# Image-based Lighting

T2

Computational Photography

Yuxiong Wang, University of Illinois

Slides adopted from Derek Hoiem

Many slides from Debevec, some from Efros

# Next two classes

Today

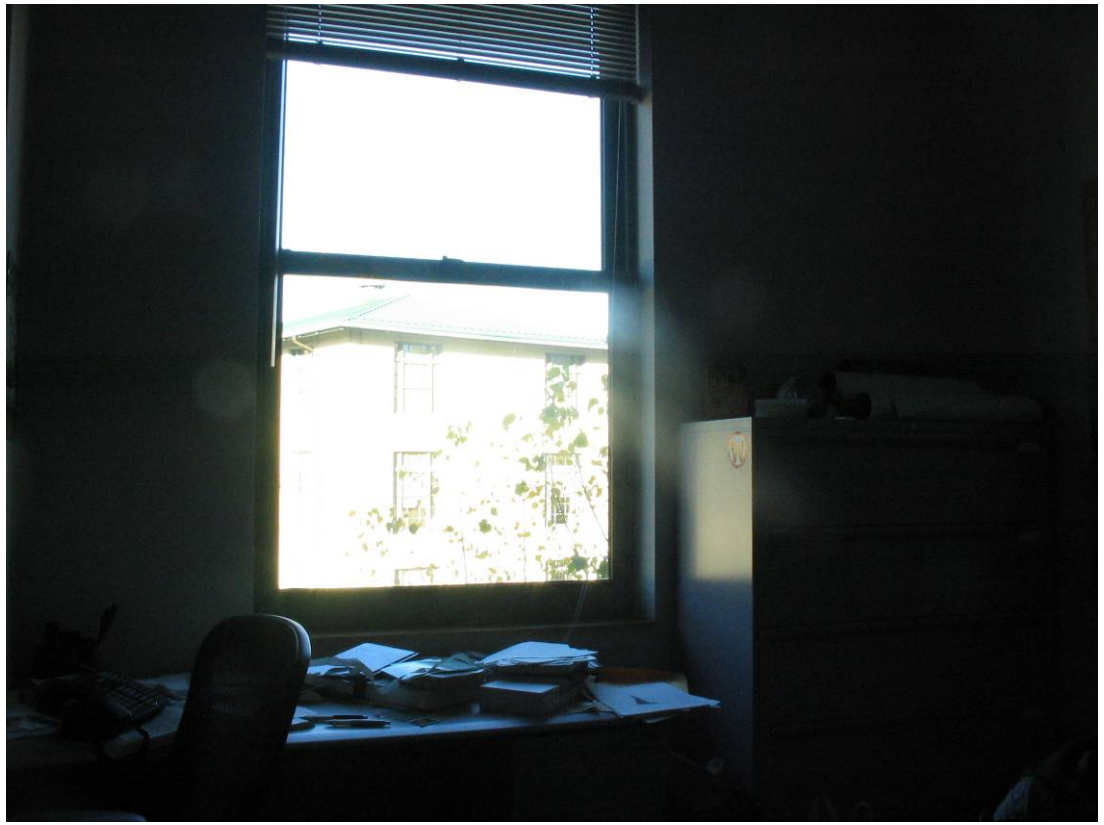
- Start on ray tracing, environment maps, and relighting 3D objects (project 4 topics)

Next class

- More HDR, light probes, etc.

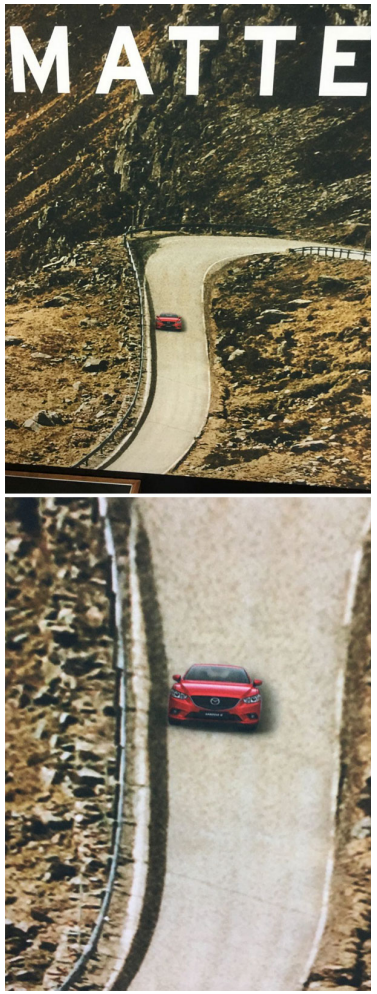
# Image-based Lighting Project

# How to render an object inserted into an image?



What's wrong with the teapot?

# Relighting is important!

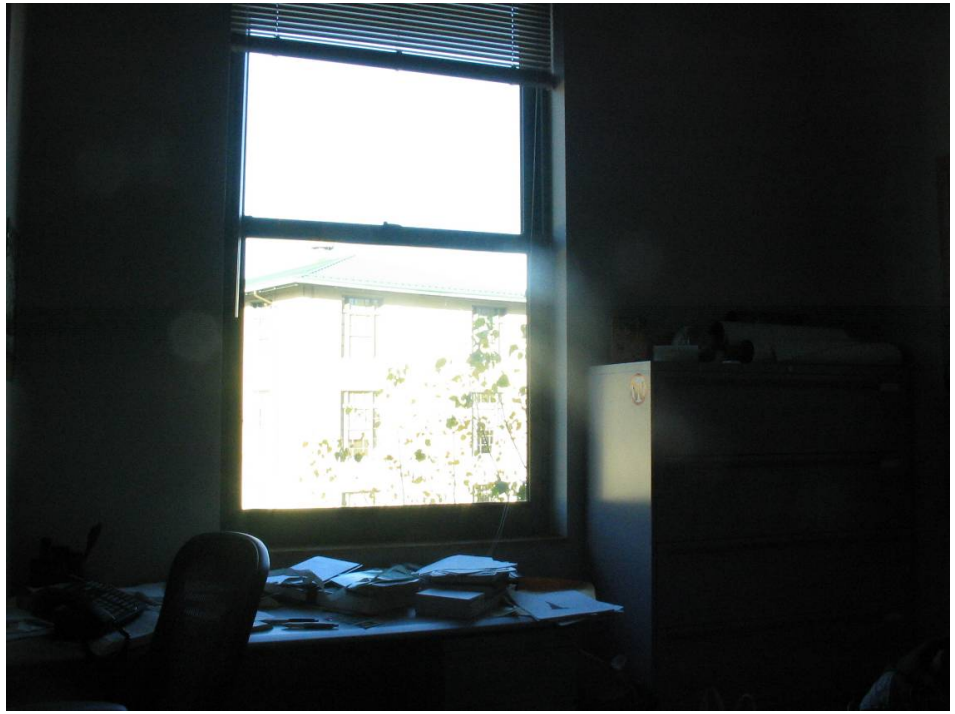



https://www.boredpanda.com/

# How to render an object inserted into an image?

## Traditional graphics way

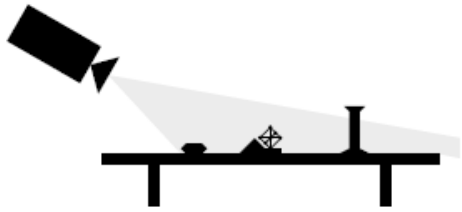
- Manually model BRDFs of all room surfaces
- Manually model radiance of lights
- Do ray tracing to relight object, shadows, etc.



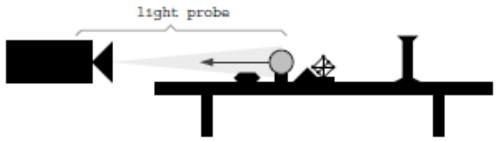
# How to render an object inserted into an image?

## Image-based lighting

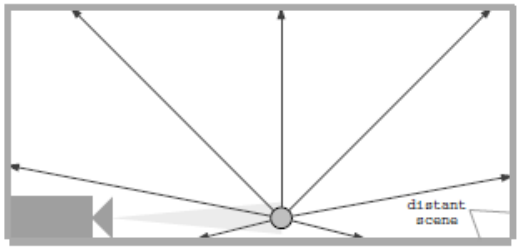
- Capture incoming light with a “light probe”
- Model local scene
- Ray trace, but replace distant scene with info from light probe



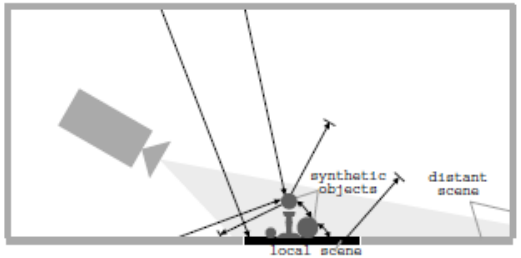
(a) Acquiring the background photograph




(b) Using the light probe




(c) Constructing the light-based model

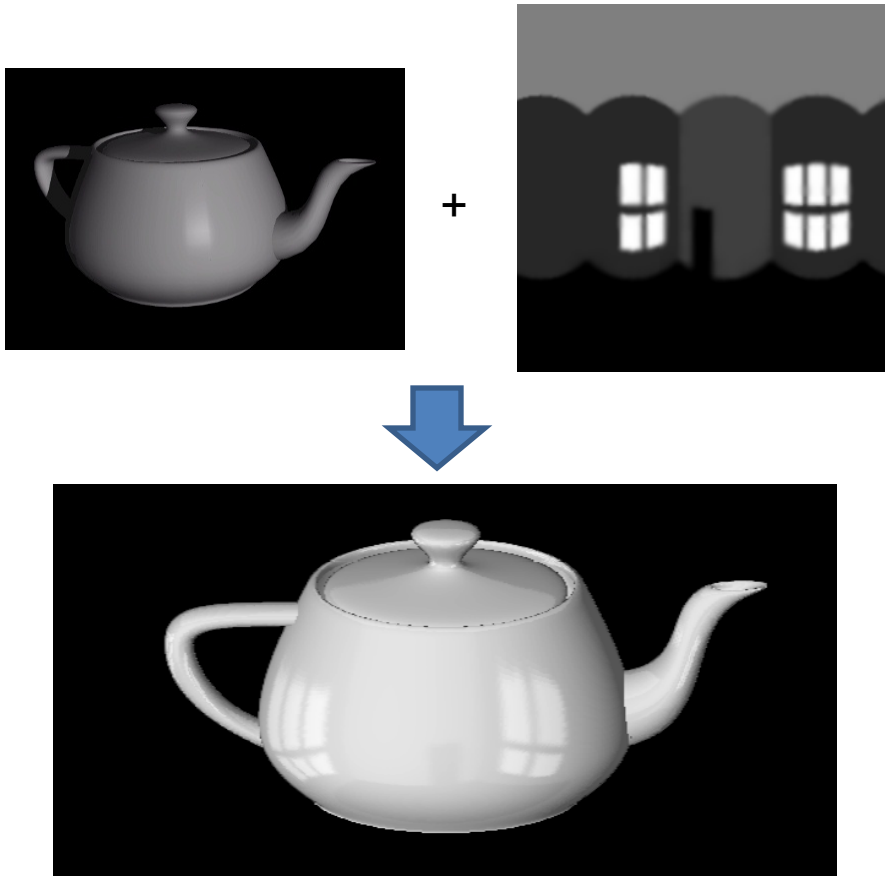



(d) Computing the global illumination solution

Debevec SIGGRAPH 1998

# Key ideas for Image-based Lighting

- Environment maps: tell what light is entering at each angle within some shell



# Key ideas for Image-based Lighting

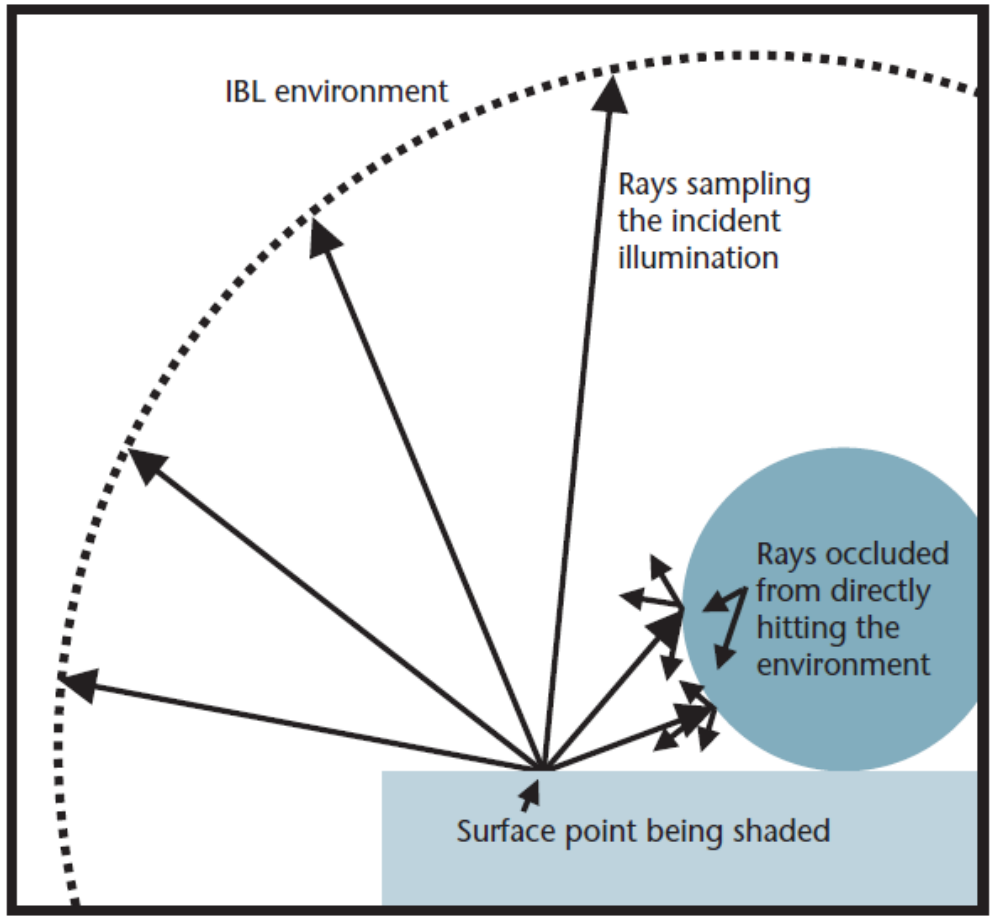
- Light probes: a way of capturing environment maps in real scenes

# Key ideas for Image-based Lighting

- Capturing HDR images: needed so that light probes capture full range of radiance

# Key ideas for Image-based Lighting

- Relighting: environment map acts as light source, substituting for distant scene

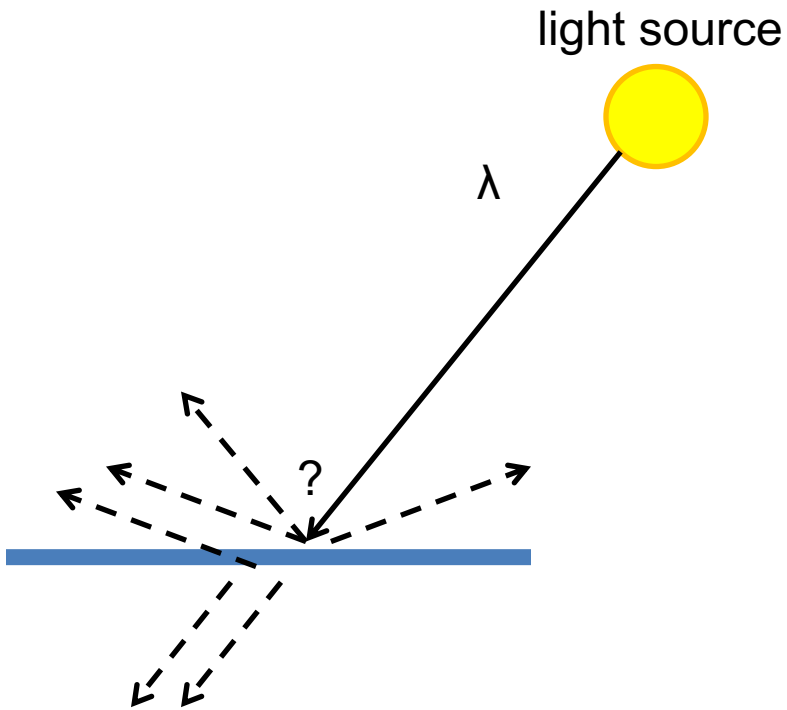


# Today

- Ray tracing
- Capturing environment maps

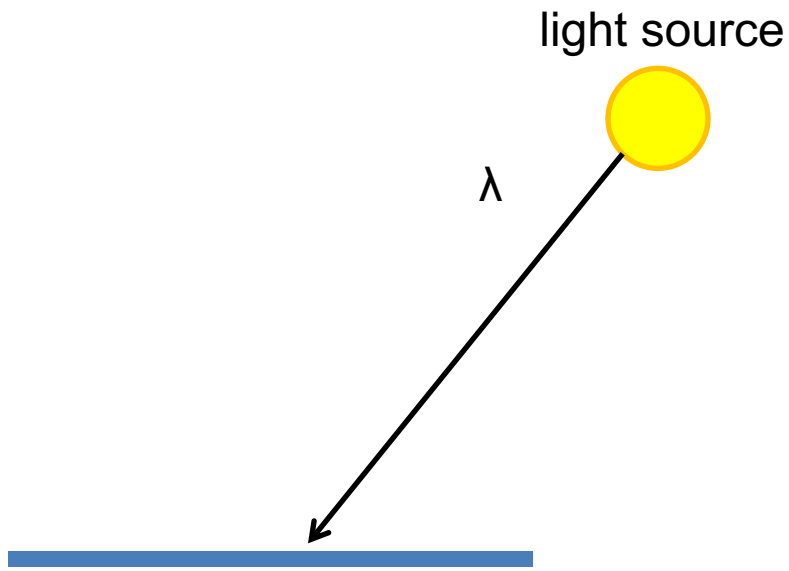
# A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- Refraction
- Fluorescence
- Subsurface scattering
- Phosphorescence
- Interreflection



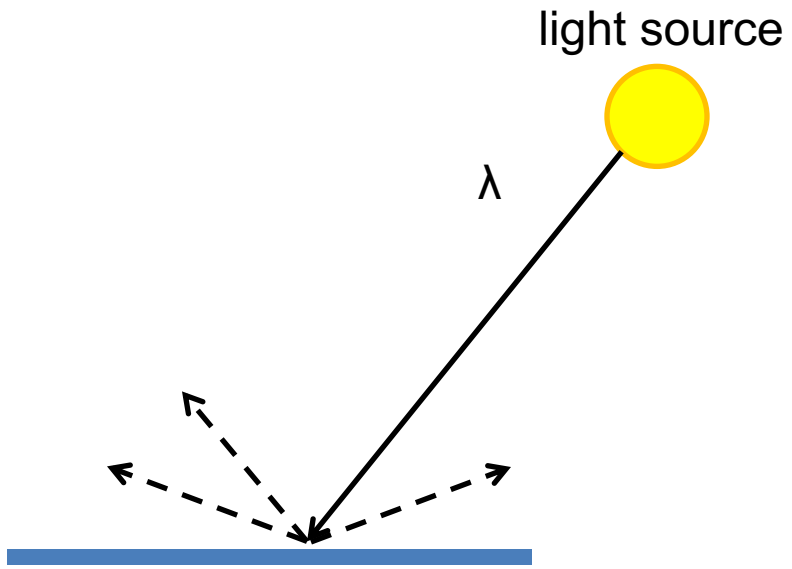
# A photon’s life choices

- **Absorption**
- Diffusion
- Reflection
- Transparency
- Refraction
- Fluorescence
- Subsurface scattering
- Phosphorescence
- Interreflection



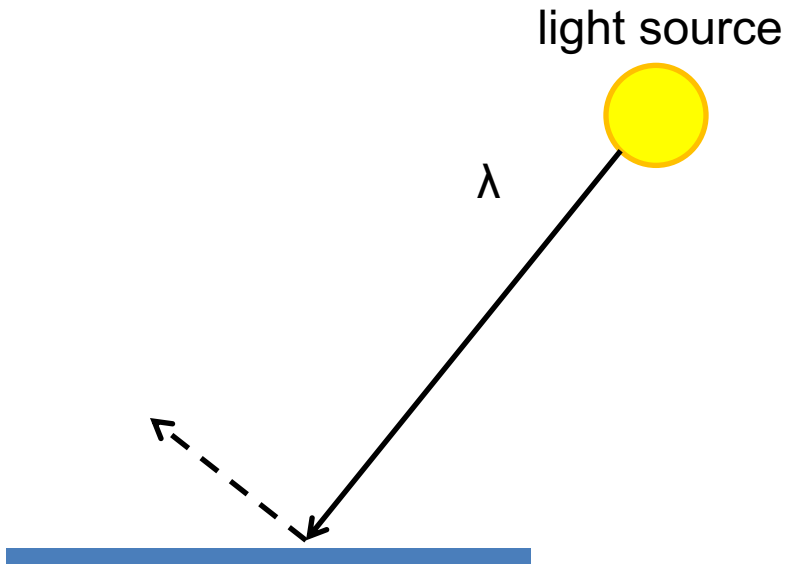
# A photon’s life choices

- Absorption
- **Diffuse Reflection**
- Reflection
- Transparency
- Refraction
- Fluorescence
- Subsurface scattering
- Phosphorescence
- Interreflection



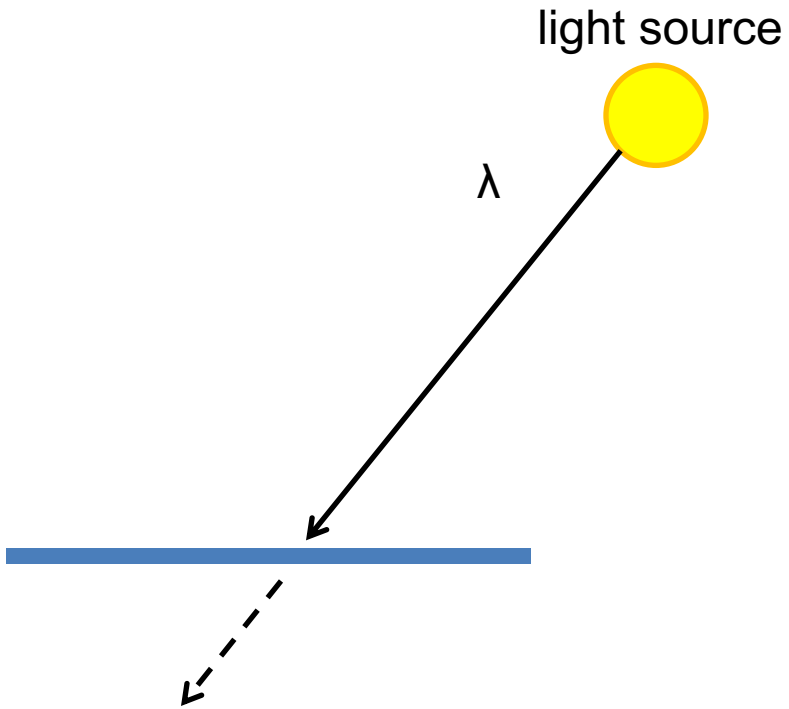
# A photon’s life choices

- Absorption
- Diffusion
- **Specular Reflection**
- Transparency
- Refraction
- Fluorescence
- Subsurface scattering
- Phosphorescence
- Interreflection



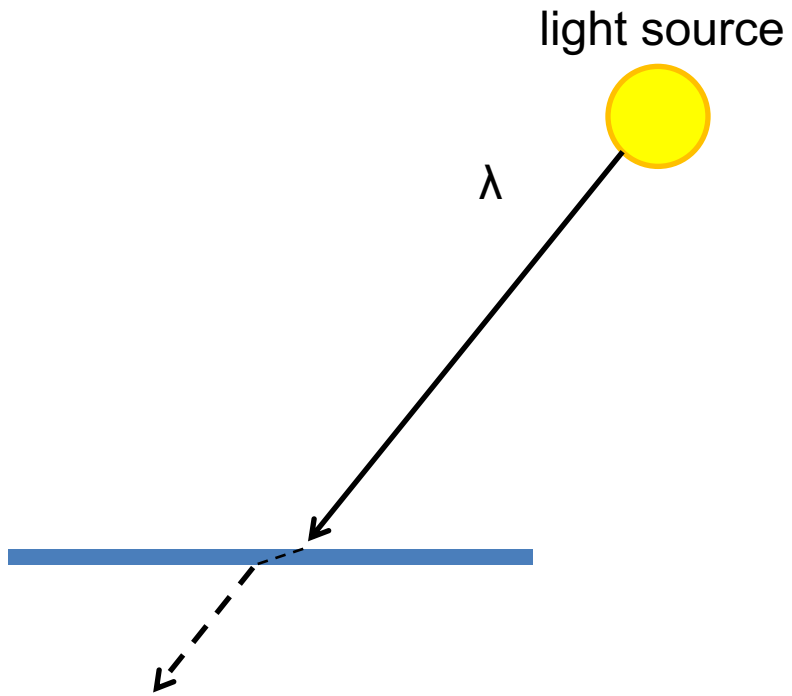
# A photon’s life choices

- Absorption
- Diffusion
- Reflection
- **Transparency**
- Refraction
- Fluorescence
- Subsurface scattering
- Phosphorescence
- Interreflection



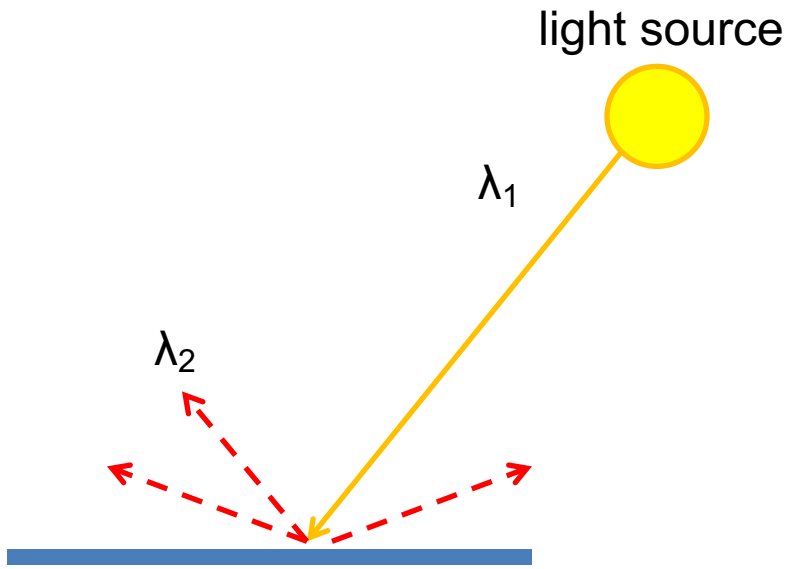
# A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- **Refraction**
- Fluorescence
- Subsurface scattering
- Phosphorescence
- Interreflection



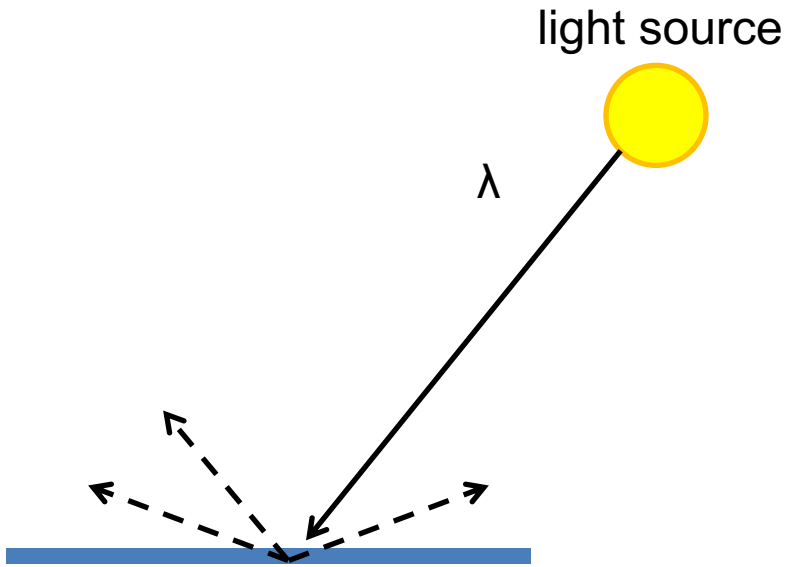
# A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- Refraction
- **Fluorescence**
- Subsurface scattering
- Phosphorescence
- Interreflection



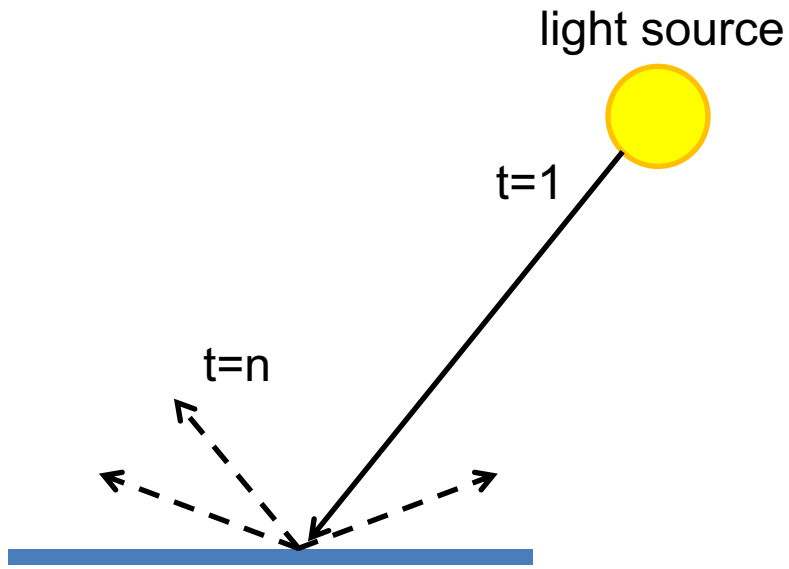
# A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- Refraction
- Fluorescence
- **Subsurface scattering**
- Phosphorescence
- Interreflection



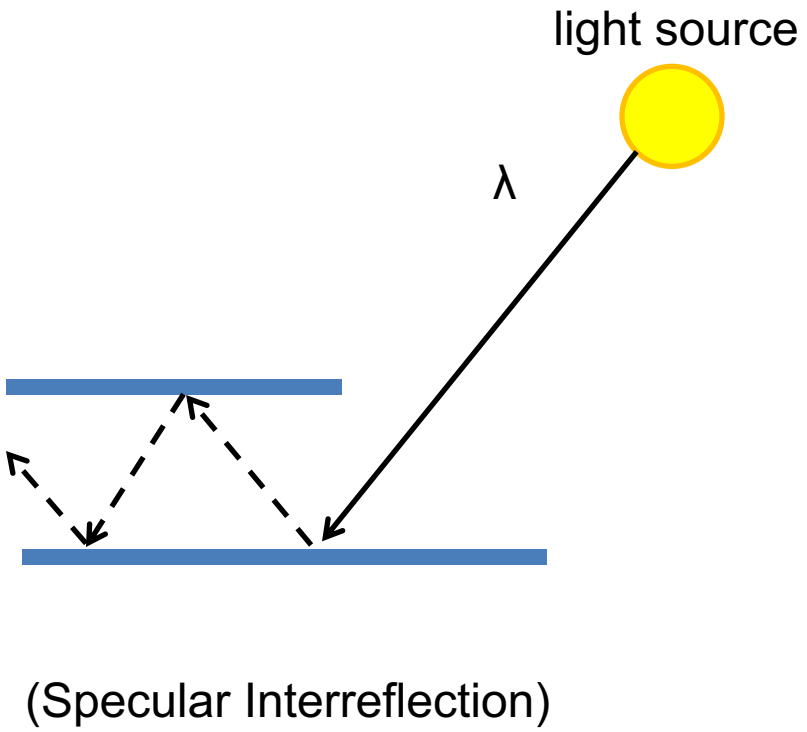
# A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- Refraction
- Fluorescence
- Subsurface scattering
- **Phosphorescence**
- Interreflection



# A photon’s life choices

- Absorption
- Diffusion
- Reflection
- Transparency
- Refraction
- Fluorescence
- Subsurface scattering
- Phosphorescence
- **Interreflection**



# Where are the light sources in this room?

http://www.flickr.com/photos/chrisdonbavand/493707413/sizes/z/in/photostream/

# Rendering Equation

Outgoing light

Generated light

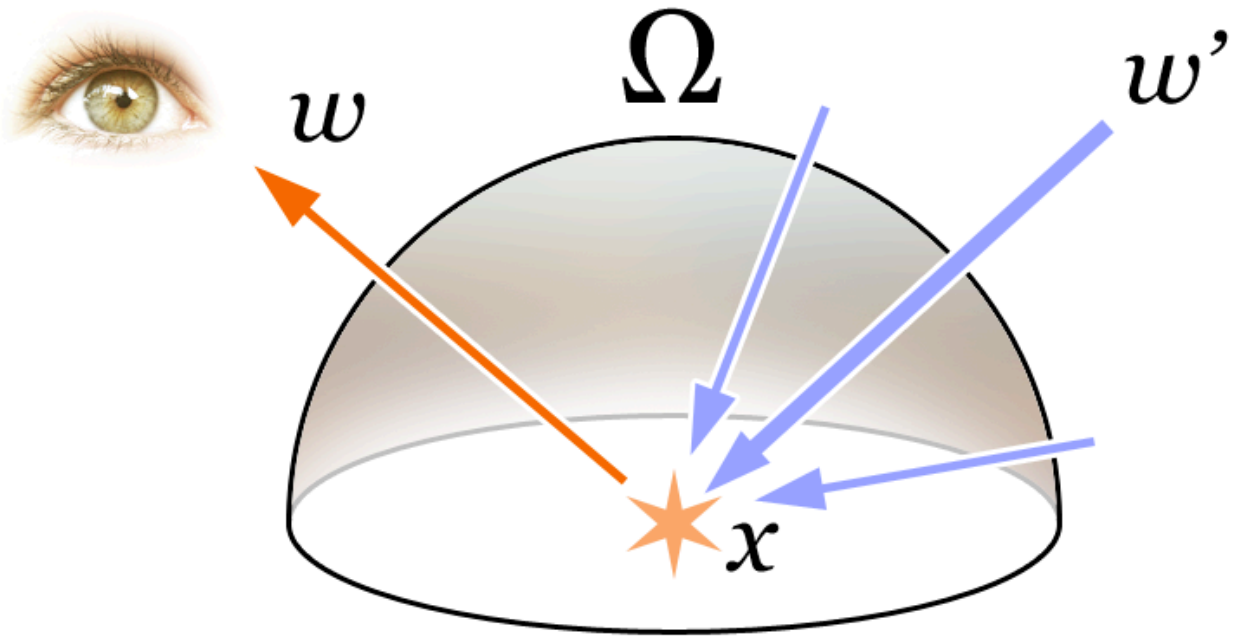
Total reflected light

$$L_o(\mathbf{x},\omega,\lambda,t) = L_e(\mathbf{x},\omega,\lambda,t) + \int_\Omega f_r(\mathbf{x},\omega',\omega,\lambda,t) L_i(\mathbf{x},\omega',\lambda,t)(-\omega' \cdot \mathbf{n}) d\omega'$$

BRDF

Incoming Light

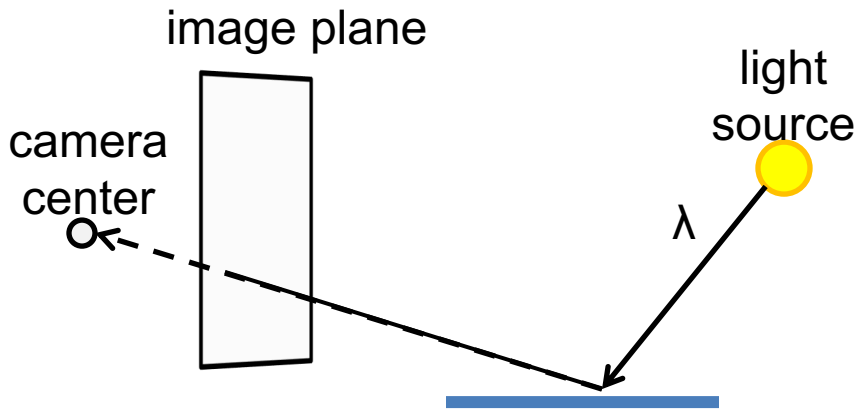
Incident angle

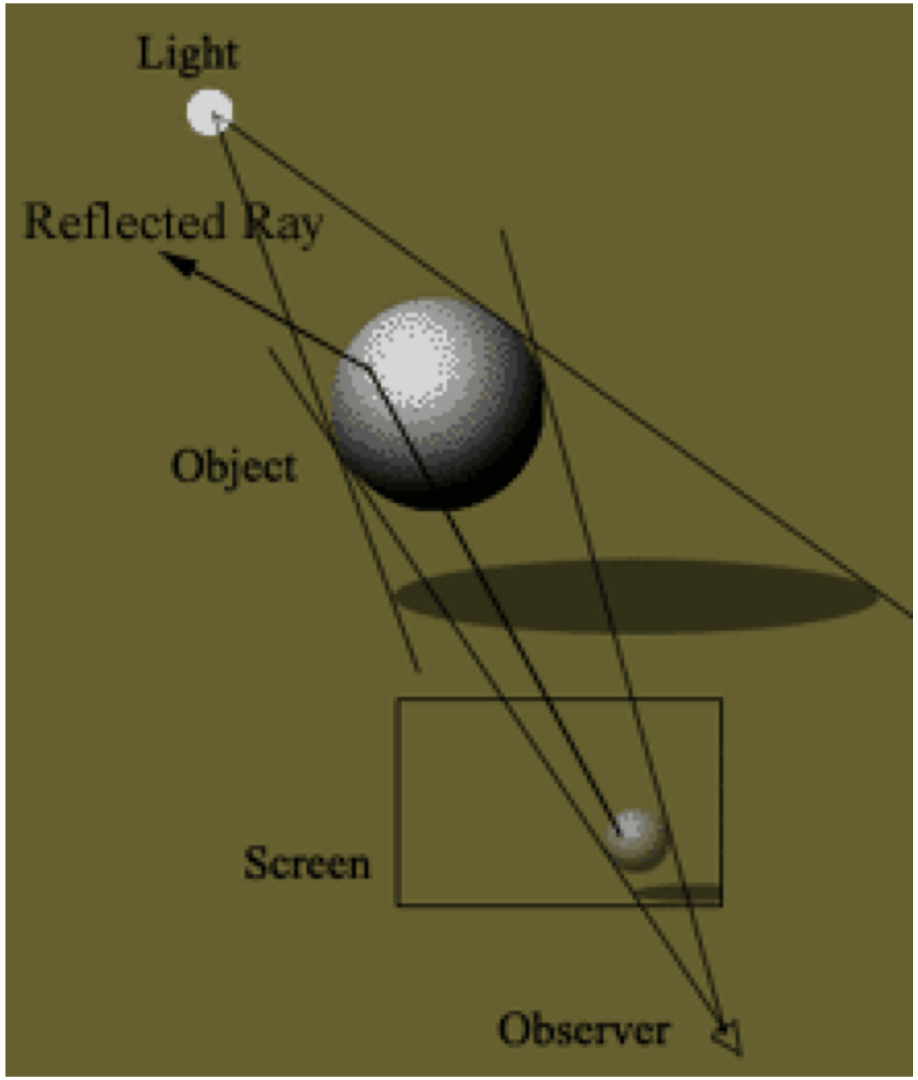


# Rendering a scene with ray tracing

http://en.wikipedia.org/wiki/File:Glasses_800_edit.png

# Ray tracing: basics



# Ray casting

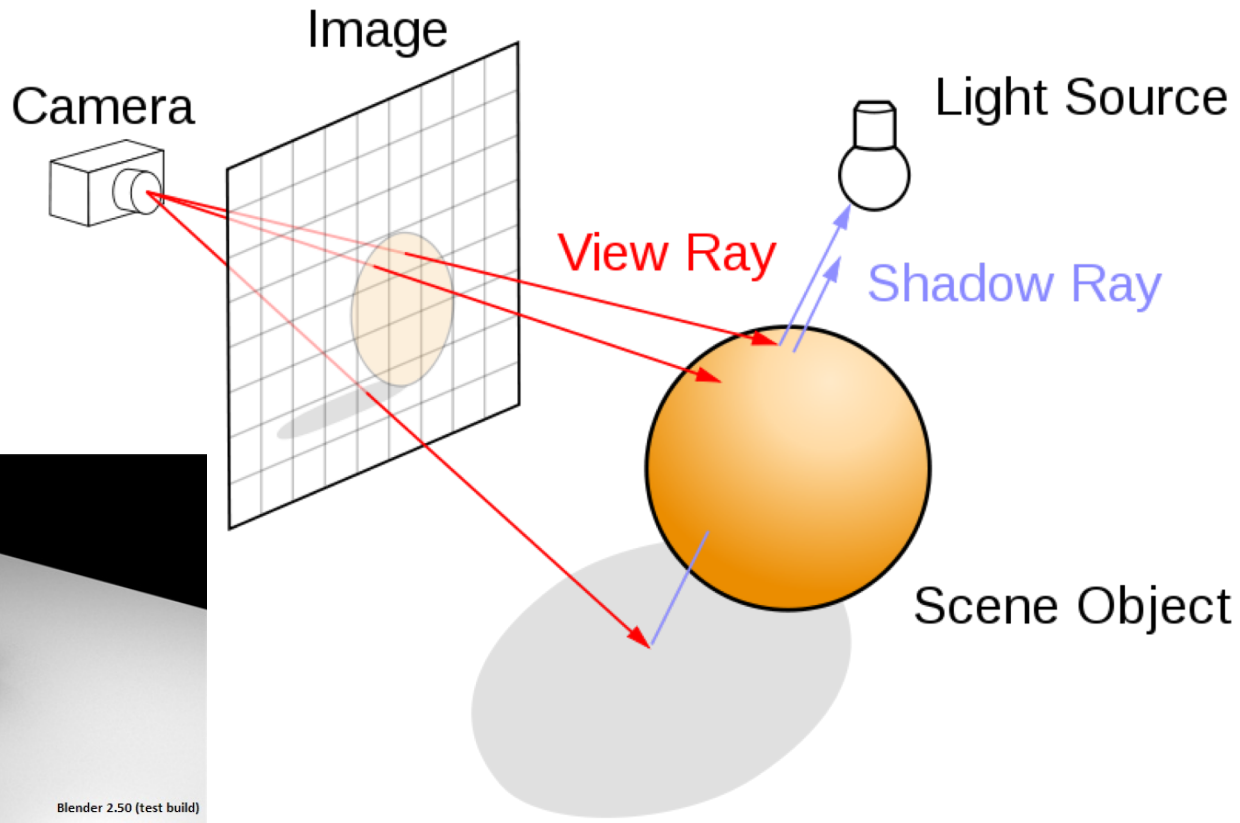
- Store colors of surfaces, cast out rays, see what colors each ray hits

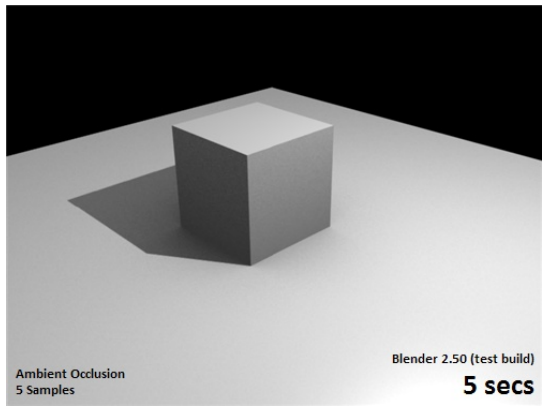

Wolfenstein 3D (1992)

# Ray tracing: fast approximation

Upon hitting a surface

- Cast reflection/refraction ray to determine reflected or refracted surface
- Cast shadow ray: go towards light and see if an object is in the way

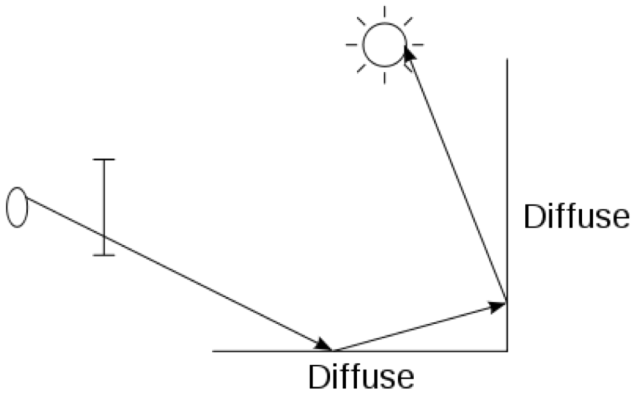


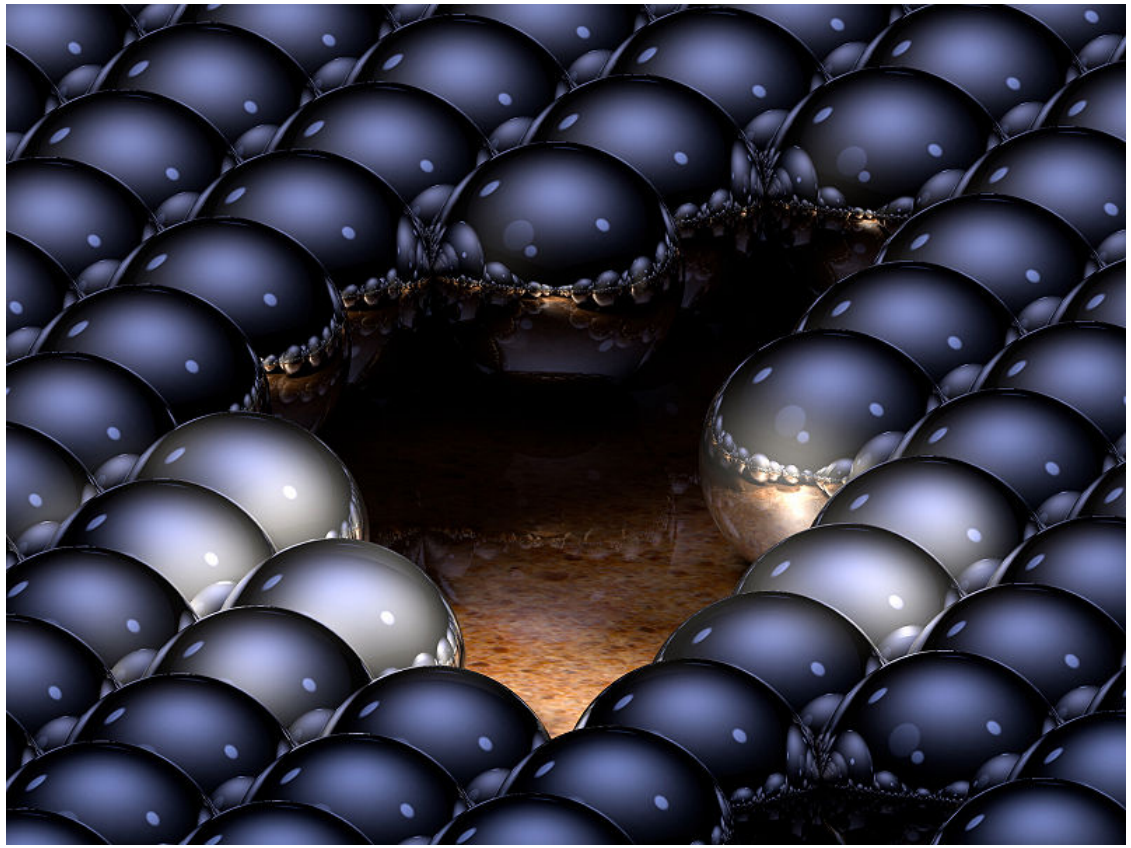

http://en.wikipedia.org/wiki/File:Ray_trace_diagram.svg

# Ray tracing: interreflections

- Reflect light N times before heading to light source




N=2



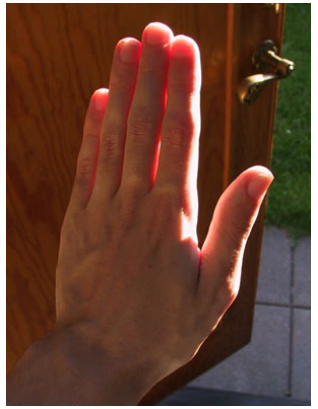
N=16



http://en.wikipedia.org/wiki/Ray_tracing_(graphics)#mediaviewer/File:Ray-traced_steel_balls.jpg

# Ray tracing

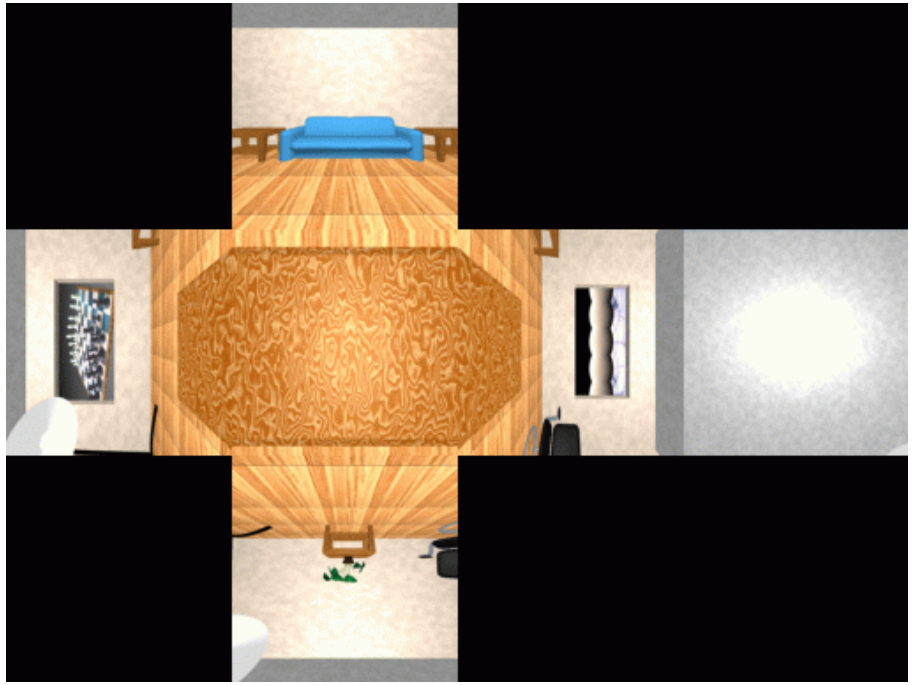
- Conceptually simple but hard to do fast
- Full solution requires tracing millions of rays for many inter-reflections
- Design choices
  - Ray paths: Light to camera vs. camera to light?
  - How many samples per pixel (avoid aliasing)?
  - How to sample diffuse reflections?
  - How many inter-reflections to allow?
  - Deal with subsurface scattering, etc?



# Environment Maps

- The environment map may take various forms:
  - Cubic mapping
  - Spherical mapping
  - other
- Describes the shape of the surface on which the map “resides”
- Determines how the map is generated and how it is indexed

# Cubic Map Example



# Cubic Mapping

- The map resides on the surfaces of a cube around the object
  - Typically, align the faces of the cube with the coordinate axes
- To generate the map:
  - For each face of the cube, render the world from the center of the object with the cube face as the image plane
    - Rendering can be arbitrarily complex (it's off-line)
- To use the map:
  - Index the R ray into the correct cube face
  - Compute texture coordinates

# Spherical Map Example

# Sphere mapping

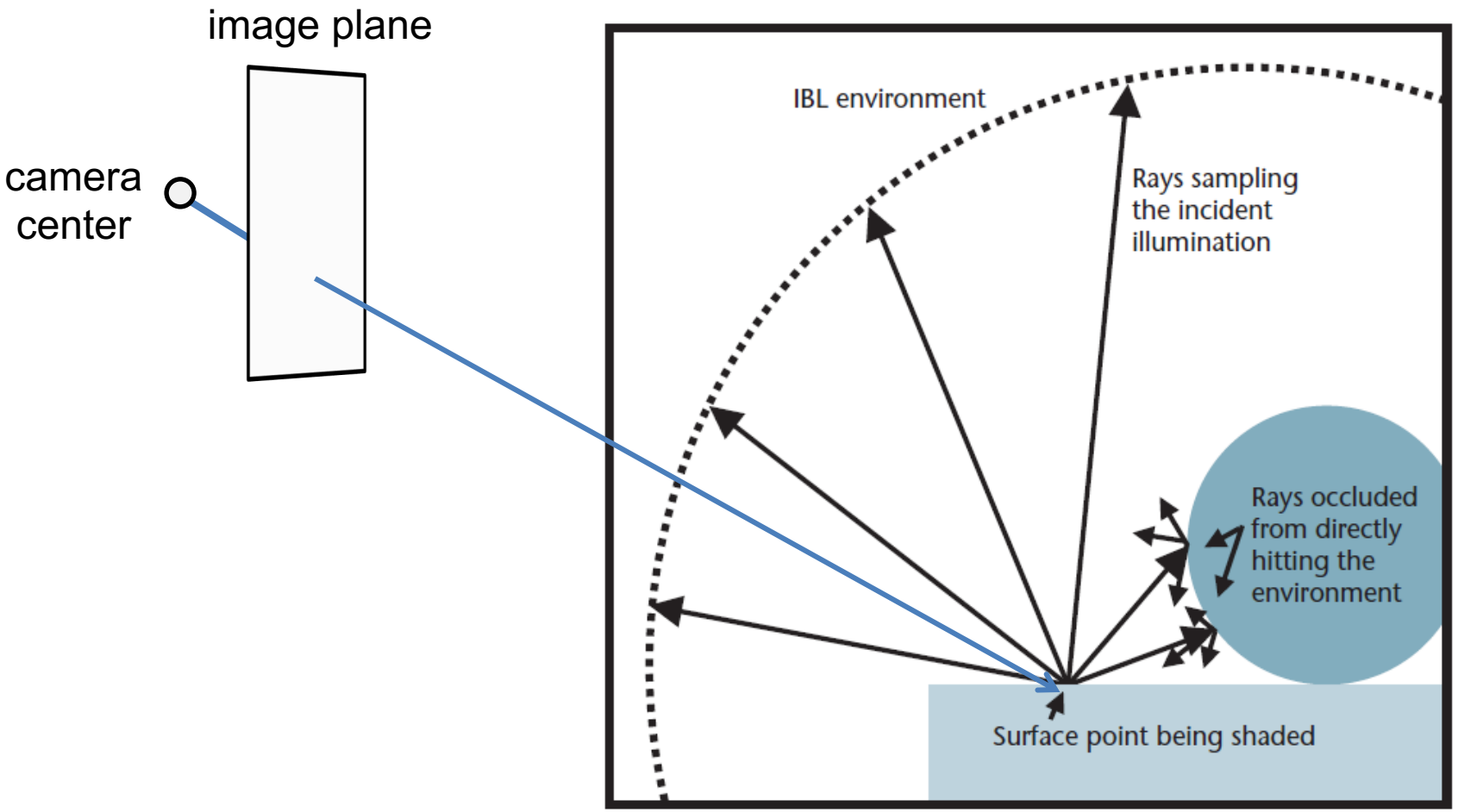
- Map lives on a sphere
- To generate the map:
  - Render a spherical panorama from the designed center point
  - “Render” means obtain pixel intensities by casting rays from the camera center through the pixel positions
- Rendering with environment map:
  - Use the orientation of the R ray to index directly into the sphere

# What approximations are made?

- The map should contain a view of the world with the point of interest on the object as the Center of Projection (CoP)
  - We can't store a separate map for each point, so one map is used with the CoP at the center of the object
  - Introduces distortions in the reflection, but we usually don't notice
  - Distortions are minimized for a small object in a large room

- The object will not reflect itself (based on the environment map)

# Rendering with environment maps and local models



# Storing environment maps

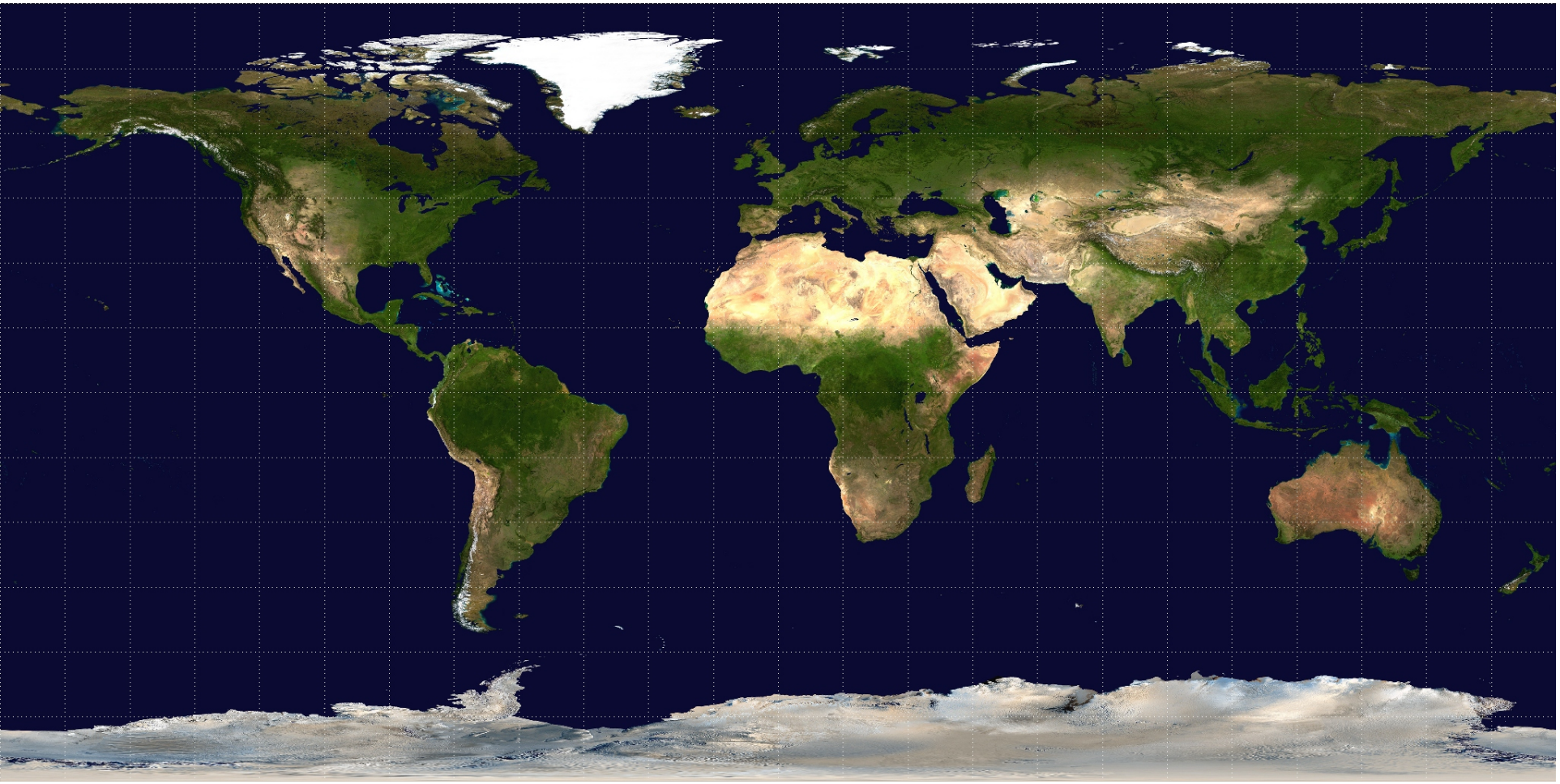
Angular mapped

Spherical
Equirectangular
LatLong
Latitude/Longitude

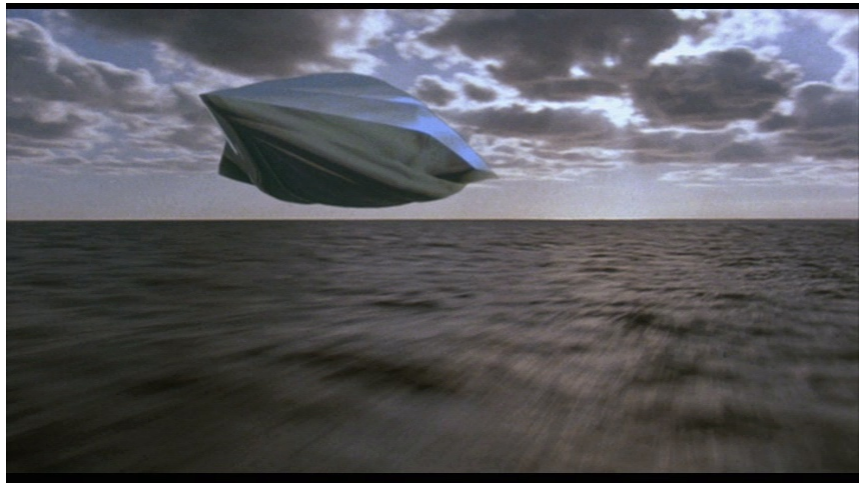
Vertical Cross
Cubic (vcross)

# Equirectangular (latitude-longitude) projection

# Equirectangular (latitude-longitude) projection



# How to capture light in real scenes?



From *Flight of the Navigator*

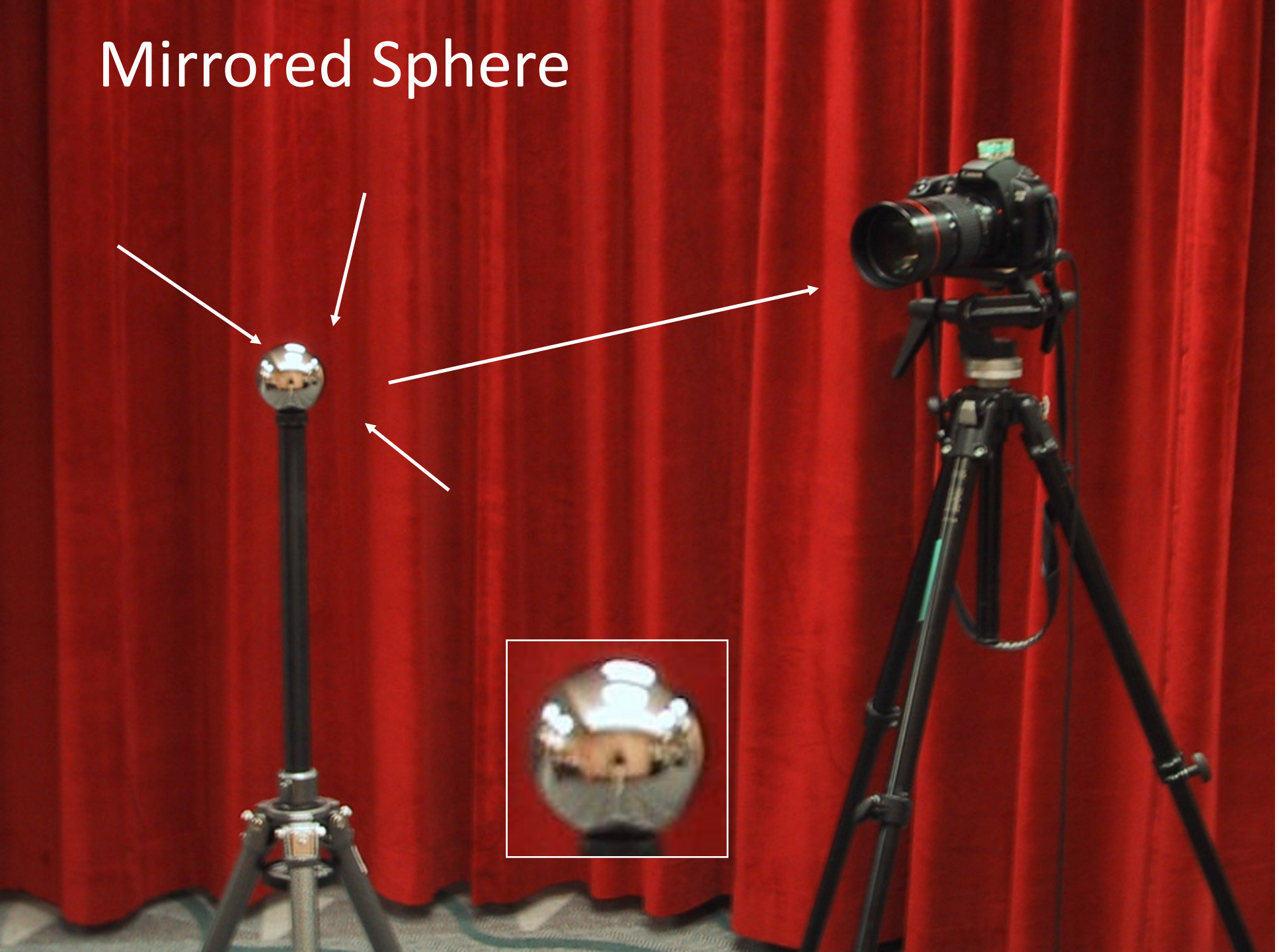
# How to capture light in real scenes?

from Terminator 2

# Real environment maps

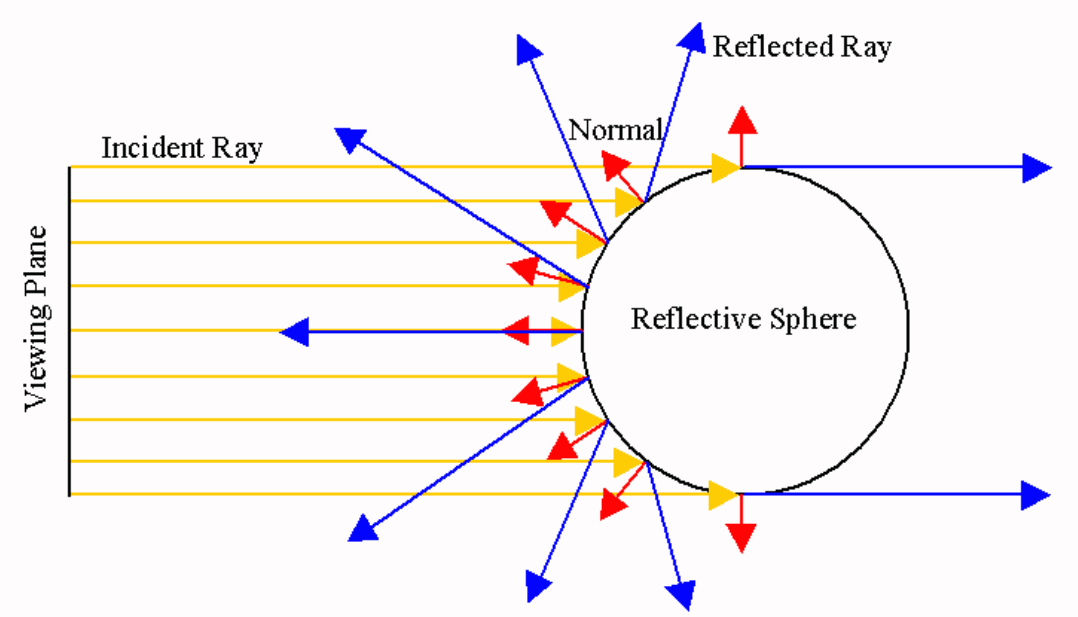
- We can use photographs to capture environment maps
  - The first use of panoramic mosaics
  - Fisheye lens
  - 360 camera
  - Mirrored balls (**light probes**)

# Mirrored Sphere




Canon
Canon

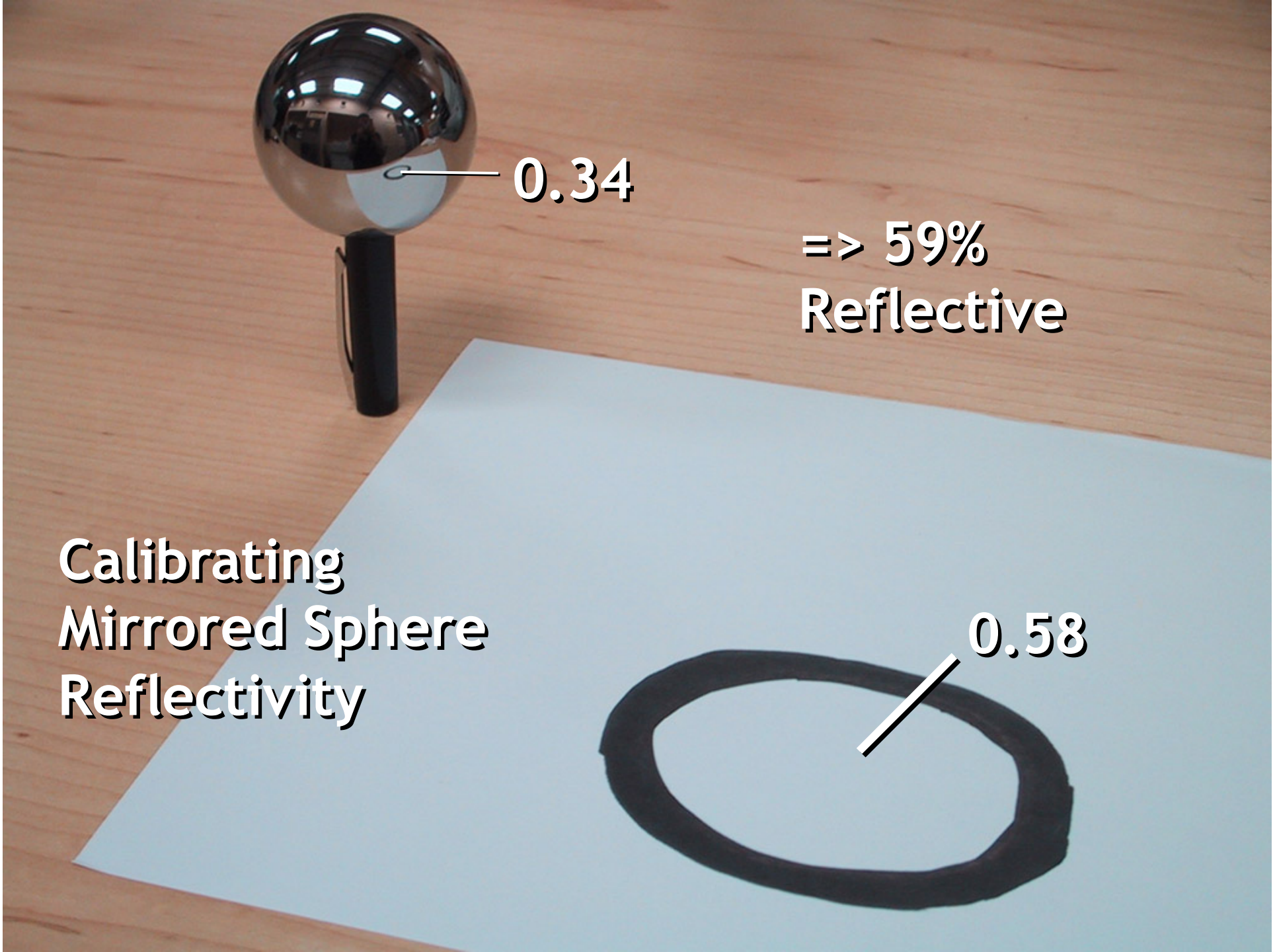
# One picture of a mirrored ball received light coming into the ball from nearly all angles (including behind)




# Mirror balls for image-based lighting

# Mirror balls for image-based lighting

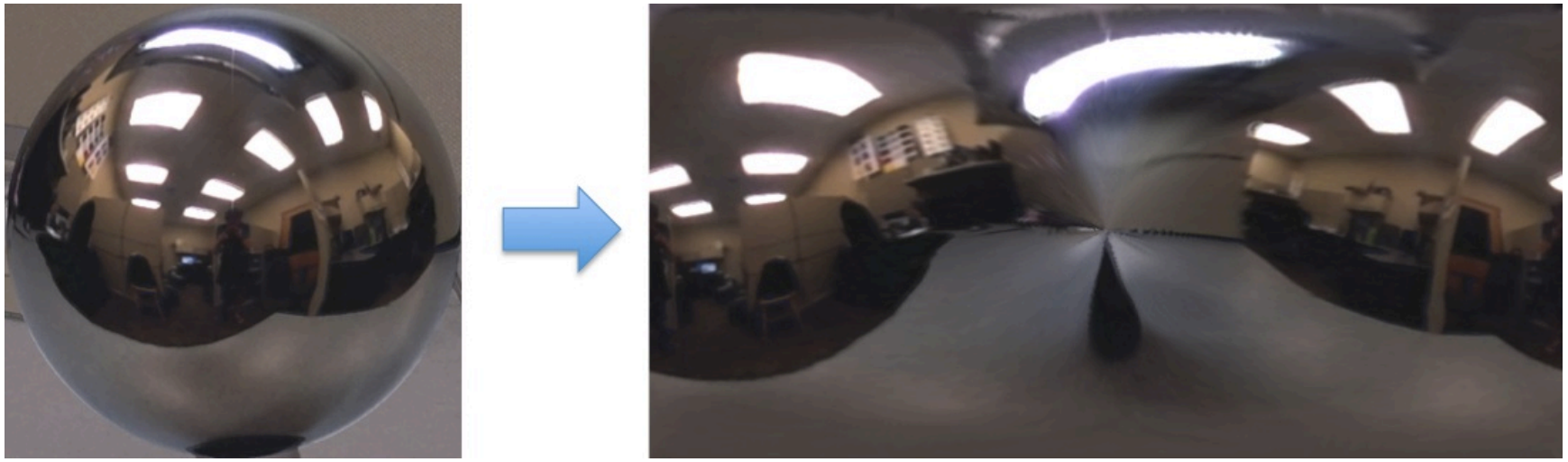
# Mirror balls for image-based lighting


0.34
=> 59%
Reflective
Calibrating
Mirrored Sphere
Reflectivity
0.58

# Spherical map domain transformations

- Many rendering programs only accept one format (mirror ball, equirectangular, cube map, etc)
  - E.g. Blender only accepts equirectangular maps
- How to convert mirror ball to equirectangular?

# Mirror ball -> equirectangular

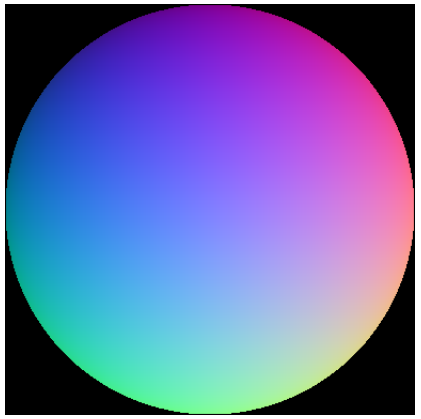


# Mirror ball -> equirectangular

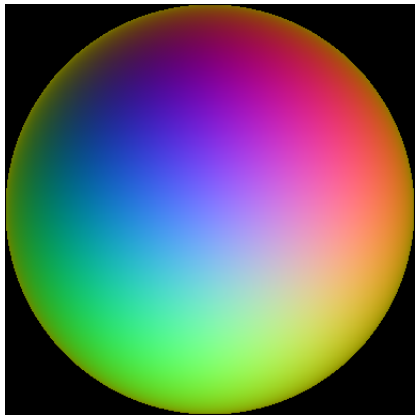
- Spherical coordinates
  - Convert the light directions incident to the ball into spherical coordinates (phi, theta)
  - Map from mirror ball phi, theta to equirectangular phi, theta

# Mirror ball -> equirectangular

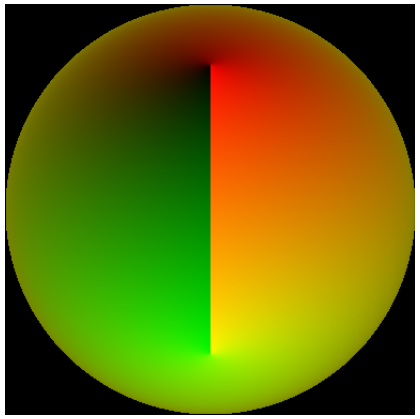
Mirror ball



Normals

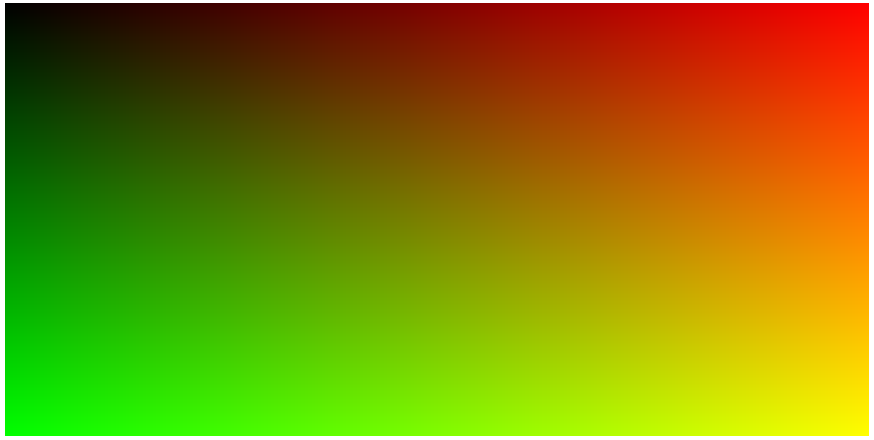


Reflection vectors



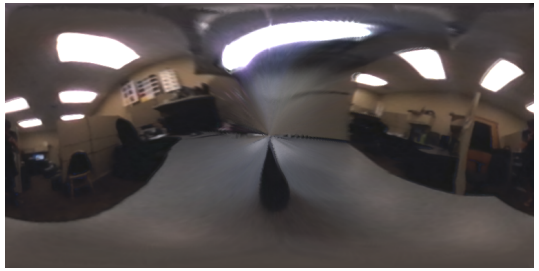
Phi/theta of reflection vecs

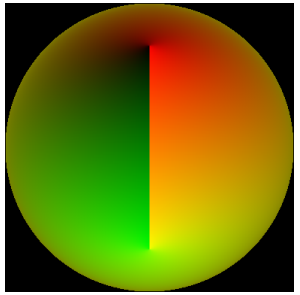
Equirectangular



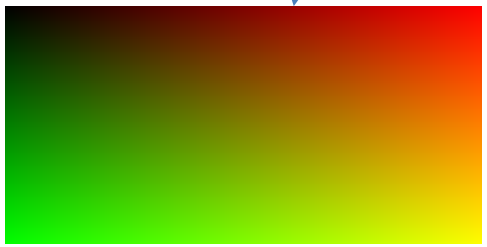
Phi/theta equirectangular domain

# Mirror ball -> equirectangular



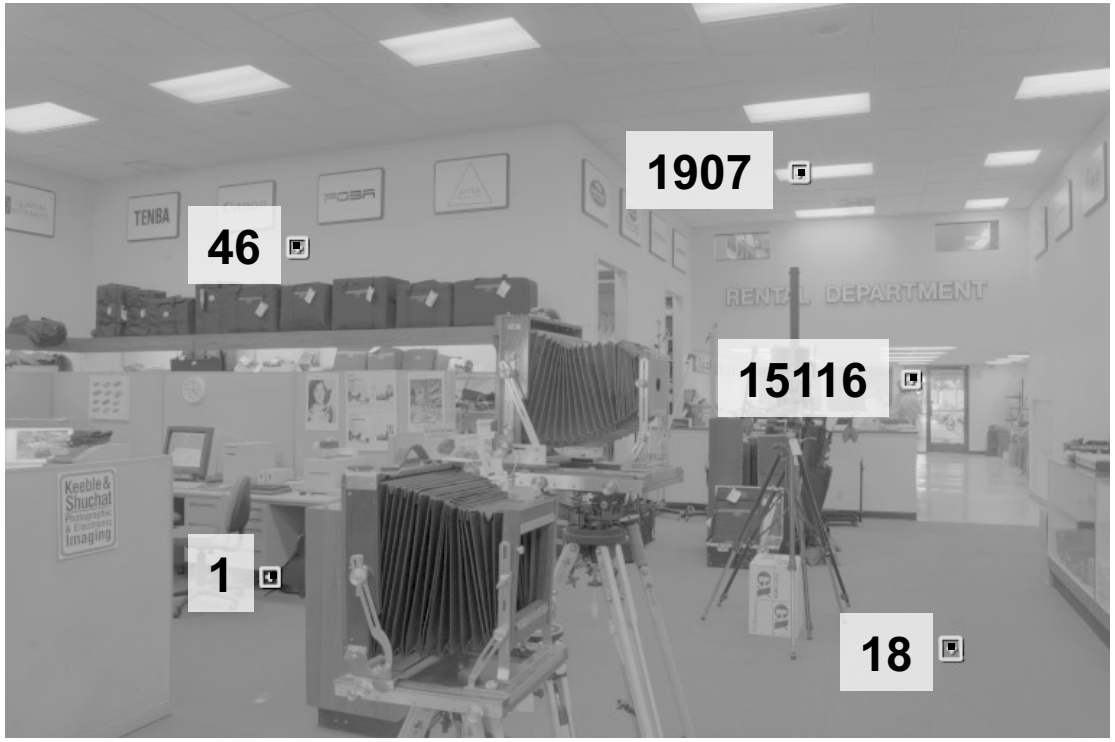


```
latlon(:, :, c) = griddata( (phi_ball, theta_ball),
                            mirrorball(:, :, c),
                            (phi_latlon, theta_latlon),
                            method='linear',
                            fill_value='0')
```



# One small snag

- How do we deal with light sources?  Sun, lights, etc?
  - They are much, much brighter than the rest of the environment



# Problem: Dynamic Range

# Problem: Dynamic Range

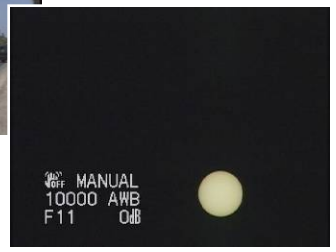

1

**The real world is high dynamic range**


1500

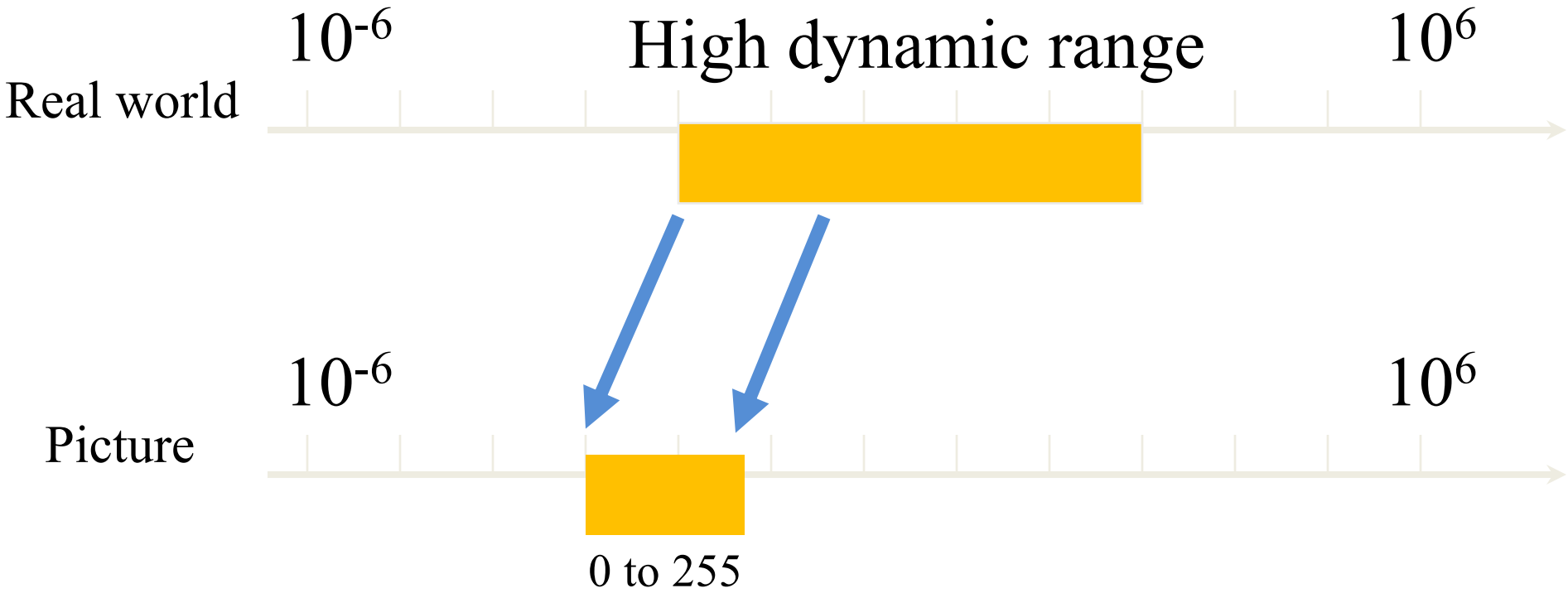

25,000


400,000

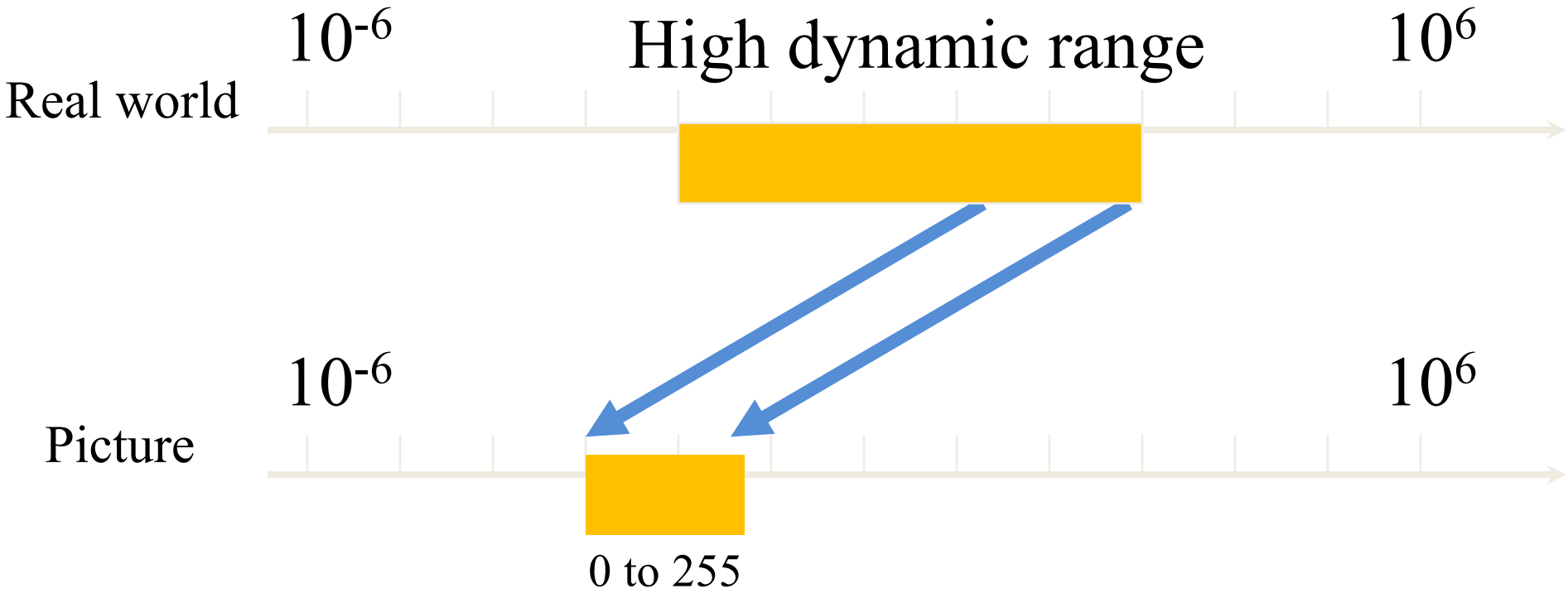



2,000,000,000

# Long Exposure



# Short Exposure



# Varying Exposure

# Camera is not a photometer

- Dynamic range is limited
  - Can use multiple exposures to capture fuller range
- Responds non-linearly to photon intensity
- Solution: Recover response curve from multiple exposures and reconstruct the *radiance map*

# Next class

- How to capture HDR image using “bracketing”
- How to relight an object from an environment map

# Next class

- How to capture HDR image using “bracketing”
- How to relight an object from an environment map
- Traveling: Pre-recorded video + TA Q&A