COMMWINDOW

By: Larry Masinter, Stan Lanning, Nick Briggs, Richard Burton (Masinter.pa@Xerox.com, Lanning.PA@Xerox.COM, Briggs.PA@Xerox.com, Burton.pa@Xerox.com)

Uses: COURIERSERVE

This document last edited on 2-Apr-87 17:49:35

INTRODUCTION

This is a demonstration of communication capabilities. The COMMWINDOW module implements a "remote window" capability, where one user can watch (with slow update) a region of another users screen.

Both participants need to have COMMWINDOW loaded.

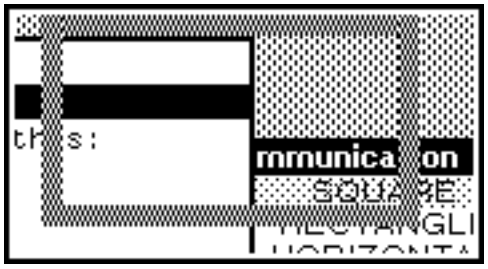
To show someone else a piece of your screen, call

(SEND-BITS *partner &OPTIONAL frame*) [Function]

The partner is the name of the machine you want to talk to. (The watcher has to be registered in the clearinghouse database; this is a NS protocol name.)

frame, if supplied, is a screen region to show. If it is omitted, send-bits will prompt the (sender) for a frame of the screen.

The sender has complete control over the frame. The frame appears as a gray frame around the area shown, like this:



(The frame consists really of 4 thin windows, so that you can really type/button inside it.)

The frame can be moved by left buttoning anywhere on it. Right buttoning on it makes it go away temporarily.

(Reshaping the frame can be accomplished by left buttoning the frame while the shift key is depressed. This doesn't reshape the remote window currently, so this isn't very useful.)

Since this is an experiment, there are a couple of parameters you can vary. The function (mode-menu) will bring up a window that lets you control any and all of these parameters.

The first is the shape of the areas sent in each packet.

Currently available shapes are:

square: a square of bits as big as will fit in a single packet

rectangle: 4 times wider than it is high.

horizontal: as wide as the frame, and then as high can be

vertical: as high as the frame and as wide as can be

h3: this is a funny mode, where the update is in horizontal chunks, but spread out.

PARAMETERS

The parameter \ETHERLIGHTNING lets you tell the low-level ethernet code to randomly drop packets. This parameter can be used to study the effects of noisy communication lines on transmision protocol.

Another parameter gives you some local control over the region within the frame that is updated. There are three available values:

Sender: Update near the sender's cursor, when it has moved, ASAP.

Viewer: Update near the viewer's cursor, when it has moved, ASAP.

NIL: don't do anything special.

Another parameter lets lets you prune out sending packets that update regions of the display that have not changed. The tradeoff here between the cost of sending the packets vs testing a region to see if it has changed.

A recent addition was the following part of the protocol: when the mouse on the sender is moving, it will send a square around the mouse interleaved with any other transmission ongoing. That means you can make sure a remote area is being updated by wiggling the mouse.

The sender's mouse is also tracked in the remote viewers viewpoint (the cursor shape is currently not tracked, however). The mouse coordinates are sent every packet, so that mouse position always updates.

The viewer has a limited option to place a pointer back on the sender's screen: if the viewer holds the shift key down while the viewer's mouse is in the view point window, a little pointer () will appear in the senders window and track the viewers cursor. (The pointer is in fact a little icon window.) When the viewer releases the shift key, the pointer icon disappears.

CAUTIONS:

Don't move the frame off the screen. It will just signal an error.

The "don't change unsent tiles" parameter doesn't seem to work.