Einführung in die Shaderprogrammierung

M. Kleffmann M. Muckhoff

Xpiriax Software

RPC Professionell, 2009

Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

Geschichte der Shader

- 60er und 70er Jahre (Algorithmenentwicklung)
  - Jack Bresenham entwickelt der Bresenham-Linien-Algorithmus (1962)
  - Ivan Sutherland entwickelt Sketchpad (1963)
  - Henri Gouraud und Bui-Thuong Phong entwickeln für Sutherland das Gouraud- (1971) und Phongshading (1975)
  - Edwin Catmull entwickelt (zeitgleich mit Wolfgang Straßer) den Z-Buffer-Algorithmus (1974)
- 80er Jahre (Hardwareentwicklung)
  - Gründung von Silicon Graphics (1981) (OpenGL)
  - erste »günstige« TARGA-Grafikkarte mit HW-Framebuffer von AT&T (1985)
  - Veröffendlichung der Grafikstandards CGA (1981), EGA (1984) und VGA (1987) durch IBM

Geschichte der Shader (2)

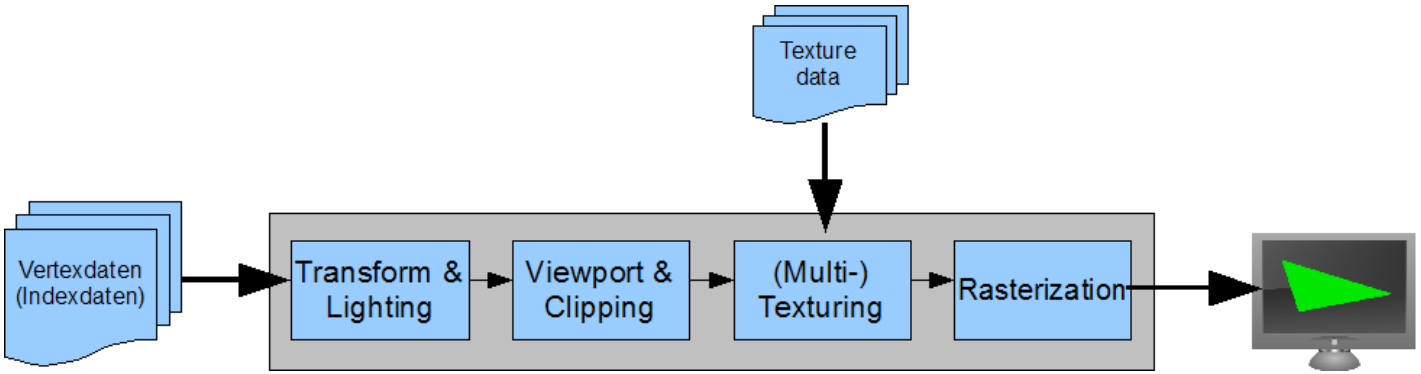
- 90er Jahre (Spieleentwicklung und 3D-Karten)
  - SGI veröffentlicht OpenGL (1992)
  - id Software entwickelt Doom (1993) und Quake (1996)
  - Microsoft entwickelt DirectX als Nachfolger von WinG (1995)
  - 3dfx stellt den Voodoo Graphics Grafikprozessor vor (1996)
  - Mit Direct3D schafft DirectX 3.0 endlich die Ablösung von DOS als Spieleentwicklungsplattform (1996)
- seit 2000 (Grafikpipeline und Shader)
  - Mit DirectX 7 hält endlich auch die Grafikpipeline Einzug bei DirectX (1999)
  - Einführung der Programmierbaren Shader mit DirectX 8 (November 2000)
  - DirectX 9 bietet HLSL (High Level Shading Language) (2004)
  - DirectX 10 führt das Uniform Shader Modell ein (Shader 4)

Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

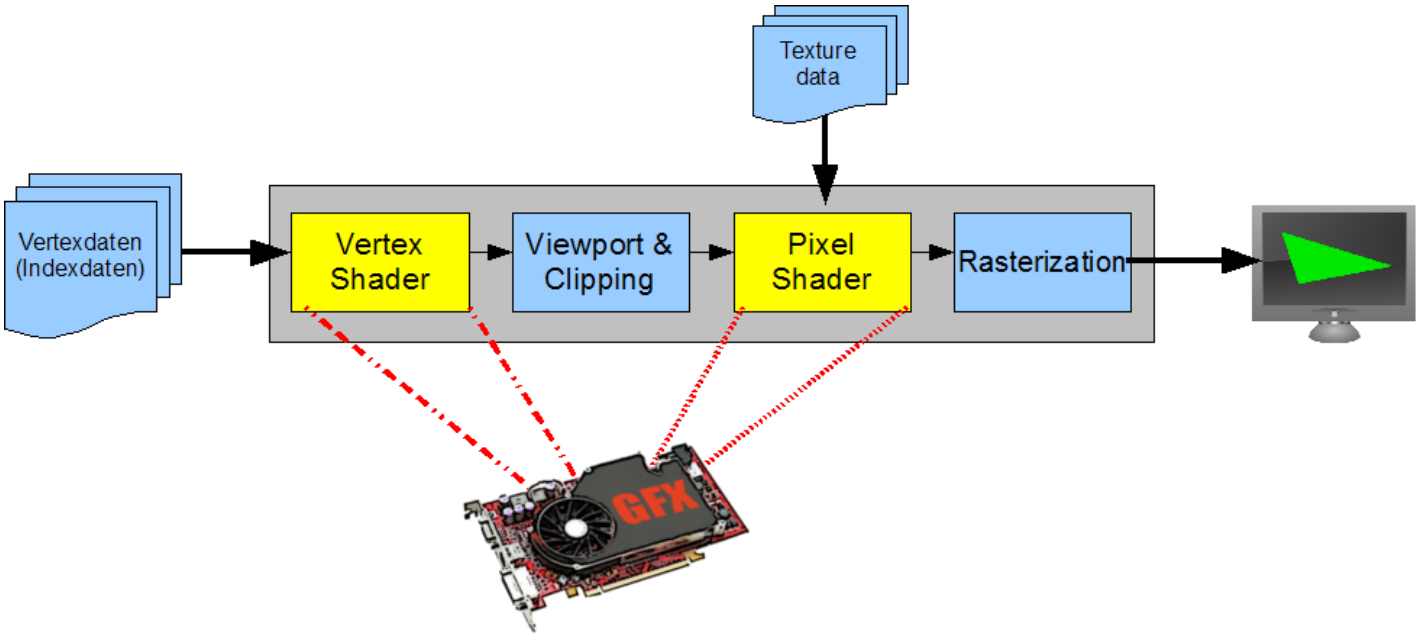
Was ist eine 3D-Pipeline?

- Angelehnt an die Fließbandarbeit in der Autoindustrie
- Grafikdaten durchlaufen Arbeitsstationen
- War ursprünglich rein Softwareimplementiert

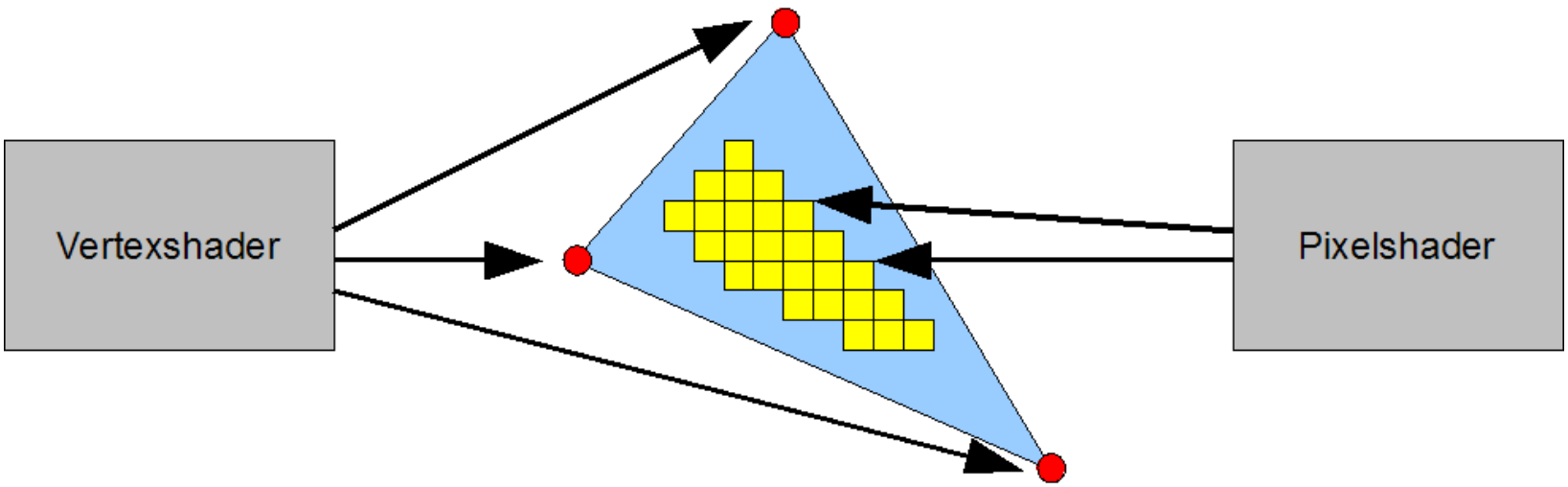


Heutige 3D-Pipeline

- Mischung aus Hard- und Softwareimplementierung
- Geschwindigkeitsrelevante Stationen (gelb) in Hardware implementiert



Vertex- & Pixelshader



Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

Was ist Shading?

- Technik zur Simulation von Oberflächendetails
- wird im Zusammenhang mit einem (lokalen) Beleuchtungsmodell genutzt
- Bekannte Shading-Verfahren
  - Flat Shading
  - Gouraud Shading
  - Phong Shading

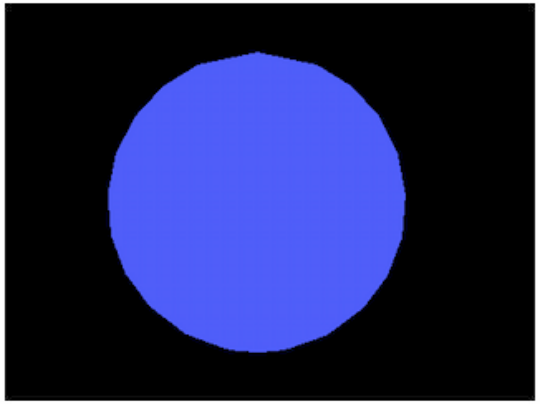
Was ist Shading?

- Technik zur Simulation von Oberflächendetails
- wird im Zusammenhang mit einem (lokalen) Beleuchtungsmodell genutzt
- Bekannte Shading-Verfahren
  - Flat Shading
  - Gouraud Shading
  - Phong Shading

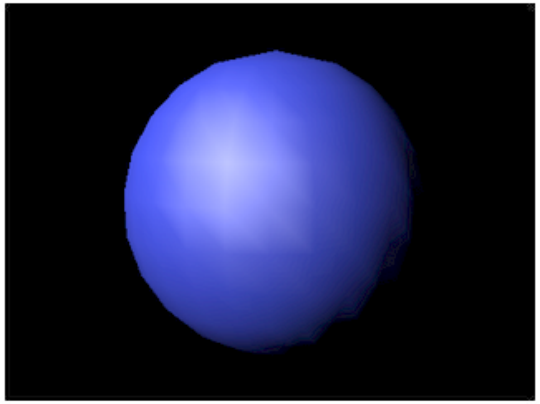
Was ist Shading?

- Technik zur Simulation von Oberflächendetails
- wird im Zusammenhang mit einem (lokalen) Beleuchtungsmodell genutzt
- Bekannte Shading-Verfahren
  - Flat Shading
  - Gouraud Shading
  - Phong Shading

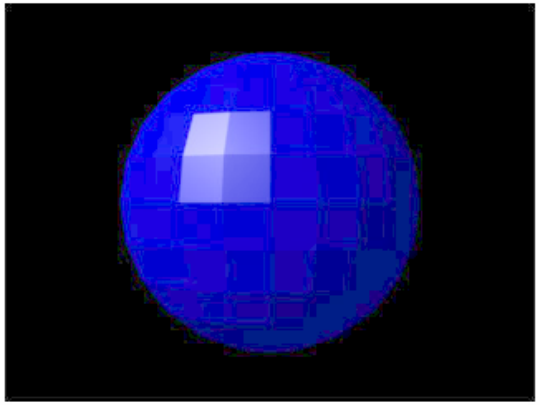
Vergleich Shadingverfahren



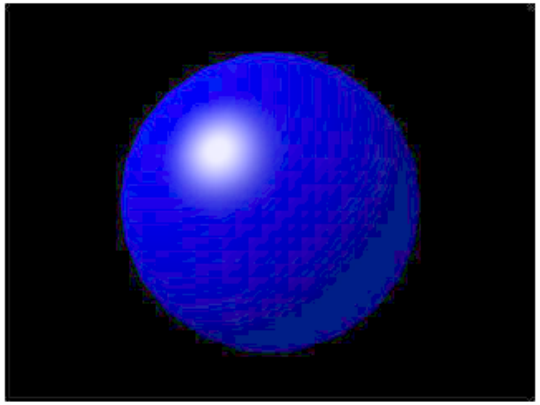
Kein Shading



Gouraud Shading



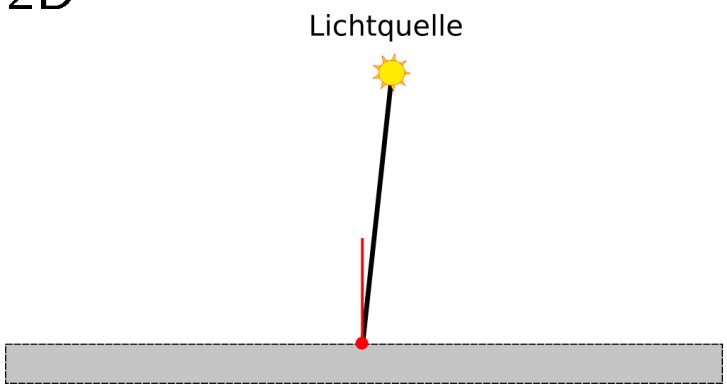
FLAT SHADING



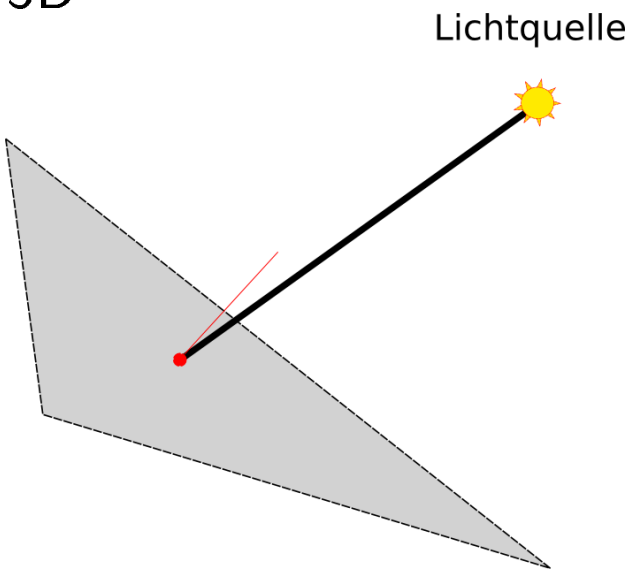
PHONG SHADING

Flat Shading

2D

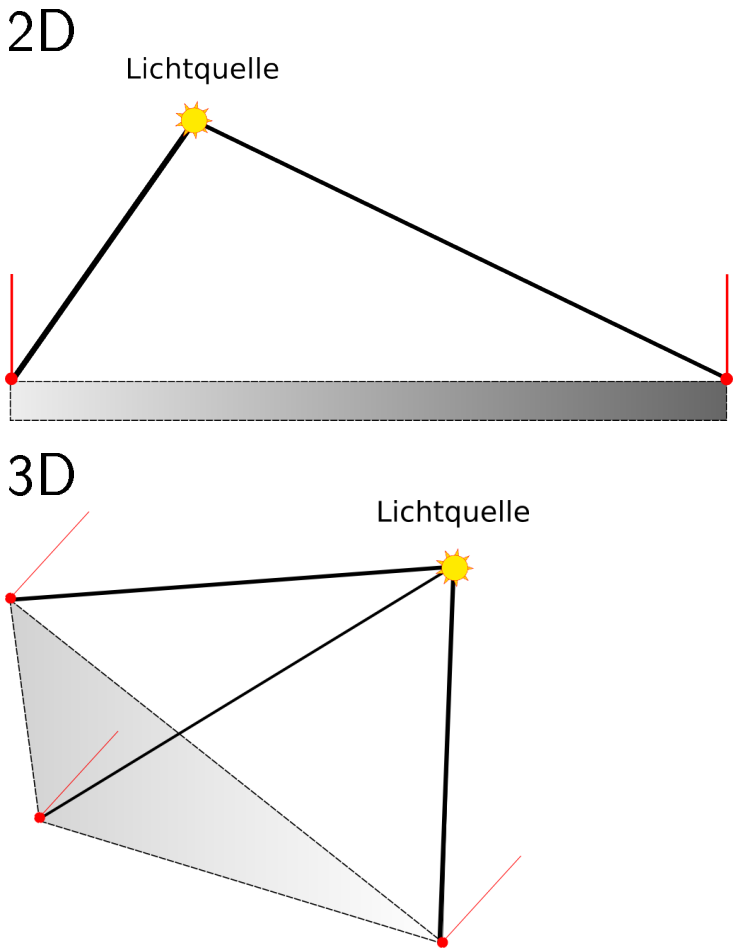


3D



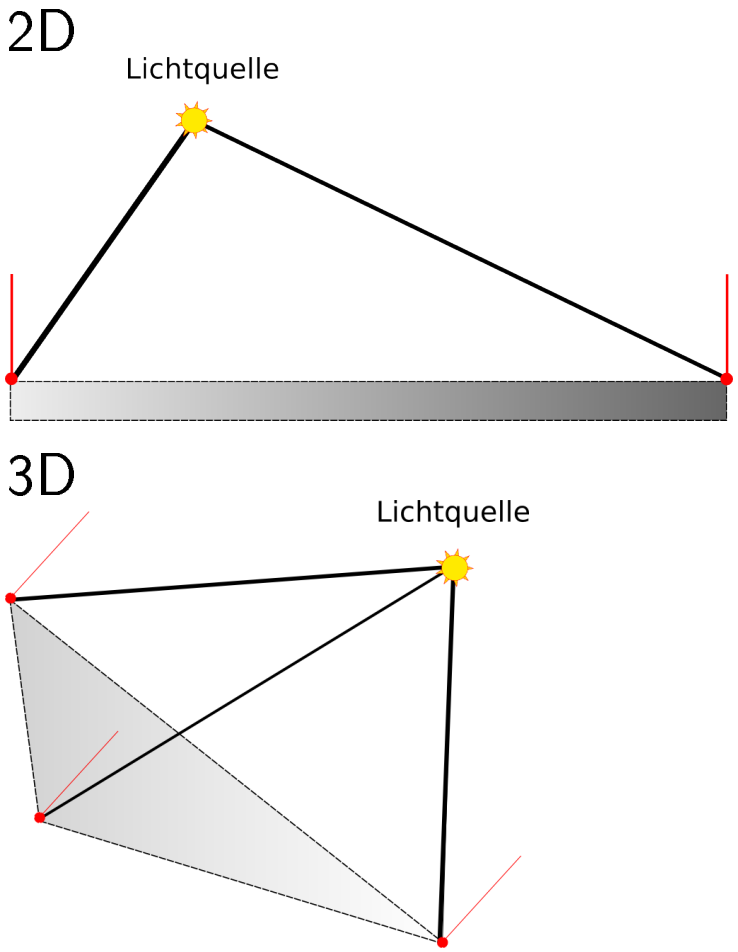
- Lichtberechnung einmal pro Dreieck
- Berechnungspunkt meist Mitte oder erster Punkt

Gouraud Shading



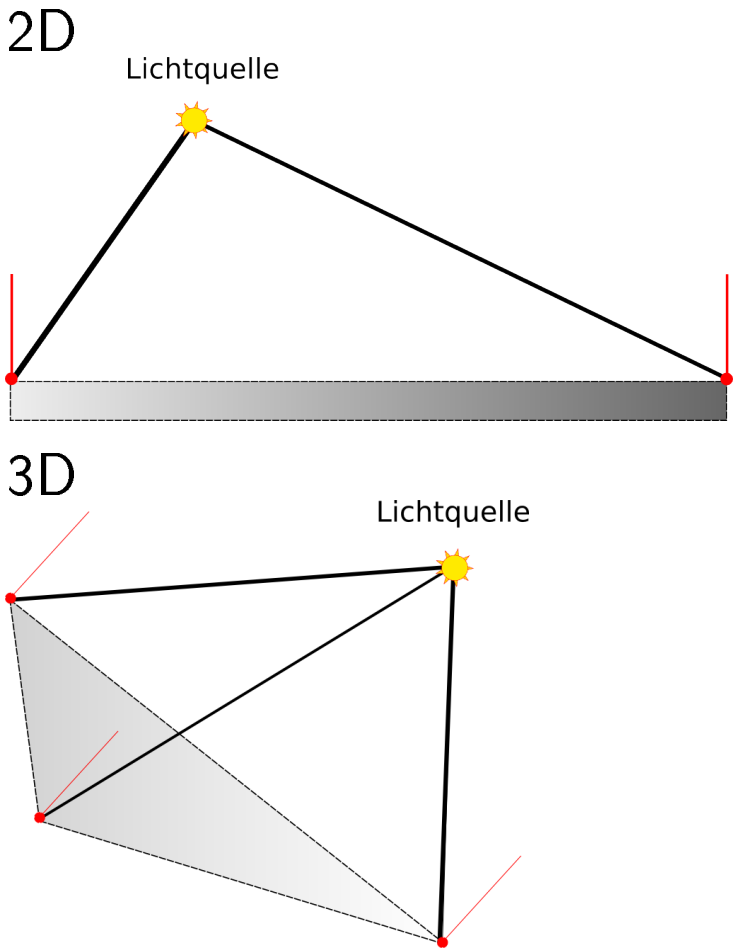
- Lichtberechnung für jeden Eckpunkt des Dreiecks
- Berechnete Lichtintensität wird interpoliert
- Bei großen Dreiecken sind manche Lichtquellen nicht sichtbar

Gouraud Shading



- Lichtberechnung für jeden Eckpunkt des Dreiecks
- Berechnete Lichtintensität wird interpoliert
- Bei großen Dreiecken sind manche Lichtquellen nicht sichtbar

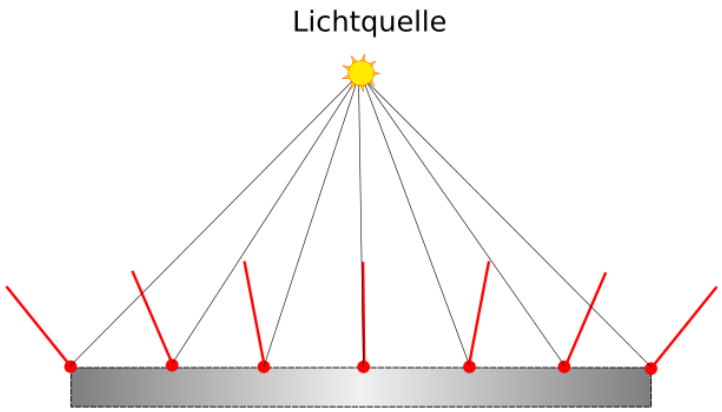
Gouraud Shading



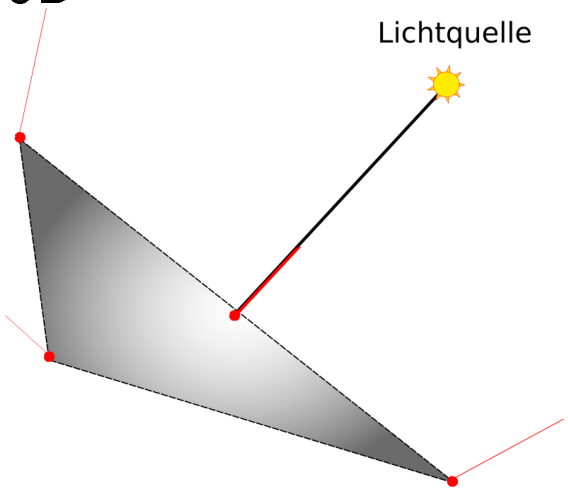
- Lichtberechnung für jeden Eckpunkt des Dreiecks
- Berechnete Lichtintensität wird interpoliert
- Bei großen Dreiecken sind manche Lichtquellen nicht sichtbar

Phong Shading

2D



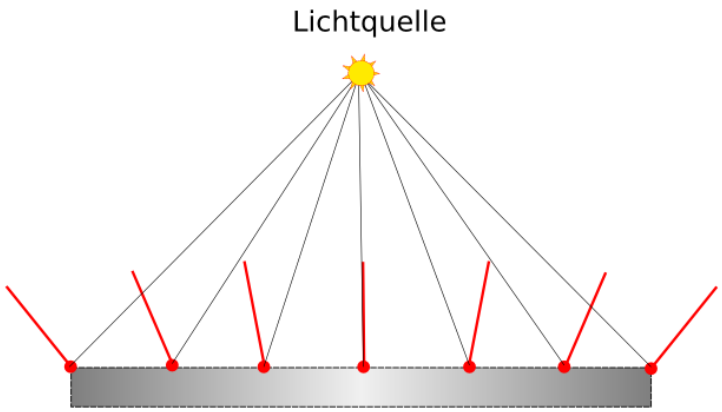
3D



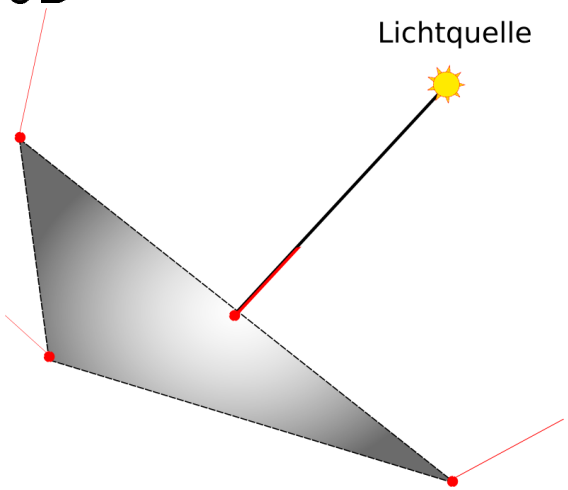
- Lichtberechnung pro Pixel
- Berechnungsmerkmale werden interpoliert
- eigentlich sehr kostenintensiv

Phong Shading

2D



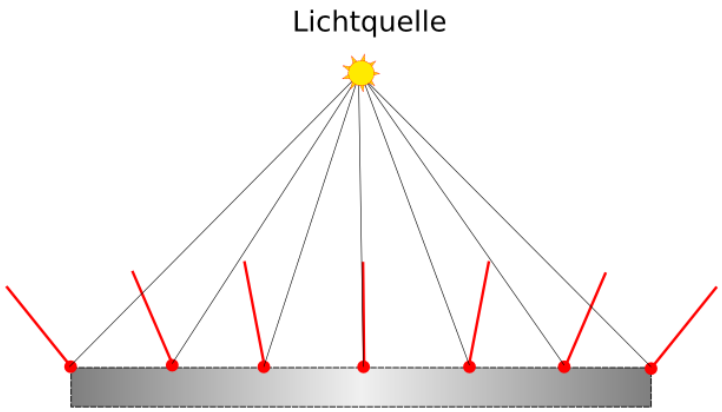
3D



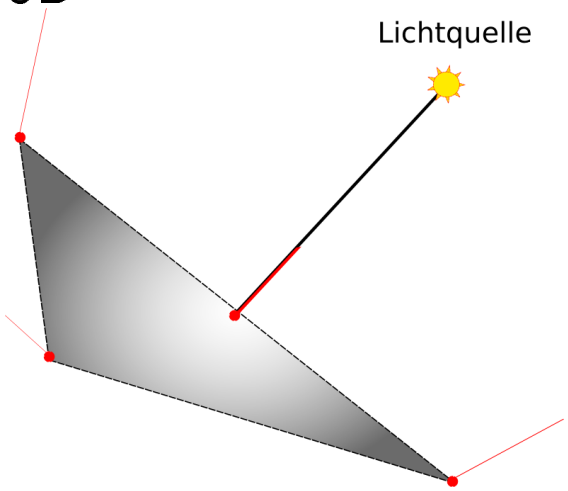
- Lichtberechnung pro Pixel
- Berechnungsmerkmale werden interpoliert
- eigentlich sehr kostenintensiv

Phong Shading

2D

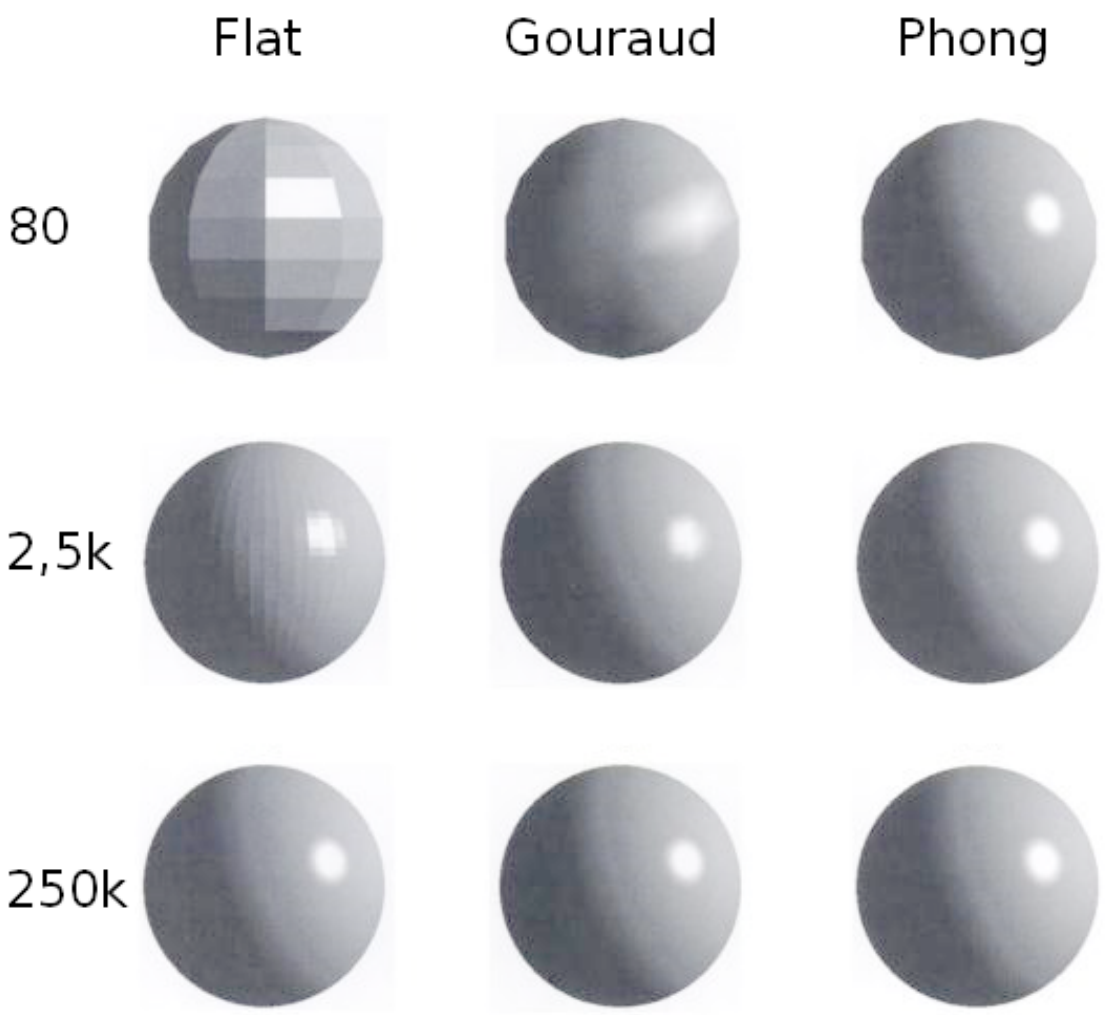


3D



- Lichtberechnung pro Pixel
- Berechnungsmerkmale werden interpoliert
- eigentlich sehr kostenintensiv

Direkter Vergleich



Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

Was ist ein (lokales) Beleuchtungsmodell?

- Simuliert die Wirkung von Licht auf Oberflächen
- meist pseudophysikalisch

- genauer:
  beschreibt die Zusammenhänge zwischen
  - Raumgeometrie,
  - Lichtquellen und
  - Oberflächenbeschaffenheit

  zur Berechnung der Leuchtdichte in einem Punkt der darzustellenden Szene.

Was ist ein (lokales) Beleuchtungsmodell?

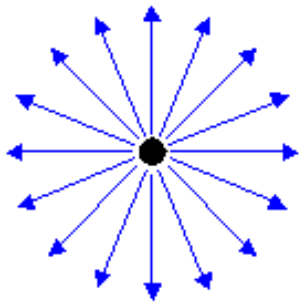
- Simuliert die Wirkung von Licht auf Oberflächen
- meist pseudophysikalisch

- genauer:
  beschreibt die Zusammenhänge zwischen
  - Raumgeometrie,
  - Lichtquellen und
  - Oberflächenbeschaffenheit

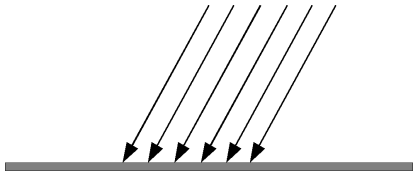
  zur Berechnung der Leuchtdichte in einem Punkt der darzustellenden Szene.

Lichtquellen

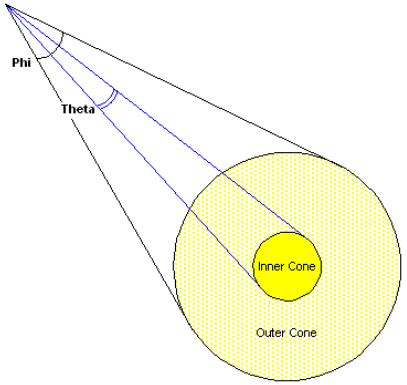
- Punktlichtquellen
  - Position
  - Radius



- Richtungslichtquellen
  - Richtung



- Strahler
  - Position
  - Richtung
  - Radius



Lichtintensität

- Lichtintensität nimmt mit dem Quadrat zum Meter ab

$$I_L = \frac{1}{\text{Abstand}^2}$$

- etwas »natürlicheres« Modell

$$I_L = \min\left(\frac{1}{c_1 + c_2 \cdot \text{Abstand} + c_3 \cdot \text{Abstand}^2}, 1\right)$$

Lichtintensität

- Lichtintensität nimmt mit dem Quadrat zum Meter ab

$$I_L = \frac{1}{\text{Abstand}^2}$$

- etwas »natürlicheres« Modell

$$I_L = \min\left(\frac{1}{c_1 + c_2 \cdot \text{Abstand} + c_3 \cdot \text{Abstand}^2}, 1\right)$$

Ein Standardbeleuchtungsmodell

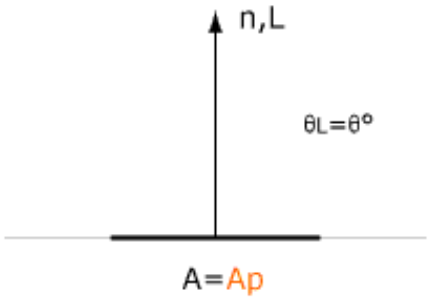
- Jede Lichtfarbe besteht aus Rot-, Grün- und Blauanteilen
  $$L = \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$
- Unser Licht besteht aus ambienten, diffusen und spekularen Lichtanteilen
- Jedes Material definiert, wie stark es auf den Anteil reagiert
  $M = \begin{pmatrix} M_A & M_D & M_S \end{pmatrix} = \begin{pmatrix} 0.5 & 0.2 & 0.1 \end{pmatrix}$
  $$L_{Ges} = M_A \cdot L_A + M_D \cdot L_D + M_S \cdot L_S$$

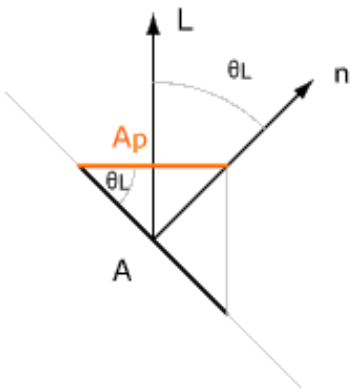
Ambientes Licht

- Umgebungslicht (Grundhelligkeit)
- simuliert indirekte Beleuchtung
- wird gleichmäßig über die Szene verteilt

Diffuses Licht

- Idee einer idealdiffusen Oberfläche
- Licht strahlt gleichmäßig in alle Richtungen ab
- Abgestrahlte Lichtintensität hängt vom Einfallwinkel ab
  - Lambertsches Gesetz: $L_D = I_L \cdot \max(0, \langle \hat{l}, \hat{n} \rangle)$





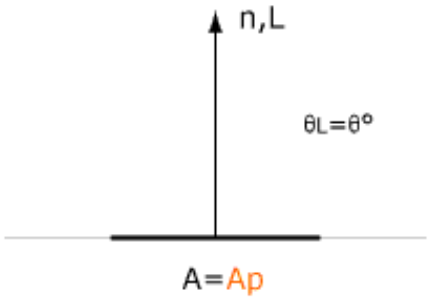
Ap= projizierte Fläche

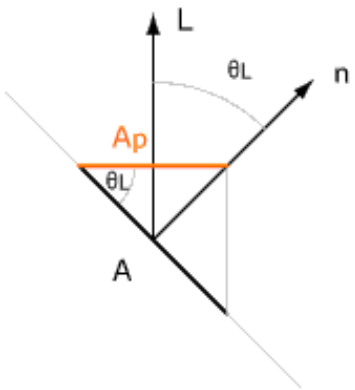
|Ap| =|A| cos θL

Erinnerung: $\cos\theta = \frac{l \cdot n}{|l| \cdot |n|}$

Diffuses Licht

- Idee einer idealdiffusen Oberfläche
- Licht strahlt gleichmäßig in alle Richtungen ab
- Abgestrahlte Lichtintensität hängt vom Einfallwinkel ab
  - Lambertsches Gesetz: $L_D = I_L \cdot \max(0, \langle \hat{l}, \hat{n} \rangle)$





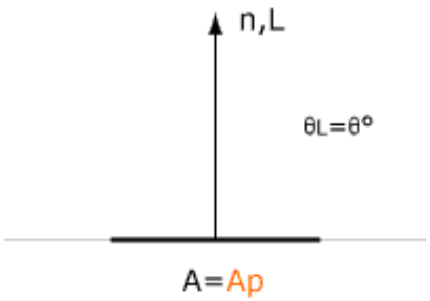
Ap= projizierte Fläche

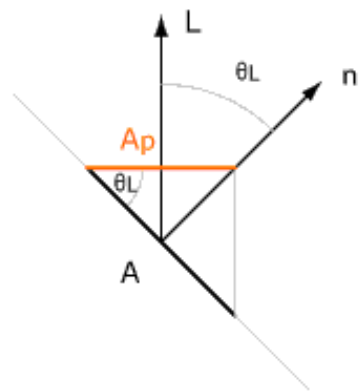
|Ap| =|A| cos θL

Erinnerung: $\cos\theta = \frac{l \cdot n}{|l| \cdot |n|}$

Diffuses Licht

- Idee einer idealdiffusen Oberfläche
- Licht strahlt gleichmäßig in alle Richtungen ab
- Abgestrahlte Lichtintensität hängt vom Einfallwinkel ab
  - Lambertsches Gesetz: $L_D = I_L \cdot \max(0, \langle \hat{l}, \hat{n} \rangle)$



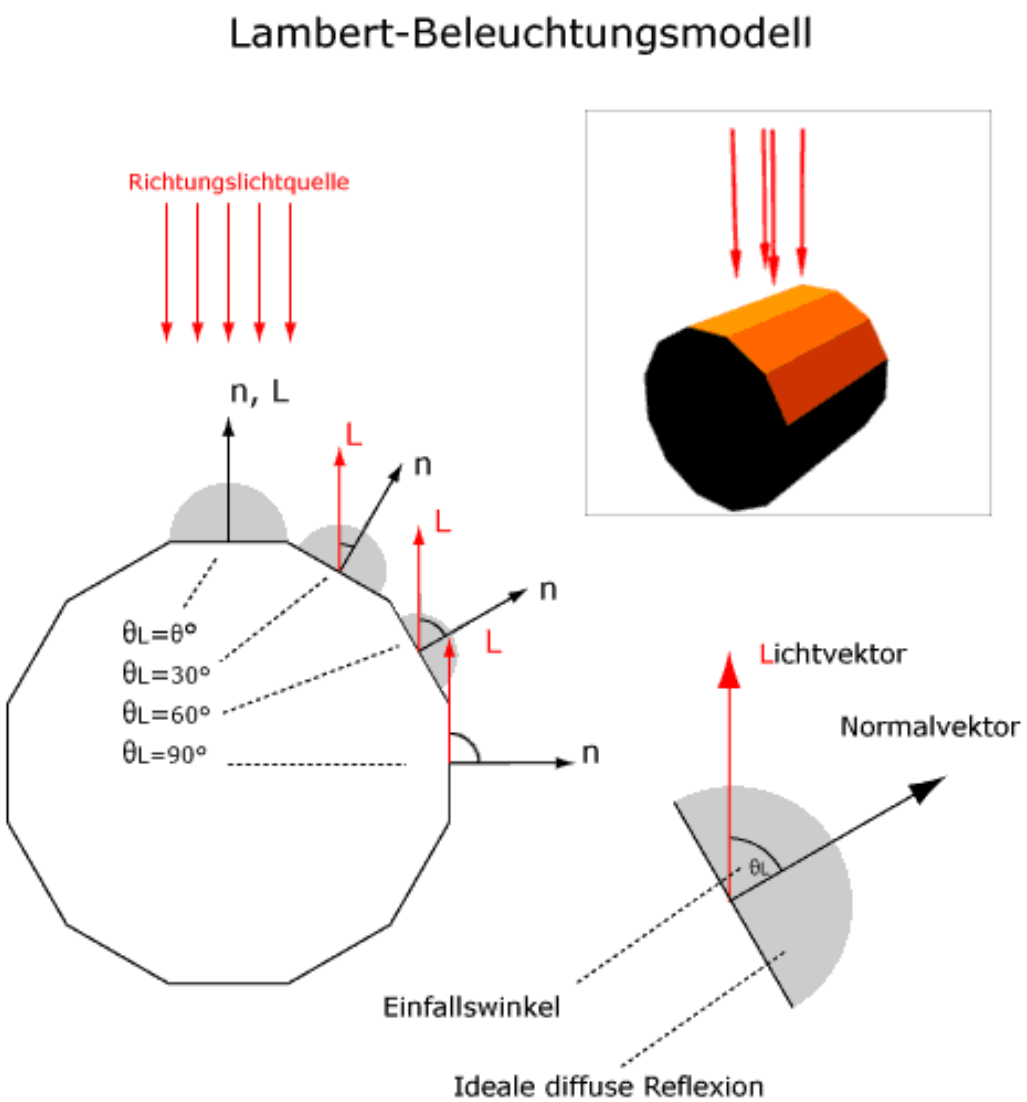


Ap= projizierte Fläche

|Ap| =|A| cos θL

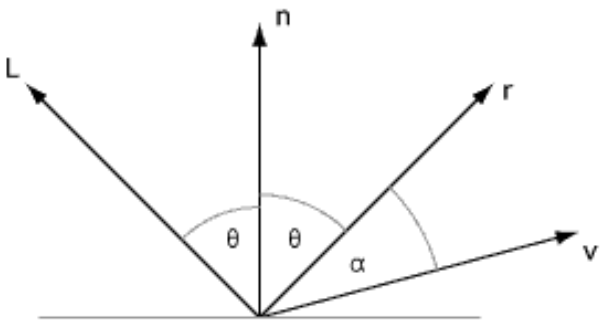
Erinnerung: $\cos\theta = \frac{l \cdot n}{|l| \cdot |n|}$

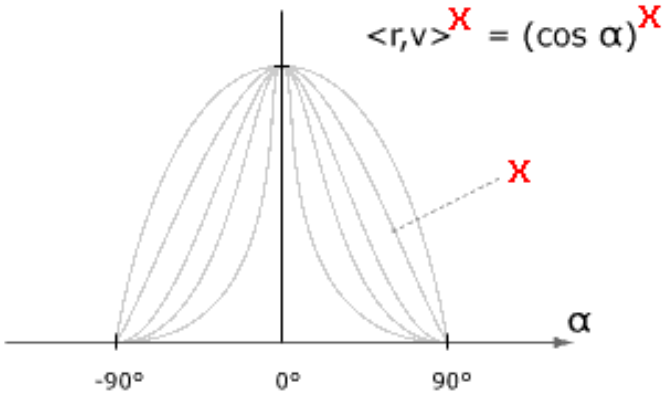
Diffuses Licht Beispiel



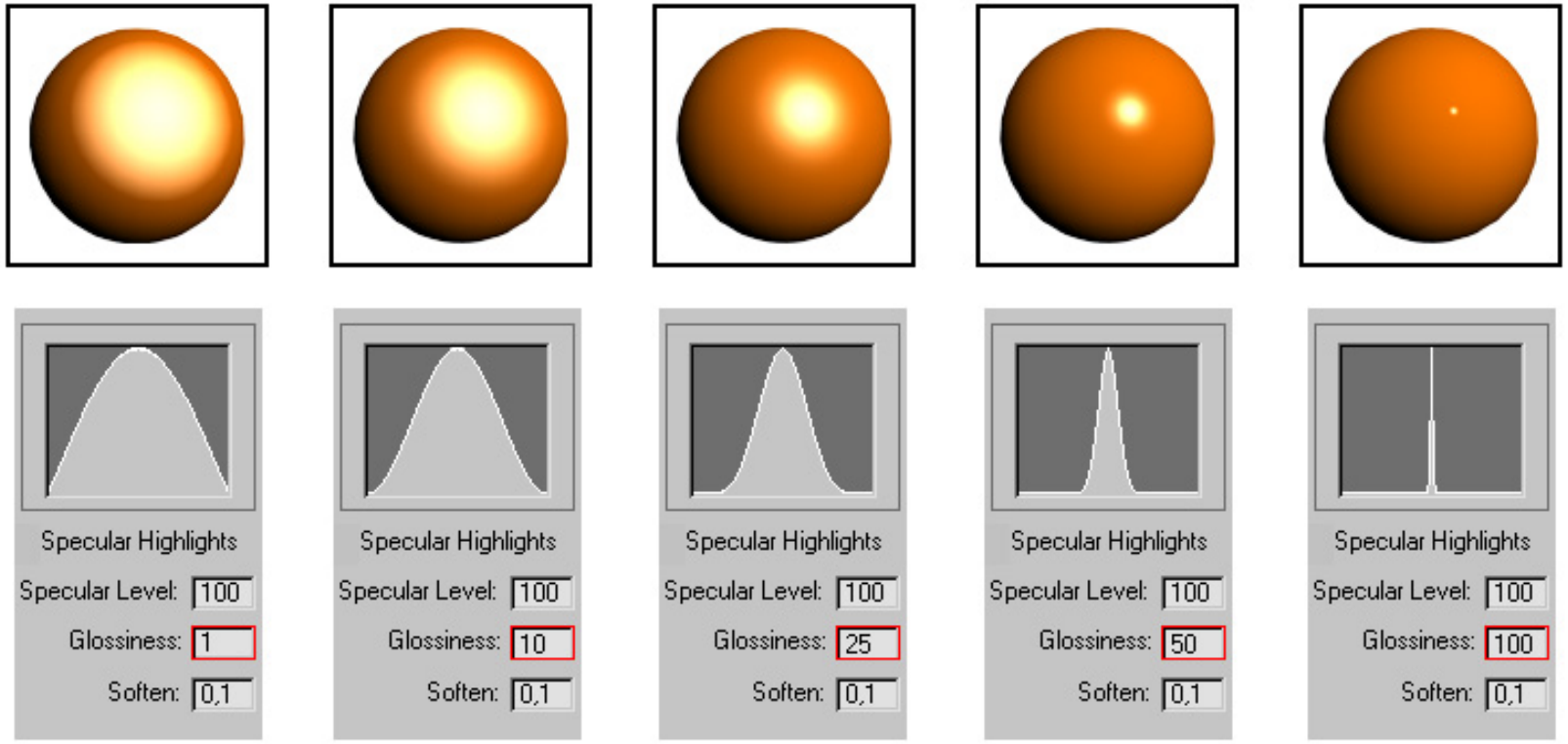
Spekulares Licht (Phong)

- Simuliert realistische Lichtreflexion
- ist vom Betrachtungswinkel abhängig
- Phong-Term: $L_S = I_L \cdot \max(0, (\cos\alpha)^x) = I_L \cdot \max(0, \langle \hat{r}, \hat{v} \rangle^x)$





Wirkung des Exponenten (Phong-Modell)



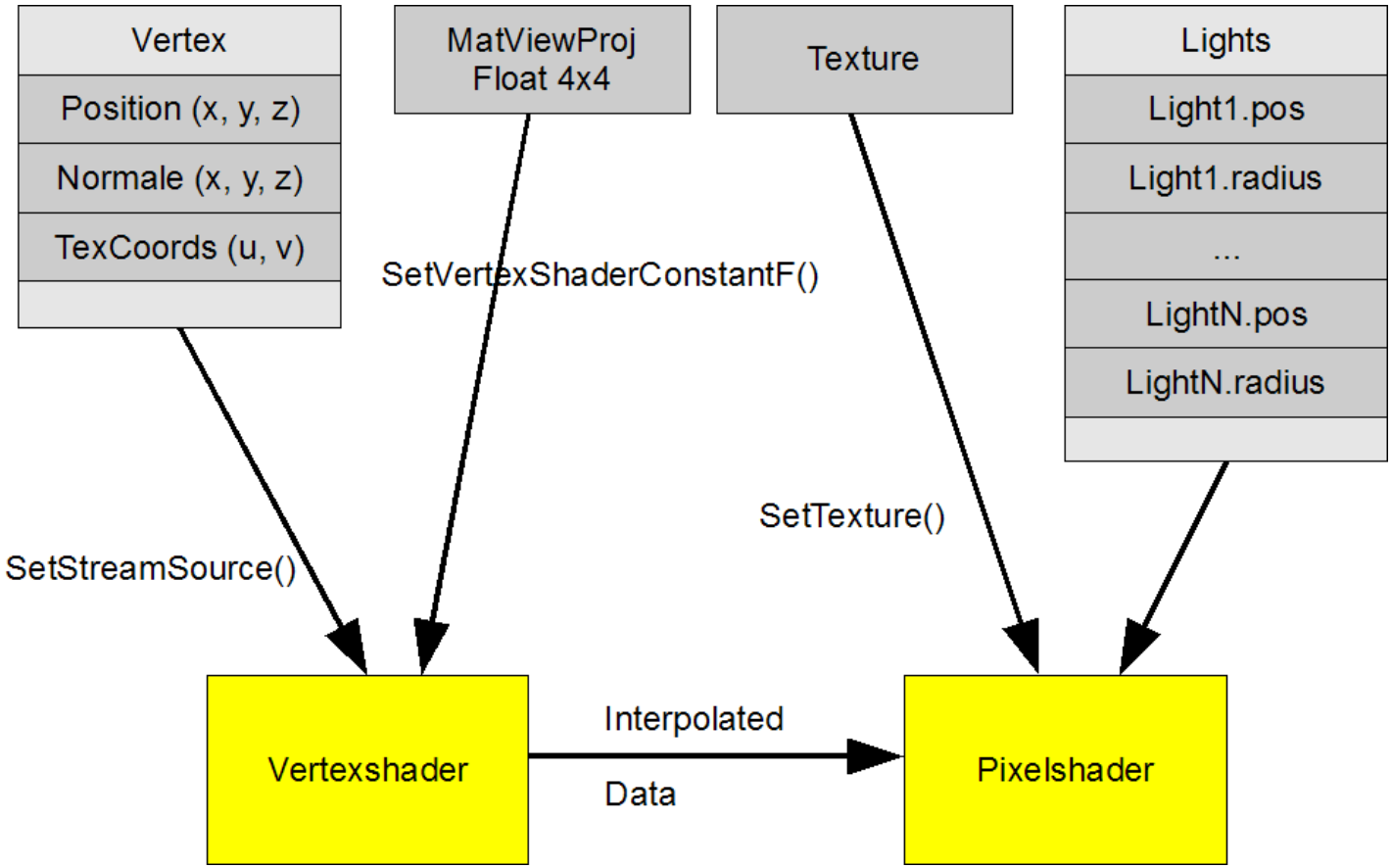
Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

Shadersprachen

- Assembler
  - nicht problem- sondern systemorientiert
  - Optimierungen werden immer schwieriger
  - wird ab DirectX 10 nicht mehr unterstützt
- HLSL (Cg)
  - DirectX und OpenGL
  - performante OpenGL-Implementierung nur von NVidia und ATI
- GLSL
  - neuer Industriestandard
  - integriert in OpenGL 2.0

Drawing Setup (DirectX)



Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

Ein erstes Beispiel

Aufgabe: Einen Vertex- und einen Pixelshader bauen, um Objekte am Bildschirm *in Rot* zu zeichnen.

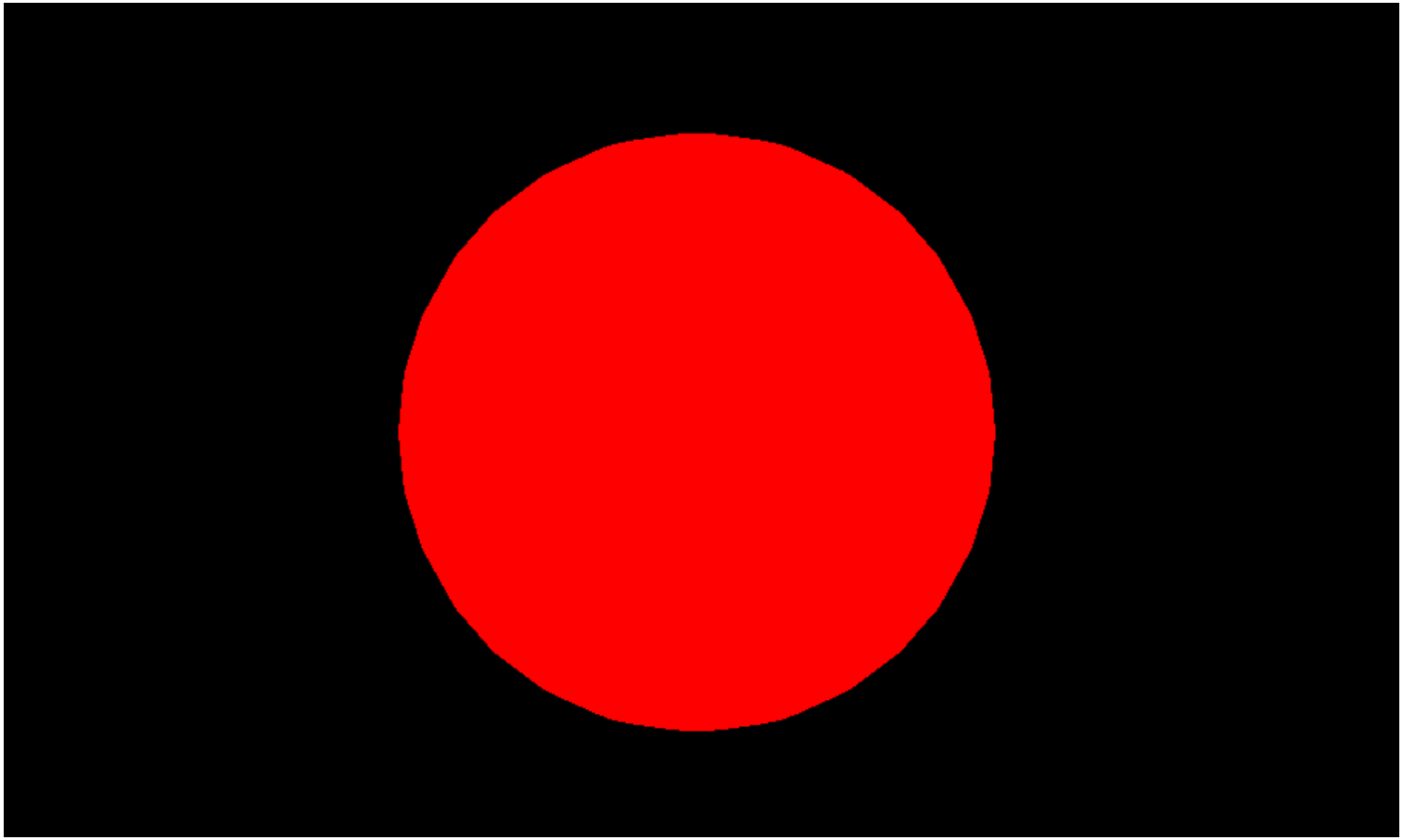
Vertexshader (HLSL)

```
1   float4x4 matViewProjection;
2
3   struct VS_INPUT
4   {
5       float4 Position : POSITION0;
6       float4 Position : NORMAL0;
7       float2 Position : TEXCOORD0;
8   };
9
10  struct VS_OUTPUT
11  {
12      float4 Position : POSITION0;
13  };
14
15  //main code
16  VS_OUTPUT vs_main( VS_INPUT Input )
17  {
18      VS_OUTPUT Output;
19
20      Output.Position = mul( Input.Position, matViewProjection );
21
22      return( Output );
23  }
```

Pixelshader (HLSL)

```
1  float4 ps_main() : COLOR0
2  {
3      return( float4( 1.0f, 0.0f, 0.0f, 1.0f ) );
4  }
```

Ergebnis



Gliederung

1. Einführung
   - Geschichte
2. Grundlagen
   - 3D-Pipeline
   - Shading
   - Beleuchtungsmodelle
3. Praxis
   - Einstieg
   - Beispiel 1
   - Beispiel 2
4. Fazit

Texturierte Kugel

Aufgabe: Einen Vertex- und einen Pixelshader bauen, um eine texturierte Kugel zu zeichnen.

Vertexshader (HLSL)

```
1   float4x4 matViewProjection;
2
3   struct VS_INPUT
4   {
5       float4 Position : POSITION0;
6       float4 Position : NORMAL0;
7       float2 Texcoord : TEXCOORD0;
8   };
9
10  struct VS_OUTPUT
11  {
12      float4 Position : POSITION0;
13      float2 Texcoord : TEXCOORD0;
14  };
15
16  VS_OUTPUT vs_main( VS_INPUT Input )
17  {
18      VS_OUTPUT Output;
19
20      Output.Position = mul( Input.Position, matViewProjection );
21      Output.Texcoord = Input.Texcoord;
22
23      return( Output );
24  }
```

Pixelshader (HLSL)

```
1   sampler2D baseMap;
2
3   struct PS_INPUT
4   {
5       float2 Texcoord : TEXCOORD0;
6   };
7
8   float4 ps_main( PS_INPUT Input ) : COLOR0
9   {
10      return tex2D( baseMap, Input.Texcoord );
11  }
```

Ergebnis

Fazit

Für Shaderprogrammierung braucht man

- ein wenig Mathematik
- Basiswissen über Shadingverfahren und Beleuchtungsmodelle
- Grundkenntnisse in DirectX und/oder OpenGL
- rudimentäre C-Kenntnisse

Tools & Books

- NVidia FX Composer
- ATI Rendermonkey
- Microsoft Effect-Compiler Tool

- **Real-Time Shader Programming**
  Ron Fosner, Morgen Kaufmann, ISBN: 1558608532
- **3D-Effekte für Spieleprogrammierer**
  Alexander Rudolph, Markt+Technik, ISBN: 3827267781

Danke für ihre Aufmerksamkeit!