Available online at www.sciencedirect.com
ScienceDirect
Int. J. Human-Computer Studies 69 (2011) 769–785
International Journal of Human-Computer Studies
www.elsevier.com/locate/ijhcs

Improving cascading menu selections with adaptive activation areas

Erum Tanvir[a], Andrea Bunt[a,*], Andy Cockburn[b], Pourang Irani[a]

[a]Department of Computer Science, University of Manitoba, Winnipeg, Canada
[b]Department of Computer Science and Software Engineering, University of Canterbury, Christchurch, New Zealand

Received 23 July 2010; received in revised form 22 June 2011; accepted 27 June 2011
Communicated by A. Sears
Available online 5 July 2011

Abstract

Cascading menus are the most commonly used hierarchical menus in graphical user interfaces (GUIs). These menus, however, tend to have elongated paths with corner steering, which can result in navigation difficulties. To resolve the corner steering problem, most current cascading menus implement an explicit time delay between the cursor entering or leaving a parent menu item and posting/unposting the associated menu. In this paper, we present adaptive activation-area menus (AAMUs), a technique to improve cascading menu performance by providing a localized triangular activation area between the menu and the child submenu. This triangular activation area aims to overcome the corner steering problem by permitting quick diagonal navigation without imposing a time delay.

We describe four experiments designed to refine and validate the AAMU technique. Our first experiment shows that AAMUs improve item selection performance in comparison to traditional menus and a number of competing techniques, including gesture-based menus and enlarged activation-area menus (EMUs). Our second and third experiments reveal, however, that in a searching task, where the user has to look through multiple submenus to find the target, the basic AAMU design suffers from a ‘‘cursor trapping’’ problem, where the user has to move the cursor out of the activation area prior to exploring another submenu. An evaluation of an improved AAMU design shows that it is as fast as or faster than traditional menus and EMUs for both selection and searching tasks.
Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.

Keywords: Cascading pull-down menus; Menu navigation; Steering; Selection; Evaluation

1. Introduction

Menus, which appear ubiquitously in WIMP (window, icon, menu, pointing device) interfaces, provide users with a convenient means of interacting with and selecting from the set of available functionality. Increasing complexity of software systems, however, has resulted in larger and larger menus, thus necessitating ways to improve menu categorization and selection efficiency. A *cascading* or hierarchical menu (see Fig. 1) addresses the issue of increasing menu size by providing a submenu of choices that are related to the item in the parent menu that invokes the submenu.

Although cascading menus provide the advantage of presenting a large number of selections within a small amount of screen space, they have their limitations. In particular, traditional cascading menus require the user to steer the cursor along an elongated, narrow path when traversing the parent item to select an item in a submenu, which increases selection errors and decreases efficiency (Accot and Zhai, 1999; Pastel, 2006). As shown in Fig. 2, an elongated and narrow path can cause unexpected selections and unintended submenu appearance or disappearance due to straying mouse movements. To mitigate the steering problem, traditional cascading menus include a *time delay*: When the user's cursor rests on a cascading item, the child submenu is posted after a pre-determined period of time (e.g., 200 ms). While the time delay reduces errors due to straying mouse movement, it also slows down the navigation process and selecting the ‘‘right’’ time delay is difficult. Users can pre-empt the delay by clicking on the cascading item to open the child submenu, however, this additional

*Corresponding author. Fax: +1 204 474 7609.
E-mail addresses: etanvir@cs.umanitoba.ca (E. Tanvir), bunt@cs.umanitoba.ca (A. Bunt), andy@cosc.canterbury.ac.nz (A. Cockburn), irani@cs.umanitoba.ca (P. Irani).

1071-5819/$ - see front matter Crown Copyright © 2011 Published by Elsevier Ltd. All rights reserved.
doi:10.1016/j.ijhcs.2011.06.005

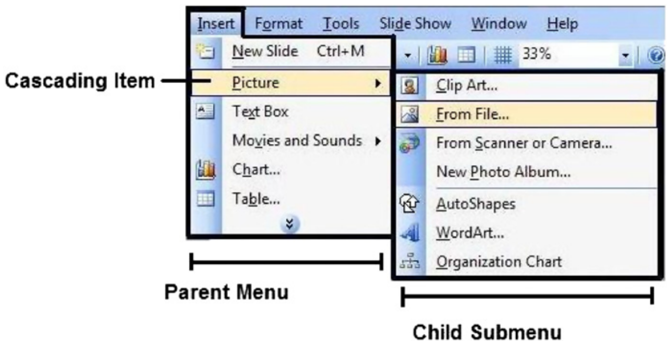

Fig. 1. An example of a two-level deep cascading pull-down menu.

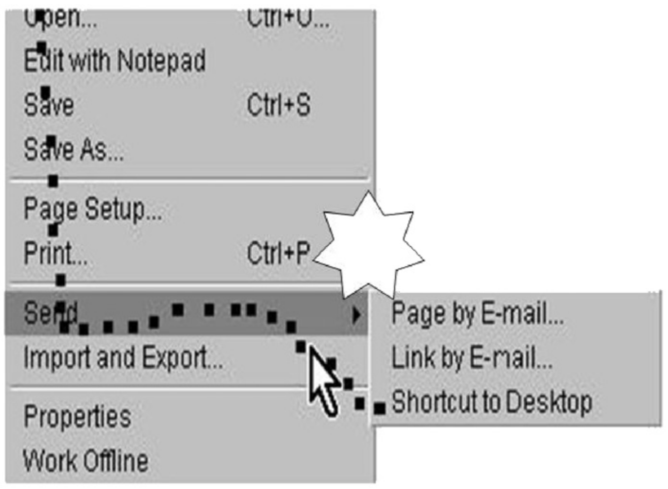

Fig. 2. An elongated path causing a movement error. The submenu disappears unexpectedly as the cursor crosses the border of the lower item (adapted from Kobayashi and Igarashi, 2003).

interface operation is not without a performance cost. Thus, traditional cascading menus require either difficult steering or performance penalties through time delays or additional interface operations.

In this paper, we introduce a new technique, *adaptive activation-area menus* or AAMUs, to improve selection and navigation in linear cascading pull-down menus. This technique introduces a triangular adaptive activation area that changes its size with respect to the size of the child cascading menu, providing the user with a broad steering path that permits diagonal movements (e.g., see Fig. 6 in Section 3). We use the term "activation area" since the submenu remains active as long as the cursor is within the area. This adaptive activation area removes the need to steer through narrow elongated paths, without introducing the cost of a click or a time delay. Through a series of controlled empirical evaluations, we validate and refine the AAMU technique. Our evaluations show that AAMUs are a robust choice for cascading menu design as their performance is as good as or better than other techniques across a number of tasks, menu depths, devices and cascading densities (the percentage of cascading items within a single level of the menu hierarchy).

The primary contributions of this work are twofold. First, we introduce AAMUs, a novel technique to improve cascading menu performance. Second, we report the results of four controlled empirical evaluations that test the advantages and limitations of the AAMU technique. Whereas prior work in the area (e.g., Cockburn and Gin, 2006; Kobayashi and Igarashi, 2003; Ahlström, 2005) has often compared their proposed techniques against only traditional cascading menus, we compare AAMUs to both traditional menus and a number of proposed improvements. Thus, our evaluations provide a more complete understanding of the design space for cascading menus than what exists to date.

The rest of this paper is structured as follows: Section 2 describes background and related work. Section 3 introduces the details of AAMUs. Section 4 summarizes the goal of four evaluations that were performed to improve and validate AAMUs, whereas Sections 5–9 describe these evaluations. Section 10 discusses implications of our evaluations and presents potential directions for future research.

2. Background and related work

In this section we survey work related to improving cascading menu interactions. We begin by reviewing two predictive models that are useful in assessing menu performance from a theoretical standpoint: Fitts' law and the Steering law. We then discuss prior work on improving the performance of linear cascading menus, to which our work most directly relates.

2.1. Theoretical models for predicting performance in menu selection

Two theoretical models that relate to the design of cascading menus are Fitts' law and the Steering law.

Fitts's (1954) law is a robust and widely adopted model for human movement that predicts the time required to move from a starting position to a final target as a function of the distance to the target and the size of the target. The Shannon formulation (MacKenzie, 1992a,b) of Fitts' law is as follows:

$$MT = a + b \times \log_2\left(\frac{D}{W} + 1\right),$$

where D is the distance, W is the width of the object, and a and b are empirically derived constants.

To extend Fitts' law to 2D navigation within constrained paths, Accot and Zhai (1997) developed the Steering law, which predicts the average time necessary to navigate or steer a pointing device (e.g., a mouse or stylus) through a 2D path, tunnel or trajectory. This model, which has been applied to traditional hierarchical cascading menus (Accot and Zhai, 1999), states that the time required to travel a trajectory is directly proportional to the distance traveled and inversely proportional to the width of the path. In its general form, the Steering law expresses the time T required to steer through a tunnel as

$$T = a + b\frac{A}{W},$$

where T is the average time to navigate through the path, W is the width of the path, A is the length of the path and a and b are empirically determined constants. A limitation of the Steering law is that it has been verified for only a few path shapes and widths. For instance, steering is difficult through sharp corners and narrow paths (Pastel, 2006), which further explains the navigation problems in traditional menus.

2.2. Improvements to cascading menus

While alternatives have been proposed and studied (e.g., Kurtenbach and Buxton, 1994; Callahan et al., 1988), linear menus remain the most common type of menu in use. Cascading linear menus, where a parent cascaded item contains the submenu, are the most commonly used technique for handling hierarchical menus. Cascading menus, however, demand a high level of steering accuracy as they require users to navigate through elongated paths from the parent items to the submenu items, which according to the Steering law, increases movement time. Researchers have designed various techniques to resolve these problems of cascading pull-down menus. These techniques involve either decreasing the distance to the menu items, or increasing the size of the menu item.

2.2.1. Techniques for decreasing distance

A simple solution to make menu selection and navigation faster is to reduce the Fitts' law targeting requirement, i.e., reduce the distance to the target. The Steering law also predicts that movement time increases with the length of the path to be covered. Kobayashi and Igarashi (2003) presented an improvement to cascading menus that reduces navigation distance and avoids unintended menu postings/unpostings. This technique has two components. The first uses the direction of the cursor movement to determine the menu behavior. Vertical movement of the cursor changes the Highlighted item within the current menu and horizontal motion opens and closes the child submenus, therefore, eliminating the unwanted submenu activation during menu navigation. Second, when the horizontal motion occurs, the submenu pops up near the cursor position, reducing the length of the movement path (see Fig. 3). A user must move the cursor to the right to open up a submenu or to the left to close the submenu and return to the parent menu. A study comparing this direction- or gesture-based menu to traditional cascading menus found a 12% decrease in menu selection times.

Although the user study showed that gesture-based menus improved upon traditional menus, there are two main limitations. The first limitation is the additional right or left movement required to invoke or revoke submenus, respectively. Second, as the child submenu opens closer to the cursor position, submenus overlap their parent menus, and hide the rest of the parent menu items. If the user wishes to select a parent menu item while a submenu is

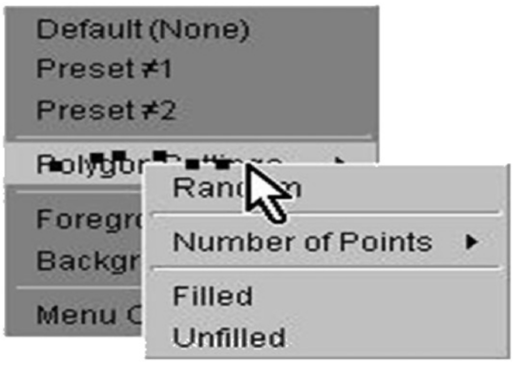

Fig. 3. Gesture-based cascading menus. Horizontal motion towards the right opens a submenu near the cursor position (adapted from Kobayashi and Igarashi, 2003).

open, this overlapping forces the user to make a left horizontal movement to close the submenu prior to interacting with the parent menu.

Another approach to decreasing target distance can be found with the force-fields introduced by Ahlström (2005). Force-fields menus partially take control of the cursor movement from the user. Two types of force-fields are used. First, when moving from left to right within a cascading item, the cursor is pushed towards the child menu and moves faster, optimizing the navigation process. Second, while moving within a non-cascading item, the force-fields keep the cursor in the middle of the item, preventing the cursor from falling outside the parent menu (see Fig. 4). The most important benefit of force-fields menus is that they keep the visual structure of the interface and the interaction technique unchanged. A study comparing force-enhanced menus to traditional cascading menus showed that the force-fields decreased selection times, on average, by 18% when a mouse, a track point, or touchpad was used as an input device. Two disadvantages of this technique, however, are the resistance felt when moving backwards (from right to left), and a loss of user control.

As a variant of force-fields, Ahlström et al. (2006) introduced jumping menus. In a jumping menu, the cursor is warped to the first submenu item immediately after the user clicks on the parent item. A study comparing jumping menus, force-field menus and traditional menus showed that both force-fields and jumping menus improved upon the performance of the default technique. When using a mouse, participants were, on average, 9% faster with force-fields than they were with jumping menus. When using touchpads, some participants benefited more from the jumping menus and others benefited more from the force-fields. Like with force-fields, a disadvantage of jumping menus is a loss of user control.

2.2.2. Techniques for increasing width

The Steering law suggests a second solution for faster steering by increasing the width of the path. A wider path is easier to navigate and less prone to movement errors, causing fewer unintended menu postings and unpostings.

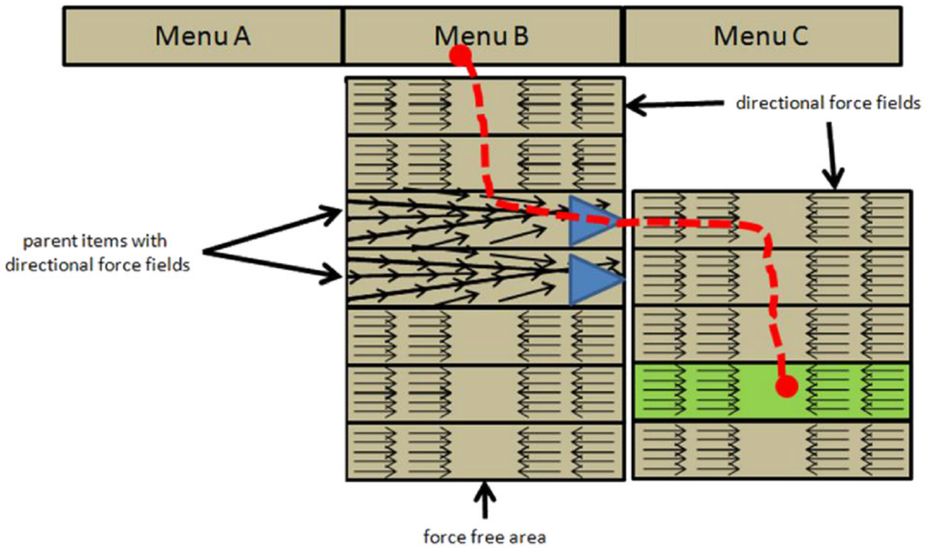

Fig. 4. Cascading menus with force-fields (adapted from Ahlström, 2005).

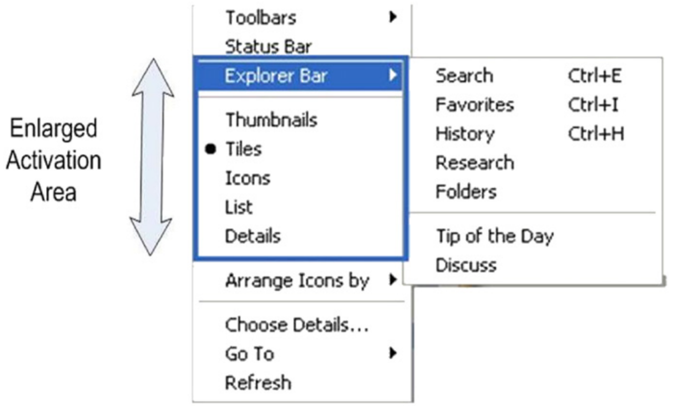

Fig. 5. Enlarged activation-area menus (adapted from Cockburn and Gin, 2006).

A technique developed by Cockburn and Gin (2006) called *enlarged activation-area menus* (EMUs) improves navigation by increasing the activation area of the parent menu associated with each cascaded submenu, providing a wider path for steering (see Fig. 5). The activation areas for each cascading item are increased by extending them up to the end of the menu or by including all the non-cascading items before the next cascading item. An evaluation showed that EMUs were up to 29% faster than traditional menus.

The problem with EMUs is that the activation area is enlarged depending on the density of the cascading items in the parent menu. As a result, in case of adjacent cascading items, the size of the activation area will be equal to that of the traditional cascading menu, offering no performance benefits. Users might also be distracted when a child cascading menu appears while they are targeting a non-cascading item that lies within the enlarged activation area.

Fitts' law also predicts that target acquisition can be improved by increasing the size of the target. Fisheye menus (Bederson, 2000), for example, dynamically increase the size of the target as the cursor approaches it. They allow many items to be listed on one screen and are a good solution for viewing on small devices like personal digital assistants (PDAs). An evaluation of fisheye menus, however, showed them to be slower than traditional cascading menus.

3. Adaptive activation area menus (AAMUs)

In this section we introduce the adaptive activation-area menus (AAMUs), which improve upon existing techniques for cascading menus by providing users with broader paths to reach the submenus efficiently, even in situations with multiple adjacent cascading items.

AAMUs, shown in Fig. 6, provide users with a broad steering path by means of adaptive activation areas. The size of the activation area is dynamically determined based on the size of the child cascading menu and position of the cursor. The activation area is triangular in shape and overlaps some area of the adjacent menu items; however, the activation area is semitransparent, allowing users to see all items in the parent menu. The broader activation area provides a means to remove the time delay before a cascading submenu is posted, since the activation area removes the ambiguity of the user's intentions.

As an AAMU adapts to the size of the child submenu and initial cursor position, two different submenu alignments are possible:

Center-aligned: If the size of the child menu permits, i.e., if there is enough space available at the top of the cascading item, then the child submenu is placed such that half of its height is above and half is below the cascading item (see Fig. 7(a)).

Top-aligned: If the child submenu is too long to be placed centrally, then following the standard MS

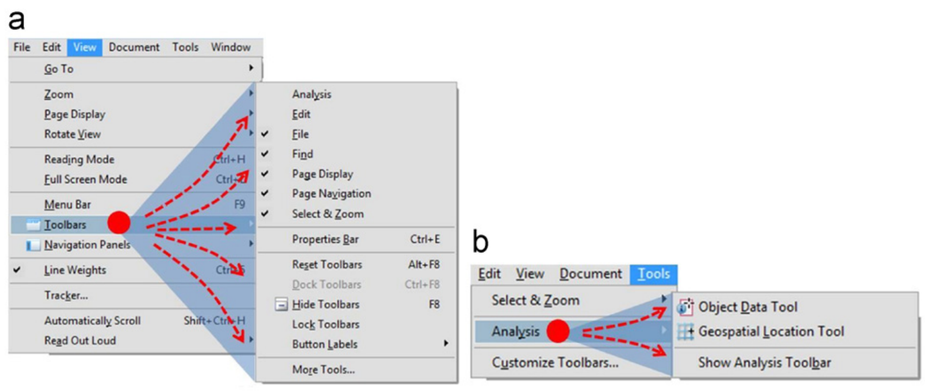

Fig. 6. Examples of adaptive activation areas: (a) a long cascading and (b) a short cascading menu, with expected cursor movements toward submenu items.

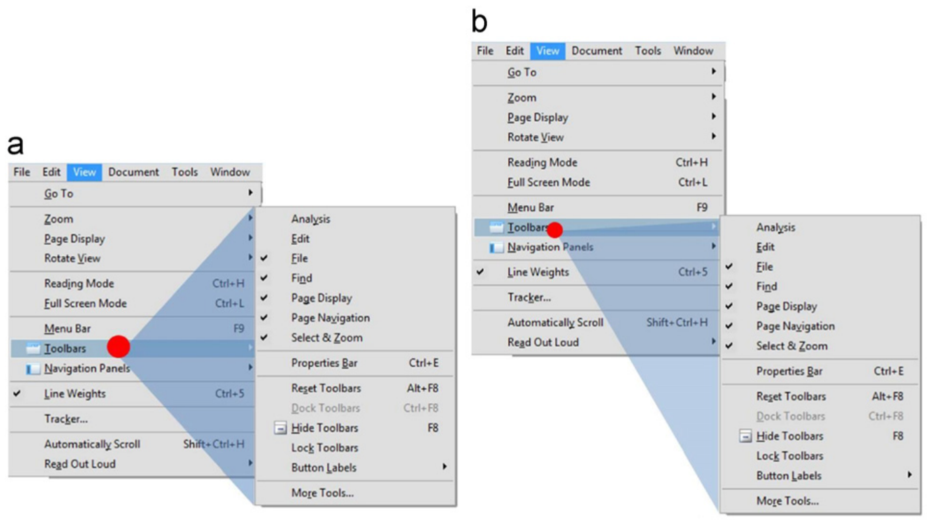

Fig. 7. Examples of different alignments for AAMUs: (a) a centrally aligned child cascading menu and (b) a top-aligned child cascading menu.

Windows layout, its top is aligned with the top of cascading item (see Fig. 7(b)).

The AAMU technique works as follows. When the user places their cursor on a cascading item, a transparent adaptively sized activation area is invoked next to the cursor along with the child submenu. To choose a submenu item, the user can move diagonally towards the child cascading menu. The activation area and child submenu remain posted as long as the cursor remains inside the triangular activation area. To activate another item in the parent menu, the user has to move the cursor outside the boundaries of the current activation area, which will cause the activation area and child submenu to disappear.

AAMUs offer three potential advantages. First, the broader activation area permits diagonal movements, which avoids the problem of steering through elongated narrow paths and sharp corner steering. Second, as opposed to EMUs, users can benefit from a wide activation area even in cases of adjacent parent items. Finally, there is no time delay involved in posting a submenu.

4. User studies to evaluate AAMUs

To validate and refine our AAMU design, we conducted a series of four experiments whose goals were as follows:

- *Experiment* 1[1]: We compared AAMUs to other cascading menu techniques in a selection task representative of a scenario where the user knows the location of the target. In this experiment, we compared the techniques using the mouse as the input device.
- *Experiment* 2: We compared AAMUs to other cascading menu techniques using a search task representative of a scenario where the user does not know the location of the target item within the set of cascading menus. In Experiment 2, we also looked at the effect of input device.

[1]Experiment 1 is based on an earlier work: AAMU: Adaptive activation area menus for improving selection in cascading pull-down menus, in CHI 2008 © ACM, 2008. http://doi.acm.org/10.1145/1357054.1357270. Experiments 2–4 have not yet been published.

- *Experiment* 3: Based on a limitation of the AAMU technique uncovered in Experiment 2, we collected further data on users' movement paths to understand situations where users experience difficulty with the AAMU technique. We then used our observations to design a number of improvements to the original AAMU design.
- *Experiment* 4: We validated our improved AAMU design, by comparing it against the original AAMU design, EMUs and traditional menus for both selection and search tasks.

We describe Experiments 1, 2 and 4 in detail. For the sake of brevity, we provide only an overview of the key findings from Experiment 3, referring the reader to Tanvir (2009) for the remainder of the details. The techniques in all experiments were implemented in Microsoft Visual Studio .NET, using C#.

5. Experiment 1

In our initial validation of AAMUs, we compared AAMUs to the following existing techniques: gesture-based menus (Kobayashi and Igarashi, 2003), enlarged activation area menus (EMUs) (Cockburn and Gin, 2006), force-fields (Ahlström, 2005) and default (i.e., traditional) menus. We also tested an AAMU variant called force-AAMU, which combines force-fields and AAMUs. Force-AAMUs provide the benefit of reduced navigation distance in addition to wide steering paths. Force-fields are implemented only within the adaptive activation area: once the cursor enters the activation area, it is pushed towards the right side. As there are no force-fields in the menu items, no resistance is experienced while entering back into a parent menu, unlike in a force-fields menu. Given that AAMUs do not include a time delay, all menu types were implemented without any time delay to level out the playing field. This decision was made after a pilot study showed that time-delay based techniques were being unfairly penalized in terms of efficiency. Experiment 1 and its results originally appeared at CHI 2008 (Tanvir et al., 2008).

5.1. *Method*

5.1.1. *Participants*

Eleven university undergraduate students participated in Experiment 1 in exchange for course credit. All had used the MS Windows default menu and were familiar with operating a mouse. None were color blind.

5.1.2. *Conditions*

The following menu types were tested in this study: default, AAMU, force-AAMU, EMU, force-fields and gesture-based. We also tested each of the techniques at cascading depths (i.e., number of levels in the hierarchy) 2–4.

5.1.3. *Task and stimuli*

Participants were required to perform 30 menu selection tasks with each technique, with 10 trials at each of three cascading menu depths (2–4). The experimental task simulates a scenario where the user knows the location of the target item. Specifically, the path to the target menu item was Highlighted in green to provide users with a visual cue (see Fig. 8), while the target menu item is displayed in red. A trial began when the participant clicked on the top-level menu heading (the File icon in Fig. 8) and ended when the participant successfully acquired the target.

Menu length was varied randomly (between 4 and 9 items) in each level of depth in every trial with a constant *cascading density* of 50%, where cascading density is the percentage of menu items that are parent items. The target menu item always appeared in the last menu depth level. For each trial, a different path and target position was randomly generated to prevent users from learning the trial path and positioning of the target item. At the start of the experiment, participants were given 5 min of training with each menu type. Participants were instructed to complete tasks as quickly and as accurately as possible.

The order of technique presentation was counterbalanced using a Latin square, while depth was randomized. With six menu types, three depths, and 10 trials per condition, the system recorded a total of 180 trials for each participant. A post-study questionnaire was also administered to obtain data on participants' preferred techniques. The experiment took approximately 25 min.

5.1.4. *Apparatus*

The experiment was conducted on Windows XP using a Pentium 4 machine with 1 GB of RAM. A monitor with a 1024×768 resolution was used. The experiment was performed using an optical mouse.

5.1.5. *Design*

The experiment employed a 6×3 repeated-measures design for factors menu type (default, AAMU, EMU,

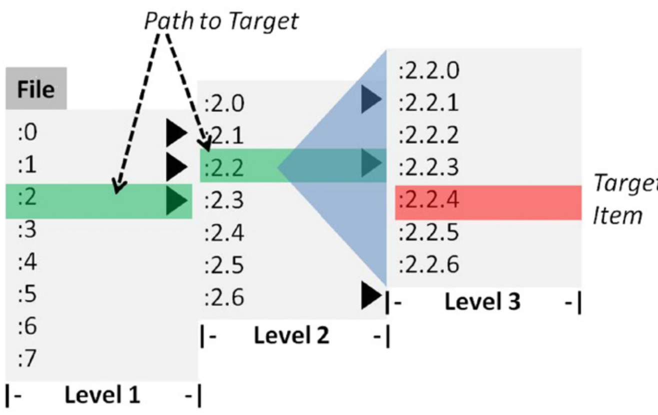

Fig. 8. An example of a three-level deep selection task in Experiment 1. The red item (item 2.2.4) is the target and the green items (items 2 and 2.2) indicate the path. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

force-fields, Force-AAMU, and gesture-based) and menu depth (targets at cascading depth 2, 3, or 4). The dependent variable was task completion time.

5.2. Results

The overall results for completion time are shown in Fig. 9. There was a significant main effect of menu type ($F_{5,50}=28.5$, $p<0.001$) on completion time. As expected, there was also a significant main effect of depth on completion time ($F_{2,20}=172.4$, $p<0.001$). In addition to the main effects, there was a significant menu type × depth interaction effect ($F_{10,100}=8.9$, $p<0.001$). As illustrated in Fig. 10, performance degraded more rapidly across depth with default and gesture-based menus than with the other techniques.

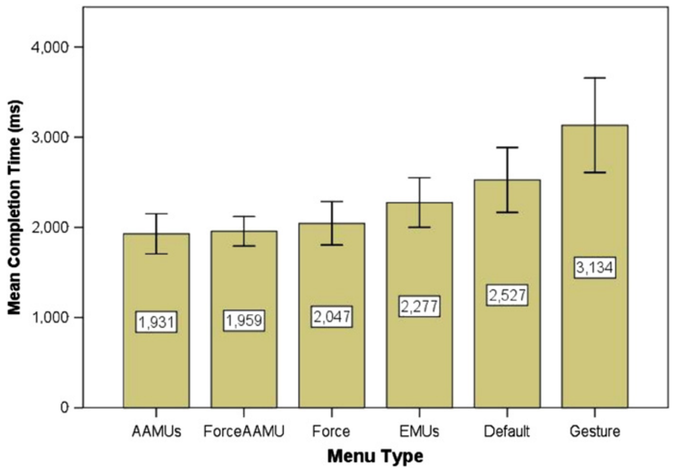

Fig. 9. Mean completion times for each menu type in Experiment 1. Error bars represent the 95% confidence interval ($N=11$).

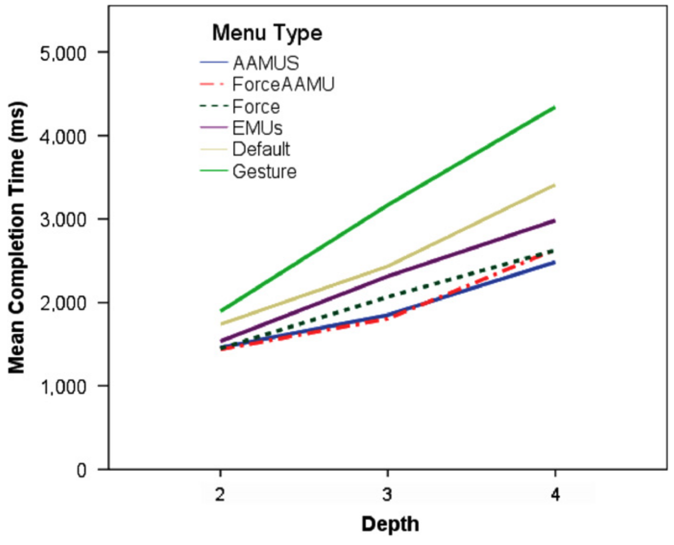

Fig. 10. An interaction graph showing the increase in completion time with increase in depth, for the different menu types in Experiment 1 ($N=11$).

Post-hoc pairwise comparisons, using a Bonferroni adjustment, revealed that, using a $p=0.05$ confidence level, AAMU (mean 1.93 s, sd 0.55 s) and force-AAMU (mean 1.95 s, sd 0.57 s) were significantly faster than EMUs (mean 2.28 s, sd 0.75 s), default (mean 2.53 s, sd 0.91 s) and gesture-based (mean 3.13 s, sd 1.36 s). Force-fields (mean 2.04 s, sd 0.62 s) were significantly better than default. There were no significant differences, however, between AAMUs, force-AAMU and force-fields. Gesture-based was significantly slower than all other menu types.

5.2.1. Subjective rankings

When asked about their preferences for the different menu types of a post-questionnaire, seven participants stated that AAMUs were their most preferred technique, while three preferred the force AAMUs. Only one participant stated that EMUs was his/her most preferred technique. Gesture-based menus were unpopular with our participants, with 10 stating it was their least preferred method. Users gave lower preference to EMUs due to the non-uniform activation area, which they found distracting and confusing. Those who did not prefer force-fields menus commented that they were more familiar with the standard speed of the mouse. The increased cursor acceleration, due to force-fields, made it feel as if the control was taken away. The majority of the users disliked the gesture-based menu on the basis that it interfered with the pace of interaction by forcing the user to change their direction of motion during posting/unposting.

5.3. Discussion

When the user knows the location of the target, our results show that AAMUs outperform traditional menus and EMUs, the latter of which also aims to improve performance by increasing the size of the activation area. Furthermore, AAMUs was the most preferred technique among the five techniques studied. The gesture-based menus performed the worst, both in terms of performance and users' subjective impressions. There were no performance differences between techniques involving force-fields and AAMUs; however, feedback obtained during the questionnaire revealed that force-fields were generally not well received. Furthermore, force-fields do not extend to all common input devices, such as the stylus. In our next study, we examine performance on a task more representative of a worst-case scenario, where the user is not familiar with the menu layout, and we look at the impact of different input devices on the efficiency of the AAMU technique.

6. Experiment 2: search and multiple devices

In many real-world scenarios, users will not know the location of the target item, often having to search through multiple cascading menus. In Experiment 2, we evaluate the performance benefits of AAMUs in this type of search task and examine the impact of three different input devices (mouse, touchpad and stylus).

6.1. *Method*

6.1.1. *Participants*

Twenty university undergraduate students participated in Experiment 2 in exchange for course credit. All had used the MS Windows default menu and were familiar with operating a mouse. Ten of the participants had prior experience using a touchpad and had used a stylus. None were color blind.

6.1.2. *Conditions*

To keep our experiment manageable for our participants, we tested three cascading menu techniques: default, AAMU, and EMU. We chose to focus our comparison on AAMUs and EMUs because they both involve area enhancements. Gesture-based menus were dropped from this study based on poor performance and lower subjective rankings in Experiment 1. While the force-fields had some support, we chose not to focus on force-based techniques in this study for two reasons. First, they can be used in combination with area-based techniques. Second, force-fields do not generalize to stylus-based input. We also tested three device types: mouse, touchpad and stylus.

6.1.3. *Task and stimuli*

Participants were required to perform 20 menu search tasks with each technique, at a fixed menu depth (level 3). Since it was a search task, no visual cue was provided for the path. The target menu item was displayed in red. Like in Experiment 1, a trial began when the participant clicked on the top-level menu heading and ended when the participant successfully acquired the target. Menu length was varied randomly (between four and nine items) in each level of depth in every trial with a constant cascading density of 50%. The positioning of the target item was determined randomly, but always appeared in the last menu depth level. For each trial, a different path and target were generated to prevent users from learning the trial path and positioning of the target item. At the start of the experiment, the participants were given 5 min of training with each menu type. Participants were instructed to complete tasks as quickly and as accurately as possible.

The order of technique presentation was counterbalanced using a Latin square, while device was randomized. With three menu types, one depth level, three devices and 20 trials per condition, the system recorded a total of 180 trials for each participant. The experiment took approximately 25 min per participant.

6.1.4. *Apparatus*

The experiment was conducted on Windows XP using a Pentium 4 machine with 1 GB of RAM and a 1024×768 monitor. The experiment was performed using an optical mouse, a touchpad and a stylus.

6.1.5. *Design*

The experiment employed a 3×3 repeated-measures design with factors menu type (default, AAMU and EMU) and input device (mouse, touchpad and stylus). The dependent variable was task completion time. We also measured the number of times the user clicked on a non-parent item other than the target, which we refer to as the number of extra “clicks”.

6.2. *Results*

The mean completion times with respect to menu type and device type are summarized in Fig. 11. There was a significant main effect of menu type ($F_{2,36} = 11.668$, $p = 0.001$) on completion time. Post-hoc pairwise comparisons, using the Bonferroni adjustment, showed EMUs (mean 4.87 s, sd 2.30 s) performing significantly faster than default (mean 5.30 s, sd 2.50 s, $p < 0.001$) and AAMUs (mean 5.26 s, sd 2.75 s, $p = 0.006$), whereas there was no difference among default and AAMUs ($p = 1.00$). There was also a significant main effect of device type ($F_{2,36} = 63.02$, $p < 0.001$) on completion time, but no interaction effect between menu type and device type ($F_{4,72} = 1.510$, $p = 0.208$).

In terms of the number of extra clicks, the effect of menu type was not significant ($F_{2,36} = 0.216$, $p = 0.807$). On average participants performed 0.13 extra clicks per trial with AAMUs (sd 0.38), 0.13 with EMUs (sd 0.38) and 0.15 with the default menu (sd 0.47). Thus, participants were not more error prone with any one technique. The effect of device type on the number of extra clicks was significant ($F_{2,36} = 5.412$, $p = 0.009$), with trends suggesting that participants made more extra clicks with the stylus (mean 0.35, sd 0.65) than with the touchpad (mean 0.03, sd 0.04, $p = 0.098$) or the mouse (mean 0.02, sd 0.04, $p = 0.093$). The

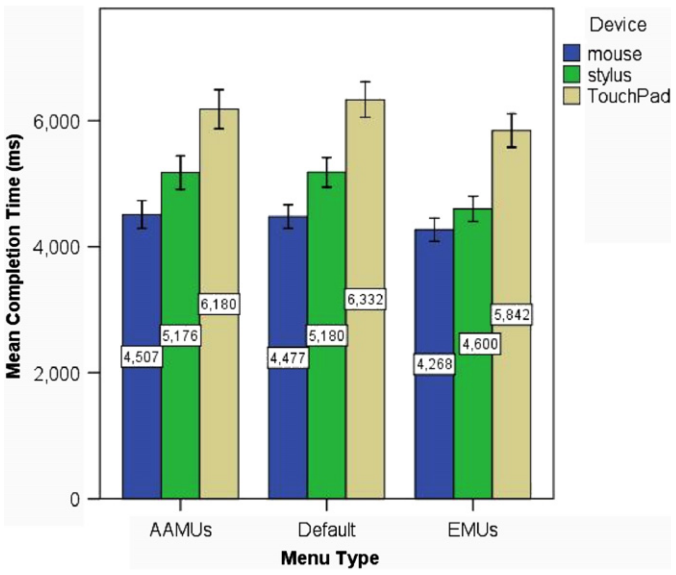

Fig. 11. Mean completion times for each menu type with respect to device type in Experiment 2. Error bars represent the 95% confidence interval ($N=20$).

interaction effect between menu type and device type, however, was not significant ($F_{4,72} = 0.007$, $p = 0.903$).

6.3. Discussion

In Experiment 1, an item selection task showed the promise of the AAMU technique. Experiment 2, however, which involved a searching task, revealed some potential design problems with the AAMU technique as it no longer outperformed default and EMUs. The lack of difference between AAMU and default suggests that some of the benefit of the wider activation area is lost when the user is exploring menus. In particular, when the wider activation area is fully expanded, it covers adjacent items in the parent menu and the user cannot activate the adjacent menu item immediately. We refer to this limitation as the "cursor trapping" problem since the user needs to "get out" of the activation area before entering the next menu item. A second potential problem, which could impact both select and search tasks, but might be more pronounced in a search task (because more menu navigation is required), is that users might not be taking full advantage of the diagonal movement path because of their prior experience with traditional cascading menus. We investigate both of these potential concerns in Experiment 3.

7. Experiment 3: investigating movement trajectories

To investigate the existence of the aforementioned problems, we next collected a number of movement paths from nine participants to examine how users interact with AAMUs both with and without the possibility of trapping. To observe "trapped cases" we manipulated cascading density, which is the percentage of cascading items within a single level of the menu hierarchy. We included trials with menus of minimal density, with only one cascading item present. Since in our experiment no trapping can occur in these cases, we refer to these trials as "clear". We also included trials where every item in the first level of the menu hierarchy was cascading (i.e., 100% cascading density), to ensure that trapping would occur. Full descriptions of Experiment 3's method and analysis can be found in Tanvir (2009). Here we summarize the types of movement patterns we observed.

7.1. Movement patterns in "clear" trials

Inspections of users' movement patterns revealed that, in general, users did begin to use the diagonal movement path offered by AAMUs after reminders during initial training. Despite the prevalence of traditional cascading menus, almost all users showed identical navigation patterns in case of clear trials, making use of the broader activation area and performing diagonal steering.

7.2. Movement patterns in trapped trials

In case of trials designed for trapping, we found that, as expected, the majority of these trials actually caused cursor trapping and users had to manoeuver the cursor out of the AAMU triangle to select the correct item. Our examination of navigation patterns revealed how disruptive cursor trapping can be. There were many patterns visible, from back tracking to extreme vertical (upward or downward) cursor movements.

Fig. 12 shows an example of the disruption. In this trial, there are two adjacent parent items (Parents 1 and 2), with the target located at position 5 in Parent 2's submenu. As seen in Fig. 12, the user started moving downwards in the parent menu and as soon as the cursor entered the boundaries of Parent 1, the respective AAMU was activated. Since the user's motion was diagonal, the cursor moved much inside the triangle before the user realized that it was the wrong item and now the only option left was to move the cursor outside the triangle to deactivate it. So the user backtracked all the way out of the AAMU triangle and as soon as the cursor entered the boundaries of Parent 2, the other AAMU activated and it lead to the target item.

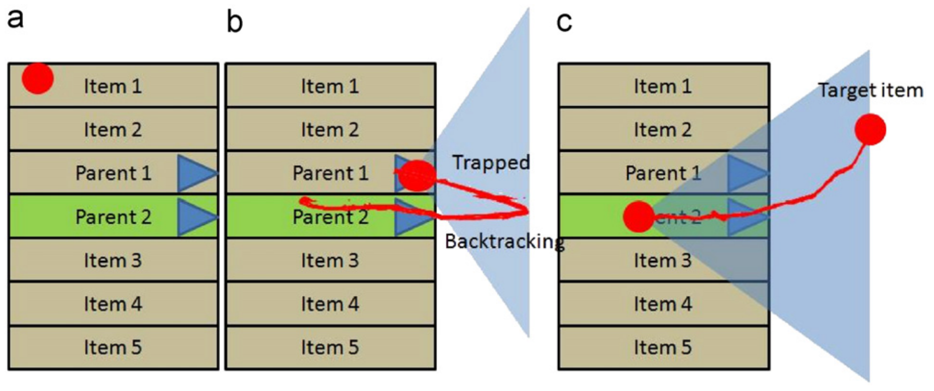

Fig. 12. Sample user data showing trapping and backtracking to get out of the "trap": (a) the initial cursor position, with the target within Parent 2; (b) when the cursor hovers over Parent 1, its corresponding AAMU gets activated. The user continues moving the cursor and gets trapped into Parent 1; (c) to get to Parent 2, the cursor moves out of the "trap" and then the Parent 2 AAMU is activated to complete the task.

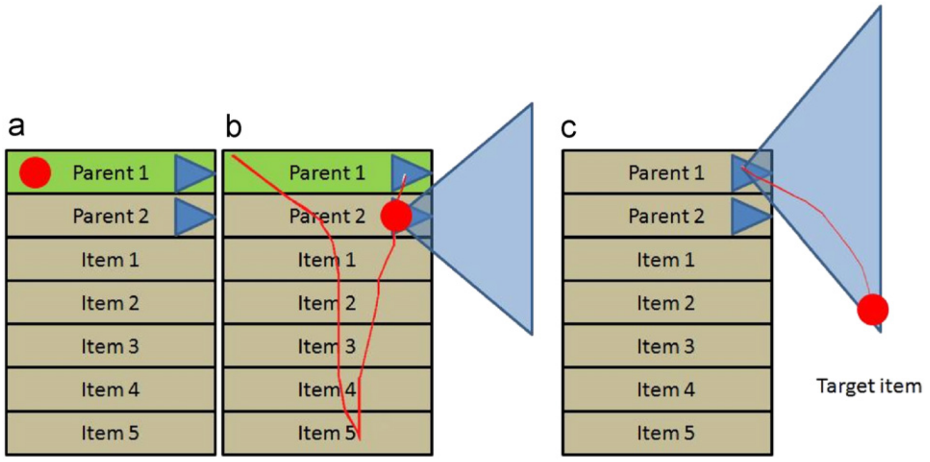

Fig. 13. Sample data where the user makes an exaggerated movement to avoid getting trapped: (a) the initial cursor position; (b) the user skips several items to avoid trapping; upon realizing that the cursor overshot considerably, the user moves back to Parent item 2, but the AAMU for Parent item 2 opens up; (c) finally the cursor is back at Parent item 1 and the user then moves to the final target item.

7.3. Trap avoidance

In addition to the difficulties users experienced when trapped in an AAMU, we also observed the development of coping strategies which negatively impacted future selections. In particular, after getting trapped a few times, almost all users would try to avoid trapping by making extreme vertical (downward or upward) movements. Fig. 13 shows a navigation pattern of trap avoidance even before any trapping occurred. In this trial, there are two adjacent parent items at the top of the parent menu (Parents 1 and 2). Since Parent 1 contained the target item, technically the user was never trapped while moving downwards into the item, but having experienced getting trapped in previous trials, the user quickly leaped downwards vertically, traversing six items before stopping and moving up again. On the way up, the user activated Parent 2's AAMU (but did not get "trapped") and kept moving the cursor upwards until Parent 1's AAMU was activated. The user then entered this newly activated AAMU and clicked on the target item.

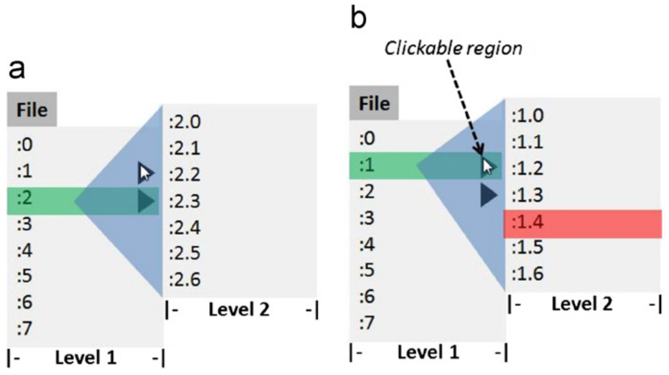

Fig. 14. Example of the AAMU-Click. (a) Before the click action: the cursor is trapped inside activation area of item 2, whereas the desired submenu is associated with item 1. (b) After the click action: while staying inside the old activation area, the user clicked on item 1 and activated it.

8. Developing alternate AAMU designs

The movement patterns illustrate some of the difficulties that users experience concerning cursing trapping with AAMUs. To lessen the performance impact of trapping, we investigated variations of the original AAMU technique that provide alternate paths or shortcuts out of cascading items. We explored the following factors: shape, visual cue and AAMU drawing position, which led to three alternative designs: AAMU-Click, AAMU-Curve and AAMU-Hover.

AAMU-Click appears identical to traditional AAMUs and it provides a shortcut path (click) to users to get out of the trap. AAMU-Click allows the users to continue interacting with other items in the menu while staying inside the AAMU triangle. A single click on any item makes the AAMU triangle disappear and activates this current item's function, as shown in Fig. 14.

AAMU-Hover, shown in Fig. 15, provides users with a visual cue to indicate an alternative path. When the mouse crosses over onto an adjacent item that is covered by the AAMU triangle, a small arrow appears inside the AAMU triangle. Hovering the cursor on this arrow activates this next item's function.

AAMU-Curve is a curved version of traditional AAMUs. Instead of an equilateral triangle, the legs joining the cursor position and the top and the bottom of the child submenu are drawn as curves. This curved triangle is also drawn a few pixels ahead of the cursor, so that the user can explore the child submenu without entering the triangle or getting trapped. Even in case of trapping, the narrow tip of the curved shape makes it easier for the user to get out of the triangle. An example of the AAMU-Curve is shown in Fig. 16(a).

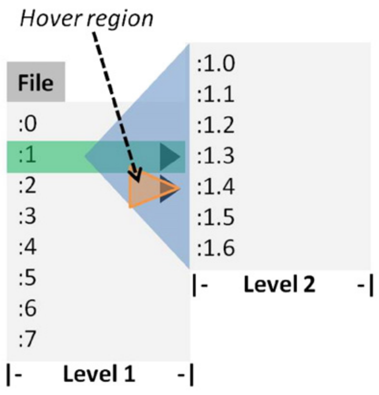

Fig. 15. A two-level deep menu showing the AAMU-Hover.

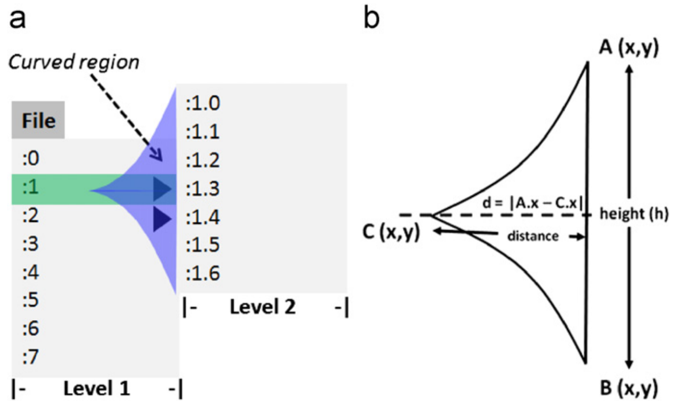

Fig. 16. The AAMU-Curve. (a) A two-level deep curve showing an example of the AAMU-Curve and (b) the geometry of the AAMU-Curve.

The geometry of the curved-AAMU shape is best explained using the parameters depicted in Fig. 16(b). The non-curved AAMU is simply a polygon drawn using three corner points: $C(x_c, y_c)$, $A(x_a, y_a)$ and $B(x_b, y_b)$, where the point C is the position of the cursor prior to invoking the submenu. Based on the cursor's distance to the submenu (d), and the menu's height (h), we define the following relations:

$$\textit{distance}\ (d) = |A \cdot x - C \cdot x|$$

$$\textit{height}\ (h) = |B \cdot y - A \cdot y|$$

We draw the curved-AAMU polygon in two parts, the upper half and lower half, using $h1 = h2 = h/2$. We use a quadratic function to draw the curved-shape polygon, based on $f(x) = kx^2$, where $k = h1/d^2$ and x is the x-value coordinate for each point from the cursor's position C to the submenu (defined by the vertical line (A,B)). The function used for k was derived through iterative refinements based on its perceived effectiveness in minimizing trapping. Similarly, we base the bottom part of the curve on $k = h2/d^2$.

We tested each of the above three alternative designs in a pilot study with 25 participants. The results of this study, which is described in detail in Tanvir (2009), suggested that AAMU-Curve and AAMU-Click both had potential to improve on the trapping problems and were preferable to AAMU-Hover. A major problem with the hover technique was that the arrow would appear on all diagonal movements, even when users were not actually trapped, at times causing unintended submenu invocations/revocations when the users accidentally hovered on the arrow while navigating. On the other hand, both AAMU-Curve and AAMU-Click showed promise: We found that the curve shape helped in the search task, whereas the click helped in the selection task. Therefore, for our final experiment, we combined the curve shape and the click functionality in an AAMU-Curve-Click design to leverage the respective advantages of both designs.

9. Experiment 4: putting the best designs to the final test

In our final experiment, we tested whether our new AAMU-Curve-Click technique could alleviate some of the trapping difficulties caused by our original design. We also sought to compare both the original and improved AAMUs to default cascading menus and to the EMU technique across a range of cascading densities and using different menu widths. Both cascading density and menu width influence the likelihood that trapping will occur with the AAMU design. For EMUs, cascading density impacts the size of the enlarged activation areas and hence their effectiveness (e.g., with high density, the activation areas shrink). We also tested the different cascading menu techniques using both search and select tasks.

9.1. Method

9.1.1. Participants

Thirty-nine undergraduate students participated in the experiment in exchange for course credit. All participants were experienced computer users, using mouse on a daily basis. None were color blind.

9.1.2. Conditions

Menus: Experiment 4 tested four menus types: AAMU-Curve-Click, AAMUs, EMUs and default.

Tasks: We tested two types of tasks: select and search.

Cascading density: Given that the techniques appear to be sensitive to cascading density, we surveyed commonly used applications to get a sense of typical density values. The applications surveyed included MS Word, MS Excel, MS Power Point, MS Internet Explorer, Mozilla Firefox, SPSS 16, MS Visual Studio 2005, Matlab, Adobe Photoshop and MS Outlook. Our survey revealed that cascading density varies between 0%

and 100%, with 20%, 50% and 80% being common values. Hence we tested these three density levels in our experiment.

Width: In addition to menu type, task and density, we included a fourth factor which was menu width (125 pixels vs. 195 pixels).

9.1.3. Tasks and stimuli

The experiment was conducted using the select and search tasks described in Experiments 1 and 2, respectively. While Experiment 2 used a menu depth of 3, in Experiment 4, menu depth was constant at 2, with 10 items in the first level and 15 items in the second level. We decided to use a depth of 2 in this experiment for two reasons. First, our survey of applications revealed depth 2 to be the more common cascading depth. Second, Experiment 1 showed that effects compound with increased depth. Therefore, to reduce participant fatigue, we chose a more pragmatic menu organization. Five target positions were tested: 2, 5, 8, 11 and 14. An example of a trial in this experiment is shown in Fig. 17.

A second change in Experiment 4 was to draw the menus in the center of the screen to enable center alignment in all scenarios to test the best-case AAMU design. We discuss the implications of this decision in Section 10.3.

Task order was counterbalanced, with 19 participants performing the search task first and 20 participants performed selection first. Within each task, the order of menu types was counterbalanced using a Latin square, while density and width were randomized. Participants completed a short practice block (10 trials) prior to each menu technique. With two tasks, four menu types, three densities, two widths and five trials per condition, there was a total of 240 trials per participant. The entire experiment lasted approximately 30 min.

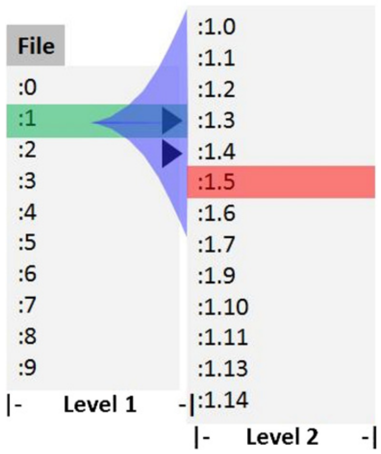

Fig. 17. An example of a task in the final experiment. A click on the "File" button activated the menu and a click on the target highlighted in red (item 1.5) ended the trial. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

9.1.4. Apparatus

The experiment was conducted on Pentium 4 desktop computers with 1 GB of RAM running the Windows XP operating system. Monitors with a 1024×768 resolution were used. A conventional optical mouse was used as the input device.

9.1.5. Design

Experiment 4 employed a $4 \times 2 \times 3 \times 2$ repeated-measures design with factors: menu type (AAMU-Curve-Click, AMU, EMU, Default), task (Search, Select), density (20%, 50%, 80%) and width (125 pixels, 195 pixels). Like in Experiment 2, the dependent variables were task completion time and the number of extra clicks.

9.2. Results

Out of the 39 participants, three completed only the first half of the experiment, in which all the three cases consisted of the select task.[2] Their data were not included in our analysis. Prior to analyzing the remaining data with an RM-ANOVA, outlier trials, defined as greater than three standard deviations from the mean completion time, were also removed representing 1.8% of all trials. While the total number of outliers is only a small percentage of our data, outlier removal resulted in us loosing an entire cell in our $4 \times 2 \times 3 \times 2$ repeated-measures design for six of our participants.[3] Thus, we had a full set of data (i.e., data in all experiment cells) for 30 participants (14 of which completed the search task first and 16 completed the select task first). All pairwise comparisons are corrected using the Bonferroni adjustment.

The RM-ANOVA revealed main effects of menu type ($F_{3,87} = 9.349$, $p < 0.001$), task ($F_{1,29} = 667.706$, $p < 0.001$), width ($F_{1,29} = 8.620$, $p = 0.006$) and density ($F_{2,58} = 252.897$, $p < 0.001$). There were also significant interactions between menu type $\times$ task ($F_{3,84} = 9.516$, $p < 0.001$), menu type $\times$ density ($F_{4.211,122.125} = 3.659$, $p = 0.007$),[4] and menu type $\times$ width $\times$ density ($F_{4.169,120.906} = 4.994$, $p = 0.001$).

Fig. 18 displays the overall means per menu type. Pairwise comparisons showed that AAMU-Curve-Click (mean 2.05 s, sd 0.28 s) was significantly faster than AAMUs (mean 2.16 s, sd 0.27 s, $p = 0.003$), default (mean 2.25 s, sd 0.31 s, $p = 0.008$) and EMUs (mean 2.16 s, sd 0.27 s, $p < 0.001$). AAMUs were significantly better than default ($p = 0.033$) as were EMUs ($p = 0.016$). No other pairwise comparisons resulted in significant differences.

Like in Experiment 2, the impact of menu type on the number of extra clicks was not significant ($F_{1.833,53.156} = 0.134$, $p = 0.858$). Participants made an average of 0.03 extra clicks per trial with AAMU-Curve-Click (sd 0.10),

[2]For two participants there was a procedural error that caused them to mistakenly complete the same task twice. The third participant asked to leave after completing the first task, likely because of boredom.

[3]The missing cells were distributed across the four menu types.

[4]The Greenhouse–Geisser adjustment is used for non-spherical data.

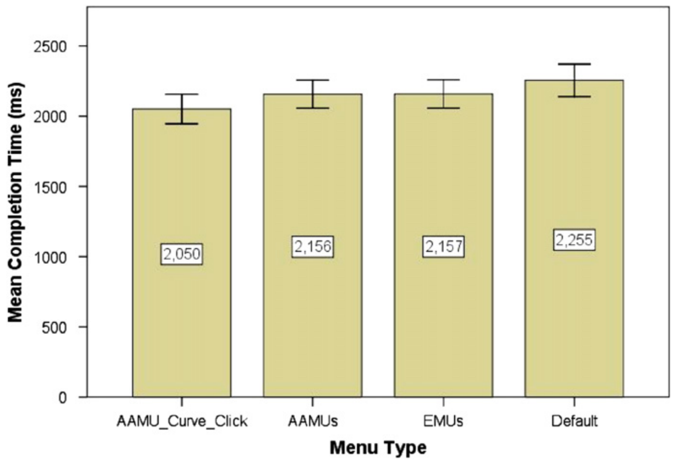

Fig. 18. Mean completion times for all menus over both tasks. Error bars represent the 95% confidence interval ($N=30$).

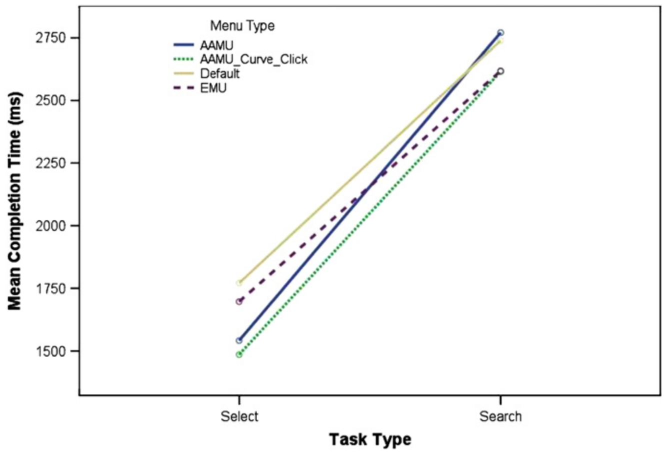

Fig. 19. The interaction between menu type and task ($N=30$).

0.03 extra clicks with AAMUs (0.11), 0.03 extra clicks with EMUs (sd 0.12) and 0.03 extra clicks with the default menus (sd 0.10). Apart from a significant effect of cascading density on the number of extra clicks ($F_{2,58}=6.046$, $p=0.003$), where participants performed more extra clicks with cascading density 80 (mean 0.05, sd 0.11) than with density 20 (mean 0.02, sd 0.09, $p=0.003$) or with density 50 (mean 0.02, sd 0.11, $p=0.024$), there were no other significant effects or interactions for this dependent measure.

We next examine the nature of the two-way interactions between menu type × task and menu type × density on completion time. These two-way interactions provide a more in depth understanding of the relative merits of the different cascading menu techniques. This is followed by a look at the three-way interaction between menu type, task and width.

9.2.1. Impact of task type

Fig. 19 illustrates the nature of the two-way interaction effect between menu type and task on completion time. Pairwise comparisons for each task revealed the following significant differences or trends:

Select: AAMU-Curve-Click (mean 1.49 s, sd 0.21 s) was significantly faster than EMUs (mean 1.67 s, sd 0.28 s, $p<0.001$) and Default (mean 1.77 s, sd 0.27 s, $p<0.001$). AAMUs (1.54 s, sd 0.25 s) were significantly faster than EMUs ($p=0.01$) and Default ($p=0.001$).

Search: There were trends suggesting that AAMU-Curve-Click (mean 2.61 s, sd 0.42 s) was faster than AAMUs (mean 2.77 s, sd 0.37 s, $p=0.08$), and that AAMU-Curve-Click was faster than EMUs (mean 2.62 s, sd 0.35 s, $p=0.08$). No other pairwise comparisons were significant, nor there were other differences with trend-level support.

To summarize the above comparisons, AAMU-Curve-Click was significantly faster than all other techniques for the select task, while AAMUs continued to outperform EMUs and Default. For the search task, there were trends suggesting that AAMU-Curve-Click improved upon the original AAMU design and was faster than the EMU technique.

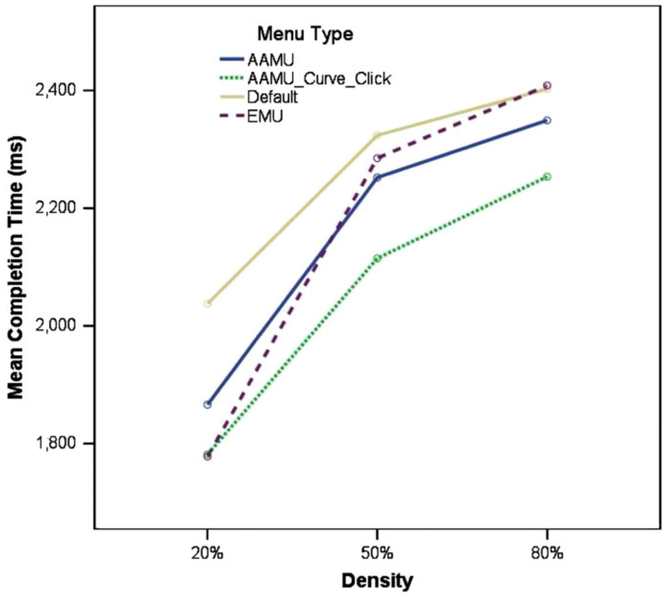

Fig. 20. The interaction between menu type and cascading density ($N=30$).

9.2.2. Impact of cascading density

Fig. 20 illustrates the nature of the two-way interaction effect between menu type and task on completion time. Pairwise comparisons for each cascading density level revealed the following:

Cascading density 20: Default (mean 2.00 s, sd 0.39 s) was significantly slower than all techniques (AAMU-Curve-Click: mean 1.81 s, sd 0.34 s, $p<0.001$; AAMUs: mean 1.85 s, sd 0.30 s, $p<0.001$; EMUs: mean 1.81 s, sd 0.26 s, $p=0.046$).

Cascading density 50: AAMU-Curve-Click (mean 2.13 s, sd 0.32 s) was significantly faster than AAMUs (mean 2.26 s, sd 0.33 s, $p=0.047$) and Default (mean 2.31 s, sd 0.43 s, $p=0.013$).

Cascading density 80: AAMU-Curve-Click (mean 2.22 s, sd 0.35 s) was significantly faster than EMUs (mean 2.41 s, sd 0.37 s, $p=0.008$) and Default (mean 2.33 s, sd 0.35 s, $p=0.034$).

To summarize the above comparisons, at low cascading density, all techniques outperformed Default but were otherwise equivalent. At medium density, AAMU-Curve-Click significantly outperformed AAMUs and Default, and was similar to EMUs. At high density, AAMU-Curve-Click outperformed EMUs and Default.

9.2.3. *Impact of menu width and cascading density*

Finally, Fig. 21 displays the nature of the three-way interaction between menu type, cascading density and menu width on completion time. While AAMU-Curve-Click consistently outperformed AAMUs and Default, its relationship with EMUs was more dependent on the width and density. It began to outperform EMUs at higher densities, particularly as the menu width increases.

9.3. *Discussion*

When collapsing the data across task, AAMU-Curve-Click performed significantly better than traditional menus, AAMUs, and the EMU technique in terms of task completion times. Further decomposition of the data provided insight into the strengths and limitations of the improved AAMU design.

When examining the data according to task, we found that AAMU-Curve-Click was significantly faster than traditional menus and EMUs in the selection task, where the user knows the location of the target. There was no difference between AAMU-Curve-Click and AAMUs for this task, which can likely be explained by the fact that trapping is unlikely to occur when the user has to open only a single submenu. In a search task, there was trend-level support that AAMU-Curve-Click is faster than both the original AAMU design and EMUs. These results suggest that the improvements to the basic AAMU design do help mitigate the trapping problem first identified in Experiment 2; however, additional data are necessary to verify these trends.

Our results also indicate that the ratio of parent items to non-parent items impacts the relative efficiency of the different techniques studied. At low cascading density, all three techniques (AAMUs, AAMU-Curve-Click and EMUs) improved upon traditional cascading menus. The specific benefit of the AAMU-Curve-Click design came at medium and high densities. At medium densities, AAMU-Curve-Click significantly improved upon AAMUs and traditional menus, and was as fast as EMUs. At high densities, AAMU-Curve-Click outperformed EMUs, most likely due to the fact that the EMU activation areas have less room to expand.

10. Discussion and future work

Our studies systematically reveal the strengths and weaknesses of our new menu navigation technique, adaptive activation area menus (AAMUs). Based on our results and observations, we Highlight the effective value of AAMUs, discuss some of the limitations inherent with the AAMU design and discuss limitations of our experiments. We then present a set of possible extensions.

10.1. *Effective value of AAMUs*

Selecting submenu items with conventionally designed menus are prone to narrow and corner steering problems. The Steering law (Accot and Zhai, 1997) dictates that narrow steering can reduce the efficiency of moving a cursor through a path. More importantly, very narrow steering can be frustrating in that users tend to steer off-path, causing submenus to be unintentionally deactivated. To avoid the pitfalls associated with narrow and corner steering, we designed AAMUs to allow users to activate a submenu by moving diagonally from the parent item to the

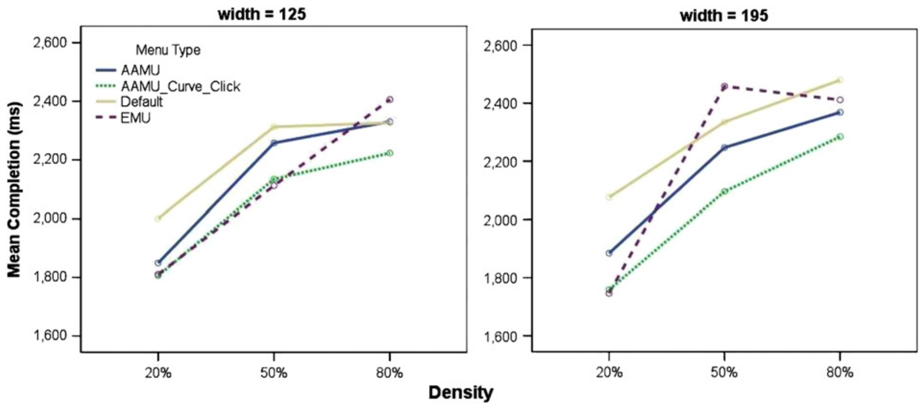

Fig. 21. The three-way interaction effect of menu type × density × width on completion time.

submenu item. This design is inspired by some of the existing solutions for minimizing the impact of narrow and corner steering toward submenu items, namely EMUs (Cockburn and Gin, 2006).

While we were not aware of this work when designing the AAMUs or conducting our experiments, Tsandilas and schraefel (2007) discuss an invisible triangular activation region present in the Mac OS menus. Thus, the idea of a triangular activation area is likely not new. The extent of the similarity between AAMUs and Mac OS menus, however, is unclear as we have not able to find the details of the Mac OS design. There are also no reported evaluations of the Mac OS technique or comparisons with other techniques.

Results from several experiments described in our paper show that under a variety of conditions, AAMUs are either more efficient than or at least as good as EMUs and traditional menus. Our first experiment also shows when the user knows where the target is located, AAMUs are faster than gesture-based menus (Kobayashi and Igarashi, 2003) and as fast as force-fields (Ahlström, 2005). The relative merits of AAMUs and force-based techniques, however, is an area for future exploration, as is combining AAMUs and force-based techniques given that they are not mutually exclusive.

In addition to being efficient, AAMUs are easy to implement. They can override the properties of existing menu designs simply by overlaying an activation area that is transparent and that takes control of the cursor movement. The overlaid activation area can also be easily adjusted in size, based on both the cursor's position as well as the length of submenu lists. We envision AAMU-style menus being developed as user interface controls for various development environments. This way, designers could abstract away from the minutia design elements and simply apply AAMUs to the application context.

Bringing out the true value of AAMUs requires that they be deployed in real-world settings. This can be most easily taken on by implementing AAMUs in environments such as Web interfaces, games or on novel platforms that have not been designed with any specific menu styles or menu guidelines. These environments can be more liberal in their choice of menu design and select the AAMU.

10.2. AAMU limitations

While AAMUs support submenu navigation and interaction, they are also limited in a number of ways. In particular, AAMUs can be affected by trapping and organization of menu content.

Results and observations of our earliest experiments reveal that our basic AAMU design is prone to creating unwanted/unexpected cursor behaviors. In particular, we found that participants would have their cursor "trapped" if an AAMU was unintentionally activated. Our AAMU-Curve-Click design helps to resolve some of the trapping issues. Nonetheless, we believe that some amount of trapping will always be present. The overlay will continue to cover other menus items, and it will not be entirely possible to predict whether the AAMU was accidentally opened or not. As a result, users might experience a slight learning curve when first exposed to AAMUs in their interfaces. This was indeed observed in Experiments 1 and 2, where a certain amount of training was required to allow users to get familiar with the AAMU design. One solution to minimize the effects of trapping is to provide a dynamic activation area that grows and shrinks based on the number of "off-the-path" crossings. We discuss this solution in further detail below.

As seen from our results in Experiment 4, when menus have a large number of adjacent parent menus, more trapping occurs and thus makes the AAMU interface less efficient. Therefore, some consideration is needed when assigning a position to a parent menu item within a menu hierarchy, a constraint that does not exist with traditional menus. While we have not fully studied the impact of reorganizing parent items, our results hint at the potential gains when such an organization is created.

10.3. Experiment limitations

While our experiments provide insight into the strengths and weaknesses of the AAMU technique, we acknowledge their limitations. First, we had only undergraduate student participants, which could impact the generalizability of our results to a less homogeneous user population. Second, in our searching task, users could not use the names of the menus to guide their search. As a result, users might have performed more menu navigation in this "worst-case" task than they would have in a more realistic setting. Third, in Experiment 4, we lost data from a number of participants (six out of 36) following a standard outlier procedure. Although the total number of outliers was reasonable (1.8% of all trials), they were concentrated within certain menu/task/density/width configurations for some of our participants. The outliers were spread out across the different menu types tested; however, further investigation would be needed to determine if there is any particular cause for these types of difficulties. Finally, in Experiment 4, we arranged the menus so that center submenu alignment was always possible, reducing the ecological validity of our study. Further studies are needed to verify that the benefits of the improved AAMU design over EMUs and traditional menus persist when a mix of center and top alignment are used. We do not, however, expect the results to change drastically in a more ecologically valid setting given that the total triangle area would remain the same, with only its shape changing.

10.4. AAMU extensions

We explored a range of alternative AAMU designs, some of which help to alleviate existing problems such as trapping. In this section we present other possible extensions.

10.4.1. *Truly adaptive?*

Our studies have shown that alternative AAMU designs, such as AAMU-Curve-Click, can minimize the effects of trapping. While they are superior to the original AAMU (and therefore conventional menus), some trapping is still possible, suggesting that there is further room for improvement. Currently, AAMUs contain both adaptive and static components: the shape is fixed as a curve, while the drawing position and size are adapted to the users' cursor position on the parent menu item and the size of the child submenu. However, there are no adaptations based on users' movements/menu navigation patterns. One possible alternative would be to design an intelligent version of AAMUs that adapts both the AAMU shape and size according to the users' movement patterns. The intelligent interface could continue to learn how users navigate with submenus to find the optimal shape and size. It could be possible that with more acute angles on the AAMU-Curve, less trapping happens. For some users, a wider activation area may be ideal, whereas for others, a narrow shape might work better.

10.4.2. *Extending AAMUs to different menu styles and environments*

AAMUs were designed for current linear styles of menus. An area of future investigation would be to apply AAMUs to other types of menus, for instance marking menus (Kurtenbach and Buxton, 1994), pie menus (Callahan et al., 1988). These other menu style might also suffer from narrow and corner steering gestures.

AAMUs were also designed with conventional desktop environments in mind. However, such menus have also been ported onto various other platforms, such as mobile devices or large multi-touch or wall-sized displays. These other environments typically do not provide users with a cursor or mouse, and are typically constrained to input based on hand gestures and direct pointing. There is a reason to believe that steering in such environments might be even more difficult. For instance, on large displays, the cursor is affected by hand jitter, causing involuntary hand gestures. It is possible that AAMUs could alleviate some of the impact of jitter in such environments. Similarly, pointing on mobile devices is suspect to the 'fat finger problem', potentially causing users to steer-off onto unintended paths. AAMUs could be potentially redesigned to provide an activation area that would be less affected by such input constraints.

11. Conclusion

Menus and submenus are common interface elements. They allow designers to group commands based on function and intent. Numerous menu designs have been proposed, but current menu designs can be frustrating to interact with. In particular, hierarchical conventional menus are prone to inadvertent submenu inactivation, particularly as users steer cursors off the menu paths. Performance with current submenu selection is governed by the Steering law (Accot and Zhai, 1997), which states that movement time to the target of interest is linearly proportional to the ratio of the length and width of parent menu items. By removing the constraint to steer through narrow parent menu items, the performance of submenu selection can be improved. Our novel design, the AAMU, is based on this assumption, i.e., removing the constraints of narrow and sharp corner steering down a parent menu path. Our experimental results support this novel design and show that there are performance improvements with the AAMU.

Primary user studies showed that when it came to item selection, AAMUs outperformed traditional menus and its closest competitor, EMUs, a technique that also enlarges the activation area. AAMUs performed as well as or better than techniques that aim to reduce target distance. For a task involving more menu searching, AAMUs suffered from the "cursor trapping" problem, where the wider activation area hindered the search process. To address this issue, we designed an AAMU variant called AAMU-Curve-Click, which reduces the total area of the activation region while still permitting diagonal steering and provides a means of quickly accessing items under the overlay. Results of the final experiment showed that AAMU-Curve-Click was better than or at least as good as both traditional cascading menus and EMUs across a range of scenarios. The new technique showed the greatest benefits in scenarios more prone to cursor trapping problems, such as wider menus and higher cascading densities.

Overall, there are many future directions to take and improving the AAMU design to work in various environments is one of our primary priorities.

References

Accot, J., Zhai, S., 1997. Beyond Fitts' law: models for trajectory-based HCI tasks. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'97), pp. 295–302.

Accot, J., Zhai, S., 1999. Performance evaluation of input devices in trajectory-based tasks: an application of the steering law. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'99), pp. 466–472.

Ahlström, D., 2005. Modeling and improving selection in cascading pull-down menus using Fitts' law, the Steering law and force-fields. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'05), pp. 61–70.

Ahlström, D., Alexandrowicz, R., Hitz, M., 2006. Improving menu interaction: a comparison of standard force enhanced and jumping menus. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'06), pp. 1067–1075.

Bederson, B.B., 2000. Fisheye menus. In: Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'00), pp. 217–225.

Callahan, J., Hopkins, D., Weiser, M., Shneiderman, B., 1988. An empirical comparison of pie vs. linear menus. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'88), pp. 95–100.

Cockburn, A., Gin, A., 2006. Faster cascading menu selections with enlarged activation areas. In: Proceedings of Graphics Interface (GI'06), pp. 65–71.

Fitts, P., 1954. The information capacity of the human motor system in controlling the amplitude of movement. Journal of Experimental Psychology 47, 381–391.

Kobayashi, M., Igarashi, T., 2003. Considering the direction of cursor movement for efficient traversal of cascading menus. In: Proceedings of the ACM Symposium on User Interface Software and Technology (UIST'03), pp. 91–94.

Kurtenbach, G., Buxton, W., 1994. User learning and performance with marking menus. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'94), pp. 258–264.

MacKenzie, I.S., 1992a. Fitts' law as a research and design tool in human–computer interaction. Human–Computer Interaction 7, 91–139.

MacKenzie, I.S., 1992b. Movement time prediction in human–computer interfaces. In: Conference on Graphics Interface (GI'92), pp. 140–150.

Pastel, R., 2006. Measuring the difficulty of steering through corners. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'06), pp. 1087–1096.

Tanvir, E., 2009. Improving cascading menu selections with adaptive activation areas. Master's Thesis, University of Manitoba, Department of Computer Science.

Tanvir, E., Cullen, J., Irani, P., Cockburn, A., 2008. AAMU: adaptive activation area menus for improving selection in cascading pull-down menus. In: Proceeding of the ACM Conference on Human Factors in Computing Systems (CHI'08), pp. 1381–1384.

Tsandilas, T., schraefel, m.c., 2007. Bubbling menus: a selective mechanism for accessing hierarchical drop-down menus. In: Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'07), pp. 1195–2004.