

Pop Through Mouse Buttons: A Simple Hardware Change and its Software UI Impact

Robert Zeleznik, Timothy Miller and Andrew Forsberg

Brown University

Department of Computer Science

Providence, RI 02912

(401) 863-7653; {bcz,tsm,asf}@cs.brown.edu

ABSTRACT

This paper presents a powerful, yet strikingly simple and novel modification in the design of a conventional computer mouse that enables a user to feel, and the computer to discretely sense, two distinct “clicks” on each mouse button, activated by pressing lightly and pressing firmly to *pop through*. We achieve this double-action effect by converting the standard mouse buttons to pop through tactile pushbuttons that are similar to those widely used as the focus/shutter-release buttons on consumer photographic cameras. We present this mouse design in a prototype stage both to describe the compelling interactions we have already developed and to inspire further investigation. The direct impact of our mouse modification is a third button state for each button that makes possible a family of fast, robust button gestures that are composable with or even preferable to well-known gestures such as double-click. Moreover, the qualitative impact creates a different user experience if the three button states are applied in a cognitively meaningful way; for example, if pressing lightly maps to default behaviors such as invoking a short menu, and then pressing firmly to pop through maps to behaviors presenting more control, such as a long menu. In addition to describing the hardware modification, we present a number of such meaningful guidelines.

KEYWORDS: mouse, double-action, gesture, interaction, click through, buttons, pop through, hardware, input devices, haptics.

INTRODUCTION

Since the advent of 2D GUIs over 30 years ago, the nature of mouse-based computer interaction has continued to evolve via both hardware design changes to the mouse and parallel adaptations in software user interface technologies. Our work continues this line of evolution by introducing both a novel mouse design in which a third button state is added to each button and software techniques that leverage the affordances of this design. By pressing lightly, the user feels a mechanical click corresponding to the button’s first acti-

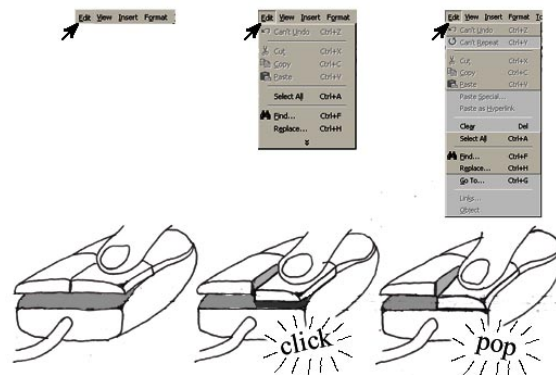


Figure 1: Pressing the left mouse button lightly clicks and in this example invokes a short menu; pressing harder pops through with a second click, in this example invoking a longer menu. (Button displacement exaggerated for clarity.)

vation state, and by pressing harder the user feels a second click, indicating *pop through*, for the second activation state. Although this additional state alone increases the input bandwidth of the mouse, the qualitative impact on the user experience can be more pronounced if the state is used meaningfully because it then enables the computer to sense a form of expression that involves inherently physically coupled actions. Our device is also easy to deploy because it does not require visible changes to a commercial mouse; in addition, legacy applications can be compatible simply by ignoring the additional button state.

Our new device enables interaction drawn from everyday physical behavior. For example, people generally exert more force when they are frustrated, so that pressing firmly to pop through could map to actions more likely to be desired by those frustrated with the response to the lighter pressure level (see Figure 1). As another example, related but different actions could be mapped to the two levels, similarly to a camera shutter button’s focus-freeze and shutter-release functions. Besides physically derived interactions, our device also suggests and makes possible other novel kinds of interaction discussed later.

We hypothesize that using discrete light and firm pressure levels, instead of two distinct buttons, may well be easier

in many circumstances for users to learn and/or remember and perform. The foundation for this belief is an extension of Buxton’s argument [2] that natural phrases in interaction should be delimited by muscular tension and relaxation, for example by holding a button down to initiate a gesture and releasing it to end the gesture. Establishing nested groupings of structures within phrases with the multiple levels of tension afforded by our device is a compatible development of Buxton’s argument.

PREVIOUS WORK

There has been a recent proliferation of mice that increase the flexibility of input through increased numbers of buttons and other additions such as finger-wheels (though not multiple levels of pressure), including four- and five-button offerings [4, 5]. Balakrishnan and Patel [1] presented a mouse with a finger-accessible touchpad on its top in place of buttons and a separate prototype 10-button mouse. However, adding buttons has complications: more dexterity is required, especially beyond a certain number of buttons—in fact, with too many buttons it may not be possible to use the device fully or smoothly, and since the buttons are typically generic their mappings can be hard to remember. Alternatively, augmenting mouse-button inputs with keyboard modifier keys is nearly ubiquitous in graphical user interfaces. Our device, with careful UI design, can essentially double the button functionality of a mouse without significantly complicating the user’s physical activity and without changing the mouse’s exterior at all.

Hinckley and Sinclair [3] added touch sensors to a mouse and to a trackball and discuss a number of ways to use the modified devices to let the computer distinguish the perceptual quality of the user’s intent. Styli that sense continuous levels of pressure have existed for some time now [10]. Our device borrows from both of these approaches by combining pressure and discrete levels of activation and tactile feedback to distinguish different perceptual qualities of the user’s action.

Although available for decades, chording mouse buttons and double-action pressure keys (as used to control autorepeat of space and hyphen keys in certain electric typewriters) are rarely used in modern interfaces. Chording in particular can be problematic because different users employ different hand postures and strategies for activating the buttons. Our pop through mouse buttons are functionally related to chording but present a qualitatively different user experience, both because the button states can be reached with only a single finger and because the user perceives their input as pressure levels rather than coordinated finger activity.

A final method for extending mouse functionality has been to add gestural interaction, including:

- Double- or even triple-clicking.
- Clicking and dragging out a gesture path [8].
- Gesturing on a touchpad on top of the mouse [1].
- Pressing and holding the button then dwelling (not moving) to, say, bring up a menu, rather than pressing and releasing to activate the interface element in the default way (as in Netscape’s [7] forward and back icons).
- Dwelling over an interface element to, say, show a tooltip.

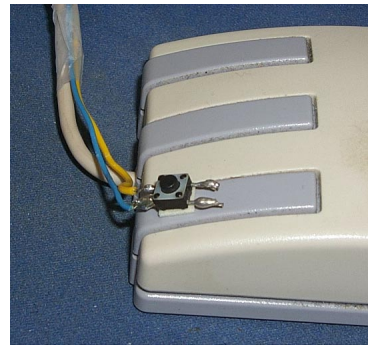


Figure 2: One of our prototypes puts a small pushbutton on an existing mouse button to provide two perceptually distinct activation states for the button.

Not all gestures are created equal: some are hard to learn, and some are slow because of either motor-control issues or intrinsic timing requirements. Even widely used gestures like double-clicking are known trouble spots for beginning and some intermediate users [9]. Double-clicking, for instance, requires the user to get the timing right and not move the mouse too much between clicks. Our pop through buttons make possible additional gestures that can be fast, robust, easily learned, and without timing restrictions. These gestures can also be composed with the other techniques or used as drop-in replacements without significant challenge.

PROTOTYPE HARDWARE AND SOFTWARE

Hardware Design

We have implemented two different prototypes of our pop through design: we mounted a button on top of a conventional mouse button and we retrofitted a mouse with a double-action camera-style button inside.¹ These two designs explore only two points in a large design space with many other tradeoffs possible. Although both prototypes only modified one of the mouse buttons, the other buttons could be similarly modified. Also, even though we discuss only mice, our design can trivially be extended to other button devices like trackballs.

Our first prototype places a small pushbutton on top of a mouse button, as shown in Figure 2. Because of the differing activation forces of the buttons, pressing down on the pushbutton first activates the button native to the mouse and then, with more pressure, pops-through to activate the additional pushbutton. This design is visually explicit and also lets the user press only the original mouse button with no chance of activating the additional one by mistake. This design also has very clearly perceptible and distinct clicks and activation force levels.

Our second prototype places a double-action surface-mount pushbutton inside the mouse so that it is activated instead of the original mouse switch. This button has three positions: off, first contact made, and first and second contact

¹We used a simple, somewhat widely known approach to connect this to an RS232 serial port: we tied DCD to RTS (held high in software) with a 10 k Ω resistor and connected the switch contact between DCD and ground, additionally using CTS for the retrofitted design.

both made, corresponding to three ranges of force applied. This design produces a mouse body that, when assembled, is indistinguishable from a normal mouse body by either vision or touch. The activation forces for this pushbutton are much more subtle than for the first prototype, with no perceptible click for the first contact and a very soft feel, not really a click, for the second, pop through, contact.

Software Techniques

One of our implementations for using the added button state in software was to write a tiny X11 program that maps (or leaves mapped) the lightly pressed button state to the standard left mouse button and maps the pop through state to the right mouse button. (We call this the “map-to-right-button” technique. For technical reasons the mapping program must use the XTest extension to generate the button events correctly.) Under Netscape [7], this enables the user to lightly click on a link to follow it and press firmly to pop through to invoke instead a menu of operations such as opening the link in a new window, saving it, etc. Also, when editing the text in the URL “Location” text entry box, the user can press lightly, drag out a selection, and then, without releasing, pop through with firm pressure to get a menu of editing operations for that selection such as cut, copy, etc. Our device immediately works well with many other X11 applications in which the left mouse button performs a simple, default action and the right mouse button brings up a menu of other choices. These applications work because of how they handle chorded mouse events, for instance, by not ignoring additional button-down events while a button is depressed.

In addition, we modified the Sketch [12] application to read the additional button state directly. In the unmodified application, the user can rubberband an axis-aligned line by pressing the left mouse button and dragging; the user can switch to rubberbanding an unconstrained straight line for the duration of the interaction by using a “tear-off” gesture while dragging. Our modified application simplifies and extends this interaction by mapping light pressure to axis-aligned drawing and firm pressure to unconstrained drawing, making possible transitions between the two while drawing. Also, the Unicam [11] interface maps both panning and zooming camera operations to a single mouse button and uses an indirect directional gesture to distinguish them. Our modification to Unicam maps panning to pressing lightly and zooming to pressing firmly. This mapping, despite remaining nonobvious, is more robust than Unicam’s original gesture because the user need never initially move in an undesired direction, and is more functional because the user can switch between panning and zooming in the same interaction.

DISCUSSION

Hardware Issues

In both of our implementations, it is difficult to release from firm pressure to soft without also releasing the button entirely, and we expect this to be true in general for naïve implementations. With the button-on-button implementation there is also an unusual protrusion (the second button) on top that feels awkward and can disrupt the user’s natural finger positioning. On the other hand, with the retrofitted implementation the light button press can be “dropped” by mistake fairly

easily due to the near absence of positive mechanical feedback, and pop through can easily be triggered accidentally because its additional force is too small. We expect that a properly engineered device could surmount these obstacles and provide easily felt clicks with a specially designed switch inside the mouse. Although we have not explored the ergonomic implications (including repetitive stress injury), we believe that the latitude in the design space and similarity to conventional mice are sufficient that problems can be limited to be tantamount to those of a standard mouse.

Software Issues

Although some applications work well with the general map-to-right-button technique, some may not, and making them take advantage of the additional button state to even this limited degree may require code modification. In particular, the hardware generates two different button events that are necessarily chorded, and chorded button events are generally not considered extensively in user interface toolkits or applications. One possibility might be to have a compatibility mode that, among other things, translates the second level into an up event on the left button followed by a down event on the right button.

On the other hand, applications written specifically to exploit the new characteristics of our technique must consider what to do with the additional button state. The choice in Sketch was straightforward: we replaced an idiomatic “tear-off” gesture by mapping light pressure to the more common operation of drawing an axis-aligned line and firm pressure to drawing the less common unconstrained lines. We present a range of more general potential guiding principles in a later section.

Evaluation

In the course of developing the device, we asked a number of coworkers to try the button-on-button version with Netscape, Sketch, Unicam, and some other ad-hoc tests contrasting with double-clicking. Our informal observations indicate that the device is intuitive, easily learnable, faster, and less error-prone than the techniques it replaced. There were no complaints about fatigue or stress from the brief trials, although further testing is necessary to evaluate this issue fully. In addition, most people were enthusiastic about the device and the simple techniques we implemented, and many people spontaneously proposed other uses. When asked if they would disconnect our new device (in the button-on-button configuration) from their computer in favor of their original mouse, only one out of a dozen or so people said yes. That person found the extra button disrupted natural hand positioning, and found that, while the prototype with the retrofitted button inside the mouse was better in that respect, it was too easy to accidentally release the first pressure level.

PROPOSED UI GUIDELINES

Maximal leverage of the potential of pop through mouse buttons requires the use of coherent, cognitively meaningful UI principles. We present a number of different potential guiding principles in this section. However, some combinations of these guidelines are mutually exclusive or have the potential to be confusing if used together; it is not clear whether it is possible to use different metaphors in different programs

(or even in the same program) without creating more problems than are solved, and this is an area of future work.

Computer Bias vs User Bias

One general principle is to map light pressure to “computer-biased” interaction, where the computer tries to be smart, and firm pressure to “user-biased” interaction, where the user has more control. The idea is that the user would press lightly to get the default, but if not satisfied the user would press harder, as if frustrated, to see more options. This can have a number of different applications, including: invoking a long menu instead of an abbreviated menu (as in many Microsoft applications), overriding default constraints such as drawing unconstrained rather than axis-aligned lines in a drawing program, selecting text by character instead of word, or moving a scrollbar or slider with fine control rather than the default coarse control.

Sequential Operations

Another guiding principle is to associate commonly sequential operations with the two pressure levels. These operations may in fact be perceived as a cognitive unit that previously had to be broken up due to the limitations of conventional mice. For instance, selecting text and then popping through to get an edit menu, selecting an object or action and then bringing up a property box, and letting the user of a spreadsheet select a range of cells and then seamlessly pop up a menu of actions such as “fill down” if the formula entry box is not active, or otherwise a menu of mathematical cell operations to be pasted into the formula entry box.

Two Operations

Although less intuitive, it often can be convenient to map different operations to the light and firm pressure levels, such as undo and redo, or activation of a drawing tool that automatically reverts back to the pointer tool after use and activation of the same tool so that it does not revert, or the Unicam modification of panning and zooming.

Miscellaneous

There are some other simple strategies for using the additional state. Pop through could replace double-clicking, be mapped to get help, or instead be used to abort an action, for example aborting drawing a polyline.

FUTURE WORK

The most important areas of future work are to develop compelling interaction techniques and to develop and extend design guidelines for this device into a coherent framework. It is also important to explore other hardware design variations including mechanisms that make it possible to easily release the second pressure level while maintaining the first, and extending the technique to have three or more distinct activation levels. Finally, adding active force-feedback might lead to some advantages, for example disabling the second activation level where it has no function, or perhaps dynamically changing the force thresholds to make it easier to release the second level without releasing the first.

We are also interested in extending the technique to other input devices, especially those used in immersive environments such as wands [6]. Additionally, we think there is

potential for use on laptops because multiple mouse buttons can be inconvenient to get one’s finger on while manipulating the pointer. Similarly, pop through buttons could be used on PDAs to make possible more easily accessible functionality in the limited space. We imagine, too, that there may be a number of unexpected uses, for instance keyboard keys in which the second pressure level produces a shifted version of the character; this might be useful to certain handicapped people who are unable to chord keys, although it is likely to only ameliorate rather than solve that problem.

ACKNOWLEDGMENTS

This work is supported in part by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization, Adobe, Advanced Network and Services, Alias/Wavefront, Department of Energy, IBM, Intel, Microsoft, National Tele-Immersion Initiative, and Sun Microsystems. Thanks to Dan Keefe for drawing Figure 1.

REFERENCES

1. Ravin Balakrishnan and Pranay Patel. The PadMouse: Facilitating selection and spatial positioning for the non-dominant hand. In *CHI 98*, pages 9–16. ACM SIGCHI, 1998.
2. William Buxton. Chunking and phrasing and the design of human-computer dialogues. In *Information Processing '86 Proceedings of the IFIP 10th World Computer Congress*, pages 475–480, 1986.
3. K. Hinckley and M. Sinclair. Touch-sensing input devices. In *CHI 99*, pages 223–230. ACM SIGCHI, 1999.
4. Logitech. URL <http://www.logitech.com>.
5. Microsoft. IntelliMouse Explorer. URL <http://www.microsoft.com/catalog/display.asp?site=10036&subid=22&pg=2>.
6. Murray Consulting, Incorporated. Wanda. URL <http://home.att.net/~glenmurray>.
7. Netscape. URL <http://www.netscape.com>.
8. D. Rubine. Specifying gestures by example. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4):329–337, July 1991.
9. Kent Sullivan. The windows 95 user interface: A case study in usability engineering. In *CHI 96*, pages 473–480. ACM SIGCHI, 1996.
10. Wacom. URL <http://www.wacom.com>.
11. Robert C. Zeleznik and Andrew Forsberg. UniCam—2D gestural camera controls for 3D environments. In *1999 ACM Symposium on Interactive 3D Graphics*, pages 169–174. ACM SIGGRAPH, April 1999.
12. Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. Sketch: An interface for sketching 3d scenes. *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 163–170, August 1996.