PAGEHOLD

A Facility to Control Window Scrolling

author: JonL White
files: [Eris]<Lisp>...Library>PAGEHOLD.DCOM
documentation: [Eris]<Lisp>...Library>PAGEHOLD.TEDIT (and .PRESS)
modified: May 21, Oct 21, and Dec 1, 1984 by JonL

The **PAGEHOLD** facility redefines **PAGEFULLFN** to achieve
-- No video inversion of the TTY window when being "held";
-- Release is effected by one of several means:
{1} depressing and releasing either SHIFT key;
{2} mousing the pop-up "button" which announces the "hold";
{3} the time period of "holding" has exceeded a timeout;
{4} typing a character during "holding" (which is discarded, but see {5} below);
{5} there is typeahead in the window's input buffer, and it is the TtyDisplayStream for the current TTY process;
-- Continued "holding" while either shift key is being depressed;
-- Override of "holding" action if CTRL key (but neither SHIFT key) is held down;
-- Gagging the "timeout" action of a "hold" indefinitely by depressing the CTRL key while holding down a SHIFT; the "hold" must then be released by one of the other means.
-- A menu of options is obtainable from the pop-up, interactive "button" signalling the "hold";

Automatic, "time out" Releases

One of the primary reasons for the existence of this facility is so that printout to a TTY window will not hang indefinitely when one "page" has filled up. The default release time is in the globalvar **PAGE.WAIT.SECONDS**, which comes initialized to 20 seconds; a value of 0 will cause immediate release (unless a SHIFT key is being depressed). If a window being "held" has a **PAGE.WAIT.SECONDS** property, then that value is used instead of the overall default.

However, if **PAGE.WAIT.SECONDS** is set to **STOP**, then the "hold" will not be released by any automatic timeout, nor will it be sensitive to the SHIFT or CTRL key actions; this mode most closely approximates the current Interlisp-D design, except that a pop-up "button" signals the "hold" rather than a video inversion (mousing the button will, nevertheless, still effect a release). The message

"Scrolling Stopped"

will appear in the "button" rather than one of the several "holding" messages.

Finally, a "hold" may be put into "indefinite hold", which is similar to the **STOP** mode just described, by depressing CTRL while also holding down either SHIFT. At least one of the three "button" styles described below will visibly indicate this indefinite "hold" mode; and release may be effected in any of the usual ways, including depress/release of SHIFT, except that there will be no automatic "time out" release.

The Pop-up "Buttons"

A secondary reason for this facility is to have a pop-up "button", which will interactively signal the user of a "holding" condition on a particular window. Not only does this design permit a different button for each window, but it also permits the selective release of "holding" by mousing the button (whereas keyboard action would be interpreted by all the page "holders" as a release signal). There are three styles of "buttons" -- **WINKING**, **FLASHING**, and **NIL** -- and the selection is determined by the value of the globalvar **PAGE.WAIT.ACTIVITY**, which comes initialized to **WINKING**. If a window has a **PAGE.WAIT.ACTIVITY** property, then that value is used instead of the overall default.

A **WINKING** "button" is a fairly hefty pad -- approximately 1/2" by 2 1/2" -- which pops up just over the right side of the window's title bar; it will alternately print and clear two short holding messages: one in

the upper half of the "button" and one in the lower half. A **FLASHING** "button" is about the same width, but half the height, and will alternately print the two holding messages. A **NIL** "button" merely shows the message "Release SHIFT for more".

LEFT-mousing any "button" will effect a release of the "hold"; MIDDLE-mousing the "button" will bring up a menu offering, one of whose options is for simple, immediate release. Other menu options permit conversion of the "hold" to indefinite "hold" or to **STOP** mode; additionally, five options are offered for setting the window's specific **PAGE.WAIT.SECONDS** property.

The **WINKING** "button" will actually have a different pattern of activity when the "hold" is placed into indefinite hold mode, but the other button styles do not visibly distinguish this state. If there isn't room to place the "button" down over the right side of the title bar (because, for example, the window is too close to the screen top), then it will be placed over another corner of the window.

Keyboard Input and Typeahead

Consistent with Interlisp-D's current action, there will be no holding on a window which is the TtyDisplayStream for the current TTY Process *and for which there is typeahead in that process's TTY input buffer*. But this action can be overridden by setting **PAGE.WAIT.IGNORETYPEAHEAD** to a non-**NIL** value; additionally, when it is set to a non-**NIL** value, then no keyboard input will release the "hold", and none will be discarded (note that depressing the SHIFT and/or the CTRL keys does not generate input). This is especially for those who dislike not knowing whether a keystroke will be "eaten" by the **PAGEFULLFN**. Otherwise, if the TTY input buffer is empty when the "hold" begins, then any character typed will release the hold, and the type character will be discarded.