektren gilt: $\rho(\lambda) \leq 1$
  - natürliche Reflektanzspektren sind zudem glatt
    (noch stärkere Einschränkung im Vergleich zu $\rho(\lambda) \leq 1$)

$$\int$$ 

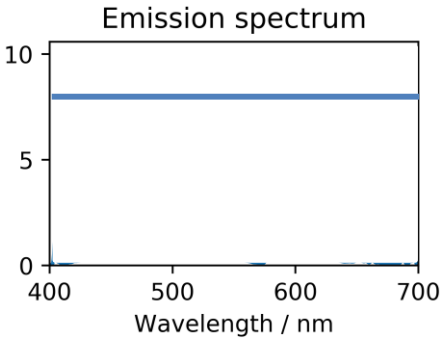
$\cdot$ 

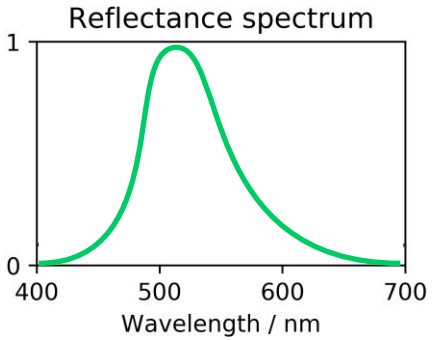
$\cdot$ 

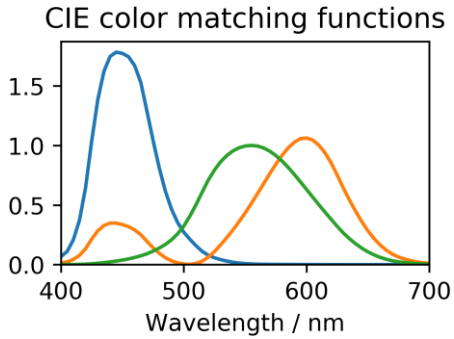
$\mathrm{d}\lambda$

Spektrales Rendering

Beispiel: Unterschied zwischen Tristimulus- und spektralem Rendering

$$\int$$ 

$\cdot$ 

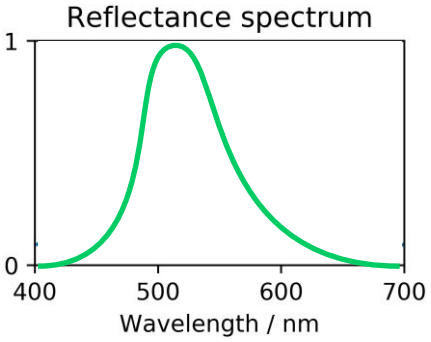
$\cdot$ 

$\mathrm{d}\lambda$

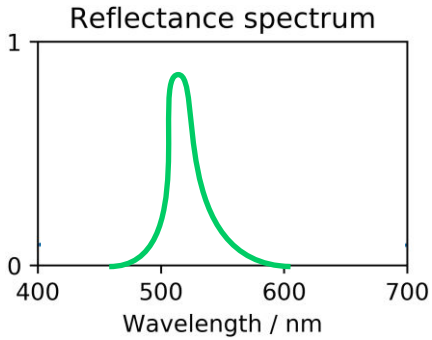
$$\Updownarrow$$

$$\mathrm{RGB} = (0,1,0)^{\mathrm{T}}$$

- bei n-facher Reflexion ($n > 1$) wird der Unterschied deutlich:
  - $(\mathrm{RGB})^n = (0,1,0)^{\mathrm{T}}$
  - aber:



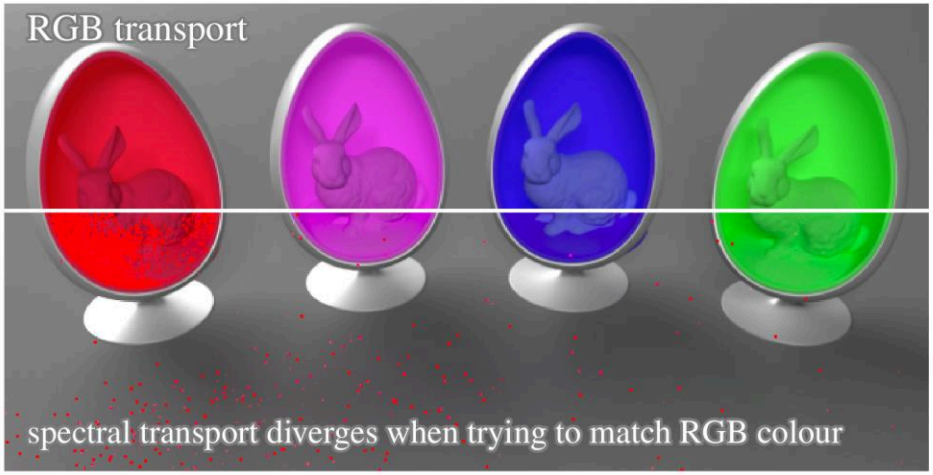
$\rho(\lambda)^n$ $\Rightarrow$



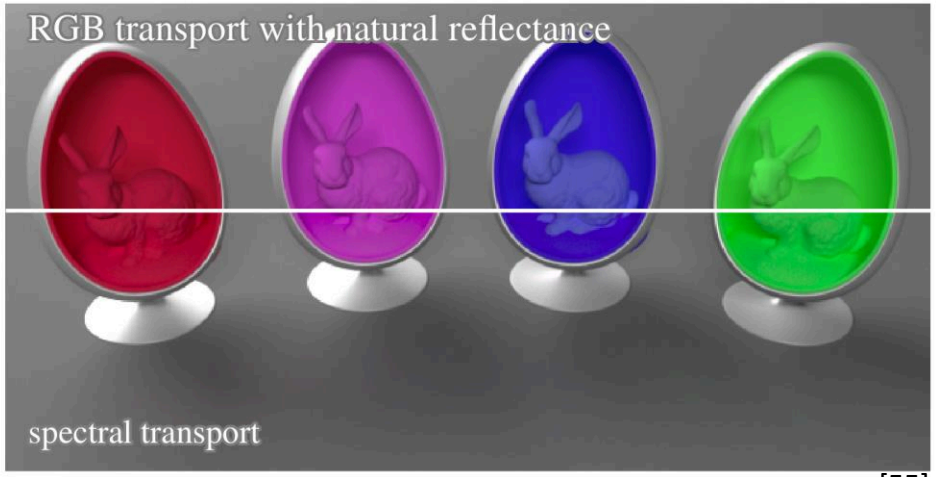
Spektrales Rendering

Beispiel: Unterschied zwischen Tristimulus- und spektralem Rendering

- bei n-facher Reflexion ($n > 1$) wird der Unterschied deutlich
- nicht für alle RGB-Werte existieren entsprechende Reflektanzspektren, übersättigte Spektren verletzten aber u.U. die Energieerhaltung
- durch Einschränkungen bei der Wahl der Tristimulus-Werte für Reflektanzen wird der Unterschied geringer



Reflektanzspektren übersättigt



Reflektanzspektren: glatt, $\rho(\lambda) \leq 1$ und so gewählt, dass „RGB=spektral“ [55]

Spektrales Rendering

- wie kommen in der Realität gesättigte Farben zustande? Fluoreszenz!

[50]

[51]

[52]

[53]

Spektrales Rendering

- wie kommen in der Realität gesättigte Farben zustande? Fluoreszenz!
- Idee: erweitere spektrales Upsampling um einen fluoreszierenden Anteil
  - verwendet, wenn Reflektanz alleine nicht genügt
- erfordert Lichttransportsimulation mit fluoreszierenden Materialien
  - größtenteils ungelöstes Problem ⇒ Forschungsthema

[54]

Bildreferenzen

- [1] https://www.amazon.de/Fundamentals-Computer-Graphics-English-Shirley-ebook/dp/B009TG9NIQ
- [2] https://pixabay.com/de/vectors/server-computergeh%C3%A4use-computer-pc-309012/
- [3] https://www.analogisdifferent.com/que-es-la-fotografia-estenopeica/
- [4] Interactive Ray Tracing of Arbitrary Implicits with SIMD Interval Arithmetc, Aaron Knoll, Younis Hijazi, Charles Hansen, Ingo Wald, Hans Hagen, IEEE 2007
- [5] Fast and Robust Ray Tracing of General Implicits on the GPU, Aaron Knoll, Younis Hijazi, Andrew Kensler, Mathias Schott, Charles Hansen, Hans Hagen, Scientitfic Computing and Imaging Institute, University of Utah. Technical Report No UUSCI-2007-014
- [6] Computer-Graphik II Ray Tracing, G. Zimmermann, Clausthal University 2007
- [7] Scalable Realistic Rendering with Many-Light Methods, Carsten Dachsbacher, Jaroslav Křivánek, Miloš Hašan, Adam Arbree, Bruce Walter, Jan Novák, EUROGRAPHICS 2013
- [8] https://www.mathaeser.de/mm/filmdetail/cars-3-evolution/44954000012PLXMQDD
- [9] https://de.wikisource.org/wiki/Underweysung_der_Messung,_mit_dem_Zirckel_und_Richtscheyt,_in_Linien,_Ebenen_unnd_gantzen_corporen
- [10] https://www.uni-due.de/~adg350u/Skripte/AnalytischeGeometrie.pdf
- [11] http://jameslanzon.com/website/resources/thesis.pdf
- [12] https://de.depositphotos.com/stock-photos/ausrufezeichen.html
- [13] http://realtimerendering.com/intersections.html
- [14] Making of "The Shaved Bumblebee" ,Till Nowak

Bildreferenzen

- [15] www.wikipedia.de
- [16] http://www.mitsuba-renderer.org/devblog/
- [17] https://www.cg.tuwien.ac.at/courses/Rendering/ShadingModels.pdf
- [18] https://zhuanlan.zhihu.com/p/138762969
- [19] Reflectometer for Measuring the Bidirectional Reflectance of Rough Surfaces, White, Saunders, Applied Optics 37(16), 1998
- [20] Microfacet-based normal mapping for robust Monte Carlo path tracing, Schüssler, Heitz, Hanika, Dachsbacher, ACM Transactions on Graphics (Proc. of SIGGRAPH Asia), 2017
- [21] M. Hullin, M. Fuchs, I. Ihrke, H.-P. Seidel, H. Lensch: Fluorescent Immersion Range Scanning (SIGGRAPH 2008)
- [22] Hullin et al., Direct Visualization of Real-World Light Transport (VMV 2008)
- [23] http://perso.univ-lyon1.fr/jean-claude.iehl/Public/educ/M1IMAGE/html/group__brdf.html
- [24] https://schrott2cash.at/altmetall/kupfer-blank-ii-kerze/
- [25] https://www.amazon.de/CHAOSE-Mehrfarben-Durchmesser-Kunststoffkugeln-Spielplatz/dp/B07H7CFM4S
- [26] http://pngimg.com/images/nature/sun
- [27] Ray Tracing, Pat Hanrahan, CS 348 B Lecture 2, Spring 2005
- [28] Ray Tracing, Prof. James O'Brien, University of California, Berkeley, V2015
- [29] https://de.wikipedia.org/wiki/Datei:Snellius-Brechungsgesetz_(gespiegelt).svg
- [30] https://de.wikipedia.org/wiki/Datei:Snells_law_wavefronts.gif

Bildreferenzen

- [31] http://www.dokspeicher.de/120501/
- [32] http://www.fizkapu.hu/fizfoto/fizfoto6.html
- [33] http://www.dieter-heidorn.de/Physik/VS/Optik/K04_Brechung/K04_Brechung.html
- [34] www.wikipedia.com
- [35] Lynch, David K., Livingston, William, 2001, Color and Light in Nature, 2. Edition
- [36] Distributed Ray Tracing Robert L. Cook Thomas Porter Loren Carpenter, Computer Graphics Volume 18, Number 3 July 1984
- [37] https://users.aalto.fi/~laines9/
- [38] [Cook et al. 1984]
- [39] Computer Graphics Forum (Proc. of Eurographics Symposium on Rendering), 34(4):87--97, June 2015
- [40] http://nestohuis.deviantart.com/art/Caustic-Chrome-Tut-55223982
- [41] https://www.researchgate.net/figure/Observed-black-and-calculated-Ca-i-l612221-with-logg-37-light-gray-red-online_fig3_235891954
- [42] https://jo.dreggn.org/path-tracing-in-production/
- [43] Selective Guided Sampling with Complete Light Transport Paths, Reibold, Hanika, Jung, Dachsbacher, ACM Transactions on Graphics (Proc. SIGGRAPH Asia), 2018
- [44] https://bargelheuser.de/how-bad-ape-sets-up-war
- [45] https://soranews24.com/2017/12/12/hollywoods-alita-battle-angel-live-action-anime-adaptation-keeps-heroines-gigantic-anime-eyes/

Bildreferenzen

- [46] Line Integration for Rendering Heterogeneous Emissive Volumes, Simon, Hanika, Zirr, Dachsbacher, Computer Graphics Forum (Proc. EG Symp. on Rendering), 2017
- [47] Improving the Dwivedi Sampling Scheme, Meng, Hanika, Dachsbacher, Computer Graphics Forum (Proc. EG Symp. on Rendering), 2016
- [48] A Two-Scale Microfacet Reflectance Model Combining Reflection And Diffraction, Holzschuch, Pacanowski, ACM Transactions on Graphics (Proc. of SIGGRAPH), 2017
- [49] A Practical Extension to Microfacet Theory for the Modeling of Varying Iridescence, Belcour, Barla, ACM Transactions on Graphics (Proc. of SIGGRAPH), 2017
- [50] https://aofmoka.com/products/blacklight-psychedelic-t-shirt-glow-cats-cats-cats
- [51] https://www.fashiola.at/herren/kleidung/tops-shirts/s_neon/
- [52] https://www.yy-99.xyz/products.aspx?cname=neon+running+shoes+womens&cid=6
- [53] https://shopee.ph/1-Liter-Nalgene-Bottle-%28Brand-new-and-original%29-i.49623366.4643869031
- [54] https://cg.ivd.kit.edu/uplifting.php
- [55] https://cg.ivd.kit.edu/spectrum.php
- [56] http://graphics.ucsd.edu/~henrik/images/raytrace.html
- [57] https://www.kevinbeason.com/scs/pane/
- [58] The Mathematics of Visual Computing, Luiz Velho, IMPA – Instituto de Matemática Pura e Aplicada
- [59] Bùi Tường Phong, Illumination for Computer Generated Pictures. Comm. of the ACM 18, 1975
- [60] Demo: https://wwwtyro.net/2018/02/25/caffeine.html