Implementing one-click caustics in Corona Renderer

Martin Šik, Chaos Czech a. s.

Before I start my presentation about Caustics rendering in Corona, let me introduce myself.
I am a senior engineer of Corona and I am mostly responsible for handling light transport related issues, including the caustics.
Beside that I am also an academic researcher and I have recently finished my PhD on topic related to efficient rendering of difficult light transport such as caustics.

Motivation

- Caustics = light focused through refraction/reflection

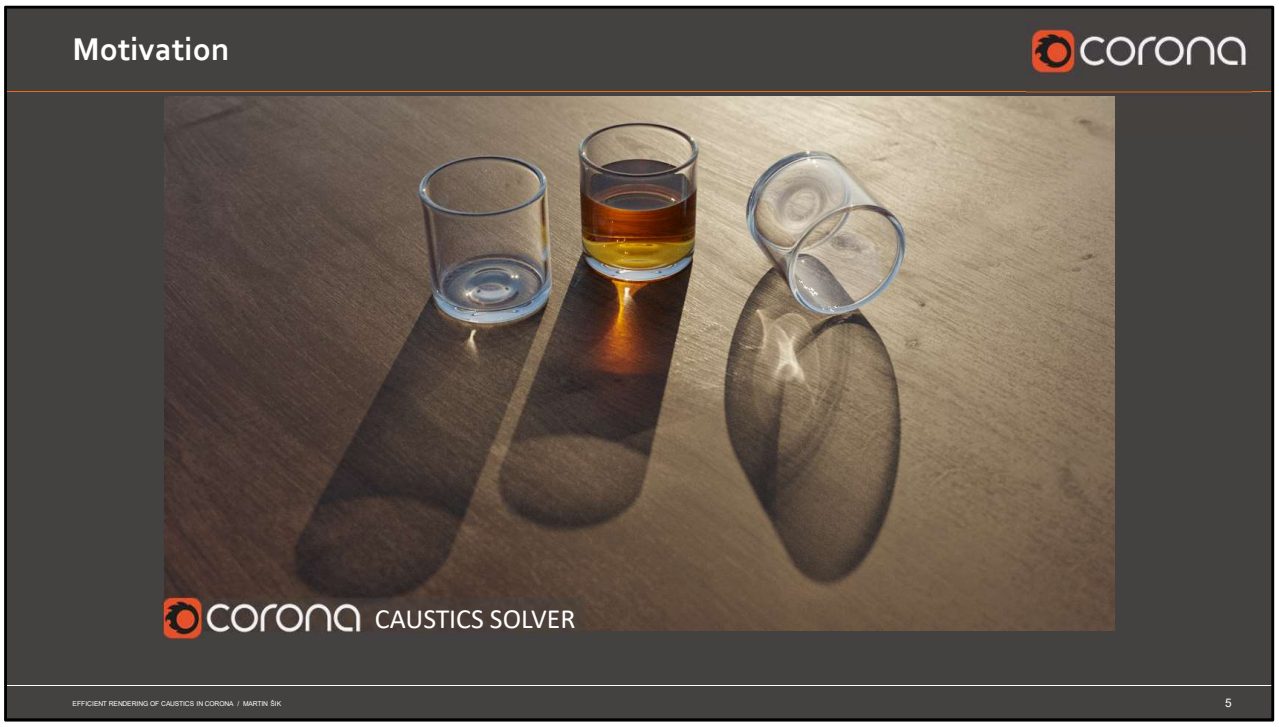
Let me start my presentation with a simple question. What actually are caustics? Basically every light that is focused by highly glossy reflection or refraction is a caustic.
Commonly people recognize caustics created by jar with a water (CLICK), caustics at the bottom of the sea floor (CLICK) or some small caustics caused by highly glossy materials (CLICK). PAUSE

Motivation

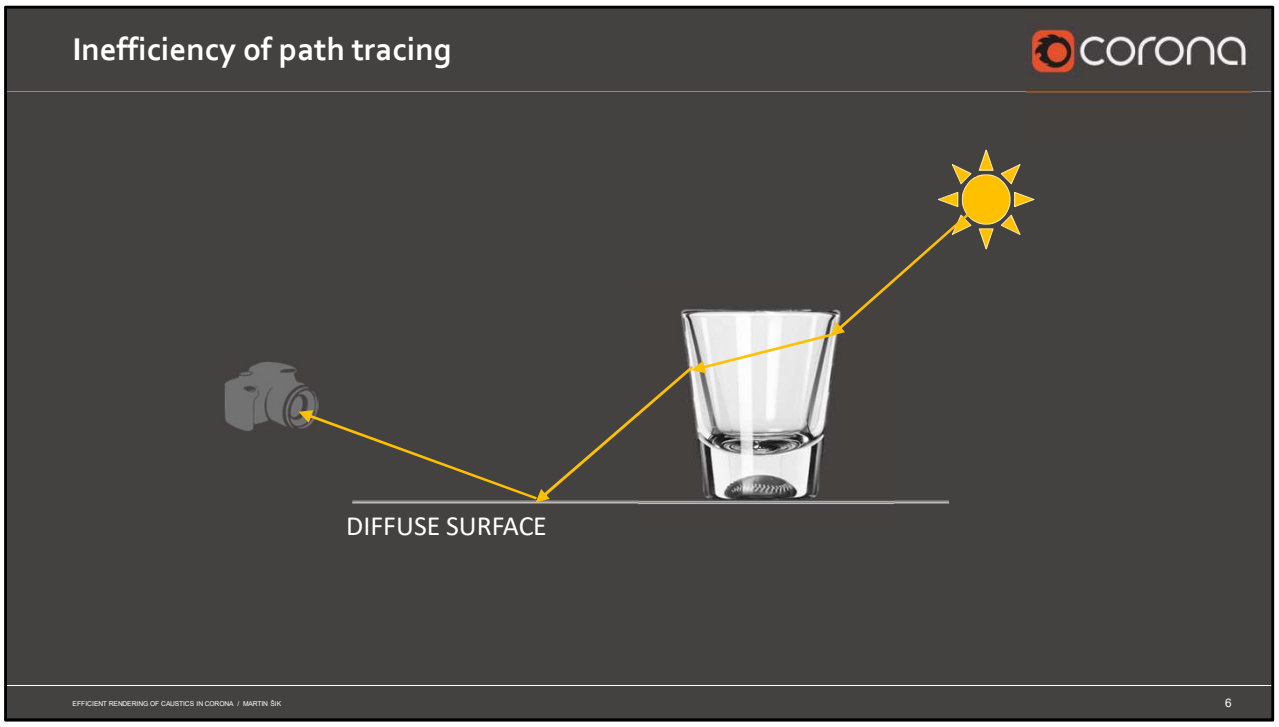
What people usually don’t realize is that all light in interior shots can also be considered a caustic or a reflection of a caustics created by the light focused by the window panes.
So we are surrounded by caustics more often that we realize.

Motivation

Unfortunately, the majority of the current rendering software including the previous versions of Corona can not efficiently render caustics and thus they are often disabled or faked by default.

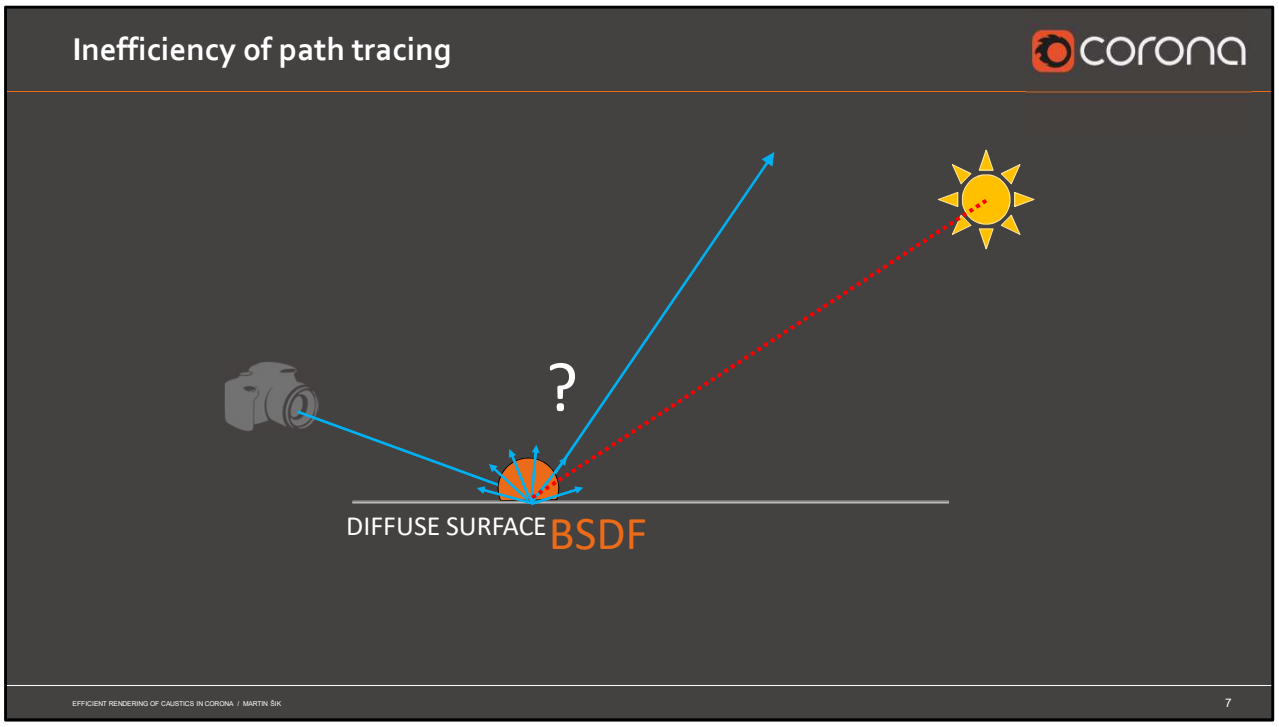


We have therefore set us a task to create a new caustic solver for the Corona version 4 that could efficiently render caustics leading to more visually pleasing and realistic images.

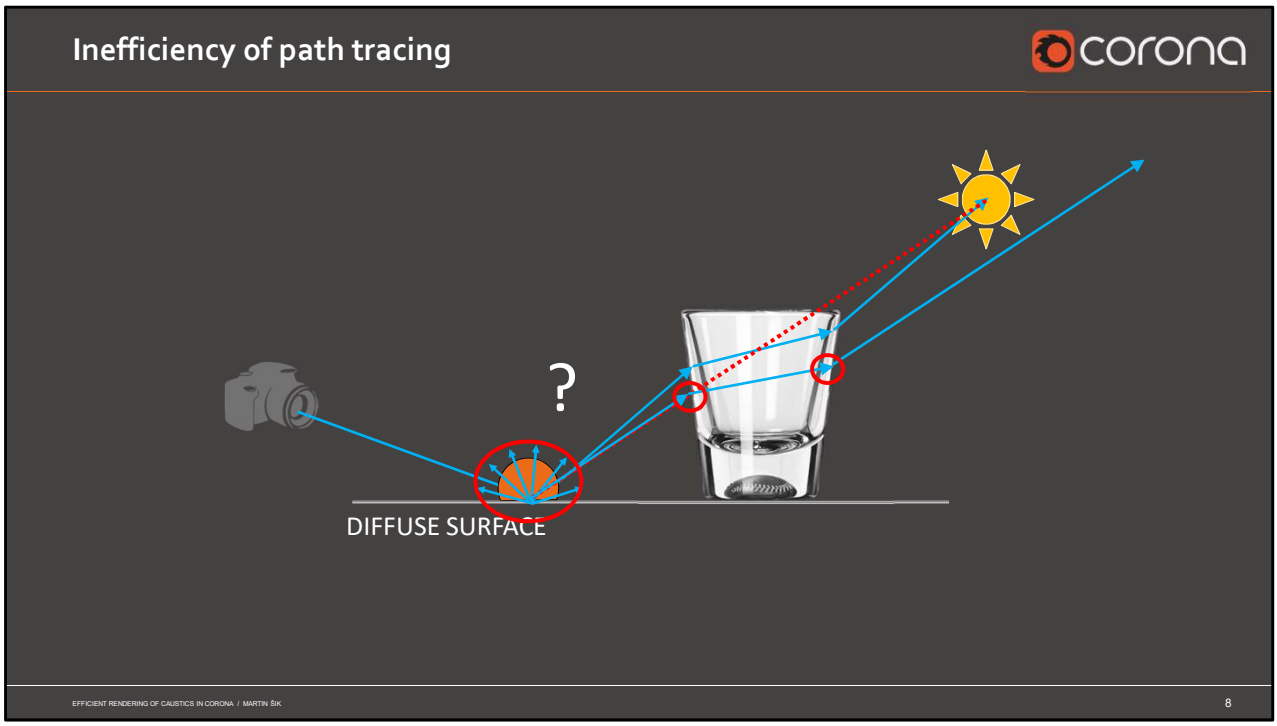


Before I discuss how we approach this goal, let us first look at why are caustics difficult to render for the current renderers.

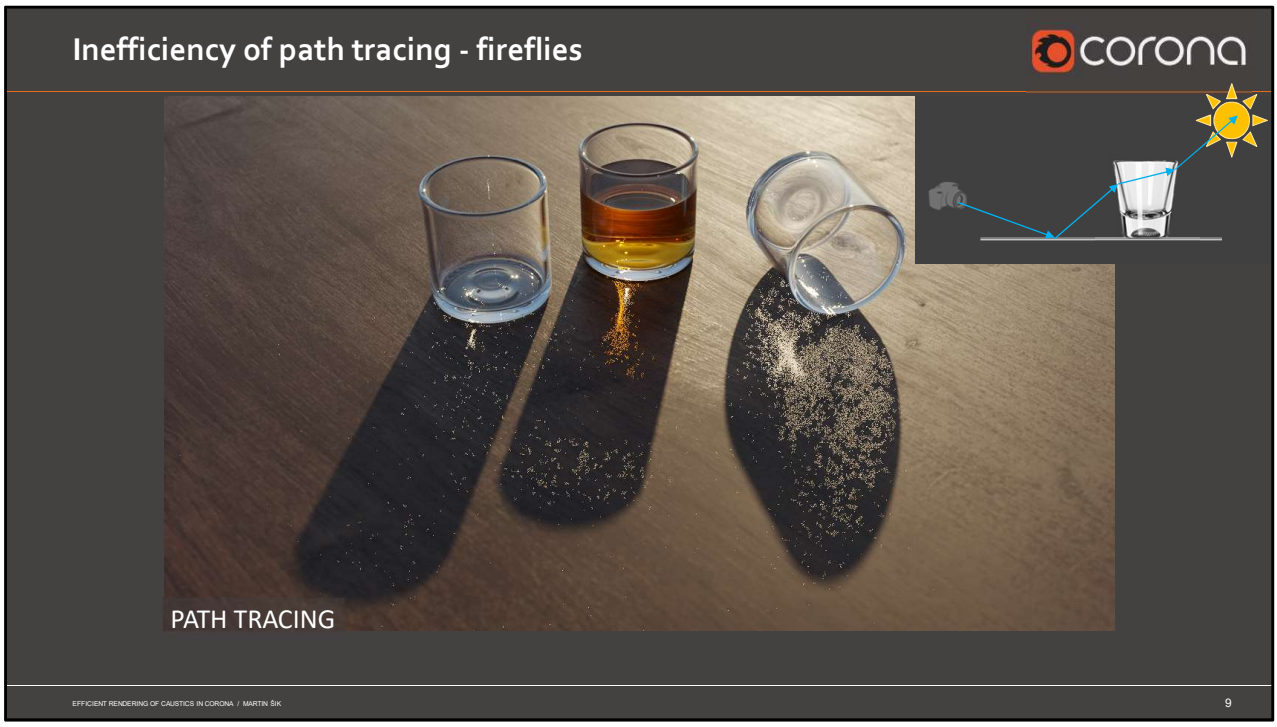
Note that a caustic is light that has been focused via reflection or refraction and which then hits a diffuse or glossy surface before arriving at the camera sensor.



Majority of the modern rendering software use path tracing as their solver of global illumination.
Path tracing starts the path at the camera and then traces a ray towards a surface, upon hitting the surface it has to randomly decide which direction it should take next.
(CLICK)
This decision is usually done based on the bidirectional scattering distribution function (or just BSDF) which defines the reflection profile of the surface material.
In the case of diffuse material, the path has a same chance of continuing in any direction
(CLICK)
and thus it has a very low chance of hitting anything important, like the light source.
(CLICK)
Fortunately, in this case instead of random decision, we can directly connect light source with the surface hit.



Unfortunately, in case of caustics such connection is usually blocked by reflective or refractive geometry.
(CLICK)
If we continue the path in direction based on the connection, such path would very likely miss the light source due to the law of refraction.
(CLICK)
Note that once we have decided the direction at this intersection (CLICK), we can not change it at the intersections with the glass.
(CLICK)
And thus in the usual path tracer, we are left with a random selection of a new direction at the first intersection that will with low probability lead to a path that hits the light source and creates a caustic.



Such a low probability of hitting the light source together with high contribution of such a caustic path will exhibit as fireflies in the place where we would expect caustics.

Inefficiency of path tracing - fireflies

The chance of hitting the light source is of course further reduced with an increasingly smaller light source.

Inefficiency of path tracing - fireflies

Which leads into even slower convergence.

Inefficiency of path tracing

- Current renderers "can" render such scenes without fireflies!
- They fake it!!!

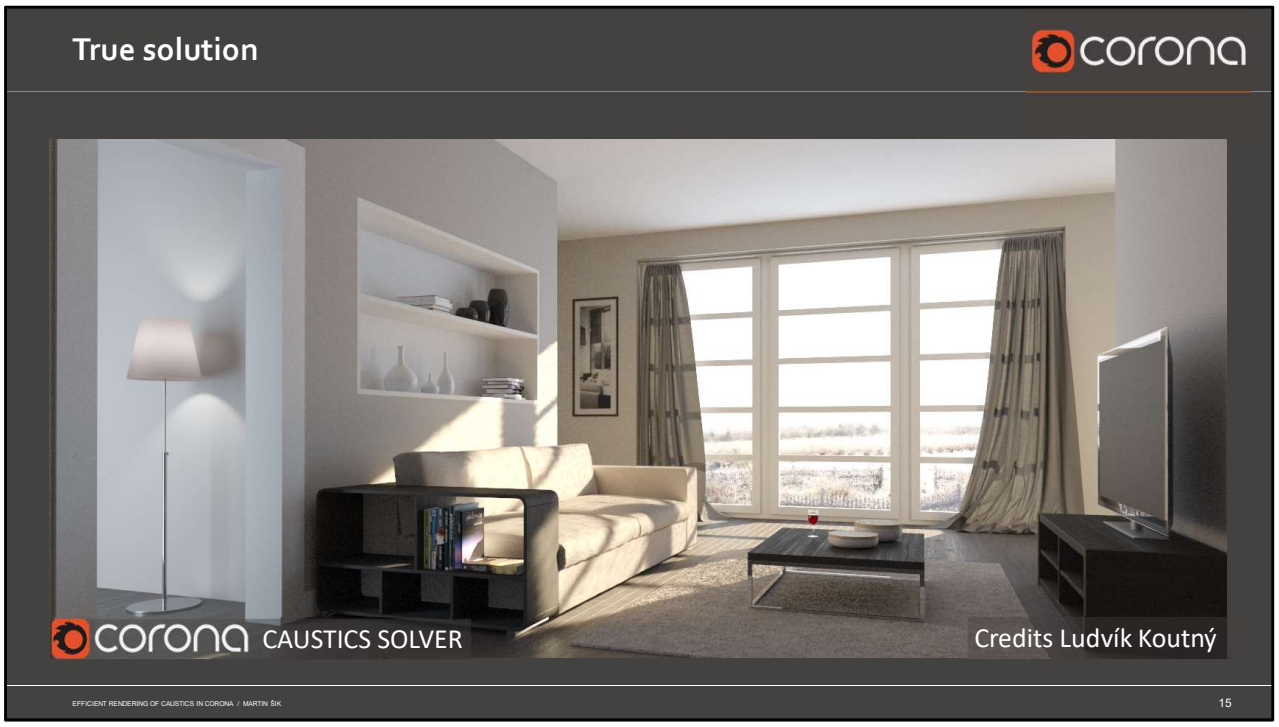
So how is it that current renderers can render scenes without these fireflies?!
(CLICK)
Well, they fake the true physical illumination!

Inefficiency of path tracing - clamping

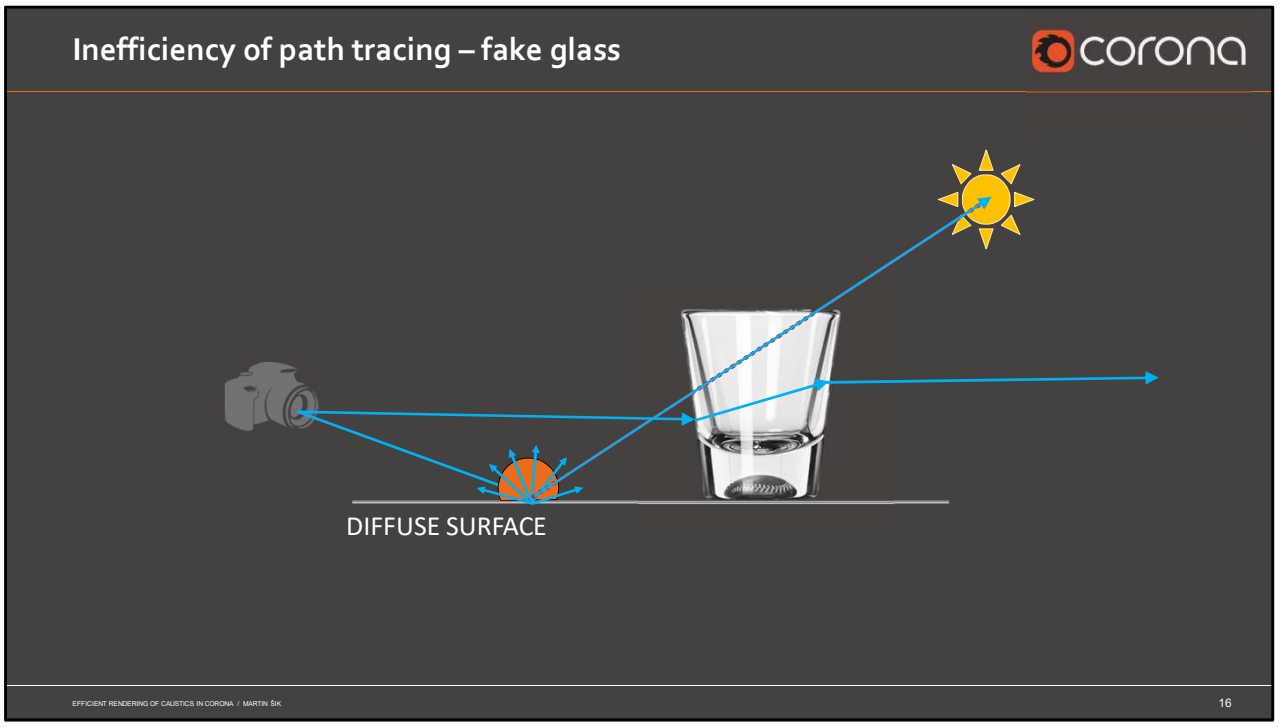
One common solution is to clamp these fireflies, which leads to unnatural dark shadows and to overall missing illumination.

Inefficiency of path tracing - clamping

This is especially apparent in the interior shots, where as we have mentioned before, all the illumination comes from caustics created by the windows.



Here is the true solution for comparison.

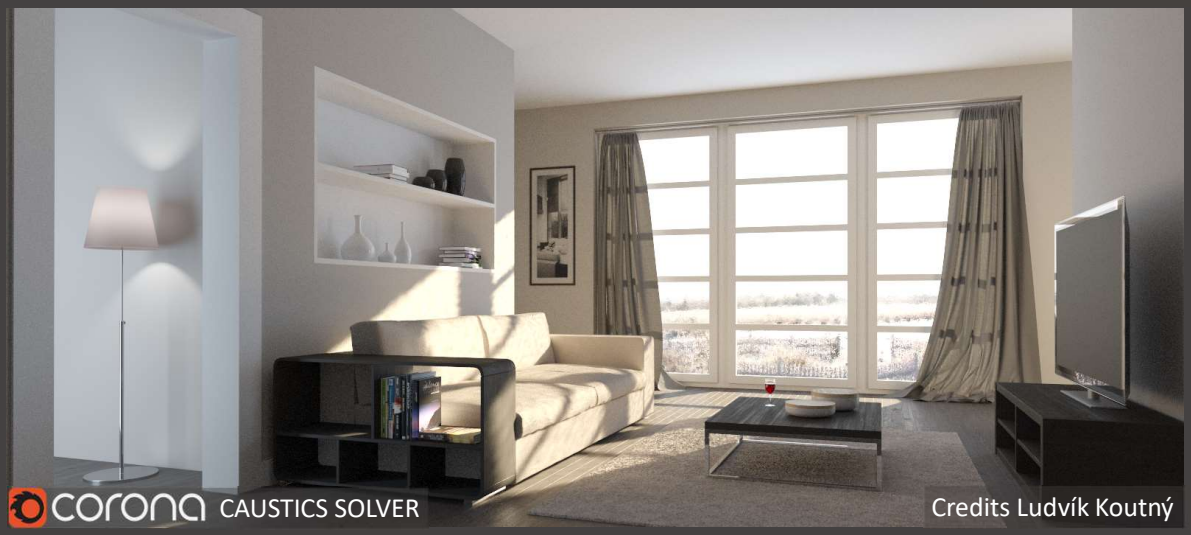


To better handle such interior shots, people have develop another fake solution, so called fake glass.
This fake glass allows the caustics paths to transmit through glass without refraction.
(CLICK)
Thus we can directly connect surface hit with the light source.
(CLICK)
On the other hand when the fake glass is directly viewed from the camera, it works physically correctly.

Inefficiency of path tracing – fake glass

And as you can see the fake glass solution can give quite convincing result in some scenes.
Which allowed the developers of renderers to avoid handling of caustics for very long time.

True solution



However even in this scene, the true solution is quite different.

Inefficiency of path tracing – fake glass

The unrealistic appearance of fake glass is especially visible in more detailed shots, where it leads to unnaturally light shadows.

True solution

… compared to the true solution.

Inefficiency of path tracing – fake glass

Credits:
Giona Andreani
gionacg.wordpress.com
Vitruvio,
Designed by Atelier Oï for Artemide

PATH TRACING + FAKE GLASS

Here is another example with the fake glass used on a lamp.

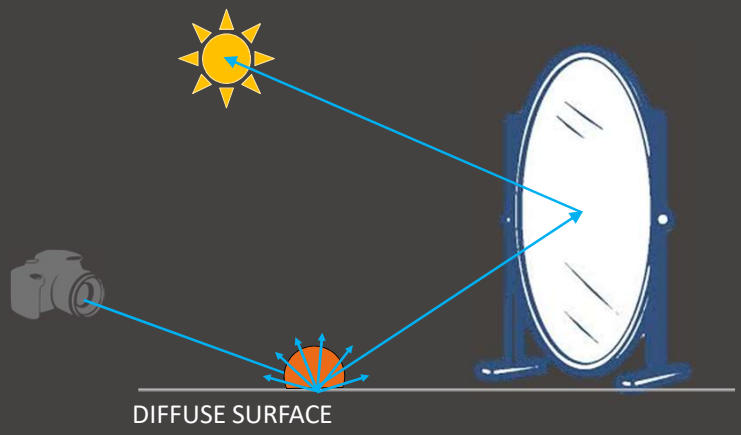
True solution

Credits:
Giona Andreani
gionacg.wordpress.com
Vitruvio,
Designed by Atelier Oï for Artemide

And here is the true solution.

Inefficiency of path tracing – fake glass

- No "fake mirror " solution!



Furthermore, the fake glass will not help in cases where caustics are created by reflection (rather than refraction)
(CLICK)
There is simply no fake mirror solution.

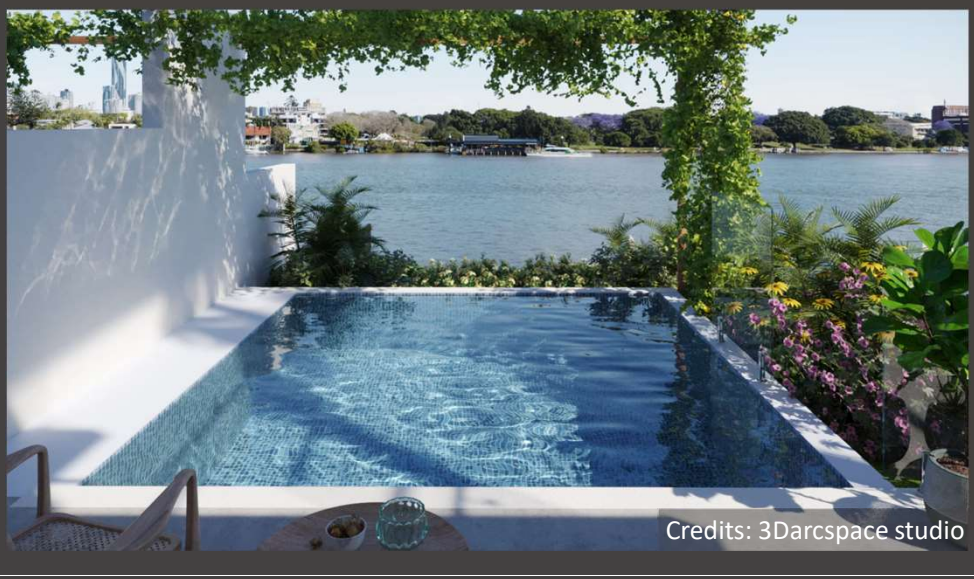
Inefficiency of path tracing – fake glass

We demonstrate this on this scene. When we compare to the true solution,

True solution

we can see that it contains many reflective caustics that cannot be faked.

Animation comparison



And here is animated example, which shows that using our caustics solver improves the realism of the scene.

Goal

- Fast caustics solver
- Minimum number of parameters
- Minimize overhead compared to path tracing
- Work as before outside of caustics

So to render these scenes with caustics and without fakes, we have set ourselves a goal to create a new caustics solver for Corona v4 that would be able to efficiently resolve any type of caustic.

Furthermore, we wanted this solver to have minimum number of parameters, ideally we wanted the users to just use a single checkbox that switches the solver on.

While we knew this solver will definitely lead to some overhead compared to ordinary path tracing, our goal was to minimize this overhead.

Finally, it was necessary for the solver to behave as when switched off in parts of the scenes that contain no caustics.

5:10

Academic solution for caustics

- Bidirectional methods
- Metropolis light transport
- Path guiding
- Manifold next event estimation

Of course we are not the first to tackle this issue, academic researchers have been addressing the issue of caustics and related difficult light transport since the nineties.
And thus we can take advantage of their work.
To this day there exists at least 4 categories of different solutions to this problem.
Let us quickly discuss them, so we can determine which one is the best for our new caustic solver.

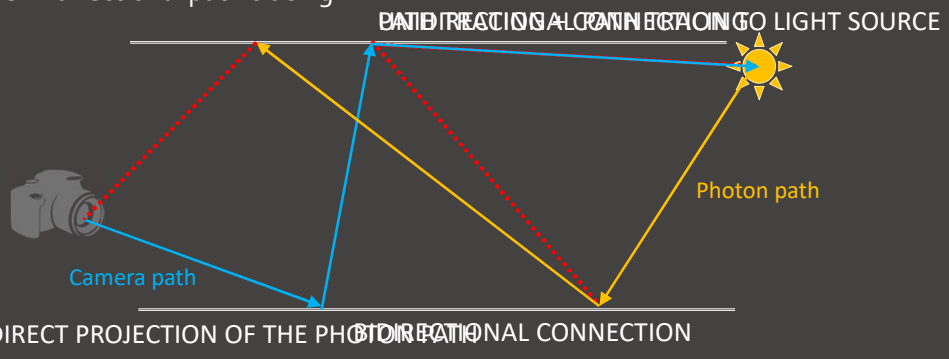
Academic solution for caustics

- Bidirectional methods
- Metropolis light transport
- Path guiding
- Manifold next event estimation

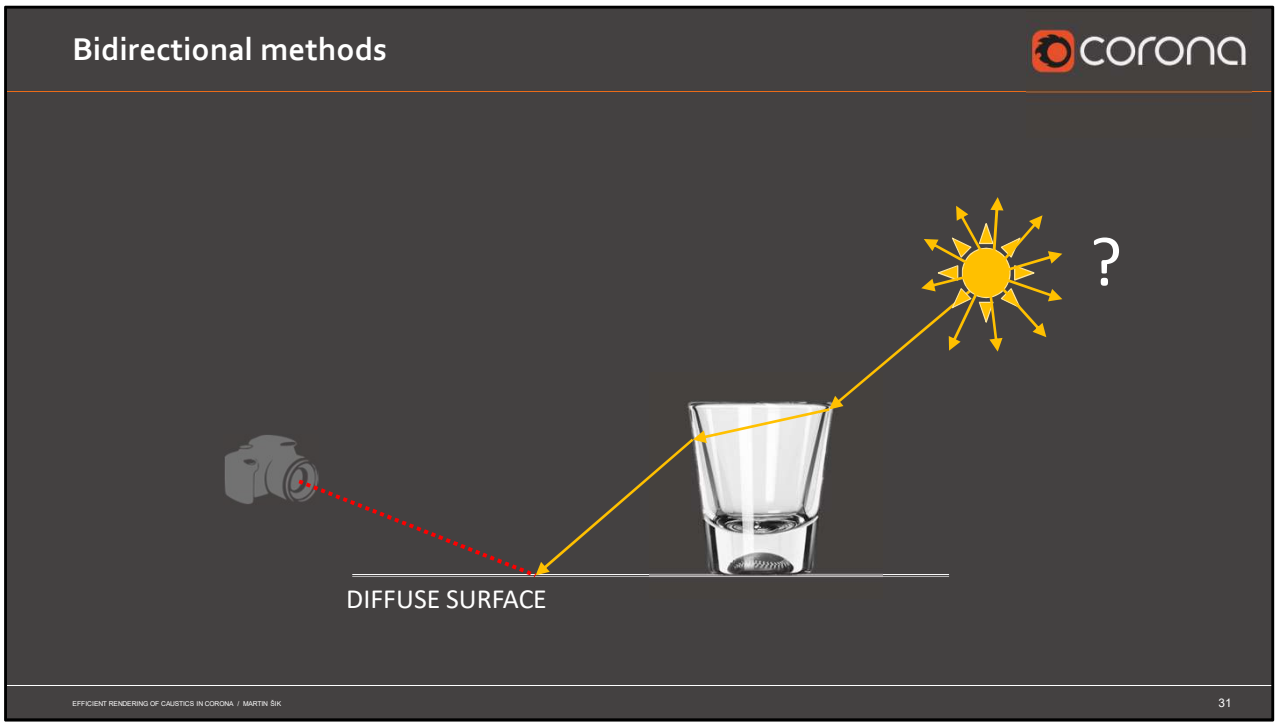
The first category are the bidirectional methods.

Bidirectional methods

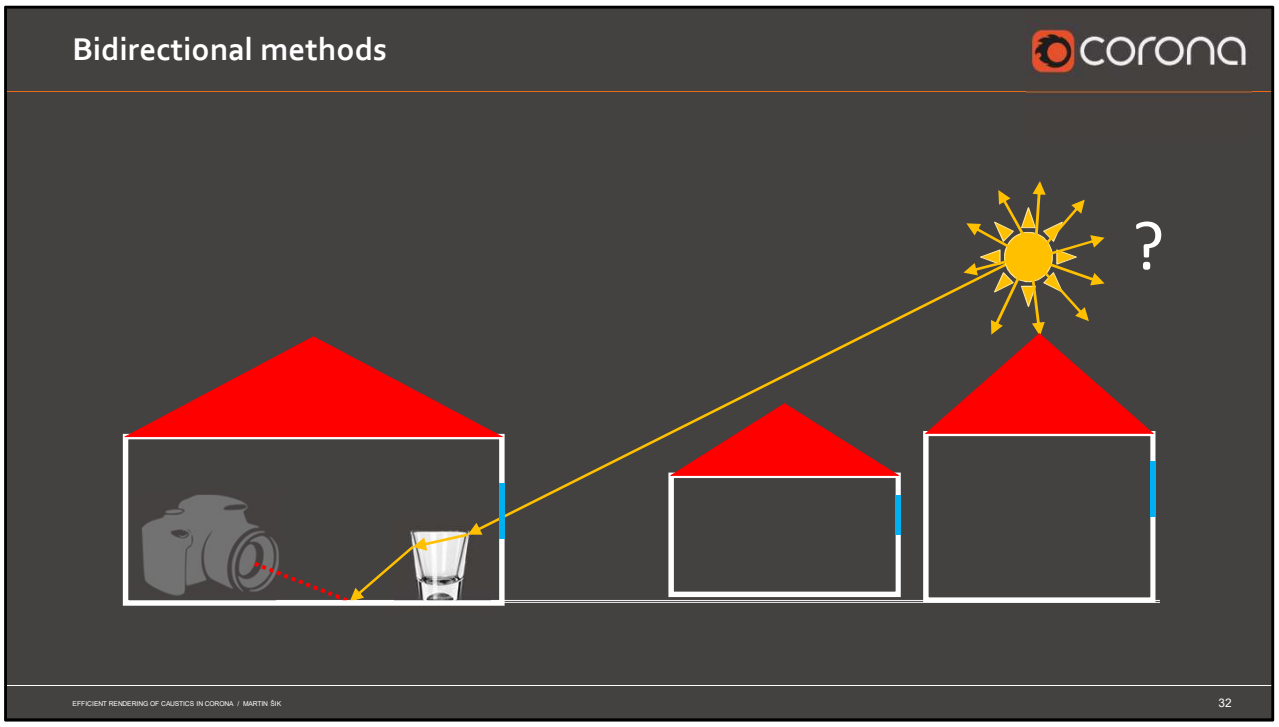
- [Lafortune and Willems 1993], [Veach and Guibas 1994] [Jensen 1996], [Georgiev et al. 2012], [Hachisuka et al. 2012], etc.
- Example: Bidirectional path tracing



These methods not only trace paths from the camera (CLICK) so called camera paths.
(CLICK)
But they also trace paths from the light sources, so called photon paths.
Various bidirectional methods then utilize these camera and photon paths in different ways.
(CLICK)
For example, the famous bidirectional path tracer utilizes the paths as follows: (CLICK)
It uses unidirectional path tracing technique that randomly hits the light source. (CLICK)
Path tracing technique that connects camera path to the light source.
(CLICK)
Bidirectional connection between camera subpath and photon subpath.
(CLICK)
And direct projection of the photon path to the camera.



Caustics can then be easily handled by some of these techniques, for example by the direct projection of the photon path to the camera.
(PAUSE)
Unfortunately bidirectional methods have their issues.
(CLICK)
Probably the most important one is determining to where should we trace photon paths for most effective rendering.



This is especially the case in larger scenes, where it is difficult to guide the photon paths toward the region visible by the camera.
This is also the reason why people mostly use path tracing, which traces paths from the camera and thus does not suffer from this issue.

Bidirectional methods - issues

For example in this scene the light has to come from the outside through the window and then randomly bounce on a glossy surface before reaching the visible kitchen counter.

Bidirectional methods - issues

Vertex connection and merging [Georgiev/Hachisuka et al. 2012]

After an hour of rendering, we can see that state-of-the-art bidirectional algorithm converges only very slowly in such a scene.

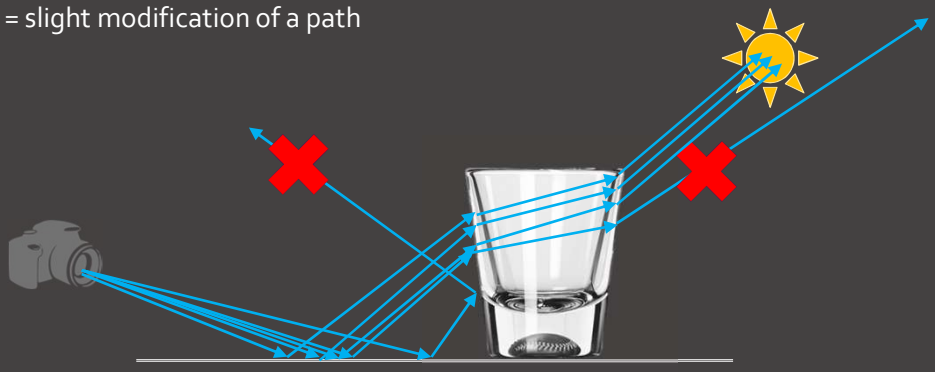
Academic solution for caustics

- Bidirectional methods
- Metropolis light transport
- Path guiding
- Manifold next event estimation

So let's try another solution, Metropolis light transport.

Metropolis light transport

- [Veach and Guibas 1997], [Kelemen et al. 2002], [Jakob and Marschner 2012], [Hachisuka et al. 2014], [Kaplanyan et al. 2014], etc.
- Mutation = slight modification of a path



This algorithm utilizes so called Metropolis sampling
(CLICK)
It randomly searches for a contributing path (CLICK) and once it finds it (CLICK), it will slightly modify this path or mutate the path in the Metropolis jargon (CLICK)
to create several different contributing paths. This way the algorithm can create many more caustics paths.

Metropolis light transport - issues

Unfortunately, using Metropolis sampling can lead to appearance of artifacts in the image.
This is especially visible in animations, where the artifacts exhibit as flickering.

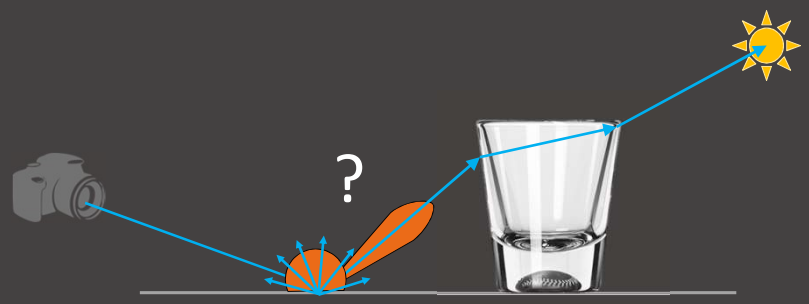
Academic solution for caustics

- Bidirectional methods
- Metropolis light transport
- Path guiding
- Manifold next event estimation

So let’s move to the next solution, which is path guiding.

Path guiding

- [Jensen 1995], [Vorba et al. 2014], [Dahm and Keller 2017], [Müller et al 2017], etc.
- Learn sampling distributions from previous samples



Few years back I and cofounders of Corona Jaroslav and Ondra have collaborated with other researches on a method that renewed interested in path guiding.
(CLICK)
Path guiding applies machine learning or AI to learn from the previous samples where to trace the next samples.
(CLICK)
Instead of randomly selecting next direction, (CLICK) the guided path tracer utilizes the learned distribution
(CLICK)
that will generate more samples in the important direction.

Path guiding - issues

Unfortunately, the learning of the distributions is often slow and imprecise and thus path guiding will often fail to find all the caustics

Path guiding - issues

unlike bidirectional methods..

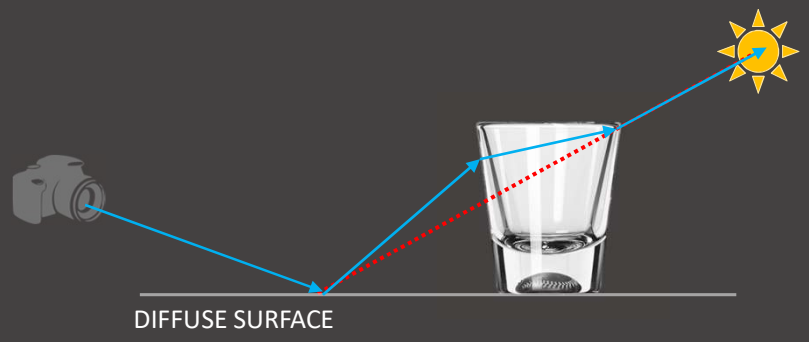
Academic solution for caustics

- Bidirectional methods
- Metropolis light transport
- Path guiding
- Manifold next event estimation

The last solution is manifold next event estimation.

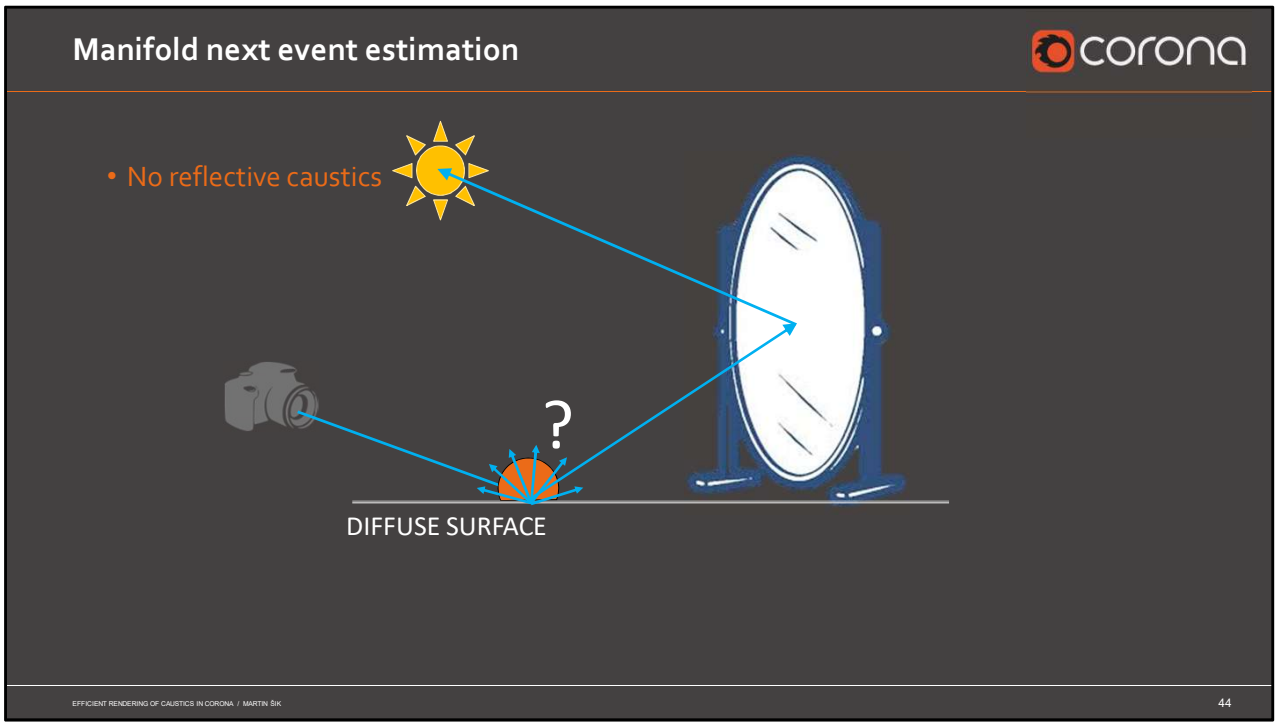
Manifold next event estimation

- Hanika et al [2015]
- Iterative search for the correct refractive path



This method starts from the fake glass solution (CLICK) and uses differential geometry to compute the accurate solution (CLICK)

Manifold next event estimation



Unfortunately, this method will not help in cases where caustics are created by reflection (rather than refraction).

Academic solution for caustics

- Bidirectional methods (very slow convergence in many scenes)
- Metropolis light transport (correlation artifacts)
- Path guiding (missing caustics)
- Manifold next event estimation (only refractive caustics)

So we can see that none of these solutions are without issues.

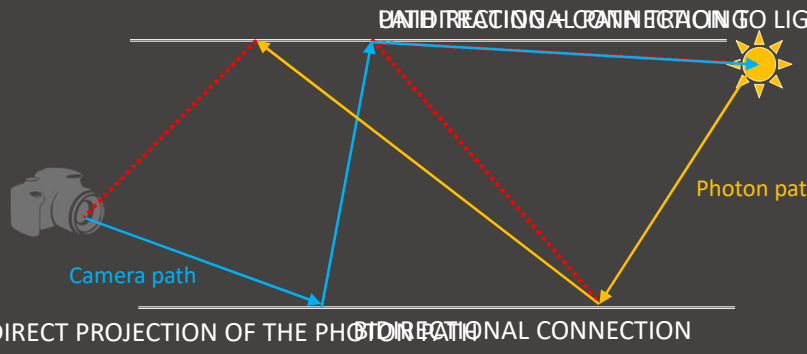
Academic solution for caustics

- Bidirectional methods (very slow convergence in many scenes)
- Metropolis light transport (correlation artifacts)
- Path guiding (missing caustics)
- Manifold next event estimation (only refractive caustics)

However, the bidirectional methods have the most easily fixable issues from these methods and therefore it make sense to base our caustics solver on them.

Vertex connection and merging (VCM)

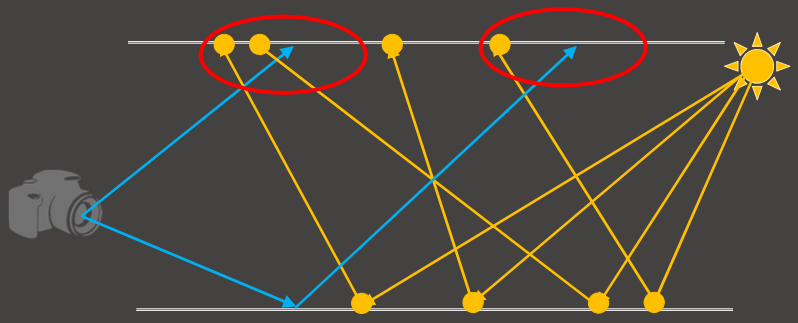
- Vertex Connection and Merging (VCM) [Georgiev/Hachisuka et al. 2012]
- Vertex connection = Bidirectional path tracer [Lafortune and Willems 1993], [Veach and Guibas 1994]



More specifically, we base our implementation on one of the most robust bidirectional methods called Vertex connection and merging or just VCM, which was codevelop by Corona's cofounder Jaroslav.
(CLICK)
The vertex connection in the method's name represents that VCM uses all the techniques from bidirectional path tracer: (CLICK)
unidirectional path tracing (CLICK)
path tracing with connection (CLICK)
bidirectional connections (CLICK)
and direct projection of the photon path

Vertex connection and merging (VCM)

- Vertex Connection and Merging (VCM) [Georgiev/Hachisuka et al. 2012]
- Vertex merging = photon lookup ← Photon mapping [Jensen 1996]



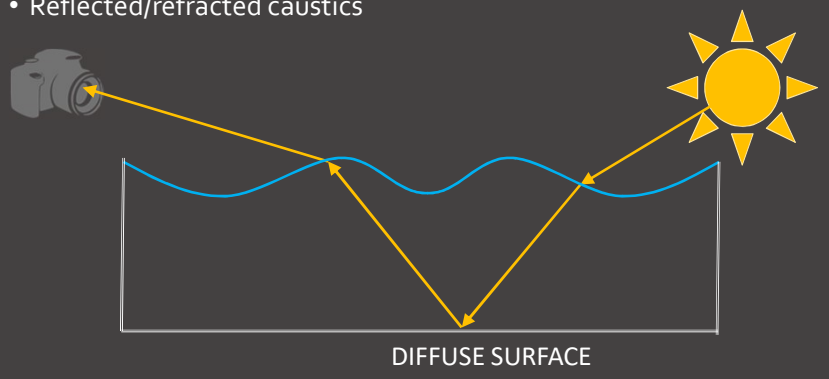
The vertex merging than means that VCM also uses so called photon lookups.
(CLICK)
Photon lookup technique comes from the famous photon mapping algorithm.
(CLICK)
The algorithm first traces all photon paths
(CLICK)
And stores their vertices also called photons.
(CLICK)
Then it traces a path from the camera and (CLICK) performs the photon lookup at each intersection – meaning it will find all photons inside the lookup radius.
(CLICK)
The camera path will then carry energy depending on how many photons fall into the lookup radius.

Indirectly seen caustics

Note that the photon lookup technique is important, otherwise the algorithm would not be able to efficiently handle the indirectly seen caustics, such as the caustics on the bottom of this pool.

Indirectly seen caustics

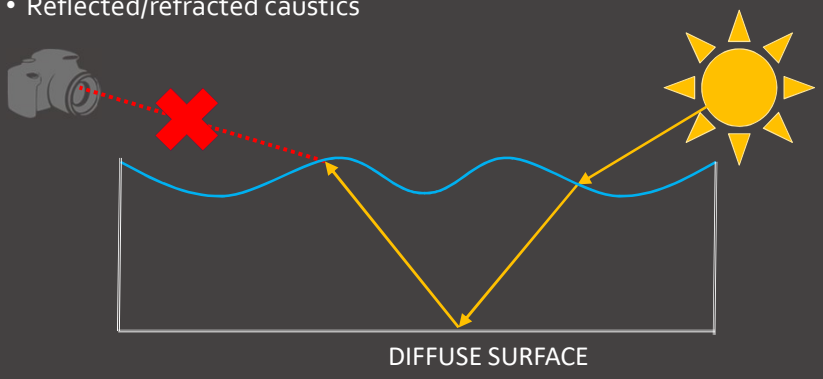
- Reflected/refracted caustics



In this case the caustic, is seen through refraction.

Indirectly seen caustics

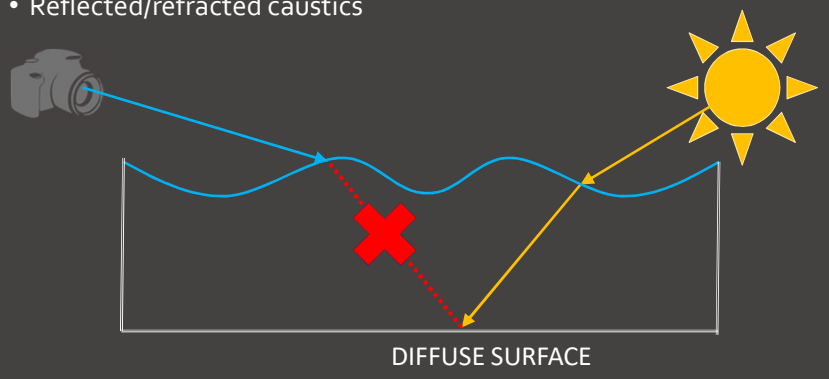
- Reflected/refracted caustics



Unfortunately, it is impossible to connect to path that ends on a specular surface – such as the water surface.

Indirectly seen caustics

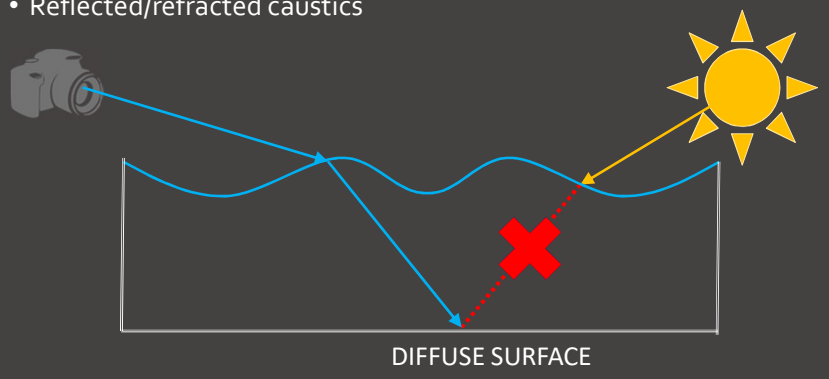
- Reflected/refracted caustics



And thus in this case no connection is possible here.

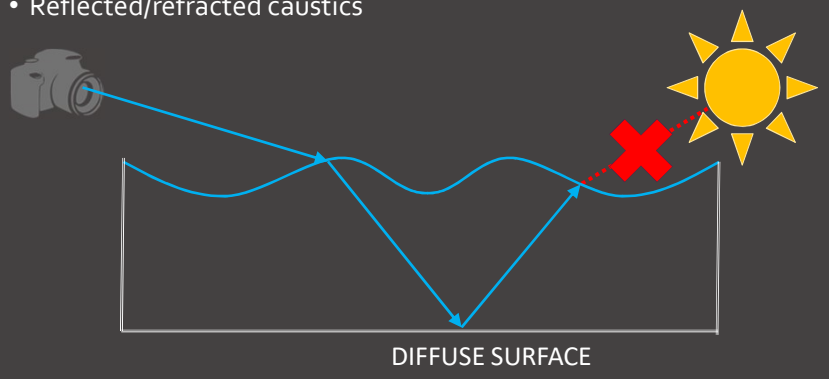
Indirectly seen caustics

- Reflected/refracted caustics



Indirectly seen caustics

- Reflected/refracted caustics

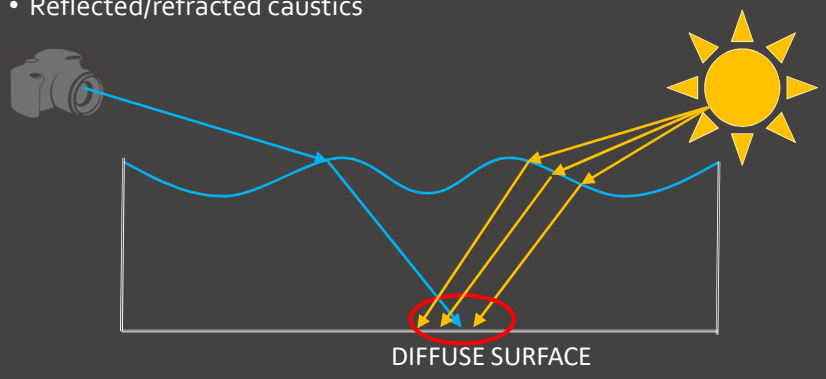


Indirectly seen caustics

And thus using just bidirectional path tracing techniques will lead to dissappearance of the caustics.

Indirectly seen caustics

- Reflected/refracted caustics



Fortunately, the photon lookup technique works in these cases

Indirectly seen caustics

And thus the full vertex connection and merging handles the indirectly seen caustics well.

Vertex connection and merging - issues

- Requires guiding of photon paths to be effective
- Overkill for scenes without difficult light transport

As I have already mentioned the bidirectional methods including the vertex connection and merging have some issues.
The most prominent one, is that without guiding the emission of photon paths from light sources, the algorithm may converge incredibly slowly in some scenes.
(CLICK)
Another issue is that the algorithm can be several times slower than path tracing.
Not only we need to trace paths from both the light sources and the camera, we also need to compute their combinations and do the photon lookups.
Therefore using VCM in scenes that are mostly handled well by path tracing is an overkill.

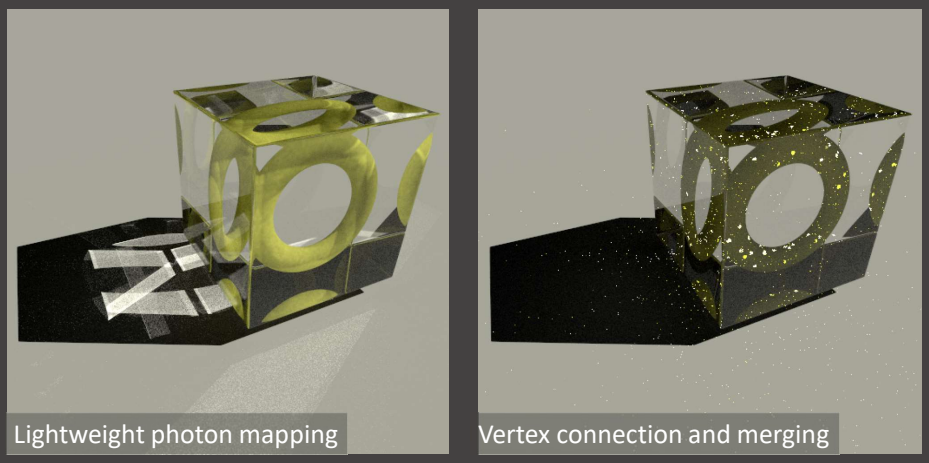
Vertex connection and merging - issues

- Requires guiding of photon paths to be effective
- Overkill for scenes without difficult light transport
- Unknown: How many photon paths should we use?

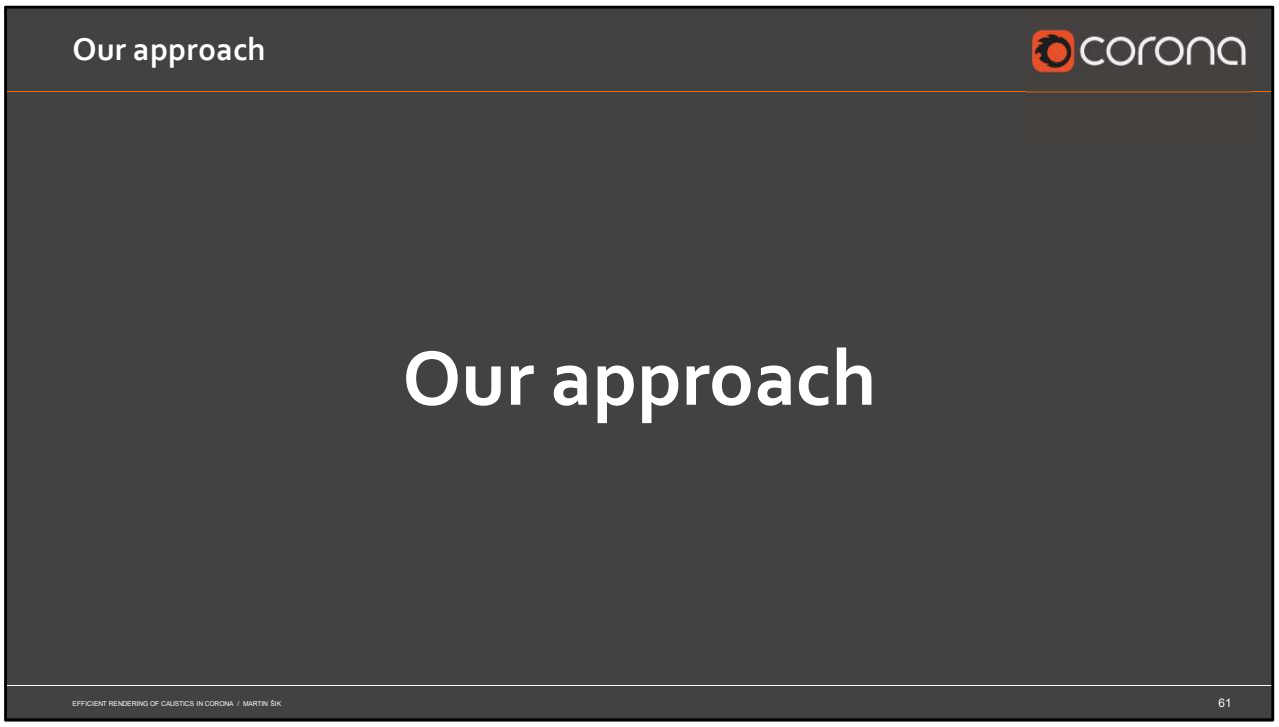
Finally, we have to trace paths from light sources as well, however no one really knows how many. The usual solution is to trace the same amount of paths from both sides, but this can be quite wasteful for scenes featuring just one small caustic. The users have to therefore tweak this parameter themselves.

Lightweight photon mapping

- Collaboration with Saarland University Lightweight photon mapping [Grittmann et al. 2018]



To solve these issues of VCM we have collaborated with Saarland university. (CLICK)
This collaboration then lead to development of the Lightweight photon mapping algorithm which was published at EGSR conference.
The algorithm is a huge improvement compared to VCM with respect to the aforementioned issues.
and therefore it is the main inspiration for our caustics solver.
However, the practical implementation of our caustics solver is quite different.



So let’s now discuss how we resolved VCM’s issues in Corona.

11:07 (6:00)

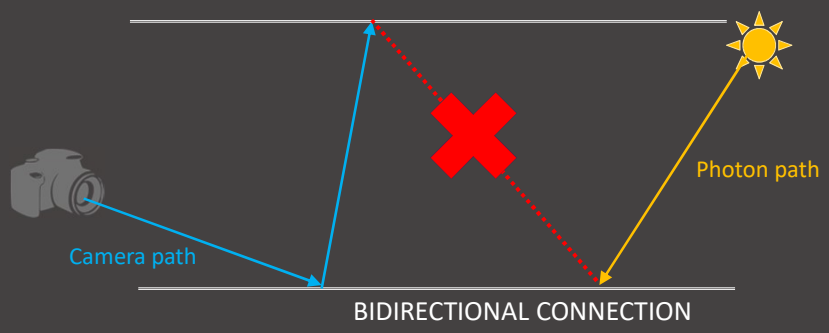
Our approach

- Reducing the overhead
- Guiding of photon paths
- Number of photon paths

Let us start with the issue of overhead. While we know our caustics solver will have some overhead compared to ordinary path tracing, we want to limit it as much as possible.

Reducing the overhead – simplified VCM

- Inspiration: Lightweight photon mapping [Grittmann et al. 2018]
- Remove bidirectional connections

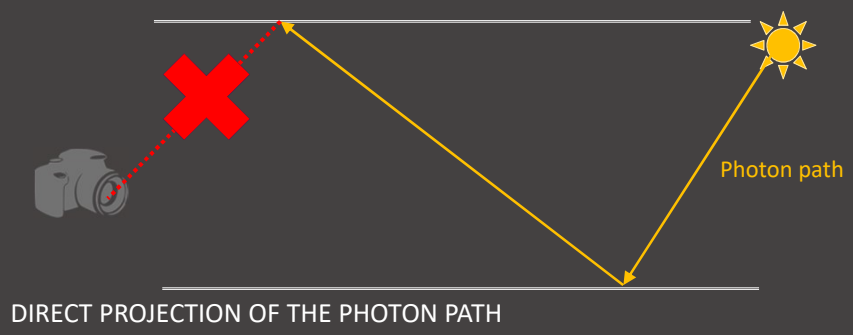


First of all, we get an inspiration from lightweight photon mapping algorithm and (CLICK) we completely remove bidirectional connections.

The reasoning behind this is that while using bidirectional connections decreases overall noise, it significantly increases overhead and the remaining techniques can still quite effectively handle all light transport phenomena as full VCM.

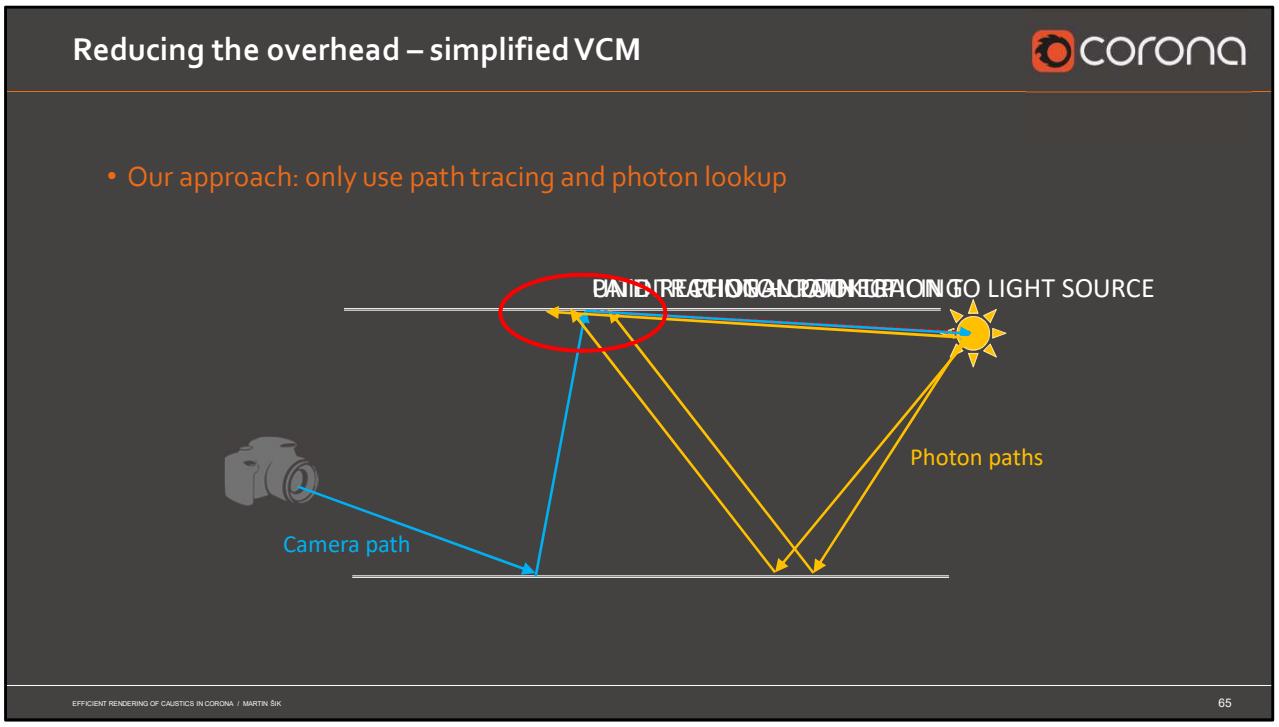
Reducing the overhead – simplified VCM

- Adaptivity => memory consuming (special buffer for each render element)
- Remove direct projection of the photon path



DIRECT PROJECTION OF THE PHOTON PATH

We go a bit further and realize that direct projection of the photon path to the camera can be memory consuming.
When using the usual path tracing adaptivity, this technique requires special image buffer for each render element, thus doubling the memory consumed by the Corona's virtual frame buffer.
(CLICK)
We therefore remove direct projection of the photon path as well

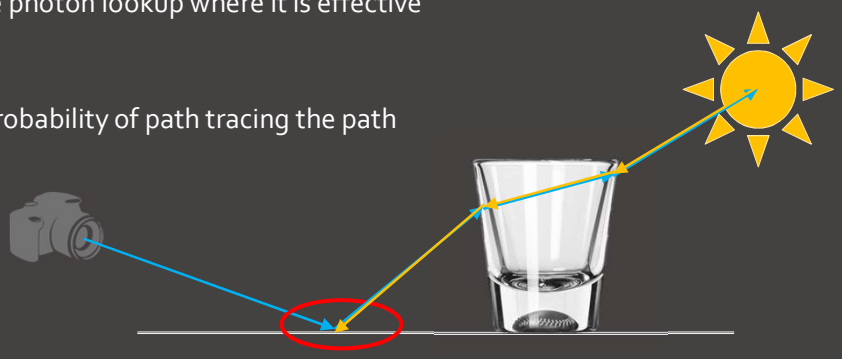


This leaves us with both path tracing techniques (CLICK) (CLICK) and (CLICK) photon lookup.

Note that such combination of techniques is still able to efficiently render all light phenomena that can be rendered by vertex connection and merging.

Reducing the overhead – photon lookup vs path tracing

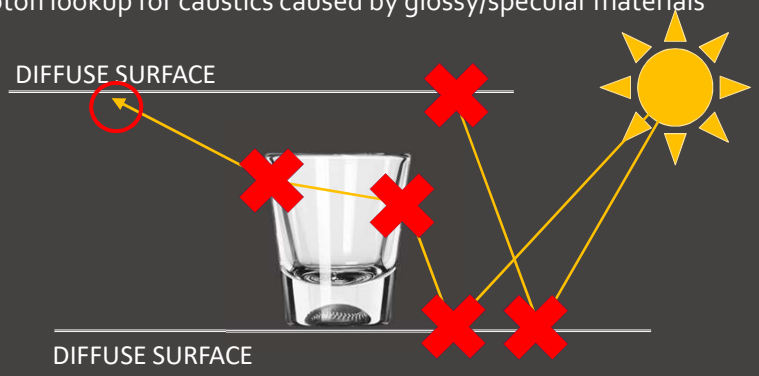
- Inspiration: Lightweight photon mapping [Grittmann et al. 2018]
- Only use photon lookup where it is effective
- Check probability of path tracing the path



Lightweight photon mapping algorithm further refines where to use photon lookup and where not by looking at the effectiveness of path tracing.
(CLICK)
Consider a path that can be created by both photon lookup and (CLICK) by path tracing.
(CLICK)
If the probability of generating the path by path tracing technique is high enough to avoid fireflies or their clamping, we don't use the photon lookup in this case to further reduce the overhead.

Reducing the overhead – storing photons

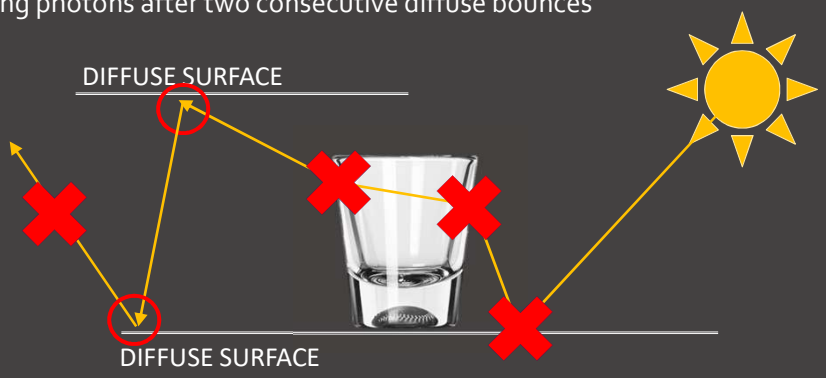
- Photon lookup => requires storing all photon path vertices (photons)
  Our approach: We further only store photons at certain locations
- Only utilize photon lookup for caustics caused by glossy/specular materials



Note that for photon lookup to work we need to first store all photon path vertices (or photons) from one iteration in memory.
To reduce both memory and computation overhead it is important to store photons only where they are important.
(CLICK)
First, we want to utilize our solver only for directly or indirectly seen caustics created by light focused through reflection or refraction.
(CLICK)
And thus we only store photons after the first interaction with a specular or a highly glossy material (CLICK) So we don't store them here (CLICK) here (CLICK) and we only store them here.

Reducing the overhead – storing photons

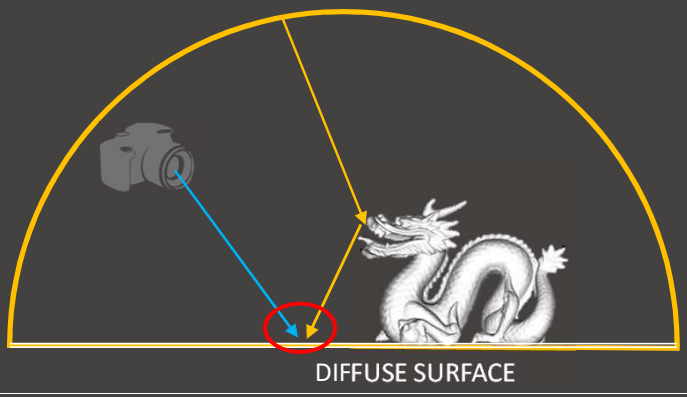
- Photon lookup => requires storing all photon path vertices (photons)
  Our approach: We further only store photons at certain locations
- Stop storing photons after two consecutive diffuse bounces



Second, we stop the photon path after two consecutive diffuse bounces(CLICK),
since then the caustics are blurred by the two diffuse reflections into weak uniform lighting.

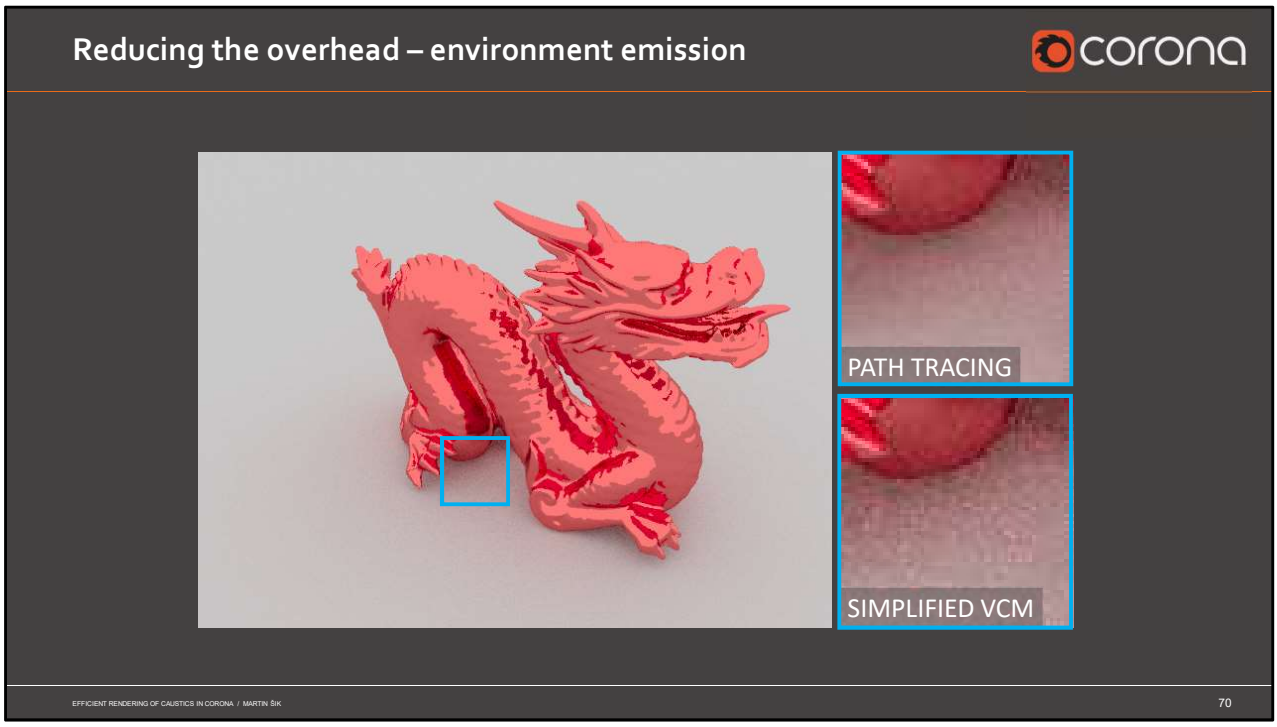
Reducing the overhead – environment emission

- Unfortunately there are still cases where photons are stored/used needlessly
- A common case: uniform environment emission



Unfortunately, even after considering all these cases, photons may still be used or stored in some places needlessly.
(CLICK)
Most notable case is the one with uniform environment illumination.
(CLICK)
Consider a photon path that starts at the environment and bounces off the specular surface of the dragon before hitting diffuse surface.

Photon lookup on the diffuse surface is still considered, even though it is highly ineffective.



We can see this in the following comparison, where the inset image computed by path tracing,
is much less noisy than the inset image computed by the combination of photon lookup and path tracing.

Reducing the overhead – environment emission

- Our approach: Prune emission from environment
- Completely remove emission from uniform environment
- Don't emit from low-value part of environment map

To avoid this issue, we further prune the emission of photon paths from the environment.
(CLICK)
First of all, we completely remove emission of photon paths from uniform environment, since caustics from such environment are easily handled by path tracing.
(CLICK)
Furthermore, we also don't emit photon paths from low-value parts of any environment map.

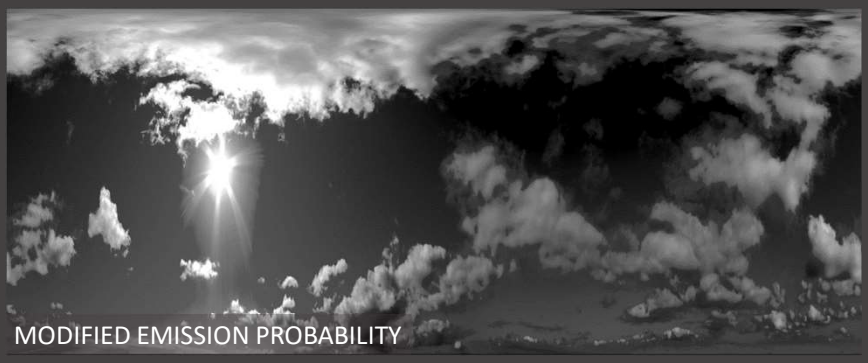
Reducing the overhead – environment emission

To better understand this, consider this environment map.

Reducing the overhead – environment emission

The emission from such map is traditionally based on the color value in each pixel.

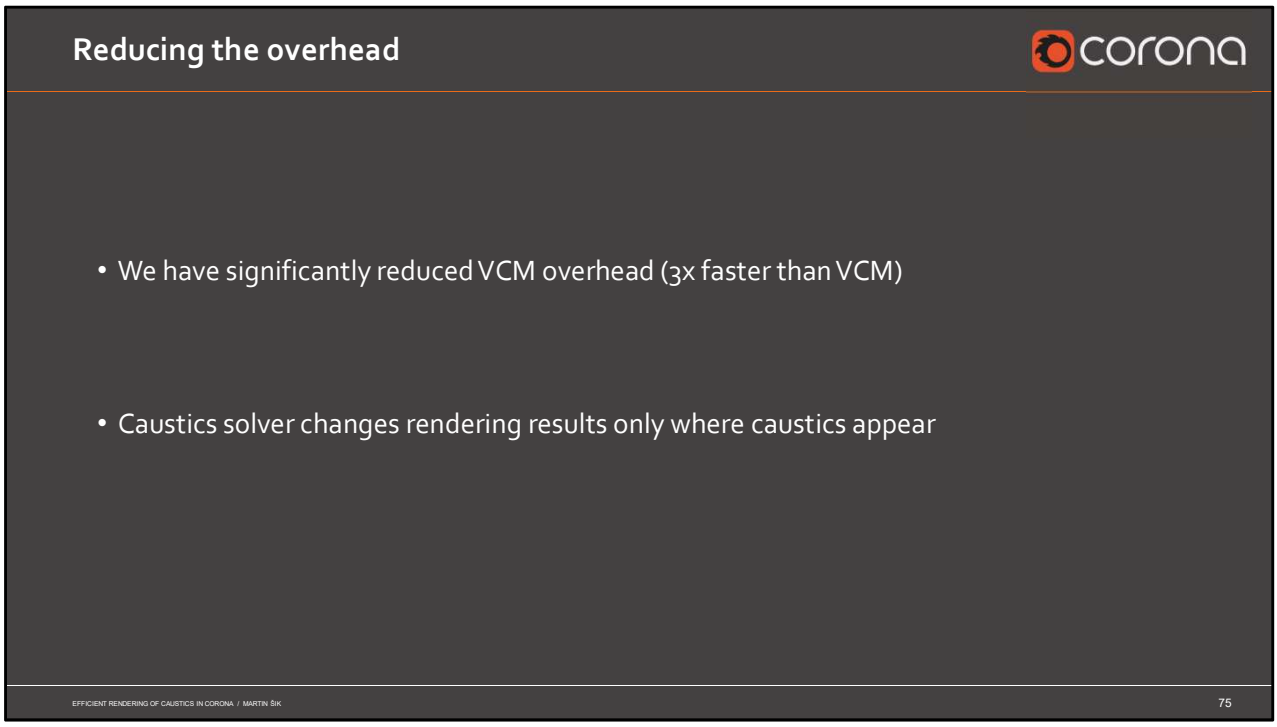
Reducing the overhead – environment emission



In our approach we further lower the emission probability of the low-value parts of the environment map, while increasing the emission from other parts.

Such change of emission ensures that photons are not emitted and stored in places where they would lead to unnecessary overhead.

Reducing the overhead



Due to discussed changes to the original vertex connection and merging, we have significantly reduced its overhead.

Our solver is roughly three times faster than the original vertex connection and merging in scenes featuring many caustics.

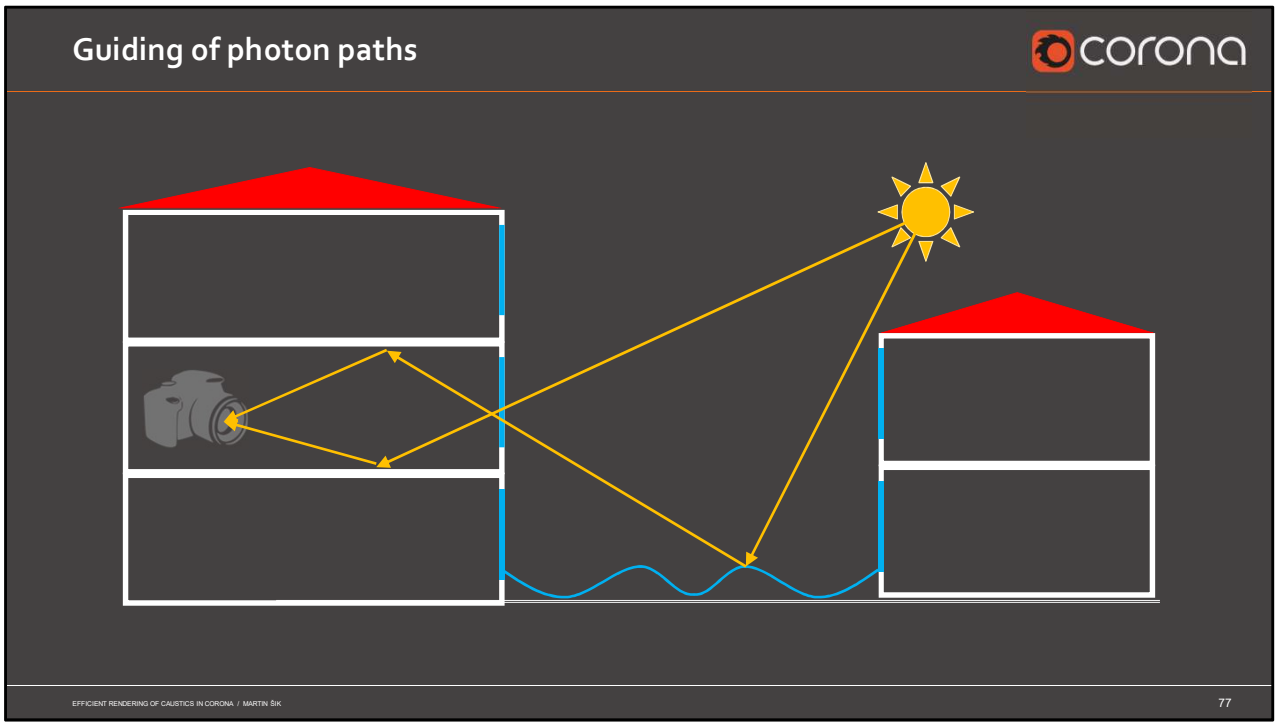
Since we apply our solver where it is really necessary and use path tracing in other cases the rest of the scene renders as before.

15:41 (4:34)

Our approach

- Reducing the overhead
- Guiding of photon paths
- Number of photon paths

Let us now discuss the issue of guiding the photon paths in large or complex scenes.

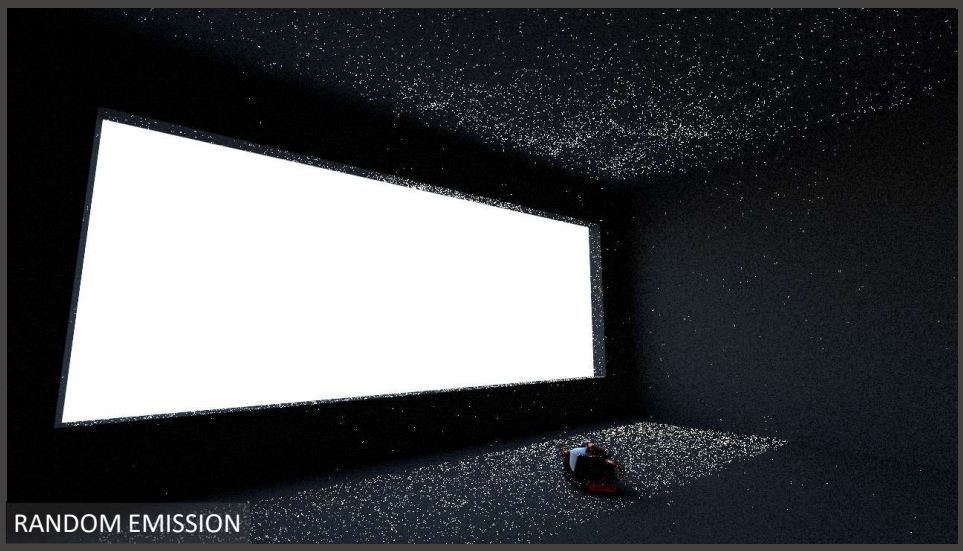


We start with a difficult test scene, where a building with many apartments is lit from the outside. (CLICK)
The camera is then placed in one of the apartments (CLICK) and the light coming through the window create caustics on the floor.
(CLICK)
There is also a pool next to the building that creates another set of caustics on the apartments‘s ceiling.

Guiding of photon paths - comparison

Here is how such rendering looks like.

Guiding of photon paths - comparison



We can see that with completely random emission of photon paths only few will actually reach the visible region leading to extremely slow convergence.

Guiding of photon paths

- Lightweight photon mapping [Grittmann et al. 2018]
- Adaptive emission = More paths to successful emission directions
- We have tried to follow this approach

The algorithm lightweight photon mapping approaches this issue by utilizing adaptive emission.
This means they attempt to emit more photon paths in directions, where previously emitted photon paths contributed to the image.
First, we have tried to follow this approach.

Guiding of photon paths - comparison

As we can see adaptive emission significantly improved the result, however parts of the caustics are less converged, since just the adaptive emission is insufficient in guiding the photon paths where they are necessary.

Guiding of photon paths – our approach

- We utilize Metropolis sampler [Metropolis et al. 1953]
- What about the image artifacts?
- [Šik et al. 2016] Careful combination Metropolis + VCM = no artifacts

So we were thinking how could we improve the result? We have decided to utilize Metropolis sampling.
(CLICK)
Wait a minute, didn't we mention that Metropolis leads to horrible artifacts in animations?
(CLICK)
Fortunately, one of the topics of my own dissertation thesis, was removing the artifacts by carefully combining Metropolis with vertex connection and merging.
So I knew how to approach this issue.

Guiding of photon paths – our approach

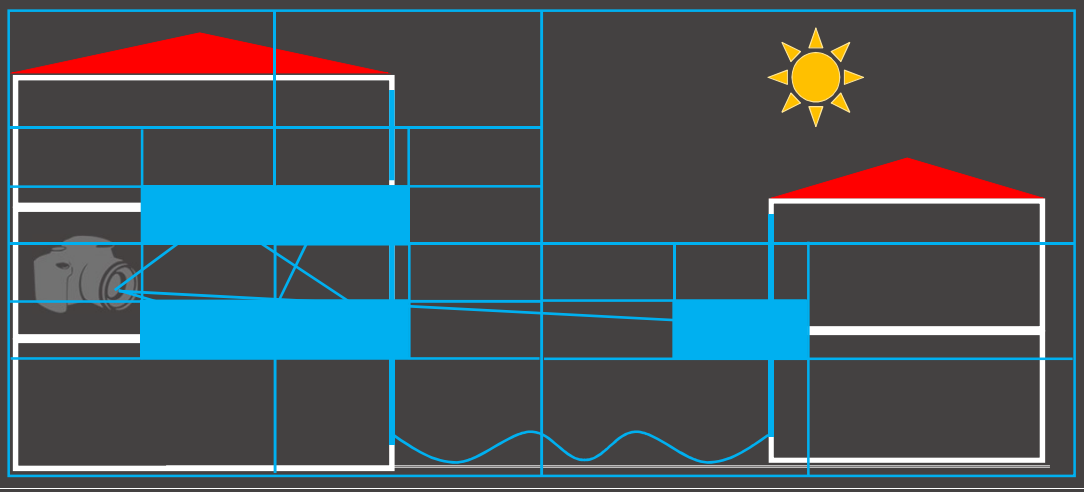
Reducing correlation in Metropolis + VCM:

- Metropolis only for paths from light sources
- Simple definition of important photon paths

Besides some more technical aspects that I will not discuss here, there are two important things to ensure the artifacts disappear.
(CLICK)
First, we trace paths from the camera as in the usual path tracer, and we only utilize Metropolis for tracing paths from the light sources.
(CLICK)
Second, we must define for the Metropolis sampler as simply as possible which paths are important and should be mutated.
Using some complicated definition of important photon paths may lead to faster convergence, but at the cost of appearance of image artifacts.

Guiding of photon paths – Metropolis

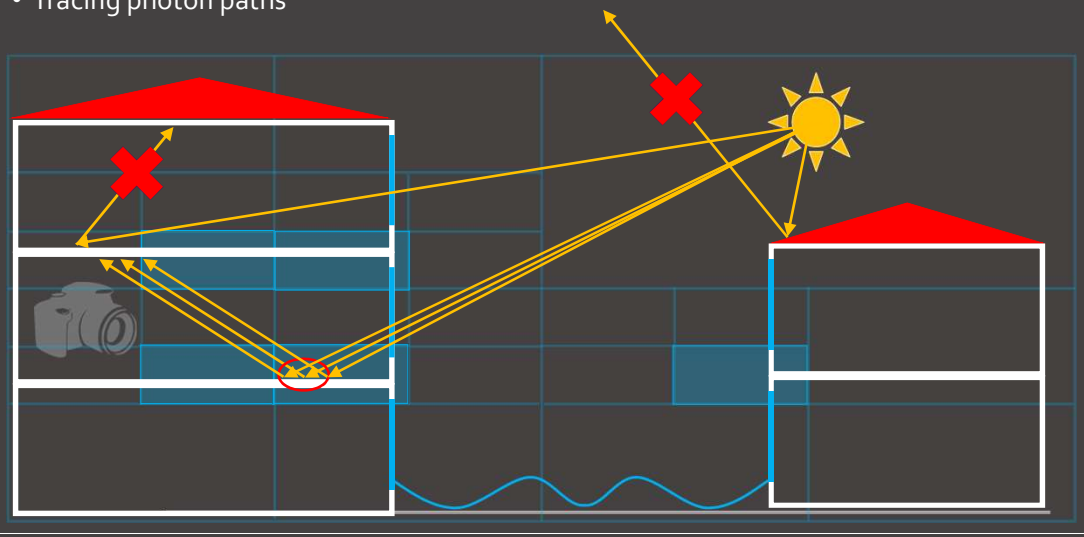
- Important paths = visible paths [Hachisuka and Jensen 2008]



Therefore to avoid the artifacts, we define important paths for the Metropolis sampler only based on their visibility to the camera paths.
(CLICK)
More specifically, we first trace paths from the camera and store their vertices.
(CLICK) 6x
After that we build a structure over the whole scene and recursively subdivide it based on the camera vertices positions.
(CLICK) 4x
Finally, we mark in this structure (CLICK) where the camera vertices lie.

Guiding of photon paths – Metropolis

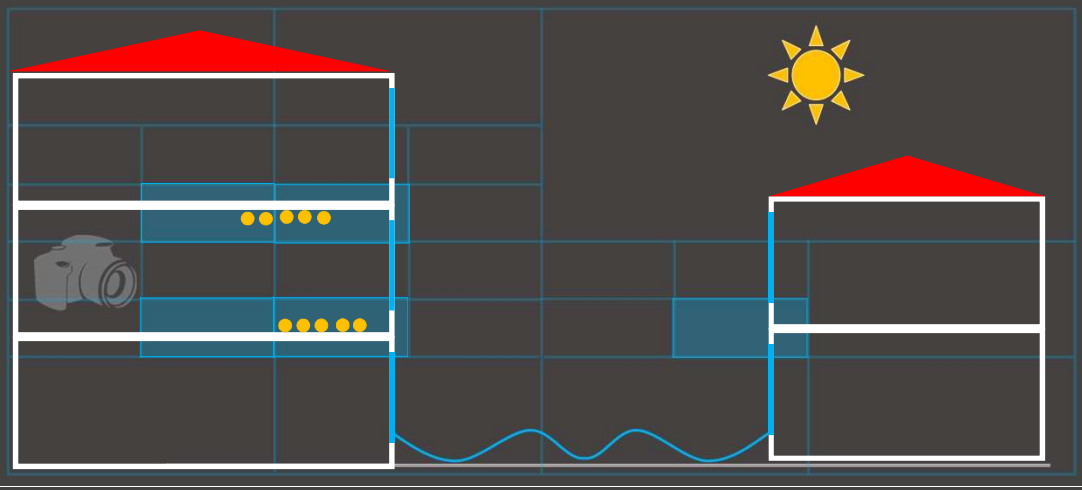
- Tracing photon paths



When we trace photon paths using Metropolis we utilize the structure as follows.
(CLICK)
We trace a photon path and look if any of its vertices lie in the visible regions stored in the structure. If not, the path is rejected.
(CLICK)(CLICK)(CLICK)
This way we continue until we find a path that has at least one vertex in the visible region.
(CLICK)(CLICK)
Metropolis will then slightly modify the path and generate several similar photon paths. (CLICK)
This way we ensure that many of the photon paths are generated in the important regions of the scene.

Guiding of photon paths – Metropolis

- Improving definition -> consider photon lookup locations



We can further improve the Metropolis sampling(CLICK) by only marking places in the structure where successful photon lookup occurred in the previous iterations.
(CLICK)
This way we will truly guide photon paths where they are useful for our caustics solver.

Guiding of photon paths - comparison

Using the Metropolis with such definition of important photon paths than gives us this result.

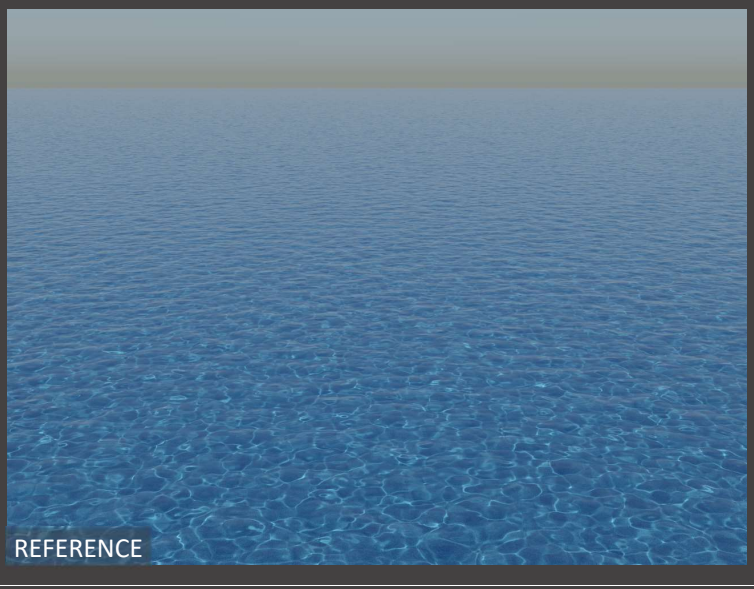
Guiding of photon paths - comparison

As you can see the result has far less noise compared to adaptive emission.

Guiding of photon paths - comparison

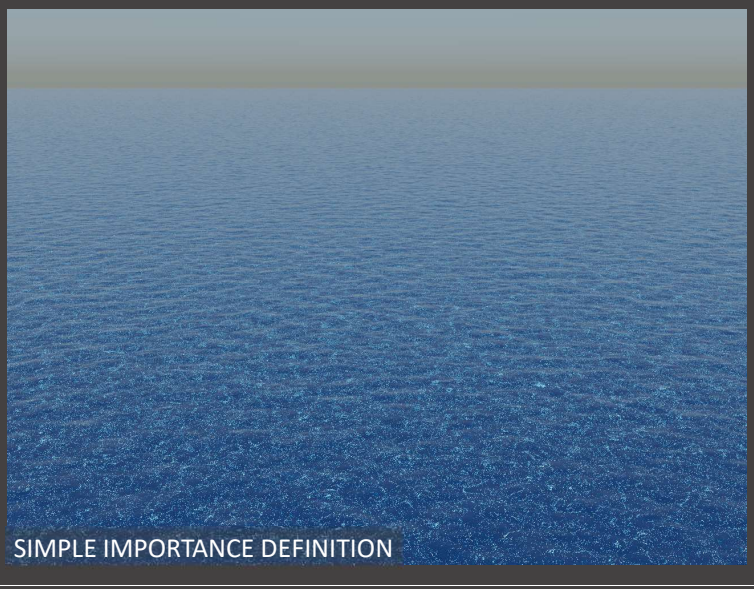
Furthermore as we can see in this video sequence, our Metropolis does not suffer from the usual flickering.

Guiding of photon paths – Metropolis



Unfortunately, there are cases where such distribution of photon paths is not effective. For example consider this scene, where we can see caustics at the botom of the ocean.

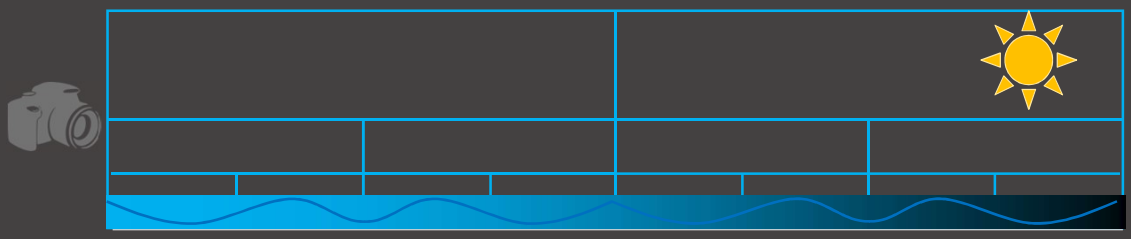
Guiding of photon paths – Metropolis



In this case the whole visible region is considered to be equally important. The photon paths are therefore distributed equally in the large visible region of the ocean, which results in a suboptimal convergence.

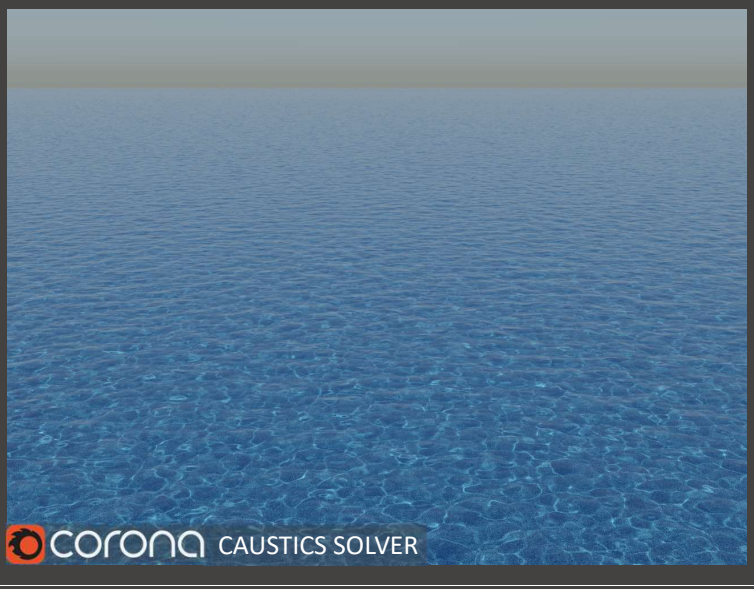
Guiding of photon paths – Metropolis

- Improving important paths definition = decrease with distance from the camera



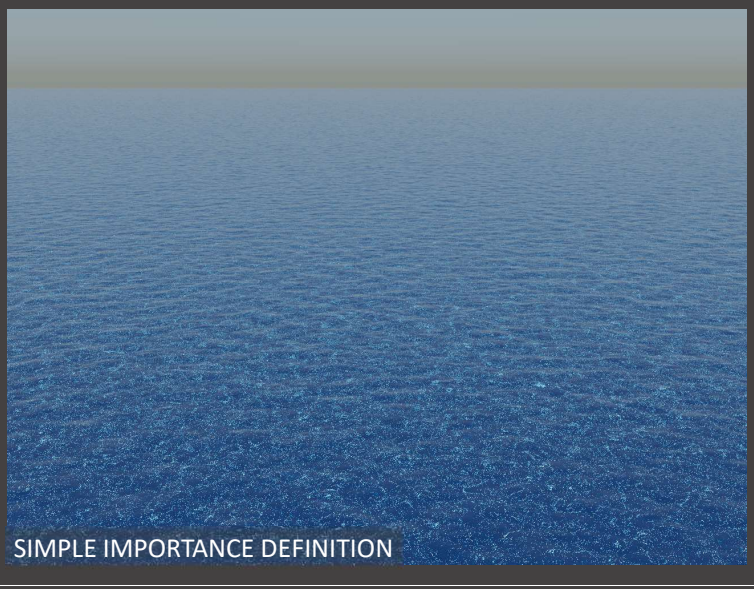
We can improve the result here, by decreasing the importance of photon paths with the distance from the camera.
(CLICK)
Here is how the original importance of photon paths looks like
(CLICK)
and we will just modulate it with the inverse squared distance from the camera.

Guiding of photon paths – Metropolis



This means Metropolis will generate more photon paths near the camera and it results to less noisy image, compared to the simpler definition of important photon paths.

Guiding of photon paths – Metropolis



…

20:14 (4:33)

Guiding of photon paths - emission probability

- Traditionally photon path emission is based on the power of light sources
- More power = more photon paths

Last aspect of our approach of guiding of photon paths is the selection of light source from which to emit a photon path.

Traditionally photon path emission is based on the power of the light source.

Therefore light sources with high power will generate more photon paths.

Initially, we have followed this approach.

Guiding of photon paths - emission probability

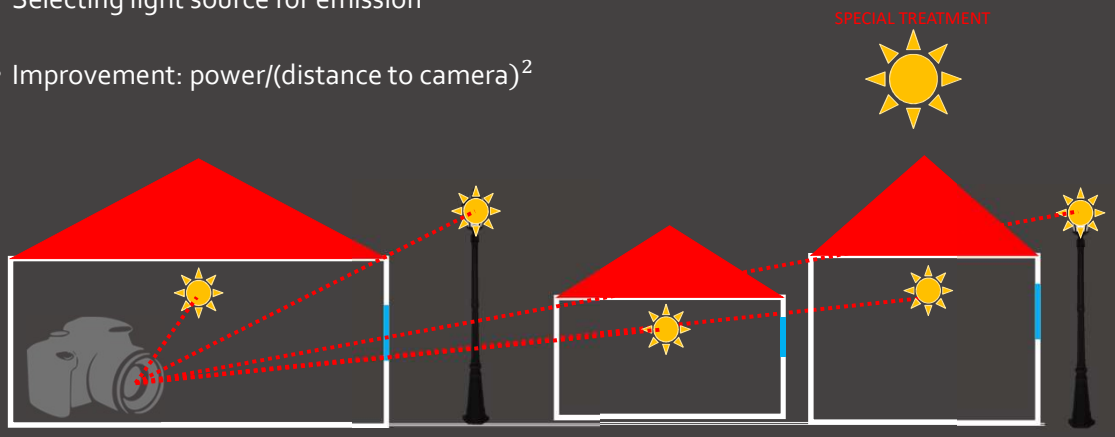
It worked quite well for simple scenes such as this one that feature one or few light sources with similar power.

Guiding of photon paths - emission probability

Unfortunately, when we added to the scene several additional light sources including sun and sky emitter, which has much higher power compared to the light bulb, only few of the photon paths were emitted from the original light bulb and the previously seen caustic disappeared.

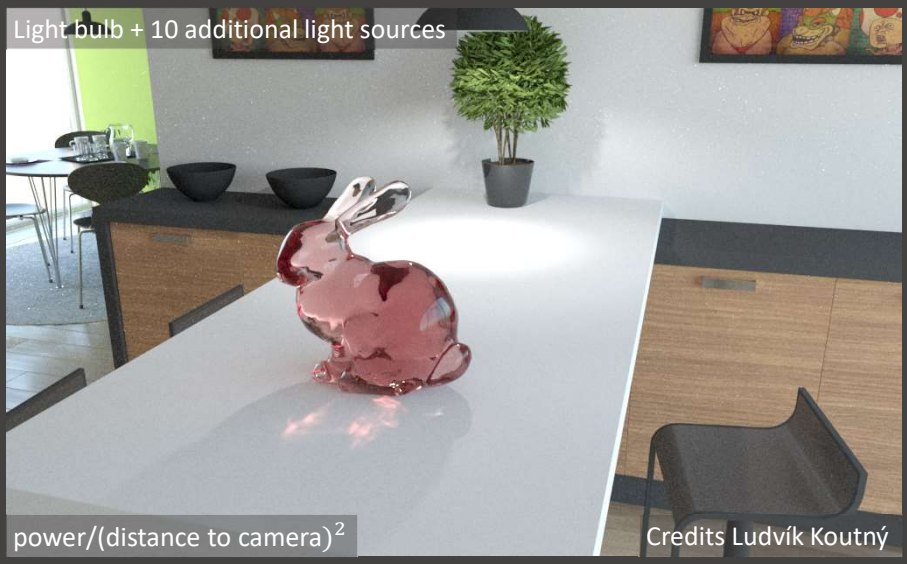
Guiding of photon paths - emission probability

- Selecting light source for emission
- Improvement: power/(distance to camera)^2



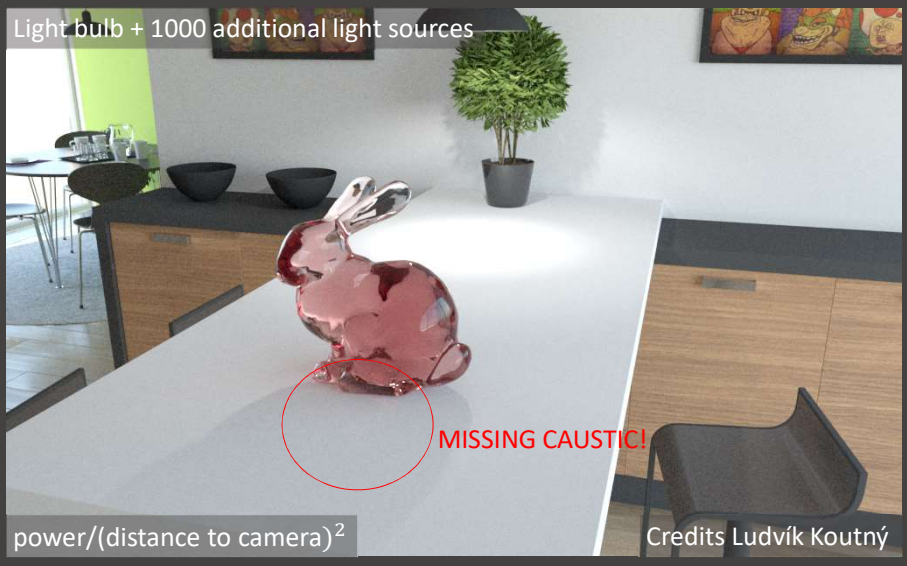
To solve this issue we have added the factor of inverse distance to the camera as in the case of the definition of important photon paths for the Metropolis sampler.
(CLICK)
Since, sun and environment are almost infinitely far away, we can not apply the distance on them.
(CLICK)
Therefore, we always generate half of the photon paths from sun and environment and half from the rest of light sources.

Guiding of photon paths - emission probability

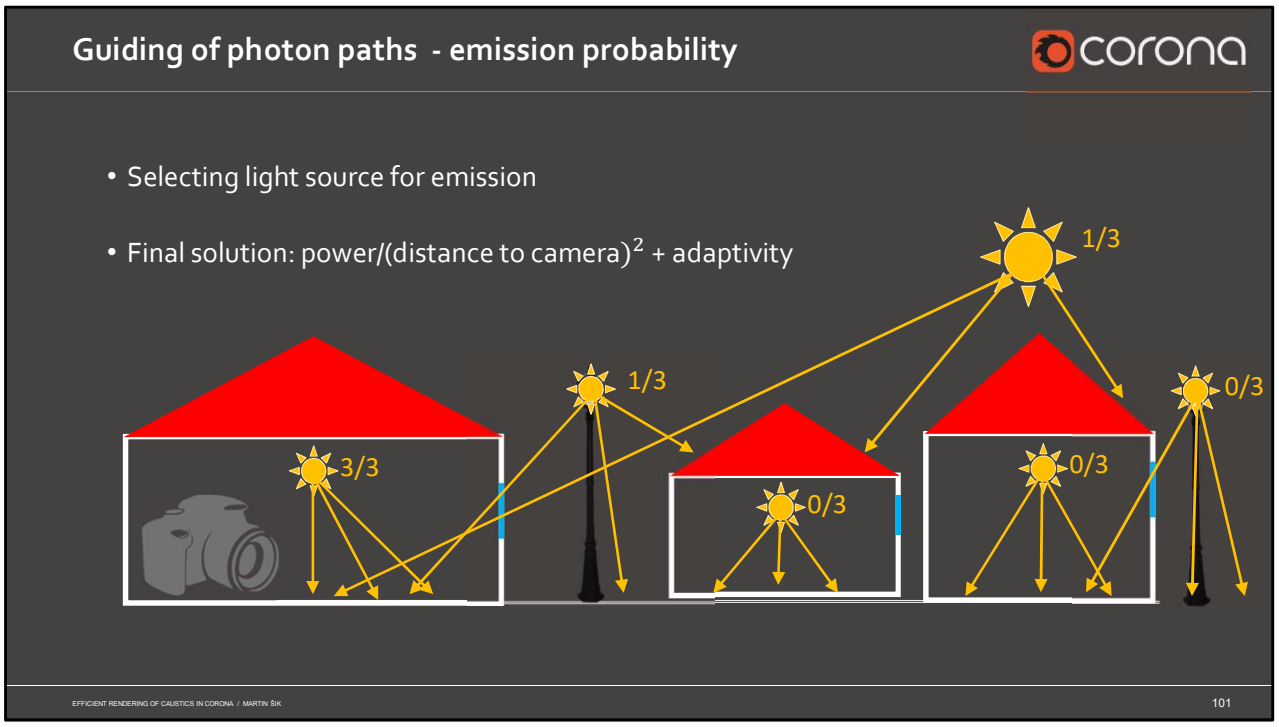


Including the factor of distance to the camera ensures the caustic was again properly rendered in this scene.

Guiding of photon paths - emission probability

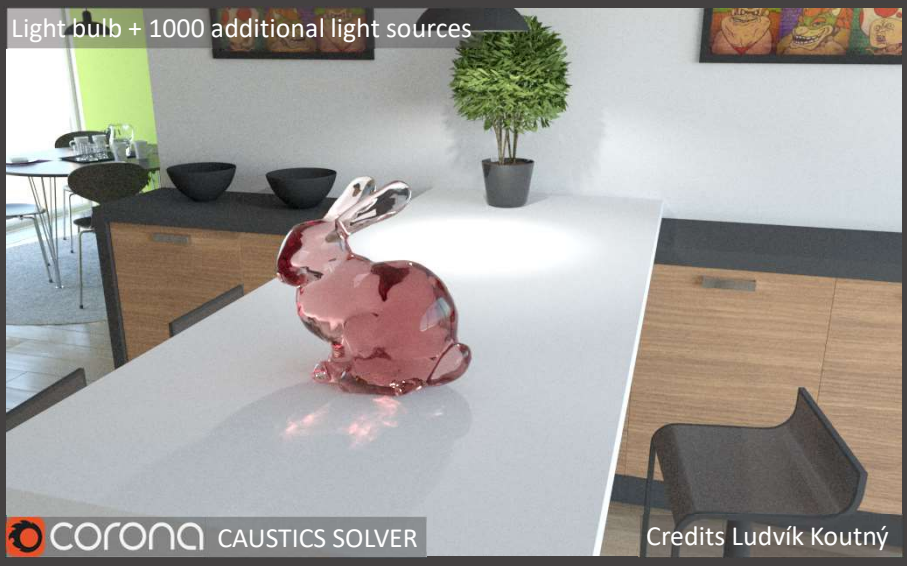


Unfortunately, adding many more light sources to the scene again reduced the number of emitted usable photon paths and the caustic again disappeared.
And therefore we had to improve the solution further.



To robustly solve this issue we have decided to further adapt the probability of emission of each light source by counting how many photon paths generated from that light source have contributed to the image.
(CLICK)
So this light source has generated only contributing paths(CLICK), so its probability of emission will be high.
(CLICK)
This light source has generated only few contributing paths(CLICK), so its probability of emission will be lower.
And so on… (CLICK)(CLICK) (CLICK) (CLICK)

Guiding of photon paths - emission probability



Our final solution, can handle even the difficult case with thousands of light sources.

Our approach

- Reducing the overhead
- Guiding of photon paths
- Number of photon paths

This brings us to the last issue, which is how many photon paths should we trace each iteration.

Number of photon paths

- Inspiration: Lightweight photon mapping [Grittmann et al. 2018]
- Compare photon lookup and overall contribution
- Number of photon paths = number of pixels with significant photon lookup contribution

OVERALL CONTRIBUTION

PHOTON LOOKUP CONTRIBUTION

Again, we take inspiration from Lightweight photon mapping.

The algorithm compares the overall contribution from all the techniques (CLICK) with the contribution coming only from photon lookup (CLICK).

If the contribution from photon lookup in a given pixel is high enough compared to overall contribution, the photon lookup is deemed to be useful in that pixel.
(CLICK)
The number of photon paths is then directly equal to the number of pixels where photon lookup was useful.

In this case, photons contribute to only a small region and thus photon paths number will be significantly lower than the number of pixels.
(CLICK)
In this case, the caustics are everywhere in the image and thus we trace as many photon paths as there are pixels.

Number of photon paths

ONLY FEW PHOTON PATHS TRACED!

- Does not work well with render regions/small resolution images

OVERALL CONTRIBUTION

PHOTON LOOKUP CONTRIBUTION

Unfortunately, this solution does not always work with Metropolis.
For example, consider a scene that requires good guiding of photon paths and where we want to render just a small region.
(CLICK)
The number of photons paths is thus limited by the small region resolution.
(CLICK)
Unfortunately, to ensure good photon path guiding in such a scene, Metropolis requires to send photons in large batches, and thus if their number is limited by the small resolution, guiding won't be effective enough and image will converge slowly.

Number of photon paths

- Our approach: We don't base maximum number of photon paths on resolution
- Number of photon paths = ratio of pixels with significant photon lookup contribution X maximum number of photon paths
- Balance camera paths/photon paths

We avoid this issue, by setting a fixed maximum number of photon paths, which is not limited by the image resolution.

We compute ratio of useful pixels and total number of pixels similarly to Lightweight photon mapping.
(CLICK)
We then multiply the maximum number of photon paths with this ratio to get the number of traced photon paths.
(CLICK)
Since such number can be much higher than the resolution of the image, we further balance the tracing of the paths from the camera and from the light sources.

Number of photon paths

- Example #1: We need to trace 4 x more photon paths

Path tracing

Photon tracing

Path tracing

Path tracing

Path tracing

Path tracing

Photon tracing

More specifically, if we need to trace 4x more photon paths than there will be paths traced from the camera, such as in the case of render regions.
(CLICK)
We first do path tracing (CLICK)followed by photon tracing.
(CLICK)
Then we do 4x times path tracing and utilize the same photons from the first photon tracing.
(CLICK)
Only after that we do the next batch of photon paths, followed again by 4 batches of path tracing.

Number of photon paths

- Example #2: We need to trace 4 x less photon paths

Path tracing

Photon tracing

Path tracing

Photon tracing

Path tracing

Photon tracing

Path tracing

On the other hand, in the case the caustics are present only in some parts of the scene, we may
trace 4x less photon paths that there is camera paths.

In such case, we interleave the photon tracing with path tracing. (CLICK)

This way we decrease the overhead of the caustics solver compared to ordinary path tracing.

Other implementation details

Of course, there is more to our implementation, besides solving the aforementioned issues of vertex connection and merging.
I will just quickly mention some of these details.

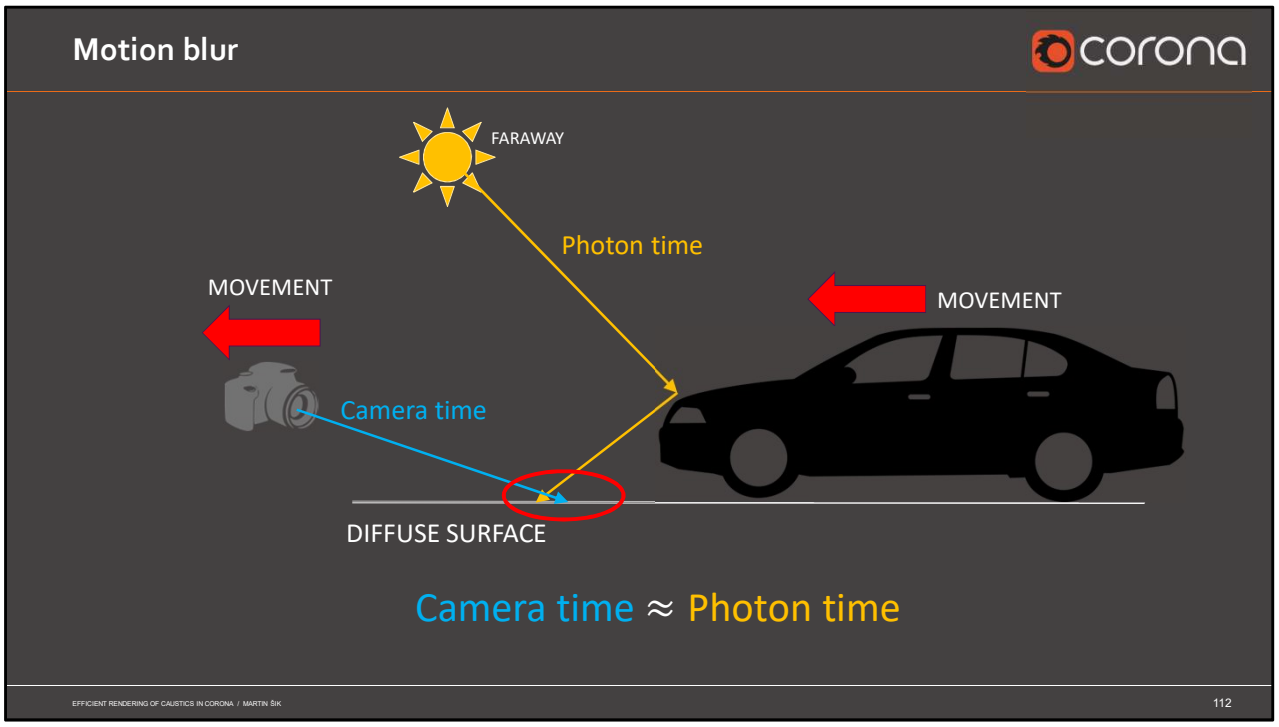
24:24 (4:10)

Motion blur

First such detail is motion blur.
Consider this scene which contains moving car (CLICK) that is tracked by the camera. When rendered without motion blur, the caustic is correct.

Motion blur

However, when rendering with motion blur, we have to take special care otherwise the caustic will be smeared even though the car is tracked by the camera and light source is faraway.



In our test example the camera (CLICK) follows the moving car (CLICK) .
To correctly handle caustics with motion blur in this case, (CLICK) we must independently randomly select an exact time for the camera path and (CLICK) for the photon path.
(CLICK)
And then when we perform photon lookup, we only consider photons that have similar time as the camera path.

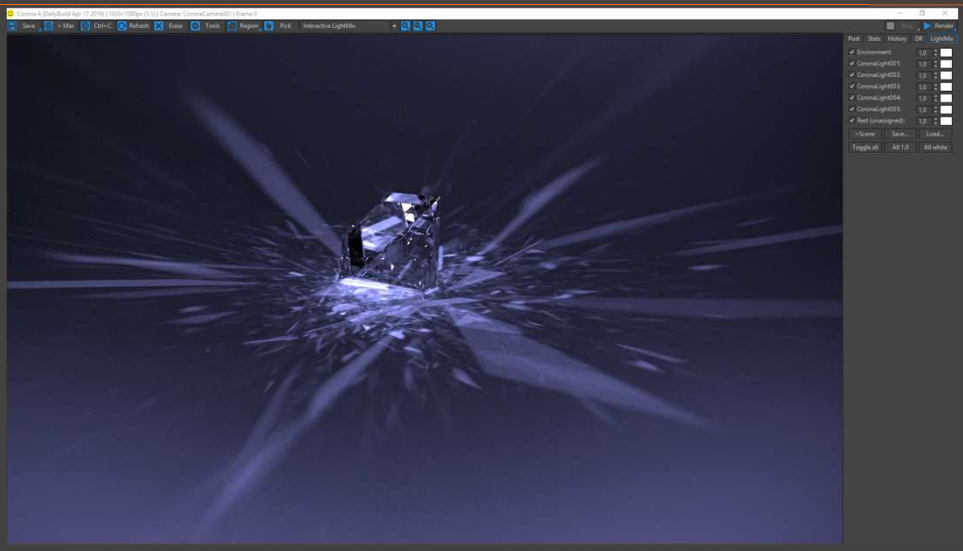
Motion blur

This leads to the correct solution displayed here, which does not smear the caustic.

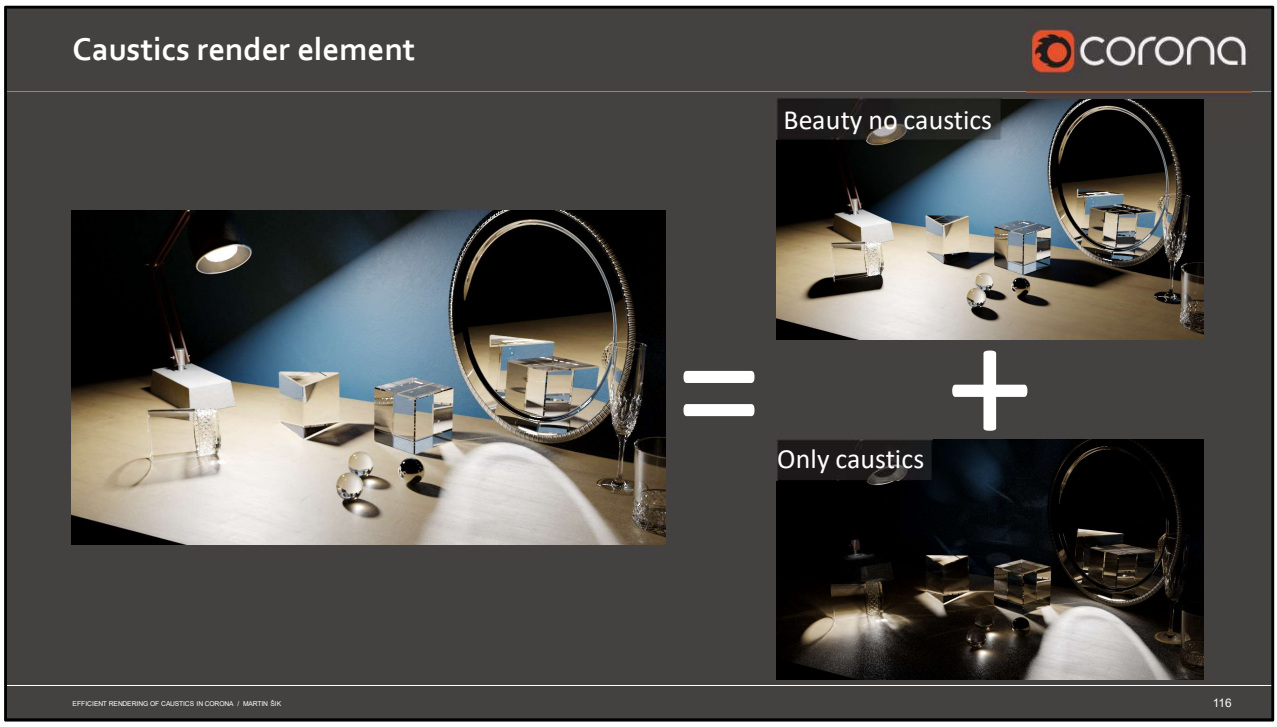
Dispersion

Similarly to motion blur, we also had to modify our caustics solver to correctly handle dispersion caused by the varying index of refraction of the crystal.
During photon lookup we only consider photons that have similar wavelength as the camera path.
This allowed us to create such nice colorful caustics.

Light mix

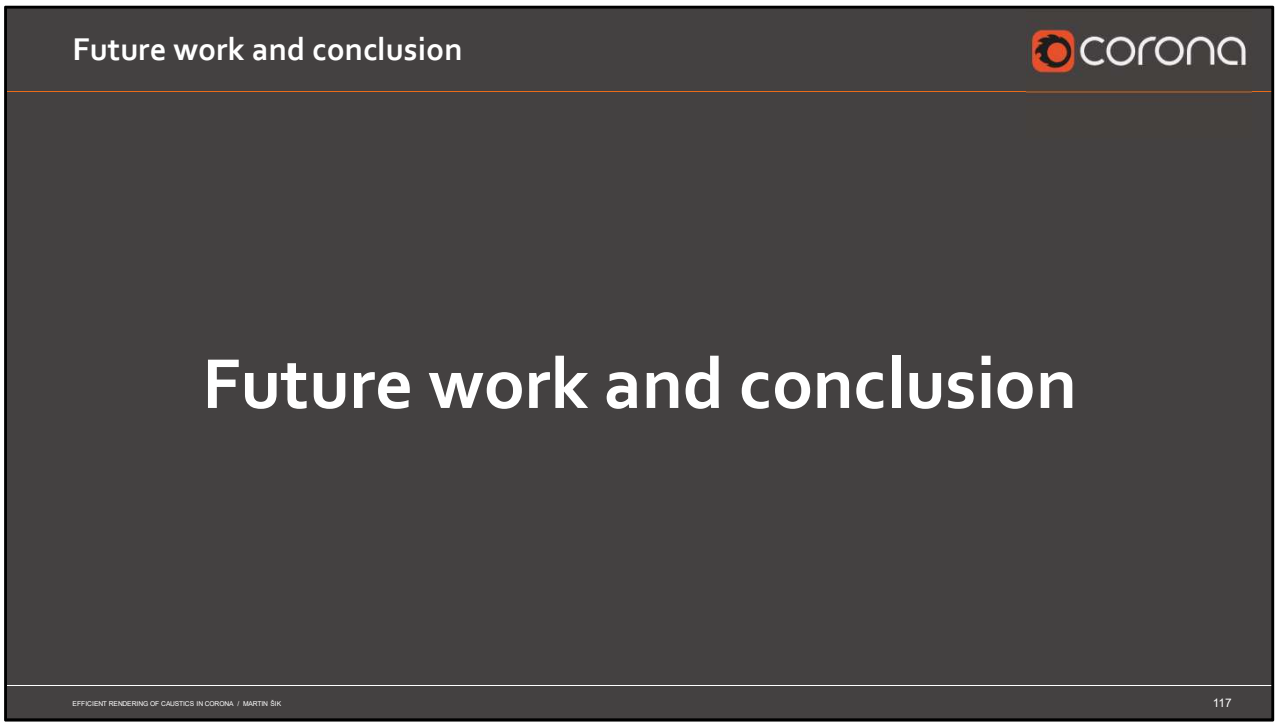


When implementing our caustics solver, we have also ensured that it fully supports Corona's popular feature light mix that allows changing light source power and color without re-rendering.
As demonstrated in this video.



Finally, to allow users to have better control over the caustics we have added an option to our caustics solver to completely remove caustics from the main beauty element (CLICK) and just store them into separate caustics element. (CLICK)

This way the users can have absolute control over which caustics they want to keep in the final image.



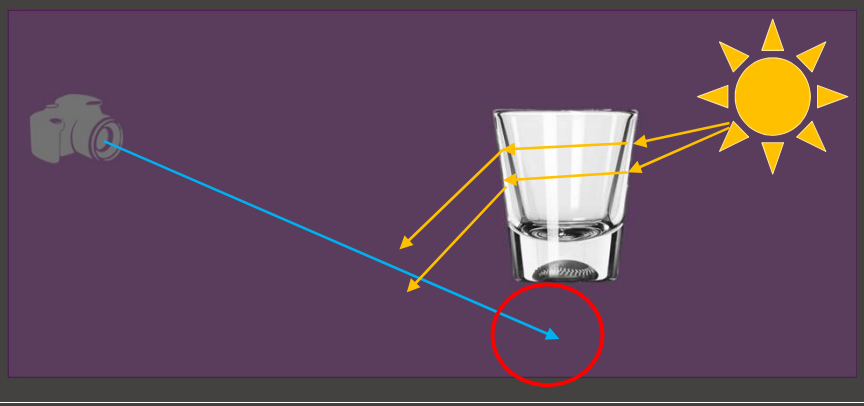
Before the end of my presentation, let me quickly mention some important future work.

Future work - caustics in media

Corona version 4 will feature only the initial version of the caustics solver. There is still lof of work that can be done.
One of the things that we left for future improvements, and our users are already frequently requesting it, is handling of caustics in media.

Future work - caustics in media

- Photon lookup can work in media as well
- However: Low chance of finding any photon!

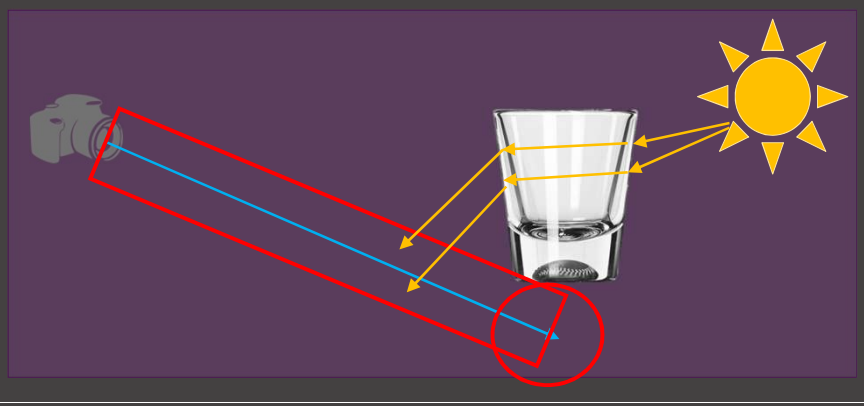


While we could utilize the same approach as we utilize on surfaces, it would not be as effective.

Consider this scenario, where the whole scene is inside global purple medium.
(CLICK)
In this case we want to perform photon lookup inside the red circle,
(CLICK)however if the medium is sparse, we will have very low chance of finding any photons inside the lookup radius.

Future work - caustics in media

- Combine different estimators in media [Křivánek et al. 2014]
- Example: beam x point estimate



To overcome this issue, team of researches including myself led by our own Jaroslav Krivanek have proposed to combine several estimators to handle caustics in media with different density.

One example of such estimator, is the beam point estimator (CLICK) that gathers the contribution of all photons that sorround the camera path. In the future we would like to update our caustics solver with such estimators.

Conclusion

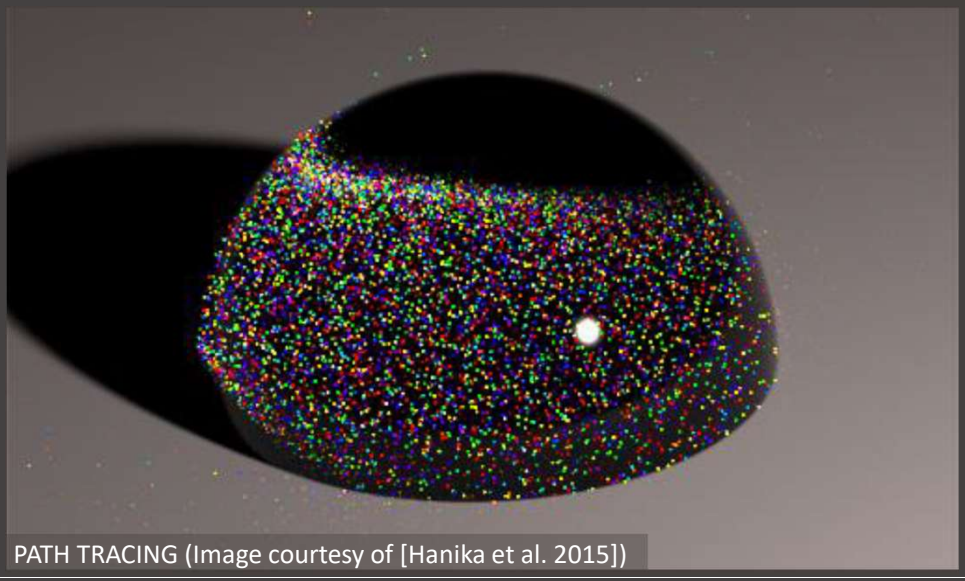
- Caustics – lightweight solution (roughly 2x times slower than path tracing in scenes with many caustics)
- Automatic – no hand setup needed:
  - Guiding of photon paths
  - Number of photon paths
- Solver supports all Corona settings

This brings me to the end of my talk.
I have presented here a new caustics solver that was implemented in Corona version 4.
While the solver is based on vertex connection and merging,
it is quite lightweight and in scenes featuring many caustics it is roughly only 2x times slower than ordinary path tracing
which fails to resolve the caustics.
The solution we propose is fully automatic, meaning the user does not have to setup anything by hand.
The solver will automatically guide the required photon paths toward the important visible regions.
We also automatically estimate the required number of photon paths.
Finally, the proposed solver is compatible with all other Corona settings and behaves the same way as ordinary path tracing in the parts of the scene without caustics.

Thank you for attention and if you have any question, I will be happy to answer them.

28:24 (4:00)

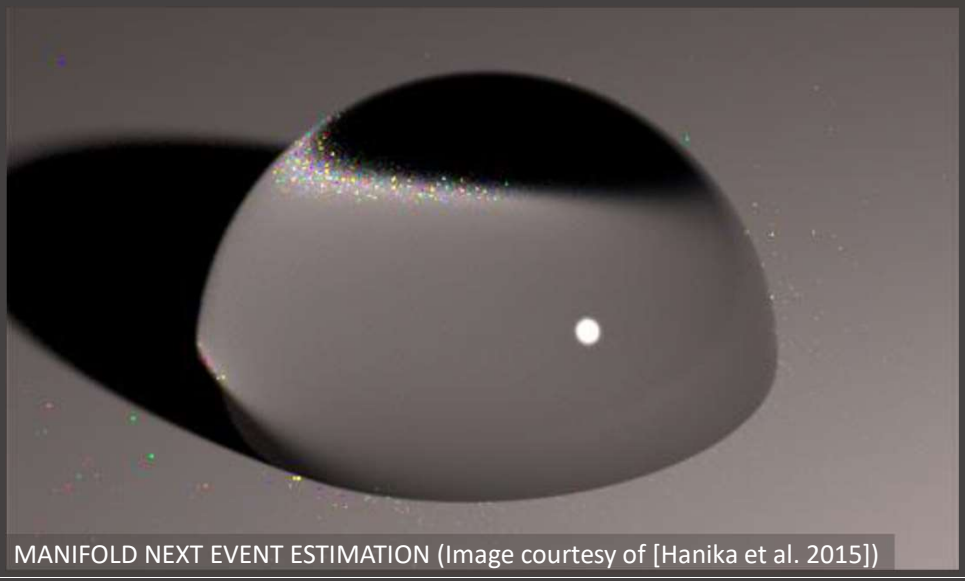
Manifold next event exploration



PATH TRACING (Image courtesy of [Hanika et al. 2015])

So it can

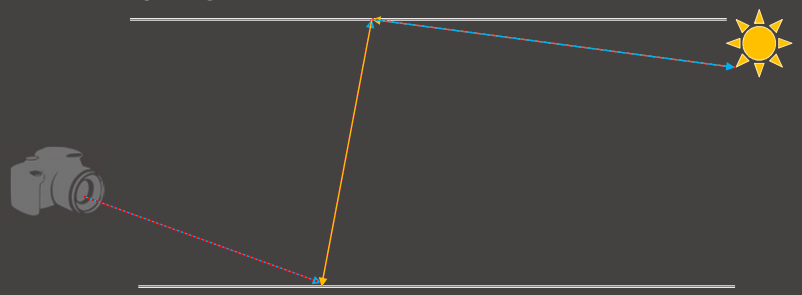
Manifold next event estimation



MANIFOLD NEXT EVENT ESTIMATION (Image courtesy of [Hanika et al. 2015])

Multiple importance sampling

- Veach and Guibas [1994]
- Effective combination of estimators: Multiple importance sampling weights

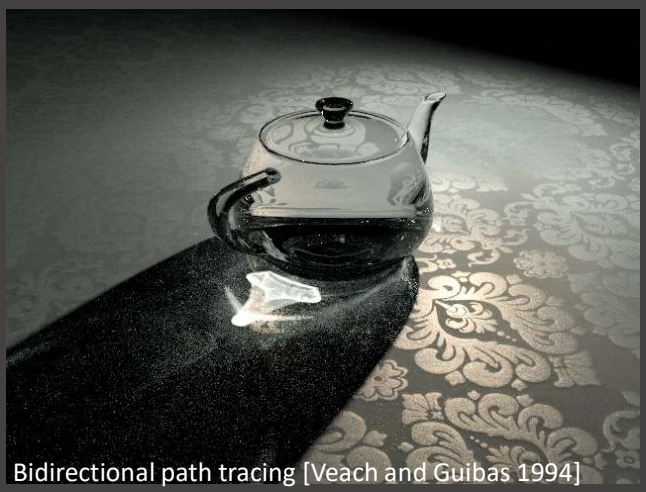


Note that the same path can be created by randomly hitting light source from the camera (CLICK), by next event estimation (e.g. directly sampling the light source) (CLICK),
by connecting two subpaths(CLICK) or just directly sampling the camera. (CLICK)
To ensure we use the most effective technique for each path and avoid fireflies coming different techniques, the various techniques or estimators are weighted using so called multiple importance sampling weights, which were introduced by Veach and Guibas.

These weights ensure only effective techniques for a given path significantly contributes to the image.

6:15

Bidirectional path tracing

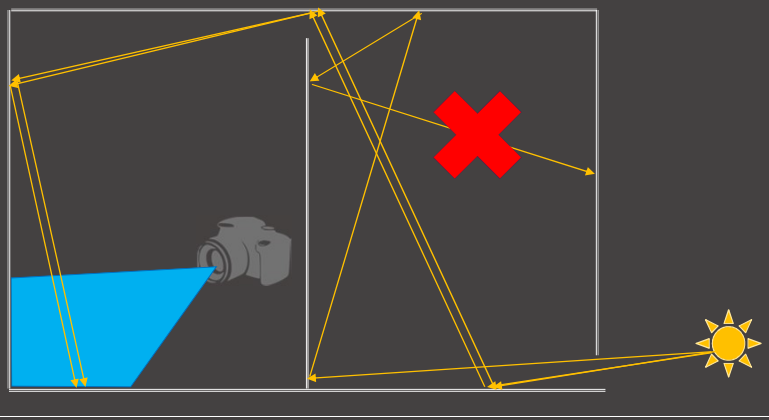


Bidirectional path tracing [Veach and Guibas 1994]

Thanks to this weighting of different combination of paths, bidirectional path tracing is able to efficiently render directly visible caustics without any fakes or prominent fireflies.

Guiding of photon paths – mutation kernel

- Mutation = creates new path from the old one
- Small mutation = correlation artifacts
- Big mutation = ineffective sampling



Another aspect that greatly influences the effiency of Metropolis sampler is the mutation kernel which defines how we create a new path from the existing one.
(CLICK)
In general, small mutations that only slightly change the current path are more effective at exploitation but lead to more correlation artifacts.
(CLICK)
Larger mutations may result in completely different paths, (CLICK)which are less likely to be accepted by Metropolis sampler.

Guiding of photon paths – mutation kernel

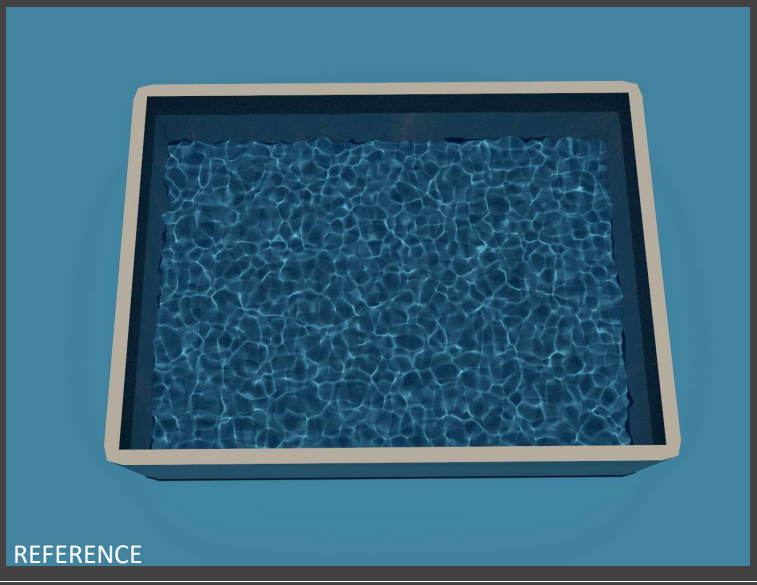
- We utilize adaptive MCMC [Hairo et al. 2001]
- First introduced by [Hachisuka and Jensen 2011]
- We adapt the original primary sample space mutation kernel by [Kelemen et al. 2002]

In our research, we use adaptive Markov chain Monte Carlo that allows for adaptation of the mutation size based on the previous samples.

Such adaptation was first introduced to light transport by Hachisuka and Jensen.

In our research, we follow their example, but we adapt a different kernel, which was originally proposed as non-adaptive by Kelemen et al.

Guiding of photon paths – mutation kernel



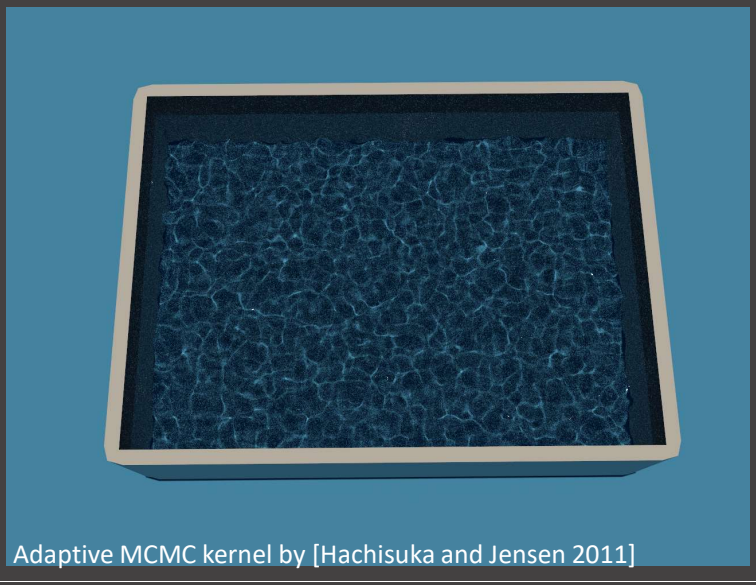
We show the results of the different kernels on this simple scene which features a pool on plane that is 40000 times bigger than the pool itself.

Guiding of photon paths – mutation kernel

MCMC kernel by [Kelemen et al. 2002]

We can see that the non-adaptive kernel by Kelemen et al. fails to effectively exploit the existing paths, since it is unable to select proper mutation size.

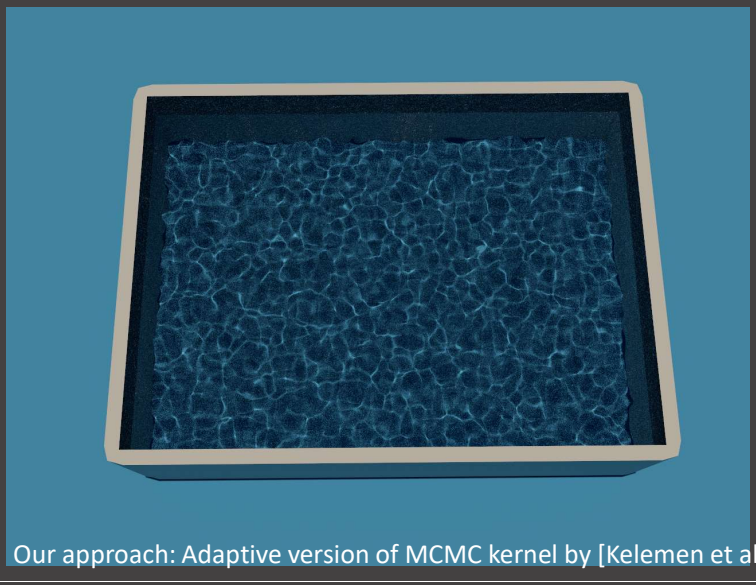
Guiding of photon paths – mutation kernel



Adaptive MCMC kernel by [Hachisuka and Jensen 2011]

Kernel by Hachisuka and Jensen uses the previous samples to adapt mutation size, however the kernel itself is not very efficient.

Guiding of photon paths – mutation kernel



Our approach: Adaptive version of MCMC kernel by [Kelemen et al. 2002]

Finally, our approach, which adapts the original kernel by Kelemen et al. generates a superior result.

30:00

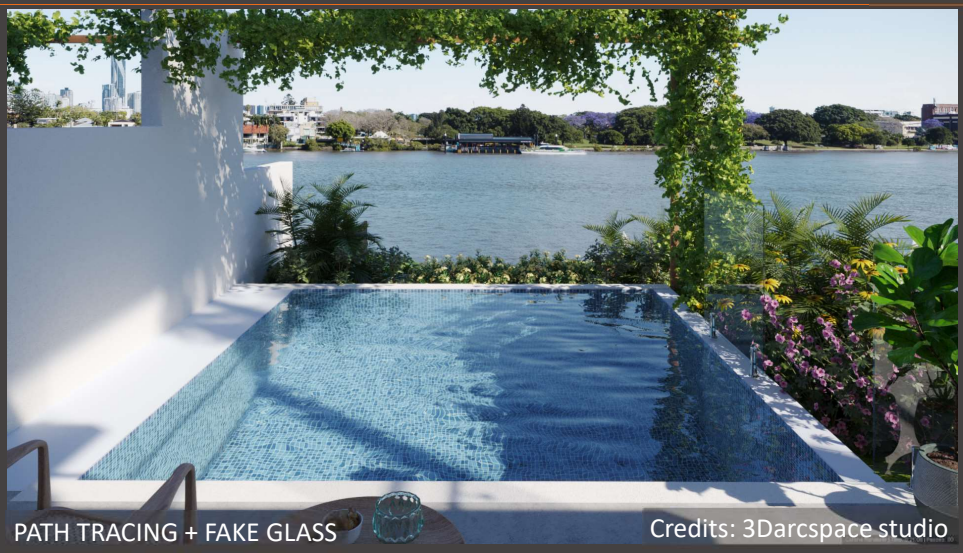
Guiding of photon paths – Metropolis

In this case large portion of the ocean is visible for the camera and thus the whole region is considered to be equally important.

Guiding of photon paths – Metropolis

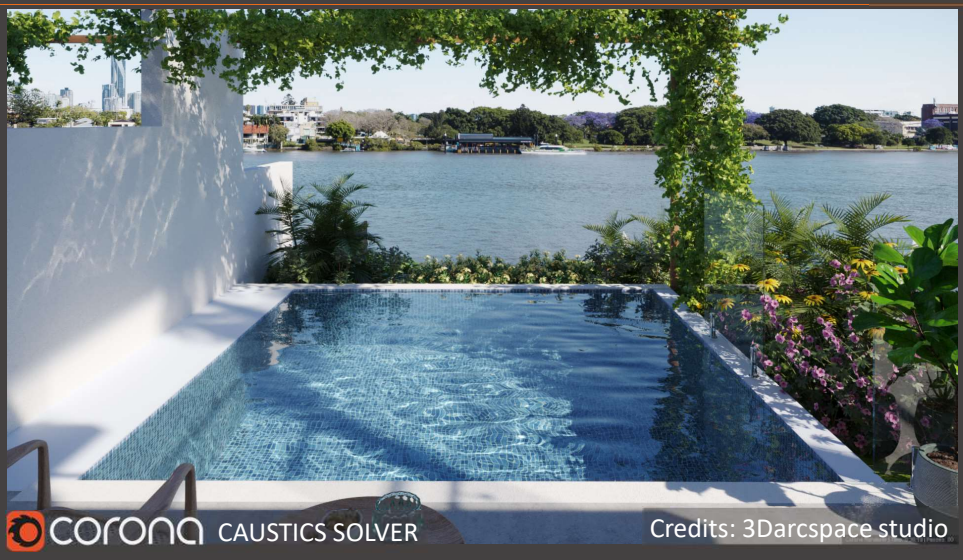
This means the paths closer to the camera will be more and Metropolis will thus generate more photon paths near the camera.

Inefficiency of path tracing – fake glass



And here is another example, where using fakes leads to visibly less realistic result

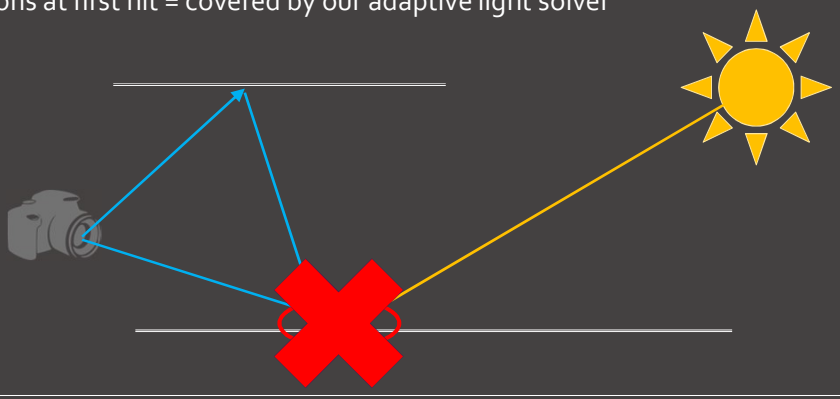
Inefficiency of path tracing – fake glass



compared to the true solution.

Reducing the overhead – storing photons

- Photon lookup => requires storing all photon path vertices (photons)
  Our approach: We further only store photons at certain locations
- No photons at first hit = covered by our adaptive light solver



Note that for photon lookup to work we need to first store all photon path vertices (or photons) from one iteration in memory.
To reduce both memory and computation overhead it is important to store photons only where they are important.
(CLICK)
First, we realize that we don't have to store photons at photon paths' first intersection with the scene (CLICK), since photon lookup in such location would not bring any advantage
compared to path tracing with Corona's adaptive light solver.