Note 09-Illumination Models

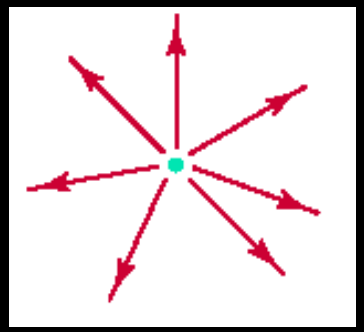
Metaverse

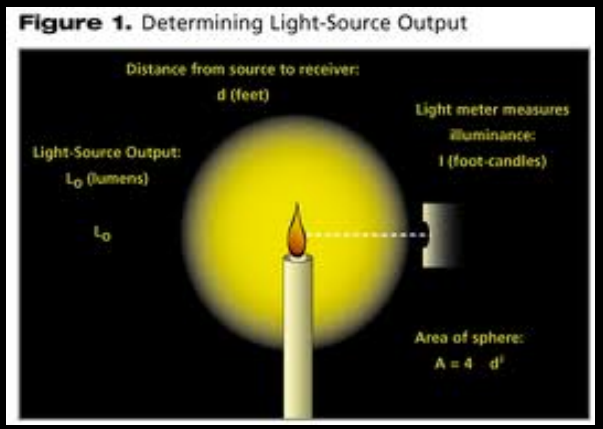
What We Will Learn

- Basic illumination models
- Polygon rendering methods
- Global illumination methods
  - Ray-tracing
  - Radiosity
- Texture mapping

Basics

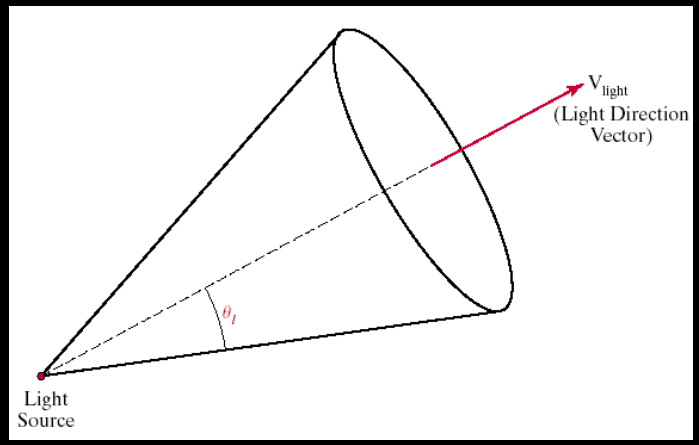
- Illumination models
  - How to calculate the color of an illuminated position on the surface of an object
- Light sources
  - Point light source
    - Position and color





Basics

- Light sources
  - Directional light source
    - Spotlight
    - Exclude outside objects



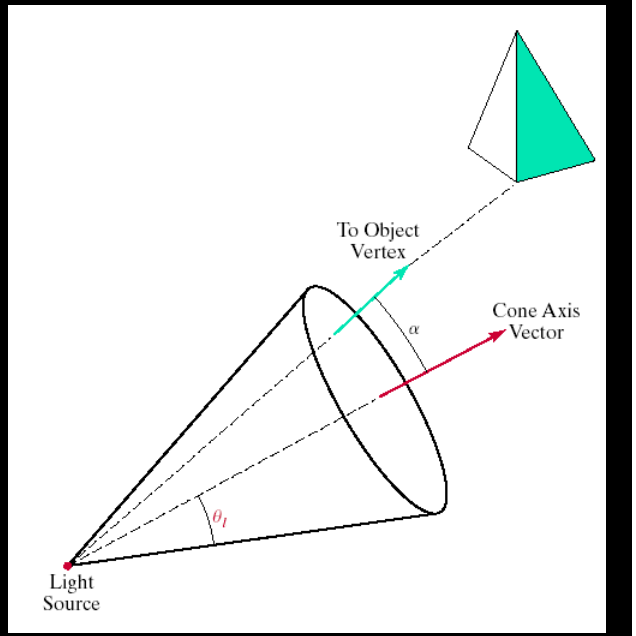
Attenuation

- Radial intensity attenuation (point light)

$$f_{\text{radatten}}(d_l) = \frac{1}{a_0 + a_1 d_l + a_2 d_l^2} \qquad (10\text{-}1)$$

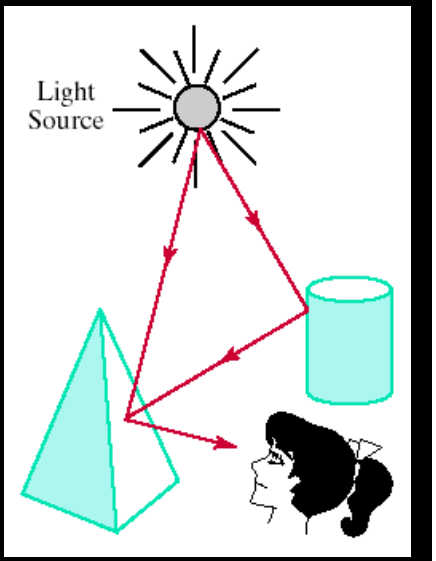
- Angular intensity attenuation (directional light)

$$f_{l,\text{angatten}} = \begin{cases} 1.0, & \text{if source is not a spotlight} \\ 0.0, & \text{if } \mathbf{V}_{\text{obj}} \cdot \mathbf{V}_{\text{light}} = \cos\alpha < \cos\theta_l \\ & \quad \text{(object is outside the spotlight cone)} \\ (\mathbf{V}_{\text{obj}} \cdot \mathbf{V}_{\text{light}})^{a_l}, & \text{otherwise} \end{cases} \qquad (10\text{-}5)$$



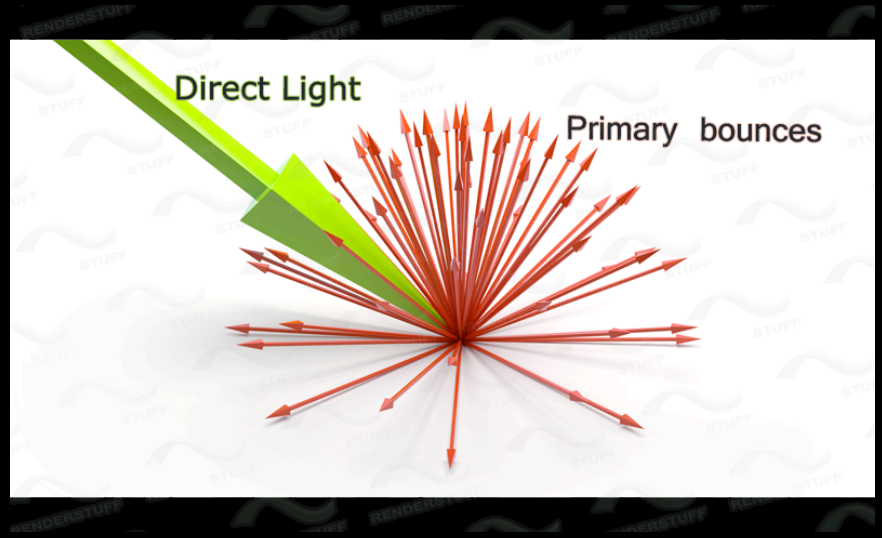
Basic Illumination Models

- Accurate surface-illumination calculations can be very expensive (exact global illumination)
- Ambient light (I_{amb})
  - Background light
  - Produced by the reflection from the various surfaces in the scene
  - Independent of a light position and orientation



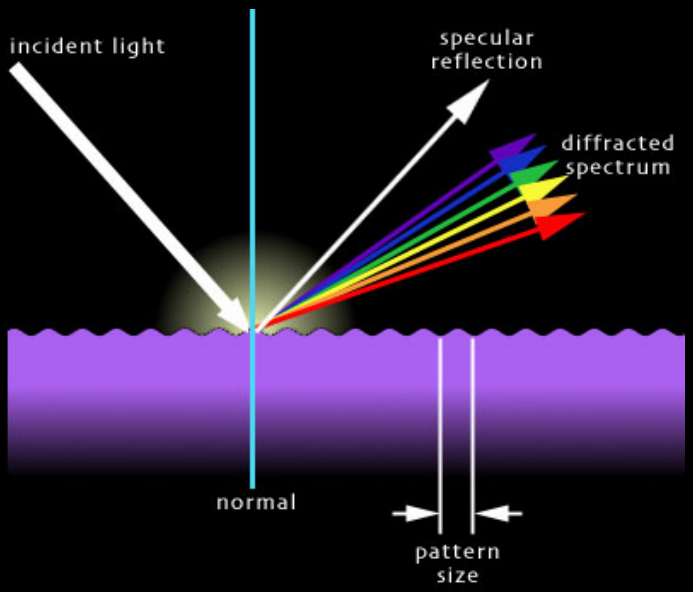
Basic Illumination Models

- Diffuse reflection (I_{diff})
  - Reflect the light in all directions

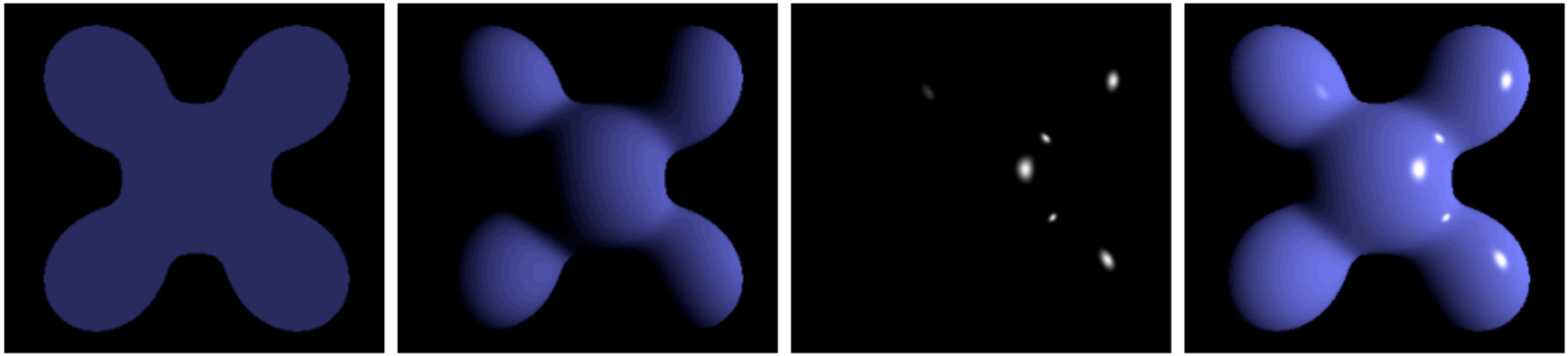


Basic Illumination Models

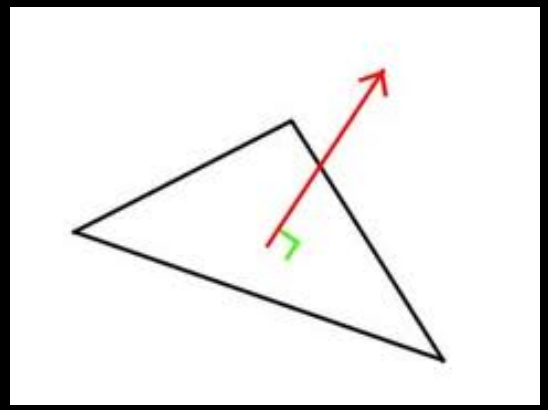
- Specular reflection (I_{spec})
  - Reflected light that is concentrated into a highlight

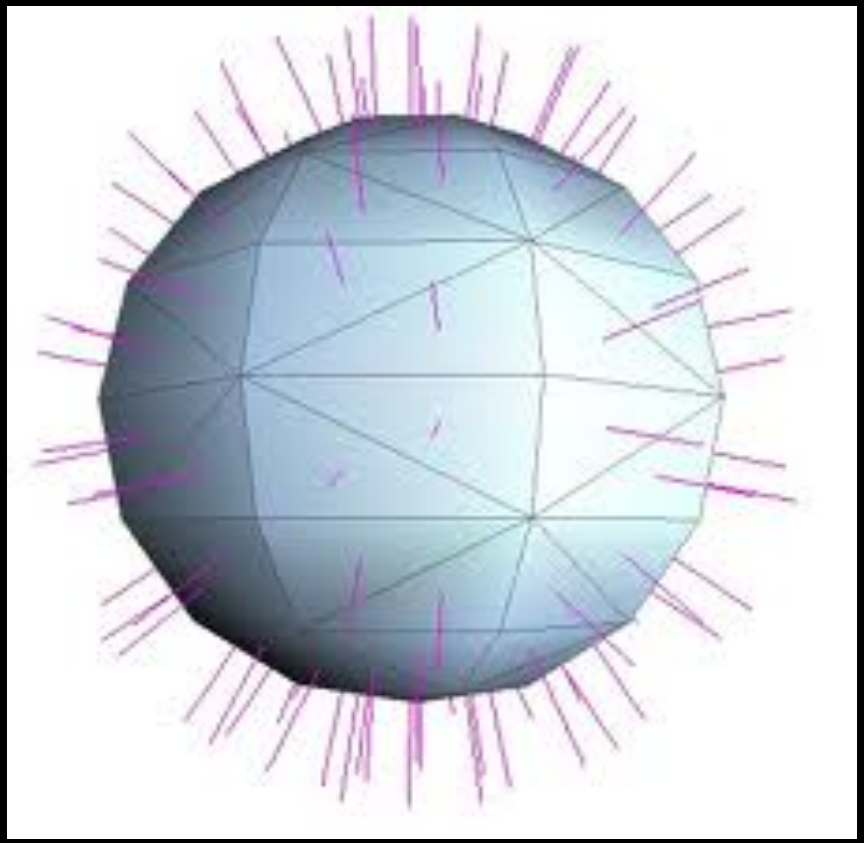


Basic Illumination Models



Normal Vector





Diffuse Reflection

- Lambertian reflector
  - The incident light is scattered with equal intensities in all directions (i.e., ideal diffuse reflector)

$$I_{\text{diff}} = \begin{cases} k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L}), & \text{if } \mathbf{N} \cdot \mathbf{L} > 0 \\ k_a I_a, & \text{if } \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases} \qquad (10\text{-}12)$$

k_a amb. refl. coeff.
I_a ambient intensity
k_d diffuse reflectivity
I_l light source intensity
$\boldsymbol{N}$ surface normal
$\boldsymbol{L}$ light source direction

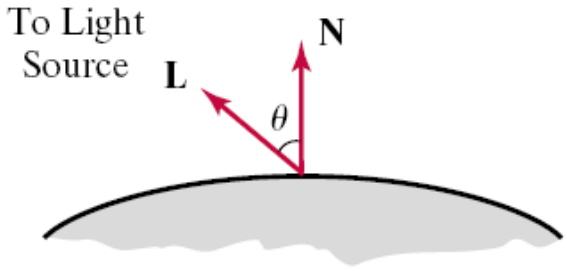
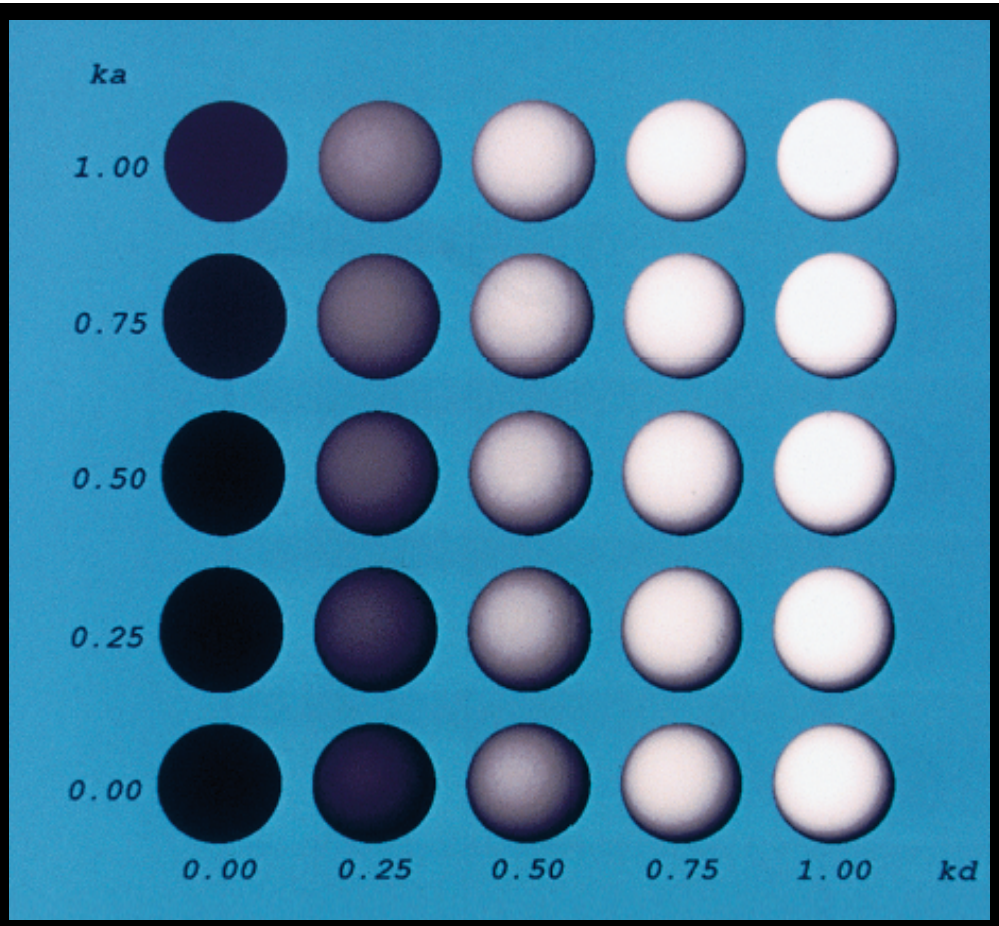


FIGURE 10–13 Angle of incidence θ between the unit light-source direction vector L and the unit normal vector N at a surface position.

Diffuse Reflection

$$I_{\text{diff}} = \begin{cases} k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L}), & \text{if } \mathbf{N} \cdot \mathbf{L} > 0 \\ k_a I_a, & \text{if } \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases} \qquad (10\text{-}12)$$

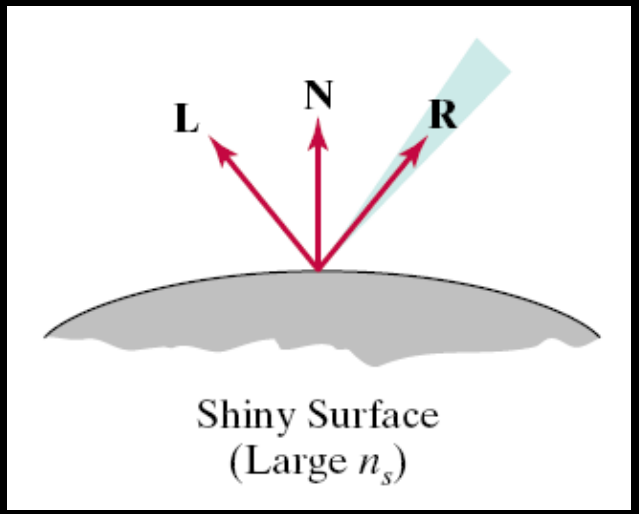


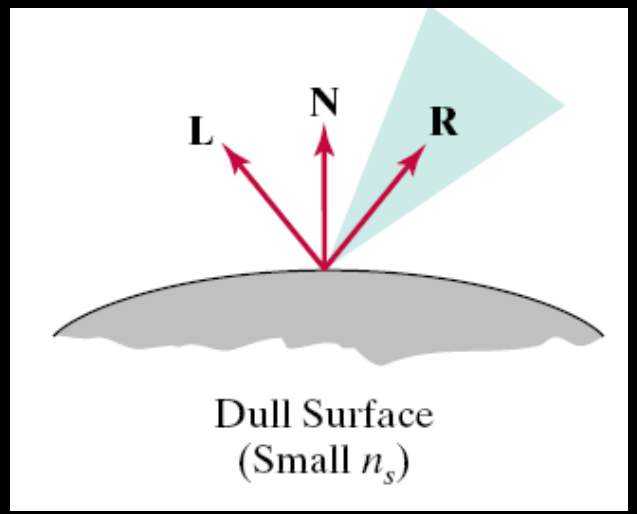
Specular Reflection

- Phong model (1973)
  - Empirical local illumination model for calculating the specular reflection

$$I_{l,\text{spec}} = \begin{cases} k_s I_l (\mathbf{V} \cdot \mathbf{R})^{n_s}, & \text{if } \mathbf{V} \cdot \mathbf{R} > 0 \quad \text{and} \quad \mathbf{N} \cdot \mathbf{L} > 0 \\ 0.0, & \text{if } \mathbf{V} \cdot \mathbf{R} < 0 \quad \text{or} \quad \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases} \qquad (10\text{-}14)$$

k_s spec. refl. coeff.
I_l light source intensity
n_s spec. refl. exponent
V view direction
R reflected light direction

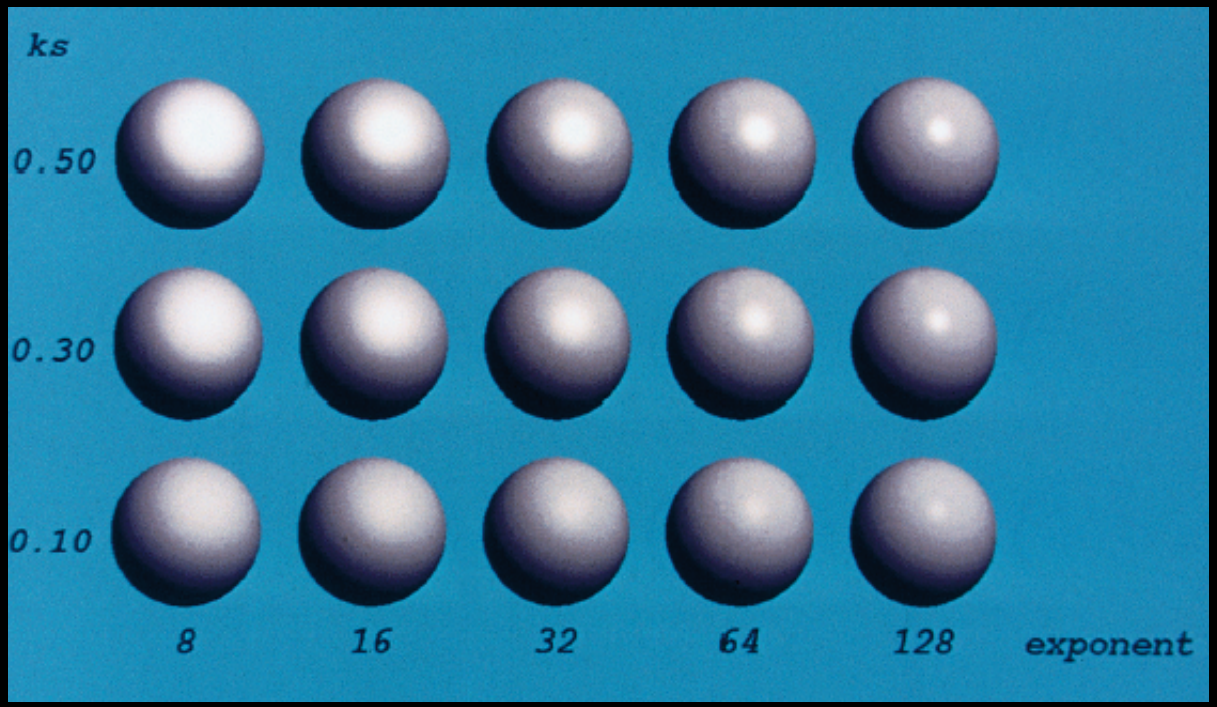




Specular Reflection

- Phong model (1973)

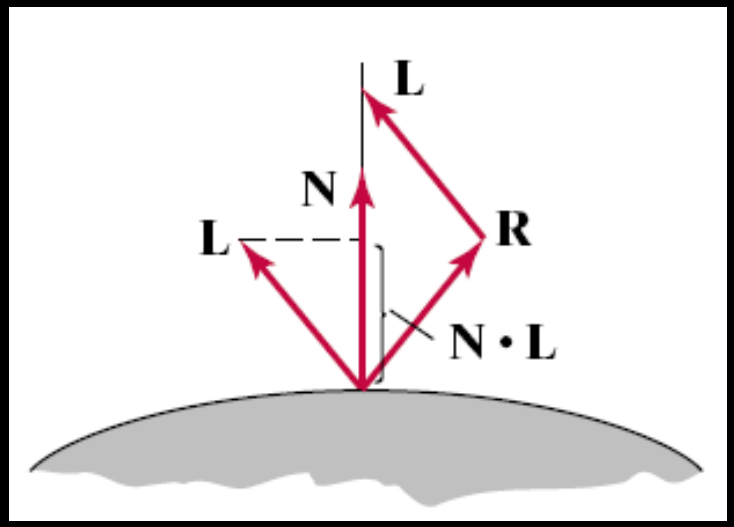
$$I_{l,\text{spec}} = \begin{cases} k_s I_l (\mathbf{V} \cdot \mathbf{R})^{n_s}, & \text{if } \mathbf{V} \cdot \mathbf{R} > 0 \quad \text{and} \quad \mathbf{N} \cdot \mathbf{L} > 0 \\ 0.0, & \text{if } \mathbf{V} \cdot \mathbf{R} < 0 \quad \text{or} \quad \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases} \qquad (10\text{-}14)$$

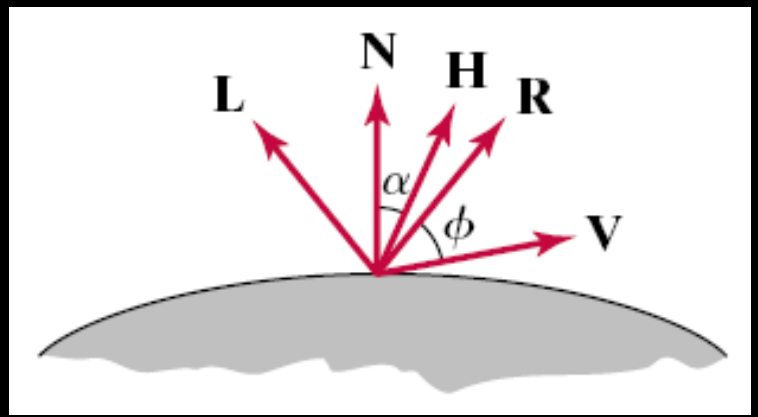


Specular Reflection

- Blinn's halfway vector (H)
  - Computing V•R can be expensive for non-planar surface
  - Replace V•R with N • H, where H=(L+V)/|L+V|

$$\mathbf{R} = (2\mathbf{N} \cdot \mathbf{L})\mathbf{N} - \mathbf{L}$$





The Phong Model

$$I = I_{\text{diff}} + I_{\text{spec}}$$
$$= k_a I_a + k_d I_l (\mathbf{N} \cdot \mathbf{L}) + k_s I_l (\mathbf{N} \cdot \mathbf{H})^{n_s} \qquad \textit{(10-17)}$$

$$I = I_{\text{surfemission}} + I_{\text{ambdiff}} + \sum_{l=1}^{n} f_{l,\text{radatten}} f_{l,\text{angatten}} (I_{l,\text{diff}} + I_{l,\text{spec}}) \qquad \textit{(10-19)}$$

$$I_{l,\text{diff}} = \begin{cases} 0.0, & \text{if } \mathbf{N} \cdot \mathbf{L}_l \leq 0.0 \text{ (light source behind object)} \\ k_d I_l (\mathbf{N} \cdot \mathbf{L}_l), & \text{otherwise} \end{cases} \qquad \textit{(10-20)}$$

$$I_{l,\text{spec}} = \begin{cases} 0.0, & \text{if } \mathbf{N} \cdot \mathbf{L}_l \leq 0.0 \\ & \text{(light source behind object)} \\ k_s I_l \max\{0.0, (\mathbf{N} \cdot \mathbf{H}_l)^{n_s}\}, & \text{otherwise} \end{cases} \qquad \textit{(10-21)}$$

Polygon Shading Methods

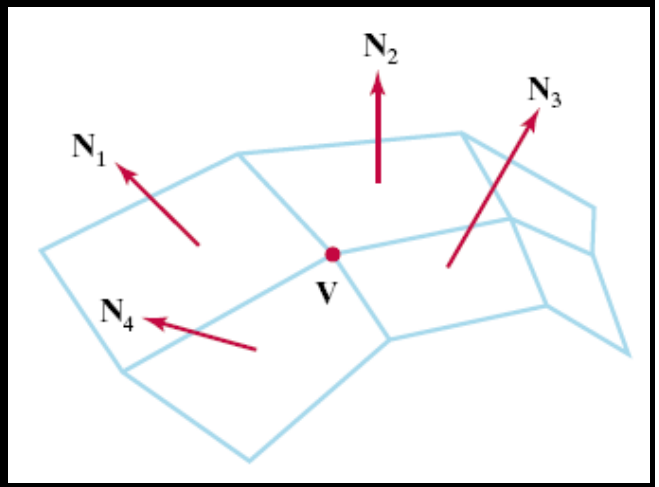
- Apply the illumination model to a few selected points and interpolate the intensity at the other surface positions during rasterization (local illumination model)

1. Flat shading
   - Assign the same color to all surface positions
2. Gouraud shading
   - Intensity interpolation
3. Phong shading
   - Normal interpolation

Gouraud Shading

1. Determine the normal vector at each vertex
   E.g.,

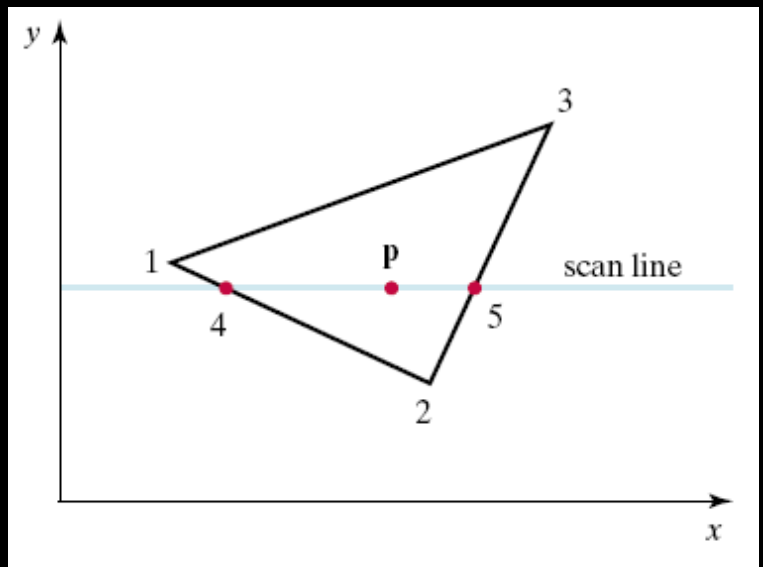
$$\mathbf{N}_V = \frac{\sum_{k=1}^{n} \mathbf{N}_k}{\left|\sum_{k=1}^{n} \mathbf{N}_k\right|}$$



2. Apply an illumination model at each vertex
3. Linearly interpolate the intensities over the entire polygon

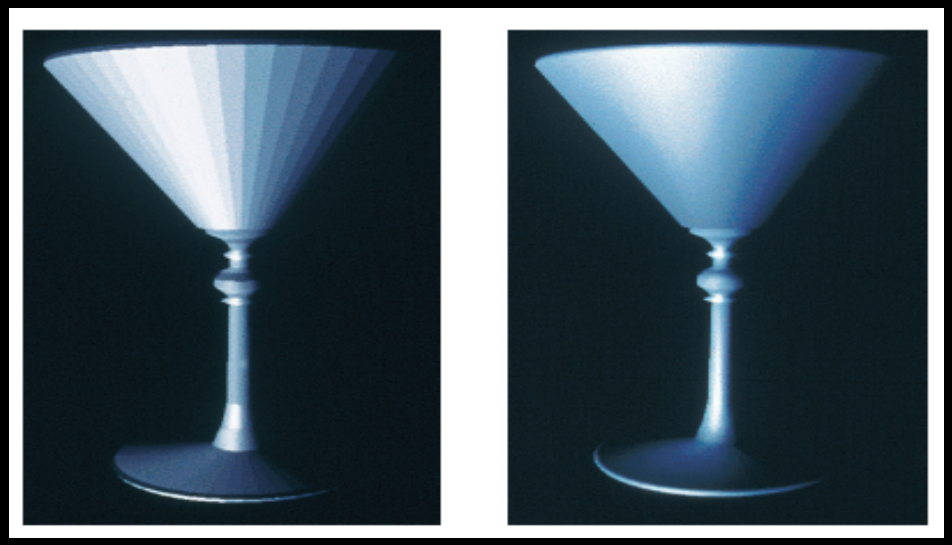
$$I_4 = \frac{y_4 - y_2}{y_1 - y_2} I_1 + \frac{y_1 - y_4}{y_1 - y_2} I_2 \qquad (10\text{-}52)$$

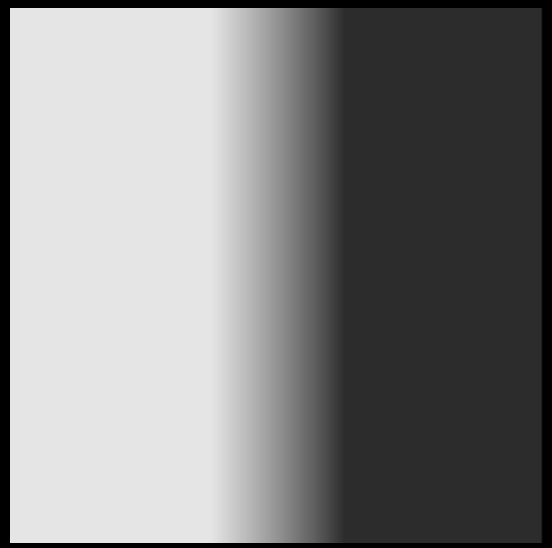
$$I_p = \frac{x_5 - x_p}{x_5 - x_4} I_4 + \frac{x_p - x_4}{x_5 - x_4} I_5 \qquad (10\text{-}53)$$



Gouraud Shading

- Flat VS Gouraud
- Gouraud shading is supported in OpenGL (a.k.a. smooth shading)
- Problems with Gouraud shading
  - Mach bands
  - Missing, anomalous highlights

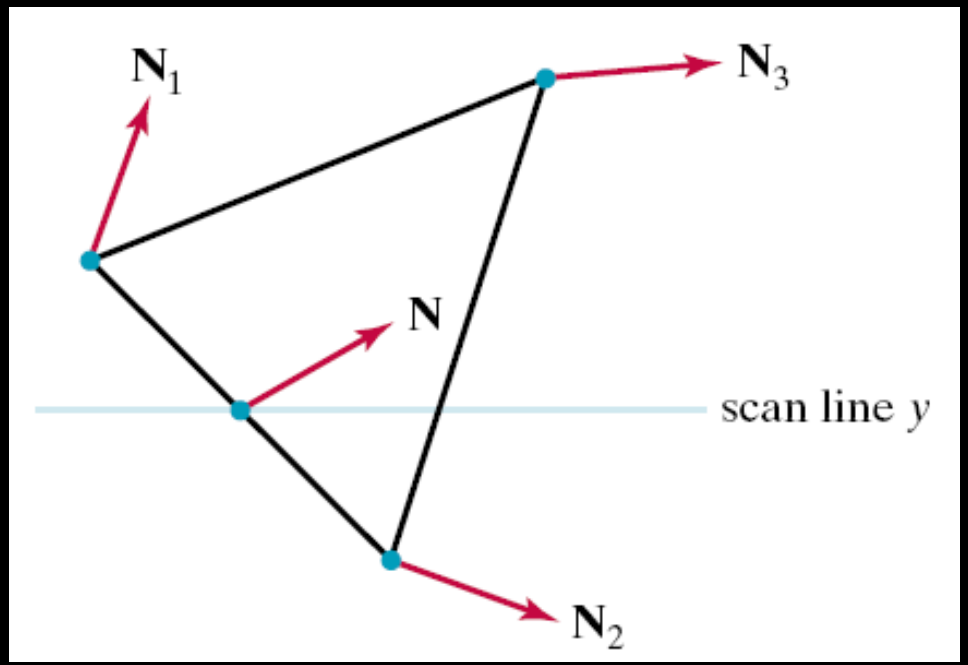




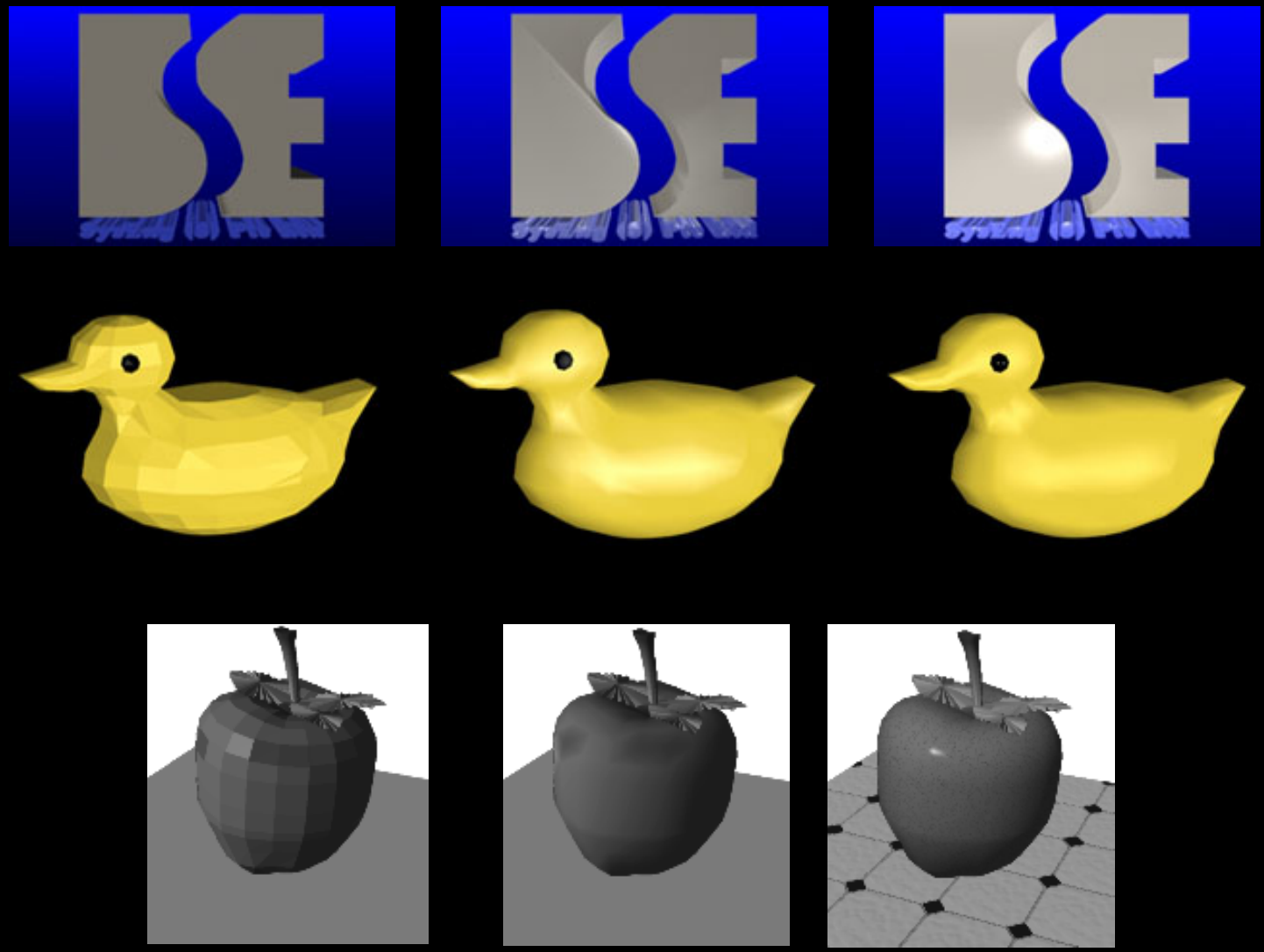
Phong Shading

1. Determine the normal vector at each vertex
2. Linearly interpolate the vertex normals over the entire polygon
3. Apply an illumination model over the entire polygon

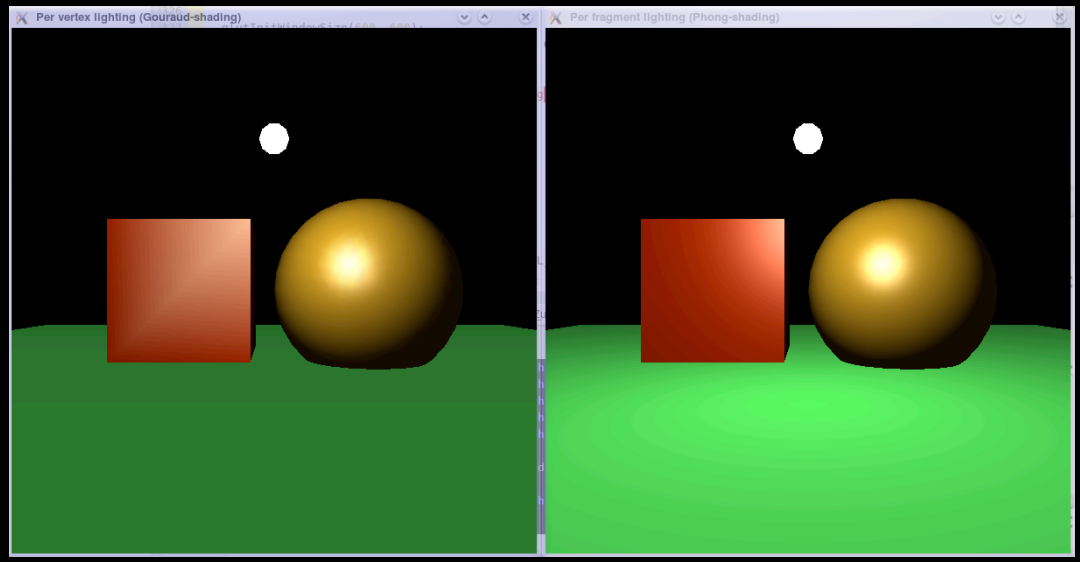
$$\mathbf{N} = \frac{y - y_2}{y_1 - y_2}\mathbf{N}_1 + \frac{y_1 - y}{y_1 - y_2}\mathbf{N}_2$$



Flat, Gouraud and Phong



Flat, Gouraud and Phong



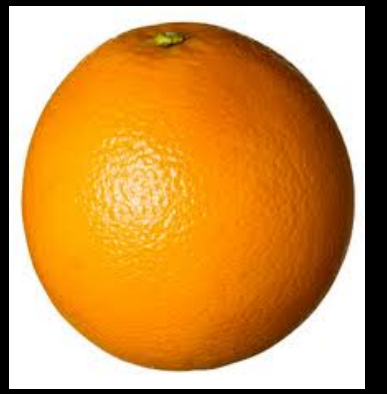
Texturing

Limits of Geometric Modeling

- Although GPU can render over 375 million vertices per second, that number is insufficient for many phenomena
  - Clouds
  - Grass
  - Terrain
  - Skin

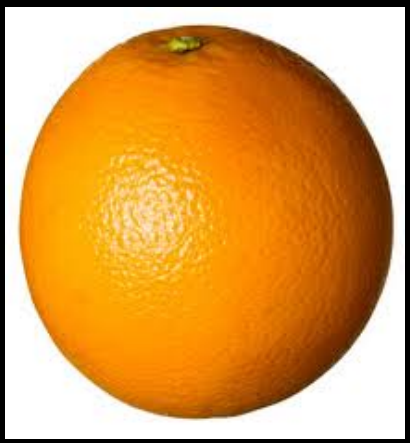
Modeling an Orange

- Consider the problem of modeling an orange
- Start with an orange-colored sphere
  - Too simple
- Replace sphere with a more complex shape
  - Does not capture surface characteristics
  - Takes too many polygons to model all the dimples



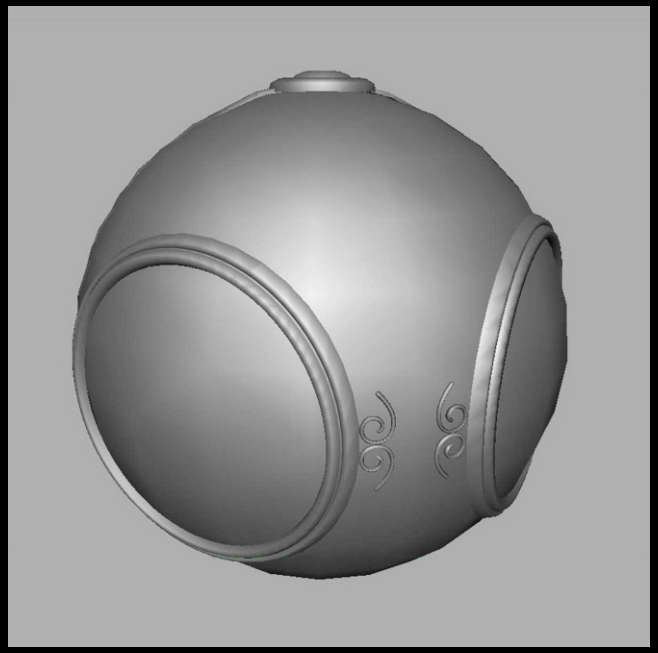
Modeling an Orange (2)

- Take a picture of a real orange, scan it, and "paste" onto simple geometric model
  - This process is known as *texture mapping*
- Still might not be sufficient because resulting surface will be smooth
  - Need to change local shape
  - Bump mapping



Three Types of Mapping

- Texture mapping
  - Uses images to fill inside of polygons

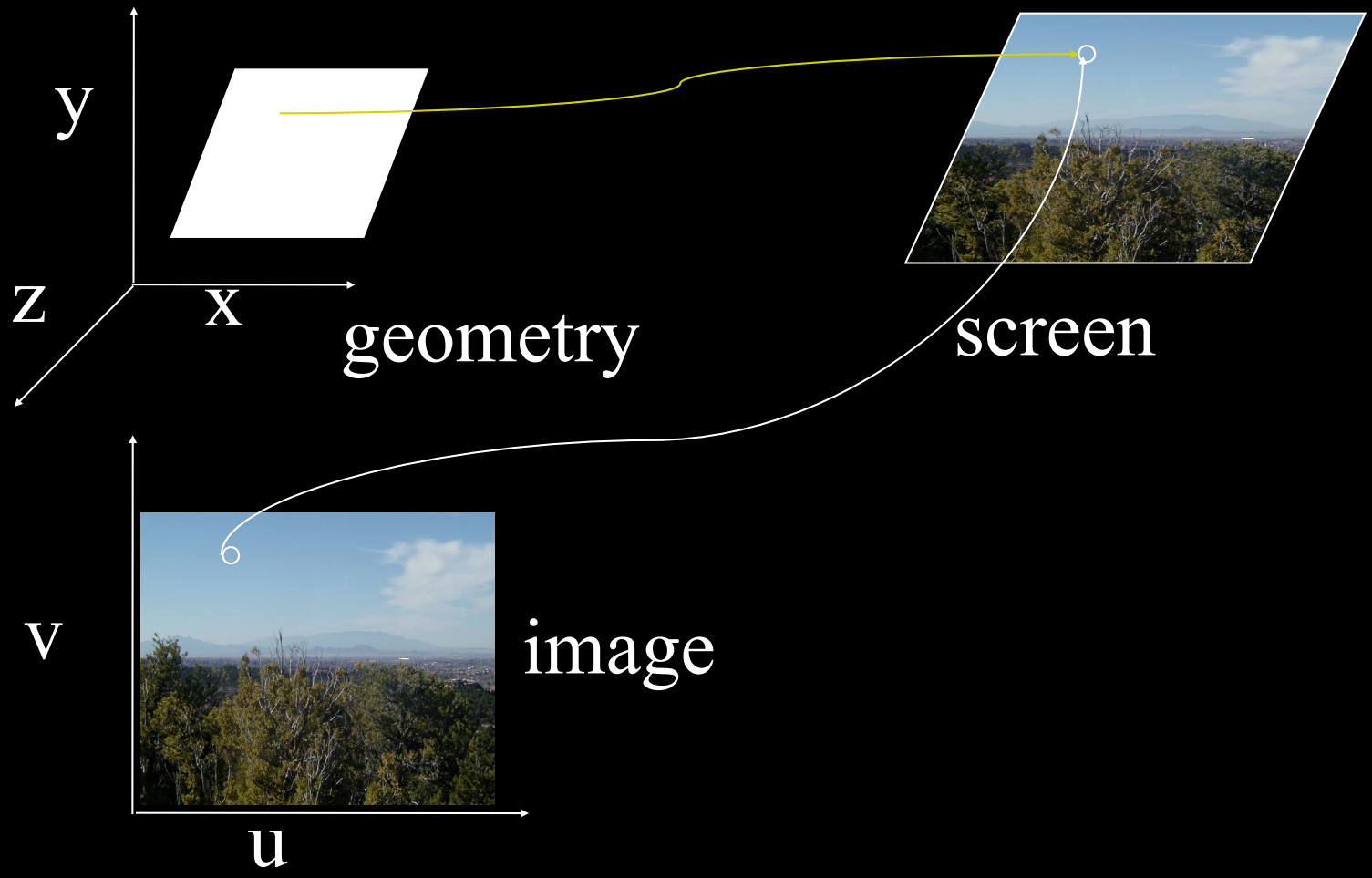


geometric model

texture mapped

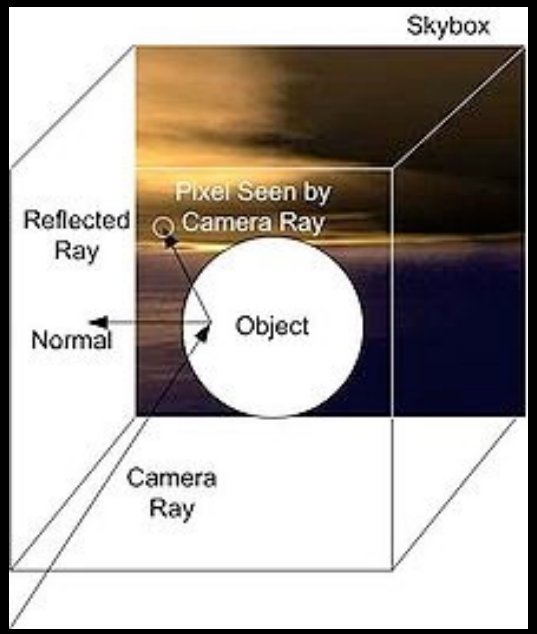
Three Types of Mapping

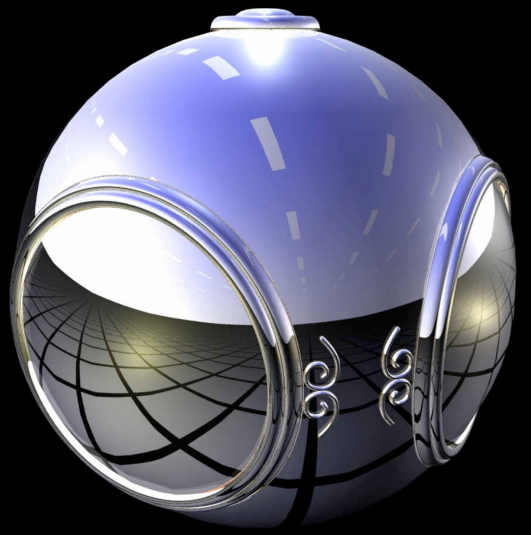
- Texture mapping



Three Types of Mapping

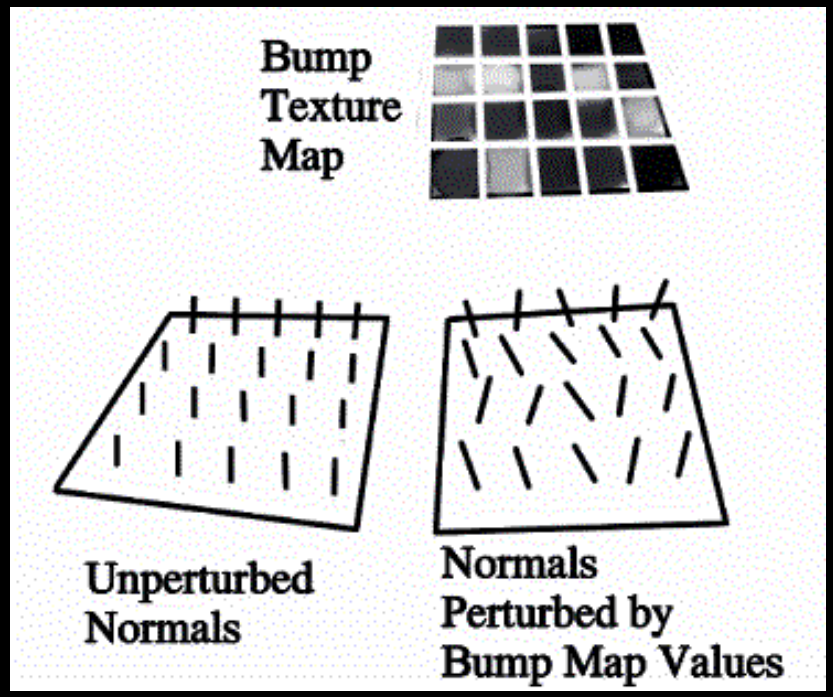
- Environmental (reflection mapping)
  - Uses a picture of the environment for texture maps
  - Allows simulation of highly specular surfaces

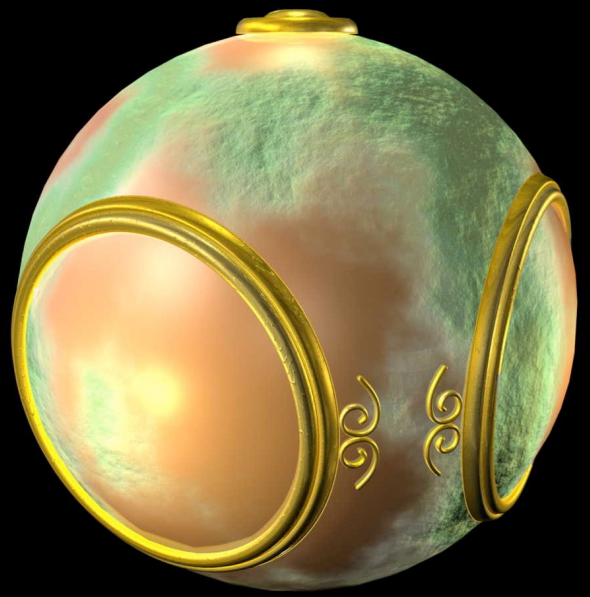




Three Types of Mapping

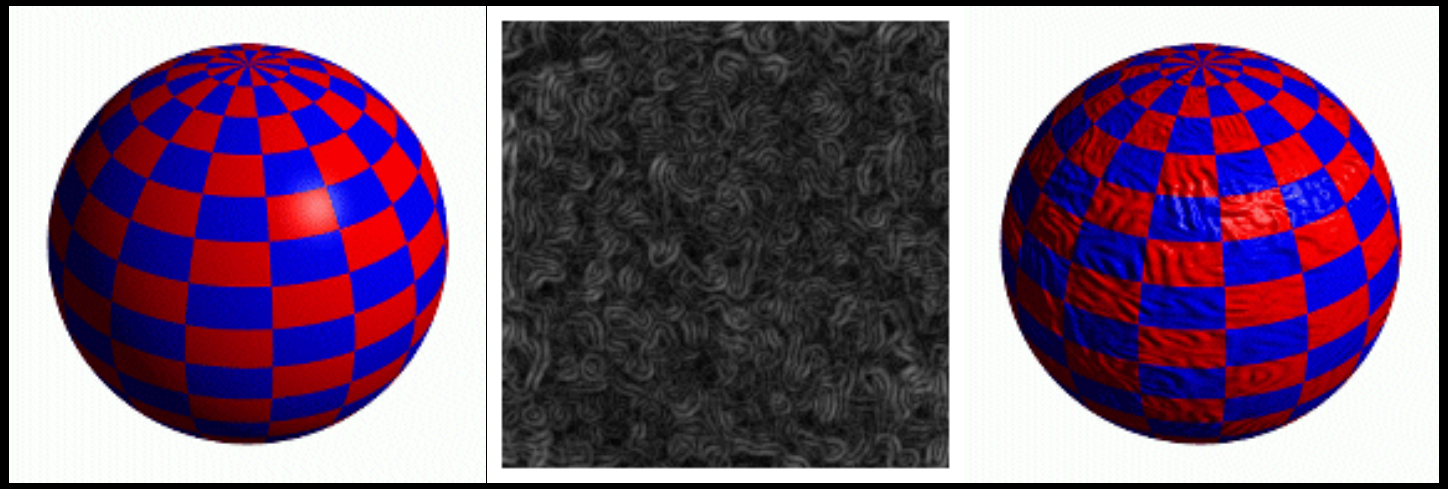
- Bump mapping
  - Emulates altering normal vectors during the rendering process





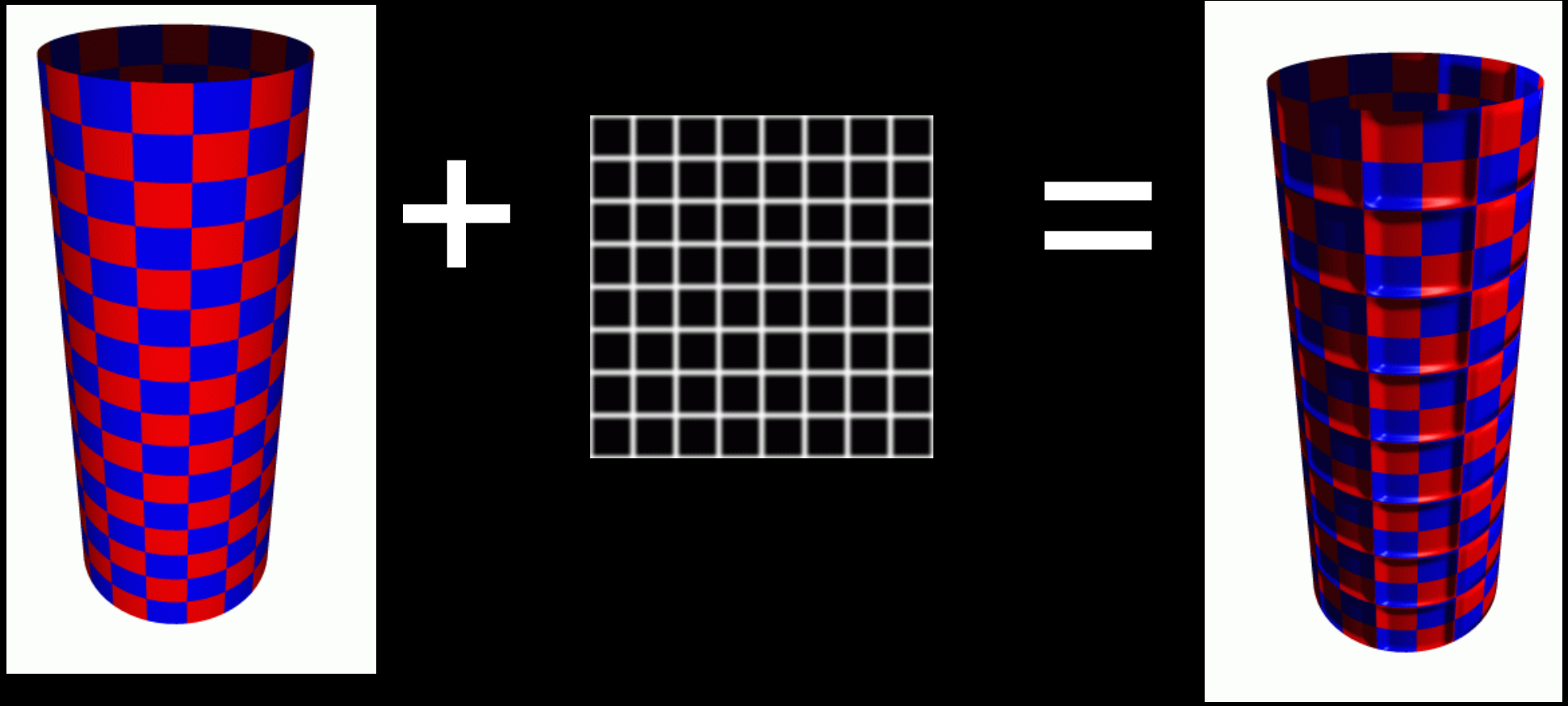
Bump Mapping

- The texture map can modulate the surface normal used for shading

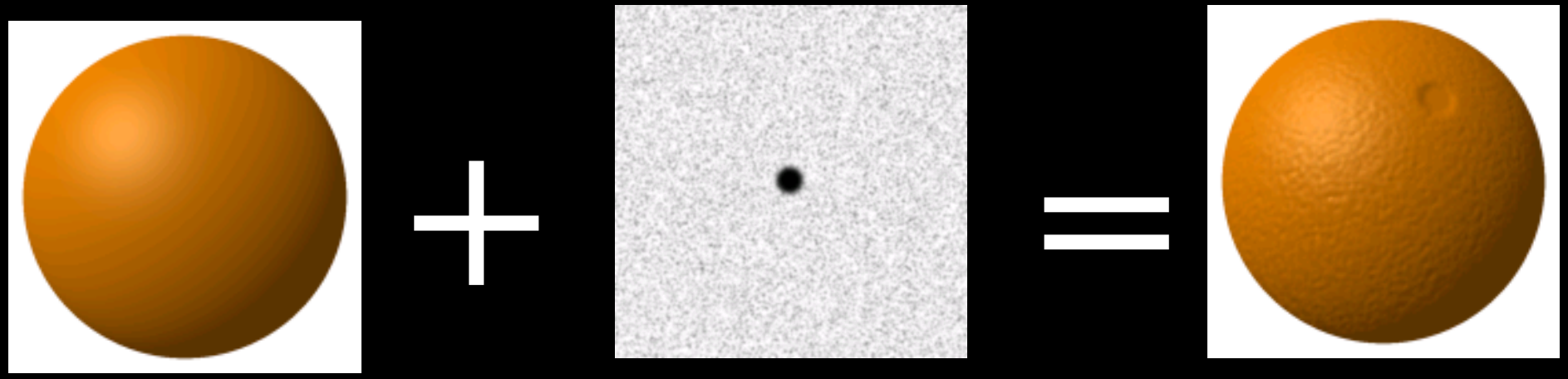


Sphere w/ diffuse texture *Swirly bump map* *Sphere w/ diffuse texture and swirly bump map*

More Bump Mapping



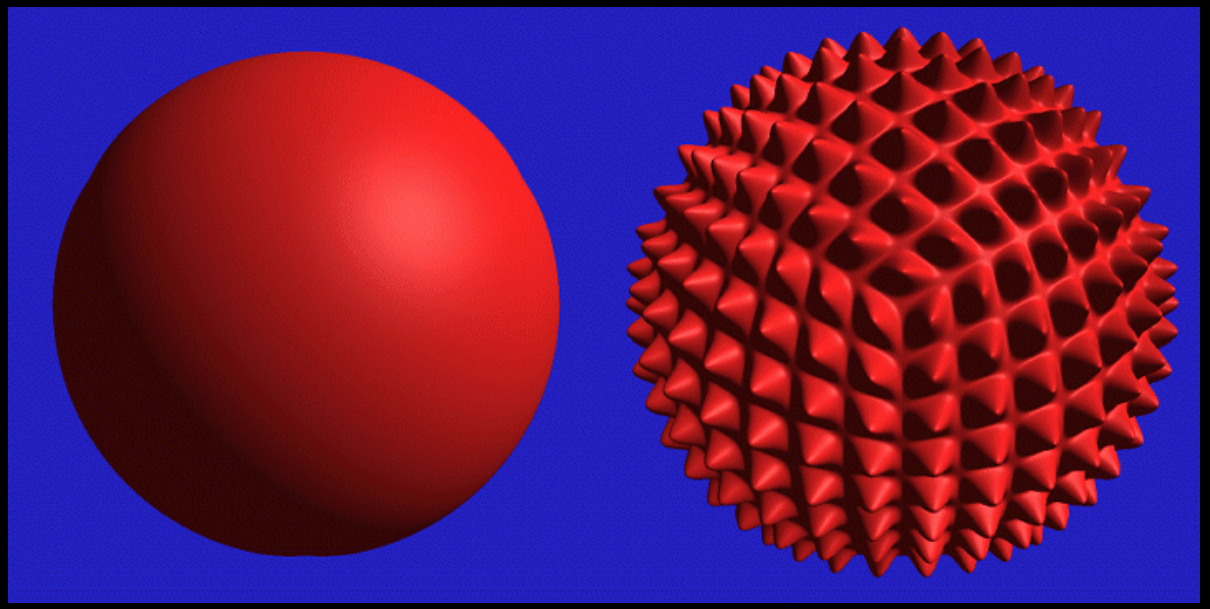
Another Bump Mapping Example



Displacement Map

- A *displacement map* actually displaces the geometry
  - Treats the texture as a height field to be applied to the surface
  - Starting to appear in the interactive graphics pipeline
    - First supported in Matrox Parhelia card
    - Can sort of implement with beta drivers in ATI & NVIDIA cards
    - Will soon appear in all cards
  - Implemented by recursive subdivision of triangles/quads

Displacement Map Example

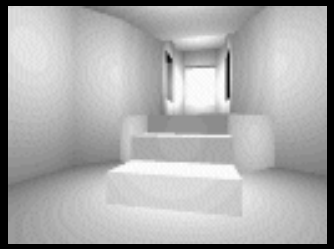


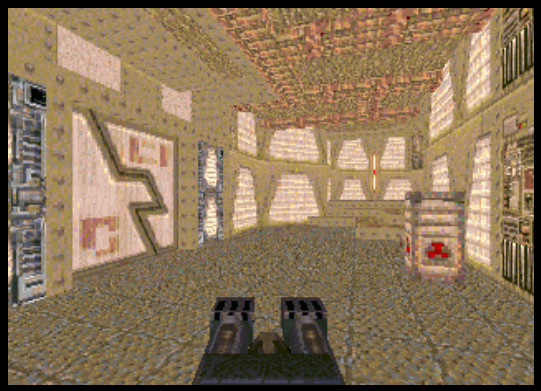
- *What is the biggest visual difference between displacement mapping and bump mapping?*

Illumination Maps

Texture map:

- Quake introduced *illumination maps* or *light maps* to capture lighting effects in video games

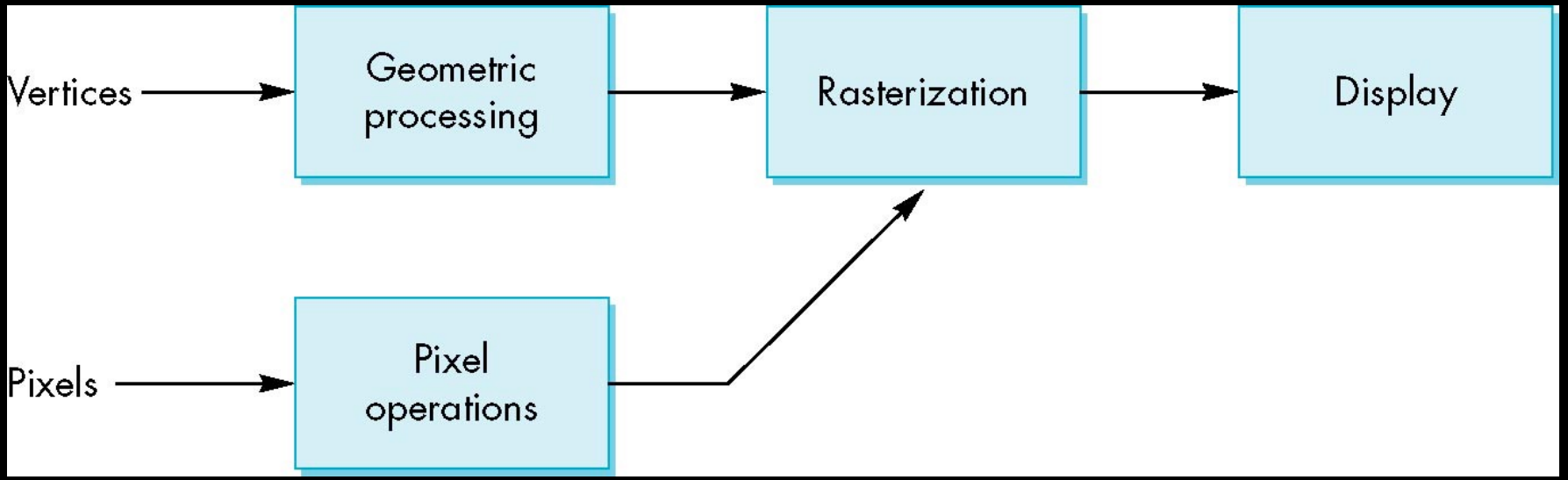




Texture map
+ light map:

Where Does Mapping Take Place?

- Mapping techniques are implemented at the end of the rendering pipeline
  - Very efficient because few polygons make it past the clipper



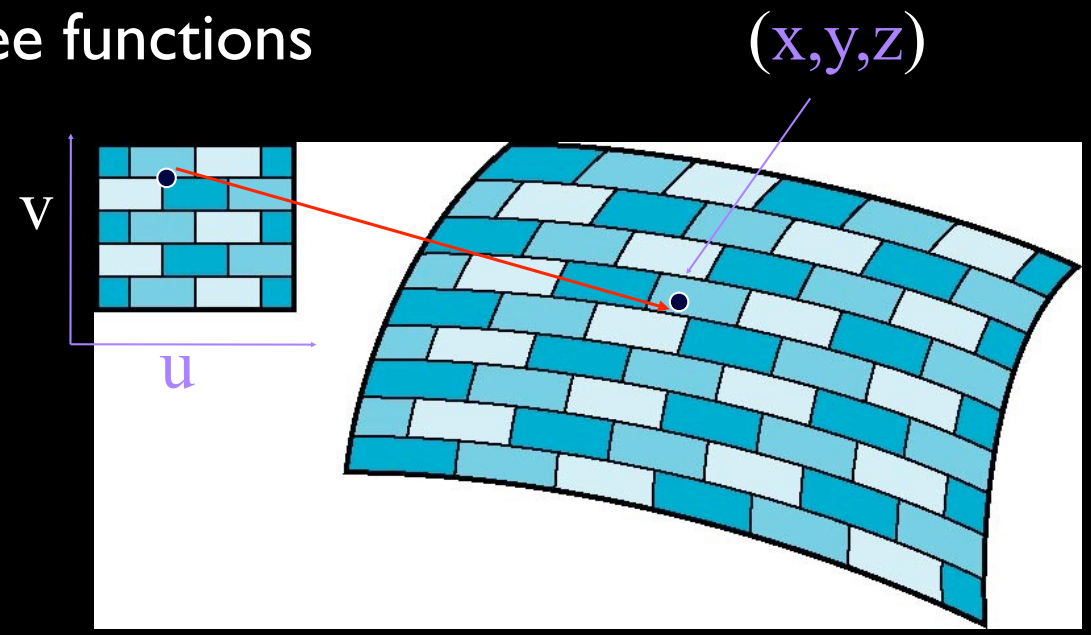
Mapping Functions

- Basic problem is how to find the maps
- Consider mapping from texture coordinates to a point a surface
- Appear to need three functions

$$x = x(u,v)$$

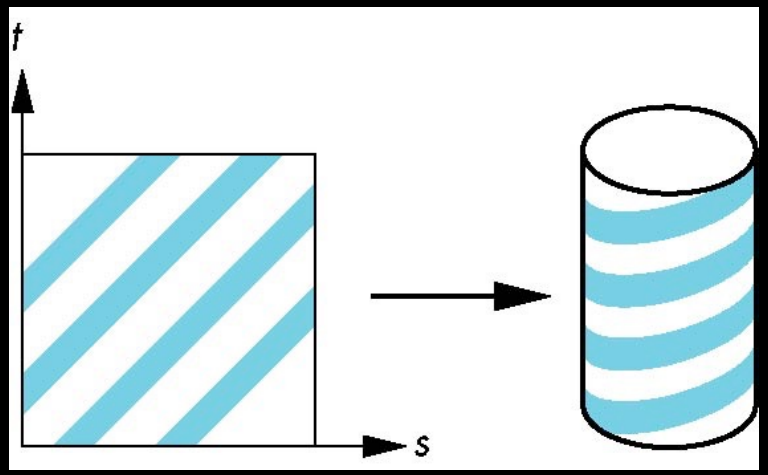
$$y = y(u,v)$$

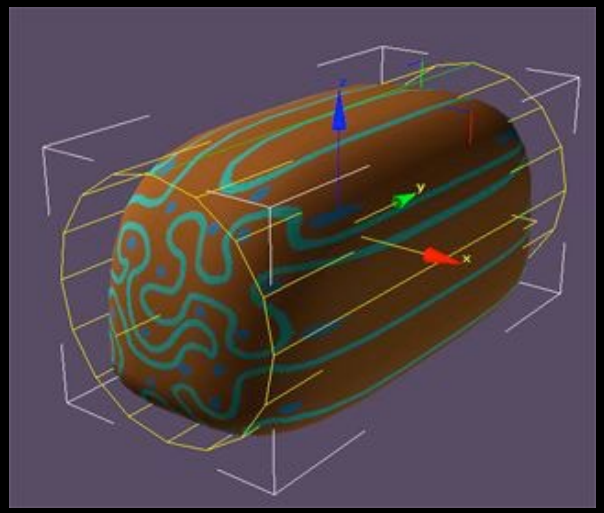
$$z = z(u,v)$$



Intermediate surface

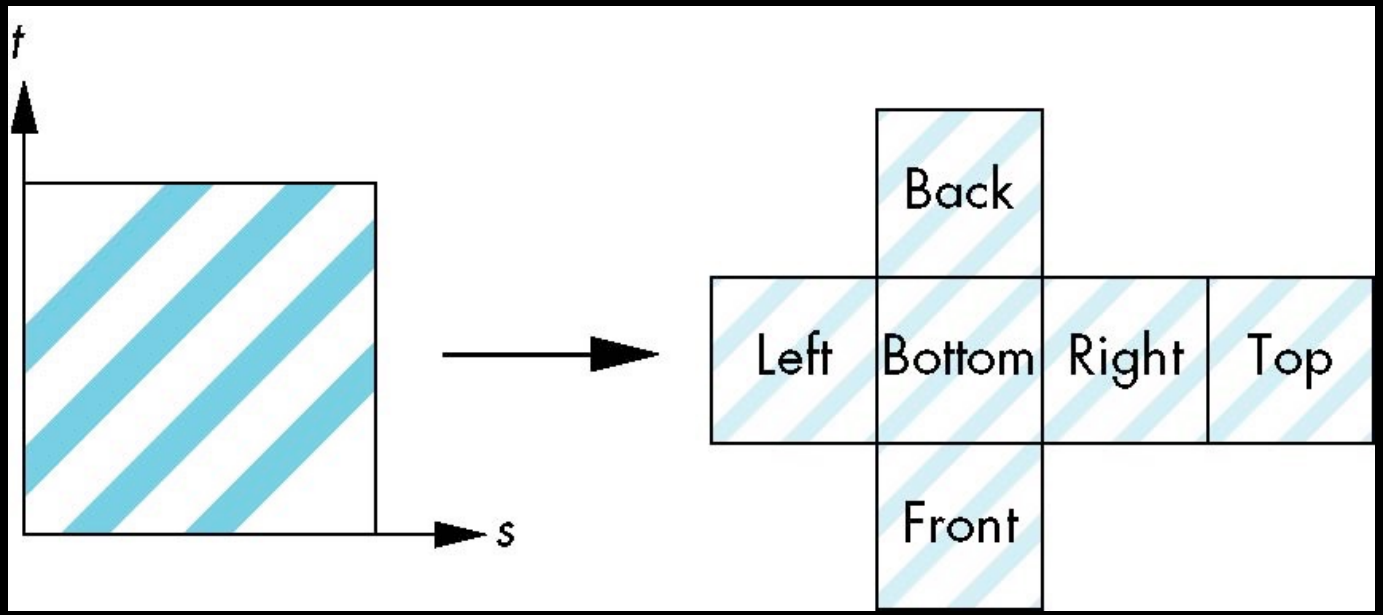
- first map the texture to a simple intermediate surface
- Example: map to a cylinder



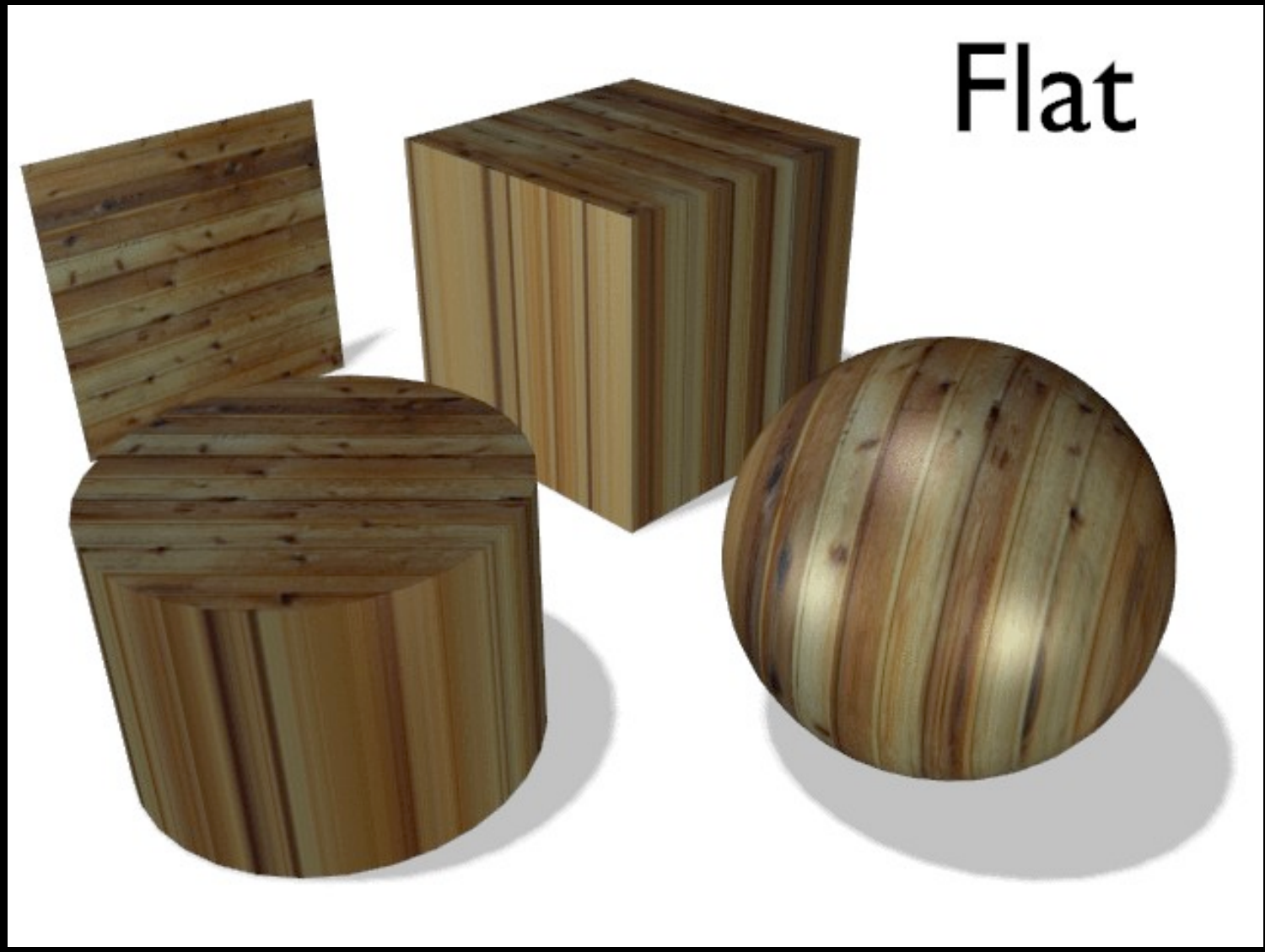


Box Mapping

- Easy to use with simple orthographic projection
- Also used in environmental maps



Other shapes



Second Mapping

- Map from intermediate object to actual object
  - Normals from intermediate to actual
  - Normals from actual to intermediate
  - Vectors from center of intermediate