# Global and Volumetric Lighting

Sean Brzoska and Katie LaRue

## Basic Overview

- Volumetric lighting is a technique used in 3D computer graphics to add lighting effects to a rendered scene. It allows the viewer to see beams of light shining across the environment.
- Global illumination is to account for light scatter when objects are illuminated by light sources and are refracted around the room, adding indirect lighting to all surfaces.

## Volumetric Lighting

## What is volumetric lighting?

- The representation of light hitting particles in the air.
- Density of particles usually varies depending on the situation and location. Clouds change their shape, fog and smoke are denser in some areas, mist is constantly moving.
- Generally pretty straightforward in concept and basic implementation, but gets more complicated the closer we get to photorealism.
- Map 3D noise to a location, inside a defined volume of space.

## How does it work, real life?

Volumetric lighting is the graphics implementation of Crepuscular rays. Crepuscular rays are sunbeams that originate when the sun is just below the horizon. Crepuscular rays usually appear orange because the path through the atmosphere at sunrise and sunset passes through up to 40 times as much air as rays from a high sun at noon.

Particles in the air scatter short-wavelength light (blue and green) more than longer-wavelength yellow and red light. The term crepuscular rays is sometimes extended to the general phenomenon of rays of sunlight that appear to converge at a point in the sky, irrespective of time of day.

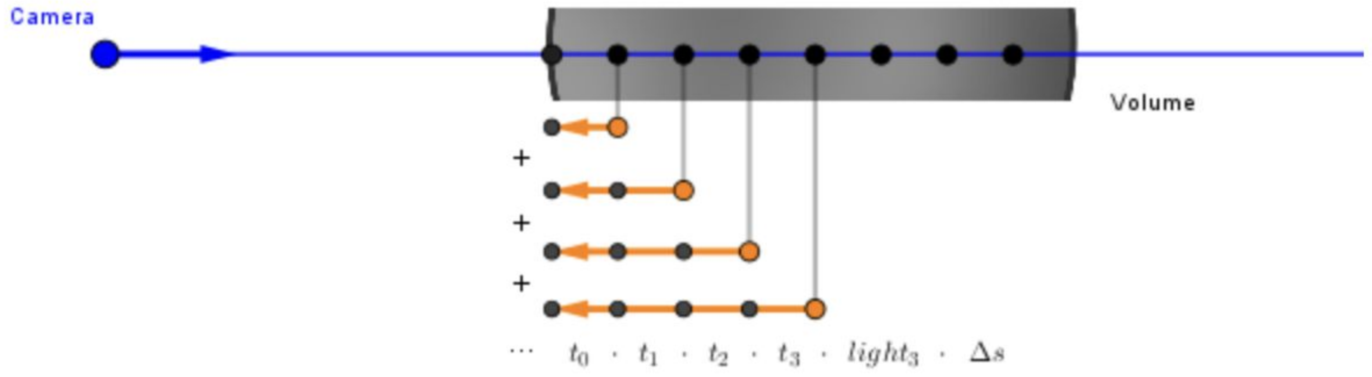
## How does it work, code-wise?

In a basic implementation, we can set a arbitrary volume with a specific density to represent our desired “air particulate form” ie cloud/fog/mist/air. The particles of our desired substance interact occasionally with the photons traveling through it. Generally speaking, a photon could either pass some particle and continue on its path or not.

To simulate this, we can walk through the volume and assess the percentage of photons that manage to travel on their straight path across some given length inside the volume. We also can calculate the percentage of photons that are bouncing around our specific volume, what percentage manages to continue through the medium, and what light manages to scatter out from our viewing ray.

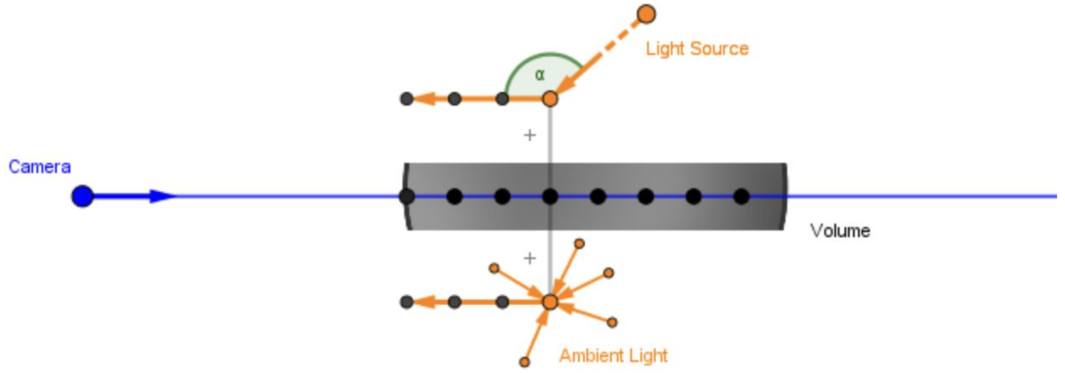
## What is the math?

- Transmittance
  - At each sample segment along viewing ray, we can take the cumulative length and use Beer's law (also known as Beer-Lambert's law) to calculate the transmittance.
  - τ(Δs)=e−σext · Δs where τ is transmittance, giving the percentage of light that was able to pass, σext is the material-specific coefficient for density and Δs is the distance travelled.


Camera
Volume
+
+
+
$\cdots t_0 \cdot t_1 \cdot t_2 \cdot t_3 \cdot light_3 \cdot \Delta s$

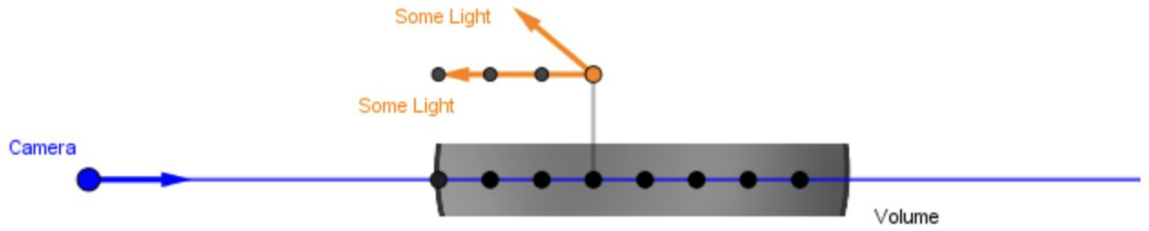
## What is the math?

- In Scattering
    - We assume that lot of the light is coming in from the direction of some light source and that direction can certainly be one source of light. We can also assume there is an amount of ambient light bouncing around in the volume and can be approximated with a constant.



## What is the math?

- Out scattering
  - The out scattering is the percentage of the light that gets redirected from the viewer at the current step. This percentage is actually a sum of the two coefficients σabsorption which specifies how much light is absorbed per unit of volume and σscattering which indicates the amount scattered. To calculate out scattering, we take σscattering and scale it by the density of the medium.
  - Out scattering values are very small, about 0.04 m-1 for a dense cloud and can be in the range of 0.01 to 0.06 m-1, and the absorption coefficient would be from 0.000005 to 0.00003 m-1.



## What does it look like?

Computer Graphics Learning - Volumetric Rendering

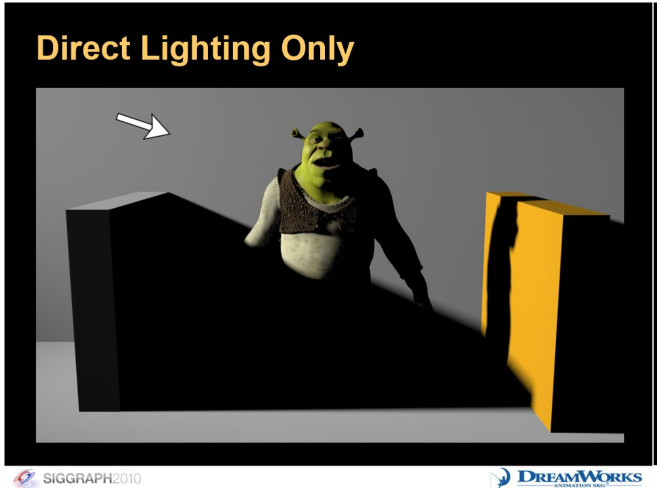
## What can you do with it? Basic

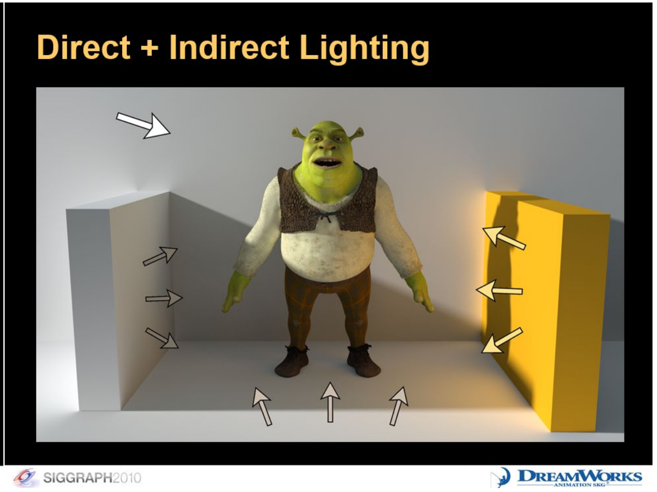
## What can you do with it?
## Subtle

## Global illumination

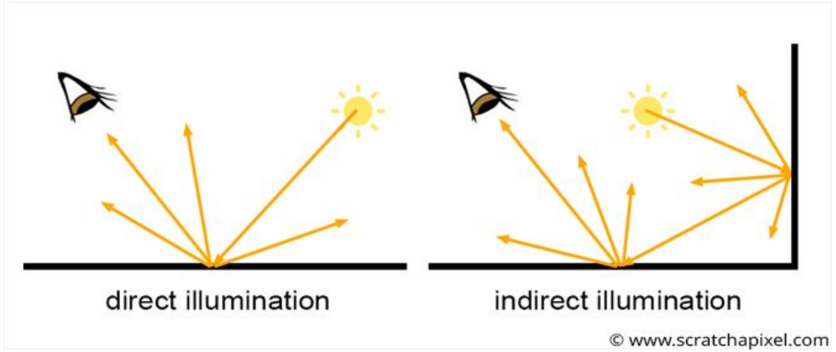
## What is Global Illumination?

Global Illumination accounts for the light scattered from directly lit sources onto all other surfaces.

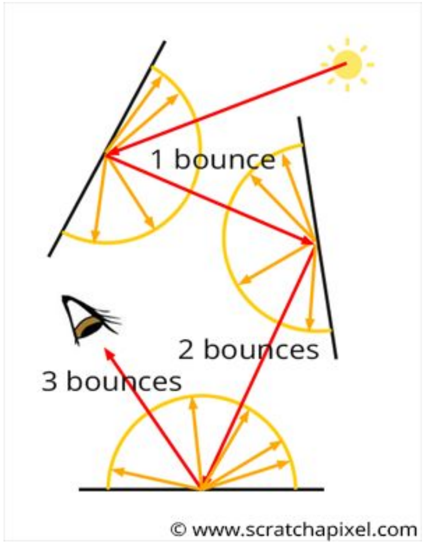




## Indirect lighting

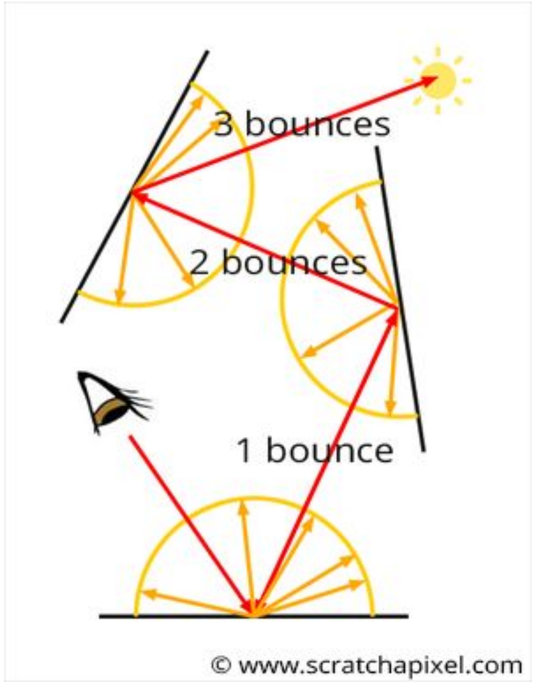


- In the real world light will travel from its source to the eye. This is known as our light path.
- This light path my not be direct, and can bounce several times.
- Going from the lightsource to the eye is known as forward tracing.



## Avoid forward tracing, do it backwards!

- Forward tracing is highly inefficient in determining if a ray travels from our lightsource to our eye!
- We must do backward tracing
- On each step, we can determine if our bounced ray has a direct path to the lightsource

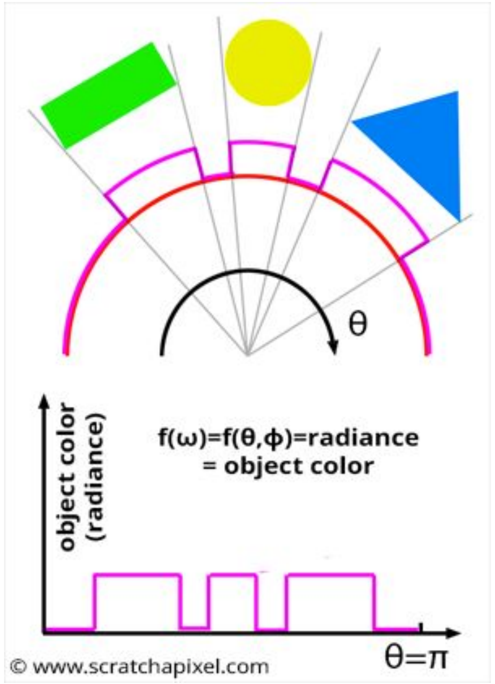


## Determining indirect lighting

When determining the color of an object that is indirectly lit. We look at all the objects in the hemisphere around the ray intersect point.

Ideally we will have an infinite number of points to cover every surface we see around our hemisphere to determine the true illumination of our point.

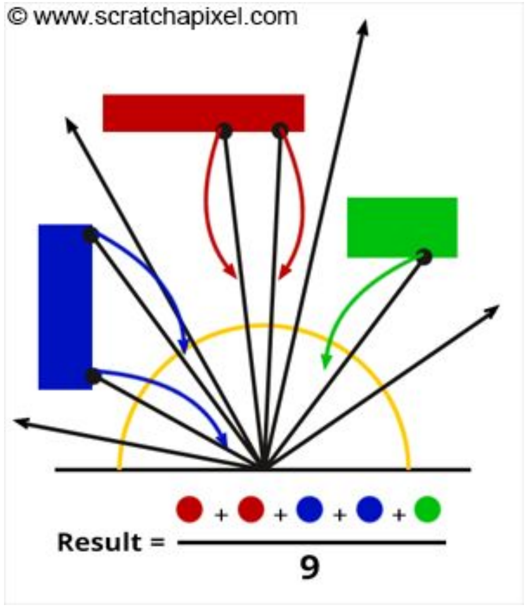
This is impractical, so we use Monte Carlo to estimate.



## Monte Carlo estimator

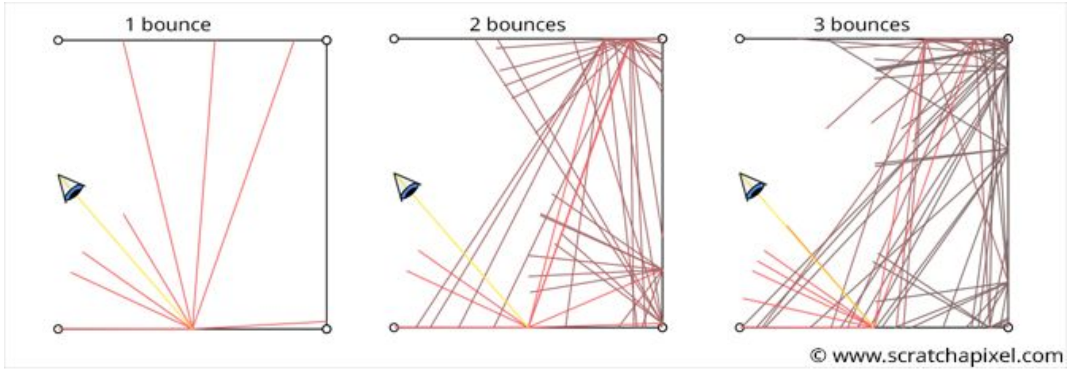
In this case, we sample 9 random rays in our hemisphere from our intersect point P. If the rays hit an illuminated object we return the color. Then divide by the total number of rays.

$$\text{Gather Light} \approx \frac{1}{N}\sum_{n=0}^{N} \text{castRay}(\text{P}, \text{randomDirectionAboveP})\,.$$



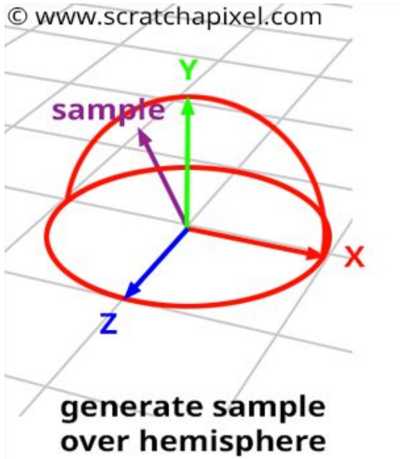
## Bouncing

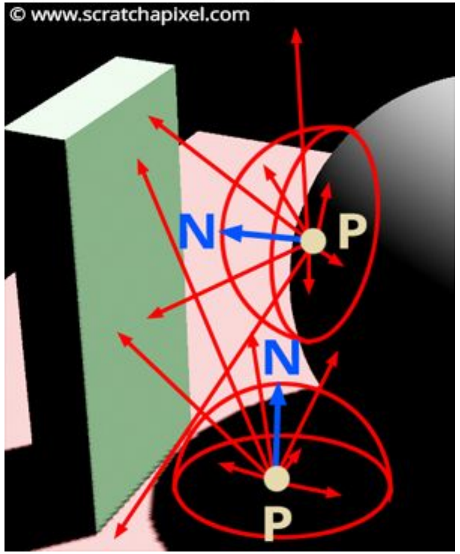
- To get more accurate results we can add depth, or bounces to our generated rays.
- This will add magnitudes of complexity for each bounce.



# Putting it together

- Return 0 if max depth
- Generate rays in the hemisphere
- Convert rays to world space
- Trace the rays to see what they hit, get color
- Recursively call to a higher depth
- Determine the intensity via dot product with ray and normal
- Some all ray colors and divide by the number of rays





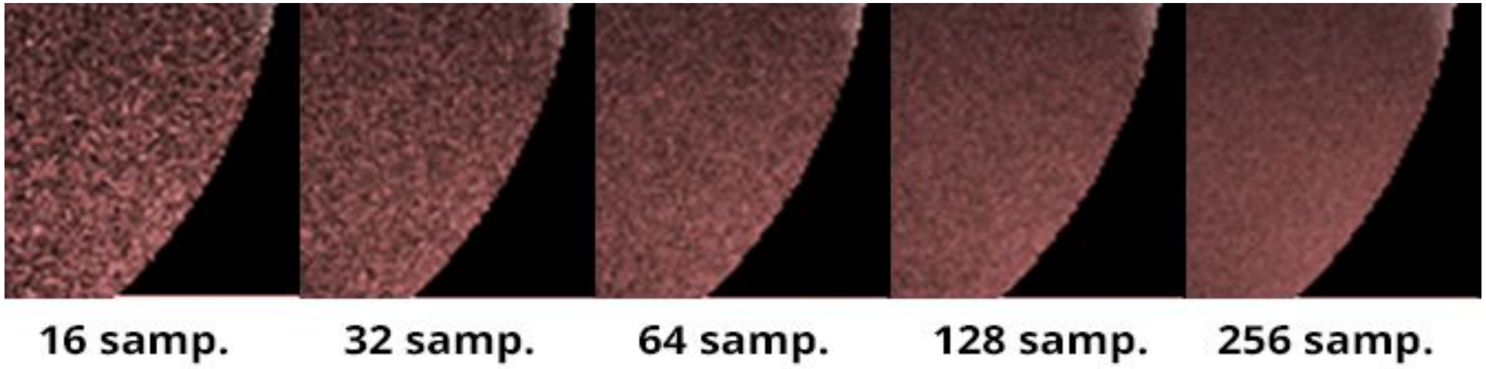
## Performance on Sampling and Bounces

We now know how to generate indirect lighting, but at what cost? If we have 100 sampled rays and 5 bounces, how many rays have we generated for this single pixel's color?

A. 5*100
B. $(5*100)^2$
C. $100^5$
D. $5^{100}$

## Performance on Sampling and Bounces

Keeping our sample size down might help performance, but it hurts our results.

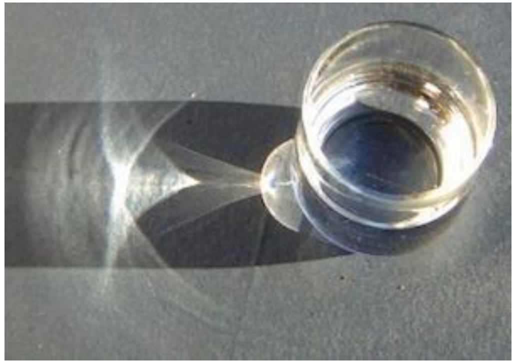


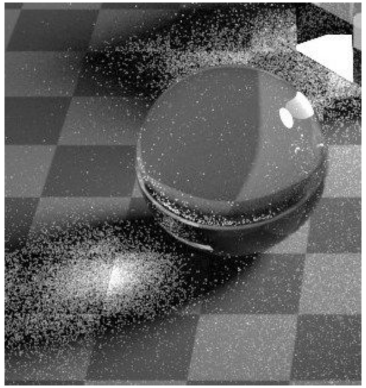
## Global Illumination

https://madebyevan.com/webgl-path-tracing/

## Limitations of Backward Tracing

- Caustics are the result of light rays being focused on specific points or regions of space.
- These are nightmares for our backward tracing method



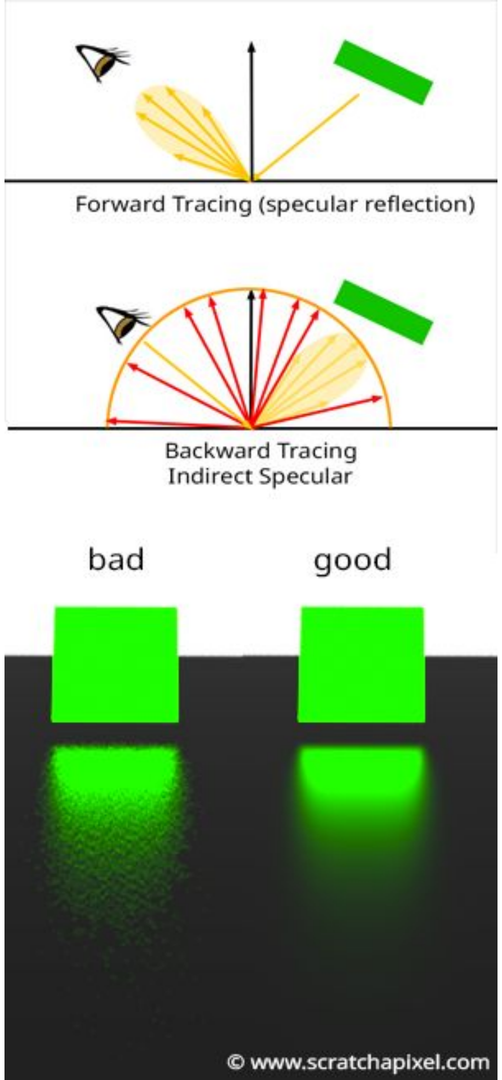


# Non Diffuse indirect lighting

What if our surface is glossy?

Simple, instead of sampling randomly, sample off the reflection ray!

We can do this by creating a cone around our reflected ray, then take samples randomly or uniformly inside the cone.



## Questions?