h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Visual Computing

Grafische Objekte und deren Programmierung

Hochschule Darmstadt

Björn Frömmer

Prof. Dr. Benjamin Meyer

KAPITEL 9

Visualisierungstechniken

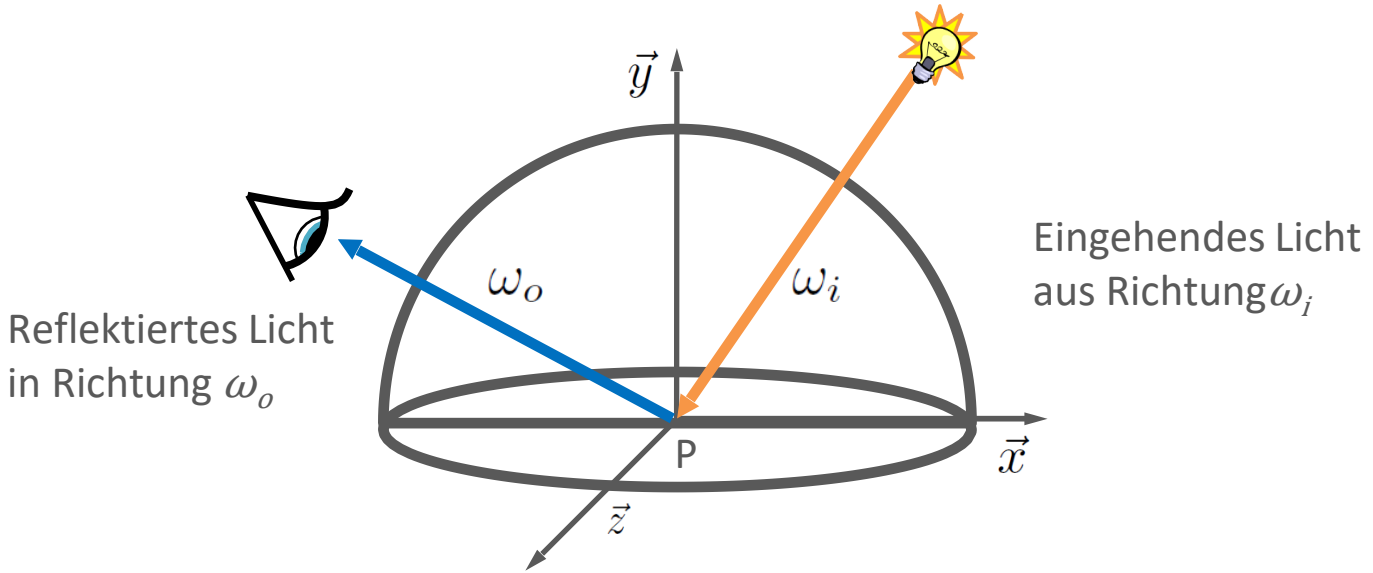
Faktoren, die das Aussehen eines Objektes bestimmen:

- Position und Orientierung des Betrachters relativ zum beleuchteten Objekt
- Position, Orientierung und Typ der Lichtquelle
- Oberflächenmaterial des Objektes

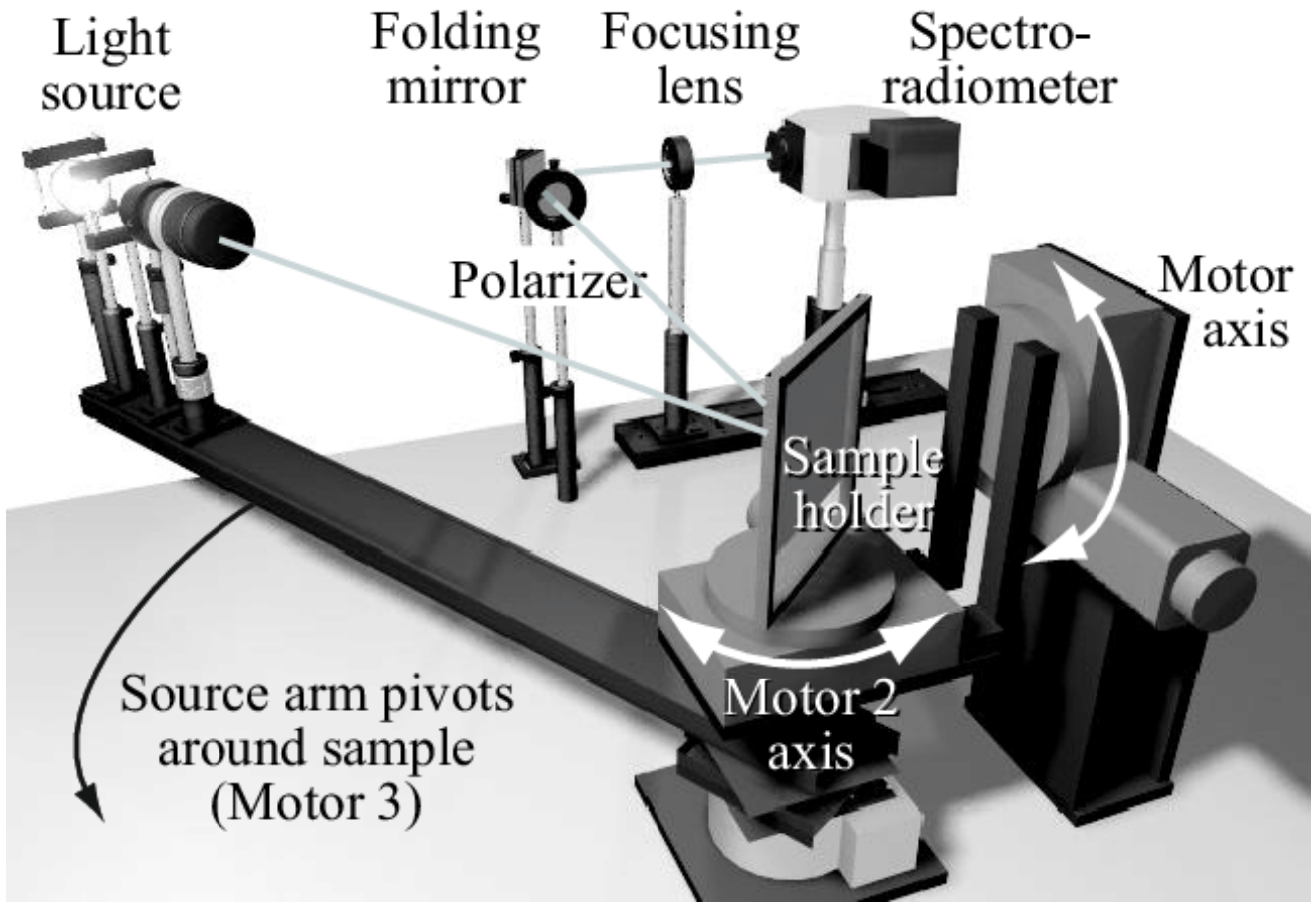
Quelle:
http://forums.cgsociety.org/showthread.php?t=233460
Zuletzt besucht: 2006-01-22.

BRDF – Bidirectional Reflectance Distribution Function

- Beschreibt, wie viel des einfallenden Lichts in eine bestimmte Richtung reflektiert wird
- 4D-Funktion
  - Vektor des einfallenden Lichts und Richtung des reflektierten Lichts (beide 2D in Kugelkoordinaten)
  - Enorme Datenmengen

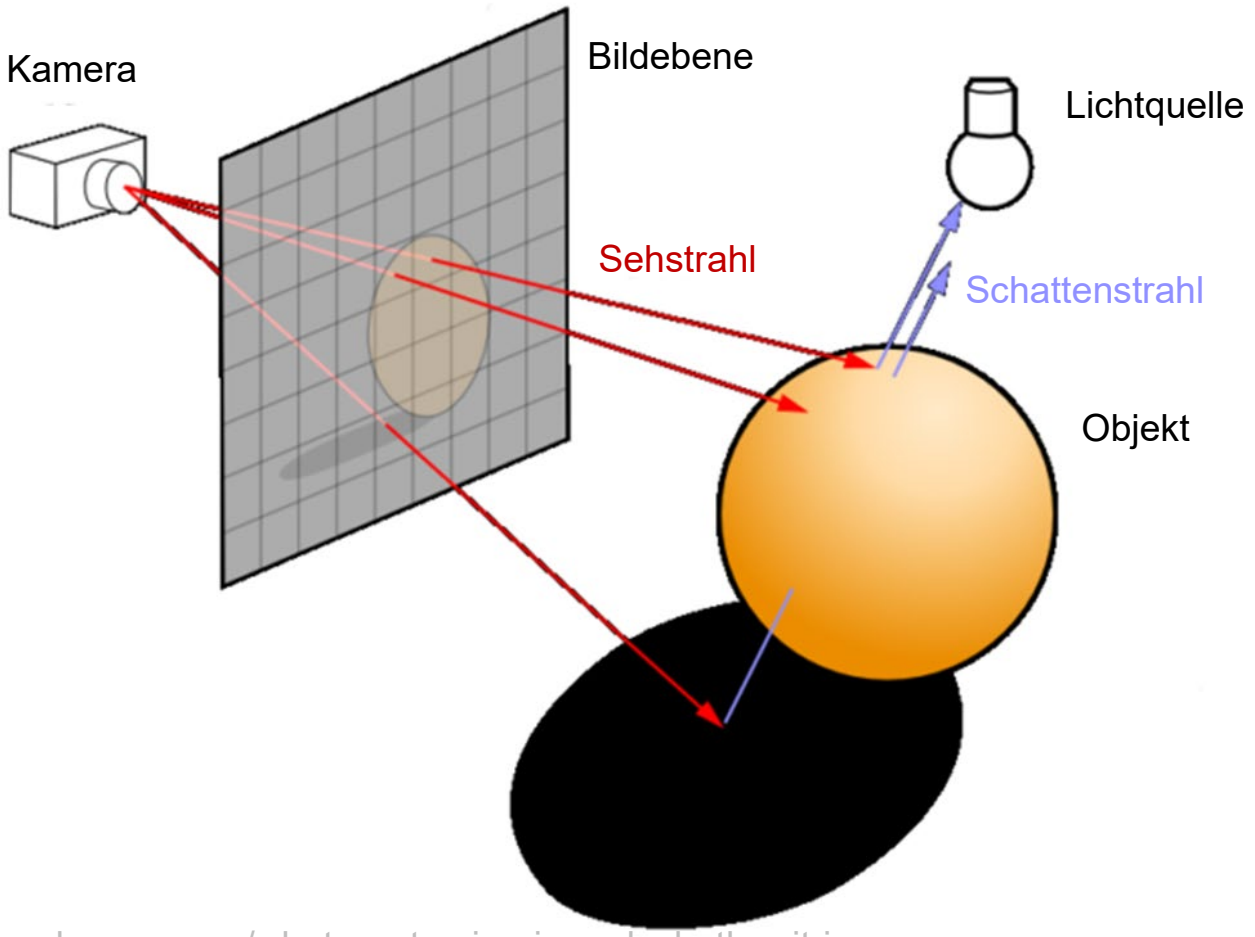


Gonio-Reflektometer



Quelle: Li, Hongsong & C. Foo, Sing & E. Torrance, Kenneth & Westin, Stephen. (2006). *Automated three-axis gonioreflectometer for computer graphics applications*. Optical Engineering. 45(4)

Raytracing



Quelle: https://tech-news.websawa.com/what-ray-tracing-is-and-whether-it-is-necessary-to-us-in-the-games/

Raytracing

Quelle: https://tech-news.websawa.com/what-ray-tracing-is-and-whether-it-is-necessary-to-us-in-the-games/

Modellierung

- Die **explizite Speicherung** der BRDF für jede ein- und ausgehende Lichtrichtung ist zu speicherintensiv
  - Beispiel: 1.2GB für 100x100 Richtungen für ein Material

- **Idee**: Eine einfache mathematische Funktion finden, die stattdessen ähnliche Ergebnisse liefert

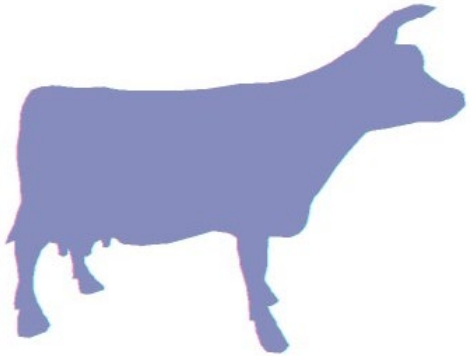
- Diese Funktionen werden **Reflexionsmodelle** (oder Beleuchtungsmodelle) genannt

Beleuchtungsmodelle / Reflexionsmodelle

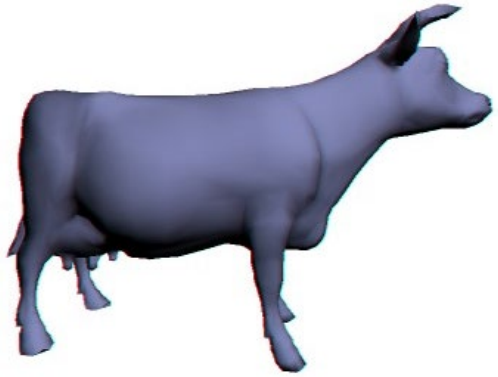
Phong‘sches Reflexionsmodell

- Nach: Bui-Tuong Phong, „Illumination for Computer Generated Pictures“, in Communications of the ACM, 1975
- Lichtquelle wird als Punktlichtquelle angenommen
- Die in Richtung des Betrachters abgegebene Lichtintensität ergibt sich aus der Summe der drei Komponenten:
  - Diffuse Reflexion
  - Ambiente Beleuchtung
  - Spiegelnde Reflexion

Phong‘sches Reflexionsmodell



Ambiente Reflexion



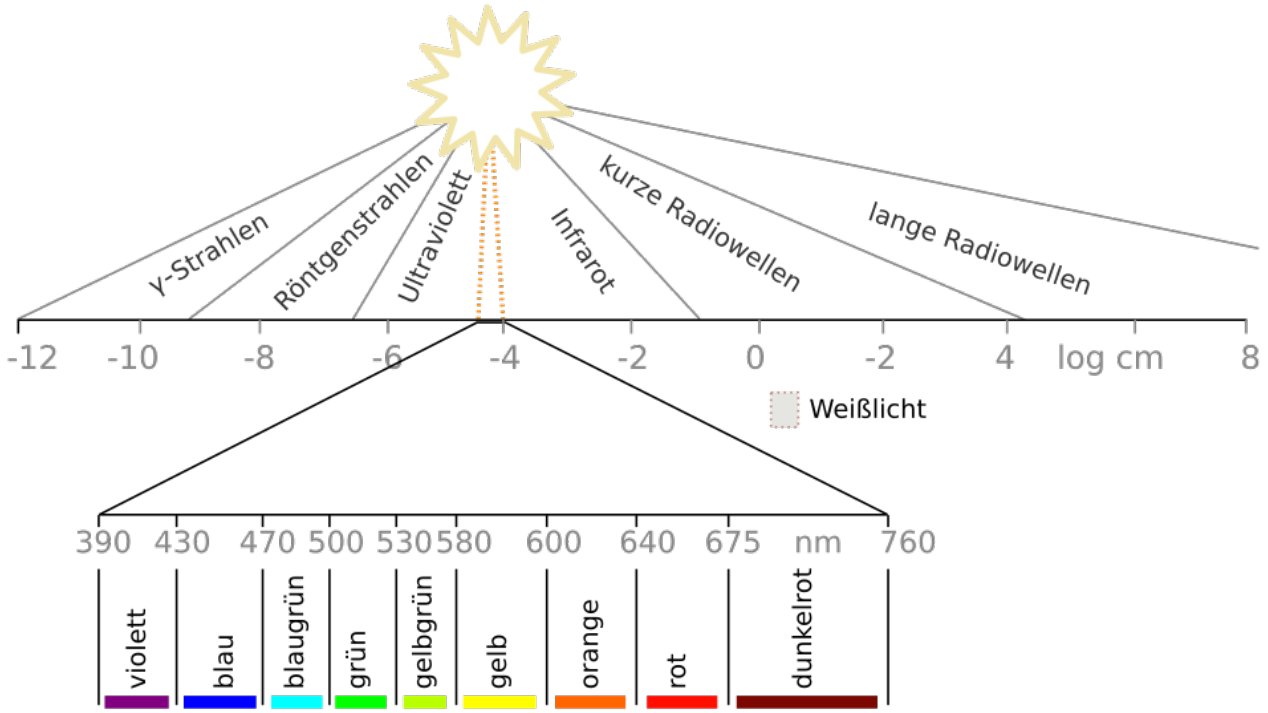
Ideal diffuse Reflexion

Ideal spiegelnde Reflexion

Ergebnis

Physikalische Herleitung

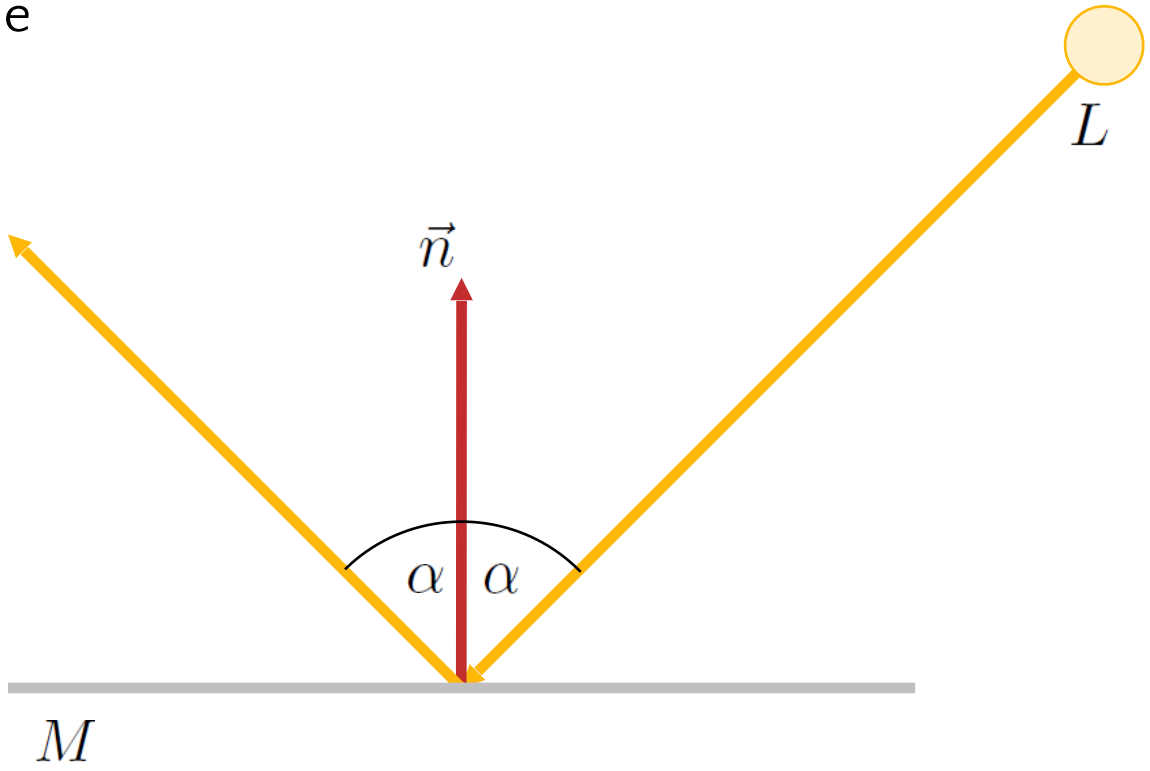
Licht = Elektromagnetische Wellen



Quelle: https://wisotop.de/lichtwellen-wellenlaenge-farbe.php

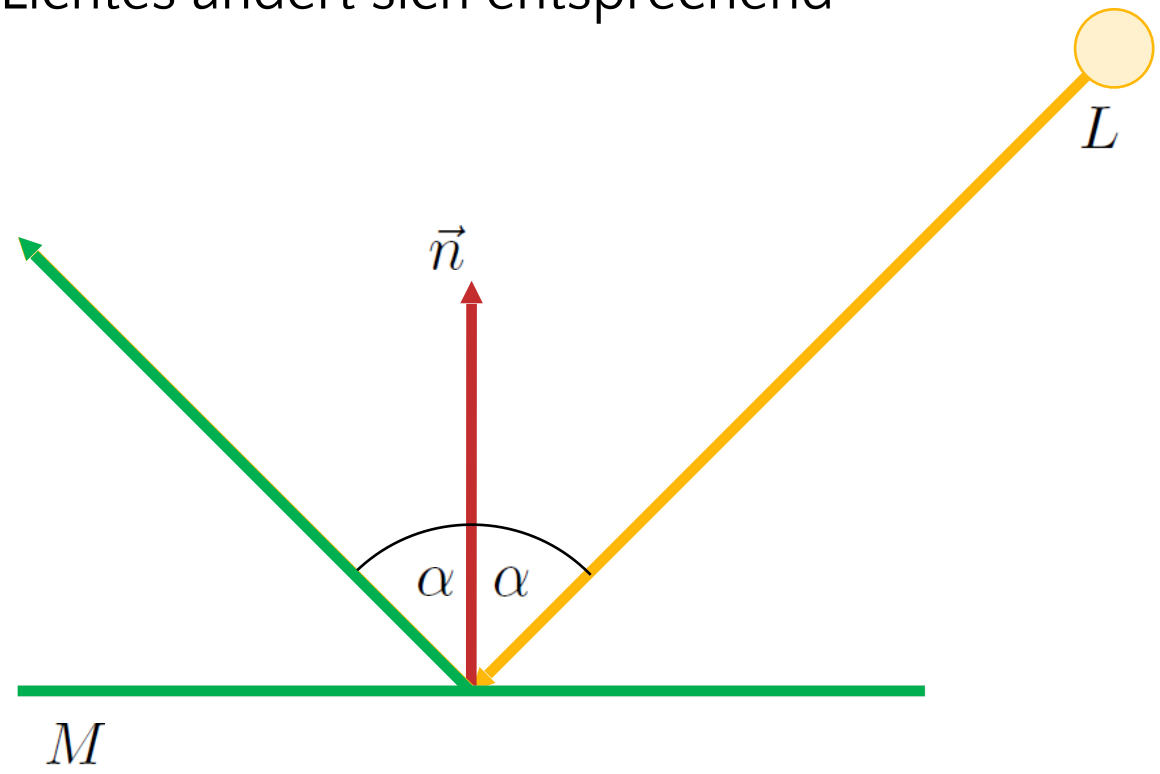
Reflexionsgesetz

- Annahme:
  - Punktlichtquelle
  - Reflektierende Oberfläche

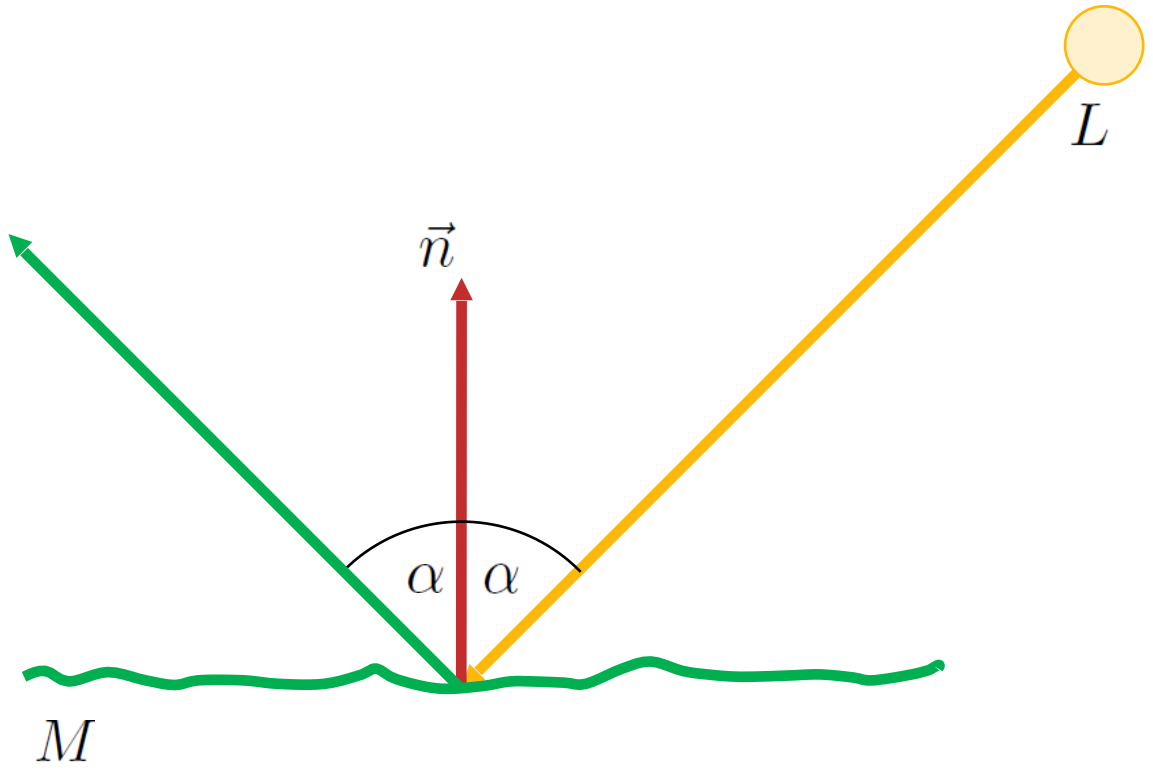


Absorption

- Es werden nicht alle Wellenlängen reflektiert
- Die Farbe des reflektierten Lichtes ändert sich entsprechend

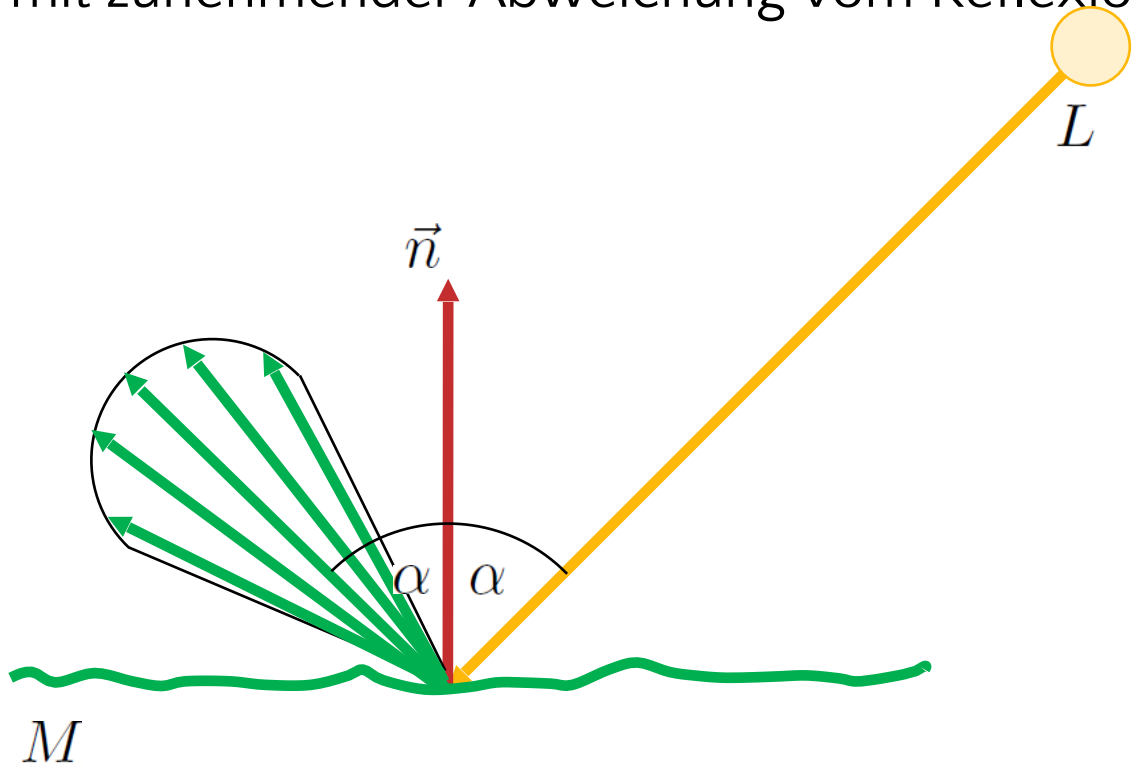


Rauheit – nicht spiegelnde Oberflächen



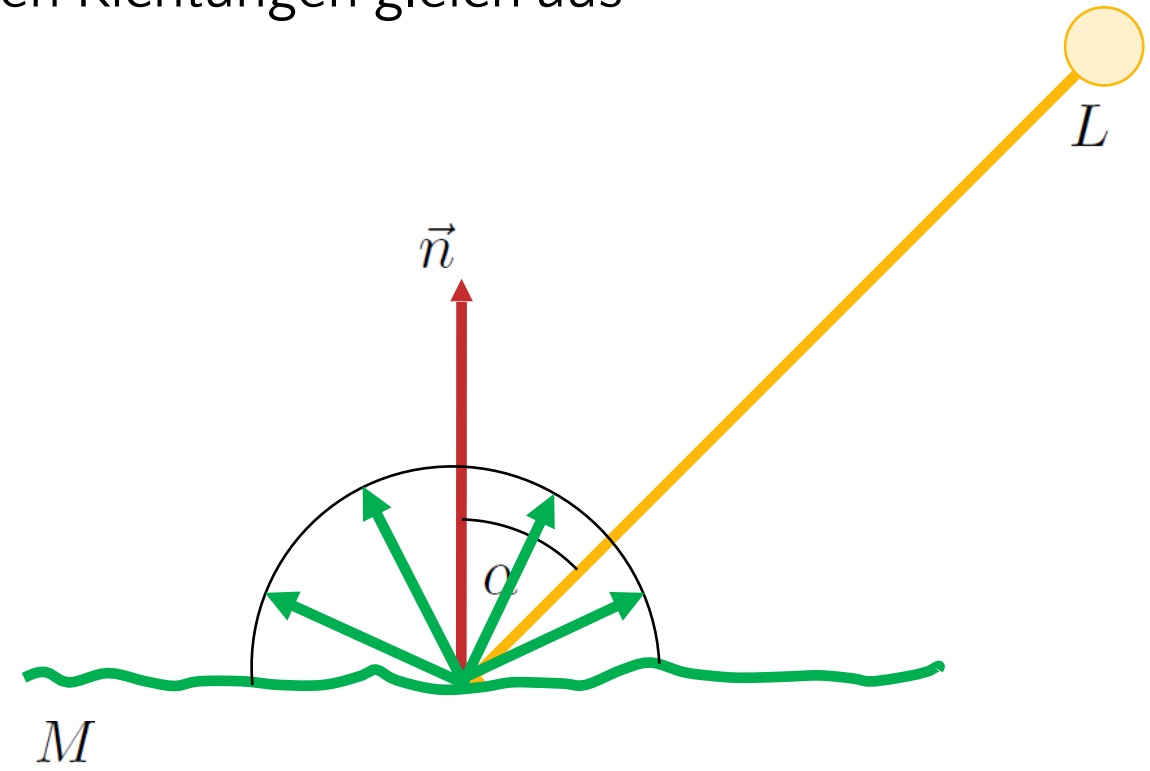
Rauheit – nicht spiegelnde Oberflächen

- Die Mikrogeometrie beeinflusst die Reflexionseigenschaften
- Das reflektierte Licht nimmt mit zunehmender Abweichung vom Reflexionswinkel α ab



Lambertsches Kosinusgesetz

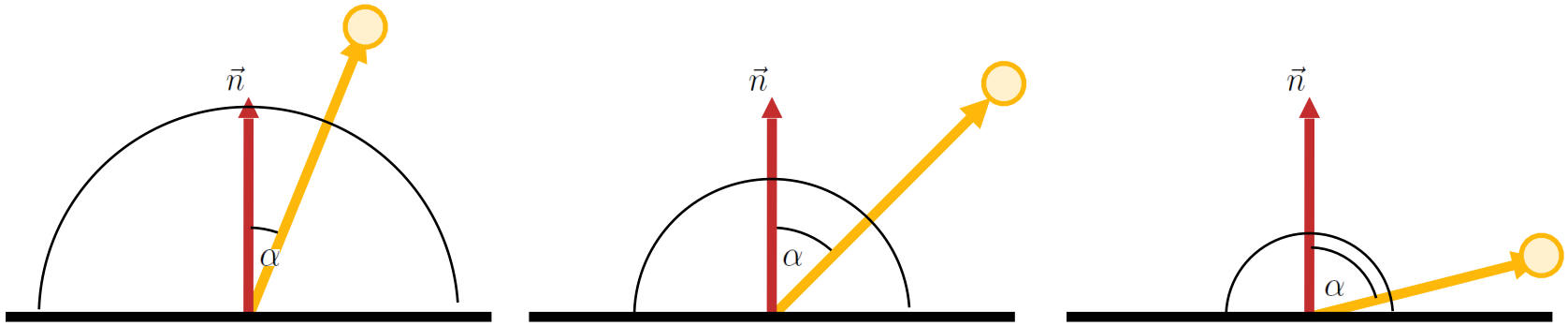
- Perfekt diffus reflektierendes Material
- Die Oberfläche sieht aus allen Richtungen gleich aus



Lambertsches Kosinusgesetz

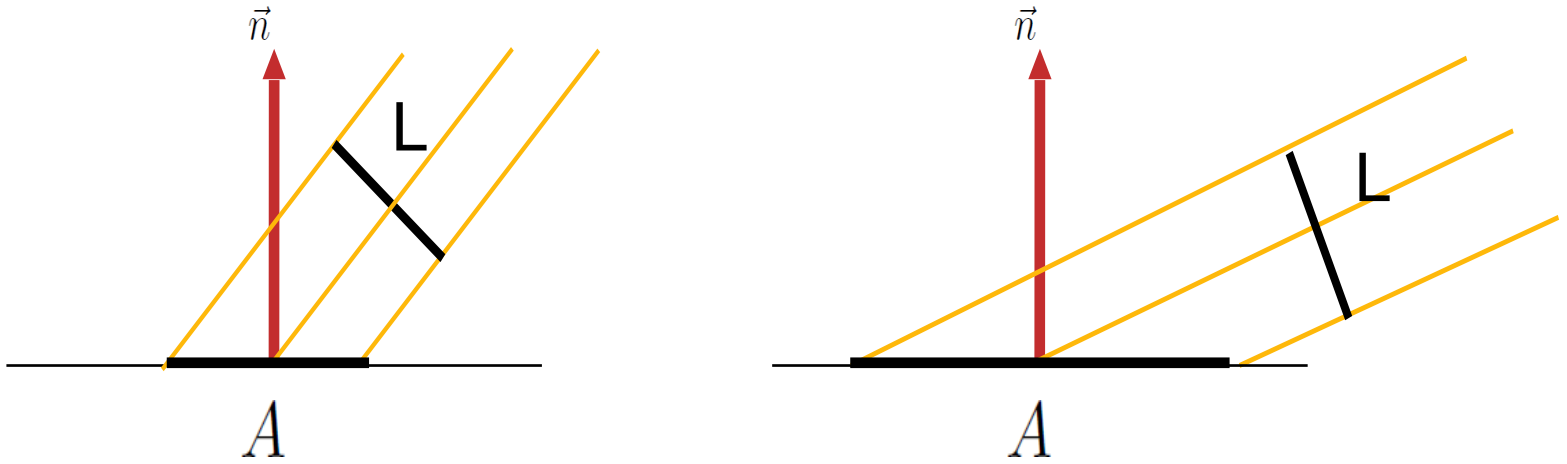
- Intensität **direkt proportional zum Kosinus des Winkels** zwischen Normalen und Lichtvektor für **diffuse Materialien**

- Begründung: Die beleuchtete Fläche nimmt mit dem Neigungswinkel einer Oberfläche zu



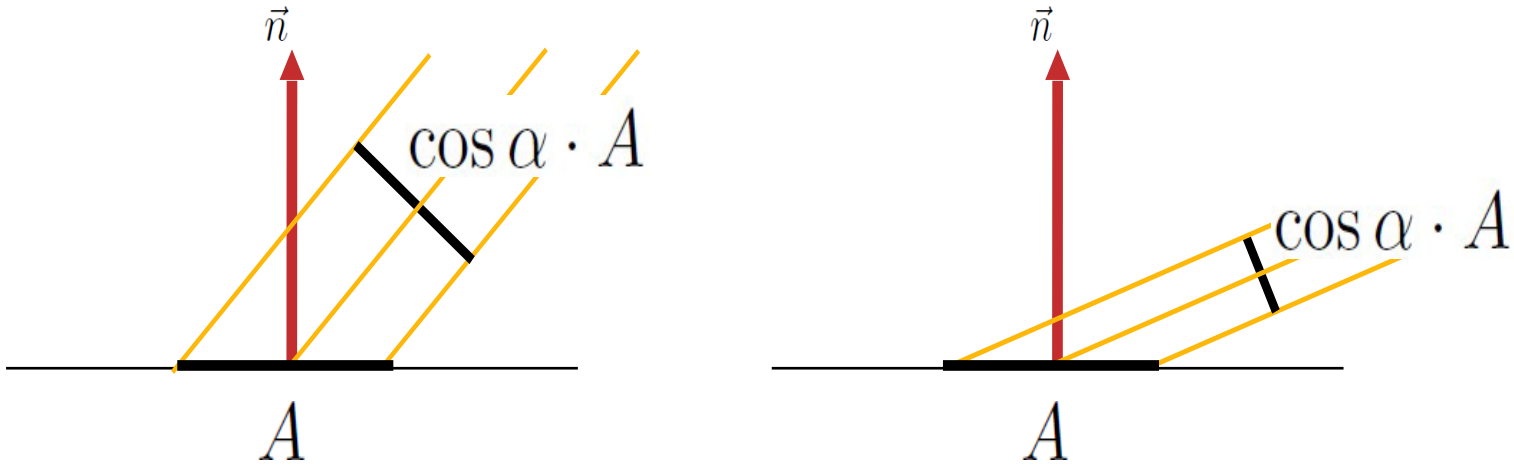
Lambertsches Kosinusgesetz

- Gleiche Anzahl von Photonen verteilt auf eine größere Fläche → Fläche wird dunkler



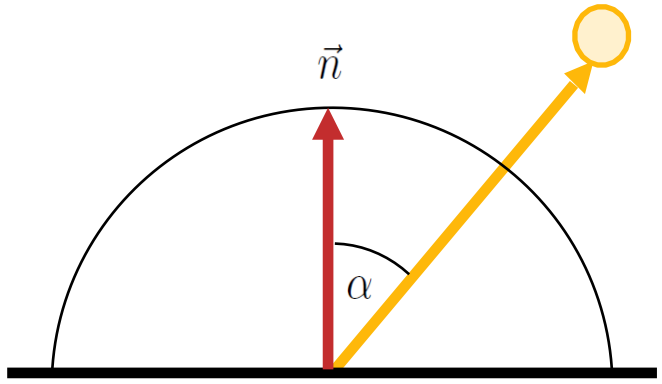
Lambertsches Kosinusgesetz

- Gleiche Anzahl von Photonen verteilt auf eine größere Fläche → Fläche wird dunkler
- Alternative Interpretation: die projizierte Fläche nimmt mit dem Neigungswinkel ab



Diffuse Reflexion

- Abhängigkeiten:
  - Lichtfarbe
  - Materialfarbe
  - Winkel α zwischen Oberflächennormale und Lichtaustrittsvektor



$$\begin{pmatrix} M_r \\ M_g \\ M_b \end{pmatrix} \cdot \begin{pmatrix} L_r \\ L_g \\ L_b \end{pmatrix} \cdot \cos\alpha$$

Multiplikation komponentenweise!

Diffuses Reflexionsmodell

$$L_o = \sum_{i=1}^{n} M_d \cdot L_i \cdot \cos\alpha$$

Für jede Lichtquelle!

- L_o: resultierende Farbe
- Material
  - **M_d: Diffuses Material**
  - **α: Winkel zwischen Oberflächennormale und Lichtaustrittsvektor**

Lichtquellen

L_i: Lichtfarbe der i-ten Lichtquelle

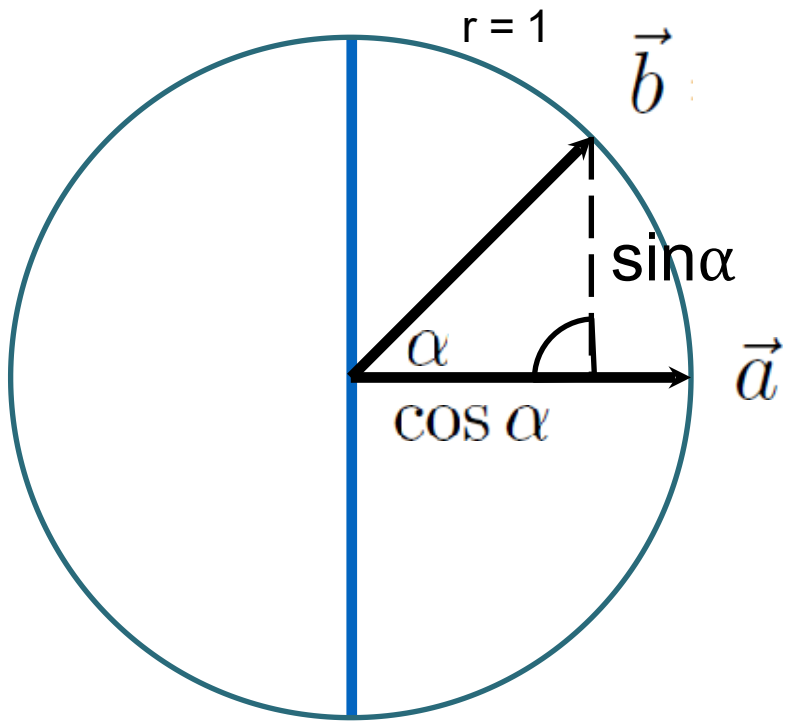
Skalarprodukt (Punktprodukt)

$$\vec{a} \circ \vec{b} = |\vec{a}| \cdot |\vec{b}| \cdot \cos\alpha = a_x b_x + a_y b_y + a_z b_z$$

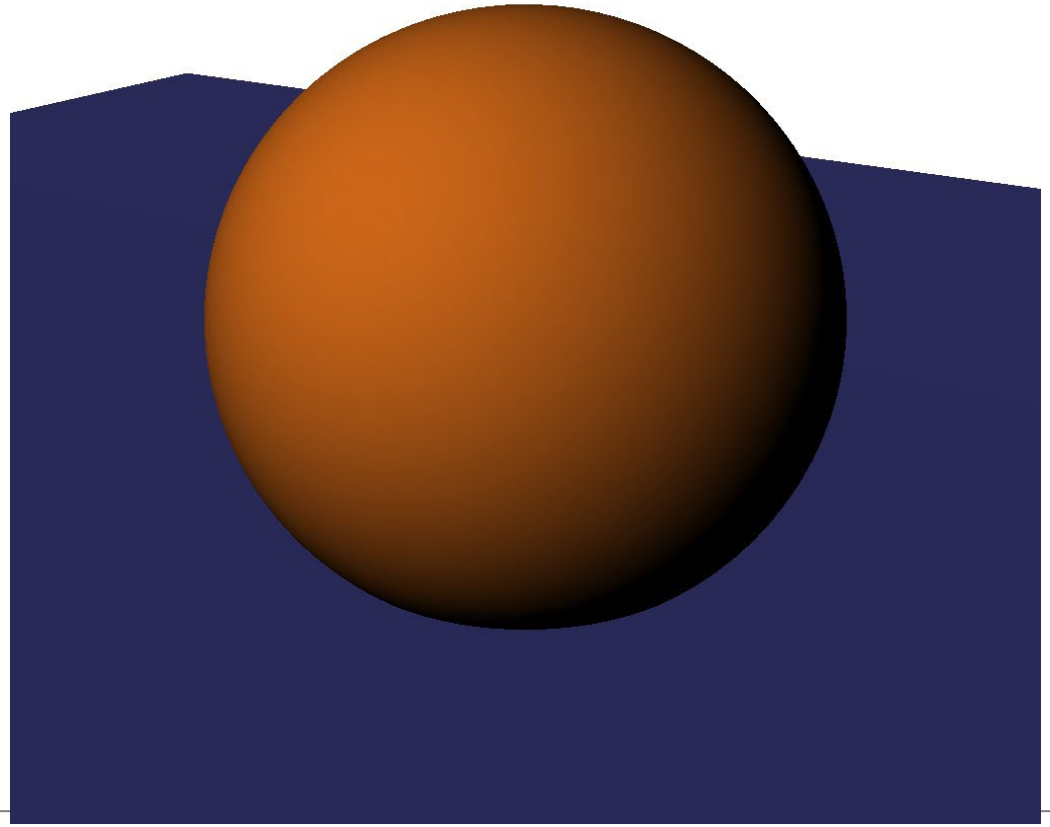
wenn $|\vec{a}| = 1$

und $|\vec{b}| = 1$ dann

$$\vec{a} \circ \vec{b} = \cos\alpha$$

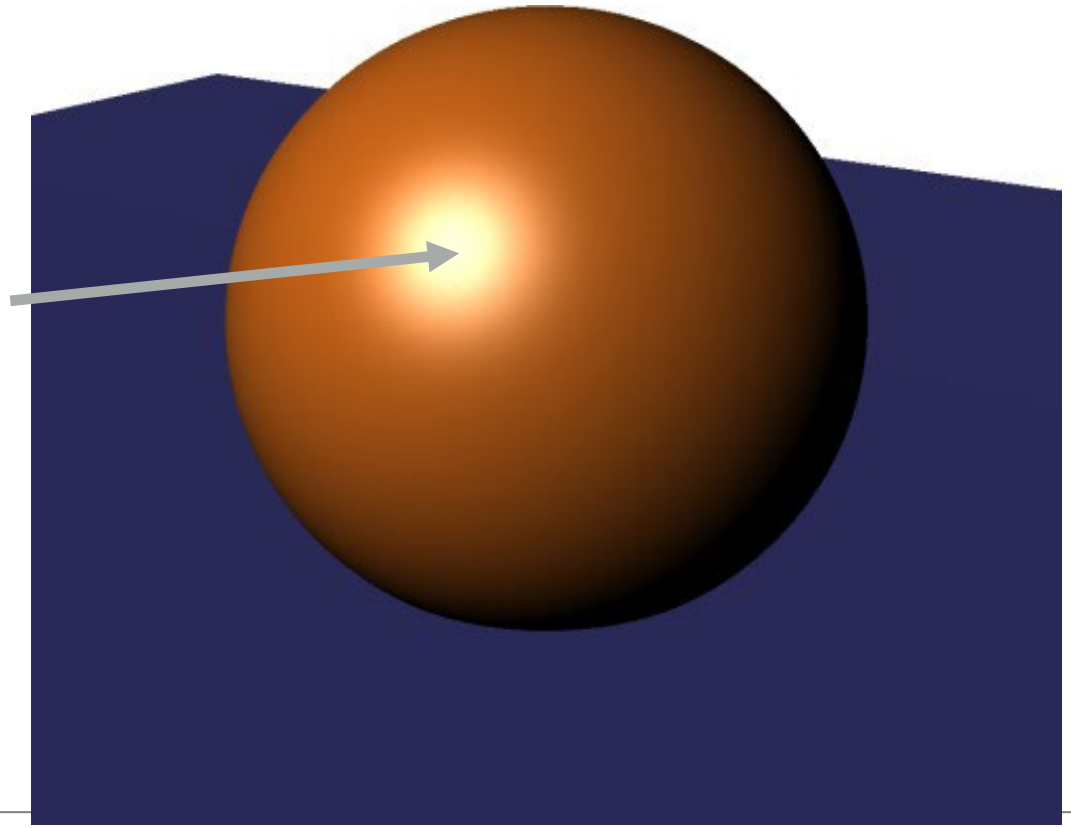


Beispiel



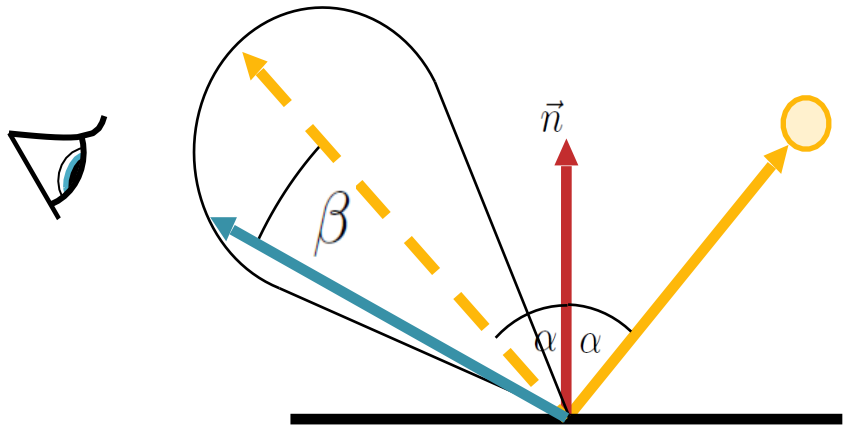
Spekulare Reflexion

Spekulares Glanzlicht



Spekulare Reflexion

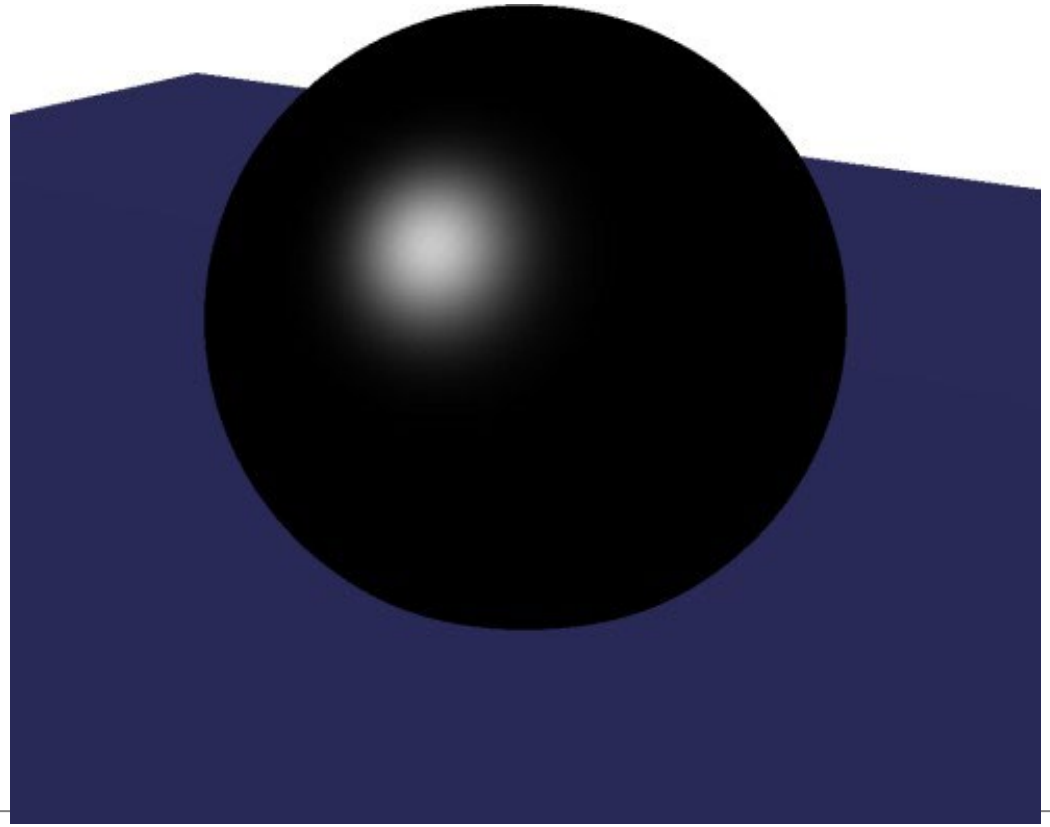
- Abhängigkeiten
  - Lichtfarbe
  - Materialfarbe (kann von der Farbe des diffusen Materials abweichen, oft weiß)
  - Winkel β zwischen Sichtvektor und reflektiertem Lichtvektor
  - Glanzfaktor (engl. shininess) k



$$\begin{pmatrix} M_r \\ M_g \\ M_b \end{pmatrix} \cdot \begin{pmatrix} L_r \\ L_g \\ L_b \end{pmatrix} \cdot \cos^k \beta$$

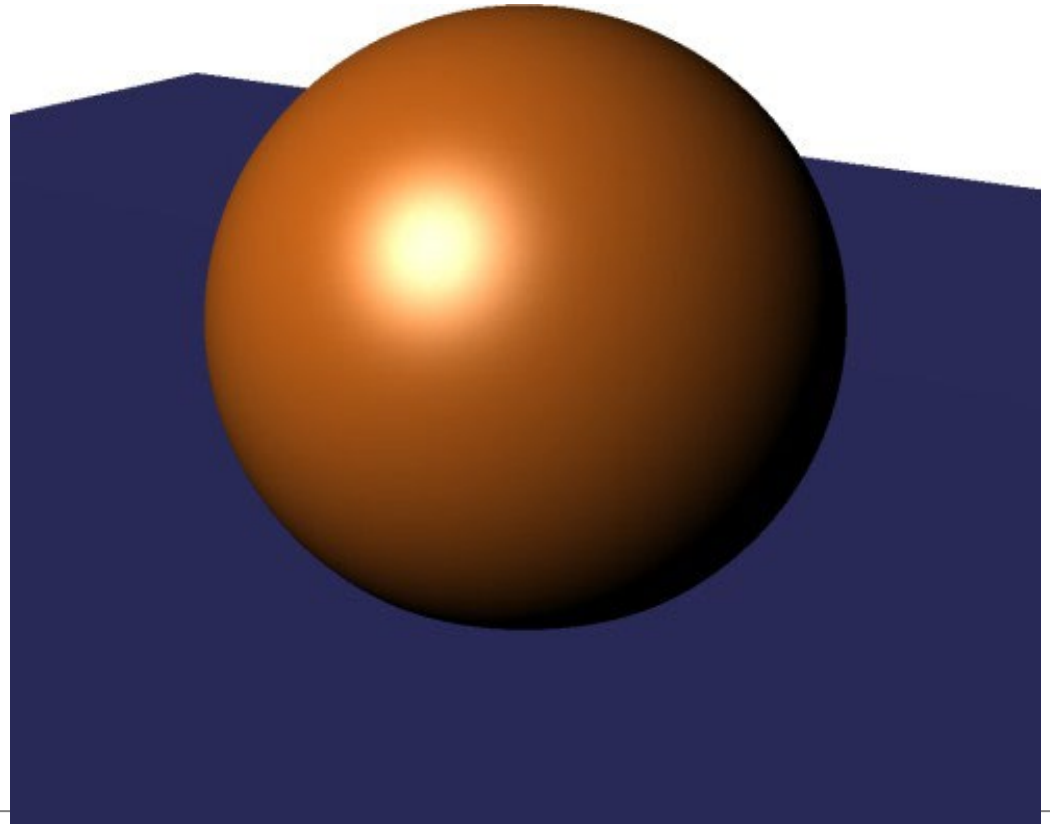
Spekulare Reflexion

Spekularer Anteil



Spekulare Reflexion

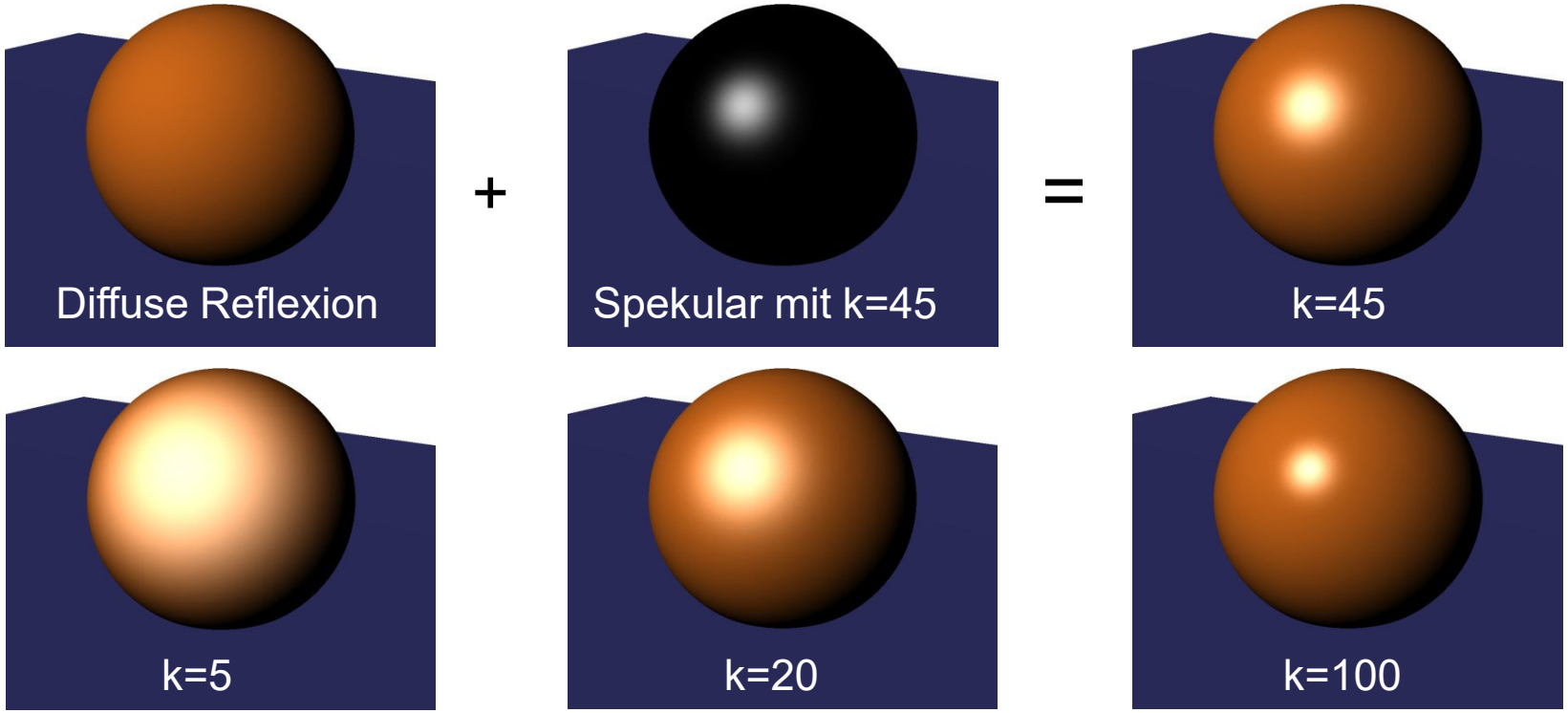
Diffuser + Spekularer Anteil



Spiegelnde Reflexion – Shininess

Die spiegelnde Reflexion kann im Phong-Modell parametrisiert werden durch die **Shininess** k.

Dabei sorgt eine hohe Shininess für kleinere Highlights und den optischen Eindruck eines "harten" Materials



Diffuses + spekulares Modell

$$L_o = \sum_{i=1}^{n} (M_d \cdot \cos\alpha + M_s \cdot \cos^k \beta) \cdot L_i$$

- L_o: resultierende Farbe
- Material
  - **M_d: Diffuses Material**
  - **α: Winkel zwischen Oberflächennormale und Lichtaustrittsvektor**
  - **M_s: Spekulares Material**
  - **k: Shininess**
  - **β: Winkel zwischen Sichtvektor und reflektiertem Lichtvektor**

Lichtquellen:

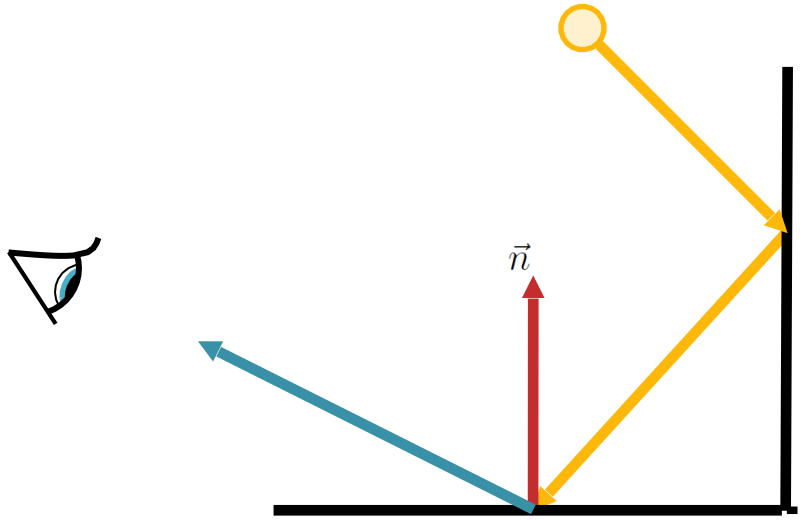
L_i: Lichtfarbe der i-ten Lichtquelle

Ambiente Beleuchtung

Quelle: https://forum.graphisoft.de/viewtopic.php?f=15&t=23856

Ambiente Beleuchtung

- Imitiert grob die indirekte Beleuchtung
  - d. h. Licht, das von anderen Oberflächen reflektiert wird



$$\begin{pmatrix} M_r \\ M_g \\ M_b \end{pmatrix} \cdot \begin{pmatrix} L_r \\ L_g \\ L_b \end{pmatrix}$$

Phong’sches Reflexionsmodell

$$L_o = M_d \cdot L_a + \sum_{i=1}^{n} (M_d \cdot \cos\alpha + M_s \cdot \cos^k \beta) \cdot L_i$$

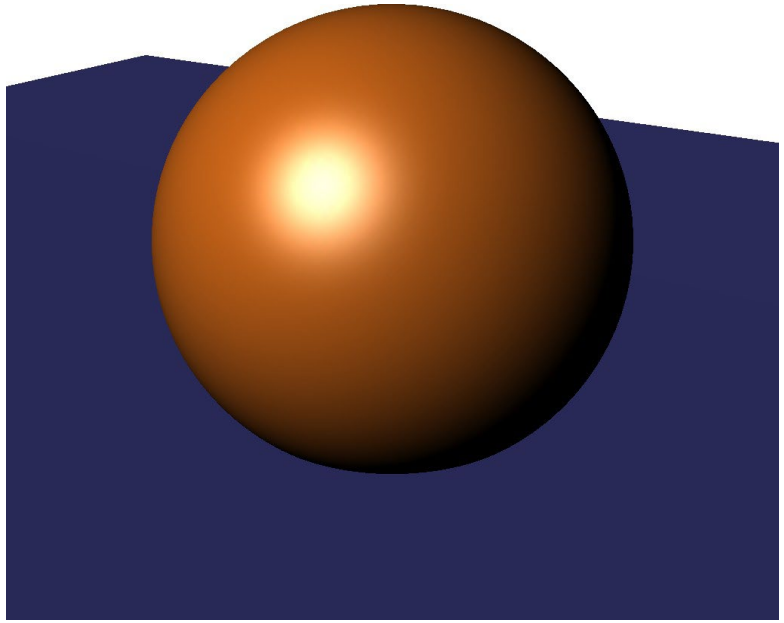
- L_o: resultierende Farbe
- Material
  - **M_d: Diffuses Material**
  - **α: Winkel zwischen Oberflächennormale und Lichtaustrittsvektor**
  - **M_s: Spekulares Material**
  - **k: Shininess**
  - **β: Winkel zwischen Sichtvektor und reflektiertem Lichtvektor**

Lichtquellen:

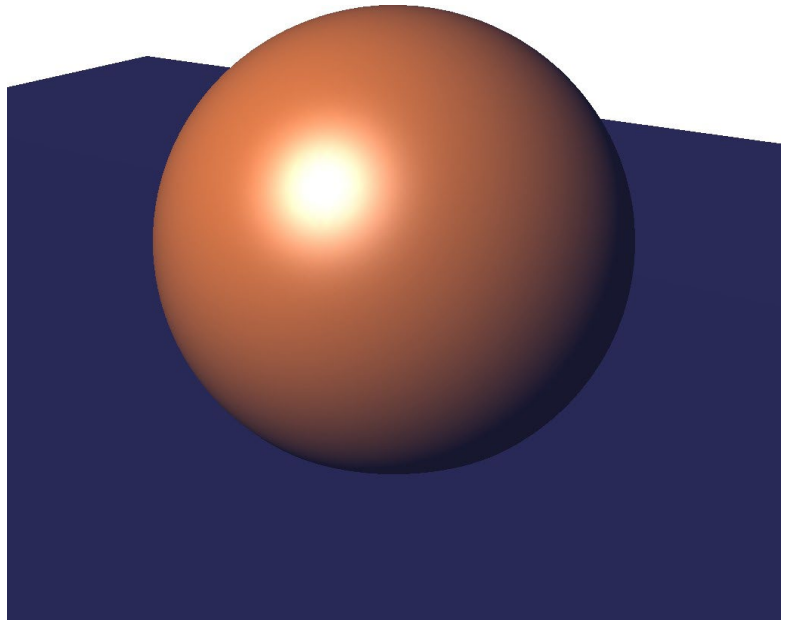
L_i: Lichtfarbe der i-ten Lichtquelle

L_a: Globale ambiente Lichtfarbe

Beispiel



Diffus + Spekular



Ambient + Diffus + Spekular

Emissive / leuchtende Oberflächen

Phong’sches Reflexionsmodell + Emission

$$L_o = M_e + M_d \cdot L_a + \sum_{i=1}^{n} (M_d \cdot \cos\alpha + M_s \cdot \cos^k \beta) \cdot L_i$$

- L_o: resultierende Farbe
- Material
  - M_d: Diffuses Material
  - α: Winkel zwischen Oberflächennormale und Lichtaustrittsvektor
  - M_s: Spekulares Material
  - k: Shininess
  - β: Winkel zwischen Sichtvektor und reflektiertem Lichtvektor
  - M_e: Emissiver Term

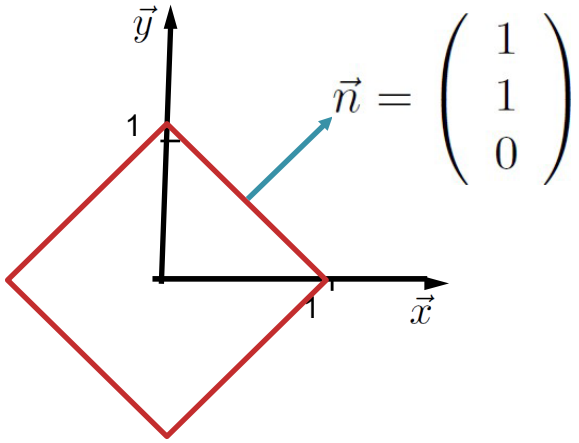
Lichtquellen:

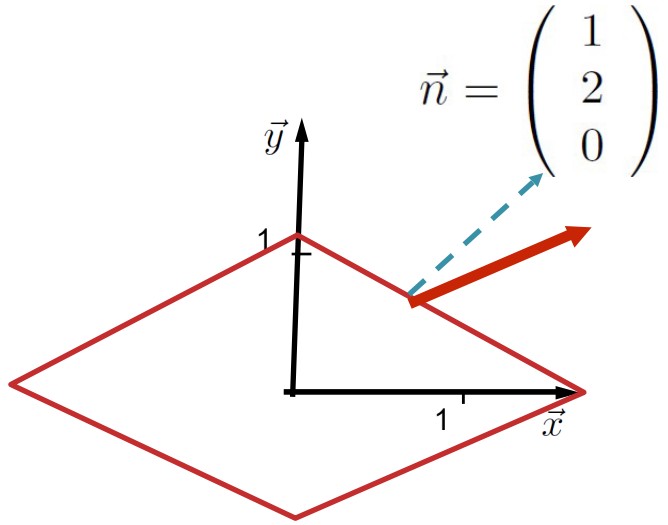
L_i: Lichtfarbe der i-ten Lichtquelle

L_a: Globale ambiente Lichtfarbe

Transformation der Normalen

- Ist die Normale nach der Modelltransformation noch korrekt?
- Beispiel



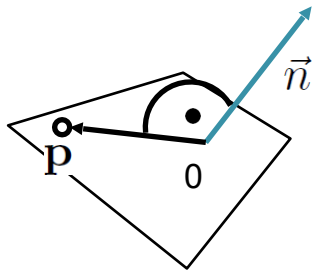


$$\vec{n}' = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix}$$

Falsche Normale!

Transformation der Normalen

Nicht klausurrelevant!



- Richtig wäre:

$$(M' \cdot \vec{n}) \circ (M \cdot \mathbf{p}) = 0$$

- Skalarprodukt:

$$\vec{n} \circ \mathbf{p} = (M' \cdot \vec{n}) \circ (M \cdot \mathbf{p})$$

- Lösung:

$$\vec{n} \circ \mathbf{p} = \vec{n}^\top \cdot \mathbf{p}$$

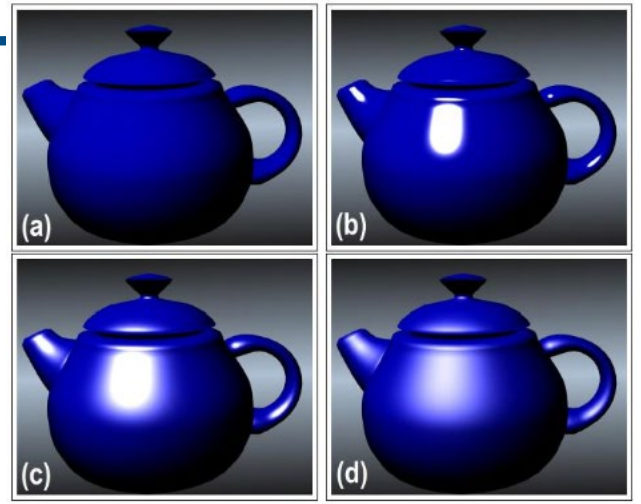
$$\begin{aligned}
\vec{n} \circ \mathbf{p} &= \vec{n}^\top \cdot \mathbf{p} \\
&= \vec{n}^\top \cdot \mathbf{I} \cdot \mathbf{p} \\
&= \vec{n}^\top \cdot (M^{-1} \cdot M) \cdot \mathbf{p} \\
&= (\vec{n}^\top \cdot M^{-1}) \cdot (M \cdot \mathbf{p}) \\
&= (\vec{n}^\top \cdot (M^{-1})^\top \circ (M \cdot \mathbf{p}) \\
&= ((M^{-1})^\top \cdot \vec{n}) \circ (M \cdot \mathbf{p})
\end{aligned}$$

$$I = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

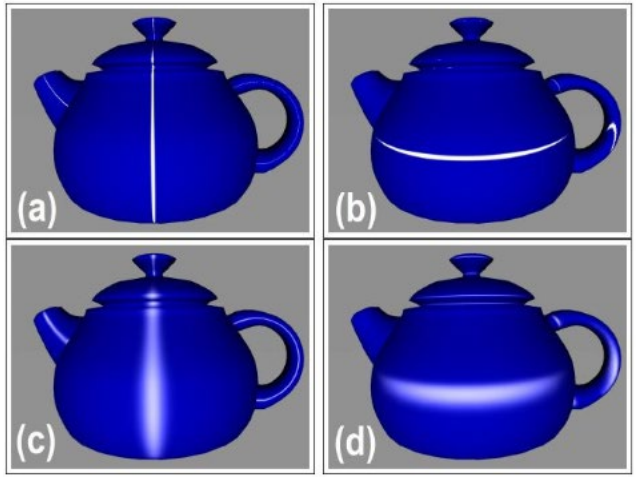
$$(A \cdot B)^\top = B^\top \cdot A^\top$$

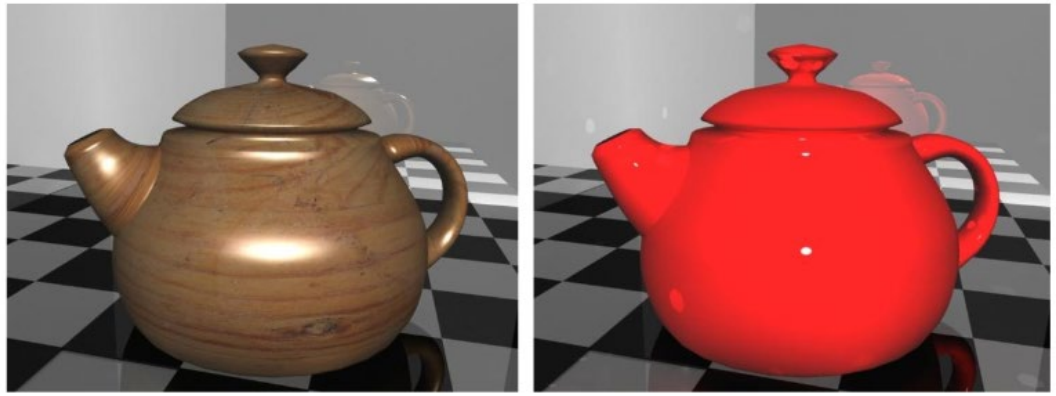
Weitere Reflexionsmodelle

- Ward (1992)
  - Isotrope und anisotrope BRDF
  - Verschiedene Materialien:
    z. B. Metall, poliertes Holz



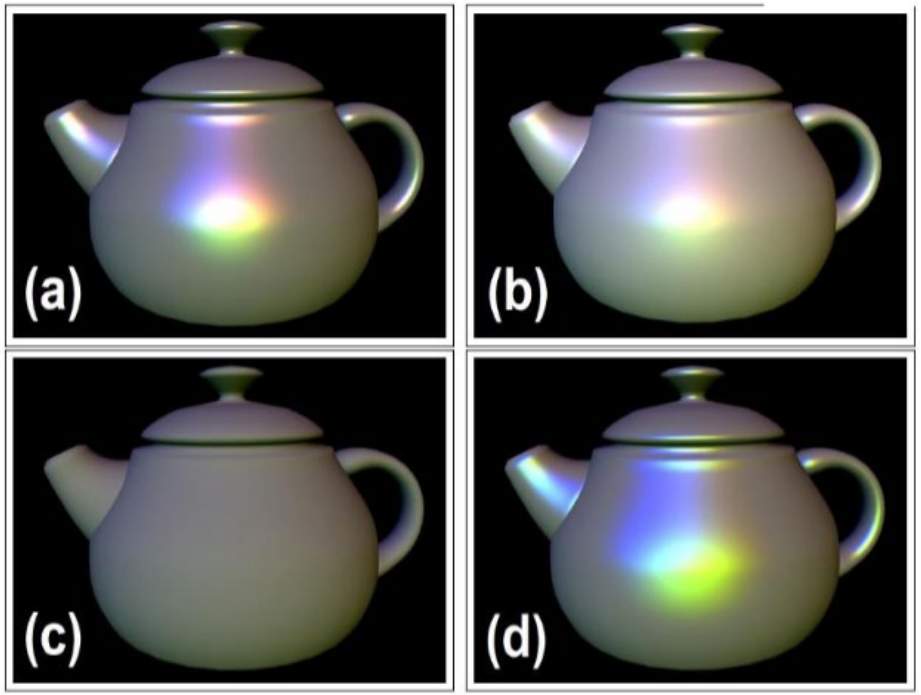
Isotrope Lichtreflexion Ward





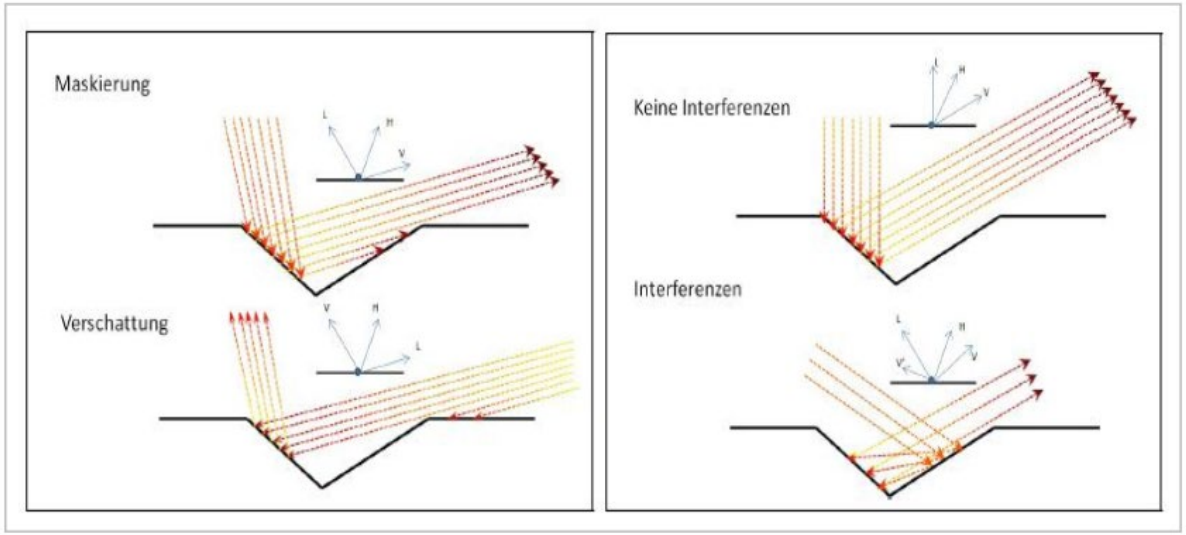
Weitere Reflexionsmodelle

- Ward (1992)
- LaFortune (1993)
  - Komplexe Reflexionseigenschaften
  - Retro-Reflexion



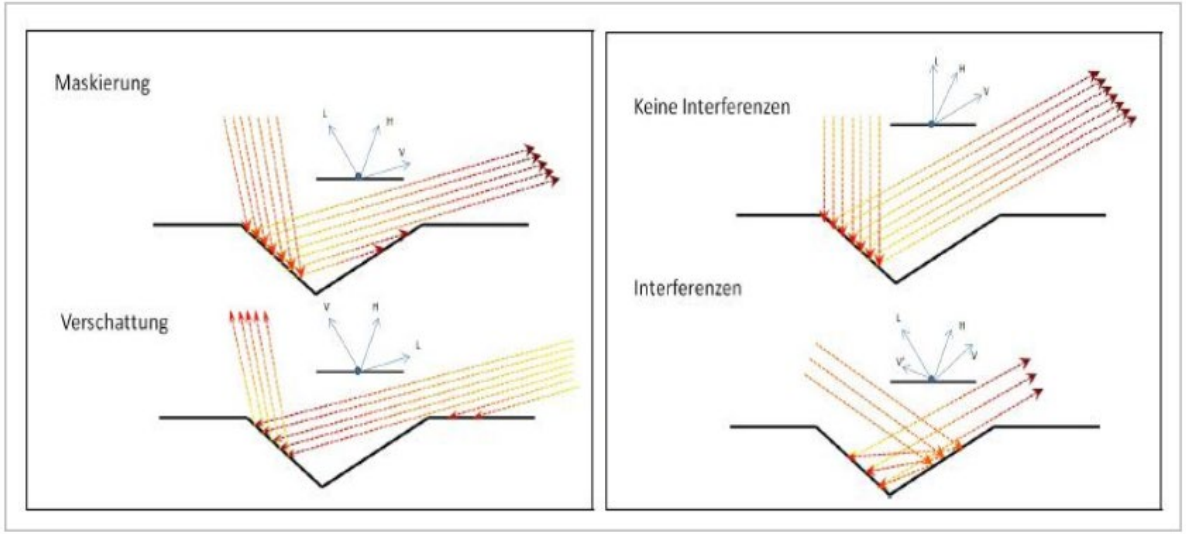
Weitere Reflexionsmodelle

- Ward (1992)
- LaFortune (1993)
- Cook-Torrance (1982)
  - physikalisch basierte Mikrofacetten (Fresnel)
  - Abschattungsfaktor



Weitere Reflexionsmodelle

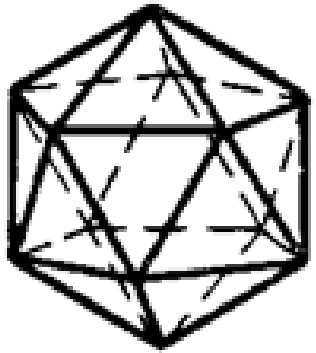
- Ward (1992)
- LaFortune (1993)
- Cook-Torrance (1982)
- Und viele mehr….



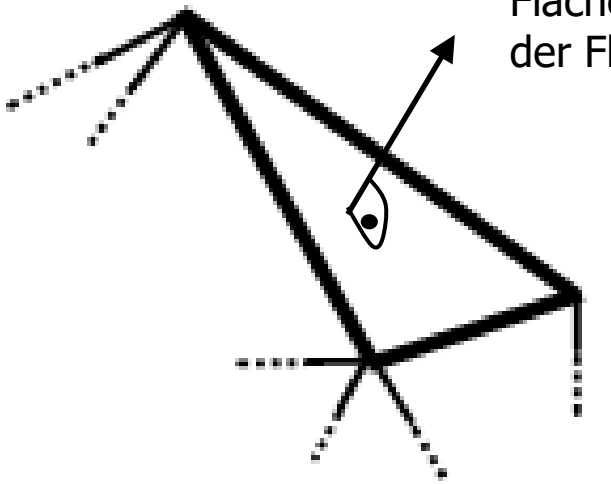
Shading

Ohne Normalen keine Beleuchtungsberechnung!

1. Variante: **Flächennormalen**



Ikosaeder



Flächennormale steht senkrecht auf der Fläche.

Auf welcher Stelle der Fläche „sitzt" diese Flächennormale im Dreieck?

Flat Shading

- **Verfahren:**
  - Für jede Fläche gibt es eine Normale.
  - Dadurch erhält jede Fläche (Facette) eine einheitliche Beleuchtung*).
  - D.h. jede Fläche besitzt nur eine Farbe.

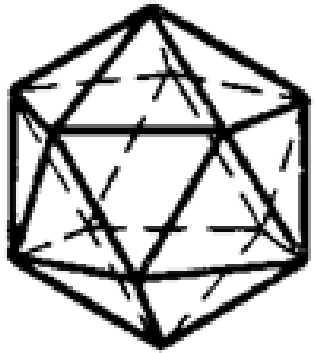
- **Vorteil:**
  - sehr einfache und sehr schnelle Beleuchtungsberechnung!

- **Nachteil:**
  - Da viele Objektoberflächen gekrümmt sind, ist die Qualität der Ergebnisse oftmals schlecht.

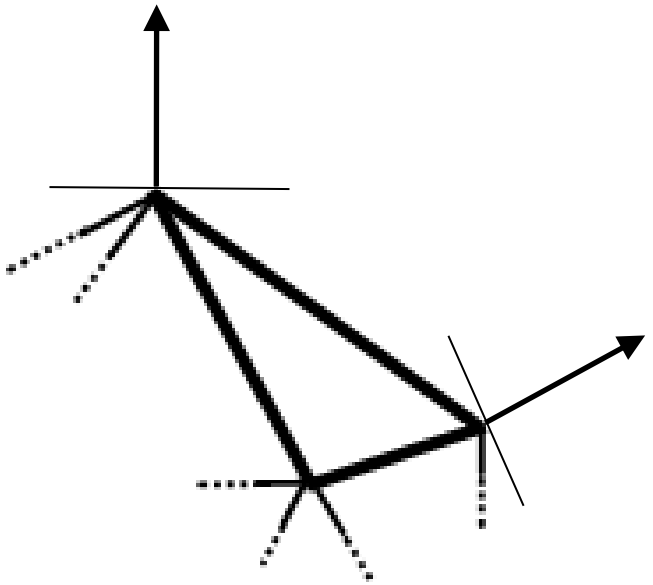
*) Beleuchtung wird meist mit dem Phong‘schen Reflexionsmodell berechnet.

Verbesserung der Beleuchtungssimulation beginnt bei den Normalen!

- 2. Variante: **Eckennormalen**



Ikosaeder

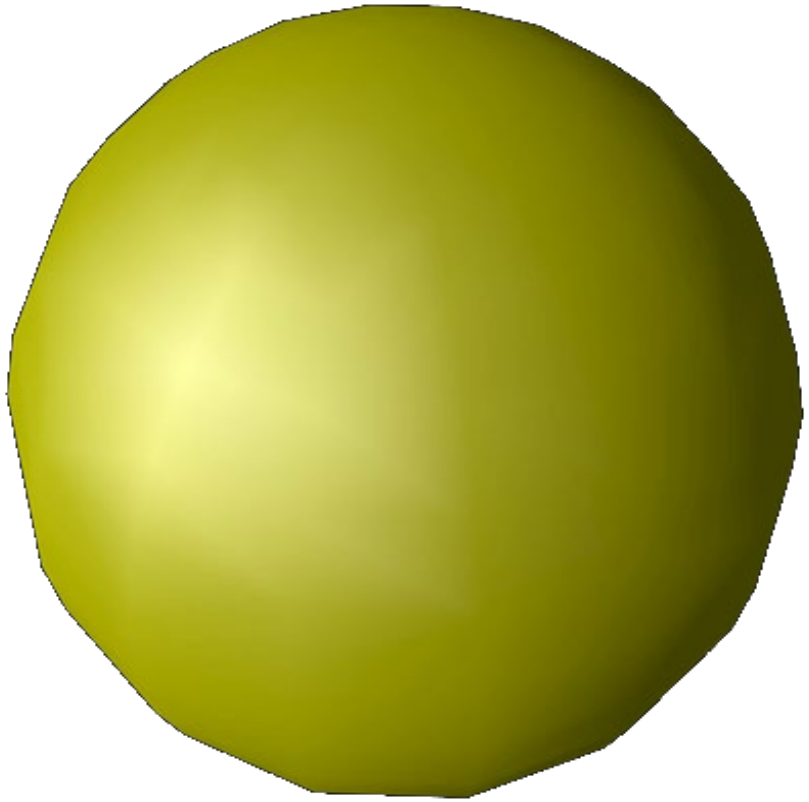


Eckennormalen stehen senkrecht auf der Tangentialebene …

… und **NICHT** wie die Flächennormale senkrecht auf der Fläche!

Eckennormalen berechnet man durch die Interpolation der Flächennormalen der benachbarten Flächen.

Gouraud Shading

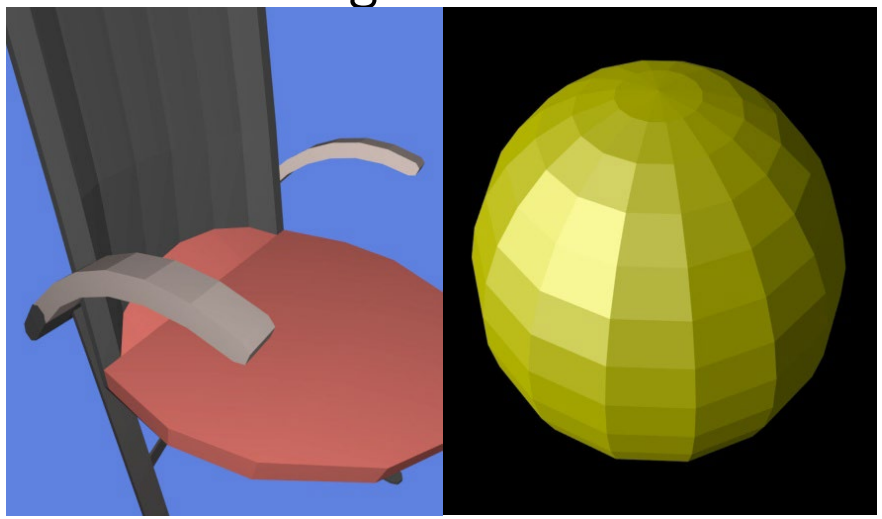


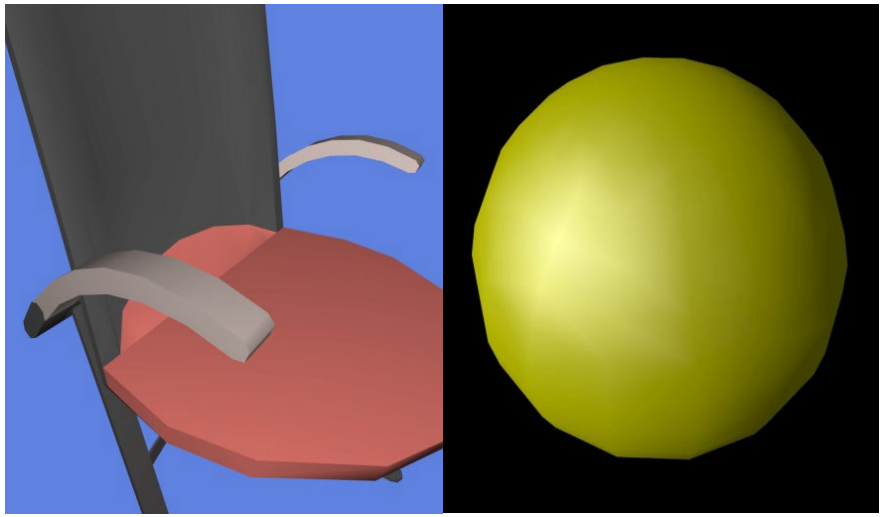
- **Verfahren:**
  - Beleuchtungsberechnung *) wird für jeden Eckpunkt durchgeführt.
  - D.h. pro Dreieck werden drei unterschiedliche Farben (Helligkeiten) berechnet.
  - Die Farben der Pixel innerhalb des Dreiecks werden durch bilineare Interpolation berechnet.
- **Vorteil:**
  - Dadurch entsteht eine optische Glättung – ohne Änderungen an der Geometrie vornehmen zu müssen.
- **Nachteil:**
  - Optische Glättung bewirkt ein „kaschieren“ der Kanten, die nicht zur Kontur gehören!

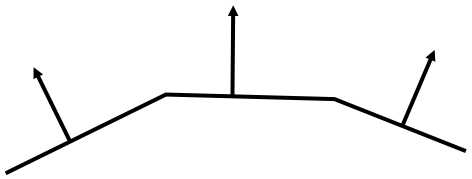
*) Beleuchtung wird meist mit dem Phong‘schen Reflexionsmodell berechnet.

Flat Shading versus Gouraud Shading

Visualisierungstechniken



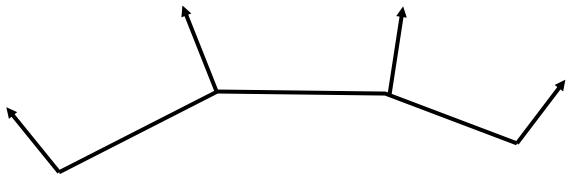




Ergebnis ist abhängig von
den berechneten Normalen!

Links: Flächennormalen

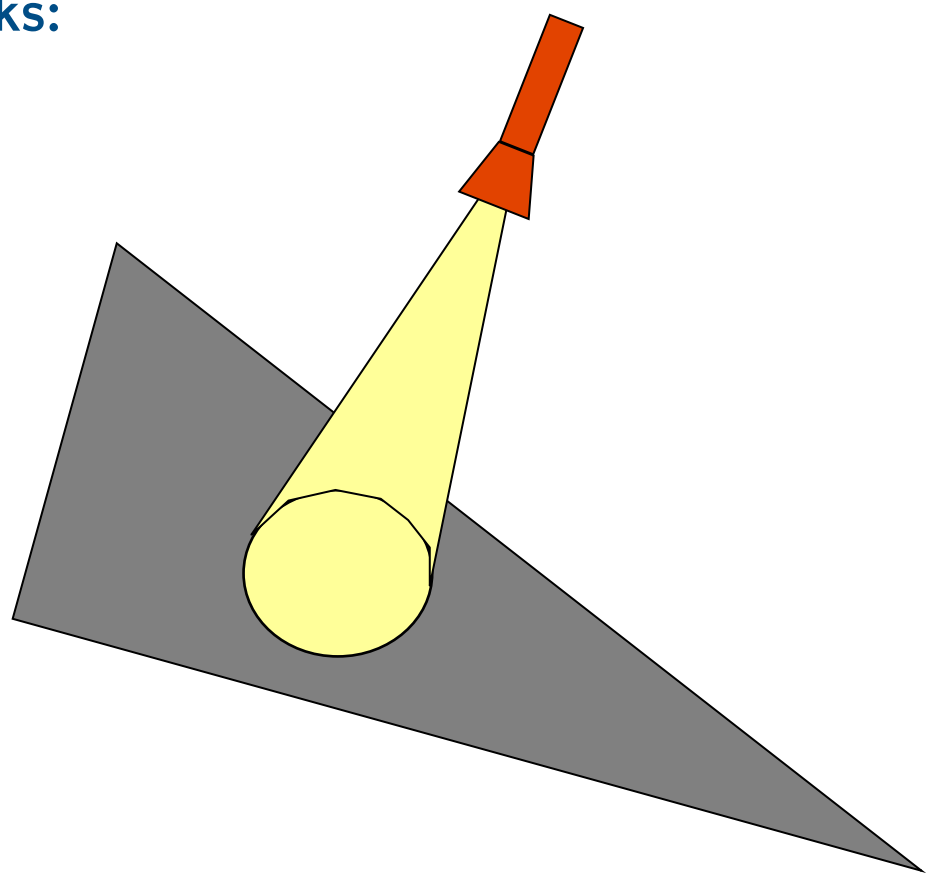
Rechts: Eckpunktnormalen



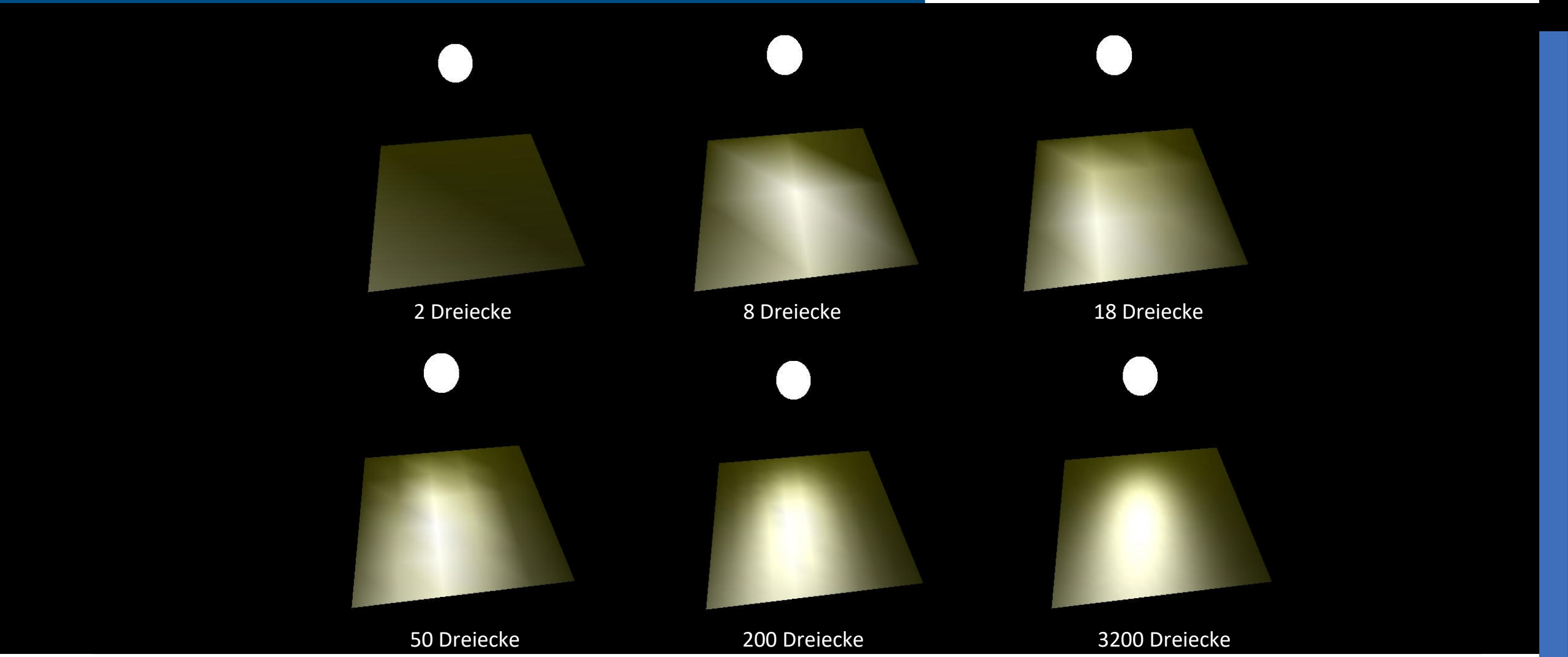
Grenzen des Gouraud Shading Verfahrens

- **Lichtquelle bestrahlt nur die Mitte eines Dreiecks:**

  - Lichtberechnung wird nur an den Eckpunkten berechnet.
  - Dadurch kann für das Dreieck keine Glanzlicht berechnet werden.
  - Außer man macht folgendes …



Glanzlichter mit Gouraud Shading berechnen:



Phong Shading

3. Variante: **Pixelnormalen**

- **Verfahren:**
  - Beleuchtungsberechnung*) wird für jeden einzelnen Pixel durchgeführt.
  - Das bedeutet: Für jeden darzustellenden Pixel muss eine Normale berechnet werden!
  - Die Normalen der Pixel werden auf Basis der Ecknnormalen ermittelt oder durch eine Art Textur (Normal Map).
  - Dadurch können Glanzlichter ohne zusätzliche Unterteilung (Tessilierung) des Gitters berechnet werden.

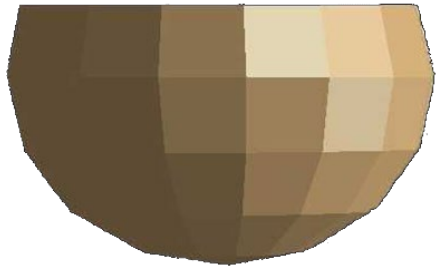
- **Vorteil:**
  - Glanzlichter & gutes visuelles Ergebnis
- **Nachteil:**
  - Normale für jeden Pixel interpolieren kostet viel Rechenzeit

*) Phong-Shading ≠ Phong's Reflexionsmodell

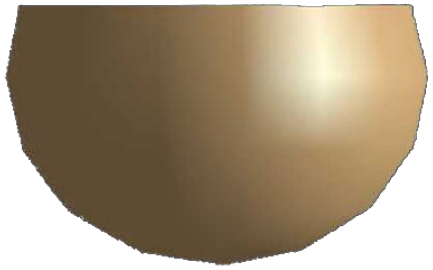
Vergleich der Verfahren

Verfahren:

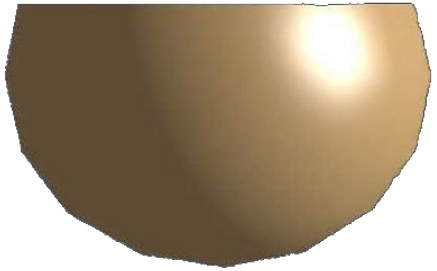
Flat Shading



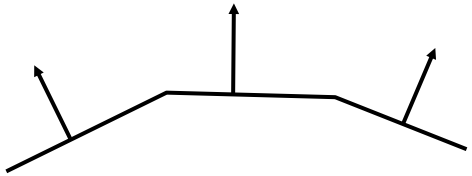
Gouraud Shading

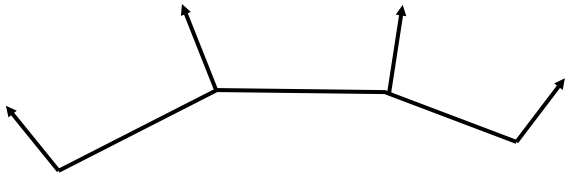


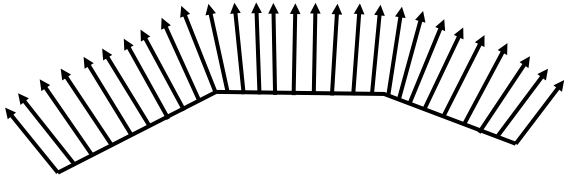
Phong Shading



Normalen:





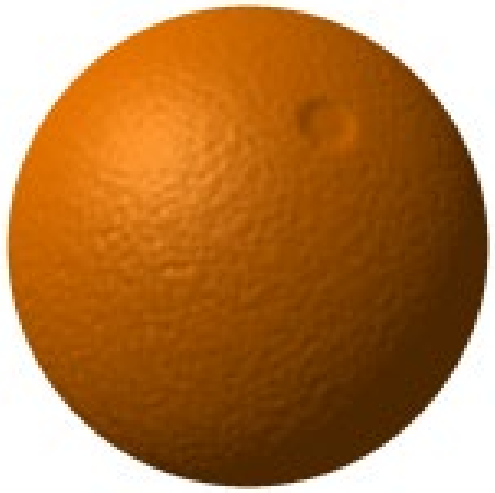


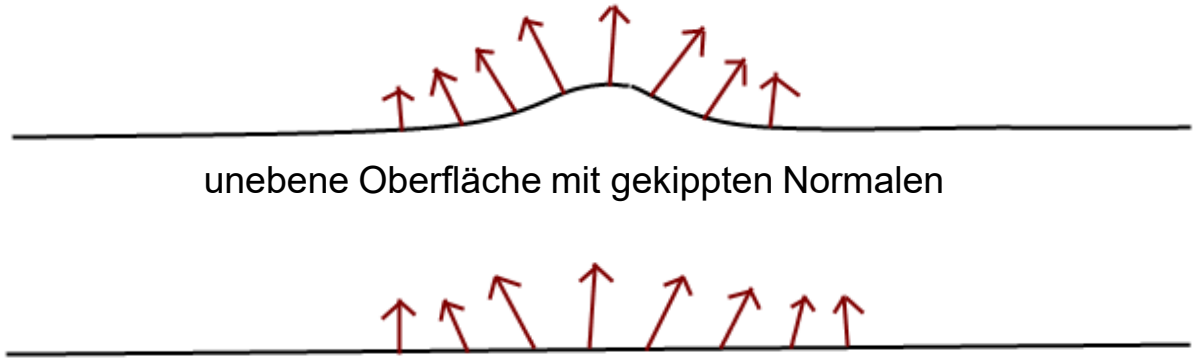
Normal Mapping

Normal Mapping

•Phong Shading als Basisverfahren zur optischen Modellierung einer Oberfläche

- Simulation von Unebenheiten (engl. Bumps) auf der Oberfläche
- Idee: Reine Anpassung der Oberflächennormalen
- → Die Geometrie der Oberfläche wird dabei nicht verändert:



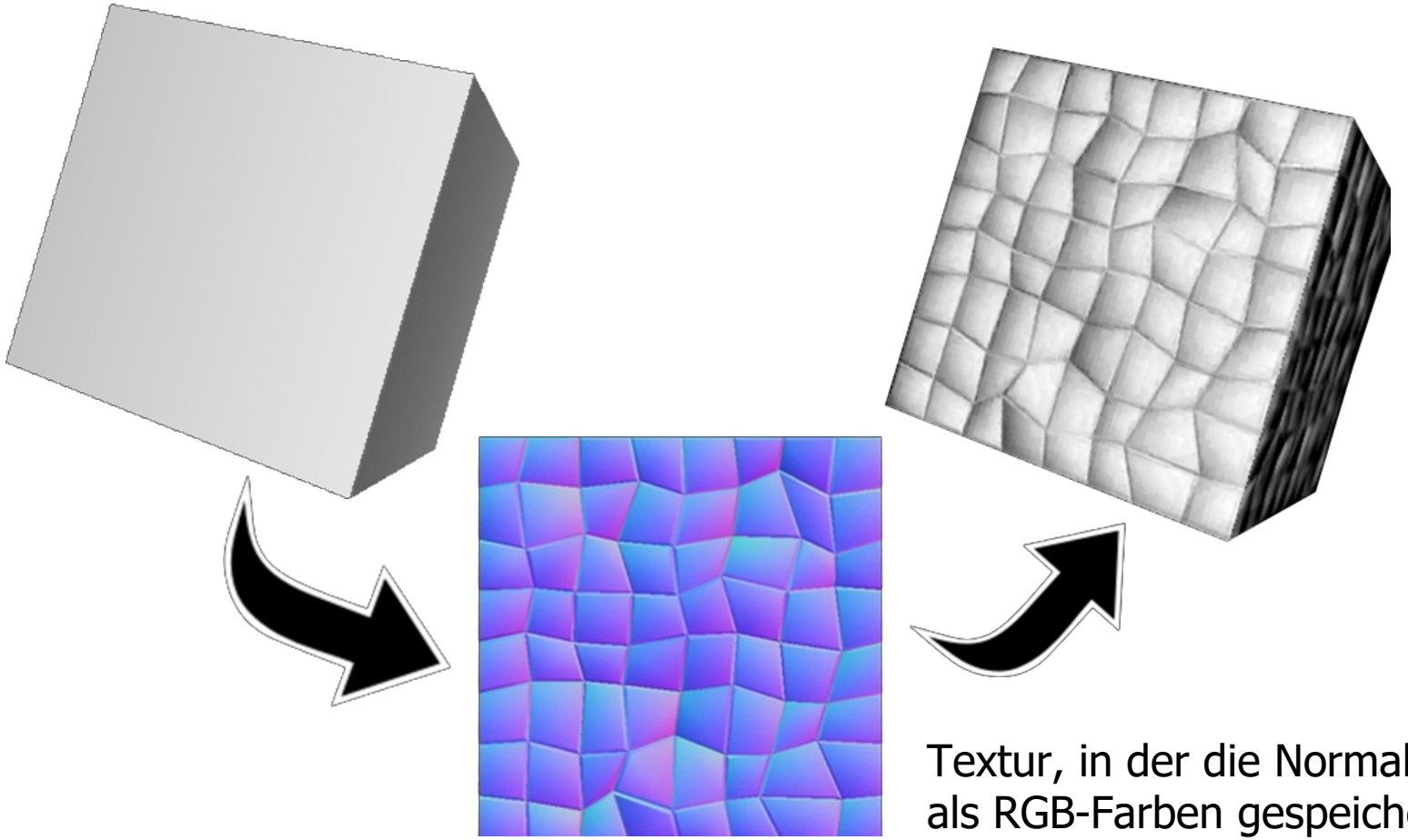


unebene Oberfläche mit gekippten Normalen

Stattdessen:
Oberfläche bleibt flach, nur die Normalen werden angepasst

Quelle: http://mathforum.org/mathimages/index.php/Bump_Mapping

Beispiel für Normal Mapping

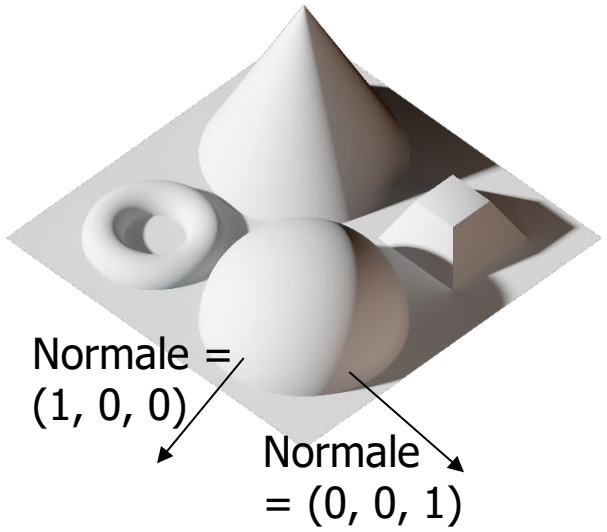


Würfel, dessen Schattierung (Beleuchtungs-reflektion) auf Basis der Normalen (unten) berechnet wurde. Die Geometrie des Würfels wurde dabei nicht verändert.

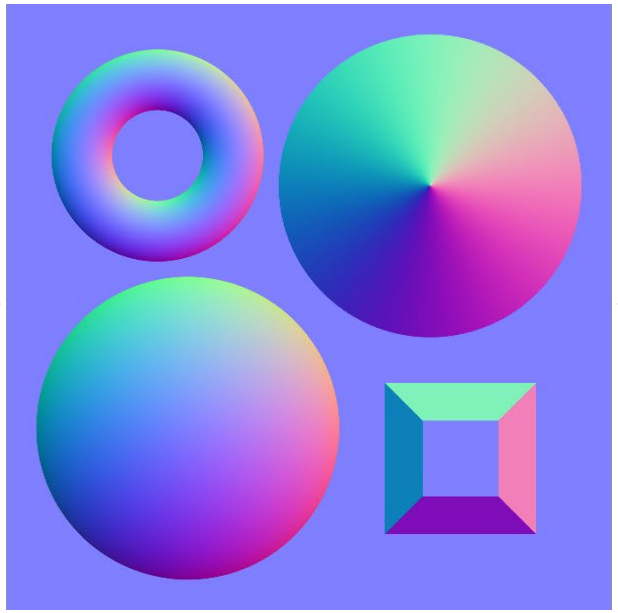
Textur, in der die Normalen pro Pixel als RGB-Farben gespeichert sind

Quelle: http://www.independentdeveloper.com/archive/tag/normal-mapping

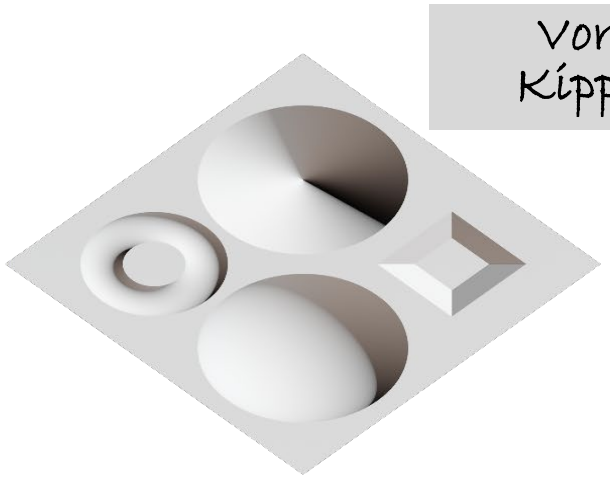
Normal-Mapping am Beispiel einer komplett ebenen Fläche



Geometrie die zur Berechnung der Normalen (2. Abb.) genutzt wird.



Textur, deren RGB-Farben den Normalen entsprechen

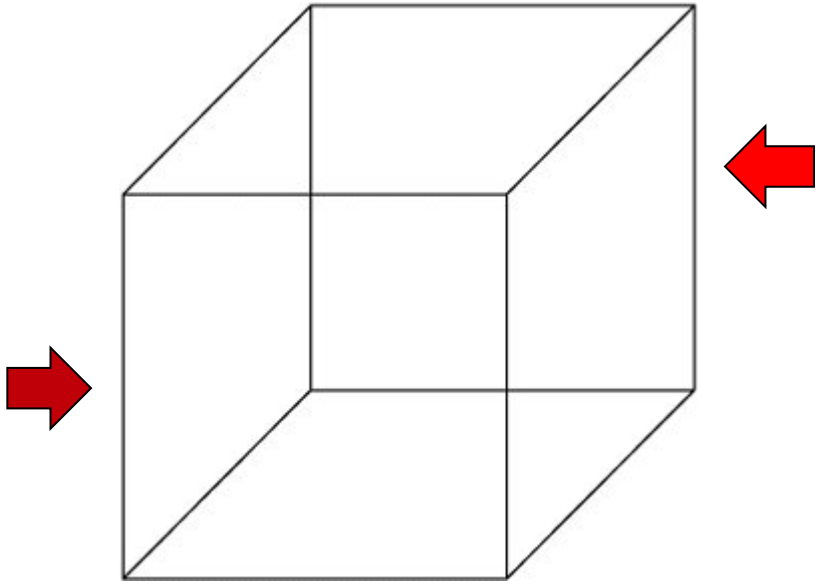


Ein ebenes Viereck, dessen Schattierung auf Basis der Normalen (mittlere Abb.) berechnet wurde.

Quelle: Wikipedia Normal Mapping

Weitere Beispiele für bekannte Kippfiguren …

Junges Mädchen
oder alte Frau?

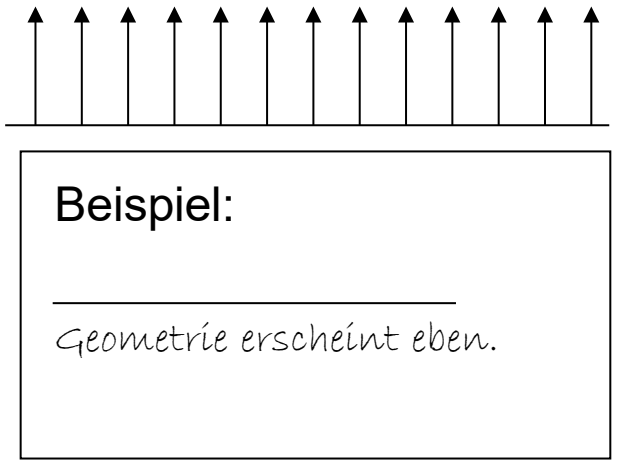


Neckerwürfel
Einmal bildet das vordere Viereck die Oberfläche des Würfels (neben grüner Pfeil) und einmal das hintere Viereck (neben rotem Pfeil).

Übungsaufgabe (Klausur WS 16/17)

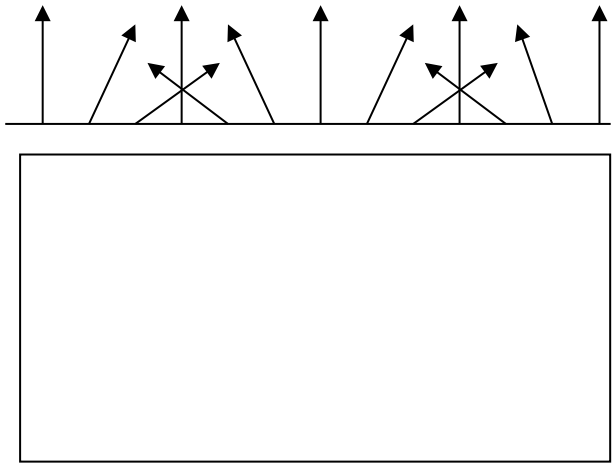
Phong Shading

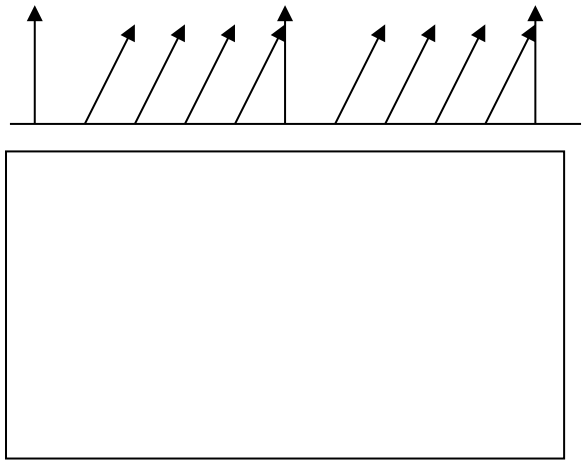
In den drei Abbildungen unten sehen Sie jeweils einen Querschnitt durch eine Oberfläche.
Die Pfeile symbolisieren die Flächennormalen pro Pixel. Durch die anschließende Beleuchtungsberechnung bspw. durch Phong Shading wird die Oberfläche optisch modelliert. Zeichnen und beschreiben Sie den visuellen Eindruck (ebenfalls im Querschnitt), den man nach der Beleuchtungsberechnung aufgrund der eingezeichneten Flächennormalen von der Geometrie haben wird.



Beispiel:

Geometrie erscheint eben.

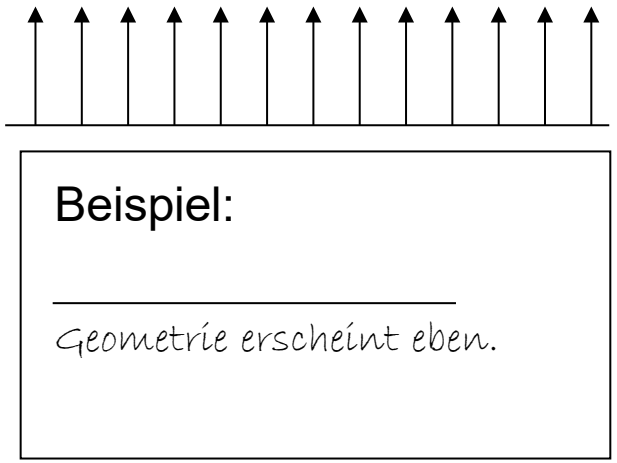


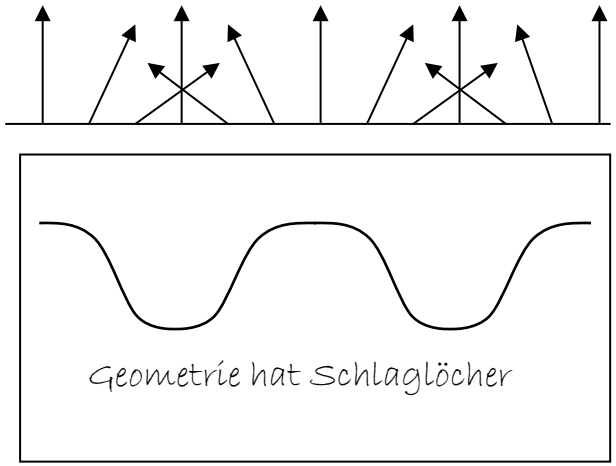


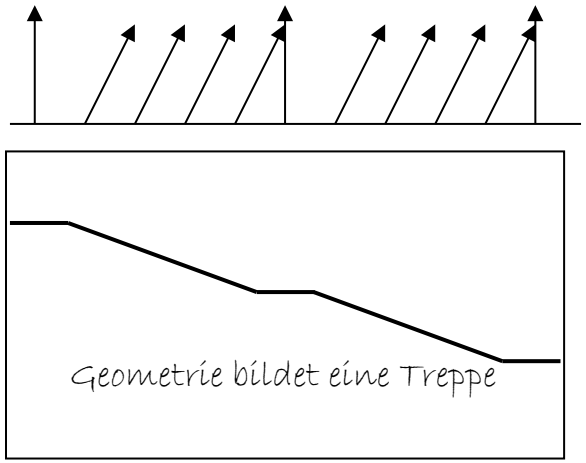
Übungsaufgabe (Klausur WS 16/17)

Phong Shading

In den drei Abbildungen unten sehen Sie jeweils einen Querschnitt durch eine Oberfläche.
Die Pfeile symbolisieren die Flächennormalen pro Pixel. Durch die anschließende Beleuchtungsberechnung bspw. durch Phong Shading wird die Oberfläche optisch modelliert. Zeichnen und beschreiben Sie den visuellen Eindruck (ebenfalls im Querschnitt), den man nach der Beleuchtungsberechnung aufgrund der eingezeichneten Flächennormalen von der Geometrie haben wird.







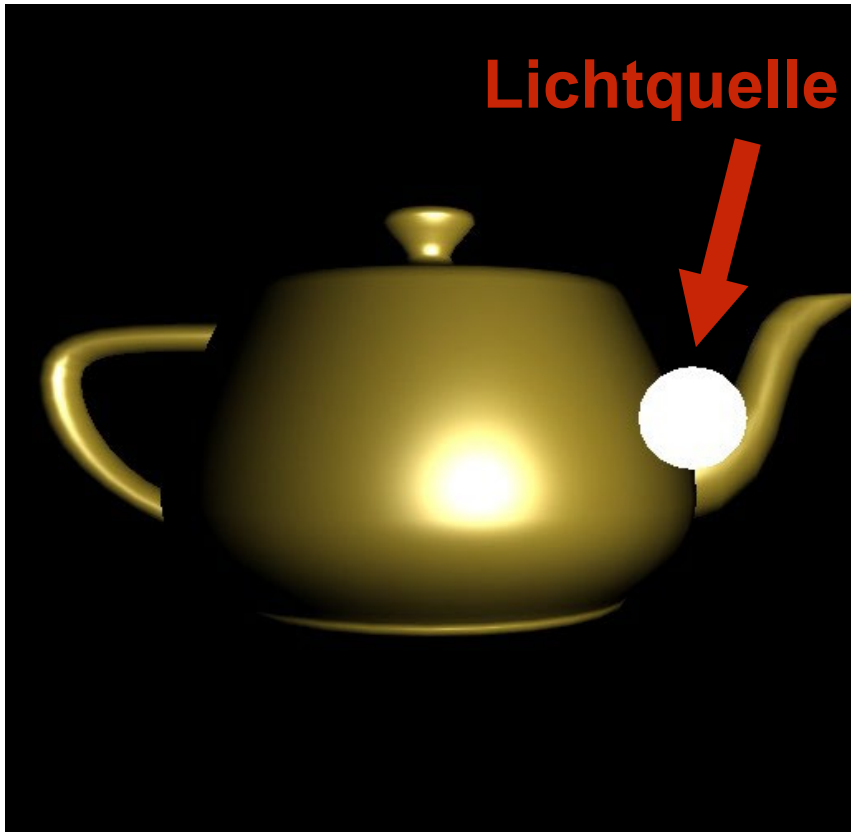
Materialbeschreibung als Textur

- **Verfahren:**
  - Jede Materialeigenschaft wird in einer eigenen Textur gespeichert
    - Diffus
    - Ambient
    - Spiegelnd
    - Selbst leuchtend (emmissiv)
    - Normalen / Tiefenkarte, uvm.

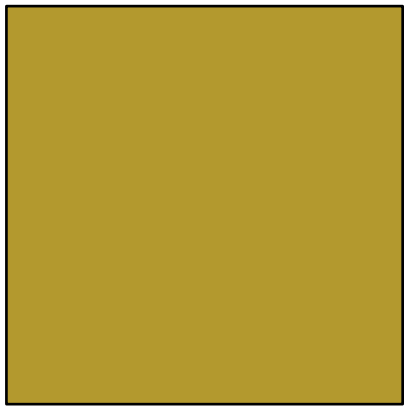
- **Vorteil:**
  - Entkoppelt Materialeigenschaften von den Geometriedaten!

- **Nachteil:**
  - Evtl. erhöhter Speicherbedarf

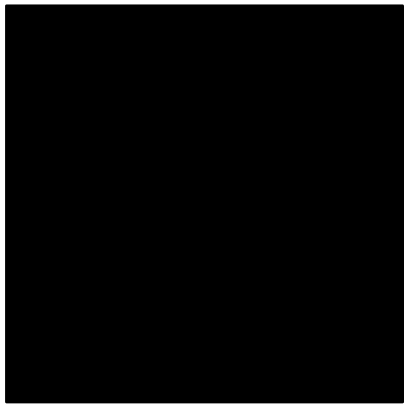
Beispiele:



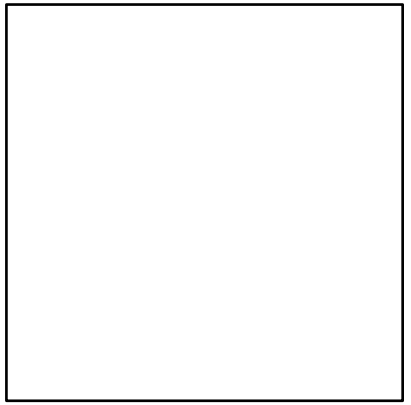
Diffus



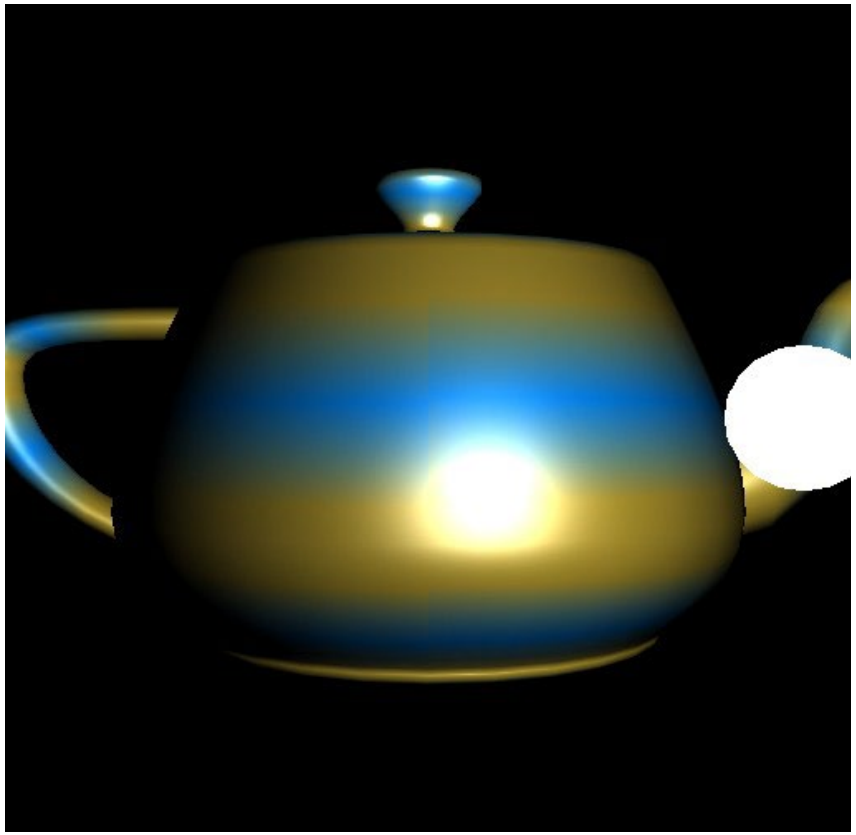
Emissiv



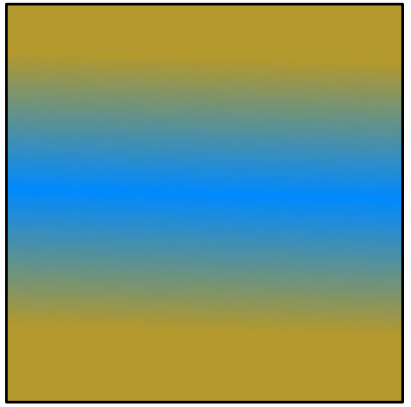
Spekular



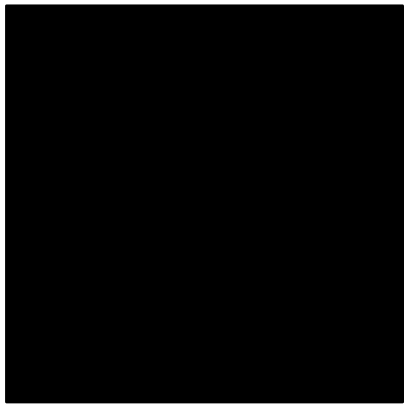
Beispiele:



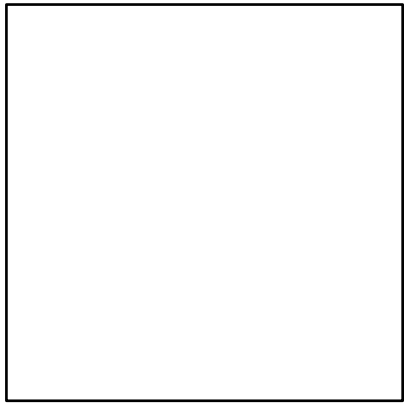
Diffus



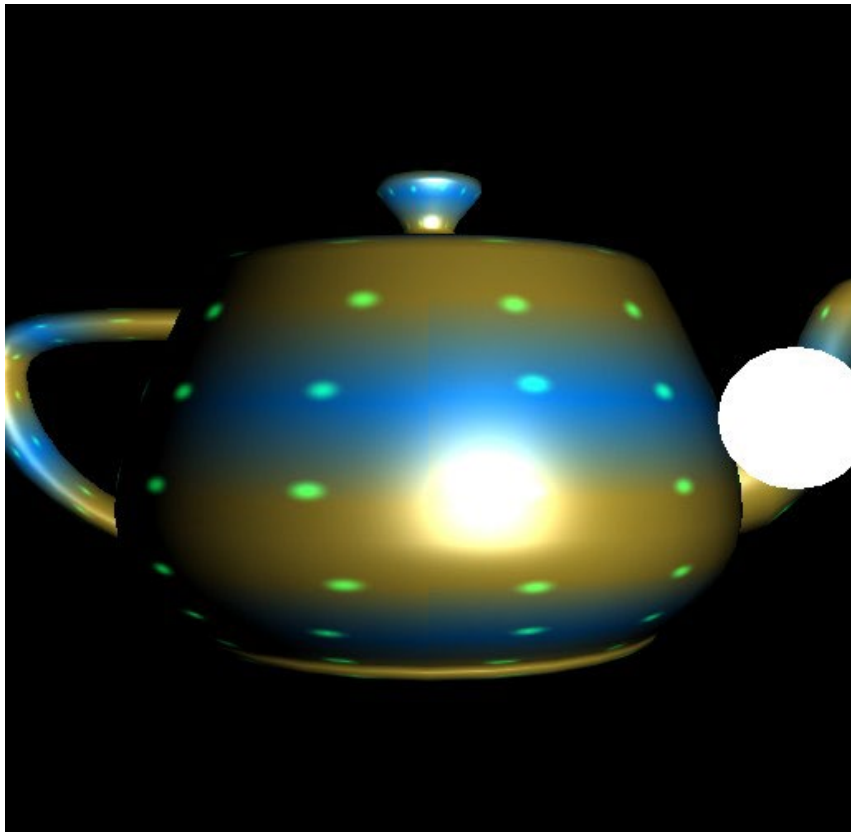
Emissiv



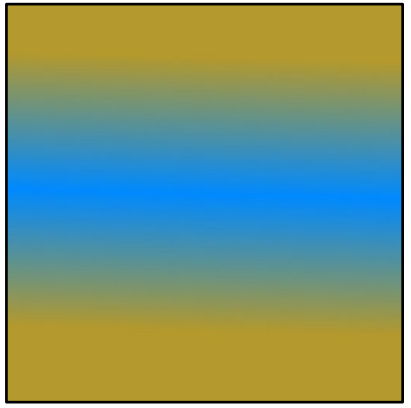
Spekular



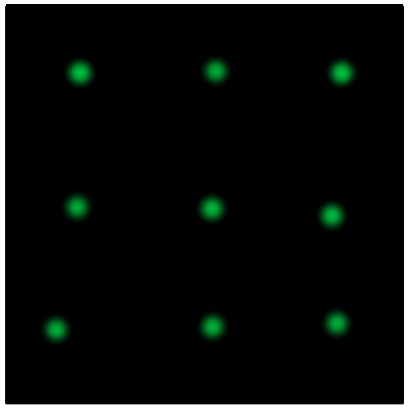
Beispiele:



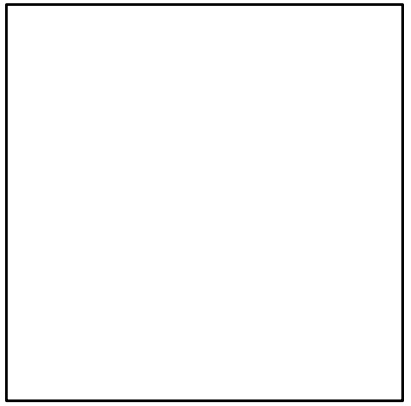
Diffus



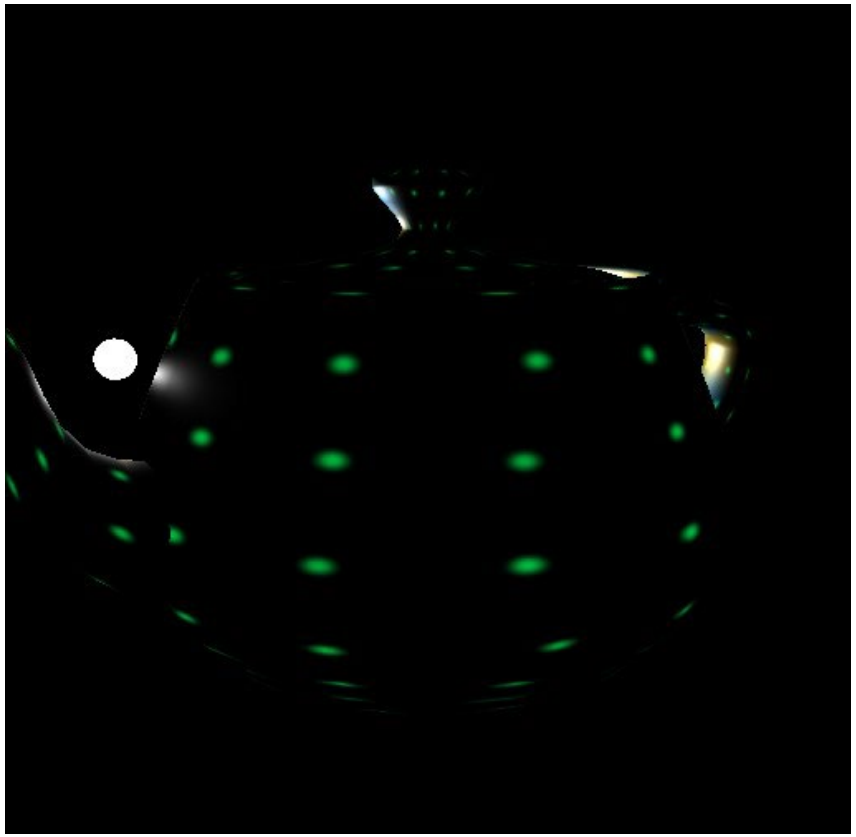
Emissiv



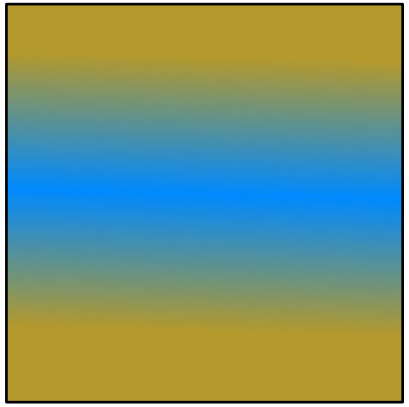
Spekular



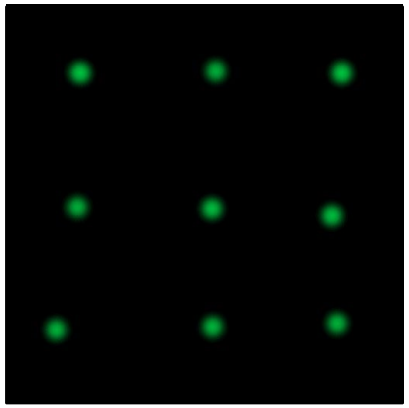
Beispiele:



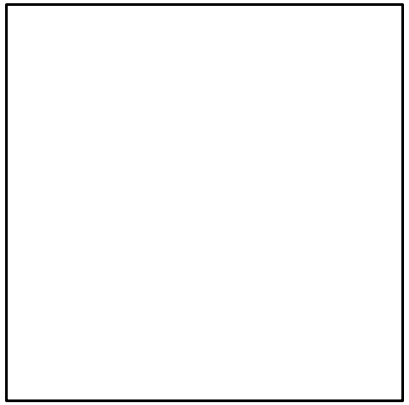
Diffus



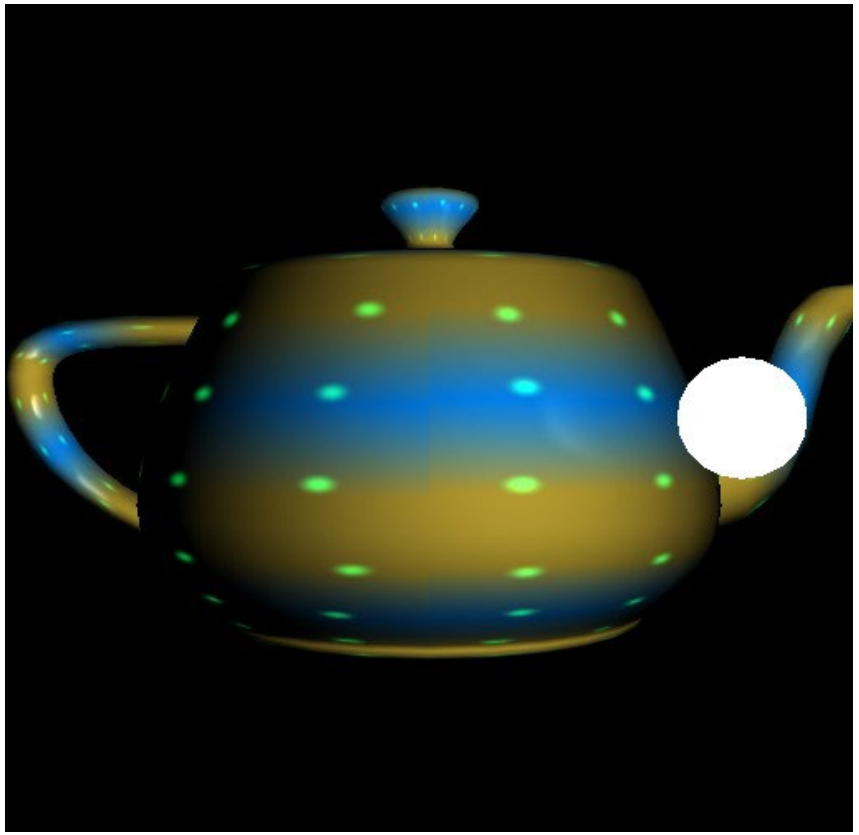
Emissiv



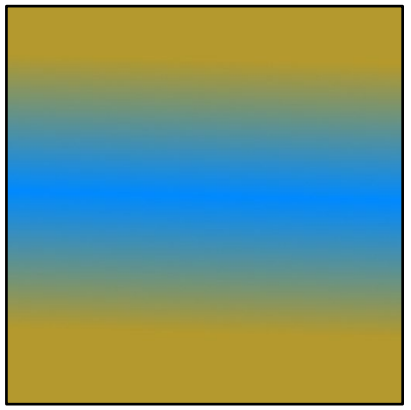
Spekular



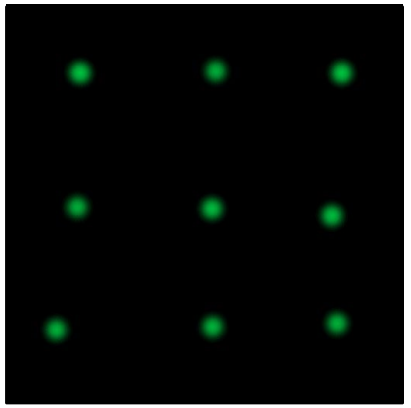
Beispiele:



Diffus

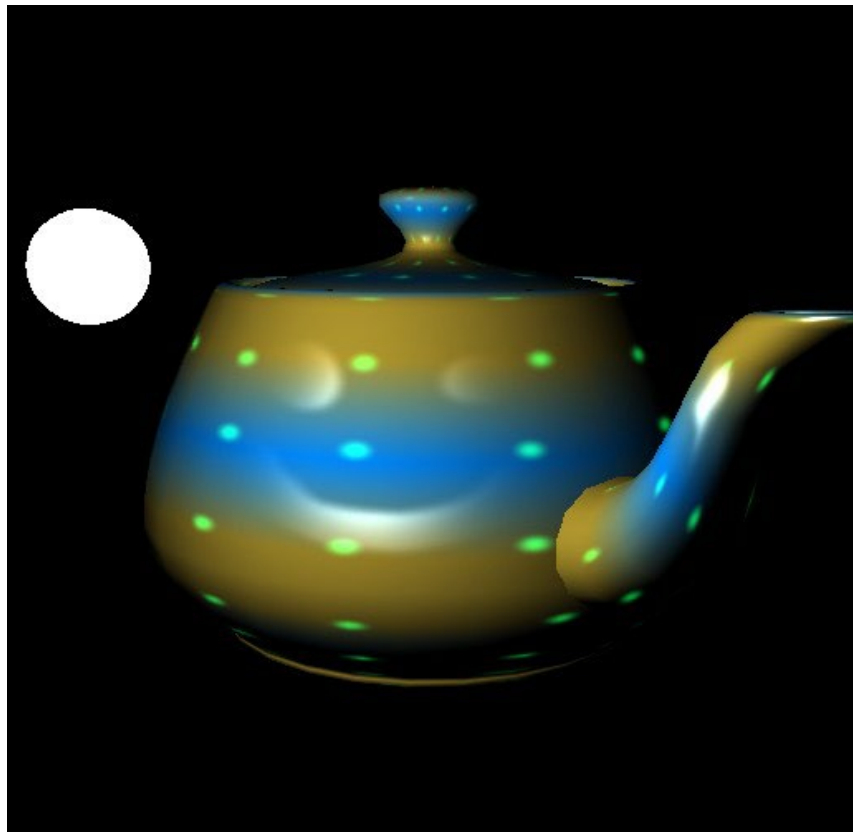


Emissiv

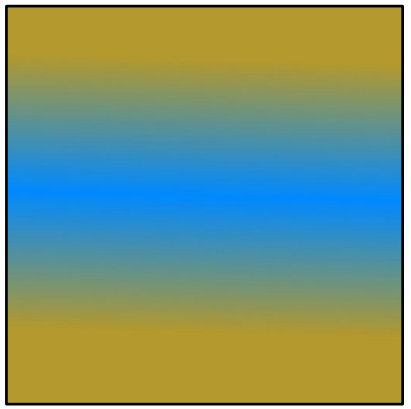


Spekular

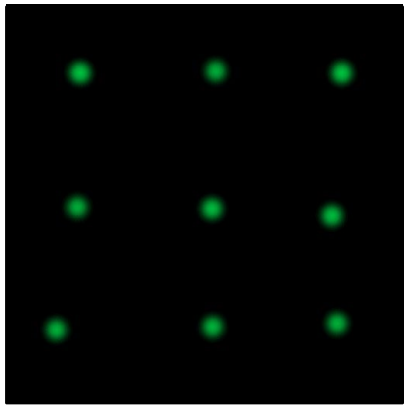
Beispiele:



Diffus



Emissiv

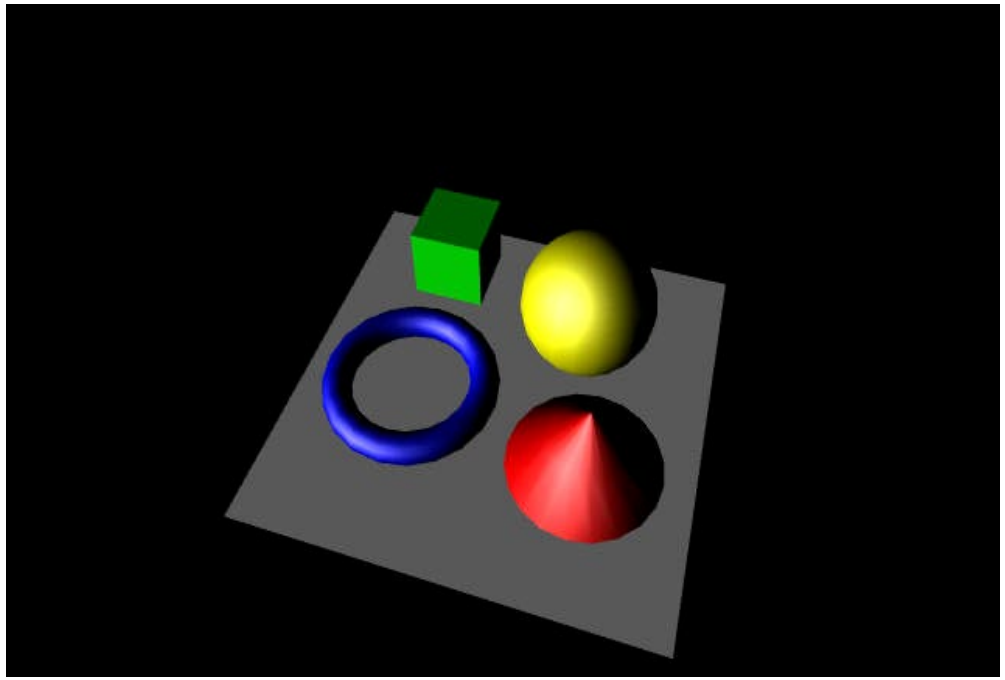


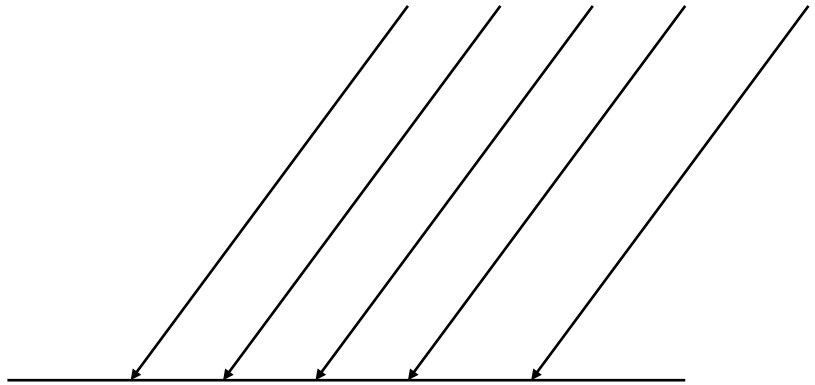
Spekular

Lichtquellen

Gerichtete Lichtquelle

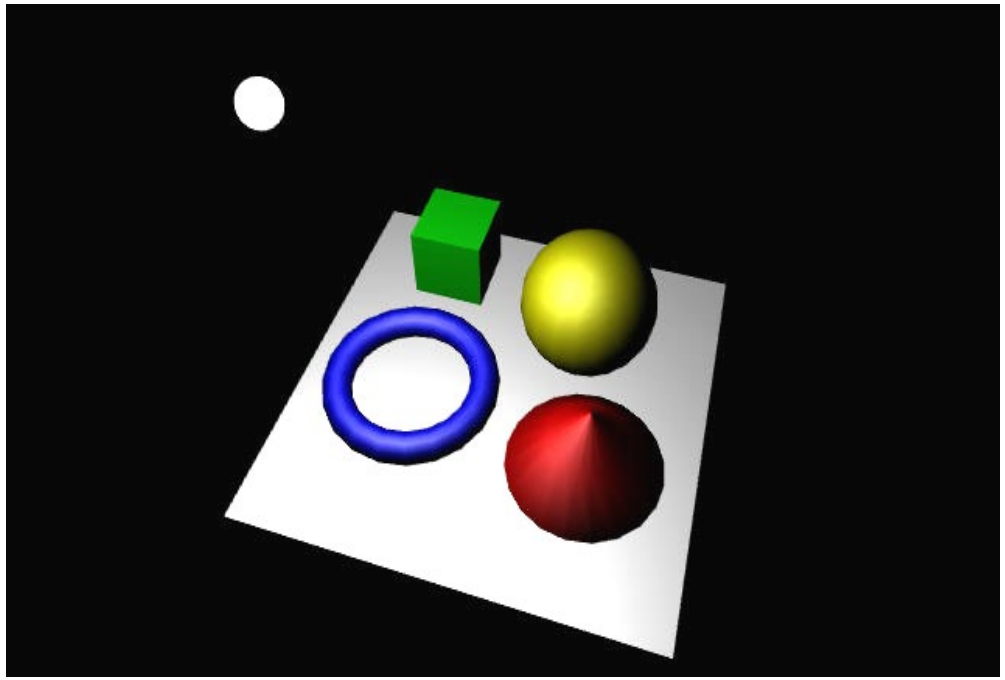
- Entsprechen einer unendlich weit entfernten Lichtquelle
- parallel einfallende Strahlen

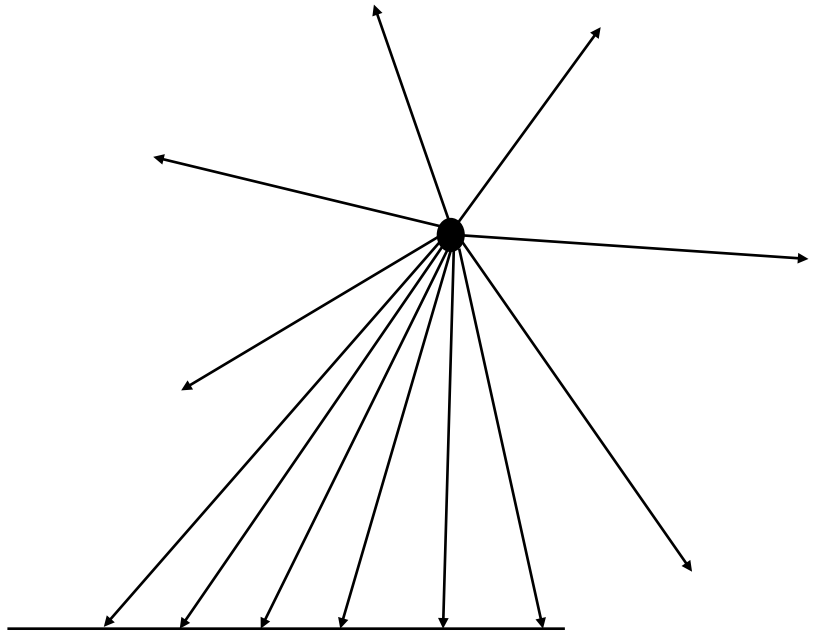




Punktlichtquelle

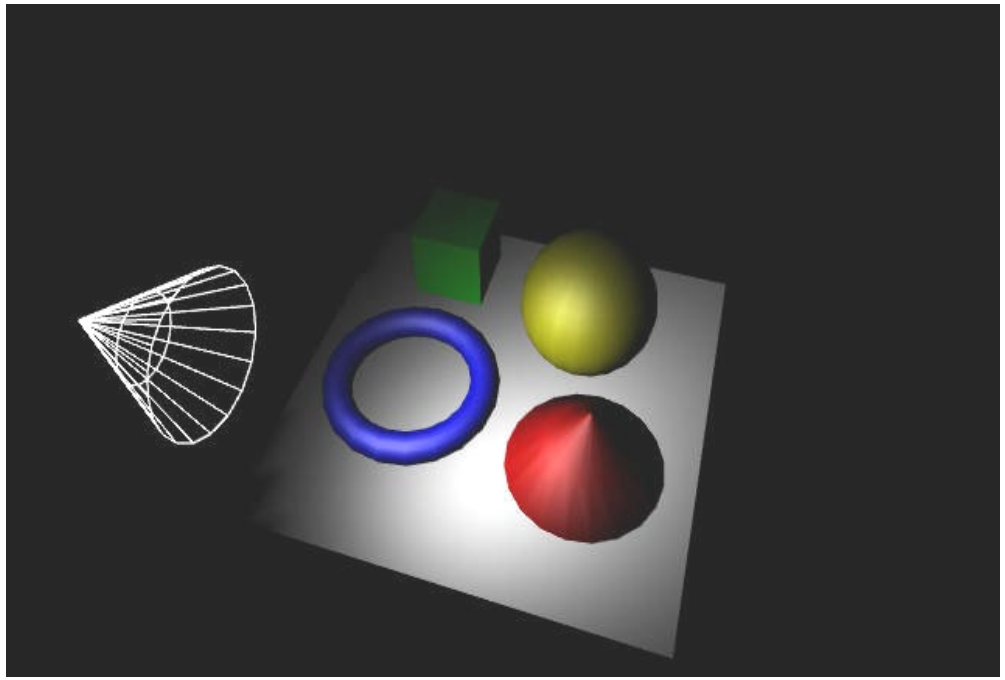
- Ein lokalisierter Punkt strahlt das Licht aus
- unterschiedliche Winkel auf eine ebene Fläche



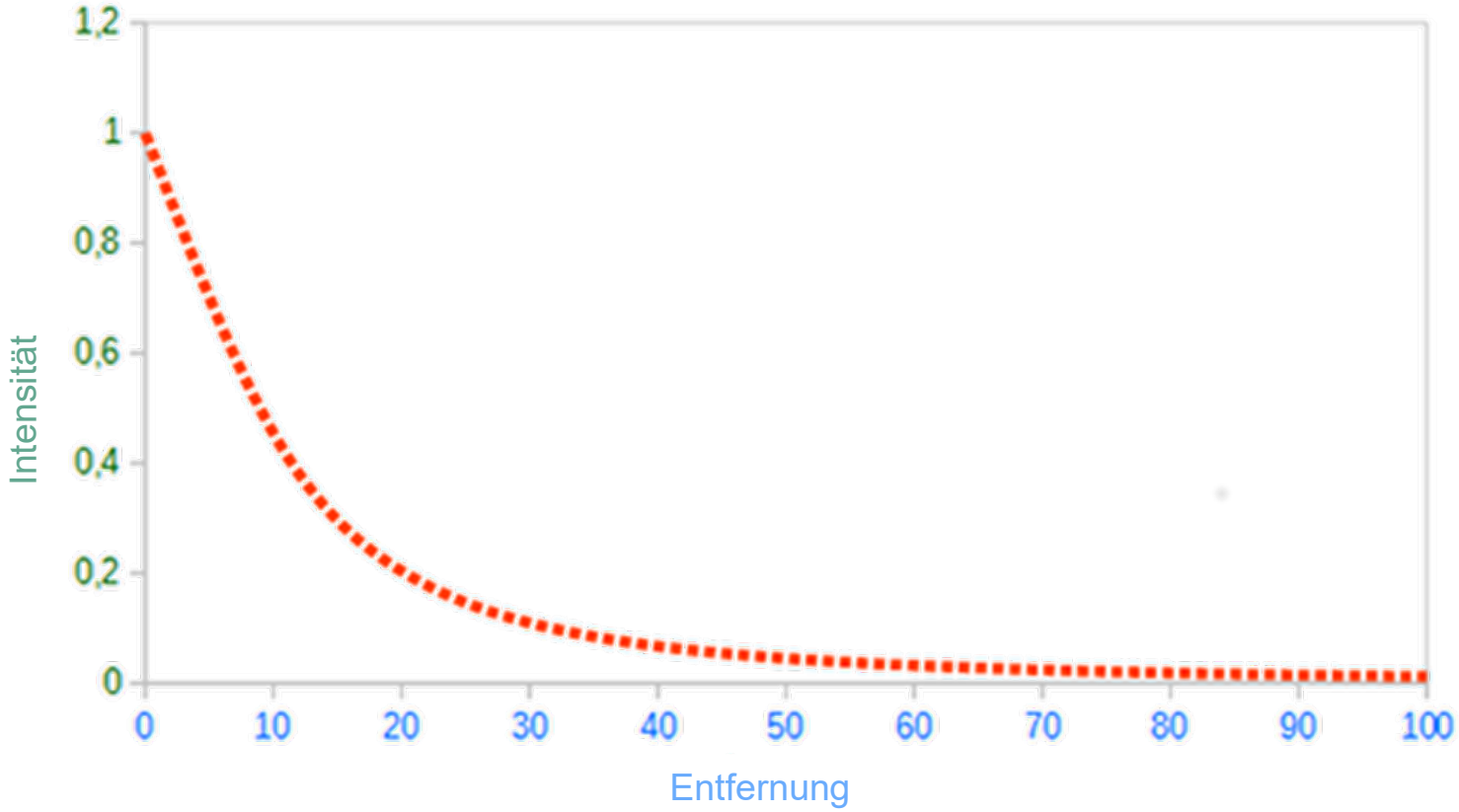


Spotlichtquelle

- Nur um einen bestimmten Winkel um eine angegebene Richtung wird Licht ausgestrahlt.
- je weiter von der Richtung weg desto schwächer wird das Licht



Lichtabschwächung (engl. Light attenuation)



Quelle: https://learnopengl.com/Lighting/Light-casters

Beispielprogramm

Lichtberechnung per Fragment

- Die Berechnung der Beleuchtung in Abhängigkeit von der Blickrichtung sollte im Fragment-Shader erfolgen!

Berechnung Per-Vertex

Berechnung Per-Fragment

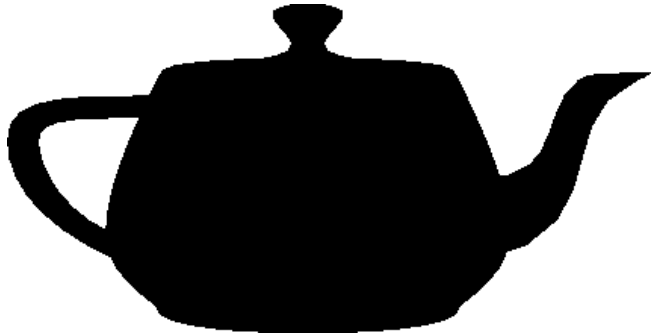
Ohne Beleuchtung

Vertex Shader

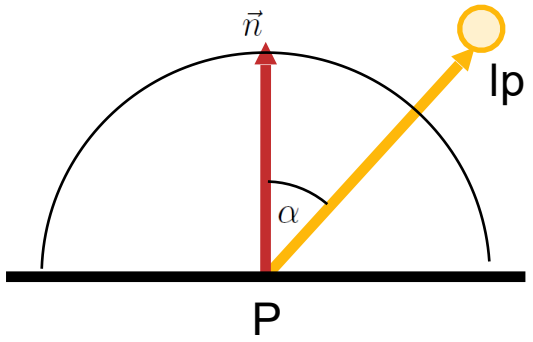
```
#version 330
layout(location = 0) in vec3 vertex;
layout(location = 1) in vec3 vertex_color;

uniform mat4 modelMatrix;
uniform mat4 viewMatrix;
uniform mat4 projMatrix;

out vec3 colorVS;

void main() {
    gl_Position = projMatrix * viewMatrix * modelMatrix * vec4(vertex, 1.0);
    colorVS = vertex_color;
}
```

Fragment Shader

```
#version 330
in vec3 colorVS;
out vec4 color;
void main() {
    color = vec4(colorVS, 1.0);
}
```



Diffuser Term



$$\begin{pmatrix} M_r \\ M_g \\ M_b \end{pmatrix} \cdot \begin{pmatrix} L_r \\ L_g \\ L_b \end{pmatrix} \cdot \cos\alpha$$

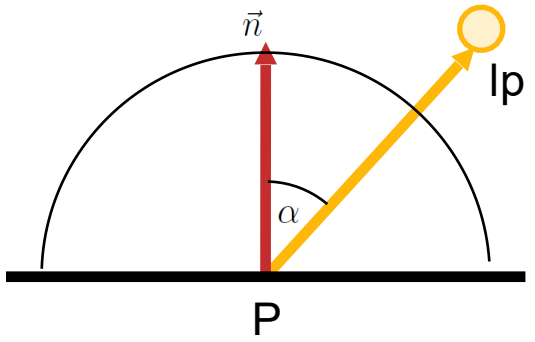
Vertex Shader

```
[…]
layout(location = ...) in vec3 vertex_normal;
uniform vec3 lightPos;
out vec3 normal, lightDir;

void main() {
        […]
        // compute normal in view space //
        vec4 n = vec4(vertex_normal, 0.0);
        mat4 normalMat = transpose(inverse(viewMatrix * modelMatrix));
        normal = (normalMat * n).xyz;

        // compute light direction in view space //
        vec4 lp = vec4(viewMatrix * vec4(lightPos, 1.0));
        vec4 P = vec4(viewMatrix * modelMatrix * vec4(vertex, 1.0));
        lightDir = (lp - P).xyz;
}
```

Diffuser Term

Fragment Shader

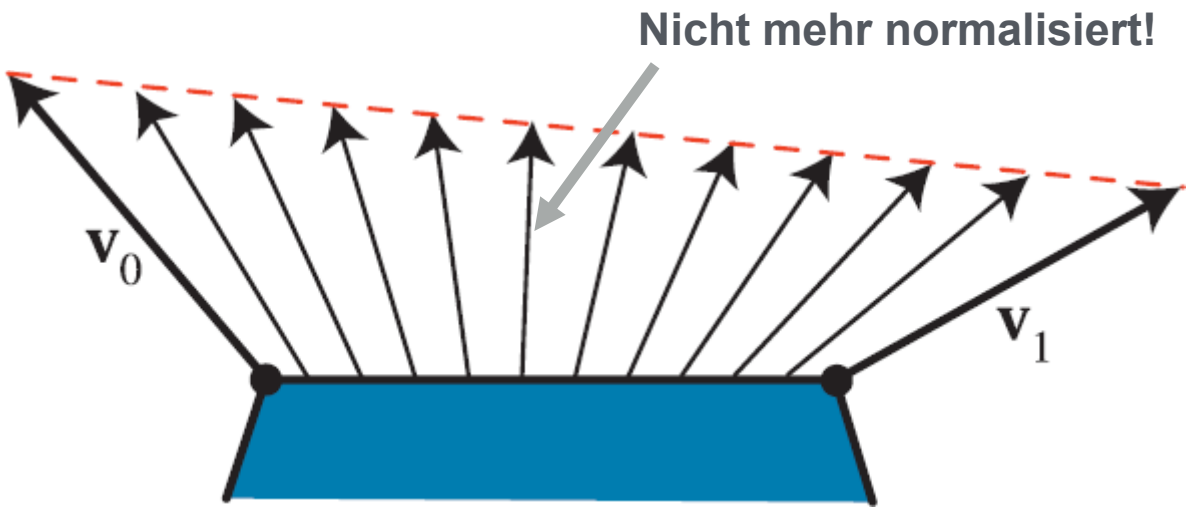
```
[…]
in vec3 normal, lightDir;
uniform vec3 lightColor;

void main() {
    […]
    // normalize everything necessary //
    vec3 N = normalize(normal);
    vec3 L = normalize(lightDir);

    // diffuse component //
    float cosa = max(0.0, dot(N,L));
    vec3 diffuseTerm = colorVS * lightColor
    color = vec4(diffuseTerm * cosa, 1.0);
}
```



$$\begin{pmatrix} M_r \\ M_g \\ M_b \end{pmatrix} \cdot \begin{pmatrix} L_r \\ L_g \\ L_b \end{pmatrix} \cdot \cos\alpha$$

Interpolation der Normalen

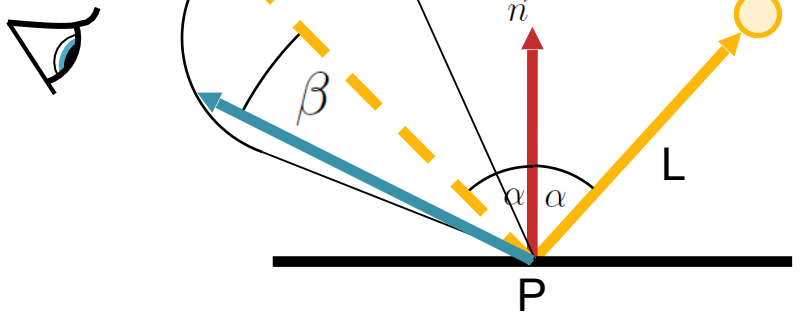
- Anzahl der Vertices < Anzahl der Fragmente
- Warum können die Vektoren nicht direkt im Vertex-Shader normalisiert werden?



Spekularer Term

Vertex Shader

```
[…]
uniform vec3 lightPos;
out vec3 Normal, LightDir, viewDir;

void main() {
        […]
        // specular term //
        viewDir = -P.xyz;
}
```



$$\begin{pmatrix} M_r \\ M_g \\ M_b \end{pmatrix} \cdot \begin{pmatrix} L_r \\ L_g \\ L_b \end{pmatrix} \cdot \cos^k \beta$$

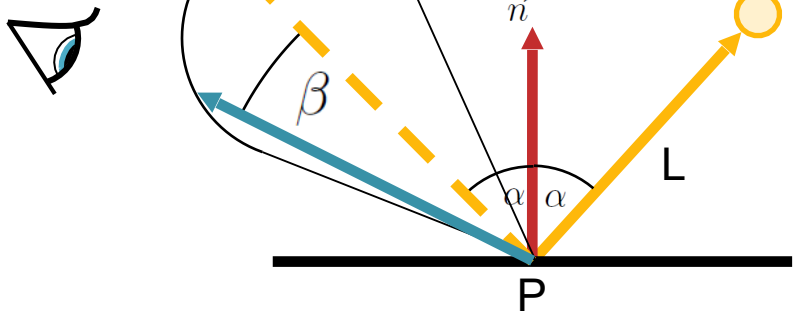
Warum ist die Blickrichtung die negative Vertexposition im ViewSpace?

Weil sich die Kamera im ViewSpace im Ursprung befindet!

Spekularer Term

Fragment Shader

```
[…]
in vec3 normal, lightDir, viewDir;

uniform vec3 matSpecular;
uniform float matShininess;

void main() {
        […]
        // specular component//
        vec3 V = normalize(viewDir);
        vec3 R = normalize(reflect(-L,N));

        float cosBeta = max(0.0, dot(R,V));
        float cosBetak = pow(cosBeta, matShininess);

        vec3 specularTerm = matSpecular * lightColor;
        color += vec4(specularTerm * cosBetak, 1.0);
}
```

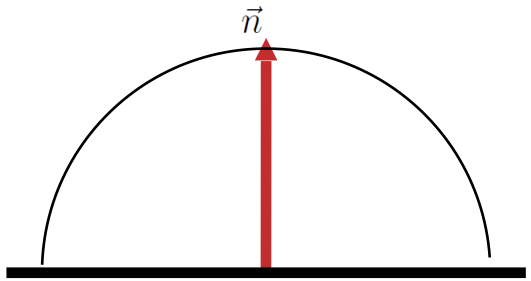


$$\begin{pmatrix} M_r \\ M_g \\ M_b \end{pmatrix} \cdot \begin{pmatrix} L_r \\ L_g \\ L_b \end{pmatrix} \cdot \cos^k \beta$$

Ambienter Term

Vertex Shader

```
[…]
uniform vec3 lightPos;
out vec3 normal, lightDir;

void main() {
[…]
}
```



Ambienter Term

Fragment Shader

```
[…]
in vec3 normal, lightDir;
uniform vec3 lightColor;

void main()
        […]
        // ambient component //
        float ambientStrength = 0.2;
        vec3 ambientColor = colorVS; // this could also be a global ambient color
        vec3 ambientTerm = ambientColor * lightColor * ambientStrength;

        color += vec4(ambientTerm, 1.0);
}
```