The Classroom of the 21st Century: The Interactive Learning Wall

Richard R. Eckert
Computer Science Department
SUNY at Binghamton
Binghamton, NY 13902
+1 607 777 4365
reckert@binghamton.edu

Jason A. Moore
Air Force Research Lab/Rome Research Site
525 Brooks Road
Rome, NY 13441
+1 315 330 4192
moorej@rl.af.mil

ABSTRACT

The Interactive Learning Wall is a Windows/PC-based virtual blackboard system that can be controlled remotely by a classroom instructor and/or students. A laser pointer used by the instructor emulates mouse actions on a computer projection screen. An inexpensive video camera sends each projected frame to a video capture card in the computer; our software detects the bright laser spot, moves a screen cursor, and performs other mouse operations according to user actions. The system frees the instructor to move about the classroom as he controls the presentation. Other software allows students, from their laptop computers, to request control of the main computer's mouse and keyboard over a local network. This facilitates student interaction with the main computer's screen to perform tasks like: go to specific parts of the presentation; enter text; annotate diagrams; run programs. Both systems can help increase interactivity between instructor and students in a large classroom environment. Technical details are given in this paper.

Keywords

Computers in education, remote mouse, virtual blackboard, laser pointer, mouse emulator, wireless LAN, learning wall.

INTRODUCTION

Teaching and learning involve the effective exchange of information between an expert and a group of novices. This kind of activity permits the novice to understand and perform new tasks. Some of the most effective teaching takes place in one-on-one or small group settings. This kind of environment provides for individual attention and an easy, non-intimidating rapport between student and teacher that is conducive to learning. As the material to be learned becomes more complex and voluminous, the number of learners (novices) becomes much greater than the number of teachers (experts), so that the small-group setting becomes impractical. To facilitate presentation to larger student groups, the teacher must have access to some sort of display device to post lessons and/or notes. Artifacts such as blackboards, whiteboards, overhead projection systems, and computerized projection systems have become prevalent. Unfortunately, in large-group settings that use this kind of equipment, interaction between the teacher and individual students is kept to a minimum at least partly because the teacher is tethered to the display system and is therefore remote from the students. This limitation is particularly serious in a large lecture hall environment. Exacerbating the problem is the fact that the remoteness of the instructor in this kind of setting makes it impossible for most students to ask questions, clarify doubts, or get any kind of individual attention.

THE INTERACTIVE LEARNING WALL

The Advanced Displays and Intelligent Interfaces group at the Rome, NY Air Force Research Laboratory has created a sophisticated and expensive presentation system called the "Interactive DataWall" that consists of an ultra-high-resolution wall display driven by an SGI ONYX computer with various types of wireless input devices shared among multiple users. The devices include: speech interaction, camera-tracked laser pointers, and flight sticks. The purpose of the DataWall is to display large amounts of information in a contiguous area and to provide multiple users with an intuitive and unencumbered means of accessing and manipulating the data. [1]

In this project, an inexpensive (but in some ways enhanced), scaled-down PC/Windows-based version of the Rome Laboratory facility, that we call the Interactive Learning Wall, was designed and implemented for use in a civilian environment. More precisely, the Learning Wall adapts and adds to the Air Force technology so as to

benefit a modern classroom. Specifically, laser pointer mouse emulation and local network communication between student and instructor computers are implemented. Figure 1 illustrates the interactive learning wall setup.

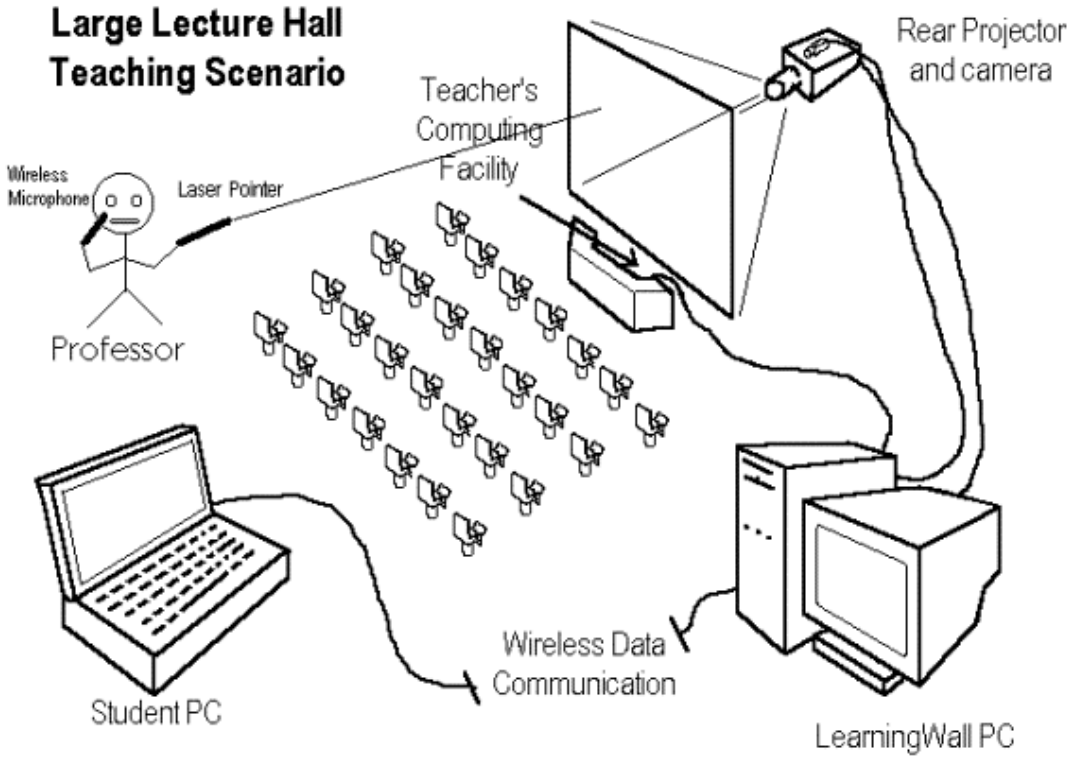

Figure 1. Rear Screen Learning Wall Setup.

THE LASER POINTER MOUSE EMULATION SYSTEM

Although a rear-screen configuration is illustrated in Figure 1, the system can also be used with the projector and camera in front of the projection screen. It is important, however, in a front-screen setup that the operator not shadow the image being displayed by the projector and viewed by the camera. The projector projects whatever content the instructor is displaying on the screen. This could be a PowerPoint™ slide presentation or any other visual material from the computer. The professor uses the laser pointer to point at and interact with the material on the screen. The interaction is very reminiscent of that obtained with a standard mouse. An inexpensive video camera looks at the screen and sends each new frame to a video capture card in the computer. Our software detects the bright laser spot on the screen, computes the corresponding screen location, and sends the system appropriate Windows "mouse messages" that make a mouse cursor image move according to user actions taken with the laser pointer.

In our implementation, the lower right-hand corner of the screen displays a white box that shows the current mode of operation. Upon startup it says LEFT. This means that if the user "clicks" with the laser pointer, it generates the same effect that clicking the left mouse button would generate. The interface is similar to that used with touch pads on some laptop computers. The first time the user directs the laser pointer toward the screen it is treated as a location identifier. In other words the user needs to accurately point at the location where he or she would like to push the button. To generate a single click the user releases the laser pointer button, and within one second in close proximity to the previous location, presses it again. To generate a double click the process is repeated. This procedure facilitates achieving accuracy at a distance in spite of unsteadiness or jitter of the user's hand. In our implementation, the first time the laser point is seen is used as an "aim" and the second is a "fire."

The other modes of operation are RIGHT and DRAW, and they are selected by pointing and releasing the laser pointer within the little white box in the lower right-hand corner of the screen. In RIGHT mode the program generates right mouse button clicks in response to laser pointer "clicks". In DRAW mode a color bar appears to the right of the screen and displays four colors: yellow, red, blue and green. The active color is highlighted by a white border. To draw the user only needs to press the laser pointer button once, and the program will continue to draw connected line segments that follow the mouse cursor until

the laser pointer is released. To switch colors the user points and releases the laser pointer over the desired color.

The Computer

Since real-time digital video processing is required, a computer with a video capture card is necessary to allow implementation of the Learning Wall. The system also has to simultaneously handle user, system, and application events. We used a 300 MHz, 128 MB Pentium II computer with a Computer Eyes/PCI video capture card. This was much more than adequate. Tests we performed determined that the maximum utilization of the processor was only 17 percent while running the Learning Wall's laser pointer software.

The Laser Pointer Component

The software to run the laser pointer mouse emulation system is broken into three pieces. The first and most hardware-related software component, DETECT, performs low-level detection of the laser pointer in the camera's viewable area. There are two other applications that use the data provided by this program: INIT during the initialization process and LASERPEN during normal operation. Both of these helper applications run with a window title of LASERXY. INIT performs the computations necessary to convert the position of the laser beam from camera coordinates to screen display coordinates. LASERXY interprets user laser actions and generates appropriate mouse emulation messages. Each program asks Windows to register a WM_LASER message through which communication takes place. All applications that request a message with that title will be returned the same unique identifier. Figures 2 and 3 illustrate the relationships between these programs.

The camera captures individual frames at 320 pixels in the x dimension and 240 pixels in the y dimension at as close to 30 frames per second as possible. Depending on how taxed the system is from user commands the frame rate can change. Each individual frame is processed to see if there was an instance of a pixel being brighter than a certain

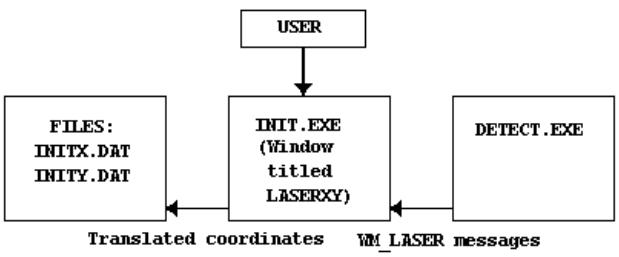

Figure 2. The DETECT and INIT Programs.

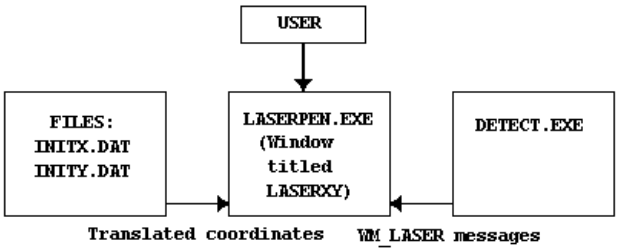

Figure 3. The DETECT and LASERPEN Programs.

threshold. The environment that is present determines this threshold. DETECT can perform some functions as a stand-alone program, but it also cooperates with other applications through Windows messages. As a stand-alone application it displays the output of the camera in a separate window; as a cooperative application it provides other programs tracking information from the laser pointer in camera coordinates. To improve speed, the tracking is performed by discarding half of the actual pixels that are visible to the camera. After an instance of a pixel above the threshold is found, DETECT discerns if the pixel that was most recently skipped had levels of red, green and blue greater than the threshold. If so, the coordinates of the skipped pixel are used. This provides the accuracy of the entire resolution of the camera along with the speed of checking only half the pixels initially. Also, to improve efficiency, the program stops looking for an instance of the laser beam in a particular frame once it has been seen. In order to run as a cooperative application, DETECT asks Windows to return the handle to LASERXY, the program that spawned it. This allows DETECT to send Windows messages to that specific application instead of a WM_BROADCAST message that could severely slow down the system. Once the camera distinguishes an instance of the laser beam, it then sends messages to either LASERPEN or INIT with the laser pointer's position in the frame.

Sending the appropriate message involves knowing information about the previous frame as well. If in the previous frame an instance of the laser pointer wasn't seen, the program sends the coordinates along with a first_seen flag. If it was seen in the last frame, then it sends a continue_seen flag. When the laser pointer is not seen, but previously it was seen, the program sends (0,0) for the camera coordinates and a flag of not_seen. To minimize the number of Windows messages, the application doesn't continually send not_seen messages. Figure 4 is a state

diagram showing the message states of DETECT. Figure 5 is a flow chart showing the logic involved in DETECT.

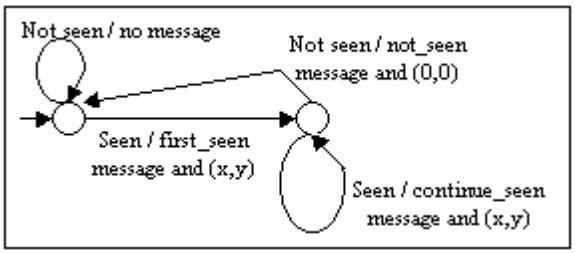

Figure 4. DETECT Message States.

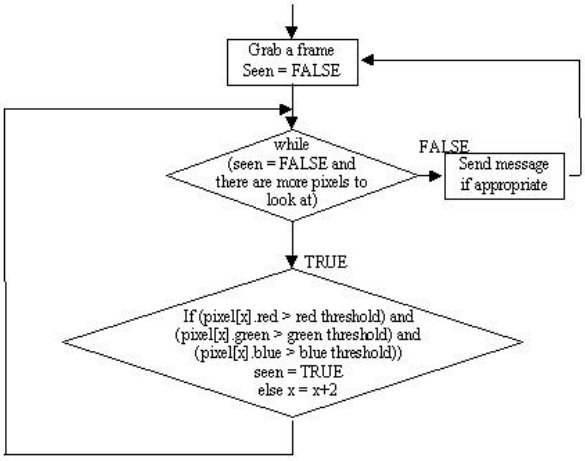

Figure 5. A High-Level Flow Chart of the DETECT Program Logic.

Menu Choices

If DETECT, running as a stand-alone program, wasn't spawned by an application with the title of LASERXY, it starts up in a normal window, displaying the captured frames along with the coordinates of the point where the laser pointer was seen. This can be toggled on and off by going to the *Image* menu and selecting *Toggle Output*. If an instance of the either INIT or LASERPEN is running, the program toggles the output off and minimizes itself. The default camera position setting is front screen and can be toggled under the *Image* menu as well. While being used for detection of the laser pointer, DETECT must capture at 320 pixels by 240 pixels, but for use as a stand-alone application, under *Image/Preview Size*, the user has many resolution options: 80x60, 160x120, 320x240, 480x360, 560x420, and 640x480.

The Camera to Screen Transformation (INIT)

In a perfect environment the camera would be inside the projector and see exactly as much as the projector. In addition, the lens would be parallel to the screen so that every pixel is the same size and shape as every other pixel. Unfortunately this is not the case, so there is a need to tell LASERXY some specific information about the geometry of the camera and the image being projected. This is done through an initialization procedure in which two functions (actually arrays) produce a discrete screen display coordinate for every visible camera coordinate generated:

x display coordinate = Fx (x,y camera coordinate)
y display coordinate = Fy (x,y camera coordinate)

The resulting transformation data is then saved to disk for LASERPEN to read.

Upon running INIT the arrow icon is replaced with a series of four targets toward which the user should point the laser. These target points are in the four corners of the projection screen. Since the user's hand is not perfectly steady the software expects the user to point at each target and release the pointer when on target. The point where the laser was last seen is taken as the initialization point. The resulting four points are stored and used in the camera-to-screen transformation. An exaggerated view of what the camera sees of the projected image and the four target points (P1, P2, P3, P4) is shown in Figure 6.

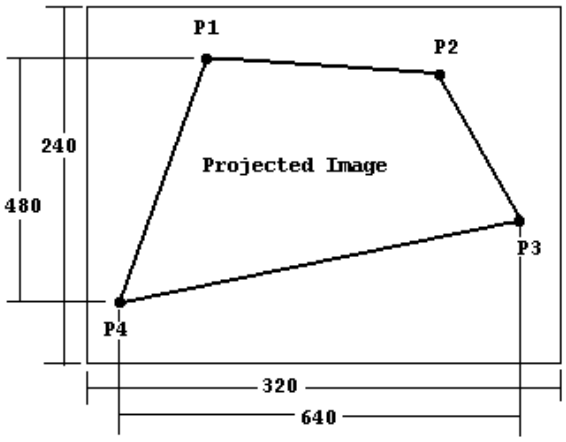

Figure 6. Simulated Camera Viewable Area.

In this figure projected image is 640 by 480 pixels, but in terms of camera viewable coordinates it is only 320 by 240 pixels. To perform the conversion, first the intersection points of the "vertical" and "horizontal" lines bounding the area enclosed by the four target points are computed using the parametric equations for straight lines. Any point within the camera's viewable area is converted to its scaled

screen coordinates by projecting a line from the point to each of the two previously calculated intersection points. As shown in Figure 7 on the next page, those lines will each intersect an opposing screen boundary line. The intersection points are stored and the result is evaluated as a percentage of the overall line length and linearly scaled. In this example, the x values are scaled as a percentage of 640 pixels and the y values are scaled as a percentage of 480 pixels. This information is calculated for every point in the camera's viewable area and is stored in two two-dimensional arrays, one for x coordinate values and one for y coordinate values. At the end of the initialization program, the two arrays are stored to disk. This ensures that once a good initialization is calculated, it will never have to be repeated until there is a physical change in the setup.

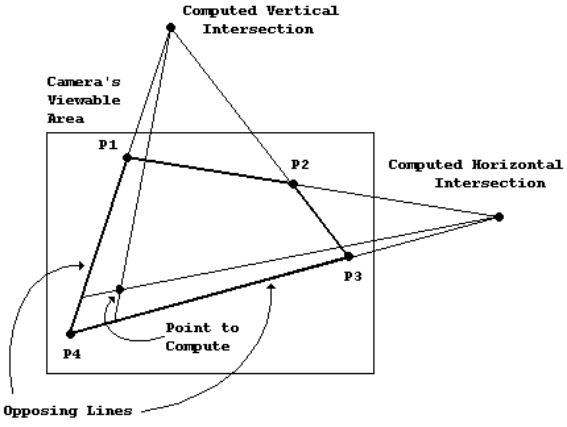

Figure 7. The Camera to Screen Transformation.

The previous example dealt with displayed image resolutions of only 640 by 480 pixels. In practice the software can handle dynamic changes in resolution. Instead of calculating the points in reference to a display of 640 by 480 pixels, it calculates the points in reference to a display of 512,000 by 288,000 pixels. This is a common factor that can be evenly divided by 640-by-480 pixels, 800-by-600 pixels and 1024-by-768 pixels. By computing the points in this fashion, the program doesn't have to know the resolution of the actual display, making it much more general and useful. This also allows calibration to occur regardless of whether the resolution of the screen is the same during calibration and normal operations.

Mouse Messages from Laser Actions (LASERPEN)

The final application that works directly with the laser pointer is called LASERPEN. This program is the meat and bones of the user interface since it listens to Windows messages and moves the pointer and generates mouse messages accordingly. The program begins by opening the files that were created by INIT and stores them into two arrays, one for the x-coordinates and one for the y-coordinates. After loading the files, a correction must be performed to change from the general (x,y) stored calibration values to the (x,y) values for the current resolution. For resolutions of 640 by 480 pixels, the (x,y) values are divided by (800,600); for resolutions of 800 by 600, they are divided by (640,480); for resolutions of 1024 by 768, they are divided by (500,375). When a resolution change occurs, the program reloads the files from the disk and redoes these calculations. By performing the calculations only once for each resolution change, the program removes two divisions that would otherwise need to be done at every detected instance of the laser beam.

LASERPEN doesn't open a standard window like most applications. Instead it creates a visible popup window in the lower right hand corner of the screen that is 25 pixels high and 50 pixels wide. This window has no borders, is solid white, and cannot be overwritten by other application windows. It is used to display the current emulation mode of the laser pointer (LEFT, RIGHT, or DRAW).

The Microsoft Windows Win 32 API provides functions that make generating button clicks fairly simple. The function mouse_event() takes as parameters the desired button to press and the screen location. To alleviate the problem of user unsteadiness, the following approach is taken: When the laser pointer is turned off, an imaginary area equivalent to 1/16 of the overall horizontal resolution is drawn around the point. If the laser is detected again within one second of the last time it was detected and falls within this imaginary area, then LASERPEN generates the proper mouse button down message (LEFT or RIGHT, depending on the current mode). In either case a button up message is generated only when the pointer is not detected. To generate double clicks the procedure is repeated; however, the mouse sensitivity under Windows needs to be reduced first.

If the program is in DRAW mode, instead of producing a button click, laser beam detection is used to generate the endpoint of a line segment. While the program still sees the laser pointer, a line is drawn to that point from its previous location and that location is updated with the current location—thereby drawing a segmented curve that follows the motion of the laser pointer.

In all modes of operation, any time the laser beam moves, a mouse move message is generated and the mouse pointer is positioned at its new location on the screen by invoking the Windows function SetCursorPos(). All the required communication between these programs is done with WM_LASER messages. As seen in Figures 2 and 3, LASERPEN receives the WM_LASER message from

DETECT. This message includes the relative location and the first_seen, continue_seen, and not_seen flags. When LASERPEN receives this message, a lookup is performed into the x and y coordinate arrays and the call is made to SetCursorPos() using those coordinate values as parameter values.

THE NETWORK COMMUNICATION COMPONENT

There are many ways of establishing TCP/IP network connectivity between the instructor's computer and student laptop computers. For example, in many new lecture halls student seats are hardwired for network access. Another alternative is to set up a wireless Local Area Network (WLAN). We opted for the latter. Our WLAN consists of a Lucent Technologies WavePOINT II base station connected via a 10base2 cable to the server machine. The laptops then use type 2 slot PCMCIA WaveLAN cards with antennas that are attached to the wires coming out of the PCMCIA cards. The base stations and the PCMCIA cards auto-connect when they are in proximity of each other and provide the laptops with access to the Intranet that the server provides. The LAN can provide up to 4 Mbps bandwidth broadcast at 2.4 GHz split among all users in the range of the base station. Connectivity is limited to a range of 200m, which is more than enough for even our largest lecture hall. The only requirement on the base station is that it be mounted in plain view on the wall and that its antenna face the classroom. There are no software modifications that have to be made to existing TCP/IP based applications as the Wireless LAN is merely an implementation that is transparent to the Operating System.

In order to allow students to control the instructor's mouse and keyboard over the WLAN, we created two custom pieces of Windows software: a server application (CLICKSERVE) that runs on the instructor's computer and a client program (CLICKCLIENT) that runs on each student laptop. These are Microsoft Foundation Class (MFC), Visual C++, dialog-based applications. When a student starts his laptop computer, the wireless LAN will automatically connect him to the intranet and provide him with an IP address for the TCP/IP protocol. This gives the student all of the facilities required to run nearly every Web-based or Internet-based application. If a student, connected to the network, runs our CLICKCLIENT application from his laptop, he can (with permission from the instructor) gain control of the instructor computer's cursor and keyboard, just as though they were those attached to the student's laptop. This type of interaction allows easy synchronization of students' questions to presentation material on the main screen by allowing the students to easily point to the area in question. It also enables students to change between slides and interact with software running on the instructor's computer. Finally it allows students to enter text and annotate diagrams on the instructor's computer. All of these possibilities can do much to increase interactivity and enhance learning.

When the instructor starts CLICKSERVE, he is presented with a minimizable window that gives him the option of accepting or denying student connections. This gives the instructor the flexibility of being able to decide when student intervention is most convenient or warranted.

When the student runs CLICKCLIENT, he is confronted with "Configure", "Connect", "Page Instructor", "Control Screen", and "Exit" buttons. Pressing the "Configure" button allows the student to enter the IP address and data port of the server. After pressing the "Connect" button, the student's computer is connected to the instructor's over the network, its screen is blanked, and any subsequent mouse or keyboard actions he takes will be reflected on the instructor's computer screen. (The instructor also has the capability of controlling his own mouse cursor; mouse/keyboard messages from the instructor's system take precedence over messages coming over the network from a student computer.) Pressing the <ESC> key on the student's laptop disconnects him (thereby disabling him from controlling the system mouse and keyboard) and returns his screen to normal. While connected, CLICKCLIENT also allows the student to raise his or her hand "electronically". Pressing the <F1> "page" key on the laptop (or the "Page Instructor" button initially) causes an audio file to be played on the server that notifies the instructor that a student has a question.

CLICKCLIENT and CLICKSERVE communicate through standard TCP/IP sockets. Whenever the user of the student computer performs any mouse or keyboard actions, CLICKCLIENT sends the appropriate Windows message and data parameters over the socket connection to CLICKSERVE.

Some of the mouse messages of interest are:

```
WM_*BUTTONDOWN     // mouse button  pressed
                   // (* = L, R, or M)
WM_*BUTTONUP       // mouse button released
                   // (* = L, R, or M)
WM_*BUTTONDBLCLK // button double clicked
                   // (* = L, R, or M)
WM_MOUSEMOVE       // user moved mouse
```

The data that is sent over the network connection is identical to the information that is packaged in a normal Windows mouse event message: namely the x and y coordinates of the mouse cursor and codes indicating what action occurred as well as the status of the mouse buttons and certain keyboard keys (Shift, Ctrl, Alt) when the event occurred.

The keyboard events of interest are:

WM_KEYDOWN // User pressed a keyboard key
WM_KEYUP // User released a keyboard key

The data that is sent over the network for these kinds of messages consists of the virtual key code that identifies the key pressed or released and other information related to the keyboard event that triggered the message.

Finally if the student presses his <F1> key (thereby attempting to "page" the instructor), the message is received as:

WM_USER+100 // User pressed his "page" key

The response of CLICKSERVE is to play a wave file on the instructor's computer. The connection and communication between the student and instructor computers as well as the salient features of the CLICKCLIENT and CLICKSERVE programs are illustrated in Figure 9.

By having the client send the information blindly and pass it on to Windows functions in the server that accept these parameters, the CLICKCLIENT and CLICKSERVE programs are general enough to sustain significant changes in the way Windows processes and packages mouse and keyboard events.

CONCLUSIONS

At Binghamton University several people have tried using our Interactive Learning Wall system. In all cases the learning curve required to become productive was very slight. The time needed to become proficient was spent mostly in becoming familiar with how the laser pointer's "click" sequence works. After becoming comfortable with that, the interface proved to be quite intuitive. On the basis of our initial tests, we feel that this type of device could be used effectively in large classroom environments at many other post-secondary institutions.

FUTURE WORK

We are in the process of attempting to add some of the following features to our Interactive Learning Wall: (1) the ability to do screen transferal over the network connection; (2) a system of user names and passwords for the network communication component; (3) a voice command

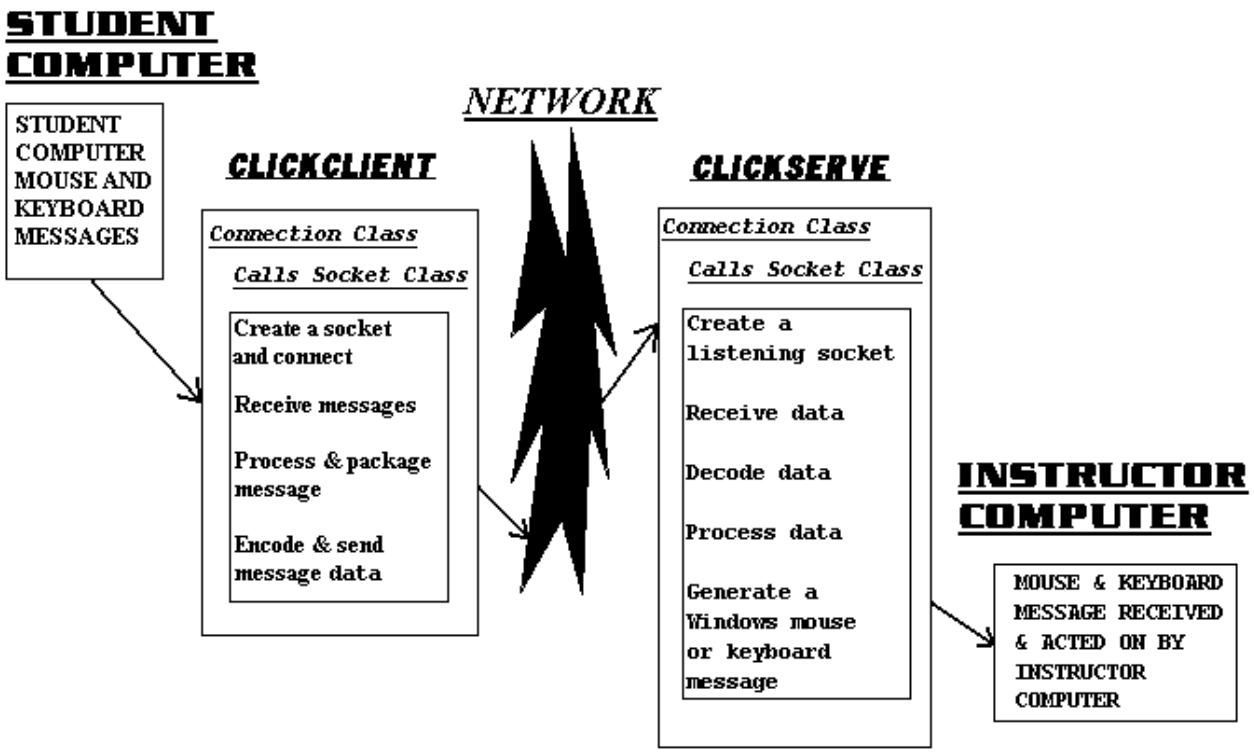

Figure 9. CLICKCLIENT and CLICKSERVE Controlling Windows Messages Over the WLAN Connection.

recognition and control system.

Currently when a student takes control of the instructor's computer, his own laptop's screen is blanked out. This makes it somewhat difficult to use his mouse and keyboard to manipulate objects on the screen—especially if the student is far away from the screen. If the main computer's screen could be downloaded to the student laptop, such manipulations would be much easier to perform. In addition, the student could then easily "copy" and "paste" information from the instructor's screen (which now appears on the student's screen), thereby providing a simple mechanism to get information from the instructor's computer to the student's laptop.

We already have designed and implemented a prototype username/password system. The advantages of such a system are obvious. Only authorized students would be permitted to connect to the instructor's machine, and the instructor would have more control over the situation.

Voice control would be a natural complement to the laser pointer mouse emulation component. In addition to using the laser pen to control his computer's mouse, the instructor could easily wear a wireless microphone as he roams around the classroom and, with it, give voice commands such as "Next Slide" to his computer. Initial attempts to use inexpensive voice recognition software such as IBM's ViaVoice to achieve this kind of control of our system have shown some promise, but there are many problems involved.

ACKNOWLEDGMENTS

We would like to acknowledge support from the Airforce Office of Scientific Research and technical assistance from the Displays and Intelligent Interfaces Group at Rome Airforce Laboratory.

REFERENCES

1. Advanced Displays and Intelligent Interfaces Group Web Site, www.rl.af.mil/programs/ADII/adii_dw.html.

2 Moore, J.A., "Interactive Learning Wall," Binghamton University Masters Thesis, 1998.