



CS 247 – Scientific Visualization

Lecture 13: Scalar Field Visualization, Pt. 6

Markus Hadwiger, KAUST

Reading Assignment #6+7 (until Mar 29)



Read (required):

- Real-Time Volume Graphics, Chapter 1
(*Theoretical Background and Basic Approaches*),
from beginning to 1.4.4 (inclusive)
- Real-Time Volume Graphics, Chapter 4 (Transfer Functions)
until Sec. 4.4 (inclusive)
- Jens Krüger and Rüdiger Westermann,
Acceleration Techniques for GPU-based Volume Rendering,
IEEE Visualization 2003,
<http://dl.acm.org/citation.cfm?id=1081482>

Read (optional):

- *Nelson Max, Optical Models for Direct Volume Rendering*,
IEEE Transactions on Visualization and Computer Graphics, 1995
<http://dx.doi.org/10.1109/2945.468400>

The marching cubes algorithm

Loop over cells:

- find sign of $\tilde{f}(x_i)$ for the 8 corner nodes, giving 8-bit integer
- use as index into (256 case) table
- find intersection points on edges listed in table, using linear interpolation
- generate triangles according to table

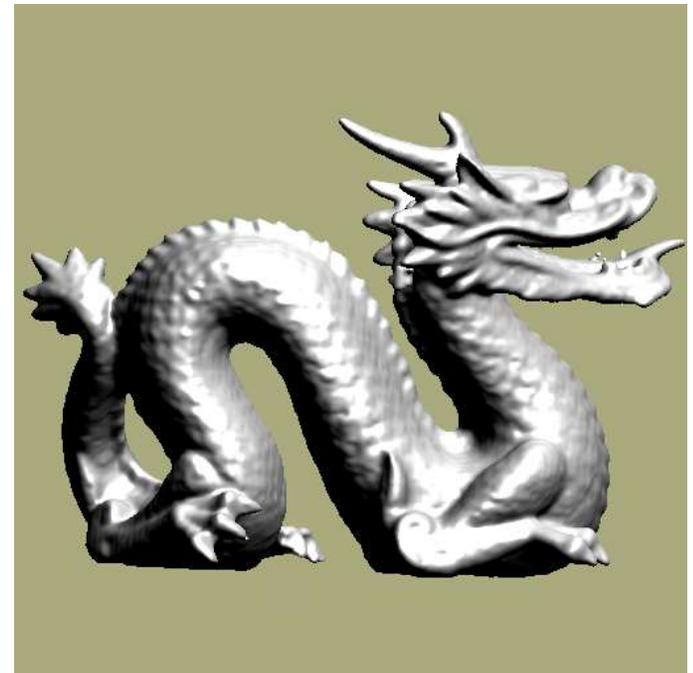
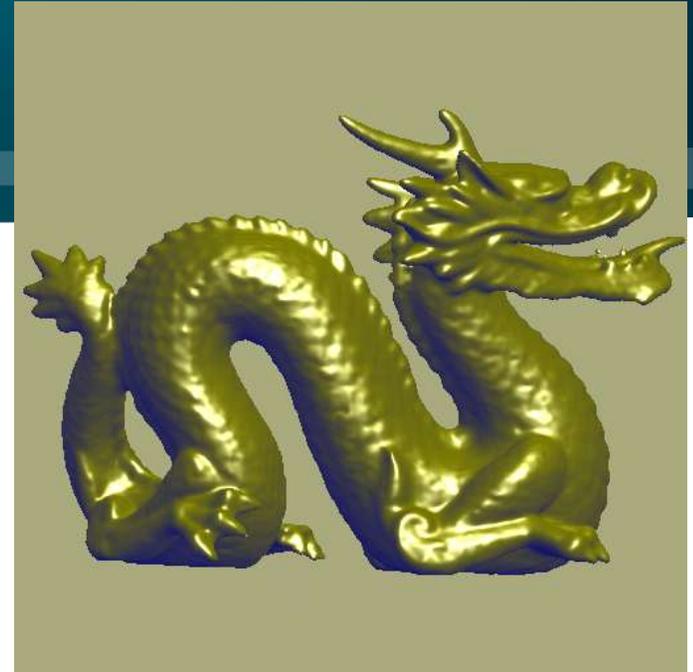
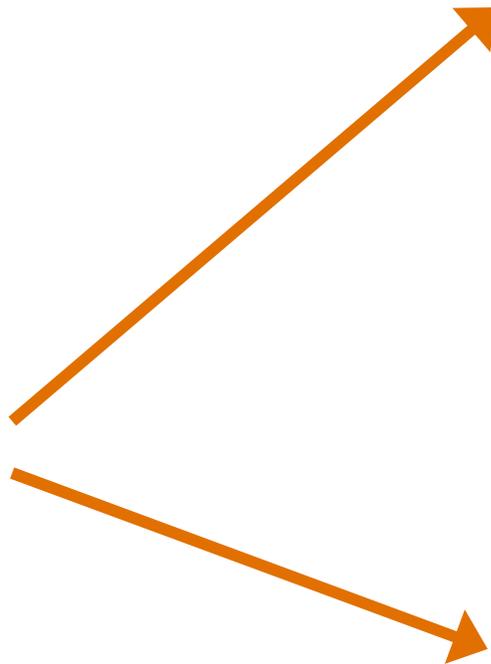
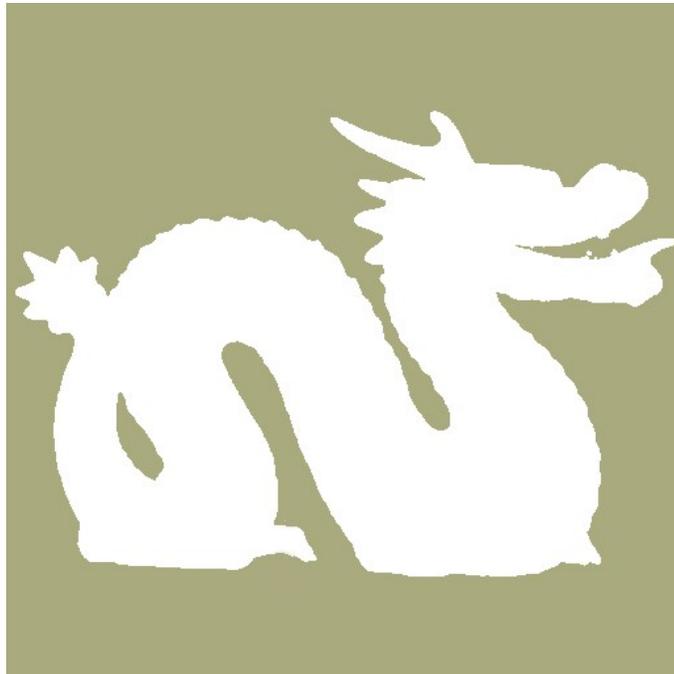
Post-processing steps:

- connect triangles (share vertices)
- compute normal vectors
 - by averaging triangle normals (problem: thin triangles!)
 - by estimating the gradient of the field $f(x_i)$ (better)

Iso-Surface / Volume Illumination

What About Volume Illumination?

Crucial for perceiving shape and depth relationships



this is a scalar volume (3D distance field)!

Local Illumination in Volumes



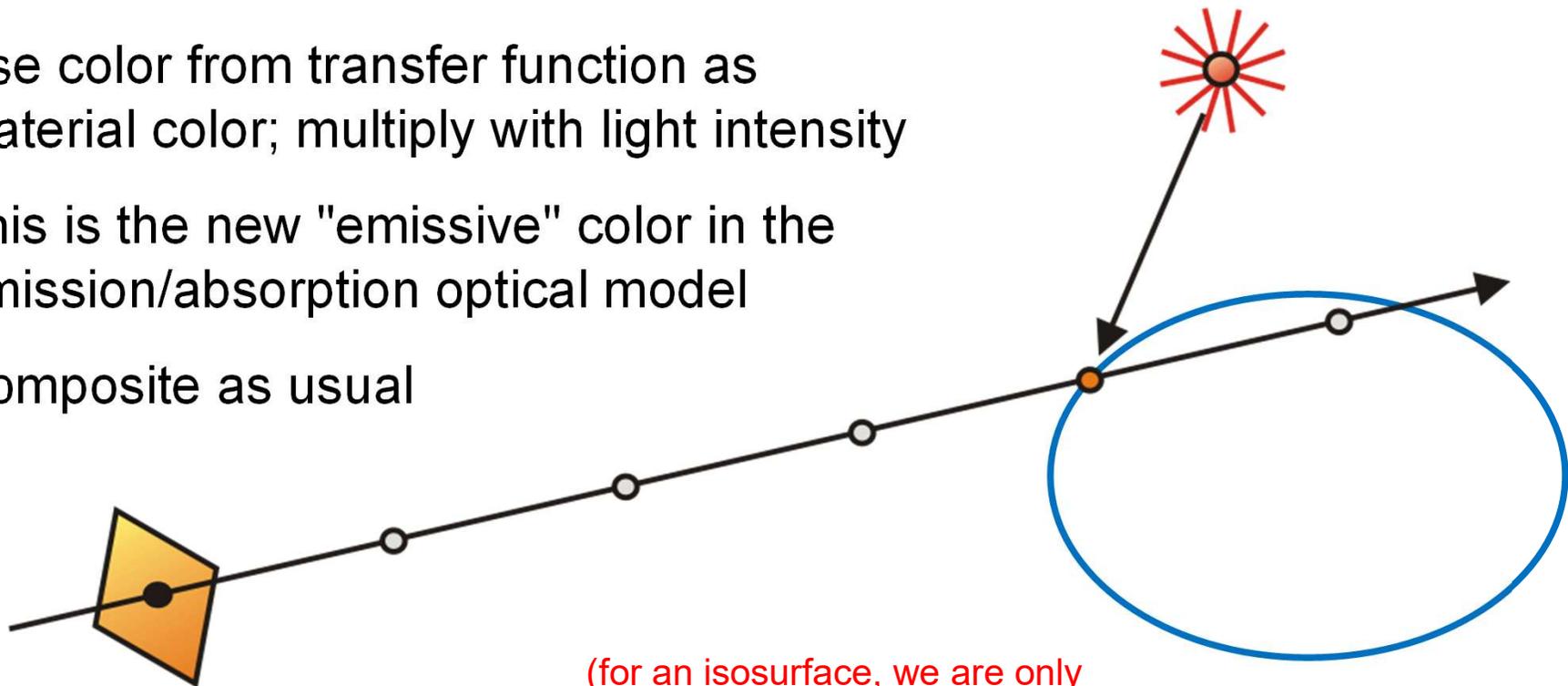
Interaction between light source and point in the volume

Local shading equation; evaluate at each point along a ray

Use color from transfer function as material color; multiply with light intensity

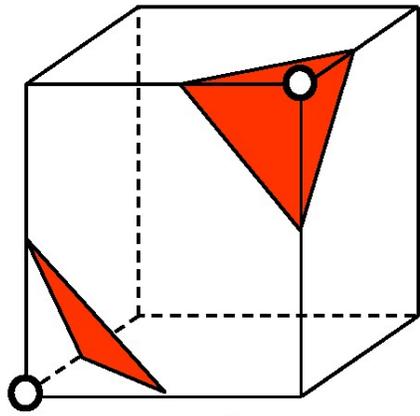
This is the new "emissive" color in the emission/absorption optical model

Composite as usual

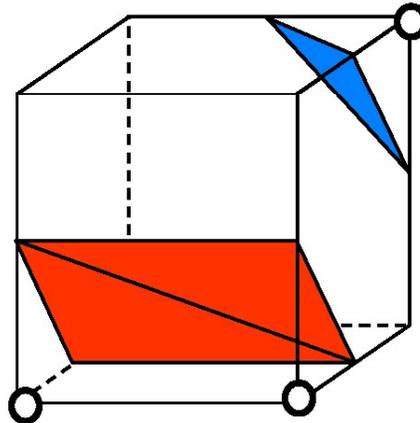


(for an isosurface, we are only interested in points *on* the surface; in marching cubes: the vertices)

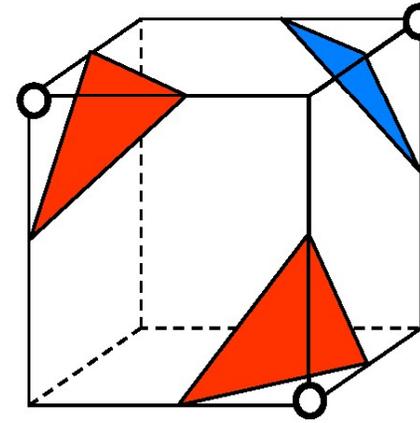
The marching cubes algorithm



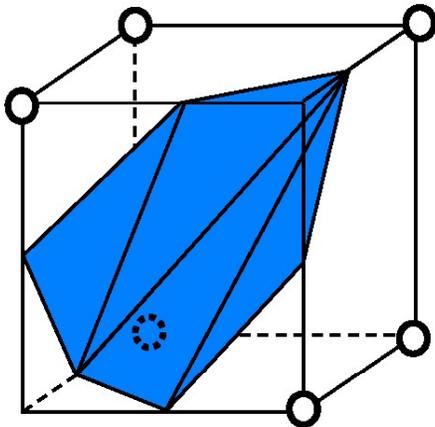
case 3



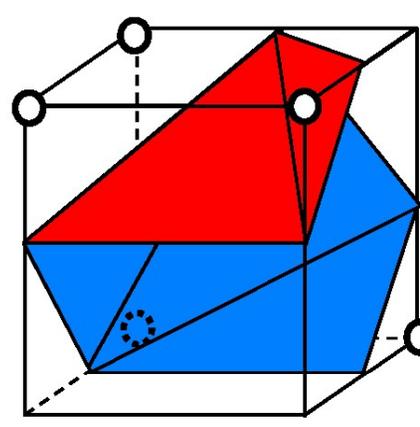
case 6



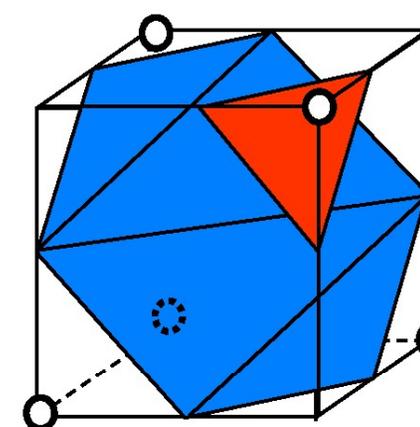
case 7



case 3c



case 6c

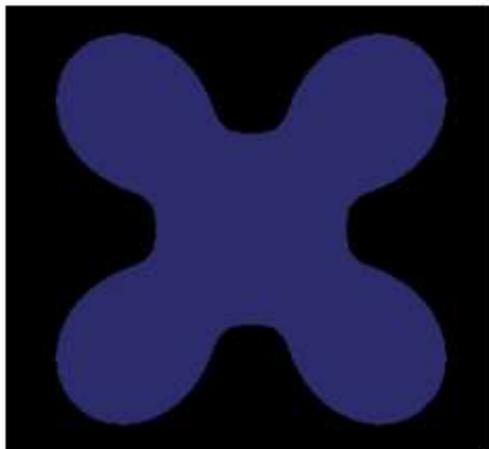


case 7c

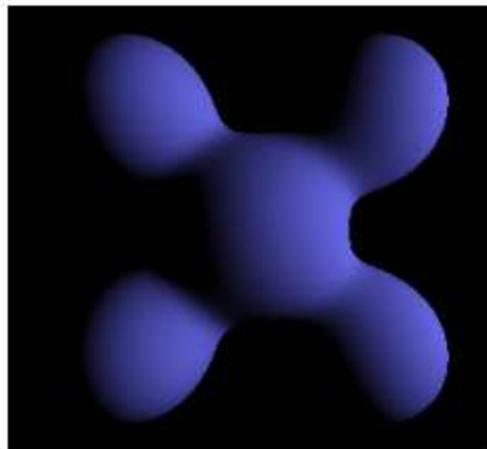
Local Illumination Model: Phong Lighting Model



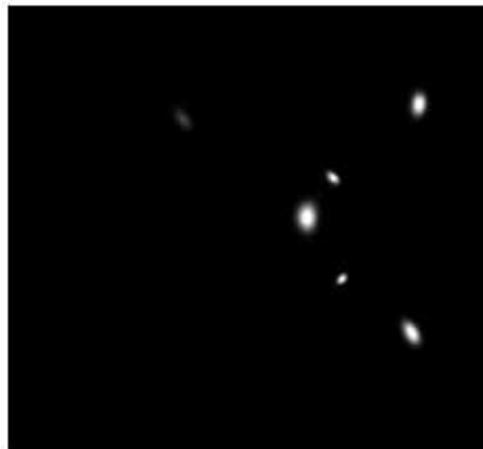
$$\mathbf{I}_{\text{Phong}} = \mathbf{I}_{\text{ambient}} + \mathbf{I}_{\text{diffuse}} + \mathbf{I}_{\text{specular}}$$



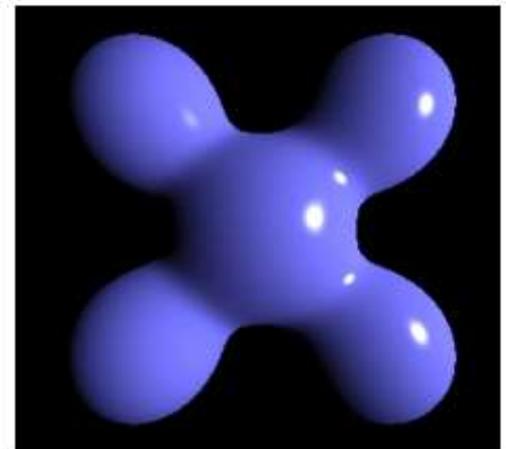
Ambient



Diffuse



Specular



= Phong Reflection

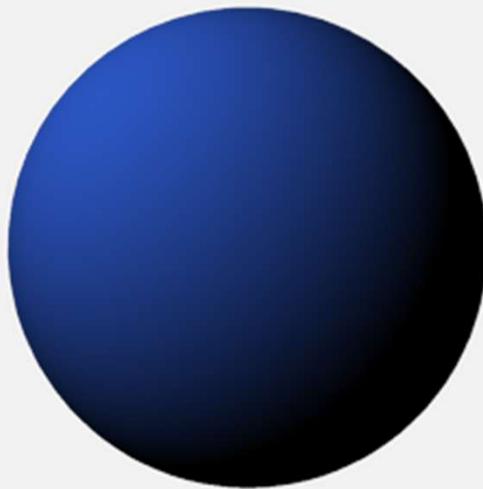
Local Illumination Model: Phong Lighting Model



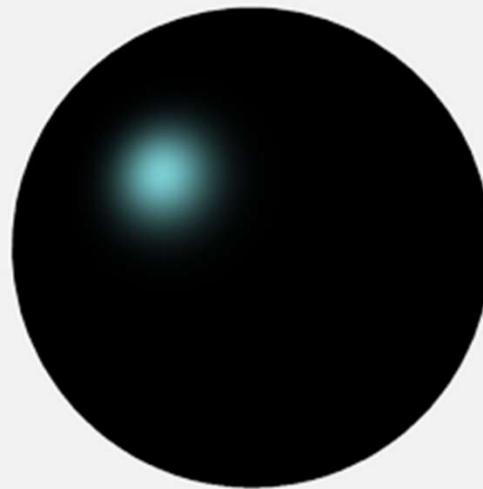
$$\mathbf{I}_{\text{Phong}} = \mathbf{I}_{\text{ambient}} + \mathbf{I}_{\text{diffuse}} + \mathbf{I}_{\text{specular}}$$



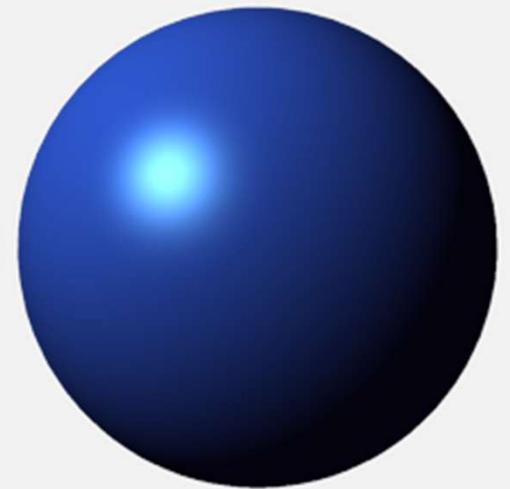
Ambient



Diffuse



Specular



Combined

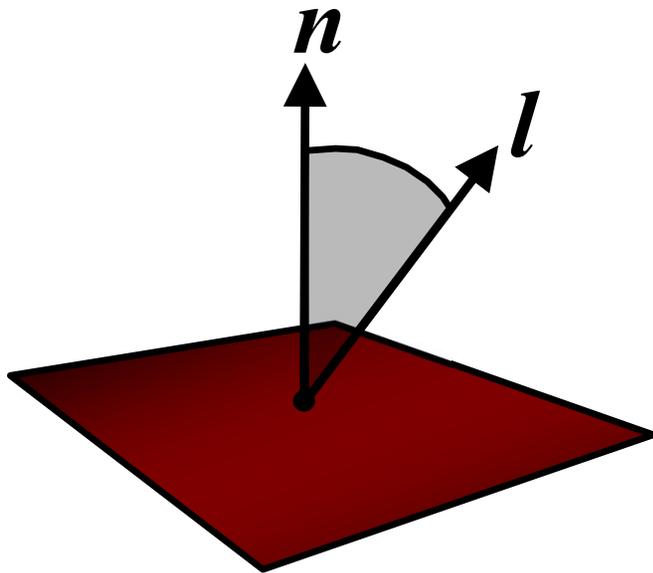
Local Shading Equations



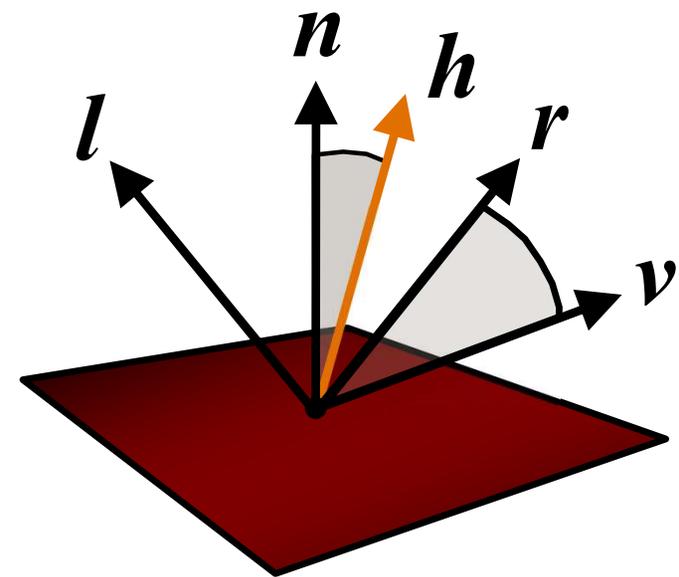
Standard volume shading adapts surface shading

Most commonly Blinn/Phong model

But what about the "surface" normal vector?



diffuse reflection



specular reflection

Local Illumination Model: Phong Lighting Model



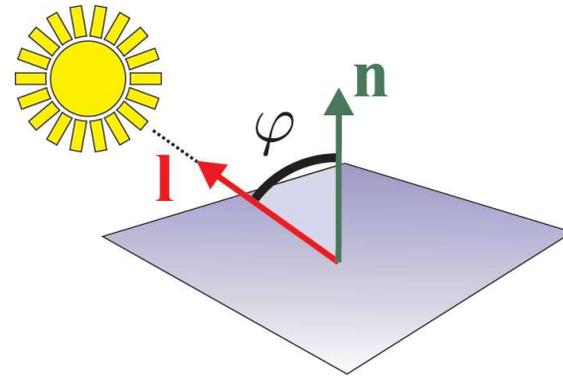
$$\mathbf{I}_{\text{Phong}} = \mathbf{I}_{\text{ambient}} + \mathbf{I}_{\text{diffuse}} + \mathbf{I}_{\text{specular}}$$

$$\mathbf{I}_{\text{ambient}} = k_a \mathbf{M}_a \mathbf{I}_a$$

Local Illumination Model: Phong Lighting Model



$$\mathbf{I}_{\text{Phong}} = \mathbf{I}_{\text{ambient}} + \mathbf{I}_{\text{diffuse}} + \mathbf{I}_{\text{specular}}$$



$$\begin{aligned}\mathbf{I}_{\text{diffuse}} &= k_d \mathbf{M}_d \mathbf{I}_d \cos \varphi \quad \text{if } \varphi \leq \frac{\pi}{2} \\ &= k_d \mathbf{M}_d \mathbf{I}_d \max((\mathbf{n} \cdot \mathbf{l}), 0)\end{aligned}$$

The Dot Product (Scalar / Inner Product)



Cosine of angle between two vectors times their lengths

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$$

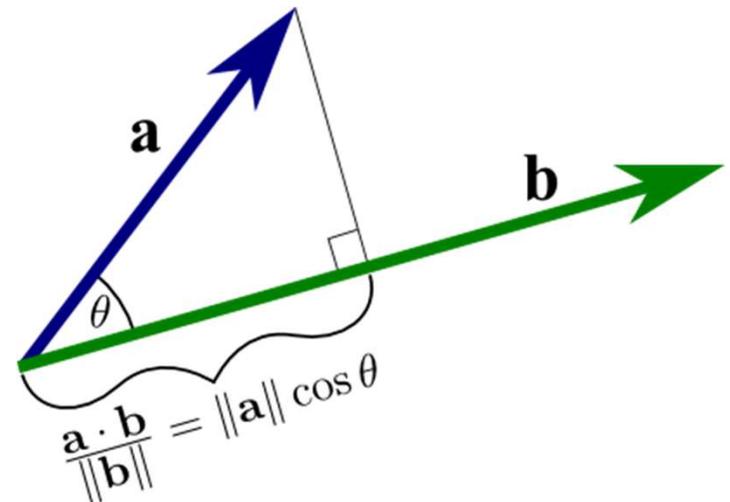
(geometric definition,
independent of coordinates)

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i$$

(standard inner product
in Cartesian coordinates)

Many uses:

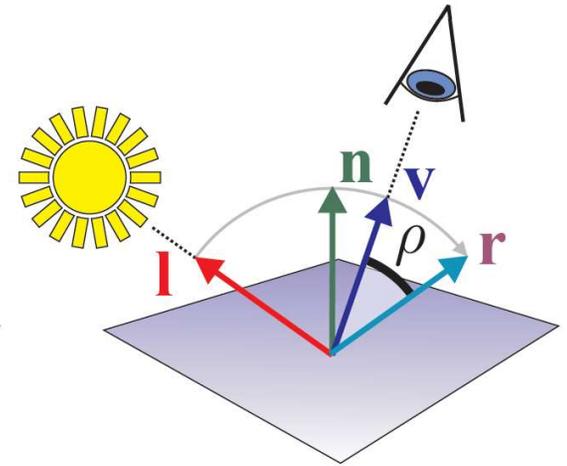
- Project vector onto another vector
- Project into basis (using the dual basis, see later)
- Project into tangent plane



Local Illumination Model: Phong Lighting Model



$$\mathbf{I}_{\text{Phong}} = \mathbf{I}_{\text{ambient}} + \mathbf{I}_{\text{diffuse}} + \mathbf{I}_{\text{specular}}$$



$$\mathbf{I}_{\text{specular}} = k_s \mathbf{M}_s \mathbf{I}_s \cos^n \rho, \quad \text{if } \rho \leq \frac{\pi}{2}$$

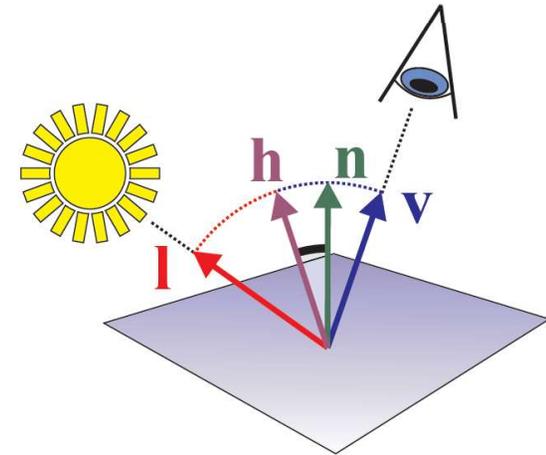
$$= k_s \mathbf{M}_s \mathbf{I}_s (\mathbf{r} \cdot \mathbf{v})^n$$

must also clamp!

Local Illumination Model: Phong Lighting Model



$$\mathbf{I}_{\text{Phong}} = \mathbf{I}_{\text{ambient}} + \mathbf{I}_{\text{diffuse}} + \mathbf{I}_{\text{specular}}$$



$$\mathbf{I}_{\text{specular}} \approx k_s \mathbf{M}_s \mathbf{I}_s (\mathbf{h} \cdot \mathbf{n})^n$$

$$\mathbf{h} = \frac{\mathbf{v} + \mathbf{l}}{\|\mathbf{v} + \mathbf{l}\|}$$

must also clamp!

half-way vector

The Gradient as Normal Vector



Gradient of the scalar field gives direction+magnitude of fastest change

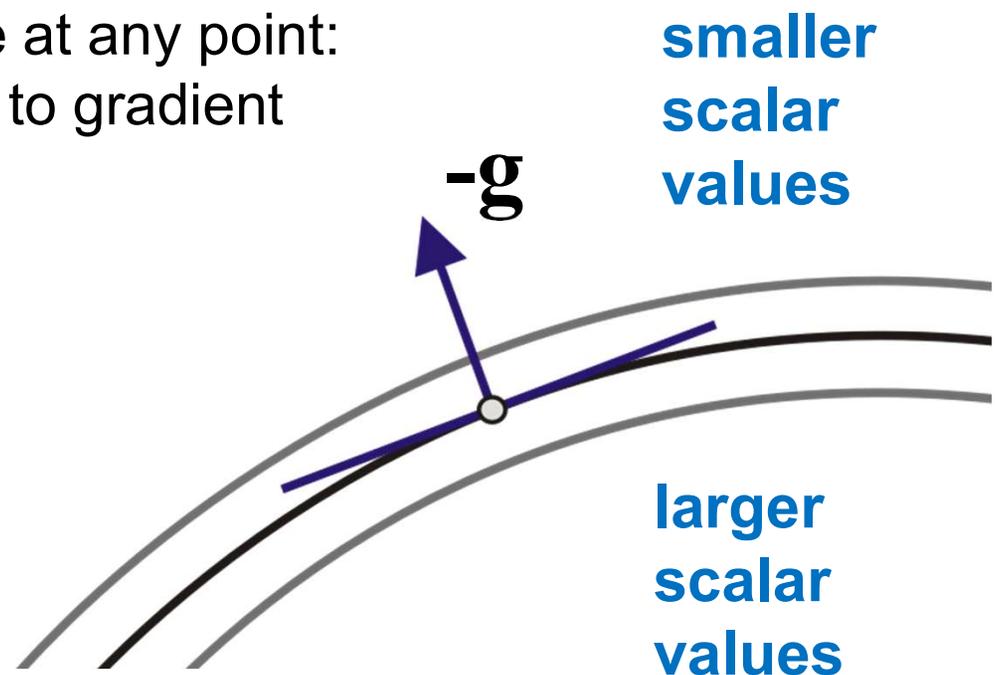
$$\mathbf{g} = \nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z} \right)^T$$

(only correct in Cartesian coordinates: see later)

Local approximation to isosurface at any point:
tangent plane = plane orthogonal to gradient

Normal of this isosurface:
normalized gradient vector
(negation is common convention)

$$\mathbf{n} = -\mathbf{g}/|\mathbf{g}|$$



The Dot Product (Scalar / Inner Product)



Cosine of angle between two vectors times their lengths

$$\mathbf{a} \cdot \mathbf{b} = \|\mathbf{a}\| \|\mathbf{b}\| \cos \theta$$

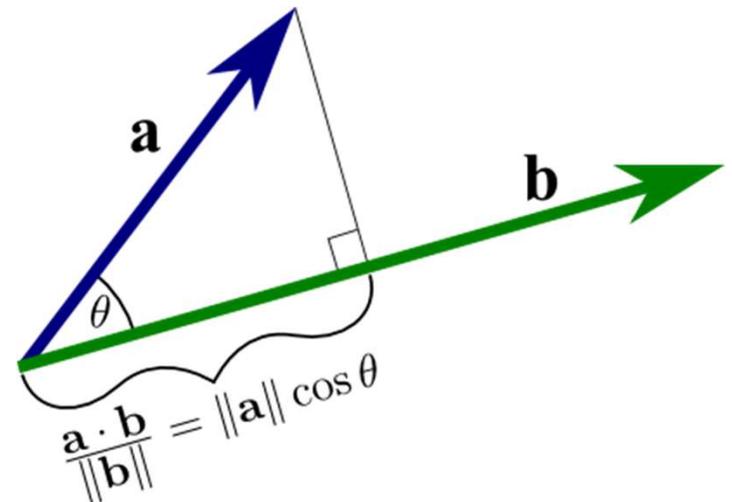
(geometric definition,
independent of coordinates)

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^n a_i b_i$$

(standard inner product
in Cartesian coordinates)

Many uses:

- Project vector onto another vector
- Project into basis (using the dual basis, see later)
- Project into tangent plane



Gradient and Directional Derivative



Gradient $\nabla f(x, y, z)$ of scalar function $f(x, y, z)$:

$$\nabla f(x, y, z) = \left(\frac{\partial f(x, y, z)}{\partial x}, \frac{\partial f(x, y, z)}{\partial y}, \frac{\partial f(x, y, z)}{\partial z} \right)^T$$

(only correct in Cartesian coordinates: see later)

Directional derivative in direction \mathbf{u} :

$$D_{\mathbf{u}}f(x, y, z) = \nabla f(x, y, z) \cdot \mathbf{u}$$

And therefore also:

$$D_{\mathbf{u}}f(x, y, z) = \|\nabla f\| \|\mathbf{u}\| \cos \theta$$

Gradient and Directional Derivative



Gradient $\nabla f(x, y, z)$ of scalar function $f(x, y, z)$:

$$\nabla f(x, y, z) = \left(\frac{\partial f(x, y, z)}{\partial x}, \frac{\partial f(x, y, z)}{\partial y}, \frac{\partial f(x, y, z)}{\partial z} \right)^T$$

(only correct in Cartesian coordinates: see later)

(Cartesian vector components; basis vectors not shown)

But: always need **basis vectors**! With Cartesian basis:

$$\nabla f(x, y, z) = \frac{\partial f(x, y, z)}{\partial x} \mathbf{i} + \frac{\partial f(x, y, z)}{\partial y} \mathbf{j} + \frac{\partial f(x, y, z)}{\partial z} \mathbf{k}$$

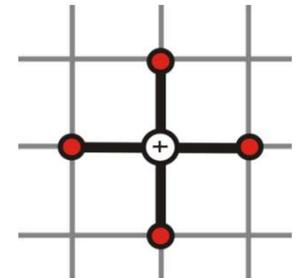
(Numerical) Gradient Reconstruction



We need to reconstruct the derivatives of a continuous function given as discrete samples

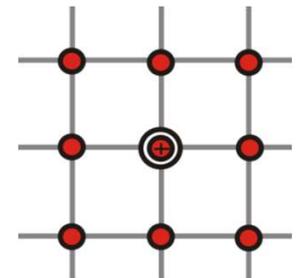
Central differences

- Cheap and quality often sufficient (2×3 neighbors in 3D)



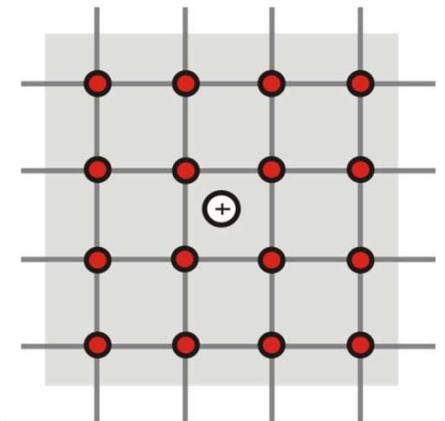
Discrete convolution filters on grid

- Image processing filters; e.g. Sobel (3^3 neighbors in 3D)



Continuous convolution filters

- Derived continuous reconstruction filters
- E.g., the cubic B-spline and its derivatives (4^3 neighbors)



Finite Differences



Obtain first derivative from Taylor expansion

$$\begin{aligned} f(x_0 + h) &= f(x_0) + \frac{f'(x_0)}{1!} h + \frac{f''(x_0)}{2!} h^2 + \dots \\ &= \sum_{n=0}^{\infty} \frac{f^{(n)}(x_0)}{n!} h^n. \end{aligned}$$

Forward differences / backward differences

$$f(x_0)' = \frac{f(x_0 + h) - f(x_0)}{h} + o(h)$$

$$f(x_0)' = \frac{f(x_0) - f(x_0 - h)}{h} + o(h)$$

Finite Differences



Central differences

$$f(x_0 + h) = f(x_0) + \frac{f'(x_0)}{1!} h + \frac{f''(x_0)}{2!} h^2 + o(h^3)$$

$$f(x_0 - h) = f(x_0) - \frac{f'(x_0)}{1!} h + \frac{f''(x_0)}{2!} h^2 + o(h^3)$$

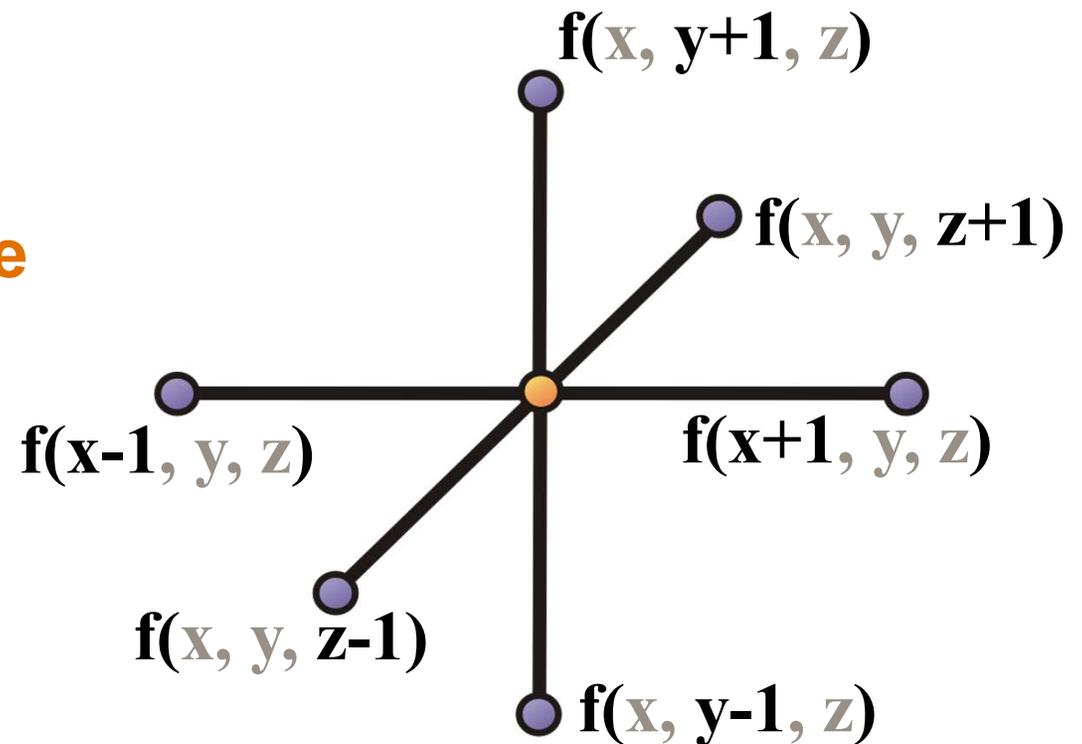
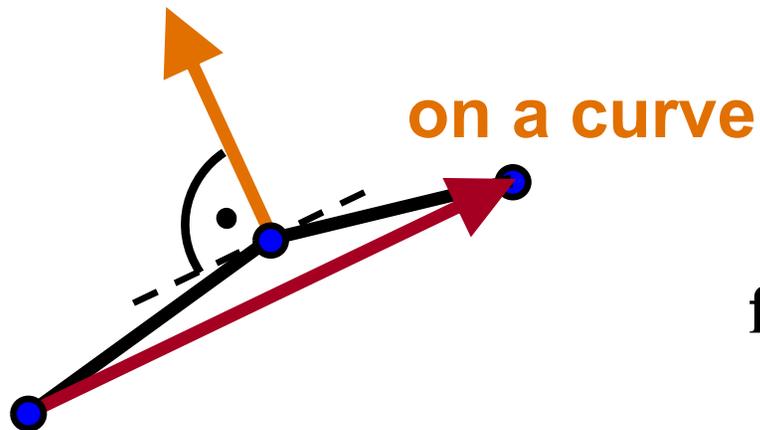
$$f'(x_0) = \frac{f(x_0 + h) - f(x_0 - h)}{2h} + o(h^2)$$

Central Differences



Need only two neighboring voxels per derivative

Most common method



$$g_x = 0.5 (f(x+1, y, z) - f(x-1, y, z))$$

$$g_y = 0.5 (f(x, y+1, z) - f(x, y-1, z))$$

$$g_z = 0.5 (f(x, y, z+1) - f(x, y, z-1))$$

in a volume

Gradients as Differential Forms (1-Forms)

The Gradient as a Differential Form



The gradient as a *differential* (differential 1-form) is the “primary” concept (also “total differential” or “total derivative”)

$$df = \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy + \frac{\partial f}{\partial z} dz$$

A differential 1-form is a scalar-valued linear function that takes a (direction) vector as input, and gives a scalar as output

Each of the 1-forms df, dx, dy, dz takes direction vector as input, gives scalar output

In the expression of the gradient df above, all 1-forms on the right-hand side get the same vector as input

df is simply a linear combination of the coordinate differentials dx, dy, dz

The Gradient as a Differential Form



The gradient as a *differential* (differential 1-form) is the “primary” concept (also “total differential” or “total derivative”)

$$df = \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy + \frac{\partial f}{\partial z} dz$$

The directional derivative and the gradient vector

$$\begin{aligned} D_{\mathbf{u}}f &= df(\mathbf{u}) \\ df(\mathbf{u}) &= \nabla f \cdot \mathbf{u} \end{aligned}$$

The gradient vector is then *defined*, such that:

$$\nabla f \cdot \mathbf{u} := df(\mathbf{u})$$

Thank you.

Thanks for material

- Helwig Hauser
- Eduard Gröller
- Daniel Weiskopf
- Torsten Möller
- Ronny Peikert
- Philipp Muigg
- Christof Rezk-Salama