Illumination and Shading

Joaquim Madeira

March 2024

Topics

- Motivation
- Phong's Illumination model
- Shading Methods
- Three.js – Light Sources & Shading

MOTIVATION

Modeling vs Rendering

- Modeling
  - Create models
  - Apply materials to models
  - Place models around scene
  - Place lights in the scene
  - Place the camera

YouTube Demo

- Rendering
  - Take picture with the camera

[van Dam]

3D visualization pipeline

- Create a scene and instantiate models
  - Position, orientation, size
- Establish viewing parameters
  - Camera position and orientation
- Perform clipping
- Compute illumination and shade polygons
- Project into 2D
- Rasterize

3D visualization pipeline

- Each object is processed separately
  - 3D triangles
- Object / triangle inside the view volume ?
  - No : go to next object / triangle
- Rasterization
  - Compute the location on the screen of each triangle
  - Compute the color of each pixel

Lighting or Illumination

- The process of computing the intensity and color of a sample point in a scene as seen by a viewer

- It is a function of the geometry of scene
  - Models, lights and camera, and their spatial relationships

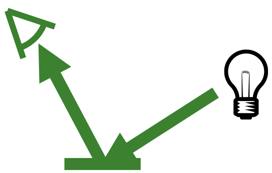
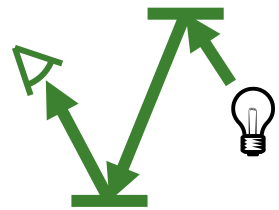
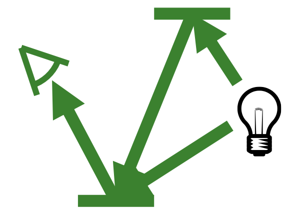
- And of material properties
  - Reflection, absorption, …

Shading

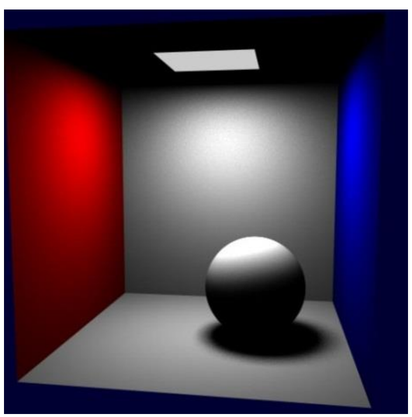
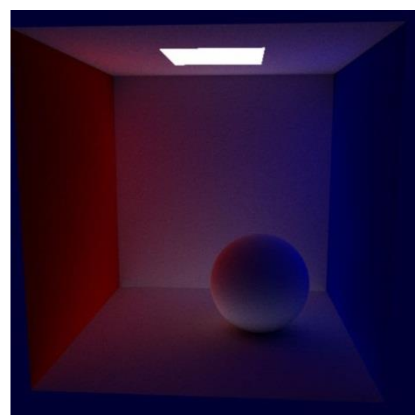
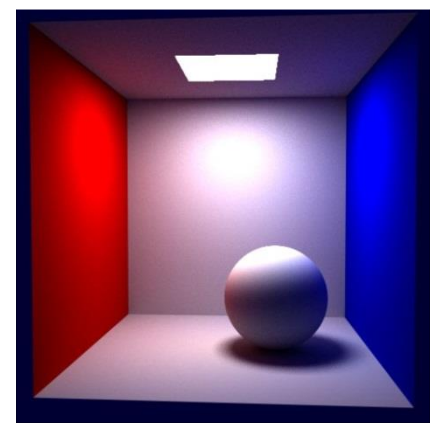
- The process of *interpolation* of color at points in-between those with known lighting or illumination
  - Vertices of triangles in a mesh
- Used in many real time graphics applications (e.g., games)
  - Calculating illumination at a point is usually expensive !
- BUT, in ray-tracing only do lighting for samples / pixels
  - More sophisticated / demanding
  - Based on pixels (or sub-pixel samples for super-sampling)
  - No shading rule

COMPUTING ILLUMINATION

Global Illumination

Direct illumination + Indirect illumination = Total illumination

[Andy Van Dam]

Local vs Global Illumination

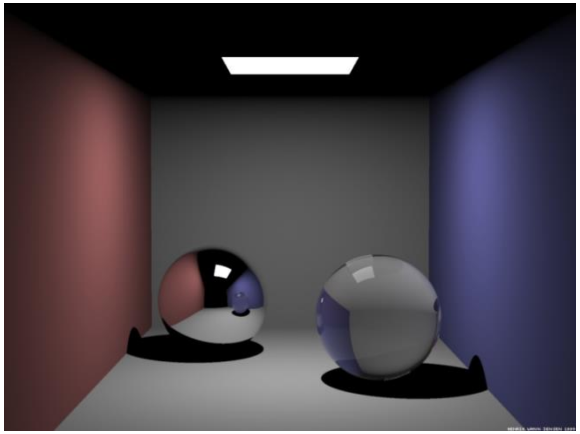
Direct (diffuse + specular) lighting + indirect specular reflection

Full global illumination

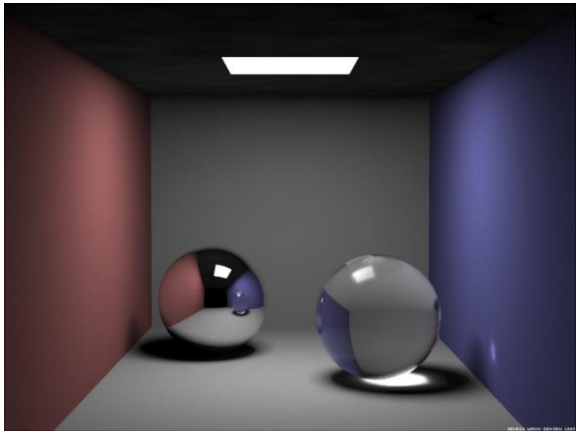
[Andy Van Dam]

Global Illumination – Examples

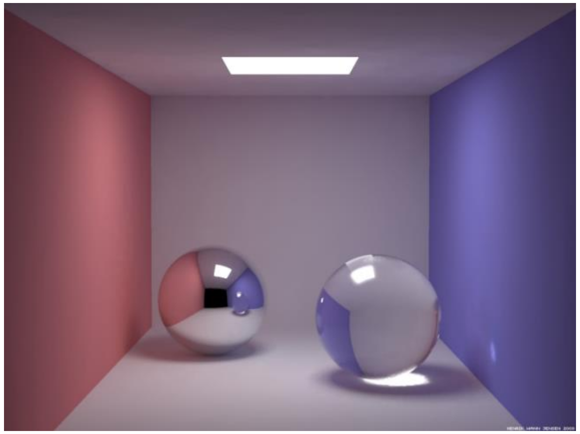
- Take into account global information of both direct (from emitters) and indirect illumination (inter-object reflections)
- Different approximations
  - Advantages and disadvantages; resource requirements
  - More computation gives better results…



Direct illumination + specular reflection
Ray trace



+ soft shadows and caustics
Ray trace + caustic photon map



+ diffuse reflection (color bleeding)
Ray trace + caustic and diffuse photon maps

http://graphics.ucsd.edu/~henrik/images/global.html [Andy Van Dam]

Light Transport Simulation

- Evaluate illumination with enough samples to produce final images without any guessing / shading
- Often used for high quality renderers, e.g., those used in FX movies
  - Can take days for a single frame, even on modern render farms
- Some implementations can run in real time on the GPU
  - But more complex lighting models that are difficult to parallelize are still run on the CPU
- Many simulations use stochastic sampling
  - Path tracing, photon mapping, Metropolis light transport

Polygon Rendering

- Evaluate illumination at several samples
- Shade (using a shading rule) in between to produce pixels in the final image
- Often used in real-time applications such as computer games
- Done in the GPU
- Lower quality than light transport simulation !!
  - But satisfactory results with various additions such as maps (bump, displacement, environment)

Polygon Rendering

- Get realistic images by :
  - using perspective projections of the scene models
  - applying natural illumination effects on the visible surfaces

- Natural illumination effects are obtained using:
  - an illumination model – allows computing the color to be assigned to each visible surface point
  - a surface-rendering method that applies an illumination model and assigns a color to every pixel

Polygon Rendering

- Photorealistic images require:

  - a precise representation of the properties of each surface

  - a good description of the scene's illumination

- And might imply modeling:
  - surface texture
  - transparency
  - reflections
  - shadows
  - etc.

Polygon Rendering

- Compute surface color based on
  - Type and number of light sources
  - Illumination model
    - Phong: ambient + diffuse + specular components
  - Reflective surface properties
  - Atmospheric effects
    - Fog, smoke
- Polygons making up a model surface are shaded
  - Realistic representation

Polygon Rendering

- Illumination models used in Computer Graphics

  - are often an approximation to the Laws of Physics
  - that describe the interaction light-surface

- There are different types of illumination models

  - simple models, based on simple photometric computations (to reduce the computational cost)

  - more sophisticated models, based on the propagation of radiant energy (computationally more complex)

LIGHT SOURCES

Light Sources

- Objects radiating light and contributing to the illumination of scene objects

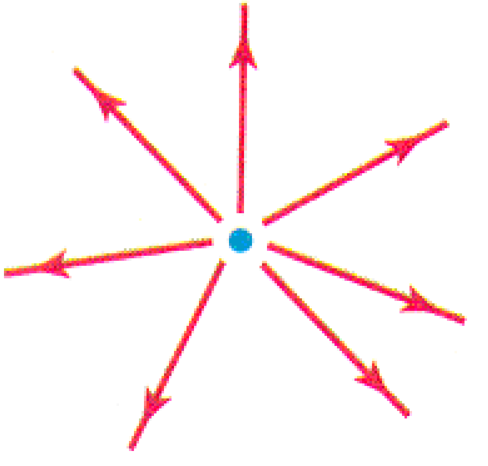
- Can be defined by several features:

  - Location

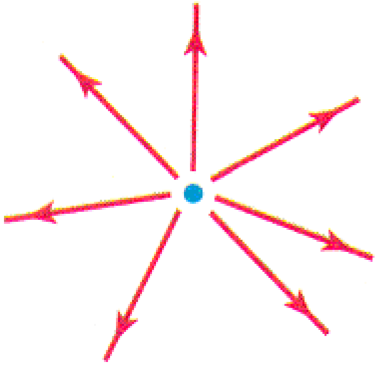
  - Color of emitted light

  - Emission direction

  - Shape

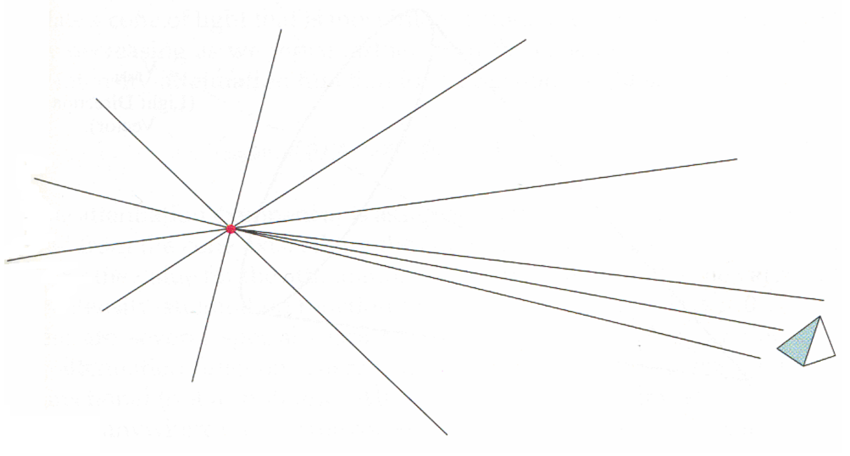


Simplified Light Sources



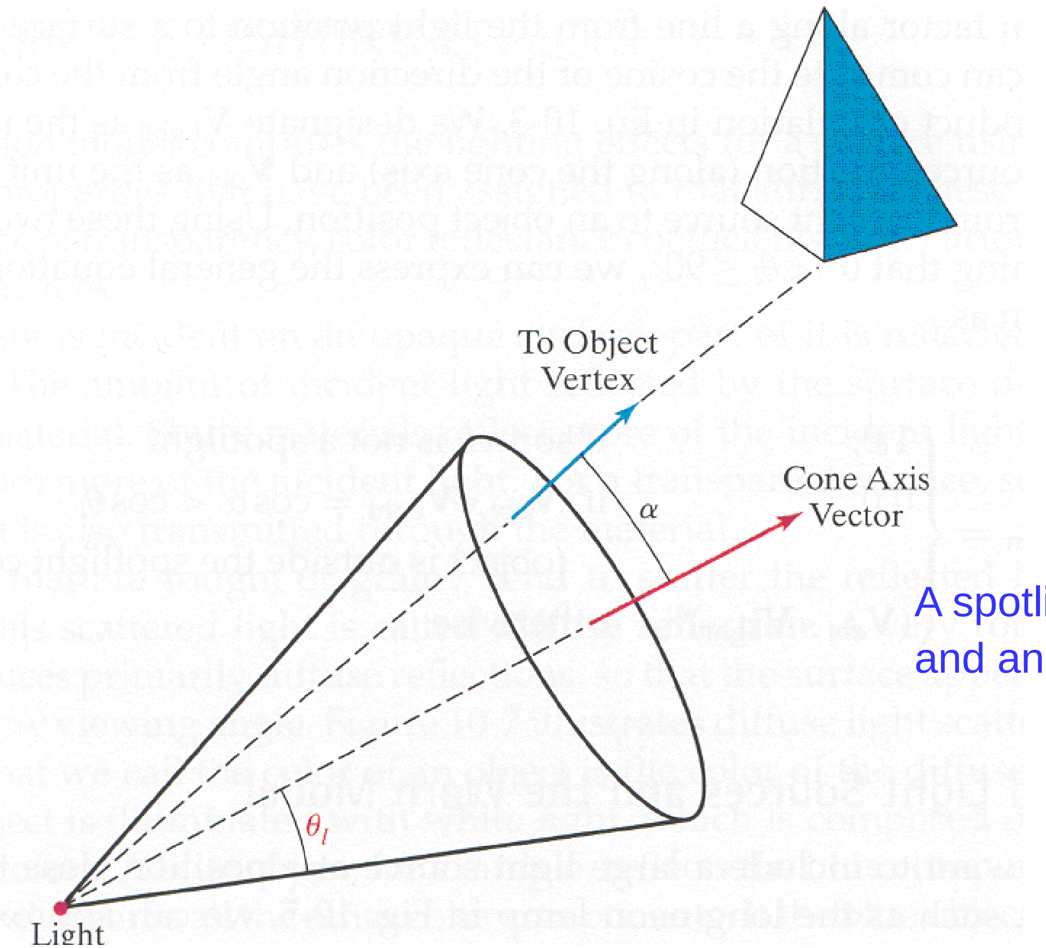
Isotropic point light source

Light source at an indefinite distance



Rays emitted by a light source at a far-away location can be considered as parallel

Spotlight



A spotlight is defined by a direction and an emission angle

SURFACE FEATURES

Surface Features

- An illumination model takes into account a surface’s optical properties:

  - reflection coefficients for each color

  - degree of transparency

  - texture parameters

- When light is incident on an opaque surface:

  - part is absorbed

  - part is reflected

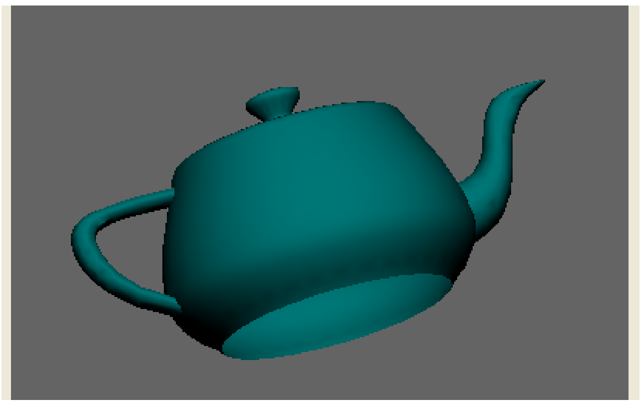
Surface Features

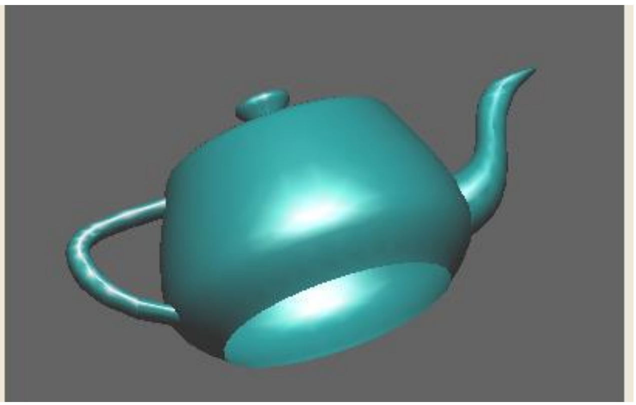
Surface Features

- The amount of reflected light depends on the surface's features

  - Shiny surfaces reflect more light

  - Mate / dull surfaces reflect less light



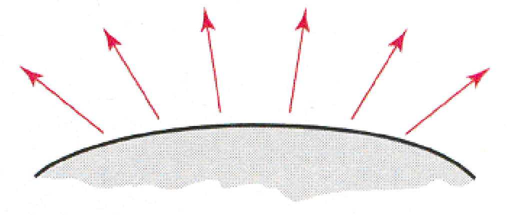


- Transparent surfaces transmit some light

Rough vs Smooth Surfaces

- Rough surfaces tend to spread the reflected light in all directions

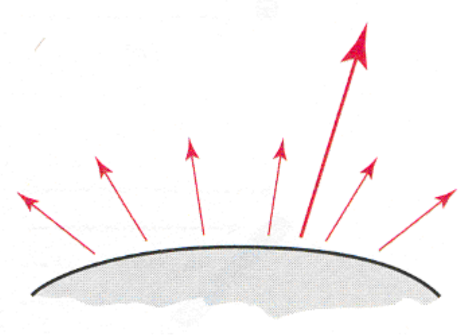
  - **diffuse reflection**



And look equally shiny from any viewpoint

- Smooth surfaces reflect more light in particular directions

  - **specular reflection (*highlight*)**



And present some shinier areas

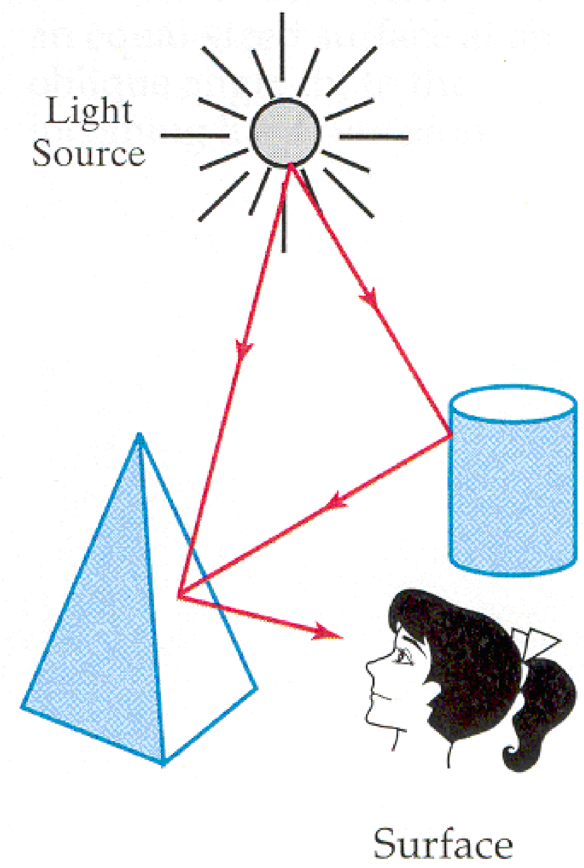
Inter-Reflections

- Inter-reflections among scene objects might be approximated by

  - an **ambient illumination** component

- A surface might not be directly illuminated and still be visible, due to light reflected by other objects in the scene

- The amount of light reflected by a surface is the sum of all contributions from the light sources and the ambiente illumination



PHONG'S REFLECTION MODEL

Basic Illumination Models

- Sophisticated illumination models precisely compute the interaction efects between the radiating energy and the surface material

- Basic models use approximations to represent the physical processes producing the illumination effects

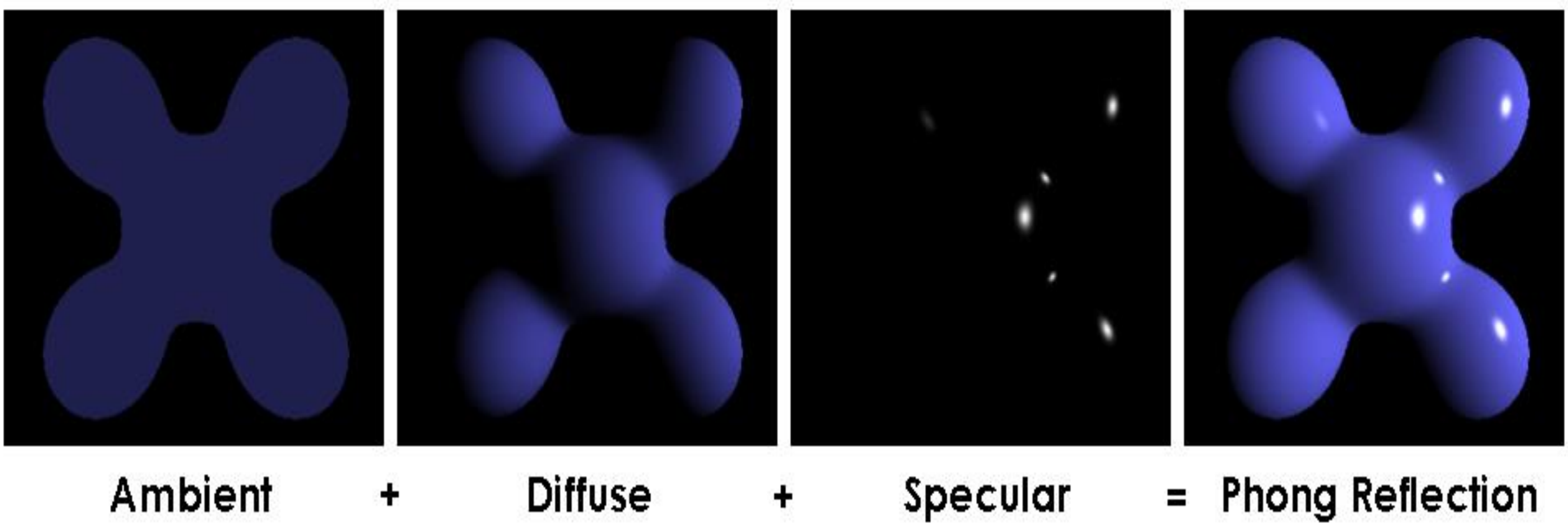
- The empirical model described next computes good enough results for most situations and includes:

  - ambient illumination

  - diffuse reflection

  - specular reflection

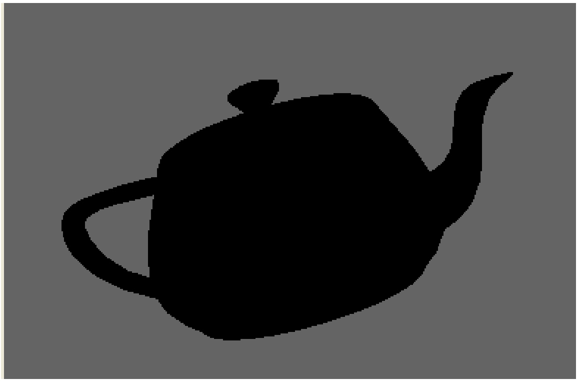
Phong reflection model – 1973

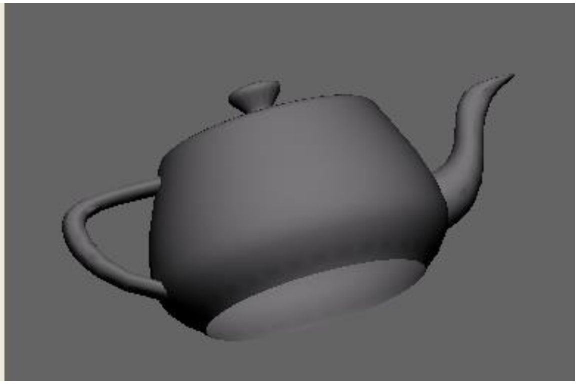


[Wikipedia]

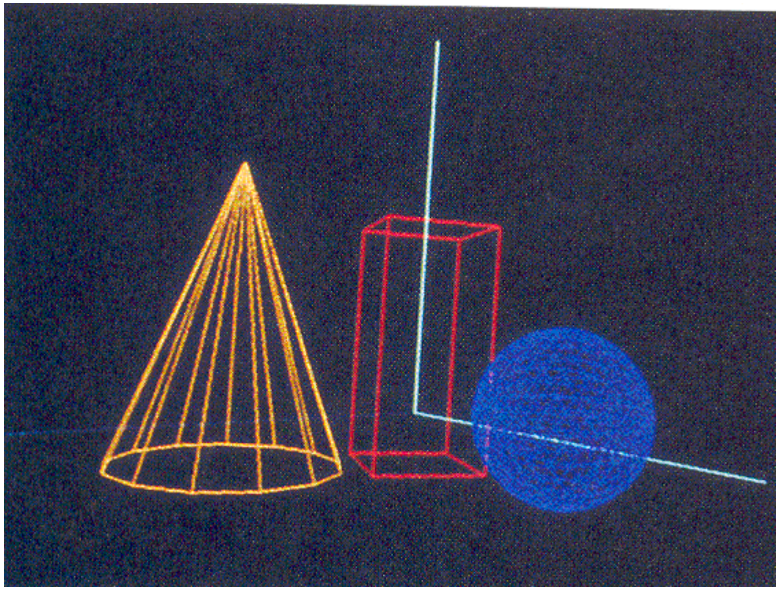
Phong Model – Ambient illumination

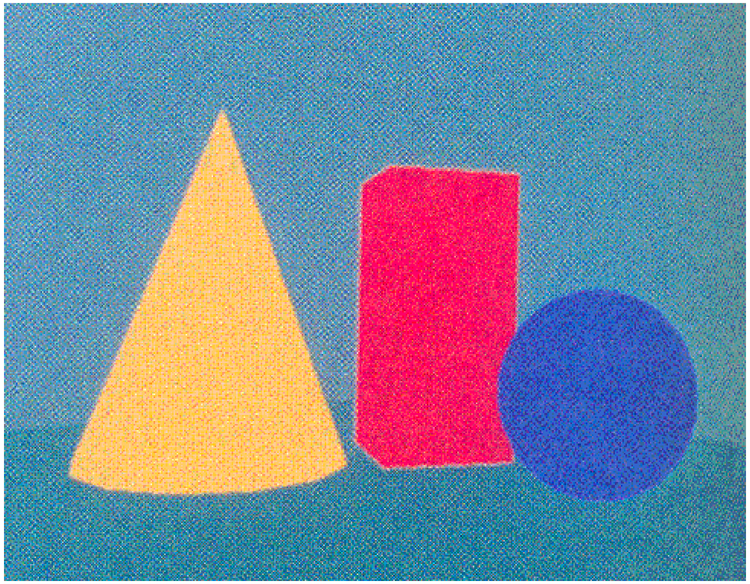
- Constant illumination component for each model
- Independent from viewer position or object orientation !
- Take only material properties into account !





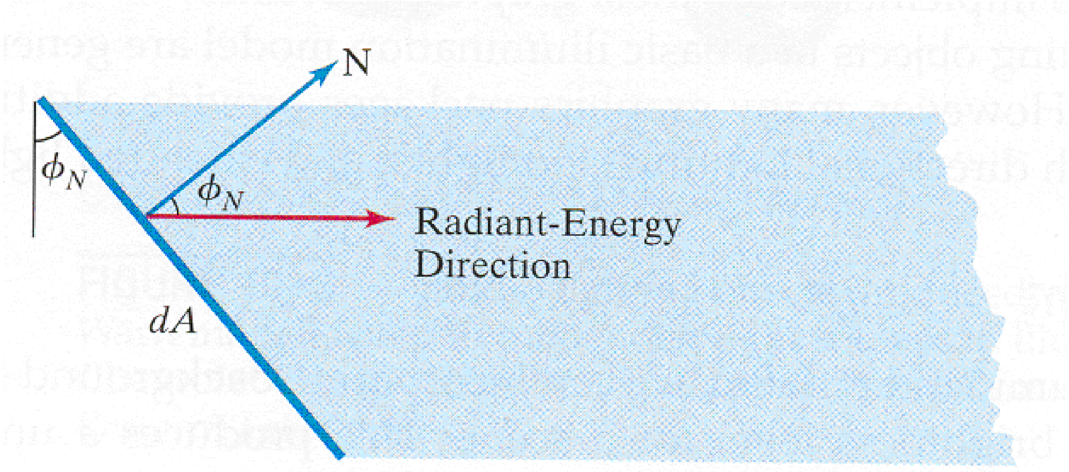
Phong Model – Ambient illumination





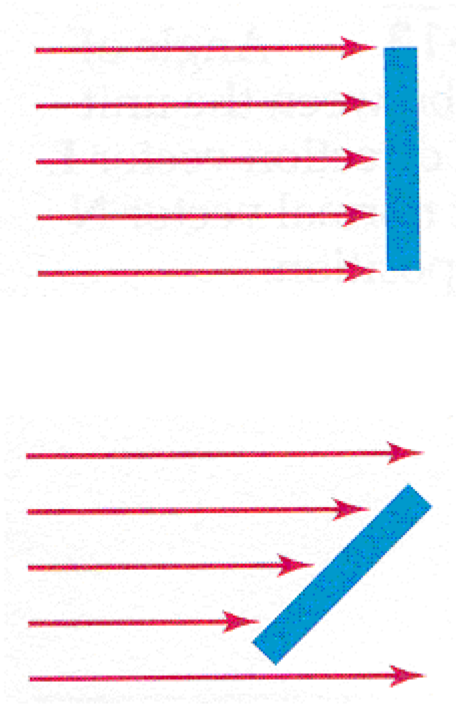
Diffuse Reflection

- It is considered that incident light is spread with equal intensity in all directions, regardless of the viewing direction
- Surfaces with that feature are called Lambertian reflectors or ideal diffuse relectors
- The intensity of reflected light is computed by Lambert's Law:



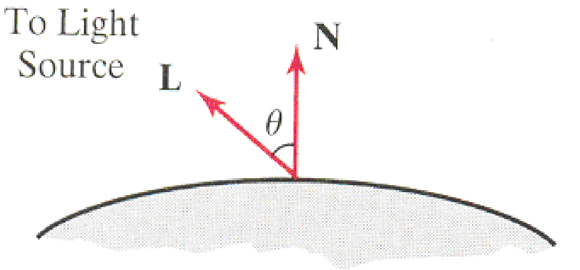
Diffuse Reflection

- There is, at least, one point light source (usually located at the viewpoint)
- The amount of incident light depends on the surface orientation regarding the direction of the light source
- A surface that is orthogonal to the light direction is “more illuminated” than an oblique surface, with the same area



Phong Model – Diffuse reflection

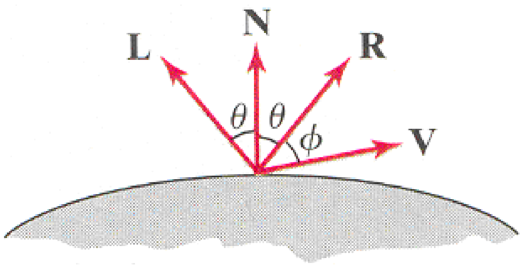
$$I_{l,\text{diff}} = \begin{cases} k_d I_l (\mathbf{N} \bullet \mathbf{L}), & \mathbf{N} \bullet \mathbf{L} > 0 \\ 0.0, & \mathbf{N} \bullet \mathbf{L} \leq 0 \end{cases}$$



- Model surface is an ideal diffuse reflector
  - What does that mean ?
- No dependency from viewer position !
- Unit vectors !!

Phong Model – Specular reflection

- Te shinier areas, specular reflections or *highlights*, that can be seen on shiny surfaces, result from the reflection of most light around concentrated areas



- The specular reflection angle is equal to the incidence angle (relative to the normal)
- **R** is the unit vector defining the ideal specular reflection direction
- **V** is the unit vector defining the viewing diretion
- An ideal reflector reflects light only in the specular reflection direction (the viewer perceives the specular reflection only if **V** and **R** are coincident $\Phi = 0$)

Phong Model – Specular reflection

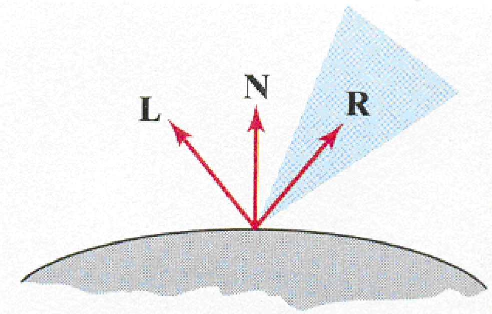
- Objects that are not ideal reflectors produce specular reflections in a finite set of directions around vector **R**

- Shiny surfaces have a narrow set of reflection directions

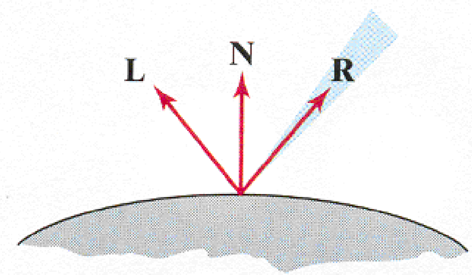
- The empirical Phong specular reflection model sets the intensity of the specular reflections as proportional to $\cos^{n_s}\phi$

$$I_{l,\text{spec}} = W(\theta)\, I_l \cos^{n_s}\phi$$

W(θ) is the specular reflection coefficient

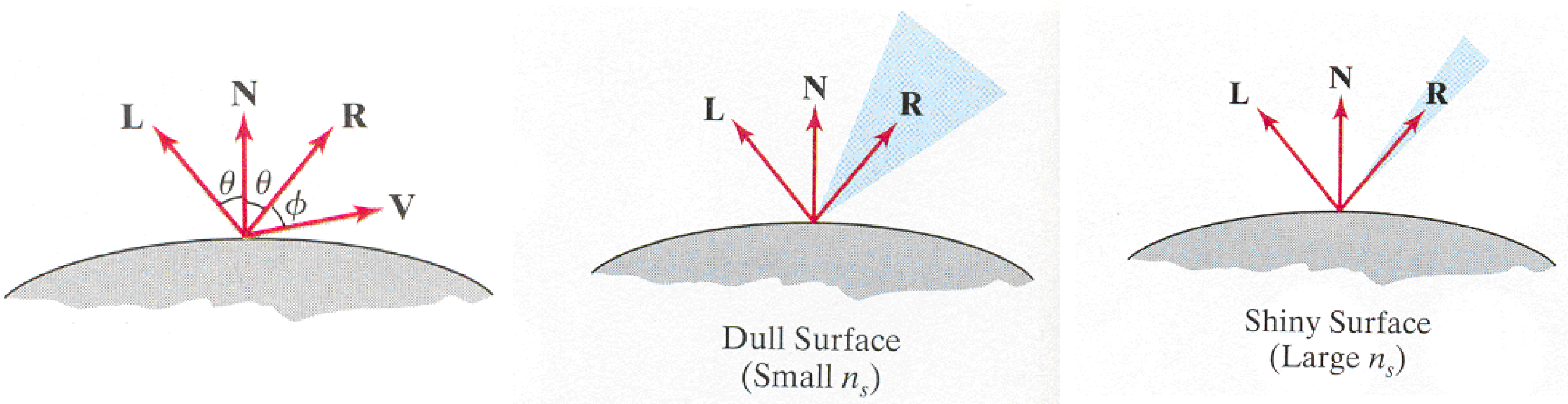


Dull Surface
(Small n_s)



Shiny Surface
(Large n_s)

Phong Model – Specular reflection

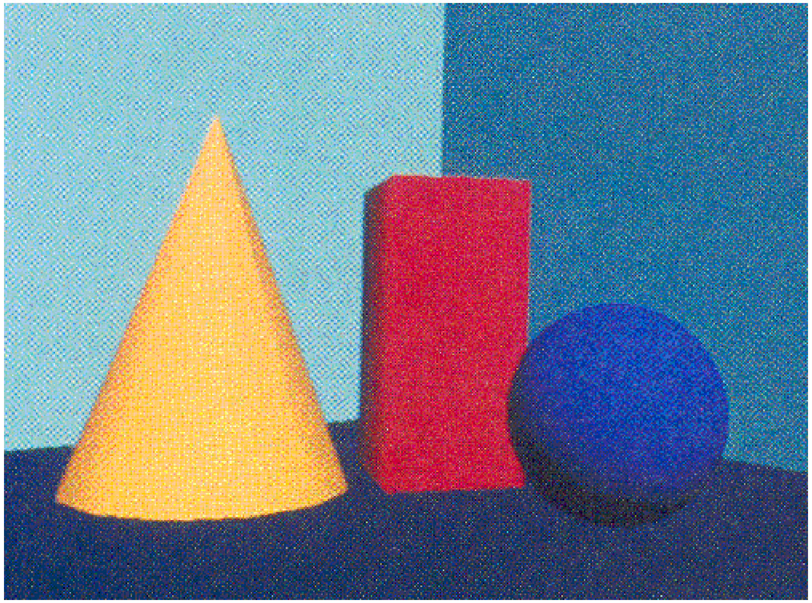


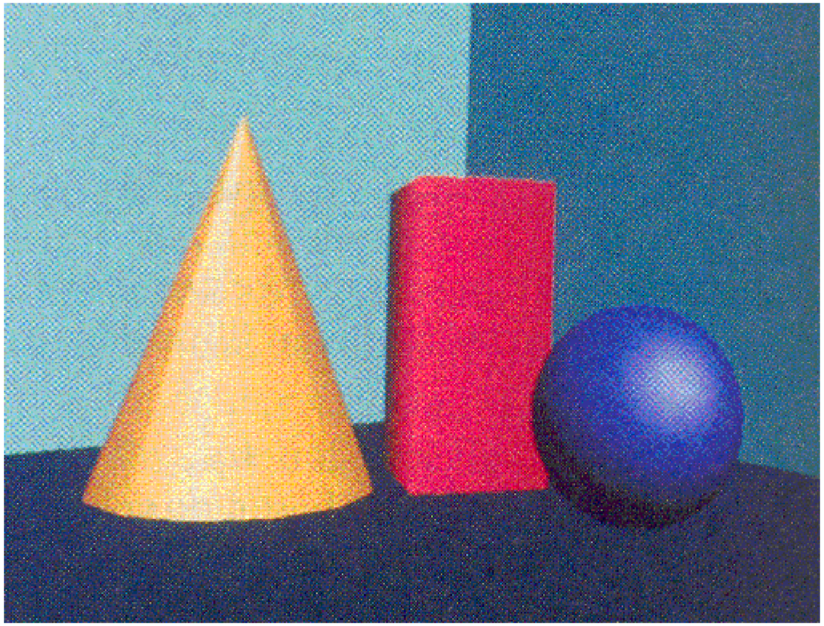
- Important for shiny model surfaces
  - How to model shininess ?

- Take into account viewer position !

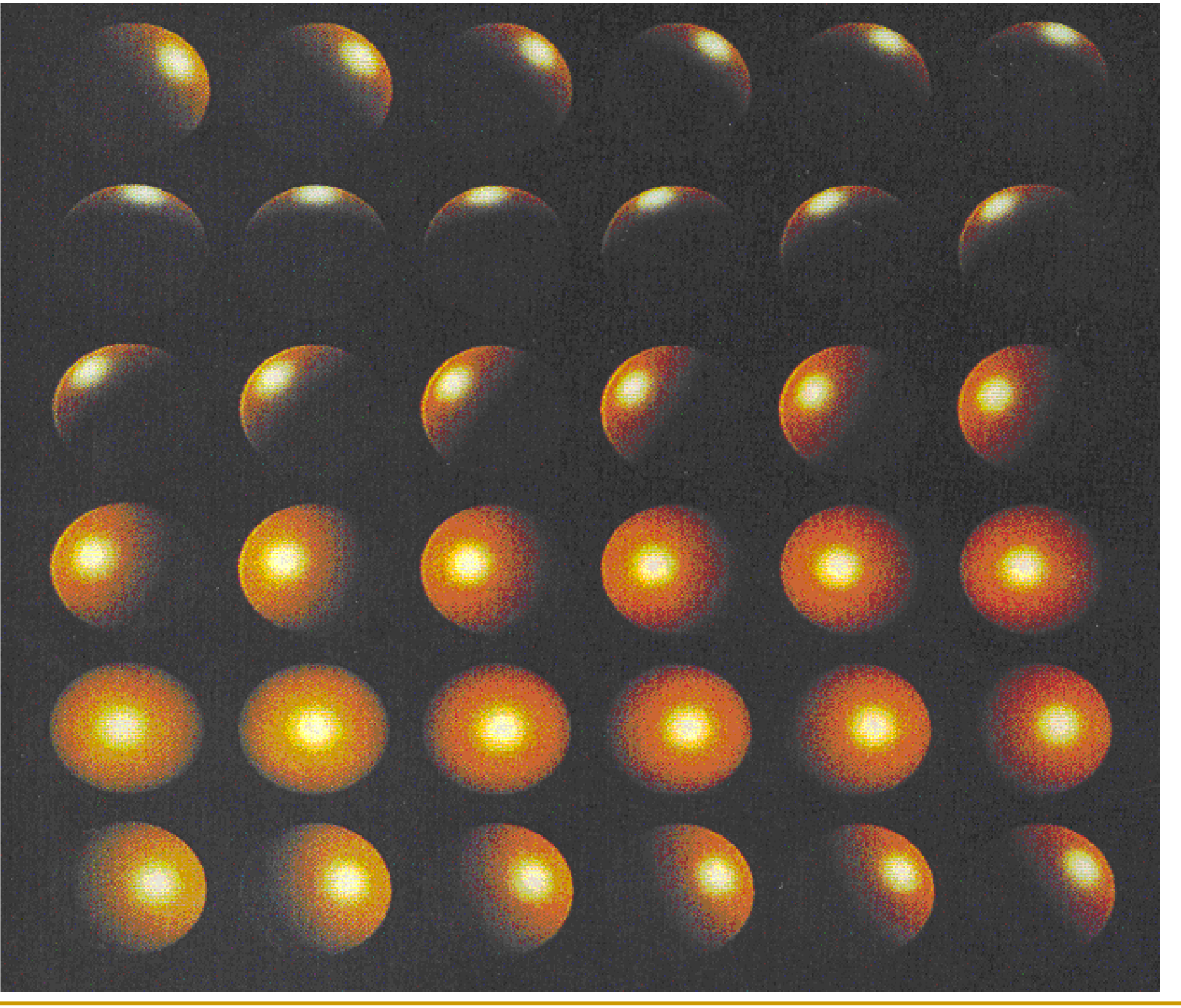
- Unit vectors !!

Phong Model – Specular reflection

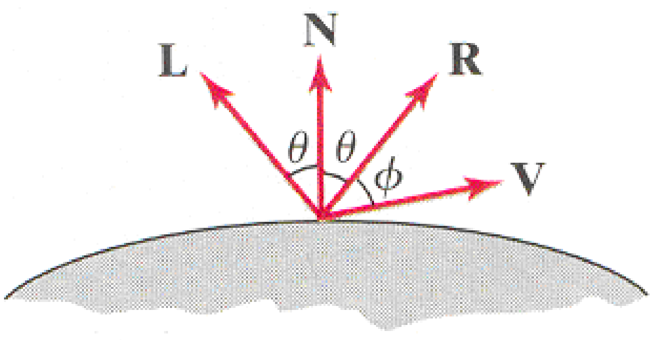




Phong Model – Specular reflection

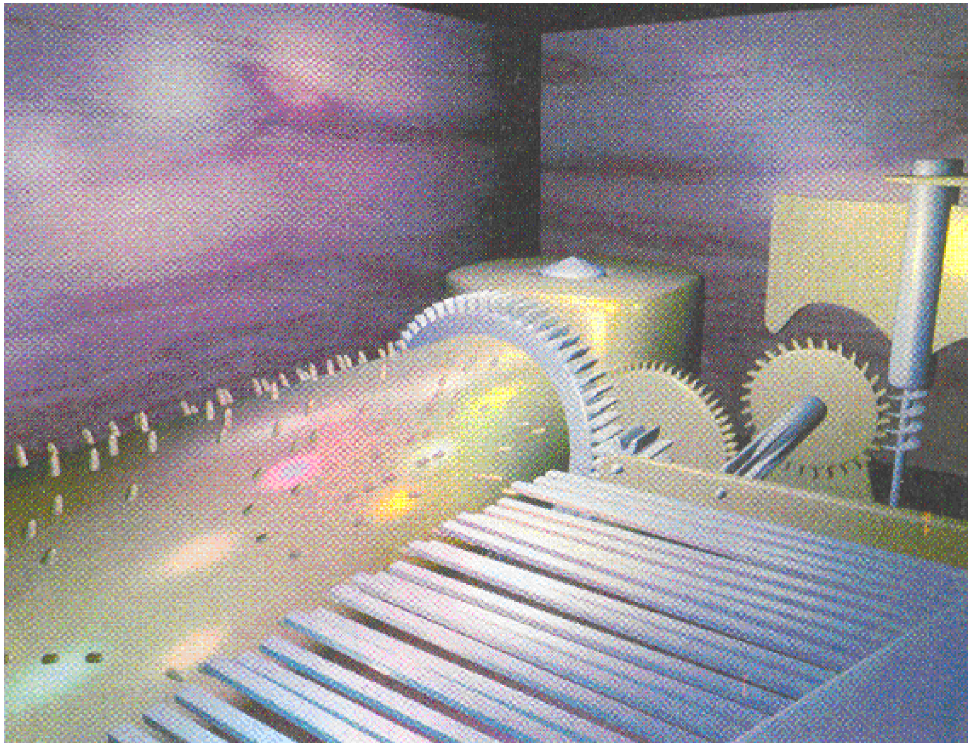


Phong Model – Specular reflection



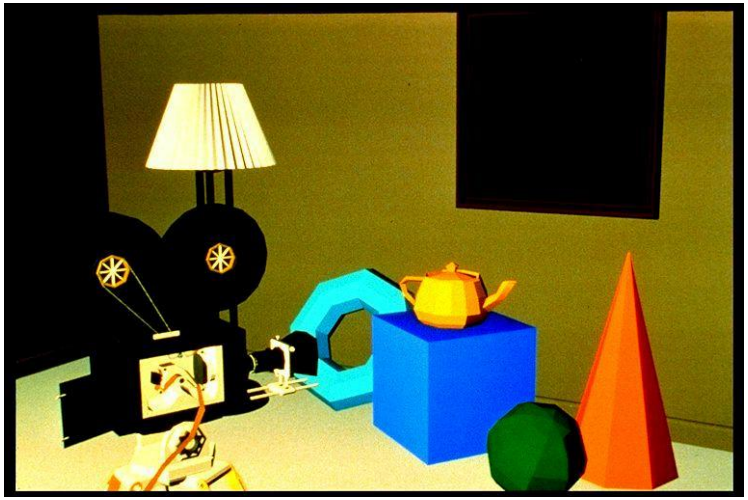
$$I_{l,\text{spec}} = \begin{cases} k_s I_l (\mathbf{V} \cdot \mathbf{R})^{n_s}, & \text{if } \mathbf{V} \cdot \mathbf{R} > 0 \quad \text{and} \quad \mathbf{N} \cdot \mathbf{L} > 0 \\ 0.0, & \text{if } \mathbf{V} \cdot \mathbf{R} < 0 \quad \text{or} \quad \mathbf{N} \cdot \mathbf{L} \leq 0 \end{cases}$$

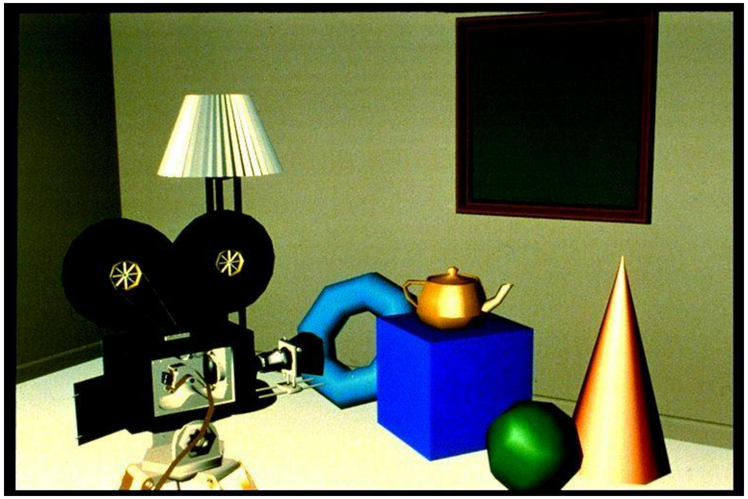
More than one light source

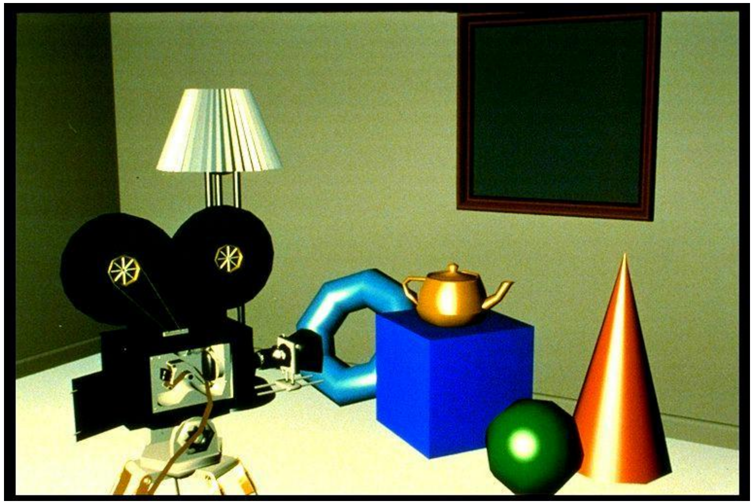


$$I = k_a I_a + \sum_{l=1}^{n} I_l [k_d (\mathbf{N} \cdot \mathbf{L}) + k_s (\mathbf{N} \cdot \mathbf{H})^{n_s}]$$

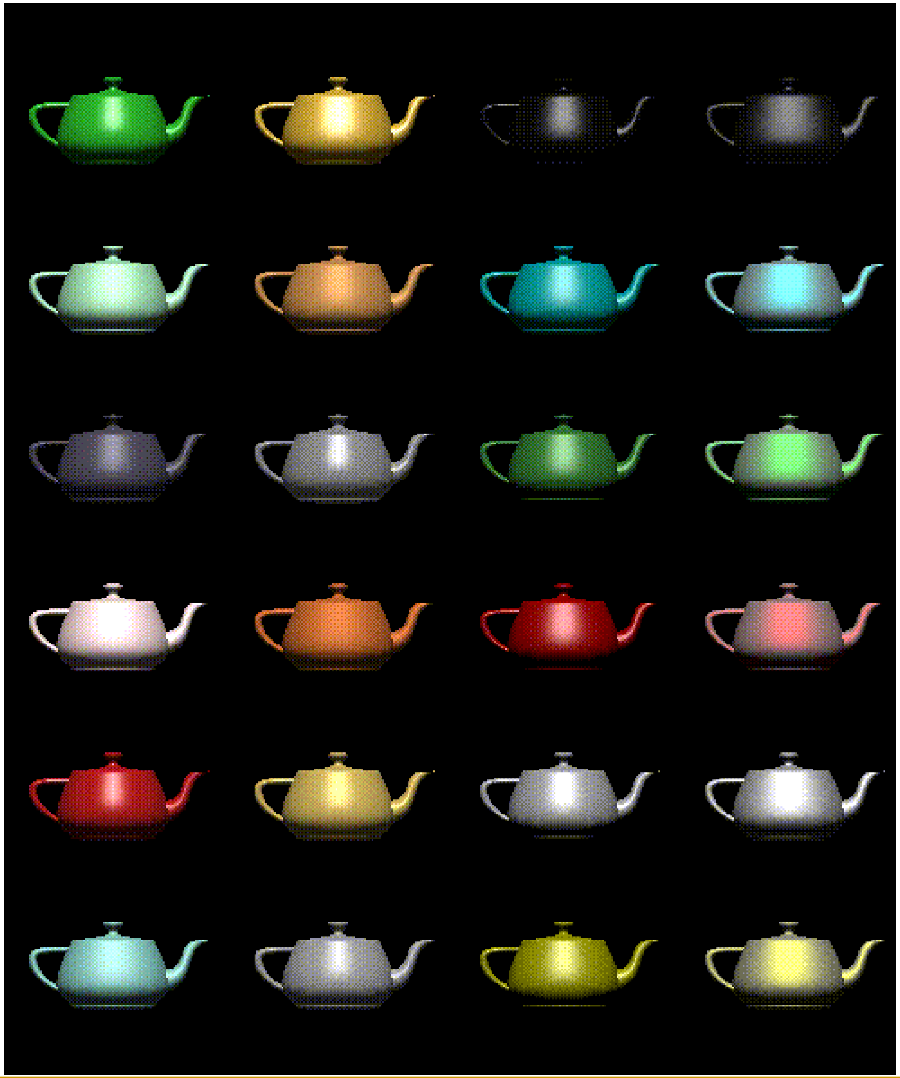
Can you spot the differences ?







Material properties



[OpenGL – The Red Book]

Material properties

Material	ambient (k_a)	diffuse (k_d)	specular (k_s)	specular exponent (m)	translucency (σ)
Brass	0.329412 0.223529 0.027451	0.780392 0.568627 0.113725	0.992157 0.941176 0.807843	27.8974	1.0
Bronze	0.2125 0.1275 0.054	0.714 0.4284 0.18144	0.393548 0.271906 0.166721	25.6	1.0
Polished Bronze	0.25 0.148 0.06475	0.4 0.2368 0.1036	0.774597 0.458561 0.200621	76.8	1.0
Chrome	0.25 0.25 0.25	0.4 0.4 0.4	0.774597 0.774597 0.774597	76.8	1.0
Copper	0.19125 0.0735 0.0225	0.7038 0.27048 0.0828	0.256777 0.137622 0.086014	12.8	1.0
Polished Copper	0.2295 0.08825 0.0275	0.5508 0.2118 0.066	0.580594 0.223257 0.0695701	51.2	1.0
Gold	0.24725 0.1995 0.0745	0.75164 0.60648 0.22648	0.628281 0.555802 0.366065	51.2	1.0
Polished Gold	0.24725 0.2245 0.0645	0.34615 0.3143 0.0903	0.797357 0.723991 0.208006	83.2	1.0

[https://people.eecs.ku.edu/~jrmiller/Courses/672/InClass/3DLighting/MaterialProperties.html]

MORE SOPHISTICATED REFLECTION MODELS

Cook & Torrance, 1982

Other Illumination Models

- Models you have seen before are empirical, but look okay
  - Phong Model
  - Blinn-Phong Model
- More complex models are physically based
  - Cook-Torrance Model
  - Oren-Nayer Model
- Performance vs. accuracy tradeoff

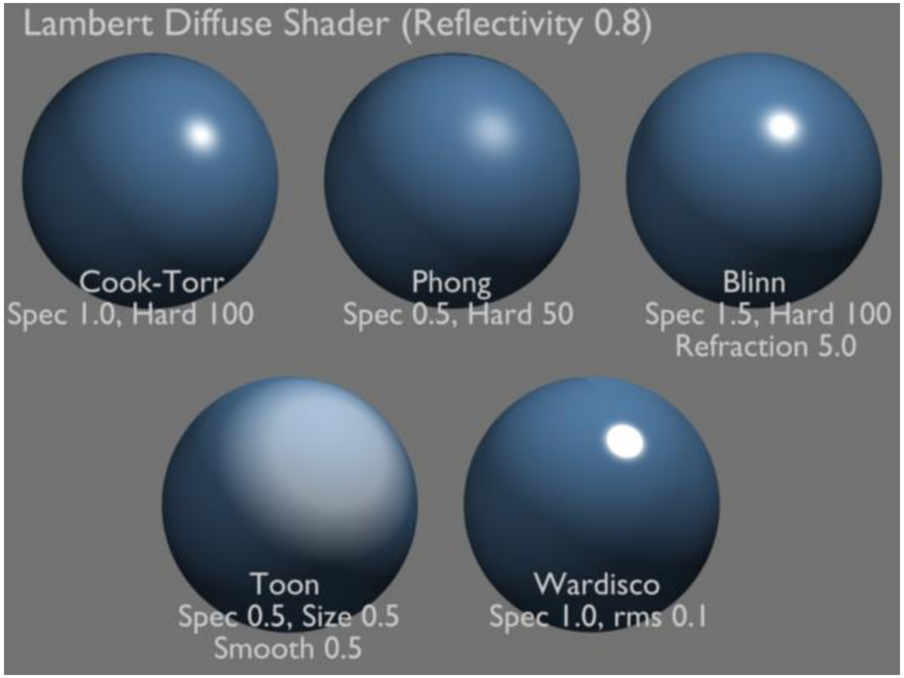


Image credit:
http://wiki.blender.org/index.php/File:Manual-Shaders-Lambert.png

[Andy Van Dam]

SHADING

Illumination and Shading

- Color values resulting from an illumination model can be used for surface rendering in different ways:

  - Compute the color values for each and every pixel corresponding to the projected surface

  - Compute the color values just for a few chosen pixels, and compute approximate color values for the rest

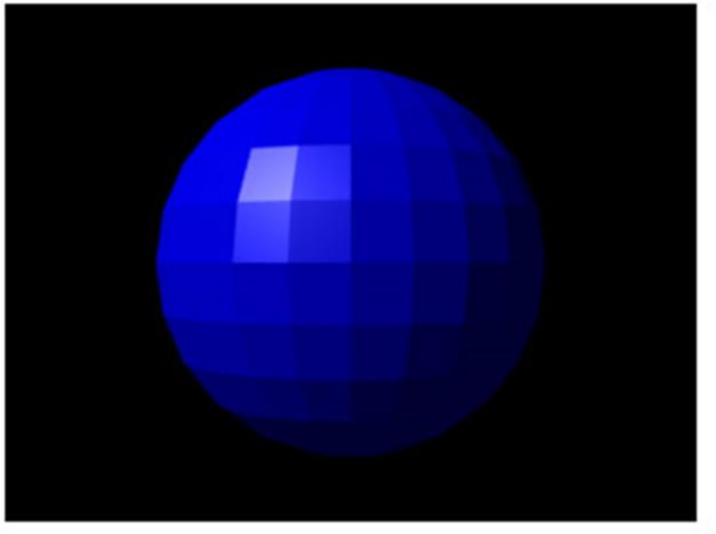
- In general, graphics APIs use *scan-line* algorithms and use the illumination model to compute color values at mesh vertices
  - Some interpolate color values along the *scan-lines*
  - Other use more precise methods

Illumination and Shading

- How to optimize?
  - Fewer light sources
  - Simple shading method
- BUT, less computations mean less realism
  - Wireframe representation
  - Flat-shading
  - Gouraud shading
  - Phong shading

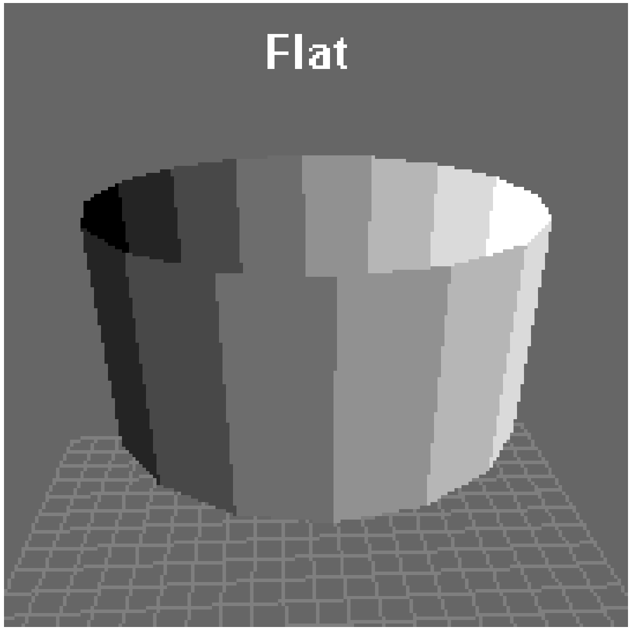
Flat-Shading

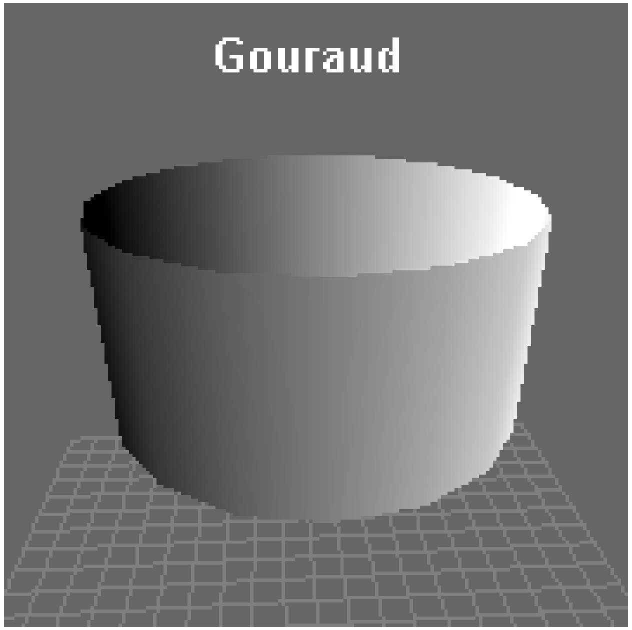
- For each triangle / polygon
  - Apply the illumination model just once !
  - All pixels have the same color
- Fast !
- But objects seem “blocky”



FLAT SHADING

Flat-Shading vs Gouraud Shading

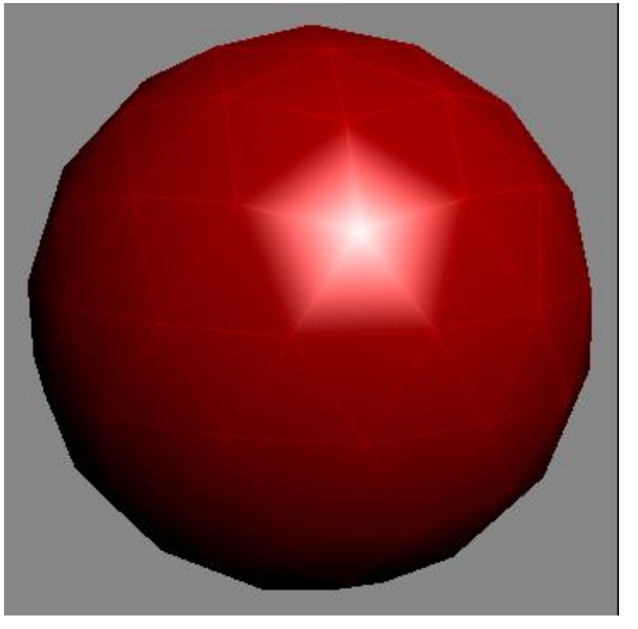




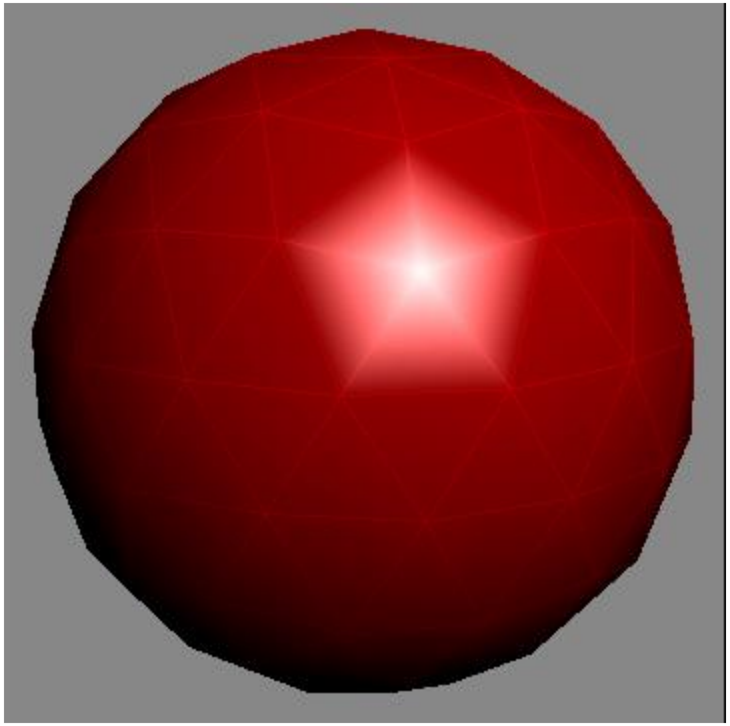
[Wikipedia]

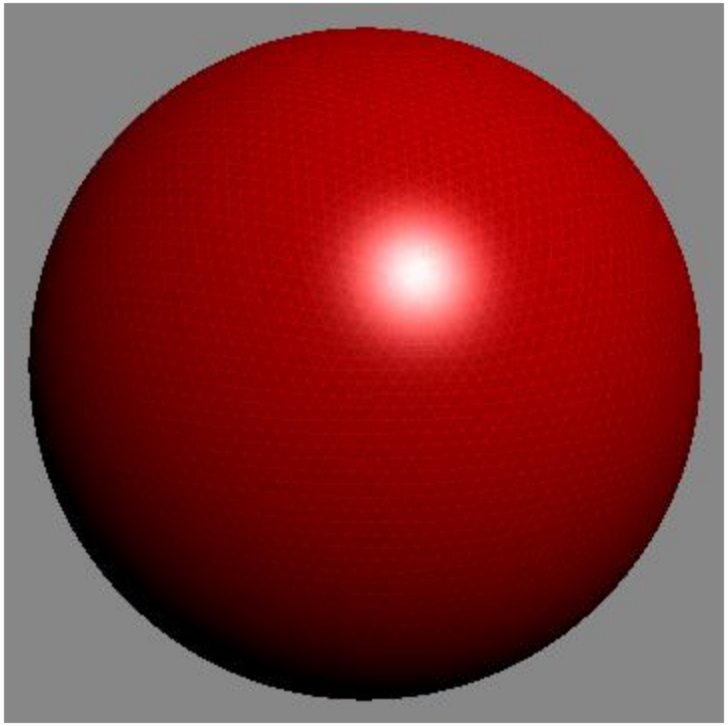
Gouraud Shading – 1971

- For each triangle / polygon
  - Apply the illumination model at each vertex
  - Interpolate color to shade each pixel
- Better than flat-shading
- Problems with highlights
- Mach-effect



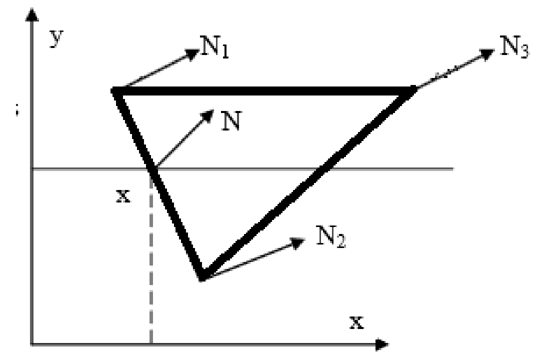
Gouraud Shading – More triangles !





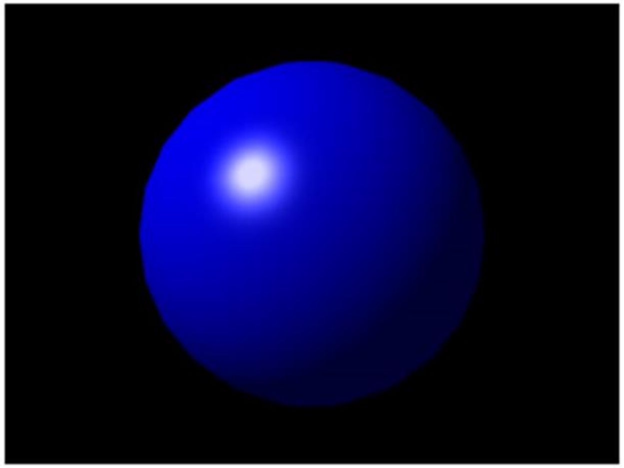
[Wikipedia]

Phong Shading – 1973



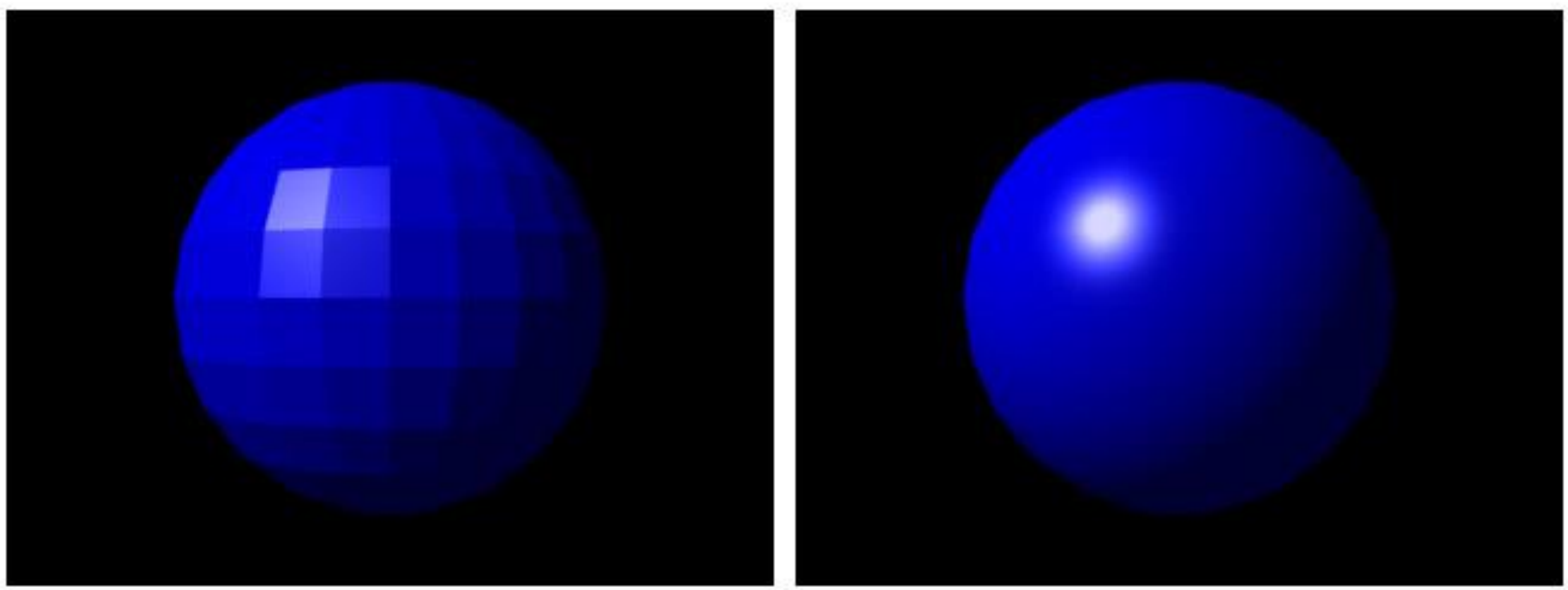
- For each triangle / polygon
  - Interpolate normal vectors across rasterized polygons

- Better than Gouraud shading
- BUT, more time consuming



PHONG SHADING

Flat-Shading vs Phong Shading



FLAT SHADING PHONG SHADING

[Wikipedia]

Possible References

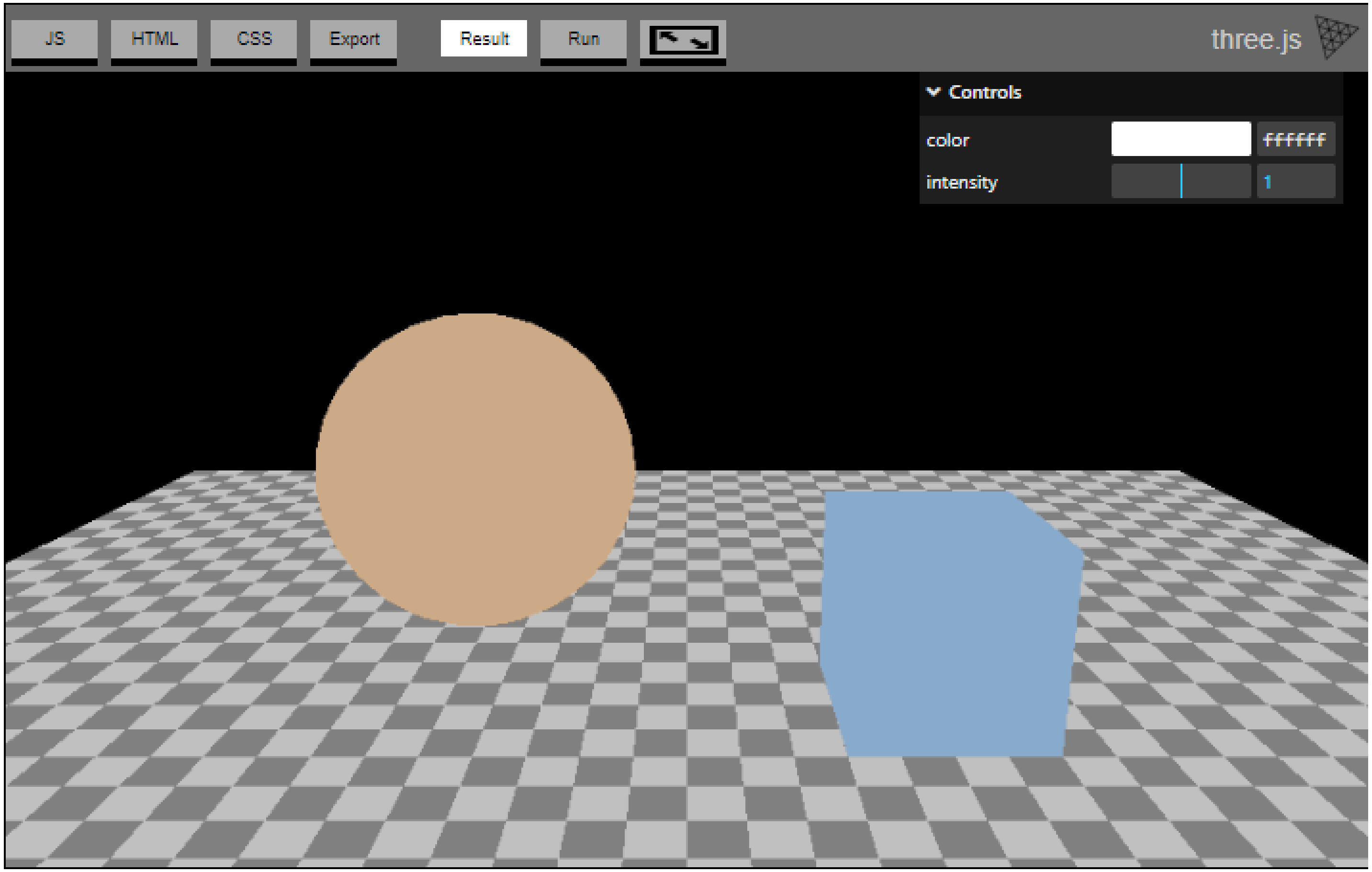
- The basics of illumination and shading are presented in any Computer Graphics book

- E. Angel and D. Shreiner. Interactive Computer Graphics, 7th Ed., Addison-Wesley, 2015

- J M Pereira, et al. Introdução à Computação Gráfica. FCA, 2018

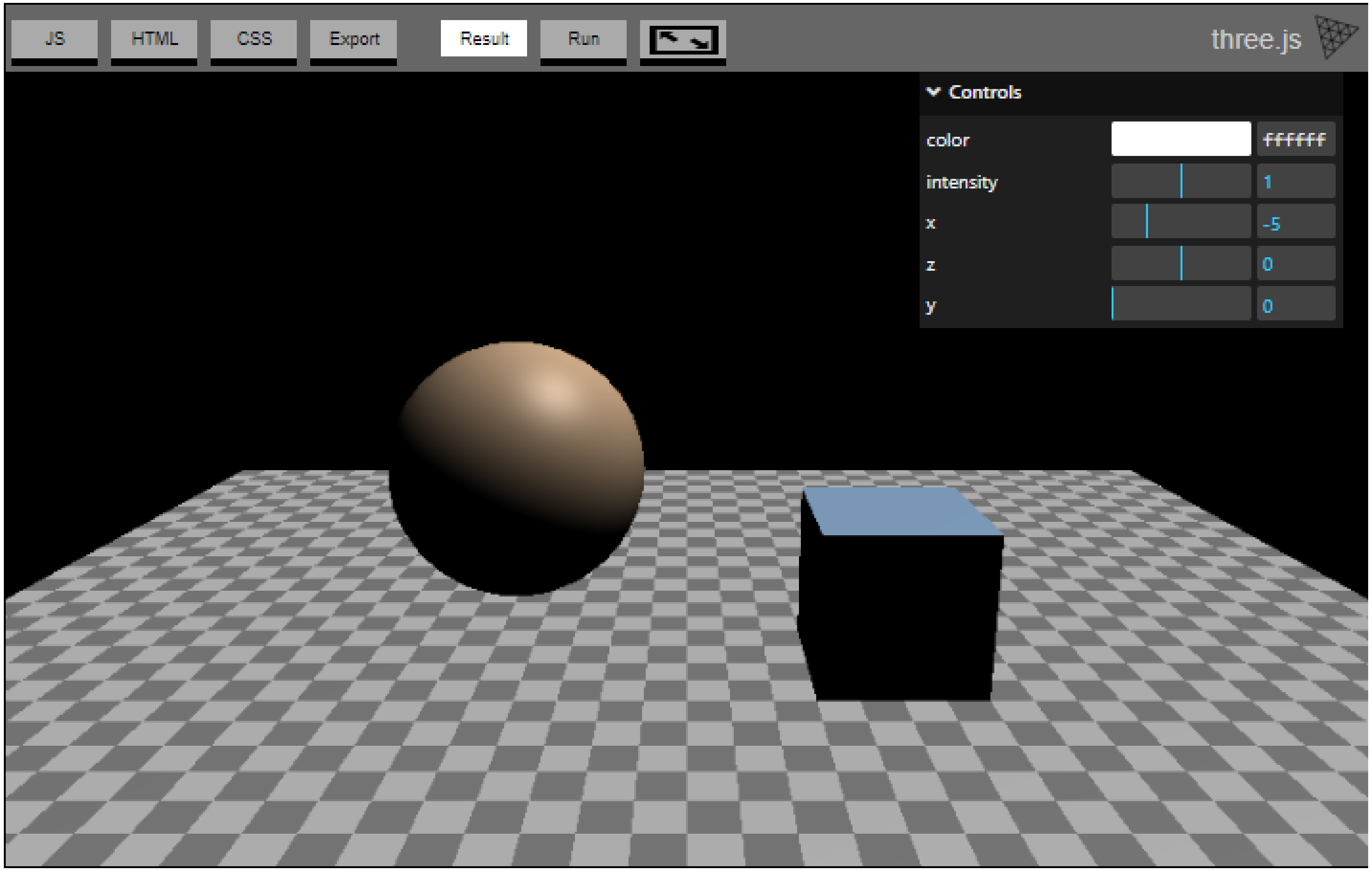
THREE.JS
LIGHT SOURCES & SHADING

Three.js – AmbientLight



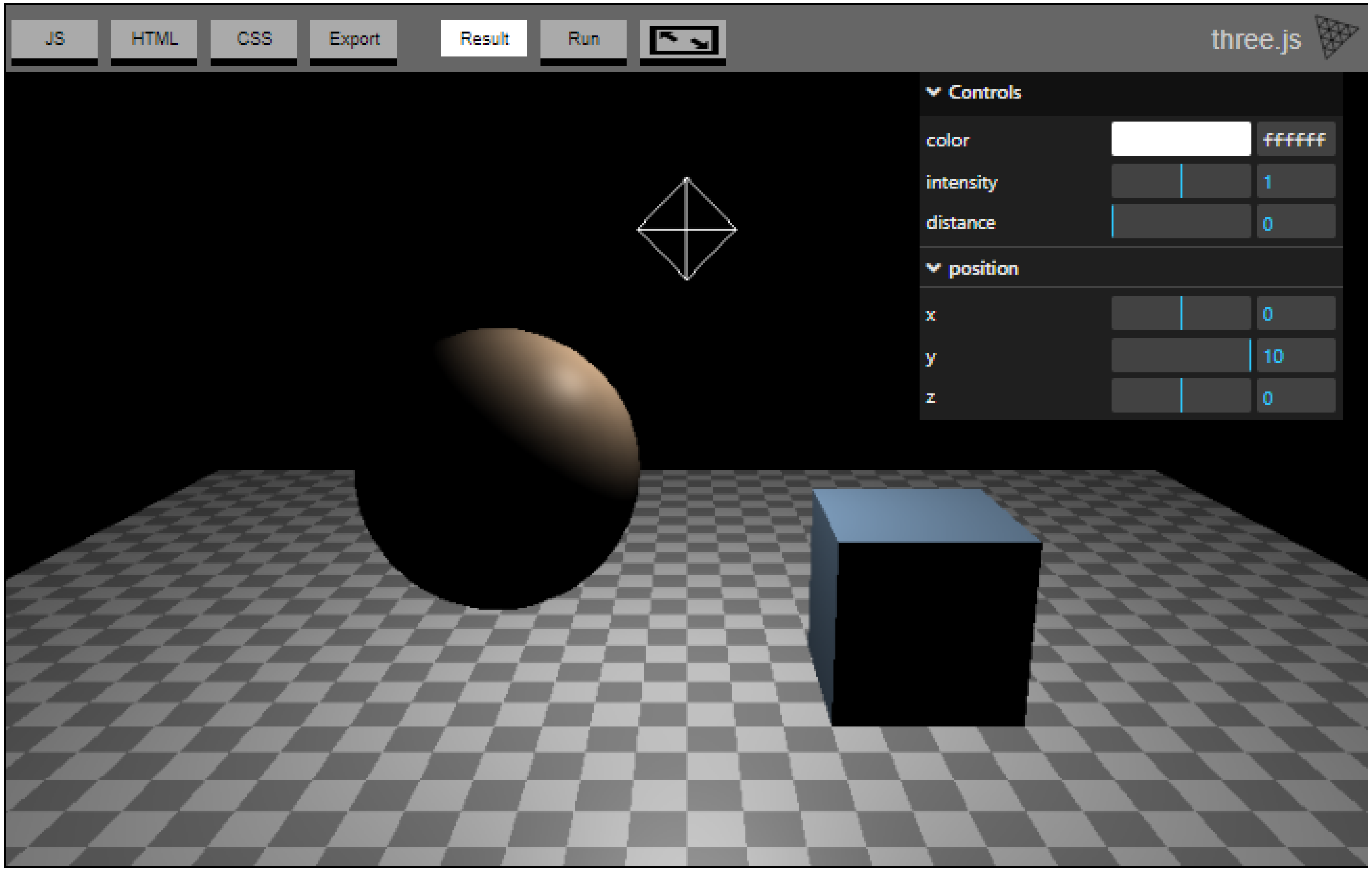
https://threejs.org/manual/examples/lights-ambient.html

Three.js – DirectionalLight



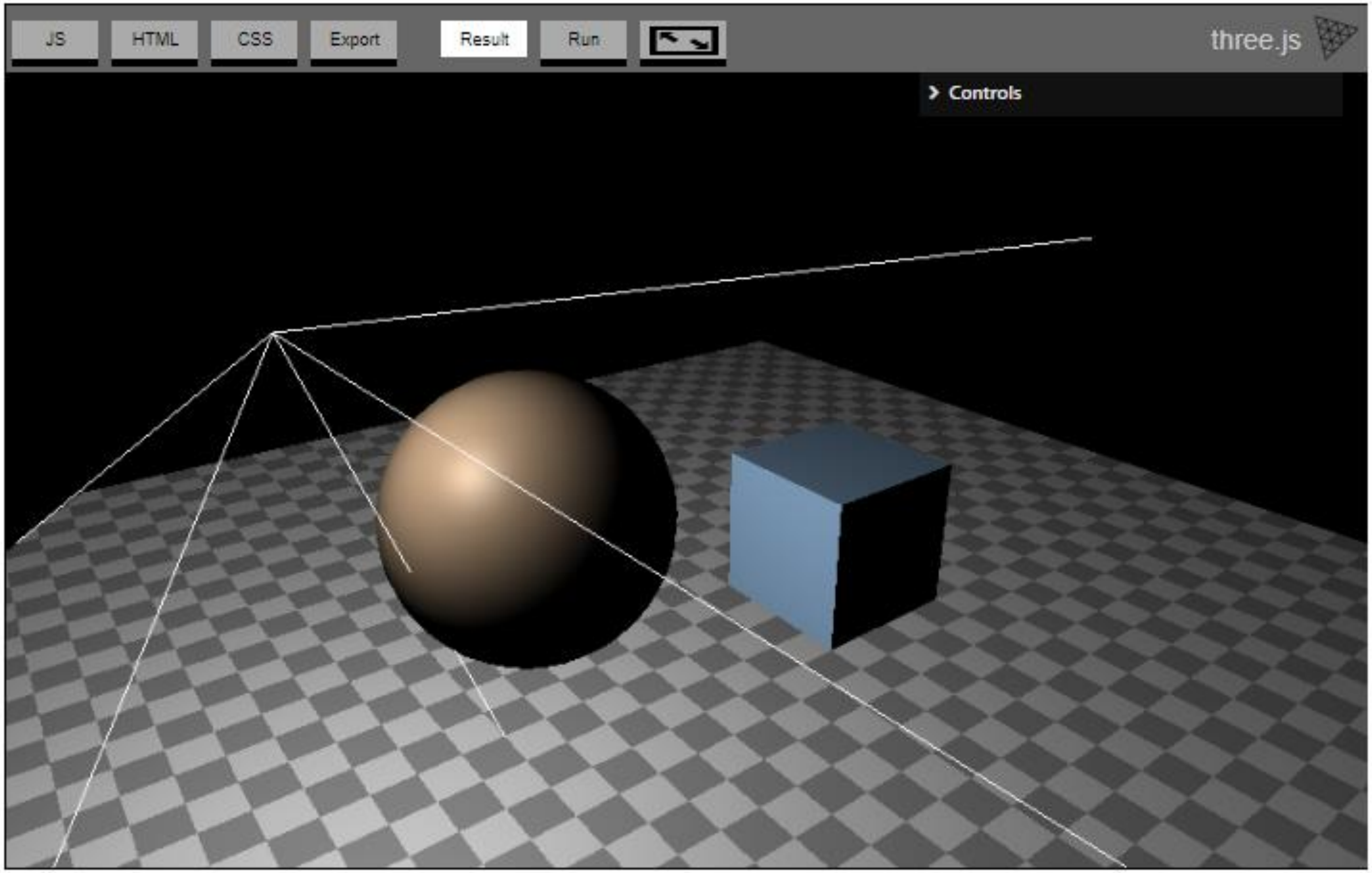
https://threejs.org/manual/examples/lights-directional.html

Three.js – PointLight



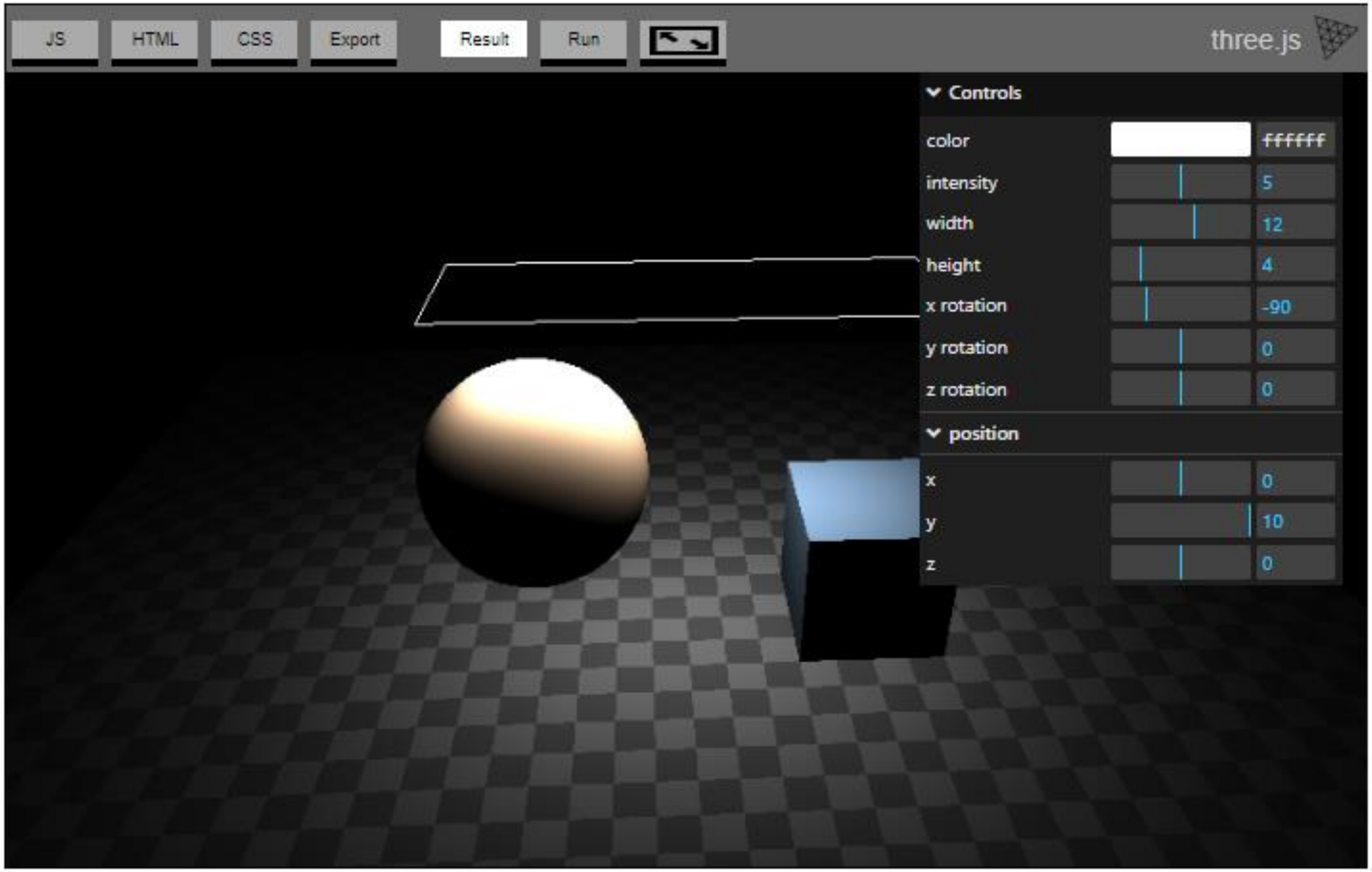
https://threejs.org/manual/examples/lights-point.html

Three.js – SpotLight



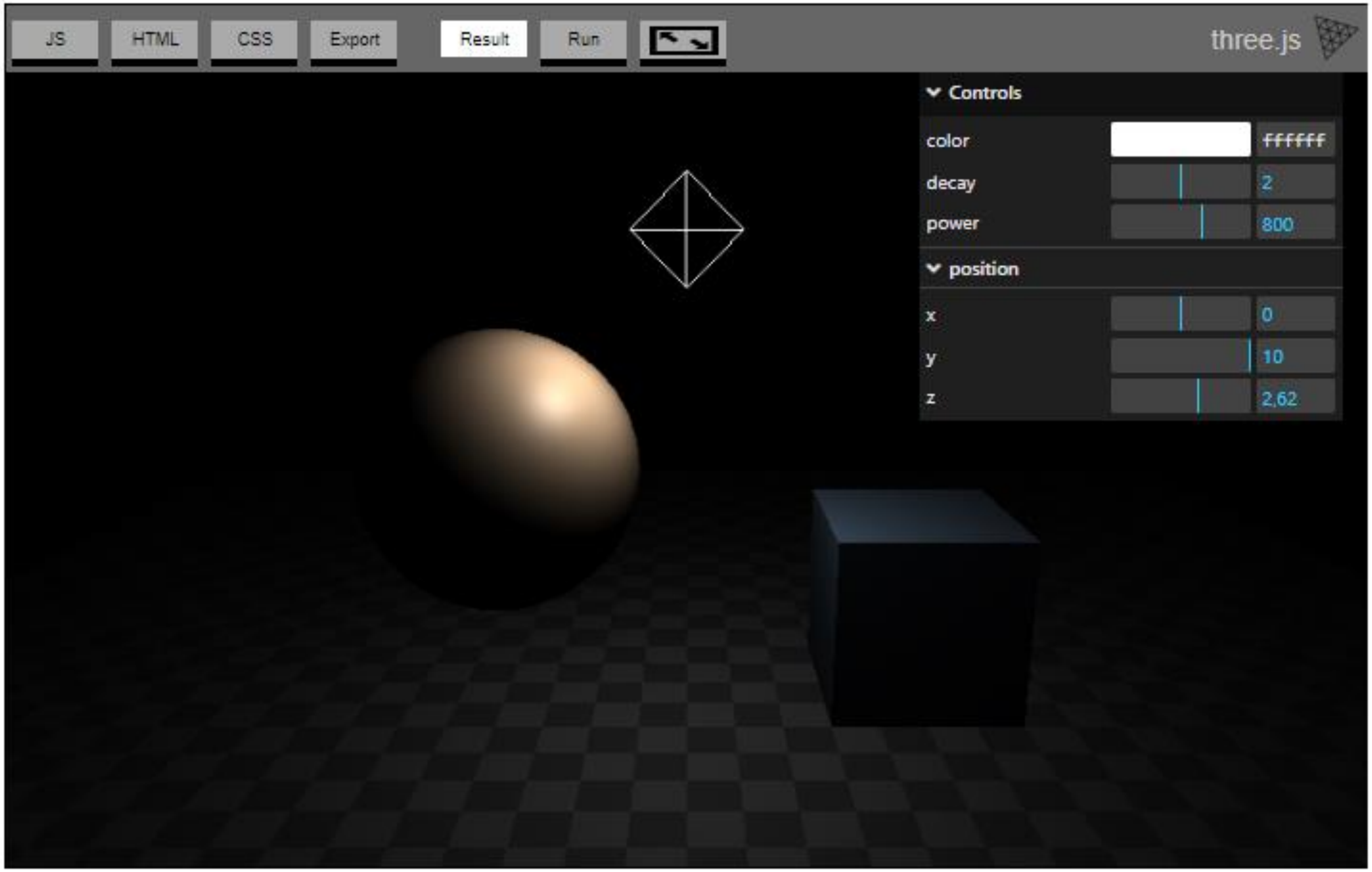
https://threejs.org/manual/examples/lights-spot-w-helper.html

Three.js – RectAreaLight



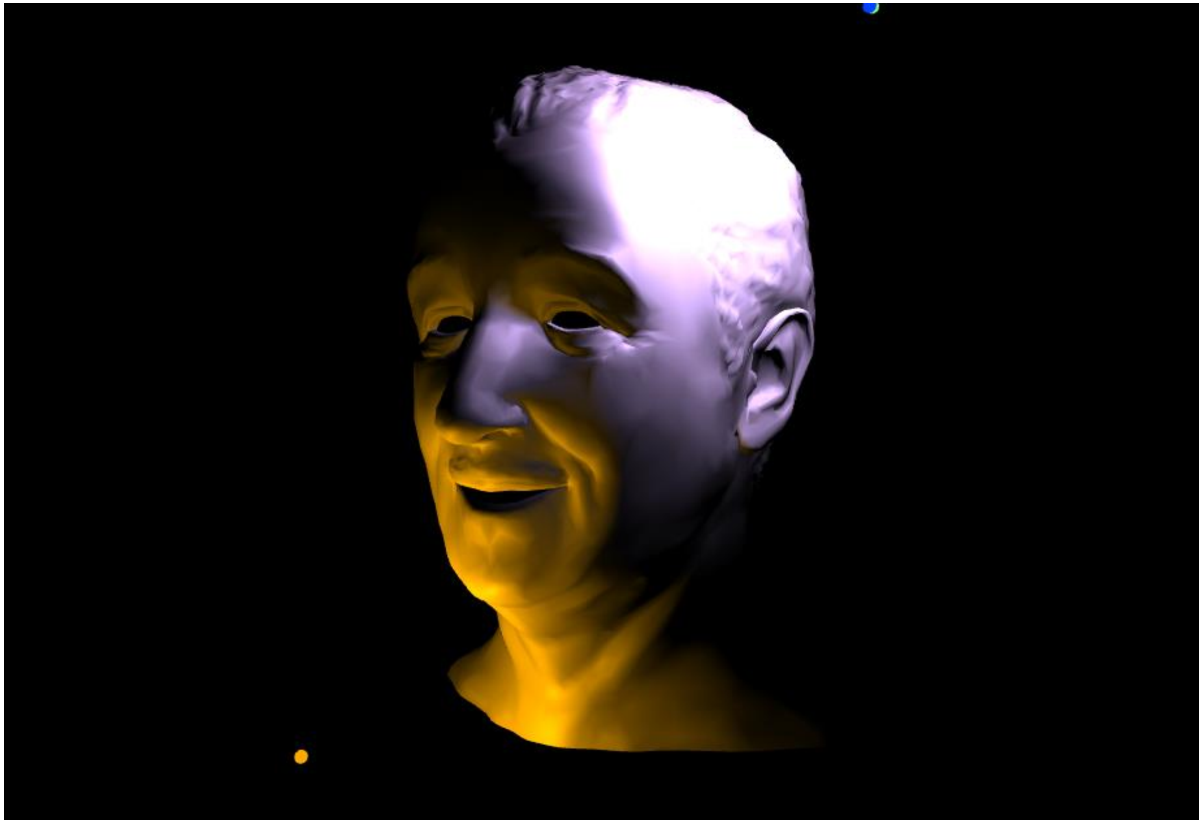
https://threejs.org/manual/examples/lights-rectarea.html

Three.js – Fade out with distance



https://threejs.org/manual/examples/lights-point-physically-correct.html

Three.js – Point light sources example

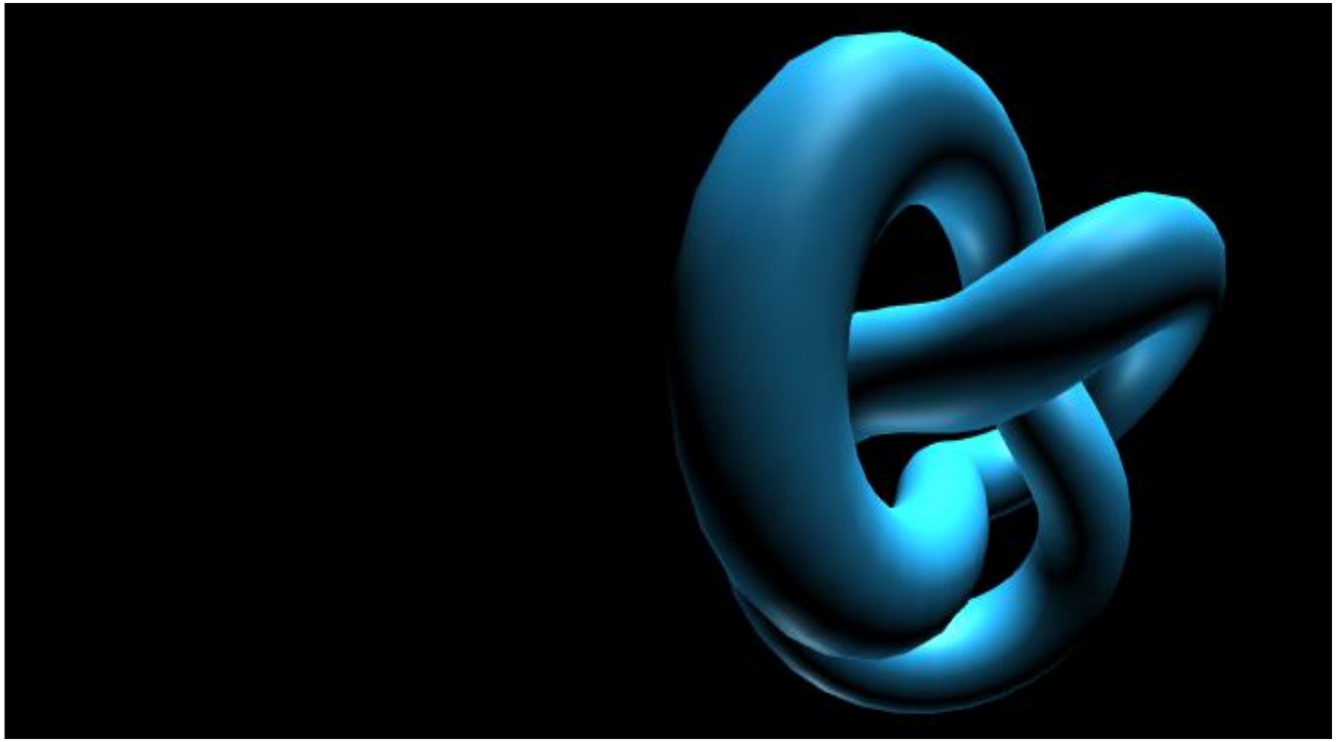


https://threejs.org/examples/#webgl_lights_pointlights

Three.js – Materials

Materials

LineBasicMaterial
LineDashedMaterial
Material
MeshBasicMaterial
MeshDepthMaterial
MeshLambertMaterial
MeshNormalMaterial
MeshPhongMaterial
MeshPhysicalMaterial
MeshStandardMaterial
MeshToonMaterial
PointsMaterial
RawShaderMaterial
ShaderMaterial
ShadowMaterial
SpriteMaterial

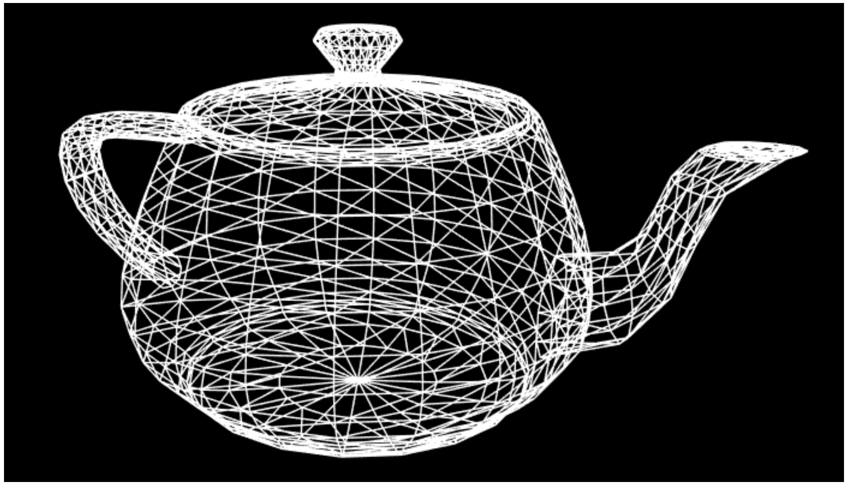


https://threejs.org/docs/index.html#api/en/materials/MeshPhongMaterial

Three.js – The Teapot

https://threejs.org/examples/#webgl_geometry_teapot

Three.js – Shading Examples




https://threejs.org/examples/#webgl_geometry_teapot

ACKNOWLEDGMENTS

Acknowledgments

- Some ideas and figures have been taken from slides of other CG courses.
- In particular, from the slides made available by Beatriz Sousa Santos, Ed Angel and Andy van Dam.
- Thanks!