Game Engines

CMPM 164, F2019

Prof. Angus Forbes (instructor)
angus@ucsc.edu

Montana Fowler (TA)
mocfowle@ucsc.edu

Website: creativecoding.soe.ucsc.edu/courses/cmpm164
Slack: https://ucsccmpm164.slack.com

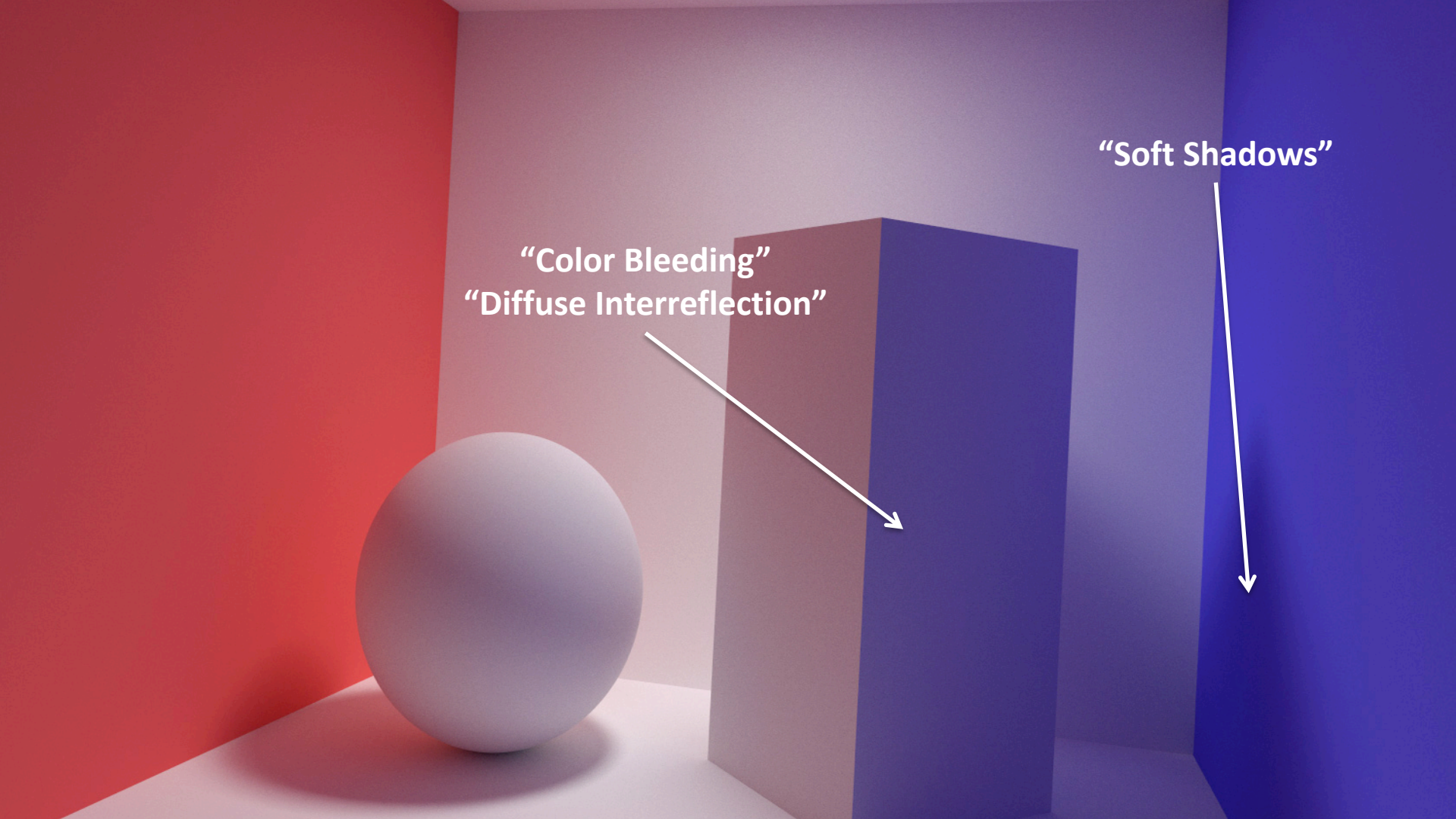
Global Illumination

Also called “indirect illumination” because it includes light that bounces off or through material, in ways that are difficult to simulate using only textures and screen-space methods, and that are more computationally intensive than classic ray tracing (ie, require more rays).

“Classic” ray tracing makes assumptions:

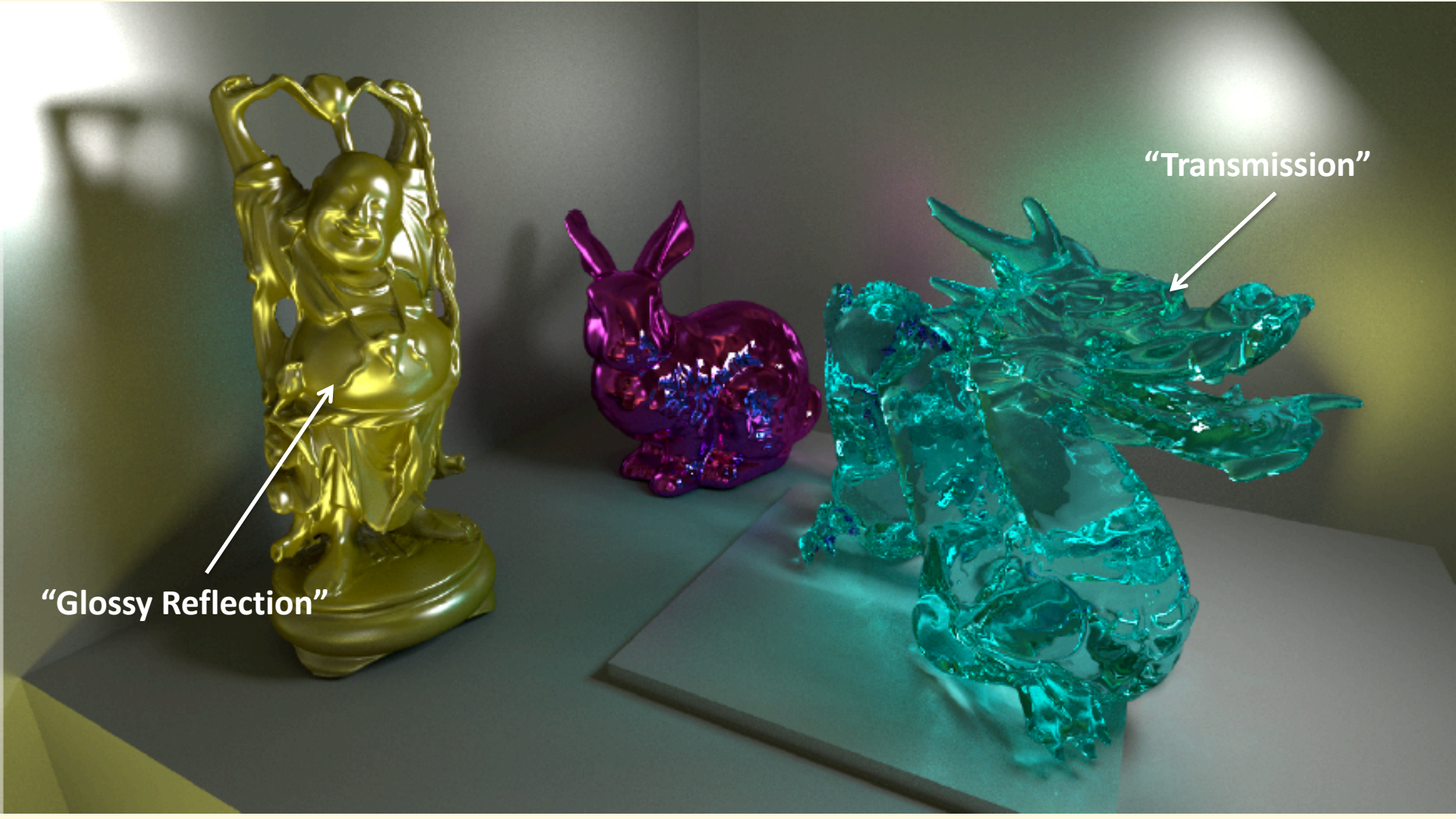
- That all lights are point lights
- That every photon that hits a surface reflects or refracts through it in the same way / that you only need to model a single photon, and that single photon can “split” into two.
- That objects without a direct path from the reflection vector to the light will be completely in shadow (or lit up using a generic “ambient” term)

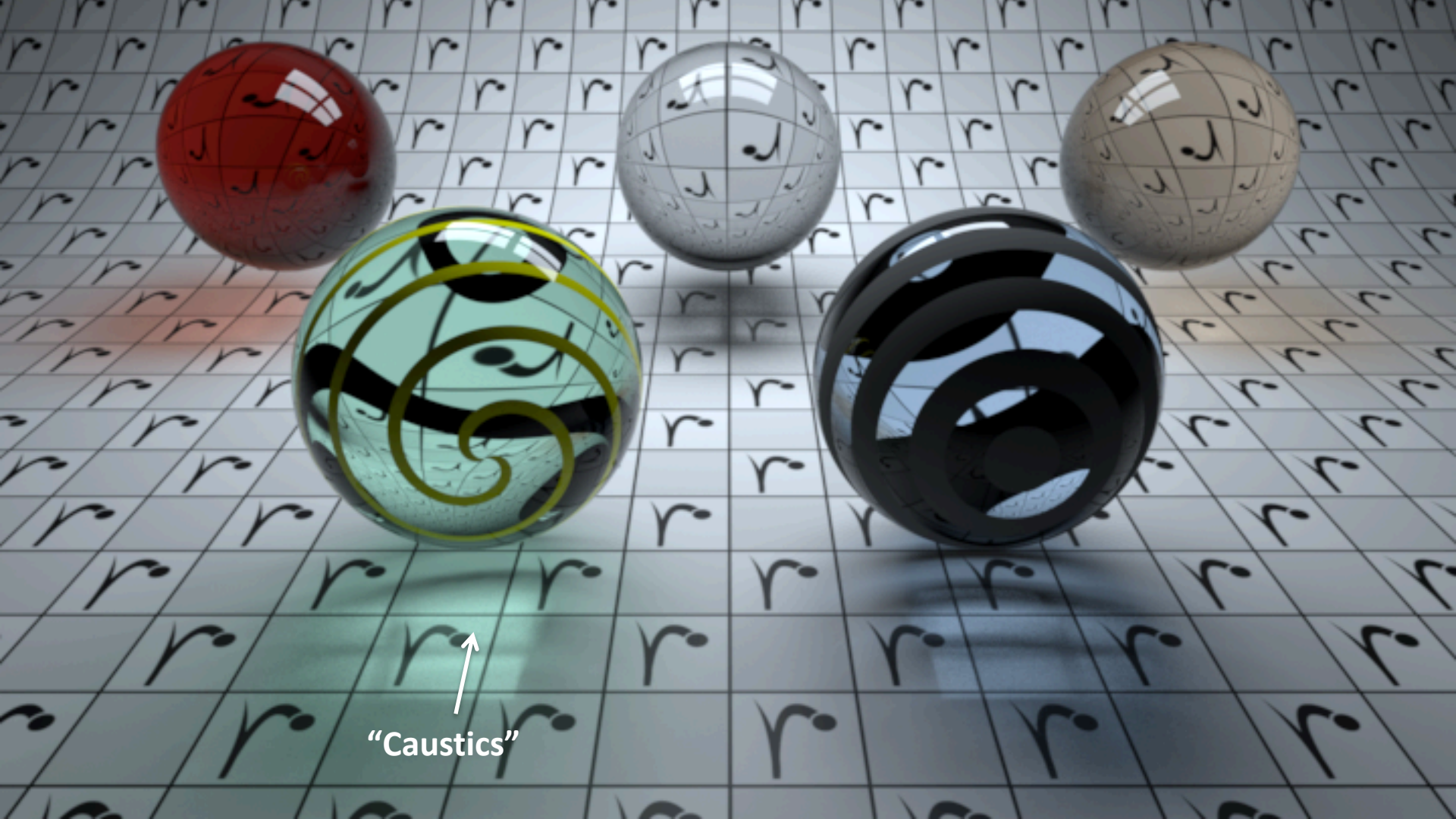
See: https://morgan3d.github.io/advanced-ray-tracing-course

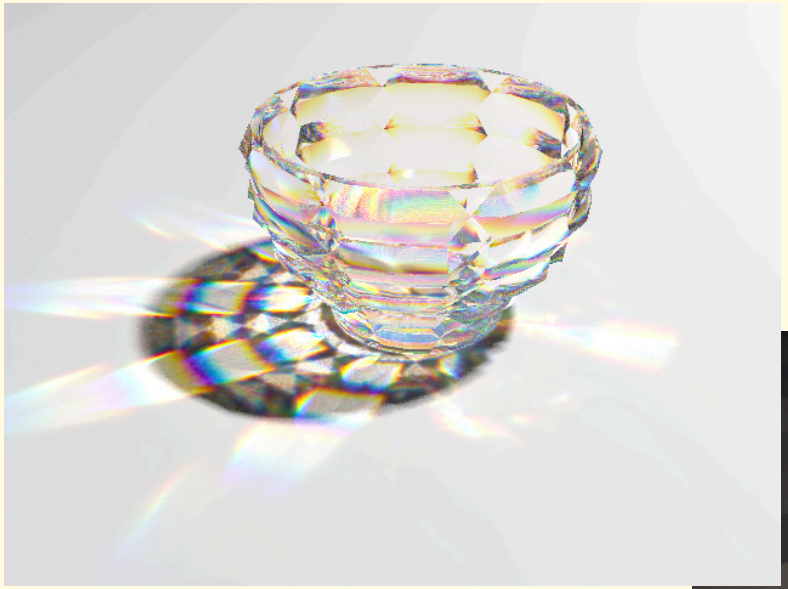

“Soft Shadows”
“Color Bleeding”
“Diffuse Interreflection”

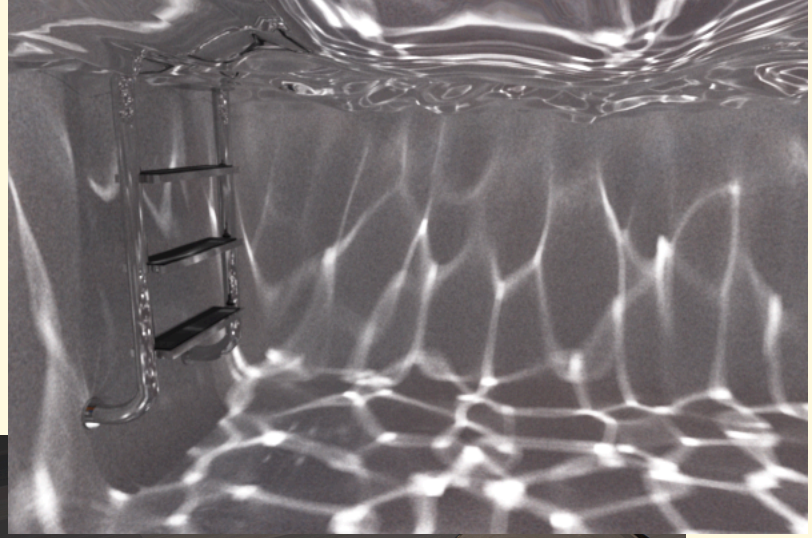

“Indirect Lighting”

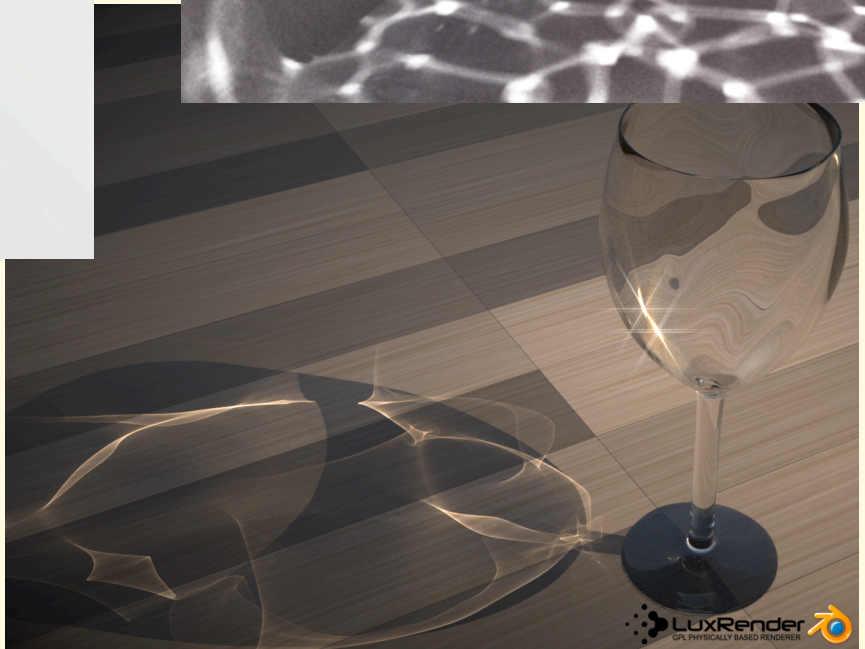

Brigade 2
PATHTRACER:
78.422(75.75)M/S 96.59%
1280 X 720 SPP822
RENDER 120832US 8.3 FPS
BLUR 0.40
SCENE:
UPDATE 0US
LIGHTS 626 AREA 196
OBJECTS 1 PRIMS 189676
ACTIVE MAT:WATERTOWER TYPE:BLINN SPECULARITY:0.75 TRANSPARENCY:0.00
CUDA (4010)
GEFORCE GTX 580 SM20(342MB)
GEFORCE GTX 580 SM20(342MB)
TRACEROPENCL
NO PARKING
ANY TIME
NO STOPPING
ON PAVEMENT
NO PARKING
ACROSS ENTRANCE
KEEP CLEAR
"Ambient Occlusion"


“Transmission”
“Glossy Reflection”


“Caustics”







"Caustics"

Path tracing

In 1981, James Kajiya introduced the concept of the *Rendering Equation*, an attempt to define the main components that are necessary to model in order to create a physically realistic scene.

The equation defines the radiance leaving a particular point as equal to the sum of *emitted* radiance plus *reflected* radiance

The Rendering Equation

$$L_{\mathrm{o}}(X,\hat{\omega}_{\mathrm{o}}) = L_{\mathrm{e}}(X,\hat{\omega}_{\mathrm{o}}) + \int_{\mathbf{S}^2} L_{\mathrm{i}}(X,\hat{\omega}_{\mathrm{i}})\, f_X(\hat{\omega}_{\mathrm{i}},\hat{\omega}_{\mathrm{o}})\, |\hat{\omega}_{\mathrm{i}} \cdot \hat{n}|\, d\hat{\omega}_{\mathrm{i}}$$

Outgoing light Emitted light Incoming light Material

The Rendering Equation

Outgoing direction

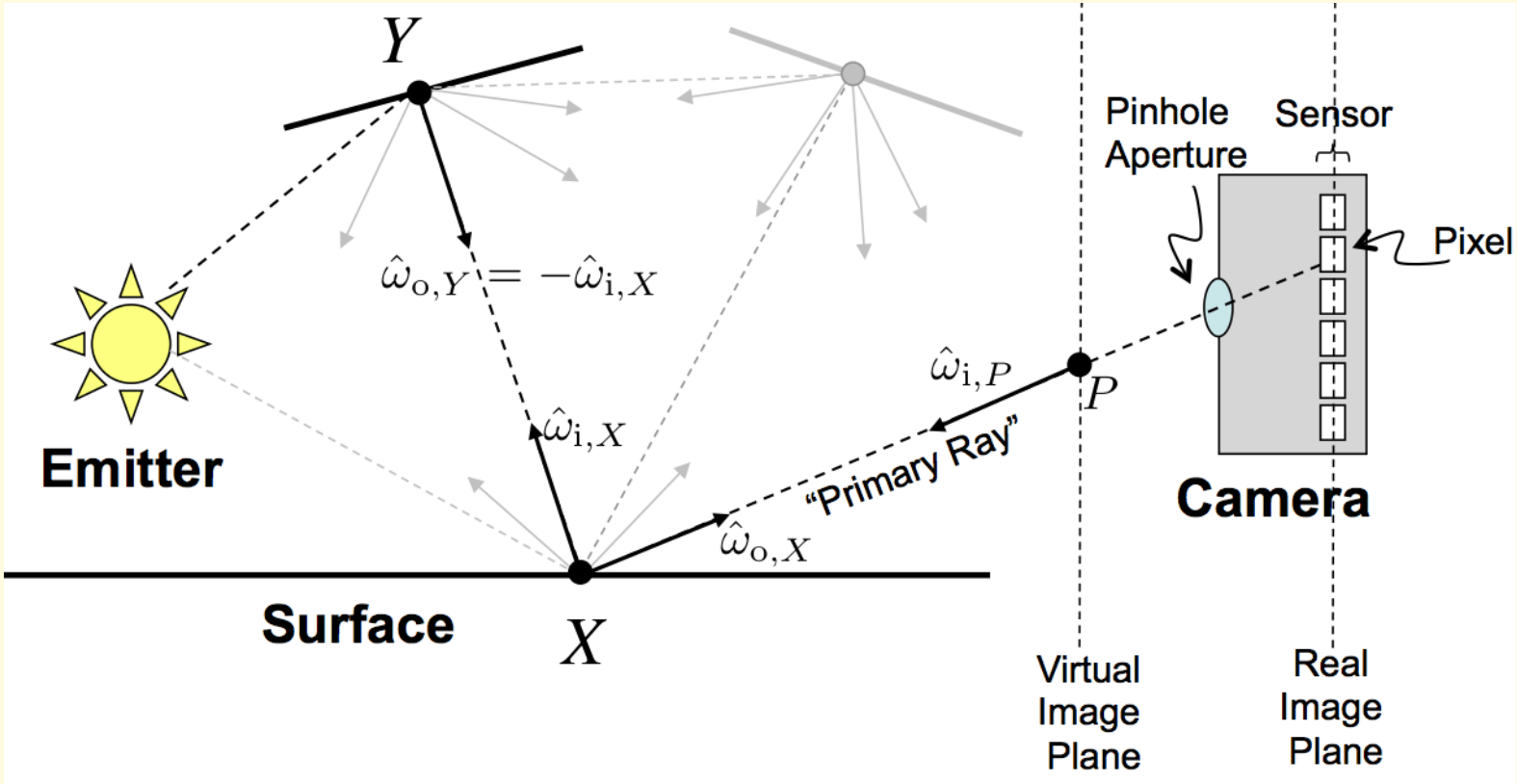
Incoming direction

$$L_o(X, \hat{\omega}_o) = L_e(X, \hat{\omega}_o) + \int_{S^2} L_i(X, \hat{\omega}_i)\ f_X(\hat{\omega}_i, \hat{\omega}_o)\ |\hat{\omega}_i \cdot \hat{n}|\ d\hat{\omega}_i$$

A point in the scene

All incoming directions
(a sphere)

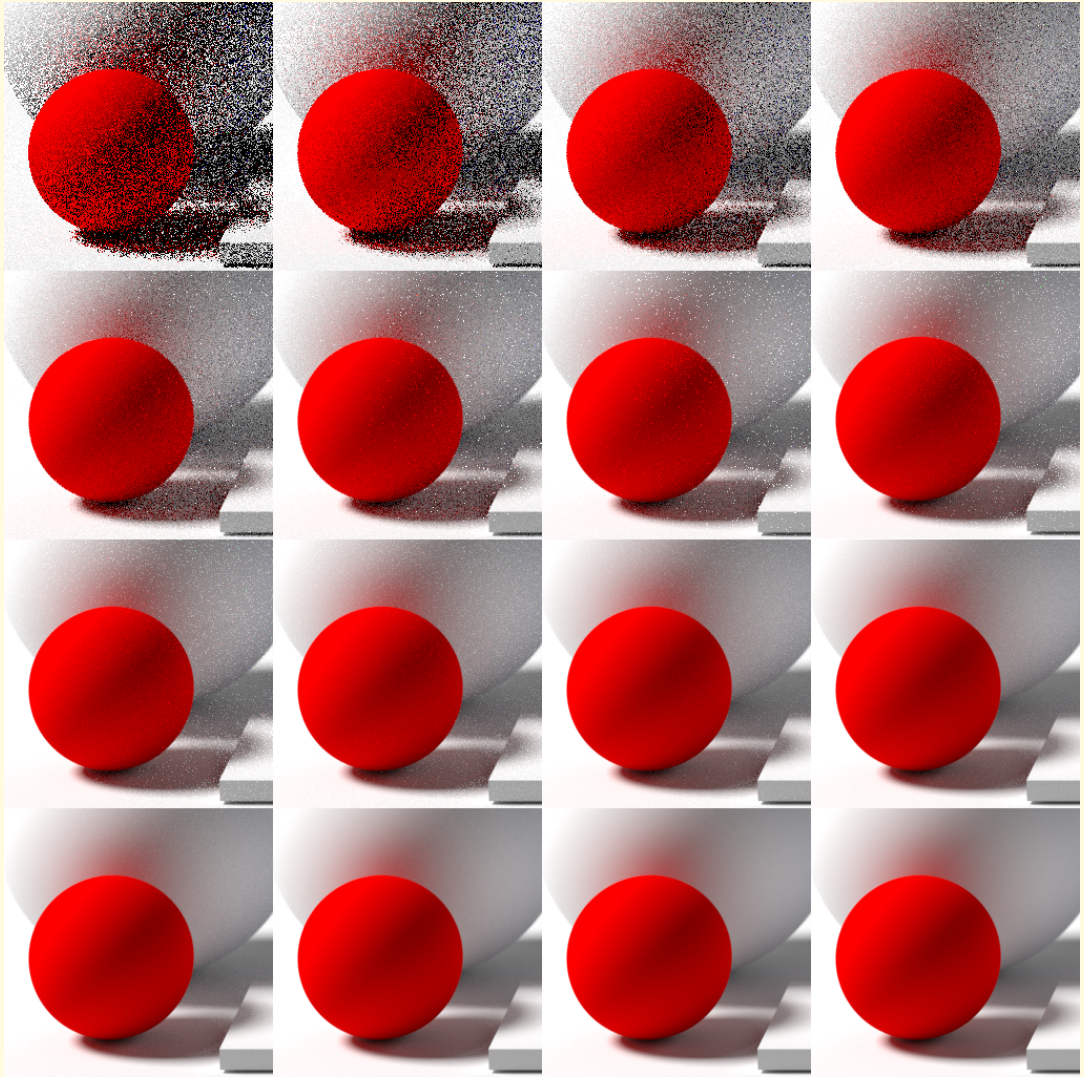
The Rendering Equation

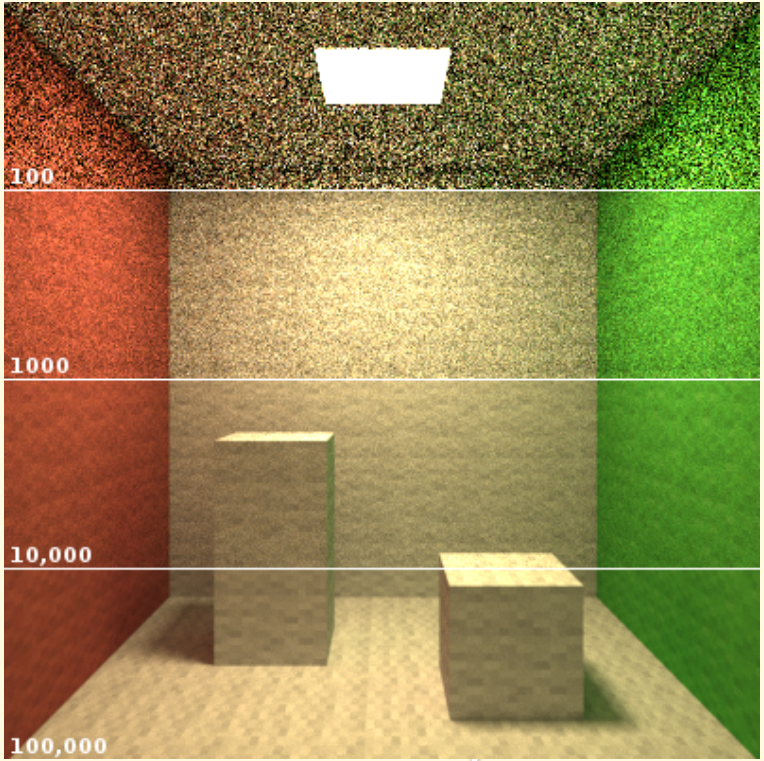


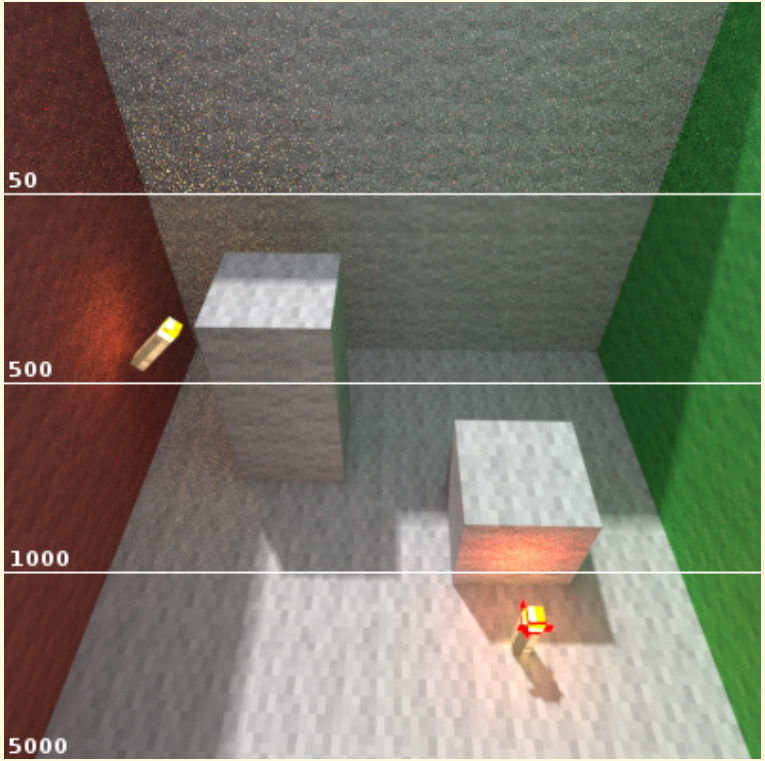
Monte Carlo

The more common approaches these days use Monte Carlo methods. Monte Carlo is the name of town in Monaco with a famous casino, and the term evokes the probabilistic nature of how the algorithm operates.

In Monte Carlo methods you attempt to render the most likely light paths by taking lots of samples, where the more samples you take the more likely you are to get an accurate rendering of a scene.

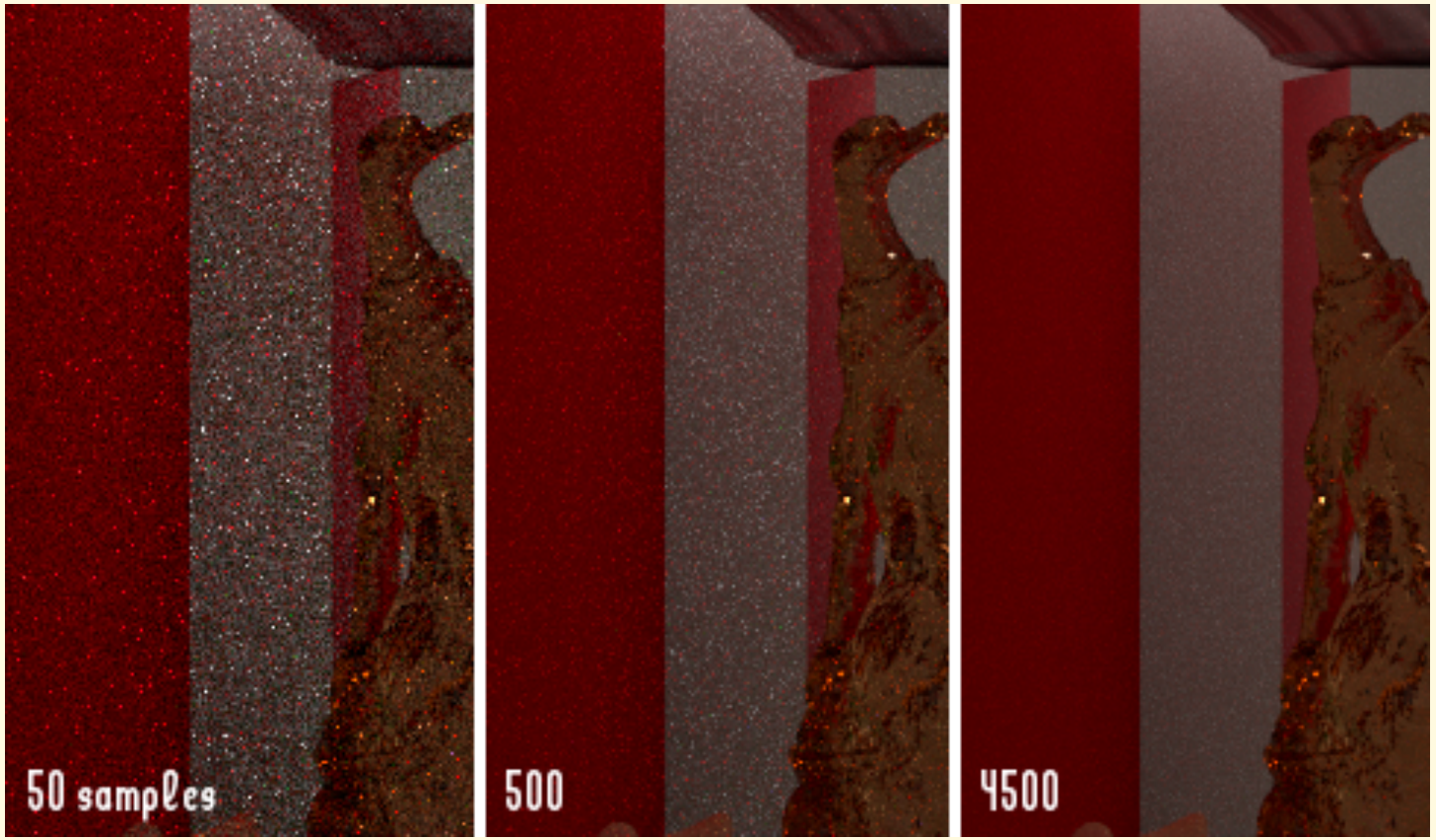






“Caustics”


RED BARRELS INC


50 samples
500
4500

Denoising

Some of the most advanced approaches attempt to create nice looking scenes quickly by taking less samples through 3D space, and then applying a fast 2D image denoising on the render buffer.

Real-time ray tracing makes use of sophisticated denoising algorithms, and is still an active area of research. The image denoising often uses a neural network that encodes a (temporally coherent) smoothing filter.

Input

Output

Ground Truth

Path tracing

Path Tracing is similar in spirit to ray tracing:

Rays are cast from a virtual camera and traced through a simulated scene.

Path tracing uses random sampling to incrementally compute a final image.

This makes it possible to render complex phenomena not handled in regular ray tracing.

However, it takes a longer time to produce a high quality path traced image.

Path tracing

Path tracing, sometimes referred to as Monte Carlo ray tracing, renders a 3D scene by randomly tracing samples of possible light paths.

Repeated sampling of any given pixel will *eventually* cause the average of the samples to converge on the correct solution of the rendering equation.

Path tracing

In the real world, objects and surfaces are visible due to the fact that they are reflecting light. This reflected light then illuminates other objects in turn.

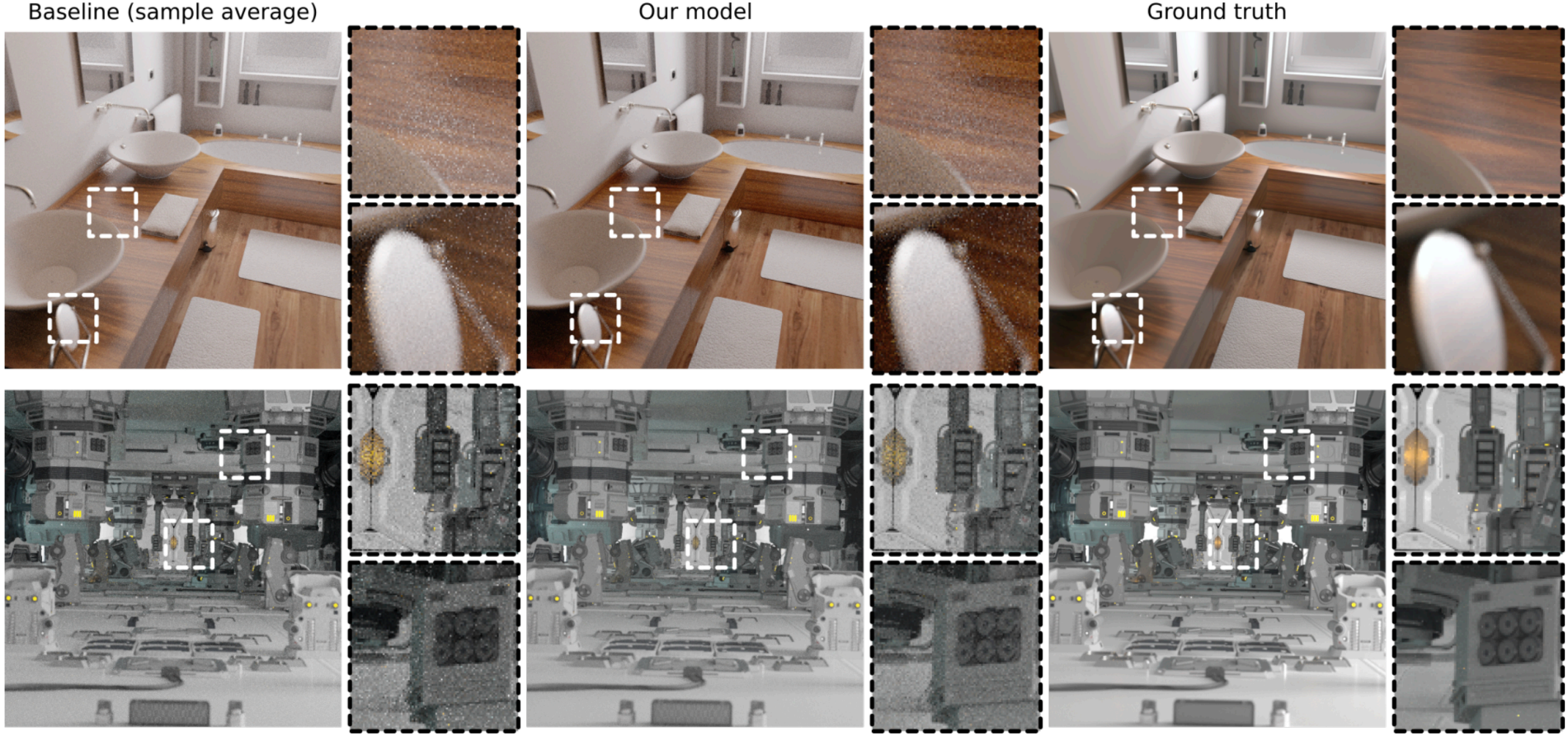
- For a given indoor scene, every object in the room must contribute illumination to every other object.
- There is no distinction to be made between illumination emitted from a light source and illumination reflected from a surface.
- The illumination coming from surfaces must scatter in a particular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

Path tracing

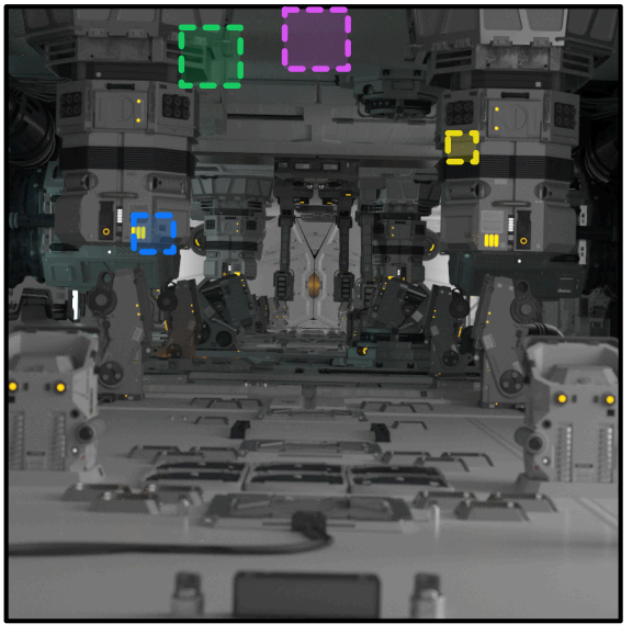
Doesn’t handle some things perfectly without additional computation:

- Caustics
- Subsurface scattering
- Fluorescence and iridescence, where light behaves differently at different spectrums.

A small number of smaller bright lights can lead to noise because paths are less likely to intersect those lights, and the ones that do get overweighted… leading to “fireflies"


Baseline (sample average)
Our model
Ground truth

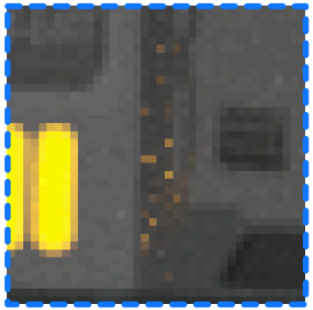
5-bounce transport



Indirect illumination

Fireflies

Noisy region 1



Noisy region 2

Speeding up path tracing

A 1997 paper by Eric Veach and Leonidas Guibas discusses "Metropolis light transport", a method of perturbing previously found paths in order to increase performance for difficult scenes.

- The core idea here is that rather than sampling at random, you mutate the previous paths that make it from the light to the camera in the fewest bounces.
- Use bidirectional ray tracing – both from the light ("shooting") and from each surface point ("gathering").

Speeding up path tracing

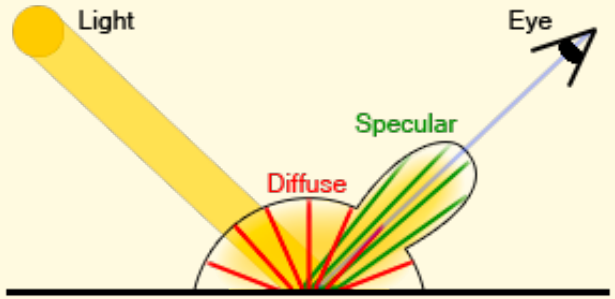
"Importance sampling":

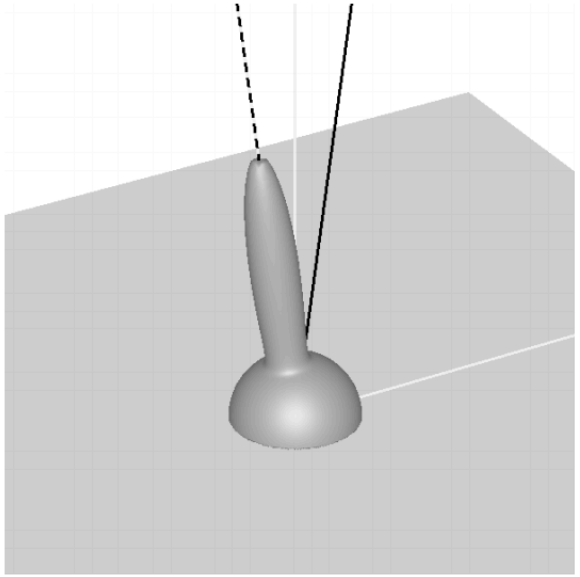
Decide how the sampling is more likely to occurs in the scene. Bias the direction of the rays to make sure most samples are likely to be brightly lit.

- On a material, we looked at BRDFs where more energy was seen from the specular highlighting — so sample there more often.

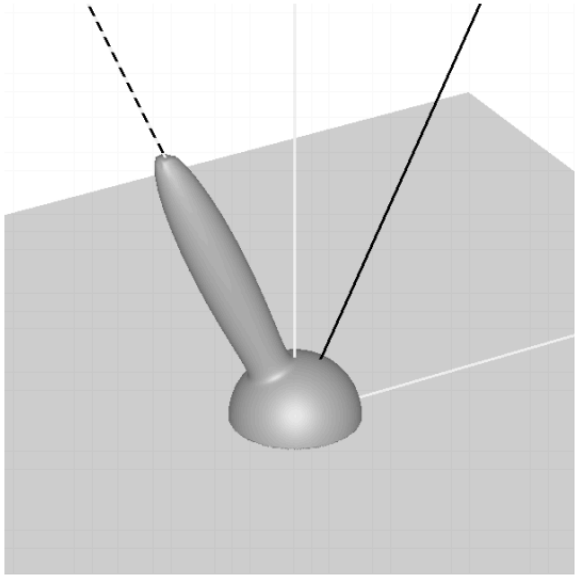
- From a light, create more samples from brighter lights

- Render a few samples at random and select the ones that contributed to brighter pixels and sample in those directions more often

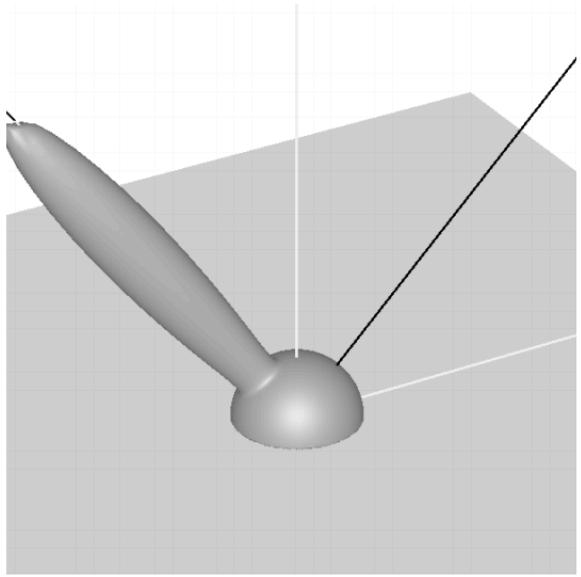




$\theta_i = 10°$



$\theta_i = 20°$



$\theta_i = 40°$

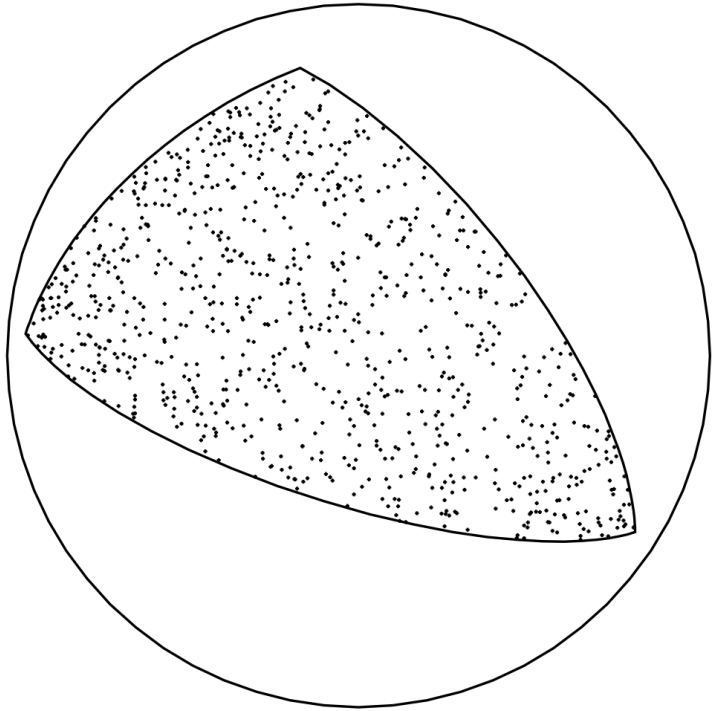
Speeding up path tracing

- When choosing randomly, use a random number generator that has nicer properties

- You may have heard the term “blue noise” which describes a way of more effective random sampling that won’t introducing artifacts like stripes or checkerboards that could appear if the sampling was more uniform.

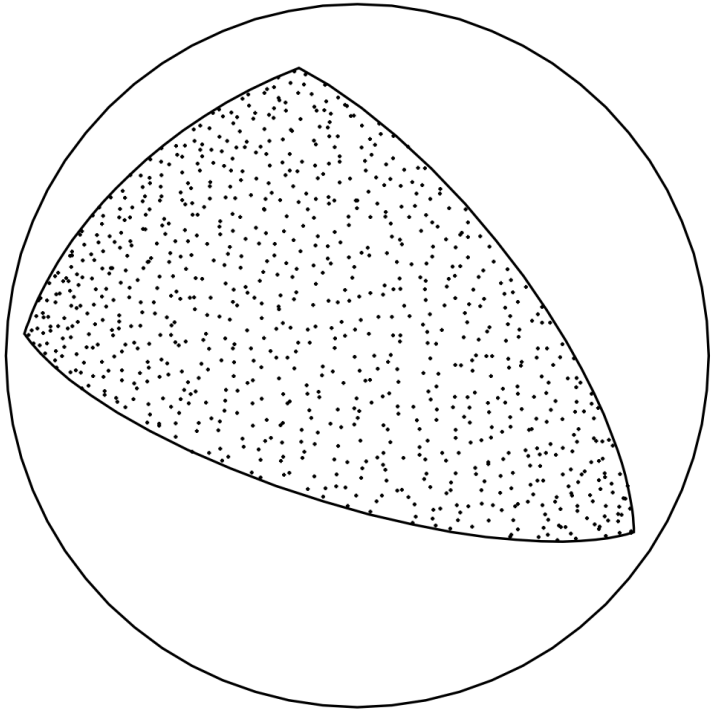
Blue noise was biologically motivated, John Yellott (1983) found the monkey photoreceptor cells on retinas of monkeys are ordered using blue noise.

uniform



"more clumpy"

stratified



"more even"

Radiosity

Radiosity uses techniques from dynamics and numerical simulation, based on “finite element methods”, an approach to discretizing energy through space (and which you may come across if you are doing fluid simulation).

The original version of the radiosity rendering algorithm (by Cindy M. Goral, Kenneth E. Torrance, Donald P. Greenberg and Bennett Battaile in 1984) assumed that all surfaces were “perfectly diffuse”:

“[Radiosity is…] a method which can be used to determine the intensity of light *diffusely reflected* within an environment.“ Goral et al.


Direct Illumination
Radiosity

Photon Mapping

A 1996 paper by Henrik Wann Jensen introduced Photon Mapping.

Photons (similar to a ray) emitted from lights… stored in a "photon map" when it hits a surface. The maps from the *light's* point of view

Only store the incoming energy from the light — NOT the radiance emitted from bouncing off of the material, (so less computationally expensive…)

Then do backwards ray tracing…

Use the photon map to approximate radiance if a "simple" material (diffuse)

If glossy, or more complex, then query the BRDF and take additional samples…

Additional steps: Shadow photons + Caustic mapping

Volumetric path tracing

Volumetric path tracing was first introduced in 1997 by Lafortune and Willems.

This method enhances the rendering of the lighting in a scene by extending the path tracing method with the effect of light scattering. It is used for photorealistic effects of participating media like fire, explosions, smoke, clouds, fog or soft shadows.

As in the path tracing method a ray gets followed backwards, beginning from the eye on, until reaching the light source. In volumetric path tracing scatter events can occur while these process. When a light ray hits a surface, a special amount of it can get scattered into the media.