MouseHints: Easing Task Switching in Parallel Browsing

Luis A. Leiva
ITI – Institut Tecnològic d'Informàtica
Universitat Politècnica de València
Camí de Vera, s/n - 46022 (Spain)
luileito@iti.upv.es

Abstract
We present a technique to help users regain context either after an interruption or when multitasking while performing web tasks. Using mouse movements as an indicator of attention, a browser plugin records in background the user's interactions (including clicks, dwell times, and DOM elements). On leaving the page, this information is stored to be rendered as an overlay when the user returns to such page. The results of a short study showed that participants resumed tasks three times faster with MouseHints and completed their tasks in about half the time. Related applications and further research are also envisioned.

Keywords
Parallel browsing, task switching, tabbed browsing, pointing devices, mouse tracking

ACM Classification Keywords
H.5.2 [User Interfaces]: Input devices and strategies, Windowing systems; H.5.3 [Group and Organization Interfaces]: Web-based interaction; H.5.4 [Hypertext/ Hypermedia]: Navigation

General Terms
Experimentation, Human Factors, Performance

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.
ACM 978-1-4503-0268-5/11/05.

Introduction and Background

The power of browsing has grown substantially in the last decade, becoming today a traditional interface for many computing tasks. We now use the Web to multi-task the activities we do every day, to the extent that it is not unusual to see users with a dozen applications and browser instances open at a time; e.g., sharing pictures, listening to music, or shopping, just to name a few. By providing tabs, web browsers have started supporting parallel browsing, allowing users to engage multiple concurrent pages simultaneously [6]. The current active tab is a *foreground task* and thus it has the user's attention, while other tabs or windows may be loading in background or contain information that is not yet needed [8].

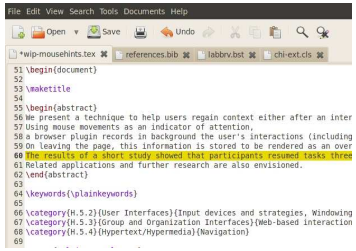


Figure 1: A usual approach for easing attention shifts in tabbed interfaces (in this case, a text editor). The last edited line is automatically marked by highlighting the text background, so when the user switches back to the current tab she can realize faster where she left writing.

People thus may cognitively coordinate multiple tasks through multi-tabbing, having many pages open at the same time and switching between them in any order. However, users are susceptible to overload, making attention and workflow both delicate and difficult to maintain [9]. The findings that multitasking over different types of tasks can reduce productivity [12] is further supported by the single channel theory, which suggests that the ability of humans to perform concurrent mental operations is limited by the capacity of a central mechanism [13]. Therefore, multitasking may seem efficient at first glance but actually it may take longer and lends itself to more errors.

Pointing and Gaze-based Devices

Researchers have developed interesting applications by using eye tracking technology to support task interruption. In fact, our work was inspired in *Gazemarks* [10]. However, the selection of mouse tracking over gaze tracking equipment for illustrating our technique was pragmatic. First, from a HCI point of view, the computer mouse, also including other pointing-related devices such as styli or touchscreens, is the most widely used instrument to browse web pages, also being the closest device to the eye tracker on the spectrum of various modalities [4]. Second, unlike a mouse, it is relatively difficult to control gaze position consciously and precisely at all times. And third, eye tracking equipment is alas still less stable and accurate than most manual input devices.

Mouse Tracking in HCI

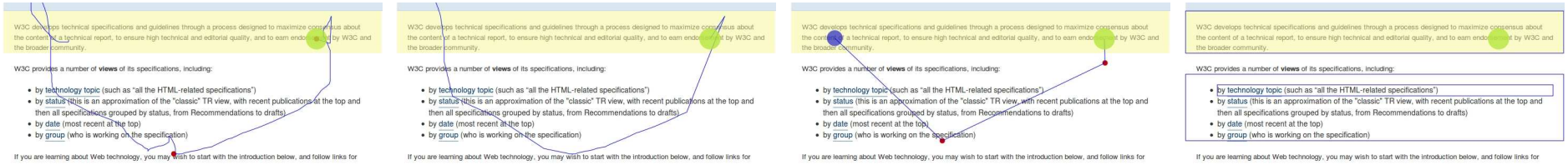
Previous studies have explored mouse movements as a fairly accurate reflection of eye movements (e.g., [5]). Mouse trajectories have also been studied as an indicator of how information is interpreted [1], to predict gaze position from mouse coordinates [7], to evaluate usability of interfaces [2], and even to cluster web documents [11]. However, we are not aware of the use of mouse tracking in the same scope as in this work, i.e., to ease attention in application/task switching while browsing the Web.

Our Proposal: MouseHints

Kern and co-authors [10] showed that some users, in order to keep track of where they were, tended to use the mouse cursor as a marker or to highlight the last line of a text paragraph. A similar approach is implemented in some text editors (see Figure 1). We exploit this notion to remove the need of having to explicitly find a placeholder and/or actively manipulate it, without requiring additional hardware or any special setting.

System Basis

Only one web page and a corresponding tab representing it can be active at the same time in a browser window. Tapping this fact, our system tracks in background the mouse activity in the current tab. Upon switching such a tab back, the system "hints" a subset of the mouse



(a) Raw trajectory (event-based) **(b)** Digest (velocity-threshold identification) **(c)** Clustering (sequential K-means) **(d)** DOM history (n-best hovering frequency)

Figure 2: Visualization options for displaying the same mouse track. The right-most (green) circle represents the last cursor position, while the smallest (red) circles represent mouse clicks. The bounding box of the last interacted HTML element is also highlighted. Figure 3 shows the original page, to allow the reader to compare each option with the source.

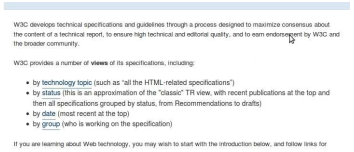


Figure 3: Original test page, with no overlays.

movements (30 seconds by default), also highlighting the last interacted element and the last cursor position (see Figure 2). Then the rendered layer fades out in 500 ms.

User-System Interaction Protocol

When the user selects a browser tab, a `focus` event is triggered and MouseHints records the position of the cursor every time she moves the mouse. When the user switches to another tab, two browser events are fired sequentially: a `blur` event from the old tab and a `focus` event from the new (now current) tab. MouseHints thus stops recording in the old tab and begins to track the activity in the current tab. When the user switches back to a previously visited tab, mouse data are overlaid on top of the HTML content, and are then reset. One may note that if the user switches to a desktop application, only a `blur` event can be detected. However, when switching back to the web browser, a `focus` event will be triggered, therefore enabling MouseHints again.

Implementation

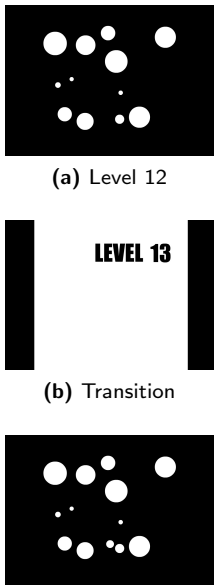
MouseHints was developed as a Firefox extension since such browser has a powerful mechanism that made it relatively easy to code and test. The browser interface was structured in XUL. Both the logic and tracking algorithms were both written entirely in JavaScript. The visualization was coded in HTML5 throughout the `canvas` element, supported since version 1.5 of that browser.

Visualization

We decided to represent the mouse path in a reasonable fashion while unobtrusively highlighting the last interacted HTML element. We developed a generic DOM selector that translated the mouse activity (e.g., hovering, clicking) into CSS selectors, so that the system could draw the corresponding bounding box of such interacted elements. Additionally, we implemented four different mouse path visualization options:

1. The raw mouse trail (Subfigure 2a).
2. A "digest" of the original trajectory (Subfigure 2b).
3. Clusters of mouse coordinates (Subfigure 2c).
4. A DOM-only visualization (Subfigure 2d).

In order to evaluate this system, we showed the four visualization options to 6 participants and let them vote which one they preferred. The option that most people selected was number 2, so we used it for the test.



(a) Level 12

(b) Transition

(c) Level 13

Figure 4: While browsing, participants were eventually interrupted to play a game.

Evaluation Methodology

The tested hypothesis was that using MouseHints should benefit the users in terms of orientation in parallel browsing, i.e., faster task resumption and work completion by having the mouse interactions as a visual remainder.

Participants 36 unpaid volunteers (11 females) were recruited via email. They were told to participate remotely in a study that would measure their reaction times while browsing. All of them were regular computer users accustomed to tabbed browsing, aged 19 to 45 (M=25.5).

Apparatus We developed two Firefox extensions: the MouseHints application and a very basic logging system with the routines of the study. Half of the participants were asked to install both extensions on their computer. The other half of users, who were not aware of the existence of MouseHints, installed the logging extension.

Design A between-groups design was employed, with half of the subjects performing the tasks in only one condition (18 users in the control group and 18 users in the experimental group, respectively). The outcome measures were task success, time for task resumption, and time for task completion.

Procedure Each user performed two tasks, which were common to both groups. Each task took them about 5 minutes to perform in average, as it was dependent on each participant's browsing capabilities. The evaluation was done remotely, to allow subjects to browse in their own working environments. The tasks consisted in searching information for different topics (to mitigate possible learning effects between tasks); e.g., "how would you make <noun> in the minimum number of steps?" or "find the name of the last chapter of the book entitled <noun>". Participants had to interrupt normal navigation flow to play a popular game[1] in a dedicated browser tab. Such a game, despite being quite straightforward, required a lot of visual attention: the user had to click the last-born circle on each level (see Figure 4). The conditions were browsing in a normal environment (control), and with the help of MouseHints (experimental).

To measure how visual attention differed between both groups, at least two tabs had to be opened: one with the game and other with a regular web page. After a random delay between 20 to 40 seconds, the browser changed the focus of navigation from the current tab to the game tab, and users had to resume playing. After another delay, the browser changed the focus to another tab, which was randomly chosen from all opened tabs, to stress the cognitive load during the test. We measured the time for task resumption (first time to move the mouse inside the page) and time for task completion (total browsing time) for all opened tabs. Users were told to close their browser when a task goal was achieved — this allowed us to easily post-process their data.

In both conditions data were saved as timestamped event sequences in the local file system. In order to preserve the user's privacy, URLs were converted to MD5 hashes and data were stored in plain text format. In this way participants could verify that their data were sufficiently anonymized, and could also review what kind of information the extension was gathering. Then they were asked to submit the log files via email.

Preliminary Results and Discussion

We report measures on the three areas suggested by the ISO 9241-11 standard: *effectiveness* (completion rates and errors), *efficiency* (time on task resumption and

[1] http://tubegame.com/camera_mind.html

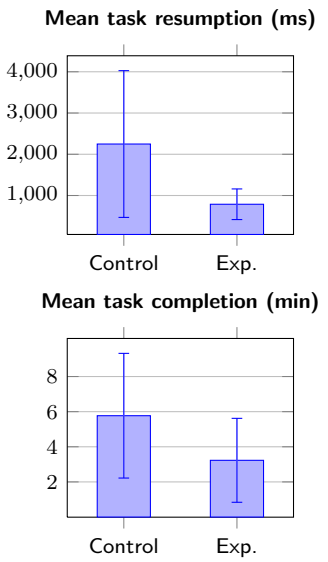


Figure 5: Between-groups efficiency comparison. Candidate removal was not adequate enough for this test due to the relatively small size of the end user sample (18 subjects in each group).

completion), and *satisfaction* (users' subjective opinions on using the system).

Study on Effectiveness
We used a Pearson's chi-square test for this study. The nominal outcomes were task success/failure, measured by assigning 1 point each time the goal was achieved (based on the manual revision of user's comments that were submitted by email). All participants except for one user from the control group were able to finish the assigned tasks, concluding that there were no statistically significant differences in effectiveness between both groups ($\chi^2_1 = 1.09$, $p = 0.29$, two-tailed). This result was almost evident. In fact, MouseHints is just an interaction assistant and, as expected, the user's success did not depend on using this system for achieving their goals.

Study on Efficiency
In this case we used a Kolmogorov-Smirnov test. The continuous outcomes were time for task resumption and time for task completion. We used the median as central tendency measure for reducing the influence of outliers. As predicted, participants were found to be considerably faster in task resumption with MouseHints (Mdn = 791.5 ms) than without (Mdn = 2363 ms), $D = 0.72$, $p < 0.001$, two-sided hypothesis. We achieved similar conclusions regarding task completion (Mdn = 3.9 minutes with MouseHints; Mdn = 6.8 minutes without), $D = 0.5$, $p < 0.05$, two-sided hypothesis.

Study on Satisfaction
Participants from the experimental group submitted an online System Usability Scale (SUS) questionnaire [3] after finishing the study. A Likert scale, from 1 (strongly disagree) to 5 (totally agree), was used to rank ten questions. Then the reported a composite measure of the overall usability of the system. The result was a score of 87.6, indicating that people indeed liked using MouseHints. (SUS scores have a range of 0 to 100.)

The form attached to the online questionnaire allowed users to submit free comments and ideas. A frequently reported comment among participants in the experimental group was that MouseHints was considered helpful. Moreover, participants often mentioned the advantage of saving time and easing task resumption (12 of 18). Eight people liked the aid to memory of not having to remember what they previously did with the mouse in a page.

Known Limitations

First of all, participants performed tasks in an uncontrolled environment and without experimenter supervision. That could explain the large variability in the gathered data (see Figure 5), maybe due to potential outside distractions, or also because some tabs could not be relevant to the assigned task. Second, our approach is not suitable for the user that does not use the mouse (or a similar pointing device) at all while browsing the Web. Besides, there are situations where the eye and the mouse are not in sync; and we believe that our approach may not be much useful if such behavior happens frequently. Clearly, users who move the pointing device according to their focus of attention may be the most benefited target from MouseHints. Third, we proposed for the study very general tasks that were relatively simple to perform, and we believe that MouseHints could be of greater help in other environments (see 'Envisioned Applications'). Fourth, gathered data comprised about ten minutes of task execution data for each user. It would be interesting nevertheless to evaluate the effects of MouseHints in a large-scale study, where users will probably be more accustomed to the system. Finally, users can assist web browsing by using more advanced I/O devices such as

speech recognizers or eye trackers. Therefore, we encourage MouseHints to be used in combination with such systems, since we believe they can unobtrusively coexist with each other.

Envisioned Applications

We have employed the browser events to track the user's interactions and detect task switching. However, our client-side implementation could provide additional analysis features to the user. For instance, the browser could also act as a personal organizer, prioritizing and reordering tabs according to browsing usage. MouseHints could also support web browsing on mobile phones or tablets, e.g., in situations where the user should halt browsing because of a call or a push notification. Additionally, at a higher level of application, one could implement our event detection method (see 'User-System Interaction Protocol') throughout an accelerometer, providing support for ambient intelligence. In this way, it would allow mobile users to resume a task, e.g., after leaving the device on a table because of an interruption.

Conclusions and Future Work

Despite the known drawbacks of our approach, results have supported our initial hypothesis, showing that MouseHints can effectively assist the user in parallel browsing. Moreover, this could be a promising technique for enhancing desktop-based interactions when multitasking. Future work includes replicating the experiment for pure visual-oriented tasks in a controlled setting, also comparing each of the four implemented visualizations. We also plan to research more related applications, since the potential fields are endless.

Acknowledgements Work partially supported by the Spanish research programme MIPRCV (CSD2007-00018).

References

[1] E. Arroyo, T. Selker, and W. Wei. Usability tool for analysis of web designs using mouse tracks. In *Ext. Abstracts CHI*, pp. 484–489, 2006.

[2] R. Atterer, M. Wnuk, and A. Schmidt. Knowing the user's every move – user activity tracking for website usability evaluation and implicit interaction. In *Proc. WWW*, pp. 203–212, 2006.

[3] J. Brooke. SUS: A "quick and dirty" usability scale. In *Usability Evaluation in Industry*. Taylor and Francis, 1996.

[4] M.-C. Chen, J. R. Anderson, and M.-H. Sohn. What can a mouse cursor tell us more? correlation of eye/mouse movements on web browsing. In *Ext. Abstracts CHI*, pp. 281–282, 2001.

[5] L. Cooke. Is the mouse a "poor man's eye tracker"? In *Proc. STC*, pp. 252–255, 2006.

[6] P. Dubroy and R. Balakrishnan. A study of tabbed browsing among mozilla firefox users. In *Proc. CHI*, pp. 673–682, 2010.

[7] Q. Guo and E. Agichtein. Towards predicting web searcher gaze position from mouse movements. In *Ext. Abstracts CHI*, pp. 3601–3606, 2010.

[8] J. Huang and R. W. White. Parallel browsing behavior on the web. In *Proc. HT*, pp. 13–18, 2010.

[9] K. Humm. Improving task switching interfaces. Technical Report COSC460, University of Canterbury, 2007.

[10] D. Kern, P. Marshall, and A. Schmidt. Gazemarks: Gaze-based visual placeholders to ease attention switching. In *Proc. CHI*, pp. 484–489, 2010.

[11] L. A. Leiva and E. Vidal. Assessing user's interactions for clustering web documents: a pragmatic approach. In *Proc. HT*, pp. 277–278, 2010.

[12] J. S. Rubinstein, D. E. Meyer, and J. E. Evans. Executive control of cognitive processes in task switching. *Journal of Experimental Psychology*, 27(4):763–797, 2001.

[13] R. Schweickert and G. J. Boggs. Models of central capacity and concurrency. *Journal of Mathematical Psychology*, 28(3):223–281, 1984.