



Benjamin Heasly &lt;benjamin.heasly@gmail.com&gt;

---

## Your pbirt on gitHub

10 messages

---

**David Brainard** <brainard@psych.upenn.edu>

Fri, Jan 10, 2014 at 9:37 AM

To: Andy Lai Lin &lt;ydna@stanford.edu&gt;

Cc: "benjamin.heasly@gmail.com Heasly" &lt;benjamin.heasly@gmail.com&gt;

Hi Andy,

Ben and I wanted to play around a bit with your pbirt, particularly to see if we can do some rendering that mimics the eye's depth of field. As I recall, you have this working in some version of pbirt-v2-spectral.

Question. There are several branches on your gitHub site. The "master" looks to be about 9 months old. The "spectral" branch is more recent, but differs in both directions from the master. Do you have a view of which is the best to use these days, particularly for the above purpose.

And, is there any documentation? The usersguide on gitHub is a placeholder.

Best,

David

=====

David H. Brainard  
RRL Professor of Psychology  
Director, Vision Research Center  
Director, Institute for Research in Cognitive Science  
University of Pennsylvania  
3401 Walnut Street, Suite 400A (mail address)  
3401 Walnut Street, Room 315C (office location)  
Philadelphia, PA, 19104  
Phone: [215-573-7579](tel:215-573-7579)  
Email: [brainard@psych.upenn.edu](mailto:brainard@psych.upenn.edu)  
Web: <http://color.psych.upenn.edu/brainard>

---

**Andy Lai Lin** <ydna@stanford.edu>

Fri, Jan 10, 2014 at 6:44 PM

To: David Brainard &lt;brainard@psych.upenn.edu&gt;

Cc: "benjamin.heasly@gmail.com Heasly" &lt;benjamin.heasly@gmail.com&gt;

Hi Professor Brainard and Ben,

It's good to hear from you. I apologize for the lack documentation on the github repository.

The spectral branch is the most updated branch, so please use that code. The reason why it shows that master is more updated is that I had to back track some unwanted changes at some point.

The DOF blur is made possible by using a modified camera module. I've attached 2 PBRT scenes for you guys, each using one of the two modified camera modules that we have created. Both of them are test scenes that

produce a psf from an input point and should run out of the box. The first scene uses an ideal thin lens camera model, and the second scene uses a realistic lens camera model (a lens with multiple lens elements).

The idealLens model should be self explanatory from the example .pbrt files. All units are in millimeters. filmdiag refers to the diagonal length of the sensor. The realisticDiffraction model is also mostly self explanatory. One thing to keep in mind is that I have enabled diffraction on both of these files. You may find it better to disable it since it requires far less samples per pixel for a good render. The realisticDiffraction model points to a lens file (in the example, 'dgauss.50mm.dat'). This format was originally proposed in the Stanford CS348 ray-tracing class, so please see the "Set up the camera" section of this website for a description of how this spec file works: <http://candela.stanford.edu/index.php/article/6>

Do note that PBRT does in fact come with a perspectiveCamera module that DOES allow for DOF blur. However, this DOF blur is not physically accurate, and also difficult to control, so we no longer use the perspectiveCamera in our simulations. The reason I mentioned this is that you may find this module to be sufficient.

Let me know if you have any questions,

-Andy

[Quoted text hidden]



**pointTest.zip**  
52K

---

**David Brainard** <brainard@psych.upenn.edu>  
To: Andy Lai Lin <ydna@stanford.edu>  
Cc: "benjamin.heasly@gmail.com Heasly" <benjamin.heasly@gmail.com>

Mon, Jan 13, 2014 at 8:50 AM

Thanks! We'll play with this and let you know if we have any problems.

Best,

David

[Quoted text hidden]

> <pointTest.zip>

[Quoted text hidden]

---

**Benjamin Heasly** <benjamin.heasly@gmail.com>  
To: David Brainard <brainard@psych.upenn.edu>  
Cc: Andy Lai Lin <ydna@stanford.edu>

Fri, Jan 24, 2014 at 10:09 AM

Hi Andy,

Thanks for the code and the examples. I got them running on my end without much trouble.

I noticed that the scaling of output data in the multispectral .dat files seems to have changed. The numbers seem to be smaller. Is this expected?

If so, is it just a change in scale factor?

Thanks.

I hope you're well.

Ben

[Quoted text hidden]

---

**Benjamin Heasly** <benjamin.heasly@gmail.com>  
To: David Brainard <brainard@psych.upenn.edu>  
Cc: Andy Lai Lin <ydna@stanford.edu>

Fri, Jan 24, 2014 at 11:24 AM

Hi again,

Actually, I think what i'm seeing is not a scaling change. It seems that the first 8 spectrum bands are being written to file, but the remaining 23 are being written out as zeros.

I can see this when I print each pixel to standard output, inside the method  
SpectralImageFilm::WriteImage(float splatScale)

I'm rendering one of the RenderToolbox3 test scenes (attached). It uses "image" film, which I think gets sent to the SpectralImageFilm class. It also uses the "perspective" camera.

Does this behavior ring any bells to you Andy? As I've described it so far, is there something I'm doing wrong?


I'll keep looking.

Thanks again.

Ben

[Quoted text hidden]

---

 **SimpleSquare-001.pbrt**  
2K

---

**Andy Lai Lin** <ydna@stanford.edu>  
To: Benjamin Heasly <benjamin.heasly@gmail.com>  
Cc: David Brainard <brainard@psych.upenn.edu>

Fri, Jan 24, 2014 at 2:44 PM

Hi Ben,

This is a known issue of Scene3D that we haven't currently fixed yet. Fortunately, there's a way around it if you specify the pbrt file in a certain way.

The problem is in the number of samples per pixel you are using. Currently, what happens is that it loops through one wavelength band at a time, for each sample per pixel. Thus, 8 samples per pixel isn't enough to cover all 31 bands, so you will only get the first 8. What I usually do is just use a very high number of samples per pixel. This way, all the spectral bands are covered, and the error from the process not covering all the wavelength bands evenly is very small. Unfortunately, pbrt will round the number of samples per pixel you are using to the nearest power of 2, so it's difficult to find a least common denominator to get it to work just right. If you want to be completely precise, you can keep track of how many times pbrt has rendered each particular wavelength, and apply a scaling factor to compensate for some bands being rendered with more samples per pixels than the others.

Also, the scaling that you are referring to will vary depending on the number of samples per pixel you specify. I believe that since the very first iteration, we did make some changes to infrastructure. Previously, one ray-tracing sample was assumed to be valid for all wavelengths. But the problem with this procedure, is that chromatic aberration due to index of refraction changes and diffraction is impossible to simulate this way. This is why we decided to ray-trace a ray representing one wavelength at a time. Although much slower, it allows us to simulate physically accurate artifacts from the imaging process.

Again, I apologize for this inconvenience, and it's definitely something we plan to sort out in the near future. Also, let me know if you would like code that reads our custom pbrt output files and converts them to ISET format - they are in our Scene3D git repository.

If you have any questions about this, let me know.

-Andy

----- Original Message -----

[Quoted text hidden]

---

**Benjamin Heasly** <benjamin.heasly@gmail.com>

Fri, Jan 24, 2014 at 5:00 PM

To: Andy Lai Lin <ydna@stanford.edu>

Cc: David Brainard <brainard@psych.upenn.edu>

Thanks Andy,

This is really helpful. I'll try increasing the samples and see what I come up with.

Ben

[Quoted text hidden]

---

**Benjamin Heasly** <benjamin.heasly@gmail.com>

Fri, Jan 31, 2014 at 3:43 PM

To: Andy Lai Lin <ydna@stanford.edu>

Cc: David Brainard <brainard@psych.upenn.edu>

Hi Andy,

I'm just getting back into this today.

Can I follow up on something you said, in a slightly unrelated way?

You said that we can apply a scaling factor to compensate for some bands being rendered with more samples per pixels than the others. I think this consistent with what we've been seeing, where PBRT's output values are proportional to the number of samples per pixel.

Do you know off hand: if we use 1 sample per pixel, and we specify our scene using standard units like meters and watts, does PBRT end up outputting some intuitive units, like radiance? I'm not sure if the answer would be a simple yes or now, but I've been curious about that part of PBRT.

Thanks again.

Best,

Ben

[Quoted text hidden]

---

**Andy Lai Lin** <ydna@stanford.edu>

Fri, Jan 31, 2014 at 5:27 PM

To: Benjamin Heasly <benjamin.heasly@gmail.com>

Cc: David Brainard <brainard@psych.upenn.edu>

Hi Ben,

Unfortunately we've never run these experiments to figure this stuff out for sure. But I am guessing that there is some sort of correspondence, if you keep the number of samples per pixel constant. Also - in the latest release, 1 sample per pixel won't work anymore, since it will only fill one wavelength bin.

-Andy

----- Original Message -----

From: "Benjamin Heasly" <benjamin.heasly@gmail.com>

To: "Andy Lai Lin" <ydna@stanford.edu>

[Quoted text hidden]

---

**Benjamin Heasly** <benjamin.heasly@gmail.com>

Fri, Jan 31, 2014 at 5:34 PM

To: Andy Lai Lin <ydna@stanford.edu>

Cc: David Brainard <brainard@psych.upenn.edu>

OK, thanks. We did some tests to try to figure this out, and we found a correspondence with radiance, to within a scale factor. I don't understand what the scale factor is, but at least we can measure it experimentally.

Ben

[Quoted text hidden]