Advanced Visualization and Interaction Techniques for Large High-Resolution Displays

Sebastian Thelen[1]

1 **University of Kaiserslautern**
67653 Kaiserslautern, Germany
thelen@cs.uni-kl.de

Abstract

Large high-resolution displays combine the images of multiple smaller display devices to form one large display area. A total resolution that can easily comprise several hundred megapixels makes them suited for the visualization of data sets that could not be perceived entirely on desktop PCs or laptops due to their size. At the same time, user collaboration benefits from an extended screen area that facilitates interaction with screen contents as well as interaction among users. This paper discusses the challenges and opportunities of large high-resolution displays and examines ways to set up display clusters both in terms of hardware and underlying software technology. Furthermore, it investigates how to effectively harness the computational power and resources of rendering clusters to visualize giga-scale data sets. Last but not least, traditional interaction metaphors and their scalability to large displays as well as the effect of new techniques on the user experience are discussed.

Keywords and phrases Large High-Resolution Displays, HCI, Interaction, Visualization, Distributed Rendering

Digital Object Identifier 10.4230/OASIcs.VLUDS.2010.73

1 Introduction

Large high-resolution displays are getting common and have entered research laboratories, conference halls, and presentation rooms. Due to their size, they astonish the audience whenever used in tech demos or presentations. Exploiting their capabilities in order to be more than just giant displays showing scaled-up versions of existing applications is a challenging task. Programmers need reliable middleware to implement their applications without having to worry about low-level aspects of the system. Collaboration requires new interaction methods that scale with the size of the display and the size of the user group to offer intuitive ways to interact with each other and the application. Furthermore, since large displays are usually driven by a cluster of render nodes, sophisticated techniques are required to exploit the computational resources of the cluster and make use of the enormous resolution comprised in these systems for the visualization of giga-scale data sets.
The following sections give an overview of visualization and interaction techniques for large high-resolution displays. Each of the above mentioned aspects is addressed in more detail, and possible solutions to the challenges are presented.

2 System Aspects

This section discusses hardware- and software-specific aspects of large high-resolution displays. Typical display systems combine the resolution of multiple smaller devices (tiles) to form one large display area. Up to two tiles are driven by a single compute/render node, and nodes are

Visualization of Large and Unstructured Data Sets– IRTG Workshop, 2010.
Editors: Ariane Middel, Inga Scheler, Hans Hagen; pp. 73–81

OpenAccess Series in Informatics
OASICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

(a) (b) (c)

Figure 1 Possible large high-resolution display configurations: (a) monitor-based tiled display wall, (b) projector-based tiled display wall, (c) Cave Automatic Virtual Environment (CAVE™).

connected via high-speed network to share data and communicate with each other. Hence, high-resolution displays can be described as network-based distributed render clusters.

2.1 Large High-Resolution Display Systems

There exist various ways to build large high-resolution displays, mainly differing in the number and type of display devices used in the setup [10]. Figure 1 illustrates three of the most common large display configurations.

Monitor-based tiled display walls (see figure 1a) combine the resolution of multiple LCDs and easily achieve a total resolution of several hundred megapixels. Since LCD screens have become cheap, monitor-based systems are an affordable way to set up large high-resolution displays, and compared to other approaches, their calibration is rather simple and mainly consists of mounting all tiles on a proper monitor rack. A major problem of monitor-based systems are screen bezels, which lead to discontinuities in visualizations and result in an ever-present French window effect.

Projector-based systems (see figure 1b) consist of multiple computer projectors that are arranged in a grid and project onto a large screen. Compared to monitor-based tiled walls, projector-based systems are capable of forming a truly seamless display area, which requires a careful projector calibration. However, calibration is challenging, since projectors differ in terms of luminance and color temperature. Also, geometric distortions have to be considered, which are due to lens effects and their location relative to the screen.

Cave Automatic Virtual Environments (CAVEs™) are non-planar projector-based installations that consist of up to six screens representing the sides of a cube/box (see figure 1c). CAVEs™ are immersive display systems in which users are literally surrounded by the virtual environment. Often, they use additional devices, such as tracking suits or stereo glasses, to increase the realism of the virtual environment.

2.2 AnyScreen - A Distributed Rendering Library

Once the hardware is set up, a distributed rendering library is required to drive the displays and take care of synchronization between nodes, data distribution, frustum culling, and I/O events. Rendering libraries can be characterized in various ways. One of them is in terms of the graphics pipeline stage, in which geometry is assigned to the tiles of a display cluster [9]. Figure 2 illustrates the stages of the graphics pipeline and the conversion of geometry data into pixel data.

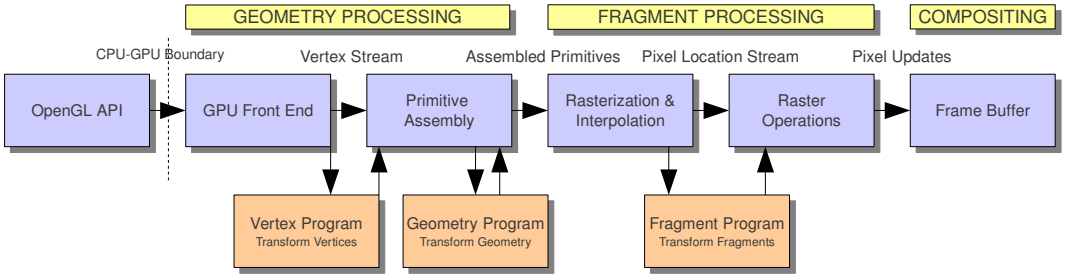


Figure 2 High-level representation of the graphics pipeline.

Sort-first libraries assign raw primitive data in the primitive assembly stage. This stage converts geometry data from three-dimensional object space into normalized screen space. After each node has been assigned its parts of data, the remaining pipeline stages are executed independently.

Sort-middle systems assign screen-space primitives between the primitive assembly stage and the rasterization stage.

Sort-last approaches assign actual pixel/fragment information in the rasterization stage. The rasterization stage is the last stage of the pipeline in which a series of fragment tests (e.g., scissor, alpha, and depth tests) are performed to determine whether and how a fragment is visualized as a pixel on the screen.

An alternative way to classify rendering libraries is in terms of the underlying execution mode [10]. Two different approaches have emerged: client-server and master-slave systems (see figure 3).

Client-server systems (see figure 3a) consist of a client node and a set of render servers. The client node is responsible for handling I/O events and is the only cluster node that runs an instance of the application. Render jobs are issued via network to the render servers, which means that the amount of network traffic depends on the complexity of a scene.

Master-slave systems (see figure 3b) execute an instance of the application on each node of the cluster. The master node in addition has to handle I/O events and forward these via network if necessary. Compared to client-server systems, the network traffic in master-slave setups is rather low, which makes this approach especially suited for the visualization of complex scenes.

AnyScreen [2] is a scalable and lightweight rendering library developed to drive arbitrary display configurations and support a variety of different stereo modes. The library consists of a set of C++ classes for managing synchronization and communication between executing threads and provides automatic viewport handling for each tile of a display cluster. AnyScreen implements a master-slave execution model, i.e., an instance of the application is executed on each compute node. Furthermore, a static sort-first approach is used to assign geometry to individual tiles in the primitive assembly stage. The library's visualization component is based on OpenGL and SDL (Simple Direct Media Layer), inter-node communication and synchronization have been implemented using MPI (Message Passing Interface). AnyScreen is flexible and has been used to drive a variety of display configurations, such as monitor-based tiled walls, stereoscopic Powerwalls, and different display-in-display configurations, such as combinations of 2D and 3D projector systems.

AnyScreen's advantage over other distributed rendering libraries is its high degree of versatility. CGLX [3], for instance, is a rendering framework with a similar architecture as AnyScreen. However, CGLX was specifically designed to drive monitor-based tiled systems and thus

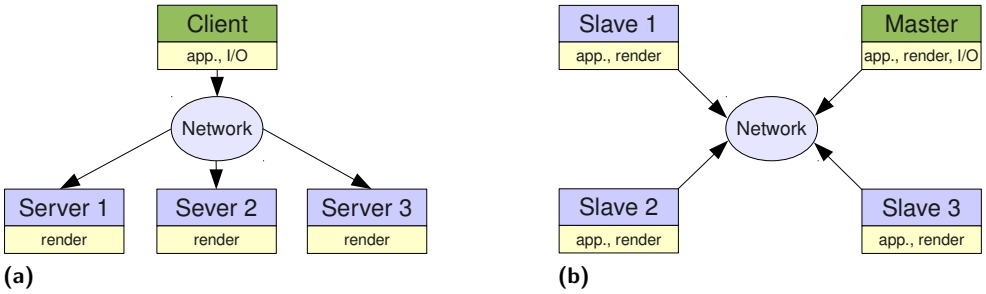


Figure 3 Execution modes: (a) client-server mode, (b) master-slave mode.

hardly adapts to more exotic display configurations. Chromium [5] is a client-server system supporting both sort-first and sort-last rendering. However, with increasing data size, network bandwidth may pose a limit to this technique, as clients constantly need to issue render requests to the servers.

2.3 Tiled++ - An Enhanced Tiled Hi-Res Display Wall

Monitor bezels are an inherent problem of monitor-based tiled display walls, causing effects similar to a French window and distracting users. Bezels either lead to discontinuities when they are completely ignored, i.e., lines are slightly offset when they cross the boarder between two adjacent screens, or they hide information when treated as overlays. Both approaches affect user perception and potentially lead to misinterpretations of a scene.
Tiled++ [4] is an approach that explicitly addresses the bezel problem. The idea is to enhance the bezel area of monitor-based display walls with the image of one or more computer projectors that are placed at arbitrary positions in front of the display wall. The projectors provide information that would normally be lost or disrupted on the bezel area, which for this reason has been covered with diffuse reflecting card board. In order to avoid optical interference, projectors only project onto the covered bezel area and spare the display area of the LC panels. They have been carefully calibrated using a camera-assisted homography-based calibration tool to compensate linear distortion effects that are due to a non-perpendicular placing of projectors in front of the wall (by placing projectors beneath the ceiling or at the side of a large display, one can prevent users from entering the projector beam and causing shadows).
Figure 4 shows an example that compares the results achieved with Tiled++ to those achieved when treating monitor bezels as an overlay to the scene. With the overlay method, bezels hide the company logo so that it is difficult to determine the type of the car or the manufacturer. Tiled++ provides the missing information on the bezel grid and the car can be easily identified.

3 Interaction Aspects

A major question when working with large high-resolution displays is how to interact with applications that run on a 200 megapixel tiled display wall or in a CAVE™? One can employ traditional input devices and metaphors designed for desktop PCs, such as keyboards, joysticks, computer mice, or tablets. Depending on the application or type of display that is used, however, one might consider using more sophisticated input devices that better fit

Figure 4 Tiled++ provides important image information that would otherwise be hidden or disrupted due to monitor bezels.

the needs of the user and allow to navigate data sets faster and more intuitively. Therefore, devices that are commonly used in virtual reality applications are full body tracking suits, data gloves, or head mounted displays.

When it comes to collaboration in front of large high-resolution displays, many input techniques suffer a couple of common issues. From an interaction point of view, the most severe one is that very often one person takes the lead and controls the application, while the rest of the group members depend on that one person navigating the data set. Furthermore, many approaches do not *scale* with the *size of the display* or the *size of user groups* working with the application. The mouse cursor metaphor, for example, hardly scales with the size of a display. In order to cope with the increasing distances that a cursor has to travel, one has to increase the mouse sensitivity, thereby making it harder for users to select small items on the display wall. Magnetic tracking on the other hand works fine for single users or very small groups, but equipping dozens of users with full body tracking devices is not feasible for obvious technical and financial reasons. Further questions that arise in the context of collaborative work with large displays are:

- How is it possible to support *user groups* and present information that is *customized* based on the status of a user (e.g., researcher and student)?
- How is it possible to deal with confidential information and implement a set of *security mechanisms*, so that not every bit of information can be accessed by every user?
- How can *independent data exploration* be realized for every user without running into scalability issues?

3.1 Large Display Interaction using Visual Tags

The interaction approach presented in this section explicitly addresses the problems and challenges discussed in previously. The technique is based on camera-enabled cell phones and two-dimensional barcodes (visual tags) acting as links between users and the application on a large display [12, 11]. The main idea is to label a set of data items using two-dimensional barcodes and display them in the application. Users employ their cell phone camera to scan and decode these barcodes with a special tag reader software installed on their mobile device. The information encoded in each tag is used to trigger a series of web services that, in their simplest form, display information about the data item on the cell phone screen. However,

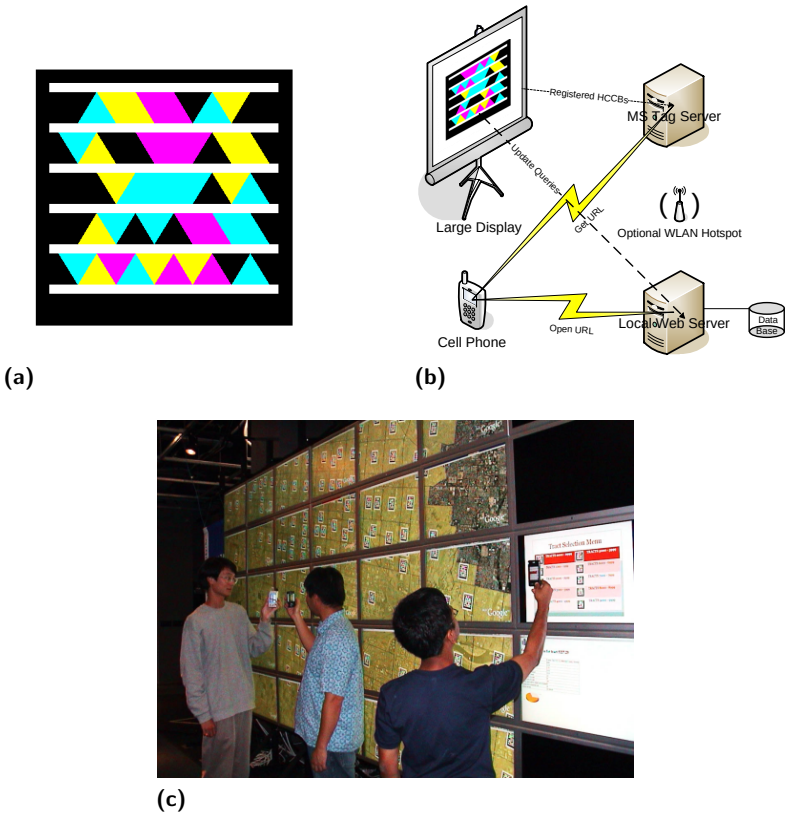
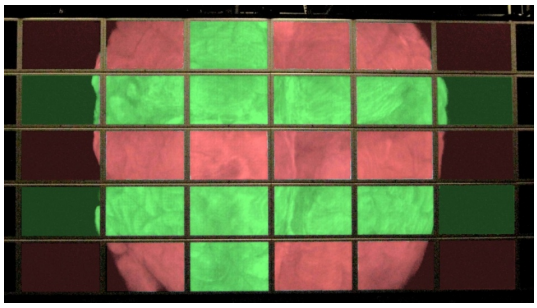


Figure 5 Tag-based interaction with large high-resolution displays: (a) two-dimensional HCCB, (b) general system setup, (c) collaboration in front of a 200 megapixel display.

more complicated interaction forms can be implemented that, for example, modify data sets or influence the visualization on the display.

The application presented here makes use of a special type of visual tag called *High Capacity Color Barcode* (HCCB) (see figure 5a). HCCBs have been invented by Microsoft (MS) and are supposed to be an alternative to already existing barcode systems. The general setup for an application using HCCBs as an interaction method is depicted in figure 5b. Data has been labeled with a set of barcodes and is visualized on a large display. The information encoded by each tag links to an entry on a MS tag server. The tag server stores the information that has been registered with the tag when it was created. At the current stage, this information is a unique URL to a web server in the local network. The cell phone retrieves the URL from the MS server in order to open it in a web browser. Querying the URL triggers a series of Python CGI scripts on the web server that generate the HTTP output to be displayed in web browser.

The approach is used in a Google Earth-based application on a fifty tile display cluster for the investigation of census data (see figure 5c). Census data consists of several hundred census tracts, each tract associated with a high-dimensional feature vector. The center of each census tract is labeled with a two-dimensional barcode that can be scanned in order to retrieve information about that tract. The system supports further interaction concepts, such as *information layers*, *public screens*, *description balloons*, and *level-of-detail information querying* that are also based on the HCCB approach (see Thelen et al. [12] for details).



(a)

(b)

(c)

Figure 6 Multi-resolution volume rendering on a 200 megapixel display wall: (a) tiles display at different levels of detail, (b) mandible data set displayed at wavelet level 1 (corresponds to $1/8 = 12.5\%$ level of detail), (c) mandible data set displayed at wavelet level 0 (corresponds to 100% level of detail).

Other smart phone-based interaction techniques featuring two-dimensional tags have been presented previously [1, 6, 7]. However, a majority of these methods seek to offer large display alternatives for navigation in typical window environments, i.e., selection, scrolling, and zooming. Instead, the focus of the HCCB approach is on exploiting two-dimensional codes to overcome common issues in collaborative large display environments.

4 Visualization Aspects

In order to handle the enormous resolution comprised in large display systems, tiled walls are usually driven by a cluster of render nodes that communicate via network. Render nodes are often built from off-the-shelf commodity components and drive up to two tiles of a display wall. They basically resemble modern desktop PCs, i.e., core 2 duo processors, 2-4 GB RAM, 512-1024 MB VRAM.

A major reason for using large display systems is their ability to display giga-scale data sets at a much higher level of detail than single monitor systems due to their pixel count. The Visual Human Project, for instance, provides 40 GB of volumetric data for the female cadaver. However, special techniques are required to visualize this enormous amount of data using the resources of the render cluster.

4.1 Handling Large Volumetric Data Sets

The volume rendering system depicted in figure 6a combines octree-based space subdivision with a (Haar) wavelet-based multi-resolution representation [8] to display large volumetric data sets at fairly interactive rates on a 200 megapixel tiled display wall. The algorithm to visualize data sets is divided into three stages:

- Per tile octree-based frustum clipping.
- Wavelet-based level of detail (LOD) determination.
- Direct volume rendering using 3D texture mapping.

The first stage of the algorithm determines for each tile of the display wall, which subvolume of the data has to be rendered on each tile and discards all other parts in order to make optimal use of system resources. The second stage of the algorithm determines the maximum level of detail, based on the Haar wavelet representation, that lets subvolumes still fit into the video memory of the respective render node. Once the subvolume has been transferred to the GPU's texture memory, 3D texture mapping is used to render the data.
As illustrated in figure 6a, the final rendering consists of multiple tiles, each displaying its part of the data set with highest possible quality. In figure 6a, all green tiles display data at wavelet level 0, corresponding to a level of detail of 100%. All red tiles display data at wavelet level 1, which corresponds to $1/8 = 12.5\%$ of the detail comprised in the original data. The difference in terms of visual quality is depicted in figure 6b and figure 6c for the mandible data set. The results shown in figure 6b correspond to what can be achieved on a single monitor PC without frustum clipping and multi-resolution representation by downsampling the entire data set to make it fit into the texture memory of the graphics card. Figure 6c shows the results achieved with the octree-wavelet data structure. Especially in the upper half, figure 6b is blurry and does not reveal as many details as figure 6c.

5 Conclusion and Future Work

Large high-resolution displays provide exciting research opportunities in various disciplines of computer science. In order to make users benefit from the applications executed on large displays, numerous challenges have to be addressed. The paper illustrated these benefits cannot be achieved without rethinking concepts of human computer interaction and developing new visualization techniques. At best, this is supported by reliable middleware abstracting from underlying system details and providing transparent programming interfaces to developers. First solutions have been proposed, yet the fact that most laboratories/institutes reimplement from scratch shows more research is required to provide standardized methods and concepts for setting up large high-resolution systems.

Acknowledgements This work was supported by the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) as part of the International Graduate School (International Research Training Group, IRTG 1131) in Kaiserslautern, Germany.

References

1 Rafael Ballagas, Michael Rohs, and Jennifer G. Sheridan. Sweep and point and shoot: phonecam-based interactions for large public displays. In *CHI '05: CHI '05 extended abstracts on Human Factors in Computing Systems*, pages 1200–1203, New York, NY, USA, 2005. ACM.

2 Matthias Deller, Sebastian Thelen, Daniel Steffen, Peter-Scott Olech, Achim Ebert, Jan Malburg, and Joerg Meyer. A Highly Scalable Rendering Framework for Arbitrary Display and Display-in-Display Configurations. In *CGVR*, pages 164–170, 2009.

3 Kai-Uwe Doerr and Falko Kuester. CGLX Project (accessed September 16, 2010). `http://vis.ucsd.edu/~cglx/`.

4 Achim Ebert, Sebastian Thelen, Peter-Scott Olech, Joerg Meyer, and Hans Hagen. Tiled++: An enhanced tiled hi-res display wall. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):120–132, 2010.

5 Greg Humphreys, Mike Houston, Ren Ng, Randall Frank, Sean Ahern, Peter D. Kirchner, and James T. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters.

6 Seokhee Jeon, Jane Hwang, Gerard J. Kim, and Mark Billinghurst. Interaction with large ubiquitous displays using camera-equipped mobile phones. *Personal Ubiquitous Comput.*, 14(2):83–94, 2010.

7 A. Madhavapeddy, D. Scott, D Sharp, and E. Upton. Using camera-phones to enhance human-computer interaction. In *Sixth International Conference on Ubiquitous Computing (Adjunct Proceedings: Demos)*, 2004.

8 J. Meyer, R. Borg, B. Hamann, K.I. Joy, and A.J. Olsen. Network-Based Rendering Techniques for Large-Scale Volume Data Sets. In *Farin, G., Hamann, B. and Hagen, H., eds., Hierarchical and Geometrical Methods in Scientific Visualization*, pages 283–296, Heidelberg, Germany, 2002. Springer-Verlag.

9 Steven Molnar, Michael Cox, David Ellsworth, and Henry Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, 1994.

10 Tao Ni, Greg S. Schmidt, Oliver G. Staadt, Robert Ball, and Richard May. A Survey of Large High-Resolution Display Technologies, Techniques, and Applications. In *VR '06: Proceedings of the IEEE conference on Virtual Reality*, page 31, Washington, DC, USA, 2006. IEEE Computer Society.

11 Sebastian Thelen, Joerg Meyer, Achim Ebert, and Hans Hagen. A 3D Human Brain Atlas. In *Modelling the Physiological Human. Proceedings of the second 3D Physiological Human Workshop*, 2009.

12 Sebastian Thelen, Joerg Meyer, Ariane Middel, Peter Scott Olech, Achim Ebert, and Hans Hagen. Tag-based interaction with large high-resolution displays. In *Proceedings of the 4th IASTED International Conference on Human-Computer Interation (IASTED-HCI)*, pages 356–363, 2009.