Plausible Blinn-Phong Reflection of Standard Cube MIP-Maps

Morgan McGuire Williams College
Daniel Evangelakos Williams College
James Wilcox NVIDIA
Sam Donow Williams College
Michael Mara NVIDIA

Abstract

We describe the technique used in the G3D Innovation Engine 9.00 to produce reasonable real-time environment lighting. It adds two lines of code to a pixel shader to reasonably approximate Lambertian and Blinn-Phong glossy reflection of a standard cube map environment with a MIP-chain without preprocessing. That is, we combine Blinn's BSDF with Blinn's environment mapping in a modern physically-based way. This technique has the advantage of operating without preprocessing on standard assets and dynamically-rendered light probes. A limitation is that it produces overly boxy blurring for highly varying environment maps with very broad glossy exponents. This technique has been widely explored before. Our contribution is a principled derivation of the constants needed to best approximate with a single texture sample per BSDF term, giving an efficient result in which glossy reflections of the environment and light sources have comparable highlights and intensity.

Technical Report CSTR201301, Williams College Department of Computer Science, Williamstown, MA, September 9, 2013
http://graphics.williams.edu/papers/EnvMipReport2013

1. Introduction

There are many more sophisticated ways of computing environment lighting for real-time rendering (see Real-Time Rendering's survey). Unfortunately, those techniques tend to require an offline asset build step. They can't be directly applied to dynamic light probes or in the context of many rendering frameworks. Those methods also tend to require a specific setup for the renderer that doesn't work with our default shaders. For research and hobbyist rendering, one often has to work with off-the-shelf

or legacy assets. This situation also arises in commercial projects that run across a variety of devices.

While this is a technique that probably will not interest next-generation console developers or the research community, it has two interesting properties. First, it may be useful to a number of mobile game developers, students, and hobbyists, because it is a lot better than having no approximation of environment lighting at all. Of course, our technique may also be useful to some more sophisticated productions and tools; Maya uses simple environment MIP maps[1] for lighting as previously described by Ashikhmin and Ghosh [2002] and doesn't even use our approximation to conserve energy or match glossy highlights.

Second, this is a nice example of how back-of-the-envelope computations and a principled understanding of the physics of rendering can very quickly yield practical improvements in image quality. Our approximation is an intentional approximation, not an arbitrary phenomenological trick, so it has the benefit of inherently robust and has known sources of errors.

2. Background on Environment Lighting

One reason that classic 3D computer graphics images (which includes nearly every video game until a few years ago) appear artificial is that they underestimate the amount of light on a surface. In the real world, light arrives at a surface from many directions. The brightest light usually comes directly from a light source such as the sun or a bulb. But there is other "fill" or "bounce" light arriving from all directions, such as the light that shines through a window and then reflects off the floor and walls. If there wasn't, then areas in "shadow" would be completely black, and thus invisible. Those sources of light are also area sources in the real world, that aren't just a single direction from which light arrives but a wedge around each surface point.

In attractive photographs, these effects are amplified to soften illumination and counteract the limited dynamic range of print and computer displays. That's why a studio photographer uses diffusers to create large rectangular light sources and places white reflectors around a model.

Classic computer graphics relies primarily on point light sources that cast harsh illumination from a single direction, plus some ambient terms. The offline rendering community adopted much more realistic illumination models decades ago, which is why CG images in well-produced films are now indistinguishable from true live action shots. Those methods tend to have poor computational efficiency and few guarantees on even that long run time, so they are not directly applicable for real-time rendering. However, in the last ten years there has been a lot of research on comparable effects

[1] As described in the product video http://www.youtube.com/watch?v=X6vnCotQ5Yw&feature=player_detailpage&t=379 at time 6:20

for real-time rendering. This is why many of the best-looking recently-released games (e.g., *Battlefield 4* and *The Last of Us*) now have some form of attractive lighting on dynamic objects.

The simplest technique for lighting a dynamic object is environment lighting. In this case, a cube map or other parameterization of the sphere stores the amount of light arriving from each direction, which is just a picture of the world around the object. Ideally, this cube map would be updated every frame to reflect changes in that lighting environment. However, it works surprisingly well even if the cube map is static, if there is a single cube map used throughout the entire scene despite the view changing, or even if the cube map doesn't look terribly much like the actual scene. The application of maps of indirect light for mirror reflection via environment mapping was first introduced by Blinn and Newell [1976]. Modern environment lighting was introduced by Gene Miller and Robert Hoffman; see Debevec's [2006] history of environment lighting for more details. The modern Miller and Hoffman form recognizes that shading is the convolution of a surface's reflection function with the lighting environment. The reflection function is driven by surface chemistry and surface roughness. For example, glass produces a sharp highlight; sand-blasted glass produces a broad, dull highlight; and dry clay always produces a broad, dull highlight. See Ramamoorthi's and Hanrahan's [2001] work on irradiance maps and a modern game developer's presentation of it at `http://www.insomniacgames.com/paper-exercise-an-efficient-representation-for-irradiance-environment-maps` for further discussion.

The convolution produces a convenient duality that a rough, matte surface under a point light looks about the same as a smooth, shiny surface under very diffuse lighting. This means that one can apply mirror environment mapping to a blurred cube map to approximate the appearance of a glossy surface. Computing that blur is a little tricky because we have to blur using the surface's reflectance function, not just any blur, and because there are some problems with not blurring over the horizon (i.e., tangent plane) of the surface. See Real-Time Rendering, 3rd edition for a full discussion. Some tools like AMD *CubemapGen* and *Skyshop* will compute this blur offline for various levels of roughness and store them in the MIP-levels of a cube map. Those MIP levels collectively model an array of increasingly blurry spheres).

3. Key Idea

Our goal is to approximate the convolution of an environment map with a cosine-power lobe parameterized on exponent s using a conventional cube map MIP chain. For Lambertian terms we'll estimate shading with a cube map sample from the lowest MIP (approximating cosine to the first power) about the normal, and for glossy terms we'll light with a cube map sample dictated by glossy exponent sampled about the

reflection vector. To do this, we need a formula for the MIP level to sample as a function of the glossy exponent. That formula is our contribution to environment lighting.

4. Limitations

We ignore the fact that the tangent plane masks the reflection at glancing angles, a common problem even in the good convolution techniques. Since the conventional MIP chain uses simple 2×2 averaging at each level and only within faces, and we then use (seamless) bilinear cubemap sampling, our final samples will represent convolution with a pyramid-density distorted quad instead of a power-cosine hemisphere. This is a source of error. It will be minimized near the center of faces and at high exponents (shiny surfaces) and worst at low exponents and corners. If the scene on average is about as bright (the lowest frequencies are nearly constant) in all directions despite lots of high frequency detail (such as in Sponza), then the approximation will work well. If the scene has highly varying illumination as a function of angle (such as in the Cornell Box), then everything will look slightly shinier than it should be because the lowest MIP levels do not actually blur across cube map faces.

These approximation errors could be minimized by taking multiple texture samples from the cube map...but if we cared about it that much, we would probably use a better method than sampling MIP maps. So we're going to accept these limitations and ask what is the best we can *reasonably* do with two texture fetches (Lambertian + glossy) from a conventional MIP map.

5. Derivation

The uniform weighted spherical cap of angle ϕ from center to edge subtends a solid angle of

$$2\pi \int_0^{\phi} \sin\theta \, d\theta = 2\pi(1 - \cos\phi). \tag{1}$$

This is just the integral of 1 over the cap; the sin term arises from the spherical parameterization in which it is easiest to take the integral. A power-cosine ($\cos^s\theta$) weighted hemisphere, such as the one arising in the Blinn-Phong factor $(\hat{\omega}_h \cdot \hat{\omega}_i)^s$, subtends a weighted solid angle with measure

$$\Gamma = 2\pi \int_0^{\pi/2} \cos^s\theta \sin\theta \, d\theta \tag{2}$$

(recall that we're ignoring the tangent plane clipping by not clamping that Blinn-Phong term against zero). Evaluating this integral shows us that in a completely uniform scene, the glossy reflection will have magnitude proportional to the solid angle

$$\Gamma = \frac{2\pi}{s+1}. \tag{3}$$

A cube map contains six $w \times w$ (where w is "width"; the faces are square) faces and covers the whole sphere of 4π steradians. MIP level m contains $6(w2^{-m})^2$ pixels. A cube map texel at level m thus subtends

$$T = \frac{4\pi}{6(w\,2^{-m})^2} \tag{4}$$

steradians on average, with the ones at the corners covering less and the ones at the centers of faces covering more. We just need to know what MIP-level m approximates integrating the same solid angle as the cosine exponent s, so we let $\Gamma = T$ and solve for m.

$$\begin{aligned}
\frac{2\pi}{s+1} &= \frac{4\pi}{6(w\,2^{-m})^2} && (5)\\
\frac{2^{2m}}{3w^2} &= \frac{1}{s+1} && (6)\\
2^{2m} &= \frac{3w^2}{s+1} && (7)\\
2m &= \log\left(\frac{3w^2}{s+1}\right) && (8)\\
2m &= \log(3w^2) + \log\left(\frac{1}{s+1}\right) && (9)\\
m &= \frac{1}{2}\log(3w^2) + \frac{1}{2}\log\left(\frac{1}{s+1}\right) && (10)\\
m &= \log(w\sqrt{3}) + \frac{1}{2}\log\left(\frac{1}{s+1}\right) && (11)\\
m &= \log(w\sqrt{3}) - \frac{1}{2}\log(s+1) && (12)
\end{aligned}$$

All logarithms are base 2 in this derivation. The $\log(w\sqrt{3})$ term depends solely on the cube map's MIP 0 resolution and can be precomputed, so we need only evaluate one logarithm and one fused multiply-add instruction per pixel. The logarithm is obviously the more expensive part. The cost of one logarithm in our pixel shader is insignificant compared to the other shading math, but one could store $\log(s+1)$ in addition to s in the roughness texture map if it was a major concern. See Listing 1 for the equivalent source code.

How one computes the Lambertian and glossy coefficients depends on the preferred shading model. We use the energy-conserving normalized Blinn-Phong with a Fresnel term, as described in *Real-Time Rendering* and the *Graphics Codex*. The

```
float MIPlevel =
     log2(environmentMapWidth * sqrt(3)) -
     0.5 * log2(glossyExponent + 1);

gl_FragColor.rgb =
    lambertianCoefficient * textureCubeLod(environmentMap,
                                           worldSpaceNormal,
                                           maxMIPLevel).rgb +
    glossyCoefficient     * textureCubeLod(environmentMap,
                                           worldSpaceReflectionVector,
                                           MIPlevel).rgb;
```

Listing 1.

normalized Blinn-Phong BRDF is

$$f(\hat{\omega}_i, \hat{\omega}_o) = \frac{1}{\pi}\left[k_L + k_G \frac{s+8}{8} \max\left(\frac{\hat{\omega}_i + \hat{\omega}_o}{||\hat{\omega}_i + \hat{\omega}_o||} \cdot \hat{n}, 0\right)^s\right]. \quad (13)$$

When integrating this over a cosine-weighted hemisphere the normalization constants are cancelled, so `lambertianCoefficient` = k_L and `glossyCoefficient` = k_G with no further factors needed. Note that this environment illumination superimposes on the direct illumination, and should produce highlights from light sources in the environment map that are similar to those of direct light sources, as shown in the teapot image in Figure 1.

In G3D 9.00 (`http://g3d.sf.net`), we also use ambient obscurance to modulate the glossy environment reflection for low exponents, since the entire hemisphere should not be integrated in those cases. The library includes our full BSDF and support code.

6. Examples

Artifacts from our approximations are most visible on completely smooth geometry such as the spheres reflecting the Uffizi in Figure 2. The cube face MIP boundaries become somewhat apparent on the mid-range glossy values at the center bottom of the grid.

Artifacts from our approximations are least visible on complex, textured geometry, such as the glossy bump-mapped crate in Figure 3. The environment map in that case is also relatively homogeneous at low frequencies, satisfying our uniformity assumptions.

References

ASHIKHMIN, M., AND GHOSH, A. 2002. Simple blurry reflections with environment maps. *J. Graph. Tools 7*, 4 (Dec.), 3–8. 2

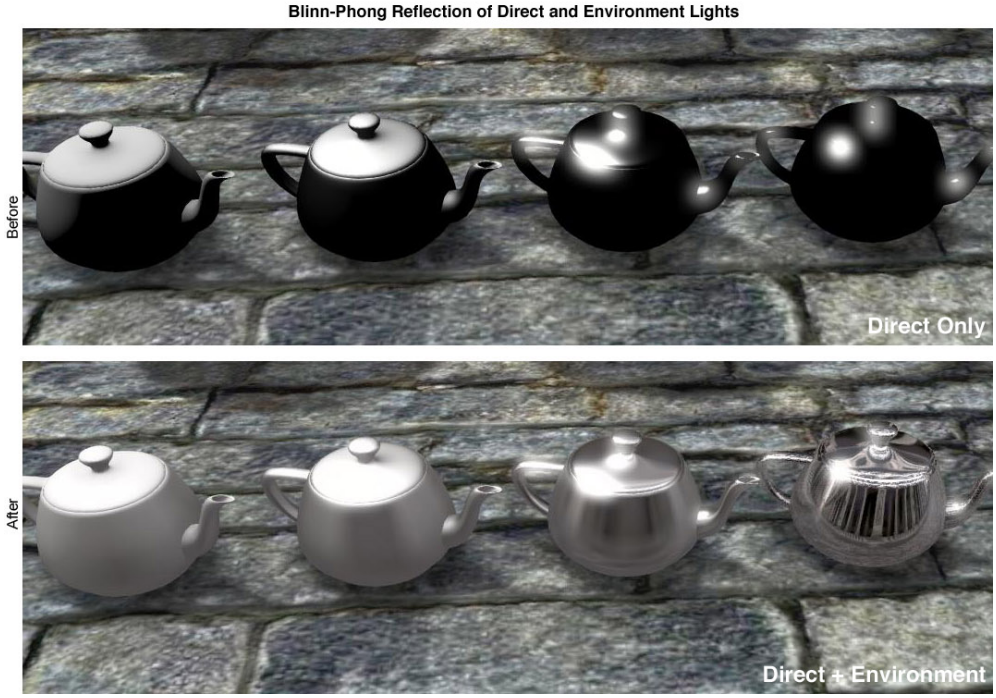

Figure 1.

BLINN, J. F., AND NEWELL, M. E. 1976. Texture and reflection in computer generated images. *Commun. ACM 19*, 10 (Oct.), 542–547. 3

DEBEVEC, P., 2006. The story of reflection mapping, September. Web page http://www.pauldebevec.com/ReflectionMapping/. 3

RAMAMOORTHI, R., AND HANRAHAN, P. 2001. An efficient representation for irradiance environment maps. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, New York, NY, USA, SIGGRAPH '01, 497–500. 3

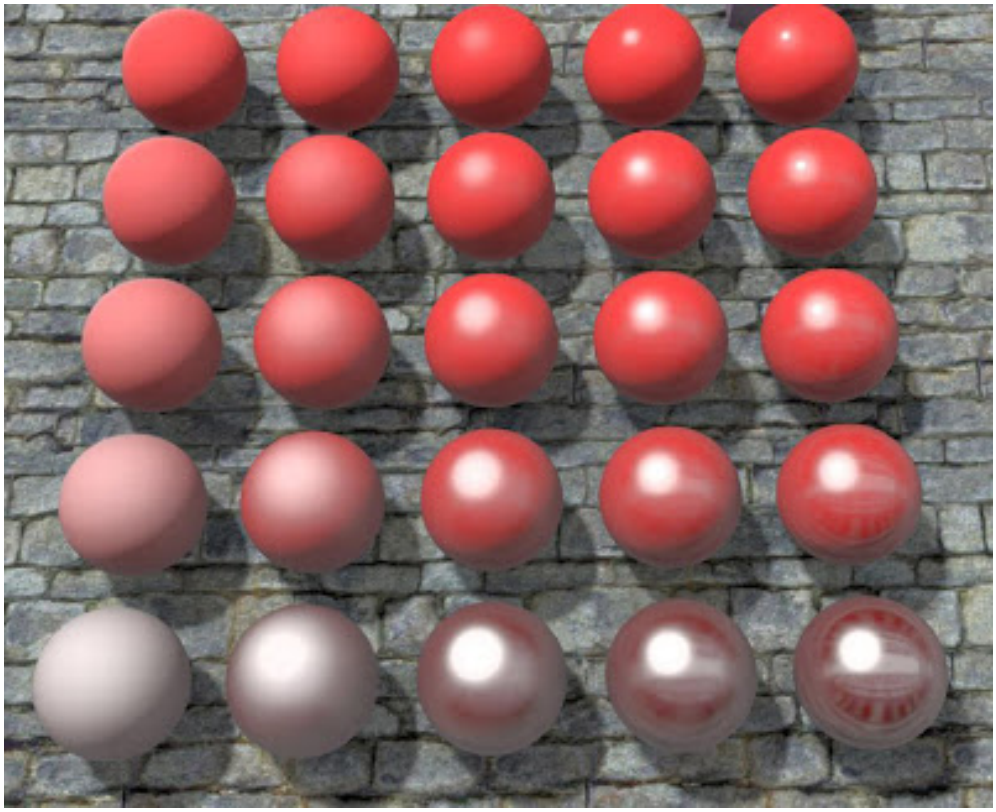

Figure 2.

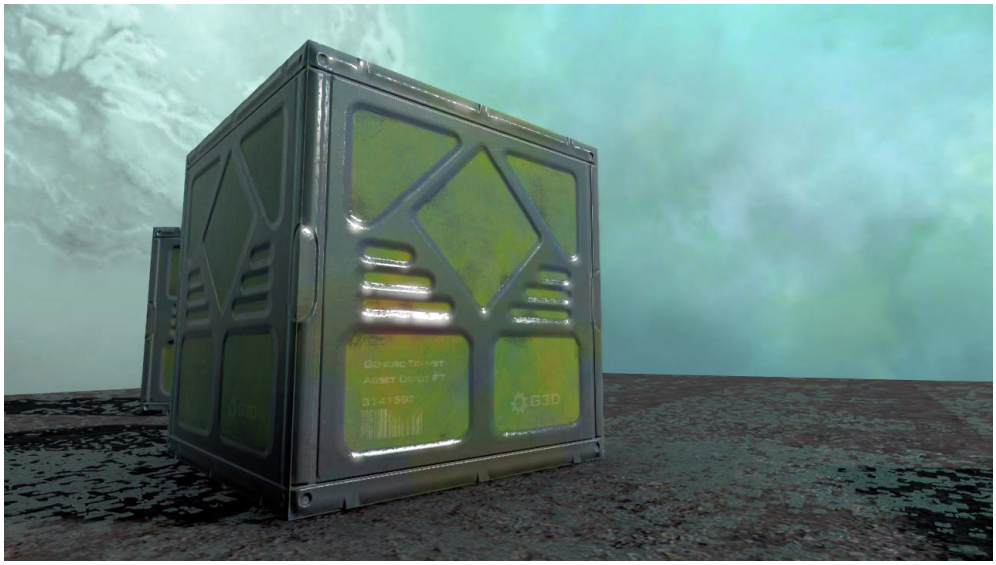

Figure 3.