6.3 STENCIL BUFFER

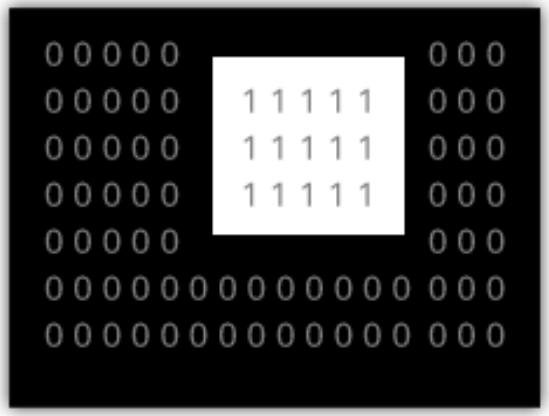
Stencil Buffer

- = Schablonenspeicher: Erlaubt pixelweise Zuweisung von Eigenschaften
- 1-Bit-Stencil: 2 Zustände
  - Dient dem Maskieren von Bereichen, die weiter verarbeitet werden sollen oder die nicht gerendert werden müssen.

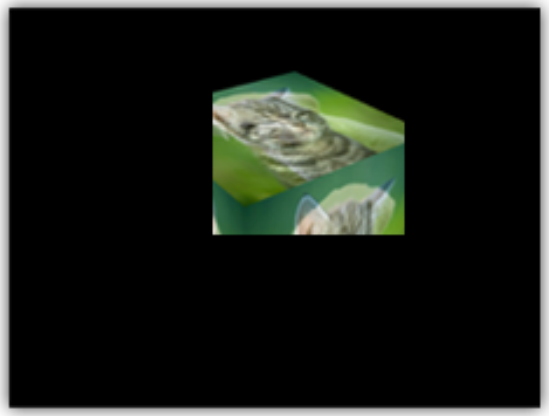
- In OpenGL:
  - Aktivierung mit **`glEnable(GL_STENCIL_TEST);`**
  - Festlegung des Stencil-Tests: **`glStencilFunc(op, ref, mask);`**
    - op: **`GL_NEVER, GL_ALWAYS, GL_EQUAL, GL_NOTEQUAL, GL_LESS, GL_LEQUAL, GL_EQUAL, GL_GREATER`**
    - ref: Integer-Wert mit dem verglichen werden soll
  - Festlegung der Aktion: **`glStencilOp(sfail, dpfail, dppass);`**
    - sfail: Test der glStencilFunc nicht erfolgreich
    - dpfail: Test der glStencilFunc erfolgreich, Depth-Test aber nicht
    - dppass: Test der glStencilFunc und Depth-Test erfolgreich.
    - Kann jeweils den Wert **`GL_KEEP, GL_ZERO, GL_REPLACE, GL_INCR, GL_INCR_WRAP, GL_DECR, GL_DECR_WRAP, GL_INVERT`** annehmen.

Stencil Buffer

Color buffer without stencil test



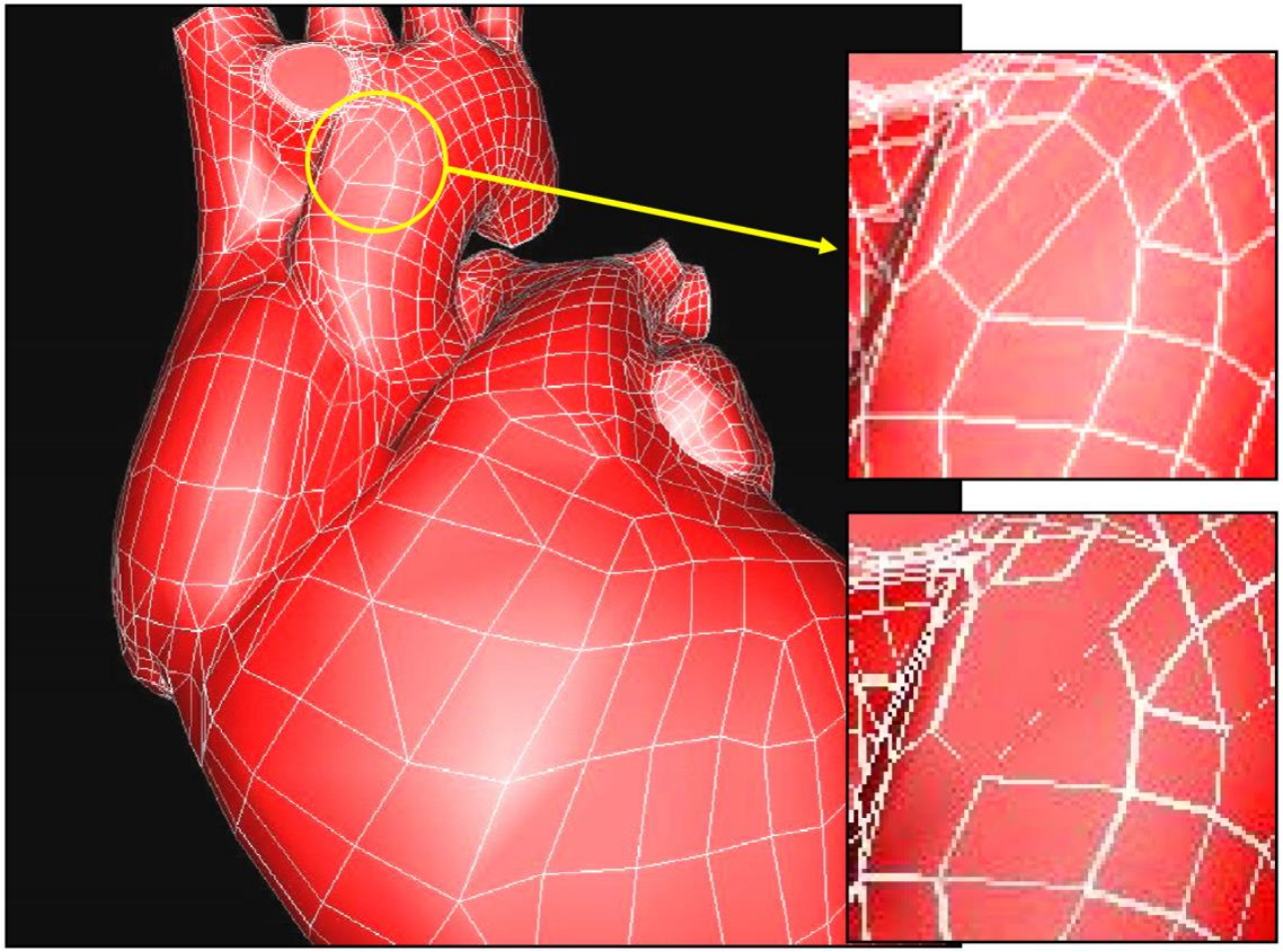
Stencil buffer



Color buffer with stencil test

https://open.gl/depthstencils

Hidden Line mit Stencil Buffer



Mit Stencil Buffer

Ohne Korrektur: Z-Fighting

M. Bailey: Stencil Buffer Examples, Oregon State University

6.4 ACCUMULATION BUFFER

Accumulation Buffer

- Vervielfältigung des Color Buffers:
  - Sammeln von Frames
  - Definierte Aktion zum Zusammenfassen der Frames und Übergabe an Color Buffer

- Nutzung indem Durchschnitt mehrerer Bilder mit unterschiedlichen Einstellungen berechnet wird, z.B. für:
  - Tiefenunschärfe: Änderung des Fokus der virtuellen Kamera
  - Antialiasing komplexer Szenen: leichtes Verschieben der Kamera
  - Bewegungsunschärfe (Motion Blur): unterschiedliche Zeitpunkte
  - Weiche Schatten: durch unterschiedliche Interpretation der Position der Lichtquelle.

Accumulation Buffer

- OpenGL-Befehle:
  - Initialisierung: **`glutInitDisplayMode(…|GLUT_ACCUM|…);`**
  - Operationen auf dem Accumulation Buffer: **`glAccum(operation, f);`**

operation	Formel	Bedeutung
GL_ACCUM	$C'_{acc} = f \cdot C_{col} + C_{acc}$	Addition der mit f skalierten Color-Buffer-Werte und der Accumulation Buffer-Werte
GL_LOAD	$C'_{acc} = f \cdot C_{col}$	Überschreiben des Accumulation Buffer durch mit f skalierten Color-Buffer
GL_ADD	$C'_{acc} = f + C_{acc}$	Addition von f auf die RGBA-Werte des Accumulation Buffer
GL_MULT	$C'_{acc} = f \cdot C_{acc}$	Multiplikation von f mit den RGBA-Werten des Accumulation Buffer
GL_RETURN	$C_{col} = f \cdot C_{acc}$	Kopieren der mit f skalierten Accumulation Buffer-Werte in den aktuellen Color-Buffer

Accumulation Buffer

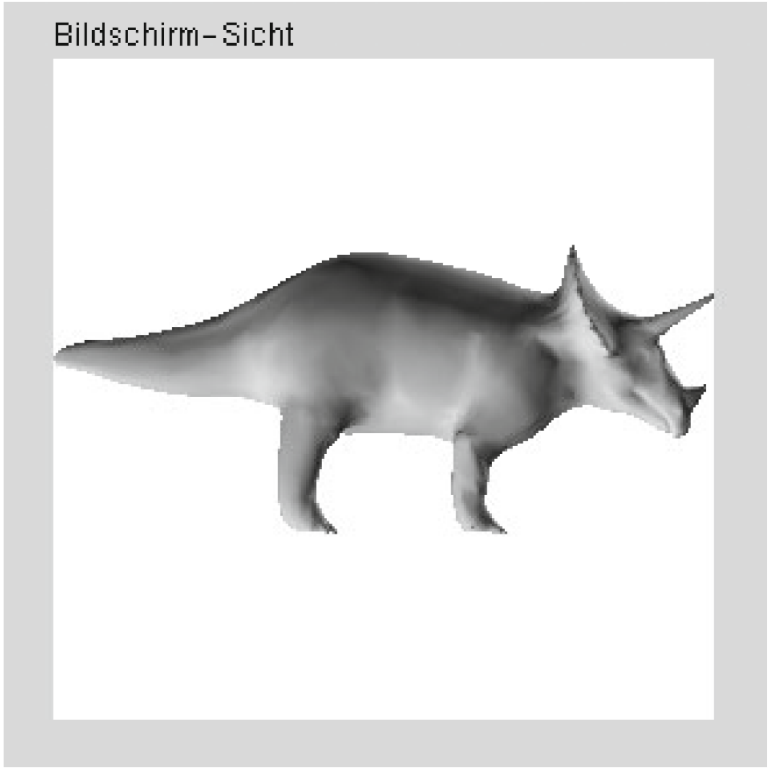
- Beispiel: Szenen-Anti-Aliasing per Accumulation Buffer
  - Idee: Reduzierung des Treppeneffektes durch höhere Abtastrate

- Verarbeitungsschritte:
  1. Zurücksetzen des Accumulation Buffers: **`glClearAccum(background);`**
  2. Simulation einer n-fach höheren Auflösung durch 2^n Bilder die sich um einen Bruchteil eines Pixels unterscheiden.

     Erzeugung der Bilder durch leichte Verschiebung der Kamera

     Aufaddieren der Bilder im Accumulation Buffer: **`glAccum(GL_ACCUM,1.0/2^n);`**
  3. Kopieren des Accumulation Buffers in den Frame Buffer: **`glAccum(GL_RETURN,1.0);`**

Accumulation Buffer



Ohne Anti-Aliasing

Anti-Aliasing mittels Accumulation Buffer

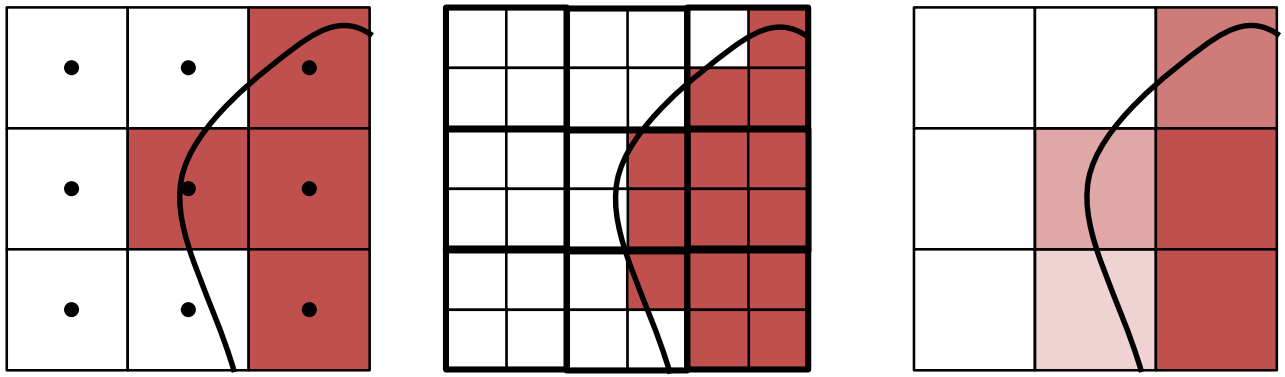
A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Szenen-Anti-Aliasing

Supersampling Anti-Aliasing (SSAA)

- Rastern in 2-fach (4-fach, 8-fach) höherer Auflösung als für Viewport erforderlich
  - Pro Pixel: 2x2 / 4x4 / 8x8 / … Subpixel, die voll gerendert werden
  - End-Intensität eines Pixels im Viewport ergibt sich aus Mittelwert der Subpixel

- Erfordert 4-fachen (16-fache, 64-fachen) Speicher und 4-fache (16-fache, 64-fache) Rechenleistung.



Szenen-Anti-Aliasing

Multisample Anti-Aliasing (MSAA)

- Nachteile von SSAA:
  - SSAA rechnet komplette Rendering-Pipeline für alle Subpixel.
  - Aliasing typischerweise nur an Kanten.
- MSAA optimiert SSAA:
  - Pixel-basierte Berechnungen (z.B. Verdeckung) wird nur einmal für das dargestellte Pixel berechnet, nicht für Subpixel.
  - Zusätzlich kann Kantendetektion eingesetzt werden um Verfeinerung zu Subpixeln auf Kantenpixel zu reduzieren.

- GLUT bietet MSAA an:
  - Initialsierung: **`glutInitDisplayMode(GLUT_MULTISAMPLE);`**
  - Aktivierung: **`glEnable(GLUT_MULTISAMPLE);`**

Szenen-Anti-Aliasing

- Bisherige Verfahren betreiben Anti-Aliasing zum Renderzeitpunkt: Vervielfachung der zu rendernden Pixel.
- GPU-Hersteller bieten heute zusätzlich **Post-Rendering-Anti-Aliasing**:
  - Arbeiten auf bereits fertig gerendertem Bild

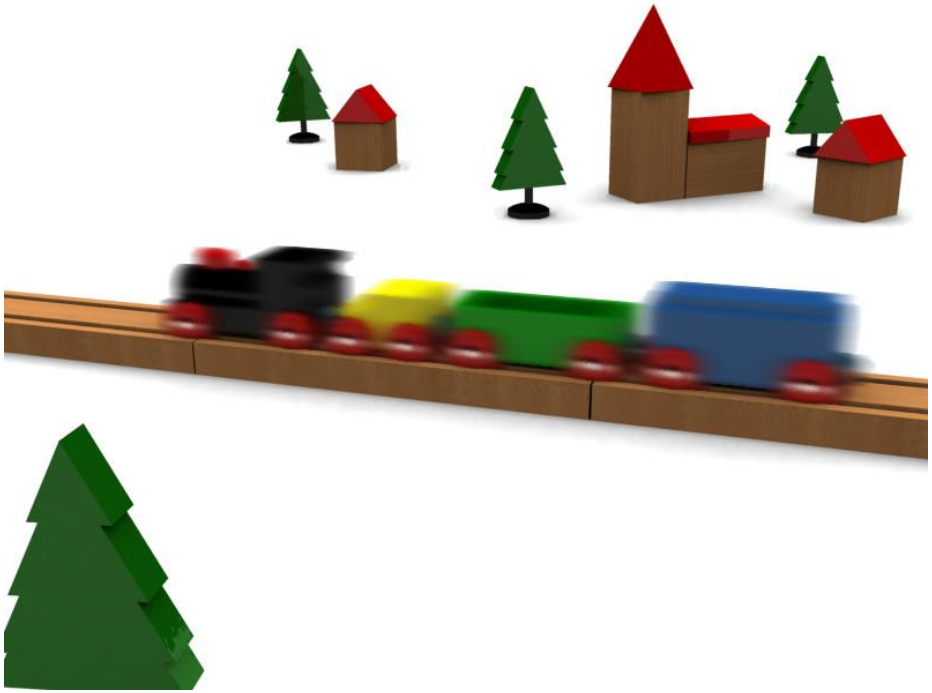
z.B. NVidia FXAA / AMD MLAA:

- Kantendetektion durch Analyse des lokalen Kontrasts
- Glättungsfilter auf detektierte Kantenbereiche

Zeitliches Anti-Aliasing

- Verwischender Effekt durch Akkumulieren mehrerer nacheinander gerenderter Bilder einer Animation („Motion Blurring“)
- Z.B. bei rotierenden Objekten, Eindruck von Bewegung

Motion Blur durch Accumulation Buffer



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“
http://www.dma.ufg.ac.at/app/link/Grundlagen:3D-Grafik/module/13136?step=all

Abfolge der Testoperationen

1. Scissor Test:

   entscheidet über neu zu zeichnende Bildschirmausschnitte

2. Alpha Test:

   gibt an ab welchem Schwellenwert der Durchsichtigkeit nicht mehr gezeichnet werden muss.

3. Stencil Test:

   entscheidet über irregulär geformte Bereiche und die Art, wie darin zu zeichnen ist.

4. Depth Test:

   entscheidet nach z-Wert welches Pixel gezeichnet wird

5. Blending:

   Verrechnet mit dem Hintergrund-Farbwert

6. Dithering:

   mischt die Farben

ZUSAMMENFASSUNG

Zusammenfassung

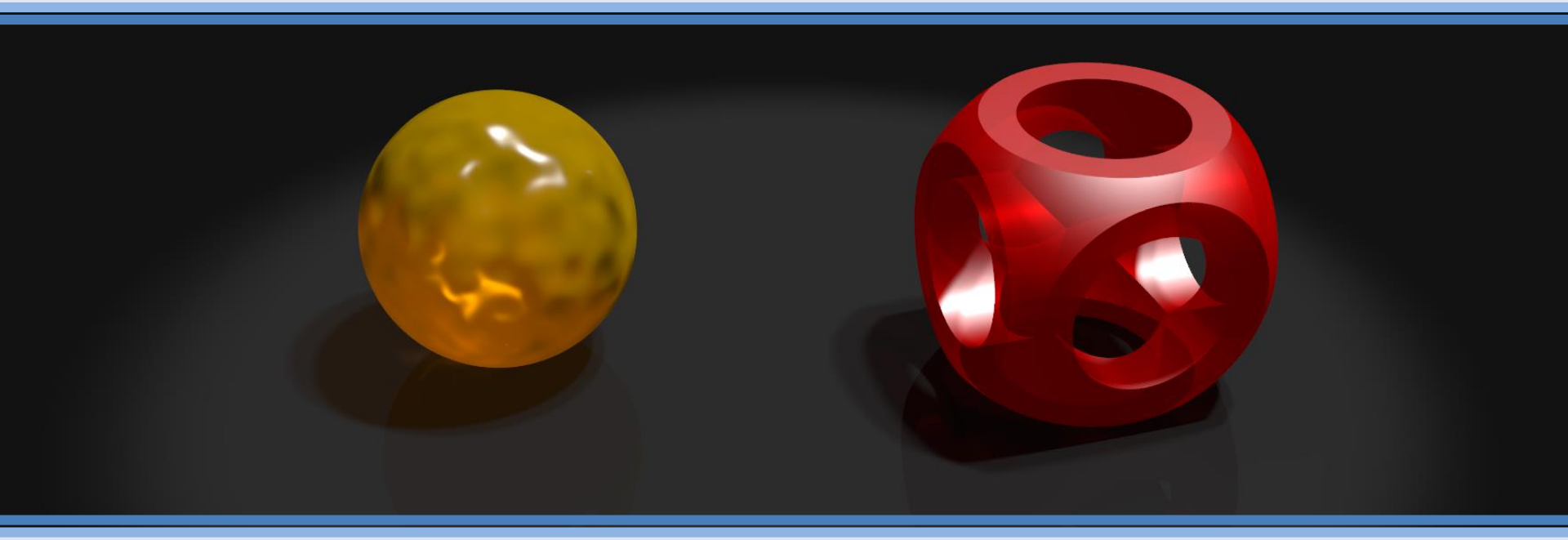
- Buffer: für alle Pixel gespeicherte Daten
- Double/Stereo-Buffer:
  - Verhindern von Flackern durch Hintergrund-Buffer in den gezeichnet wird
  - Stereo-Darstellung
- Depth- bzw. Z-Buffer:
  - Pixelgenaue Verdeckungsberechnung
  - Z-Buffer enthält für jedes Pixel die Tiefeninformation
  - Interpolation der z-Werte je Pixel anhand der Vertices
- Stencil Buffer
  - Pixelgenaue Maskierung
- Accumulation Buffer
  - Sammeln von Frames
  - Definierte Aktion zum Zusammenfassen der Frames und Übergabe an Color Buffer
  - Nutzung der Darstellung von Effekten wie z.B. Tiefenschärfe, Bewegungsunschärfe, Szenen-Antialiasing

Übungsfragen Kapitel 6

- Was ist VSYNC? Wozu wird es verwendet?
- Nennen und Beschreiben Sie ein Verfahren für eine Hidden-Line-Darstellung eines Würfels. Welche Zeichen- und Verarbeitungsschritte sind nötig?
- Was ist z-Fighting? Wann kann es auftreten? Was ist der Grund dafür?
- Beschreiben Sie die Verarbeitungsschritte zur Erzeugung einer Bewegungsunschärfe mit dem Accumulation Buffer.

Computergrafik

T. Hopp



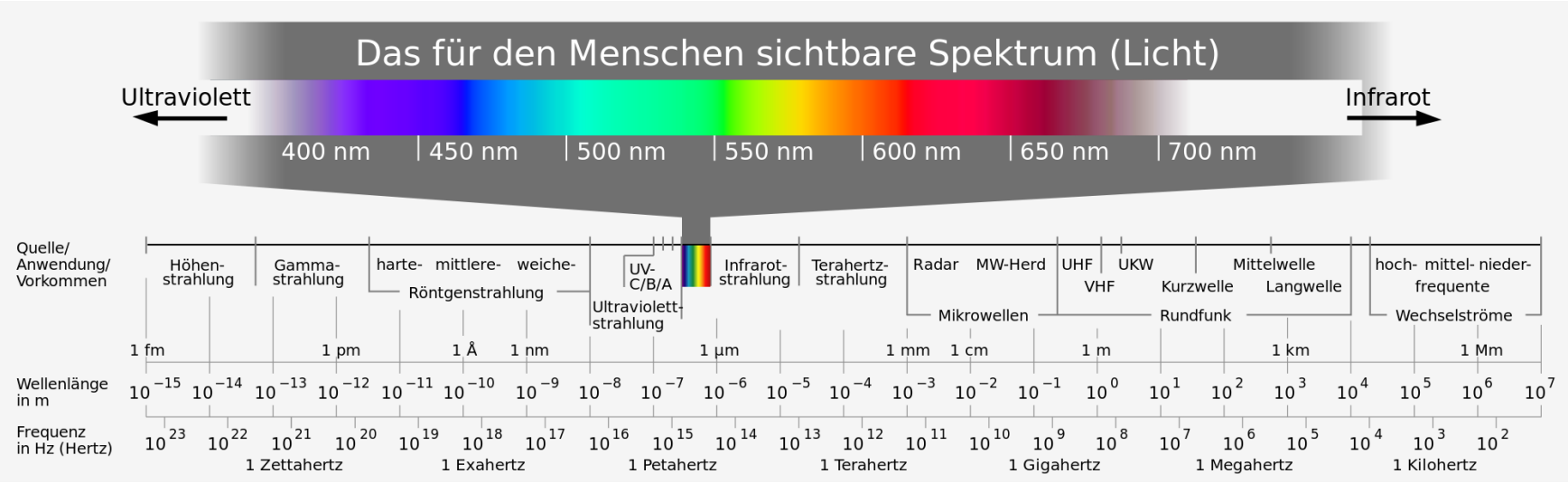
Themenübersicht

1. Einführung
2. Programmierbibliotheken / OpenGL
3. Geometrische Repräsentation von Objekten
4. Koordinatensysteme und Transformationen
5. Zeichenalgorithmen
6. Buffer-Konzepte
7. **Farbe, Beleuchtung und Schattierung**
8. Texturen
9. Animationen
10. Raytracing
11. Volumenvisualisierung

7.1. LICHT UND FARBEN

Licht

- Ursache der Sehwahrnehmung des menschlichen Auges
- Lichtquellen: künstlich oder natürlich
- Menschliches Auge kann Gegenstände nur wahrnehmen wenn von Ihnen Licht ausgeht oder wenn von Ihnen Licht reflektiert wird
- Wellenförmige Ausbreitung elektromagnetischer Strahlen



https://de.wikipedia.org/wiki/Licht

Achromatisches Licht

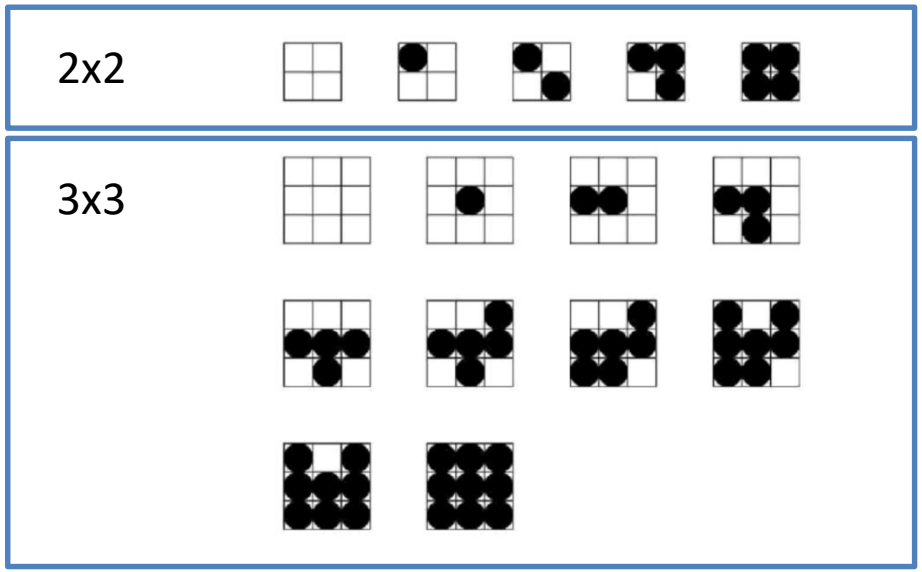
- Achromatisch = Abwesenheit von Farbe
- Betrachtung der Intensität (Luminanz)
  - Meist Angabe als Gleitkommawert zwischen 0.0 und 1.0
  - Alternativ als Integerwert, z.B. zwischen 0 und 255 (8 Bit)

- Konvention:
  - Intensität = 0.0 bzw. 0: schwarz
  - Intensität = 1.0 bzw. 255: weiß

- Darstellbare Intensitätslevel sollten logarithmisch (dem Auge angepasst) werden

Halbtonverfahren

- Darstellung von Intensitätsstufen auf Bi-level-Displays
- Erzeugung durch räumliche Integration:

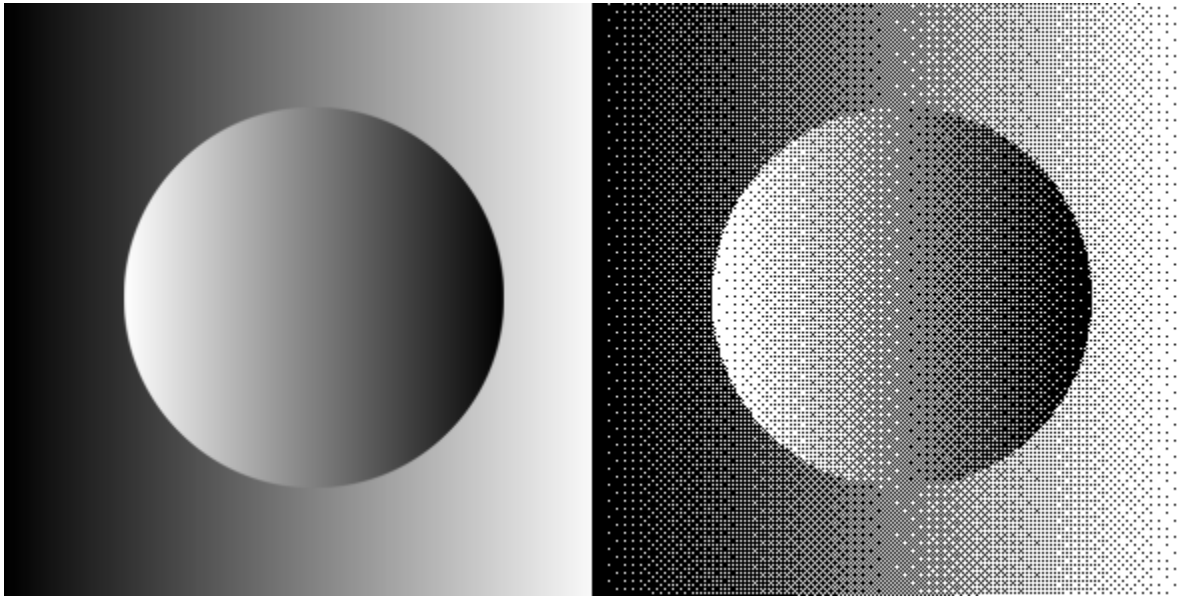


- Durch $n \times n$ Pixel $n^2 + 1$ Intensitätsstufen darstellbar.
- „Dither-Matrizen“

F. Klawonn: Grundkurs Computergrafik mit Java, 3. Auflage, Vieweg+Teubner, 2010

Halbtonverfahren

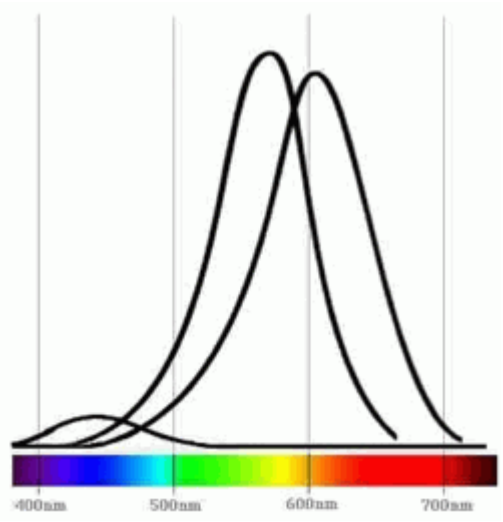
- Beispiel: Farbverlauf mit 8x8 Dither-Matrizen

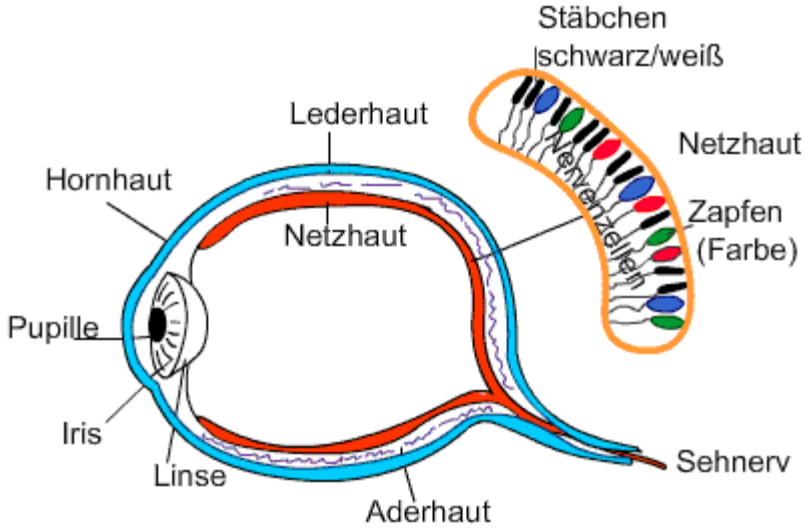


https://de.wikipedia.org/wiki/Dithering_(Bildbearbeitung)

Chromatisches Licht und Farbmodelle

- Chromatisches Licht = „farbiges Licht“
- Das menschliche Auge besitzt 3 Typen von Zapfen, die bei unterschiedlichen Wellenlängen ihre maximale Sensitivität haben



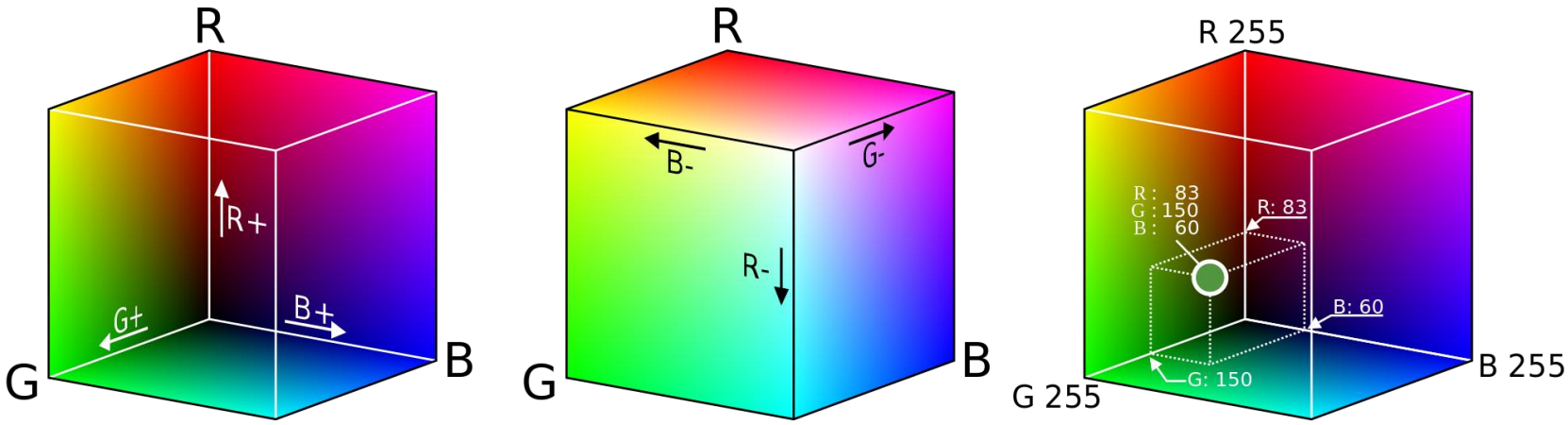


- In der CG Verwendung von Farbmodellen: Eindeutige Zuordnung von Zahlenwerten zu jeder darstellbaren Farbe
  - Unterschiedliche Modelle für unterschiedliche Anwendungsgebiete

http://www.joggysite.de/Seiten/Fotoschule/4_2Farbe_sehen.html
http://www.kidsweb.de/gesundheits_spezial/das_auge.html

Farbmodelle: RGB

- Nachbildung der Farbwahrnehmungen durch additives Mischen der drei Grundfarben Rot (R), Grün (G) und Blau (B).
- Meist in der Computergrafik verwendet

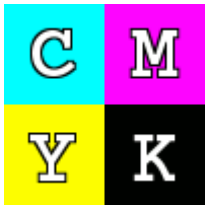


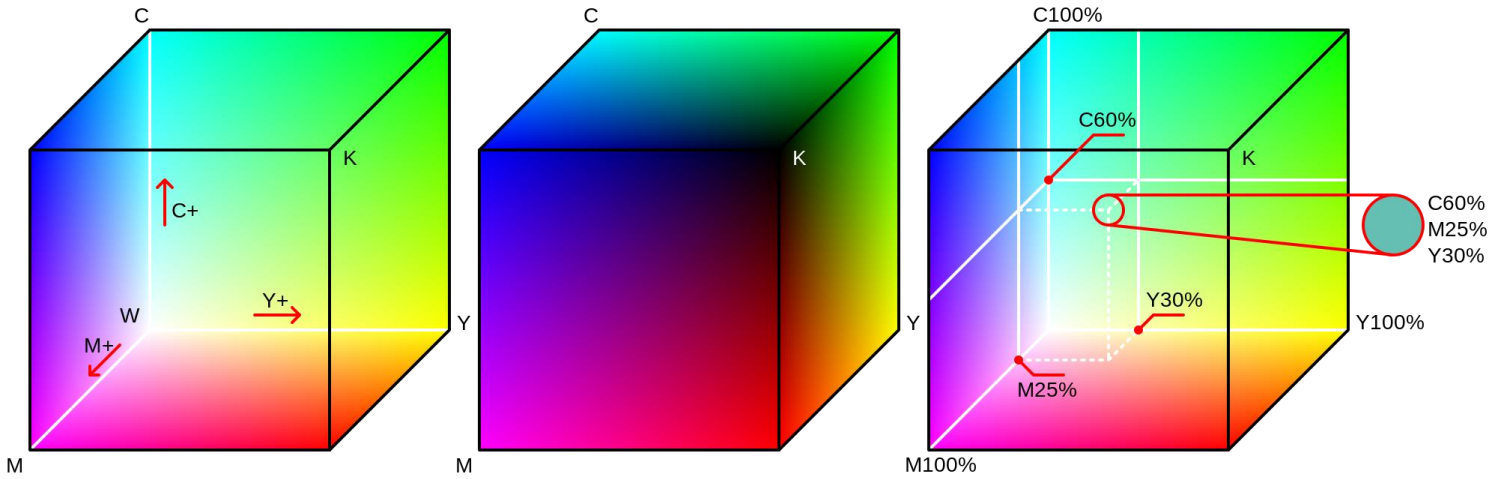
- Wertebereich: typischerweise Gleitkommazahlen zwischen 0.0 und 1.0 oder Integerzahlen zwischen 0 und 255 (= 3 x 8 Bit = 24 Bit Farbtiefe)
- Erweiterung zu RGBA-Modell: Hinzufügen eines Alphakanals für Transparenz

https://de.wikipedia.org/wiki/RGB-Farbraum

Farbmodelle: CMY(K)

- Ursprung in der Drucktechnik: subtraktives Farbmodell.
- Stellt Farben durch Mischen von Cyan (C), Magenta (M) und Gelb (Y) dar.
- Der Schwarzanteil wird im K-Kanal codiert.





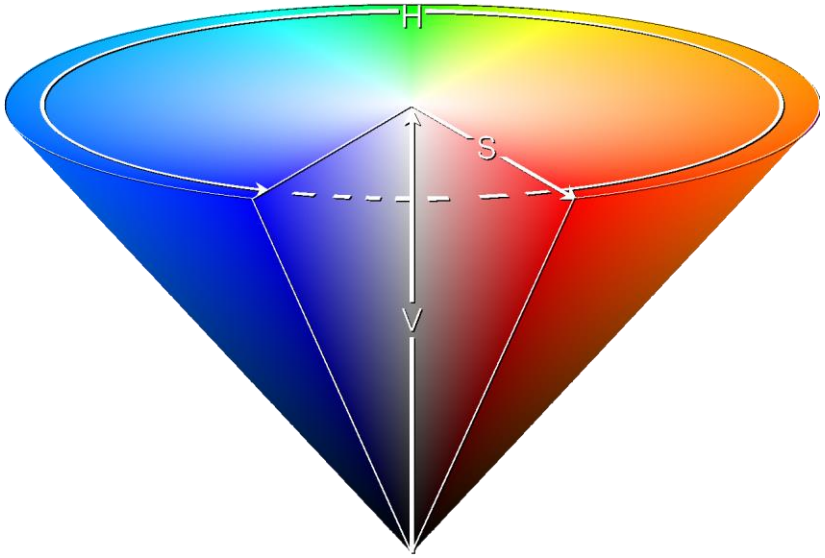
- Wertebereich: typischerweise Angabe in % zwischen 0 und 100
- Dualität von RGB und CMYK: Umrechnung erfolgt durch

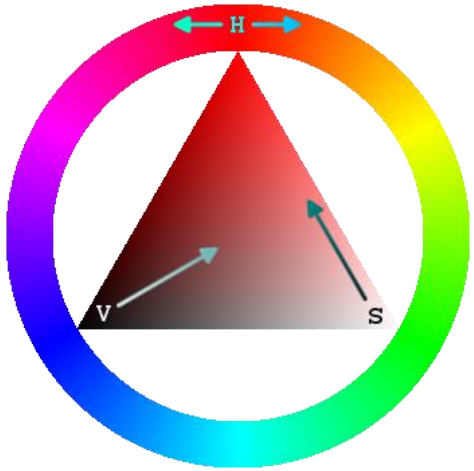
$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad \text{bzw.} \quad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

https://de.wikipedia.org/wiki/CMYK-Farbmodell

Farbmodelle: HSV

- Ähnelt der menschlichen Farbwahrnehmung: Definition einer Farbe durch
  - Farbton H: Farbwinkel auf einem Farbkreis: $0° =$ rot, $120° =$ grün, $240° =$ blau
  - Sättigung S: Angabe in %: $0\% =$ grau bis $100\% =$ gesättigte reine Farbe
  - Hellwert V: Angabe in %: $0\% =$ keine Helligkeit, $100\% =$ volle Helligkeit





https://de.wikipedia.org/wiki/HSV-Farbraum

Farbmodelle: HSV

- Umrechnung von HSV zu RGB:
  - Vorbedingung: $H \in [0°, 360°]$, S und $V \in [0,1]$.
  - $C = V \cdot S$
  - $X = C \cdot \left(1 - \left|\frac{H}{60°} \; mod \; 2 \; - \; 1\right|\right)$
  - $m = V - C$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0° \leq H < 60° \\ (X, C, 0) & , 60° \leq H < 120° \\ (0, C, X) & , 120° \leq H < 180° \\ (0, X, C) & , 180° \leq H < 240° \\ (X, 0, C) & , 240° \leq H < 300° \\ (C, 0, X) & , 300° \leq H < 360° \end{cases}$$

  - $(R, G, B) = ((R' + m) \cdot 255, \; (G' + m) \cdot 255, \; (B' + m) \cdot 255)$

Farbmodelle: HSV

- Umrechnung von HSV zu RGB: Beispiel
  - $H = 60°,\ S = 1,\ V = 0{,}5$
  - $C = V \cdot S\ =\ 0{,}5$
  - $X = C \cdot \left(1\ -\ \left|\frac{H}{60°}\ mod\ 2\ -\ 1\right|\right) = 0{,}5 \cdot \left(1\ -\ \left|\frac{60°}{60°}\ mod\ 2\ -\ 1\right|\right) = 0{,}5$
  - $m = V - C = 0{,}5 - 0{,}5 = 0$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0° \leq H < 60° \\ (X, C, 0) & , 60° \leq H < 120° \\ (0, C, X) & , 120° \leq H < 180° \\ (0, X, C) & , 180° \leq H < 240° \\ (X, 0, C) & , 240° \leq H < 300° \\ (C, 0, X) & , 300° \leq H < 360° \end{cases}$$

$R' = 0{,}5$
$G' = 0{,}5$
$B' = 0$

  - $(R, G, B) = ((R' + m) \cdot 255,\ (G' + m) \cdot 255,\ (B' + m) \cdot 255)$
    $= ((0{,}5 + 0) \cdot 255, (0{,}5 + 0) \cdot 255, (0 + 0) \cdot 255)$
    $= (128, 128, 0)$

Wechselwirkung von Licht und Materie

- Physikalische Grundlagen beruhen auf Welle-Teilchen-Dualismus
  - Modellierung der Ausbreitung von Licht als elektromagn. Welle: Wellentheorie
  - Modellierung der Wechselwirkung von Licht und Materie: Teilchentheorie

- Viele Effekt nur im mikroskopischen Bereich relevant.

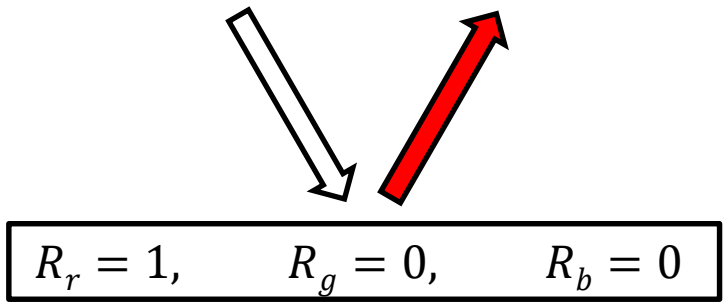
- Vereinfachungen in der Computergrafik auf makroskopische Effekte
  - Lichtausbreitung in homogenen Medien wird als gerader Lichtstrahl modelliert
  - Kontinuierliches Spektrum wird an drei Stellen abgetastet: rot, grün, blau
  - Interferenz, Beugung, Polarisation werden vernachlässigt
  - Wechselwirkung von Licht und Materie wird stark vereinfacht.

➔ Geometrische Optik

Wechselwirkung von Licht und Materie

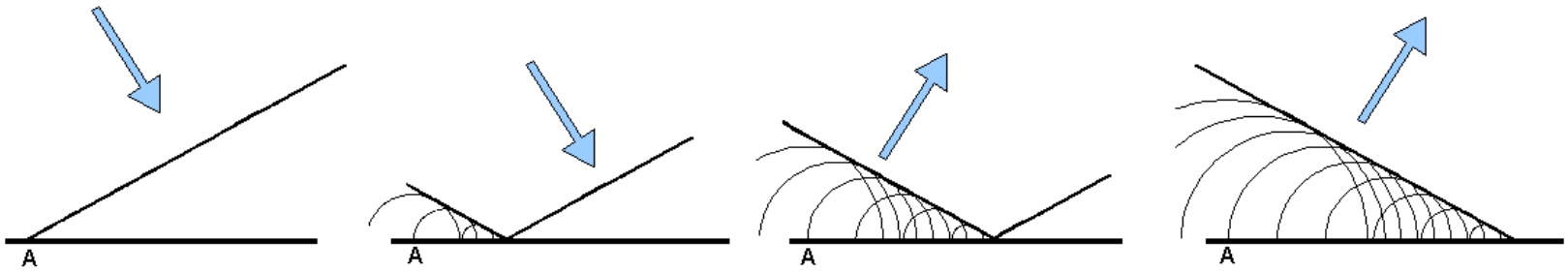
- Photonen regen Elektronen und Protonen zur Schwingung an
  - Wenn Anregungsfrequenz = Resonanzfrequenz des Materials → Absorption
  - Sonst: Streuung/Reflexion

- Resonanzfrequenz (=Materialkonstante) bestimmt Absorptions- ($A(\lambda)$) und Reflexionskoeffizient ($R(\lambda)$).
- Energieerhaltung: $A(\lambda) + R(\lambda) = 1$
- Computergrafik: Abtastung an drei Wellenlängen reduziert Koeffizienten auf drei Komponenten: z.B. (R_r, R_g, R_b)
- Oberfläche erscheint bei Bestrahlung mit weißem Licht in der Farbe entsprechend der Größe der drei Reflexionskoeffizienten.



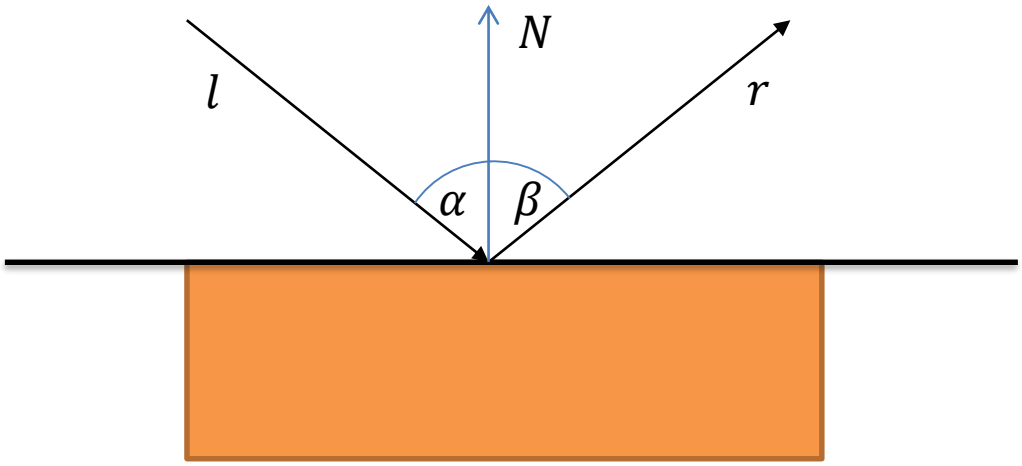
Streuung / Reflexion

- Ausbreitung des Lichts als Wellenfront
- Streuung folgt dem Huygen‘schen Prinzip
  - In jedem Punkt einer Wellenfront sitzt ein Streuzentrum, von dem aus elementare Kugelwellen ausgehen.
  - Überlagerung aller Kugelwellen ergibt eine neue Wellenfront



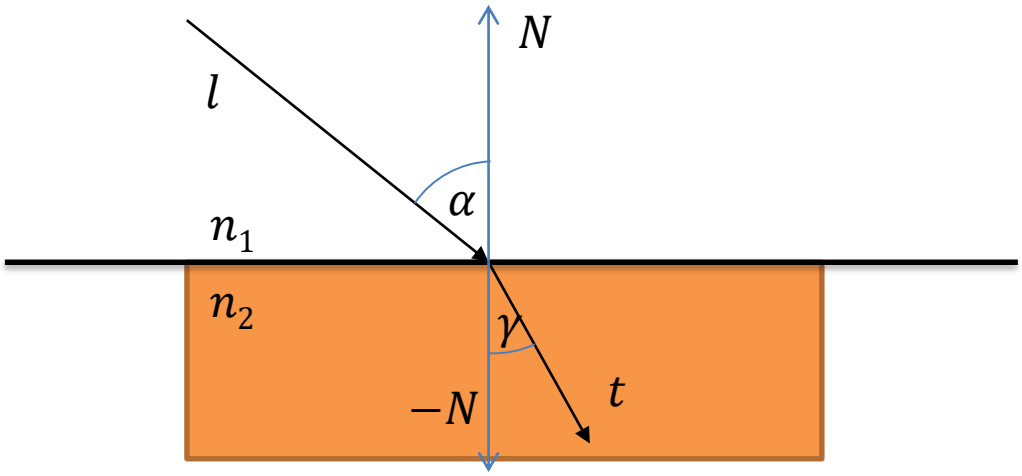
https://de.wikipedia.org/wiki/Huygenssches_Prinzip

Reflexionsgesetz



- Reflexion erfolgt an Oberflächen bzw. Mediengrenzen
- Einfallswinkel α = Reflexionswinkel β
- Hier: Ideale Reflexion → komplettes einfallendes Licht wird reflektiert

Brechungsgesetz

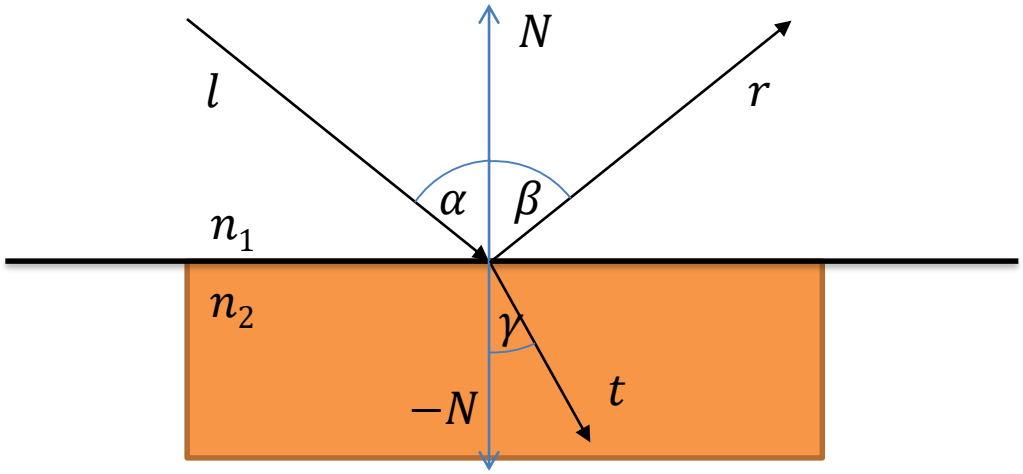


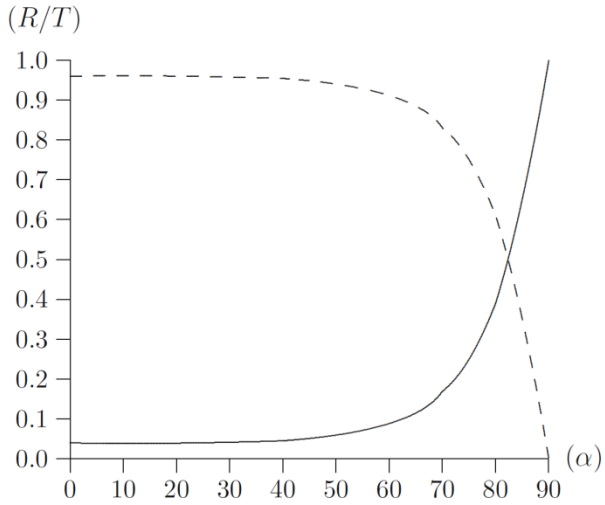
- Verhältnis zwischen dem Sinus des Einfallswinkels α und dem Sinus des Brechungswinkels γ entspricht dem Verhältnis der Brechungsindizes n_1 und n_2 der beiden Medien:

$$\frac{\sin \alpha}{\sin \gamma} = \frac{n_2}{n_1}$$

- Dispersion: Brechungsindex ist abhängig von Wellenlänge des Lichtes
  - D.h. dreifacher Berechnungsaufwand für die Brechungsindizes bei Rot, Grün, Blau

Fresnel-Effekt



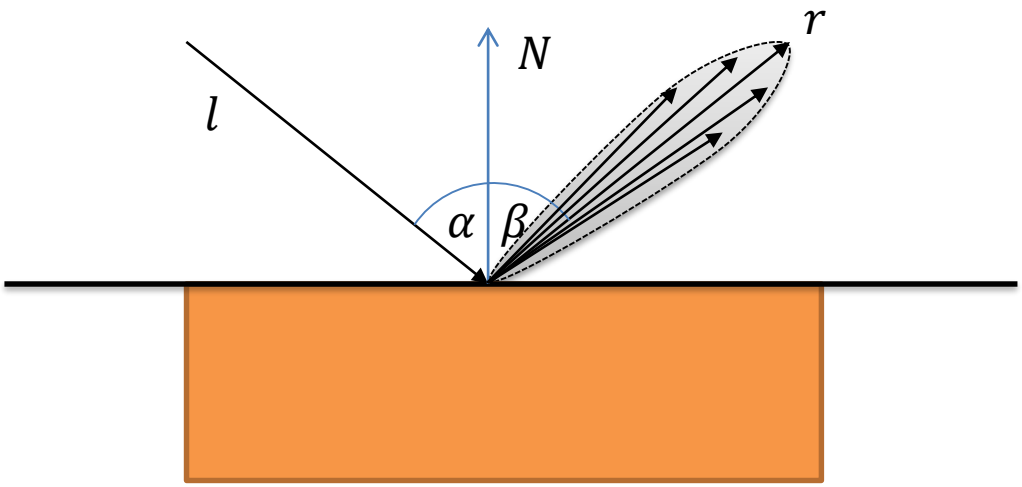


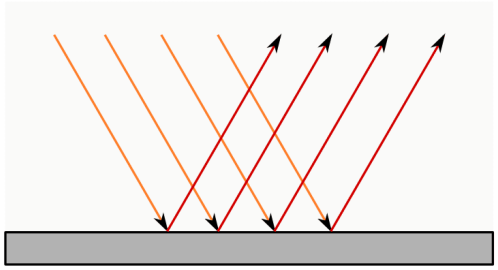
- Bei der Brechung wird auch an ideal glatten Grenzflächen ein Anteil des Lichts reflektiert
- Je größer (flacher) der Einfallswinkel
  - desto größer wird der Anteil des reflektierten Lichts
  - desto geringer wird der Anteil des gebrochenen Lichts
- Vereinfachte Gleichung: $R = \frac{1}{2}\left(\frac{\sin^2(\gamma-\alpha)}{\sin^2(\gamma+\alpha)} + \frac{\tan^2(\gamma-\alpha)}{\tan^2(\gamma+\alpha)}\right)$

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

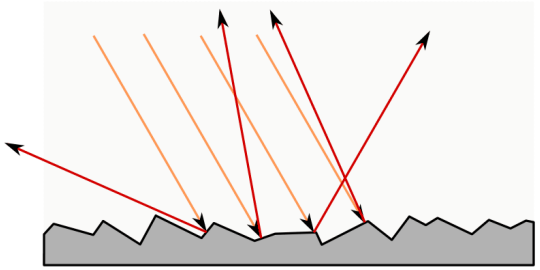
Diffuse Reflexion

- Bisher: ideale Reflexion
- Realität: Rauigkeit einer Oberfläche spaltet Lichtstrahl auf (→ Leuchtkörper)





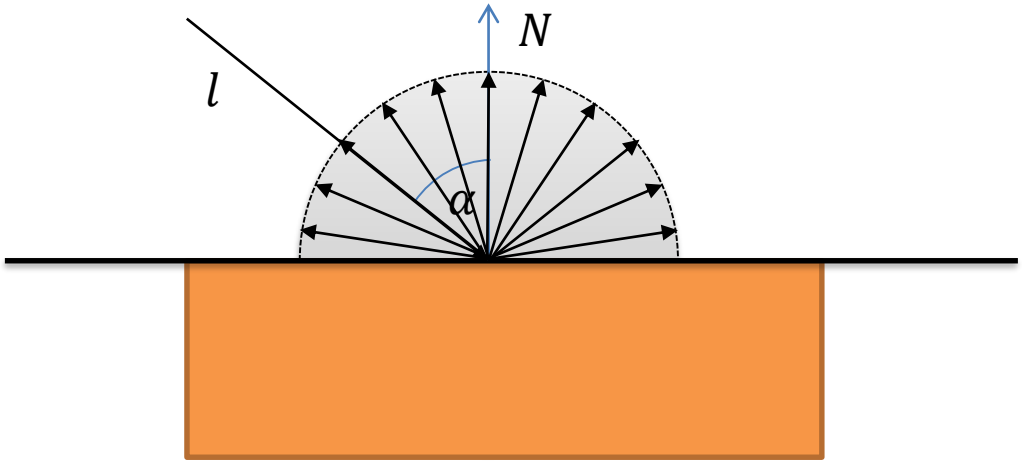
Glatter Spiegel
Direkte Reflektion



Rauher Spiegel
Diffuse Reflektion

http://www.grund-wissen.de/physik/optik/lichtreflexion.html

Ideal diffuse Reflexion



- Lambert‘sches Streumodell: gleichmäßige Streuung des einfallenden Lichtes in alle Richtungen (ideal diffus streuend)
- Allgemein: beliebig geformte Leuchtkörper, Anisotropie etc.

Bidirectional Reflectance Distribution Function

- Allgemein: Anteil der Lichtintensität der in eine Raumrichtung reflektiert wird abhängig von 5 Variablen:
  - Elevations- und Azimuthwinkel des einfallenden Strahls α und θ
  - Elevations- und Azimuthwinkel des reflektierten Strahls β und φ
  - Wellenlänge des Lichtes λ.

- Bei bekannten Materialeigenschaften kann Intensität der in alle Raumrichtung reflektierten Strahlen I_r berechnet werden:

$$I_r(\alpha, \theta, \lambda) = R(\alpha, \beta, \theta, \varphi, \lambda) \cdot I_e\ (\alpha, \theta, \lambda) \cdot \cos(\alpha)$$

- Geht von der Idealvorstellung punktförmiger Quellen aus.
- Bei Lichtquellen mit endlicher Ausdehnung treffen Lichtstrahlen aus verschiedenen Raumrichtungen auf die Oberfläche: es muss über die Fläche der Lichtquelle integriert werden → aufwendige Integralformel!
- Analog lässt sich die Bidirectional Transmssion Distribution Function für Brechung berechnen.

7.2. BELEUCHTUNGSMODELLE

Lokale und globale Beleuchtung

- Trotz aller Vereinfachungen bleibt Beleuchtungsrechnung komplex:
  - Z.B. reflektiertes Licht von Oberflächen entspr. erneutem Aussenden von Licht, ausgehend von allen Punkten auf dieser Oberfläche

- Lokale Beleuchtung:
  - Nur punktförmige Lichtquellen
  - Berechnung der (einfachen) Reflexion anhand z.B. BRDF
  - Interaktive Computergrafik

heute

- Globale Beleuchtung:
  - Berechnet auch Licht das vom Punkt einer Oberfläche ausgestrahlt wird (indirekte Beleuchtung)
  - Integralformel! → sehr aufwändig.
  - Raytracing, Radiosity.

Kapitel 10

Phong Reflexionsmodell

- Einfaches Reflexions- bzw. Beleuchtungsmodell für die interaktive Computergrafik
  - Lokales Beleuchtungsmodell
  - Streulicht von Oberflächen: Approximation durch „*ambient*“
  - Kein Schattenwurf
  - Abtastung des kontinuierlichen Spektrums an drei Stellen: Rot, Grün, Blau
  - Atmosphärische Effekte (z.B. Verblassung) werden nicht berücksichtigt
  - Transparenz wird vernachlässigt
  - Brechungseffekte, Dispersion, Fresnel-Effekte werden vernachlässigt

- Farbe eines (elementaren) Objektes wird definiert über
  - Zugehörigkeit zu einem Objekt mit definierten Materialeigenschaften
  - Farbe und Position der Lichtquelle

Phong Reflexionsmodell

- Realisierung über drei Beleuchtungskomponenten
  - *Ambiente*: ungerichtetes Streulicht, Ersatz für globale Beleuchtung
  - *Diffuse*: gerichtetes Licht - Lambert‘sches Beleuchtungsmodell – ideal diffuse Oberfläche
  - *Spekulare*: gerichtetes Licht - spiegelnde Oberfläche

- Summe der 3 Beleuchtungskomponenten ergibt gesamte Intensität des reflektierten Lichtes an einem Punkt der Oberfläche:

$$I = I_{ambiente} + I_{diffus} + I_{spekular}$$

- Teilweise vierte Komponente: *emissive*
  - Selbstleuchtende Oberfläche, unabhängig von Umgebungslicht
  - Phong: Emissive Oberflächen beleuchten andere Oberflächen nicht.

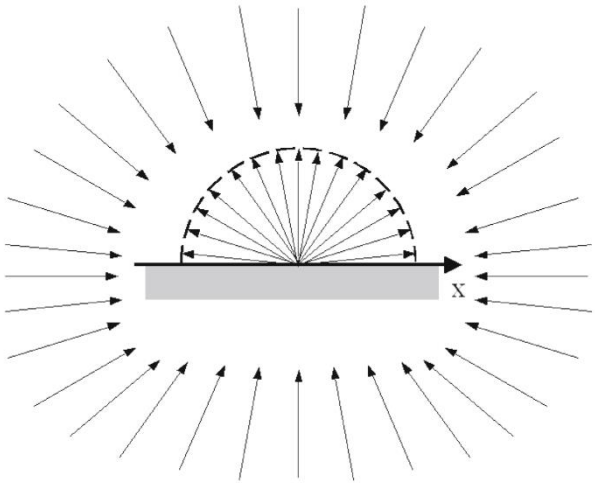
Umgebungslicht (*ambiente*)

- Approximation von Licht das nicht direkt auf die Oberfläche fällt, sondern zuerst von anderen Oberflächen gestreut wird.
  - Unabhängig vom Einfallswinkel des Lichtes

$$I_{ambiente} = I_a \cdot k_a$$

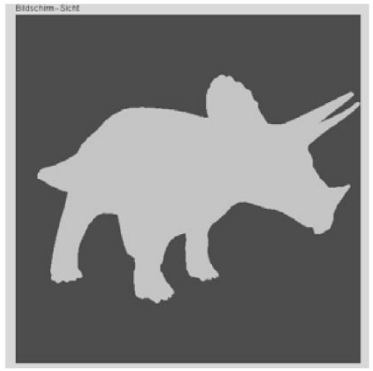
Materialkonstante

Intensität des Umgebungslichtes



Ungerichtetes Licht aus allen Richtungen

(a)



(b)

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Diffuses Streulicht (*diffus*)

- Ideal diffuse Reflexion an der Oberfläche:
  - Reflektierte Lichtstärke unabhängig vom Standpunkt des Betrachters
  - Reflektierte Lichtstärke abhängig vom Einfallswinkel α.

$$I_{diffus} = I_{in} \cdot k_d \cdot \cos(\alpha)$$
$$I_{diffus} = I_{in} \cdot k_d \cdot (l \cdot N)$$

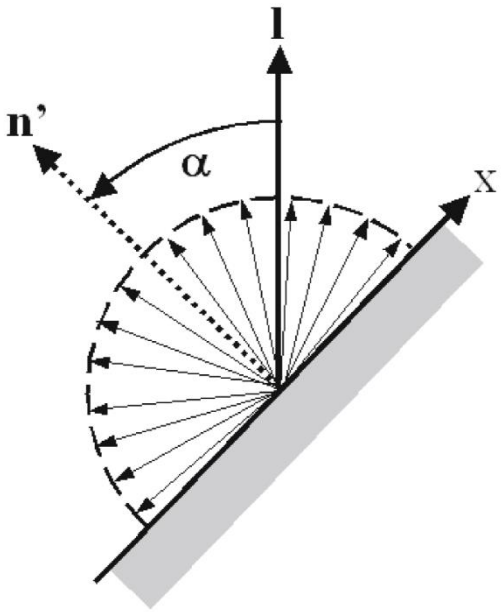
Skalarprodukt: $\cos(\alpha) = \frac{l \cdot N}{|l| \cdot |N|}$

Empirisch bestimmter Reflexionsfaktor (Materialkonstante)

Intensität des einfallenden Lichtstrahls der Punktlichtquelle

- Nur Oberflächen die der Lichtquelle zugewandt sind können beleuchtet werden: $\cos(\alpha)$ wird über Maximumfunktion ausgedrückt: $\max(l \cdot N, 0)$
- Bei n Lichtquellen ist die Summe über alle einzelnen Intensitätsquellen und die Richtungen zu bilden: $I_{diffus} = k_d \sum_n I_{in,n}(L_n \cdot N)$

Diffuses Streulicht (*diffus*)

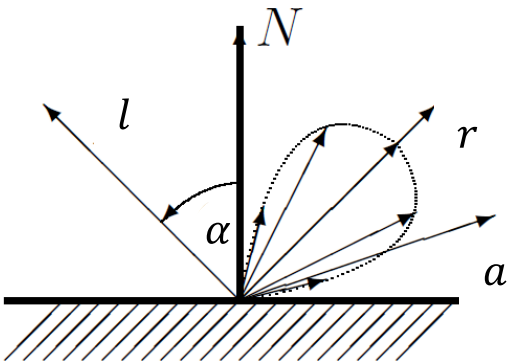


A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Spiegelnde Reflexion (*spekular*)

- Größte Lichtintensität in der idealen Reflexionsrichtung r.

- Je größer der Winkel ϕ zwischen r und dem Augpunktvektor a wird, desto kleiner wird die gespiegelte Lichtintensität in Richtung Augpunkt



$$I_{spekular} = I_{in} \cdot k_s \cdot (\cos(\phi))^s$$

$$I_{spekular} = I_{in} \cdot k_s \cdot (r \cdot a)^s$$

Spiegelungsexponent

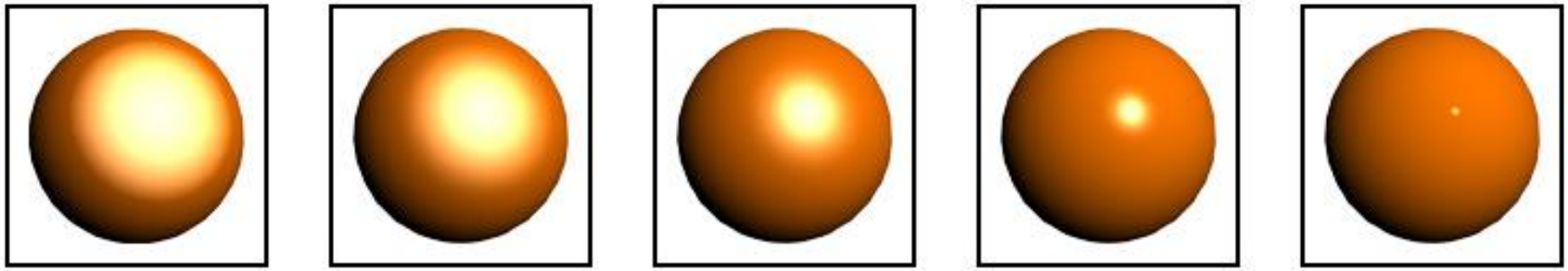
Empirisch bestimmter Reflexionsfaktor (Materialkonstante)

Intensität des einfallenden Lichtstrahls der Punktlichtquelle

- Große S: sehr glatte Oberfläche (punktförmiges Glanzlicht)
- Kleine S: matte Oberfläche (ausgedehntes Glanzlicht)

Spiegelnde Reflexion (*spekular*)

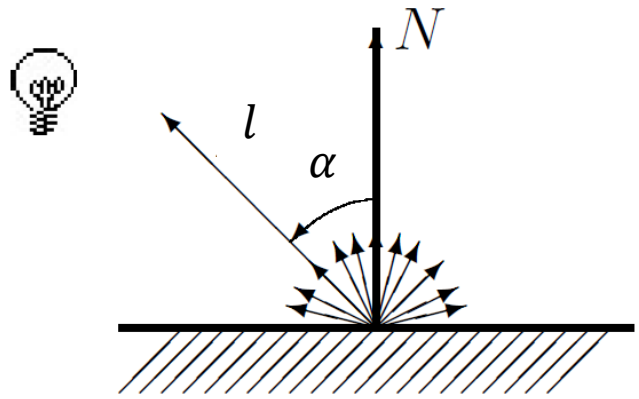
- Auswirkung des Spiegelungsexponenten

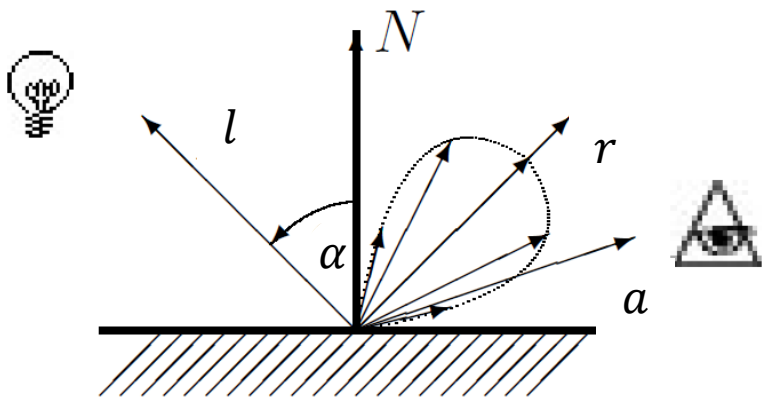


http://www.dma.ufg.ac.at/app/link/Grundlagen%3A3D-Grafik/module/9746?step=2

Vergleich diffuse/spiegelnde Reflexion

Diffuse Reflexion	Spiegelnde Reflexion
Unabhängig von Betrachterposition	**Abhängig** von Betrachterposition
Farbe des reflektierten Lichtes = **Farbe des Objektes + Farbe der Lichtquelle**	Farbe des reflektierten Lichtes oft nur **Farbe der Lichtquelle**





S. Krömker: Computergrafik I, Universität Heidelberg

Phong Reflexionsmodell

- Aus der Summe der Beleuchtungskomponenten ergibt sich:

$$I = I_a k_a + I_{in} k_d \, (l \cdot N) + I_{in} k_s (ra)^S$$
$$= I_a k_a + I_{in} (k_d \, (l \cdot N) + k_s (ra)^S)$$

- Bei mehreren Lichtquellen wird über alle Intensitäten und Positionen der verschiedenen Quellen aufsummiert:

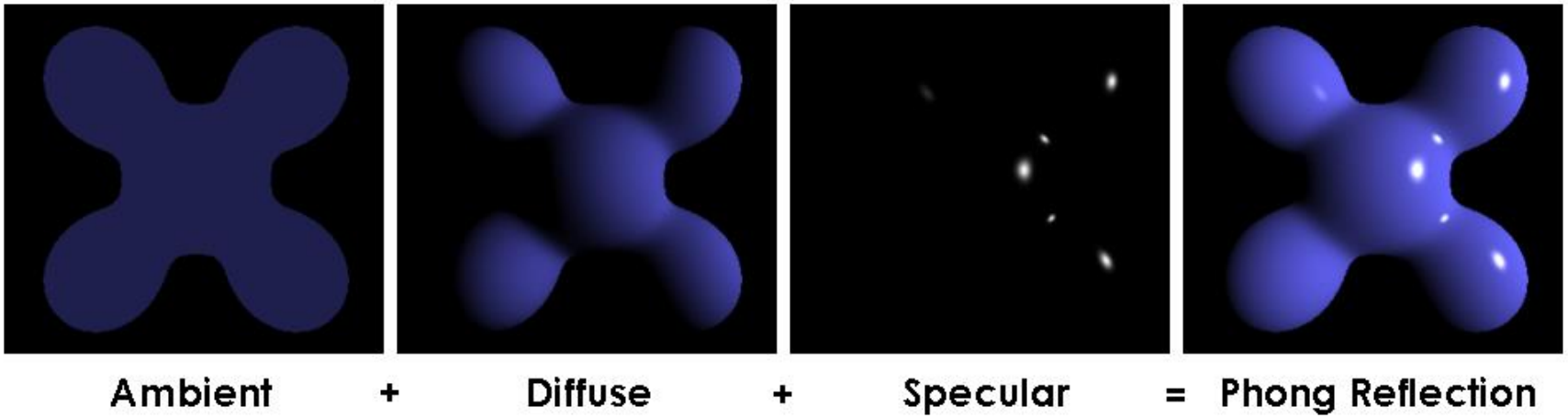
$$I = I_a k_a + k_d \sum_j I_{in,j} \, \left(l_j \cdot N\right) + k_s \sum_j I_{in,j} \, (ra)^S$$

- In der Regel wird die Berechnung auf drei Farbanteile aufgeteilt:

$$I_{\mathbf{r}} = I_{a,\mathbf{r}} k_{a,\mathbf{r}} + I_{in,\mathbf{r}} \left(k_{d,\mathbf{r}} \, (l \cdot N) + k_s (ra)^S\right)$$
$$I_{\mathbf{g}} = I_{a,\mathbf{g}} k_{a,\mathbf{g}} + I_{in,\mathbf{g}} (k_{d,\mathbf{g}} \, (l \cdot N) + k_s (ra)^S)$$
$$I_{\mathbf{b}} = I_{a,\mathbf{b}} k_{a,\mathbf{b}} + I_{in,\mathbf{b}} (k_{d,\mathbf{b}} \, (l \cdot N) + k_s (ra)^S)$$

Phong Reflexionsmodell

- Komponentendarstellung:



https://en.wikipedia.org/wiki/Phong_shading

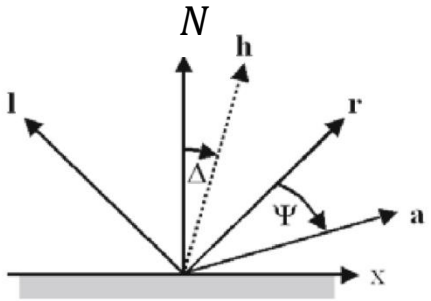
Blinn-Reflexionsmodell

- Motivation: Aufwand der Berechnung von r bei Phong-Reflexionsmodells recht hoch:

$$r = 2N\cos\theta - l = 2(l \cdot N) \cdot N - l$$

- Blinn: Statt r zu berechnen ermittelt man den Vektor h:
  $h = \frac{l+a}{|l+a|}$ verhält sich proportional zum r-Vektor.
  - Der *Halfwayvektor* h entspricht der Normalen zu einer hypothetischen Oberfläche eines perfekt spiegelnden Körpers
  - Der spiegelnde Anteil wird also wie folgt berechnet:

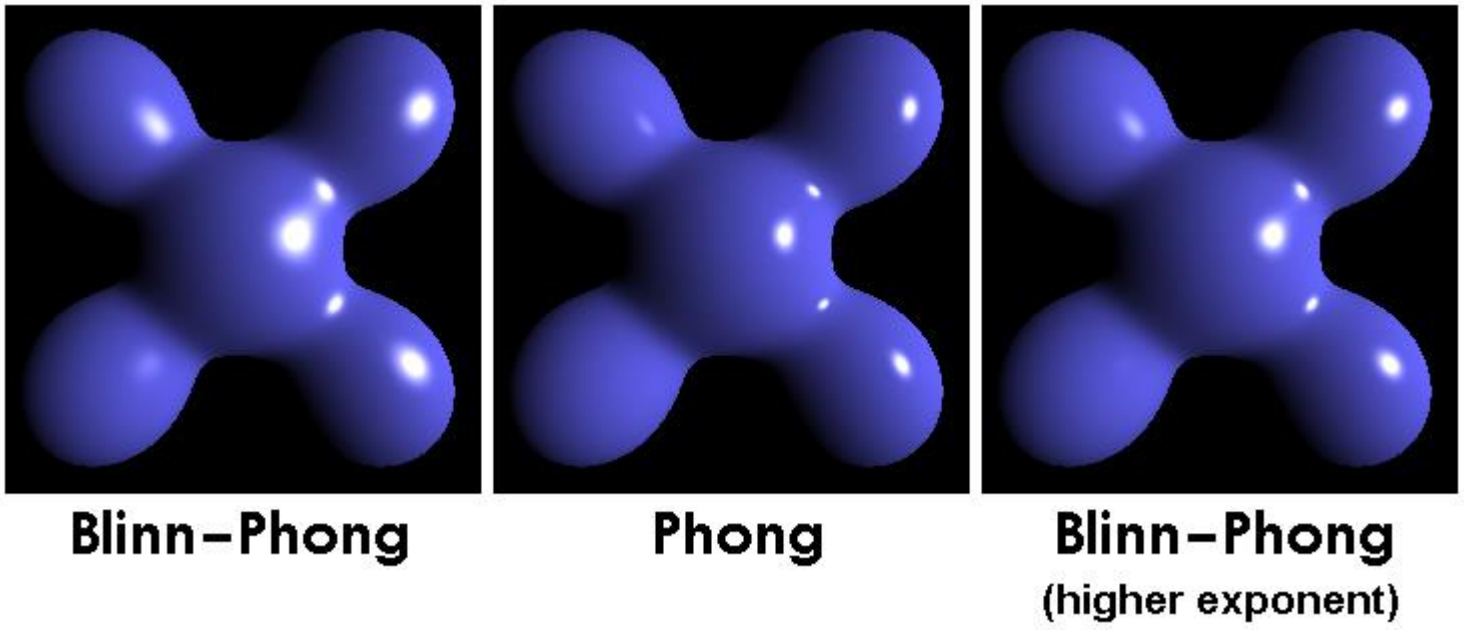
$$I_s = I_{in} \cdot k_s \cdot (h \cdot N)^S$$



A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Blinn-Reflexionsmodell

- Betrag des Winkels zwischen N und h ist nur halb so groß wie zwischen r und a
  - Lösung: um gleiches Ergebnis zu erzielen Exponent S erhöhen.



https://de.wikipedia.org/wiki/Blinn-Beleuchtungsmodell

Abschwächung des Lichts durch Abstände

- Einfache Verbesserung: Modellierung der Streuung des Lichts in Abhängigkeit von der Entfernung c des Objektes zum Betrachter:

$$I = I_a k_a + \frac{I_{in}(k_d\ (l \cdot N) + k_s(ra)^S)}{c + \varepsilon}$$

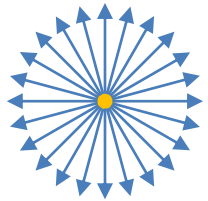
- Zum Abstand wird eine konstante ε addiert um eine Division durch 0 zu vermeiden.
- In diesem Beispiel linearer Abfall, Alternative: Modellierung eines exponentiellen Abfalls

7.3. LICHTQUELLEN

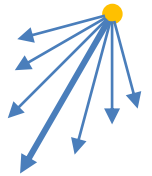
Arten von Lichtquellen

- Headlight
  - Lichtquelle, die direkt aus der Betrachterposition eine Szene beleuchtet, d.h. $l = a$

- Punktquellen (Standard bei lokalen Beleuchtungsmod.)
  - Punktförmige Lichtquelle an Position P
  - Radialsymmetrisch gleichmäßige Abstrahlung
  - Parallele Strahlen bei unendlich entfernter Lichtquelle

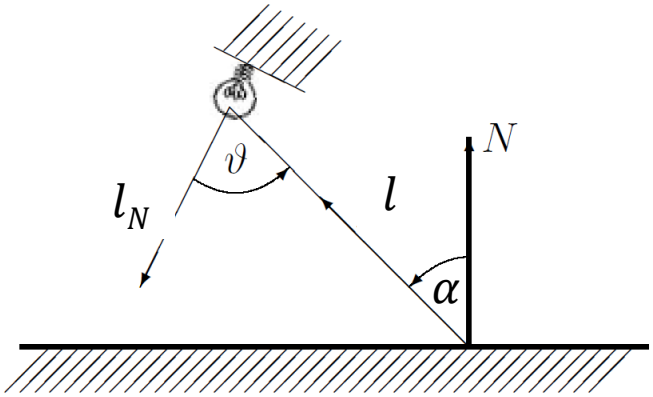


- Gerichtete Lichtquellen (Spotlights)
  - Strahler an Position P
  - Abstrahlintensität abhängig vom Abstrahlwinkel



Gerichtete Lichtquellen im Beleuchtungsmodell

- Erweiterung des Phong-Reflexions-Modells:
  - Quellintensität I_{in} hängt von Winkel ν zwischen den Vektoren $-l$ und l_N ab.



  - Gewichtung der Intensität durch $\cos(\nu) = l_N \cdot (-l)$
  - Fokussierung mit Exponent t

$$I = I_a k_a + I_{in}(l_N \cdot (-l))^t (k_d\,(l \cdot N) + k_s(ra)^S)$$

S. Krömker: Computergrafik I, Universität Heidelberg

Beleuchtung in OpenGL

- Aktivierung eines Lichtmodells: **`glEnable(GL_LIGHTING)`**
  - Setzt Befehl **`glColor`** außer Kraft!
  - Voreingestellt ist ein graues Ambiente-Licht!

- Einstellungen am Reflexionsmodells: **`glLightModelf(name, param)`**
  - **`Z.B. GL_LIGHT_MODEL_TWO_SIDE:`** Einseitig oder zweiseitige Beleuchtungsberechnung

- Spätestens beim Einschalten des Lichts müssen normierte Normalen der Oberflächen bekannt sein: **`glNormal3f(…)`**
  - Das muss der Programmierer selbst übernehmen!

Beleuchtung in OpenGL

- Festlegung von Lichtquelleneigenschaften: **`glLightf(light, name, param)`**
  - **`GLenum light`** ist eine Durchnummerierung der Lichtquellen von **`GL_LIGHT0`** bis **`GL_LIGHT7`**
  - **`Name`** und **`parameter`** legen Eigenschaften und deren Parameter fest:
    - **`GL_AMBIENT:`** RGBA-Wert für ambienten Anteil der Lichtquelle
    - **`GL_DIFFUSE:`** RGBA-Wert für diffusen Anteil der Lichtquelle
    - **`GL_SPECULAR:`** RGBA-Wert für den spekularen Anteil der Lichtquelle
    - **`GL_POSITION:`** (x,y,z,w)-Position der Lichtquelle. w=0: Lichtquelle im Unendlichen
- Einschalten einer Lichtquelle: **`glEnable(GL_LIGHT0)`**.
- Spotlights werden ebenfalls über **`glLightf(…)`** spezifiziert.

Beleuchtung in OpenGL

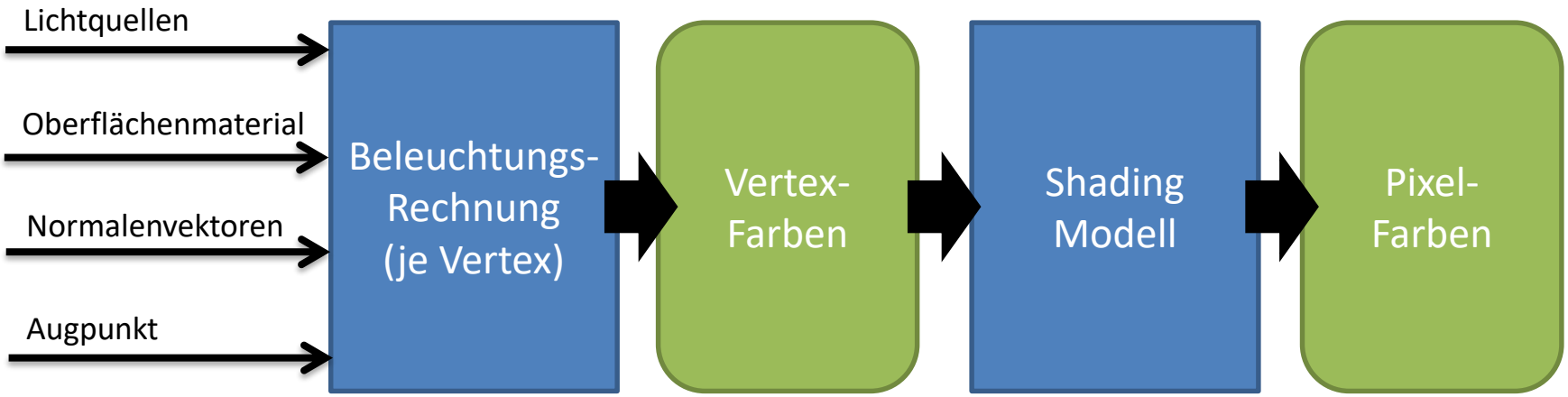
- Festlegung von Oberflächeneigenschaften: **`glMaterialf(face,name,param)`**
  - **`Face`** legt fest welche Seite die Eigenschaften zugewiesen werden (front, back)
  - **`Name`** und **`parameter`** legen Eigenschaften und deren Parameter fest:
    - **`GL_AMBIENT:`** RGBA-Wert für ambiente Reflexion
    - **`GL_DIFFUSE:`** RGBA-Wert für diffusen Reflexion
    - **`GL_AMBIENT_AND_DIFFUSE :`** gleiche RGBA-Werte für *ambiente* und *diffus*
    - **`GL_SPECULAR:`** RGBA-Wert für den spekularen Anteil der Reflexion
    - **`GL_SHININESS:`** Spiegelungsexponent S
    - **`GL_EMISSION:`** emmisive Farbe des Materials

- Zweite Möglichkeit: Color Material Mode **`glEnable(GL_COLOR_MATERIAL)`**
  - **`glColorMaterial(face, mode)`** legt fest welche Seite der Oberfläche bezüglich welcher Materialeigenschaft durch **`glColor`**-Aufrufe geändert wird.

7.4. SCHATTIERUNG

Schattierungsverfahren

- Reflexions- bzw. Lichtmodelle beschreiben die Intensität an jedem beliebigen Punkt im Raum
- Bei Berechnung für jedes Pixel (Millionen!) sehr aufwendig
- Abhilfe:
  - ➔ Auswertung nur an Vertices, Färben der Pixel anhand schnellerer Schattierungsverfahren („Shading“)
  - ➔ Ausnahme: Phong-Shading, *dazu später mehr*



Flat Shading

- Auswertung des Beleuchtungsmodells an einem Vertex eines Polygons
- Alle Pixel des Polygons erhalten diesen gleichen Farbwert
- Als Normalenvektor wird die Flächennormale des Polygons verwendet:

$$\begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} = \begin{pmatrix} v_{1,x} - v_{2,x} \\ v_{1,y} - v_{2,y} \\ v_{1,z} - v_{2,z} \end{pmatrix} \times \begin{pmatrix} v_{2,x} - v_{3,x} \\ v_{2,y} - v_{3,y} \\ v_{2,z} - v_{3,z} \end{pmatrix} = \begin{pmatrix} (v_{1,y} - v_{2,y})(v_{2,z} - v_{3,z}) - (v_{1,z} - v_{2,z})(v_{2,y} - v_{3,y}) \\ (v_{1,z} - v_{2,z})(v_{2,x} - v_{3,x}) - (v_{1,x} - v_{2,x})(v_{2,z} - v_{3,z}) \\ (v_{1,x} - v_{2,x})(v_{2,y} - v_{3,y}) - (v_{1,y} - v_{2,y})(v_{2,x} - v_{3,x}) \end{pmatrix}$$

  - In OpenGL Definition beim Anlegen des Polygons:

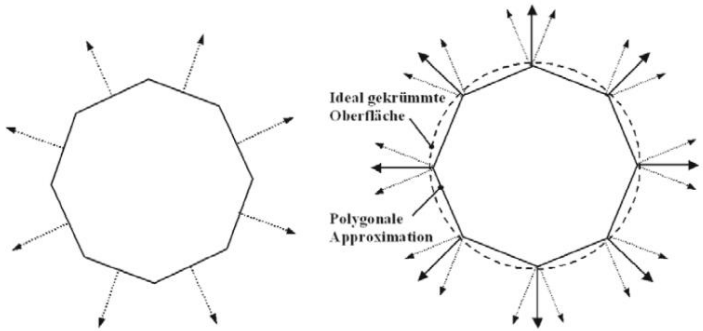
```
glBegin(GL_POLYGON)
   glNormal3f(0.0, 0.0, 1.0);
   glVertex3f(0.0, 0.0, 1.0);
   glVertex3f(0.0, 1.0, 1.0);
   glVertex3f(1.0, 0.0, 1.0);
glEnd();
```

- Aktivierung des Flat-Shadings in OpenGL: **`glShadeModel(GL_FLAT);`**
- Facettenartiges aussehen: Sprünge zwischen Polygonen
  - Abhilfe: kleinere Polygone bis Pixelgröße
- Sehr schnell, facettenartige Ansicht manchmal gewünscht (z.B. CAD)

Gouraud Shading

- Ziel: stetiger Farbverlauf durch lineare Interpolation der Pixelfarbe

- Ablauf der Berechnung:
  1. Berechnung der Flächennormalen des Polygons.
  2. An jedem Vertex Mittelung der Normalen angrenzender Flächen → **Vertex-Normalenvektor**



  3. Beleuchtungsrechnung für jeden Vertex → Vertexfarbe
  4. Farbwerte der Pixel einer Fläche durch lineare Interpolation
     - Zunächst entlang der Kanten
     - Dann zwischen den Kanten entlang der Scanline

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Gouraud Shading

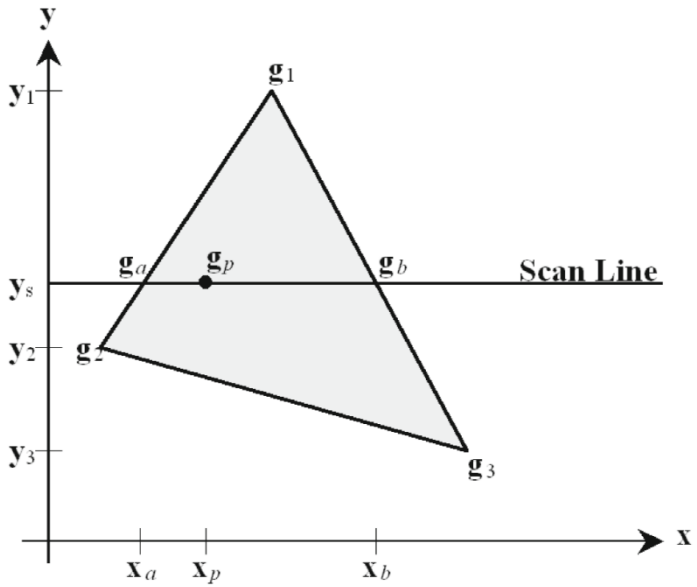
- Interpolation der Intensität: Scan-Line-Algorithmus

$$g_a = g_1 - (g_1 - g_2) \cdot \frac{y_1 - y_s}{y_1 - y_2}$$

$$g_b = g_1 - (g_1 - g_3) \cdot \frac{y_1 - y_s}{y_1 - y_3}$$

$$g_p = g_b - (g_b - g_a) \cdot \frac{x_b - x_p}{x_b - x_a}$$

Rückführung auf konstante Inkremente möglich



- Aktivierung in OpenGL: **`glShadeModel(GL_SMOOTH);`**
- Wichtig: Jedem Vertex muss eine Normale zugeordnet werden:

```
glBegin(GL_POLYGON)
   glNormal3f(0.5, 0.5, 0.707);
   glVertex3f(1.0, 1.0, 0.0);
   glNormal3f(-0.5, 0.5, 0.707);
   glVertex3f(0.2, 0.5, 0.0);
   :
glEnd();
```

A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

Gouraud Shading

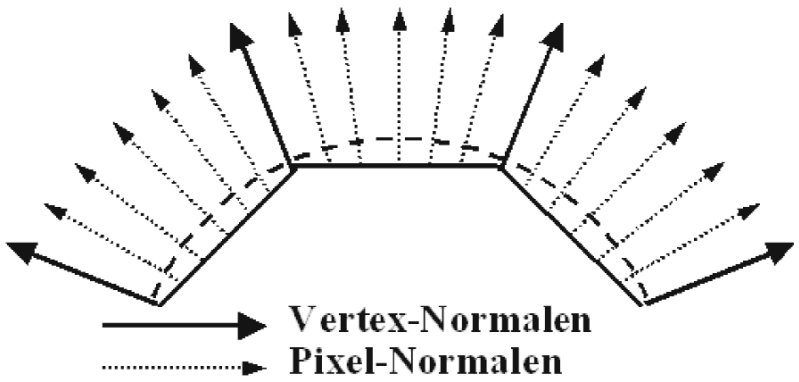
- Bewertung:
  - Farb- und Helligkeitssprünge werden eliminiert
  - Mittlere Polygonanzahl reicht bereits aus für akzeptable Bildqualität
  - Glanzlichter werden nicht korrekt dargestellt
  - Bei Bewegung: Aufblitzen von Glanzlichtern
  - Silhouette des Objektes bleibt so eckig wie das polygonale Modell
  - Probleme bei Spotlights und lokalen Lichtquellen: Ausfransen der Ränder

- Vertex-Normale nur bis zu einem gewissen Unterschied der Flächennormalen sinnvoll → Limitierung durch Grenzwinkel

- Guter Trade-off zwischen Qualität und Geschwindigkeit

Phong Shading

- Interpolation der Normalen (Phong) statt der Intensitäten (Gouraud) pro Pixel



- Ablauf der Berechnung:
  1. Berechnung der Flächennormalenvektoren und Vertexnormalenvektoren wie beim Gouraud Shading
  2. Interpolation der Pixel-Normalenvektoren: Scanline-Algorithmus.
  3. Beleuchtungsberechnung für jedes Pixel → Farbwert pro Pixel

Phong Shading

- Interpolation der Normalen: Scan-Line-Algorithmus

1.) $N_a = N_1 - (N_1 - N_2) \cdot \frac{y_1 - y_s}{y_1 - y_2}$

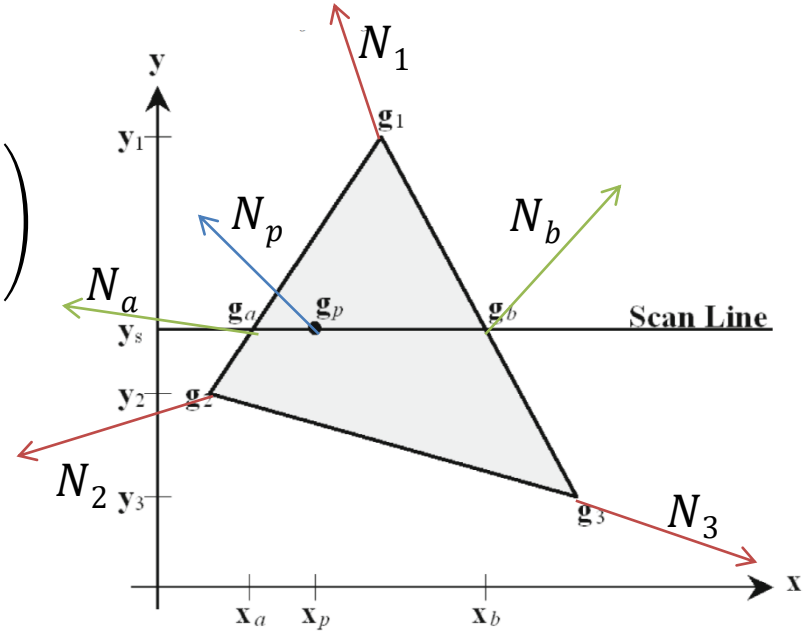
2.) Normalisierung:

$$N_{a'} = \frac{N_a}{|N_a|} = \frac{1}{\sqrt{{N_{a,x}}^2 + {N_{a,y}}^2 + {N_{a,z}}^2}} \cdot \begin{pmatrix} N_{a,x} \\ N_{a,y} \\ N_{a,z} \end{pmatrix}$$

3.) $N_b = N_1 - (N_1 - N_3) \cdot \frac{y_1 - y_s}{y_1 - y_3}$

4.) Normalisierung → $N_{b'}$

5.) $N_p = N_{b'} - (N_{b'} - N_{a'}) \cdot \frac{x_b - x_p}{x_b - x_a}$



Rückführung auf konstante Inkremente möglich!

Phong Shading

- Bewertung:
  - Beseitigt gravierende Probleme des Gouraud-Shadings (Glanzlichter, Ränder)
  - Silhouette des Objektes bleibt weiterhin so eckig wie das polygonale Modell
  - Grundlage für weitere Verfahren wie z.B. Bump Mapping

- Im Compatibility-Profile von OpenGL nicht wählbar

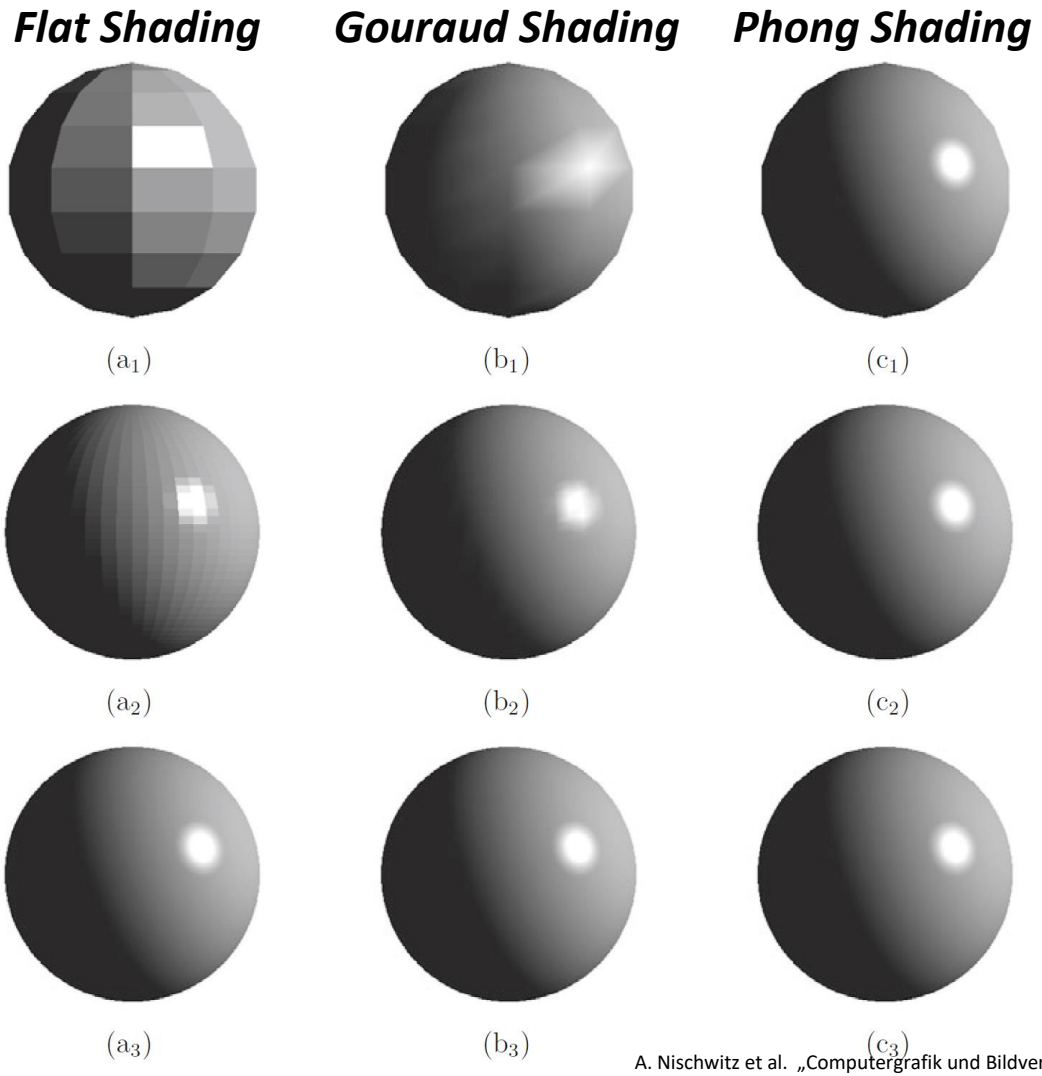
  → eigene Implementierung als Shader notwendig

Vergleich Flat/Gouraud/Phong Shading

- Vergleich der Schlüsselkomponenten

	Flat-Shading	Gouraud-Shading	Phong-Shading
Verwendete Normale	Flächennormale	Vertexnormale	Interpolierte Normale
Beleuchtungsrechnung	An einem Vertex des Polygons	An allen Vertices des Polygons	An jedem gerasterten Pixel
Farbe des Pixels	Alle Pixel im Polygon gleiche Farbe	Lineare Interpolation anhand Farbe der Vertices des Polygons	Farbe durch Beleuchtungsrechnung an jedem Pixel

Vergleich Flat/Gouraud/Phong Shading

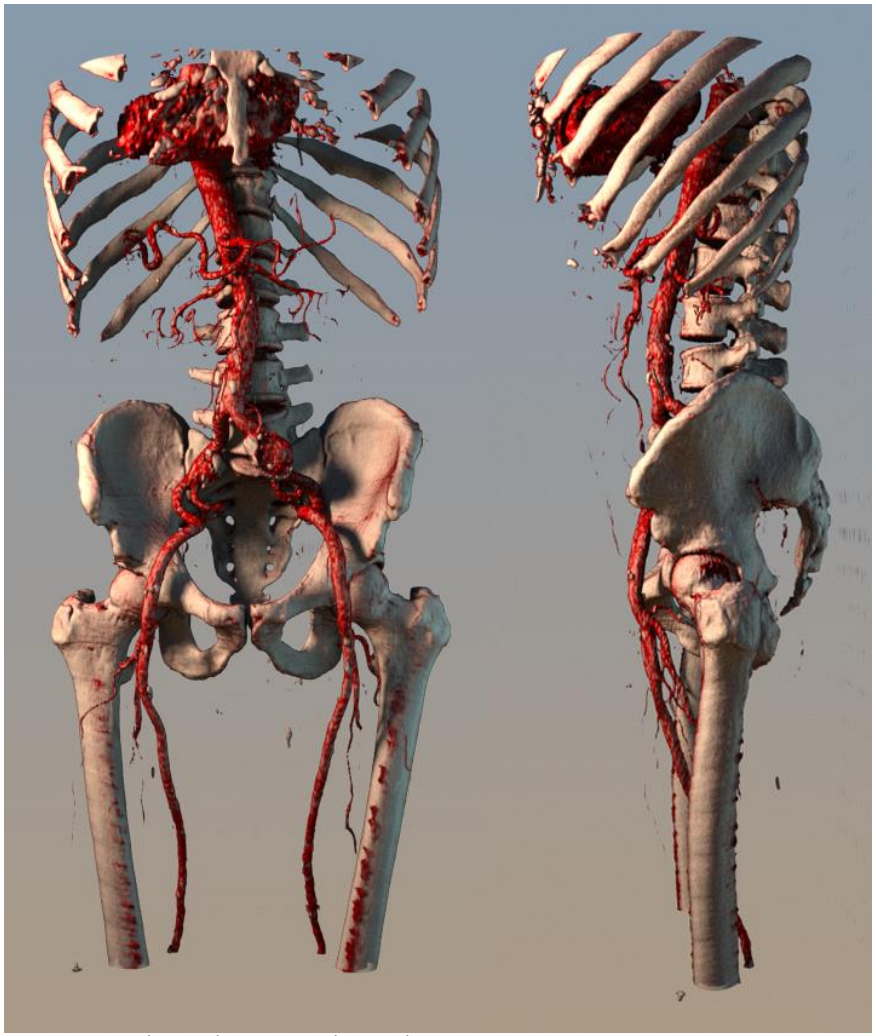


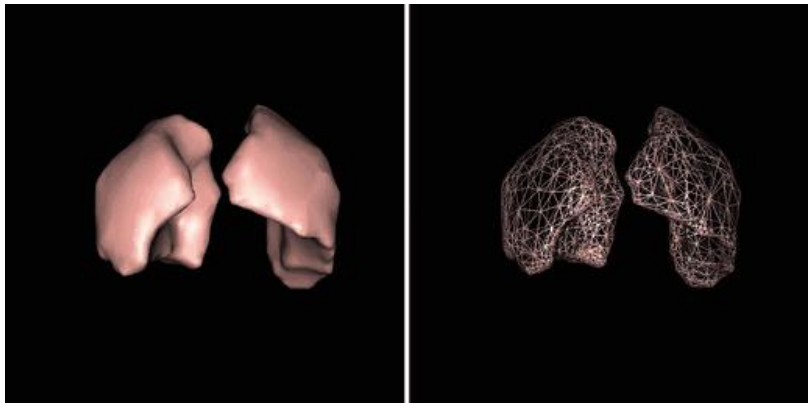
A. Nischwitz et al. „Computergrafik und Bildverarbeitung“, Band 1: Computergrafik“

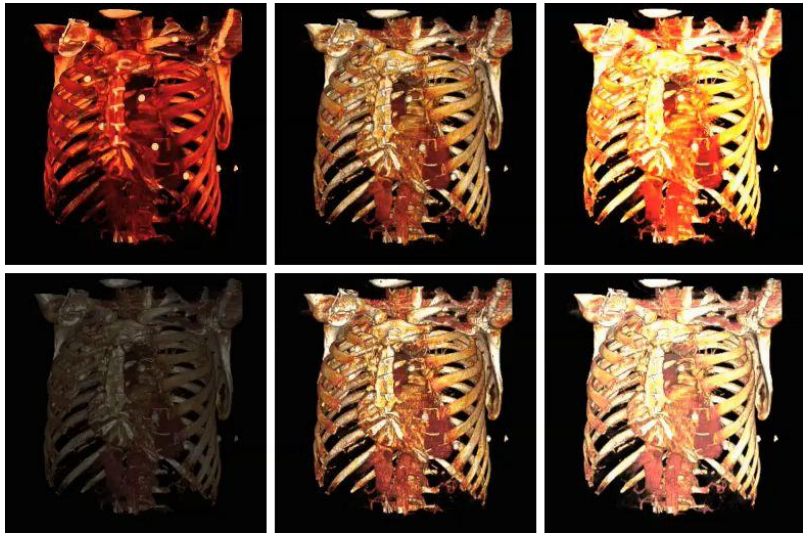
Vergleich Flat/Gouraud/Phong Shading

- Schattierungsverfahren und Polygonauflösung (Tesselierung) sind austauschbar im Hinblick auf Bildqualität
  - Je einfacher das Schattierungsverfahren desto feiner muss das Polygonnetz sein.
- Bei gleicher Bildqualität: was ist der effizienteste Weg?
  - Flat Shading vs. Phong Shading:
    - Flat Shading mit vielen Polygonen langsamer als Phong Shading mit wenigen Polygonen
  - Flat Shading vs. Gouraud Shading
    - Gouraud-Shading benötigt 10-100 mal weniger Polygone als Flat Shading
    - Gouraud-Shading wertet an 3 Vertices die Beleuchtungsfunktion aus, Flat Shading nur 1 mal
    - In der Summe: Gouraud Shading ca. 3-30 mal effizienter als Flat Shading
  - Gouraud Shading vs. Phong Shading
    - Phong benötigt ca. Faktor 10 weniger Polygone, Limit durch sichtbare Silhouette
    - Auswertung pro Pixel deutliche aufwändiger
- Gouraud Shading und Blinn-Reflexionsmodell oft der Standard in der interaktiven Computergrafik

Beispiele aus der medizinischen Visualisierung

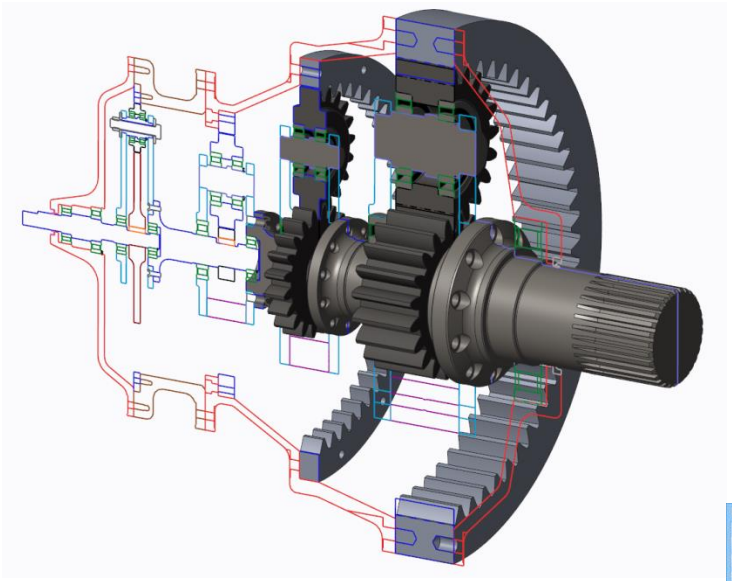


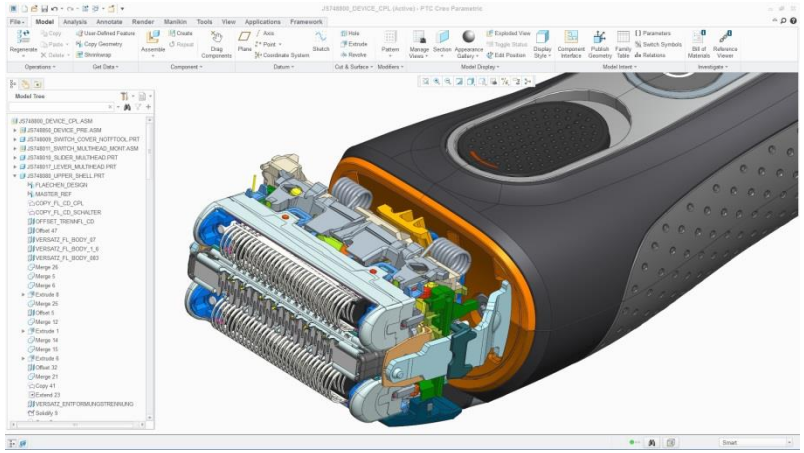


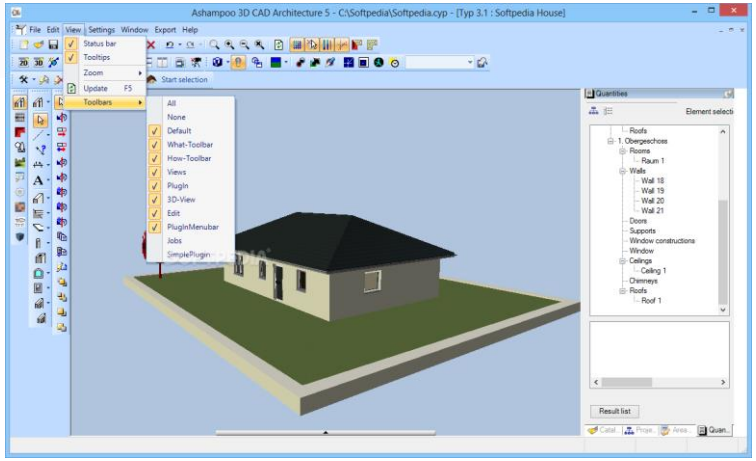


https://plus.google.com/photos/+ChadHaney/albums/5936877961446459345
https://en.wikibooks.org/wiki/Basic_Physics_of_Nuclear_Medicine/Three-Dimensional_Visualization_Techniques

Beispiele aus der CAD-Visualisierung







http://web-designbristol.com/wp-content/uploads/2016/03/computer-aided-desig-and-drafting.png
http://www.softpedia.com/get/Science-CAD/Ashampoo-3D-CAD-Architecture.shtml
http://www.heise.de/ct/ausgabe/2014-18-aktuell-Technische-Anwendungen-2284326.html

ZUSAMMENFASSUNG

Zusammenfassung

- Licht ist Ursache der Farbwahrnehmung
  - Farbmodelle in der Computergrafik zur Beschreibung von Farben: RGB, CMY(K), HSV, LAB
- Komplexe Interaktion von Licht mit Materie:
  - Spiegelnde und diffuse Reflexion, Absorption, Brechung, Dispersion, Fresnel Effekt
  - Vereinfachungen für Computergrafik notwendig
- Lokale vs. Globale Beleuchtung
- Reflexionsmodelle in der interaktiven Computergrafik
  - Phong Reflexionsmodell: ambiente, diffuse, spekulare Intensitätskomponenten
  - Blinn-Phong Reflexionsmodell: Vereinfachte Berechnung gegenüber Phong
- Lichtquellen
  - Gerichtet, Punktförmig,
- Schattierungsverfahrung: setzen Vertex-bezogene Beleuchtungsberechnung für alle Pixel eines Objektes um
  - Flat Shading, Gouraud Shading, Phong Shading

Übungsfragen Kapitel 7

- Was ist der Fresnel-Effekt?
- Wie entsteht die Farbwahrnehmung eines Objekts bei Bestrahlung mit weißem Licht?
- Aus welchen drei Komponenten setzt sich das Phong-Reflexionsmodell zusammen und von welchen Parametern ist es abhängig?
- Worin liegt der Unterschied zwischen Phong- und Blinn-Reflexionsmodell?
- Beschreiben Sie die Unterschiede der Schattierungsverfahren Flat, Gouraud und Phong Shading. Welches Verfahren ist bei gleich bleibender Bildqualität das effizienteste Verfahren?