21 - Sampling

Whitted-style ray tracing - Presented at SIGGRAPH 1979 by Turner Whitted.
Cook-style ray tracing - Presented at SIGGRAPH 1984 by Robert Cook, "Distributed Ray tracing".
The rendering equation - Path tracing - SIGGRAPH 1986, James T. Kajiya.

This lectures topic is how to go from whitted-style ray tracing to the rendering equation, computing indirect term

The rendering equation $= L_o(\omega_o) = \int_\Omega L_i^{direct}(\omega_i)\cos\theta_i f_r(\omega_i,\omega_o)\,d\omega_i$

$+ \int_\Omega L_i^{indirect}(\omega_i)\cos\theta_i f_r(\omega_i,\omega_o)\,d\omega_i$

Split up because indirect lightning complexity, direct being light sources and indirect being light from objects

Whitted-style: $L_o(\omega_o) \approx \sum_i L_i^{direct}(\omega_i)\cos\theta_i f_r(\omega_i,\omega_o)$

$+ \int_\Omega L_i^{indirect}(\omega_i)\cos\theta_i f_r(\omega_i,\omega_o)\,d\omega_i$

$f_r(\omega_i,\omega_o) = K_d + K_s\frac{(\cos\varphi_i)^\alpha}{\cos\theta_i}$ ← Blinn/phong

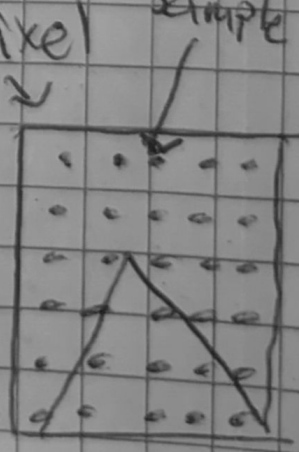
If perfect reflective surface, K_d = black and α = infinity, so the specular term becomes zero unless $\cos\varphi_i = 1$. So the BRDF can be simplified:

$$f_r(\omega_i,\omega_o) \approx \begin{cases} K_s/\cos\theta_i, & \text{if } \omega_i = \omega_r \\ 0, & \text{otherwise} \end{cases}$$

So the indirect term simplifies to $L_i^{indirect}(\omega_r)K_s$

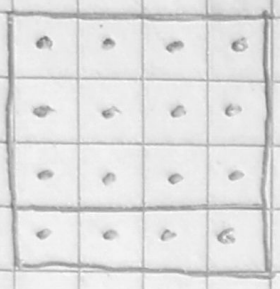
Sampling - in the case of AA

Sampling is the process of generating a good estimation of the color values at edges etc. (anti-aliasing). The ideal case is to take the integral of the pixel area, $\int_{pixel\ area} f(x)dx$, that's in most cases impossible so we take average $\frac{1}{N}\sum_i^N f(x_i)$, where N is the amount of samples.

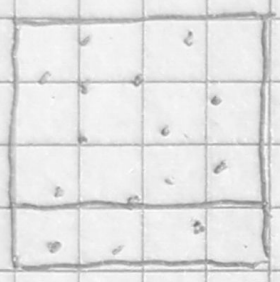


Monte carlo sampling

Problem with having a uniform set of samples is that we limit the amount of generated colors in some cases, for example:

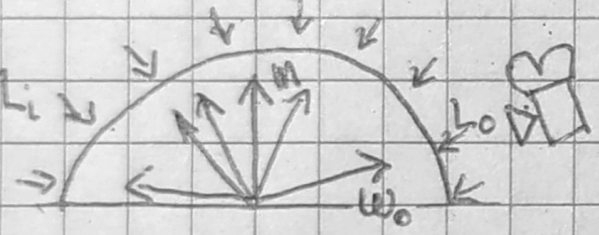


In this case the triangle edge is paallell to the samples, so we only get 5 color values from 16 samples.



Monte carlo sampling fixes this by having more random samples.

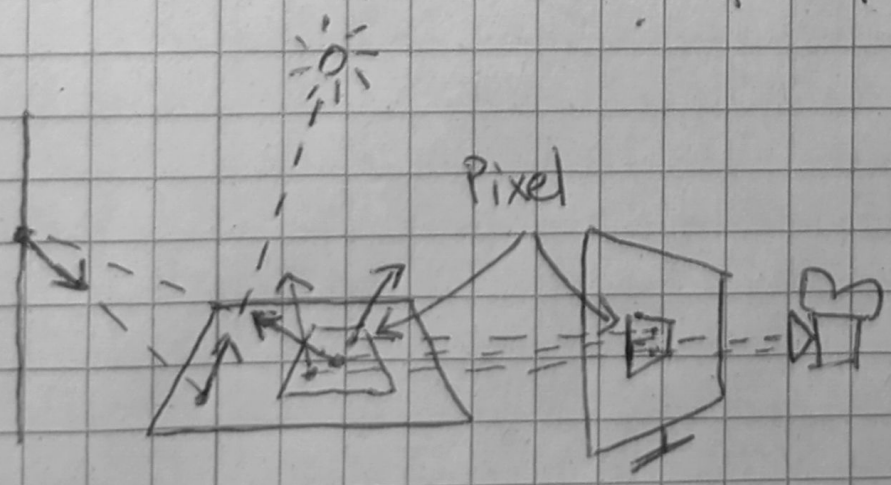
Monte Carlo ray tracing (cook-style/distributed)

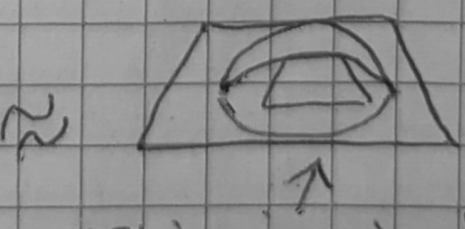


We can also apply Monte Carlo sampling to ray tracing, taking random ray samples to compute either direct light (creating soft shadows) or indirect light (creating super realistic images). The problem with this is that it becomes exponentially more computationally expensive the more bounces we have.

Path tracing

Path tracing solves this problem by setting N=1, and instead takes multiple pixel samples with N=1.





This approximates taking the integral for the pixel. The more samples per pixel (SPP) we have, the less noise.

Denoising

The problem with path tracing is that it will generate a lot of noise if we have few pixel samples, but be computationally expensive if we have many. We can apply a denoising filter after the rendering to combat the noise.