Beleuchtung

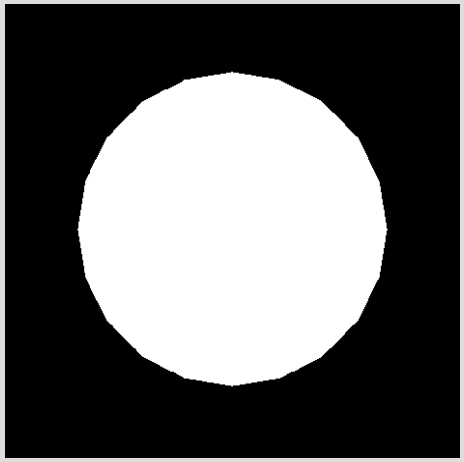
Matthias Nieuwenhuisen

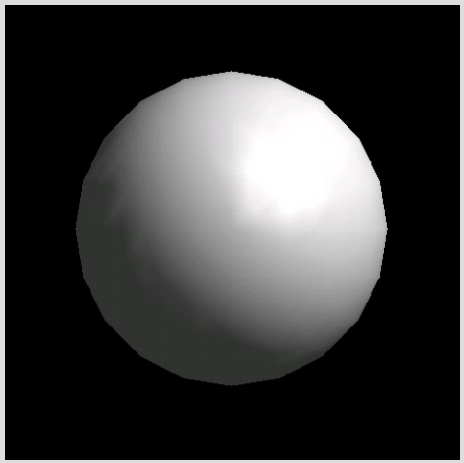
Überblick

- Warum Beleuchtung?
- Beleuchtungsmodelle
- Lichtquellen
- Material
- Reflexion
- Shading

Warum Beleuchtung?

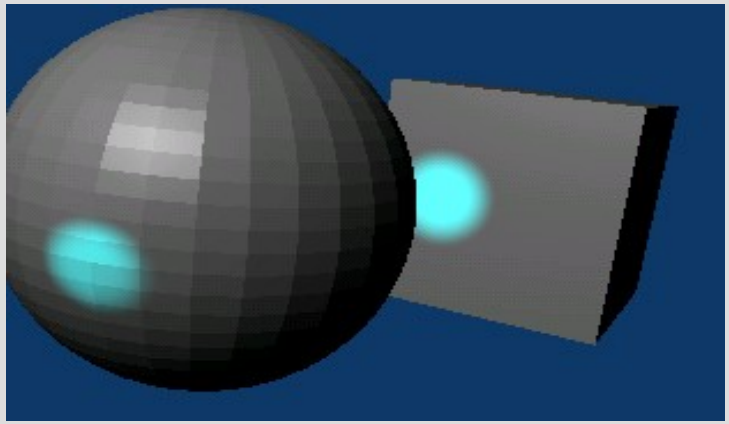
- Tiefeneindruck
- Realitätsnähe:
  - Reflexionen
  - Spiegelungen
  - Schatten
- Modelle:
  - global
  - lokal





Beleuchtungsmodelle

- Globale Modelle
  - komplexe Berechnung
  - Radiosity / Raytracing
- Lokale Modelle
  - einfache Berechnung
  - Hardwarebeschleunigung
  - Anwendung auf ein Objekt
  - kein Schattenwurf o.ä.

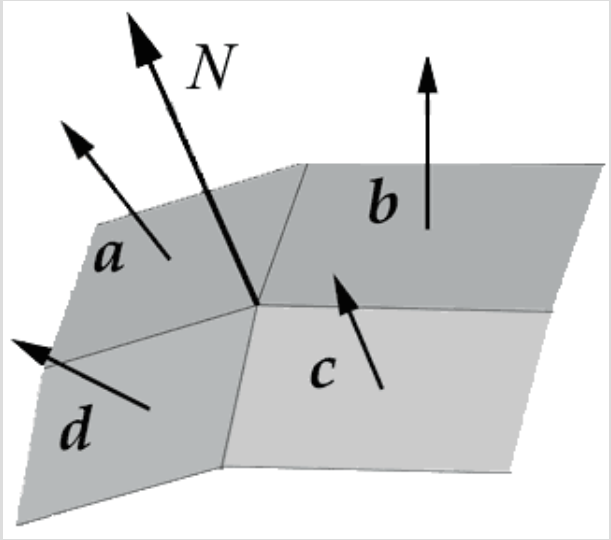


Lokales Beleuchtungsmodell

- Beleuchtungsbeschreibung einer Szene
  - Lichtquellen
  - Material
  - Shadingmodell
  - Position des Betrachters

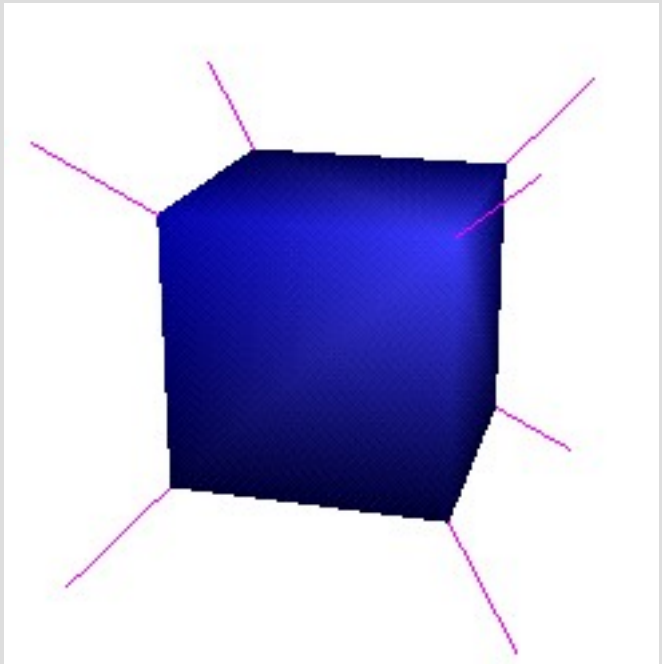
Normalen

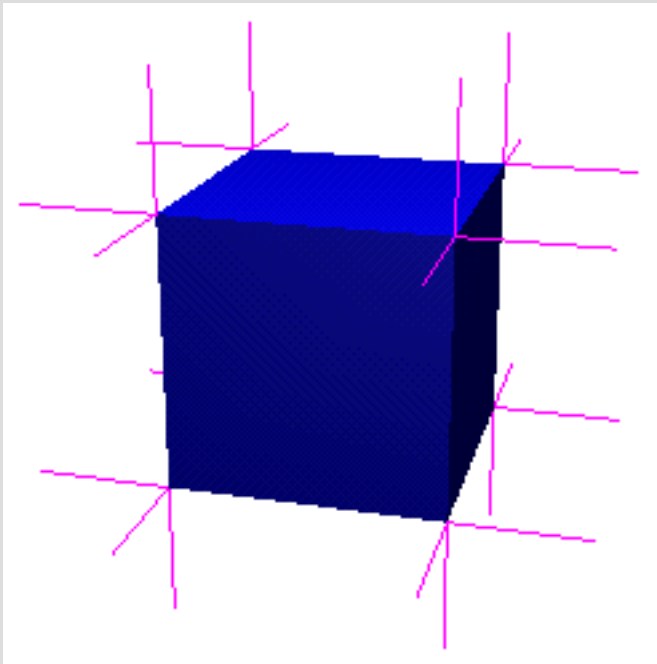
- Flächennormale
  - steht Senkrecht auf Fläche
- Vertexnormale
  - geht durch Vertex
  - Durchschnitt der Flächennormalen



Normalen

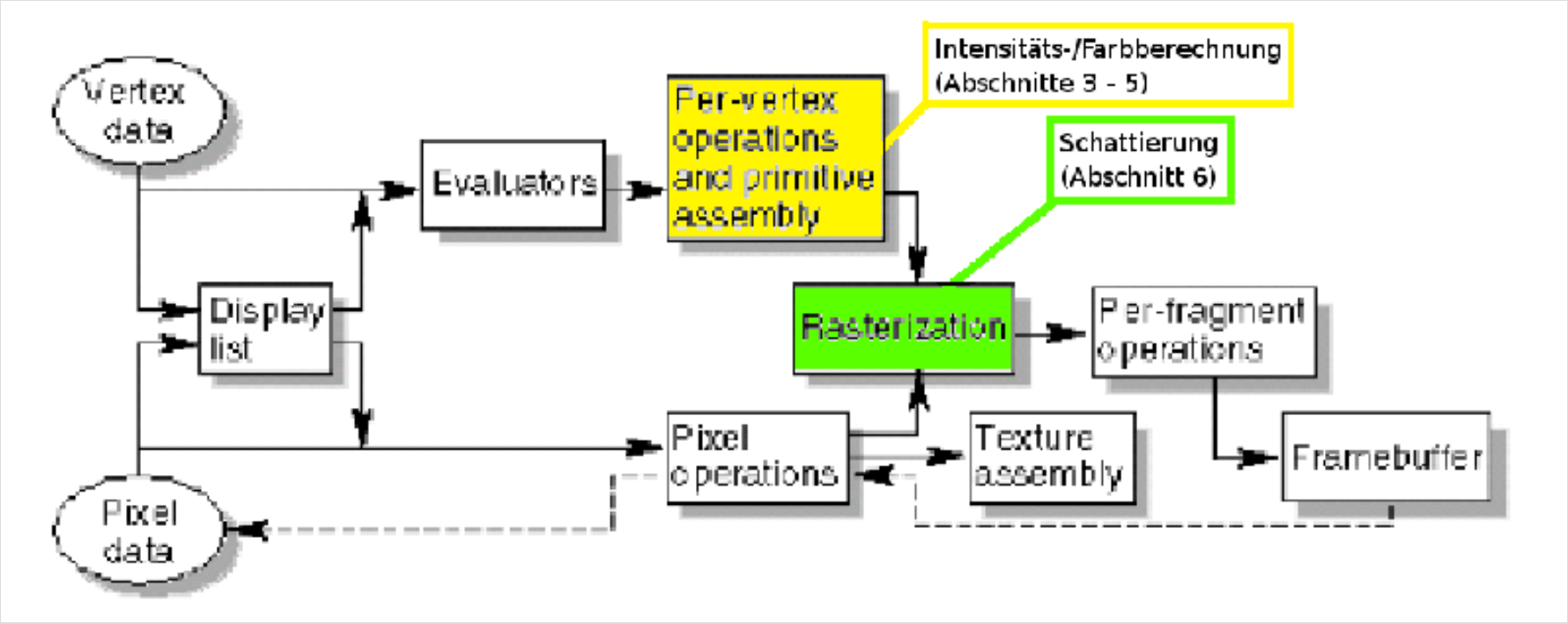
- Problem: keine scharfen Kanten
- Lösung: mehr Vertexnormalen





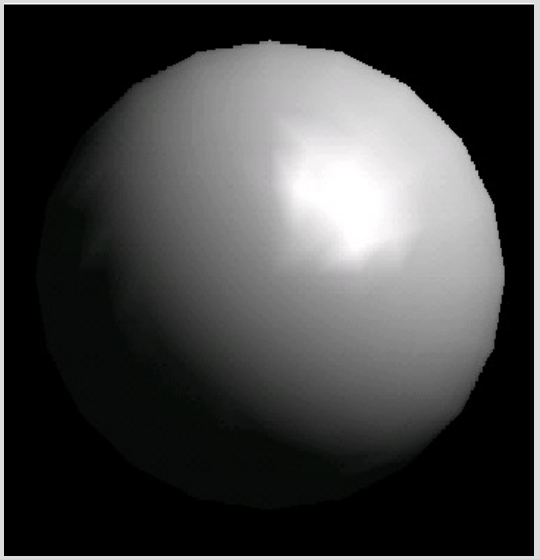
Rendering Pipeline

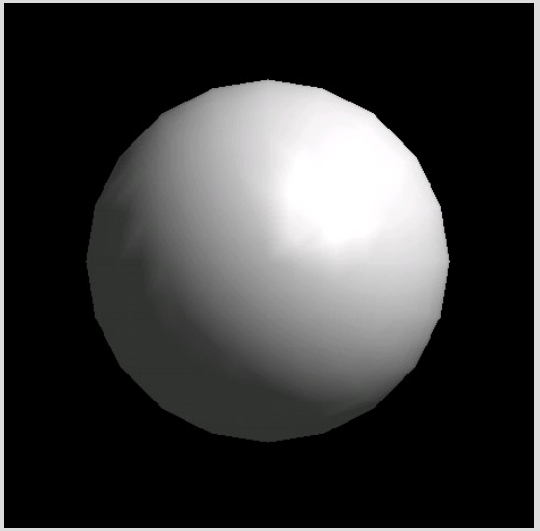
- Berechnung besteht aus zwei Schritten:
  - Intensitätsberechnung
  - Shading



Globales Licht

- kein Ursprung
- diffuse Reflexion
- aus allen Richtungen gleich stark
- entsteht durch häufige Reflexion
- GL_LIGHT_MODEL_AMBIENT



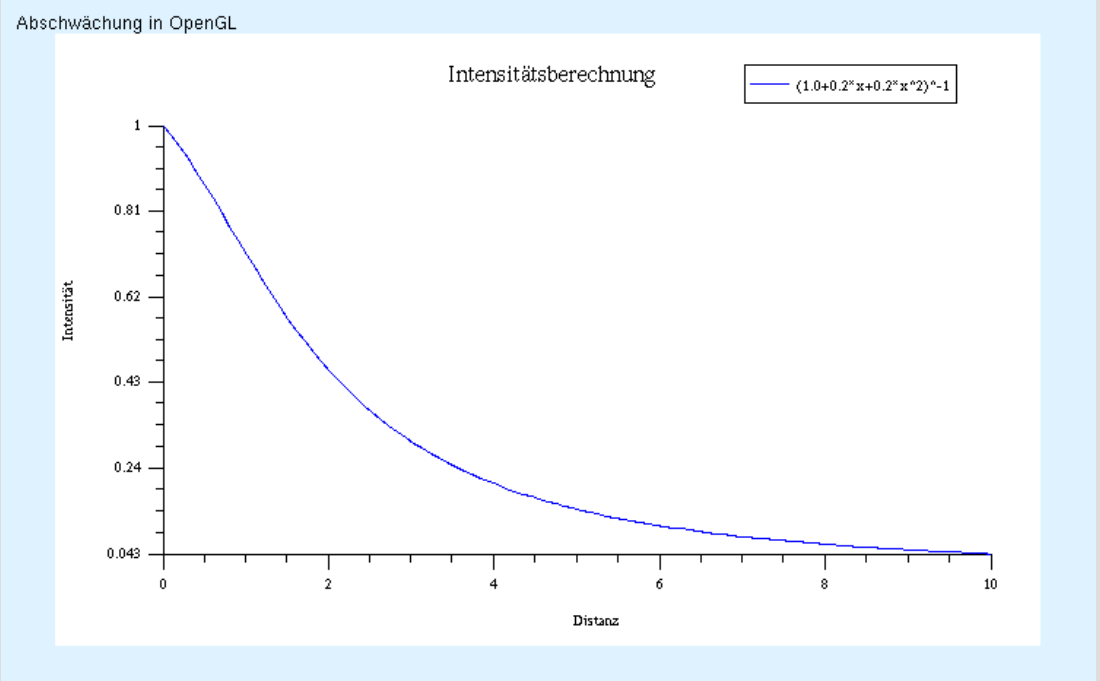


Lichtquellen

- bestrahlt Objekte von einer Seite
- Position
  - GL_POSITION
- ambiente, diffuse, spekulare Farbe
  - GL_AMBIENT
  - GL_DIFFUSE
  - GL_SPECULAR

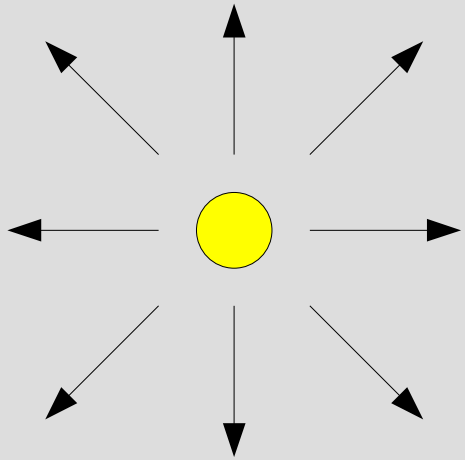
Abschwächung

- Licht wird schwächer auf Distanz
- $Intensität = (k_c + k_l \cdot d + k_q \cdot d^2)^{-1} \cdot Grundintensität$
- d = Distanz zur Lichtquelle
- k = Konstanten



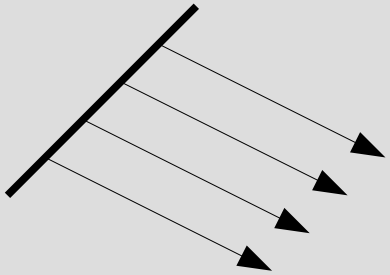
Punktlichtquelle

- sendet Licht in alle Richtungen aus
- z.B. Glühbirne
- bei großer Distanz selbe Wirkung wie Flächenlichtquelle (nächste Folie)



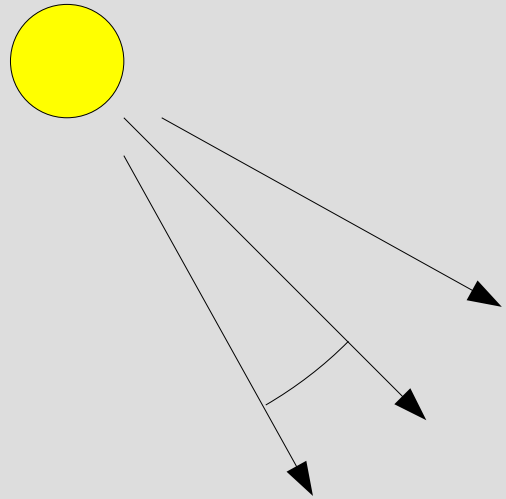
Flächenlichtquelle

- Parallele Lichtstrahlen
- kann entferntes Punktlicht sein
  - z.B. Sonne
- wird durch Länge und Breite beschrieben



Spotlicht

- Punktlicht mit eingeschränktem Winkel
- Schreibtischlampe
- Autoscheinwerfer
- Abtrahlrichtung
  - GL_SPOT_DIRECTION
- Öffnungswinkel
  - GL_SPOT_CUTOFF



Material

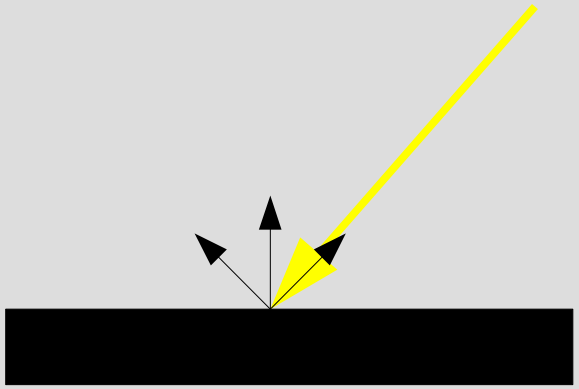
- beschreibt Oberfläche eines Objektes
- Farbe
- spiegelnd oder matt
- Eigenemission
  - z.B. Lampenschirm
  - keine Lichtquelle!

Reflexion

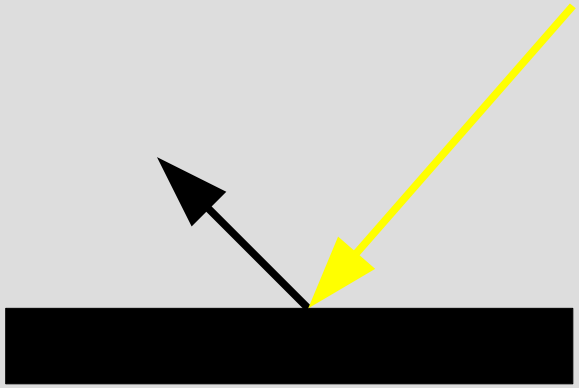
- wenn Licht auf Objekt trifft
- Arten:
  - diffus
  - ideal spiegelnd
  - spekular
- keine Spiegelungen anderer Objekte!

Reflexion

- diffuse
  - eintreffendes Licht wird ungerichtet reflektiert
  - z.B. Stoffe

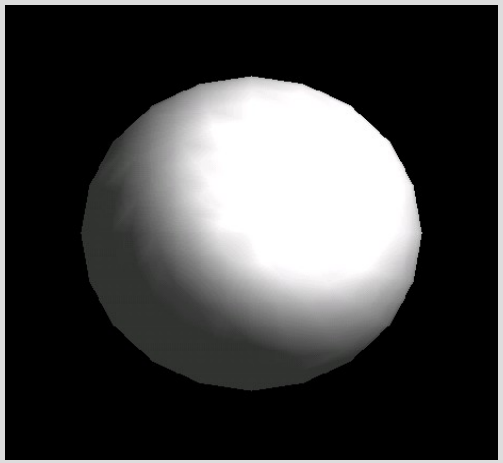


- ideal spiegelnde
  - Einfallswinkel = Ausfallswinkel

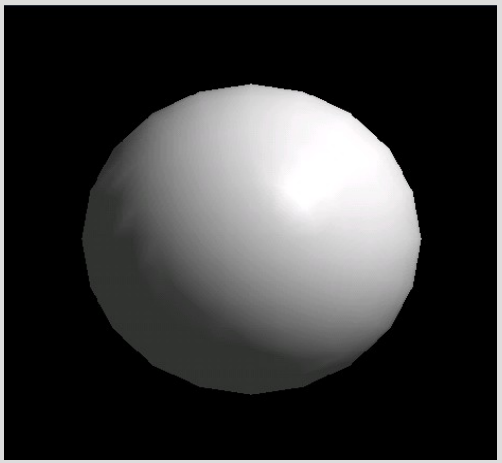


spekulare Reflexion

- diffuser und spekularer Anteil
- Intensität ist
  - am größten in idealer Reflexionsrichtung
  - abnehmend mit Abweichung davon



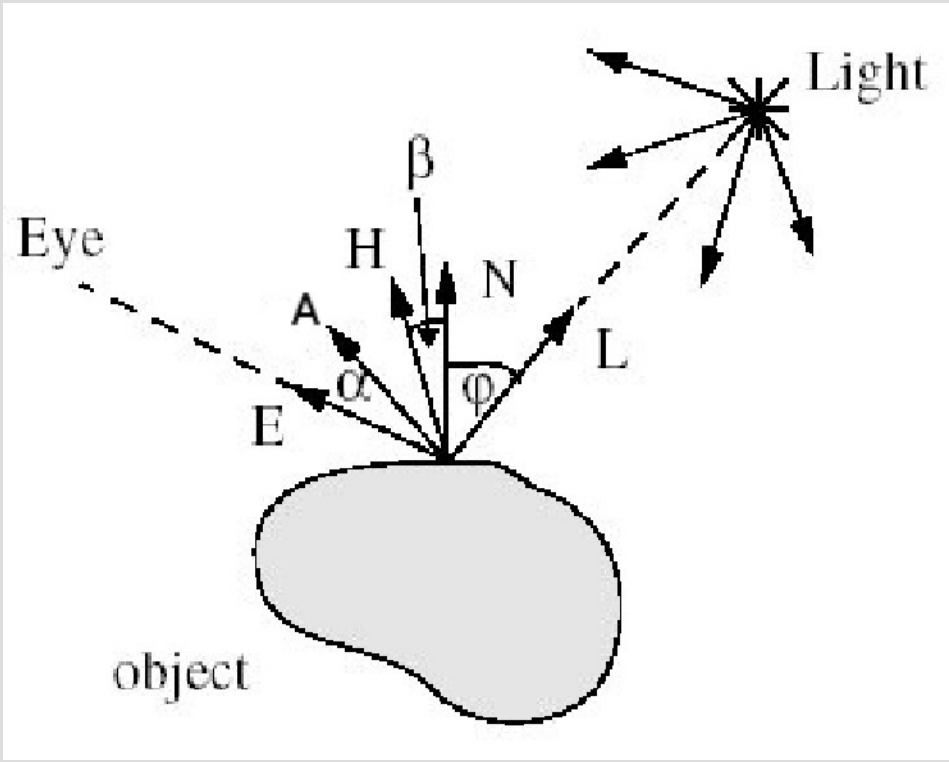
diffus



spekular

Phong Reflexion

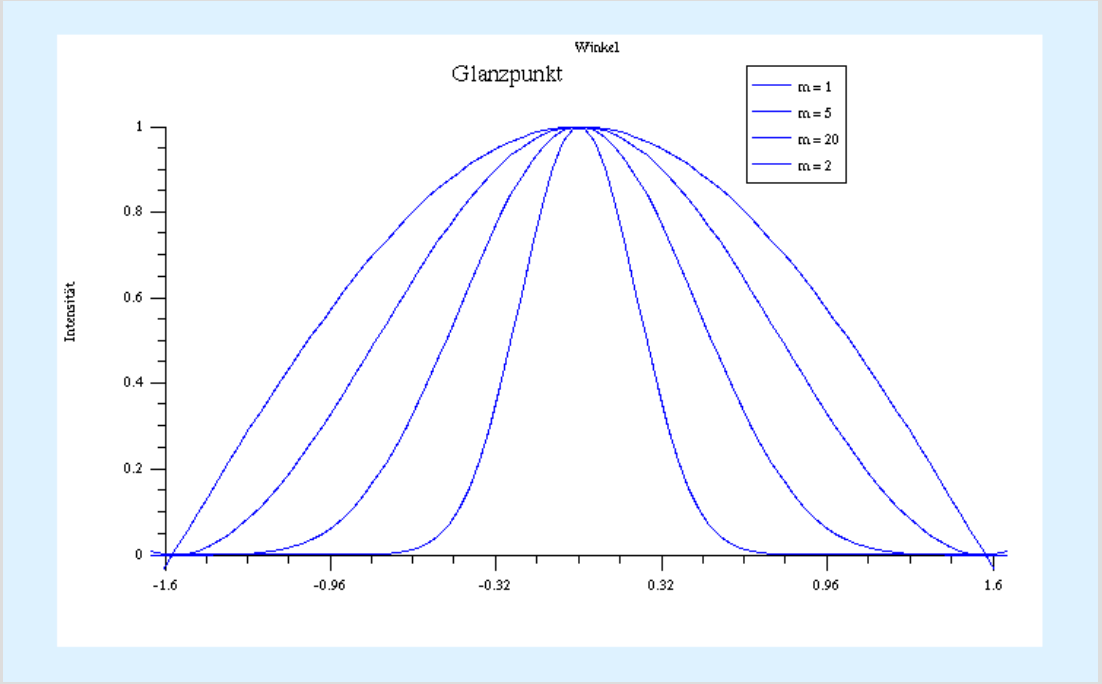
- rein heuristisch
- zwei Konstanten:
  - stärke der Reflexion
  - Abfall der Intensität
- Hängt ab von α
  - in der Praxis wird meist β benutzt



A = ideale Reflexion
H = Halbvektor
N = Normale

Phong Reflexion

- $Intensität = R \cdot \cos^{m} \alpha \cdot I$
- R ist Materialkonstante (GL_SPECULAR)
- bei großem m kleiner Glanzpunkt
  - GL_SHININESS

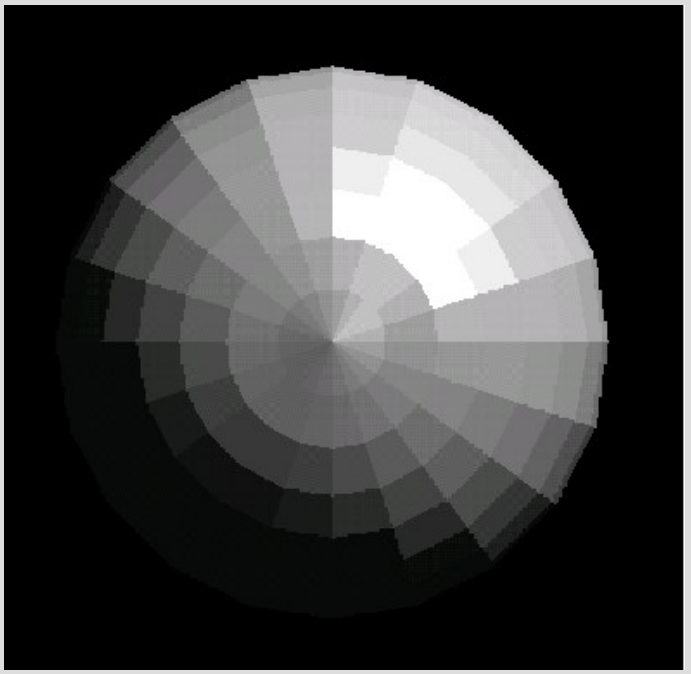


Shading

- Zuordnung von Farben zu Pixeln
- meistens Interpolation
- verschiedene Modelle:
  - Flat
  - Gouraud
  - Phong
- nicht Schattenwurf!

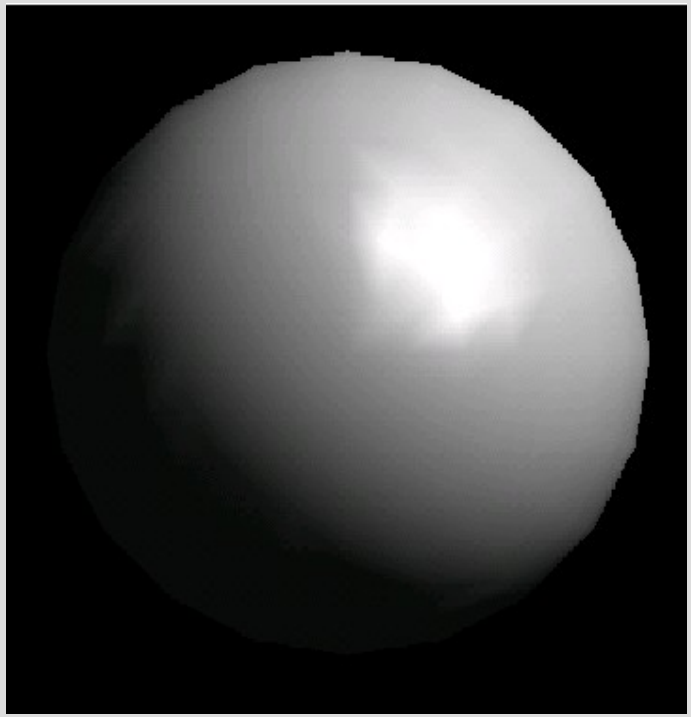
Flat Shading

- einfachste Variante
- auf einen Pixel anwenden → interpolieren
- schlechte Ergebnisse bei Rundungen
- In OpenGL: GL_FLAT



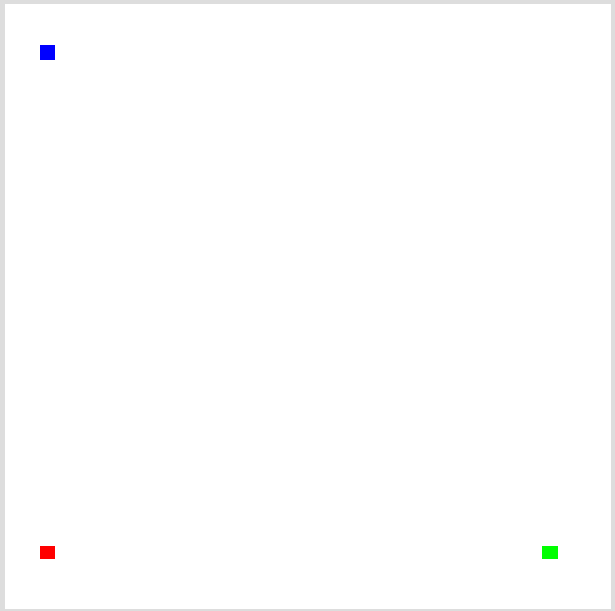
Gouraud Shading

- mehr Rechenaufwand
- realistischere Reflexionen
- Probleme:
  - Glanzpunkte eckig
  - zu kleine verschwinden
- GL_SMOOTH



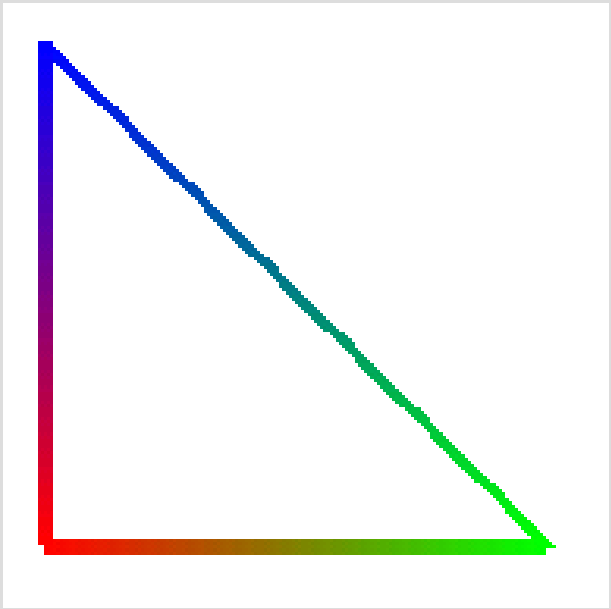
Gouraud Shading

- Interpolation der Farben
- 1. Beleuchtungsmodell auf die Ecken anwenden



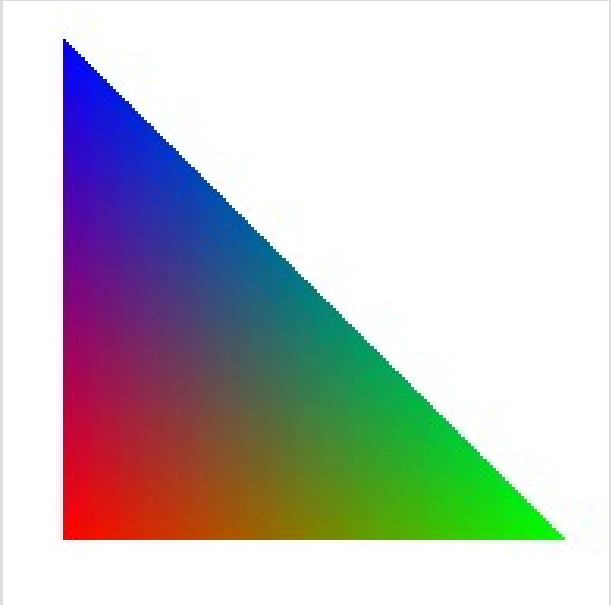
Gouraud Shading

- Interpolation der Farben
- 1. Beleuchtungsmodell auf die Ecken anwenden
- 2. Seiten interpolieren



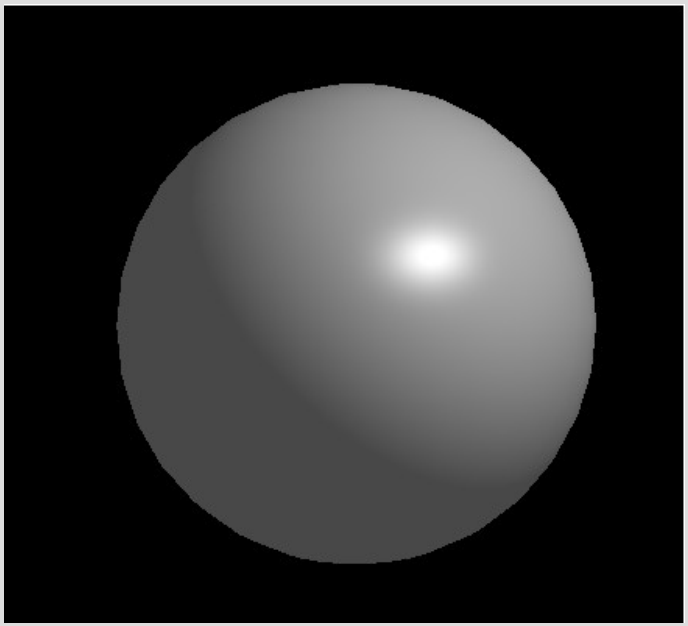
Gouraud Shading

- Interpolation der Farben
- 1. Beleuchtungsmodell auf die Ecken anwenden
- 2. Seiten interpolieren
- 3. Fläche interpolieren (Scan-Line)



Phong Shading

- Interpolation der Normalen
  - selbes Vorgehen wie bei Gouraud
- in OpenGL nur über Shader Programme
- scharfe Glanzpunkte möglich
  - z.B. Metalle



Beispiel

```
void init() {
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 128.0 }
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat lmodel_ambient[] = { 1.0, 1.0, 1.0, 1.0 };
    glClearColor (0.0, 0.0, 0.0, 0.0);









}
```

Beispiel

```
void init() {
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 128.0 }
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat lmodel_ambient[] = { 1.0, 1.0, 1.0, 1.0 };
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);









}
```

Beispiel

```
void init() {
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 128.0 }
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat lmodel_ambient[] = { 1.0, 1.0, 1.0, 1.0 };
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
lmodel_ambient);




}
```

Beispiel

```
void init() {
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0, 1.0 };
    GLfloat mat_shininess[] = { 128.0 }
    GLfloat light_position[] = { 1.0, 1.0, 1.0, 0.0 };
    GLfloat lmodel_ambient[] = { 1.0, 1.0, 1.0, 1.0 };
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
    glLightfv(GL_LIGHT0, GL_POSITION, light_position);
    glLightModelfv(GL_LIGHT_MODEL_AMBIENT,
lmodel_ambient);
    glShadeModel (GL_SMOOTH);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_DEPTH_TEST);
}
```

Danke!