Ultrascale Collaborative Visualization Using a Display-Rich Global Cyberinfrastructure

Byungil Jeong ■ *University of Texas at Austin*

Jason Leigh, Andrew Johnson, Luc Renambot, Maxine D. Brown, Ratko Jagodic, Sungwon Nam, and Hyejung Hur ■ *University of Illinois at Chicago*

The Scalable Adaptive Graphics Environment (SAGE) is high-performance graphics middleware for ultrascale collaborative visualization using a display-rich global cyberinfrastructure. Dozens of sites worldwide use this cyberinfrastructure middleware, which connects high-performance-computing resources over high-speed networks to distributed ultraresolution displays.

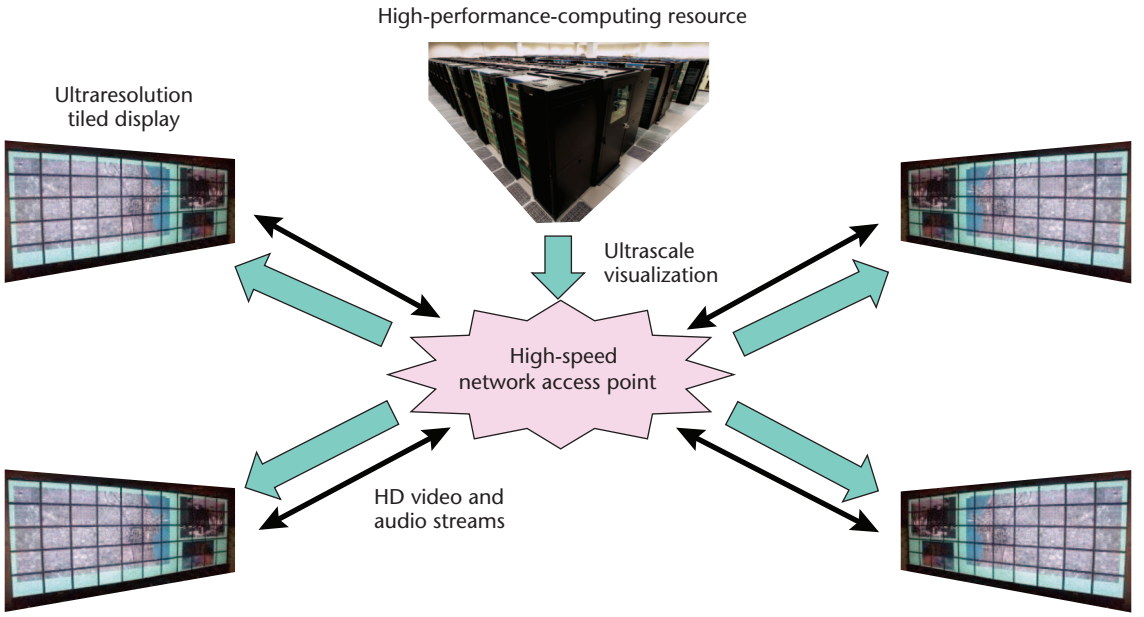
As data that sensors and simulations collect exceeds a PC's or average computing cluster's capacity, scientists must rely on shared cyberinfrastructure (CI) for acquiring, analyzing, and visualizing that data. CIs are federally funded research environments that let scientists access remote sensor data, unique instruments, major data stores, and high-performance computing and networking resources shared cross-institutionally, nationally, and internationally. Such environments include TeraGrid, the BlueWaters Petascale Facility, the European Organization for Nuclear Research (CERN) Large Hadron Collider, and global networks with funding from the US National Science Foundation (NSF) International Research Network Connections program.

Ultrascale data in geoscience, atmospheric science, astrophysics, and bioscience is driving the growing use of high-resolution displays as visualization tools. Furthermore, scientific problems involving complex data and enormous scale often require remote, interdisciplinary collaboration. Remote team members need visualization tools that facilitate communication, collaboration, and discovery, whether the collaborators are in different cities or different countries.

These trends motivated us to investigate a unified hardware and software environment to support display-rich global collaboration. In this environment, collaborators generate very-high-resolution visualizations on shared CI and distribute the results over high-speed networks to heterogeneous, high-resolution tiled displays at the collaborating end points. A tiled display is a large, high-resolution display system comprising an array of LCD panels that a computer cluster drives. Each computer typically drives one to four panels, on the basis of its graphics capability. In the *SAGE* (Scalable Adaptive Graphics Environment) model, tiled displays typically have high-speed (multigigabit) network connectivity to remote high-performance-computing resources. Multipoint high-definition (HD) videoconferencing capability on high-resolution tiled displays combines the end points into a virtual collaboratory. Figure 1 illustrates the shared CI architecture and a typical collaborating end point, a 20-megapixel, LCD-based tiled display.

Over the past six years, the Electronic Visualization Laboratory (EVL) at the University of Illinois at Chicago (UIC) has spearheaded construction of the OptIPlanet Collaboratory, a persistent display-rich CI for global-scale distributed visualization. This hardware environment connects more than 40 ultra-high-resolution tiled displays (some over 100 million pixels) via optical networks. We dubbed these networked tiled-displays *OptIPortals* because

 Published by the IEEE Computer Society



(a)

(b)

Figure 1. A virtual collaboratory. (a) The display-rich distributed collaboration environment uses a shared cyberinfrastructure. (b) A 20-megapixel, LCD-based tiled display is a typical collaborating end point.

we based the research on the NSF OptIPuter project's outcomes.[1]

For the software environment, we developed and deployed SAGE[2] and Visualcasting.[3] SAGE is cross-platform middleware that works in concert with many visualization applications, letting users simultaneously access, stream, and juxtapose multiple high-resolution visualizations in windows on one or more OptIPortals. Visualcasting is a SAGE network service that supports large-scale collaborative visualization in the OptIPlanet Collaboratory. It broadcasts high-resolution visualizations and videos to multiple OptIPortals in real time. Using Visualcasting, collaborators can see and interact with the same visualizations while communicating with each other through HD videoconferencing.

SAGE and Visualcasting

Fundamentally, SAGE leverages a shared CI to convert petascale data into very-high-resolution visualizations. Because networking costs are dropping faster than computing and storage costs, acquiring higher-bandwidth networking to a shared CI is cheaper than installing and maintaining

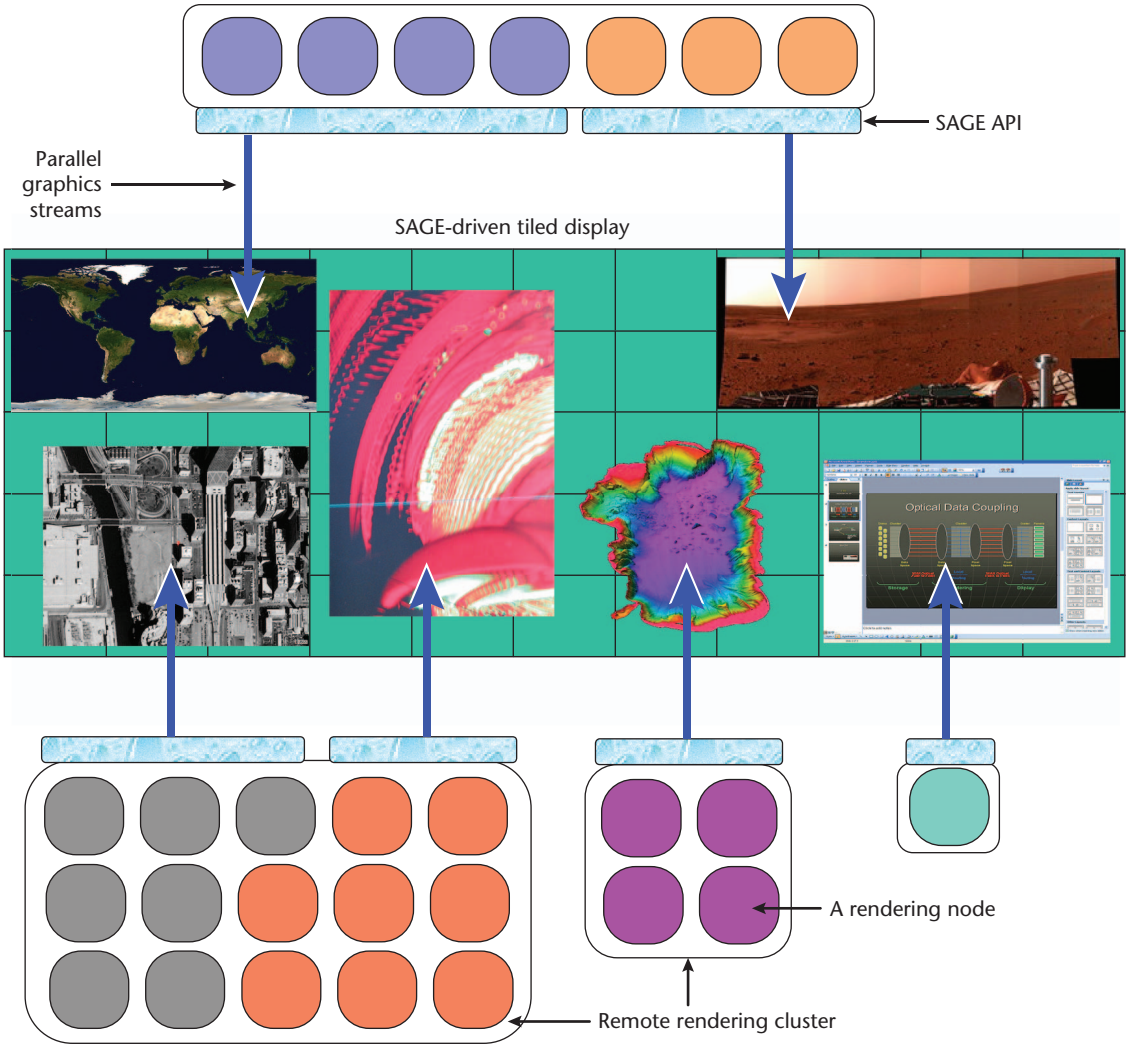


Figure 2. The SAGE (Scalable Adaptive Graphics Environment) architecture. Each visualization application runs on nodes in a distributed rendering cluster. The node color represents the application running on the node. SAGE manages the parallel graphics streams between the rendering nodes and tiled display nodes.

local computational resources. SAGE assumes users have thin-client computers with high-speed network connectivity driving tiled displays.

This model doesn't require high-end computing and graphics capability on the client side. CI-driven large-scale remote clusters process data and render visualizations. Then, SAGE streams the visualizations as pixels to the clients at user-requested resolutions. Furthermore, SAGE provides mechanisms to capture and stream HD video and audio directly from real-time sources.

Additionally, SAGE lets distributed visualization applications running on separate and distant rendering clusters stream their visualizations in real time to the same tiled display wall (see Figure 2). SAGE works with parallel rendering algorithms so that independent remote cluster nodes create pieces of the overall image. The system stitches these pieces together in real time on the display.

The SAGE user interface lets users interact directly with the display and manipulate multiple visualizations on it. Behind the scenes, SAGE intelligently routes the pixels from the rendering source to the nodes that display the pixels for that portion of the display. This capability lets scientists juxtapose multiple high-resolution visualizations side-by-side. Scientists then can compare the results of simulations with various parameters or compare sensor and simulation data to determine how well the simulation models the phenomenon under investigation.

Visualcasting extends this model to support distance collaboration among high-resolution tiled displays (see Figure 1). Visualcasting is a visualization distribution scheme that uses PC clusters and high-speed networks without requiring traditional router-based multicasting, which is difficult to deploy. This architecture uses SAGE Bridge software, which runs on a high-performance PC cluster strategically located at a high-speed network

access point (such as StarLight in Chicago). The SAGE Bridge redistributes visualization—the video and audio streams SAGE captures—to multiple end points (see Figure 3). This process lets collaborators share their visualization applications and videos on local and remote displays while communicating through multipoint HD videoconferencing.

SAGE Pixel Streaming

To transport image data, SAGE uses pixel-block streaming and a pixel pipeline architecture.

Pixel-Block Streaming

One obvious approach to distributing an application's output image over a tiled display is partitioning the image according to each tile's application layout and streaming the fragments to appropriate tiles. However, this approach can't support Visualcasting efficiently. Visualcasting distributes the source image to arbitrarily sized and positioned output windows on the displays. So, using that approach would require separately partitioning the source image for each destination's tiled display. As the destinations and applications in a Visualcasting session increase, the partitioned image fragments rapidly increase. The memory copies that the system requires to create these partitions incur significant system overhead, limiting the Visualcasting service's scalability.

To address this problem, SAGE streams uniformly sized pixel blocks rather than arbitrarily sized image fragments (see Figure 4). SAGE generates the pixel-block routing table, such as in Figure 4b, by comparing the pixel-block and tile layouts. It continues to send the pixel blocks according to the table until the window layout changes. So, the system must recalculate the table only when a user moves or resizes a window. SAGE uses each block's frame ID to synchronize the pixel-block display. It then uses each block's configuration ID to ensure that the window layout it uses to locate pixel blocks on the display is consistent with the pixel-block routing table it uses for sending pixel blocks.

This pixel-block streaming relies on routing pixel blocks rather than arbitrarily partitioning the application image (see Figure 5). For Visualcasting, the SAGE Bridge routes pixel blocks to each end point, requiring neither memory copies nor additional image-buffer allocation. This enables Visualcasting to scale with the number of end points and applications until the SAGE Bridge cluster's total bandwidth reaches its maximum. However, users can relieve this constraint by adding more resources (bridge nodes and network bandwidth) to the cluster.

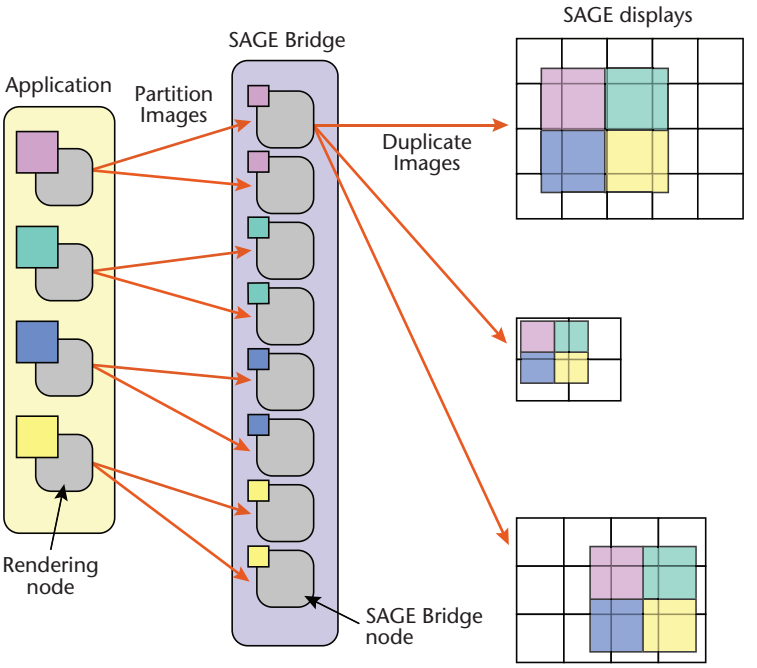
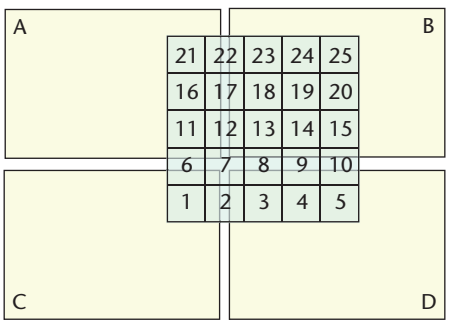


Figure 3. The SAGE Bridge. This cluster accepts parallel visualization streams, replicates them, and distributes them to multiple, SAGE-driven tiled displays. The window operations on each display are independent. An application can have a different window layout on each display.



(a)

1	C	11	A
2	C,D	12	A,B
3	D	13	B
4	D	⋮	
5	D	⋮	
6	A,C	21	A
7	A,B,C,D	22	A,B
8	B,D	23	B
9	B,D	24	B
10	B,D	25	B

(b)

Figure 4. An example of pixel block partitioning. (a) SAGE partitions an application image into a 5 × 5 pixel-block array and places the array over four tiles (display nodes), A, B, C and D. It assigns each block a unique ID from left to right and bottom to top and uses the ID to locate the pixel block on the tiled display. (b) Using the pixel block routing table, SAGE sends each pixel block to tiles needing any portion of the block's data.

Another advantage pixel-block streaming has over image partitioning is shorter window operation latency. For every window operation, image partitioning requires repartitioning the application images and reallocating the image buffers for the partitioned image fragments. In contrast, pixel-block streaming requires updating only the pixel-block routing table. This method reduces the window operation latency enough to enable animated moving of an application window on a tiled display by combining multiple small window-repositioning operations.

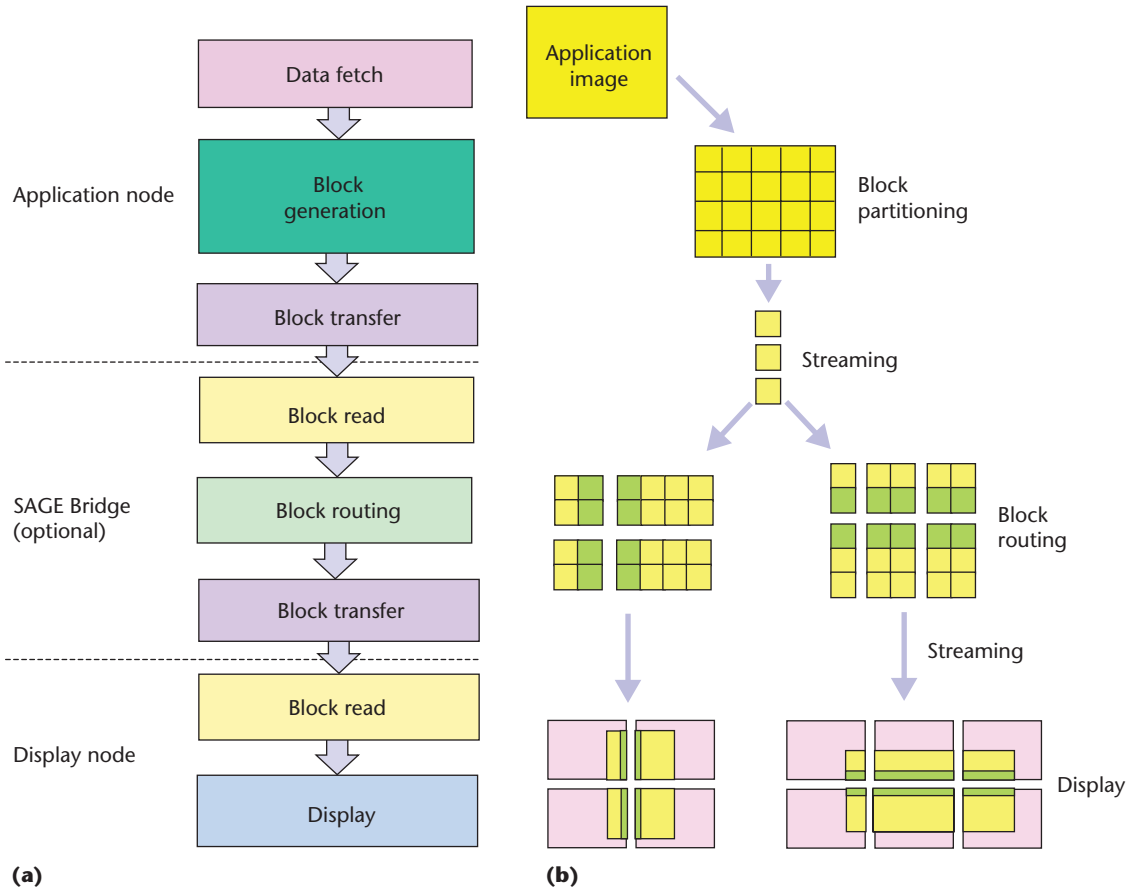


Figure 5. Pixel-block pipeline. (a) Without the SAGE Bridge, the SAGE pixel pipeline comprises the three stages on each application node and the two stages on each display node. The SAGE Bridge adds three optional stages to the pipeline. A pixel data buffer or network buffer connects each stage to the next so that all stages can run concurrently. (b) The diagram shows pixel-block streaming along the pipeline.

Pixel-block streaming uses more network bandwidth than image partitioning for an image frame when pixel blocks cover multiple tiles, such as the green blocks in Figure 5b. Because this overhead increases with pixel-block size, we typically use a small block, such as 32 × 32 pixels or 64 × 64 pixels, to keep the overhead trivial.

The SAGE Pixel Pipeline

SAGE transfers pixel data from an application to a tiled display through the SAGE pixel pipeline (see Figure 5). The SAGE API receives an application's output image in the data fetch stage, generates pixel blocks from the image, and routes them to their destinations (display nodes) in the block generation stage. For a parallel application, SAGE synchronizes the block generation stage across the application nodes for seamlessly repositioning and resizing application windows. Without synchronization, each node might have a different pixel-block routing table for the same image frame, thus updating the table at different frames during window operation. This timing would result in broken imagery on the display. To guarantee synchronous updates, SAGE delivers window layout messages through synchronization signals to each rendering node, which then updates its local routing table according to these messages.

The pixel blocks from the block generation stage stream to display nodes in the block transfer stage. For network streaming, SAGE uses both TCP and the User Datagram Protocol (UDP). TCP provides reliable, ordered data delivery to the destination host. Because TCP flow control relies on feedback messages from the destination host, it performs well over local-area networks with short round-trip times but not over wide-area networks with long round-trip times. UDP shows much higher and more stable performance over wide-area networks than TCP does, but it doesn't guarantee reliable, ordered data delivery. Applications using UDP should employ a mechanism to guarantee safe data delivery. In SAGE, we typically use TCP for local streaming and UDP for remote streaming.

For TCP, SAGE sends a group of pixel blocks using a *vectored* I/O system call. Vectored I/O is an I/O method that supports reading a single data stream to multiple buffers or writing multiple buf-

fers to a single data stream through a system call. Because SAGE stores pixel data in nonsequential memory locations (blocks), it uses vectored I/O operations to send and receive multiple blocks at a time. A vectored I/O operation replaces many ordinary read or write operations, relieving system call overhead and increasing network performance using less CPU time than streaming block by block.

For UDP, however, we experienced more data loss with block-group streaming than streaming block by block. The 64-Kbyte maximum UDP buffer size limits the data quantity a system call can send. This, in turn, limits the vectored I/O operation's advantage. Although TCP has its own flow control mechanism, UDP requires application-level flow control to minimize data loss. Streaming block by block lets SAGE better control pixel-block flow. So, recent SAGE versions send each pixel block separately for UDP. The UDP streaming's data loss produces visual artifacts on the display, but they're visible for only a short time if the application runs at an interactive frame rate, typically higher than 24 frames per second (fps). Minimal momentary data loss is acceptable, except in applications requiring high-precision images, such as medical applications. A detailed description of SAGE's UDP flow control algorithm appears elsewhere.[3]

SAGE reads the pixel blocks from the network and inserts them into the pixel-block buffer at the block read stage. Each display node creates a separate network read thread for each application to concurrently read each application's pixel streams. For a parallel application, SAGE at this stage serializes multiple pixel streams to a display node into a single stream. Then, in the display stage, SAGE fetches the pixel blocks from the buffer and synchronously updates them on the display. Each display node creates two OpenGL (Open Graphics Library) textures for each application and uses them for drawing and scaling multiple application images on a screen.

The coupled textures work as double buffers for each application: one for drawing on the screen and the other for receiving a new application image. To synchronously update multiple application images across multiple tiles, SAGE employs a synchronization server that monitors the coupled textures' status at each node. When a new image is ready at every node, the server sends synchronization signals that trigger the texture swapping and screen refresh. Details on the SAGE synchronization algorithm appear elsewhere.[3]

The block read and block transfer stages on the SAGE Bridge (for Visualcasting) are identical to the ones we've described in this section. The block-routing stage uses multiple threads to route pixel blocks to multiple end points concurrently and independently. To avoid unnecessary memory copies, each thread has a reference to the pixel blocks rather than a local copy. Each pixel block has a reference count to ensure that every thread has sent the pixel block before returning it to the pixel-block pool in the SAGE Bridge.

User Interaction

Because SAGE effectively turns any tiled display into a collaborative environment, enabling multiuser interaction was imperative. Initially, users could use a cross-platform, desktop-based user interface (SAGE UI) from their laptops to manipulate windows, share multimedia files through drag-and-drop, share desktops, or share applications with remote displays. However, direct display interaction is often more appropriate, especially when users want to be mobile in front of the display for more natural physical navigation. So, we developed Direct Interaction Manager (DIM), which manages numerous heterogeneous physical-interaction devices.

Because SAGE effectively turns any tiled display into a collaborative environment, enabling multiuser interaction was imperative.

Supporting heterogeneous devices is crucial because no device has emerged as the most appropriate for large high-resolution displays. We've experimented with devices such as a traditional mouse, Gyromouse, trackball, joystick, Wiimote, touch screen, and six-degree-of-freedom CAVE (CAVE Automatic Virtual Environment) wand. DIM also handles drawing user interface objects (widgets) on the display and performs device event handling while enabling true simultaneous multiuser interaction.

SAGE Applications

The following major applications highlight different ways of using SAGE.

Multimedia Applications and Scientific Animation

As 10-megapixel cameras become prevalent, users can easily create high-resolution images that desktop screens can't accommodate at their native resolutions. The SAGE ImageViewer provides

a simple way to load and view a high-resolution image file at its native resolution on a tiled display. It supports most conventional image formats, including JPEG, PNG, GIF, and TIFF.

For video and animation, Mplayer is a cross-platform media player that supports a range of media file formats, such as MPEG, VOB (Video Object), AVI (Audio Video Interleave), VIVO (Video In Video Out), and RealMedia. We implemented an MPlayer plug-in for SAGE so that users can display any MPlayer-supported media file. Mplayer, however, is limited to conventional resolution and lossy media formats. Scientific uses require higher-resolution, lossless animation.

Scientific animation is essential for teaching and disseminating scientific-visualization results.

To support higher-resolution animation, the US National Center for Supercomputing Applications (NCSA) developed Bitplayer, a high-resolution animation playback tool. We ported Bitplayer to SAGE and enhanced it to support uncompressed (24-bit RGB) and compressed (DXT texture format) content at any resolution. DXT is a group of lossy texture compression algorithms S3 Graphics developed, under the name S3 Texture Compression. SAGE supports DXT1, which converts 4 × 4 input pixels into 64 bits of output. This conversion results in an 8:1 compression ratio for the 32-bit RGBA (RGB and alpha) pixel format. Most modern GPUs support DXT decompression. SAGE supports transferring and displaying DXT format image data onto a tiled display. DXT compression lets users stream full HD (1,920 × 1,080 pixels) video over a gigabit connection and 4K (4,096 × 2,160 pixels) animations without needing a high-end storage system or an expensive codec.

Scientific animation is essential for teaching and disseminating scientific-visualization results. Preserving advanced simulation and rendering's intricate details, such as for climate or molecular simulations, requires very high resolution—HD at minimum, but preferably 4K-resolution. As we've demonstrated at numerous workshops and conferences over the past four years, Bitplayer effectively supports this need.

High-Definition Video and Audio Streaming

We developed HD video and audio streaming (HDS), which streams low-latency, uncompressed full-HD video and audio from a camera using an HDMI (High-Definition Multimedia Interface) capture card. HDS passes the captured video frames in the YUV422 format directly to SAGE for streaming. The frame rate can control total bandwidth use, which is usually under 1 Gbit per second (Gbps) without significant quality loss. We used HDS in the successful Visualcasting demonstrations and experiments we describe in the next section.

Several organizations have developed HD video and audio streaming solutions using SAGE: the University of Queensland (HDS with DXT compression), ResearchChannel (iHDTV), Masaryk University's ANTLab (UltraGrid), and Nippon Telegraph and Telephone (I-Visto, a multirate HD-video transmission system).

Ultraresolution Image Viewer

MagicCarpet is a cluster-based interactive ultraresolution image viewer for scalable tiled displays. It uses preprocessed multiresolution images that reside on a local disk and loads appropriate detail at different zoom levels by paging texture data to video memory on demand.

MagicCarpet can work as a standalone application or with SAGE to stream the rendered image to local or remote tiled displays. The SAGE ImageViewer is useful for loading high-resolution image files onto a tiled display. However, scientists who deal with extremely high-resolution images, such as aerial and satellite imagery (365K × 365K pixel maps) or tiled electron microscope images, require an advanced tool to explore their large-scale image data.

Parallel Real-Time 3D Rendering

ParaView is a parallel application for visualizing and interacting with large-scale 3D data on a remote visualization cluster through an interactive client on a laptop or desktop computer.[4] However, the clients' screen resolution and available network bandwidth constrain the visualization's resolution and interactivity. ParaView itself can support high-resolution visualization on a tiled display, but it has two limitations. First, the rendering nodes must directly drive the tiled display; ParaView doesn't support remote visualization for a tiled display. Second, it allows visualizing only one data set at a time, occupying the entire display.

To overcome these limitations, we integrated ParaView into the SAGE framework. This integration enables large-scale remote visualization using ParaView on a high-resolution tiled display. ParaView enables distributed visualization of a very large data set on a remote rendering cluster. SAGE streams the high-resolution visualization over a

Figure 6. Multipoint videoconferencing. The team at the Electronic Visualization Laboratory in Chicago holds an HD videoconference using Visualcasting with (clockwise from top left) the Gwangju Institute of Science and Technology, University of Michigan, SARA Supercomputing and e-Science Support Center, and Korea Institute of Science and Technology Information.

high-speed network in parallel to a user's display. Users can employ a ParaView client to interact with the visualization.

Accomplishments

Multiple organizations and user communities have successfully deployed and demonstrated SAGE.

Streaming from Distributed Rendering Sources

At the iGrid 2005 workshop, we simultaneously streamed various content, including

- real-time parallel volume-rendered images of the Visible Korean Human from the Korea Institute of Science and Technology Information (KISTI) over KreoNet2,
- NCSA's prerendered HD tornado simulation animations from the SARA Supercomputing and e-Science Support Center in Amsterdam over SURFnet,
- 1-foot-resolution aerial photographs of New Orleans in the aftermath of Hurricane Katrina, and
- live HD video from EVL over CAVEwave/National LambdaRail.

We allocated 10 Gbps for this iGrid experiment, which ran for six hours over three workshop sessions. SAGE achieved 9 Gbps of sustained throughput, transferring approximately 24 Tbytes of dynamic image data. This was the first successful international demonstration showing SAGE could enable and manage simultaneous real-time streaming of distributed, high-resolution visualization applications from distributed rendering sources to a large-scale tiled display.

Visualcasting Experiments and Demonstrations

On 18 April 2008, Visualcasting enabled HD videoconferences among five international sites. Figure 6 shows the videoconference from EVL's viewpoint. This was the first successful multipoint videoconference among globally distributed tiled displays. Visualcasting enabled casual conversation among all the participants through its short-latency uncompressed HD video and audio distribution. Each end point sent an audio stream and a full HD camera live feed at a rate of 17 to 18 fps and 0.7 Gbps network bandwidth to two SAGE Bridge nodes at StarLight. Each end point received multiple HD video streams according to its capacity. The total Visualcasting throughput was 9.2 Gbps. This experiment showed that we could realize high-resolution image multicasting on the order of tens of gigabits per second by using a PC cluster connected to high-speed networks.

At Supercomputing 2008, EVL and some OptIPlanet Collaboratory partners demonstrated Visualcasting's full capabilities throughout the week. We streamed full HD video, audio, and 4K scientific animations among three booths on the show floor (SARA, KISTI, and EVL/California Institute for Telecommunications and Information Technology/San Diego Supercomputing Center), UIC, the University of Michigan, Masaryk University, and the University of Queensland. We thus created a sustained global teleconference sharing massive-data-set visualizations (see Figure 7).

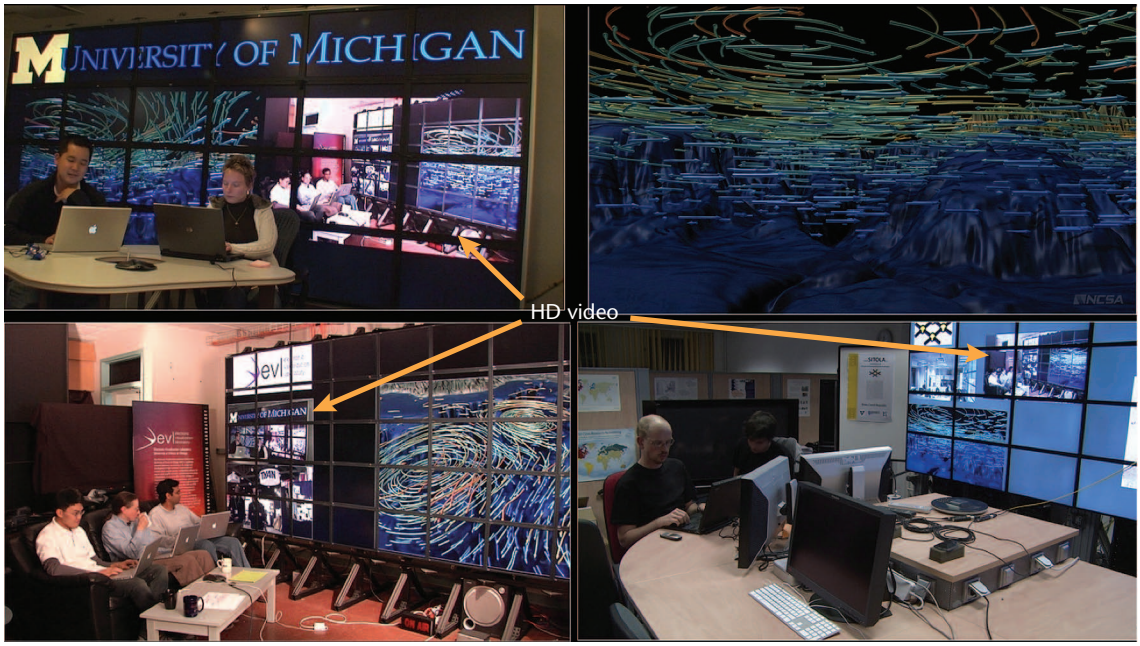


Figure 7. Three remote participants, (clockwise from bottom right) Masaryk University, the University of Illinois at Chicago, and the University of Michigan, collaborate through Visualcasting at Supercomputing 2008. The participants can view each other as well as the scientific visualization in the top-right pane.

We used UDP for all the experiments and demonstrations we describe in this section.

SAGE User Communities

More than 40 OptIPlanet sites have built high-resolution display walls and adopted SAGE (Figure 8 shows a few). These sites include universities, research institutions, and companies, both nationally and internationally.

Texas Advanced Computing Center. The Texas Advanced Computing Center (TACC) at the University of Texas at Austin has built Stallion, the world's largest high-resolution tiled display. It's a 307-Mpixel display comprising 75 (15 × 5) Dell 30-inch flat-panel monitors (see Figure 8b). TACC adopted SAGE as the tiled-display manager and became one of the most active SAGE user sites.

SAGE has successfully supported scientific visualizations on Stallion. TACC's Ranger, one of the world's largest computing systems for open science research, and Spur, the high-end remote-visualization system tightly coupled with Ranger, generate these visualizations. TACC is interested in expanding SAGE's capability and has recently started a partnership with EVL to investigate opportunities as SAGE development progresses.

University of Michigan. The university's School of Information (SI) has been an OptIPlanet Collaboratory partner since 2005. As such, SI has enthusiastically participated in various international research experiments and demonstrations of Visualcasting. SI researchers have investigated how SAGE can support collaboration in research and education on the campus using the Virtual Space Interaction Testbed (VISTI; http://visit.si.umich.edu). This SI initiative aims to apply advanced CI capabilities to the challenges of effective distance collaboration.

SI has deployed four OptIPortals across three departments. This implementation lets researchers and students interact with remote collaborators and visualization on a total of 235 Mpixels of digital canvas. SAGE has allowed collaboration in such disciplines as atmospheric science, computational social science, civil engineering, and humanities.

SARA Computing and Networking Services. An OptIPlanet partner since 2004, the e-Science Support Center of SARA, the Dutch supercomputer center, has contributed greatly to SAGE development, transatlantic tests, and demonstrations. SARA also supports SAGE use in the broader Dutch e-science community. For example, the Dutch Climate Research Institute has performed several extremely large-scale climate simulations on numerous European supercomputers. These simulations created hundreds of terabytes of data, which SARA stored. Climate institute researchers selectively visualize the data sets at SARA and use SAGE to stream and display the results to their tiled displays. This practice has become a part of their daily workflow.

The University of Amsterdam's Bioinformatics Institute uses SAGE to stream and display very-large-scale microarray data over optical networks.

Figure 8. OptIPortals and SAGE. (a) At EVL's CyberCommons, a user is directly interacting with the SAGE-driven tiled display. (b) The TACC's SAGE-driven Stallion is the world's largest tiled display. (c) The University of Michigan's collaborative ultraresolution environment uses SAGE. (d) At SARA's National Science Day demonstration, SAGE shows scientific animations and 4K movies on SARA's tiled display.

SARA, the European Space Agency, and the Dutch Air Force are collaborating on a project to reduce collision danger between migrating birds and airplanes using a complex simulation. They use SAGE to remotely disseminate simulation results to heterogeneous tiled displays at various institutes.

Louisiana State University. In 2009, Louisiana State University (LSU) placed first at the IEEE International Scalable Computing Challenge at the IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing. LSU demonstrated a scalable, end-to-end, interactive system for simulating and visualizing black holes.

The LSU researchers ran the simulation on 2,048 cores of TACC's Ranger cluster for 160 hours, generating 42 Tbytes of data. They transferred this data over the 10-Gbps Louisiana Optical Network Initiative network to rendering nodes at an LSU visualization cluster. A parallel renderer at LSU used GPU acceleration to render images. The researchers used SAGE to stream the visualization result to the final display.

Future Challenges

SAGE is a research prototype that must now transition to a hardened technology because it's in high demand in the SAGE user community. To achieve this goal, we must address several problems.

SAGE is a research prototype that must now transition to a hardened technology because it's in high demand in the SAGE user community.

Enhancing 3D Parallel Visualization in SAGE

Although SAGE is designed to support 3D parallel visualization applications for large-scale data visualization, its users face two primary hurdles to fully using this capability. First, the variety and number of SAGE-enabled 3D parallel visualization applications is limited, and they require complex user configurations. Second, users prefer to stay with visualization tools they're familiar with rather than switch to a new SAGE-enabled visualization tool.

To address these problems, we'll integrate widely used, massively parallel visualization tools such as VisIt (https://wci.llnl.gov/codes/visit/home.html)

SAGE versus the Competition

Several well-known tiled-display middleware systems exist, including Chromium,[1] DMX (Distributed Multihead X Project; http://dmx.sourceforge.net), Equalizer,[2] and CGLX (Cross-Platform Cluster Graphic Library).

In Chromium, one or more servers intercept OpenGL graphics primitives from an unmodified application and stream them to clients driving the tiled-display wall. These clients then must render the graphics primitives on their GPUs.

DMX provides a single unified X Window desktop by coupling multiple X servers on multiple machines. In DMX, a master node distributes X Window primitives to other client nodes, which render the visualization. This lets users view general X applications on a tiled display, but not hardware-accelerated OpenGL-based applications. However, DMX can work with Chromium to provide a complete desktop solution across multiple displays. DMX renders an application's original GUI on the tiled display without any source code change, whereas Chromium renders the 3D graphics windows.

Equalizer offers a hybrid approach in which the user can combine various rendering techniques. It supports screen-space, database, time-multiplex, pixel, and stereo task distribution. Users can select parallel compositing algorithms.

CGLX doesn't distribute graphics primitives, but it runs the same copies of an OpenGL-based application on all clients and replicates visualization data on all the clients. Users must modify applications to support CGLX. The amount of graphics data it can render is limited to the GPUs' and onboard main memory's capacities.

Table 1 summarizes the key differences among SAGE and the middleware we just described. Chromium's chief advantage is that it requires no modification to an existing OpenGL program. However, it works only with OpenGL-based applications, and the entire tiled wall can run only one application at a time. DMX works with various applications but supports only Unix-based (Linux) applications. Furthermore, the master node becomes a bottleneck as the number of running applications increases. In all cases, the local graphics card's capability constrains a clients' ability to visualize large-scale data.

SAGE has four chief advantages:

- The resolution and frame rate of visualization streams can scale with the number of rendering and display nodes.[3]
- Multiple applications can run simultaneously, and users can organize them as windows on a tiled display.
- Applications needn't be OpenGL-based.
- SAGE works for Linux, Mac, and Windows applications.

SAGE doesn't require powerful graphics cards at the clients; rather, it leverages networked shared CI to perform ultrascale visualization. Moreover, SAGE is completely independent of the method for generating applications' pixels (software or hardware rendering, GPU or many-core systems, or Mac/Windows/Linux OS). So, it's future-proof in a rapidly changing technology landscape. Finally, Visualcasting enables simultaneous visualization distribution to multiple tiled displays.

References

1. G. Humphreys et al., "Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters," *Proc. Siggraph*, ACM Press, 2002, pp. 693–702.
2. S. Eilemann, M. Makhinya, and R. Pajarola, "Equalizer: A Scalable Parallel Rendering Framework," *IEEE Trans. Visualization and Computer Graphics*, May 2009, pp. 436–452.
3. B. Jeong et al., "High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment," *Proc. 2006 ACM/IEEE Conf. Supercomputing* (SC 06), IEEE CS Press, 2006, article 108, doi:10.1145/1188455.1188568.

Table 1. Comparing Tiled-Display Middleware Systems.

	SAGE	Chromium	DMX*	Equalizer	CGLX*
Supports multiple applications simultaneously	Yes	No	Yes	No	No
Supports non-OpenGL applications	Yes	No	Yes	No	No
Application transparency	No	Yes	Yes	No	No
Supports distance collaboration	Yes	No	No	No	No
Platforms	Linux, Mac, Windows	Linux, Mac, Windows	Linux	Linux, Mac	Linux, Windows
Open source	Yes	Yes	Yes	Yes	No

** DMX stands for Distributed Multihead X Project; CGLX stands for Cross-Platform Cluster Graphic Library*

in addition to ParaView. All these visualization packages are based on the Visualization Tool Kit (VTK), a cross-platform and open source software system for 3D computer graphics, image processing, and visualization.[5] We'll develop a VTK output module for SAGE so that researchers can use potentially any VTK-based visualization tool with SAGE. We expect these solutions will help scientists and engineers use high-resolution display walls and CI for their ultrascale visualizations.

Another solution is to integrate commonly used parallel-rendering systems for tiled displays, such as Chromium or Equalizer, with SAGE (see the "SAGE versus the Competition" sidebar). These tools would feed SAGE rather than drive a tiled display directly. This integration will let users run several instances of these tools at one time on a display and benefit from the expertise invested in the systems' development. Furthermore, integration with Chromium or Equalizer will let users of OpenGL-based visualization applications use parallel rendering without modifying their applications. They'll also be able to harness SAGE's ability to show results from these multiple applications side-by-side.

Developing a Persistent Visualcasting Service

The international Visualcasting demonstrations confirmed that Visualcasting is a requirement for display-rich global collaboration. However, users still lack the capability to easily share, discover, browse, and initiate collaboration sessions. In addition, Visualcasting resources must be able to automatically scale as the number of streams and resolution grow. We plan to leverage the accumulated knowledge from the Access Grid project, which supports distributed collaborative interactions over computational grids, to develop a persistent, more robust Visualcasting service.[6] We'll initially deploy it at the StarLight international communications exchange facility in Chicago. Subsequently, we'll work with OptIPlanet partners to create more Visualcasting service points internationally.

Another problem in building a persistent Visualcasting service is supporting the heterogeneous end points in network and display capacity. Because of this heterogeneity, end points consume (display) streamed data at different rates, and a slow end point might degrade overall Visualcasting performance. For example, the maximum frame rate of an uncompressed 4K animation at an end point with 6 Gbps of network bandwidth is 30 fps; the maximum frame rate at another end point with 1 Gbps of network bandwidth is only 5 fps. We'll address this problem by developing multilayered Visualcasting. This idea is analogous to layered multicasting—that is, distributing a version of the same image data with different image quality or resolution through each data layer.[7] Each end point subscribes to a layer appropriate to its data bandwidth and display resolution without affecting the streams to other end points.

Enhanced Support for Interactivity

Although SAGE applications run on remote servers and stream visualizations to tiled displays, SAGE doesn't forward an application's GUI to the displays. This is important because a GUI often carries information about the visualization. Without access to the application GUI, users at remote sites can only view the application output, rather than interact with it. Leveraging Virtual Network Computing (VNC), we'll develop a service that lets users view and control an interface directly on the tiled display as well as through a wireless laptop.

To fully exploit increased display resolution, SAGE's current API lets application developers pass interaction gestures as events to their applications. However, SAGE still needs a more complete set of interaction capabilities, including a widget library to create visual interfaces for tiled displays. These widgets will be smart enough to resize themselves appropriately according to the tiled display's size and resolution. We've built an early prototype of a widget framework to validate the concept. A comprehensive set of widgets will require additional work. The next step would be extending these widgets so that collaborating users can control them from different tiled displays.

Using SAGE and OptIPortals, scientists can create visualization pipelines from multiple sources to access and share information, in various resolutions and formats.

Today, ultra-high-resolution display environments are fast becoming the lenses of virtual microscopes and telescopes, enabling researchers to observe data in CI repositories. Using SAGE and OptIPortals, scientists can create visualization pipelines from multiple sources to access and share information, in various resolutions and formats. This capability lets users access up-to-date and related information sets to display them simultaneously, quickly see context as well as detail, and make more informed and timely decisions. This technology will have a profound, transformative effect on ultrascale collaborative visualization, making CI more accessible to end systems and users, in both the laboratory and classroom.

The SAGE software, including full source code, user manuals, and applications, is available at www.sagecommons.org.

Acknowledgments

We appreciate the time and effort OptIPlanet Collaboratory partners contributed to SAGE's development and enhancement. We also thank Paul A. Navratil, Greg Abram, and Kelly P. Gaither for editing our manuscript. US National Science Foundation awards CNS-0420477, OCI-0441094, OCI-0225642, and OCI-0943559 partly supported our research.

References

1. L.L. Smarr et al., "The OptIPuter," *Comm. ACM*, vol. 46, no. 11, 2003, pp. 58–67.
2. B. Jeong et al., "High-Performance Dynamic Graphics Streaming for Scalable Adaptive Graphics Environment," *Proc. 2006 ACM/IEEE Conf. Supercomputing* (SC 06), IEEE CS Press, 2006, article 108, doi:10.1145/1188455.1188568.
3. B. Jeong, "Visualcasting: Scalable Real-Time Image Distribution in Ultra-high Resolution Display Environments," doctoral dissertation, Dept. of Computer Science, Univ. Illinois at Chicago, 2009.
4. A. Cedilnik et al., "Remote Large Data Visualization in the ParaView Framework," *Proc. Eurographics Workshop Parallel Graphics and Visualization* (EGPGV 06), Eurographics, 2006, pp. 162–170.
5. W. Schroeder, K. Martin, and B. Lorensen, *The Visualization Toolkit: An Object-Oriented Approach to 3D Graphics*, 4th ed., Kitware, 2006.
6. L. Childers et al., "Access Grid: Immersive Group-to-Group Collaborative Visualization," *Proc. 4th Int'l Immersive Projection Technology Workshop* (IPT 00), 2000; http://academic.research.microsoft.com/Paper/181952.aspx.
7. S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-Driven Layered Multicast," *Proc. ACM Sigcomm*, ACM Press, 1996, pp. 117–130.

Byungil Jeong *is a research associate with the Texas Advanced Computing Center at the University of Texas at Austin and a primary Scalable Adaptive Graphics Environment architect. His research interests include scalable parallel graphics architecture, collaborative remote visualization, large-scale data visualization, and high-resolution display systems. Jeong has a PhD in computer science from the University of Illinois at Chicago. Contact him at bijeong@tacc.utexas.edu.*

Jason Leigh *is an associate professor of computer science and the director of the Electronic Visualization Laboratory at the University of Illinois at Chicago. His research interests include tele-immersion, high-performance transport protocols, and collaboration and visualization technologies for application support, such as remote large-scale-data exploration, education, and entertainment. Leigh has a PhD in computer science from the University of Illinois at Chicago. Contact him at spiff@uic.edu.*

Andrew Johnson *is an associate professor of computer science and a member of the Electronic Visualization Laboratory at the University of Illinois at Chicago. His research and teaching interests include interaction and collaboration using advanced visualization displays and applying those displays to enhance discovery and learning. Johnson has a PhD in computer science from Wayne State University. Contact him at ajohnson@uic.edu.*

Luc Renambot *is a research assistant professor at the Electronic Visualization Laboratory at the University of Illinois at Chicago. His research interests include high-resolution displays, scientific visualization, and high-speed networking. Renambot has a PhD in computer science from the University of Rennes. Contact him at renambot@uic.edu.*

Maxine D. Brown *is an associate director of the Electronic Visualization Laboratory at the University of Illinois at Chicago. Her research interests include computer graphics, scientific visualization, collaboration, human-computer interfaces, petascale computing, and optical networking infrastructure. Brown has a master's degree in computer science from University of Pennsylvania. Contact her at maxine@uic.edu.*

Ratko Jagodic *is a PhD candidate in the Department of Computer Science at the University of Illinois at Chicago and a research assistant at the university's Electronic Visualization Laboratory. His research interests include human-computer interaction and computer-supported cooperative work in large, high-resolution display environments. Jagodic has a bachelor's degree in computer science and chemistry from Lake Forest College. Contact him at rjagodic@uic.edu.*

Sungwon Nam *is a PhD student at the Electronic Visualization Laboratory at the University of Illinois at Chicago. His research interests include high-resolution displays, high-performance computing, and high-speed networks. Nam has a master's degree in computer science from University of Southern California. Contact him at snam5@uic.edu.*

Hyejung Hur *is a PhD student in the Department of Computer Science at the University of Illinois at Chicago. Her research interests include human-computer interaction, visualization, and collaboration. Hur has a master's degree in computer science from Duksung Women's University. Contact her at hhur2@uic.edu.*