NISVAS

Three-dimensional Interactive Visualization in Java3D

BRUNO CAIADO[1]
LUIS CORREIA[1]
JOÃO BRISSON LOPES [2]
[1]Lic. Engª. Informática e de Computadores, Instituto Superior Técnico, Av. Rovisco Pais, 1049-001, Lisboa, Portugal
(currently at Megamedia, R. António Pedro 111, 1º, 1150-045 Lisboa, Portugal)
{bruno.caiado, luis.correia}@megamedia.pt
[2]Departamento de Engenharia Informática, Instituto Superior Técnico, Av. Rovisco Pais, 1049-001 Lisboa, Portugal
brisson@ist.utl.pt

Abstract. This paper describes the design and implementation of NISVAS, an application for interactive visualization of large three-dimensional datasets produced by scientific applications that was developed in Java/Java3D. Data visualization with NISVAS maps dataset magnitudes into color, symbols and vectors. This paper presents the concepts and requirements for NISVAS graphical processing and user interface. The user interface is easy to use and learn, and supports different levels of user expertise. User reactions to NISVAS showed that all objectives were met, especially those relating to application performance and ease of use. Concluding remarks include directions for future development.

1 Three-dimensional Visualization

The analysis of results produced by the numerical simulation of complex phenomena is very difficult because of the large data volumes involved and the different magnitudes (e.g., deformation, stress, temperature, velocity and concentration fields) that must be simultaneously analyzed. The solution is to visualize this data, combining what is visible (e.g., a part that deforms) with what is not (e.g., stress).

The number of scalar or vector magnitudes to visualize can be very large. Therefore, engineers and scientists need simple, yet powerful, interfaces to quickly change the way a magnitude is represented or exchange one magnitude by another. Moreover, users need to identify very quickly the magnitudes he/she can manipulate and which operations can be performed on each of them. This sets special requirements to the user interface, namely the presentation of the visualization application status, operations available and error prevention.

Traditional visualization applications were specialized applications that depended on proprietary technologies, operating systems and graphics hardware. Such applications were developed for specific platforms that, in a general way, were very expensive. Platform dependencies also lead to windowing system dependency and specific look and feel. This is the case of visualization applications as Wavefront Data Visualizer, PV-WAVE or VIS5D. This was also the case of the ISVAS (Interactive System for Visual Analysis, Karlsson [1-2], Haase [3]) visualizer that depended on SGI hardware and the GL API under the 4Dwm windowing system for visualizations, and on Motif for the user interface.

The enormous increase of hardware capabilities, and the emergence of new software construction paradigms (e.g., Object Oriented design) and user interface design methods has changed this scenery. Today, these paradigms, together with programming languages and APIs like C++, OpenGL, Java and Java3D, allow the development of new visualization applications that are simpler to build and whose interfaces keep the same look and feel across platforms.

This paper presents an application for interactive visualization of large datasets that was developed as a graduation project of Information and Computers Engineering at Instituto Superior Técnico (Caiado et al [4]). The main objective was to make the existing ISVAS vizualizer portable and significantly improve its usability. The development of the new vizualizer, NISVAS (New Interactive System for Visual Analysis), was a demanding challenge because the project objective was to design and implement a simple to learn and use user interface and, at the same time, do it with a new API, Java3D, whose definition had yet to stabilize. Moreover, NISVAS code had to be written from scratch since ISVAS code, although available, had become cluttered from many patches and additions.

NISVAS usability was assessed by heuristic evaluation methods (Nielsen [10]) and by comparison with ISVAS.

2 Objectives

NISVAS design started with the analysis of existing scientific visualizers and user surveys that identified problems of existing visualizers. This way, the objectives for this project were:

- Portability
- User interface
- Performance
- Documentation and on-line help

2.1 Portability

The great majority of interactive visualizers is not portable and often relies on proprietary standards with heavy historical backgrounds that limit their dissemination and use. To avoid this pitfall, NISVAS had to be developed in such a way that at least its source code could be easily ported. This called for solutions based on tools like C++ and Java, and the OpenGL and Java3D APIs.

2.2 User Interface

The user interfaces of many visualization applications are non-intuitive, inconsistent and difficult to learn and use. For instance, the ISVAS user interface requires some user expertise with ISVAS, even for simple commands. It also presents many inconsistent user dialogues, with different layouts for similar situations. Users complained about the memory effort required to use the interface and that they are frequently lost in the interface. Another user complaint regarded the mandatory file loading order and the need to individually pick and load all files of a dataset.

These comments lead us to define as an objective that NISVAS user interface should rely on simple and consistent dialogues, should be easy to learn and use and should also provide an easy way to display the status of the application. The interface should provide shortcuts to commands (e.g., menu navigation and shortcut keys) and command redundancy so that both newcomers and expert users might use the interface. Lastly, tasks like loading a dataset should be performed with a single command.

2.3 Performance

Performance is a critical factor in visualization applications that must be kept in mind at all times, since performance depends heavily on the resources required by graphical data processing. This means that all design options, especially the architecture, should assess their consequences in terms of performance and that it was necessary to balance ease of implementation and usability of the user interface with performance.

2.4 Documentation and On-line Help

Most applications for interactive visualization come with large complicated manuals. At the same time, on-line help is either scarce or cryptic. NISVAS objective was to provide contextual on-line help and full documentation to ease and reduce the time to learn how to use NISVAS.

3 Architecture

NISVAS architecture has two main subsystems, the user interface and the graphical processing core, that communicate through communication modules. The user interface subsystem interacts with the user and passes user commands to the graphical processing subsystem after checking for errors (error prevention). The graphical processing subsystem executes all graphical commands, creates and destroys visualization windows and manages events on those windows.

The concrete architecture depended on the way communication between the two subsystems would be implemented. One solution would be to have the two subsystems communicating via sockets. This would ensure source code portability of a graphical processing core written in C++ (and OpenGL) and total portability of a Java coded user interface subsystem. This solution required a very clear definition of the communication protocol. However, socket programming is not easy to port. Moreover, tests showed that the expected intense communication between the two subsystems would degrade overall application performance.

The solution that was adopted for the architecture consisted on developing the two subsystems in Java (Arnold [5]), and on using the emerging Java3D API (Bouvier [6]) in the graphical processing core. Use of the Java3D API was a challenge, but it allowed an object oriented approach and total portability of the application. It would also allow easy functionality upgrade and simplification of the communication between the two subsystems. The only disadvantage was that Java programs must run under a Java virtual machine, which is slower than C or C++ compiled code.

With this solution, the graphical processing core is an object that graphically processes the datasets and presents a well-defined interface whose public methods are invoked by the user interface. Each dataset can be realized as an object that the graphical processing core manipulates.

NISVAS was developed on a Pentium III PC running at 500 MHz under Windows 98 and 128 MB of RAM.

The development tools used were JDK 1.2.2, Java3D 1.1.3, JavaHelp 1.1 and Forte for Java CE 1.0.

4 NISVAS Functionality

NISVAS operates on geometry-based datasets where scalar and vector magnitudes refer to geometries assembled from elements used by Finite Element methods. NISVAS supports several element geometries. Figure 1 shows some of these. Elements may be placed together to create a single object or as several separated objects, as figure 2 shows.

Datasets can be loaded and unloaded with a single user command. Loaded datasets can be copied and renamed. New magnitudes can be added to an existing dataset or removed from it.

Loading a dataset opens its visualization window and shows the dataset geometry under a uniform light source.

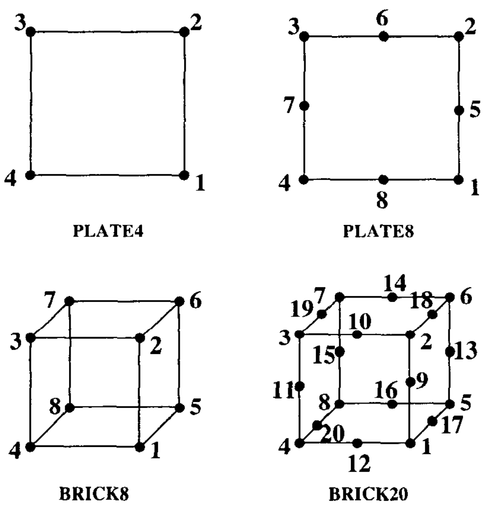


Figure 1 Finite Elements supported by NISVAS, showing node placement and numbering.

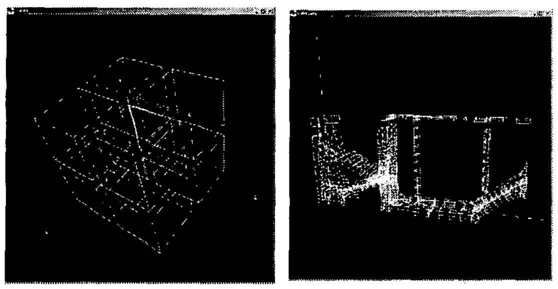


Figure 2 Two datasets geometries.

The user can then change object color or, through user selectable transfer functions and color scales, map scalar magnitudes to local color of the objects. Magnitudes can be mapped into symbols and vectors, their color and size, and vector direction in the case of vector magnitudes. Deformations can also be applied to the geometry of the objects.

NISVAS allows users to change illumination, make objects transparent (with transparency level control) and define cut planes that can be precisely positioned.

All information on a dataset is kept in a coherent way and it is possible to store dataset views. Loading a view loads its dataset and presents it with the magnitude mappings and the illumination that were set when the view was saved.

NISVAS uses Java3D behaviors. A behavior is a link between an event and an action. NISVAS uses Java3D predefined behaviors and implements new ones to perform object rotation, panning and zooming in visualization windows.

4.1 Mapping Magnitudes onto the Geometry

NISVAS maps scalar and vector magnitudes to object color, symbols and vectors. The user can also control the color of objects such as light sources and the background with color values specified in RGB or HSV.

Scalar magnitudes are mapped onto color by transforming magnitude ranges into 4 color scales: Physics (visible light color spectrum), Hueramp (from red to red through all saturated colors of the HSV color model), Grayscale (from black to white) and Hotiron (red to white through orange and yellow). Mapping vectors containing RGB color components can also set object color.

Vector magnitudes can be interpreted as node displacement to visualize objects that are submitted to deformations. Displacements can be scaled and can even be animated if several displacements with timing information are loaded into NISVAS.

Users control the visualization of the geometry and of NISVAS predefined global objects (coordinate axes, geometry bounding box, light sources and window background). Users can make these objects invisible to remove cluttering when too many objects, symbols or vectors are displayed. Dataset geometry is usually shown as oriented polygons, but the user can choose to display it in wireframe mode.

4.2 Symbols and vectors

Users can choose to display magnitudes as symbols and vectors, as figure 3 shows. Symbols can be twodimensional (circles, triangles or squares) or three-dimensional (spheres, octahedrons or cubes). Vectors can be displayed as simple lines, flat or smooth pyramids, and detailed vectors. Symbol and vector color can be set like geometry color. Symbol and vector size can be scaled according to scalar magnitudes in the datasets.

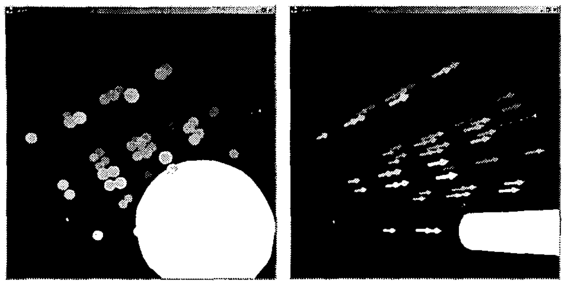


Figure 3 Display of a scalar magnitude with symbols and a vector magnitude with vectors.

4.3 Global Objects and Light Sources

NISVAS defines global objects (coordinate axes, geometry bounding box, labels, background color, cut planes and light sources) that can be visible or invisible.

In addition to the default ambient illumination, NISVAS allows users to control illumination by means of light sources whose location is predefined. These can be directional, point or spot light sources and their color and intensity can also be set. NISVAS also allows users to change the diffusiveness and opacity properties of the surfaces of the objects.

4.4 Visualization Options

NISVAS users can define and position cut planes. On top of this, NISVAS users can also select to see only the front or back faces of the objects, view objects on full screen mode and turn light sources off and on. By default, NISVAS uses perspective projection to display objects. However, the use can select parallel projection or selected views (front, side or top), either one at a time or simultaneously.

5 NISVAS User Interface

NISVAS user interface was carefully designed and implemented to meet requirements and make all the above functionality available to the user. Interface design was based on task analysis and prototype heuristic evaluation.

The goal was to maximize direct manipulation of the objects and provide a coherent and easy to learn user interface. Starting at the main window, users should be able to reach all dialogues within two clicks. There should not be more than 6 options at any one level (Mayhew [7], Preece [8]). There should also be several ways to reach a dialogue (redundancy) in order to support new or infrequent users as well as expert users. Commands should be easy to identify and should not required any memory effort.

At the same time, the user interface should present the application status in a clear and simple way, without too much detail that might confound the user. All application status details should be presented according to dialogue context.

NISVAS has two types of windows: the main window, from which all commands are available to users, and visualization windows, one per dataset. Figure 4 shows a screenshot with the main window on the left and two visualization windows.

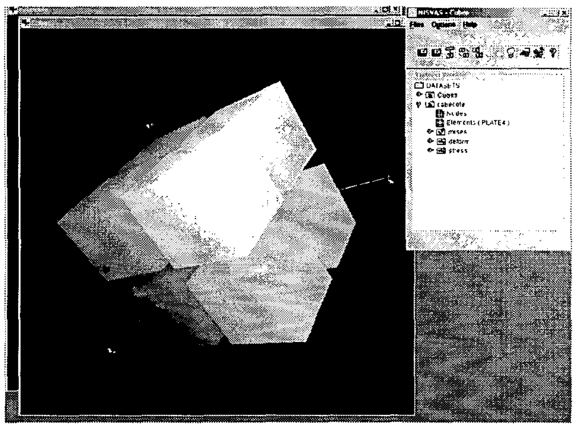


Figure 4 NISVAS main window (left) and two visualization windows.

5.1 Direct Manipulation in Visualization Windows

When NISVAS loads a dataset, it immediately opens the dataset visualization window and presents the dataset geometry with a uniform color. Users map dataset magnitudes by means of specialized dialogues.

Users can directly manipulate (Schneiderman [[9]) all objects in the visualization window. Moving the mouse with the left button pressed rotates the objects. Moving the mouse with the middle button pressed zooms in and out on the scene. Panning is achieved by moving the mouse with the right button pressed.

Moving the mouse in the opposite direction reverses the current action. The user can reverse all previous actions by pressing the left mouse button for a time without moving the mouse.

5.2 Application Status

The visualization windows and the main window show the general application status. Visualization windows impart on the user the current viewing transformation of the geometry and magnitude mappings that have been set, together with illumination and global objects settings.

NISVAS main window (see figure 5) has three areas: the menu bar, the tool bar and the explorer area. The explorer area shows the status of the application relative to the datasets that are currently loaded. At any one time there is one active dataset whose name is highlighted. Clicking on the name of a dataset in the explorer area makes it the currently active dataset.

By default, each dataset is shown in expanded mode (all magnitude names are shown) in the explorer area (dataset *cabecote*). Clicking on the - icon on the left of a dataset name collapses the dataset, while clicking on a + icon expands the dataset and shows all its magnitudes. Magnitudes that span several moments in time (e.g., the *mises*, *deform* and *stress* magnitudes of the *cabecote* dataset) can be expanded or collapsed in the same way.

5.3 Redundancy and User Adaptation

NISVAS provides redundancy of dialogue selection to support users of different expertise levels. NISVAS user interface presents up to 5 different ways to select a dialogue. In general, a dialogue can be selected using:

- The menu bar
- The tool bar
- PopUp menus in the explorer area
- Menu navigation keys
- Shortcut keys

To unload a dataset one can use NISVAS menu (Files, Unload Dataset), the tool bar (second icon from the left), a PopUp menu (right mouse button click on the dataset name in the explorer area, followed by clicking on Unload Dataset), menu navigation keys (Alt+F, U) or a shortcut key (CTRL-U). An inexperienced user can select a dialogue through a step-by-step procedure with the menu, while an expert user can do the same either through the tool bar or a shortcut key.

5.4 Dialogues

The main menu organization in just two levels reflects the task analysis that was carried out during NISVAS design, which defined a task hierarchy organized in 3 categories:

- Files – dataset and views loading and unloading, exit from the application.

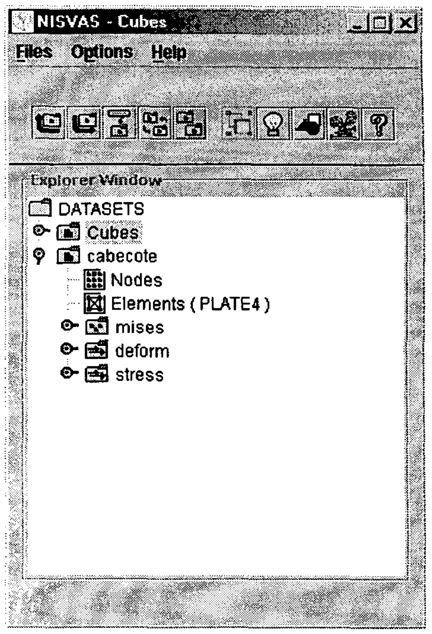


Figure 5 NISVAS main window showing the menu bar (top), the tool bar (middle) and the explorer area (bottom).

- Options – magnitude mapping, and light sources, viewing options and animation control.
- Help – access to on-line help and documentation (includes index, an introduction, how to use NISVAS and dataset formats).

Figure 6 shows all options available from each of the menu groups (users can also access the Help menu group by clicking on the rightmost icon on the tool bar). Note the small number of options in each menu group.

The Options menu group is the most complex menu group. Task analysis showed that there were 4 types of tasks (Mapping, Lights, Viewing Options and Animations) and that each task's dialogue could be very simple, as figures 7 to 10 show. All dialogues show a Close button on the left and a Help button on the right. The Help button provides contextual help.

5.5 Files Mapping Dialogue

The Mapping dialogue is the most important dialogue of NISVAS user interface since this dialogue controls how magnitudes are mapped into color, symbols, vectors and displacements. This dialogue also controls the presentation of global objects (geometry bounding box, etc). There are 4 groups of objects: geometry, symbols, vectors and global objects. Figures 11 and 12 show the dialogues for two of these groups.

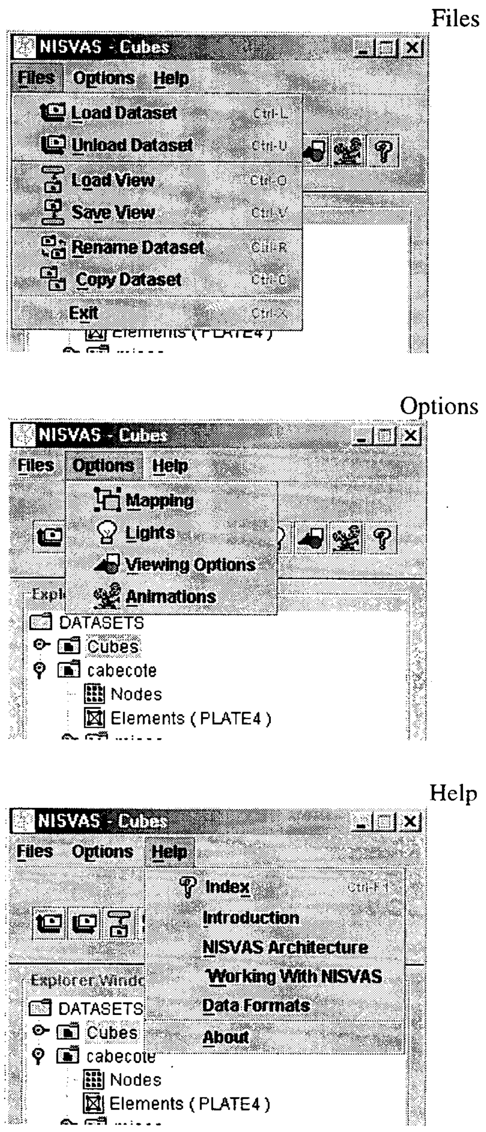


Figure 6 Options available from the menu groups.

The Mapping dialogue is as a Java Tabbed Panel that allows users to identify all available options very quickly since all options are simultaneously presented. Table 1 shows how Mapping options are organized.

Group	Object	Symbol	Vector	Global Objects
Section	Color	Color	Color	Bounding Box
	Deformation	Size	Length	Labels
	Details	Details	Details	Coordinate Axis
				Background Color

Table 1 Mapping dialogue hierarchy.

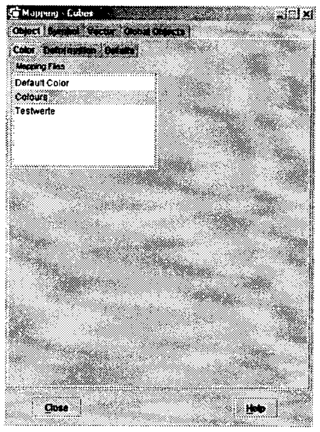
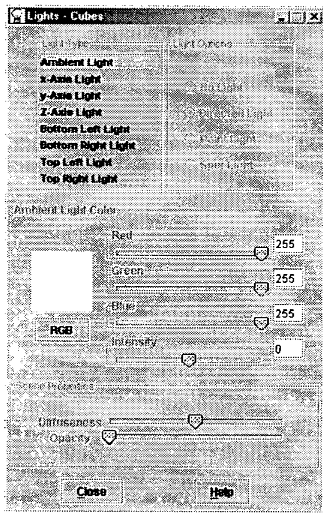
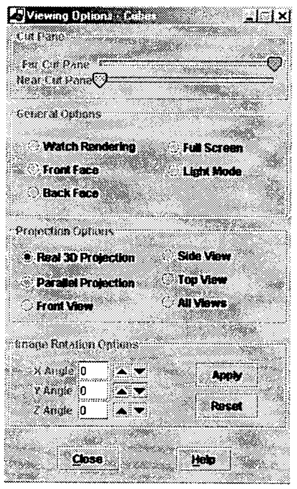


Figure 7 Tabbed panel of the Mapping dialogue.



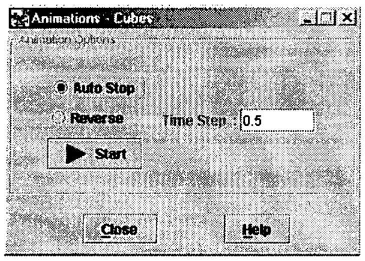
Red 255
Green 255
Blue 255
Intensity 0

Figure 8 Lights (light sources) control dialogue.



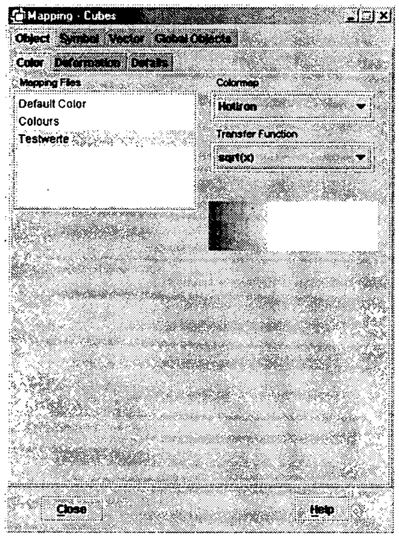
X Angle 0
Y Angle 0
Z Angle 0

Figure 9 Viewing Options control dialogue.



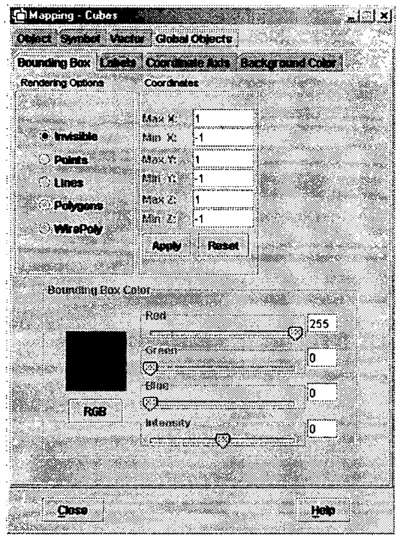
[✓ Auto Stop] Reverse
Time Step: 0.5

Figure 10 Animations control dialogue.



Colormap: Hotiron
Transfer Function: sqrt(x)

Figure 11 Mapping dialogue to map magnitudes onto the geometry of the objects.



Rendering Options: [✓ Invisible] Points Lines Polygons WirePoly

Coordinates

Max X:	1
Min X:	-1
Max Y:	1
Min Y:	-1
Max Z:	1
Min Z:	-1

Bounding Box Color

Red	255
Green	0
Blue	0
Intensity	0

Figure 12 Dialogue to control NISVA global objects.

6 Example

Figure 13 shows the visualization of the deformation of a railway carriage part that protects carriages against strong impacts. The geometry of the part is made from 1452 nodes assembled as type PLATE4 elements (see figure 1). The dataset includes deformation, local stress and the local values of the von Mises criterion at 21 different times during part deformation under impact.

The visualization was animated. Figure 13 shows part deformation (scaled on purpose) at 2 times in the process. Wireframe visualization mode was used to avoid screen cluttering. Figure 14 shows the part as a solid deformed object with the local stress modulus mapped into the color of the geometry.

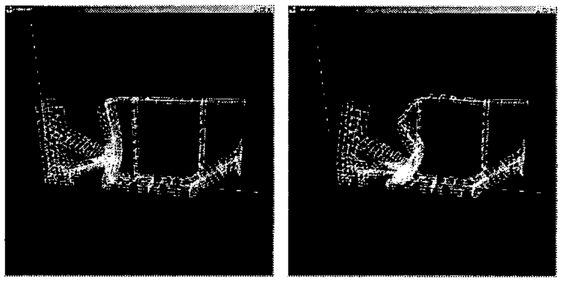


Figure 13 Visualization of a part deformation under a violent impact at 2 times in the process.

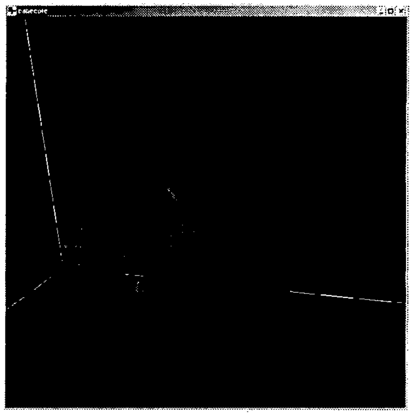


Figure 14 Part deformation with local stress modulus represented by local color.

7 Concluding Remarks

The development of the NISVAS visualizer for three-dimensional visualization of large datasets generated by numerical simulation of complex phenomena required special care in the design of the architecture, choice of development tools and, above all, in the design and realization of the user interface. The project faced several challenges. One was the use of Java3D technology that was the object of several upgrades and corrections during the project's lifetime. Parts of the source code had to be rewritten several times in order to use new functionality and dispose of code that had been written to workaround problems of earlier versions of Java and Java3D.

However, it was the design and realization of the user interface that constituted the most difficult challenge since the success of applications such as NISVAS resides more on the ease of use of the user interface than on application functionality. The design methodology based on task analysis and the evaluation of prototypes that was adopted was essential for the excellent results that were achieved.

One of the most important results is the simplified command hierarchy that was achieved without sacrificing functionality. The simplification of the user interface allows users to have a clear view of all the commands available and to invoke commands in several ways (redundancy and adaptation to the user). This allows users to make the most of NISVAS powerful commands through a user interface that is easy to use and learn, without the need to build complex mental models of the application.

Heuristic evaluation of the application showed user positive reactions to NISVAS and confirmed that all the objectives set at the beginning of the project were met. Users stressed the functionality available, the ease of use of the user interface and the good performance of NISVAS.

7.1 Future Developments

User reactions to NISVAS show that the solutions that were implemented constitute a sound base to continue NISVAS development. This will be the subject of more detailed and formal usability studies to be carried out in the next development phases, as new functionality is added.

In response to users request, NISVAS will be able to load datasets using data formats other than the current format used by NISVAS. Moreover, ISVAS will also be able to receive on-line data from simulation applications.

Future work will also address the upgrade of NISVAS animation capabilities and the introduction of functionality to visualize stationary or time-dependent flows of particle systems (e.g., atmospheric, river and sea flows with particle transport or virtual wind tunnels).

Another upgrade that is planned is the possibility to manipulate magnitudes of the datasets using the pocket calculator metaphor. This way, users will be able to, e.g., combine scalar magnitudes into vector magnitudes and apply transformations to magnitudes.

8 References

[1] K. Karlsson, ISVAS 3.1 Interactive System for Visual Analysis User's Guide, Fraunhofer Institute for Computer Graphics, 1993.

[2] K. Karlsson, Ein Interaktives System zur Visuellen Analyse von Simulationsergebnissen, PhD diss., THD, Darmstadt, 1994.

[3] H. Haase et al, ISVAS 3.2 Interactive System for Visual Analysis User's Guide, Fraunhofer Institute for Computer Graphics, 1995.

[4] B. Caiado, L. Correia, Visualization of Large Volume Datasets (in Port. Visualização de Dados de Grande Volume, Trabalho Final de Curso), Graduation in Information and Computers Engineering Final Project, Instituto Superior Técnico, 2001.

[5] K. Arnold, J. Gosling, The Java Programming Language, Addison-Wesley, 1996.

[6] D. J. Bouvier, Getting Started with Java3D API, Sun MicroSystems Inc., 1999.

[7] D. J. Mayhew, Principles and Guidelines in Software User Interface Design, Prentice Hall, 1992.

[8] J. Preece, Human Computer Interaction, Addison-Wesley, 1994.

[9] B. Shneiderman, Designing the User Interface, Addison-Wesley Longman, 1998.

[10] J. Nielsen, R.L. Mack, Usability Inspection Methods, John Wiley and Sons, New York, 1994.