Lighting and Shading II
Prof. Dr. Markus Gross
Shading Models

• Flat Shading
  – One color per primitive
Shading Models

• Gouraud Shading
  – Linear interpolation of vertex intensities
Shading Models

• Phong Shading
  – Linear interpolation of vertex normals
Shading Models

• In screen space
  – Flat shading
  – Gouraud Shading

• In object space
  – Phong Shading

• Impacts performance
Gouraud Shading

- Calculate face normals
Gouraud Shading

• Calculate face normals
• Calculate vertex normals by averaging
Gouraud Shading

- Calculate face normals
- Calculate vertex normals by averaging
- Evaluate illumination model for each vertex
Gouraud Shading

• Calculate face normals
• Calculate vertex normals by averaging
• Evaluate illumination model for each vertex
• Interpolate vertex colors bilinearly on the current scan line
Gouraud Shading

- Interpolate vertex colors bilinearly on the current scan line

\[
I_a = I_1 - (I_1 - I_2) \frac{(y_1 - y_s)}{(y_1 - y_2)}
\]

\[
I_b = I_1 - (I_1 - I_3) \frac{(y_1 - y_s)}{(y_1 - y_3)}
\]

\[
I_p = I_b - (I_b - I_a) \frac{(x_b - x_p)}{(x_b - x_a)}
\]
Gouraud Shading

- Problems with scan line interpolation
  - Perspective distortion
Gouraud Shading

- Problems with scan line interpolation
  - Orientation Dependence

Interpolate between $AB$ and $AD$

Interpolate between $CD$ and $AD$
Gouraud Shading

- Problems with scan line interpolation
  - Shared Vertices

![Diagram of shared vertices A, B, and C]
Gouraud Shading

• Quality depends on the size of primitives
Phong Shading

- Barycentric Interpolation of normals on the triangles

\[ n_x = \lambda_a n_a + \lambda_b n_b + \lambda_c n_c \]

\[ \lambda_a = \frac{\triangle a}{\triangle ABC} \quad \lambda_b = \frac{\triangle b}{\triangle ABC} \]

\[ \lambda_c = 1 - \lambda_a - \lambda_b = \frac{\triangle c}{\triangle ABC} \]
Phong Shading

• Barycentric Interpolation of normals on the triangles

Properties
Lagrange
\[ x = a \implies n_x = n_a \]

Partition of unity
\[ \lambda_a + \lambda_b + \lambda_c = 1 \]

Reproduction
\[ \lambda_a \cdot a + \lambda_b \cdot b + \lambda_c \cdot c = x \]
Phong Shading

- Barycentric Interpolation of normals on the triangles
- Color of $x$ is determined by the interpolated normal
Phong Shading

- Problem: normal not defined/representative
Phong Shading

- Phong vs. Gouraud shading
Transparency

• Composite transparent objects
• Alpha blending
  – RGBA channels
Transparency

- Alpha blending
  - linearizes exponential attenuation of intensity

\[ \alpha : \text{absorption} \]
Transparency

- Alpha blending

\[ I_\lambda = I'_{\lambda_1} + I'_{\lambda_2} \]

\[ = I_{\lambda_1} \alpha_1 \Delta t + I_{\lambda_2} e^{-\alpha_1 \Delta t} \]
Transparency

• Alpha blending

\[ I_{\lambda} = I_{\lambda_1} \alpha_1 \Delta t + I_{\lambda_2} e^{-\alpha_1 \Delta t} \]

Linearization

\[ I'_{\lambda_2} = I_{\lambda_2} (1 - \alpha_1 \Delta t) \]

Result

\[ I_{\lambda} = I_{\lambda_1} \alpha_1 + I_{\lambda_2} (1 - \alpha_1) \]
Transparency

- Problem: rendering order
Transparency

- Problem: rendering order
  - Object-order rendering
  - We need: sorted traversal of polygons
Transparency

• Back to front rendering
Transparency

- Back to front rendering
Transparency

- Problems with back to front rendering
Transparency

• Solution: depth peeling
  – Multiple passes
  – Each pass renders the next closest fragment
Transparency

- Solution: depth peeling
Transparency

• **Solution:** depth peeling

![Diagram showing layers with different depths](image-url)
Transparency

• Solution: depth peeling
Further Lighting Models

• Cook-Torrence
  – Metal objects
  – Replace the specular term
  – Reflection from micro facets
  – Self-shadowing effects
Further Lighting Models

• Cook-Torrence
  – Metal objects
Further Lighting Models

- Ashikhmin (anisotropic)

Isotropic microfacet distribution  Anisotropic micro facet distribution (au=0.3, av=0.1)