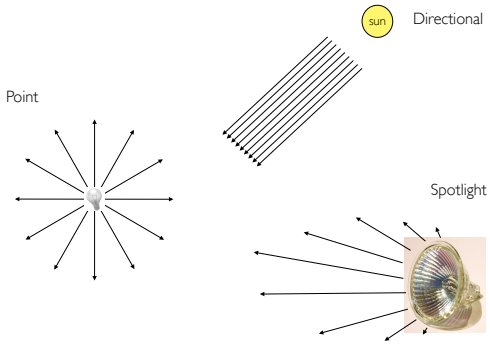
COMP175: Computer Graphics

Lecture 13
Illumination and GPUs

Review: Direct vs. global illumination

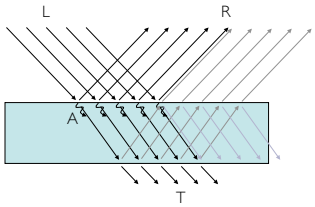
Review: Light sources



Review: Light-object interaction

The total light energy is conserved: A + R + T = L

The relative amounts depend on wavelength, material, shinyness

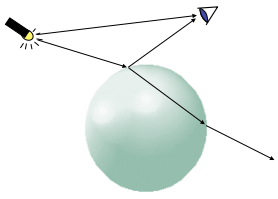


Reflection, transmission, and absorption

Review: Light-object interaction

Path of light to the eye is complex
- Direct to the eye
- Via an object

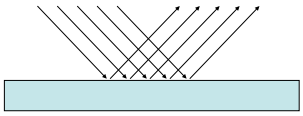
The interaction depends on
- Wavelength
- Geometry
- Material properties



Types of reflection: Specular

Most of the reflected light is scattered in a narrow range of angles

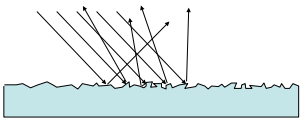
Smooth, shiny surfaces, e.g., mirrors, polished silver



Types of reflection: Diffuse

Reflected light is scattered in many directions

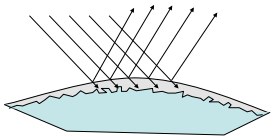
Rough surfaces, e.g., matte walls, fabric, unpolished wood



Types of reflection

Objects can exhibit both specular and diffuse reflection

Specular reflection from shiny surface layer
- Reflected color matches *light* color

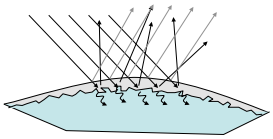


Types of reflection

Objects can exhibit both specular and diffuse reflection

Specular reflection from shiny surface layer
- Reflected color matches *light* color

Diffuse reflection from rougher sub-surface layer
- Reflected color matches diffuse *object* color (depends on absorption)

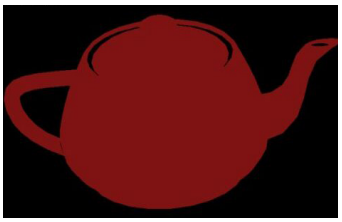


Phong illumination model

$$I = k_a I_a + k_d I_L(\mathbf{n}\cdot\mathbf{l}) + k_s I_L(\mathbf{r}\cdot\mathbf{v})^s$$

Ambient | Diffuse | Specular

Ambient reflection



Depends on the object's intrinsic color and the scene's ambient (i.e., background light) intensity

Constant over the surface of the object

Phong: Ambient component

$$I = k_a I_a + k_d I_L(\mathbf{n}\cdot\mathbf{l}) + k_s I_L(\mathbf{r}\cdot\mathbf{v})^s$$

Ambient

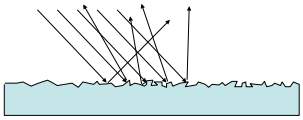
I_a : the color and intensity of the scene's ambient light

k_a : a material property of the object, its ambient reflection coefficient, $k_a \in [0,1]$

Diffuse reflection

Light reflected from rough surfaces, scattered in many directions

If no preferred direction of reflection, the surface is **Lambertian**



Diffuse + ambient reflection

Proportional to the cosine of the angle between the surface normal and the direction to the light source

Varies across the object's surface

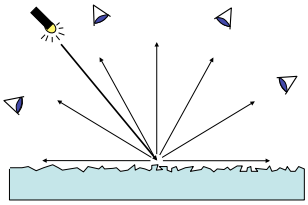
Diffuse illumination

Affected by

- The amount of light incident on the surface
- The amount of light reflected off of the surface to the eye

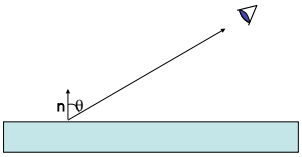
For purely diffuse (Lambertian) surfaces

- The amount of light reflected from the surface and seen by the eye is independent of the reflection direction



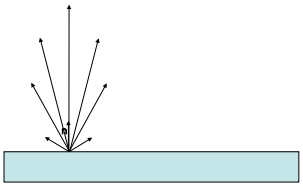
Lambertian surfaces

Amount of light seen by a viewer is independent of viewing direction – why?



Lambert's cosine law

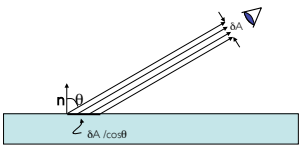
The amount of light reflected in a direction θ from the surface normal n is proportional to cos θ



Lambert's cosine law

The amount of light reflected in a direction θ from the surface normal n is proportional to cos θ

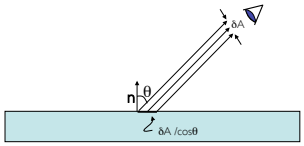
However, the area of the surface seen by the viewer is inversely proportional to cos θ



Lambert's cosine law

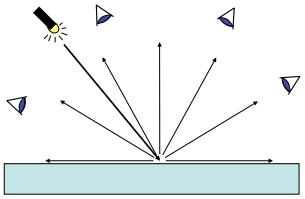
The amount of light reflected in a direction θ from the surface normal n is proportional to cos θ

However, the area of the surface seen by the viewer is inversely proportional to cos θ



Lambertian surfaces

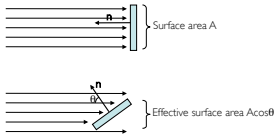
These two effects cancel each other out …



Diffuse reflection

A surface perpendicular to the light source appears brighter than one tilted away

The number of incident rays on a patch depends on the angle of the patch to the light direction, $\boldsymbol{I = I_L \cos\theta}$



Phong: Diffuse component

$$I = k_a I_a + k_d I_L(\mathbf{n}\cdot\mathbf{l}) + k_s I_L(\mathbf{r}\cdot\mathbf{v})^s$$

Diffuse

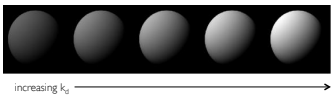
$\boldsymbol{I_L}$: the color and intensity of a point light source

$\boldsymbol{k_d}$: a material property of the object, its diffuse reflection coefficient, $k_d \in [0,1]$

$\boldsymbol{n}$: the surface's outward facing normal

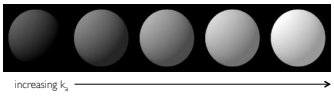
$\boldsymbol{l}$: the light direction vector, points from the surface to the light source

Diffuse reflection



increasing k_d

Diffuse reflection
+ Ambient reflection



increasing k_a

Diffuse + ambient + specular reflection

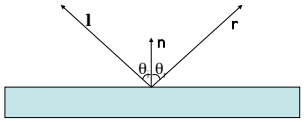
Provides specular highlights for shiny surfaces

Brightest where the surface normal is almost the same as the direction to the light source

Specular reflection

A perfectly shiny material (mirror) reflects in only one direction

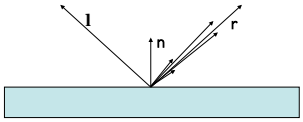
angle of reflectance θ_r = angle of incidence θ_i



Specular reflection

Even most shiny surfaces (metal) are not perfect reflectors

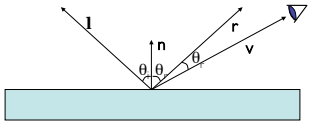
Less shiny surfaces reflect in a broader range of directions



Specular reflection

Phong developed a model for non-perfect reflectors
Amount of specular reflection decreases with increasing θ_r

Approximate the rapid falloff by $(\cos\theta_r)^s$
s is the material's specular reflection exponent



Phong: Specular component

$$I = k_a I_a + k_d I_L(\mathbf{n}\cdot\mathbf{l}) + k_s I_L(\mathbf{r}\cdot\mathbf{v})^s$$

Specular

I_L : color and intensity of the light source

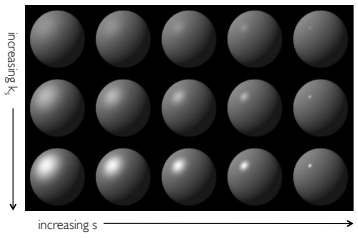
k_s : specular reflection coefficient, $k_s \in [0,1]$

s : specular exponent

r : reflection vector

v : view direction vector

Specular + diffuse + ambient reflection



Phong illumination model

$$I = k_a I_a + k_d I_L(\mathbf{n}\cdot\mathbf{l}) + k_s I_L(\mathbf{r}\cdot\mathbf{v})^s$$

Obj's specular exponent
Direction to the camera (eye)
Direction of (perfect) reflection)
Object's specular coefficient
Direction to light source
Object's outward facing normal
Color of light source
Object's diffuse component
Scene's ambient light color
Object's ambient coefficient

Other considerations

Light source attenuation
- Roughly follows the inverse square law, $I = 1/d^2$

Colored lights and surfaces
- Calculate separately for R, G and B intensities

Other considerations

Light source attenuation
- Roughly follows the inverse square law, $I = 1/d^2$

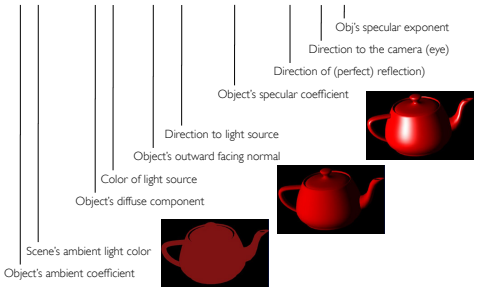
Colored lights and surfaces
- Calculate separately for R, G and B intensities

Multiple light sources
- Sum contributions for diffuse and specular components

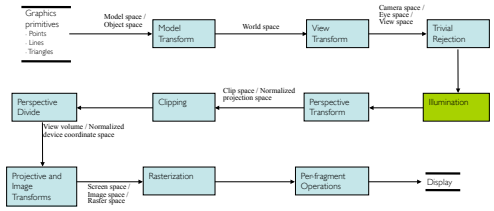
Color overflow
- Control total illumination
- Clip colors
- Determine max color intensity and scale accordingly

Phong illumination model

$$I = k_a I_a + k_d I_L(\mathbf{n}\cdot\mathbf{l}) + k_s I_L(\mathbf{r}\cdot\mathbf{v})^s$$



Vertex illumination



Graphics Processing Unit (GPU)

PC hardware dedicated to graphics computation

Massively parallel single instruction, multiple data (SIMD) processor

Several ALUs perform the same operation on different piece of data

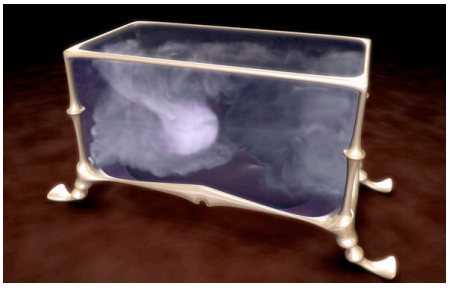
Performance pushed by game industry

Graphics Processing Unit (GPU)

Data streams
- Collection of records requiring similar computation
- Provide data parallelism

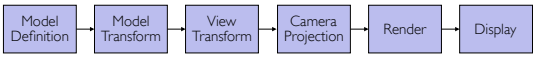
Kernels
- Functions applied to each element in stream
- Few dependencies between stream elements

NVIDIA GeForce 8800 "Human Head" Demo

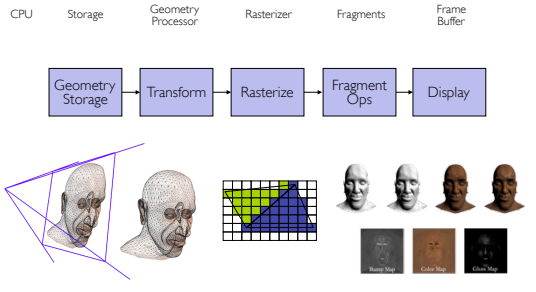


NVIDIA GeForce 8800 "Box of Smoke" Demo

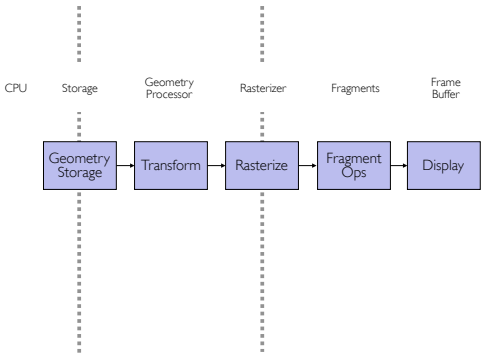
ATI Radeon HD 4800 "Froblins" Demo



Adapted from NVIDIA

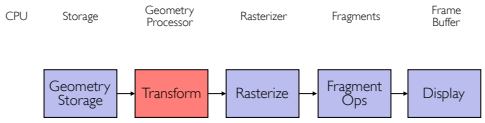


Adapted from NVIDIA



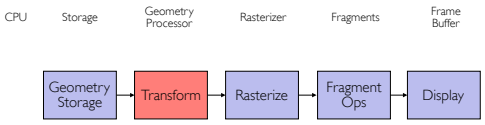
Adapted from NVIDIA

Transformations

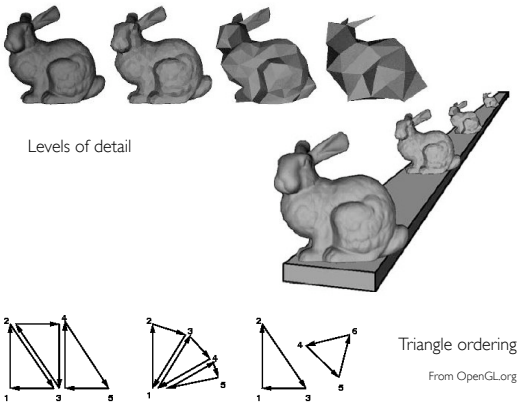


Coordinate transformations
Vertex processor: per-vertex lighting

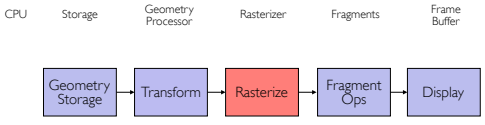
Transform bottlenecks



e.g.
- Many vertices
- Many computations per vertex
- Inefficient vertex caching



Rasterization

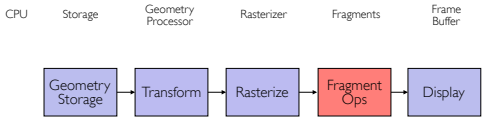


Convert geometric (vertex) representation to image representation
Interpolate per-vertex values

Rarely the bottleneck – why?

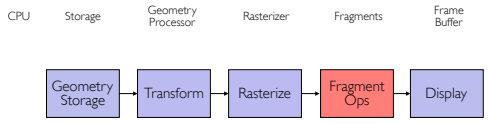
What about texture?

Fragment operations



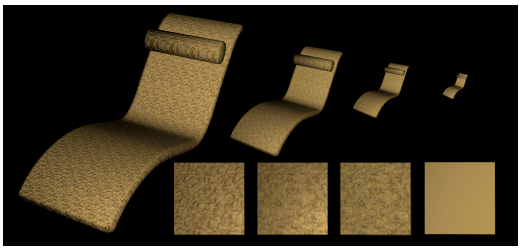
Shading
Fragment processor: Compute a color for each pixel

Fragment operation bottlenecks

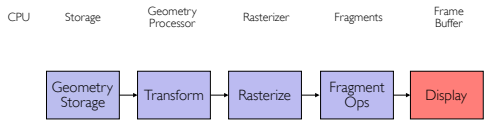


e.g.
- Many fragments
- Many computations per fragment

Shader LOD



Framebuffer bottlenecks



An efficient GPU workload

Thousands of independent pieces of work
- Many ALUs on many cores
- Massive interleaving

Amenable to instruction stream sharing

Compute-heavy
- High ratio of math operations to memory access

Convergence?

GPGPU

General purpose computation on the GPU

Computational problems traditionally handled by CPU
- Map to graphics rendering pipeline

Designed to process vertices/fragments in parallel
- **Stream processing**
  - Data parallelism
  - Run a single kernel on many records in the stream