Building a Large Scale, High Resolution, Tiled, Rear Projected, Passive Stereo Display System Based on Commodity Components

Glenn Bresnahan*, Raymond Gasser, Augustinas Abaravichyus,
Erik Brisson, and Michael Walterman

Boston University
Scientific Computing and Visualization Group
Office of Information Technology
111 Cummington Street
Boston, MA 02215

ABSTRACT

The Boston University Deep Vision Display Wall is a large scale, high-resolution, tiled, rear-projected, passive stereo display system based on commodity components. Using Linux on PC workstations provides an affordable infrastructure for distributed processing and high-end graphics. Passive stereo eliminates the need to genlock the display video cards as well as allowing the use of very inexpensive glasses and inexpensive projectors. By careful selection of projectors, polarizing filters, and screen material, problems such as cross-talk, chromatic aberration, and low luminance are minimized. The 8'x15' display surface at Boston University is installed as one wall of the viewing room. The final installation will use 24 workstations driving 24 projectors to produce a 4096x2304 stereo image. This paper discusses development issues including synchronization of displays, alignment of projectors, blending of overlapping projected images, and constraints on projectors and projection surface to support passive stereo. Performance comparisons are given for configurations including workstations driving one vs. two projectors, Ethernet vs. Myrinet for intermachine communication, and overall display performance. Also discussed are software issues involved in putting the system together and with providing an application environment for our user community.

Keywords: stereo display, tiled display, passive stereo, high-resolution, commodity components, distributed graphics.

1. INTRODUCTION

The goal of the authors' work was to build a large, high-resolution stereo display at a reasonable cost. The authors are members of a group which serves a large, diverse research community involved in a range of computational science, art, and other projects. Because the group collaborates with a number of research groups within the University and with other institutions, it was desired that the design be one which could be communicated to and duplicated easily by others. Because the system would be used for a variety of purposes, it was also important that it be general and easy to use.

The main concept of a tiled wall is a set of computers, the display machines, connected to a set of projectors arranged so that the set of projected images form a single large image. The screen space is partitioned (virtually) into a rectangular grid of tiles, each of which is produced by one projector. Each display machine may drive one or more projectors, depending on its graphics capabilities and various performance trade-offs. Tiled walls based on commodity components have been investigated by a number of researchers [2, 6, 8, 10, 14], and single-tile stereo displays based on commodity components have been developed [9].

* glenn@bu.edu

To build a stereo tiled wall involves producing two tilings of the type just described, one for each eye. There are two feasible methods for delivering the different images to the viewer's eyes. The first uses active shutter glasses synchronized with the display refresh, which, by alternately blocking each eye, allows successive images to be passed to each eye in turn. The second, referred to as passive stereo, displays two separate images at all times and uses polarizing filters and glasses to pass the desired images to the two eyes. To achieve the same effective frame rate relative to a standard display, the first method requires doubling the frame rate, and the second method requires doubling the number of graphics pipes and projectors.

The software infrastructure to drive the Wall, along with an application, comprise a distributed graphics system. The display machines need graphics drivers, low-level graphics software, and possibly higher-level software for tracking graphics state. Software for processing, coordinating, and communicating graphics data between the application, remote visualization, and display machines is needed. If synchronization of the display machines is desired, then this must be integrated into the system. Synchronization ensures that the geometry, viewpoint, and other graphics properties are consistent across all of the tiles over time. Part of the development of the Wall has included building a software infrastructure capable of supporting the diverse set of projects we collaborate on.

We started construction of the first prototype in the summer of 2001, and displayed it at the annual Supercomputing Conference in Denver in November, 2001. It consisted of four stereo tiles (arranged as 2x2 grid), using four computers and eight projectors, giving a wall resolution of 2048 x 1536 pixels per eye. Our current (second) prototype consists of six stereo tiles (a 3x2 grid), using twelve computers and twelve projectors, giving 3072 x 1536 pixels per eye. The final installation will consist of twelve stereo tiles (a 4x3 grid), using twenty-four computers and twenty-four projectors, giving 4096 x 2304 pixels per eye.

2. TILING ISSUES

The ultimate goal of a tiled wall is to produce a high resolution image which shows no artifacts from the tiling. The projection screen can be divided into a regular grid of rectangles, each of which represents the ideal tile (projection area) for one projector. The main artifacts which occur come from three sources: deviations of the projected tile shape from its ideal grid rectangle, giving rise to the need for tile alignment and edge blending; differences in color characteristics of the projected tiles, requiring color matching; and brightness differences across tile boundaries caused by dispersion effects. Each of these artifacts makes the boundaries between the tiles apparent and thus breaks the illusion of a seamless image.

2.1 Tile Alignment

As the screen is divided into a rectangular grid, ideally each projector would be positioned in space so that its projected tile would exactly match its associated screen grid rectangle. If the projectors are not aligned in this way, the images will be misaligned (see Figure 1). There are two sets of issues in aligning the projected tiles. The first set includes properties of the tiles relative to one another, in particular having tile edges meet cleanly and having straight lines remain straight across tile boundaries. The second set has to do with the tiles meeting physical "ground truth" criteria, including the top and bottom edges being truly horizontal (level), the side edges being truly vertical (plumb), and the dimensions of the tiles matching the ideal grid cells.

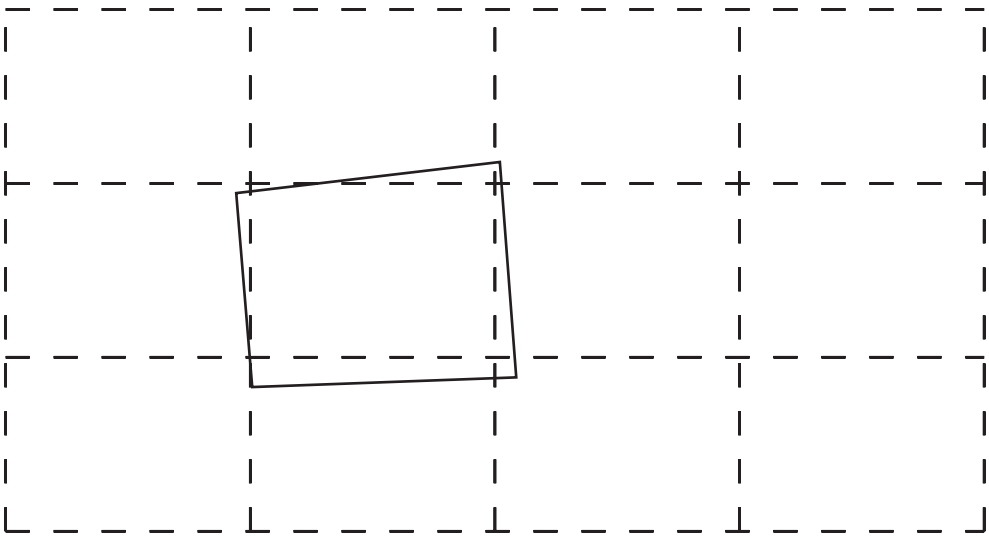

Figure 1: One misaligned tile (solid lines) shown with ideal grid (dashed lines)

One solution to this problem is to perform a transformation on the image which is to be projected so that the resulting tile on the screen matches the screen grid. This warping transformation may be represented as a 4x4 matrix using standard homogeneous coordinates. This may be thought of as an additional transformation at the end of the graphics pipeline [8, 10, 12]. In the case of OpenGL the last transformation in rendering geometry is application of the 4x4 projection matrix. If these two matrices are multiplied, giving the composition of the warping and projection, the resulting matrix may be substituted for the original projection matrix, effectively incorporating the warp into the existing rendering process at no additional computational expense. Note that this method requires modifying the rendering source code. Alternatively, on some platforms it may be possible to warp the image without modifying the source code by intervening at a low level in the graphics subsystem [14]. This method involves operations on the frame buffer contents after the application has finished rendering, which hurts performance. The warping may also be done in the projector itself, avoiding this last issue - such projectors are appearing on the market at the time of this writing.

As the alignment process is a very difficult task to perform manually, there have been methods developed to automate it. One method uses a video camera placed in front of the display screen. Each of the display machines projects a known pattern. A control program identifies the patterns in the video camera image, and uses this to compute warping transformations for each of the tiles. This can be done iteratively. This method of camera-controlled software alignment has been developed by [8, 10, 12, 14].

Another solution to the problem is to build mechanical positioners, one per projector, which may be adjusted so as to bring the projector into proper alignment. This requires adjusting six degrees of freedom (three translations and three rotations). We had eight simple positioners, based on those described in [2, 3], fabricated for our first prototype, which used manual adjustment of screw mechanisms. Designing and building such mounts requires a significant amount of mechanical expertise, is expensive, and performing the alignment by manual adjustments is difficult. By incorporating actuators into the construction of the positioners it is possible to make the adjustments via computer control. Using a camera-guided program approach as described for software alignment, the process could be similarly automated. We are actively investigating the construction of such automated positioners for our final installation.

Consider these methods in the context of a stereo tiled wall. Now there are two tile grids, one for each eye. Suppose that each projector needs to be placed so that its optical axis is perpendicular to the screen and hits the center of the corresponding screen grid cell. Then for a given tile the projectors for the left and right eye images would have to be placed at the same location, making the mechanical positioning method impossible. Various projectors have different projection geometries, each of which will impose different positioning constraints. Some projectors allow an optically-

corrected off-axis projection. We chose one such projector which includes "lens-shifting", which allows horizontal and vertical translation of the thrown image, optically correcting for the shift. It is possible to place the left and right eye projectors so that lens shifting overlaps on the desired screen rectangle. We use lens shifting to get the two projected tiles roughly aligned, including overlap for edge blending, then fine adjustments may be made using physical positioners or software warping.

We developed a system for doing software tile alignment interactively. Each display machine projects a regular grid which is transformed by a warping transform. Initially this transform is the identity, i.e., there is no warping. Superimposed on this grid are small gray squares indicating tile corners and one larger white square indicating an "active" tile corner (see Figure 2). The operator views the Wall and moves the active corner horizontally and vertically by pressing the appropriate arrow keys. As the active corner is moved, the warping transform for the tile containing it is updated so as to move the active corner while keeping the other three corners fixed. Thus the appearance of the grid for this tile is warped interactively. Different levels of movement are allowed, allowing sub-pixel adjustment. The operator changes the active corner by moving the white square along adjacent tile corners. The warping transforms computed during this procedure are maintained as 4x4 matrices. They are stored in a common place where they can be found by all applications which need them.

While it would be preferable to have the entire alignment process completely automated using video cameras, the method we use is much quicker than manual adjustment of mechanical positioners, and is not an unreasonable burden on the person doing it. It is cheaper and simpler than using the camera-based method. The procedure is roughly as follows. Each display machine sends a uniform grid to its projector or projectors. The first step is to make the projected tiles the proper size and aspect ratio by using the projector zoom feature and measuring the projected size on the screen. In the second step, we use a laser leveler which has both horizontal and vertical projected lines, aligned with pre-marked points at the edge of the screen, to locate the ideal grid corners on the screen. Lens shifting is used to bring the tile corners close to the grid corners. Then our software tool is used to move the tile corners so that they coincide with these points. The final step is to use the software tool to make fine adjustments to all of the corners to make the projected grid lines look as uniform as possible across the entire scene.

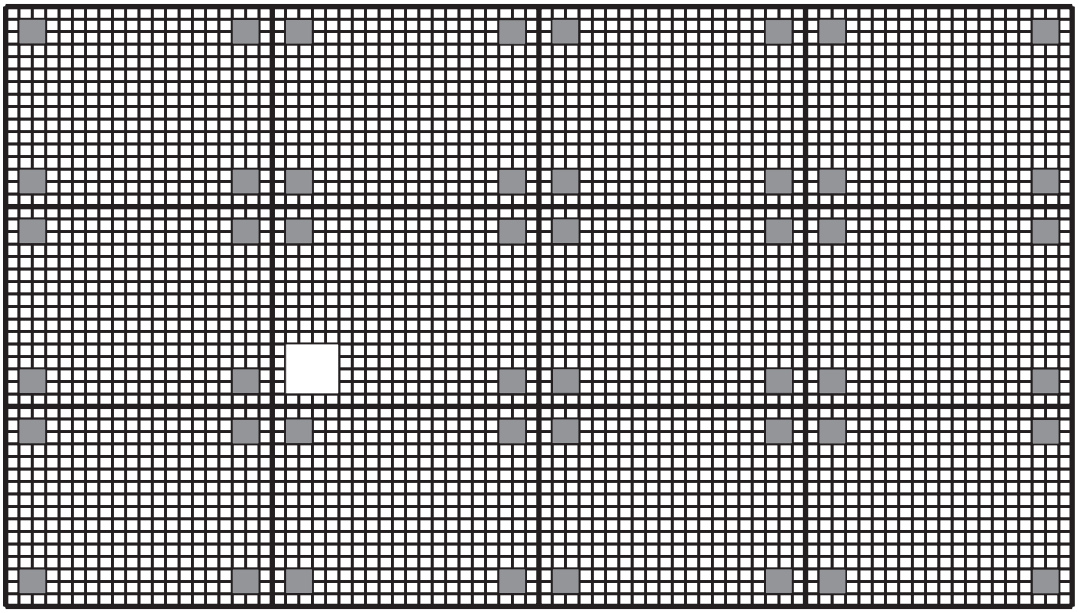

Figure 2: Software alignment Wall display

One of the criteria we evaluated in selecting projectors was the ability to control the projectors programmatically. For instance, when designing physical positioners we had hoped to use fine control of lens shifting to control two degrees of freedom of the adjustment. The projectors we bought allowed connections via serial cables or by TCP/IP over Ethernet.

However, after we implemented this type of control we discovered that the control given by this method was too crude to be useful. For example, the electronic control of lens shifting was unpredictable - when attempting to move the minimum allowed step, the motion might be anywhere in the range of 2 to 6 pixels.

Each of the two solutions described have advantages and disadvantages. Physical positioners work for any program running on the display machines since no code modification is necessary, but are complicated and expensive. The software solution is inexpensive but generally requires code modification.

2.2 Edge Blending

Ideally the projected tiles would match perfectly along their common edges, providing a seamless image. In practice the edges never line up exactly: if two tiles are separated, then there will be a black line between them; if they overlap, there will be a bright line if the background is not black. A very small error in this is quite visually noticeable.

A solution used by many builders of tiled walls is edge blending. Adjacent tiles are aligned so their edges overlap by a small amount. The edges of the two tile images are made to fade to black as they near the edge, using a fading function which ensures that the combined intensity in the overlapping region equals that of the intended image [8, 11, 14]. A small mismatch of tile edges will then result in only a small variation across the overlapping region. The method we implemented was based on [15, 16] – essentially a Gaussian falloff is applied to a fixed-width border around the edge of the pre-warped image. This was incorporated into Chromium (discussed below) using alpha blending of a precomputed fading map with the rendered image.

Edge blending also may be implemented using physical barriers at the edge of the projected frustum, to "soften" the edges optically [2, 3, 8].

2.3 Color Matching

Due to variations in projection bulb characteristics, as well as differences between the projectors per se, the appearance on the screen of a colored pixel in the frame buffer may vary considerably in different projected tiles. Both the intensity and color of light produced by bulbs change considerably with age. The difference in color makes the tiles look noticeably different, which breaks the illusion of a seamless image. The color needs to be modified across the entire tile. Unlike alignment and edge blending, the authors know of no way to do color matching mechanically. Many projectors allow manual adjustment of the color using a handheld controller, and some allow electronic control. Our projectors allow adjusting the brightness and contrast for each color using both types of control.

An alternative to using projector controls is to modify the color written into the frame buffer. There are several methods for doing this. The first uses a color lookup table. If the response function of the projector is measured for each of the three colors, then the inverse of these response functions may be used to load the color lookup table. When the color lookup is applied, the corrected colors will be written into the frame buffer. (Note that the color lookup table may already be used for gamma correction, so that the gamma correction may have to be worked into the lookup table values for the projector's color correction.) A second method uses a color matrix [13]. If we consider the set of (R,G,B) colors as a 3-dimensional space, then the 4x4 color matrix operates on this space just as the usual geometric 4x4 transformation matrix operates on 3-D geometric spaces.

2.4 Dispersion at Tile Boundaries

One problem with using rear projection is that the screens do not disperse the light uniformly with angle. This means that at a tile boundary, in the general case where the viewer is at a location such that the scatter angle at one boundary is different than the scatter angle of the other, there will be a noticeable brightness differential which makes the tile boundaries noticeable. To address this, we tried to pick a screen with the best dispersion available which satisfied our polarization constraints. Given available space behind the screen, another improvement may be obtained by placing the projectors at the greatest possible distance from the screen, so as to minimize the difference in incidence angles of light hitting the screen at the common boundaries of adjacent projected tiles. Note that if head tracking was used for one viewer, and the dispersion function was known, then the variations due to viewing angle could be compensated for by image operations on the image to be displayed.

3. MATERIALS AND PHYSICAL ISSUES

Building and installing the Wall involved choosing projectors, a display screen, and a method of delivering stereo images to the viewer's two eyes. It also involved building supports for the projectors, support for the screen, access to projectors and machines, and addressing user comfort and perception issues in the viewing area.

In determining a method for providing stereo, we decided early on that the only feasible choices were active shutter systems, such as StereoGraphics CrystalEyes, and passive systems using linear filters and circular filters. We decided that if possible we would use passive stereo rather than active shutter glasses. People are generally more comfortable with the lightweight passive glasses than with active shutter glasses. Passive glasses are easier to maintain since there are no batteries to replace and no parts to break, and they are so cheap that breakage and loss are not major concerns. We also wanted to avoid having to genlock all of the display machines, as that would add complexity and there were few if any genlock-capable graphics boards available for Linux machines at the time. A further reason for choosing passive stereo is that it did not require projectors with an extra high refresh rate. We believed that with the passive stereo images available at all times (rather than temporally interlaced) the human visual system would be able to integrate the non-genlocked (out-of-phase) images.

In choosing projectors, polarizing materials, and a screen, we were trying to optimize the visual trade-offs between brightness, dispersion, and cross-talk (light intended for one eye reaching the other eye). We wanted the image to be as bright as possible, with the cross-talk acceptably low, and the brightness dropoff with angle to be as low as possible. Considerations involved when choosing projectors included brightness, contrast, resolution, controls, cost, and polarization effects. Choices involved in choosing a screen involved transmission (gain) and dispersion. Considerations when choosing polarization materials involved transmission, cross-talk, and chromatic effects. We tried numerous projectors in combination with several screen materials and both linear and circular polarizing filters. We purchased Stewart Filmscreen 150 material for the projection screen, NEC GT950 LCD Projectors, and linear polarizing filters and glasses from Reel 3D Enterprises. (Since the time of our purchase an apparently superior screen material, Stewart Filmscreen 200, has come to market.)

Most of the unanswered questions had to do with polarization, or more specifically with the interaction of the polarization between LCD projectors, polarizing filters between the projector and the screen, rear-projection screen material, and polarizing filters in the available passive glasses. We obtained samples of materials and loans of projectors, and tested all the combinations of these elements, to produce a subjective, optimum trade-off.

Many people had expressed doubt that a stereo wall could be built based on LCD projectors because of polarization problems. The set of elements which we put together avoid the potential problems in some interesting ways. The projectors we chose use three internal LCD matrices (as do most LCD projectors), one for each color: red, green, and blue. White light is emitted by the bulb, which passes through a polarizing filter and then through a sequence of three dichroic mirrors. The first passes red light to the first LCD matrix, the second passes green light to the second LCD matrix, and the third passes blue light to the third LCD matrix. Essentially, the polarized light passing through the LCD assembly is modulated by the polarizing properties of the liquid crystals. When the light leaves the projector, the red and blue light have the same polarization, while the green light is polarized at a 90 degree offset from the red and blue. The standard linear polarizing filter glasses have a lens for each eye, whose orientations are offset from each other by 90 degrees. It would seem that the difference in polarization angle between the red and blue light and the green light from the projector would make it impossible to use the glasses. However, the polarization angles in the projectors we chose are offset by 45 degrees from that of the glasses. So when light leaves the projector and passes through the polarizing filter, all three colors are attenuated equally since they are all 45 degrees offset from the filter (in one of the two possible directions). Thus the transmission of the red, green, and blue through the glasses will be the same. There are more expensive LCD projectors which project all three colors with the same polarization, oriented in such a way as to work with passive stereo glasses. This eliminates the need for a filter between the projector and the screen, thus increasing brightness.

Given what seemed a good combination of projector and screen material, we compared linear to circular polarization. The advantage of using circular polarization would be that it does not change its transmission characteristics based on rotation angle, i.e., the viewer does not get increased cross-talk when tilting his or her head. However, circular polarization does decrease the transmission considerably. There were also chromatic effects, in particular there were noticeable color shifts when using circular polarization, and furthermore these color aberrations changed with rotation angle. This is consistent with the different polarization angles associated with the colors produced by the projector, as described in the previous paragraph. Given that linear filters are cheaper, have better transmission, and have no noticeable color problems, we decided to go with that method. Head tilting has not seemed to be a problem with the Wall. Note that, with either linear or circular polarization, to properly handle head tilt in a stereo environment requires adjustment of the eye positions in the rendering process, which requires head tracking of some sort.

There are many optical measurements which could be made in characterizing our Wall, but we have only attempted to look at those which are relevant to our task of building a stereo wall. These are the transmission and cross-talk characteristics of the polarizing materials. We tested all combinations of projector polarization (no filter, filter for left eye projector, filter for right eye projector) with the glasses' filters (none, glasses' left eye filter, glasses' right eye filter). A colorimeter was used to test the luminance of solid red, green, and blue projected images, at the location where the viewer's eye would be, i.e., close to the glasses' filter. These tests were all done with the given components in their normal positions and orientations. Additionally we tested one of each of the glasses' filters in the orientation which gave the minimum and maximum luminances. These occur when the glasses are rotated 45 degrees clockwise or counterclockwise, which is consistent with the polarization of the light leaving the projectors (discussed earlier in this section).

First filter	Glasses' filter	Red	Green	Blue
None	None	4.67	25.50	2.10
None	Left eye (min)	0.48	2.69	0.23
None	Left eye	1.32	11.48	0.76
None	Right eye	1.43	9.96	0.73
None	Right eye (max)	2.73	14.80	1.33
Right proj	None	1.48	10.24	0.70
Right proj	Left eye	0.16	0.93	0.06
Right proj	Right eye	0.94	6.14	0.40
Left proj	None	1.84	8.90	0.80
Left proj	Left eye	0.98	5.56	0.41
Left proj	Right eye	0.11	0.92	0.12

Table 1: Projector-to-eye transmission with various polarization configurations (fL)

Table 1 gives the results of these tests in footlamberts.

	Red	Green	Blue
Left eye	0.21	0.45	0.36
Right eye	0.31	0.39	0.35

Table 2: Polarization transmission ratio

Table 2 gives the transmission ratio of the system with polarizing filters, to the system without filters, for each eye.

	Red	Green	Blue
Left eye	0.16	0.17	0.15
Right eye	0.12	0.15	0.30

Table 3: Polarization cross-talk

Table 3 gives the cross-talk for each eye, computed as the ratio of the luminance reaching the eye from the opposite-eye projector to the luminance reaching the eye from the intended projector.

Other issues arose concerning the physical housing for the system. As part of a larger space reorganization we modified our Graphics Lab to incorporate the Wall. Components of the Wall system occupy three rooms: the viewing area is part of the Lab, there is a room behind the screen devoted to housing the projectors, and beyond this is a machine room which holds the Wall Linux cluster and the compute Linux cluster. As length is an issue for video cables, a set of holes was made in the wall between the projector room and the machine room to allow cables from the Wall cluster machines to reach the projectors. The 8' x 15' screen occupies one wall of the viewing area within the Lab. Light-blocking blinds were installed in the viewing area windows. A light-tight curtain was installed in the projection room, and lighting is kept off in the projection room so as to not interfere with the projected images. Lighting and air conditioning are controlled independently for the three rooms. The air conditioning in the projection room needs to be kept at a lower temperature than in the viewing area because the projectors generate so much heat. The air conditioning ducts in the viewing area and the projector room were placed in the ceiling to be away from the screen, to avoid affecting the flexible screen. The separate air conditioning zones still give rise to pressure differentials between the two rooms which can cause the flexible screen to bulge one way or the other, so it is necessary to provide an opening between the two spaces to allow pressure equalization. The light output of each NEC GT 950 projectors is 2000 lumens. Even with the attenuation caused by passing through the filters, screen, and glasses, the tiled image is very bright. The recessed ceiling lights in the viewing area, controlled by dimmer switches, may be turned up enough to allow viewers to easily see each other and interact without interfering with Wall viewing.

4. GRAPHICS INFRASTRUCTURE AND APPLICATIONS

In its final configuration the Wall will be run by 24 IBM Intellistations, i.e., 24 dedicated display machines. The Linux cluster and the Wall machines are all interconnected via Myrinet. There is a dedicated "head node" Intellistation for the user to log in to for access to the Wall machines. Our computing facilities include a 192-processor Origin 2000 system, a 64-processor IBM SP system, a 112-processor IBM p690 system, and a 104-processor IBM x330 Linux Cluster. This Linux cluster is connected to the Wall machines via Myrinet. This equipment is connected to other large national installations by very high speed networks. It is expected that computing and graphics tasks running on these large machines and the larger grid will be communicating with the Wall to produce a variety of displays.

In our environment, user input may come in the form of typing on the head node keyboard, using the head node mouse, using a 6-degree-of-freedom input device, or via multiple video cameras watching the users. Compute cycles may come from any or all of the machines listed above. Visualization/filtering may also be performed on any of these machines. Rendering is typically performed on the display machines to take advantage of those machines' graphics hardware as well as immediate access to their frame buffers and video outputs. In some cases it may make sense to perform rendering on the larger machines; as an example it might make sense to do radiosity calculations on one of the SPs or p690s and then send all the portions of the final image to the display machines. One of our development goals is to efficiently distribute the elements of the computing/visualization pipeline, in this complex distributed computing environment. We are still at a very early stage in this development effort.

In addition to the goal of providing a large, high-resolution display at a reasonable cost, tiled walls offer the potential for increased graphics performance by utilizing distributed rendering. Distributed visualization may be broken down into sub-tasks, including user input (control parameters, navigation, etc.), data input and/or collection, computation, visualization/filtering, and rendering. The whole system may run on one machine, or the sub-tasks may run on different machines, and any of the sub-tasks may be distributed over multiple machines, depending on optimal use of resources, network latency and throughput, and caching/data management. There are a number of ways of categorizing the

programming paradigms used to drive a tiled wall. One characterization is by the type of information that is passed from the application machines to the display machines. We have used these three paradigms for communicating graphics information to the display machines: pixel streams, graphics commands/primitives, and higher level scene description commands. There are many details and much research into distributed graphics, for instance sort-first, sort-last, and related hybrid methods, which we will not delve into in this paper. For discussions of distributed rendering in the context of tiled walls, see [1, 4, 6].

Using pixel streams, essentially the image to be displayed is partitioned into sub-images of the right size for the display machines. In the case of remote rendering, for instance, doing radiosity calculations on a high-performance computing machine, the sub-images would be sent from the computation machines to the display machines. If the images have been precomputed, then they may be played back as a movie. If pre-partitioned, they may be streamed off local disk or streamed from remote machines. If not pre-partitioned, they may be partitioned and distributed on the fly. This method requires a large amount of data, demanding a lot of local disk space or very high bandwidth. We will not go into this method here.

In the graphics command/primitives paradigm each display machine receives a stream of commands which it can send to its graphics engine to render into the frame buffer. For many applications this will require less data to be sent across the network to the display machines than pixel streams would. If a standard graphics protocol such as OpenGL is used, this can provide a very general interface. WireGL [4, 5] and its successor, Chromium [7], do exactly this, serving as wrappers around existing OpenGL programs. We have used WireGL and Chromium in several of our applications. Given an executable built to use a dynamic OpenGL library, if this executable uses the Chromium library at run time, then all of its OpenGL calls will be caught and interpreted by the Chromium library. The commands are distributed across a network to all of the Chromium servers running on Wall display machines, which will then execute the OpenGL commands. The fact that Chromium is open source has allowed us to incorporate our alignment and edge blending techniques into our Chromium libraries and servers.

Using the higher-level scene description paradigm basically attempts to minimize communication by caching as much state as possible, for instance by keeping geometric models stored on the local machines and only communicating transformation changes. This is well-suited for scene graph based simulations. We use this paradigm when running DAFFIE applications on the Wall. DAFFIE (Distributed Applications Framework for Immersive Environments) is a software package and network architecture, developed by our group, for easily building distributed/collaborative tele-immersive environments. Complete environments may be created from pre-existing data using only simple scripts or arbitrarily complex worlds may be created by using network-based agents which are capable of fully manipulating any virtual object. DAFFIE provides participants representations via avatars, multiple virtual spaces, animated models, 3D localized sound using multi-channel audio, network-based telephony and streaming video. DAFFIE is designed to run on high-speed, low-latency networks, such as Internet2. In addition to the features included in the DAFFIE system, we support rendering synchronization on the Wall through a set of distributed barrier calls based on the DAFFIE library. The graphics component of DAFFIE is based on the SGI Performer library.

In addition to writing OpenGL applications and running them using Chromium, and using DAFFIE, we have developed applications using the Visualization ToolKit (VTK). VTK is an open source software package for 3D computer graphics, image processing, and visualization. We have found it to be a useful way to build scientific visualization programs for our user community, as it is well-supported and provides much of the functionality needed for common tasks. The fact that it is open source was crucial, as we needed to make modifications to build an application whose display could be mapped to the Wall using WireGL, and later, Chromium. As supplied, VTK has support for several forms of stereo, but none which generate two images in a single window, which was required for use with WireGL. We modified the VTK source to provide side-by-side stereo in a single window. This could then be mapped to the tiles on the wall using the functionality provided by WireGL and Chromium.

5. PERFORMANCE

In this section we will give performance results. The first set of tests measure one OpenGL application under different system configurations. The baseline is running this application fullscreen at 1024x768 resolution on one workstation. All workstations and Wall display machines are IBM 1.7 GHz Pentium 4 IntelliStations with 1GB RAM, and ATI FireGL 4 graphics running RedHat Linux 7.2. The performance of the Wall was measured in its second prototype configuration of six stereo tiles arranged in a 2x3 grid, which uses 12 projectors, giving a wall resolution of 3072 x 1536. The set of tests is the rendering of a large number of cubes arranged in a regular three-dimensional grid. The entire configuration of cubes is rotated about the vertical axis. Each cube is composed of one tristrip. The first test used 1500 cubes (18000 triangles), the second used 3000 cubes (36000 triangles), and the last used 4500 cubes (54000 triangles). The parameters which were varied were the following: either 6 workstations driving 2 projectors each, or 12 workstations driving 1 projector each; using 100Mb Ethernet, IP over Myrinet, or native Myrinet for communication; and running the display workstations synchronized or non-synchronized.. The software we used for these tests include WireGL 1.2.1 (built with Intel C++ 6.0 compiler), Chromium pre-1.0 CVS version (built with Intel C++ 6.0), VTK 3.2 (built with gcc 2.96), Performer 2.5.0, RedHat 7.2 Linux, kernel 2.4.9-31, and Myrinet drivers 1.5.1. The refresh rate is 75 Hz. All measurements reported are in frames per second (fps). Numbers in parentheses are the number of calls to the draw routine, which may be higher than the refresh rate.

	# triangles = 18000	# triangles = 36000	# triangles = 54000
Workstation	75.0 (160.1)	75.0 (82.6)	56.0
Wall w/100 Mb Ethernet	6.2	3.2	2.3
Wall w/IP over Myrinet	40.9	20.2	13.6
Wall w/native Myrinet	51.4	24.8	16.7

Table 4: Effect of network type

Table 4 gives the results of running the straightforward OpenGL application running on a workstation and on the Wall using WireGL, synchronized, as the distribution mechanism. The overhead of distributing data to the display machines causes a significant performance loss compared to a single workstation. The improvement in performance of Myrinet over 100Mb Ethernet is clear.

	# triangles = 18000	# triangles = 36000	# triangles = 54000
100Mb Ethernet no sync	7.7	3.6	2.4
100Mb Ethernet w/sync	6.2	3.2	2.3
IP over Myrinet no sync	38.3	18.5	12.4
IP over Myrinet w/sync	40.9	20.2	13.6
Native Myrinet no sync	54.2	25.5	17.0
Native Myrinet w/sync	51.4	24.8	16.7

Table 5: Cost of synchronization

Table 5 gives the results of running the application on the Wall using WireGL using IP over Myrinet, comparing the performance of running with and without synchronization. It would be expected that running without synchronization would always give better performance than running with synchronization, due to overhead in the latter case. This was borne out in the case of 100Mb Ethernet and native Myrinet cases. In these cases the performance loss due to the synchronization was small. There seems to be an anomaly in the IP over Myrinet case, where the expected trend is reversed. At the time of writing we have not identified the cause of this anomaly.

	# triangles = 18000	# triangles = 36000	# triangles = 54000
No alignment or blending	18.9	9.4	7.1
With alignment, blending	17.7	8.7	7.0

Table 6: Cost of tile alignment and edge blending

Table 6 gives a comparison of running with and without tile alignment and edge blending. Because of the way that tile alignment is implemented in software, the overhead is negligible. The small performance loss due to edge blending can be attributed to the use of the alpha blending. We are using Chromium for this comparison rather than WireGL as we have not implemented edge blending in WireGL.

	# triangles = 18000	# triangles = 36000	#triangles = 54000
OpenGL / WireGL 12x1	40.9	20.2	13.6
OpenGL / WireGL 6x2	44.1	21.9	14.6
OpenGL / Chromium 12x1	17.7	8.7	7.0
OpenGL / Chromium 6x2	17.3	8.8	7.0
VTK / WireGL 12x1	5.5	2.8	1.8
VTK / WireGL 6x2	6.3	3.0	1.8
VTK / Chromium 12x1	2.3	1.1	0.8
VTK / Chromium 6x2	4.2	2.1	1.5

Table 7: Driving 1 vs. 2 projectors

Table 7 shows the results of using 12 workstations each driving 1 projector vs. 6 workstations each driving 2 projectors. Overall the results are comparable, although there are many cases where using fewer workstations is a performance advantage. The authors believe that this indicates cases where the performance bottleneck is in the communications rather than in rendering.

	# triangles = 18000	# triangles = 36000	#triangles = 54000
OpenGL / WireGL	40.9	20.2	13.6
OpenGL/ Chromium	17.7	8.7	7.0
VTK / WireGL	1.8	2.8	5.5
VTK / Chromium	0.8	1.1	2.3
DAFFIE	75.0 (223.6)	75.0 (122.2)	75.0 (84.3)

Table 8: Comparison of implementations

The second set of measurements compares the performance of doing a particular task using the applications built on the three programming environments which we have been using: raw OpenGL using WireGL, raw OpenGL using Chromium, VTK using WireGL, VTK using Chromium, and DAFFIE. Each of these applications performs the same cube rendering task described in the previous test, so as to be as similar as possible. The VTK program explicitly uses tristrips. The DAFFIE program, based on Performer, reads in an Inventor file which implements the cubes as tristrips.

Table 8 shows the results of the measurements of the different implementations. The overhead of using VTK is surprisingly high. The poor performance of Chromium compared to WireGL we attribute to using an early release of Chromium. The DAFFIE program is doing all of its graphics locally, so uses network communication only for control, hence has a very low network cost. DAFFIE is based on Performer, which uses display lists and other optimizations internally, leading to a significant performance improvement. The DAFFIE timings do not factor in the time for loading graphics models.

6. CONCLUSIONS AND FUTURE WORK

The experiment to determine if it is possible to build a large scale, high resolution, tiled, rear projected, passive stereo display system based on commodity components has been answered affirmatively. We have had a very enthusiastic response to the Deep Vision Display Wall. The illusion of depth is very effective. The large size makes for a large

field-of-view, increasing the feeling of immersion. The brightness of the projectors reduces many potential lighting problems. The use of commodity components keeps the price down. And the nature of a tiled Wall makes the size and price scalable for small and large installations.

This is very much an active and ongoing development effort. Future work includes finishing the final installation of the stereo twelve-tiled wall. We are working on designs for computer-controlled, actuator-driven physical positioners, as well as physical solutions for edge blending, and software solutions for color matching. We would like to automate these adjustments using camera-based techniques. To serve our user community we will be continuing to improve and expand our software infrastructure. To better the system, we will augment our current measurement and instrumentation capabilities.

7. REFERENCES

[1] Buck, G. Humphreys, and P. Hanrahan. Tracking Graphics State for Networked Rendering. *Proceedings of SIGGRAPH/Eurographics Workshop on Graphics Hardware 2000,* pp 87-95, August, 2000.

[2] M. Hereld, I. Judson, J. Paris, and R. Stevens. Developing Tiled Projection Display Systems. *Proceedings of the Fourth International Immersive Projection Technology Workshop*, June 19-20, 2000.

[3] M. Hereld, I. Judson, and R. Stevens. Introduction to Building Projection-based Tiled Display Systems. *IEEE Computer Graphics and Applications*, vol 20, no 4, pp 22-28, Spring/Summer 2000.

[4] G. Humphreys, I. Buck, M. Eldridge, and P. Hanrahan. Distributed Rendering for Scalable Displays. *Proceedings of IEEE Supercomputing 2000*, October 2000.

[5] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan. WireGL: A Scalable Graphics System for Clusters. *Proceedings of SIGGRAPH 2001*, pp 129-140, August, 2001.

[6] G. Humphreys and P. Hanrahan. A Distributed Graphics System for Large Tiled Displays. *Proceedings of IEEE Visualization '99, pp 215-224, October 1999*.

[7] G. Humphreys, M. Houston, Y.-R. Ng, R. Frank, S. Ahern, P. Kirchner, and J. T. Klosowski. Chromium: A Stream Processing Framework for Interactive Graphics on Clusters. *Proceedings of SIGGRAPH 2002,* pp 693-702, July 2002.

[8] K. Li, H. Chen, Y. Chen, D. W. Clark, P. Cook, S. Damianakis, G. Essl, A. Finkelstein, T. Funkhouser, A. Klein, Z. Liu, E. Praun, R. Samanta, B. Shedd, J. P. Singh, G. Tzanetakis and J. Zheng. Early Experiences and Challenges in Building and Using A Scalable Display Wall System. *IEEE Computer Graphics and Applications*, vol 20 no 4, pp 671-680, 2000.

[9] D. Pape, J. Anstey, G. Dawe. A Low-Cost Projection Based Virtual Reality Display. The Engineering Reality of Virtual Reality 2002, *SPIE Electronic Imaging: Science and Technology 2002*, San Jose, CA, 2002.

[10] R. Raskar. Immersive Planar Display using Roughly Aligned Projectors. *IEEE VR 2000*, New Brunswick, NJ, March 18-22, 2000.

[11] R. Raskar, J. van Baar, J. X. Chai. A Low-Cost Projector Mosaic with Fast Registration. *Mitsubishi Electric Research Laboratories Technical Report TR-2002-14*, February 2002.

[12] R. Raskar, P. Beardsley. A Self-Correcting Projector. *IEEE Computer Vision and Pattern Recognition (CVPR)*, Hawaii, December 2001.

[13] M. Stone. Color Balancing Experimental Projection Displays. *Proceedings of the 9th IST/SID Color Imaging Conference*, April 1, 2001.

[14] R. Surati, *Scalable Self-Calibrating Display Technology for Seamless Large-Scale Displays.* Ph.D. thesis, Massachusetts Institute of Technology, 1999.

[15] M. Walterman and F. Weinhaus. Antialiasing Warped Imagery using Look-up Table Based Methods for Adaptive Resampling. Proceedings of the S.P.I.E., vol 1567, pp 204-214, S.P.I.E. San Diego CA, July 22-26 1991.

[16] F. Weinhaus and M. Walterman. A Flexible Approach to Image Warping. Proceedings of the S.P.I.E., vol 1224, pp 108-122, S.P.I.E. Santa Clara, CA, Feb. 12-14 1990.