Benjamin Heasly <benjamin.heasly@gmail.com>

Pixel Sampling and Spectrum Coverage

1 message

Benjamin Heasly <benjamin.heasly@gmail.com> Fri, Jan 31, 2014 at 5:41 PM
To: David Brainard <brainard@psych.upenn.edu>

Hi David,

Here are some findings about Andy's code and some renderings I made today.

The Code

I've been looking at Andy's code to figure out how we can get intuitive behavior WRT ray samples per pixel and spectrum coverage.

I see in the code where Andy made changes, and how the changes lead to the behavior we're seeing, where the renderer "runs out of samples" before it covers the whole spectrum.

I think the fix is not quite straightforward because of the overall design of PBRT. Basically, the code that decides how many samples to generate per pixel (the "sampler") is distinct from the code that decides what to do with the samples (the "renderer"). In this case, Andy modified the renderer to assign a single wavelength to each sample, and potentially run out of samples before covering all wavelengths.

The sampler and renderer are not designed to communicate about how many samples are necessary (i.e 1 sample at a time as usual vs 31 samples at a time when the renderer wants do spectral lens simulation). Rather, the sampler is supposed to know all about samples, and the renderer is just supposed to accept as many samples as it's given.

I think PBRT is designed this way to allow for many different kinds of samplers. They're all supposed to interoperate with the renderer, without either one having to know how the other works, or how the other might have been modified by Andy.

So I don't see an immediately obvious way for Andy's modified renderer to request the extra samples it needs. To obey PBRT's design, the renderer should be able to request the extra samples from any type of sampler, but adding this behavior might mean modifying all the code for every kind of sampler.

I suspect Andy saw all this and decided to punt by increasing the number of samples from the scene file, instead fighting against PBRT's nice object oriented design.

I think what we want is to keep our modifications inside the the "renderer" code and not to modify any of the "sampler" code. I think this is possible, but I'm not sure yet how ugly it would be.

Some Renderings

In the mean time I made a series of renderings, as a test of our understanding of Andy's changes. I used the primary colors plus white, from the Color Checker chart. The first 4 renderings use the "new-PBRT" renderer that has Andy's recent changes. As I increased the samples per pixel from 8 to 256, the renderings got greener and redder. This is what I expect: in order to cover all 31 spectrum bands, we need at least 31 samples per pixel.

So with 32 samples per pixel, we get the full spectrum. But the image is grainy. It's roughly as though we were using the old-PBRT with 1 sample per pixel. With 256 samples per pixel, we're getting 8-fold coverage for each pixel and each spectrum band. This is roughly what we're used to with the old-PBRT at 8 samples per pixel.

For comparison, I rendered the same squares with the old-PBRT at 8 samples per pixel. This is what we usually

do for the SimpleSquare scene. I'd say it looks a bit nicer and cleaner than the rendering from the new-PBRT at 256 samples per pixel. I expected these two renderings to look almost identical and I'm not sure why they're different.

But overall, I think our understanding of Andy's code is good.

Have a great weekend!

Ben