Improving Code Reading and Comprehension on Large Displays

Selvihan Nazlı Kaptan[1] and Mehmet Göktürk[2]

[1] Bahçeşehir University, Çırağan Caddesi, 34353 Beşiktaş, İstanbul, Turkey
[2] Gebze Institute of Technology, Gebze, 41400, Kocaeli, Turkey
syavuzer@bahcesehir.edu.tr, gokturk@bilmuh.gyte.edu.tr

Abstract. Due to advances in display technologies and continuous decrease in large display prices, more users are choosing larger displays or multiple monitors for personal and professional use although standard size desktop monitors are still widely used. As programmers use a larger display surfaces to read and understand their code, current code editors are designed for standard monitor sizes and they do not exploit the extra space that comes with a larger display. In this paper, we discuss the use of a large display for code reading and test whether code reading can be improved by utilizing larger screen space.

Keywords: large displays, user performance, usability, code reading, reading and comprehension.

1 Introduction

Even though many programmers now prefer a larger display surface to read and understand code, one can see that current code editors are designed for standard monitor sizes. These code editors do not address advantages of extra space that comes with larger displays and multiple monitor extended desktops. One can understand that most lines of code is short in length and many programmers tend to split longer lines with multiple short ones as programming standards and practice advise shorter (approximately maximum of 80 characters due to historical reasons) line lengths.

Most programmers do not use extra large fonts even when they move to a large viewing space and they keep their previous user settings (font sizes, etc.). This results in a large code editor window with half of the code window being frequently empty. To address this issue, we propose to utilize the empty half of a large screen to provide user with textual information on the source code being manipulated aiming to reduce the mental load on the programmer.

In our preliminary survey, we tried to identify basic problems that programmers experience when reading a piece of code. To proceed on this track, it is quite important to know the type of information programmers search for most while navigating/scrolling since this knowledge can help to reduce navigational needs. When processing a single line of code, each variable declaration, function call or object creation is a potential reason for a navigational requirement as reader will need to recall a previous fact, and if not successful, he will have to navigate to the related

A. Marcus (Ed.): Design, User Experience, and Usability, Pt II, HCII 2011, LNCS 6770, pp. 128–134, 2011.

part of code. According to our survey results, while reading a piece of software, participants spend time on navigating between lines to follow the code and memorizing functions/variables. Over 80% of the participants reported that they navigate to the same part of the code to recall information when they are reading source code. Furthermore, when the participants navigate back to the line where they left off, they re-read the same lines to remember (e.g. after navigating to a function definition, and scrolling back to the line where the function was called). Navigating back and forth, short-term memory is used inefficiently and the reader consumes more mental resources. This also relates to the age related problems that occur in programmers where coding ability usually decrease after certain age.

2 Related Work

Research on reading electronic text traditionally suggests reading on a computer screen is slower than reading from a book; however, no significant effect on comprehension is observed. Moreover, reading speed is affected by the format of information that is the number of characters per line, the number of lines per page and interlines spacing affects the speed at which one reads [1]. This suggests that reading speed will be different on larger displays where more lines will fit into a single page.

Studies on large displays were mostly focused on collaboration, and less has been done to discover the benefits to individual users. Richardson et al (1989) studied the effects of display size on readers' comprehension and manipulation on electronic texts. User experiments showed comprehension rate and performance did not change with display size, yet users preferred larger displays for reading [2]. On the other hand, Lin and Hsieh (1996) measured the effects of physical display size on task completion time as a part of their user experiments, and in result they concluded users completed tasks faster on larger displays while this did not change the correct response percentage.

Code reading, the process of careful reading of source code for modification, refinement and debugging purposes, has been one of the most important skills to work in both industries and academia [3]. Reading process involves both resolution of information and decoding the location of information in 2D space [4]. Thus, users form a cognitive map that defines where a word or a phrase resides within the text. The task of reading and understanding a source code mostly requires re-reading certain part of it. Cognitive mapping activity is expected to be higher when reading a piece of code since the reader will try to create a map of variables and values, functions and algorithms in order to understand the code being viewed.

This process of mapping can be thought as encoding the source document. In regular texts, highlighting has been considered as an effective tool in encoding process. Research has repeatedly shown that tactics such as highlighting, note taking, and underlining help with the encoding process. These tactics on computers has been found to decrease the time needed to locate a target [5]. This may suggest marking a piece of information as important decreases the time spent on reviewing the relevant material to access it.

3 User Experiment

Goal behind this study is to reveal the differences between users' code comprehension performances on medium and large displays. For code assistance window, textual views of code is preferred because text has the advantages of begin easily communicated, effectively manipulated and highly scalable. A textual model may also improve comprehension by expressing information directly within the source code being manipulated. Any attempt to successfully observe the effects of code assistance tool on a large display should address the following issues: (i) On large displays, more lines will be visible at a time. This may provide a better performance on larger displays than on medium displays. This difference should be distinguished from the improvement introduced by the code assistance tool. (ii) Many studies have shown that even with similar backgrounds, performances of programmers show significant differences [6, 7, and 8]. Personality is also a factor which affects the programmers' performance on different stages of development process.

3.1 Physical Experiment Setup

In the experiments, 2 (two) different screens have been used. One 17" Acer AL1716 and one 40" Alba LCD TV. In each experiment users are given a display, a mouse, a pen and paper-based task assignment. Since users are not expected to type, keyboards and other peripherals were removed from the setup.

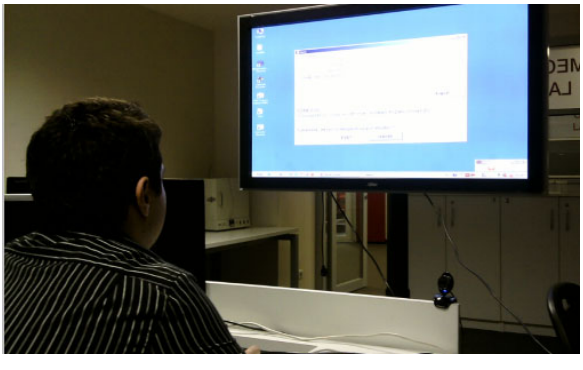

Fig. 1. Experiment setup

Displays are placed to achieve a FOV between 45^{o}-60^{o} which is the range where users performed better in numerous studies. Participants were free to adjust their positions as they wish in order to simulate a comfortable working environment.

Participants were recorded during the entire experiment. For each experiment, response time, number of correct responses and scroll displacements were also logged.

3.2 Subjects

Subjects were sophomore and junior students of Department of Computer Science at Bahçeşehir University in a course on Data Structures and Algorithms (using C++). In total, 24 students participated and all students had attended at least two computer science classes before.

3.3 Objects

Objects are simple C++ programs that use basic programming structures such as arrays, functions, variable declarations, logic/arithmetic expressions and input/output commands. To minimize performance differences between participants based on their level of knowledge in C++, programs used in experiments are kept as simple as possible.

3.4 Experiment Procedure

Each experiment consisted of an output evaluation for a given C++ program. Each subject experienced all three setups and was time-tested with 2 experiments on each display setup. For each display setup, participants were placed in the lab and were given a C++ code and they were asked to determine its output. Experiment timer started as soon as the subject accepts to see the code to be evaluated.

The experiments were conducted in three main groups; medium display size with no code assistance, large display with no code assistance, large display with code assistance. Code-assistance window is populated with textual information on the code being evaluated. Code-assistance window contents were statically prepared for each source code separately and loaded from files with the assignment code. This information window code included reminders on variable declarations, function declarations and previous and further references to these (according to currently active line of code). To simplify the generation of window contents, source codes for the tasks are divided into main segments, and for each segment block of text information is prepared. When the participants focused on a line (when the cursor is active on a line) within a segment, segment-related part of the text information was loaded in the assistance window.

After completing two experiments on a display, users scored their assignment according to their difficulty on 0-5 scale where 5 indicates hardest and 0 the easiest task.

4 Results

Response times collected during user experiments are normalized and are given in Figure 2. While there is slight improvement on response times between medium display and large display with no code assistance, this difference needs to be explored in detail with further experiments. Response times showed that code comprehension performance is higher with code assistance window used on large displays.

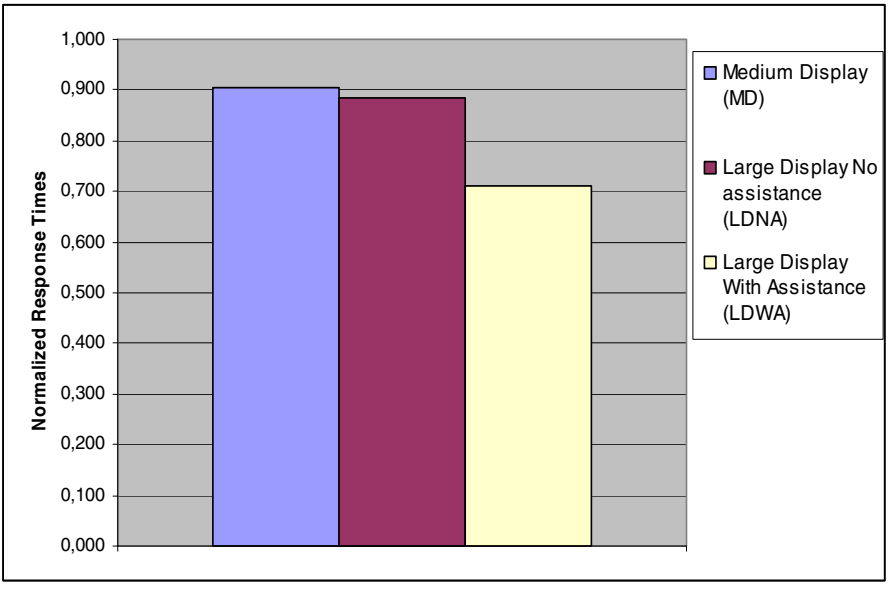

Fig. 2. Normalized response times on three different setups

After the experiments on each setup, participants were asked to score the task difficulties. Figure 3 summarizes subjects' responses to task difficulties. The differences between 3 setups are important since tasks are similar and monitor sizes and code assistance is altered. Figure shows subjects consider code comprehension as more difficult on medium displays then on large displays.

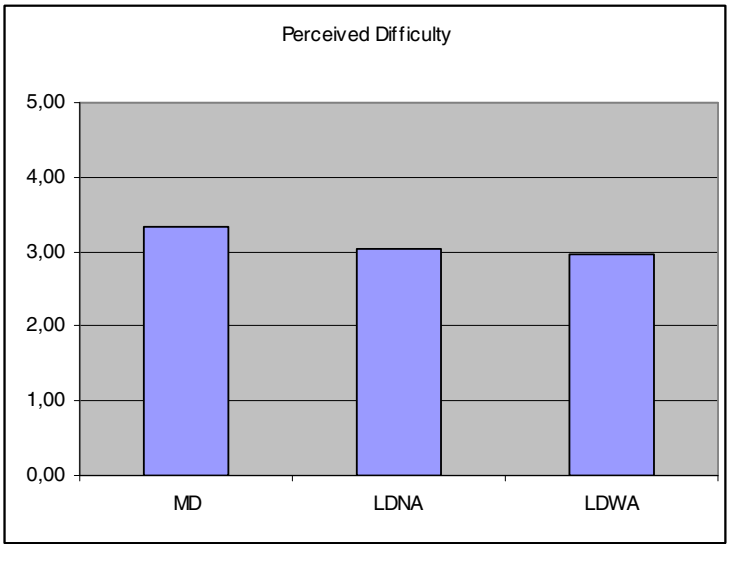

Fig. 3. Users comments on task difficulties on different setups

Although response times benefit from code assistance window, perceived task difficulty scores are close on large displays with and without code assistance frame. On the other hand, the difference between medium displays and large displays is mostly based on reduced navigation requirements.

5 Conclusions

In this study we aimed to examine how the skills of participants in understanding the functioning of a piece of code are affected by reading it on different display sizes and with and without code-reading assistance window. In user experiments, we used a code-reading assistance frame located on the empty half of the large display which included textual information on the code being manipulated. The window presents constant reminders on specific code parts (variables and function declarations of interest and previous references to these), recently visited code blocks and comments extracted from the current code block.

Our results suggest that task completion times are improved by the use of larger display space and reading assistance frame, although success rates are not significantly affected by the setup. Although majority of programmers still use displays less than 22" in size, larger displays, dual (or multiple) monitors and projection systems are becoming prevalent. As a common skill in both industries and academia, both professional programmers and programming course students may benefit from extended display surface with the help of appropriate assistance tools which do not exist in currently available environments.

References

1. Kruk, R.S., Muter, P.: Reading of Continuous Text on Video Screens. Human Factors 26(3), 339–345 (1984)
2. Richardson, J., Dillon, A., McKnight, C.: The effect of window size on reading and manipulating electronic text. In: Megaw, E. (ed.) Contemporary Ergonomics 1989, pp. 474–479. Taylor and Francis, London (1989)
3. Ueno, D.: Linenum-≫Info: System Support for Code Reading. In: Eighth IEEE International Conference on Advanced Learning Technologies, ICALT 2008, July 1-5, pp. 600–602 (2008)
4. Wästlund, E., Norlander, T., Archer, T.: The effect of page layout on mental workload: A dual-task experiment. Comput. Hum. Behav. 24(3), 1229–1245 (2008)
5. Fisher, L., Coury, B.G., Tengs, T.O., Duffey, S.A.: Minimizing the time to search visual displays: the role of highlighting. Human Factors 31(2), 167–182 (1989)
6. Shneiderman, B.: Software psychology: human factors in computer and information systems. Winthrop Publishers, Cambridge (1980)
7. Grant, E.E., Sackman, H.: An Exploratory Investigation of Programmer Performance Under On-Line and Off-Line Conditions. IEEE Transactions on Human Factors in Electronics HFE-8(1), 33–48 (1967)
8. Boehm, B.W.: Software engineering economics. Prentice-Hall, Englewood Cliffs (1981)

9. Goldberg, J., Helfman, J.: Evaluating User Expectations for Widescreen Content Layout. Paper presented at the Usability Professionals' Association Conference, Austin, TX, USA (June 2007)
10. Tan, D.S., Gergle, D., Scupelli, P., Pausch, R.: Physically large displays improve performance on spatial tasks. ACM Trans. Comput.-Hum. Interact. 13(1), 71–99 (2006)
11. Kawashima, H., Oda, K., Kobayashi, A., Suzuki, M.: Does limited window size affect text comprehension? In: Vision 2005 - Proceedings of the International Congress held between 4 and 7 April 2005, London, UK. International Congress Series, vol. 1282, pp. 622–626 (September 2005) ISSN 0531-5131