Lecture 3: Introduction to Computer Graphics

Respond at **PollEv.com/ronisen**

Text **RONISEN** to **22333** once to join, then text your message

Feel free to share your questions...

Neural Rendering

Current Image

Vision

3D Intrinsic Components

Edit Components

Graphics

New Image under different conditions

Explicit: Reconstruct 3D (Introduction to Graphics Lectures)

Implicit: Neural Representation (Generative Models Lectures

Recap

- How do we define geometry/shape of an object?
- How do we define a camera model? – 3D object to 2D image
- How do we define material property? – glossy, metallic

Geometry: How do we represent shape of an object?

2.5D representation:

1) Depth & Normal map

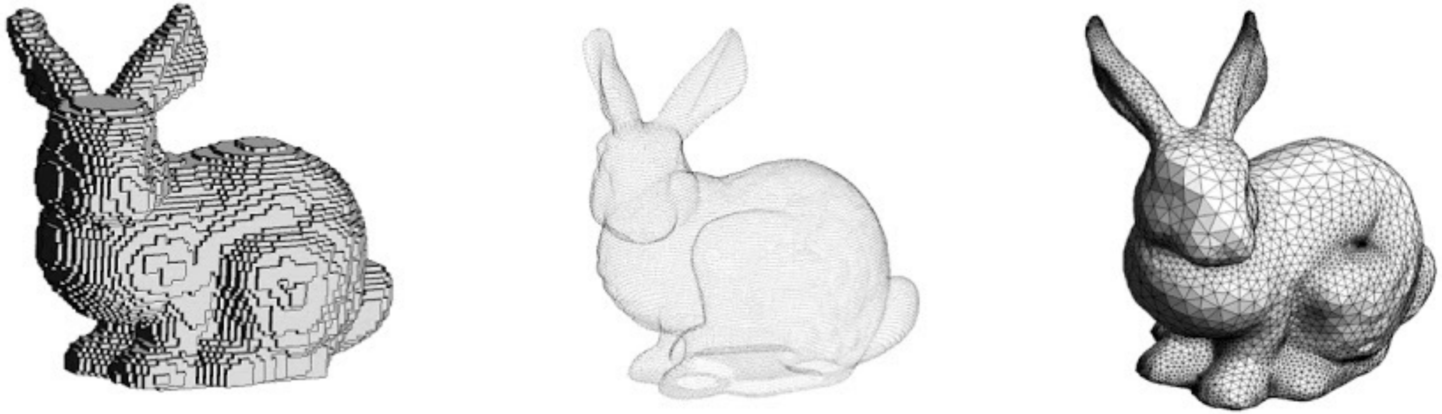
Explicit representation:

2) Mesh
3) Voxels
4) Point Cloud

Implicit representation:

5) Surface Representation (SDF)

3D Representations (Explicit)



	Voxel	Point cloud	Polygon mesh
Memory efficiency	Poor	Not good	Good
Textures	Not good	No	Yes
For neural networks	Easy	Not easy	Not easy

We adopt **polygon mesh** for its high potential

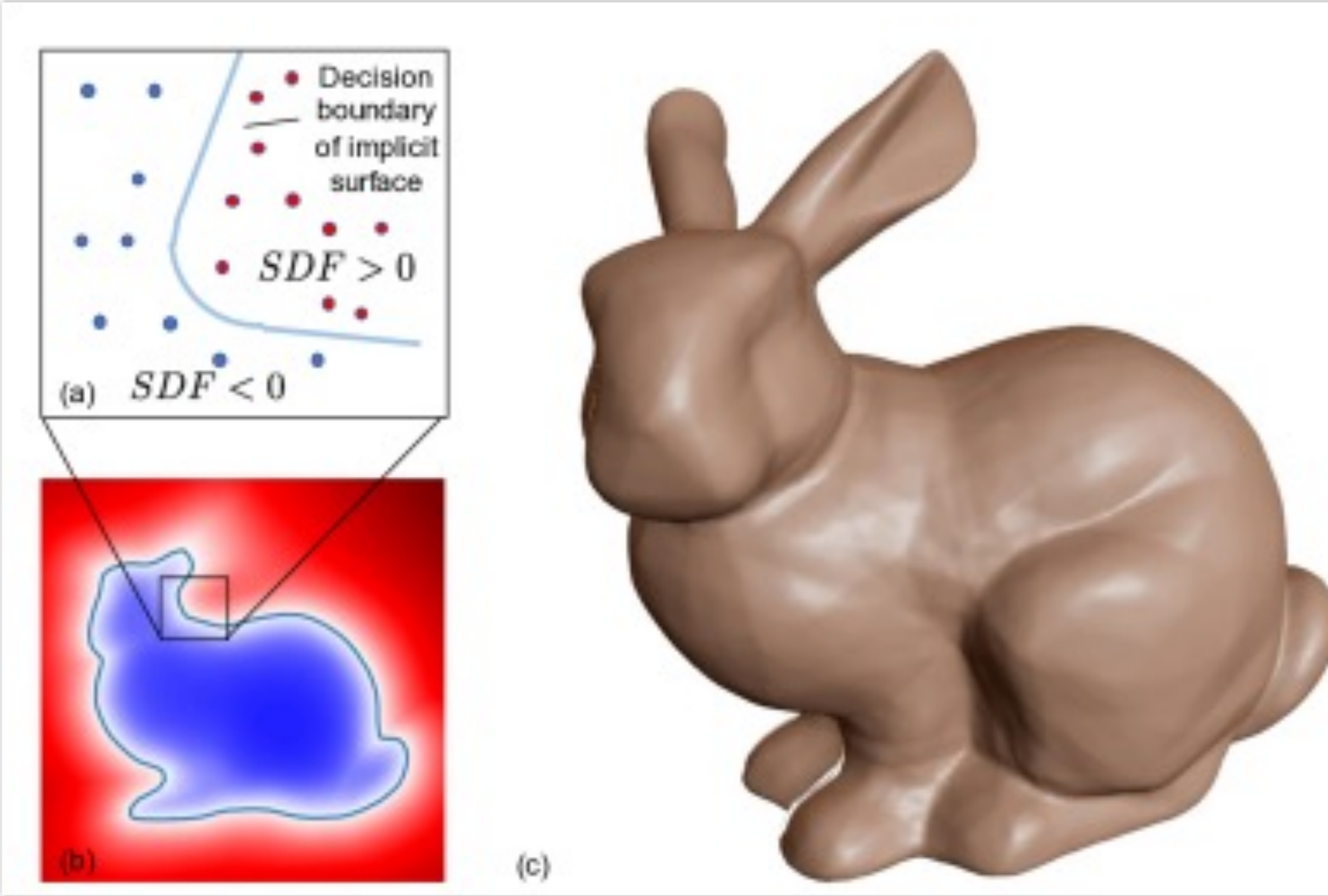
Images are from
http://cse.iitkgp.ac.in/~pb/research/3dpoly/3dpoly.html
http://waldyrious.net/learning-holography/pb-cgh-formulas.xhtm
http://www.cs.mun.ca/~omeruvia/philosophy/images/BunnyWire.gif

Surface Representation:
Signed Distance Function (SDF)
- implicit representation via level set

SDF(X) = 0, when X is on the surface.

SDF(X) > 0, when X is outside the surface

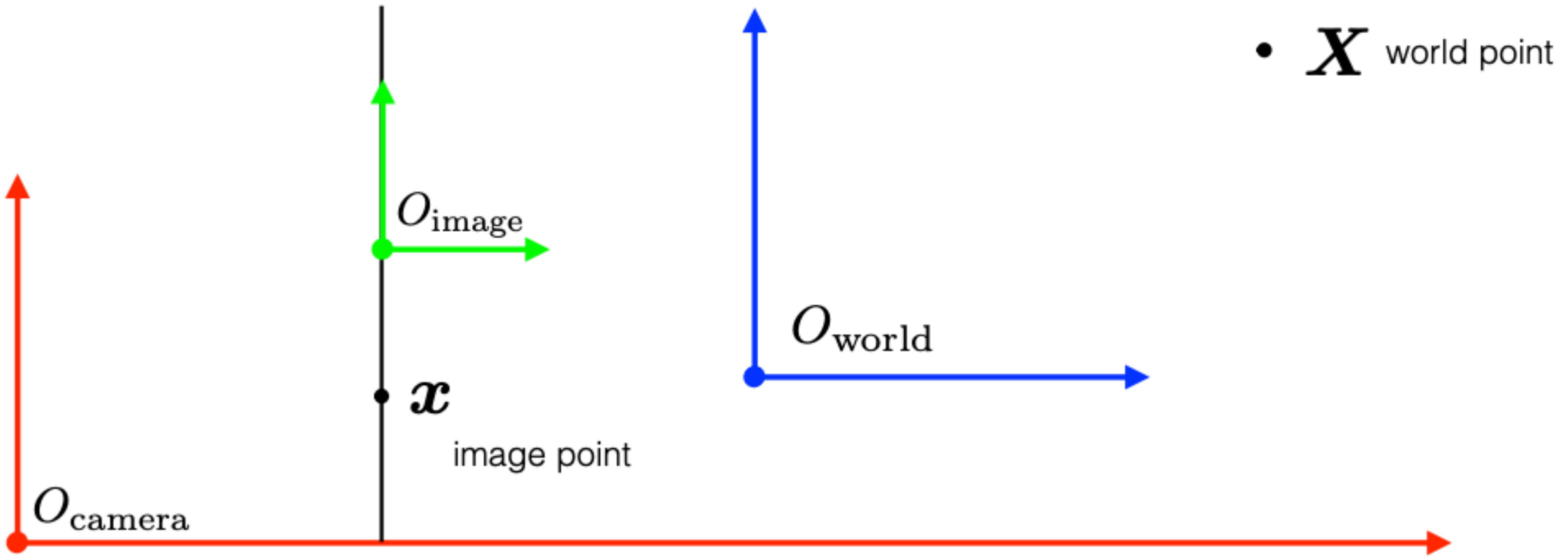
SDF(X) < 0, when X is inside the surface



Deep SDF: Use a neural network (co-ordinate based MLP) to represent the SDF function.

How do we define a camera model? – 3D object to 2D image

In general, there are **three different** coordinate systems…



so you need the know the transformations between them

How do we define a camera model? – 3D object to 2D image

- Relationship between image & camera coord. Systems.
- Camera Calibration matrix
- Camera Intrinsics
- Can be obtained from image meta data.

General mapping of a pinhole camera

$$\mathbf{P} = \mathbf{K}\mathbf{R}[\mathbf{I}| - \mathbf{C}]$$

(translate first then rotate)

- Relationship between world & camera coord. Systems.
- Camera Extrinsic
- Often known as 'Camera Pose Estimation/ Camera Localization problem'.

Another way to write the mapping

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

where

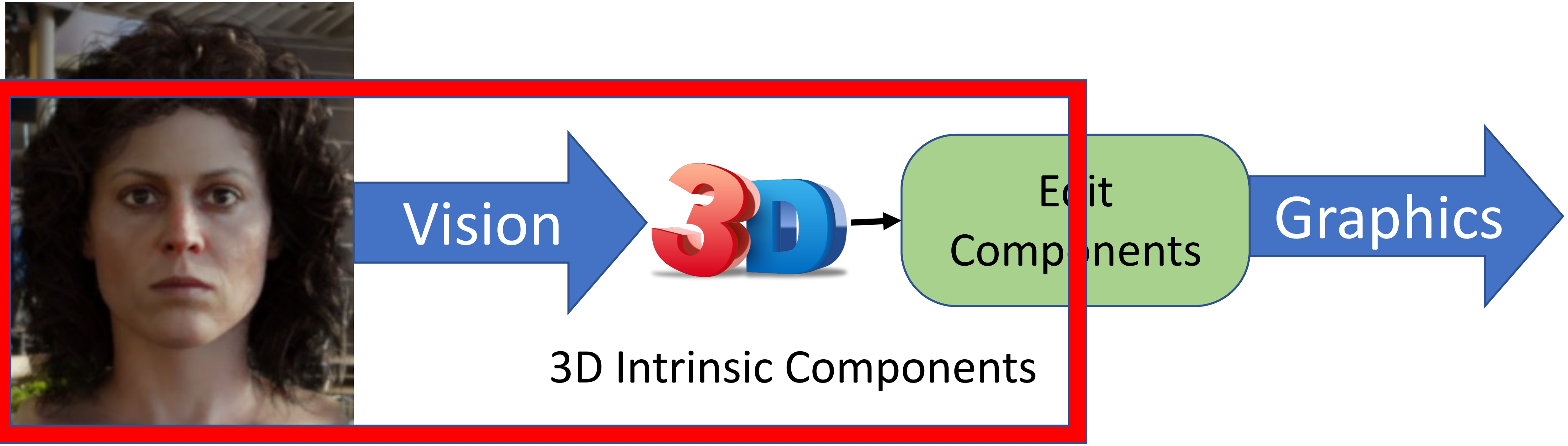
$$\mathbf{t} = -\mathbf{R}\mathbf{C}$$

(rotate first then translate)

3.2.1 Camera Parameterization

We use a perspective pinhole camera model and assume constant intrinsic camera parameters that have been calibrated in advance using established calibration procedures [114]. We denote the projective mapping for observation $i \in N_k$ and keyframe $k \in K$ as: $\pi_i^k : \mathbb{R}^3 \to \mathbb{R}^2$ and represent the extrinsic component (camera pose) of this mapping in world coordinates by a unit quaternion $\mathbf{q}_i^k \in SO(3)$ and a translation vector $\mathbf{t}_i^k \in \mathbb{R}^3$. Note that we use a redundant representation (i.e., the camera pose of an observation neighboring multiple keyframes is represented once per keyframe) to enable memory efficient optimization, one keyframe at a time, while enforcing consistency via additional soft constraints.

Computer Vision for 3D reconstruction



Current Image

New Image under different conditions

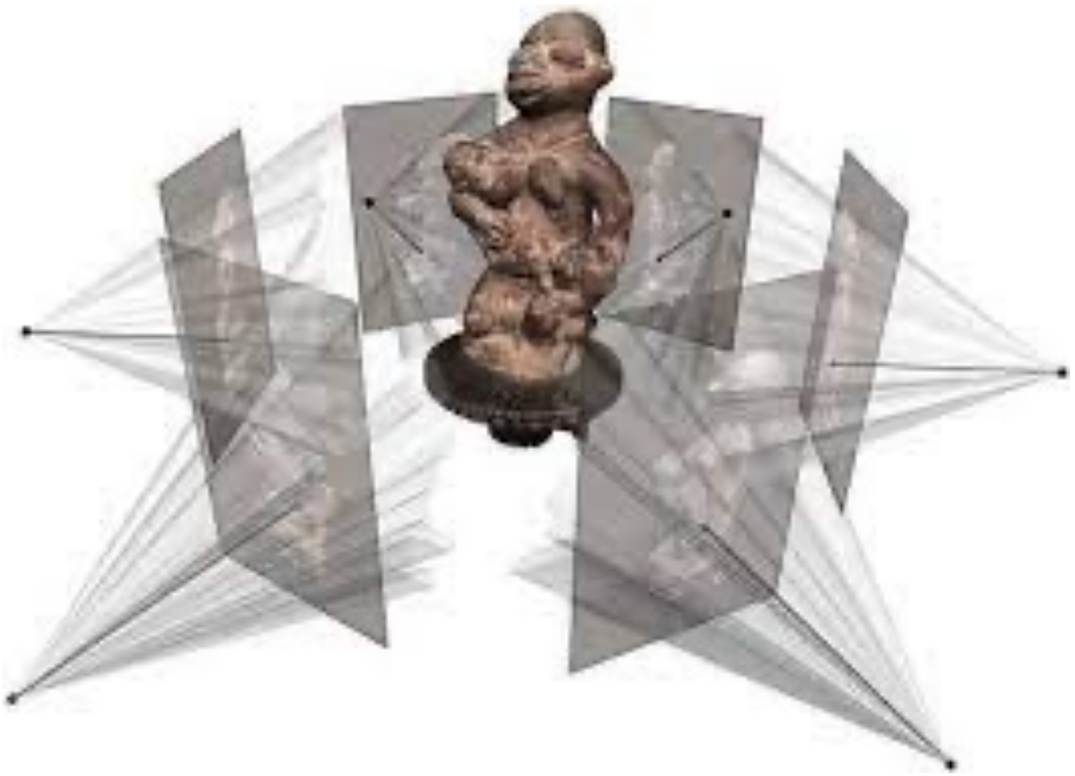
Explicit: Reconstruct 3D
(Introduction to Graphics Lectures)

Implicit: Neural Representation
(Generative Models Lectures

Change:

- Viewpoint
- Lighting
- Reflectance
- Background
- Attributes
- Many others…

Multi-View Stereo



Problem: Given a set of N images of an object, i1, i2, … iN, and a set of camera parameters P1,P2, … , PN, reconstruct the 3D object.

Classical approach and recent deep learning-based approach share a lot of similarity.

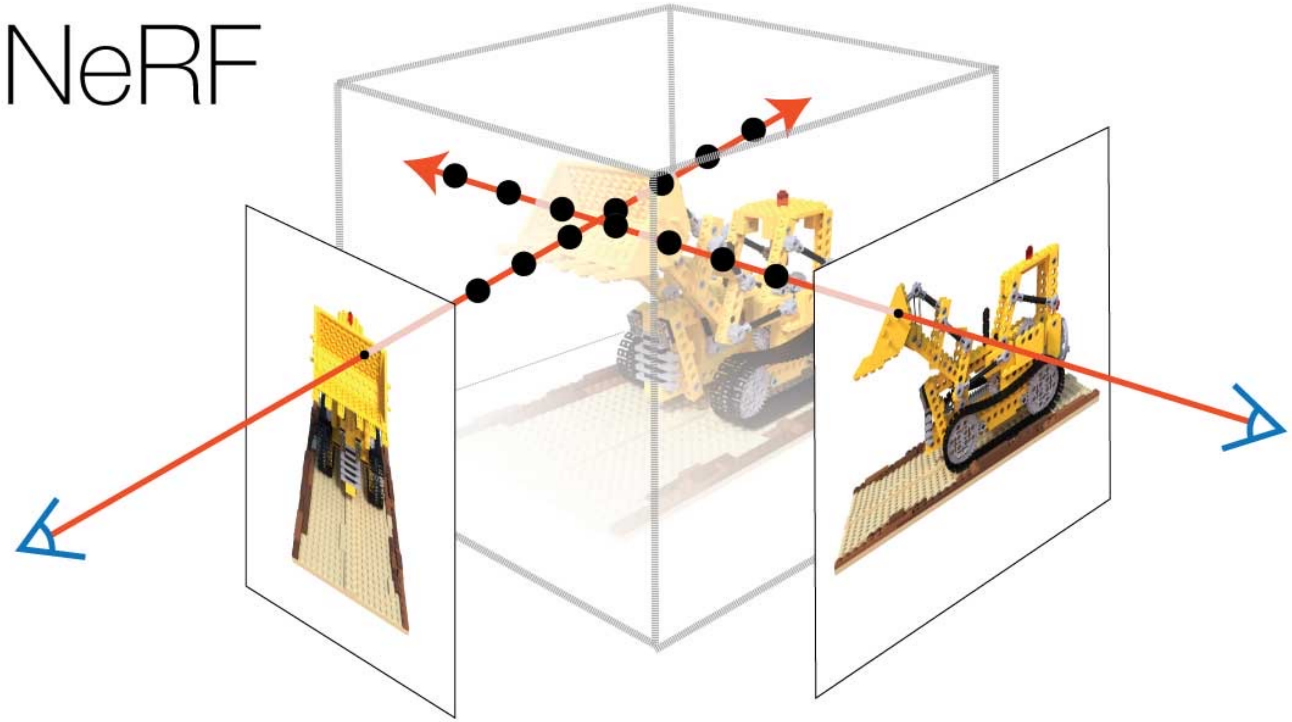
Structure from Motion (SfM)

Problem: Given a set of N images of an object, i1, i2, … iN, ~~and a set of camera parameters P1,P2, … , PN,~~ reconstruct the 3D object.

First find the set of camera parameters P1, P2, … , PN.

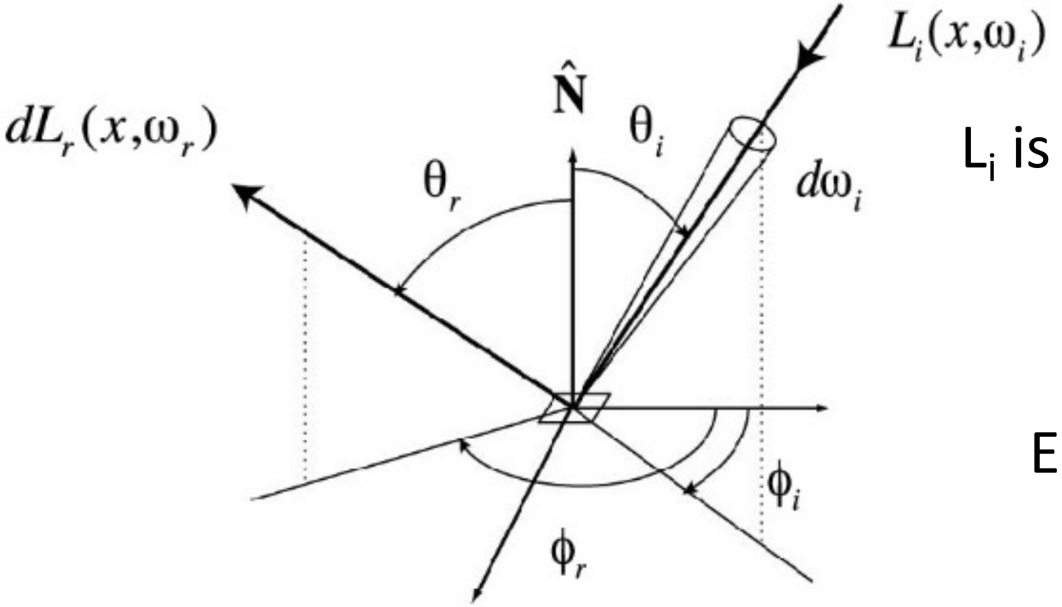
Where in the course will we encounter it?

… entire 2nd half of the course!



BRDF

Definition: The bidirectional reflectance distribution function (BRDF) represents how much light is reflected into each outgoing direction ω_r from each incoming direction



L_i is intensity of the light source.

Energy received by the surface is:

$$E_i = L_i cos\theta_i$$

$$f_r(\omega_i \to \omega_r) = \frac{\mathrm{d}L_r(\omega_r)}{\mathrm{d}E_i(\omega_i)} = \frac{\mathrm{d}L_r(\omega_r)}{L_i(\omega_i)\cos\theta_i\,\mathrm{d}\omega_i} \quad \left[\frac{1}{\mathrm{sr}}\right]$$

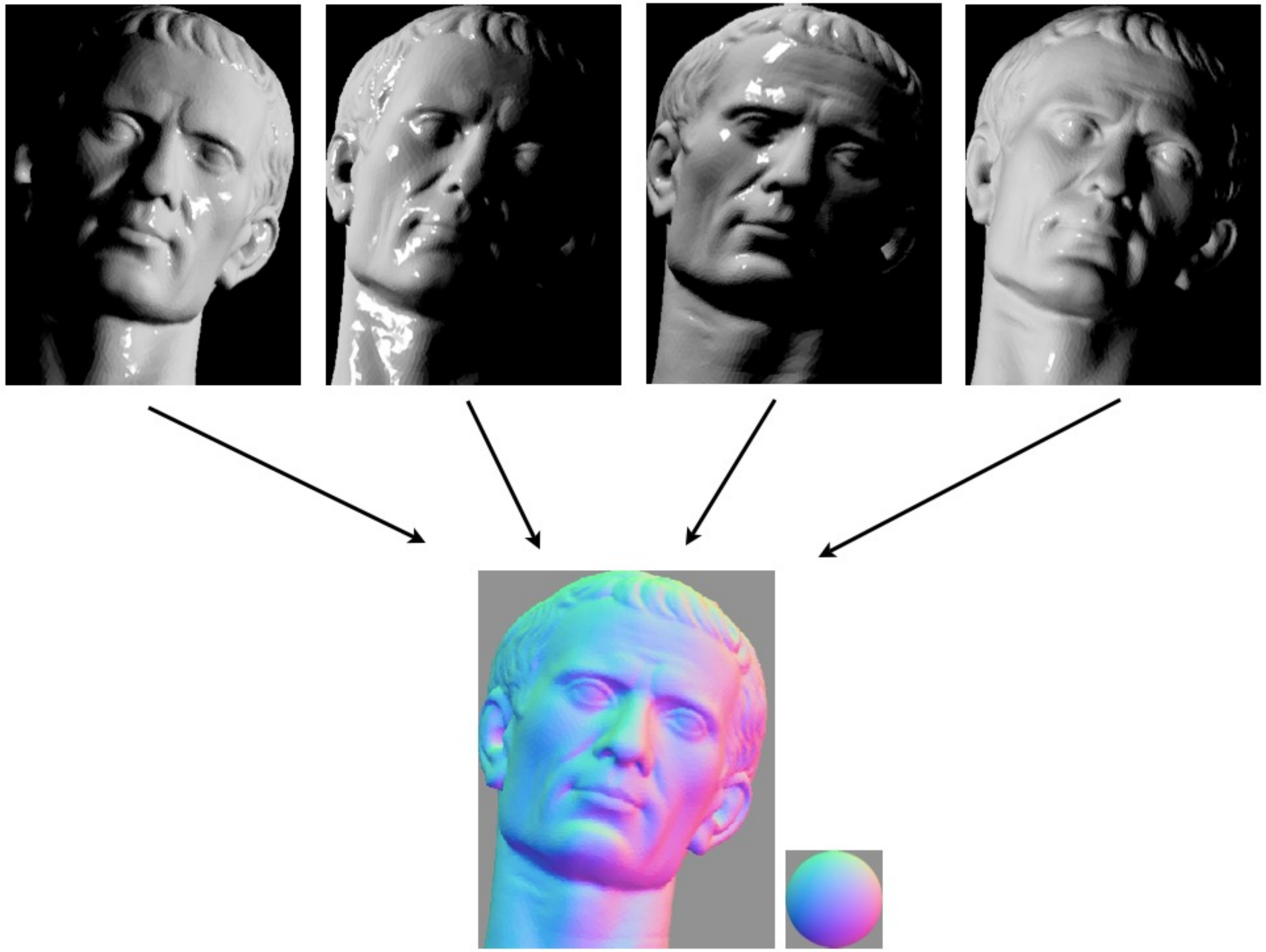
Types of BRDF

- Diffuse/Lambertian: light is reflected equally in all directions. Represented by Albedo.
- Shiny surfaces:
  - Spatially invariant (whole object has same amount of glossiness): Phong Reflectance model.
  - Spatially variant (glossiness varies for different part of the object): Microfacet model (Cook-Torrance model)

Other types:

- Isotropic vs anisotropic (metals)
- Subsurface scattering (human skin) (whiter skin -> more scattering, darker skin -> more specular reflections)

Photometric Stereo



Problem: Given N images of an object, i1, i2, … iN, captured with a fixed camera and N different lighting direction, reconstruct the surface geometry.

- Calculate surface normal.
- Integrate normal to depth.

Past Works assume:

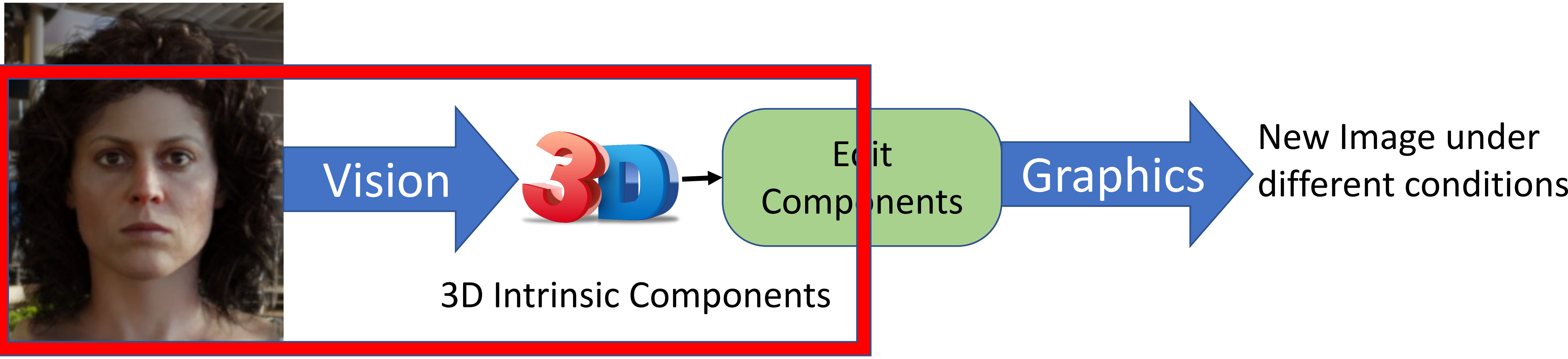
- Directional point light source
- Dark room
- Diffuse Reflection

Recent works do not require these assumptions + they also reconstruct BRDF!

Captured Images: Right

“Shape & Material Capture at Home”, Lichy, Wu, Sengupta, Jacobs, CVPR 2021
“Real-Time Light-Weight Near-Field Photometric Stereo”, Lichy, Sengupta, Jacobs, CVPR 2022

Computer Vision for 3D reconstruction



Current Image

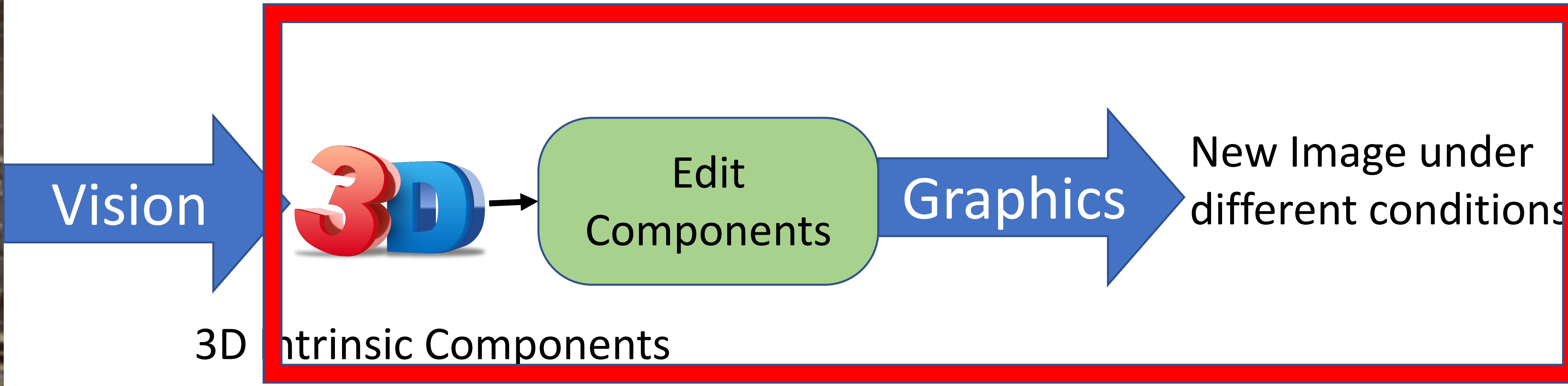
Explicit: Reconstruct 3D

- Reconstruct only geometry (Multiview Stereo, Structure from Motion)
- Inverse Rendering (geometry + BRDF)

We have 2 papers on Inverse Rendering with NeRF, where we will encounter these BRDF models again!

Computer Graphics for Rendering

Current Image



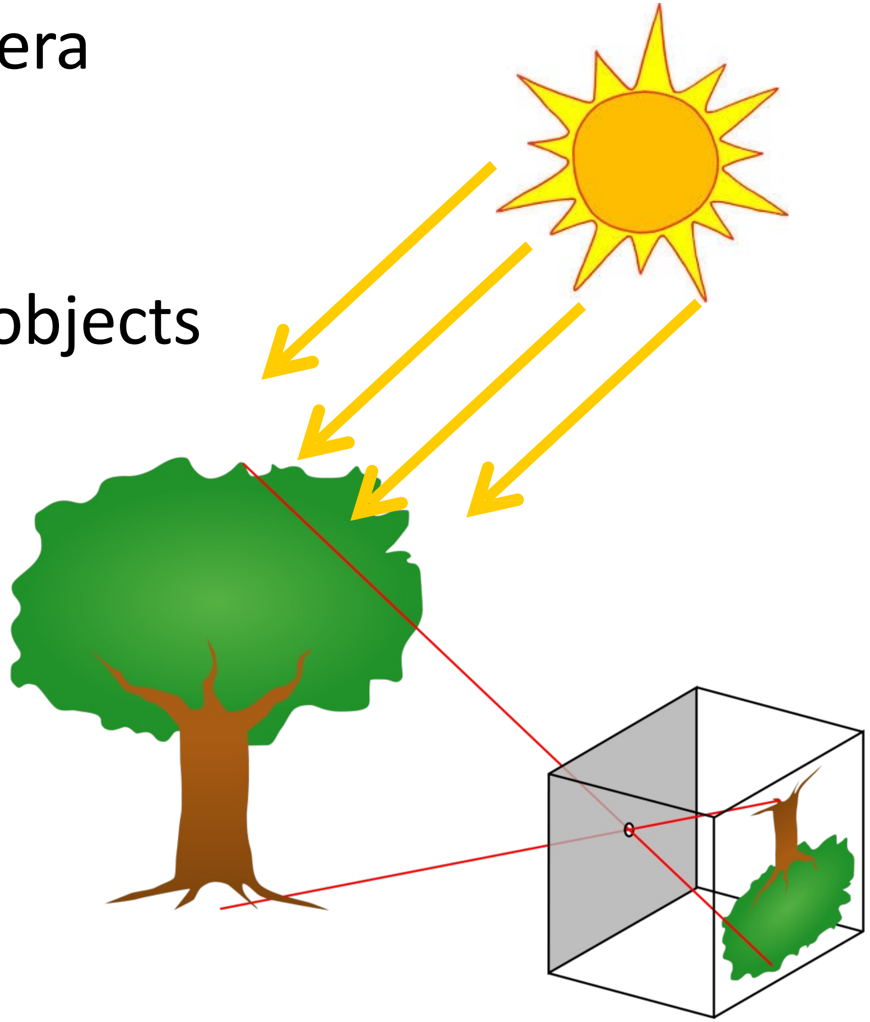
Explicit: Ray-Tracing, Image-based Rendering

Basics of Ray Tracing

Rendering: Reality

- Eye acts as pinhole camera

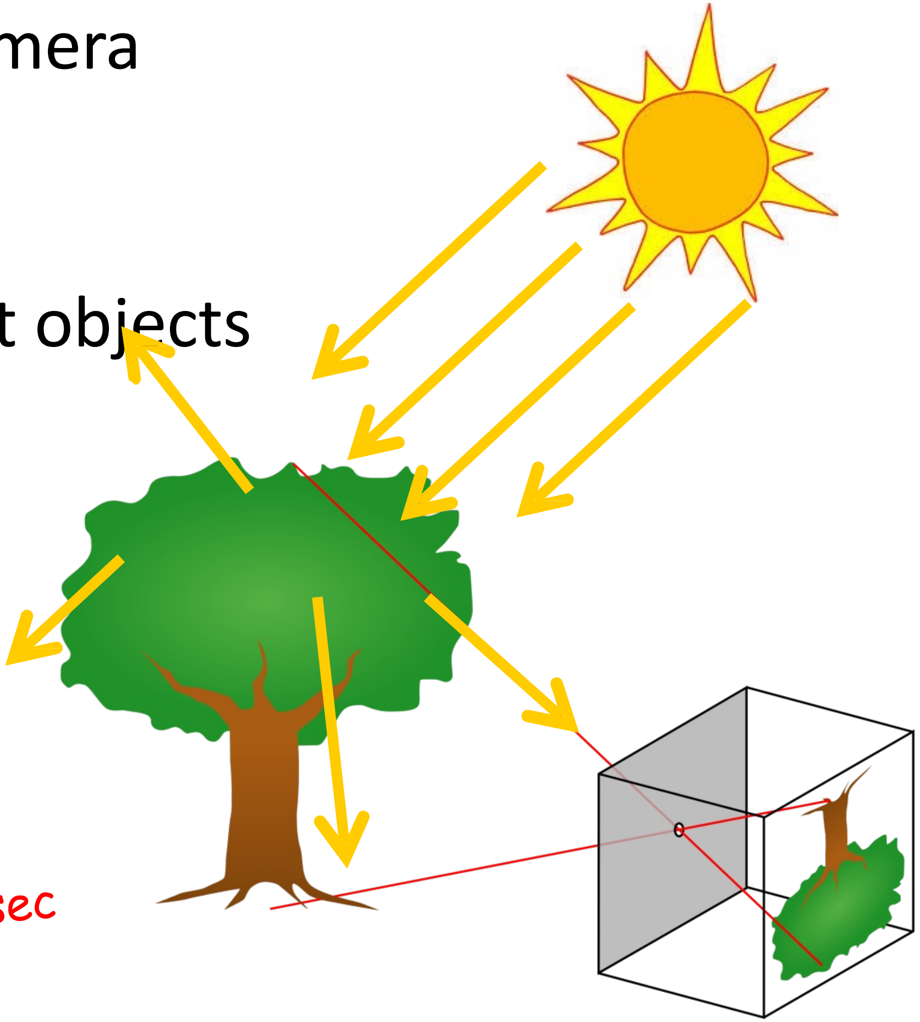
- Photons from light hit objects



Rendering: Reality

- Eye acts as pinhole camera

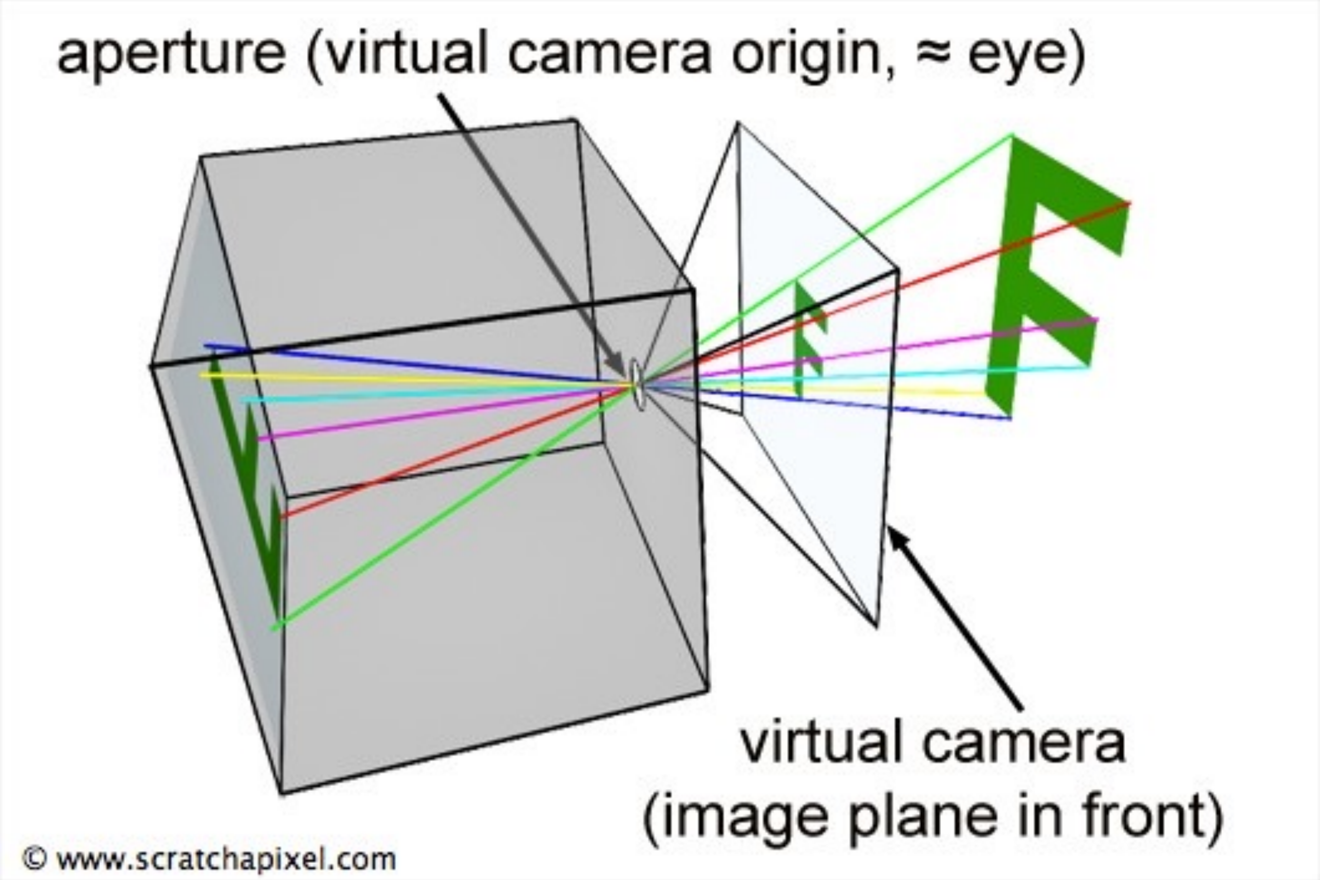
- Photons from light hit objects
- Bounce everywhere
- **Extremely** few
  - hit eye, form image



one lightbulb = 10^{19} photons/sec

Synthetic Pinhole Camera

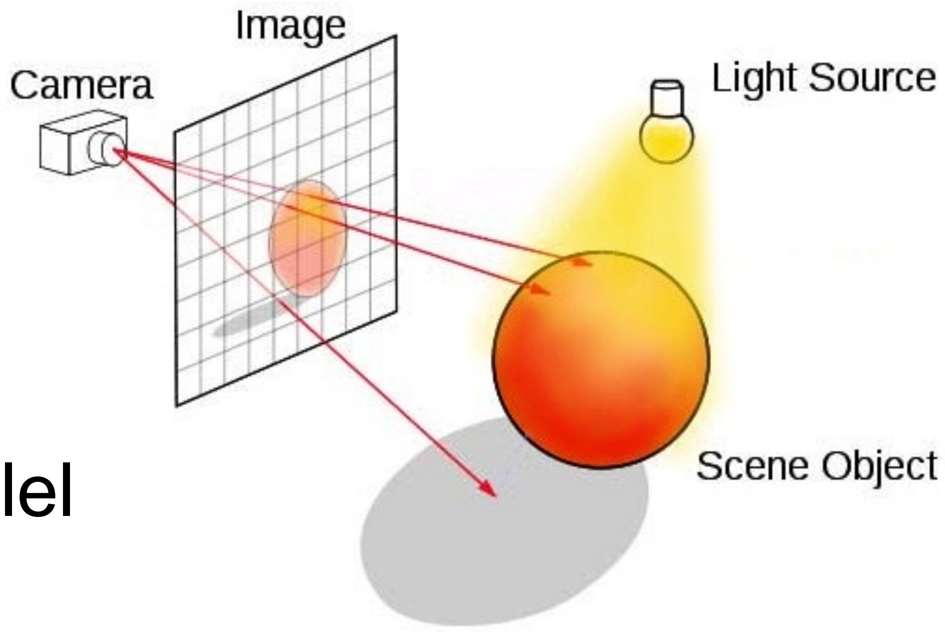
Useful abstraction: virtual image plane



Rendering: Ray Tracing

Reverse of reality

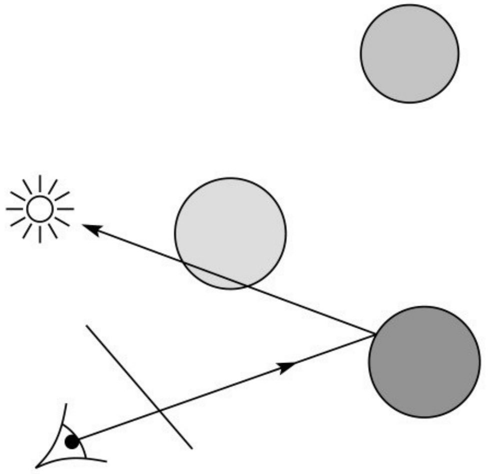
- Shoot rays through image plane
- See what they hit
- **Secondary** rays for:
  - Reflections
  - Shadows
- Embarrassingly parallel



Local Illumination

Simplifying assumptions:

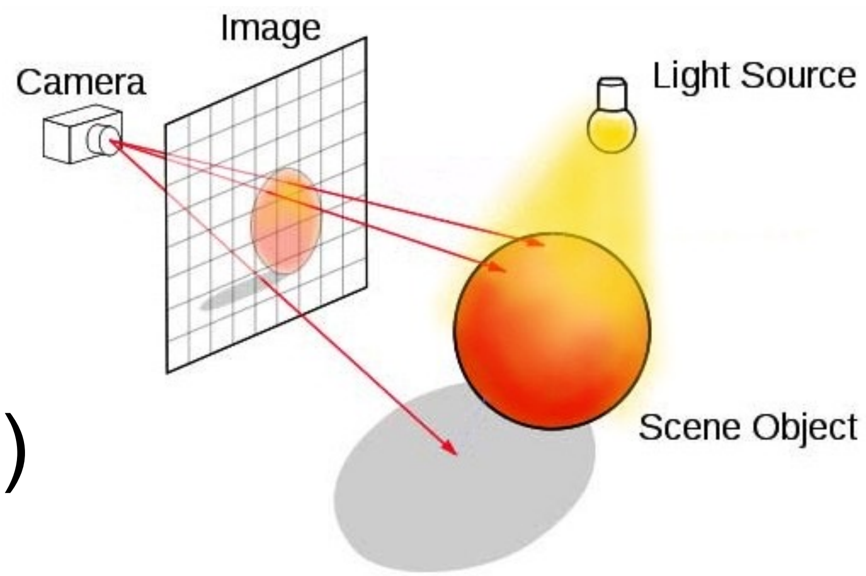
- Ignore everything except eye, light, and object
  - No shadows, reflections, etc



Big Hero 6 (2014)

Control (2019)

Why Slow?



Naïve algorithm: O(NR)

- R: number of rays
- N: number of objects

But rays can be cast in parallel

- each ray O(N)

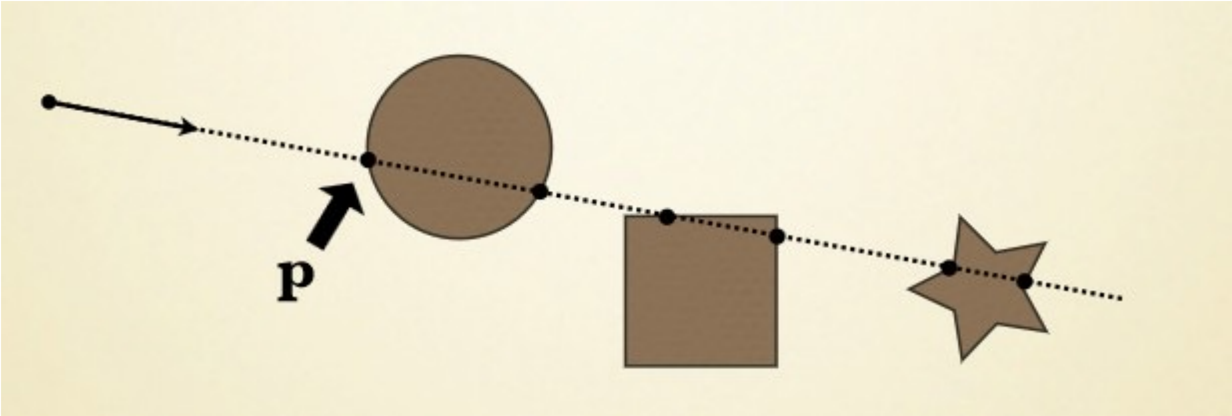
Basic Algorithm

For each pixel:

- Shoot ray from camera through pixel
- Find first object it hits
- If it hits something
  - Shade that pixel
  - Shoot secondary rays

Find First Object Hit By Ray?

Collision detection: find all values of t where ray hits object boundary



Take smallest **positive** value of t

Skipping: How to detect collision? How to do it fast and memory efficient?

- So we understand how to shoot rays and how to determine the intersection between ray and scene and choose the nearest point (this problem is often known as visibility test)

- Next question is:
    - How do we define lighting in a scene?
    - How do we assign shading/color to each pixel?
    - How do we find the effect of illumination at each 3D point in space?

- Next question is:
  - How do we define lighting in a scene?
  - How do we assign shading/color to each pixel?
  - How do we find the effect of illumination at each 3D point in space?

How do you define Lighting?

- HDR (High Dynamic Range) Environment Map
  - Basically a HDR panorama
  - Captured by placing a mirror ball
  - Awesome for rendering in Graphics
  - Bad for Inverse Rendering, as lots of parameter

- Spherical Harmonics
  - Effect of lighting on an object can be represent as a 27 dimensional vector (9 each for RGB channels)
  - Lighting is represented using spherical harmonics basis functions.
  - Popular in Computer Vision

- Many other representation exists
- Recent SOTA methods: approximate HDR Environment map as low-resolution (often 16x32) LDR Envrionment map

- Next question is:
  - How do we define lighting in a scene?
  - How do we assign shading/color to each pixel?
  - How do we find the effect of illumination at each 3D point in space?

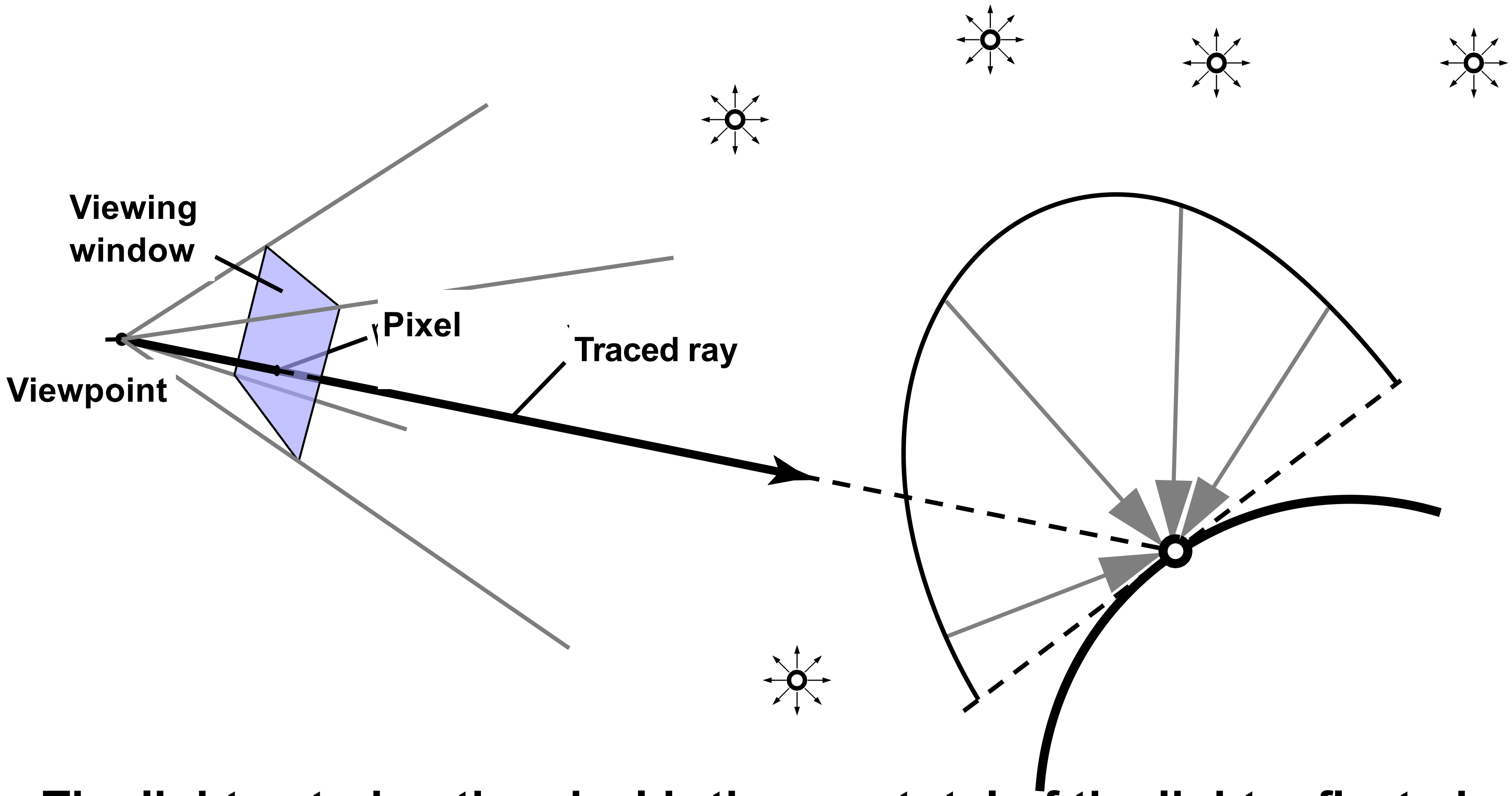
Global Illumination &

Path Tracing

Computer Graphics and Imaging

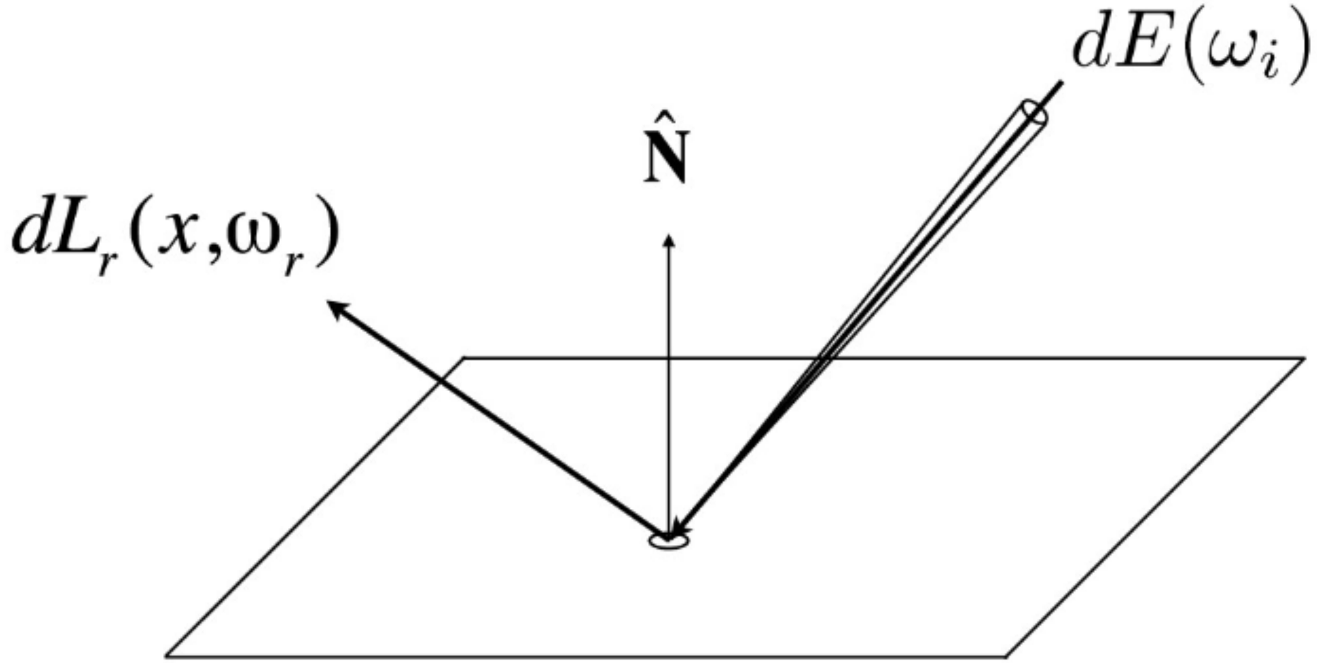
UC Berkeley CS184/284A

Ray Tracer Samples Radiance Along A Ray



The light entering the pixel is the sum total of the light reflected off the surface into the ray's (reverse) direction

Reflection at a Point

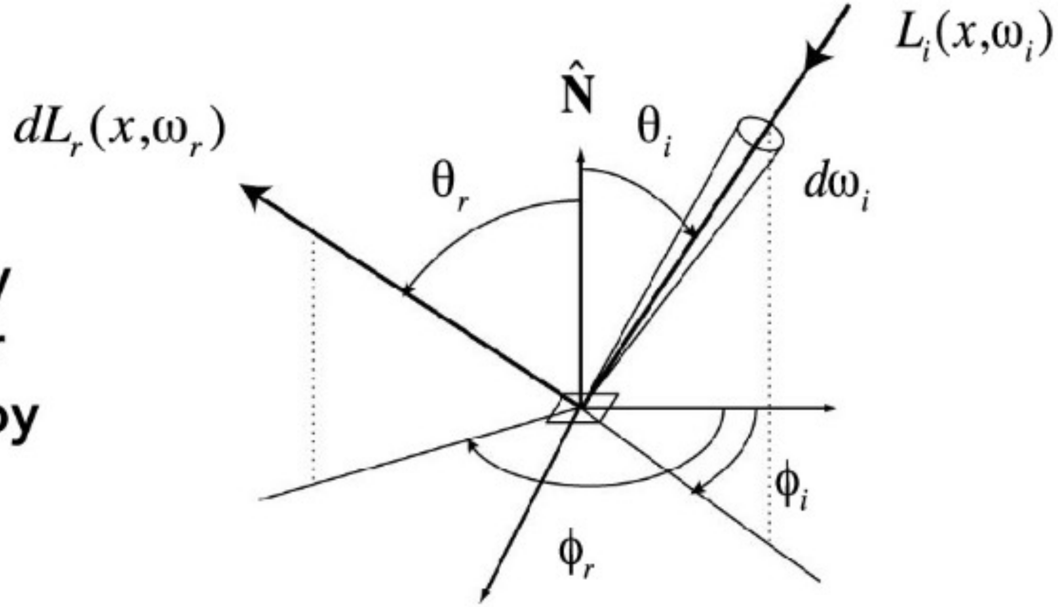


Differential irradiance incoming: $dE(\omega_i) = L(\omega_i)\, cos\, \theta_i\, d\omega_i$

Differential radiance exiting (due to $dE(\omega_i)$) $dL_r(\omega_r)$

BRDF

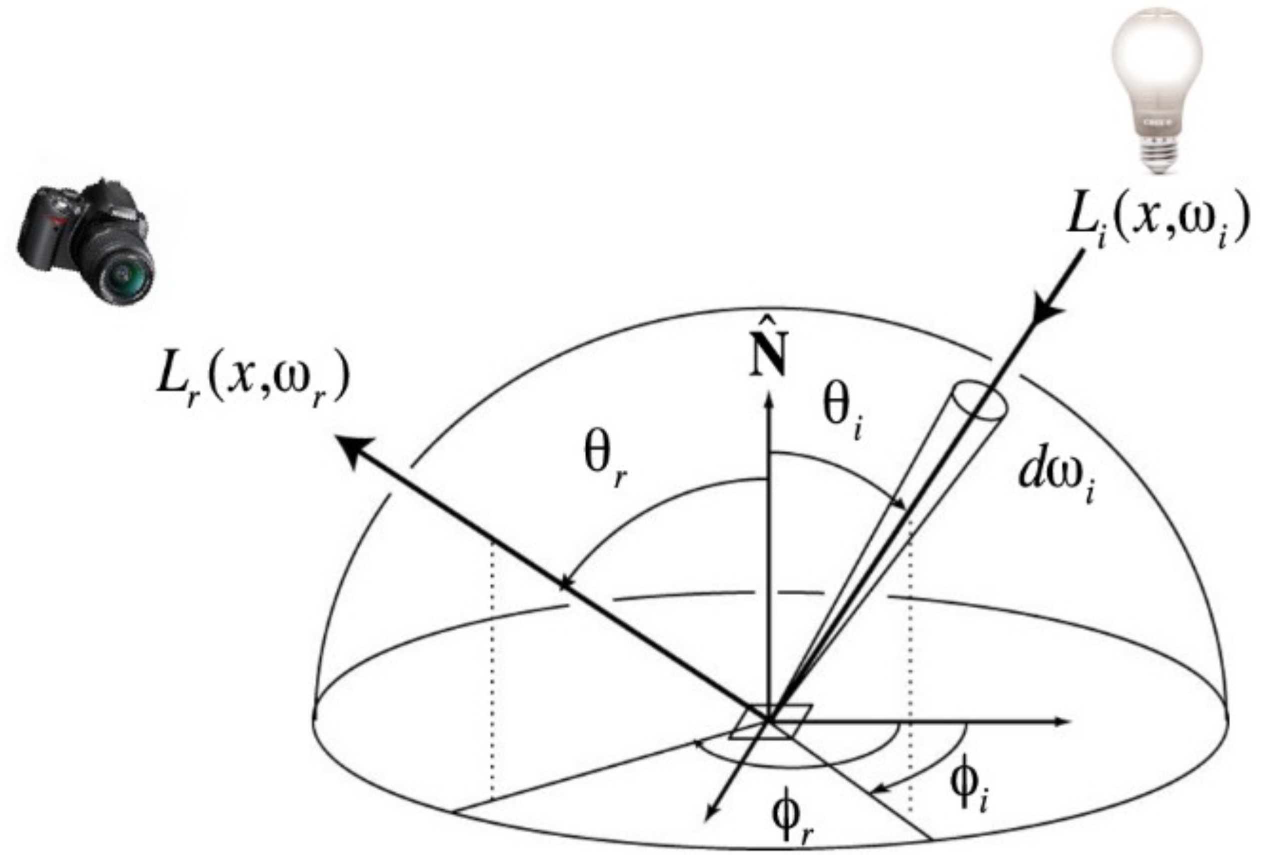
Definition: The bidirectional reflectance distribution function (BRDF) represents how much light is reflected into each outgoing direction ω_r from each incoming direction



NB: ω_i points away from surface rather than into surface, by convention.

$$f_r(\omega_i \to \omega_r) = \frac{\mathrm{d}L_r(\omega_r)}{\mathrm{d}E_i(\omega_i)} = \frac{\mathrm{d}L_r(\omega_r)}{L_i(\omega_i)\cos\theta_i\,\mathrm{d}\omega_i} \quad \left[\frac{1}{\mathrm{sr}}\right]$$

The Reflection Equation



How do you perform this integration?

How do you sample all incoming lighting directions?

$$L_r(\mathrm{p}, \omega_r) = \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_r)\, L_i(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

Solving the Reflection Equation

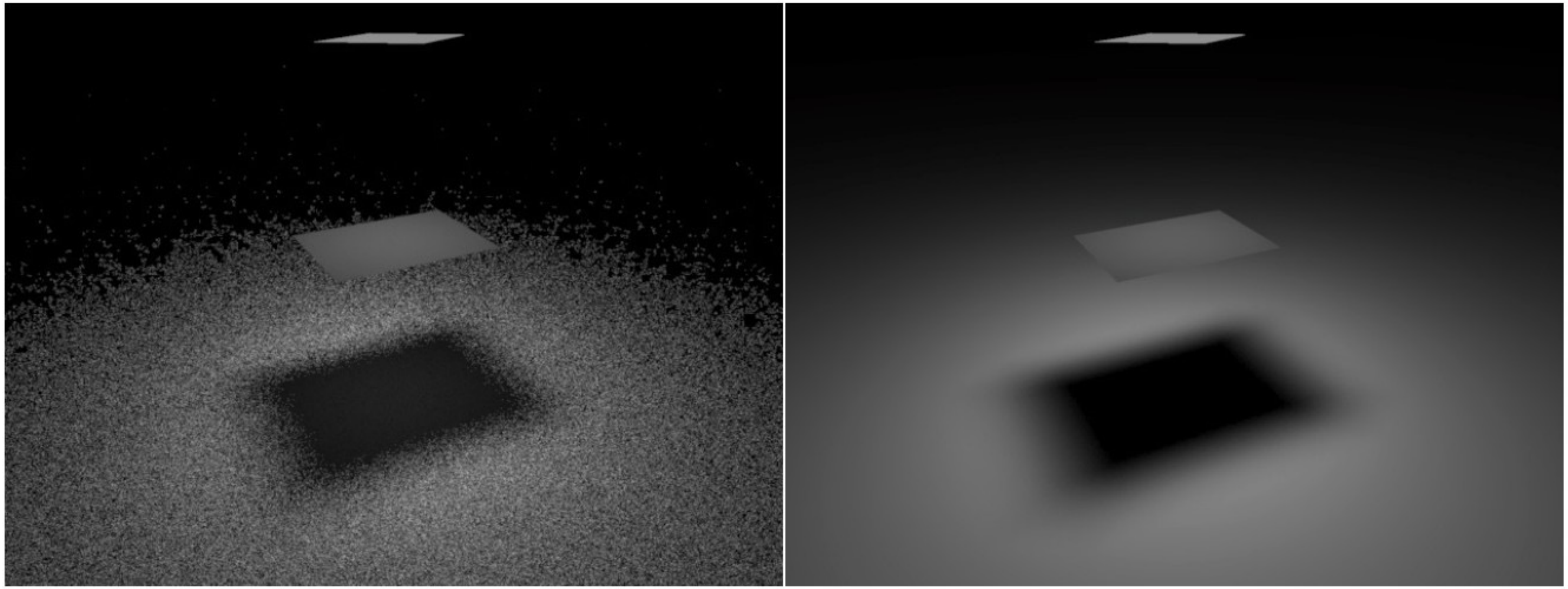
$$L_r(\mathrm{p}, \omega_r) = \int_{H^2} f_r(\mathrm{p}, \omega_i \to \omega_r)\, L_i(\mathrm{p}, \omega_i)\, \cos\theta_i \,\mathrm{d}\omega_i$$

Monte Carlo estimate:

- Generate directions ω_j sampled from some distribution $p(\omega)$
- Choices for $p(\omega)$
  - Uniformly sample hemisphere
  - Importance sample BRDF (proportional to BRDF)
  - Importance sample lights (sample position on lights)
- Compute the estimator

$$\frac{1}{N} \sum_{j=1}^{N} \frac{f_r(\mathrm{p}, \omega_j \to \omega_r)\, L_i(\mathrm{p}, \omega_j)\, \cos\theta_j}{p(\omega_j)}$$

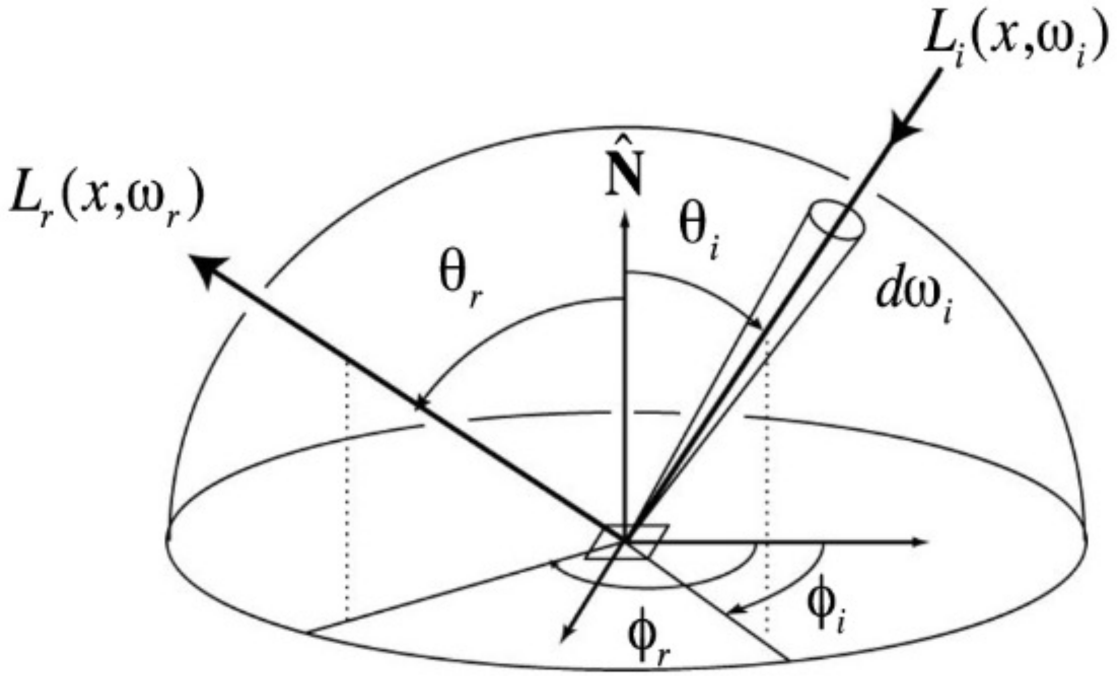
Recall: Hemisphere vs Light Sampling



Sample hemisphere uniformly **Sample points on light**

Global Illumination:
Deriving the Rendering Equation

Again: Reflection Equation

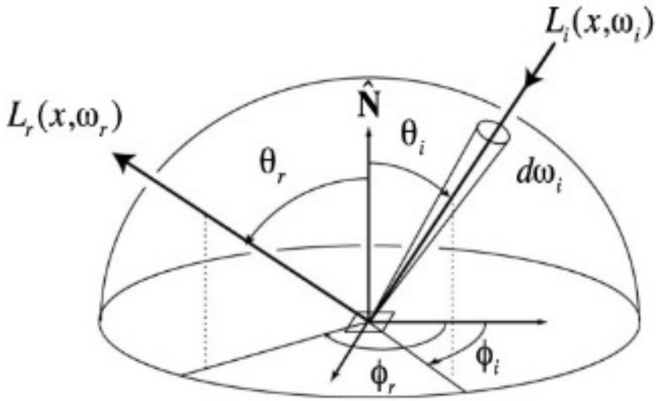


$$L_r(\mathrm{p},\omega_r) = \int_{H^2} f_r(\mathrm{p},\omega_i \to \omega_r)\, L_i(\mathrm{p},\omega_i)\, \cos\theta_i \,\mathrm{d}\omega_i$$

Challenge: This is Actually A Recursive Equation

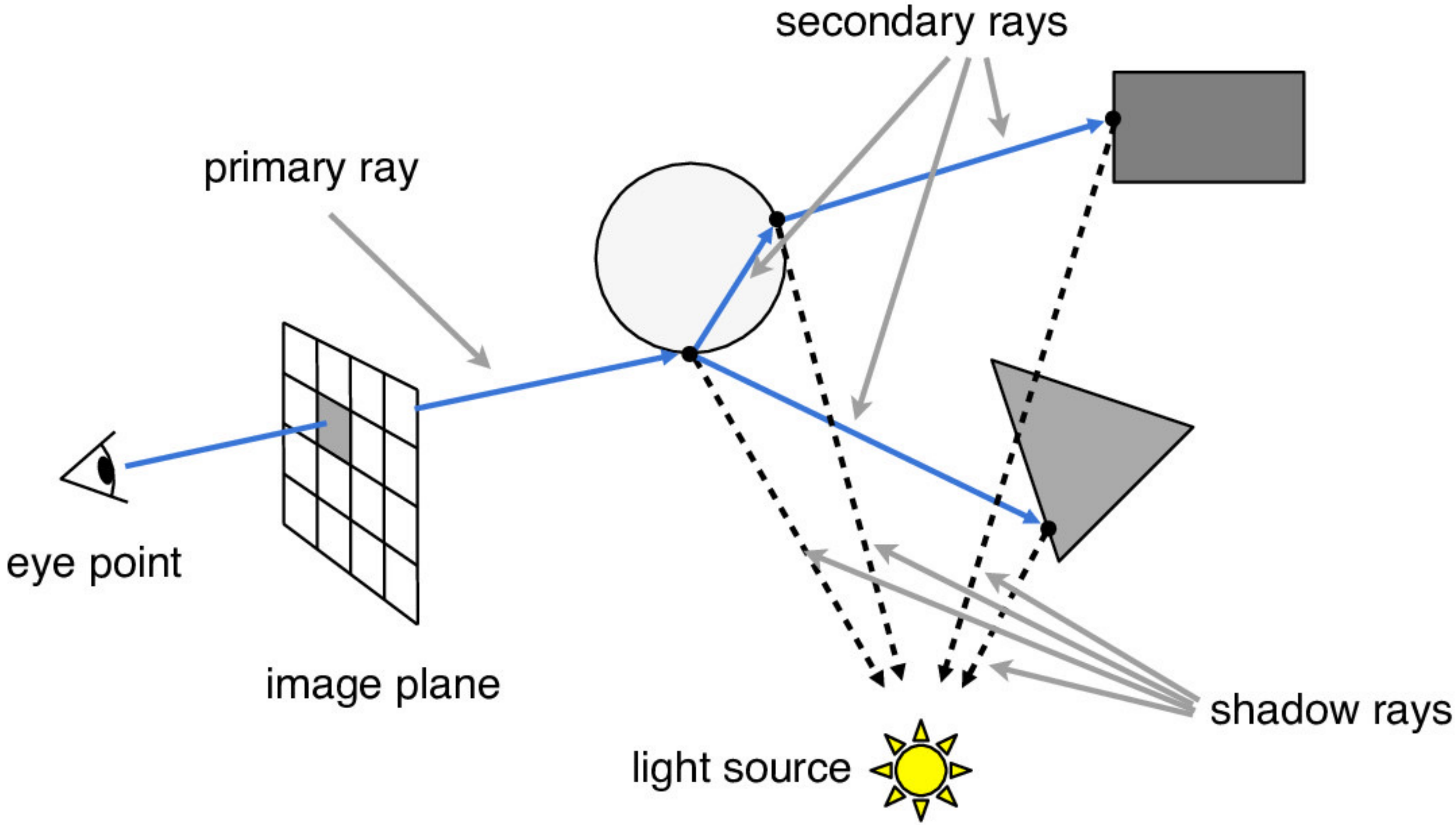
Reflected radiance depends on incoming radiance

$$\boxed{L_r(\mathrm{p}, \omega_r)} = \int_{H^2} f_r(\mathrm{p}, \omega_i \to \omega_r) \boxed{L_i(\mathrm{p}, \omega_i)} \cos\theta_i \, \mathrm{d}\omega_i$$



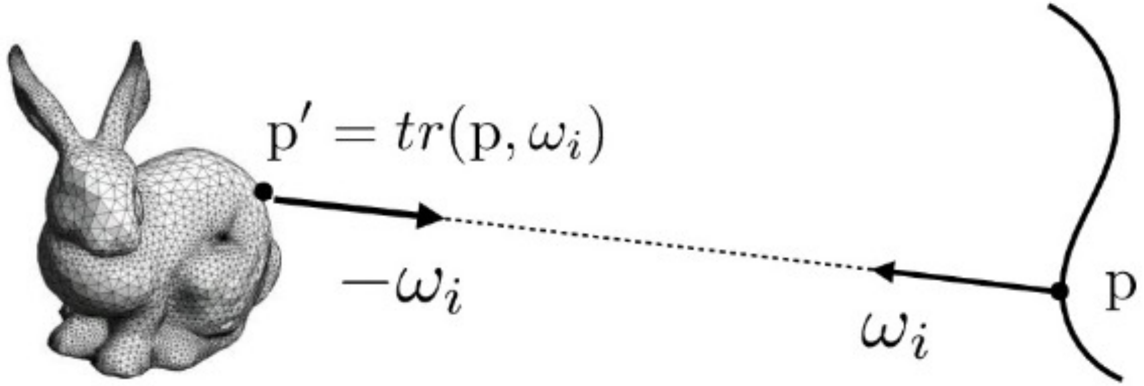
But incoming radiance depends on reflected radiance (at another point in the scene)

Recursive Ray Tracing



Transport Function & Radiance Invariance

Definition: the Transport Function, $tr(\mathrm{p}, \omega)$, returns the first surface intersection point in the scene along ray (p, ω)



Radiance invariance along rays: $L_o(tr(\mathrm{p}, \omega_i), -\omega_i) = L_i(\mathrm{p}, \omega_i)$

"Radiance arriving at p from direction ω_i is equal to the radiance leaving p' in direction $-\omega_i$"

The Rendering Equation

L_e is light emitted by the point p itself! (This term is 0 unless p is an emitter, one that emits light!)

Re-write the reflection equation:

$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, L_i(\mathrm{p}, \omega_i)\, \cos\theta_i\, \mathrm{d}\omega_i$$

Using the transport function: $L_i(\mathrm{p}, \omega_i) = L_o(tr(\mathrm{p}, \omega_i), -\omega_i)$

The Rendering Equation

$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, L_o(tr(\mathrm{p}, \omega_i), -\omega_i)\, \cos\theta_i\, \mathrm{d}\omega_i$$

Note: recursion is now explicit

How to solve?

Solving the rendering equation with Light Transport operator.

$$L_o(\mathrm{p}, \omega_o) = L_e(\mathrm{p}, \omega_o) + \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, L_o(tr(\mathrm{p}, \omega_i), -\omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

Using operators.
Operators = higher order functions.

$$L_o = L_e + (R \circ T)(L_o)$$

- **Reflection operator:**

$$R(g)(\mathrm{p}, \omega_o) \equiv \int_{H^2} f_r(\mathrm{p}, \omega_i \rightarrow \omega_o)\, g(\mathrm{p}, \omega_i)\, \cos\theta_i \, \mathrm{d}\omega_i$$

$$R(L_i) = L_o$$

- **Transport operator:**

$$T(f)(\mathrm{p}, \omega_o) \equiv f(tr(\mathrm{p}, \omega), -\omega)$$

$$T(L_o) = L_i$$

Define full one-bounce light transport operator: $K = R \circ T$

$$\boxed{L_o = L_e + K(L_o)}$$

Solving the Rendering Equation

- Rendering equation:

$$L = L_e + K(L)$$

$$(I - K)(L) = L_e$$

***L* is outgoing reflected**

- Solution desired:

$$L = (I - K)^{-1}(L_e)$$

- How to solve?

Solution Intuition

For scalar functions, recall:

$$\frac{1}{1-x} = 1 + x + x^2 + x^3 + \cdots$$
$$\text{converges for } -1 < x < 1$$

Similarly, for operators, it is true that

$$(I-K)^{-1} = \frac{1}{I-K} = I + K + K^2 + K^3 + \cdots$$

(Neumann series)

$$\text{converges for } ||K|| < 1$$

where $||K|| < 1$ means that the "energy" of the radiance function decreases after applying K. This is intuitively true for valid scene models based on energy dissipation (though not trivial to prove, see Veach & Guibas).

Rendering Equation Solution

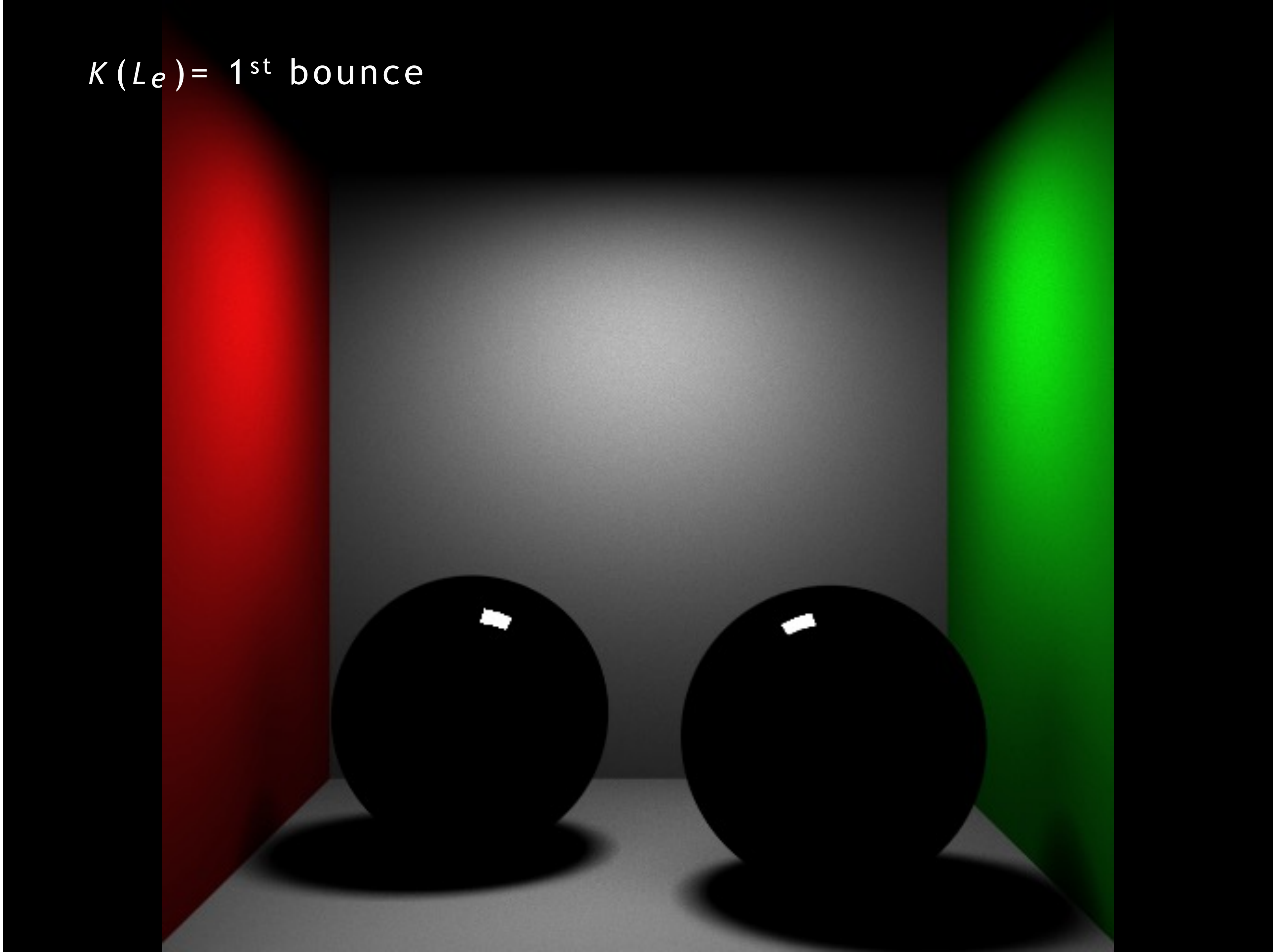
$$L = (I - K)^{-1}(L_e)$$
$$= (I + K + K^2 + K^3 + \cdots)(L_e)$$
$$= L_e + K(L_e) + K^2(L_e) + K^3(L_e) + \cdots$$

(arrows pointing to each term: L_e — **Emitted**; $K(L_e)$ — **1-bounce**; $K^2(L_e)$ — **2-bounce**; $K^3(L_e)$ — **3-bounce**)

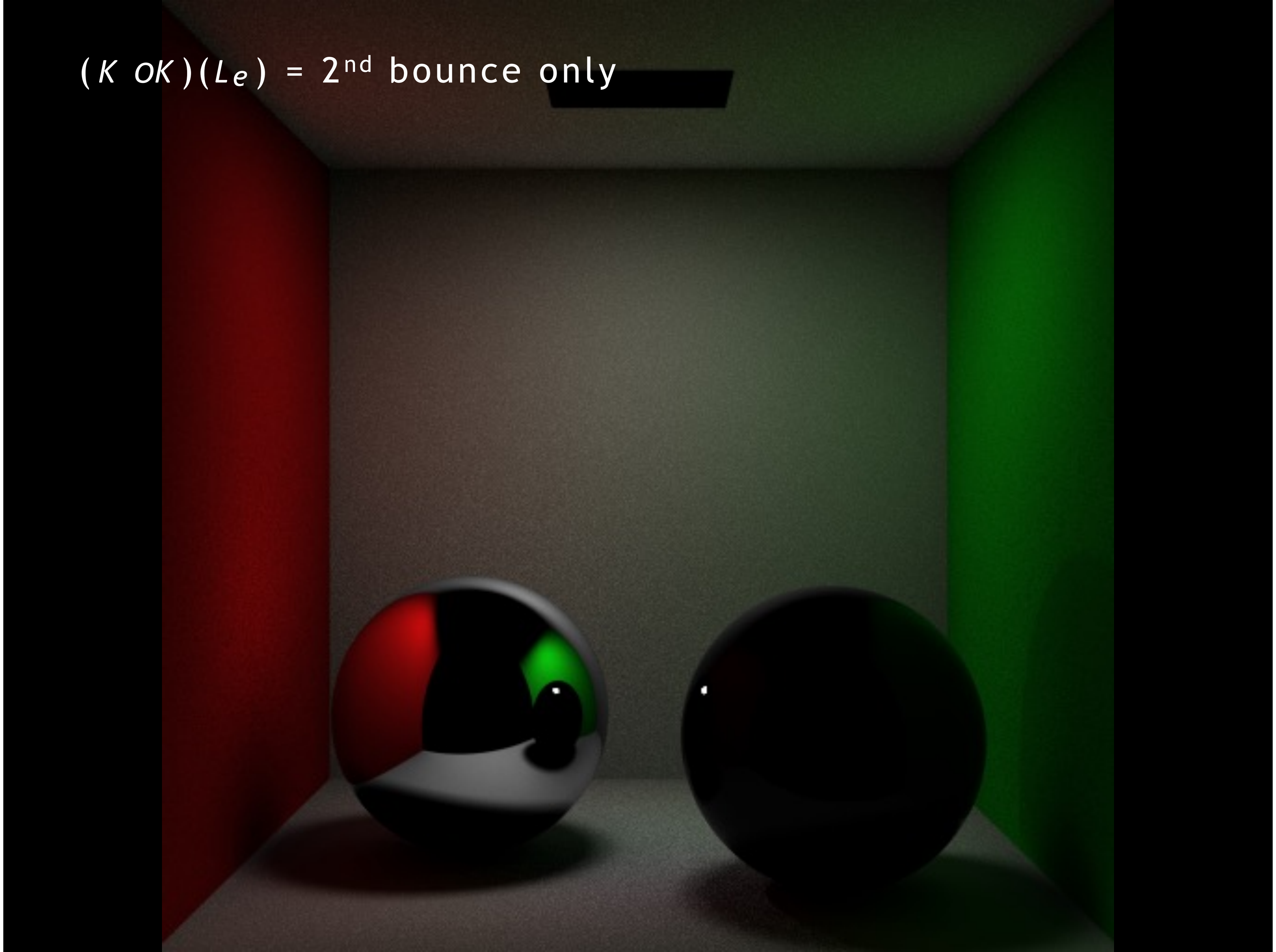
Intuitive: Sum of successive bounces of light

This calculates the steady-state surface light field over the scene.

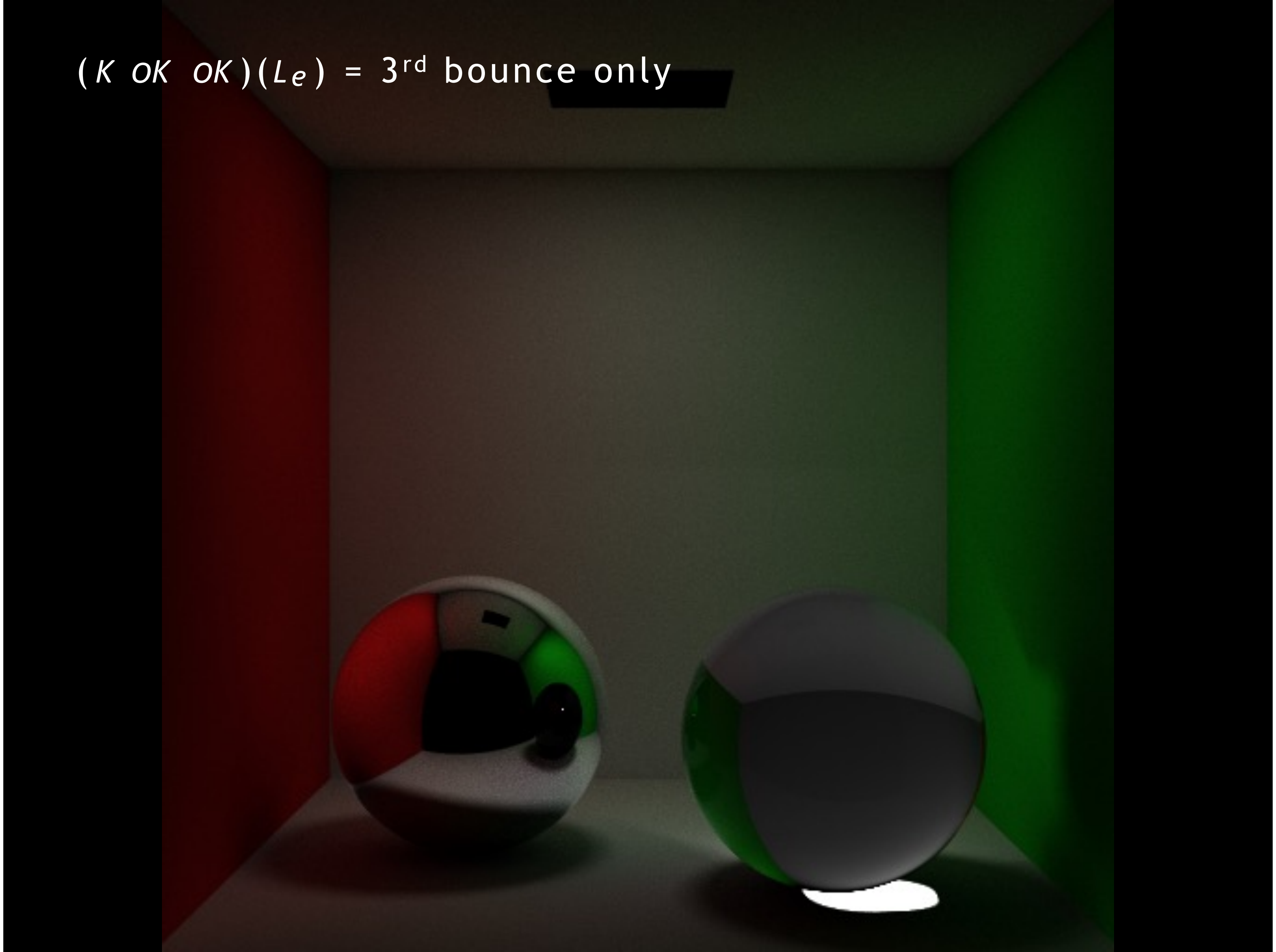
$K(L_e) =$ 1st bounce



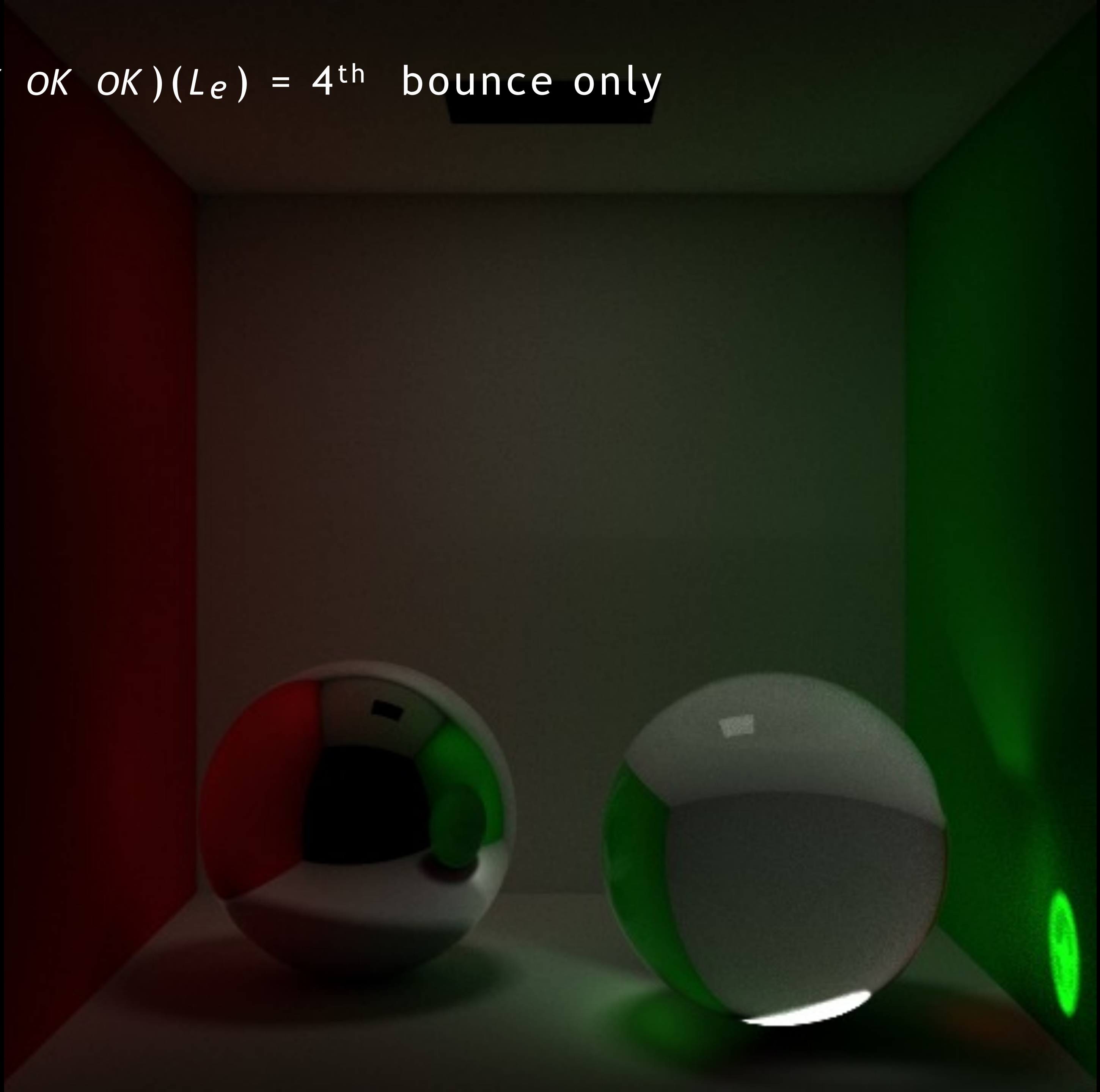
$(K \; OK)(L_e) = 2^{nd}$ bounce only



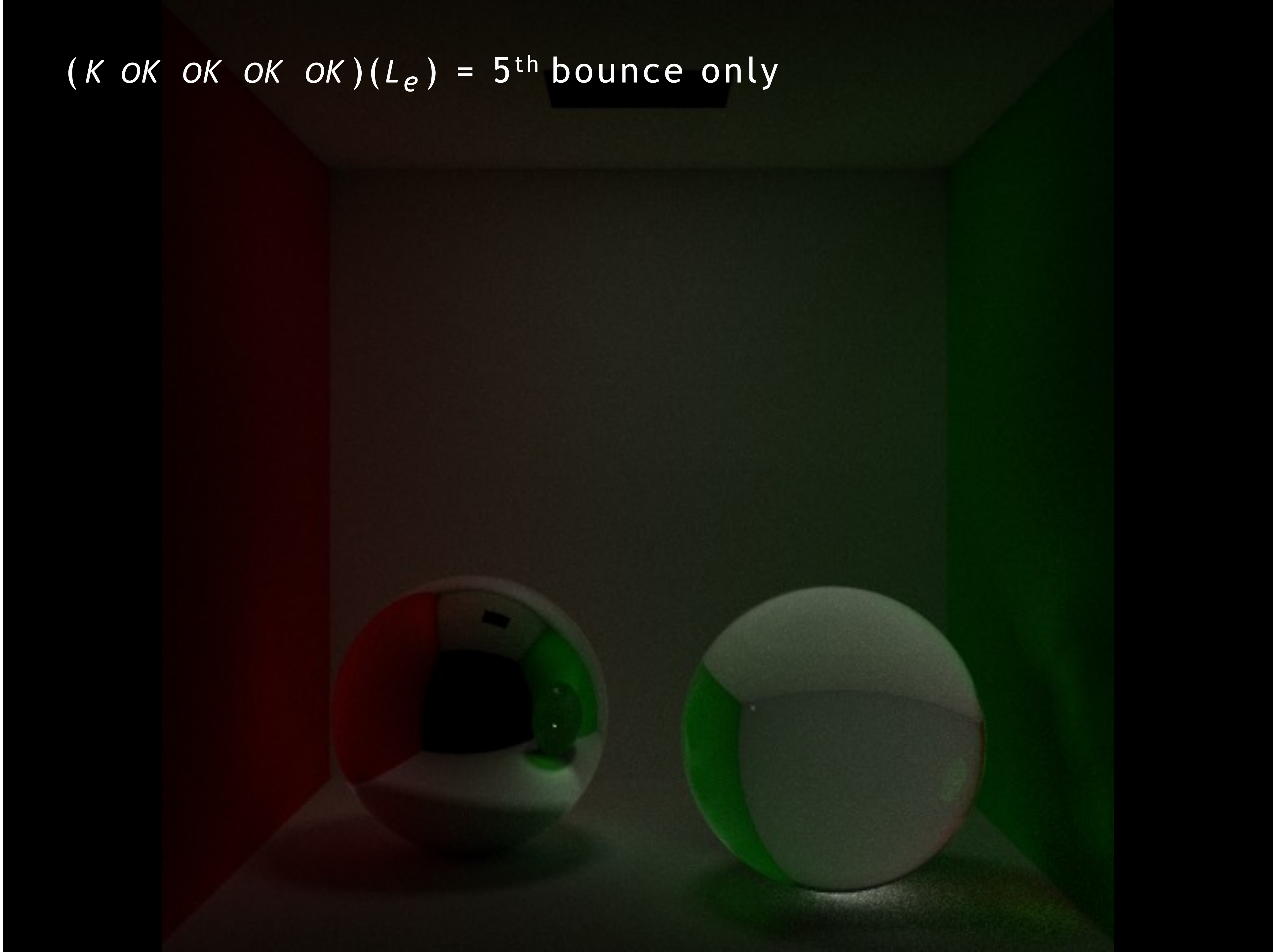
$(K\ OK\ OK)(L_e)$ = 3rd bounce only



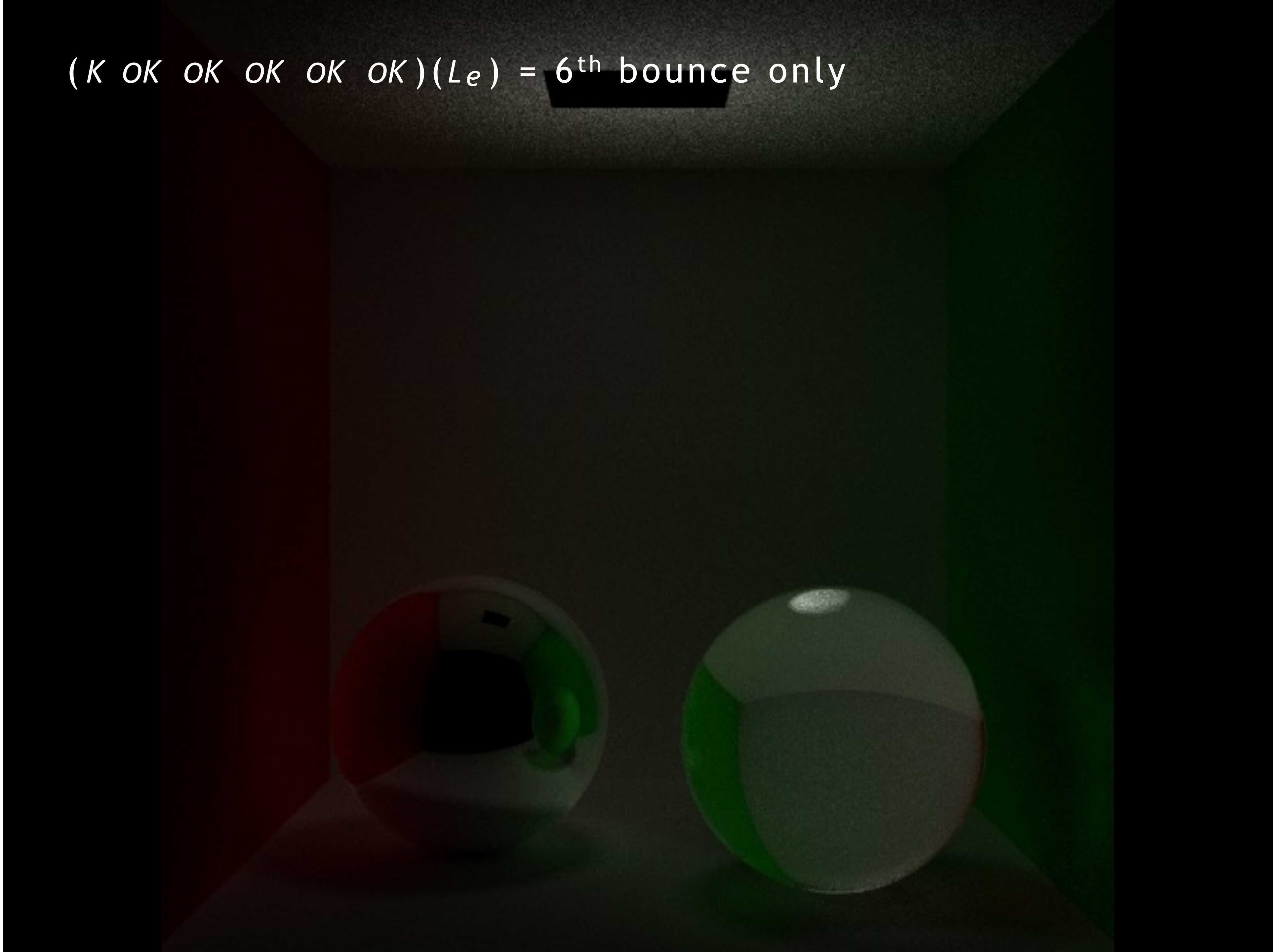
$(K \; OK \; OK \; OK)(L_e) = 4^{th}$ bounce only

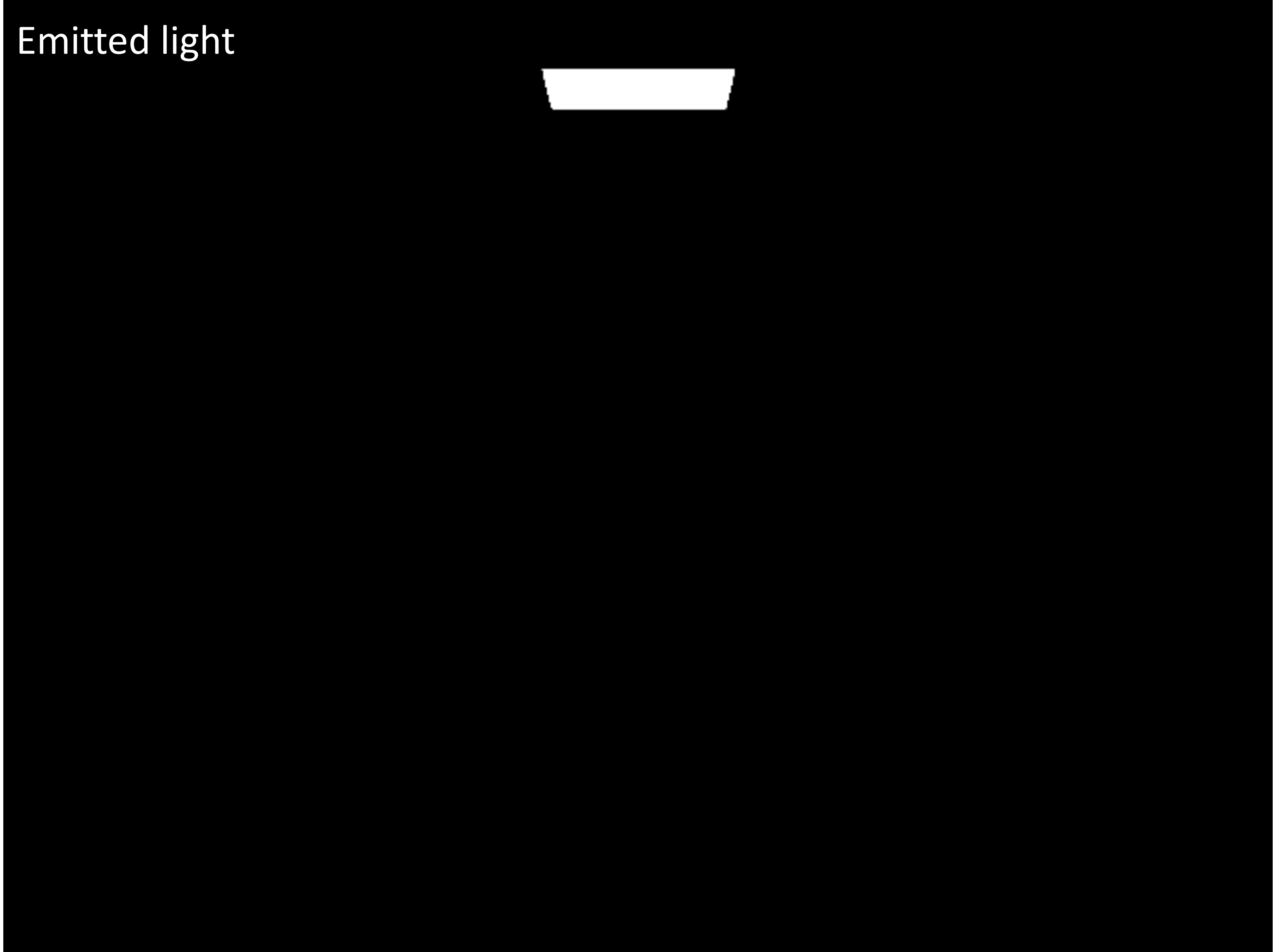


$(K\ OK\ OK\ OK\ OK)(L_e)$ = 5^{th} bounce only

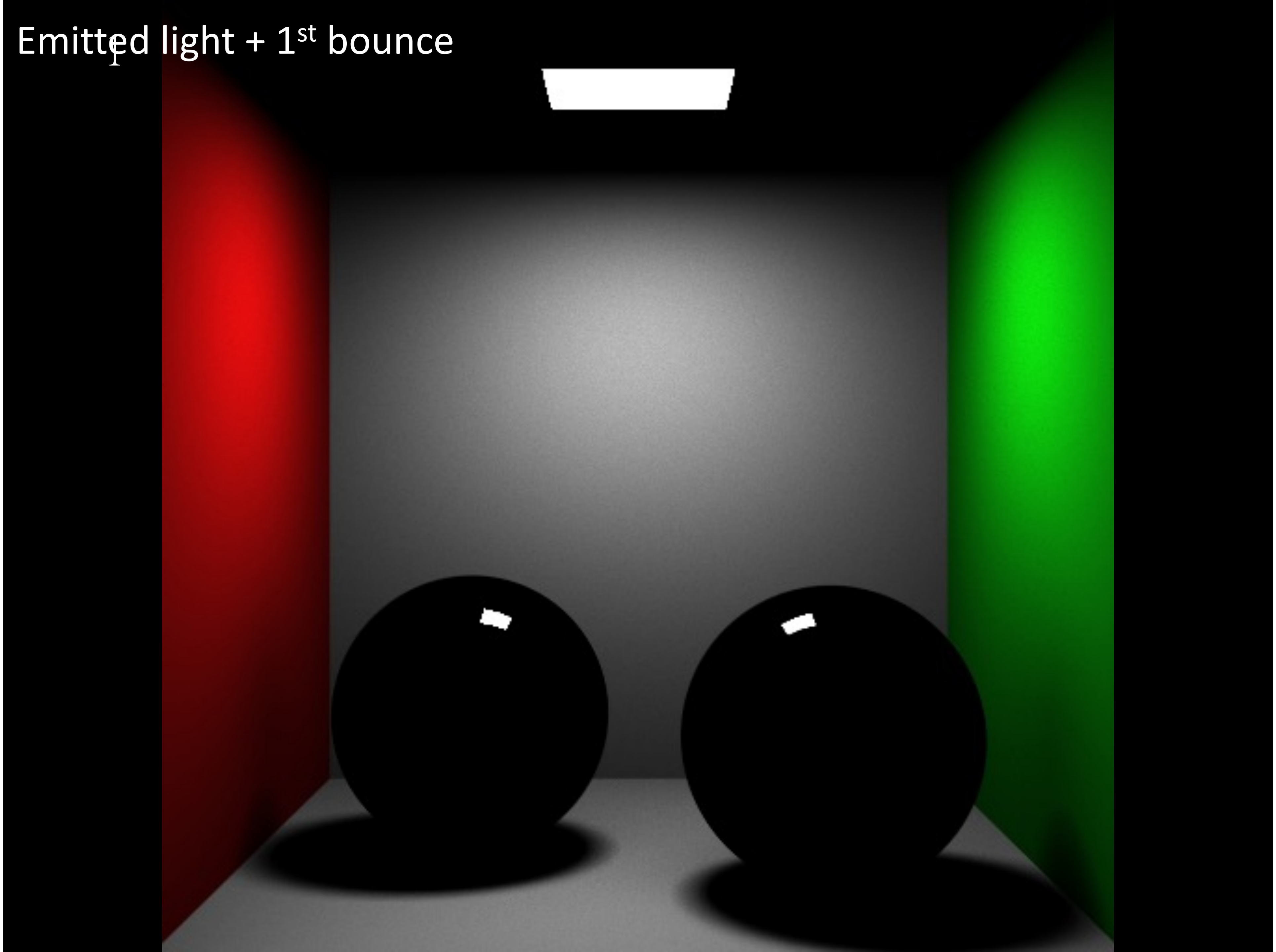


$(K\ OK\ OK\ OK\ OK\ OK)(L_e)$ = 6^{th} bounce only

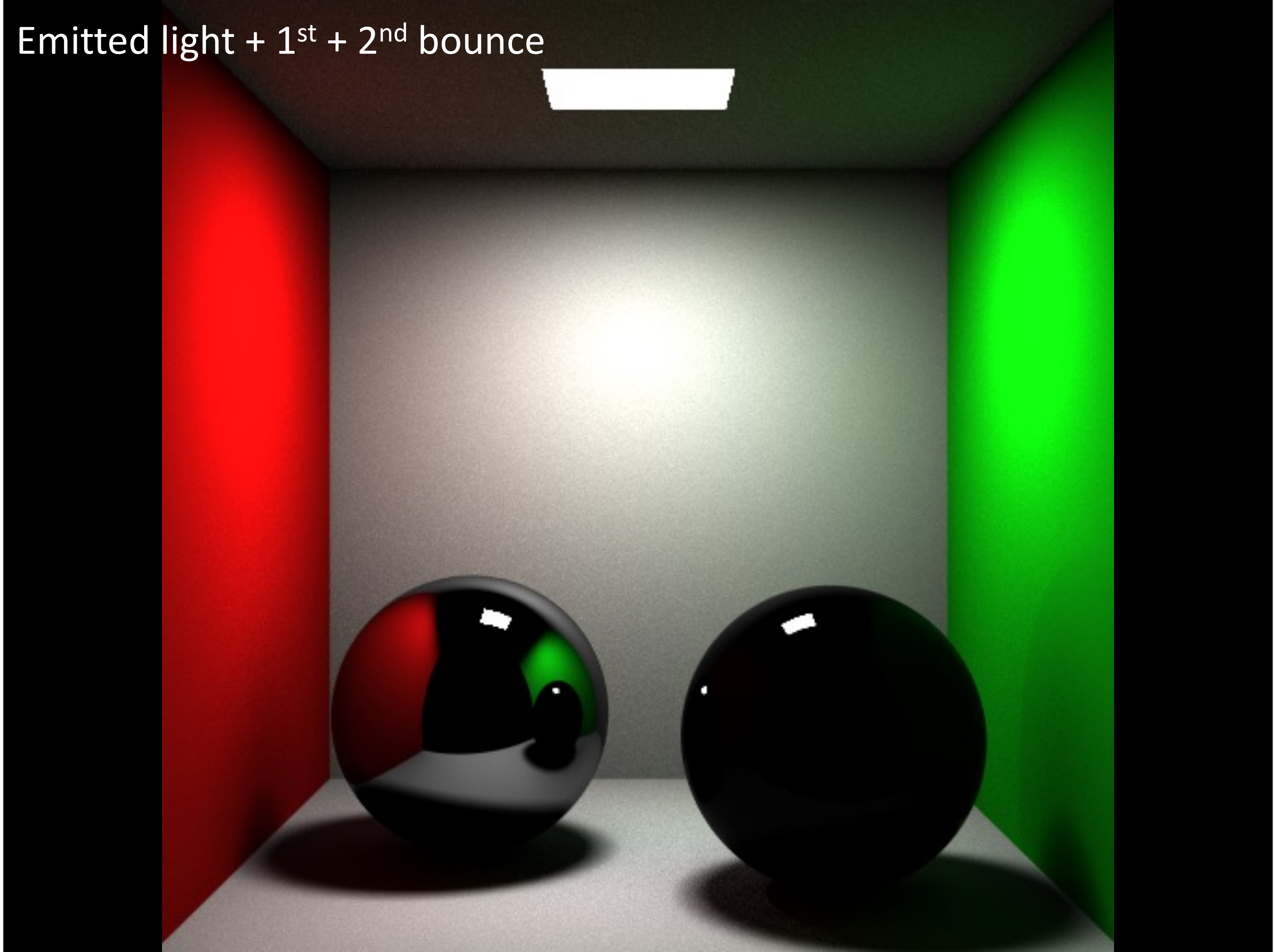


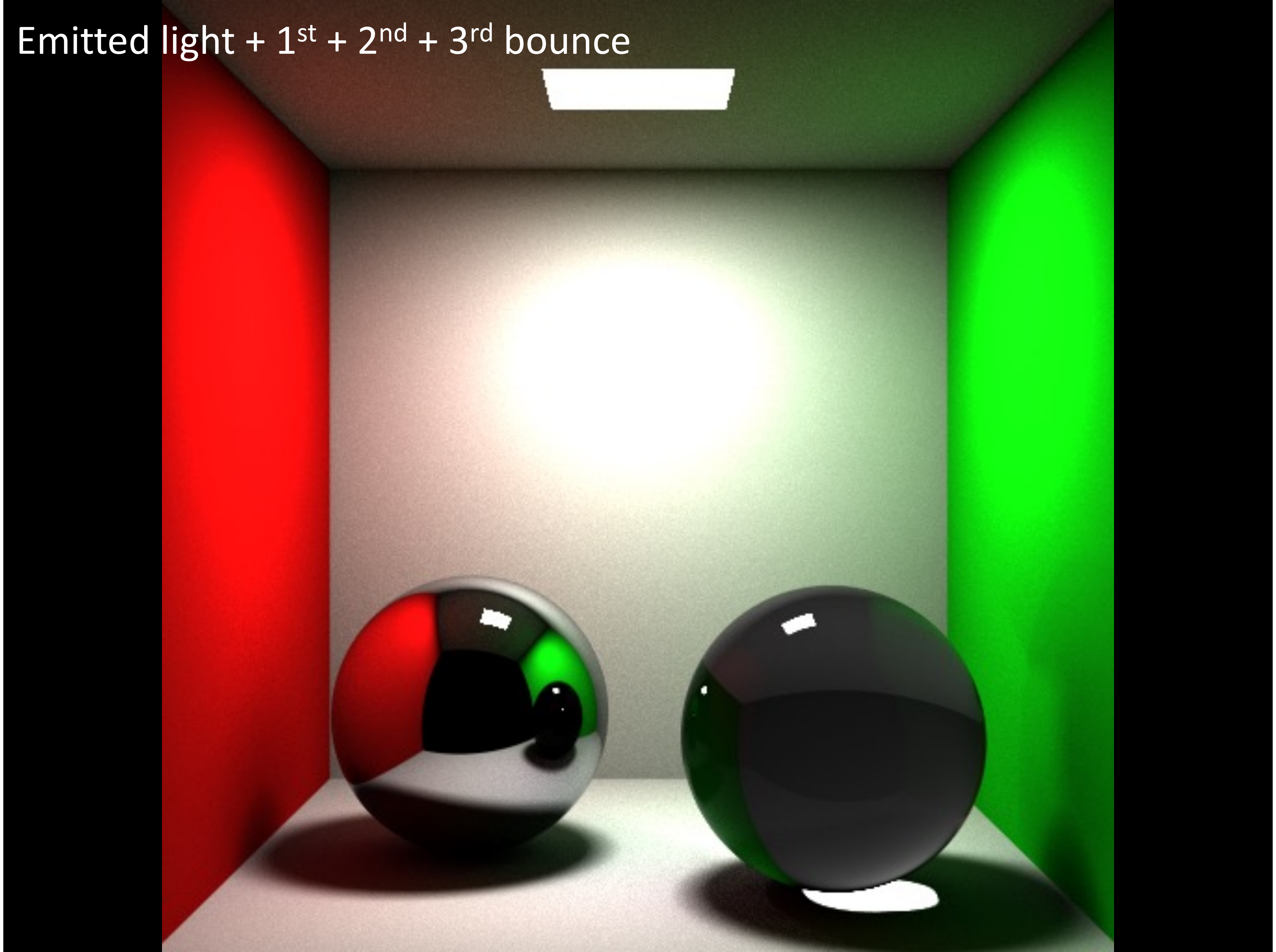

Emitted light

Emitted light + 1st bounce

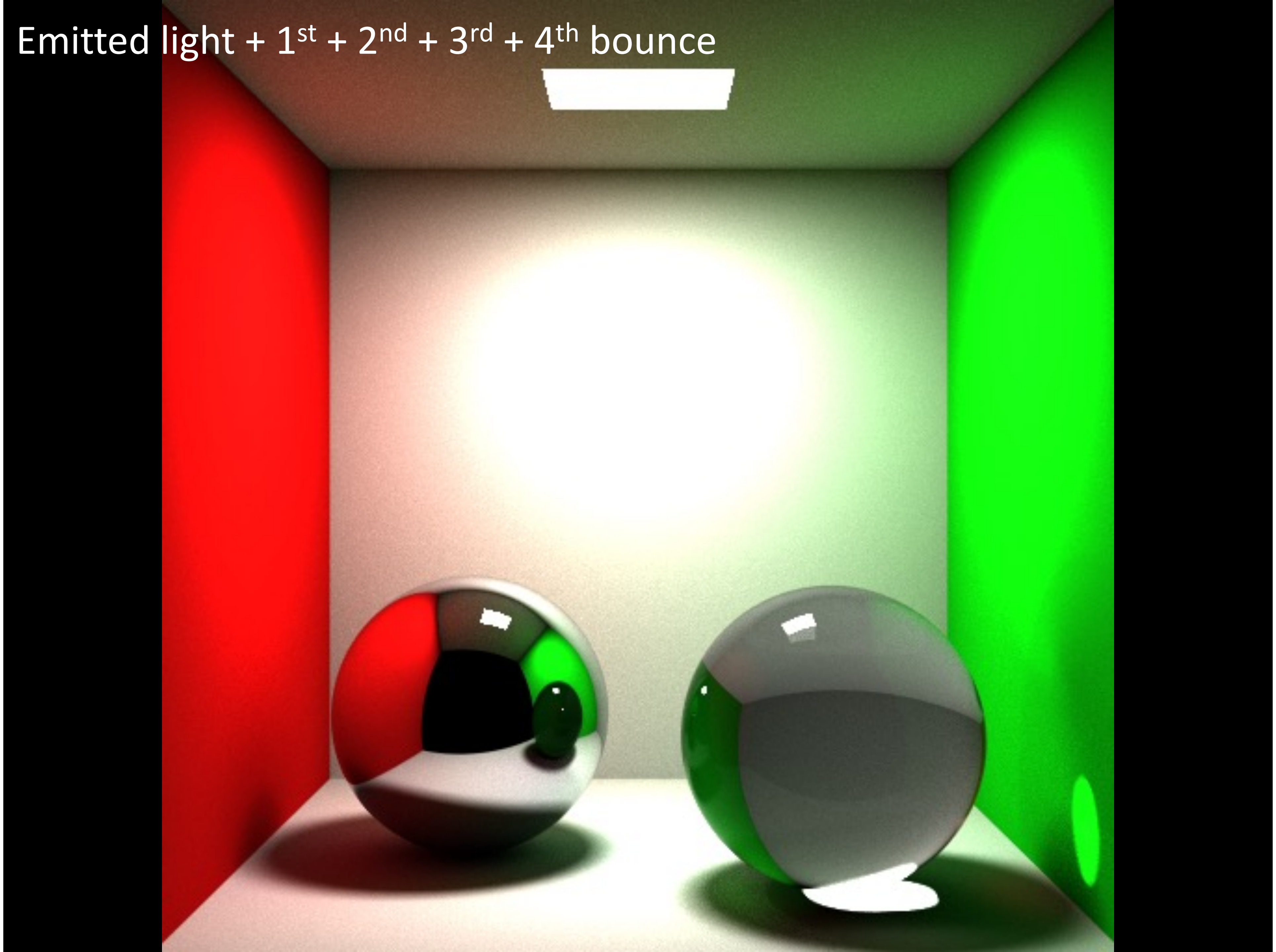


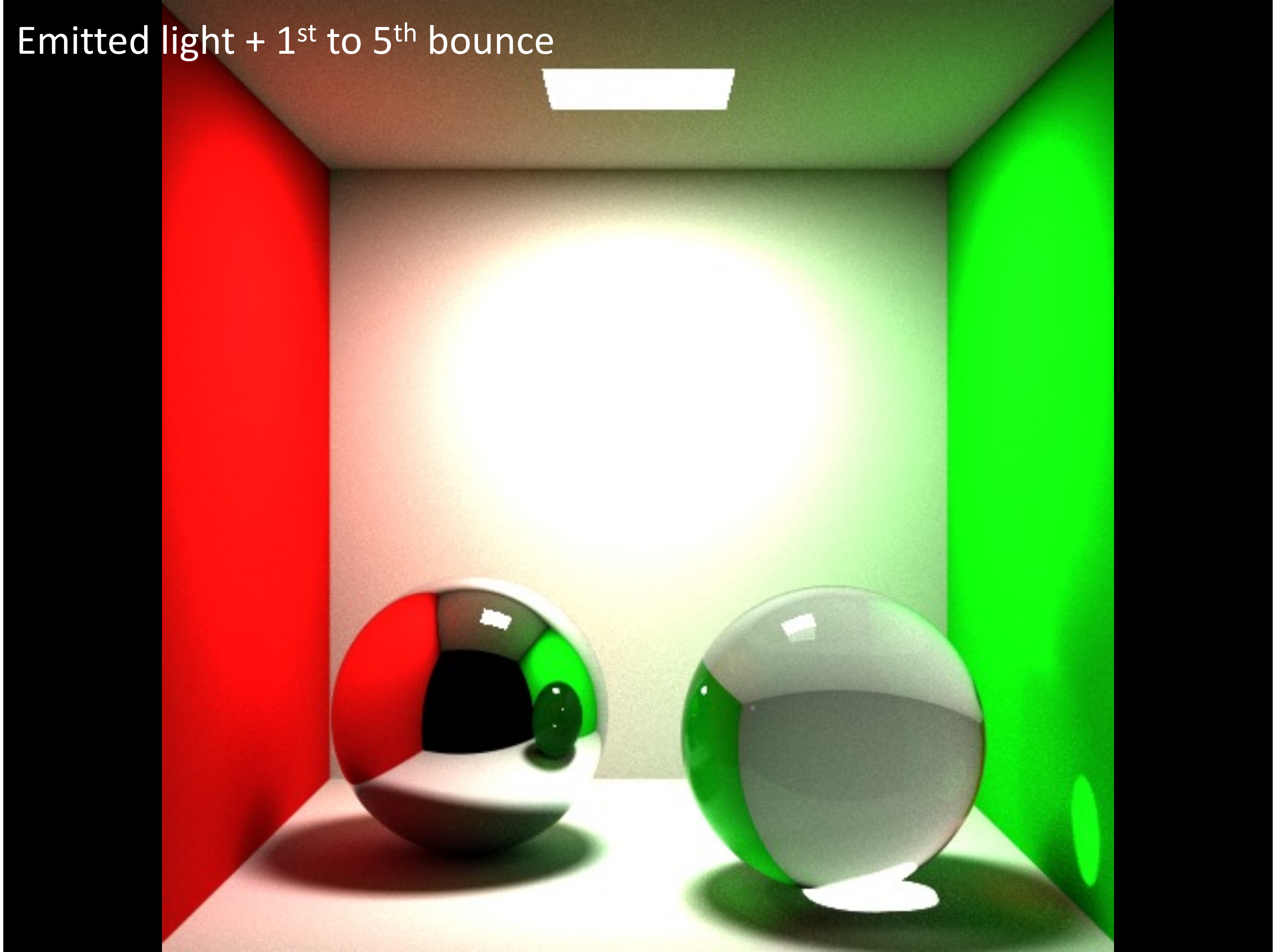
Emitted light + 1st + 2nd bounce

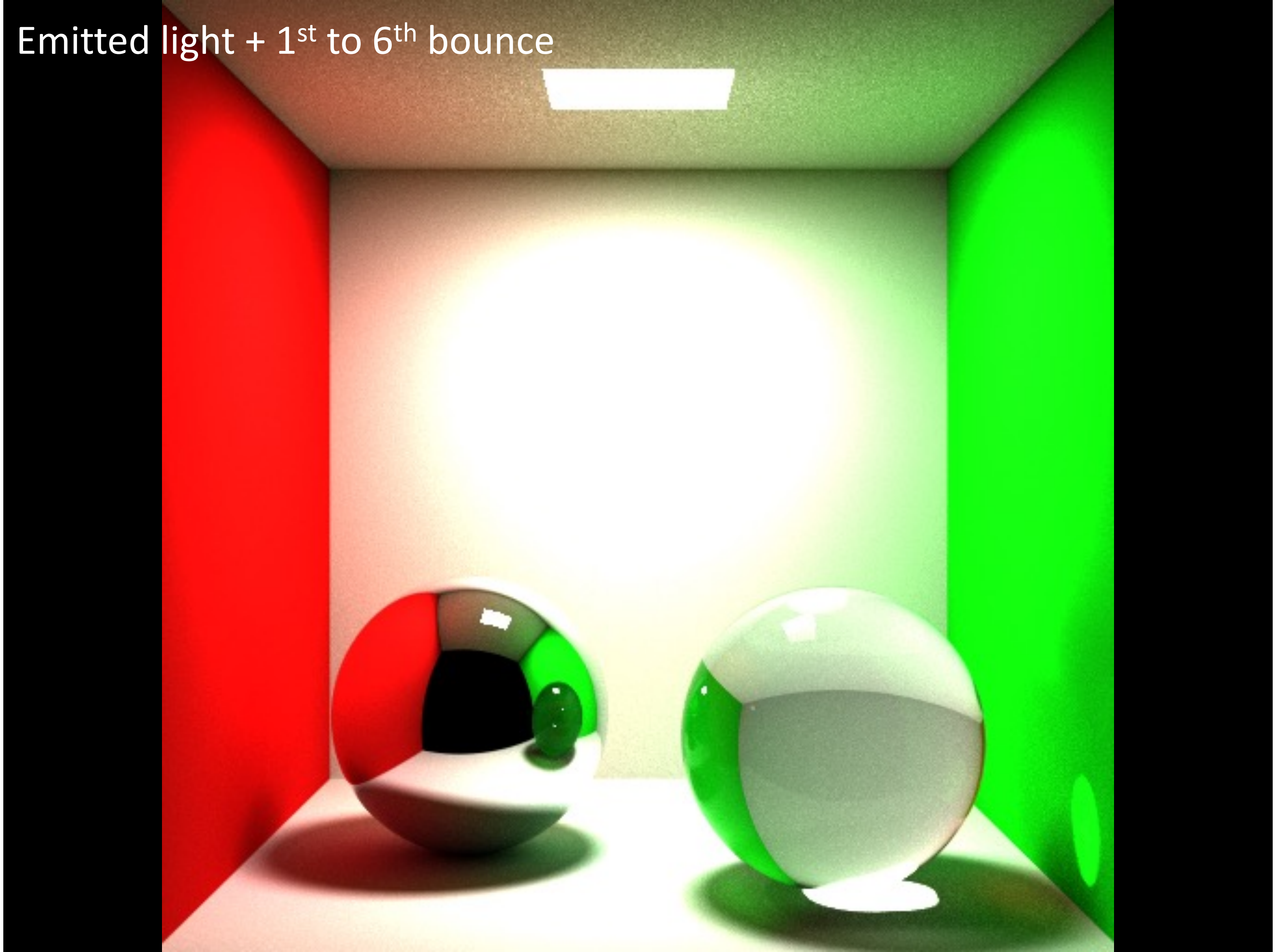



Emitted light + 1st + 2nd + 3rd bounce

Emitted light + 1st + 2nd + 3rd + 4th bounce




Emitted light + 1st to 5th bounce


Emitted light + 1st to 6th bounce

Direct illumination (1st bounce)

Direct illumination + 1 round of global illumination (2nd bounce)

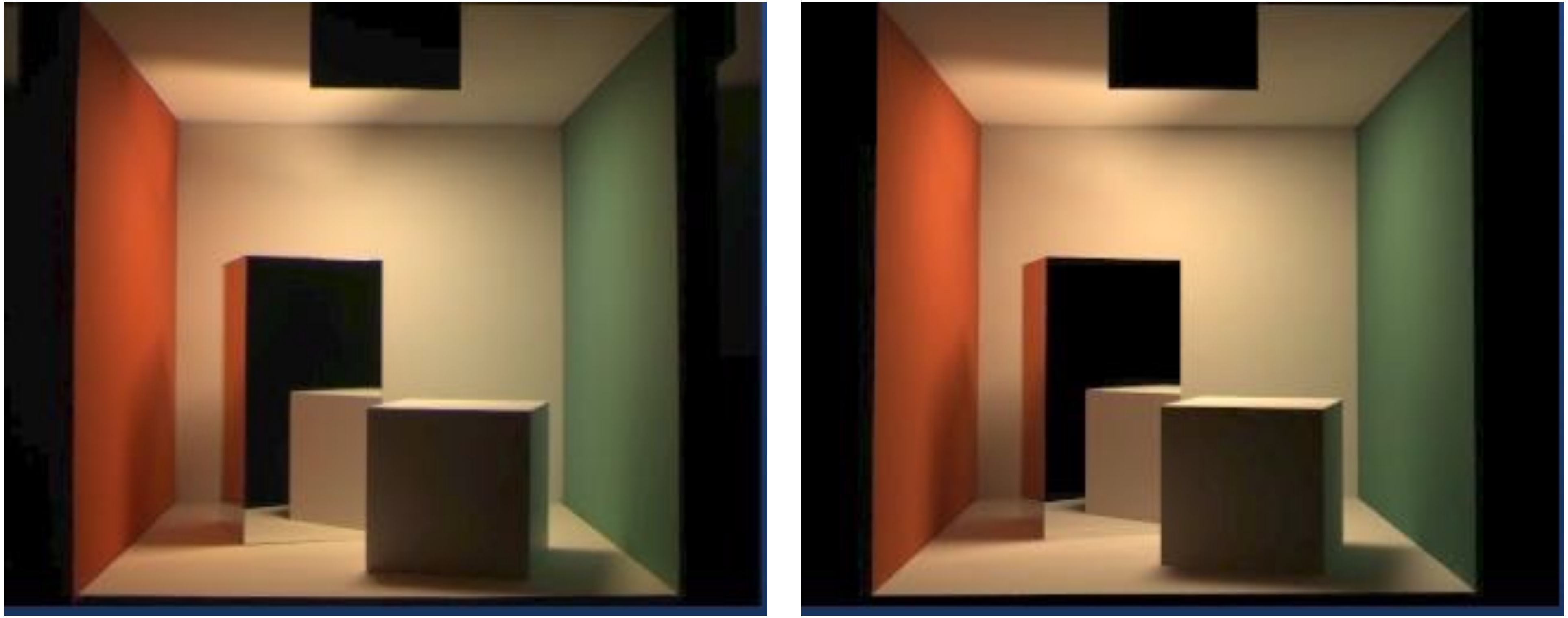
Direct illumination + 2 round of global illumination (3^{rd} bounce)

Direct illumination + 4 round of global illumination (5th bounce)

Direct illumination + 8 round of global illumination (9th bounce)

Direct illumination + 16 round of global illumination (17th bounce)

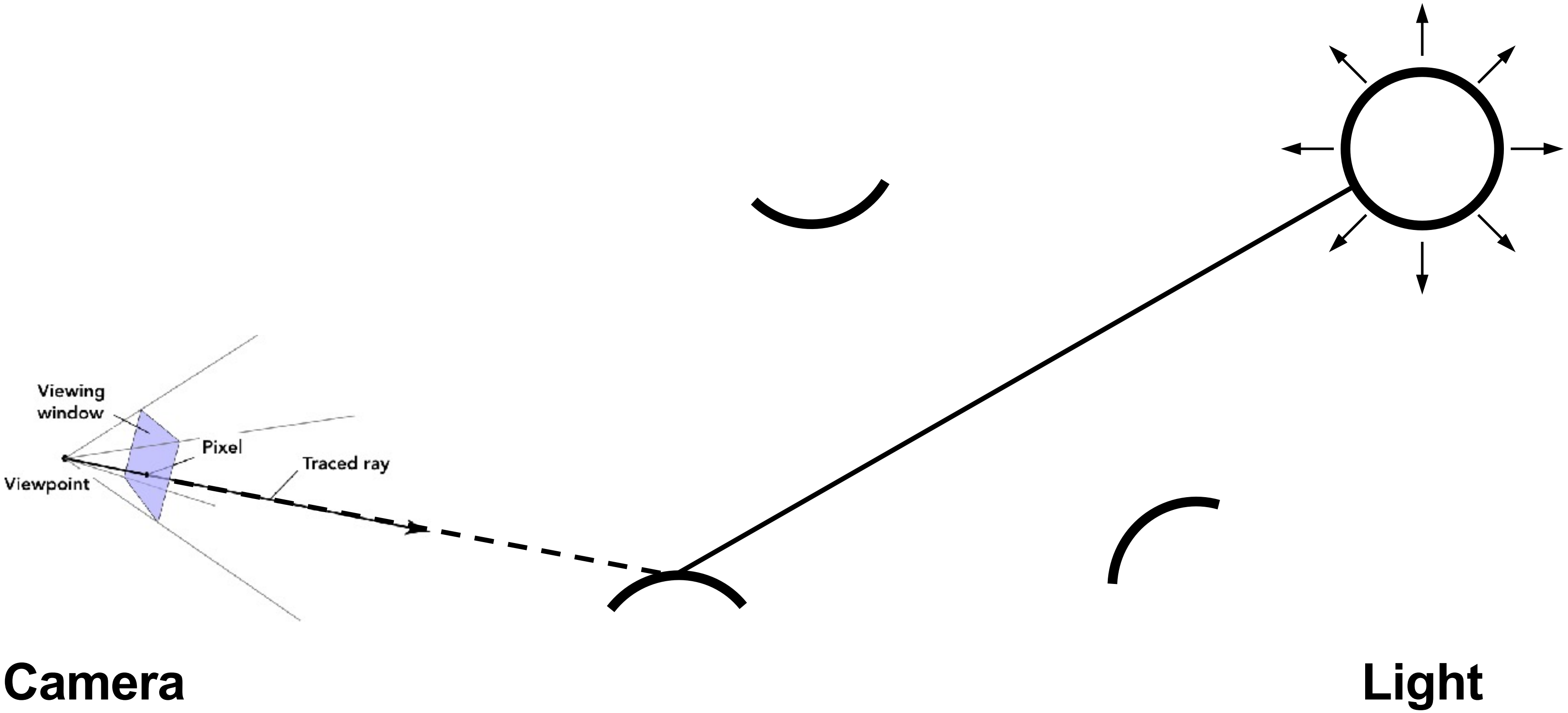
Cornell Box – Photograph vs Rendering



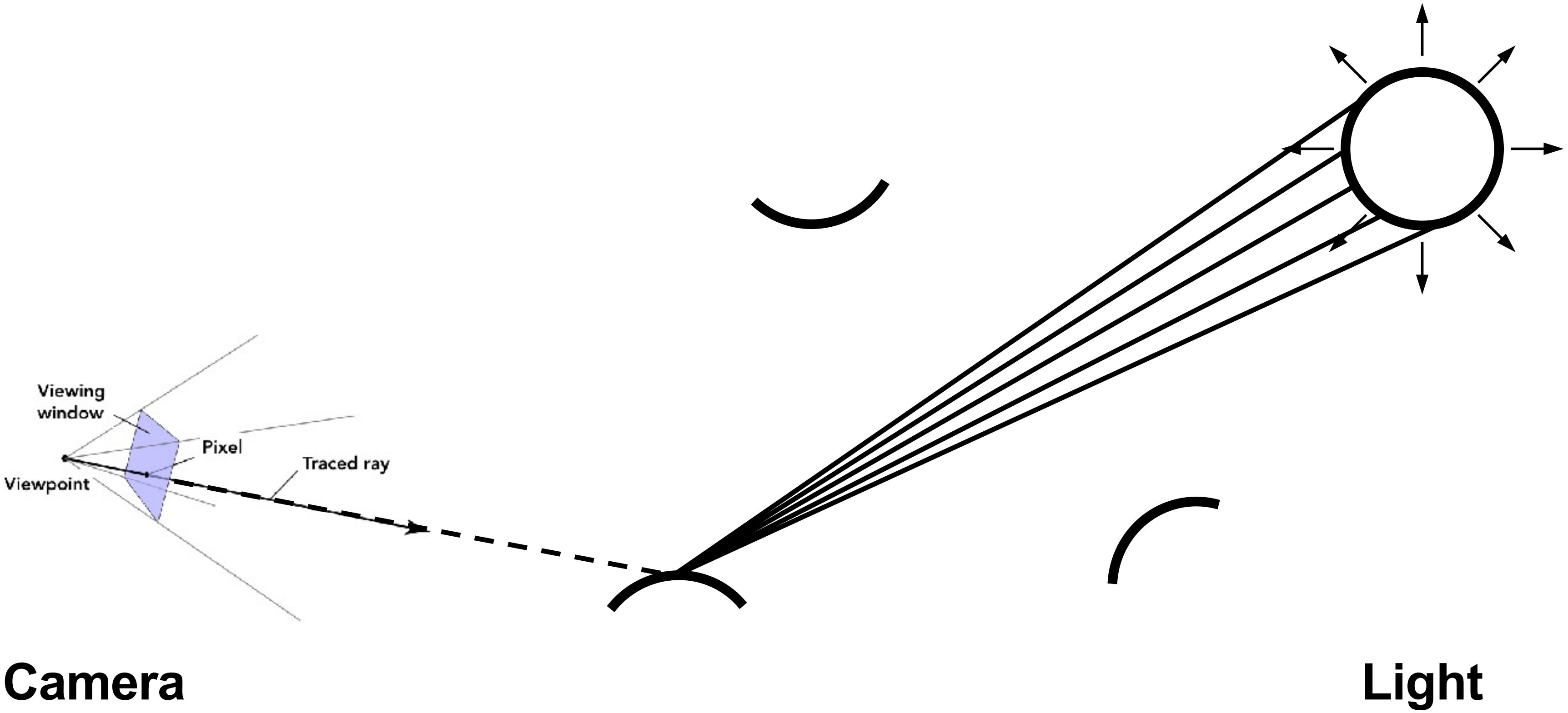
Photograph (CCD) vs. global illumination rendering

Light Paths

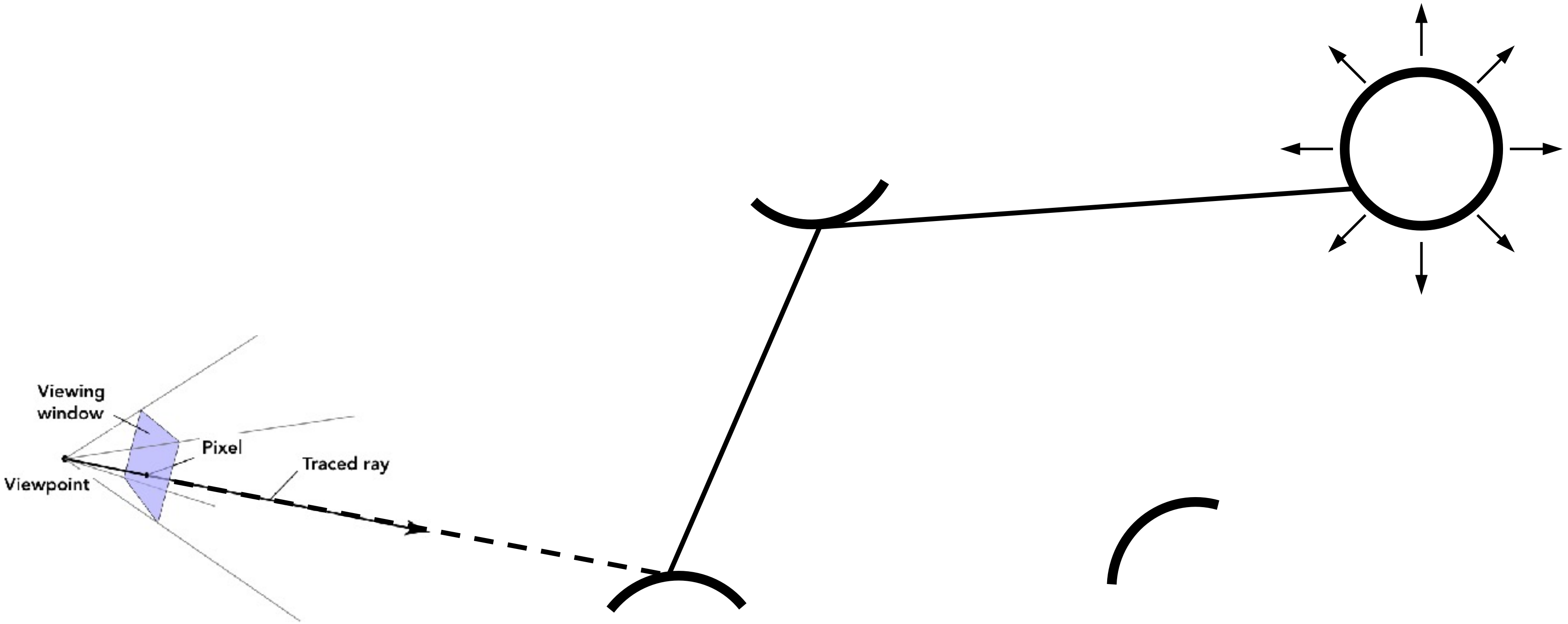
1-Bounce Path Connecting Ray to Light



1-Bounce Paths Connecting Ray to Light

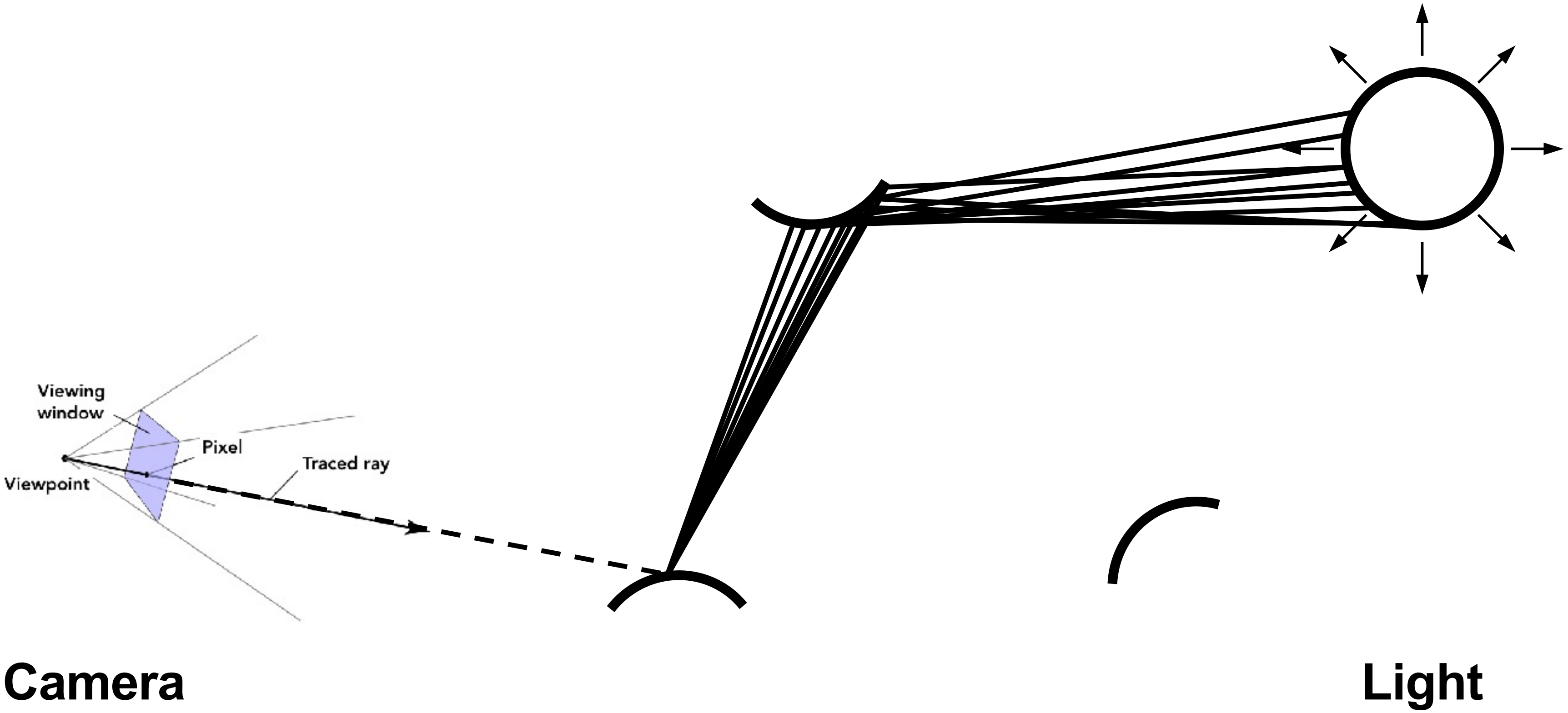


2-Bounce Path Connecting Ray to Light

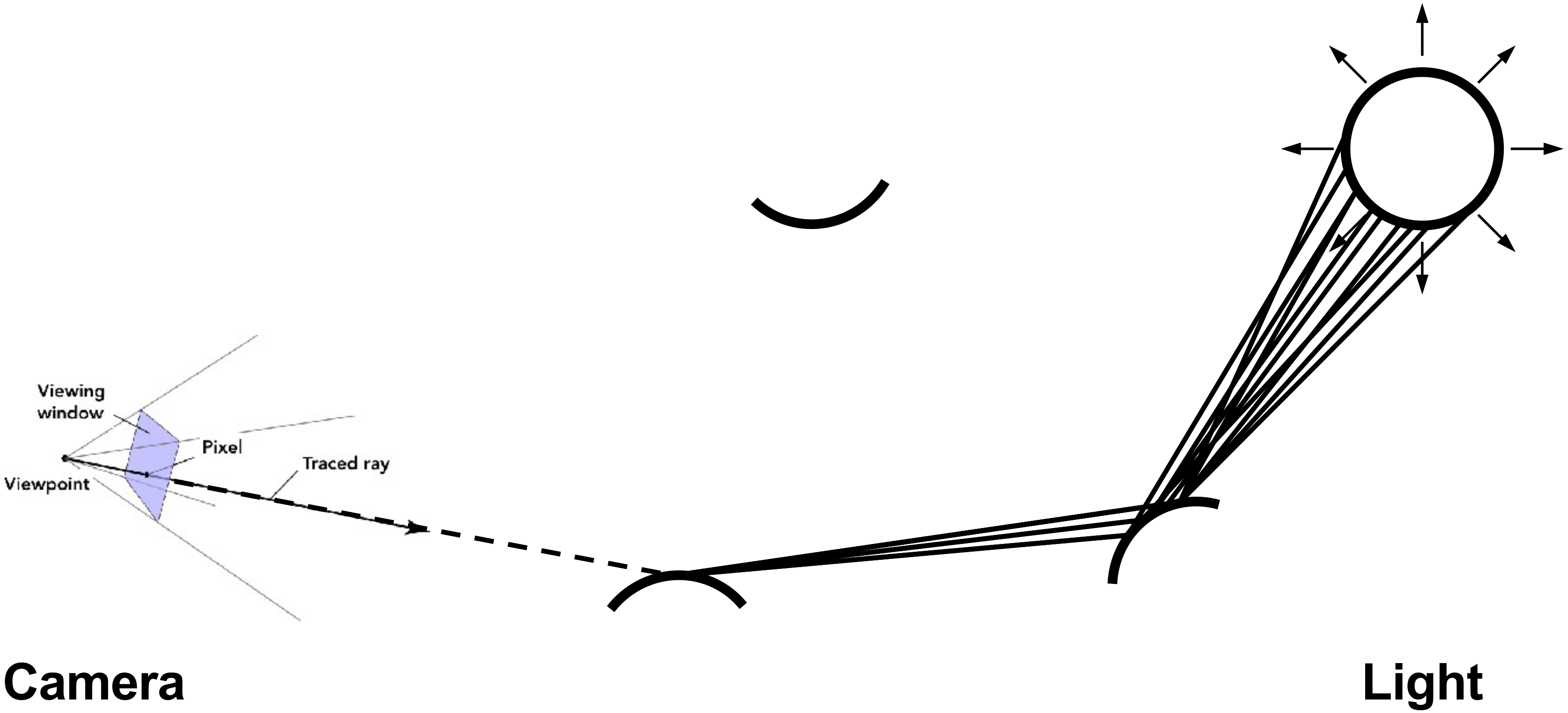


Camera **Light**

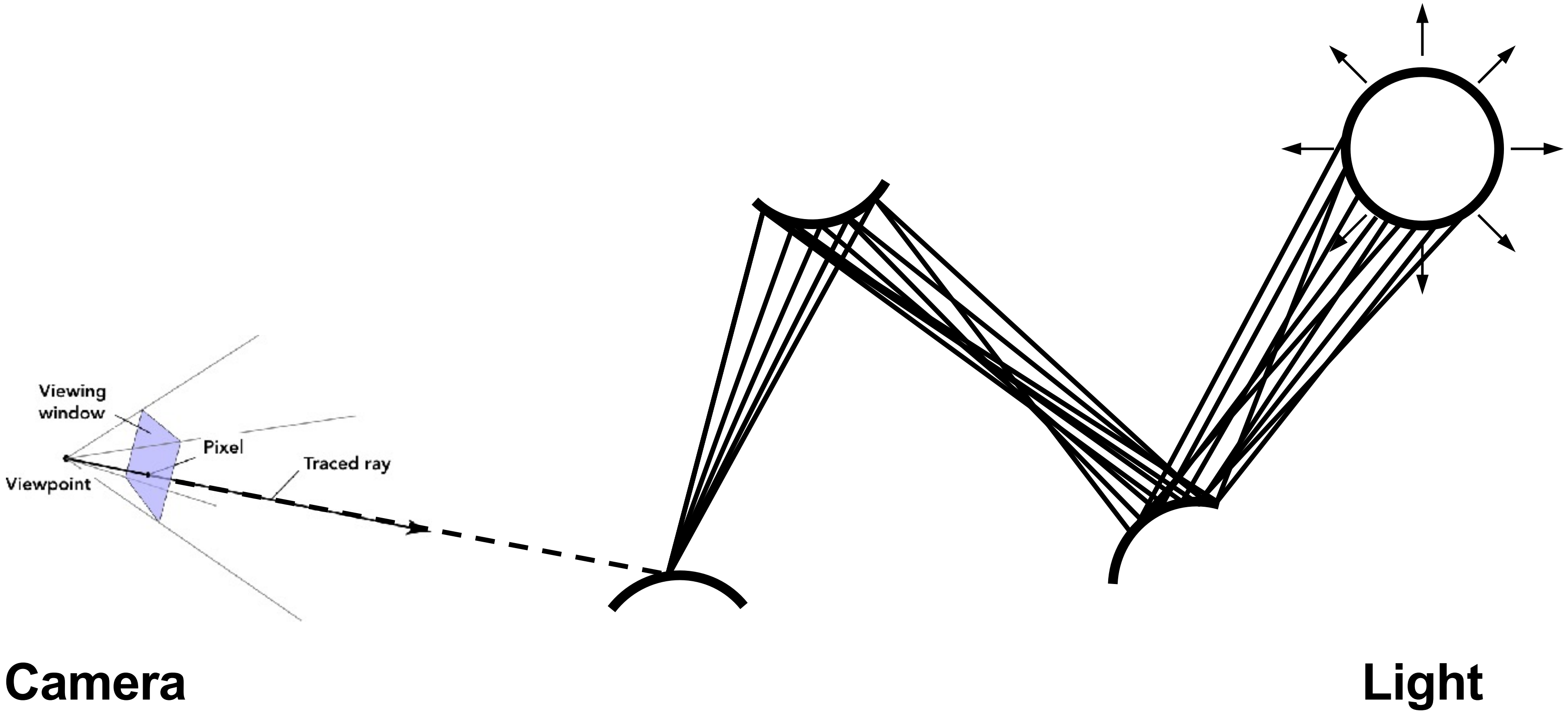
2-Bounce Paths Connecting Ray to Light



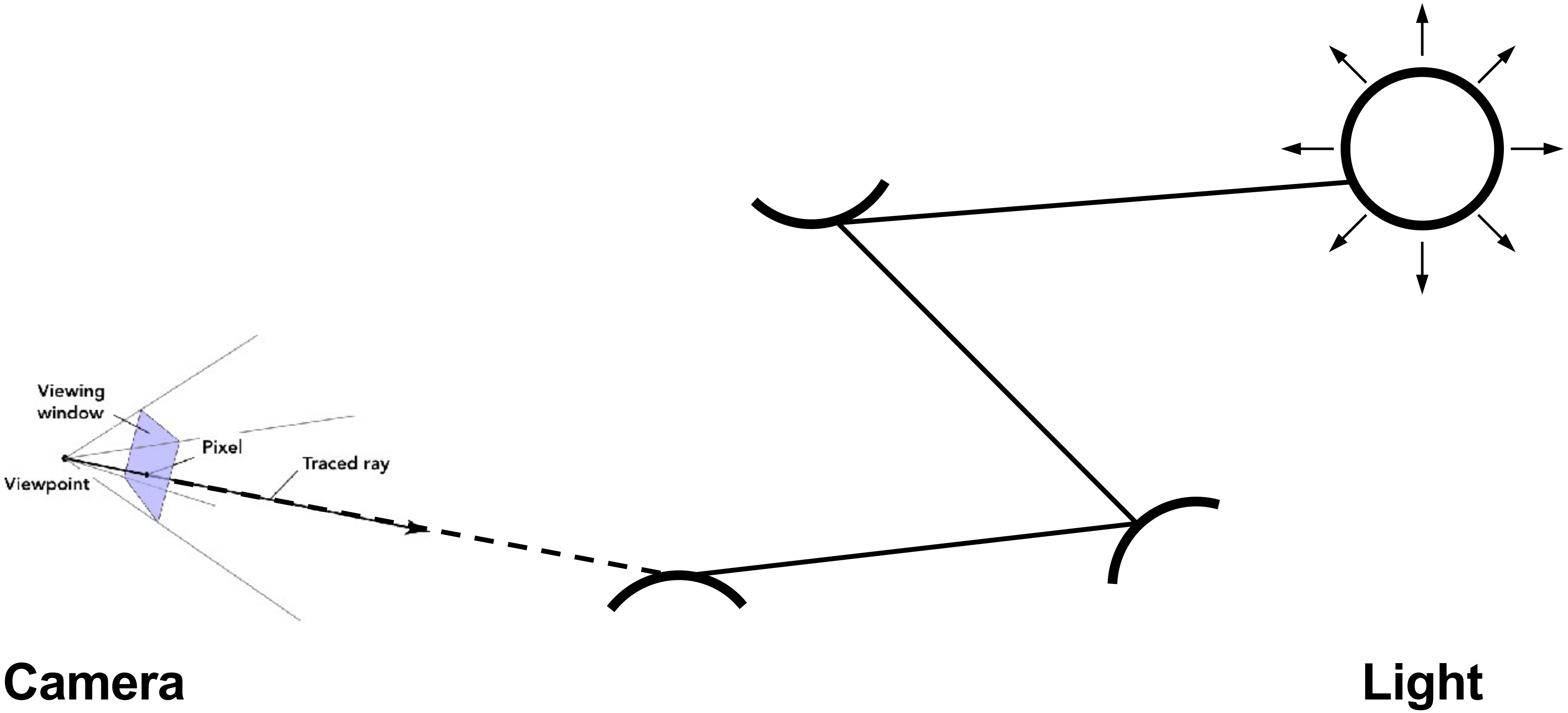
2-Bounce Paths Connecting Ray to Light



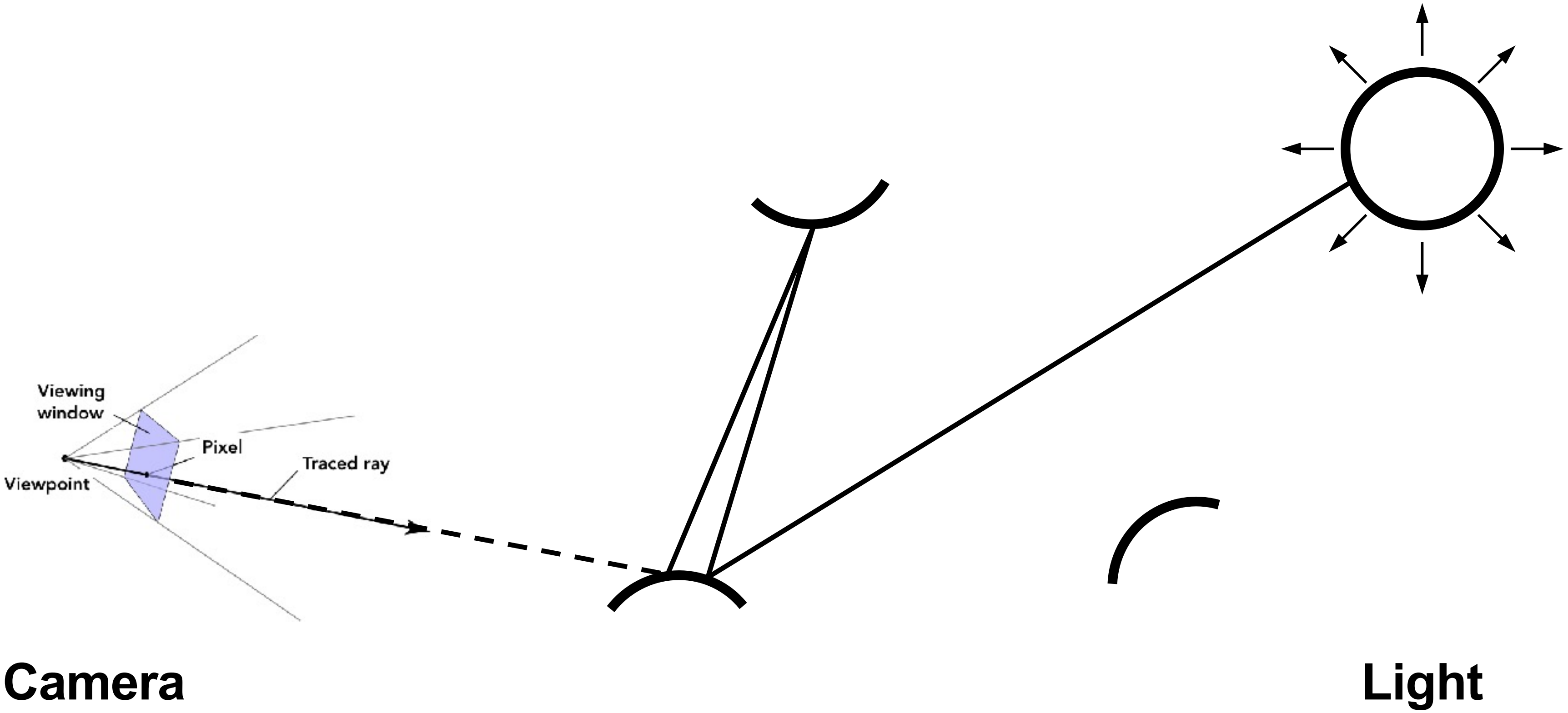
3-Bounce Paths Connecting Ray to Light



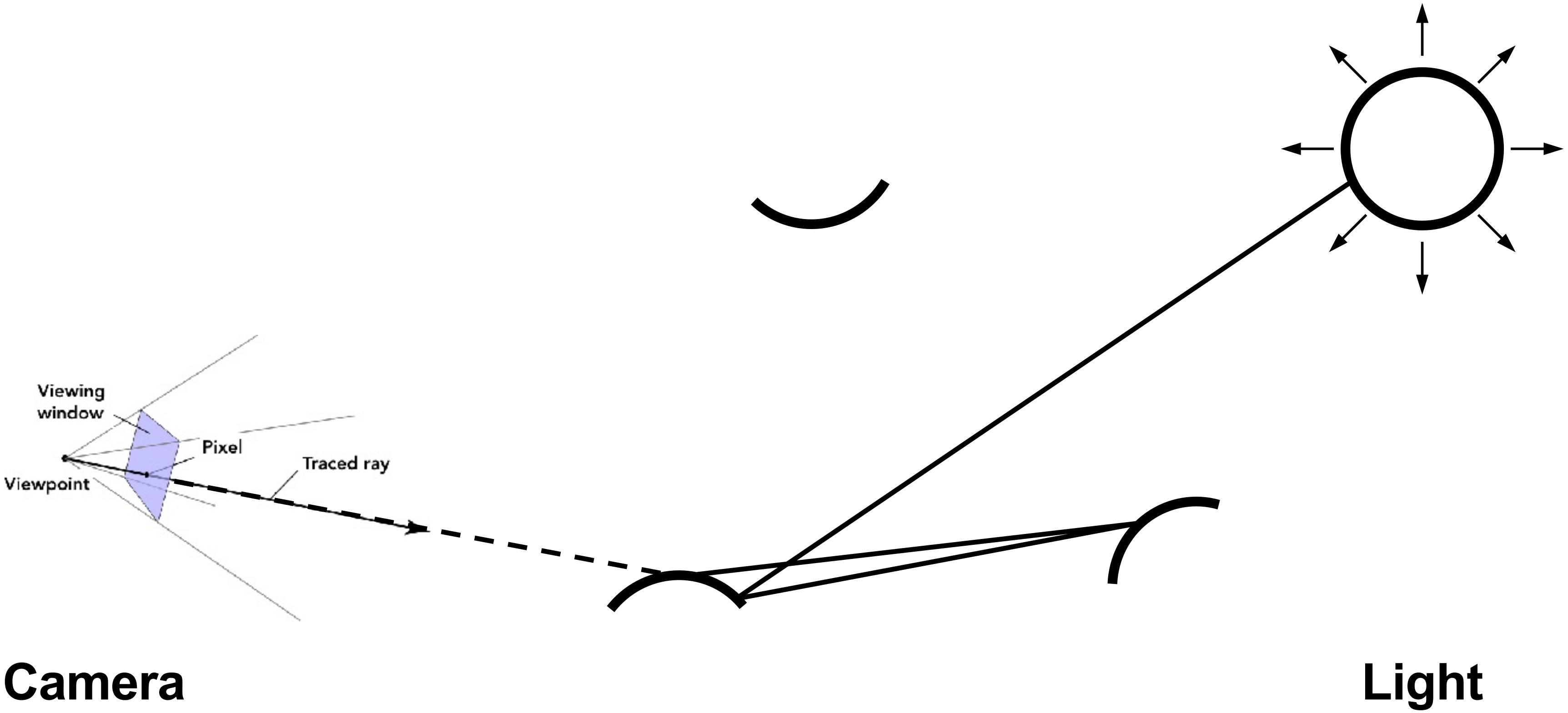
3-Bounce Path Connecting Ray to Light



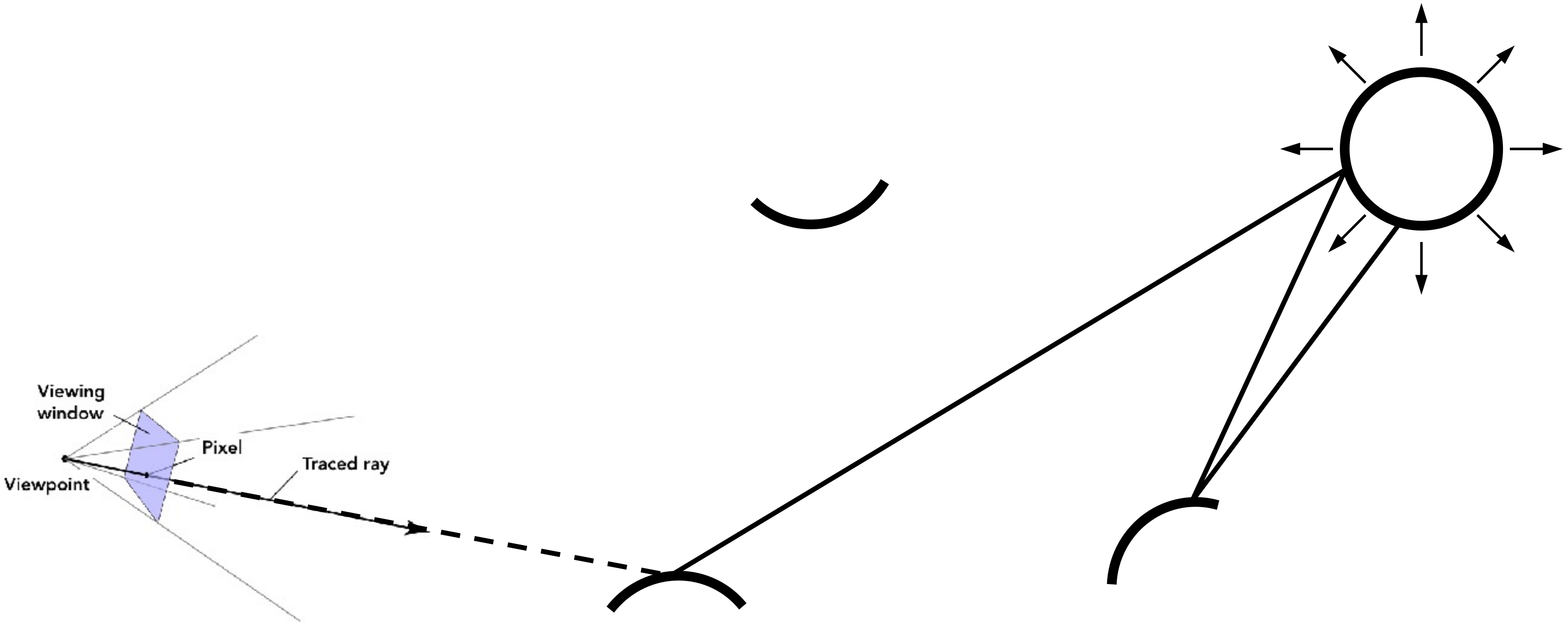
3-Bounce Path Connecting Ray to Light



3-Bounce Path Connecting Ray to Light

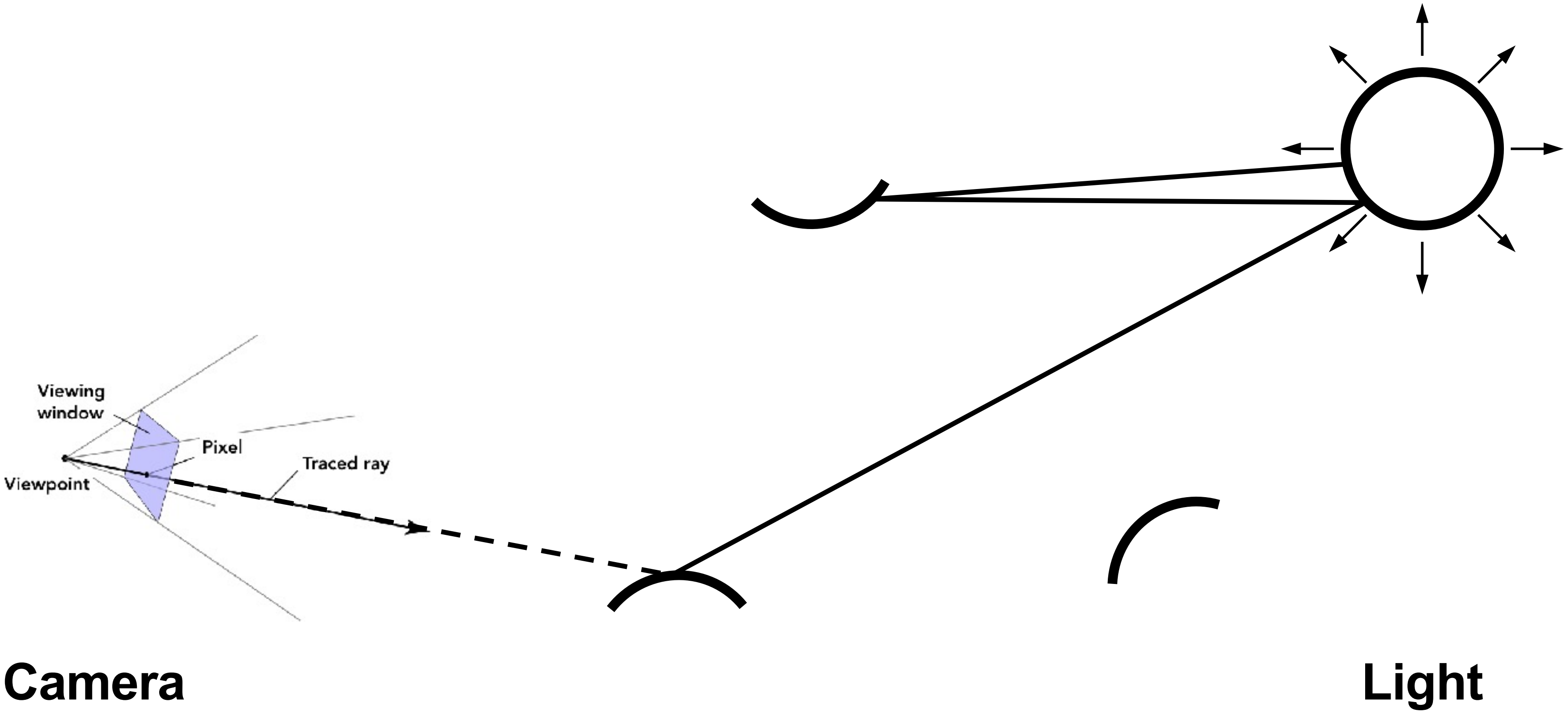


3-Bounce Path Connecting Ray to Light

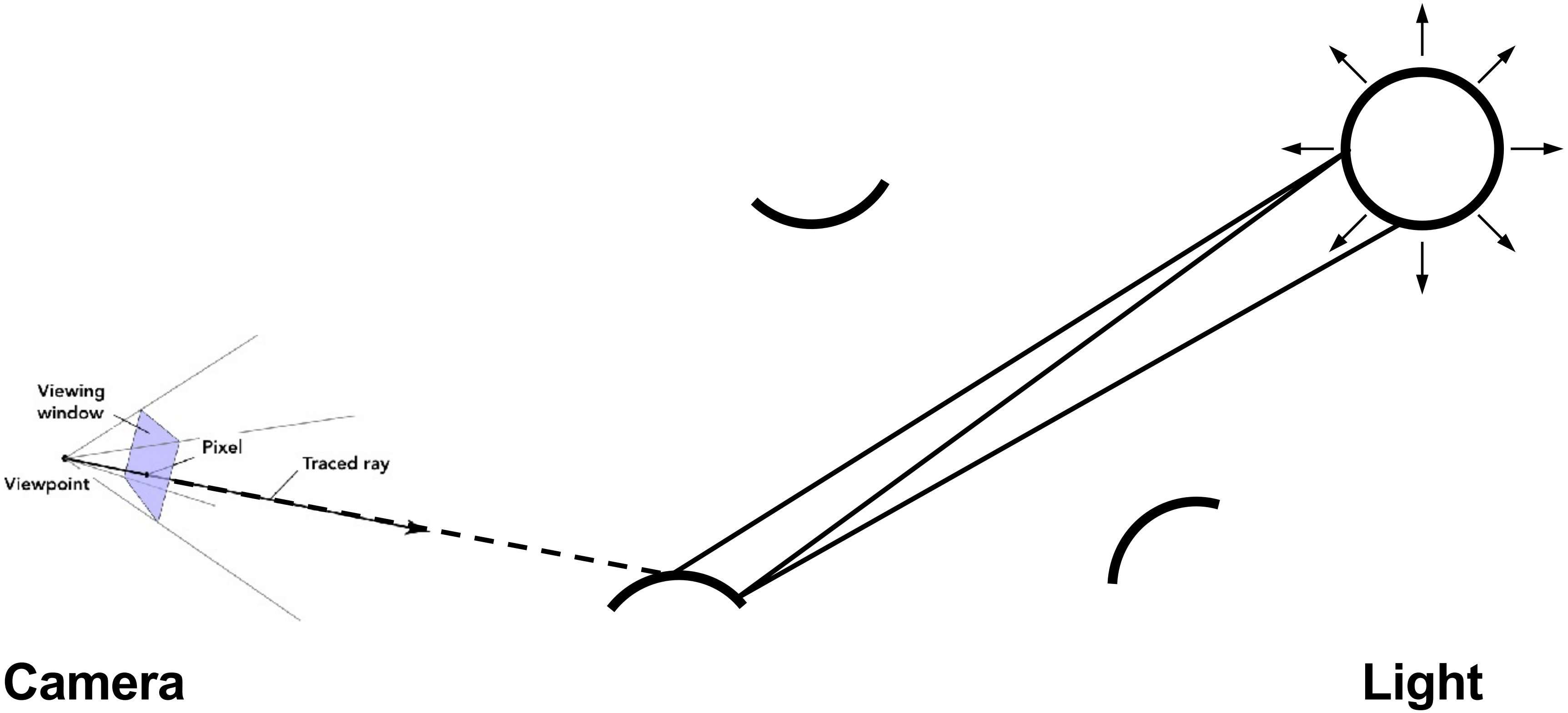


Camera **Light**

3-Bounce Path Connecting Ray to Light



3-Bounce Path Connecting Ray to Light




One sample (path) per pixel

32 samples (paths) per pixel


1024 samples (paths) per pixel

Discussion: Global Illumination Rendering

Sum over all paths of all lengths

Challenges:

- **How to generate all possible paths?**
- **How to sample space of paths efficiently?**

“Real-time Ray-tracing” research spearheaded by NVIDIA focus on developing algorithms and GPU architecture that can lead to real-time & memory efficient rendering.

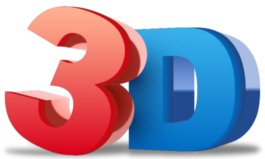
How does AI/ML help in accelerating rendering?

How is this related to Neural Rendering?

To render new images from reconstructed 3D

Current Image

Vision



3D Intrinsic Components

Edit Components

Graphics

New Image under different conditions

Change:

- Viewpoint
- Lighting
- Reflectance
- Background
- Attributes
- Many others…

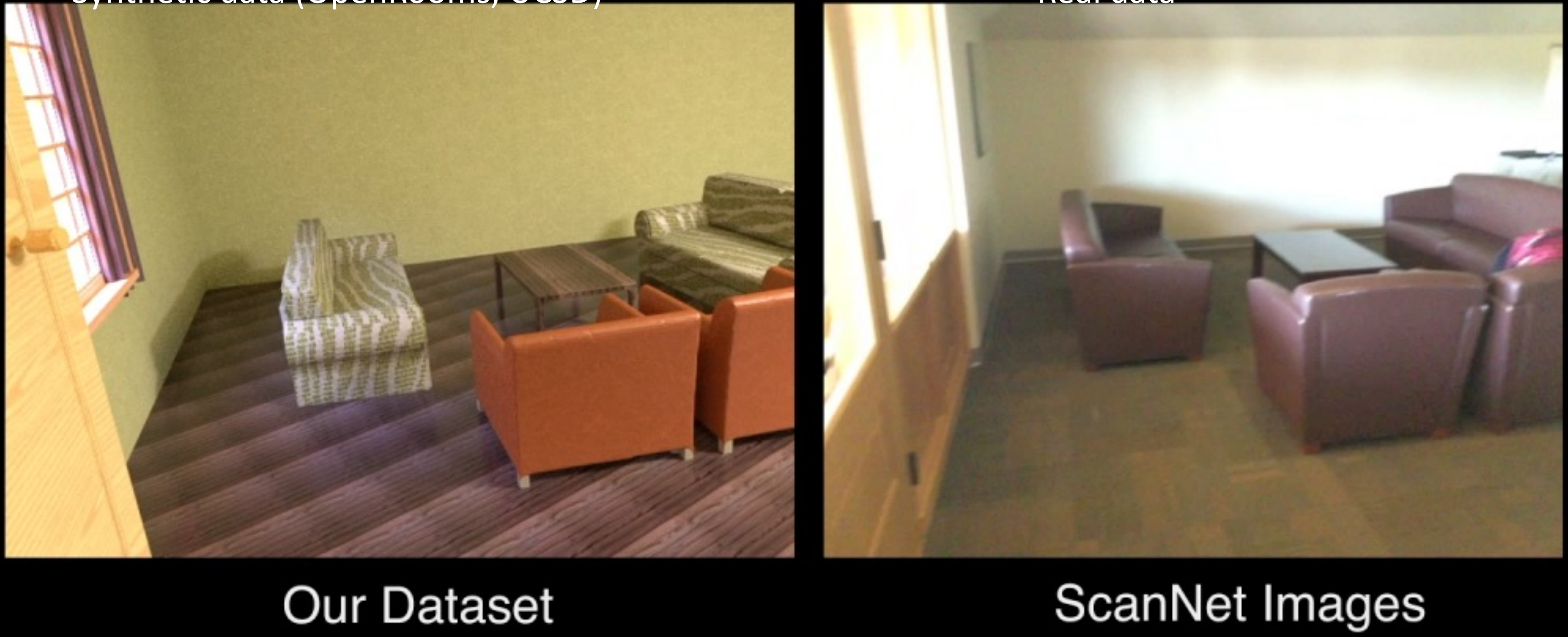
Use Computer Graphics to render new images from reconstructed 3D components.

To generate training data

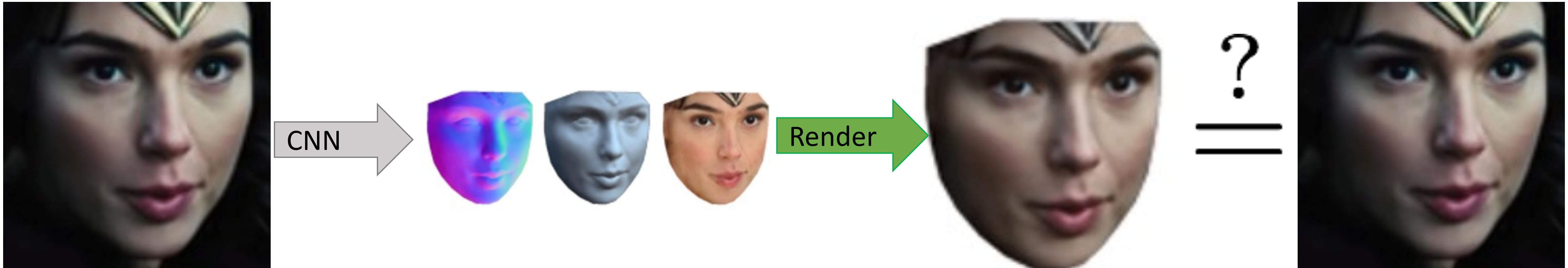
Using Computer Graphics to generate realistic synthetic data for training Deep Networks in Computer Vision.

- Easy to obtain large scale data.
- Better Graphics = less domain gap with real world

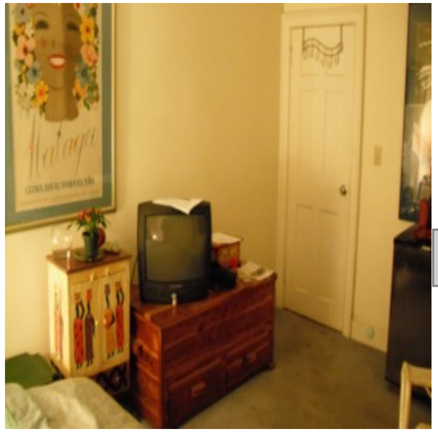
Synthetic data (OpenRooms, UCSD) Real data



Self-supervised learning from real images

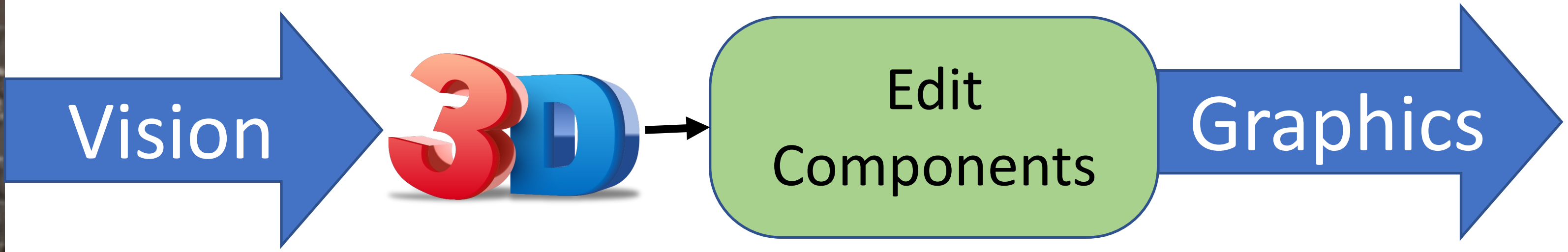


- Rendering is recursive, thus not differentiable.
- You can not backprop loss gradient through a ray-tracer!
- So what do you do?
  - Make some easy assumption – Direct illumination only (good for faces, not for scenes)
  - Differentiable Rendering (Active Research Area in Graphics community!)



Recap

Current Image



New Image under different conditions

3D Intrinsic Components

Questions that you should answer now:

- How do you represent 3D geometry, camera, BRDF, and lighting?
- How do we generate new images from these components.

Questions that we have not answered:

- How do we reconstruct 3D components from image(s) ?
  - You learn a bit in NeRF
  - Mostly covered in any advanced 3D Vision Course.
  - Are you interested to learn more about this?

Next few lectures: Generative models for direct image based rendering.

Current Image

Vision

3D Intrinsic Components

Edit Components

Graphics

New Image under different conditions

Change:

- Viewpoint
- Lighting
- Reflectance
- Background
- Attributes
- Many others…

Implicit: Use a Neural Network (Conditional Generative networks) Often, end-to-end.

Important Deadlines

- 590: Assignment 1 due next Thursday, Aug 25.
- 590/790: Please sign up on your paper presentation/review preference!
- 590: If you want to switch to 790, please submit the form (sent in email).
- 590: You will have 5 assignments instead of 4 (but easier! Trust me!)
- 590/790: Start forming your project group. If attempting self project, please come and talk to me!
- Slack Channel setup by Michael Womick

Slide Credits

- UC Berkeley CS 184/284a – Spring 2021 (Ren Ng, Angjoo Kanazawa)
- U Texas CS 354 – Spring 2022 (Sarah Abraham)
- Many amazing research papers!