Dynamic Context Switching for Gaze Based Interaction

Antonio Diaz Tula*
University of São Paulo, Brazil

Filipe M. S. de Campos†
University of São Paulo, Brazil

Carlos H. Morimoto‡
University of São Paulo, Brazil

Abstract

This paper introduces Dynamic Context Switching (DCS) as an extension of the Context Switching (CS) paradigm for gaze-based interaction. CS replicates information in each context. The user can freely explore one context without worrying about the Midas touch problem, and a saccade to the other context triggers the selection of the item under focus. Because CS has to display two contexts simultaneously, the amount of useful screen space is limited. DCS dynamically adjusts the context sizes, where the context that has the focus is displayed in full size, while the other is minimized, thus improving useful screen space. A saccade to the minimized context triggers selection, and properly readjusts the sizes of the contexts. Results from a pilot user experiment show that DCS improves user performance and do not cause disorientation due to the dynamic context resizing.

CR Categories: H.5.2 [Information Systems]: INFORMATION INTERFACES AND PRESENTATION—User Interfaces H.1.2 [Information Systems]: MODELS AND PRINCIPLES—User/Machine Systems

Keywords: dynamic context switching, gaze based interaction, selection by gaze

1 Introduction

Perhaps the most commonly used method to make selections in gaze-based interfaces is dwell time. This approach use only fixations, hence is very simple and easy-to-use, but its main limitation is the Midas touch problem [Jacob 1990]. Alternatives to dwell time are continuous gaze gestures [Hansen et al. 2008; Ward and MacKay 2002] and discrete gaze gestures [Huckauf and Urbina 2008; Wobbrock et al. 2008].

In [Stellmach et al. 2011] a touch-and-tilt device is used to make selections along with a fisheye lens to enhance pointing, thus eliminating dwell-time activation. [Kumar et al. 2007] proposed a *look-press-look-release* action to make a selection more fluid and accurate. More recently, [Huckauf and Urbina 2011] introduced an alternative method using antisaccades for selection.

Context Switching (CS) [Morimoto and Amir 2010] is an activation mechanism for gaze controlled interfaces that was suggested as an alternative to dwell time. The method consists of two identical regions called ”contexts”. To make a selection the user needs to focus on the desired key within one of the contexts and saccade to the other context. This method is comfortable and easy to learn due to the use of a short fixation for focus and a single saccade for selection, which are natural eye movements, thus eliminating the Midas touch problem. In addition, the users can naturally adjust their selection speed without the use of an explicit button. Compared to dwell time, CS clearly separates focus and selection, associating focus to eye fixations and selection to saccades.

One major limitation of CS is that because two contexts must be displayed at all times, useful screen space is limited. This paper introduces an extension to traditional CS: Dynamic Context Switching (DCS). The purpose of DCS is to increase the useful screen space by dynamically adjusting the size of the contexts. We conducted a pilot user study to evaluate CS and DCS in terms of performance for selection tasks.

The rest of the paper is structured as follows. Section 2 describes DCS in more detail. Section 3 describes the experimental design. Section 4 shows the experimental results and discussions, and Section 5 concludes the paper.

2 Dynamic Context Switching

In CS objects are grouped within areas called contexts. When the user looks at an object for a short time (typically 150 ms) the object receives the focus. The selection of the key in focus is made by saccading to the other context. To avoid unintentional switch of contexts due to the noise of the eye tracker, there is a space in between the contexts called *bridge*. A saccade must completely cross the bridge between the two contexts to make a selection.

A limitation of the method, that is also shared among other gaze based interaction techniques such as [Ward and MacKay 2002; Hansen et al. 2008], is the limited screen space left to display other useful information. In [Morimoto and Amir 2010], the bridge has been used to display the typed text. Because the technique requires two contexts, each context must also be limited in size, since small virtual keys are not appropriate for use with eye trackers.

The screen can be better used if the contexts can be resized. The basic idea is that the context that receives the focus is maximized, while the other context can be minimized, so the whole screen can, in principle, display one context at a time. Because resizing may cause disorientation, the contexts do not need to be fully maximized or minimized, but their sizes can be adjusted dynamically, so that the context that has the focus is bigger than the other one, thus having more useful space than with fixed size contexts. As soon as the user switch contexts, the size of the contexts must to be properly updated.

Applications can benefit from DCS by presenting more items on the screen than it would be possible with traditional CS. An example is to have more keys in a DCS-based virtual keyboard, including punctuation, numeric keypads and/or symbols. Another advantage of DCS is that, because the keys can be made bigger, the interaction becomes more robust to gaze tracking errors if both CS and DCS have the same number of keys.

One concern about DCS is if it causes disorientation due to the dynamic resizing, so that it could be slow or cause fatigue. To evaluate the benefits of DCS regarding speed and user experience, we designed and conducted a longitudinal pilot experiment, described in the next section.

*e-mail: diaztula@ime.usp.br
†e-mail: fmsc@ime.usp.br
‡e-mail: hitoshi@ime.usp.br

ETRA 2012, Santa Barbara, CA, March 28 – 30, 2012.
© 2012 ACM 978-1-4503-1225-7/12/0003 $10.00

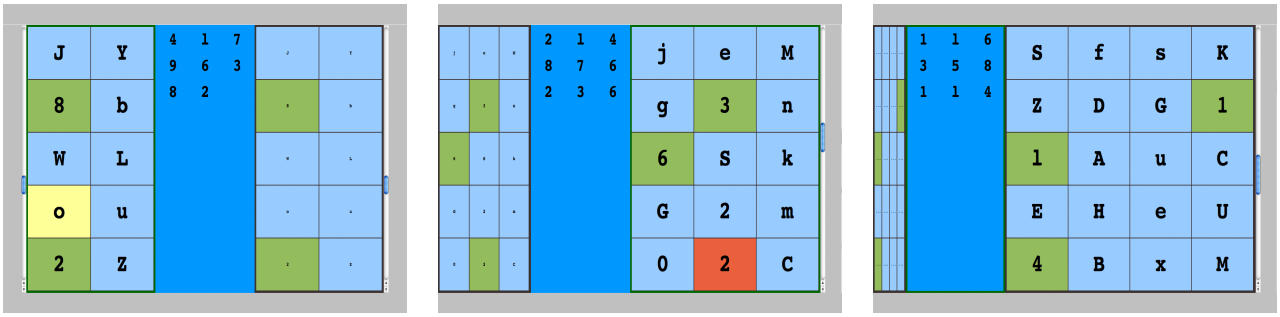

Figure 1: *2, 3, and 4 column layouts used in the experiment.*

3 Experimental Design

3.1 Space vs Speed

Assume that a context has the same density of keys per screen area (in general defined by the accuracy of the eye tracker). If a DCS context has twice the area of a CS context then it would hold twice as many keys/objects. One way to make a CS-based interface with the same number of keys is to have a paging mechanism.

To compare the speed of CS and DCS using the same number of objects, we used a task consisting of searching and selection of items within a large collection, common in any search application such as image retrieval (Flickr and Picasa), web search (Google), file browsers (Finder) etc. Because we are interested in measuring performance, we defined a task to select digits (numeric characters) from a set of alphanumeric characters (lower and upper case letters from the English alphabet). This task helps to keep the cognitive load to a minimum so that the participants could focus on the interaction, and allows a fair comparison of the methods using large collections of data, since it requires the users to browse across several pages to complete the task.

We used three different methods for the experiment: a traditional CS layout with 2 columns (2C), and two DCS layouts with 3 and 4 columns (3C and 4C) respectively, all having 5 rows. The size of the keys were kept the same for all layouts. This way we guarantee that the accuracy of the eye tracker interferes with the speed of all layouts about the same way. Nonetheless, the size of the keys were large enough to avoid noise in key focusing and selection. The size of the bridge was kept the same for all layouts to guarantee a minimum saccade distance to all methods and layouts. Figure 1 shows the 3 different layouts that were developed for the experiment.

The task consisted of selecting a random number of digits uniformly distributed within the interval [18, 28] from a collection of 120 characters. This number of characters can be displayed using 12 full pages for 2C, 8 pages for 3C and 6 pages for 4C. Because the number of digits is random for each trial, users are always required to browse all pages.

Simple gaze gestures were used to browse pages. Markers displayed around the contexts were used to activate page up and page down. A gesture consisted of looking from the context to the marker and then back to the context, similar to [Ohno 1998].

3.2 Experimental Protocol

A total of 6 people participated in our pilot experiment, all male, able-bodied, with normal or corrected to normal vision. Two participants had no previous experience with eye trackers, two had little experience with other gaze based applications, and the other two had large experience. Before the first session the participants were interviewed and introduced to the experiment, and signed a consent form. A training session, about 10 minutes long, was performed by all users in order to learn the task and operate the gaze interface.

All volunteers participated in 6 sessions that last about 15 minutes each. In every session, the participant had to perform 9 trials, 3 for each layout. In every trial, the user was told to select the digits as fast as possible, and to be careful not to leave digits unselected. The order of the layouts presented to the user at each session changed randomly. A session could not be repeated within 30 minutes, so that most volunteers took 2 or 3 days to complete their sessions.

If a participant lost calibration during a session, results of that trial were discarded and the user repeated the trial. Still there where cases when a trial could not be repeated because of the participant's schedule, so fewer trials were considered for those participants in the data analysis. At the end of the experiment participants were interviewed and answered a questionary about their impressions of the interaction using the three layouts.

3.3 Implementation Issues

A low-cost pupil-corneal reflexion remote eye tracker was used during the experiments. The eye tracker runs at 30Hz and has about 1^o accuracy. A short dwell time of 150ms was used for detecting focus on a virtual key, and the maximum time for selection by context switching (i.e. maximum duration for the saccade) was set to 450 ms for 2C and 3C, and to 550 ms for 4C (because on average 4C requires a longer saccade).

Users could select and deselect items, so it was easy to correct any wrong selection at any time during the experiment. Deselection procedure is similar to regular selection: just focus on a selected item and switch to the other context.

Selections (and deselections) are made using horizontal saccades, that are faster and more natural than vertical ones. Visual feedback is provided for the context that has the user focus by displaying a green border. Regular keys (characters in a context that could be selected) were painted light blue, indicating that they were not selected. After selection, the selected keys were painted green in both contexts. A blue key turns yellow when it receives the focus (gaze), indicating that it can be selected, and a green key turns orange when it receives the focus, indicating that it can be deselected.

The bridge is used to list the current selected items. When the user looks at this area for at least 500 ms, it gets the focus (as in Figure 1 4C). In this case, no selection is made when the focus returns to the left or right context. In our current implementation, no action is permitted within the bridge.

3.4 Data Analysis

To compare the selection speed of CS and DCS we separate the time users dedicated to selection from the time spent for paging. For every page in a task we consider the *selection time* from the moment the user enters a page to the last selection within that page, and the *paging time* as the remaining time from the last selection to the execution of a paging action. Because each page could have a different (random) number of digits, for every page we computed the *selection time per digit (STPD)* as the *selection time* divided by the number of actual selections on that page.

To compare the overall user performance per task, that includes the selection and paging time, we computed the *total time (TT)* as the time from the beginning of the task to the last selection in the last page.

For a given trial, let $nPages$ be the number of visited pages, TP the number of digits actually selected (true positives), FP the number of non-digits selected (false positives), and FN the set of missing digits (false negatives). To evaluate the users performance for each method the following metrics are used:

- Precision: $P = TP/(TP + FP)$
- Recall: $R = TP/(TP + FN)$
- Average selection time: $AST = \frac{1}{nPages} \sum STPD$
- Average task time: $ATT = TT/(TP + FP)$

Precision and recall tell us how careful the user was at the selection task. A precision under 100% reveals the selection of a few letters, while a recall under 100% reveals that the user missed some digits.

AST is the mean of the $STPD$ for all pages. Pages with no selection were not taken into account. AST measures how fast a person can make a selection, independently of the number of pages.

ATT is the average task time considering the time for paging. ATT measures how fast the task can be completed, including selection and paging. We expect that selection is faster using simpler layouts, but the overall completion time may be better using more complex layouts.

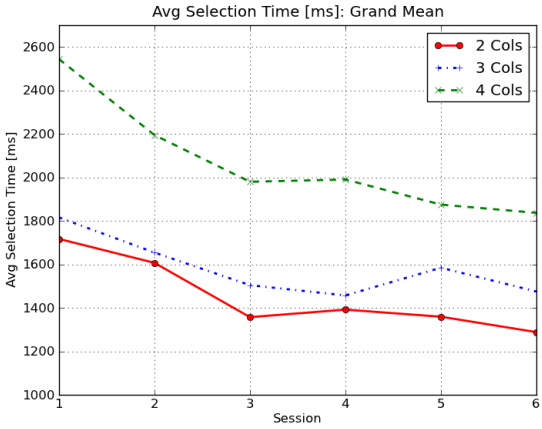

Figure 2: *Grand mean for Average Selection Time (AST).*

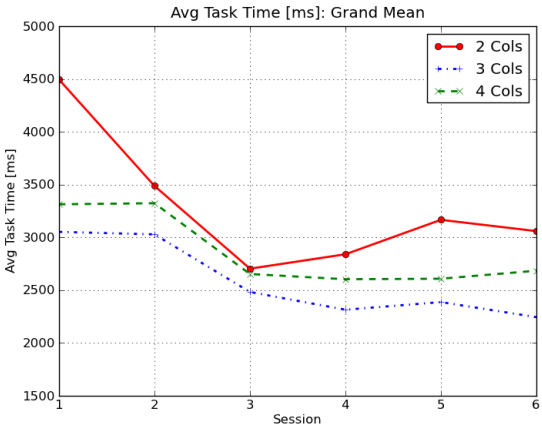

Figure 3: *Grand mean for Average Task Time (ATT).*

4 Experimental Results

Table 1 shows the results of the grand mean of P and R for all users, for the 3 methods over the 6 sessions. Figures 2 and 3 show, respectively, the results of the grand mean of AST and ATT for all users, for the 3 methods evaluated in the 6 sessions.

4.1 Discussion

The results from the Table 1 of P and R show that the participants were equally careful for all methods during the experiments. In particular, the grand mean for P and R are similar for all methods in all sessions. Observe that most of the precision values are very close to 100%, i.e., most of the time only digits were selected. The most common selected letter (false positive) was the lowercase letter "a", two times by users 2 and 6 and one time for user 4, respectively. We have expected the letter O to be mistaken by a zero, but that did not happen.

Table 1: *Grand mean for Precision and Recall for all sessions*

		Session					
	Method	1	2	3	4	5	6
P	2C	99.1	98.1	99.4	99.3	98.7	99.4
	3C	98.1	99.5	99.4	99.5	99.8	99.3
	4C	98.8	99.3	99.4	100	99.4	99.7
R	2C	97	96.6	98.2	97	98.7	98.1
	3C	94.4	97.3	97.2	98.2	98.9	97.1
	4C	97.6	94	98.1	98.6	98.3	96.1

The values of recall are somewhat worse than precision, i.e., it was very common, among all participants, to miss a selection. One possible explanation for misses is that, some participants, trying to finish the task as fast as possible, scanned just one column of a context, trying to find digits in nearby columns using their periphery vision.

We can observe from Figure 2 that the grand mean of AST for 2 columns is considerably faster than for 4 columns, and a bit better than for 3 columns over all sessions. A pairwise t-test shows that there was a significant difference between 2C and 3C ($t(5) = 4.49, p = .01$), between 2C and 4C ($t(5) = 13.76, p = 0$) and also between 3C and 4C ($t(5) = 7.80, p = 0$).

This is expected since for 2C the average saccade distance is lower than for 3C and 4C using the same bridge distance, and there is a relationship between saccade amplitude and duration. Exploring the entire context for 2C is also faster than for 3C and 4C (because of the fewer number of keys), which could also justify the faster 2C.

The 4C layout shows the worst time. A possible reason is that because with 4C saccades are longer when selecting items close to the vertical screen edges, sometimes the participants ended the saccade in the central area and had to repeat the selection process. This can also be a result, noticed by some participants, that occasionally the interface had a small delay to respond to context switching. We believe this problem to be an implementation issue, since the interface was implemented in Java/Swing, with 4C there are more keys to resize after every context switch than with 2C and 3C.

When we consider the ATT, as seen in Figure 3, the 3C layout is faster than 4C, which in turn is a little better than 2C. A pairwise t-test shows there was a significant difference between 2C and 3C ($t(5) = 4.11, p = .01$) and also between 4C and 3C ($t(5) = 7.50, p = 0$), however there was not significant difference between 2C and 4C ($t(5) = 2.56, p = .05$).

This can be explained because 3C and 4C requires less paging than 2C, which reveals that the time spent for paging was significant in the overall performance of the interface. Assuming that each user spent a constant time to switch pages (since the paging procedure was the same for all methods), the results reveal that it is preferable to use DCS to place more items and have less paging, instead of using CS with more paging to hold the same amount of selectable keys/objects.

4.2 Subjective evaluation of DCS

Subjective evaluation of the participants were consistent with the quantitative results: the 2C and 3C received the same evaluation regarding perceived speed. Only one user perceived the 4C as the second faster. The 2C method was evaluated as the simplest to use by all participants. One participant with previous experience in eye tracking said that with 3C it was easier to use peripheral vision to quickly explore a context and also was more comfortable to use than with the other two methods.

Participants were asked about how easy it was to select using CS and DCS. In a Likert scale from 1 (very hard) to 5 (very easy), the average response was 4.7, i.e., the participants found it very easy to make selections using both CS and DCS. Regarding context resizing, none of the participants found it disorienting. This can be explained by saccadic suppression (or masking): the inability to perceive whether a target has moved or not during a saccade [Bridgeman et al. 1975].

5 Conclusion

This paper introduced the concept of Dynamic Context Switching (DCS) as an extension of the traditional Context Switching (CS) paradigm for gaze-based interaction. Like CS, DCS has two replicated contexts, but the context that has the focus is displayed in full size, while the other one is reduced. A short fixation is used to detect focus within the maximized context, and a saccade to the other context triggers the selection and dynamically adjusts the sizes of the contexts. Therefore in CS and DCS, focus is controlled by fixations, and selections by saccades. DCS allows a better use of screen space and can improve the robustness of the system to gaze tracking noise. A region between the contexts, the bridge, is required to avoid unintended selections.

We have conducted a pilot user experiment with 6 participants to compare the performance of a 2 column fixed CS layout, 3 column and 4 column DCS layouts in a selection task, using simple gaze gestures for switching pages. Experimental results show that DCS improves user performance for the 3 column layout, but further experiments needs to be done to verify the usability of the 4 column layout, due to implementation issues that were revealed during the pilot experiment. The participants do not feel disoriented by the constant resizing. Actually, some have not even noticed the resizing. This can be explained due to saccadic masking phenomenon which suppresses our visual perception during saccades. All users felt DCS to be comfortable and easy to learn.

Acknowledgements

We thank Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP) for the financial support.

References

BRIDGEMAN, B., HENDRY, D., AND STARK, L. 1975. Failure to detect displacement of the visual world during saccadic eye movements. *Vision Research 15*, 6, 719–722.

HANSEN, D. W., SKOVSGAARD, H. H. T., HANSEN, J. P., AND MØLLENBACH, E. 2008. Noise tolerant selection by gaze-controlled pan and zoom in 3d. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, ACM, New York, NY, USA, ETRA '08, 205–212.

HUCKAUF, A., AND URBINA, M. H. 2008. Gazing with peyes: towards a universal input for various applications. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, ACM, New York, NY, USA, ETRA '08, 51–54.

HUCKAUF, A., AND URBINA, M. H. 2011. Object selection in gaze controlled systems: What you don't look at is what you get. *ACM Trans. Appl. Percept. 8* (February), 13:1–13:14.

JACOB, R. J. K. 1990. What you look at is what you get: eye movement-based interaction techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Empowering people*, ACM, New York, NY, USA, CHI '90, 11–18.

KUMAR, M., WINOGRAD, T., PAEPCKE, A., AND KLINGNER, J. 2007. Gaze-enhanced user interface design. Technical Report 2007-20, Stanford InfoLab, April.

MORIMOTO, C. H., AND AMIR, A. 2010. Context switching for fast key selection in text entry applications. In *Proceedings of the 2010 Symposium on Eye-Tracking Research & Applications*, ACM, New York, NY, USA, ETRA '10, 271–274.

OHNO, T. 1998. Features of eye gaze interface for selection tasks. In *Mahallas in Uzbekistan. Aline Coudouel, Sheila Marnie and John Micklewright*, IEEE Computer Society Press, 176–181.

STELLMACH, S., STOBER, S., NÜRNBERGER, A., AND DACHSELT, R. 2011. Designing gaze-supported multimodal interactions for the exploration of large image collections. In *Proceedings of the 1st Conference on Novel Gaze-Controlled Applications*, ACM, New York, NY, USA, NGCA '11, 1:1–1:8.

WARD, D. J., AND MACKAY, D. J. C. 2002. Fast hands-free writing by gaze direction. *CoRR cs.HC/0204030*.

WOBBROCK, J. O., RUBINSTEIN, J., SAWYER, M. W., AND DUCHOWSKI, A. T. 2008. Longitudinal evaluation of discrete consecutive gaze gestures for text entry. In *Proceedings of the 2008 symposium on Eye tracking research & applications*, ACM, New York, NY, USA, ETRA '08, 11–18.