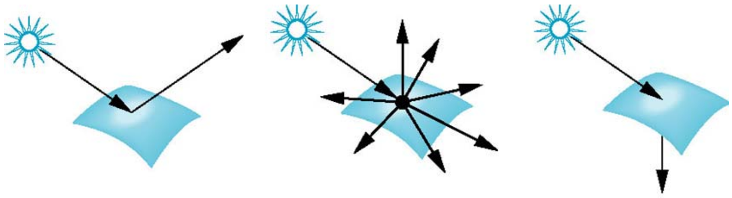
			Readings	Homework
Tue	3	Materials	Chap 14	
Wed	4	Lighting setup exercise		HW #3
Thur	5	<Children's Day>		Due (5/6)
	10	Shaders (Review+)	Chap 1~14	HW #4
	11	Open Lab		
	12	Color / Shading	Chap 19/Ext	
	17	Raytracing	Chap 20	
	18	Open Lab		
	19	Light	Chap 21	**Due:**
	24	Texture Mapping 1	Chap 15	**May 24 11:59PM**
	25	Texture mapping exercise		HW #5
	26	Lab		
	31	Texture Mapping 2	Chap 15	
7-10PM	1	CUDA Special Lab (by NVIDIA)		
	2	Sampling	Chap 16	
	7	Samplling/Reconstruction	Chap 16/17	
	8	Open Lab		
	9	Geometirc modeling	Chap 22	
	14	Animation	Chap 23	
	21	Final Exam		

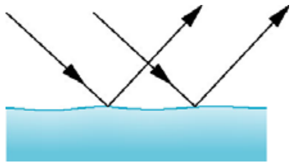
Materials
Light & Shading (2)

May 12, 2016

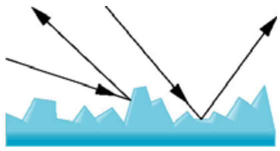
Light-Material Interactions



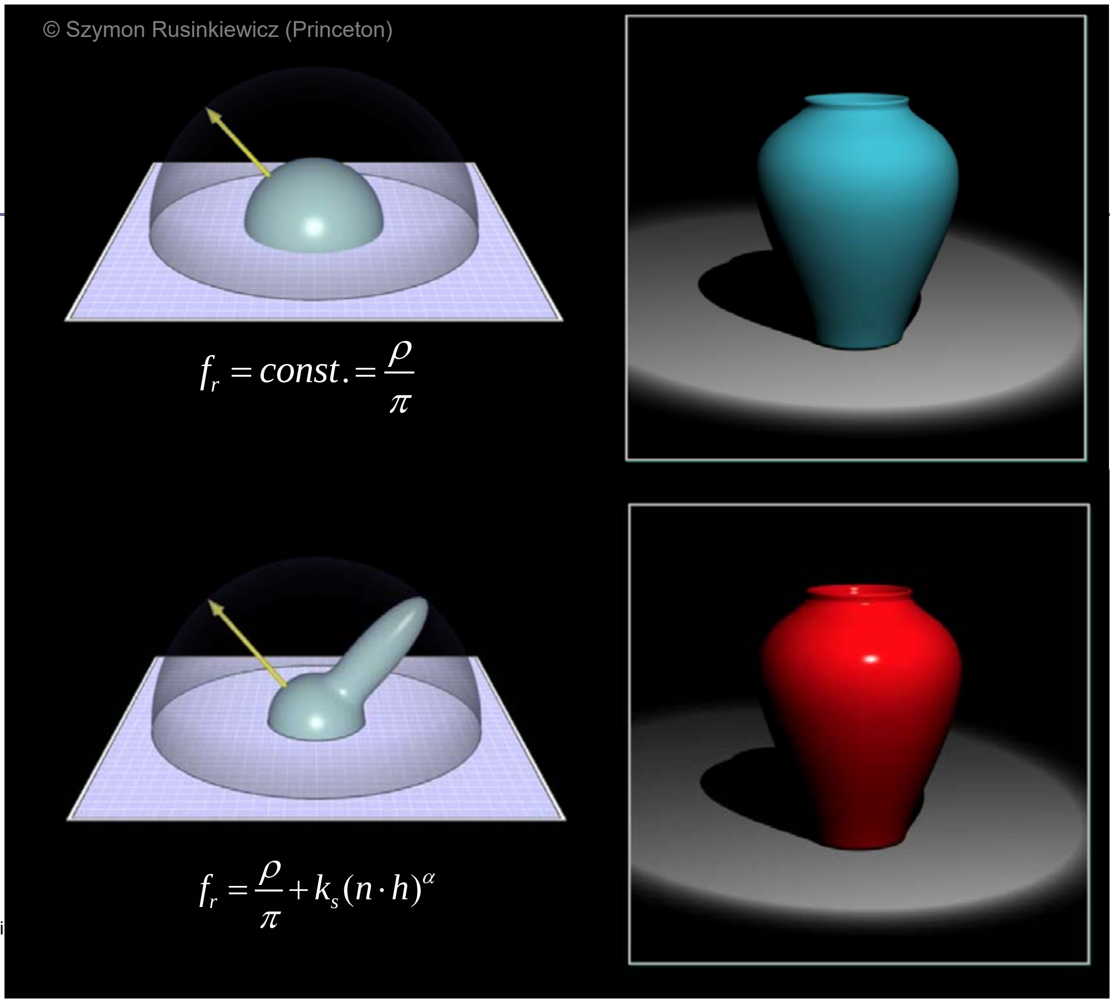
- Specular surfaces
  - Appear shiny because most of the reflected light is scattered in a narrow range. (mirror)
  - Perfectly *Specular* Surface
    = very smooth surface

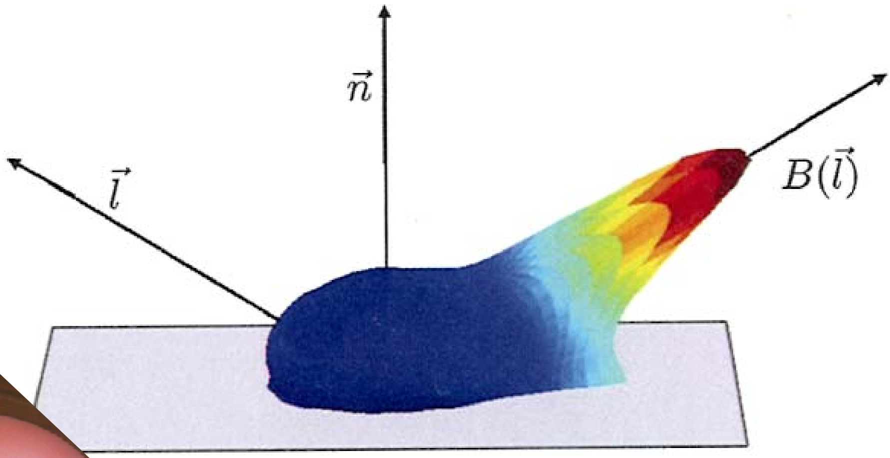


- Diffuse surfaces
  - Reflected light is scattered *in all directions*
  - There is no preferred angle of reflection
  - Perfectly *Diffuse* Surface
    = very rough surface

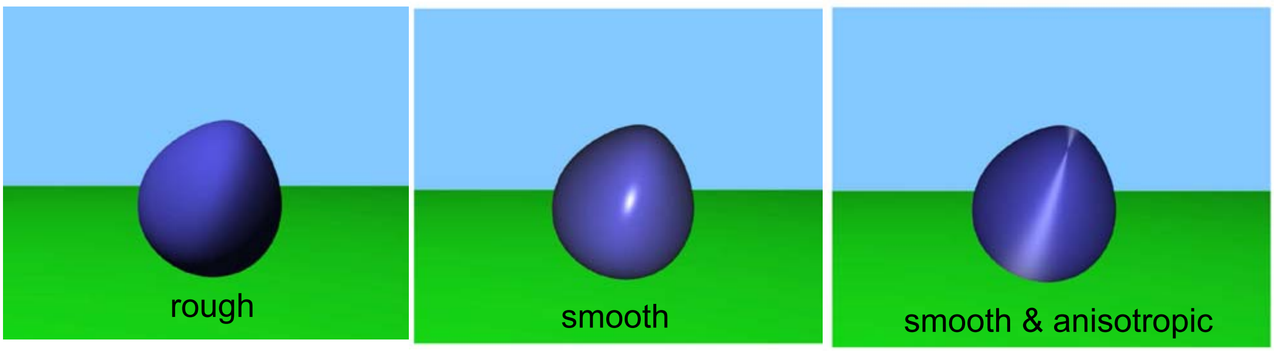


- Translucent surfaces

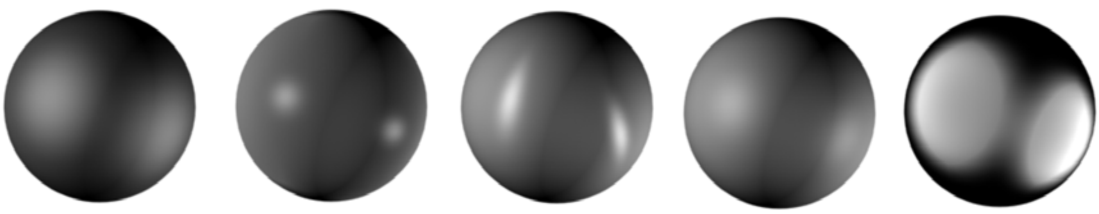




Material properties

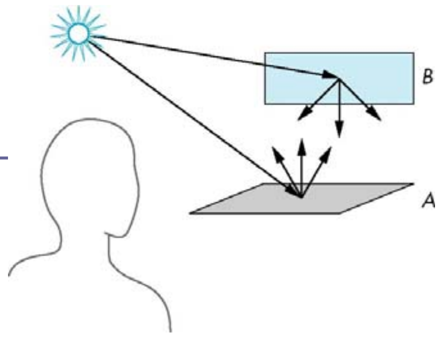


“Shading” Examples

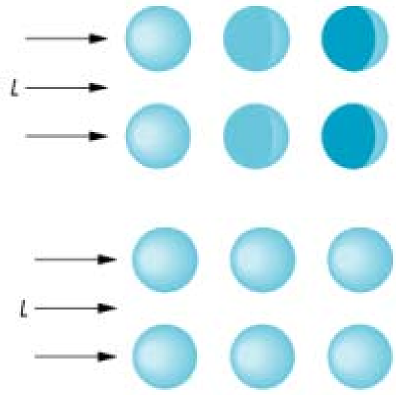


Phong Oren-Nayar-Blinn Anisotropic Strauss Metal

Light and Matter



- Rendering equation
  — very complex
- Ray tracing / Radiocity
  - global model
  - Not suitable for the graphics pipeline
- **Phong reflection model**
  - local model
  - A point on the surface is independent of the other points



Phong Reflection Model

- 3 types of material-light interactions

Reflection ⊗ **Illumination** ⟶ intensity

  - Ambient : same at every point
  - Diffuse : Lambert's law
  - Specular : shininess
- 3 color model (R, G, B)

Phong Reflection Model

- Light source
  - L : illumination
  - For each light source i

$$L_i = \begin{array}{c} \\ \\ \end{array}\begin{array}{c|c} \begin{array}{ccc} \text{red} & \text{green} & \text{blue} \end{array} & \\ \begin{pmatrix} L_{ira} & L_{iga} & L_{iba} \\ L_{ird} & L_{igd} & L_{ibd} \\ L_{irs} & L_{igs} & L_{ibs} \end{pmatrix} & \begin{array}{l} \textit{ambient} \\ \textit{diffuse} \\ \textit{specular} \end{array} \end{array}$$

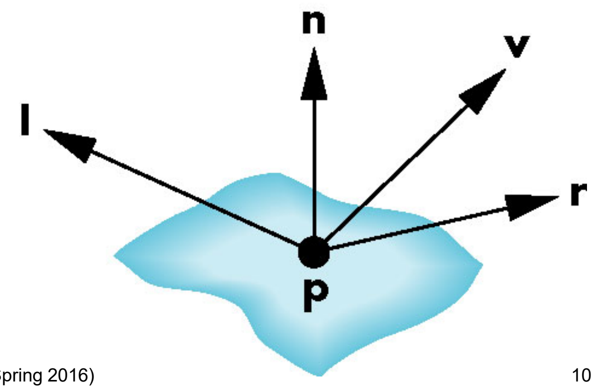
- Material model
  - R : reflection (how much of each of the incident lights is reflected at the point of interest)
  - At a point, it has the reflection for each light source

$$R_i = \begin{pmatrix} R_{ira} & R_{iga} & R_{iba} \\ R_{ird} & R_{igd} & R_{ibd} \\ R_{irs} & R_{igs} & R_{ibs} \end{pmatrix}$$

Intensity at **a <u>point</u>** p: **I = Reflection ⊗ Illumination**

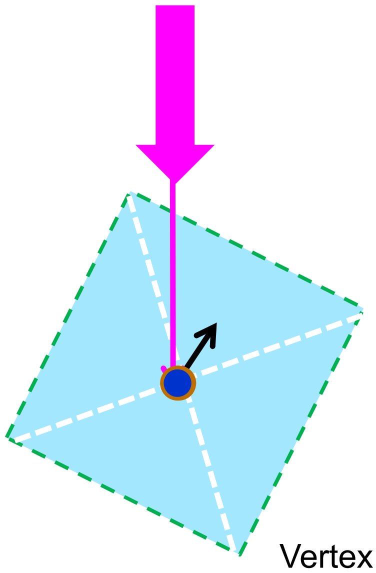
Phong Reflection Model

- Efficient, and close enough to physical reality
- Supports ambient, diffuse and specular (material-light interactions)
- To compute a color at a point **p** on the surface, use 4 vectors
  - Surface normal
  - Direction from p to the viewer
  - Direction of a line from p to a light source
  - Direction of reflection



So far …

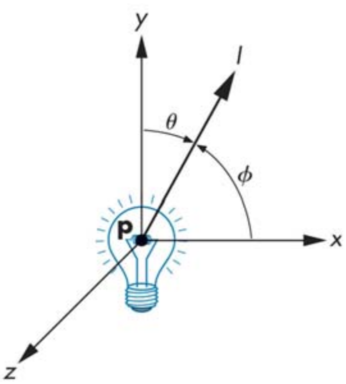
- Phong Reflection Model
  - Ambient reflection
  - Diffuse reflection
  - Specular reflection
- Computation of Vectors
- ***Light Sources***
  - Ambient light
  - Point sources
  - Spotlight
  - Distant light sources



Vertex attributes
- position
- color
- normal

Light Sources

- A light source: an object that emits light only through internal energy source
  - We neglect the reflection term for simplification
  - An object with a surface



Illumination function

$$I\,(\,x,y,z,\theta,\phi,\ \lambda\,)$$

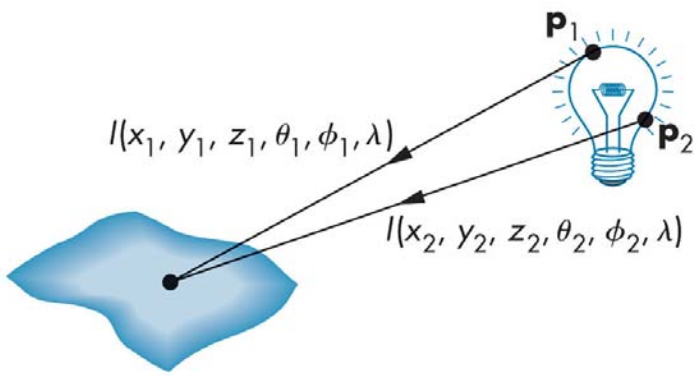
position of a point

direction of emission

intensity of energy emitted

Light Sources

- Total contribution of the source
  - Integrate over the surface of the source
  - A light bulb? Difficult
  - Model the distributed source with polygons or an approximating set of point sources.



$$\int \boldsymbol{I}\,(\,x,y,z,\theta,\phi,\lambda\,)$$

Light Sources

- We also apply three-color theory of the human visual system to model the light source.
  - We compute independently the intensity of the red, green and blue components

$$I(\lambda) \Rightarrow I = \begin{bmatrix} I_r \\ I_g \\ I_b \end{bmatrix}$$

- Types in <u>Computer Graphics World</u>
  - Ambient light
  - Point sources
  - Spotlights
  - Distant light sources

Ambient Light

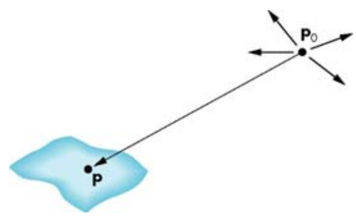
- Uniform illumination throughout the room.
- "Background" light

$$I_a = \begin{bmatrix} I_{ar} \\ I_{ag} \\ I_{ab} \end{bmatrix}$$

- Although every point in the scene receives the same illumination from I_a, each surface can reflect this light differently

Point Sources

- Emits light equally in all direction



$$I(P_0) = \begin{bmatrix} I_r(P_0) \\ I_g(P_0) \\ I_b(P_0) \end{bmatrix}$$

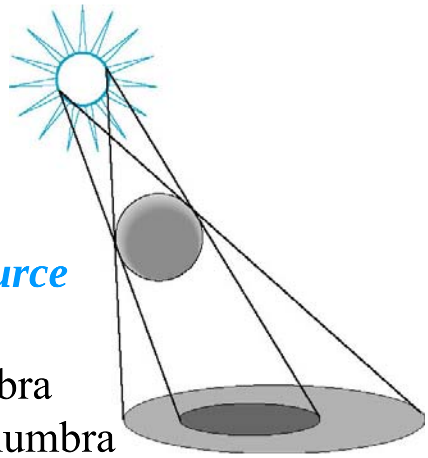
$$I(P) \propto \frac{I(P_0)}{|P - P_0|^2}$$

- Intensity of illumination received from a point source is proportional to the inverse square of the distance between the source and surface.

Point Sources (vs. area light)

- Creates very high contrast image
  - Add ambient light to mitigate the effect
- Distance term is modified to soften the light

$$\frac{1}{d^2} \rightarrow \frac{1}{a + bd + cd^2}$$



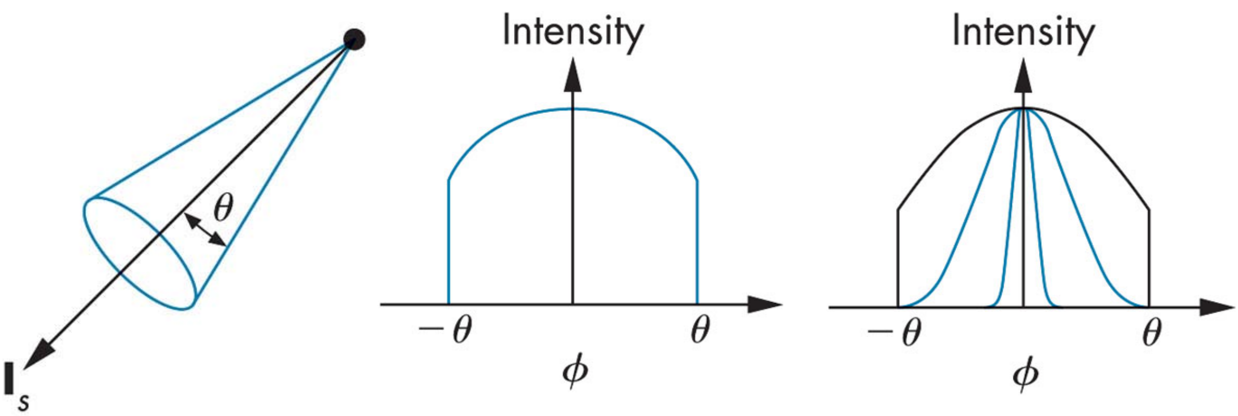
Shadows created by finite-size(area) light source

The blackest part of a shadow from which all light is cut off → umbra

A partial shadow between regions of complete shadow and complete illumination → penumbra

Spotlights

- Light is emitted through a narrow range of angles



Cutoff angle

$$I \propto \cos^e \phi$$

Distant Light Sources

- Point source $\rightarrow$ Distant source

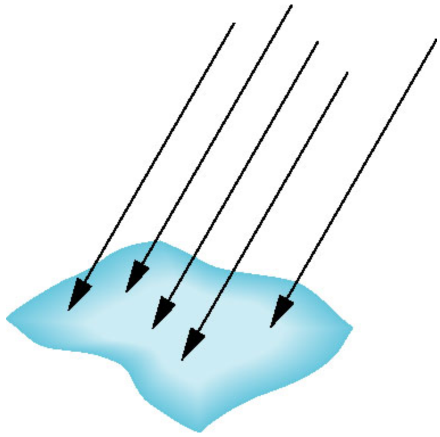
$$P_0 = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

location

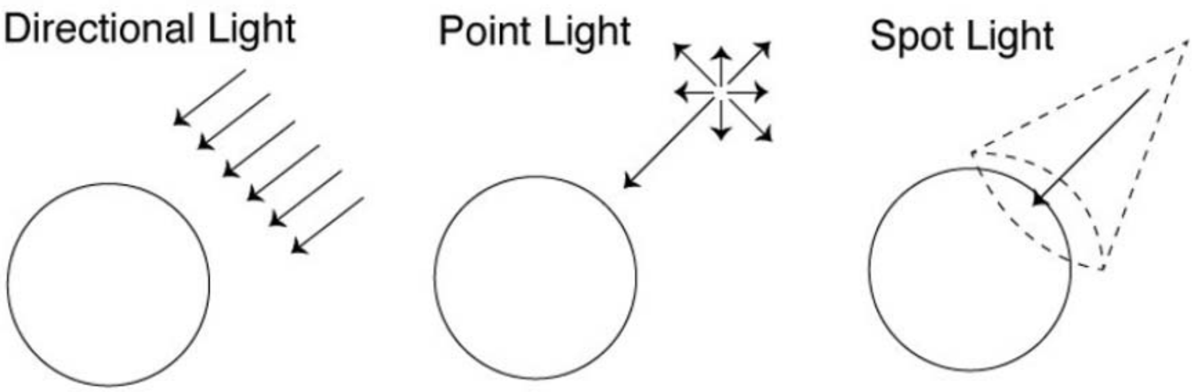
$$P_0 = \begin{bmatrix} x \\ y \\ z \\ 0 \end{bmatrix}$$

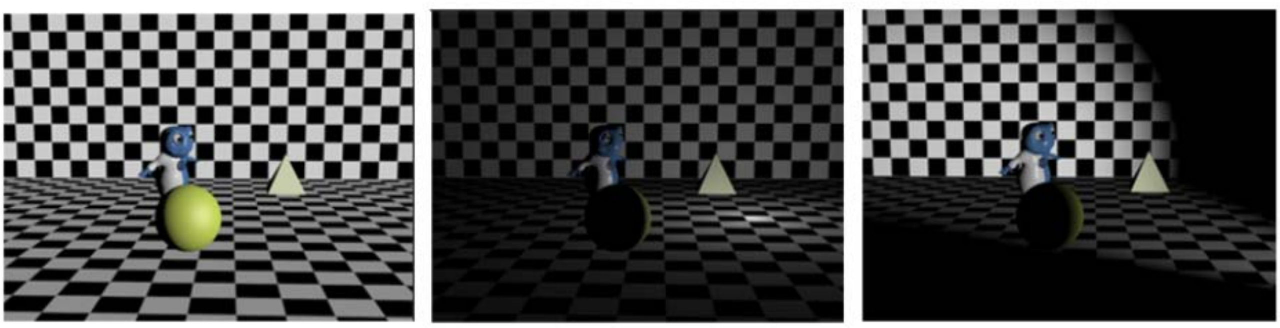
direction

Parallel rays of light



Homework #4





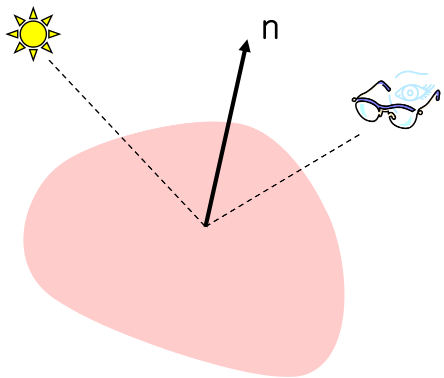
Now … with lighting

- Phong illumination model

$$I = k_a L_a + \sum \frac{1}{a + bd + cd^2} [k_d (\ell \cdot n) L_d + k_s (r \cdot v)^{\alpha} L_s]$$

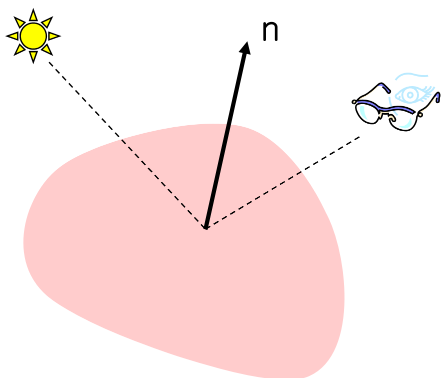
f (distance)

  - Computes a color for the plane defined by the normal vector
  - → The normal vector is associated with a vertex
  - → computes a color for a vertex



Now … with lighting

- Polygonal **Shading**
  - Flat shading
  - Gouraud shading
  - Phong shading

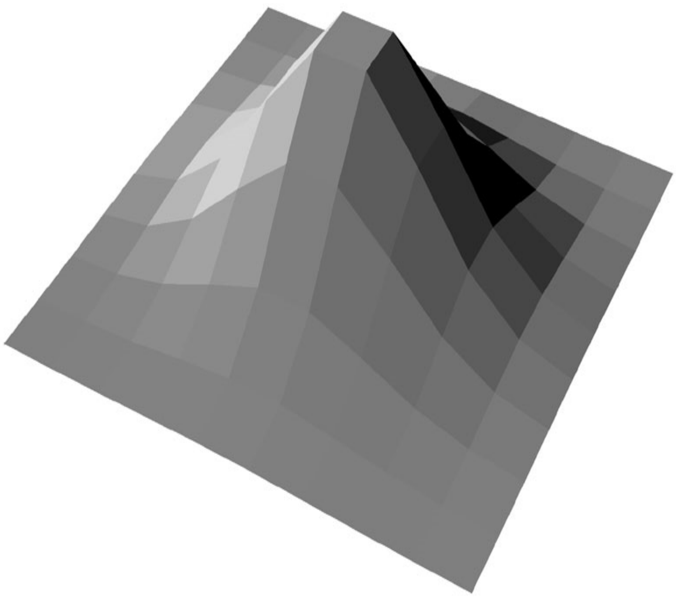


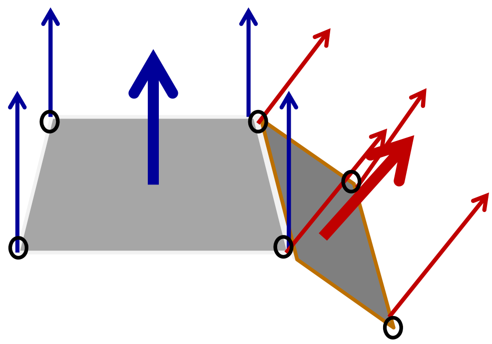
Flat Shading

- The 3 vectors l, n, and v can vary as we move from point to point. But
  - For a flat polygon, n is constant.
  - For a distant viewer, v is also constant.
  - For a distant light source, l is also constant.

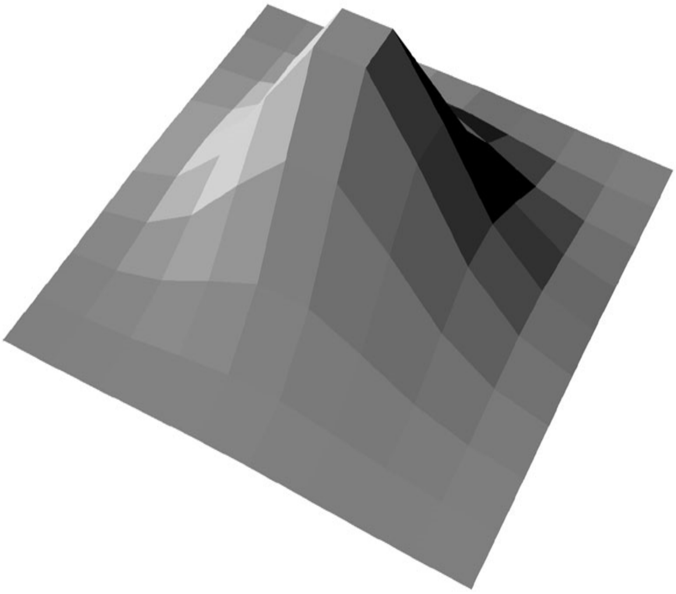
- Problems
  - The human visual system has a remarkable sensitivity to small differences in light intensity
  - Mach band

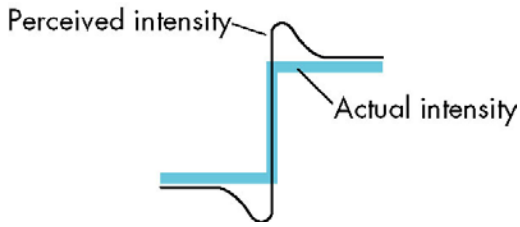
Flat shading

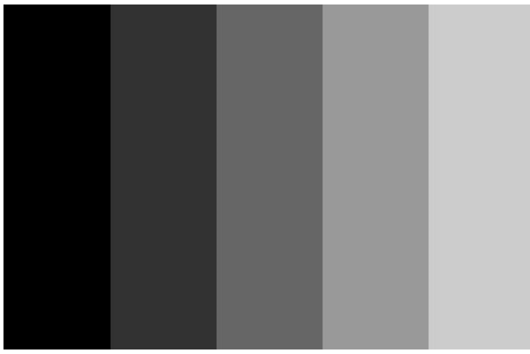




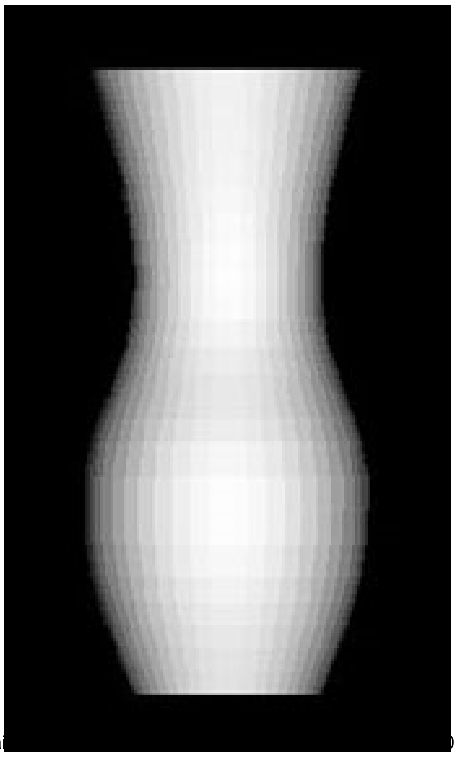
Flat Shading

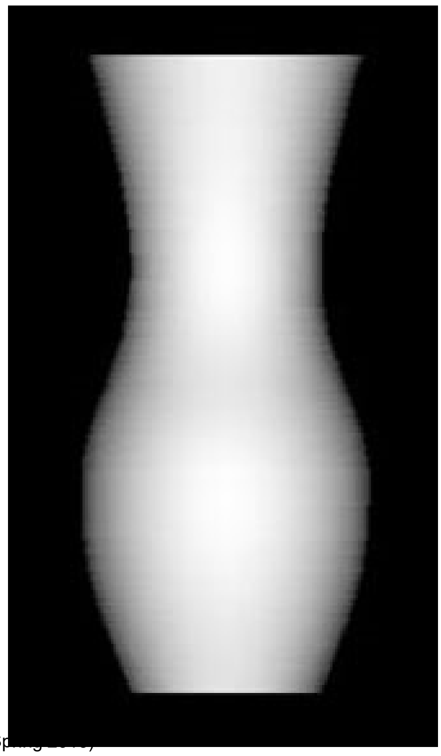






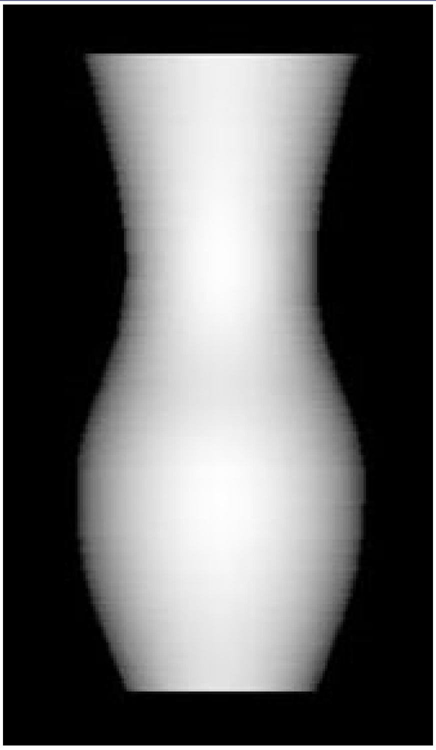
Shading

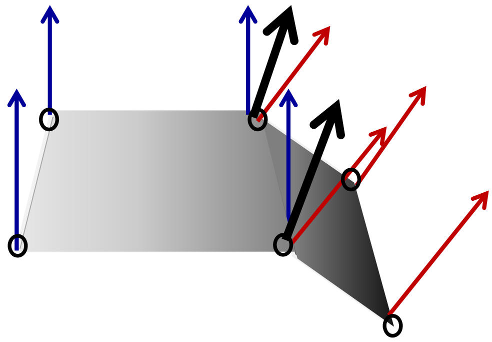




Cornell University

Gouraud Shading (smooth shading)

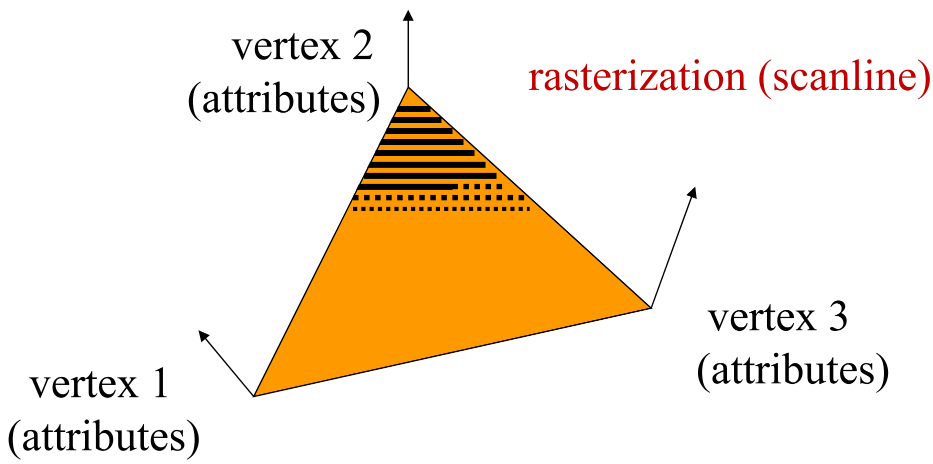




Smooth Shading

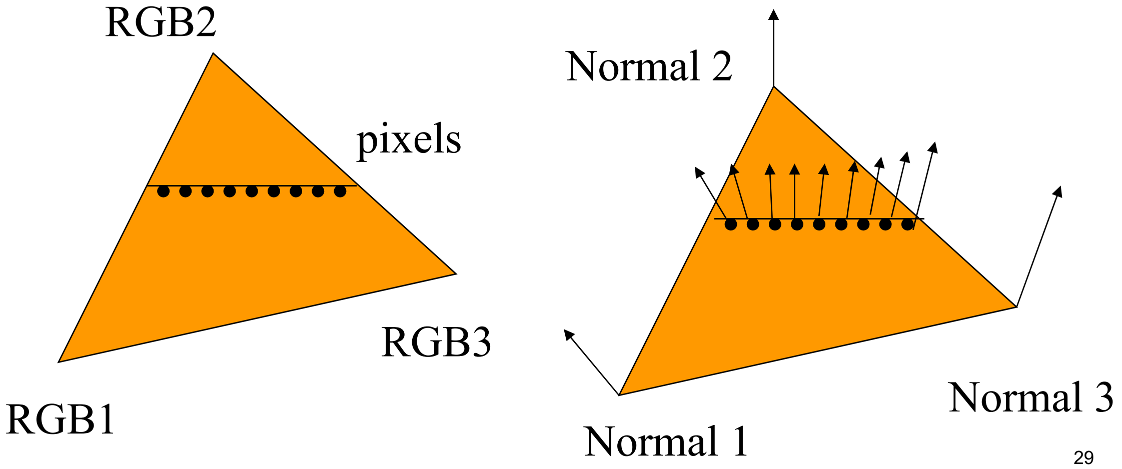
- Gouraud (interpolate colors)
- Phong (interpolate normals)

"Varying variables"



Shading

- Gouraud (interpolate colors) (Supported in fixed pipeline) ← Color computed in Vertex shader
- Phong (interpolate normals) (was NOT supported in fixed pipeline) ← Color computed in Fragment shader

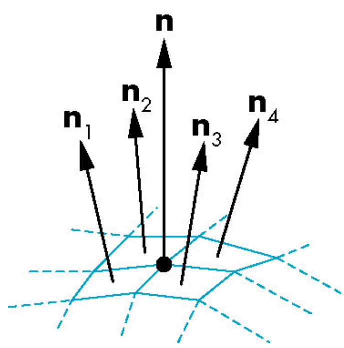


Gouraud Shading

- Normal at a vertex to be the normalized average of the normals of the polygons that share the vertex

$$\mathbf{n} = \frac{n_1 + n_2 + n_3 + n_4}{|n_1 + n_2 + n_3 + n_4|}$$

- algorithm
  - for each vertex
    - Find the surrounding polygons.
    - Compute the normal vector at the vertex.
    - Compute the shading.
  - for each pixel inside the polygon
    - *Interpolate the shading.*
    - → Already done in hardware
- Issue - data structure for representing meshes.



Phong Shading

- Instead of interpolating vertex intensities, interpolate normals across each polygon
- algorithm
  - for each vertex
    - Find the surrounding polygons.
    - Compute the normal vector at the vertex.
  - for each pixel inside the polygon
    - Interpolate the normal vector.
    - Compute the shading.

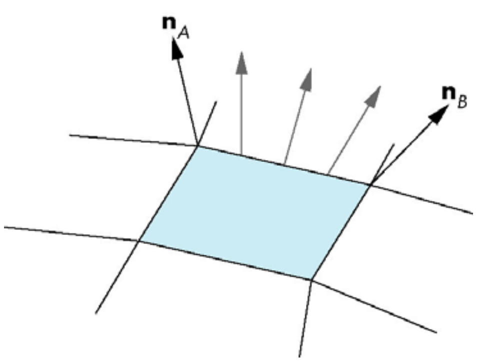
- More expensive
- Hardware implementation was hard in the fixed pipeline.
- But we can do this in fragment shader!!

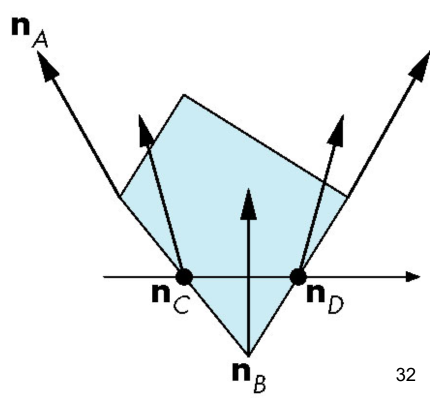
Phong Shading

- Bilinear interpolation
  - Find edge normals

    $\mathbf{n}(\alpha) = (1 - \alpha)\ \mathbf{n_A} + \alpha\ \mathbf{n_B}$
  - Normal at any interior point

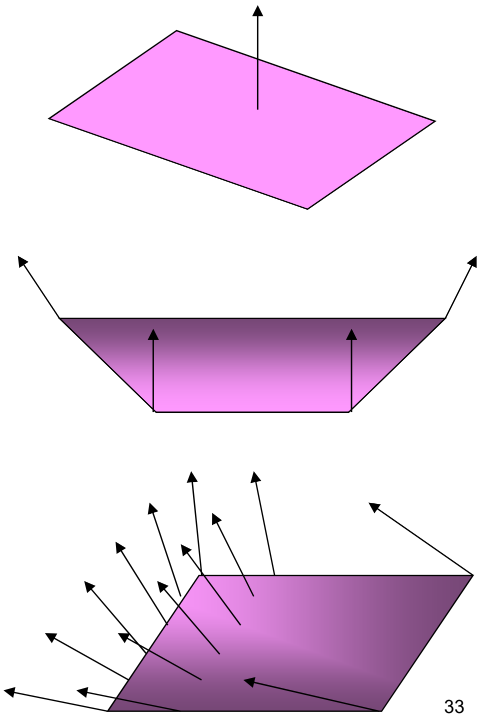
    $\mathbf{n}(\alpha,\beta) = (1 - \beta)\ \mathbf{n_C} + \beta\ \mathbf{n_D}$



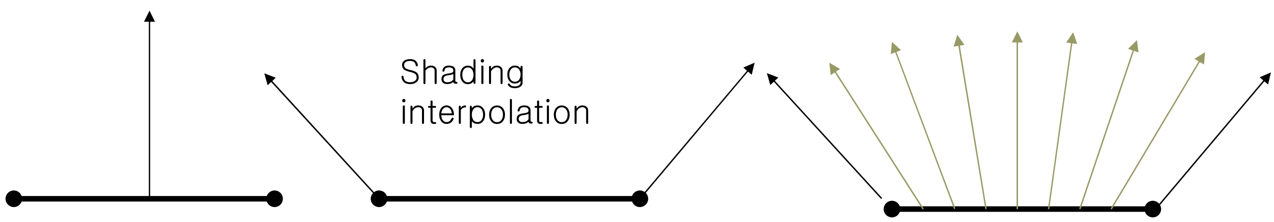


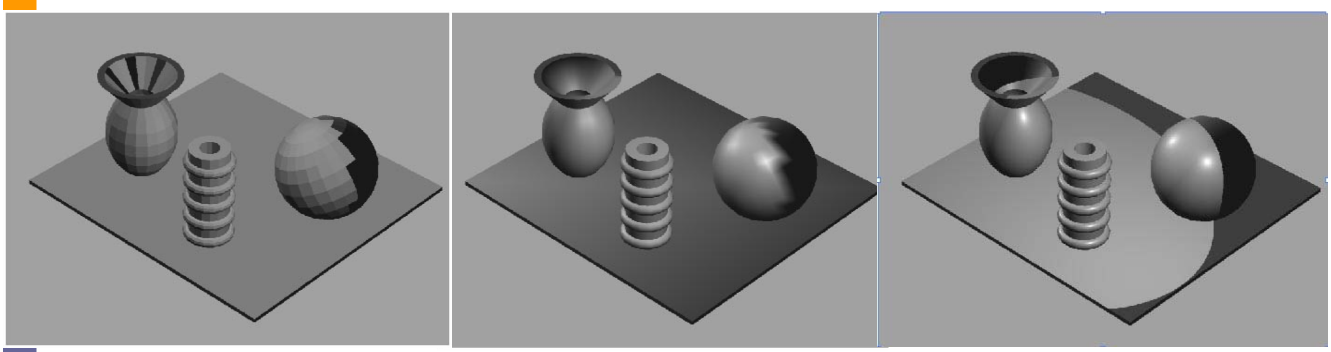
Shading Algorithms

- Flat (constant) shading
  - Lighting model is applied to one point of each polygon
- Gouraud shading
  - The lighting equation is applied to the vertices of a polygon only, and the interior points are calculated using these
  - (not give correct highlights)
- Phong shading

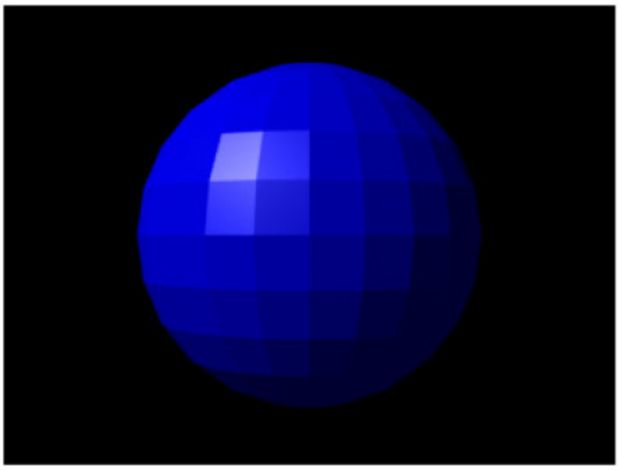
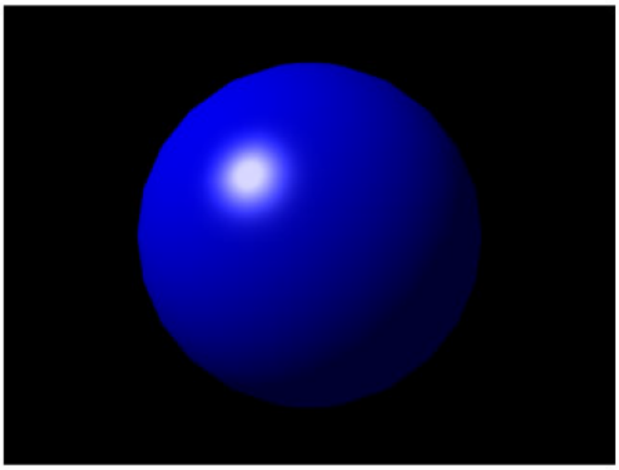


Shading





Homework #4

FLAT SHADING PHONG SHADING

Phong Lighting model applied

Per-vertex

Per-fragment Phong shading

http://www.lighthouse3d.com/tutorials/glsl-12-tutorial/toon-shading/

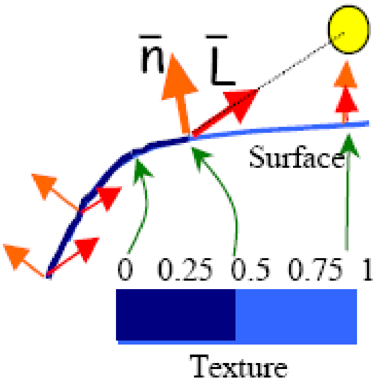
Toon Shading

- One of the simplest **non-photorealistic** shader.
  - It uses very few colors, usually tones.
    - The tones in the teapot are selected based on the angle, actually on the cosine of the angle, between a virtual light's direction and the normal of the surface.
- If we have a normal that is close to the light's direction, then we'll use the brightest tone.
- As the angle between the normal and the light's direction increases darker tones will be used.
  In other words, the cosine of the angle provides an intensity for the tone.

Nonphotorealistic Shading

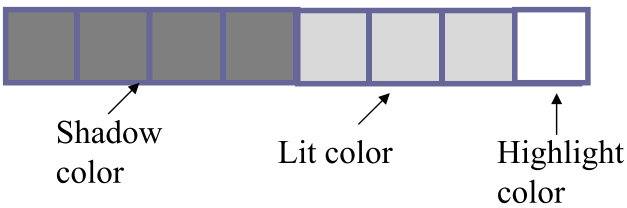
- Wide range of goals
  - Similar to technical illustrations
  - Only those details relevant to the goal of the particular application are the ones that should be displayed
  - Simulation of painterly styles and natural media
  - Pen and ink, charcoal, watercolor, …
  - **A cartoon rendering style** → Toon Shading
  - Information convey

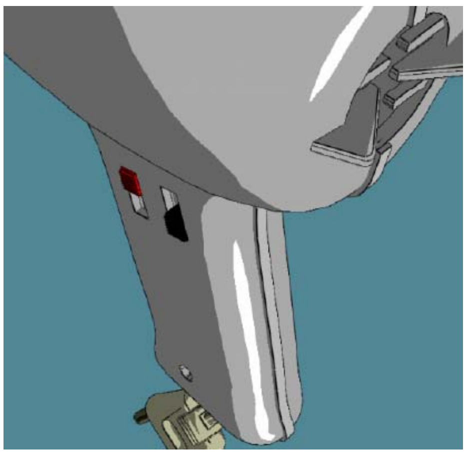
Toon Shading

- One-dimensional *texture*
- Ex) 2 colors
  - One for the illuminated color and
    one for the *shadowed* color.
  - Based on **L • n** term
- 

Toon Shading

- Higher resolution texture map
  - We can get flexibility for the location of the shadow boundary
    (8 texels set for shadow color, lit color, highlight color)
  - Highlight → view-dependent
    specular term





Toon Shading

- For shading
  - Fill the polygonal area with solid (unlit) color
  - Use a two-tone approach, representing lit and shadowed areas → hard shading
- Silhouettes are often rendered explicitly in a black color

Gouraud shading

Cartoon shading

Jérôme Thoma's Thesis (2002)

Toon Shader / Fragment Shader

- Vertex shader
  - Output:

```
float intensity = dot( L, N );
```

- Fragment shader
  - Algorithm:

```
vec4 color;

if (intensity > 0.95)
        color = vec4(1.0,0.5,0.5,1.0);
else if (intensity > 0.5)
        color = vec4(0.6,0.3,0.3,1.0);
else if (intensity > 0.25)
        color = vec4(0.4,0.2,0.2,1.0);
else
        color = vec4(0.2,0.1,0.1,1.0);

fColor = color;
```

Last Drawing of Canaletto (UCLA)

- SIGGRAPH 2000
- The Last Drawing of Canaletto is a 3D computer animated re-creation of an 18th century drawing by the Venetian artist Canaletto.
- The viewer is able to enter the "space" of the two-dimensional drawing and look around, while the moving light of the sun animates the otherwise motionless setting.
- A conscious effort was made to combine the visual qualities of claymation, model photography, and time-lapse photography with the unique possibilities offered by computer animation.

- **San Marco: the Crossing and North Transept, with Musicians Singing**
  1766
  Pen and ink with washes,
  47 x 36 cm
  Kunsthalle, Hamburg

http://www.wga.hu/frames-e.html?/html/c/canalett/index.html