Investigating the role of programmers' peripheral vision: a gaze-contingent tool and an experiment proposal

Roman Bednarik
University of Eastern Finland
roman.bednarik@uef.fi

Paul A. Orlov
St.Petersburg State Polytechnical University
paul.a.orlov@gmail.com

Keywords: eye tracking, programming, perceptual span, experiment proposal

Abstract

Previous research of visual attention in programming has shown differences in how expert and novice programmers attend to the available information. What has not been yet sufficiently investigated is the degree with which the information is sampled by the peripheral vision during programming. Such issues have been investigated by a contingent-window paradigm in other domains and we have thus developed a tool allowing such studies in programming. In this paper, we introduce the tool and a proposal for an experiment we plan to conduct.

1. Visual attention in programming

Computer programmers typically develop software using highly dynamic, integrated development environments (IDE). These tools often present information in adjacent windows that provide multiple representations about the software. One line of psychology of programming (PoP) research employs so called contingent window paradigm to investigate how programmers coordinate the representations during comprehension and debugging (Romero et al., 2002).

Other studies have employed eye-tracking as a proxy to visual attention during programming. Since Crosby and Stelovsky's 1990 seminal paper (Crosby and Stelovsky, 1990), a body of knowledge about attention in programming has been growing, indicating that expertise in programming is characterized by distinct patterns of visual attention (Bednarik, 2012). For example, expert programmers employ a wider range of strategies during debugging with multiple representations (Bednarik, 2012) that develop over time (Bednarik and Tukiainen, 2008), they attend to beacons (meaningful areas of code) (Crosby and Stelovsky, 1990), and attend to output more often than novices during debugging (Hejmady and Narayanan, 2012).

Beyond single user studies, recent developments in dual eye tracking methodology allowed expansion of eye tracking programming studies to multiple user situations (Jerman et al., 2012). Jermann and Nüssli (2012) conducted a study in which pairs of programmers collaboratively comprehended source code while their visual attention was captured using an eye-tracker. They found that programmers' attention is often coupled and this coupling increases during code selection episodes accompanied by speech. Bednarik et al. (2011) transferred the point of gaze of an expert programmer to the display of novice programmers while explaining an algorithm. They showed that the attention of the novice programmers was more similar when the gaze of the expert was shown, but that had no effect on the comprehension outcome.

An area deserving a deeper investigation in PoP is the role of visual attention beyond the fixation point. In other domains, such as in reading, the perceptual span - the breadth of attention surrounding the point of gaze - plays a significant role during stimuli processing. The size of perceptual span directly influences the spread of covert attention during scanning of stimuli, and thus mediates performance. In search, for instance, the span varies depending on the difficulty of the distractor.

In programming, we hypothesize, wider perceptual span provides (expert) programmers faster and more effective navigation in source code and better information extraction in coordination of various representations. Further, we conjecture, compound and more difficult expressions have effect on perceptual span and the difficulty again interacts with expertise.

2. Perceptual span and Gaze contingent moving window paradigm

For most of the awaken time human's vision fixates objects. Fixations, the relatively stable movements of the eyes when perception is turned on, typically last about 300ms. Saccades, on the other hand, are rapid ballistic movements lasting no more than tens of milliseconds during which perception is virtually shut down. While fixations are required for perception to happen, there is a question about the size of the perceptual area around the center of fixation within which detailed information can still be sampled without an additional eye movement. Dodge suggested calling this area a fixation zone (Dodge, 1907).

In his research of the perceptual span in reading, Rayner discovered that the size of perceptual span is not constant but varies as a function of text difficulty. The size of the span decreases when text is difficult to read (Inhoff et al., 1989; Rayner, 1986). Rayner also suggested that the visual field can be divided into three regions: foveal, parafoveal and peripheral. Although acuity is very good in the fovea (the central 2° of vision), it is not nearly as good in the parafovea (which extends out to 5° on either side of fixation), and it is even poorer in the periphery (the region beyond the parafovea) (Rayner, 1998). Studies by Gippenreiter (1964) shown that participants recognize objects positioned 10-15° from the center of fixation. The perception of the peripheral signal occurred simultaneously with the preparation of the saccade, when the eye was still on the fixation point (Gippenreiter, 1964).

Several methods have been developed in past to evaluate the breadth of attention, many of which fall under the category of moving window paradigm. One of the most prominent techniques to investigate the role of perceptual span under this paradigm is *gaze-contingent multiresolutional display* (GCMRD) (Reingold, Loschky et al., 2003). Such systems combine eye-tracking and a stimulus so that the center of fixation is determined by an eye-tracker and the current fixation areas on the stimulus are rendered in sharp focus while the rest of the stimuli is blurred. The properties of the focused area can be contingent on the characteristics of the fixation, for example, longer fixations cause the focused area to be enlarged.

There are numerous applications for GCMRD systems, such as in resource-demanding graphics environments to allow for efficient 3D rendering and modeling, and in video and image compression methods. The following lists some of the main issues that arise when constructing GCRMD systems (Reingold, Loschky, et al., 2003):

- ≅ Can we construct just undetectable GCMRDs that maximize savings in processing and bandwidth while eliminating perception and performance costs?
- ≅ What are the perception and performance costs associated with removing above-threshold peripheral resolution in detectably degraded GCMRDs?
- ≅ What is the optimal resolution drop-off function that should be used in guiding the construction of GCMRDs?
- ≅ What are the perception and performance costs and benefits associated with employing continuous vs. discrete resolution drop-off functions in still vs. full-motion displays?

The range of applications of GCMRD extends beyond restricting views and bandwidth savings. For instance, intelligent interfaces can make use of gaze-contingent zooming, where the available information about the object increases with the proximity of attention to the object. One example of measuring attention in similar systems builds on face-detection algorithms. The closer a person is to the monitor, the higher level of attention and the higher level of detail are employed. The amount of information depends on the distance of the human face of the monitor (Harrison, Dey, 2008).

3. Gaze-contingent moving window in PoP

In programming research, the gaze-contingent window paradigm has previously been employed in a series of studies of Romero et al.. As an alternative to gaze-contingency, the authors employed the Restricted Focus Viewer (RFV) (Jansen et al., 2003). RFV emulates gaze-contingency by mouse-contingency in such a way that a user controls the position of the only focused area on the screen

using the computer mouse. The movements of the mouse are recorded and are supposed to be the estimates of the point of gaze in time. In a replication study Bednarik and Tukiainen (2007) however demonstrated that in programming such assumption does not hold, and in addition, the restriction interferes with natural strategies.

One explanation of the findings is that programmers do need to use the peripheral information for some purposes, and when the periphery is masked, the performance decreases because peripheral information is not available directly. Thus, a question arises concerning the role of periphery and size of perceptual span in programming. Bednarik and Tukiainen reported that when using RFV, programmers glanced towards the blurred areas and that the contingency had slightly different effects depending on expertise.

Their study does not provide an answer why the programmers perform in this way. A hypothesis to investigate is whether expert programmers are better able to extract parafoveal and peripheral information without moving the eyes and when the cost of accessing the information increases because of blurring, they need to perform extra eye-movements.

There are numerous questions spanning from the notion of role of the perceptual span that need to be investigated to provide a qualified answer. What is the size of perceptual span of a programmer? Is there a relationship between expertise and size of perceptual span? How do changes to the size of the visible area influence behavioral patterns of programmers? Would limiting the perceptual span for novice programmers have detrimental or positive effects on performance?

These questions determine the objectives of the research planned here. Answers to the above questions will not only help in understanding the role of attention in programming, but could also will be useful when asking what effects expertise in programing has on visual attention skills.

In the rest of this paper, we present a study proposal to investigate the role of perceptual span in programming and a tool allowing GCMRD in programming.

4. Objectives and Methods of the planned research

The primary goal of the study is to determine the role of visual attention span in programming. We aim to conduct studies and test hypotheses about the influence of the perceptual span area on the behavioural patterns of programmers. To achieve this goal it is necessary to set up a number of experiments.

We put forward the following working hypothesis:

- ≅ The pattern of gaze of programmers when working with a restricted window IDE would be different to the pattern of gaze without blurring, dependent on the expertise of the programmer and task at hand.

We aim to answer the hypothesis by running a series of studies in which participants of distinct expertise would be engaged in various tasks such as comprehension and debugging. We will systematically modify the size of the contingent window. The dependent measures of the studies would include debugging and comprehension performance, task difficulty index, and eye-tracking measures.

The following section describes the design of the gaze contingency tool developed for the planned studies.

5. Gaze contingent IDE

To test the hypotheses it is necessary to first create an appropriate research framework. To our knowledge, existing solutions such as RFV do not allow unrestricted stimuli presentation and thus we decided to develop an extension with such functionality by ourselves.

We began developing the platform adhering to the following requirements:

- The system has to be able to serve as a standard software development tool in Java programming language.
- The source code in the editor panel will be blurred except for an area whose center is defined by a direction of programmer gaze.
- The transition from focused to blurred areas should be gradually smoothed.
- All movements of the focused area should be logged along with all source code elements that were at the point of fixation.

To satisfy the first requirement, we selected NetBeans as the experimental environment. Implementation of the source editor window blurring was carried out with the help UI library SWING. We have overridden the render method so the whole image on the screen was blurred. It was decided that the method of implementation of the blur will be based on a combination of pre-prepared maps of the gradient and the current image. This allows modifying the gradient map in a graphics editor or using programming. Changing the map scale is possible too, as well as affine mapping that allows using any 2D image as a gradient map. A screenshot of the application is shown in Figure 1.

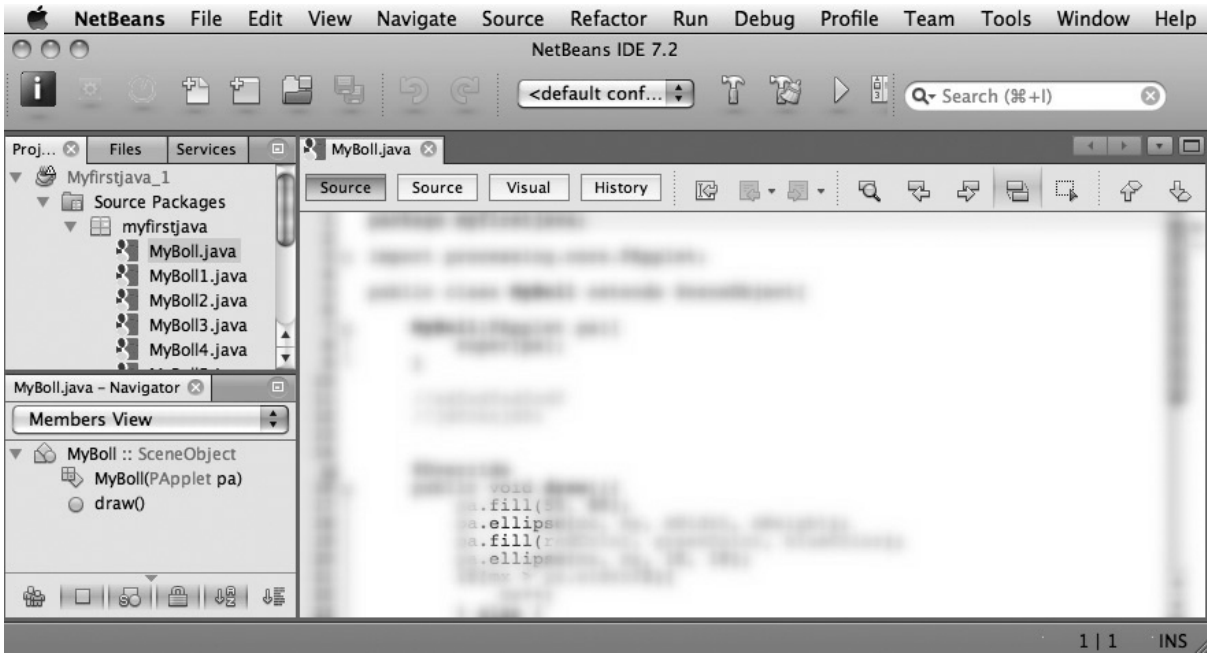

Figure 1. Screenshot of the experimental application.

The figure shows that the source editor window is blurred (and washed and line numbers and vertical scrolling, and color tips for scrolling). The sharp zone is shown only at the fixation point.

Developing of the logging mechanism for source code elements that have hit the point of fixation was carried out also by using the SWING library. The position on the screen indicates the corresponding substring in the source code. After that, when we find the substring, we use a Java parser to detect the type of substring. Such information can further be used for modifying the contingency, for example, only certain elements or types of stings could be rendered in full, responding immediately to the user's input. As a result, the output log files are similar the following example (Table 1):

Dot	Expression	Kind	Time	File
385	noFill	METHOD	1348066278437	MyBall.java
385	noFill	METHOD	1348066278437	MyBall.java

…	…	…	…	…
238	points	FIELD	1348066278721	MyBall.java
433	strokeWeight	METHOD	1348066279859	MyBall.java
621	Math	CLASS	1348066280619	MyBall.java
…	…	…	…	…
627	sin	METHOD	1348066281232	MyBall.java
633	Math.sin((float)	NONE	1348066281421	MyBall.java

Table 1. – log file example.

In this fragment, the "Dot" field corresponds to the caret location in the text file of source code, "Expression" is the substring in user's focus. The "Kind" column contains the type of substring which is obtained from JavaSource parser, "Time" is a precise timestamp of the fixation and "File" is the name of the opened source file. We will employ the logs for analysis of the visual attention patterns during navigation in the source code. A demonstration video can be found online at http://vimeo.com/49874161.

5. Future gaze-contingent IDEs

The phenomena of perceptual span and peripheral vision are not only of theoretical interests. They may be incorporated into the solutions of important practical HCI problems such as design of graphical user interfaces in general and software development interfaces in particular. Designers of interactive user interfaces wish to display information to the users so that it is attended, processed and used in the right time. Information about what information is processed and when thus contributes to better designs of interactive systems.

We wish to expand the debate about visual attention in programming by discussing the role of perceptual span and peripheral vision. We presented a gaze-contingent extension to a commonly used programming tool and an outline of studies to investigate its effects in programming tasks. Further steps include refining the experimental design and executing the study.

5. References

Bednarik, R.(2012) Expertise-dependent Visual Attention Strategies Develop Over Time During Debugging with Multiple Code Representations. International Journal of Human - Computer Studies 70, pp. 143-155.

Bednarik, R., Tukiainen, M.(2008) Temporal eye-tracking data: Evolution of debugging strategies with multiple representations. In Proc. ETRA Symposium, 99-102

Crosby, M. E., Stelovsky, J. (1990) How do we read algorithms? A case study. IEEE Computer 23, 1, 24–35.

Dodge, R.(1907) On experimental study of visual fixation.-"Psychol.Monogr.", v.35, p.95.

Gippenreiter J.B. (1978) Movements of the human eye. Monography, Moscow, pub. Moscow University, 256p.

Harrison Chris, Anind K. Dey.(2008) Zoom: Proximity-Aware User Interface and Content Magnification. CHI 2008 Proceedings I am here. Where are you? April 5-10, Florence, Italy.

Hejmady, P., Narayanan, H. N.(2012) Visual attention patterns during program debugging with an IDE. In Proceedings of the Symposium on Eye Tracking Research and Applications (ETRA '12), Stephen N. Spencer (Ed.). ACM, New York, NY, USA, 197-200.

Inhoff, A. W., Pollatsek, A., Posner, M. I., & Rayner, K. (1989). Covert attention and eye movements during reading. Quarterly Journal of Experimental Psychology, 41A, 63-89.

Jansen, A. R., Blackwell, A. F., & Marriott, K. (2003). A tool for tracking visual attention: The Restricted Focus Viewer. Behavior Research Methods, Instruments, & Computers, 35, 57-69.

Jermann, P., Nüssli, M. (2012) Effects of sharing text selections on gaze cross-recurrence and interaction quality in a pair programming task. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW '12). ACM, New York, NY, USA, 1125-1134

Jermann, P., Gergle, D., Bednarik, R., Brennan, S.(2012) Duet 2012: dual eye tracking in CSCW. In Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work Companion (CSCW '12). ACM, New York, NY, USA, 23-24

Rayner K. (1998) Eye Movements in Reading and Information Processing: 20 Years of Research. Psychological Bulletin 1998, Vol. 124, No. 3, 372-422

Reingold E. M., Lester C. Loschky, George W. McConkie, David M. Stampe. (2003) Gaze-Contingent Multiresolutional Displays: An Integrative Review. HUMAN FACTORS, Vol. 45, No. 2 pp. 307-328.

Romero, P., Lutz, R., Cox, R., Du Boulay, B. (2002) Co-ordination of multiple external representations during Java program debugging. In HCC '02: Proceedings of the IEEE 2002 Symposia on Human Centric Computing Languages and Environments (HCC'02), IEEE Computer Society, Washington, DC, USA, 207.