

In 1992, Pat Hanrahan and Don Mitchell introduced³ a new algorithm that allowed an exact computation of the reflected illumination from curved mirror surfaces onto other surfaces (see Figure 1). That superb work was based on Fermat's principle, which says that this computation is equivalent to finding extremal paths from the light source to the visible surface via the mirrors. Once pathways were found, irradiance was computed from the Gaussian curvature of the geometrical wavefront. Another approach is bi-directional ray tracing⁴, but it has as a major drawback: the inherent high computational cost of ray tracing.

In this sketch, I present a high-speed scan-line based algorithm to achieve the same effect. The key ideas on which the whole work is based are:

- There is no light-reflecting mirror that is not "seen" by the light sources.
- We can think of the whole process as if each curved reflector that receives light acts as a secondary light, that is, transforms itself into a non-isotropic extended light source.
- We can perform Phong-like interpolation, within a certain error, of any quantity that can be pointwise obtained, such as the intensity (Gouraud Shading), the surface normal (Phong Shading) or even its curvature.

Armed with these three ideas, we can introduce our scan-line based solution. As we build the shadow maps, we store the reflected illumination info within the mirror (see below), and later, when performing the final rendering step, we treat it exactly as any other light source. To do so, we only need a data structure to hold the illumination information in an efficient and accurate way. But that structure was already developed. It's a modification of the Environment Map¹ called Z-Buffered Environment Map⁵. It is very similar to the Light Buffers², where, for each pixel in the map, a complete sorted list of all the surfaces seen through it is stored (Figure 2). However, instead of storing the surface ID, we store its distance from the map center, its material ID, and the light intensity accumulated on it. Given a reflection ray in the map coordinate system (given by the position on the reflector, W , and the reflection direction, R), we only need to search in the line given by the intersection of the Env. Map and the plane determined by the Center of Projection, W and R ; checking only the lists stored in the pixels that are between W and R , in order to find the closest to the desired intersection point.

So, the first step of our algorithm proceeds as follows (see Figure 3): Firstly, compute the Z-buffered Env. Map for each reflector. Then, when computing the shadow buffer for each light source, scan-line the mirrors as usual, but linearly interpolating the curvatures (stored with their respective vertices) as we use to do with the color or the normal. For each point of the surface rendered, calculate the light ray to it from the source and reflect it. With the Z-Buffered Env. Map, compute the intersection of this new ray with the environment, leaving the intensity information (computed exactly as described in Hanrahan, P & Mitchell) in the corresponding entry of the map.

During the final rendering, we must transform any point to be shaded to all the light sources' spaces and to all the mirrored objects own spaces, too. With the second ones, it's just a matter of taking the illumination information from the corresponding entry from the right pixel and computing the illumination (see Figure 4). If the object being rendered is the mirror itself, I use the indirect illumination stored in the first entry, along with the position (and coordinates of the pixel) to get the direct component, treating the surface as a pure lambertian one.

If we take into account that the original images were rendered in an eight-processor Silicon Graphics Iris in about 10 hours and that a small image (200*200) only took 10 minutes in a 486DLC PC, we would see that scan-line algorithms are just as capable of generating sophisticated lighting effects as ray tracing, at noticeably lower computational costs.

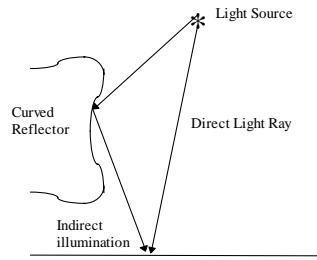


FIGURE 1 Direct and Indirect illumination to be computed at a surface.

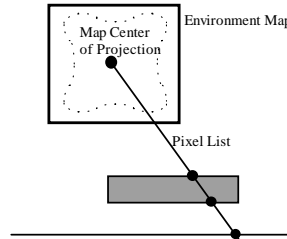


FIGURE 2 The Z-Buffered Environment Map geometry: A pixel list is shown with the three intersection points it contains marked as black dots.

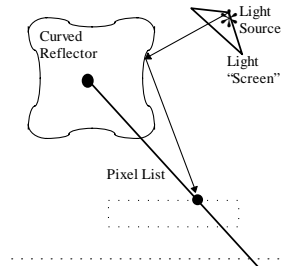


FIGURE 3 Storing the reflected illumination into the Environment Map. When computing the shadow buffer, each light ray that reaches the mirror surface is reflected and stored in its respective intersected list item.

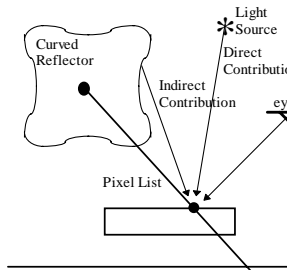


FIGURE 4 The final illumination step: the shading of the point seen by the observer is computed both from the light sources and from the curved reflector.

Gustavo A. Patow
LIFIA - UNLP - ARGENTINA
La Plata
Buenos Aires 1900 ARGENTINA
dagush@sol.info.unlp.edu.ar

REFERENCES

- 1 Blinn, J. F. & Newell, M. E. Texture and Reflection in Computer Generated Images, Comm. ACM, 19(10), October 1976, pp. 542-547.
- 2 Haines, E.A., & Greenberg, D.P. The Light Buffer: A Shadow-Testing Accelerator, CG & A, 6(9), September 1986, pp. 6-16.
- 3 Hanrahan, P & Mitchell, D. Illumination from curved reflectors, Computer Graphics (Proc. SIGGRAPH 92), 26(2), July 1992, pp. 283-291.
- 4 Heckbert, P. Adaptive Textures for Bidirectional Ray Tracing, Computer Graphics (Proc. SIGGRAPH 90), 20(4), August 1990, pp. 315-321.
- 5 Patow, G. A. Accurate Reflections Through Z-Buffered Environment Maps, (Proc. XV Int. Conf. of the Chilean Computer Science Society) 1995, pp. 385-392.