Fuzzy reflections rendering

Anastassiya S. Kulikova
Moscow State University, Faculty of Computational Mathematics
Kirill A. Dmitriev
Keldysh Institute of Applied Mathametics RAS
Moscow, Russia

Abstract

The method proposed in the article adds the possibility of rendering fuzzy reflections to the existing ray tracing systems. It is based on the idea of special blurring. Depending on the roughness of the reflecting surface, the specular component of the image is blurred. Even such special filtering does not require much time, besides it can be optimized to concrete processor architecture. Even implementation of the method in C++ (without any assembly code) results only in 2 % decrease in rendering speed.

Keywords: *Ray Tracing, Fuzzy reflections, Rendering.*

1. INTRODUCTION

Fuzzy (or glossy) reflections introduce greater realism to rendered images. In real life we can find a lot of materials, which produce such reflections. High computational complexity does not allow wide usage of such materials in computer graphics applications. We've found the following methods are currently used for this interesting effect simulation:

Plugin to 3D Studio Max Raygun [3] allows rendering of glossy materials by utilizing ray jittering in every point where backward ray hits the surface. This method gives physically precise results, but is very computationally expensive, since at least 16 samples per pixel are required to receive noiseless image appearance.

Frank Suykens et al. [5] suggest using differential footprint of the ray to filter texture of reflected object. In this way it is possible to calculate accurate fuzzy reflection of textured surface. Though, approach does not allow to receive glossy reflection of non-textured surfaces, border between two objects (even textured ones) also cannot be rendered correctly.

James Arvo in his PhD thesis [6] investigated analytical method for fuzzy reflections simulation. The algorithm seems to work efficiently for simple scenes (e.g. flat mosaic plate is reflected is flat mirror). Method becomes computationally expensive if number of reflected polygons is large since it supposes integral evaluation along the border of each reflected polygon.

It is worth to mention the draft versions of fuzzy reflection implementation in other plugin to 3D Studio Max called InSight. Algorithm acts close to bump mapping approach. That is, in the point of backward ray hit to surface, normal to surface is perturbed in some random way with the variance of normal distribution depending on surface roughness. Adaptive super sampling based on color criteria allows reducing noise, since typically forces several rays to be fired through single pixel. The advantage of approach is that it gives only about 2 times deceleration in worst case, which is less compared to above-mentioned algorithms. Its big disadvantage is that it produces very grainy appearance of the surface, which is not physically plausible in most cases.

2. METHOD DESCRIPTION

The approach suggested also treats the surface as a set of random micro-facets, assuming that their size is much less than pixel size, thus no *granularity* is visible. Under some conditions this approach is physically accurate; in other cases it is expected to give a good approximation sufficient for a photo-realistic appearance.

We implemented the method via two-pass rendering. The 1st pass is the usual backward ray tracing assuming *specular* reflection. Product of this pass is image, where for every pixel the following information is known:

1. Physical luminance L_0 calculated for perfect specular reflection (because only "specular" component will be subsequently blurred).
2. 3D coordinates of intersection point *a* in the glossy surface (Figure 1).
3. 3D coordinates of the "end of ray" b, which is a hit point of ideally reflected ray to the scene surface (Figure 1).

In case of the so-called "subsampling" mode, when some pixels are not traced but interpolated, we calculate all the above values with bi-linear interpolation.

In the second pass, the above image is "filtered", which means that for those pixels whose intersection point belongs to the glossy surface, the following function is evaluated:

$$L(x,y) = \frac{\sum'_{y',x'} L_0(x',y')\, f(x,x',y,y')}{\sum'_{y',x'} f(x,x',y,y')} \quad (1)$$

where $L_0(x,y)$ is the luminance of the original image at pixel (x,y), and $L(x,y)$ is the resulting luminance which represents fuzzy reflections. Here $f(x, x', y, y')$ are weight coefficients depending on the material properties and on the ϑ angle. As in our renderer the reflective properties of material were characterized by the shininess and shininess strength, the following formula was suggested:

$$f = 2 * \text{ShinStrength} * \text{pow}(\cos(\vartheta), \text{Shininess}) \quad (2)$$

In renderers where specular characteristics of material are determined by Phong coefficient (glossiness), the following formula is suggested:

$$f = \mathrm{pow}\ (\cos(\vartheta),\ p) \quad (3)$$

Where p is Phong coefficient and $\vartheta(x,x',y,y')$ is the angle between direction from intersection point to the end of "its" ray (=specular ray) and ray fired from this point to the end of ray for ***neighbour*** pixel (x',y') (see Fig.1).

From the Figure 1 one can calculate that

$$\cos\vartheta(x,x',y,y') = \frac{\left(\boldsymbol{b}(x',y')-\boldsymbol{a}(x,y), \boldsymbol{b}(x,y)-\boldsymbol{a}(x,y)\right)}{\left|\boldsymbol{b}(x',y')-\boldsymbol{a}(x,y)\right| \cdot \left|\boldsymbol{b}(x',y')-\boldsymbol{a}(x,y)\right|}$$

We assume that the reflecting surface is glossy, so that nearly all reflected energy contains in cone $\vartheta \le \Theta < 10°$. In this case contribution of far pixels is negligible, and we can confine the sum in (1) to the neighbourhood of pixel (x, y). The latter comprises pixels (x', y') such that the rays fired to them from camera deviate from ray fired to the central pixel (x, y) be angle less than:

$$\alpha = \Theta/(1 + s/r) \quad (4)$$

where s is the distance from camera to intersection point on glossy surface (point $\boldsymbol{a}$) and r is the distance from the latter to the end of ray (point $\boldsymbol{b}$). From the angular size we can easily estimate the radius in pixels:

$$\rho = \alpha\times(\text{image size})/(\text{view angle}) \quad (5)$$

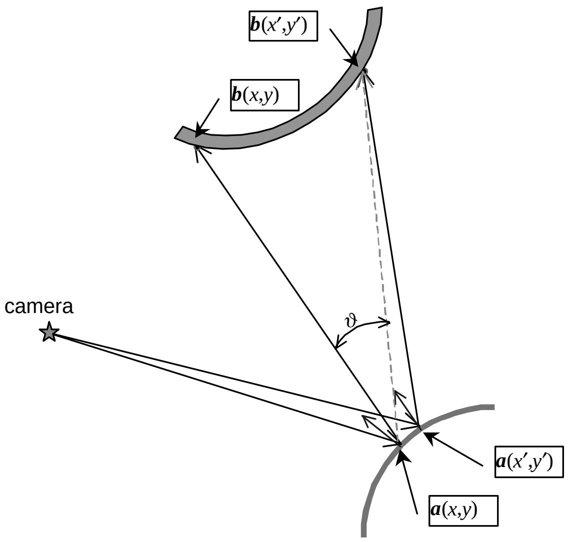

Figure 1: The thick curve at right bottom corner is glossy reflector; thick curve at the top is the object reflected in it; small arrows on reflector show local normals and solid long arrows show specularly reflected rays which were traced in the 1st pass; the dashed arrow is the ray considered for fuzzy reflection simulation and ϑ is the angle between it and direction of ideal specular reflection.

Thus, we obtain the following formula for determining the blurred color:

$$L(x,y) = \frac{\sum_{|y'-y|<d,|x'-x|<d} L_0(x',y')\, f(x,x',y,y')}{\sum_{|y'-y|<d,|x'-x|<d} f(x,x',y,y')} \quad (6)$$

(or equally we can use round instead of rectangular area)

The filter function is evaluated at the centres of pixels. It would be better to evaluate $f(x,x',y,y')$ as *average* over pixel, but that is too expensive. (3)

3. BASIC OPTIMIZATIONS

When implemented as described above the algorithm considerably decreases rendering speed. The following optimizations were made. They do not lead to drawbacks in quality, but allow faster rendering.

The filtering (1) may be expensive in case of large filter size. The following means can be used to accelerate that:

a) Force restriction on the filter size.

b) Adaptive interpolation is possible. We can "blur" the image ignoring antialiasing. While blurring we do not split pixel in subpixels.

c) The weight coefficients can be tablulated on a regular mesh. It is done on the first pass of renderer. Then the calculation of arccos is obviated, as well as e.g. raising to power γ in Phong model.

d) While testing the preliminary implementation it was observed that the coefficients calculated in the above-mentioned way are similar to Gauss kernel. Therefore it is possible to use Gauss convolution. The subtle differences in images are usually not seen by pure eye.

e) The filter size can be calculated at each fourth pixel, because it is reasonable to assume that the angles do not change considerably between adjacent pixels.

3.1 RESULTS

Lower we give images for surface roughness equal to 0%, 10% and 50% correspondingly. One can clearly see that roughness level has strong effect on image appearance.

Figure 2: Sample scene, no fuzzy reflections

Figure 3: Sample scene, surface roughness 10%

Figure 4: Sample scene, surface roughness 50%

4. CONCLUSION

The advantages of the method are its speed, physical accuracy under certain conditions. The algorithm can be added to any renderer as a second pass. The following limitations are inherent in the method:

Reflections after reflection by the first encount ered glossy surface are not handled. That is the method can not accurate handle the case when e.g. glossy surface reflects ideal mirror which in turn reflects something. This is because the method assumes that rays form intersection point till ray end are straight lines.

Similarly, if one fuzzy object is reflected in other fuzzy object, the fuzziness of furthest object (first one) will be ignored. This assumption looks reasonable since fuzziness of primary reflection should hide sharpness of secondary one a nyway.

Then, the method is inaccurate when the reflected object has BRDF with narrow lobe. This is because the luminance of object point ***b*** in filtering is assumed to be the same when we observe it from point ***a*** and ***a′*** that is at different directions, see Fi g.1. But usually the angle between those directions is small, and for BRDF smooth enough that will not affect image quality.

5. REFERENCES

[1] (Internet link)

http://www.cs.byu.edu/courses/cs455/phase4.raytracing.html

[2] (Internet link)

http://math.ucsd.edu/~sbuss/Math155/18_Assign4B.html

[3] (Internet link)

http://www.righthemisphere.com/raygun30/product_information.htm

[4] (Internet link)

http://www.geisswerks.com/ryan/PROG-STIL

[5] Frank Suykens, Yves Willems. Path diffe rentials and Applications. Proceedings of the 12 th Eurographics Workshop on Rendering, 2001 pp. 254 -265

[6] James Arvo. “Analytic Methods for Simulated Light Transport", Ph.D. Thesis, Yale University, December, 1995.

About authors

Anastassiya S. Koulikova is the student of Computational Mathematics Faculty.
E-mail: **akulikova@hotbox.ru**

Kirill A. Dmitriev is the PhD student in Keldysh Institute of Applied Mathematics RAS.

E-mail: **kadmitr@gin.keldysh.ru**

kirill@mpi-sb.mpg.de