Vol. 7, No. 4: 3–8

Simple Blurry Reflections with Environment Maps

Michael Ashikhmin and Abhijeet Ghosh
SUNY at Stony Brook

Abstract. We present a technique which uses existing OpenGL capabilities to approximate the effect of blurry specular reflections and indirect diffuse illumination. It makes use of environment maps, mipmapping with level of detail control, and possibly texture borders. The method is extremely simple to implement, in some cases requiring just a single additional OpenGL statement.

1. Introduction

Physically based rendering systems describe surface reflection behavior using the bidirectional reflectance distribution function (BRDF). They then use it as a weighing function applied, along with an extra cosine term, to the illumination incident upon the surface to compute the final color using the rendering equation [Kajiya 86]. This process can take into account light coming from all objects in the scene (indirect illumination) rather than only directly from light sources (direct illumination). In recent years, substantial research effort has been directed at incorporating both arbitrary BRDFs and indirect illumination effects into hardware-based rendering [Cabral et al. 99], [Heidrich and Seidel 99], [Ramamoorthi and Hanrahan 01]. In most cases, however, the techniques proposed are rather complex and require significant extra programming effort to implement. On the other hand, it is well known that accurate computation of these effects is not critical for most applications as long as the main character of the BRDF and illumination are both correctly reproduced. This led to the development of numerous ad-hoc approximation techniques

1086-7651/02 $0.50 per page

which still produce the desired visual experience. In this paper, we present one such method, the main advantage of which is extreme simplicity achieved by using existing OpenGL capabilities. Our work can be considered a simplification of the approach taken by Kautz et al. [Kautz et al. 00] Readers interested in a more detailed presentation of underlying ideas and theoretical background are referred to this work. An alternative approach of hardware-assisted multipass rendering [Diefenbach and Badler 97] can also be used to incorporate several illumination effects, including blurry reflections.

2. The Technique

Perhaps the simplest algorithm which can be used to approximate the effects in question is Environment Mapping [Blinn and Newell 76]. Environment map (EM) is simply a 2D array(s) which stores precomputed illumination incident on some point in the scene from all directions. There are at least three popular parameterizations of EMs: spherical, cubic, and parabolic [Heidrich and Seidel 98]. In its original form, EMs are most commonly used to incorporate mirror reflections into hardware-based systems with spherical (`GL_SPHERE_MAP` texture coordinate generation mode) and cubic (`GL_TEXTURE_CUBE_MAP_ARB` extention) maps currently included as a part of OpenGL and supported in hardware on many modern systems. We will use cubic maps in this paper since they start to emerge as the most popular method, but the technique can be used with other parameterizations as well, inheriting both their advantages and disadvantages.

As has been discussed by several researchers (most recently in [Cabral et al. 99], [Kautz et al. 00], [Ramamoorthi and Hanrahan 01]), if we first appropriately preblur the EM, it can then be successfully used to approximate reflections with arbitrary BRDFs. To do this correctly, the result has to be view- and/or surface normal-dependent (at the very least because of the extra cosine weighting present in the rendering equation) requiring significant computational effort and storage.

We will use an even simpler strategy of view-independent blurring which we found to produce surprisingly convincing results. Moreover, we notice that if we are interested only in surfaces with Phong-style single lobe specular reflections, which are arguably the most common surfaces in both computer graphics and real world, all the necessary components are already available in standard OpenGL. In particular, if a mipmap is constructed for the EM, higher mipmap levels already store blurry versions of EM which can be directly used. Since high accuracy is not required for a blurry reflection, the exact algorithm used to construct the mipmap is not important. All we now need to turn a mirror reflection obtained with an EM into a blurry one is to instruct the rendering system to use only mipmap levels above some fixed scale factor λ

during rendering. Fortunately, it is easy to do so in OpenGL with the following statement:

```
glTexParameterf(GL_TEXTURE_CUBE_MAP_ARB,
                GL_TEXTURE_MIN_LOD, lambda);
```

where `lambda` is the desired mipmap level. This statement can be issued just before an object is rendered, with different λ for different objects, but with *the same* EM, simulating materials with different Phong exponents. This is contrary to the common preblurring methods which usually require creating separate blurry versions of the same EM for each material. This statement also does not interfere with other material parameters used, for example, in a separate direct lighting calculations. Since we turn off only low mipmap levels and keep higher ones in place, antialiasing properties of the system, if anything, are improved, as one would expect in case of a generally more blurry image.

We can also note that with extreme blurring (high mipmap levels) and `GL_NORMAL_MAP_ARB` texture generation mode, the appearance of a diffuse surface under illumination given by the EM can be obtained.

We emphasize again that our technique is designed for a very specific class of Phong-style and diffuse reflectors and uses view-independent prefiltering which does not include cosine weighting required by exact theory. Therefore, it is not physically correct for the more diffuse surfaces, so although it can be used as a preview of more advanced global illumination algorithms, it should be used with care in such cases. Also, as any EM-based technique, only reflections of distant objects can be included.

The results of our procedure (with some important details described below) are shown in Figure 1 which demonstrates a mirror reflecting teapot, two Phong-style teapots, and a diffuse teapot, all obtained with the same 128x128 cubic EM shown. We found the results to look surprisingly convincing given the simplicity of the technique.

3. Difficulties with Current Hardware

If mipmapped EMs are applied in a consistent way by the hardware, the single OpenGL statement from the previous section is sufficient to turn a mirror object into a Phong-like or diffuse (with change of coordinate generation mode in the second case). “Consistent” in this case means that mipmapping should be done with filtering *across* the cube face boundaries. Current hardware, however, filters mipmaps *separately* for each of the cube faces which produces noticeable seams if high λ values are used. Note that this is clearly incorrect behavior and it is used currently only due to performance considerations [1].

[1] According to a review of a GeForce4-based card at http://www.3dbeyond.com, correct filtering is available, but disabled by default. However, we were not able to locate any information on how to turn on this feature.

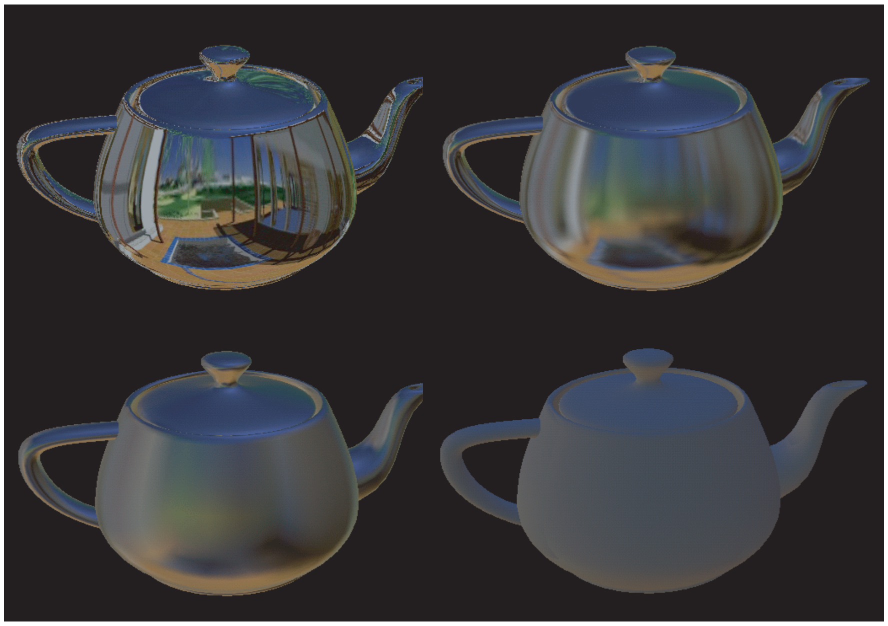

Figure 1. Teapots with different values of λ: 0.0 (original cube EM), 3.5, 5.0, and 7.0. In the last case, texture coordinates were generated based on surface normal. Notice slight trace of cube EM face boundaries which is due to our imperfect filtering procedure.

Note also that for only slightly blurry reflections or if spherical maps are used instead of cube ones, this issue is not important.

In the absence of consistent EM mipmapping, we can use texture borders to emulate the correct behavior. In this case, mipmap levels for each cube face have to be constructed by hand since `gluBuild2DMipmaps` does not support borders. We simply combine a cube face with the four adjacent faces into a single texture, filter this larger texture to the needed level (we use repeated application of a simple box filter) and cut out the central part to use in `glTexImage2D` with border parameter set to 1. Texture wrap mode should be set to `GL_CLAMP`. For simplicity, we fill in the corners of the larger texture directly by the images from the cube faces as shown in Figure 2. Although this strategy produces satisfactory results in most cases, some seam artifacts can still be observed for diffuse surfaces and more sophisticated filtering methods might be needed in this case if it is critical to better hide the seams.

Unfortunately, current NVIDIA graphics cards, although they produce correct results, seem to default to software rendering when texture borders on EMs are used. Given that on GeForce2 even `GL_REPEAT` on EMs and `GL_CLAMP`

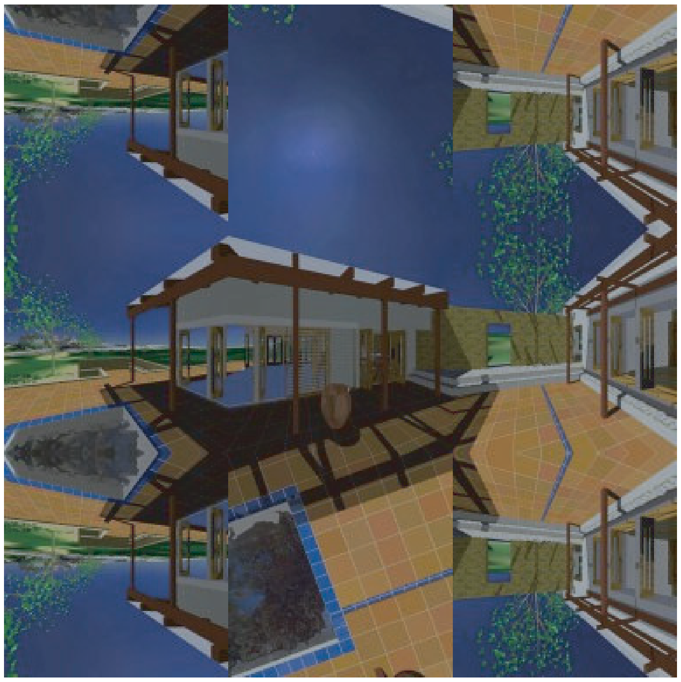

Figure 2. The initial image for the cube map filtering procedure. Such images are constructed for each of the six faces and then filtered to obtain mipmap levels. We simply copy one of the available cube face textures to fill in the corners.

with fixed color borders are not hardware supported while on GeForce3 they are, increasing hardware support for texture borders seems to be the current trend. We expect this to continue, partially because they are part of the OpenGL core and partially because of the cube EM functionality itself. We sincerely hope that this paper will provide a compelling case for hardware implementation of consistent EM filtering and/or texture borders in future graphics hardware. Another useful simple extension would be the ability to specify the minimal mipmap level on a per-vertex basis directly (through, for example, a variant of `glTexCoord*`) in which case such noticeable effects as sharper reflections near the grazing angle could be simulated.

References

[Blinn and Newell 76] James Blinn and M. Newell. "Texture and Reflection in Computer Generated Images." *Communications of the ACM* 19:10 (1976), 542–547.

[Cabral et al. 99] B. Cabral, M. Orlano, and P. Nemen. "Reflection Space Image-Based Rendering." In *Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series*, edited by Alyn Rockwood, pp. 165–170, Reading, MA: Addison Wesley Longman, 1999.

[Diefenbach and Badler 97] Paul J. Diefenbach and Norman I. Badler. "Multi-Pass Pipeline Rendering: Realism for Dynamic Environments." In *Symposium on Interactive 3D Graphics*, pp. 59–70, City: Publisher, 1997.

[Heidrich and Seidel 98] W. Heidrich and H.-P. Seidel. "View-Independent Environment Maps." In *Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pp. 39–45, New York: ACM Press, 1998.

[Heidrich and Seidel 99] W. Heidrich and H.-P. Seidel. "Realistic, Hardware-Accelerated Shading and Lighting." In *Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series*, edited by Alyn Rockwood, pp. 171–178, Reading, MA: Addison Wesley Longman, 1999.

[Kajiya 86] James T. Kajiya. "The Rendering Equation." *Proc. SIGGRAPH '86, Computer Graphics* 20:4 (1986), 143–150.

[Kautz et al. 00] J. Kautz, P. Vazquez, W. Heidrich, and H.-P. Seidel. "A Unified Approach to Prefiltered Environment Maps." In *Eurographics Workshop on Rendering*, pp. 185–196, City: Publisher, 2000.

[Ramamoorthi and Hanrahan 01] R. Ramamoorthi and P. Hanraham. "An Efficient Representation for Irradiance Environment Maps." In *Proceedings of SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Series*, edited by E. Fiume, pp. 497–500, Reading, MA: Addison-Wesley, 2001.

Web Information:

http://www.acm.org/jgt/papers/AshikhminGhosh02

Michael Ashikhmin, Computer Science Department, SUNY at Stony Brook, Stony Brook, NY 11794 (ash@cs.cunysb.edu)

Abhijeet Ghosh, Computer Science Department, SUNY at Stony Brook, Stony Brook, NY 11794 (ghosh@cs.cunysb.edu)

Received April 2002; accepted September 2002.