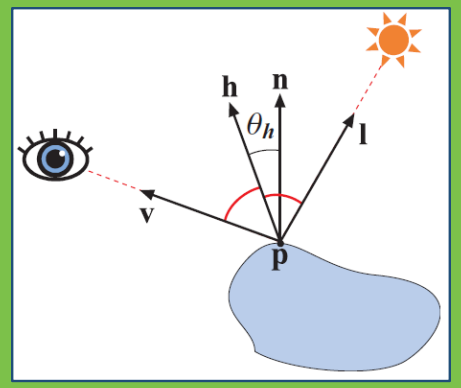
Visual Computing – Beleuchtungsrechnung

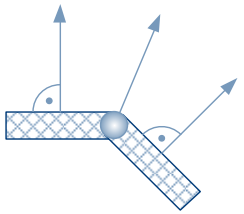


Yvonne Jung

Beleuchtungsrechnung

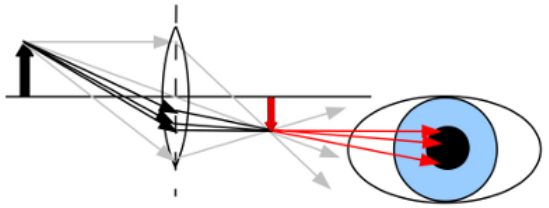
h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

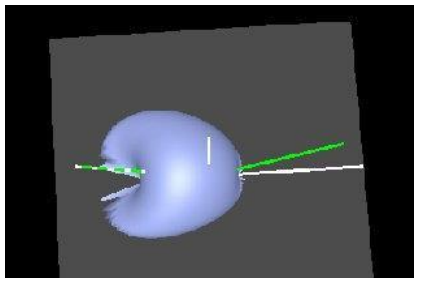
- Simulation der Beleuchtung von 3D-Objekten
- Ziel: Plausible Bilder mit akzeptablem Rechenaufwand
- Probleme
  - Interaktion von Licht und Materie (→ Physik)
  - Beleuchtung erfolgt durch Lichtquellen oder Objekte, die Licht reflektieren (z.B. Spiegel, helle Flächen)
  - Aussehen eines Objekts abhängig von Beleuchtung, Material sowie Position und Richtung der Kamera
    - Geometrie (Position und Richtung) der Oberflächen sowie der Lichtquellen berechnen
  - Polygonale Modelle wirken facettiert
    - Müssen visuell geglättet werden (durch gemittelte Normale)

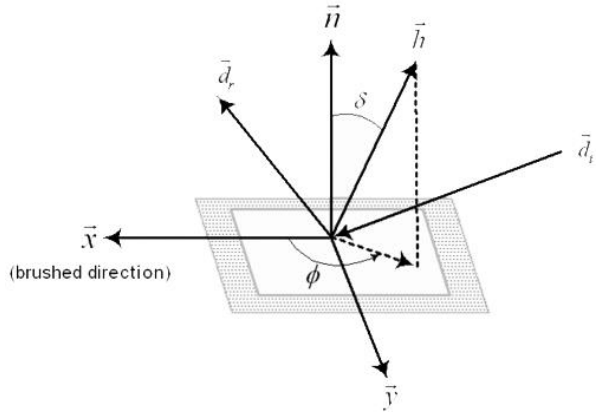


Licht und Materie

- Licht ist elektromagnetische Strahlung
  - Zwei für Graphik bedeutende duale Modelle
    - Strahlenoptik: geradlinige Ausbreitung (einfach)
    - Wellenmodell (seltener nötig, meist nur zum Verstehen von Farbe, Interferenz, Polarisation)
    - (Quantenphysik: in Computergraphik kaum benutzt, außer z.B. für Photon Mapping)
- Verschiedene Materialien
  - homogen – inhomogen
  - leitend – isolierend
  - isotrop – anisotrop

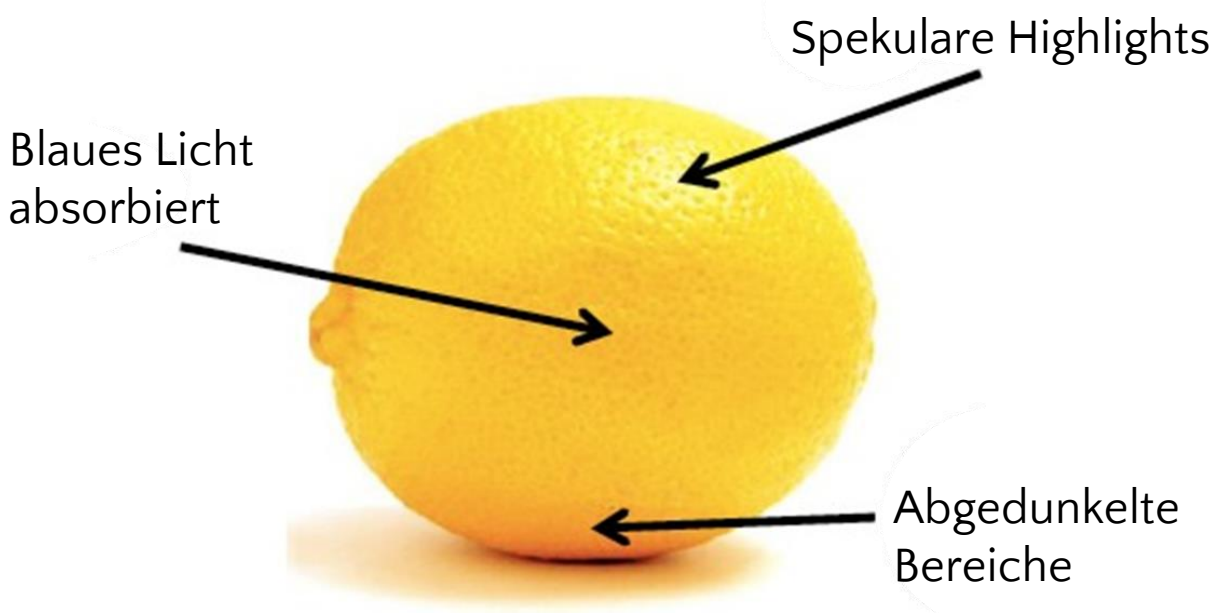






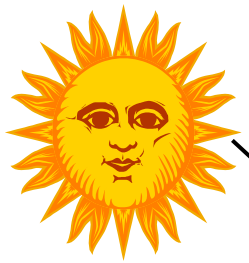
Entstehung des Farbreizes

- Lichtquellen strahlen Licht ab
  - Mit bestimmter Intensität und Farbe
- Reflektanz von Gegenständen
  - Bezeichnet Teil des Lichtes, der vom beleuchteten Gegenstand reflektiert wird
  - Beispiel: Zitrone erscheint deshalb gelb, weil blauer Teil des Lichts absorbiert wird



Entstehung des Farbreizes

- Strahlung, die Farbempfindung auslöst



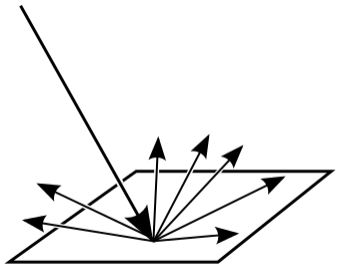
Licht

$E(\lambda)$

Beobachter

Reflektiertes Licht

$L(\lambda) = E(\lambda) \cdot \rho(\lambda)$

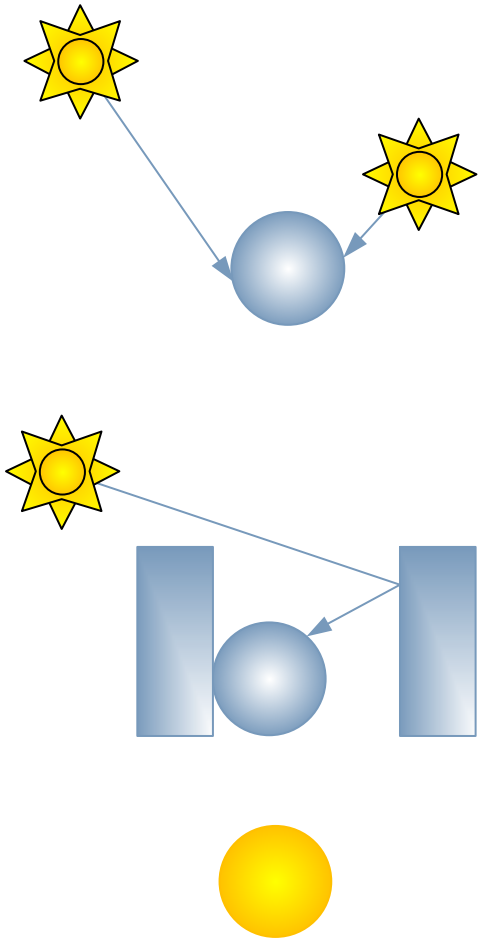


Reflektanz des Objekts $\rho(\lambda)$

Spektralverteilung E(λ)
Strahldichte L(λ)

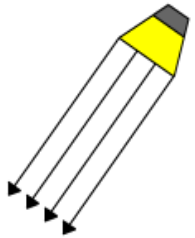
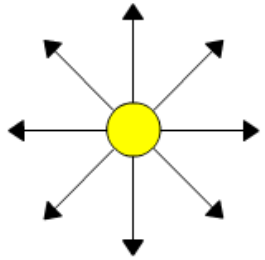
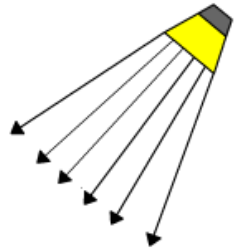
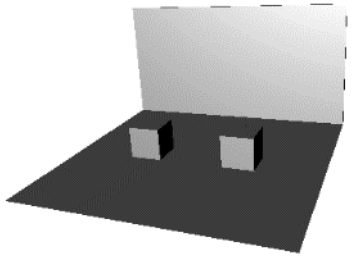
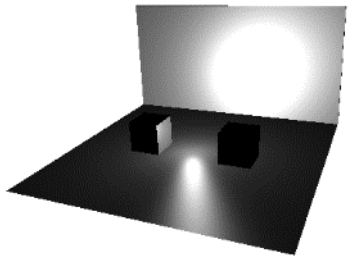
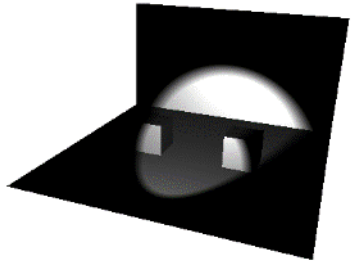
Lichtarten

- Direktes Licht
  - Objekt wird direkt von Lichtquellen bestrahlt
- Indirektes Licht
  - Objekt wird von reflektierten Lichtstrahlen getroffen
- Emittiertes Licht
  - Objekt selbst ist ein leuchtender Körper
- Lichtquellen emittieren Licht
  - Haben Intensitäten (und Farbe)
  - Haben eigentlich Ausdehnung (Fläche), aber für Echtzeit-Graphik zur Vereinfachung als punktförmig angenommen
- Objekte reflektieren Licht
  - Materialien haben Reflexionseigenschaften



Idealisierte Standard-Lichtquellen

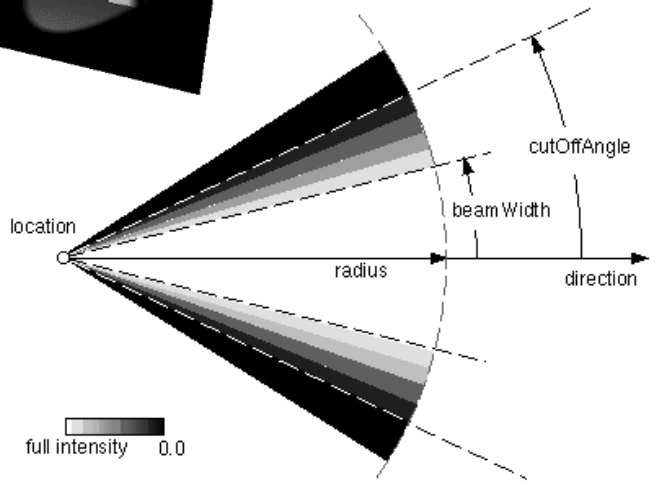
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

	Direktionale Lichtquelle	Punkt-Lichtquelle	Spot-Lichtquelle
			
Parameter:	Richtung, Intensität, Farbe	Position, Intensität, Farbe	Position, Richtung, Intensität, Farbe
			

Achtung: gibt es in Realität alle nicht!

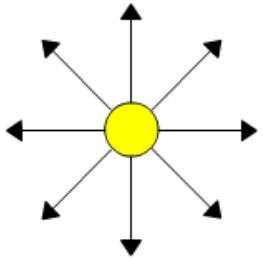
Intensitätsberechnung Spotlight:

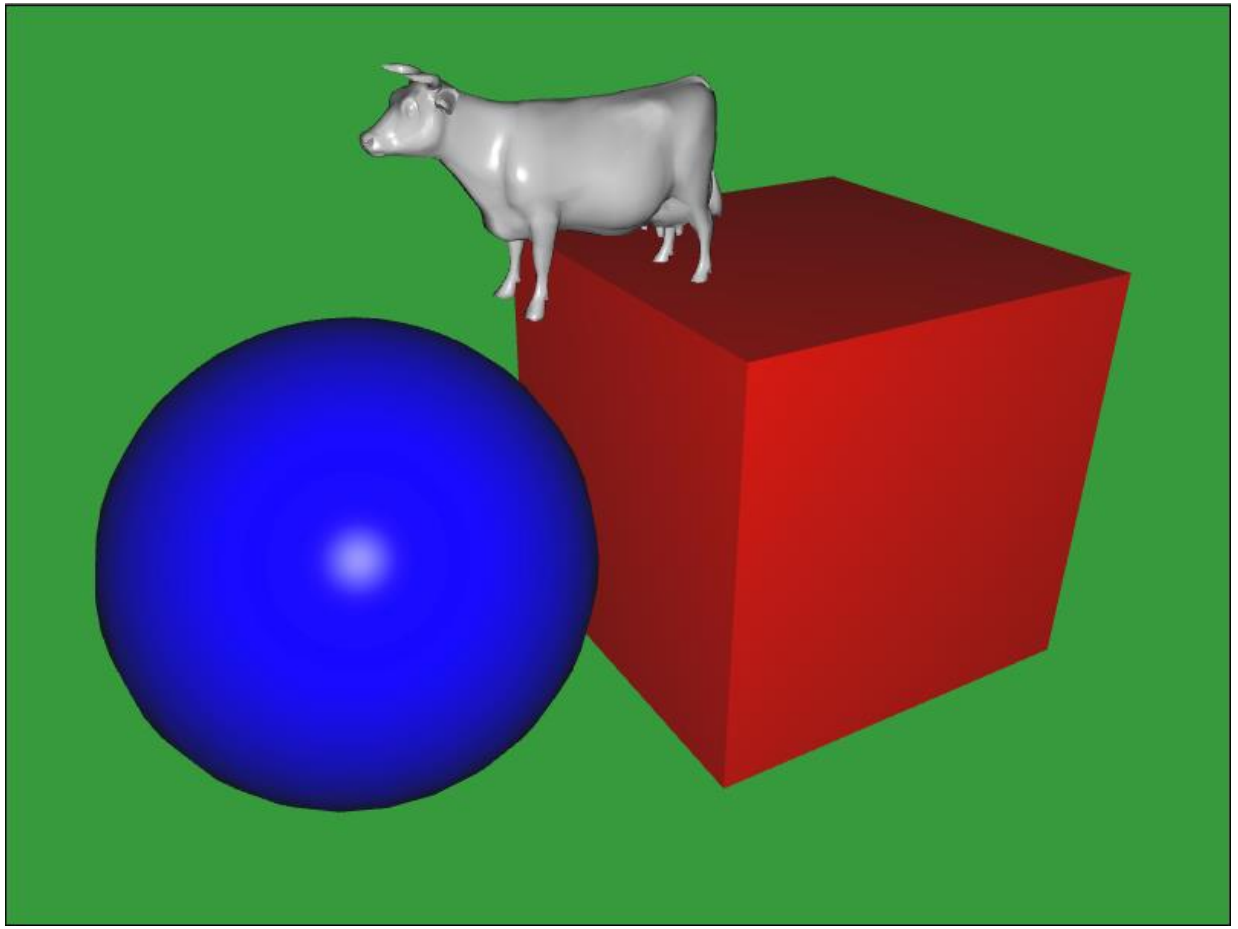
```
if (angle ≥ cutOffAngle) multiplier = 0
else if (angle ≤ beamWidth) multiplier = 1
else multiplier = (angle-cutOffAngle)/(beamWidth-cutOffAngle)
intensity = intensity × multiplier
```



Headlight

- Punktlichtquelle an Kameraposition
  - Lichtrichtung identisch Blickrichtung





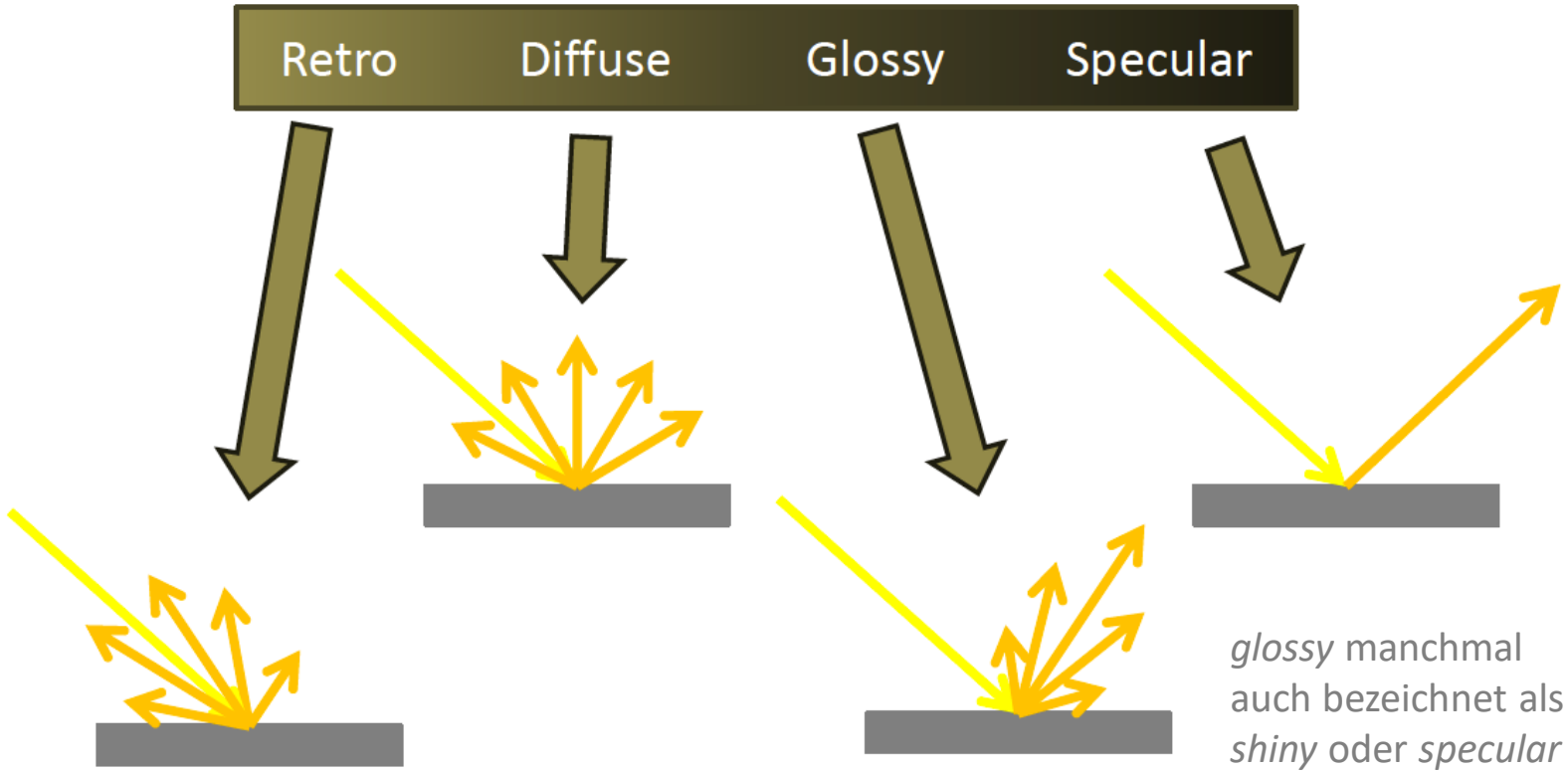
Interaktion von Licht und Oberflächen

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

- Emission
  - Material leuchtet selbst (z.B. Glühen bei Hitze)
- Absorption
  - Material schluckt Licht (z.B. bei Nebel)
- Scattering
  - Material streut Licht (Grundlage für Sehen)
    - Bestimmt durch optische Eigenschaften des Materials
  - Typen: volumetrisch, Subsurface-, und Surface-Scattering
    - Bei letzterem wichtig sind Transmission und Reflexion
    - Bei Reflexion wiederum: Spiegelnde, diffuse, glossy und retro-reflektive Reflexion

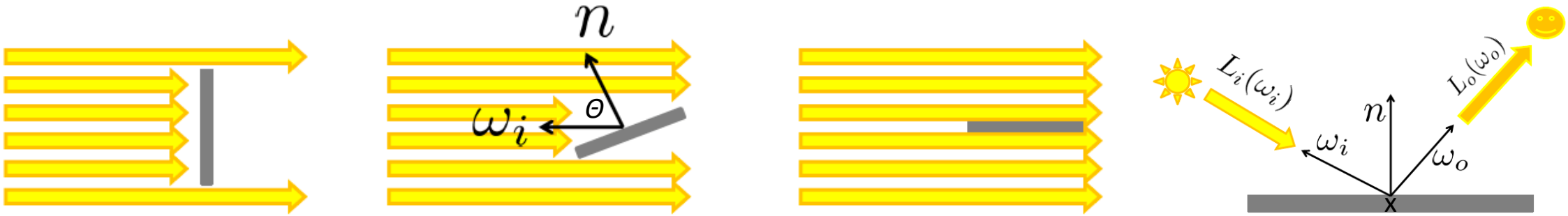
Lichtstreuung auf Oberflächen

- Retro-reflektive, diffuse, glossy u. spiegelnde Reflexion
  - Im folgenden nur lokale Reflexion an einem Punkt betrachten



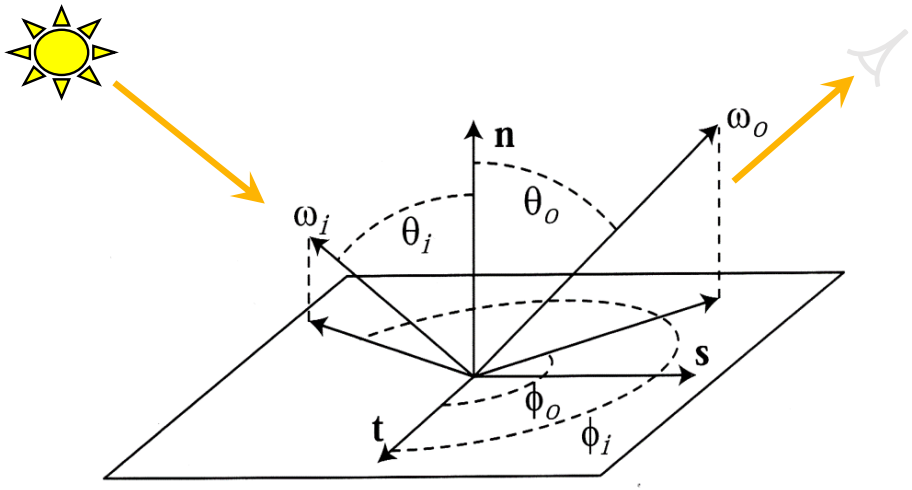
glossy manchmal auch bezeichnet als *shiny* oder *specular*

Lichtstreuung auf Oberflächen



- Oberfläche erhält weniger einfallendes Licht, wenn Einfallswinkel größer wird (abgeschwächt mit $\omega_i \cdot n$)
  - Einfallendes Licht ergibt sich aus $L_i(x, \omega_i)(\omega_i \cdot n)$ — *Skalarer Fluss Φ des Vektorfelds*
- BRDF $f(x, \omega_i, \omega_o)$ beschreibt Reflexionseigenschaften an Punkt x
  - Verhältnis von reflektierter Strahldichte (irradiance) $L_o(x, \omega_o)$ zu einfallender Bestrahlungsstärke (radiance) $L_i(x, \omega_i)$
  - Reflexion bei einer Lichtrichtung: $L_o(x,\omega_o) = f(x, \omega_i, \omega_o)\ L_i(x,\omega_i)(\omega_i \cdot n)$ — *cos Θ*

Exkurs: BRDF



$$f(\omega_i, \omega_o) = \frac{L(\omega_o)}{E(\omega_i)} =$$

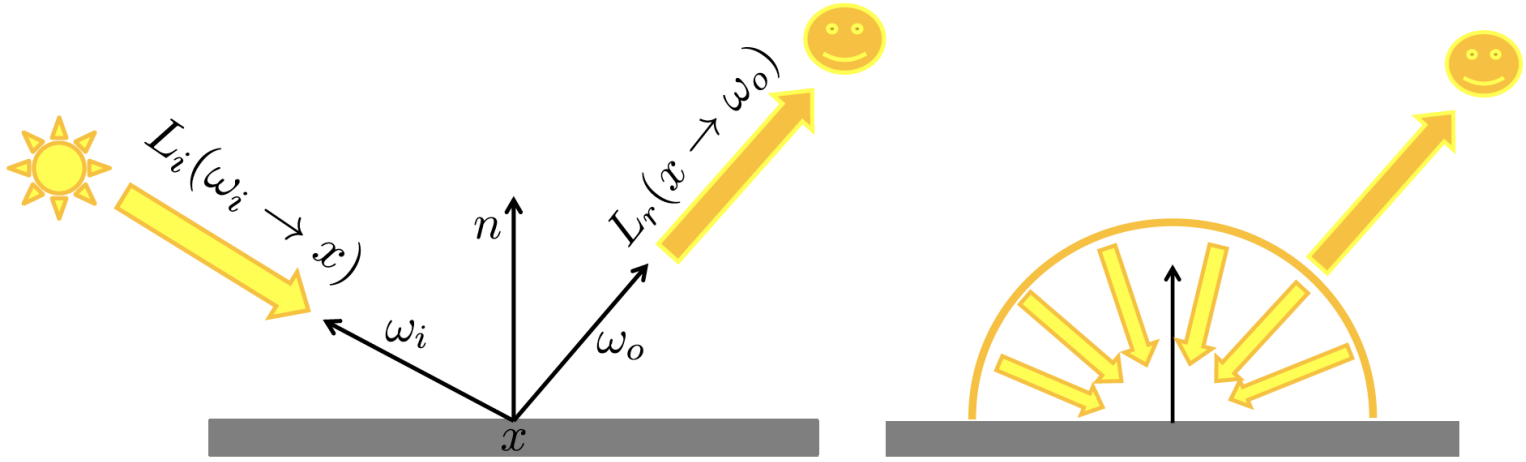
$$\frac{\text{ausgehende Strahldichte } \left[\text{W}/(\text{m}^2\text{sr})\right]}{\text{eingehende Bestrahlungstärke } \left[\text{W}/\text{m}^2\right]}$$

ω ist sog. Raumwinkel

- BRDF: Bidirectional Reflectance Distribution Function
- Beschreibt Material, d.h. wie Licht von Oberfläche reflektiert wird
  - Abhängig von gezeigten (Raum-) Winkeln (u. Wellenlänge)
  - Beschreibt Materialeigenschaften einer Objekt-Oberfläche
  - Gibt Wahrscheinlichkeit an, mit welcher eingehendes Photon in eine Richtung emittiert wird
- Exkurs: Eigenschaften
  - $f(\omega_i, \omega_o) \geq 0$
  - Erfüllt Reziprozität: $f(\omega_i, \omega_o) = f(\omega_o, \omega_i)$
    - Vertauscht man Einfalls- und Ausfallrichtung des Lichts, ändert sich am Wert der BRDF nichts
  - Energieerhaltung: $\int_\Omega f(\omega_i, \omega_o)(n \cdot \omega_o) d\omega_o \leq 1$

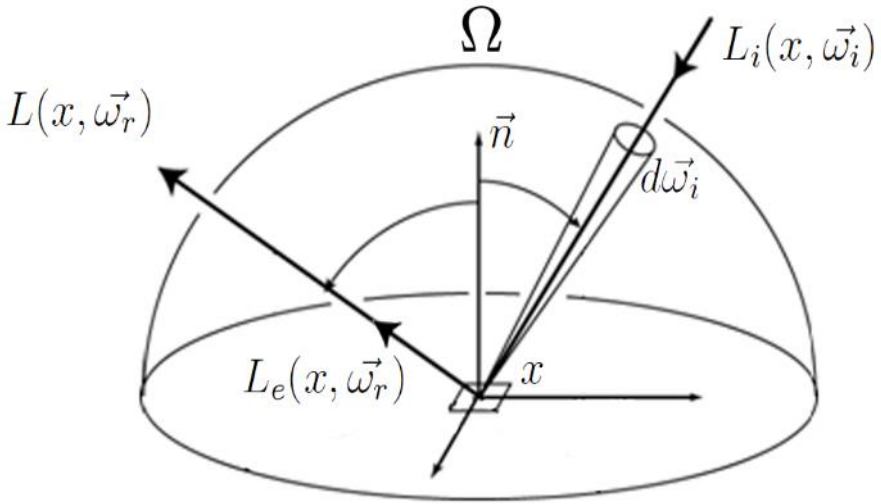
Lichttransport und Reflexion

- Gesucht: in Kamera einfallendes Licht
  - Formal: an Punkt *x* reflektierte Strahldichte (engl. radiance) L_r in Richtung ω_o



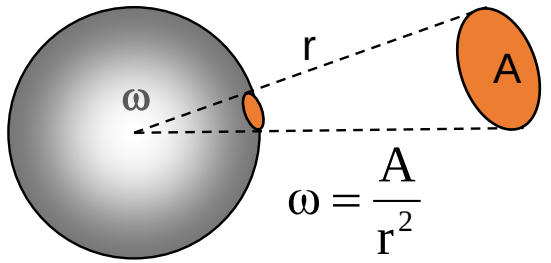
  - Problematisch: in Realität kommt Licht aus unendlich vielen Richtungen
    - Und kann theoretisch sogar innerhalb von Objekt streuen (sog. Subsurface Scattering)

Die Rendering-Gleichung



Ω ist obere Hemisphäre über *x*

Raumwinkel



BRDF

$$L(x,\vec{\omega_r}) = L_e(x,\vec{\omega_r}) + \int_{\Omega(\vec{n})} f(x,\vec{\omega_i},\vec{\omega_r}) L_i(x,\vec{\omega_i})(\vec{\omega_i}\cdot\vec{n}) d\vec{\omega_i}$$

Vereinfachung:

$$L(x,\vec{\omega_r}) = L_e(x,\vec{\omega_r}) + \sum_{j=1..k} f(x,\vec{\omega_j},\vec{\omega_r}) L_i(x,\vec{\omega_j})(\vec{\omega_j}\cdot\vec{n})$$

Weitere Vereinfachung: ☺

$$B = c \sum_{j=1..k} \max(0,\vec{\omega_j}\cdot\vec{n}) \cdot I_j$$

(diffuse Reflexion mit Farbe *c* für *k* Lichter mit Intensität *I*)

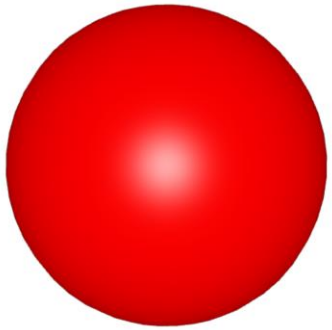
Phong‘sches Beleuchtungsmodell

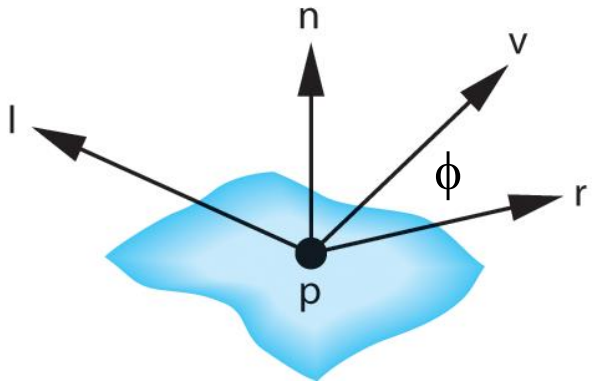
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

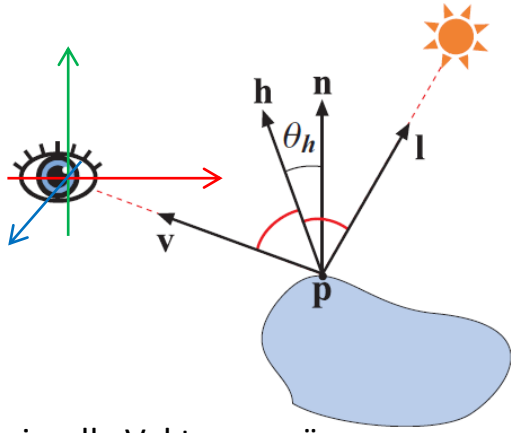
- Betrachte Interaktion zwischen Licht und Materie
  - Je nach Material zu schwierig für physikalisch korrekte, geschlossene Lösung
  - Für Echtzeitgraphik daher (früher) Aufteilung der Reflexionseigenschaften in einfacher zu behandelnde Komponenten (entwickelt 1975)
    - Beleuchtungsmodell war Teil der OpenGL Fixed Function Pipeline
- Aufteilung durch Addition verschiedener Beiträge
  - Ambienter Anteil + diffuse Reflexion + spiegelnde Reflexion (+ Emission)
    - Nutzt Superposition aus: bei Materialeigenschaften, die unabhängig von einander beschrieben werden können, addieren sich deren Effekte
    - Phong-Modell trotzdem physikalisch nicht korrekt: nicht reziprok, kein Energieerhalt

Phong‘sches Beleuchtungsmodell

- Aufteilung der Reflexionseigenschaften in ambienten, diffusen und spekularen Teil
  - In *Qt* einfach mittels *QPhongMaterial*
- Nutzt vier Vektoren
  - ***n*** : Flächennormale
  - ***l*** : Richtung zur Lichtquelle
  - ***v*** : Richtung zur Kamera
  - ***r*** : Reflexionsvektor
- Blinn-Phong‘sches Modell
  - ***h*** : Nutzt statt ***r*** Halbvektor $\vec{h} = \frac{\vec{l} + \vec{v}}{|\vec{l} + \vec{v}|}$



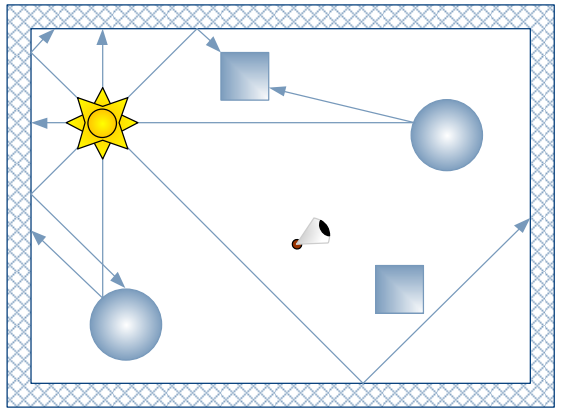




Hinweis: alle Vektoren müssen normalisiert sein (Länge ist 1)

Komponenten

Ambientes Licht

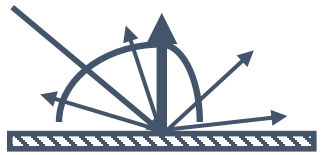


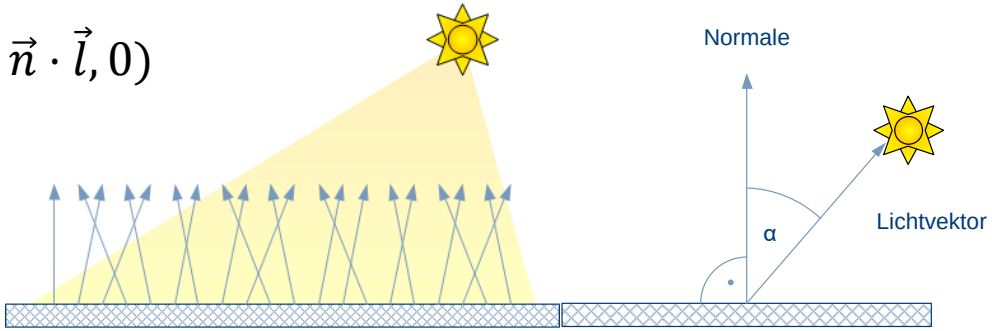
$k_{amb} \otimes I_{amb}$

Diffuses Licht

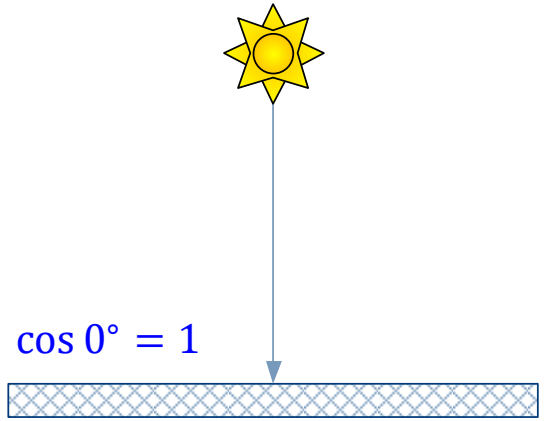
$k_{diff} \otimes I_{diff} \max(\vec{n} \cdot \vec{l}, 0)$

$\vec{n} \cdot \vec{l} = \cos\alpha$





- Lichtintensität abhängig von Einfallswinkel
- Reflexion in alle Richtungen gleichstark
- Unabhängig von Betrachterposition



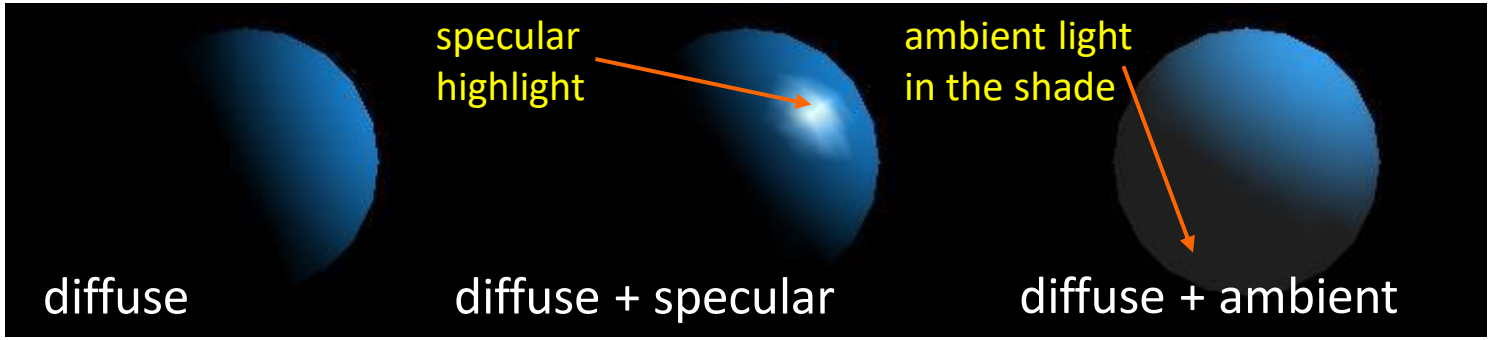
Maximale Intensität

Keine Intensität

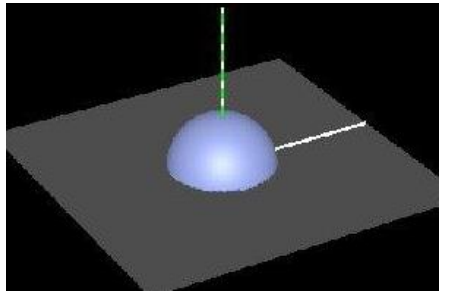
$\otimes = komponentenweise\ Multiplikation$

Ambiente Beleuchtung

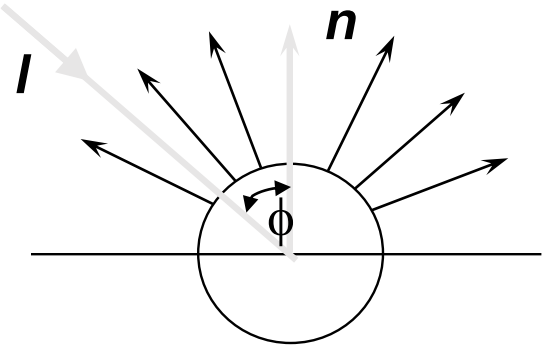
- Ambienter Term simuliert indirekte Beleuchtung
  - Letztere zu teuer für Echtzeitanwendungen
    - Licht durch Umgebung mehrfach reflektiert, ursprüngliche Richtung nicht mehr identifizierbar, Intensität unabhängig von Betrachterposition
  - Berücksichtigt Beiträge von Reflektionen anderer Flächen, die Grundhelligkeit erzeugen
- Billiger Fake: meist mit einfachem additivem globalen Term berücksichtigt
  - Ambientes Licht in Szene: $\mathbf{i}_{amb} = \mathbf{m}_{amb} \otimes \mathbf{I}_{amb}$
  - Genau gleicher Farbwert an jeder Stelle eines 3D-Objekts, erhält so nicht geometrische Struktur

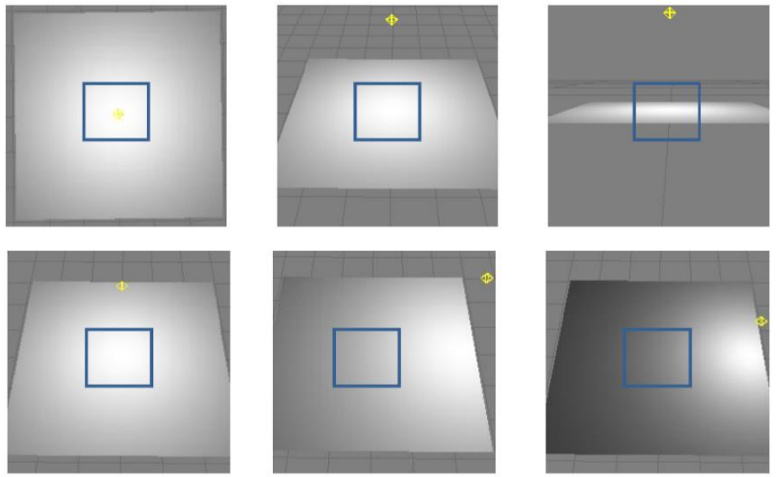


Diffuse Reflexion



- Licht streut gleichmäßig in alle Richtungen
  - Menge des reflektierten Lichts hängt nur ab vom Winkel zwischen Flächennormale und Richtung zur Lichtquelle
    - Lambertsches Gesetz: In eine Richtung reflektierte Energie ist proportional zum Kosinus des Winkels zwischen dieser Richtung und der Normalen
    - BRDF (d.h. „Farbe“) konstant: $f(\omega_i, \omega_o) = k_d = \text{const}$
- Kamera bewegt sich
  - Helligkeit konstant
  - Unabhängig von Blickwinkel
- Licht bewegt sich
  - Helligkeit ändert sich
  - Abhängig von Lichtrichtung





$$d = k_d \otimes I_d \max(\vec{n} \cdot \vec{l}, 0)$$

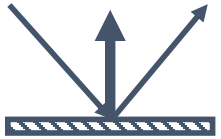
reflected intensity

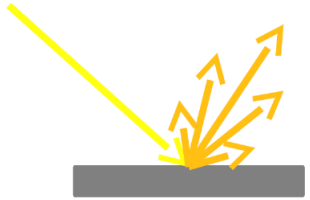
diffuse surface reflection coeff.

incoming light intensity

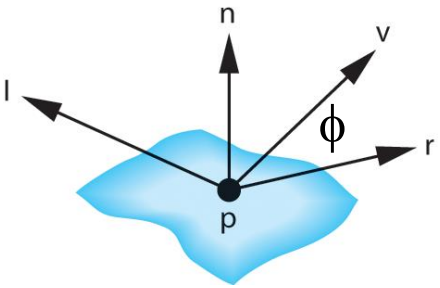
Maximum, da negative Werte, falls backfacing

Spiegelnde Reflexion





- Bündelung des Lichts in eine Richtung (“Einfallswinkel = Ausfallswinkel”)
  - Oberflächen reflektieren meist weder ideal diffus noch spiegelnd, sondern glossy
  - Licht streut aber primär um Reflexionsrichtung herum
    - Führt eigentlich zu verschwommener Spiegelung der Umgebung
    - Oft zu teuer, daher zumindest verschwommene Spiegelung der Lichtquelle
- Modellierung durch Term, der abhängig ist vom Winkel zwischen Betrachter und idealer Spiegelung
  - Anteil des spekularen Lichts abhängig von Position des Betrachters
    - Blick- und Reflexionsvektor gleich: spekulares Highlight am größten
  - Phong-Modell nutzt View- und Reflection-Vektor:
    - Mit $\vec{r} = 2\left(\vec{n} \bullet \vec{l}\right)\vec{n} - \vec{l}$



$$s = k_s \otimes I_s \max(\vec{v} \cdot \vec{r}, 0)^{shi}$$

shininess coefficient

reflected intensity

specular surface reflection coeff.

incoming light intensity

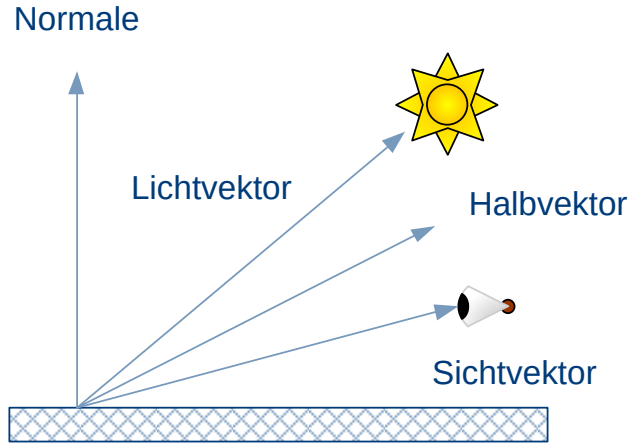
Spiegelnde Reflexion

- Schnellere Berechnung nach Blinn-Phong-Modell
  - Nutzt Normale u. Halbvektor:
    - Phong hat rundes Highlight, Blinn-Phong hat längliches (→ wirkt realistischer)

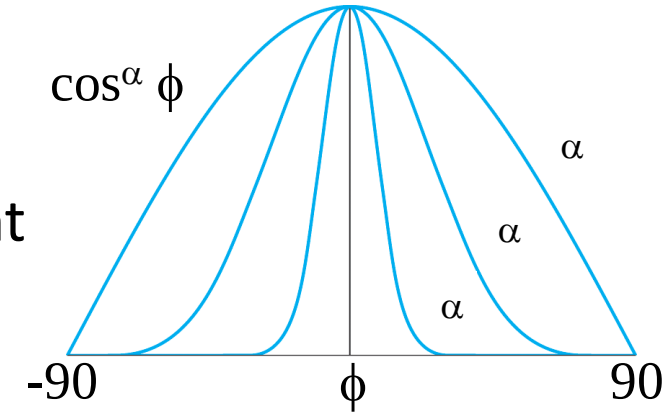
$$s' = k_s \otimes I_s \max(\vec{n} \cdot \vec{h}, 0)^{shi'}$$

$$mit \quad \vec{h} = \frac{\vec{l} + \vec{v}}{|\vec{l} + \vec{v}|}$$

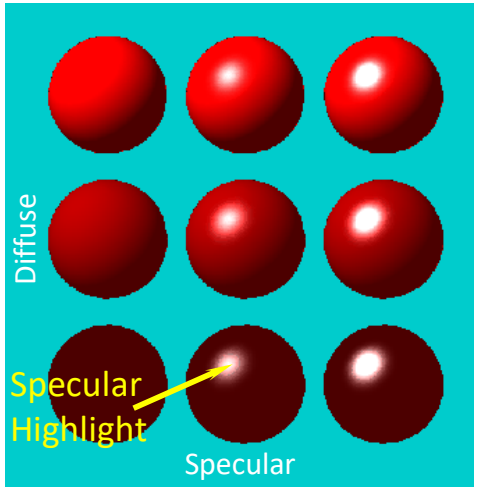
Maximum, da negative Werte, falls backfacing



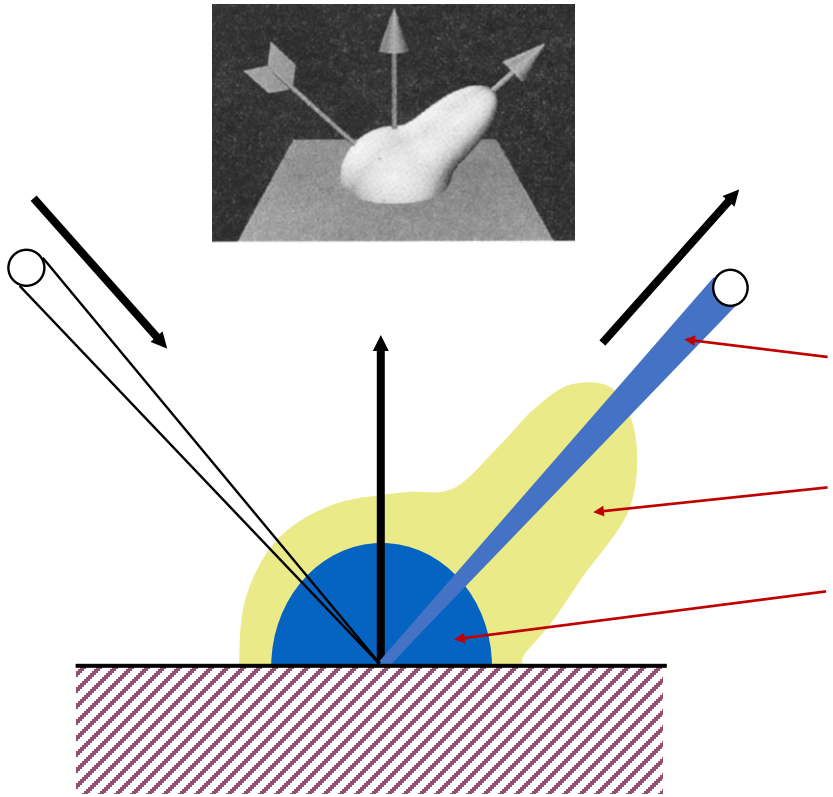
- Hoher Spekularanteil bei Metall- und Plastikoberflächen
  - Spekularer Exponent beschreibt „Schärfe“ des Effekts
    - 0 ≤ *shi* (bzw. α) ≤ 128 (manchmal normiert auf [0, 1])
  - Größere Werte bei „Shininess“ bedeuten kleineres Highlight
    - Hohe Werte (über 100) korrespondieren mit Metall
    - Kleinere Werte ergeben plastikartiges Aussehen



Darstellung der Komponenten



- Summe aus ambienten, diffusen und spekularen Anteilen (hier für verschiedene Einfallswinkel ϕ)



Phong	$\rho_{ambient}$	$\rho_{diffuse}$	$\rho_{specular}$	ρ_{total}
$\phi_i = 60^\circ$				
$\phi_i = 25^\circ$				
$\phi_i = 0^\circ$				

Phong Beleuchtungsgleichung

- Für jeden Punkt der Objektoberfläche kann nun Beleuchtungsgleichung aufgestellt werden (alle Vektoren normalisiert)
  - Jeder Farbanteil wird dabei separat berechnet

$$I_{ges} = k_a \otimes I_a + k_d \otimes I_d \, max(\boldsymbol{l} \cdot \boldsymbol{n}, 0) + k_s \otimes I_s \, max(\boldsymbol{v} \cdot \boldsymbol{r}, 0)^{\alpha}$$

Ambient Diffus Spekular

- I_{ges} muss für jede Lichtquelle bestimmt werden
  - Finale Farbe durch anschließendes Summieren der Teilergebnisse I_{ges} pro Lichtquelle
- Spekularer Anteil bei Blinn-Phong: $k_s \otimes I_s \, max(\boldsymbol{n} \cdot \boldsymbol{h}, 0)^{\alpha'}$

Entfernungsabhängige Dämpfung

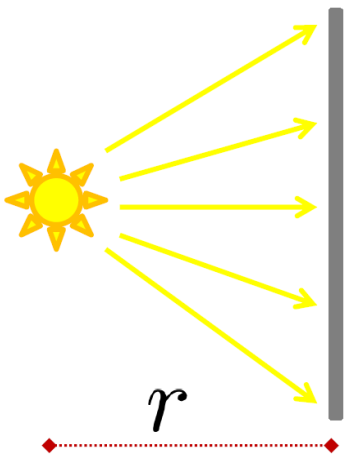
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

- Wieviel Licht kommt auf Oberfläche noch an?
  - Nur für Lichtquellen mit Position (Point-/Spotlight)
  - Basisgleichung: $I_{ges} = I_{amb} + I_{diff} + I_{spec}$
  - Entfernungsabhängige Dämpfung *d*:

$$d = \frac{1}{s_c + s_l \cdot r + s_q \cdot r^2}$$

$$I_{ges} = I_{amb} + d(I_{diff} + I_{spec})$$

Mit ***r*** = Entfernung Licht zu betrachtetem Punkt



$$I_{ges} = a_{glob} \otimes m_{amb} + m_{em} + \sum_k c_{spot}^k \left(I_{amb}^k + d^k (I_{diff}^k + I_{spec}^k)\right)$$

  - Realistisch: $s_c = s_l = 0$, $s_q = 1$ (damit $d = {}^1/_{r^2}$)
    - Physikalisch zwar falsch, aber mehr Kontrolle durch s_c und s_l (abschalten: $s_c = 1$, $s_l = s_q = 0$)
    - Damit kommt also weiterer Licht-Parameter hinzu, nämlich Attenuation (hält s_c, s_l und s_q)

Mehrere Lichtquellen

- Blinn-Phong Basisgleichung

$$I = k_{amb} \otimes L_{amb} + k_{diff} \otimes L_{diff} \max(\vec{n} \cdot \vec{l}, 0) + k_{spec} \otimes L_{spec} \max(\vec{n} \cdot \vec{h}, 0)^{shi}$$

- Gesamtgleichung (vgl. OpenGL) inklusive:
  - Emissivem Materialbeitrag m_{em}
  - Globalem ambientem Licht a_{glob}

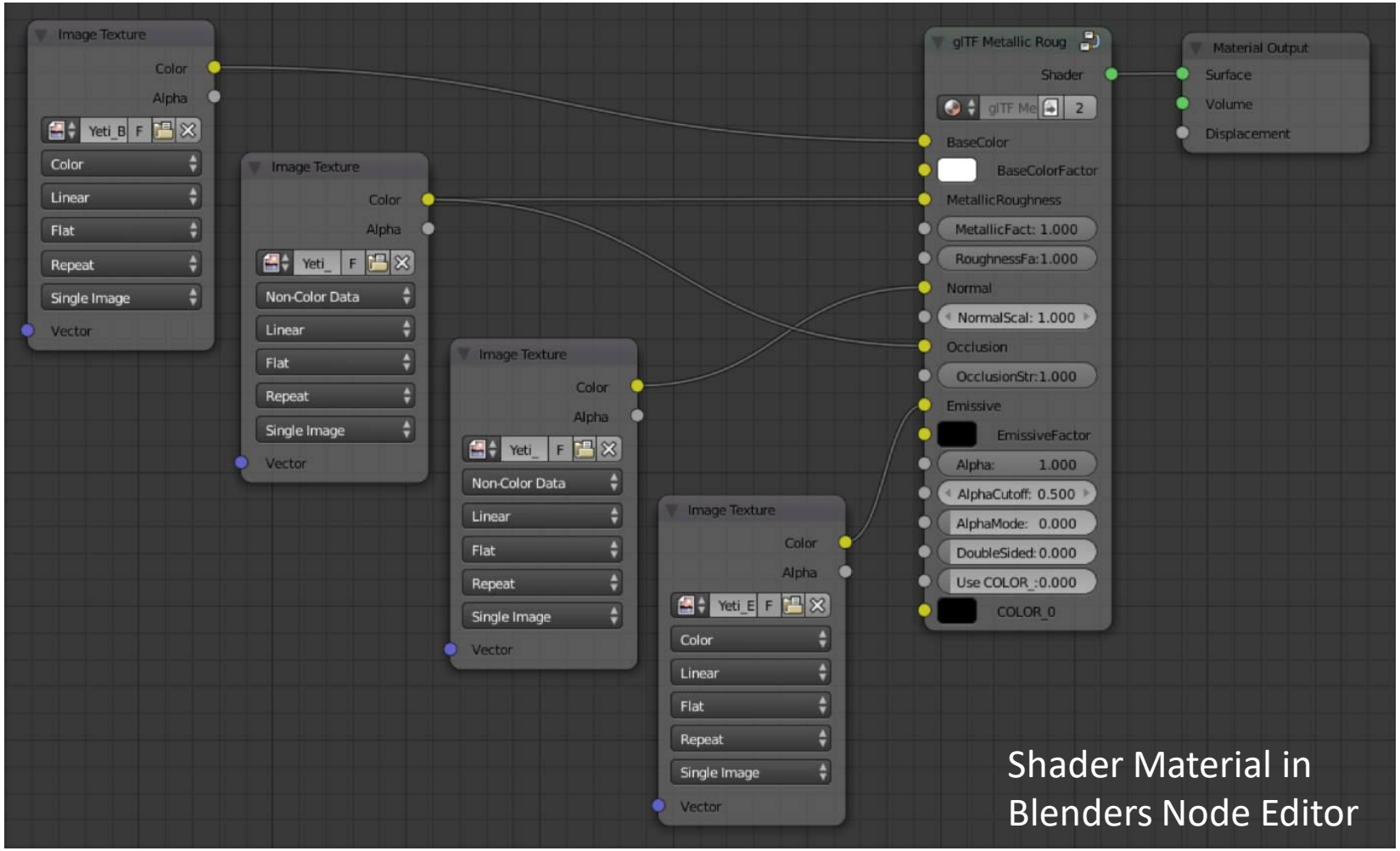
$\otimes =$ *komponentenweise Multiplikation*

$$I_{ges} = m_{em} + a_{glob} \otimes m_{amb} + \sum_k d^k c_{spot}^k \left(I_{amb}^k + I_{diff}^k + I_{spec}^k\right)$$

Spotlight–Beitrag: c_{spot}

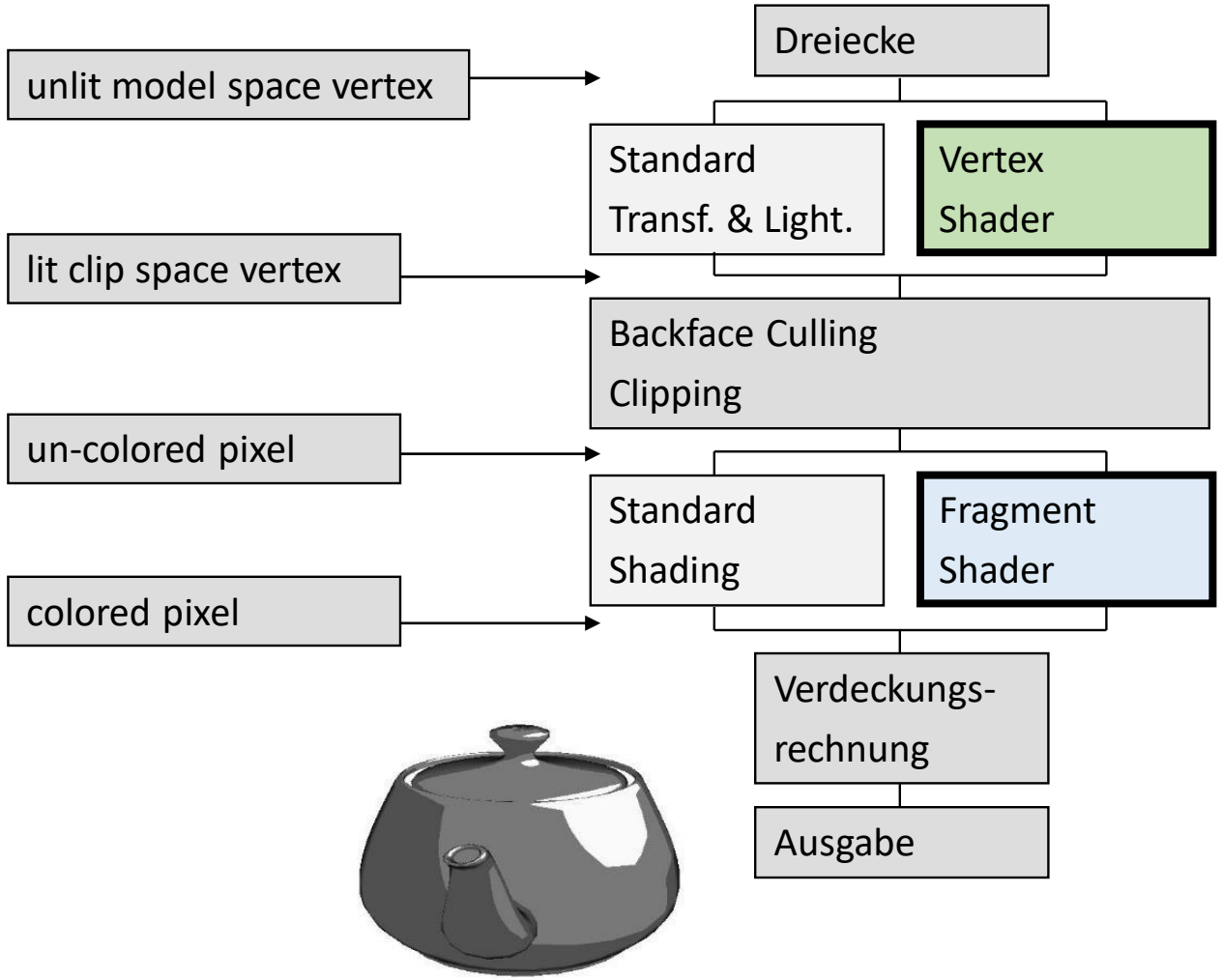
- Achtung: Summe kann größer 1 sein
  - An 1 'clampen' (Farbverschiebungen möglich)
  - Anteilig skalieren (unterschiedliche Wirkungen)

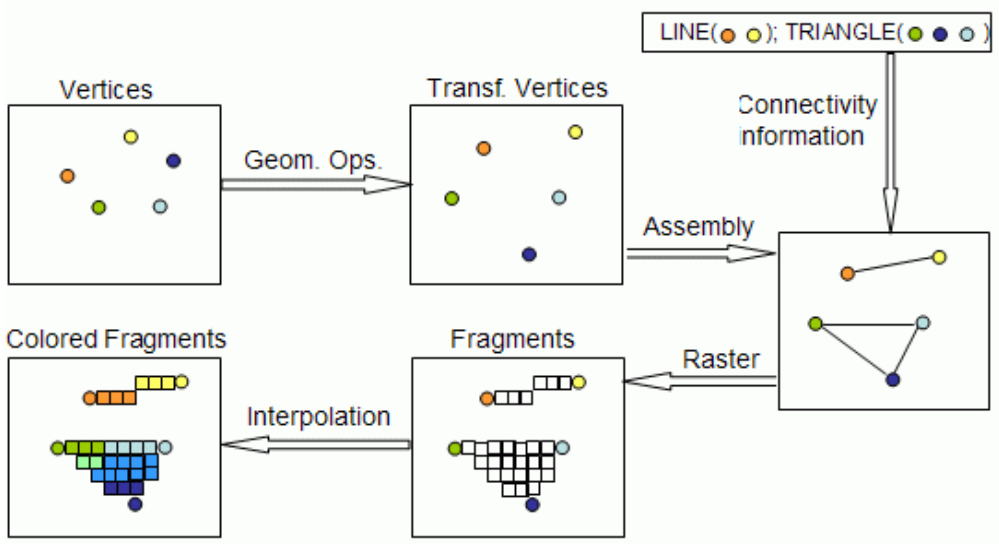
Materialien und Shader für Artists



Shader Material in Blenders Node Editor

Programmierbare GPUs





- Seit ca. einem Vierteljahrhundert möglich, 3D-Grafik-Pipeline über Shader zu erweitern
- Verwendbarkeit inzwischen recht universell
- Transformationen, Lighting usw. werden dabei selbst programmiert

Shader: die Anfänge – GeForce 2

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

```
!!VP1.0
DP4 o[HPOS].x, c[0], v[OPOS];
DP4 o[HPOS].y, c[1], v[OPOS];
DP4 o[HPOS].z, c[2], v[OPOS];
DP4 o[HPOS].w, c[3], v[OPOS];
DP3 R5.x, c[4], v[NRML];
DP3 R5.y, c[5], v[NRML];
DP3 R5.z, c[6], v[NRML];
DP4 R0.x, c[8], v[OPOS];
DP4 R0.y, c[9], v[OPOS];
DP4 R0.z, c[10], v[OPOS];
DP4 R0.w, c[11], v[OPOS];
ADD R0, -R0, c[20];
DP3 R8.w, R0, R0;
RSQ R8.w, R8.w;
MUL R8, R0, R8.w;
DP3 R0.x, R5, -R8;
MAD R1.x, -R0.x, R0.x, c[23].y;
MUL R1.x, R1.x, c[22].y;
```

```
ADD R1.x, c[23].y, -R1.x;
RSQ R2.x, R1.x;
RCP R2.x, R2.x;
MAD R2.x, c[22].x, R0.x, R2.x;
MUL R2, R5, R2.x;
MAD R2, c[22].x, -R8, R2;
MUL R0, R5, c[23].z;
DP3 R4.w, R5, R8;
MAD R3, R4.w, R0, -R8;
DP3 o[TEX0].x, c[12], R2;
DP3 o[TEX0].y, c[13], R2;
DP3 o[TEX0].z, c[14], R2;
DP3 o[TEX1].x, c[12], R3;
DP3 o[TEX1].y, c[13], R3;
DP3 o[TEX1].z, c[14], R3;
ADD R4.w, c[23].y,-R4.w;
MUL R4.w, R4.w, R4.w;
MUL o[COL0],R4.w, c[21];
END
```

GeForce 2 erschien Mitte 2000

```
GLint surfaceCombiner[2] = glGenLists(1);
glNewList( surfaceCombiner[2], GL_COMPILE );
  nvparse(
    "!!RC1.0                          \n"
    "const0 = ( 1.0, 0.6, 0.3, 1.0 ); \n"
    "{                                \n"
    "  rgb                            \n"
    "  {                              \n"
    "    discard = const0*unsigned_invert(col0);
  n"
    "    spare0 = tex1*col0;          \n"
    "    spare1 = sum();              \n"
    "  }                              \n"
    "}                                \n"
    "out.rgb = spare1;                \n"
    );
glEndList();
```

- Erstmals programmierbarer Vertex- u. (eingeschränkt) Fragment-Prozessor
- Nachteile: Low-Level-Assembler schwer lesbar, schlecht wartbar, wenig intuitiv

Shader-Hochsprachen

- Entwicklung von High-Level-Shader-Sprachen
  - HLSL und GLSL (OpenGL Shading Language)
- GLSL, inkl. dazugehörige API zur Verwaltung von Shadern, gehört seit OpenGL 2.0 zu Kern-Features von OpenGL
  - C-artige Sprache zur Programmierung des Vertex- und Fragment-Prozessors (später auch für Geometry- und Tesselation-Shader)
- Eingebaute Funktionen für Skalar- u. Vektoroperationen
  - Trigonometrische Funktionen wie sin, cos, tan
  - Exponentialfunktionen wie pow, exp, sqrt
  - Allgemeine Fktn. wie sign, fract, min, max, step
  - Geometrische Fktn. wie length, dot, cross, reflect
  - Texture-Lookup: z.B. texture[1,2,3]D, textureCube
  - …

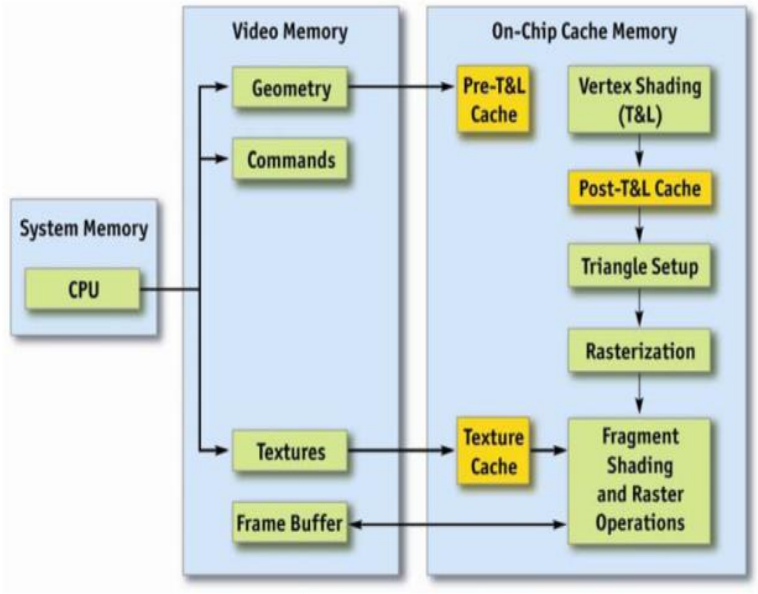
Vertex Shader

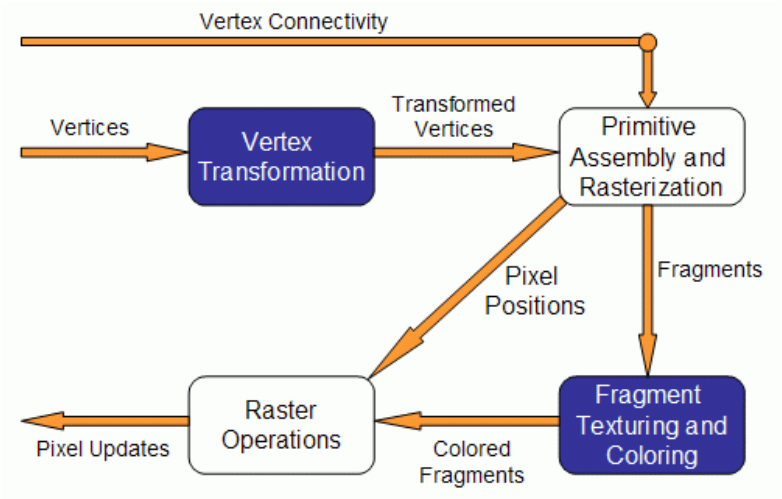
```
attribute vec3 position;
attribute vec3 color;
uniform mat4 transMat;
varying vec3 vColor;
void main() {
    vColor = color;
    gl_Position = transMat * vec4(position, 1.0);
}
```

Fragment Shader

```
varying vec3 vColor;
void main() {
    gl_FragColor = vec4(vColor, 1.0);
}
```

Rendering Pipeline (Shader Model 3.0)





- Input: Geometrie
  - Vertices in Objektkoordinaten
- Intern mehrere Stufen
  1. Transform & Lighting
  2. Primitive Assembly
  3. Rasterization
  4. Fragment Shading
  5. Raster Operations
- Output: Image
  - Pixel in Framebuffer / Textur
- Entspricht DirectX 9
  - ~ Funktionsumfang WebGL 1.0

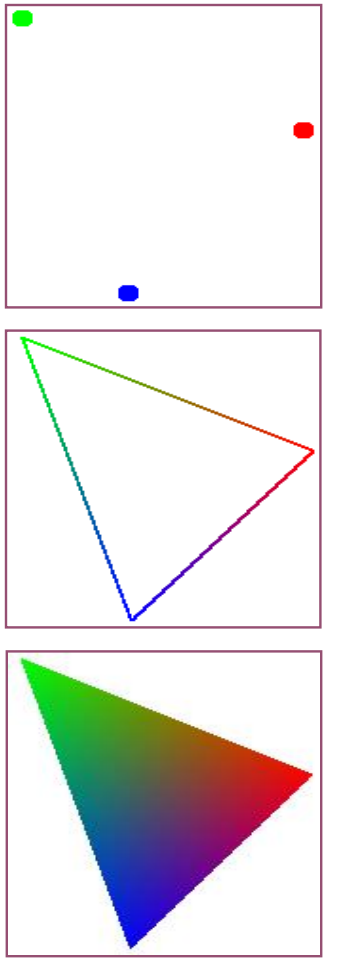
Neue Shadermodelle verbessern Programmierbarkeit der GPU:

- SM 4.0: Geometry Shader sowie Transform Feedback
- SM 5.0: Tessellation Control und Tesselation Evaluation Shader

Optimierungsansatz

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

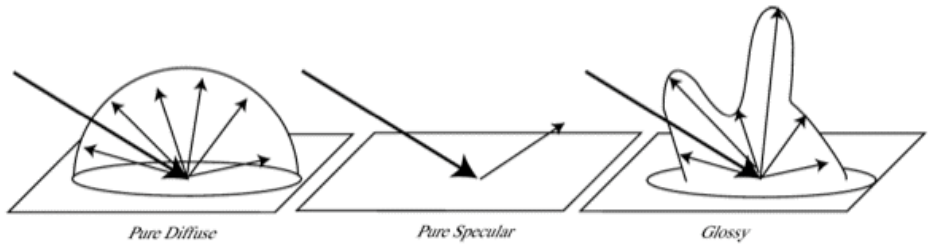
- 3D-Graphik-Pipeline beschreibt Weg vom 3D-Modell zum Bild
  - Unter Verwendung von Kamera, Lichtquellen, Materialbeschreibungen
- Dabei typisch für Detailgrad von synthetischen Bildern:
  - Anzahl Eckpunkte ≪ Anzahl zu errechnender Pixel
- Optimierungsidee für leistungsschwache Hardware
  - Beleuchtungsrechnung nicht für jeden (potentiellen) Pixel lösen, sondern nur für jeden Vertex (Eckpunkt)
    - Dazwischen Farbe für jeden Pixel linear interpolieren
  - Durch Interpolation wird Bild aufwandsreduziert erzeugt
    - → Gouraud Shading (bedeutet per-Vertex Lighting)



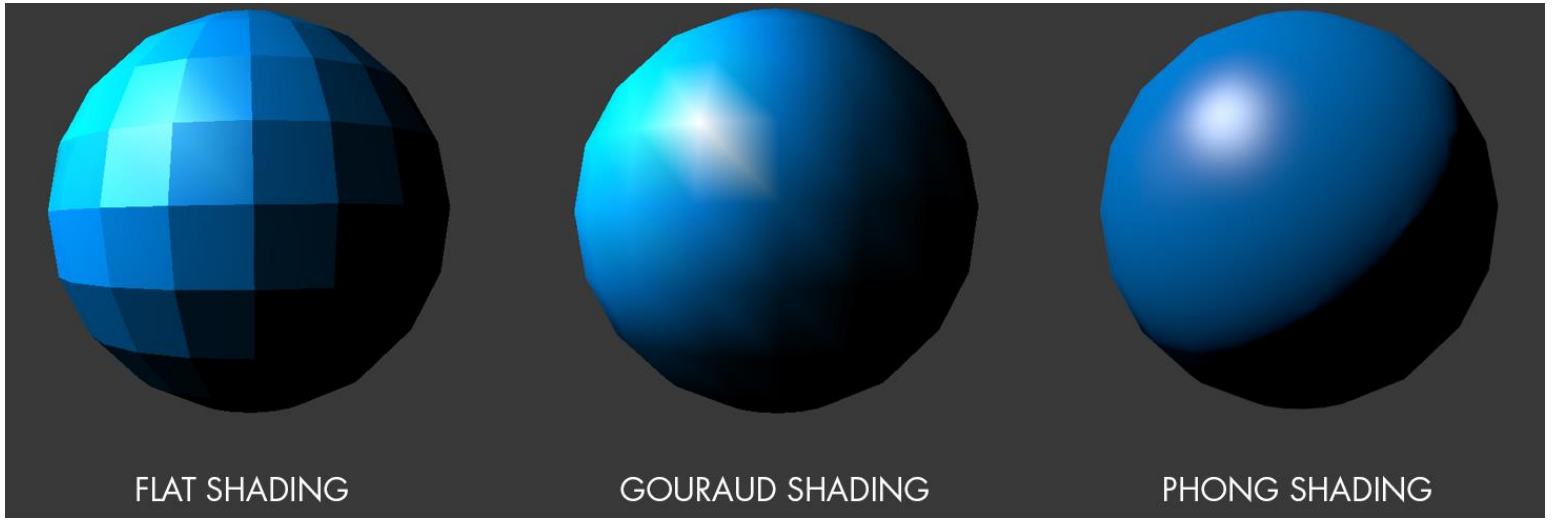
Shading vs. Lighting

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

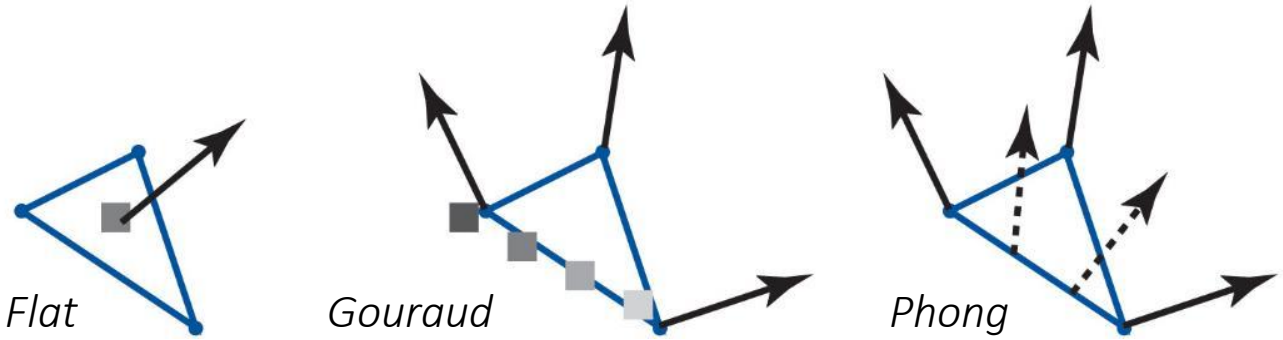
- Pixel müssen (beim Rasterisieren) eingefärbt werden (→ Shading)
  - Ansätze beim Shading:
    Zuweisung von Farbe pro Fläche, pro Vertex oder pro Pixel
- Lösung des Problems
  - Für plausibles Aussehen: Simulation, wie Fläche auf Licht reagiert (→ Lighting)
  - Lighting bedeutet Auswertung eines Beleuchtungsmodells an Sample-Position
    - Kann physikalisch-basiert sein (PBR), phänomenologisch begründet (z.B. Blinn-Phong) oder künstlerisch (d.h. non-photorealistic)
    - Anmerkung: Rendern bedeutet das Visualisieren einer beleuchteten Szene



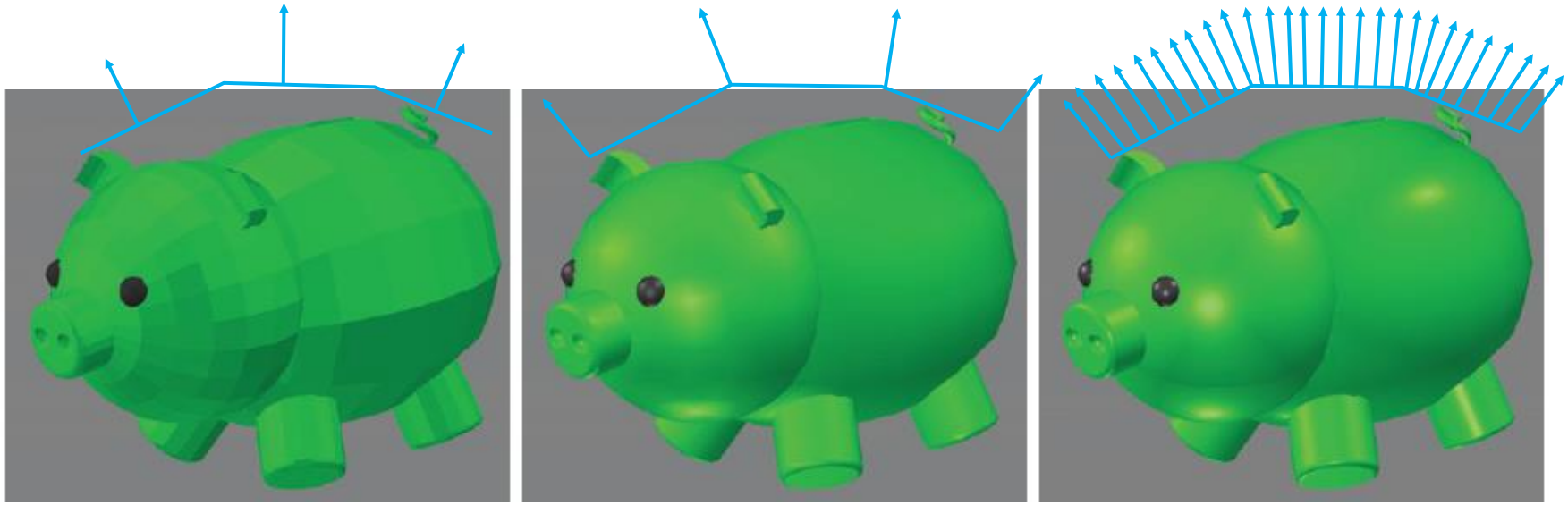
Shadingverfahren



- Funktionsweise der drei Shadingverfahren im Vergleich
  - Heutiger Standard für aktuelle GPUs: Phong-Shading
  - Flat-Shading von low-level APIs nicht mehr unterstützt



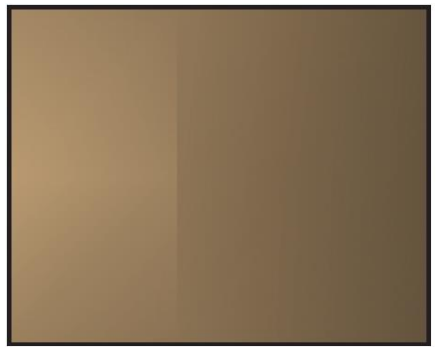
Flat-, Gouraud- & Phong-Shading

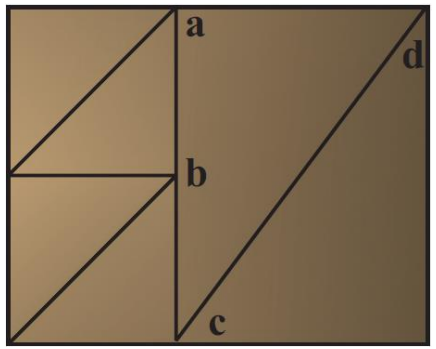


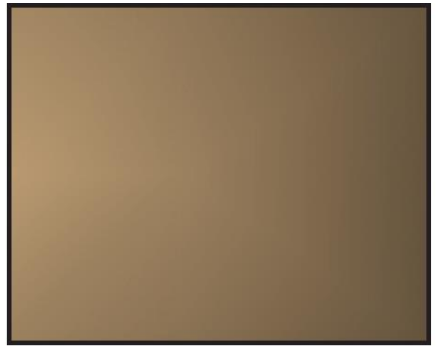
- Schwein mit verschiedenem Shading, aber je gleichem Beleuchtungsmodell gerendert
  - Links: Flat Shading (Beleuchtungsrechnung pro Face, mit Face Normals)
  - Mitte: Gouraud Shading (Beleuchtungsrechnung pro Vertex, Ergebnis interpoliert)
  - Rechts: Phong Shading (Normale interpoliert, Beleuchtungsrechnung pro Pixel)
- Blinn-Phong Lighting (hier für eine Lichtquelle und ohne Dämpfung):
  - $I = k_{amb} \otimes L_{amb} + k_{diff} \otimes L_{diff} \max(\vec{n} \cdot \vec{l}, 0) + k_{spec} \otimes L_{spec} \max(\vec{n} \cdot \vec{h}, 0)^{shi}$ mit $\vec{h} = \frac{\vec{l}+\vec{v}}{|\vec{l}+\vec{v}|}$

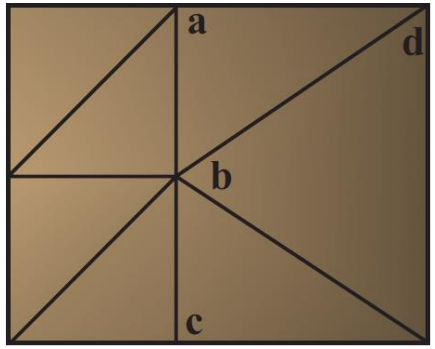
color := ambient + diffuse + specular

Probleme bei Gouraud-Shading



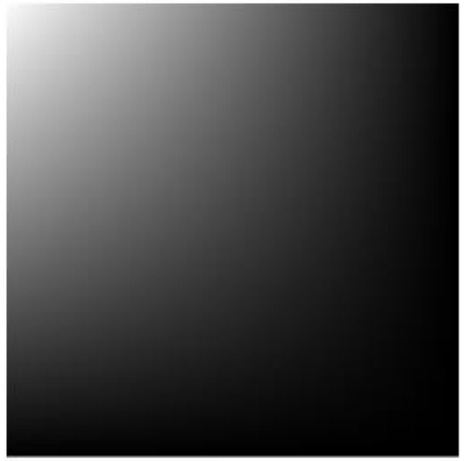


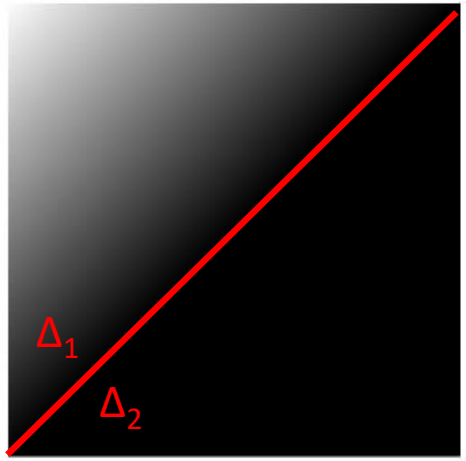


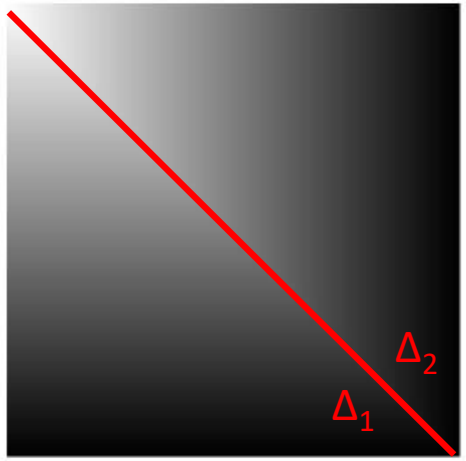


- Unterschiede möglichst minimieren, z.B. durch kürzeste Strecke oder weniger Farbunterschiede

Highlights nur sichtbar, wenn sie an Eckpunkt-Positionen auftreten, da im Inneren einer Fläche hier keine Beleuchtung ausgerechnet wird!



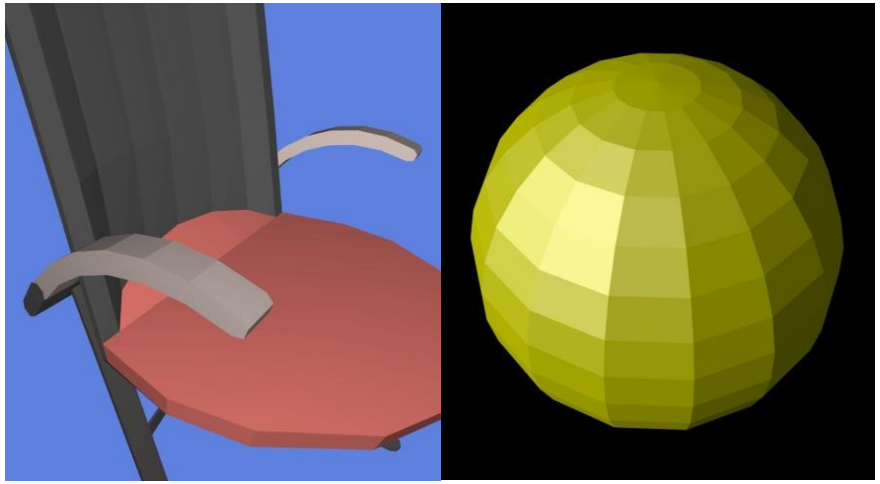


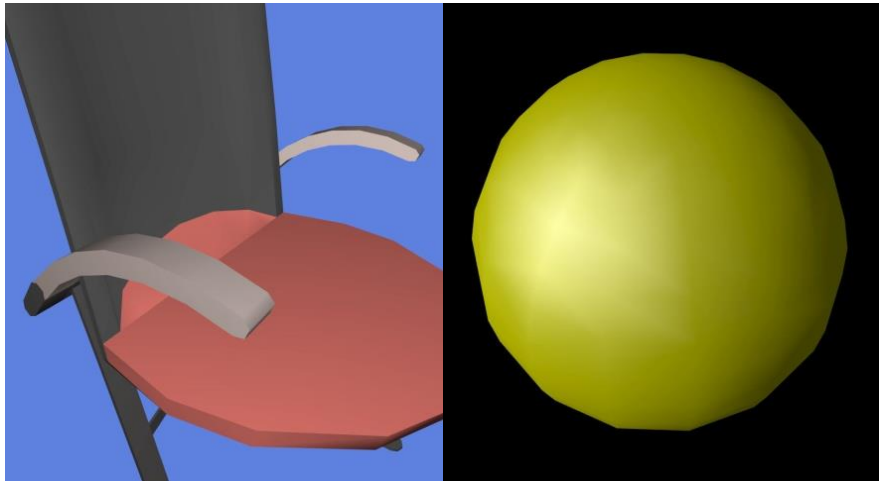


Übungsaufgabe 1

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

- Geben Sie eine Beleuchtungsgleichung an und erläutern Sie diese!
- Beschreiben Sie die Vorgehensweise bei den Ihnen bekannten Shading-Verfahren
- Welche Shading-Verfahren wurden bei diesen Bildern je verwendet?

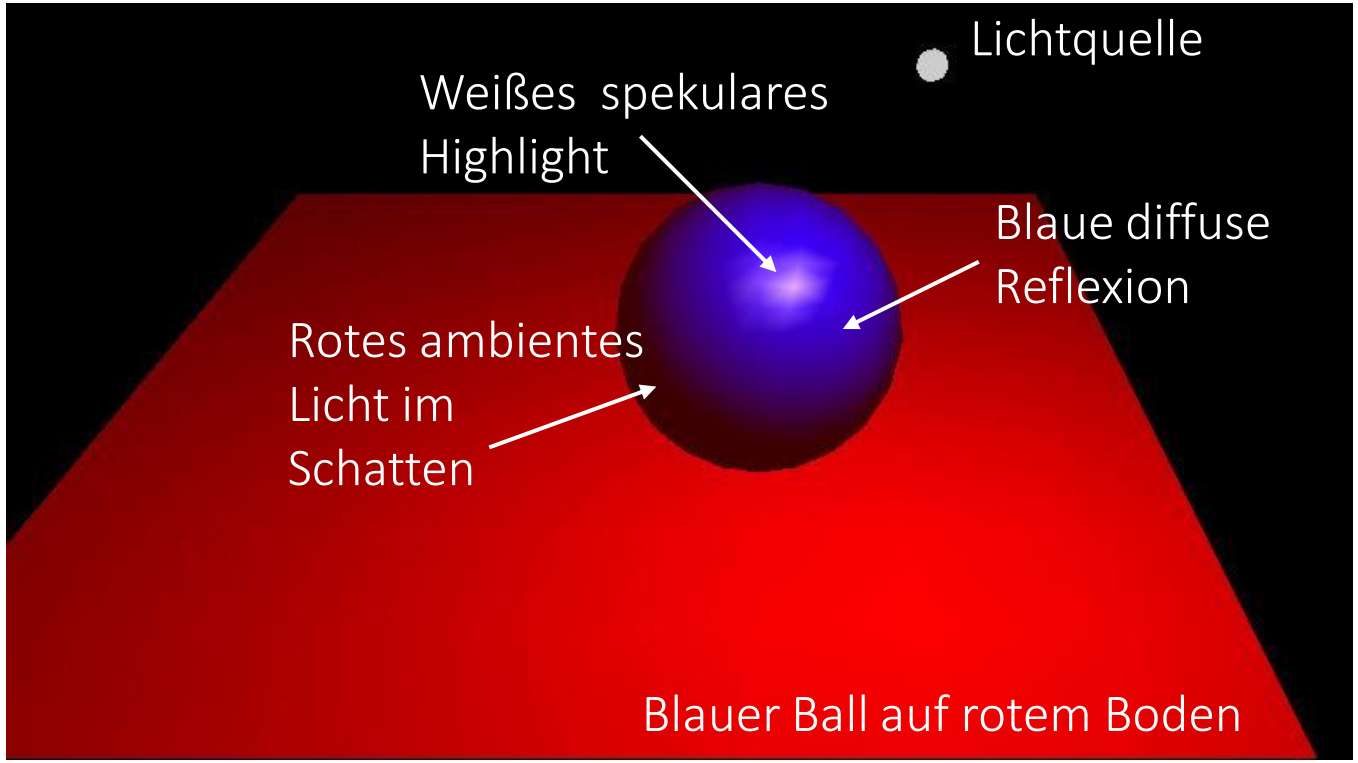




Übungsaufgabe 2

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

- Geben Sie für Lichtquelle und Kugel grob die Beleuchtungsparameter an!



Übungsaufgabe 3

- Gegeben ist ein Dreieck mit den Eckpunkten a=(1,0,0)^T, b=(0,1,0)^T, c=(0,0,0)^T
  - Berechnen Sie die Flächennormale $\vec{n}$ dieses Dreiecks

n = (0, 0, 1)^T

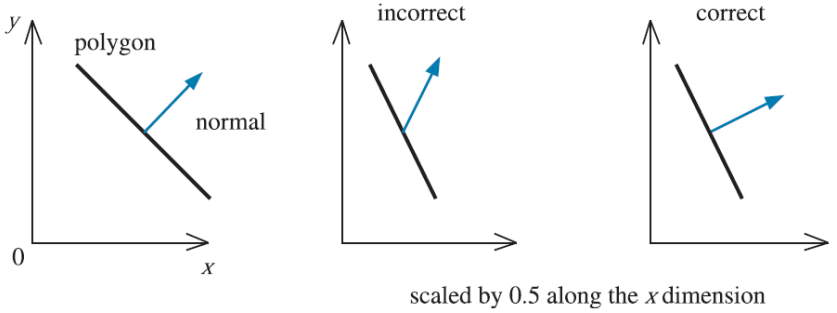
- Gegeben sei weiterhin Blickrichtung $\vec{v} = \begin{pmatrix} 0.8 \\ 0.0 \\ 0.6 \end{pmatrix}$
  - Berechnen Sie entsprechend dem Blinn-Phong-Modell die reflektierte Farbe c bei weißem Licht, Shininess $shi = 10$, diffuser Farbe $k_d = \begin{pmatrix} 0.3 \\ 0.6 \\ 0.9 \end{pmatrix}$ und spekularer Farbe $k_s = \begin{pmatrix} 0.4 \\ 0.4 \\ 0.4 \end{pmatrix}$
    - Berechnung für ein Headlight und ohne Dämpfung
    - Zur Erinnerung: bei Headlight gilt $\vec{l}$ = $\vec{v}$ und damit $\vec{h}$ = $\vec{v}$

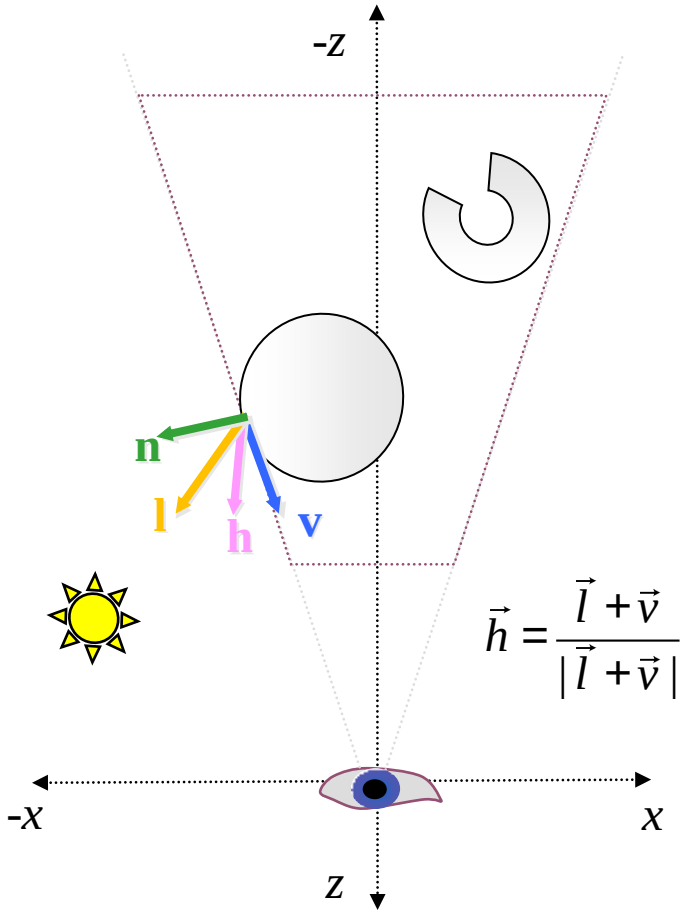
c = (0.1824, 0.3624, 0.5424)

Beleuchtungsrechung oft im Eye Space

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

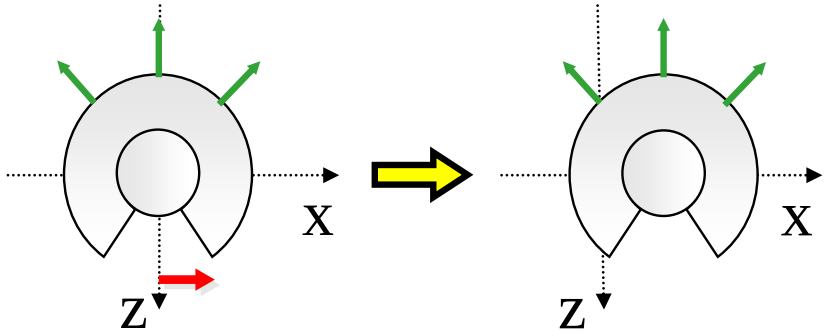
- Eckpunkte und andere Vektoren werden dazu in Kamerakoordinaten transformiert
  - Multiplikation je mit ModelView-Matrix M
- Achtung:
  - Normalen mit invers transponierter Matrix $(M^{-1})^T$ transformieren
    - Hier ohne Beweis…



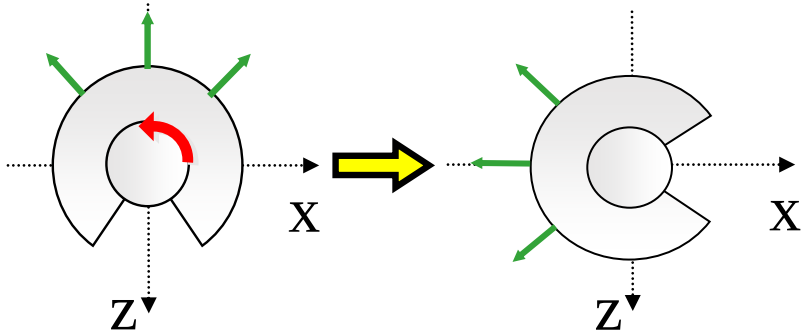


Transformation von Normalen

- Translation ändert Normalenvektoren nicht
  - Normale ist kein Punkt, sondern Vektor (→ $w = 0$)

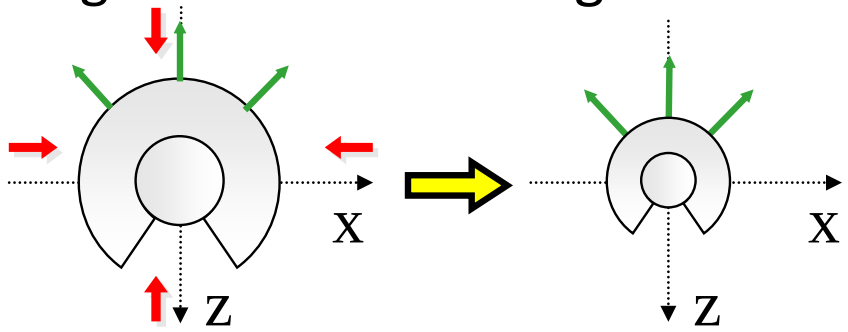


- Rotation ändert Normaleneigenschaft nicht
  - Vektoren sind rotiert, stehen aber noch senkrecht

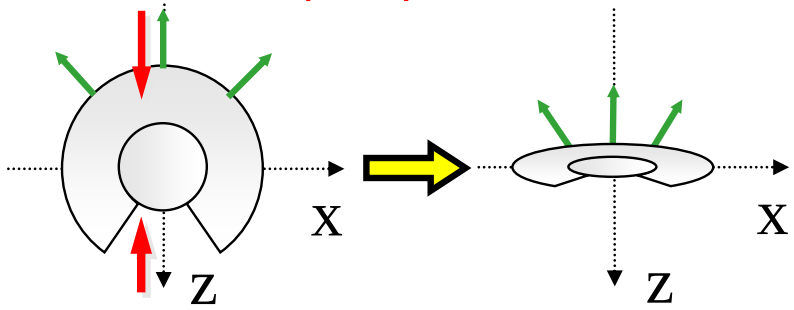


Transformation von Normalen

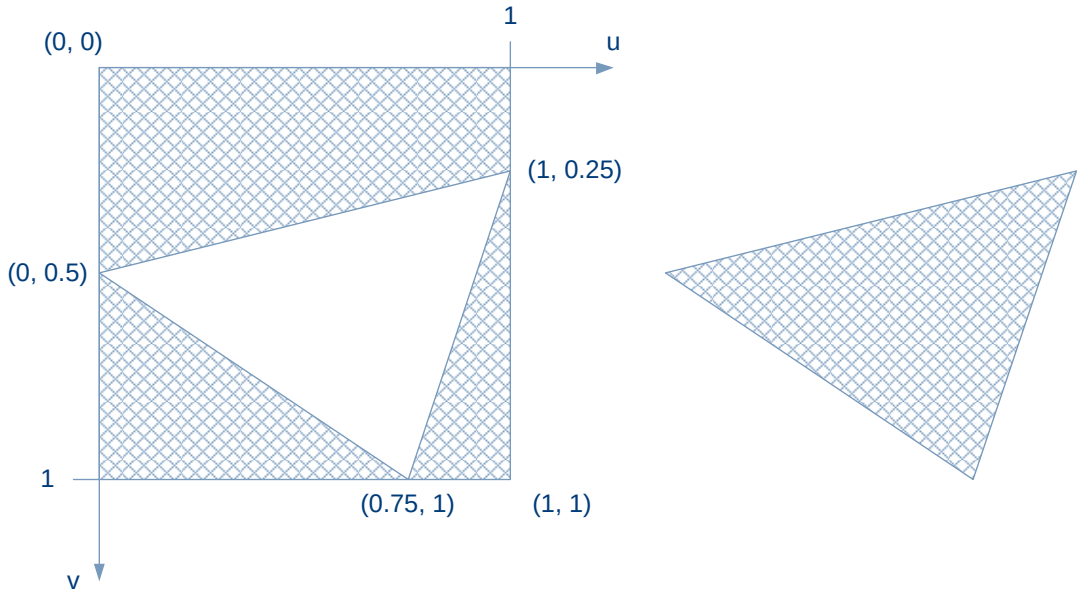
- Uniforme Streckung ändert nur Länge der Normalen
  - Nicht Richtung → nach Skalierung noch normalisieren!



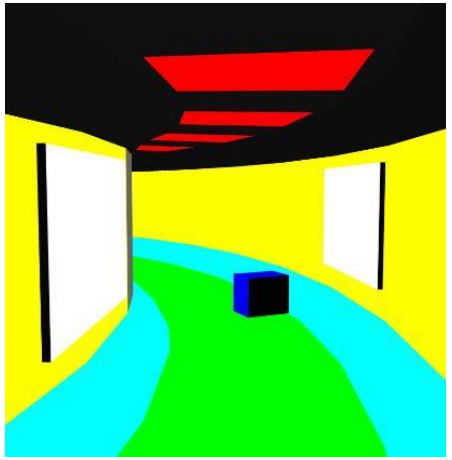
- Nicht-uniforme Streckung ändert Normalenrichtung
  - Bei Multiplikation mit $(M^{-1})^T$ bleibt Normale senkrecht

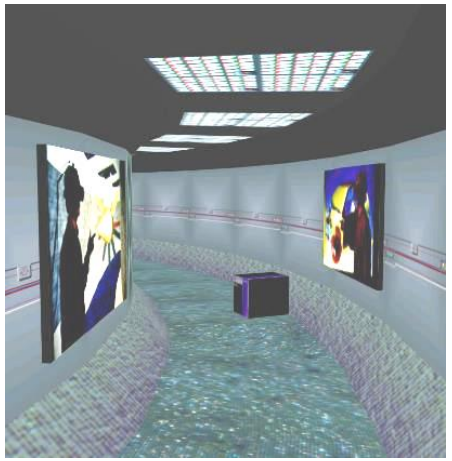


Texturierung

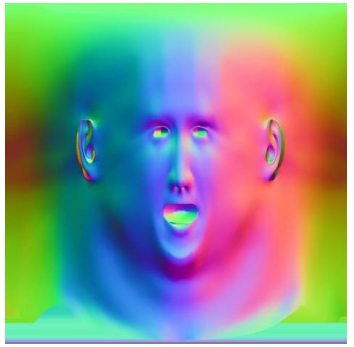


Texturen



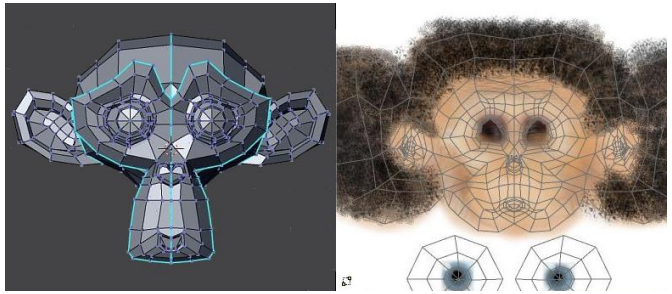


- Bilder müssen nicht nur schnell erzeugt sein, sondern auch realistisch aussehen
  - Texturen sind Bilder, die zur Erhöhung des scheinbaren Detailgrades auf Oberflächen aufgebracht werden
- Sie bedeuten Aufwand pro Bildpunkt und sind ohne GPU-Unterstützung langsam
  - GPU kann auf Pixeln Operationen ausführen
  - Texturen sind Parameter für Per-Pixel Lighting
- Verschiedene Texturtypen möglich
  - Color-, Specular-, Normalmap usw.
  - Cubemaps, Environment-Maps
  - Bumpmaps, Displacement-Maps



Texture Mapping

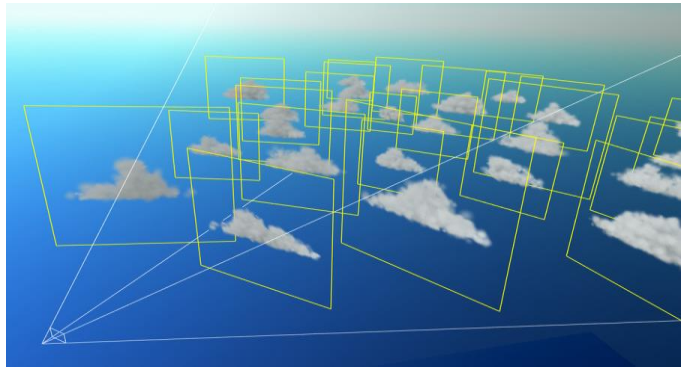
- Mehr Variation ohne Erhöhung der geometrischen Komplexität
- Texturen sind 2D-Bilder beliebiger Größer
  - Vorzugsweise aber Power-of-Two (z.B. 512 × 256)
  - Aber Vorsicht: auch Texture Space ist auf GPU limitiert
- Texturkoordinaten werden angegeben in (*s*, *t*) bzw. (*u*, *v*)
  - Von (0,0) unten links bis (1,1) oben rechts



Alpha-Texturen

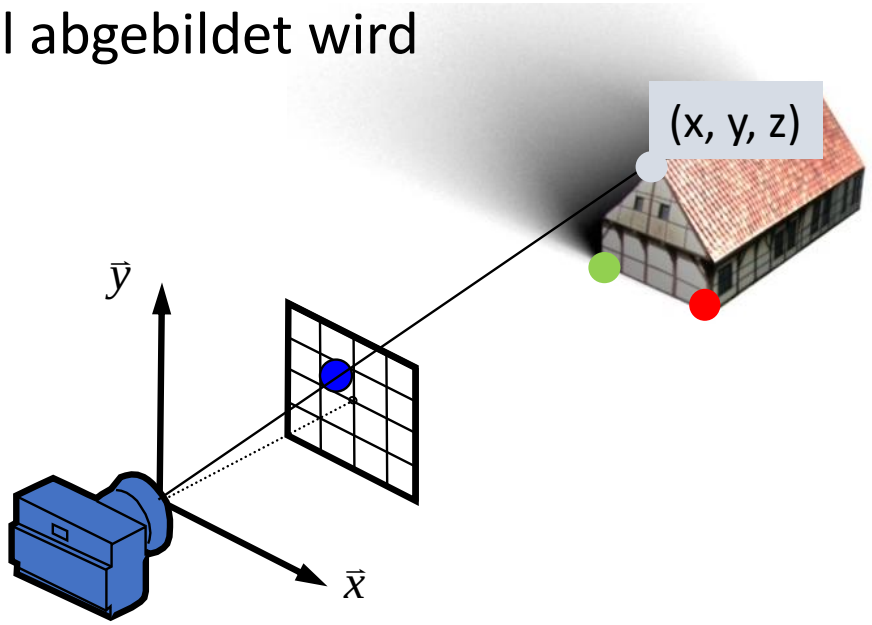
h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

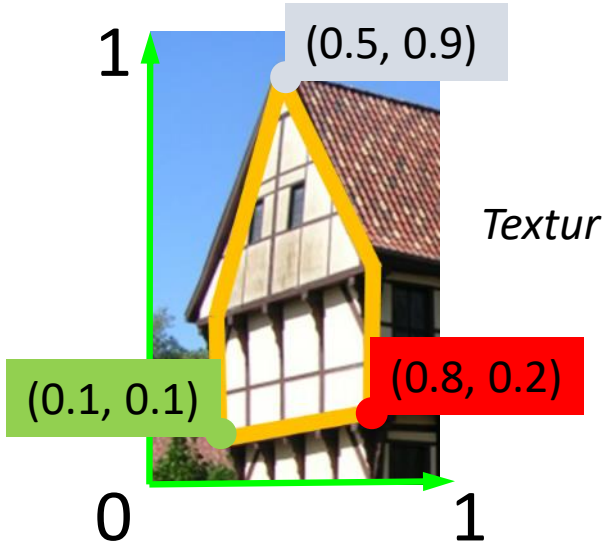
- Alpha-Texturen haben neben den üblichen RGB-Kanälen noch Alpha-Kanal
- Dient als Transparenzkanal und gibt an, ob und wie stark jeweiliges Texel durchsichtig ist
  - Z.B. für getönte Scheiben oder bei Billboards für Wolken, Bäume oder Laternen
    - Semi-Transparenz durch Alpha-Blending (und Z-Sorting)
  - Billboard:
    Texturiertes Quad, immer auf Kamera gerichtet



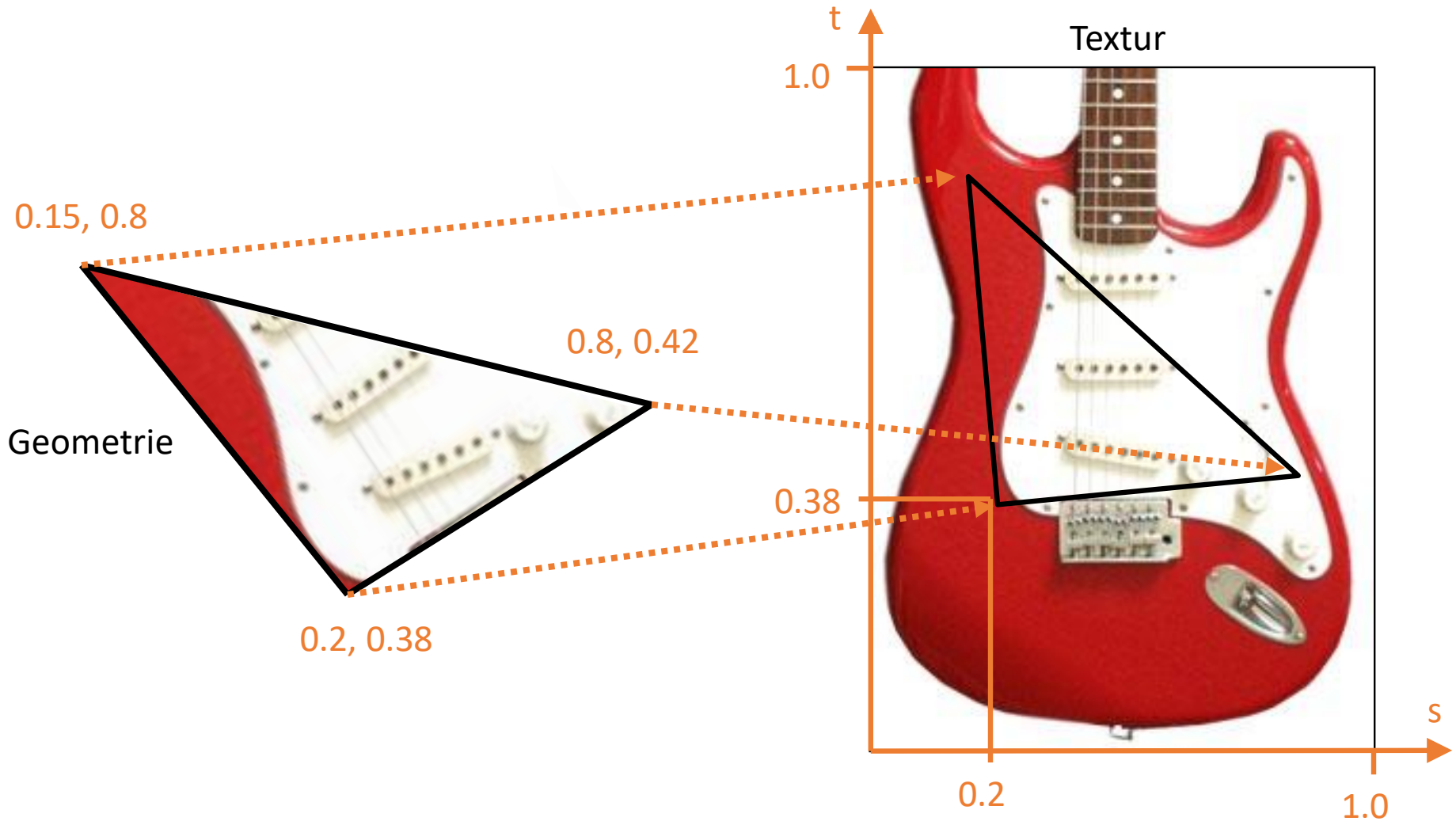
Texturkoordinaten

- Durch Texturkoordinaten ist das zum Polygon korrespondierende Koordinatensystem festgelegt für Texturbild
- Für jedes Pixel wird Oberflächenkoordinate (*s*, *t*) ermittelt
- Diese bestimmt im Koordinatensystem der Textur den Ausschnitt (*Texel*), welcher auf konkreten Pixel abgebildet wird





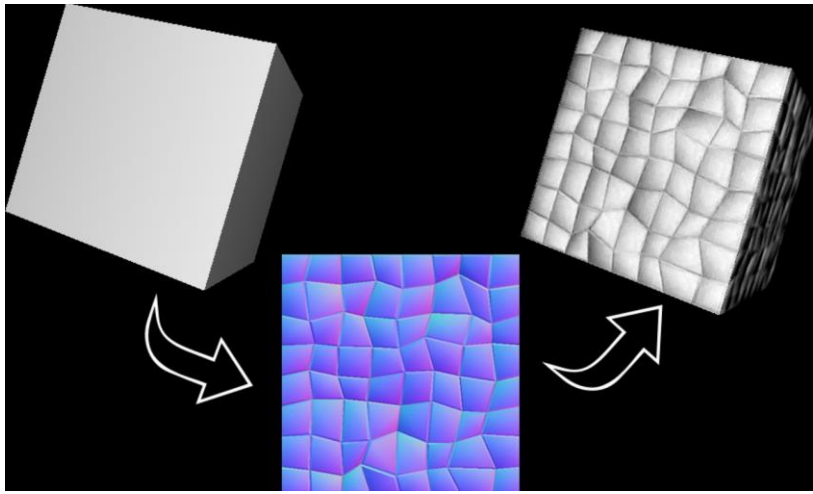
Texture Mapping (2D Image)



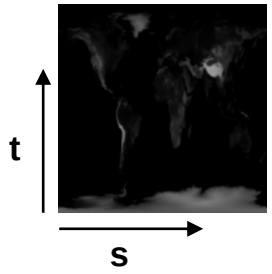
Bump Mapping

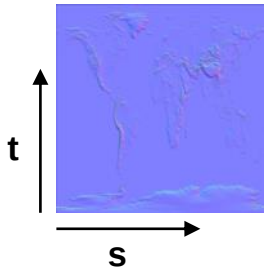
h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

- Vortäuschen höherer Polygonzahl
  - Durch Modifikation der Fragment-Normalen
- Benötigt Tangenten und Binormalen
  - Per-Pixel Lighting im Tangent-Space
- Heightmap hält zu simulierenden Höhenverlauf
  - Berechnung der Normalmap i.A. aus Heightmap
  - Unmodifizierte Tangent-Space Normale: $(0, 0, 1)^T$

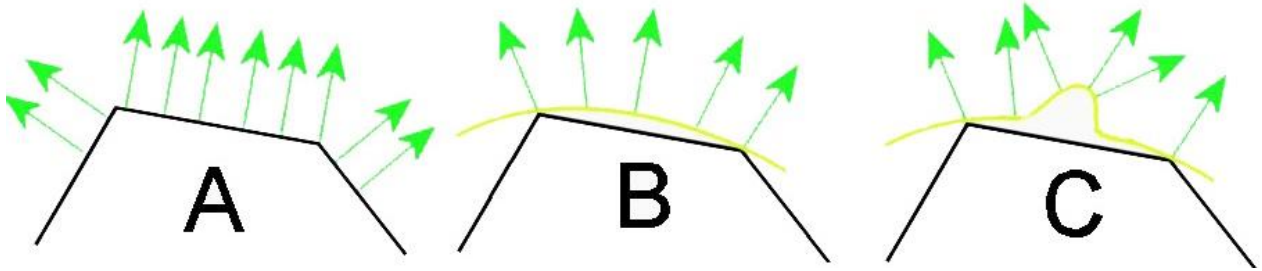


Oberfläche bleibt, nur Normalen werden angepasst



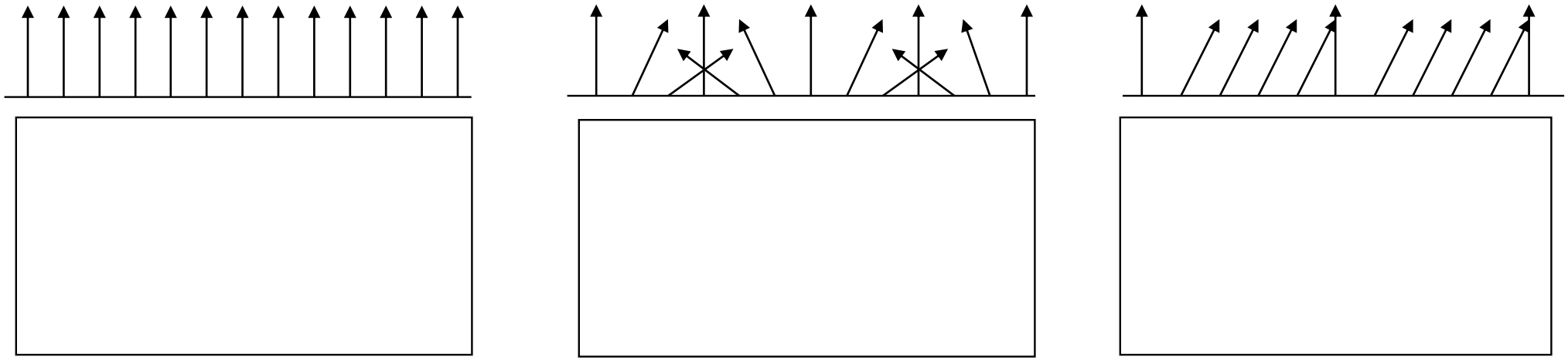


$$M = \begin{bmatrix} T_x & B_x & N_x \\ T_y & B_y & N_y \\ T_z & B_z & N_z \end{bmatrix}^T$$



Übungsaufgabe 4

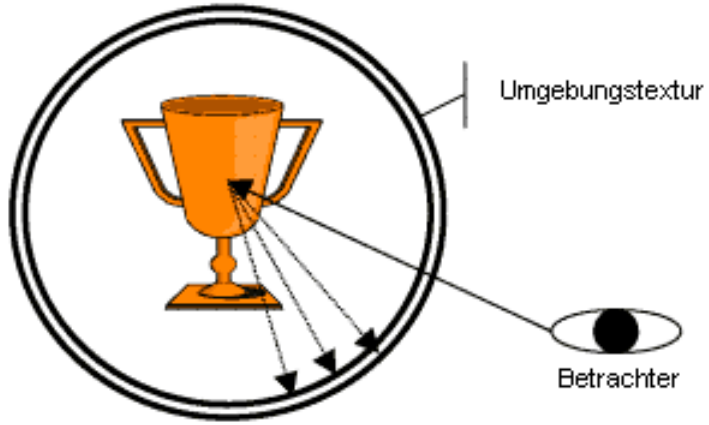
- Die Abbildungen unten zeigen jeweils einen Querschnitt durch eine Oberfläche, wobei die Pfeile die Normalen pro Pixel symbolisieren
- Bei der Beleuchtungsberechnung (während des Phong Shadings) wird die Fläche optisch modelliert. Zeichnen Sie im Querschnitt den visuellen Eindruck, den man aufgrund der eingezeichneten Normalen dadurch von der Geometrie haben wird



Exkurs: Environment Mapping

HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

- Environment Mapping
  - Simuliert Spiegeleffekte ohne Lichtverteilungsberechnung
  - Einfallende Beleuchtungsstärke hängt nur ab von Richtung, nicht von Position eines Punktes auf dem Objekt
- Ansatz
  - Reflektierendes Objekt ist klein im Vergleich zur Umgebung
  - Einfallende Umgebungsbeleuchtung kann vorberechnet und in einer 2D-Textur (Environment Map) gespeichert werden
    - …als Cubemap oder im Lat-Long-Format



h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Vielen Dank!

Noch Fragen?