# COVISE in the CUBE:
# An Environment for Analyzing Large and Complex Simulation Data

**Dirk Rantzau, Karin Frank, Ulrich Lang, Daniela Rainer, Uwe Wössner**

Scientific Visualization Department
High Performance Computing Center (RUS/HLRS)
University of Stuttgart,
Allmandring 30, 70550 Stuttgart, Germany
{rantzau,frank,lang,rainer,woessner}@hlrs.de

Keywords: Projection based virtual environments, scientific visualization, high performance computing, computational steering, surface simplification techniques

## Abstract

In this paper we present a projection based virtual reality system that tries to meet the demands of scientific data analysis in a distributed high performance computing environment. After discussing the requirements and design issues for such a system in terms of flexibility and handling of large data, we give an overview about the hardware environment installed at our high performance computing center. Then we present the VR software system developed to meet the described requirements and finally provide some examples of currently investigated applications in the area of thermal comfort analysis, water turbine design and laser hardening process simulation.

## 1. Introduction

As iteration cycles in product development become shorter and shorter, the early availability of reliable data resulting from large scale simulations can be crucial for an efficient development process in industrial areas such as automotive and aerospace industry. Here, large amounts of scientific data coming from structural mechanics or computational fluid dynamics simulations running on supercomputers have to be handled and analyzed in relatively short time. Especially the analysis time is growing with respect to raw simulation time considering complex time dependent computational fluid dynamics data sets e.g. generated by a combustion engine simulation. VR based techniques for the exploration of those 'virtual prototypes' seem to be very promising [1] here, one early example is the virtual windtunnel [2]. However, the tools currently available are often very much tailored to the specific needs of one specific application case and are not easily adaptable or dynamically configurable. Another problem with currently available VR systems is that those systems simply don't fit into a given hardware infrastructure including supercomputers like vector or MPP systems, often secured behind firewalls.

From experience with typical end users at our center we found the following requirements that finally led to the development of the software environment described in this paper:

- The ability to access remote supercomputing resources from within the virtual environment,
- The possibility to do interactive post-processing and simulation steering,
- Being able to return to the desktop for further in depth analysis like creating detailed quantitative 2D plots (a must for engineers).

The possibility of collaboration is another important issue we are considering. Since development teams in large companies such as in the aerospace industry are often working at several locations, means have to be provided to allow consultations or group discussions over

the network or e.g. allow to synchronously view the results of the latest simulation. For these reasons we have developed a modular virtual reality environment called COVER [3] which has been integrated in the distributed and collaborative simulation and visualization environment called COVISE [4]. COVISE is a long term development effort in our group over the last five years. The integrated VR part COVER has now been ported to our recently installed CAVE [5] like back projection system called the CUBE [6], which is primarily used for the analysis of scientific data resulting from simulations performed by academic and industrial researchers and engineers using the HPC facilities at our computing center.

Our CUBE is a classical four side back projection system .The floor size of the CUBE is 2800 x 2800mm, the height of the front, left and right wall is 2500mm. The whole construction is made of wood, minimizing the influence on the magnetic tracking systems used. The video resolution currently used is 992 x 992 pixels on the floor and 1024 x 916 on the three walls, all driven at 114Hz frequency. A picture of the CUBE can be seen in figure 1.

Figure 1: The four side CUBE projection system.

The CUBE is connected to a Silicon Graphics Onyx2 double rack system with 14 R10000 CPUs and 4GB of main memory. The Onyx2 is equipped with three InfiniteReality pipes, each equipped with two raster managers. A video crossbar connects the graphics pipes with the four Electrohome 8500 projectors and up to six monitor displays. The machine can be configured in three alternative modes: in a triple keyboard mode with separate monitors allowing three different people to use the machine for applications and development, in CUBE mode, using two pipes via the multi channel option for driving the four projectors, and the third pipe as user interface and console for controlling the application, or, alternatively using three pipes to drive three projectors (2 walls and floor) for achieving higher performance. The network connection to our supercomputing platforms is established via HIPPI, OC3-ATM, FDDI and fast ethernet. A third rack with a striped fiber channel based disk array delivers the I/O performance needed to load big pre-computed data sets. For interacting in the CUBE we support several magnetic tracking systems (currently Polhemus Fastrak with Stylus pen and the Ascension Ethernet-Motionstar with 3D mouse).

## 3. The Software Environment for the CUBE

### 3.1. COVISE

COVISE is a software environment which tries to integrate visualization and simulation tasks across heterogeneous hardware platforms in a seamless manner. It has been optimized especially for efficient network transfer and high performance computing environments [7]. The user interface is based on the visual programming paradigm as used also in other visualization packages such as e.g. AVS [8] or SCIRun [9]. Distributed applications can be built by combining modules (modeled as processes) from different application categories on different hosts to form more or less complex module networks. At the end of such networks usually the rendering step does the final visualization. This application building step is done in the mapeditor module, the central user interface and visual application builder of COVISE. Session management for adding new hosts and synchronizing the tasks in the module network is done by a central controller which has the only knowledge about the whole application topology. The data management and efficient network transfer including conversion is done by request brokers (crb's) which are running on each host in a session. The crb's

provide also services like searching for files in remote directories or sending back information about available modules on the selected user account. In COVISE the notion of data objects is used instead of relying on a pure data flow paradigm. Therefore the underlying data management takes care of assigning system-wide unique names to data generated during a session. On a single host a shared data space (SDS) is used the exchange of data objects between locally running modules to minimize copying overhead. On most platforms this is realized as shared memory communication. A special feature of COVISE is that it allows several users to work in a collaborative way which e.g. allows to provide online consulting to end users at remote sites. Therefore every user interaction is sent to every partner in a collaborative session. Each user has a copy of the user interface and renderer process running locally (unlike e.g. in shared X tools). When geometry objects have been generated they are duplicated into the SDS on every host running a renderer module. After this, scene modifications initiated by a partner like rotating an object or camera zooming result only in short synchronization messages between the renderer modules (forwarded by the controller).

### 3.2. Building a VR Application

For the end user, the VR part of COVISE is just another 'renderer' which has been plugged into the module network. Behind the scenes however the VR part COVER extends the normal renderer functionality: first it partly maps the 2D user interface into the 3D space and second it provides additional (intuitive) ways of interacting with the data by establishing a feedback to the/some COVISE application modules that have been placed on the mapeditor. A typical screen snapshot of the mapeditor user interface can be seen in figure 2. In the top right corner the 2D control panel for adjusting module input parameters is located.

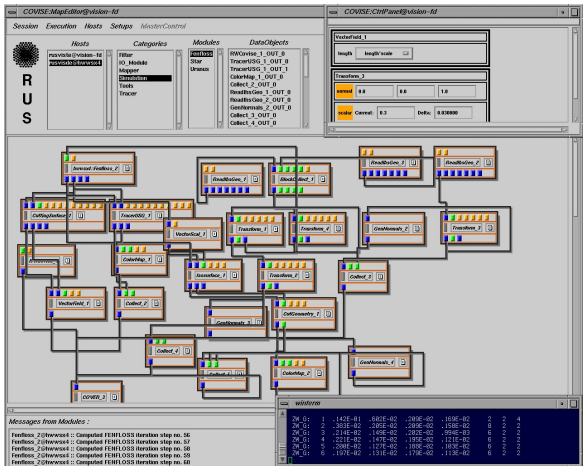

Figure 2: The mapeditor application builder

The feedback link back to the running application modules has been implemented by using the COVISE message protocol. This protocol allows e.g. sending back new parameters such as a new starting position of a particle trace in a flow field, without the need for relevant modifications in the application module itself. The module doesn't care whether a modified input was triggered by an input in the 2D control panel or by triggering an interactor in 3D space. Therefore, the creator module of some data object, e.g. a cutting plane, attaches some special feedback attributes to the data: the name of the module, its host, and of course the parameters to be modifiable in VR. If this data object finally is communicated to the VR part for rendering, that information is extracted and used to create the appropriate 3D user interface parts such as 3D menu entries and the interaction metaphors to be used with the VR devices. In figure 3 a snapshot of the currently used 3D menus can be seen. In the upper area of the right menu widget the generic functionality like scaling or switching to fly instead of pick mode is placed. Depending on the module pipeline, the rest of the menu is generated on the fly, in this case the single "CuttingSurface 1" entry at the bottom. Thus, to a certain degree the VR user interface generated by this mechanism replaces the traditional 2D GUI components available in the mapeditor control panel in the desktop environment.

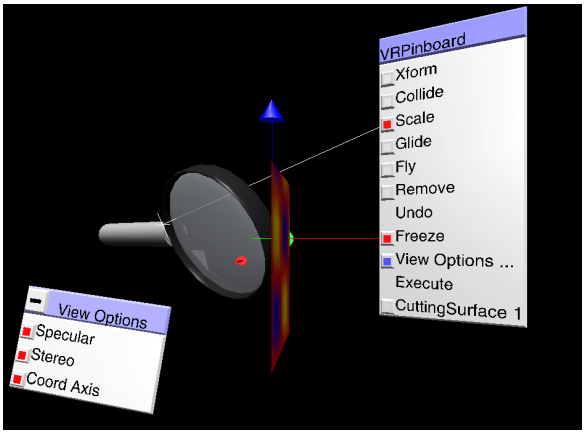


Figure 3: 3D menus in COVER: widgets can be moved around and placed arbitrarily. The magnifying glass represents an active scale interaction.

### 3.3. Handling of Large Data

For the handling of large data we apply two main strategies. First we try to distribute the application modules in such a way that we can use supercomputers not only for the simulation itself but also for the post-processing steps and thus reduce the amount of data being actually sent to the graphics system over fast network. Second, we use fast data reduction techniques in order to further reduce the size of geometry such as boundary surfaces or iso-surfaces generated from modules using e.g. the marching cubes algorithm.

For the first case we simply use the COVISE user interface to distribute the tasks on the desired hosts. For large multi block (or multiple time step) data we can also serialize the work in that we process one block or time step after the other, calculate the isosurfaces, cutting planes, particle traces etc. and then reduce the resulting geometry before we finally collect all the blocks together in the rendering step and hopefully end up with a small enough amount of data that we can still display in real-time.

For the reduction of surfaces in a scene, we developed a data-dependent simplification algorithm. Existing surface simplification algorithms use purely geometric criteria to reduce the number of triangles contained in a surface. This approach can cause significant loss of important information, if other data components are attached to the vertices of the surface. In scientific visualization, this is rather common, e.g. for cutting surfaces through simulation data. Therefore, we developed a method [10] which combines geometric criteria with data-dependent criteria to select the vertices to be removed from the surface.

The simplification algorithm belongs to the class of geometry removal methods. They are characterized by removing iteratively one geometry element (a vertex, an edge, or a triangle) at a time, possibly after performing some topology-preserving tests. In our case, we chose vertex removal as basic operation since this approach insures that the vertex set of the simplified surface is a subset of the vertices in the original surface, thus avoiding interpolation problems for the vertex-attached data components. The order of the vertices to be removed is defined by a priority queue. The iteration stops if a user-defined percentage is attained, or the priority queue is empty. Essentially, the data dependency is enclosed in the weight assigned to a vertex, i.e. in the priority criterion.

To take care about the geometric properties of the surface as well as about the changes in the data components we employ a combined priority criterion:

$$\text{weight}(v) = \propto \text{curv}(v) + (1-\propto)\,\text{grad}(v)$$

Here, curv($v$) is the discrete curvature of the surface in the vertex $v$, and grad($v$) can be interpreted as the maximum of the norm of the directional derivatives of the data, taken over all edges starting at $v$. The user-defined parameter $\propto \in [0,1]$ serves to adapt the weight function to the actual geometric complexity of the surface. For details see [11], where we compared several curvature and gradient heuristics with respect to its efficiency and reliability.

Experiments show that the theoretical O(N log N) complexity of the algorithm (with N being the number of vertices in the original surface) leads to almost linear performance in practice. We hope to increase the performance in future by improving the re-triangulation procedure which has to be involved in each iteration to fill the hole resulting from the vertex removal.

Figure 4 shows the result of our reduction technique applied to an iso-surface in a thermal comfort analysis of a car cabin, described below in more detail. The iso-value corresponds to the air temperature inside the car cabin, whereas the coloring of the isosurface corresponds to the pressure component.

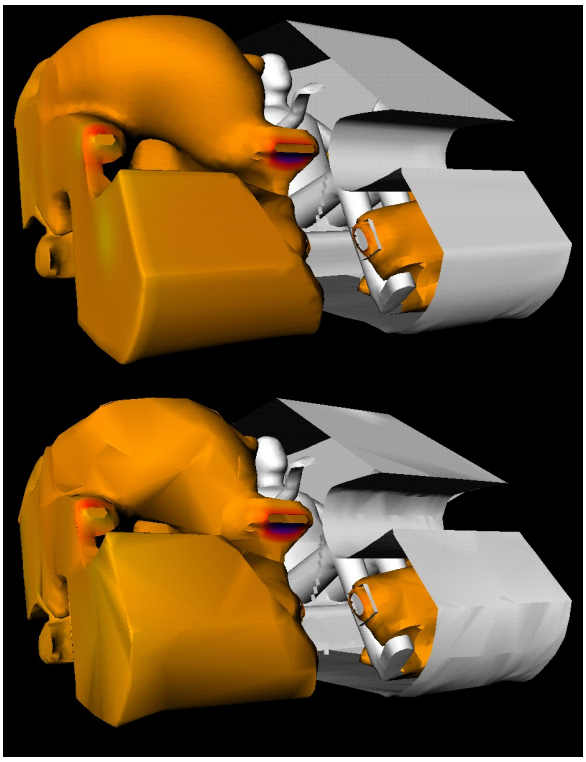

Figure 4: Data dependent reduction applied to an isosurface (left side, 50160 triangles) and the boundary surface (right side, 29104 triangles) inside a car cabin.
Top: original, bottom: reduced to 25% of original number of vertices.

## 4. Applications

In the following, we will describe three different scenarios to which the described environment has been applied.

### 4.1. Thermal Comfort Analysis

The first example is derived from the work done in the collaborative research center 'Rapid Prototyping' established at the University of Stuttgart together with the Daimler Benz car company. The simulation was done with STAR-CD, a commercial package developed by Computational Dynamics. The resulting data provided by Daimler Benz has been loaded into the VR system for analyzing the thermal comfort in an Mercedes Benz car cabin. We configured the application to allow the setting of arbitrary cutting planes through the unstructured grid, placing iso-surfaces and initiating particle traces to find areas of high velocity or optimize the air flow around the drivers head. Using our data reduction module on the original data set (180000 grid cells) we easily achieve frame rates in the range of 20Hz and higher using the CUBE running in the four side setup and stereo mode.

The setting of a cutting plane can be seen in figure 5. On the right hand side the 3D menu shows the possible interactions. When using the Stylus pen the user selects a menu entry by intersecting a beam which is visually attached to the pen movement with one of the menu entries. When selecting the "CuttingSurface" entry the beam changes its shape to a half transparent plane which can be now freely positioned inside the car cabin. Similarly the user can select the "TracerUSG" entry which results in a small sphere which can be used for setting starting points as input for the particle tracer module.

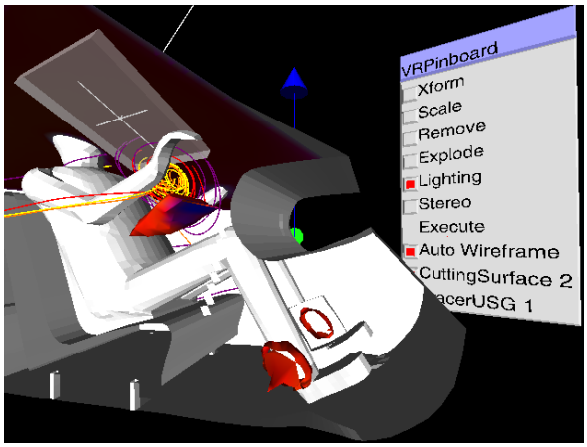


Figure 5: Interactive analysis of the thermal comfort in a car cabin

### 4.2. Water Turbine Optimization

The second example shows how computational steering of an ongoing simulation in the area of computational fluid dynamics has been realized. The institute for hydraulic machinery (IHS) at the University of Stuttgart is developing a code called FENFLOSS, which can be used to simulate the behavior of water turbines in order to increase the efficiency of such systems e.g.

for usage in power plants. The optimal geometrical design of the components is driven by many correlated parameters. We have integrated the FENFLOSS simulation code into the VR environment for varying the boundary conditions like the velocity of the incoming water stream. The FENFLOSS code itself is running on our NEC SX-4/32 supercomputer connected via an OC3 ATM connection to our Onyx2 driven CUBE.

The simulation code, which has been written in FORTRAN, has been modified by inserting a small set of calls (data exchange, parameter input, pause/resume functionality etc.) using the COVISE application API. The COVISE communication library (written in C++) is then linked together with the application. From the data generated by a simulation step, particle traces and iso-surfaces of zero velocity areas (regions of interest) are generated by the appropriate COVISE modules. The user is able to interact with the simulation by altering the angle of the turbine blades. The interactor we designed for this module presents a model of the turbine blades to the end user. By using this interactor the user can change the angle of incoming water stream by simply grabbing and rotating one of the displayed blades, which is a fairly intuitive interaction for the task to be fulfilled: the new angle is immediately communicated to a module which computes the new updated boundary conditions for the simulation. Although the recalculation of the flow currently takes some time in the order of 5-10 seconds, the user gets a good impression of the parameter influence on the resulting flow field.

### 4.3 Laser Hardening Simulation

The third example results from work done together in collaboration with the institute for laser treatment (IFSW), again in the collaborative research center "Rapid Prototyping". In the area of metal surface treatment it is often desirable to harden distinct parts of the surface with high power lasers. The results depend not only on the material properties but also on laser beam characteristics. To optimize the laser hardening process a simulation code called DIABLO has been developed at the IFSW. We have ported to and optimized the code for the vector supercomputer Cray C90 and recently the NEC SX4/32 in order to reach nearly interactive speed in the calculation. The simulation itself is a two step process. First the FIDAP package is used to calculate the temperature distribution inside the material. After this, DIABLO calculates the actual hardening values. We've used the available user subroutines in FIDAP to send data over to DIABLO via simple TCP/IP socket communication. With the source code available for modification, DIABLO has been fully integrated into the COVISE data management which connects the supercomputer to the virtual environment.

Figure 6 shows the data from a metal tube. The cutting plane interactor is used to position a slice through the tube in the wireframe model in order to find the regions with critical hardening properties. For interactive steering, parameter changes such as material properties can be adjusted. We also established a distributed COVISE session between the partners at the computing center and the IFSW for discussing the results of the computation in a collaborative way. At IFSW the desktop version of COVER was used, which provides a simple mouse interface for interacting with the 3D scene.

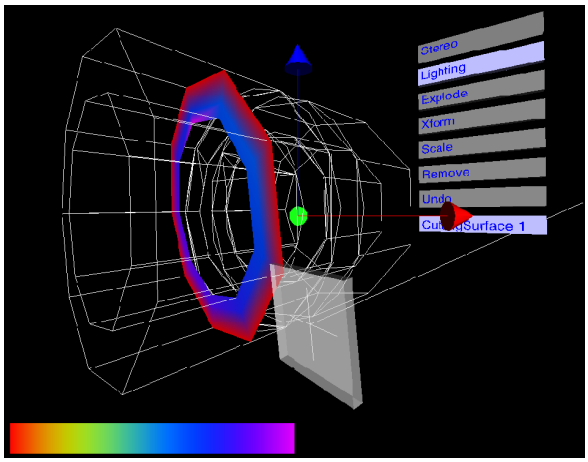


Figure 6: Results from the online laser hardening process simulation. The slice shows the hardening distribution inside the material.

## 5. Conclusion

We have presented a modular virtual environment for simulation steering running in a projection based VR system. The visual application builder allows an easy adaptation to different scenarios. The system presented is especially suited for distributed applications running in an HPC hardware environment. For the handling of large data sets, data reduction techniques have been integrated in the visualization system.

In the future we are planning to work on new interaction techniques applied to applications currently under preparation. We are also working on a MPI version of COVISE running on different parallel systems such as our Cray T3E to further push the overall system performance. We also consider the integration of other simulation packages and grid generators into the VR system.

## Acknowledgments

We would like to thank the whole COVISE team at RUS: Reiner Beller, Paul Benoelken, Lars Frenzel, Ruth Lang, Franz Maurer, Harald Nebel, Joerg Rodemann, Andreas Werner and Andreas Wierse; our colleagues at the IHS: Ralf Eisinger, Dr. Albert Ruprecht and Prof. Eberhard Goede; Julian Sigel at IFSW; our colleagues at Daimler Benz: Harald Echtle and Dr. Franz Klimetzek. During the work on this subject the first author was granted by the Deutsche Forschungsgemeinschaft in the project SFB374 , “Rapid Prototyping”. The second author was supported by European Community within the ESPRIT project INDEX (EP22745). The car cabin data set could be used by courtesy of Daimler Benz AG, who is our partner in both projects.

## References

[1] T. Fruehauf, F. Dai, Scientific Visualization and Virtual Prototyping in the Product Development Process, in: Virtual Environments and Scientific Visualization ‘96, Springer, Vienna, 1996.

[2] S. Bryson and S. Johan, The Virtual Windtunnel: An Environment for the Exploration of Three-dimensional Unsteady Flows, in: Proceedings Visualization ‘91, (IEEE Computer Society Press, Los Alamitos, 1991).

[3] D. Rantzau and U. Lang, A Scalable Virtual Environment for Large Scale Scientfic Data Analysis, in: Proceedings Of the Euro-VR Mini Conference, 10-11 Nov 97, Amsterdam, Elsevier, 1998 (to appear).

[4] A. Wierse, U. Lang, R. Rühle, Architectures of Distributed Visualization System and their Enhancements, in: Proceedings Fourth Eurographics Workshop on Visualization in Scientific Computing, Abingdon, UK, 1993.

[5] C. Cruz-Neira, D.J. Sandin and T.A. DeFanti, Surround screen projection based virtual reality: The design and implementation of the CAVE, in: Proceedings ACM SIGGRAPH ’93, p. 135-142, ACM SIGGRAPH, 1993

[6] CUBE Information in the World Wide Web: http://www.hlrs.de/structure/organisation/vis/vrlab.

[7] A. Wierse, Performance of the COVISE Visualization System under different conditions, in: Proceedings SPIE ‘95, Visual Data Exploration and Analysis, (SPIE 2410, San Jose, 1995).

[8] C. Upson et al, The Application Visualization System: A Computational Environment for Scientific Visualization, IEEE Computer Graphics and Applications, Vol. 9, No. 7, 1989.

[9] S. G. Parker and C. R. Johnson, SCIRun: A Scientific Environment for Computational Steering, in: Proceedings of the SC ’95, ACM Press, 1995.

[10] K. Frank and U. Lang, Data-Dependent Surface Simplification, in: Proceedings of the 9th Eurographics Workshop on Visualization in Scientific Computing, Blaubeuren, Germany, April 1998 (to appear).

[11] K. Frank and U. Lang, Curvature and Gradient Approximation in Data-Dependent Surface Simplification, submitted to IEEE Visualization ’98, North Carolina, October 1998.