pbrt v3 – Multiple Importance Sampling

Luís Paulo Santos – Dep. de Informática

Universidade do Minho

May, 2022

1- Setting up pbrt for a new Integrator

For this tutorial you will add a new `Integrator` to your `pbrt` project. This is identical to what has been done in Tutorial 2 to add the `ViTutorial2` integrator.

Download the file "`veach-mis.zip`" from the elearning platform and unzip it to your `pbrt-v3/src/integrators` folder. There should now be two new files: `veach-mis.cpp` and `veach-mis.h`.

Depending on the operating system and project management system you are using (`Windows`, `MacOS`, `Linux`, `make`, `CMake`, `MSVS`, `xcode`, etc.) you have to add these 2 files to your project, under the `integrators` tab.

Opening either `veach-mis.cpp` or `veach-mis.h`, you'll verify that it defines a new class `MISIntegrator`, which is the integrator we will be working on.

The new integrator has to be included in the `pbrt` library and a relationship has to be established between the integrator name used in the scenes' descriptions (`veachMIS`) and the new class `MISIntegrator`. This is done in the file `api.cpp` belonging to pbrt's core (`pbrt-v3/src/core`). Open this file and make the two modifications indicated below:

1. where it reads:

```
#include "integrators/whitted.h"
```

   change it to:

```
#include "integrators/whitted.h"
#include "integrators/veach-mis.h"
```

2. where it reads:

```
    if (IntegratorName == "whitted")
        integrator    =    CreateWhittedIntegrator(IntegratorParams,
sampler, camera);
```

   change it to:

```
    if (IntegratorName == "whitted")
        integrator    =    CreateWhittedIntegrator(IntegratorParams,
sampler, camera);
    else if (IntegratorName == "veachMIS")
        integrator = CreateMISIntegrator(IntegratorParams, sampler,
camera);
```

Rebuild `pbrt`. Try it by rendering the `veach_mis.pbrt` scene that you can download from the elearning platform.

The Scene Setup and Rendering Algorithm

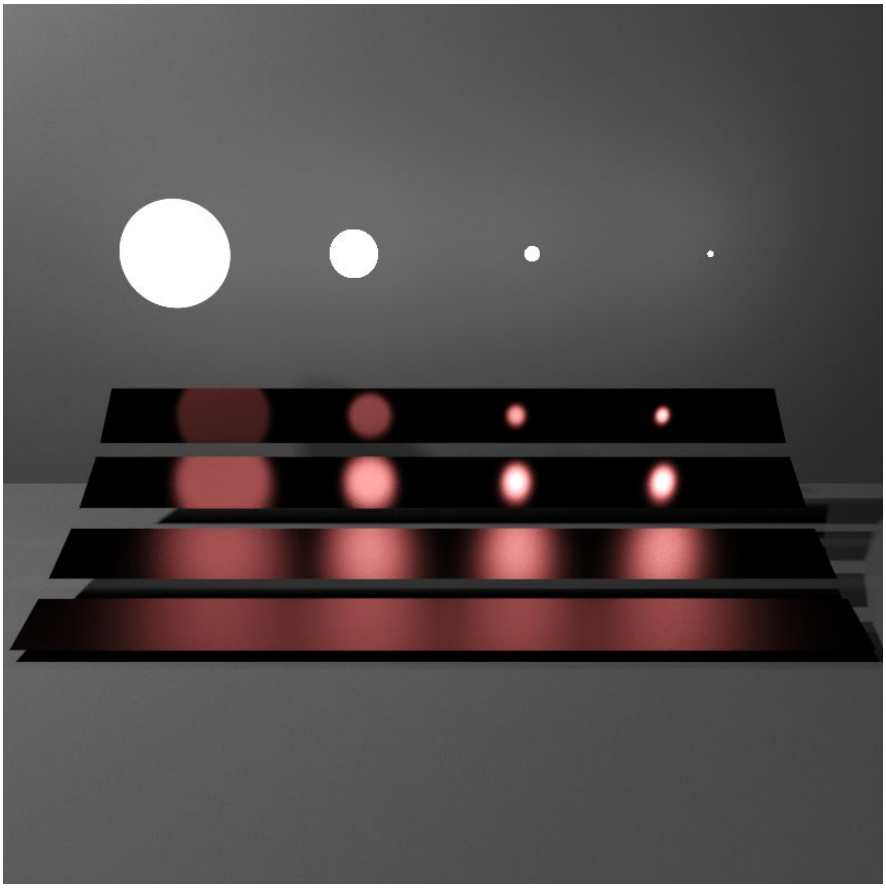

Figura 1 - Veach scene rendered using one-sample MIS and 4096 spp

The image above is a reference image for the scene we will be using. You can see it is constituted by 4 plates with different glossiness. The farthest away plate is purely specular (the BRDF is a Dirac delta: to any view direction corresponds a single incident direction), while the roughness of the surfaces increases for the other plates (the BRDF lobe for a given view direction becomes larger). You can also see that there are four different sizes light sources. This scene is directly inspired on the famous scene appearing on Eric Veach's Ph.D. dissertation, Stanford University, December 1997, entitled "Robust Monte Carlo Methods for Light Transport Simulation".

The rendering algorithm used is extremely simple: only paths of length 2 are shot. The primary ray is shot from the eye, through the image plane into the scene. Then a single secondary ray is shot (see figure 2). The question is: how to select the direction for this secondary ray?

Remember that reflected radiance at a point *p* is given by the integral over the hemisphere Ω of an integrand that is the product of three terms: the BRDF, the incident radiance and the cosine of the incident direction and the surface normal at *p* – see equation

$$L_r(p \rightarrow \omega_r) = \int_\Omega f_r(p, \omega_r \leftrightarrow \omega_i) L_i(p \leftarrow \omega_i) \cos(\vec{N}_x, \omega_i) d_{\omega_i}$$

Equação 1- Reflected radiance integral

One of the main issues when evaluating this integral using Monte Carlo techniques is which directions ω_i to select for sampling.

Importance sampling tells us that directions should be selected according to how important they are for the value of the integral, i.e., we should select directions that maximize the value of the integrand and these directions must be selected according to a given probability distribution function (pdf).

The problem is that the integrand is the product of three terms and it is often impossible to build a pdf that follows the overall shape of the integrand (one of the main reasons is that in the general case the incident radiance L_i is unknown).

Consider Figure 2 below. If you are given the chance to STOCHASTICALLY select a single secondary ray, would you select it according to the light source pdf (solid angle represented in yellow) or to the BRDF lobe (solid angle in green)?

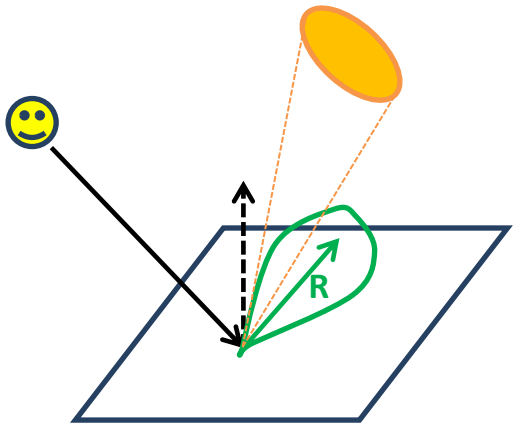

Figure 2- our algorithm setup. Should we select the sampling direction according to a pdf based on the lobe of the BRDF of according to the solid angle of the light source?

2 - Sampling according to the light source

The `veachMIS` integrator allows us to select, for this very simple length two path tracing algorithm, which pdf to use to select the secondary ray.
Change your `veach_mis.pbrt` scene such that the `Sampler`, `Film` and `Integrator` are:

```
Sampler "stratified" "integer xsamples" [8] "integer ysamples"
[8] "bool jitter" ["true"]
Integrator "veachMIS" "string MISstrategy" ["light"]
Film "image" "string filename" ["veachMIS_L.exr"] "integer
xresolution" [800] "integer yresolution" [800]
```

Please comment the results obtained, keeping in mind that the secondary ray direction is stochastically selected according to the solid angle of the light source and completely ignores the BRDF shape.

FILL IN TABLE 1 AT THE END the execution time in seconds. Also fill the rightmost column, by making a cross on the squares corresponding to the best rendered reflections (you can use more than one cross).

For which conditions (light source size versus BRDF lobe width) did this sampling process performed better? And worse? Why?

Open the file `veach-mis.cpp` and carefully study the code associated with

```
if (MISStrategy == SAMPLE_L ) { // sample light sources
…
}
```

making sure you understand every step of this very simple importance sample integrator.

3 - Sampling according to the BSDF

Change your `veach_mis.pbrt` scene such that the `Film` and `Integrator` are:

```
Integrator "veachMIS" "string MISstrategy" ["bsdf"]
Film "image" "string filename" ["veachMIS_BSDF.exr"] "integer xresolution" [800] "integer yresolution" [800]
```

Comment the results obtained, keeping in mind that the secondary ray direction is now stochastically selected according to the lobe of the BRDF and ignores the light source.

FILL IN TABLE 1 AT THE END the execution time in seconds. Also fill the rightmost column, by making a cross on the squares corresponding to the best rendered reflections (you can use more than one cross).

For which conditions (light source size versus BRDF lobe) did this sampling process performed better? And worse? Why? How does this result compare to the one obtained with light sampling?

Open the file `veach-mis.cpp` and carefully study the code associated with

```
else if (MISStrategy ==SAMPLE_BSDF) {  // sample BSDF
…
}
```

making sure you understand every step of this very simple importance sample integrator.

4 - Multiple importance sampling: multi-sample model

Your task now is to develop the code for the strategy of sampling according to the two terms of the integrand – BSDF and light source – using multiple importance sampling.

You have to uncomment and complete the code in

```
else if (sample_method==SAMPLE_MIS_ALL) {  // sample light and BSDF using MIS ALL

…

}
```

Remember that this is just a matter of shooting two samples instead of one. One sample will be shot according to the light source (just reuse the code from that section) and another one will be shot according to the BSDF (once again reuse the code).

But now each of these samples has to be weighted against the probability that could have been generated using a different pdf.

So for the light source sample, after you generate the direction `wi` using the appropriate probability density and getting a given `lightPdf`:

```
Spectrum Li = light->Sample_L(p, isectp->rayEpsilon, lightSample, ray.time,
                                                    &wi, &lightPdf, &visibility);
```

you have to evaluate what would be the probability density of this same direction `wi` if it had been generated from the BSDF lobe:

```
bsdfPdf = bsdf->Pdf(wo, wi, BSDF_ALL);
```

And then using the balance heuristic you compute the MIS weight:

```
        // compute the weight for MIS
        // this is the balance heuristic with one sample per distribution
        weight = lightPdf / (lightPdf + bsdfPdf);
```

When evaluating the light source contribution the weight has to be factored in:

```
        L += f * Li * weight * (AbsDot(wi, n) / lightPdf) * nLights;
```

Then you do the same for the BSDF Sample:

```
        Spectrum f = bsdf->Sample_f(wo, &wi, outgoingBSDFSample, &bsdfPdf,
                                    BSDF_ALL, &flags);
        lightPdf = light->Pdf(p, wi);

        weight = bsdfPdf / (lightPdf + bsdfPdf);
```

and use this weight to evaluate this sample contribution:

```
      pathThroughput *= f * AbsDot(wi, n) * weight * nLights / bsdfPdf;
```

Ok, go ahead and use this new MIS integrator, by changing your scene header to

```
Sampler "stratified" "integer xsamples" [6] "integer ysamples"
[6] "bool jitter" ["true"]
Film "image" "string filename" ["veach_MIS_ALLx36.exr"] "integer
xresolution" [600] "integer yresolution" [600]
```

```
SurfaceIntegrator "veachMIS" "string sample_method" ["mis_all"]
```

Note that the number of samples per pixel has been reduced to 6x6=36, which is approximately half of 64, because now we shoot two secondary rays per primary ray.

FILL IN TABLE 1 AT THE END the execution time in seconds.

Compare this image with the previous ones.

5 - Multiple importance sampling: one-sample model

The one-sample model allows us to stochastically select one of the pdf's according to some probability and then collect only one sample from that distribution, obviously still weighted using the balance heuristic.

You have to complete the code in

```
else if (sample_method== MISStrategy) {  // sample light or BSDF

…

}
```

Remember that you'll have to draw a random number and then according to it select either light or BSDF sampling:

```
// control the selection rate of each of the distributions (light vs BRDF)
const float sample_L_pdf = 0.5f;
// select which distribution to sam,ple from
const int whichFactor =(sampler.Get1D() <= sample_L_pdf ? SAMPLE_L:SAMPLE_BSDF);
if (whichFactor == SAMPLE_L) {  // do light sources
      …
} else if (whichFactor == SAMPLE_BSDF) { // do BSDF
      …
}
```

So for sampling according to the light sources, after you generate the direction `wi` using the appropriate probability density and getting a given `lightPdf`:

```
// get the light sampling direction and respective pdf
Spectrum Li = light->Sample_Li(isect, uLight, &wi, &lightPdf, &visibility);
```

you have to evaluate what would be the probability density of this same direction `wi` if it had been generated from the BSDF lobe:

```
const float BSDFpdf = isect.bsdf->Pdf(isect.wo, wi, BSDF_ALL);
```

And then using the balance heuristic you compute the MIS weight:

```
// compute the weight for MIS
// this is the balance heuristic with one sample per distribution
float MISweight = lightPdf * whichLightPdf / (lightPdf * whichLightPdf +
bsdfPdf);
```

Also keep in mind that you selected `SAMPLE_L` stochastically with probability `sample_L_pdf`, therefore you have to divide `MISweight` by this:

```
MISweight /= sample_L_pdf;
```

For sampling according to BSDF, after you generate the direction `wi` using the appropriate probability density and getting a given `BSDFpdf`:

```
// get the BSDF sampling direction and respective pdf
Spectrum f = isect.bsdf->Sample_f(wo, &wi, sampler.Get2D(), &BSDFpdf, BSDF_ALL,
&flags);
```

You have to evaluate what would be the probability density of this same direction `wi` if it had been generated from the light sources:

```
const float lightPdf += light->Pdf_Li(isect, wi);
```

And then using the balance heuristic you compute the MIS weight:

```
// compute the weight for MIS
// this is the balance heuristic with one sample per distribution
float MISweight = bsdfPdf / (lightPdf * whichLightPdf + bsdfPdf);
```

Also keep in mind that you selected `SAMPLE_BSDF` stochastically with probability `1.f - sample_L_pdf`, therefore you have to divide `MISweight` by this:

```
MISweight /= (1.f - sample_L_pdf);
```

Run the new integrator using the headers and compare with the previous images.

```
Integrator "veachMIS" "string MISstrategy" ["mis"]
Film "image" "string filename" ["veachMIS_MIS.exr"] "integer
xresolution" [800] "integer yresolution" [800]
```

FILL IN TABLE 1 AT THE END the execution time in seconds.

Tabela 1 - Time for the various images

File	Time (secs)	Quality			
				BSDF	
				Broad	Sharp
veachMIS_L		L	Small		
			Large		
				BSDF	
				Broad	Sharp
veachMIS_BSDF		L	Small		
			Large		

veachMIS_MIS				BSDF	
				Broad	Sharp
		L	Small		
			Large		