Computer Organization Design Project

— A Networked Painting Program —
(With Voice Mail)

ECE 341F

TA: Dan Sladic

Lab Day: Monday 9 am

December 3, 1997.

David Tam & Thomas Huang

Introduction and Motivation

Our goal in this design project was to implement a network-oriented ASCII paint program in which two networked terminals can work cooperatively on the same drawing board using mice. Additionally, the program would be capable of transferring voice files between terminals.

The idea was initially motivated by the fact that we have not had any experiment nor experience that relates to networked data-transferring, and we considered this design project a great opportunity for us to explore this area. Also, we wanted to make our project a simple version of an application that would be useful in real-life. That was why the voice-transferring component was added, since it will greatly enhance the communication between two remote terminals and hence increasing the efficiency of the collaboration.

Description of the Design Project

The software can be broken down to three parts: a painting program, a sound processor, and a network data-transferring program. A mouse driver and a simple pseudo-protocol were developed to handle the interaction between software and hardware.

Originally, we planned to implement our data-transferring with the use of 'Write', a Unix command that can be used to send text messages between users within a network. A pseudo-protocol was developed around 'Write' to distinguish between various types of data-transferring: sound bytes, the remote active brush, commands issued by the mouse, and commands issued by the keyboard.

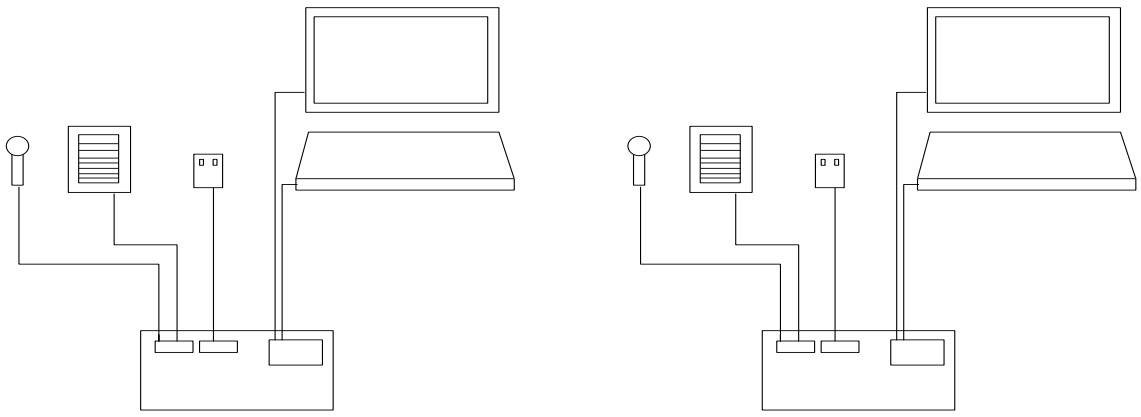
In order to achieve real-time voice-transferring, the sound processor will lower the sampling rate to meet the network's baud rate constraint. When the real-time conversation proves unfeasible, a proposed alternative was to implement voice mail with reasonably fast

speed to minimize delay while achieving the goal of enhanced communication between the two users.

The paint program is mouse-driven, but can take input from the keyboard as well. When used as a stand-alone software, the user can set the active brush to any ASCII character by pressing that character on the keyboard once. The drawing board's resolution is set at 25x80, and the functions to erase a pixel and to clear the whole drawing board are also included.

The main program, packaged with the above three components, keeps looping to check if there is any input from the local and the remote terminal. In network-mode Paint, the local terminal will keep track of the remote cursor coordinates. When the user at the remote terminal draws a pixel, the local cursor will jump to the corresponding pixel, draw with the remote active brush, and then jump back to the original location.

A detailed schematic of our hardware layout is shown below:



Results

The networked painting portion of the design project was completed, however, the voice mail feature was coded but not implemented. This was due to the difficulty of the project and a number of complications. Testing of the design was very time-consuming and resource-intensive as it required us to occupy two terminals, two mice, and two sound boards. It was time-consuming because there were many possible variables to consider during the debugging sessions, including software bugs and hardware failures. Due to the time constraints, and limited lab hours, we were unable to test and debug the voice mail portion of the project.

Originally, we planned to send real-time, streamed audio across the network but after a few initial tests, it was found to be impractical. The speed of the DUART serial ports were specified to be 9600 bits per second. Dividing this number by 8 gives us a speed of 1200 bytes per second. Since each byte represents a sampled sound value from the sound board, we can see that the sampling rate would be limited to 1200 samples per second. Equivalently, this would be 1.2 kHz. We were not sure whether we could get recognizable speech at this rate. We tested this limitation in the first lab and found the sampling rate to be totally inadequate. We could only hear a few ‘bleeps' and ‘blops' every few seconds. Based on these results, we decided to implement the voice mail feature instead, since it would give higher quality samples.

Writing and debugging the mouse driver was completed fairly quickly and easily. We were able to paint ASCII characters on the local terminal without too much difficulty. The 'clear screen' feature was also implemented successfully. The major difficulty encountered was in receiving input from the network (DUART 0, port B). Once the mouse driver functioned correctly on the local terminal, we tested to see if it would transmit over the network. From our test results, we saw that the correct data was sent from the local terminal to the remote terminal. We performed this test by simply running the program on the local terminal and observing the raw data that was received by the remote terminal.

We found discontinuities when we tested to see if the program was able to receive and process the raw data correctly. By examining the mouse-tracking variables ROW2 and COLUMN2, we found that the program did not keep proper count of these variables. This problem occurred whenever a remote mouse was used to generate the raw data. However, when the cursor pad was used to simulate slowly changing mouse movements, the program kept perfect track of the remote cursor. These opposing results were quite disturbing. The individual pieces of the painting program seemed to work fine as separate components, but when they were glued together (in a very simple manner), they failed to work as a functional whole. Quite a large amount of time was spent debugging this problem. Eventually, we hypothesized that the network was the source of the problem. Our extensive use of the EECG network placed our project in uncharted waters since, in the past, not many students or TAs have extensively used the network for purposes such as ours.

By the last session of the lab, we had analyzed the problem and determined the probable source. From a few modifications to the program (to display debugging information) we could see that the program was unable to capture all of the raw data coming from the network. Initially, we assumed that the input would be buffered correctly, following the handshaking method we had seen in our lectures. Supposedly, the remote terminal would not send the next character to the local terminal until the local terminal had read the data in the receive buffer. Our tests showed that the Unix 'Write' command did not follow this convention. We realized that this was the source of the problem and we quickly designed a solution. Instead of sending and receiving data through the EECG network, we decided to connect the two terminals directly together through DUART 0, port B. This change in plans required only a few minor modifications to the code.

Once the modifications were made, our tests showed that the paint program functioned correctly in the new setup. However, due to either a bad serial cable or bad RS-232-C connection jacks, the paint program was functional in one direction only. We were able to paint on the local terminal and have the results displayed on the remote terminal as well. Using the mouse on the remote terminal caused garbled characters to appear on the local terminal. Based on the fact that both terminals were executing the same code and only one of them was showing signs of trouble, we concluded that the program must have been functioning correctly. The only uncertain variables left to consider were the cable and the connection jacks.

When we reached this stage of the design, our lab time had run out. By the end, the paint program was functioning correctly but the sound component had not been implemented.

If We Did It Again

We realized that we were over-ambitious in our design. Implementing just the painting aspect of the project was a very large and difficult task because it required use of the EECG network and exhaustive testing and debugging. We have learned that troubling-shooting network problems is very difficult and time-consuming.

If we were to do our project again, we would use interrupts rather than the polling technique for detecting mouse, keyboard, and network input. This would make the program run smoother and quicker. We would also use a different method to paint on the remote terminal. Instead of having the local terminal keep track of the remote cursor position, we would send only the remote mouse position and the paint character when the remote terminal painted to the screen. This method of communication could potentially speed up the program because it no longer needs to keep track of the remote cursor position. Instead, the local terminal would keep track of its own position, reducing network traffic. We would not use the EECG network as the communication medium. Rather, we would connect two terminals directly together via a DUART port. This decision would allow use to by-pass many of the network problems we had experienced. Finally, we would implement either just the paint program or just the voice mail program. The time-constraints prevented us from implementing both features.

Appendix A

Motorola 68000 Assembly Code of Design Project

Appendix B

Project Approval Form: Description and Milestones