Usage of Color During Visualization of Rays Propagation Simulated with Monte-Carlo Method

Elissey Birukov [1], Irina Ryzhova[1] and Alexey Voloboy [1]

[1] *The Keldysh Institute of the Applied Mathematics of RAS, Miusskaya Sq. 4, Moscow, 125047, Russia*

Abstract

This work describes specifics of rays colorization during visualization of trajectories of such rays obtained by Monte-Carlo simulation method in optical modeling systems. It is described how showing rays visualized with different colors in viewport of a CAD system can improve optical modeling process. Two basic methods of rays visualization are mentioned: directly during simulation process and after it. In more detail there is described application of coloring criteria for simulating different situations and problems with optical systems. In particular, drawing of an entire ray with some color specified by a criterion is mentioned, as well as drawing separate ray segments with different colors (when some criteria are applied separately to each segment). Construction of complicated criteria is described also (when some event occurred with ray or its segment should be repeated several times to satisfy the criterion). Besides that, drawing rays with natural colors is mentioned. Some words are also said about methods of commands and data synchronization for implementation of optical modeling system as a plug-in for existing CAD software, in particular, CATIA CAD.

Keywords

Raytracing, Monte-Carlo method, rays visualization, photorealistic images, optical modeling, lighting simulation

1. Introduction

Visualization of the rays calculated using raytracing with Monte Carlo method provides vivid demonstration of the results of calculations. Also it allows to see how these results will change when the model parameters are changed. Other ways of representing the results of computer optical modeling are mainly graphics, tables, or images representing distribution of such output light characteristics as luminance, illuminance or intensity on some virtual measurement devices (observers). This form of representation is convenient when we are primarily interested in the modeling result itself, and not how it has been obtained, for example, modeling the brightness distribution on the surface of a liquid crystal display or an image generated by a photographic lens. But the most convenient way to obtain comprehensive information about light spreading in the optical system is visualization of the ray trajectories in the optical system. Also, visual representation of the trajectories of light rays in the optical system is useful for software developers, as means of debugging and optimizing algorithms [1].

For all kinds of rays visualization we use forward Monte-Carlo method. Each ray should be traced from a light source till the end of the trajectory which can be absorption, exiting out of scene boundary or achieving too great number of reflections and refractions [2, 3]. In some cases also it is possible indefinite state of a ray at the end of its trajectory. This state (called ray killing) will be described later.

One approach to scene definition for Monte-Carlo simulation is concept of "closed geometrical object" [4]. But it is currently absent in our algorithm. Instead of it for each object in the scene tree we set inside and outside medium. After interaction of ray with surface of any object the current ray medium is set. In case of reflection the ray medium stays unchanged, while in case of refraction it is changed to inside or outside medium of the object depending on whether there was already refraction

GraphiCon 2021: 31st International Conference on Computer Graphics and Vision, September 27-30, 2021, Nizhny Novgorod, Russia
EMAIL: birukov@gin.keldysh.ru (E. Birukov); ryzhova@gin.keldysh.ru (I. Ryzhova); voloboy@bin.keldysh.ru (A. Voloboy)
ORCID: 0000-0003-4297-6813 (E. Birukov); 0000-0003-1613-3038 (I. Ryzhova); 0000-0003-1252-8294 (A. Voloboy)

on the surface of this object or not. Unfortunately, sometimes this algorithm may cause errors in case of complicated geometry, so its improvement is going to be done in the further works.

Process or visualization of light rays in workspace of an optical modeling system can be held in two variants. As it was already told, trajectory of each ray in these cases is being calculated from its start point (any light source) to the end point (absorption or getting out of scene boundary). But in the first variant visualization of rays is done simultaneously with simulation, so that each newly calculated ray is immediately drawn in the workspace. As for the second variant simulation, in this case at first the entire simulation for given rays quantity should be completed. Then user should call an additional command to display calculated rays in the viewport. In this case usually not all rays are drawn but some of them selected by a certain criterion. Most often this criterion represents itself some area on a surface of a scene object or of a virtual measurement device which should be hit by rays. Both ways of rays' visualization are used in the optical modeling software complex developed by us.

Need for setting special ray coloring has many causes. One of these causes is efficiency analysis of an optical system. In this case user can vividly see ratio of the rays with required trajectories and all other rays, including rays reflected from the body of an optical system [1]. Also check for rays hit in some given scene object may be required when user wants to know if any rays hit this object or not. It is even possible to add such criteria for developing some complicated optical systems. Analysis of rays' trajectories which were deviated from the supposed trajectory is also possible, as well as many other applications. In some cases it is also necessary to paint even separate segments of the same ray with different colors.

2. Color modes for rays visualization

2.1. Natural color mode

In optical modeling complex developed by us three coloring modes are currently used for displaying rays' trajectories: natural color mode, artificial color mode with criterion applied to an entire ray, and artificial color mode with criterion applied to each ray segment. The simplest mode from the user's point of view is natural coloring mode. In this mode each segment of each ray is painted with the same color as the color which this segment has in reality. No additional settings are required in this case. This mode can be useful for analyzing color perversion in an optical system [5].

2.2. Artificial color mode for the entire rays

The next described color mode is painting of each ray with some color specified by certain criterion. In this case user should set a list of criteria and specify colors which correspond to each criterion. If some ray doesn't correspond to any criteria, then it is painted with default color. If a ray corresponds to several criteria, then it is painted with the color of the last satisfied criterion in the list. All other criteria in the list are ignored, there is no possibility or additional priority setting for criteria.

For example, let we have a criterion which requires hit to a certain object. This criterion corresponds to yellow color, and the default color is green. Then all rays which interacted with this object in any way (reflection, refraction, absorption) will be painted with yellow color. All other rays will be painted with green. Example of such coloring mode is shown on Fig. 1.

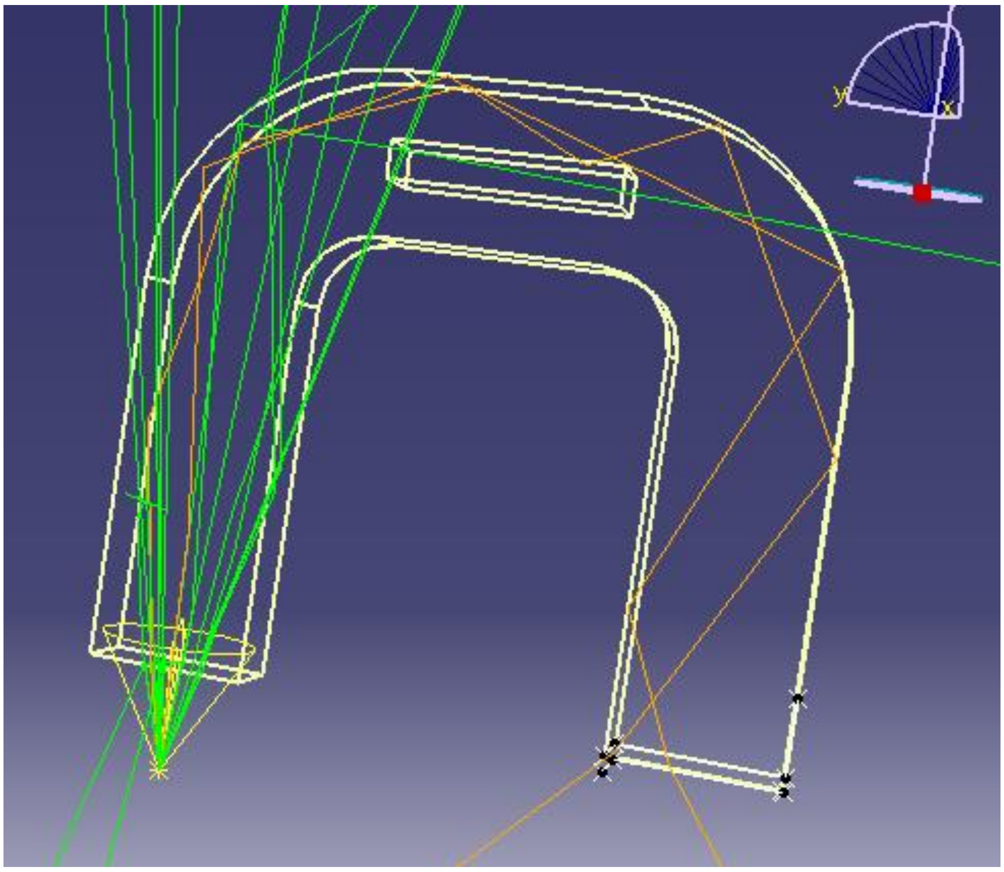


Figure 1: Example of painting rays with color criterion applied to the entire ray

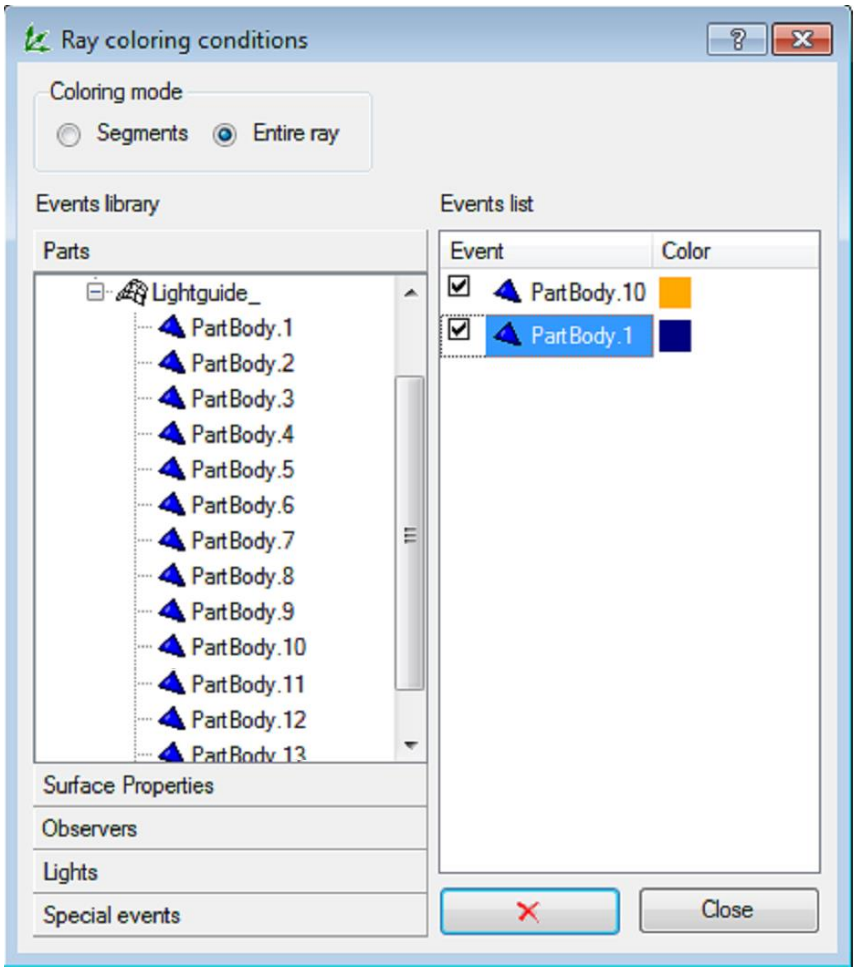
Conditions for coloring criterion can be different:

- Ray start at certain light source;
- Ray hit in certain object;
- Ray hit in any object with certain material applied;
- Ray hit in some observer (virtual measuring device);
- Certain type of event occurred with ray (reflection, refraction, absorption), on any scene object;
- Ray killing, i.e. indefinite state of the ray at the end of its trajectory. It occurs usually if the scene is set incorrectly, or if some computational error took place.

These conditions are almost equivalent to the conditions which were previously used for rays hide/show criterion [5, 6].

On Fig. 2 criterion building dialog is shown. In its left column one can see all types of events available for setting as such criterion – Parts (geometrical objects), Surface properties (materials), Observers, Lights and Special events. User creates new criteria by double-clicking items in this column. The right column contains list of created color criteria.

“Special events” kind of criterion allows setting certain type of event (reflection, refraction, absorption etc.) on any object. It can be useful, for example, to detect stray reflections on a lens system which cause highlights [7]. More complicated conditions for color criterion are also possible and often are used in this case. For example, for any event occurred with ray it is possible to set number of times which this event must occur to satisfy the criterion. In such a case several types of conditions are possible: the event should happen a strictly certain number of times, more that this number or less than it. Dialog for setting additional conditions for coloring criterion is shown on Fig. 3.



Ray coloring conditions

Coloring mode: Segments [✓ Entire ray]

Events library

Parts

- Lightguide_
  - PartBody.1
  - PartBody.2
  - PartBody.3
  - PartBody.4
  - PartBody.5
  - PartBody.6
  - PartBody.7
  - PartBody.8
  - PartBody.9
  - PartBody.10
  - PartBody.11
  - PartBody.12
  - PartBody.13

Surface Properties

Observers

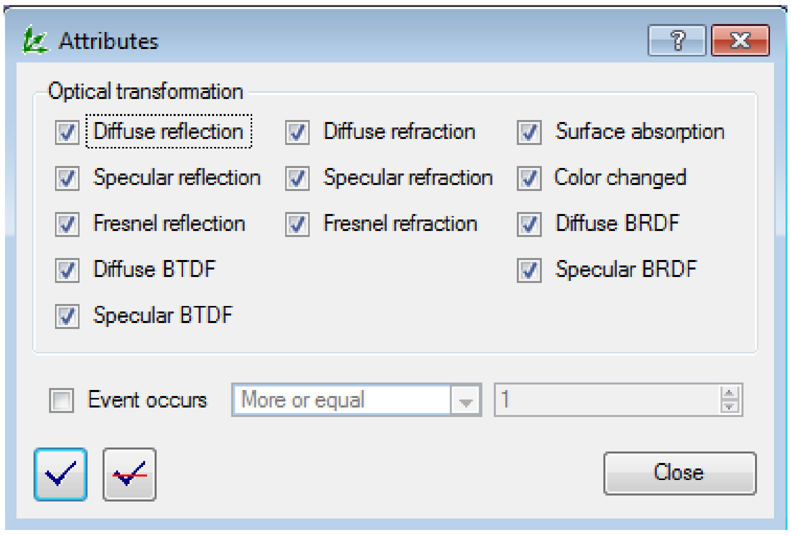
Lights

Special events

Events list

Event	Color
☑ PartBody.10	🟧
☑ PartBody.1	🟦

Figure 2: Color criterion dialog



Attributes

Optical transformation

- ☑ Diffuse reflection
- ☑ Diffuse refraction
- ☑ Surface absorption
- ☑ Specular reflection
- ☑ Specular refraction
- ☑ Color changed
- ☑ Fresnel reflection
- ☑ Fresnel refraction
- ☑ Diffuse BRDF
- ☑ Diffuse BTDF
- ☑ Specular BRDF
- ☑ Specular BTDF

☐ Event occurs: More or equal 1

Figure 3: Dialog for selection of additional conditions for the criterion.

2.3. Artificial color mode for particular ray segments

Finally, the last way of rays coloring is selection of particular ray segments with certain colors. In this mode each segment of each ray is painted with the color corresponding to some criterion only if this particular segment satisfies the criterion. The rest segments of the same ray will be painted with default color. Conditions for the criteria in this case are generally the same as for the entire ray. In the optical modeling complex developed by us it was taken that a ray segment satisfies the criterion if it's starting point satisfies this criterion. So, if some criterion implies ray hit into certain object, and some ray reflects from this object, then the segment just after reflection will be painted with color corresponding to the criterion, not the segment before it.

This mode allows to avoid limitations of the previous mode when some ray satisfies several criteria at once [8]. On Fig. 4 one can see that ray segments after reflection or refraction on the top face of the fiber are painted with orange color, and the segments after reflection or refraction on the bottom face at the end of the fiber are painted with dark blue, including cases when some previous segment of the same ray is painted with orange.

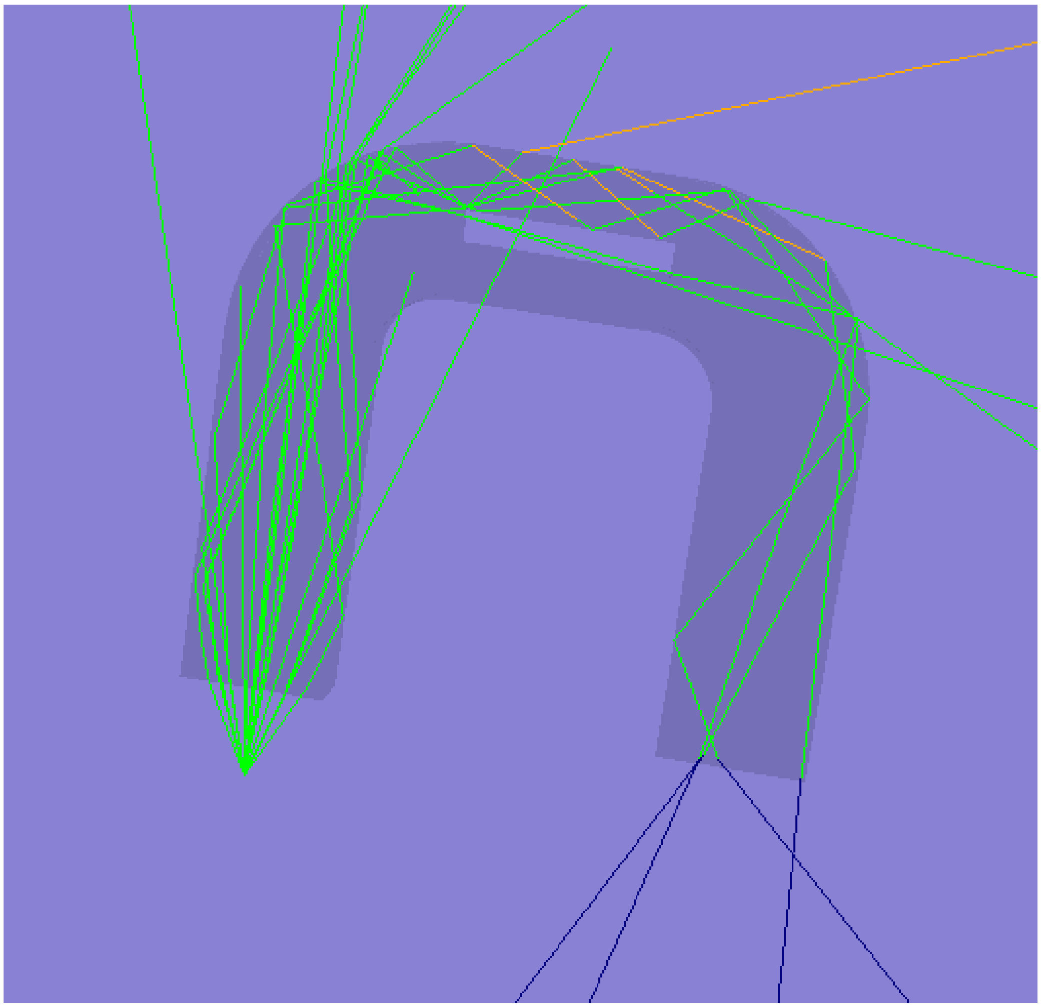


Figure 4: Visualization mode with each segment painted with its own color

3. Appearance of killed rays and special conditions of their display

Special attention should be drawn to usage of color criterion for selection of so called killed rays. A ray is called "killed" if its trajectory can't be defined after some event. If the scene geometry is set correctly and raytracing algorithm works properly then in theory such killed rays must not exist. Appearance of such rays can be caused by three circumstances:

- Error in the scene geometry made by user;
- Error or inaccuracy in the scene geometry obtained at the stage of scene preprocessing;
- Incorrect raytracing algorithm.

Error in the scene geometry made by user itself can occur, for example, if scene contains some geometrical element with an absent face, so that this element is not fully closed. A ray which hits this object and refracts its face, then enters object medium. After it this ray should cross some other surface of this object once more and exit from it, but such crossing doesn't occur. The ray goes into infinity, but at the same time it stays in the medium of the previous object because there was no more refraction. So trajectory of this ray can't be definitely calculated. Appearance of significant number of such rays in some part of the scene probably means that geometry in this place is incorrect and it should be corrected.

Error or inaccuracy during scene preprocessing occurs most often if a ray hits a degenerate triangle. Generally, appearance of relatively small quantity of degenerate triangles (and therefore, several killed rays) on complicated geometry does not indicate serious problems. If there are many such places, then probably geometry is set incorrectly, or there are errors in the preprocessor.

Sometimes it can be useful to kill the rays which do not satisfy certain criterion. It allows avoiding unnecessary calculations of rays which get off from required trajectory [9].

In the optical modeling complex developed by us, last segment of each killed ray is always painted with red color in any of two artificial coloring modes. Thus, creation of a separate criterion for killed rays in per-segment coloring mode does not seem possible.

4. Special features of rays visualization in our plugin to CATIA CAD

Optical modeling complex developed by us is produced in two variants – as a standalone application and as a plugin for CATIA computer-aided design system [10]. In the CATIA plug-in version there are some limitations of rays' visualization system caused by the architecture of this version of our complex. Most of simulation procedures in this version were moved to a separate utility, and as a result, we had to solve additional problems of commands and data transmission between these two programs. Thus, of the two modes of the entire ray visualization system (with immediate per-ray visualization and with complete tracing and delayed visualization) only the second mode is implemented. It is caused by the fact that messages synchronization and data transmission between two programs takes significant time. So if it is done each time for each new ray it will terribly slow the entire procedure. Due to it, in this version of our complex only the delayed visualization is provided. User should first start the simulation process for given number of rays and wait till its completion. Then rays data is written to special segment file. After it, when user initiates visualization of the simulation results, CATIA calls the special procedure of reading this file and selection of some number of rays for display from it, most often the rays are selected by some display criterion [5].

Data transmission between CATIA and the helper utility is implemented using shared memory system. Rays visualization with usage of this algorithm requires transmission through shared memory data arrays corresponding to these rays. This data includes information about rays (or segments) colors with color criteria taken into account.

Rays data for visualization sent from the additional utility to CATIA represent itself per-segment array of the following structures:

- Start point of the segment;
- Segment color;
- Segment type. Following types are used:
    - Intermediate segment;
    - Last segment of an absorbed ray;
    - Last segment of a ray gone out of scene boundary;
    - Last segment of a killed ray;
    - "End of ray" sign. It means that previous segment was the last one in its ray.

Thus, all segments of all rays are located in the single array. Drawing is performed according to the following algorithm. Each segment, if its type is no defined as "End of ray" sign, is drawn from the starting point of the current segment to the starting point of the next segment, forming a continuous polyline. If the type of this segment is defined as "End of ray" sign, then drawing of this polyline terminates and drawing of a new polyline for next ray starts (if end of the array is not achieved yet). Keeping additional data of the rays fate in its end point is used for selection last segments of killed rays with red color as well as for drawing particular symbol at the end of ray – arrow in case of ray gone out of scene boundary and cross in case of absorbed ray.

Additional limitation for ray segments array from additional utility to CATIA is caused by complicated procedure of dynamical arrays (with variable length) allocation in the shared memory. Thus, taking into account the fact that user will hardly get some valuable information from display of too great quantity of rays in the screen (about several hundred in reality) it was decided to keep segment array with constant length with some sufficient value. In the current version it is equal to 100 000 segments.

5. Conclusion

Ways of ray visualization coloring for ray trajectories simulated with Monte-Carlo method allow solving many different tasks during design of optical systems. These ways provide quick search of patterns of rays propagation in an optical system, and also finding and correction of errors allowed when designing such system.

Further work can be directed towards creating more highly specialized and complex criteria used for coloring rays and their separate segments, and also towards more deep integration of the developed optical modeling complex with different CAD systems. Besides this, improvement of the tracing algorithm itself may be provided for better handling of complicated geometry and decreasing of killed rays number.

6. References

[1] A.G. Voloboy, V.A. Galaktionov, A.D. Zhdanov, D.D. Zhdanov, Sredstva vizualizatsii rasprostraneniya svetovykh luchey v zadachakh proyektirovaniya opticheskikh sistem, Informatsionnye Tekhnologii I Vychislitel'nye Sistemy 4 (2009) 28-39. (in Russian)

[2] Matt Pharr, Greg Humphreys, Physically Based Rendering, Elsevier, 2004.

[3] T. Hachisuka, H. Wann Jensen, Stochastic progressive photon mapping, ACM Transaction on Graphics 28(5) (2009) 1–8. doi: 10.1145/1661412.1618487.

[4] D.D. Zhdanov, S.V. Ershov, S.G. Pozdnyakov, Obyektno-oriyentirovannaya model' postroyeniya fotorealistichnykh izobrazheniy, in: Proceedings of the 23rd International Conference on Computer Graphics and Vision, 2013 (in Russian).

[5] B. Kh. Barladyan, E.D. Birukov, L.Z. Shapiro, A.G. Voloboy, Visualization of ray propagation in physically accurate lighting simulation, Scientific Visualization 10(4) (2018) 75-92. doi:10.26583/sv.10.4.06.

[6] E.A. Kopylov, K.A. Dmitriev, Light propagation visualization as a tool for 3D scene analysis in lighting design, Computers & Graphics 24(1) (2000) 31-39. doi: 10.1016/S0097-8493(99)00135-1.

[7] D.D. Zhdanov, A.A. Garbul, V.A. Maĭorov, I.S. Potemin, V.G. Sokolov, Indeterminate ray tracing in problems of the analysis of light scattering and the design of illuminating systems, Journal of Optical Technology 81(6) (2014) 322-326.

[8] A. Voloboy, V. Vishnyakov, V. Galaktionov, D. Zhdanov, Visualization techniques of light propagation in optical design tasks, Preprints of KIAM of RAS, number 54, 20 pages, 2007.

[9] D.D. Zhdanov, I.S. Potemin, A.A. Garbul, V.G. Sokolov, Metody stokhasticheskoy trassirovki luchey v zadachah postroyeniya izobrazheniy, formiruyemykh realnymi opticheskimi sistemami, in: Proceedings of the 26th International Conference on Computer Graphics and Vision, 2016 (in Russian).

[10] Design Engineering | CATIA – Dassault Systemes, 2018. URL: http://www.3ds.com/products-services/catia