Lecture 11
Computer graphics
Rendering

June 14th, 2021

Meirav Galun

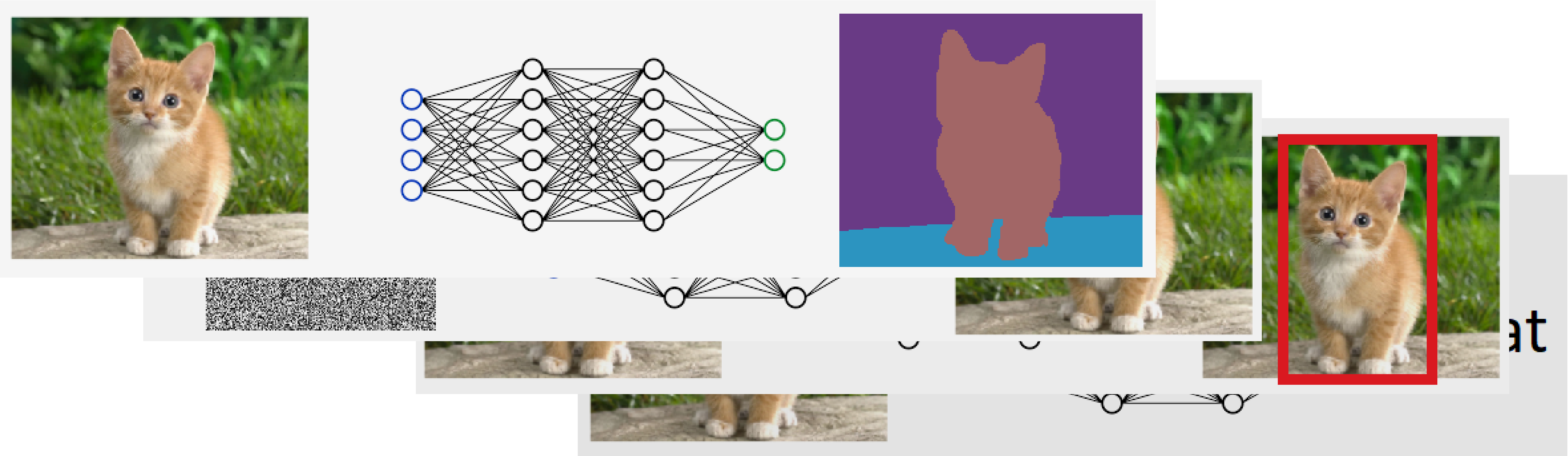
Based on Keenan Crane's course on Computer Graphics, CMU 15-462/662

and the book Computer Graphics: principles and practice by Foley

Computer vision

So far…Computer vision tasks

Processing over 2D images
classification, detection, generation, segmentation...

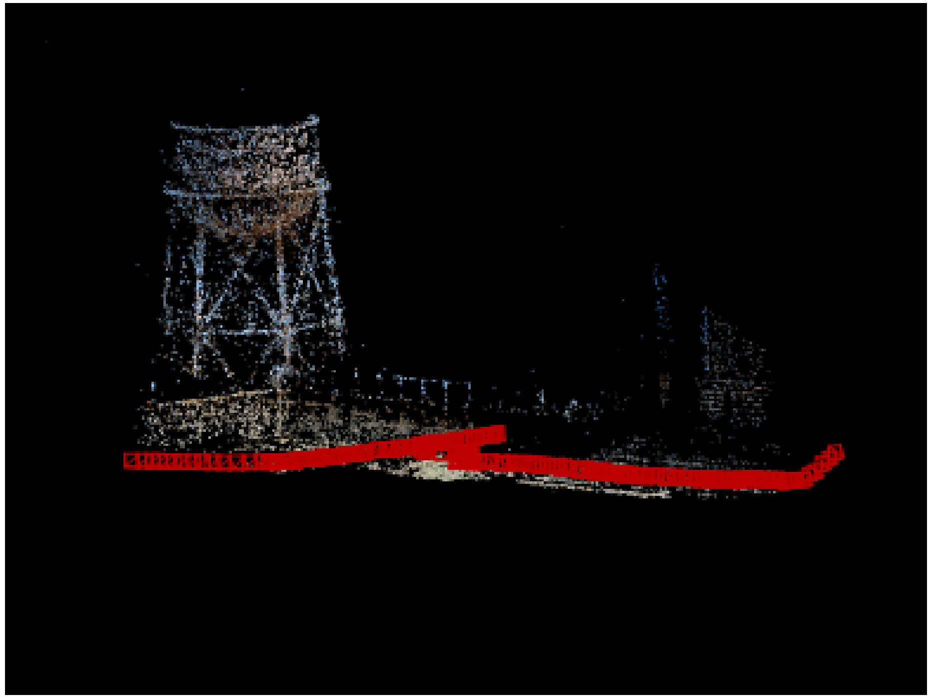


Slide by Amnon Geifman

Geometry in computer vision

Computer vision

Computer graphics



Images
2D

Geometry
3D

Structure from motion problem

Koslowsky* and Moran* et al

Computer graphics

Computer graphics

photorealistic image formation = rendering

Classical rendering

The process of transforming a scene definition including cameras, lights, surface geometry and material into a simulated camera image is known as rendering

All ingredients of physical image formation are modelled in computer graphics

- light sources
- scene geometry
- material properties
- light transport
- optics
- sensor behavior

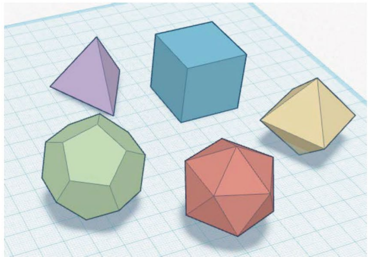
Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

Application

camera

geometry



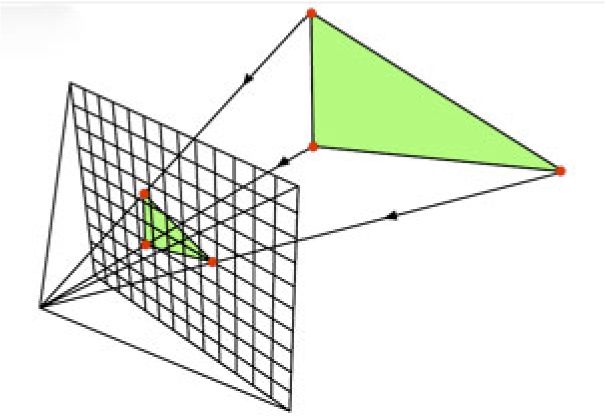
materials

lights

("scene")

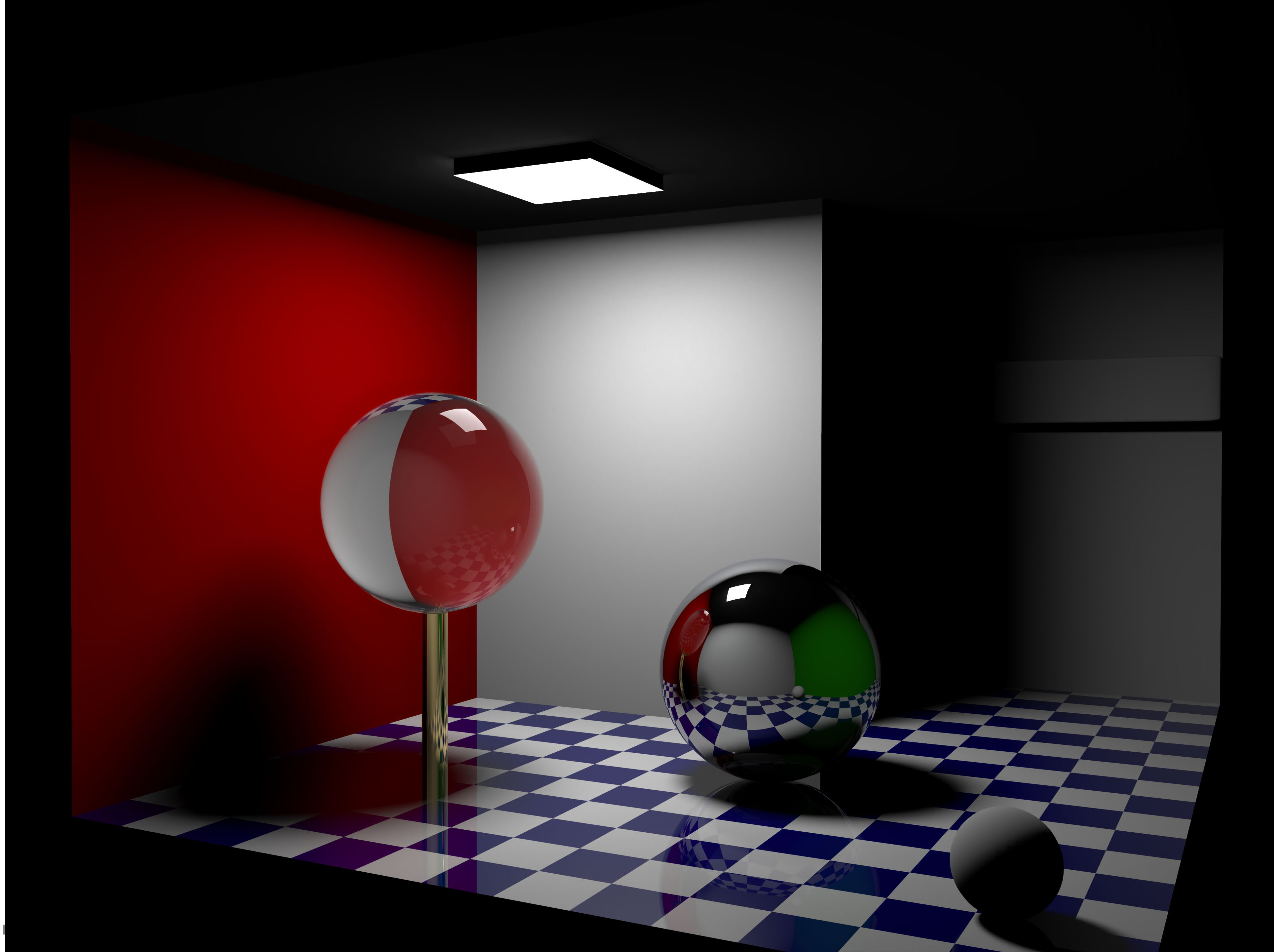
Rasterization

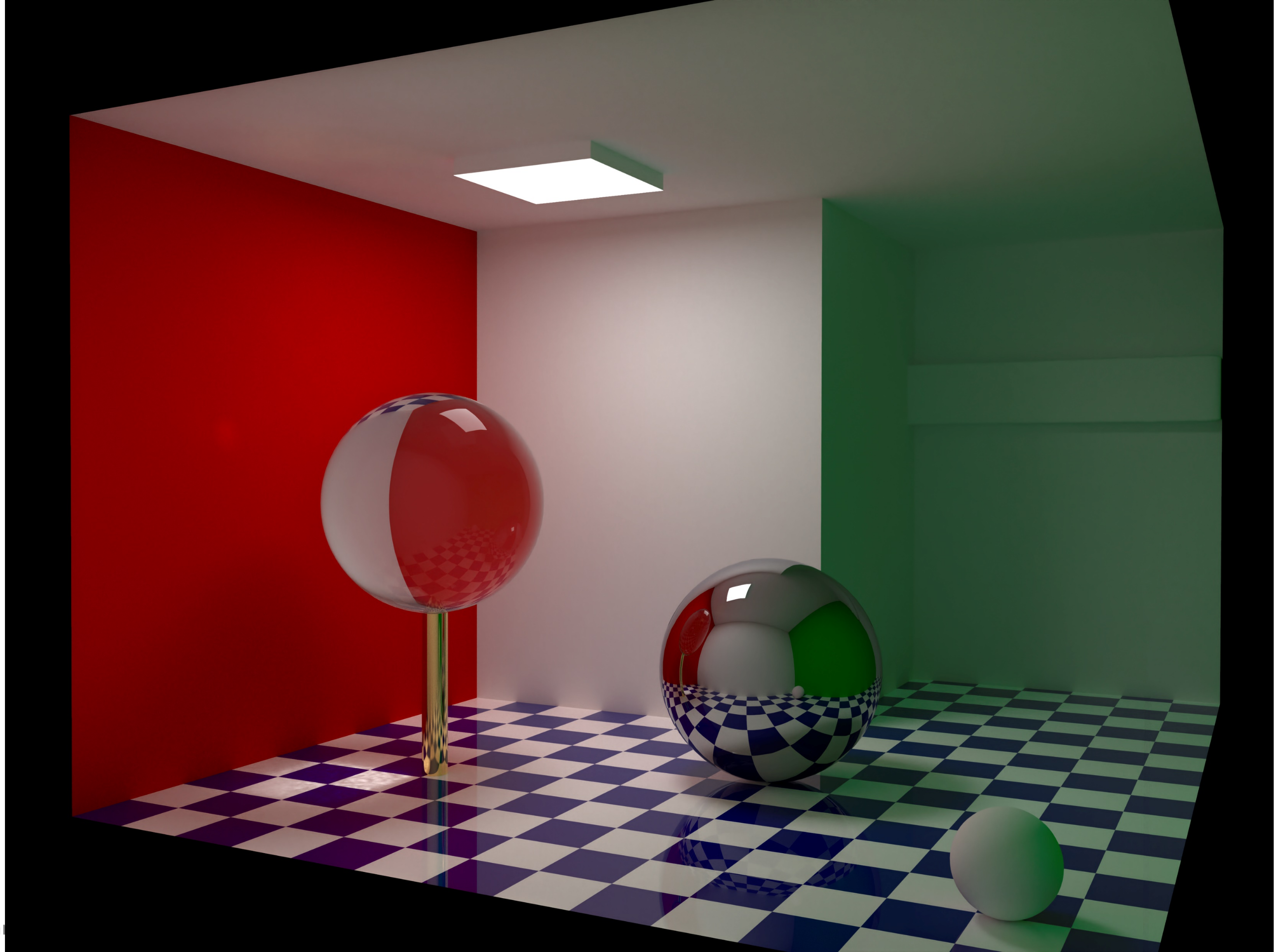
image



GPU

CMU 15-462/662





Computer graphics

Neural rendering

Deep image or video generation approaches that enable explicit or implicit control of scene properties such as illumination, camera parameters, pose, geometry, appearance and semantic structure

Neural rendering brings the promise of addressing both *reconstruction* and *rendering* by using deep networks to learn complex mappings from captured images to novel images

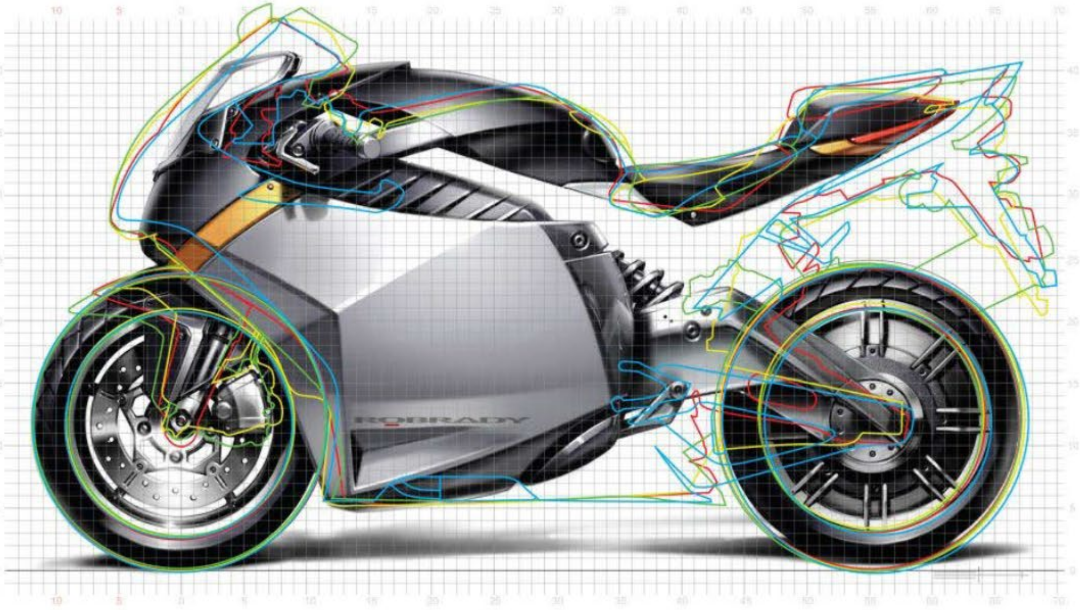
State of the Art on Neural Rendering, A. Tewari et al., 2020

SIGGRAPH 2020 Technical Papers Trailer

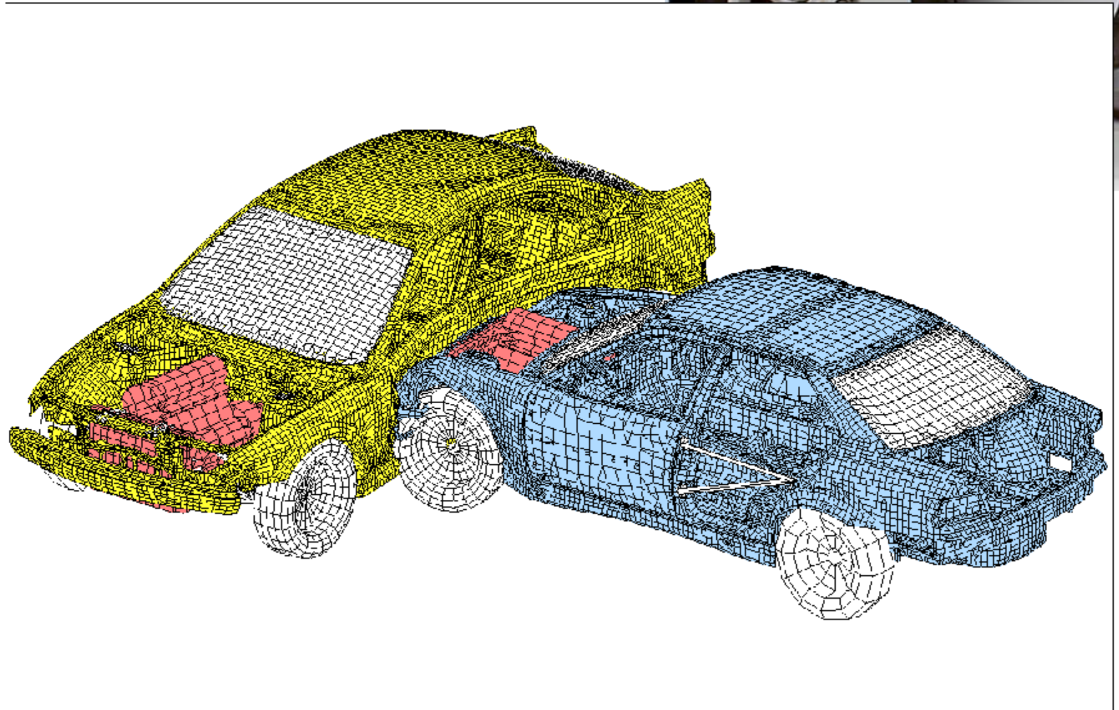
Computer graphics is everywhere

Entertainment (movies, games)

Industrial design

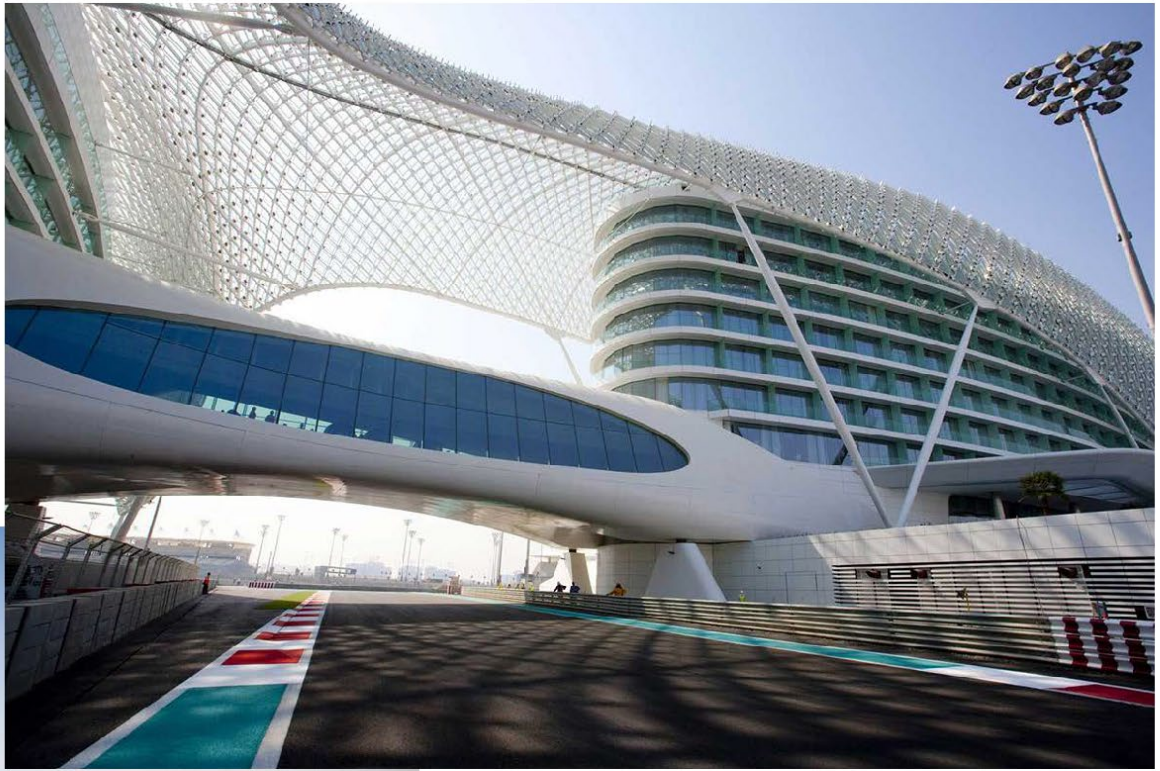


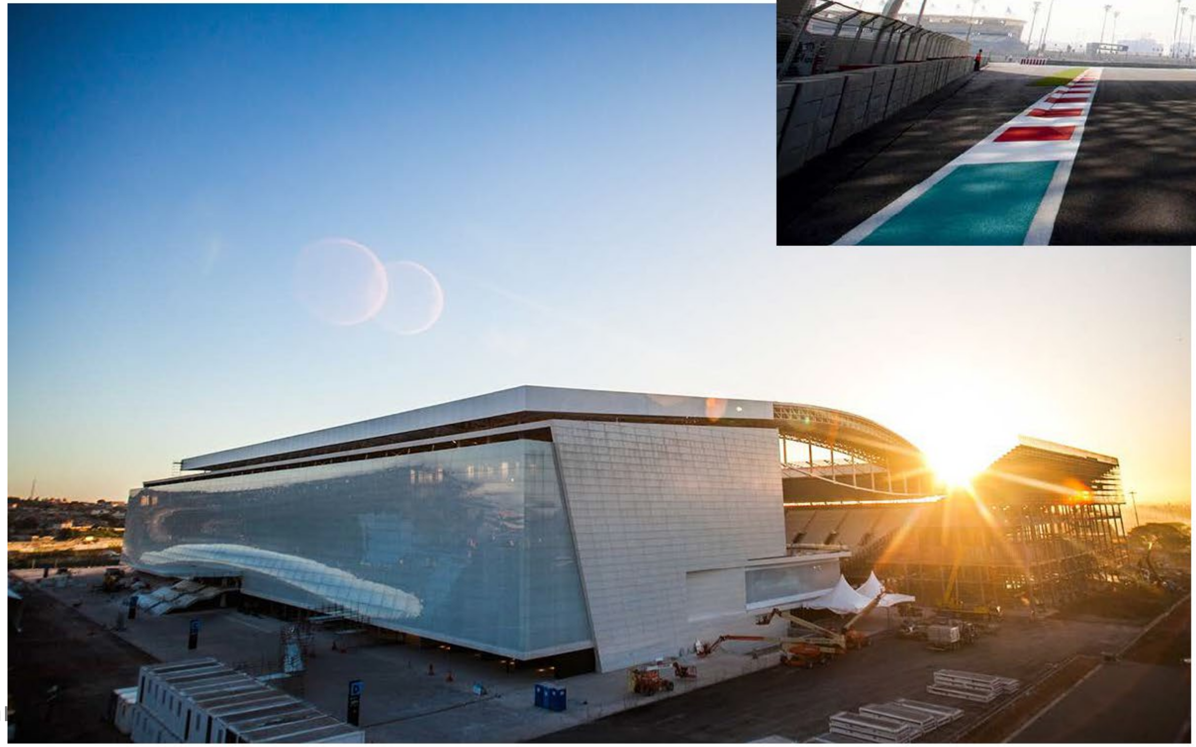
Computer aided engineering (CAE)



Courtesy : BMW

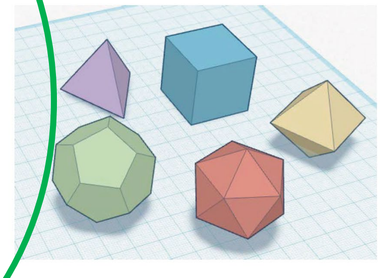
Architecture





Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

camera	geometry	materials	lights
			

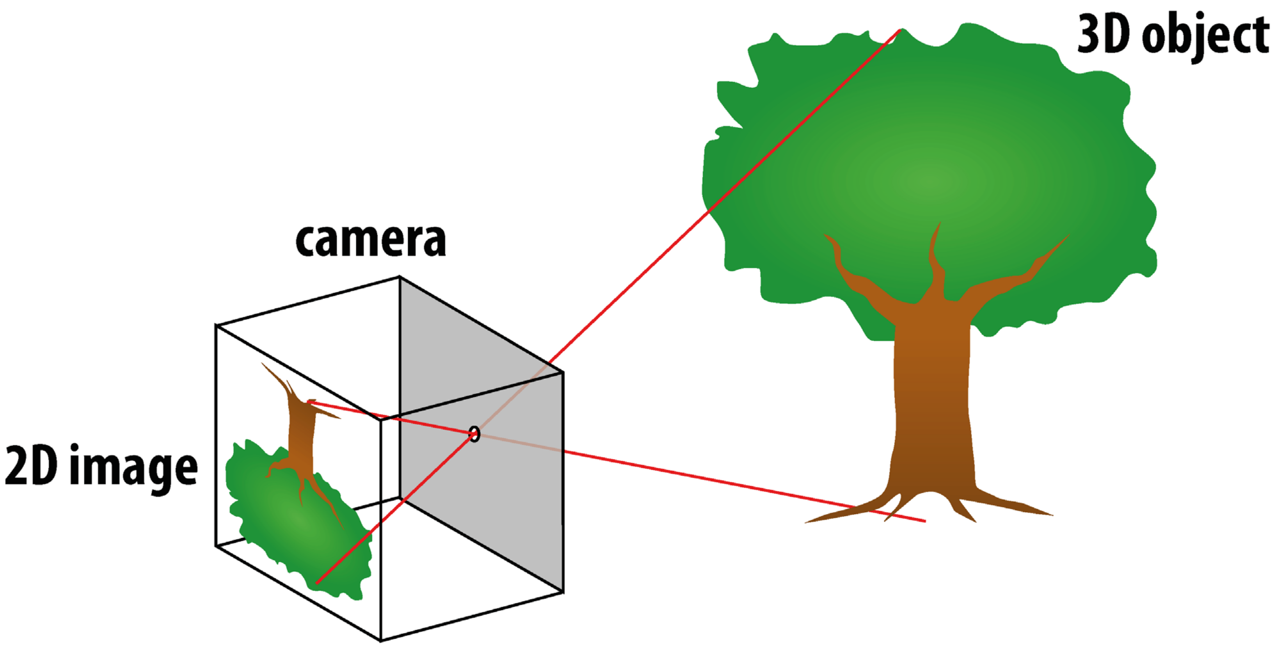
("scene")

Ray Tracer

image

Perspective projection
Pinhole camera model

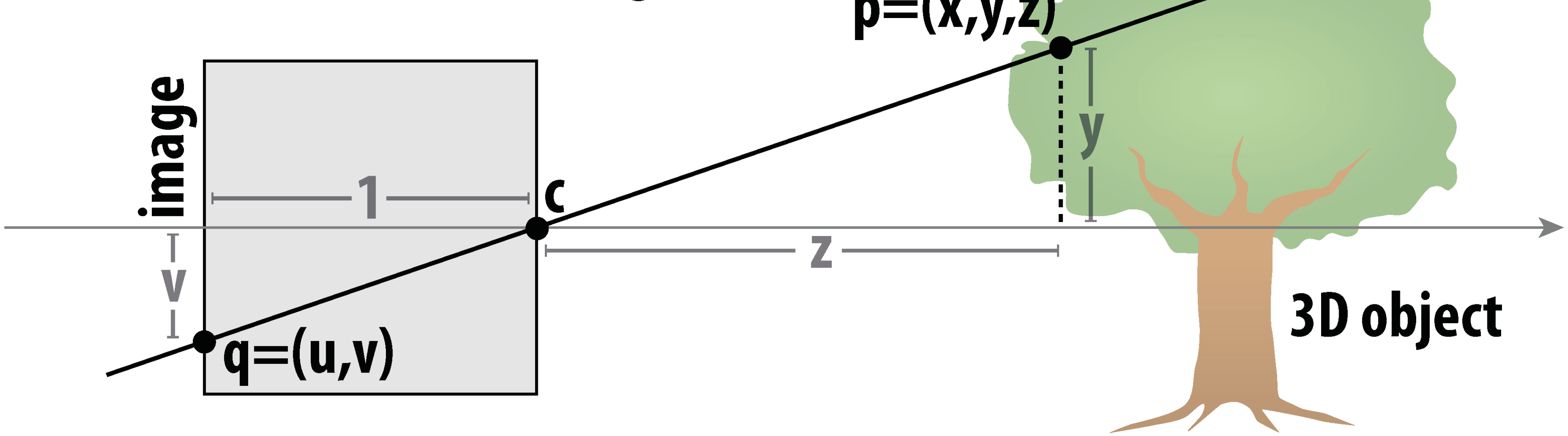
- Objects look smaller as they get further away
- Parallel lines “meet” at infinity



Perspective projection

Pinhole camera model

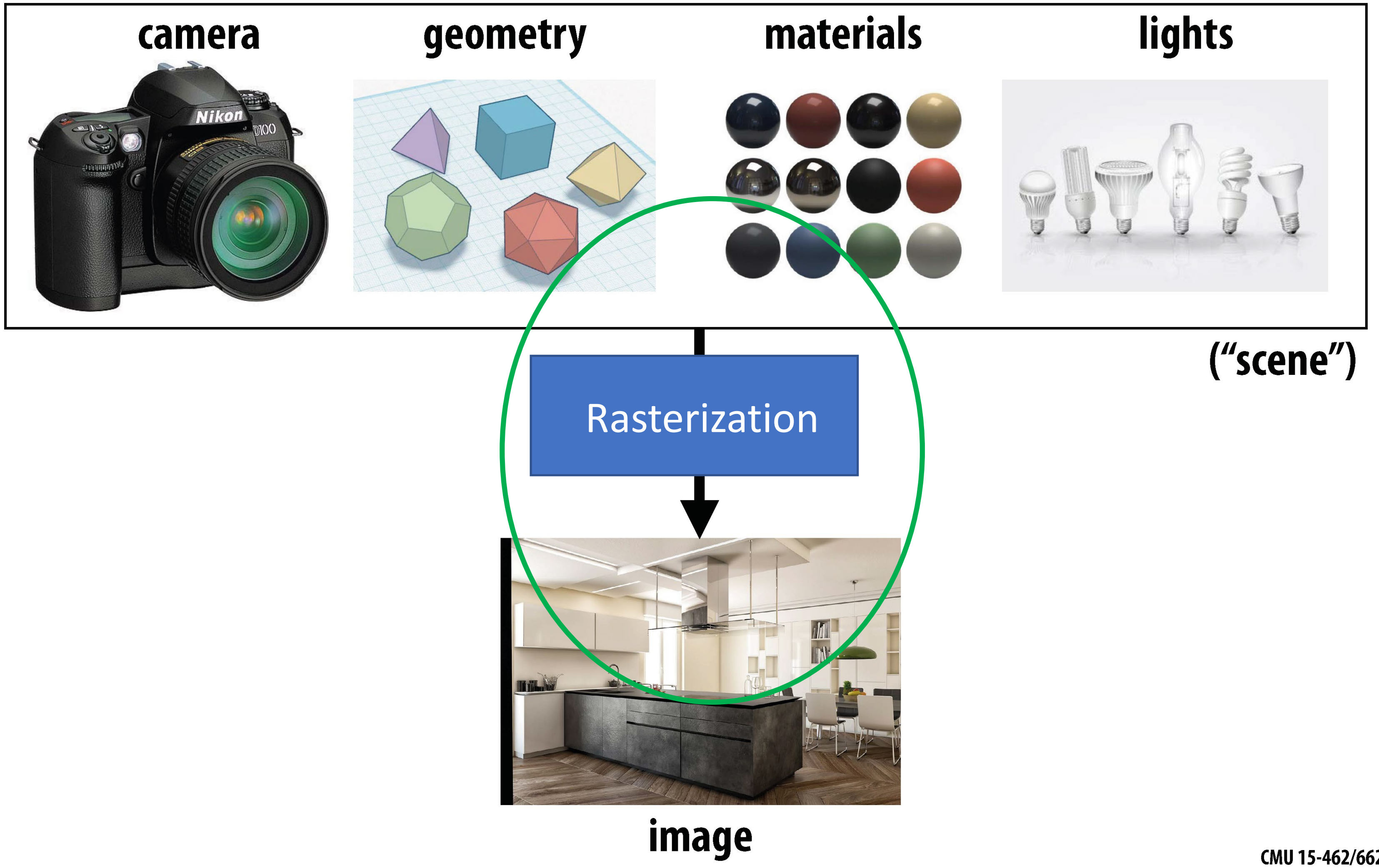
- **Notice two similar triangles:**



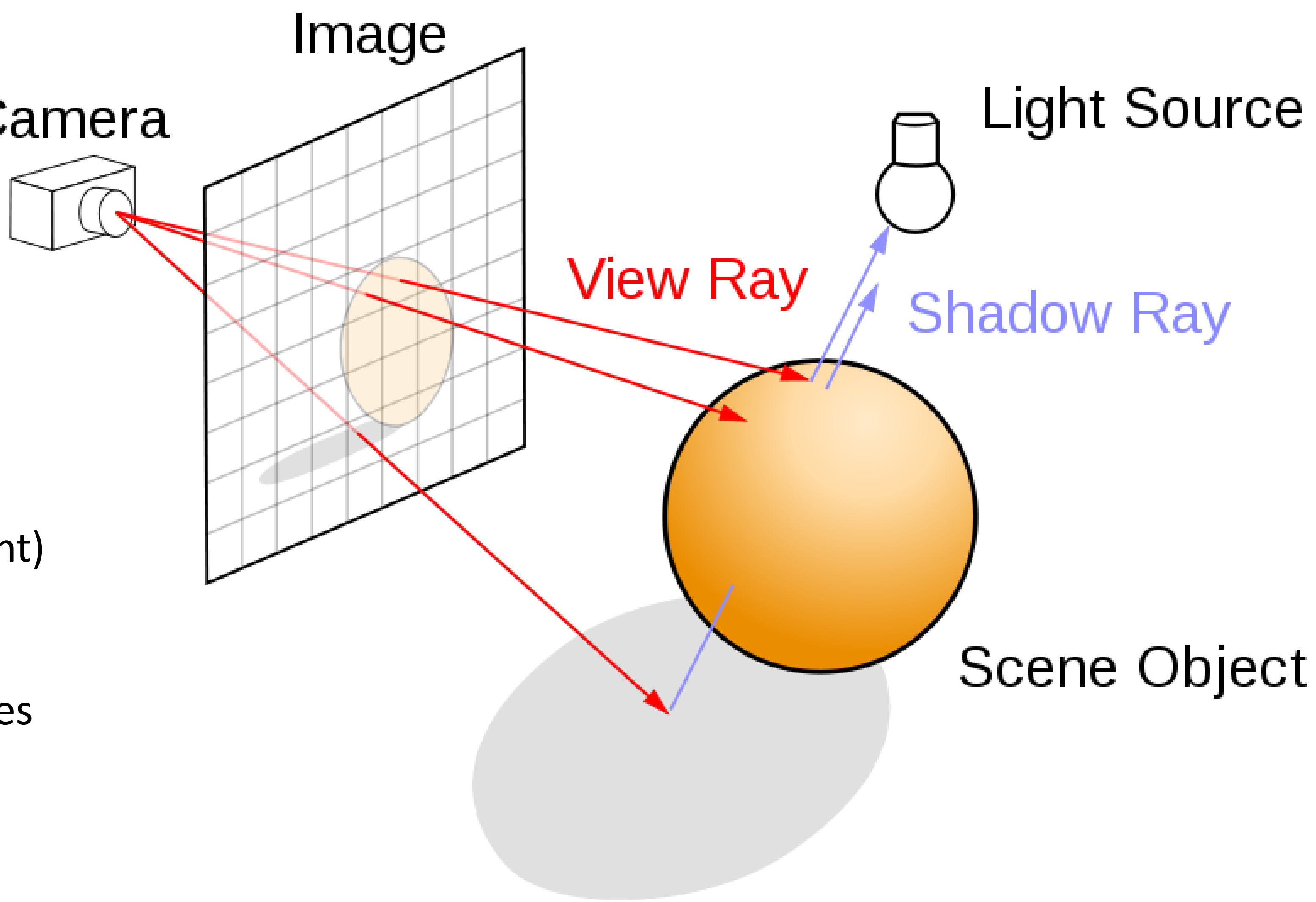
- Camera pinhole at $c = (0,0,0)$
- The image plane located $z = -1$
- Using similar triangles $v = \frac{y}{z}$ and $u = \frac{x}{z}$

Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?



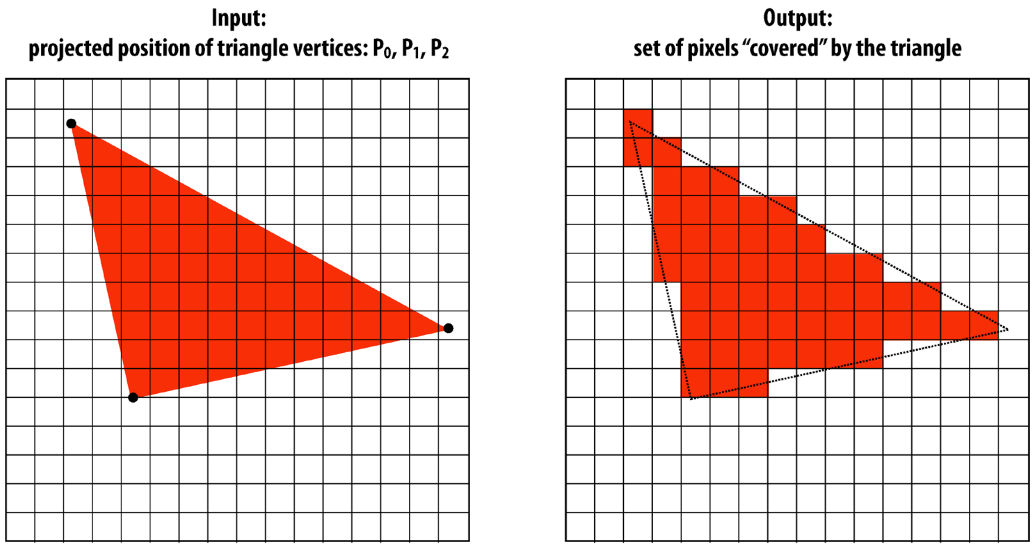
Rendering
Drawing on the screen

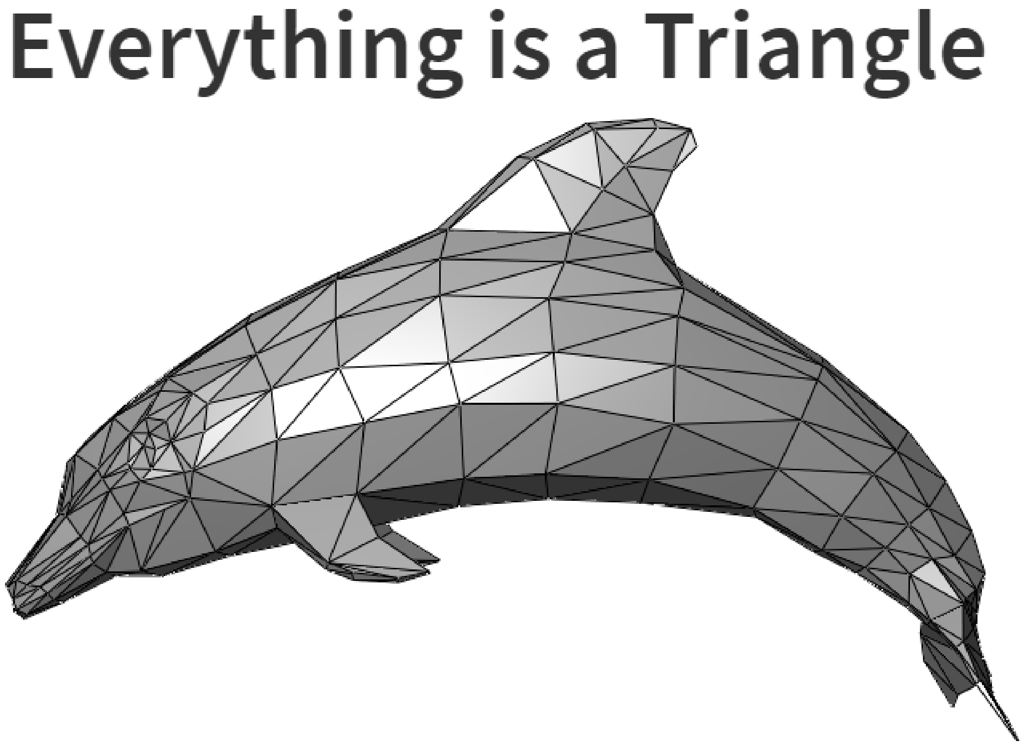


- camera (view point)
- light sources
- geometry
- material properties

Rendering
Drawing on the screen

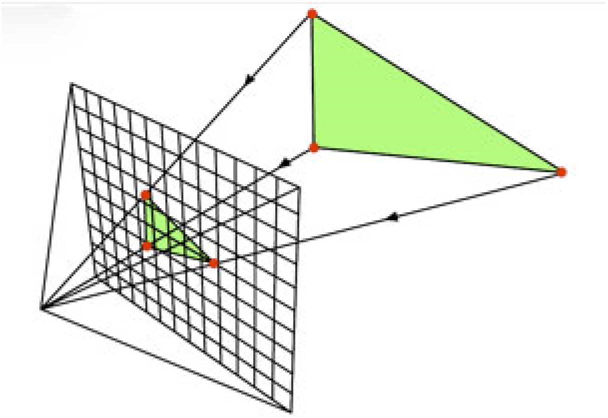
Two ways of turning triangles into image
- Rasterization
- Ray tracing



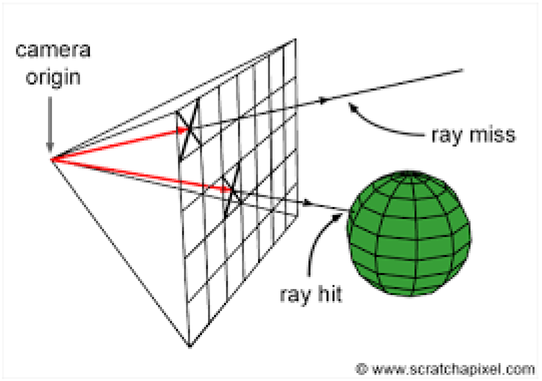


Rendering
Drawing on the screen

Rasterization
- for each primitive (triangle), which pixels are covered?
- extremely fast (Billions of triangles per second on GPU)
- harder (but possible) to achieve photorealism
- games and real-time applications



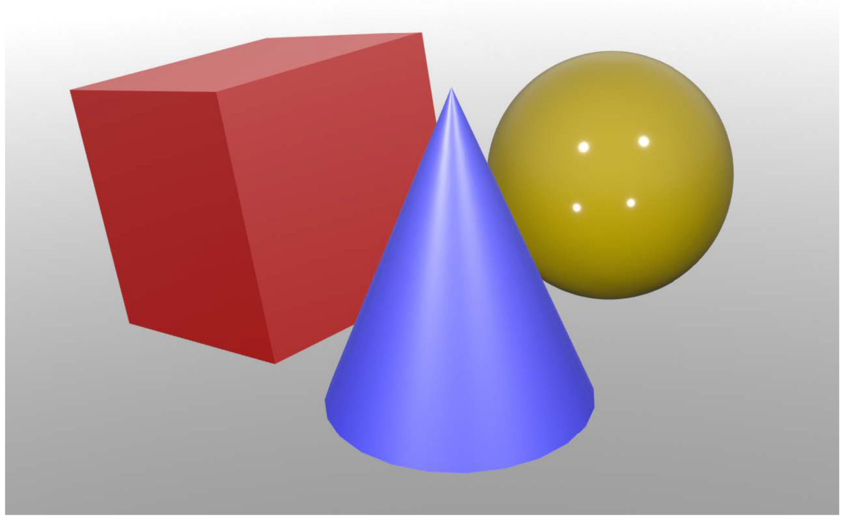
Ray tracing
- for each pixel, which primitives (triangles) are seen?
- generally slower
- easier to get photorealism
- movies and video clips



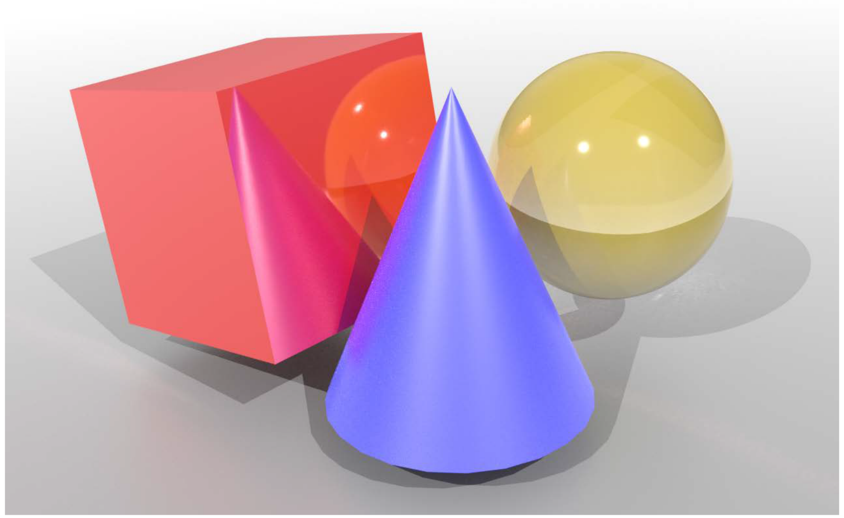
Ray Tracing vs. Rasterization—Illumination

- More major difference: sophistication of illumination model
  - [LOCAL] rasterizer processes one *primitive* at a time; hard* to determine things like "A is in the shadow of B"
  - [GLOBAL] ray tracer processes on *ray* at a time; ray knows about everything it intersects, easy to talk about shadows & other "global" illumination effects

RASTERIZATION



RAY TRACING



Q: What illumination effects are missing from the image on the left?

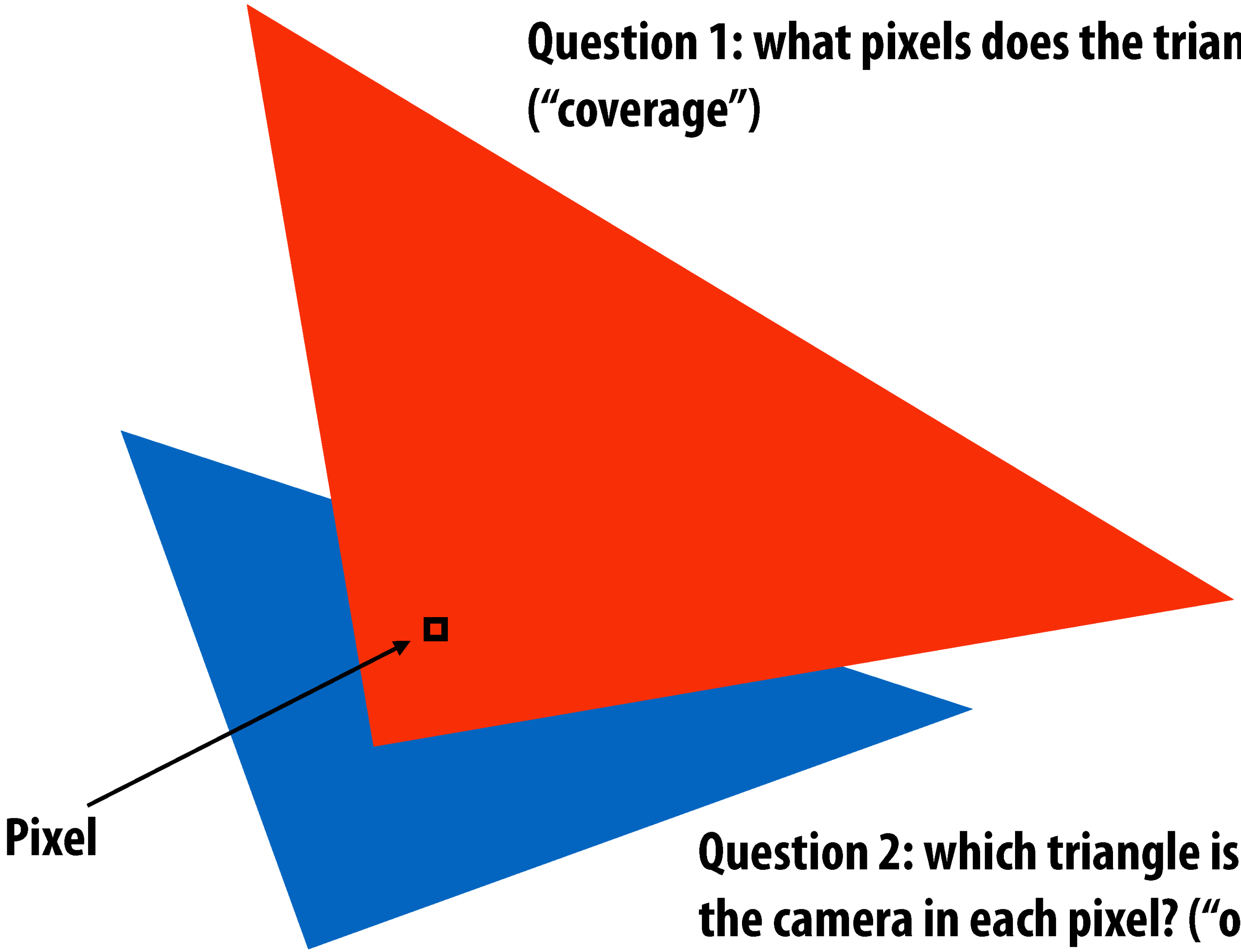
*But not *impossible* to do <u>some</u> things with rasterization (e.g., shadow maps)… just results in more complexity

Rendering

The visibility problem

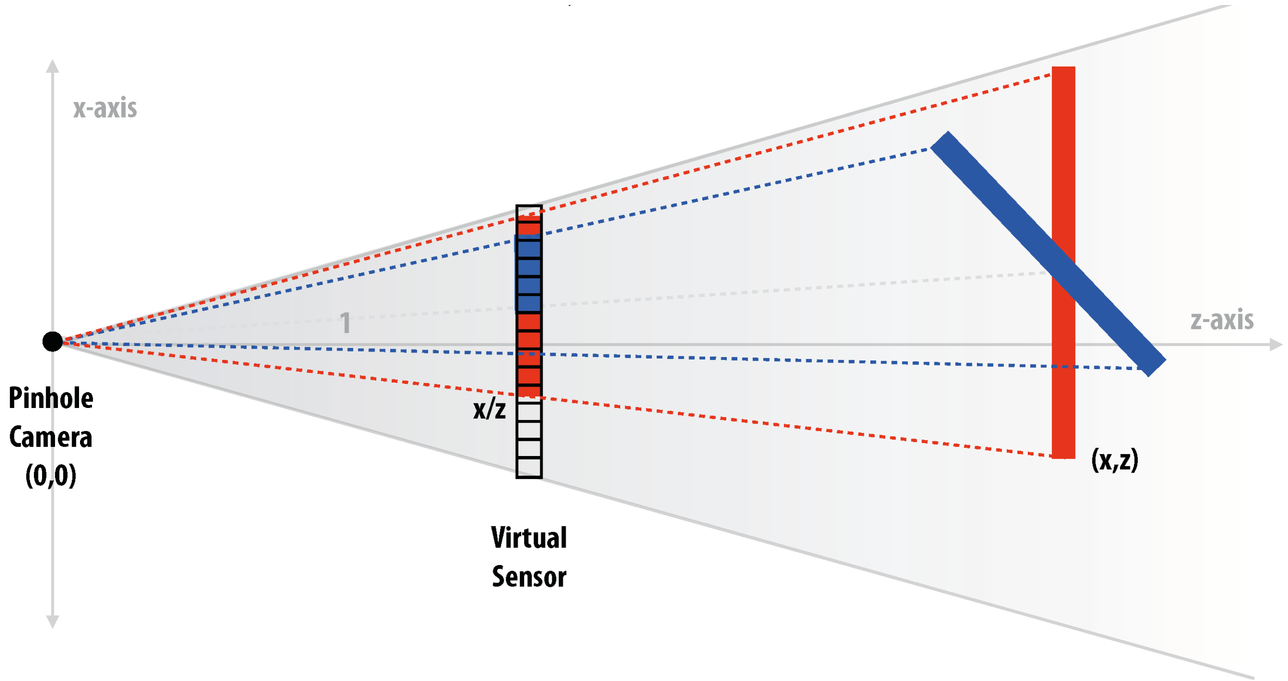
rasterization

Question 1: what pixels does the triangle overlap? ("coverage")



Question 2: which triangle is closest to the camera in each pixel? ("occlusion")

Rendering
The visibility problem
ray tracing



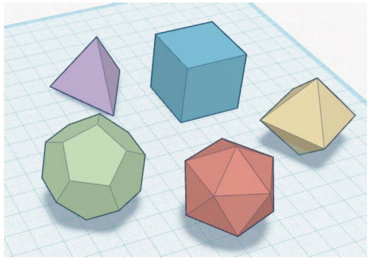
- **Visibility problem in terms of rays:**
  - **COVERAGE: What scene geometry is hit by a ray from a pixel through the pinhole?**
  - **OCCLUSION: Which object is the <u>first</u> hit along that ray?**

Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

camera

geometry



materials

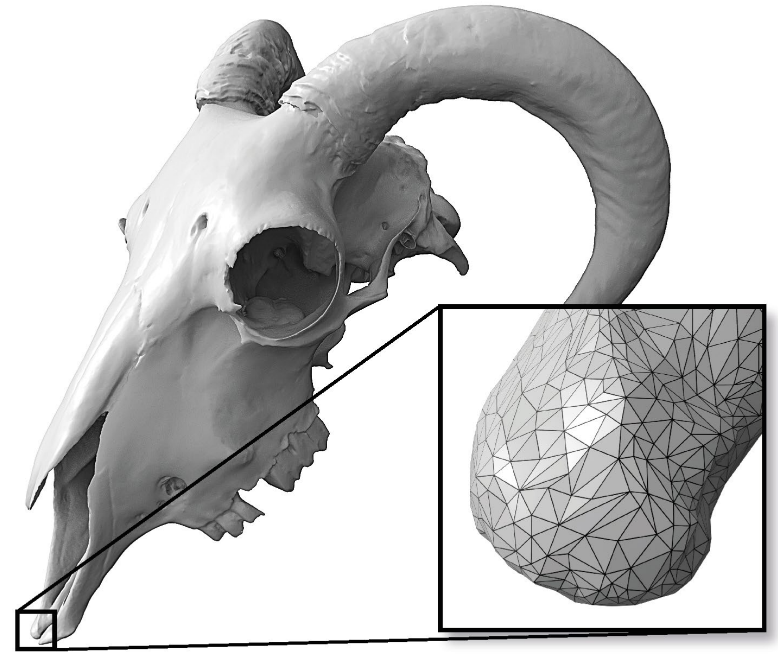
lights

("scene")

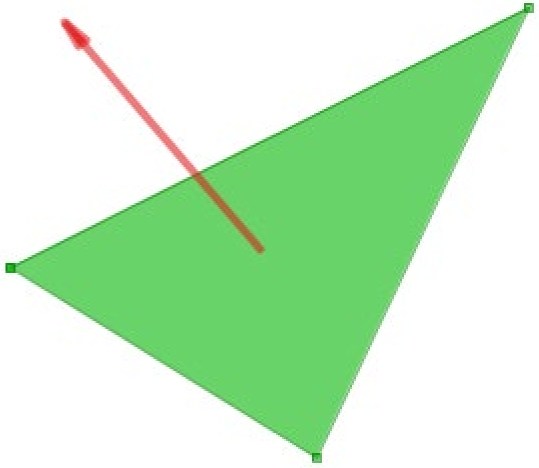
Rasterization

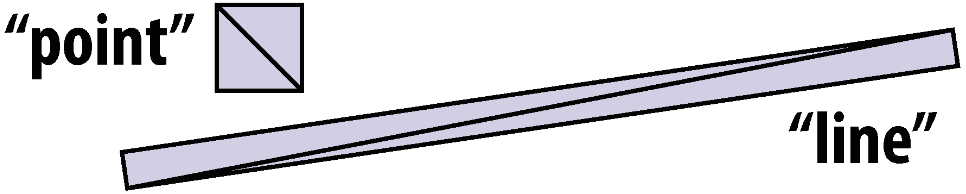
image

Geometry
Why triangles?



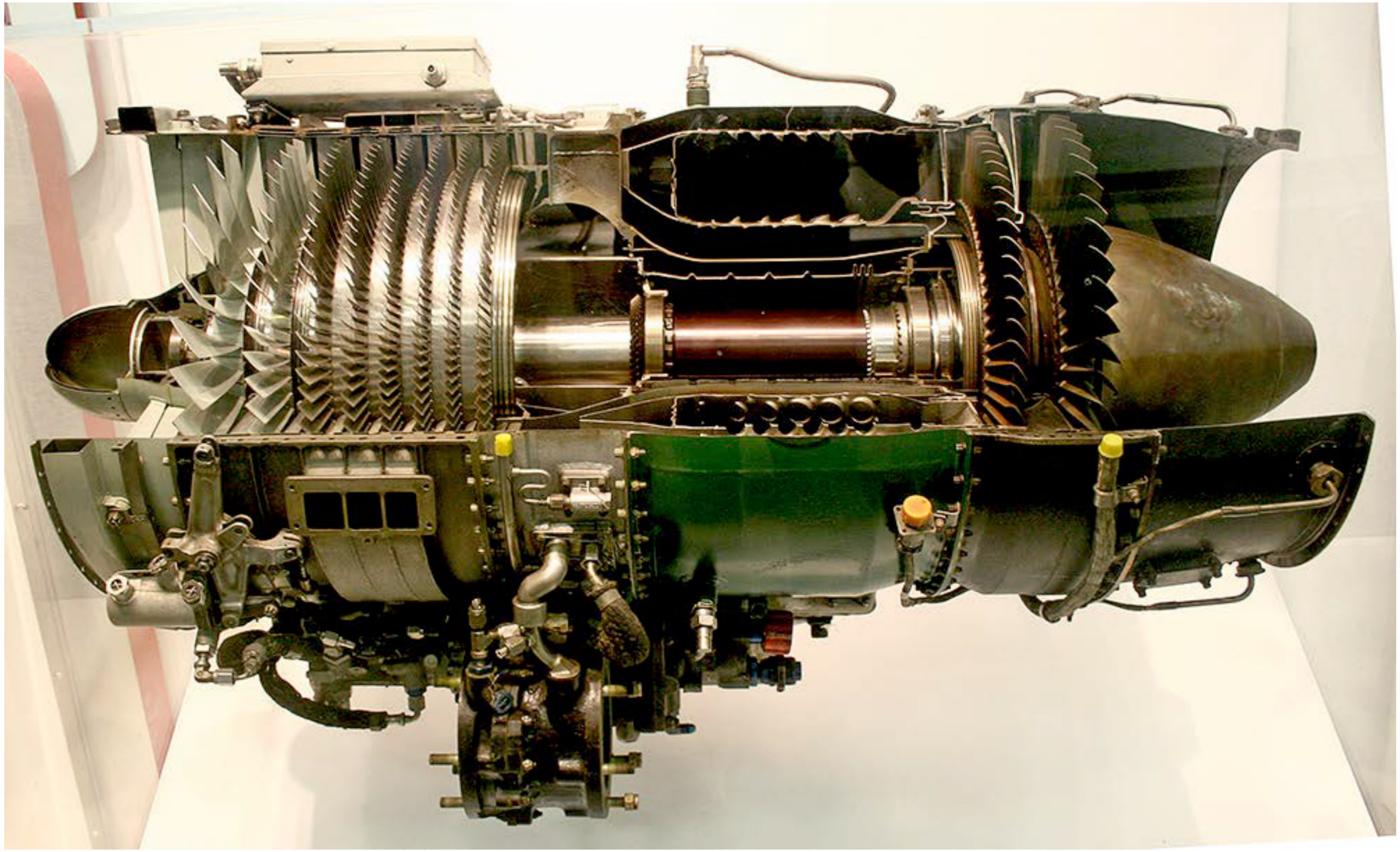
- can approximate any shape
- always planar, well-defined normal
- easy to interpolate data, using "barycentric coordinates"
- optimized and uniform drawing pipeline





Examples of geometry

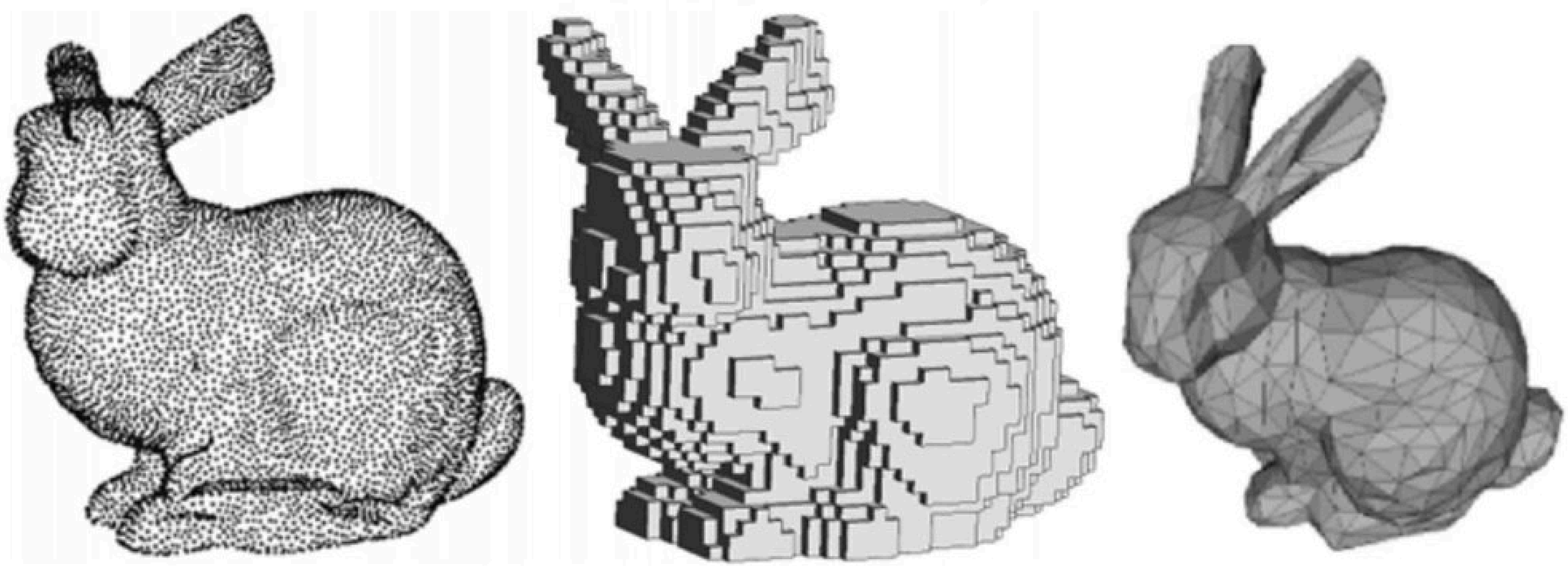
Examples of geometry



Examples of geometry

Geometry
Scene representation

Explicit (discretization of the object geometry)
- point cloud
- voxels
- polygon mesh
- triangle mesh

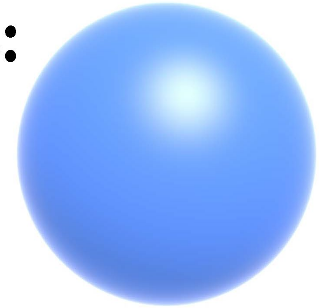


Geometry
Scene representation

Implicit (continuous representation)

- algebraic surfaces
- level set $f: R^3 \to R,\ f(x, y, z) = 0$
- more general, signed distance function

■ **Examples:**



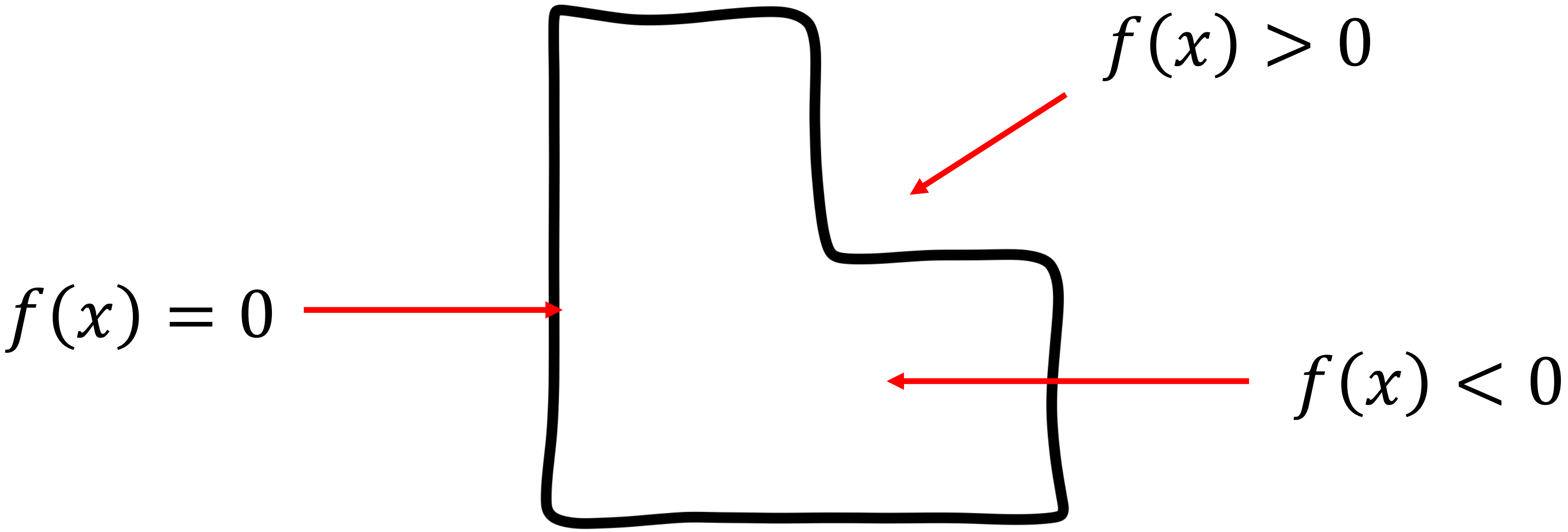
$x^2+y^2+z^2=1$

$(R - \sqrt{x^2+y^2})^2 + z^2 = r^2$

Geometry

Scene representation

Implicit shapes



Surface represented implicitly

$$s = \{x \in \mathbb{R}^3 | f(x) = 0\}$$

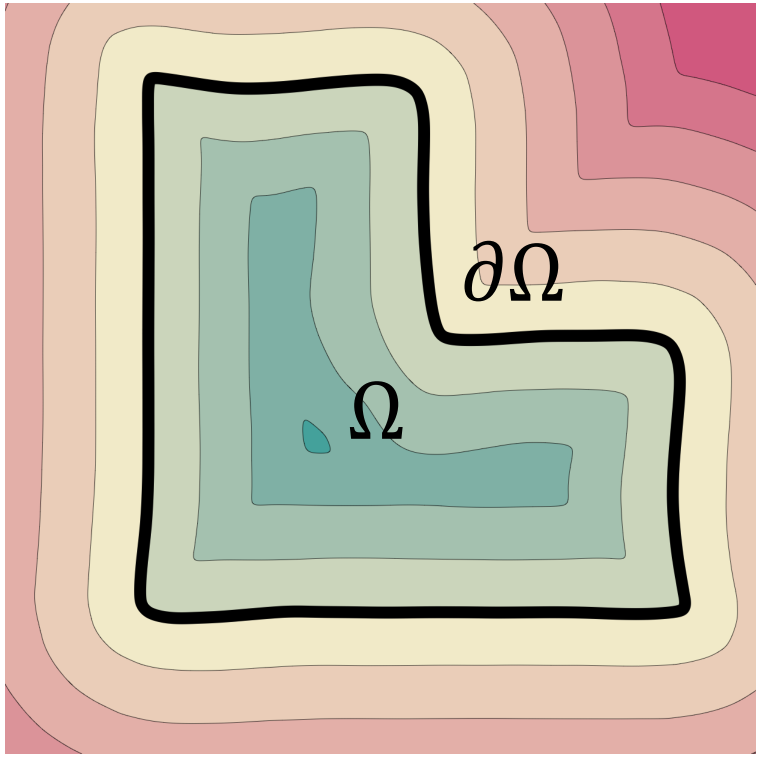
Geometry

Scene representation

Implicit shapes

Eikonal equation

$$\|\nabla f(\boldsymbol{x})\| = 1, \boldsymbol{x} \in \Omega$$
$$f(\boldsymbol{x}) = 0, \boldsymbol{x} \in \partial\Omega$$



Signed distance function (SDF)

Geometry
Scene representation

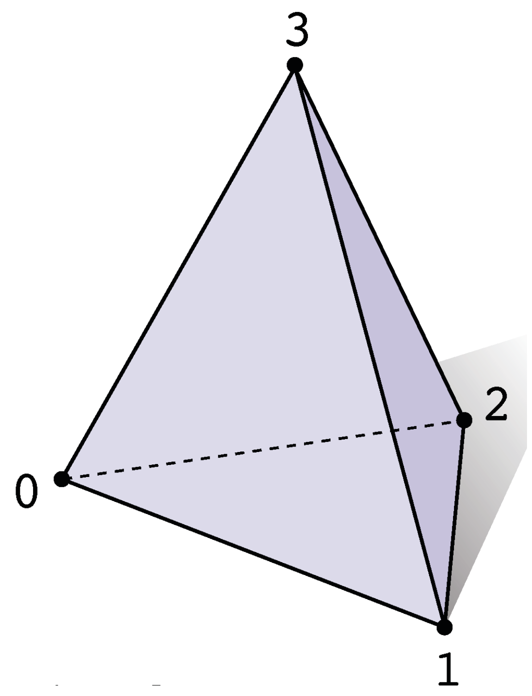
Implicit
- description can be compact (algebraic surfaces)
- easy to determine if a point in (inside / outside) on the shape
- expensive / not easy to generate all points of shape

Explicit (point cloud, triangle mesh)
- easy representation (list of points (x, y, z))
- hard to test whether a point is inside / outside the shape
- easy to generate the geometry

Geometry

Triangle mesh (explicit)

- store vertices as triplets of coordinates (x, y, z)
- store triangles as triplets of indices (i, j, k)



- **E.g., tetrahedron:**

VERTICES

	x	y	z
0:	−1	−1	−1
1:	1	−1	1
2:	1	1	−1
3:	−1	1	1

TRIANGLES

i	j	k
0	2	1
0	3	2
3	0	1
3	1	2

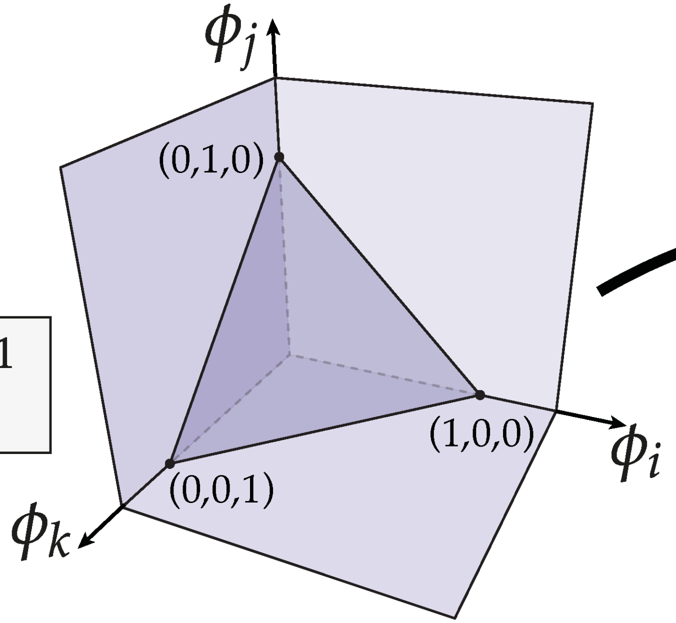
Geometry

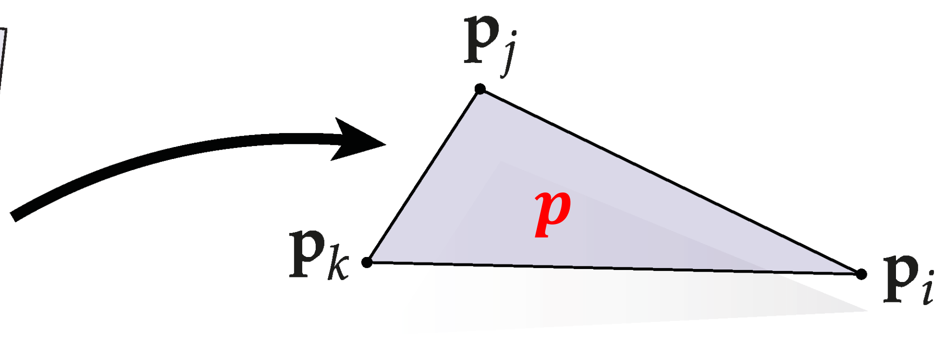
Triangle mesh

Barycentric coordinates $\boldsymbol{\phi_i, \phi_j, \phi_k}$

for interpolation inside triangles

$$\boxed{\begin{gathered}\phi_i + \phi_j + \phi_k = 1 \\ \phi_i, \phi_j, \phi_k > 0\end{gathered}}$$





$$p = \phi_i \mathbf{p}_i + \phi_j \mathbf{p}_j + \phi_k \mathbf{p}_k$$

CMU 15-462/662

Geometry

Triangle mesh

2D Linear interpolation

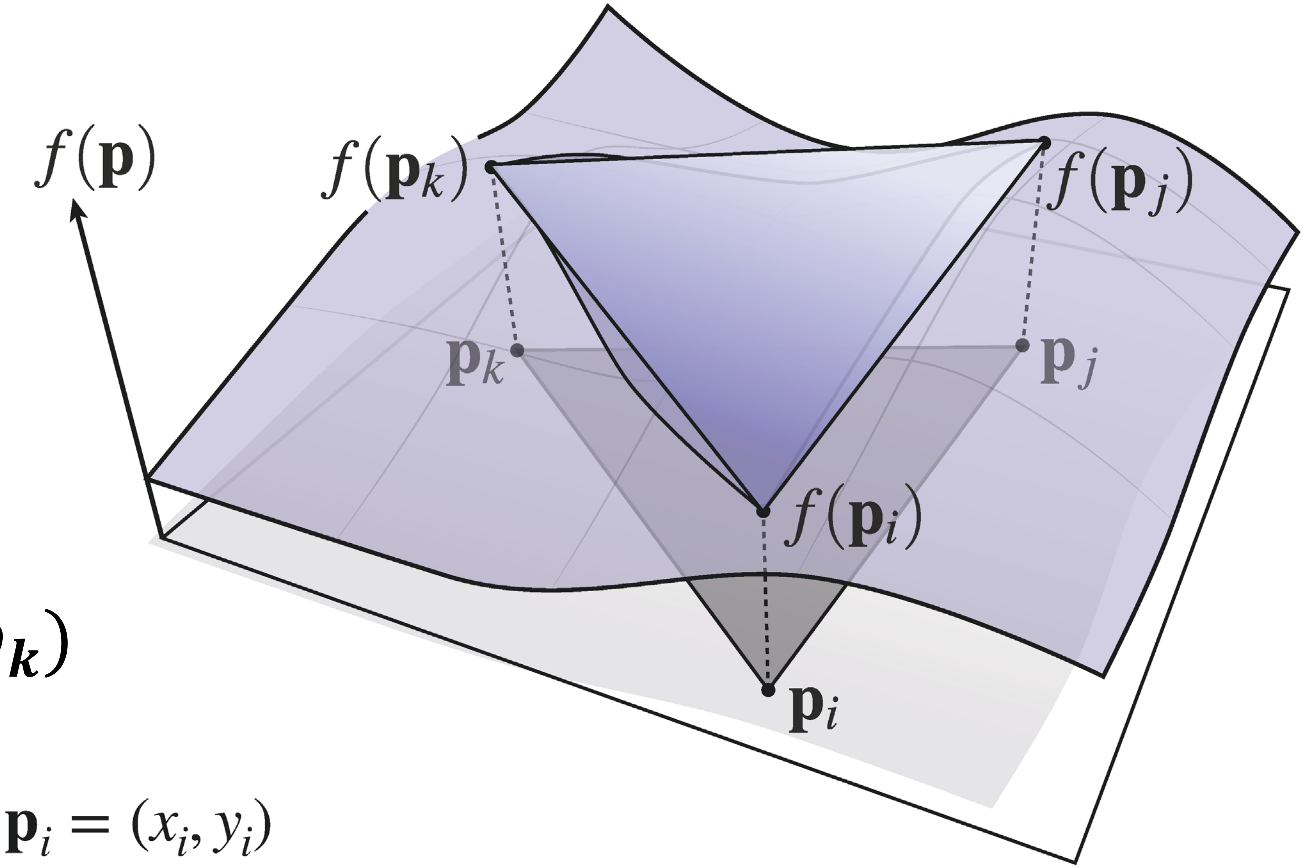
Look for a, b, c

$$\hat{f}(x, y) = ax + by + c$$

such that

$$\hat{f}(x_n, y_n) = f_n \quad n \in \{i, j, k\}$$

$$\hat{f}(\boldsymbol{p}) = \phi_i f(\boldsymbol{p_i}) + \phi_j f(\boldsymbol{p_j}) + \phi_k f(\boldsymbol{p_k})$$



$\mathbf{p}_i = (x_i, y_i)$

Geometry

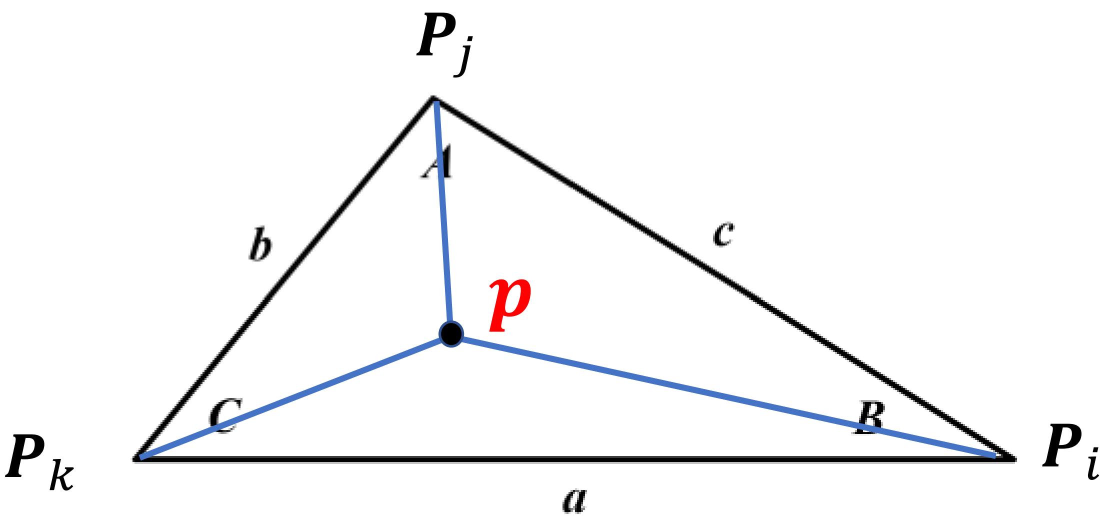
Triangle mesh

Barycentric coordinates $\boldsymbol{\phi_i, \phi_j, \phi_k}$

$$0 \leq \phi_i, \phi_j, \phi_k \leq 1$$

$$\phi_i + \phi_j + \phi_k = 1$$

$$\boldsymbol{p} = \phi_i \boldsymbol{P}_i + \phi_j \boldsymbol{P}_j + \phi_k \boldsymbol{P}_k$$



$$\phi_i = \frac{\Delta APC}{\Delta ABC}$$

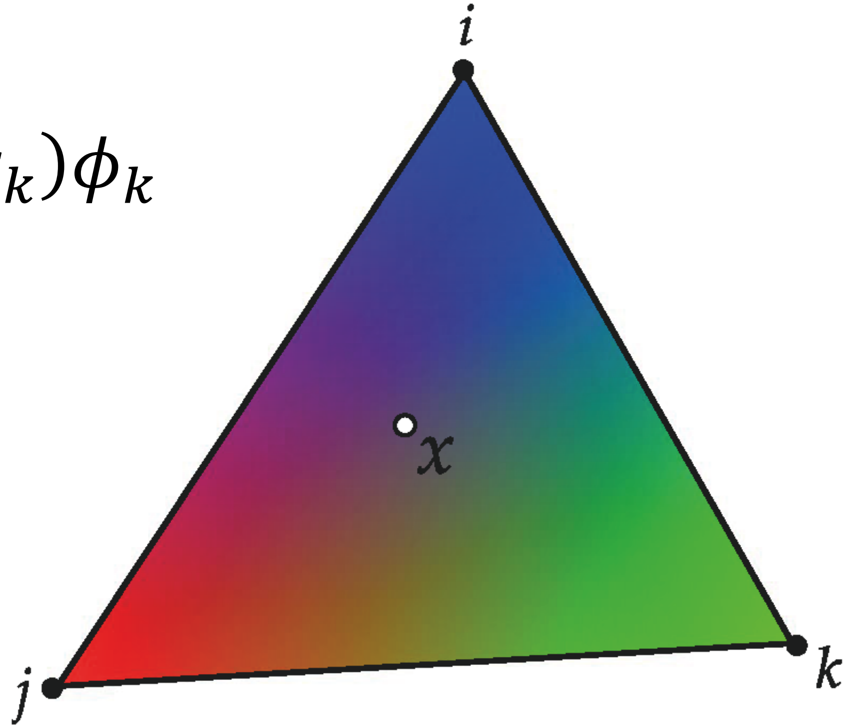
$$\phi_j = \frac{\Delta CPB}{\Delta ABC}$$

$$\phi_k = \frac{\Delta BPA}{\Delta ABC}$$

Geometry
Triangle mesh

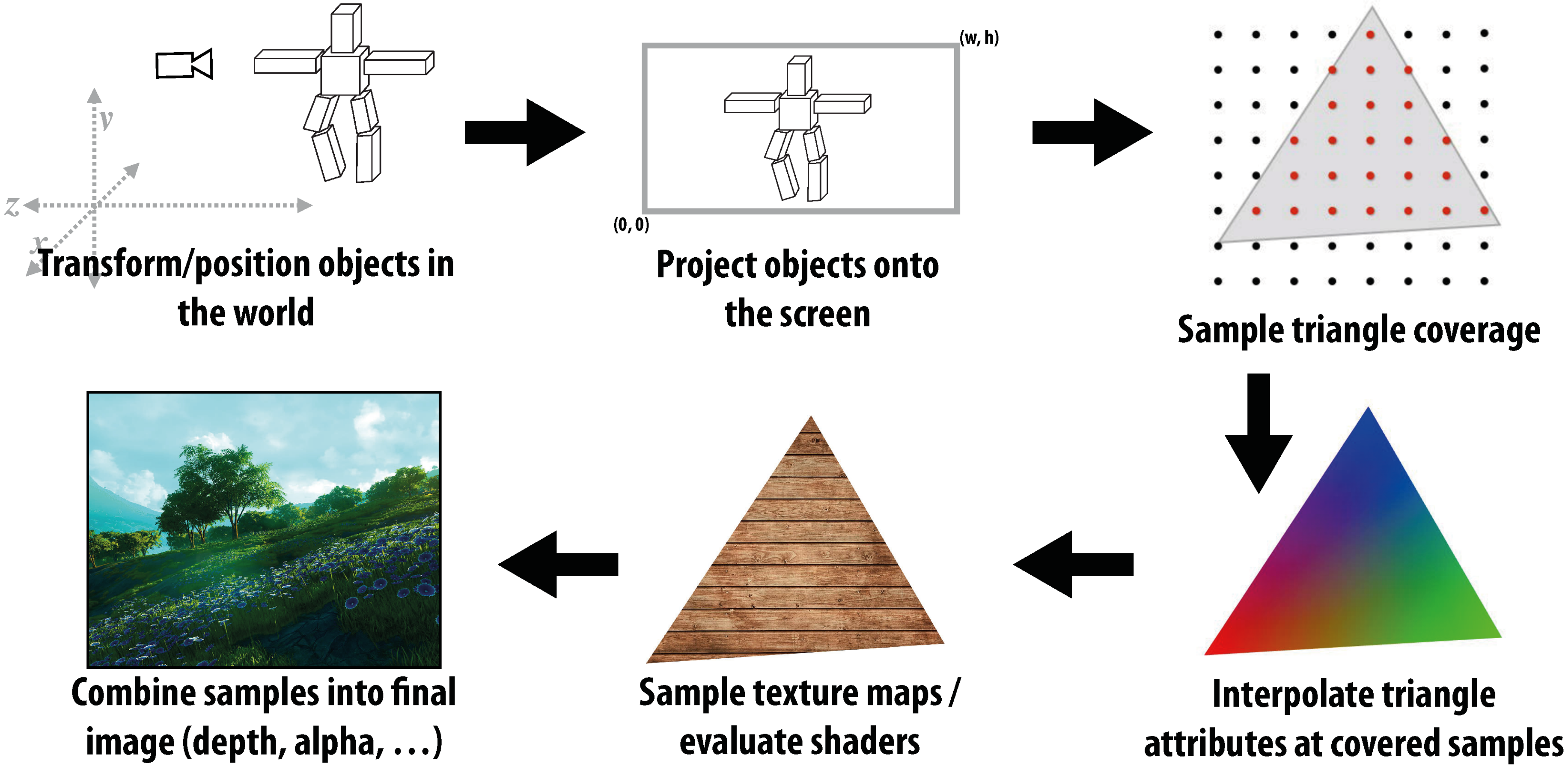
Barycentric coordinates can be used to interpolate any attribute associated with vertices, e.g. color, texture coordinates

$$\text{color}(x) = \text{color}(x_i)\phi_i + \text{color}(x_j)\phi_j + \text{color}(x_k)\phi_k$$



The Rasterization Pipeline

Rough sketch of rasterization pipeline:

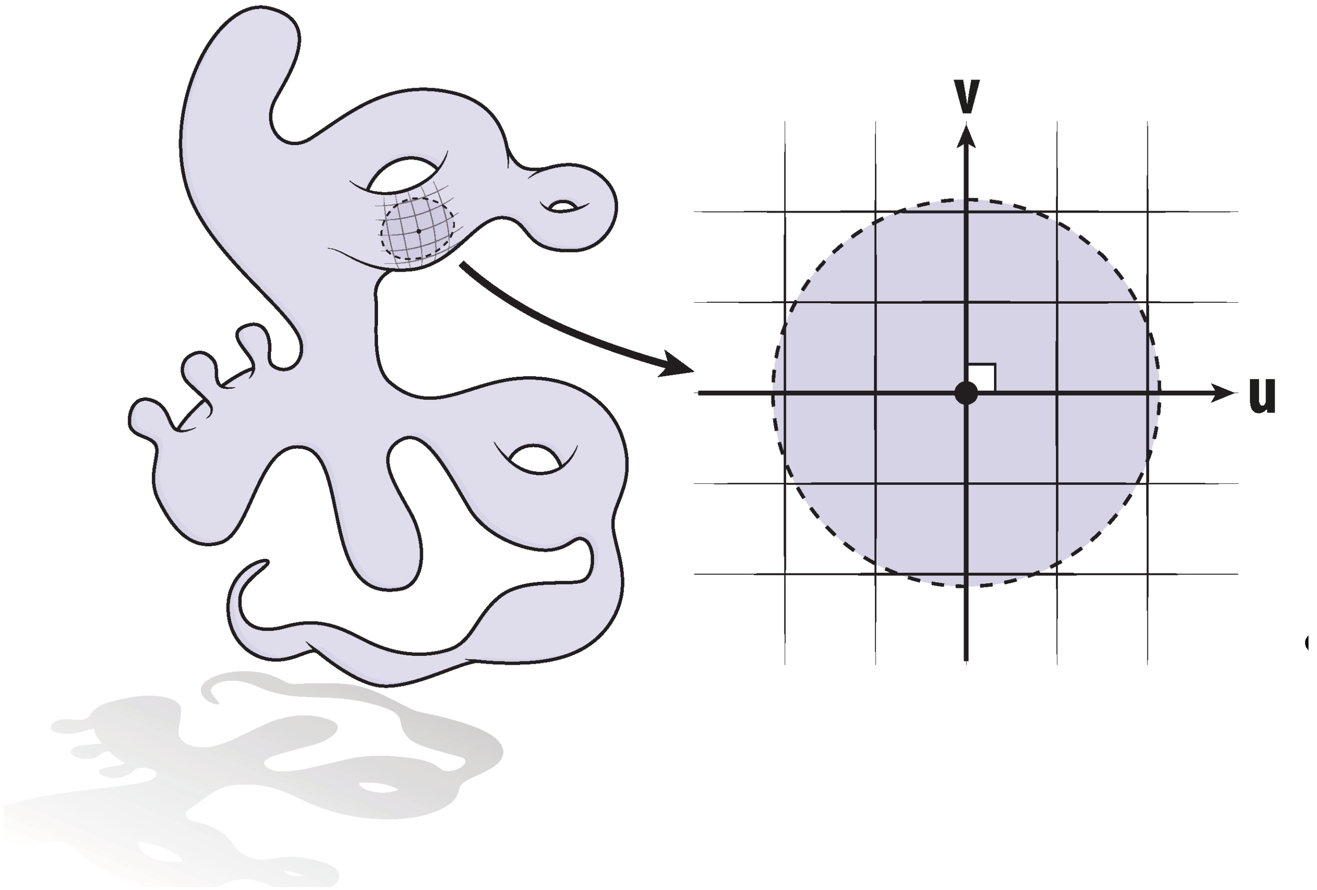


- **Reflects standard "real world" pipeline (OpenGL/Direct3D)**

Geometry

Surfaces, manifolds and meshes

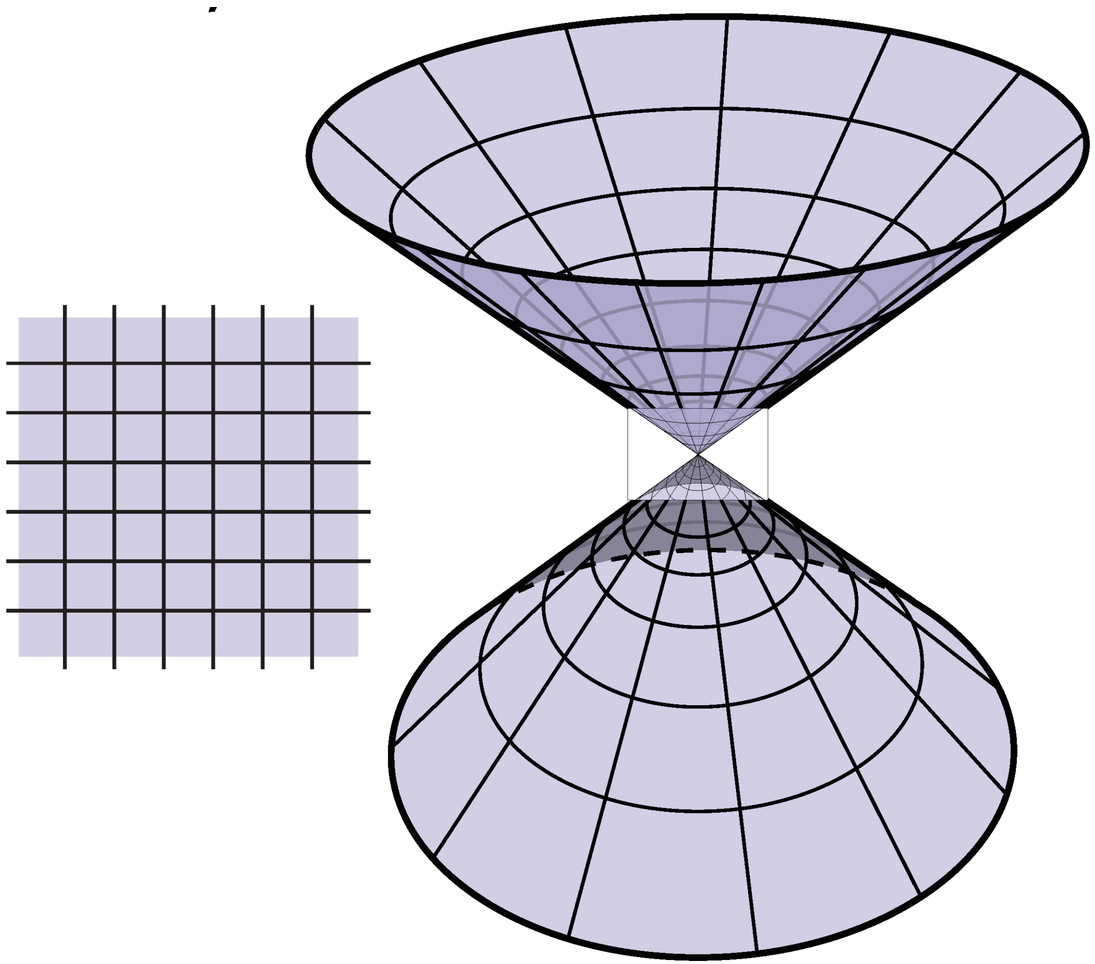
- Intuitively, a surface is the boundary or “shell” of an object
- Surfaces are manifold
  - if you zoom in,
    you can draw a regular coordinate grid



Geometry

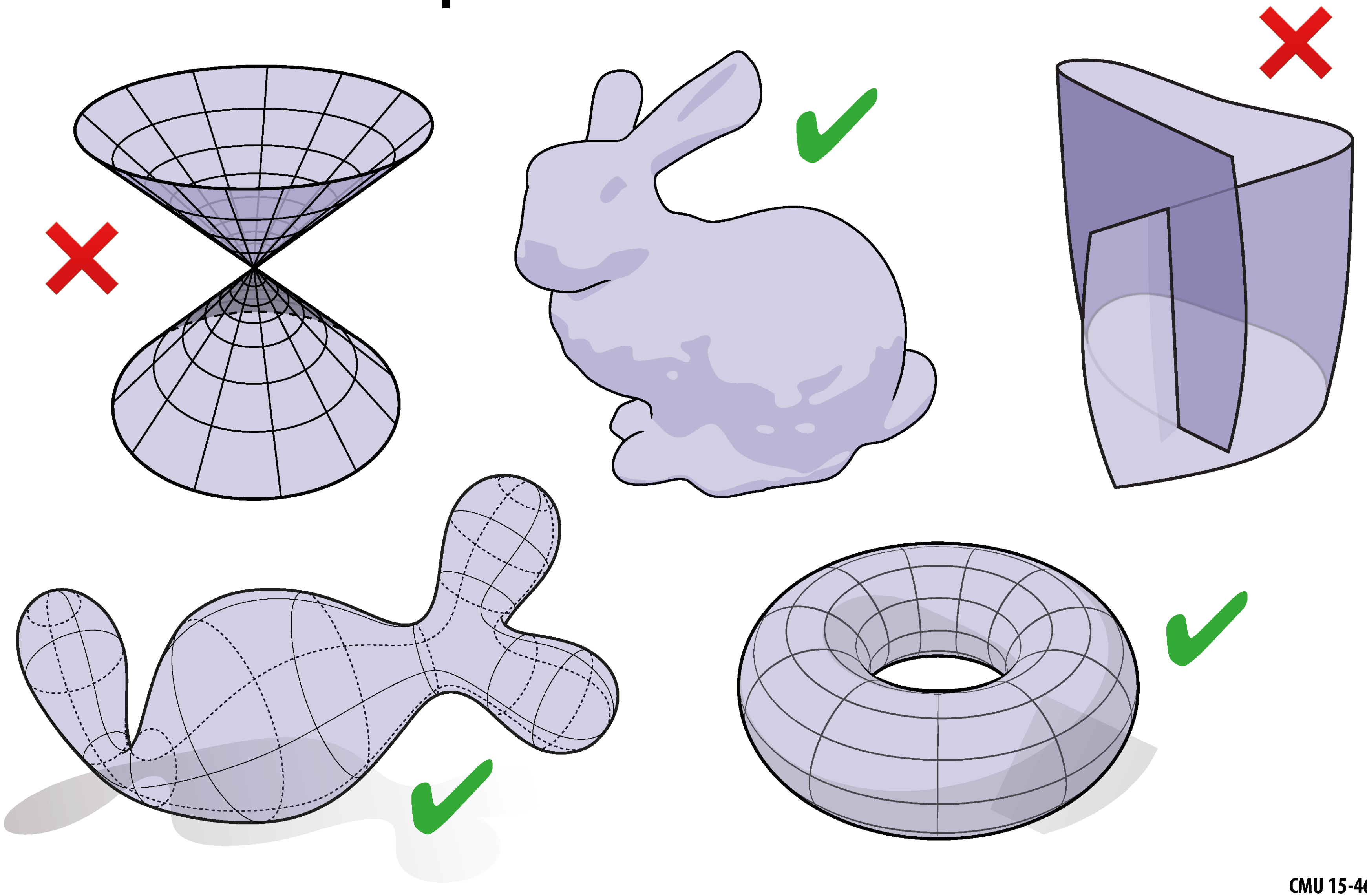
Surfaces, manifolds and meshes

Example. Non-manifold shape
can’t draw ordinary 2D grid at the center
no matter how close we get



Examples—Manifold vs. Nonmanifold

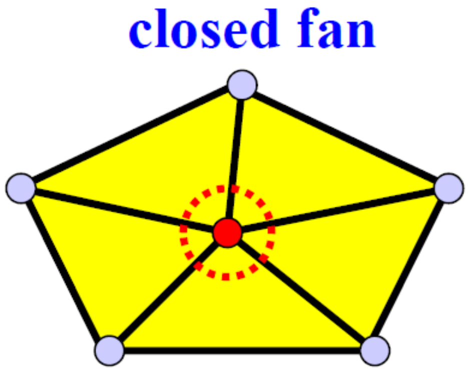
- **Which of these shapes are manifold?**

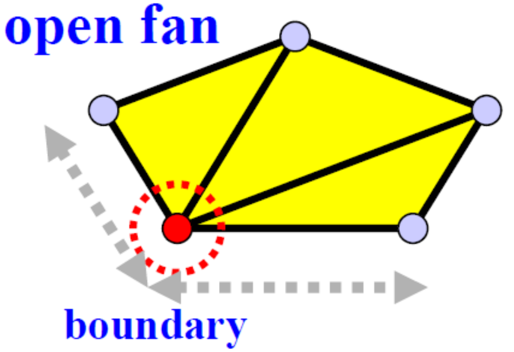


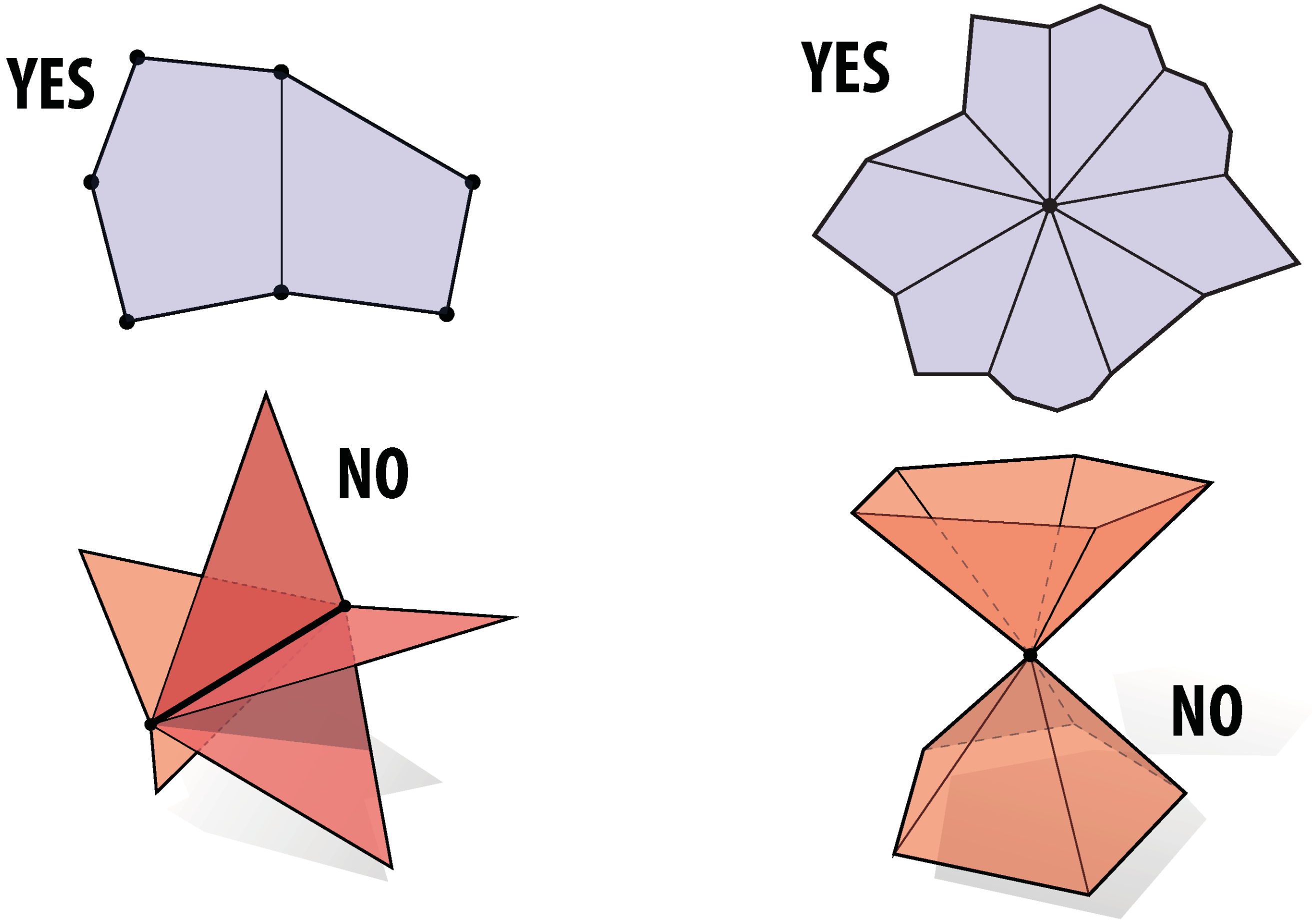
Geometry
Surfaces, manifolds and meshes

For polygonal mesh, easy to check whether it is a manifold

- every edge is incident to only one or two polygons
- the polygons incident to a vertex form a closed or an open fan



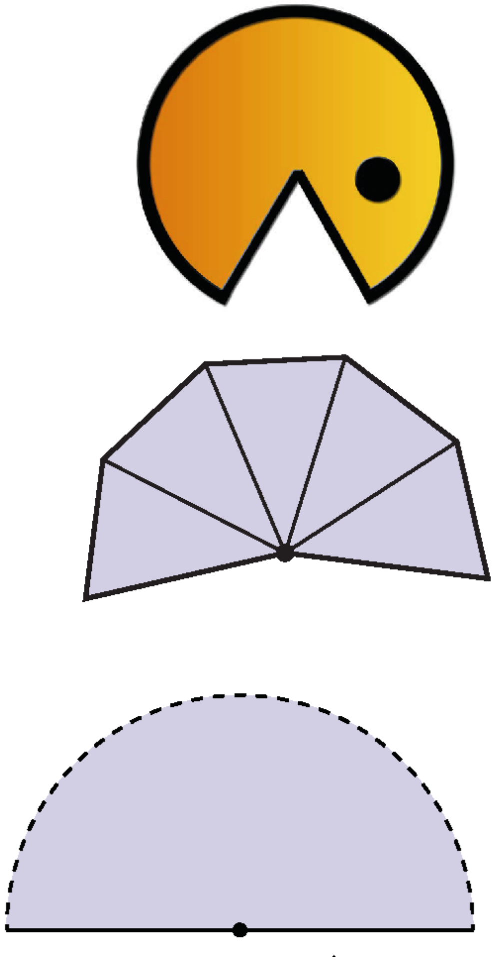




Geometry

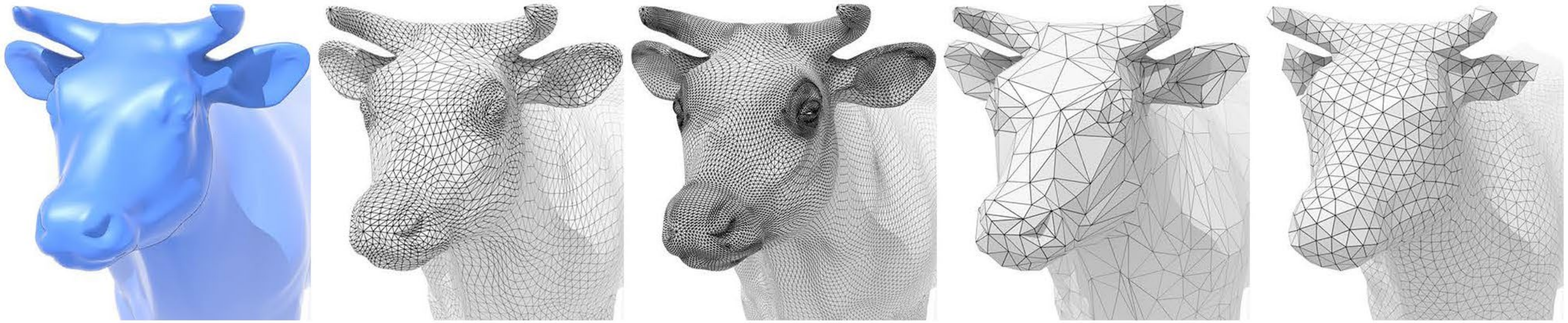
Surfaces, manifolds and meshes

- What about the boundary of the surface?
- The boundary is where the surface "ends"
- Locally, looks like a half disk
- Globally, each boundary forms a loop
- Polygon mesh
  - one polygon per boundary edge
  - boundary vertex looks like a "pacman"



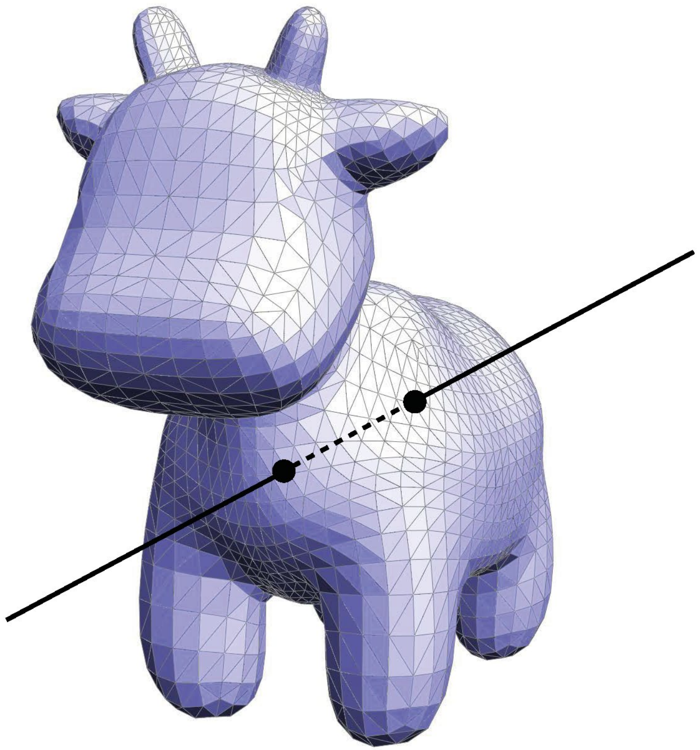
Geometry

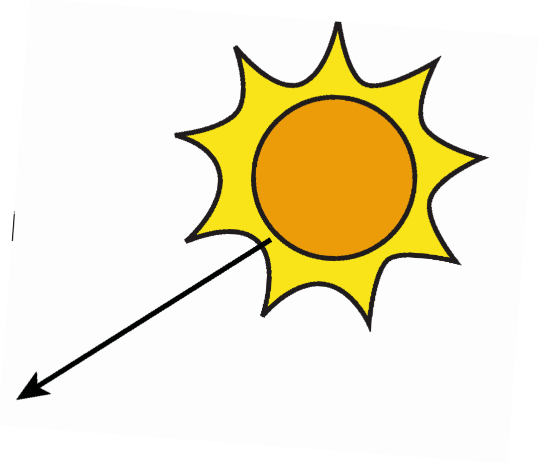
Surfaces, manifolds and meshes



CMU 15-462/662

Geometry
Ray-mesh intersection





- Think about a ray of light traveling from the sun
- Want to know where a ray pierces a surface
- Might pierce surface in many places
- A significant step towards visibility and ray tracing

Geometry

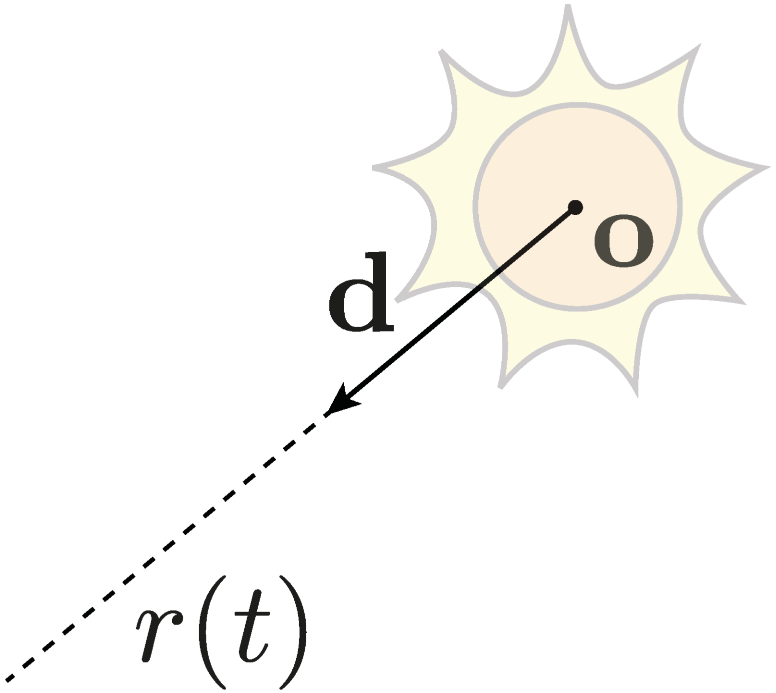
Ray equation

Ray source

Unit direction

$$r(t) = o + td$$

Point along ray parametrized by t



Geometry

Ray-plane intersection

Intersection between

plane $N^T x = c$

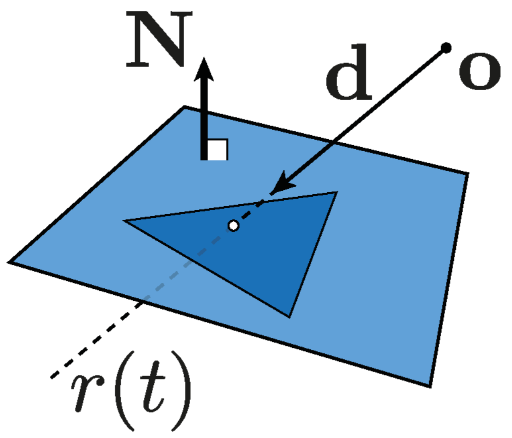
and

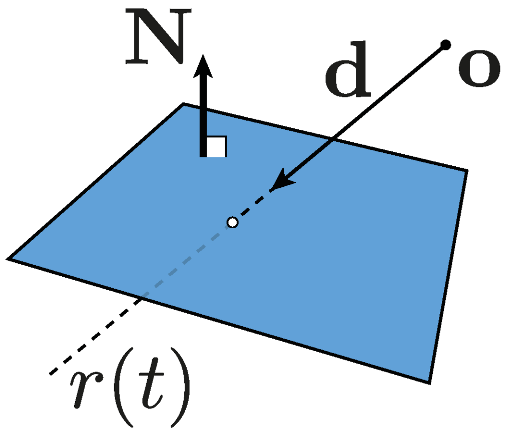
ray $r(t) = o + td$

$$N^T r(t) = c$$

$$N^T(o + td) = c \Rightarrow t = \frac{c - N^T o}{N^T d}$$

$$r(t) = o + \frac{c - N^T o}{N^T d} d$$



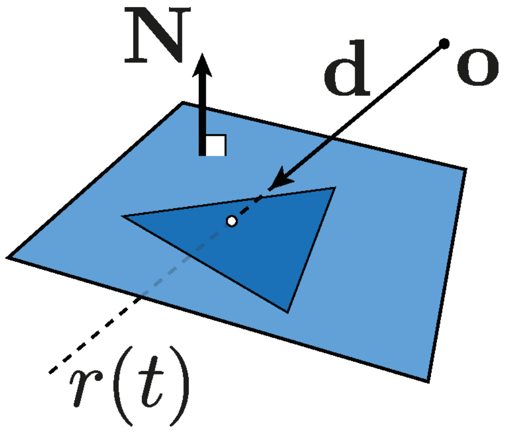


N unit normal

C offset

Geometry

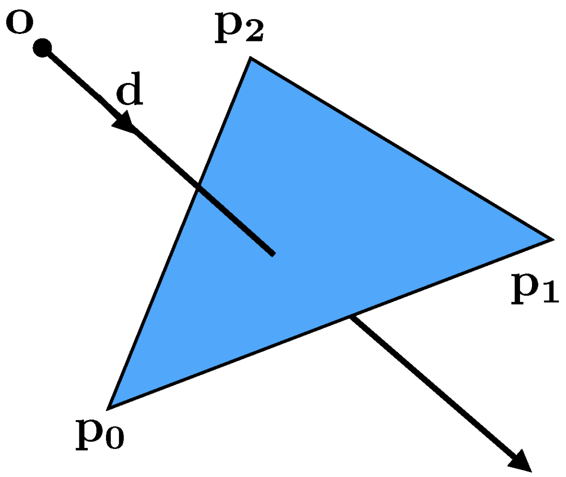
Ray-triangle intersection



- need to determine if point of intersection is within the triangle
- compute ray-plane intersection
- compute barycentric coordinates of hit point
- if all barycentric coordinates are positive, point in triangle

Geometry

Ray-triangle intersection

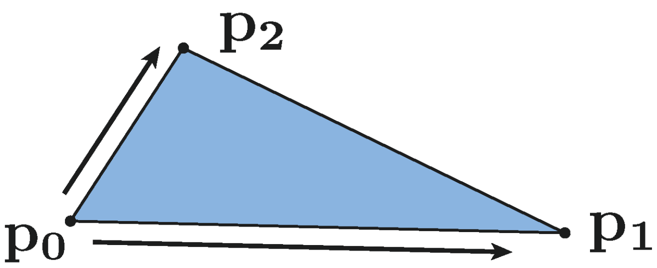


- Parametrize triangle given vertices $\boldsymbol{p_o}, \boldsymbol{p_1}, \boldsymbol{p_2}$ using barycentric coordinates

$$f(u, v) = (1 - u - v)\boldsymbol{p_0} + u\boldsymbol{p_1} + v\boldsymbol{p_2}$$

- Does a point $\boldsymbol{p}$ within the triangle?
- Solve for u, v

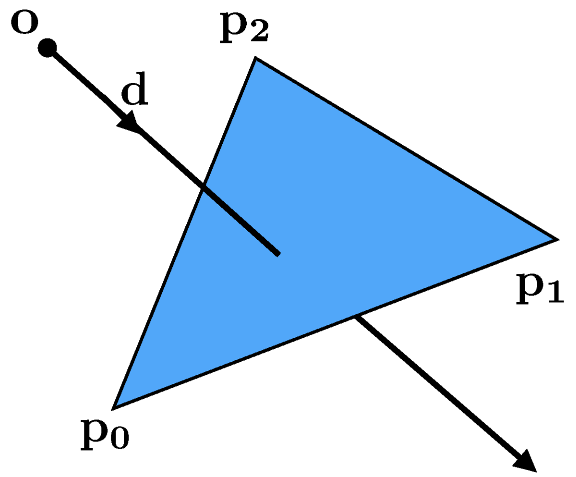
$$\boldsymbol{p} = \boldsymbol{p_0} + u(\boldsymbol{p_1} - \boldsymbol{p_0}) + v(\boldsymbol{p_2} - \boldsymbol{p_0})$$



- If $u, v, 1 - u - v \geq 0$ the point $\boldsymbol{p}$ is within the triangle

Geometry

Ray-triangle intersection



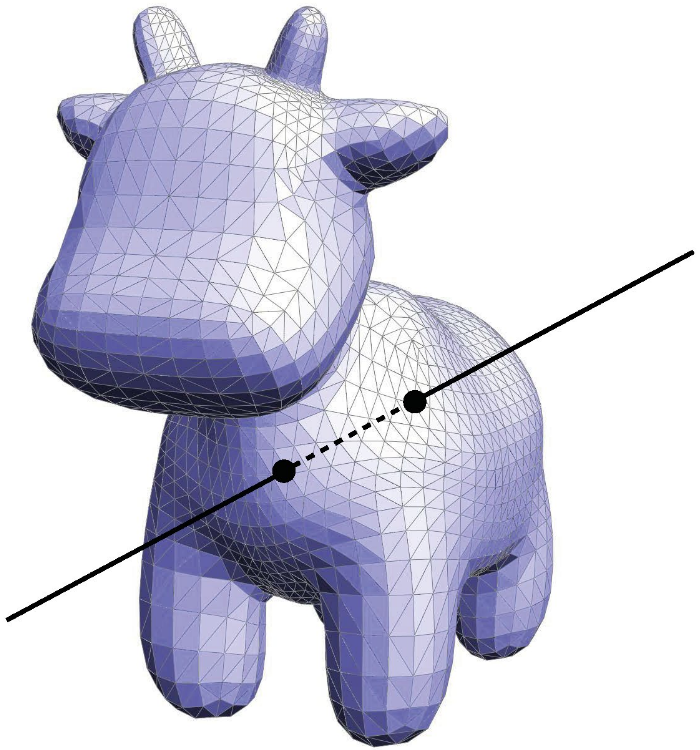
- Does a point $\boldsymbol{p} = \boldsymbol{o} + t\boldsymbol{d}$ within the triangle?
- Solve directly for u, v, t

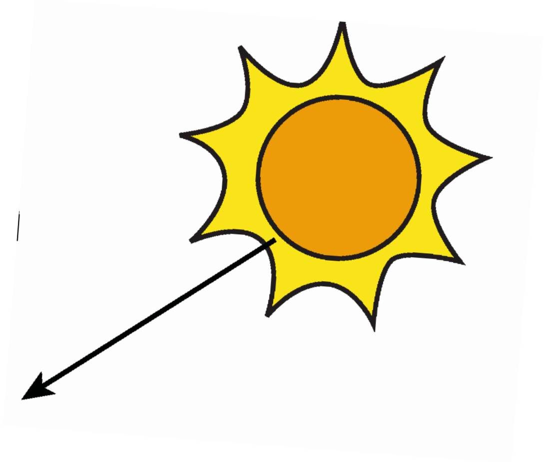
$$\boldsymbol{o} + t\boldsymbol{d} = \boldsymbol{p_0} + u(\boldsymbol{p_1} - \boldsymbol{p_0}) + v(\boldsymbol{p_2} - \boldsymbol{p_0})$$

$$[\boldsymbol{p_1} - \boldsymbol{p_0} \quad \boldsymbol{p_2} - \boldsymbol{p_0} \quad -\boldsymbol{d}] \begin{bmatrix} u \\ v \\ t \end{bmatrix} = \boldsymbol{o} - \boldsymbol{p_0}$$

Geometry

Ray-mesh intersection





Challenges in performance

- How to accelerate the naïve algorithm, given a ray, scan all triangles
- There are a lot of triangles and a lot of rays
- By hierarchical approach and dedicated hardware

Why care about performance?

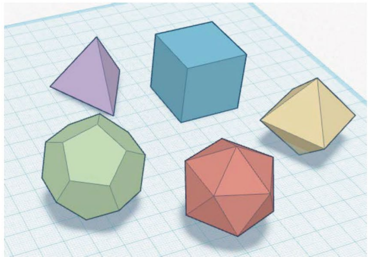
Pixar's "Coco" — about 50 <u>hours</u> per frame (@24 frames/sec)

Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

camera

geometry



materials

lights

("scene")

Rasterization

image

Radiometry = measuring light

<u>Aim: Photo realistic images</u>

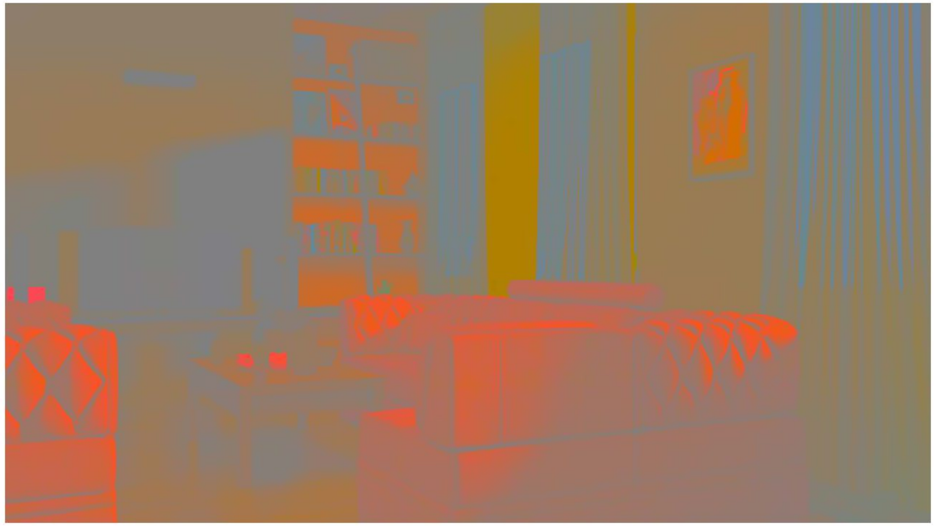
- Which color at each pixel?
- How much light (illumination) at each pixel?
- Why some parts of the surface look lighter or darker?
- Final image = at every point, what color and how intense or bright it is

[redacted]

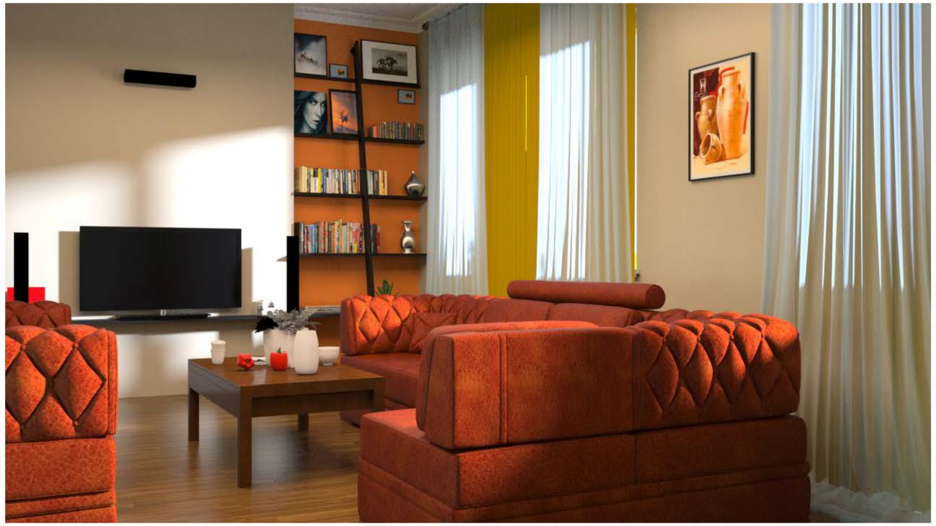
Rendering is more than just color!

- Also need to know ***how much*** light hits each pixel:

color



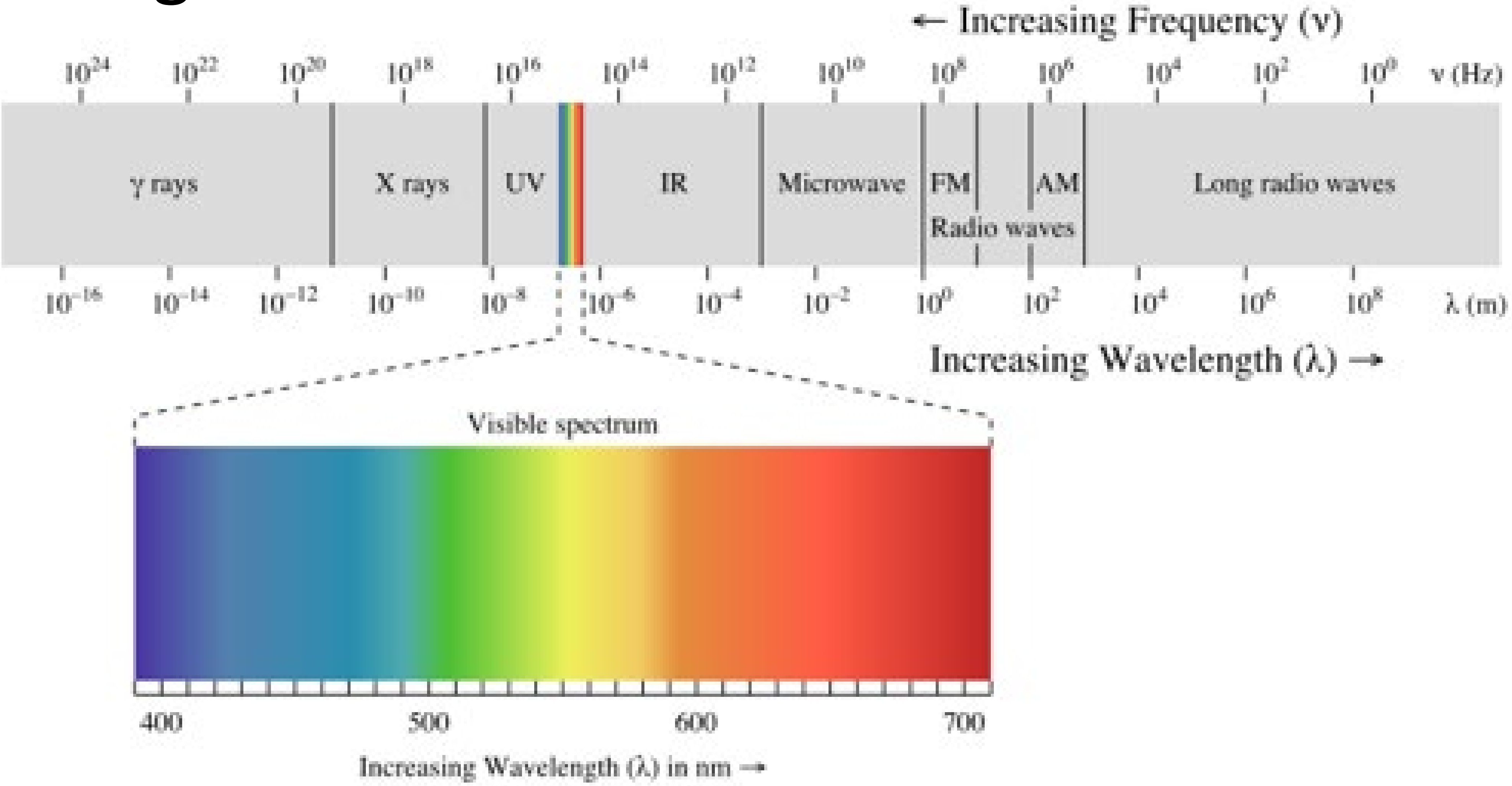
intensity



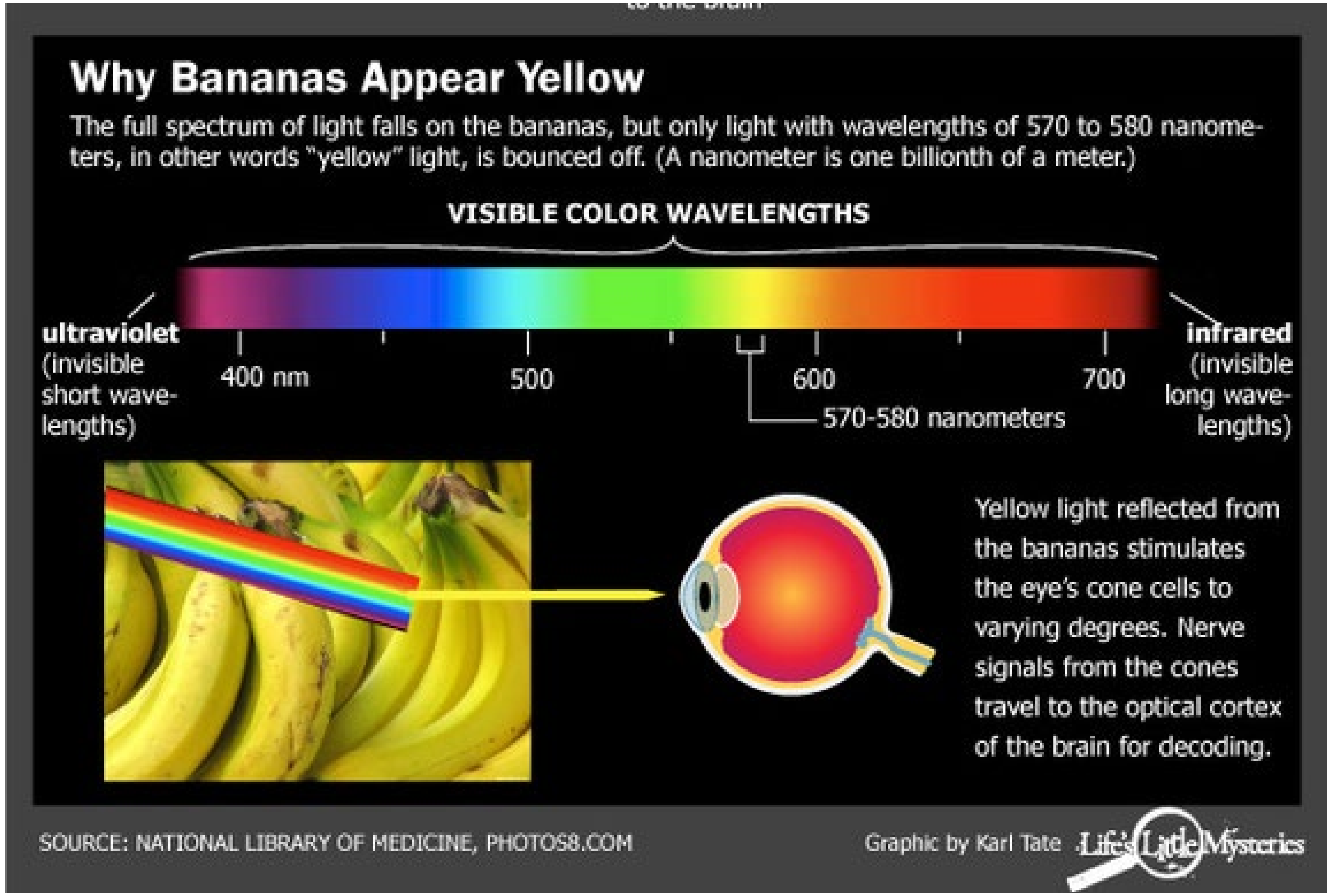
image

Radiometry

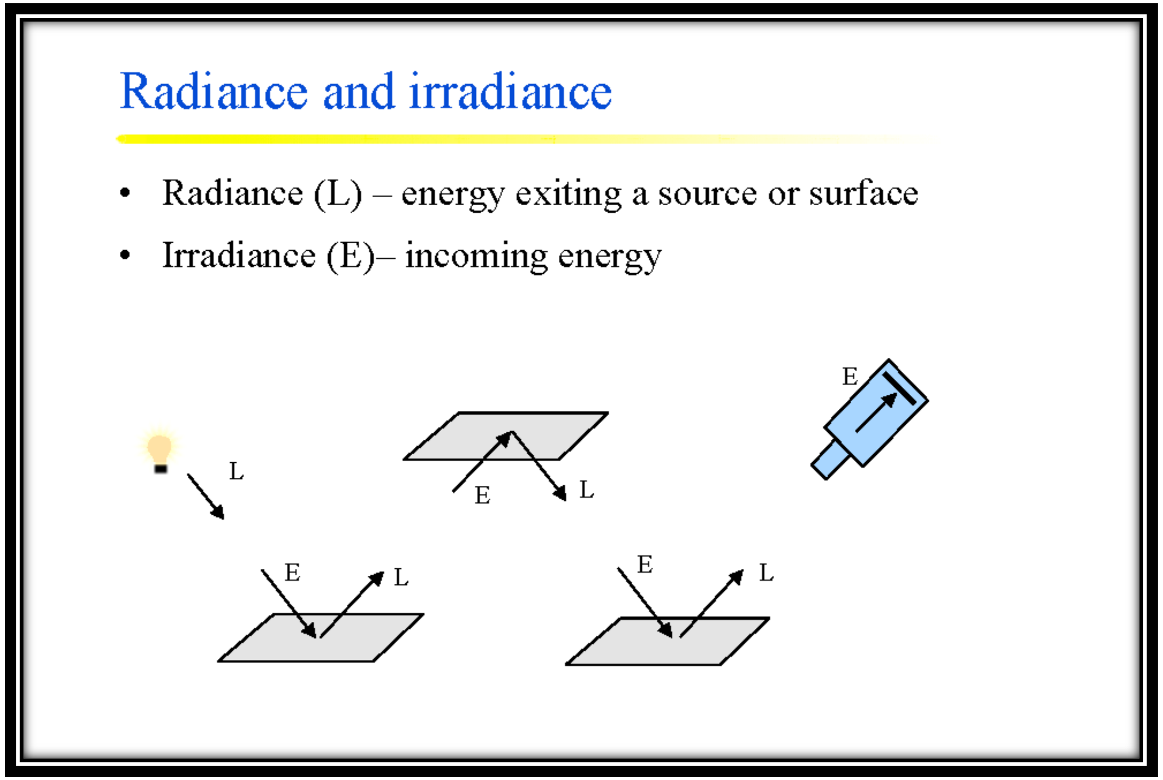
Electromagnetic radiation



Radiometry
Electromagnetic radiation



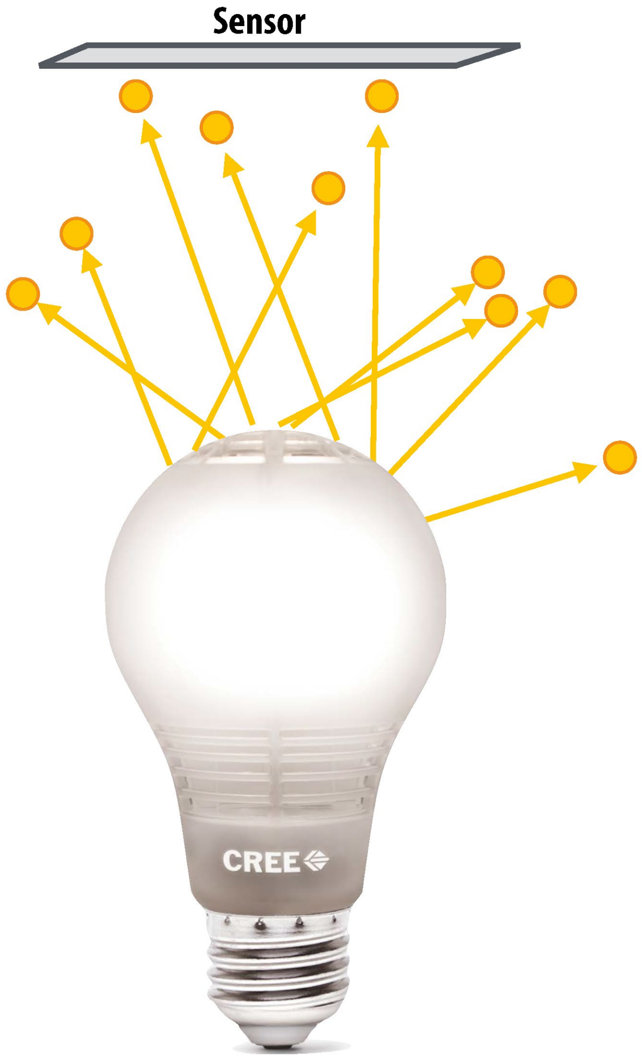
Radiometry
Radiance and irradiance



Radiometry
Radiant flux

Radiant flux: energy per unit time [Watts] received by the sensor (or emitted by the light)

Φ = Time density of energy [Watts]



Radiometry
Irradiance

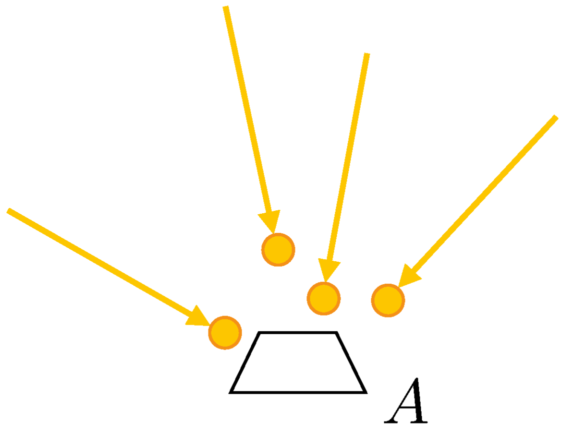
Irradiance: area density of radiant flux

Given a sensor with area A, we consider the average flux over the entire sensor area

$$\frac{\Phi}{A}$$

Irradiance (E) = flux density, i.e., the incident flux per unit surface area

$$[\frac{\text{Watts}}{\text{m}^2}]$$



Given what we now know about radiant energy…

Why do some parts of a surface look lighter or darker?

Radiometry

Lambert's law

Consider beam with flux Φ incident on surface with area A

irradiance (energy per time, per area)

radiant flux (energy per time)

$$E = \frac{\Phi}{A}$$

$$\Phi = EA$$

Radiometry
Lambert’s law

Consider beam with flux Φ incident on tilted surface with area A'

$$A = A' \cos\theta$$

A = projected area of surface relative to direction of beam

Radiometry
Lambert's law

Irradiance at surface is proportional to cosine of angle between light direction and surface normal

$$A = A' \cos\theta$$

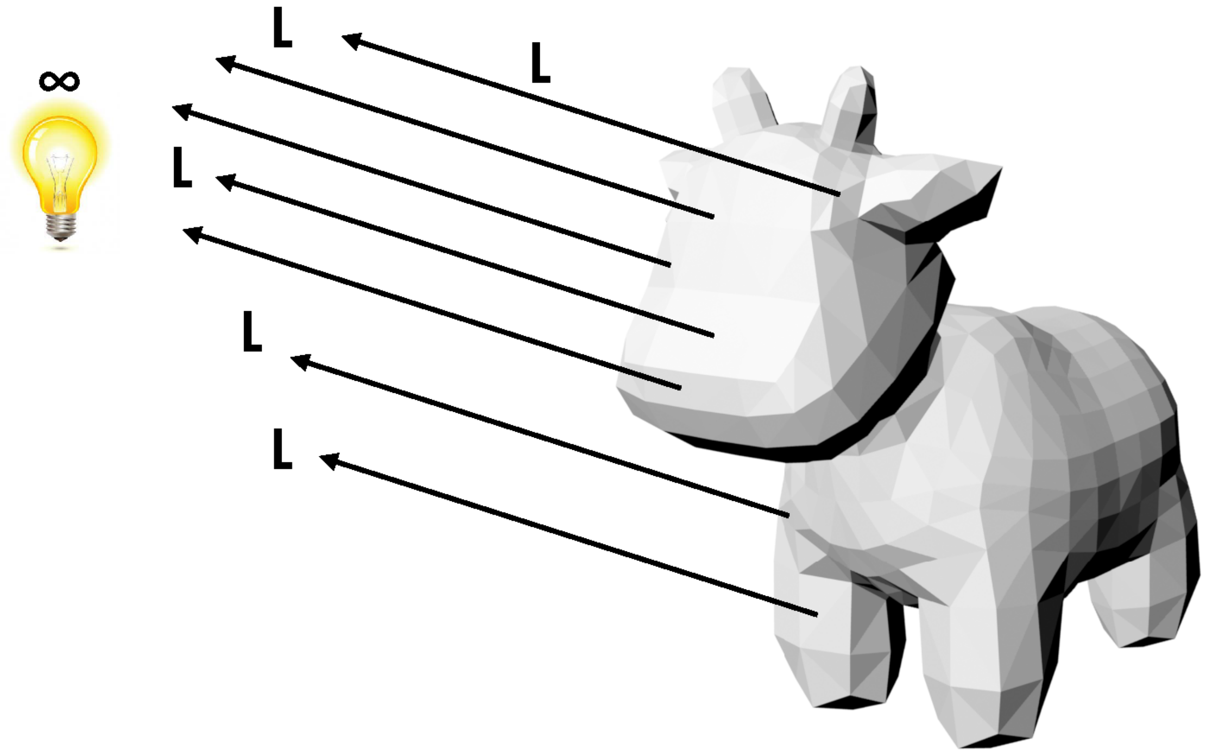
$$E = \frac{\Phi}{A'} = \frac{\Phi \cos\theta}{A}$$

Radiometry

Types of light

directional lighting

- Common abstraction: infinitely bright light source "at infinity"
- All light directions (L) are therefore identical

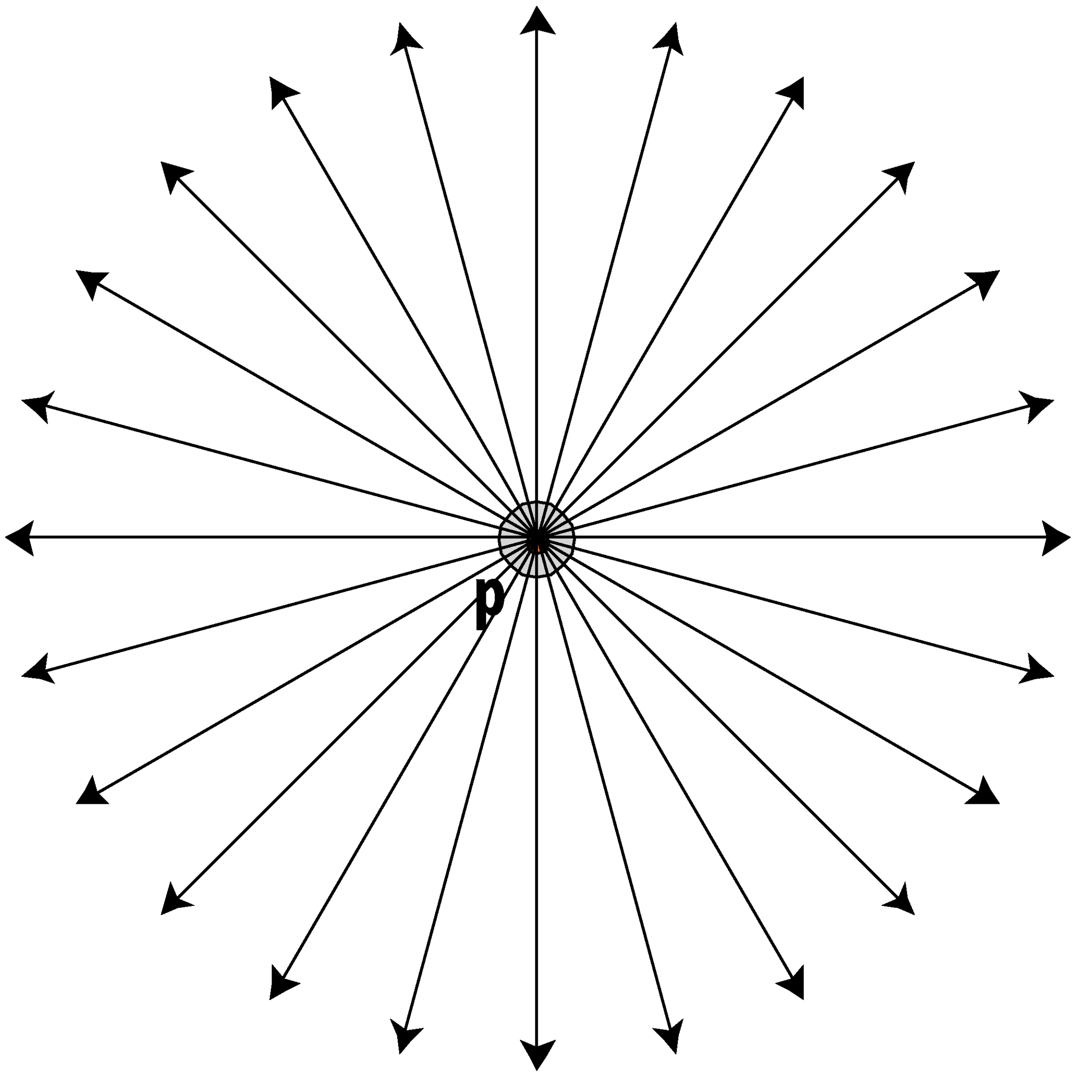


- Infinitely bright light source at infinity
- All light direction (L) are identical

Radiometry

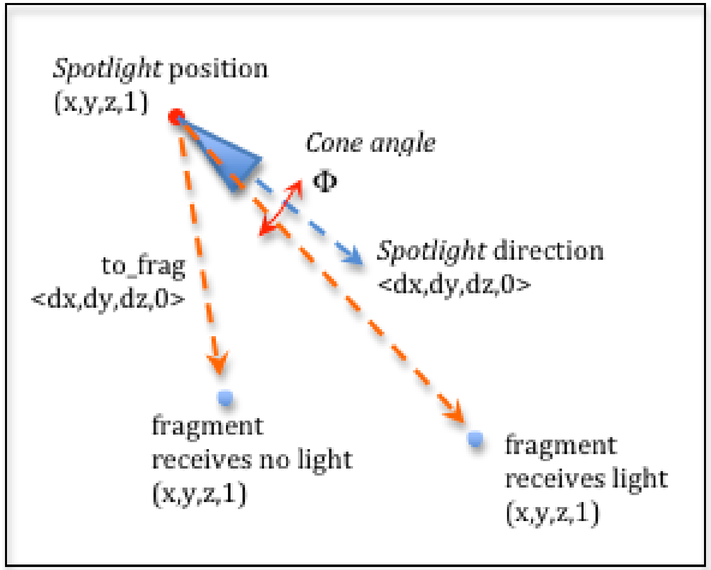
Types of light

Isotropic point source

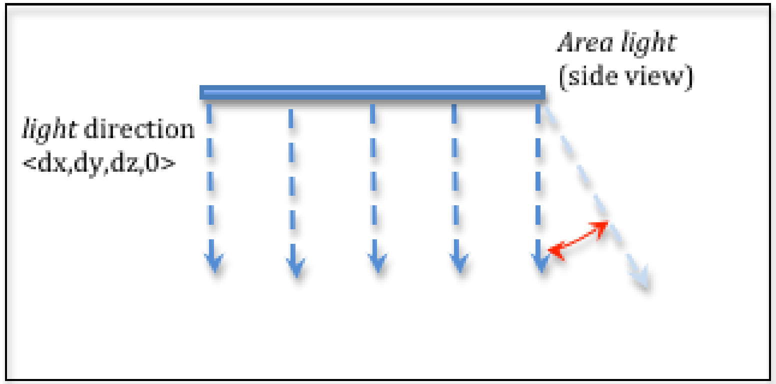


Radiometry
Types of light

Spotlight source

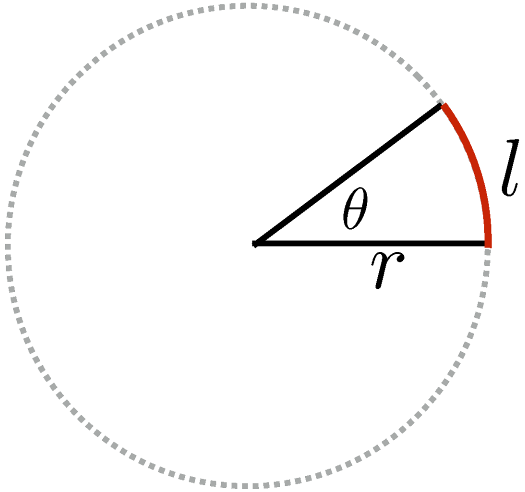


Area light source

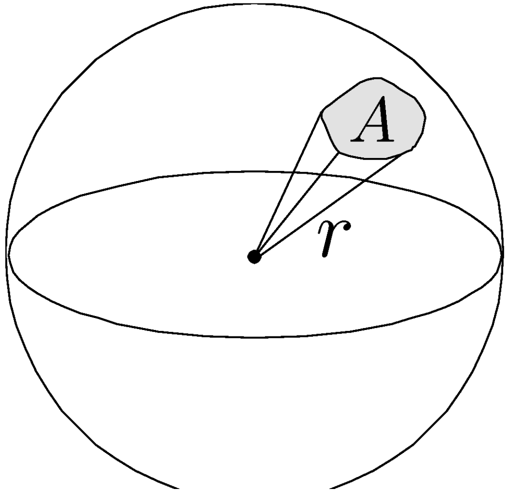


Radiometry
Solid angle

- **Angle: ratio of subtended arc length on circle to radius**
  - $\theta = \frac{l}{r}$
  - **Circle has 2π radians**

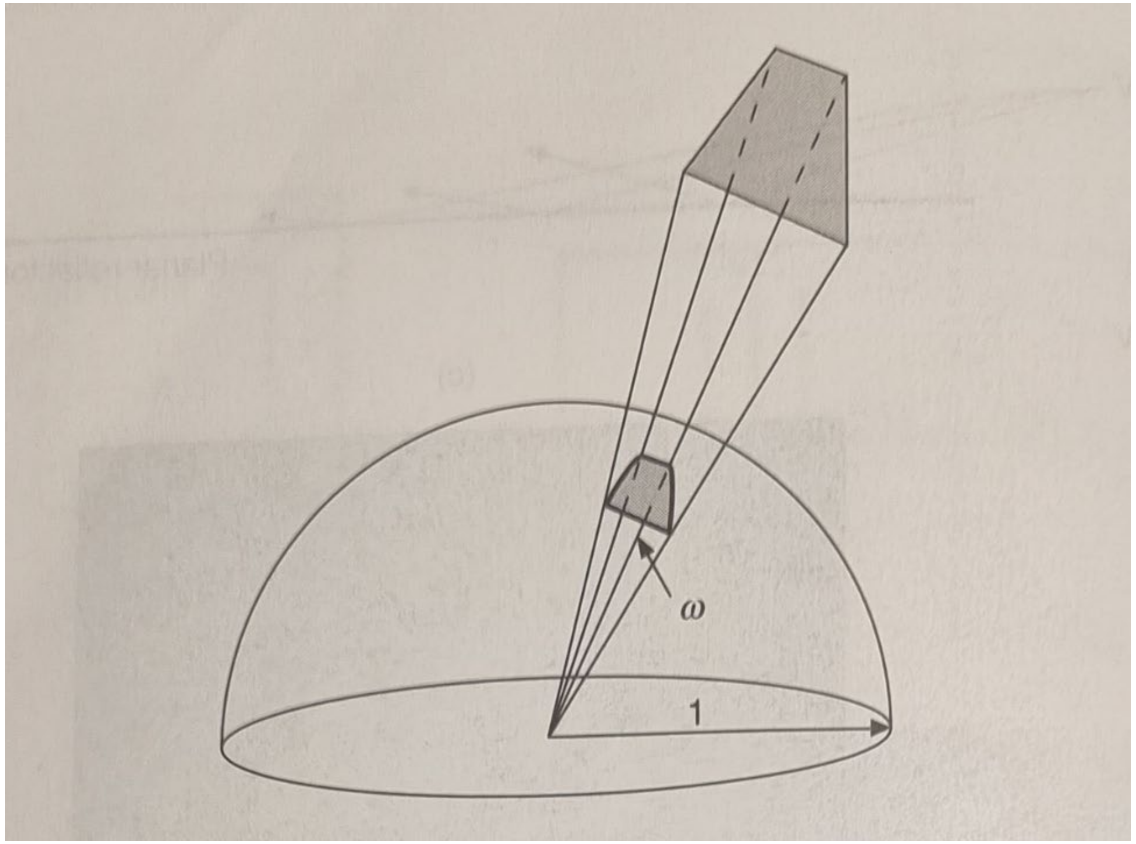


- **Solid angle: ratio of subtended area on sphere to radius squared**
  - $\Omega = \frac{A}{r^2}$
  - **Sphere has 4π steradians**



Radiometry
Solid angle

The solid angle subtended by an object from a point on a surface =
The area covered by the object's projection onto a unit hemisphere above the point



Radiometry
Radiance

Radiance is the solid angle density of irradiance

$$L(\boldsymbol{p}, w)$$

Radiance is energy along a ray defined by origin point $\boldsymbol{p}$ and direction w

Radiant energy per unit time per unit area per unit solid angle

$$[\frac{\mathrm{W}}{\mathrm{m}^2\ \mathrm{sr}}]$$

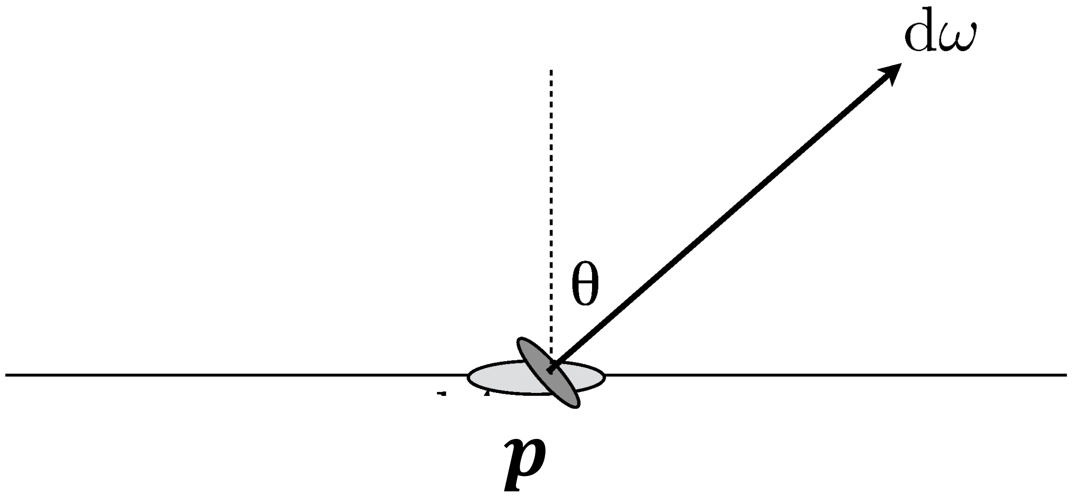
Radiometry

Radiance

A surface experiencing radiance $L(p, w)$, coming in from solid angle dw experiences irradiance

$$dE(\boldsymbol{p}) = \quad L(\boldsymbol{p}, w) \quad \cos(\theta) \quad dw$$

Radiance Lambert's law Solid angle



Radiometry

Radiance properties

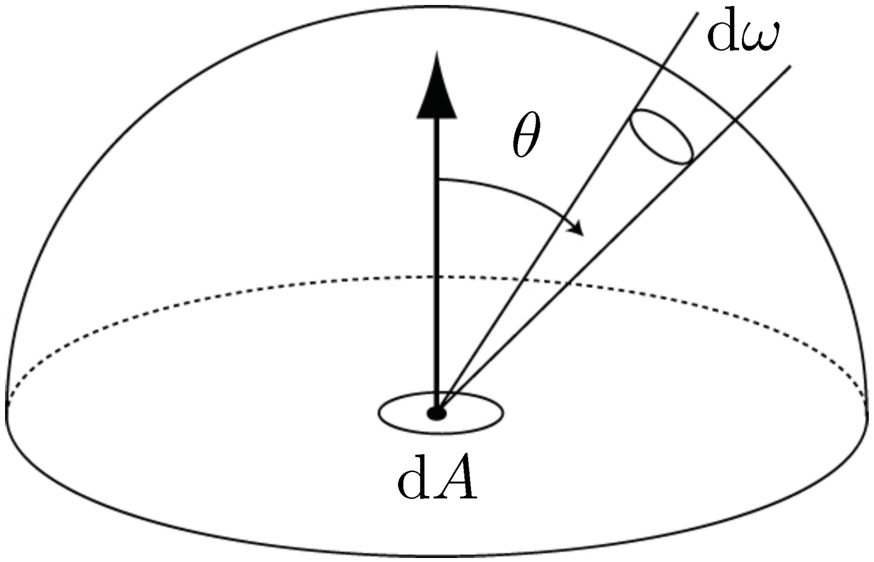
- Radiance is a fundamental quantity that characterizes the distribution of light an environment
- Radiance is the quantity associated with a ray (constant a long a ray)
- Rendering is all about computing radiance
- A pinhole camera measures radiance

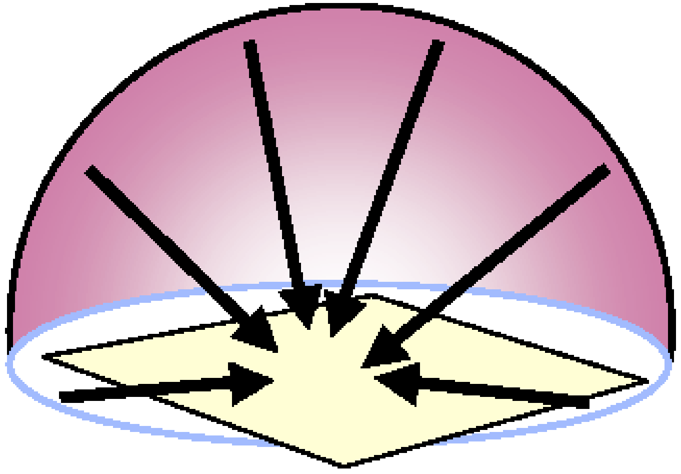
[redacted]

Radiometry
Irradiance from the environment

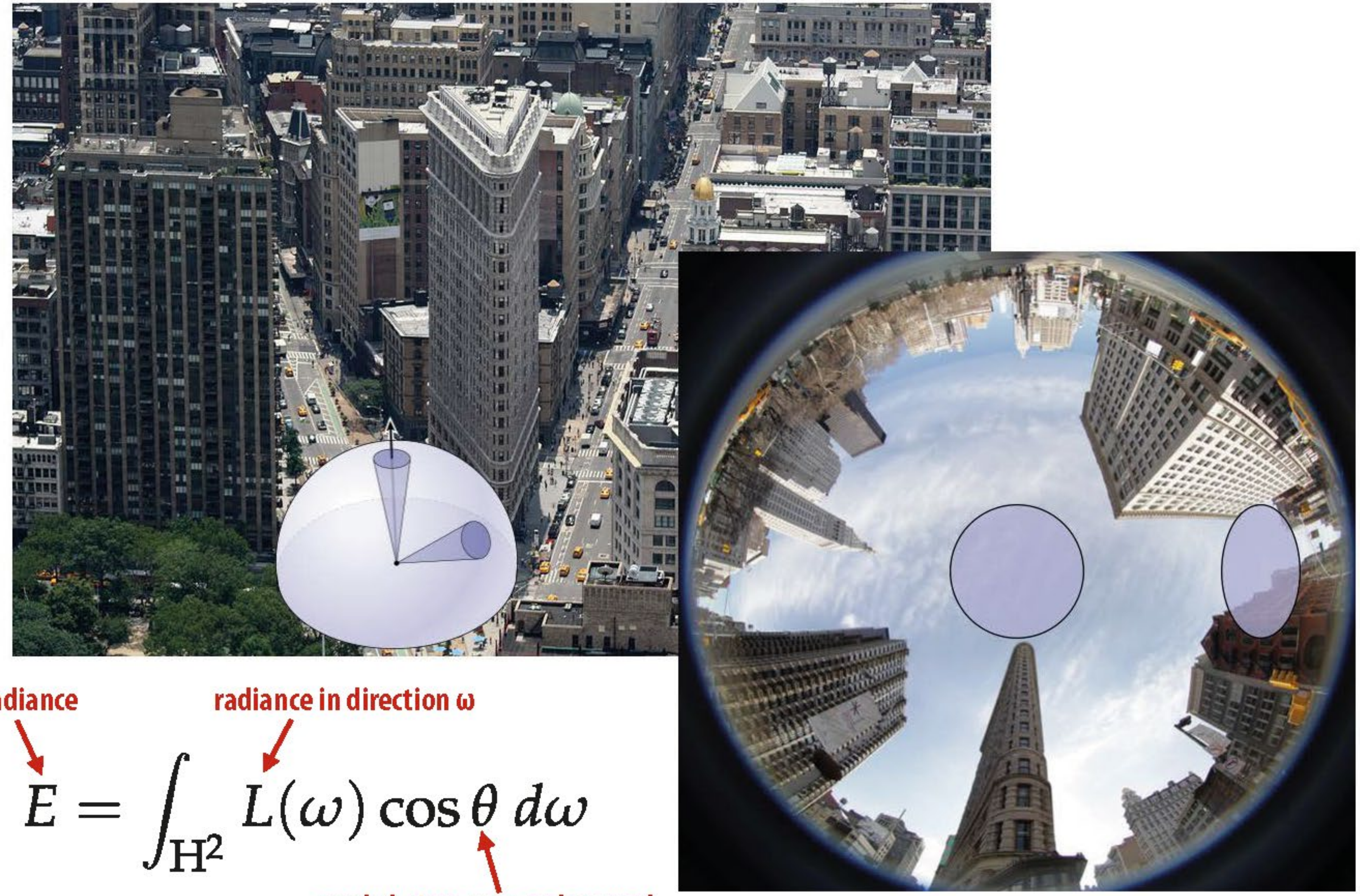
Computing flux per unit area on surface, due to incoming light from all directions

$$E(\mathsf{p}) = \int_{H^2} L_i(\mathsf{p}, \omega) \cos\theta d\omega$$





Recap: Radiance and Irradiance



irradiance

radiance in direction ω

$$E = \int_{H^2} L(\omega) \cos\theta \, d\omega$$

angle between ω and normal

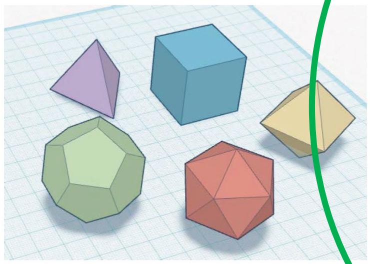
Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

[redacted]

camera

geometry



materials

lights

("scene")

Rasterization

image

Radiometry
Bidirectional reflectance distribution function

- When light hits a surface, the way it is reflected (scattered off the surface), depends on the surface material properties

- This is encoded by the "Bidirectional reflectance distribution function" (BRDF)

- Given incoming direction w_i, how much light gets scattered in any given outgoing direction w_o?

- The BRDF tells us how bright a surface appears when viewed from one direction while light falls from another one

- Local illumination model (direct light), light directly from light sources to surfaces

Radiometry

Reflected radiance and incident irradiance

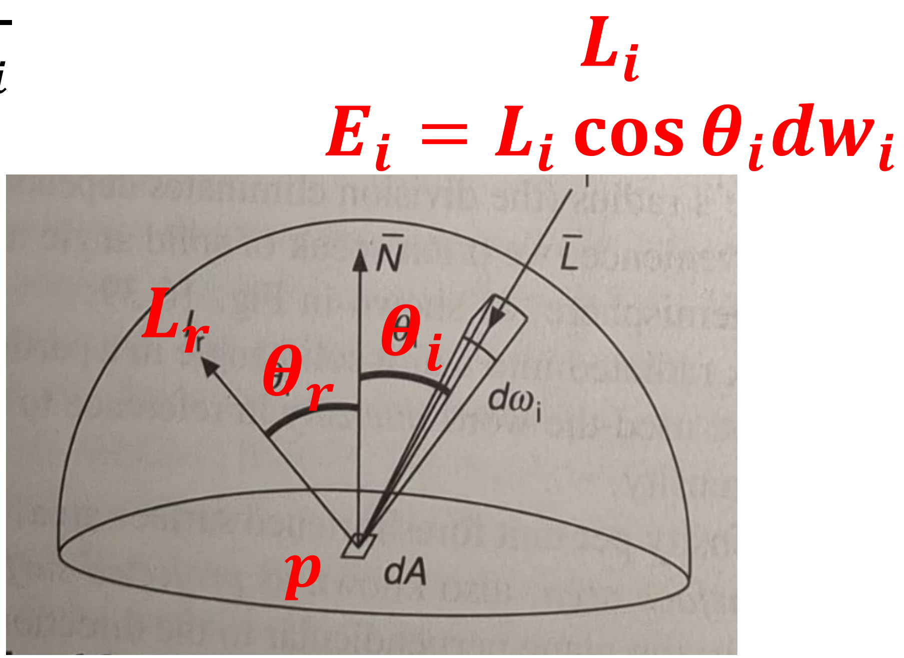
The incident radiance L_i

The incident irradiance $E_i = L_i \cos\theta_i dw_i$

The reflected radiance L_r

$$\text{BRDF} = f(p, w_i, w_r) = \frac{reflected\ energy}{incident\ energy} = \frac{L_r}{E_i}$$

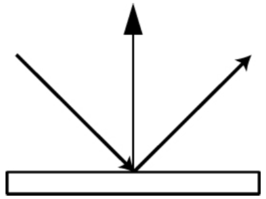
$$[\frac{1}{\text{sr}}]$$



Some basic reflection functions

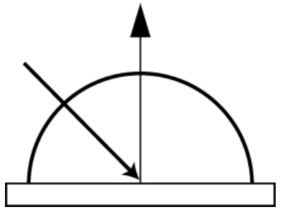
- **Ideal specular**

  Perfect mirror



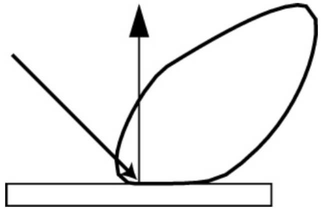
- **Ideal diffuse**

  Uniform reflection in all directions



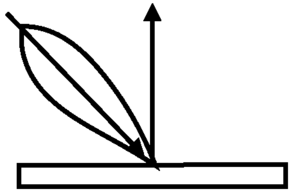

- **Glossy specular**

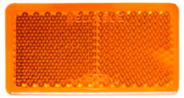
  Majority of light distributed in reflection direction



- **Retro-reflective**

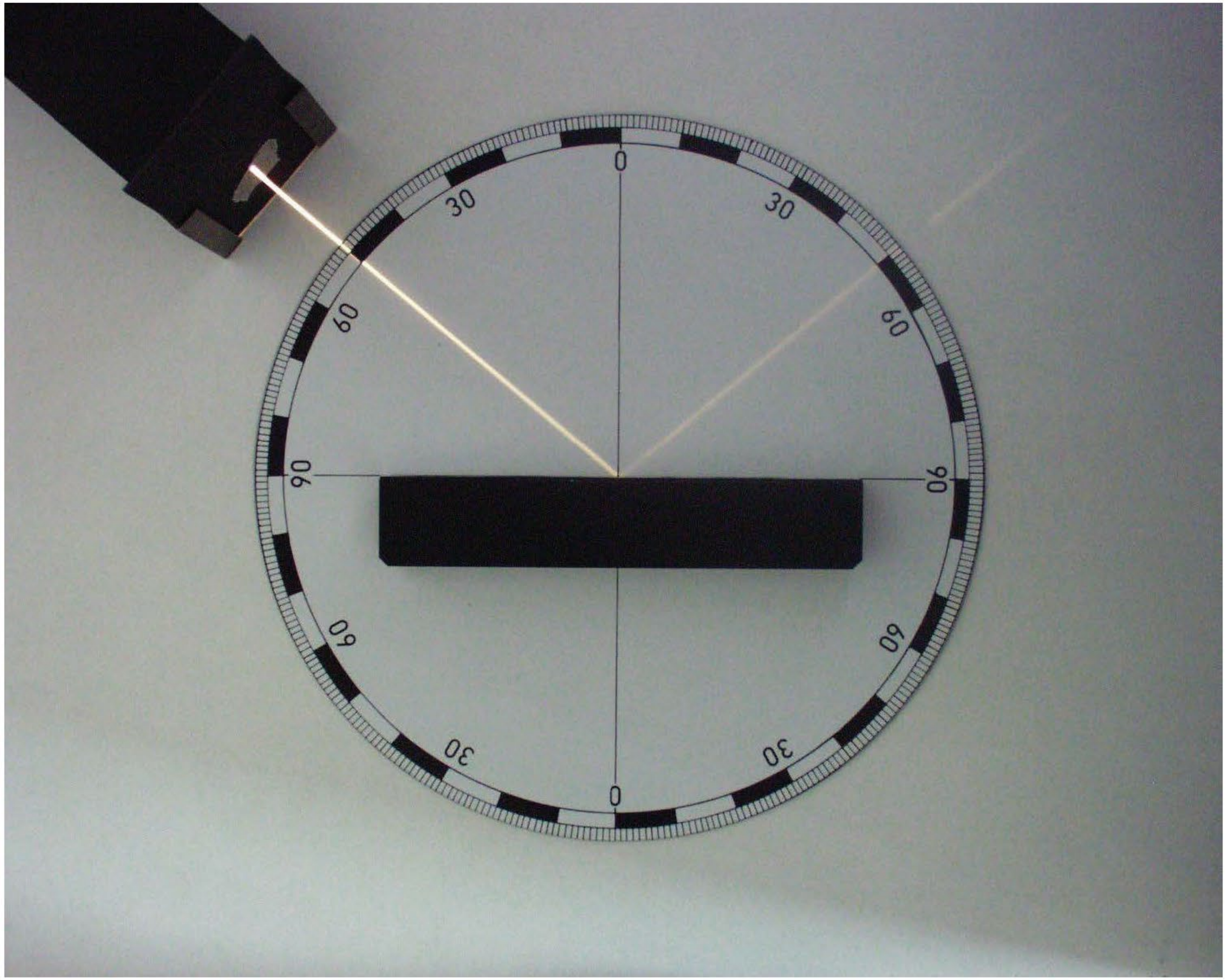
  Reflects light back toward source



Diagrams illustrate how incoming light energy from given direction is reflected in various directions.

Example: perfect specular reflection



[Zátonyi Sándor]

Materials: diffuse

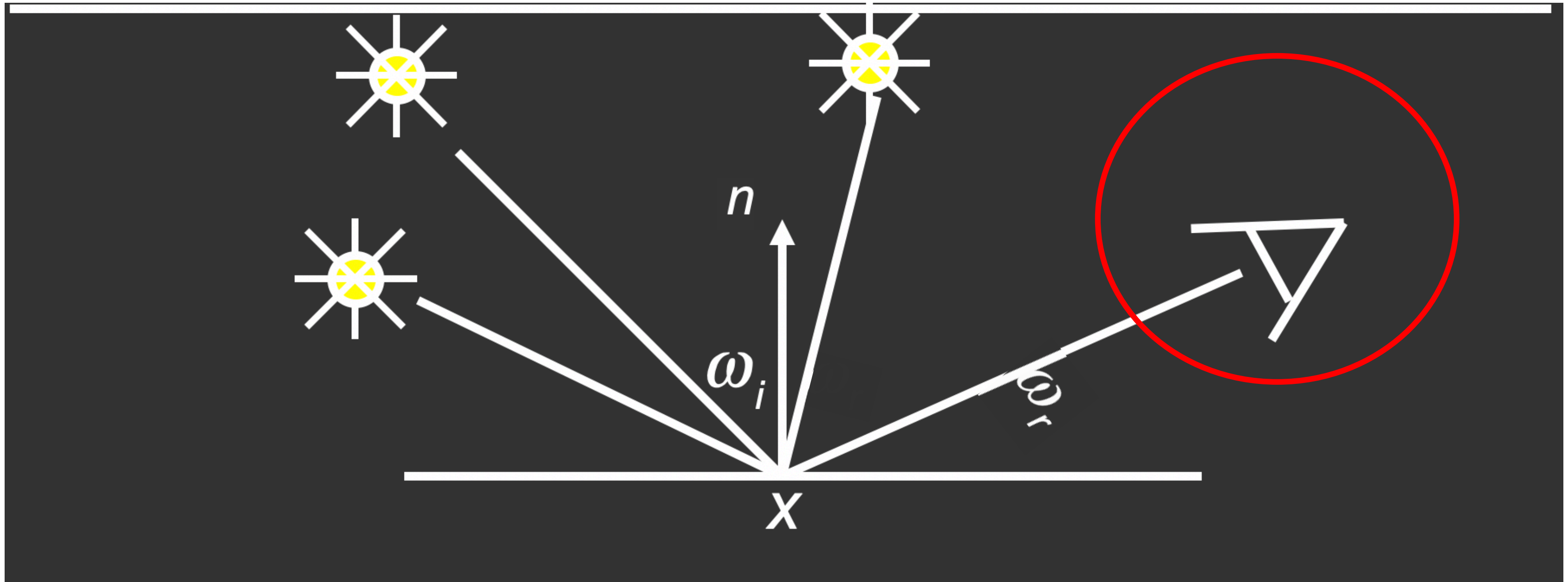
Materials: plastic

Materials: red semi-gloss paint

Materials: mirror

Reflection equation

Multiple light sources



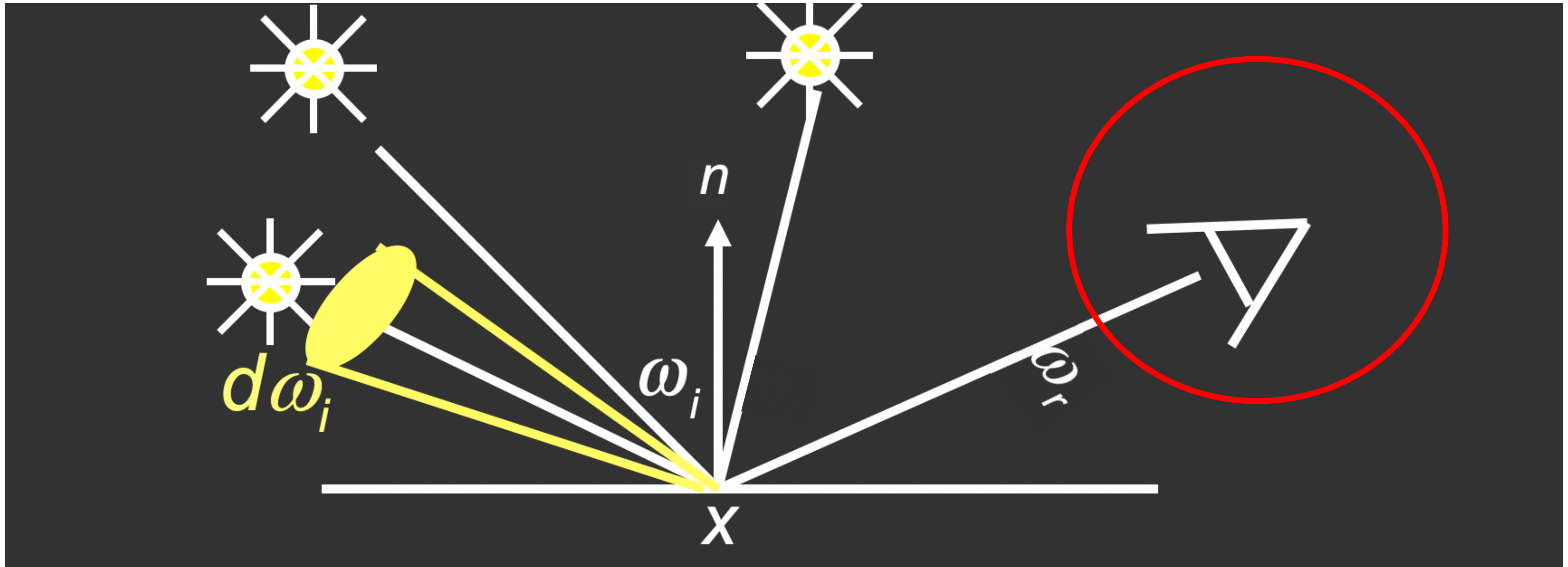
Sum over all light sources

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \sum L_i(x,\omega_i) f(x,\omega_i,\omega_r)(\omega_i \bullet n)$$

Reflected Light (Output Image) | Emission | Incident Light (from light source) | BRDF | Cosine of Incident angle

Reflection equation

Environment of light sources



Replace sum with integral

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i) f(x,\omega_i,\omega_r)\cos\theta_i \, d\omega_i$$

Reflected Light (Output Image)	Emission	Incident Light (from light source)	BRDF	Cosine of Incident angle

Reflection equation (local illumination)
Recap

- The image of a three dimensional object depends on its shape, its reflectance properties, and the distribution of the light sources

- The interactions of light with scene surfaces depend on the material properties of the surfaces. Materials may be represented as bidirectional reflectance distribution functions (BRDF)

- The BRDF leads to the reflection equation

- The reflection equation considers only local illumination (direct light), i.e., light directly from light sources to surfaces

Rendering equation (global illumination)

- Core functionality of photorealistic renderer is to estimate radiance at a given point, in a given direction

- To get photorealism we need to consider global illumination, multiple bounces (indirect light), called interreflections.

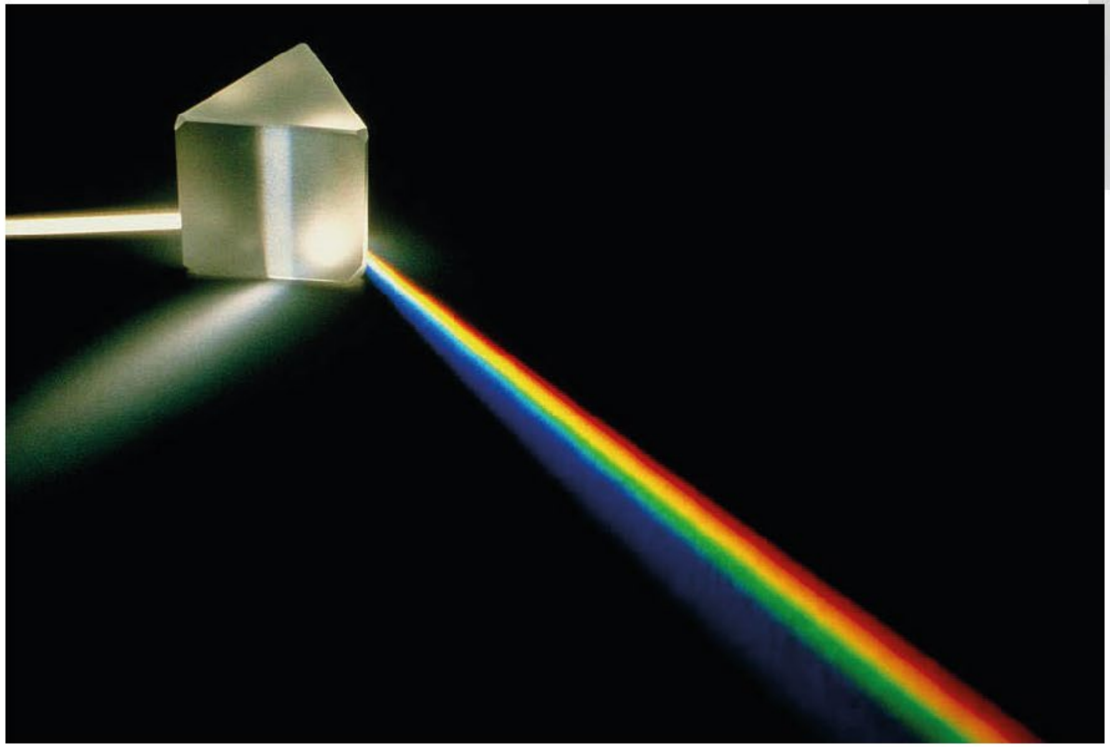
- In real scenarios, light reflected from an object strikes other objects in the surrounding area, illuminating them

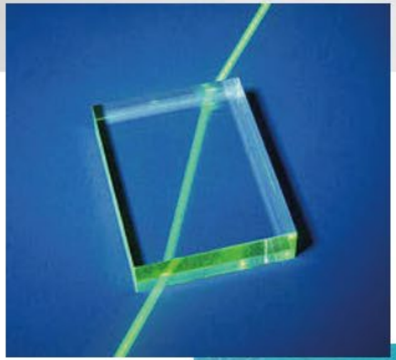
- When light energy hits a surface, several things can happen, depending on the surface properties
  - Reflection and interreflections
  - Refraction
  - Absorption

Transmission

In addition to reflecting off surface, light may be transmitted through surface.

Light refracts when it enters a new medium.





Rendering equation (global illumination)

Principles, James Kajiya, 1986

- For a given indoor scene, every object in the room must contribute illumination to every other object
- There is no distinction to be made between illumination emitted from a light source and illumination reflected from a surface
- The illumination coming from surfaces must scatter in a particular direction that is some function of the incoming direction of the arriving illumination, and the outgoing direction being sampled

Rendering equation (global illumination)

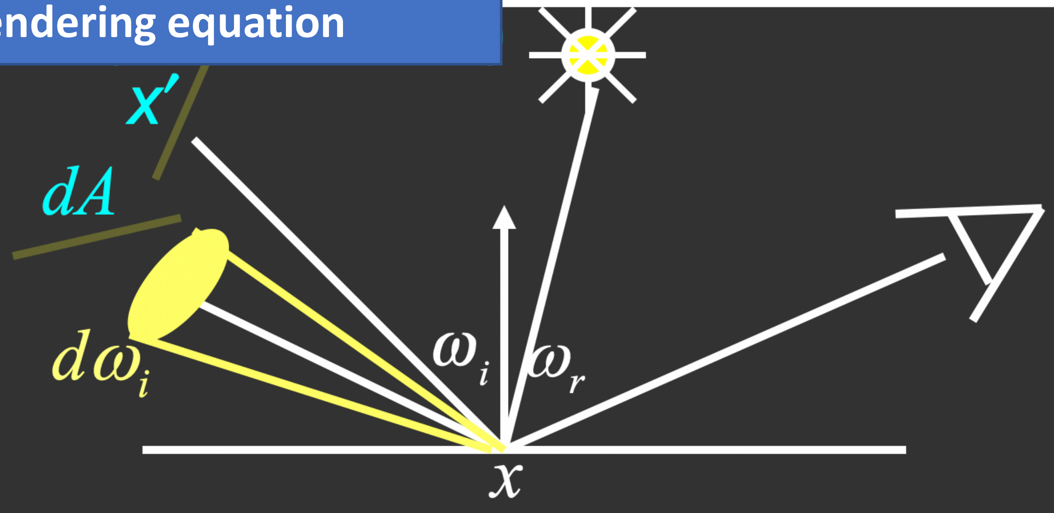
Challenge

- Computing reflection equation requires knowing the incoming radiance from surfaces
- But determining incoming radiance requires knowing reflected radiance from surfaces
- So we have to compute another integral, we have exactly the same equation
- Rendering equation is recursive

Reflection equation

$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_i(x,\omega_i) f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

Rendering equation



$$L_r(x,\omega_r) = L_e(x,\omega_r) + \int_{\Omega} L_r(x',-\omega_i) f(x,\omega_i,\omega_r) \cos\theta_i d\omega_i$$

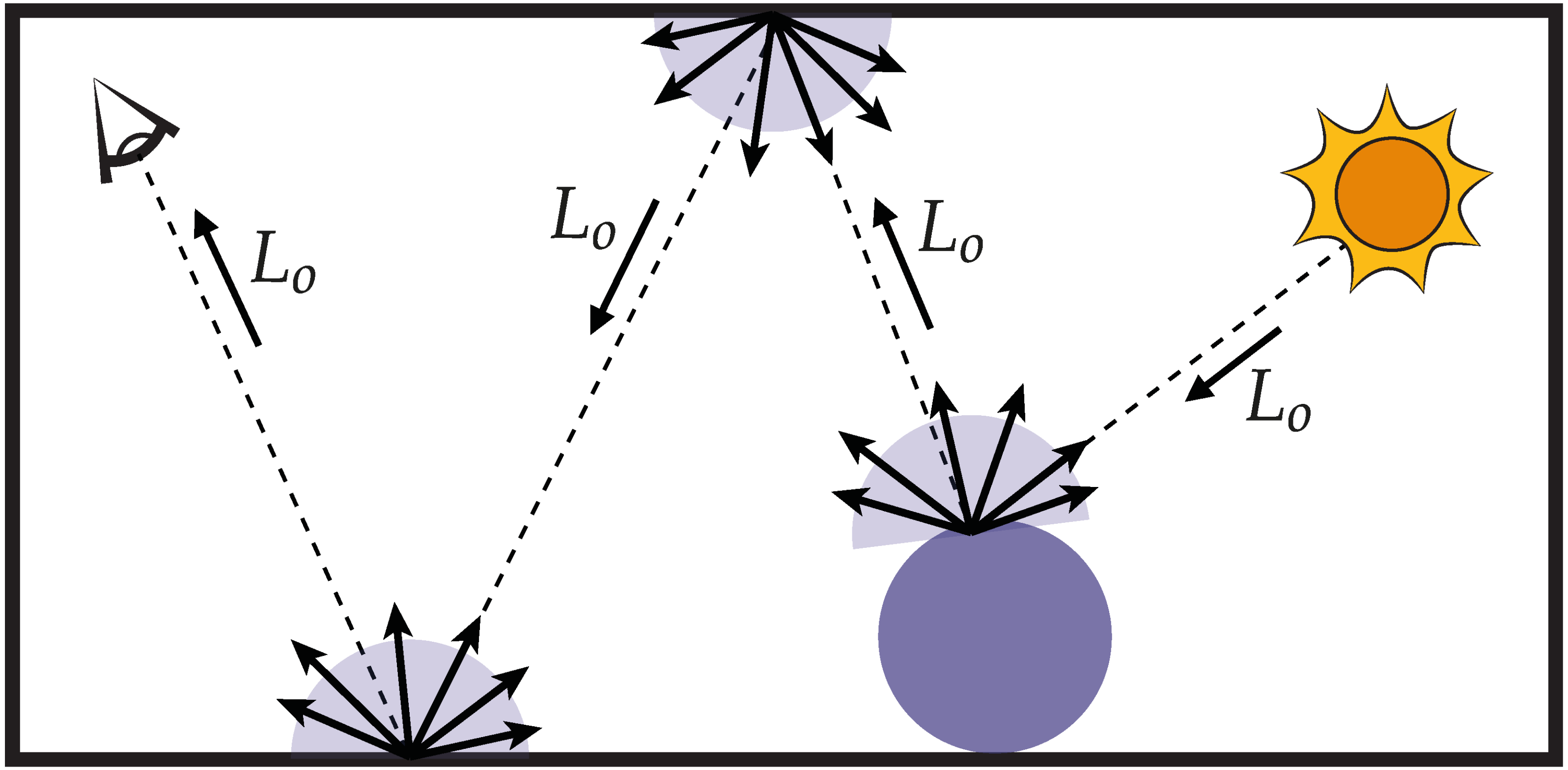
Reflected Light (Output Image)	Emission	Reflected Light	BRDF	Cosine of Incident angle
UNKNOWN	KNOWN	UNKNOWN	KNOWN	KNOWN

Slide by Lior Yariv

[redacted]

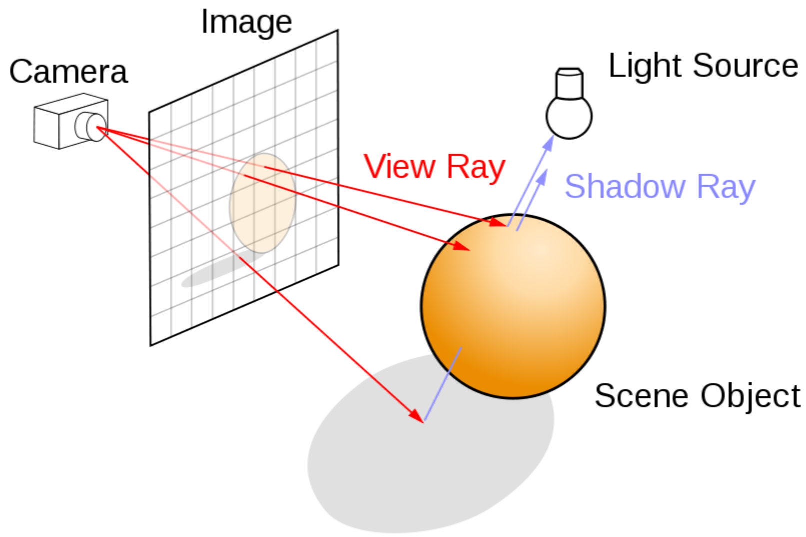
Recursive Raytracing

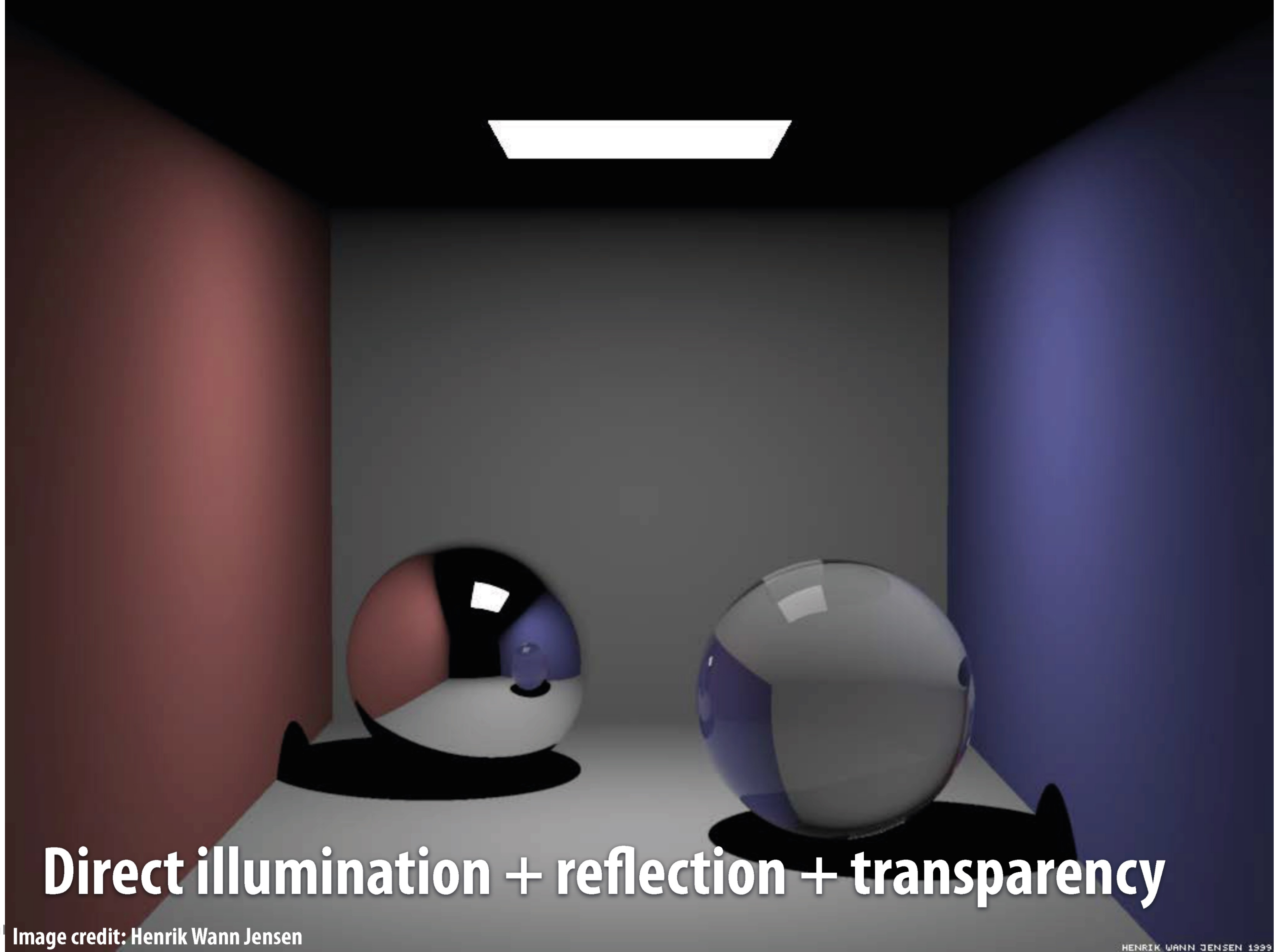
- **Basic strategy: recursively evaluate rendering equation!**



Rendering equation
How to solve?

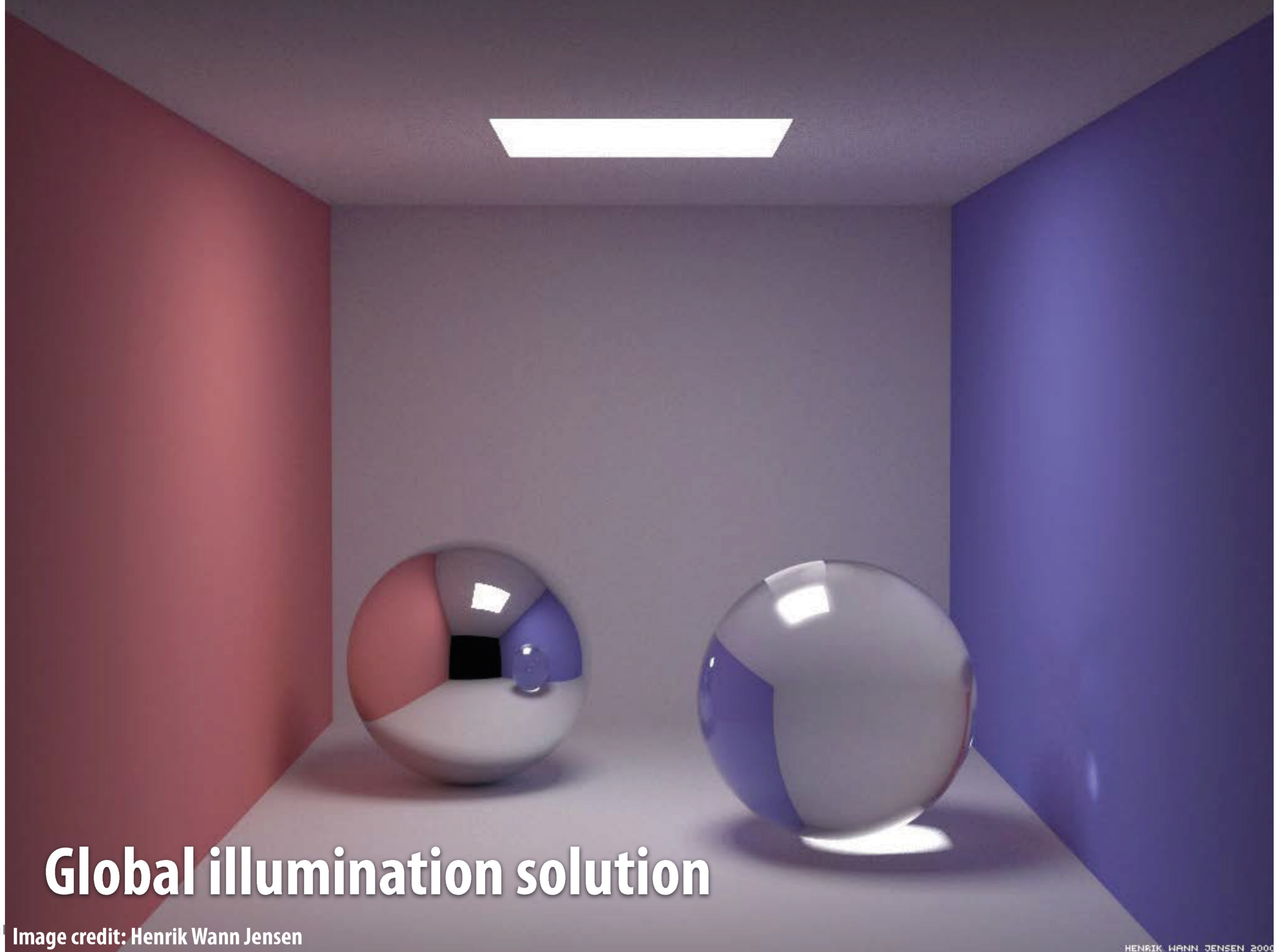
- Too hard for analytic solution
- Very challenging to apply directly recursive ray tracing
- Monte-Carlo rendering
- Ray tracing is crucial here
- Little control in rasterization, which rays we evaluate?





Direct illumination + reflection + transparency

Image credit: Henrik Wann Jensen



Global illumination solution

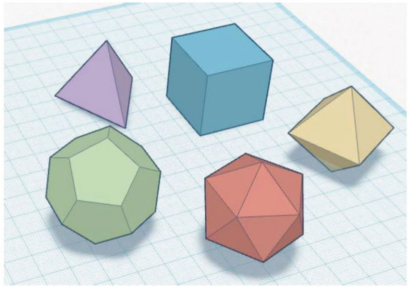
Image credit: Henrik Wann Jensen

Photorealistic Rendering—Basic Goal

What are the INPUTS and OUTPUTS?

camera

geometry



materials

lights

(“scene”)

Ray Tracer

image

Monte-Carlo rendering

How do we render a photorealistic image?
Combine
- Color
- Material
- Radiometry
- Ray tracing
- Rendering equation

into Monte-Carlo ray tracing algorithm

Monte-Carlo rendering
Integration

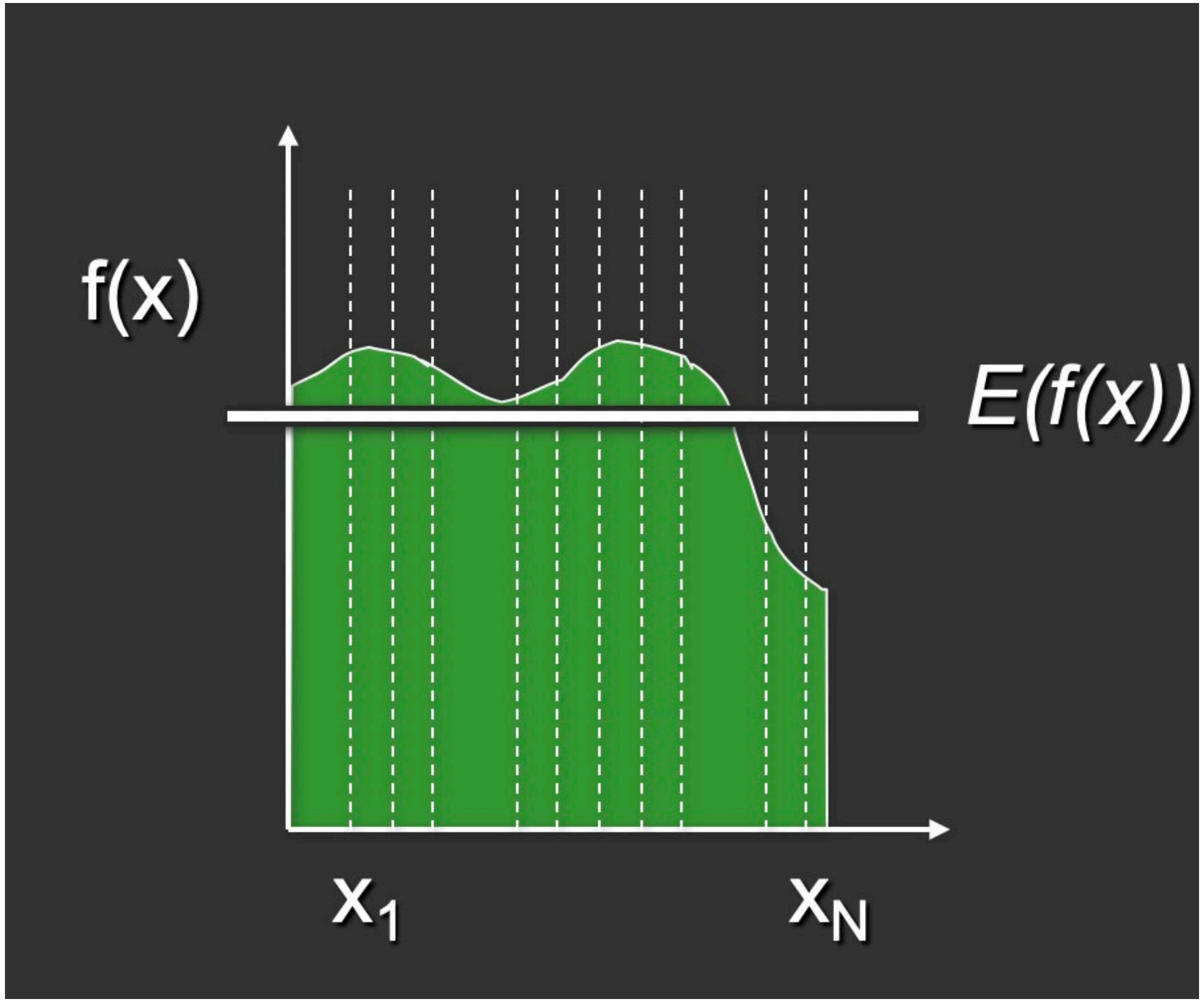
We want to estimate the integral

$$\int_0^1 f(x)dx = ?$$

Average (uniform sampling)

$$\int_0^1 f(x)dx = \frac{1}{N}\sum_{i=1}^{N} f(x_i)$$

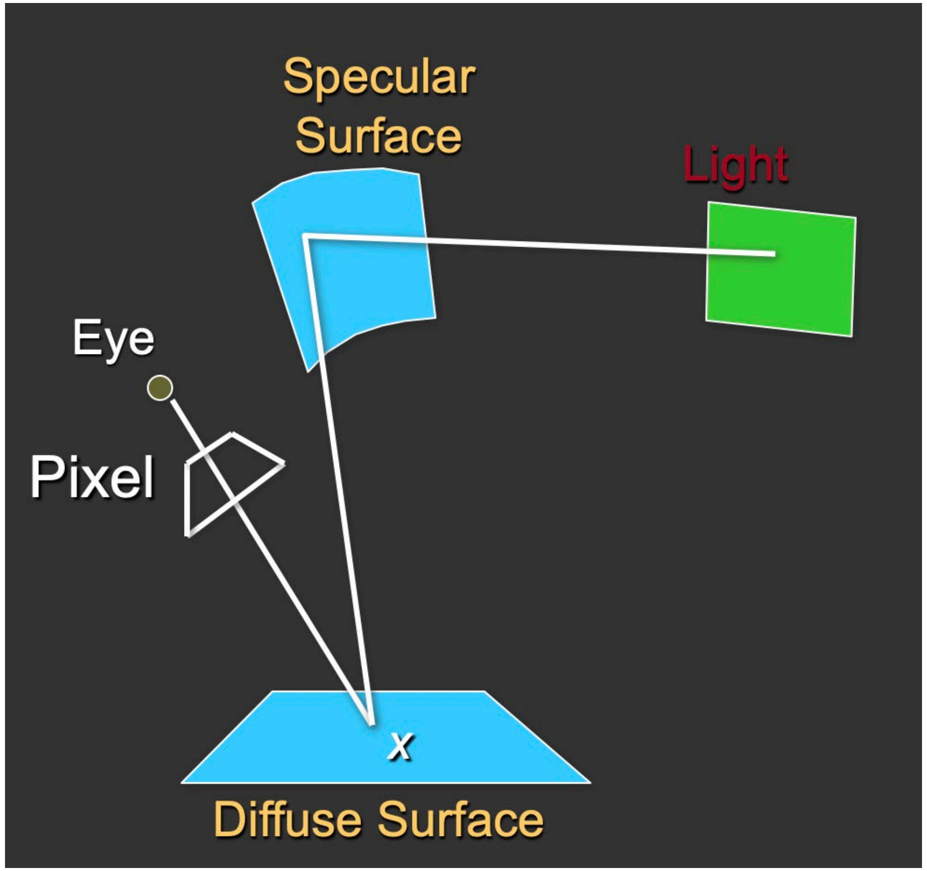
Monte-Carlo methods randomly choose samples



Monte-Carlo rendering

Monte-Carlo path tracing

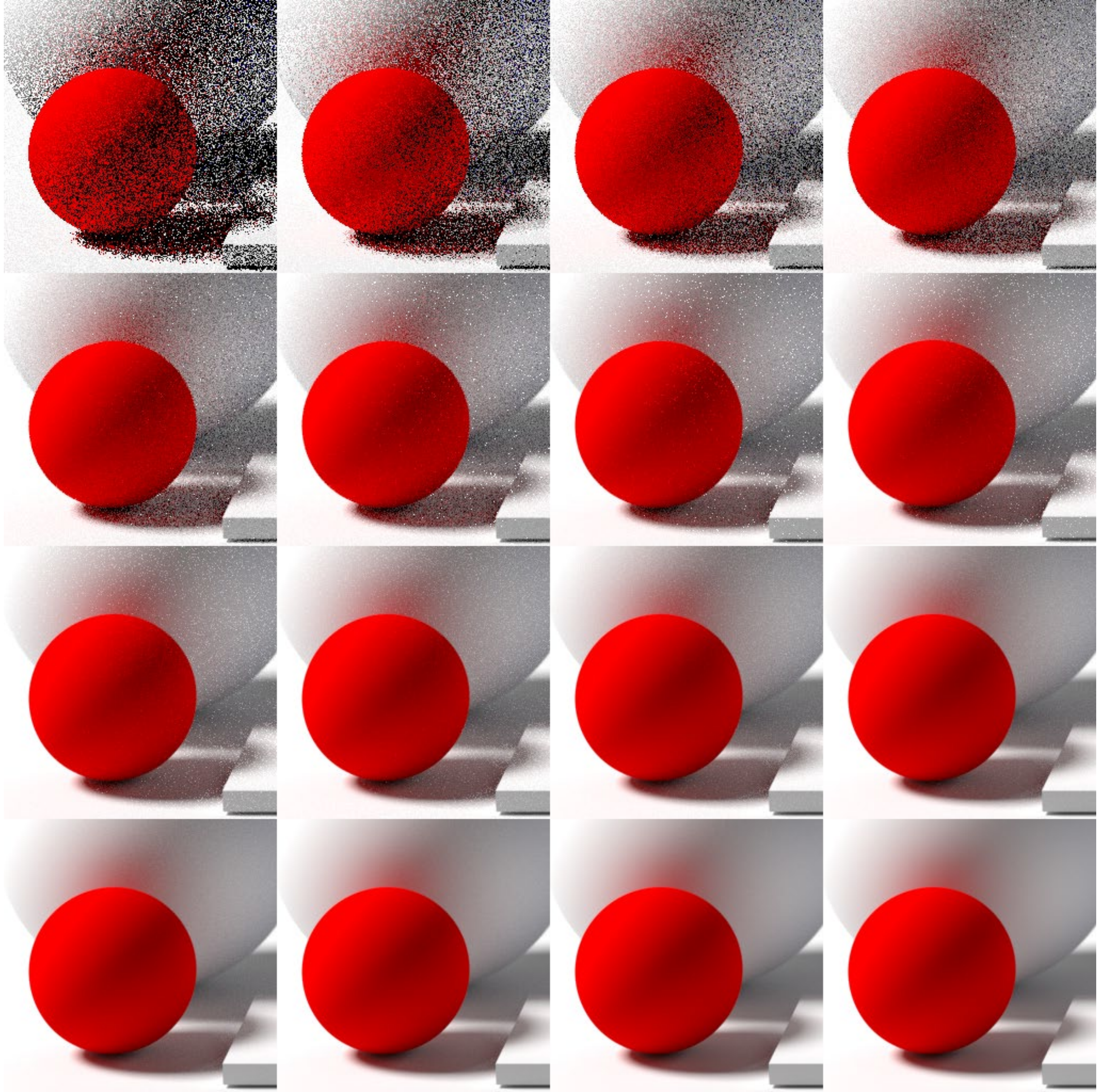
- Solving the rendering equation
- Integrate radiance for each pixel by sampling paths randomly
- Partition the rendering equation into direct and indirect illumination
- Use Monte-Carlo to estimate each partition
- Importance-sampling, according to BRDF and light sources



Slide by Lior Yariv

Noise decreases as the number of samples per pixel increases.

The top left shows 1 sample per pixel, and doubles from left to right each square.




p
Direct illumination


p
One-bounce global illumination


p
Two-bounce global illumination


p
Four-bounce global illumination


p
Eight-bounce global illumination


p
Sixteen-bounce global illumination

Summary

- Computer graphics, in particular rendering: **ray tracing** and **rasterization**

- Geometry representation, including implicit and explicit (**triangular mesh**),
  and the use of barycentric coordinates

- **Radiometry**, including radiance and irradiance

- **Materials** properties are encoded by **BRDF** (Bidirectional reflectance distribution function)

- Illumination models
    - **local model** -> reflection equation
    - **global model** -> rendering equation

- Very challenging to solve the **rendering equation**

- Simplifications by **Monte-Carlo sampling**

- Also handful of ways using **deep learning** (next time)