Color Perception in Programming

Sohini Mazumder
Clemson University
Clemson, SC, USA
sohinim@clemson.edu

Mary Walker Felder
Clemson University
Clemson, SC, USA
mwfelde@clemson.edu

Connie Ku
Clemson University
Clemson, SC, USA
ku@clemson.edu

ABSTRACT

Syntax highlighting is one of the most commonly used features in programming, as it is believed to make the code more comprehensible. While various studies have examined the effect of syntax highlighting on code reading, few have investigated its impact on error detection, which is one of the common challenges faced by programmers. This study aims to study the effect of syntax highlighting on error detection by having participants with coding experience to read 3 color-coded and 3 non-color-coded snippets. Performance speed, accuracy, and eye movements will be measured in the experiment for analysis.

CCS CONCEPTS

• **Software and its engineering** → *Syntax*; • **Theory of computation** → *Programming logic*; • ;

KEYWORDS

Eye Tracking, Visual Attention, Syntax Highlighting, Error detection

ACM Reference Format:
Sohini Mazumder, Mary Walker Felder, and Connie Ku. 2024. Color Perception in Programming. In . ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The ability to write and understand computer code is essential in today's digital world where it serves as the foundation for developing software, websites, applications, and various other technologies that drive innovation. It allows developers to design solutions that streamline processes and automate repetitive tasks across various domains. It is also necessary to be used in both the front end and back end applications for a majority of the industries.

However, one of the most common and mildly irritating errors that programmers make while writing code are syntax errors. In a study conducted on student programmers in 2015, the type of error with the highest frequency was noted to be syntax errors [Altadmri and Brown 2015]. Since they are easy to detect based on the programming language, most modern integrated development environments (IDEs) and text editors commonly used by programmers use color-coded syntax, or syntax highlighting.

Conference'17, July 2017, Washington, DC, USA
© 2024 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Syntax highlighting is a feature in text editors where different parts of the source code, such as keywords, variables, strings, and comments, are highlighted in distinct colors according to the programming language. This is done to enhance the visual contrast between different syntactical elements, making it easier for programmers to distinguish between them. For example, using rainbow parenthesis, where brackets are colored in matching pairs, provides visual cues that helps programmers detect unclosed brackets. The benefit of the color grouping effect was demonstrated by Michalski [2014] in a visual search study where they found that the speed of finding the target and the accuracy of performance was enhanced by it. This suggests that highlighting code in a certain color could be a valuable tool for improving productivity in coding environments.

In Gestalt psychology, a theory of mind that focuses on how people perceive and process information, the human brain naturally organizes similar visual stimuli (e.g., color) into groups [Wertheimer 1938] [Koffka 2013]. This innate perceptual grouping process, reduces users cognitive workload as they can simplify information processing by reading them in as a group, rather than a single word or item. When this is considered in the context of detecting coding errors, highlighted syntax prompts the brain to group those elements into distinct categories which promotes readability and error detection. This could result in fewer fixation counts because users are able to process the grouped elements more efficiently. Similarly, fixation duration may also be reduced as a result of perceptual grouping stimulated by highlighted syntax, as was hypothesized in one of the more recent studies by Liu et al. [2021].

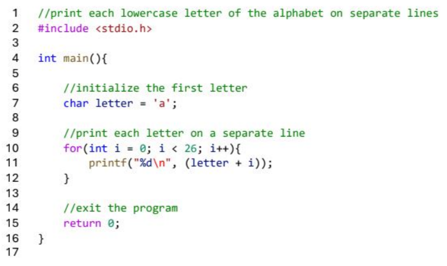
Though theoretically syntax highlighting may seem to improve error detection, the different aspects of potential improvement (such as accuracy and speed), and the extent of impact syntax highlighting has on error detection has yet to be thoroughly investigated. Newer results on the effect of syntax highlighting as used in modern Integrated Development Environments (IDEs) are inconclusive [Hannebauer et al. 2018].

This study aims to empirically investigate the effect of syntax highlighting on error detection by examining error detection speed and accuracy. Eye fixation counts and durations will be captured for analysis and discussion. Based on Gestalt principles and other existing literature such as Michalski [2014], Sarkar [2015] and Beelders and du Plessis [2016], we hypothesize that:

(1) **H1**: The speed of error detection will be faster in color-coded syntax compared to non-color-coded syntax.
(2) **H2**: The performance of error detection will be more accurate in color-coded syntax compared to non-color-coded syntax.

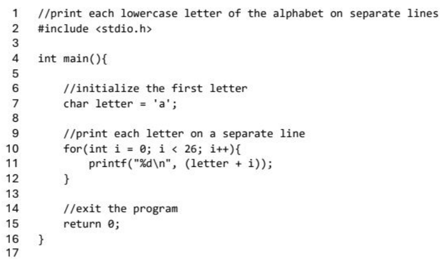
2 BACKGROUND

Eye tracking is often used to understand and quantify the way the participant interacts with visual information. While there are



```c
//print each lowercase letter of the alphabet on separate lines
#include <stdio.h>

int main(){

    //initialize the first letter
    char letter = 'a';

    //print each letter on a separate line
    for(int i = 0; i < 26; i++){
        printf("%d\n", (letter + i));
    }

    //exit the program
    return 0;
}
```

Figure 1: Sample Code with Syntax Highlighting



```c
//print each lowercase letter of the alphabet on separate lines
#include <stdio.h>

int main(){

    //initialize the first letter
    char letter = 'a';

    //print each letter on a separate line
    for(int i = 0; i < 26; i++){
        printf("%d\n", (letter + i));
    }

    //exit the program
    return 0;
}
```

Figure 2: Sample Code without Syntax Highlighting

other options to learn about the choices a participant makes, like asking them to explain their actions and thoughts, participants' perceptions do not always align with their underlying process [Sharafi et al. 2020]. Eye tracking allows for an understanding of how the participants act without the interference of their perception and quantifiable information on how that interaction occurs.

Eye tracking can show where individuals found areas of interest, where they fixated on the image or text, and how long each of these fixations lasted. This allows for a quantitative understanding of the participants' attention and effort. According to Sharafi et al. [2020], eye tracking in software engineering is used in a few typical studies like program comprehension, diagram comprehension, code reviews, traceability, and code summation. This understanding can be applied to many areas of software engineering to better understand how programmers interact with the code that they work with. In this particular case, this paper is focused on code reading and the possible impacts that syntax highlighting might have on how individuals read code.

There aren't many modern studies that examine how highlighted or color-coded syntax affects code reading and error detection. One of the earlier studies in this domain, conducted by Gilmore and Green [1988], concluded that syntax highlighting improves comprehension speed of the task. In the same year, Baecker [1988] conducted a study on source code readability and comprehension accuracy that found that the incorporation of color increased the percentage of correct answers by 11%. Another study showed that participants could find the output of a given piece of code faster if the syntax was highlighted than if it was not along with the context switches being lower Sarkar [2015],.

In contrast, Hakala et al. [2006] found that syntax highlighting did not have a significant impact on the speed of visual search on screen. Moreover, one of the more recent significant studies Beelders and du Plessis [2016] in this domain, used heatmaps ([Busjahn et al. 2011]) to show that while there is no significant difference between the number of fixations per line/AOI or in total fixation counts between the black-and-white and color-coded snippets. However, the color-coded syntax was noted to have minutely higher values in a readability context.

These studies provide a strong base to our hypotheses that the ease and accuracy of reading syntax when it's highlighted is likely going to be easier than when it is not.

3 METHODOLOGY

3.1 Participants

Ten participants were recruited from Clemson University (5 male, 4 female, 1 non-binary/third gender with average age of 24.2). All participants were undergraduate or graduate students with an average of 3.93 years of programming experience (SD = 0.94). Participants were recruited by word of mouth through student spaces and groups. There was no incentive or compensation provided for participating, and participation was entirely voluntary.

3.2 Apparatus

The apparatus used for this experiment is a GazePoint eye tracker with a sampling rate of 60 Hz and a reported 0.5-1 degree visual angle accuracy. The eye tracker was calibrated to each participant before they were shown the stimuli. The software PsychoPy v2023.1.3 was used for the experiment. The monitor that was used to display the stimuli to the participants was a DELL P2422H monitor with a 23.8" diagonal screen, 1920 x 1080p resolution, and a 60 Hz refresh rate. Qualtrics Survey was used to conduct the intro and exit surveys.

3.3 Stimulus

Six short programs were selected as the code stimuli. A syntax highlighted and black and white version of each program was created as shown in Figure 1 and 2. Each program was written for this experiment and included documentation as seen in typical programs. Each program was written in size 11 font and transferred to a blank page and converted to PDFs, so that no error notation would be visible. For the syntax highlighting, it was done in the style of VS Code's Light+ default highlighting which appeared to be the default light mode highlighting. Light mode highlighting was used as the black and white version of the stimulus is black text on a white background and light mode is the option for a white background.

3.4 Experimental Design

This study used a within-subject design, in which each participant was exposed to both color-coded and non-color-coded conditions. The order of conditions was randomized. Half of the participants (n

= 5) started with a color-coded snippet, whereas the other half (n = 5) started with a non-color-coded snippet. Participants alternated between color-coded and non-color-coded tests based on their starting condition. Each participant was provided with six code snippets to read and was asked to identify the line containing the error.

The time spent detecting errors and the accuracy of performance were measured. Although fixation counts and durations were intended to be recorded using an eye tracker to examine the effect on participants' eye movements, this was unsuccessful due to experimental errors (discussed in the limitations section). There was no time limit for the tasks.

3.5 Procedures

The main part of the experiment before the analysis is the actual data collection, which was done using the apparatus mentioned above, and the test methods as discussed earlier. During this process information about consenting to participate was shown and read out . The participants were provided a consent form with all the information about the experiment being conducted as well as a verbal explanation. It was explained that they have the right to withdraw at any time. Then, they were asked to fill out a demographic survey. They were asked to provide some basic information about themselves, or an alias for the purpose of the interaction during the experiment. All of this data was cleared after the experiment, and the data collected was assigned to non-personal identifiers for each participant. After the intro demographic survey had been filled out, an explanation of how the experiment will go was given to the participants. This included an explanation of how the eye tracker works and what will be expected of them throughout the experiment, such as staying still so that the eye tracker can read their gaze. The experiment methodology was clearly explained to the participants, and they were given enough time to understand and raise any questions they might have before the experiment begins.

Once all questions have been answered, the experiment started. Participants were shown to the computer and the eye tracker was calibrated for the participant. Participants were shown 6 code snippets, as described above. For Test 1, group was provided the colored code snippet for the first test code, while group B was provided the uncolored code snippet. For Test 2, group A was provided with the uncolored code snippet of the second test code, while group B was provided with the colored code snippet. The same parameters were noted. The experiment was repeated until Test 6, during which group A were reading the uncolored version of code snippet 6, while group B were reading the colored version of code snippet 6. At the end of each test, the participants were asked to enter the line number they thought the error was on. The duration of time taken to find the error for the snippet is noted, and the eye tracker was also be programmed to note the fixation durations or dwell time on each line of code. Each participant was given as much time as needed to complete each test.

Once all tests were over, the participants were asked to fill out a survey about their prior coding background and questions related to their perception of color-coded and non-color-coded snippets. An example question was, "Did you notice that some of the code snippets had highlighted syntax while some didn't?". They were also asked for any final questions or commentary on the experiment, such as possible improvements at the end of the experiment.

4 RESULTS

Data collected from the 10 participants were used for all analyses. Table 1 shows the descriptive statistics of response time and accuracy on error detection by conditions. Paired sample t-tests were conducted to examine the effect of color-coded and non-color-coded syntax on response time and accuracy of error detection. Results showed that there was no significant difference in response time between color-coded and non-color-coded condition, t (9) = 2.12, p > .05 (Figure 3). However, there was a significant difference in accuracy between the two conditions, t (9) = -2.45, p < .05 (Figure 4). Participants were better at detecting errors when working on color-coded syntax (M = 2, SD = 0.94) than non-color-coded syntax (M = 1.2, SD = 0.63).

Participants' answers to our survey questions were analyzed for qualitative analysis. It was found that 3 of the participants did not notice the difference between black-and-white syntax and color-coded syntax. Among the ones that noticed the difference, one of them responded that the black-and-white code was easier to find errors in, while the others agreed that the color-coded syntax was easier to read.

Color-coded		Non-color-coded	
Response time (seconds) M (SD)	Accuracy M (SD)	Response time (seconds) M (SD)	Accuracy M (SD)
63.84 (27.85)	2 (0.94)	104.75 (81.65)	1.2 (0.63)

Table 1: Descriptive statistics of response time and accuracy by conditions

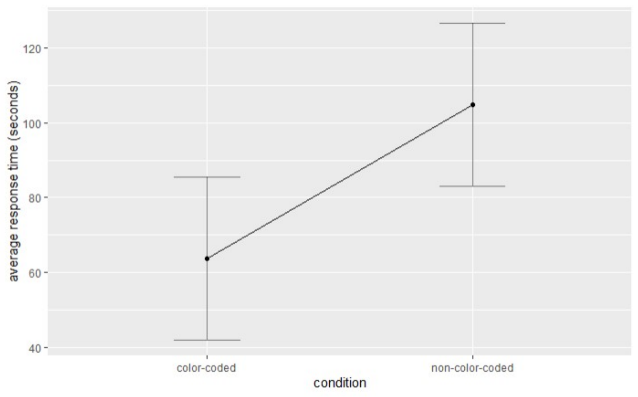


Figure 3: Average Response time in Color-Coded and Non-Color-Coded Conditions

5 DISCUSSION

This study investigated whether color-coded syntax influences response time and accuracy in error detection compared to non-color-coded syntax. Contrary to Hypothesis 1, the speed of error detection was not significantly faster in the color-coded syntax

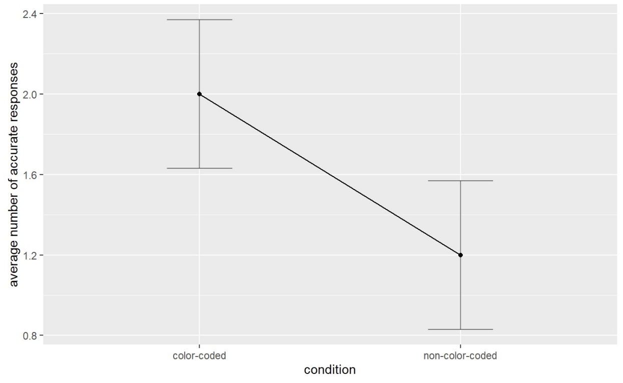


Figure 4: Average Number of Accurate Responses in Color-Coded and Non-Color-Coded Conditions

condition compared to non-color-coded syntax condition. In other words, color-coded syntax did not reduce the time they spent on detecting the errors. While grouping similar elements by color can enhance readability, the use of too many colors may have diminished this effect. Research suggests that there is limited capacity for working memory - the ability to hold and manipulate information [Miller 1956]. It is possible that the number of groups by color passed the limit of working memory capacity. In our experimental task, the color-coded syntax consists of 7 colors or groups (see Figure 1 for reference). This is very close to the maximum number of items working memory can hold according to Miller [1956]. As a result, the cognitive load imposed by processing multiple (7 or above) colors may have exceeded participants' working memory capacity, leading to no significant improvement in detection speed.

However, our findings were consistent with Hypothesis 2 - the accuracy of error detection was better in color-coded syntax compared to non-color-coded syntax. Participants were able to detect errors more accurately when the syntax was color-coded. This suggests that the visual grouping provided by color-coded syntax helped participants focus on relevant information and distinguish key elements, enhancing their ability to detect errors. While using a variety of colors may have hindered the potential benefit for error detection speed, it enhanced participants' ability to detect errors accurately. This suggests that error detection may involve several cognitive abilities such as working memory and attention control. It is possible that processing speed is associated with working memory, whereas error detection accuracy is related to attention control (the ability to focus on relevant information and disengage from unimportant information). Thus, color-coded syntax did not have the same effects on all aspects (i.e., speed and accuracy) of error detection.

6 LIMITATIONS & FUTURE WORK

There were several limitations within this study. One limitation was the participant pool as there were only ten participants involved in this study. The participants were all college students and in their twenties. This means that the results were created from a small set of data which brings up questions about the applicability of the findings. Future studies recreating this study with a larger and more diverse pool of participants would be helpful to determine how generalizable the results are and if the results hold true with further research.

Another limitation in this study was the lack of usable gaze data. Gaze data was recorded for each participant, but due to a lack of message events within the experiment file in PsychoPy, there was no way to determine what gaze data applied to which stimuli. Attempts were made to add these events to the experiment file, however there appears to be an error within PsychoPy regarding these events. There were no solutions that were able to be found and all examples of how to use such events caused the same error. If these events were able to be added, then new participants could have been recruited to gather new data for the study. The likely solution to this would be to shift future studies to another application or develop another way of separating out each stimulus' gaze data without relying on message events.

In the future, further studies to determine if gaze data does correlate to accuracy would be relevant as well as further studies to determine the generalizability of the findings of this study. Further research into whether these findings apply to all coding languages or code editors may be of interest.

7 CONCLUSION

This study highlights a crucial trade-off in the design of color-coded systems. While color coding can enhance accuracy, the number of colors used need to be carefully controlled to avoid overwhelming working memory capacity and reducing efficiency. Future research should examine the optimal amount of color used in programming to enhance both speed and accuracy in error detection.

REFERENCES

Altadmri, A. and Brown, N. C. (2015). 37 million compilations: Investigating novice programming mistakes in large-scale student data. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*, SIGCSE '15, page 522–527, New York, NY, USA. Association for Computing Machinery.

Baecker, R. (1988). Enhancing program readability and comprehensibility with tools for program visualization. In *Proceedings of the 10th International Conference on Software Engineering*, ICSE '88, page 356–366. IEEE Computer Society Press.

Beelders, T. and du Plessis, J.-P. L. (2016). Syntax highlighting as an influencing factor when reading and comprehending source code. *Journal of Eye Movement Research*, 9(1).

Busjahn, T., Schulte, C., and Busjahn, A. (2011). Analysis of code reading to gain more insight in program comprehension. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, Koli Calling '11, page 1–9, New York, NY, USA. Association for Computing Machinery.

Gilmore, D. J. and Green, T. R. G. (1988). Programming plans and programming expertise. *The Quarterly Journal of Experimental Psychology Section A*, 40(3):423–442.

Hakala, T., Nykyri, P., and Sajaniemi, J. (2006). An experiment on the effects of program code highlighting on visual search for local patterns.

Hannebauer, C., Hesenius, M., and Gruhn, V. (2018). Does syntax highlighting help programming novices? In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, page 704. Association for Computing Machinery.

Koffka, K. (2013). *Principles of Gestalt psychology*. routledge.

Liu, Y., Ma, W., Guo, X., Xuefen, L., Wu, C., and Zhu, T. (2021). Impacts of color coding on programming learning in multimedia learning: Moving toward a multimodal methodology. *Frontiers in Psychology*, 12:773328.

Michalski, R. (2014). The influence of color grouping on users' visual search behavior and preferences. *Displays*, 35(4):176–195.

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97.

Sarkar, A. (2015). The impact of syntax colouring on program comprehension.

Sharafi, Z., Sharif, B., Guéhéneuc, Y.-G., Begel, A., Bednarik, R., and Crosby, M. (2020). A practical guide on conducting eye tracking studies in software engineering. *Empirical software engineering : an international journal*, 25(5).

Wertheimer, M. (1938). Laws of organization in perceptual forms. *Psycologische Forschung, 4, 301-350.*