

Zusammenfassung zu Shading-Modellen

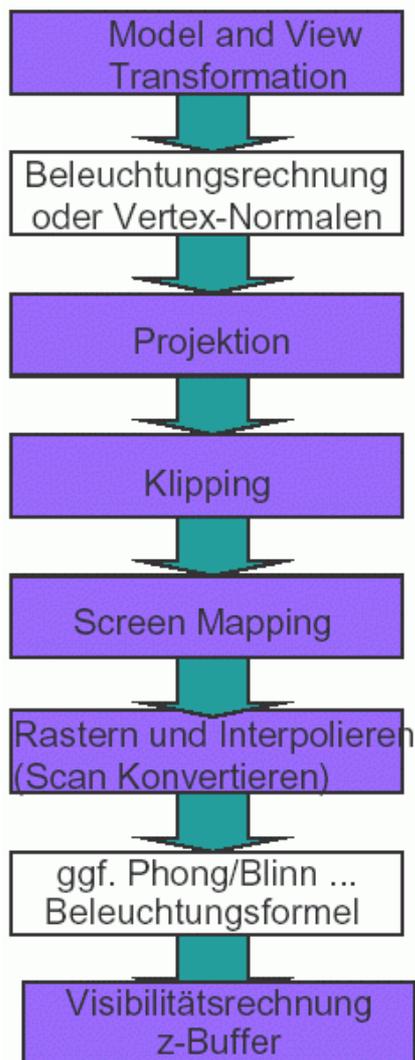
Die Schattierung (*Shading*) umfasst sowohl die Auswahl eines geeigneten *Beleuchtungsmodells* als auch die Entscheidung, in welchen Bildpunkten das jeweilige Modell explizit zur Anwendung gelangen soll und in welchen Punkten *Interpolationsmethoden* (billiger) eingesetzt werden. Letztere sind erforderlich, da die explizite Berechnung des aus dem Beleuchtungsmodell resultierenden Intensitäts- bzw. Farbwertes für jedes Pixel insbes. bei interaktiven Echtzeitanwendungen zu aufwändig wäre.

- Varianten des Phong-Shading sind die "Arbeitspferde" in der hochwertigen Computergrafik
- heute häufig in Hardware realisiert!

nicht verwechseln:

- *Shading*: verbunden mit geometrischer Glättung (Interpolation)
 - Gouraud: Farbwerte, Phong: Normalen
- Reflektion, Beleuchtungsgleichung: Berechnung der Farbintensitäten
 - Gouraud: i.allg. nur diffuser + ambienter Anteil;
 - Phong: spiegelnder + diffuser + ambienter Anteil.

Integration in die Rendering-Pipeline:



Historisch und praktisch werden zusammen verwendet:

- Gouraud-Shading + Lambertscher Reflektor (rein diffuse Abstrahlung) (auch Phong-Reflektor möglich, aber nicht im Bewegtbild)
- Phong-Shading + Phong-, Blinn-, Cook/Torrance-, ... -Reflektor

Transparenz

Manche Objekte sind (teilweise) durchsichtig

⇒ spezielle Probleme (Lichtbrechung, -abschwächung...)

3 einfache Ansätze ohne Berücksichtigung der Brechung:

1. *Screen-Door-Transparenz*

Das transparente Polygon wird mit einem Schachbrettmuster überzogen; ein Pixel wird voll transparent, das nächste wieder undurchsichtig (opak) gerendert.

Vorteile:

- einfach
- keine Tiefensortierung notwendig

Nachteile:

- Transparenzgrad 50 % (schwierig variierbar)
- Artefakte bei mehreren transparenten Objekten hintereinander

2. Interpolierte Transparenz (*Alpha-Blending*)

Beim Rasterisieren, wird jedem Pixel nicht nur ein Farb- und z-Wert, sondern auch ein α -Wert zugeordnet: $0 \leq \alpha \leq 1$.

$\alpha = 1$ entspricht vollständig opakem Objekt

$\alpha = 0$ vollständig transparentes Objekt

Jedes Pixel, das von einem transparenten Objekt bedeckt wird, wird als Farbe

$$\mathbf{c}_{new} = \alpha \cdot \mathbf{c}_{ren} + (1 - \alpha) \cdot \mathbf{c}_{old}$$

zugeordnet, wobei c_{ren} die Farbe des transparenten Objekts und c_{old} die Pixelfarbe vor dem Blending ist.

Das transparente Objekt wird *über* die schon gerenderten Objekte geschrieben.

Alpha-Blending erfordert Tiefensortierung der transparenten Objekte (back-to-front-Sortierung).

3. Gefilterte Transparenz

Das transparente Polygon wird als Filter aufgefasst, der Licht gewisser Wellenlängen bevorzugt durchlässt:

$$I_{P,\lambda} = I_1 + k_t(\lambda)I_2.$$

Der Transmissionskoeffizient $k_t(\lambda)$ hängt von der Wellenlänge ab. Damit lässt sich als Material von Objekt 1 etwa rotes Glas modellieren. Im Rotbereich ist $k_t(\lambda)$ dann sehr klein, im komplementären Cyanbereich dagegen groß.

Reduzierung der Lichtintensität beim Durchgang durch das transparente Objekt:

Transparenzkoeffizient (multiplikativ)

$$T = te^{-ad}$$

(wichtig für Raytracing).

Darin ist d die Länge des Strahls innerhalb des transparenten Objekts. t = "Durchlässigkeitsfaktor" = Anteil der Lichtmenge, der nicht reflektiert wird; a = "Absorptionsfaktor", gibt an, wie schnell das Licht von dem transparenten Material geschwächt wird.

Lichtintensität vor und nach dem Durchgang durch das Objekt:

$$I_{aus} = T \cdot I_{ein}.$$

Zusätzlich kann die *Lichtbrechung* nach dem Gesetz von Snellius berücksichtigt werden (s.o., Abschnitt über physikal. Grundlagen).

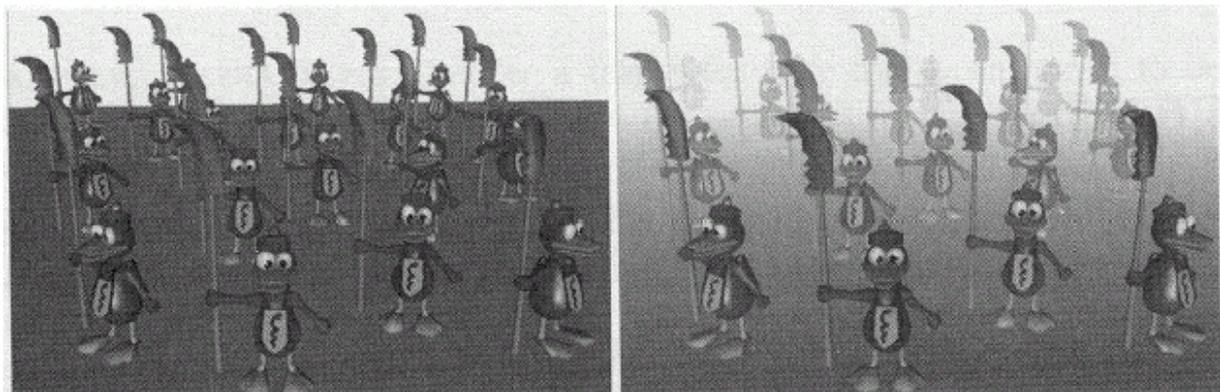
Nebel

Nebel (oder Dunst) kann relativ einfach simuliert werden, auch in Echtzeitsystemen

Er kann für viele Zwecke sinnvoll eingesetzt werden:

- Realismus von Landschaftsszenen (*outdoor scenes*) erhöhen
- Verbesserung der Tiefenschätzung / des 3D-Effekts
- Nebel verdeckt Artefakte in der Nähe der hinteren Clipping-Ebene (*far clipping plane*)

Nebel wird inzwischen oft von der Grafik-Hardware unterstützt.



(aus Krömker 2001)

Ähnlich wie bei der Transparenz, werden Nebeleffekte auch beim Rasterisieren realisiert, indem die Farbe c_{ren} eines Pixel mit der Farbe c_{fog} des Nebels gemischt wird:

$$\mathbf{c}_{new} = f\mathbf{c}_{ren} + (1-f)\mathbf{c}_{fog}$$

wobei f eine Funktion des Abstands vom Beobachter mit Werten in $[0, 1]$ ist.

Einfach gesagt: Tiefe im Bild ändert die Farbe von Pixeln.

Arten von Nebel:

- **Linearer Nebel:**

$$f(z) = \frac{z_{\max} - z}{z_{\max} - z_{\min}}$$

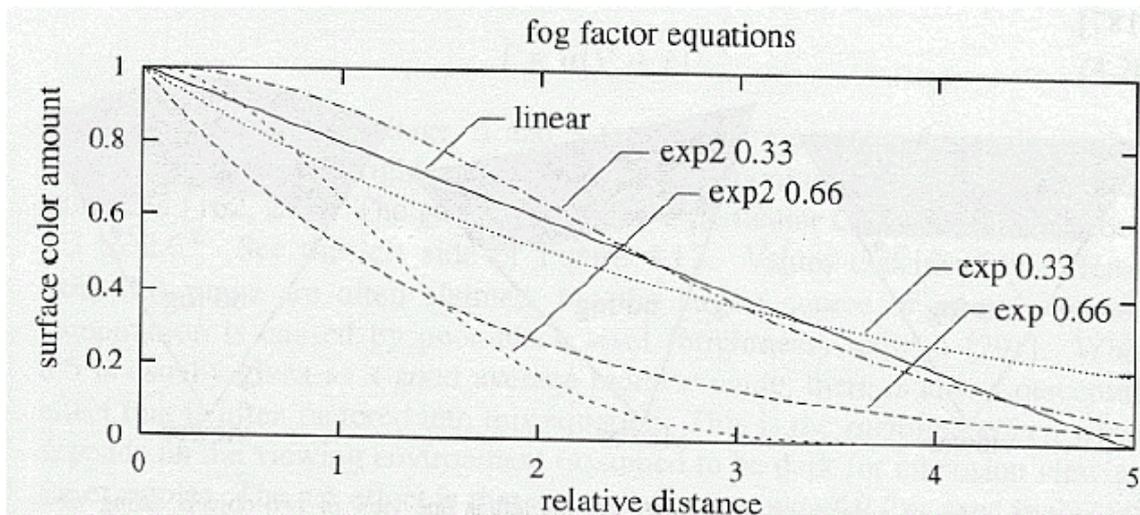
- **Exponentieller Nebel:**

wobei d_f die Dichte des Nebels bestimmt.

$$f(z) = e^{-d_f z}$$

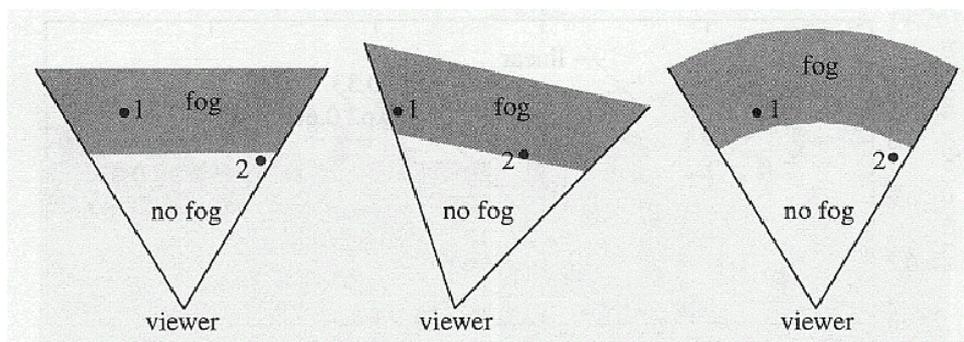
- **Quadrierter exponentieller Nebel:**

$$f(z) = e^{-(d_f z)^2}$$



Implementierungsprobleme beim Nebel:

- die z -Werte verhalten sich nach der perspektivischen Transformation (in den kanonischen Sichtkörper) nicht mehr linear \Rightarrow Nebel-effekt müsste eigentlich im Objektraum berechnet werden
- Häufig wird der Abstand nicht für jedes Pixel ermittelt, sondern es wird einfach der Abstand entlang der zentralen Sichtgeraden verwendet. Das führt zu Artefakten. *Radialer Nebel* (Euclidean Distance Fog) vermeidet dieses Problem.



- Nebel auf Pixel-Ebene wird zur Zeit von Grafik-Hardware i.allg. nicht unterstützt, stattdessen auf Vertex-Ebene, d.h. nur an den Polygon-Eckpunkten berechnet und dann interpoliert.

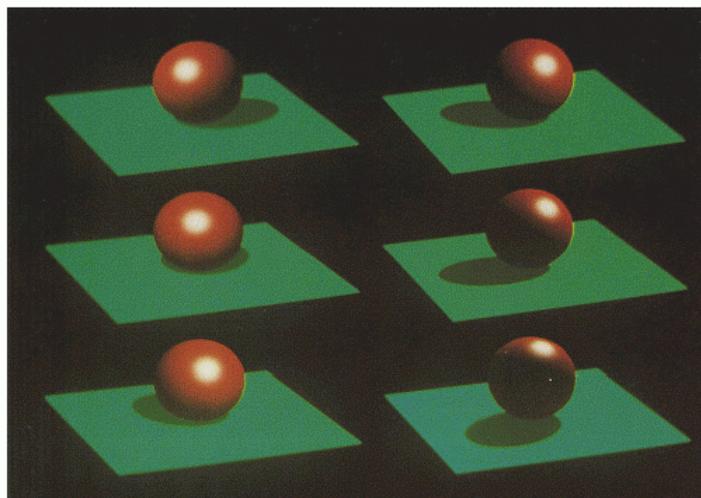
Schattenberechnung

hier geht es um Beschattung von Objekten durch andere
– zu unterscheiden von *shading* (Schattierung)!

wichtiges Element des Fotorealismus!

Funktionen des Schattens

- „verankert“ Objekte in der Szene (keine „fliegenden“ Objekte)
- hebt die Richtung der Beleuchtung hervor



- Steigerung des Realismus des Bildes
- Schatten geben Hinweise zum Aufbau der Szene
- Verbesserung der Tiefenwahrnehmung
- verwendbar für Simulationen (Energie-, Wärmeverteilung)

Kern- und Halbschatten

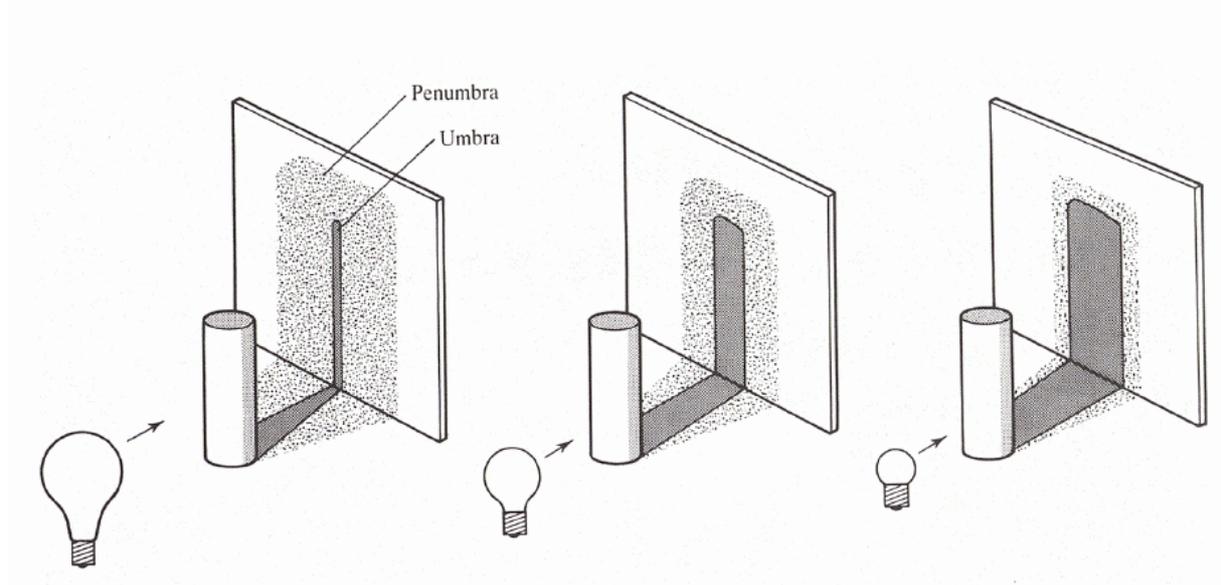
Schatten bestehen i.allg. aus 2 Teilen: Kernschatten (*umbra*) und Halbschatten (*penumbra*)

- Kernschatten: zentraler, scharf abgegrenzter dunkler Teil; hier kommt kein Licht der betrachteten Lichtquelle an, die Lichtquelle ist von hier aus nicht sichtbar.
- Halbschatten: helleres Gebiet, das den Kernschatten umgibt; von hier aus ist die Lichtquelle *partiell* sichtbar.

- Punktlichtquellen generieren nur Kernschatten
- flächige Lichtquellen erzeugen beides

Halbschatten aufwändig zu berechnen; da in der Computergrafik i.d. Regel Punktlichtquellen eingesetzt werden, *häufig nur Kernschattenberechnung*.

Kern- und Halbschatten bei unterschiedlich großen Lichtquellen



Position der Lichtquelle

- Komplexität der Schattenberechnung abhängig von Position der Lichtquelle
- keine Schatten, wenn (einzige) Lichtquelle im Betrachterstandpunkt
- Lichtquelle im Unendlichen: orthographische Projektion zur Bestimmung des Schattens → einfachster Fall
- Lichtquelle an endlicher Position außerhalb des Gesichtsfeldes: perspektivische Projektion zur Bestimmung des Schattens → etwas komplizierter
- Lichtquelle innerhalb des Gesichtsfeldes: Unterteilung der Szene in Sektoren, Schattenberechnung separat für die Sektoren → kompliziertester Fall
- Schatten ändern sich bei Animation

Schatten und Beleuchtung

- HSR/VSD-Algorithmen bestimmen, welche Flächen vom Betrachterstandpunkt aus sichtbar sind.
- Schattenberechnung bedeutet, die Flächen zu bestimmen, die von der Position der Lichtquelle aus nicht sichtbar sind.
- → VSD/HSR-Algorithmen können genutzt werden
- Sichtbarkeit aus Sicht der Lichtquelle: entweder sichtbar oder nicht
- wenn Oberflächenpunkt von Lichtquelle aus nicht sichtbar
 - Punkt liegt im Schatten
 - Beleuchtungsberechnung muß entsprechend angepaßt werden

- Faktor S_j bestimmt, ob Licht von Lichtquelle j ein Objekt an einem gegebenen Punkt erreicht:

$$S_j = \begin{cases} 0, & \text{wenn Lichtquelle } j \text{ blockiert ist} \\ 1, & \text{wenn Lichtquelle } j \text{ nicht blockiert ist} \end{cases}$$

- Hinzufügen zur Beleuchtungsberechnung:

$$I_\lambda = I_{a\lambda} k_a O_{d\lambda} + \sum_{j=1}^m S_j f_{\text{att}_j} I_{i\lambda_j} (k_d O_{d\lambda} (N \cdot L_j) + k_{s\lambda} O_{s\lambda} (R_j \cdot V)^n)$$

- Gebiete im Schatten immer noch durch das ambiente Licht beleuchtet

Übersicht: Verfahren der Schattenberechnung

- Schatten auf einer einzelnen Grundebene (Spezialfall)
- Algorithmen zur Schattenberechnung im allgemeinen Fall:
 - Scanlinien-Schatten-Berechnung
 - Schattenvolumen-Ansatz
 - Hidden Surface Removal (HSR) aus Sicht der Lichtquelle
 - Schatten-z-Buffer
 - Raytracing
 - Radiosity

Raytracing und Radiosity ergeben (weitgehend) exakte Ergebnisse, werden aber erst später behandelt