X-SB-MOUSE

X11 Mouse Support For GNU Emacs

Sullivan Beck
beck@qtp.ufl.edu

$Date: 1992/06/27 00:00:00 $
$Revision: 1.6 $

1 Introduction to x-sb-mouse

This file documents x-sb-mouse (version 1.6), an extension to standard emacs with extended mouse support in X windows.

X-sb-mouse is an enhanced version of x-mouse containing all of the features of alt-mouse and x-mouse-drag. It is fully compatible with x-mouse which means that any package that runs on top of x-mouse will also run on top of x-sb-mouse. It is known to work with 18.58, but should work with most earlier versions as well.

X-sb-mouse was designed to increase the poser of the mouse when GNU emacs is running under X11. Very basic X11 mouse support is built into the standard distribution (see x-mouse.el in the standard distribution), but this package contains only the most rudimentary functions.

In order to increase the amount of mouse functionality available, two different things have been done in the past. First, several different packages have been written to essentially replace x-mouse while including greater functionality. Two of these are alt-mouse (by Eirik Fuller) and x-mouse-drag (by Peter Moore). Both are more powerful then x-mouse, but still have rather limited power. X-sb-mouse combines all of the functionality of these two packages (plus a great deal more) into one package.

The other thing that has been done is to write packages which run on top of x-mouse. These packages use the simple functions provided by x-mouse to build more complicated functions. Examples include hyperbole (a very nice HyperText like system written by Bob Weiner) which has been tested quite thoroughly and Dave Gillespie's info replacement. In order to take advantage of these packages, x-sb-mouse was designed to be completely compatible with x-mouse. Both hyperbole and info work fine with my package. There are several others, but I have not had time to test them extensively.

If you have any suggestions on how to improve x-sb-mouse, or discover any bugs, please email me at:

`beck@qtp.ufl.edu`

1.1 Mouse Terminology

Although I THINK most of the terms used in this manual will be clear to anyone who has ever touched a mouse before, a quick list won't hurt.

press
release
: These are the two mouse events recognized by emacs. When a mouse button is pressed, three pieces of information are available: which button was pressed (or released), which keyboard modifiers were used (none, shift, control, and meta are available as well as any combination or two or all three of the modifiers), and the position of the mouse in the window.

position
: The position of a mouse event is give as an x and y coordinate, the x coordinate being the character column (usually in the range of 0 to 79 for an 80 column wide window) and the y begin the line of text).

  If a mouse release occurs outside of the emacs window, the position given is the position of the character closest to the event. So, if the release occurs to the

left of the emacs window, the x position will be 0 and the y position will be the row directly over from the mouse.

click
drag
: If a mouse press and release event occurred in exactly the same location, it is called a click. A drag means that the release occurred in a different location.

point
mark
mouse-point
: The point is just the location of the cursor in a particular buffer. The mark is an invisible position in the buffer which is set by a number of commands. These should not be confused with the mouse-point which is the point under the mouse pointer.

region
drag-region
: A region is defined in emacs as the text between the point and the mark. This should not be confused with the drag-region. When a mouse drag occurs, the text between where the button was pressed and where it was released is called the drag-region. It is meaningless if the drag does not occur within a single buffer.

kill-ring
x-cut-buffer
: In emacs, when you kill or copy text, it goes into the kill-ring. When you kill or copy text in X11, it goes into the x-cut-buffer. These two are treated quite differently, but emacs can access both of them. Using the mouse, text can be stored in either the x-cut-buffer or the kill-ring or both.

1.2 Window Terminology

Each position in an emacs window is one of five types:

mode
: The mode (short for mode line) is the bottom row of each window.

border
: When two windows are split side-by-side (as with the "C-x 5" command), there is a one column border between the two filled with the character "|".

inter
: An inter (short for intersection) is the single position where a border intersects a mode line.

mini
: Mini (short for minibuffer) are the positions in the minibuffer (which is NOT limited to a height of a single row).

window
: Everything else in the screen is taken up by the actual windows where text is or can be. These positions are simply referred to as type "window".

1.3 X-mouse Features

The x-mouse package included in the standard distribution has only a small set of functions available.

A small set of window oriented functions are available. You can select a window, split it, and delete all other windows. Clicking in a window also allows you to move the point to where the mouse is or set the mark there.

You are also able to copy or kill text into the x-cut-buffer and paste the contents of the x-cut-buffer into your emacs buffer.

For those who can use popup menus, there is also a function to pop up a menu of all the buffers available. Not everyone can use popup menus though. See see Section 5.2 [Popup Menus], page 16, for more information.

All of these functions are also available in x-sb-mouse.

1.4 Alt-mouse Features

Alt mouse has most of the functionality of x-mouse (it does not have the built in popup menus) with some extra features. It has the ability to resize a window by dragging the mode line and it has a partial implementation of dragging the border. You can also scroll the window using the mode line.

Unfortunately, alt-mouse is not compatible with x-mouse.

1.5 X-mouse-drag Features

X-mouse-drag is a great improvement over x-mouse. It appears to be completely compatible with x-mouse, but adds a number of very nice features. First, it allows you to use keyboard modifiers (shift, control, meta) with your mouse. It also distinguishes between clicks and drags.

In addition to all of the functions available in x-mouse, it has some nice scrolling commands (scroll mouse point to the top of the window and similar functions), an indent region command, the ability to copy or kill drag regions to the x-cut-buffer, and a few other fairly simple functions.

Not all of the simple functions (such as indent region) have been bound to mouse keys in x-sb-mouse, but the way it is set up, such functions are very easily added by the user in their .emacs file. See see Chapter 2 [Installation and Customization], page 5, for examples.

1.6 Additional Features

X-sb-mouse combines all of the functionality of the other packages discussed, plus some.

First of all, it distinguishes between the different types of positions described in see Section 1.2 [Window Terminology], page 2. Thus, it is possible to click in the mode line and execute one function, click in the border and execute something else, and then click in the minibuffer and run a third function. Also, it is possible to redefine ALL 120 valid click events (3 buttons, 5 different types of window positions, 8 different keyboard modifier combinations) simply (see see Chapter 2 [Installation and Customization], page 5, for detail). Practically any valid elisp function can be bound to these click events.

In addition, there are 24 different drag events that can be redefined. Any drag that starts and stops in the same window (3 buttons with any of the 8 keyboard modifier combinations) can be used to define a region of text. Any elisp function that operates on a region can be bound to these drag events.

Also, every one of these events is mode specific. That is, the mouse event may perform one operation when in text-mode and another when in c-mode.

In the future, other valid user definable drag events may be defined (such as dragging between two windows, dragging inside a mode line, etc.) may be added.

Other types of drag events are also recognized. Drags starting in the mode, inter, or border and ending in an adjacent window (including the minibuffer) can be used to resize windows with only two restrictions. First, the minibuffer must be at least one row high but may be larger if desired. Second, if you ever try to shrink a window narrower than the minimum allowed width or shorter than the minimum allowed height, the window disappears completely.

A large number of cut and paste commands are available. X-sb-mouse allows you to copy/kill text to both the x-cut-buffer and the emacs kill- ring simultaneously or to the emacs kill ring alone. You can also copy or kill text from one location directly to another without ever storing it in either the kill-ring or the x-cut-buffer.

A number of simple functions have also been added such as copying the kill-ring to the x-cut-buffer, scrolling a window without selecting it, append a drag-region to the kill-ring and x-cut-buffer, and others.

Some rectangle commands have also been added to the default mouse bindings. This was done simply by assigning the rectangle commands to a drag event. Since the drag event naturally works with regions of text, no special modification had to be made to the package in order to add these bindings.

A more complete list of the functions available can be found in see Section 2.3 [Functions], page 8.

2 Installation and Customization

This section gives the details on how to test x-sb-mouse out just once (who knows, you may decide you don't like it.... but I doubt it :-) or how to install permanently and how to customize your personal mouse bindings.

2.1 Installation

Installing x-sb-mouse is very easy with the Makefile included. It requires that you have the makeinfo program. If you do not have it, you can get it by anonymous ftp from prep.ai.mit.edu:/pub/gnu/. It's in the file texinfo-2.15.tar.Z (or whatever the most recent version happens to be). This is only necessary if you want to use the info file. The actual mouse package can be compiled and installed without the makeinfo program.

Edit the Makefile and decide where you want to install the program and set ELISPDIR to that directory. If you are going to create an info file, set the INFODIR variable to the directory where it should be installed.

Then just type one of the following:

```
make install       # to make and install everything
make install.src   # to make and install just the program
make install.info  # to make and install just the info file
make               # to make (but not install) everything
make src           # to make (but not install) just the program
make info          # to make (but not install) just the info file
```

To try this out once (but NOT have it load every time you start up emacs), just load it with the 'M-x load-file' command to load the file x-sb-mouse.elc.

To have it automatically start up each time you use emacs, just add the following to your .emacs file:

```
(setq term-setup-hook '(lambda () (load-library "x-sb-mouse")))
```

This assumes that ELISPDIR is in the default load path (i.e. where all of the other .el and .elc files are kept). If they are stored somewhere else, you will need to add two lines to your .emacs file:

```
(setq x-mouse-dir "DIR")
(setq term-setup-hook
      '(lambda () (load "DIR/x-sb-mouse")))
```

where you must replace DIR in the above lines with whatever you defined ELISPDIR to be in the Makefile. For example, I keep a copy of the package in my "~/lisp/xsbm/" directory that I am working on and that's the one I want loaded, so in my .emacs file, I have:

```
(setq x-mouse-dir "~/lisp/xsbm")
(setq term-setup-hook
      '(lambda () (load "~/lisp/xsbm/x-sb-mouse")))
```

If you use any packages that require x-mouse (such as hyperbole or gdbsrc), they should be loaded AFTER x-sb-mouse. They will probably redefine some of the default key bindings. The function x-mouse-init-mouse-map will redefine these keys to whatever functions I have defined or that you have defined in your .emacs file. The problem is that this will remove the bindings used by the other packages. Hyperbole does NOT have this problem. It allows

you to toggle between it's bindings and your personal mouse bindings. I am planning on adding a simple toggle function to x-sb-mouse which will hopefully solve this problem.

2.2 Customization

X-sb-mouse was designed to easily customizable. It should be very easy to bind nearly any emacs function to the different buttons.

All mouse functions are bound to variables (rather than the mouse-map). The reason for this is simple. There is no SINGLE function that a mouse release does. It can cause the execution of lots of different functions depending on where it occurs, where it is in relation to where the press occurred, etc. As a simple example, you would want a button click to do one thing if it happened in the window and another thing if it happened in the mode line and yet you would like to be able to change each independently of the other. The way I chose to do this is to use a set of variables bound to the desired function.

The variable names are very easy to remember. An example of two of the variables are:

```
x-mouse-cm1-mode-click
x-mouse-3-window-drag
```

All of the variables start with x-mouse-. Following that are the one character description of any keyboard modifiers (c for control, m for meta, s for shift). The characters MUST come in alphabetical order, so the variable

```
x-mouse-mc1-mode-click
```

is not a valid name. Following the keyboard modifiers is the number of the button (1, 2, or 3). Then comes where an event occurred (one of the following strings: window, mode, border, inter, mini) and the type of event (either click or drag). The ONLY recognized drag events currently are those occurring in the window and some special ones which resize the window. The latter are not user redefinable, so the only drag variables available all end in -window-drag.

There are a large number of functions appropriate for binding to either click or drag events included in this package. They are described in see Section 2.3 [Functions], page 8. They are very easy to bind. All of them can be bound to either a click or drag event (although most of them were clearly meant to be one OR the other but not both).

One of the functions is x-mouse-select which selects the window clicked on. This can be bound to a mode line click, border click, inter click, or window click. You might want to bind it as the default operation that occurs, but you may want to do something else if you click in the mode line of a window in Buffer-menu-mode (say delete the window). To do this, as well as make clicking button 2 in the window when in text mode recenter that window, add the following to your .emacs file:

```
(setq x-mouse-bind-hook
  '(lambda ()
    (x-mouse-define-key "x-mouse-c1-mode-click" t
      'default  'x-mouse-select
      'Buffer-menu-mode  '(lambda () (delete-window nil)))
    (x-mouse-define-key "x-mouse-2-window-click" t
      'text-mode 'recenter)))
```

You can also assign drag events. One function named x-mouse-copy-text will copy the the text in the drag region to the X cut buffer. It may also copy it to the emacs kill ring at the same time (see x-mouse-duplicate-cut in see Section 2.4 [Variables], page 12). Alternately, it can be bound to a click event in which case it will copy the text between the point and the click. The other drag functions are similar affecting the drag-region or the region between the point and the click. All click functions can be bound to drag events and only the position where the mouse was released is important.

You can also bind regular emacs functions to mouse events. Almost any valid emacs expression can be used. For example, if you wish to bind the indent-region to a drag event as the default command, add the following to your .emacs file:

```
(setq x-mouse-bind-hook
  '(lambda ()
    (x-mouse-define-key "x-mouse-cs2-window-drag" t
      'default '(lambda ()
                  (indent-region x-mouse-pos-d x-mouse-pos-u t)))))
```

Several variables are available for use when you write your own bindings:

x-mouse-point-0
x-mouse-win-0
: These two variables store the window that was active at the time of the button press and the point in that window.

x-mouse-pos-d
x-mouse-pos-u
x-mouse-click
: These store the position of the cursor (relative to the upper left hand corner of the emacs window) where the button press or release occurred. X-mouse-click is t if both positions are the same and nil if they are different.

x-mouse-win-d
x-mouse-win-u
: The window in which the mouse press occurred and the window in which the release occurred are stored in these two variables.

x-mouse-mode-0
x-mouse-mode-d
x-mouse-mode-u
: The major mode of the current buffer, the buffer in the window where the mouse was pressed, and the buffer in the window where the mouse was released.

x-mouse-coords-d
x-mouse-coords-u
: These store the position of the cursor (relative to the upper left hand corner of the window they were in). They are nil if the event occurred in the mode line or border.

x-mouse-point-d
x-mouse-point-u
: The point under the mouse event are stored in these two variables if the event occurred in a window.

x-mouse-type-d
x-mouse-type-u
: The type of window location (window, mode, border, inter, or mini) is stored in these two variables.

2.3 Functions

This section describes the functions which are currently available for binding. I have tried to provide a large number of functions (though not all are bound to buttons by default). If you have suggestions for additional functions, be sure and send them to me.

2.3.1 Misc Functions

x-mouse-ignore
x–mouse-ignore
: These functions do nothing. The reason for having both is because of a bizarre behavior which occurs when hyperbole is loaded. For some reason, when hyperbole loads, it sets the function x-mouse-ignore to it's value in x-mouse.el, even if x-sb-mouse.el has been loaded. I do not know why it does this. All other functions are left with the definition provided by x-sb-mouse. The only difference between these two is that x-mouse-ignore requires and argument while x–mouse-ignore does not. Some day I will sit down and fix this.

x-mouse-set-mark
: This will store the position under the mouse point as the mark.

x-mouse-call-last-kbd-macro
: This will move the cursor to the mouse point and call the last keyboard macro. Very nice if you use macros.

x-mouse-execute-extended-command
: This is the mouse equivalent of M-x. It is bound to csm3-window-click and csm3-window-drag (and although ALL other bindings can be redefined, you should not rebind these two). This will allow you to execute any click or drag even by name.

2.3.2 Scrolling and Motion

x-mouse-set-point
: This moves the point to the location of the button release.

x-mouse-scroll-down
x-mouse-scroll-up
: These scroll the buffer up and down without selecting the window.

x-mouse-scroll-to-top
x-mouse-scroll-to-center
x-mouse-scroll-to-bottom
: This will move the line under the mouse to the top, center, or bottom of the window respectively.

x-mouse-scroll-to-proportion
: This can be bound to either a window, mode, or border event. If it is bound to the mode line, clicking on the leftmost part of the mode line will go to the top of the buffer. Clicking on the rightmost position will go to the bottom of the buffer. Clicking 25% of the way from the left will jump to approximately 25% of the way from the top of the buffer. The action is similar in the window and the border where clicking at the top of the window goes to the top of the buffer, etc.

x-mouse-scroll-line
: Use this function to drag a line around. The buffer will be scrolled as necessary to make the line that was under the mouse at the press event move to where the mouse release occurs.

2.3.3 Window Functions

x-mouse-select
: This function selects the appropriate window.

x-mouse-select-and-split
: This selects the window and splits it vertically.

x-mouse-keep-one-window
: This selects the window under the mouse and deletes all others.

x-mouse-split-vertically
x-mouse-split-horizontally
: These split the window without selecting it. The first should either be bound to a window or border function. It splits the window vertically placing the mode line under the mouse. The second should be bound to a mode or window operation. It will split the window horizontally placing the border under the mouse.

x-mouse-delete-this-window
: This deletes the window under the mouse.

2.3.4 Copying and Cutting Text

x-cut-text
x-cut-and-wipe-text
x-mouse-copy-text
x-mouse-cut-text
: The first two are provided solely to provide compatibility with x-mouse.el. These will copy or cut (i.e. delete it from the buffer) the drag region and store it in the X cut buffer. It will also store it in the emacs kill ring if desired.

x-mouse-append-drag
: This appends the drag region to the text stored in the X cut buffer. It will also append it to the last text in the emacs kill ring.

x-paste-text
x-mouse-paste-text
x-mouse-paste-there
: The first is provided solely to provide compatibility with x-mouse.el. These insert the text stored in the X cut buffer in the window the button release occurred in. x-mouse-paste-text moves the point to where the mouse is and then inserts the text. x-mouse-paste-there inserts the text at the current point (since this function does not use the position of the mouse event for anything, it can be bound to a mode or border event if desired).

x-mouse-copy-text-to-point
x-mouse-cut-text-to-point
: These either copy or cut the text in the drag region and insert it into the text in the currently active window (even if it is different then the one the mouse events occur in) at the point.

x-mouse-yank-here
x-mouse-yank-there
: These insert the last element in the emacs kill ring at either the mouse point or the current point respectively.

x-mouse-copy-kill-to-x
: This copies the last element in the emacs kill ring to the X cut buffer.

x-mouse-copy-bol-to-x
x-mouse-copy-line-to-x
x-mouse-copy-eol-to-x
: These copy part or all of the line under the mouse to the X cut buffer and the emacs kill ring if desired. They copy from the beginning of the line to the mouse point, the whole line, and from the mouse point to the end of the line respectively.

x-mouse-copy-rect-to-x
x-mouse-cut-rect-to-x
: This defines a rectangle as the drag region and stores the text in the rectangle in the X cut buffer (and the kill ring). The text is stored as the text in each line of the rectangle separated by newlines.

x-mouse-copy-rect-to-000
x-mouse-cut-rect-to-000
x-mouse-insert-rect-000-here
x-mouse-insert-rect-000
x-mouse-insert-rect-000-there
: The first two functions either copy or cut the rectangle defined by the drag region and store the text in register 000. The latter three functions insert this rectangle into the text at the mouse point (without moving the current point), at the mouse point (and moves the point there), and at the point respectively.

x-mouse-open-rect
: This opens the rectangle defined by the drag region.

x-mouse-copy-thing
x-mouse-cut-thing
x-mouse-copy-thing-to-point
x-mouse-cut-thing-to-point

These functions use the thing package (thing.el) which has been included in this package. Thing allows you to define a region of text. If you click on a brace, bracket, or parenthese, it finds the matchine brace, bracket, or parenthese and matches everything inside the two. Clicking at the end of the line matches the whole line. Clicking on a double quote matches everything up to the next double quote. Clicking anywhere else matches the word you clicked on.

Once the region of text is defined, this region is either copied or cut to the X cut buffer (and the emacs kill ring if appropriate) or is copied/cut directly to the current point.

2.3.5 Popup Menu Commands

x-buffer-menu
x-mouse-buffer-menu

The first is provided solely for compatibility with x-mouse.el. This pops up a menu with all of the buffers currently available. Selecting one of the lines in the menu will switch to that buffer.

x-help
x-mouse-menu-help

This pops up a menu of common emacs help commands.

x-mouse-help

This function will describe all the current mouse bindings. If you don't have popup menus (see see Section 5.2 [Popup Menus], page 16) or if x-mouse-help-to-menu is nil, the help goes to a temporary *Mouse Help* buffer. Otherwise, it goes to a popup menu.

x-mouse-get-file

Pops up a menu of files in the current directory and allows you to select one to edit. See x-mouse-file-ignore-regexp in see Section 2.4 [Variables], page 12.

2.3.6 Support Functions

These functions are not meant to be bound to mouse buttons. Instead, they can be bound to normal emacs keyboard commands to help in using x-sb-mouse.

x-mouse-global-set-key

This function allows you to bind the default mouse event. The default mouse event is the one which is executed when the mouse event occurs unless it is executed in a buffer who's major-mode has a mode specific binding.

x-mouse-local-set-key

This function allows you to bind a mode specific binding to a mouse event. The mode used is major mode of the current buffer unless a C-u was typed first. If the prefix was included, you are prompted for the mode.

x-mouse-define-key
: This function is used to define key events. It is not called interactively but is useful for adding to the .emacs file. The first argument is a string containing the variable for a mouse event (for example 'x-mouse-1-window-click'). The second is a flag (t or nil). If it is t, any previous binding is overrided. If nil, previous bindings are left unchanged, but if no previous binding is present, the new one is added. Following this are a list of modes and bindings. See see Section 2.2 [Customization], page 6, for examples.

x-mouse-global-unset-key
: This can be used to make the default mouse operation do nothing.

x-mouse-local-unset-key
: This actually removes the binding for the current local mode (or the one prompted for if the C-u prefix is used). From then on, whenever the mouse event is run in a buffer with this major-mode, the default binding is used instead.

x-mouse-undefine-key
: Used non-interactively to actually remove bindings from a mouse event. The default binding should never be removed.

x-mouse-describe-event
: This is equivalent to the describe-key-briefly command. It is currently bound to C-h C-m. It will ask you to enter a mouse event and then describe the function that is bound to that event.

2.4 Variables

x-mouse-dir
: This is the location of the x-sb-mouse files. Currently there are only two (plus this info file) but more are anticipated as it grows larger.

x-mouse-blink-cursor
: If this is set to 't, when the mouse button is pressed, the cursor blinks to the location of the press and stays there for one second before going back to its original location. When the button is released, the cursor blinks to the location of the release for one second followed by the location of the press for one second (only if it is a drag event) and then back to the original location.

  Note: Pressing any key or button cancels the blink and executes the new command. The only time this blink is known to cause problems is when used in conjunction with popup menus (see see Chapter 3 [Known Bugs], page 14).

  The default value is nil.

x-mouse-auto-set-mark
: If this is non-nil, every time the point is moved to a new location in the buffer, the old location will be stored as the mark. If it is nil, the location is not saved. The default value is t.

x-mouse-duplicate-cut

If this is non-nil, every time text is saved into the X cut buffer, it is duplicated in the emacs kill ring. The default is t.

x-mouse-init-map

If this is non-nil, all of the mouse map bindings are set up to my default values. The default is t.

Note: If you do NOT set up the mouse map bindings, this package will NOT work! Simply binding a functions to the variables does not do anything unless the mouse map is initialized, the functions stored in the variables are never executed. This is only useful if you want to use x-sb-mouse as a replacement for x-mouse but don't want to use the extended features of x-sb-mouse.

x-mouse-help-to-menu

If this is non-nil, the x-mouse-help function will send it's output to a popup-menu. If it is nil, the output is sent to a *Mouse Help* buffer instead. The advantage of sending it to the buffer is that you can type in another window (or execute mouse commands) while continually referring back to the buffer.

Note: The popup menu has an option to redisplay the help in the help buffer as a one time only command, so you do not ever HAVE to set this variable unless you want the help to always go to the buffer.

x-mouse-describe-only

If this is non-nil, all mouse events will be described (i.e. a message telling what command would be executed appears in the minibuffer), but not actually executed.

x-mouse-file-ignore-regexp

This contains a list of file endings which are ignored when the x-mouse-get-file command is run. Files ending in .o, .elc, etc. are ignored.

3 Known Bugs

There are a couple of known bugs in this package.

If x-mouse-blink-cursor (see see Section 2.4 [Variables], page 12) is t and you want to execute a popup-menu command, if you press the button and immediately release it, when the menu goes away, it does not refresh that part of the screen UNTIL some X event occurs (move the mouse out of the window, type something, etc.). If you hold the button down 1 second (so that the cursor has gone back to it's original location) and then release it, the screen is refreshed when the menu is dismissed. If x-mouse-blink-cursor is nil, the refresh bug is absent.

NOTE: I am not the only one to have seen this bug, but it mysteriously disappeared when we upgraded the operating system here. As a result, I can no longer test this. It also probably means that it's not a bug in the lisp code. The upgrade was from SunOS 4.1.1 to 4.1.2.

Another problem is that when you bring up a buffer menu (or any other menu), clicking inside the menu (in the top or the bottom) dismisses it cleanly, but clicking OUTSIDE the menu to dismiss it sometimes treats this mouse click as an x-sb-mouse command. I have not yet figured out why it does this. Since it does not do it with all the commands, I'm pretty sure it is a bug in my code rather than something to do with the way emacs works. I plan on trying to fix this someday.

I'm sure there are others as well.

To report a bug, please email me at:

`beck@qtp.ufl.edu`

Please include the version number you are using (which will be in one of the top few lines in x-sb-mouse.el).

4 Program Organization

This program has been written to be just as modular and easily modifiable as possible. Anyone may modify it, but if you add something you consider useful, please send me a copy of the change so I can use it to.

This program is divided up into several files. The main program is in x-sb-mouse.el. The functions in this file only need to be changed if you're trying to make changes to how x-sb-mouse actually works. For the most part, the average person shouldn't have to make changes in this file.

X-sb-mouse.el also contains some basic instructions on installation and customization, a description of the newest features available, etc.

Xsbm-funs.el contains miscellaneous functions used by the click, drag, and resize functions. Use these to build other functions. For the most part, you will probably not want to modify existing functions since a large number of other functions depend on their current behavior, but you may find something you want to add to it.

Xsbm-keys.el contains a number of functions useful in binding mouse buttons to events. It also contains all the default mouse bindings.

Xsbm-userfuns.el contains all the currently defined clicks and drags. If you add other click or drag functions, they should go here. Also, if you do add something, please mail me a copy so that I can include it in my copy. I will gladly accept suggestions for other functions to put here.

The file thing.el is the package written by Joshua Guttman. It is unmodified except that the final two functions (x-mouse-copy-thing and x-mouse-cut-thing) have been commented out and rewritten by me. My versions are included in xsbm-userfuns.el.

5 Frequently Asked Questions

These are some common question I get.

5.1 Highlighting

How do I make x-sb-mouse highlight functions?

The problem is that emacs 18 does not have any real good way to do highlighting. Version 19 will, but until then, there is really only one way to do it and that involves applying an unofficial patch to the emacs source and recompiling it. Since the patch is unofficial, I have not chosen to support it yet. I may in the future, but I haven't yet decided whether or not to. I dislike applying unofficial patches to my emacs source since it tends to make updating it tedious.

I'm sorry for this rather important feature being absent. I miss it too.

5.2 Popup Menus

Popup menus: Popup menus only work if emacs was compiled with HAVE_X_MENUS defined in the config.h file. This does not work on all computers, but is part of the standard source code so I use it.

6 History and Acknowledgements

1.5 6/92 On a mouse release, the cursor moves to the release position for 1 second followed by the press position for 1 second (if the two are different). If you do something else in those 2 seconds, the jumping around is cancelled and your new command executed. Thanks to Mike Gunter. Also, added the x-mouse-blink-cursor variable. Thanks to Massimo and Stephen Gabe.

Fixed a problem that x-mouse-help couldn't deal with undocumented functions. Thanks to Mark Wright for the fix. Also now handles lambda function documentation. Thanks to Vasco Lopes Paulo.

Rearranged some of the mouse bindings. I won't do this again but I wanted to organize them a little to make them easier to remember. Use x-mouse-help to find out what's changed. Also renamed some of the functions to make the names follow the same convention as the others. I will not rename functions again.

Modified the mini-click bindings. Thanks to Vasco Lopes Paulo.

Added x-mouse-scroll-line and x-mouse-call-last-kbd-macro. If you use macros, check out this second one. Thanks to Vasco Lopes Paulo for the code.

Added (defvar x-mouse-press nil) so that if the first mouse press made was in some other package, x-mouse-release does not report an error. Thanks to Stephen Tweedie.

Moved the user functions to a separate file.

Added x-mouse-execute-extended-command. It is bound to cms3-window-click and cms3-window-drag. This will allow you to execute any click or drag command by name. Thanks to Don Comeau.

x-buffer-menu now executes buffer-menu if popup-menus aren't available. Thanks to Mats Lidell and Ric Claus.

Buffer menu fields are now lined up nicely. Thanks to Chris.

x-mouse-scroll-to-proportion will now go all the way to the bottom rather than only most of the way to the bottom. I.e. clicking on the bottom line of a 10 line buffer will now go to 100% of the way through the file rather then only 90% of the way. Thanks to Mats Lidell.

Changed all the setq's to defvar's in the bottom section and removed the 'x-sb-mouse-hook. It was stupid not to do it this way to begin with. Also added the provide line. Thanks to Torp Anders.

Added x-mouse-dir variable.

Added the info file. This is only a first draft. Please send me comments/suggestions/corrections.

Added x-mouse-split-vertically and x-mouse-split-horizontally. Thanks to Mats Lidell.

x-mouse-help now handles nil documentation and very long documentation. Thanks to Nitan More. I also added the x-mouse-help-to-menu variable and the option in the help menu to send the output to the *Mouse Help* buffer. Thanks to Chris Moore.

Added x-mouse-copy-thing and x-mouse-cut-thing. Thanks to Nitin More, Dag Wanvik. Thanks to Vasco Lopes Paulo for pointing out thing.el as an easy way to do this. These allow copying/cutting words and sexps.

Added x-mouse-delete-this-window. Thanks to Adam Hudd.

1.4.2 5/92 Fixed a problem for people who don't have x-popup-menu. Thanks to Richard Gonzalez and Joerg-Cyril Hoele.

1.4.1 5/92 Fixed x-mouse-help when output is to "*Mouse Help*" buffer.

1.4 5/92 Added x-mouse-scroll-to-proportion function. Thanks to Shekhar Bapat and Jim Hetrick. Works in the mode line, border, and window.

Added x-mouse-copy-bol-to-x, x-mouse-copy-line-to-x, and x-mouse-copy-eol-to-x for copying lines and parts of lines to the X cut buffer.

Rectangles now work if you start at the bottom right.

Added the x-mouse-init-map variable.

Added x-help from x-mouse.el.

Added x-mouse-help. Suggested by lots of people.

Pushing the mouse moves the point there for one second. Thanks to Alastair Burt.

1.3 5/92 Added a check to see if minibuffer-prompt-width is bound in x-mouse-point. Apparently some version(s) of emacs don't have this built into them or some non-standard interferes with it. Thanks to Dinesh Katiyar.

Fixed a minor bug in x-mouse-cut-text. Thanks to Shekhar Bapat.

Made all drag functions callable also as clicks. Thanks to Shekhar Bapat for the suggestion.

Added copy/kill rectangle to x-cut-buffer.

Added bindings to scroll mouse point to the top/center/bottom of the window.

Fixed a small bug in rectangle commands.

1.2 5/92 Fixed buffer menu so that nothing happens when HAVE_X_MENU was not defined.

Fixed a bug in x-mouse-scroll-{up/down}. Thanks to Rohit Namjoshi and Stephen Gabe for the fix.

Made x-sb-mouse compatible with x-mouse (lots of thanks to Mike Gunter, Hunter Marshall, and Fred Brunner for suggestions and for their help in degugging it).

It now REALLY handles lines with tabs and long lines.

Made setting the mark when the point is set optional (see the x-mouse-auto-set-mark variable). Thanks to Mike Gunter and Mosur Mohan for fixes/suggestions.

Added complex-command stuff to minibuffer clicks.

Added the x-mouse-duplicate-cut variable.

1.1 4/92 It now handles lines with tabs and lines that are longer than the width of the window correctly. This works for windows that are split side by side as well as windows taking up the full width of the screen.

Fixed a bug that made clicking in a window containing the same buffer as the selected window position the buffer in the new window the same as in the current one.

Added rectangle commands to the default bindings.

1.0 3/92 Initial release.

I have tried to acknowledge all people who have contributed something, but occasionally I forget. I apologize if I forgot you. Mail me and I'll correct it.