OpenSG Symposium (2003)
D. Reiners (Editor)

Advanced Environment Mapping in VR Applications

Jan Kautz, Katja Daubert, and Hans-Peter Seidel

Max-Planck-Institut für Informatik
Saarbrücken, Germany

Abstract

In this paper, we propose an approach for rendering diffuse and glossy reflections using environment maps. This approach is geared towards VR applications, where realism and fast rendering is important. We exploit certain mathematical properties of diffuse reflections and certain features of graphics hardware for glossy reflections. This results in a very fast, single-pass rendering algorithm, which even allows to dynamically vary the incident lighting.

Keywords: Hardware Acceleration, Shading Models, Global Illumination, Texture Mapping, Frame-Buffer Tricks.

1. Introduction

Environment maps[3] are a widely used technique to approximate specular reflections in interactive rendering. Although environment maps make the assumption that the reflected environment is far away — thus being an approximation — they nevertheless achieve convincing results.

Recently, environment maps have been introduced as a means to render reflections from diffuse and glossy materials[4, 9, 10, 12, 14, 16, 28, 18, 29, 21]. We will propose a combination of different techniques, which allows to render diffuse and glossy reflections in a single pass. Our proposed approach — in contrast to many other techniques — also allows dynamically changing incident lighting. We therefore believe, that this technique is well-suited for virtual reality applications.

1.1. Overview

After reviewing related work, we explain the mathematical background of rendering reflections with filtered environment maps (Section 3). We then show, how this filtering can be done in real-time (Section 4). Rendering (Section 5) turns out to be very simple, and can be even done in a single pass (including filtering). Finally, we show what kind of results can be achieved by our technique and conclude.

2. Prior Work

Blinn and Newell[3] first introduced the *environment map* technique for producing mirror-like reflections on curved objects. An environment map stores the radiance incident from all directions at a single point, see Figure 1 for a 2D example. A reflection on an object is created by computing the reflected viewing direction (reflected about the surface normal) and then using this reflection direction for a lookup into the environment map. Since the environment map is only valid for a single point but a real object has some extent, this technique introduces some parallax error. It basically assumes that the environment is at infinity.

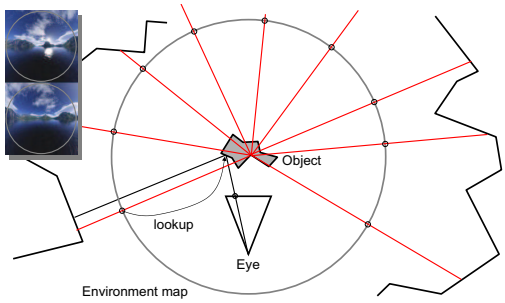

Figure 1: *Radiance incident at a single point is stored in an environment map. Here, we have used the parabolic parameterization[11] to store the incident radiance.*

Greene[9, 10] observed that a pre-convolved environment map could be used to simulate diffuse and glossy reflec-

tions. Instead of storing the incident radiance, Greene simply stored exit radiance, i.e. the incident radiance already integrated against the BRDF; Figure 2 depicts this method. This is the basis which most environment map methods are derived from, including the ones we propose to use for VR applications.

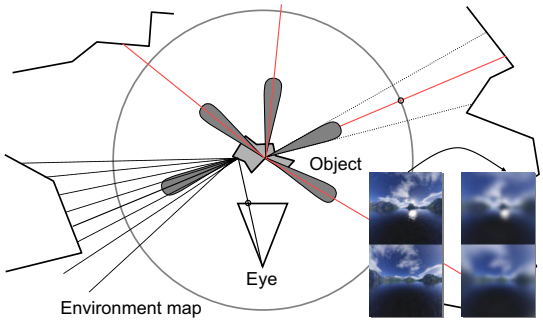

Figure 2: *Filtered environment maps store exit radiance, i.e. radiance filtered with a BRDF, instead of incident radiance.*

Several approaches have been proposed to simulate glossy reflections based on pre-filtered environment maps[4, 9, 10, 12, 14, 16, 28, 18, 29, 21]. These algorithms assume a simple, fixed BRDF model[9, 10, 12, 16, 21] (e.g. Phong model[1] or Lafortune model[17]) or generalize but only to isotropic BRDFs[4, 14]. This generalization either makes the assumption that the BRDF (times the cosine between surface normal and light direction) is radially symmetric[14], or it assumes BRDFs with a central reflection direction[4], resulting in a four-dimensional environment map. Ramamoorthi et al. propose view-dependent filtering of an environment map for isotropic BRDFs with spherical harmonics[29], which is not quite interactive for changing illumination. For the more general case of arbitrary anistropic BRDFs some new techniques[15, 19] have been proposed, which are similar in spirit to Ramamoorthi et al.[28, 29]. These two techniques also use spherical harmonics to convolve the environment map, but work with arbitrary BRDFs by tabulating the view-dependent kernel's coefficients. Unfortunately, these general methods also don't work in real-time. Latta and Kolb[18] propose a compression technique for full four-dimensional environment maps.

Nonetheless, techniques exist to filter environment maps on-the-fly. Kautz et al. [16] introduced a method, which convolves environment maps with the help of graphics hardware. The filter kernels are limited to approximations of BRDFs [14]. For the special case of a diffuse BRDF, Ramamoorthi and Hanrahan [28] propose a fast method based on spherical harmonics for computing the convolution in real-time.

Since environment maps are defined over the sphere, a way has to be found to represent them in two-dimensional textures. A widely used format in software renderers are cube maps, which are now commonly supported by graphics hardware [25]. Cube maps have the advantage of mapping spherical lines onto lines in texture space, but need six faces, which can cause artifacts across boundaries. A spherical parameterization, which is directly supported by OpenGL, was introduced by Blinn and Newell [3]. It only needs one texture map, but has a poor sampling rate near the horizon. Heidrich and Seidel [11] proposed dual paraboloid mapping which uses two texture maps, one for the front facing hemisphere and one for the backfacing, see Figure 1. This parameterization is now also supported by a variety of newer graphic boards.

2.1. Glossy Reflections

A few other techniques have been proposed for interactive rendering of glossy reflections, which are not based on environment maps. Diefenbach and Badler [7] used multi-pass methods (Monte Carlo integration) to generate glossy reflections. Photon maps [13] were used by Stürzlinger and Bastos [32]; photons were "splatted" and weighted with an arbitrary BRDF. Precomputed glossy reflections were stored in surface light fields by different authors [23, 34, 5]. Bastos et al. [2] used a convolution filter in screen-space to produce glossy reflections. Lischinski and Rappoport [20] used a large collection of low resolution layered depth images to store view-dependent illumination. These methods are either limited to fixed illumination or require planar geometry.

3. Environment Map Filtering Overview

Generally speaking, prefiltered environment maps capture all the reflected exitant radiance towards all directions $\hat{v}$ from a fixed position $\underline{x}$:

$$L_{\text{env}}(\underline{x};\hat{v},\hat{n},\hat{t}) = \int_{\Omega} f_r(\hat{\omega}(\hat{v},\hat{n},\hat{t}),\hat{\omega}(\hat{l},\hat{n},\hat{t}))L_{in}(\underline{x};\hat{l})(\hat{n}\cdot\hat{l})\,d\hat{l}, \tag{1}$$

where $\hat{v}$ is the viewing direction and $\hat{l}$ is the light direction in world-space, $\{\hat{t},\hat{n}\times\hat{t},\hat{n}\}$ is the local coordinate frame of the reflective surface, $\hat{\omega}(\hat{v},\hat{n},\hat{t})$ represents the viewing direction and $\hat{\omega}(\hat{l},\hat{n},\hat{t})$ the light direction relative to that frame, f_r is the BRDF, which is usually parameterized via the local viewing and light direction. A prefiltered environment map stores the radiance of light reflected towards the viewing direction $\hat{v}$, which is computed by weighting the incoming light L_{in} from all directions $\hat{l}$ with the BRDF f_r. Note, that L_{in} can be viewed as the unfiltered original environment map. This map should use high-dynamic range radiance values to be physically plausible. As you can see, in the general case we have a dependence on the viewing direction as well as on the orientation of the reflective surface, i.e. the local coordinate frame $\{\hat{n},\hat{t},\hat{n}\times\hat{t}\}$.

This general kind of environment map is five dimensional. Two dimensions are needed to represent the viewing direction $\hat{v}$ (a unit vector in world coordinates) and three dimensions are necessary to represent the coordinate frame

$\{\hat{n}, \hat{t}, \hat{n} \times \hat{t}\}$; e.g. three angles can be used to specify the orientation of an arbitrary coordinate frame.

The filtered environment maps which we will use drop some dependencies (e.g. on the tangent $\hat{t}$) and are reparameterized (e.g. the lookup is not done with the viewing direction $\hat{v}$, but the reflected viewing direction).

3.1. Diffuse Environment Maps

Miller[22] has proposed to use a purely diffuse BRDF to prefilter environment maps. A diffuse BRDF can be written as:

$$f_r(\hat{v}, \hat{l}) \;:=\; \frac{k_d}{\pi}, \tag{2}$$

where $k_d \in [0,1]$ describes the absorption of the surface. Moving this into Equation 1, we get:

$$L_{\text{diffuse}}(\underline{x}; \hat{v}, \hat{n}, \hat{t}) \;=\; \int_\Omega \frac{k_d}{\pi} L_{in}(\underline{x}; \hat{l})(\hat{n} \cdot \hat{l})\, d\hat{l}. \tag{3}$$

We can drop all dependencies except the one on the normal $\hat{n}$ and we obtain the following two dimensional environment map:

$$L_{\text{diffuse}}(\underline{x}; \hat{n}) \;=\; \frac{k_d}{\pi} \int_\Omega L_{in}(\underline{x}; \hat{l})(\hat{n} \cdot \hat{l})\, d\hat{l}. \tag{4}$$

This environment map accurately stores the diffuse illumination at the point $\underline{x}$. It is only two-dimensional and it is indexed by the surface normal.

Generating such a filtered environment map seems expensive, since the original environment map needs to be filtered with a hemispherical kernel. But recently Ramamoorthi and Hanrahan[28] showed that a diffuse environment map can be quickly generated and represented using spherical harmonics. Rendering from this representation can be accelerated by graphics hardware and works in real-time. This is the technique we propose to use for VR applications, see Section 4.1.

3.2. Phong Environment Maps

Heidrich[12] and Miller[22] used the original Phong reflection model[27] to prefilter environment maps. The Phong BRDF is given by:

$$f_r(\hat{v}, \hat{l}) \;:=\; k_s \, \frac{(\vec{r}_v(\hat{n}) \cdot \hat{l})^N}{(\hat{n} \cdot \hat{l})}, \tag{5}$$

where $\hat{r}_v(\hat{n})$ is the reflected viewing-direction in world-space. The parameters k_s and N are used to control the shape and size of the lobe. Using the Phong model, Equation 1 becomes

$$\begin{aligned} L_{\text{phong}}(\underline{x}; \hat{v}, \hat{n}, \hat{t}) \;&=\; \int_\Omega k_s \frac{(\hat{r}_v(\hat{n}) \cdot \hat{l})^N}{(\hat{n} \cdot \hat{l})} L_{in}(\underline{x}; \hat{l})(\hat{n} \cdot \hat{l})\, d\hat{l} \\ &=\; k_s \int_\Omega (\hat{r}_v(\hat{n}) \cdot \hat{l})^N L_{in}(\underline{x}; \hat{l})\, d\hat{l}. \end{aligned} \tag{6}$$

Obviously the tangent $\hat{t}$ is not used and can be discarded. Instead of indexing the environment map with $\hat{v}$ and $\hat{n}$, it can be reparameterized so that it is directly indexed by the reflection vector $\hat{r}_v$:

$$L_{\text{phong}}(\underline{x}; \hat{r}_v) \;=\; k_s \int_\Omega (\vec{r}_v \cdot \hat{l})^N L_{in}(\underline{x}; \hat{l})\, d\hat{l}. \tag{7}$$

Although the Phong model is not physically based, the reflections make a surface look metallic, only at grazing angles one expects sharper reflections. This indexing via the reflection vector $\hat{r}_v$ is also used for specular environment maps and is therefore supported in OpenGL[24] via the spherical, parabolic and cube map parameterizations.

Miller[22] and Heidrich[12] proposed to use a weighted sum of a diffuse and a Phong environment map to get a complete illumination model. They also propose to add a Fresnel term so that the ratio between the diffuse and glossy reflections can vary with different viewing angles:

$$L_o(\hat{r}_v, \hat{n}) \;=\; (1 - F(\hat{r}_v \cdot \hat{n}))L_{\text{diffuse}} + F(\hat{r}_v \cdot \hat{n})L_{\text{phong}} \tag{8}$$

This way a wide range of materials can be created. Other BRDFs than the Phong BRDF can be approximated using different filter kernels[14].

4. Real-Time Filtering

In this section, we will propose how an environment map can be filtered in real-time. This filtering depends on the desired BRDF. We will first show a method for filtering of diffuse environment maps based on Ramamoorthi and Hanrahan's method[28]. Then we show how glossy environment maps can be filtered using a simple space-invariant filter kernel.

4.1. Diffuse Environment Map

Equation 4 describes how an original environment map L_{in} has to be filtered. Basically, a hemispherical cosine kernel has to be applied to L_{in}. Since this is a low-frequency kernel, the resulting filtered environment map will also be low-frequency.

Ramamoorthi and Hanrahan[28] proposed to do this filtering in frequency space in order to exploit the low-frequency nature of the kernel. This can be done using spherical harmonics[8], which are the analogue on the sphere to the Fourier basis on the line or circle.

First we would like to introduce the spherical harmonic basis, before continuing with the filtering process. We omit the actual definition of spherical harmonics (can be found in [8]), instead we show only the first 9 basis functions, as those suffice for our purposes. Spherical harmonics Y_i are simply polynomials in the cartesian components of a unit

vector $\hat{l} = (x,y,z)$:

$$\begin{aligned} Y_0 &= \frac{1}{\sqrt{4\pi}}, \\ Y_{1;2;3} &= \sqrt{\frac{3}{4\pi}}(y;z;x), \\ Y_{4;5;7} &= \sqrt{\frac{15}{4\pi}}(xy;yz;xz), \\ Y_6 &= \sqrt{\frac{5}{16\pi}}(3z^2-1), \\ Y_8 &= \sqrt{\frac{15}{16\pi}}(x^2-y^2), \end{aligned} \tag{9}$$

Now we can represent an environment map using the spherical harmonics basis:

$$L_{in}(\hat{l}) = \sum_i L_i Y_i(\hat{l}), \tag{10}$$

where the coefficients L_i are computed numerically:

$$L_i = \int_\Omega L_{in}(\hat{l}) Y_i(\hat{l}) d\hat{l}. \tag{11}$$

As it turns out[28], convolving an environment map in spherical harmonics with the hermispherical cosine kernel is very simple. In fact, convolution and lookup with the surface normal $\hat{n}$ can be combined:

$$L_{\text{diffuse}}(\hat{n}) \approx \frac{k_d}{\pi} \sum_{i=0}^{9} \hat{A}_i L_i Y_i(\hat{n}), \tag{12}$$

where $\hat{A}_0 = \pi$, $\hat{A}_{1;2;3} = \frac{2}{3}\pi$, and $\hat{A}_{4;5;6;7;8} = \frac{1}{4}\pi$. For a complete derivation, please see [28].

This filtering and lookup step is so simple, that it can be easily implemented in a vertex shader on modern graphics hardware. Per-pixel evaluation is also possible but not necessary, since $L_{\text{diffuse}}(\hat{n})$ only varies slowly.

4.2. Glossy Environment Map

As seen in Section 3.2, filtering an environment map uses a two-dimensional filter kernel, since the input is two-dimensional as well. The shown Phong filter kernel is shift-invariant over the sphere. Unfortunately, mapping an environment map to a two-dimensional texture, e.g. cube maps[25] or parabolic maps[11], makes the filter kernel shift-variant in texture space.

Kautz et al.[16] proposed to filter an environment map using the convolution operation supported by graphics hardware. They use parabolic maps[11] to represent the environment. This leads to a good approximation of the shift-variant filter kernel with a two-pass algorithm. Hence this approach works well for the Phong model, as described in Section 3.2. It achieves fast filtering rates, that even allow dynamic filtering of environment maps. Unfortunately, the filtering cost for almost specular BRDFs becomes expensive.

Here, we propose to use a much simpler and faster method. Looking at Equation 7 describing the filtering process, it becomes clear that an environment map can be filtered with kernels other than the Phong kernel. Since the Phong model is not physically correct anyway, it is a valid approximation to choose a different filter kernel, which leads to a faster filtering method.

We decided to use a cube map representation together with a simple box-filter (in texture space) instead of the Phong cosine kernel. The reason is, that the box-filter is directly supported by graphics hardware through a feature which automatically generates mip-maps[33] using a box-filter[†]. Obviously, this shift-invariant box-filter in texture space becomes shift-variant over the sphere. This means, that for different reflection directions $\hat{r}_v$ (see Equation 7), a slightly different filter kernel is used. But as already noted by Kautz et al.[16], this is hardly noticeable in practice.

Although this box-filter does not correspond to a physically meaningful BRDF, the generated reflections are similar to the ones of a Phong BRDF, see Section 6.

5. Rendering

Rendering an object with on-the-fly filtered environment maps is very simple. First we would like to quickly describe how an environment map can be acquired on-the-fly and then how we render with it.

5.1. Environment Map Acquisition

The goal is to capture all the incident radiance $L_i(\hat{l})$ at a fixed position $\underline{x}$ (usually being the center of the reflective object). As we said before, we want to store the incident radiance in a cube map.

This allows to quickly acquire an environment map. From $\underline{x}$ we need to render the environment six times, i.e. one pass for every face of the cube map. All we need to do, is to set the field-of-view to 90 degrees and to rotate the view direction according to every face. We can either render directly into a texture map, or load the texture for each face from the frame buffer.

Alternatively, we can use a precomputed or digitized environment map, if the incident illumination does not change.

5.1.1. Projection into Spherical Harmonics

As part of the environment map acquisition, we need to represent it in spherical harmonics as well (for diffuse filtering). To do so, we need to numerically evaluate Equation 11 for all 9 basis functions. As shown by Sloan et al.[31], a very small

[†] When this feature is enabled, the hardware will automatically compute all mip-map levels for every texture map, even if the texture map is grabbed from the frame buffer.

Figure 3: *Different models illustrating the proposed method. Bunny model with $f_\lambda = 0.05$ and $b = 2.3$, bust with $f_\lambda = 0.24$ and $b = 1.5$, car with $f_\lambda = 0.15$, $b = 0.2$, and $b = 3.3$. All models are rendered with a frame rate of more than 100Hz on an ATI Radeon 9700.*

environment map (even 8×8 pixels per face) is sufficient to achieve accurate results. This integration can be performed in about 2 milliseconds (Athlon 1Ghz), yielding the SH coefficients L_i.

5.2. Rendering

Rendering the object with the diffusely filtered environment map is simple. All we need to do, is to evaluate Equation 12. As has already been shown[28], this can be done in a vertex shader.

Rendering the glossy part is even easier. As we have explained before, filtering is automatically done by the hardware. We only need to select an appropriate mip-map level for rendering (higher levels correspond to blurrier reflections). This can be easily done by setting the LOD bias b for mip-mapping[26].

Now the diffuse and the glossy part needs to be combined according to Equation 8. For the Fresnel term F, we use Schlick's approximation[30], which has only one parameter f_λ corresponding to the color of reflected white light at normal incidence. This Fresnel approximation is also evaluated in the vertex shader. The actual sum is computed in a single pass using the OpenGL combiner extension.

6. Results and Discussion

In Figure 3, different renderings illustrate our method. The bias parameter b selects how glossy the reflections are. Since our method only needs a single pass on all modern graphics hardware, rendering is very fast.

Figure 6 compares renderings with different settings for b and f_λ. As can be seen in these image, increasing f_λ simulates more reflective objects. Rendering is performed with about 500fps.

A comparison with the method by Kautz et al.[16] is shown in Figure 5. The parameters are chosen such that the reflections have a comparable glossiness. Differences are not very noticable. The box-filtered version exhibits some blockiness, but would go away, if graphics hardware supported better filter kernels (e.g. cubic filtering). The box-filtered version is much faster than the previous method[16], which is partially because it needs only a single pass, but also because the box filter can be evaluated much faster.

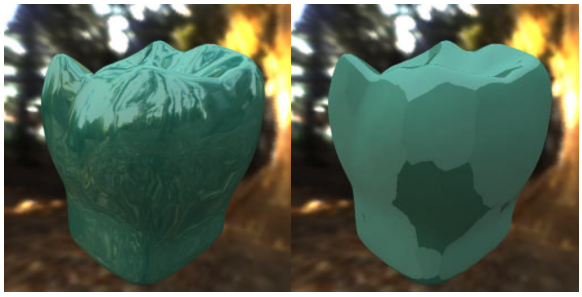

Figure 4: *Border problem. If b is high (here $b = 9.4$), the six different faces will show up. This can be avoided with a slightly more expensive implementation.*

The proposed method has one drawback. The hardware accelerated mip-map generation filters each cube map face invididually. This means that for the highest mip-map level, there are six different down-filtered values. These six different values will show up, when b is high, see Figure 4 for an example. This problem can be avoided with a filtering implementation, that pays attention to the borders, i.e. filters across faces.

7. Conclusions and Future Work

We have presented a combination of methods, which allows fast, single-pass rendering of reflective objects. The object's material can have a diffuse as well as a glossy component. Incident lighting is represented as an environment map,

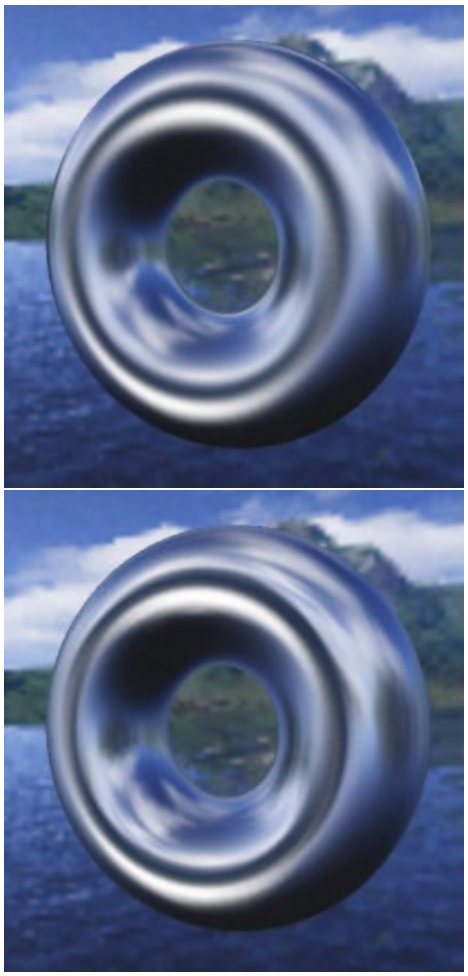

Figure 5: *Top image: Previous method*[16] *with exponent* $N = 225$*; renders at about 25fps. Bottom image: Our method with* $b = 2.25$*; renders at 500 frames per second.*

which allows globally incident lighting, and not only point or parallel light sources.

Dynamic lighting is possible, as the proposed approach allows to interactively filter the environment map in order to render the diffuse and glossy reflections.

We believe, that this approach is very well suited for VR applications. It is fast and does not need complicated multi-pass rendering. We have implemented this technique within OpenSG, and have made good experience with it.

8. Acknowledgements

We would like to thank Paul Debevec for the permission to use his high-dynamic range environment maps [6] (available from www.debevec.org).

References

1. D. Banks. Illumination in Diverse Codimensions. In *Proceedings SIGGRAPH*, pages 327–334, July 1994.
2. R. Bastos, K. Hoff, W. Wynn, and A. Lastra. Increased Photorealism for Interactive Architectural Walkthroughs. In *1999 ACM Symposium on Interactive 3D Graphics*, April 1999.
3. J. Blinn and M. Newell. Texture and Reflection in Computer Generated Images. *Communications of the ACM*, 19:542–546, 1976.
4. B. Cabral, M. Olano, and P. Nemec. Reflection Space Image Based Rendering. In *Proceedings SIGGRAPH*, pages 165–170, Los Angeles, California, August 1999.
5. W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light Field Mapping: Efficient Representation and Hardware Rendering of Surface Light Fields. In *Proceedings SIGGRAPH*, pages 447–456, July 2002.
6. P. Debevec and J. Malik. Recovering High Dynamic Range Radiance Maps from Photographs. In *Proceedings SIGGRAPH*, pages 369–378, August 1997.
7. P. Diefenbach and N. Badler. Multi-Pass Pipeline Rendering: Realism For Dynamic Environments . In *1997 ACM Symposium on Interactive 3D Graphics*, April 1997.
8. A. Edmonds. *Angular Momentum in Quantum Mechanics*. Princeton University, Princeton, NJ, 1960.
9. N. Greene. Applications of World Projections. In *Proceedings Graphics Interface*, pages 108–114, May 1986.
10. N. Greene. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics & Applications*, 6(11):21–29, November 1986.
11. W. Heidrich and H.-P. Seidel. View-Independent Environment Maps. In *Eurographics/SIGGRAPH Workshop on Graphics Hardware*, pages 39–45, 1998.
12. W. Heidrich and H.-P. Seidel. Realistic, Hardware-accelerated Shading and Lighting. In *Proceedings SIGGRAPH*, pages 171–178, August 1999.
13. H. W. Jensen. Global Illumination using Photon Maps. In *Seventh Eurographics Rendering Workshop 1996*, pages 21–30, June 1996.
14. J. Kautz and M. McCool. Approximation of Glossy Reflection with Prefiltered Environment Maps. In *Proceedings Graphics Interface*, pages 119–126, May 2000.
15. J. Kautz, P.-P. Sloan, and J. Snyder. Arbitrary BRDF Shading for Low-Frequency Lighting Using Spherical

Harmonics. In *13th Eurographics Workshop on Rendering*, pages 301–308, June 2002.

16. J. Kautz, P.-P. Vázquez, W. Heidrich, and H.-P. Seidel. A Unified Approach to Prefiltered Environment Maps. In *Eleventh Eurographics Workshop on Rendering*, pages 185–196, June 2000.

17. E. Lafortune, S.-C. Foo, K. Torrance, and D. Greenberg. Non-Linear Approximation of Reflectance Functions. In *Proceedings SIGGRAPH*, pages 117–126, August 1997.

18. L. Latta and A. Kolb. Homomorphic Factorization of BRDF-based Lighting Computation. In *Proceedings SIGGRAPH*, pages 509–516, July 2002.

19. J. Lehtinen and J. Kautz. Matrix radiance transfer. In *SIGGRAPH/Eurographics Symposium on Interactive 3D Graphics*, April 2003.

20. D. Lischinski and A. Rappoport. Image-Based Rendering for Non-Diffuse Synthetic Scenes. In *Nineth Eurographics Workshop on Rendering*, pages 301–314. Eurographics, June 1998.

21. D. McAllister, A. Lastra, and W. Heidrich. Efficient Rendering of Spatial Bi-directional Reflectance Distribution Functions. In *Proceedings Graphics Hardware*, pages 79–88, September 2002.

22. G. Miller and R. Hoffman. Illumination and Reflection Maps: Simulated Objects in Simulated and Real Environments. In *SIGGRAPH Course Notes – Advanced Computer Graphics Animation*, July 1984.

23. G. Miller, S. Rubin, and D. Ponceleon. Lazy Decompression of Surface Light Fields for Precomputed Global Illumination. In *Nineth Eurographics Workshop on Rendering*, pages 281–292. Eurographics, June 1998.

24. J. Neider, T. Davis, and M. Woo. *OpenGL - Programming Guide*. Addison-Wesley, 1993.

25. OpenGL Architecture Review Board. *ARB_texture_cube_map*, December 1999. Available from http://www.opengl.org/.

26. OpenGL Architecture Review Board. *EXT_texture_lod*, July 2001. Available from http://www.opengl.org/.

27. B.-T. Phong. Illumination for Computer Generated Pictures. *Communications of the ACM*, 18(6):311–317, June 1975.

28. R. Ramamoorthi and P. Hanrahan. An Efficient Representation for Irradiance Environment Maps. In *Proceedings SIGGRAPH*, pages 497–500, August 2001.

29. R. Ramamoorthi and P. Hanrahan. Frequency Space Environment Map Rendering. In *Proceedings SIGGRAPH*, pages 517–526, July 2002.

30. C. Schlick. An Inexpensive BDRF Model for Physically based Rendering. In *Eurographics '94*, pages 149–162, September 1994.

31. P.-P. Sloan, J. Kautz, and J. Snyder. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *Proceedings SIGGRAPH*, pages 527–536, July 2002.

32. W. Stürzlinger and R. Bastos. Interactive Rendering of Globally Illuminated Glossy Scenes. In *Eighth Eurographics Workshop on Rendering*, pages 93–102. Eurographics, June 1997.

33. L. Williams. Pyramidal Parametrics. In *Proceedings SIGGRAPH*, pages 1–11, July 1983.

34. D. Wood, D. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle. Surface Light Fields for 3D Photography. In *Proceedings SIGGRAPH*, pages 287–296, July 2000.

Figure 6: *Grid of images showing different settings for b and f_λ. Horizontally, we set f_λ to 0.03, 0.09, 0.26, and 0.42. Vertically, b is set to 0.0, 0.8, 1.7, and 2.4.*