Understanding of programmers’ attention provides benefits for developing comprehension models and facilitating programming education activities. However, the visual attention studies in a psychology of programming explore central vision mostly and do not study the extrafoveal usage before. This work reports on a first-ever investigation of the role of extrafoveal information during programming. Here we provide a Gaze-contingent Tool, a Latency Evaluation Method, and experiments results.
PAVEL A. ORLOV


Publications of the University of Eastern Finland
Dissertations in Forestry and Natural Sciences
No 256

Academic Dissertation
To be presented by permission of the Faculty of Science and Forestry for public examination in the Auditorium AU100 in Aurora Building at the University of Eastern Finland, Joensuu, on December, 19, 2016, at 12 o’clock noon.

School of Computing
ABSTRACT

This dissertation is a synthesis of four publications and is focused on the role of extrafoveal information processing during source code comprehension.

The extrafoveal information is the information collected from the visual area that located on a 2° from the gaze fixation point. We used custom gaze-contingent systems to study extrafoveal information processing in reading and visual search domains. Gaze-contingent systems use gaze location on a screen to render the final picture. Latency is an important feature of such systems. The time between eye movements and corresponding screen updates should not exceed 80 ms. This limit prevents subjects from understanding the system’s delay, which could otherwise affect the results of the study. First contribution of this work is a low-cost latency measurement system that we build to evaluate the latency of gaze-contingent system in our study.

Second contribution of our work is a novel gaze-contingent software–ScreenMasker–was built to conduct the study of extrafoveal information processing. It employs a window-moving paradigm and restricts the area of view in real time. ScreenMasker runs on top of each software window in the computer screen–this functionality makes it ideal for the study of source code comprehension.

Third contribution of the research is the finding that both experts and novices use the extrafoveal information, but in different manners. Experts make more active use of the information from the extrafoveal area to solve the task. This information is used not only to plan the next saccade, but also to encode source code elements. Where the extrafoveal information is unavailable, experts’ behavior becomes similar to that of novices.

And fourth contribution of our work is the finding that the lesser the availability of the extrafoveal objects in the restricted-
view mode, the stronger the emotional feedback given by expert programmers.

ACM Classification: H.5.2, H.5.1, H.5.2, H.5.1, I.3.6, H.1.2, D.2.2

Keywords: gaze-contingent; eye movements; eye-tracking; programming; expertise level; fovea; extrafoveal area; parafovea; periphery vision
This research is the result of cumulative work carried out at the Interactive Technology Group, University of Eastern Finland, and the Laboratory of Computer-Human Interaction and Usability, St. Petersburg Polytechnic University. I would like to express my gratitude to all faculty members from both universities who helped me during my study.

I am especially thankful to Roman Bednarik, my supervisor. It was a privilege to work with him. Most of the time I was physically in Russia, and Roman was supervising my work remotely. I know that it is not easy.

I am also grateful to the reviewers of the dissertation. I want to thank Yann-Gaël Guéhéneuc and Andrew T. Duchowski for agreeing to review my work and providing helpful critique and comments. I am especially thankful to Martha Crosby—I greatly appreciate that you found the time and consented to become my opponent.

Many thanks to my friends and colleagues. I am really thankful to those colleagues with whom I have worked in Russia, Finland, and Germany, especially Anna Kholina, Artem Dudkevich, Niko -lay Apraksin, Vladimir Laptev, Victor Yanchus, Vladimir Ivanov, Markku Tukiainen, Hana Vrzakova, Shahram Eivazi, Tersia Gw ases, and Teresa Busjahn. I do not specify the help rendered by each of you, because I know that it is more important to find your name here in this list.

I also greatly appreciate my big family, who have given me strength during these years.

This thesis will end an important period of my life. This period was hard but very interesting. Having studied over three years, I know that I am supposed to be ready to learn again, because there is no final point in science.


Pavel A. Orlov
Preface

This research is the result of cumulative work carried out at the Interactive Technology Group, University of Eastern Finland, and the Laboratory of Computer-Human Interaction and Usability, St. Petersburg Polytechnic University. I would like to express my gratitude to all faculty members from both universities who helped me during my study.

I am especially thankful to Roman Bednarik, my supervisor. It was a privilege to work with him. Most of the time I was physically in Russia, and Roman was supervising my work remotely. I know that it is not easy.

I am also grateful to the reviewers of the dissertation. I want to thank Yann-Gaël Guéhéneuc and Andrew T. Duchowski for agreeing to review my work and providing helpful critique and comments. I am especially thankful to Martha Crosby—I greatly appreciate that you found the time and consented to become my opponent.

Many thanks to my friends and colleagues. I am really thankful to those colleagues with whom I have worked in Russia, Finland, and Germany, especially Anna Kholina, Artem Dudkevich, Nikolay Apraksin, Vladimir Laptev, Victor Vanchus, Vladimir Ivanov, Markku Tukiainen, Hana Vrzakova, Shahram Eivazi, Tersia Gowases, and Teresa Busjahn. I do not specify the help rendered by each of you, because I know that it is more important to find your name here in this list.

I also greatly appreciate my big family, who have given me strength during these years.

This thesis will end an important period of my life. This period was hard but very interesting. Having studied over three years, I know that I am supposed to be ready to learn again, because there is no final point in science.

St. Petersburg – Joensuu, September 2016. Pavel A. Orlov
LIST OF PUBLICATIONS

This thesis summarizes the following original publications reproduced here by permission:


AUTHOR’S CONTRIBUTION

All publications that form the parts of this thesis are co-authored by the author’s supervisor. The most recent publication has three authors. Roman Bednarik, author’s supervisor, is a co-author of all four papers, Lyudmila Orlova is a collaborator of the most recent paper. Each work is a result of a collaborative effort of its authors. The publications selected in this thesis are original research papers on gaze-contingent environment and extrafoveal information processing. The author of this thesis made the major contributions to all the papers, including the design of the empirical work, carrying out of the experiment, and collection of data. He wrote the first draft and was responsible for the final drafting as well as analysis and reporting of the results. The manuscripts of all four papers were written in cooperation with other contributors, with the author of this thesis performing the bulk of the work.

The work presented in this dissertation has not been conducted individually, but is a result of joint efforts of several contributors. The CUDA (Compute Unified Device Architecture by Nvidia Corporation) part of the ScreenMasker software was developed by the author with the assistance of Artem Dudkevich. Anna Kholina helped develop the video clip used to demonstrate the Low-Cost Latency Measurement System.
Contents

1 INTRODUCTION ........................................... 1
   1.1 Motivation ......................................... 4
   1.2 Goal of Research and Research Questions ........ 6
   1.3 Method ........................................... 7
   1.4 Research Process and Organization of the Thesis ....... 8

2 USING GAZE-CONTINGENT SYSTEMS FOR STUDYING PARAFOVEAL INFORMATION PROCESSING .. 11
   2.1 Eye Movements ..................................... 12
   2.2 Visual Information Processing ....................... 13
   2.3 Parafoveal Processing in Reading .................... 15
   2.4 Parafoveal Processing in Visual Search ............... 17
   2.5 Eye-tracking Methods and Gaze-contingent Systems .. 21
   2.6 Latency of Gaze-contingent Displays ................. 26
   2.7 Summary ........................................ 28

3 VISUAL ATTENTION IN THE PSYCHOLOGY OF PROGR\- AMMING ........................................... 31
   3.1 Eye-tracking Studies of Source Code Comprehension .... 31
   3.2 Eye Movements During Source Code Comprehension: Experts and Novices .................. 34
   3.3 Summary ........................................ 37

4 SUMMARY OF PUBLICATIONS ............................ 39
   4.1 Paper I: Low-cost Latency Measurement System for\n       Eye-mouse Software ............................... 39
       4.1.1 Background and Aims ......................... 39
       4.1.2 Results and Discussion ........................ 40
   4.2 Paper II: ScreenMasker: An Open-source Gaze-contingent\n       Screen Masking Environment ..................... 42
Visual perception is one of the most actively researched areas in applied psychology today. Human vision is said to be made up of a central vision and a peripheral vision (extrafoveal and peripheral areas). The visual attention studies in a psychology of programming explore central vision mostly and do not study the extrafoveal usage before. The objective of this work is to research the role of extrafoveal information during programming.

For several decades, software development companies have played a vital role in global economy and business. New job positions—e.g., programmers—have emerged. They have firmly established themselves in society. A programmer's work is associated with a high level of mental workload, intensive cognitive processing, and obtaining information from visual stimuli (e.g., from the source code). Source code comprehension is one of the main components of software production. It is highly important to understand and describe the programming process, to determine the psychology foundations behind the source code comprehension, and to build models of programmers' behavior.

During their workday, programmers typically produce and read source code from a computer screen. Reading is directly connected with visual information processing and eye movements. The visual system is highly complex and consists of different parts used to obtain, stream, and process visual information [1]. Human eyes have a fovea—the area of retina that collects data with high accuracy. The visual angle that corresponds to the foveal area is quite small, and it is necessary to point it at the locations that contain the required information [2]. In other words, human vision is active in terms of selecting information from an environment. The locations selected for gaze fixation constitute an observable behavior that provides insight into the perceptual and cognitive processes. The dominant point of view suggests that eye movements open for us a window...
1 Introduction

Visual perception is one of the most actively researched areas in applied psychology today. Human vision is said to be made up of a central vision and a peripheral vision (extrafoveal and peripheral areas). The visual attention studies in a psychology of programming explore central vision mostly and do not study the extrafoveal usage before. The objective of this work is to research the role of extrafoveal information during programming.

For several decades, software development companies have played a vital role in global economy and business. New job positions—e.g., programmers—have emerged. They have firmly established themselves in society. A programmer’s work is associated with a high level of mental workload, intensive cognitive processing, and obtaining information from visual stimuli (e.g., from the source code). Source code comprehension is one of the main components of software production. It is highly important to understand and describe the programming process, to determine the psychology foundations behind the source code comprehension, and to build models of programmers’ behavior.

During their workday, programmers typically produce and read source code from a computer screen. Reading is directly connected with visual information processing and eye movements. The visual system is highly complex and consists of different parts used to obtain, stream, and process visual information [1]. Human eyes have a fovea—the area of retina that collects data with high accuracy. The visual angle that corresponds to the foveal area is quite small, and it is necessary to point it at the locations that contain the required information [2]. In other words, human vision is active in terms of selecting information from an environment. The locations selected for gaze fixation constitute an observable behavior that provides insight into the perceptual and cognitive processes. The dominant point of view suggests that eye movements open for us a window
Pavel A. Orlov:

The first investigations of eye movements during natural-language reading were done in the late 19th century, and interest in such studies has been growing ever since [5]. The studies of foveal information processing assume that both attention and gaze fixation are directed at the same point of the visual stimuli [2]. We will based on the classic definition of attention by William James: 'It [Attention] is the taking possession of the mind, in clear and vivid form, of one out of what seem several simultaneously possible objects or trains of thought.' [6] The position of human gaze plays a key role in the interpretation and understanding of visual information when reading or performing a visual search. Nevertheless, human attention can be directed at a location that is different from the point of gaze fixation [7, 8]. Attention can take place not only in the foveal area, but in the periphery [9]. As we move away from the center, the resolution gradually decreases, but it still provides valuable visual information. Therefore, the extrafoveal zone is important for human visual perception [10].

The therapeutic practice knows that there exist a number of pathologies of the extrafoveal zones, such as the tunnel vision effect [11]. Generally, when the gaze switches to the extrafoveal area, a human predicts the information from the future fixation. For instance in reading, a human predicts the letter for the next gaze fixation or even the meaning of the next word [12, 13].

The reading of source code differs from natural-language reading [14]. Programmers have more complex gaze fixation patterns and strategies [15]. Eye-tracking is one of many data collection tools (verbal protocols, electrodermal activity, electroencephalography, NASA TLX survey) employed in the study of programming [16,17]. Typically, foveal vision is a well-studied area when it comes to describing the patterns and strategies of source code comprehension [15].

Eye-tracking is an experimental technique used to study the reading and visual search processes by monitoring eye movements. Eye-tracking methods have been used in eye movement studies.
since the 19th century [2]. Today, video-based eye-tracking systems are a standard research tool; these contact-free systems employ a computer for real-time data processing [18, 19].

In the 1970s, McConkie and Rayner restricted the visual area in real time to study the role of extrafoveal information processing during natural-language reading. They used the first gaze-contingent tool to replace a letter (a character) with an X symbol in the words located in the extrafoveal area. When the subjects fixated on a certain position in a word, the characters switched back to normal, replacing the X symbols. The researchers found a clear correlation between the window size and the process of reading [20]. This study opened the door for perception span studies under the window-moving paradigm.

Gaze-contingent tools are now widely used in reading studies [5, 21]. The role of extrafoveal information processing is highly complex and not limited to reading. The fact that subjects with a higher level of expertise in various professional fields also reveal a larger perceptual span is one of many interesting findings in the field [22–24]. McConkie and Rayner define “perceptual span” in terms of the functional demands of reading: ‘readers actually pick up letter and word shape information from a rather limited area [perceptual span] during a [gaze] fixation’ [20]. A window-moving gaze-contingent tool must be built to investigate the role of extrafoveal information processing by both experts and novices in programming.

The key problem of gaze-contingent development is updating the computer screen with low latency following to eye movement. This latency duration should be small enough for subjects not to notice the update. The upper limit for this delay is 60–80 ms [25]. There were no substantial recommendations on how to measure the latency of gaze-contingent software. The latency measurement system for a gaze-contingent tool was to be developed prior to an extrafoveal study.

Currently, there exist no theories that cover all aspects of source code comprehension. Similarly, there are no models for source code
reading that are comparable to E-Z Reader or SWIFT (models of eye-movement control in reading) for natural-language text reading [26, 27]. Source code comprehension differs greatly from both natural-language reading and visual searching. While the models for reading and searching have already been developed and proven [26–29], the researchers have yet to create the models for their subject through data collection and results analysis. The place of fixation and fixations duration are determined by several factors included visual stimuli, task, level of expertise [2, 30], that is why their prediction in such models is not a trivial task. When done, it will be instrumental in understanding the psychology of programming.

The role of extrafoveal information processing during programming is a not researched topic. Understanding of the role of extrafoveal information during source-code comprehension benefits the understanding of the programmer behavior.

1.1 MOTIVATION

The primary motivation for this study is to research the role of extrafoveal information processing to obtain new insights into the cognitive process during programming. This motivation comes from the fields of reading and visual searching together with a special methodological framework.

Source code comprehension is a multi-level process that involves visual processing, mental encoding, and maintenance of mental representation of the program’s source code [31, 32]. Understanding this process is the key to reducing the number of errors in the code and increasing the quality of the final software product. Development of a procedure for expert-level evaluation could assist HR (Human Resources) departments in their tasks. Defining the best strategy for task solving could be equally instrumental to build guides systems. Prediction models of a task’s subjective complexity can help improve the real-time monitoring of software development. This serves as a strong motivation for research into the
psychology of programming.

The development of eye-movement models for source code comprehension is another motivation for the study of programmers' visual attention. Such models can be a milestone in future studies of software development [33]. To build an eye-movement model, we must uncover the roles of foveal and extrafoveal information processing in the first place. Comparing the eye movements of novice programmers with such models could be used for educational purposes. It is likely that expert programmers and novices use information from the extrafoveal area differently [34], but these differences remain unclear. The insights in this aspect benefit training methods of programming. This motivated us to compare the use of the extrafoveal area by experts and novices.

There were no readily available window-moving tools that restricted the viewing area and could be used in programming studies. Neither did we find a framework to be used when designing our experiment. This methodological gap motivated us to develop a gaze-contingent tool.

The size and the form of perceptual span should be studied in a gaze-contingent environment that should be validated against a number of critical parameters. However, we investigated one criterion—the duration between eye movement and the corresponding response on the screen. Latency of a gaze-contingent tool determines whether the screen updates will be imperceptible for the subjects. If subjects perceive screen updates, it affects their behavior and leads to incorrect explanations and results. The perception speed affects the rendering cycle’s duration. We found no framework that could be used to evaluate the latency of gaze-contingent software, so we developed it from scratch. The framework presented below is based on the effect of temporal blindness of the eye-tracker system, which is disadvantage for researchers when it is necessary to measure the system latency from real eye-movements.

We believe it is time to delve deep into all the issues regarding the role of extrafoveal information processing in source code comprehension.
1.2 GOAL OF RESEARCH AND RESEARCH QUESTIONS

This dissertation is primarily concerned with the technical aspects of developing gaze-contingent systems and with investigation of the role of extrafoveal information processing during source code comprehension. In particular, I have solved technical issues to solve psychological questions.

One technical issue corresponds to the development of a window-moving software that works as a gaze-contingent tool. The experiments require that the latency of this software be known, so we also developed a latency measurement system. The gaze-contingent tool was to be used in the experimental part of the study. The measurement of the tool’s latency was also to be carried out during our research.

Our general psychological hypothesis was that expert programmers can process more visual information than novices. To define the latent mechanism of the task-solving process of source code comprehension, we compared a programmer’s work in two conditions—when the extrafoveal area is available and when it is not. Hence, we can estimate the use of the extrafoveal area by both experts and novices.

The gaze-contingent environment and the latency measurement system were built to test this hypothesis. The experiments were set accordingly.

The literature review presented below has left an open question about the role of extrafoveal information processing for experts and novices. There is theoretical evidence that experts in programming use extrafoveal area more effectively (we will discuss this further), but there are no experimental studies in this field so far. We decided to build a new experimental system to find a way to measure its latency and then to obtain new experimental knowledge about the role of extrafoveal information during source code comprehension.

Our objective was to develop a gaze-contingent environment that would enable us to use a common video card with CUDA 1 (CUDA is a parallel computing platform and programming model

1.2 GOAL OF RESEARCH AND RESEARCH QUESTIONS

This dissertation is primarily concerned with the technical aspects of developing gaze-contingent systems and with investigation of the role of extrafoveal information processing during source code comprehension. In particular, I have solved technical issues to solve psychological questions. One technical issue corresponds to the development of a window-moving software that works as a gaze-contingent tool. The experiments require that the latency of this software be known, so we also developed a latency measurement system. The gaze-contingent tool was to be used in the experimental part of the study. The measurement of the tool's latency was also to be carried out during our research.

Our general psychological hypothesis was that expert programmers can process more visual information than novices. To define the latent mechanism of the task-solving process of source code comprehension, we compared a programmer's work in two conditions—when the extrafoveal area is available and when it is not. Hence, we can estimate the use of the extrafoveal area by both experts and novices.

The gaze-contingent environment and the latency measurement system were built to test this hypothesis. The experiments were set accordingly. The literature review presented below has left an open question about the role of extrafoveal information processing for experts and novices. There is theoretical evidence that experts in programming use extrafoveal area more effectively (we will discuss this further), but there are no experimental studies in this field so far. We decided to build a new experimental system to find a way to measure its latency and then to obtain new experimental knowledge about the role of extrafoveal information during source code comprehension.

Our objective was to develop a gaze-contingent environment that would enable us to use a common video card with CUDA (CUDA is a parallel computing platform and programming model invented by NVIDIA) and an office computer to conduct a window-moving study. The window-moving studies use the window within which normal stimuli was displayed (more detailed we will discuss it at the Section 2.3). This gaze-contingent environment should work with different window sizes, forms, and opacity. It should allow operation by people without technical knowledge. This environment should secure the achievement of our study’s goal. The Research Questions for the dissertation are the following:

1. Latency of window-moving software can affect the subject’s behavior and the experiment’s results. What is the latency of ScreenMasker, the custom gaze-contingent tool developed for this thesis and is it acceptable for window-moving studies?

2. What are the differences in the way the extrafoveal area is used by expert and novice programmers?

(a) How are the experts’ and novices’ performances regarding: duration of task solving and number of correctly solved tasks affected by the availability of extrafoveal information from visual stimuli during source code comprehension?

(b) How do experts’ and novices’ behaviors change when the field of view is restricted?

(c) Do expert and novice programmers pay attention to the extrafoveal area when encoding a foveal object?

(d) How do expert and novice programmers report on their source code comprehension experience in the restricted-view mode?

1.3 METHOD

To meet the requirements of each research question we used empirical research methods and paradigms. We used several experimental methods and a variety of equipment to achieve the goal of the study and to find the answer to each question. The SMI RED250 eye-tracker system was used to track the subjects’ eye movements.
and to stream the data to the gaze-contingent software. The ScreenMasker gaze-contingent tool was built to mask the extrafoveal area during the source code comprehension. The latency measurement framework was verified through an acceptance testing and then used to examine the ScreenMasker. We reviewed the literature on the subject to build an appropriate framework and to evaluate the role of extrafoveal information processing during source code reading. The gaze-contingent simulation was followed by an experimental method to ascertain the differences in the use of extrafoveal information by expert and novice programmers.

Each experimental design was carried out using research methods mentioned previously. The numerical data were analyzed using LibreOffice Calc and R-Studio software. The experiments were preceded by the required pilot tests. The reports on the pilot studies are not included in the current dissertation. The use of the eye-tracker in applied research was discussed in a separate paper [35]. The author published one of the pilot studies and the proposed experimental design [36]. That paper investigates the role of programmers’ peripheral vision by means of a gaze-contingent tool. A gaze-contingent pilot study in the field of visual searching was also published [37]. The Visual Evaluation Tool (VETool) was built to perform the visual estimation of ELAN (is a professional tool for the creation of complex annotations on video and audio resources) annotation data and of the duration of eye-movement fixation during a programming task [38,39].

1.4 RESEARCH PROCESS AND ORGANIZATION OF THE THESIS

The dissertation strictly follows the research questions’ logic. The results of each step were published as separate papers and the thesis also follows the chronological order of publications.

This is a multiple-paper thesis that consists of an introduction and original research articles. From the next chapter on, we give an overview of the current situation in two research areas that are
Introduction

the focus of this work—the study of technical problems of developing gaze-contingent systems and evaluating their speed and the psychological study of the perceptual span in various domains (including the psychology of programming). The overview is followed by a report on and retrospective discussion of the research process and its results. This general discussion complements the results and discussions in each of the original publications. The implications of the findings are discussed in more detail in Chapter 5 (Discussion). The thesis ends with an outline of future work.
Human eyes are permanently moving. This movement comes in several types, depending on velocity and density. It is important to stress that eye movements are by no means a chaotic motion. While some micromovements (these will be discussed later) are described as chaotic in some previous research, an opposing hypothesis suggests that two attention systems drive eye movements and human behavior. The first attention system is goal dependent and employs the will to guide attention—a person chooses what to look at and to what to pay attention to [40–43]. The second attention system is stimulus driven—it uses various stimuli to guide the attention [44–46]. More precisely, it uses properties of stimuli, such as saliency.

Vision presumably plays a major role when it comes to perceiving and learning about the world around us. Human eye movements are of particular interest to researchers, and this chapter will look at various types of eye movements and the methodology of their study. We will also explore gaze-contingent software tools.
2 Using Gaze-contingent Systems for Studying Parafoveal Information Processing

The human eye, the organ of sight, has a relatively small area – the fovea–where special photosensitive cells are found in high density. The density and structure of these photoreceptor cells allow us to see clearly. However, the physical angle of clear vision is rather small at about $2^\circ$ [2], hence the metaphor of ‘the spotlight of attention’ in the psychology of visual perception [26, 30].

Human eyes are permanently moving. This movement comes in several types, depending on velocity and density. It is important to stress that eye movements are by no means a chaotic motion. While some micromovements (these will be discussed later) are described as chaotic in some previous research, an opposing hypothesis suggests that two attention systems drive eye movements and human behavior. The first attention system is goal dependent and employs the will to guide attention–a person chooses what to look at and to what to pay attention to [40–43]. The second attention system is stimulus driven–it uses various stimuli to guide the attention [44–46]. More precisely, it uses properties of stimuli, such as saliency.

Vision presumably plays a major role when it comes to perceiving and learning about the world around us. Human eye movements are of particular interest to researchers, and this chapter will look at various types of eye movements and the methodology of their study. We will also explore gaze-contingent software tools
2.1 EYE MOVEMENTS

Various types of eye movements are usually distinguished by their speed: from very slow (drifts) to fast (saccades). Smooth and low-speed eye movements are referred to as drifts by scientists. Their duration varies from 30 to 5,000 ms [2, 47]. Drifts are considered to provide the best conditions for receiving and processing visual information [47, 48].

The ocular microtremor (tremor) is the next type of eye movement. It consists of small and frequent movements with the frequency of up to 250–270 Hz [2, 49]. Tremor is thought to have little influence on vision [49, 50].

Microsaccades are fast, involuntary movements with the duration of 10–20 ms [47, 51]. The role of microsaccades in perception is still unclear and remains the focus of research interest. However, there is a view that microsaccades are not just random, but play an important role in visual information processing [9, 10, 52, 53]. In particular, Engbert and Kliegl claim that “microsaccades can be used to map the orientation of visual attention in psychophysical experiments” [9].

These three types of eye movements (drift, ocular microtremor, and microsaccade) constitute a fixation [2, 54]. A fixation of a motionless object is a dynamic balance of those micromovements. In other words, a fixation is not a stable position of the gaze direction. The role of fixations is of great importance as their analysis can provide valuable and extensive information about subjects’ attention. A fixation gives time to subject to process the information about the image on the retina and to plan the next fixation position (saccadic planning) [48]. The duration of a fixation can indicate cognitive activity based on the hypothesis that cognitive processing of the visual information is performed during fixations [7, 51, 55].

Visual perception studies show that visual attention plays a central role in the control of saccades. Saccades are rapid shifts of the
gaze position that bring the fovea from one selected location to another in a fast and accurate way [2, 54]. The amplitude of saccades varies widely, from $40'–50'$ to $50°–60°$, but in ecologically valid conditions it does not exceed $20°$ [47]. The length, speed, and acceleration of the saccade are in a power-law dependence on its amplitude. Saccades occur when it is necessary to change the fixation position. In some situations, saccades can be quite arbitrary (e.g., when looking at a tree through a window). However, for well-defined tasks (e.g., reading), saccades and fixations follow certain patterns. During left-to-right reading, our eyes fixate only on some point (characters, words) in a line and then switch to the beginning of the next line. The conditions for receiving the optical information are less favorable during the saccades [48]. This fact is used by gaze-contingent systems, when rendering mechanism modifies stimuli, but subject did not perceive this modification [56]. Such systems should work with high speed to prevent the visual lag.

A different typology of eye movements exists in certain narrow research areas that have a particularly specific research goal: smooth pursuits that take place when tracking a moving object; vergence eye movements – convergence or divergence of the optical axes of the eyes; rotary movement – the rotational movement of the eye relative to the optical axis; and several types of nystagmus – a stable oculomotor structure comprising alternating saccades and smooth pursuits movements [54, 57]. Those movements, however, are too specific for our study.

### 2.2 VISUAL INFORMATION PROCESSING

In general, this research will focus on fixations and saccades, which bring to the eyes new visual information. There are two types of photoreceptor cells in the retina, namely, rods and cones. The density of cones is the greatest in the foveal area of the retina. The cones provide information about color and work better in daylight. Their high density provides the greatest visual acuity. Color perception and spatial resolution gradually decrease when moving away from
the fovea. There are only rods in the periphery of the retina; this is why the periphery does not provide information about color. However, the light sensitivity of cones is less than that of rods. Humans can still see in the twilight, though the increased role of peripheral rods and the decreased activity of cones result in our inability to see the colors [7].

The light is focused through the eye lens onto the foveal area from the visual angle of 2° [7]. This means that the human visual system obtains the best quality image from the 2° angle [58,59]. The parafoveal area is next in terms of image quality. It processes images projected onto the retina from the 5° angle [7, 60]. Parafoveal information is obtained by retinal cells with less cone density than at the fovea, which is why it has poorer resolution. Therefore, parafoveal processing can be different from processing performed at the foveal area. The third area of the retinal vision capacity is the periphery, characterized by rod cells domination [5, 7]. Finally, a compressed retina image translates via the optic nerve through the optic chiasma to the visual cortex [61].

The retina representation of the image is the place where humans can switch attention to various areas. That area of attention does not fully correlate with the physical area of the fovea. Hence, human can observe one visual object, but paying attention to another in the same time, that is why it is important for the eye-tracking methodology [62]. Rayner’s research into the perceptual span in reading 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 [5,63]. Studies by Gippenreiter in 1964 showed that the participants recognized objects positioned at a 5°–7.5° (10°–15° full visual angle) from the center of fixation. The peripheral signal was perceived simultaneously with the preparation of the next saccade while the gaze was still at the fixation point [7]. Engbert and Kliegl claim that ‘a key finding in research about visual attention is that the orientation of attention can differ from the orientation of gaze position. In this case, the term covert attention is frequently used to indicate this separation, which
is typically implemented in experimental conditions of attentional cueing’ [9]. We (Bednarik and Orlov) provided the description of the framework as an ability to register the processing of extrafoveal information which allows us to identify its role in solving programming tasks in 2012 [36].

We have not found any research focused directly on the role of extrafoveal information processing during source code comprehension. Therefore, we should resort to the findings made in other domains of visual perception. The source code comprehension is somewhat similar to reading in natural languages, because the texts have similar linear structure [64, 65]. At the same time, it is also close to visual searching because of the selective attention paid to the elements of the source code [22, 66, 67].

We will now discuss how extrafoveal information is studied in the fields of natural-language reading and visual searching. We will describe the major findings of those domains to show the methods, experimental paradigms, and experimental environments used to identify the role of extrafoveal information. And then we will apply the concept of extrafoveal studies to the field of source-code comprehension.

2.3 PARAFOVEAL PROCESSING IN READING

In the course of visual perception, the foveal and extrafoveal information are blended to form a common image of the perceived object. The evaluation of extrafoveal contribution to the perceived image is a non-trivial task, but it had already been solved for natural-language reading. The reading process is aimed at understanding the meaning of written texts. To that end, the words in the text must be identified and analyzed. The visual properties of written texts are strongly determined by rules of the natural language. The comprehension process takes place at all language levels (phonology, morphology, syntax, and semantics) and builds on the orthographic rules of the language in question [5, 21]. Schotter et al. suggest that a person obtains visual information about character sequences and
then encode the spelling to identify letters and the length of words. This process is closely followed by phonological and morphological analysis. Only these stages in their entirety allow recognition of a word from its visual shape. Finally, the semantics and syntax of words join to create the meaning of the sentence [21].

The eye movements during reading are determined by the following factors: word frequency, age of acquisition, word predictability, and word length [5, 21, 68, 69]. Researchers are strongly motivated to build computation models of eye movements to explain what happens during reading when higher-level linguistic processing [26].

The duration of fixation is used to process the foveal and the extrafoveal information and to perform saccadic programming [21, 26]. The place of the next fixation and fixations duration are two elements to be predicted by eye-movements models. That is why foveal information must match with extrafoveal information, especially when the foveal information is more complex. There are several models for eye-movement prediction during natural-language reading (E-Z Reader, SWIFT, etc.) [26, 70]. Nevertheless, there is no unified theory that would exhaustively describe what happens during fixations while reading.

Because of the strict reading rules of natural languages, researchers single out some of the most controversial aspects of parafoveal processing during reading: (1) word skipping, (2) parafoveal-on-foveal effects, and (3) \( n+1 \) and \( n+2 \) (where\( n \) is the fixated word) preview benefit effects [21, 68, 71, 72]. Schotter et al. single out several types of methodologies used to study parafoveal processing in natural-language reading, such as corpus analyses and experimental manipulations. These include gaze-contingent display change experiments under the boundary, the moving window, the moving mask, and the fast priming paradigms [21].

Reading studies of natural languages showed that readers can process parafoveal words [20]. Parafoveal processing takes place both with respect to single words and words in context. The processing time (dwell-time) and the mean fixation duration increase
when the reader’s vision area is limited. This effect can be explained by the inability to pre-process the next word from the preventively hidden extrafoveal area. Hence, it takes more time to program the next fixation or rather some preprocessing of the next word cannot happen and must be done explicitly, and the total fixation duration increases. During left to right reading, humans can obtain useful visual information from an area of up to 14–15 letter spaces to the right of the fixation point [13, 73]. The properties of perceptual span include its size and the degree of asymmetry [5,74].

The factors influencing eye movements during reading (e.g., word frequency and predictability) are reader specific, and their impact is determined by the reader’s level of expertise. For instance in reading of foreign language text more experienced people has wider perceptual span [75]. Numerous studies show that the ability to process parafoveal information while reading natural-language texts is typical of the experts’ behavior [74,75]. Rayner et al. show one more factor that effects perception span: a smaller and more symmetric perception span is typical of older readers. The authors claim that it may be the consequence of the processing of non-foveal information becoming less efficient [74].

A person can use parafoveal information differently while reading and while exploring or searching the visual field [21]. The source code comprehension includes both downstream evaluation of the program—where programmers are directed to calculate the result of the program—and upstream evaluation from the program’s result to input [76,77]. These processes correspond with reading and visual search. This makes it necessary to review the findings of research into parafoveal information processing in visual search.

2.4 PARAFOVEAL PROCESSING IN VISUAL SEARCH

Visual search and scene perception are useful for investigations of visual attention in general and for evaluating the role of extrafoveal information processing in particular. Visual search is a complex activity involving two attention systems: the bottom-up and the
Pavel A. Orlov:

top-down [78–80]. The former is involuntary and stimuli driven—the attention is automatically drawn to a stimulus. The latter is characterized by attention being guided in a voluntary manner.

The object’s visual features (e.g., color, form, lines’ slope) can be detected in a “bottom-up” way or pre-attentively. The feature integration theory of attention suggests that the pre-attentive process goes in parallel at various locations across the visual field. For instance if subject searches one target, distractors’ features processed in parallel with each other. What is important is that pre-attentive process can include the processing of peripheral information in order to identify a stimulus [79,81].

The two systems work together in parallel. Their result is a form of ‘activation map’ that contains peaks of activity at likely target locations [82]. If the stimulus location becomes the most active area on that map, attention will be pointed in that direction. A subject detects targets faster if they appear in locations that commonly contain a target in searching tasks [83]. there is a correspondence assumption in a field of source-code comprehension: Begel suggests that programmer ‘understands code, given the limitations he has on short-term memory and his predilection for reading code out of control flow order.’ [84].

The search strategy is also influenced by the semantic grouping of the elements and their emotional differences [85,86]. For example, people were 1.13 times as likely to use one fixation per group in semantically organized layouts when the groups were labeled [86]. Emotional states influence the process of obtaining extrafoveal information, for example, if the target is the happy face picture [60,87,88]. When several objects are presented in the parafovea simultaneously, the attention is fixated on a more emotionally charged object.

Extrafoveal selection is one task where it is necessary to program a saccade to the endpoint based on the extrafoveal information obtained during the preceding fixation [89]. Typically, this is a multi-fixation task where the information can be extracted from the fovea and the parafovea during each fixation [82,90]. It is assumed
that fixation duration is influenced by the complexity of the foveal information, the extrafoveal information, and the information obtained from previous fixations [3,91].

The predictive coding theory suggests that the brain might use generative models for perception [92]. According to that theory, humans use their experience to predict future information obtained from stimuli after the stimuli are updated. From this perspective, a delay can be noted in the control of fixation time during visual searching [82,93]. The decision about the next fixation depends more on the information from several previous fixations and humans’ experience than on rules of context, such as reading rules. Hence, humans’ experience plays an important role in obtaining visual information.

Gaze-contingent systems and the window-moving paradigm are used in the study of extrafoveal information obtained during visual search. For instance, Van Diepen and d’Ydewalle use these to compare the early extraction of foveal and extrafoveal information. They found that scene perception is influenced more by early foveal masking than by the early peripheral masking [94]. Consequently extrafoveal information can be used differently during the fixation.

Pereira and Castelhano used a gaze-contingent system that performs a semantic change of stimuli. The authors manipulated high-level properties of the scene and found that average saccade lengths grow together with an increase in the amount of visual information available [95]. They showed a stimuli scene with distractors and a target: a 3D room with toys on the floor. In the first instance, the scene had no distractors (toys) in the extrafoveal area. In the second instance, there was no scene (no 3D room) in the extrafoveal area, and only distractors (toys) were presented to the participants. In both instances, the scene was fully available in the foveal area. In the third instance, a black-colored rectangle covered the screen, and the scene was visible only within the foveal circle. It was found that the fixation duration was higher or similar when the scene context was available to the subjects, compared to the scenes that contained only the objects. This contradiction result shows that eye move-
ments are modulated by an interaction between the information obtained from scene context and from object information [95]. Authors found that the average saccade length was longer when there was more information available in the periphery. A more effective selection of target locations was recorded when the scene context information was available in the periphery [95].

Pomplun et al. also used a gaze-contingent window-moving system to evaluate the influence of parafoveal and peripheral cueing and masking effects on global visual search performance and on the pattern of saccadics [96]. The distractors and targets were masked using various geometry. The elements without masks were shown only when captured by the subjects’ gaze. The authors found that effects of cueing and masking were still significant, even for a 10° window. These findings illustrate the considerable contributions of peripheral information processing to visual search performance [96]. The restriction of the visual field (either by a gaze-contingent window-moving technique or by masking) decreases the saccade amplitude, increases initial latencies, increases fixation durations, and influences visual strategies [96–98].

Greene et al. suggest that the effective field of vision (i.e., perceptual span) depends on the task demands. For example, in left-to-right reading, the perceptual span has asymmetry to the right. For right-to-left languages, like Hebrew, the span is asymmetric to the left [99–101]. Therefore, many researchers suggest that ‘... saccades are necessarily preceded by a covert shift of attention to the saccade goal’ [101]. Parafoveal masks were placed along the vertical and horizontal meridians during the task of a visual target localization. New evidence was presented of the perceptual span being asymmetric in the south direction of the visual field. However, when the mask was placed within the northern part of the visual field, the southbound saccades were required more effort from the subject, which was regarded as an unexpected outcome. The authors concluded that ‘the pattern of data suggests that parafoveal information selection and the extent of spatial parafoveal processing in a given direction may be controlled by somewhat different mecha-
Using Gaze-contingent Systems for Studying Parafoveal Information Processing

nisms’ [101].

Cornelissen et al. used a gaze-contingent display to build artificial scotomas and to simulate tunnel vision in a study of eye movements during visual search. In an experiment with artificial tunnel vision, the authors found the unexpected result that the fixation duration increases when the tunnel’s size is reduced: ‘the more we see, the shorter we look’ [102]. Their results are corresponded with previous studies [96–98]. This could also be partly explained through the Feature Integration Theory, which suggests that the basic features are ‘registered early, automatically, and in parallel’ [79]. Hence, this ‘bottom-up’ strategy explains why the fixations did not decreases with decreasing the extrafoveal information. And ‘top-down’ strategy could be used to explain the increasing the fixation duration: subject needs more time for saccadic programs when necessary information is restricted from the extrafoveal area.

As discussed above, gaze-contingent displays are widely used to explore extrafoveal information processing. The window-moving paradigm is typically used with fully framed (black screen with a stimulus image in the foveal circle) or masked extrafoveal objects. Gaze-contingent tools are typically customized or developed by researchers to fit their specific needs. We will use gaze-contingent display and the window-moving paradigm to study the role of extrafoveal vision during source code comprehension.

2.5 EYE-TRACKING METHODS AND GAZE-CONTINGENT SYSTEMS

Despite the fact that eye movements are relatively quick, and their sizes are relatively small, researchers have been making attempts to register them since the 19th century. The first recording of eye movements required direct contact with the eye [2]. In the 19th century, the eye-movement detection process became contact free and comfortable for users. In 1974, Merchant presented a remote eye-tracker (oculometer) for remote measurements of eye direction and pupil diameter [103]. Dual-Purkinje-image (DPI) systems were
Presented from 1965 to 1988—gaze-contingent systems for building artificial scotomas environments [104].

Today, contactless eye-tracker systems have become widespread [18], with manufactures such as SMI, Tobii, and SR Research. Eye-tracking systems now provide researchers with comfortable tools and devices. Current eye-tracker systems use an infrared video-based technique to record the picture of a subject’s eye. The eye is illuminated by an infrared source to produce a reflection on the cornea. The video camera streams the picture to the program that detects the pupils’ position, their dimensions, and the position of the corneal flare [105]. Using these parameters, after calibration, they can map the gaze to points on the screen. The direction of the gaze can be calculated using the vector connecting the position of corneal glare to the center of the pupil [18, 51, 105]. The camera should be fast enough and the processing time sufficiently short, to detect small and fast eye movements.

The minimum speed of an eye-tracker system is about 60Hz [18, 106]. Such systems are used in software engineering, but they cannot detect the smallest eye movements. In contrast, systems with a frequency of more than 250 Hz are used in psychological studies because their speed allows detection of more types of eye movements [51, 107].

Eye-tracking systems collect the raw data of the gaze position. This data can be analyzed by different algorithms to detect fixations and saccades [108, 109]. The analysis can be done after recording as well as in real time. In the latter case, the eye-tracker can stream the information about fixations to the client programs.

The opportunity to obtain real-time data on gaze position allows development of gaze-contingent systems and gaze-contingent displays [18]. Exploring different aspects of gaze contingency paradigms, it is possible to deploy gaze-contingent tools in a number of different fields of applications. Such systems can be used to facilitate computer-human interaction and to develop environments for psychology studies [18].

There are several types of gaze-contingent systems, depend-

Today, contactless eye-tracker systems have become widespread [18], with manufactures such as SMI, Tobii, and SR Research. Eye-tracking systems now provide researchers with comfortable tools and devices. Current eye-tracker systems use an infrared video-based technique to record the picture of a subject’s eye. The eye is illuminated by an infrared source to produce a reflection on the cornea. The video camera streams the picture to the program that detects the pupils’ position, their dimensions, and the position of the corneal flare [105]. Using these parameters, after calibration, they can map the gaze to points on the screen. The direction of the gaze can be calculated using the vector connecting the position of corneal glare to the center of the pupil [18, 51, 105]. The camera should be fast enough and the processing time sufficiently short, to detect small and fast eye movements.

The minimum speed of an eye-tracker system is about 60Hz [18, 106]. Such systems are used in software engineering, but they cannot detect the smallest eye movements. In contrast, systems with a frequency of more than 250 Hz are used in psychological studies because their speed allows detection of more types of eye movements [51, 107].

Eye-tracking systems collect the raw data of the gaze position. This data can be analyzed by different algorithms to detect fixations and saccades [108, 109]. The analysis can be done after recording as well as in real time. In the latter case, the eye-tracker can stream the information about fixations to the client programs.

The opportunity to obtain real-time data on gaze position allows development of gaze-contingent systems and gaze-contingent displays [18]. Exploring different aspects of gaze contingency paradigms, it is possible to deploy gaze-contingent tools in a number of different fields of applications. Such systems can be used to facilitate computer-human interaction and to develop environments for psychology studies [18].

There are several types of gaze-contingent systems, depending on the fields of applications and goals. The first type highlights the gaze control function in the field of computer-human interaction [18]. Such systems include reading support tools [110, 111], tools for gaze-enhanced scrolling [112, 113], eye-typing systems [114], eye-drawing systems [115], and eye-controlled games [37, 116]. The second type of gaze-contingent displays are used in psychological studies of visual perception. This is the type on which we will focus.

McConkie and Rayner were the first to employ a gaze-contingent display for psychological purposes for reading in 1975. They introduced the moving-window paradigm for the study of natural-language reading [20]. The letters in the original text were replaced with an ‘X’ symbol. This made the text unreadable, but the length of words and their positions were the same as in the source text. The resulting text was displayed on the screen. As soon as a reader fixated his gaze on a line, the ‘X’ symbols changed back to the original characters. The visible characters not replaced by the ‘X’ symbol were collectively called a window. When the reader made the next fixation, the window moved on to the next position. This experiment revealed that the moving window’s size has an effect on eye-movement behavior. The reduction of the window’s size increased fixation durations and reduced the length of saccades [20].

In 1979, Rayner and Bertera studied the reading process in the conditions of disabled foveal information processing [117]. They masked the foveal and parafoveal parts of vision by a window-moving framework. When the reader fixated on a word, the original characters were masked with the ‘X’ symbols, but the extrafoveal area remained unchanged. In that experiment, the window played the role of the mask. They found that the mask’s size affects the mean fixation duration: it grew significantly when the mask’s size increased [117].

A part of the retina can lose the acuity of photoreceptors due to a visual field defect called scotoma. Though in the experiment described above the foveal area was represented by ‘X’ symbols, we can consider that Rayner and Bertera were the first to emulate the
experience of subjects with retinal scotoma. This success inspired numerous researchers to use gaze-contingent displays in their studies [118–120]. Gaze-contingent displays with the moving-window paradigm and foveal masking are widely used in natural-language reading studies [5, 74, 121, 122].

Both the window-moving paradigm and the replacement of characters in the original text suit the reading studies well. However, problems such as visual search or simulation of retinal diseases require gaze-contingent systems with additional functionality. An interesting technique for simulating retinal diseases employs two gaze-contingent screens instead of one. The first screen is for the target, while the other is for rendering the simulation [104]. Wensveen et al. used this technique to create an artificial central scotoma in a study of oral reading rates. The scotomata were placed at the right eye’s fovea by the Scotoma Device tool and SRI dual-Purkinje Eyetracker. The artificial scotoma was created on a glass plate between the eye and the target stimuli to simulate retinal pathologies [123].

In contrast to the SRI dual-Purkinje Eyetracker that uses two screens, gaze-contingent displays present visual simulations on the same screen with stimuli. The gaze-contingent displays focus on rendering a combined image that is later translated to the subject’s retina. The image on the screen is contingent on and is continuously determined by the current gaze position measured by an eye-tracking device in real time [124]. Such techniques can be used in scotoma simulators or software with a window-moving paradigm [125].

The motivation for developing gaze-contingent displays was to restrict extrafoveal area during source code comprehension. Such displays can render the informative details on the gaze fixation point and degrade the information in the periphery. Duchowski suggests that ‘The purpose of these displays is usually to minimize bandwidth requirements, as in video telephony applications, or in graphical applications where complex data sets cannot be fully displayed in real-time’ [18]. Duchowski defines two main types
of gaze-contingent applications: the screen-based and the model-based. Both types manipulate the entire screen picture at the pixel-level or at the level of graphical objects and models [18].

Finding a suitable image degradation scheme that could match the foveal acuity is the main difficulty in the development of the gaze-contingent displays mentioned above [18]. A model-based approach is typically used in multi-resolution displays (GCMRDs) [126]. The main idea of GCMRDs is to directly manipulate the model geometry prior to rendering and to reduce resolution of the resulting image [18].

Screen-based systems manipulate the pixel information for image rendering. While masking the remaining part of the screen, the gaze-contingent system’s window can be displayed in a number of different forms and shapes. These types of gaze-contingent displays are useful for psychological research into visual perception [124,127–129]. For example, they allow researchers to study the role of peripheral information in visual search strategies [130,131].

The main purpose that motivated the development of gaze-contingent displays did not prevent them from becoming widely used as window-moving tools in studies of perceptual span and extrafoveal information processing [132–135]. Both types of gaze-contingent displays facilitated research into the impact of simulated visual field defects on the heading task performance [136,137], on walking in virtual environments [138], and on visual search [102].

Visual search and reading are not the only domains of visual attention that can be explored by means of gaze-contingent methodology. For instance, recent research into computer programming focused on the behavior of expert and novice programmers’ reading and comprehension of a source code [65]. There is also research into the role of peripheral vision in programming [36], which further necessitates the development of new tools applicable to research in emerging domains. We presented a gaze-contingent IDE (Integrated Development Environment) and background research without the experimental part [36].
2.6 LATENCY OF GAZE-CONTINGENT DISPLAYS

The development of a gaze-contingent display for existing eye-tracker devices requires a researcher to program a custom tool for each study, we did not find the one and only tool for different studies. This situation can be observed in numerous psychological experiments [139, 140]. Most focus on psychological research goals and prefer not to explore the development.

Latency is one of the most important parameters to consider when developing a gaze-contingent system. Users should not notice the lag between their gaze switching and the corresponding feedback in screen rendering, as this may affect their eye-movement behavior. Duchowski mentions that for the gaze-contingent displays ‘... it is often necessary to distinguish between two main types of causalities: those affecting perception and those affecting performance.’ [18]. Loschky and McConkie show that 5 ms is a no-delay baseline, and 60–80 ms is the upper limit of updating a gaze-contingent display without the lag being consciously perceived by the user [128, 141]. For multiresolutional gaze-contingent displays, it was assumed by Loschky and Wolverton that the update delay of 80 ms significantly increased the time of image blur detection: the extrafoveal area was blurred [25, 142].

Eye-tracking manufacturers provide Software Development Kits (SDK) that can be used by different client frameworks: with MATLAB (via psychotoolbox) or PYTHON (via PyGaze). These approaches require programming experience, knowledge of graphic card programming, and proprietary software [143, 144]. Texture mapping and OpenGL are used for the development of screen-based gaze-contingent displays. The growing processing capabilities of graphics processors (GPU) show that GPUs are well suited for the development of gaze-contingent displays [145]. The latency of the system depended by the development technique.

A system’s latency can be calculated as the sum of the processing time at each step from the eye movement to rendering on the screen. A few studies simply accept this assumption, as this di-
mension is of little importance in terms of the main results of their research \[102,140,146–148\]. The computer screen frequency is a bottleneck in the gaze-contingent environment. A typical 60Hz screen gives a delay of 16 ms for each frame.

In their research, Aguilar and Castet used a CRT (Cathode Ray Tube) color monitor (GDM-F520, Sony, Japan, 100Hz) to decrease the latency. They used a head-mounted eye-tracker (EyeLink II, 500Hz) to investigate the spatio-temporal relationship between the gaze and the actual gaze-contingent image transformation as a function of different oculomotor events \[139\]. They worked on artificial scotoma rendering and focused on two issues: 1) the unwanted triggering of slow eye movements when these eye movements are not required and 2) the lags between the gaze and the scotoma locations that occur during blinking. The authors present a framework with detailed implementation of PsychoPy programming library and OpenGL \[149\], thus stressing the great importance of latency of gaze-contingent systems.

Because of sequential information processing, the latency of a gaze-contingent display depends on many factors. Each step (processing, video capturing, and rendering) delays evaluation and makes its results highly unpredictable. A theoretical latency calculation would be an indirect latency measurement technique: however, the unpredictable factors render it very inaccurate. That is why a direct latency evaluation technique is required to ensure more accurate results. Saunders and Woods provided Direct Measurement of the System Latency of Gaze-contingent Displays for the first time in 2014 \[150\]. The main idea was to blind the eye-tracker by infrared light and to record the time before the response on the screen with a high-speed video camera (with a known frame-rate of 1,000 Hz). This method was presented to be both direct and simple \[150\]. Its disadvantage is that an eye-loss event cannot always be used as an indicator. This can occur when, for example, we evaluate mouse (cursor)-contingent displays or the latency in gaze switching positions. It is clear that new types of latency measurement frameworks and systems should be developed for direct
Pavel A. Orlov:  

2.7 SUMMARY

In conclusion, the extrafoveal information processing plays an important role in reading, visual search, and scene perception. The physical quality of an extrafoveal image is less detailed than that of a foveal one. However, it guides the attention and takes part in saccadic programming. The cortical representation of an image loses its accuracy from the fixation point to the periphery and this representation is completely open for and available to the attention [61]. During a specific task, the perceptual span does not have the same size as that representation. As mentioned by Greene et al., ‘it is tempting to think of the perceptual span as being solely determined by the retinal structure: photoreceptor to ganglion cell convergence ratio increases rapidly into the peripheral retina leading to a rapid decline in visual acuity in peripheral vision’ [101]. But the attention, depended by the expertise level, could be moved to the extrafoveal area [151].

The studies of perceptual span and of the role of extrafoveal information processing make use of the gaze-contingent window-moving paradigm. Typically, researchers either restrict the extrafoveal area by a mask color or blur it. The restricted-view conditions influence eye-movement behavior and can provide evidence of the use of extrafoveal information. Analysis frameworks and experimental designs depend on particular research questions.

The role of extrafoveal information processing during source code comprehension (RQ 2a,b,c,d) should be studied through eye-movement analysis using a gaze-contingent display. This determines the eye-tracking methodology of the study. We found no suitable framework for development of the gaze-contingent displays. Therefore, we should develop a custom gaze-contingent display for our purpose. We decided to develop a user-friendly environment that allows the creation of gaze-contingent textured displays (RQ 1). This gaze-contingent software is based on the window-moving
2.7 SUMMARY

In conclusion, the extrafoveal information processing plays an important role in reading, visual search, and scene perception. The physical quality of an extrafoveal image is less detailed than that of a foveal one. However, it guides the attention and takes part in saccadic programming. The cortical representation of an image loses its accuracy from the fixation point to the periphery and this representation is completely open for and available to the attention [61]. During a specific task, the perceptual span does not have the same size as that representation. As mentioned by Greene et al., ‘it is tempting to think of the perceptual span as being solely determined by the retinal structure: photoreceptor to ganglion cell convergence ratio increases rapidly into the peripheral retina leading to a rapid decline in visual acuity in peripheral vision’ [101]. But the attention, depended by the expertise level, could be moved to the extrafoveal area [151].

The studies of perceptual span and of the role of extrafoveal information processing make use of the gaze-contingent window-moving paradigm. Typically, researchers either restrict the extrafoveal area by a mask color or blur it. The restricted-view conditions influence eye-movement behavior and can provide evidence of the use of extrafoveal information. Analysis frameworks and experimental designs depend on particular research questions.

The role of extrafoveal information processing during source code comprehension (RQ 2a,b,c,d) should be studied through eye-movement analysis using a gaze-contingent display. This determines the eye-tracking methodology of the study. We found no suitable framework for development of the gaze-contingent displays. Therefore, we should develop a custom gaze-contingent display for our purpose. We decided to develop a user-friendly environment that allows the creation of gaze-contingent textured displays (RQ 1). This gaze-contingent software is based on the window-moving paradigm. We did not find a latency measurement system that would fit our goals, we developed a direct latency measurement system to evaluate the latency of our gaze-contingent tool (RQ 1).
The psychology of programming (PoP) focuses on fundamental issues about theoretical and methodological issues of the programming process, language design, acquisition of programming skills, experts' programming skills and job aids, and team and organizational behavior [16].

The source code could be regarded as a medium for human communication [152]. During source code comprehension, a programmer tries to understand what the previous person working on the code expected from the computer. We will, however, focus solely on the programmers' visual perceptual process.

3.1 EYE-TRACKING STUDIES OF SOURCE CODE COMPREHENSION

In the field of PoP, numerous studies have employed eye-tracking as a way to explore visual attention during programming [153]. Since the seminal paper published by Crosby and Stelovsky in 1990 [14], the body of knowledge on the topic has been growing. The cognitive control paradigm is the major paradigm for eye-tracking studies. It assumes that fixation duration is affected by the fixated visual stimulus as well as by semantics [82,154]. In general, analysis of eye movement follows the 'fixation-by-fixation' workflow that is typical for studies of natural-language reading. It is assumed that attention takes place at the point of gaze fixation (the eye-mind hypothesis) [15, 155]. To specify the implementation of the cognitive control hypothesis, we should add that 'the cognitive control hypothesis states that the processing of the properties of the fixated stimulus influences fixation duration, regardless of whether this processing
3 Visual Attention in the Psychology of Programming

The psychology of programming (PoP) focuses on fundamental issues about theoretical and methodological issues of the programming process, language design, acquisition of programming skills, experts’ programming skills and job aids, and team and organizational behavior [16].

The source code could be regarded as a medium for human communication [152]. During source code comprehension, a programmer tries to understand what the previous person working on the code expected from the computer. We will, however, focus solely on the programmers’ visual perceptual process.

3.1 EYE-TRACKING STUDIES OF SOURCE CODE COMPREHENSION

In the field of PoP, numerous studies have employed eye-tracking as a way to explore visual attention during programming [153]. Since the seminal paper published by Crosby and Stelovsky in 1990 [14], the body of knowledge on the topic has been growing. The cognitive control paradigm is the major paradigm for eye-tracking studies. It assumes that fixation duration is affected by the fixated visual stimulus as well as by semantics [82,154]. In general, analysis of eye movement follows the ‘fixation-by-fixation’ workflow that is typical for studies of natural-language reading. It is assumed that attention takes place at the point of gaze fixation (the eye-mind hypothesis) [15,155]. To specify the implementation of the cognitive control hypothesis, we should add that ‘the cognitive control hypothesis states that the processing of the properties of the fixated stimulus influences fixation duration, regardless of whether this processing
was initiated while this stimulus was first foveated or when it was processed extrafoveally (i.e., via extrafoveal or peripheral vision) prior to the first fixation on the stimulus’ [82]. Following this definition of the cognitive control hypothesis, many researchers take into account only foveal information and opt not to direct their attention toward the role of extrafoveal information processing in their studies of source code comprehension.

Several basic eye-movement measurements are used in the PoP for source code comprehension, including [106, 156]:

1. Metrics based on the number of fixations.
2. Metrics based on the duration of fixations.
3. Metrics based on saccades.
4. Pupil size and blink rate metrics.

Based on these measurements, researchers form more complex evaluation techniques based on scanpaths. For example, researchers explored a series of fixations in chronological order, a dwell time of specific reading patterns, time of target’s finding, and visual effort [157–159].

The definition of the area of interest in a source code plays an important role for source code comprehension. Experts use these kinds of chunks to solve problems in various professional domains [160]. Area of interest is generally assumed to be the area containing several source code elements, typically on the same line: for example, in the line \( \text{int } i = 10 * k + 1 \); the area of interest could be \( 10 * k + 1 \).

Fixation duration is used as an indicator of information processing—it shows the level of difficulty in extracting information. The total number of fixations correlates negatively with search efficiency. The proportional time spent on each area of interest and the number of fixations together reflect the subjective importance of the area for a particular programmer [15].
Programming involves a series of diverse tasks [161]. Two cognitive processes are common to all of them: reading the code (chunking) and searching through the code (tracing) [161,162]. In practice, programmers rarely read through or chunk every statement in a program [14,163]. Instead, they trace through the code to find the chunks of code relevant for the task at hand. Beaconing is a term in source code comprehension that is similar to chunking – sometimes they are used interchangeably [155]. The term chunking came into the PoP from the task-solving and decision-making studies [160], while the term beaconing is native to PoP. Beacons are stereotypical segments of a code and serve as typical indicators of a program’s functionality [66,164]. Two types of beacons exist: simple beacons and compound beacons [165].

The process of beacon exploration can be explained by their potential for semantic grouping. The studies of the visual field make it clear that semantically organized groups can be easily manipulated [85,86]. Evidence shows that the understanding of semantic meaning during source code comprehension grows with the level of expertise [155]. That is why experts use beacons more effectively: they pay more attention to them than novices do [66]. Fan reported that gaze scanning patterns for areas with simple beacons barely differ from scanning patterns for areas containing compound beacons [165]. The type of beacon does not affect code scanning sequences.

Programmers used compound beacons to aggregate multiple lines as a meaningful unit during source code reading [166]. Various types of visual strategy and gaze patterns were found during source code comprehension and debugging [167–169]. The patterns of visual exploration are influenced by the level of programming expertise. For example, experts read code in a less linear manner than novices [65]. This difference in visual strategy and gaze patterns makes us question the use of extrafoveal information during source code comprehension. This also brings us to the discussion about the difference in eye-movement behavior of experts and novices in terms of visual perception.
Pavel A. Orlov:
Extrafoveal Vision During Source Code Comprehension: a
Gaze-contingent Tool, a Latency Evaluation Method, and Experiments

Such tasks as visual explorations of source code for review purposes show that errors are effectively identified when there is sufficient time to scan through the code [166]. The authors discovered a particular reading pattern called a scan. During the scan, fixations go through the source code line by line (one fixation per line) with regressions (i.e., backwards saccades to the previously fixated lines). It was suggested that subjects first attempt to understand the program structure by scanning the whole code [166, 170]. As a strategy, one fixation per line provides foveal information from the source code line partly. One central motivation for our study is to find the answer to the question: ‘Do expert and novice programmers pay attention to the extrafoveal area when encoding a foveal object?’

3.2 EYE MOVEMENTS DURING SOURCE CODE COMPREHENSION: EXPERTS AND NOVICES

Several theories explain the differences between experts and novices. In this regard, Gegenfurtner and colleagues consider the following: 1) long-term working memory purposes, 2) the information-reduction hypothesis, and 3) the holistic model of image perception [22].

The long-term working memory hypothesis is based on the idea that the working memory has a limited capacity. This limited capacity is used by humans to store mental representations of the world [171, 172]. The information-reduction hypothesis focuses on the learned selectivity of information processing. Haider and French claim that ‘people learn, with practice, to distinguish between task-relevant and task-redundant information and to limit their processing to task-relevant information’ [67]. Finally, the holistic model of image perception focuses on the extension of the visual span [173]. Following this theory, experts build initial holistic perception of the stimuli. They encode the scene into hierarchical structural components. In theory, this process should use information from parafocal areas [23]. Experts’ behavior is mainly characterized by the follow-
Visual Attention in the Psychology of Programming

1. Fixation durations by experts are shorter than by novices.

2. Experts make more fixations on task-relevant areas. Experts also solve tasks with fewer fixations in general.

3. Experts have a wide perception span and longer saccades. They need a shorter time to fixate on task-relevant areas.

These features of experts’ behavior are rather descriptive and must be explained in greater detail. For example, experts have wide perception span, but what objects from this span draw their attention, and how do they select relevant objects for this particular task? Furthermore, we should consider that experts have fixations that are quite short for their relatively bigger perception span (with a greater number of elements). Hence, experts do not process all elements one by one, but actually select the elements they need to process. There is still a gap in the understanding of experts’ behavior in the field of programming.

Eye-tracking studies of source code comprehension demonstrate that expertise in programming is characterized by distinct patterns of visual attention [15, 174, 175]. For example, expert programmers employ a wider range of strategies during debugging using multiple representations [175]. These strategies develop over time [174]. Expert programmers pay attention on beacons [14, 66]. They also deal with a program’s output during debugging more often than novices [168, 176]. So, experts do not use a single optimal strategy. This constitutes the principal difference to the natural-language reading process [65]. Specifically, there is neither ‘ideal source code reader’ nor ‘ideal observer’ in visual search.

The concept of ideal reader helps to build the eye movement models of reading and it is possible to compare real eye movements with an ideal one. Melloy et al. consider that there are models of visual search: a systematic (memory-full) model and the random (memory-less) model [177]. Human visual search is hypothesized to fall somewhere in between. Najemnik and Wilson show
that humans achieve nearly optimal search performance in a compare with ‘ideal bayesian observer’ for visual search [29]. Programmers’ visual strategy could be potentially modeled by Melloy’s et al. framework or bayesian modelling, but with necessary modifications [29, 177, 178]. The one modification should includes the fact that during source-code comprehension the target is not the visual object itself. Then the size of perceptual span during source code comprehension should be clarified.

The target’s visual properties play an important role in the visual searching process, but the visual representation of source code elements (like camel case and underscore style) has less effect on experts than on novices [179]. Furthermore, no differences were found between the styles of an identifier in terms of accuracy [180, 181].

Experts spend more time viewing the output of the program during a later stage–debugging [182]. This could evidence that experts effectively select what is needed in the process. Further evidence can be found in the study of beacons’ roles in source code comprehension: it was found that more experienced programmers use beacons as guides to focus on the important segments of a program [66].

Authors compared IDEs with different visual source code representations [183]. The results showed that novices who work in command line environments move on to work with graphical IDE with greater ease [183]. These results correspond with the results by Bednarik and Tukiainen in a study of multiple representations of a source code [182]. They show that expert programmers required less time to view the visual representation of a source code (a class diagram) than novices. Experts predominantly deal with text representations that are similar to command line environments.

In 2009, Pietinen et al. showed the fixation points of an expert programmer to novice programmers while explaining an algorithm [184]. They showed that the novices’ visual attention became similar to that of an expert after the expert view had been shown to them. However, that had no effect on the outcome of comprehension. These findings could be explained by the fact that
programmers need peripheral information for some purposes, and when the periphery is masked, the performance decreases because peripheral information is not directly available. Thus, the question about the role of periphery vision in programming remains open.

Bednarik and Tukiainen used RFV (Restricted Focus Viewer) to mask the source code lines above and below the active line that was selected manually with the mouse. They reported that experts are most probably processing much information through peripheral vision during debugging and that is why the processing time is longer. Their study does not explain why the programmers performed this way. Another factor that could affect their results is that in RFV conditions, programmers still glanced toward the blurred areas without selecting them with the mouse [135]. A Research Question that must be answered in this thesis follows: ‘What are the differences in the way the extrafoveal area is used by expert and novice programmers?’

3.3 SUMMARY

There is no doubt that ‘programs have a dual nature—they can be executed for effect and they can be read as communicative entities’ [185]. In terms of visual perception, the comprehension of source code could be compared with both natural-language reading and visual search.

Certainly, we can find similarities between source code reading and natural-language reading: synthesis of elements (words, spaces), line structure, and keyboard as the main input tool (here we consider only text-based source code). Reading of a source code line usually proceeds from left to right. However, source code reading is not natural-language reading: it follows different rules [65, 181, 186]. The attempts to transfer the frameworks and methods of natural-language eye-tracker studies to source code ‘reading’ reveals gaps and discrepancies. The rigid system of natural languages (writing and reading rules) and significant amount of practice in natural-language reading for each of the subjects
makes it impossible to directly implement the scientific methods of reading studies for source code comprehension [186,187]. Moreover, only some steps of natural-language reading mirror those of source code comprehension. For example, verbalization is typically skipped during programming [188].

At the same time, source code reading could be compared to searching tasks. Programmers look for logical answers: for example, to understand ‘how it works.’ There are several similarities to visual searching: the multifixation strategy; a delayed control of fixation duration based on the information obtained from previous fixations; and, probably, the strategy of map activation that links peaks of activity to likely locations of source code elements. Nevertheless, there are differences as well. Studies of visual search are focused on identification of a visual target among distractors. This could hardly be compared to source code reading.

The gaze-contingent window-moving paradigm is the most common method used to study the role of extrafoveal information in reading and searching. We decided to use it in our study of source code comprehension. During programming, eye-movement behavior is different in experts and novices. This could be due to the different use of extrafoveal information. The foveal area of the retina collects data from the visual angle of 2°. If a programmer is sitting 60 cm in front of the computer screen, this angle corresponds to a circle with a diameter of 90 px (for a typical 1280×960 px screen) approximately. This foveal area allows the programmer to obtain several source code elements. A programmer must select the elements to direct his attention at and then involve these elements in the process of encoding during source code comprehension. Thereafter, a programmer explores a bigger visual area than foveal area to decide where to saccade next. In our study, we decided to mask the extrafoveal area by means of a gaze-contingent software to check the influence of the extrafoveal information on experts’ and novices’ eye movements.
4 Summary of Publications

The last few years have seen a surge of interest in understanding the source-code comprehension process. Numerous studies have been conducted to evaluate the visual strategies and gaze patterns of expert programmers [15,65,66,168]. Gaze-contingent methods, which are widely used in visual perception studies, can be employed to investigate the process of source-code reading.

The main findings and results of the original publications are provided below with respect of the main focus of this dissertation: to find the role of extrafoveal vision during source code comprehension. We summarize the main goals and objectives of the reported studies as well as the evaluation methods and software tools.

4.1 PAPER I: LOW-COST LATENCY MEASUREMENT SYSTEM FOR EYE-MOUSE SOFTWARE


4.1.1 Background and Aims

The first article discusses the latency measurement system that we developed as a user-friendly and low-cost technique for a eye-mouse software. The motivation was to design a system for latency evaluation of gaze-contingent software and eye-mouse software in general.

Latency is the time between an eye movement and the corresponding screen update. Low latency is vital for solving visual tasks, because high latency can influence the eye-movement behavior significantly. The computer screen must be updated following
an eye movement within the threshold period of 80 ms [25,142].

The aggregate of delays at each step of data processing comprises the delay of the entire gaze-contingent system. The delays at each step are hard to ascertain due to different hardware manufacturers, different algorithms of intermediate software (e.g., client-server, drift correction procedure, etc.) Furthermore, there are limits of eye-tracker frequency synchronization and screen rendering. These factors make the delay estimation difficult, although it is a key factor of gaze-contingent software. The current project aims to determine a way to measure the latency of gaze-contingent systems easily, quickly, and cost effectively.

4.1.2 Results and Discussion

We present the architecture used for the direct measurement of eye-mouse latency. The method of measurement is based on the high-speed video detection of the subjects’ actions (mouse moving or eye switching) and the screen updates in the same video frame. The system should obtain eye images with a resolution that is sufficient for visual detection of both events in one video frame. Figure 4.1 shows a design scheme. The high-speed camera should be inexpen-

![Figure 4.1: The design scheme of the latency measurement system for eye-mouse software.](image)

The aggregate of delays at each step of data processing comprises the delay of the entire gaze-contingent system. The delays at each step are hard to ascertain due to different hardware manufacturers, different algorithms of intermediate software (e.g., client-server, drift correction procedure, etc.). Furthermore, there are limits of eye-tracker frequency synchronization and screen rendering. These factors make the delay estimation difficult, although it is a key factor of gaze-contingent software. The current project aims to determine a way to measure the latency of gaze-contingent systems easily, quickly, and cost effectively.

4.1.2 Results and Discussion

We present the architecture used for the direct measurement of eye-mouse latency. The method of measurement is based on the high-speed video detection of the subjects’ actions (mouse moving or eye switching) and the screen updates in the same video frame. The system should obtain eye images with a resolution that is sufficient for visual detection of both events in one video frame. Figure 4.1 shows a design scheme. The high-speed camera should be inexpensive and accessible for the eye-mouse developers. Our choice was a GoPro HERO3+ Black Edition camera: 848×480 in the WVGA mode and 240 fps. It meets our requirements and it costs about $400. To evaluate the latency of an eye-mouse tool, we must find the video frame with the initial action manually and then scroll the frames until we reached the frame with the corresponding screen update.

We conducted a case study to test our system on three types of software. We performed thirty trials with initial events for each type. For mouse-based interaction, the event was ball pushing (to prevent head tremor); for the gaze-contingent tool, the gaze was switching from one desktop icon to another. A mirror was used to capture fixation of eye switches in the same video frame. Figure 4.2 shows a summary of the latency tests of three testing cases.

A classical mouse-based interaction tool was the first type of software tested. We found the cursor’s latency (titled Cursor at Figure 4.2) to be enough for speed-demanding studies: < 80 ms (M:30.67 ms, SD:12.17 ms).

![Figure 4.2: The latency of three testing cases, ms.](image)

The second type of software that we tested used the coordinates of the mouse cursor to render the next image (titled Subject tool in a...
section Subject tool with mouse at Figure 4.2). During this process, the measured latency went above the threshold level of 80 ms (M:93.33 ms, SD:17.83 ms). We concluded that image rendering methods that involved obtaining information about the cursor’s coordinates were unsuitable for the developing of gaze-contingent tools.

A gaze-contingent tool not using cursor coordinates was the third type of software tested. It renders the image based on the gaze position (titled Eye-tracker in a section Subject tool with eye-tracker at Figure 4.2). As this tool’s delay was below the threshold level (M:26 ms, SD:12.44 ms, MAX: 44), we concluded that it can be used in the gaze-contingent studies of visual perception.

Our system is important to evaluate display rendering software running in background mode. We believe that gaze-contingent environments should be evaluated by a direct latency measurement system similar to ours. We produced a demonstration video to show how our system works in real-life conditions (youtube.com/watch?v=eGKSeMrALcY).

The development of the gaze-contingent tool is further discussed in Paper II, which explores the window-moving paradigm.

4.2 PAPER II: SCREENMASKER: AN OPEN-SOURCE GAZE-CONTINGENT SCREEN MASKING ENVIRONMENT


4.2.1 Background and Aims

The second study used gaze position on the screen to develop a gaze-contingent system. We looked at the effectiveness of various programming frameworks in terms of how and when they reach low-latency level. Then we introduce the architecture of ScreenMasker and relevant case studies.
Gaze-contingent systems are generally used for studies of extrafoveal information processing and the perceptual span’s properties. Such tools are custom-made builds developed to suit a particular experimental design and study goals. We did not find general gaze-contingent window-moving tool that could be used with a variety of stimuli. The literature review of perception span research showed that stimuli can come in a variety of sizes or forms. They can also have different levels of asymmetry around the gaze fixation point. These aspects, as well as the latency threshold level, lay down the requirements for the development of a universal system.

4.2.2 Results and Discussion

We developed the ScreenMasker: a software that allows customizable gaze-contingency experiments with a textured (masked) computer screen. The mask (the tile) can be selected by a researcher as a necessary black-and-white picture. Each white pixel of that tile will be rendered by the ScreenMasker as transparent and the tile will be reproduced on the entire screen. The black pixels of the tile’s image will be non-transparent. The resulting mask will be run as the top layer of the Windows system. Figure 4.3 shows a running ScreenMasker above text processes windows and the Minesweeper game’s window.

Researchers can define the color for the visible pixels and customize them according to their needs. Pixels that remain transparent allow to see the window of a computer program displayed at the background of the screen (e.g., a text processor or a Web browser). The combination of transparent and non-transparent pixels allows the user to present the stimuli with the required amount of visual noise. As mentioned, the pixel patterns are rendered at the upper layer, so the mask covers small elements of the screen and leaves bigger elements available for perception. The words and letters on the screen are ‘broken up’ (separated) by the mask, but the bottom of the user interface still can be identified. This can be used in experiments where it is necessary to break single characters while the
Pavel A. Orlov:

Figure 4.3: The ScreenMasker runs above desktop applications. Gaze fixation point is the one normally visible area. Extrafoveal area is covered by mask and it is hard or impossible to read visual elements there.

other part of the text should remain intact. This is only one example of the numerous experiments where the ScreenMasker can be applied.

The ScreenMasker can be used with SMI eye-tracers (SensoMotoric Instruments GmbH) by default. It also is distributed under an open-source license and can be modified to work with other eye-tracking systems. The gaze coordinates that are provided by an eye-tracker server (SMI iViewX by default) are used to locate the window. The unmasked area can be rendered at the position of the subject’s gaze similar to a window in the window-moving paradigm.

The window is called a stencil. The stencil allows for viewing of the stimuli and can be set in different forms, sizes, and transparency levels. Researchers can decide what size and what form of visual screen area they would like to use in their study. The stencil itself is a picture in a portable graymap format. A picture for the stencil can also be asymmetric. The stencil’s size is unlimited, but we suggest that the system’s latency should be determined prior to experiments.

We carried out a comprehensive latency evaluation to demonstrate the latency measurement procedure and to evaluate the latency with SD, ms.

<table>
<thead>
<tr>
<th>Eye-tracker frequency, Hz</th>
<th>Screen: 60 Hz</th>
<th>Screen: 144 Hz</th>
</tr>
</thead>
<tbody>
<tr>
<td>200x200</td>
<td>67.1</td>
<td>67.67</td>
</tr>
<tr>
<td>400x400</td>
<td>27.15</td>
<td>73.55</td>
</tr>
<tr>
<td>800x800</td>
<td>25.62</td>
<td>28.2</td>
</tr>
</tbody>
</table>

Figure 4.4: The comparison of latency time on three frequencies (60 Hz screen) and on three stencil sizes and 250-Hz eye-tracker frequency (144 Hz screen).

We used CUDA technology for pixel processing during the application of the stencil to the screen mask to receive the utmost benefit from low latency. CUDA provides parallel processing of data on a GPU, which explains the impact of various stencil sizes on the system is insignificant. The use of GPU calculations for gaze-contingent displays was recommended by Duchowski [18]. We, however, took this idea to the next level by building a user-friendly...
Summary of Publications

tency of the ScreenMasker on a variety of hardware and with various stencil sizes. To that end, we used our Low-cost Latency Measurement System [189] [Paper I]. We found that the system has a mean latency of about 67–74 ms on 60 Hz screen and 25–28 ms on 144 Hz screen. In both cases, these latencies fall under the 80 ms limit. Moreover, the stencil size (200 px×200 px, 400 px×400 px, 800 px×800 px) does not impact the ScreenMasker’s latency (see Figure 4.4). These results allow us to recommend the ScreenMasker for studies requiring latency that is imperceptible to the subjects: for example, research that requires extrafoveal information or the studies of perception span.

![Figure 4.4: The comparison of latency time on three frequency types (60 Hz screen) and on three stencil sizes and 250-Hz eye-tracker frequency (144 Hz screen).](image)

We used CUDA technology for pixel processing during the application of the stencil to the screen mask to receive the utmost benefit from low latency. CUDA provides parallel processing of data on a GPU, which explains the impact of various stencil sizes on the system is insignificant. The use of GPU calculations for gaze-contingent displays was recommended by Duchowski [18]. We, however, took this idea to the next level by building a user friendly
tool with graphical user interface for gaze-contingent studies.

The main function of the ScreenMasker is to cover the screen with a mask and apply a gaze-controlled stencil (window) to it. The calibration procedure and other possible functionality of the stimuli-presenting software depend on the requirements of a particular experiment and are not provided by the ScreenMasker. The eye-tracking data can be stored by the eye-tracker server itself and the data format depends on the manufacturer. The ScreenMasker can be used with SMI Experiment Center. It is also possible to use the ScreenMasker’s source code to develop custom builds.

We presented the ScreenMasker, a gaze-contingent window-moving tool that satisfies the threshold of 80 ms (both mean and max values), which was measured by our Low-cost Latency Measurement System. Thus, we have developed a verified tool required to investigate the role of extrafoveal information processing during source code comprehension.

Discussion on extrafoveal information processing is provided in Paper III, which deals with source-code comprehension by experts and novices under restricted-view conditions.

4.3 PAPER III: THE ROLE OF EXTRAFOVEAL VISION DURING SOURCE CODE COMPREHENSION


4.3.1 Background and Aims

In the third study, we used the ScreenMasker software to conduct a gaze-contingent window-moving experiment. In this experiment, we restricted the visual area with a mask and left a viewing window that was moving synchronously with the gaze in real time.

There did not exist direct evidence concerning extrafoveal information processing during source-code comprehension. Generally,
the extrafoveal area is used to plan the place of the next fixation or to determine the next visual object to be fixated upon. Both low-level and high-level information about the object of future fixation are processed during the planning. The former includes the form of the object: the features of the stimulus image and its geometry. The latter includes the object’s meaning, its semantics. Furthermore, the extrafoveal information can help encode the foveal object to a mental representation of the source code. In this case, the extrafoveal area is used by the covert attention process without macro-saccadic eye movement. Extrafoveal information processing can be influenced by the expertise level. Thus, our goal in this study was to assess and compare the process of obtaining extrafoveal information by experts and novices.

The first question that we wanted to answer concerns task-solving by experts and novices under restricted-view conditions, i.e., when the extrafoveal information is covered by a mask: ‘Does the restricted-view mode affect the task-solving performance of the experts and the novices?’ The measurement of the actual performance provides direct data about the role of extrafoveal information processing during source code reading by experts and novices.

The second question concerns behavioral patterns: ‘How will the behavior of programmers change when extrafoveal information processing is restricted?’ This question required us to register and expose the process of extrafoveal information processing by both experts and novices.

The third research question was about comparing experts’ and novices’ behavior under restricted-view conditions: ‘How will the behavior of the expert programmers differ from that of the novices, when the extrafoveal information processing is restricted?’

The last question related to the nature of information processing during fixation: ‘Do programmers shift attention to the extrafoveal area when encoding a foveal object?’
4.3.2 Results and Discussion

The experiment involved 12 experts and 12 novices. We used the ScreenMasker to restrict the visual area. The mask was designed to restrict the letters in the words in the given source code. It was still possible to identify the words' form (geometry) but the words were made illegible enough to prevent the subjects from understanding their meaning. We used the mask pattern with pixels texture that corresponds with font size and font colors of the source-code to restrict the words in extrafoveal area. The viewing window was a 2° circle (the size of foveal area) with a smooth transition (a gradient) to the mask pattern. The fully restricted region started at the 5° angle (see Figure 4.5).

Our subjects were asked: 'What will be the output of the Java program?' At the beginning of the experiment, all subjects performed test trials (five test tasks without time limits) to become familiar with the gaze-contingent environment. The experiment had a repeated measures design to provide control over the results. Twelve subjects (six experts and six novices) solved five tasks in restricted-view conditions and then five tasks in unrestricted conditions. The other twelve subjects (six experts and six novices) solved five tasks in unrestricted-view conditions and five tasks in restricted-view conditions. In total, the subjects solved 240 tasks (24
Summary of Publications

subjects × 10 tasks for each).

Our study yielded a number of interesting results, the main findings are as follows:

1. Experts made more mistakes in the restricted-view mode than in the free-viewing mode.

2. Experts spent more time solving the tasks in the restricted-view mode than in a free-view mode, while novices’ task-solving time was similar in both modes. In the restricted-view mode both experts and novices spent the same time for task solving.

3. The gaze behavior of programmers changes when the processing of extrafoveal information is restricted.

4. The behavior of expert programmers becomes similar to that of novices.

5. Both expert and novice programmers showed an increased fixation duration when processing the foveal and the extrafoveal information.

One of results is that the experts’ performance is affected by the restriction of extrafoveal area in both measurements: the number of correct answers and the task-solving duration. When the experts could not obtain information from the extrafoveal area, they spent more time solving the tasks. In both viewing conditions, the experts’ task-solving duration was significantly lower than that of the novices. The novices’ duration of task-solving was the same in both conditions. We explain this observation by the experts using some information about the task from the extrafoveal area, which helps them to solve tasks faster. They still coped with their tasks well (more correct answers compared to novices), but their task-solving speed declined. In general, the experts’ performance lowered under the restricted-view conditions, which indicates that they are using extrafoveal information. The novices’ performance was not much
Pavel A. Orlov:

affected by the restricted view, which means they do not use extrafoveal information in the way that experts do.

The next step of analysis showed how behavior changed when the extrafoveal information was restricted. We found that the experts performed more fixations in the restricted view than in the unrestricted view. On average, the experts made about 30 percent more fixations under restricted-view conditions. As for the novices, the difference was not significant, but they performed fewer fixations in the restricted-view mode.

The restriction of the extrafoveal area affected fixation durations in both groups. For both expertise levels, fixations were longer under restricted conditions. The experts’ fixations were, on average, 100 ms longer in the restricted-view mode, while the novices’ fixations were 40 ms longer. The effect was statistically significant only for experts. We can explain this effect by the fact that the subjects use extrafoveal information during fixations and, when that information is unavailable, the time of information processing during fixation increases. The surprising fact is that in the restricted-view mode, the experts’ fixations became similar to those of the novices, which was not the case in the free-view conditions.

The means of saccadic amplitude were similar for experts and novices under different viewing conditions. We believe it is explained by the fact that there is enough information to program long saccades even when the extrafoveal area is restricted. However, the saccadic amplitude remains similar in both viewing conditions. During restriction of extrafoveal area it was still possible to see the geometry of the source-code lines. This visible geometry could be enough for the saccadic planning. But the meaning and semantic of a restricted piece were available only after saccade, when this piece moved to the foveal area.

We concluded that experts’ behavior is influenced by the restriction of extrafoveal area. Experts do use extrafoveal information while reading source code. Fixation durations become longer and the numbers of fixations increase when extrafoveal information is inaccessible. Novices’ behavior is not influenced by the restriction
of extrafoveal area: the numbers of fixations and fixation durations were statistically the same for both viewing conditions. Novices use extrafoveal information differently from experts or do not use it at all.

The final step was to understand why programmers use extrafoveal information. We studied the influence of the restricted view on the fixations preceded by saccades with length below 2.5° and to those preceded by saccades over 2.5°. Theoretically, the fixations that occur before a short saccade do not involve extrafoveal information for saccadic planning. Thus, we subdivided fixations into foveal (occurring before short saccades under 2.5°) and out-of-foveal (occurring before long saccades over 2.5°) categories. We found that the proportion of foveal and out-of-foveal fixations is different for the experts and for the novices in both viewing modes. However, the means of proportion are similar in both viewing modes and for both groups.

In the restricted-view mode, the experts’ foveal fixation durations were statistically the same as those of the novices. The situation was different in the unrestricted mode. The foveal fixations became longer only for the experts. Hence, experts use extrafoveal information even when they do not plan saccades longer than 2.5°. In the unrestricted mode, the durations of out-of-foveal fixations were similar for both experts and novices. For experts, the cost of saccades increases in the restricted-view mode that is why the out-of-foveal fixations became longer. Interestingly, the experts’ out-of-foveal fixations became longer than those of the novices in the restricted-view mode. By contrast, the novices had similar out-of-foveal and foveal fixations in both viewing conditions. We concluded that experts use information from the extrafoveal area not only for saccadic planning but also to encode foveal objects.

The analysis of the rate of foveal and out-of-foveal fixations shows that the restriction of extrafoveal area generally affects the behavior of experts more than that of novices. Again, extrafoveal information plays a bigger role for experts. The visual analysis of the diagram showing saccadic amplitudes and fixation durations
Pavel A. Orlov:
Extrafoveal Vision During Source Code Comprehension: a
Gaze-contingent Tool, a Latency Evaluation Method, and Experiments

allows us to see that the experts’ pattern is more vertically compact
when compared to that of the novices, especially where the angle of
saccadic amplitude ranges from 1.5° to 2° (see Figure 12 at [Paper
III]).

These findings suggest that experts use extrafoveal information
more effectively than novices. In the restricted-view mode, the ex-
erts, however, could not use their advantage of parafoveal percep-
tion of the semantic information and their eye-movement behavior
and performance became similar to that of novices.

4.4 PAPER IV: PROGRAMMERS’ EXPERIENCES WITH WORK-
ING IN THE RESTRICTED-VIEW MODE AS INDICATIONS
OF PARAFOVEAL PROCESSING DIFFERENCES

Orlov, P., Bednarik, R., & Orlova, L. (2016). Programmers’ expe-
riences with working in the restricted-view mode as indications of
parafoveal processing differences. In L. Church (Ed.), Proceedingsof
the 27th Annual Workshop of the Psychology of programming interest
group PPIG 2016 (pp. 96–105). University of Cambridge, UK.

4.4.1 Background and Aims

Our fourth study employed the same experimental design used in
our previous work to explore the role of extrafoveal vision during
source-code comprehension. The procedure, the tasks, and
the environment are described above. The main aim of the fourth
study was to understand programmers’ behavior in restricted-view
conditions through analysis of their reported sentiments. We as-
sumed that the experts’ and the novices’ different experiences un-
der restricted visual conditions would produce different feedback
because they use the extrafoveal area differently.

To understand the programmers’ sentiments, we performed a
textual analysis of their self-reports collected at the interview after
the experiments. We selected the discourse analysis as our study
method because it is distinctly different from the method we used in
the previous study. We use different methods of analysis to provide
different point of views on the one problem. It allowed us to look at
the role of extrafoveal information processing from a different an-
gle. This can certainly extend our knowledge about programmers’
working process when the extrafoveal area is unavailable. Thus, we
analyzed how experts and novices reflect on their experience.

In the current study, we checked the following hypothesis: if
the experts use the extrafoveal area, they will give feedback on its
restriction differently from the novices who do not use extrafoveal
information.

There were two central questions in the current study. The first
research question was: 'What topics can be singled out in the ex-
erts’ and the novices’ reports?' To find an answer, we analyzed the
content of self-reports to divide it into topics and we did not use
any pre-defined categories.

The second research question concerned the differences between
the experts’ and the novices’ reports: 'What are the differences
between the experts’ and the novices’ reports on working in the
restricted-view mode?' To find those differences, we used the top-
ics (categories) identified in the previous step. We broke the self-
reports down by these categories: what the subjects were talking
about, what was interesting, or what seemed important to them.

4.4.2 Results and Discussion

Thirteen male participants were involved in the experiment: six
experts (aged 19–25) and seven novices (aged 18–21). The program-
mers solved tasks in the restricted-view mode. The extrafoveal area
was restricted by the ScreenMasker to prevent the processing of
extrafoveal objects even when the subjects paid attention to them.

After the experimental sessions, the subjects verbally described
their feelings and experiences during the experiments in free form.
The analysis of self-reports allowed us to single out six categories
of topics and one emotional state, namely:

1. Analysis of the working process.
Two categories were present in the experts’ answers only. First, the expert programmers gave their assessments of experimental conditions (i.e., the gaze-contingent restricted-view mode). Second, they used emotionally charged and abusive language, which we singled out as an independent category.

In both groups, comments about the working process comprised about 50 percent of utterances (see Figure 4.6). The programmers described the strategies employed. It was unexpected because professional programmers tend to focus on more specific technical information when generating talk-aloud data [190, 191]. We expected
Summary of Publications

that experts would describe the strategies they used with a large proportion of special words [168,192,193]. However, the professionals preferred to describe the experimental conditions instead. They spoke of the gaze-contingent restricted-view mode and reported on how they dealt with a habitual source code in a different way as compared to normal conditions. Conversely, the novices reported nothing on the experimental conditions.

While discussing the subjects’ feelings about the effect of the restricted-view mode, both groups reported the conditions to be disturbing. Both groups reported that they perceived the mask pattern in the parafoveal area. The experts expressed themselves in a more negative way. For example, expert E1 reported: ‘I had psychological pressure due to the restricted view. It was distracting me.’ Expert E2 said: ‘It was difficult to see the whole line – it’s annoying.’ By contrast, novice N1 reported more reservedly: ‘This white viewing window was a little annoying.’

Interestingly, the professionals’ feedback was more emotional. For example: ‘It was awesome; It was tough; It distracted me quite a deal; I am psychologically crushed’. The emotionally charged utterances were used by some experts but were not used by the novices. The professionals expressed themselves both more emotionally and more nervously. We concluded that the unavailable extrafoveal objects of the source code evoked stronger feedback from the professionals. Our results support the previous research into this area that found that professionals universally dislike the restriction of viewing conditions [34].

We attribute this increased emotional response to the fact that experts use extrafoveal objects for source code comprehension. The visual information is obtained from the extrafoveal area, especially when human attention shifts from the foveal object during a fixation [74]. The previous studies report that more emotionally charged stimuli are more likely to be processed even when located in the extrafoveal area [60]. Experts have more expectations about parafoveal objects because they are generally more aware of the relevance of information to which they choose to direct their attention. PRO-
professionals are better at predicting the next object of their attention before switching the gaze position—just like professional readers of natural-language texts [74]. When the next object of attention is masked, the information becomes unavailable. The betrayed expectations result in an increased emotional feedback. This supports the importance of extrafoveal information.

We also found that both groups generally report on the working process, life hacks, and source-code features in the same proportion. The content of self-reports was different: the professionals discussed possible optimization, while the novices were speaking about syntax. We can consider these as a different levels of attentional focus. The novices focused more on the lower level of understanding. They talked about general syntax and the naming of variables. This approach was also typical of task-solving in the unrestricted mode. A possible explanation is that the restricted-view mode does not influence mental representation of the program. Experts and novices remain at the same level of abstraction regardless of the conditions of source code comprehension. Further work must be done to confirm this conjecture.

We concluded that programmers can direct their attention to the extrafoveal objects that are located at places different from the direction of visual attention focus (i.e., the place of gaze fixation). When extrafoveal objects are visually unavailable and the information about them is hard to obtain, there occurs a stronger emotional feedback due to disappointed expectations. Expert programmers routinely use extrafoveal objects in source code comprehension.

Discussion

The role of extrafoveal information is widely studied in the fields of reading, visual search, and scene perception. But numerous questions remain about the use of such information, even in those domains. For example, in 2012 Schotter et al. wondered: 'How many words can readers process at the same time? If readers can obtain semantic information from the upcoming word, what conditions are necessary for that semantic preprocessing to occur?' [21]. Similar questions must be addressed with respect to source-code comprehension.

In this thesis, we ascertained the role of extrafoveal information processing during source-code reading. We also compared expert programmers with novices when using the extrafoveal information. We conducted a literature review of the studies of extrafoveal information processing in related domains. We were fully aware of the differences between natural-language reading and source-code reading. With those differences in mind, we used only the basic principles of the experimental designs used to study extrafoveal information processing and perceptual span. Furthermore, we realized that source-code comprehension is similar in some respects to visual searching. Thus, we consulted the visual-search domain to gain insight into the methods and theories applied to explore the parafoveal and peripheral areas. Eventually, we defined a method for our study: the window-moving method based on the gaze-contingent paradigm. While the window-moving paradigm with mouse control was used in studies of understanding of mathematical equations (in 2003) [194] and source-code comprehension (in 2005) [135], our approach, in which a window is controlled by gaze in real time, is original in the field [135, 194].
5 Discussion

The role of extrafoveal information is widely studied in the fields of reading, visual search, and scene perception. But numerous questions remain about the use of such information, even in those domains. For example, in 2012 Schotter et al. wondered: 'How many words can readers process at the same time? If readers can obtain semantic information from the upcoming word, what conditions are necessary for that semantic preprocessing to occur?' [21]. Similar questions must be addressed with respect to source-code comprehension.

In this thesis, we ascertained the role of extrafoveal information processing during source-code reading. We also compared expert programmers with novices when using the extrafoveal information.

We conducted a literature review of the studies of extrafoveal information processing in related domains. We were fully aware of the differences between natural-language reading and source-code reading. With those differences in mind, we used only the basic principles of the experimental designs used to study extrafoveal information processing and perceptual span. Furthermore, we realized that source-code comprehension is similar in some respects to visual searching. Thus, we consulted the visual-search domain to gain insight into the methods and theories applied to explore the parafoveal and peripheral areas. Eventually, we defined a method for our study: the window-moving method based on the gaze-contingent paradigm. While the window-moving paradigm with mouse control was used in studies of understanding of math equations (in 2003) [194] and source-code comprehension (in 2005) [135], our approach, in which a window is controlled by gaze in real time, is original in the field [135, 194].

The next step was to build a gaze-contingent software for the window-moving experiment. From the literature review, we found the latency threshold for gaze-contingent displays. We needed a
direct latency measurement method to verify our gaze-contingent software. No such software had yet been developed. Thus, we approached the research goals in the following order: first, we focused on developing a latency measurement system (RQ1), then we focused on the gaze-contingent tool (RQ1), and, finally, we proceeded to the issue of extrafoveal usage during source-code comprehension (RQ2 a, b, c, d).

The latency measurement system was developed for a wide research community and is distinguished by its low price [Paper I]. We used a high-speed GoPro camera to fixate the initial action and the corresponding screen update. The initial action could be either mouse movement or gaze switching. The delay is defined by the number of video frames between the initial action and the screen update. It is necessary that the initial event and the screen response are captured in the same frame. We used a mirror to provide the picture of the eye switching.

We then applied our latency measurement system to the ScreenMasker, a gaze-contingent tool that we developed [Paper II]. The ScreenMasker is based on the window-moving paradigm. It meets the requirements and recommendations for gaze-contingent display development. To ensure low latency, CUDA technology was used to process pixels on the GPU. We answered first Research Question that latency of the ScreenMasker was about 27 ms (on 250 Hz eye-tracker, 144 Hz screen). It was lower then limit (80 ms) and quite stable for various sizes of the window (the stencil), which allowed us to use the ScreenMasker in a variety of research problems (RQ1).

Finally, we carried out an eye-movement experiment to determine whether and how readers use extrafoveal information during source code comprehension [Paper III, Paper IV].

The answer for Research Question 2a is that extrafoveal information is not vital for task-solving (the subjects still solved the tasks). However, the availability of extrafoveal information allows experts to solve tasks more quickly. The programmers behavior changed when extrafoveal information about source code becomes unavailable. The changes in eye movements are stronger for experts than

Direct latency measurement method to verify our gaze-contingent software. No such software had yet been developed. Thus, we approached the research goals in the following order: first, we focused on developing a latency measurement system (RQ1), then we focused on the gaze-contingent tool (RQ1), and, finally, we proceeded to the issue of extrafoveal usage during source-code comprehension (RQ2 a, b, c, d).

The latency measurement system was developed for a wide research community and is distinguished by its low price [Paper I]. We used a high-speed GoPro camera to fixate the initial action and the corresponding screen update. The initial action could be either mouse movement or gaze switching. The delay is defined by the number of video frames between the initial action and the screen update. It is necessary that the initial event and the screen response are captured in the same frame. We used a mirror to provide the picture of the eye switching.

We then applied our latency measurement system to the ScreenMasker, a gaze-contingent tool that we developed [Paper II]. The ScreenMasker is based on the window-moving paradigm. It meets the requirements and recommendations for gaze-contingent display development. To ensure low latency, CUDA technology was used to process pixels on the GPU. We answered first Research Question that latency of the ScreenMasker was about 27 ms (on 250 Hz eye-tracker, 144 Hz screen). It was lower then limit (80 ms) and quite stable for various sizes of the window (the stencil), which allowed us to use the ScreenMasker in a variety of research problems (RQ1).

Finally, we carried out an eye-movement experiment to determine whether and how readers use extrafoveal information during source code comprehension [Paper III, Paper IV]. The answer for Research Question 2a is that extrafoveal information is not vital for task-solving (the subjects still solved the tasks). However, the availability of extrafoveal information allows experts to solve tasks more quickly. The programmers behavior changed when extrafoveal information about source code becomes unavailable. The changes in eye movements are stronger for experts than for novices. Furthermore, the restricted-view conditions make experts’ behavior similar to that of novices.

From the literature review, we knew that the extrafoveal information is used to program saccades to the next fixation point [21]. With a respect to Research Question 2b, we found that experts make longer fixations when the extrafoveal information is restricted. This result corresponds to similar findings from the different ‘professionals’ domains, visual search, and reading [21, 24, 68].

We showed that experts use extrafoveal information even when there are no long saccades to program (to the extrafoveal area). When experts want to make longer saccades, they use extrafoveal information more than novices do (RQ2b). To answer the Research Question 2c we showed that during source-code comprehension the extrafoveal information is processed even when a programmer does not plan saccades to the extrafoveal area. This could possibly be explained by the covert attention process taking place during the fixation phase [8, 195, 196].

We explained our results in a way that extrafoveal information helps experts encode the foveal object to a mental representation of the source code. Encoding of the foveal object goes with the covert attention process without macro-saccadic eye movement. This finding is new for the field of PoP and it corresponds with findings in a visual perception domain [9, 197].

Experts potentially can make more out-of-foveal fixations to prevent the lack of extrafoveal information during the restricted-view mode. The proportion of foveal fixations to out-of-foveal fixations was not affected by the restriction condition. During the restricted-view mode, the low-level vision process provides information for the high-level object encoding process. The encoding process is difficult in restricted condition. Experts are not familiar with this situation. The experts’ vision system is not trained to make more out-of-foveal fixations and the emotional reaction can be the one of the experts’ first reactions. This was a motivation for us to test this hypothesis.

We followed the suggestion that “more detailed and qualitative
approaches to analysis have to be employed and combined with the quantitative views that eye-tracking measures naturally support" [15]. When we analyzed the programmers’ self-reports after the source-code comprehension during the restricted-viewing mode, we found that experts provided more emotional feedback [Paper IV]. The answer for the Research Question 2d is that experts express themselves in a more emotional way and they mostly react negatively to the restricted-viewing mode. We concluded that experts direct their attention to extrafoveal objects during source-code comprehension. When extrafoveal objects are visually restricted and the information about their semantics is unavailable, there occurs a stronger emotional feedback due to disappointed expectations. We found no such reaction in novices.

Our results may improve training methods of programming. Better understanding of the experts’ visual patterns and strategies during source-code comprehension may benefit modeling of visual-attention process. In different domains, there is an assumption that attentional guidance is needed for novices to improve their expertise level [198]. There is a problem in designing such attentional guidance: "Directing the attention of novices to important parts of a video does not necessarily mean that relevant schemata will be activated" [198]. To solve this problem, additional information must be offered to novices. Our results showed that this additional information could be obtained from the extrafoveal area and could be involved in this type of attentional guidance for novices.

The educational implications are not the only possible. There is no formal theory of optimal eye-movement strategies during source-code comprehension. In a comparison with visual-search domain, "such a theory would provide insight into the design requirements [of software tools] for effective control of eye movements and attention" [29]. We believe that our findings could be involved in the design of an ‘ideal source code reader’ model to build a powerful framework for analyzing the behavior of eye movements and attention during source code comprehension, and for developing intelligent applications for IDEs.
Discussion

Finally, our results can be helpful for future studies of intelligent user interfaces for programmers. For example, additional information may be placed at the extrafoveal area to ensure better comprehension of a program. Today’s IDEs provide such information, like possible methods calling or attributes matching, followed by the cursor’s position mostly. In the future, this kind of new gaze-contingent interface should be examined in studies of computer-human interaction.

The generalisability of these results is subject to certain limitations. For instance, the scope of this study was to evaluate the role of extrafoveal information during source-code comprehension, but we did not evaluate the attribution of it to expertise level, rather than to a general perceptual. We found that in a chess domain, ‘a perceptual encoding advantage for experts attributable to chess experience, rather than to a general perceptual or memory superiority’ [199]. It would be interesting to study this issue in a future work. The next potential limitation of our study is that we used only Java programming language and textual IDE. These can bias the generalisability of our results. From the reading studies we found that the visual orientation of text (Chinese) and language properties (Hebrew) effect perception span [99, 100, 200, 201]. Further studies need to be carried out to validate the effect of programming languages’ type to the perceptual span.
Future research may focus on the cause-effect relationship between expertise level and the size of perceptual span. We still do not know if the perceptual span size influences the expertise level. If it does, this might mean that students with a wide perceptual span are more likely to become expert programmers. Neither do we know whether the expertise level affects the perceptual span. The background review allows us to conclude that it is possible to train individuals to improve their performance in obtaining parafoveal information [202]. However, it is known from the domain of chess playing that the visual span is affected by the level of expertise rather than by a general perceptual or memory superiority [199].

The answer to this debate question has the potential benefit of future work in the education field.

The ScreenMasker can potentially be used to run various research in PoP: for example, to evaluate the form, the size, and the asymmetry of the perceptual span. Cross-discipline studies with natural-language reading are another possible area of application. The constancy of our results could be evaluated by other methods of eye movements data analysis. Krejtz et al. suggested special Coefficient K in analysis of ambient/focal attention [203]. We did not found the studies of the scan path comparison during source code comprehension. There are several frameworks in the visual search domain, for instance Levenshtein distances using or T-pattern detection [204–206]. We did not find implementation of these frameworks in a source-code comprehension.

Future research can focus on situation modeling. Creating a prediction model of programmers' eye movements is a very ambitious task. It should build on the understanding of particular differences in extrafoveal information processing between experts and novices.
6 Questions for Future Research

Future research may focus on the cause-effect relationship between expertise level and the size of perceptual span. We still do not know if the perceptual span size influences the expertise level. If it does, this might mean that students with a wide perceptual span are more likely to become expert programmers. Neither do we know whether the expertise level affects the perceptual span. The background review allows us to conclude that it is possible to train individuals to improve their performance in obtaining parafoveal information [202]. However, it is known from the domain of chess playing that the visual span is affected by the level of expertise rather than by a general perceptual or memory superiority [199]. The answer to this debate question has the potential benefit of future work in the education field.

The ScreenMasker can potentially be used to run various research in PoP: for example, to evaluate the form, the size, and the asymmetry of the perceptual span. Cross-discipline studies with natural-language reading are another possible area of application.

The constancy of our results could be evaluated by other methods of eye movements data analysis. Krejtz et al. suggested special Coefficient $K$ in analysis of ambient/focal attention [203]. We did not find the studies of the scan path comparison during source code comprehension. There are several frameworks in the visual search domain, for instance Levenshtein distances using or T-pattern detection [204–206]. We did not find implementation of these frameworks in a source-code comprehension.

Future research can focus on situation modeling. Creating a prediction model of programmers’ eye movements is a very ambitious task. It should build on the understanding of particular differences in extrafoveal information processing between experts and novices.
These differences have traditionally been presumed but remained abstract and unspecified. Our research has contributed to understanding their specifics.
7 Conclusion

Our study produced three outcomes. The first one is the low-cost latency measurement system. This is a major contribution the gaze-contingent development. The second outcome is the ScreenMasker software. The ScreenMasker can be used to study visual perception and usability as well as to carry out CHI experiments. The ScreenMasker was published under an open-source license and can be used by anyone. The third outcome is theoretical. We answered questions about the role of extrafoveal information processing during source-code reading. We performed a ran experiments with the gaze-contingent window-moving environment (i.e., the ScreenMasker, a software tool that we built and evaluated ourselves).

The following conclusion can be reached: experts use extrafoveal information during source-code comprehension. Expert programmers use the extrafoveal area far more than novices. The restricted-view conditions influence the experts behavior more than that of novices. In the restricted mode, experts’ behavior becomes closer to that of novices. The results–a combination of quantitative analyses of programmers’ performance and eye movements [Paper III] and qualitative analyses of programmers self-reports [Paper IV]–allow us to indicate that expert programmers routinely use extrafoveal information in source-code comprehension.
Pavel A. Orlov:
Extrafoveal Vision During Source Code Comprehension: a
Gaze-contingent Tool, a Latency Evaluation Method, and Experiments
References


References


References


Pavel A. Orlov:

methods in the study of cognitive processes and activities. (Institut psychologii RAN, Moscow, 1994).


References


cons Play in Comprehension for Novice and Expert Program-
mers,” in 14th Workshop of the Psychology of Programming Inter-

acquisition: More evidence for the information-reduction hy-
pothesis.,” Journal of Experimental Psychology: Learning, Mem-

[68] K. Rayner, “Eye movements and attention in reading, scene
perception, and visual search.,” Quarterly journal of experimen-

[69] S. Schroeder, J. Hyönä, and S. P. Liversedge, “Develo-
pmental eye-tracking research in reading: Introduction to the special

[70] D. J. Schad and R. Engbert, “The zoom lens of attention: Sim-
ulating shuffled versus normal text reading using the SWIFT

[71] D. Drieghe, M. Brysbaert, and T. Desmet, “Parafoveal-on-
foveal effects on eye movements in text reading: Does an ex-
tra space make a difference?,” Vision Research 45, 1693–1706
(2005).

[72] S. Risse and R. Kliegl, “Dissociating preview validity and pre-
view difficulty in parafoveal processing of word n + 1 during
reading.,” Journal of experimental psychology. Human perception

[73] K. Rayner, S. J. White, G. Kambe, B. Miller, and S. P. Liv-
ersedge, “On the processing of meaning from parafoveal vi-
sion during eye fixations in reading,” The mind’s eye: Cognitive
References


Pavel A. Orlov:


References


References


References

symposium on Eye tracking research applications ETRA 00 6, 97–103 (2000).


References


[151] D. Ryu, Examining the role of central and peripheral vision in expert decision-making, PhD thesis (The University of Hong Kong (Pokfulam, Hong Kong), 2014).


[153] T. Busjahn, C. Schulte, B. Sharif, Simon, A. Begel, M. Hansen, R. Bednarik, P. Orlov, P. Ihantola, G. Shchekotova, and


References


[165] Q. Fan, The effects of beacons, comments, and tasks on program comprehension process in software maintenance, PhD thesis (University of Maryland at Baltimore County, Catonsville, {MD}, {USA}, 2010).


[169] P. Hejmady and N. H. Narayanan, “Multiple visualizations and debugging: how do we co-ordinate these?,” in Proceedings of the 2012 ACM annual conference extended abstracts on
Pavel A. Orlov:


References


References


Pavel A. Orlov:  


Understanding of programmers’ attention provides benefits for developing comprehension models and facilitating programming education activities. However, the visual attention studies in a psychology of programming explore central vision mostly and do not study the extrafoveal usage before. This work reports on a first-ever investigation of the role of extrafoveal information during programming. Here we provide a Gaze-contingent Tool—a Latency Evaluation Method, and experiments results.