View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Journal of Eye Movement Research

Syntax highlighting as an influencing factor when reading and comprehending source code

T.R. Beelders
University of the Free State
Bloemfontein, South Africa

Jean-Pierre L. du Plessis
University of the Free State
Bloemfontein, South Africa

Syntax highlighting or syntax colouring, plays a vital role in programming development environments by colour-coding various code elements differently. The supposition is that this syntax highlighting assists programmers when reading and analysing code. However, academic text books are largely only available in black-and-white which could influence the comprehension of novice and beginner programmers. This study investigated whether student programmers experience more difficulty in reading and comprehending source code when it is presented without syntax highlighting. Number of fixations, fixation durations and regressions were all higher for black-and-white code than for colour code but not significantly so. Subjectively students indicated that the colour code snippets were easier to read and more aesthetically pleasing. Based on the analysis it could be concluded that students do not experience significantly more difficulty when reading code in black-and-white as printed in text books.

Keywords: Eye-tracking; syntax highlighting; code comprehension, reading behaviour

Introduction

Students in Information Technology (IT) are accustomed to writing and reading code in Integrated Development Environments (IDEs), such as Visual Studio® and Eclipse®. The IDEs provide code-specific visualisation in the form of syntax highlighting in the sense that code elements are colour-coded. For example, within a certain language, a class name will be a certain colour as will a primitive type declaration. Students (and programmers alike) become attuned to the colours used for syntax highlighting which could simplify the understanding of a code piece or increase the ability of a student to extrapolate meaning from an unfamiliar piece of code. However, based on experience from academic texts used at tertiary institutions, it is not uncommon for text books to be printed in black-and-white, possibly as a cost saving mechanism. This includes the code samples that are given in most programming text books. Some text books use bold face printing or italics to emphasise or indicate a certain type of code element but the fact remains that the code that students use to study and understand basic principles is not presented in a way that is familiar to them. As students become more adept at coding, the colour or lack thereof may cease to be an influencing factor when reading code but as beginner or novice programmers the colour syntax highlighting might assist them more than they realise.

Electronic devices are becoming more and more prevalent in today's society and are often being used as a reading device in place of printed material. Academic libraries have also increasingly been moving towards purchasing e-books (Olsen, Kleivset, & Langseth, 2013) and it therefore stands to reason that there will be a drive toward the use of e-book versions of textbooks in higher education. Apart from the advantages of saving paper and adjustable text size (Smith, Kukulska-Hulme, & Page, 2012), e-books are also potentially cheaper than hard-copy versions (Horcher & Tejay, 2012). In the field of IT, the prevalence of online code samples and free text books (cf. Nakov et al., 2013; Oracle, n.d.; MSDN, n.d.) are an even greater motivation towards the use of e-readers in tertiary education.

This study investigated whether IT students will be better served by making use of programming study material in colour as opposed to material presented in black-and-white only. To this end the aim of this study was to determine whether black-and-white material hinders the student in terms of how they read a code piece and whether they were able to understand a presented code piece if it was given in black-and-white as opposed to colour. In order to determine whether increased difficulty is experienced, eye movements while reading code snippets in black-and-white and colour were evaluated.

DOI 10.16910/jemr.9.1.1
This article is licensed under a
Creative Commons Attribution 4.0 International license.
ISSN 1995-8692

If students do indeed struggle more to read code in black-and-white, then there is ample motivation for making e-readers, specifically those capable of rendering text in colour, or tablets the norm for IT students as text books could then be made available in colour. Conversely, if students do not struggle more with reading in black-and-white, then low-cost e-readers or tablets are a viable solution for exposing students to a wider variety and larger code repository for educational purposes.

This paper will proceed by first discussing some background on eye movements, specifically as they relate to reading. This will be followed by a discussion on the methodology used for the study. Finally the results of the data analysis and the conclusions drawn from the results will be presented.

Background

Eye movements

The human eye exhibits a number of movements. Of interest to this study are the moments when the eye is relatively stable. These periods are called fixations and typically last between 200-300 milliseconds (Rayner, 1998). Fixations are needed in order for a human to be able to "see" an object of interest. Rapid ballistic movements, called saccades, are used to position gaze over these objects (Gregory, 1966) which need to be seen.

Eye movements while reading

The basic eye movements discussed in the previous section are also relevant to behaviour during reading. However, the nature of reading differs from other visual tasks such as scene perception. Reading requires line by line comprehension, and while the physiology of the eye requires positioning the fovea over an object of interest, a visual scene allows for processing of more information around the fixation point than during reading (Rayner, 2009).

Fixations and saccades can be used to determine the difficulty experienced during reading. The duration of a fixation, or the length of time the eye remains stable on a word, can give an indication of the cognitive processing that is occurring at that moment (Menz & Groner, 1984). This can be indicative of the amount of resources required to assimilate and understand the word.

Fixations during reading are generally shorter than during scene perception and also differ based on whether the reading is silent or oral (Rayner, 2009). Mean fixation lengths during silent reading range between 225 and 250 milliseconds, but can vary substantially from 50 to 600 milliseconds (Rayner, 2009). Saccade length is also shorter during reading than during scene perception and visual search and is generally measured according to the number of letters that are traversed during the saccade (Rayner, 2009). Regressive saccades are saccades in the direction opposite to that of the reading text, for example, in a left to right text, regressive saccades are right to left. These indicate the reader is struggling with comprehension of the text or finding the text difficult to read (Rayner, 2009). Regressive saccades should not be confused with line sweeps, which are the natural movement of the eyes from the end of one line of text to the start of the next.

Eye movements while reading code

Reading source code is an exercise that differs vastly from reading normal text. As such, Bednarik and Tukianinen (2006) cautioned that a methodological framework is required in order to study programming behavioural facets through analysis of eye movement. They made use of a visualisation tool in order to investigate comprehension through eye gaze. Results indicate that eye movement is a valid means of assessing cognitive processing levels involved in comprehension, debugging and visualisation (Bednarik & Tukianinen, 2006). Furthermore, it was shown that eye movement analysis enhances single methodology analysis by uncovering additional important information (Bednarik & Tukianinen, 2006).

Therefore, the same reading movements and metrics can be used to evaluate behaviour when reading code, but it has been shown that mean fixation times when reading source code (351 ms) are longer than when reading natural text which could be indicative of increased attentional demands (Busjahn, Shulte, & Busjahn, 2011). This study therefore used the standard reading metrics to evaluate reading behaviour for black-and-white and colour code snippets. Notably, the number of regressive saccades present when reading source code (37%) is substantially more than when reading natural text (Busjahn, Shulte, & Busjahn, 2011) which could be due to the nature of some coding elements, such as loops which would naturally cause the programmer to return to previously read code element, i.e. regress. Fixation durations and number of fixations confirm that expert and novice programmers pay attention to different elements of code but that there is no difference between these groups in terms of reading strategy (Crosby & Stelovsky, 1990).

Camel case, where each word in the compound word starts with a capital letter (e.g. authorName and VariableNameExample) and underscores (e.g. authorName

This article is licensed under a
Creative Commons Attribution 4.0 International license.

and Variable_Name_Example) are two common methods used as naming conventions and standards within the domain of programming. However, camel casing appears to require more visual effort to process based on the number of fixations (Binkley et al., 2013). Various experimental designs were used in order to evaluate the difference between camel casing and underscoring identifiers but results were varied. It was however found that reading code differs from reading natural language and the suggestion is that the structure and syntax of source code allows for rapid comprehension thereof, irrespective of the styling used for identifiers (Binkley et al., 2013). This study controlled for the effect of styling by using consistent naming conventions between code snippets in order to ensure that only the colour of the code snippets could influence reading behaviour.

A recent study investigated the difference in fixation count and duration between C++ and Python code, some of which contained errors. The aim was to determine whether programming language affected the comprehension of source code. No difference was found in terms of accuracy or time, but there was a difference detected in terms of number of fixations on lines of code containing errors (Turner, Falcone, Sharif, & Lazar, 2014). The code snippets presented in the current study did not contain any errors as the focus was on the colour of the code and including errors would only introduce another potential influencing factor.

None of the prior studies focussed on the difference between code presented in colour and code presented in black-and-white. However, recent studies did show that syntax highlighting significantly decreases the amount of time required to complete a task for mental execution of a code snippet (Sarkar, 2015), writing a piece of code and debugging a piece of code (Dimitri, 2015). Furthermore, the effect of syntax highlighting on code comprehension has been investigated, wherein fixation counts and duration were used to estimate visual effort. It was found that syntax highlighting has no significant difference on these metrics. The metrics were only calculated for the entire code snippet given to participants (Sarkar, 2015). This study used a finer granularity for these metrics by distinguishing between a code snippet, code lines and words.

METHODOLOGY

Experimental methodology

The study consisted of code snippets presented with and without syntax highlighting. Eye movements that occurred during the reading of the source code were captured using a Tobii® TX-300 eye tracker, which has a sampling rate of 300Hz. Tobii Studio's® Velocity-Threshold Identification (I-VT) fixation classification algorithm was used to identify fixations. This is a velocity-based fixation algorithm that was set using a velocity threshold of 30 degrees/seconds and a minimum fixation duration of 60 ms for the purpose of this study. Code snippets were presented on a 24" monitor with a resolution of 1920×1080 pixels. Participants were seated approximately 60 cm from the stimulus.

Two console-based code snippets were selected based on students' familiarity with concepts, difficulty of the concepts, and number of branches in the code (cyclomatic complexity). Cyclomatic complexity (McCabe, 1976) was developed to analyse code complexity in terms of the number of decision branches present within a piece of software. This measure has been used in previous studies to predict defective software components (Zhang, Zhang, & Gu, 2007) and uncover weaknesses in code (McCabe, McCabe, & Fiondella, 2012). The cyclomatic complexity value was used to ensure that the two code snippets contained comparable number of branches and to ensure that participants would have a reasonable chance of comprehending the code. Similar to previous studies (Busjahn et al., 2014) participants were told there were no errors in the code.

A font size of 20 pt was used for all source code to ensure that code would be readable at a distance of up to 70 cm from the stimulus. Each participant had to read both code snippets in either black-and-white or colour. There was no time limit placed on reading a code snippet. Participants were requested to complete a task while viewing each code snippet. The task required the participant to determine the output of each code sample. This was done to force the participant to study the source code, rather than just give it a cursory reading. Any input values that would normally be supplied by a user were given to the participant beforehand on an instruction sheet. This was also verbally communicated to the participant before the task commenced. Participants were asked to write the output of the code snippet on the instruction sheet provided for each task. Since the task could cause the participant to look up and down frequently, lines of code were spaced to account for inaccuracies present in the gaze mapping due to the constant shifting of a participant's head position. Due to the additional spacing, one code sample required minimal scrolling; the other was completely visible on a single page. To facilitate easier scrolling, the code was presented within a web browser that allows Tobii Studio® to compensate for scrolling within the stimulus, thereby ensuring that the fixations are correctly mapped to the lines of code, regardless of scrolling behaviour. The order in which the

This article is licensed under a
Creative Commons Attribution 4.0 International license.
ISSN 1995-8692

code snippets was shown to participants was also alternated to ensure a balanced study.

Further questions were posed after the participant had completed each task to determine the perceived level of difficulty of the code, the readability of the code, and the personal preference of the code presentation. Responses were captured using a five point Likert scale.

Gaze data was captured using Tobii Studio® and exported to a database. Regressions were counted by hand using a custom visualisation tool developed for this purpose. In the context of this study, regressions were considered to be any eye movement backwards or upwards to a new code element. Heatmaps were constructed using areas of interest (AOIs) and the fixation data exported from Tobii Studio®. All statistics were conducted using the statistical software package Statistica 12® while G*Power 3 (Faul, Erdfelder, Lang & Buchner, 2007) was used to conduct the power analysis. All data was first tested to determine whether it was normally distributed. T-tests were used for further analysis if the data were normally distributed. For non-normal data the non-parametric equivalent, namely the Mann-Whitney test, was used as an alternative, since the small sample size precludes the use of parametric tests where the normality assumption is not met.

Participants

A convenience sample was used for the study since participants were drawn from the student body of the university where the study was conducted. The only prerequisite for participation was that participants had to be currently enrolled for an IT degree. The pre-test questionnaire evaluated programming experience based on the number of years the participant had been programming and their subjective evaluation of their expertise. Results indicated that all participants were approximately on the same level of programming experience. Consequently this was not used as criterion to distinguish between the participants. A total of 34 participants, with an average age of 21.7 (SD = 3.4), participated in the study. All participants were familiar with the C#.Net programming language, which the stimuli were presented in. Each group, namely black-and-white code and colour code, had 17 participants. Participants were randomly assigned to either the black-and-white or colour group.

Metrics

It has been established that fixations can be used to determine difficulty experienced during reading and that this metric is equally applicable when reading code. Therefore, this study will use fixation durations and fixation counts to evaluate difficulty experienced. Furthermore, regressive saccades will also be used as a measure of difficulty experienced while reading code. The code snippets were analysed separately.

RESULTS

Fixation Count

The fixation count metric, indicates the total number of fixations, per participant, that occurred during the test for the entire stimulus. As previously mentioned, the number of fixations can be indicative of the difficulty the participant is experiencing with the piece of text or code. The following hypothesis was formulated:

H_0: There is no difference between the number of fixations when reading code in black-and-white versus colour.

For the first code snippet, the Mann-Whitney test showed no significant difference (U = 99.0, p = 0.12) between the black-and-white code ($\bar{x}$ = 1456.1) and colour code ($\bar{x}$ = 974.5) in terms of the number of fixations. Similarly, for the second code snippet, there was no significant difference (U = 136.0, p 0.78) between the number of fixation for the black-and-white code ($\bar{x}$ = 1409.2.0) and the colour code ($\bar{x}$ = 1343.8). Therefore, the null hypothesis cannot be rejected and it can be concluded that there is no significant difference between the number of fixations when reading black-and-white code and when reading code in colour. It was however noted that the mean number of fixations was higher for the black-and-white than for the colour versions of the code.

Fixation Count per line

Since each line of code can be viewed as a single concept or a single instruction, the number of fixations were calculated for each line of code for both code snippets. For this metric, only the fixations that could be identified as being on a specific line were included, as opposed to the previous metric which included all fixations on the stimulus. Figure 1 illustrates the mean number of fixations per line for the colour code snippet (blue bar) and the black-and-white snippet (orange bar) for the first code snippet. The same visualisation is given for the second code snippet in Figure 2.

It appears, on average, that there were more fixations per line on the black-and-white code snippets than on the colour snippets. As could be expected, lines with loops, calculations or more complex logic had a larger mean number of fixations. The string formatting lines (Figure 3) present in

DOI 10.16910/jemr.9.1.1
This article is licensed under a
Creative Commons Attribution 4.0 International license.
ISSN 1995-8692

the code were conceptually familiar to participants but may have been presented with more information than what the participants were accustomed to. This could account for the larger number of fixations on those lines, in particular the second string formatting line.

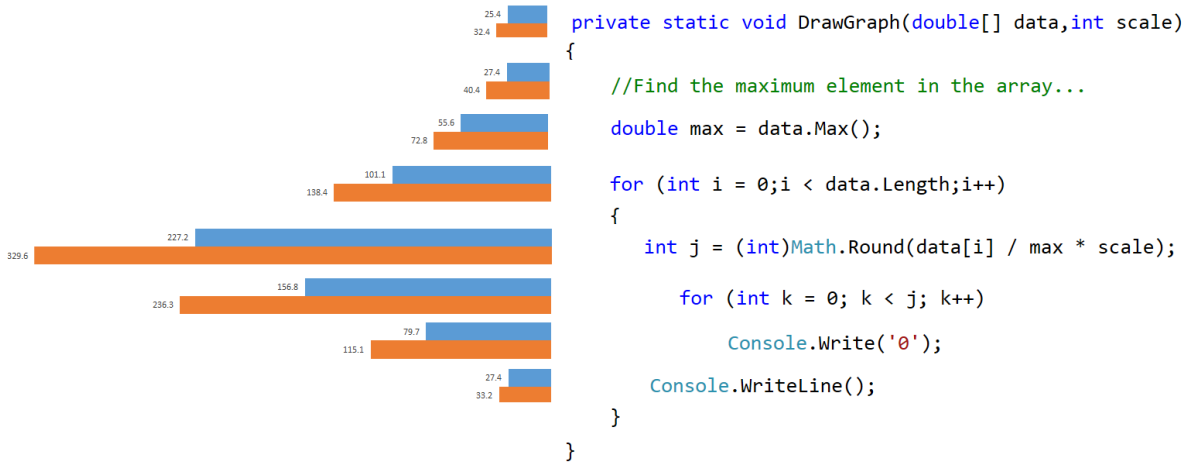


Figure 1. Fixations per line of code for first code snippet for both the colour snippet (blue bar) and the black-and-white snippet (orange bar)

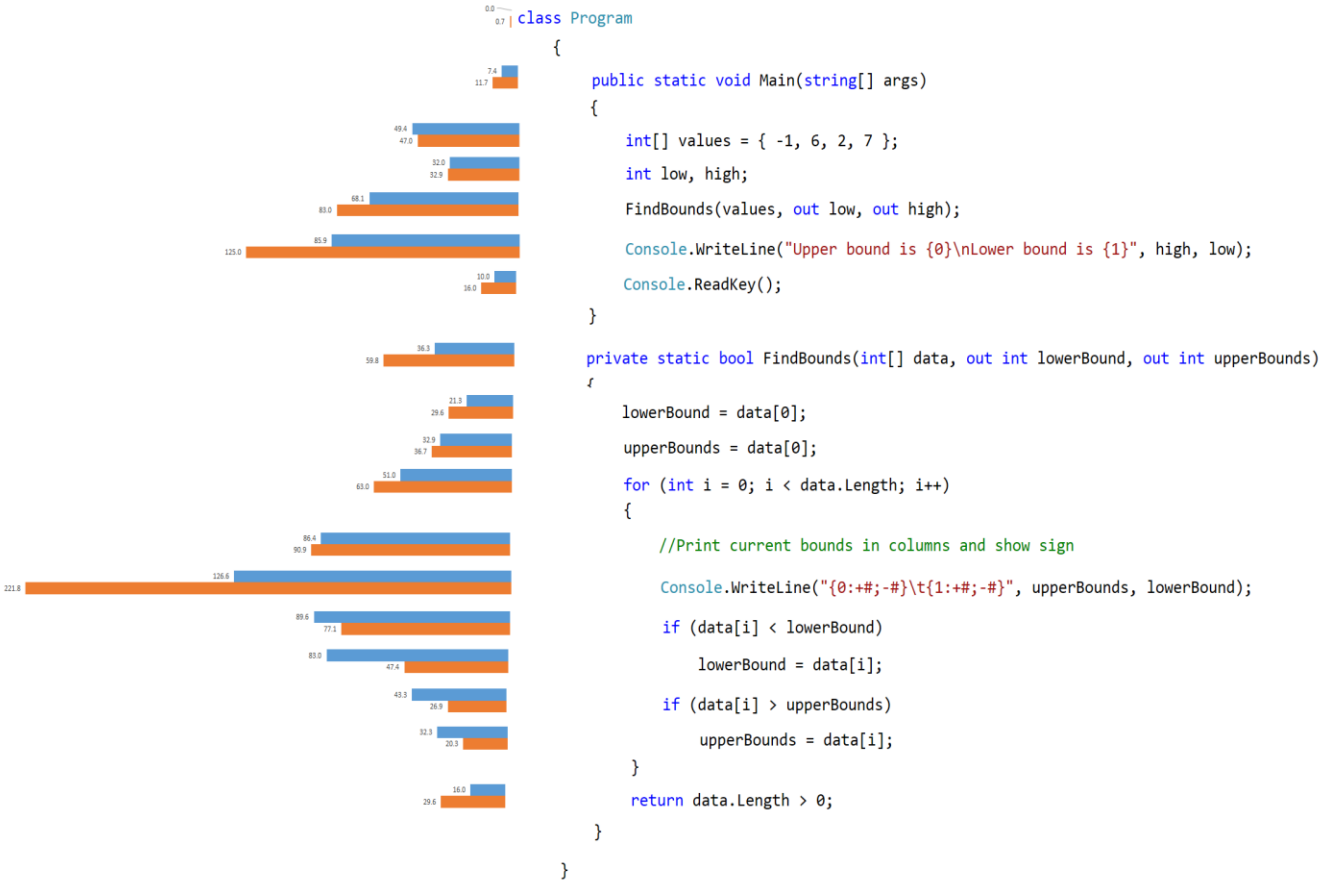


Figure 2. Fixations per line of code for second code snippet for both the colour snippet (blue bar) and the black-and-white snippet (orange bar)

This article is licensed under a
Creative Commons Attribution 4.0 International license.

```
Console.WriteLine("Upper bound is {0}\nLower bound is {1}", high, low);

Console.WriteLine("{0:+#;-#}\t{1:+#;-#}", upperBounds, lowerBound);
```

Figure 3: The two string formatting lines as used in the second code snippet

The following hypothesis was formulated:

H_0: There is no difference between the number of fixations per line of code when reading code in black-and-white versus colour.

A Mann-Whitney test was conducted since the data was not normally distributed. The null hypothesis could not be rejected for either the first code snippet (U = 23.0, p = 0.37) or the second code snippet (U= 158.0, p = 0.91).

Fixation Count per word

Although fixations per line were evaluated, it was surmised that in some instances there may be isolated method invocations or coding elements (such as object instantiation) on a line that garnered more attention from the participants. AOIs were therefore drawn around each word (apart from comments which were viewed as a single concept and thus as a single AOI). The number of fixations for each AIO was then calculated. The following hypothesis was stated:

H_0: There is no difference between the mean number of fixations per AOI when reading code in black-and-white versus colour.

The mean number of fixations per AOI for each stimulus is tabulated below:

Table 1
Mean number of fixations per AOI

	First code snippet	Second code snippet
Black and white	$\bar{x}$ = 37.8	$\bar{x}$ = 18.8
Colour	$\bar{x}$ = 27.7	$\bar{x}$ = 18.2

Once again, the mean values are higher for the black-and-white code than for the colour code, more so for the first snippet than the second, which could be indicative of difficulty experienced. After conducting a Mann-Whitney test, the null hypothesis could not be rejected for the first code snippet (U = 104, p = 0.17) or the second code snippet (U = 141.5, p = 0.93), indicating that there is no significant difference between the mean number of fixations per AOI when reading in black-and-white or colour.

Fixation duration

Mean fixation durations were calculated for each participant for the entire stimulus. Furthermore, since participants could look away from the stimuli, the mean duration of fixations spent within the bounds of an AOI were also calculated for all participants and stimuli to analyse the time spent on coding elements. Table 2 summarises the mean fixation durations.

Table 2
Mean fixation durations per code snippet and mean fixation duration spent within an AOI

	First code snippet		Second code snippet	
	Whole stimulus	Duration within AOI	Whole stimulus	Duration within AOI
Black and white	$\bar{x}$ = 174.5	$\bar{x}$ = 183.5	$\bar{x}$ = 187.8	$\bar{x}$ = 189.5
Colour	$\bar{x}$ = 171.9	$\bar{x}$ = 168.4	$\bar{x}$ = 181.8	$\bar{x}$ = 177.5

Fixation durations were, on average, lower for the colour code than for the black-and-white code although not largely so. For example, over the entire stimulus, the mean fixation duration for the colour code was 171.9 and 174.5 milliseconds for the black-and-white, while for the second snippet the average fixation duration was 181.8 milliseconds for the colour code and 187.8 milliseconds for the black-and-white. As indicated by the Mann-Whitney test, the mean fixation durations were not significantly different for either the first snippet (U = 142.0, p = 0.95) or the second snippet (U = 144.0, p = 1.00). Similarly, the mean fixation duration within AOIs did not differ significantly when reading in black-and-white or colour for either the first code snippet (U = 124.0, p = 0.49) or the second code snippet (U = 140.0, p = 0.89).

In order to visualise the time spent on the lines of code, a heatmap was generated that aggregated all participant data. However, instead of a standard heatmap that tends to obscure the underlying stimulus, for the purposes of the paper the words (or AOIs) of each code snippet were colour-coded, similar to the visualisation used by (Busjahn, Shulte, & Busjahn, 2011), using the same colour scale as a heatmap. Heatmap colours were scaled relative to the longest fixation within an area of interest. Words that are shaded in warm colours were fixated on for longer than words that are shaded in cooler colours. Brackets, semi-colons and words with no fixations are shown in black.

As can clearly be seen in Figure 4, there were minor differences in the fixation durations between the black-and-

This article is licensed under a
Creative Commons Attribution 4.0 International license.

white and colour, code snippets, as expected based on the results of the statistical tests. For the second snippet (Figure 5), there were more differences detected with the heatmap than with the first snippet. However, in general the distribution of the fixations and durations are similar between pieces. Cooler colours largely correspond between the black-and-white and its corresponding colour code snippet, and while the degree of the warmer colours differ slightly, areas that are warm in black-and-white are, to a large extent, also warm in the colour version.

Regressions

The number of regressions or regressive saccades made while reading is an indication of the difficulty experienced while reading the piece of text. Due to the nature of code reading, it was expected that there would more be regressions than with normal text reading but that an elevated number of regressions would still be indicative of difficulties in comprehension. For example, the presence of a loop may naturally lead to regressions while tracing the code but should the participant experience difficulty in comprehending the code, the regressions would increase beyond those naturally required to read the code. The following hypothesis was thus formulated:

H_0: There is no difference in the number of regressions when reading code in black-and-white versus colour.

Mann-Whitney tests show that there is no significant difference between the number of fixations on the colour code snippet and the black-and-white snippet for either the first snippet ($U = 115.0$, $p = 0.31$) or the second snippet ($U = 129.0$, $p = 0.61$).

The chart (Figure 6) shows the spread of regressive fixations (fixations directly following a regressive saccade) versus total fixations for each code snippet. As can clearly be seen, the regressions were between 20-30% of the total fixations, with a higher percentage of regressions on the colour code snippets for both snippets of code.

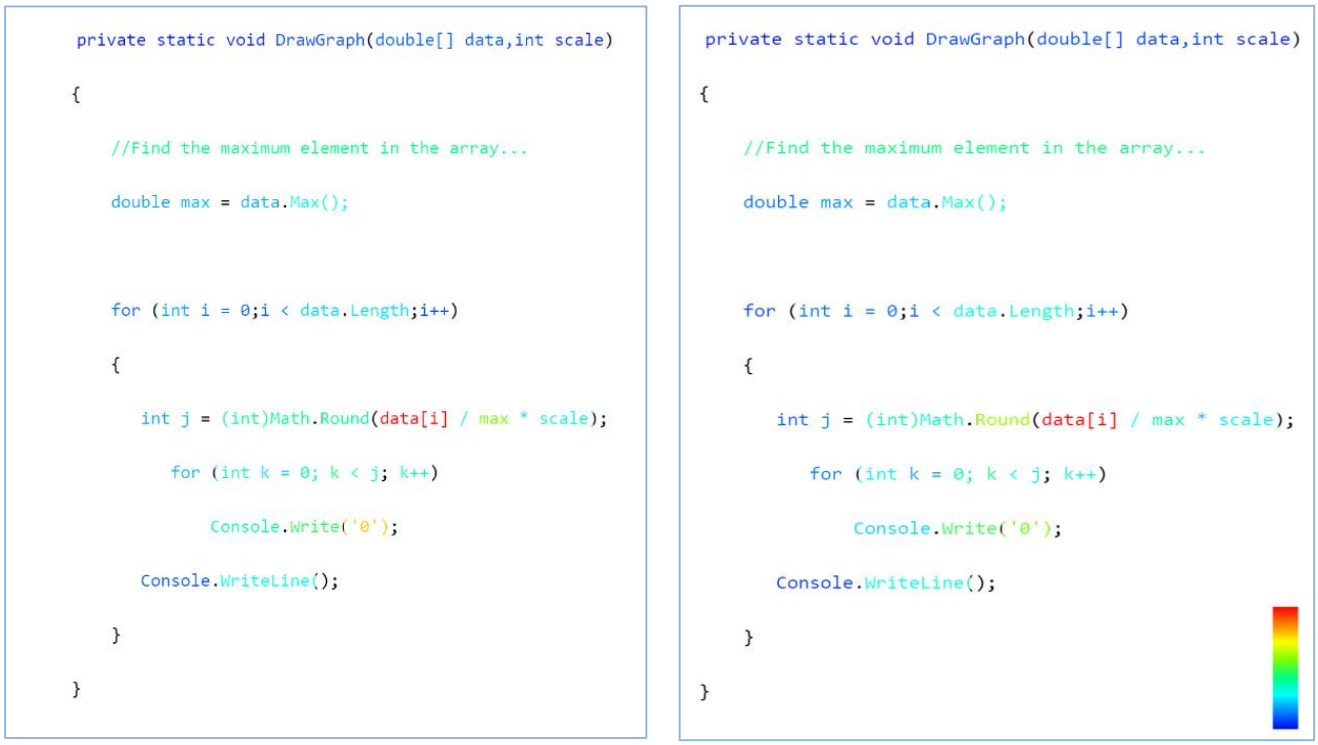
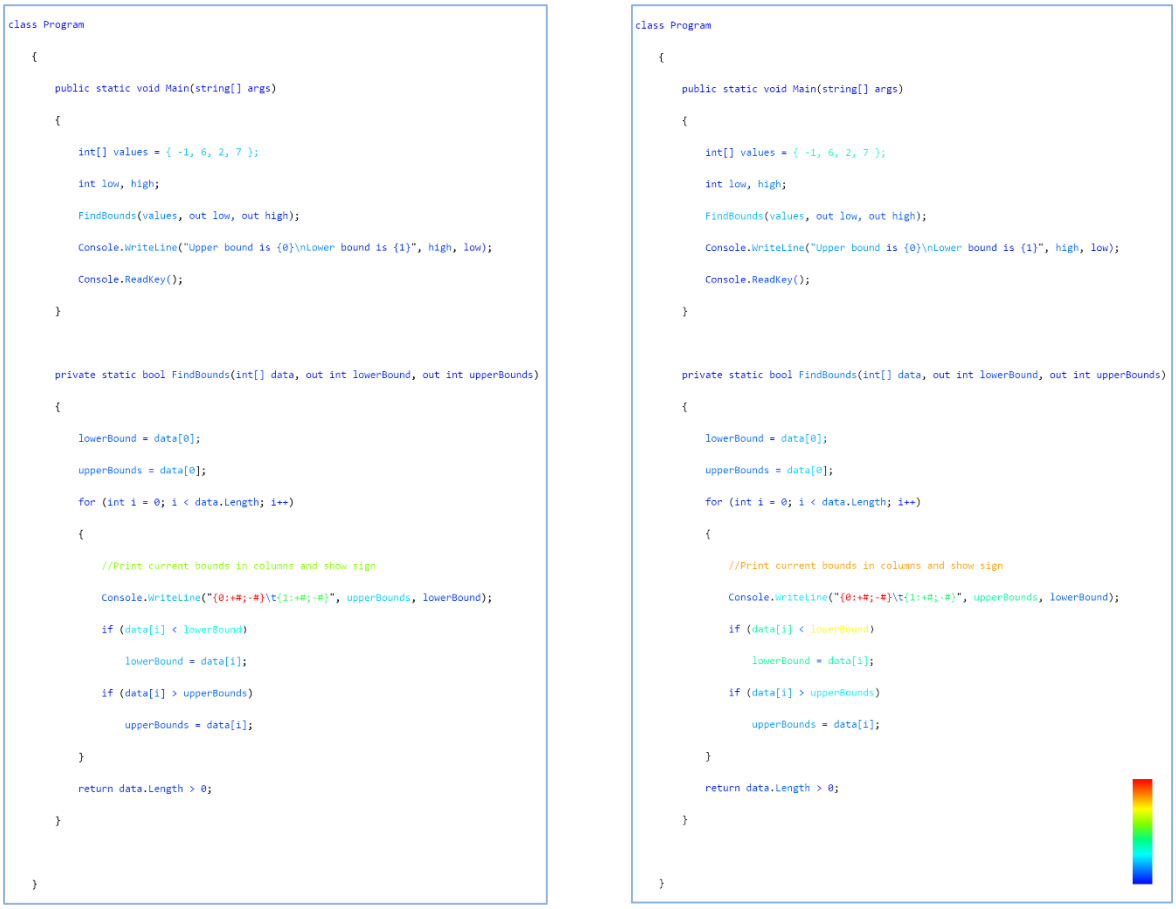


Figure 4. Heatmaps of first snippet for (a) black-and-white code on the left and (b) colour code on the right

This article is licensed under a
Creative Commons Attribution 4.0 International license.

ISSN 1995-8692



```
class Program
    {
        public static void Main(string[] args)
        {
            int[] values = { -1, 6, 2, 7 };
            int low, high;
            FindBounds(values, out low, out high);
            Console.WriteLine("Upper bound is {0}\nLower bound is {1}", high, low);
            Console.ReadKey();
        }

        private static bool FindBounds(int[] data, out int lowerBound, out int upperBounds)
        {
            lowerBound = data[0];
            upperBounds = data[0];
            for (int i = 0; i < data.Length; i++)
            {
                //Print current bounds in columns and show sign
                Console.WriteLine("{0:+#;-#}\t{1:+#;-#}", upperBounds, lowerBound);
                if (data[i] < lowerBound)
                    lowerBound = data[i];
                if (data[i] > upperBounds)
                    upperBounds = data[i];
            }
            return data.Length > 0;
        }

    }
```

Figure 5. Heatmaps of second snippet for (a) black-and-white code on the left and (b) colour code on the right

Figure 6. Fixations (blue bar) vs regressive fixations (orange bar) expressed as a percentage of total fixations for all code snippets

Subjective analysis of the code

After each completed code snippet, the participants were asked to answer three questions about that particular piece of code. Answers were rated on a 5-point Likert scale.

The first question evaluated the perceived difficulty of the code, the second the readability and the third the appearance of the code. Readability relates to how easily the participant felt the code was to read, while appearance is related to how aesthetically pleasing the code was to view. Summaries and results of the Mann-Whitney test are shown in Table 3 for the first code snippet.

There was no significant difference between the black-and-white and colour code for any of the three questions. However, based on the average, the colour code was perceived more positively than the black-and-white in all three instances. At a more lenient α-value, there was a significant difference in opinion for the readability of the code.

This article is licensed under a
Creative Commons Attribution 4.0 International license.

ISSN 1995-8692

Table 3
Subjective analysis results for first code snippet

	Difficulty	Readability	Appearance
Black and white	$\bar{x}$ = 2.7	$\bar{x}$ = 2.5	$\bar{x}$ = 2.5
Colour	$\bar{x}$ = 2.3	$\bar{x}$ = 1.8	$\bar{x}$ = 1.5
	U = 98.0, p = 0.27	U = 79.0, p = 0.07	U = 101.5, p = 0.34

Summaries and results of the Mann-Whitney test are shown in Table 4 for the second code snippet.

Table 4
Subjective analysis results for second code snippet

	Difficulty	Readability	Appearance
Black and white	$\bar{x}$ = 2.6	$\bar{x}$ = 2.5	$\bar{x}$ = 2.8
Colour	$\bar{x}$ = 2.8	$\bar{x}$ = 1.9	$\bar{x}$ = 2.1
	U = 105.5, p = 0.58	U = 75.5, p = 0.07	U = 75.0, p = 0.07

Once again, the colour code was perceived more positively than the black-and-white in terms of readability and appearance, although not significantly so. The difficulty of the colour code was, in this instance, marginally more negative than the black-and-white which is surprising. However, using an α-value of 0.1, there is a significant difference between both the readability and appearance, with the colour version being more positively received. The longer length of this code snippet could have contributed to the difference in the satisfaction in this instance.

In summary it could be said that the colour of the code was subjectively found to be more aesthetically pleasing and had a direct effect on the perceived difficulty to the participants.

Power Analysis

Since the null hypothesis could not be rejected for any of the analysis performed, a post-hoc power analysis was performed for each of the metrics. Table 5 summarises the calculated power for each of the metrics and the code snippets.

As is clearly illustrated in Table 5, the majority of the values all represent extremely low statistical power for detecting the small observed effect sizes. Furthermore, an a priori power analysis estimates a sample size of 51 in order to achieve 0.8 power with a moderate effect size of 0.5. Therefore, the study is underpowered and it is suggested that the study be repeated with a larger sample size to determine the reliability of the findings reported in this paper.

Table 5
Power analysis results for all metrics and both code snippets

	First snippet	Second snippet
Fixation count	0.63	0.08
Fixation count per line	0.31	0.13
Fixation count per word	0.41	0.06
Fixation duration (whole stimulus)	0.07	0.10
Fixation duration per AOI	0.24	0.19
Regressions	0.41	0.10

Limitations

While every effort was made to ensure the groups were homogeneous, it may be advantageous to repeat a similar study using a paired protocol. For this each participant would read a code snippet in colour and one of comparable difficulty in black-and-white. This could provide more insight into whether there may be a difference in reading behaviour based on the presentation of the code. The small sample size caused by the necessity to conduct the tests on an individual basis is a limitation of the study. Further research should include a larger sample to ensure robustness of the results. This is further supported by the fact that the study was shown to be underpowered. Additionally, reading behaviour for code on e-readers compared to printed material can be investigated to determine if the medium causes a difference.

Conclusion

The study investigated whether there was a difference between reading code in black-and-white and colour with the view to determine whether students are better served by access to electronic resources via electronic devices. Electronic devices may also facilitate provision of academic text books in colour, with the added advantage that they are potentially cheaper and more accessible than printed material.

Participants were required to read code snippets in black-and-white or colour. The task associated with the code snippet was formulated to ensure proper reading and understanding of the code as participants were required to write down the output of the code snippet. Each code snippet

This article is licensed under a
Creative Commons Attribution 4.0 International license.
ISSN 1995-8692

was followed by a questionnaire evaluating the appearance of the code.

Analysis of reading behaviour was based on number of fixations, fixation durations and number of regressions, which are common metrics used for analysing reading behaviour and were shown to be relevant to reading code as well.

The number of fixations were measured for the whole code snippet, per line of code and per word. In all three cases, no significant difference was found between black-and-white and colour. In terms of the total number of fixations for the whole code snippet, this confirms prior findings that syntax highlighting does not significantly affect fixation count (Sarkar, 2015).

Fixation durations were analysed per code snippet and per word. Contrary to prior findings (Busjahn, Shulte, & Busjahn, 2011) fixation durations are comparable to those of normal reading and were not longer. This study found mean fixation durations between 100 and 200 milliseconds, which is not in range of the 351 ms found in (Busjahn, Shulte, & Busjahn, 2011). Again, as with the fixation counts, the mean difference between the fixation durations between black-and-white and colour was not significant, which confirms prior findings (Sarkar, 2015). Furthermore, it was not significant for mean fixation durations within AOIs between the code snippets.

The percentage of regressions was marginally less than found by Busjahn, Shulte, & Busjahn (2011). The slightly lower regression count can be attributed to the subjective way in which the regressions were counted. The percentage of regressions was fairly consistent between code snippets, a fact confirmed by the analysis since there was no significant difference between code snippets in terms of the number of regressions.

In light of the results, it could be concluded that participants did not experience more difficulty when reading code in black-and-white versus colour.

However, the metrics for the colour snippets were consistently lower than for the black-and-white. This indicates that to a degree, although not significantly so, there was less difficulty experienced with the colour code than the black-and-white. This could perhaps be attributed to the fact that novice programmers were used who do not have much experience reading code and consequently struggled to interpret the code regardless of the way in which the code was presented. This could possibly change if more experienced programmers were used who are accustomed to the syntax highlighting. However, their experience may allow them to read black-and-white code just as efficiently. This is therefore something that should be investigated with a further study.

Therefore, in practical terms it could be surmised that there should be no real disadvantage to students making use of text books in black-and-white as opposed to colour, although students did seem to prefer code in colour. Therefore, low cost e-readers may provide a viable platform for students seeking alternatives to printed material.

References

Bednarik, R. & Tukianinen, M. (2006). An eye-tracking methodology for characterizing program comprehension processes. In *Proceedings of the 2006 Symposium on Eye Tracking Research and Applications*, 125 – 132, San Diego.

Binkley, D., Davis, M., Lawrie, D., Maletic, J.I., Morrell, C., & Sharif, B. (2013). The impact of identifier style on effort and comprehension. *Empirical Software Engineering*, 18(2), 219-276.

Busjahn, T., Shulte, C., & 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*, 1-9, Finland.

Busjahn, T., Shulte, C., Sharif, B., Simon, Begel, A., Hansen, M., Bednarik, R., Orlov, P., Ihantola, P., Shchektova, G., & Antropova, P. (2014). Eye tracking in computing education, In *Proceedings of the tenth annual conference on International Computing Education Research*, 3-10, Glasgow, Scotland.

Crosby, M. & Stelovsky, J. (1990). How do we read algorithms? A case study. *Computer*, (1), 25-35.

Dimitri, G.M. (2015). PPIG 2015: The impact of syntax highlighting in Sonic Pi. In *Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015)*, 59 – 68.

Faul, F., Erdfelder, E., Lang, A-G., & Buchner, A. (2007). G*Power 3: A flexible statistical power analysis program for the social, behavioral, and biomedical sciences. *Behavior Research Methods*, 29(2), 175 – 191.

Gregory, R.L. (1966). *The eye and the brain: The psychology of seeing*. London: World University Library.

Horcher, A-M., & Tejay, G. (2012). Usability and the acceptance of e-books and e-reading devices. In T-T. Goh, B-C. Seet & P-C. Sun (Eds) *E-books and e-readers for e-learning*. Wellington, New Zealand: Victoria Business School, Victoria University of Wellington, 223-260.

McCabe, T.J. (1976). A complexity measure. *Software Engineering*, SE-2(4), 308-320.

This article is licensed under a
Creative Commons Attribution 4.0 International license.

ISSN 1995-8692

McCabe, T.J., McCabe, T.J., & Fiondella, L. (2012). Uncovering weaknesses in code with cyclomatic path analysis. *CrossTalk*, July/August, 9-14.

Menz, C. & Groner, R. (1984). The acquisition of a new letter system: effects of word length and redundancy. In A. Gale, F. Johnson (Eds.), *Theoretical and Applied Aspects of Eye Movement Research*, Elsevier Science Publishers B.V, Amsterdam.

MSDN. (n.d.) MSDN Library. Retrieved from https://msdn.microsoft.com/en-us/library/ms123401.aspx.

Nakov, S., Dimitrov, D., Germanov, H., Stoynov, M., Valkov, K, Bivas, M., …Yosifov, Y. (2013). Fundamentals of computer programming with C#. Retrieved from http://www.introprogramming.info/english-intro-csharp-book/.

Olsen, A.N., Kleivset, B., & Langseth, H. (2013). E-book readers in higher education: Student reading preference and other data from surveys at the University of Agder. *Sage Open*, April-June 2013, 1-8.

Oracle. (n.d.). Oracle documentation. Retrieved from http://docs.oracle.com/javase/7/docs/api/.

Rayner, K. (1998). Eye movements in reading and information processing: 20 years of research. *Psychological Bulletin*, 124(3), 372-422.

Rayner, K. (2009). Eye movements and attention in reading, scene perception, and visual search, *The Quarterly Journal of Experimental Psychology*, 62:8, 1457-1506.

Sarkar, A. (2015). The impact of syntax colouring on program comprehension. In *Proceedings of the 26th Annual Conference of the Psychology of Programming Interest Group (PPIG 2015)*, 49 – 58.

Smith, M., Kukulska-Hulme, A., & Page, A. (2012). Education use cases from a shared exploration of e-books and iPads. In T-T. Goh, B-C. Seet & P-C. Sun (Eds) *E-books and e-readers for e-learning*. Wellington, New Zealand: Victoria Business School, Victoria University of Wellington, 25-53.

Turner, R., Falcone, M., Sharif, B., & Lazar, A. (2014). An eye-tracking study assessing the comprehension of C++ and Python source code. In *Proceedings of Symposium on Eye-Tracking Research and Applications (ETRA)*, 231 – 234.

Zhang, H., Zhang, X., & Gu, M. (2007). Predicting defective software components from code complexity measures. In *Proceedings of 13th IEEE International Symposium on Pacific Rim Dependable Computing*, 93-96.

This article is licensed under a
Creative Commons Attribution 4.0 International license.

ISSN 1995-8692