Int. J. Human-Computer Studies 97 (2016) 58–76

Contents lists available at ScienceDirect

Int. J. Human-Computer Studies

journal homepage: www.elsevier.com/locate/ijhcs

The design, use, and performance of edge-scrolling techniques

J. Aceituno [a,b,*], S. Malacria [a,c], P. Quinn [c], N. Roussel [a], A. Cockburn [c], G. Casiez [b]

[a] Inria Lille - Nord Europe, 40 avenue Halley, 59650 Villeneuve d'Ascq, France
[b] Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France
[c] Department of Computer Science and Software Engineering, University of Canterbury, Private Bag 4800, Christchurch, New Zealand

ARTICLE INFO

Available online 8 August 2016

Keywords:
Edge-scrolling
Scrolling
Autoscroll
Dragging

ABSTRACT

Edge-scrolling techniques automatically scroll the viewport when the user points near its edge, enabling users to reach out-of-view targets during activities such as selection or drag-and-drop. Despite the prevalence of edge-scrolling techniques in desktop interfaces, there is little public research on their behaviour, use or performance. We present a conceptual framework of factors influencing their design. We then analyse 33 different desktop implementations of edge-scrolling by reverse-engineering their behaviour, and demonstrate substantial variance in their design approaches. Results of an interactive survey with 214 participants show that edge-scrolling is widely used and valued, but also that users encounter problems with control and with behavioural inconsistencies. Finally, we report results of a controlled experiment comparing four different implementations of edge-scrolling, which highlight factors from the design space that contribute to substantial differences in performance, overshooting, and perceived workload.

1. Introduction

Edge-scrolling is an interactive technique that automatically scrolls a viewport when the user points near its edge. It is typically used when other scrolling tools are unavailable or unsuitable, for example when the user is already engaged in a dragging action. Fig. 1 shows several of these situations commonly encountered in desktop applications: (a) selecting a font in a large menu, (b) selecting a wide range of cells in a spreadsheet application, (c) resizing a shape in a drawing application, and (d) moving a file in a subdirectory. In these examples, the user has their pointer near the edge of the viewport and needs to scroll the view to complete their task; edge-scrolling assists by automatically scrolling the view at a certain speed in the desired direction.

Edge-scrolling is used daily by millions of desktop computer users, and is available in most desktop applications. However, there are substantial differences between implementations. For example, a Windows 8 user will experience different behaviours when selecting text in Internet Explorer and in Notepad: in the former, the viewport automatically scrolls while the pointer remains below its bottom edge, but the latter additionally requires continual pointer movement for scrolling to occur. Many other behaviours are displayed both within and between operating systems. These differences are near certain to have important implications for user performance, errors, and preferences, as suggested by results of prior studies into alternative techniques for general scrolling (e.g., Zhai et al., 1997a; Hinckley et al., 2002; Quinn et al., 2012). Yet, despite the ubiquity of edge-scrolling use and the seemingly ad hoc variance of the available implementations, there has been little research to understand how users interact with edge-scrolling and how it should be designed.

To advance understanding and design of edge-scrolling techniques, we present an analysis in four parts, respectively addressing the following questions: (1) what factors influence the design of edge-scrolling techniques; (2) exactly how do current implementations behave; (3) how are existing techniques used and perceived by users; and (4) how is user performance influenced by different edge-scrolling behaviours? After a review of related literature, we answer these questions by first providing a framework for understanding edge-scrolling techniques based on task requirements and the method's behavioural characteristics. We then use this framework to examine 33 existing edge-scrolling implementations, reverse-engineering their behaviour and demonstrating substantial variance in design approaches. Results of an interactive online survey, completed by 214 participants, confirm that edge-scrolling is widely used, and identifies common usability

* Corresponding author at: Inria Lille - Nord Europe, 40 avenue Halley, 59650 Villeneuve d'Ascq, France.
E-mail addresses: join@oin.name (J. Aceituno), sylvain.malacria@inria.fr (S. Malacria), philip.quinn@canterbury.ac.nz (P. Quinn), nicolas.roussel@inria.fr (N. Roussel), andy@cosc.canterbury.ac.nz (A. Cockburn), gery.casiez@univ-lille1.fr (G. Casiez).

http://dx.doi.org/10.1016/j.ijhcs.2016.08.001
1071-5819/

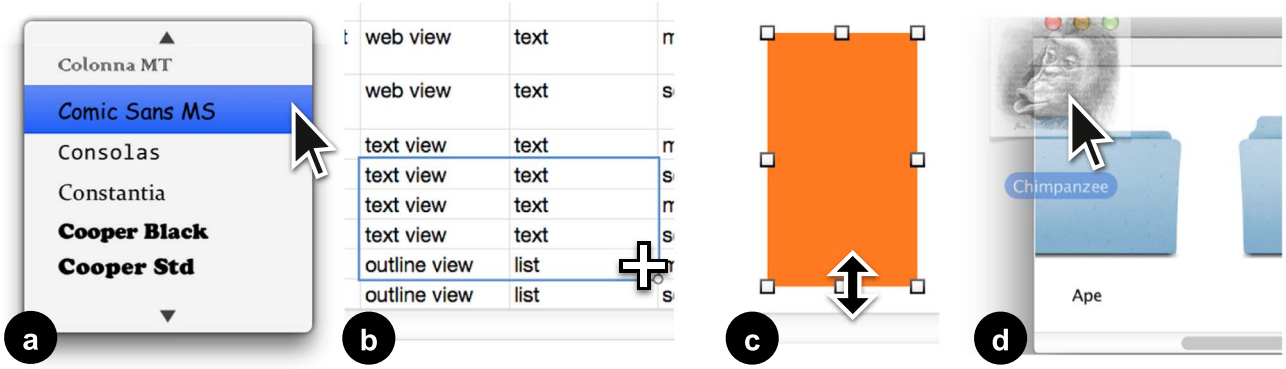

Fig. 1. Situations commonly encountered in desktop applications where *edge-scrolling* can be employed to automatically scroll a viewport to complete a task: (a) selecting a font in a large menu, (b) selecting a wide range of cells in a spreadsheet, (c) resizing a shape in a drawing application, and (d) moving a file in a subdirectory.

problems. We then describe a controlled experiment evaluating user performance when using various edge-scrolling designs for a text selection task, with results showing that different approaches to supporting edge-scrolling cause substantial differences in task completion time and perceived workload. Finally, from these results, we derive a series of recommendations for the design of future edge-scrolling techniques.

2. Background

When a *document* has more information than can be displayed inside a *viewport* placed on the *display*, the interface must allow scrolling so that relevant document parts can be brought into view. Various scrolling techniques and devices have received much attention in the literature, but it seems to us that academic research has missed an opportunity to formalize understanding on an important interaction that is widely disseminated yet problematic. Edge-scrolling defines a *scrolling area* relative to the *viewport edge* (Fig. 2), and when the pointer is inside the scrolling area, pointing actions control the scrolling rate.

This section reviews related literature relevant to edge-scrolling, including general scrolling techniques and transfer functions, pointing-based scrolling techniques, edge-based pointing techniques, as well as edge-scrolling disclosures.

2.1. Scrolling techniques and transfer functions

Scrolling has been a basic component of interaction since the earliest graphical interfaces (Sutherland, 1963), and a wide array of input devices and techniques have been used to support it. Widely deployed hardware devices with scrolling adaptations include the keyboard (McLoone et al., 2003), touchpad (Arthur et al., 2008; Bial et al., 2010), and mouse (Ohno et al., 1985; Zhai et al., 1997b; Hinckley and Sinclair, 1999). Desktop and touch-enabled systems also commonly support scrolling methods that are based on pointing actions, such as scrollbar controls, middle-click rate-based scrolling (Zhai et al., 1997b), and touch-based flick-scrolling (Aliakseyeu et al., 2008; Quinn et al., 2013). Other forms of input for scrolling control include gaze (Kumar and Winograd, 2007), the position of a mobile device in 3D space (Mehra et al., 2006), and device orientation (Rekimoto, 1996).

All scrolling techniques require a *transfer function* that determines the mapping from the input signals to the resultant scrolling output, which is typically in terms of either the *absolute* position of the document, its *relative* position, or its *rate* of movement (although other variants are theoretically possible, such as document acceleration). The following subsections review work on these three primary forms of control. We use the conceptual model of transfer functions provided by Quinn et al. (2012), which describes three successive transformations: *translation* (where the input signal is converted to display space units), *gain* (which modifies the amplitude of the control signal), and *persistence* (where effects, such as simulated friction, are applied across time).

Fig. 3 provides an overview of relationships between literature reviewed in this section.

2.1.1. Absolute position control

Absolute position control functions map each point inside an area in a physical or virtual coordinate space to a single position of the viewport in document coordinates—they map an input position to a scroll position. Such functions have been used in dedicated devices (Buxton and Myers, 1986; Bial et al., 2010) and on touchscreens (Fitchett and Cockburn, 2010). They allow users to quickly jump to remote parts of a document, but precise selections and short-distance movements are compromised (Fitchett and Cockburn, 2010).

Precision problems occur at the *translation* stage of the transfer function, either because the number of sampled elements of the input space does not match the number of elements in the document space or because the transfer function itself does not

Fig. 2. Edge-scrolling is used in the course of a task (here, selecting text) in order to scroll a *document* that is displayed inside a *viewport* placed on the *display*. The user activates edge-scrolling by entering a *scrolling area* near the *viewport edge* (left). Inside the scrolling area, pointing actions control the pace of scrolling (middle, right).