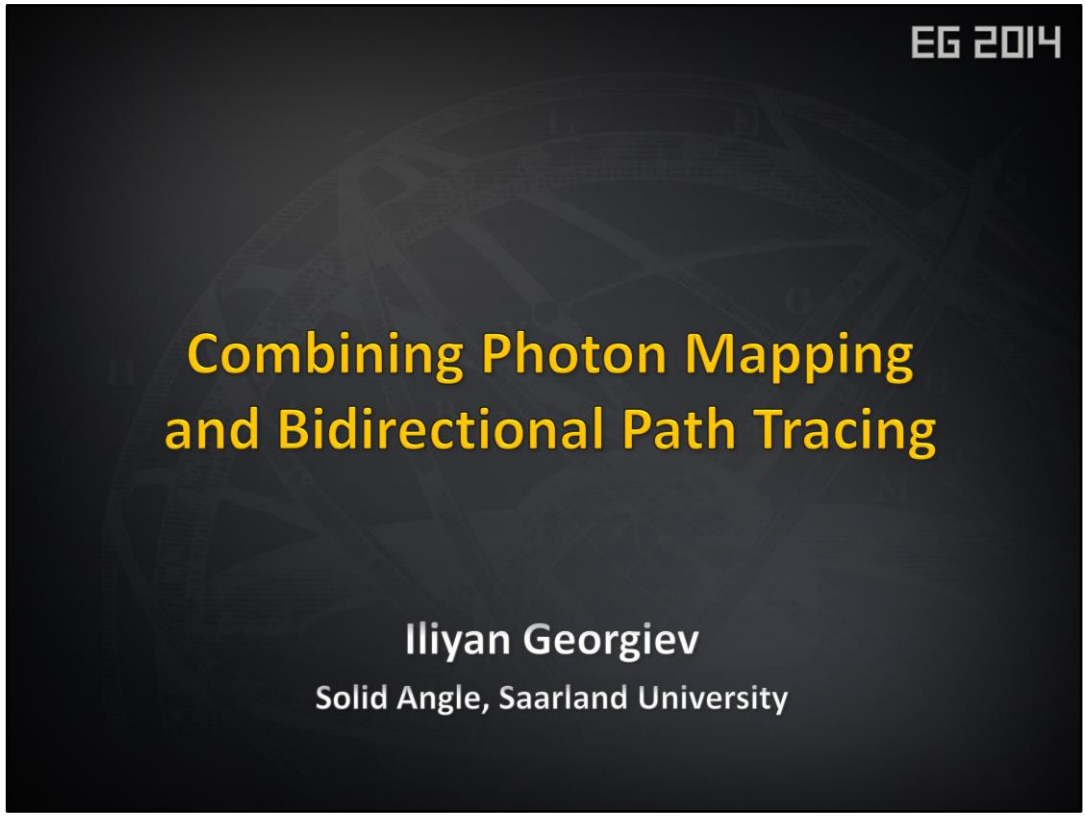
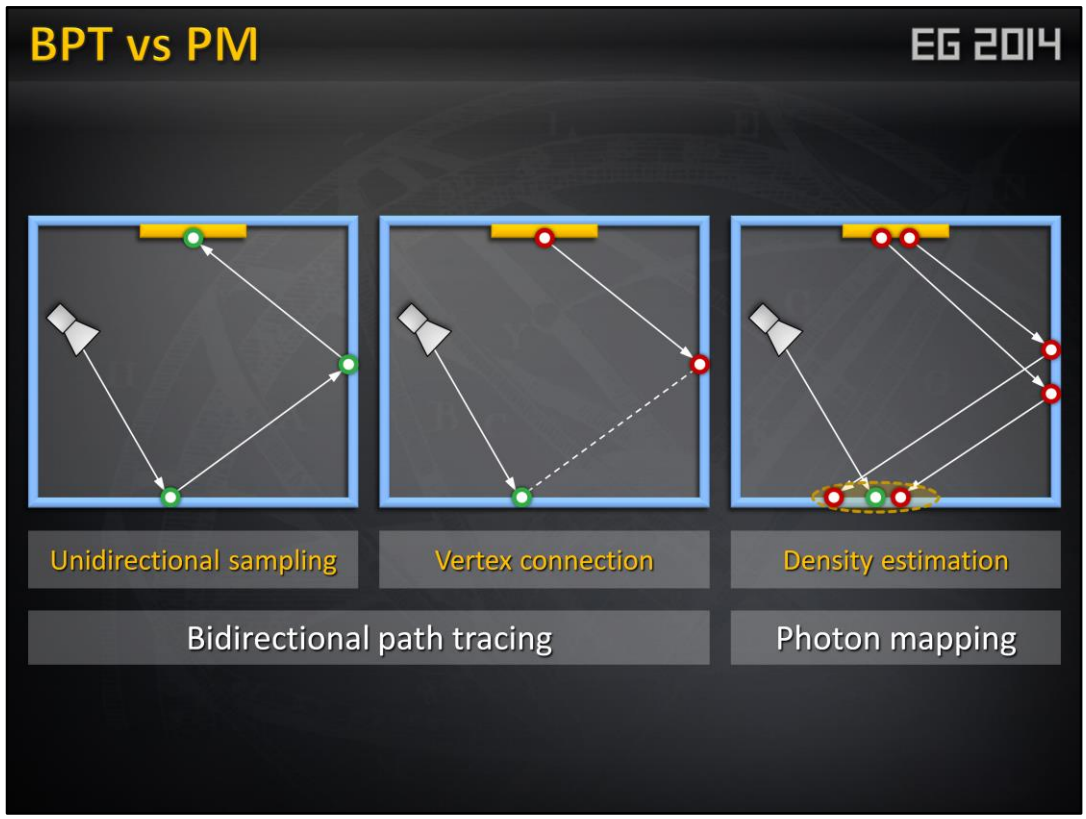


In the previous talk, Jaroslav discussed the path integral formulation of light transport and demonstrated its conceptual simplicity and flexibility. I will now show how we can leverage this framework to seamlessly combine bidirectional path tracing and photon mapping via multiple importance sampling.

Bidirectional path tracing is one of the most versatile light transport simulation algorithms available today. It can robustly handle a wide range of illumination and scene configurations, but is notoriously inefficient for specular-diffuse-specular light interactions, which occur e.g. when a caustic is seen through a reflection/refraction.

On the other hand, photon mapping (PM) is well known for its efficient handling of caustics. Recently, Hachisuka and Jensen [2009] showed a progressive variant of PM that converges to the correct result with a fixed memory footprint. Their stochastic progressive photon mapping (PPM) algorithm captures the reflected caustics in our scene quite well. However, it has hard time handling the strong distant indirect illumination coming from the part of the scene behind the camera.

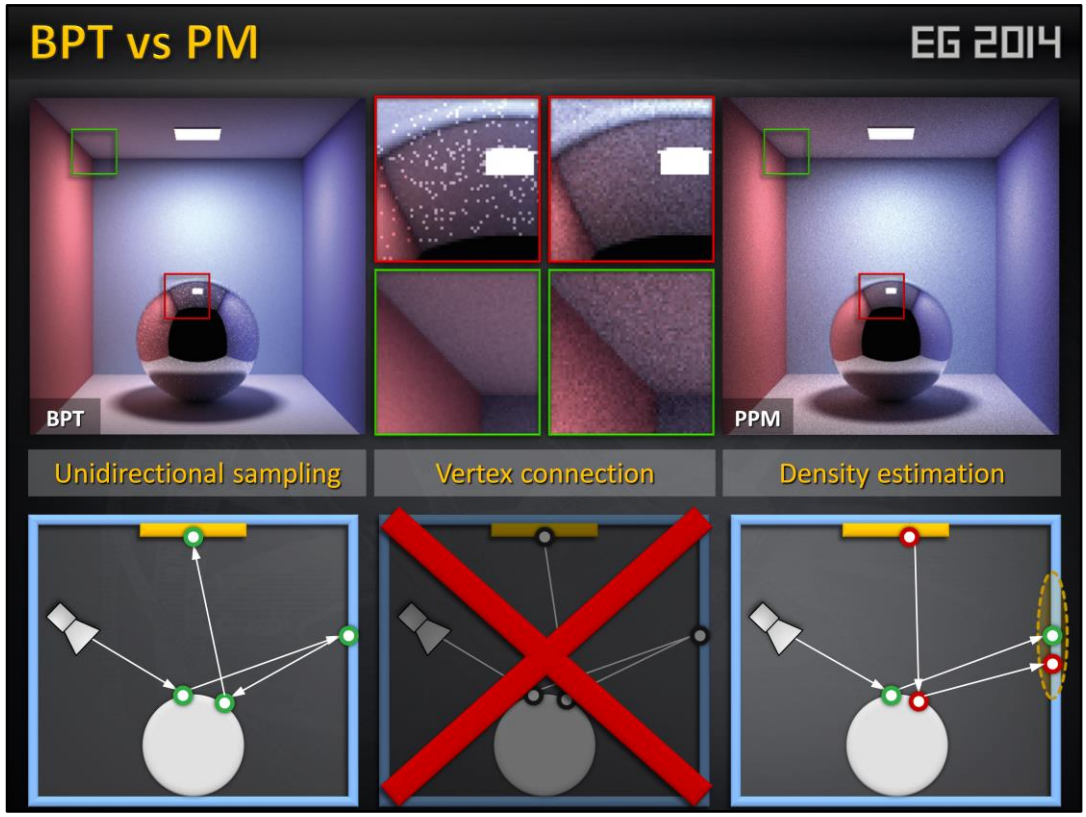
By using multiple importance sampling to combine estimators from bidirectional path tracing and photon mapping, the algorithm I will talk about today automatically finds a good mixture of techniques for each individual light transport path, and produces a clean image in the same amount of time.



Let us start by reviewing how bidirectional path tracing (BPT) and photon mapping (PM) sample light transport paths that connect the light sources to the camera:

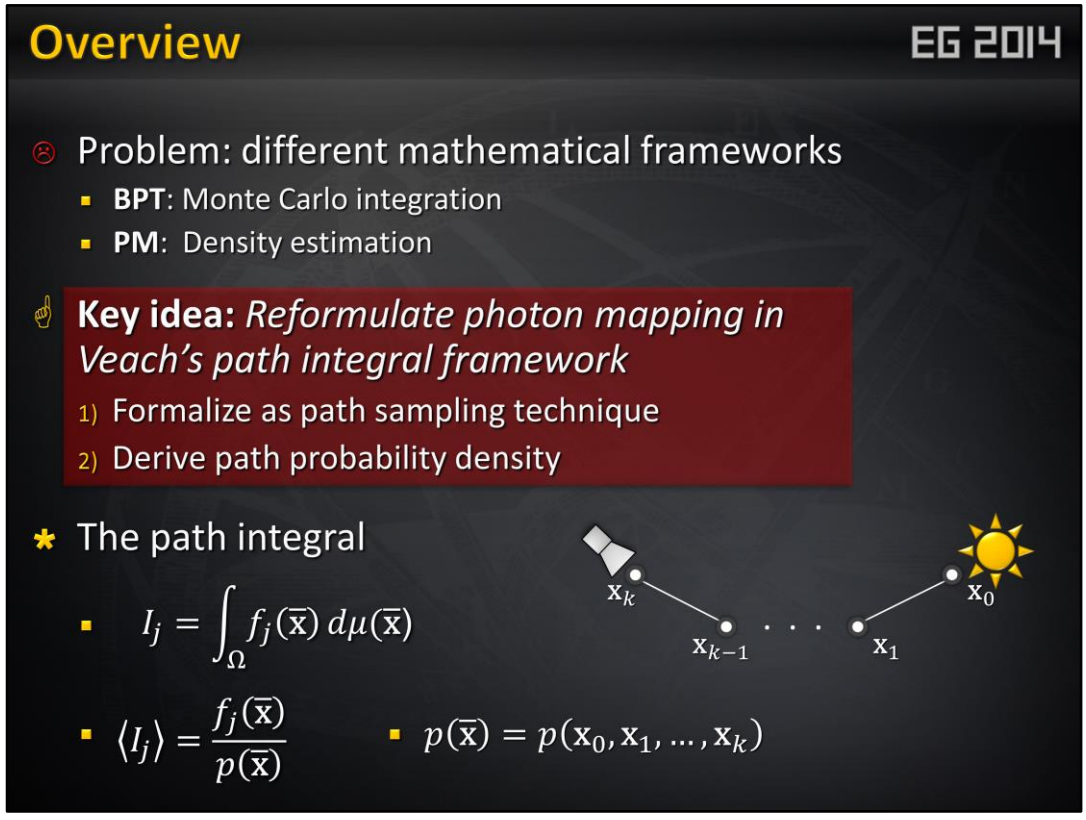
The techniques BPT employs can be roughly categorized to *unidirectional sampling* (US) and *vertex connection* (VC). US constructs a path by starting either from a light source or the camera and tracing a random walk in the scene until termination. On the other hand, VC traces one subpath from a light source and another one from the camera, and then completes a full path by connects their endpoints.

In contrast, PM first traces a number of light subpaths and stores their vertices, a.k.a. photons. It then traces subpaths from the camera and computes the outgoing radiance at the hit points using density estimation by looking up nearby photons.



BPT can efficiently capture directly visible caustics, as it can connect light subpath vertices to the camera. However, for sampling specular-diffuse-specular paths, BPT can only rely on unidirectional sampling, as VC cannot perform connections with specular vertices. Since the probability of randomly hitting the light source is often very low, the resulting images suffer from excessive noise.

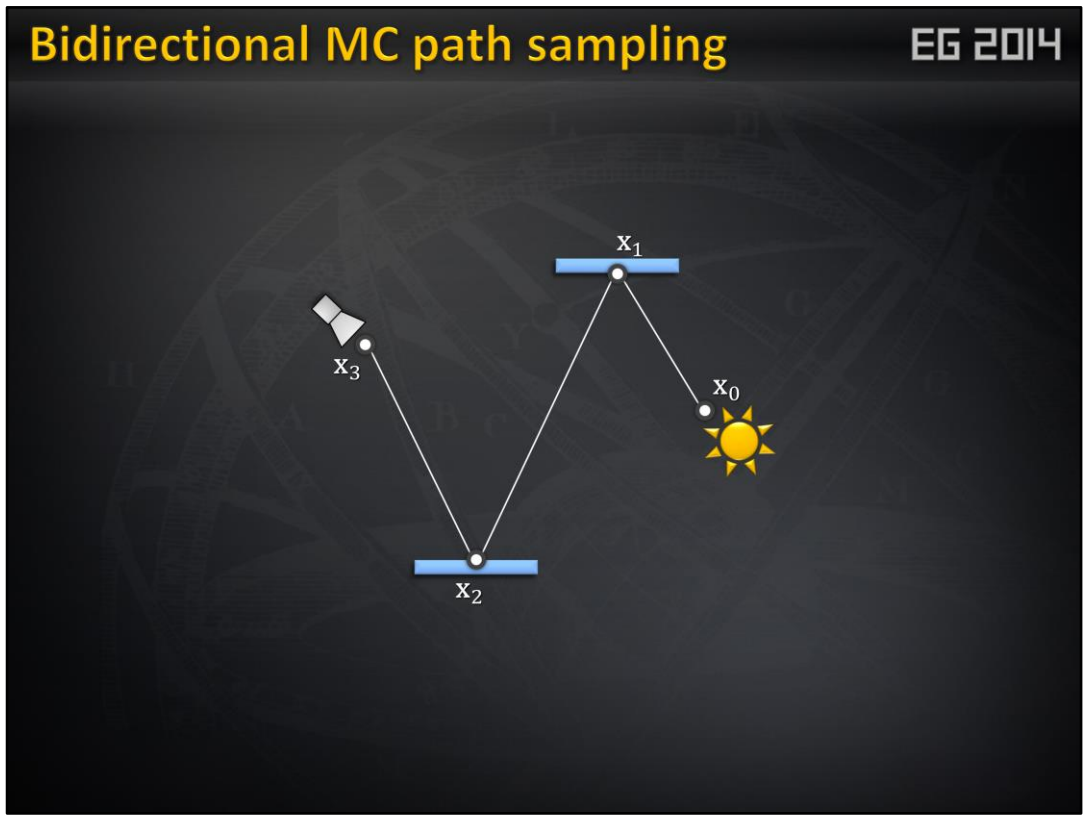
On the other hand, photon mapping handles both direct and reflected caustics pretty much the same way – by loosely connecting nearby eye and light sub-path vertices. But as can be seen in the images above, it is less efficient than BPT for diffuse illumination.



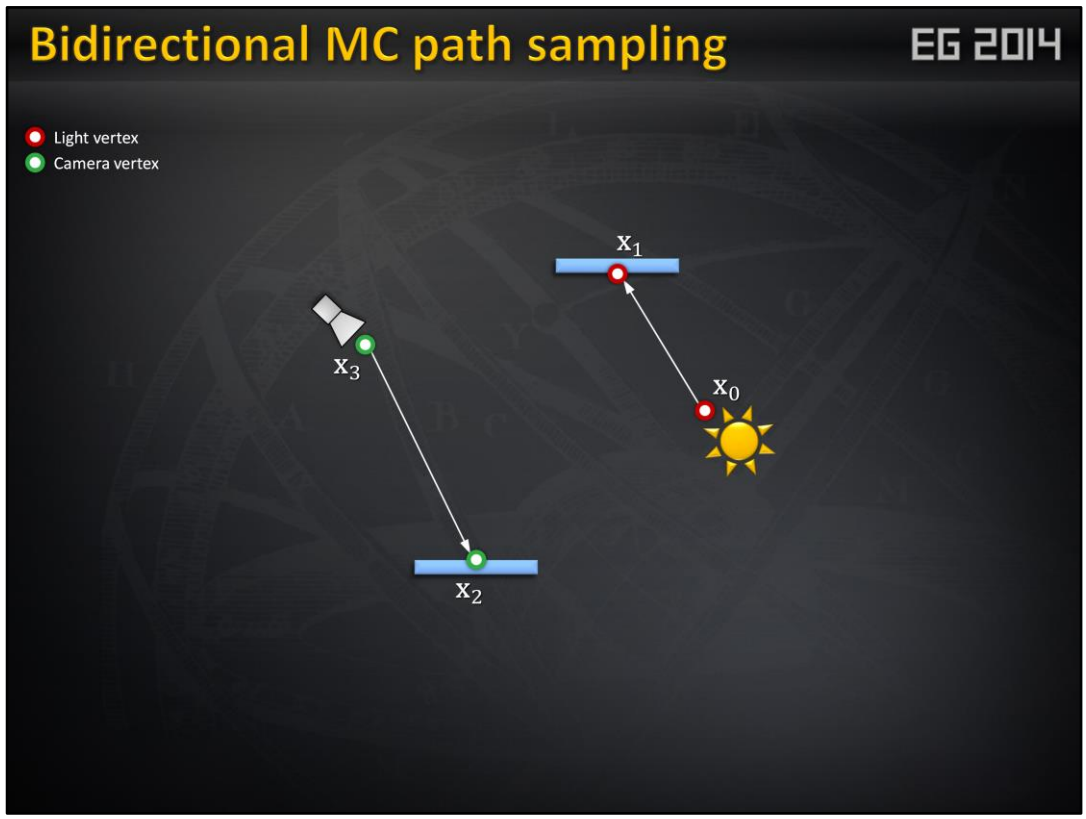
It has been long recognized that bidirectional path tracing (BPT) and photon mapping (PM) complement each other in terms of the light transport effects they can efficiently handle. However, even though both methods have been published more than 15 years ago, neither a rigorous analysis of their relative performance nor an efficient combination had been shown until very recently. The reason for this is that BPT and PM have originally been defined in different theoretical frameworks – BPT as a standard Monte Carlo estimator to the path integral, and PM as an outgoing radiance estimator based on photon density estimation.

The first step toward combining these two methods is to put them in the same mathematical framework. We choose Veach's path integral formulation of light transport, as it has a number of good properties (which Jaroslav discussed) and also because BPT is already naturally defined in this framework.

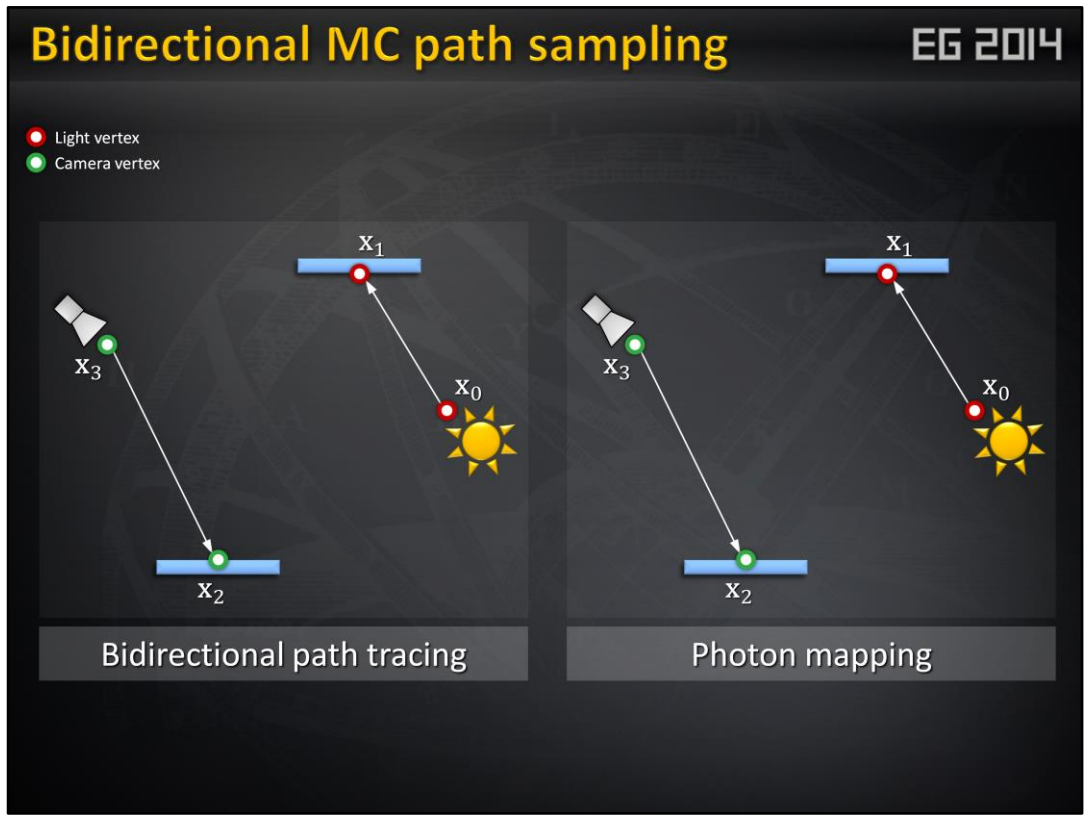
We need two key ingredients: (1) express PM as a sampling technique that constructs light transport paths that connect the light sources to the camera, and (2) derive the probability densities for paths sampled with this technique. This will give us a basis for reasoning about the relative efficiency of BPT and PM. And more importantly, it will lay the ground for combining their corresponding estimators via multiple importance sampling.



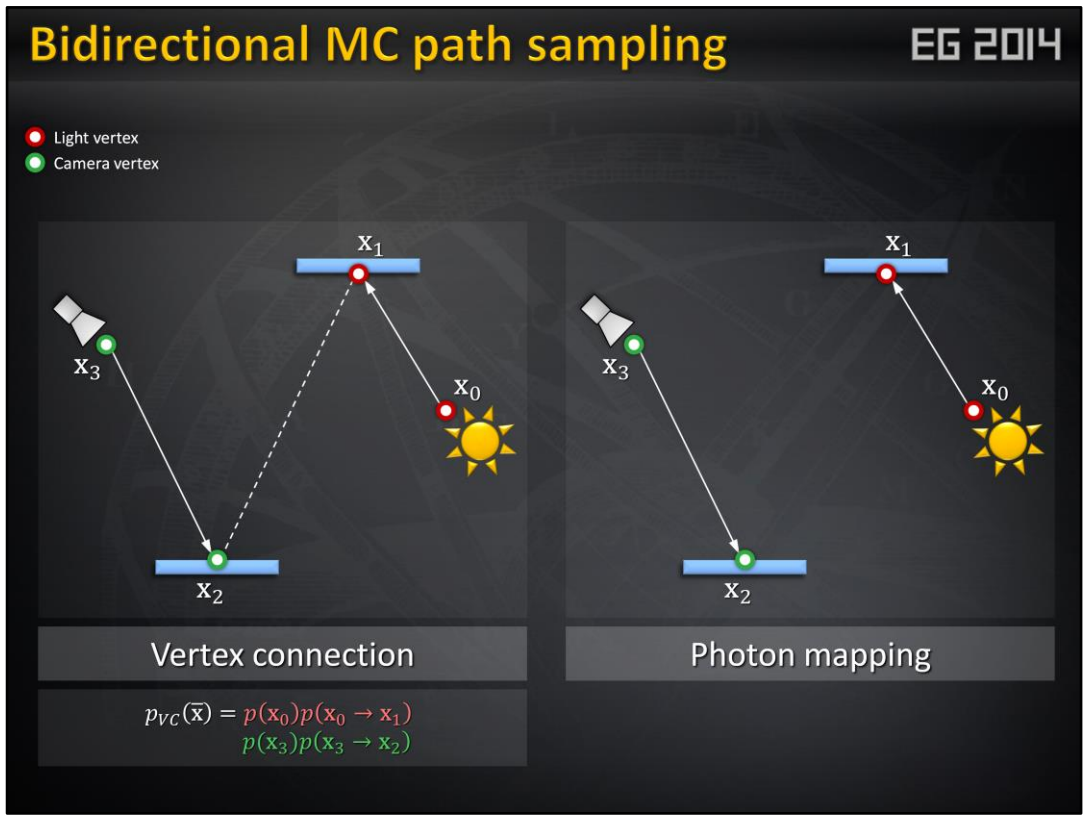
Let us start by taking a simple length-3 path and see how it can be constructed bidirectionally.



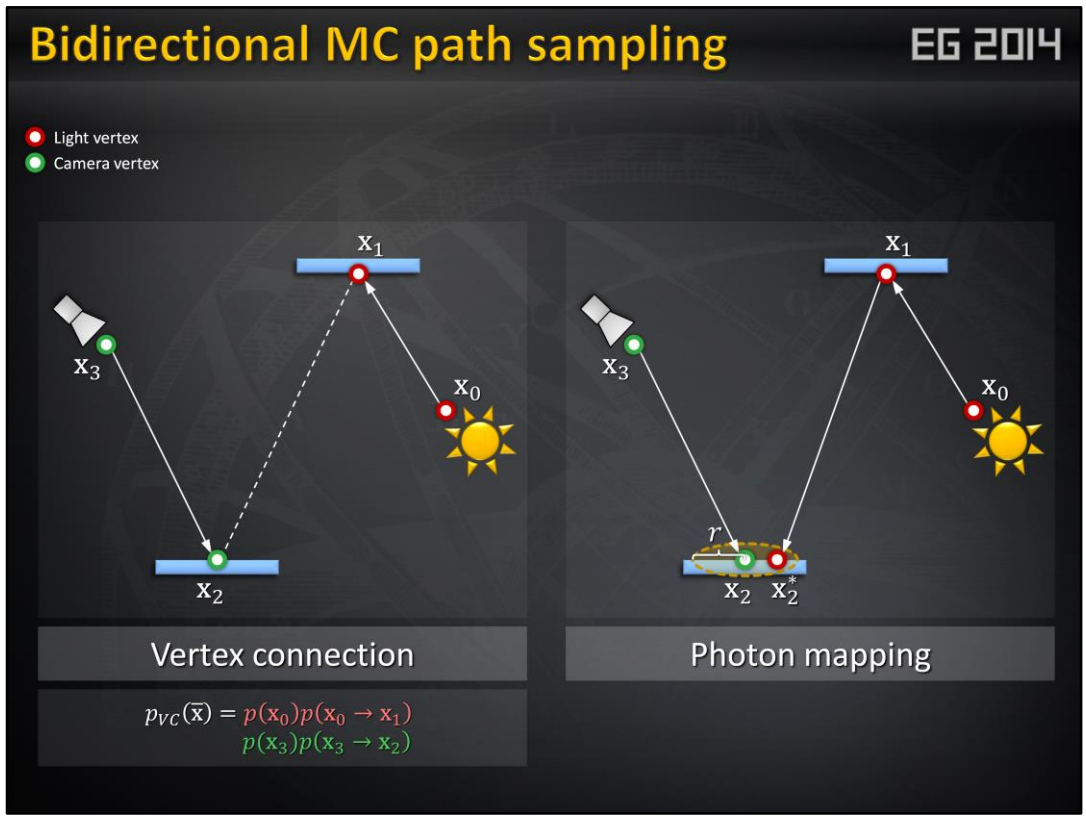
We first trace one subpath from the camera and another one from a light source.



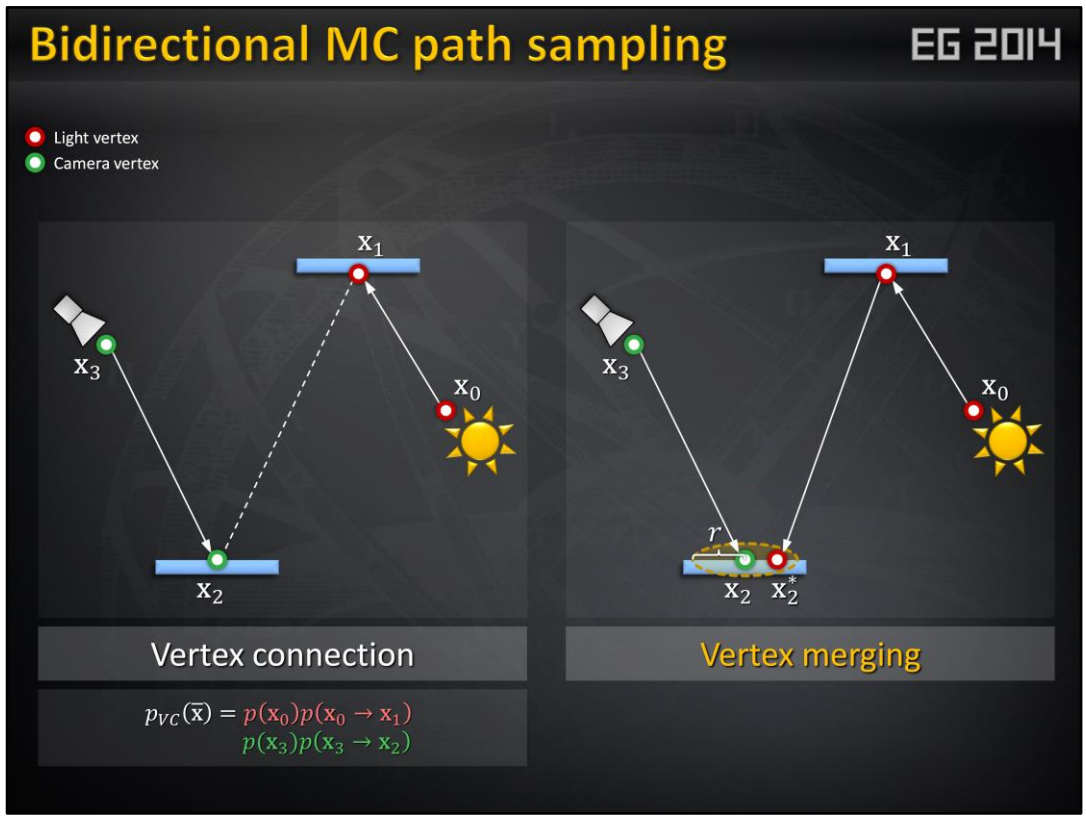
Now let's see how we complete a full path in BPT and PM.



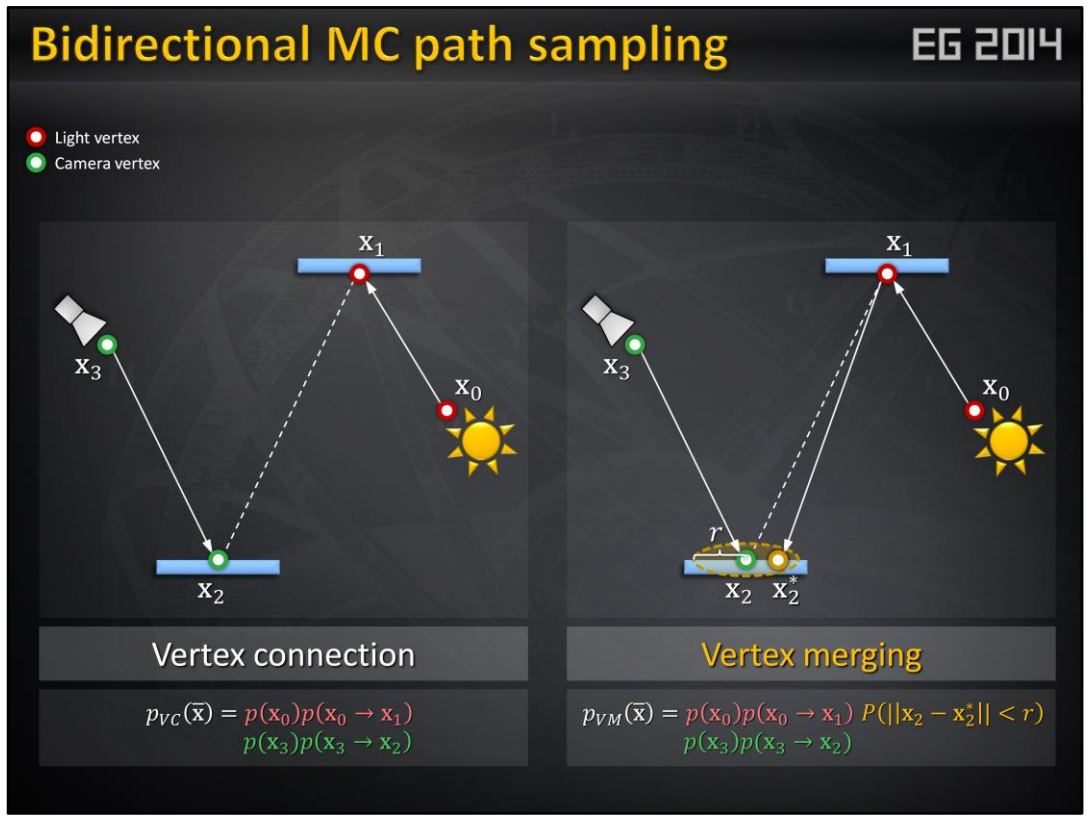
Bidirectional path tracing connects the subpath endpoints deterministically. We call this technique *vertex connection*. The PDF of the resulting full path is well known, and is simply the product of the PDFs of two subpaths, which have been sampled independently.



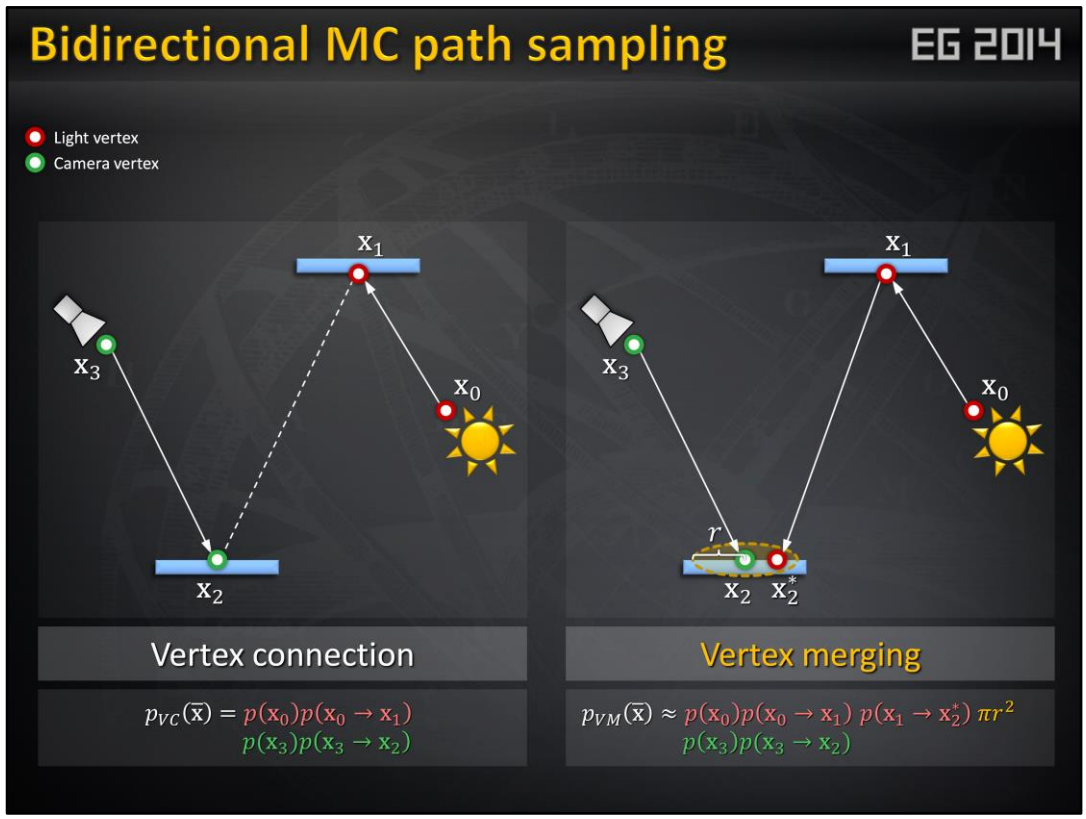
Photon mapping, on the other hand, will extend the light subpath by sampling one more vertex from $\mathbf{x}_1$, and will concatenate the two subpaths only if the "photon" hit-point $\mathbf{x}_2^*$ lies within a distance r from $\mathbf{x}_2$.



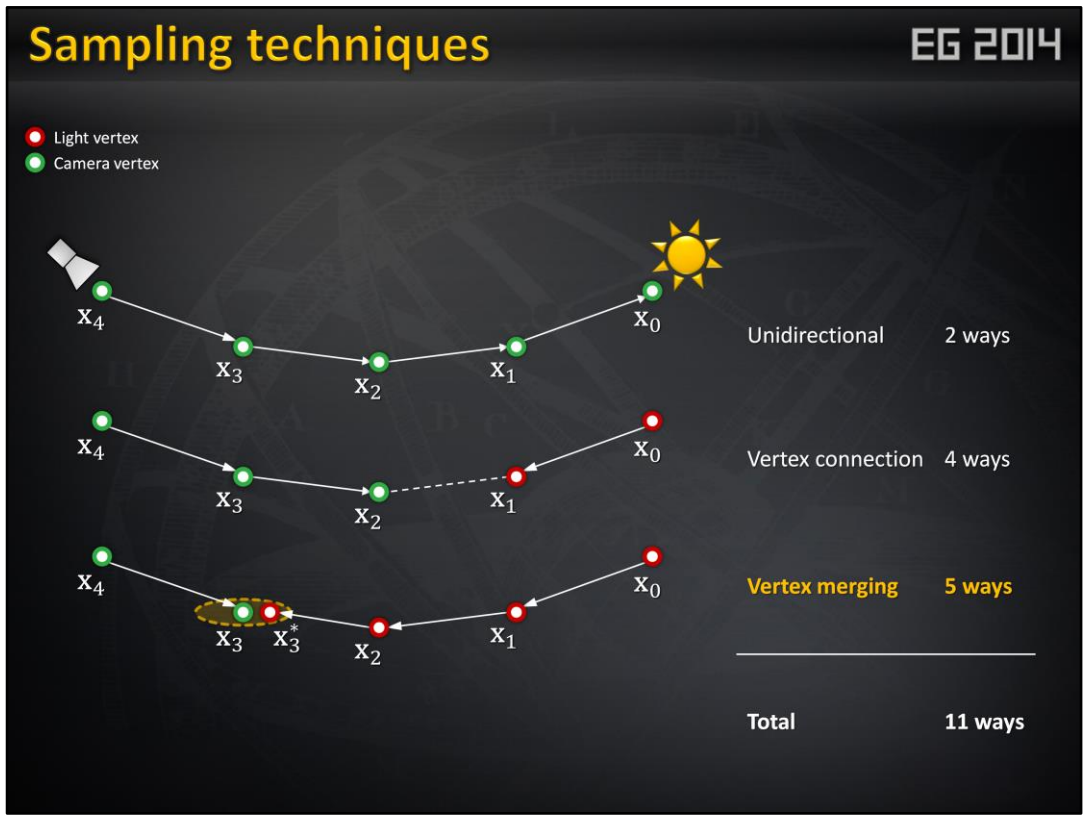
We label this technique *vertex merging*, as it can be intuitively thought to weld the endpoints of the two subpaths if they lie close to each other.



What remains is to derive the PDF of the resulting full path. To do this, we can interpret the last step as establishing a regular vertex connection between $\mathbf{x}_1$ and $\mathbf{x}_2$, but conditioning its acceptance on the random event that a vertex $\mathbf{x}_2^*$ sampled from $\mathbf{x}_1$ lands within a distance r to $\mathbf{x}_2$. This probabilistic acceptance is nothing more than a Russian roulette decision. The full path PDF is then again the product of the subpath PDFs, but in addition multiplied by the probability of sampling the point $\mathbf{x}_2^*$ within a distance r of $\mathbf{x}_2$. This acceptance probability is equal to the integral of the PDF of $\mathbf{x}_2^*$ over the r-neighborhood of $\mathbf{x}_1$.

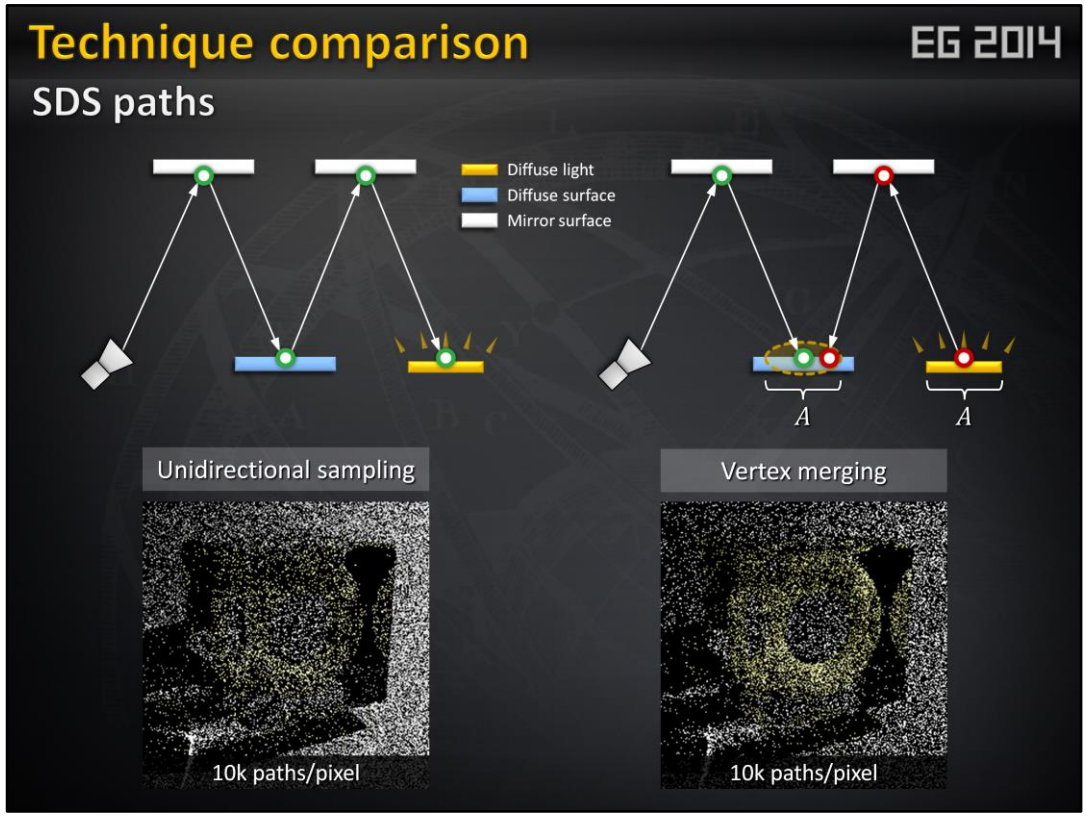


Under the reasonable assumptions that the surface around $\mathbf{x}_1$ is locally flat, i.e. that this neighborhood is a disk, and that the density of $\mathbf{x}_2^*$ is constant inside this disc, the integral can be well approximated by the PDF of the actual point $\mathbf{x}_2^*$ we have sampled, multiplied by the disc area πr^2.



Now that we have formulated the vertex merging path sampling technique, we can put it side by side with the already available techniques in BPT. There are two ways to sample a length-4 path unidirectionally, and four ways to sample it via vertex connection. Vertex merging adds five new ways to sample the path, corresponding to merging at the five individual path vertices. In practice, we can avoid merging at the light source and the camera, as directly evaluating emission and sensitivity is usually cheap.

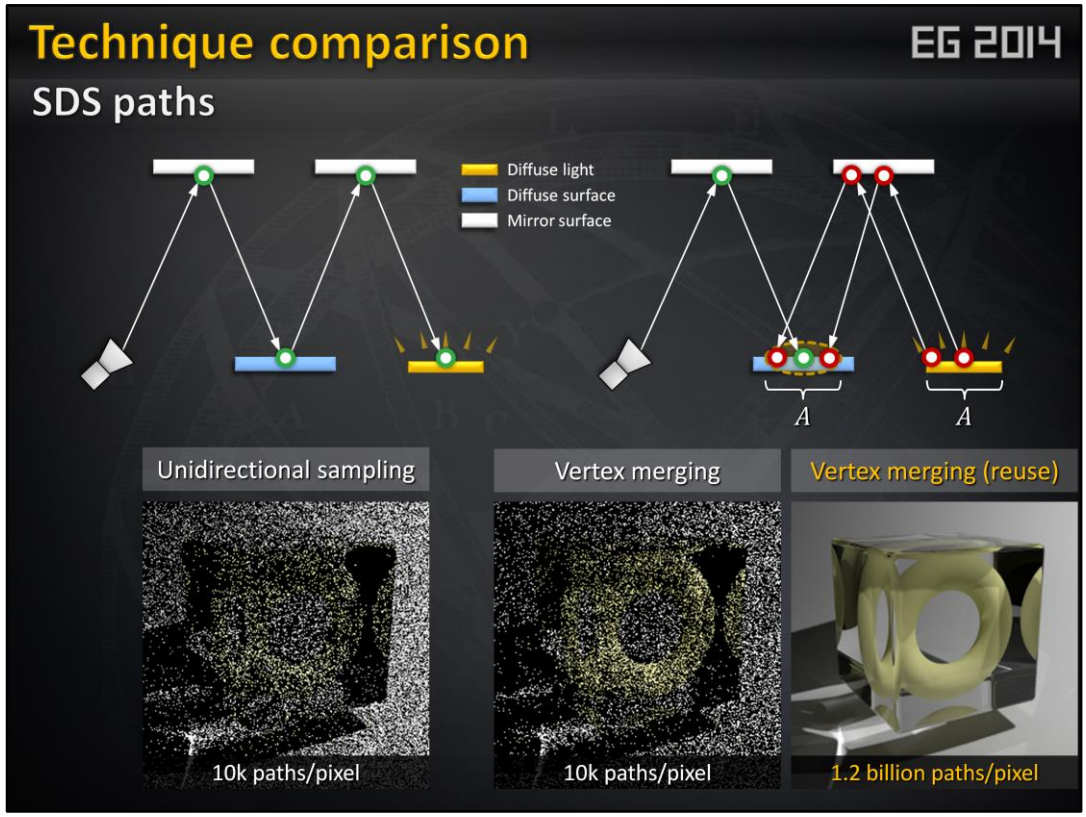
But with so many ways to sample the same light transport path, a question naturally arises in the mind of the curious: which technique is the most efficient for what types of paths?



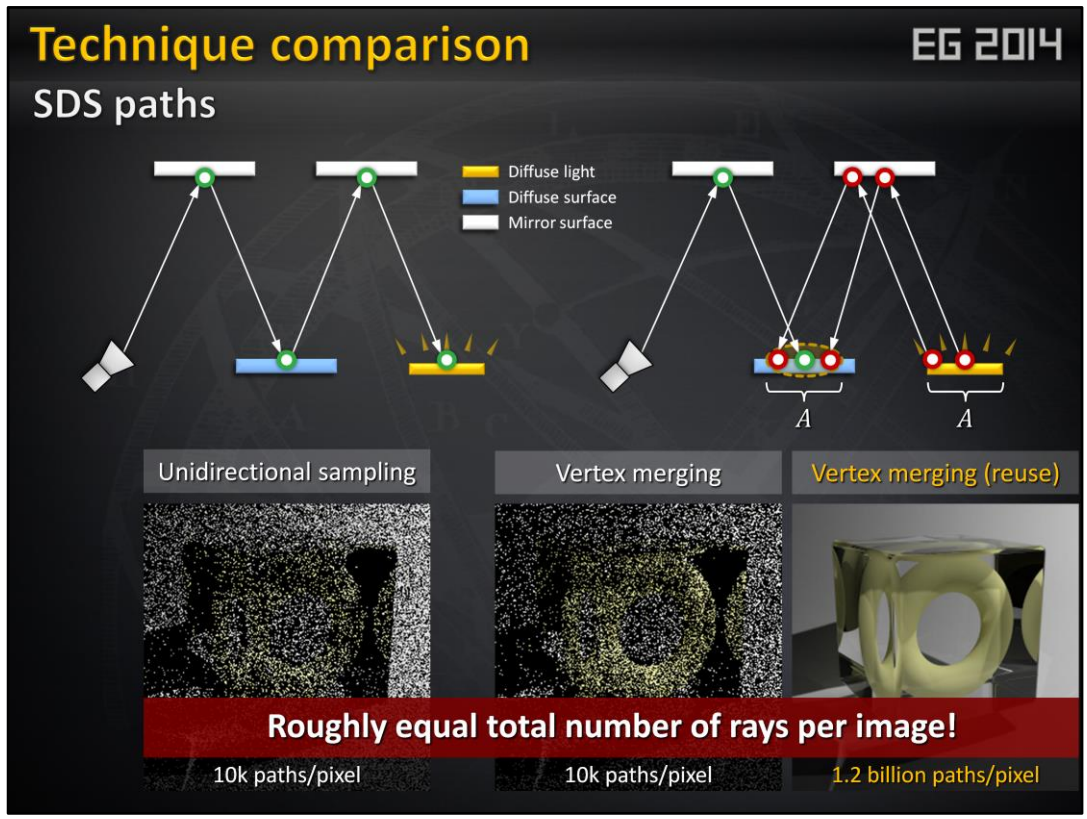
To answer this question, let us first take a look at specular-diffuse-specular (SDS) paths. Here, bidirectional path tracing can only rely on unidirectional sampling: it traces a path from the camera hoping to randomly hit the light source. With vertex merging, we can trace one light and one camera subpath, and merge their endpoints on the diffuse surface.

It can be shown that if the light source and the merging disk have the same area A, then unidirectional sampling and vertex merging sample paths with roughly the same probability density. This means that we should expect the two techniques to perform similarly in terms of rendering quality.

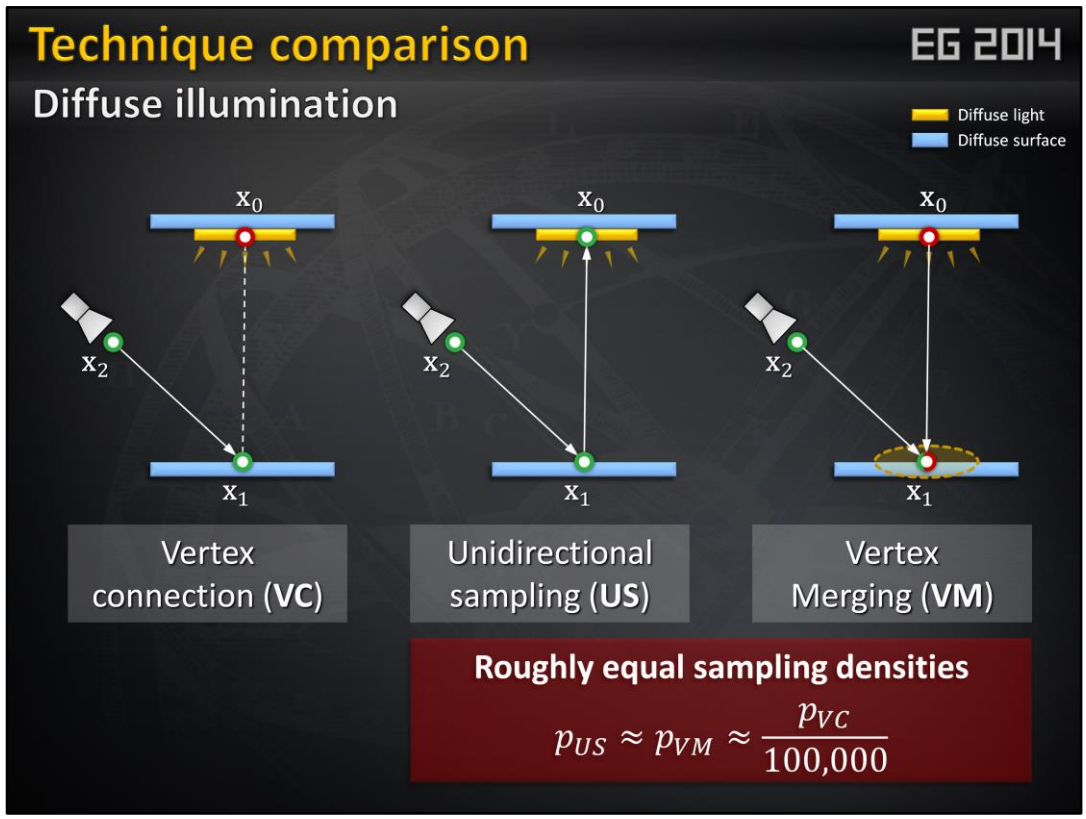
We render these two images progressively, sampling one full path per pixel per iteration. For the left image we trace paths from the camera until they hit the light. For image on the right, we trace subpaths from both ends, and merge their endpoints if they lie within a distance $r = \sqrt{A/\pi}$ from each other. Both images look equally noisy, even after sampling 10,000 paths per pixel. This confirms that vertex merging, and thus photon mapping, is *not* an intrinsically more robust sampling technique for SDS paths than unidirectional sampling.



However, the strength of vertex merging is computational efficiency – we can very efficiently reuse the light subpaths traced for *all* pixels at the cost of a single range search query. This allows us to quickly construct orders of magnitude more light transport estimators from the same sampling data, with a minimal computational overhead, resulting in a substantial quality improvement.



For all these three images we have traced roughly the same number of rays, and the only difference between the one in the center and the one on the right is that the for right image we have enabled path reuse, by storing, and looking up, the light subpath vertices in a photon map at every rendering iteration.

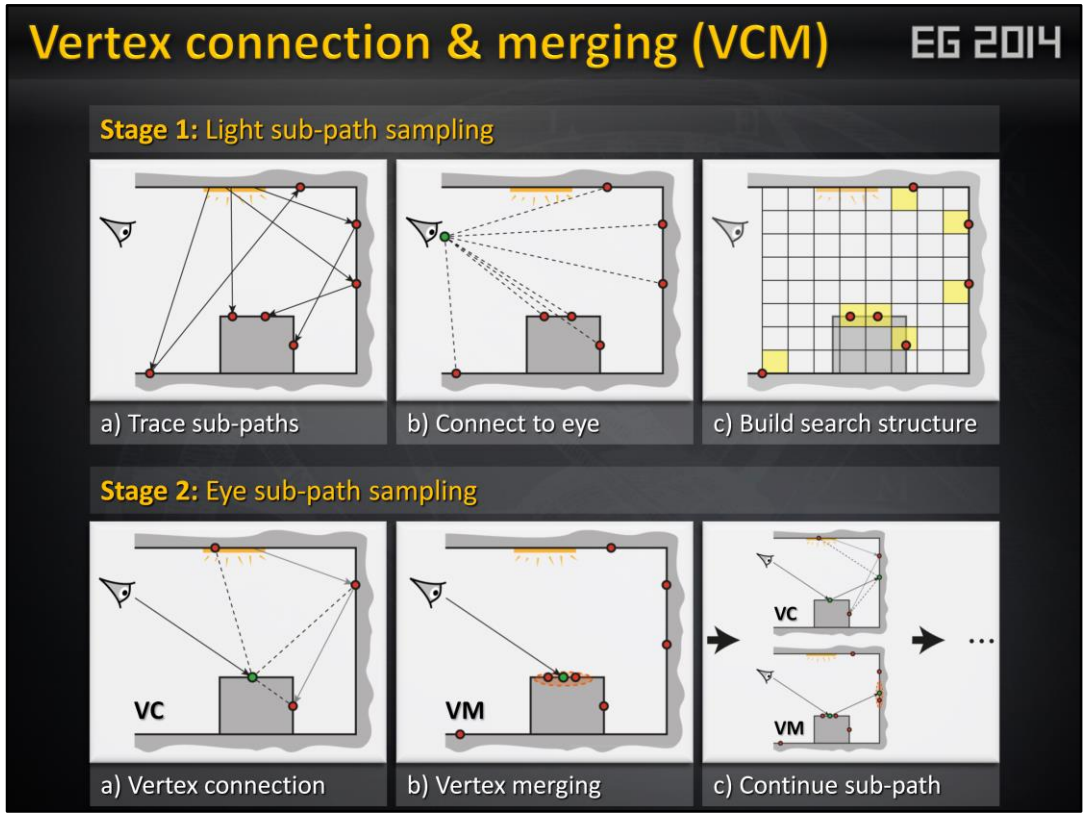


Now let's look at another extreme example – diffuse illumination. Note that vertex connection (VC) constructs the edge between $\mathbf{x}_1$ and $\mathbf{x}_2$ deterministically, while unidirectional sampling (US) and vertex merging (VM) both rely on random sampling.

Once again, it can be shown that if the light source and the merging disk have the same area, then US and VM sample this path with roughly the same probability density.

For the specific case shown on this slide, this density is about 100,000 lower than that of VC. This demonstrates that VM is not an intrinsically more robust sampling technique than VC either. This is not surprising – if we recall the expression for the VM path PDF, we see that it can only be lower than that of the corresponding VC technique, as their only difference is the probability factor in the VM PDF, which is necessarily in the range $[0; 1]$. Still, by reusing paths across pixels, vertex merging, and thus photon mapping, gains a lot of efficiency over unidirectional sampling.

All these useful insights emerge from the reformulation of photon mapping as a path sampling technique.



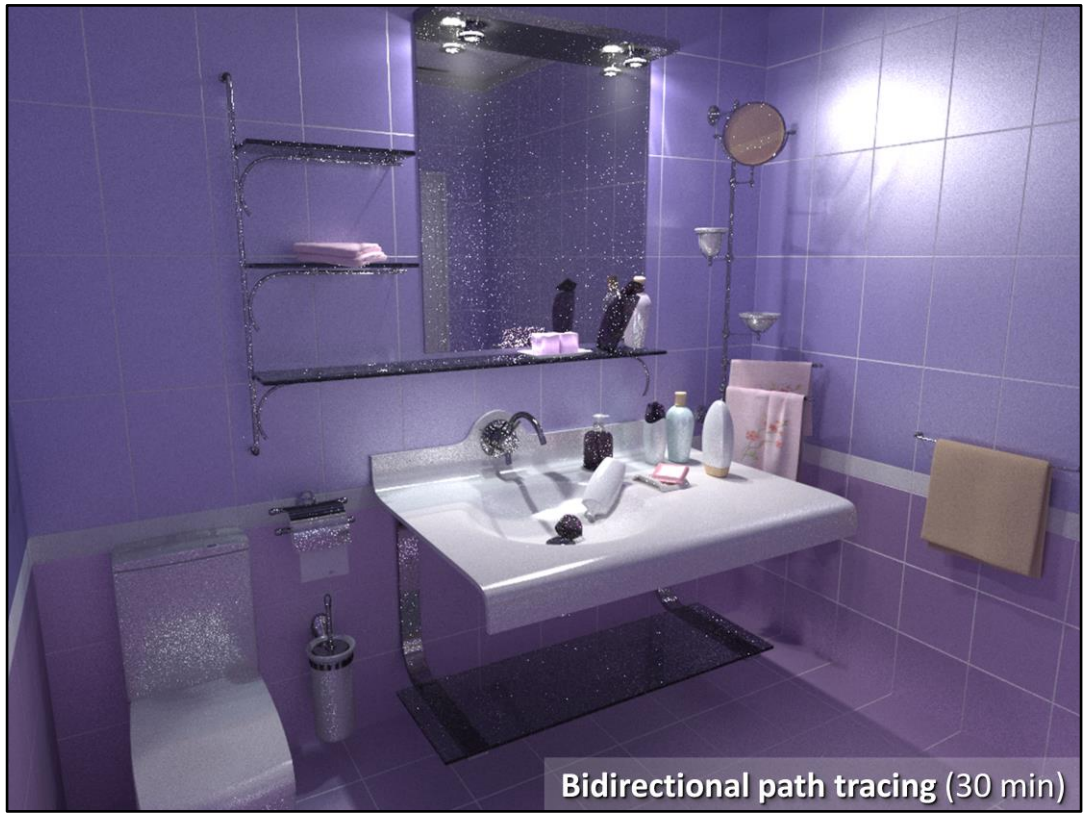
Even more usefully, we now have the necessary ingredients for combining photon mapping and bidirectional path tracing into one unified algorithm. The vertex merging path PDFs tell us how to weight all sampling techniques in multiple importance sampling, and the insights from the previous two slides command to strive for path reuse.

The combined algorithm, which we call *vertex connection and merging* (VCM), operates in two stages.

1. In the first stage, we
    a) trace the light subpaths for all pixels,
    b) connect them to the camera, and
    c) store them in a range search acceleration data structure (e.g. a kd-tree or a hashed grid).

2. In the second stage, we trace a camera subpath for every pixel.
    a) Each sampled vertex on this path is connected to a light source (a.k.a. next event estimation), connected to the vertices of the light subpath corresponding to that pixel, and
    b) merged with the vertices of *all* light subpaths.
    c) We then sample the next vertex and do the same.

In a progressive rendering setup, we perform these steps at each rendering iteration, progressively reducing the vertex merging radius . For details on this, please refer to the cited papers below for details.



Let us now see how this combined algorithm stacks up against bidirectional path tracing and stochastic progressive photon mapping on a number of scenes with complex illumination.


Stochastic progressive photon mapping (30 min)


Vertex connection and merging (30 min)

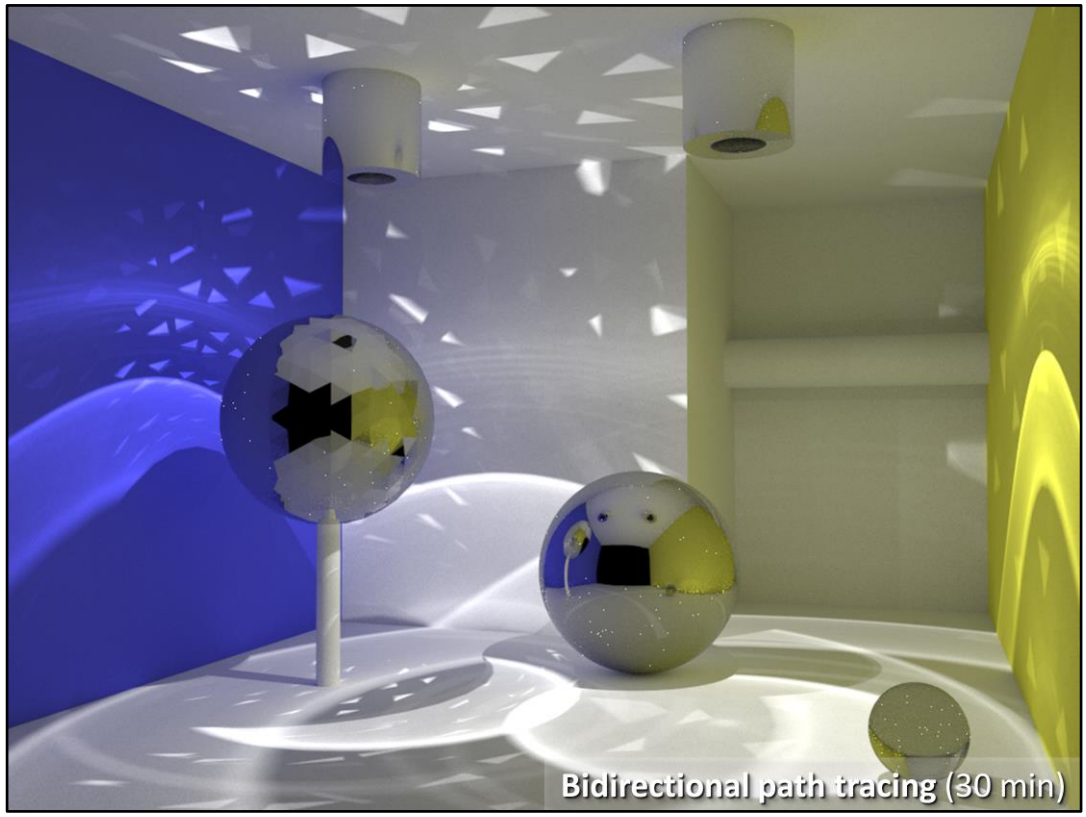
Here, we visualize the relative contributions of VM and VC techniques to the VCM image from the previous slide. This directly corresponds to the weights that VCM assigned to these techniques.

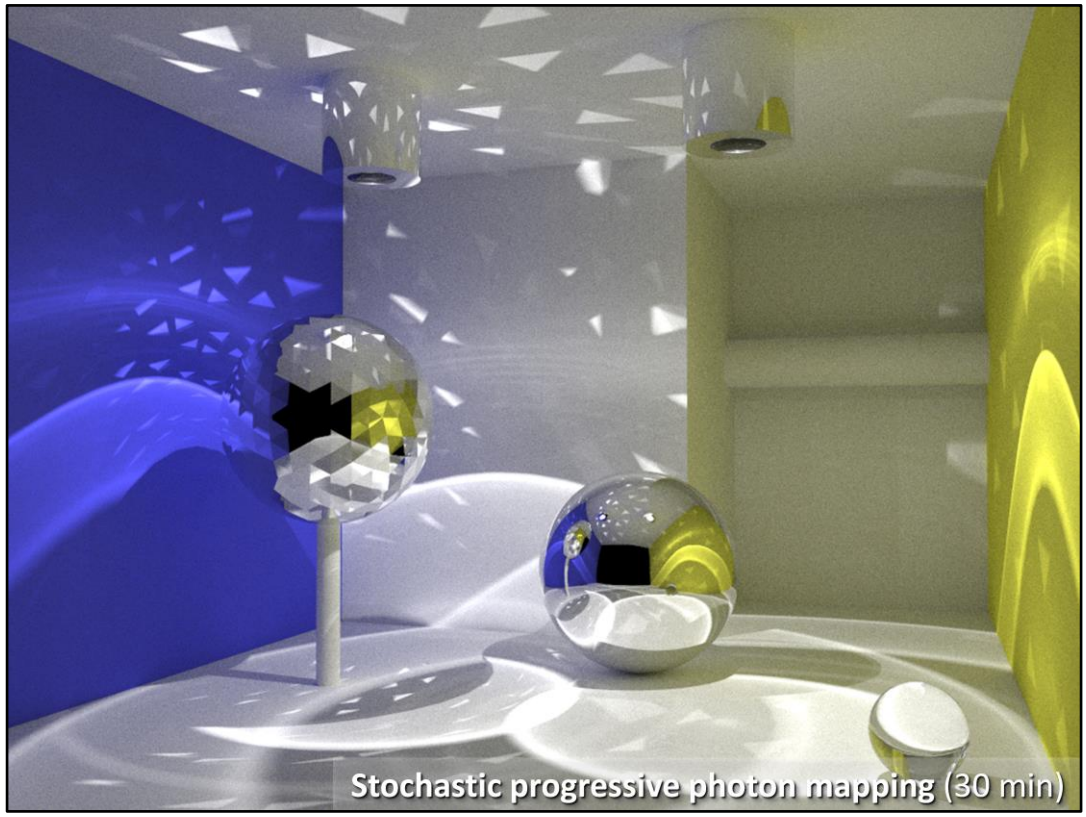

AA-340
Bidirectional path tracing (30 min)

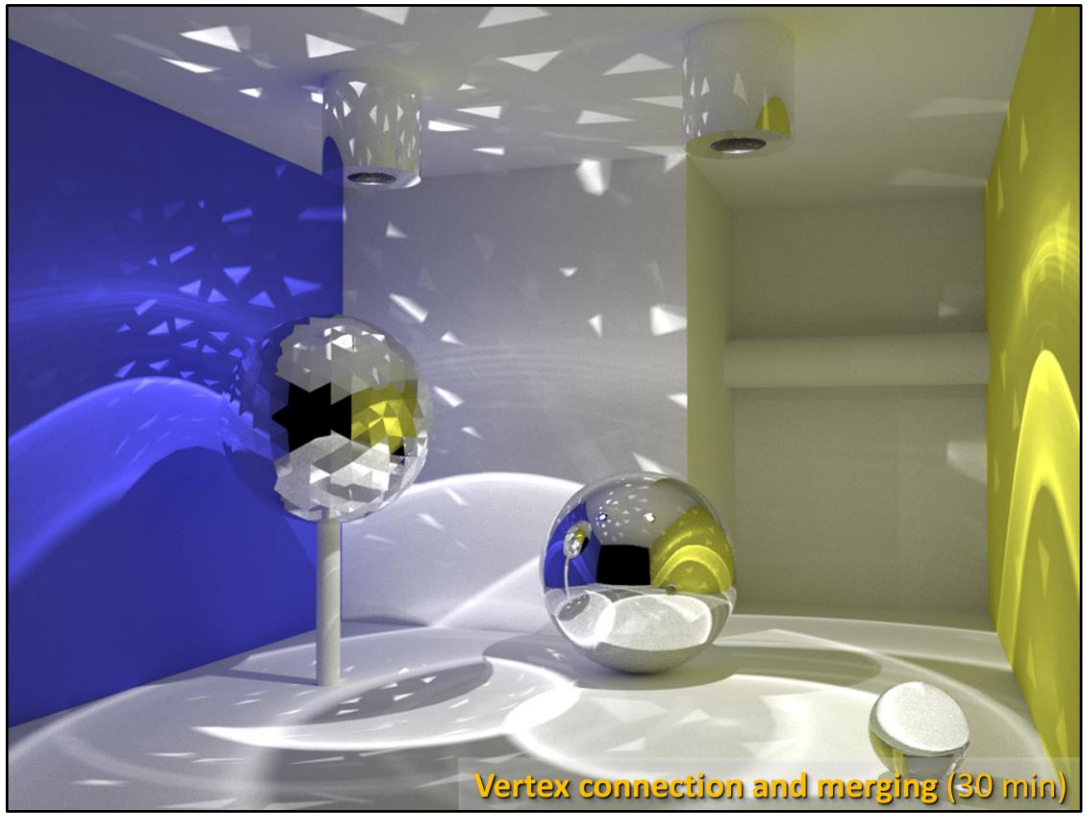

AA-340
Stochastic progressive photon mapping (30 min)

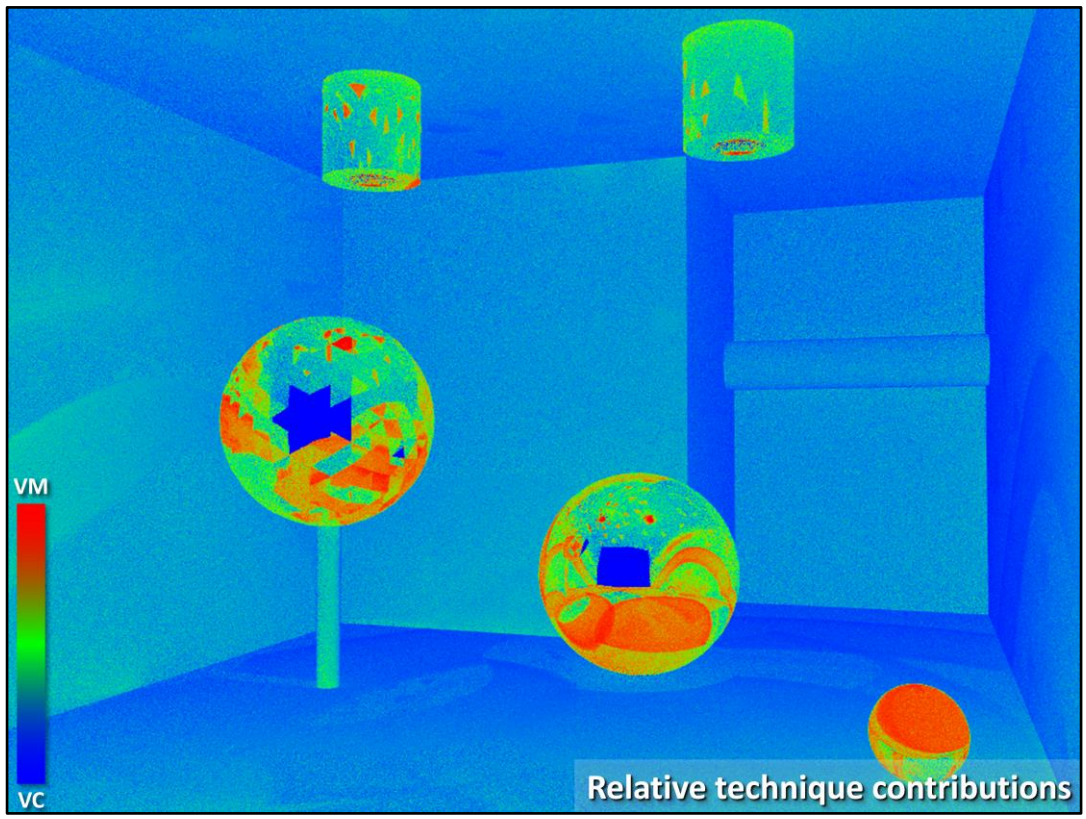

AA-340
Vertex connection and merging (30 min)

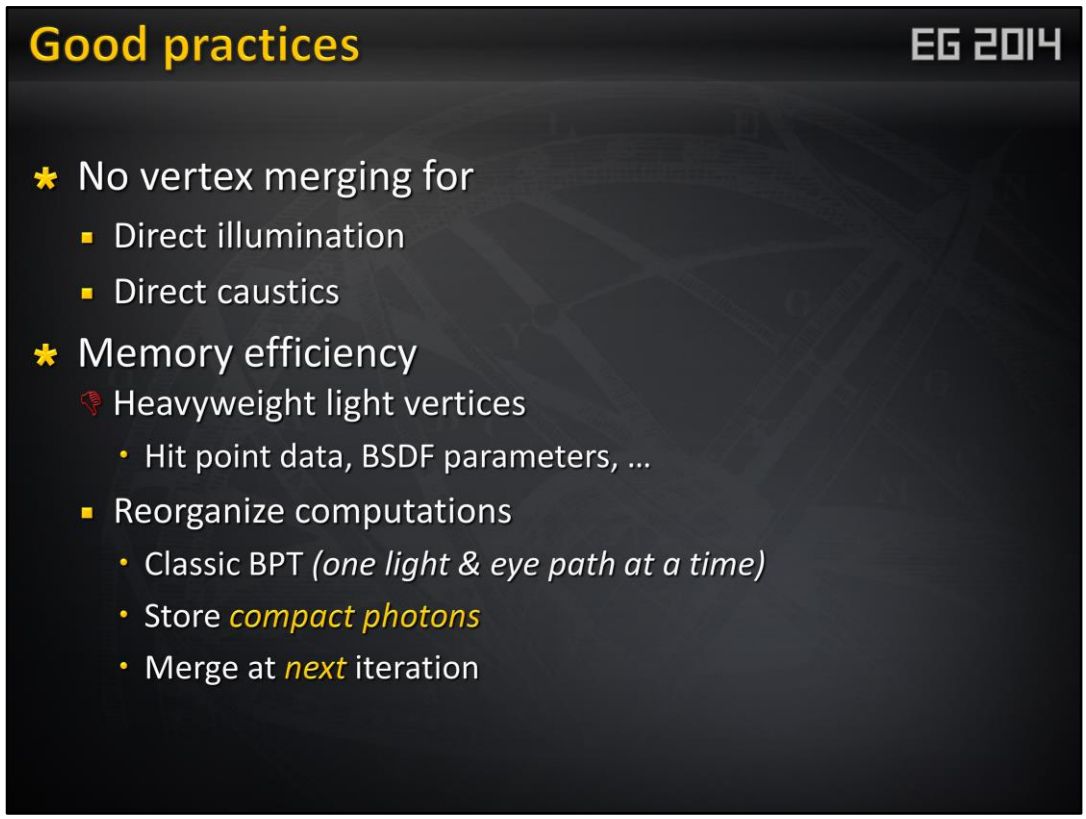

VM
VC
AA-340
Relative technique contributions


Bidirectional path tracing (30 min)


Stochastic progressive photon mapping (30 min)


Vertex connection and merging (30 min)


VM
VC
Relative technique contributions

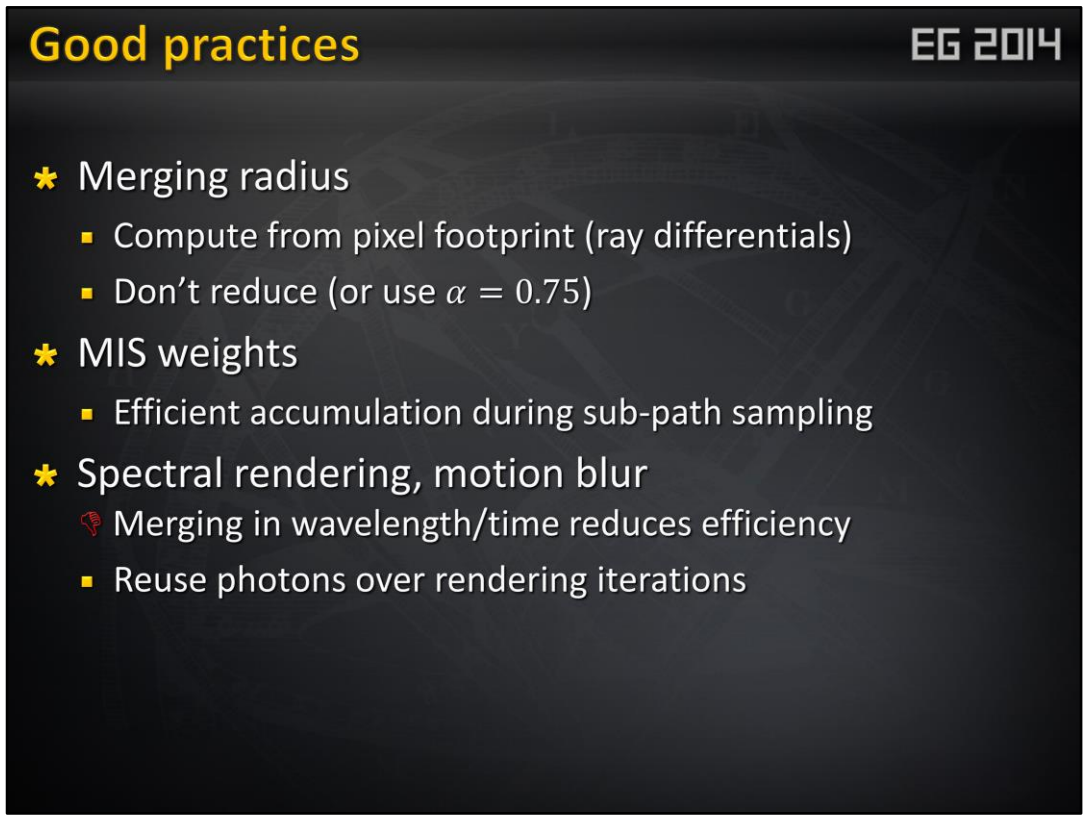


I will now discuss some good practices for the practical implementation of VCM in a production renderer.

We can almost always skip vertex merging (VM) for direct illumination and directly visible caustics, as vertex connection usually performs better. Doing this also avoids the correlated noise and bias inherent to VM.

The combined VCM algorithm I just described has the practical issue that its memory footprint can be much larger than that of (progressive) photon mapping. The problem comes from using the stored light vertices not only for merging, but for connection as well. We need to store hit point data with these vertices, which includes coordinate frame, BSDF structure, and possibly other data, making the vertex footprint as large as 1KB in some cases. This results in 1GB of storage for 1 million vertices.

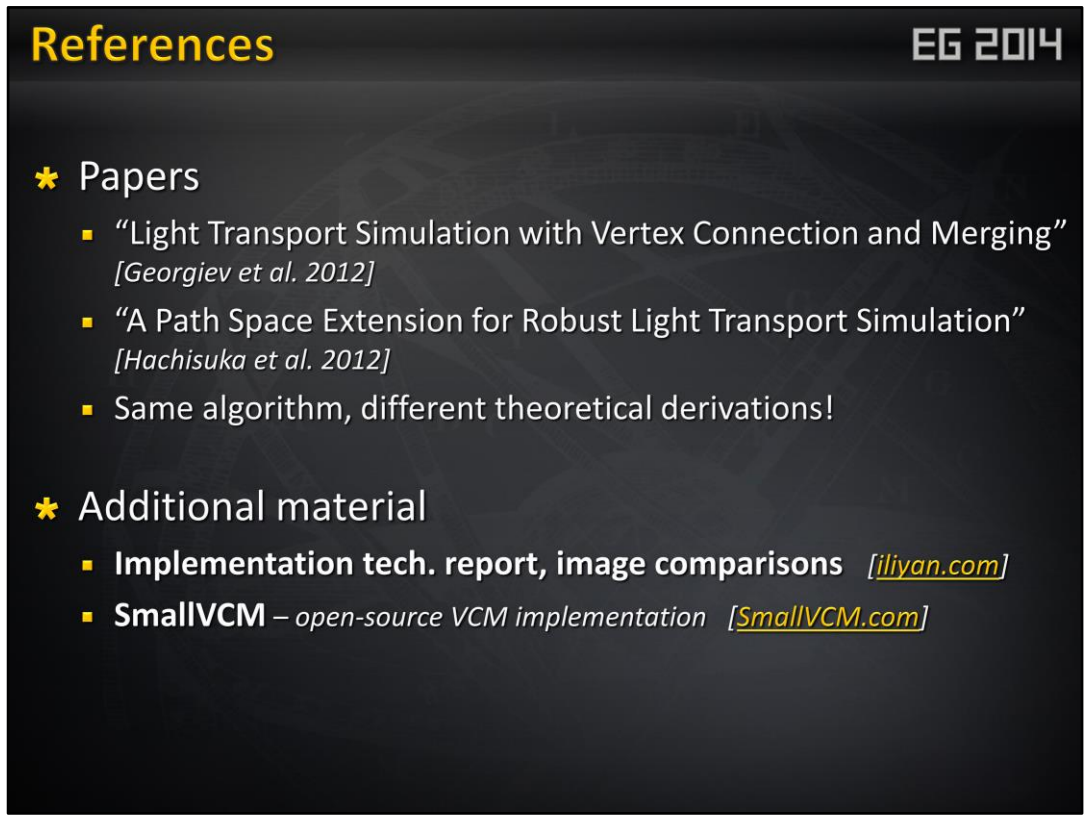
For the progressive variant of VCM, we can dramatically reduce this footprint by rearranging its computations. We follow the classical BPT implementation, where for each pixel we trace one light subpath and one camera subpath. After performing the vertex connections, we store the light subpath vertices in the acceleration structure. These vertices will then be used for merging at the *next* rendering iteration, while the camera subpath vertices for the current iteration are merged with the light vertices from the *previous* iteration. This allows us to safely strip the hit point data off the light vertices before storing them in the structure, as they will be only used for merging. And just like in photon mapping, for merging we only need to keep around a small amount of vertex data, i.e. position, direction, and throughput.



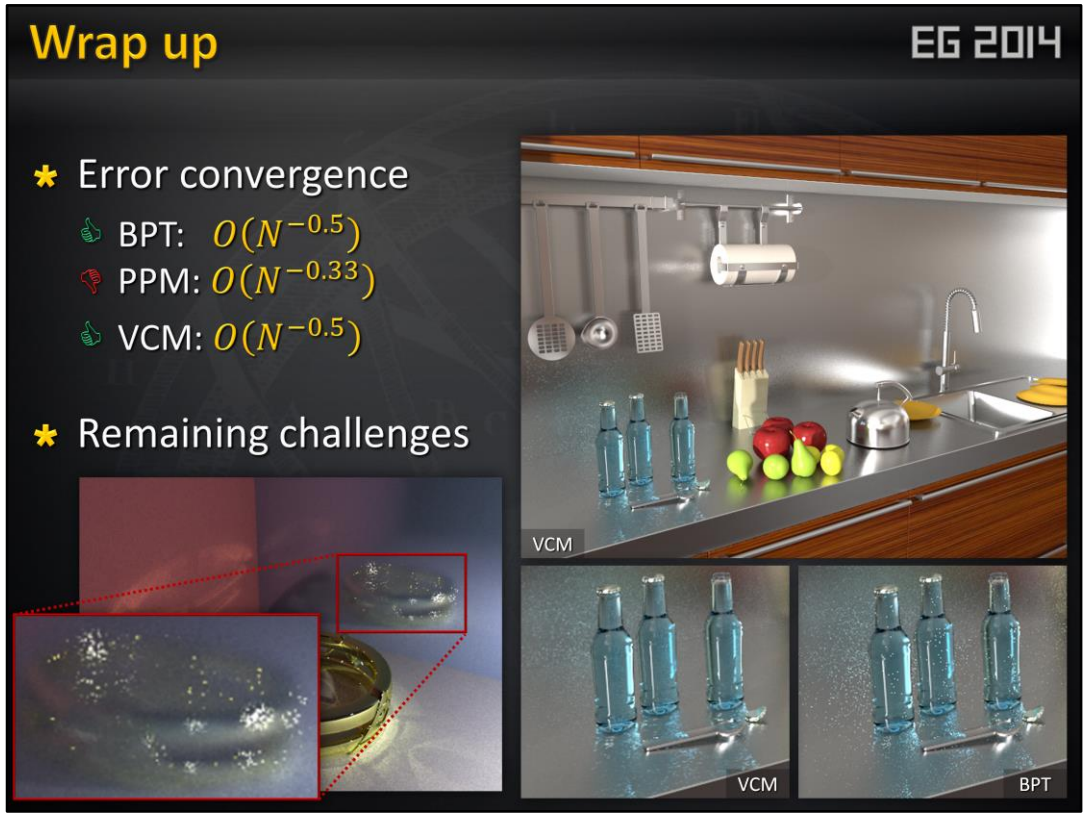
The progressive VCM algorithm has two parameters: the initial merging radius r and its reduction rate α (see "Progressive Photon Mapping" [Hachisuka et al. 2008]). We can often afford to use a much smaller radius than in PPM, as VCM mixes a large number of path sampling techniques, and VM is mostly used for caustics. An automatic and robust way to compute a good merging radius for each camera path is to derive it from the pixel footprint, which is given by the ray differentials that most renderers already implement and use for texture filtering. As for the reduction parameter α, I recommend using $\alpha = 0.75$, which is a provably good value (see VCM paper [Georgiev et al. 2012]), Alternatively, we can also opt to not reduce the radius altogether (i.e. setting $\alpha = 1$), especially when using ray differentials which give a small enough radius to mostly avoid noticeable correlated noise and bias.

Efficient MIS weight computation is another important practical aspect of VCM, and fortunately there exists a scheme for cumulative computation of weight data during the subpath random walks. This scheme is discussed in length in the technical report cited on the next slide.

And finally, a note on spectral rendering and motion blur. When a (sub)path can only carry a single randomly sampled wavelength and/or time instant, in order to merge two subpath endpoints, they not only need to be close to each other in Euclidean space, but also in wavelength and/or time. This can significantly reduce the efficiency of vertex merging, since the VM PDFs are then additionally multiplied by a corresponding wavelength/time acceptance probability, and can thus be much lower. A simple solution to ameliorate this problem is to accumulate and reuse light vertices (photons) over N iterations. This number N depends on the wavelength/time merging radius – the larger the radius, the smaller N can be. Note that efficient light vertex storage in this case becomes even more important.

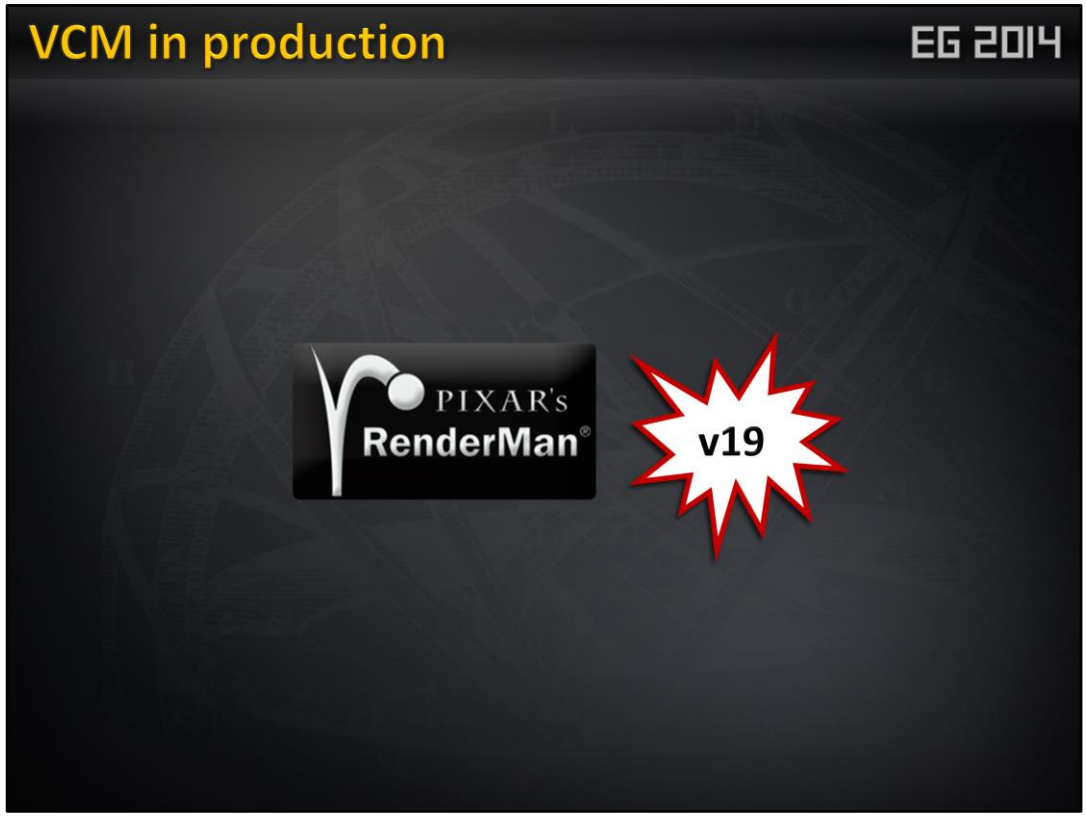


In this talk, I could only give a high-level description of VCM. For more details, please refer to these two papers, which derive the same practical algorithm using two different theoretical formulations. They were simultaneously published by two independent research groups at SIGGRAPH Asia 2012, which we believe only reaffirms the correctness of the approach. Lots of additional material can be found on my web site, and we have also released a reference open source implementation in the SmallVCM renderer.



In summary, vertex connection and merging tries to combine the best of bidirectional path tracing (BPT) and (progressive) photon mapping. An important property of the algorithm is that it retains the higher order of convergence of BPT, meaning that it approaches the correct solution faster than PPM as we spend more computational effort (i.e. sample more paths). The asymptotic analysis can be found in the VCM paper.

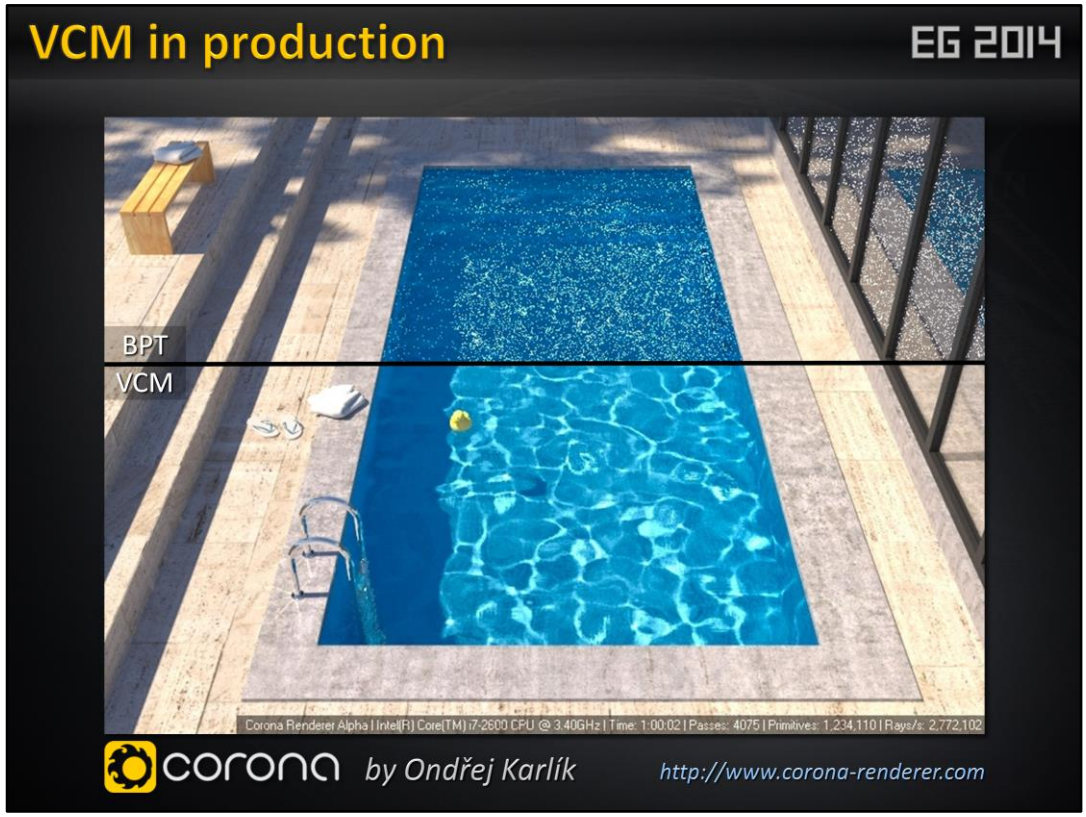
Even though VCM is a step forward in Monte Carlo rendering and has proven very useful in practice, it doesn't come without limitations. Specifically, it cannot handle more efficiently those types of light transport paths that are difficult for both BPT and PM to sample. A prominent example are caustics falling on a glossy surface. On the kitchen scene, even though VCM brings practical improvements over BPT, there is still a lot to be desired from the caustics on the glossy surface.



A number of companies have announced VCM integration in the upcoming releases of their commercial renderers.

For example, VCM is coming in Pixar's Photorealistic RenderMan v19.

Chaos Groups have also shown first images from V-Ray 3.0 with VCM support.



VCM is also implemented in the public Alpha release of Corona renderer, which I encourage you to download and play around with.