ISSN 0361-7688, Programming and Computer Software, 2008, Vol. 34, No. 5, pp. 271–278. © Pleiades Publishing, Ltd., 2008.
Original Russian Text © V.A. Debelov, G.G. Smirnova, L.F. Vasilyeva, 2008, published in Programmirovanie, 2008, Vol. 34, No. 5.

An Extension of the Light Meshes Method for Three-Dimensional Scenes with Semitransparent Surfaces

V. A. Debelov, G. G. Smirnova, and L. F. Vasilyeva

Institute of Computational Mathematics and Mathematical Geophysics, Siberian Branch, Russian Academy of Sciences, pr. Akademika Lavrent'eva 6, Novosibirsk, 600090 Russia

Novosibirsk State University, ul. Pirogova 2, Novosibirsk, 630090 Russia

e-mail: debelov@oapmg.sscc.ru; gasmi@gorodok.net; ludvas@oapmg.sscc.ru

Received November 24, 2006

Abstract—The light meshes method (a modification of the Whitted backwards recursive ray tracing) was justified and studied for scenes consisting of opaque surfaces. Its main difference from the basic method is that the rendered image may include soft shadows (i.e., point sources are simulated by area sources). This study makes a further step in the development of this method: it is extended for rendering scenes containing semitransparent surfaces. A few computational schemes are considered and, for each of them, the difference between the image calculated by the standard scheme and obtained by the application of the light meshes method is shown.

DOI: 10.1134/S0361768808050046

1. INTRODUCTION

Among all the methods for producing realistic images, the classic light-backwards recursive ray tracing algorithm (RRTA) proposed by Whitted [1] in 1980 and its modifications are most commonly used. This algorithm affords a suitable cost (calculation time) to quality (realism) ratio for the resulting images. A scene is supposed to consist of *transparent* and *opaque* surfaces illuminated by several point and directional light sources. Theoretically, the RRTA can produce only sharp shadows because only point sources are used for scene illumination. In real scenes, the sources normally have quite appreciable dimensions resulting in softened (blurred) shadows of objects. Moreover, unrealistically sharp shadows are often identified by an observer as independent (phantom) objects in the scene. Hence, the desire to make the rendered images more realistic by introducing soft shadows. Many studies were aimed at the simulation of soft shadows by modifying the RRTA. This means that one needed to blur sharp shadows in the framework of the RRTA so that the computational effort be not very high. The light meshes method (LMM), which was proposed in [2] for scenes with *opaque* surfaces, was developed for simulating soft shadows while rendering scenes with point light sources. In this paper, we justify the applicability of this method for scenes including *transparent* and *semitransparent* surfaces.

A *light mesh* is a uniform mesh in the space of a scene with points collecting generalized data concerning the illumination of the corresponding point in the scene space (rather than a point of an object) by light sources that are visible from it. The most important thing is that the visibility of light sources from the mesh points was weighted by the intensity of sources and divided by the distance function. The illumination at the displayed object points *was multiplied by the average illumination* at the nearest mesh points [2]. Without going into details, we note that method made it possible to produce sufficiently realistic scene images with fairly realistic soft shadows. More details on soft shadows can be found in [3]. It turned out that, as the complexity of the problem increases (as the number of primitives, number of sources, and image resolution increase), the LMM becomes faster than the RRTA. This feature seemed very attractive because it could be used for test calculations in debugging the geometry and illumination in a scene. However, the relationship between images produced by the LMM and the RRTA must be somehow explained to the user.

Another approach in which source visibility masks are calculated and stored was proposed in [4, 5]. In the framework of that approach, the modified light meshes method can be compared with the RRTA due to the following reasons:

(a) If an object point is located far away from the boundaries of sharp shadows, both images (produced by the LMM and RRTA) coincide.

(b) All the sharp shadows in the image produced by the RRTA look blurred (soft) in the image produced by the LMM.

(c) As the mesh step decreases, the images produced using the LMM converge to the image produced using the RRTA.

Therefore, it is clear that the LMM is an approximation of the RRTA.

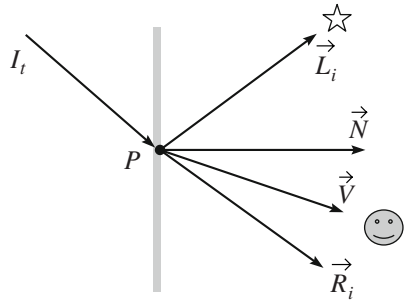

Fig. 1. Calculation using the local illumination model with filtering transparency.

In Section 2, we briefly describe various image calculation techniques (including the RRTA). These descriptions are used in Section 3 to describe the LMM for scenes with opaque surfaces. In Section 4, we consider the specific features of RRTA calculations for scenes with semitransparent surfaces.In Section 5, we formulate the LMM for such scenes. In Section 6, we analyze numerical results and rendered images to clearly demonstrate the differences and specific features of the LMM.

2. ANALYSIS OF THE RAY TRACING FORMULA

Let a scene be illuminated by nL point sources with the intensities $I_1, \ldots, I_{nL}$, located at the points $LP_1, \ldots, LP_{nL}$, and let it consist of a set of surfaces each point of which is characterized by the properties of reflection and semitransparency defined by the coefficients k_d for diffuse reflection, k_s for specular reflection, and k_t for transparency.

All the subsequent formulations may be considered only for the achromatic case. Following [6] accurate to the notation, we write the main calculation formula of the popular OpenGL and DirectX libraries, which is called the local point illumination; this formula ignores source visibility:

$$I_{out} = I_a k_a O_d + \sum_{i=1}^{nL} f_{att,i} I_i [k_d O_d (\vec{N}, \vec{L}_i) + k_s O_s (\vec{R}_i, \vec{V})^{nShiny}] + k_t O_t I_t.$$

Here, (,) is the scalar product, and all the values are taken at the given object point P; I_{out} is the resulting intensity; I_a and k_a are the intensity and the reflection coefficient of the ambient light, respectively; O_d is the diffuse color of the surface; $f_{att,i}$ is the coefficient of attenuation with distance for the ith source; O_s is the color of the specular component; $nShiny$ is the index of surface specularity; $\vec{N}$, $\vec{L}_i$, $\vec{R}_i$, and $\vec{V}$ are the normal, the direction to the ith light source, its corresponding reflected vector, and the direction to the camera, respectively; O_t is the color of the semitransparent surface; and I_t is the intensity passed through this surface. Figure 1 demonstrates the directions of the vectors used in formula (1) to calculate the intensity at the point P of a semitransparent surface. Here, the light refraction can be taken into account as in the classical algorithm [1] or ignored as in OpenGL or DirectX. The difference is caused by the direction of incoming intensity I_t. It is well known that the local illumination model cannot calculate shadows and ignores specular reflections; therefore, we are interested in an algorithm in which the calculation formula includes the visibility function $V(i, P)$ between the point P and the ith source. First of all, this includes the calculation by the so-called *local illumination model with shadows* or *ray casting*:

$$I_{out} = I_a k_a O_d + \sum_{i=1}^{nL} V(i, P) \cdot f_{att,i} I_i [k_d O_d (\vec{N}, \vec{L}_i) + k_s O_s (\vec{R}_i, \vec{V})^{nShiny}] + k_t O_t I_t. \quad (1)$$

We will not describe this very popular model. The reader can make sure that this computational scheme can also be modified using the LMM to blur shadows.

Let us write the RRTA formula (according to [6]):

$$I_{out} = I_a k_a O_d + \sum_{i=1}^{nL} V(i, P) \times f_{att,i} I_i [k_d O_d (\vec{N}, \vec{L}_i) + k_s O_s (\vec{R}_i, \vec{V})^{nShiny}] + k_s I_r + k_t I_t; \quad (2)$$

here, I_r is the intensity incoming from the direction of the reflected ray $\vec{R}$. If the point P is located on an opaque surface, the visibility function $V(i, P)$ can take the following values: 1 if the ith source is visible from the point P and 0 otherwise. For the case when the scene contains semitransparent surfaces, Whitted proposed the following method for calculating $V(i, P)$:

(1) First, $V(i, P)$ is set to 1.

(2) Opaque surfaces are assigned $k_t = 0$;

(3) The surfaces intersected by the segment connecting the point P with the ith source are determined.

(4) $V(i, P)$ is multiplied by the coefficients k_t of the intersected surfaces.

Here, it makes sense to note that the proposed technique makes the resulting images less realistic; however, this algorithm is widely used, and we decided to apply the LMM to this formulation. Since the three terms $I_a k_a O_d + k_s I_r + k_t I_t$ in the LMM and RRTA are treated and calculated similarly, we consider only the second term rewritten in the form

$$U_{RT}(P) = \sum_{i=1}^{nL} V(i, P)\Omega(i, P), \quad (3)$$

where $U_{RT}(P)$ is the unknown intensity of the object point P calculated by the RRTA, nL is the number of sources, $V(i, P)$ is the visibility of the ith source from the point P, and $\Omega(i, P)$ is the part of formula (2) that accounts for all reflecting properties of the surface at the point P and parameters of the ith source.

3. THE LIGHT MESHES METHOD FOR OPAQUE SURFACES

The known approaches to the simulation of soft shadows in the RRTA are based on specific geometrical considerations in the object space or in the space of the image; these are the method of shadow volumes [7], the method of shadow maps [8], etc. The main idea of the LMM is to consider and modify computations of the visibility function $V(i, P)$ of the ith source from the rendered point P. Therefore, we disregard formulas for calculating the local illumination of the point and concentrate on the calculation of $V(i, P)$. The scene space is divided into domains, each with a constant mask of source visibility $\{V(1, P), \ldots, V(nL, P)\}$. A change in even one of these values (for example, the jth value) indicates that there is a boundary of a sharp shadow for the jth source at this point. It only remains to "blur" these boundaries between domains. Thus, we do not have to consider the total geometry of the scene: it is sufficient to make a local estimate in the neighborhood of the rendered object point.

A *light mesh* (LM) is a uniform mesh of points in the space of the scene; h is the mesh resolution. The LMM has another important parameter kI determining the interpolation sphere with the radius $r = h \cdot kI$ that will be defined below.

Preprocessing step. At each mesh point x, we define a mesh value—the mask of visibility of light sources $N_x = \{V(i, x)\}_{i=1}^{nL}$. In an implementation of the LMM, it is more rational to determine the mesh values "on demand" (i.e., at the moment when these values are required for calculations). It is seen from the statistics presented in [4] that only about 1 to 10% of all LM points in the scene space are used in the calculations.

Let us denote $N_x(i) = V(i, x)$. Then, the intensity $U_{LM}(P)$ of the point P can be calculated using the LMM approach by the following algorithm Ψ (details can be found in [3]):

(1) Construct the interpolation set $DLM(P)$ of mesh points that first includes all the light points located within a sphere centered at P: $\|x_k - P\| \le r$.

(2) Remove x_k that are located on the invisible (reverse) side of an opaque surface: $(\vec{N}, x_k - P) \le 0$.

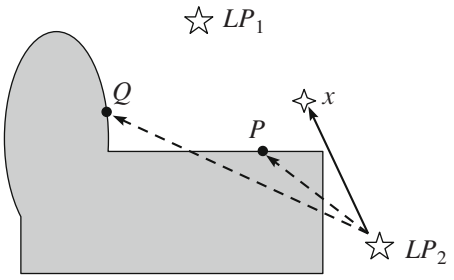

Fig. 2. Calculation by formula (4).

Here, (,) is the scalar product and $\vec{N}$ is the normal at the point P.

(3) Remove x_k that are invisible from the point P or, to put it differently, are occluded by other (opaque) surfaces of the scene. Thus, all the remaining points satisfy the visibility condition $V(x_k, P) = 1$.

(4) The resulting set $DLM(P) = \{x_1, \ldots, x_m\}$ (consisting of light mesh points) will be used for calculating the intensity of the object point P.

(5) If $DLM(P) \ne \varnothing$ (i.e., if $m > 0$), we calculate the quantity

$$U_{LM}(P) = \frac{1}{m}\sum_{k=1}^{m}\left[\sum_{i:\ V(i, x_k)=1} [\xi(i, P) \cdot \Omega(i, P)]\right], \quad (4)$$

where the function

$$\xi(i, P) = \begin{cases} 1, & \text{if } ((LP_i - P), \vec{n}(P)) > 0 \\ 0, & \text{otherwise} \end{cases}$$

indicates whether the ith source can yield a shadow in the neighborhood of the point P; i.e., if the source is located on the outer side, then $\xi(i, P) = 1$; otherwise, if it is located on the back side, $\xi(i, P) = 0$. In the latter case, this source is ignored in the calculation. Figure 2 shows a scene with two sources: $\xi(1, P) = 1$ and $\xi(2, P) = 0$. The mesh point x is visible from both sources. The second source cannot take part in the generation of a shadow because it illuminates x "from behind" and, therefore, should not be used in the calculation of its illumination. If we take into account this source in formula (4), the situation called the light leak would occur. For the object point Q with $\xi(1, Q) = 1$ and $\xi(2, Q) = 1$, the second source will be used in the calculation of illumination to form half-shadow even though Q is not visible from that source.

(6) If $DLM(P) = \varnothing$, we apply formula (3) (i.e., the standard RRTA algorithm). Note that, in experiments even with very complex scenes (hundreds of thousands of triangles) not more than a dozen such cases occurred in images with the resolution of 1000×1000 pixels and higher.

Figure 3 shows a scene rendered by both algorithms. The scene is illuminated by a single source and consists

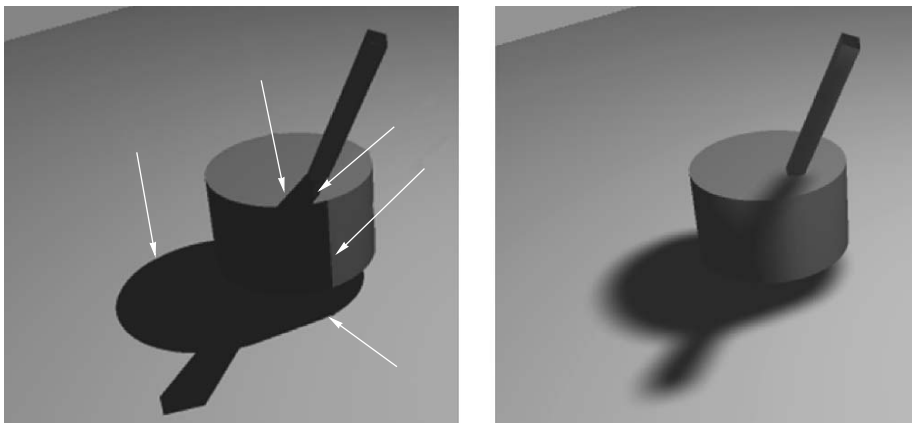

Fig. 3. Images rendered using the RRTA (left) and LMM (right) algorithms (ray casting technique).

of a tabletop (box) and a solid cylinder hanging over it with a bar (rectangular parallelepiped) partially sealed into the cylinder. All the surfaces are opaque. Soft shadows appear due to interpolation (averaging over the semispherical neighborhood) of the visibility of the point P from the source according to the algorithm Ψ (see [5] for details). In the left part of this figure the visible parts of sharp shadow edges are marked by arrows; the same edges shown in the right part of the figure are blurred. Pay attention to the sharp shadow from the point source at the side of the cylinder in the left part of the figure; in the right figure, it changes into a penumbra. It is seen from the shadow configuration under the cylinder that it hangs over the tabletop. This figure also demonstrates another feature of the LMM: sharp boundaries between faces are not blurred (i.e., the geometry is preserved).

4. SPECIFIC FEATURES OF RAY TRACING FOR SEMITRANSPARENT CONDITIONS

The situation is quite different when the scene includes semitransparent surfaces [1, 6]. Here, the recursive tracing means not a sequence of reflections but a tree; at each node of this tree, a refraction direction is constructed in addition to specular reflection. The coefficient k_t is responsible for attenuating the ray intensity I_t arriving from the "direction of refraction" (see Fig. 1). Calculations performed on the basis of the RGB-model normally use the characteristics of the environment on both sides of the semitransparent surface, which provides a basis for the Snell law application; then, the path for the refracted ray can be built. One can assume that a more accurate intensity value can be obtained by using the technique described above for opaque surfaces (i.e., construct a tree once and calculate the intensity based on spectral considerations). However, this is fundamentally wrong because of the so-called *optical dispersion* when the direction of the refracted ray depends on the light wavelength. We will not consider the issue of how seriously this phenomenon affects the realism (more correctly, physical accuracy). Note that studies on the RRTA usually omit a detailed consideration of how the local illumination is calculated at the object point when the scene includes semitransparent surfaces. The standard RRTA calculates the function $V(i, P)$ by multiplying the transparency coefficients of the surfaces intersected by the segment connecting the ith source with the point P (i.e., refractions are neglected). Let us again point out that the backwards ray tracing is performed taking refractions on semitransparent surfaces into account, but visibility is calculated using the segment connecting the source and the point although a physically accurate approach requires that the path from the source to the object point be traced with account of ray refractions on each of the surfaces.

Now, let us rewrite formula (4) used by the RRTA to find the local illumination of a point in the form

$$U_{RT}(P) = \sum_{i=1}^{nL}\left(\sum_{j=1}^{nJ(i)} v(i, j, P)\cdot I_i \cdot \Omega'(i, j, P)\right), \quad (5)$$

where $nJ(i)$ is the number of paths along which the light ray can travel from the ith source to the point P with account of refractions; the path can be a straight line (if there are no semitransparent objects between the source and the point P) or a polyline (if there are semitransparent objects refracting the light ray). Note that this formula is correct for both cases: with or without refraction. $v(i, j, P)$ is the visibility function of the ith source for the jth path from the point P. If the source is visible, then $v(i, j, P) = 1$; if there is at least one opaque object on the path, then the source is invisible and $v(i, j, P) = 0$; if there are nT semitransparent surfaces on the path j to the source, then $v(i, j, P) = \prod_{p=1}^{nT} t_p$, where t_p are the transparency coefficients of these surfaces. I_i is the intensity of the ith source. This multiplier can be removed from the sum sign because it linearly appears in Ω'. $\Omega'(i, j, P)$ is the illumination function depending on the reflecting properties of the surface material, on

the properties of the light source i with account of attenuation due to distance, and on the path j.

5. THE LIGHT MESHES METHOD AND SEMITRANSPARENT SURFACES

Now, the value at a mesh point is not simply a visibility mask of nL bits (as in Section 3), but an array of nL real numbers. To obtain LMM formulas, we modify formulas (5) similar to how it was done in Section 3.

The *preprocessing step* is more complex than in Section 3 because, at each mesh point, the visibility of a source is not simply 0 or 1. At each mesh point x_l, for the ith source (i ranges from 1 to nL), the value of the visibility function is determined with account of transparency $v(i, x_i)$ by the formula

$$v(i, x_l) = V(i, x_l) + \sum_{k=1}^{nK}\left(\prod_{p=1}^{nP(k)} t_p\right), \tag{6}$$

where $V(i, x_l)$ is the visibility function without account of semitransparency. To determine its value, we shoot rays from the mesh point in the direction of the source. If there are no objects on the path of the ray to the source, then $V(i, x_l) = 1$; otherwise, $V(i, x_l) = 0$. Next, we check whether the rays passing (or refracting) through semitransparent objects fall on mesh points; nK is the number of paths consisting of refracted rays that hit the mesh point x_l; $nP(k)$ is the number of semitransparent surfaces that were penetrated by the kth path; and t_p is the transparency coefficient of the pth surface.

The interpolation set is again chosen using the algorithm Ψ (see Section 3) modified in accordance with a possible semitransparency of scene surfaces.

(1) Construct the interpolation set $DLM(P)$ of mesh points that first includes all the light points located within a sphere centered at P: $\|x_k - P\| \le r$.

(2) Remove x_k that are located on the invisible (reverse) side of an opaque surface: $(\vec{N}, x_k - P) \le 0$.

(3) Remove x_k that are invisible from the point P or, to put it differently, are occluded by *opaque* surfaces of the scene. Thus, all the remaining points satisfy the partial visibility condition $v(x_k, P) > 0$.

(4) The resulting set $DLM(P) = \{x_1, \ldots, x_m\}$ consisting of light mesh points will be used for calculating the intensity at the object point P. At each of these points x_k, the visibility $v(x_k, P) > 0$ of P is known; it is calculated by formula (6).

(5) If $DLM(P) \ne \varnothing$ (i.e., if $m > 0$), calculate the quantity

$$U_{LM} = \frac{1}{m}\sum_{l=1}^{m}\left[v(x_l, P) \times\left(\sum_{i=1}^{nL}\xi(i, P)\cdot v(i, x_l)\cdot I_i\cdot\tilde{\Omega}(i, P)\right)\right], \tag{7}$$

where $\tilde{\Omega}(i, P)$ accounts for the reflecting properties of the surface material, for the intensity, and for the distance from the source i.

(6) If $DLM(P) = \varnothing$, formula (5) is applied (i.e., the standard RRTA algorithm).

Let us analyze formula (7) in more detail. The multiplier $\xi(1, P)$ is used to avoid light leaks similarly to the case of opaque surfaces (see Section 3). However, this cannot prevent all the cases of light leak when the visibility is calculated by formula (6) using ray refractions on media interfaces; we simply made an empirical assumption that the paths in formula (6) are usually directed towards the light source. Thus, the calculations in most cases will be correct. However, in order to completely prevent light or shadow leaks, formula (7) should be replaced by a more complex and computationally expensive algorithm. For example, at each mesh point x_l and for each source, we store all the components of the visibility function $v(i, x_l)$ separately (in the notation of formula (6)) rather than as a single total value:

- direct visibility $V(i, x_l)$;
- for each of the paths $j = 1, \ldots, nK$, the partial visibilities $\alpha_j = \prod_{p=1}^{nP(j)} t_p$;
- for each of the paths $j = 1, \ldots, nK$, the first segment of the path $[x_l, P_j]$; i.e., the point P_j.

Now, in formula (7), we replace the subexpression $\xi(i, P)\cdot v(i, x_i)$ by $V(i, x_i)\cdot\xi(i, P) + \alpha\cdot\xi(P_j, P)$.

6. EXPERIMENTS

The experiments were performed using the same scene as shown in Fig. 2 except that the cylinder surface was assumed to be semitransparent. Figure 4 shows six images produced using different algorithms:

- The first row shows the images produced by the standard RRTA algorithm, and the second row shows the images produced by the LMM.
- The left image in each row was rendered while neglecting refraction on semitransparent surfaces. Such images can normally be obtained using OpenGL or DirectX.
- The center image in each row was rendered taking refraction into account in backwards ray tracing and

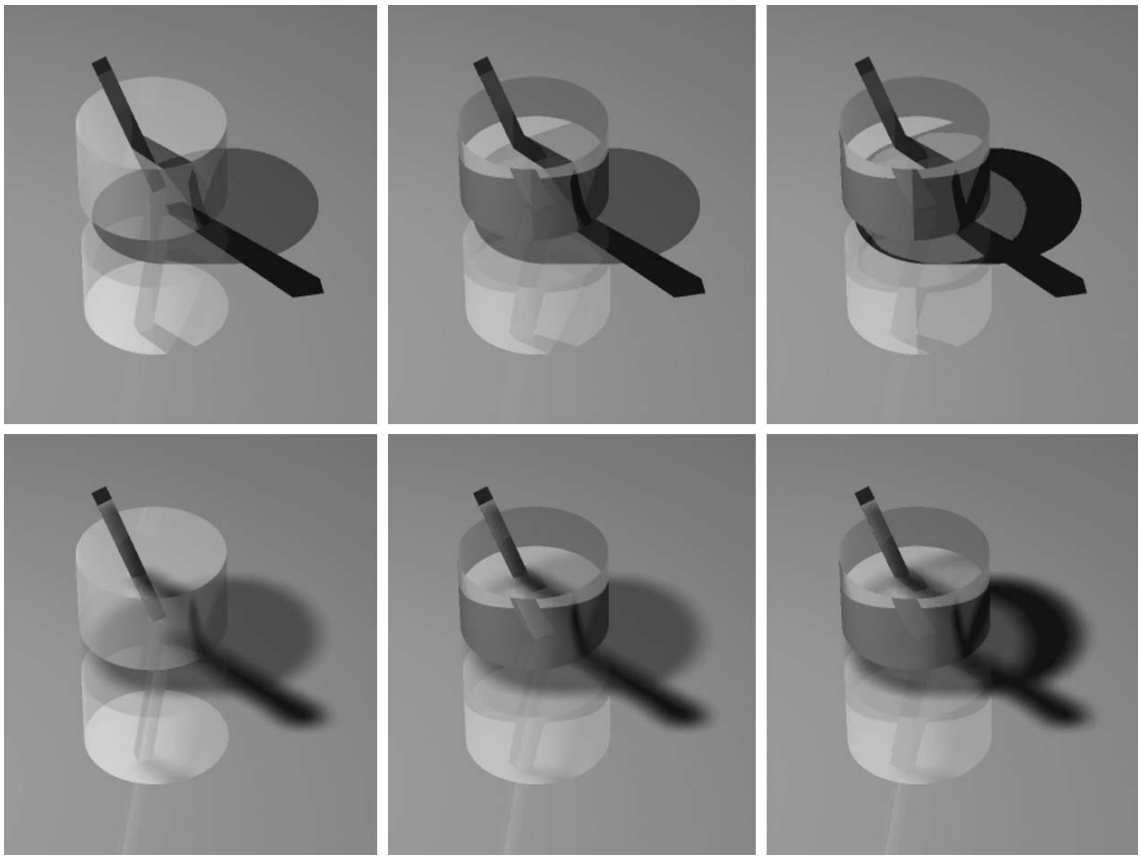

Fig. 4. Different rendering techniques for a scene with a semitransparent surface.

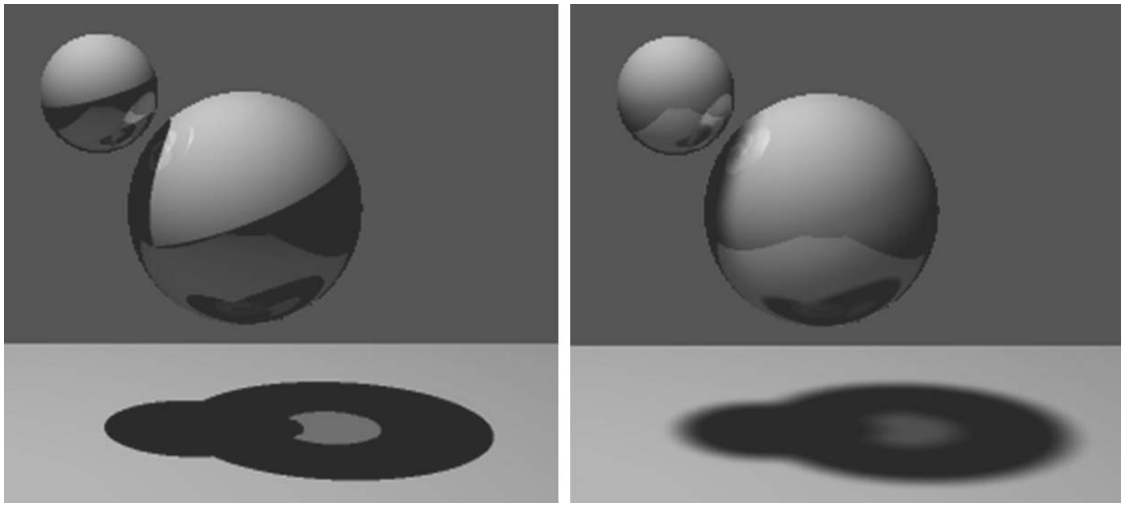

Fig. 5. Semitransparent sphere: RRTA (left) and LMM (right).

neglecting it while determining the visibility of sources. Such images are obtained using the standard RRTA.

• The right image in each row. Refraction was taken into account both in backwards ray tracing and in determining the visibility of sources. Such images are usually obtained using other methods (for example, Monte-Carlo tracing or photon maps). The Whitted model can also be used; however, in this case, for each scene, one has to develop a program adapted to the geometry of this particular scene, which is just what we have done in calculating the right images in each row.

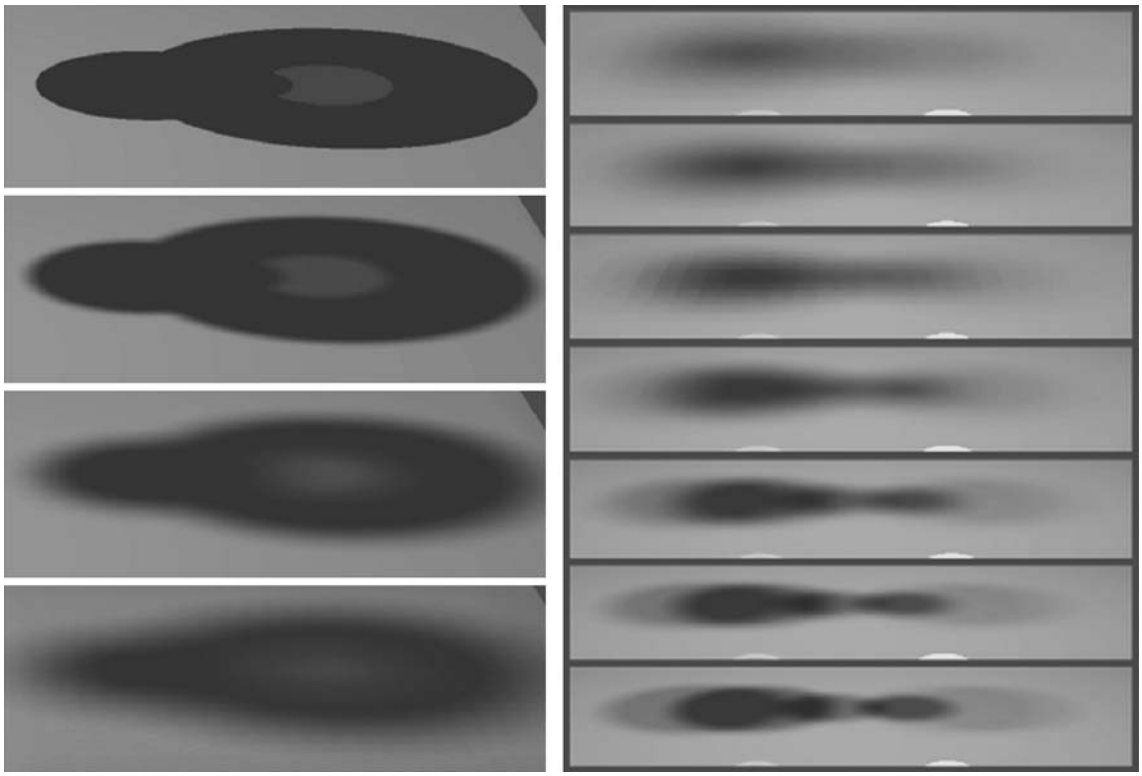

Fig. 6. Dependence of shadow blurring on the light mesh resolution (left) and on the radius of the interpolation hemisphere (right).

In the scene depicted in Fig. 5, the larger sphere is semitransparent. In our calculations, the visibility was determined with account of refraction; therefore, the sphere works as a light-collecting lens. Figure 6 shows (using a series of images) the dependence of the soft shadow on the light mesh size and on the radius of the interpolation hemisphere. The left part of this figure shows (bottom-up) four images of a shadow fragment rendered with a decreasing light mesh size and a constant radius of the interpolation hemisphere. The right part shows the images rendered using a fixed mesh size and increasing (from the bottom to the top) radius.

7. CONCLUSIONS

In this study, we considered various techniques for calculating images with shadows and showed that all the approaches can be modified using the light meshes method. In other words, if there is a computer program that renders images on the basis of ray tracing, the functionality of this program can be extended by introducing the LMM mode. It is obvious that, using this approach, the user can be allowed to control the process of shadow blurring by specifying: (a) the sources that yield soft shadows and the sources that yield sharp shadows; (b) the surfaces casting soft shadows and the surfaces casting sharp shadows; (c) the surfaces on which soft shadows are allowed, etc. The size of the neighborhood (hemisphere) over which visibility is averaged determines the width of the penumbra area, and the light mesh resolution is responsible for grading the intensity in the penumbra area; therefore, by varying the mesh resolution, one can obtain softer or sharper shadows; while varying the radius of the interpolation sphere, one can control the penumbra area. It is clear that this approach would fail for arbitrary scenes. However, the experimental results show that the use of the LMM without refraction of shadow rays yields quite realistic images for scenes with semitransparent surfaces. Blurring sharp shadow boundaries allows the user to simulate non-point sources and, therefore, the LMM can be used together with and in place of the standard ray tracing algorithm.

ACKNOWLEDGMENTS

This study was partially supported by the Russian Foundation for Basic Research, project no. 06-07-89216.

REFERENCES

1. Whitted, T., An Improved Illumination Model for Shaded Display, *Comm. ACM*, 1980, vol. 23, no. 6, pp. 343–349.

2. Debelov, V.A. and Sevastyanov, I.M., A Novel Approach to Simulation of Soft Shadows and Diffuse Color Bleeding in Ray Tracing, *Proc. of the 11th Int. Conf. on Computer Graphics and Machine Vision Graphicon-2001*, Nizhni Novgorod, 2001, pp. 18–24.

3. Debelov, V.A. and Sevastyanov, I.M., Soft Shadows as Interpolation of Visibility, *Future Generation Computer Systems*, 2004, vol. 20, no. 8, pp. 1299–1315.

4. Debelov, V.A. and Vasilyeva, L.F., Approximation of the Solution obtained by a Whitted-like ray tracing algorithm in Synthesis of Realistic Images, *Proc. of the Int. Scientific and Practical Conf. "Svyaz-2004," Seminar "Computational Methods and Solution of Optimization Problems," 2004*, Issyk-Kul', 2004, pp. 64–69.

5. Debelov, V.A., Vasilyeva, L.F., and Novikov, I.E., Improvement of the Light Meshes Method for the Ray Tracing Algorithm: Approximation of Solution, Implementation on a Graphical Processing Unit, *Proc. of the 15th Int. Conf. on Computer Graphics and Its Applications Graphicon-2005*, Novosibirsk, 2005, pp. 355–359.

6. Foley, J. and Van Dam, A., *Computer Graphics: Principles and Practice*, Reading, Mass.: Addison-Wesley, 1990.

7. Crow, F., Shadow Algorithms for Computer Graphics, *Comput. Graphics*, 1977, vol. 11, no. 2, pp. 242–247.

8. Williams, L., Casting Curved Shadows on Curved Surfaces, *Comput. Graphics*, 1978, vol. 10, no. 2, pp. 270–274.