Indentation: A Simple Matter of Style or Support for Program Comprehension?

Jennifer Bauer
University of Passau

Johannes C. Hofmeister
University of Passau

Janet Siegmund
University of Passau

Sven Apel
University of Passau

Abstract

Most modern programming language style guides suggest a specific indentation depth to convey the structure of code (e.g., two or four spaces). In this study, we examine the effect of indentation depth on program comprehension, to provide justification for suggested levels of indentation made by modern style guides. In the course of our study, we asked 22 participants to evaluate the output of Java code snippets with different levels of indentation. We also measured correctness and speed of the responses, and asked participants to rate the difficulty of the code presented. We employed an eye tracker to gain additional insights into the influence of indentation on visual effort. We found a small effect of indentation depth on fixation duration, but could not detect differences in correctness or visual effort. Our findings suggest that indentations affect program comprehension at a lower, perceptual level of processing, rather than higher level reasoning. Our research design and methods with respect to visual effort could be a promising starting point for future studies in this field.

1 Introduction

Program comprehension is an important task, as programmers spend about half of their time comprehending source code [2]. Most style guides suggest particular levels of indentation to support developers in the process. However, such recommendations are often unjustified, and empirical research on the topic is scarce. In 1983, Miara and colleagues [3] found that an indentation of 2 to 4 spaces is most helpful for comprehending Pascal code [3]. To this end, they asked novices and experts to answer questions about several programs' function and rate the difficulty, while the programs were treated with either 0, 2, 4 or 6 spaces of indentation. They found that a medium-level indentation resulted in the most correct answers. We replicate the study of Miara and colleagues but adopt its design to apply to a more modern context, in which we use Java instead of Pascal. Additionally, we included an eye tracker, to be able to explain differences in participant behavior. This way, we intend to provide empirical evidence for an optimal level of indentation.

Program comprehension encompasses several cognitive processes. On the most fundamental level, it requires that readers *decode* the presented code (i.e., they can differentiate the different keywords, identify its structure, etc.). In a nutshell, programmers must first be able to perceive the code's structure, before they can understand it.

Indentations help to convey the structure of source code by visually grouping parts of the code into cohesive units, for example, indented lines under a function declaration indicate that the code below the signature are subordinate to the declaration, and thus represent the function's body.

Indentations should improve the speed with which readers can process code visually, as they make it easier to perceive its structure. However, we believe that there is a limit to this idea. If indentations are too wide, they shift different units far apart, making it difficult to still identify which units belong together. The surrounding code moves out of focus and the readers might have trouble to retain the context of lines read. Thus, we argue that code should be indented far enough so that its structure is conveyed, but close enough to retain context.

We reasoned that the level of indentation affects two aspects of program comprehension: correctness and time. Furthermore, as indentation changes the layout of code, we assumed that visual effort is also influenced by the depth of indentation.

2 Method

To answer our research questions, we designed an experimental study. We asked participants ($n = 22$) to determine the console output of four Java snippets and later rate the difficulty of these snippets by ordering them by difficulty. During each trial we tracked the participants' gaze. Most participants were Computer Science students or working at the department of Computer Science ($n = 19$). Three participants were employed as software developers in a software company.

Each snippet presented a problem involving an array (of numbers or subparts of a string) and contained two block structures (i.e., if-then-else block or loops), that is, within each method, two levels of indentation were applied. All snippets were 17 lines long. We varied indentation depth as an independent variable with four levels: zero, two, four, and eight spaces. We followed the Oracle Java coding conventions [4]

to decide when to indent. The study used a repeated-measures design, in which every participant saw every code snippet and every level of indentation once. We used this design to account for inter-individual differences.

To measure program comprehension, we asked participants to determine the output of a snippet. We further asked participants to rate the snippets' difficulty, because we assumed that indentation could affect the participants perception of difficulty (e.g., snippets without indentation would be considered more difficult). We asked participants to first rate all snippets presented with a normalized indentation depth of four spaces. Second, we showed the code with the actual indentation that participants had seen during the trials before.

We included eye tracking in our study, because it provides additional evidence for the comprehensibility or difficulty of programs, as gaze behavior can indicate challenges while reading code. We operationalized participants' visual effort by capturing *fixations* and *saccades* [1]. Fixations are moments during which participants' eyes rest in a certain location. Saccades are transitions between fixations. In our study, we evaluated duration and frequency of fixations, and looked at saccadic amplitudes (the distance the eyes travel during saccades).

3 Results

We found no significant effect of level of indentation on comprehension, as measured by the correctness of the stated output ($\chi^2(3) = 3.32, p = .36$) nor on response time ($F(3, 63) = 0.44, p = .72$). We found no significant effect on subjective difficulty with equal indentation ($\chi^2(3) = 4.64, p = .20$) nor with actual indentation ($\chi^2(3) = 5.35, p = .15$). Regarding visual effort, there was no significant effect of indentation on fixation rate ($\chi^2(3) = 7.36, p = .06$) and saccadic amplitude ($F(3, 63) = 1.69, p = .18$). We found a significant, small effect on fixation duration ($F(3, 63) = 2.85, p = .045, \eta_p^2 = 0.028$). F denotes Friedman's test, except for response times, where we used a repeated-measures ANOVA.

4 Discussion

Although we observed subtle differences for correctness, response times, difficulty, and visual effort, only the main effect of fixation duration was statistically significant.

We interpret these results as support for the idea that while the perceptual processing of code is required to understand it, higher level processing, such as understanding its semantics and reasoning about its functionality, affect program comprehensibility more strongly. The influence of indentation could have been masked by these side effects, so it might well be that the effect of indentation comes more into play when the code is longer and more complex.

It is possible that indentation does not influence program comprehension at all. This would render the actual realization (two, four, six or another depth) of indentation irrelevant, as long as it's consistent within a certain group.

The level of indentation did not influence the participants' rating of difficulty. We conclude that different indentations neither influence measures of actual performance or personal preferences, and that other factors outweigh the influence of indentations. However, given that ours small sample size, we suspect that our experiment was underpowered to detect effects of this magnitude.

We argued that indentation may affect visual effort. We found a small main effect for fixation duration, indicating that participants look at code longer when it is further apart.

Based on our data, we cannot offer a definitive suggestion for an optimal level of indentation, as no level of indentation had a clear influence on comprehension correctness or speed. Nevertheless, it might be worth to investigate how indentation interacts with other properties of source code, such as complexity, nesting depth, or length, or even other kinds of spacing. Surely, programmers would be unhappy if they were to read long programs condensed into one line. This should also have have a measurable impact on their performance.

References

[1] Holmqvist, K., Nystrm, M., Andersson, R., Dewhurst, R., Jarodzka, H., and Van de Weijer, J. *Eye tracking: A comprehensive guide to methods and measures.* OUP Oxford, 2011.

[2] LaToza, T. D., Venolia, G., and DeLine, R. Maintaining mental models: a study of developer work habits. In *Proceedings of the 28th international conference on Software engineering* (2006), ACM, pp. 492–501.

[3] Miara, R. J., Musselman, J. A., Navarro, J. A., and Shneiderman, B. Program indentation and comprehensibility. *Communications of the ACM 26*, 11 (1983), 861–867.

[4] Sun Microsystems Inc. *Code Conventions for the Java Programming Language.* `http://www.oracle.com/technetwork/java/codeconvtoc-136057.html`, visited on September 18, 2017.