OpenSG Symposium (2003)
D. Reiners (Editor)

Parallel Stereo Visualization For Clusters With OpenInventor: A Case Study For The Automotive Industry

Fernando Vega[*†], Gerd Sußner[*†], Thomas Reuding[•‡] and Günther Greiner[*]

[*]Computer Graphics Group
University of Erlangen-Nuremberg, Germany
[•]BMW AG

Abstract

Stereo visualization is an area which can greatly benefit from cluster computing due to the parallelizable nature of the rendering task. In order to implement this idea, we have developed a novel software architecture which allows the construction of parallel OpenInventor-based stereo applications. As a result of this work, we present the OpenInventor Stereo Library for Clusters. The library provides tools to port transparently OpenInventor applications to stereo cluster-based OpenInventor applications. The distribution of the rendering tasks is encapsulated, and the developer does not have to take care of non-graphics-related tasks. Evaluation was carried out on a prototype cluster consisting of a master and two slave rendering Linux PCs. The resulting pair of stereo images was visualized with polarization-filter projector and glasses. A standard X desktop *is available and multiple OpenInventor based windowed applications can be used simultaneously. The value of the present work was demonstrated with an example 3D-visualization application for the automotive industry.*

Categories and Subject Descriptors (according to ACM CCS): I.3.2 [Computer Graphics]: Graphics SystemsDistributed Network Graphics; I.3.2 [Computer Graphics]: Three Dimensional Graphics and RealismVirtual Reality; C.2.4 [Computer-Communication Networks]: Distributed SystemsClient/Server;

1. Introduction

Application of 3D stereo visualization is a must in the Computed Aided Design (**CAD**) process within the industry nowadays. Visual correctness of a new car body model must be evaluated during its design phase and while standard 3D visualization provides some depth cues [5], the designer does not get a complete impression of how the model is going to look as a finished product. This is where stereo visualization comes into play, giving the user the possibility of an immersive experience, in which it is possible to acquire a more accurate idea of the final result. Within this concept, applications like the *CAVE* [2] and the *Power Wall* [11] are extensively used within the industry and have demonstrated the importance of virtual reality systems. Nevertheless, those applications are based on high-end graphics workstations and their use imposes high costs. On the other hand, the mentioned systems are optimized for use with multipipe-aware applications and support for a desktop environment is limited and restricted by severe performance penalties [9].

Application of consumer hardware for high-performance interactive visualization has recently been recognized as an alternative to traditional workstation-based visualization systems. PC-Clusters have proved to be a powerful tool in the field of scientific computing, and lately, projects like **WireGL** [6] from the Stanford University, have explored the feasibility of cluster application for the construction of very high resolution tiled-display systems. Muraki et al [7] have

† Lehrstuhl für Graphische Datenverarbeitung,
Am Weichselgarten 9, 91058 Erlangen, Germany,
Email:Fernando.Vega.Higuera@informatik.uni-erlangen.de

‡ Forschungs und Innovationszentrum (FIZ) BMW Group,
Knorrstraße 147, 80788 München, Germany,
Email: Thomas.Reuding@bmw.de

constructed a high resolution volume-visualization system using PC-clusters equipped with volume-rendering graphics cards and additional image compositing hardware.

In the context of stereo graphics, OpenInventor [10] is a widespread standard for the development of interactive 3D visualization applications; its backbone is the *scene-graph*, which is the database containing the objects that compose the scene. OpenInventor provides stereo rendering by creating two images from the scene; the camera is placed at the right and left eye positions alternately. This means, that in order to produce the pair of stereo images (so called *stereo pair*) the same scene must be processed and rendered twice. We introduce a novel approach by parallelizing this task: each one of these images is created on a node of a prototype PC-cluster; that is, the computers have to work independently and synchronously in order to produce a consistent *stereo pair* (see Fig 1). Prior work on collaborative

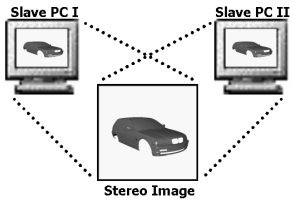

Figure 1: *Parallelizing Stereo Rendering*

virtual reality environments, like the DIV [4] and the SGAB [12] have focused on a distributed approach for *scene-graph* data sharing, and show the feasibility of constructing a system with interactive rendering capabilities.

In this work, a multi-layer architecture has been developed over the *X-Windows* system and OpenInventor making use of a *distributed-scene-graph* scheme. A master node does not render but it is only responsible for X event handling and slave coordination. At *X-Windows* level, synchronization is realized through X-event replication at the slaves. Synchronization at OpenInventor level is achieved trough *scene-graph* replication. The rendering slaves in the cluster run a local copy of the visualization application with its corresponding scene data. A slave is designed as primary and is the only one to process Inventor interaction. Its copy of the *scene-graph* is used as reference and changes made to scene are recreated by the other slave. In contrast to other multi-pipe approaches, this system provides a standard *X desktop* and multiple windowed stereo applications can be used concurrently.

The present work has been developed and evaluated over a HelpVR [8] node, which consists of a *master* and two *slave* Linux PCs. The slaves are equipped with consumer oriented high-performance graphics hardware (NVidia Quadro 4 XGL 900), and the network connection between them consists of 100 Mbit Ethernet cards and a hub. A CAD visualization application was ported and its output is displayed on a polarization-filter stereo projector.

After presenting an overview of the architecture in section 2, implementation of the OpenInventor Stereo Library for Clusters (**OISLC**) is reviewed in section 3. Finally, the results achieved with a visualization application for the automotive industry show the value of our approach and further developments of the system are discussed.

2. Architecture

We have created a system where the user perceives the PC-cluster as a stereo rendering machine and not as a computer network. User interaction is performed at the master while the slaves react accordingly. One of the clients create the left eye image, while the other one creates the image that corresponds to the right eye. These images are to be combined and viewed with standard VR visualization techniques (active or passive stereo visualization). Since those images correspond to a *X Windows* desktop (see Fig 2), areas which do not correspond to stereo rendered images must be identical. Other

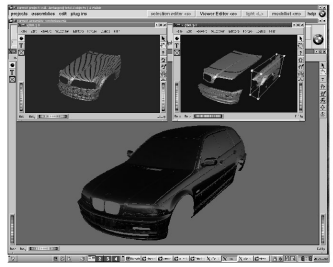

Figure 2: *A 2D desktop with multiple 3D-stereo applications*

aspect to be taken into account is scalability. It was required to develop an architecture that can be used in the construction of systems with a higher count of rendering pipes in order to provide multiple concurrent views (ie. CAVE), but maintaining the performance level by adding slaves to the cluster. With this constraint in mind we keep the rendering slaves as free as possible from non-rendering tasks; synchronization tasks are delegated to the server keeping rendering performance unaffected by the growth of the system.

In order to achieve synchronization of the *X Windows* system and OpenInventor applications, realization of the system is accomplished at two different levels (See Fig 3). At a first stage, the X Windows interface of the slaves is synchronized with the X events generated at the master. In a second phase, additional synchronization of the OpenInventor scene-graph is performed.

Since the slave desktops are identical, keyboard and mouse events produce the same behavior at both slaves. The user can start OpenInventor applications in parallel over the

cluster by normal desktop interaction. OpenInventor synchronization becomes simpler, since tasks such as object creation and deleting are performed simultaneously by all the slaves and do not need to be shared. OpenInventor synchronization is restricted to changes made by the user on the *scene-graph* through direct interaction over the scene.

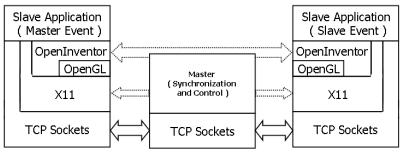

Figure 3: *Software Architecture*

2.1. X11 Synchronization

Synchronization of the *X11* system was achieved through the specialization of an existing X-based software application (**x2x**). This application makes it possible to control different client X servers from a master X server. Some of the original flexibility was sacrificed in order to optimize the software for our architecture. A modified approach was used in oder to reduce network traffic and in this manner improve responsiveness of the system to user interaction. The **x2x** application allows the user to send mouse and keyboard events from a master *X server* to different slave *X servers*. Events captured on the master are forwarded to the slaves through calls to **FakeEvent** functions which are part of the Xlib library [1]. X sends these events over the network making use of the *X protocol*. We have taken advantage of the cluster architecture in order to customize and optimize the **x2x** application. Since only mouse and keyboard events are sent, the use of the *X protocol* between different hosts is a suboptimal approach. This step was eliminated and only the required information to recreate the X events at the slaves' side is sent. Size of the data packets is reduced from the 32 Bytes used by the X protocol, to the 9 Bytes required by our synchronization protocol. A reduction of 35 % in network throughput was achieved and a decrease in the response time to user interaction was clearly perceptible.

2.2. OpenInventor Synchronization

After synchronization at X Windows level it became clear that this was not enough to achieve scene-graph replication. Interaction on Open Inventor applications executed simultaneously on the slaves produced different results although the X events registered on both machines were identical. Further testing revealed that Inventor dropped mouse events triggered on the render area, depending on the instantaneous work load of the system. Inventor synchronization was implemented in order to ensure scene consistency at the slaves.

OpenInventor bases its architecture around the *scene-graph* [10], which is the database containing the objects (called **nodes**) that describe the scene to be rendered (geometry, lights, transformations, cameras, etc.). The properties describing those objects are stored in their corresponding data members known as **fields**. In order to obtain a consistent *stereo pair*, synchronized copies of the *scene-graph* are kept at the slaves. One of the slaves act as reference for interaction, and its copy of the *scene-graph* is replicated by the second slave. The primary slave is the only one to process the

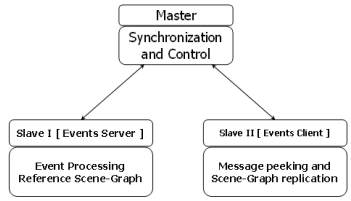

Figure 4: *Scene-graph Synchronization*

X events generated over the render area. This guarantees that its copy of the *scene-graph* is the only one that is modified by user interaction. The primary slave is in charge of transmitting those changes to the master; then the master broadcasts the information to the second slave. This architecture enables the addition of new slaves to the cluster without increasing the complexity of the non-rendering related tasks for them (see Fig. 4) since their work load is independent from the cluster's node count.

The master does not hold a copy of the scene-graph and its in charge of synchronization only. This makes it possible to avoid the need of a 3D graphics-card at the master.

2.2.1. Node Synchronization

Primary slave data is sent only when changes to the *scene-graph* have been made; OpenInventor provides a mechanism to sense modifications made to the nodes and it allows the installation of callbacks functions in order to respond properly to those events. This mechanism was used and a callback function is in charge of storing changes made to the corresponding node. Sending every single change proved to be unnecessary and too expensive in terms of communication. An independent thread sends the stored changes at regular time intervals and only when there is new data.

At the secondary slave's side the modifications are received and applied to the *scene-graph*. Since the Open Source implementations of OpenInventor are non thread-safe, changes to the nodes are made from within the rendering thread. A communication thread stores the received data in memory, and a timer-object peeks those changes at regular time intervals and applies the required modifications. (see Fig. 5)

2.2.2. Stereo Camera

A virtual monoscopic camera-object is employed for peeking. This object is synchronized through the normal node-

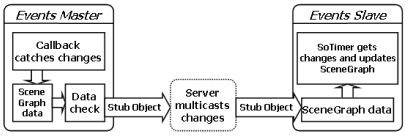

Figure 5: *Nodes' Synchronization*

synchronization mechanism. Each slave sets its camera at the corresponding stereo-position (*left or right eye*) before rendering the scene. The implemented approach does not prevent the user from making use of OpenGL extensions such as vertex and pixel shaders. A particular rendering code can be introduced without worrying about the stereo tasks.

2.2.3. Frame Synchronization

According to the work from Cruz-Neira et al [2], frame refreshing for the different displays must occur within a time window of 8 ms in order to avoid artifacts. The double buffering mechanism from OpenGL was used to control the simultaneous frame refreshing; whenever a slave is ready to perform a swap buffer operation, it sends a request to the master and enters a wait state. The server checks whether all the slaves are ready, and then issues a swap-buffer command to all the rendering nodes (as shown in Fig 6). If the redraw

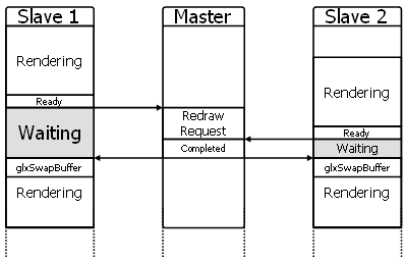

Figure 6: *Frame Synchronization*

request has not been completed after a given time (i.e. not all the slaves need to swap buffers), the server issues the redraw command; this will allow the blocked slave to continue and won't have any effect on the other slave.

3. The OISLC

The functionality of the architecture has been encapsulated in a set of classes that we named the ***OpenInventor Stereo Library for Clusters (OISLC).*** Users have to introduce a minimal amount of modifications in order to port their applications to the system. These will run as clients to the OISLC and the server application does not need to be modified to suit any particular client. The server provides a simple graphical interface allowing the user to select between four different rendering modes (convergent cameras -*variable eye separation*-, fixed convergent cameras, parallel cameras and monoscopic), and adjust stereo settings.

At the clients' side, event processing and frame synchronization is carried out by the viewers. A full hierarchy of **SoXt** viewer classes is provided. Camera positioning is integrated in a specialized **render action**. The viewer classes of the application to be converted must be replaced by one of the mentioned stereo viewer classes, and a stereo render action must be provided.
Finally, the most important component of the library is the main client class: **ClientOISLC**. It provides the interface for connecting to the server and node sharing.The user needs to interact only through two functions:

- `ShareNode( SoNode* node );`
- `RemoveNode( SoNode* node );`

We have based our approach on a concept from Hesina et al [4], where the library checks for the type of the node being shared and creates the necessary underlying classes for synchronization. The first function creates and registers the corresponding node-sharing classes, and the second one makes a clean up, and has to be called when the node in question needs no longer to be synchronized. The provided set of classes allows for almost transparent porting of OpenInventor applications in order to take full advantage of the developed architecture.

The following example demonstrates creation of a StereoExaminerViewer with the corresponding stereo camera and a point light:

```
// Create the StereoExaminerViewer
SoXtStereoExaminerViewer* stereoViewer = new
SoXtStereoExaminerViewer(TheWindow,
                         "Server OISLC" );

// Create the StereoRenderAction
SoStereoRenderAction* stereoRA = new
SoStereoRenderAction((*stereoViewer).
                     getViewportRegion() );

// Create the OISLC client
ClientOISLC* clientOISLC = new
ClientOISLC( serverPort, clientPort );

// Initializate the Stereo library.
(*clientOISLC).Start( stereoViewer,
                      stereoRA );

// Share the camera
(*clientOISLC).ShareNode( stereoViewer->
                          getCamera() );

// Share a point light manipulator
(*clientOISLC).ShareNode( pointLight );
```

4. Results

A cubing application (*the process of checking the visual correctness of a CAD design is known as cubing process*) developed by BMW with OpenInventor, was ported to our architecture. This tool provides standard features such as lights

and textures, clipping, markers, etc. It is used extensively during the development of car parts, up to car bodies within the engineering department. Initial evaluations of the system were carried out by connecting the graphic outputs of the slaves to standard PC-monitors in order to check consistency of the obtained *stereo pair* (see Fig 8). Once satisfactory results were achieved, the video signals of the slaves were used as input for a stereo projector. In this manner visualization with polarization-filter glasses is possible (passive stereo) (see Fig 9). This proved the effectiveness of the approach to create a consistent *stereo desktop* in parallel. The system was evaluated in a work environment, through cubing of CAD car body models at different mesh resolutions. Additional performance benchmarks were performed. Time delay between redraws at different slaves was evaluated (see Table 1). Although a maximum value of 12 ms was recorded, almost no frames are beyond the 8 ms mark and the average time is smaller than 1,5 ms. We have to note that frames with timeout created some visual discomfort, nevertheless, the percentage of frames with timeout did not reach 1 %.

Frames	Max.	Avg.	Pct. under 8 ms	Timeouts
10.158	12 ms	1,45 ms	99,9705 %	0,423 %

Table 1: *Time needed for frame synchronization*

The graphics performance of the cluster was measured with the cubing tool; a spin animation is performed with a polygonal model of a car body at different resolutions, using a light source and OpenGL lighting activated. Graphics-performance loss due to synchronization was measured (see Table 2).

Triangle Count	Tri/sec Mono	Tri/sec Stereo	Graph. load
700.398	4.146.356	3.172.109	89,52 %
350.199	4.188.380	2.917.157	69,64 %
66.137	2.702.357	1.693.107	62,65 %

Table 2: *Stand alone vs. Stereo performance*

It can be seen that synchronization costs for low complexity models turns out to be significant. Nevertheless, it is low for higher resolutions. Software tests with our library on machines running IRIX shown minimal performance loss due to synchronization, in contrast to our LINUX system. This problem seems to be related to the accuracy of the operating system task scheduler under LINUX. Work in this area is being carried out in order to use real-time scheduling.

It should be noted also, that the given numbers correspond to frame rates measured on each rendering machine, and in consequence the performance numbers must be multiplied by two in order to measure the overall performance of the system (see Table 3).

Triangle Count	Tri/sec Octane2	Tri/sec 3-PC-cluster
700.398	2.661.512	7.424.218
350.199	2.591.472	5.834.314
66.137	1.587.288	3.386.214

Table 3: *Octane2 / Quadro-cluster comparison*

Performance drops due to communication load were not detected while running multiple stereo applications simultaneously. This is a very interesting feature, since multipipe virtual reality systems such as the Onyx2 recently introduced the use of windowed single-pipe applications only through the OpenGL multipipe library [9]. These suffer severe performance drops which increase with the complexity of the rendered models. This limitation is also apparent in **WireGL**, due to the overhead created by catching graphic primitives and forwarding them to the independent pipes. Our approach is better suited, for applications such as CAD visualization where graphical data is static, and only changes in the scene must be shared. Nevertheless, our system is not suited for visualization of large-scale data, since neither object-space nor image-space parallelization is carried out.

5. Conclusions and Future Work

We have presented an architecture for *parallel OpenInventor stereo rendering over clusters*. A prototype Linux PC-cluster was built, where two slaves are in charge of rendering, and a server performs synchronization tasks. The presented work introduces a new approach for stereo rendering with user-oriented graphics hardware. Practical application for a virtual reality CAD environment inside the automotive industry was demonstrated. The scalability of our architecture allows us to think of further applications like the construction of a PC-based high resolution stereo desktop.
Further additions to the system are required in order to implement this idea. The current camera synchronization is

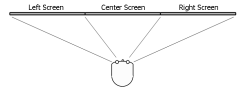

Figure 7: *Off axis projections in a multi-display system*

based on the SoCamera classes from OpenInventor, which implement symmetric projection frustums; nevertheless, for a multi-screen display, where the stereo image is conformed by complementary segments, off-axis projections must be implemented [2] (Fig. 7). This can be achieved adding a new OpenInventor node class without affecting the structure of

our architecture, since a new PC-pair would render each image segment. Research is currently under way in order to combine ideas from the Distributed Multihead Project [3] with our system.

Synchronization penalties of the system are neglectable for high complexity models and this feature makes it attractive for OpenInventor applications that require high performance rendering of static CAD data. Existing OpenInventor applications can be ported to our system with a minimal effort from the programmer, thanks to the simplified interface that *OISLC* offers. The user can make use of multiple stereo applications over a desktop without performance drops due to communication.

An architecture for PC-clusters was developed, which achieves stereo rendering performance levels comparable to those from graphics-workstations, making possible the construction of low-cost high-performance virtual reality solutions based on OpenInventor.

Acknowledgements

We would like to thank BMW AG which kindly provided the necessary computing equipment for this project, as well as CAD datasets.

References

1. H. Abdel-Wahab and K. Jeffay. Issues, problems and solutions in sharing X clients on multiple displays. *Journal of Internetworking research and experience*, 5:1–15, 1994. 3
2. C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-Screen Projection-Based Virtual Reality: The design and implementation of the CAVE. In *Proceedings SIGGRAPH*, pages 135–142. ACM, 1993. 1, 4, 5
3. DMX. Distributed multihead x project. http://http://dmx.sourceforge.net, 2001. Accessed in February 2003. 6
4. G. Hesina, D. Schmalstieg, A. Furhmann, and W. Purgathofer. Distributed OpenInventor: a practical approach to distributed 3D graphics. In *Proceedings of the ACM symposium on Virtual reality software and technology*, pages 74–81. ACM, December 1999. 2, 4
5. H. Hu, A. Gooch, and W. Thompson. Visual Cues for Imminent Object Contact in Realistic Virtual Environments. In *Proceedings of the conference on Visualization 2000*. ACM, October 2000. 1
6. G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan. WireGL: A Scalable Graphics System for Clusters. In *Proceedings of SIGGRAPH 2001*. ACM, August 2001. 1
7. S. Muraki, M. Ogata, K.-L. Ma, K. Koshizuka, K. Kajihara, X. Liu, Y. Nagano, and K. Shimokawa. Next generation supercomputing using pc clusters with volume graphics hardware devices. In *Proceedings of Supercomputing 2001 Conference*, pages 10–16, November 2001. 1
8. Siemens AG. High end low price virtual reality HELPVR. http://www.tu-bs.de/rz/software/graphik/Onyx/HELP_flyer_eng3.pdf, 2000. Accessed in February 2003. 2
9. Silicon Graphics Inc. OpenGL multipipe. http://www.sgi.com/software/multipipe, 2001. Accessed in February 2003. 1, 5
10. P. Strauss and R. Carey. An Object-Oriented 3D Graphics Toolkit. In *Proceedings of SIGGRAPH 1992*, pages 341–349. ACM, August 1992. 2, 3
11. University of Minessota. Power wall. http://www.lcse.umn.edu/research/powerwall/powerwall.html, 1994. Accessed in February 2003. 1
12. B. Zeleznik, L. Holden, M. Capps, H. Abrams, and T. Miller. Scene-Graph-As-Bus: Collaboration Between Heterogenous Stand-alone 3-D Graphical Applications. *EUROGRAPHICS 2000*, 19:91–98, 2000. 2

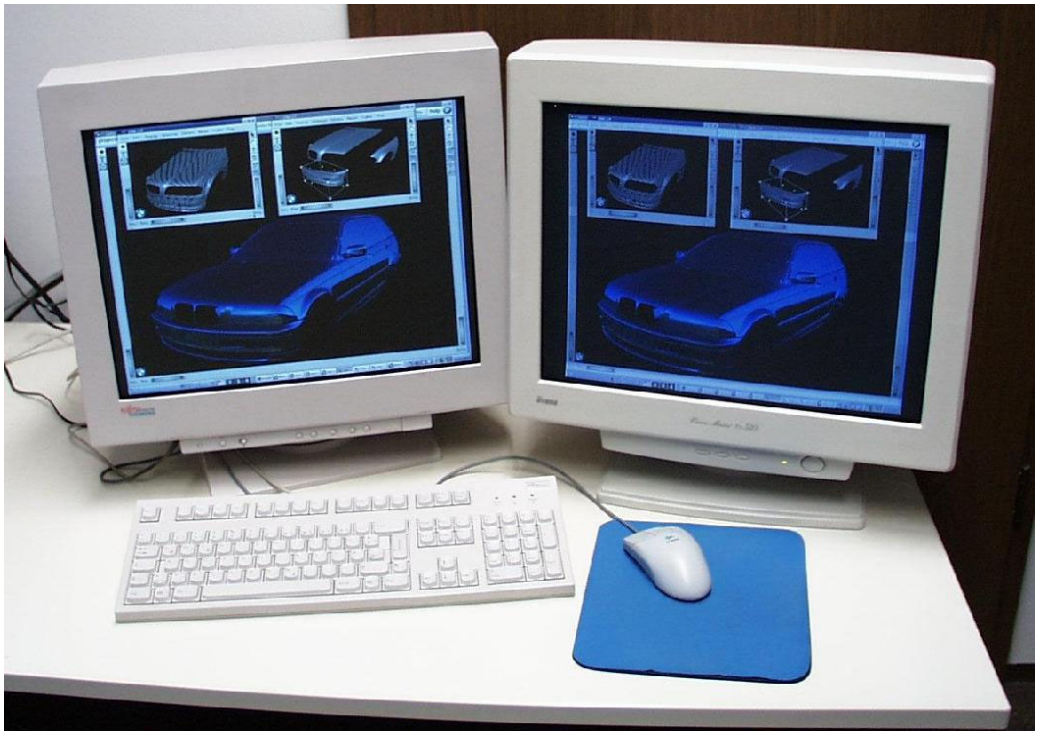

Figure 8: *Stereo pair displayed on PC-monitors*

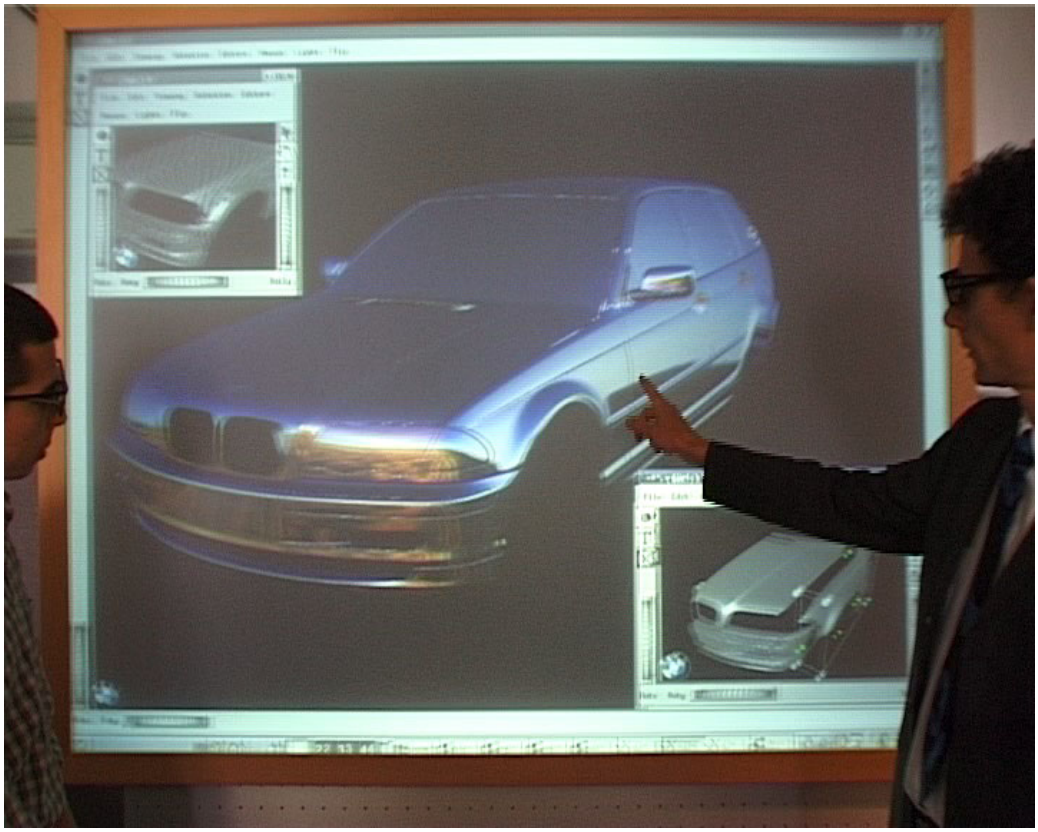

Figure 9: *Stereo cluster application within BMW*