

# Cost-Effective Parallel Tiled Display

Mohamed Ibrahim  
Center of Excellence for Computer Applications  
University of Tennessee at Chattanooga  
Chattanooga, TN 37403  
(423) 425-4787  
Khartoum1@yahoo.com

Stephanie Smullen, Ph.D.  
Computer Science  
University of Tennessee at Chattanooga  
Chattanooga, TN 37403  
(423) 425-4395  
Stephanie-Smullen@utc.edu

## ABSTRACT

This paper describes the CECA VisLab Tiled Display—a reconfigurable, multi-screen, multi-computer, distributed image display system with zoom and pan functions. The project software utilizes distributed computing algorithms to control a parallel system to render high resolution images and scientific data on multiple displays. The system extends open source software to provide a large tiled image display using commodity hardware. The current version runs on Microsoft Windows but it can be easily ported to other platforms. The intent is to provide a cost-effective, multi-platform, distributed image rendering and display program usable by non-computer specialists. Possible applications include high image resolution display for physicists, astronomers, and biologists.

## 1. INTRODUCTION

Interest in visualizing different models on large displays from high resolution images has increased recently and many universities and research institutions have introduced, or are working on, a variety of solutions [5, 6, 15, 17, 18, 19]. This increased interest is fueled by many factors including the drop in prices of LCD displays (allowing for large display walls in relatively small spaces), the expansion of scientific visualization and geographic information systems (due to the large amount of geographic and scientific data available) and the advances in parallel computing (leading to low cost parallel systems).

Common computer display systems are not capable of handling and displaying large size and high resolution images effectively. Even if the computer is capable of processing the image, it may not be able to display all of the details. Simultaneously, large commercial display systems do exist, but they are too expensive for many applications.

The goal of the tiled display project is to create a cost-effective multi-platform distributed image display program with zoom and pan functions usable by non-computer specialists. The images are divided into tiles with each computer displaying a portion of the image. A variety of image formats (BMP, JPG,

PPM, GIF, and PNG) can be used on the display system.

The two major enabling technologies used to implement the system are OpenGL [16] and MPI [14].

## 2. SIMILAR WORKS

The CECA tiled display system can be easily installed using commodity hardware. It implements the distributed display using the existing MPI parallelization library. There are a number of other projects in this area of research today with different goals and different implementation methodologies.

The Geowall project uses fast graphics cards and inexpensive computers “to visualize structure and dynamics of the Earth in stereo” and help in the study of spatial relationships.[5] The images are generally produced on a single computer. The Geowall2 project uses 15 LCD panels driven by separate computers with high-end graphics cards to display a high resolution, tiled image[6].

Chromium, based on Stanford University WireGL project [21], “... is a system for interactive rendering on clusters of graphics workstations. [It] allows filtering and manipulation of OpenGL command streams for non-invasive rendering algorithms” [4]. Chromium is used for a number of exploratory applications. Princeton University’s Scalable Display Wall is “[An exploration and research] on how to build and use immersive computer systems ... building immersive systems for users to collaborate across space and time” [18]. Lawrence Livermore’s VIEWS Visualization Project also utilizes Chromium’s OpenGL abstraction that includes a parallel interface and a generalized stream processing layer [19]. The NCS VTK Geometry Viewer utilizes Chromium [15].

PowerWall technology, such as that developed at the University of Minnesota [17], is based on specialized hardware and software as well as open source software such as Chromium.

Chromium can be difficult to install and use. The MPI libraries that we use provide a much simpler way to

communicate between machines. OpenGL display programs do need to have the main routine tailored to use MPI, unlike those programs that utilize Chromium.

### 3. METHODOLOGY

The Microsoft Windows platform was selected as the target platform for the first implementation since other tiled image display systems run on UNIX/Linux systems. All of the libraries and tools used are available for other platforms enabling porting of future versions to different platforms.

#### 3.1 Tools and Libraries

The software systems and libraries used to create Tiled Display were gathered from several sites. To simplify the task of controlling multiple computers we used the VNC (the Virtual Network Controller). This software "...allows you to view and interact with one computer (the 'server') using a simple program (the 'viewer') on another computer anywhere on the Internet. The two computers don't even have to be the same type" [20]. We used VNC on a computer that was not part of the display wall so that we could easily launch applications on the display wall computers.

The Message Passing Interface [14] Channel (MPICH) [13] provides a solution to the process control, communication, and synchronization problems. It includes the following tools and libraries:

MPD, the MPICH Daemon is a process manager (launcher) for a cluster of Microsoft Windows NT/2000/XP computers. It enables users to start/stop processes on a cluster of computers from a single point of access. With this tool, the Tiled Display utility controls process creation and termination and specifies the target computer for each process. Additionally, it provides a system dependent authentication scheme for a controlled access to system resources.

MPIRegister is a tool to encrypt the account and password for users of the MPD processes launcher. Processes are launched in the context of the specified user account on all computers in the cluster. With this tool, the tiled display utility gains access to remote machines and launch process.

MPIRun is the tool that communicates with the MPD process launcher to start MPI applications. It is the interface through which applications can request the services of the MPD process launcher. With this tool, the tiled display utility gains access to MPD and issues process control requests. Process control requests include starting and stopping processes on remote computers, specifying the number of processes to create, and specifying the target host for each process.

The MPI Libraries provide communication and synchronization support for applications. Applications create communicator objects through which a group of processes communicate data objects and synchronize actions. The library provides support for synchronous and asynchronous data transfers, synchronization, and virtual topology creation and control. Using the data transfer functions, each process sends messages to either a single process or broadcast messages to the entire processes group at once. Processes coordinate command execution through the synchronization functions. The MPI library provides utility functions to create virtual Cartesian grids; display processes use this functionality to create an internal representation of the display grid to determine which grid portion each process controls.

The OpenGL [16] environment and libraries are used for image display. Open source libraries provide support for input and output of a variety of image file formats including JPEG, PPM, BMP, GIF, and PNG formats. The following open source libraries and source code files provide support for reading and displaying a variety of common image formats:

libjpeg [10] for the JPEG image format.

libgif [9] for the GIF image format.

libpng [11] for the PNG image format.

ReadBMP.cpp, ReadBMP.h [1] for a variety of BMP image formats.

PPM read functions by Russ Burdick [2].

The support for loading and displaying images in these formats is included from the StereoViewer program, a sister project to the Tiled Display project [3].

The GIMP Toolkit [8] is a multi-platform toolkit for creating graphical user interfaces. The toolkit provides a simple interface for creating a graphical user interface for the utility functions. With the help of the Glade [7] user interface construction program, creation of complex interfaces requires much less time and debugging efforts. We used these tools to create a GUI for user control of the system and zoom and pan functions (see Figure 1).

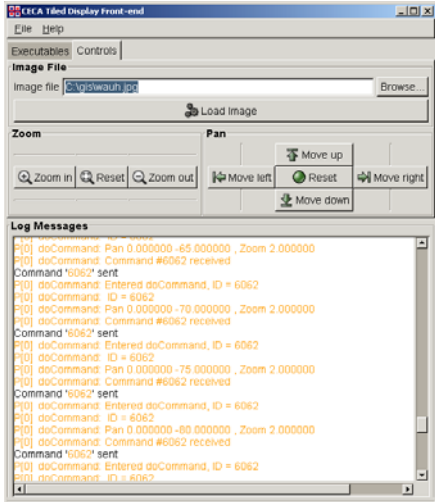


Figure 1. User Interface Control Panel

### 3.2 Implementation Details

A simple server/client design fits the problem of process control comfortably (see Figure 2). Once the server receives a command from the user interface module, it broadcasts it to all display clients for immediate execution. The communications between display processes is implemented through the functions provided by the MPI [14] library. The server sends the user's instructions to all display clients for synchronous execution.

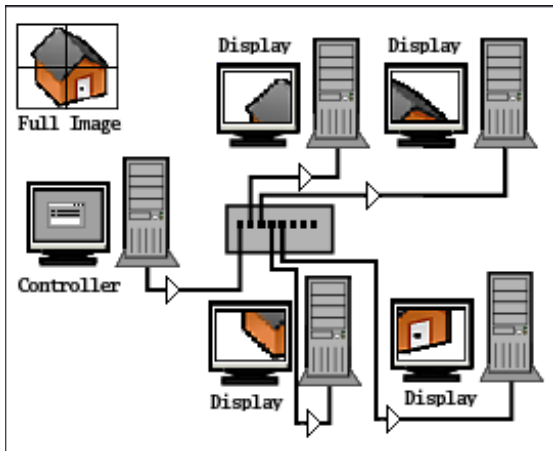


Figure 2. Server Client Design

A specialized TCP/IP-based communication protocol is specifically designed for this program and serves as the main channel between the user controller and the display system. It is a collection of simple predefined ASCII-formatted messages. The protocol messages are parsed using a shared parsing module, then interpreted and executed on both the client and the server. channel between the user controller and the display system. It is a collection of simple predefined ASCII-formatted messages. The protocol messages are parsed using a shared parsing module, then interpreted and executed on both the client and the server.

The graphical user interface is a separate process from the server/client processes group. Decoupling the user interface from the server—where commands are sent—allows for a cleaner and simpler UI design. In this case, the user interface process acquires input from the user and sends appropriate commands to the display wall server.

A standard OpenGL display program is modified so that the main routine that initializes the glut library functions is replaced by a subroutine. The main program then initializes MPI, sets up the grid and input commands. It starts the display thread which invokes the glut library initialization subroutine. The OpenGL display program's output is then distributed across the tiled display.

For image display, clients construct a virtual Cartesian grid—which may correspond to the physical layout of the displays—where each client controls a portion of the image (as in Figure 3).



Figure 3. Entire image displayed on four screens and partial image displayed at 200% zoom

Each client location in the virtual grid is dictated by the location of the display it controls, and the image it displays changes dynamically with the zoom factor and the pan displacement. Whenever the zoom factor or the pan displacement change, the client recalculates its new tile starting point and size and renders it on the display. These dynamic actions are synchronized to provide a seamless display for the user (see Figure 3).

### 3.3 The Display System

The display system constructed at the CECA VisLab is composed of four computers: one server and three display clients. Each display client is equipped with a dual output video card enabling us to construct a three by two tiled display grid. Once the utility is started from the server, users can specify the grid dimensions, start the display clients, and use the utility to view and study images (see Figure 4).



Figure 4. CECA's Tiled Display

Great detail is available when an image is displayed on the three by two tiled grid. The display card on each of these systems is currently set to Vertical Span (each column of monitors is controlled by one system unit). The resolution displayed on each monitor is 1280 by 1024 pixels. This results in a tiled image with a resolution of 3840 by 2048 pixels.

#### 4. SUMMARY AND CONCLUSIONS

The CECA VisLab Tiled Display is a cost-effective solution for displaying high resolution images. The hardware used for this demonstration is readily available:

- 3 Dell Dimension 4600 workstations (P4 2.8 GHz with 1.0 GB RAM)

- Dual output NVIDIA GeForce4 MX 440 with AGP8X

- 6 17" Flat Panels (3x2 Display configuration)

The total costs for this 7.86 megapixel display system is around \$6,000. Doubling the size (and cost) of the system--6 dual output workstations with 12 flat panels--would yield an impressive 15.72 megapixel display (5120 by 3072).

The current system can be installed on Microsoft Windows/XP systems. The user interface controller allows one to easily configure the system for any size display grid. Since no modifications need to be made to the software for different configurations, this system can be easily used by non-computer specialists. We provide detailed directions for the installation and use of this software on our web site [3].

We are working on enhancements to the display program. We plan to add image rotation and flip. Faster and smarter rendering and support for small memory graphics cards are also in development. The space occupied by the flat panel bezels can be disconcerting since it causes spaces between the frames of the image image [12]; we are working on resizing each panel's image to clip the portion of the image that would appear in the bezel's location. The human mind easily completes the missing portion of the image, and with zoom and pan functions the hidden portions of the image can be easily investigated.

Enhancements to add multiple logical displays from different source programs are also planned.

#### 5. ACKNOWLEDGEMENTS

The UTC Center of Excellence for Computer Applications supported this project.

#### 6. REFERENCES

- [1] Brown, Wayne, ReadBMP.cpp, <http://www1.mmu.edu.my/~akram/ReadBMP.cpp>
- [2] Burdick, Russell, <http://www-users.cs.umn.edu/~wburdick/geowall/viewer.html>
- [3] CECA VisLab project web site, <http://juno.utc.edu/~geowall>
- [4] Chromium, <http://chromium.sourceforge.net/>
- [5] Geowall, <http://geowall.geo.lsa.umich.edu/>
- [6] Geowall2, <http://www.evl.uic.edu/cavern/optiputer/geowall2.html>
- [7] Glade, <http://glade.gnome.org/>
- [8] GTK, <http://www.gtk.org/>
- [9] Libungif, Libgif, <http://gnuwin32.sourceforge.net/packages/libungif.htm>
- [10] libjpeg, <http://www.ijg.org/>
- [11] libpng, <http://www.libpng.org/pub/png/libpng.html>
- [12] Mackinlay, Jock D. And Jeffrey Heer, Wideband Displays: Mitigating Multiple Monitor Seams (*CHI 2004*) (Vienna, Austria, April 24-29, 2004) ([http://guir.berkeley.edu/pubs/chi2004/Wideband\\_short.pdf](http://guir.berkeley.edu/pubs/chi2004/Wideband_short.pdf)).
- [13] MPICH Implementation, <http://www-unix.mcs.anl.gov/mpi/mpich/>
- [14] MPI Specifications, <http://www-unix.mcs.anl.gov/mpi/>
- [15] NCSA VTK Geometry Viewer, <http://www.ncsa.uiuc.edu/TechFocus/Deployment/DBox/downloads.html>
- [16] OpenGL, <http://www.opengl.org/>
- [17] PowerWall, <http://www.lcse.umn.edu/research/powerwall/powerwall.html>
- [18] Scalable Display Wall, <http://www.cs.princeton.edu/omnimedia/index.html>
- [19] Views Visualization Project: Scalable Rendering Infrastructures, <http://www.llnl.gov/icc/sdd/img/infrastructures.shtml>
- [20] VNC, Virtual Network Controller, <http://www.netcontrol2.com/vnc/>
- [21] WireGL, <http://www.graphics.stanford.edu/software/wiregl/index.html>