Lighting

CSU44052 Computer Graphics

Binh-Son Hua

NISSAN
xanavi
BRIDGESTONE
POTENZA

Ballet
Swan Lake
I ♥ MOMMY

Introduction

- Realistic image synthesis
  - Photorealism vs. physically-based realism
  - Film and visual effects, architecture, ergonomic design of buildings and offices, computer games, lighting engineering, predictive simulations, flight and car simulators, advertising
- Non-photorealistic rendering
  - Suited for an artistic, technical, or educational approach
  - Pen-and-ink drawings, cartoon-style drawings, technical illustrations, watercolour painting etc.

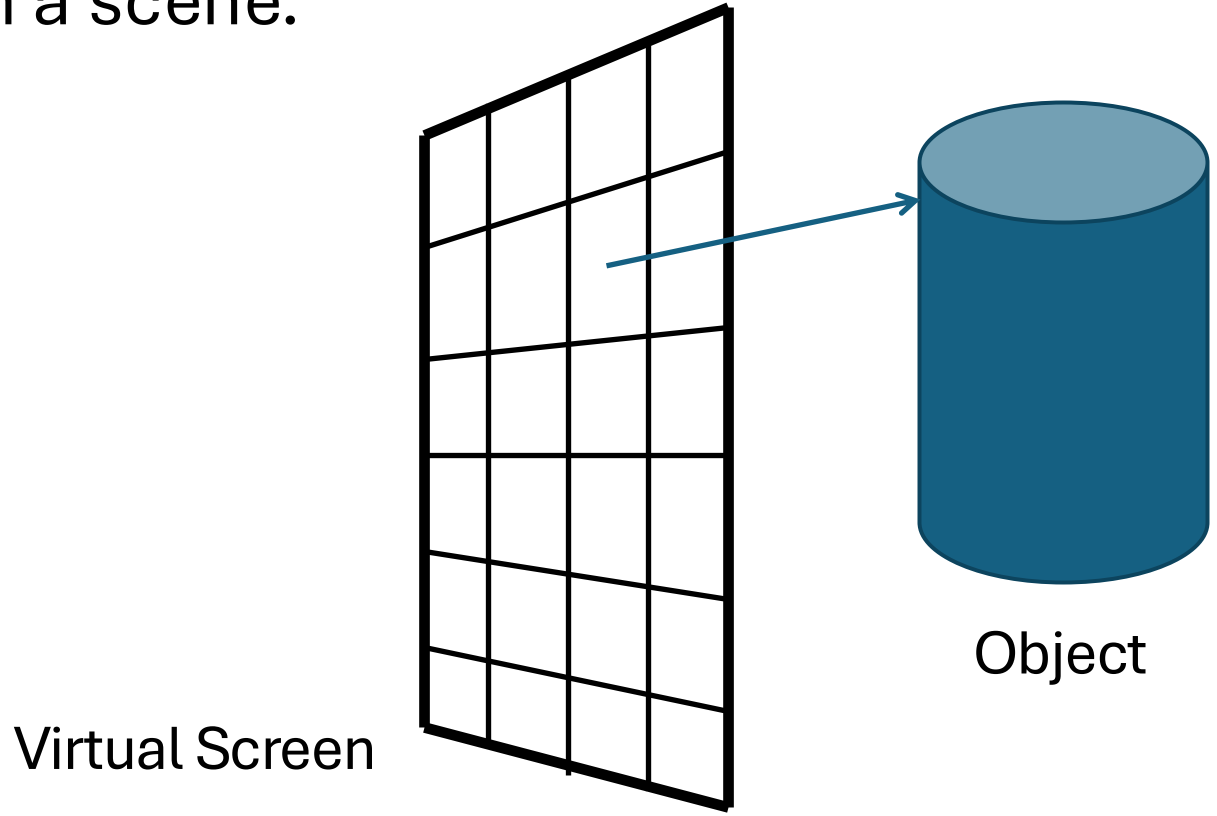
RECARO
ALPINE

Overview

- Rendering algorithms (local, global, view dependent, view independent)
- Light sources
- Lambertian illumination model

Rendering

- Rendering is fundamentally concerned with determining the most appropriate colour (i.e. RGB tuple) to assign to a pixel associated with an object in a scene.



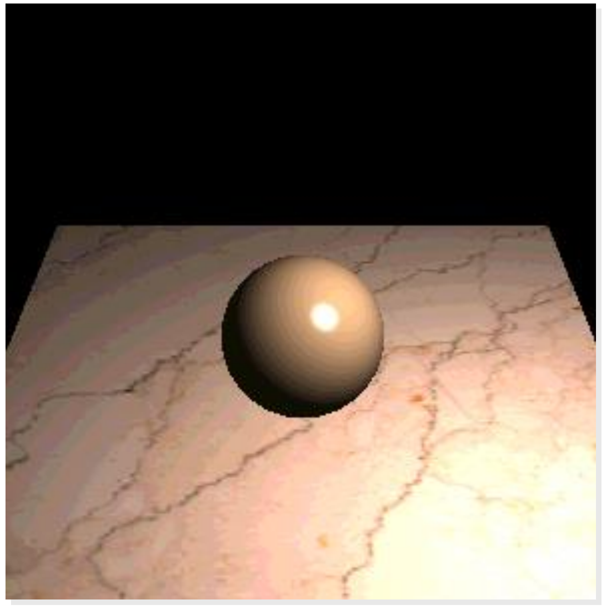
Rendering

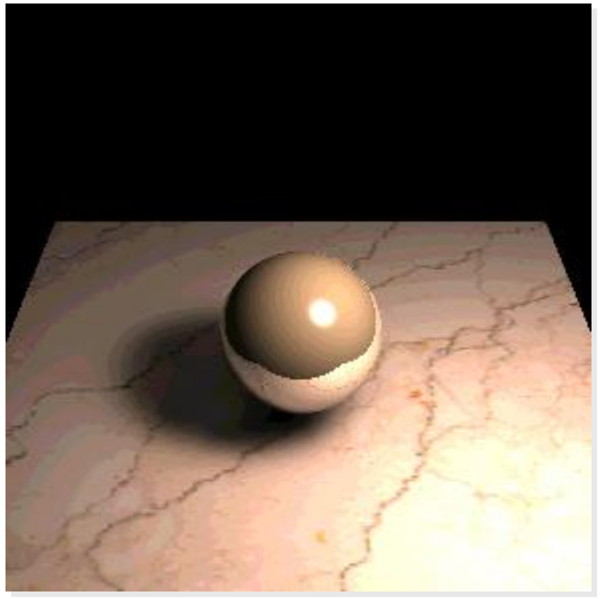
- The colour of an object at a point depends on:
  - geometry of the object at that point (*normal* direction)
  - position, geometry and colour of the light sources (*luminaires*)
  - position and visual response of the viewer
  - *surface reflectance properties* of the object at that point
  - scattering by any *participating media* (e.g. smoke, rising hot air)

Rendering Algorithms

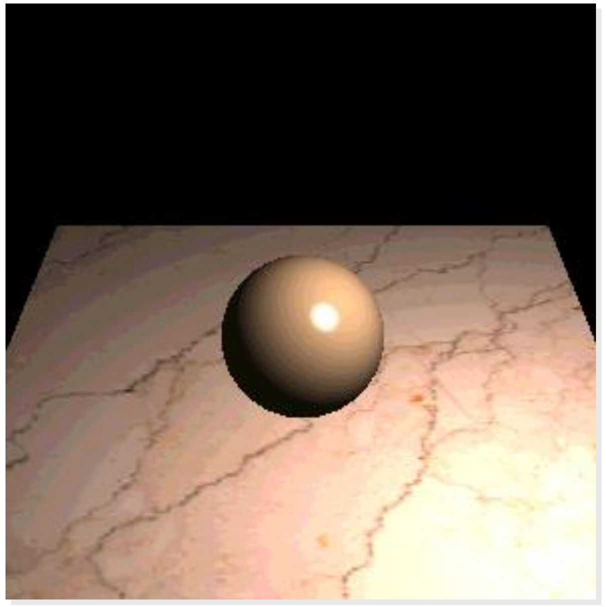
- Rendering algorithms differ in the assumptions made regarding lighting and reflectance in the scene and in the solution space:
  - **local illumination** algorithms: consider lighting only from the light sources and ignore the effects of other objects in the scene (i.e. reflection off other objects or shadowing)
  - **global illumination** algorithms: account for all modes of *light transport*
  - **view dependent** solutions: determine an image by solving the illumination that arrives through the viewport only.
  - **view independent** solutions: determine the lighting distribution in an entire scene regardless of viewing position. Views are then taken after lighting simulation by sampling the full solution to determine the view through the viewport.

Which Lighting Model?



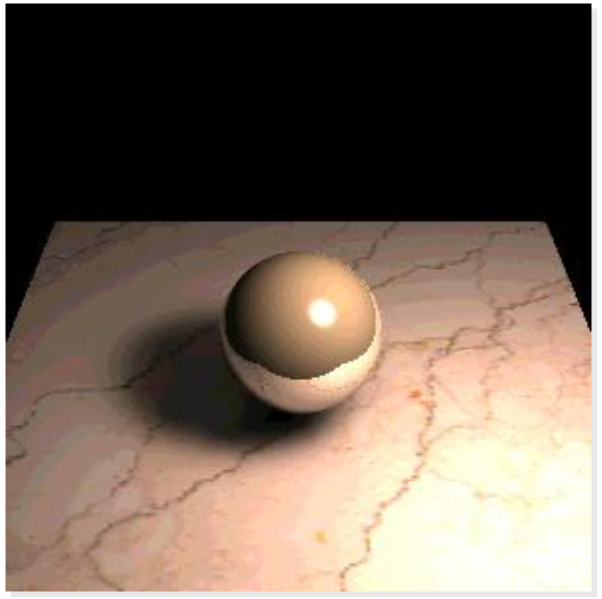


Local vs. Global Illumination



Local

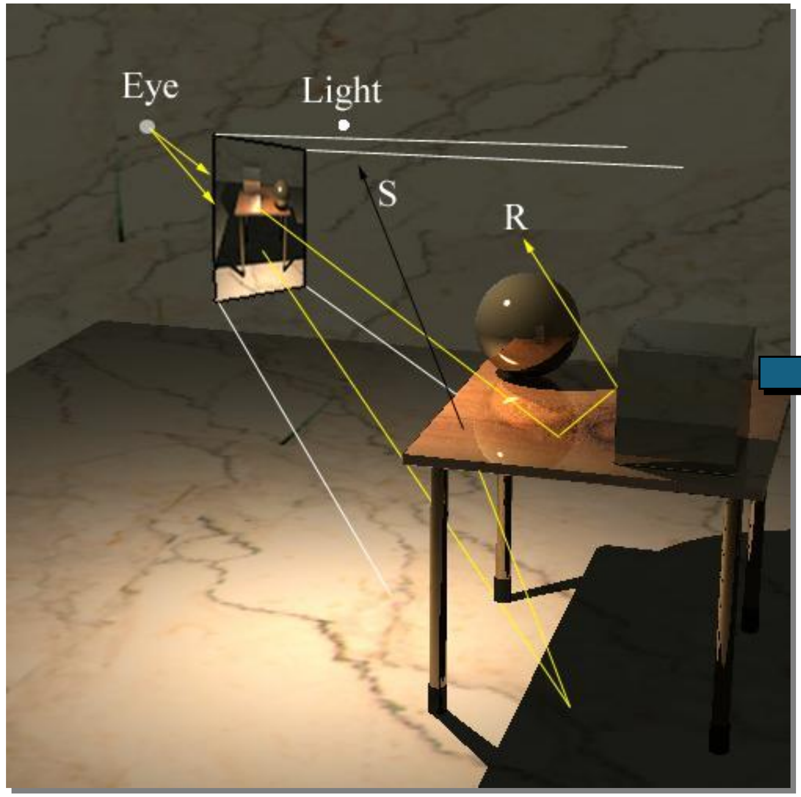
Illumination depends on local object & light sources only



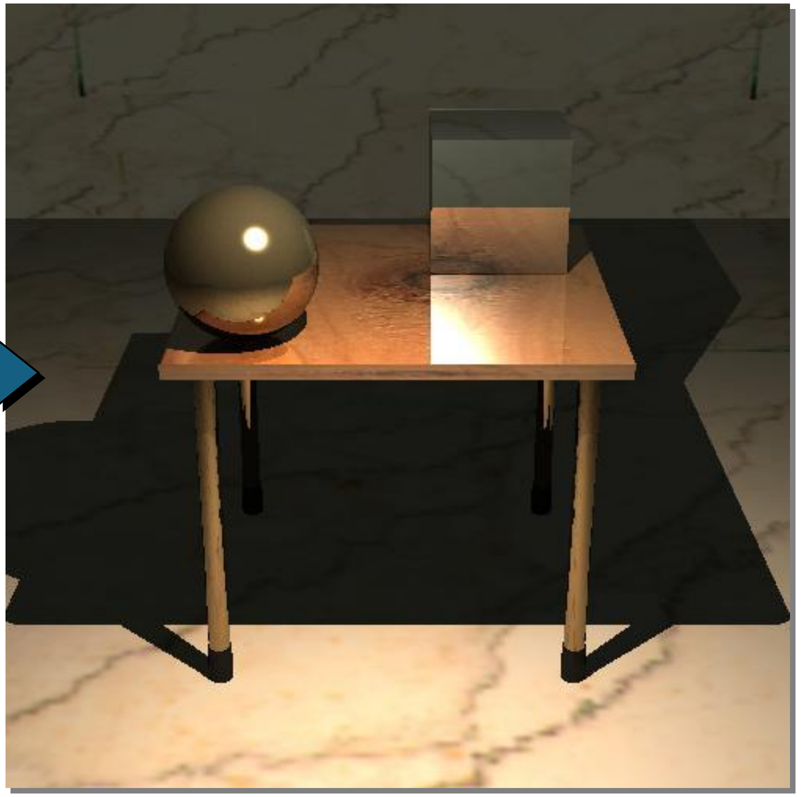
Global

Illumination at a point can depend on any other point in the scene

View Dependent Solution (Ray Traced)



Scene Geometry

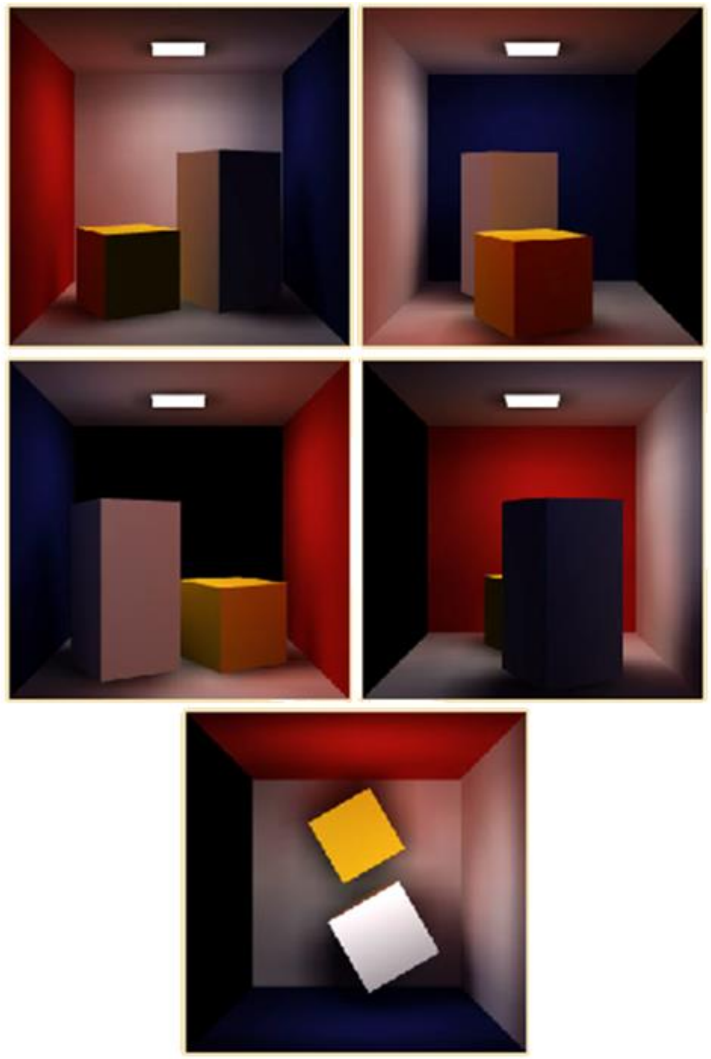


Solution determined only for directions through pixels in the viewport

View Dependent

- Advantages
  - Only the relevant parts of the scene are taken into consideration
  - Can work with any kind of geometry
  - Can produce very high-quality results
  - In some methods, view-dependent portions of the solution can be cached as well (glossy reflections, refractions etc).
  - Require less memory than a view-independent solution.
- Disadvantages
  - Requires updating for different camera positions; still, in some implementations portions of the solution may be re-used.

View Independent (Radiosity)



A single solution for the light distribution in the entire scene is determined in advance.

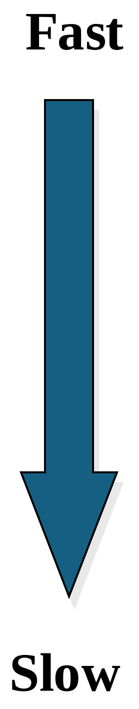
Then we can take different snapshots of the solution from different viewpoints by sampling the complete solution for specific positions and directions.

View Independent

- Advantages
  - Solution needs to be computed only <u>once</u>.
- Disadvantages
  - All of the scene geometry must be considered, even though some of it may never be visible.
  - The type of geometry in the scene is usually restricted to trianglular or quadrangular meshes (no procedural or infinite geometry allowed).
  - Detailed solutions require lots of memory.
  - Only the diffuse portion of the solution can be cached; view-dependent portions (glossy reflections) must still be computed.

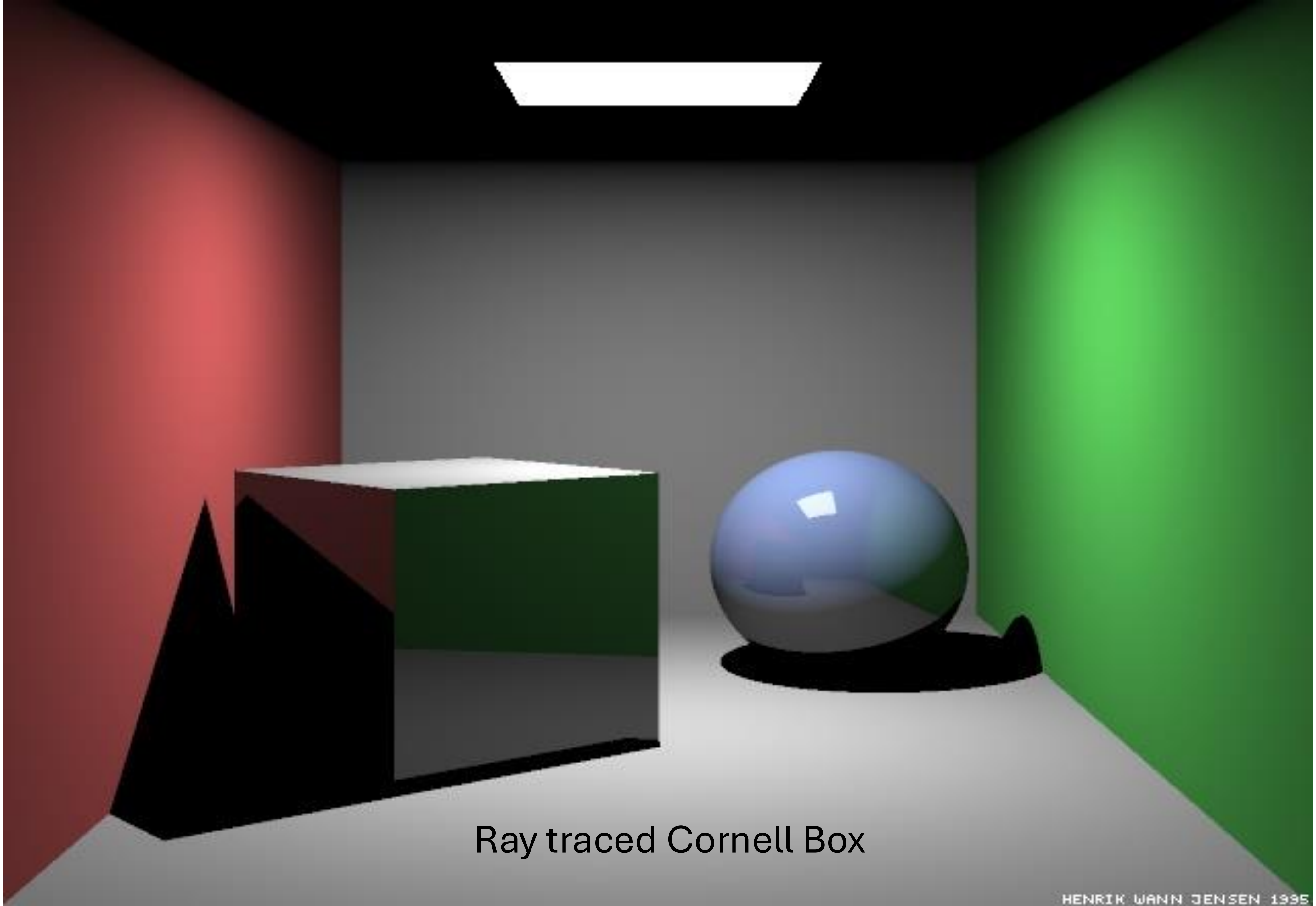
Global Illumination Algorithms

- Different algorithms solve the illumination problem with making different assumptions to vary the speed/accuracy tradeoff.
  - ***Z-Buffer Algorithms***:
    - can compute approximate shadows and reflection from planar surfaces
  - ***Ray Tracing Algorithms***:
    - determine exact shadows, reflections and refractive effects (transparency) assuming point light sources (no volume).
  - ***Radiosity Algorithms***:
    - computes approximate solutions assuming no *shiny* surfaces, but light sources can be arbitrarily large and all surfaces polygonal.
  - ***Path Tracing Algorithms***:
    - employing an expensive Monte-Carlo solution to handle arbitrary geometries, reflectance and lighting.

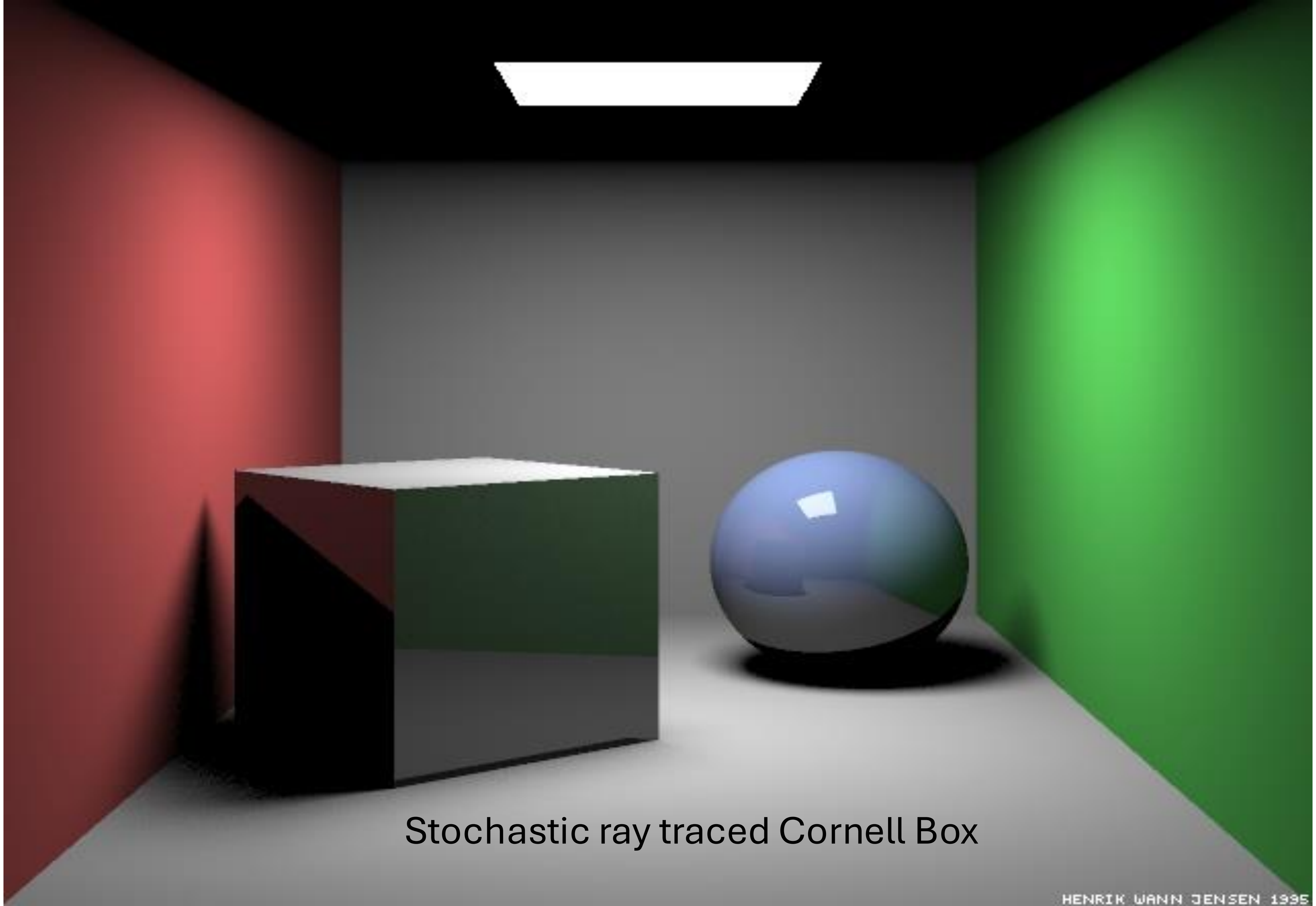


Z-buffer methods only

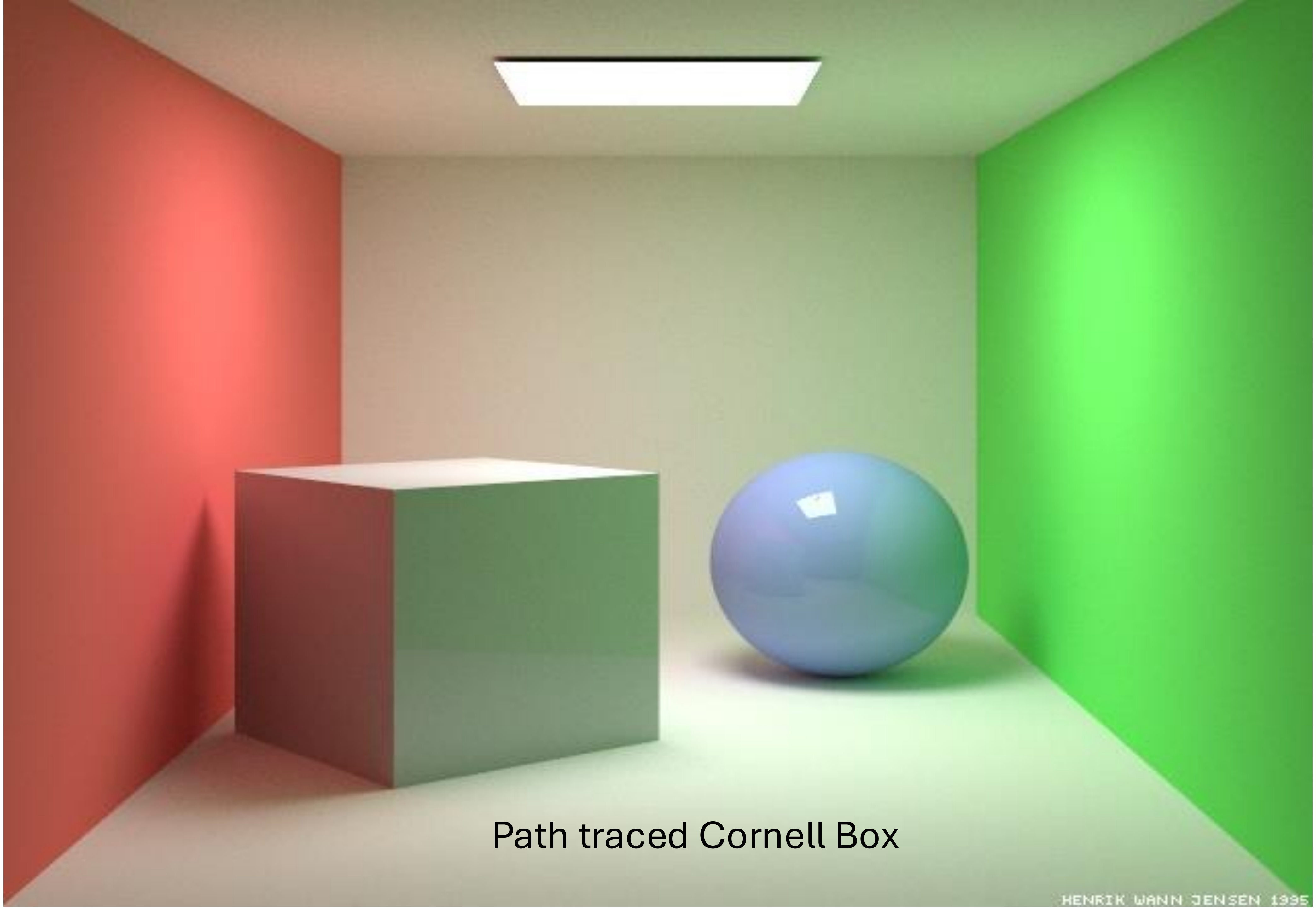
Crystal glass rendering using path tracing



Ray traced Cornell Box



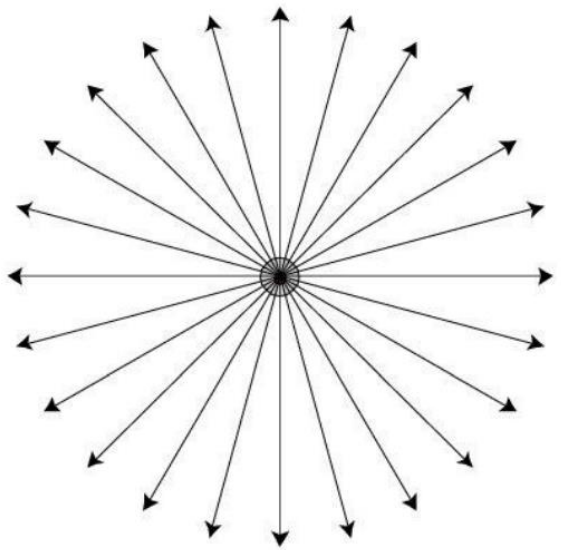
Stochastic ray traced Cornell Box



Path traced Cornell Box

Light Sources

- An ***isotropic point light source*** is defined by:
    - Position and Colour



Isotropic $\Rightarrow$ radiates energy equally in all directions

Radiometric Units

- The total *energy* (Joules) leaving a surface per unit time:

  $\Rightarrow$ *power* or ***Flux*** (Watts) = $\mathbf{\Phi}$
- The flux leaving a surface can change with position on the surface (i.e. some points are brighter), so flux per unit area:

  $\Rightarrow$ ***Radiosity*** (Watts/m^2) = **B**
- Flux arriving per unit area:

  $\Rightarrow$ ***Irradiance*** (Watts/m^2) = **E**
- The point on the surface might emit different energy in different directions so radiosity per direction:

  $\Rightarrow$ ***Radiance*** (Watts/m^2/sr) = **L**
- Radiance is usually of most interest in computer graphics
  - i.e. flux per area reflected towards the viewer

Light Sources

- Normally we wish to associate radiance with a light source, but the definition of radiance assumes an area over which energy is emitted/distributed
- Point sources have no area!
- Use radiant intensity instead (units Watts/sr).
- A point source radiating energy *in all directions equally* has a radiant intensity of:

$$I = \frac{\Phi}{\omega} = \frac{\overset{\text{Total energy}}{\Phi}}{4\pi r^2}$$

Total energy

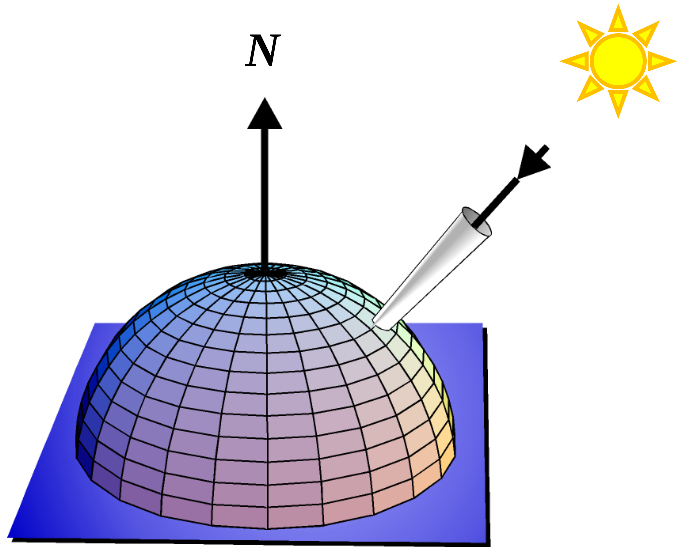
Solid angle
e.g., for the sun this would be a sphere

$4\pi r^2$

= sphere surface area

Diffuse Lighting

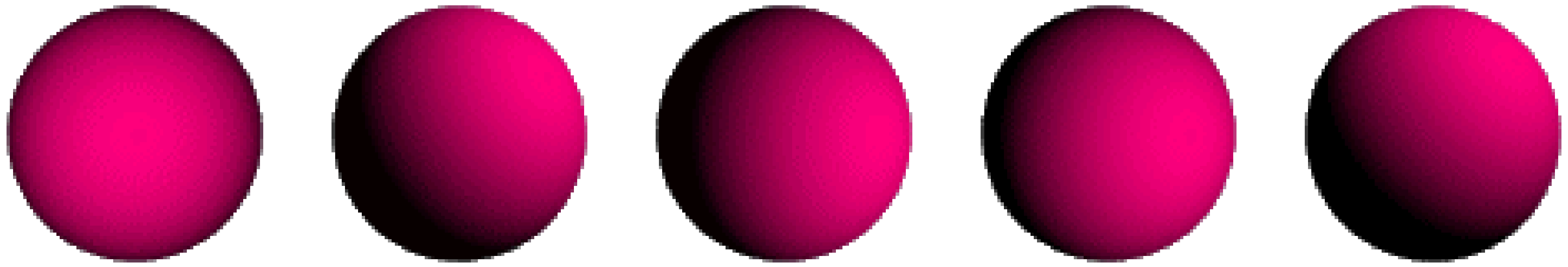
- The illumination that a surface receives from a light source and reflects **equally** in all directions
- This type of reflection is called Lambertian reflection.
- The brightness of the surface is independent of the observer position (since the light is reflected in all direction equally)



Lambertian Illumination Model

- To shade a diffuse surface we need to know:
  - normal to the surface at the point to be shaded
  - diffuse reflectance of the surface
  - positions and powers of the light source in the scene

- Below are several examples of a spherical diffuse reflector with varying lighting angles.



Lambertian Illumination Model

The contribution from a single source is given by:

$$\underbrace{L(x,\cdot)}_{\text{Reflected radiance}} = \underbrace{\frac{\rho_d}{\pi}}_{\text{BRDF}} \underbrace{\cos\theta}_{\text{cosine rule}} \underbrace{\frac{\Phi}{4\pi r^2}}_{\text{Light source irradiance}}$$

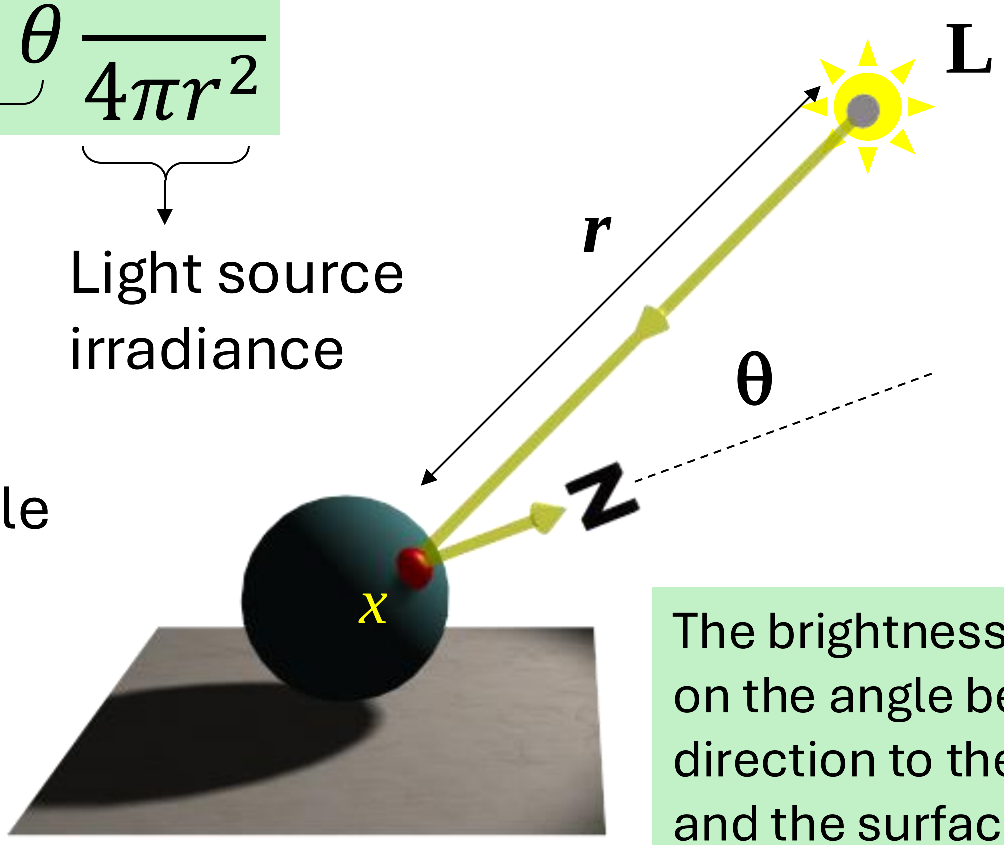
Reflected radiance

BRDF

Light source irradiance

cosine rule

outgoing direction
(can be any)



The brightness depends only on the angle between the direction to the light source and the surface normal

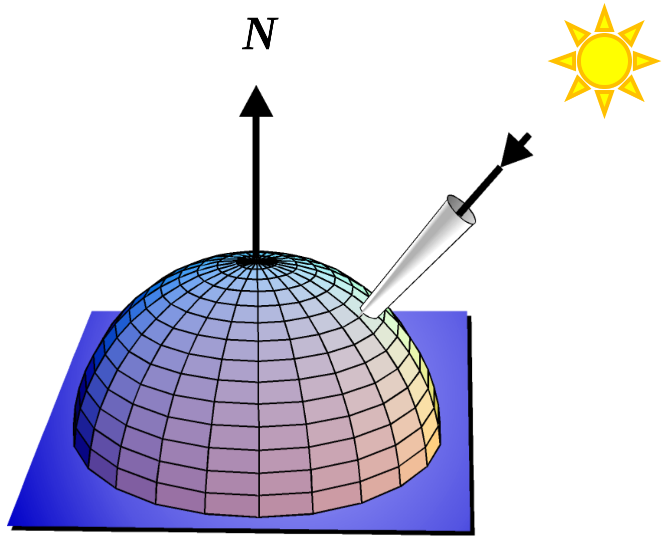
Reflectivity

- The reflectivity varies from zero for a completely absorbing ("black") surface, to one for a completely reflecting ("white") surface.

- There are no Lambertian surfaces in nature, but matte paper is good approximation
  - except at grazing angles and near 90 degrees, where the surface begins to look "shiny“.

Reflectivity

- Diffuse surfaces reflect light in all directions equally:
  - BRDF is a constant with respect to reflected direction
  - surface may be characterized by a *reflectance* ρ_d rather than a BRDF
  - the reflectance ρ_d gives the ratio of the total reflected power Φ_r to the total incident power Φ_i



$$f_r(x) = \frac{\rho_d(x)}{\pi}$$

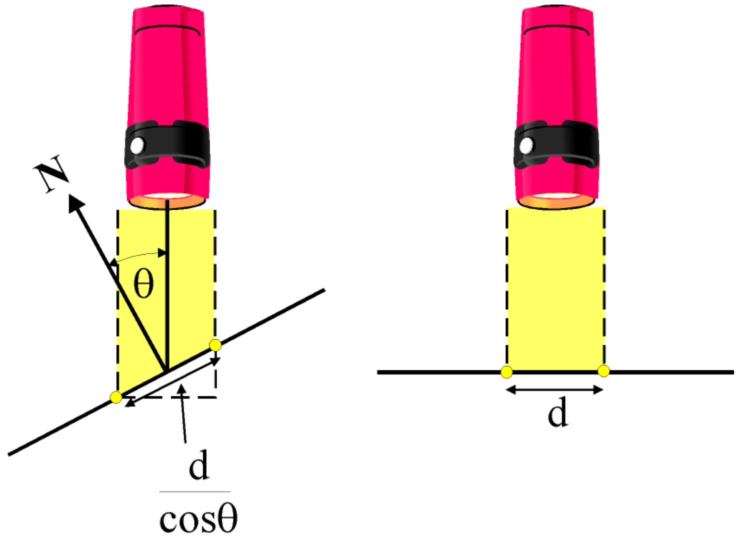
← With unit reflectivity

$$\rho_d(x) = \frac{\Phi_i}{\Phi_r}$$

The Cosine Rule

- A surface which is oriented perpendicular to a light source will receive more energy per unit area (and thus appear brighter) than a surface oriented at an angle to the light source.
- The irradiance *E* is proportional to $\frac{1}{area}$
- As the area increases, the irradiance decreases therefore:

θ	*Cos(θ)*
0	1.00
10	0.98
30	0.87
50	0.64
70	0.34
90	0.00
110	-0.34
130	-0.64



$$E = \frac{\cos\theta\,\Phi}{4\pi r^2}$$

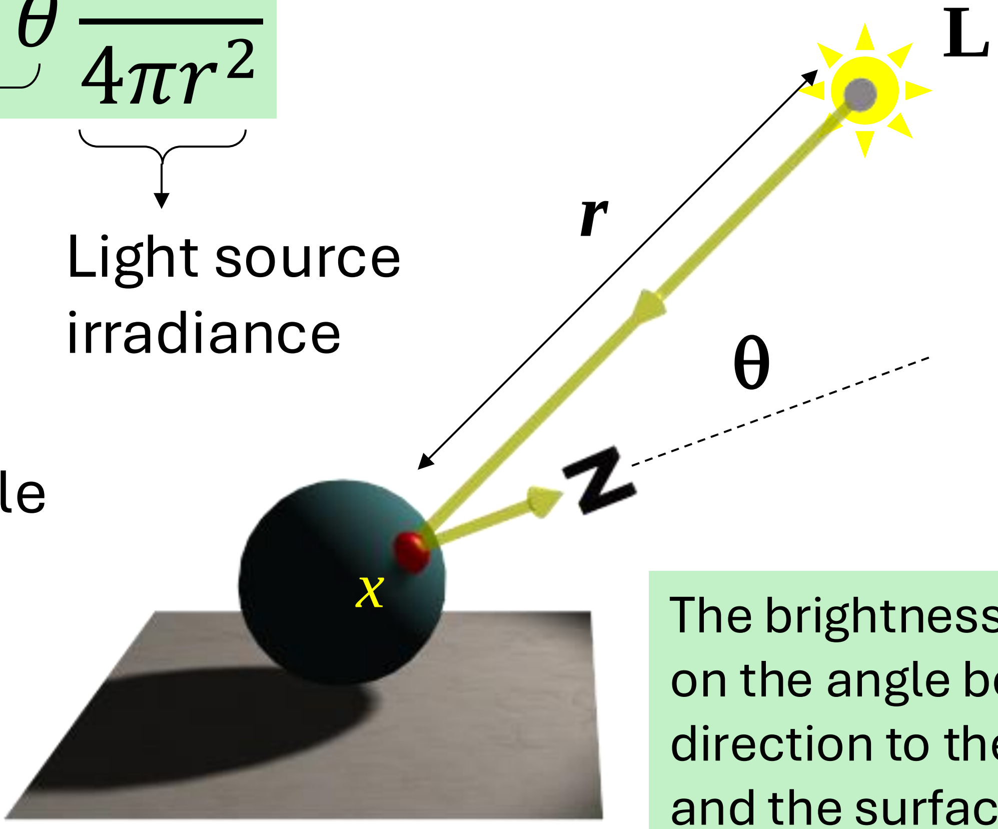
As θ increases, the irradiance and thus the brightness of a surface decreases by cos θ

Lambertian Illumination Model

The contribution from a single source is given by:

$$\underbrace{L(x,\cdot)}_{\text{Reflected radiance}} = \underbrace{\frac{\rho_d}{\pi}}_{\text{BRDF}} \underbrace{\cos\theta}_{\text{cosine rule}} \underbrace{\frac{\Phi}{4\pi r^2}}_{\text{Light source irradiance}}$$

outgoing direction (can be any)



The brightness depends only on the angle between the direction to the light source and the surface normal

Shadow Mapping

- Any point seen by the light is illuminated, otherwise in shadow.

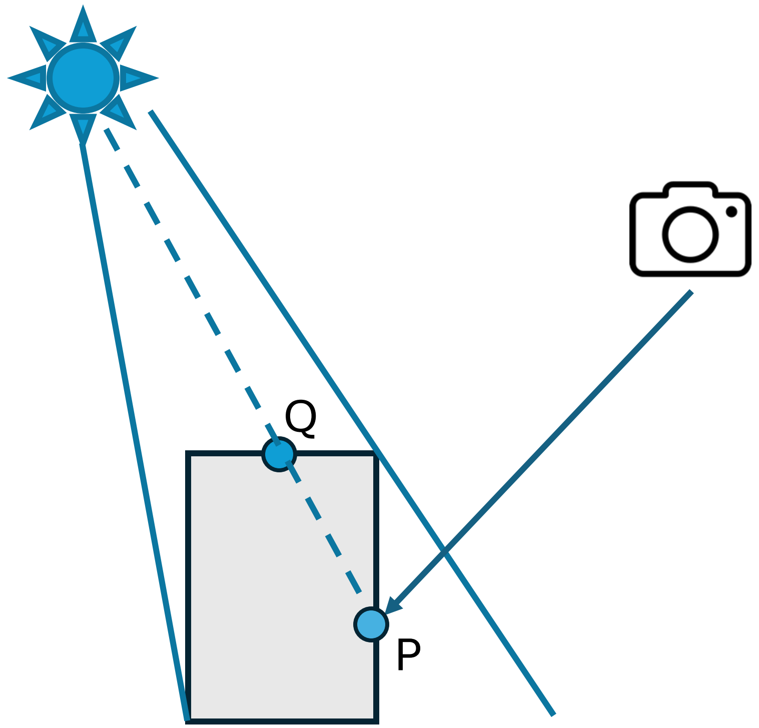


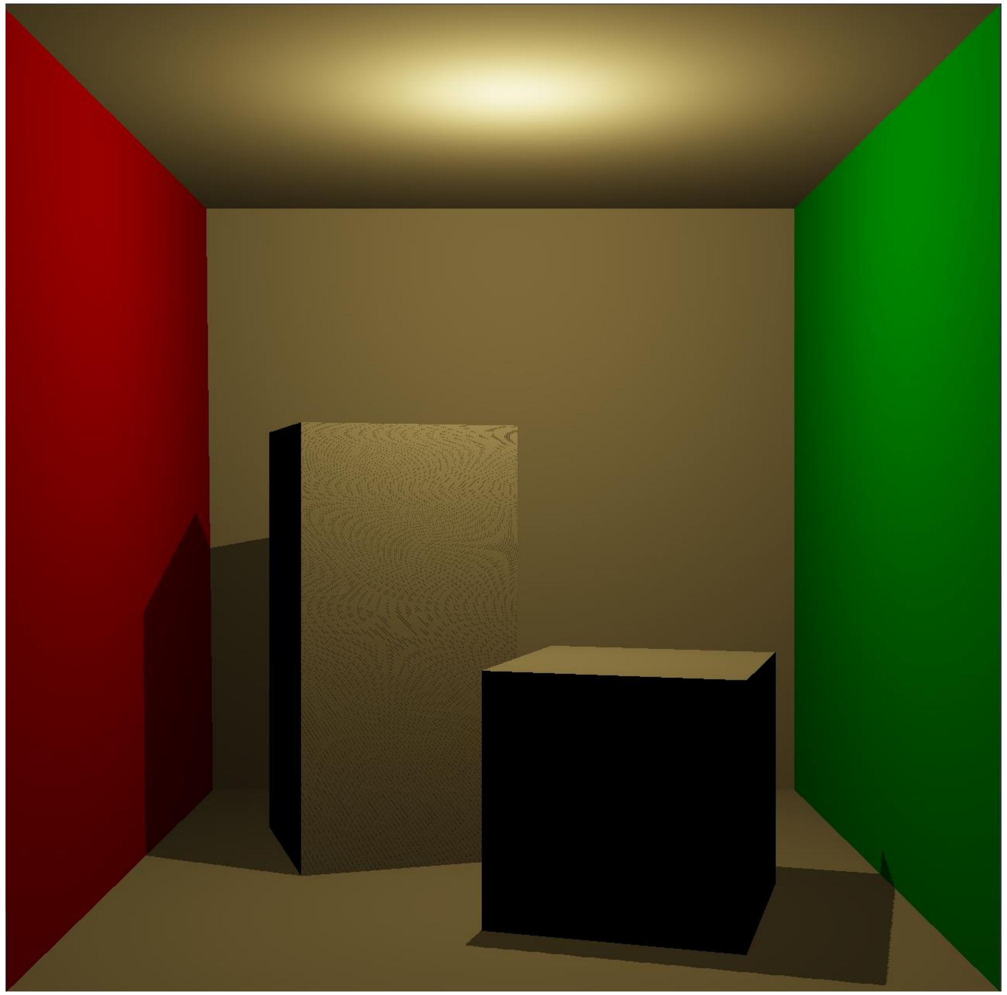
Image by Datta et al., Neural Shadow Mapping, SIGGRAPH

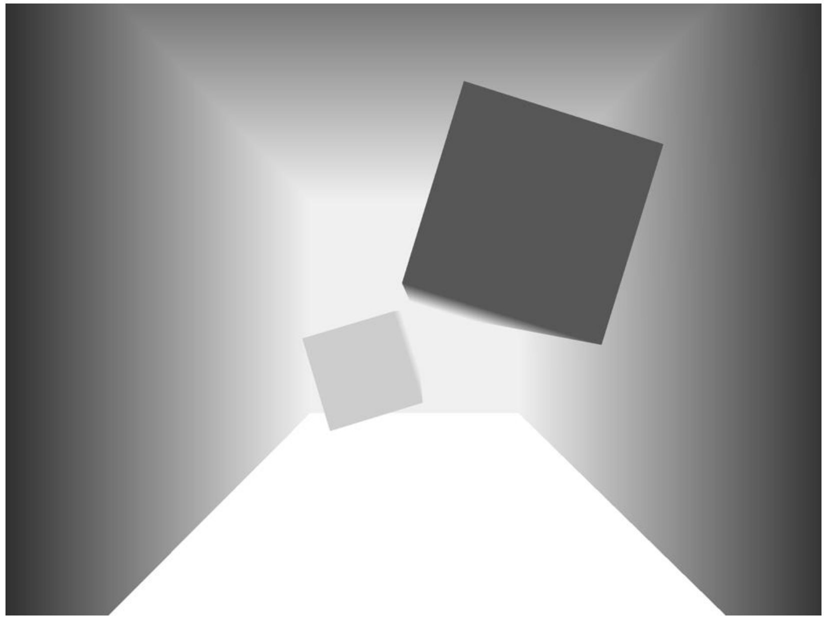
Shadow Mapping

- Step 1: Render the scene from the light’s viewpoint.
  Store the depth map (shadow map).
- Step 2: Render the scene again, this time from the camera’s viewpoint.
  For each fragment, transform its 3D position **p** from the camera space to the light space.
  If **p**’s depth to the light > the depth stored in the shadow map, **p** is occluded, else **p** is lit.

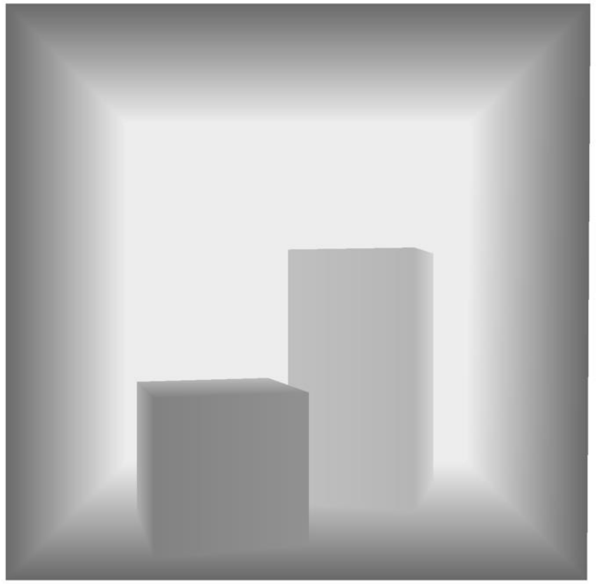
- Shadow mapping works well for directional light and spotlight using normal textures. For point lights, use cube texture maps.

Shadow Mapping





Depth buffer viewed from the light source



Depth buffer viewed from the camera

Further Reading

- Cornell Box: https://www.graphics.cornell.edu/online/box/data.html

- Cascaded shadow maps https://developer.download.nvidia.com/SDK/10.5/opengl/src/cascaded_shadow_maps/doc/cascaded_shadow_maps.pdf