Jr. of Industrial Pollution Control 32(2)(2016) pp 453-458
www.icontrolpollution.com
Research

VISUALIZATION OF EXPERIMENTAL DATA FOR THE PROCESS OF SOLID MATERIALS ATOMIC STRUCTURES PROPERTIES MODELING IN THE ANTARES QUANTUM-CHEMICAL CALCULATIONS SOFTWARE PACKAGE

S. KUZNETSOV [1], A. OBKHODSKY [1]*AND A. POPOV [2]

[1] National Research Tomsk Polytechnic University, 634050, Tomsk, Russian Federation

[2] LLC "Depos", 634033, Tomsk, Russian Federation

(Received 6 July, 2016; accepted 20 August, 2016)

Key words: Material properties modeling, Software architecture, OpenGL, Graphic objects texturing, Crystal lattice, Data visualization, Temperature gradient

ABSTRACT

This paper focuses on modern ways of visualization of large amounts of data. The authors describe a data visualization program based on the OpenGL library that provides static and dynamic modes of representation of the heat transfer process in the scope of material piece and alteration of the material atomic structure. The visualization program is integrated in the Antares material modeling software package that operates on the basis of the distributing computing GRID. Experimental test of the designed program is performed using the example of Ti_3Al system atomic structure visualization.

INTRODUCTION

Modern graphic systems are powerful enough to create complex animated and dynamic images. Computer graphics allows to show ongoing physical processes visually, while computing methods let us alter different physical parameters, thus helping to investigate these phenomena more widely. Computer technologies help us concentrate on the most important things, which favors deeper understanding of the ongoing process. It is well known that information which is represented in several ways (animated, graphical) is more effectively apprehended. Animated models allow to show visually how exactly the modeled process in going on (Gorohov, *et al.*, 2007).

During the last decade possibilities to represent results in computing modeling have also increased due to the development of computing devices, appearance of parallel computations and supercomputers.

RESULTS AND DISCUSSION

Application of OpenGL for experimental data visualization

In modern scientific and technical applications the OpenGL graphic library is often used for complex graphic visualization. This library is nowadays the standard of three dimensional graphics construction. The OpenGL library is a highly efficient software interface for graphic hardware. This library is most effective in hardware systems based on modern graphic accelerators with an integrated graphic processor. In this case resources of the central processor are released and it can perform mathematical calculations that are necessary for visualization.

OpenGL architecture and algorithms were developed in 1992 by experts of Silicon Graphics, Inc. for Iris graphic workstations hardware. Several years later the library was ported to many hardware platforms (Jejndzhel, 2001).

Graphic libraries ease the work of designers and save them the trouble of manual imaging. In order to represent the data it is enough to generate a three dimensional scene with engaged objects, identify optical properties of these objects and set up characteristics of the sources of illumination.

The OpenGL library uses two types of data:

- vector data (vertexes, lines, polygons);
- raster data (pixels, raster images, bitmaps).

***Corresponding author's email: art707@tpu.ru**

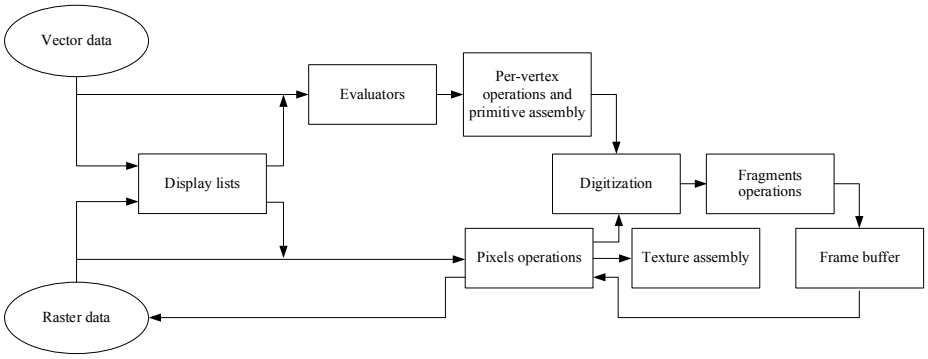

Fig. 1 OpenGL graphic library conveyor-like architecture.

The OpenGL vector primitive element is a vertex. This is an elementary vector object that can be represented as a dot. All the other vector objects are represented as a set of several vertexes. The OpenGL raster primitive element is a pixel. All other raster objects are represented as pixel bitmaps.

The OpenGL library has a conveyor-like architecture (Fig. 1) that allows to divide the required actions into independent operations that can be performed at the same time in a separate software or hardware block. This is the conveyor-like architecture that allows high efficiency while solving difficult graphic problems.

In order to be represented, vector geometric primitive elements should be presented as a set of vertexes. However, many curves and surfaces can be described using a much less number of parameters. Computing machines allow to form a set of vertexes using a simple description of the geometric primitive element. At the same time it is possible to define the number of vertexes which allows to control the quality and productivity of the visualization process. The result of the computing device passing is a scene represented by vertexes.

The next stage is processing of the vertexes and assembling of the primitive elements. Every vertex undergoes operations required for its representation, i.e. projecting of coordinates on the two-dimensional screen surface, identification of the vertex color with consideration of illumination and material properties, application of textures, etc. Besides, invisible lines and vertexes that do not get into the port and the output depth are being cut off or covered by other primitive elements. The result of this stage is full description of geometric primitive elements and their properties that are located in the space according to the set system of coordinates. Then the received data undergo the digitization stage.

Raster data undergo an additional stage of pixels operations. Pixels from bitmaps located in the system memory are transformed from the storage format to the picture format. If necessary, they are zoomed or relocated. The obtained result is also directed to the digitization stage or saved in the memory as a texture.

At the texture assembly stage every vertex of graphic primitive elements is being corresponded to the texture coordinates. The obtained texture is permanently stored in the memory and copied to the frame buffer, if necessary.

During digitization some fragments are formed from vector and raster data. Each fragment corresponds to a frame buffer pixel. At this stage, color filled polygons, dash lines with a set width and dots of a set size are formed, shading is used.

Before the obtained color values for every pixel are located in the frame buffer, each fragment may be additionally processed. At this stage the prepared textures from the memory are overlapped, transparency is defined, templates and masks are applied, logical operations are performed and images are applied. Only after this the processed fragments are transferred to the active frame buffer, its pixels are assigned the values of the finally represented color.

The frame buffer is a memory area where the pixel bitmap displayed on the screen is stored. The OpenGL library can handle several frame buffers at the same time, i.e. while one frame is being displayed the following ones are being formed. This ensures high speed of formation and visualization of animated frames. The obtained pixel image may be stored from the frame buffer to the system memory for a future reuse (OpenGL Programming Guide).

Normally, application of texture in three dimensional visualization applications is associated with object decoration. However, architecture of most modern graphic systems works in a way that allows to use texture to successfully solve the problems that cannot be solved using direct methods. Besides, using of textures in most cases greatly decreases

computation efforts, which allows to use an interactive visualization mode. The texturing method implies overlapping of an image (texture image) on the surface of a certain object.

Texturing is also effective in terms of visual representation of physical properties of the studied object in scientific visualization applications. For instance, information about the temperature of an object can be coded using a certain color and applied to the object itself, thus allowing to see clearly how the geometry of the object influences heat transfer processes inside it.

Problems that are solved using texturing can be solved using other methods, for instance using the method of direct color modification of the object. In this case division of the processes of geometry formation, application of texture and color filling takes more computing resources, the time period for frame preparation also increases. Graphic accelerators of modern work stations are designed in a way that a dedicated graphic processor for texturing and the random access memory for storing texture images are given. As a result, during the image rendering, texturing is performed with a dedicated hardware conveyor that works together with a geometrical one. Such hardware allows to decrease the time period devoted to texturing (Richard, *et al.*, 2006).

Coloring of polygon surfaces is implemented in OpenGL in a way that the color is set only for the grid vertexes. In the surface points that are not vertexes the color is identified using interpolation. Modern graphic architectures are based on two main ways of shading interpolation: flat shading and Gouraud shading.

Flat shading is, from the point of view of computing, a simple and quick way. Shading of any polygon interior is performed using one color tone that is set in a certain point of the polygon. As a result, visualization does not always provide quality representation of the object, since one can see sides of the adjacent polygons.

Gour aud shading produces continuous shading between sides of polygons. This continuous shading is performed by computing the color linearly interpolation for each polygon pixel. As a result, the shaded surface displays light reflection and the object itself in a more realistic way (Teschner and Henn, 1995). This way of shading is chosen for experimental data visualization in material properties modeling hardware

Structure of the data visualization program

In order to increase the quality of industrial production and products in general one needs the goods that meet high requirements in terms of physical and mechanical properties, geometrical precision of prefabrication, surface unevenness, strength and several other parameters. All these properties depend on the structure of a material to some extent. Therefore, it is necessary to consider the structure of materials in more details when studying the processes of heat and mass transfer, deformation and destruction.

Computer-aided material properties modeling is characterized with a high predictive ability and plenty of potential for further development of new materials with set properties and characteristics. Such modeling is based on multi-level physical models that describe structure and properties of materials at the atomic level and on a computing approach that integrates quantum chemistry and molecular modeling methods (Grigorev, *et al*, 2010; Bagaturyants, *et al*, 2007; Pas'ko and Piljugin, 2009).

The program of material structure visualization at the atomic level is an obligatory tool in the process of analysis of theoretical and experimental data results. It allows to combine different levels of modeling and compare the obtained results of these studies with known experimental properties of materials. Besides, graphical representation of the atomic and molecular structure of the material, primarily of the mutual location of atomic cores in space as well as of the characteristics of the electronic system structure allows to visualize areas of the most significant structural alterations during certain physical or chemical processes (Tarini, *et al.*, 2006; Al'es, 1999).

In order to visualize experimental data in the Antares material modeling software package a special program was designed. It allows to represent temperature distribution in the scope of the material piece as well as the crystal lattice structure of the area in question. The program features static and dynamic heat transfer visualization modes. The program can be operated on the bases of the hardware represented in Fig. 2.

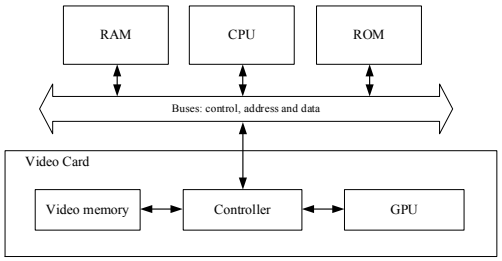

Fig. 2 Structure of hardware for the data visualization program operation.

Pattern of interaction between the visualization program and the Antares hardware is represented in Fig. 3.

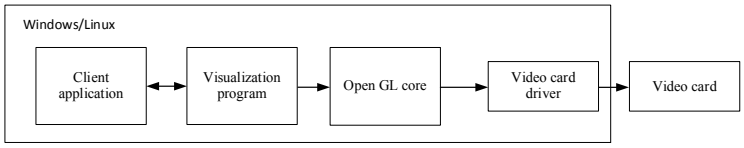

Fig. 3 Pattern of interaction.

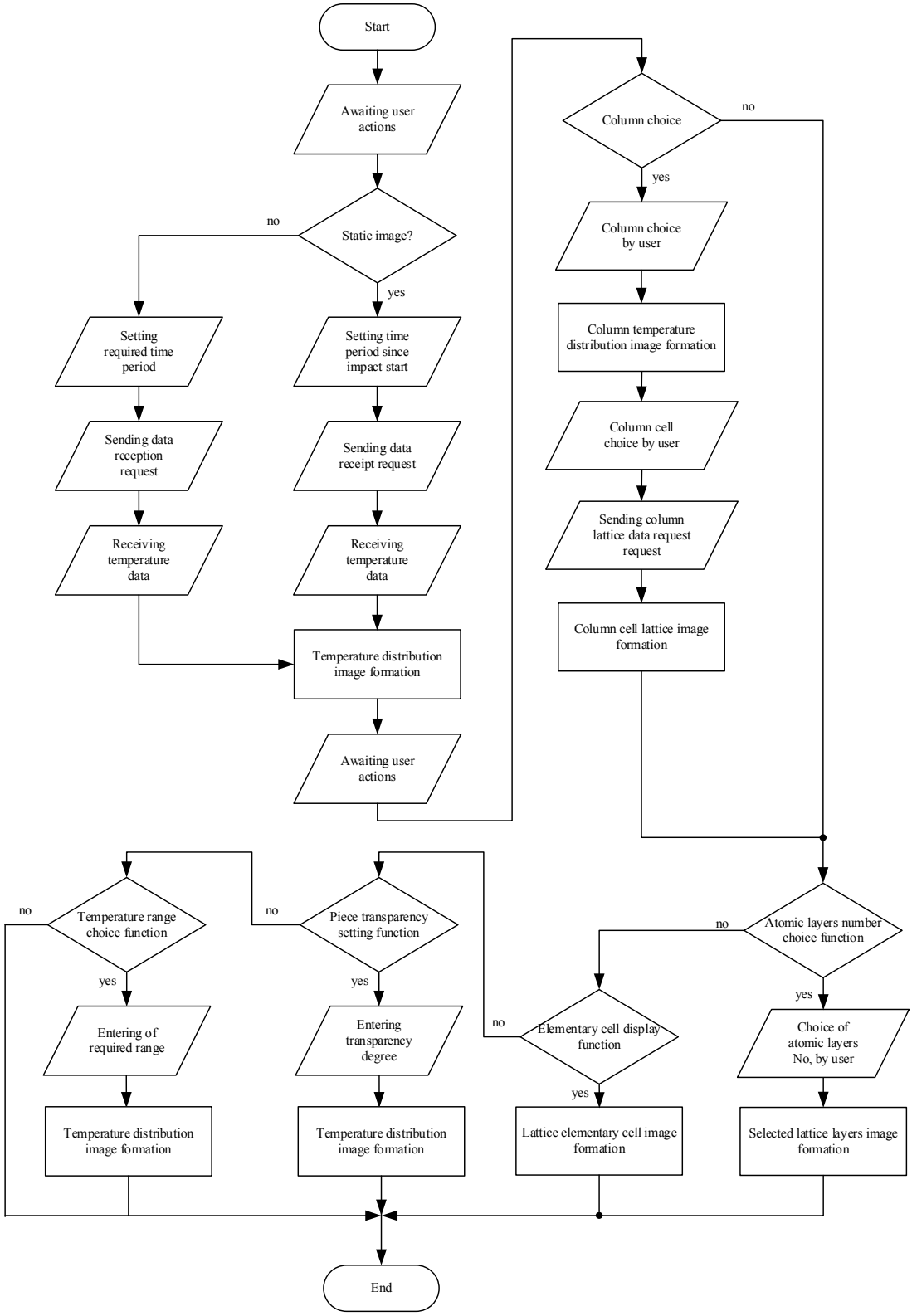

Fig. 4 Algorithm of data visualization program operation.

The visualization program is cross-platform and based on application of OpenGL program library functions.

Algorithm of data visualization program operation

The program is launched by the client application that is a part of the Antares software. The visualization program is operated according to the algorithm presented in Fig. 4.

After the initialization of the program the user should set up a visualization mode. If the user has to visualize data for a certain period of time, the user needs to specify the period of time beginning with the moment of impact application to the material and the duration of the impact application. After that, the program prepares a data reception request to the client application taking into account the user's choice. According to the received information, a temperature distribution image is formed in the scope of the material piece using the method of projecting values to the color palette (Fig. 5).

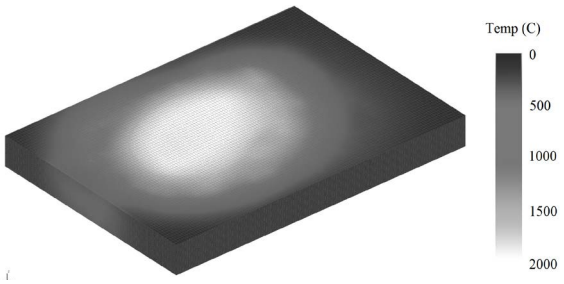

Fig. 5 Result of visualization.

During the analysis of the obtained image the user may choose a temperature range. In this case all material areas with the temperature values outside the specified range are filtered out. Besides, there is an option for choosing a degree of material transparency integrated in the data visualization program.

Visualization of temperature distribution in the material piece volume and on the surface allows us to see the general picture of the temperature impact process. Types of atomic structure of the studied material are also interesting. It is not reasonable to visualize the crystal lattice of the whole material piece since it requires a large amount of data which needs to be transferred, processed and displayed. This excessive amount of information is not going to be apprehended by the user.

A separate interactive algorithm is used for the user's convenience and for speeding up the program operation process. A coordination grid is applied to the whole surface of the piece. The user chooses an area of interest, after which the program prepares a request to the client application to receive data regarding the crystal lattice of the chosen area. According to the received data, an image of the crystal lattice is formed.

The user may represent an elementary crystal lattice cell (Fig. 6) or a set of atomic layers (Fig. 7).

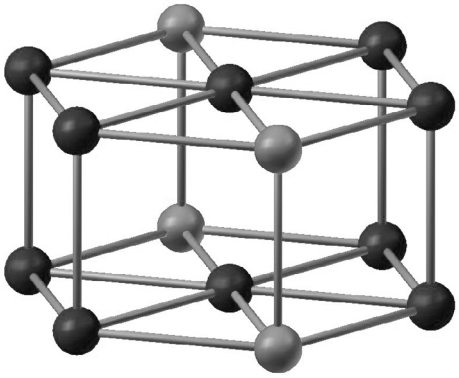

Fig. 6 Elementary crystal lattice cell.

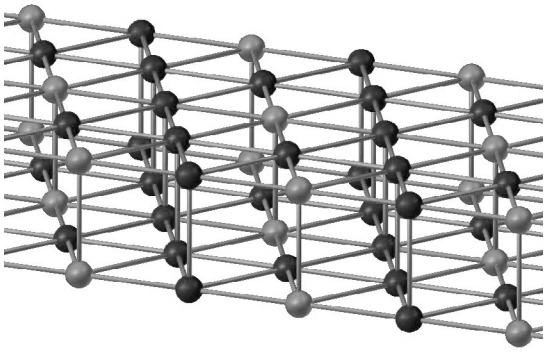

Fig. 7 Set of crystal lattice atomic layers.

The process of forming a crystal lattice image is going on in the following way. The visualization program receives a file from the client application that contains information about the location and connections of atoms. Each atom takes its place in the three dimensional space in accordance with received coordinates. Below there is an example of a program code for representing one atom.

```
glPushMatrix();
glTranslated(X, Y, Z);
glColor3f(0.7f, 0.7f, 1.0f);
glutSolidSphere(2, 16, 16);
glPopMatrix();
```

The visualization program also receives information regarding the temperature gradient from the client application. A coordinate grid is applied in the original material piece. After this, each cell is assigned a color value that corresponds to a certain temperature value. Below is an example of a code to represent one grid cell.

```
glColor3f(R, G, B);
glBegin(GL_QUADS);
glVertex3f(0.0f, 0.0f, 0.0f);
glVertex3f(0.0f, 1.0f, 0.0f);
glVertex3f(1.0f, 1.0f, 0.0f);
glVertex3f(1.0f, 0.0f, 0.0f);
glEnd();
```

For more convenient perception, the interpolation method is used to shade the colors of cell borders.

CONCLUSION

So, the result of the work is a data visualization program that provides static and dynamic modes of the heat transfer process representation in the scope of the material piece and alteration of the material atomic structure. Further expansion of the program functions is planned. An option to model liquid and gas mixing processes will be added as well as to model processes of mechanical impact on materials.

The work is performed with financial support of the Ministry of Education and Science of the Russian Federation. Grant agreement No. RFMEFI57814X0095 as of November 28, 2014.

REFERENCES

Al'es, M.J. 1999. Mathematical modeling in science and technology. (MMNT'98). Published house IPM UrO RAN. Izhevsk.

Bagaturyants, A.A., Deminskii, M.A., Knizhnik, A.A., Potapkin, B.V., and Umanskii, S.Y. 2007. Integrated Approach to Dielectric Film Growth Modeling: Growth Mechanisms and Kinetics. Thin Films and Nanostructures: Physico-Chemical Phenomena in Thin Films and at Solid Surfaces. Elsevier.

Gorohov, V.L., Luk'janec, A.A., and Chernov, A.G. 2007. Modern methods of cognitive visualization of multidimensional data. Non-commercial fund of regional energy development. Tomsk.

Grigorev, .FV., Romanov, A.N., Lajkov, D.N., Zhabin, S.N., Golovacheva, A.J., Oferkin, I.V., Sulimov, A.V., Bazilevskij, M.V., Bagaturjanc, A.A., Sulimov, V.B., and Alfimov, M.V. 2010. Methods of molecular modeling of supramolecular complexes: hierarchical approach. Russian nanotechnology. 5 : 47-53.

Jejndzhel, J.e. 2001. Interactive computer graphics. Moscow.

OpenGL Programming Guide: The Official Guide to Learning OpenGL. Version 2. Fifth Edition. Addison-Wesley.

Pas'ko, A.A., and Piljugin, V.V. 2009. Scientific visualization and its application in the nanostructures research. Rusnanotech. Moscow.

Richard, S.R., and Lipchak, B. 2006. OpenGL. Superbible. 3. Moscow.

Tarini, M., Cignoni, P., and Montani, C. 2006. Ambient Occlusion and Edge Cueing for Enhancing Real Time Molecular Visualization. IEEE Transactions on Visualization and Computer Graphics. 12 : 1237-1244.

Teschner, M., and Henn, C. 1995. Texture Mapping in Technical. Scientific and Engineering Visualization. Technical Report. Silicon Graphics Inc.