

Tiled Displays to Enable Collaboration on the Grid

Luc RENAMBOT¹ Tom VAN DER SCHAAF¹ Henri E. BAL^{1,2}
Desmond GERMANS² Hans J.W. SPOELDER² Thilo KIELMANN¹

¹Division of Mathematics and Computer Science

²Division of Physics and Astronomy

Faculty of Sciences, Vrije Universiteit

De Boelelaan 1081, 1081 HV Amsterdam, The Netherlands

1 Introduction

An important innovation of computational grids is their support for new types of collaborative applications, where people at different locations cooperate by interacting with each other and with application programs. In science and engineering, there is a large demand for collaborative grid applications, especially for multidisciplinary projects, where researchers with different backgrounds (and in different institutes) interact. Examples range from collaborative car design to climate modeling. Often, scientists want to interact not only with each other, but also with a running application program, such as a simulation, and they want to discuss new results as soon as they have been produced by their software. We believe that another major demand for collaborative applications will come from the educational world. As distance education is becoming more commonplace, teachers want to interact with multiple groups of students at different locations simultaneously, as if they were all located in a single classroom. Interaction with running application programs also is important here, but now for didactic purposes.

Unfortunately, current hardware and grid software still miss much functionality to easily realize large-scale collaborative applications. On the hardware side, the display technology that is now typically used for collaborative applications does not scale to large numbers of users. For example, Access Grid nodes typically use a single PC and graphics card for display purpose, severely limiting the amount of information (pixels) that can be displayed, and thus the number of people that can effectively participate. At least for educational applications, this is a severe restriction. Even more important, little software exists yet that supports interaction with application programs. The Access Grid, for example, is designed for interaction between humans only.

In this paper, we will describe hardware and soft-

ware technology that together enable scalable collaborative grid applications. Our hardware is based on scalable tiled display technology, which uses relatively inexpensive, commodity components (PCs and projectors). We have set up such a display, called the *ICWall*, and designed efficient low-level rendering and calibration software for it. In addition, we have built an integrated software environment to develop collaborative and interactive 3D grid applications for research and education. The resulting system is highly suitable for displaying large amounts of information to large audiences. The *ICWall* is being used both for teaching (see Figure 2) and research (see Figure 10), using traditional presentation tools like Powerpoint (see Figure 7) as well as tailor-made interactive applications (see Figure 8). It can be used both for local applications and it can be coupled to similar display hardware elsewhere for remote grid applications.

In the rest of the paper, we will first describe the *ICWall* hardware and the low-level systems software (Section 2). Next, we discuss a general and flexible software architecture that eases the development of interactive and collaborative applications for the wall (Section 3). Finally, we discuss several scientific and educational applications that we have developed together with several other departments (Section 4).

2 The *ICWall* Tiled Display

Due to the rapid developments in computer graphics, which outperform similar development in processor architecture, all kinds of visualization techniques are playing an increasingly important role in education and research. The key characteristics for choosing a tiled display in an education environment are among:

- Scalability: The size of the display is determined by the number of projectors used.
- Standard components: The setup can be realized using commodity components, for a reduced price.

- Resolution: The actual resolution is the sum of the resolution of the individual projectors.
- Size: The large format allows a wide variety of different applications, by simultaneous projection of different types of information and saturation of the observer's field of view.

All these characteristics make a tiled display more suited to be used in educational environments than for instance a CAVE [2] or a standard projector. The techniques used determine to a large extent the number of people that can observe it and also the type of interaction. Especially in an educational setting, this is a key factor. Our project takes into account three aspects. The current trend in hardware for parallel graphics is to use clusters of off-the-shelf PCs instead of high-end super computers. Using large, high-resolution displays is another trend recently emerging. High resolution allows for detailed scientific visualization, and overcomes the limited screen resolution of standard monitors. Large displays also have applications in teaching environments, where multiple people (from small groups to full classroom) interact with a single large screen, either locally or remotely in a distance learning setting. Last, group-to-group collaboration becomes widespread using the AccessGrid technologies, making use of the increase of network bandwidth.

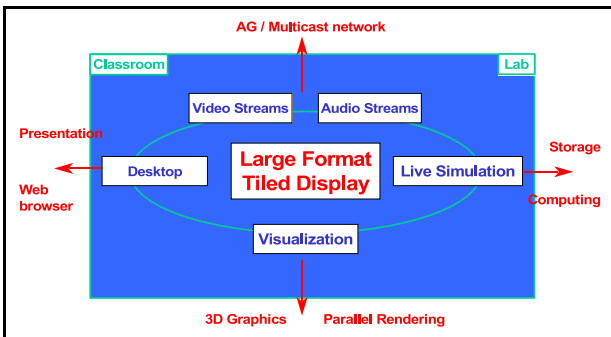


Figure 1. Tiled Display on the Grid

Our global vision is schematically described in Figure 1 with the display facility at the center of a Grid infrastructure. A large-format display in a laboratory or in a classroom becomes the access point to all kinds of resources and media. We can imagine various challenging scenarios. Lecturers get a very large desktop to present their slides, and simultaneously also course notes, source code examples, or actual running applications. Distant learning is enabled by audio and video streams and a multicast enabled network. Researchers can display datasets while discussing on-site and re-

motely with numerous colleagues. Other researchers can interactively steer a simulation running on a remote powerful computer and visualize large amounts of data produced. Finally, scientists can use the powerful rendering cluster driving the display for remote visualization over high-bandwidth networks, and can engage a collaborative session with their colleagues. All these scenarios and their various combinations induce a new way of doing visualization and collaboration on the Grid.



Figure 2. Overview of the *ICWall* tiled display

2.1 Setup

The global view for the room of the tiled display is given in Figure 2. The approach is that the room should be as multi-functional as possible. The *ICWall* room is capable of seating about 50 people for a projection area of roughly 5.0 by 2.5 meters.

We opt for a back-projection construction in which the projectors are placed in a frame behind the screen. This solution is cost effective, and leaves the room empty of any equipment in front of the projection screen. This is particularly beneficial in a lecture environment. Moreover, users do not cast any shadow while moving in front of the screen. To limit maintenance, only an approximate manual alignment of the projectors is needed, leaving sufficient overlapping projection areas. A fast and accurate software alignment procedure, described latter, produces a seamless image out of all the projectors.

The *ICWall* is driven by a cluster of eight rendering nodes and a front-end server. The nine machines are built out of standard components connected by Ethernet and Myrinet (gigabit/sec) networks. The latter is used for the rendering communication. Each rendering node is connected to a video projector. To limit manual operations, each projector is linked to a serial port of the driving node, allowing for a global control of the

setup from a single web interface. Audio and video capabilities (speakers, frame grabbers, etc) are also included to allow video conferencing.

To generate a collaborative experience from this infrastructure, several software components are required such as a parallel rendering toolkit to drive the cluster, an accurate calibration technique to produce a seamless projection surface, an application development environment, and a human computer interface.

2.2 Parallel rendering

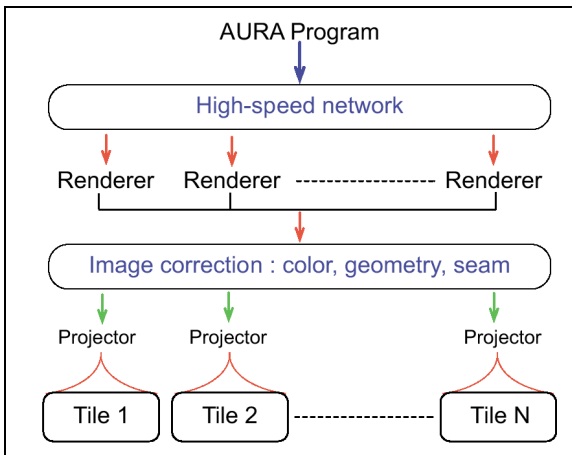


Figure 3. Parallel rendering for a tiled display

Figure 3 presents an overview of the rendering process from a single application to a large uniform image on the display. The environment enables graphics applications to be run on a cluster of machines connected by a high-speed network. The generated images are corrected in terms of geometry, edge blending, and color and then fed to a set of independent video projectors.

The application programs are written using the Aura API. This API was designed at the *Vrije Universiteit* as a platform-independent graphics API built upon OpenGL. It offers the user a C++ scene graph interface, for retained-mode rendering. The software development is based on a parallel implementation of Aura. It enables tailor-made visualization and steering of applications to be run on a cluster of PCs, by maintaining the 3D scene graph coherent across the rendering nodes. It offers high-performance and ease of use to the application programmer. The software calibration of the tiled display is transparently included in the software. More details are given in sidebar A.

2.3 Calibration

The different software calibration phases are required to produce a seamless image out of a set of roughly aligned projectors. The main steps are the geometric alignment of the images, the edge blending of neighboring tiles, and the color calibration of the different projectors.

In our approach, one static camera is used to capture the entire screen, and a fast and accurate computer vision algorithm recognizes markers in the stimuli for each individual projector. Then, an 8-parameter homographic transformation is fitted to the found corner points, resulting in a projector to camera coordinate transformation matrix, one for each projector. From these matrices, the extents of the projector surfaces can be determined, and using this, the largest displayable rectangle is calculated. Finally, this rectangle is intersected with each projector surface to yield display regions, overlap regions and respective alpha masks to account for intensity blending at the overlap regions. More details are given in sidebar B.

3 Software Infrastructure for Collaboration

To enable various types of collaboration, we extended the low-level rendering and calibration layer, described in the previous section, with various applications for a successful use of the tiled display in both education and research. We developed two groups of applications: legacy applications such a desktop environment or various file format loaders, and tailor-made interactive applications.

3.1 Legacy applications

Perhaps the most desired tool by users in general is a desktop environment. We provide a high-resolution desktop rendering using a VNC virtual display server [7]. A VNC server receives requests from clients, upon which it returns screen updates of the virtual desktop, as shown in Figure 4. Changes to the screen are compressed and sent to the requesting client. Aura has a VNC client module built in, that is optimized for parallel display. It unpacks the incoming screen updates and places them into a large texture. The texture is displayed over the entire screen. Since VNC supports both Windows and Linux, we can display any Windows or Linux application. An important advantage of VNC is that it is not limited to any physical constraints, and thus the desktop can be of any size or bit depth. The downside of this approach is that demanding applications, such as movies, can be

slow. This drawback is strongly compensated by the ability to run any standard application, such as web browsers, presentation tools, or visualization packages.

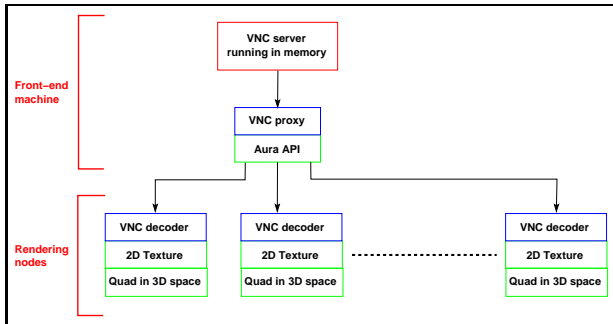


Figure 4. Desktop system using the VNC server

We also wrote numerous file format loaders for images (format jpg, gif, png, ...), for 3D scenes (format obj, nff, lwo, ...), and for MPEG1 movies. Overall, these software components give us a compelling multimedia environment.

3.2 Interactive 3D applications

From our previous work on VR environments such as CAVEs, the low-level Aura library has been extended to provide a high-level framework to develop interactive 3D applications. Several case-study applications have been implemented with researchers from various departments of the *Vrije Universiteit*. Among them are an application for the Human Movement Department showing the walking cycle of patients, an interactive probe of an electric field in molecule for the Chemistry Department, a viewer for very large images for the Geology Department, and a steerable laser simulation for the Physics Department. These applications are research tools but are also used in education to illustrate lectures.

The current software environment is being extended to support collaboration in a more efficient manner. For instance, Access Grid tools work already using the previously described desktop environment. However, the VNC server performance limits the number of video streams one can display concurrently. We developed a prototype of a video conference tool which efficiently uses the native low-level graphics combined with an RTP (real time protocol) software stack. This software will enable the display of a large number of video streams or a few high-resolution ones combined with any of our applications.

3.3 Interaction and User Interface

Our display has been set up within an educational environment and is used for several lectures from computer science (graphics, parallel programming, AI), physics, and chemistry. We describe a system consisting of several free software components that allows novice users to display their multimedia onto the tiled display. The user interface is simple and requires no knowledge about the underlying architecture, hiding the complex parallel rendering system driving the display. Since the wall is also used for research, the system allows for flexible switching between desktop applications and virtual reality applications.

The physical environment should be as multi-functional as possible (see global view in Figure 2). In the current design, several components are present, such as the large projection screen, a plasma panel with touch screen (Smartboard), and a wireless network. The latter enables lecturers and students to exchange documents and to access on-line materials with their laptops. The lecturer uses the touch screen as interface for control and interaction of the tiled display.

Our display will be used extensively for teaching purposes, bridging the gap between the laboratory and the classroom. Researchers are primarily interested in using tiled displays for visualization of complex simulations or high-resolution data. Teachers are more interested in showing their presentations, movies, and websites. In order to fulfill these different needs, the setup must meet several requirements:

The system

- The hardware must appear as a single host with a single (albeit very large) display. This implies that the parallelism of the underlying graphics cluster needs to be transparent.
- The system should support as many applications as possible, from presentation and desktop tools to interactive 3D applications. The possibility to visualize standard file formats is an important demand from teachers (for instance, VRML for 3D scene or PDB molecule for protein visualization). Performance should be at least comparable to a standard environment (single PC), and hopefully better using the resources (computing and graphics) available on the driving cluster.

The user interface

- It should be possible to start programs with a single mouse click (for instance, using a web in-

terface). Users should not require high-level computer skills to run their programs.

- The human-computer interface should be open to accept a large variety of input to accommodate advanced users or demanding applications (for instance, joystick, PDA, tracking devices, speech system).

For the implementation of the above described environment, we made use of the following free software packages:

Parallel Aura for the low level graphics driving the cluster,

Apache web server to host the user interface (Java applet) and documentation,

VNC offering a remote display system for displaying a large desktop. We use the following packages to support popular media: *OpenOffice* for displaying presentations (for instance, MS powerpoint), *Acrobat Reader* for displaying PDF documents, *Mozilla* for web browsing, *Mplayer* for showing movies in numerous file formats.

XML-RPC (in Java, C++ and Python) to serve as software glue between all the different distributed components.

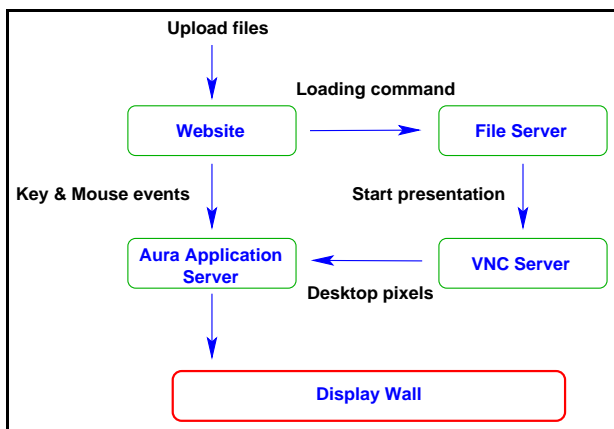


Figure 5. Interface and Interaction System

For the interaction, we focus on the user perspective. Not all users are experienced with the Linux command line tools, so a simple user interface should be provided. Therefore, our wall can be fully controlled from a website, as shown in Figure 5. Teachers can upload their presentations, movies, and datasets to their account via the website. After the upload, their new file is immediately available. Simply clicking on the file displays it on the wall. To achieve this behavior, we use a combination of two servers and a Java client.

(Figure 6 shows a picture of the Java applet.) To integrate these different components, all communication is made through standard XML-RPC.

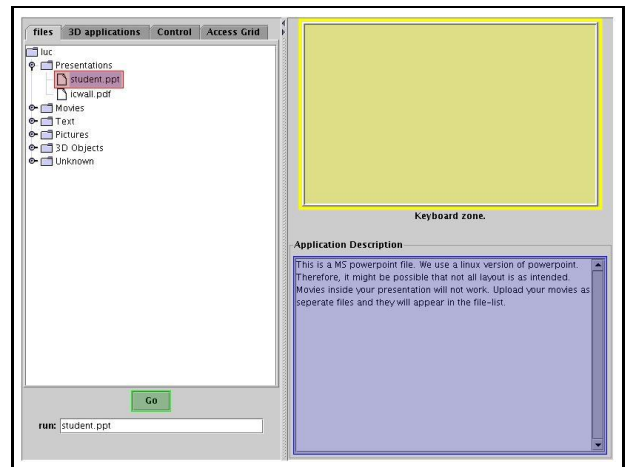


Figure 6. The Java ICWall control applet

The first server is a file manager (written in Python). It handles all file issues: uploading, starting programs in the VNC desktop, and giving account info to the web client.



Figure 7. An actual lecture

The second server is the Aura application server that will show the output of the program. This server, upon request, loads the dynamically loadable library matching the request (e.g. a VNC desktop if the request is a Powerpoint presentation, or the Aura image viewer if the request is an image). The loaded library uses the parallel Aura API to provide the requested functionality. Upon loading a new library, the previous one is discarded. However, we plan to support multiple libraries simultaneously. Figure 7 shows a

lecturer using the desktop library with a Powerpoint presentation and a movie. Several virtual reality applications, each implemented as a separate library, have been also developed. Figure 8 shows the same lecturer, demonstrating an interactive visualization of a tooth.



Figure 8. A lecturer running a 3D visualization

In the future, we will write more libraries. For example, movies can be fairly slow using the VNC server (especially high-resolution movies). We will write a separate movie player for Aura that uses low-level features to improve performance. Movie files can then be redirected to the new library instead of the VNC desktop.

The user can control the application from the website: mouse moves and keyboard strokes are transmitted to the application server and passed to the library to interpret them. Experienced Linux users can use a local VNC client (via the website), that allows full control, so that they can start and control any Linux program themselves, bypassing the website.

We have build a user friendly interface to a tiled display wall, that hides all technical details. However, we do not have sufficient day-to-day experience with users to make a full usability report yet, but the first lecturers using it were very positive. Starting in september, the tiled display will be used on a daily basis by lecturers from different departments of the University. This will give us the required feedback to improve the system.

4 Grid Applications

Having an integrated environment (Figure 11 in sidebar A) based on low-level parallel rendering and the software components (desktop and simulation steering

for instance) eases the development of new types of collaborative applications on the Grid. With previously existing technology, like the Access Grid, the lack of development tools and the limited resolution makes the development of new applications much more difficult.

In this section, we present three selected applications in the domains of challenging 3D visualization and computational steering: a high-resolution geology map visualization, an interactive steering of a parallel molecular dynamics simulation, and a remote visualization technique. These applications required little programming effort to be brought to our tiled display. The three applications case studies together illustrate the various advantages of our system:

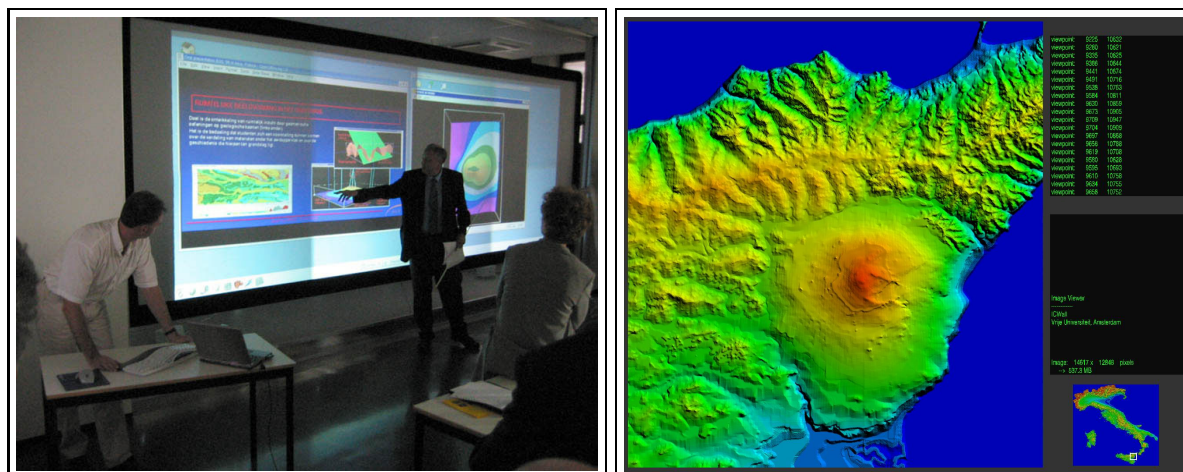
- high-resolution maps exploit the large number of pixels of the wall,
- the MD simulation shows how interaction with a running simulation program can be implemented with our software environment,
- remote visualization shows the coupling between a display and a remote rendering facility.

4.1 High-resolution Maps

A high-resolution visualization environment such as the *ICWall* enables the display of full resolution images without loss of detail. Compared to a monitor visualization, the large projection area also makes it possible to see a large part of the map at any time without losing context. These two points make such an application useful for research and education, where colleagues or a lecturer can explore and show interactively any detail of such a map. We developed an application for the Earth Science Department showing height maps (color-coded altitude) for two regions of their interest: Italy and Spain. The map files are streamed from the disk of the control machine to the rendering nodes while the user walks through the map. The files are very large due the high resolution of the maps (100 meter per pixel):

- *Italy* map: width 14617 pixels, height 12848 pixels, for 537 MByte of data,
- *Spain* map: width 15501 pixels, height 9616 pixels, for 426 MByte of data.

The high-speed network and the large texture memory of the graphics card enable an interactive walk-through of the maps at about 5 frames per second. Figures 9(a) shows the actual lecture given by a professor of the Earth science department using the wall. Figure 9(b) depicts the interactive application using the map of *Italy*: the main part is the map of interest, while some statistics and an overview are presented on the side.



(a) Presentation

(b) High resolution map of Italy

Figure 9. Earth science lecture

4.2 Interactive MD Simulation

The computational steering toolkit CAVEStudy [6] was designed to steer remote simulation from a VR environment. CAVEStudy wraps an unaltered remotely running stateless simulation program, and presents a simple programming interface to the user in the VR environment. By describing the inputs and outputs of a simulation into a simple XML file, CAVEStudy generates two components, a *proxy* and a *server* (see Figure 11 in Appendix A). First, a server program controls the execution of the simulation (without modification of the sources or binary). A proxy component is also generated to send parameters and to receive the data produced by the simulation. The data generated by the simulation is automatically propagated to the proxy object, which can be seen as a local copy of the remote simulation. These two components communicate through the CAVERNsoft [3] network library. For visualization, the proxy object is plugged into our visualization toolkit for interactive steering of a running simulation, closing the loop between the user immersed into the data space and the corresponding simulation.

By using the reflector mechanism of CAVERNsoft, it is possible to access one simulation from multiple sites which all receive the same data. This way, a basic collaboration setup can be realized among multiple sites through the control of the simulation. Each site, depending on their visualization setup, display different representations of the data.

This particular application deals with the coupling of a molecular dynamics (MD) simulation to a virtual

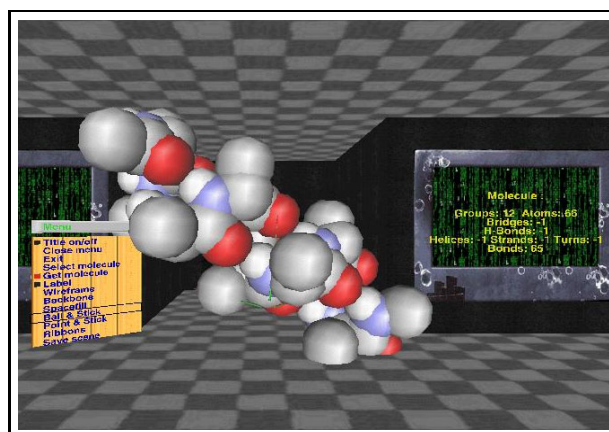


Figure 10. Interactive Molecular Dynamics

reality system. Molecular modeling tools are essential to design and study new molecules. For example, when steering a molecular dynamics simulation, the user can express external forces to help the system to overcome energy barriers, or can help in the search for likely geometric configurations in docking problems [4]. 3D visualization allows the scientist to gain a deeper understanding of the complex molecular formations.

Our current implementation allows the molecular dynamics simulation NAMD [5], running remotely on the DAS parallel cluster computer [1] to be visualized on the ICWall environment, in a Grid scenario.

The selected input parameters for steering the simulation are the name of the molecule on which the

simulation will be applied, the number of time steps of the simulation, and the temperature. It corresponds to the minimal set of parameters among the large possibilities of NAMD. As output of the simulation, we use the PDB description files produced as intermediate result during the execution. These files, which contain the position and velocity of all the atoms, are read by the proxy process and sent continuously to the visualization, showing the dynamics of the molecule. Several classic molecule representations are available (wireframe, sphere, backbone). An example is shown in Figure 10.

The coupling of the MD simulation to visualization has already been done before, but always by modifying source code. Usually, they are made by the original developers modifying their own software and using simple TCP/IP connections, and in a very application-specific manner. Using *CAVEStudy*, we were able to very quickly couple NAMD to our virtual environment. With *CAVEStudy*, we can easily switch between several simulation packages. Furthermore, to steer such a simulation adequately, 3D forces should be expressed, which can efficiently be done in a 3D environment implemented by *CAVEStudy*. The large and high-resolution display allows to show to a large classroom typical examples of molecular dynamics whereas molecule behavior could be explored collaboratively by a group of scientists, standing in front of the large display.

4.3 Remote Visualization

With the advance of high bandwidth networks, such as continent or country wide optical networks, high quality and high resolution remote visualization becomes possible. The parallel rendering protocol we developed for the Aura API can be used for remote rendering and collaborative work. The scenario is the following: an Aura application runs on one site (using a local dataset or connected to a simulation) sending scene graph updates to a rendering site (single machine or cluster) which replies by sending the generated pixels back to the application or to any visualization setup. The display can be a single monitor or a tiled display. Numerous configurations can be set up for different needs, for instance high-performance rendering for large datasets or coupling of two or more tiled displays. We are currently working on such a demonstration for the *iGrid'2002* conference coupling rendering clusters and tiled displays across the ocean. A first prototype shows the feasibility of the remote visualization between a single workstation in Amsterdam and a small cluster (4 nodes) in Chicago. Future work will focus on the optimization of the communication over optical networks using new protocols (such as reliable UDP)

to achieve the very high bandwidth promised by such networks, and to reduce the latency required for interactive applications.

5 Conclusion and Future Works

We presented a new hardware and software infrastructure which enables collaboration on the Grid, not only between humans, but also with running application programs. The main enabling technology is the tiled display, providing higher resolution and larger projection surface than any other system. The main advantage is its scalability in terms of performance and resolution. However, a key factor in its success is the software environment which exploits this hardware. We designed an efficient parallel rendering layer, accurate calibration software, and several software components into a global framework which eases the development of new collaborative applications. For instance, a full desktop solution is provided, controllable by a web interface. Also, several applications showing the benefits of the infrastructure have been described, such as a high-resolution map visualization tool and an interactive molecular dynamics simulation and visualization. Furthermore, we showed how these applications can be used both for education and research settings.

In the future, we will explore more challenging applications such as remote rendering and visualization using the same protocol that we designed for the low-level parallel rendering. This will exploit next-generation networks being currently deployed between major universities and research centers around the world. Moreover, we are currently converting our tiled display to passive stereo mode for semi-immersive visualization, using two projectors per tile (one for each eye). This leads new to new challenges for the calibration software. Finally, the infrastructure will be heavily used in the coming semesters by numerous courses, giving us valuable experiences for the validation of our developments.

References

- [1] H. Bal, R. Bhoedjang, R. Hofman, C. Jacobs, K. Langendoen, T. Rühl, and F. Kaashoek. Performance Evaluation of the Orca Shared Object System. *ACM Transactions on Computer Systems*, 16(1):1–40, Feb. 1998.
- [2] C. Cruz-Neira, D. Sandin, and T. DeFanti. Surround-screen Projection-based Virtual Reality: The Design and Implementation of the CAVE. In J. T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, Aug. 1993.

- [3] J. Leigh, A. Johnson, T. DeFanti, and M. Brown. A Review of Tele-Immersive Applications in the CAVE Research Network. In *IEEE Virtual Reality'99*, pages 180–187, 1999.
- [4] D. Levine, M. Facello, P. Hallstrom, G. Reeder, B. Walenz, and F. Stevens. Stalk: An Interactive System for Virtual Molecular Docking. *IEEE Computational Science*, 4(2):55–65, April-June 1997.
- [5] M. T. Nelson, W. F. Humphrey, A. Gursoy, A. Dalke, L. V. Kalé, R. D. Skeel, and K. Schulten. NAMD: A Parallel Object-Oriented Molecular Dynamics Program. *The International Journal of Supercomputer Applications and High Performance Computing*, 10(4):251–268, 1996.
- [6] L. Renambot, H. E. Bal, D. Germans, and H. J. Spoelder. Cavestudy: an infrastructure for computational steering in virtual reality environments. In *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing*, pages 57–61, Pittsburgh, PA, Aug. 2000.
- [7] T. Richardson, Q. Stafford-Fraser, K. R. Wood, and A. Hopper. Virtual network computing. *IEEE Internet Computing*, 2(1):33–38, Jan. 1998.

A Aura Graphics API

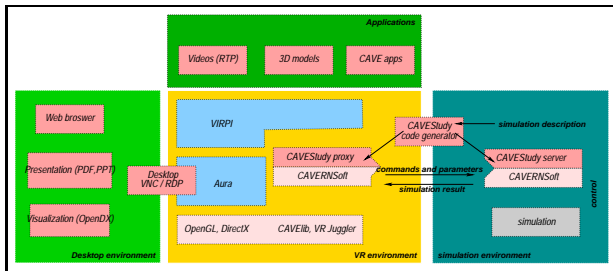


Figure 11. Software Architecture

The software development is based on the Aura API as shown in Figure 11. The Aura API was designed as a platform-independent graphics API. It runs on several platforms and operating systems, such as IRIX, Linux, and Windows. It offers the user a C++ scene graph interface, for retained-mode rendering. For a complete description of Aura functionalities and some case study applications, we refer to [1].

A parallel implementation of this library allows programs to be run unmodified on parallel architectures such as the *ICWall* PC cluster. We designed a communication protocol to keep several copies of the same scene graph coherent. For performance reasons, only the visible part of the scene graph on each node is kept consistent.

Initially, the entire scene graph is replicated on all nodes and the master. Updates are then sent to the rendering nodes to maintain all the copies consistent, in contrast to immediate mode parallel rendering where the complete scene is sent every frame [2]. Communication is optimized for dynamic objects where only the nodes displaying the objects receive the updates. If an object moves from one tile to another, global updates are sent to maintain the visible objects on each tile consistent with the master. Sorting is done in a “per-frame sort-first” fashion: the master calculates by means of bounding-boxes which object should be sent to which node. Each node renders only those objects that fall into the tile of its associated projection area.

Updates to the scene graph are of two kinds:

- Creation and destruction: separate messages are sent to create nodes, to add nodes to the scene graph, and to remove nodes from the scene graph. These types of operations are broadcast to every server to keep the graph consistent on every node.
- Object modification: whenever the user-program modifies an Aura object, the client marks it as

dirty. Before the rendering loop, it transmits a message containing the new data for each dirty object. Each type of object has its own set of message types and several state bits, meaning that only modified fields are sent. For instance, a dirty bit per vertex is maintained in vertex buffers and then, only dirty vertices are transmitted. The same approach is used for other parameters (normal vector, color, etc). Unmodified objects do not generate communication.

The parallel implementation of the Aura API and a performance evaluation are described in detail in [3].

References

- [1] D. Germans, H. J. Spoelder, L. Renambot, and H. E. Bal. VIRPI: A High-Level Toolkit for Interactive Scientific Visualization in Virtual Reality. In *5th Immersive Projection Technology Workshop*, May 2001.
- [2] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan. WireGL: A scalable graphics system for clusters. In *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, pages 129–140. ACM Press / ACM SIGGRAPH, 2001.
- [3] T. van der Schaaf, L. Renambot, D. Germans, H. Spoelder, and H. Bal. Retained Mode Parallel Rendering for Scalable Tiled Displays. In *Proc. 6th annual Immersive Projection Technology (IPT) Symposium, Orlando Florida, Mar. 2002*.

B Calibration of Tiled Displays

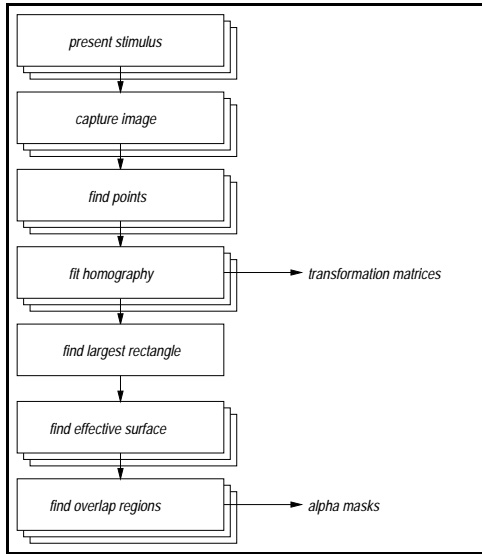


Figure 12. Calibration Procedure

The calibration process (Figure 12) of a tiled display consists of several steps: geometric alignment of the tiles, blending of the edges, and color calibration of the different projectors.

To accurately detect features on the stimulus for a projector surface, we use a regular 8x8 checkerboard pattern. Of this pattern, each crossing is detected by convolving the captured image with a matched filter. From the convolved image, everything above a certain threshold is regarded as a crossing and added to the list of found crossings. Groups of pixels are treated as one crossing, using a sub-pixel localization algorithm [2].

The list of crossings is fitted against a homographic transformation, similar to Chen *et al.*[1]. It is essentially a 3x3 homogeneous 2D transformation, where the 3rd coordinate behaves like a perspective denominator (as in 4x4 homogeneous matrix calculus):

$$\begin{bmatrix} x_{out} \\ y_{out} \\ w_{out} \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix} \begin{bmatrix} x_{in} \\ y_{in} \\ 1 \end{bmatrix}$$

and $x = \frac{x_{out}}{w_{out}}; y = \frac{y_{out}}{w_{out}}$

The homographic transformation adequately models the parameters of the transformation for projecting onto a flat surface. In the fitting process, all points are fitted to the model simultaneously using a non-linear least-squares fitting process.

Repeating this process for each projector results in a series of homographic transformations that transform projector coordinates to coordinates in camera space. This now lets us view the tiled display as a one-projector display with a (virtual) projector at the location of the camera. Knowing the configuration of the tiles, we can now determine the largest rectangle that fits inside the projected surfaces.

When the largest rectangle is found, intersection with each projector surface quad in camera space yields the effective surface. Using the inverse of the homographic transformation, the effective surface can be transformed back to projector space. Knowing the corner points of these quads in camera space is sufficient to generate a static mapping with which, for instance, a desktop application can be spanned geometrically correct on the entire tiled display.

To generate alpha masks for overlap gradients, we intersect the effective surface of a projector with its neighboring projectors. This results in small polygons that cover the overlap regions. Simple distance criteria can be used to mark the corner points of these polygons full (1.0) or black (0.0). Interpolating between these corner points results in gradients that should seamlessly fit in each overlap region on the display. Our implementation generates an alpha mask for each projector that is multiplied with the projected image in hardware (using texture mapping).

Finally, the homographic transformation can be converted to a 4x4 homogeneous 3D transformation matrix that can be concatenated to the matrix stack of the used 3D API (OpenGL, Direct3D), which performs the geometric calibration without impact on graphics performance. The output is then warped by the homographic transformation, and the resulting image is geometrically aligned.

The above procedure is implemented transparently in the Aura parallel scene graph library, producing seamless images for a tiled display.

References

- [1] Y. Chen, D. W. Clark, A. Finkelstein, T. C. Housel, and K. Li. Automatic Alignment of High-Resolution Multi-Projector Displays Using an Uncalibrated Camera. In *IEEE Visualization '2000*, pages 125–130, Oct. 2000.
- [2] F. Vos. *A System for Measuring, Modelling and Visualizing Corneal Shapes Based on Pseudo Random Encoding*. PhD thesis, Vrije Universiteit Amsterdam, 1998.