(https://schuttejoe.github.io/img/Posts/GgxImportanceSamplingImages/GgxDPreview.png)

Importance Sampling techniques for GGX with Smith Masking-Shadowing: Part 1

March 7, 2018

Today I'll be writing about importance sampling the GGX BRDF. GGX used with the Smith masking-shadowing function has become ubiquious in the game industry so it was the first specular model I wanted to implement in the path tracer I'm writing. About two weeks before I wrote this blog post I sat down and googled "importance sampling ggx" and while I was able to find a number of articles and papers that described specific details of the technique I was left a little unsure about whether I was combining the pieces together correctly. This blog post is a result of my investigation that followed and will hopefully provide enough details along the way that anyone in the future who searches for a similar phrase will be confident in their implementation.

Additionally, Eric Heitz and Eugene D'Eon have published some wonderful research in the last few years that made improvements on the technique which I was not able to find any mention of in other blogs so hopefully by writing about that in Part 2 (/post/ggximportancesamplingpart2) of this post I am able to increase the visiblity of their improvements.

Ok let's get started by describing the problem we want to solve. I'm pretty sure there's a rule that you need to include the rendering equation in your first blog post. Or maybe it was something you did for good luck? Either way, I'm going to lead with that and try to make use of it.

$$L_o(X, \hat{\omega}_o) = L_e(X, \hat{\omega}_o) + \int_\Omega L_i(X, \hat{\omega}_i)\ f(\hat{\omega}_i, \hat{\omega}_o)\ |\hat{\omega}_i \cdot \hat{\omega}_g|\ d\hat{\omega}_i$$

When evaluating a single ray hit in a path tracer we want to zoom in on the interior of that integral. Furthermore, we know that $L_i(X, \hat{\omega}_i)$ will be information gathered by casting a ray in direction $\hat{\omega}_i$ so we just want to focus here on choosing a direction $\hat{\omega}_i$ and then solving for $f(\hat{\omega}_i, \hat{\omega}_o)\ |\hat{\omega}_i \cdot \hat{\omega}_g|$. The equation $f(\hat{\omega}_i, \hat{\omega}_o)$ here is the Cook-Torrance microfacet BRDF described by:

$$f(\hat{\omega}_i, \hat{\omega}_o) = \frac{F(\hat{\omega}_i, \hat{\omega}_m)\ G_2(\hat{\omega}_i, \hat{\omega}_o, \hat{\omega}_m)\ D(\hat{\omega}_m)}{4\ |\hat{\omega}_i \cdot \hat{\omega}_g|\ |\hat{\omega}_o \cdot \hat{\omega}_g|}$$

In our specific example we are looking at the GGX distribution of normals for $D(\hat{\omega}_m)$ and the Smith masking-shadowing function for $G_2(\hat{\omega}_i, \hat{\omega}_o, \hat{\omega}_m)$. These equations are written as:

$$D(\hat{\omega}_m) = \frac{\alpha^2}{\pi((\hat{\omega}_g \cdot \hat{\omega}_m)^2(\alpha^2 - 1) + 1)^2}$$

$$G_2(\hat{\omega}_i, \hat{\omega}_o) = \frac{2(\hat{\omega}_g \cdot \hat{\omega}_i)(\hat{\omega}_g \cdot \hat{\omega}_o)}{(\hat{\omega}_g \cdot \hat{\omega}_o)\sqrt{\alpha^2 + (1-\alpha^2)(\hat{\omega}_g \cdot \hat{\omega}_i)^2} + \hat{\omega}_g \cdot \hat{\omega}_i\sqrt{\alpha^2 + (1-\alpha^2)(\hat{\omega}_g \cdot \hat{\omega}_o)^2}}$$

$D(\hat{\omega}_m)$ is defined in Microfacet Models for Refraction through Rough Surfaces (https://www.cs.cornell.edu/~srm/publications/EGSR07-btdf.pdf) and the uncorrelated version of $G_2(\hat{\omega}_i, \hat{\omega}_o)$ that we use can be found in PBR Diffuse Lighting for GGX+Smith Microsurfaces (https://twvideo01.ubm-us.net/o1/vault/gdc2017/Presentations/Hammon_Earl_PBR_Diffuse_Lighting.pdf).

Now back to describing the problem we are solving. We know that we want to find $\hat{\omega}_i$ and calculate $f(\hat{\omega}_i, \hat{\omega}_o)\ |\hat{\omega}_i \cdot \hat{\omega}_g|$. As inputs we have a surface roughess α, an outgoing direction $\hat{\omega}_o$, and a geometric normal $\hat{\omega}_g$ from our surface. Both of those vectors will have been transformed to tangent space so $\hat{\omega}_g$ will actually be just our up direction. In my case, that is the Y axis. We have everything we need so let's get started.

The first technique I want to write about is to importance sample only $D(\hat{\omega}_m)$. This is quite effective since the distribution of normals does have a significant impact on shape of the entire BRDF. To importance sample $D(\hat{\omega}_m)$ we will use the inverse of the CDF of $D(\hat{\omega}_m)$ to generate a microfacet normal $\hat{\omega}_m$. If you are familar with using GGX in game rendering this can be thought of as creating a half vector in tangent space. The derivation of the inverse of the CDF is covered in detail in Cao Jaiyin's post here (https://agraphicsguy.wordpress.com/2015/11/01/sampling-microfacet-brdf/) so I'll just refer you to that for the details and list the relevant equations here:

$$\theta_m = \arccos\sqrt{\frac{1-\xi_0}{\xi(\alpha^2-1)+1}} \qquad \phi = 2\pi\xi_1$$

Now, if you remember how importance sampling works, you'll know that we are eventually going to need to divide our result by the pdf of the sample we take. As we will see the convienance of later, the pdf for generating the spherical coordinates is very similar to the distribution of normals itself. From Cao's post we have:

$$p(\theta_m, \phi) = \frac{\alpha^2 \cos\theta_m \sin\theta_m}{\pi((\alpha^2-1)\cos^2\theta_m + 1)^2}$$

The next step we'll want to take is to transform these spherical coordinates into Cartesian coordinates. I think if you are familar with everything I've said so far you probably already know the equations for this but I'll list them for the sake of completeness. With one last reminder that I am using Y-up, here they are:

$$x = r\sin\theta\cos\phi$$

$$y = r\cos\theta$$

$$z = r\sin\theta\sin\phi$$

where r = 1 in this case. This gives us $\hat{\omega}_m$.

Ok, let's step aside for a moment and talk about what the transformation from spherical to Cartesian coordinates does to our pdf. Assuming we drew our sample x from a random variable X and then we apply some transformation T to that sample it can be thought of as us having drawn a sample from a different random variable Y with the property that

$$p_y(y) = p_y(T(x)) = \frac{p_x(x)}{|J_T(x)|}$$

where $|J_T(x)|$ is the absolute value of the determinant of the Jacobin of transformation T. That's a lot of words to describe one thing O_O. If you want even more words and to see more details of the derivation you can find that in section 3 of Notes on the Ward BRDF (https://www.graphics.cornell.edu/~bjw/wardnotes.pdf).

Armed now with that knowledge and given that the Jacobian of the transformation from spherical to Cartesian is given as:

$$|J_T| = r^2\sin\theta$$

we can divide that from the pdf $p_m(\theta_m, \phi)$ with r still equal to 1 to see that we've canceled out the $sin\theta$ term. We can also now substitude in $D(\hat{\omega}_m)$ and write $\cos\theta$ as $(\hat{\omega}_m \cdot \hat{\omega}_g)$ to get:

$$p_m(\hat{\omega}_m) = D(\hat{\omega}_m)(\hat{\omega}_m \cdot \hat{\omega}_g)$$

We're almost there. Now that we have our microfacet normal $\hat{\omega}_m$ we can use that to calculate the light direction $\hat{\omega}_i$ that we want to sample by reflecting our view vector $\hat{\omega}_o$ by $\hat{\omega}_m$. This is the final step for generating $\hat{\omega}_i$ but note that we'll be doing another transformation when we reflect the vector so we're going to need to divide the pdf by the determinant of that Jacobian as well. The equation for reflection is given by:

$$\hat{\omega}_i = 2|\hat{\omega}_o \cdot \hat{\omega}_m|\hat{\omega}_m - \hat{\omega}_o$$

and the determinent of the Jacobian of that transformation is given by:

$$|J_R| = 4|\hat{\omega}_o \cdot \hat{\omega}_m|$$

Time for another aside. Many of you might have raised some question marks with that last equation. A number of papers will state that the determinent of the Jacobian of the reflection is $|J_R| = \frac{1}{4|\hat{\omega}_o \cdot \hat{\omega}_m|}$. After some confusion followed by some more searching I'm going to give you an intuitive derivation for how I arrived at what I did. From Notes on the Ward BRDF (https://www.graphics.cornell.edu/~bjw/wardnotes.pdf) we get this diagram:

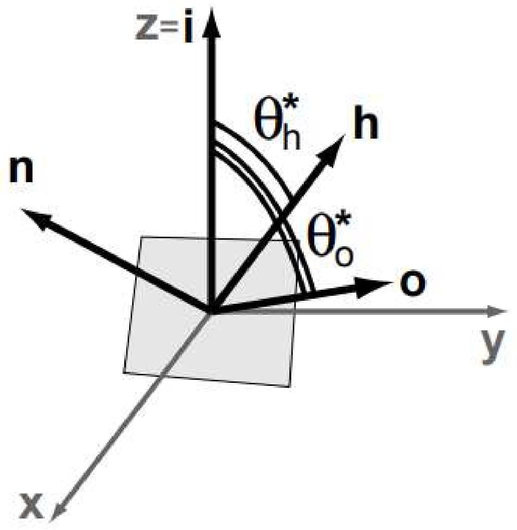


Figure 4: A coordinate system where the incident vector **i** is the z-axis, and the surface normal **n** lies in the x-z plane (i.e. $\mathbf{n} \cdot \mathbf{y} = 0$). As in Figure 1, we can specify directions using two spherical angles, $\theta^\star$ and $\phi^\star$. The star indicates that they are relative to this coordinate frame. The relationship between **h** and **o** is, $\theta_o^\star = 2\theta_h^\star$ and $\phi_o^\star = \phi_h^\star$.

Using Wards notation for a generic set of vectors i, h, and n what we want to do is:

- Rotate the vectors to have the orientation described in Ward's image. This has a Jacobian of 1 since rotation is a linear transform.
- Transform the vector into spherical coordinates. This has a Jacobian of $\frac{1}{\sin\theta_h}$.
- Apply the transform $f(\theta_h, \phi_h) = (2\theta_h, \phi_h) = (\theta_o, \phi_h)$. This has a Jacobian of 2.
- Transform the new coordinates back to Cartesian coordinates. This has a Jacobian of $\sin\theta_o$.
- Perform the inverse rotation from our first step. This has a Jacobian of 1 again.

Now we have calculated the reflected vector o in a far more complicated way than we described earlier but this now has much easier steps for deriving the Jacobian. Since the Jacobian of the composition of two transformations is just the product of those two Jacobians we just need to multiply the above Jacobians together. This gives us:

$$|\frac{2\sin\theta_o}{\sin\theta_h}| = |\frac{2\sin(2\theta_h)}{\sin\theta_h}| = |\frac{4\sin\theta_h\cos\theta_h}{\sin\theta_h}| = 4|cos\theta_h|$$

Ah, well, there it is.

All right, back on topic. When we now take that Jacobian and divide that out from $p_m(\hat{\omega}_m)$ we get the final pdf for sampling $\hat{\omega}_i$. I'm going to play a little fast and loose with notation here and call it $p_i(\hat{\omega}_m, \hat{\omega}_o)$

$$p_i(\hat{\omega}_m, \hat{\omega}_o) = \frac{D(\hat{\omega}_m)(\hat{\omega}_m \cdot \hat{\omega}_g)}{4|\hat{\omega}_o \cdot \hat{\omega}_m|}$$

Now that we have direction $\hat{\omega}_i$ and the pdf of the random variable that generated $\hat{\omega}_i$ we have everything we need to solve for the reflectance. We now take the $f(\hat{\omega}_i, \hat{\omega}_o)\ |\hat{\omega}_i \cdot \hat{\omega}_g|$ term we mentioned at the beginning and divide that by the pdf $p_i(\hat{\omega}_m, \hat{\omega}_o)$ to solve for the reflectance coming from direction $\hat{\omega}_i$.

$$\frac{f(\hat{\omega}_i, \hat{\omega}_o)|\hat{\omega}_i \cdot \hat{\omega}_g|}{p_i(\hat{\omega}_m, \hat{\omega}_o)} = \frac{\frac{F(\hat{\omega}_i,\hat{\omega}_m)\ G_2(\hat{\omega}_i,\hat{\omega}_o,\hat{\omega}_m)\ D(\hat{\omega}_m)|\hat{\omega}_i\cdot\hat{\omega}_g|}{4\ |\hat{\omega}_i\cdot\hat{\omega}_g|\ |\hat{\omega}_o\cdot\hat{\omega}_g|}}{\frac{D(\hat{\omega}_m)(\hat{\omega}_m\cdot\hat{\omega}_g)}{4|\hat{\omega}_o\cdot\hat{\omega}_m|}}$$

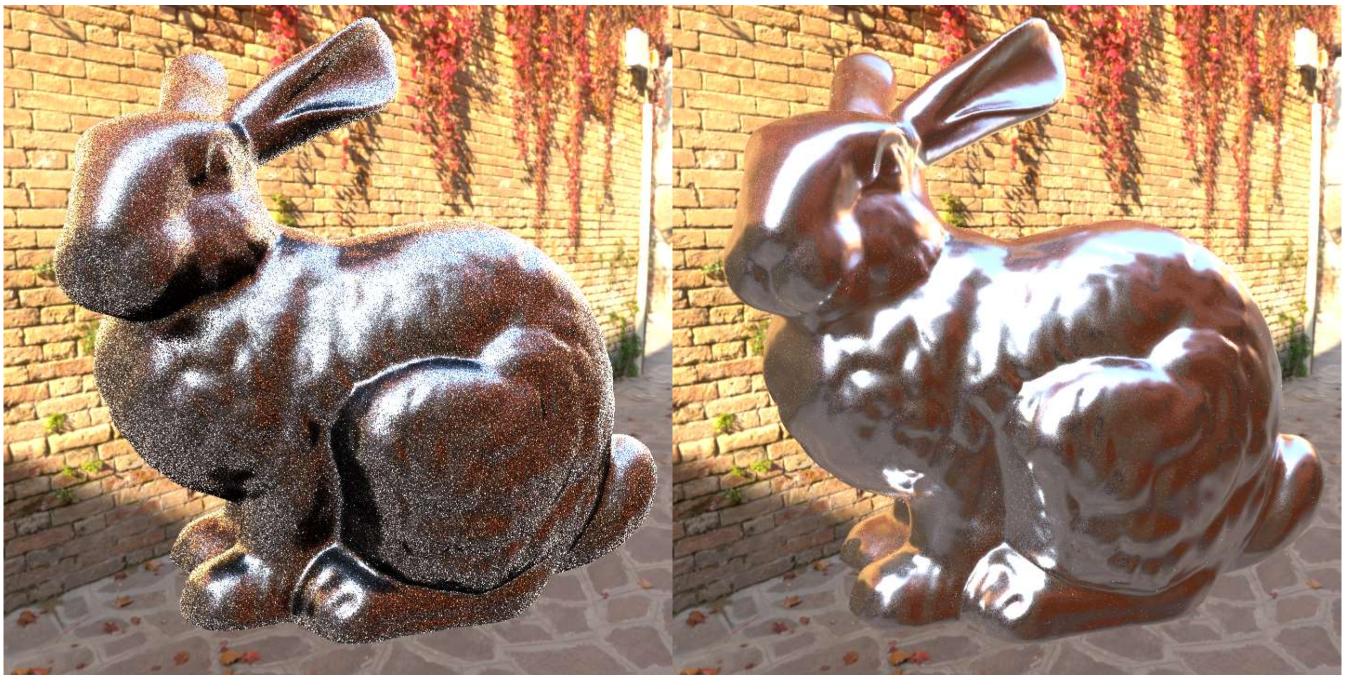
$$= \frac{F(\hat{\omega}_i, \hat{\omega}_m)\ G_2(\hat{\omega}_i, \hat{\omega}_o, \hat{\omega}_m)\ D(\hat{\omega}_m)|\hat{\omega}_i \cdot \hat{\omega}_g|}{4\ |\hat{\omega}_i \cdot \hat{\omega}_g|\ |\hat{\omega}_o \cdot \hat{\omega}_g|} * \frac{4|\hat{\omega}_o \cdot \hat{\omega}_m|}{D(\hat{\omega}_m)(\hat{\omega}_m \cdot \hat{\omega}_g)}$$

Conveniently you can see that a few terms cancel out and we're left with only this for our reflectance:

$$\frac{f(\hat{\omega}_i, \hat{\omega}_o)|\hat{\omega}_i \cdot \hat{\omega}_g|}{p_i(\hat{\omega}_m)} = \boxed{\frac{F(\hat{\omega}_i, \hat{\omega}_m)G_2(\hat{\omega}_i, \hat{\omega}_o, \hat{\omega}_m)|\hat{\omega}_o \cdot \hat{\omega}_m|}{|\hat{\omega}_o \cdot \hat{\omega}_g||\hat{\omega}_m \cdot \hat{\omega}_g|}}$$

From here we will transform $\hat{\omega}_i$ from the tangent space we've been working in back to world space, cast a ray in that direction to solve for incoming energy $L_i(X, \hat{\omega}_i)$, and multiply that by the reflectance we calculated here.

Now let's look at the results!



In order to have something to compare to on the left I generated an image by importance sampling a cosine distribution and on the right I generated an image using the technique described above. The scene is the Stanford bunny with roughness 0.05 lit only with an environment light. Both images were generated with 512 rays per pixel.

As you can see the results are significantly higher quality than importance sampling with just a cosine lobe. However, there are still quite a few fireflys visible in the image so this technique would require a lot more samples to converge. In my next post I will talk about why we still see so many fireflys and about a more recent technique from Eric Heitz and Eugene D'Eon to importance sample the distribution of visible normals that fixes some of these problems.

One last thing before we move on to part 2 is I want to provide source code to help ground all of this. You'll need to fill in some blanks but the key parts are covered here.

```
//====================================================================
static float3 SchlickFresnel(float3 r0, float radians)
{
    // -- The common Schlick Fresnel approximation
    float exponential = Powf(1.0f - radians, 5.0f);
    return r0 + (1.0f - r0) * exponential;
}

//====================================================================
// non height-correlated masking-shadowing function is described here:
static float SmithGGXMaskingShadowing(float3 wi, float3 wo, float a2)
{
    float dotNL = BsdfNDot(wi);
    float dotNV = BsdfNDot(wo);

    float denomA = dotNV * Sqrtf(a2 + (1.0f - a2) * dotNL * dotNL);
    float denomB = dotNL * Sqrtf(a2 + (1.0f - a2) * dotNV * dotNV);

    return 2.0f * dotNL * dotNV / (denomA + denomB);
}

//====================================================================
void ImportanceSampleGgxD(Random::MersenneTwister* twister, float3 wg,
                          float3 wo, Material* material,
                          float3& wi, float3& reflectance)
{
    float a = material->roughness;
    float a2 = a * a;

    // -- Generate uniform random variables between 0 and 1
    float e0 = Random::MersenneTwisterFloat(twister);
    float e1 = Random::MersenneTwisterFloat(twister);

    // -- Calculate theta and phi for our microfacet normal wm by
    // -- importance sampling the Ggx distribution of normals
    float theta = Acosf(Sqrtf((1.0f - e0) / ((a2 - 1.0f) * e0 + 1.0f)));
    float phi   = Math::TwoPi_ * e1;

    // -- Convert from spherical to Cartesian coordinates
    float3 wm = Math::SphericalToCartesian(theta, phi);

    // -- Calculate wi by reflecting wo about wm
    wi = 2.0f * Dot(wo, wm) * wm - wo;

    // -- Ensure our sample is in the upper hemisphere
    // -- Since we are in tangent space with a y-up coordinate
    // -- system BsdfNDot(wi) simply returns wi.y
    if(BsdfNDot(wi) > 0.0f && Dot(wi, wm) > 0.0f) {

        float dotWiWm = Dot(wi, wm);

        // -- calculate the reflectance to multiply by the energy
        // -- retrieved in direction wi
        float3 F = SchlickFresnel(material->specularColor, dotWiWm);
        float G = SmithGGXMaskingShadowing(wi, wo, a2);
        float weight = Absf(Dot(wo, wm))
                     / (BsdfNDot(wo) * BsdfNDot(wm));

        reflectance = F * G * weight;
    }
    else {
        reflectance = float3::Zero_;
    }
}
```

And that's it for today. For a description of the flaws with this method and a writeup about an even better method see part 2 (/post/ggximportancesamplingpart2) of this post. Thanks for reading!

(Edit 3/12/18: I found an error in the Smith masking shadowing equation and function and corrected it.)

References:

- Microfacet Models for Refraction through Rough Surfaces (https://www.cs.cornell.edu/~srm/publications/EGSR07-btdf.pdf)
- PBR Diffuse Lighting for GGX+Smith Microsurfaces (https://twvideo01.ubm-us.net/o1/vault/gdc2017/Presentations/Hammon_Earl_PBR_Diffuse_Lighting.pdf)
- Sampling microfacet BRDF (https://agraphicsguy.wordpress.com/2015/11/01/sampling-microfacet-brdf/)
- Notes on the Ward BRDF (https://www.graphics.cornell.edu/~bjw/wardnotes.pdf)

Additional thanks to:

- HDRI Haven (https://hdrihaven.com/hdri/?h=red_wall) for the free IBL
- My friends who helped review this. Special credit to Matt Pettineo (https://twitter.com/MyNameIsMJP) for providing a link to the Ward paper.

While I had an absolute blast writing this series and I definitely want to write more in the future I'm waiting to gague reactions before I decide how much sense it makes to put effort into the appearance of the site. Until I decide to flesh things out more you can leave comments at:

@schuttejoe (https://twitter.com/schuttejoe)