Reading Code: It's Not the Message—It's the Medium

Mor Shamy
Dror G. Feitelson
The Hebrew University of Jerusalem
Jerusalem, Israel

ABSTRACT

Eye tracking studies have shown that code reading is non-linear, with many vertical jumps. As different lines of code may have very different functions (e.g. variable definition, flow control, or computation), it is important to accurately identify the lines being read. We design experiments that require a specific line of text to be scrutinized, to see how this depends on font size and spacing. The results indicate that, even after correcting for systematic bias, unnaturally large fonts and spacing may be required. The experiments also show that subjects repeatedly re-checked their task and that they are looking at the right line, leading to vertical jumps similar to those observed when reading code. This suggests that observed code reading patterns may also have more to do with the task and the structure of the code than with its content. If true, this implies that reading patterns do not necessarily reflect the process of building a mental model of the code.

CCS CONCEPTS

• **Software and its engineering**;

KEYWORDS

code reading, eye tracking

ACM Reference Format:
Mor Shamy and Dror G. Feitelson. 2021. Reading Code: It's Not the Message—It's the Medium . In *proocidings.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/xxxxxx.yyyyyy

1 INTRODUCTION

Eye trackers are increasingly being used by researchers in software engineering to gain insights into what developers—or experimental subjects—are actually doing and thinking [3, 21, 27, 31, 32]. A recurring result is that code, unlike natural language texts, is not read in a largely linear fashion. Rather, subjects seem to repeatedly scan the code, and to jump from one place to another.

As a consequence, it is important to correctly identify exactly what part of the code the subjects are looking at at each instant. Our work started out as a methodological study of how accurately this can be done. Specifically, we were interested in the trade-off between realism (trying to be as close as possible to normal working conditions) and accuracy (which can be improved by using unrealistically large fonts and spacing). We did this by designing experiments that require subjects to focus on a specific line of text. As we know exactly where they are supposed to be looking, we can analyze the distribution of gazes around this target.

Quite unexpectedly, these experiments led to reading patterns similar to those observed for code. In particular, the subjects often jumped from the target line to the question specifying what they were supposed to look for, and back again. This suggests that the patterns observed when reading code may also be associated with search and verification activities.

Such observations lead to the conclusion that developers do not really "read" code in the conventional sense of the word. Oftentimes they are not interested in the message conveyed by the code, namely what it is supposed to do. Rather, they are interested in the low-level details of how the code works—that is, in the artifact, in the medium. In fact, developers may actually employ an "only as needed" approach when reading code [18], and try to avoid actual understanding [17, 29].

This has profound implications for code comprehension research: We need to be punctilious regarding the distinction between comprehension in the sense of understanding *what* the code does (it's functionality), and comprehension in the sense of understanding *how* the code does it, that is, the code's structure and the interrelationships between different parts of the code. In academic research we often emphasize the first, e.g. in studies of code summarization [1, 27]. But in real-life work developers are typically more interested in the second, e.g. when performing debugging tasks or adding features to existing code.

2 BACKGROUND AND MOTIVATION

Eye tracking has been used in reading research for many years. Extremely detailed information about reading is now known, including the distribution of saccade lengths (the distance the eye moves from one fixation to the next), the perceptual span (how much can be seen in a single fixation, typically about 8 letters), and the prevalence of regressions (looking back at previously read text, around 10–15% of saccades) [5, 25].

Eye tracking was first used to study code reading by Crosby and Stelovsky in 1990 [10]. One of their findings was that reading code often involved multiple scans of the code. In addition, they found wide differences between subjects—for example, employing different numbers of scans. Both findings indicate a departure from reading in a largely linear order, as one may expect for, say, a newspaper story.

ETRA, 2021, Germany
© 2021 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/xxxxxx.yyyyyy

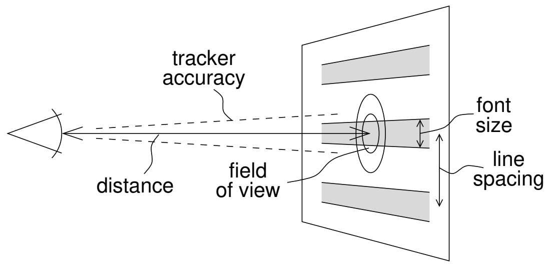


Figure 1: Factors affecting the ability to correctly identify the line being read.

Nevertheless, the prevailing assumption is that source code is similar to natural language, and the same mental processes underlie its reading [7, 28]—and also the same maladies, e.g. dyslexia [19]. At the same time, it has been recognized that code reading is different from text reading [4, 6, 10, 28]. For one thing, reading code is slower, perhaps due to its complexity. In addition, the attention given to code elements depends on their function: for example, more emphasis is given to identifiers relative to keywords [8].

Significant research has been focused on the issue of reading order. Regressions in reading text typically involve the previously fixated word, and rarely go to previous lines [5]. But practically all studies of code reading have noticed vertical scans of the code as identified by Crosby and Stelovsky. According to Uwano et al., such scans are a prerequisite for efficient finding of locations of interest in the code [34]. Busjahn et al. found that the reading order also depends on experience: reading by experts is less linear than reading by novices [7]. In addition to scans, a whole catalogue of reading patterns has been identified, including jumps back to previously read parts of the code, and jumps forward to preview what is coming [6, 15].

Given the prevalence of non-linear reading patterns, code reading studies must place a premium on correctly identifying what line of code is being read at each instant. The ability to distinguish between reading of adjacent lines depends on the interaction of four factors (Fig. 1):

(1) The angular discrimination of the eye tracker. Most trackers claim an accuracy of 0.5–1°.
(2) The distance to the screen, which translates the angular discrimination into an uncertainty regarding the location on the screen.
(3) The characteristics of the target line, specifically the font size and line spacing used. This dictates the differences in screen locations that must be discriminated.
(4) The vertical deviation between the center of the target line and where the eye is actually focused. The high-resolution center of the field of view, corresponding to the fovea in the eye's retina, is about 2°, which may span more than one line [25, 33]. It is therefore not really necessary to focus exactly on the target line to read it. In addition, deviations may arise from small movements and drift during a fixation [25].

Software engineering research papers using eye tracking rarely mentions calibration or systematic bias when describing their methodology [13]. But figures showing gaze paths superimposed on code stimuli often show short codes, relatively large inter-line spaces, and bias where more fixations fall between the lines than on the lines. Recommendations typically suggest that the font used be "large enough" to provide good discrimination, say around a size of 18pt [31]; a recent study on the parsing of URLs used a font size of 64(!) [24]. But even 18pt is larger than what developers usually use in routine work. As a result we reduce the realism of the study, and also limit the amount of code that can be displayed in one screen.

Our initial goal was to study this more systematically. We set out to design experiments in which we can measure the vertical distribution of gaze points, as measured by an eye tracker, around a known target. This would enable a precise calculation of the probability to misinterpret the line being looked at as a function of the font size and line spacing.

The actual results exceed these initial goals. They include a simple methodology for handling systematic biases in the eye tracking, and observations on the nature of reading code and how it differs from reading normal text.

3 LINE FOCUS EXPERIMENTS

Most research on reading text has focused on the horizontal dimension: the length of saccades, the perceptual span, skipping words or re-fixating on the same word, and the nature of regressions [25, 26]. Our focus is on the vertical dimension. One reason for this is the non-linear reading patterns that have been observed in previous code reading studies. Another is that in code different lines often have completely different functions: a variable declaration, an assignment, a branching instruction, a function header, etc. Identifying exactly which line is being read is therefore extremely important in order to understand what the reader is doing.

Results from previous studies commonly include fluctuations between adjacent lines (see, for example, [15, 34]). This may indicate that under normal viewing conditions the vertical resolution may not be adequate to correctly and consistently identify the line being read. This obviously depends on the font size and line spacing. The sizes developers use in everyday work may be too small to resolve with the desired fidelity.

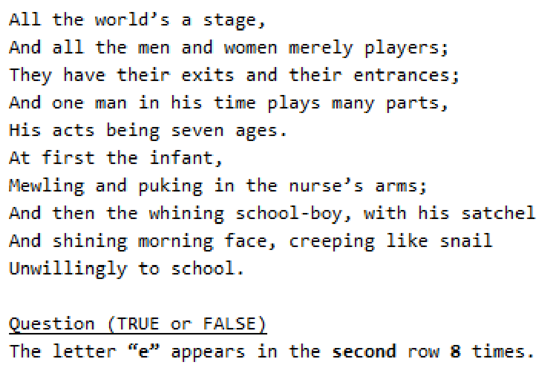
Our experiments were designed to study this systematically. The approach is to design tasks that require experimental subjects to focus on one specific line in the text. We then collect data on how they find this line, and on how focused they appear to be, given the experimental apparatus and the parameters of displaying the text. We do this in a real experimental setting, as opposed to using an artificial eye [13], as our goal is to characterize what can be achieved in practice and not to assess eye tracking apparatus.

3.1 Texts and Tasks

Eight English texts were used. They come from different categories, such as songs ("Yesterday"), plays ("All the world's a stage") and short descriptions of nature or leaders around the world. All texts were 10 or 11 lines long.

The texts were displayed in different ways:

- Font size – four texts differed in their font sizes: 10, 13, 16, and 20 points. All had a standard line spacing of 1.2.
- Spacing – four texts with a standard font size of 13 differed in their spacing: 1, 1.2, 1.5, and 2.



```
All the world's a stage,
And all the men and women merely players;
They have their exits and their entrances;
And one man in his time plays many parts,
His acts being seven ages.
At first the infant,
Mewling and puking in the nurse's arms;
And then the whining school-boy, with his satchel
And shining morning face, creeping like snail
Unwillingly to school.

Question (TRUE or FALSE)
The letter "e" appears in the second row 8 times.
```

Figure 2: Example of text and question used in the experiment.

- Text and background colors – either black lettering on a white background or the other way around. A dark background is considered better for eye tracking, but is unusual in everyday work settings.

The first two variations form the heart of our study. Given that we know where subjects are supposed to look, we can see what fraction of fixations is indeed assigned to the correct line, and how many are mis-assigned to adjacent lines.

Two tasks were used to make subjects focus on a specific line:

- Count the number of times a given letter appears in the line.
- Verify whether a set of three letters all appear in the line.

The question appeared after the text. An example is given in Fig. 2.

We decided not to use questions regarding the meaning of the texts, even if they are about information contained in a specific line. The reason was that subject might still read other lines, thinking they may be relevant. In addition, subject may answer from memory without even looking at the target line. This is not expected to happen for questions about letters that appear in the line.

The target line was specified in either of two ways:

- A line number. We chose either the second line or the seventh line of the text. The reason for choosing them was that the second line can be identified at a glance, but the seventh most probably requires counting. We avoided the first and last lines so that adjacent lines would exist in both directions.
- Using visual cues to identify the line. Two alternative cues were used: being indented, or being set in boldface. The goal was to see whether subjects would be able to home in on the target line rapidly (presumably using their peripheral vision) without scanning all the lines.

The experimental plan with the combinations that were used is described in Table 1. To reduce variability all experimental subjects received the same 8 texts with the same attributes. However, the order of the texts was randomized so as to avoid systematic fatigue and learning effects.

3.2 Experimental Setup

The experiments were conducted using a Gazepoint GP3 eye tracker operating at 60 Hz. This is one of the most low-cost remote eye trackers available today. The eye tracker was positioned just below a 24" screen with a resolution of 1080×1920. The texts were presented in the middle of the screen, in fullscreen mode. There were no other distracting elements on the screen.

Data exported from the tracker included (X,Y) locations, pupil diameter, and validity for each eye. A separate output contained fixations computed from the raw data by the Gazepoint analysis software.

The experiment was conducted at a desk situated far from the window, at a right angle, and behind an open-space partition, to reduce ambient light from outside and prevent it from interfering with the eye tracking. We checked the option of performing the experiments in the evening when it was dark outside, but saw no significant effect. Subjects were seated in a wheel-less chair to reduce movement, at a distance of ~70 cm from the screen.

Table 1: Combinations used in experimental plan.

			target		
Text	Size	Space	Line	Indication	Question
Shakespeare quote	10 pt	1.2	2	number	'e' × 8
Yesterday (Beatles)	13 pt	1.2	7	bold	'y', 'w', 'l'
Facts about giraffes	16 pt	1.2	7	indent	'f' × 4
History of Yemen	20 pt	1.2	7	number	'n', 'p', 'k'
Fidel Castro biography	13 pt	1	2	number	'a' × 4
Facts about stars	13 pt	1.2	6	indent	's', 'c', 'b'
Bear in Vancouver	13 pt	1.5	7	bold	'e' × 6
Facts about sharks	13 pt	2	7	number	'w', 'g', 'd'

3.3 Experiment Execution

The experimenter initially gave a general overview about the experiment and the eye tracker. Participants were told that the experiment is about testing the reliability of the eye tracker device.

After the subject confirmed he is sitting comfortably, he was asked to perform a 9 points calibration. We decided in advance that only results of subjects that will achieve a perfect score (9/9) in both eyes, in two tries, will be considered. Luckily, all 17 subjects met this criterion.

After the calibration, the eight texts appeared in a random order one after another. Subjects were instructed to read the text and then to answer the question. Answers were given verbally, so no physical touch with mouse or keyboard was needed. After an answer was received, the experimenter moved to the next text.

The subjects were 17 students from the Hebrew University, 11 females and 6 males. The average age was 24.5 (SD = 1.54). They were paid 20 Shekels for their participation. The whole procedure (including obtaining consent, explanations, and calibration) took about 15–20 minutes.

One subject was disqualified because she moved toward the screen to read better during the experiment, . Two of the results of another subject were disqualified for the same reason.

4 DATA ANALYSIS

All our analysis is based on the average of data from the left and right eyes [14]. As noted above, fixations were computed by the tracker software.

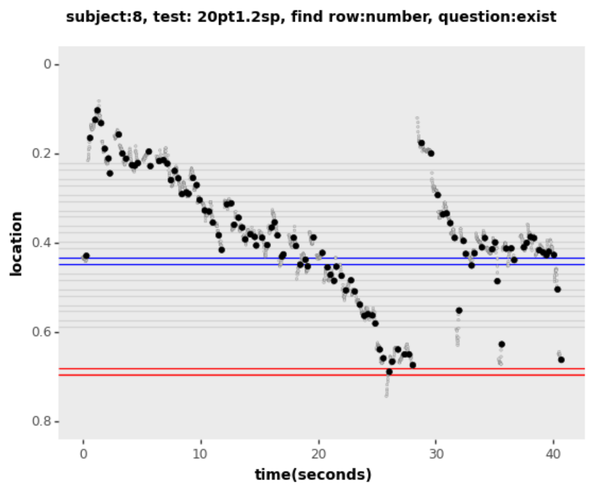


Figure 3: Example of results, showing the reading of the text, followed by reading the question (red line), counting the lines from the top, and focusing on the target line (the 7th line, in blue). Raw gaze data is in gray, and fixations in black.

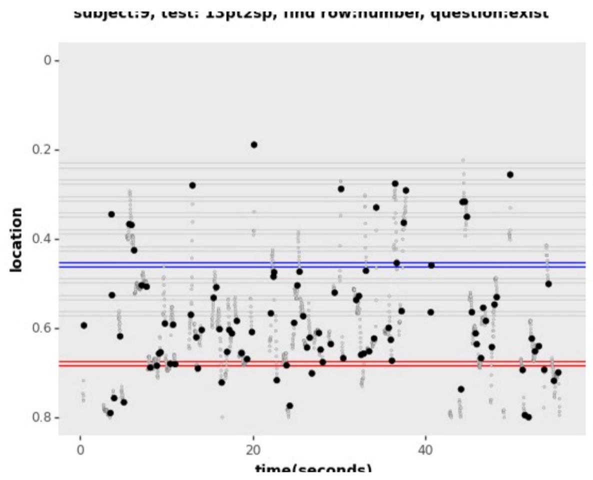


Figure 4: Example of noisy results that were excluded, as no focus on the target line could be identified.

We plotted the gaze and fixation data relative to the lines of text, using time for the horizontal axis and the screen Y coordinate as the vertical axis. In this display the screen X coordinate is not shown, which matches our focus on the vertical dimension. An example is shown in Fig. 3.

Initial observations revealed that some data was extremely noisy, and no period of focus on the target line could be identified (Fig. 4). both authors independently reviewed the graphs depicting each subject/text combination, and judged which were unusable. There was complete agreement on excluding one subject altogether, and also 6 specific texts of other subjects. While this exclusion rate is high, it is not unprecedented, and much higher exclusion rates of up to 60% have been reported in eye tracking studies [20].

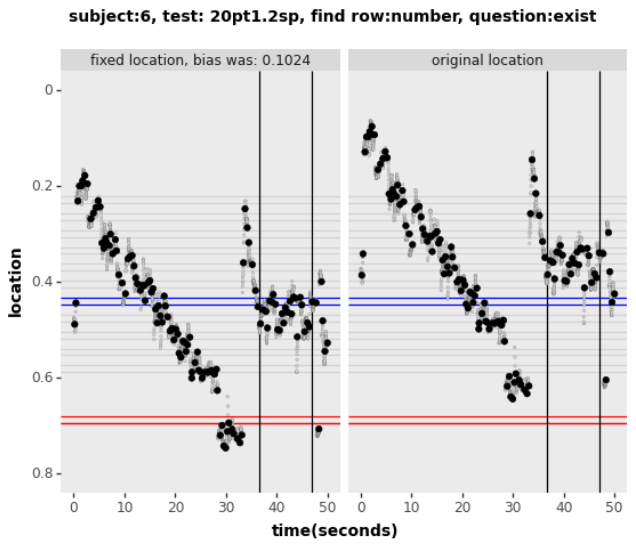


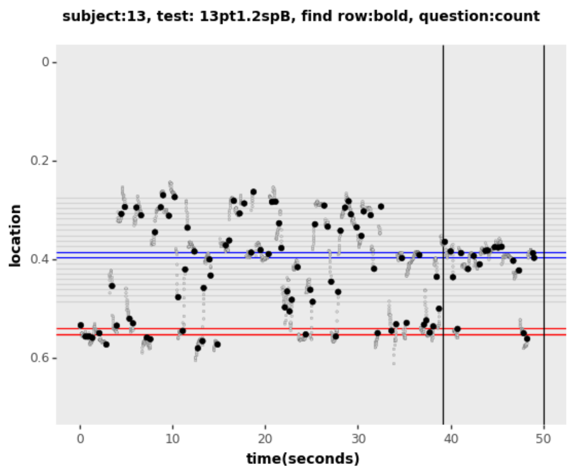
Figure 5: Example of correction of bias. The identified focus period is marked by vertical lines. Note that the correction is based on an average of all the texts viewed by the participant, so it may be sub-optimal for any specific text.

To identify the target line focus period in each case, we recruited an external judge. This judge had previously participated in the experiment, so she had first-hand experience with it. The first author and this judge independently viewed the graphs of all texts for all subjects, and noted when they thought the subject focused on the target line. In most cases they agreed to within 3 seconds. There were 11 cases of larger disagreements; in these cases the second author acted as arbiter.

Given the identification of periods in which it was believed that the participants were focused on the target line, it was obvious that many of the observations suffered from systematic bias. For most participants, the fixations appeared to be located above the target line, sometimes by a considerable margin. for a few, the fixations appeared below the line. We note that this is not unusual, and such bias has also been noticed in the context of developing the iTrace tool for eye tracking when using an IDE [30]. Palmer and Sharif have suggested a methodology for the automatic correction of such a bias, based on creating clusters of fixations, and trying to adjust them so that they fit likely areas of interest in the stimuli [22].

As we know exactly where the participants are supposed to be looking, we can use a simple and effective strategy. For each displayed text, we find the median height of all the fixations in the focus period, and compute the deviation of this height from the center of the target line—similar in principle to the task-embedded calibration suggested by Pi and Shi [23]. We then average these deviations for all the texts displayed to each participant. This produces a participant-specific correction that can be applied to all texts. An example is shown in Fig. 5.

After correcting the systematic bias, we can finally study the distribution of fixations around the target line.



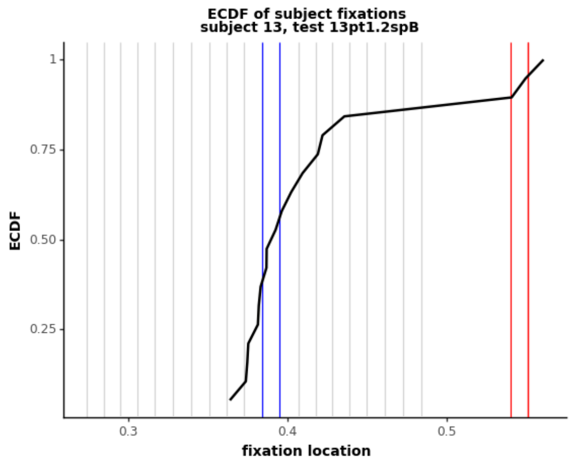


Figure 6: Example of the distribution of fixation heights during the focus period (seconds 39–50).

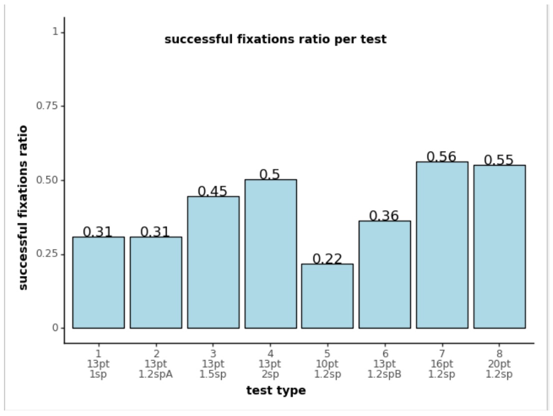


Figure 7: Effect of display parameters on how many of the fixations are identified as focusing on the target line.

5 RESULTS

Our main goal from the outset was to characterize the distribution of fixations around the target line. We want to know, during the period when the experimental subjects are focused on the target line, what fraction of the fixations indeed identifies the target line.

5.1 Deviations from the Focus

The vertical distribution of fixations during the focus period can be visualized using a CDF of the screen Y coordinate of these fixations, as shown in Fig. 6. The top panel shows all the fixations in the experiment and the identified focus period (delineated by the two vertical lines). The bottom panel shows the CDF. As one can see, most of the fixations are indeed distributed around the target line, although a few are actually on adjacent lines. But a few deviate to the question line. It seems that the subject took a couple of quick looks at the question during the focus period, to make sure he is counting the right thing.

This behavior is not unique to this example. Given the simplicity of the questions (count appearances of a letter, or verify that 3 letters appear at least once) we expected participants to focus exclusively on the target line to answer them. But surprisingly, In many cases they interrupted the execution of the task to re-check the question. In a few cases they also re-checked the line they were looking at. As a result the distribution of fixations could be bimodal or stretched, in a way that does not represent focusing on the target line.

To counter such behavior, in the following analysis we exclude fixations to locations below the bottommost line. This filters out instances of glancing down at the question. We return to this behavior in Sect. 5.3 below.

5.2 Distribution of Focus

Given the data on fixations on the target line, we can analyze the effect of font size and spacing on our ability to identify the line being read. We start by defining a fixation during the focus time as “indicative” if it can be assigned to the target line, namely if the fixation’s Y coordinate is closer to the target line than to any other line. Indicative fixations are those between the upper limit of the target line plus half of the gap between the target line and the line above it, and the bottom limit of the target line minus half of the gap between the target line and the line below it.

We use this definition to find the fraction of fixations that are indicative as a function of the text parameters. The results are shown in Fig. 7. Interestingly, the fraction of indicative fixations does not exceed 56% even for the most extreme settings checked—double space or a 20pt font. This implies that the fixations are not just distributed around the target line. Looking at the plots depicting the fixations in each case, we find that some of these deviations represent random glances at other places. Others are caused by systematic bias that was not corrected because it was different in different texts.

As these problems were more typical of some experimental subjects than others, we perform a second analysis in which we find the fraction of experimental subjects for whom the majority of fixations are indicative (i.e. at least 50%). The results are shown in Fig. 8. As we can see, when the font or the spacing are too small none of the experimental subjects meet this criterion. Using normal viewing conditions (text size of 13 pt with 1.2 spacing) about 10%

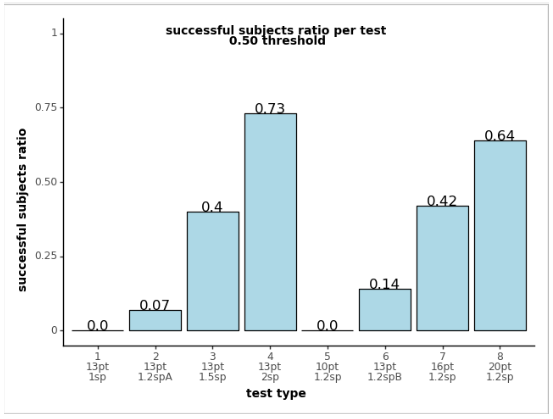


Figure 8: Effect of display parameters on how many participants are identified as focusing on the target line.

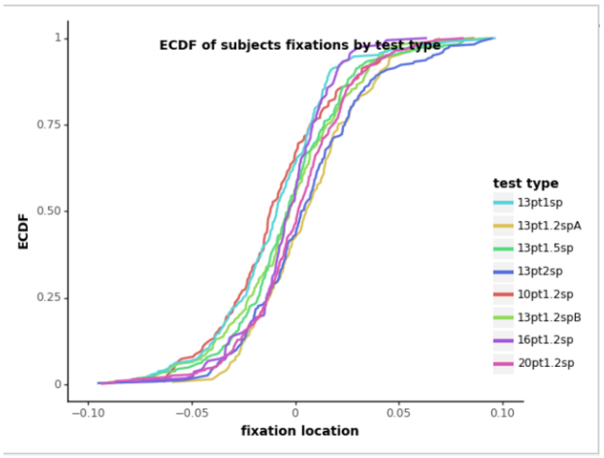


Figure 9: Distributions of deviations from the target line based on all participants.

of the subjects meet it. The majority of experimental subjects meet the criterion only with the largest font (20 pt) or the largest spacing (double space).

Another way to look at the results is to check whether the distribution of fixations depends on the text parameters. In other words, when the font and spacing are large, do participants "allow themselves" to be more lax, and let their eyes wonder farther away from the center of the target line?

To check this we create combined CDFs of the fixation heights across the participants. Thus we get 8 lines, one for each experimental setting, each of which includes all of the results for all participants. The results are shown in Fig. 9. The distributions are all rather similar to each other, leading to the conclusion that the distribution of gazes is not affected by the font and spacing settings.

5.3 Regressions to the Question

Looking back at text you have already read is called a "regression" in eye tracking reading studies. For normal text this may refer to lines above the current one. In our experiments, a common form of regression was looking again at the question. As the question appeared at the bottom of the screen, these are regressions to a line *below* the current line.

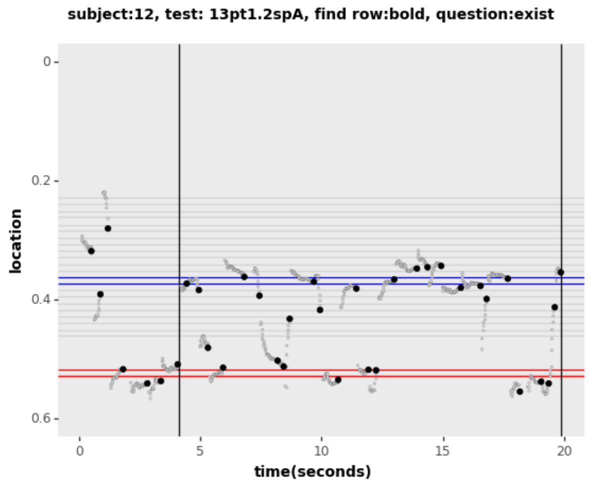


Figure 10: Example of multiple regressions to the question line.

Table 2: Regressions to the question line during focusing on the target line. (sem=standard error of the mean)

Question type	n	Regressions avg±sem
Count letter appearences	54	0.44±0.08
Verify 3 letters exist	58	1.50±0.25

Jumping back to the question was most probably done for verification, and may reflect the limits of working memory capacity [9]. We identified these regressions as sequences of consecutive fixations that are below the bottommost line. for example, at the extreme right end of Fig. 10 there are 3 consecutive fixations on the question, that count as one regression. In the experiments, such regressions occurred in exactly half of the cases. In some cases not one but several such regressions occurred (up to 5, Fig. 10).

Interestingly, a clear distinction was found between question types (Tab. 2). When subjects needed to check the presence of three different letters, there were more than 3 times as many regressions on average.

5.4 Finding the Target Line

In many cases it appeared that subjects found the target line effortlessly: They read the question, and immediately moved to the target line—presumably being able to identify it using their peripheral vision [16]. But when directed to the seventh line, the fixation pattern usually indicated that they moved to the first line and counted. However, the number of fixations on the way was typically less than 7: The field of view was large enough to count without fixating on every line.

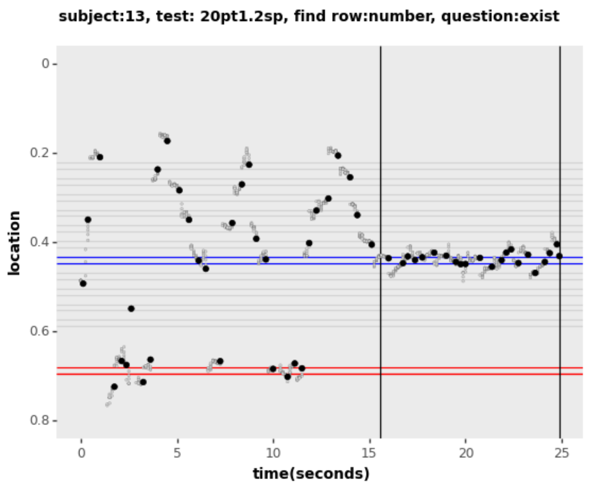


Figure 11: Example of counting lines multiple times, presumably to ascertain the target line.

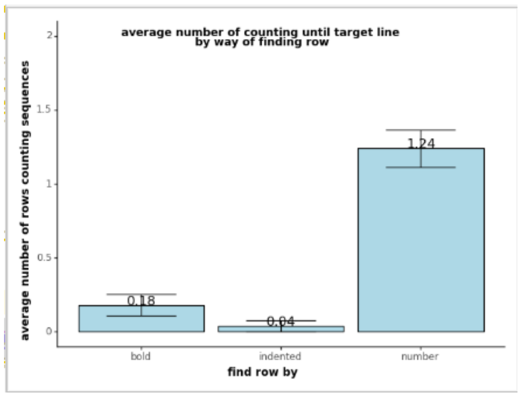


Figure 12: Subjects counted lines on average more than once when they needed to get to the seventh line, but seldom scanned previous lines when the target line was identified by a visual cue. Error bars indicate standard error of the mean.

Also, similarly to the regressions to the question, in many cases the subjects counted more than once (Fig. 11). Fig. 12 summarizes the average number of times that the lines above target line were scanned. Recall that when visual cues were used (target line set in boldface or indented) the target line was also the 6th or 7th line, so this is a fair comparison. Cases where the target line was the second line are excluded, because it was impossible to decide whether any counting took place due to variability. Counting to the seventh line was done at least once and often more, leading to an average of 1.24 times. Indented lines were apparently the easiest to identity, and there was only one (questionable) case of looking at previous lines.

6 DISCUSSION

Busjahn et al. identify two possible reading orders: "story order" (top-to-bottom, left-to-right) and "execution order" (tracing the

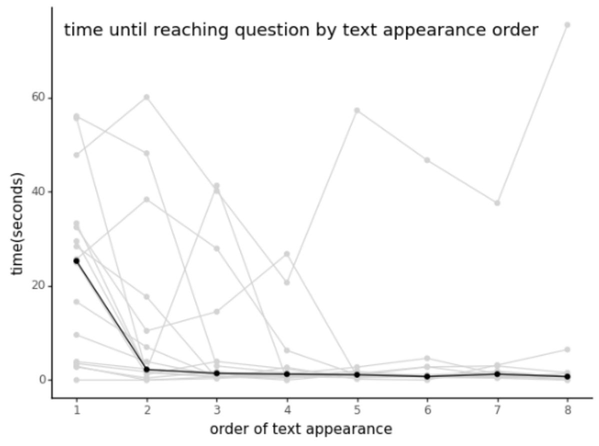


Figure 13: Time till subjects started to read the question, as a function of text serial number. The dark line is the median.

execution of the program, including function calls and loops) [7]. It is doubtful whether either of these actually represents what developers do when they read code.

When we read a story for fun, we most probably indeed read it in "story order"—from beginning to end, passing through all the words on the way. When we read a news story we might do the same, or we might skim some paragraphs that seem less interesting. But we rarely just read a piece of code. Usually we have some specific goal in mind, like fixing a bug or adding a feature.

In our experiments, the instructions given to the subjects were to initially read the texts, and then to read the question and figure out the answer. But they soon learned that they do not really need to read the whole text in order to answer the questions. So they started to skip the text and go straight to the question. As shown in Fig. 13, initially nearly all the subjects read or at least skimmed the text. From the second text, half already skipped it. after the 4th text, only one subject continued to read the text each time.

Once the question was read, the subjects needed to find the relevant line. The results show a difference between counting and using visual cues. It appears that visual cues can be noted using peripheral vision, and there is no need to fixate on each line. This suggests that the importance of indentation and color-coding keywords is that they provide beacons for navigation. Crosby and Stelovsky frown upon the practice of printing keywords in boldface, saying that "keywords are the least observed portions of a program's text" [10]. But this misses the point that clear visual identification of the keywords helps developers to easily focus on the code *between* these keywords. In contrast, Bauer et al. claim that indentation has no effect on gaze pattern [2]. However, they consider only aggregate metrics such as fixation duration, fixation rate, and saccade amplitude, and did not analyze gaze paths.

Counting is easily identified by a series of fixations going down from near the top of the text to the target line. Interestingly this was sometimes repeated and interspersed with jumps back to the question, presumably to ascertain the line number. We also observed jumps to the question when reading the target line, presumably to ascertain the letters that need to be looked for. This behavior is somewhat surprising given the extreme simplicity of our tasks: they require only to remember the line number and 1 to 3 letters. Nevertheless, our subjects needed to reassure themselves during the execution of the tasks that they were doing the right thing.

Returning to Busjahn et al.'s reading order speculations [7], we suggest an additional "top-down" order: Scan the code to get a grasp on its structure, and read service routines as needed to understand the methods that call them[1]. When reading classes or packages composed of multiple functions this may align with "execution order", as the order that routines are called in the code is the order in which they will be called during execution. However, one shouldn't expect people to read a loop body multiple times, unless they are explicitly required to trace the execution of the code in detail. Note also that top-down reading order may be affected by the ordering of methods in the code [11].

Our results also indicate that the search for a comprehensive reading order may be misguided. Humans seem to require constant validation during their work. This was manifested in the multiple repeated countings of the lines leading to the target line, and in the jumps to re-check the question when already focusing on the target line. It stands to reason that the jumps identified in code reading studies serve similar purposes. But different people need such verifications to different degrees and at different times, leading to different reading patterns.

7 THREATS TO VALIDITY

As we set out to characterize the accuracy of eye tracking in the vertical dimension, a major threat is whether our setup is optimal and representative. One specific concern is that we use a low-cost eye tracker. To ensure that the results are indeed representative the experiments need to be repeated with a range of trackers and experimental conditions.

Another concern is dealing with drift and the need for re-calibration. Our experiments were not very long, so we decided to settle for correction for systematic bias at the level of individual experimental subjects. But this issue too deserves additional research, including the option of using the knowledge of the target line to perform the adjustment per experiment.

Finally, using extremely simple experiments with completely synthetic tasks enabled us to make observations and theorize about the causes of reading patterns. But we did not prove that they indeed occur in code reading. This needs to be verified by carefully designed experiments with more realistic tasks, and probably also by soliciting intent from the subjects.

In addition, contemporary code development is done using IDEs, which affects the way code is accessed. This adds a whole new level of complexity to code reading studies [30], similar to the difference between online and physical reading of news [12]. We have not even started to scratch the surface of this issue.

8 CONCLUSIONS

Eye tracking studies are promoted as a window to the mind of the reader. In the context of reading code, they are thought to be useful for identifying mental models used for code comprehension—and specifically whether a top-down or bottom-up approach is being used [1, 3, 8]. Our work casts a doubt on whether this is so, for two reasons. First, we show that reading patterns similar to those seen when reading code actually reflect the need of people to reassure themselves in what they are doing, and are not necessarily related to building a specific mental model. Second, we show that accurate identification of the line being read may require unnaturally large fonts and spacing, thereby impairing experimental validity.

In practical terms, we show that conventional settings of font sizes and line spacing do not allow for adequate identification of which line is being read. A font size of at least 20pt and a wide spacing are apparently needed in order to guarantee that most fixations are assigned correctly, at least for our setup. However, additional research is needed to derive recommendations that are suitable for different eye tracking devices.

Eye tracking studies of developers working with code are burdened by the myriad factors that may affect the results. Our experiments are extremely simple and synthetic, thereby removing most of these factors. Such experiments can be used as an extension to the conventional calibration procedure, and can be interspersed with experimental tasks in order to correct shifts in calibration. Moreover, the results should apply also to the horizontal dimension, when we want to distinguish individual tokens in a program (which may be as fine as a single parenthesis).

Importantly, the accuracy of line identification can depend not only on the physical attributes of the stimuli (font size and line spacing) but also on the experimental subject. For some subjects, certain sizes can be adequate, while for others a larger size may be needed. Thus focus experiments like ours can also be used as a criterion for excluding certain subjects. By requiring the ratio of fixations that are assigned to the target line correctly to be above a certain threshold, the criterion becomes both quantifiable and directly relevant to reducing threats to validity of the actual study.

EXPERIMENTAL MATERIALS

The experimental materials (texts used int he experiment, raw results from the eye tracker, and analyses) are available at === ADD URL ===

ACKNOWLEDGMENTS

This research was supported by the ISRAEL SCIENCE FOUNDATION (grant no. 832/18).

REFERENCES

[1] Nahla J. Abid, Jonathan I. Maletic, and Bonita Sharif. 2019. Using Developer Eye Movements to Externalize the Mental Model Used in Code Summarization Tasks. In *Symp. Eye Tracking Res. & App.* Article 13. https://doi.org/10.1145/3314111.3319834

[2] Jennifer Bauer, Janet Siegmund, Norman Peitek, Johannes C. Hofmeister, and Sven Apel. 2019. Indentation: Simply a Matter of Style or Support for Program Comprehension?. In *Intl. Conf. Program Comprehension.* 154–164. https://doi.org/10.1109/ICPC.2019.00033

[3] Roman Bednarik and Markku Tukiainen. 2006. An Eye-Tracking Methodology for Characterizing Porgram Comprehension Processes. In *Symp. Eye Tracking Res. & App.* 125–132. https://doi.org/10.1145/1117309.1117356

[4] Tanja Blascheck and Bonita Sharif. 2019. Visually Analyzing Eye Movements on Natural Language Texts and Source Code Snippets. In *Symp. Eye Tracking Res. & App.* Article 14. https://doi.org/10.1145/3314111.3319917

[1]Abid et al. have also used the term "top-down" in reference to reading [1]. However, their classification was based on low-level patterns which we think are overly simplistic: "bottom-up" when a fixation was in the same basic block as the previous one, and "top-down" when it was in a new basic block—apparently including the next one.

[5] Robert W. Booth and Ulrich W. Weger. 2013. The Function of Regressions in Reading: Backward Eye Movements Allow Rereading. *Memory & Cognition* 41, 1 (Jan 2013), 82–97. https://doi.org/10.3758/s13421-012-0244-y

[6] Teresa Busjahn et al. 2014. Eye Tracking in Computing Education. In *Intl. Computing Education Research Conf.* 3–10. https://doi.org/10.1145/2632320.2632344

[7] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H. Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. 2015. Eye Movements in Code Reading: Relaxing the Linear Order. In *Intl. Conf. Program Comprehension.*

[8] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. 2011. Analysis of Code Reading to Gain More Insight in Program Comprehension. In *Koli Calling Intl. Conf. Computing Education Res.* 1–9. https://doi.org/10.1145/2094131.2094133

[9] Nelson Cowan. 2010. The Magical Mystery Four: How is Working Memory Capacity Limited, and Why? *Current Directions Psychological Sci.* 19, 1 (Feb 2010), 51–57. https://doi.org/10.1177/0963721409359277

[10] Martha E. Crosby and Jan Stelovsky. 1990. How Do We Read Algorithms? A Case Study. *Computer* 23, 1 (Jan 1990), 24–35. https://doi.org/10.1109/2.48797

[11] Yorai Geffen and Shahar Maoz. 2016. On Method Ordering. In *Intl. Conf. Program Comprehension.* https://doi.org/10.1109/ICPC.2016.7503711

[12] Kenneth Holmqvist, Jana Holsanova, Mari Barthelson, and David Lundqvist. 2003. Reading or Scanning? A Study of Newspaper and Net Paper Reading. In *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyönä, R. Radach, and H. Deubel (Eds.). North Holland, Chapter 30, 657–670. https://doi.org/10.1016/B978-044451020-4/50035-9

[13] Kenneth Holmqvist, Marcus Nyström, and Fiona Mulvey. 2012. Eye Tracker Data Quality: What it Is and How to Measure it. In *Symp. Eye Tracking Res. & App.* 45–52. https://doi.org/10.1145/2168556.2168563

[14] Ignace T. C. Hooge, Gijs A. Holleman, Nina C. Haukes, and Roy S. Hessels. 2019. Gaze Tracking Accuracy in Humans: One Eye is Sometimes Better than Two. *Behavior Research Methods* (2019). https://doi.org/10.3758/s13428-018-1135-3

[15] Ahmad Jbara and Dror G. Feitelson. 2017. How Programmers Read Regular Code: A Controlled Experiment Using Eye Tracking. *Empirical Softw. Eng.* 22, 3 (Jun 2017), 1440–1477. https://doi.org/10.1007/s10664-016-9477-x

[16] Matthew Kean and Anthony Lambert. 2003. Orienting of Visual Attention Based on Peripheral Information. In *The Mind's Eye: Cognitive and Applied Aspects of Eye Movement Research*, J. Hyönä, R. Radach, and H. Deubel (Eds.). North Holland, Chapter 2, 27–47. https://doi.org/10.1016/B978-044451020-4/50003-7

[17] Omer Levy and Dror G. Feitelson. 2019. Understanding Large-Scale Software – A Hierarchical View. In *Intl. Conf. Program Comprehension.* 283–293. https://doi.org/10.1109/ICPC.2019.00047

[18] David C. Littman, Jeannine Pinto, Stanley Letovsky, and Elliot Soloway. 1987. Mental Models and Software Maintenance. *J. Syst. & Softw.* 7, 4 (Dec 1987), 341–355. https://doi.org/10.1016/0164-1212(87)90033-1

[19] Ian McChesney and Raymond Bond. 2019. Eye Tracking Analysis of Computer Program Comprehension in Programmers with Dyslexia. *Empirical Softw. Eng.* 24, 3 (Jun 2019), 1109–1154. https://doi.org/10.1007/s10664-018-9649-y

[20] Marcus Nyström, Richard Andersson, Kenneth Holmqvist, and Joost van der Weijer. 2013. The Influence of Calibration Method and Eye Physiology on Eyetracking Data Quality. *Behavioral Res. Meth.* 45, 1 (Mar 2013), 272–288. https://doi.org/10.3758/s13428-012-0247-4

[21] Unaizah Obaidellah, Mohammed Al Haek, and Peter C.-H. Cheng. 2018. A Survey on the Usage of Eye-Tracking in Computer Programming. *ACM Comput. Surv.* 51, 1, Article 5 (Jan 2018). https://doi.org/10.1145/3145904

[22] Christopher Palmer and Bonita Sharif. 2016. Towards Automating Fixation Correction for Source Code. In *Symp. Eye Tracking Res. & App.* 65–68. https://doi.org/10.1145/2857491.2857544

[23] Jimin Pi and Bertram E. Shi. 2019. Task-Embedded Online Eye-Tracker Calibration for Improving Robustness to Head Motion. In *Symp. Eye Tracking Res. & App.* Article 8. https://doi.org/10.1145/3314111.3319845

[24] Niveta Ramkumar, Vijay Kothari, Caitlin Mills, Ross Koppel, Jim Blythe, Sean Smith, and Andrew L. Kun. 2020. Eyes on URLs: Relating Visual Behavior to Safety Decisions. In *Symp. Eye Tracking Res. & App.* Article 19. https://doi.org/10.1145/3379155.3391328

[25] Keith Rayner. 1998. Eye Movements in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin* 124, 3 (Nov 1998), 372–422. https://doi.org/10.1037/0033-2909.124.3.372

[26] Keith Rayner. 2009. Eye Movements and Attention in Reading, Scene Perception, and Visual Search. *Quarterly J. Experimental Psychology* 62, 8 (Aug 2009), 1457–1506. https://doi.org/10.1080/17470210902816461

[27] Paige Rodeghero, Cheng Liu, PAul W. McBurney, and Collin McMillan. 2015. An Eye-Tracking Study of Java Programmers and Application to Source Code Summarization. *IEEE Trans. Softw. Eng.* 41, 11 (Nov 2015), 1038–1054. https://doi.org/10.1109/TSE.2015.2442238

[28] Paige Rodeghero and Collin McMillan. 2015. An Empirical Study on the Patterns of Eye Movement During Summarization Tasks. In *Intl. Symp. Empirical Softw. Eng. & Measurement.* 11–20. https://doi.org/10.1109/ESEM.2015.7321188

[29] Tobias Roehm, Rebecca Tiarks, Rainer Koschke, and Walid Maalej. 2012. How Do Professional Developers Comprehend Software?. In *Intl. Conf. Softw. Eng.* 255–265. https://doi.org/10.1109/ICSE.2012.6227188

[30] Timothy R. Shaffer, Jenna L. Wise, Braden M. Walters, Sebastian C. Müller, Michael Falcone, and Bonita Sharif. 2015. iTrace: Enabling Eye Tracking on Software Artifacts within the IDE to Support Software Engineering Tasks. In *ESEC/FSE.* 954–957. https://doi.org/10.1145/2786805.2803188

[31] Zohreh Sharafi, Bonita Sharif, Yann-Gaël Guéhéneuc, Andrew Begel, Roman Bednarik, and Martha Crosby. 2020. A Practical Guide on Conducting Eye Tracking Studies in Software Engineering. *Empirical Softw. Eng.* 25, 5 (Sep 2020), 3128–3174. https://doi.org/10.1007/s10664-020-09829-4

[32] Zohreh Sharafi, Zéphyrin Soh, and Yann-Gaël Guéhéneuc. 2015. A Systematic Litareture Review on the Usage of Eye-Tracking in Software Engineering. *Inf. & Softw. Tech.* 67 (Nov 2015), 79–107. https://doi.org/10.1016/j.infsof.2015.06.008

[33] Hans Strasburger, Ingo Rentschler, and Martin Jüttner. 2011. Peripheral Vision and Pattern Recognition: A Review. *J. Vision* 11, 5, Article 13 (Dec 2011). https://doi.org/10.1167/11.5.13

[34] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken ichi Matsumoto. 2006. Analyzing Individual Performance of Source Code Review Using Reviewers' Eye Movement. In *Symp. Eye Tracking Res. & App.* 133–140. https://doi.org/10.1145/1117309.1117357