DOCUMENT CONTROL SHEET

	ORIGINATOR'S REF. NLR TP 96446 U		SECURITY CLASS. Unclassified

ORIGINATOR
National Aerospace Laboratory NLR, Amsterdam, The Netherlands

TITLE
A real-time visualization system for computational fluid dynamics

PUBLISHED IN
NEC Research & Development, Vol. 37, No. 1, pp. 114-123, January 1996.

AUTHORS S. Doi, T. Takei, Y. Akiba, K. Muramatsu, H. Matsumoto, L.J.C. van Rijn, B.C. Schultheiss	DATE 960715	pp ref 14 5

DESCRIPTORS

Computer networks	Numerical flow visualization
Data compression	Real time operation
Graphical user interface	Software tools
Image processing	Supercomputers
NEC computers	Three dimensional flow

ABSTRACT
We have developed a real-time visualization system for CFD (Computational Fluid Dynamics) in a network computing environment. This system, named RVSLIB (Real-time Visualization and Steering Library), makes it possible to visualize the results of a CFD simulation on a client simultaneously while it is being computed on a server (tracking). The GUI (Graphical User Interface) of the RVSLIB system also enables the user to specify parameters of the calculation and visualization graphically. The system generates pixel image data on the server, and compresses the data by JPEG (Joint Photographic Experts Group). Therefore, the amount of data transferred from the server to the client is relatively small. The system is built on standard X Window System#, OSF/Motif#, and socket-based communications, and therefore has high portability.

NLR TECHNICAL PUBLICATION
TP 96446 U

A REAL-TIME VISUALIZATION SYSTEM FOR COMPUTATIONAL FLUID DYNAMICS

by

S. Doi*, T. Takei**, Y. Akiba***, K. Muramatsu*,
H. Matsumoto****, L.C.J. van Rijn and B.C. Schultheiss

Published in: NEC Research & Development, Vol. 37, No. 1, pp. 114-123, January 1996.

* C&C Research Laboratories
** Application Software Division
*** Manufacturing Industries System Development Division
**** NEC Informatec Systems, Ltd.

Division	: Informatics	Completed	: 960715
Prepared	: LCJvR [illegible] 15/8 BCS/ [illegible] 15/8	Order number	: 249.519
Approved	: RJPG [illegible] 16/8	Typ.	: MM

Contents

(14 pages in total)

This page is intentionally left blank.

A Real-Time Visualization System for Computational Fluid Dynamics

By Shun DOI,* Toshifumi TAKEI,† Yukinori AKIBA,‡ Kazuhiro MURAMATSU,*✢
Hideki MATSUMOTO,§ Laurens C. J. van RIJN‖ and Bert C. SCHULTHEISS‖

ABSTRACT We have developed a real-time visualization system for CFD (Computational Fluid Dynamics) in a network computing environment. This system, named RVSLIB (Real-time Visualization and Steering Library), makes it possible to visualize the results of a CFD simulation on a client simultaneously while it is being computed on a server (tracking). The GUI (Graphical User Interface) of the RVSLIB system also enables the user to specify parameters of the calculation and visualization graphically. The system generates pixel image data on the server, and compresses the data by JPEG (Joint Photographic Experts Group). Therefore, the amount of data transferred from the server to the client is relatively small. High performance visualization is achieved by effectively vectorizing the visualization module on a supercomputer. The system is built on standard X Window System#, OSF/Motif#, and socket-based communications, and therefore has high portability.

KEYWORDS **Real-time visualization, Tracking, Steering, Computational Fluid Dynamics (CFD), Network computing environment, Pixel image data, Image data compression**

1. INTRODUCTION

The speedup of computers and the spread of network computing environments, have resulted in increased demands for systems that visualize numerical results on a user client simultaneously while it is being computed on a computational server.

A system satisfying this requirement is, for example, Visual3 developed at MIT [1]. Visual3 generates graphical objects as polygonal data on the server (mapping process), and sends the data to the client. Rendering of the data is performed by the client on a graphics workstation such as IRIS# Series, using a general-purpose graphics library such as OpenGL#. Since the rendering process can be performed by an existing graphics library, this approach has the merit of easy implementation of the system. However, the disadvantage of this approach is that the amount of polygonal data transferred to the client increases when a computational grid becomes large; the amount of data to be transferred becomes a bottleneck for the process speed. Also, the portability is low due to the dependency on the graphics library.

On the other hand, systems which carry out the rendering process on parallel computers have been reported at NASA Langley and MIT [2,3]. However, these systems have not been equipped with a GUI (Graphical User Interface) by which users are able to change visualization parameters while looking at output images.

In this context, we have developed a real-time visualization system for CFD (Computational Fluid Dynamics), named RVSLIB (Real-time Visualization and Steering Library), based on NaS/RVS (Navier-Stokes Real-time Visualization System) [4,5]. This system has been designed to have the following characteristics.

1. Mapping and rendering are both performed by the server; pixel image data is directly generated by the server. Moreover, we attain high performance in the pixel image generation process on

*C&C Research Laboratories
†Application Software Division
‡Manufacturing Industries System Development Division
§NEC Informatec Systems, Ltd.
‖National Aerospace Laboratory NLR of the Netherlands
✢Current address: Japan Atomic Energy Research Institute
#X Window System is a trademark of X Consortium, Inc. and OSF/Motif is a trademark of Open Software Foundation, Inc., respectively. IRIS and OpenGL are registered trademarks of Silicon Graphics, Inc.

vector computers by paying attention to the vectorization of visualization operations.
2. The system is not based on any graphics library, and attains high portability.
3. A library style is adopted, enabling users to easily incorporate the system into their CFD codes.
4. Pixel data generated by the server is compressed using JPEG (Joint Photographic Experts Group), and the compressed data is sent to the client. On the client, the data is expanded and displayed in X Window System. This data compression further decreases the amount of data transferred over the network.
5. VBI (Visualization Based Input) functionality is supported. VBI makes it possible to control visualization parameters graphically.
6. In order to realize a quick system response for VBI, an additional simplified rendering process is implemented on the client.
7. Compressed pixel data can be saved on a hard disk, and can be replayed as off-line visualization.

This paper describes the configuration and the characteristics of RVSLIB. Section 2 describes the characteristics of three different strategies on distributing the visualization operations to the server and the client. In Section 3, system overview and its basic requirements are described. Section 4 describes the system configuration, the action modes and the visualization module. In Section 5, the library style is described. Section 6 describes the GUI. Some concluding remarks are presented in Section 7.

2. DISTRIBUTION OF VISUALIZATION OPERATIONS TO SERVER AND CLIENT

In RVSLIB, the CFD computation and visualization operations are performed on the server. This section will discuss the merits of this strategy.

In general, real-time visualization for 3D time-dependent CFD computation has the following characteristics:

1. The access frequencies to computational results are low.
   · In time: Only one image (or less) will be generated per time step.
   · In space: For 2D contour and vector images, the grid points used in visualization are restricted to a small portion of the original 3D grid space.
2. The portion of the grid referred to by visualization operations changes dynamically.
   · Example: Tracer visualization.
3. For CFD visualization, polygon-based rendering is not always efficient.
   · Example: Contour visualization.

Graphics workstations equipped with dedicated hardware for high-speed drawing are widely used for standard post-processing. In such post-processing, the following characteristics are observed.

· A number of different images are generated from a single data set. Therefore, the data transfer between the computation and visualition platforms will not be a bottleneck.
· High-speed drawing of polygons is especially suited to visualization of CAD objects.

From these fundamental differences between real-time visualization and post-processing, it is clear that using graphics workstations is not necessarily an efficient way to realize a real-time visualization system. In the following, we will discuss the characteristics of three different strategies on visualization task assignment.

Here, two different platforms are considered; the server and the client. We classify the whole computational process into three sub-processes: CFD computation, mapping operation, and rendering operation. Here, the "mapping operation" indicates the generation of graphical objects such as polygons and vectors from the computational results (physical values defined on grid points). Outputs of the mapping operation are graphics commands including graphical objects. The "rendering operation" includes projection, lighting, hidden surface operation, etc. The output of the rendering operation is pixel image data. Now there are three strategies for the distribution of the CFD computation, mapping and rendering operations.

Strategy A

CFD computation is assigned to the server, while the mapping and rendering operations are assigned to the client. Data to be transferred from the server to the client are computational results.

Strategy B

CFD computation and mapping operation are assigned to the server, while rendering operation is assigned to the client. Transferred data are graphics commands which are mainly composed of graphical objects.

Strategy C

CFD computation, mapping and rendering operations are assigned to the server, while the client is used only for displaying pixel data. Transferred data are pixel images.

Merits and demerits of the individual strategies are summarized as follows.

Strategy A
A merit of this strategy is that the server can be dedicated to CFD computation (the most time-consuming part). However, it is necessary to reduce the amount of data to be transferred from the server to the client by, for example, limiting grid points which are candidates for visualization, since the computational results are huge in size. In case of tracer visualization, this visualization needs to be assigned to the server as an exception, since the data portion referred to by tracer computation changes dynamically as the computation goes on. Since the volume rendering essentially requires all computational results, this visualization operation also needs to be performed on the server. These observations imply that it is practically necessary to distribute the mapping operations both to the server and to the client, which may make the interface complicated. Another drawback is that the performance of the client may become a bottleneck in the mapping operation process (which implies inefficient use of the client).

Strategy B
In this case, the interface between the server and the client is clear and simple, since all mapping operations will be performed on the server. The mapping operation will not become a bottleneck if the operation is well vectorized on the server. A merit is that this strategy will effectively use the graphics hardware on the client. A possibly serious problem in this strategy may be that, as pointed out above, polygon-based rendering is not always suited for CFD visualization; it may result in too much polygon data to be transferred to the client.

Strategy C
This strategy retains high portability, since the system does not depend on specific graphics hardware. The amount of transferred data depends only on the resolution on the screen (the number of pixels and bits in color table), and does not depend on the amount of computational results. Also, the amount of transferred data is small compared to Strategy B. Since all mapping and rendering operations must be done by software, we need to introduce high-speed computation algorithms which are suited for server's architecture.

Table I summarizes the above consideration.

3. SYSTEM OVERVIEW AND BASIC REQUIREMENTS

3.1 System Overview

The basic concept of RVSLIB may be described as follows. RVSLIB enables users to visualize the results of their CFD simulation programs on a

Table I Distribution of CFD computation, mapping and rendering operations.

Strategy	A	B	C
Server	CFD computation, Part of mapping operation (tracer, volume rendering)	CFD computation, Mapping operation	CFD computation, Mapping operation, Rendering operation
Client	Graphics operation, Rendering operation	Rendering operation	(Pixel output)
Data transfer	Computational results	Graphical commands	Pixel image data
Merits	Server can be dedicated to CFD	Efficient use of graphics hardware	Portability
Demerits	CPU load of mapping and rendering operations. Distribution of mapping and rendering operations. Data transfer amount.	Vectorization of mapping operations. Amount of graphical objects.	Algorithms for mapping and rendering operations.

workstation (or PC) simultaneously while it is being computed on a High Performance Computing Server (HPCS) in a network computing environment. Also, users can easily show a movie of the simulation off-line, by compressing and conserving the image data during the simulation run. This animation can be used as an alternative to the conventional analog video. The basic requirements of RVSLIB are described in the following subsection.

3.2 Basic Requirements

Ease of use

It is required that RVSLIB can be accessed by user's code. A library style format, where users will perform some functional calls, is hence a preferable interface. The library style makes it possible to realize the real-time visualization by only calling a few subroutines. In RVSLIB, the following three routines are considered to be the basic subroutines.

- Initialization routine: This routine initializes RVSLIB parameters when the RVSLIB module is started up.
- Main routine: This routine generates pixel image data and displays it. It will be called at every time step in the time marching calculation.
- Finalization routine: This routine terminates the RVSLIB module.

In addition to these routines, several other routines are being prepared for data transfer.

Portability

Users may intend to use their desktop computers as the client. Their computers are not always equipped with specific graphics hardware or a graphics library. Therefore, the client system has to satisfy the following requirements.

- The system does not need a specific graphics library.
- The system should be constructed using FORTRAN77, C, and standard UNIX* environment products (X Window System, OSF/Motif, and sockets), so that it can be implemented easily on different hardware platforms.

The same requirements are applicable to the server system.

Performance

The CPU load for visualization operations and the communication load between the server and the client have to be small compared to the calculation load for the CFD code. To achieve this objective, a vectorized algorithm for generating image data has been developed for RVSLIB.

Real time

Users may want to use real-time visualization for the following objectives:

- Interactive and graphical debugging of a program which is still in the development phase.
- Users can easily check the validity of parameters for CFD calculation after the calculation has been started.
- During production runs, the trial-and-error time for the optimization of parameters can be reduced by changing these parameters in the middle of the calculation.

User friendliness

It shall be possible to easily and interactively change visualization parameters, which prescribe visualization conditions, by using a GUI. Therefore, it is needed that users can determine visualization parameters by trial-and-error after the calculation is interrupted. For this purpose, the system supports three modes depending on the status of the CFD computation and the visualization operations.

- Real-time mode: The calculation and the visualization are carried out simultaneously.
- Halt solver mode: In order to adjust visualization parameters, the calculation will be interrupted, and only the visualization module will be carried on.
- Halt visualizer mode: If users no longer need real-time visualization, the visualization module will be interrupted, and only the CFD calculation will be carried on.

Generality

Users' programs are generally aimed at the analysis of various fields (fluid, heat, structure,

*UNIX is a registered trademark in the United States and other countries, licensed exclusively through X/Open Company Limited.

electromagnetic field and their coupling problems, etc.). The programs may also adopt various data formats. Although RVSLIB has been developed mainly for CFD, it is applicable to other fields.

Adjustment to a network computing environment

Effective use of resources is an indispensable condition for applications that run in a distributed computing environment. It is necessary that RVSLIB should not become a bottleneck on the network, of which the bandwidth may be narrow. RVSLIB minimizes the communication overhead of data transfer by using the image data compression technique.

Facility to realize animation

Animations of calculated results have been made by means of an analog video so far. Since a frame-by-frame recording of the video is generally a time-consuming task, an alternative to reduce this task is desired. The spread of high performance image data compression techniques, such as Motion JPEG and MPEG (Moving Picture Experts Group), has made it possible to play back a movie easily on a workstation or PC. In the near future, speedup of CPU will make this digital image processing much faster and cheaper. In such an environment, animation using the compressed digital image data will replace the conventional analog video systems. In RVSLIB, transferred data can be stored in files and be reused for an off-line digital video.

4. SYSTEM CONFIGURATION

4.1 General Configuration

The general configuration of RVSLIB is shown in Fig. 1. This system is composed of the library and User-Interface (UIF) modules. The library module is called from the users' CFD code, and runs on the server. The UIF module runs on the client connected to the server via TCP/IP (Transmission Control Protocol/Internet Protocol). Although a high performance vector supercomputer is desirable as server, any UNIX server can be used. On the client workstation, received image data is expanded (if compressed on the server) and drawn using X-calls.

The library module carries out the visualization operations, and communicates with the UIF module using socket communication. The module controls the CFD calculation and the visualization operations. Image data generation by the library module will be performed at every time integration step. The UIF module, which is activated by the library module, displays the GUI. The GUI is built using standard X Window System and OSF/Motif, and controls CFD and visualization parameters.

In Fig. 2, the visualization module configuration is shown. The visualization module is activated by library module calls from the CFD program. This module generates individual pixel image data for solid objects, contours, vectors, particle tracers, volume rendering as well as any combination of these. Individual image generation is performed with reference to computational results, color table, view-port, light source, etc. Users can change the pixel image size, depending on their resolution requirements (the system default is 512×512).

Pixel image generation is performed basically in the following three steps.

Step 1

Individual image generation is performed if the corresponding switches are "on." The outputs are pixel image (RGB) data and Z-buffer (depth) data.

Step 2

Hidden surface operation is performed with reference to Z-buffer data, and a single-image plane is generated from the multiple image planes generated in Step 1.

Step 3

If specified, data compression will be performed. The image data will then be sent to the client via socket communication.

The visualization module attains high portability, since it is written in FORTRAN77 and C.

4.2 System Action

RVSLIB has the following three action modes.

Real-time mode

The CFD calculation and visualization are carried out simultaneously. It is possible to visualize the results of user's CFD simulation simultaneously as it is being computed (tracking). The library module sends the pixel image data to the UIF module using socket communication, and controls the CFD calculation and the visualization operations. Users can change the parameters of the calculation and visualization in the middle of the calculation (steering).

Halt solver mode

Calculation is interrupted, and only visualization is carried out. This mode will be used for investigating

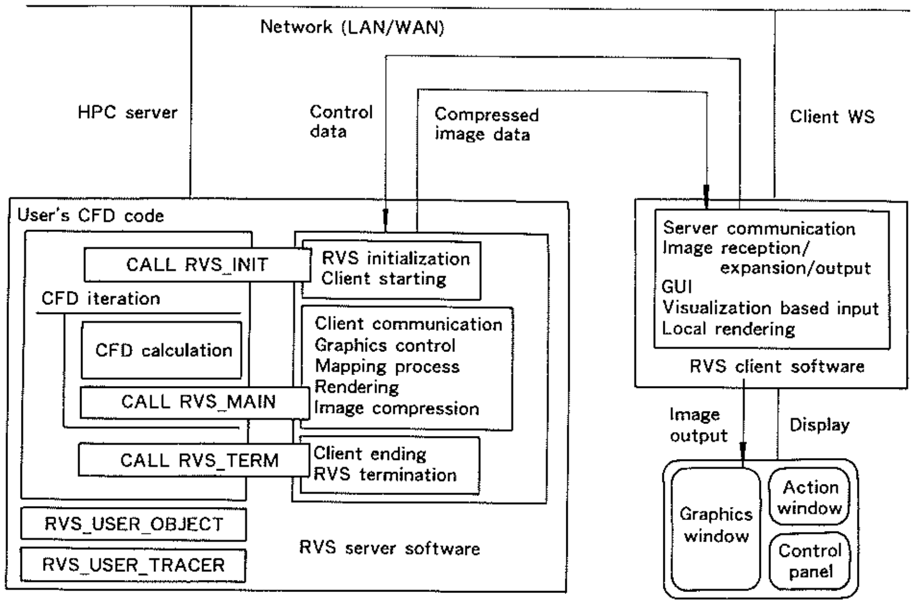


Fig. 1 General configuration of RVSLIB.

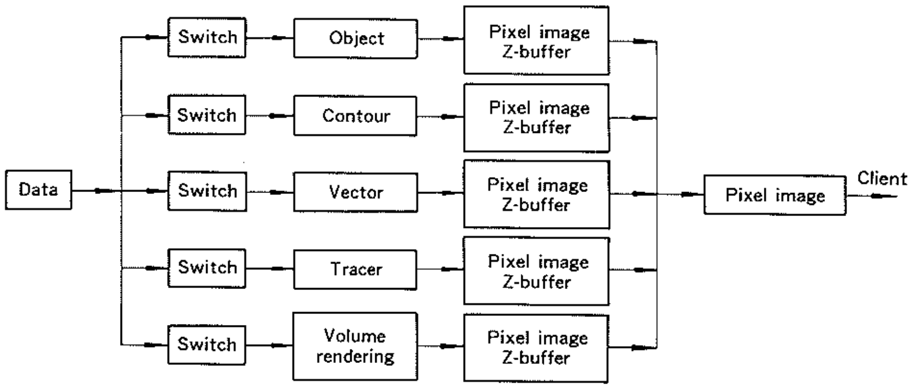


Fig. 2 Visualization module configuration.

the CFD calculation results for one time step in detail.

Halt visualizer mode

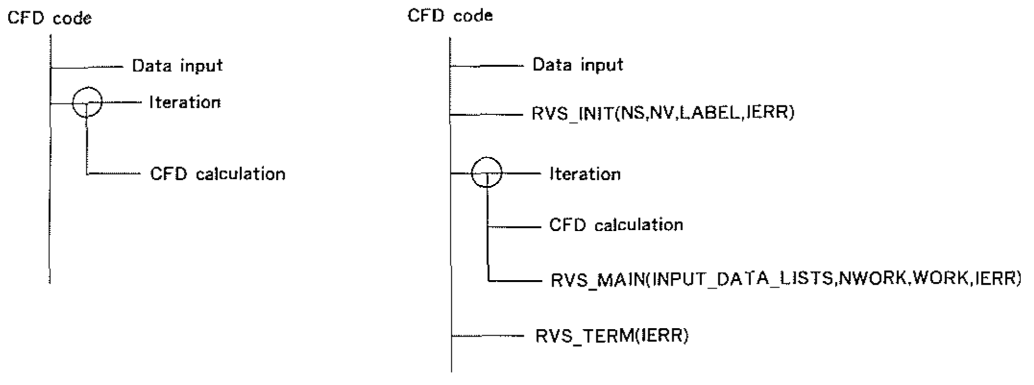
Visualization will be interrupted, and only the CFD calculation will be carried on. This mode will be used if there is no need for real-time visualization.

5. LIBRARY STYLE

RVSLIB has adopted a library style format so that users can easily call this real-time visualization functionality from their code. There are three control routines, namely initialization, main and finalization routines. Figure 3 shows how these routines should be called.

RVS_INIT

Initialization of the library and UIF modules: opening menu and graphics windows, setting of intrinsic valuables to standard values, and initialization of socket communication. This routine will be called just before a CFD iterative calculation. This routine has the following arguments.



(a) Before library calls. (b) After library calls.

Fig. 3 Positions of library routine calls.

· NS (input):	The number of scalar data types
· NV (input):	The number of vector data types
· LABEL (input):	Physical names of scalar and vector data types
· IERR (output):	Error indicator

RVS_MAIN

Visualization of calculation results and control of the visualization operations. This routine will be called just after the calculation of a time step. In the halt solver mode, the server CPU will be assigned only for this routine. In the halt visualizer mode, this routine processes nothing.

· INPUT_DATA_LISTS (input):	Computational results and some related information corresponding to grid points and boundary.
· NWORK (input):	The size of the work area.
· WORK (work):	Work area, required in particular for vectorization. This area is also used to store permanent parameters used in direct image generation.
· IERR (output):	Error indicator.

RVS_TERM

Termination of the library and UIF modules: closing menu and graphics windows, and finalization of socket communication. This routine is called just after the CFD iteration loop.

· IERR (output): Error indicator.

RVS_LOAD

Input of visualization parameters from files.

RVS_SAVE

Output of visualization parameters to files.

In addition to these control routines, there are some user-supplied routines for additional information transfer between the library and the users' code. There are four user-supplied routines in RVSLIB.

RVS_USER_INIT

Activation of the UIF and pixel image data relay modules.

RVS_USER_OBJECT

Supply of the object data.

RVS_USER_TRACER

Supply of the position data of cut-off plane on a boundary-fitted coordinate grid system for tracer display.

RVS_USER_TIME

Supply of the time interval.

To use RVSLIB, users should do the following. First, the users should make a library call to RVS_INIT, RVS_MAIN and RVS_TERM in their program. Second, the users should program the user-supplied routines depending on their need. Then they can compile their program, and link the generated objects and the library.

6. GRAPHICAL USER INTERFACE

RVSLIB has a GUI based on OSF/Motif for use on X terminals and workstations that support

X Window System. By using the GUI, users can visualize the results of a CFD simulation while it is being computed. They may change the parameters of visualization during the calculation by filling out parameter popup forms.

Figure 4 shows a windows image displayed by RVSLIB. A shot in an incompressible flow simulation around a cylinder is shown in the drawing area. Figure 5 shows the hierarchical structure of the GUI menus.

A brief explanation of the GUI operation will be given.

1. Specifications of common visualization parameters
   Users can specify color mode, color table, background color, view-port information, and light source as common visualization parameters.
2. Setup of visualization tools
   Users can set up rendering, tracer, contour, object, and vector tools by specifying parameters in popup forms.
3. Visualization based interaction
   Users can manipulate the graphical objects in the drawing area by pointing (e.g. with the mouse) at the graphics output. In this way, graphical objects can be rotated, translated, and resized. Some visualization parameters can also be specified in the same way.
4. Parameter file
   Users can save visualization parameters in files with the help of the GUI. These parameter files are used to replay the settings of the visualizer. The behavior of the visualization process can be controlled automatically as a parameter file function.

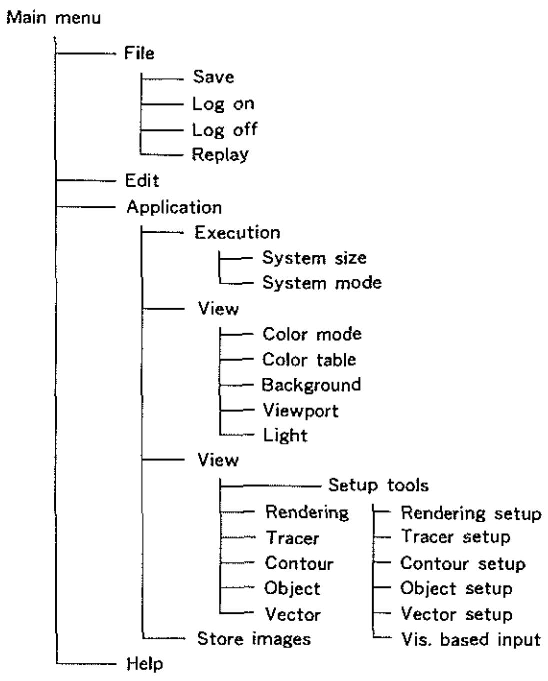
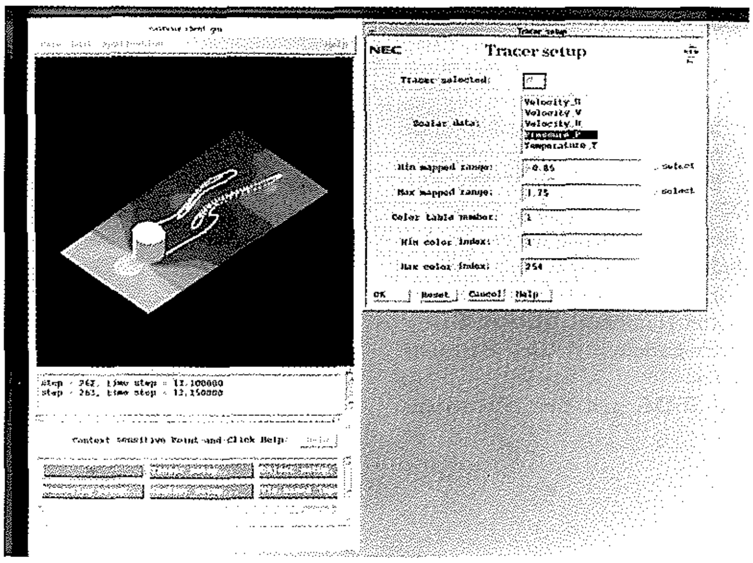


Fig. 5 Hierarchical structure of GUI menu.


Tracer setup

NEC Tracer setup

Tracer selected: ✓

Scalar data: Velocity_U, Velocity_V, Velocity_W, [Pressure_P], Temperature_T

Min mapped range: -0.85 select

Max mapped range: 1.75 select

Color table number: 1

Min color index: 1

Max color index: 254

OK Reset Cancel Help

Step - 262, time step - 13.100000
Step - 263, time step - 13.150000

Context sensitive Point and Click Help:

Fig. 4 Example of real-time visualization and windows image.

5. Storing image data in files
   Users can store displayed images in files.

7. CONCLUSION

This paper has introduced RVSLIB (Real-time Visualization and Steering Library) developed by NEC and NLR (National Aerospace Laboratory NLR of the Netherlands). This system makes it possible to visualize the results of CFD simulation simultaneously while it is being computed (tracking), and to control the parameters of calculation and visualization in the middle of the calculation from a client workstation (steering). RVSLIB can be used by simply incorporating some subroutine calls into user's code.

With the advent of low-cost and high performance vector supercomputers, it becomes possible for individual users to utilize an enormous computer capability in full for his or her own purposes. In such a "personal" supercomputing environment, real-time visualization and steering of on-going calculation will be a natural way of carrying out numerical simulations. Design and development of RVSLIB has been started from this quite natural desire of users. This system is expected to provide user-friendly environment for CFD researchers if it is used with NEC SX-4 supercomputers.

ACKNOWLEDGMENTS

Professor Kunio Kuwahara of the Institute of Space and Astronautical Science pointed out the importance of a real-time visualization environment for CFD computation, thus providing the driving force for this development.

REFERENCES

[1] R. Haimes, et al., "Visual3: Interactive Unsteady Unstructured 3D Visualization," *AIAA Pap. 91-0794*, 1991.
[2] R. Haimes, "pV3: A Distributed Systems for Large-Scale Unsteady CFD Visualization," *AIAA Pap. 94-0321*, 1994.
[3] T. W. Crockett, "Design Consideration for Parallel Graphics Libraries," *NASA Contractor Rep. 194935*, 1994.
[4] S. Doi, et al., "A Real-Time Visualization System for Computational Fluid Dynamics," *Proc. 26th ISATA – Dedicated Conf. Supercomput. Applications in Automotive Industries*, Aachen, Germany, Paper No. 93SC042, 1993.
[5] T. Takei, et al., "A Real-Time Visualization System for Computational Fluid Dynamics," *Proc. 7th CFD Symp.*, pp. 701-704, Tokyo, Japan, 1993. (in Japanese)

Received October 20, 1995

* * * * * * * * * * * * * * *

Shun DOI received his B.E. and M.E. degrees in precision engineering from Hokkaido University in 1979 and 1981, respectively. He also received the Ph.D. degree from the same University in 1984 for his research on parallel computation techniques for finite element analysis. He joined the C&C Systems Research Laboratories, NEC Corporation in 1984. He was a visiting scholar at INRIA (Institut National de Recherche en Informatique et en Automatique, France) from 1988 to 1989, conducting research on parallel numerical methods. Currently, he is supervising the Numerical Analysis Group in Computer System Research Laboratory, C&C Research Laboratories, NEC Corporation.

Dr. Doi is a member of SIAM, and the Information Processing Society of Japan.

* * *

Toshifumi TAKEI received his B.S. and M.S. degrees in physics from Kyoto University in 1985 and 1987, respectively. He joined NEC Corporation in 1987, and is now a member of the Scientific Software Department, Application Software Division. He is engaged in the development of computational fluid dynamics software and its real-time visualization system.

Mr. Takei is a member of the Information Processing Society of Japan.

* * *

Yukinori AKIBA received his B.E. and M.E. degrees in electric engineering from Hokkaido University in 1984, and 1986 respectively. He joined NEC Corporation in 1986, where he was engaged in the research and development of CFD software at the C&C Information Technology Research Laboratories. He is now Assistant Manager of the Manufacturing Industries System Development Division, and is engaged in technical support to industrial customers of NEC SX supercomputers.

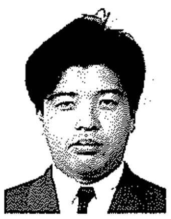
Mr. Akiba is a member of the Information Processing Society of Japan, the Japan Society of Fluid Dynamics, and the Japan Society for Industrial and Applied Mathematics.

*   *   *

Kazuhiro MURAMATSU received his B.S., M.S. and D.S. degrees in physics from University of Tsukuba in 1982, 1984 and 1987, respectively. He joined NEC Corporation in 1992, and was Assistant Manager of the Computer System Research Laboratory, C&C Research Laboratories until 1995. He is now Research Scientist of the Center for Promotion of Computational Science and Engineering, Japan Atomic Energy Research Institute. He is engaged in the research and development of parallel computational fluid dynamics and its real-time visualization system.

Dr. Muramatsu is a member of the Physical Society of Japan, the Information Processing Society of Japan, and the Japan Society for Industrial and Applied Mathematics.

*   *   *


Hideki MATSUMOTO received his B.S. degree in chemistry from Tohoku University in 1988. He joined NEC Scientific Information System Development, Ltd. in 1988, and now belongs to the Scientific & Engineering Analysis Department, NEC Informatec Systems, Ltd. He is engaged in the development of computer simulation systems on UNIX and OSF/Motif user interfaces on X Window System.

Laurens C. J. van RIJN received his ingenieurs degree in informatics from University of Twente in 1990. He joined the Dutch National Aerospace Laboratory NLR in 1991, and is a research engineer at the Informatics Division. He is now Project Manager of the NLR project that is engaged with the research and development of graphical user interfaces for control of real-time visualization applications.

*   *   *

Bert C. SCHULTHEISS received his ingenieurs degree in informatics from University of Twente in 1991, and completed the post graduate course 'the design of technical information systems' at University of Twente in 1993. He subsequently joined the Dutch National Aerospace Laboratory NLR, and is now a research engineer at the Informatics Division. He is engaged in the research and development of graphical user interfaces for control of real-time visualization applications.

* * * * * * * * * * * * * * *