ISSN 0361-7688, Programming and Computer Software, 2018, Vol. 44, No. 4, pp. 225–232. © Pleiades Publishing, Ltd., 2018.
Original Russian Text © B.Kh. Barladian, A.G. Voloboy, V.A. Galaktionov, L.Z. Shapiro, 2018, published in Programmirovanie, 2018, Vol. 44, No. 4.

# Integration of Realistic Computer Graphics into Computer-Aided Design and Product Lifecycle Management Systems

**B. Kh. Barladian[a,*], A. G. Voloboy[a,**], V. A. Galaktionov[a,***], and L. Z. Shapiro[a,****]**

[a] *Keldysh Institute of Applied Mathematics, Russian Academy of Sciences, Moscow, 125047 Russia*

**e-mail: bbarladian@gmail.com*

***e-mail: voloboy@gin.keldysh.ru*

****e-mail: vlgal@gin.keldysh.ru*

*****e-mail: pls@gin.keldysh.ru*

Received March 15, 2018

**Abstract**—Various approaches to the integration of realistic image rendering and lighting simulation into computer-aided design systems are considered. An approach that ensures the effective integration of existing realistic image rendering systems into a CAD system is proposed. This approach makes it possible to utilize ready-to-use modules of a realistic image generation system, including the computation kernel and user interface modules. Synchronization of the executable modules of the basic and embedded systems is described. The proposed approaches and solutions were implemented in a project of integrating the lighting simulation and realistic image generation system Inspirer2 with the CAD CATIA.

**Keywords:** CAD, PLM, realistic image rendering, lighting simulation, integrated systems

**DOI:** 10.1134/S0361768818040047

## 1. INTRODUCTION

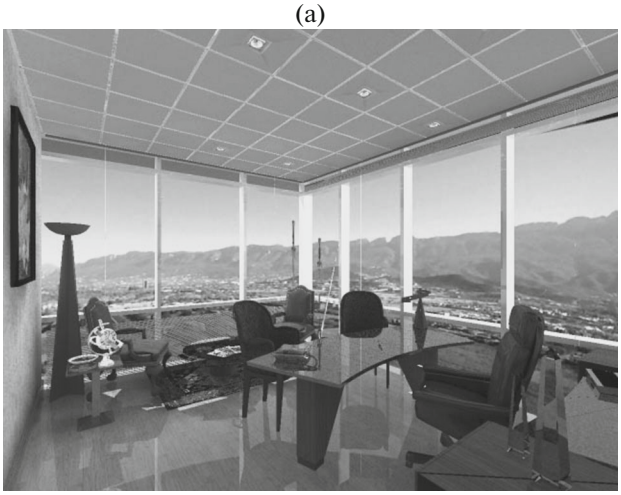
All modern manufactures use product lifecycle management (PLM) technologies for the design, manufacture, and maintenance of complex engineering products. These technologies support the management of the whole bulk of data concerning the product and related processes during its entire lifecycle, beginning from its design through manufacture, maintenance, and removal from service. The information about the product is stored in a PLM system; this information forms a digital model of the product. In such systems, the geometric model of the product is typically created using a CAD system. An important part of CAD is the computer-aided engineering (CAE) subsystem. In the architecture and design of various optical systems, the CAE system should simulate the light propagation in various media. This allows one to obtain optical characteristics of objects and products, such as illuminance, luminance in a given direction, visibility, and legibility of devices and displays already at design stage. The lighting simulation also makes it possible to generate physically correct realistic images of objects, including placing them into real images for the visual assessment of the final product (the augmented reality technology).

Presently, a large number of realistic graphics and optical simulation computer programs are available; these are Maxwell renderer, V-Ray, Mental Ray, SPEOS, ASAP, LightTools, ZEMAX, and Inspirer2. However, the independent use of such programs in the manufacturing process is limited. This is explained by two main factors—the need to import data concerning the product from the CAD system (during this process, some data can be lost or corrupted), and the need to teach engineers to work with one more system.

For this reason, in order to be used effectively, lighting simulation systems should be integrated into CAD and PLM systems. The integration should ensure a smooth interaction with the basic CAD system in which the geometry of objects is designed and their characteristics are stored. Ideally, the product designer should be able to use the optical simulation and realistic rendering module as a part of his (her) CAD system that automatically utilizes the geometry created in the CAD system and the attributes that must be used for simulation. The lighting simulation module must automatically take into account the changes in the geometry and attributes that can be made at various stages of design. Without such an effective integration, the designers will not use the program, except for critical cases in which the optical simulation cannot be avoided.

A widely used CAD system in aerospace and automotive industries is CATIA [1]. In the fields of architectural and interior design the most popular CAD is Autodesk 3DS Max [2]. For this reason, we decided to

(a)



(b)

**Fig. 1.** Realistic images of interiors produced by the module Insight in 3DS Max.

integrate our lighting simulation and realistic computer graphics program Inspirer2 with CATIA (under the name Lumicept/CA) and with 3DS Max (under the name Insight).

## 2. VARIOUS VERSIONS OF INTEGRATING THE LIGHTING SIMULATION MODULE WITH CAD

The implementation of the simulation and realistic computer graphics module in a CAD system is a difficult task. Attempts at such integration were made by Discreet (Lightscape), Mental Images, and Chaos Group (VRay) [3] for 3DS Max. In KOMPAS, the design of units of optical devices is automated [4], but no simulation is available. The company OPTIS developed an embedded lighting simulation module for CATIA—SPEOS CAA V5 Based [5] and for SolidWorks [6]. The company Mental Images integrated its lighting simulation program with 3DS MAX, Maya, and CATIA (PhotoStudio).

The integration of an earlier created lighting simulation program with a CAD system allows one to specify a detailed digital model of the scene to be rendered. Modern models include hundreds of thousand or even millions of objects, and they cannot be created without sophisticated mechanisms proposed by CADs. The integration modules for 3DS Max and CATIA developed by the authors of this paper allow us to generate realistic images for various materials and lighting conditions and to perform the engineering analysis of lighting and optical design. Examples of the realistic images produced by the Insight module for large scenes created in 3DS Max with a large number of components are shown in Figs. 1a and 1b.

A specific version of the integration of Inspirer2 with CATIA was proposed in [7]. A part of data is stored in CATIA (geometry) and the other part (attributes, light sources, and auxiliary parameters) in Inspirer2. Effectively, this solution is a geometry converter from internal CATIA format to the Inspirer2 format, where the realistic image is generated (Fig. 2). However, a feature of this approach to integration is that only the geometry that was modified in the basic system (CATIA) could be updated. In this situation, the entire set of data specified in Inspirer2 for the original geometry, which contains the optical properties of surfaces and spaces characteristics of light sources, etc., is stored. To this end, the CATIA data model was extended to enable it to store additional data. This approach considerably facilitated the user work on modifying the geometry, but it did not solve the main integration task because the user still had to work in two systems with different interfaces.

The direct and complete integration into a CAD system requires at least reprogramming of the entire user interface of the optical simulation system using the corresponding software development tools provided by the basic system. For CATIA V5, this is the Rapid Application Development Environment (RADE). In fact, such a direct approach also requires some parts of the kernel to be reprogrammed because RADE imposes certain restrictions on the objects stored in the system, the rules for their interaction, data storage, etc. Such an approach to integration can be implemented at reasonable cost only for relatively simple models of light sources and optical properties of surfaces and media in which the light propagates. An implementation of this approach in 3DS Max and CATIA is described in [7, 8].

The approach of direct integration of the optical simulation system into a CAD system gives users minimal functional capabilities, but it has two significant drawbacks.

1. High cost of implementation—several man years for the implementation of the minimal set of functional capabilities.

2. The effort needed to reprogram and debug the interface for complex models of light sources and optical properties of surfaces and media in which the light

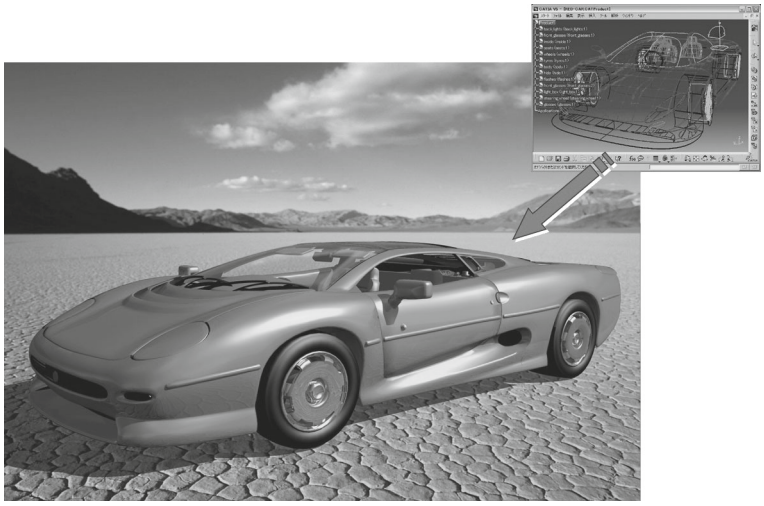


**Fig. 2.** The initial version of integration of Inspirer2 with CATIA: only the object geometry is passed from CAD.

propagates is so high that the implementation cannot be completed in a reasonable amount of time. The cost of implementation of sophisticated visualization algorithms, such as visualization of light propagation in an optical system using ray tracing, is also very high.

Due to these reasons, we decided to develop an effective and flexible integration scheme. The main requirements for the creation of an integrated system are as follows.

(1) The maximally possible use of parts of the existing and debugged lighting simulation and realistic image generation. In addition to the system kernel, it is desired to use some specific and difficult to implement modules of the user interface. This can save the development time and preserve the familiar user interface.

(2) The integrated system should be perceived by the user as a unified system; i.e., all the data needed for simulation, such as geometry and attributes, should be stored in the CAD system. The response time to modifications of geometry, attributes, and parameters should be minimized.

If these two requirements are satisfied, then the user-friendly integrated system can be developed in a reasonable amount of time. The satisfaction of these requirements also simplifies the support and extension of the system during its lifecycle.

## 3. THE OPTICAL SIMULATION MODULE

The optical simulation system Inspirer2 [9] developed in the Keldysh Institute of Applied Mathematics, Russian Academy of Sciences and based on the physically correct simulation of light propagation can produce realistic images taking into account various complex physical effects. The perception of generated images is close to the perception of photos of real objects. Such images are widely used in architectural design, urban development, landscape design, and in automotive and aviation industries. The system takes into account such intricate optical effects as light scattering from rough surfaces and regular surface microreliefs, volumetric scattering from microparticles inside a material, and a number of other phenomena. The system supports various models of light sources, in particular, light sources with a specified scattering indicatrix (goniogram). For the skylight and sunlight, the standard CIE model of natural daylight lighting or a high dynamic range image can be used as light sources. The system can make simulation in the RGB and in spectral color spaces; it can also simulate light propagation in scattering and fluorescent media. As a result, it can produce realistic images and optical characteristics, such as distribution of illuminance and luminance (both angular and spatial ones) on various real and virtual surfaces and scene objects.

The capabilities of the system just described required the development of rich user interface and the creation of special auxiliary data structures and internal interfaces. The complete integration of such a system into an external CAD system is extremely difficult, if possible at all.

For this reason, we proposed an approach that satisfies the requirements formulated in the preceding section and ensures the effective integration of the optical simulation system into a CAD system. The first results were presented in the conference [10].

## 4. LUMICEPT/CA INTEGRATION SCHEME

The components of the integrated system Lumicept/CA are shown in Fig. 3.

The user interface of CATIA was developed using MFC, and the user interface of the other components is based on QT SDK. The combined use of these two libraries in the same executable module is impossible.

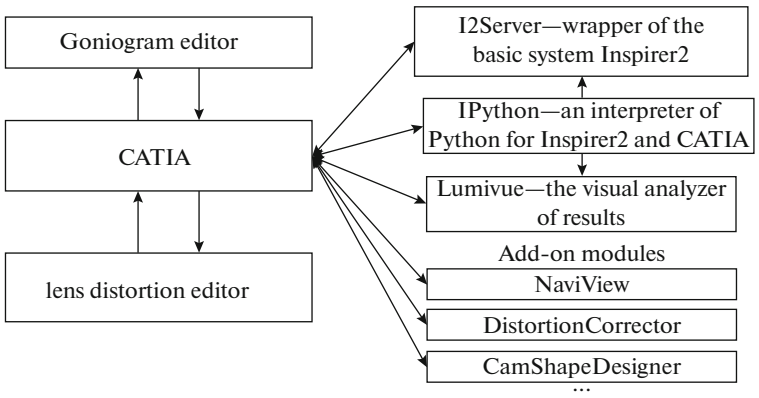



**Fig. 3.** Basic components of Lumicept/CA.

For this reason, each component shown in Fig. 3 is implemented in a separate executable module.

**CAD CATIA** is the basic CAD system. From the viewpoint of lighting simulation, it provides capabilities for the creation and modification of geometry of scene objects and data storage. To simulate lighting, CATIA was extended by user interfaces designed for the description and modification of optical attributes of surfaces, light sources, and virtual measuring objects called observers. Based on Inspirer2 SDK, the calculation of global lighting, measuring optical characteristics of the scene using observers, and generating realistic images using ray tracing were implemented. These basic functions were implemented directly in CATIA.

The user interfaces for specifying light sources and optical attributes of surfaces are implemented as additional tabs in CATIA interfaces; some native CATIA attributes are used. This approach facilitates CATIA users to master new capabilities because they should specify only the attributes that are specific for the simulation of lighting. The observers were implemented as CATIA objects. They are linked to geometric objects and visualized directly in CATIA windows.

The component **I2 Server** is implemented as a software shell of Inspirer2 objects using Inspirer2 SDK. It implements algorithms of realistic computer graphics based on the physically correct simulation of light propagation and various specific user interfaces available in Inspirer2.

Currently, this module provides such user interfaces as editing parameters of the light propagation medium, access to libraries of light propagation media available in Inspirer2, editing complex criteria used in the visualization of rays traced by the Monte Carlo method, and a number of user and programming interfaces.

The component **Lumivue** implements the analysis and processing of the generated realistic images and results of lighting simulation gathered by observers [11]. In particular, this module provides the following capabilities.

(1) Transform the images obtained in terms of physical quantities (high dynamic range images) into RGB space of a graphical display using various operators for compressing the dynamic range.

(2) Render simulation results using color contours (artificial colors).

(3) Visualize physical quantities in various cuts in graphical and tabular form.

(4) Analyze static characteristics of rectangular and elliptic regions of images.

(5) Visualize the representation of physical quantities in photometric and radiometric units and in various color spaces (RGB, spectral, HSV, etc.). For the quantities obtained using stochastic simulation methods, this module can visualize estimates of the computation accuracy.

(6) Analyze the simulation results obtained in the RGB and spectral color spaces.

(7) Analyze the simulation results obtained by various types of observers (angular brightness distribution, distribution of brightness or illuminance over the rectangle plane observer, etc.).

The component **IPython** is a Python interpreter that allows one to work with Inspirer2 and CATIA objects and procedures [12]. The object oriented architecture of Inspirer2 allowed us to implement the Python interpreter in the form of an independent component IPython, which provides access to practically all functions of Inspirer2, CATIA, and LumiVue. In particular, it makes it possible

- to specify one or more parameters of various objects;
- solve optimization problems;
- automate regression testing;
- create parametric objects.

The implementation of IPython is thoroughly described in [13]. Using the package of physically cor-

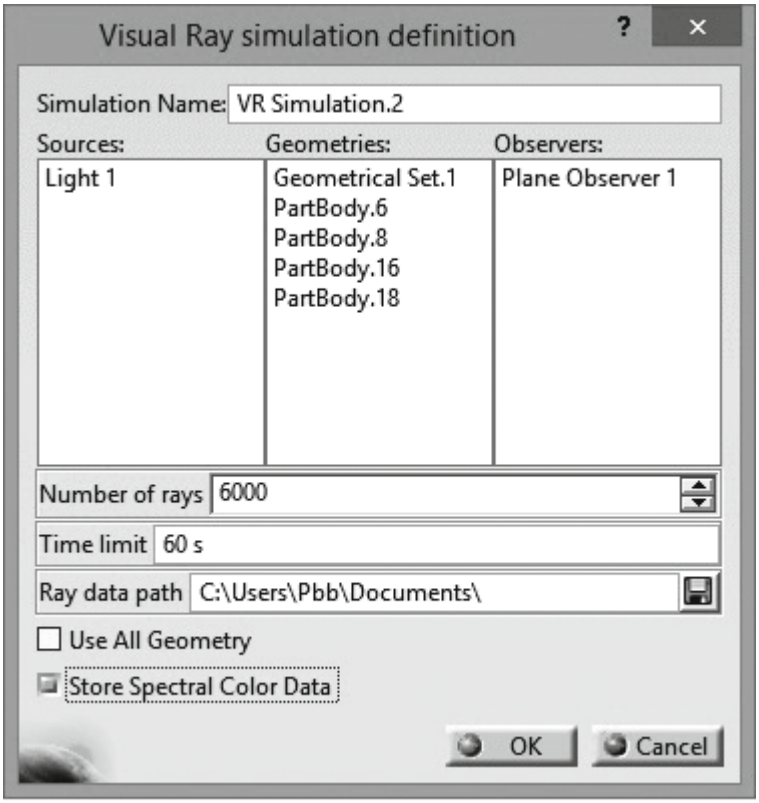



**Fig. 4.** Scene objects involved in the visualization of traced rays.

rect optical simulation as an example, Inspirer2 implements an approach based on using a unified interface of the subject area. In particular, a mechanism for extending the package by new types of parametric objects by writing simply structured extension classes is proposed. Special attention is paid to providing user-friendly application interface and to adhering to object-oriented paradigm.

CATIA supports three embedded scripting languages [14]:

- Visual Basic;
- VBA;
- CATScript.

The support of these languages is based on the Component Object Model (COM). COM is a standard for creating software components introduced by Microsoft. Programs created based on the COM standard are actually not autonomous programs but rather a set of interacting COM components. Each component has a unique identifier (GUID) and can be simultaneously used by many programs. The components interact with other programs through COM interfaces, which are sets of abstract functions and properties. Windows API provides basic functions for using COM components. IPython uses the Python extension library PyWin32 [15] for accessing Windows API functions. Thus, IPython provides access to all objects and procedures of CATIA for which COM interfaces are implemented.

The native objects and procedures of CATIA have these interfaces due to the architecture of the system. To provide access to optical simulation objects and procedures integrated into CATIA, we implemented COM interfaces for them where this was possible and seemed to be reasonable. In particular, these interfaces were implemented for two basic simulation procedures—the Monte Carlo ray tracing and realistic image generation procedures. The first procedure computes the global lighting and simulation results on observers, the second procedure generates realistic images using backward tracing of the rays emanating from the camera. We also implemented an interface for setting various parameters in procedures for the creation, modification, and saving the simulation results on observers.

The integrated system Lumicept/CA includes such auxiliary components as the editor of scattering indicatrices (goniograms) of light sources and the lens distortion editor. The interaction and synchronization of components is implemented through Windows messages and events. Data between components are passed using shared memory of processes.

I2 Server can be relatively easily extended to support new Inspirer2 capabilities in Lumicept/CA. For example, a new model of light propagation environ-

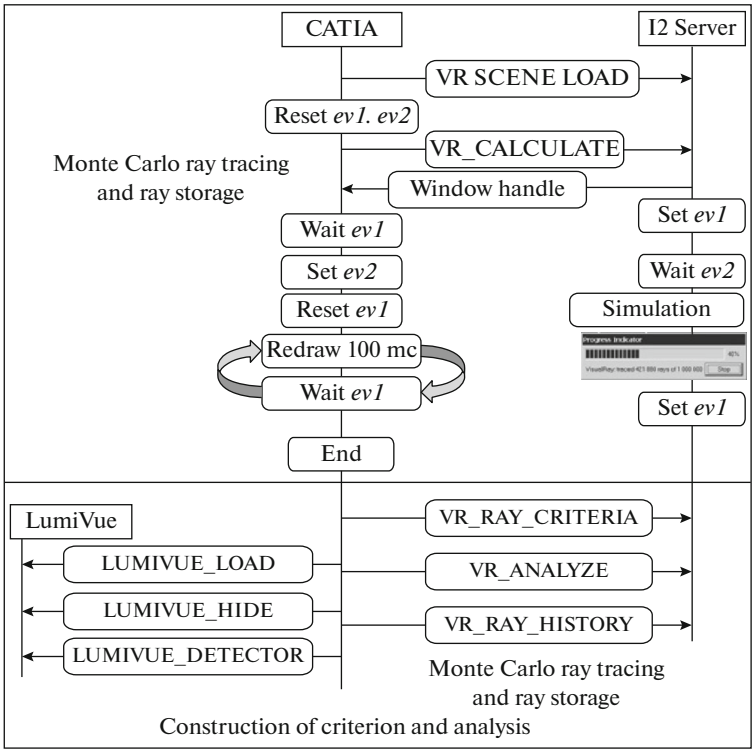



**Fig. 5.** Interaction scheme of CATIA with I2 Server and LumiVue in the visualization of light rays.

ment is added automatically because the corresponding user interface, object storing data, and the simulation procedure (Monte Carlo ray tracing) are implemented in DLLs that are common for I2 Server and Inspirer2. If new functionality requires the transmission of additional data between units, this is implemented using relatively simple add-ons in the corresponding units.

Note that the set of **Lumicept/CA** components can be extended if new tasks emerge such that it is better to implement them in a separate component. Figure 3 shows examples of three such components.

• **NaviView** simulates the rear view camera in a automobile navigation system.

• **DistortionCorrector** provides fish eye lens image correction in a form adapted for hardware implementation.

• **CamShapeDesigner** generates the geometry restricting the camera field of view.

## 5. SYNCHRONIZATION OF COMPONENTS IN LUMICEPT/CA

When an integrated system is implemented as a set of separate units, the main problem is to efficiently synchronize the subsystems implemented in different executable modules. The interaction between subsystems should preferably be hidden from the user to create a comfortable work environment.

The interaction between components was implemented using special Windows messages and events. For this purpose, the main Windows procedure of I2 Server and Lumiview was replaced by a special procedure that processes the messages sent by CATIA. To synchronize the other (simpler) components, a few Windows events turned to be sufficient: to continue its execution, a components waits until another component throws the corresponding event. Data between components can be passed in two ways. Small amounts of data are passed using the memory shared by processes. Large amounts of data are passed using the file system in the binary format of Inspirer2.

I2 Server is implemented as an ordinary Windows application, which starts simultaneously with CATIA. Its main window is invisible, and its main window procedure is replaced by processing the messages sent by CATIA. Special Windows events are used to synchronize CATIA with I2 Server in the process of computations and CATIA window repaint.

By way of example, consider in more detail the synchronization of the visualization component of light propagation in an optical system in the form of traced rays [16]. The operation of CATIA with the component Lumivue and the goniogram and lens distortion function editors is synchronized similarly.

At the first step, the user selects light sources, geometric objects, and observers involved in the simulation. Actually, at this step the user specifies a new

(a)

(b)

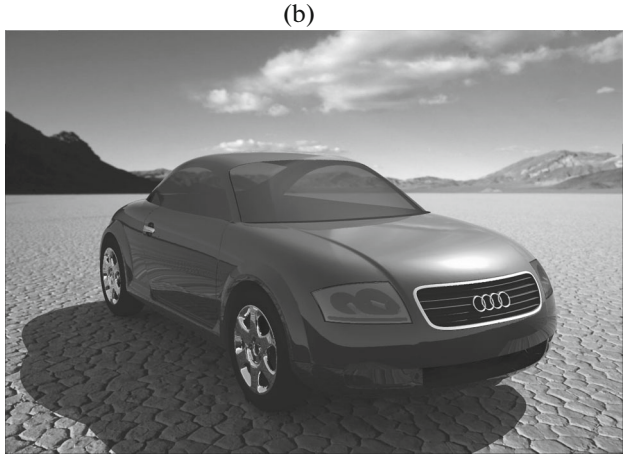


**Fig. 6.** Examples of lighting simulation results in the integrated Lumicept/CA system.

scene using objects of another already existing scene. This is done directly in the CATIA interface. The user selects objects in the visualization window or in the scene tree, and the names of the selected objects are shown in a dialog. Alternatively, the user can indicate that the entire scene will be used by selecting the corresponding check box. In the same dialog, the path for saving the simulation results, the number of rays to be saved, and a restriction on the simulation time are set. An example of this dialog is shown in Fig. 4. After the dialog has been closed, the specified parameters of the simulation session are shown in the scene tree, and the simulation can be started from the dropdown dialog by a right-click. The user may create any number of such simulation sessions for various needs.

The synchronization scheme of the simulation performed in I2 Server is illustrated in the upper part of Fig. 5. CATIA forms a scene description in the binary format of Inspirer2, saves it to disk, and sends the message VR_SCENE_LOAD to load the scene; next, it sends the message VR_CALCULATE (ray tracing with storing the results in a file for future visualization). Such additional data as the path to the scene description, simulation time restriction, and the number of rays are passed through shared memory.

The execution schemes of various I2 Server commands are close to each other. Here we describe the most complicated execution scheme of VR_CALCULATE in detail.

(1) CATIA throws off the events *ev1* and *ev2*, sends the message VR_CALCULATE, and waits for the occurrence of *ev1.*

(2) Having received the message VR_CALCULATE, I2_Server retrieves the file from the shared memory, loads the scene from it, places its window handle into the shared memory, throws the event *ev1,* and waits for the occurrence of the event *ev2.*

(3) CATIA retrieves from the shared memory the window handle, throws off the event *ev1,* and throws *ev2.*

(4) After the occurrence of the event *ev2*, I2 Server executes the simulation and then throws the event *ev1.* During the simulation, the progress bar shows the progress of the simulation execution as the ratio of the number of traced rays to the total number of rays. Then, I2 Server writes the simulation results to disk.

(5) When waiting for the occurrence of the event *ev1*, CATIA repaints its window with an interval of 100 milliseconds, makes the I2 Server window active, and places it on top of other windows. In response to the event *ev1*, CATIA reads from the disk the updated scene and accordingly updates the data in its document and the scene tree.

## 6. CONCLUSIONS

The approach proposed in this paper was used to include the main functions of the system Inspirer2 (such as computation of the direct and indirect lighting, visualization of ray tracing, and support of complex optical attributes and light sources) in the CATIA CAD system. As a result, the users working with CATIA can perform computational experiments with the models they design. For example, they can test the appearance of a paint-and-lacquer coating on the body of a car being designed under different external lighting as shown in Figs. 6a and 6b.

This approach enabled us to considerably speed up the development of the integrated system and improve its reliability because the most complicated modules have already been debugged in the autonomous system. As a result, the time needed for the implementation of new functions in Lumicept/CA that were already available in Inspirer2 was drastically reduced. This is especially important because the cost of the CATIA programming environment (the license cost) is high and the development requires highly skilled experts with good working experience in this environment. The proposed approach was implemented for CAD CATIA, but it also can be applied to other CAD systems.

## ACKNOWLEDGMENTS

This work was supported in part by the Russian Foundation for Basic Research, project nos. 16-01-00552 and 18-01-00569.

## REFERENCES

1. Dassault Systems, CATIA. http://www.3ds.com/ru/products/catia/. Cited January 10, 2018.
2. Autodesk 3DS MAX. https://www.autodesk.ru/products/3ds-max/overview. Cited January 10, 2018.
3. Osinev, A., V-Ray: Appreciate your time, *SAPR Grafika,* 2010, no. 9, pp. 88–89.
4. Kolpakov, A. and Tostoba, N., Automating the design of a unit of an optical devise, *SAPR Grafika,* 2012, no. 12, pp. 68–70.
5. Gol'dovskii P. and Kokova, A., Simulation of optical phenomena and properties of various products, *SAPR Grafika,* 2004, no. 8, pp. 46–47.
6. Alyamovskii, SolidWorks/OptisWorks—An integrated environment for the analysis anb synthesis in optics: Optical analysis and software structure, *SAPR Grafika,* 2006, no. 4, pp. 73–79.
7. Barladian, B. Kh., Voloboy, A.G., and Shapiro, L.Z., Integration of illumination simulation by ray tracing method into CAD systems, in *Proc. of the 23rd Int. Conf. GraphiCon'2006,* Novosibirsk, Russia, 2006, pp. 275–278.
8. Barladian, B. Kh., Voloboy, A.G., Galaktionov, V.A., and Shapiro, L.Z., Integration of illumination simulation software into CAD and CAM systems, *Proc. of the Int. Conf. and Exhibition CAD/CAM/PDM-2006,* Moscow, 2006, pp. 16–20.
9. Zhdanov, D.D., Potemin, I.S., Galaktionov, V.A., Barladyan, B. Kh., Vostryakov, K.A., and Shapiro, L.Z., Spectral Ray Tracing in Problems of Photorealistic Imagery Construction, *Program. Comput. Software,* 2011, vol. 37, no. 5, pp. 236–244.
10. Barladian, B. Kh., Voloboy, A.G., and Shapiro, L.Z., Generating realistic images in CAD systems, in *Proc. of the 23rd Int. Conf. GraphiCon'2013,* Vladivistok, Russia, 2013, pp. 186–190.
11. Barladian, B.K., Potemin, I.S., Zhdanov, D.D., Voloboy, A.G., Shapiro, L.S., Valiev, I.V., and Birukov, E.D., Visual analysis of the computer simulation for both imaging and non-imaging optical systems, in *Proc. of the Society of Photographic Instrumentation Engineers 10021, Optical Design and Testing VII,* 2016.
12. Deryabin, N.B., Sokolov, V.G., Zhdanov, D.D. and Kopylov, M.S., Automating the generations of series of realistic images using Python scripting language, in *Proc. of the 25th Int. Conf. GraphiCon'2015,* Protvino, Russia, 2015, pp. 132–136.
13. Deryabin, N.B., Zhdanov, D.D. and Sokolov, V.G., Embedding the script language into optical simulation, *Program. Comput. Software,* 2017, vol. 43, no. 1, pp 13–23.
14. Introduction to CATIA V5 Automation. http://catiav5automation.blogspot.ru/2013/05/ introduction-to-catia-v5-automation.html. Cited February 5, 2018.
15. Python for Windows Extensions. https://sourceforge.net/projects/pywin32/. Cited February 7, 2018.
16. Barladian, B., Shapiro, L., and Voloboy, A., Ray maps technique for effective interrogation of results of MCRT simulation, in *Proc. of the 21th Int. Conf. on Computer Graphics and Vision GraphiCon'2011,* Moscow, 2011, pp. 46–49.

*Translated by A. Klimontovich*