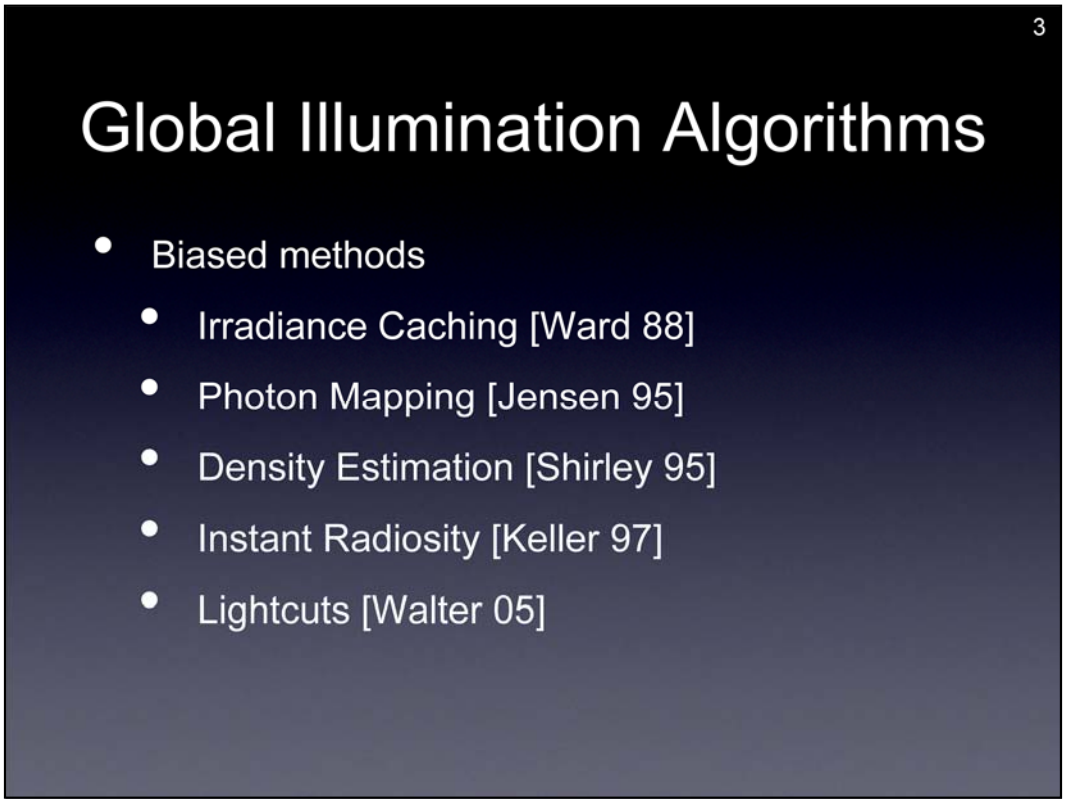
Progressive Photon Mapping

Toshiya Hachisuka[*] Shinji Ogaki[†] Henrik Wann Jensen[*]

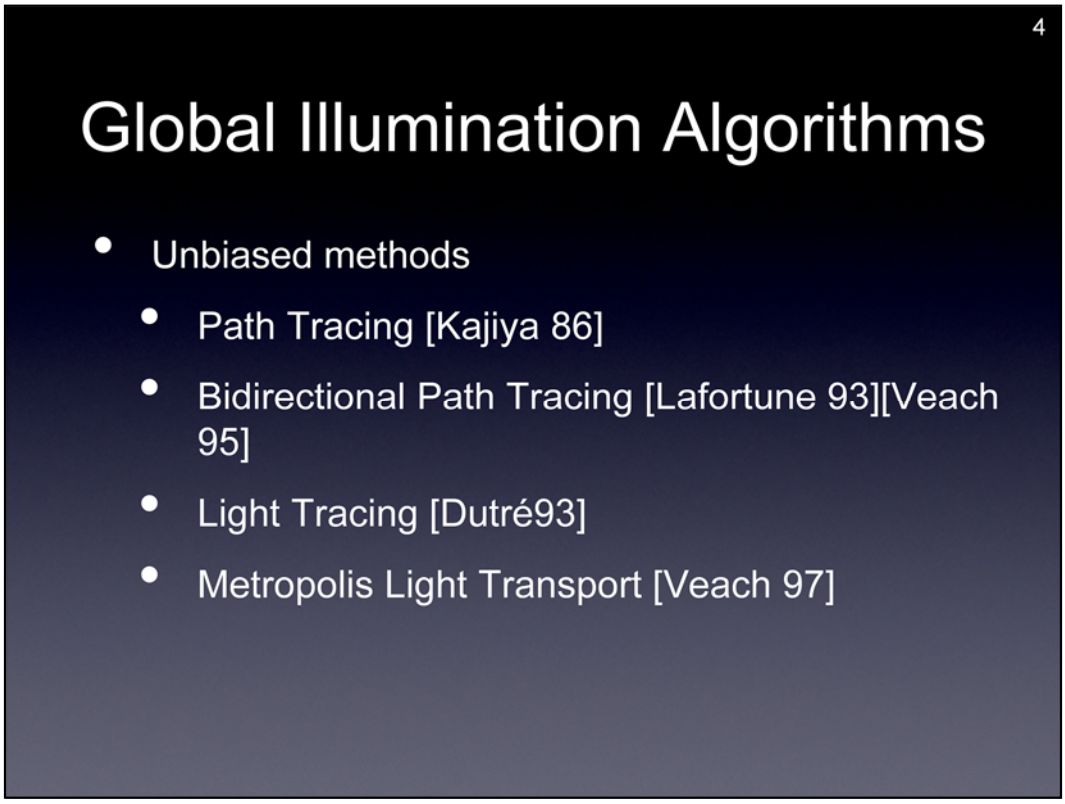
[*]University of California, San Diego
[†]University of Nottingham

Gulfstream interior

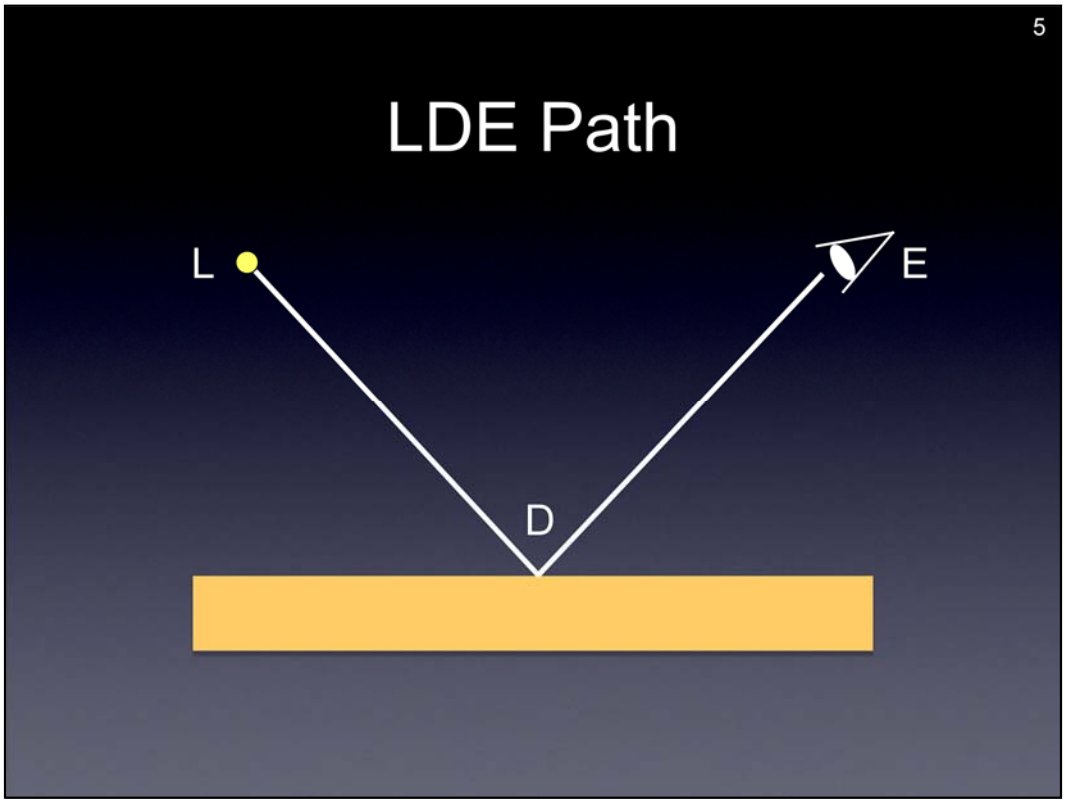
Using recent global illumination techniques, it is possible to render realistic images as shown here.



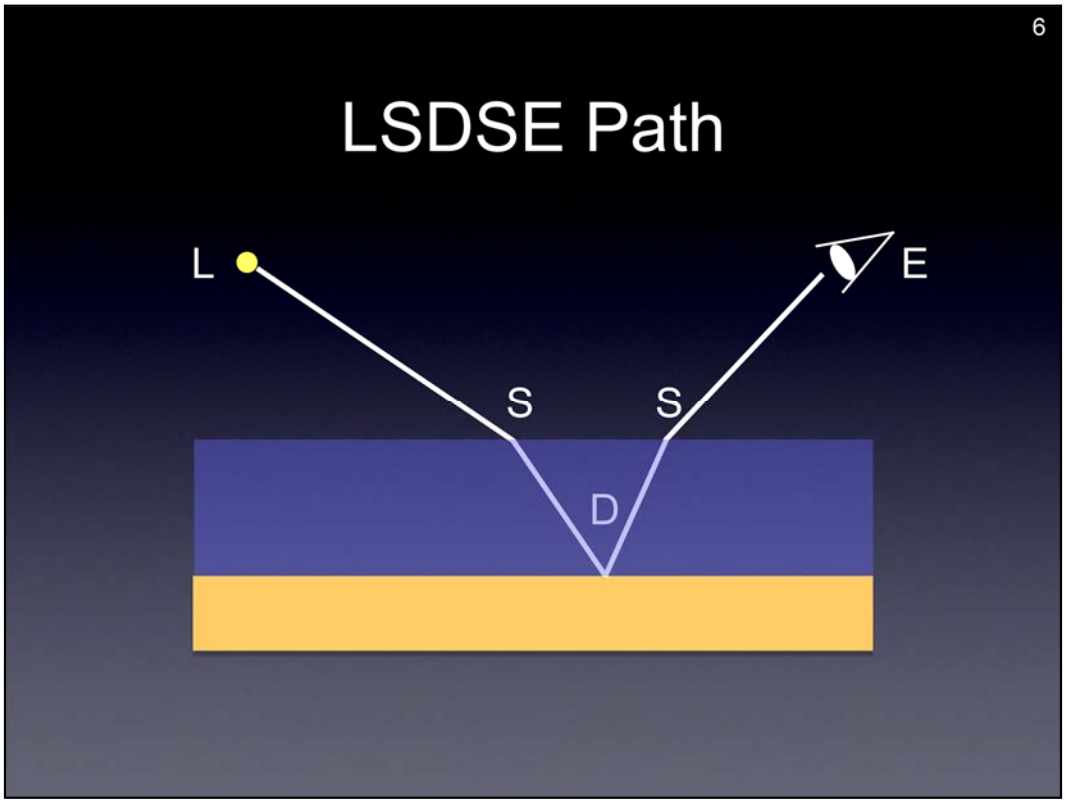
Global illumination techniques are typically based on Monte Carlo method, which are further classified into unbiased methods and biased methods. Biased methods are often faster than unbiased methods, and widely used in many rendering systems. However, images rendered by biased methods contain systematic errors associated with each algorithm.



Unbiased methods can compute correct images, in the sense that, it gives the correct solution to the rendering equation on average. If we need a very accurate image, we usually choose an unbiased method because the result can be arbitrarily accurate just by increasing the number of samples. Given all the algorithms, it is natural to think that global illumination is a solved problem. We claim that this is not true.

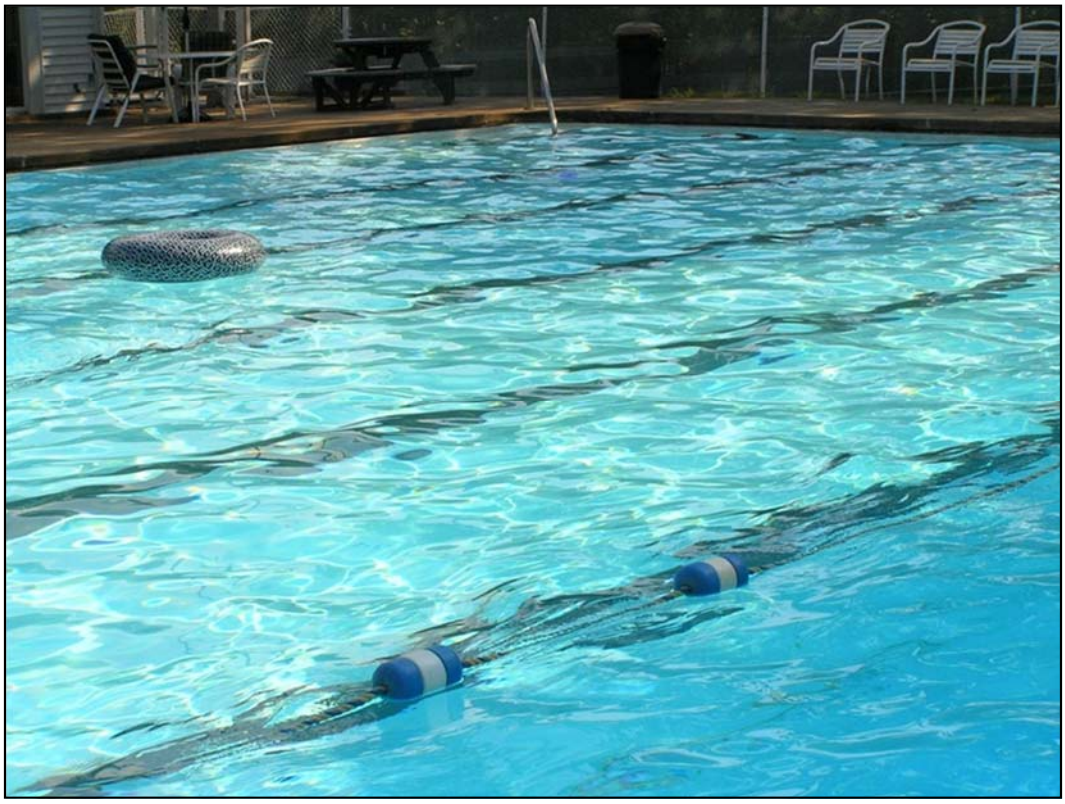


Let’s consider this scene to highlight why we claim global illumination is not solved. This path is called LDE path, where L is a light source, D is a diffuse reflection, and E is the viewer. If we see a diffuse object directly illuminated by a point light source, it is easy to construct a path of light.



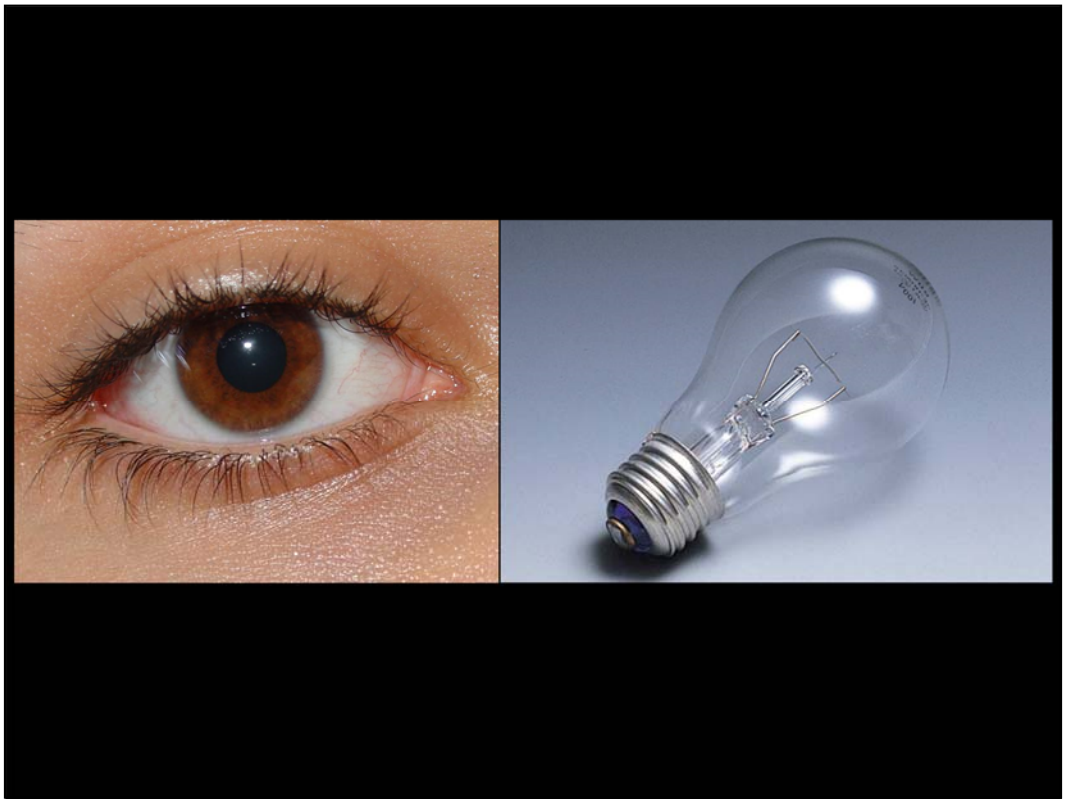
However, just by adding a refractive object on top of the diffuse object, it is no longer easy to construct a light path. The path which connects between the light source and the viewer refracts at the boundary of the refractive object, which is now called LSDSE path where S is a specular refraction. In order to construct a path, we need to find a point on the diffuse object that connects the viewer and the light source after refractions. If the light is a point light source, then it is in fact impossible to compute this path with any unbiased method.

You might think it is a rare case, but we see this type of illumination very often in our daily life. For example, let’s take a look at this simple photograph of a store window. Since the window causes specular refractions and the sunlight is coming through the window, everything you see through the window contains an SDS path.

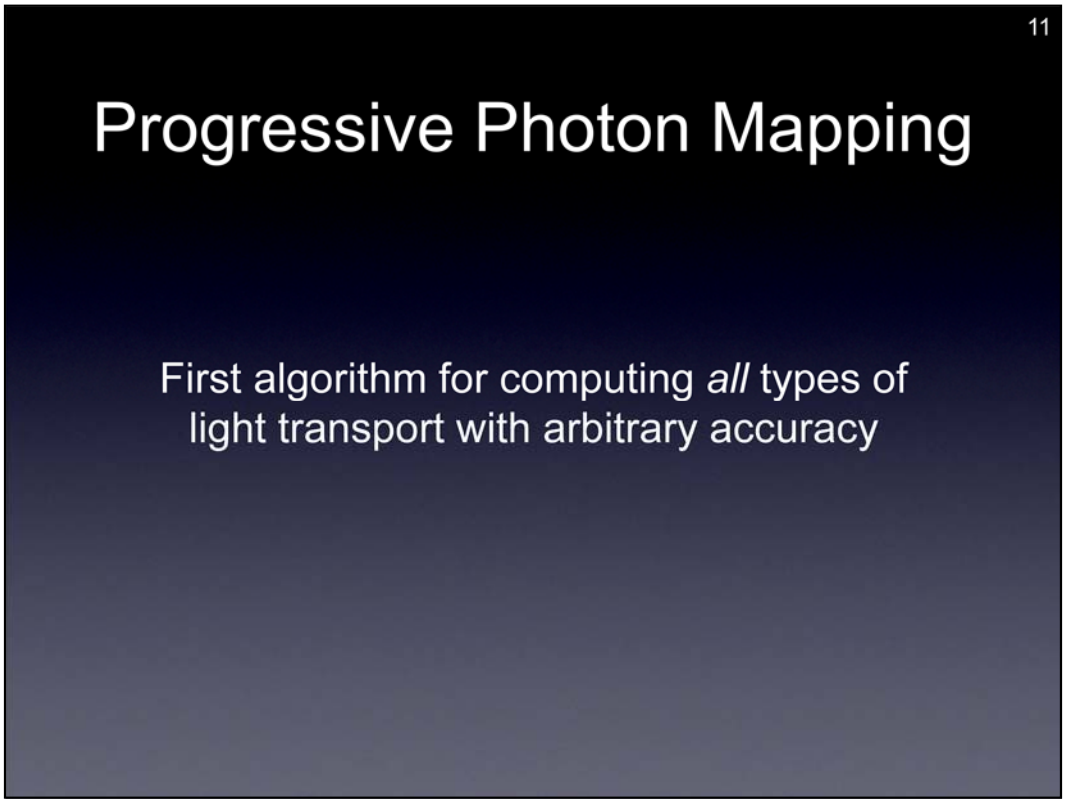


The bottom of a swimming pool is another example of SDS paths, where it is illuminated by light coming through the water surface, and we see that above the water surface.

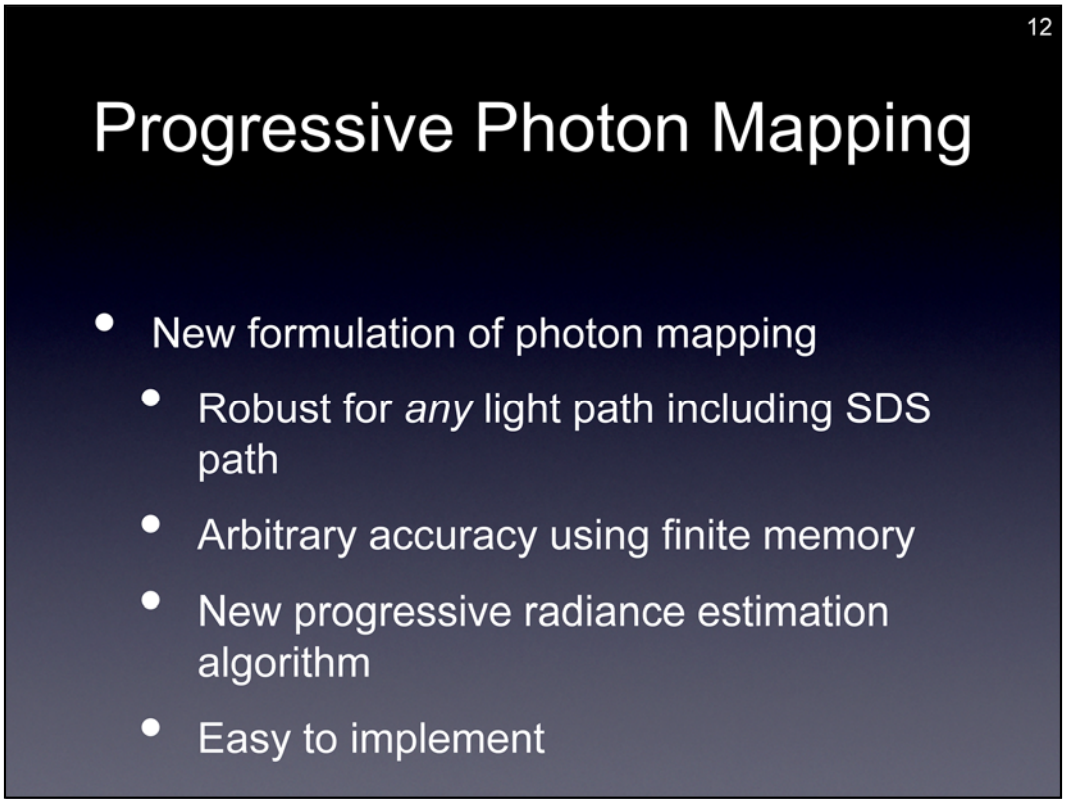
This is a picture of an ordinary bathroom. Almost everything you see in the mirror is a SDS path. The reason is that a glass casing of light bulb causes specular refraction before illuminating any diffuse surface. Therefore, what we see in the mirror is dominated by SDS paths.



If you want to be extremely precise, you would need to consider that the lens of our eyes or camera and the glass casing around a light bulb, which ultimately create SDS paths everywhere. I hope you are convinced that it is important to handle SDS paths in global illumination.

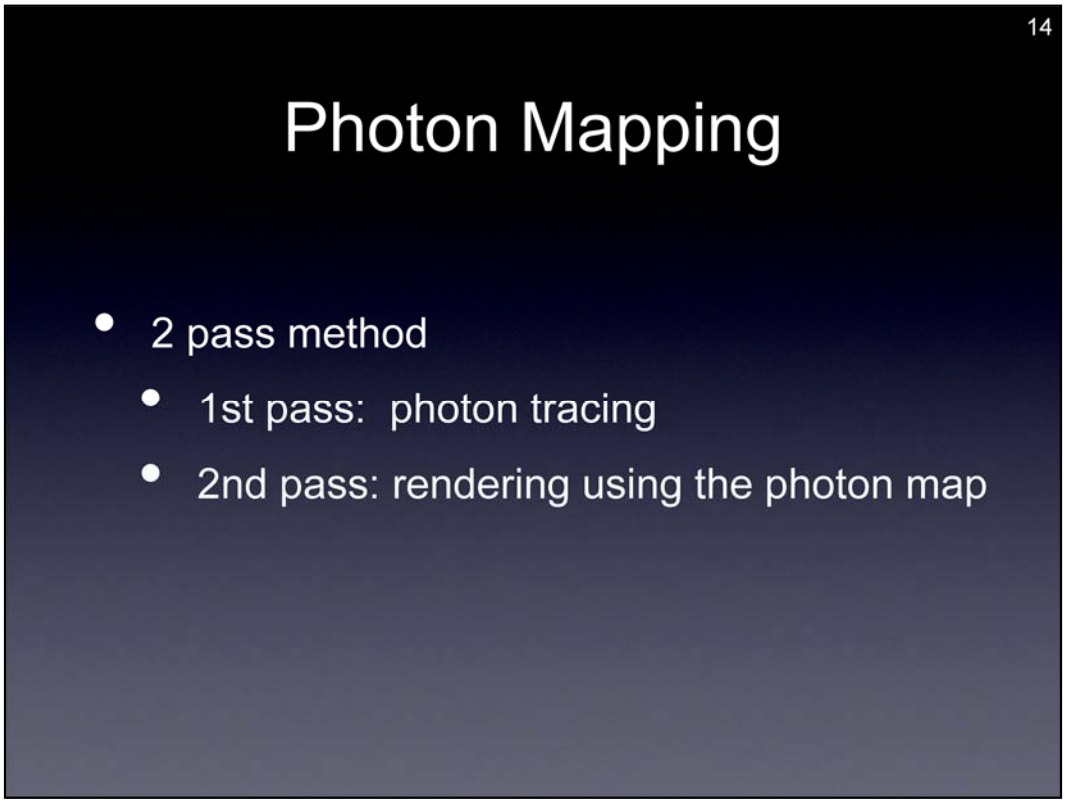


Our progressive photon mapping is the first method for computing all types of light transport, including SDS paths, with arbitrary accuracy.

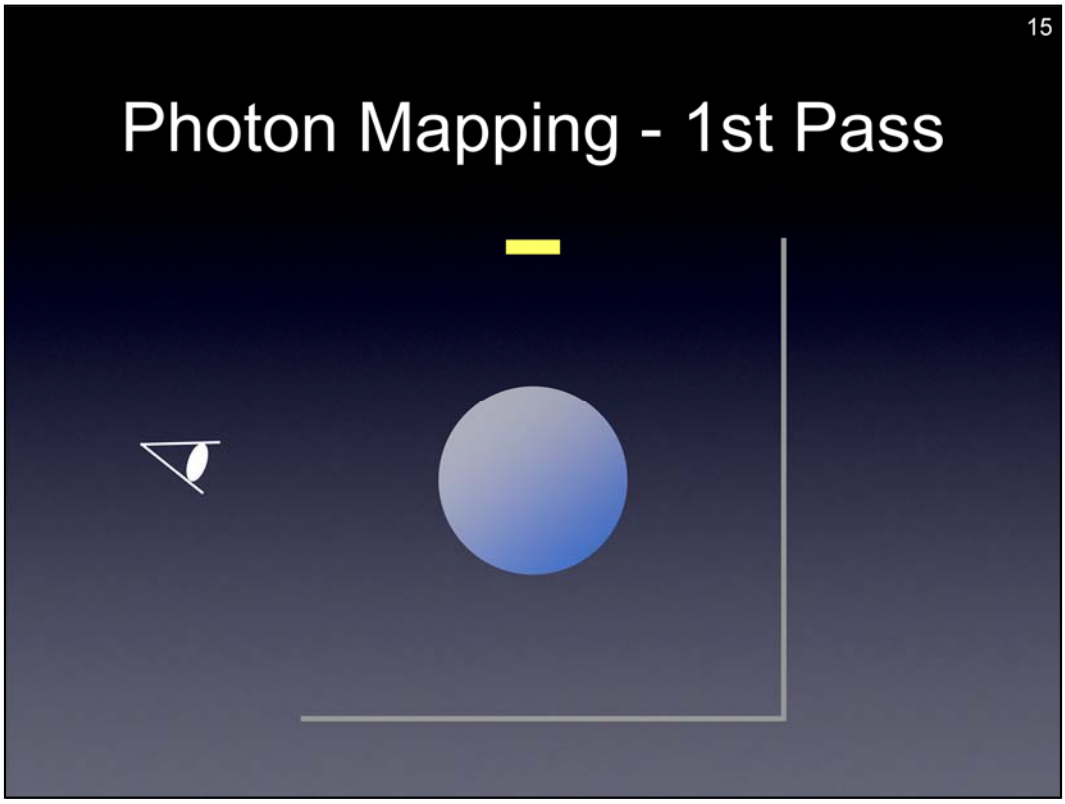


To be more precise, our progressive photon mapping is a new formulation of photon mapping. Our method is robust for any light path including SDS path. We can compute images with arbitrary accuracy just by increasing the number of photons without storing all the photons. To do this, we introduce a new progressive radiance estimation algorithm, which is easy to implement.

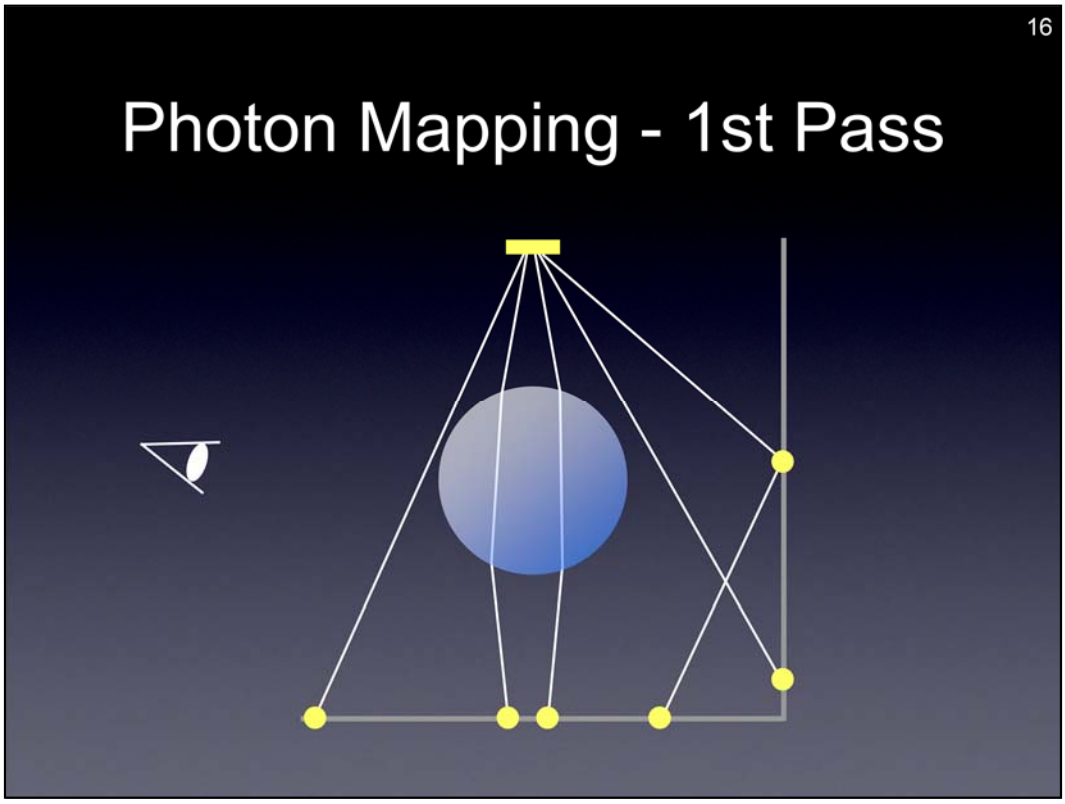
Since our method is based on photon mapping, let’s for a moment look at the standard photon mapping.



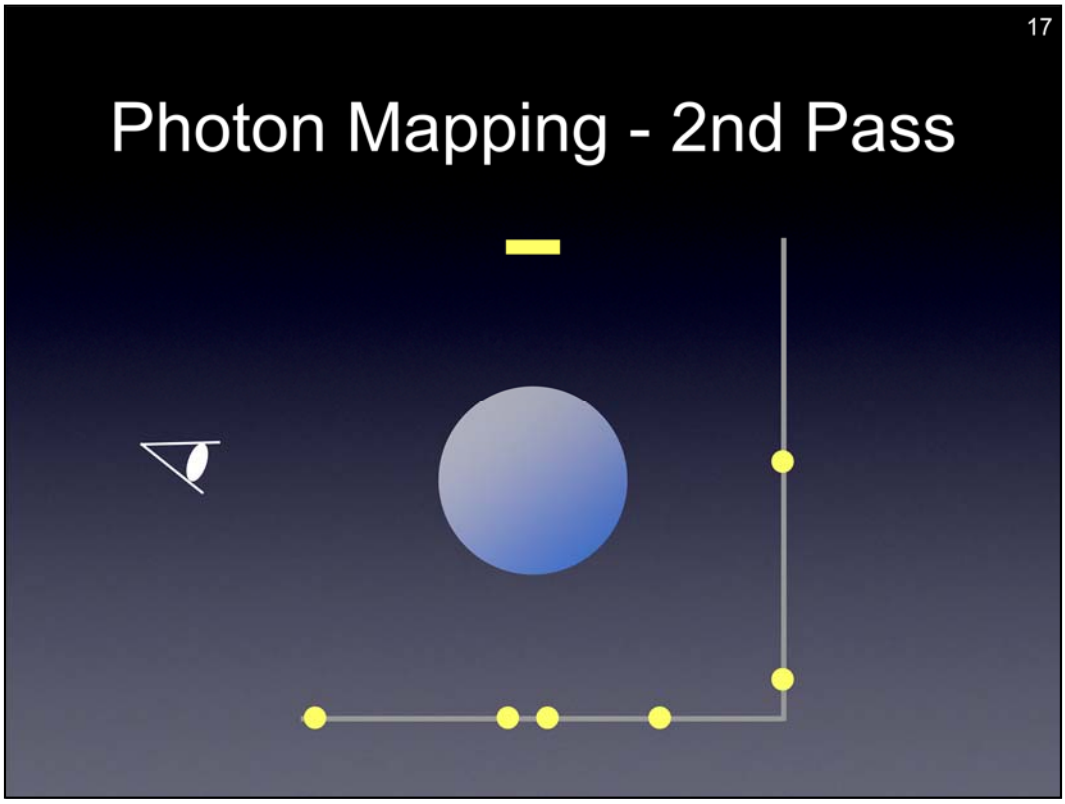
Photon mapping is a two pass method. In the first pass, photons are emitted from light sources and interactions of photons with surfaces are stored as a photon map. In the second pass, an image is rendered using the photon map from the first pass.



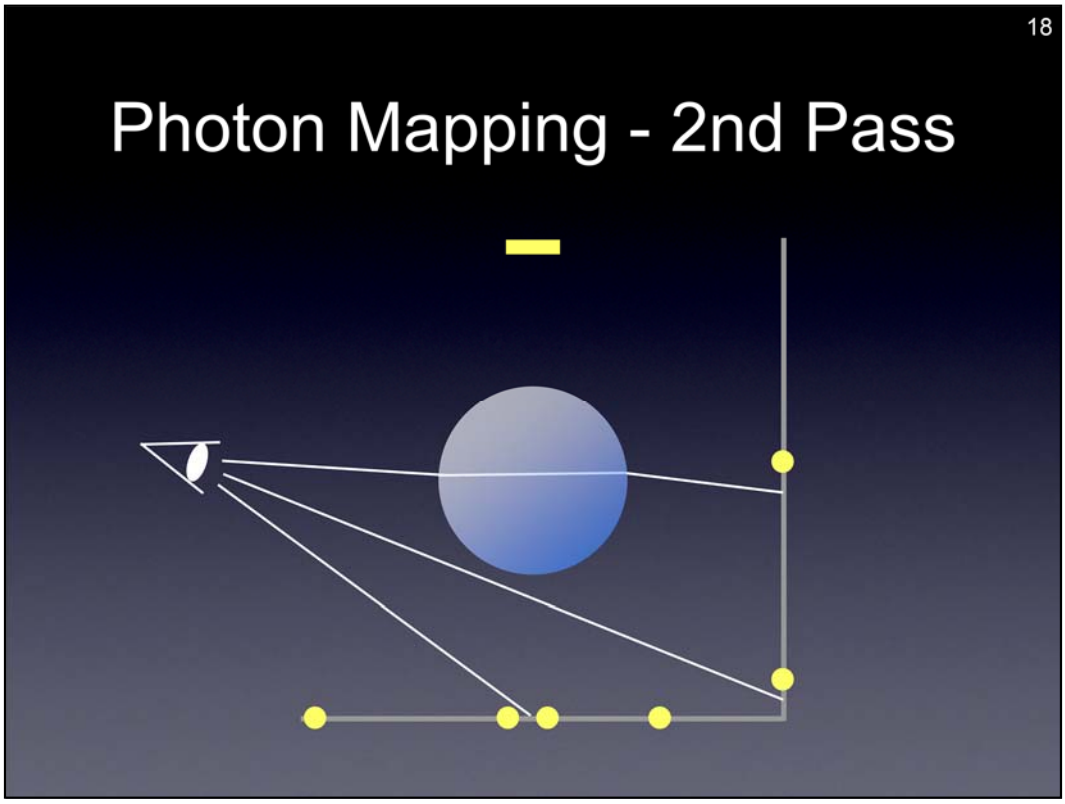
Let’s look at this example scene, where there is a glass ball, diffuse walls, and the light source at the top.



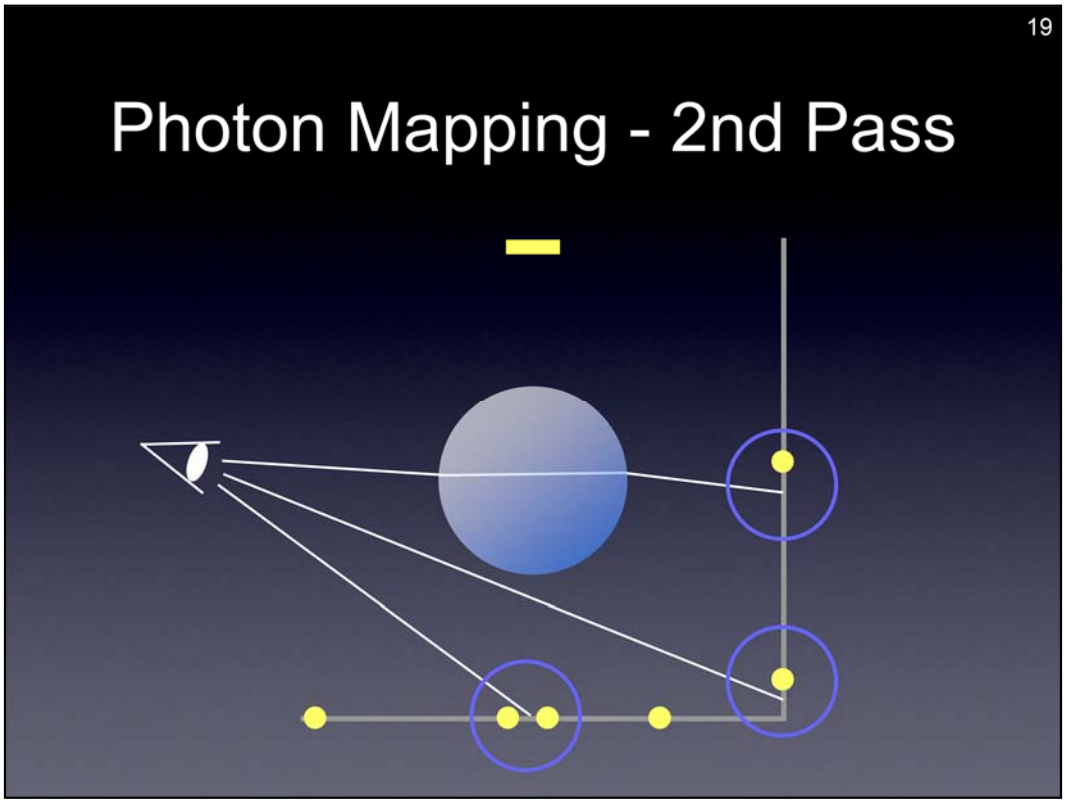
In the first pass of photon mapping, we trace photons from the light source, and store the intersections with diffuse surfaces as a photon map.



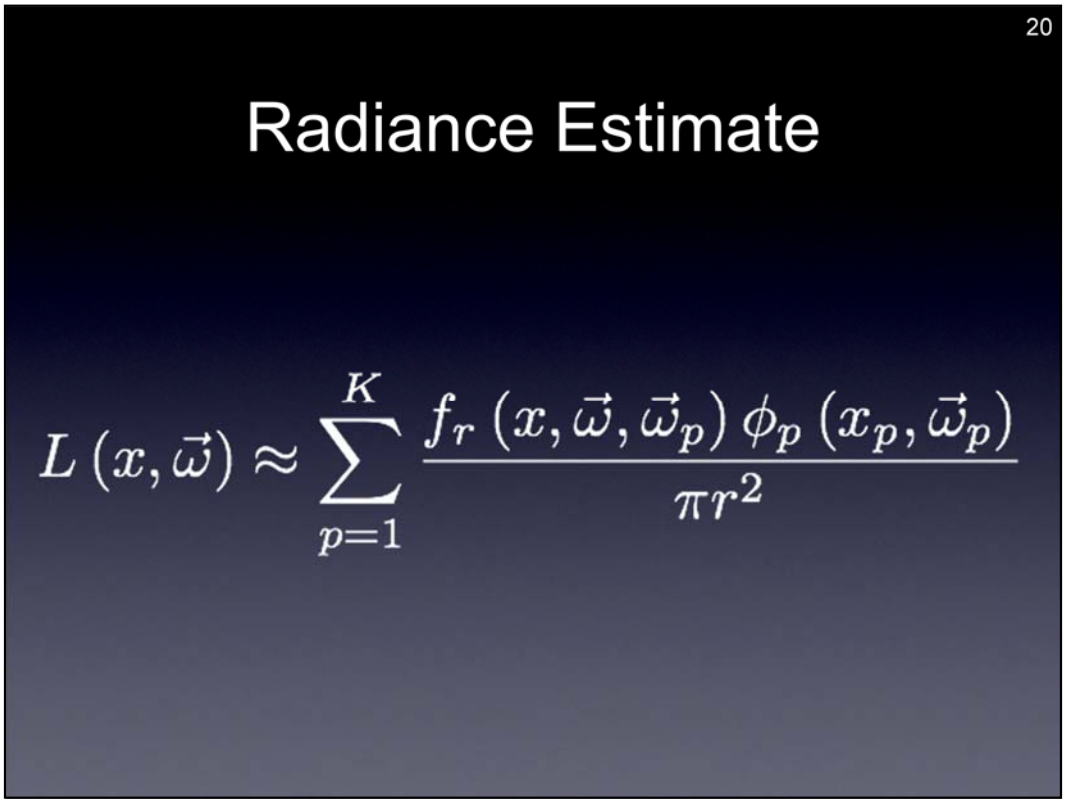
In the second pass, we trace rays from the eyes,



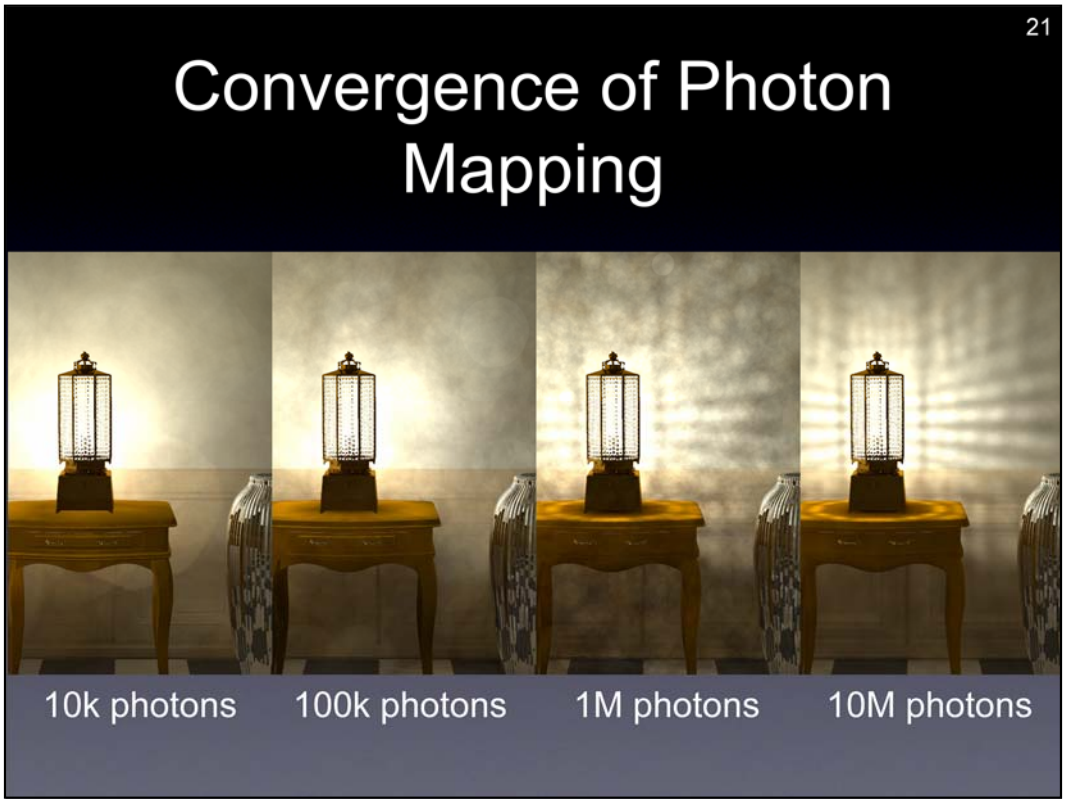
In the second pass, we trace rays from the eyes,



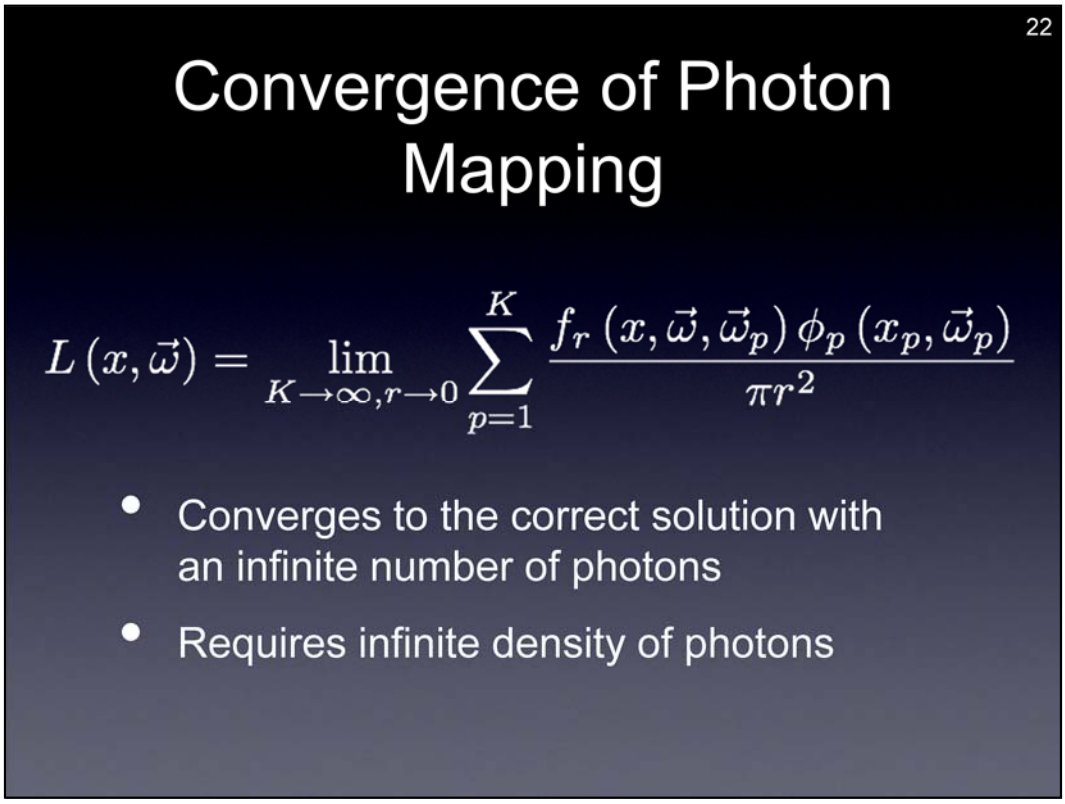
and estimate the resulting radiance by finding nearby photons around each intersection point of eye ray.



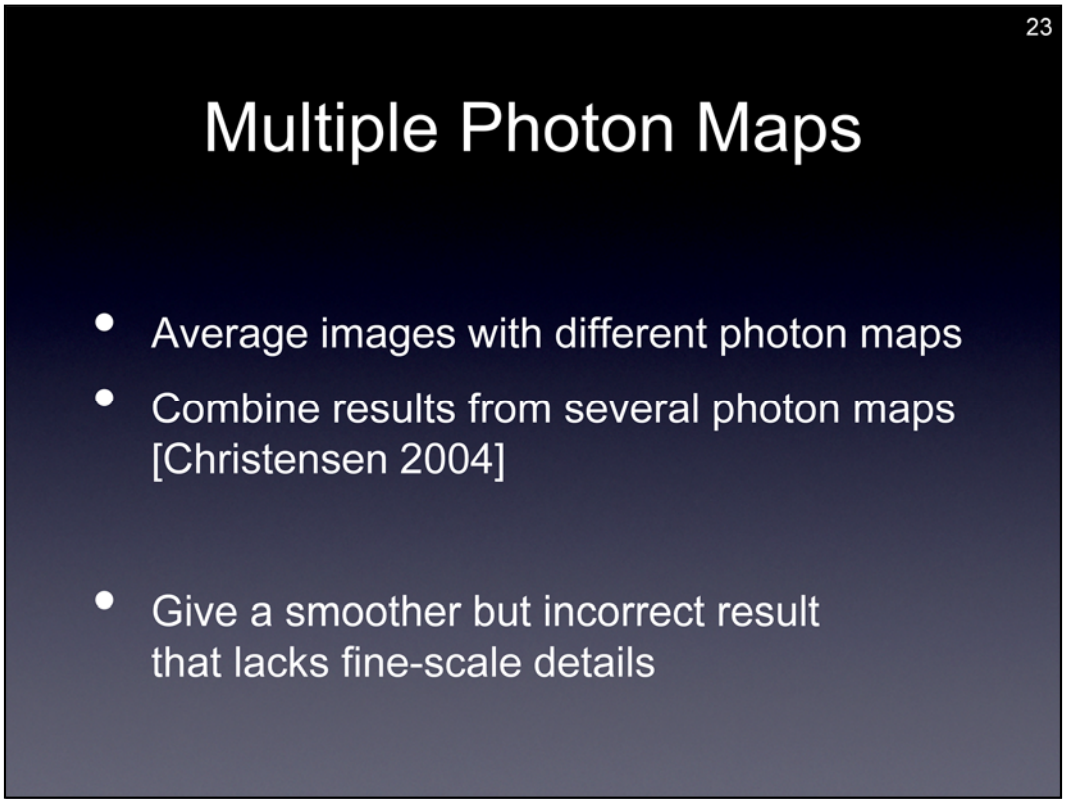
We use this equation to estimate radiance, where K is the number of nearby photons around x, f_r is a BRDF, phi_p is flux (or power) of each photon, and r is the search radius of all the nearby photons. Note that this equation is an approximation of the correct radiance using K photons.



Although photon mapping is a biased method, it is a consistent method. It means is that the image rendered by photon mapping converges to the correct solution by increasing the number of photons.

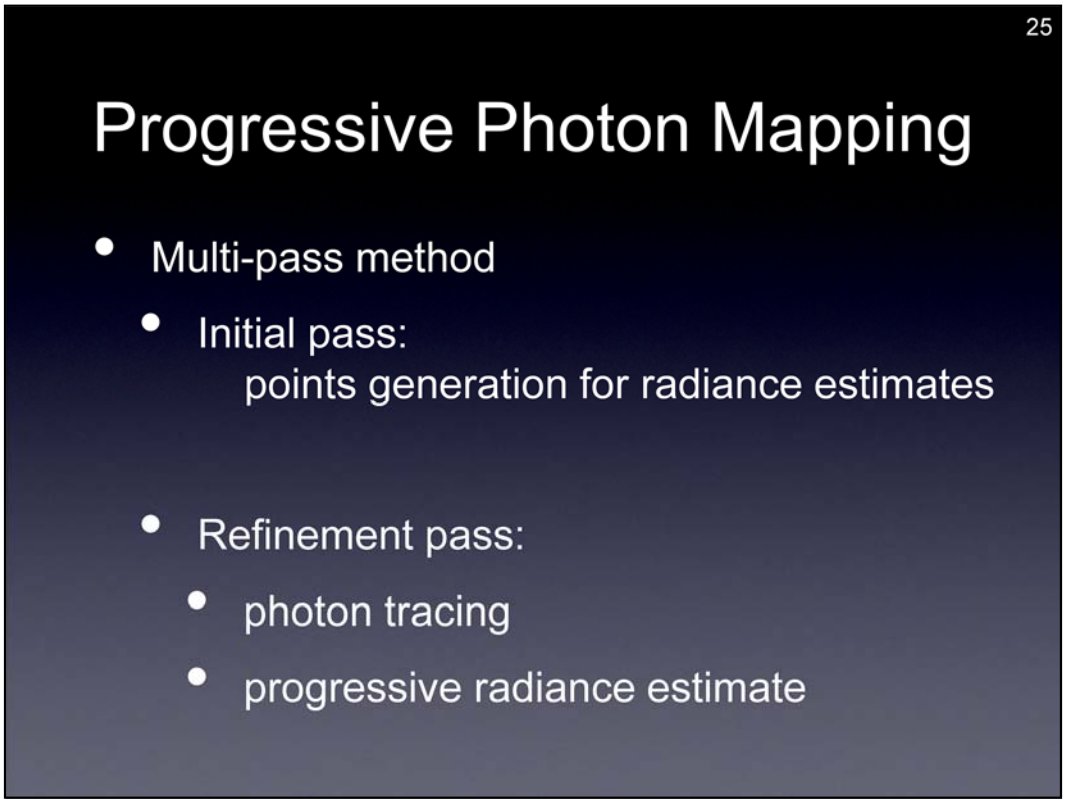


To be more precise, radiance computed from nearby photons converges to the correct solution of the rendering equation, if we use an infinite number of photons within an infinitely small search radius. Unfortunately this is not practical since it would require an infinite amount of memory.

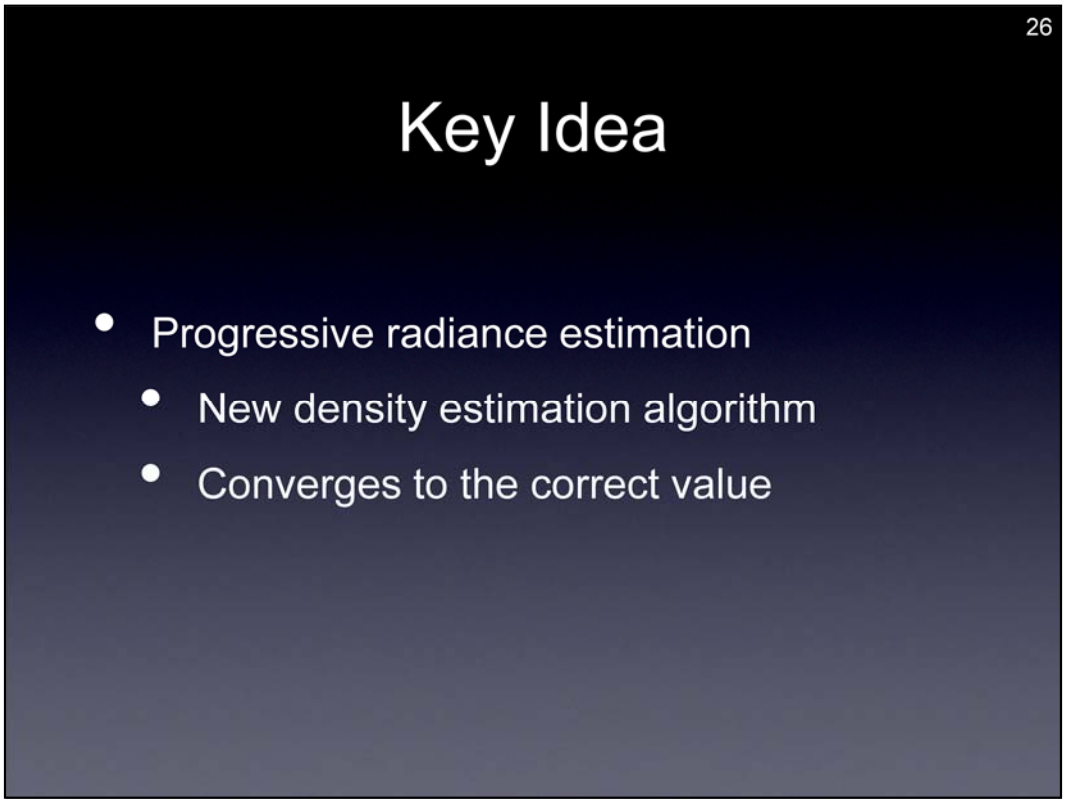


Instead of using single photon map with a large number of photons, one can think of combining results from several photon maps with a small number of photons, to increase the total number of photons. The simplest method would be to take the average of images rendered by different photon maps. Christensen presented a more sophisticated method to combine several photon maps. These methods give a smoother result, but details of lighting would be missing if they are not captured by individual photon map. In other words, the result does not converge to the correct solution even if an infinite number of photons is used.

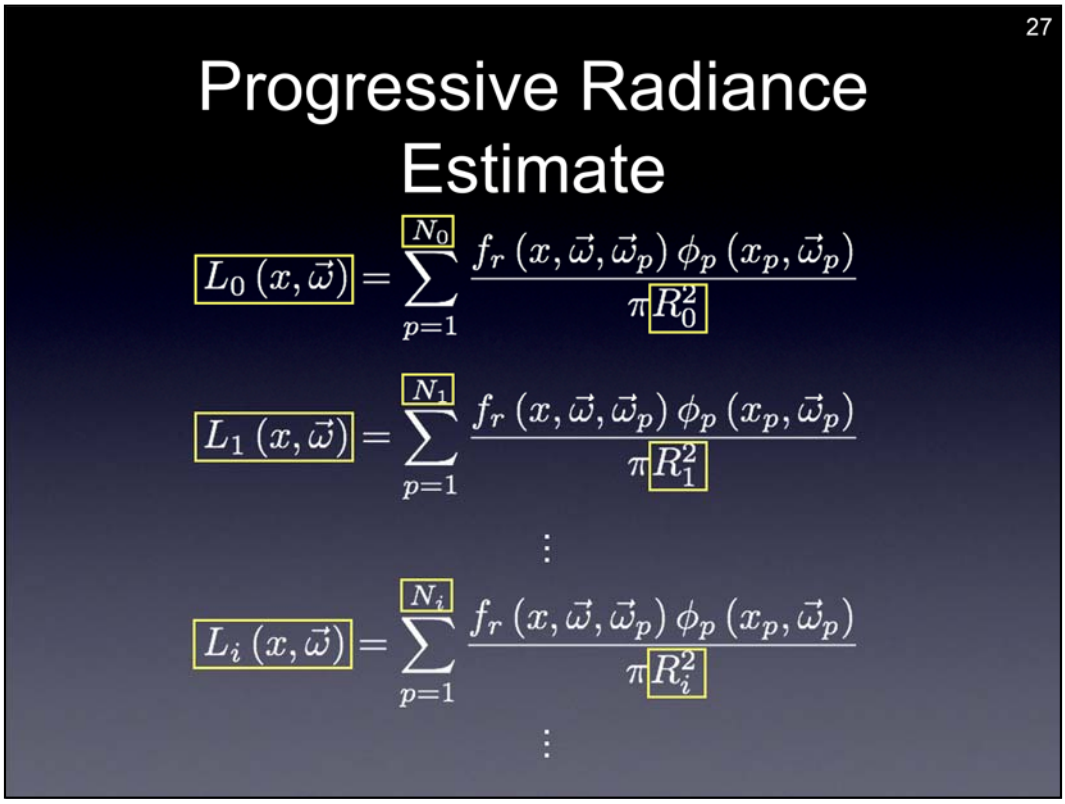
In contrast, progressive photon mapping converges to the correct solution, and I will now describe how this is achieved.



Progressive photon mapping is a multi-pass method. In the initial pass, we generate points where we want to estimate radiance, which is usually done by ray tracing. In the succeeding refinement passes, we trace photons exactly in the same way as the standard photon mapping. We then apply a new progressive radiance estimate to compute radiance at each point.



The key idea of progressive photon mapping is in progressive radiance estimation. It is based on a new density estimation algorithm where the result converges to the correct value after an infinite number of refinement passes.

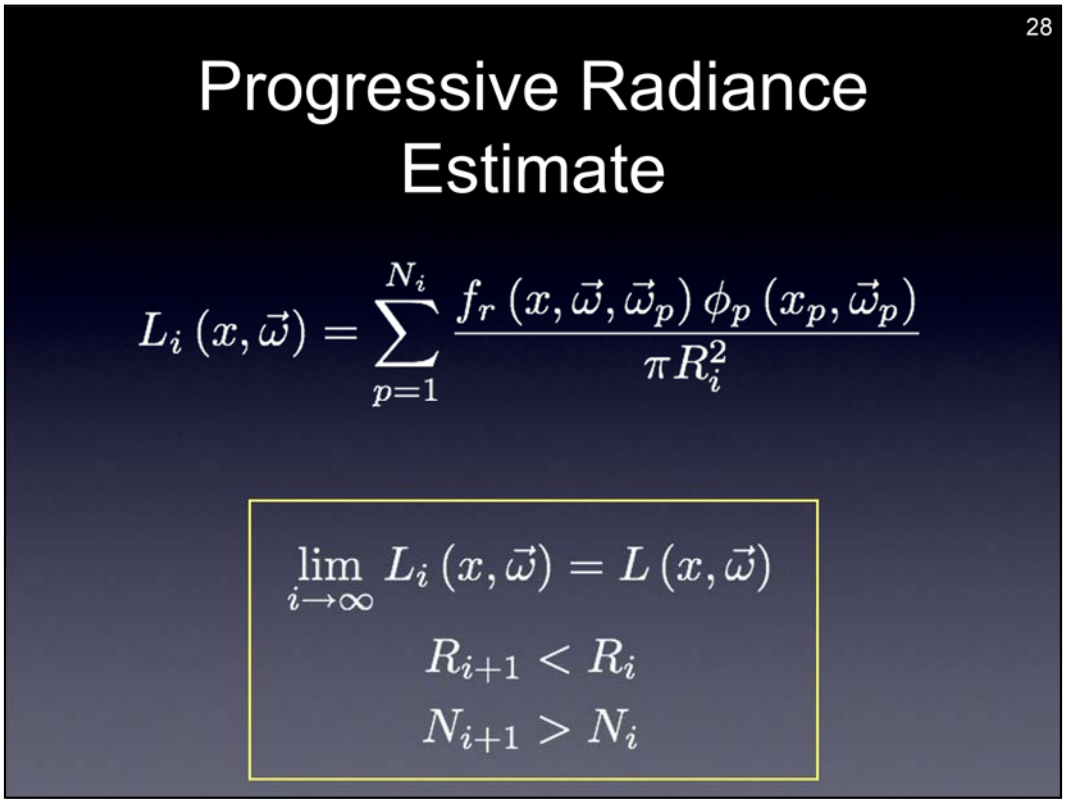


In progressive radiance estimate, we estimate radiance at a specific point using an iterative approach.

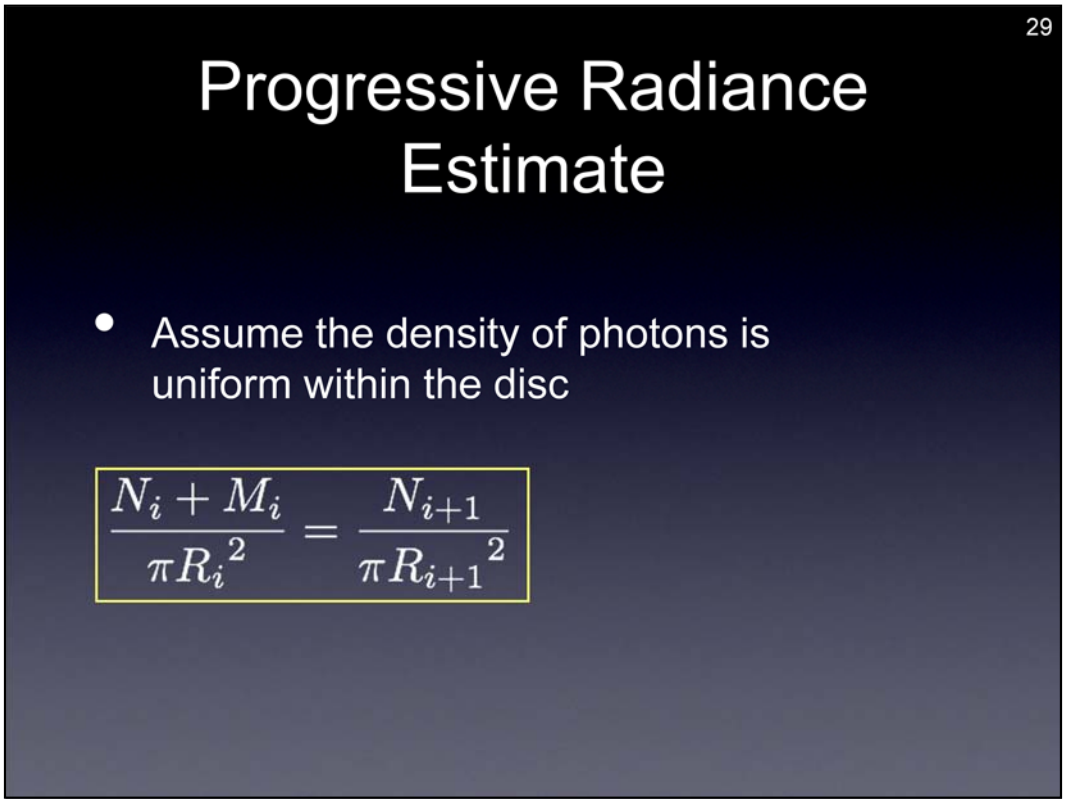
In the first pass, we have N_0 photons within a disc of radius R_0, and we compute radiance using this equation.

In the second pass, we accumulate more photons and refine the estimated radiance.

We keep repeating this process to obtain more accurate radiance estimates.



In order to achieve convergence to the correct value after an infinite number of refinement passes, we refine the estimate of radiance iteratively. After each iteration, the search radius should decrease and the number of nearby photons should increase to ensure convergence. I will now describe how this can be done.



Assume we have a point with N_i photons and we would like to add the contribution from M_i new nearby photons. Our goal is to obtain N_i+1 photons within a disc of radius R_i+1 under the conditions I showed before. If we assume that the density of photons within the disc is uniform, we can express the density before and after the iteration as shown in this slide.

Progressive Radiance Estimate

- Keep a fraction α of newly added photons M_i

$$\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_{i+1}}{\pi {R_{i+1}}^2}$$

$$\boxed{N_{i+1} = N_i + \alpha M_i}$$

To ensure that the number of photons is increasing, we accumulate a fraction alpha of the new nearby photons.

Progressive Radiance Estimate

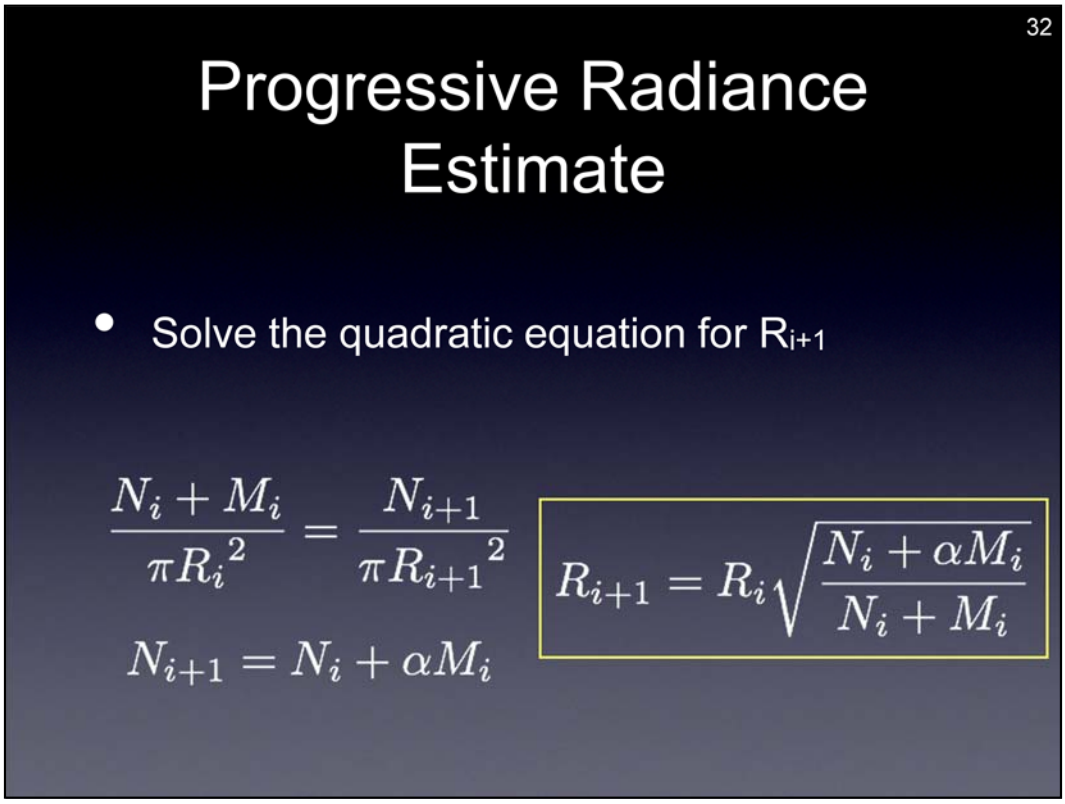
- Solve the quadratic equation for R_{i+1}

$$\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_{i+1}}{\pi {R_{i+1}}^2}$$

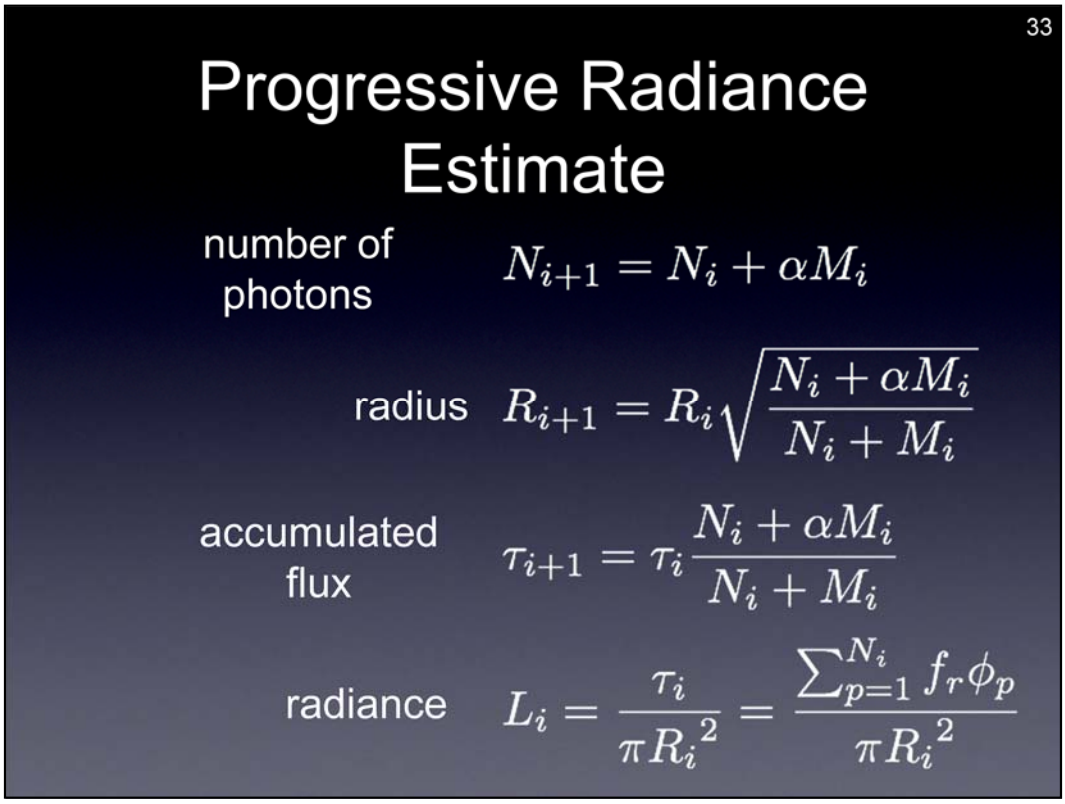
$$N_{i+1} = N_i + \alpha M_i$$

$$\boxed{\frac{N_i + M_i}{\pi {R_i}^2} = \frac{N_i + \alpha M_i}{\pi {R_{i+1}}^2}}$$

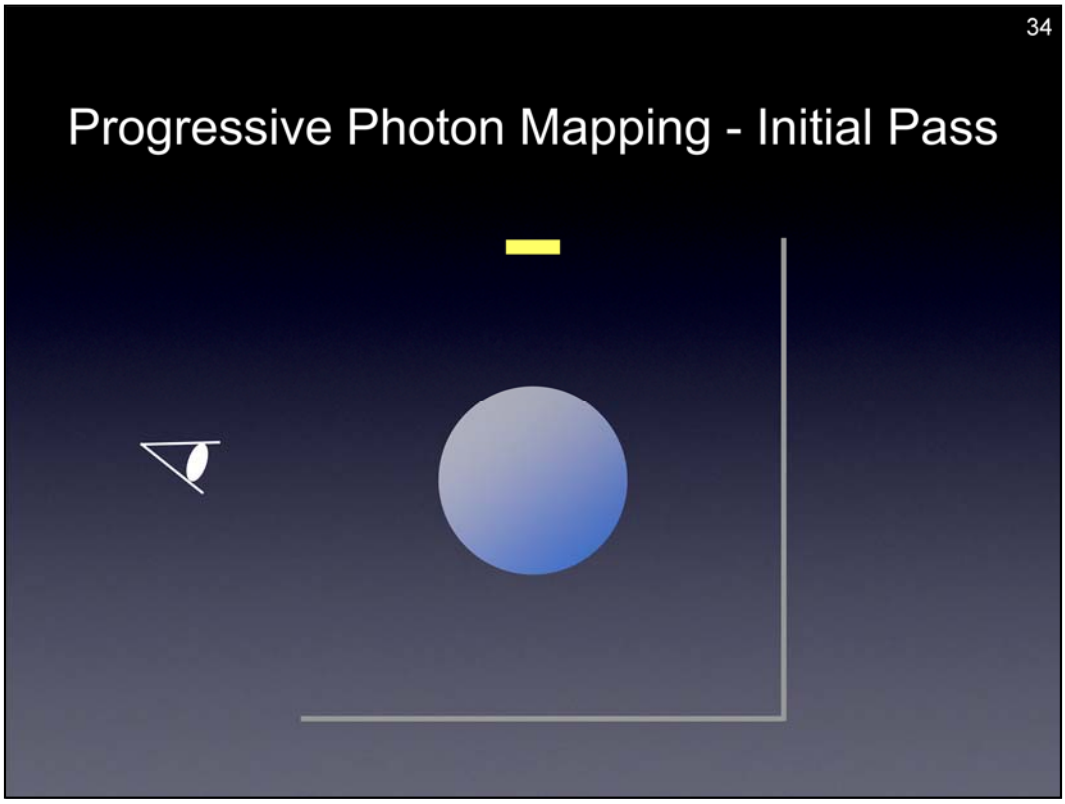
We then combine these two equations to obtain a quadratic equation of the new radius,



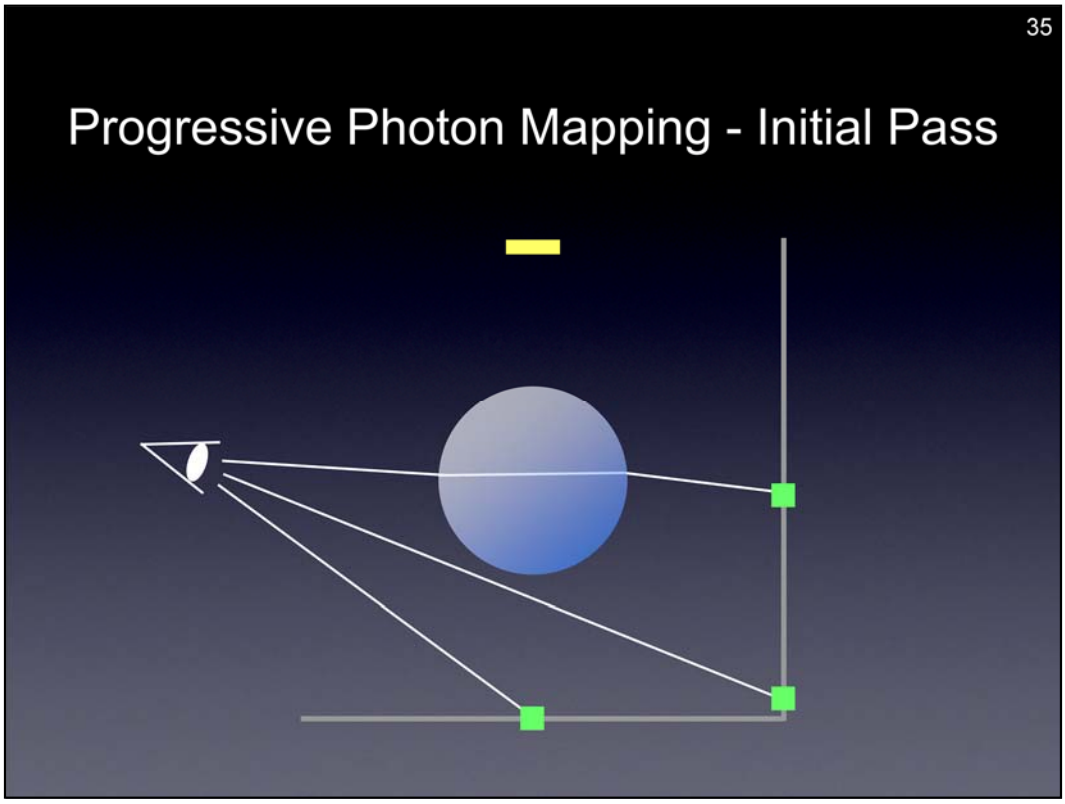
and solve this equation to get the new radius R_i+1. We use a similar approach to accumulate the flux associated with a point.



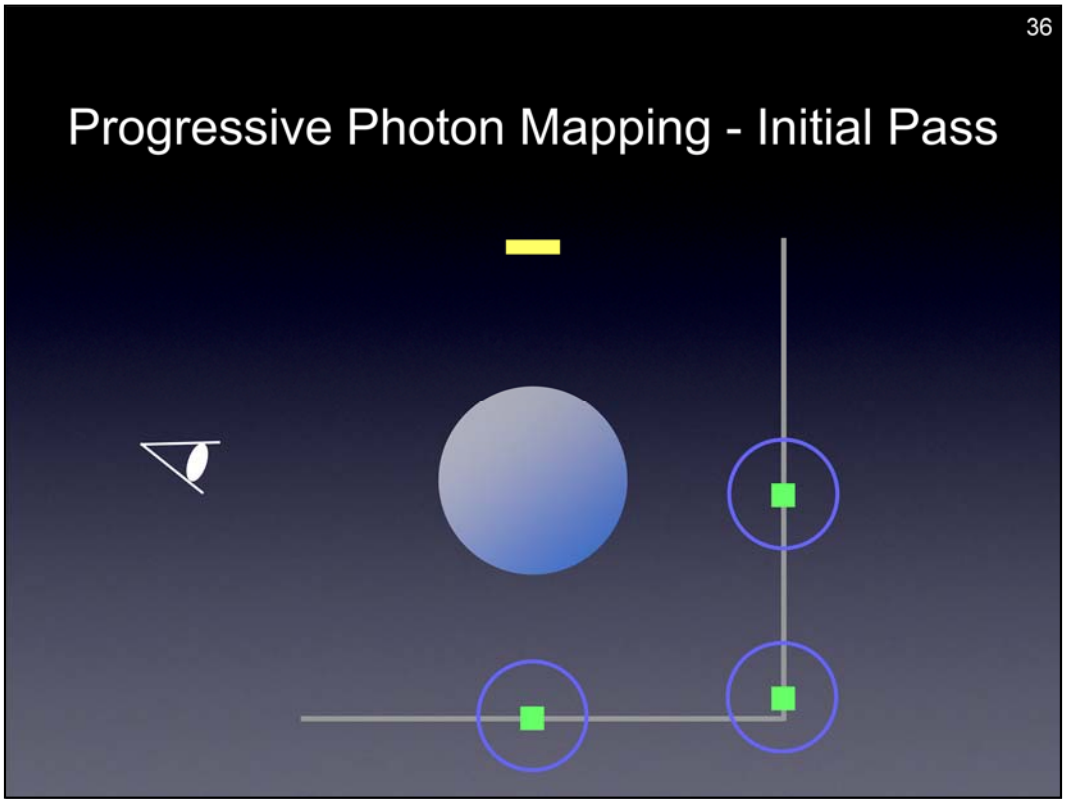
In summary, we store the number of nearby photons, the search radius, the accumulated flux at each point, and simply update the values using these equations after each iteration.



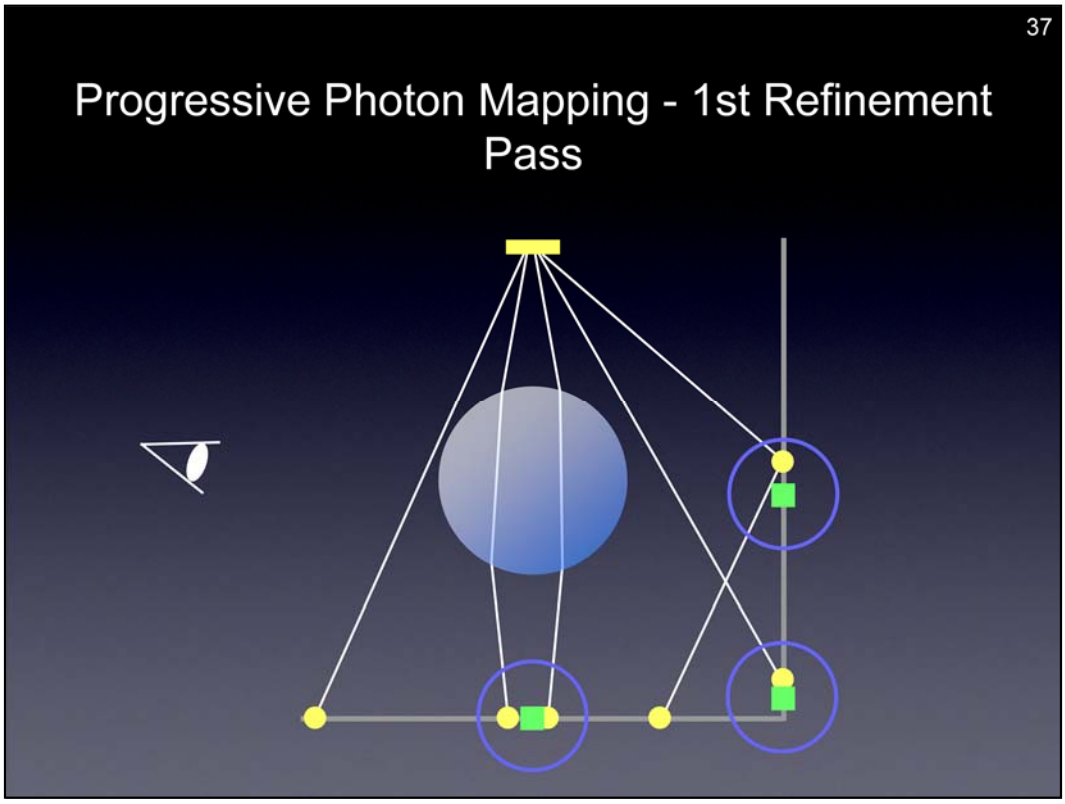
To summarize the overall algorithm, let’s go back to the same example as we have seen in the standard photon mapping.



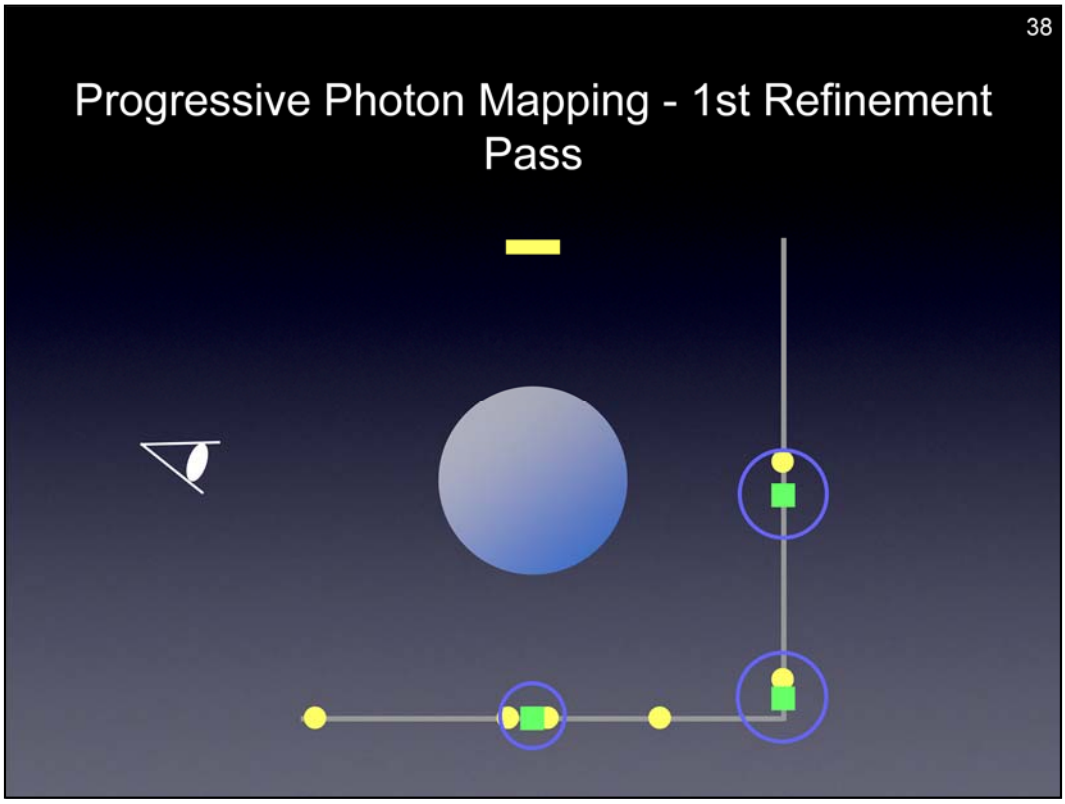
In the initial pass of progressive photon mapping, we generate points where we want to estimate radiance by tracing rays from the viewer. This process is very similar to the second pass of the standard photon mapping except we now store information of each point.



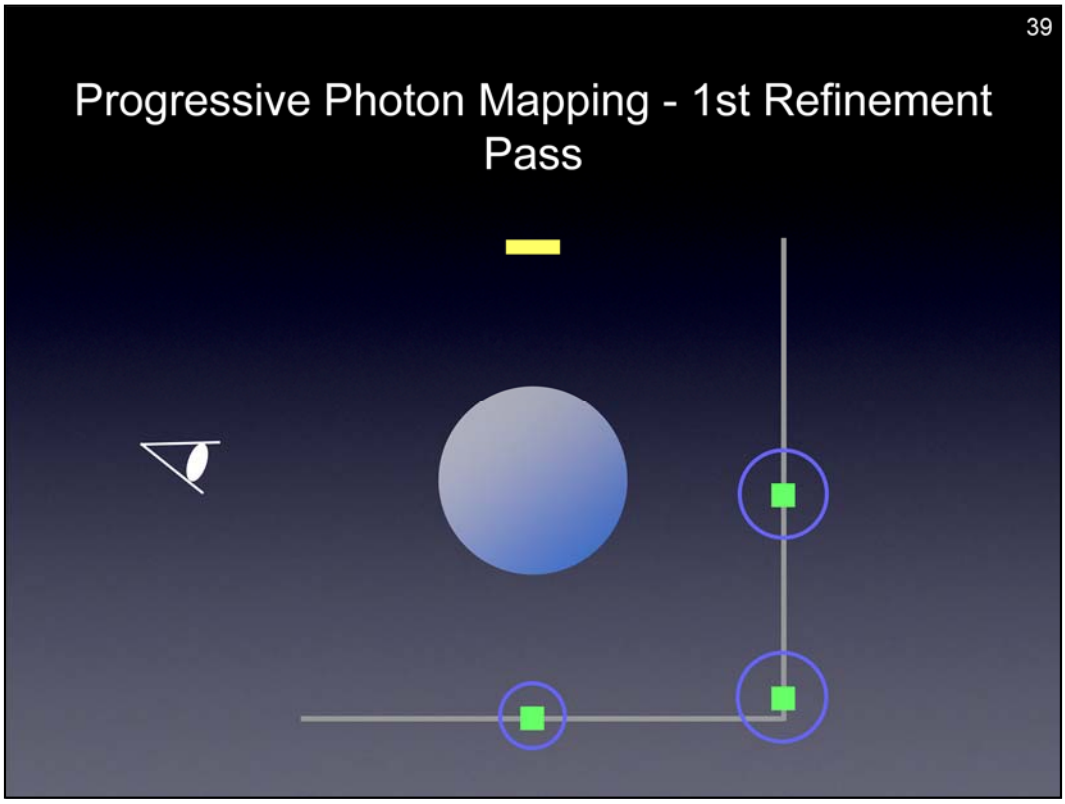
After the first iteration, each point is assigned an initial search radius.



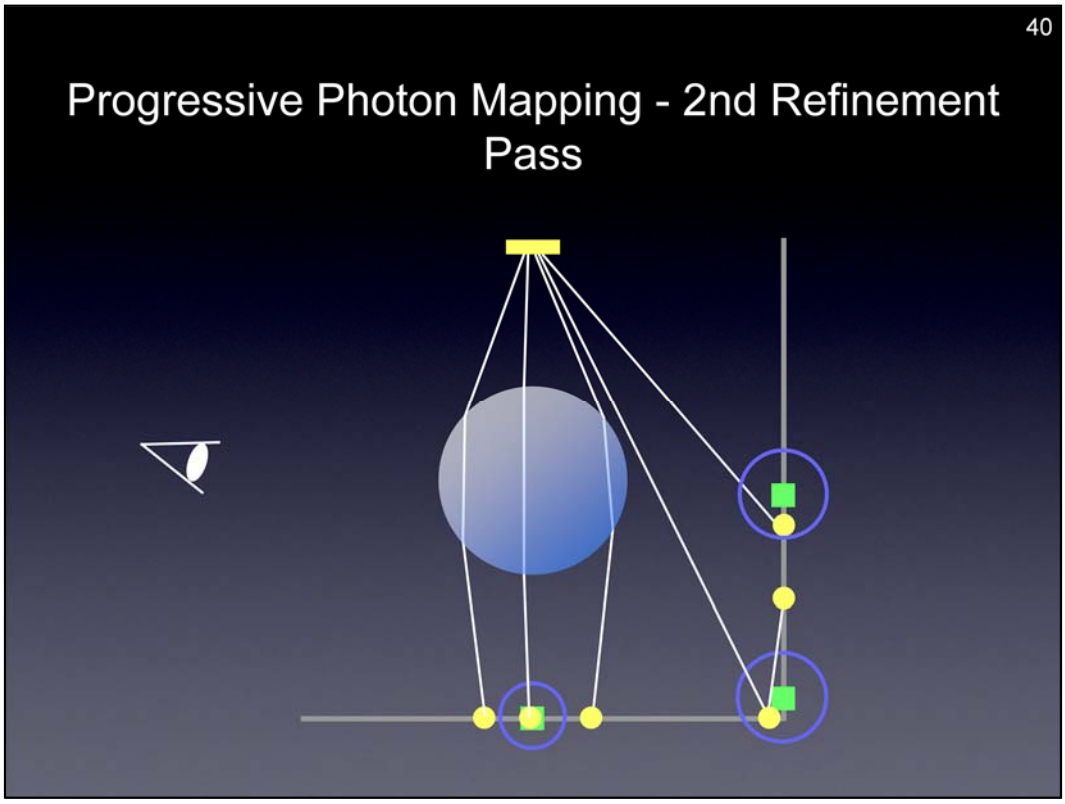
In each refinement pass, we first trace photons in the same way as the standard photon mapping. We then find photons within the radius of each point.



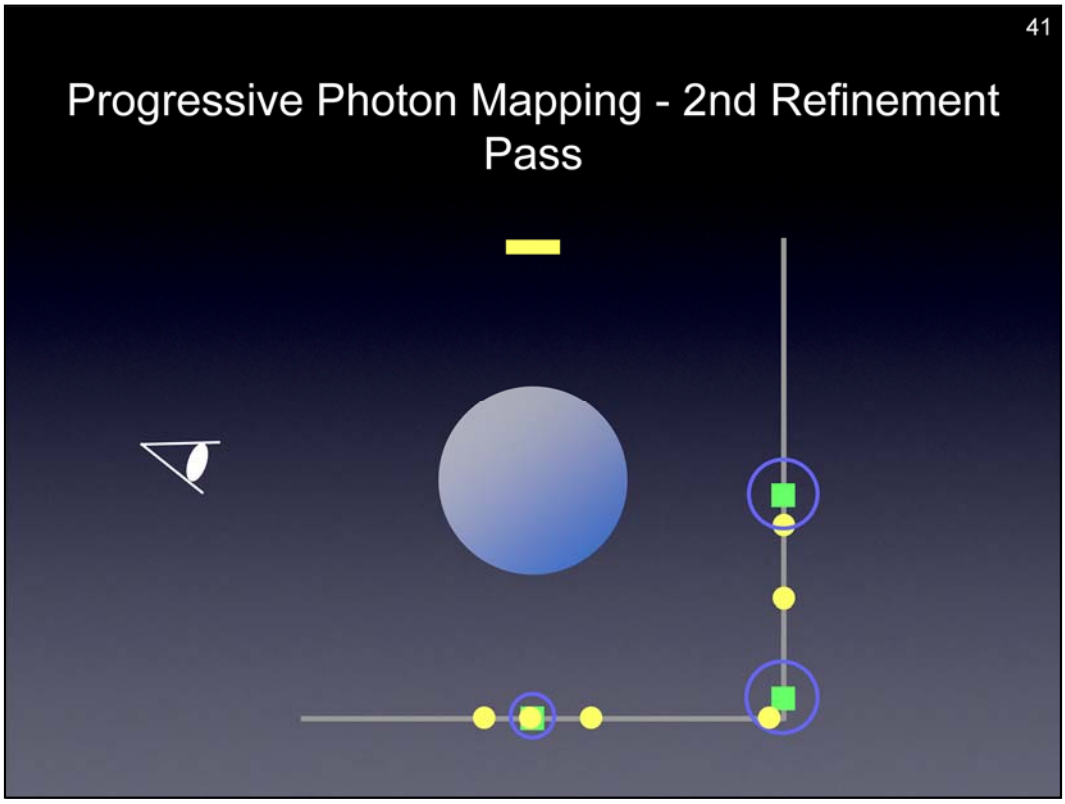
Based on the nearby photons we update the statistics of each point. This includes reducing the radius as shown on the slide.



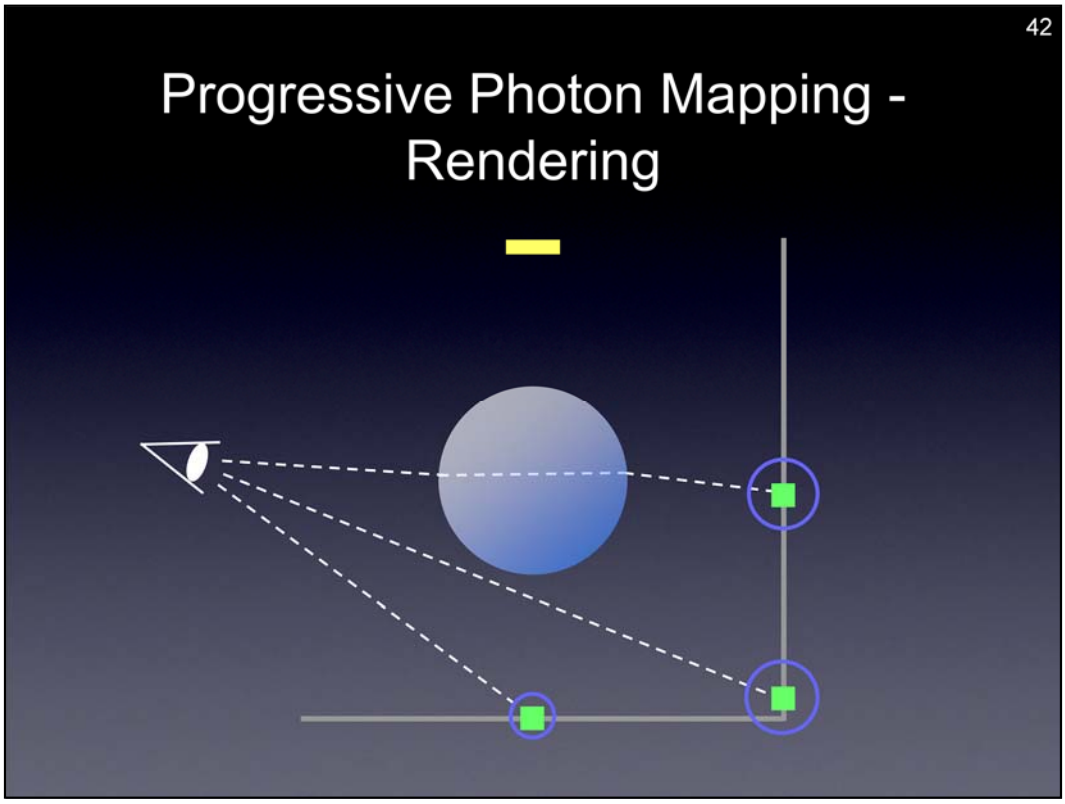
We then discard all photons and prepare for the next iteration.



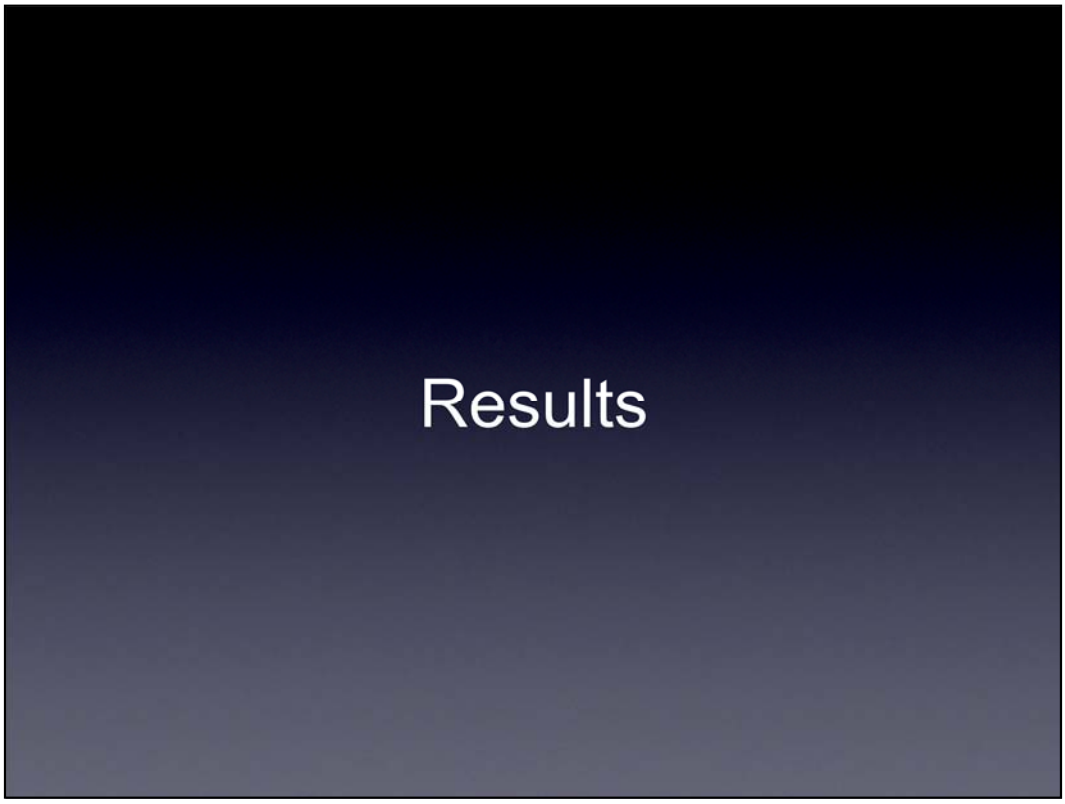
The succeeding refinement passes proceed exactly in the same way,



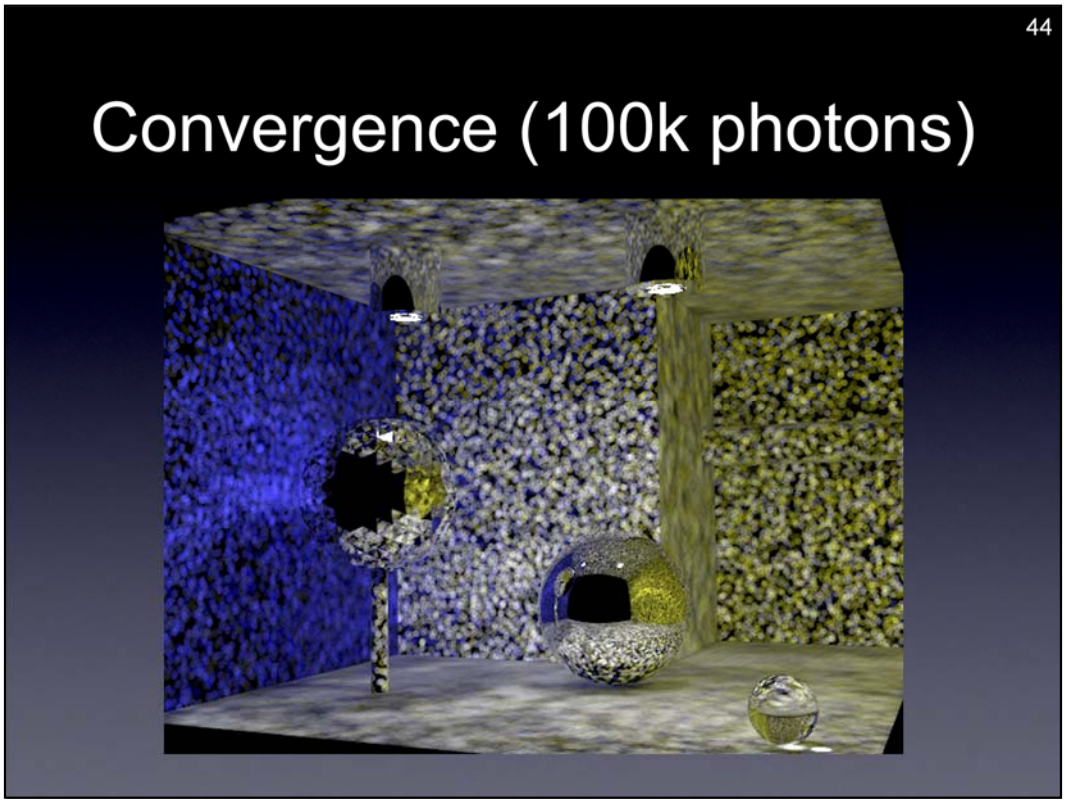
but we use updated radii and statistics.



Finally, we can render the image at any iteration by estimating the radiance at each point.

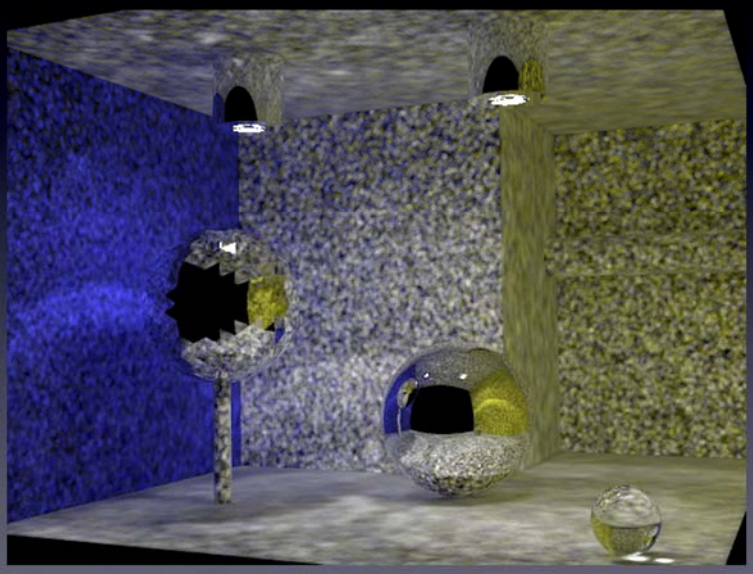


Now, I am going to talk about our results.

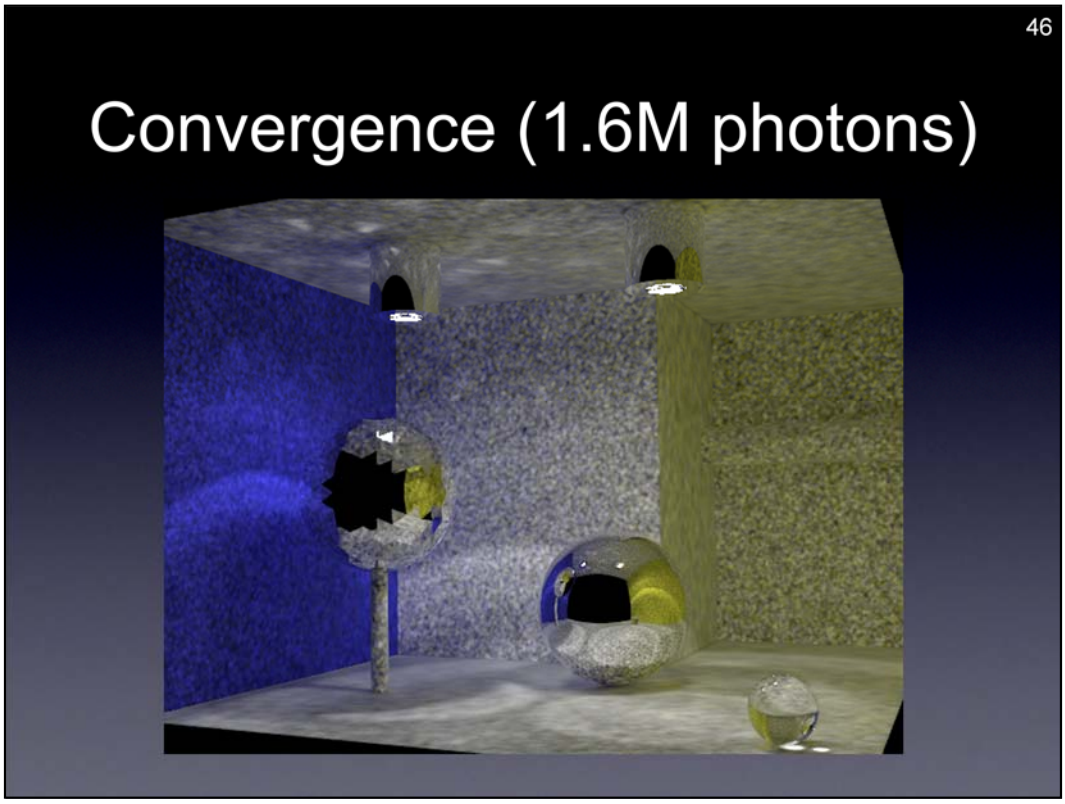


First I will show how images rendered by our method converges to the correct solution. This image is rendered using one hundred thousand photons.

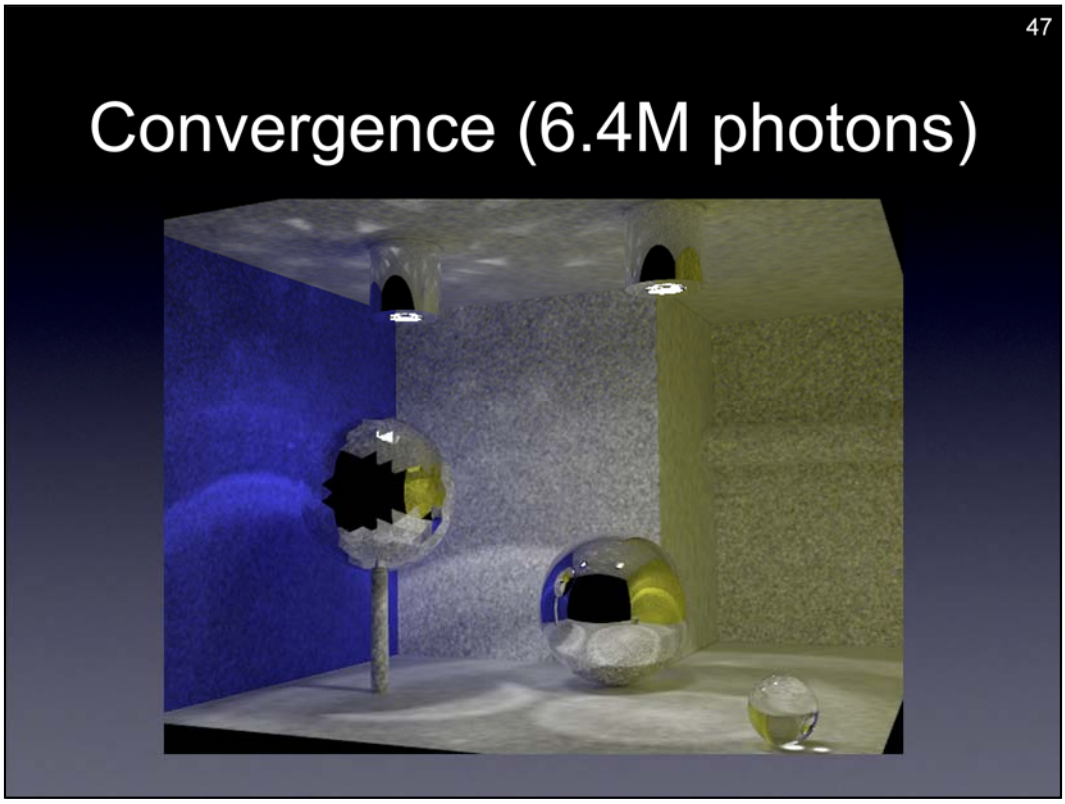
Convergence (400k photons)



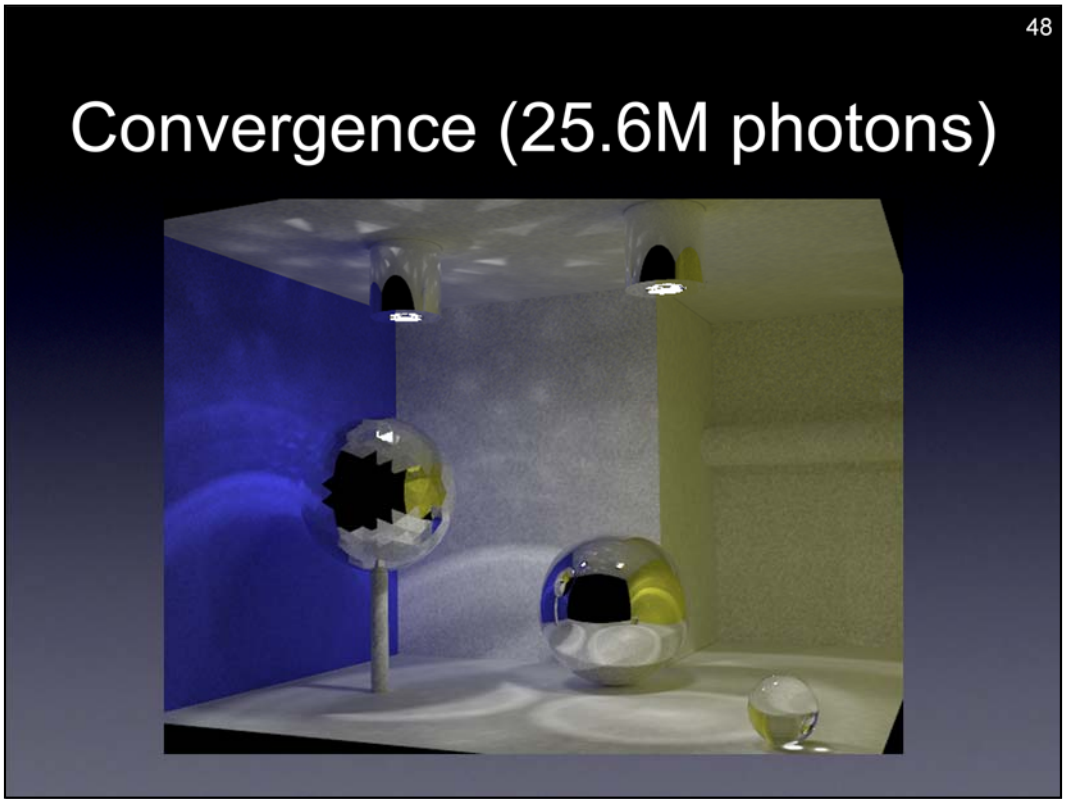
as you increase the number of photons, thereby adding more refinement passes,



we can obtain more details and smoother result. Note that even with a relatively low number of photons,



the image already gives us an idea of the illumination in the scene.



and finally this image is rendered using about twenty-five million photons in total. Note the absence of bright noisy pixels in the image sequence just shown.

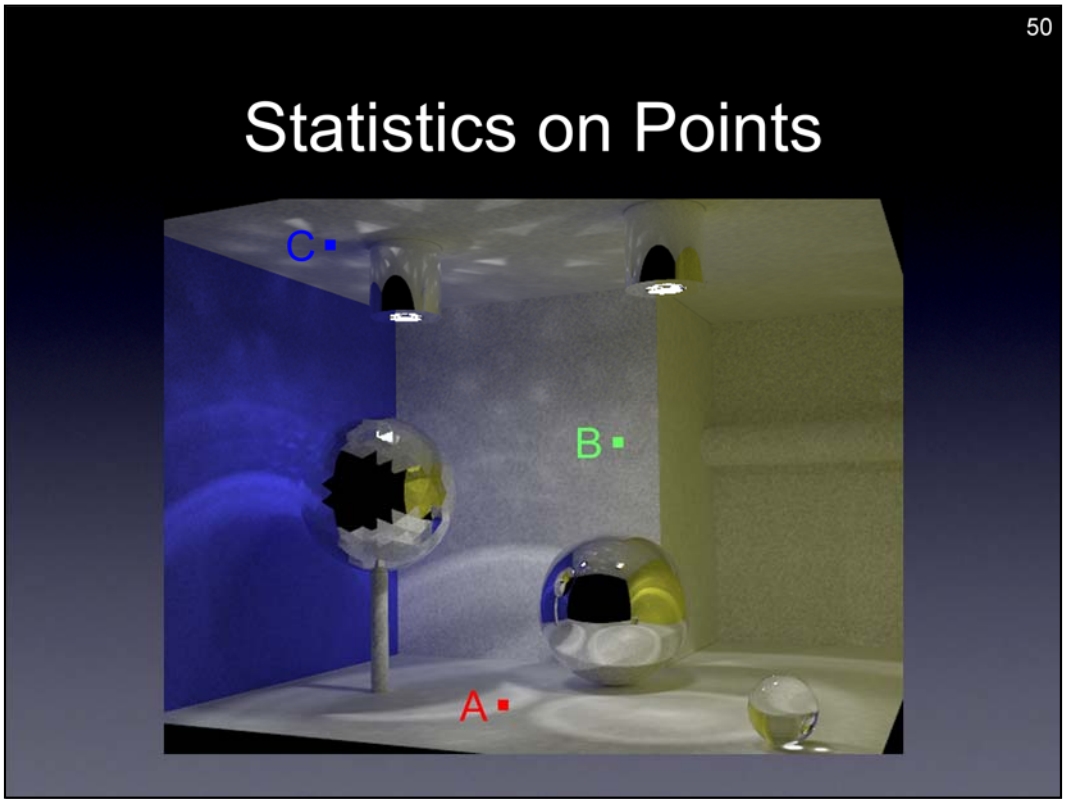
Robustness

$$L(x,\vec{\omega}) = L_e(x,\vec{\omega}) + \int_A L(z \to x)\, f_r(x,\vec{\omega}, z \to x)\, \boxed{G(x,z)}\, dA_z$$

$$G(x,z) = \frac{\cos(n_x,\vec{\omega})\cos(n_z,-\vec{\omega})\,V(x,z)}{\boxed{||x-z||^2}}$$

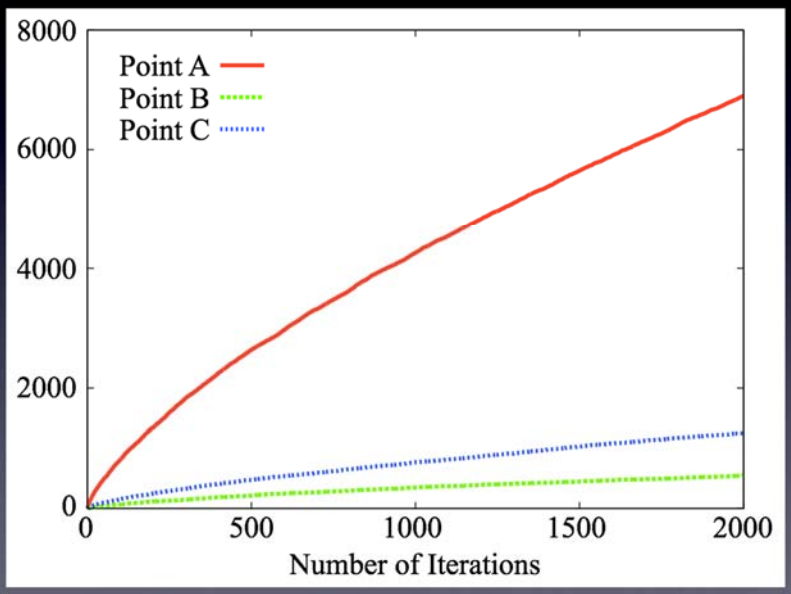
- Progressive photon mapping avoids the singularity since no lighting is sampled explicitly

The reason for this is that progressive photon mapping avoids the singularity in the geometry term of the rendering equation by not sampling the light sources explicitly.



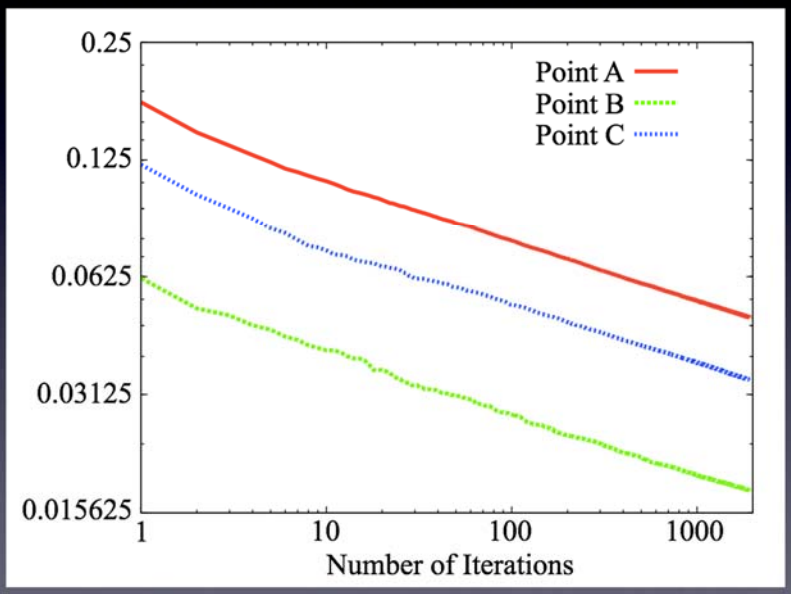
Let’s look at how the radius and the number photons change at different points on the scene. We plot graphs of the radius, the number of nearby photon, and the estimated radiance on the different point shown in here.

Number of Local Photons

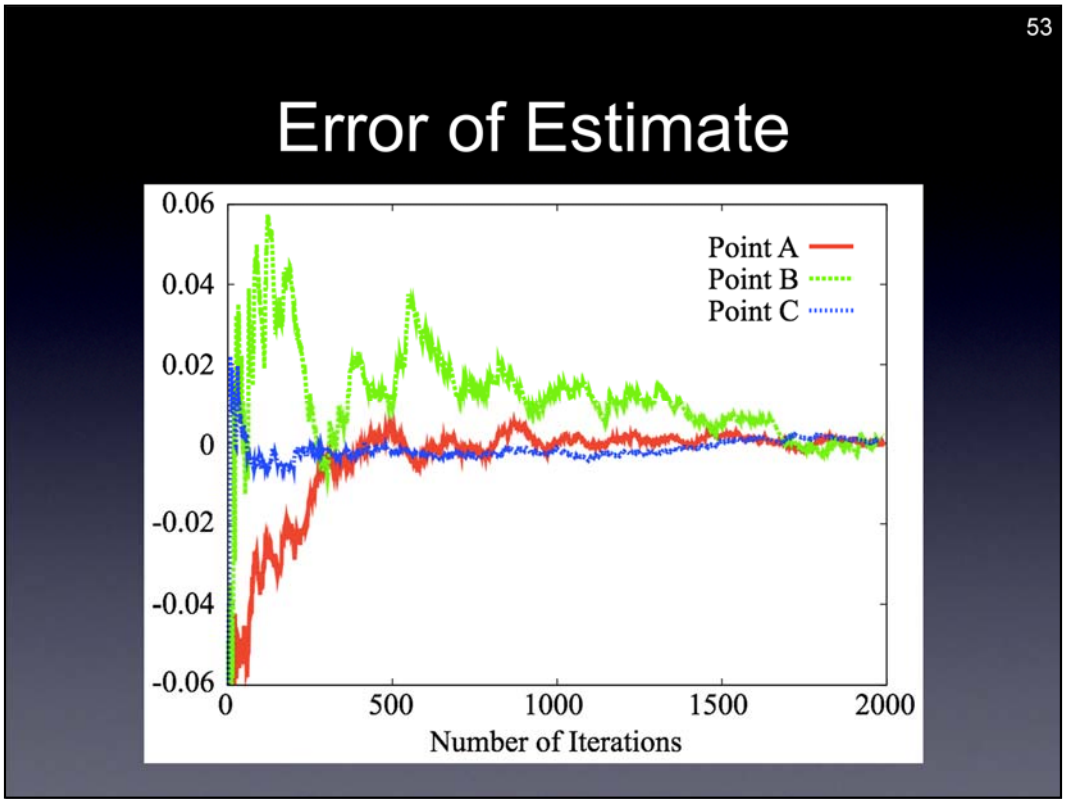


This graph shows the change of the number of photons within the radius of each point. As can be seen, the number of photons increases monotonically as we increase the number of iterations.

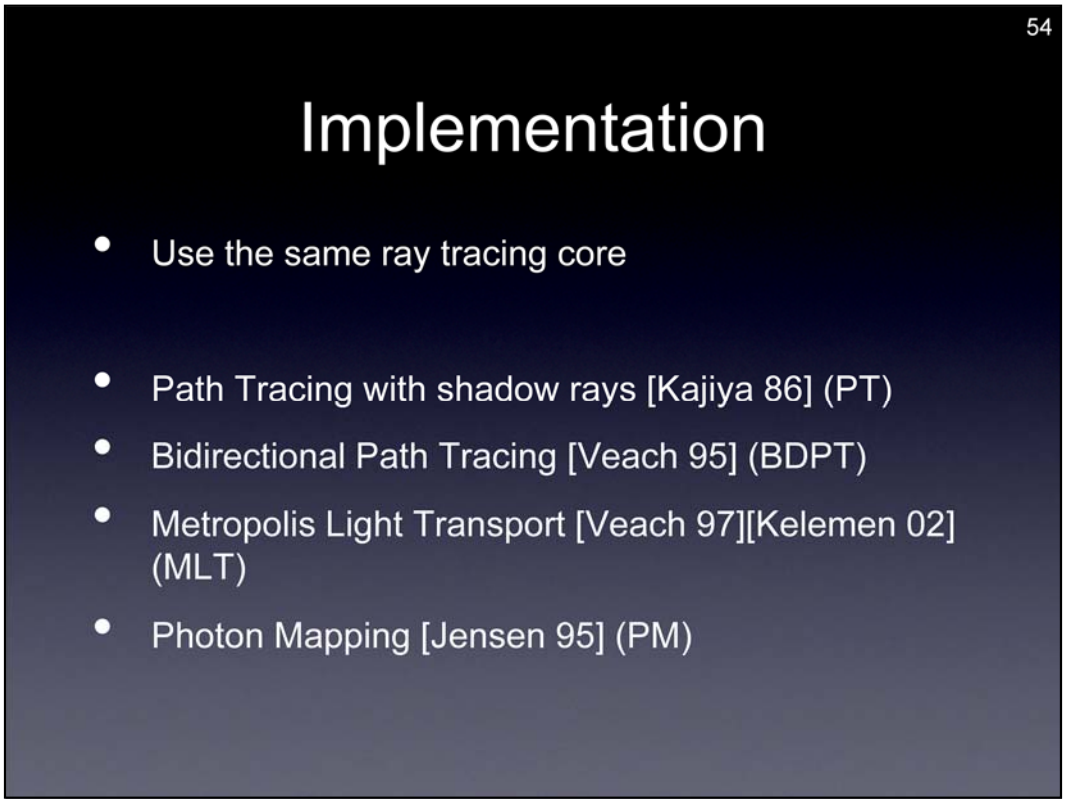
Radius



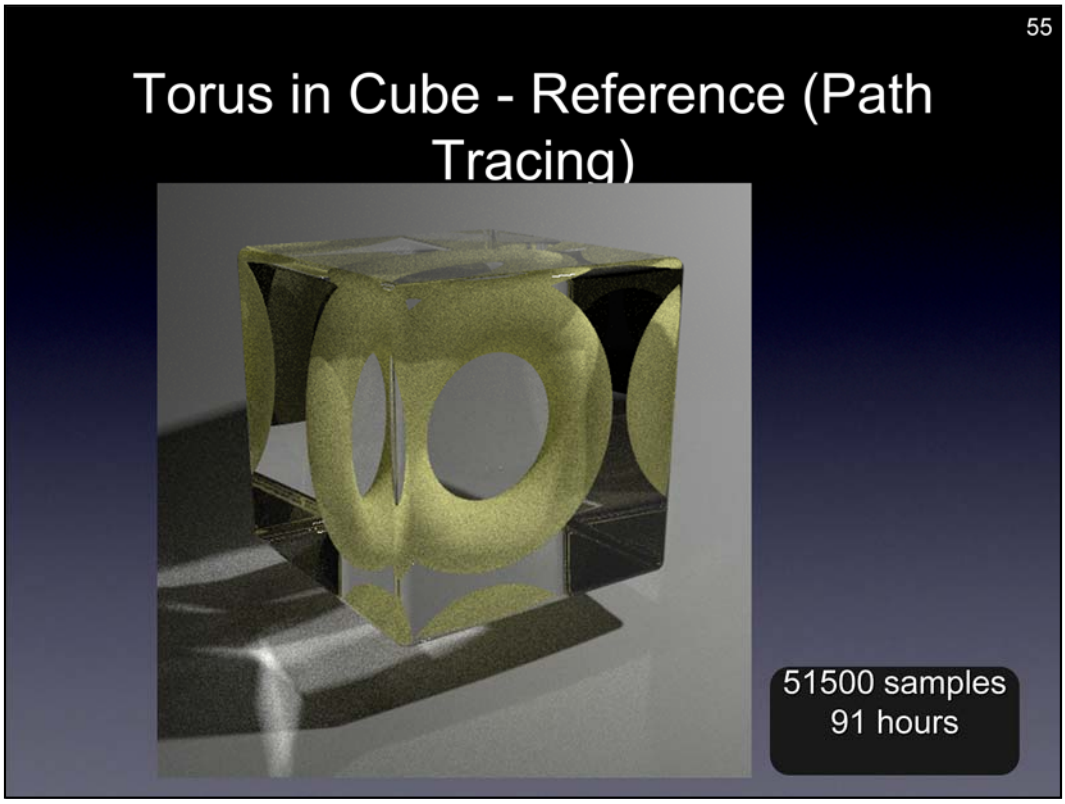
This graph shows the change of the radius. Note that the radius is monotonically decreasing as the number of iterations increase.



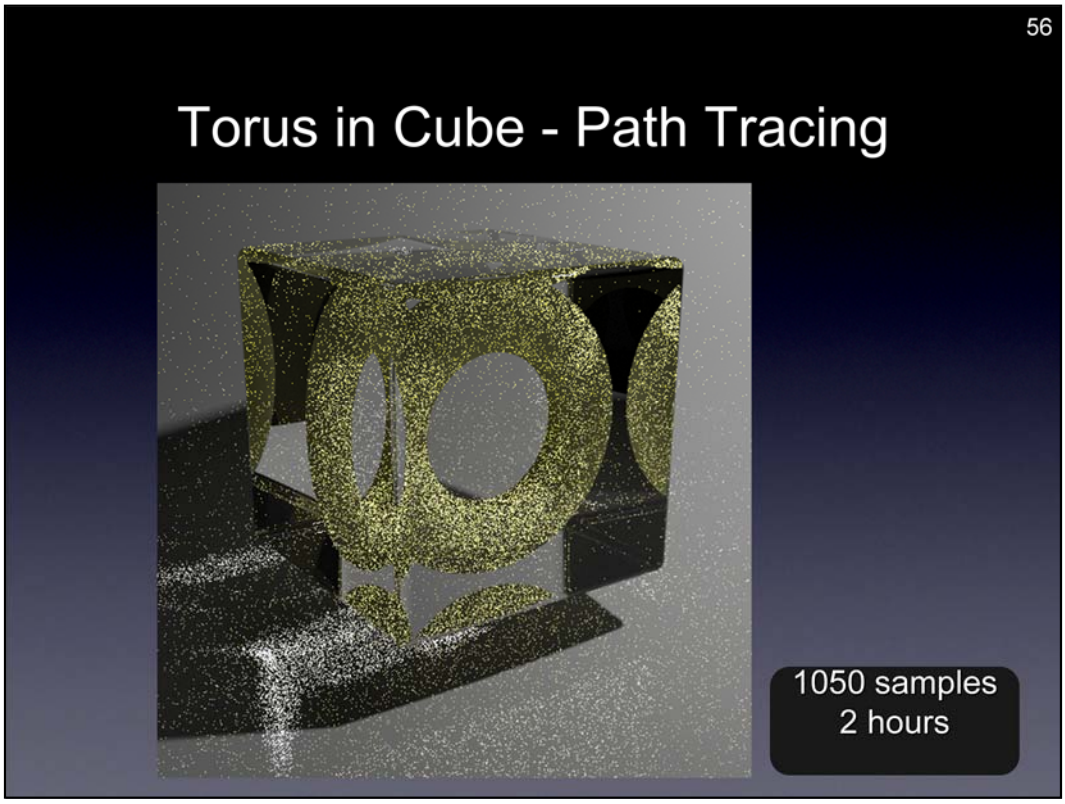
This graph shows the error of the radiance estimate. As can be seen, the radiance estimate converges to the correct solution.



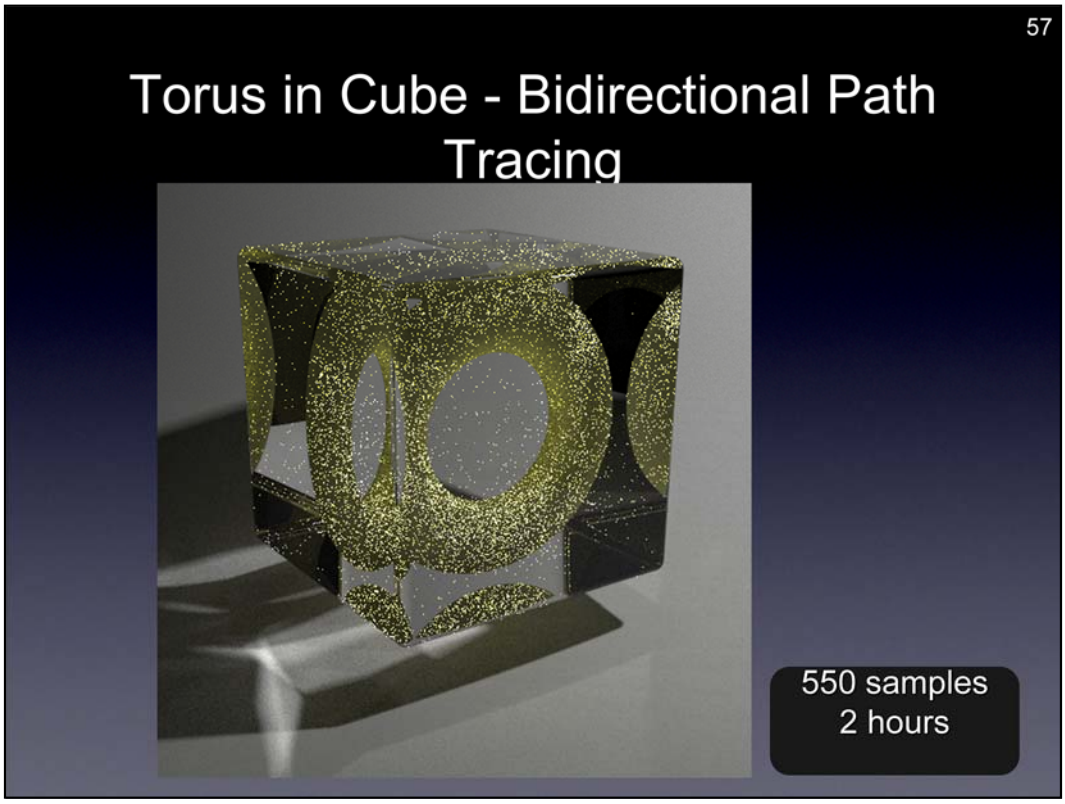
To compare our method with existing methods, we implemented these algorithms using the same ray tracing core.



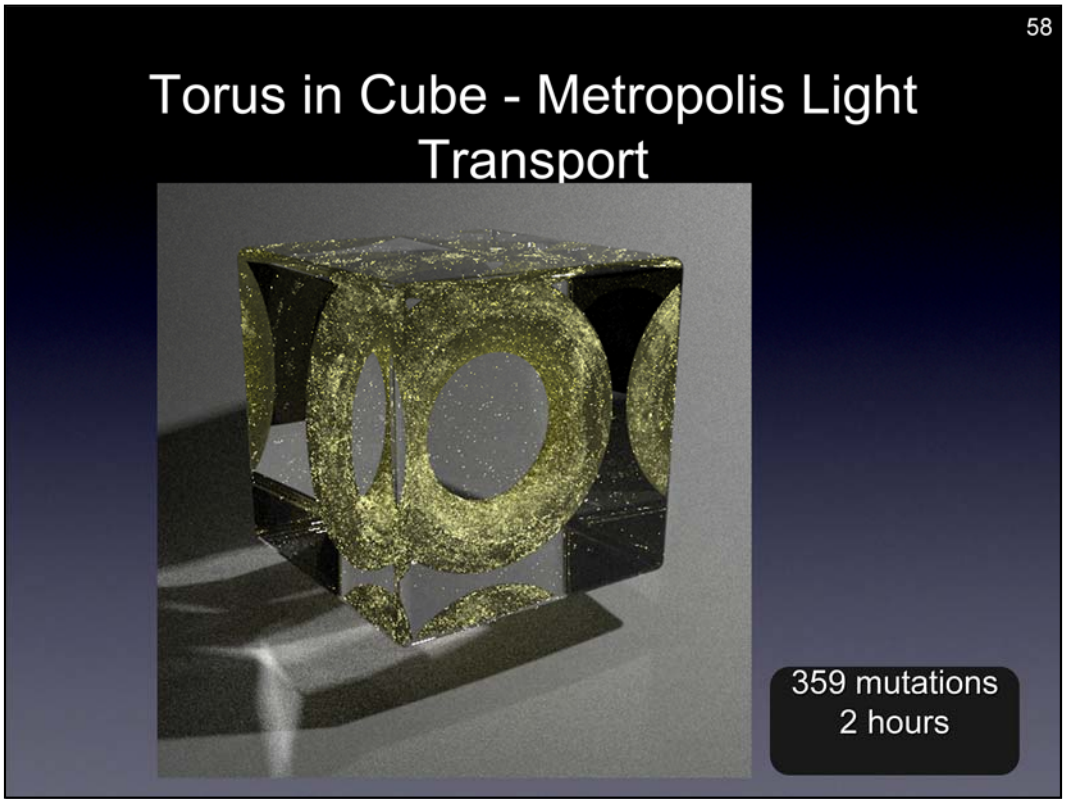
First we rendered a torus embedded in a transparent cube illuminated by sunlight. Note that all illumination on the torus is caustic and we see the torus though specular transmission of the cube, which is a SDS path. This is the reference solution with 91 hours of rendering using path tracing.



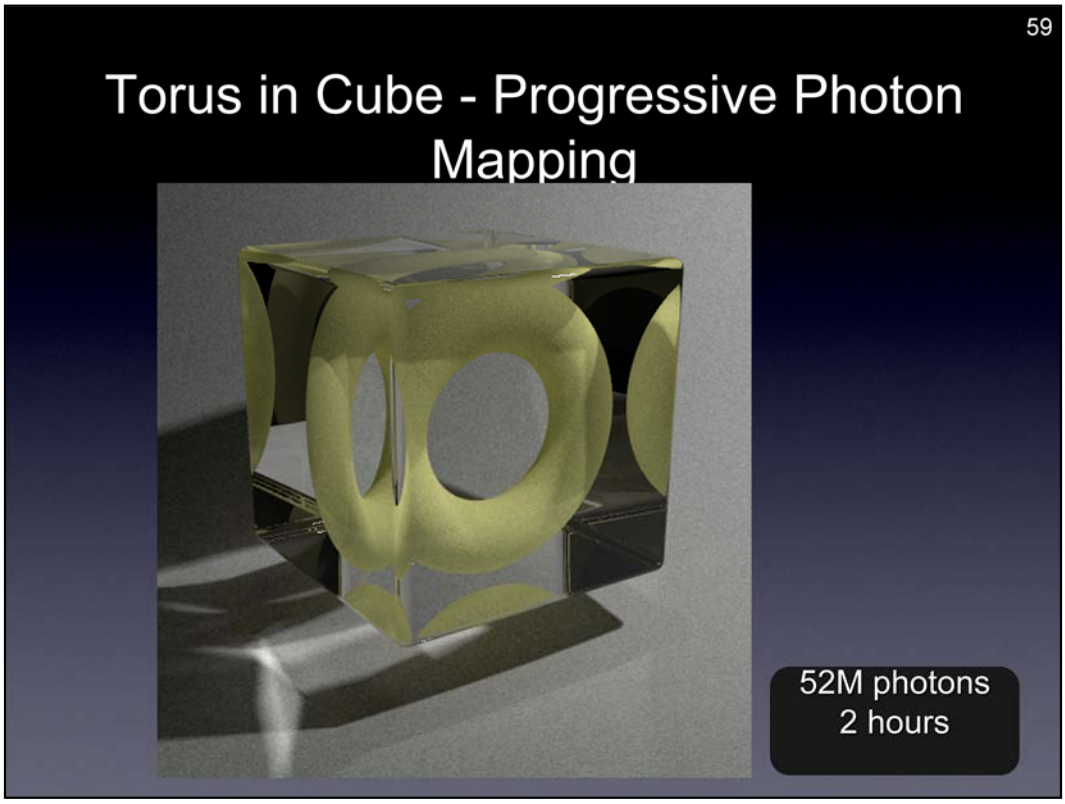
If we just use 2 hours with path tracing, the image looks very noisy.



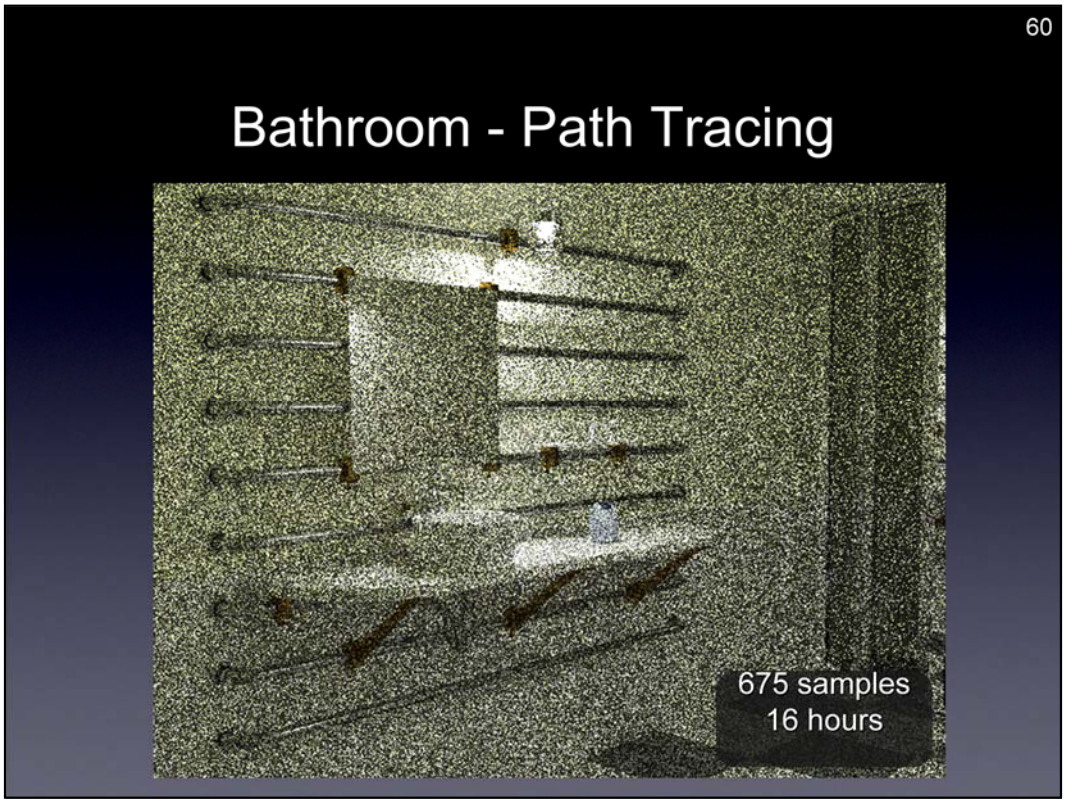
Bidirectional path tracing gives less noisy results especially for the caustics caused by the transparent cube, but the torus is still significantly noisy.



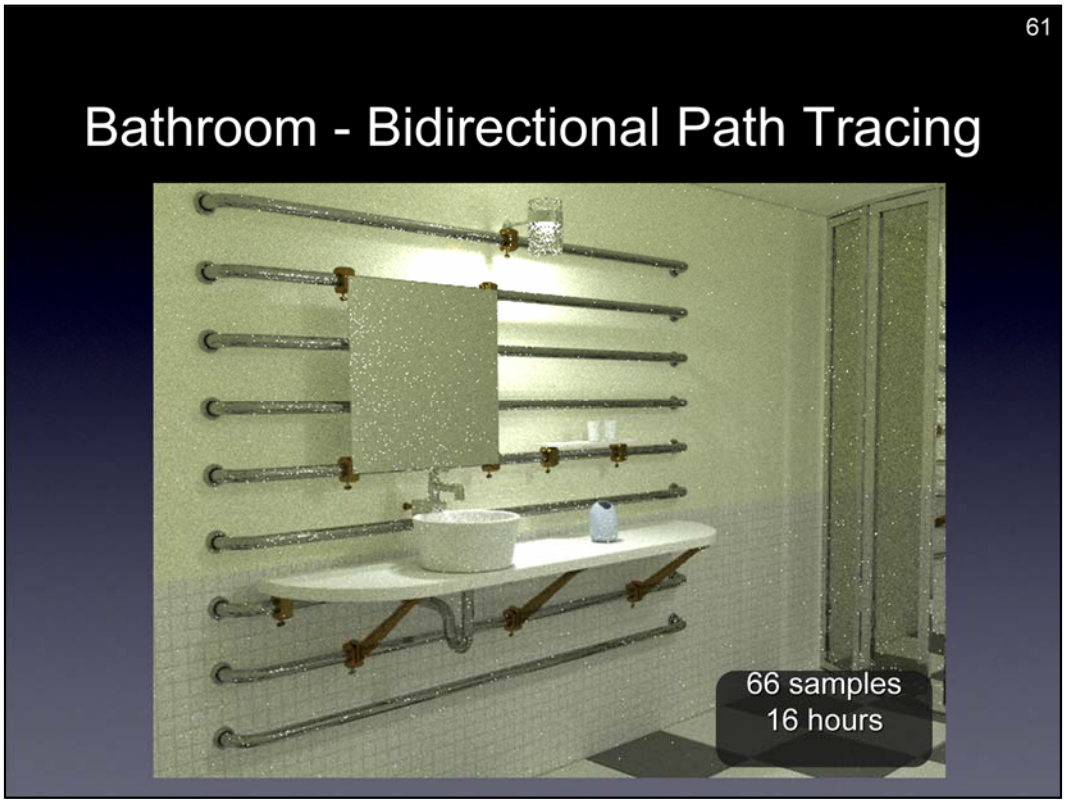
Metropolis Light Transport does not really work well either in this scene, but it just gives us a different type of artifact.



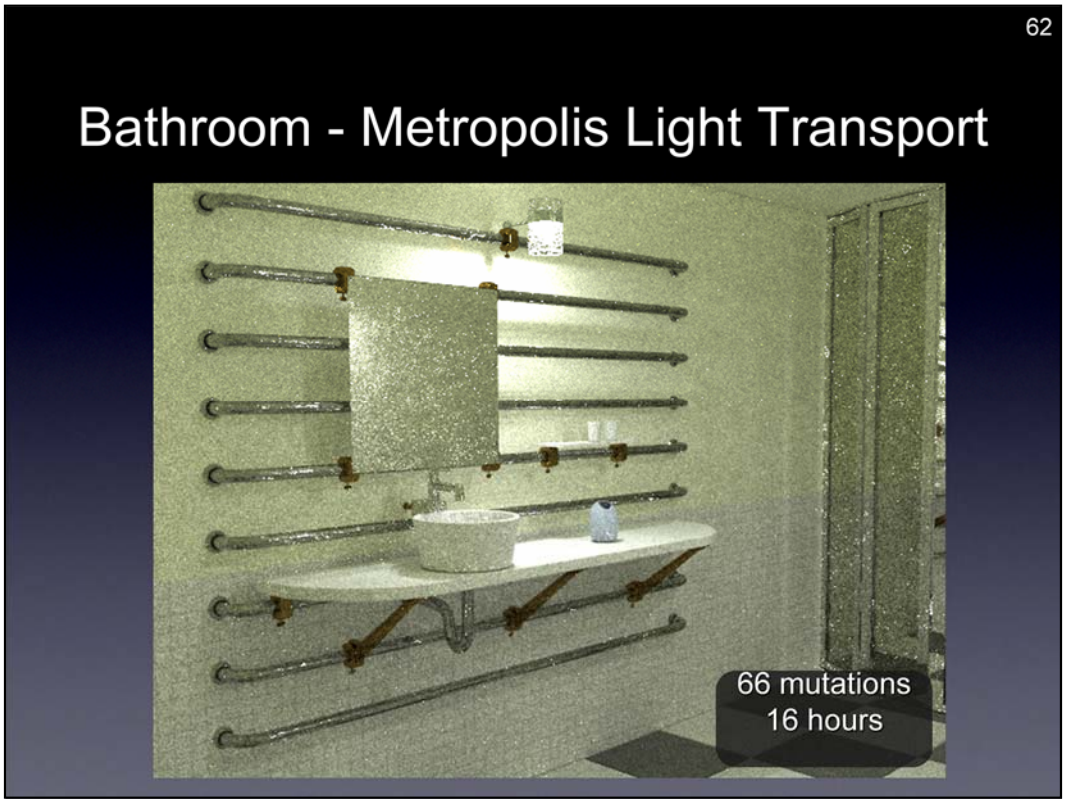
In the same rendering time, our method can handle illumination on the torus very robustly as well as caustics by the cube.



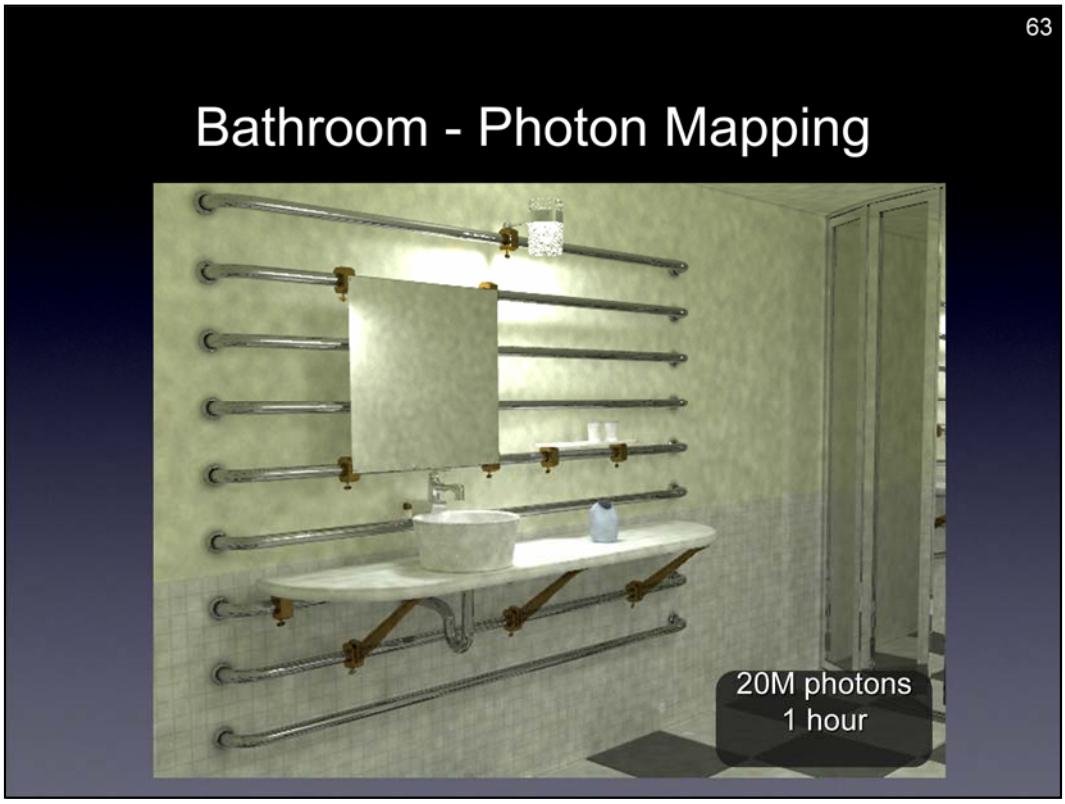
This bathroom scene shows an example of realistic lighting design. In this scene, there are two spherical light sources enclosed by glass casing, which is similar to a typical lighting fixture. Note that reflection on the mirror causes SDS paths. Path tracing results in noisy image because almost everything is illuminated by caustics in this scene.



Bidirectional path tracing gives you much better result in the same rendering time. However, note that the reflection of the light on the mirror is missing.

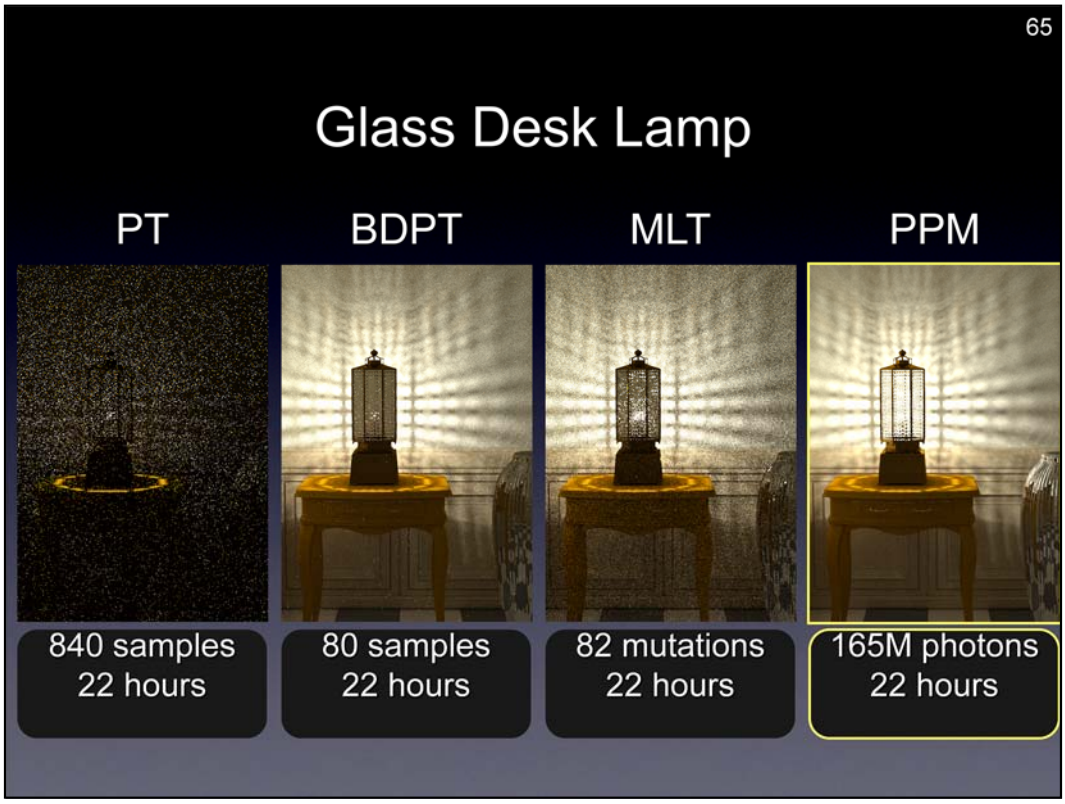


Metropolis Light Transport can capture some of reflections on the mirror, but the results looks still very noisy.

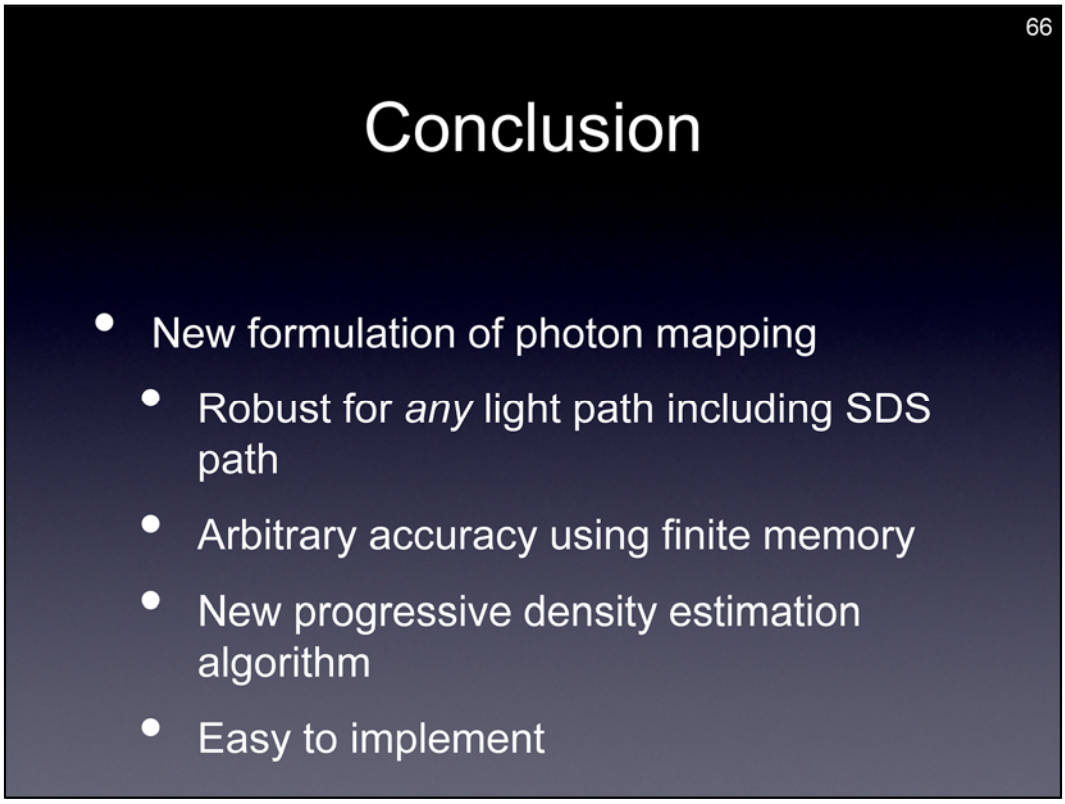


Here we used standard photon mapping. The number of photons is as large as we can use in 1GB of memory. Since the number of photons is limited by the amount of memory, the rendering time is actually faster. However the image looks blotchy because the number of photons is not enough to get rid of the noise. This means that the quality of the image is bounded by the available amount of memory.

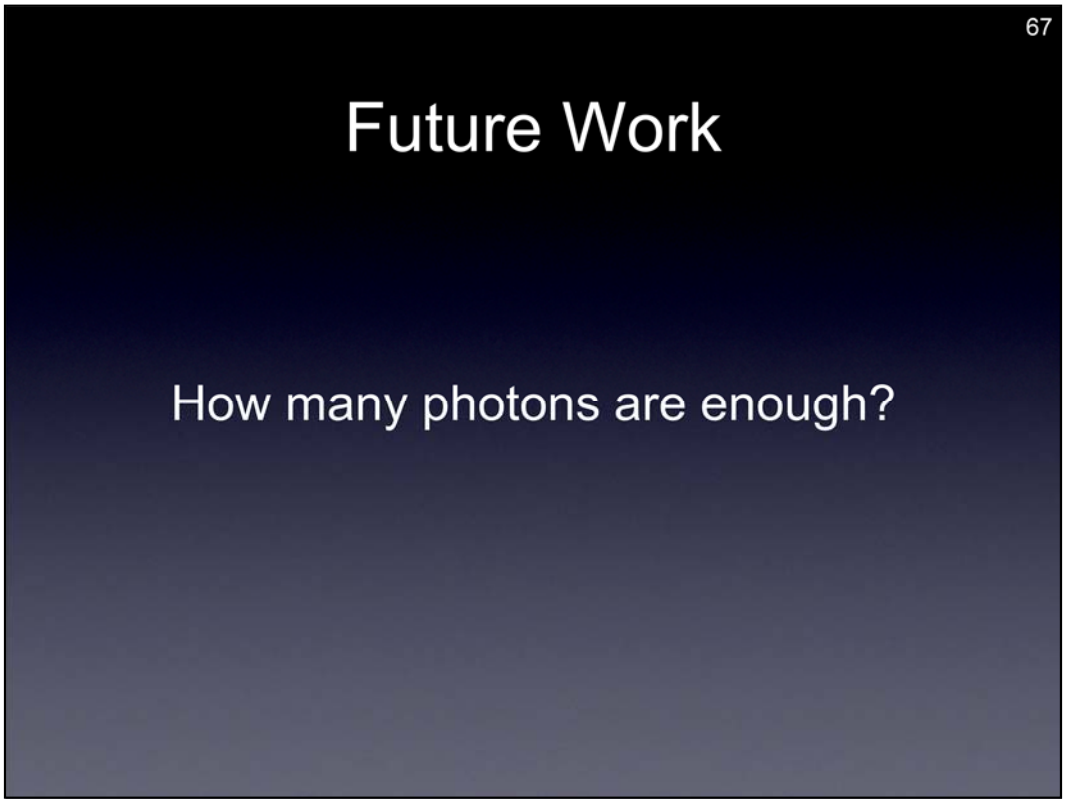
With Progressive Photon Mapping, we can use 612 million photons, which is equivalent to 30GB of photons without consuming that amount of memory. The image accurately captures the reflection in the mirror, as well as all other fine scale illumination details.



Finally, we rendered a glass desk lamp to show the robustness of our method. The results using existing methods are either too noisy, or they cannot handle the refraction through the lamp. Only progressive photon mapping is able to render this scene without noise.



In conclusion, we have presented a new formulation of photon mapping, called progressive photon mapping. Our algorithm is robust and it can compute all types of light transport with arbitrary accuracy using a finite amount of memory. To achieve this, we have introduced a new progressive density estimation algorithm, which is easy to implement. We believe that our method has a lot of interesting future work,



but the most significant question we hope to answer is ‘how many photons are enough for a given error criterion?’

.

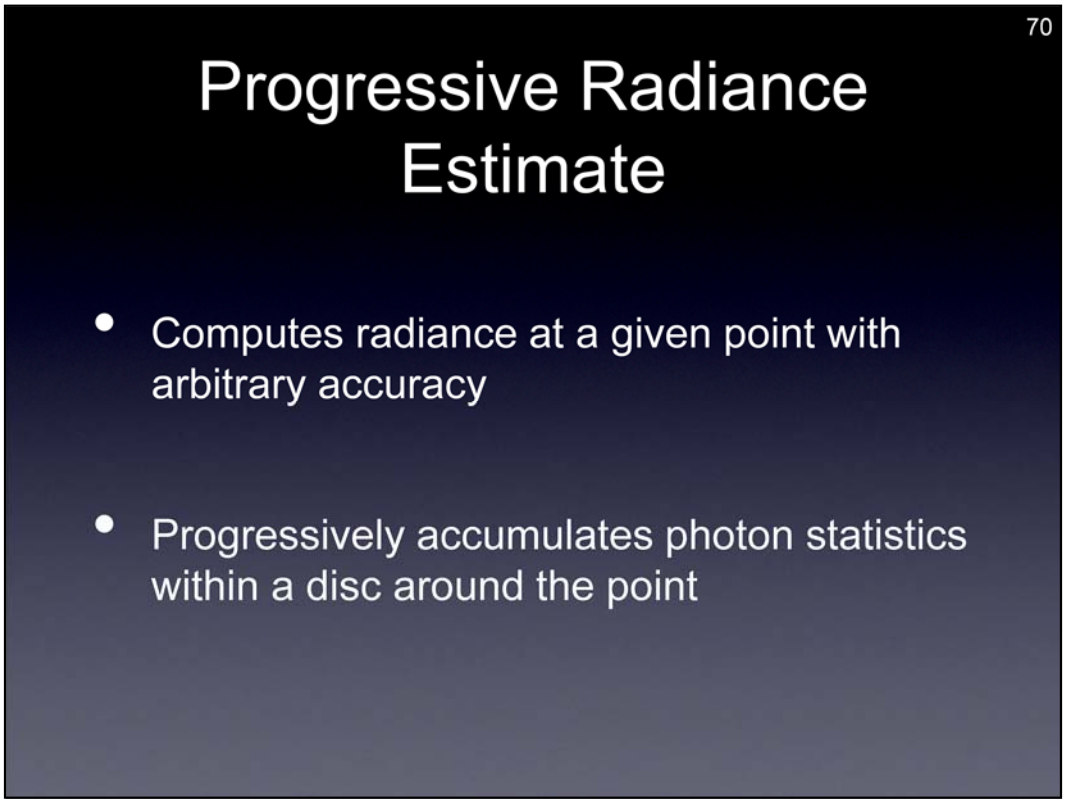
Acknowledgements

- NSF grant CPA 0701992
- Youichi Kimura (modeling)
- UC San Diego graphics lab members

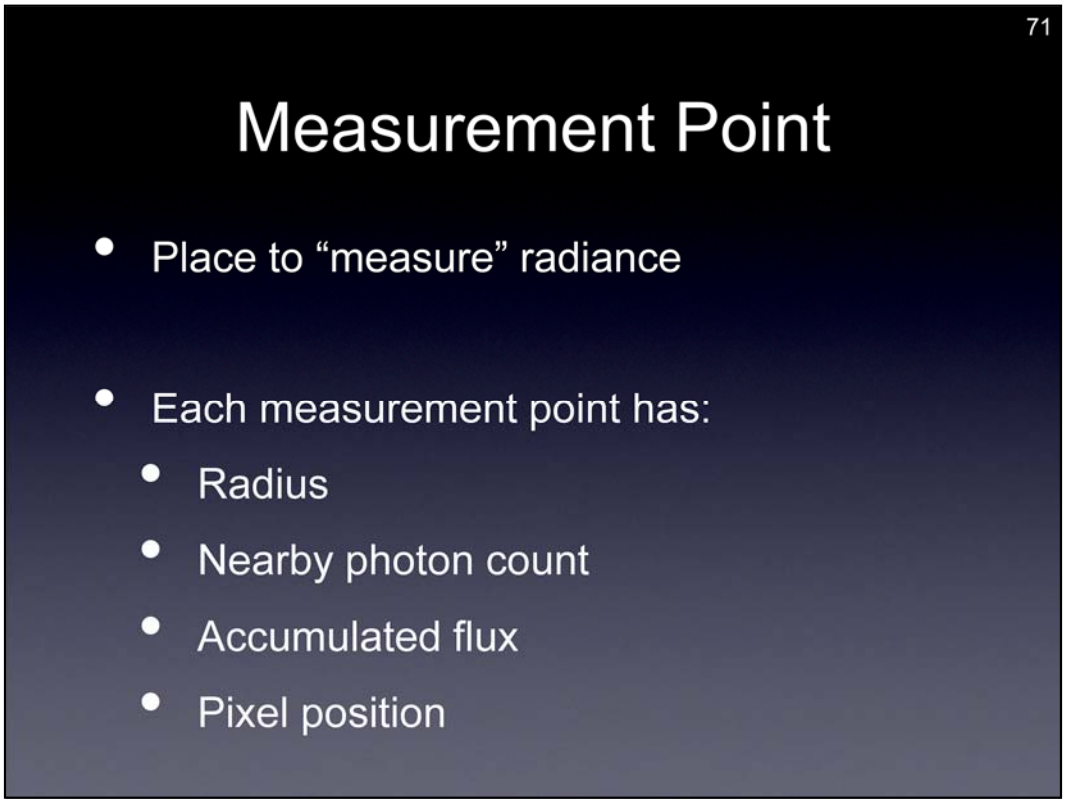
Here are acknowledgments, and thank you for your attention.

Our method is the only method that can robustly handle this difficult, yet simple illumination setting. We believe that our method is a robust alternative to render accurate images over existing unbiased Monte Carlo method.

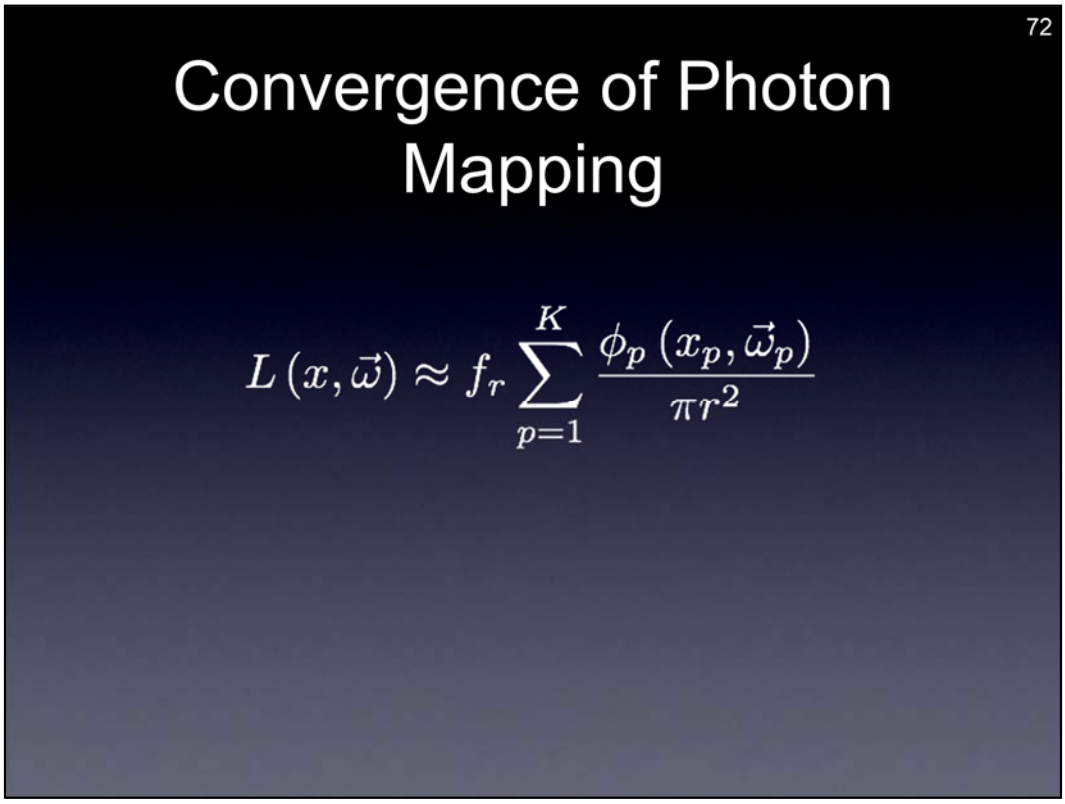
m



The goal of progressive photon mapping is to compute radiance at a given point with arbitrary accuracy. We achieve this by progressively accumulating photon statistics at



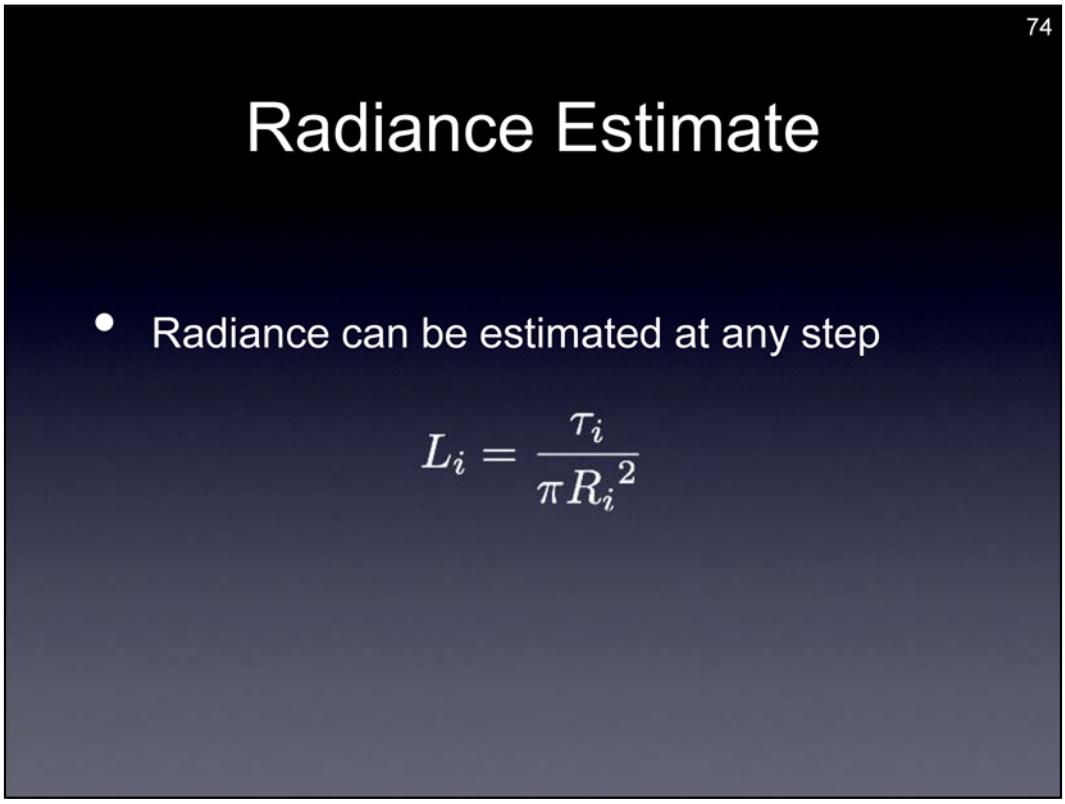
Measurement points generated in the first pass are places to measure radiance. Generating measurement points is typically done by ray tracing for rendering images. Each measurement point stores radius to search nearby photons, the number of nearby photons, accumulated flux of nearby photons and its pixel position.



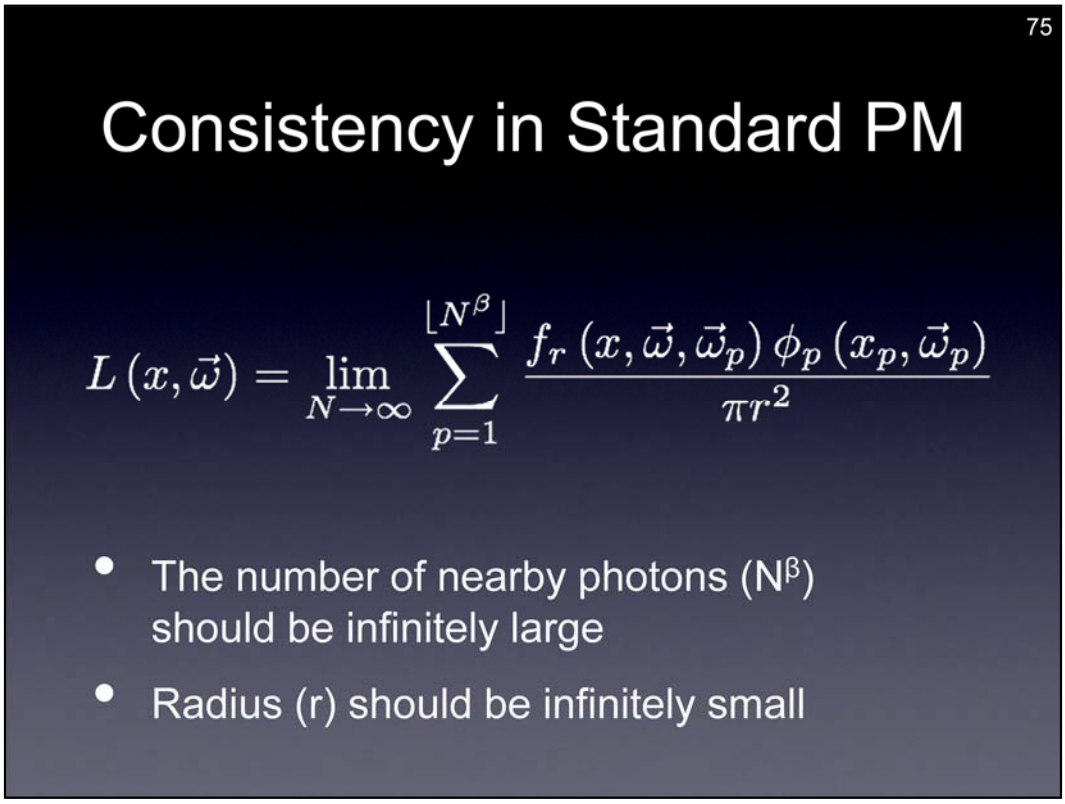
To be more concrete, radiance computed from neighboring photons converges to the correct value based on the rendering equation if we use an infinite number of photons. Of course, directly utilizing this property is not practical because an infinite number of photons requires infinite amount of memory to store.

Future Work

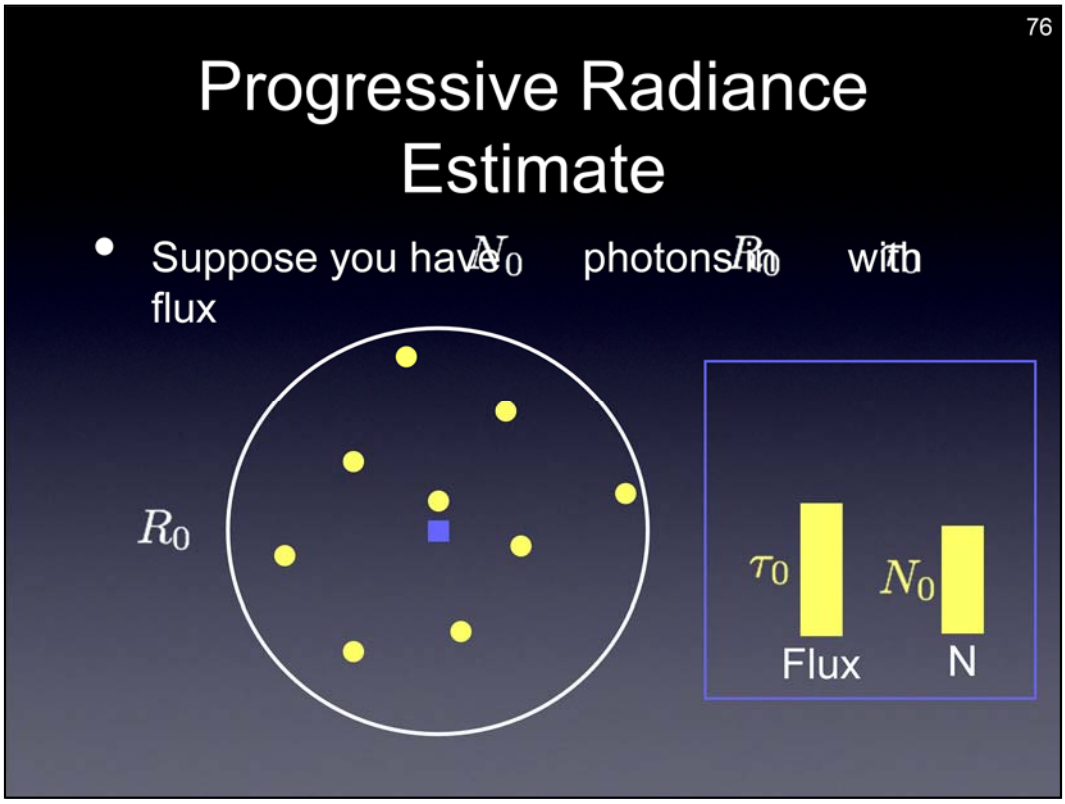
- Application to participating media
- Application to subsurface scattering
- Spectrum sampling
- Adaptive photon tracing
- GPU implementation
- Automatic parameter tuning
- ...



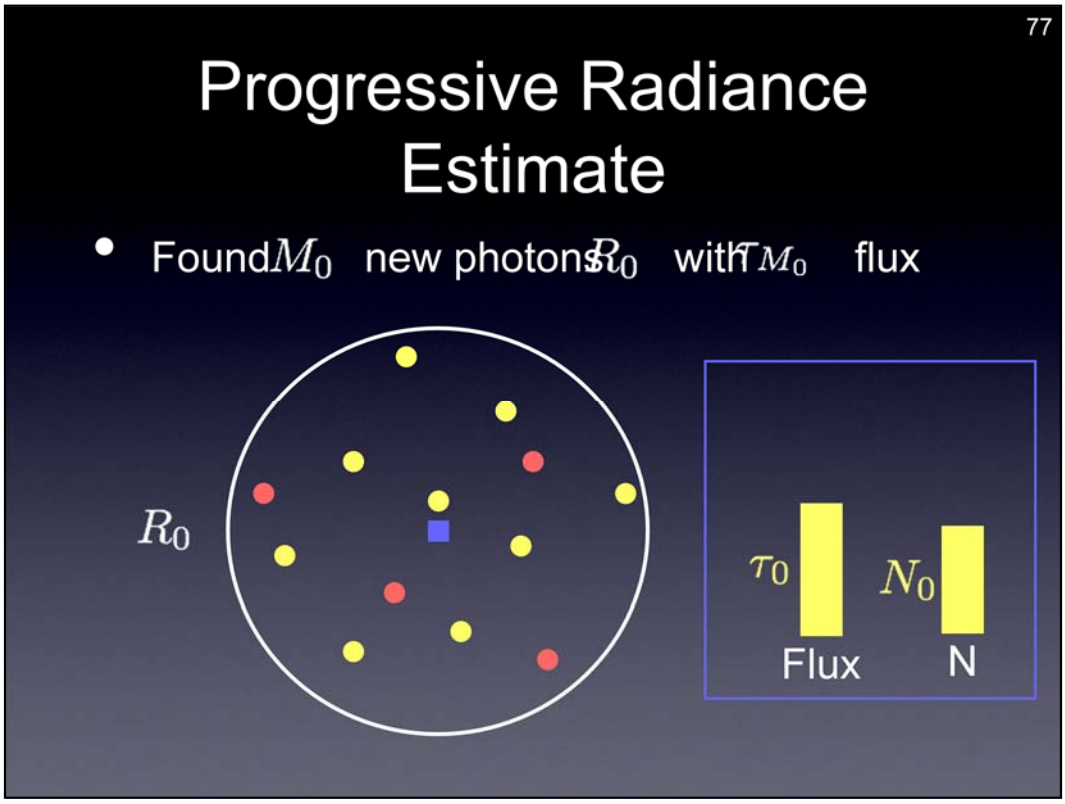
Since flux and radius is available in every refinement, we can estimate radiance to show intermediate images to user. The equation to compute is exactly the same as standard photon mapping, except it uses accumulated, but unnormalized photon flux, which needs to be divided by the number of emitted photons so far over all refinement passes.



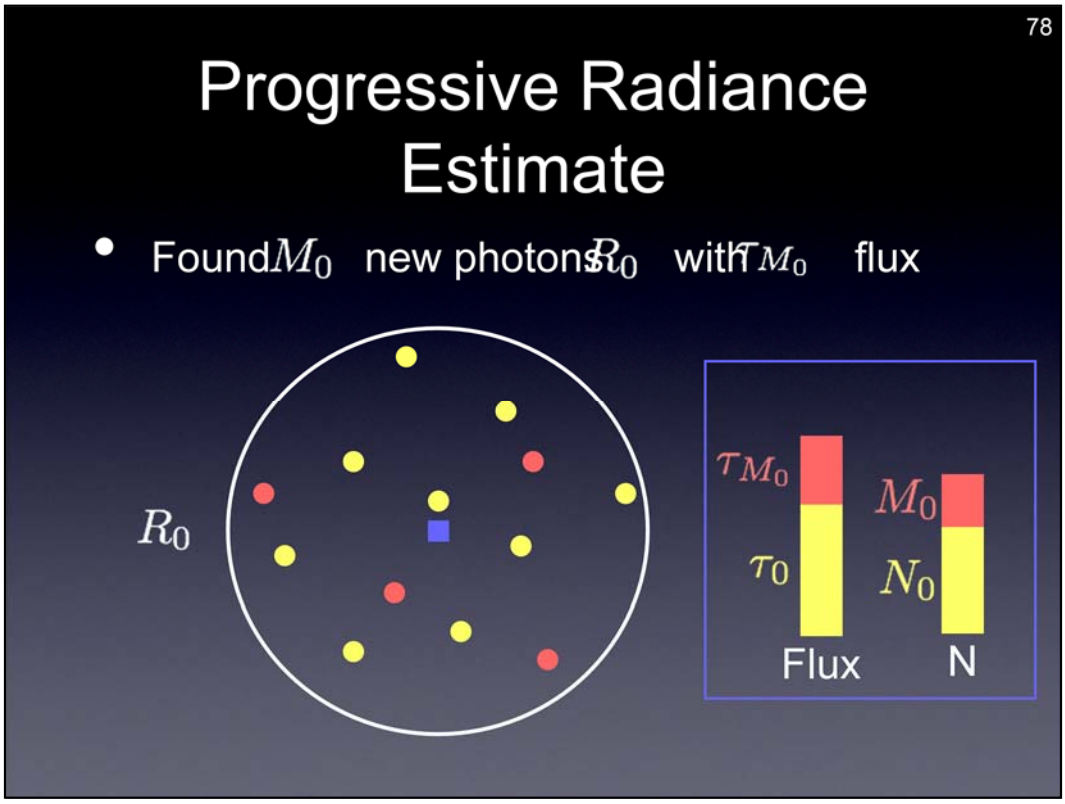
In order to describe how we update statistics on each measurement point, let me recap the consistency of the standard photon mapping. This equation basically says, photon mapping will give us correct radiance if the number of nearby photons is infinitely large and the radius that contains nearby photons is infinitely small.



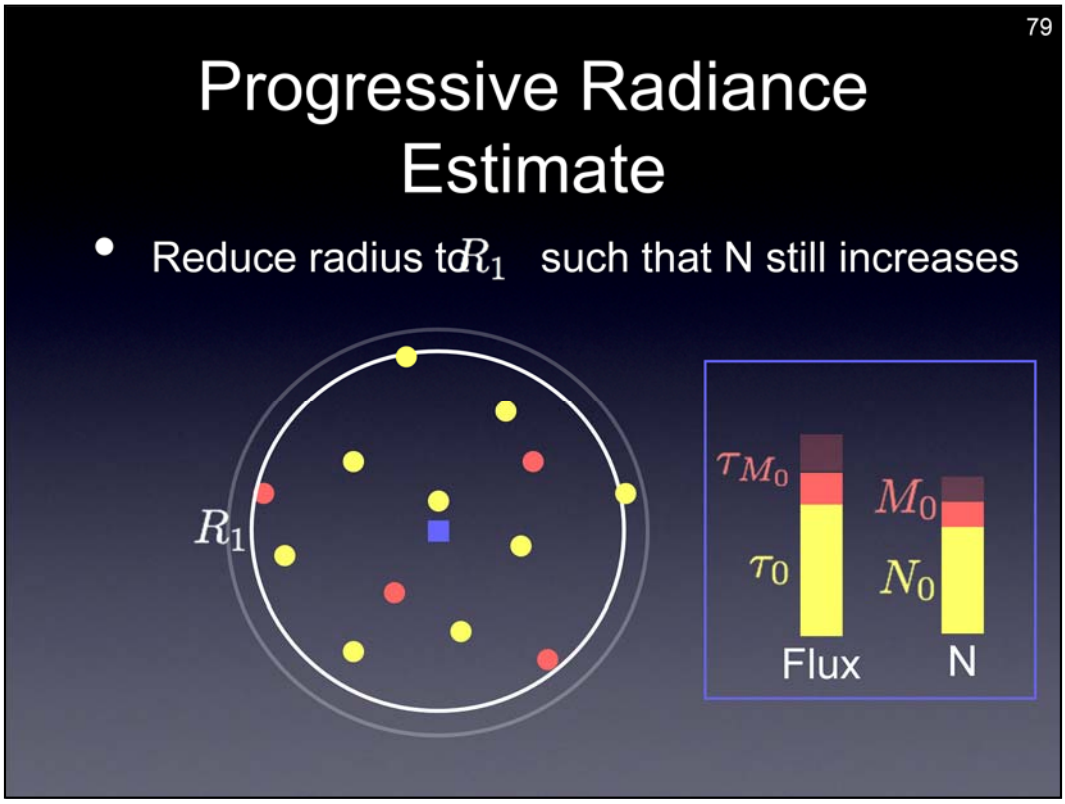
In order to understand how this can be achieved, let's look a single measurement point in more details. Suppose that you already have N0 photons within the radius of R0 with tau0 as flux.



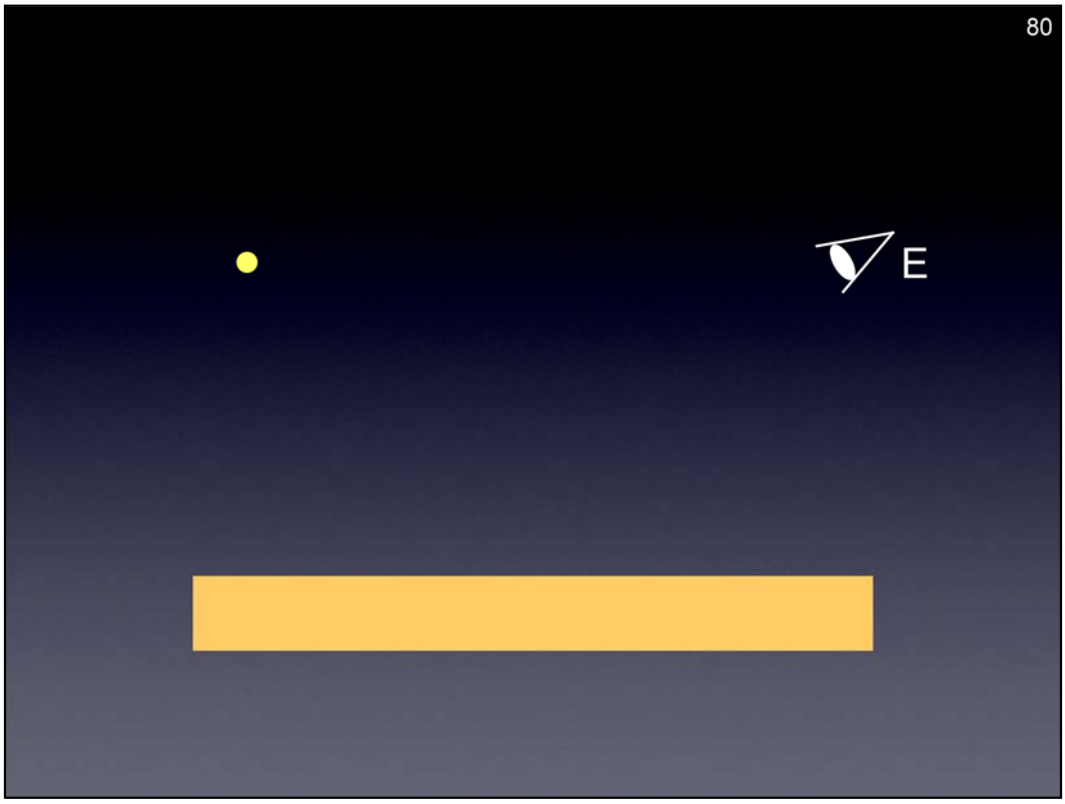
Now, say we find M0 new photons by shooting photons in this iteration.



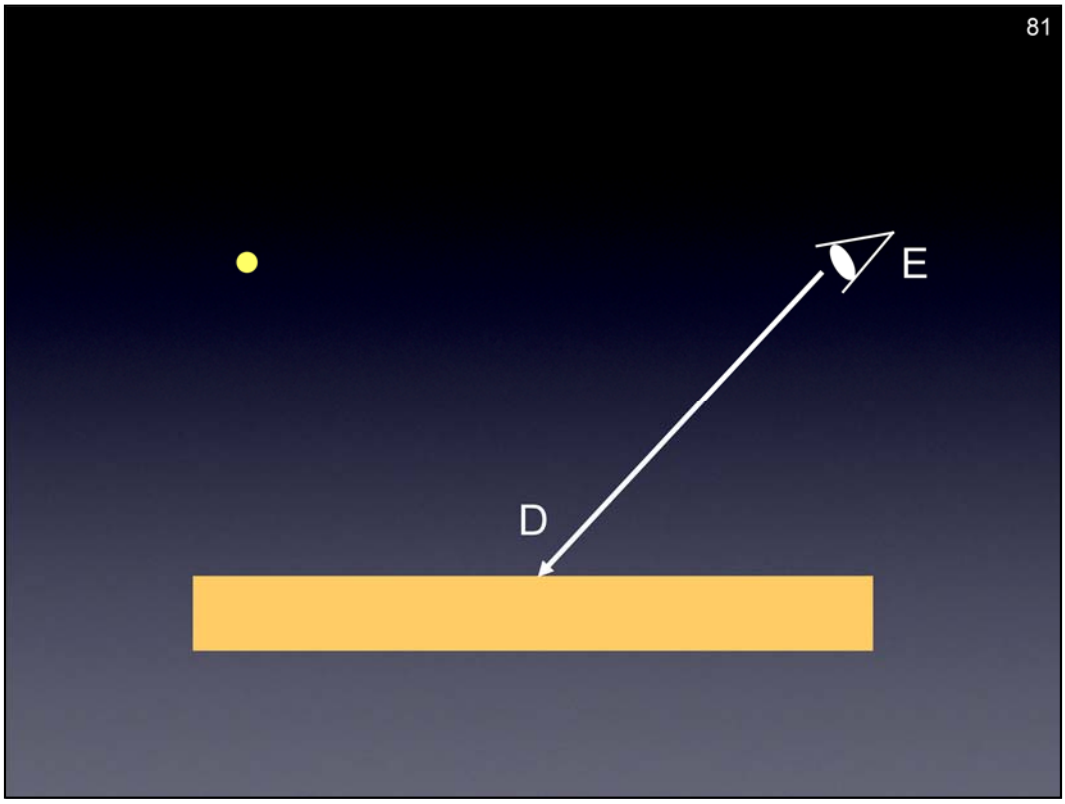
Since every photon is in the radius, we can simply accumulate flux and the number of local photons.



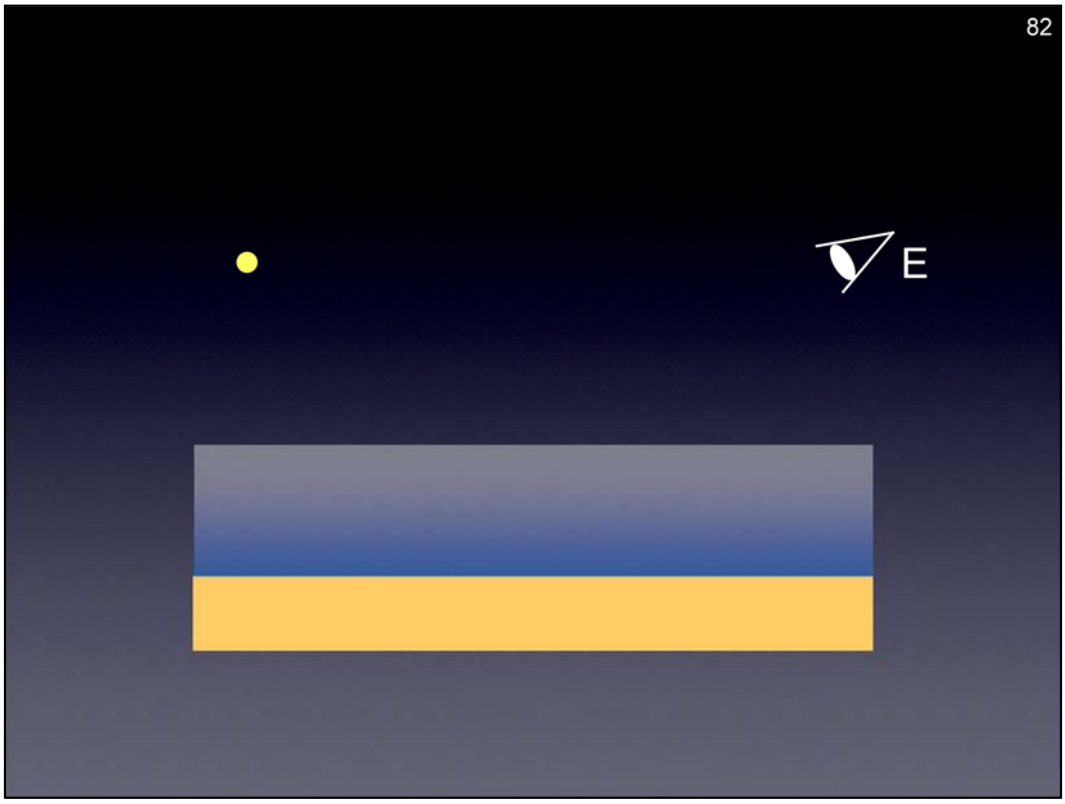
What we really want to do is to determine new radius R1, which is smaller than R0. R1 also needs to keep the number of photons N increasing as well as flux. In other words, we need to have non-zero gain in the number of photons and flux even after the reduction of radius.



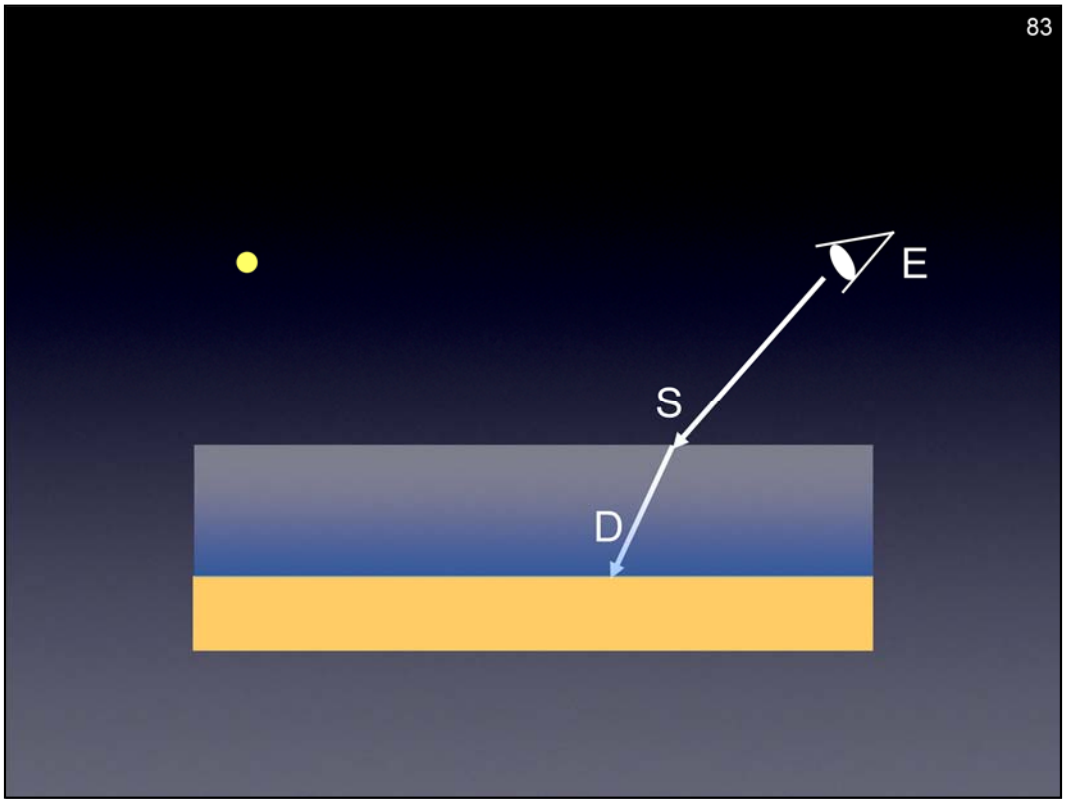
However, there is a type of path which is problematic. In order to see such example, let’s look at this simple scene, where a point light is illuminating a diffuse object. To compute the contribution from the light source to the eyes,



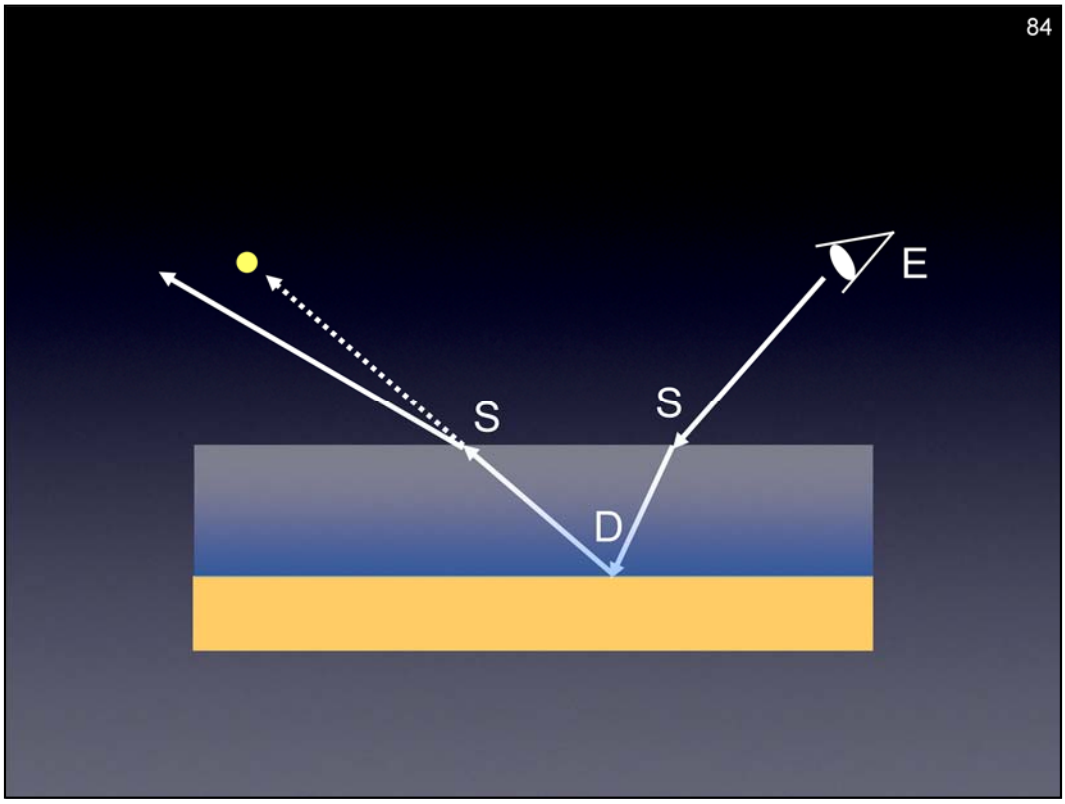
we shoot a ray from the eyes,



Let’s modify the scene slightly by adding a refractive object on top of the diffuse object. In order to compute the contribution from the light source to the eyes,

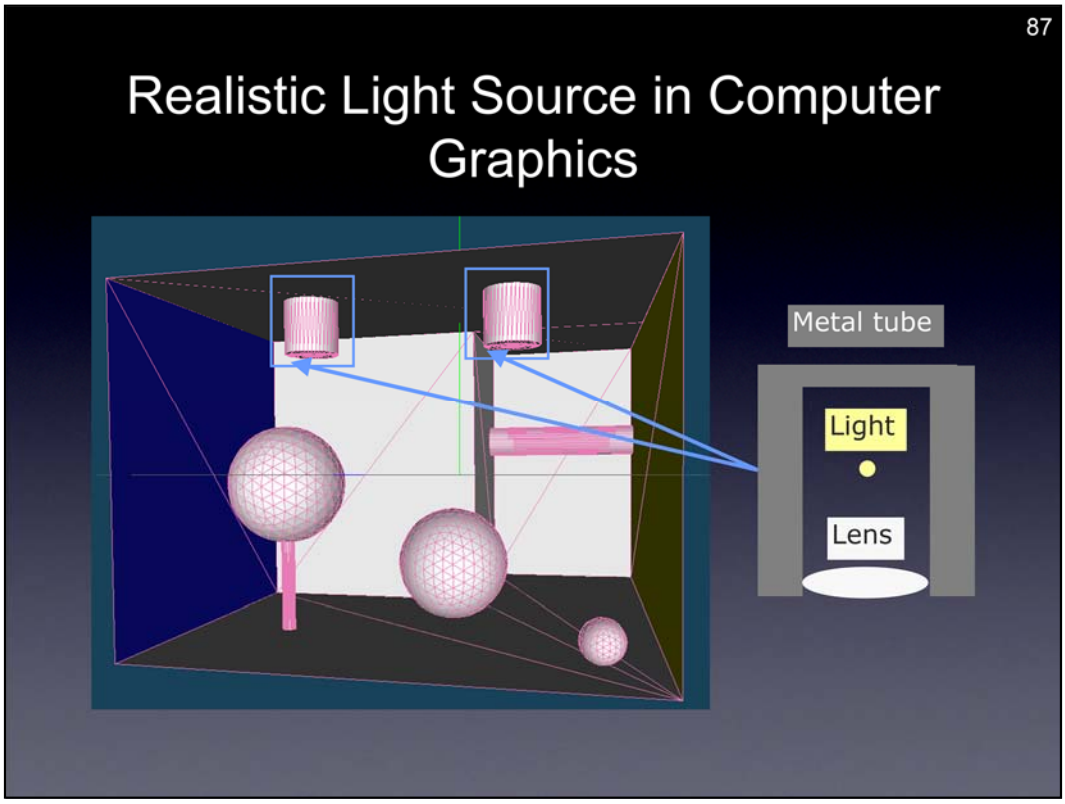


we first shoot a ray from the eyes as before. The only difference is that we need to take into account the specular refraction, which is denoted by S. So far, the computation is still as easy as before.

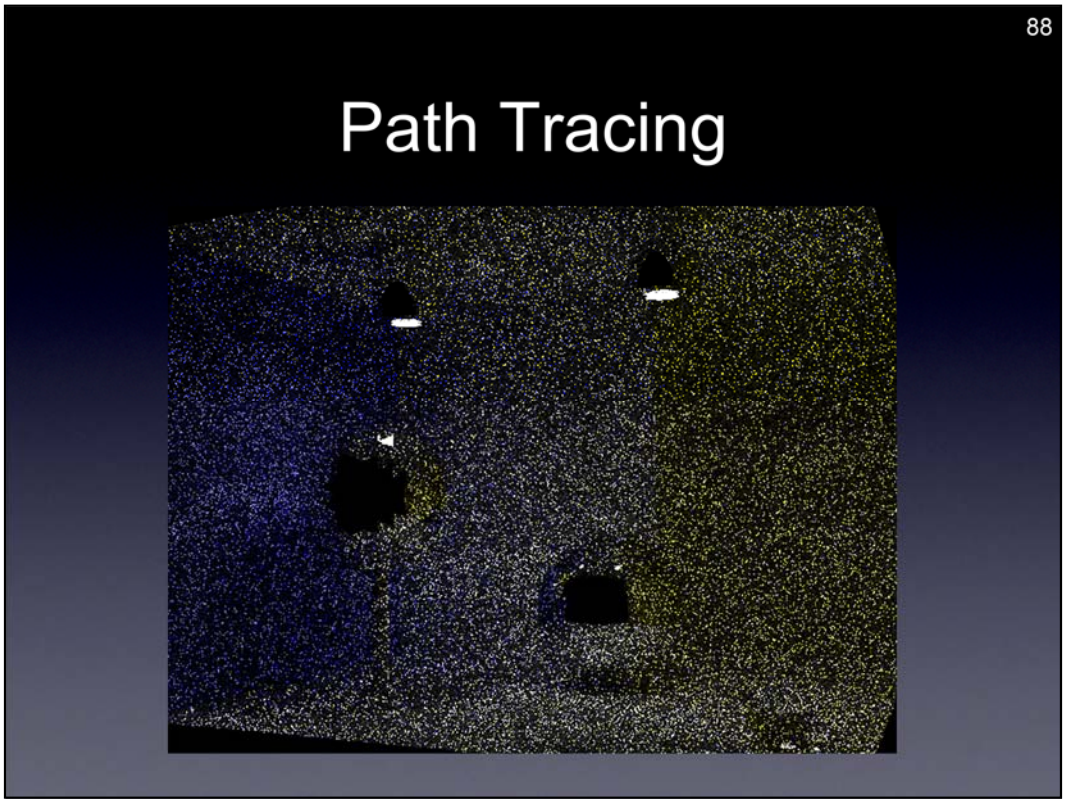


However, if we try to connect D with the light source by shooting a ray toward the light source, it could miss the light source because of the specular refraction.

Consistency of Photon Mapping

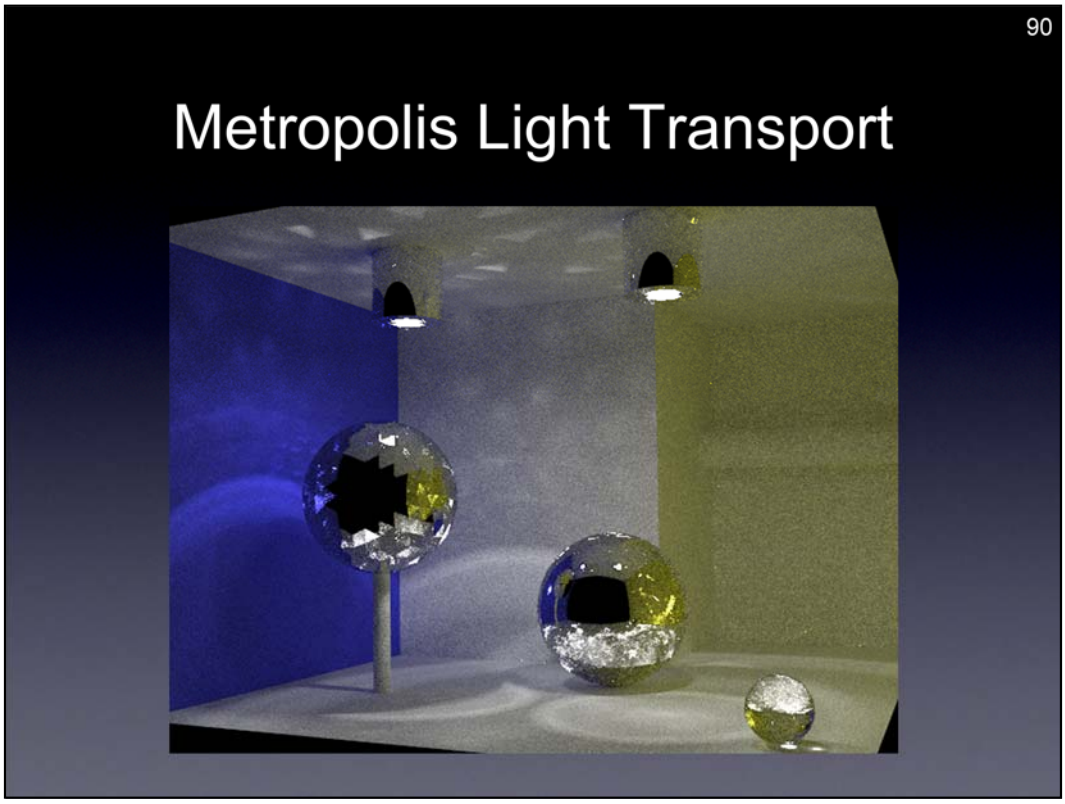


Let’s see if we can use this kind of realistic light sources in rendering. In this test scene, there is two small spherical light enclosed by a metal tube capped with lens which is similar to a real world light sourceI have shown before. Note that everything will be illuminated light coming through lens, which is caustic

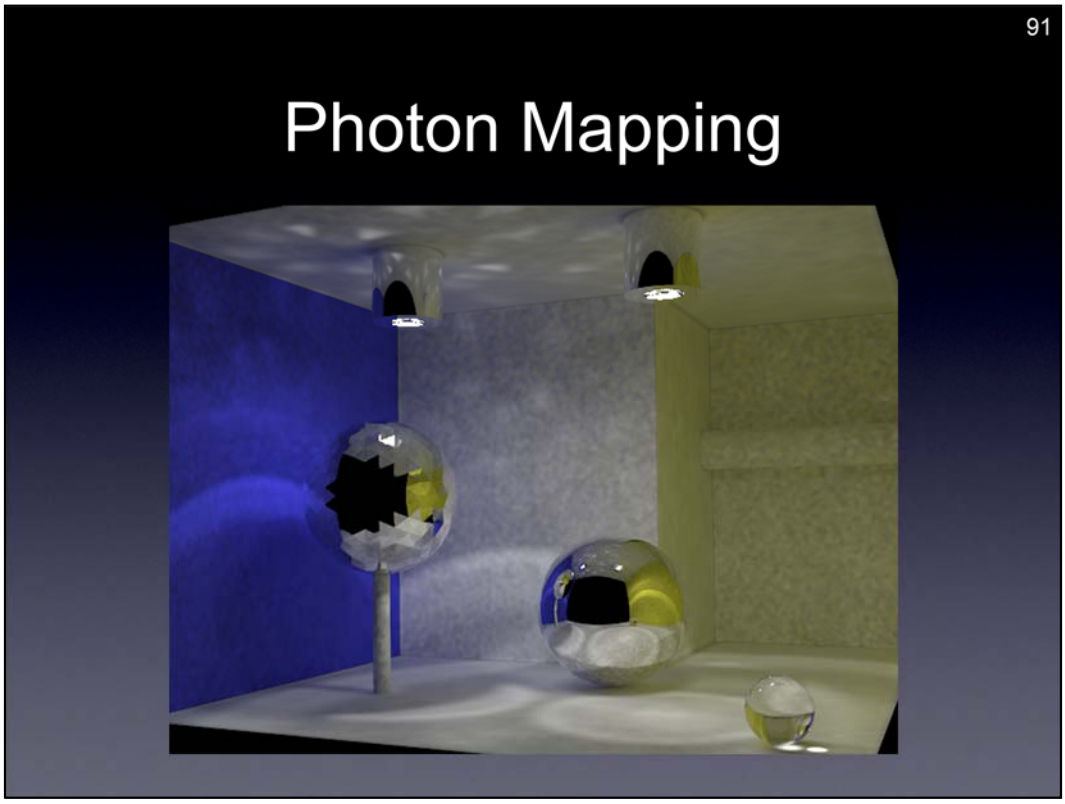


If you render this scene with path tracing, this is what you get. Image is very noisy because everything is illuminated by caustics and path tracing is not especially robust for rendering caustics.

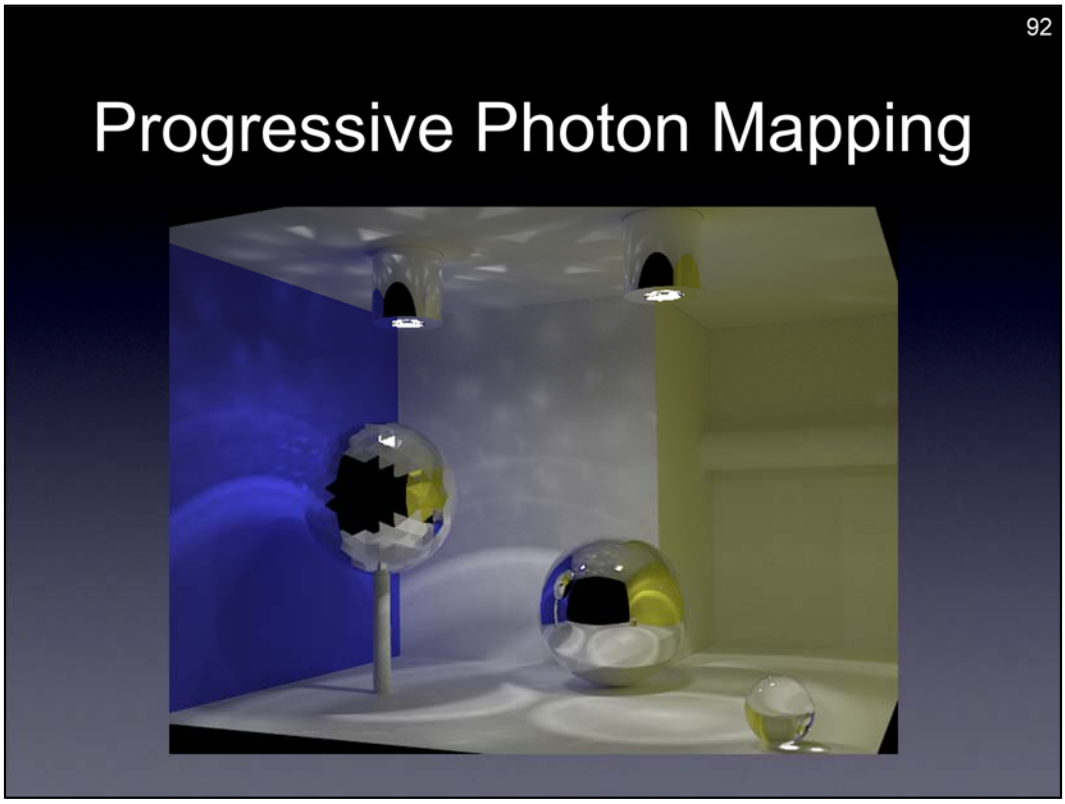
If you use bidirectional path tracing, it looks much better than path tracing. However, note that reflection on the mirror balls and refraction through the glass ball is very dark. This is because they are specular reflection or refraction of caustics, which is extremely difficult handle with any unbiased Monte Carlo ray tracing algorithm.



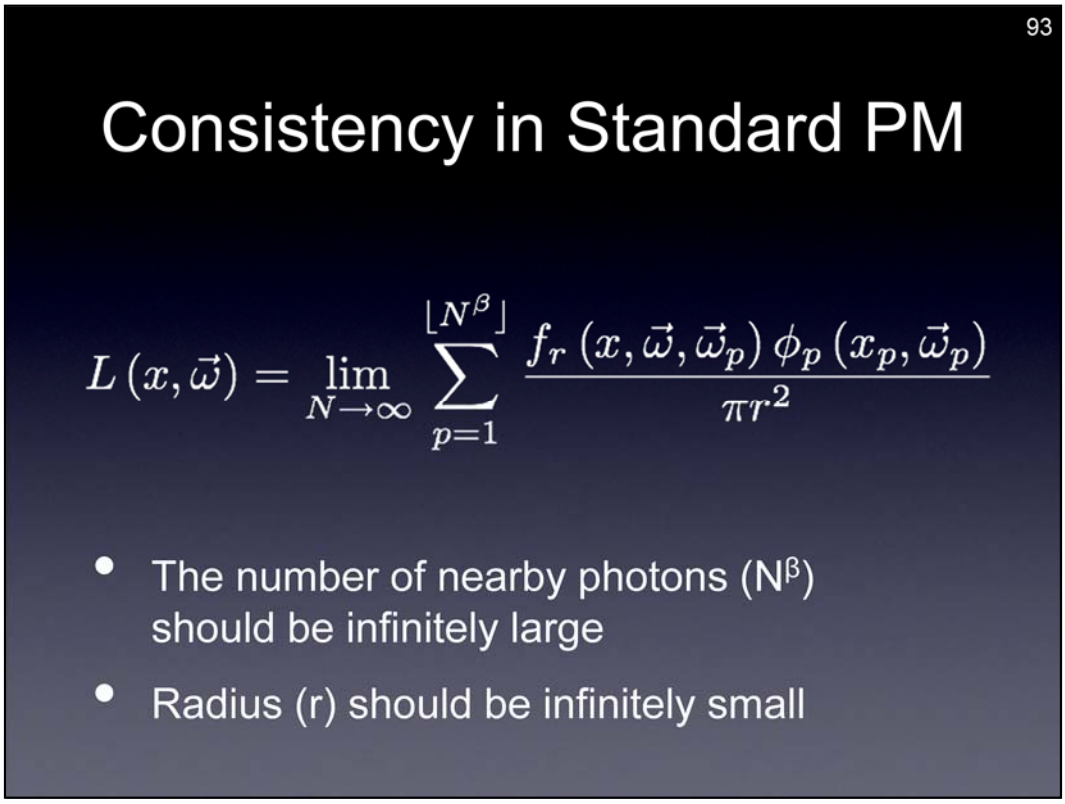
This is also the case even with Metropolis Light Transport which is considered to be the most robust algorithm to handle difficult lighting. Path in Metropolis Light Transport stacks reflection and refraction on the balls which caused bright spot noise.



If we allow some bias in rendering, Photon Mapping can robustly render this type of illumination. However, Photon Mapping runs out all memory before obtaining noise-free results because we need to store all the photons. Note that we have to directly visualize photon map in this scene because everything is caustic.



If you use our method, you can finally render this simple, yet difficult scene accurately and robustly. I will describe how this can be done in this talk.



Since flux and radius is available in every refinement, we can estimate radiance to show intermediate results to user. The equation to compute is exactly the same as standard photon mapping, except it uses accumulated, but unnormalized photon flux, which needs to be divided by the number of emitted photons so far over all refinement passes.

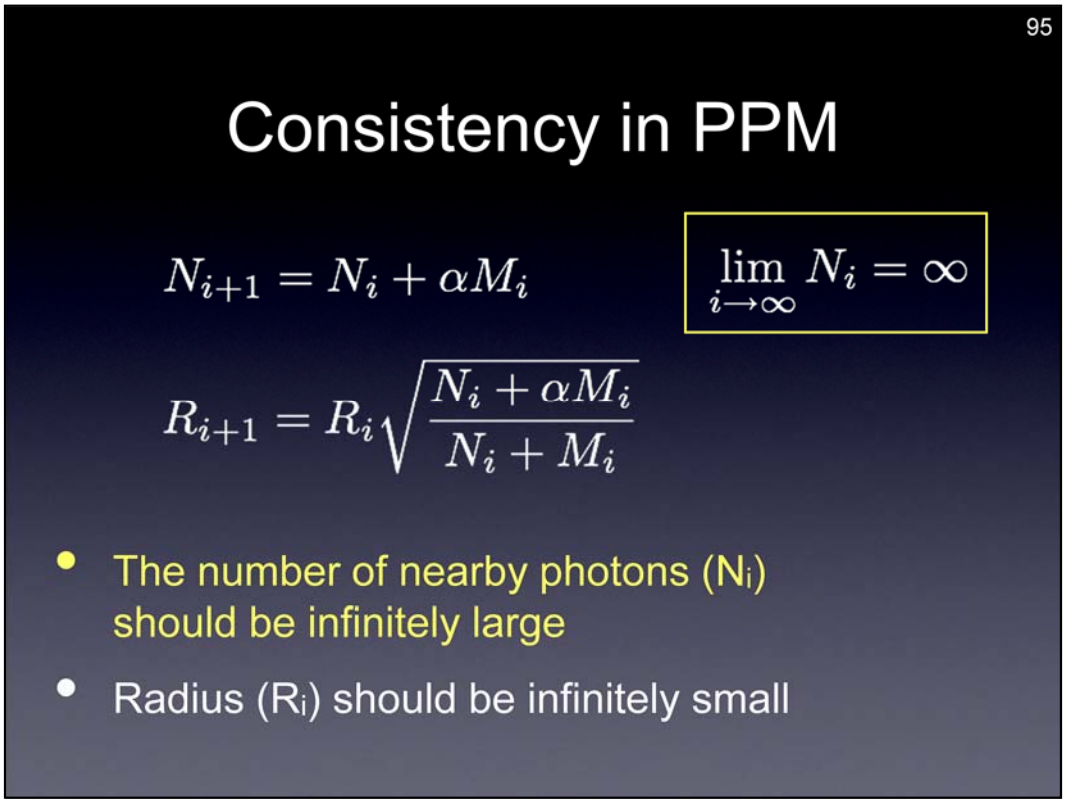
Consistency in PPM

$$N_{i+1} = N_i + \alpha M_i$$

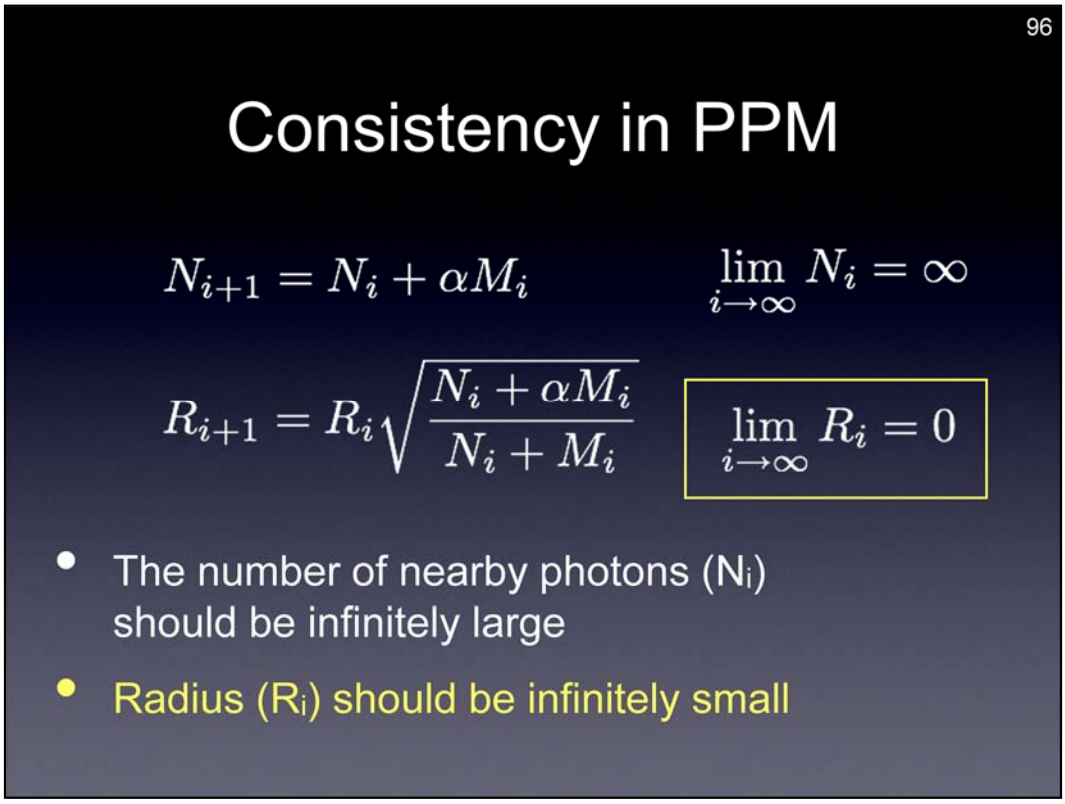
$$R_{i+1} = R_i \sqrt{\frac{N_i + \alpha M_i}{N_i + M_i}}$$

- The number of nearby photons (N_i) should be infinitely large
- Radius (R_i) should be infinitely small

I will not describe the derivation in this talk, but it turns out that we just need to use these relatively simple equation to update statistics on each measurement point. Alpha here is the parameter chosen by user. All we need to do is to update the number of photons, radius and flux using these equations in each iteration.



I will not describe the derivation in this talk, but it turns out that we just need to use these relatively simple equation to update statistics on each measurement point. Alpha here is the parameter chosen by user. All we need to do is to update the number of photons, radius and flux using these equations in each iteration.



I will not describe the derivation in this talk, but it turns out that we just need to use these relatively simple equation to update statistics on each measurement point. Alpha here is the parameter chosen by user. All we need to do is to update the number of photons, radius and flux using these equations in each iteration.