CHAPTER 21

Simple Environment Map Filtering Using Ray Cones and Ray Differentials

Tomas Akenine-Möller and Jim Nilsson
NVIDIA

ABSTRACT

We describe simple methods for how to filter environment maps using ray cones and ray differentials in a ray tracing engine.

21.1 INTRODUCTION

Environment maps (EMs) are commonly used in rendering as an inexpensive way of visually representing a scene far away. Another common usage is to let an EM represent the incoming illumination from a surrounding environment and use it to shade geometry [4]. Two common environment-mapping layouts are latitude-longitude maps [2] and cube maps [5].

Rasterization occurs in quads, i.e., 2 × 2 pixels at a time, which means that differentials can be estimated as horizontal and vertical pixel differences. These differentials can be used to compute a level of detail in order to perform a texture lookup using mipmapping. The concept of quads is not available in ray tracing, however. Instead, texture filtering is usually handled using ray differentials [6] or ray cones [1, 3]. These two methods are presented in Chapter 20. For ray differentials, Pharr et al. [7] used a forward differencing approximation to compute ray differentials in texture space for EMs. The major parts of the computations involved are three vector normalizations and six inverse trigonometric function calls.

Since the environment map is assumed to be positioned infinitely far away, environment mapping using rasterization depends on only the reflection vector, i.e., the directional component, and not on the position where the reflection vector was computed. For ray cones and ray differentials, there are also positional components of the ray representations. Similar to rasterization, however, these need not be used, as argued in Figure 21-1. In this chapter, we provide the formulas to compute EM filtering for both ray cones and ray differentials.

© NVIDIA 2019
E. Haines, T. Akenine-Möller (eds.), *Ray Tracing Gems*, https://doi.org/10.1007/978-1-4842-4427-2_21

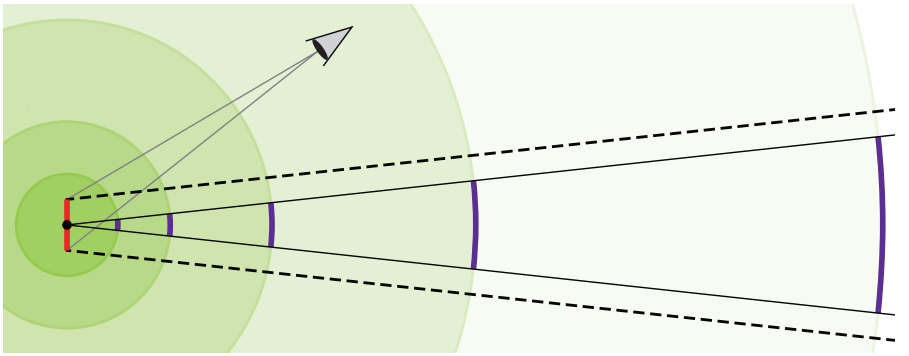

Figure 21-1. *A ray differential or a ray cone consists of a positional (red) and a directional component (dashed lines). The environment map is assumed to be infinitely far away, as usual, and so the coverage from a single point (black dot) with a directional component (solid black lines) is the same, independent of the size of the circle. However, for the dashed lines, a smaller and smaller fraction of the circle is covered as the circle radius grows. At infinity, this fraction will be the same as the purple region. As a consequence, we can use only the directional components when accessing the environment map, even for ray cones and ray differentials.*

21.2 RAY CONES

In this section, we describe how ray cones [1] can be used to access the mip level hierarchy in an environment map. A ray cone can be described by a width, w, and a spread angle γ (see Chapter 20).

For a latitude-longitude environment map, with resolution $2h \times h$, i.e., twice as wide as high, we compute the level of detail, λ, as

$$\lambda = \log_2\left(\frac{\gamma}{\pi / h}\right), \tag{1}$$

where h is the height of the texture and the denominator is set to π/h, which is approximately equal to the number of radians per texel in the map because the texture covers π radians in the vertical direction. The $\log_2$ function is used to map this to the mip hierarchy. The rationale behind this is that if $\gamma = \pi/h$ then we have a perfect match, which results in $\log_2(1) = 0$, i.e., mip level 0 will be accessed. If, for example, γ is eight times as large as π/h, we get $\log_2(8) = 3$, ie., mip level 3 will be accessed.

For a cube map with square sides and resolution $h \times h$ on each face, we use

$$\lambda = \log_2\left(\frac{\gamma}{0.5\pi / h}\right), \tag{2}$$

with similar reasoning as above, except that each face now covers 0.5π radians.

21.3 RAY DIFFERENTIALS

A ray differential [6] is defined as

$$\left\{\frac{\partial O}{\partial x}, \frac{\partial O}{\partial y}, \frac{\partial \hat{\mathbf{d}}}{\partial x}, \frac{\partial \hat{\mathbf{d}}}{\partial y}\right\}, \tag{3}$$

for a ray $R(t) = O + t\hat{\mathbf{d}}$, where O is the ray origin and $\hat{\mathbf{d}}$ is the normalized ray direction (see Chapter 2). We compute the spread angle, shown in Figure 21-2, for a ray differential as

$$\gamma = 2\arctan\left(\frac{1}{2}\left\|\frac{\partial \hat{\mathbf{d}}}{\partial x} + \frac{\partial \hat{\mathbf{d}}}{\partial y}\right\|\right). \tag{4}$$

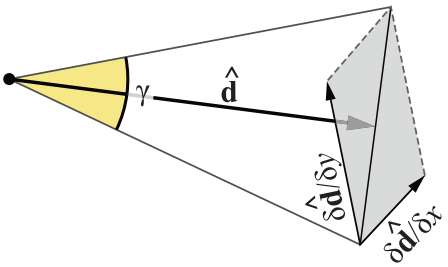

Figure 21-2. *When disregarding the differentials of the ray origin, the spread angle γ can be computed using the normalized ray direction $\hat{\boldsymbol{d}}$ and its differentials.*

This γ can then be used in Equations 1 and 2 to compute the level of detail for use with ray differentials.

Note that our simple methods do not provide any anisotropy nor do they take the possible distortion of the mapping into account.

21.4 RESULTS

As a result of our work on textured reflections, we got used to having filtered textures in reflections. However, our first implementation did not handle environment maps, and as a consequence of that, reflections of the environment map always aliased in our tests. This chapter provides one solution. Results are shown in Figure 21-3.

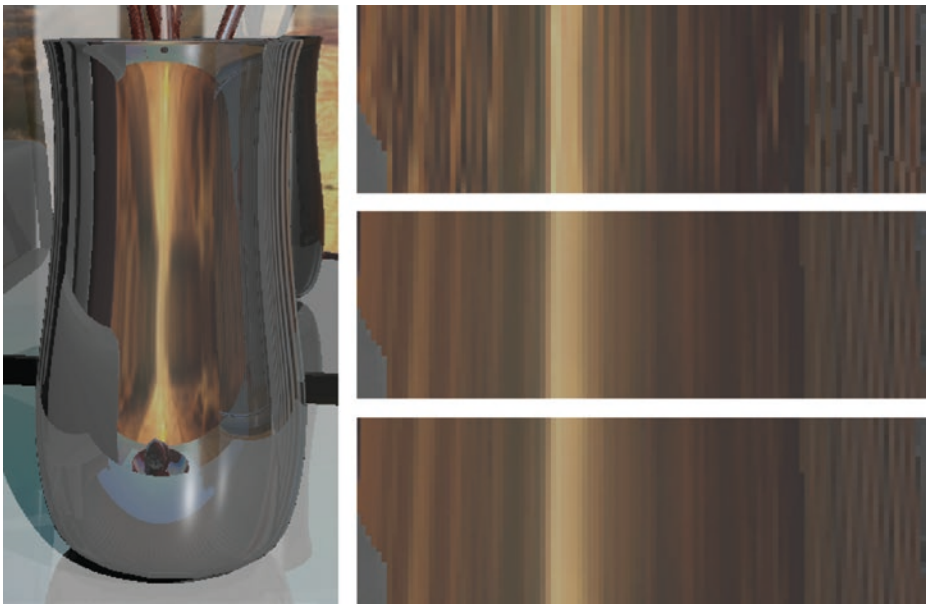

Figure 21-3. *A vase with exaggerated reflections from an environment map (mostly brown area) using different methods. The closeups show, from top to bottom, use of mip level 0, ray cones, and ray differentials. Note that the two latter images are similar, which is expected since they filter the environment map. During animation, severe aliasing occurs in the image on the top right, while the other two are temporally stable.*

REFERENCES

[1] Amanatides, J. Ray Tracing with Cones. *Computer Graphics (SIGGRAPH) 18*, 3 (1984), 129–135.

[2] Blinn, J. F., and Newell, M. E. Texture and Reflection in Computer Generated Images. *Communications of the ACM 19*, 10 (1976), 542–547.

[3] Christensen, P., Fong, J., Shade, J., Wooten, W., Schubert, B., Kensler, A., Friedman, S., Kilpatrick, C., Ramshaw, C., Bannister, M., Rayner, B., Brouillat, J., and Liani, M. RenderMan: An Advanced Path-Tracing Architecture for Movie Rendering. *ACM Transactions on Graphics 37*, 3 (2018), 30:1–30:21.

[4] Debevec, P. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-Based Graphics with Global Illumination and High Dynamic Range Photography. In *Proceedings of SIGGRAPH* (1998), pp. 189–198.

[5] Greene, N. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications 6*, 11 (1986), 21–29.

[6] Igehy, H. Tracing Ray Differentials. In *Proceedings of SIGGRAPH* (1999), pp. 179–186.

[7] Pharr, M., Jakob, W., and Humphreys, G. *Physically Based Rendering: From Theory to Implementation*, third ed. Morgan Kaufmann, 2016.

Open Access This chapter is licensed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License (http://creativecommons.org/licenses/by-nc-nd/4.0/), which permits any noncommercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if you modified the licensed material. You do not have permission under this license to share adapted material derived from this chapter or parts of it.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.