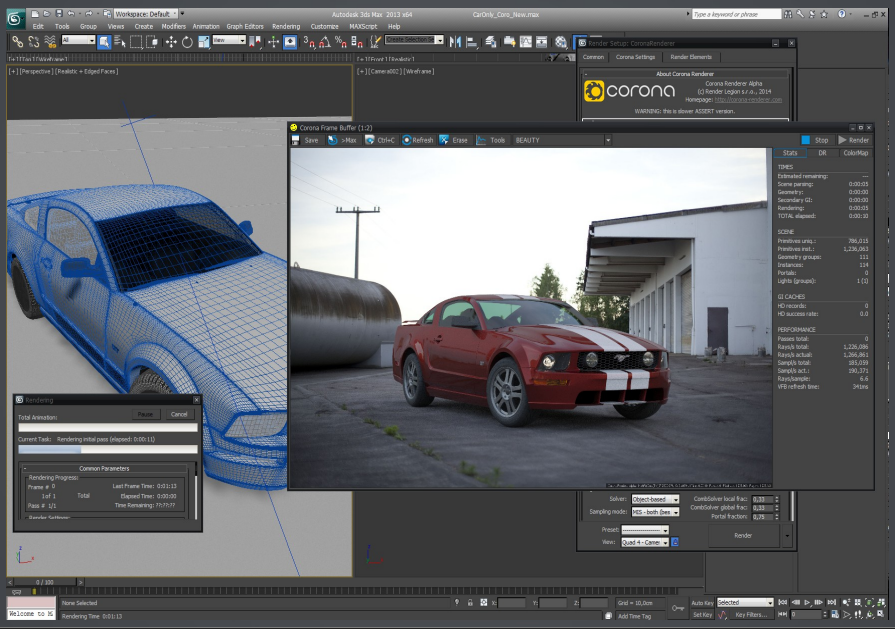
These are slides for the presentation „It's all about usability“ given by Ondra Karlík on 2014/08/12 at SIGGRAPH 2014 in Vancouver.

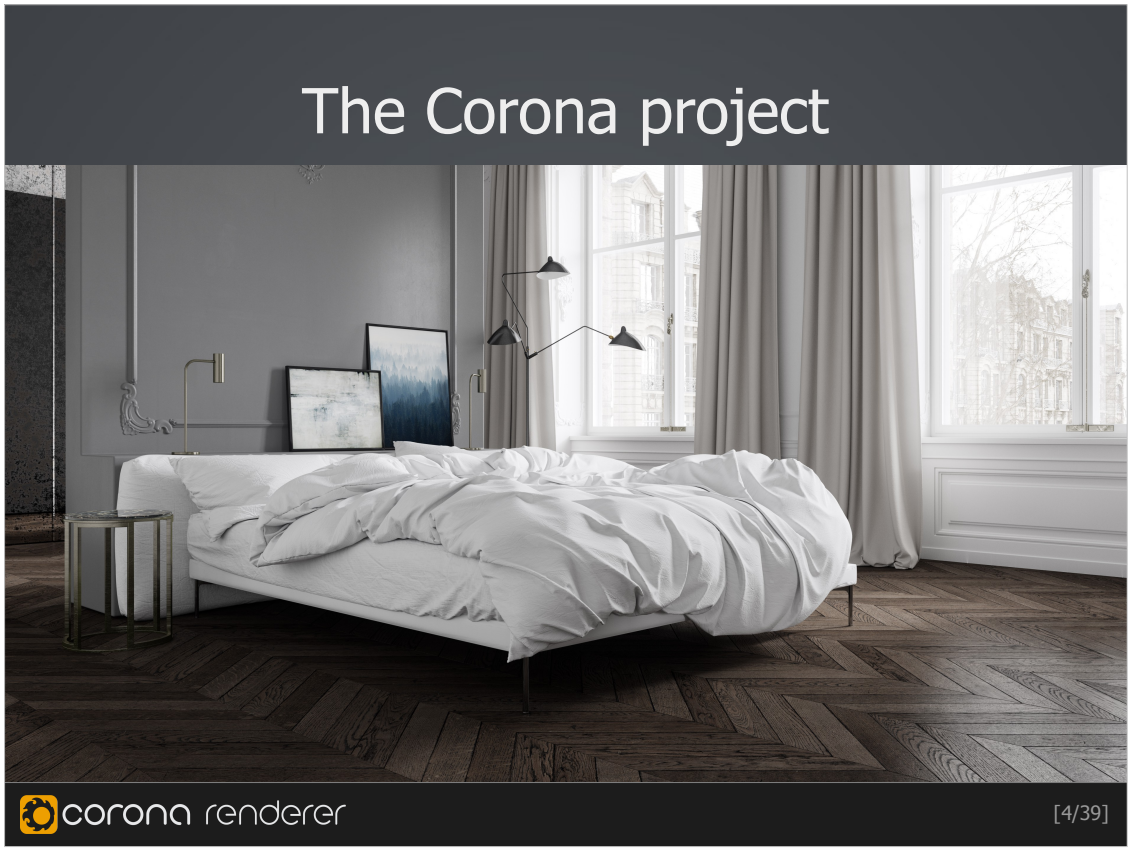
It is a part of the course „Recent Advances in Light Transport Simulation: Some Theory and a lot of Practice“.

Corona is a photorealistic ray tracer, used by computer graphic artists for the final frame production rendering. It is focused on architectural and product visualization.

3ds max integration

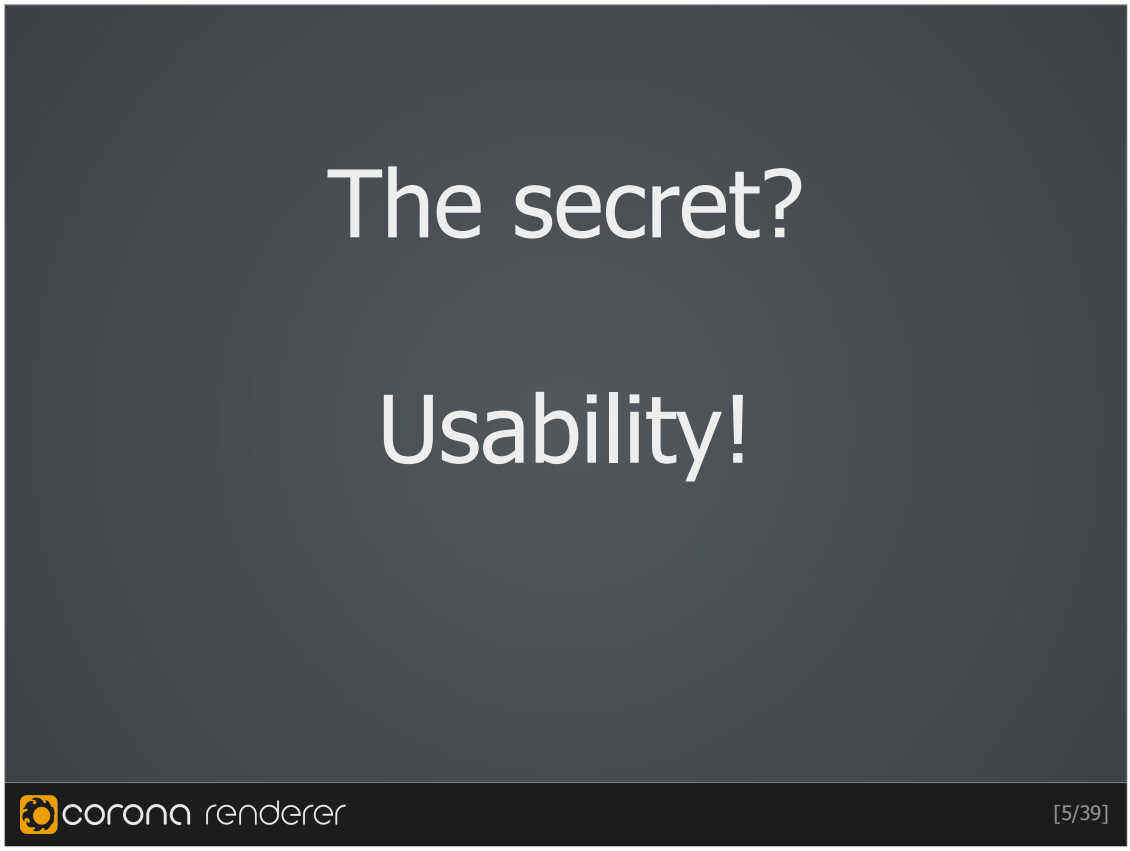


Corona is fully integrated in 3ds max, but there is also a standalone application version. Plugins for other software are on the way.

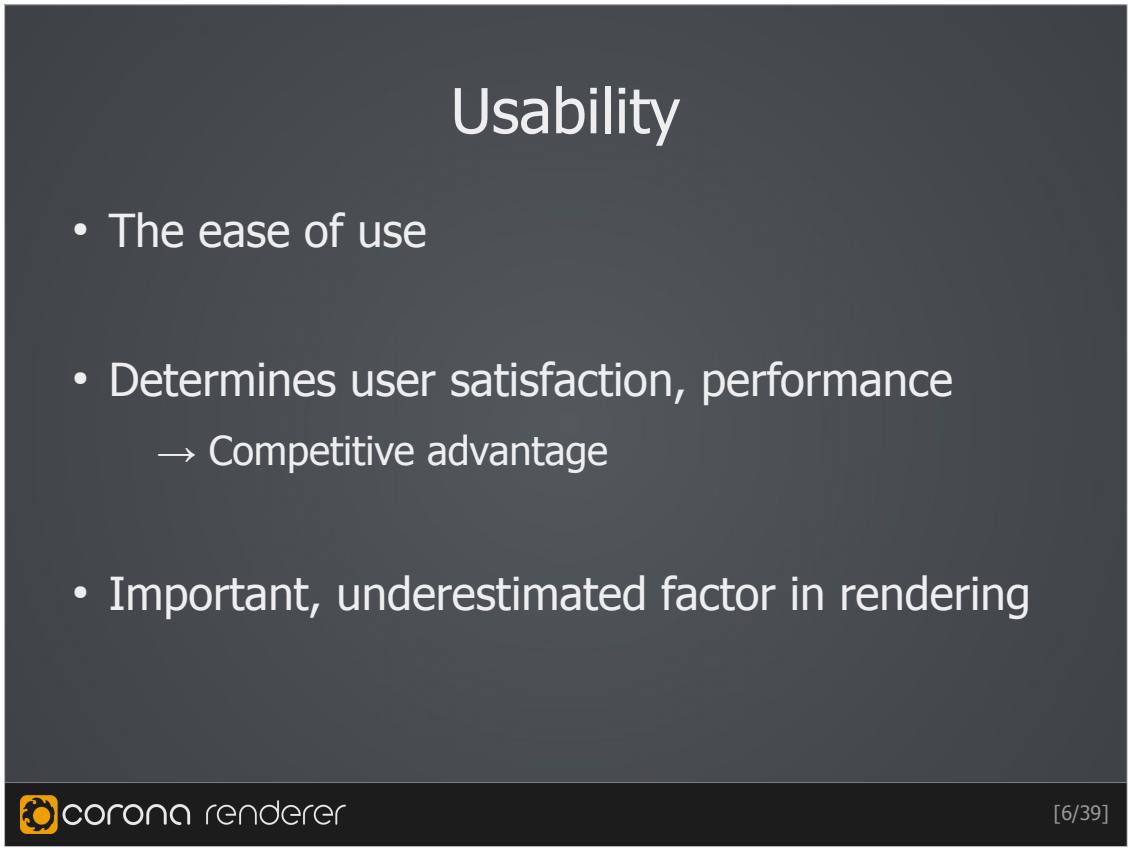


Corona is currently free to use, and the first commercial release is being prepared at the moment.

The project started five years ago as a one-man show, but now there is a team of three working on it, and it had attracted thousands of users, and a lot of attention in the archviz industry.



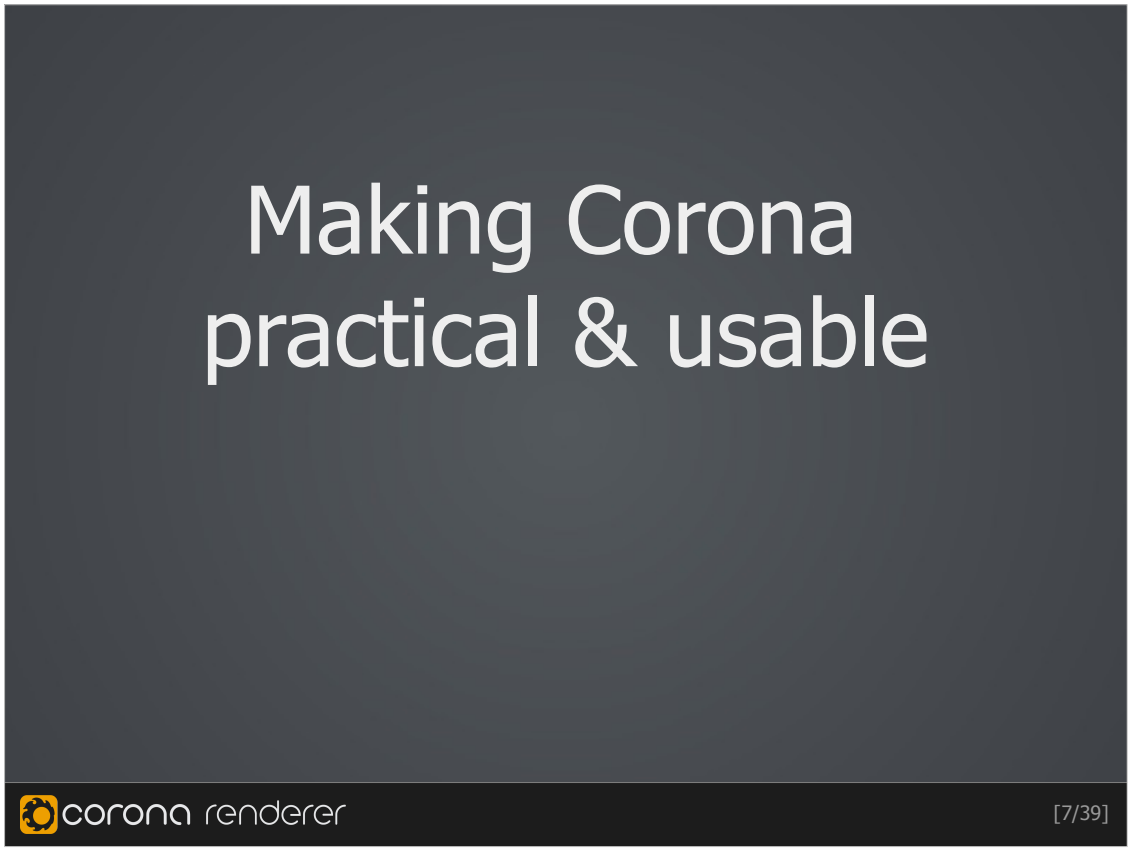
I believe that there is one secret responsible for this success: usability.



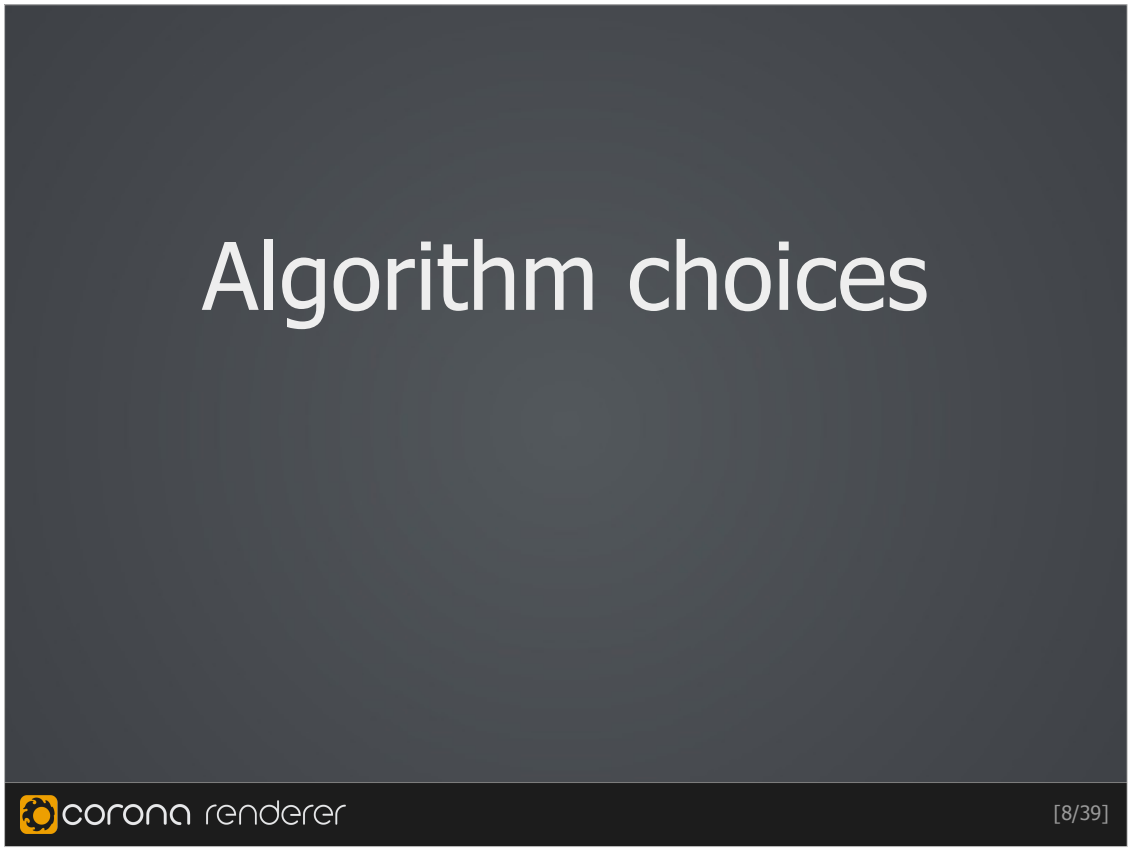
Usability means how easy is the software to use in practice. Corona is not the fastest, most physical, or most feature-complete renderer on the market, but I believe it is the simplest one to work with.

Usability is what ultimately determines artist satisfaction and performance. This is always true, but it is especially important when dealing with people with little or no technical background.

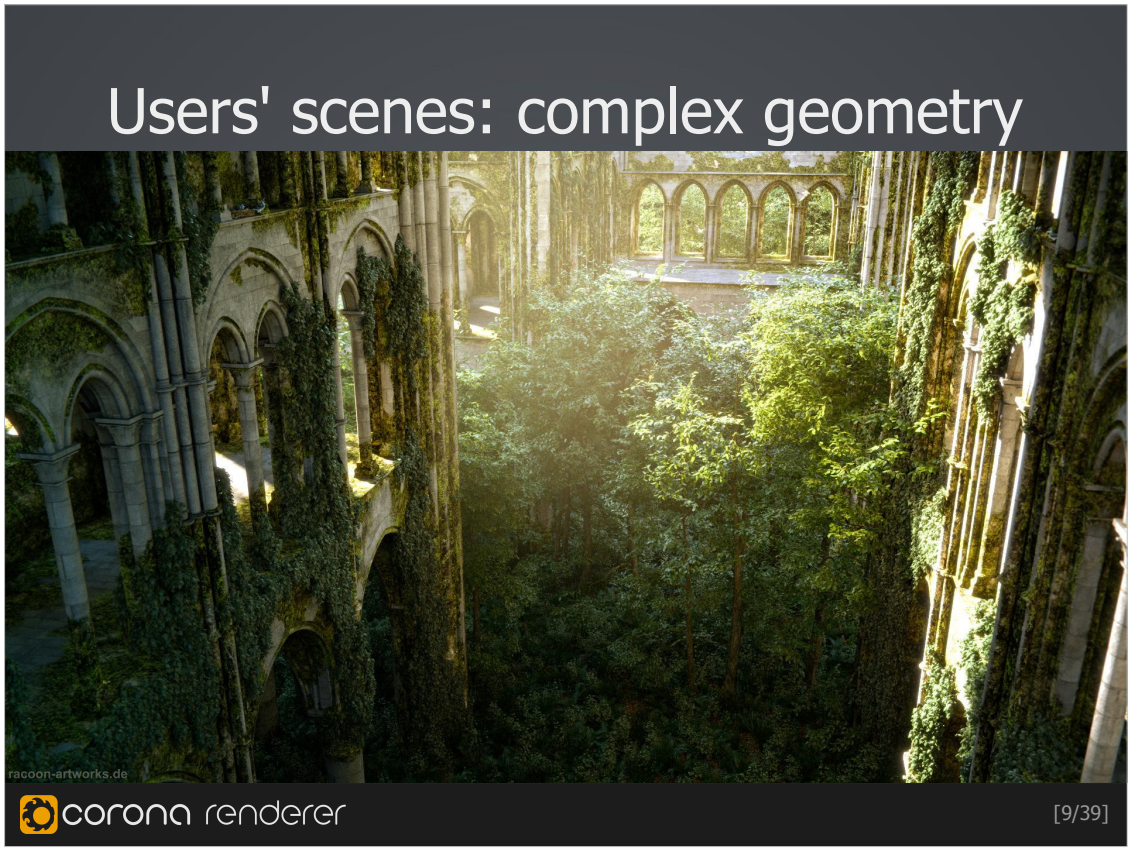
It is important to note that this factor is extremely underestimated in rendering.



I will go over some specific problems we had to solve in Corona to make it really practical and usable.



First is the issue with the biggest software design impact – what rendering algorithm should Corona use.



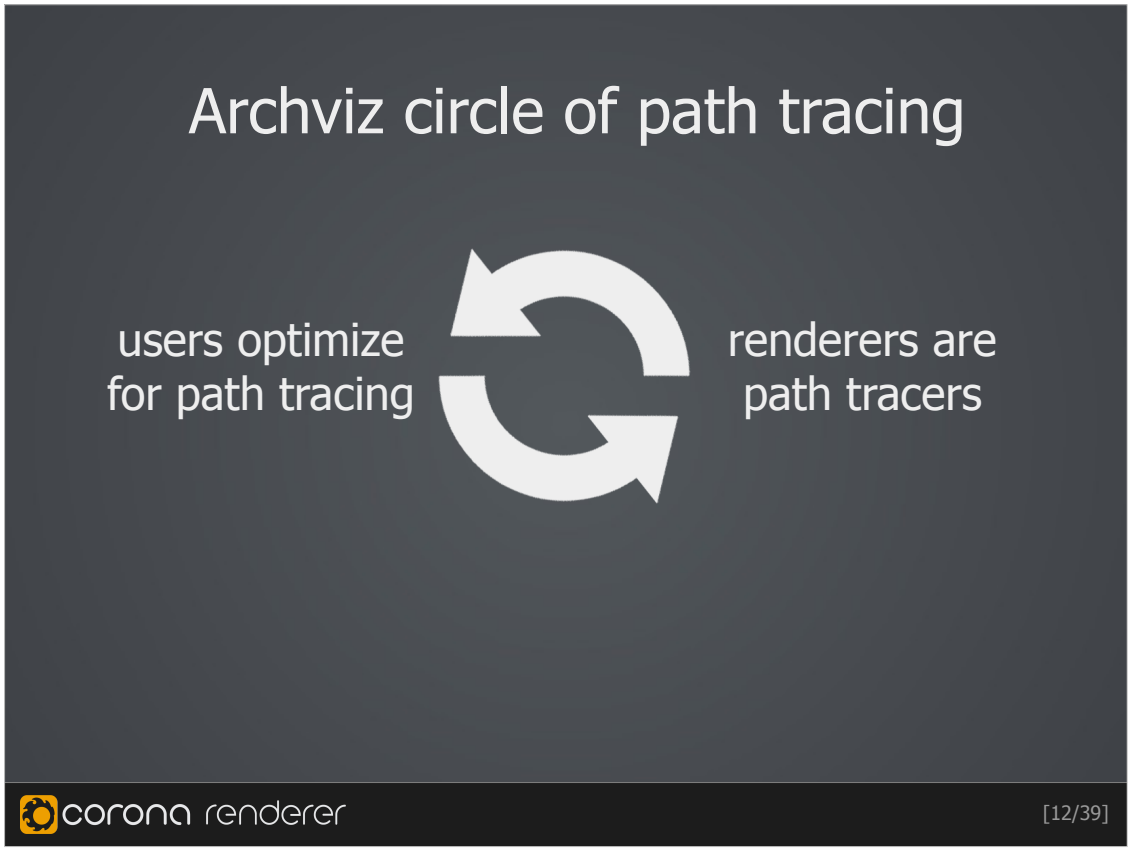
It is important to first take a look at the scenes users produce.

Most of the archviz scenes are very complex, but only from the geometry point of view. What is interesting is that they contain mostly diffuse materials and direct lighting.

Here is an example from another shot of the cathedral project. This is the end result with full global illumination solution.

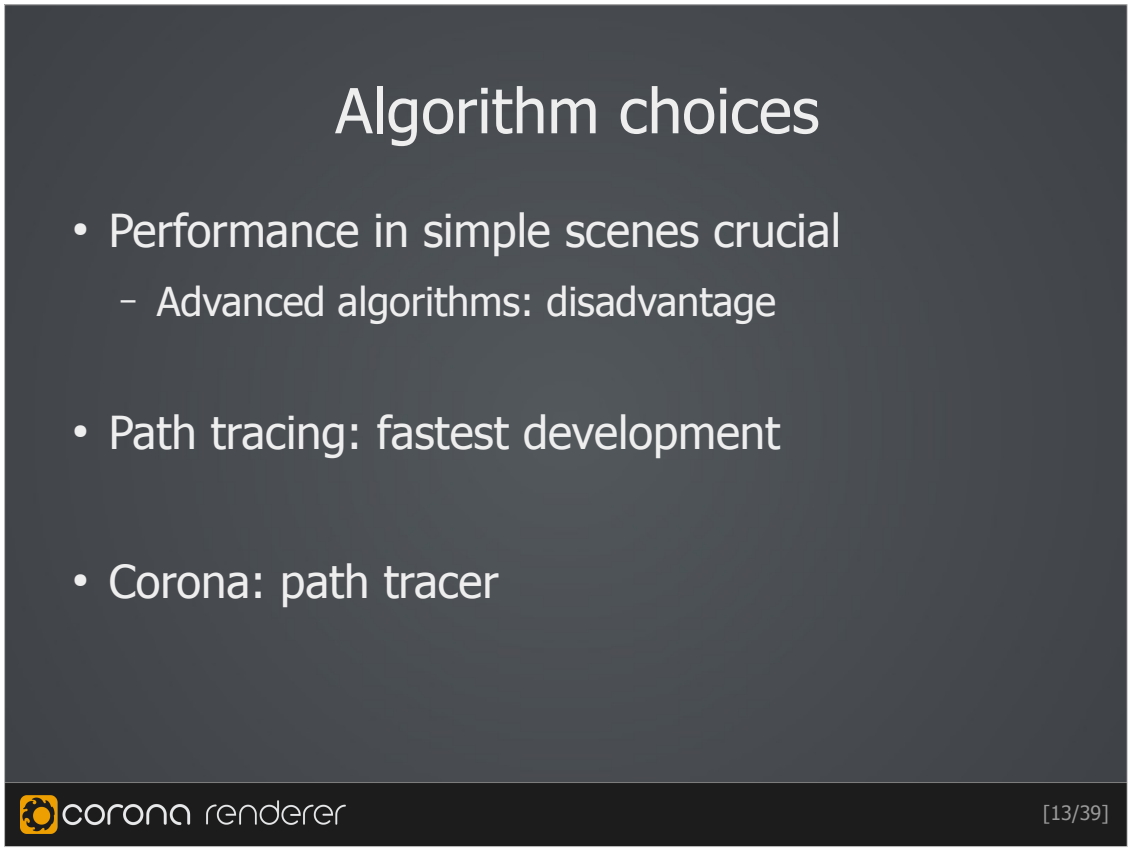
This is the direct light only. Even though global illumination si clearly needed, it does not dominate the image.

This makes the scene extremely simple to render with just the simple path tracing.



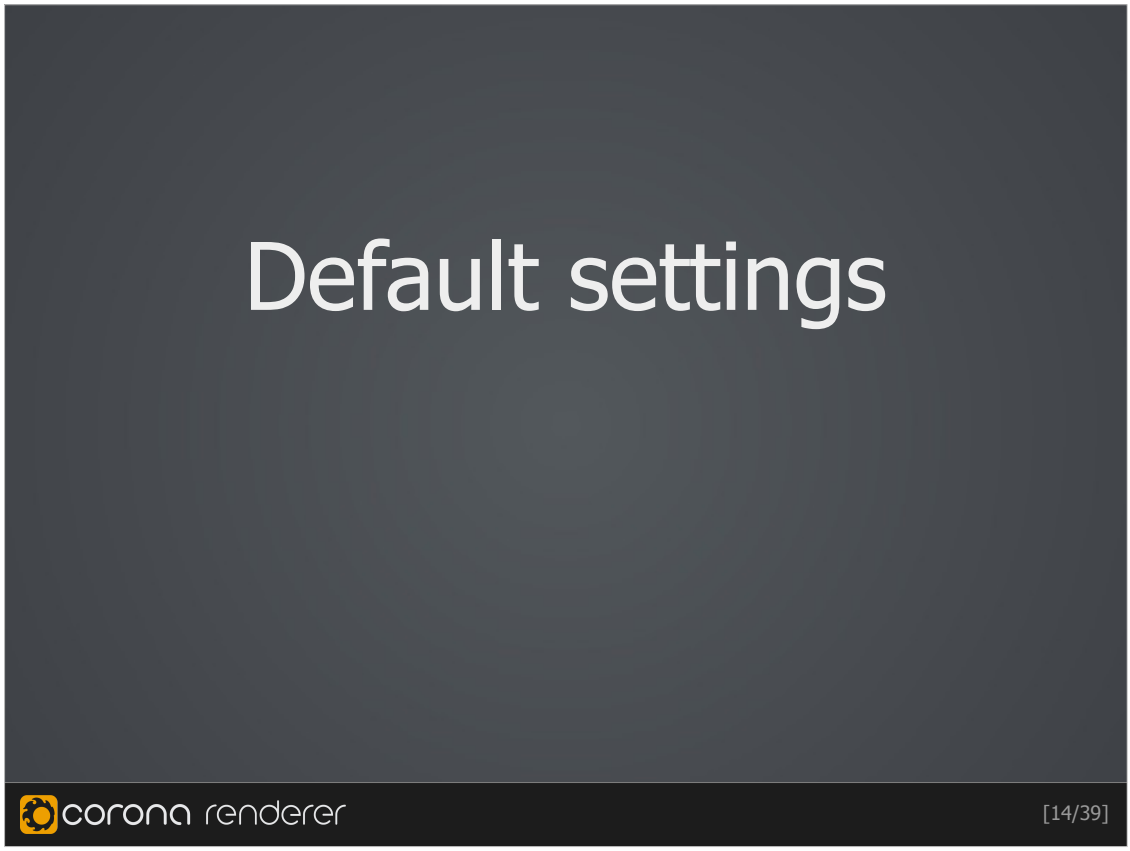
In the visualization field, there is a vicious circle: most renderers are only path tracers with multiple importance sampling. To get acceptable render times, users have learned to optimize their scenes for path tracing. Which in turn makes path tracing the best algorithm choice to implement for new renderers.

This cycle prevents introducing newer, advanced algorithms to practice.

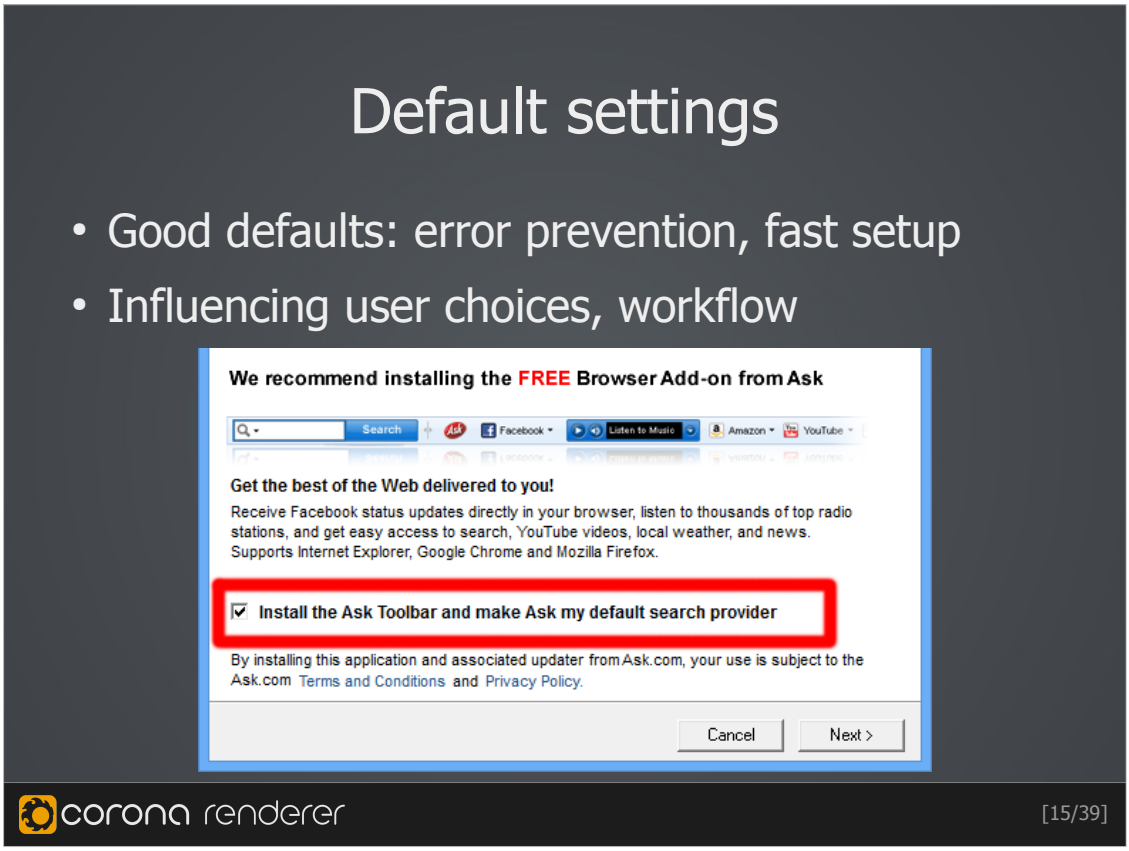


If one would introduce some advanced algorithm, say Vertex Connection and Merging (VCM), users would inevitably compare it to path tracing in scenes optimized for path tracing. That is not a „fair“ comparison and VCM would lose.

This, together with the fact that users demand a lot of artistic and nonphysical features that do not work well with advanced algorithms, is the reason that Corona is almost exclusively path tracer, even though it has the advanced algorithms (including VCM) implemented.

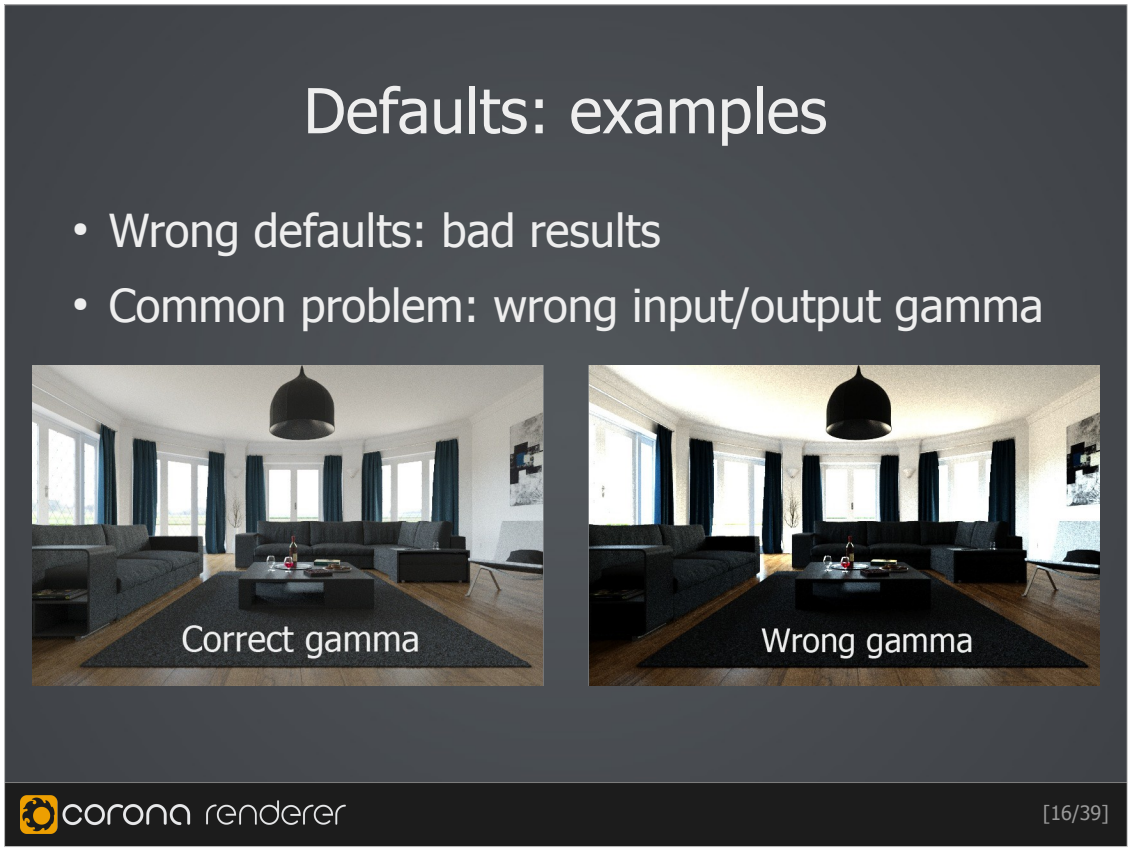


Next issue is probably the most overlooked one: the default render settings.



Each parameter in UI has to have some default value. It is extremely important for this value to work well in most cases, because novice users won't know which parameters they are supposed to change and how. Even expert users don't like having to tweak many parameters every time they create a new scene as it slows them down.

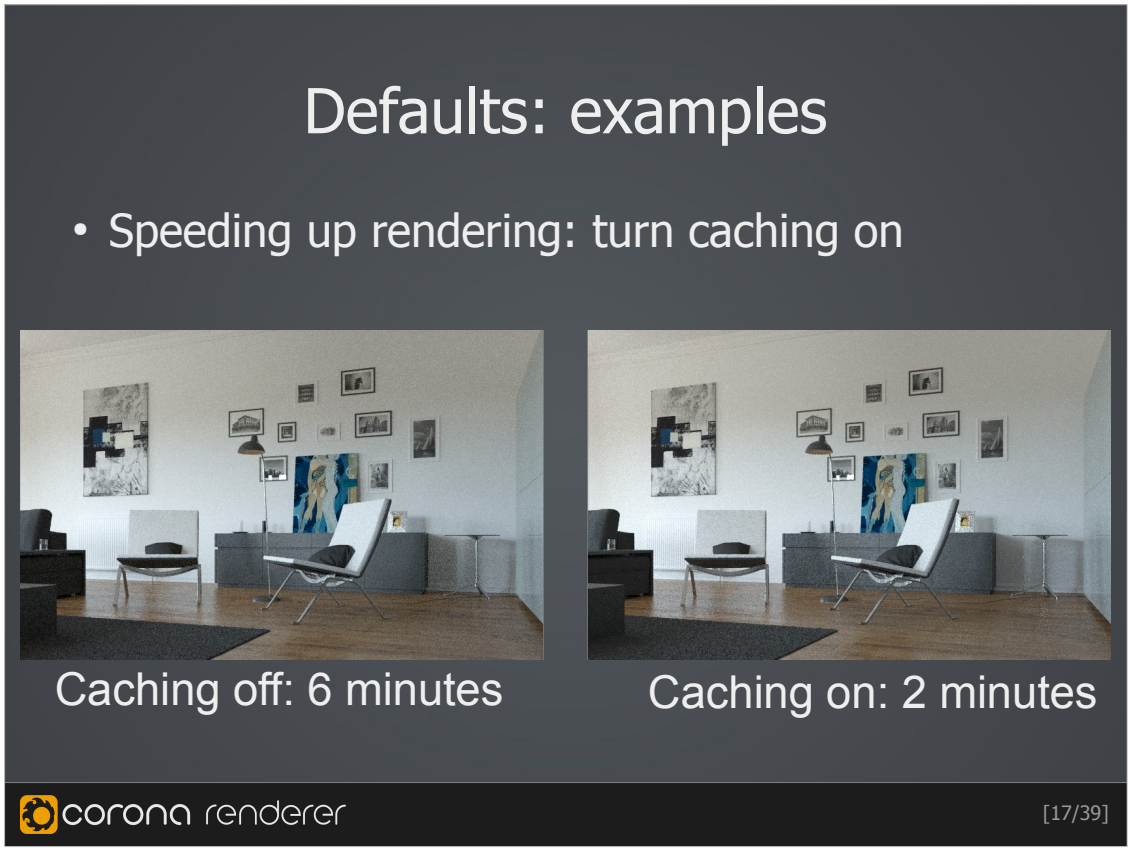
There is also a second very important effect: different defaults can greatly influence the way people use the software. Many have realized this before, and are widely using it. A very prominent example is common software installing additional unwanted spyware by default.



If the defaults are wrong, inexperienced users (who make the majority of user base) will make mistakes because of it.

For example: common problem in some 3D tools is incorrect handling of input and output gamma in some image formats (JPG, PNG, ...). 3ds max has the option to do it correctly (using gamma 2.2), but this was turned off by default, and had to be enabled in settings. It takes only about 10 seconds, but many inexperienced users do not know they have to do it before they start working.

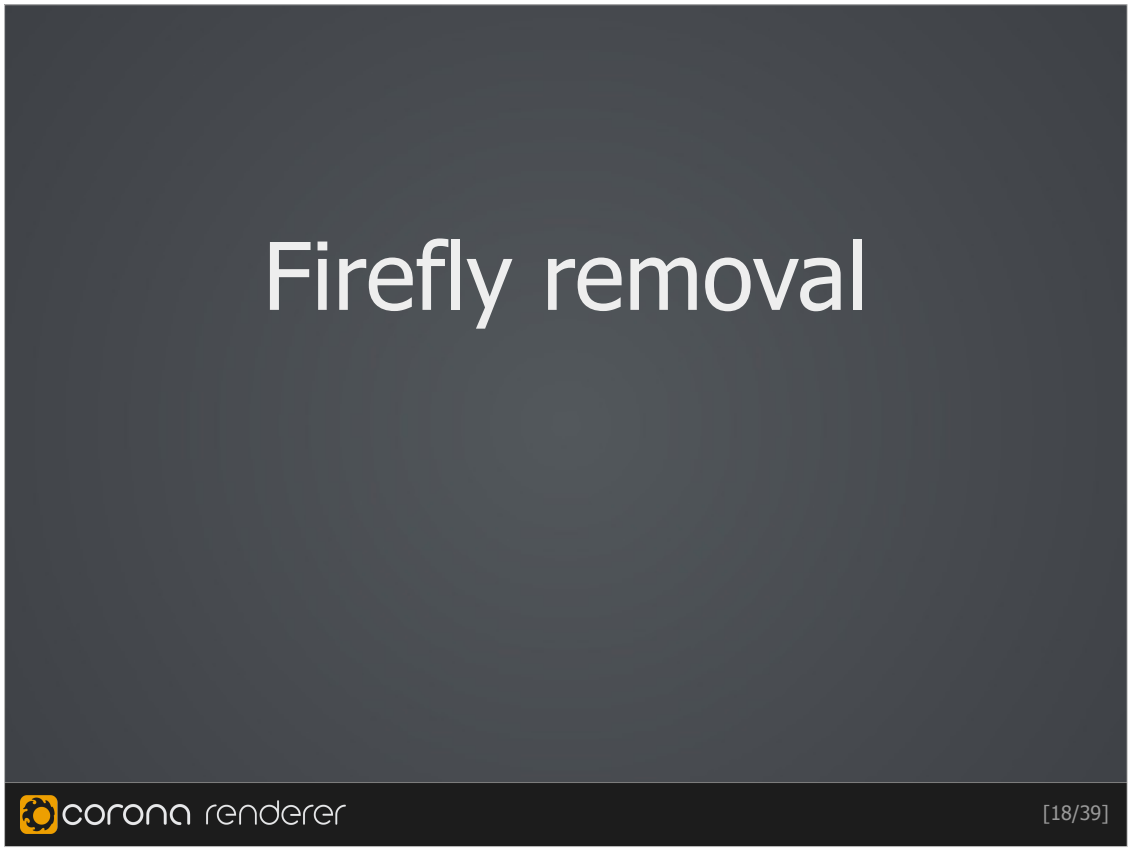
Because of this, they often produce wrong pictures with oversaturated textures, and burned-out whites (right image), and blame the software for it.



Next example: when the option to use partial irradiance caching was first added to Corona, it was off by default. We recommended to enable it for interior scenes, where it produces a decent speedup.

Later it was switched to on-by-default to save some clicking when setting up scenes, as most Corona scenes are interiors.

But what actually happened was that the Corona forum got filled with posts saying the renderer is now 3-4 times faster. Ordinary users simply were not aware that this feature even existed. This illustrates that even the best algorithm is completely useless if users don't know how to set it up.



Next issue is how to get rid of fireflies.

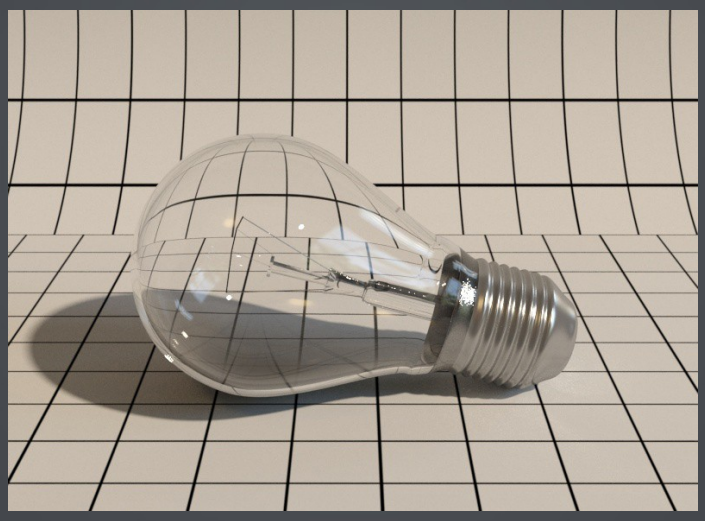
Fireflies are the extremely bright pixels that appear every time when rendering with a Monte Carlo-based algorithm.

In this image is a scene featuring heavy caustics that the path tracer is not able to resolve. It produces just a lot of fireflies. But even advanced algorithms like VCM produce fireflies sometimes.

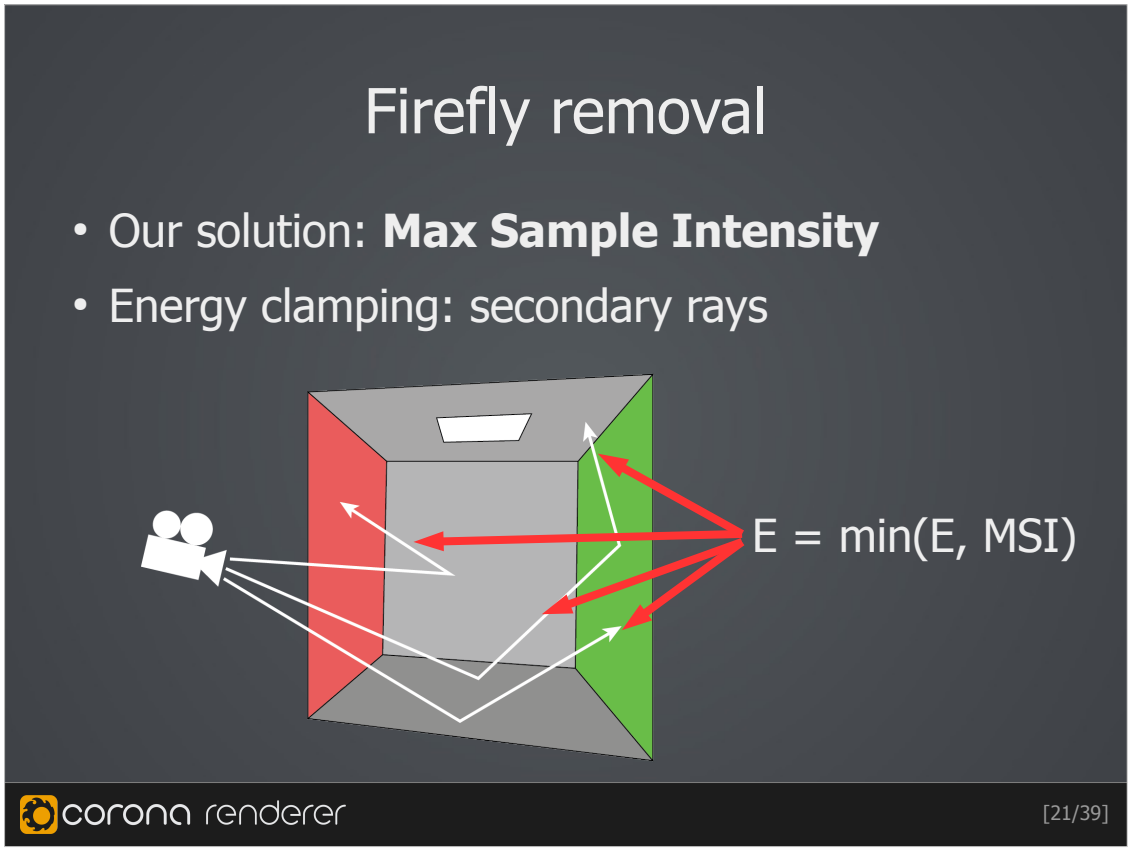
The universal industry solution for this is removing some energy from the picture to obtain biased, but noise-free image.

Firefly removal

- Traditional solution: excluding caustics light paths



The traditional solution to this problem is to remove entire classes of light paths that form caustics. This produces heavily biased result – the shadow is very dark.



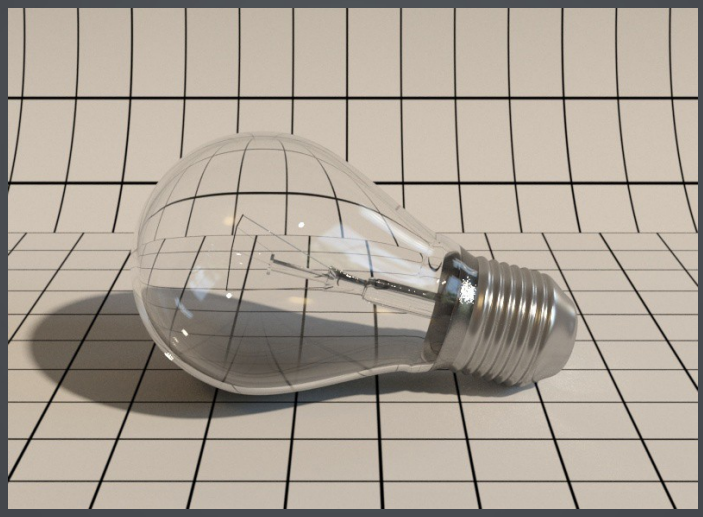
The previous solution is inadequate, as it removes even easy-to-compute caustics. We have replaced it with a method we call Max Sample Intensity.

It is very simple: for all secondary (global illumination) rays, we clamp their returned intensity (radiance) to be at most some user-defined constant. This is similar to VPL clamping.

This automatically removes all fireflies while keeping most of the computable light transport in the image, resulting in more plausible results.

Firefly removal

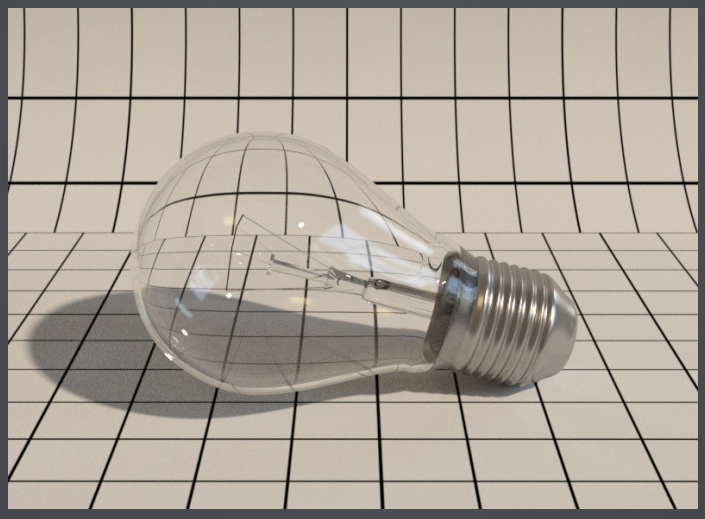
- Excluding caustics light paths - again



This is again the result of removing all caustic light paths.

Firefly removal

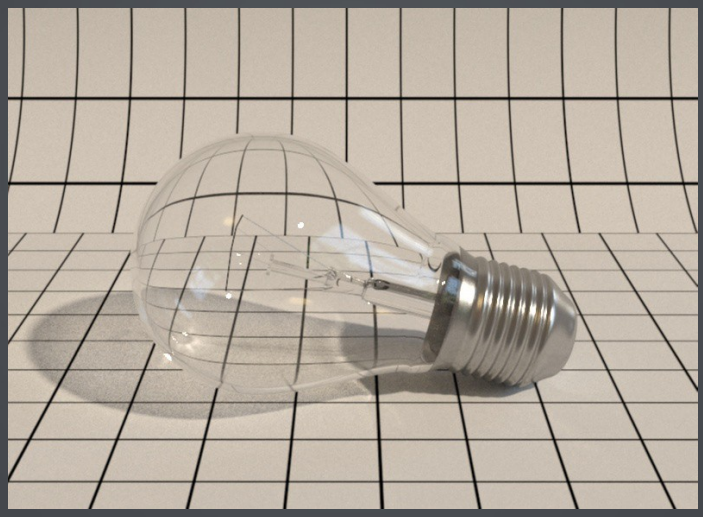
- Max sample intensity clamping



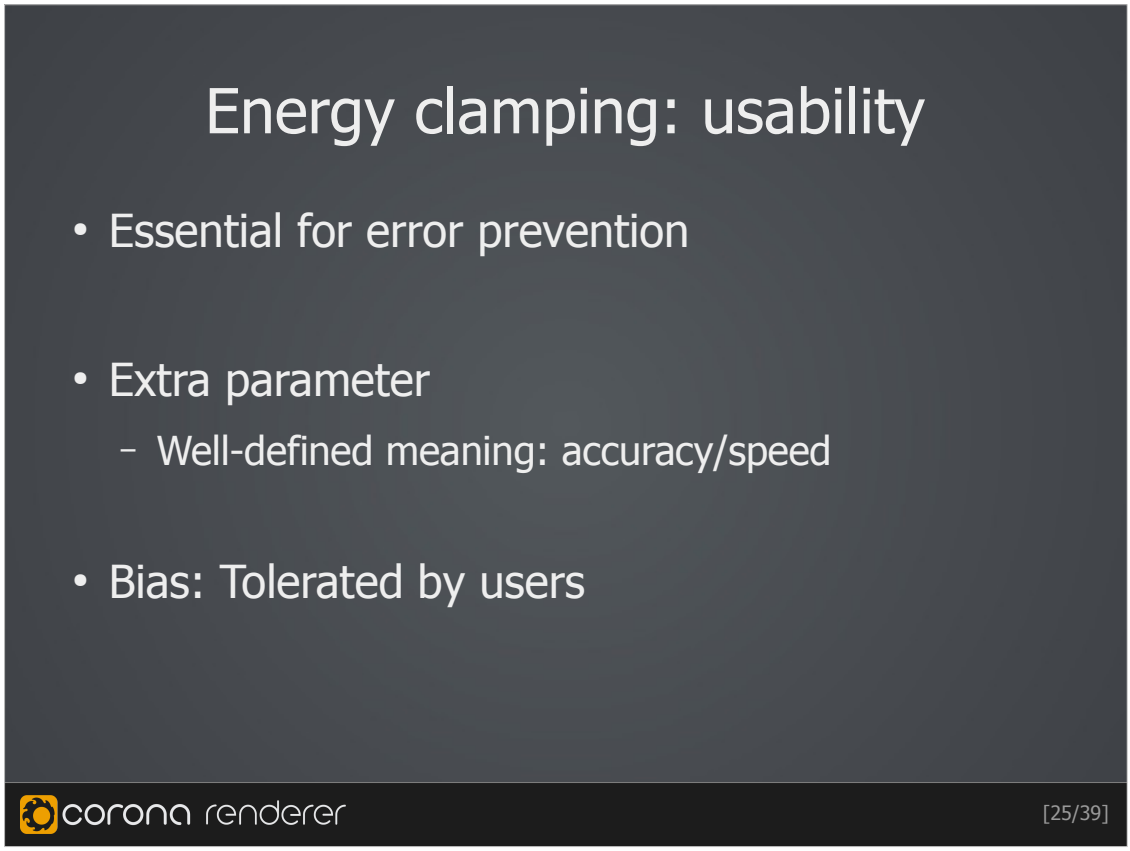
This is the result with max sample intensity. All fireflies are gone, and the shadow is not as dark as before.

Firefly removal

- Reference



But there is still room for improvement. This is the reference image rendered with VCM.

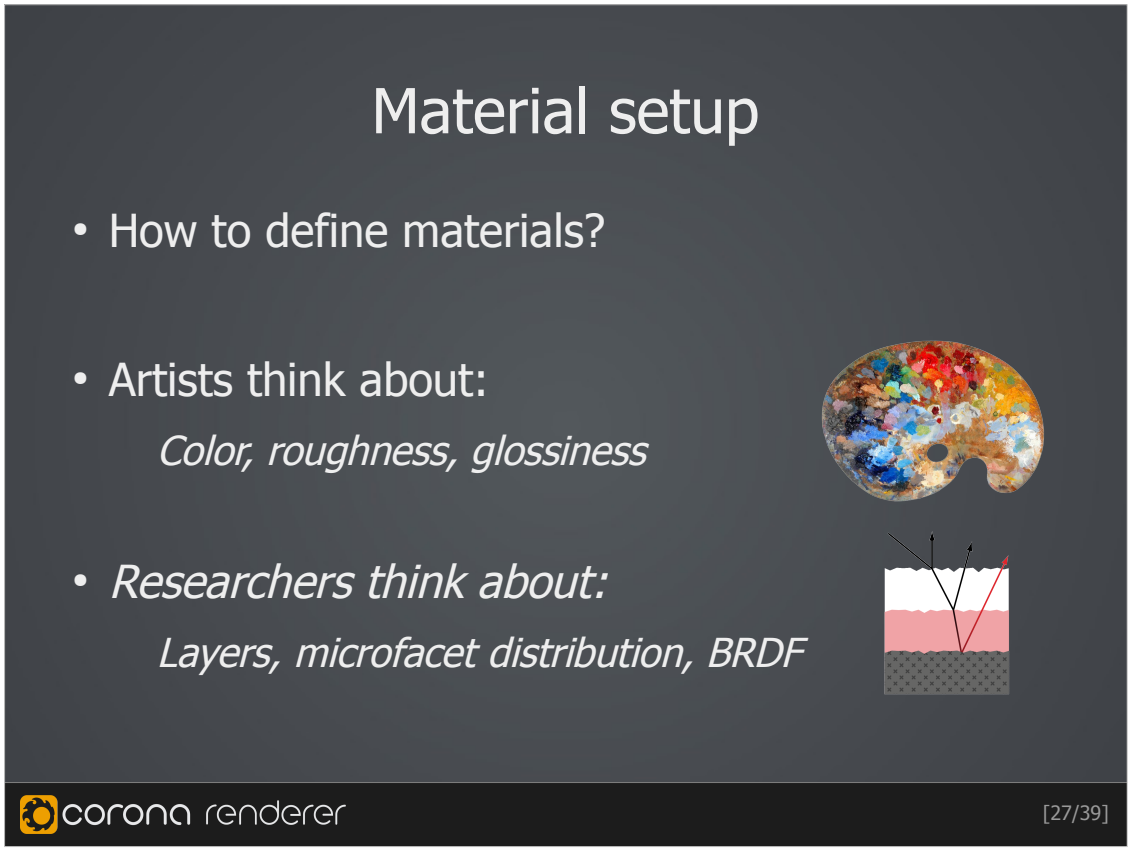


From the usability point of view it is essential that this is turned on by default. Because users have no knowledge of the algorithms, they do not know how to avoid fireflies by changing scene setup.

The method introduces an extra parameter, which is usually a bad thing. But this parameter has a very well-defined meaning: it is the ratio between rendering speed and rendering accuracy - so it adds flexibility for power users while not being confusing.

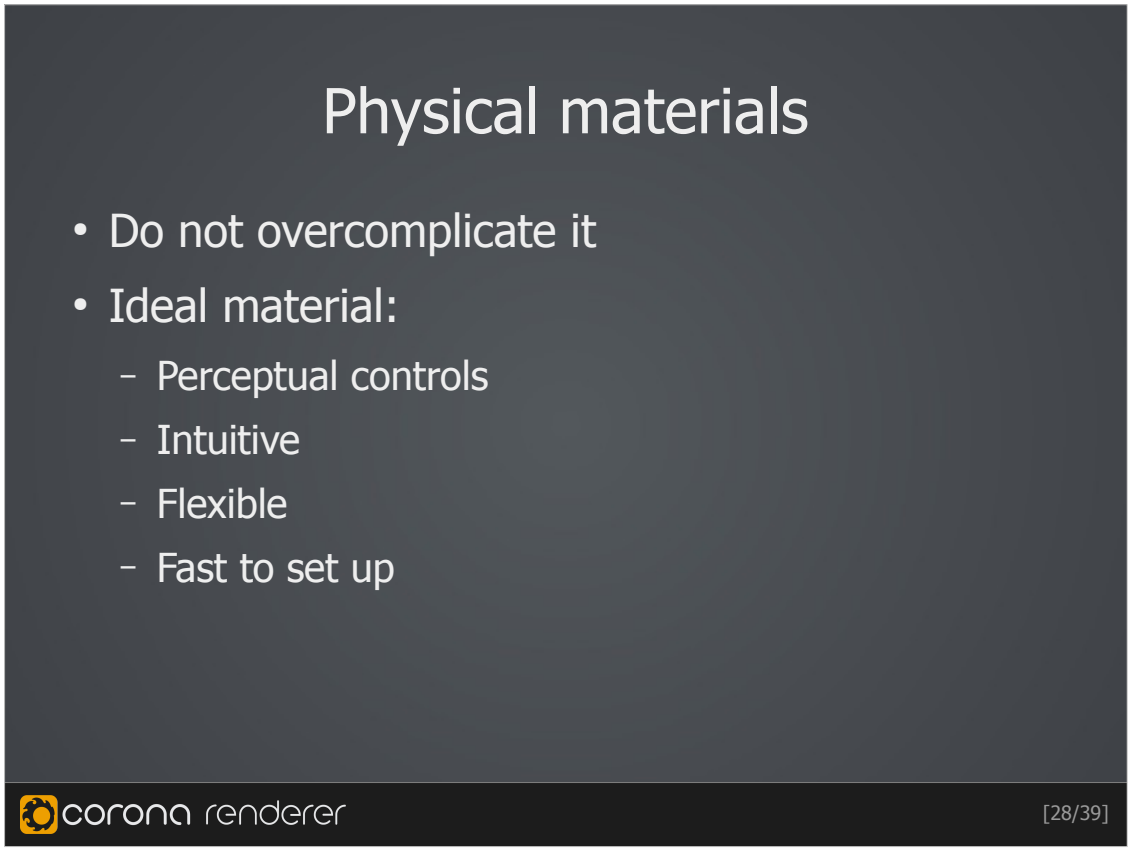
It of course produces bias, but users don't mind the bias; they actually don't even call this biased. As long as there are no splotches or missing shadows, they consider the result unbiased.

Next dilemma is how material controls should look like.



After doing some research amongst users, we noted that there was a common problem of having the inputs too physical.

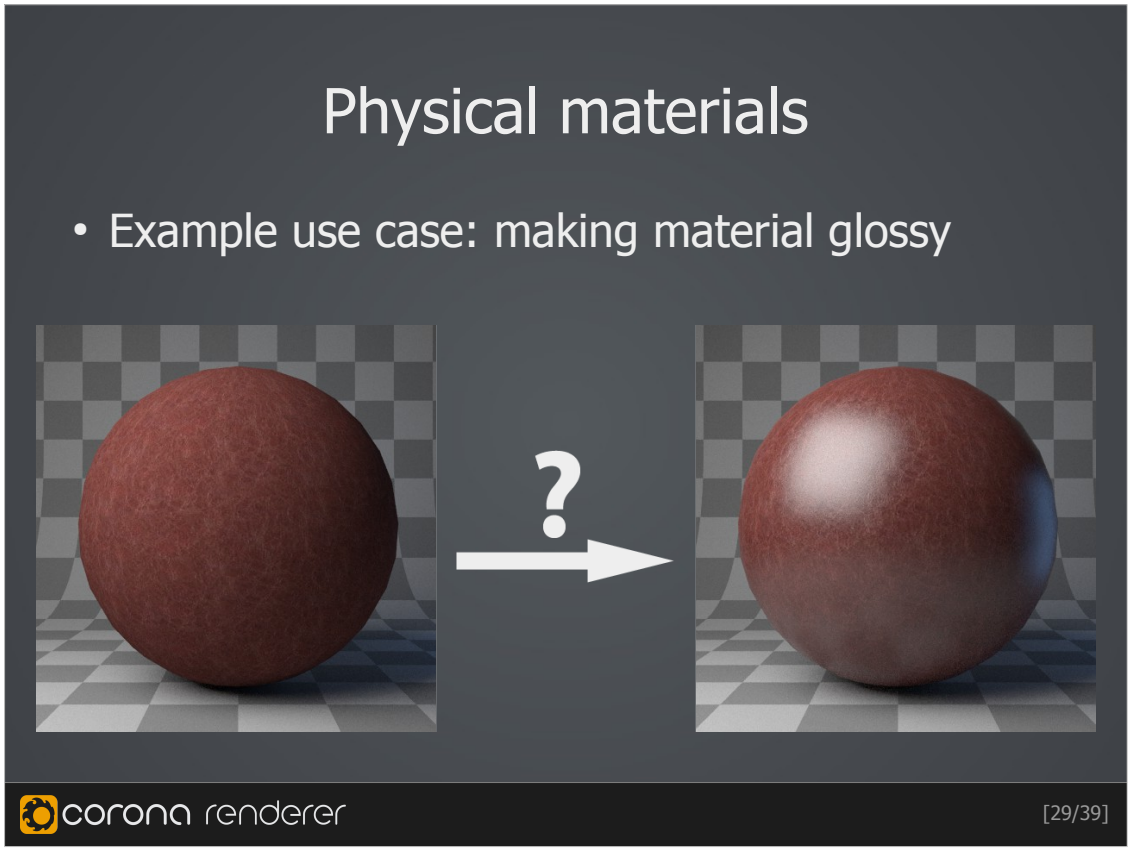
Sadly, Corona users are not physicists. They think about how the material looks: what is its color, roughness, transparency, etc. But researchers think about **why** the material looks how it looks: what is its BRDF, microfacet distribution, IOR, etc.



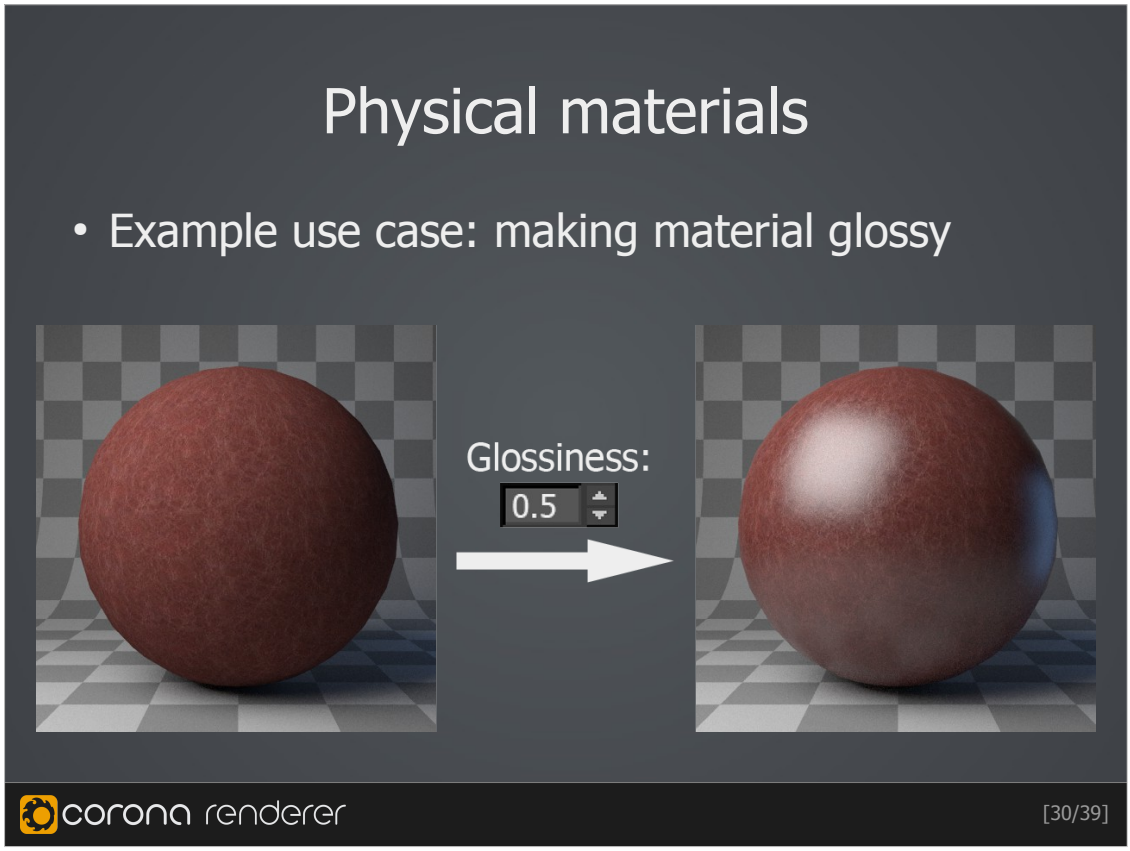
So while there is of course nothing wrong about using physically based BRDFs, their controls should always respect users' point of view.

According to our experience, most users prefer having one “main” universal material for 95% of situations, which has perceptual controls, is simple, flexible, and fast to set up.

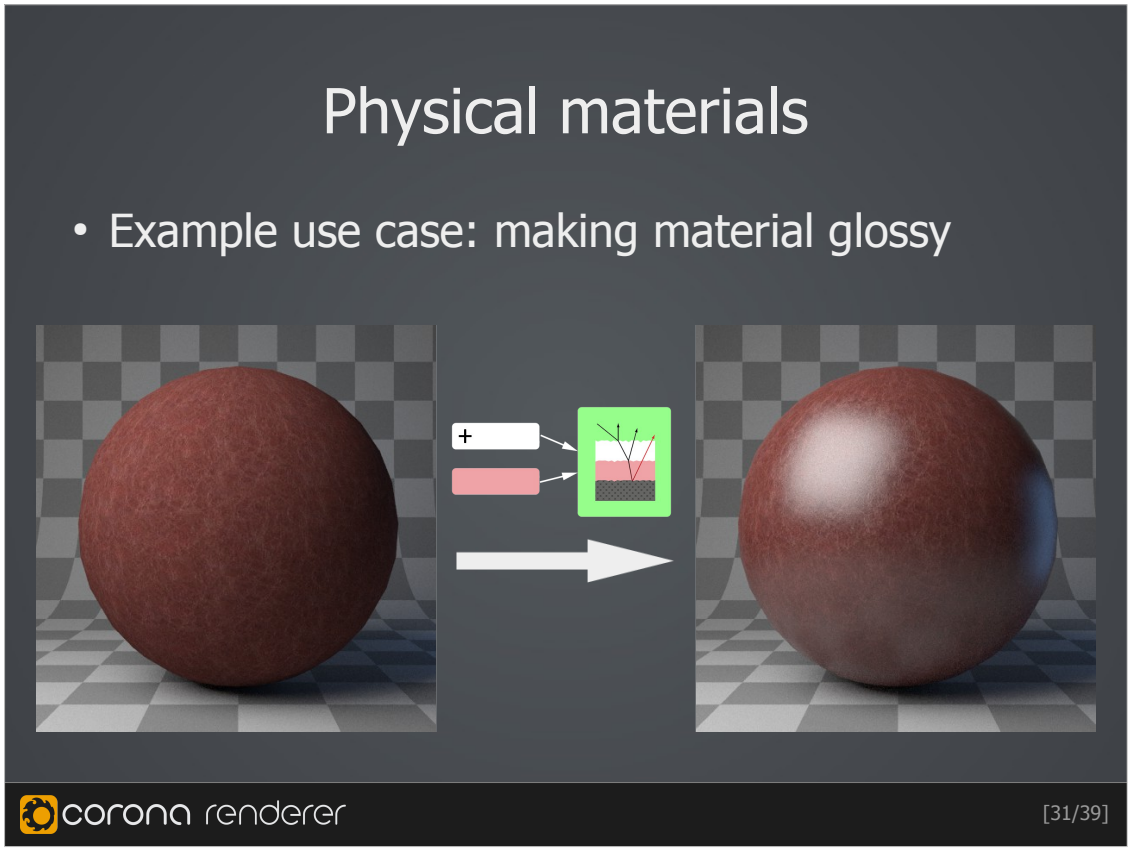
This makes the repetitive task of setting and tweaking basic materials intuitive and fast.



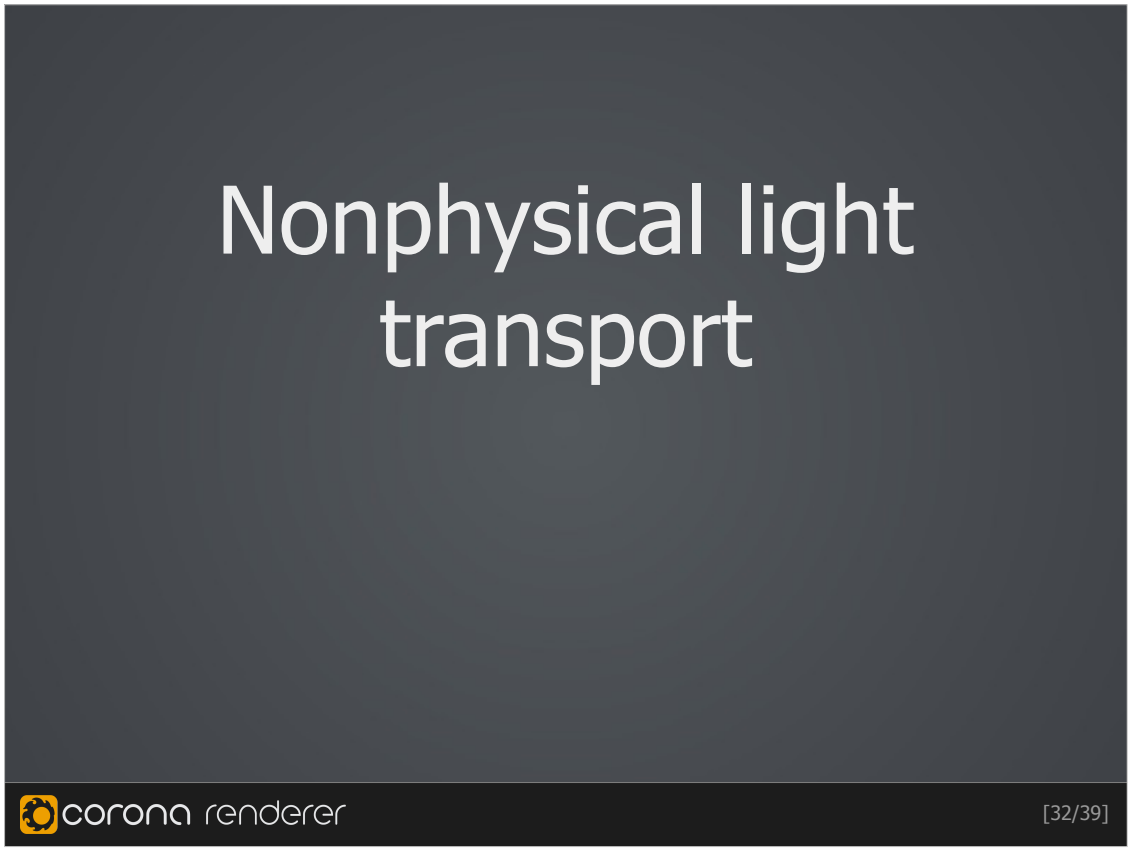
Let's illustrate the difference between intuitive and too technical controls on the use case of making a material slightly glossy. This is a task users may need to do hundred times a day.



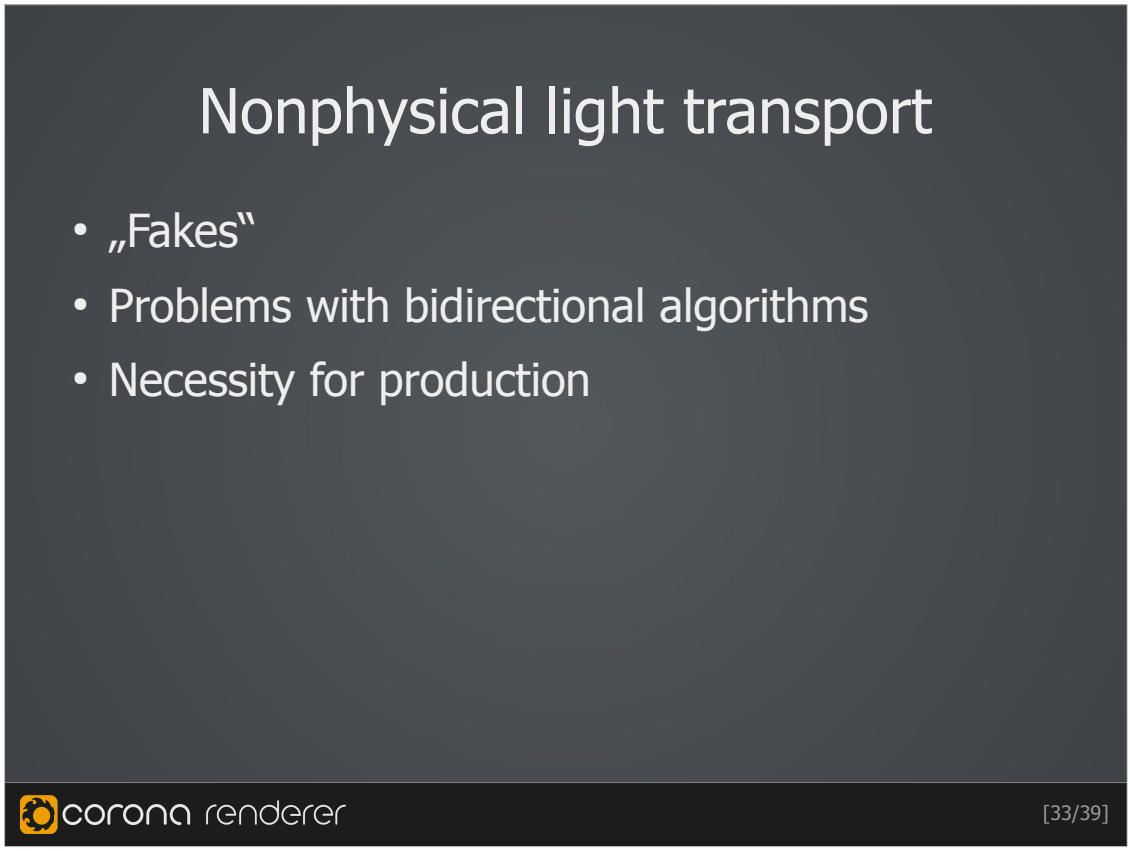
In a good UI, it would be a matter of adjusting one spinner.



In an overcomplicated GUI one would need to add new layer to the material and adjust its properties, which requires too many steps and is unintuitive.



The final issue is implementing nonphysical light transport.



Nonphysical light transport is also called „fakes". Many fakes are notoriously hard to implement with advanced algorithms such as bidirectional path tracing or VCM, but they are absolutely necessary for production. There is not a single successful renderer in archviz that does not support them.

There are two main categories of fakes. First are the ones providing users with artistic control.

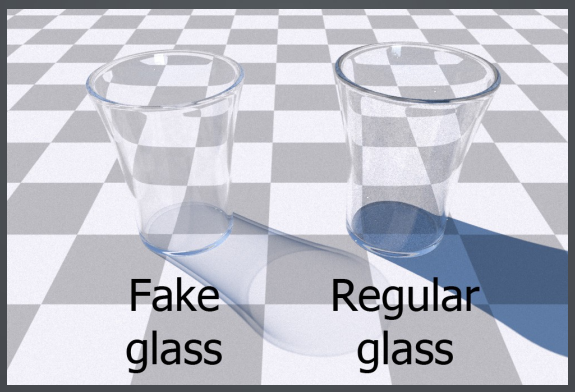
To illustrate why they are necessary, here is an image rendered with VCM. Althought it is physically correct, every artist's first instinct would be to somehow remove the weird „square shadow corner“ (circled), even though it is actually a physically correct reflection.

Just because it is physically correct does not mean it is visually pleasing and desired. This is why features such as disabling shadows, direct visibility, or overriding reflection environment are necessary even in modern physical workflows.

Nonphysical light transport

2) Performance cheats

- Fake glass (thin glass approximation, glass without caustics)

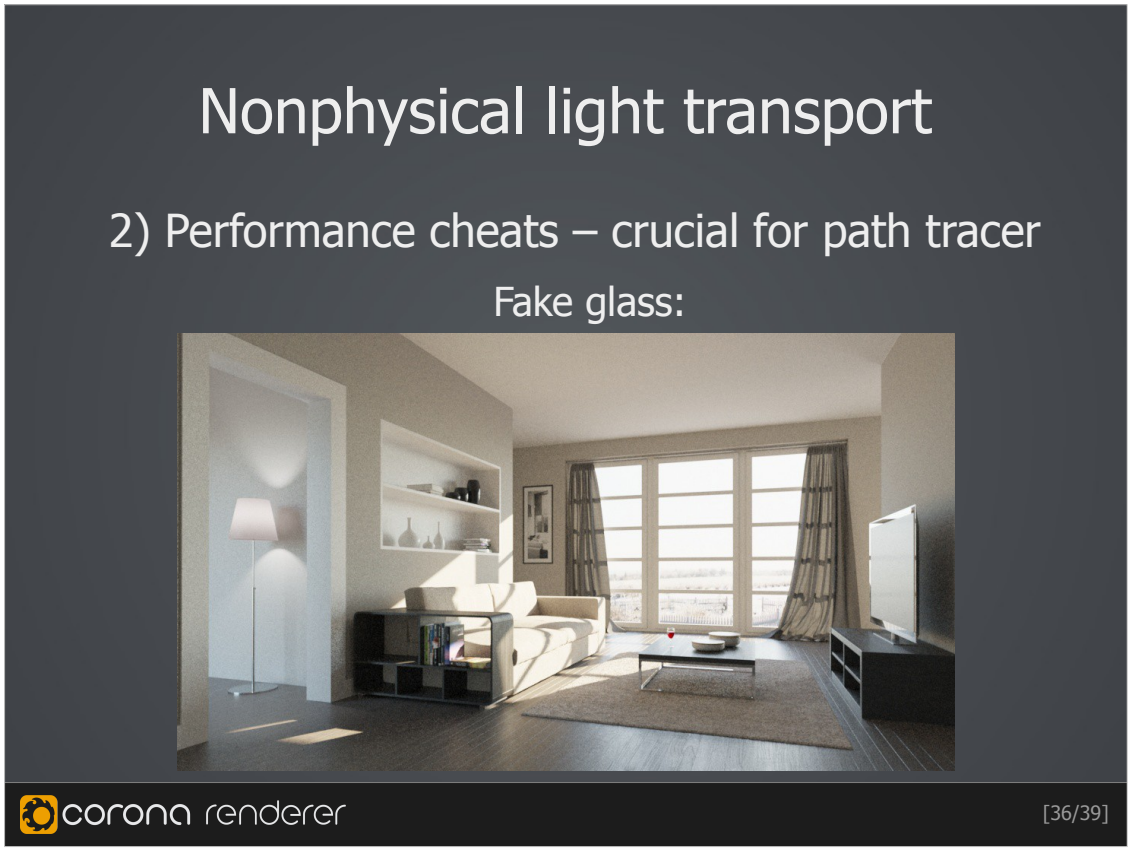


The second category are the fakes that make it possible for the used algorithm to render scenes in reasonable time.

The most prominent example from this category is the fake glass, also called thin glass approximation.

It is a material that acts as a regular glass when viewed directly, but is transparent to indirect rays, meaning it does not block light or create caustics (image on left).

It is used with path tracing, when caustics would have to be clamped otherwise (image on right).



This is what powers the vicious circle of path tracing shown before, as it can make the light transport simple in most scenes. It is typically used in windows.

Without it this scene would be impossible to render just with path tracing.

Nonphysical light transport

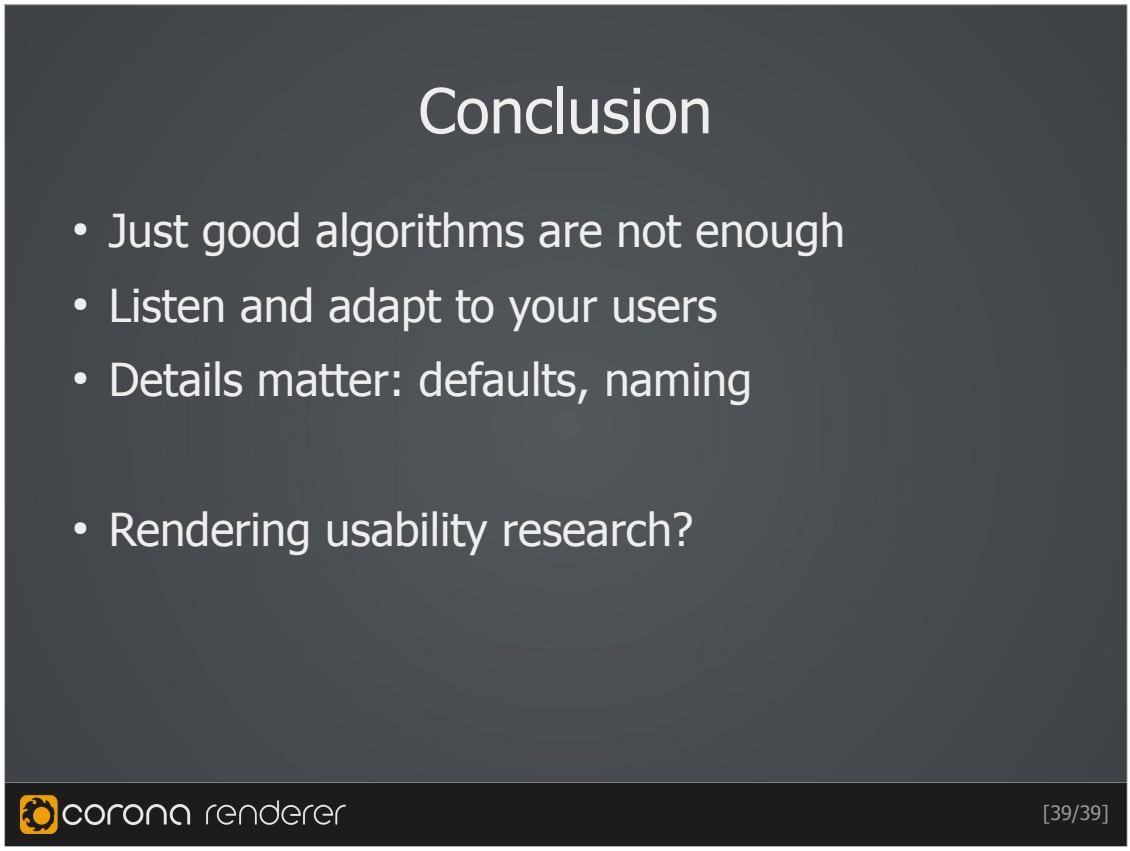
2) Performance cheats – crucial for path tracer

True glass: sun removed by MSI clamping

This is the result with true glass for comparison. The sun illumination has to be clamped away.

There is a fundamental difference between the two categories: artists use the artistic control fakes because they want to, and the performance fakes because they have to. So while the first category improves the usability, being forced to use performance fakes decreases usability.

As a result, while we have implemented many artistic control fakes, we always thoroughly search for a better solution than resorting to a performance fake, and we have refused to implement many traditional performance fakes, such as different BRDFs for direct and global illumination.



The take-home message is that just implementing newest papers is not the whole story of making usable software. Fine tuning the details is also necessary.

Even small things like default values or naming of features can make a huge difference. Even the best GI algorithm is useless if the user does not know how to set it up.

There is also a lot of open problems in the usability of rendering, for example on the already mentioned problem of making physically based materials easier to use.