3D VISUALIZATION OF EXTRA LARGE DATA ARRAYS
BY MEANS OF SCIENTIFICVR® VISUALIZER

Alexey V. Zibarov*, Andrey N. Karpov*, Alexey V. Medvedev*,
Ilya Y. Komarov* and Dmitriy A. Orlov*
*GDT Software Group,
Pervomaiskaya 44-16, 300034, Tula, Russia
e-mail: info@cfd.ru
Web page: http://cfd.ru/

Key words: Parallel Computing, Visualization System, Voxel, Distributed Computing,
Multiprocessor Computer Systems

Abstract. Graphic presentation of the results in the form of color lay-outs or functional
dependence diagrams makes it considerably easier to perceive new information and
contributes to a more sophisticated and efficient analysis of complicated processes and
phenomena. This becomes particularly important in the course of processing big 3D arrays
that one has to deal with mainly when implementing computational simulation or processing
experimental or analytic data. The existing visualization systems are characterized by a
number of limitations. The main limitations are a maximal visualized data array size and bad
3D data distribution presentation. That's why the GDT Software Group has developed a new
visualization system ScientificVR® (SVR) offering numerous facilities and devoid of all the
limitations typical for other systems. The main SVR advantages are: 3D voxel graphics,
stereoscopic visualization, visualized data array size limit absence (this is of especial
importance for data obtained from parallel computation systems) and on-the-fly visualization
possibility. Voxel is a 3D analogue of pixel. The voxel is semitransparent, and its color is
defined by the visualized function value. Voxels transparency allows the user to see the
function distribution in the whole 3D area, not only in selected sections. Transparency can be
adjusted to provide the most informative visualization. Stereoscopic visualization allows
creating stereo images and animations. Stereo presentation is much more effective than usual
and makes the data analysis process more comfortable. The visualized data array size limit
absence is provided by means of a special data averaging procedure, which is to be done on
parallel computation nodes by means of controlled downsampling procedure. As a result the
maximal size of visualized data depends on the amount of RAM in a parallel computation
system only. Required RAM size on the computation system rendering a visual presentation
depends on the resolution of a resulting image only. Special programming strategy, including
SDK, was developed to implement such behavior in an assumed, user-developed parallel
application. This article is devoted to description of development ways of a new visualization
system, working on various distributed computing systems with a random user application.
The most prospective and efficient way to develop the new visualization system on the basis of
a SVR visualization system, developed by the GDT Software Group is proposed.
1 INTRODUCTION

Today numerical simulation with packages meant for multiprocessor computer systems becomes more and more widespread. However, graphic processing of the data generated in the course of such computations brings about huge amounts of computing resources being used. This is caused by the necessity to transfer data to the visualization server, as well as “raw” data processing and immediate rendering of complex images. Implementation of online dynamic visualization involves additional difficulties. Nevertheless, it is this very type of visualization that offers researchers the most possibilities: to observe the simulated processes and analyse the obtained data immediately in the course of computation, to promptly detect errors made at the project development stage or reveal the necessity to adjust the model itself for a more adequate representation of real processes.

Considering all the aforesaid, the new visualization system development featuring the following capabilities stays quite an issue of the day:

- dynamic visualization of data obtained from multiprocessor computer systems immediately in the course of computation;
- visualization of huge amounts of data (100 GByte – 1 TByte and more);
- visualization of huge amounts of data with simultaneous display in various representations (coordinate and functional transformations included).

The SVR developed by GDT Software Group can serve as a basis for such a software package.

2 FUNCTIONING AND ABILITIES OF THE SVR

SVR is a high-performance and flexible scientific visualization package. The version that has been implemented by now allows working with single-processor systems under Windows and Linux operating systems, to implement dynamic data visualization for the GasDynamicsTool® (GDT) CFD package, developed by the GDT Software Group, to implement static data visualization for user applications. As part of the hybrid version of the GDT package, it is featuring “on-the-fly” dynamic visualization of data obtained from nodes of distributed computing systems (see fig. 1, 2).

Figure 1: Acetylene tank explosion near a building
The data is visualized using the semitransparent voxel technique. SVR is based on a modified variant of visualization scheme on a dedicated node (fig. 3).

The most significant features of the package are the following: plug-in architecture, semitransparent voxel graphic and full color 3-d stereoscopic presentations. The SVR structure can be presented as a set of five subsystems:
• data input subsystem;
• coordinate transformations subsystem;
• functional transformations subsystem;
• image generation subsystem;
• data output subsystem.

Each of the given subsystems, in its turn, consists of separate plug-ins, each having a certain function. One can expand the capabilities of any subsystem with the help of additional plug-ins, including those developed by a user.

SVR package supports the following types of data presentation4:
• graphs;
• color plane;
• 3D voxel graphic – presentation of scalar data serving as functions of three variables, based on a unique technology of voxel graphic and semitransparent color palettes, by assigning a certain color from the available palette to the value of a quantity;
• isolines;
• isosurfaces;
• streamlines;
• vectors;
• various combinations of the listed visualization types.

SVR package allows to perform the following actions with the available data: gradient computation; first and second derivatives computation; divergence computation; circulation computation; rotation computation; determination of scalar quantity module; determination of vector quantity module; addition of constant vector; computation of components of vector quantity; integration of data serving as a function of one variable.

ScientificVR also allows to see sections of the process under research (one-dimensional – for 2D problems, one- and two-dimensional – for 3D problems), aligned along the coordinate axes or planes.

The obtained data presentation can be displayed in a few ways:
• on the screen;
• in the form of graphic files of BMP, TIFF, JPEG, EPS, PNG formats;
• in the form of animation files of AVI, MNG formats (rotation of a 3D picture of the process, dynamic progress of one-, two- and three-dimensional processes);
• printed out on a paper.

Besides that both graphic and animation files of data presentation for 3D processes can be obtained in full color stereo-format (ScientificVR supports the interlaced mode and page-flipping mode; the page-flipping mode is based on the quad-buffered OpenGL technology), which is one of the main features of the SVR visualizer.

SVR package allows visualizing data obtained with the help of various software products and referring to various branches of science and technology5. Figure 4 shows visualization of results of MR tomography of upper abdomen and retroperitoneal space (fig. 4a), where one can clearly see arteries of retroperitoneal space and kidneys, and a head (fig. 4b).
Figure 4: Tomographic data visualization

The data has been kindly provided by the Russian Cardiological Scientific Production Center, www.cardioweb.ru. Figure 5 shows visualization of ballistic damage in an encapsulated ceramic armor target, where one can clearly see damage morphology.

Figure 5: Damage morphology

The data, kindly provided by US Army Research Laboratory, is visualized using the semitransparent voxel technique.
3 POTENTIAL WAYS OF THE SVR DEVELOPMENT

GDT Software Group has proceeded to the implementation of the data visualization system (on the basis of the existing SVR package) that would allow using broad capabilities of parallel computing systems and would feature the above-enumerated capabilities. Preliminary scheme of the system under development is presented on fig. 6.

Developing this system, we are trying to make use of all the experience, acquired in the course of developing the hybrid version of the GasDynamicsTool package. Considerable efforts in developing this system are focused on ensuring the ease and reliability of its interaction with any user application, working on multiprocessor computing systems. This
system is due to become a user-friendly and easy-to-operate tool that will allow to implement “on-the-fly” dynamic visualization and, basically, monitor considerable amounts of data of random user software products with low time costs.

The system will be featuring dynamic visualization of data obtained from nodes of the distributed computing system when the problem is being solved by a user application, and will allow to efficiently use resources of parallel computing systems for visualization of huge amounts of data (100 GByte – 1 TByte and more) (with simultaneous display in various representations included).

Adjustment of the system to a certain user application will be accomplished with the help of special plug-ins, attached to communicators «Solver – SVR» and «User Application – SVR», and these plug-ins can be developed by the users themselves.

This system provides for the implementation of several approaches to visualization on distributed computing systems.1

3.1 Visualization on a dedicated node

With this approach data visualization is implemented on a dedicated node, which is a cluster node or a special graphical workstation (see fig. 7).

This approach allows to use computing resources entirely and solely for the computation process itself. However, the approach, just as it is, can be applied only when the amount of visualized data is relatively small.
The point is that if the amount of data exceeds the memory capacity of the visualization node (and in fact it happens even earlier), visualization will turn out to be impossible or (in case of virtual memory being used) inadmissibly slow. Besides that, processing of huge amounts of data will take quite a lot of time in itself. To ensure quick transfer of data from computational nodes to the visualizing one, one will also need a network circuit of high efficiency. Thus, it is advisable to use this approach only in consideration of the following changes.

A. The computational nodes, where the data are stored, receive certain general instructions on data processing, for example, instructions on sampling from a distributed array. Thus, the amount of data being transferred is considerably reduced, since not all the data but only a certain sample required for visualization is being transferred.

B. The data from nodes do not get transferred to the visualization node all at once but are being added by turn. Visualization is being accomplished in parts, and then the visualization node forms a general image. In this case the problem of memory overflow on the visualization node is excluded.

The modified variant of visualization on a dedicated node is presented on the fig. 3.

The suggested changes also bring about certain disadvantages. Thus, sampling generation results in computing resources being spent, which will lead to computation slowdown. Besides that the amount of data (even if preliminarily processed) might be rather huge, which will require a high-speed network hardware. When the data are being transferred by turn, a certain reduction in the processing power of computational nodes may take place since there is a kind of a waiting line for the data to be visualized. Here also arises the problem of meshing the received image into a single whole, which means this variant should be used only in case of data parallelism, while in case of general visualization the specified type of parallelism is missing (for instance, streamlines, isosurfaces, etc.).

However, one can minimize the majority of the drawbacks of the described approach by using a balanced combination of the specified modifications. For example, in case of large-scale resampling of initial data one can use sampling with a simultaneous processing on the visualization node, and in case of small-scale resampling (or the absence of such) and data parallelism – sampling with successive data transfer.

3.2 Visualization on computational nodes

With this approach the data are being visualized in parts, each portion on a separate computational node (see fig. 8), and then parts of an image are transferred to the visualization node, where they get merged into a single image. Thus, one does not have to waste time on resending huge amounts of data to the visualization node, which is done for an already received part of image only.
The problem of memory overflow on the visualization node is, thus, also excluded. Besides that one gets an opportunity to take into account physical and other peculiarities of the visualized processes while preparing an image on the computational node.

However, this approach is not devoid of drawbacks. Firstly, meshing an image into a single whole is generally quite a nontrivial problem which presents certain difficulties (with the exception of the already mentioned case of data parallelism). Secondly, although due to paralleling the visualization speed increases, the immediate computation process is slowing down. And, thirdly, when implementing visualization on the computational nodes, one has to face certain technical difficulties. This becomes particularly relevant if one uses special operating systems, their functionality being minimized in order to increase the computing efficiency.

Depending on the user hardware either the first or the second variant may be implemented. Thus, when dealing with computing systems with high-capacity hardware on the nodes but “slow” connection, it makes sense to use visualization on computational nodes. And, vice versa, for systems with quick connection and nodes with configuration mainly targeted at computation – it is better to use visualization on a visualization node (which makes it possible to install additional software and hardware only on a computer responsible for visualization).

### 3.3 Visualization on several dedicated nodes

This approach is most flexible and efficient, yet most difficult to implement. Here (see fig. 9) visualization is implemented on several visualization nodes (either constituting a part of a multiprocessor computational node, on which computation is being accomplished, or
serving as a separate graphics workstation).

The data from computational nodes are transferred to visualization nodes. Since the number of visualization nodes generally does not coincide with that of computational nodes, there is a possibility of some portions of data from certain computational nodes being transferred to one visualization node, and the rest of the data – to another visualization node. Distribution of the data flow is accomplished through the control node, which also receives parts of an image from visualization nodes, meshes them into a single whole and displays it as is specified by the user. Besides that in the absence of data parallelism, the control node sees to it that each visualization node received not only the data that are to be visualized but also any additional information (from other computational nodes) required for the correct “adapting” of the received part of an image to the results obtained from neighboring nodes.

Thus, this approach is devoid of the problems typical of the above described methods
(connected with a limited RAM capacity, a limited network capacity of data communications equipment, meshing parts of images into a single whole, computation slowdown, etc.). That is, this approach allows to most efficiently and flexibly use the capabilities of a multiprocessor computing system for the visualization purposes. However, it has a number of drawbacks that hamper the application process. Among those are difficulties in programming behavior of the control node, as well as in establishing a connection between computing and visualization nodes (because of the discrepancy between the number of computational nodes and that of visualization nodes). However, the authors consider this very approach to be the most prospective and promising one.

4 CONCLUSIONS

- A single-processor version of the scientific visualization package with a set of different types of visualization, some of them being unique, has been developed. Some of its capabilities, including those responsible for processing data of various user applications, are presented.
- A version of the specified package for the hybrid version of the GDT program, featuring “on-the-fly” dynamic visualization of data obtained from nodes of a distributed computing system, has been developed.
- A scheme of the system of data visualization on multiprocessor computing systems, which is now under development in the GDT Software Group, and a review of a number of approaches to paralleling data visualization are presented.

REFERENCES