Illumination 101 Pixar Animation Studios

- Light integration optimization
  - Multiple Importance Sampling
  - Control Variates
  - Problems/Solutions with Control Variates
- Modern Path Tracing Considerations

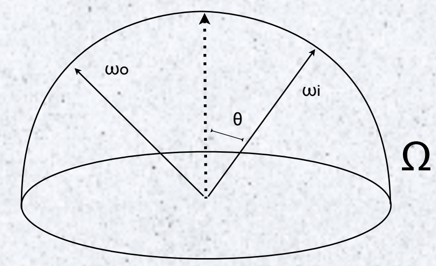
Solving the rendering equation

$$L(x,\omega_o) = \int_\Omega f(x,\omega_i,\omega_o) L(x,\omega_i) cos(\theta) d\omega$$

L(x, ωo) : outgoing radiance

L(x, ωi) : incoming radiance

f(x, ωi, ωo) : BRDF



Physically Based Lighting is about solving the rendering equation.
Obviously, we use Monte-Carlo technique to compute this integral.

- Multiple Importance Sampling
- Control Variates

brute force is good, always work
but we want to speedup the process
principal piece -> MIS

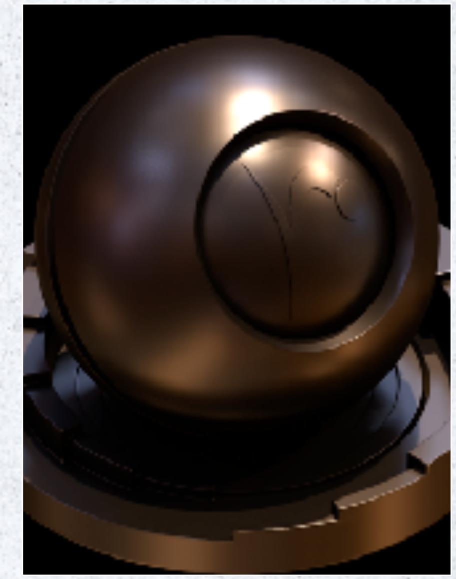
- Sample from lobes using BRDF shaders
- Sample from lights using light shaders
- Combine samples using Multiple Importance Sampling

BRDF sampling only

- good sampling everywhere except in the highlights where BRDF sampled directions miss bright spots
- fuzzy shadows

Light Sampling Only

- good sampling in the highlights, well defined shadows
- noise in low brightness region where the sample directions don't match the BRDF peak



With MIS...

Best of both worlds!

- Classic variance reduction technique
- Here we rely on the fact that some light sources can provide an analytical integral of their illumination (without shadowing)

Control variates is a classic Monte Carlo technique.
We use it here to reduce variance in the direct illumination integral, assuming that lights can make use of some sort of analytic integral of their illumination.
Note that visibility/shadowing is unknown and needs tracing (thus sampling) to the full scene.

$$\int fv = \int fv + \alpha\left(\int f - \int f\right) = \alpha \times \int f + \int f \times (v - \alpha)$$

$$F = \int f \qquad \text{analytic}$$

$$\int fv = \alpha \times F + \int f \times (v - \alpha)$$

This is the canonical form of control variates. v is the visibility. f is the light, for which we can somehow provide an analytic integral F.
Alpha is some constant.
The remaining integral on the right still requires some form of Monte-Carlo sampling, both on the light f (to get the exact direction towards it) and to get the visibility v to the point being shaded from that direction.

$$\alpha = 1$$

$$\int fv = F + \int f \times (v - 1)$$

Simplest variant.
Alpha = 1.

The problem with this is that, since v is alway 1 or 0 (thus less than 1), the remaining integral can be negative, and the overall sum to F can produce negative values, negative pixels.
This is unbiased obviously, so in the end, this provides a valid answer, but for low sample count, it may be problematic.

- Use of average visibility to further reduce variance

One solution to this issue is to make use of the average visibility as we do the sampling.

$$\alpha = \bar{v}$$

$$\int f v = \bar{v} F + \int f \times (v - \bar{v})$$

So this time, we take alpha, the constant, equal to the average visibility.
Advantage, v and v_ are much closer together, even at low sample count, and the final pixels tend to have less negative values.
Other advantage, if v_ is 0, ie all vs are 0, so we are all in shadow, then this returns exactly 0. If v_a is 1, ie all vs are 1, we are in full light, then this returns exactly F, ie the analytic integral, without noise. We thus have limited the variance to the penumbra regions, which we cannot predict ahead of time.

- Sphere: sub-hemispherical light source integration

  [Area Light Sources for Real-Time Graphics - TR 1996]

- Rect, Disk: polygonal light source analytical integration
- Dome: image space convolution

Many techniques, some exact, some approximatives…
Good reference, including the sub-hemerispherical integration, is the John Snyder Tech Report from 1996, seen as useful for real-time (we will come back to that).

- Rendering Equation:

$$L=\int_{\Omega} L f_r \cos(\theta) d\omega$$

- If the BRDF fr is constant (Lambert for surfaces, Kajiya for hair) and
If the light is visible for the whole hemisphere, then the equation becomes:

$$L=f_r \int_{\Omega_{dome}} L_{dome} \cos(\theta) d\omega$$

For dome light, we can make use of the fact that, for some constant brdfs, the constant can be moved out and the integral can be pre-computed.

Pre-computed convolution for a Dome light texture

This pre-convolution is slow, so it is done off-line, as an adjoint channel to the Dome texture.

- 1760: Lambert's irradiance formula

For polygonal light sources, we can make use of Lambert's irradiance formula (from 1760!).

- 1760: Lambert's irradiance formula

$$\Phi(r) = \frac{M}{2\pi} \Sigma \Theta_i \Gamma_i$$

- 1760: Lambert's irradiance formula
- Arvo's Double Axial Moments

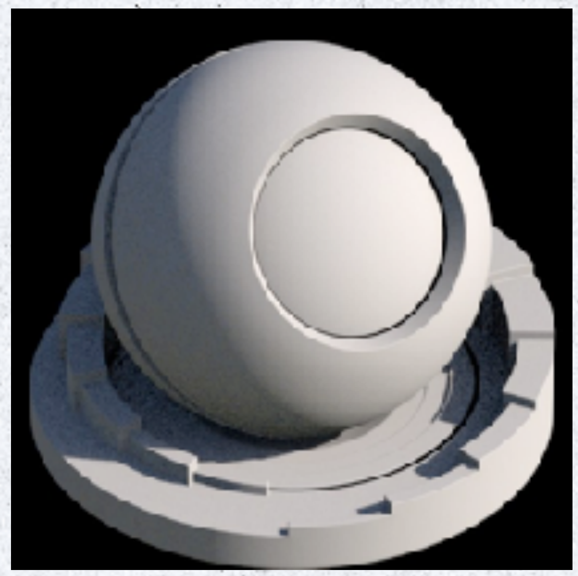
  [Applications of Irradiance Tensors to the Simulation of Non-Lambertian Phenomena - Siggraph 1995]

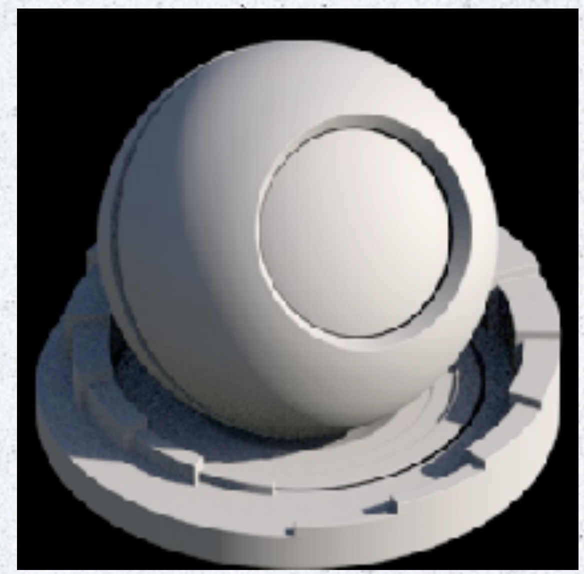
Its generalization is James Arvo's Double Axial Moments in 1995

Approximate solutions:

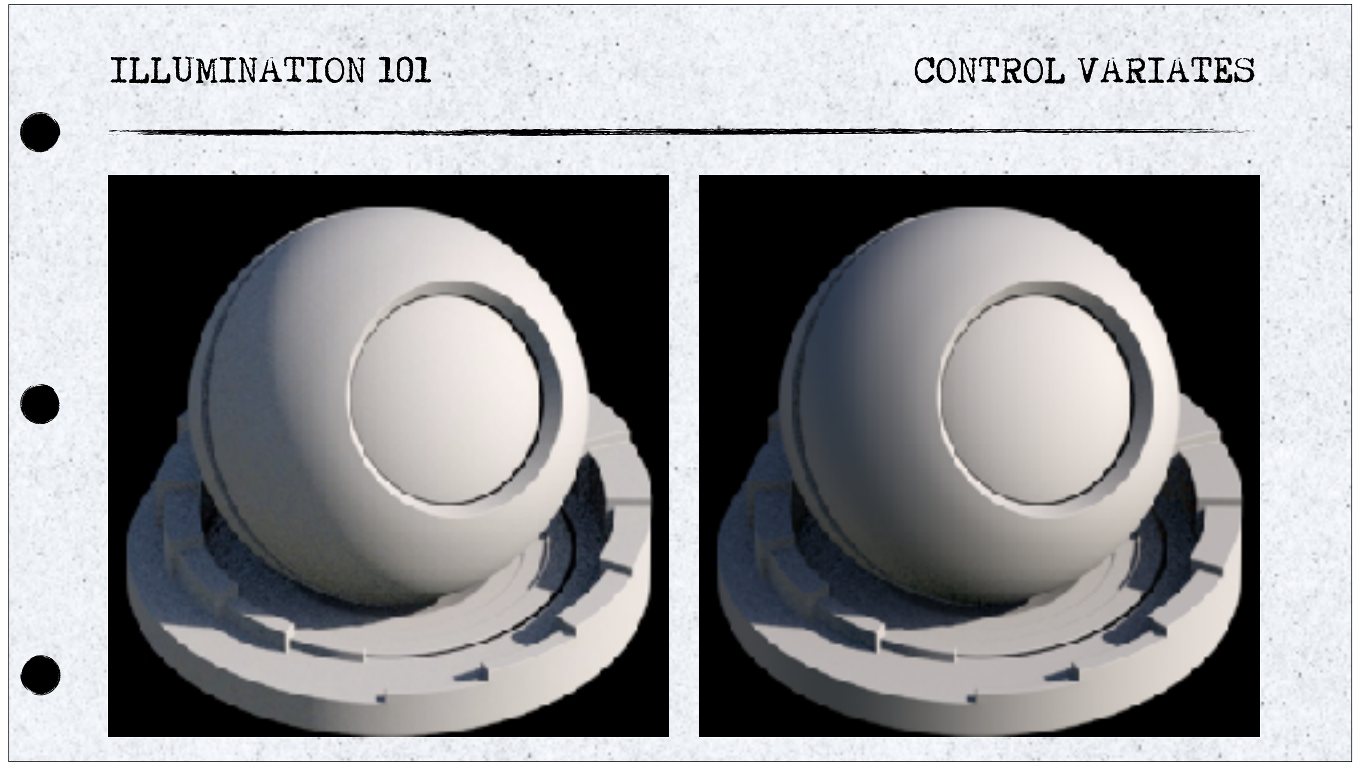
- Spherical Harmonics
  [An Efficient Representation for Irradiance Environment Maps - Siggraph 2001]
  [Analytic Tangent Irradiance Environment Maps for Anisotropic Surfaces - EGSR 2012]
- Zonal Harmonics
  [Integrating Clipped Spherical Harmonics Expansions - Siggraph 2018]
- Linearly Transformed Cosines
  [Real-Time Polygonal-Light Shading with Linearly Transformed Cosines - Siggraph 2016]
  [A spherical cap preserving parameterization for spherical distributions - Siggraph 2017]

Here I list some approximative solutions.
The point is that we want the analytic integral to be very fast, otherwise there would been point in comparison to pure MC sampling.

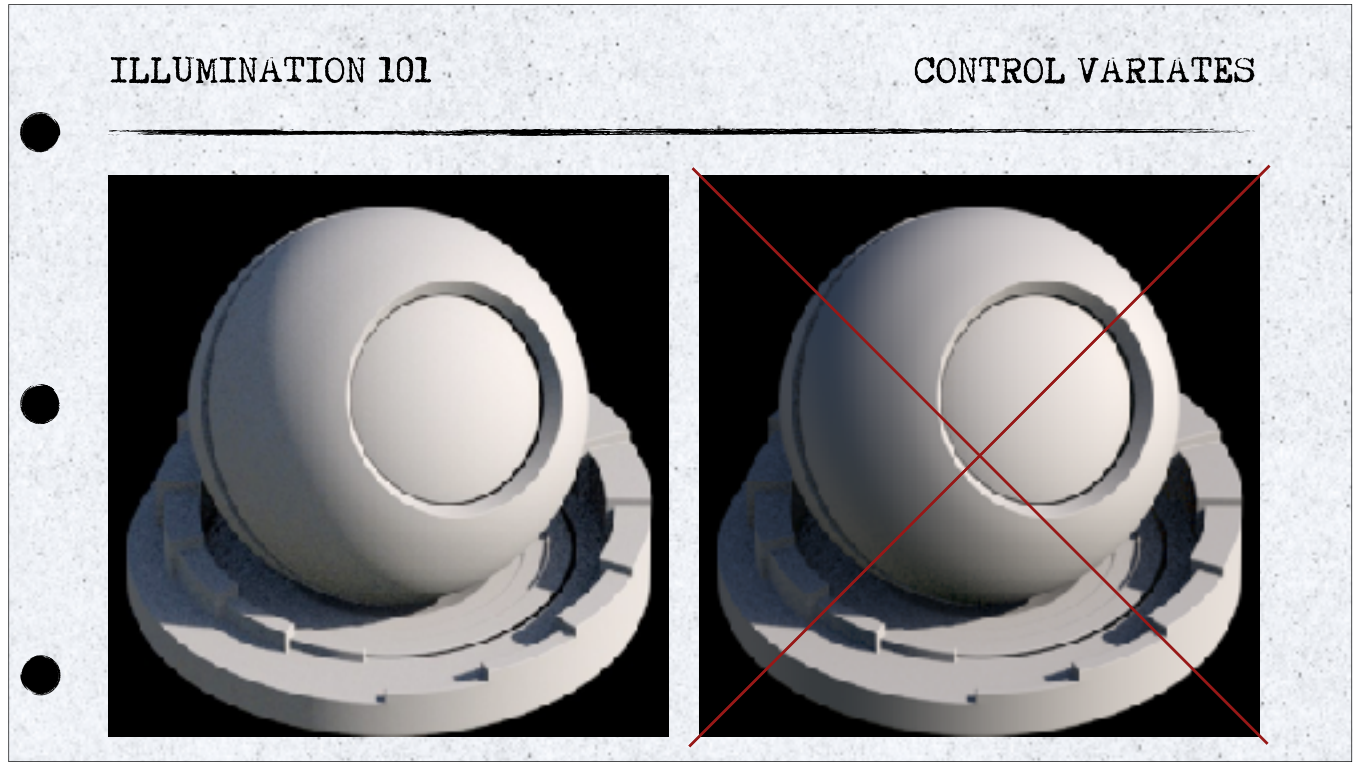




This is a Dome light with the convolution pre-computed in texture space.

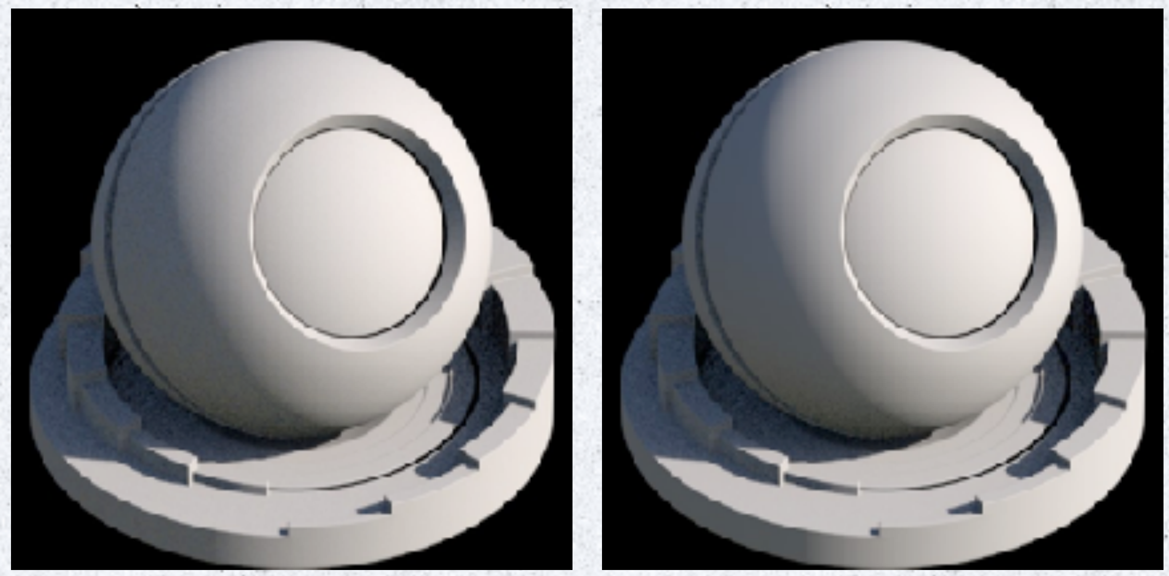


If done approximately with Spherical Harmonics, here is what we get, the shadow transition is not respected…

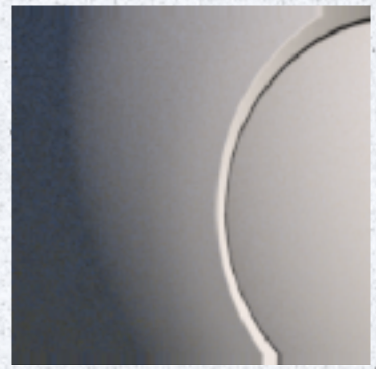


So in our domain of off-line film production, as opposed to games where speed and memory are uber critical, we do not like that !
(Now note that this was the case for SH - more modern representations, such as the mentioned Zonal Harmonics or the Transformed Cosines, do not suffer as much from being poor approximations)

ILLUMINATION 101 CONTROL VARIATES



Here is again our convolution texture introduced with control variates.



If noise reduction was not obvious…

- Use of average visibility to further reduce variance

So we are happy…

- Use of average visibility to further reduce variance
- But... this is biased...

However there is a big problem.
This approach is biased, consistent but biased.

$$\alpha = \bar{v}$$

$$\int fv = \bar{v}F + \int f \times (v - \bar{v})$$

Let’s look into it.

$$\int fv = \bar{v}F + \int f \times (v - \bar{v})$$

$$\begin{aligned}
&\mathbb{E}\Big[\frac{1}{N}\sum_{m=1}^{N} V_m \mathbb{E}[F] + \frac{1}{N}\sum_{n=1}^{N} F_n(V_n - \frac{1}{N}\sum_{m=1}^{N} V_m)\Big] \\
&= \frac{1}{N}\mathbb{E}\Big[\sum_{m=1}^{N} V_m \mathbb{E}[F]\Big] + \frac{1}{N}\mathbb{E}\Big[\sum_{n=1}^{N} F_n(V_n - \frac{1}{N}\sum_{m=1}^{N} V_m)\Big] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \mathbb{E}[FV] - \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m=1}^{N}\mathbb{E}[F_n V_m] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \mathbb{E}[FV] - \frac{1}{N^2}\sum_{n=1}^{N}\sum_{m\neq n}^{N}\mathbb{E}[F_n V_m] - \frac{1}{N^2}\sum_{n=1}^{N}\mathbb{E}[F_n V_n] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \mathbb{E}[FV] - \frac{N(N-1)}{N^2}\mathbb{E}[F]\mathbb{E}[V] - \frac{1}{N}\mathbb{E}[FV] \\
&= \frac{1}{N}\mathbb{E}[V]\mathbb{E}[F] + \frac{N-1}{N}\mathbb{E}[FV]
\end{aligned}$$

Here is our estimator.
For speed, we are making use of the same samples to compute the average visibility and the light directions. This means they are not independent.
And we end up with a (slightly) biased estimator.
(We could of course independently compute v_, but this is expensive, as this is the part that requires heavy ray-tracing into the scene).

$$\int f v = \bar{v} F + \int f \times (v - \bar{v})$$

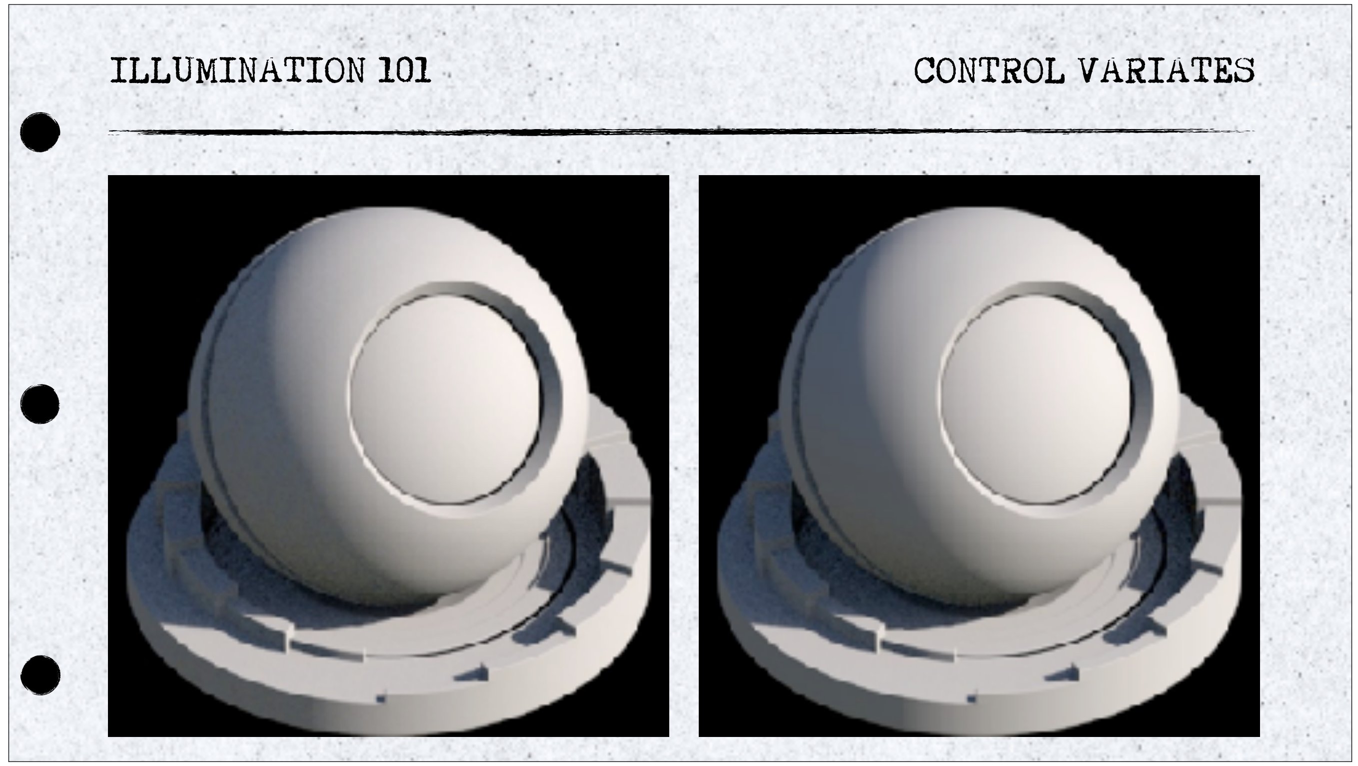
$$
\begin{aligned}
&\mathbb{E}\left[\frac{1}{N}\sum_{m=1}^{N} V_m \mathbb{E}[F] + \frac{1}{N-1}\sum_{n=1}^{N} F_n\Big(V_n - \frac{1}{N}\sum_{m=1}^{N} V_m\Big)\right] \\
&= \frac{1}{N}\mathbb{E}\left[\sum_{m=1}^{N} V_m \mathbb{E}[F]\right] + \frac{1}{N-1}\mathbb{E}\left[\sum_{n=1}^{N} F_n\Big(V_n - \frac{1}{N}\sum_{m=1}^{N} V_m\Big)\right] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \frac{N}{N-1}\mathbb{E}[FV] - \frac{1}{N(N-1)}\sum_{n=1}^{N}\sum_{m=1}^{N}\mathbb{E}[F_n V_m] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \frac{N}{N-1}\mathbb{E}[FV] - \frac{1}{N(N-1)}\sum_{n=1}^{N}\sum_{m \neq n}^{N}\mathbb{E}[F_n V_m] - \frac{1}{N(N-1)}\sum_{n=1}^{N}\mathbb{E}[F_n V_n] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \frac{N}{N-1}\mathbb{E}[FV] - \mathbb{E}[F]\mathbb{E}[V] - \frac{1}{N-1}\mathbb{E}[FV] \\
&= \mathbb{E}[FV]
\end{aligned}
$$

There is a simple "trick" to fix it. Just scale the sampled integral on the right by 1/(N−1) in place of 1/N. Now this is unbiased, and still cheap. (Note that this is not exactly true if one makes use of QMC or stratified sequences, even with extra randomization, as each sample is not fully, by definition, independent of the other ones).

$$\int fv = \bar{v}F + \int f \times (v - \bar{v})$$

$$\begin{aligned}
&\mathbb{E}\Big[\frac{1}{N}\sum_{m=1}^{N} V_m \mathbb{E}[F] + \frac{1}{N-1}\sum_{n=1}^{N} F_n\big(V_n - \frac{1}{N}\sum_{m=1}^{N} V_m\big)\Big] \\
&= \frac{1}{N}\mathbb{E}\Big[\sum_{m=1}^{N} V_m \mathbb{E}[F]\Big] + \frac{1}{N-1}\mathbb{E}\Big[\sum_{n=1}^{N} F_n\big(V_n - \frac{1}{N}\sum_{m=1}^{N} V_m\big)\Big] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \frac{N}{N-1}\mathbb{E}[FV] - \frac{1}{N(N-1)}\sum_{n=1}^{N}\sum_{m=1}^{N}\mathbb{E}[F_n V_m] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \frac{N}{N-1}\mathbb{E}[FV] - \frac{1}{N(N-1)}\sum_{n=1}^{N}\sum_{m\neq n}^{N}\mathbb{E}[F_n V_m] - \frac{1}{N(N-1)}\sum_{n=1}^{N}\mathbb{E}[F_n V_n] \\
&= \mathbb{E}[V]\mathbb{E}[F] + \frac{N}{N-1}\mathbb{E}[FV] - \mathbb{E}[F]\mathbb{E}[V] - \frac{1}{N-1}\mathbb{E}[FV] \\
&= \mathbb{E}[FV]
\end{aligned}$$

There is a simple “trick” to fix it. Just scale the sampled integral on the right by 1/(N−1) in place of 1/N. Now this is unbiased, and still cheap. (Note that this is not exactly true if one makes use of QMC or stratified sequences, even with extra randomization, as each sample is not fully, by definition, independent of the other ones).



In truth, our image from before was computed that way, with average visibility and its rescaling.

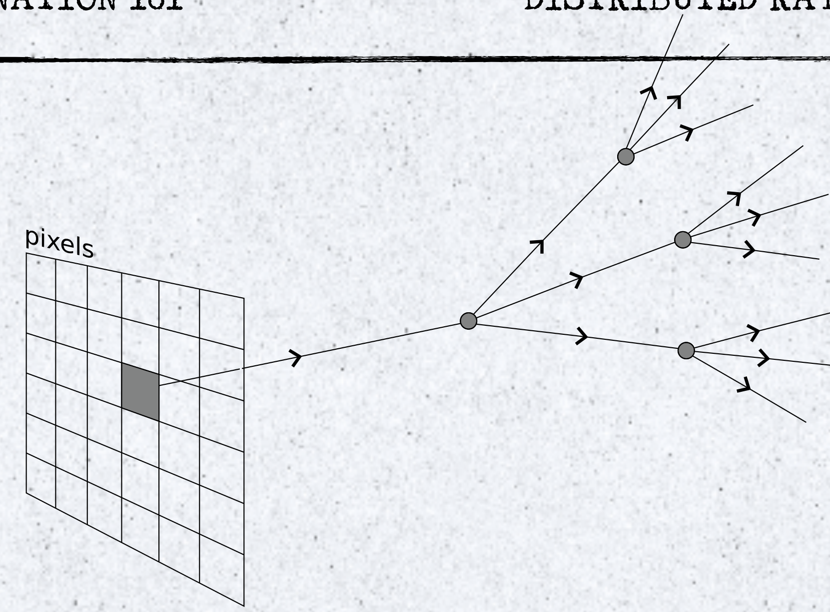
ILLUMINATION 101 INTO PATH-TRACING

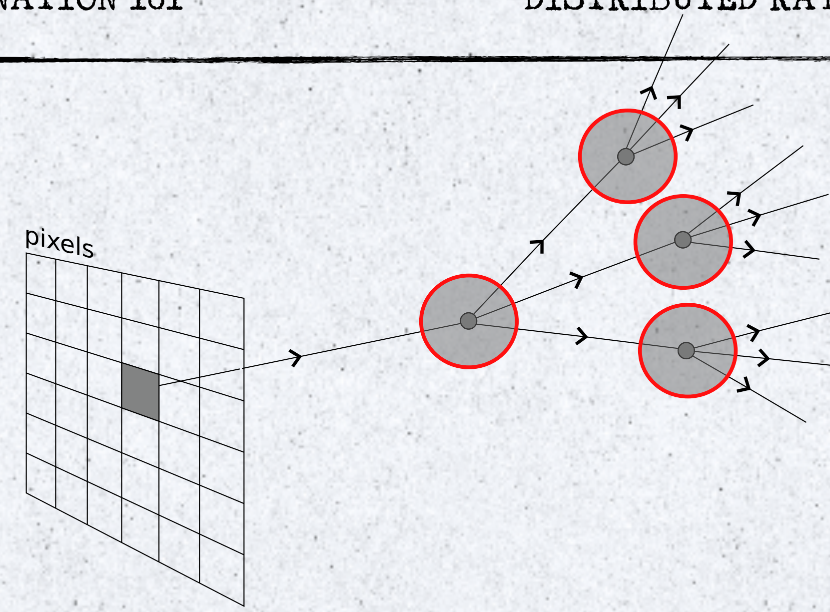
Big transition for pixar
done in multiple steps incrementally every movie:
. MU, move to physically based shading, normalized correct bsdfs, area-lights, raytraced shadows/reflections, 1 bounce indirect diffuse
. Finding Dory, full path tracing, multi bounce indirect, russian roulette, stochastic light picking , lobe sampling

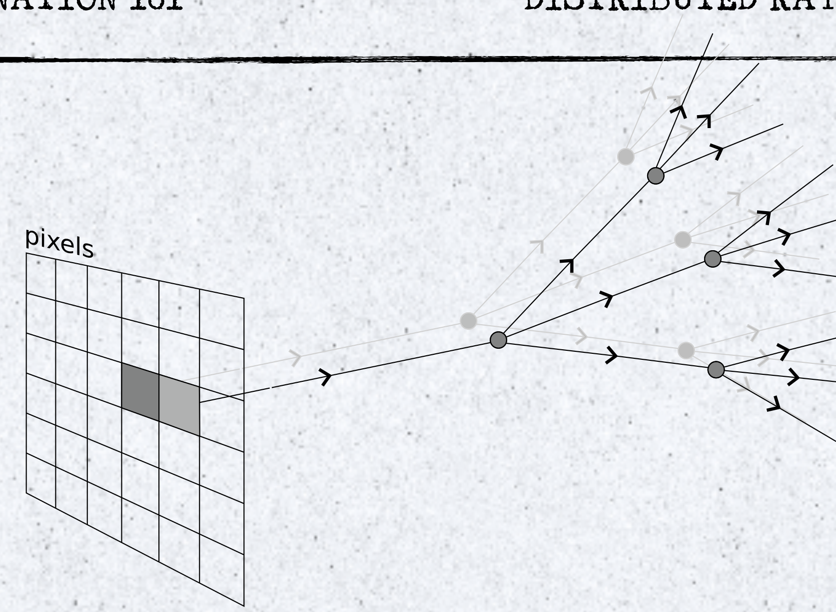
Why on Finding Dory?

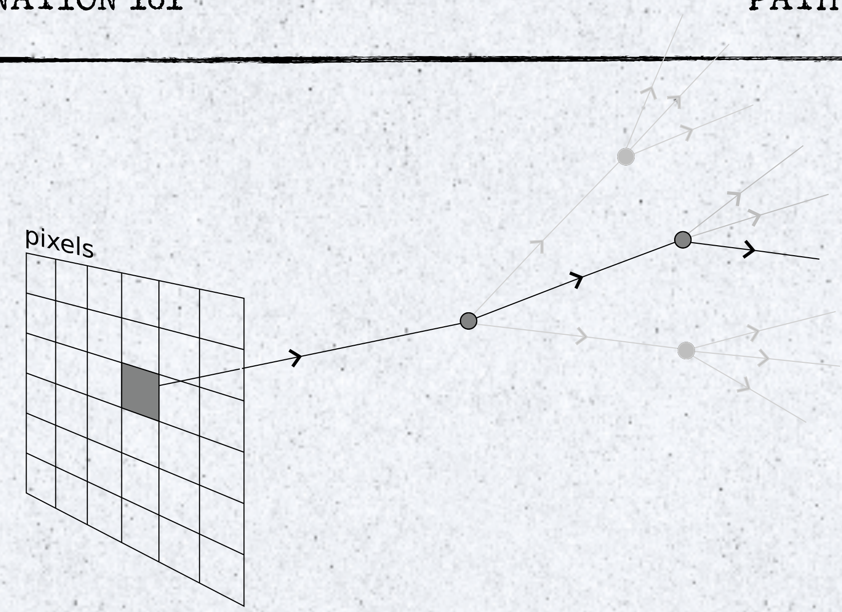
Mainly because of water or underwater rendering…
Full path tracing, i.e. via RIS and better light transport simulations, was required.
Here images from the short film Piper.

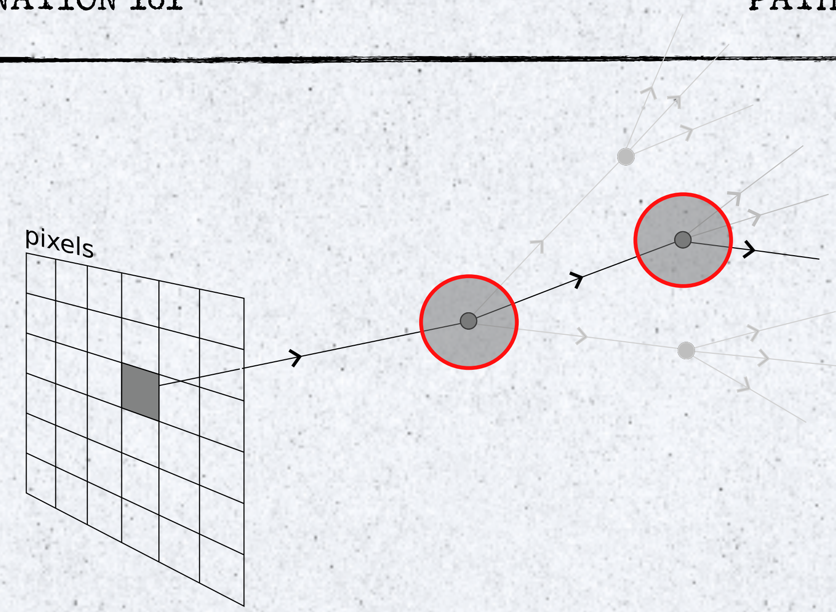
DISTRIBUTED RAY-TRACING

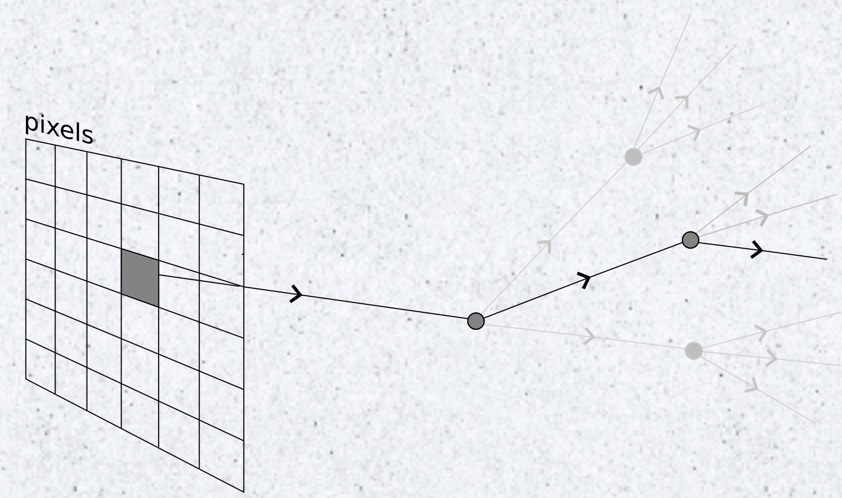



pixels

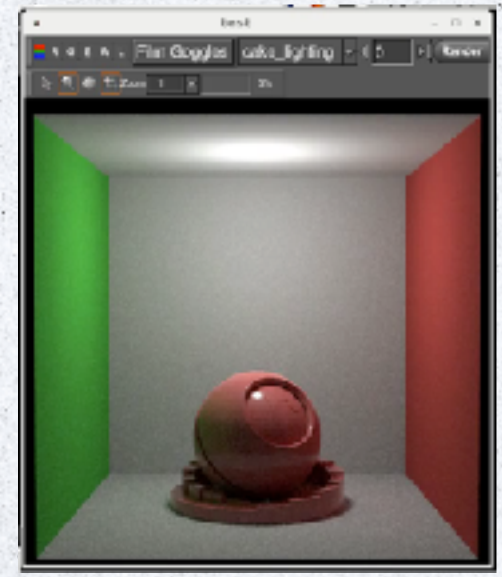

pixels


pixels


pixels


pixels

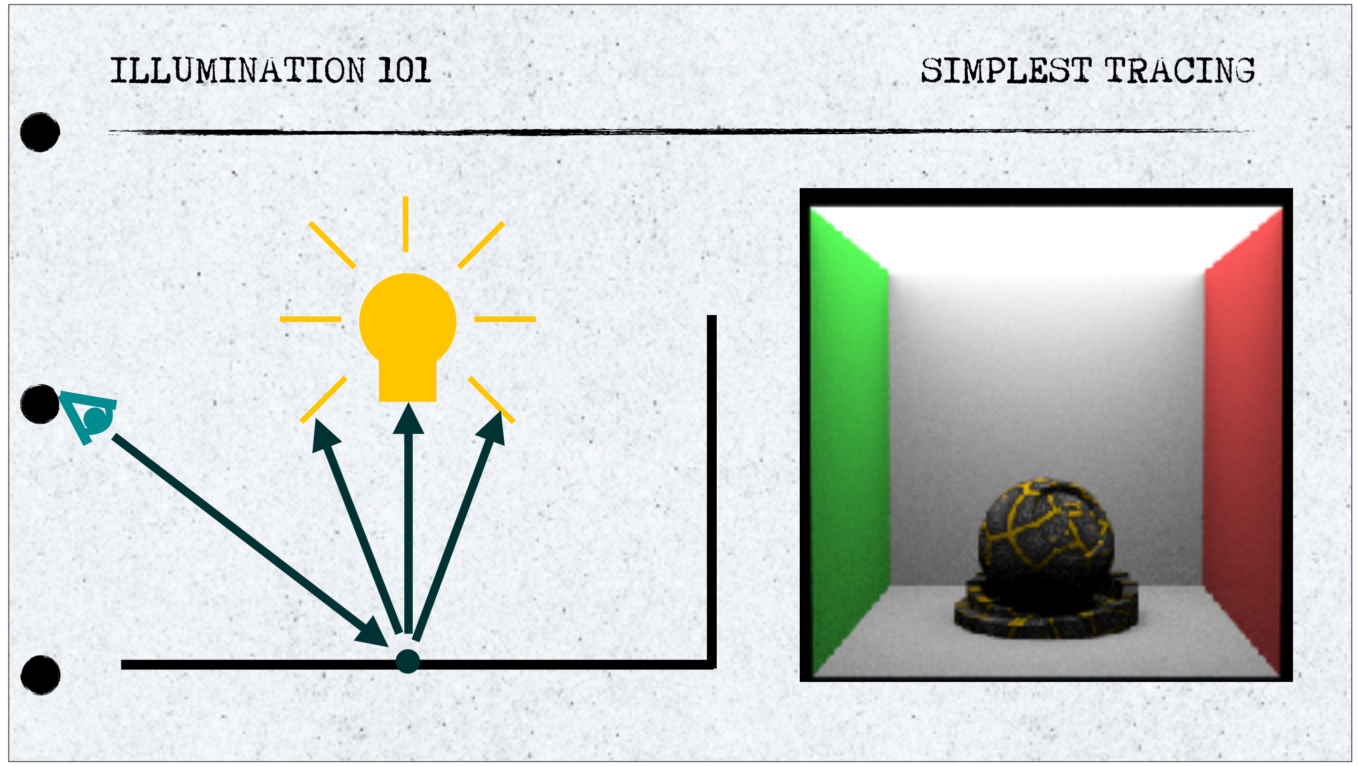
ILLUMINATION 101 PROGRESSIVE RENDERING



Moving to Path Tracing gave us an obvious mean to progressive rendering, where we display all pixels at once during each sampling iteration.

- Forward Ray-Tracing
- + Next Event Estimation

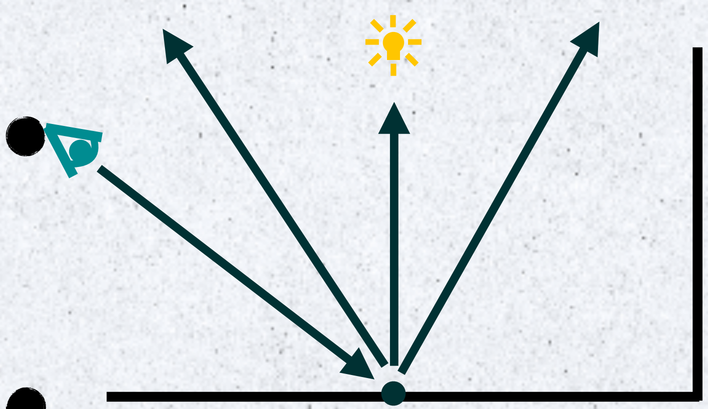
In general, what is used in 99% of production rendering is forward raytracing along with next event estimation.

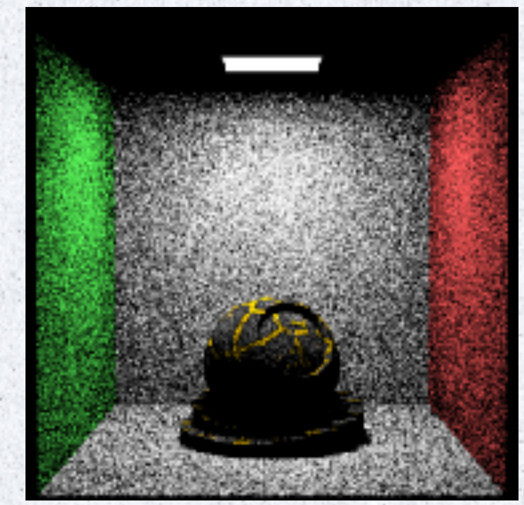


Intuitively it's better to start from the camera even if physically it's the opposite. And this works as long as there is a good probability of hitting the light sources

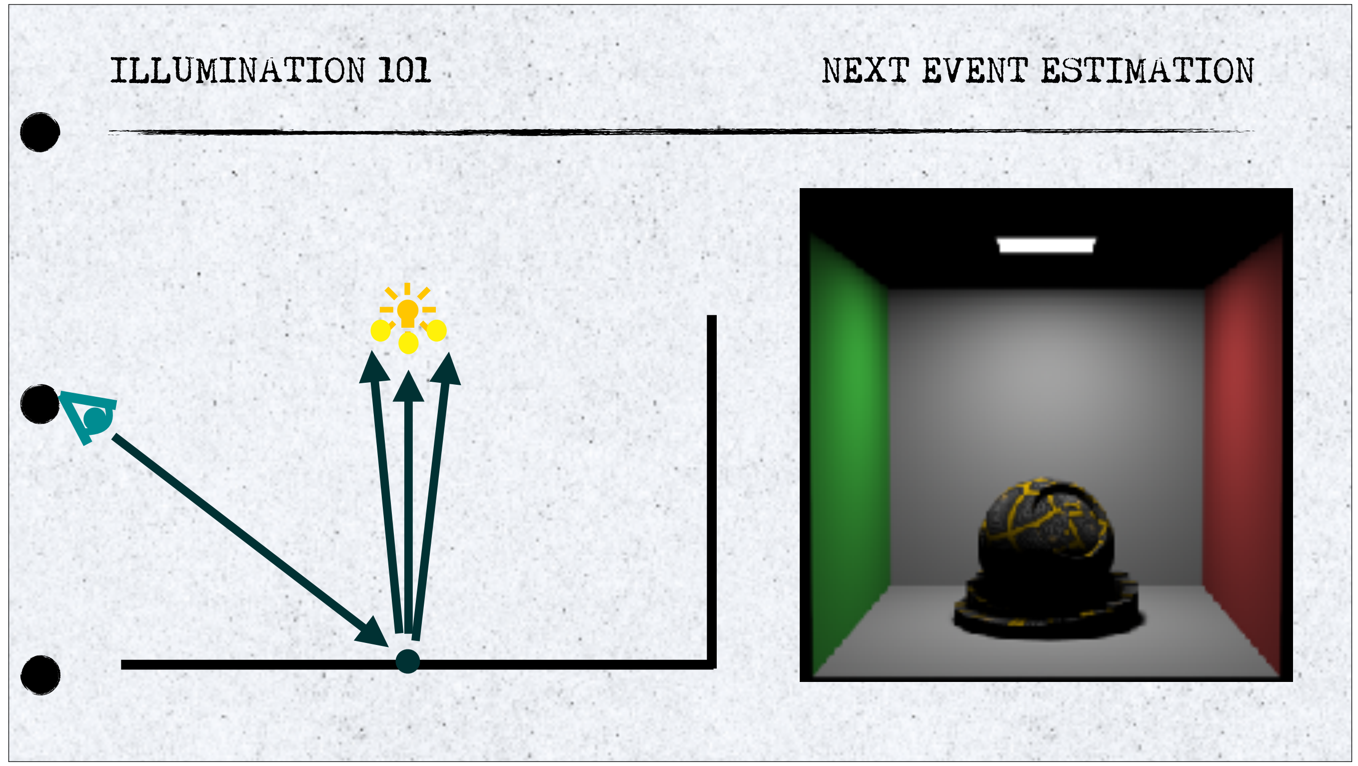
ILLUMINATION 101

SIMPLEST TRACING

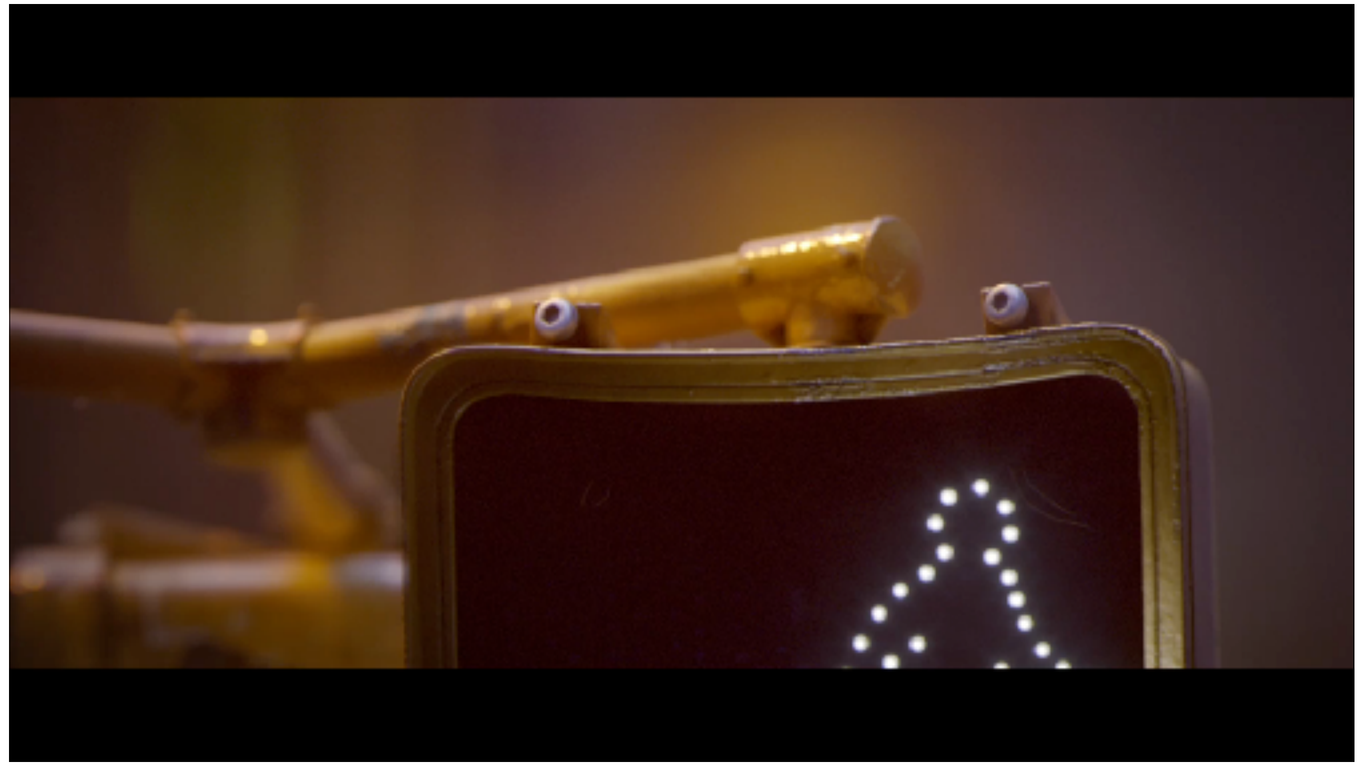




if the light is small we are wasting most of the samples



We usually add next event estimation to trace directly to the light sources that can be small (low probability of hitting it blindly)



THE NEW NORM: LOTS OF LIGHTS

car lights, neon signs and city luminaires in The Blue Umbrella

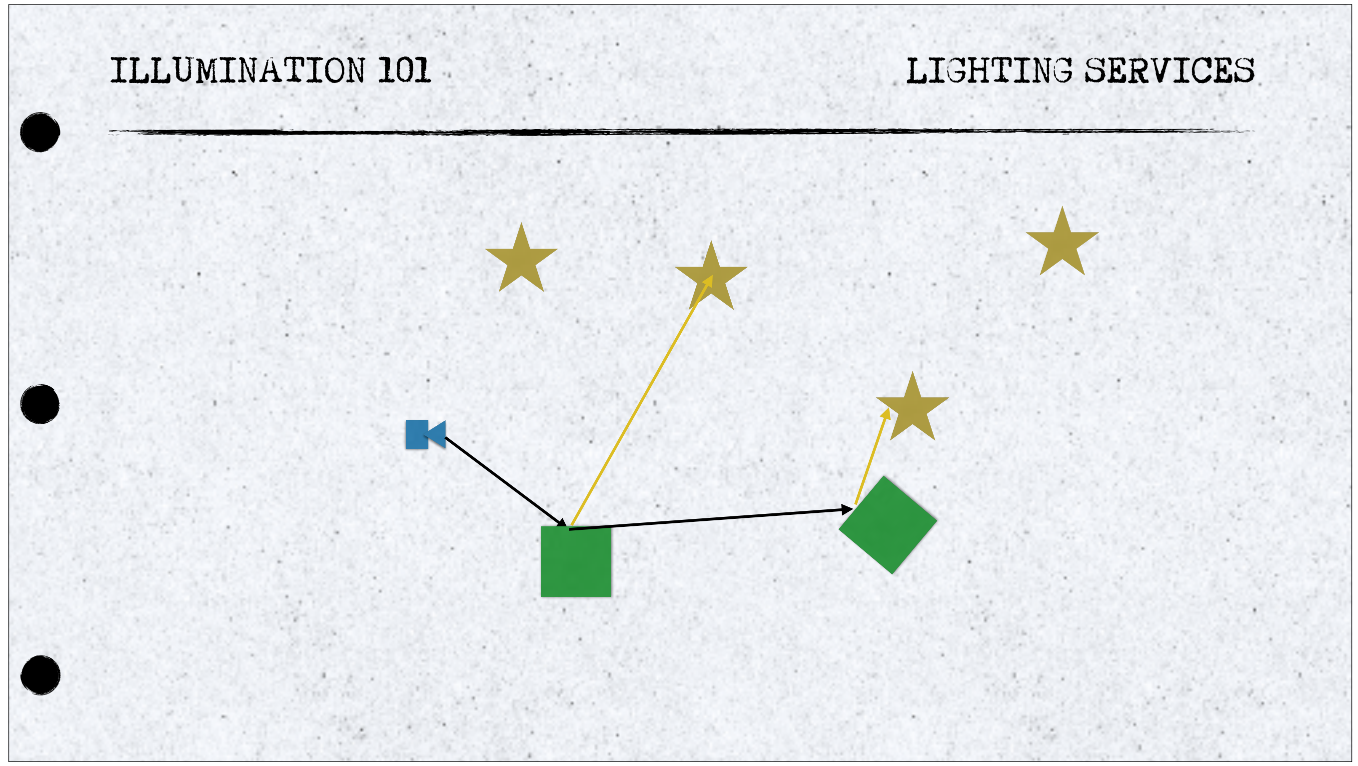
car lights, neon signs and city luminaires in The Blue Umbrella

THE NEW NORM: LOTS OF LIGHTS

~6000 fireflies on-camera in The Good Dinosaur

- we cannot do next event estimation to all these lights (too expensive)
- so we introduce an abstraction, the Lighting Services, whose purpose is to pick one of these lights (hopefully a good one) for a given next event connection

Too many lights. We cannot loop over them all the time. So we abstract out, in the form of a Lighting Services, and return one of these lights for a given next event connection.



Maybe we pick the closest one.

At the next vertex, we may pick and connect to another light.

$$L_o(x,\omega) = \int_\Omega f_r(x,\omega',\omega) L_i(x,\omega')(\omega' \cdot n)\, d\omega'$$

Integrator Bxdf LightingServices

So now our integral is not a sum over each light, but a sum over the lighting services.

Between

- progressive rendering
- and the many lights / lighting services abstraction

With all this…

Between

- progressive rendering
- and the many lights / lighting services abstraction

We cannot really compute, on the fly, a fully correct average visibility!

… we cannot use our average visibility for control variates (we get the average over the whole lighting services at the current iteration count).

Conclusion (with Path-Tracing):

- go back to the simple alpha = 1 approach (and accept "oscillations" around potential negative results along the iteration count);

So… either we go back to alpha = 1

Conclusion (with Path-Tracing):

- go back to the simple alpha = 1 approach (and accept "oscillations" around potential negative results along the iteration count);
- or accept some bias/approximation.

Or we accept some bias and/or approximation. In any case, our 1/(N-1) rescaling seems quite irrelevant nowadays.

Other approaches:

- Make use of a rather complex 4D visibility map

[Exploiting Visibility Correlation in Direct Illumination - EGSR 2008]

Visibility map, fairly approximative (since based on proximities), almost like our Spherical Harmonics from before.
Also remember that we want this computation to be fast, certainly faster than pure MC, otherwise there would be no point. This is not a given here.

Other approaches:

- Make use of a rather complex 4D visibility map

  [Exploiting Visibility Correlation in Direct Illumination - EGSR 2008]

- Or turn to a new "gaming" ratio estimator

  [Combining Analytic Direct Illumination and Stochastic Shadows - I3D 2018]

This gaming approach is similar to a ratio estimator. It is very biased. Something the game guys are willing to accept

To summarize:

- the ideal estimator is with average visibility (with the rescaling trick)

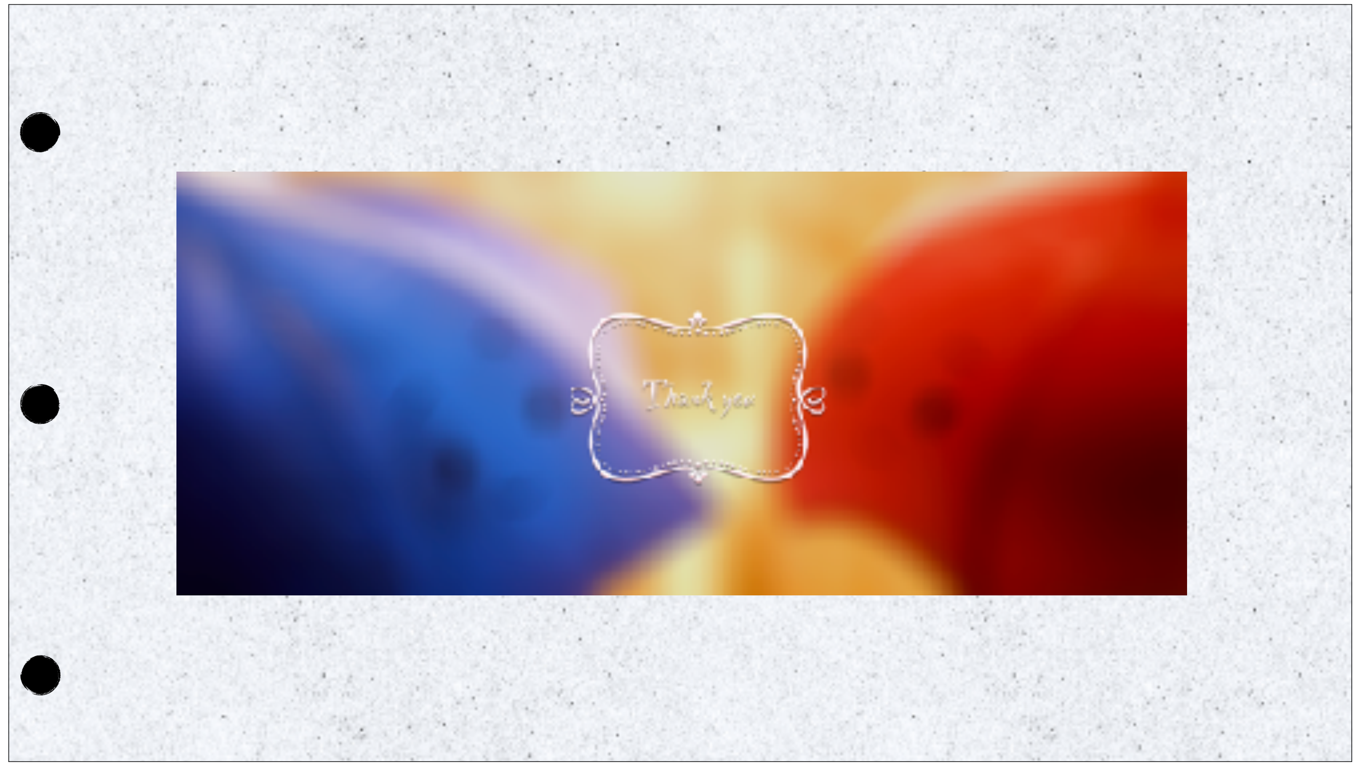
In conclusion, I have presented our ideal estimator with control variates (making use of the average visibility).

To summarize:

- the ideal estimator is with average visibility (with the rescaling trick)
- but in practice we go back to the simplest alpha = 1 approach.

But since the introduction of Path Tracing in production rendering (with its many advantages), we go back to the alpha = 1 approach.

Thank you



Questions?