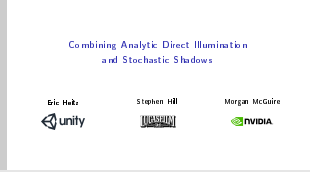
Combining Analytic Direct Illumination and Stochastic Shadows

Eric Heitz

Stephen Hill

Morgan McGuire

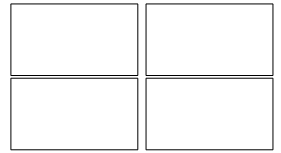


This presentation contains many animated slides that do not display well with all PDF viewers.

We recommend using **Adobe Acrobat Reader**.

Introduction



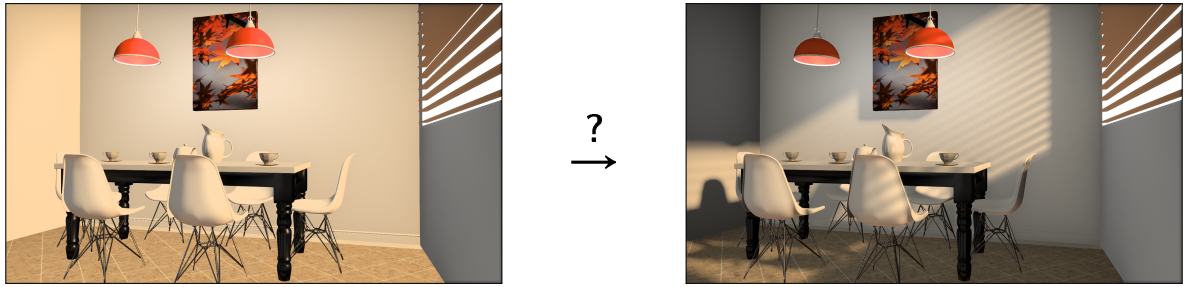



In this paper, we are interested in the problem of shadowing scenes rendered in real time with complex lighting models.

⚠This slide is animated (works with Acrobat Reader).

Introduction

- Problem: real-time soft shadows are difficult!

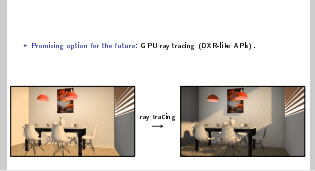


Indeed, shadowing is still one of the most challenging problems for real-time rendering.

Introduction

- Promising option for the future: GPU ray tracing (DXR-like APIs).

ray tracing
⟶



Fortunately, NVIDIA has announced a new API for GPU ray tracing:
`https://developer.nvidia.com/rtx`

It is thus easy to imagine that the future of real-time rendering is hybrid rendering. Hybrid in the sense that we will keep the raster-based pipeline for what it is already good at and add new complex effects on top of it using ray tracing.

Introduction

- *"Ray tracing makes shadowing obvious."*

ray tracing
⟶

This is a very good news for people working on shadows. Indeed, raster-based shadows are extremely hard to get right but ray tracing makes them simple.

Introduction

- Wait a minute... what rays should I trace?

ray tracing
→
...
what?

Well... does it really?

If you sit at your computer and try to implement a shadowing algorithm using ray tracing, the first thing you will have to figure out is which rays you actually want to trace.

Introduction

- *"Well, I am like... shooting rays towards the light source?*

ray tracing
→
...
what?

Combining Analytic Direct Illumination and Stochastic Shadows

The most obvious answer that might come to your mind is to trace rays towards the light source. This is what most people are going for when they talk about ray tracing shadows.

Introduction

- How do I sample the rays? Over the light? Over its solid angle?
- Do I have to weight the rays? How?
- How to make sure that the result is correct?

ray tracing
→
...
what?

However, tracing rays towards the light source is still too ambiguous. There are many questions to answer: how do you choose these rays? Do they have to be weighted somehow? For instance, what if a sampled ray points inside the surface of the shading point? Does it count as a shadow or not? What if the BRDF is a mirror? Do we have to make a special case for mirrors? What if the BRDF is not a mirror but it is a near-specular BRDF?

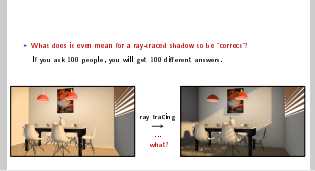
Furthermore, an important question for validation and debugging is how to verify that the result is correct.

Introduction

- What does it even mean for a ray-traced shadow to be "correct"?

  If you ask 100 people, you will get 100 different answers.

ray tracing
⟶
...
what?



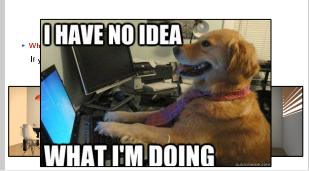
This leads another more important question: what does it mean for a ray-traced shadow to be correct in the first place?

If you ask around in the real-time community, you will never get a consistent and accurate answer.

Introduction

- Wh

If y

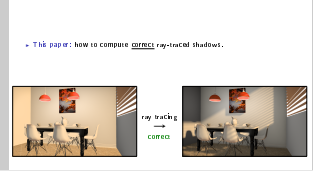


This is problematic. If we don't know the correct result we should be targeting, it means that we don't understand the approximations we're making in our real-time renderers. If we don't understand what we do, we might get bugs, visual artifacts, or just poor-looking results, and be clueless about how to solve these problems.

Introduction

- This paper: how to compute correct ray-traced shadows.

ray tracing
⟶
correct



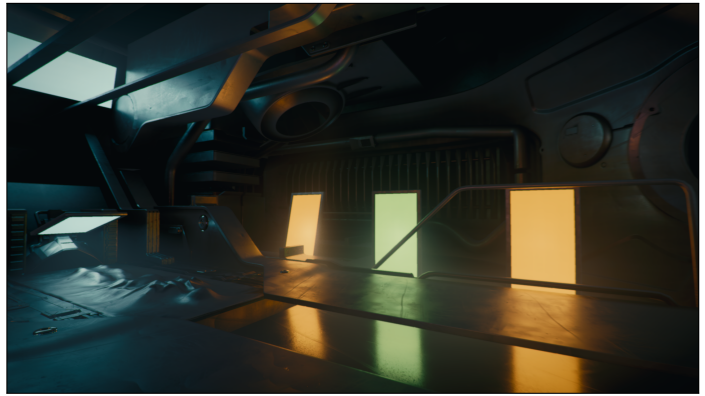
This is what this paper is about. It is about making the point in the real-time rendering community that "*shooting rays towards the light source*" is a flawed approach to ray-traced shadows and providing the definition of correct ray-traced shadows, along with a practical algorithm to compute them.

With this, we hope that the real-time rendering community will be better prepared for the ray-tracing abilities of future GPUs.

Backstory

Before getting to ray-traced shadows, I would like to provide you with some backstory and our personal motivation for investigating this problem.

Backstory

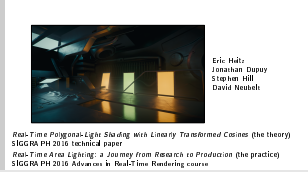


Eric Heitz
Jonathan Dupuy
Stephen Hill
David Neubelt

Real-Time Polygonal-Light Shading with Linearly Transformed Cosines (the theory)
SIGGRAPH 2016 technical paper

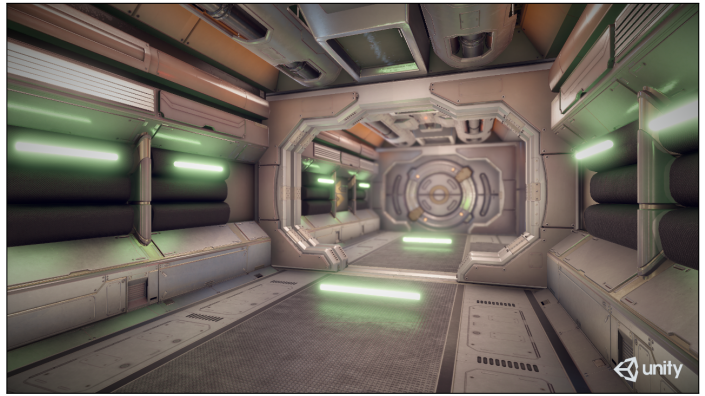
Real-Time Area Lighting: a Journey from Research to Production (the practice)
SIGGRAPH 2016 Advances in Real-Time Rendering course

Combining Analytic Direct Illumination and Stochastic Shadows



The collaboration between Steve and I started a few years ago when we worked together on area lights. We came up with an efficient solution for shading with polygonal lights in real time. This solution was based on the introduction of a new spherical distribution: Linearly Transformed Cosines (LTC) that have good properties for spherical integration.

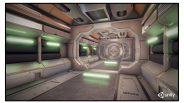
Backstory



Eric Heitz
Stephen Hill

Linear-Light Shading with Linearly Transformed Cosines
book chapter in GPU Zen 2017

Combining Analytic Direct Illumination and Stochastic Shadows



We later extended this solution to support line lights and...

Backstory

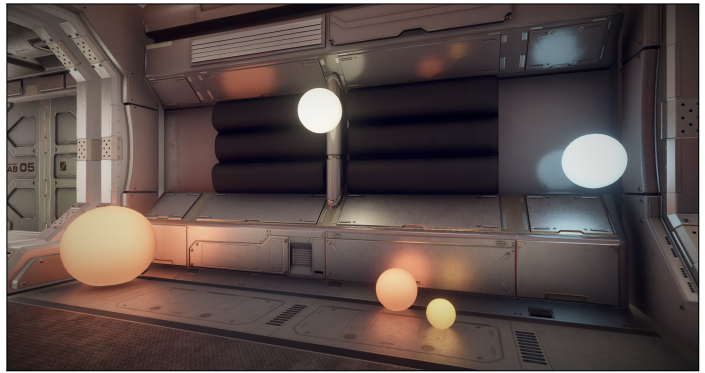
Eric Heitz
Stephen Hill

Real-Time Line- and Disk-Light Shading with Linearly Transformed Cosines
SIGGRAPH 2017 Physically Based Shading course

Combining Analytic Direct Illumination and Stochastic Shadows

...disk lights as well.

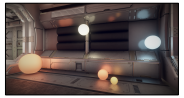
Backstory



Jonathan Dupuy
Eric Heitz
Laurent Belcour

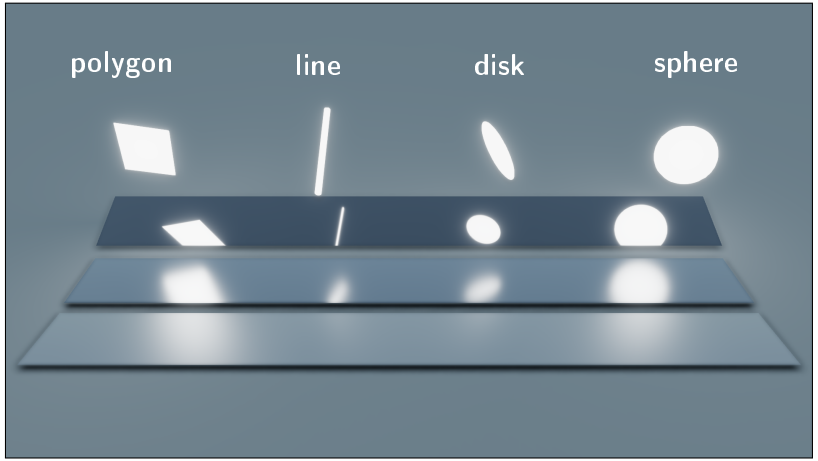
A Spherical Cap Preserving Parameterization for Spherical Distributions
SIGGRAPH 2017 technical paper

Combining Analytic Direct Illumination and Stochastic Shadows

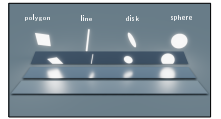


In parallel, at Unity, we worked out another spherical distribution that has good properties for sphere lights.

Backstory



We arrived at a pretty complete real-time area-lighting framework. But...



We arrived at a pretty complete real-time area-lighting framework. But...

With all these research projects, we arrived at a pretty complete area-lighting framework that supports a good set of primitives. But...

Backstory

Morgan @CasualEffects · 21 avr. 2016

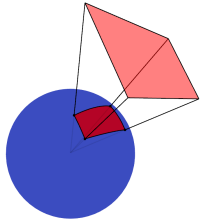
@eric_heitz @unity3d @self_shadow @daveneubelt Excellent! How do I shadow them?

Combining Analytic Direct Illumination and Stochastic Shadows

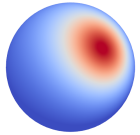
... there was one main limitation of this framework: these area lights cannot be shadowed.

This is where the third author of this paper comes into play. :-)

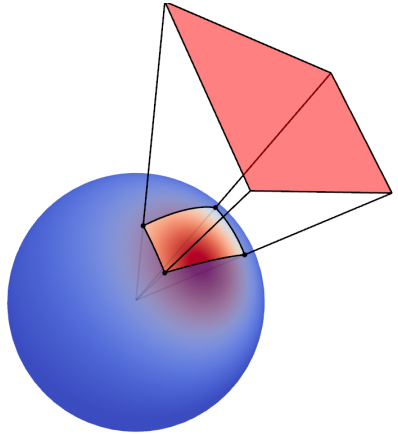
Backstory



Light



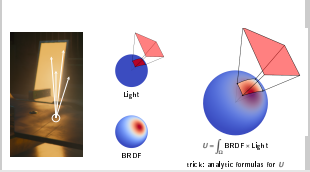
BRDF



$$U = \int_{\Omega} \text{BRDF} \times \text{Light}$$

trick: analytic formulas for U

Combining Analytic Direct Illumination and Stochastic Shadows



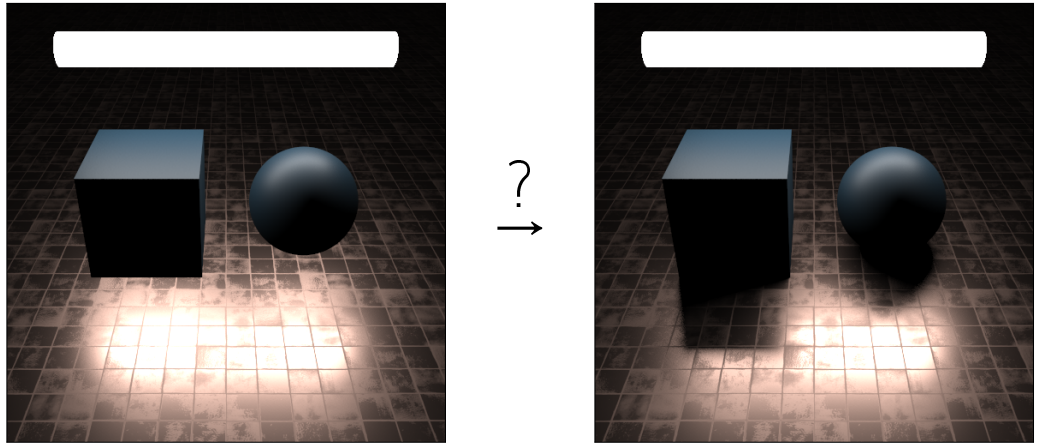
The reason why our area lights cannot be shadowed is related to how they work.

Shading a pixel with an area light means considering two spherical functions – the BRDF and the spherical domain covered by the area light – and computing their integral. The trick with our area-lighting framework is that we were able to find analytic solutions for this integral, thanks to the new spherical distributions we designed.

Backstory

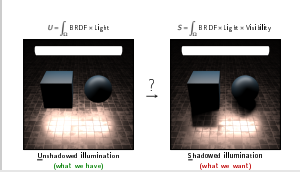
$$U = \int_\Omega \text{BRDF} \times \text{Light}$$

$$S = \int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility}$$



Unshadowed illumination
(what we have)

Shadowed illumination
(what we want)

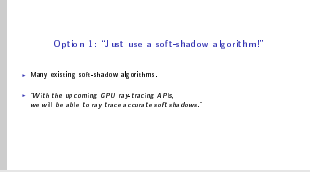


However, we are only able to compute U that stands for the unshadowed direct illumination. What we really want instead is S that stands for the shadowed direct illumination. The difference between U and S is the presence of the Visibility function inside the integral that accounts for occluders between the light and the shading point, i.e. the shadowing.

This is the point where we were when we started this project. We had a good solution for U but what we want is S. What should we do?

Option 1: “Just use a soft-shadow algorithm!”

- Many existing soft-shadow algorithms.
- *“With the upcoming GPU ray-tracing APIs, we will be able to ray trace accurate soft shadows.”*



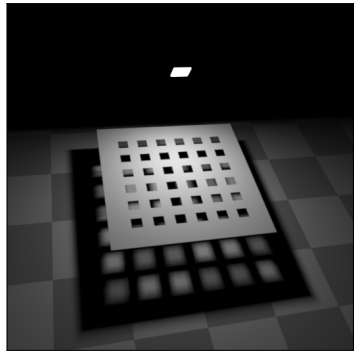
If you show this problem to a rendering developer, he might think:

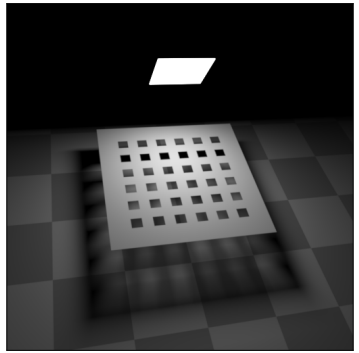
“*Well, if the light were a point light I would use a shadow map. However, you have an area light and I know that area lights produce soft shadows. So... why don’t you just use a soft shadow algorithm?*”

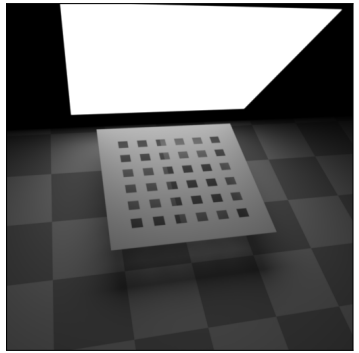
This leads us to the first option to consider for solving our problem.

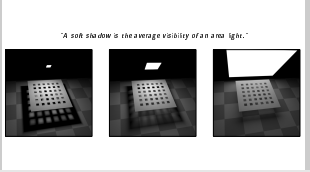
Option 1: "Just use a soft-shadow algorithm!"

"A soft shadow is the average visibility of an area light."





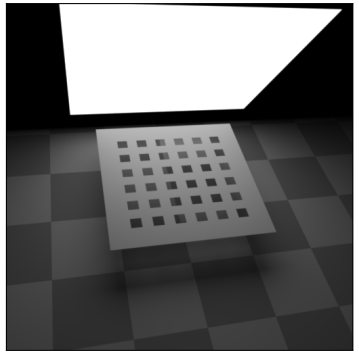


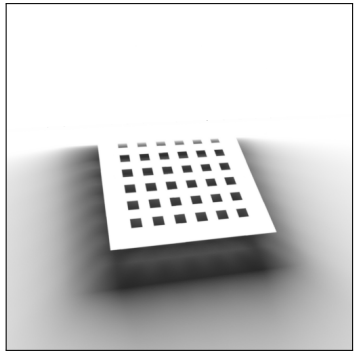


Typically, a soft shadow is defined as the average visibility of an area light. If the light is a single point, the visibility is binary, which creates a hard shadow. The wider the light, the more this binary visibility is averaged, which creates this blurring effect that makes it soft.

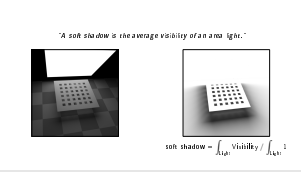
Option 1: "Just use a soft-shadow algorithm!"

"A soft shadow is the average visibility of an area light."





$$\textbf{soft shadow} = \int_{\text{Light}} \text{Visibility} \;/\; \int_{\text{Light}} 1$$

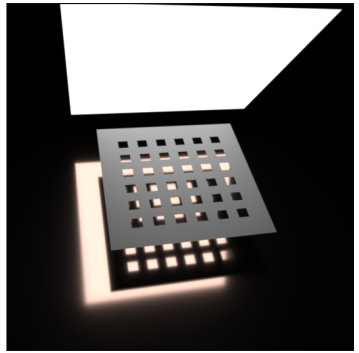
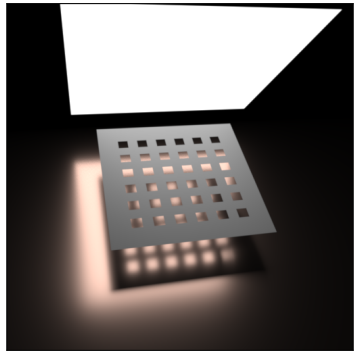
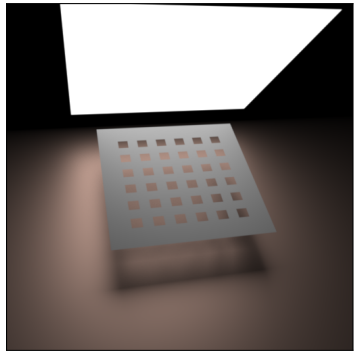


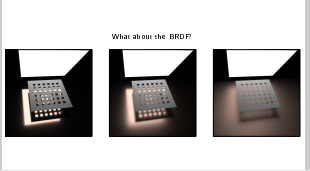
With this definition, a soft shadow can be defined by this equation: this is literally shooting rays towards the light and averaging their contribution. This is what most people have in mind when they think of using ray tracing to compute shadows.

An interesting property of this definition is that it is a purely geometric quantity: if you know the geometry of the scene (the meshes), you know the soft shadow.

Option 1: “Just use a soft-shadow algorithm!”

What about the BRDF?

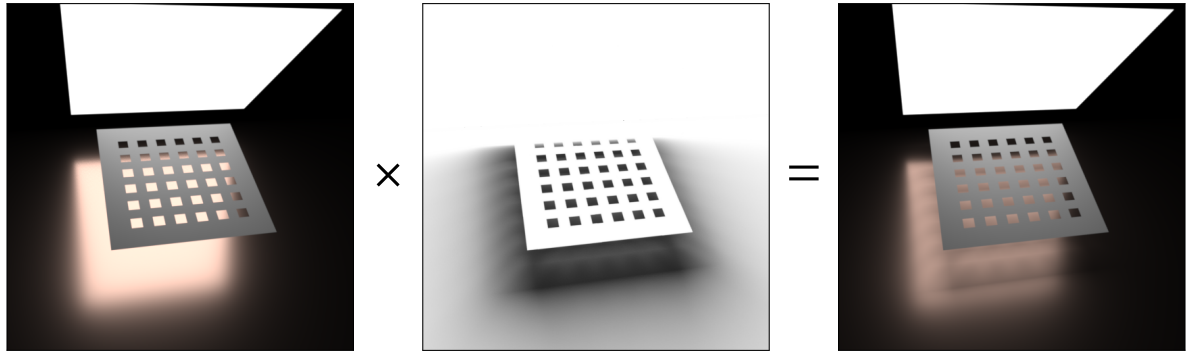


But what about other quantities, such as the materials? These scenes are offline reference results that have been computed with the same geometry, and only the BRDF of the receiver plane has been changed. Still, we can see that the softness of the shadow changes with the BRDF, which shows that it doesn't just depend on the geometry.

This is pretty simple to understand if we consider a mirror surface. You will never see a soft shadow on a mirror. The reason is that a mirror reflects the light in a single direction, which is then either occluded or not. This binary result makes the shadow hard. In general, a very smooth surface (near-specular) will receive almost-hard shadows and the glossier the surface becomes, the softer the soft shadows can be.

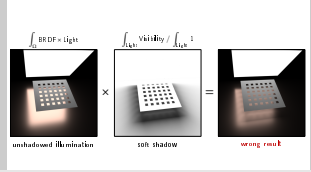
Option 1: "Just use a soft-shadow algorithm!"

$\int_{\Omega} \text{BRDF} \times \text{Light}$ $\int_{\text{Light}} \text{Visibility} \,/\, \int_{\text{Light}} 1$



unshadowed illumination | soft shadow | wrong result

Combining Analytic Direct Illumination and Stochastic Shadows

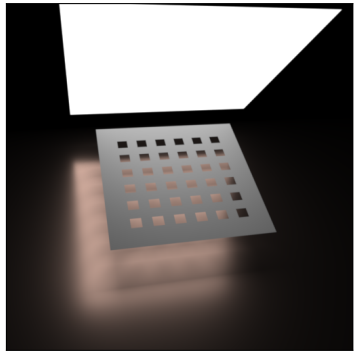


This is a real-time attempt at reproducing the offline result in the middle of the previous slide. In this attempt, we use our real-time area-lighting framework to compute the unshadowed direct illumination and a real-time soft-shadow algorithm (for instance, GPU ray tracing) to modulate the result.

The result looks obviously wrong.

Option 1: "Just use a soft-shadow algorithm!"

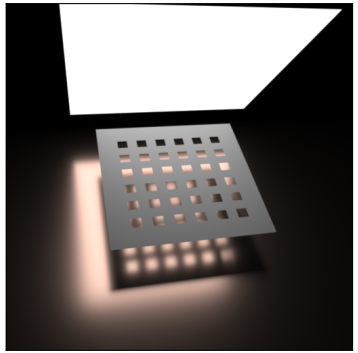
$$\int_{\Omega} \text{BRDF} \times \text{Light} \times \int_{\text{Light}} \text{Visibility} \Big/ \int_{\text{Light}} 1$$



wrong result

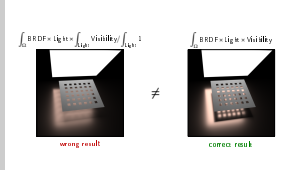
$\neq$

$$\int_{\Omega} \text{BRDF} \times \text{Light} \times \text{Visibility}$$



correct result

Combining Analytic Direct Illumination and Stochastic Shadows



The reason is that we are not computing the right equation.

This shows that real-time soft shadows as they are usually computed, i.e. *"shooting rays towards the light source"*, is not the right approach. In some cases it works OK, but in other cases it can produce poor-looking results that are obviously wrong.

With this in mind, it is kind of a shame to think of using GPU ray tracing to compute soft shadows in this way. All of this computational power would be wasted on something that is not even guaranteed to look good!

Option 2: "Want the correct result? Go full-stochastic!"

Computing the right equation is important to be sure that the result will look good. This leads us to the next option to consider for our problem. If we want to be certain of getting the right result, we can do as in offline rendering and use a Monte Carlo estimator of the direct illumination.

Option 2: "Want the correct result? Go full-stochastic!"

Morgan @CasualEffects · Aug 14

Replying to @CasualEffects @eric_heitz

The problem is: they don't work with preintegrated shading. Stochastic shadowing methods force stochastic shading if you want correctness.

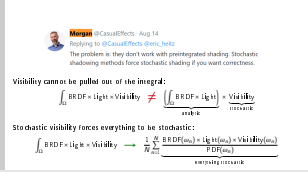
Visibility cannot be pulled out of the integral:

$$\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility} \neq \underbrace{\left(\int_\Omega \text{BRDF} \times \text{Light}\right)}_{\text{analytic}} \times \underbrace{\text{Visibility}}_{\text{stochastic}}$$

Stochastic visibility forces everything to be stochastic:

$$\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility} \longrightarrow \underbrace{\frac{1}{N}\sum_{n=1}^{N} \frac{\text{BRDF}(\omega_n) \times \text{Light}(\omega_n) \times \text{Visibility}(\omega_n)}{\text{PDF}(\omega_n)}}_{\text{everything stochastic}}$$

Combining Analytic Direct Illumination and Stochastic Shadows



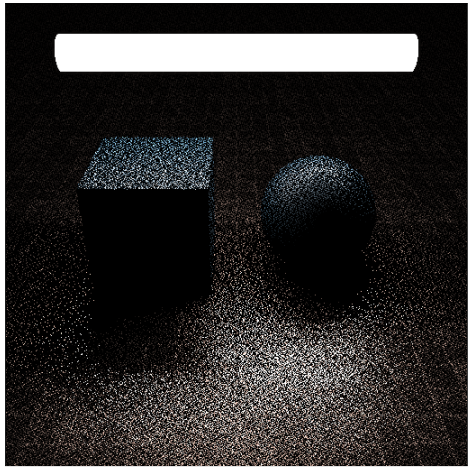
Morgan pointed a fundamental limitation of our area-lighting framework. What he said means that if we want to compute the right result, we have to make sure that the visibility term remains inside the integral. We have seen with soft shadows that computing an external visibility factor does not work.

So, if visibility has to remain inside the integral and we want to use a stochastic estimate of visibility (like ray tracing) then it will force the other terms of the integral (the BRDF and the Light) to be stochastic too and we get a full-stochastic estimator. This is what offline rendering people compute to get the right result.

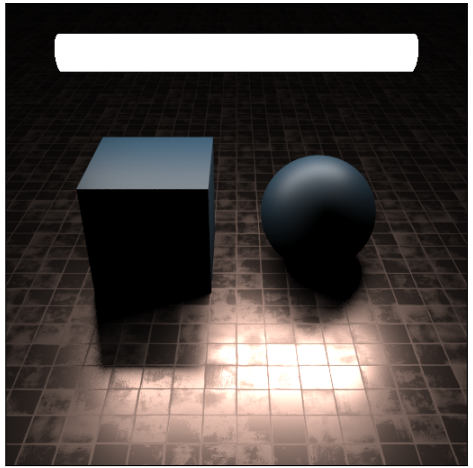
In summary, our analytic solution for the unshadowed illumination is useless, because if we want the right result we have no choice but to go full-stochastic and then we can't use it.

Option 2: "Want the correct result? Go full-stochastic!"

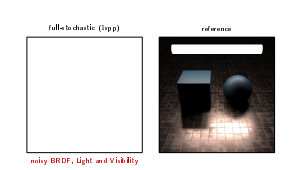
full-stochastic (1spp)



reference



noisy BRDF, Light and Visibility

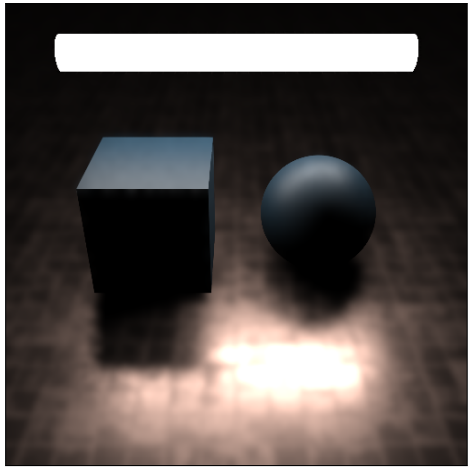


The problem with a full-stochastic estimator is that... it makes everything stochastic. Noise is present everywhere in the render because of the stochastic evaluation of the terms of the integral. This noise can be reduced by averaging a large number of evaluations but this is too costly for real-time rendering.

⚠This slide is animated (works with Acrobat Reader).

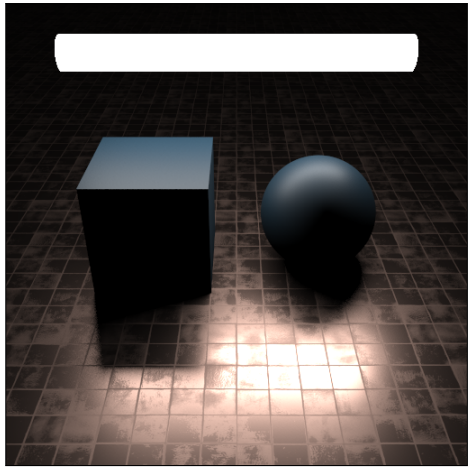
Option 2: "Want the correct result? Go full-stochastic!"

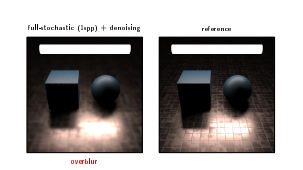
full-stochastic (1spp) + denoising



overblur

reference





An approach that has been investigated recently is to use a full-stochastic evaluation at very low sample counts and then apply a real-time denoiser. Several papers presented at High Performance Graphics 2017 were based on this idea, and used temporal accumulation to emulate a larger number of samples per pixel.

The problem with real-time denoising is that – even with temporal accumulation – it has to be extremely aggressive, and this results in overblurred details. The latest real-time denoisers are pretty good at denoising smoothly shaded surfaces and preserving their edges, but they cannot deal with per-pixel shading details such normal maps or roughness maps. If you look carefully at the results of these HPG 2017 papers, you will see that their scenes are flatly shaded, without normal maps.

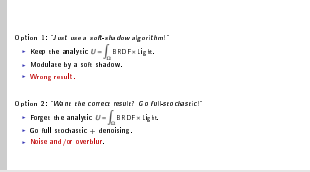
State-of-the-art options for real-time shadows

Option 1: *"Just use a soft-shadow algorithm!"*

- Keep the analytic $U = \int_\Omega \text{BRDF} \times \text{Light}$.
- Modulate by a soft shadow.
- Wrong result.

Option 2: *"Want the correct result? Go full-stochastic!"*

- Forget the analytic $U = \int_\Omega \text{BRDF} \times \text{Light}$.
- Go full stochastic + denoising.
- Noise and/or overblur.



This is a bit oversimplistic, but it's pretty much the state of the art for our problem. We want to shadow a real-time render and there are two main options to consider. We either go for a real-time soft-shadow algorithm and accept a wrong result, or we go for a full-stochastic evaluation (as in offline rendering) and live with noisy or overblurred results.

State-of-the-art options for real-time shadows

Morgan @CasualEffects · Aug 14

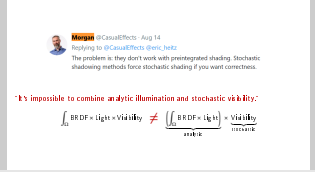
Replying to @CasualEffects @eric_heitz

The problem is: they don't work with preintegrated shading. Stochastic shadowing methods force stochastic shading if you want correctness.

"It's impossible to combine analytic illumination and stochastic visibility."

$$\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility} \neq \underbrace{\left(\int_\Omega \text{BRDF} \times \text{Light}\right)}_{\text{analytic}} \times \underbrace{\text{Visibility}}_{\text{stochastic}}$$

Combining Analytic Direct Illumination and Stochastic Shadows



The dilemma between the unsatisfying options 1 and 2 is due to what Morgan pointed out here. It is because we cannot pull visibility out of the integral that we are doomed to either do it approximately or make everything stochastic.

Insight

Morgan @CasualEffects · Aug 14

Replying to @CasualEffects @eric_heitz

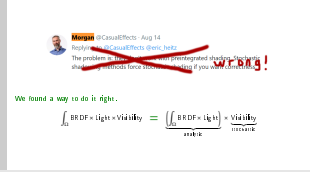
The problem is: they don't work with preintegrated shading. Stochastic shadowing methods force stochastic shading if you want correctness.

wrong!

We found a way to do it right.

$$\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility} = \underbrace{\left(\int_\Omega \text{BRDF} \times \text{Light}\right)}_{\text{analytic}} \times \underbrace{\text{Visibility}}_{\text{stochastic}}$$

Combining Analytic Direct Illumination and Stochastic Shadows



The main idea of the paper came when we realized that this was actually a misconception. We found a way to pull visibility out of the integral without breaking the equation! Thanks to this, we don't have to choose between option 1 and 2 and we obtain something much better instead.

So, the cornerstone of our method is to solve this equation, which is equivalent to asking...

What is a correct soft shadow?

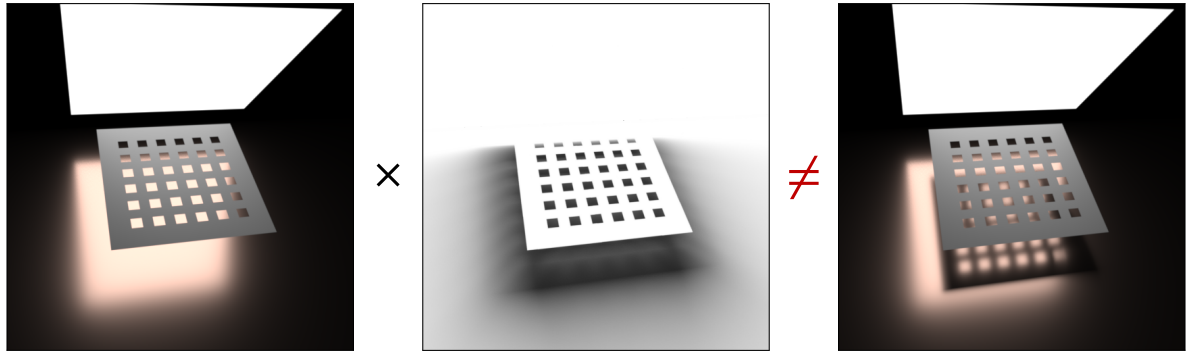
What is a correct soft shadow?

...what is a correct soft shadow?

This is the question that we were teasing in the introduction. What does it mean for a ray-traced shadow to be correct?

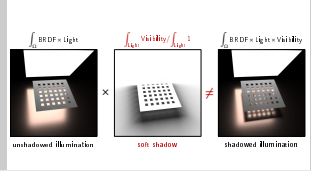
What is a correct soft shadow?

$\int_\Omega \text{BRDF} \times \text{Light}$ $\int_{\text{Light}} \text{Visibility} / \int_{\text{Light}} 1$ $\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility}$



unshadowed illumination soft shadow shadowed illumination

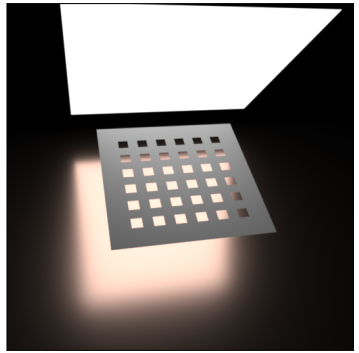
Combining Analytic Direct Illumination and Stochastic Shadows



We have seen that soft shadows do not compute the right result. So we might wonder...

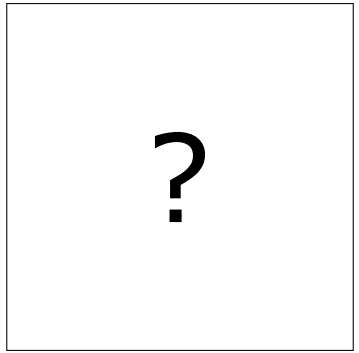
What is a correct soft shadow?

$\int_\Omega$ BRDF × Light



unshadowed illumination

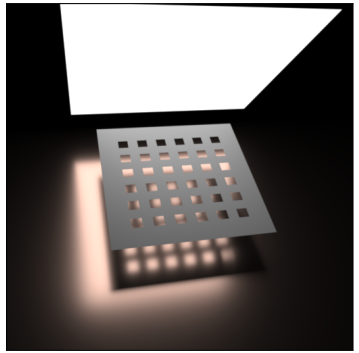
×



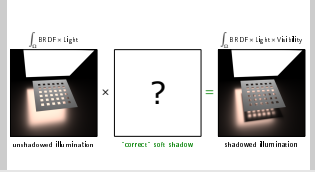
"correct" soft shadow

=

$\int_\Omega$ BRDF × Light × Visibility

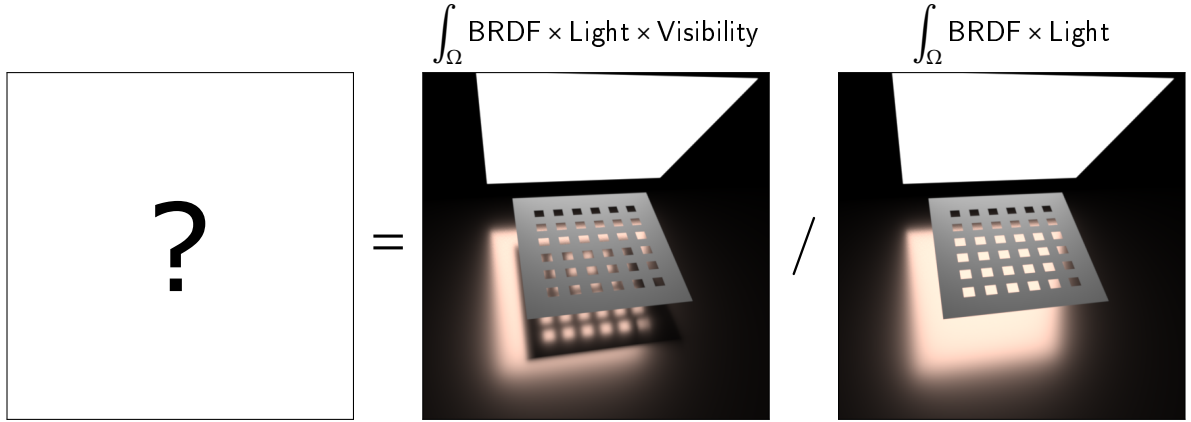


shadowed illumination



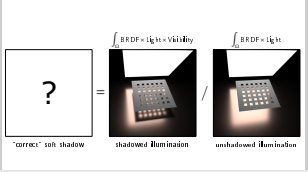
...what should we compute instead? What should "?" be here for the result to be correct?

What is a correct soft shadow?



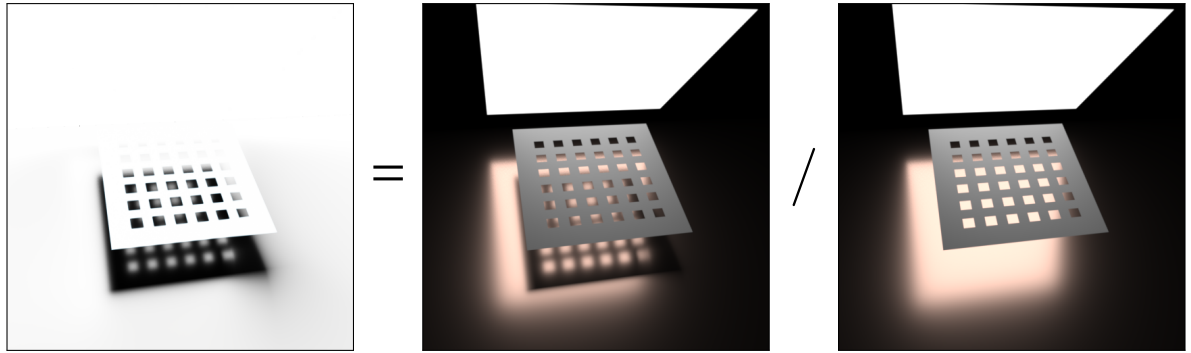
"correct" soft shadow | shadowed illumination | unshadowed illumination

Combining Analytic Direct Illumination and Stochastic Shadows



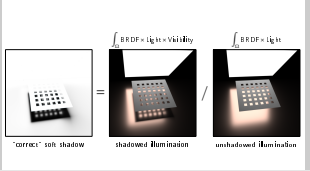
This is kind of obvious: if we put the first image on the other side of the equals sign, we find that "?" is the shadowed illumination divided by the unshadowed illumination.

What is a correct soft shadow?



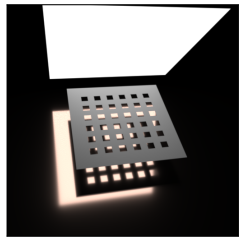
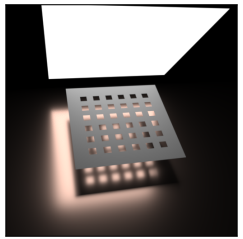
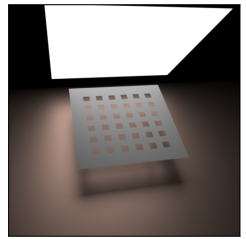
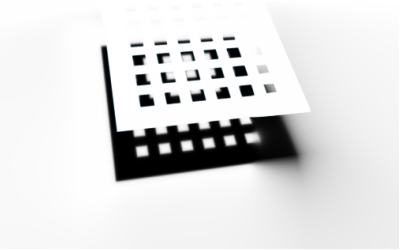
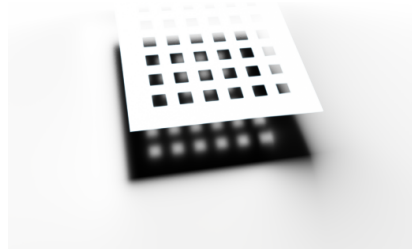
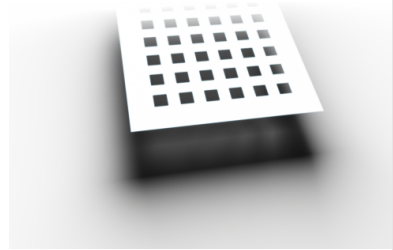
"correct" soft shadow | shadowed illumination | unshadowed illumination

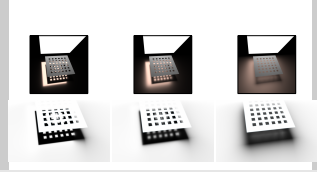
Combining Analytic Direct Illumination and Stochastic Shadows



If we compute this image, what we obtain is an image whose values are always between 0 and 1.

What is a correct soft shadow?



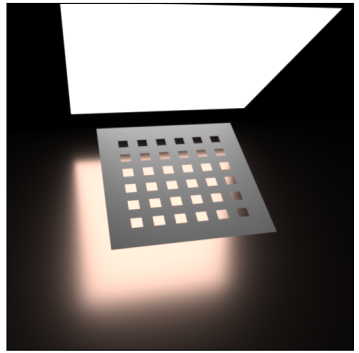
With this definition, the shadow image has the right softness when the BRDF changes, and it matches the shadow that we effectively see in the reference image.

In the paper, we call it the "illumination-weighted shadow" because visibility values are weighted by the BRDF-Light product. The more a direction matters for the BRDF and the Light, the greater its weight in the visibility average.

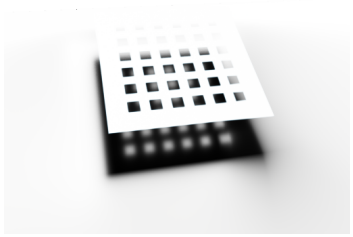
This correct shadow is nothing more than the correct per-pixel modulation factor for the result to be correct. It is the percentage of the energy that should theoretically be reflected from the light that happens to be lost due to occluders in the scene.

What is a correct soft shadow?

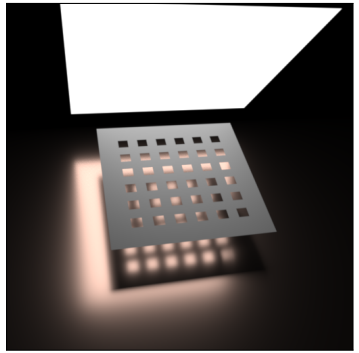
$$\int_\Omega \text{BRDF} \times \text{Light} \quad \times \quad \frac{\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility}}{\int_\Omega \text{BRDF} \times \text{Light}} \quad = \quad \int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility}$$



unshadowed illumination

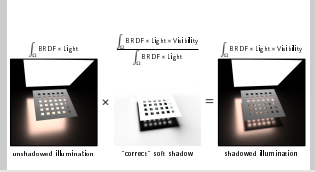


"correct" soft shadow



shadowed illumination

Combining Analytic Direct Illumination and Stochastic Shadows



With this definition, the equation is obviously correct: $a \cdot b/a = b$.

What is a correct soft shadow?

Morgan @CasualEffects · Aug 14

Replying to @CasualEffects @eric_heitz

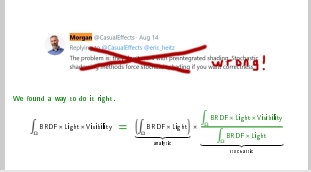
The problem is: they don't work with preintegrated shading. Stochastic shadowing methods force stochastic shading if you want correctness.

wrong!

We found a way to do it right.

$$\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility} = \underbrace{\left(\int_\Omega \text{BRDF} \times \text{Light}\right)}_{\text{analytic}} \times \underbrace{\frac{\int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility}}{\int_\Omega \text{BRDF} \times \text{Light}}}_{\text{stochastic}}$$

Combining Analytic Direct Illumination and Stochastic Shadows



This is how we solve this equation and break the dilemma between option 1 and option 2. We no longer have to choose between a fake external visibility factor or making everything stochastic.

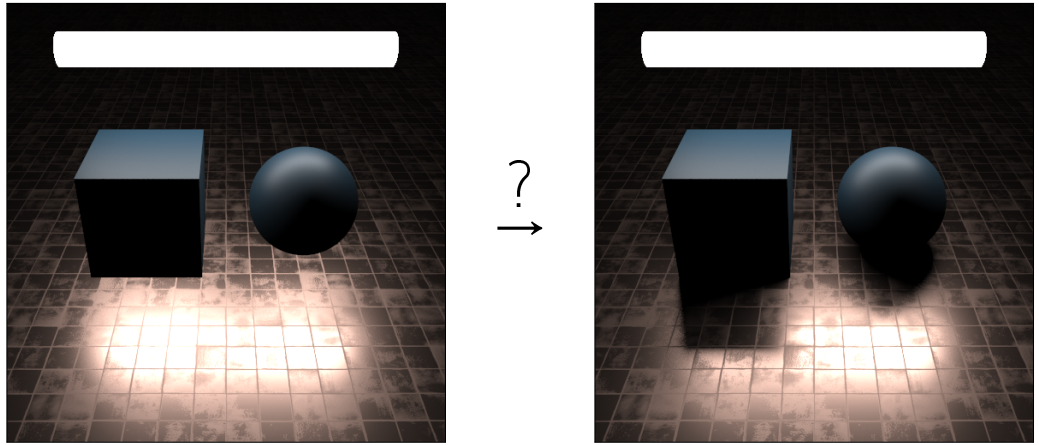
By defining the visibility in this way, we obtain a stochastic visibility term that can be computed outside of the integral as an external modulation factor of our shadowless analytic solution, and that does not break the equation. In summary, we get the right result without making everything stochastic!

Our algorithm

We will now see what our method does in practice.

Our algorithm

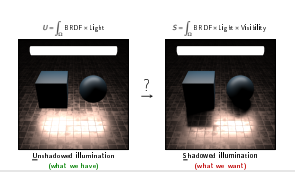
$$U = \int_\Omega \text{BRDF} \times \text{Light} \qquad S = \int_\Omega \text{BRDF} \times \text{Light} \times \text{Visibility}$$



Unshadowed illumination
(what we have)

Shadowed illumination
(what we want)

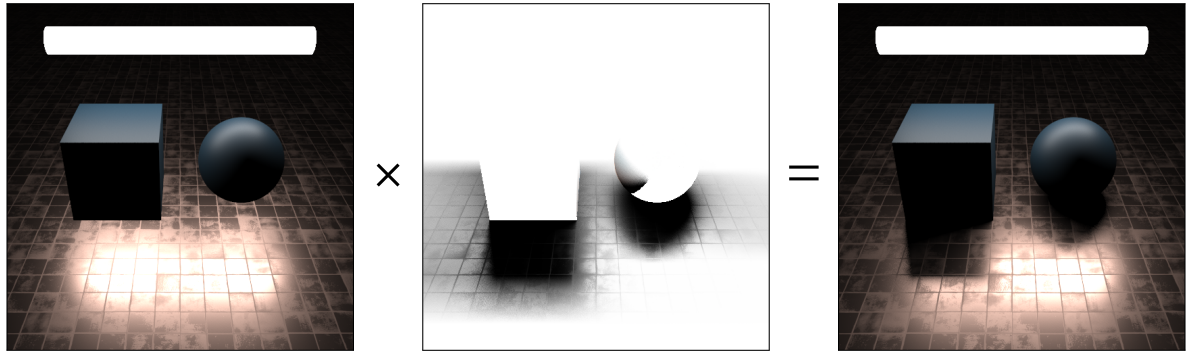
Combining Analytic Direct Illumination and Stochastic Shadows



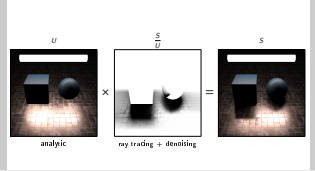
Remember, the problem we want to solve is to obtain S given that we have an analytic solution for U.

Our algorithm

U $\quad$ $\frac{S}{U}$ $\quad$ S



analytic $\quad$ ray tracing + denoising

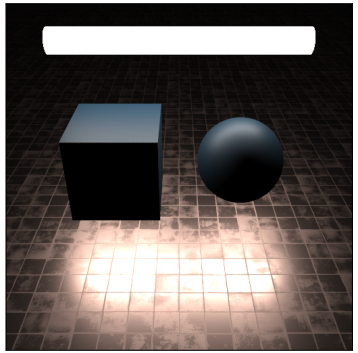


With the formulation that we have seen, we obtain $S = U \cdot S/U$.

By itself, this formulation doesn't seem very useful: we are defining the result we want to compute as a function of the result we want to compute. Of course, since we don't know S, we cannot directly compute S/U. The trick is that we use ray tracing and denoising to obtain a good estimate of S/U.

Our algorithm

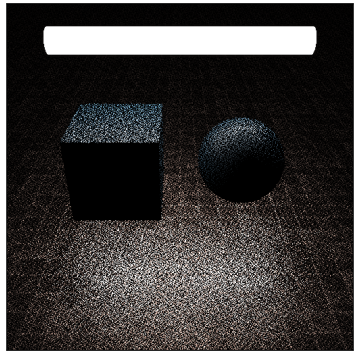
U



$$\int_\Omega \text{BRDF} \times \text{Light}$$

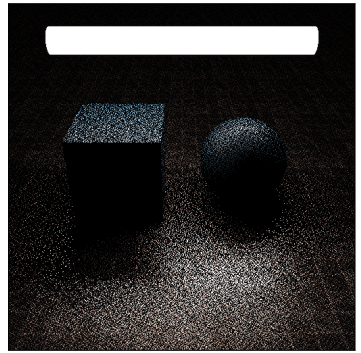
analytic

U_N



$$\frac{1}{N}\sum_{n=1}^{N}\frac{\text{BRDF}(\omega_n)\times\text{Light}(\omega_n)}{\text{PDF}(\omega_n)}$$

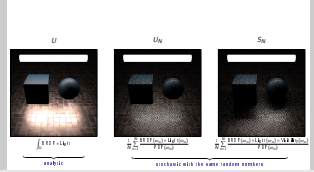
S_N



$$\frac{1}{N}\sum_{n=1}^{N}\frac{\text{BRDF}(\omega_n)\times\text{Light}(\omega_n)\times\text{Visibility}(\omega_n)}{\text{PDF}(\omega_n)}$$

stochastic with the same random numbers

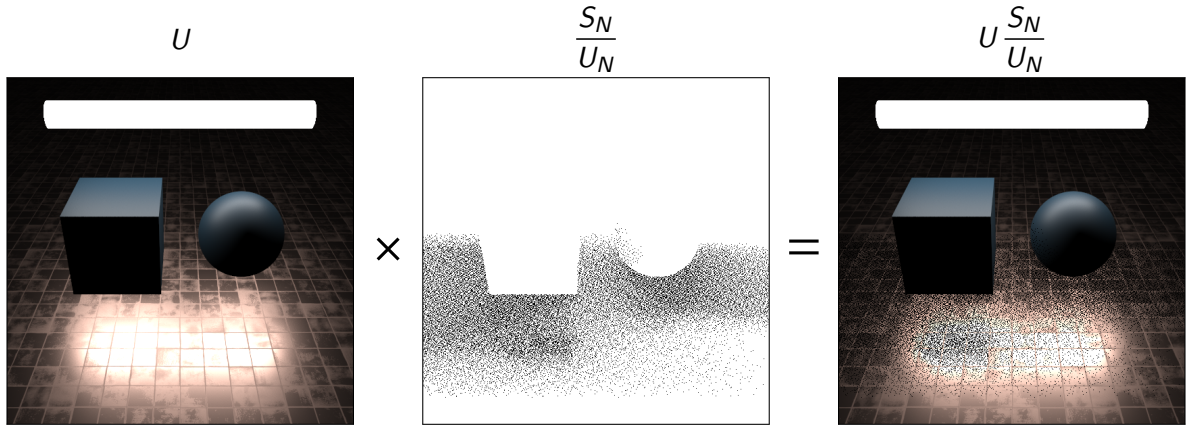
Combining Analytic Direct Illumination and Stochastic Shadows



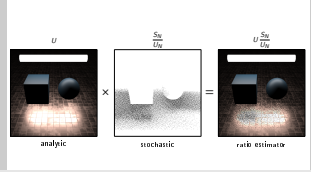
Our algorithm computes three images: the analytic U using our area-lighting framework, a stochastic estimate U_N of U, and a stochastic estimate S_N of S. The subscript N stands for the number of samples per pixel.

An important point is to use exactly the same random numbers for U_N and S_N. First, thanks to this, most of the computations for U_N and S_N are the same and can be factorized. Second, and more importantly, since U_N and S_N are evaluated stochastically, their terms produce some variance. But the variance due to the BRDF and the Light is the same in U_N and S_N. Thanks to this...

Our algorithm



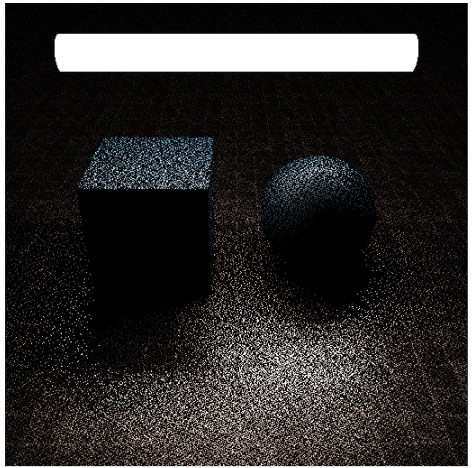
Combining Analytic Direct Illumination and Stochastic Shadows



...the variance cancels out when we divide S_N by U_N to obtain the noisy shadow image. The shadow image is noisy only because of the visibility contained in S_N that is absent from U_N, since the variations due to the BRDF and the Light have cancelled out. For instance, all of the unshadowed regions of the image are free of noise in the shadow image.

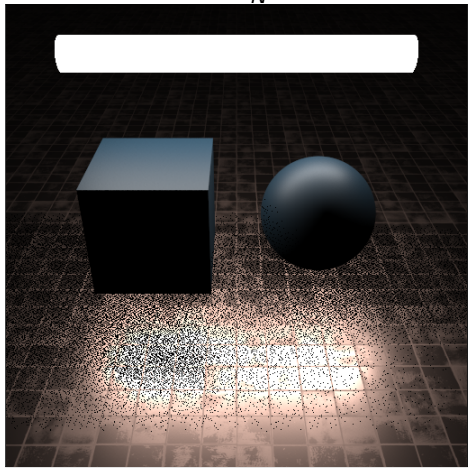
Our algorithm

S_N



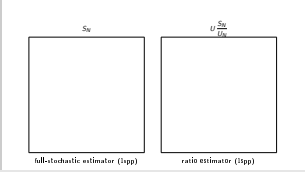
full-stochastic estimator (1spp)

$U \frac{S_N}{U_N}$



ratio estimator (1spp)

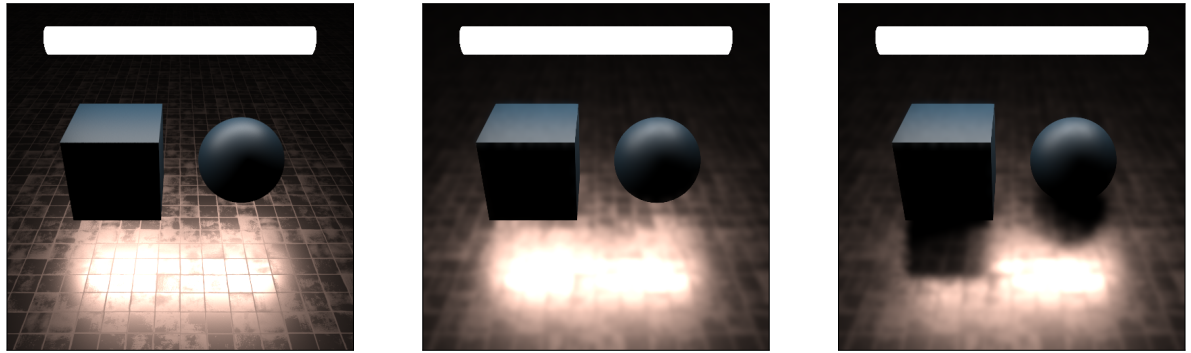
Combining Analytic Direct Illumination and Stochastic Shadows



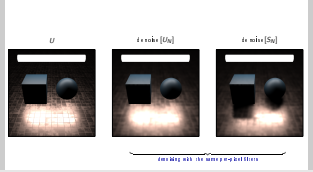
This composition yields a Monte Carlo estimator that is known as a ratio estimator. This ratio estimator of the direct illumination is better than the classic full-stochastic estimator. This is because all the information related to the shading (the BRDF and the Light) is provided by the analytic U (which has no noise) and only the visibility is stochastic. This results in significantly reduced variance and noise-free unshadowed regions of the image. ⚠This slide is animated (works with Acrobat Reader).

Our algorithm

U denoise$[U_N]$ denoise$[S_N]$



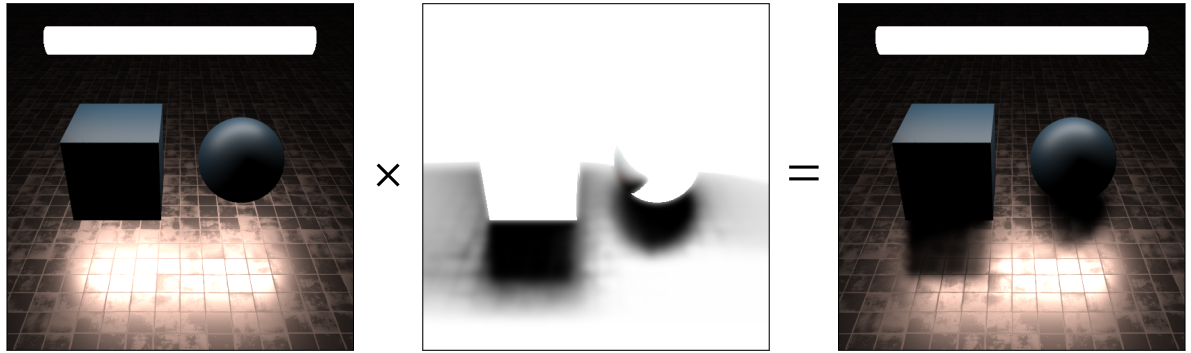
denoising with the same per-pixel filters



Still, we don't want noise at all in our real-time renders. So we are going to apply a real-time denoiser on the stochastic estimators U_N and S_N. For the same reason as for the random numbers, the per-pixel denoising weights should be the same for the two images.

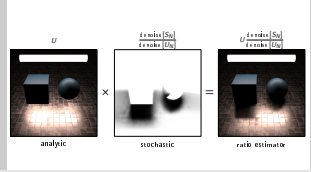
Our algorithm

U | $\frac{\text{denoise}[S_N]}{\text{denoise}[U_N]}$ | $U\frac{\text{denoise}[S_N]}{\text{denoise}[U_N]}$



analytic | stochastic | ratio estimator

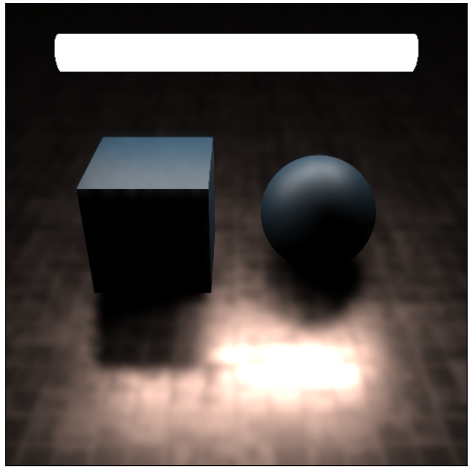
Combining Analytic Direct Illumination and Stochastic Shadows



After the composition we obtain a denoised shadow image. Because of the aggressive real-time denoiser, the shadow (and only the shadow) image is overblurred. In contrast, the shading details contained in the analytic U image remain sharp since they are not denoised.

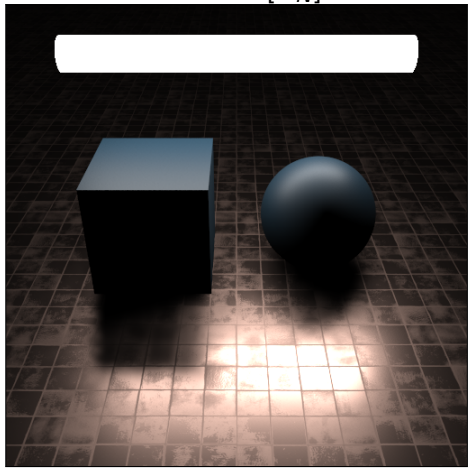
Our algorithm

denoise$[S_N]$



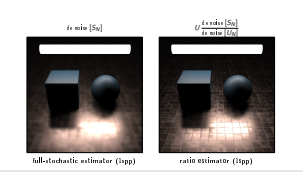
full-stochastic estimator (1spp)

$U \dfrac{\text{denoise}[S_N]}{\text{denoise}[U_N]}$



ratio estimator (1spp)

Combining Analytic Direct Illumination and Stochastic Shadows



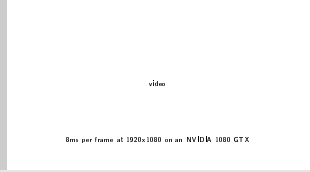
We obtain a much better result than a denoised full-stochastic estimator, where all the shading details are overblurred. With our formulation, the shading details are preserved. Thus, even with a low number of samples per pixel and a low-quality denoiser, we can still get good looking results.

Our algorithm

video

8ms per frame at 1920x1080 on an NVIDIA 1080 GTX

Combining Analytic Direct Illumination and Stochastic Shadows

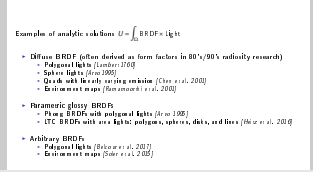


Our algorithm

Examples of analytic solutions $U = \int_\Omega \text{BRDF} \times \text{Light}$

- **Diffuse BRDF (often derived as form factors in 80's/90's radiosity research)**
  - **Polygonal lights** *[Lambert1760]*
  - **Sphere lights** *[Arvo1995]*
  - **Quads with linearly varying emission** *[Chen et al. 2001]*
  - **Environment maps** *[Ramamoorthi et al. 2001]*
- **Parametric glossy BRDFs**
  - **Phong BRDFs with polygonal lights** *[Arvo 1995]*
  - **LTC BRDFs with area lights: polygons, spheres, disks, and lines** *[Heitz et al. 2016]*
- **Arbitrary BRDFs**
  - **Polygonal lights** *[Belcour et al. 2017]*
  - **Environment maps** *[Soler et al. 2015]*

Combining Analytic Direct Illumination and Stochastic Shadows

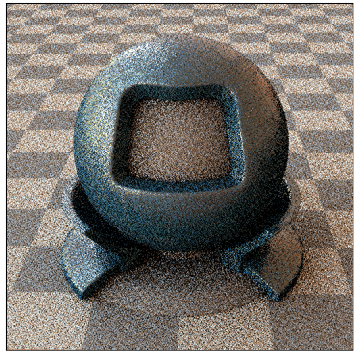


In the paper and the presentation, we largely focused on area-lighting examples, which were our first source of motivation. However, our algorithm is not limited to area lighting: it can be applied to any BRDF and lighting model as long as an analytic U is available.

Fortunately, deriving analytic U for different scenarios is an active research topic and many solutions are already available. For instance, form factors derived from the time of radiosity research provide several analytic solutions for diffuse BRDFs. Additionally, some recent works provide solutions based on efficient SH decompositions of arbitrary BRDFs.

Our algorithm

S_N

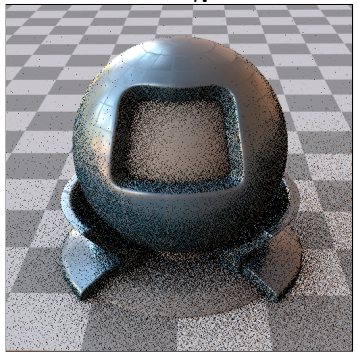


state-of-the-art estimator

- Multiple Importance Sampling
- VNDF sampling for GGX
- stratified sampler
- ...

1spp

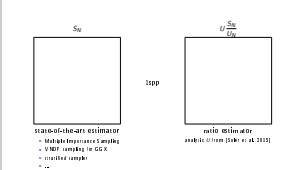
$U \frac{S_N}{U_N}$



ratio estimator

analytic U from [Soler et al. 2015]

Combining Analytic Direct Illumination and Stochastic Shadows

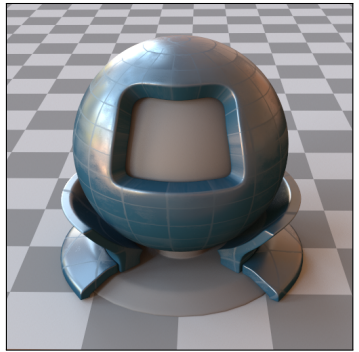


For instance, we can use the solution proposed by Soler et al. to obtain a ratio estimator of the direct illumination for arbitrary materials and environment lighting, which outperforms the state-of-the-art full-stochastic estimator used in offline rendering.

The analytic solution U by Soler et al. is from *Efficient and Accurate Spherical Kernel Integrals using Isotropic Decomposition*, TOG 2015

Our algorithm

S_N

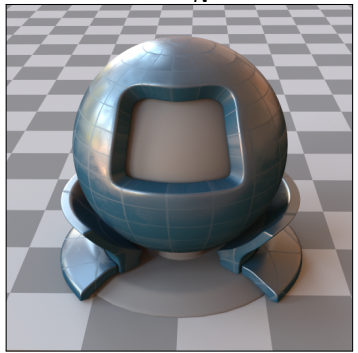


state-of-the-art estimator

- Multiple Importance Sampling
- VNDF sampling for GGX
- stratified sampler
- ...

1024spp

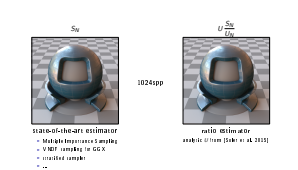
$U\frac{S_N}{U_N}$



ratio estimator

analytic U from [Soler et al. 2015]

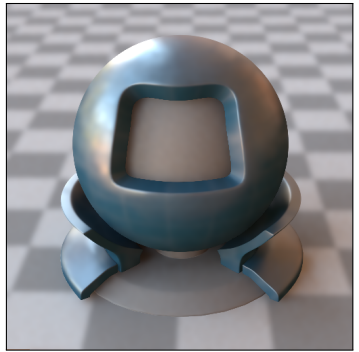
Combining Analytic Direct Illumination and Stochastic Shadows



Both estimators converge towards the same result. What we propose is thus not just a real-time rendering solution, it is also a Monte Carlo estimator of the direct illumination that would perfectly fit and improve an offline renderer as well.

Our algorithm

$\text{denoise}[S_N]$

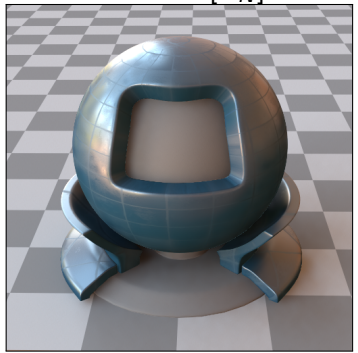


state-of-the-art estimator

- Multiple Importance Sampling
- VNDF sampling for GGX
- stratified sampler
- ...

1spp
+
denoising

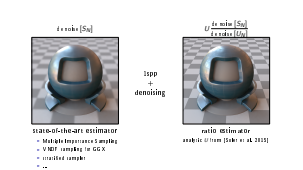
$U\,\frac{\text{denoise}[S_N]}{\text{denoise}[U_N]}$



ratio estimator

analytic U from [Soler et al. 2015]

Combining Analytic Direct Illumination and Stochastic Shadows



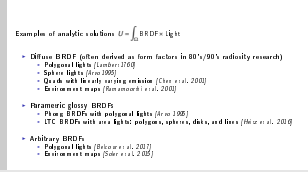
As expected, our formulation also improves the denoising.

Our algorithm

Examples of analytic solutions $U = \int_\Omega \text{BRDF} \times \text{Light}$

- **Diffuse BRDF (often derived as form factors in 80's/90's radiosity research)**
  - **Polygonal lights** *[Lambert1760]*
  - **Sphere lights** *[Arvo1995]*
  - **Quads with linearly varying emission** *[Chen et al. 2001]*
  - **Environment maps** *[Ramamoorthi et al. 2001]*
- **Parametric glossy BRDFs**
  - **Phong BRDFs with polygonal lights** *[Arvo 1995]*
  - **LTC BRDFs with area lights: polygons, spheres, disks, and lines** *[Heitz et al. 2016]*
- **Arbitrary BRDFs**
  - **Polygonal lights** *[Belcour et al. 2017]*
  - **Environment maps** *[Soler et al. 2015]*

Combining Analytic Direct Illumination and Stochastic Shadows



Though deriving analytic solutions U such as these is an active research topic, it has admittedly been left a bit on the side by the rendering community. The reason is that people do not care so much about computing analytic solutions if they cannot be shadowed.

We have shown that it is actually possible to add correct shadows on top of the analytic solutions and that doing so yields a ratio estimator of the direct illumination that exhibits low noise and performs well with denoisers. With this observation, we hope that the interest in this research topic might be strengthened in the future.

Conclusion

The insights that we share in this paper and the algorithm we came up with are the result of a long journey. Before arriving there, we tried many other things that did not work out for us.

We tried to look for perfect importance sampling solutions of the BRDF-Light product, we tried control-variate algorithms...

The method we propose, the ratio estimator, can be seen as a multiplicative control-variate method, which emulates a perfect importance sampling of the BRDF-Light product. But it performs much better than a classic control-variate (see the comparisons in the paper) and does not require complex machinery, unlike some BRDF-Light product samplers.

Still, looking back at this paper, it somehow feels like...

joined forces to crack this equation

$$a \cdot \frac{b}{a} = b$$

unity LUCASFILM Ltd NVIDIA
joined forces to crack this equation
$a \cdot \frac{b}{a} = b$

...it took no less than three big companies to come up with a formulation that looks obvious.

It is true that this formulation looks kind of obvious. However, we think that it is the kind of idea that looks obvious in retrospect but not obvious to come up with in the first place. If it were, why did real-time people want to ray trace soft shadows? Why did offline-rendering people publish several papers on control variates and (to our knowledge) never published a ratio estimator, which is simpler and better?

Furthermore, it is somewhat cool that there was still so much improvement to find in such a simple idea. Simple ideas that become obvious once you see them are often the ones that remain.

Combining analytic direct illumination and stochastic shadows

DXR demo available soon in NVIDIA's Falcor engine.

With this, we have shown that the naive approach of "shooting rays towards the light source" is flawed, which was the important point we wanted to make within the real-time rendering community. We analyzed the problem and found the right way to define correct ray-traced shadows and an algorithm to compute them, which turns out to be pretty simple and practical.

We are very excited about the future of hybrid ray tracing and we are looking forward to seeing people making use of our algorithm in the near future!

⚠This slide is animated (works with Acrobat Reader).