13.2 PLYMOUTH STATE UNIVERSITY'S ELECTRONIC MAP WALL

James P. Koermer[*], Eric G. Hoffman and Theodore T. Wisniewski
Plymouth State University
Plymouth, New Hampshire

1. OVERVIEW

With the impending demise of the National Weather Service (NWS) facsimile map service and reductions in similar maps available over the Internet, the Plymouth State University (PSU) Meteorology Program conceived and created an electronic map wall. As the next generation replacement of the previous paper map wall, the electronic displays allow for full color renderings, animations, and automatic updates of a wide variety of meteorological products from models to satellite imagery.

The initial concept was conceived during the spring of 2002 when the PSU Development Office was seeking ideas for new technological initiatives that might be of interest to prospective donors or funding agencies in conjunction with the Meteorology Program and other sciences moving into a new building during the summer of 2003. There was to be a concerted effort to seek additional capital funding to bring state-of-the-art equipment and capabilities into this new facility. The original configuration envisioned 12 display monitors, each driven by a dedicated networked computer. A few would be readily interactive and the others would have pre-scheduled displays.

However, during the following months, several configurations were tested and we soon found that it was possible to drive multiple displays from a single CPU using additional graphics cards. This discovery allowed us to expand the vision to include more monitors, since we would need fewer CPUs. Further refinements of the plans came after we were able to get into the new science building during construction and scope out available space and after funding had been secured.

This paper will describe the details of the evolution of the PSU map wall project from the initial testing to final configuration. We will also discuss the problems encountered along the way and the methods and/or equipment that we used to overcome them.

[*]*Corresponding author address:* James P. Koermer, MSC#48, Meteorology Program, Plymouth State University, Plymouth, NH 03264; e-mail: koermer@plymouth.edu.

2. MULTIPLE DISPLAYS FROM A SINGLE CPU

The idea of a single CPU to drive multiple displays had some important benefits. Among them, would be a significantly lower cost per display, such as fewer CPUs, lower associated power and network infrastructure costs, and reduced systems administration requirements.

Our initial success involved using a spare DELL GX-400 and using multiple PCI graphics cards. This system seemed to accept multiple older NVIDIA 32MB cards without problem. Using FreeBSD as the operating system and XFree86 for the graphics, we were able to drive to separate independent displays. With these types of cards, we could get four cards working on this machine.

Since we would need additional video cards, we purchased some additional ATI Radeon 64MB PCI video cards. However, we had no significant success with these cards. They worked wonderfully alone, but seemed to have conflicts when a second card (identical or different type) was installed.

Since our computer cluster was scheduled for upgrade, we planned to use its DELL GX-1 computers that were being replaced to drive the displays rather than purchase new systems. However, we quickly discovered that these systems did not like to function with more than two video cards at a time. That had not been a problem with the GX-400. We suspected that the GX-1 problem may have been due to the video on the motherboard of the GX-1.

As a result, we had several bare bones PCs built with no onboard graphics. These would also give us a new hard drive and faster processor for the display systems. Much to our chagrin, these systems also balked with more than three cards installed. With our evolving plans that now included 32 LCD panels, we really wanted to have 4 or 5 displays per computer as a minimum. So, it was back to the drawing board.

Fortunately, we came across a fairly new and innovative graphics card—the MATROX G200 MMS (Multi-Monitor Series) Quad as shown in Figure 1. One of these PCI cards could drive four monitors and the product specifications indicated that up to four of these cards could be used in a single PC could actually drive 16 monitors.

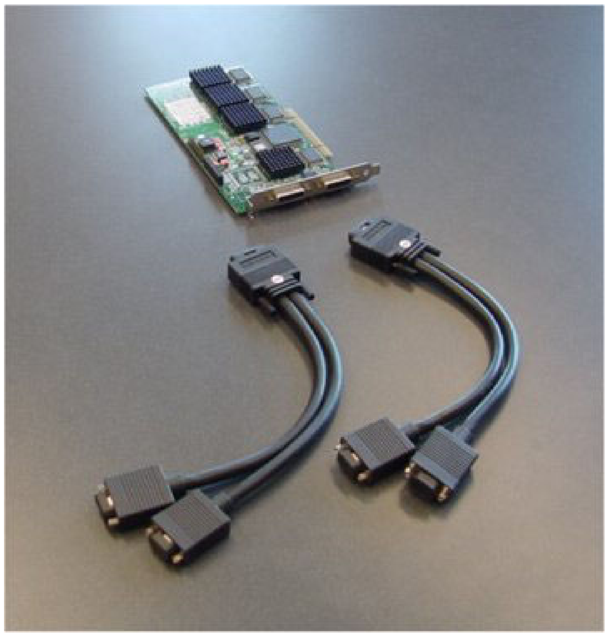

Figure 1. A single MATROX G200 Multi-Monitor Series PCI Video card and VGA connector cables.

This card proved to be the answer to our problem. It worked as advertised and eliminated video conflict issues. We decided to use two cards in four bare bones computers to drive the 32 LCD monitors. More than two cards would have been an extremely tight fit in the available PCI slots in these computers. It also would have meant that we would have needed longer VGA cables from some of the displays to reach the computer. We were also fairly certain that the memory requirements might not be sufficient to drive more than eight displays.

3. DISPLAY SOFTWARE FOR LCD MONITORS

As mentioned earlier, FreeBSD and XFree86 were used for the operating system and X-window system, respectively. The map wall computers were set up to each control eight LCD monitors that were accessible as display devices, :0.0, :0.1, :0.2, …, :0.7.

Since the Matrox G200 Quad has 32MB of onboard RAM, it was necessary to configure XFree86 to allocate 8MB and set MGASDRAM options for each video device on each G200. Initially, we had some difficulty with screen remnants (portions of a previous image not being erased) when a new image was displayed. This was overcome by tuning the video configuration with the following parameters for each video device (HWcursor to off) and (ShadowFB to true). Our original configuration also had the color depth set to 24-bit, but we backed this off to 16-bit color to enhance network performance and reduce memory requirements.

We tested a number of different applications for the purpose of displaying the images and image loops. Quickly we settled on two components of the ImageMagick application suite, “display” and “animate”. These applications make it very easy to direct the graphics to a specific monitor. Image files could be pulled from the local hard drive, a networked drive, or even through a web link. The applications also did not have to be run on the individual map wall computers, they could be run on a remote server and displayed on a map wall monitor. Other important benefits of these applications include that the LCD monitors could run in full screen mode (no window frames, buttons, etc.); that images could be resized on the fly to fill the full screen; and that they could be embedded in UNIX shell scripts.

When included in a looping script, the “display” command has the added benefit that newly updated images will automatically be displayed. As a result, the script can run and loop continuously with no special intervention. On the other hand, the “animate” command does its own internal looping and in order to access new images for the loop, the program has to be stopped and then restarted.

The appendix contains several examples of scripts. The first script shows how the “display” command can be used to run a slide show of various surface maps, which have been locally generated at Plymouth State. Most of these products are generated at a 1024x768 pixel-resolution solely for the map wall screens that are set for that resolution, but this script also shows how we can use a smaller raw image and enlarge it to fit the screen. Comments are provided in the script to help explain most of the details.

The second example shows the short script used to animate a sequence of the last 24 Gray (ME) NEXRAD images that are locally produced. Since they also were not built at 1024x768 resolution, they are resized by using the “geometry” option. As indicated earlier, new images for the loop are not automatically accessed with the “animate” command, hence these processes need to be stopped and then restarted, when new data become available. This is done through another script that is run every minute. If no new data are found, it exits; Otherwise, it will find the processes, stop them and then restart the animation.

4. SETUP FOR PLASMA SCREENS

As part of the electronic map wall, space and resources allowed us to add two large plasma screen displays to the configuration. We deemed it important that these be fully interactive, but still capable of running automated displays like the LCD monitors. Because of this, we felt that each plasma screen should have its own dedicated CPU and use MICROSOFT XP Professional for the operating system.

XP offered all the standard features, such as web browsing. We could also run the FX-Net weather visualization package from the Forecast Systems Laboratory. By using Xwin32, users could also run UNIX applications (e.g. McIDAS, GEMPAK, WXP, and IDV) for accessing and displaying weather data from the program's UNIX servers.

When not in interactive mode, these systems can use MICROSOFT PowerPoint and an add-in called "Live Image Update" to run a looping slide show sequence and/or multiple animations (e.g. animated GIFs), similar to the other LCD systems. The add-in insures that the latest images will automatically get displayed.

5. FINAL MAP WALL CONFIGURATION

During the testing of hardware and software components, we were also planning for the site preparation needed for the map wall. After measuring the available space, identifying the most likely sizes of the display devices, and estimating the number of CPUs needed to drive the displays, we designed a large cabinet that was only 6 inches deep on top with a wider base to hold the CPUs. The LCD and plasma monitors were to be mounted on the large face of the cabinets with the VGA and power cables fed from the monitors, through holes in the face and running behinds the panel down to the CPUs below.

Based on these plans, a custom cabinet was built and installed against a wall that was augmented with many additional power outlets and network connections. The 32 19" LCD panels and two 42" plasma monitors were mounted and connected to a total of six CPUs. Four are used with the MATROX cards to support the 32 LCD displays and the other two drive the plasma screens. The final cabinet layout is shown below in Figure 2.

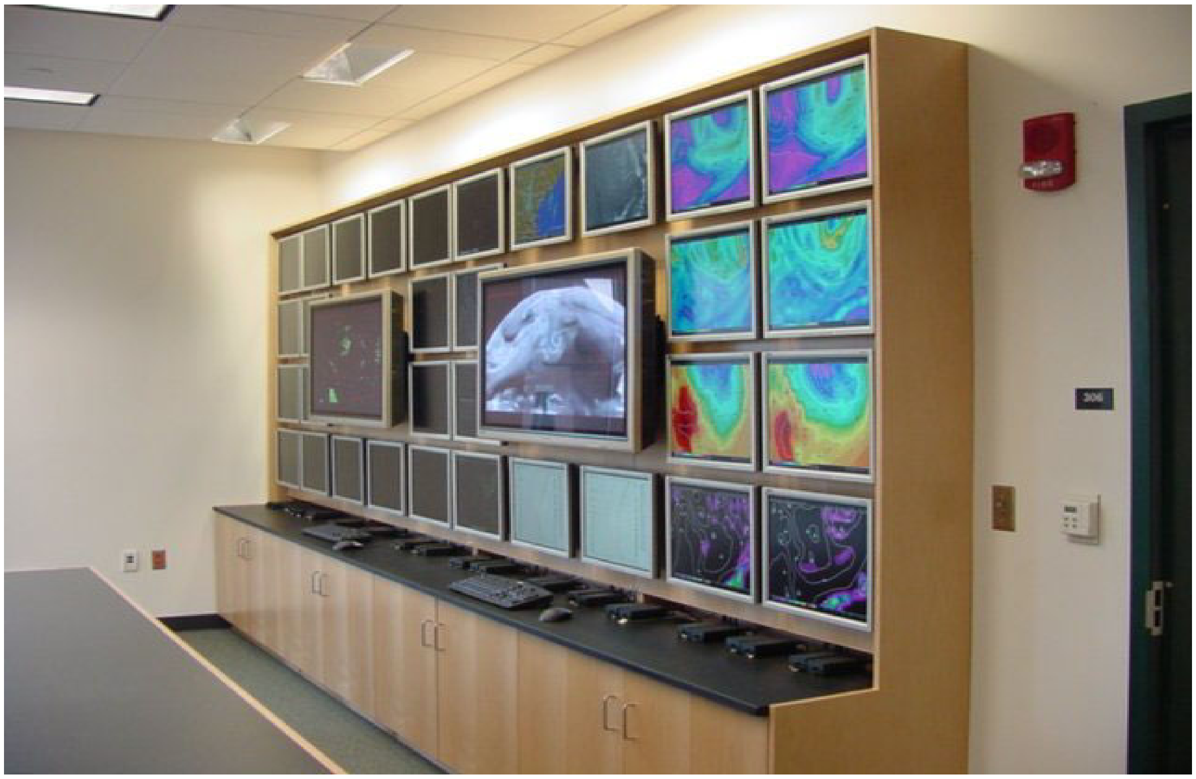

Figure 2. The Plymouth State University electronic map wall cabinet display layout. CPUs are housed in the wide base behind the cabinet doors.

Many of the currently displayed products are either Plymouth State University Weather Center web products, but some maps were designed exclusively for the map wall. There is a wide variety of products including forecast model maps, satellite images/loops, radar images/loops, surface and upper air maps, thermodynamic diagrams based on radiosonde observation and forecast soundings, max/min/24-hour precipitation maps, meteograms, and NLDN lightning summary/loops.

6. CURRENT ACTIVITIES AND FUTURE PLANS

The displayed products can be considered works in progress. We are still experimenting with timings, layout, contents, and animations—both locally produced and some accessible via the web. Some ideas include displaying the last several days of upper air charts along with the current ones either as an animation or slide show.

As we start to settle on products, we also want to set up some long-term archives for displayed maps. Access to these data would be available for recall via an interactive web page, so that they could be used to review interesting situations and for other case studies.

On the hardware side, our only surprise after getting the systems functioning has been the high memory utilization both for the windowing system and for running the display and animate programs. CPU utilization is not so pronounced, but we are currently acquiring additional memory (2GB up from 512MB) for the LCD map wall computers. This will allow us to run the display or animate scripts directly from the map wall computers to reduce load on the network. Instead, we will be able to push new images to the map wall computers as they become available. This change will also keep display operations on fewer computers, making the map wall system easier to administer.

7. ACKNOWLEGEMENT

The Plymouth State map wall project was funded through NOAA award, NA03NWS4680003.

APPENDIX

```
#!/bin/csh -f
# Script: display.surface
# This script displays the latest CONUS surface maps including NCEP frontal
# positions, temperatures, dewpoint temperatures, weather depiction,
# streamlines, heat indices, and wind chill temperatures in sequence at 15
# second intervals.
# Notes:
#     - All maps are sent to mapwall03 display :0.0
#     - Display resolution is set to 1024x768
#     - Frontal map needs to be resized with the "geometry" option
#
#
# Switch to directory with maps
cd /home/mapwall/maps
#
START:
display -display mapwall03:0.0 -size 1024x768 -geometry 1024x768 \
-window root -backdrop -delay 1500 psc_usfront.gif
display -display mapwall03:0.0 -size 1024x768 -window root -backdrop \
-delay 1500 temp.gif dewp.gif depict.gif strm.gif heat.gif wchill.gif
goto START
#
# Note 1500 delay = 15 seconds (each unit is 1/100 of a second)
#
```

Example 1. Script to "display" a sequence of maps.

```
#!/bin/csh –f
# Script: animate.GYX
# This script animates the last 24 GYX NEXRAD images at .4 second intervals.
#
# Notes:
#     - Animation is sent to mapwall03 display :0.7
#     - Display resolution is set to 1024x768
#     - Images need to be resized with the “geometry” option
#
#
# Switch to directory with maps
cd /home/mapwall/maps
#
# Start animation
animate -display mapwall03:0.7 -geometry 1024x768 -size 1024x768 -window root
-backdrop -delay 40 GYX.101.gif GYX.102.gif GYX.103.gif GYX.104.gif \
GYX.105.gif GYX.106.gif GYX.107.gif GYX.108.gif GYX.109.gif GYX.110.gif \
GYX.111.gif GYX.112.gif GYX.113.gif GYX.114.gif GYX.115.gif GYX.116.gif \
GYX.117.gif GYX.118.gif GYX.119.gif GYX.120.gif GYX.121.gif GYX.122.gif \
GYX.123.gif GYX.124.gif
exit
```

Example 2. Script to “animate” a sequence of maps.

```
#!/bin/csh -f
# Script: update.GYX
# This script checks to see if a new GYX radar image has been built. If it
# has, the currently running animation processes are found and killed, then
# the animation is restarted.
#
#
setenv PATH /usr/bin:/etc:/usr/sbin:/bin:/usr/X11R6/bin:/sbin:/usr/local/bin
#
# Check to see if a new GYX.gif was built
#
cd /home/mapwall/maps
set current=`ls -l GYX.gif | cut -c41-45`
set last=`cat last_GYX`
if ( "$current" == "$last") exit
echo $current > last_GYX
#
# Get the associated process PIDs and kill processes
#
set pid1=`ps -auwx -w | grep animate.GYX | grep -v grep | cut -c8-12`
set pid2=`ps -auwx -w | grep mapwall03:0.7 | grep -v grep | cut -c8-12`
echo pid1 is $pid1
echo pid2 is $pid2
kill -9 $pid1 $pid2
#
# Start animation program
/home/mapwall/slide_shows/animate.GYX &
#
exit
```

Example 3. Script used to update a running animation.