


Building and Using A Scalable Display Wall System



Kai Li, Han Chen, Yuqun Chen, Douglas W. Clark, Perry Cook, Stefanos Damianakis, Georg Essl, Adam Finkelstein, Thomas Funkhouser, Timothy Housel, Allison Klein, Zhiyan Liu, Emil Praun, Rudrajit Samanta, Ben Shedd, Jaswinder Pal Singh, George Tzanetakis, and Jiannan Zheng
Princeton University

Princeton's Scalable Display Wall project explores building and using a large-format display with multiple commodity components. The prototype system has been operational since March 1998. Our goal is to construct a collaborative space that fully exploits a large-format display system with immersive sound and natural user interfaces.

Unlike most display wall systems today, which use high-end graphics machines and high-end projectors, our prototype system is built with low-cost commodity components: a cluster of PCs, PC graphics accelerators, consumer video and sound equipment, and portable presentation projectors. This approach has the advantages of low cost and of tracking technology well, as high-volume commodity components typically have better price-performance ratios and improve at faster rates than special-purpose hardware. The challenge is to use commodity components to construct a high-quality collaborative environment that delivers display, rendering, input, and sound performance competitive with, or better than, that delivered by the custom-designed, high-end graphics machine approach.

A schematic representation of our current display wall system appears in Figure 1 (next page). It comprises an 8 × 18-foot rear-projection screen with a 4 × 2 array of Proxima LCD polysilicon projectors, each driven by a 450-Mhz Pentium II PC with an off-the-shelf graphics accelerator. The resolution of the resulting image is 4,096 × 1,536 pixels. We integrated the display system with several components, including

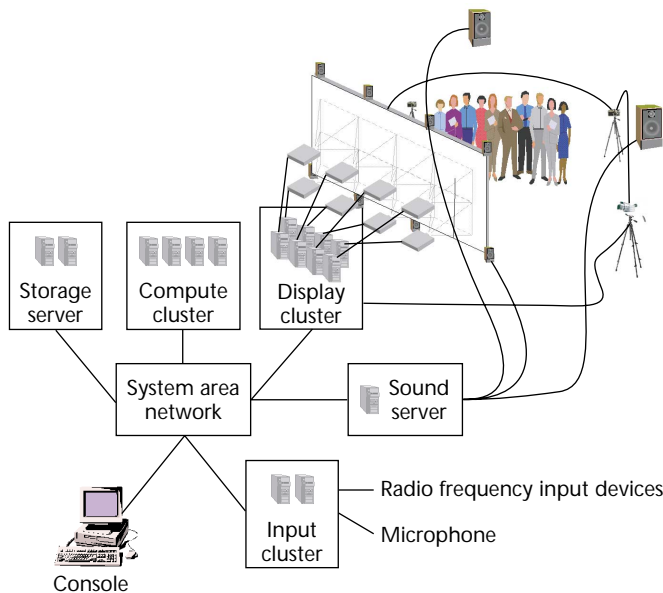
- a *sound server* PC that uses two 8-channel sound cards to drive 16 speakers placed around the area in front of the wall;
- an *input cluster* that uses two 300-MHz Pentium II PCs to capture video images from an array of video cameras, to gather input from a gyroscope mouse, and to receive audio input from a microphone;
- a *storage server* that uses two PCs, each with 5 inexpensive EIDE disks;
- a *local compute cluster* of 4 PCs that provides high-bandwidth access to compute cycles;
- a *remote compute cluster* containing 32 PCs; and
- a *console PC* that controls execution of the system.

All the PCs are connected together with a 100 Base-T Ethernet network. In addition, a Myrinet system area network connects the PCs of the display cluster, local compute cluster, and storage server. We use the Virtual Memory-Mapped Communication (VMMC) mechanism developed in the Scalable High-Performance Really Inexpensive Multiprocessor (Shrimp) project.¹ VMMC implements a protected, reliable, user-level communication protocol. Its end-to-end latency at the user level is about 13 microseconds, and its peak user-level bandwidth is about 100 Mbytes per second on the Myrinet.^{2,3}

Our research focuses on usability and scalability. In order to address usability, we must investigate new user interfaces, new content design methodologies, and learn from human perception studies in teaching design courses. In order to achieve scalability, we must carefully address three key system design issues:

- *Coordination among multiple components.* Commodity components are usually designed for individual use rather than as building blocks for a larger, seamless system. To achieve seamless imaging and sound requires developing methods to coordinate multiple components effectively.
- *Communication performance and requirements.* Immersive and collaborative applications require that multiple components communicate effectively. A scalable system should provide a low-latency, high-bandwidth mechanism to deliver high-performance communication among multiple commodity components. At the same time, software systems and applications must be carefully designed to achieve high quality and performance while minimizing communication requirements.
- *Resource allocation.* Effective resource allocation and partitioning of work among components is critical at both the system and application levels.

In this article we report our early experiences in building and using a display wall system. In particular, we describe our approach to research challenges in several specific research areas, including seamless tiling, parallel rendering, parallel data visualization, parallel MPEG decoding, layered multiresolution video input,



1 A scalable, inexpensive display wall system.

multichannel immersive sound, user interfaces, application tools, and content creation.

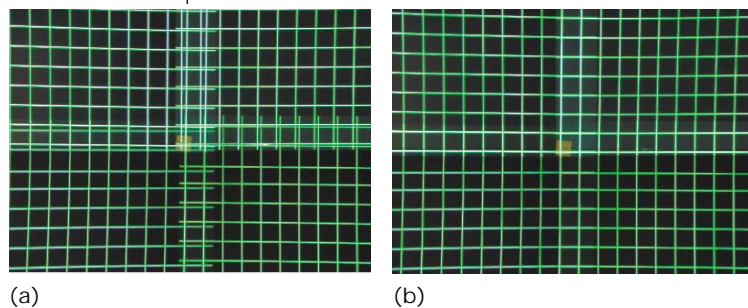
Seamless tiling

Despite much recent progress in the development of new display technologies such as Organic Light Emitting Diodes (OLEDs), the current economical approach to making a large-format, high-resolution display uses an array of projectors. In this case, an important issue is the coordination of multiple commodity projectors to achieve seamless edge blending and precise alignment.

Image blending

Seamless edge blending can remove the visible discontinuities between adjacent projectors. Edge blending techniques overlap the edges of projected, tiled images and blend the overlapped pixels to smooth the luminance and chromaticity transition from one image to another. The current state-of-the-art technique employs specially designed hardware to modulate the video signals corresponding to the overlapped region.⁴ This electrical edge-blending approach works only with CRT projectors; it doesn't work well with commodity LCD or DLP (digital light processing) projectors because these new projectors leak light when projected pixels are black, making them appear dark gray. Overlapped dark gray regions then look lighter gray—brighter than nonoverlapped regions. To avoid seams, we reduce the light projected in the overlapped regions.

2 Photographs of a projected grid pattern on the display wall (a) without correction and (b) with correction.



Our approach relies on the technique of *aperture modulation*—placing an opaque object in front of a lens (between the projector lens and the screen) to reduce the luminance of the image without distorting the image itself. Thus, by carefully placing an opaque rectangular frame, we can make its shadow penumbra coincide with the interprojector overlap regions.⁵

Computational alignment

To make a multiprojector display look seamless, projected images must align precisely in all directions. Aligning projectors manually takes time. The traditional alignment method uses a sophisticated adjustable platform to fine-tune projector position and orientation. This approach requires expensive mechanical devices and extensive human time. In addition, it doesn't work for commodity projectors whose lenses tend to produce image distortions.

To overcome both misalignment and image-distortion problems, we use image-processing techniques to correct the source image before display by misaligned projectors. In other words, we prewarp the image in such a way that the projected images align. We call this approach *computational alignment*. It requires only the coarsest physical alignment of the projectors.

Our alignment algorithm currently calculates for each projector a 3×3 projection matrix, with which an image-warping process resamples the images to counter the effects of physical misalignment. Figure 2a shows a picture without correction. Figure 2b shows the picture after each projector resamples the image according to its correct perspective matrix. As a work in progress, we adapt our alignment algorithm to correct some distortions caused by imperfect lenses, such as radial distortions.

We obtain precise alignment (or misalignment) information with an off-the-shelf camera that has much lower resolution than our display wall. We zoom the camera to focus on a relatively small region of the display wall and pan the camera across the wall to get a broader coverage. The camera measures point correspondences and line matches between neighboring projectors. We then use simulated annealing to minimize alignment error globally and solve for the projection matrices.

Our approach differs from the solutions of Rasker et al.,¹⁴ which use carefully calibrated, fixed-zoomed camera(s) to obtain projector distortion measurements. The cameras in their approach must see the entire screen or a significant portion of it. Therefore, they cannot easily obtain subpixel alignment information.

Parallel rendering

We are investigating parallel rendering algorithms⁷ for real-time display of very large, high-resolution images partitioned over multiple projectors. Here we face all three general types of research challenges: coordination of PCs and graphics accelerators to create consistent, real-time images; communication among multiple PCs and their graphics accelerators; and resource allocation to achieve good utilization.

Our efforts focus on developing “sort-first” and “sort-

last” parallel rendering methods that minimize communication requirements and balance the rendering load across a cluster of PCs.⁸ Our general approach partitions each frame into a number of virtual tiles. It then assigns a set of virtual tiles to each rendering machine to balance the load as evenly as possible. Since the virtual tiles usually don’t correspond to the physical tiles on the wall, rendered pixels must often be read back from the rendering PC’s frame buffer and transferred over the network to the projecting PC’s frame buffer. We use the VMMC mechanism to achieve low latency and high-bandwidth communication for the pixel redistribution phase, as well as to provide fast synchronization of the frame-buffer swapping.

The research issues are to develop algorithms that compute the shapes and arrangement of virtual tiles dynamically, sort graphics primitives among virtual tiles in real time, deliver graphics primitives to multiple PCs in parallel, and redistribute pixels across a network efficiently. To explore this space we designed and implemented several sort-first virtual tiling algorithms. The best of these algorithms uses a KD-tree partition of the screen space followed by an optimization step to ensure the best possible balance of the load.⁹

Figures 3 and 4 show the cases with a static screen-space partition without load balancing and a KD-tree partition after load balancing, respectively. The colors indicate which machines render the different parts of the scene. Looking at the load bars on the bottom right of the figure reveals the imbalance in the first case. The load is much better balanced in the KD-tree case. As a result the final frame-time is up to four times lower with eight PCs.

Parallel data visualization

Increases in computing power have enabled researchers in areas ranging from astrophysics to zoology to amass vast data sets from both observation and simulation. Since the data itself is quite rich, the display wall presents an ideal medium for scientific visualization at high resolution. The magnitude of the data sets motivates the use of parallel computation, a fast network, and separation of computation and rendering across different machines.

Initially, our research focused on developing parallel algorithms that permit users to interactively view isosurfaces in volumetric data on the display wall. Our system uses the PCs in the display cluster to perform rendering, the PCs in the compute cluster to perform isosurface extraction, and storage servers to hold data sets. We coordinate these three sets of PCs in a pipelined fashion on a per-frame basis. Data are sent from the storage servers to the isosurface-extraction PC cluster. Triangles for an isosurface are generated in parallel using a marching cubes algorithm¹⁰ accelerated with an interval method¹¹ based on Chazelle’s filtering search. The



3 Parallel rendering without load balancing. Note the load bars in the lower right.



4 Parallel rendering with load balancing. Note the load bars in the lower right.

triangles then go to the appropriate rendering PCs.

We have experimented with lossless compression methods to reduce communication requirements. Even with compression, we find that low-latency, high-bandwidth communication between the isosurface extraction PCs and rendering PCs is critical.

Figure 5 shows the result of using our parallel visualization system to visualize part of the Visible Woman data set from the Visible Human Project at the National Library of Medicine (<http://www.nlm.nih.gov/research/visible/>). We’re currently focusing on better isosurface extraction algorithms, large-scale storage server development, and load-balancing methods to improve the utilization of computing resources.

Parallel MPEG-2 decoding

MPEG-2 is the current standard format for delivering high-quality video streams for entertainment, collaboration, and digital art. Our goal is to develop fast, pure-software, MPEG-2 decoding methods on a PC cluster to bring HDTV or even higher resolution MPEG-2 videos to a scalable display wall.

To achieve the desired 60-frames-per-second (fps) real-time frame rate—including the overhead in scaling and loading pixels into the frame buffer—a decoder should decompress one frame in less than about 14 ms. We approach the problem in two steps: (1) developing a fast decoder on a single PC and (2) designing a fast, par-

5 Parallel visualization of the Visible Woman data set.





6 Multilayered video registration program.

allel, scalable decoder for a PC cluster. The key research challenges here are coordination among PCs to split an MPEG-2 stream and fast communication among PCs to decode high-resolution streams in real time.

To improve the MPEG-2 decoding performance on a single PC, we exploited both instruction-level parallelism and memory/cache behavior. We developed our decoder based on the open source MPEG Software Simulation Group reference design, which decodes 720p HDTV (1280 × 720) at about 13 fps on a 733-MHz Pentium III PC. We use Intel MMX/SSE instructions extensively to accelerate arithmetic operations, and carefully design the data structures and their layouts to improve the data cache utilization. Our preliminary result is a decoder capable of decompressing 720p HDTV stream at more than 56 fps on a 733-MHz Pentium III PC. The speed-up exceeds a factor of four.

To further improve the performance, we use parallel decoding on a PC cluster. Previous work on parallel MPEG decoding almost exclusively used shared-memory multiprocessors,¹² parallelizing MPEG-2 video decoding at either the picture or slice level. However, the amount of data movement among the PCs is too high if we use these methods for a PC cluster. Thus, we developed a novel macroblock-level parallelization. We use a single PC to split an MPEG-2 stream into multiple substreams at macroblock level and send them to the display cluster PCs for decoding, scaling and display.

With the previous picture-level or slice-level parallelization, the per-link bandwidth requirement of the decoding PC depends on the whole video size. With our macro-block-level parallelization, it depends only on

the size of the portion that the local node decodes. This makes our approach highly scalable. Our preliminary result shows that with 4 PCs (in a 2 × 2 setup) decoding 720p HDTV streams in parallel, the aggregate communication bandwidth requirement among all nodes is only about 100 megabits per second (Mbps). As a comparison, this number can reach 1.7 Gbps when using a picture or slice level parallelization.

Multilayered video input

Video resolution has always been limited by the TV standards. To take advantage of the high resolution of a scalable display wall, we're working on methods to create video streams at a scalable resolution that matches the display resolution, using a small number of commodity video cameras. The main research challenge is the coordination among video cameras.

The traditional approach uses juxtaposed cameras with edge overlapping and stitches multiple images together.¹³ It has several disadvantages. First, juxtaposed cameras make zooming awkward—the cameras must be synchronized, and the angles between them must be adjusted mechanically at the same time. Second, since each camera has its own focal point, scenes with many depths can look unnatural with multiple focal points. Third, it requires many video cameras. For example, it requires 28 640 × 480 video cameras for a 4,096 × 1,536-resolution display wall. The aggregate communication requirement of the video streams is also too high for the network. We would like to overcome all of these problems.

Our approach, called layered multiresolution video, uses a number of cameras to cover different fields of view. Each camera can be panned, tilted, and zoomed individually. We're developing a fast registration algorithm to find the correspondence of the different layers and merge them into one. This method not only solves the three problems mentioned, but also fits nicely into the MPEG-4 video compression framework.

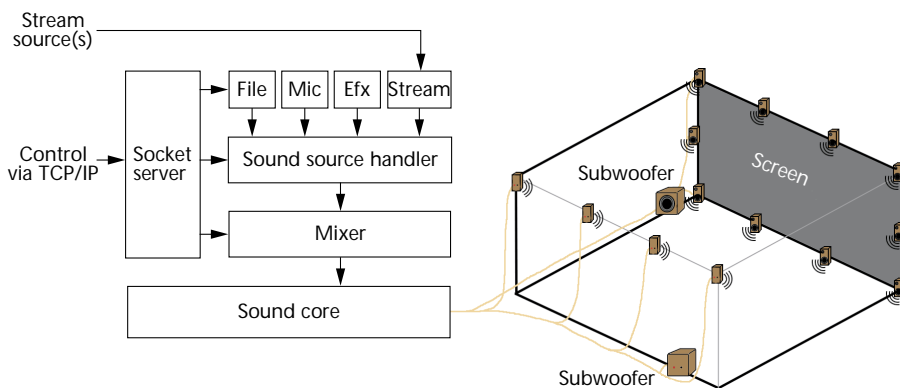
Our current registration algorithm runs at 30 registration passes per second for 2 images. Figure 6 demonstrates the registration process. Our goal is to develop a registration algorithm that runs at real time.

Multichannel immersive sound

Sound guides the eyes, enhances the sense of reality, and provides extra channels of communication. Since the visual display spreads over a large surface, large amounts of the displayed data might lie outside any user's visual field. Sound can draw directional auditory attention to an event, causing users of a large display system to turn their heads toward the sound and thus bring important visual information into their field of view.

To investigate the integration of immersive sound with a large-scale display wall, we speakers positioned around the space in front of the display wall to provide immersive

7 Multichannel sound system.



sound synthesis and processing in real time. The key challenge is the coordination of multiple sound devices to create immersive sound.

We implemented the display wall sound system on commodity PCs, using inexpensive multichannel sound cards. These cards, designed for digital home-recording use, can be synchronized through SPDIF/AESEBU cables and special calls to the software drivers. We've written a sound server that takes commands from any computer via a transmission-control protocol/Internet protocol (TCP/IP) connection. The server can play back sound files through any combination of the 16 speakers in the present configuration (see Figure 7). Other possible sound sources include onboard synthesis of sound effects, microphone signals, sound streams from any machine on the network or Web, and effects (reverb, echo, and so forth) processing of any sound source.

User interfaces

A large collaborative space presents interesting challenges for user interfaces, both display and control. Because of the display wall's scale, it's important to track human positions, recognize gestures, and construct imagery and sound appropriate for the user's position. Many methods developed in the past require users to carry cumbersome tracking or sensing devices. We have focused on developing natural methods for users to interact with the system. We use multiple cameras in the viewing field to track human and input devices. We also developed image-processing algorithms to understand gestures in a collaborative environment. The main research challenge is the coordination among commodity input devices and with the computers in the display wall PC cluster.

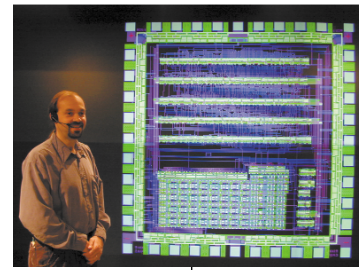
We wrote a multi-input mouse server program that runs on a master cursor-control computer. Any other computer can take control of the display wall mouse by running a mouse client program and connecting to the server. This has allowed us to quickly construct and test a number of new pointing devices, including a swivel chair (the Quake Chair), voice-input mouse control, and pressure-sensitive floor panels. Figure 8a shows a camera-tracked wand used as a pointer device, and Figure 8b, a wireless microphone used as a speech recognition device. Research challenges include allowing multiple cursors at once, as well as further refinement and integration of camera tracking.

Methods to design application tools

It's important and nontrivial to bring many applications to a scalable display wall and run them at the intrinsic resolution supported by the display surface. Most video-wall products use special-purpose hardware to scale relatively lower-resolution content, such as NTSC, VGA, SVGA, and HDTV formats to fit large display surfaces. Only a few expensive solutions use high-end graphics machines to render directly in the intrinsic resolution of a multiprojector display system. Coordination and communication are the two main challenges in developing tools to port off-the-shelf applications to a scalable display wall using its native display resolution.



(a)



(b)

8 Input methods for a display wall include (a) Magic Wand input and (b) voice recognition.

We have investigated four methods to design tools for applications: custom-designed, distributed application, distributed 2D primitive, and distributed 3D primitive. We illustrate each method by an example.

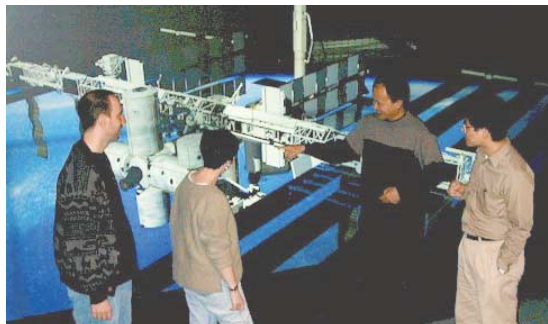
Custom-designed method

Our first tool on the display wall, a Still Image Viewer, lets a naive user display still images and perform cross fading between images on the wall. The image viewer contains two parts: a controller program and an image-viewer program. An image viewer program runs on every PC in the display cluster. The controller program runs on a different PC and sends commands over the network, such as loading an image from the disk, displaying a cached image, or cross fading between two cached images. The image viewer loads JPEG images from a shared file system, decodes only its portion of the image, and synchronizes with other viewers on other PCs prior to swapping the frame buffer.

The controller program also implements a scripting interface so that users can write scripts to control image and video playback synchronized with our multichannel sound playback. Many students have made multimedia presentations on our display wall using the image viewer and the multichannel sound system. Figure 9 shows an image of the International Space Station on the display wall.

Distributed application method

We distribute application-level input commands to bring the Building Walkthrough program, designed for a uniprocessor system, to the display wall. We run an instance of the program on every PC in the display cluster. To coordinate among these programs, we run another instance on the console PC. A user drives the walkthrough using the console PC. The console translates the user inputs and sends the camera information and screen

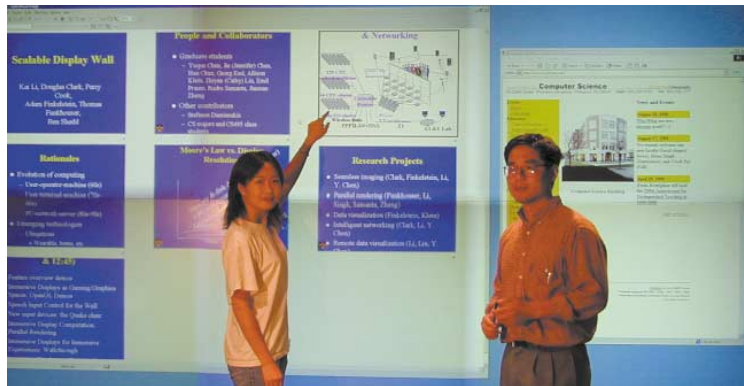


9 Looking at an image of the NASA Space Station with the Still Image Viewer.



10 The distributed Building Walkthrough program used with the display wall.

11 Windows 2000 VDD running Microsoft PowerPoint and Internet Explorer on the display wall.



space information to each PC, which renders a different part of the screen from its own copy of the scene database. This method provides interactive frame rates (about 20 fps) without noticeable synchronization problems. Figure 10 shows the walkthrough program running with a 3D model created by Lucent Technologies.

Distributed 2D primitive method

We developed a Virtual Display Driver (VDD) to bring existing Windows applications to the display wall, using a distributed 2D primitive method. VDD fakes a high-resolution graphics adapter to the Windows 2000 operating system. It leverages the feature in Windows 2000 that supports multiple monitors on a single PC. VDD intercepts all device driver interface (DDI) calls and executes them remotely as remote procedure calls (RPCs) on the PCs in the display cluster.

Users can drag application windows from the regular CRT display into our virtual display, the contents of which are subsequently drawn on the display wall. All drawing done by the application on VDD is performed in the intrinsic resolution of the virtual display—the same resolution as the display wall. Therefore, users can see a lot more details in any Windows applications than with existing commercial video walls.

Figure 11 shows Microsoft PowerPoint and Internet Explorer running on our display wall through VDD. At close range, where people stand directly in front of the display wall, both applications show adequate details and no fuzziness with line drawings and text.

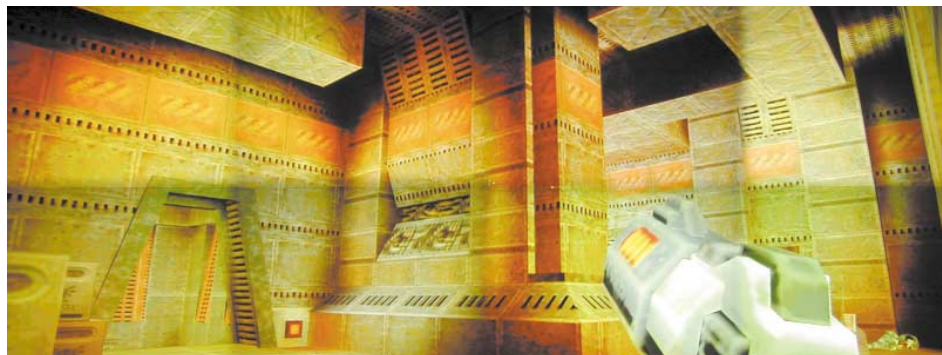
Distributed 3D primitive method

We developed a user-level, distributed OpenGL tool using a 3D primitive distribution method. Unlike the distributed 2D primitive method, where our tool works at the device driver level, the distributed OpenGL library lives at the user level. We take advantage of the fact that on all Win-

dows platforms, an application's calls to the OpenGL API are made through a dynamically linked library (DLL), `opengl32.dll`. We implemented our own `opengl32.dll` to intercept all the OpenGL calls and forward them to the display cluster PCs. These PCs receive the RPC calls and execute them, with the exception that the view frustums are properly modified so that each projector renders only its own tile portion of the screen space.

This distributed OpenGL mechanism lets many off-the-shelf Windows OpenGL applications run on the display wall without modification. We have successfully used this mechanism with many such applications including games, CAD, and visualization tools. Figure 12 shows the game *GIQuake* running on the display wall using our distributed OpenGL mechanism. Currently, we're investigating methods for integrating our parallel rendering algorithms into this OpenGL library.

12 *GIQuake* running on the display wall using the distributed OpenGL mechanism.



Content creation and design

We started studying content creation and design methods at the same time as other research topics. We have taught two design courses using the display wall, mainly to provide experience using desktop-size screens to create effective wall-size images. Figures 13 to 15 show students' creations on the display wall.

Compared to traditional, expensive display walls, the inexpensive aspect of the scalable display wall makes a big difference in content creation. Suddenly, we have a new design space available to all users, particularly nontechnical users. This rapid democratization of billboard-size display space is quite provocative. We ask students in the design class to imagine future applications and implications when many such walls are widely used, and to investigate the best uses for these large displays.

One implication of a wall-size image is that it completely fills our visual field, which creates a one-to-one experience with the onscreen imagery. There is no border or frame for scale reference as on small monitors. This single shift creates a whole new design paradigm.¹⁴ Designers must add areas of interest and focus to the image composition.

A second implication is that a group can interact with information on just a portion of the screen while others focus on a different area. Different viewers at different distances from the high-resolution screen can move around in the room space while viewing.

Third, objects can be seen life-size or intensely magnified. For example, an image of a dense computer chip reads like a road map.

Fourth, there isn't necessarily a need to change the images rapidly, as they can be so densely filled with data that it takes a while to absorb it all. Often, a single high-resolution screen can be displayed for 10 to 20 minutes and remain continuously interesting.

Fifth, the light from the screen can become the room light for the working group.

All of these elements, especially the frameless nature of the image, require new thinking and new ways of approaching design.^{15,16}

This new design paradigm motivates future work in composition tools for large-format displays. Self-expression has a new form. TCL scripting adds the dimension of time to wall presentations, providing capabilities for timed displays and dissolves from image to image. By synchronizing music and sounds to changing images, the wall has become a storytelling space for presentations of 5 to 10 minutes, as complex and engrossing as any short film or video.

The wall room, with its billboard-sized images, has served three times as a performance art and theater space. Virtually everyone who visits the display wall expresses some kind of emotional response about being in the huge visual and aural space.¹⁷

Early experiences

We have used the Princeton Scalable Display Wall prototype system as an infrastructure to conduct our research as well as to teach two design courses. The approach of using a multiplicity of commodity parts to construct a scalable display-wall system works well, but it requires us to address design tradeoffs to deal with coordination, communication, and resource allocation issues. We have successfully addressed these tradeoffs and developed solutions in several research areas as outlined in this article. In seamless rendering, we have developed a combination of optical edge-blending and software image manipulation for projector alignment. In parallel rendering, we have developed a "sort-first" screen partitioning method that achieves good load balance and parallel speedup. In parallel data visualization, we have developed a parallel isosurface extraction algorithm for a PC cluster architecture. In parallel



13 Multiple small windows on the display wall.



14 Sketches on a digital canvas.



15 A fractal image.

MPEG-2 decoding, we have developed a fast splitter and a fast decoder that achieve real-time decoding entirely in software with minimal communication overhead. In layered multiresolution video, we interactively combine multiple video streams with a fast registration algorithm. And in application tools design, we developed four methods to let existing applications use the native resolution of the display system while minimizing communication requirements.

Study of user interface issues and human perceptions is very important in building a collaborative environment with a scalable display wall system. We have developed and experimented with several user interfaces beyond the traditional keyboard and mouse, including a gyroscope mouse, a “magic wand” implemented by multi-camera tracking, and a speech-recognition user interface. Our experience shows that natural, unencumbered user interfaces based on passive sensors are useful in such an environment and that it’s very desirable to let multiple users control a shared display wall simultaneously.

Finally, in teaching design courses using our display wall system, we have found that the resolution and scale of the display require new ways of approaching design. For instance, vast amounts of information can be presented in a single image, rather than as a sequence of images as in a desktop display. Typographic layouts where the font sizes can range from 2 to 600 points bring new capabilities to the use and meaning of text. Sound, especially spatial sound integrated with imagery, is critical for storytelling. An emerging design aesthetic for large scale, high-resolution images depends on the center of the images rather than on the frame of the wall. Perhaps, from the high magnifications seen in wall-size imagery, we will discover new insights and experiences not previously available. ■

Acknowledgments

The Princeton Display Wall Project is supported in part by the Department of Energy under grant ANI-9906704 and grant DE-FC02-99ER25387, by an Intel Research Council and Intel Technology 2000 equipment grant, and by the National Science Foundation under grant CDA-9624099 and grant EIA-9975011. An NSF Career award and an Alfred P. Sloan Fellowship also support the research programs of Adam Finkelstein and Thomas Funkhouser, respectively. Jaswinder Pal Singh is also supported by an NSF PECASE award and an Alfred P. Sloan Fellowship. We are also grateful to the Arial Foundation, Interval Research, Microsoft, and Viewsonic for their generous donations.

We would like to thank John DiLoreto, for building special large-format screens, and several colleagues from Intel Corporation—Konrad Lai, Dick Hofsheier, Steve Hunt, Paul Pierce, and Wen-Hann Wang—for sharing their ideas, projector-mount design, and contents. We also would like to thank all the students who took the design classes and who conducted independent studies using the display wall for their content creation.

References

1. M. Blumrich et al., “Virtual Memory Mapped Network Interface for the Shrimp Multicomputer,” *ACM/IEEE Proc. of the 21st Ann. Int’l Symp. on Computer Architecture*, IEEE Computer Society Press, Los Alamitos, Calif., Apr. 1994, pp. 142-153.
2. C. Dubnicki et al., “Design and Implementation of Virtual Memory-Mapped Communication on Myrinet,” *Proc. IEEE 11th Int’l Parallel Processing Symp.*, IEEE Computer Society Press, Los Alamitos, Calif., Apr. 1997, pp. 388-396.
3. Y. Chen et al., “UTLB: A Mechanism for Translations on Network Interface,” *Proc. ACM Architectural Support for Programming Languages and Operating Systems (ASPLOS-VIII)*, ACM Press, New York, Oct. 1998, pp. 193-204.
4. T. Mayer, “New Options and Considerations for Creating Enhanced Viewing Experiences,” *Computer Graphics*, Vol. 31, No. 2, May 1997, pp. 32-34.
5. K. Li and Y. Chen, “Optical Blending for Multi-Projector Display Wall System,” *Proc. 12th Lasers and Electro-Optics Society 1999 Ann. Meeting*, IEEE Laser and Electro-Optics Society, IEEE Press, Piscataway, N.J., Nov. 1999.
6. R. Raskar et al., “Multi-Projector Displays Using Camera-Based Registration,” *Proc. IEEE Visualization 99*, IEEE Computer Society Press, Los Alamitos, Calif., Oct. 1999, pp. 161-168.
7. T.W. Crockett, “An Introduction to Parallel Rendering,” *Parallel Computing*, Vol. 23, No. 7, 1997, pp. 819-843.
8. S. Molnar et al., “A Sorting Classification of Parallel Rendering,” *IEEE Computer Graphics and Applications*, Vol. 14, No. 4, July 1994, pp. 23-32.
9. R. Samanta et al., “Load Balancing for Multi-Projector Rendering Systems,” *Proc. Siggraph/Eurographics Workshop on Graphics Hardware*, ACM Press, New York, Aug. 1999, pp. 107-116.
10. W. Lorensen and H. Cline, “Marching Cubes: a High-Resolution 3D Surface Construction Algorithm,” *Computer Graphics (Proc. Siggraph 87)*, Vol. 21, No. 4, 1987, pp. 163-170.
11. P. Cignoni et al., “Speeding Up Isosurface Extraction using Interval Trees,” *IEEE Trans. on Visualization and Computer Graphics*, Vol. 3, No. 2, June 1997, pp. 158-170.
12. A. Bilas, J. Fritts, and J. P. Singh, “Real-Time Parallel MPEG-2 Decoding in Software,” *Proc. Int’l Parallel Processing Symp.*, IEEE Computer Society Press, Los Alamitos, Calif., 1997, pp. 197-203.
13. R. Szeliski and H.-Y. Shum, “Creating Full View Panoramic Image Mosaics and Environment Maps,” *Proc. Siggraph 975*, ACM Ann. Conf. Series, ACM Press, New York, 1997, pp. 251-258.
14. B. Shedd, “Exploding the Frame: Seeking a New Cinematic Language,” *Proc. SMPTE 135th Conf.*, Society of Motion Picture and Television Engineers, White Plains, N.Y., 1994.
15. R. Arnheim, *The Power of the Center*, University of California Press, Berkeley and Los Angeles, Calif., 1988.
16. E.R. Tufte, *Visual Explanations*, Graphics Press, Cheshire, Conn., 1997.
17. M. Lombard and T. Ditton, “At the Heart of It All: The Concept of Presence,” <http://www.ascusc.org/jcmc/vol3/issue2/lombard.html>.

Kai Li is a professor in the Department of Computer Science at Princeton University. He received his PhD from Yale University in 1986. He is a member of IEEE Computer Society and a Fellow of the ACM.

Han Chen is currently a PhD student in Department of Computer Science at Princeton University. He received his BS in computer science from Tsinghua University, China, in 1997.

Yuqun Chen is completing his PhD degree in computer science at Princeton University. He has been the lead student for the Princeton Scalable Display Wall project. He obtained a BS in computer science with Highest Distinction from the University of Maine.

Douglas Clark received a BS from Yale University in 1972 and a PhD from Carnegie Mellon University in 1976. He has been a professor of computer science at Princeton University since 1993.

Perry Cook studied music at the University of Missouri, Kansas City Conservatory of Music, and electrical engineering at UMKC and Stanford University. He joined the faculty of Princeton, Computer Science and Music, in 1996.

Stefanos Damianakis is a research staff member at Princeton University, where he completed his PhD in 1998.

Georg Essl received his engineer's degree from Graz University of Technology in 1996. He is currently a PhD candidate at Princeton University.

Adam Finkelstein joined the faculty of Department of Computer Science at Princeton University in 1997. He received his PhD degree from the University of Washington in 1996 and BS degree from Swarthmore College in 1987. He received a Sloan fellowship in 1999.

Thomas Funkhouser is an assistant professor in the Department of Computer Science at Princeton University. He received a BS in biological sciences from Stanford University in 1983, an MS in computer science from UCLA in 1989, and a PhD in computer science from UC Berkeley in 1993. He received a Sloan Fellowship in 1999.

Timothy C. House is cinematographer for feature films and giant-screen IMAX films, and visual consultant for the Scalable Display Wall project.

Allison Klein received her BA degree in Japanese Studies from Stanford University in 1992. She is currently a PhD student in computer science at Princeton University.

Zhiyan Liu is a PhD student in the Department of Computer Science, Princeton University. She received her BS degree in computer science from Tsinghua University, China, in 1998.



Left to right: Stefanos Damianakis, Zhiyan Liu, George Tzanetakis, Jaswinder Pal Singh, Allison Klein, Douglas Clark, Rudrajit Samanta, Emil Praun, Kai Li, Georg Essl, Perry Cook, Yuqun Chen, Ben Shedd, Han Chen, Thomas Funkhouser, Adam Finkelstein.

Emil Praun is a graduate student in Department of Computer Science at Princeton University. He received his BS from the California Institute of Technology in 1997 and an MA from Princeton in 1998.

Rudrajit Samanta is currently a PhD graduate student in the Department of Computer Science, Princeton University. He received his BS degree in physics and mathematics from Bates College in 1995.

Ben Shedd is a Senior Visiting Research Scholar and Lecturer in the Department of Computer Science at Princeton University. He received the 1978 Academy Award for Best Documentary Short Subject for the film *The Flight of The Gossamer Condor* and directs IMAX films. He received his MA in 1973 from the University of Southern California's School of Cinema/Television.

Jaswinder Pal Singh is an associate professor in the Department of Computer Science at Princeton University. He obtained his PhD from Stanford University in 1993 and his BSE from Princeton in 1987. He received a Presidential Early Career Award for Scientists and Engineers and a Sloan Research Fellowship.

George Tzanetakis is currently a PhD student in the Department of Computer Science at Princeton University. He received his BS from the University of Crete, Greece.

Jiannan Zheng is currently a design engineer at EMC Co. He received an MSE from Princeton University and an MSE degree in computer science from Tsinghua University, China, in 1997.

Contact the authors at Dept. of Computer Science, Princeton University, Princeton, NJ 08544. Contact Li by e-mail at li@cs.princeton.edu or on the Web at <http://www.cs.princeton.edu/omnimedia/>.