Click Control: Improving Mouse Interaction for People with Motor Impairments

Christopher Kwan, Isaac Paquette, John J. Magee, Paul Y. Lee, Margrit Betke
Image and Video Computing Group
Department of Computer Science, Boston University
111 Cummington Street, Boston, MA 02215 USA
{ckwan, paquette, mageejo, luc2pl, betke}@cs.bu.edu

ABSTRACT

Camera-based mouse-replacement systems allow people with motor impairments to control the mouse pointer with head movements if they are unable to use their hands. To address the difficulties of accidental clicking and usable simulation of a real computer mouse, we developed Click Control, a tool to augment the functionality of these systems. When a user attempts to click, Click Control displays a form that allows him or her to cancel the click if it was accidental, or send different types of clicks with an easy-to-use gesture interface. Initial studies of a prototype with users with motor impairments showed that Click Control improved their mouse control experiences.

Categories and Subject Descriptors

K.4.2 [**Computers and Society**]: Social Issues—*assistive technologies for persons with disabilities*; H.1.2 [**Models and Principles**]: User/Machine Systems—*human factors*; H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*input devices and strategies*

General Terms

Human Factors

Keywords

Accessibility, assistive technology, Camera Mouse, human computer interaction, mouse gestures, mouse-replacement system, video-based interface

1. INTRODUCTION

For users with motor impairments who cannot use their hands to operate a computer mouse, camera-based mouse-replacement systems, e.g. [1, 2, 3, 4], have been developed to allow control of the mouse pointer with head movements.

One difficulty in using these systems is accidental clicking. If a user is not attentive to whether or not the system is in a clicking-state, e.g. because his or her attention is elsewhere or because there is insufficient feedback, items may be selected unintentionally (especially in systems that require a user to keep the mouse pointer still for a period of time to simulate a click). A lack of precision in using the system, e.g. due to tracking limitations or movement abilities, may also cause a user to select wrong neighboring targets.

Difficulties can also arise when using these systems to simulate the capabilities of a real computer mouse, e.g. left and right-clicking, double-clicking and dragging. For this functionality, some systems impose additional requirements on a user's movement abilities, e.g. certain movements with the nose [2] or winking with both eyes [3]. Also, when using the Camera Mouse [1] with tools[1] for additional mouse functions or the system of Varona *et al.* [4], a user must first move the mouse pointer to an external window at a fixed location to select a command and then must move the mouse pointer back to the target to issue the command. This can be frustrating if a user sends the command to the wrong target (common in cluttered areas like the desktop) and tiring if he or she must repeatedly move the mouse pointer back and forth between the target and the command window.

To address these difficulties, we developed Click Control, a tool that augments the abilities of camera-based mouse-replacement systems. Click Control: (1) works with any system that allows control of mouse pointer movement, (2) provides visual feedback to notify a user before clicks will occur and allows him or her to dismiss unintentional clicks, and (3) simulates the commands of a real computer mouse using a gesture-based approach that does not impose requirements on a user's movement abilities.

2. CLICK CONTROL FEATURES & USAGE

A user launches Click Control as a separate application along with his or her mouse-replacement system (we used Camera Mouse [1]). Click Control waits in the background until the user tries to perform a mouse command. By default, Click Control is triggered by the `MouseDown` event sent by the mouse-replacement system to the operating system. However, this trigger could be changed to another binary switch, e.g. a keypress, voice command or physical input.

When the user triggers that they want to perform a mouse command, Click Control stores the current location of the mouse pointer and displays a gesture form (Fig. 1) near that location. The semi-transparent gesture form displays a crosshair to show the user exactly where the command will be sent and displays a set of buttons with which he or she can perform "gestures" to send different types of mouse commands. The form serves as an indicator that the system

[1]ClickAid (`http://www.polital.com/ca`), Point-N-Click (`http://www.polital.com/pnc`), Dwell Clicker (`http://www.sensorysoftware.com/dwellclicker.html`)

ASSETS'11, October 24–26, 2011, Dundee, Scotland, UK.
ACM 978-1-4503-0919-6/11/10.

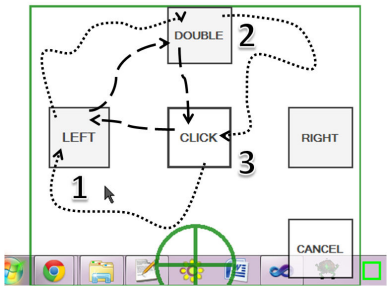

Figure 1: Double-clicking on the sunflower icon on the taskbar with the gesture form. Dashed and dotted lines show two different mouse trajectories.

is about to click, giving the user both visual feedback and the opportunity to cancel the action if it was unintended.

A "gesture" entails moving the mouse pointer over the buttons on the form in a certain order. E.g., a left-click is sent if the user touches the left then center buttons; a double-left-click is left, top, then center (Fig. 1). Once a valid gesture is recognized, the associated mouse command is sent to the previously stored mouse pointer location. If the click was accidental and the user does not wish to send any command, he or she can dismiss the gesture form or wait until it automatically disappears. We define "gestures" as touching targets rather than following a specific trajectory so as to not constrain the movements a user must be able to make. The gestures were designed to be easy to perform, requiring only the ability to move the mouse pointer and explicit in that they are unlikely to be performed unintentionally.

We implemented Click Control for Windows using the open source Global Mouse and Keyboard Library[2] to capture the `MouseDown` events that trigger the gesture form and to simulate mouse events after gestures are performed.

3. INITIAL USER STUDIES

We conducted a study of Click Control with C, a high school student with cerebral palsy. In previous sessions, C was able to use Camera Mouse along with the external tools for additional mouse functionality. However, C had difficulties knowing when the system was in a clicking-state and as a result would end up clicking on things unintentionally. Also, since it was difficult for C to make very precise movements with Camera Mouse, he would often accidentally send mouse commands to unintended targets in cluttered areas.

We introduced Click Control to C and he was able to understand its usage within a few minutes of experimenting. C was now notified by Click Control before Camera Mouse was about to click, so that he could react appropriately. When trying to click on targets in cluttered areas, C was able to quickly cancel clicks to unintended targets. C was also able to use the gesture form to send left and double-clicks to targets of various sizes. He was able to perform tasks that were previously difficult for him, such as opening files on the desktop and manipulating an online video.

C believed that Click Control improved his experience with Camera Mouse. He commented, "I am not struggling as much as before... now it feels like a real mouse."

[2]`http://www.codeproject.com/KB/system/globalmousekeyboardlib.aspx`

We also worked with F, a speech language pathologist, to conduct a study with two middle school students with cerebral palsy. In previous studies, the students were able to use Camera Mouse to interact with several custom applications but were often interrupted by accidental clicks.

Click Control was able to alleviate the problem of accidental clicking. The gesture form would appear before the system was about to send a click, at which point the students dismissed the click if it was accidental. The students were also able to perform gestures to send left-clicks but often needed guidance from F.

The students were able to use Click Control but it may not have been intuitive to them. They seemed to understand F's analogy that it was like a teacher asking "Are you sure?" One student told us, "I liked it... I know what I'm doing but it's confusing." To make Click Control more usable to these students, F recommended using images with the gestures and creating a mode where users could confirm or cancel only left-clicks with simplified gestures.

4. ONGOING WORK

We plan to conduct user studies to see how Click Control affects interactions in the longer-term. We will take quantitative measurements of its usage, such as how its use affects the completion time and error rate of performing various mouse-based tasks. We also plan to verify its compatibility with other mouse-replacement systems and to add customization options for gestures. Click Control will be improved from feedback and eventually available for download.

5. ACKNOWLEDGMENTS

We thank the participants of our studies and also Robin Berghaus, Danna Gurari, Fletcher Hietpas, Tessa Skinner, and Ashwin Thangali. We also thank the reviewers for their insightful feedback. Funding was provided by the NSF (HCC grants IIS-0910908, IIS-0855065, and IIS-0713229).

6. REFERENCES

[1] M. Betke, J. Gips, and P. Fleming. The Camera Mouse: Visual tracking of body features to provide computer access for people with severe disabilities. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 10(1):1–10, Mar. 2002.

[2] D. Gorodnichy, E. Dubrofsky, and A. A. Mohammad. Working with a computer hands-free using the Nouse Perceptual Vision Interface. In *Proceedings of the International Workshop on Video Processing and Recognition*, VideoRec'07. NRC, May 2007.

[3] E. Missimer and M. Betke. Blink and wink detection for mouse pointer control. In *Proceedings of the 3rd International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '10, pages 23:1–23:8, New York, NY, USA, 2010. ACM.

[4] J. Varona, C. Manresa-Yee, and F. J. Perales. Hands-free vision-based interface for computer accessibility. *J. Netw. Comput. Appl.*, 31:357–374, November 2008.