Pop Through Mouse Button Interactions

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

We present a range of novel interactions enabled by a simple modification in the design of a computer mouse. By converting each mouse button to *pop through* tactile pushbuttons, similar to the focus/shutter-release buttons used in many cameras, users can feel, and the computer can sense, two distinct "clicks" corresponding to pressing lightly and pressing firmly to pop through. Despite the prototypical status of our hardware and software implementations, our current pop through mouse interactions are compelling and warrant further investigation. In particular, we demonstrate that pop through buttons not only yield an additional button activation state that is composable with, or even preferable to, techniques such as double-clicking, but also can endow a qualitatively novel user experience when meaningfully and consistently applied. We propose a number of software guidelines that may provide a consistent, systemic benefit; for example, light pressure may invoke default interaction (short menu), and firm pressure may supply more detail (long menu).

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

INTRODUCTION

Inspired both by the fluid interactivity of the camera button mechanism and the observation that people often press elevator buttons harder as they grow impatient, we considered a number of approaches for integrating pressure controls into computer interfaces. The choice we present, replacing mouse buttons with pop through pushbuttons (see Figure 1), increases the information bandwidth of a mouse, but we hypothesize may well be easier for users to learn, remember, and control than other choices, such as adding spatially distinct or continuously pressure-sensitive buttons. Moreover, our device is easy to deploy because it does not require visible changes to a commercial mouse, and legacy applications can be compatible simply by ignoring the added button state.

After developing a collection of novel interaction techniques to demonstrate the potential of pop through mouse interaction, we performed a patent seach that uncovered a 1994 patent on a substantially similar device by Apple Computer [1]. Interestingly, the patent does not consider any of the interaction techniques that we present.

UIST 01 Orlando FLA

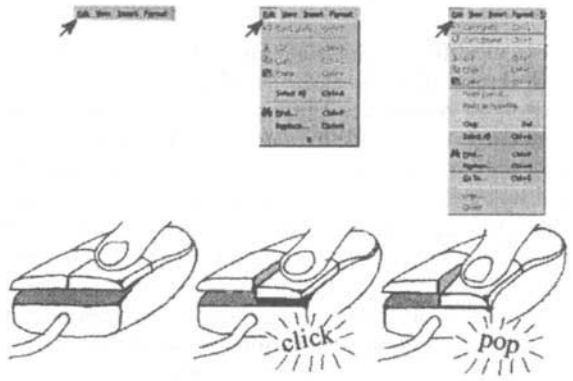

Figure 1: Pressing the button lightly clicks and in this example invokes a short menu; pressing harder pops through with a second click, in this example invoking a longer menu. (Button displacement exaggerated.)

PROTOTYPE HARDWARE AND SOFTWARE

Hardware Design For the first of two different pop through mouse prototypes, we glued a small pushbutton on top of a mouse button and connected it to an RS232 port for software polling.[1] 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. In informal evaluation, users had no difficulty distinguishing or controlling the activations of the native or additional button.

For our second protoype, we replaced the internal button mechanism with an integrated double-action surface-mount pushbutton[2], similar to a camera's focus/shutter-release button. The activation forces for this pushbutton are quite subtle, with no perceptible click for the first contact and a very soft feel, not really a click, for the second, pop through, contact. More sophisticated and complicated design possibilities that provide better tactile feedback are described in [1].

[1]Specifically, we tied DCD to RTS (held high in software) with a 10 kΩ resistor and connected the switch contact between DCD and ground. Our second prototype additionally uses CTS for the second contact.

[2]Digi-Key part #P10845S-ND

Software Techniques To demonstrate pop through interactions in legacy applications, we wrote a tiny X11 driver program, "map-to-right-button", using the XTest extension, that causes the pop through button to generate the same events as the right mouse button. In Netscape, the user can click lightly on a link to follow it and firmly to invoke instead a menu of operations for bookmarking, saving, etc. Also, when editing in the URL "Location" text entry box, the user can press lightly, drag out a selection, and then, without releasing, press firmly to get a menu of text editing operations. Many other legacy X11 applications that map the left mouse button to simple, default actions and the right mouse button to a menu of choices immediately leverage this driver as well.

We also modified a drawing application to read the pop through button directly and to replace a "tear-off" gesture for switching between rubberbanding unconstrained versus constrained lines with the firm pressure that activates the pop through button state. Also, we modified the Unicam [2] interface by replacing the circuitous directional gesture that distinguishes panning from zooming with the activation of panning through light pressure and zooming through firm pressure. In both cases, the pop through button interaction is more functional than the gestures it replaces because it makes possible dynamic transitions between the interactions mapped to the light and firm pressure states.

DISCUSSION

Hardware Issues In both prototypes, 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. The retrofit implementation suffers because pop through can be easily triggered and the first button click can be "dropped" accidentally. We expect that a properly engineered device, perhaps like those described in Apple's patent, could surmount these obstacles with a specially designed switch. We have not explored the device's ergonomic implications (including repetitive stress injury) but expect problems can be limited to those of a standard mouse.

Software Issues Although some legacy applications are synergistic with the map-to-right-button driver, others fail to leverage the pop through state without code modification. This incompatibility arises because the pop through button generates a necessarily chorded, second button event that is often ignored by user interface toolkits and applications. A workaround for some applications is, among other things, to translate the pop through event into a left button up event followed by a right button down event.

Applications written specifically to exploit the pop through button must consider how best to leverage the additional button state. In our two application examples, we made the obvious decision to map light pressure to a common operation and firm pressure to a less common operation. More general potential guiding principles are described 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 from their computer in favor of their original mouse, only one out of a dozen or so people said yes, citing ergonomic issues of the prototype hardware.

PROPOSED UI GUIDELINES

In order to best leverage the pop through interaction, we believe that applications should adhere to a common guideline for mapping the additional button state. For instance, the additional button could replace double-clicking, be mapped to get help, or used to abort an action. This section proposes a number of additional guidelines and overviews how various applications might benefit.

Computer Bias vs User Bias One guideline is to map light pressure to "computer-biased" interaction, where the computer provides "smart" defaults, and firm pressure (as if frustrated) to "user-biased" interaction, where the user is presented with more control. This guideline might be used to: invoke a long menu instead of an abbreviated menu (as in many Microsoft applications), override default constraints such as drawing unconstrained rather than axis-aligned lines in a drawing program, select text by character instead of word, or move a scrollbar or slider with fine control rather than the default coarse control.

Sequential Operations Another principle is to associate commonly sequential operations with both 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".

Two Operations Convenient but less intuitive, different operations can be mapped to the light and firm pressure levels, such as undo and redo, activation of a drawing tool that after use either reverts to the pointer tool or stays, or the Unicam modification of panning and zooming.

ACKNOWLEDGMENTS

NSF STC for Computer Graphics and Scientific Visualization, Alias|Wavefront, Adobe, DOE, Intel, Microsoft, Sun.

REFERENCES

1. B. Duchon, A. Nguyen, and J. Baldwin. Multi-state one-button computer pointing device. U.S. patent 5,585,823, assigned to Apple Computer, Inc., 1996. Filed 1994.

2. R. Zeleznik and A. Forsberg. UniCam—2D gestural camera controls for 3D environments. In *1999 ACM Symp. on Interactive 3D Graphics*, pp. 169–174, 1999.