Intrinsic scene properties

- Properties of a scene that are independent of lighting and view are called *intrinsic*.
- Two types of intrinsic properties that can be recovered from images:
  – **Shape** (object geometry)
  – **Reflectance** (light scattering)
- Capturing intrinsic properties leads to more realistic models
  – arbitrary views
  – arbitrary lighting

What we'll cover

- Some basic physics
  – radiometry&light reflection
  – Lambert's cosine law
  – polarization & color
- How to **recover** reflectance
  – accounting for Lambert's law
  – separating diffuse & specular
- How to **model** reflectance
  – lighting models
  – texture mapping

Shape & Shading

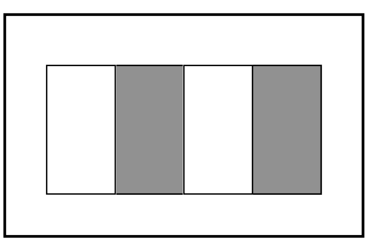
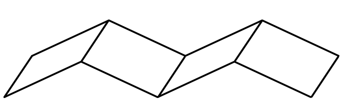
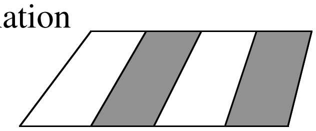


image of unknown object

Two possible interpretations



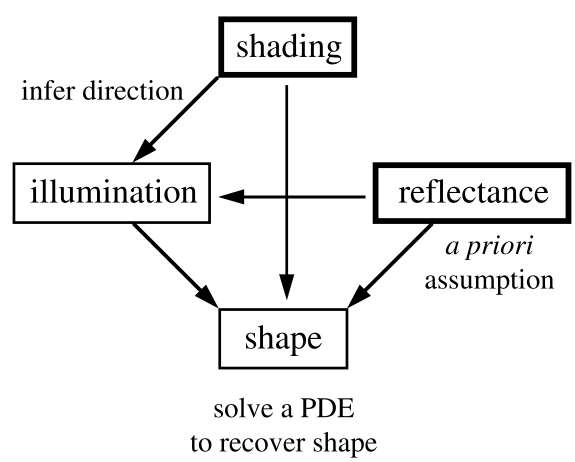
stripes due to illumination



stripes due to reflectance

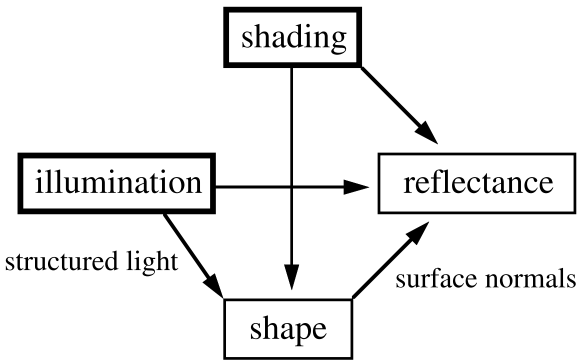
Shape & Shading

Recovering shape from shading and assumptions about reflectance:



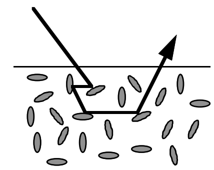
Shape & Shading

Recovering reflectance from shading and illumination.



Types of light reflection

Body reflection



Much absorption

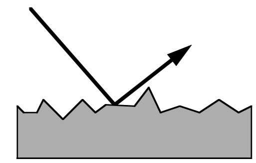
Relatively uniform

Weak dependence on incoming direction

Unpolarized

Color determined by material

Surface reflection



Possibly high intensity

Highly directional

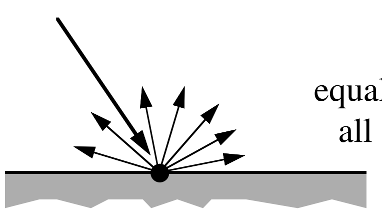
Strong dependence on incoming direction

Polarized

Color determined by light source

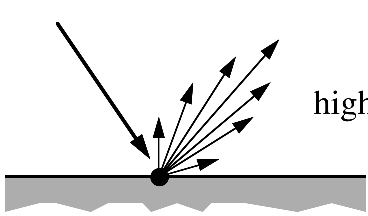
Diffuse & Specular

- Body reflection is frequently assumed to be *diffuse* (Lambertian).



equally bright in all directions

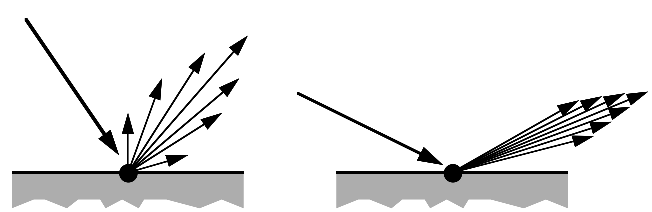
- Surface reflection is frequently assumed to be *specular*.



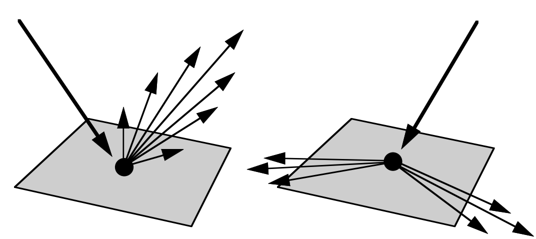
highly directional scattering

The specular component

- The distribution can depend strongly on the angle of incidence.

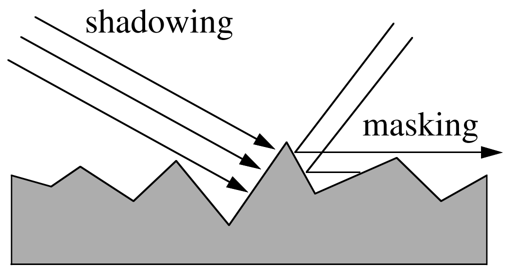


- For *anisotropic* materials, it even depend on the azimuthal angle.



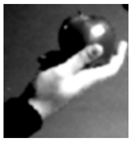
Torrance-Sparrow reflection model (1967)

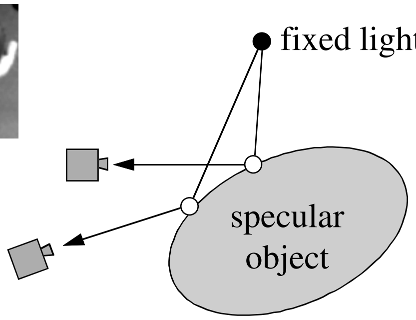
- Surface is comprised of specular micro-facets.
- Orientations are Gaussian distributed.
- Assume only one reflection.
- Accounts for masking & shadowing.



Applying surface appearance: Approximation 1 (decal)

- Map pixel intensities to the surface.
- Corresponds to fixed lighting, except for specular highlights.





Although Lambertian reflection remains fixed, specular highlights should move with viewer position.

Applying surface appearance: Approximation 2

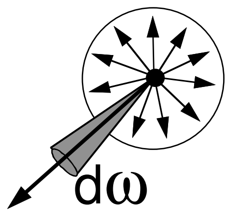
- Map pixel intensities to the surface after removing specular highlights.
- Add in view-dependent highlights.
- Corresponds to correct fixed lighting.
- The challenges are
  - How to remove highlights
  - How to recompute them later

Applying surface appearance: Approximation 3

- Identify surface reflectances (diffuse + specular).
- Compute light source dependent shading.
- Add in view-dependent highlights.
- Corresponds to arbitrary lighting.
- The new challenges are
  - How to compute diffuse reflectance
  - How to associate changing reflectances to the surface

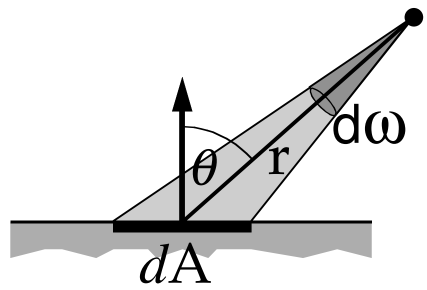
Power from a point light

Point light source
emitting P watts
into 4π steradians



Brightness given by
radiant intensity,
which is $P/4\pi$ watts/steradian

Area to solid angle

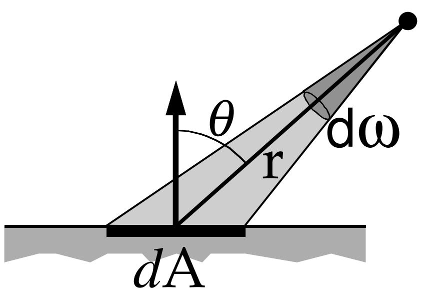


$$d\omega = \frac{\cos\theta}{r^2}\, dA$$

differential solid angle — differential area

Power from a point light

point light source, P watts



Lambert's cosine law

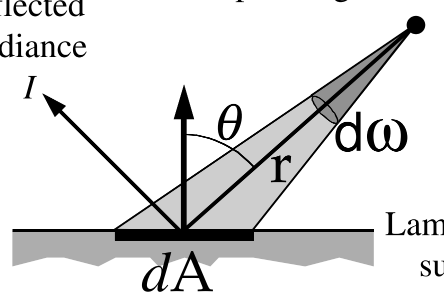
$$\frac{P}{4\pi}\, d\omega = \frac{P\cos\theta}{4\pi r^2}\, dA$$

power reaching differential area

inverse square law

Reflected radiance

point light source, P watts



reflectance

Lambert's cosine law

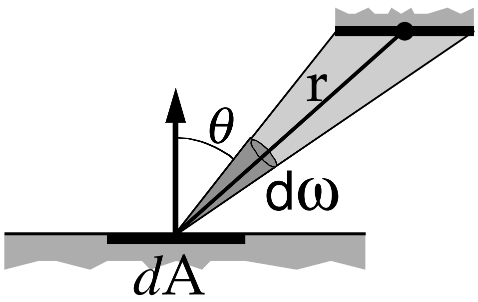
$$\rho\, \underbrace{\frac{P\, d\omega}{4\pi\, dA}} = \rho k \frac{\cos\theta}{r^2} = I$$

irradiance =
power per unit area
reaching surface

inverse square law

Power from an area light

area light source, radiance L



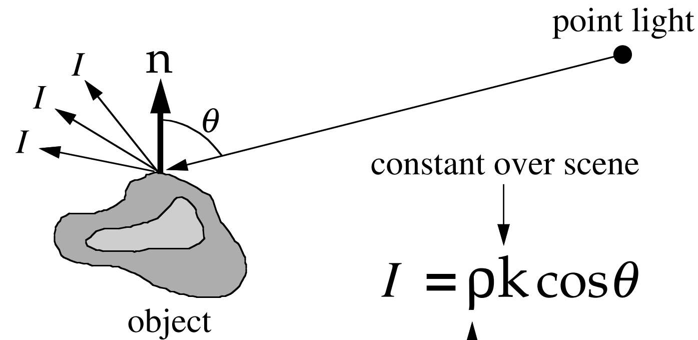
Lambert's cosine law

$$L \cos\theta \, d\omega \, dA$$

power reaching differential area

Simplifying assumptions

- Projector is a point light source.
- Projector is far from object relative to object size, so r^2 = constant.
- Surface is Lambertian + specular.

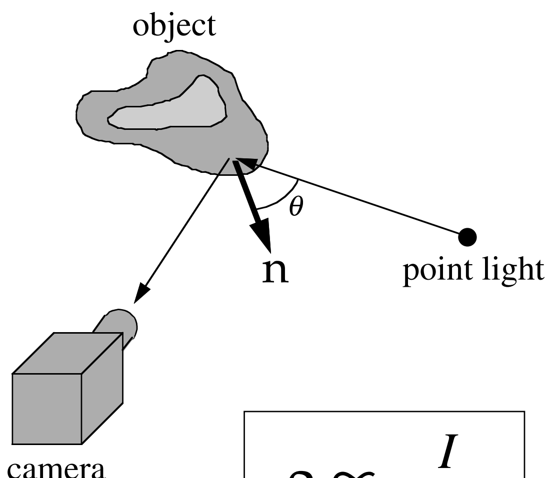


constant over scene

$$I = \rho k \cos\theta$$

constant over material (intrinsic property)

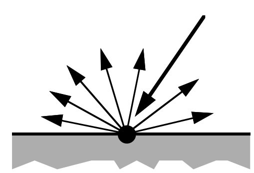
Recovering diffuse component



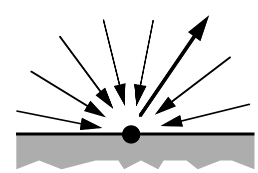
$$\boxed{\rho \propto \frac{I}{\cos\theta}}$$

Using ambiant lighting

The diffuse component can also be computed using another thermodynamic principle



directional-hemispherical reflection



hemispherical-directional reflection

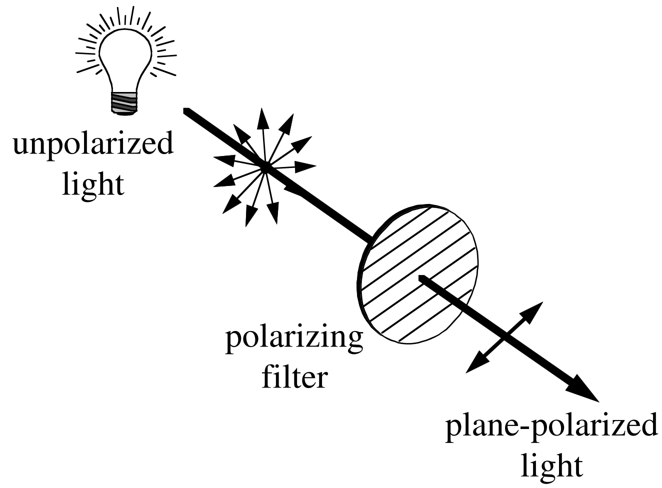
These functions are equal.

Therefore, for a Lambertian surface, the reflectance can be recovered by illuminating the object with uniform *ambiant* light.

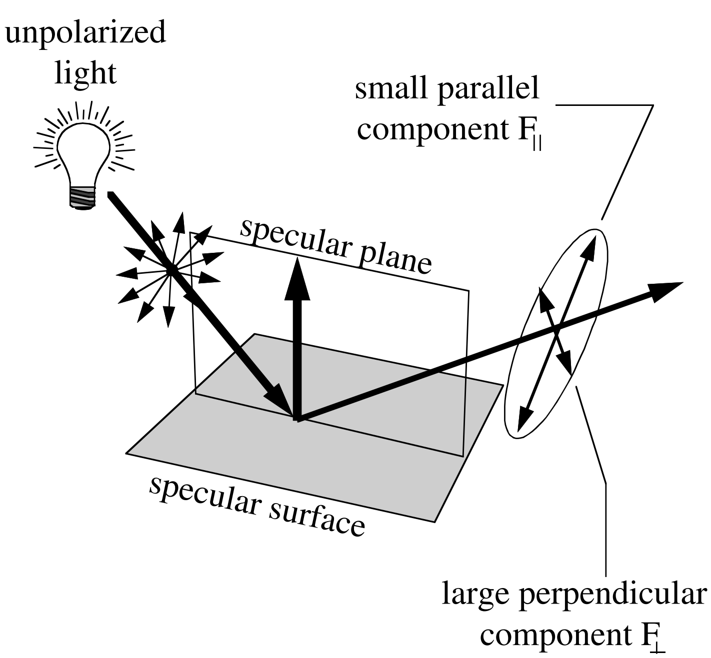
Separating diffuse & specular

- There are numerous ways to separate the diffuse and specular components
  - color
  - brightness
  - sharpness
  - polarization
  - view-dependence

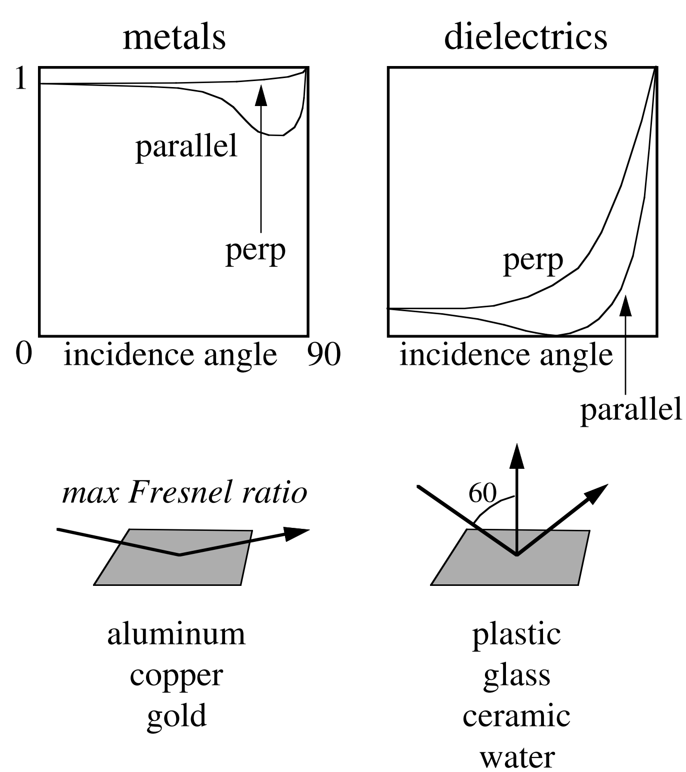
Polarized light



Polarized reflection



Polarization by metals and dielectrics



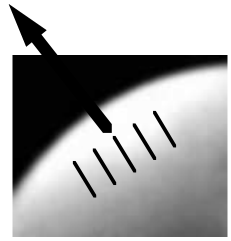
Polarization by a dielectric

Plastic sphere & three lights

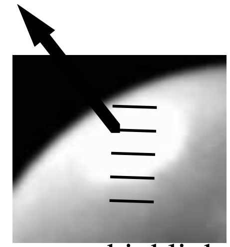
Polarizer with angle = 0

Polarizer with angle = 30

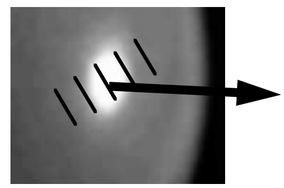
Closeups of highlights



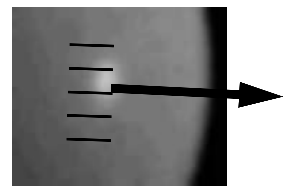
minimized highlight



strong highlight



strong highlight



minimized highlight

Finding all highlights

- Polarization varies as a cosine.
- Needn't find max and min.
- For each θ, the intensity I at each pixel satifies $\alpha \cos(\theta + \beta) = I$
- Three or four values of θ suffice.
- If specular >> diffuse, the latter can be ignored.
- Segment pixels into highlighs.
- Estimate specular cone.

Advantages of polarization

- Non-specular component invariant
- Constrains surface orientation
- Differentiates dielectrics & metals
- Local operation (per pixel)
- Works with ordinary lighting

Disadvantages of polarization

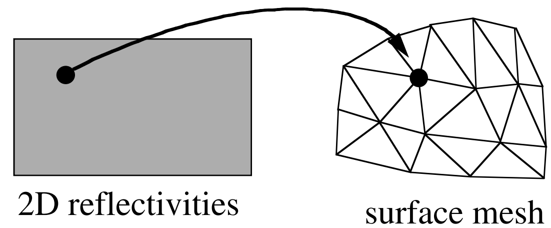
- Need three or four more images
- Doesn't work well with metals

Confounding factors

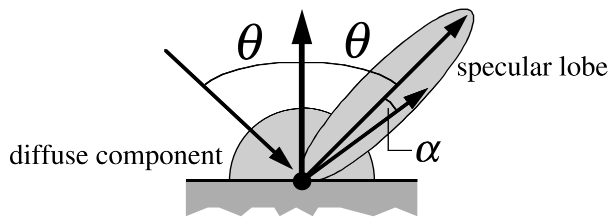
- Shadowing
- Non-ideal reflection
  - increased specular near grazing
  - retro-reflection
  - complex scattering functions
  - anisotropic reflection
- Inter-reflection
- Dynamic range of camera
- Different color spaces

Graphics tools for display

- Texture mapping

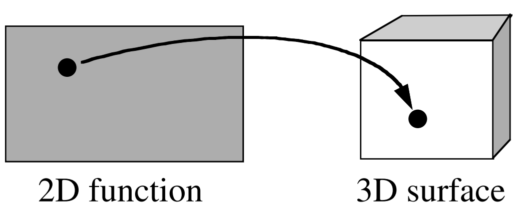


- Phong lighting $\cos^n \alpha$



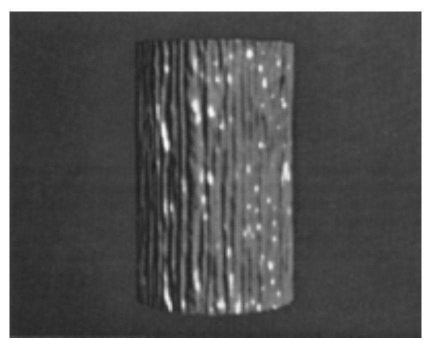
Texture mapping

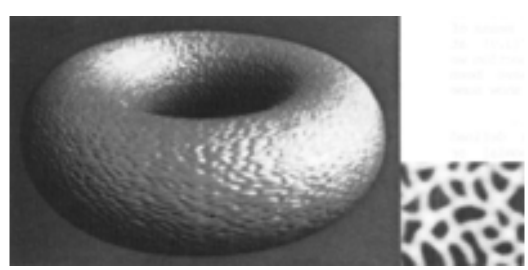
A texture map is a means of associating a 2D function with a surface in 3D.



The 2D function can represent
  color
  reflectivity
  displacement
  normal vector perturbation

Texture mapping by normal vector perturbation



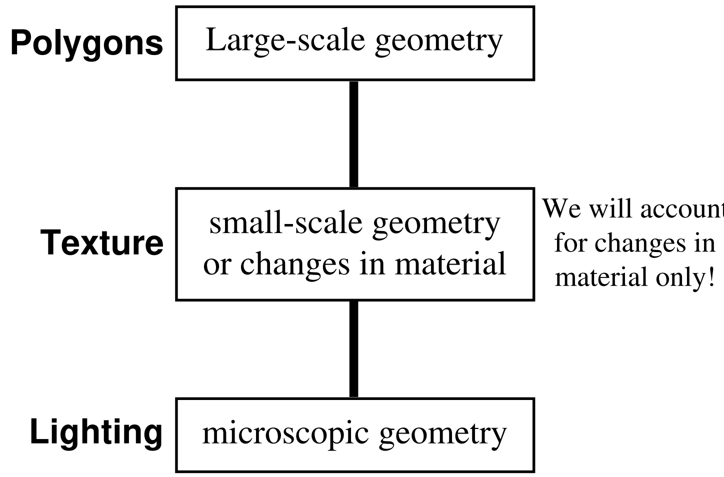


Lighting + Texture

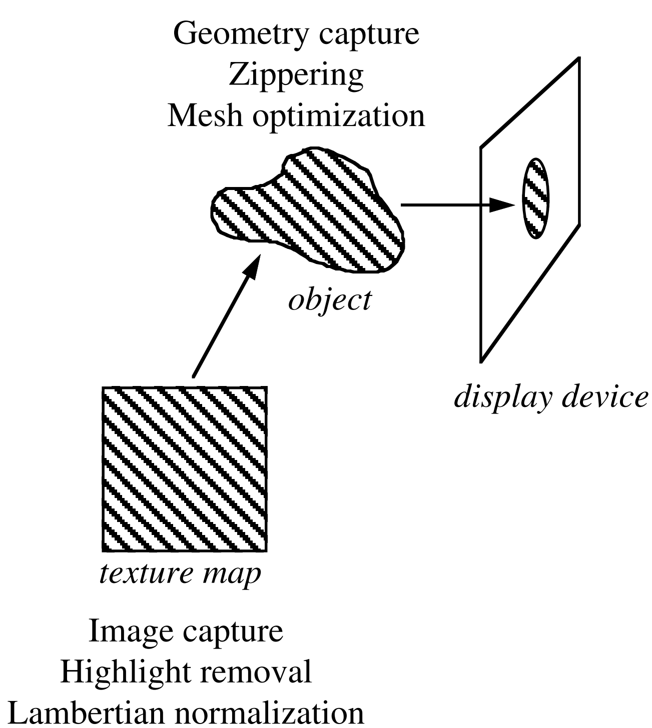
Lambertian surfaces & point light sources

Lambertian surfaces modulated by a texture

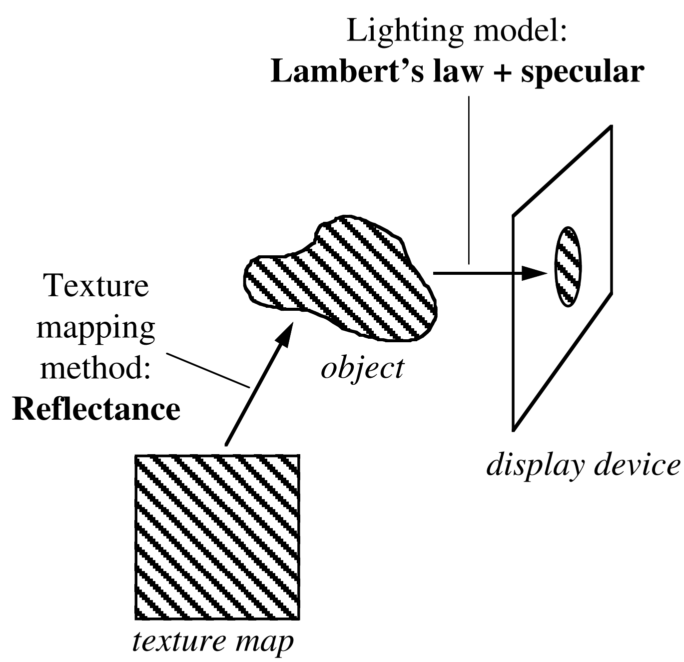
Textures and lighting



Rendering sequence



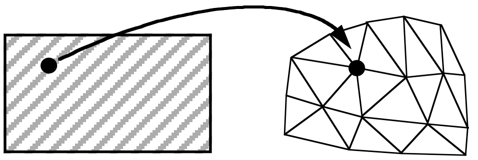
Rendering sequence



Texture mapping

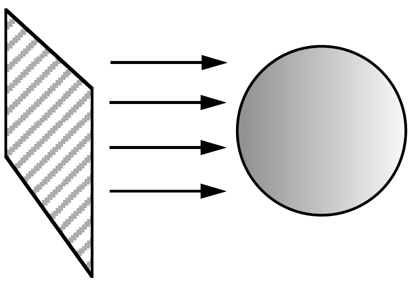
There are several ways to map the texture onto the surface:

Explicit texture coordinates



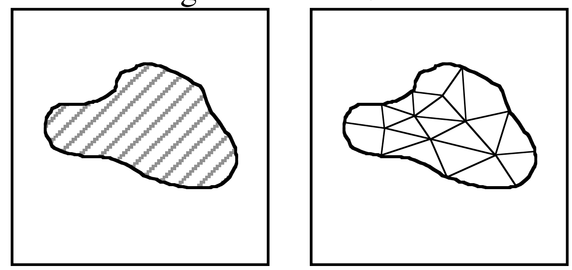
2D function surface mesh

Projection from a plane



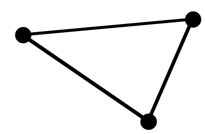
Texture coordinates

2D image 3D mesh



obtained with unstructured light

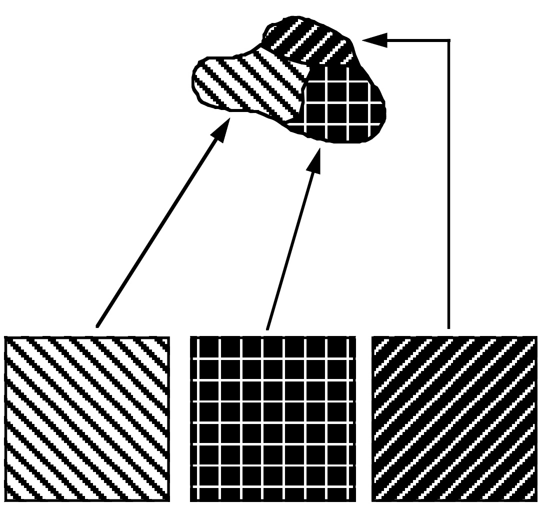
constructed from sequence of projected grids



Each vertex gets an (s,t) coordinate pair, which is the 2D image coordinate.

Multiple texture maps

object



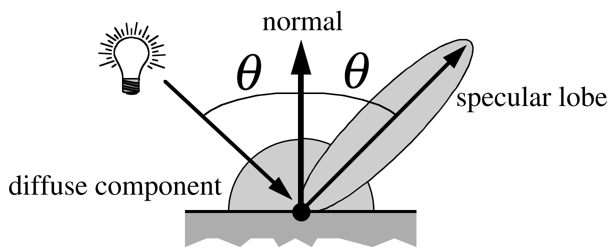
Texture maps corresponding to different views of the object. Must keep geometry segmented for this reason.

"Inventor" options

- What the texture controls
  - – color (**Decal**)
  - – diffuse reflectance (**Modulate**)
- How the mapping is done
  - – explicit coordinates
  - – plane projection

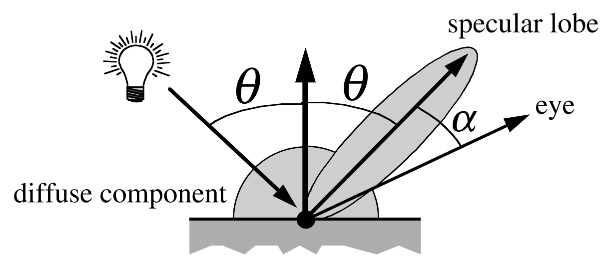
Phong reflection

- The Phong model of reflection is an empirical model that
  - simulates specular highlights
  - is part of most rendering packages
  - requires two parameters



Phong reflection

- The parameters control the color and the sharpness of the highlight.



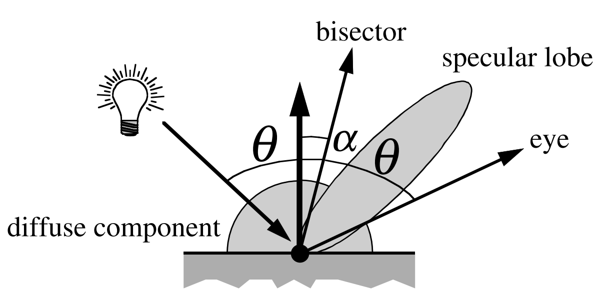
$$\mathbf{C}\cos^n\alpha$$

Phong exponent

Highlight color

Phong reflection

- An alternative formulation

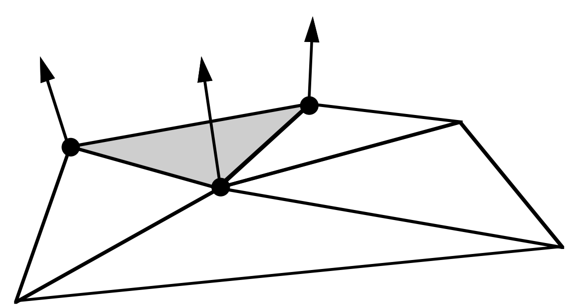


$$\mathbf{C}\cos^n\alpha$$

Phong exponent

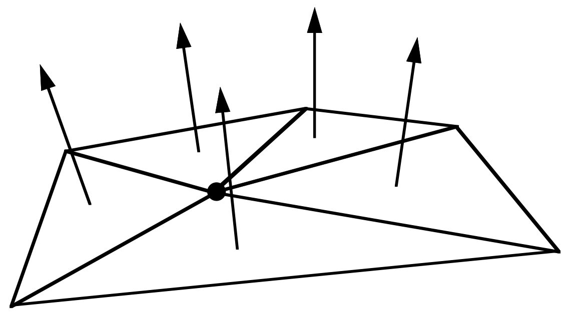
Highlight color

Gouraud interpolation



Phong reflection applied to each vertex, using associated normal vector. Resulting color is then interpolated.

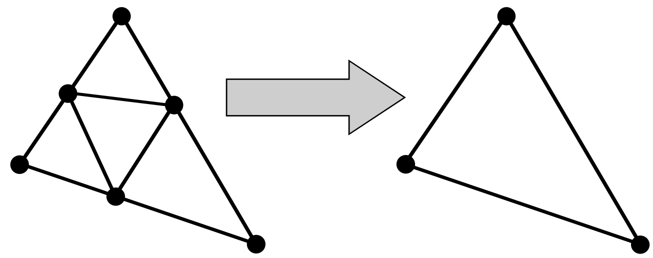
Vertex normal vectors



Normal vectors at the vertices are computed by averaging the normals of all the adjacent polygons.

Effects of mesh optimization

- Distortion of texture map
- Distortion of specular highlights



Levels of rendering

- Realistic rendering
  - evaluate shading per pixel
  - compute shadows
  - compute inter-reflection
  - allow for area light sources
  - Torrance-Sparrow lighting model
- Interactive rendering (e.g. Inventor)
  - Gouraud shading
  - point light sources
  - Phong lighting
- "Web" rendering (e.g. Java)
  - No native support for 3D