University of Passau

Department of Informatics and Mathematics

Bachelor Thesis

Indentations - A Simple Matter of Style or Support for Code Comprehension?

Author:

Jennifer Bauer

October 12th, 2017

Advisors:

Dr.-Ing. Janet Siegmund

University of Passau

Johannes Hofmeister

University of Passau

Bauer, Jennifer:
Indentations - A Simple Matter of Style or Support for Code Comprehension?
Bachelor Thesis, University of Passau, 2017.

Abstract

Based on the positive results of a study by Shneiderman, this thesis reexamines the effect of indentation on program comprehension. We also included the subjective assessment of difficulty, and extended the original design to gain additional insights about the influence of indentation on visual effort. The aim was to give an empirical justification for recommendations of indentation made by style guides. In the course of our study, we asked 22 participants to calculate the output of Java code snippets with different levels of indentation, while their gaze was logged by an eye tracker. Although we found no significant evidence for the examined effects, this thesis could be a starting point for future studies in this field.

In this thesis, I refer to myself by saying ”we”, as this makes reading my paper more comfortable in my opinion.

Contents

List of Figures

List of Tables

List of Code Listings

1 Introduction

Unlike text in a book, a program's code is not linear, but it uses spaces, line breaks, and empty lines to organize code and convey, which structures belong together. These and other attributes are summarized under the term code style. A carefully chosen code style cannot only help the coder himself, but also reviewer of his code, people who need to maintain or test it. In order to have a common ground, coding conventions were established, either for a whole language or individually established by a company. A suggested level of indentation is often part of those recommendations for programming languages. Those proposals, however, are not explicitly justified and might only be an educated guess about the optimal or most suitable level of indentation by the authors of those guidelines.
This surprising, because although indentation is to a large degree a subtle component of style, it is omnipresent and can hardly be escaped when working with code. Code is indented when each line of code is shifted a certain number of spaces to the right in relation to the surrounding code parts, with cascading depth. Figure 1.1 shows an example of a code with and without indentations. Some languages, such as Python [Foua], explicitly require indentations for indicating the beginning and end of block structures. In those cases, programmers have to commit to the specific requirements of those languages, and are limited in the usage of indentations as stylistic device. However, in languages such as Java, indentations are purely optional. Here they do

```
for (int i=0;i<6;i++){
if(i%2 == 0) {
System.out.print("o");
} else {
System.out.print("e");
}
}
```

(a) Non-indented Code

```
for (int i=0;i<6;i++){
        if(i%2 == 0) {
                System.out.print("o");
        } else {
                System.out.print("e");
        }
}
```

(b) Indented Code

Figure 1.1: Comparison of Code with and without Indentations

not bear any syntactic meaning. They can therefore be set arbitrarily and are left to personal preferences. For Java there are several suggestions for Code Conventions, but the most known might be style guides of Oracle [SM] and Google [Goob] who propose an indentation of four spaces. In order to motivate this choice as a recommendation for Java coders, more research is needed. For Pascal a study by Shneiderman et al. showed that an indentation of 2-4 spaces is most helpful for comprehending code [MMNS83]. This thesis aims to find out, whether the same holds true for Java. Like Shneiderman, we also look at the subjective assessment of difficulty and whether it changes depending on indentation. Additionally, since indentation is likely to supplementary affect gaze behavior by influencing the layout of the code, the effects of indentation on visual effort are examined. In summary, the goals of this thesis are:

- Comparing the comprehensibility of Java code with different levels of indentation
- Determining the effect of indentation on subjective assessment of difficulty
- Examining, whether indentation affects visual effort.

The study we conducted to answer this questions, was a loose replication of Shneiderman's study, extended with eye tracking (see [KUMA+84] for the advantages of non-exact replications).

Structure of the Thesis

This thesis is organized as follows: In Chapter 2 we motivate, how and why indentations could have an impact on program comprehension, subjective assessment of difficulty and visual effort. The resulting research questions this thesis wants to answer are stated there as well. Chapter 3 describes the details of the original study by Shneiderman. Thereafter, we describe the study conducted for this thesis in Chapter 4. Chapter 5 lists the results found, as well as their statistical analysis. The findings of this thesis are discussed in Chapter 6. Potential threats to validity are mentioned in Chapter 7. Finally, Chapter 8 gives an outlook on future work.

2 Background

Before describing the original study by Shneiderman and our study, we present why indentation could affect program comprehension, subjective assessment of difficulty and visual effort. At the end of this section, the resulting research questions are listed.

2.1 Program Comprehension and Indentation

In Java or Pascal, indentation is viewed as a stylistic device, since it does not affect the execution of a code in those languages. But, as Shneiderman showed, indentation can have an influence on the *comprehension* of the execution of a program. In the following section, we want to discuss why/how indentations could help readers in understanding programs. One possible explanation might be the "Law of Proximity", an aspect of the Gestalt psychology: People tend to group objects that are close to each other and think of them as belonging to the same category [Wer23]. Indentation supports this process: it aligns statements that belong to the same structure (e.g. a loop or an if-else-statement). When the level of indentation corresponds with the nesting depth, block structures can be easier recognized without concentrating on other indicators, such as brackets. Thus, indentations help to convey the code's semantic. The reader might use the time benefit gained this way to focus on other aspects of the code, for instance variable assignment. When the time for comprehension tasks is limited, this could make the difference in completing the problem *in time*. To figure out whether that is indeed the case, it is important to measure response times along with comprehension.
We expect that comprehension behaves like a parabola in dependence of indentation: None to little indentation results in bad comprehensibility, medium levels of indentation do support the process of understanding code and with deep indentation, the comprehensibility decreases again. This assumption is based on the following: We think that a moderate level of indentation (two to four spaces) supports the grouping of code into coherent chunks, as code indented this way stands out enough to be easily recognized as belonging to the same structure. A greater depth of indentation, however, separates the single parts of the code too much, so that the embedding of

those parts into the whole picture might become more difficult. Non-indented code completely misses the visual clues provided by indentation. Here, the readers needs to examine other features of block structures more carefully in order to relate code lines to the respective code entities.
In accordance with Shneiderman's findings and due to the reasons above, we therefore expected codes with an indentation of two to four spaces to perform better in terms of program comprehension and response time than zero or eight space.

2.2 Subjective Assessment of Difficulty and Indentation

Besides objective difficulty of the different levels of indentation, the *subjective* perception is also worth looking at. Finding that one level of indentation is indeed the best in terms of measured comprehensibility does not necessarily mean that a programmer's attitude concurs with this result. It might be that, for reasons of habit, aesthetics or others, a different level of indentation makes people believe to perform better. Those discrepancies (if they do occur) should be kept in mind, when recommending a level of indentation. Trading off the objective best result for the experienced best one then depends on how much the subjective well being is emphasized.
Two spaces and four spaces are the standard level of indentation in most coding conventions. For example, two spaces are recommended for JavaScript [Gooc] and C++ [Gooa], and four spaces for Java [SM], Python [Foub] and C-Sharp [Mic]. Since those guidelines are often the basis for the auto-formatters of Integrated Development Environment (IDE)s, we expected codes with this indentation to be rated as easier, compared to the rather unusually zero- or eight-space indented code.

2.3 Visual Effort and Indentation

Aside from habit and the discussion about potential effects on comprehensibility, one possible explanation, for why indentions are used at all, might be that unindented code is not very convenient to read. In contrast, very deep indentation could make code harder to scan, as it increases the horizontal dimension of jumps between differentially indented code. To figure out, whether this is just a vague intuition, or indentation indeed affects visual effort, we included eye tracking in our study. An example for how eye tracking can actually give new insights into existing findings is the following: D. Binkley et al. investigated in 2009 the effect of the style of identifier names (camelCase vs. under_score) on how fast and accurate people were able to alter code [BDLM09]. They found that under_score resulted in faster response times. This study was then replicated by B. Sharif and J.Maletic in 2010 [SM10] extended by eye tracking. This time, they additionally focused on visual effort using measurements such as fixation duration. By doing so, they additionally found that underscore style also improves visual effort, next to response times.

There exists a large number of variables measuring gaze behavior and visual effort. Therefore, one has to decide in advance, on which aspects to look at in order to prevent fishing for effects. Before delving into the measurements of gaze behavior that were chosen for this study, a short explanation of important terms is given.
The two key terms in eye tracking are *fixation* and *saccade*. A fixation takes place, when the eye is resting on a point. It "lasts anywhere from some tens of milliseconds up to several seconds" [HNA+11]. There are numerous measures regarding fixations, among others the fixation positions, duration, count, and rate.
The counterpart of fixation is the *saccade*. A saccade is the transition between two fixations. It does not need to follow a straight line, but can be curved [HNA+11] before ending in a fixation. Measured regarding saccades include saccadic velocity, saccadic amplitude and the directions of saccades.
There are many more measurements regarding gaze, such as for example pupil diameter, blink rate, dwell time in areas of interest. The list below explains the ones we choose to investigate in this study.

Fixation Duration The duration of a fixation is "likely to be the most used measure in eye-tracking research" [HNA+11]. In 1980, Just and Carpenter proposed, "that there is no appreciable lag between what is being fixated and what is being processed." [JC80]. This implies that the longer a fixation is, the longer the fixated part is processed. This statement has to be handled with some caution, because "some processing trace of a fixated item may continue for a very long time after the eye has left the fixated position" [HNA+11]. Nevertheless, fixation duration is a good indicator for processing text or code, because only visually perceived items can be handled by the reader.

Fixation Rate The number of fixations per second is known as the *fixation rate*. It is related to fixation duration, but "includes saccade and blink duration" [HNA+11]. Nakayama et al. found that fixation rate (or "gazing time" as they call it) decreases when task difficulty increases [NTS02]. A high fixation rate implies that the reader jumps more from fixation to fixation, so from item to item. In contrast, a low fixation rate may either occur, when people have a harder time to process some parts of the task, or when they feel no need for looking back for understanding single items.

Saccadic Amplitude The spatial length of a saccade is also known as the *saccadic amplitude*. It relates to the jumps made by the reader in the text/code, and plays an important role in search tasks. The more difficult the search task, the shorter the saccadic amplitude [PE08]. Higher difficulty in tasks related to counting was also shown to result in decreased saccadic amplitudes [MKW+90]. Naturally, the saccadic amplitude can also be influenced by the layout of the stimulus. If visual clues lie farther apart from each other, the saccades between them cover a larger distance.

2.4 Research Questions

The above discussed topics lead to the following research questions:

$\mathbf{RQ_{Correctness}}$	Does indentation affect correctness?
$\mathbf{RQ_{Time}}$	Does indentation affect response time?
$\mathbf{RQ_{Rating}}$	Does indentation affect the subjective rating of difficulty?
$\mathbf{RQ_{Fix:Duration}}$	Does indentation affect fixation duration?
$\mathbf{RQ_{Fix:Rate}}$	Does indentation affect fixation rate?
$\mathbf{RQ_{Sacc:Amplitude}}$	Does indentation affect saccadic amplitude?

3 Original Study

This thesis is of course not the first to examine the effects of indentation on program comprehension. Shneiderman et al. [MMNS83] conducted a study in 1983 targeting the correlation of indentation and program comprehensibility. Programming novices and experts were asked to answer questions about code snippets with varying indentation and to estimate their difficulty. Since Shneiderman's study is the base of this study, the following gives a short description about it.

Variables

The one code used in the study was treated with zero, two, four, and six spaces of indentation. Additionally, the code within Pascal's begin-end blocks was either aligned with the begin and end statements, or also indented. Those alternations to the code and the level of programming experience were investigated to have an effect on the score in the comprehension quiz and the subjective assessment of difficulty.

Participants

The authors divided the participants into novices and experts by the years of programming experience they declared to have. Participants were seen as novices, when they "had less than three years of programming experience in school and/or less than two years of professional programming experience" [MMNS83]. All remaining subjects were counted as experts. Novices were mostly participants in an "intermediate-level programming class in Pascal" [MMNS83] and had already worked with more complex code than the one used in the study. The experts were recruited from an advanced programming course and "were graduating computer-science majors" [MMNS83]. Overall, 86 participants were included in the statistical analysis.

Material

The Pascal code used in the study "calculated the number of occurrences of a word for a given string of input" [MMNS83]. It included several syntactic structures such as " sets, records, packed-arrays,while-loops, and if-then-elses." [MMNS83]. To measure comprehension, the authors asked nine multiple choice questions about the program. In addition, the participants had to give a short explanation regarding its function, and an assessment of the difficulty in a rating of 1-7. Another two questions at the end were dedicated to programming experience. All tasks were listed on one sheet.

Procedure

In the study, the subjects had 20 minutes to fill out the questionnaire. Each subject got *one* of eight code versions. The supposed novices were assessed simultaneously in the same room. The experts were also tested together.

Results

Shneiderman found that two to four spaced indentation supported comprehension the most, with two space having the best results. Experts performed generally better than novices. Zero-indented code had the worst quiz results, and was also rated as most difficult. Their explanation for the better performance of moderate levels of indentation is that "[w]ith 2–4-space indentation levels [...], the program is more compact and the control blocks do not become obscured by increased nesting levels." [MMNS83]

4 Study

We conducted an empirical study with a within-subject design to answer the proposed research questions. The participants were asked to calculate the output of Java codes. During these tasks, we measured their response time as well as their gaze positions. Afterwards they had to rate the difficulty of the codes. The independent variable was the level of indentation.

4.1 Participants

We recruited 39 participants for study. Due to a bug in an auxiliary code script, only 22 could be considered in the later evaluation. Hence, all following statements refer to those 22 participants.
The participants were recruited via personal invitation, word-of-mouth recommendation, and an invitation mass mail addressing all students of the Computer Science and Mathematics Faculty of Passau. To motivate potential candidates to participate, we advertised the use of an eye tracker, and underscored the value of the scientific contribution. Additionally, we incited participation with sweets and a lottery for an amazon gift card.
The participants were mainly Computer Science students or students of Computer Science related subjects such as Internet Computing (see Table 4.3) and therefore rather young (see Table 4.1). Most of the students were undergraduates in the fourth or fifth semester (M[1]=4.7, SD[2] =2.1), who spent on average seven hours per week on programming (M=7.4, SD=7.3). In addition to their studies, half of the students were also working as student assistants.

[1]Mean
[2]Standard Deviation

Attribute	M	SD
Age	24.7	6.2
Years of Java Exp.	5.5	3.7
Year of overall Exp.	7.8	6.9

Table 4.1: Summary Participants

Knowledge	Number
Project Experience	10
Regular	9
Basis Knowledge	2
No Experience	1

Table 4.2: Summary Java Knowledge

Another group of participants was working at the department of Computer Science (Ph.D. students, post docs, university staff). Three other participants were employees in a software company located in Passau.
The participants' level of education and self-assessed level of Java knowledge are summarized in **??** and Table 4.2. The participants further answered questions about they sizes of projects they have read or worked on. Due to the homogeneity of the participants in terms of job, we did not take experience into account in the evaluation of our study.

Job	Number
Student	15
Employed in a private enterprise	3
Employed in university	3
Other	1

Table 4.3: Summary Job

4.2 Design

This study has a within-subject design (every participant sees every code and every indentation). This is supposed to even out individual attributes such as reading speed. To balance the impact of confounding factors, such as personal skill, we randomized the code-piece order. Sorting four codes into chunks of four without repetition results $4 \times 3 \times 2 \times 1 = 24$ possible combinations. Additionally, the four levels of indentation were randomly assigned. Here, the number of possible sequences of the four levels without repetition is again 24. This results in a total of $24 \times 24 = 576$ possible trial sequences, where no code and no indentation is repeated This should counter interactions between code and indentation, as it might be that, for example, one code is the easiest to understand with an indentation of eight spaces and using only this version might lead to the false conclusion that eight space indentation is preferable independently of the code. These steps should also prevent learning effects.

4.2.1 Independent Variable

Indentation is the separation of a code block by shifting its lines to the right. We manipulated this factor in order to measure its effects on code comprehension and the other dependent variables.

It had four different levels: zero, two, four and eight spaces. Code blocks that belong to the following constructs were indented:

- Class header
- Method header
- Loop header
- *If*, *else if* or *else*

In doing so, lines of the same depth of embedding were equally indented.

4.2.2 Dependent Variables

This section presents the dependent variables and how they were measured during the study. It also gives an explanation, why the according ways of measurement were used.

4.2.2.1 Program comprehension

A top-down approach to measure comprehension would ask the participants about the function of the program. In contrast, making them perform mental execution is asking for a bottom-up strategy. To find the correct answer, one is not required to (but still can) understand the code's overall purpose. If the code or rather its function is not known to the reader he is forced to go through every line and compute the values of attributes and the output in his/her mind. We therefore chose this way of measuring program comprehension. Other questions were discarded in order to keep the study easy to survey, but also due to the limited complexity of the codes. An answer was only counted as correct, when it matched exactly with the actual output.
Response time was measured for each task, from the time the stimulus (the code) was presented, until the participants completed the task.

4.2.2.2 Subjective Assessment

In order to get the participants subjective assessment of the difficulty of the codes used in the tasks, they were asked to order the snippets from easy to hard in comparison with each other. This method was chosen, because it forces the rater to compare the codes directly with each other and not with other codes they have seen or worked with. If, for example, a programming expert is used to much more complex code, he/she still has to decide which code is the easiest one, even though all presented codes are rather 'easy' for him/her. The resulting rankings are thus not absolute, but more comparable than by an individual assessment of difficulty,

for example via a Likert-scale. There comparing difficulties may be problematic, if all codes have approximately the same difficulty for the rater and he is not taking subtle differences into account.
To help the participants remember the codes, they were asked to rate, the snippets were displayed next to each other along with the respective answers given in the preceding trials. The participants had to perform the sequencing twice: First, the codes are displayed with four-spaces of indentation. This was supposed to get the subjective assessment of difficulty without giving hints about the role of indentation in the study. As the codes are the same as in the task part of the study, the conscious or unconscious memory of the level of indentation could influence the decision. Additionally, the codes are displayed with the actual indentation as used earlier in the tasks. Here, participants had the chance to take the depth of indentation into account for the rating.

4.2.2.3 Visual Effort

For measuring fixation duration, fixation rate and saccadic amplitude the participants' gaze points were logged during the trials. We used a remote eye tracker that was attached to the screen for collecting the gaze position in an unobtrusively way.

4.2.3 Differences to original Study

Although this study aimed to replicate the findings of [MMNS83], their original design was changed to a great extend. Eye tracking was not part of the original study, hence all questions and measurements regarding the participants' gaze were newly introduced. Another modification was the transition from the between-subject design to the within-subject design. The latter allows a smaller group of participants be used (although the number of participants should of course always be as high as possible). The examined programming language was also altered: Java was preferred over Pascal, as Java is a more modern language and has been one of the most widely deployed languages since 2001 [TIO] . The within-subject design, the desired shortness of the codes and the different language resulted in four new code snippets not related to the original Pascal code. The examined levels of indentation were also different: Originally, the focus lied on indentations of zero, two, four, and six spaces. We decided to replace the six-space indentation by one with eight spaces, in order to amplify the effects that a high level of indentation could have. Also, the evaluation of program comprehension differs: While Shneiderman asked the participants multiple questions about the code's function, here, we asked one question about the output of the code. In contrast to the original study, where the processing time was fixed, this study recorded additionally the response time. Last but not least, a distinction of novices and experts was omitted here, because a) the group of participants was substantially smaller, and b), it was expected to be rather homogeneous.

4.3 Material

The snippets shown in the study were selected under the aspects of being equal enough to be comparable, but at the same time being different enough for not support learning effects. Each of them treats a problem involving an input string and its splitting into subparts. The codes do not solve standard text book problems and are inspired by the codes used in [HSH17]. They all contained two block structures (i.e., if-else-block or loops). A small pilot study was conducted to get an estimate for the average trial time and for the difficulty of the snippets. As a result, one code was replaced.

```java
public class Do {
    public static void main(String[] args) {
        int[] array = { 5, 6, 11, 0, 2 };
        int integer = 0;
        int number1 = 0;
        int numberA = 0;
        while (integer < array.length) {
            if (array[integer] % 2 == 0) {
                number1 += array[integer];
            } else {
                numberA += array[integer];
            }
            integer++;
        }
        System.out.print(numberA - number1);
    }
}
```

(a) Do

```java
public class Main {
    public static void main(String[] args) {
        int[] values = { 3, 0, 1, 0, 2 };
        StringBuilder result = new StringBuilder();
        int variable = 3;
        for (int value : values) {
            if (value == variable) {
                result.append("x");
            } else if (value < variable) {
                result.append("m");
            }
            result.append("o");
            variable--;
        }
        System.out.print(result);
    }
}
```

(b) Main

```java
public class Program {
    public static void main(String[] args) {
        String input = "1-3,10-11";
        int output = 0;
        for (String part : input.split(",")) {
            String[] numbers = part.split("-");
            int left = Integer.parseInt(numbers[0]);
            int right = Integer.parseInt(numbers[1]);
            int number = left;
            while (number <= right) {
                output += number;
                number++;
            }
        }
        System.out.println(output);
    }
}
```

(c) Program

```java
public class Test {
    public static void main(String[] args) {
        int variable = 0;
        String string = "3 21 4 2 55 0 13";
        int start = 2;
        int end = 18;
        String[] keys = string.split(" ");
        for (int i = 0; i < keys.length; i++) {
            int key = Integer.parseInt(keys[i]);
            boolean check = (key >= start && key <= end);
            if (check) {
                variable += 1;
            }
        }
        System.out.print(variable);
    }
}
```

(d) Test

Figure 4.1: The Four Snippets

All snippets have 17 lines of code. This was supposed to ensure that the codes are non-trivial and that the eye-tracker was able to differentiate the points of gaze. The classes are named deliberately vague ('Do', 'Test','Program','Main') and have a single method, *main*. This method contains at least one print-statement, which had to be considered in determining the output. Identifier names are chosen so that they would not particularly give any hints about their function in the code (e.g. 'number1', 'variable'). In order to normalize the snippets, there were no blank lines in the code. Each snippet was recorded from the standard Eclipse environment with the syntax highlighting of the default theme and the auto-formatter of Eclipse was used (except for indentation). Figure 4.1 shows the snippets with an indentation of four spaces and syntax highlighting. The processing time for one code was restricted to 5 minutes, due to experiences made in the pilot study.

```
while (integer < array.length) {
if (array[integer] % 2 == 0) {
number1 += array[integer];
} else {
numberA += array[integer];
}
integer++;
}
```

(a) 0-space Indentation

```
while (integer < array.length) {
  if (array[integer] % 2 == 0) {
    number1 += array[integer];
  } else {
    numberA += array[integer];
  }
  integer++;
}
```

(b) 2-space Indentation

```
while (integer < array.length) {
    if (array[integer] % 2 == 0) {
        number1 += array[integer];
    } else {
        numberA += array[integer];
    }
    integer++;
}
```

(c) 4-space Indentation

```
while (integer < array.length) {
        if (array[integer] % 2 == 0) {
                number1 += array[integer];
        } else {
                numberA += array[integer];
        }
        integer++;
}
```

(d) 8-space Indentation

Figure 4.2: The Four Levels of Indentation

The codes are treated with four different levels of indentation. To get a sense for the differences, an example of the four levels is shown in Figure 4.2.
A warm up code was shown before the actual beginning of the study to get the participants accustomed to the task and the interface. This code had all the attributes of the task snippets (non-helpful identifiers, class with main method) expect for having only one block construct (if-else-block). Its simple structure was also meant as a test to check for those, who did not participate seriously. This snippet had an indentation of 4 spaces as this is the recommended level of indentation in the Oracle Java coding conventions [SM]. It is shown in Figure 4.3.

```
public class WarmUp {
    public static void main(String[] args) {
        int test = 3;
        if (test > 3) {
            System.out.print("k");
        } else {
            System.out.print("r");
        }
    }
}
```

Figure 4.3: Warm Up Code

4.4 Procedure

Before the actual study started, the participants took place in front of the screen and were asked to sit down as if they would do when they had to use the keyboard and mouse. In order to obtain good gaze data, they were then asked to change their position until they felt both comfortable and the Tobii Tracking Software showed that their eye movements were detectable in such a way that a recording seemed possible. Since the experimenter stayed in the room, they were told to ignore her as far as possible. Then, the study software was executed. It was a self-written .NET- program, which uses the WPF-Framework and handles the logging of the gaze data and the participants' input. The language used in the instructions was German. In the following, the flow of the actual study will be explained with the help of Figure 4.4:
The study started with a welcome screen, which was also where the participants gave their informed consent [*Welcome screens*]. They were then asked general personal questions [*Personal questions*] and about their programming experience [*Prog. questions*]. When they declared to be students in the personal questions, they were directed to answer questions about their studies [*Student questions*], before getting to the questions regarding programming experience that all participants answered. Afterwards, the eye-tracker was calibrated [*Calibration*]. The participants were asked to remain in their current position as far as possible to keep the calibration valid. Subsequently, they were given the instructions for the tasks [*Instructions*], including the warm-up snippet [*Warm Up*], before which an fixation cross already appeared [*Fixation Cross*]. Then the actual tasks [*Task*] started. The snippets were preceded by a fixation cross [*Fixation Cross*] of 1.5 seconds. The following task screen showed the stimulus code in the main part of the window. Separated by a thin line, the input field for the output answer was at the bottom part of the screen. The question to be answered ('What output does this code produce?') was written to the left side of this field. With a click on 'Done', the participants got to the next screen. The time limit was enforced in a way that after four minutes, a message dialog appeared, warning the readers that only one minute is left. If the time ran out eventually, another message box popped up, stating that the time was over and the window switched to the next screen. In this case, the non-existing answer was set to a default wrong value. After each snippet, they had the chance to decide for themselves when to continue [*Relaxation*]. After the trial phase, the participants had to assess the difficulty of the code snippets twice. The snippets were presented to them together with the answers the particular participant had given. Via drag-and-drop, participants could order the snippets after increasing difficulty. First, the snippets were displayed with a four-space indentation [*Rating Equal*]. The second time, the snippets had the same indentation as they were shown to the participant earlier [*Rating Real*]. After the main part, the participants were asked, whether they participated seriously and whether they got distracted during the trials [*Final Questions*]. Finally, they were asked, whether they wanted to take part in the lottery and be informed about the results [*Contact data*]. This was the end of the study [*End*].
Each trial took maximally 30 minutes (up to 20 minutes for the code tasks plus ~10 minutes to answer the questionnaire, read the instructions and do the warm-up.)

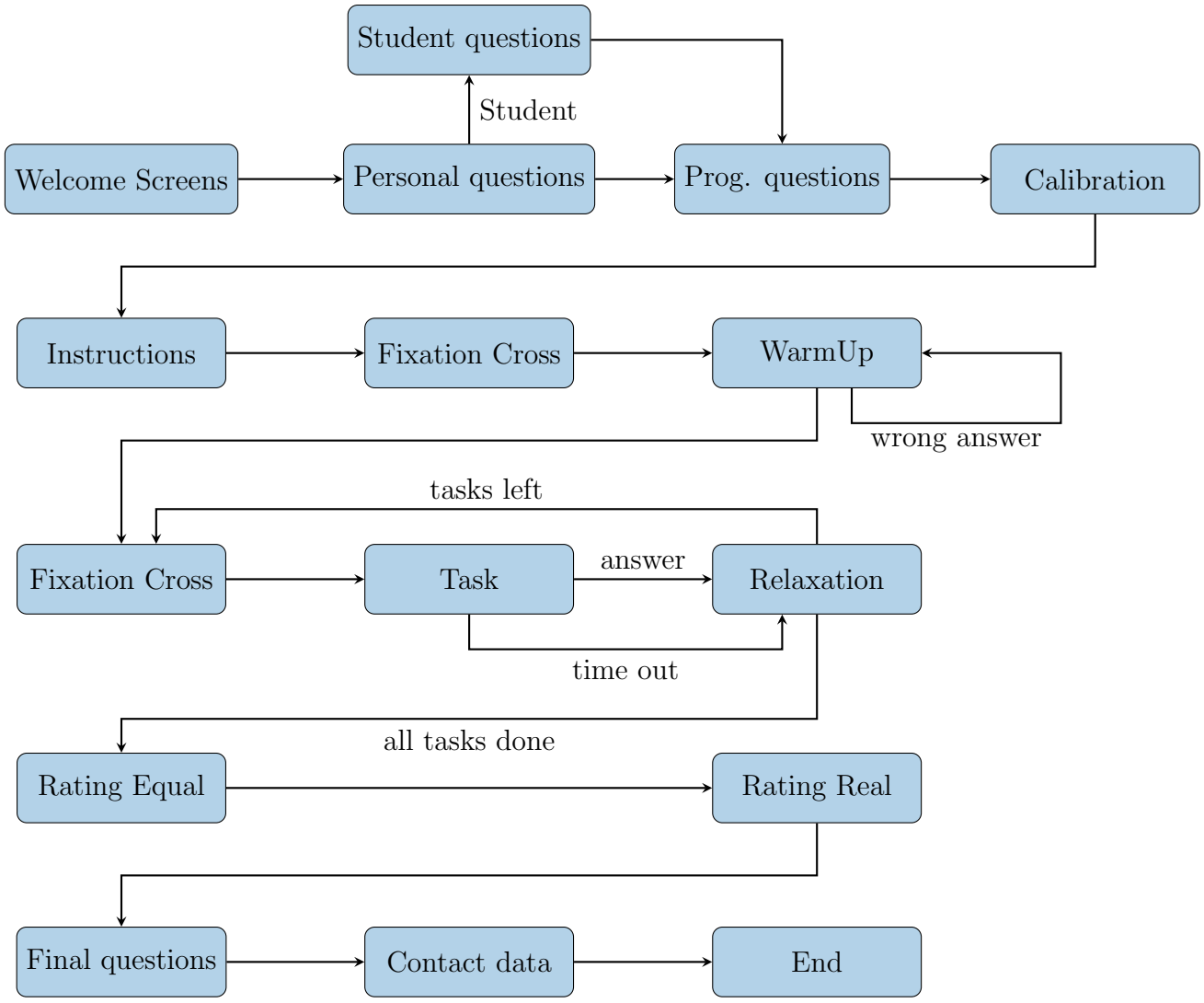


Figure 4.4: Flow Diagram of the Study

Execution

This study used the Tobii EyeX tracker with the "Tobii Eye Tracking Core Software" in the version 2.9.0. The tracker has a sampling rate of 60 Hz or more and an operating range of 50-90 cm. It is suitable for a display size up to 27" [Tob]. The used monitor's screen had a diagonal of 24" and a resolution of 1920x1080 pixels with a refresh rate of 60 Hz. It was placed ~50 cm away from the desk's edge and ~ 19 cm above the desk's level. The participants had to use a standard keyboard with the German QWERTZ layout and a wireless mouse.
The study was conducted in an office with two desks facing each other. On one the monitor and keyboard for the participants was assembled, the other one was empty. The experimenter took place on this desk. The curtains and roller shutters were closed as far as possible and the light in the room was turned on. This was done in order to get better results from the eye tracker. During the conduct of the study, a construction site outside the building repeatedly caused a higher noise level. The participants stated in the questionnaire that they have not been disturbed during the tasks.

5 Results

In this section, the results gathered from the execution of the study are presented, followed by the analysis of the influence of indentation on the depending variables.

5.1 Summary of data

This section gives an overview about the values of the measured dependent variables and their distributions. The data preparation is briefly described. The summary of values in the tables show the mean and standard deviation of the values, if the data is normally distributed. Otherwise, the median and the interquartile range (IQR) are listed.

5.1.1 Correctness and Time

The distribution and the summary of the response times can be taken from Figure 5.1 and Table 5.1. Table 5.1 also contains the percentage of right answers given. The time values are not normally distributed, but skewed to the left. In order to minimize the effect of outliers, data can be transformed via a function, for example by inversion or applying the log-function [Rat93]. Here, the response times were log-transformed, because this "tends to normalize the distributions" [Rat93]. This is indeed the case, shown by a p-value of 0.1 for the Shapiro-Wilk normality test [SW65]. Insignificance of this test implies that the tested distribution is most likely not to be normally distributed. The normality of distribution is a prerequisite for the ANOVA later in this chapter.

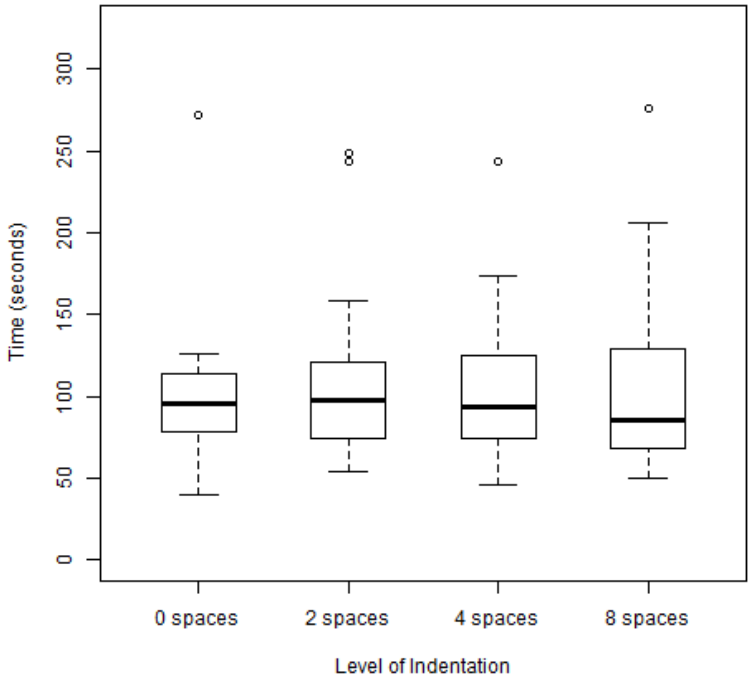


Figure 5.1: Distributions of Response Times

Time (mm:ss)			
Indentation	**Median**	**IQR**	**% right answers**
0	01:35.001	00:34.773	73 %
2	01:37.547	00:41.071	68 %
4	01:33.422	00:47.791	77 %
8	01:25.078	00:59.024	55 %
Total	01:33.983	00:47.171	68 %

Table 5.1: Summary of Response Times and Correctness

5.1.2 Rating

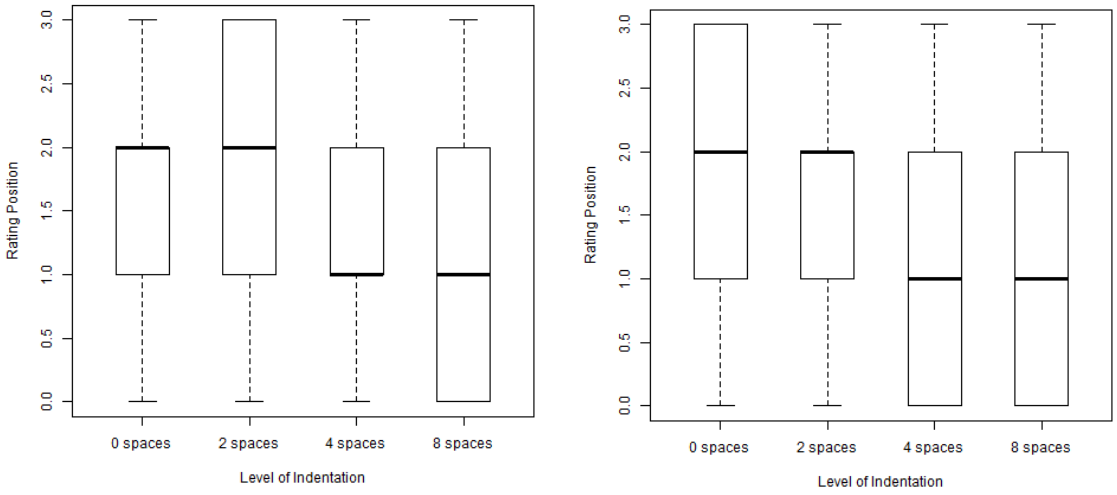
Table 5.2 shows a summary of the rating positions for the respective level of indentation the rated code had, when the rating took place with equal indentations. Table 5.3 contains the values for the codes rated with the actual indentation of the trials. The medians of the two conditions are equal, only the distributions differ for the four levels of indentation.

	Rating Position	
Indentation	**Median**	**IQR**
0	2	1
2	2	1.75
4	1	0.75
8	1	2

Table 5.2: Summary of Rating Position with Equal Indentations

	Rating Position	
Indentation	**Median**	**IQR**
0	2	2
2	2	1
4	1	2
8	1	2

Table 5.3: Summary of Rating Position with Real Indentations



(a) Equal Indentations shown

(b) Real Indentations shown

Figure 5.2: Distribution of Rating Position

5.1.3 Fixation Duration

The median fixation duration of each trial is approximately normally distributed, as the p-value of the Shapiro-Wilk normality test is about 0.18. It is summarized in Table 5.4 and Figure 5.3. Instead of the mean of fixation durations, the median is chosen, because the fixation durations are log-normally distributed [Lug07] and therefore more skewed to the left.

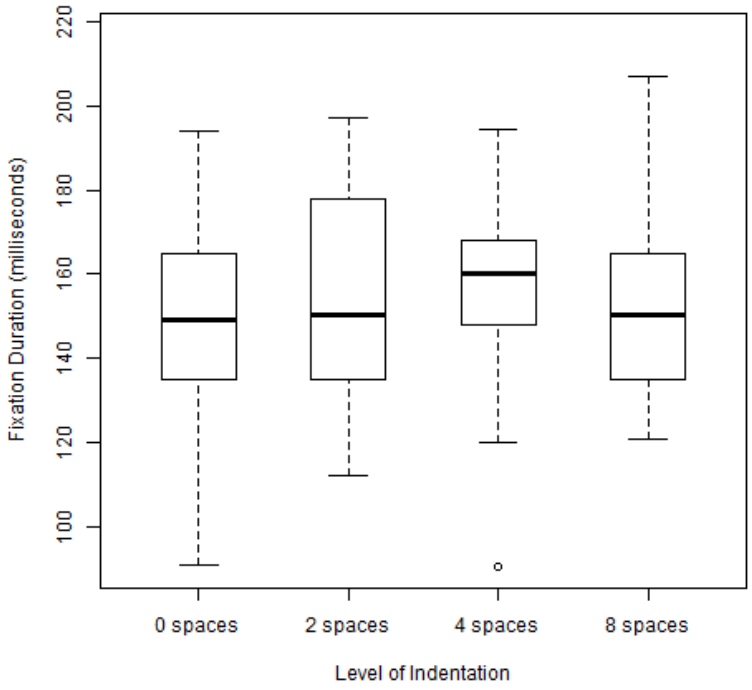


Figure 5.3: Distribution of Median Fixation Duration

	Median Fixation Duration per Trial (ms)	
Indentation	**Mean**	**Standard Deviation**
0	146.09	25.94
2	153.75	24.45
4	157.41	25.83
8	153.50	22.04
Total	152.69	24.54

Table 5.4: Summary of Median Fixation Duration per Trial

5.1.4 Fixation Rate

The fixation rate again is not normally distributed, but this time skewed to the right with a long tail on the left side. It is summarized in Figure 5.4 and Table 5.5

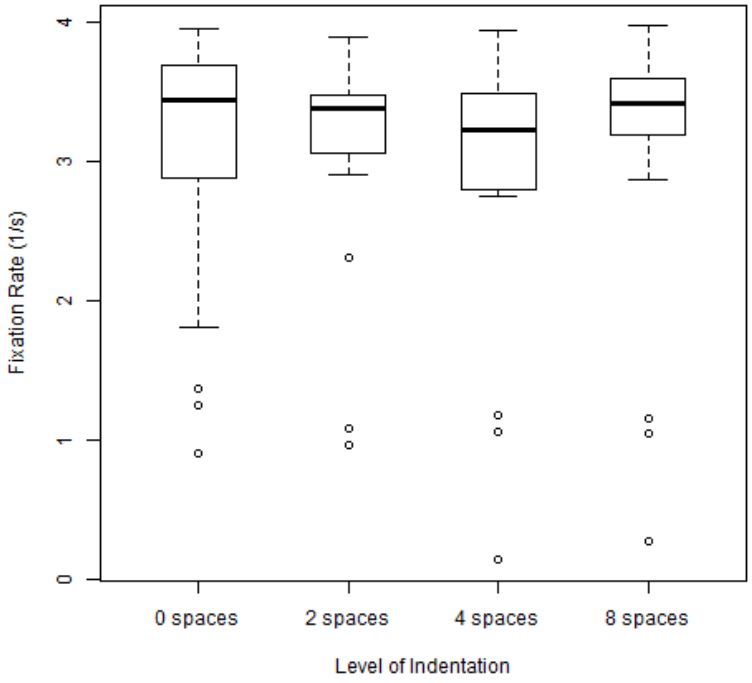


Figure 5.4: Distribution of Fixation Rate

	Fixation Rate (fixations/second)	
Indentation	**Median**	**IQR**
0	3.45	0.81
2	3.38	0.39
4	3.23	0.67
8	3.42	0.38
Total	3.35	0.69

Table 5.5: Summary of Fixation Rate

5.1.5 Saccadic Amplitude

The average saccadic amplitudes per trial are not normally distributed. They are skewed to the left and thus were log-transformed for later analysis (similar to response times). The transformed values are now normally distributed according to the Shapiro-Wilk test ($p = 0.11$). The original values are summarized in Figure 5.5 and in Table 5.6

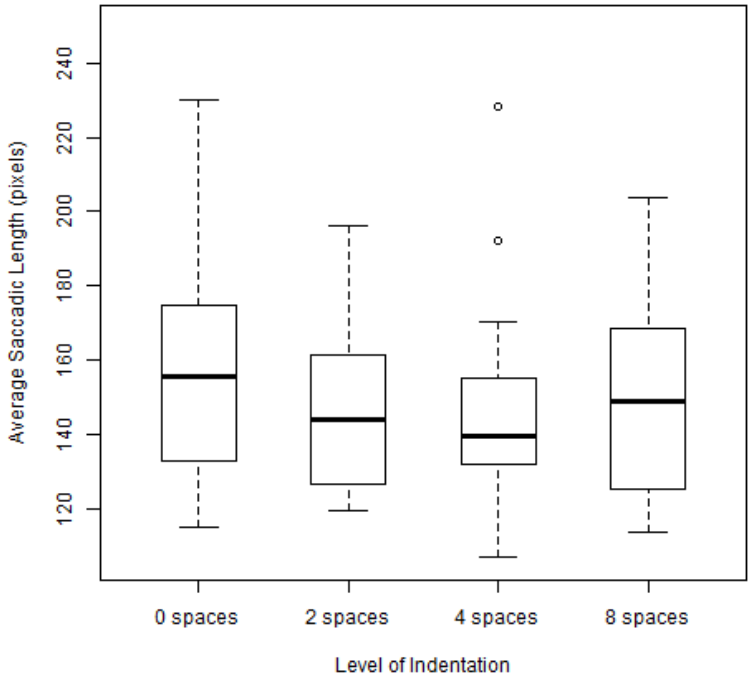


Figure 5.5: Distribution of Saccadic Amplitude

	Average Saccadic Amplitude (pixels)	
Indentation	**Median**	**IQR**
0	155.53	41.36
2	144.11	33.23
4	139.48	22.58
8	149.09	40.42
Total	145.24	32.91

Table 5.6: Summary of Saccadic Amplitude

5.2 Statistical analysis

In the following, the influence of indentation on the dependent variables is statistically analyzed, divided into the respective research questions. Before, the statistical tests used in the analysis are briefly explained.

5.2.1 Test description

We used the two tests below for statistical significance. In which cases they were applied, is described in Section 5.2.2.

One-way ANOVA with repeated measures This test is used for comparing means of groups that are differentiated by one factor (here: level of indentation), but were collected from the same source [Zü]. It requires that the depended variable is interval-scaled and normally distributed, and that sphericity can be assumed (for example by the significance of the Mauchly test [Mau40]).

Friedman test This test is used, when the data is not normally distributed or not interval-scaled, and hence a One-Way ANOVA is not applicable [Fri37]. By ranking the values, it analyzes the variance of repeated measures derived from one independent variable. It requires that the values between samples are paired and independent within a sample.

5.2.2 Test results

This section lists the results of the test of significance for each research question together with a justification for the choice of test.

$\mathbf{RQ_{Correctness}}$
The Friedman test showed that the number of correct answers was not significantly affected by the levels of indentations $\chi^2(3) = 3.32, p = 0.36$.
This test was used, because the dependent variable ('Number of correct answers') was not interval-scaled.

$\mathbf{RQ_{Time}}$
A One-way ANOVA with repeated measures showed that the effect of level of indentation on the log-transformed response times was not significant, $F(3, 63) = 0.44, p = 0.72, n_p^2 = 0.028$.
This test was selected since the dependent variable ('Time') is interval-scaled and is also normally distributed after the log-transformation. Sphericity was assumed, $Mauchly - W(3) = 0.85, p = 0.65$

$\mathbf{RQ_{Rating}}$
For determining whether the rating positions of the codes differed depending on the level of indentation the Friedman-Test was applied, once for the rating with equal indentations, and once for the one with the actual indentations. For both, the differences are not significant, $\chi^2(3) = 4.64, p = 0.20$ and $\chi^2(3) = 5.35, p = 0.15$. Since the dependent variable ('Rating Position') is ordinal-scaled, the Friedman test was chosen.

$RQ_{Fix:Duration}$

A One-way ANOVA with repeated measures showed that the effect of level of indentation on median fixation duration per trial was significant, $F(3, 63) = 2.85, p = 0.045, n_p^2 = 0.006$. This test was chosen, because the dependent variable ('Median Fixation Duration Per Trial') is interval-scaled and normally distributed. Sphericity was assumed, $Mauchly - W(3) = 0.77, p = 0.39$

$RQ_{Fix:Rate}$

There was no significant difference among the distributions of the fixation rates for the four levels of indentation according to the Friedman test, $\chi^2(3) = 7.36, p = 0.06$.

As the dependent variable ('Fixation Rate') is not normally distributed, the Friedman test was chosen instead of an ANOVA.

$RQ_{Sacc:\ Amplitude}$

A One-way ANOVA with repeated measures showed that the effect of level of indentation on the log-transformed average saccadic amplitude was not significant, $F(3, 63) = 1.69, p = 0.18, n_p^2 = 0.029$ Sphericity was assumed, $Mauchly - W(3) = 0.87, p = 0.74$.

6 Discussion

In summary, code indented with four spaces resulted in the highest number of right answers given, followed by zero, two and eight spaces. In contrast, codes with an indentation of eight spaces were processed the fastest and two-spaced indented codes the slowest. Codes that were originally indented with eight spaces, were also rated as the most easiest, when presented with equal indentations for all codes, and second easiest, when shown with the original indentation. The order from easiest to hardest code changed from indentation of eight, four, zero and two spaces, when presented with equal levels of indentation, to four, eight, two and zero spaces, when presented with the original indentation. The codes with the highest fixation duration were the ones of four-space indentation, while the ones of two- and eight-space indentation were both approximately the second highest. This order was reversed, when focusing on fixation rate, but with codes with two-space indentation leading to a slightly higher fixation rate than codes with eight-space indentation. The saccadic amplitude was ordered similarly to the fixation rate.

These differences of the individual measurements were not significant, except for fixation duration. We can therefore neither support nor disagree with the level of indentation recommended in coding conventions based on our study.
The following section discusses the possible reasons for the insignificance, divided into the topics of the research questions.

6.1 Program Comprehension and Times

The level of indentation showed no significant effect on correctness, nor on response times. This was surprising, as non-indented code in particular was expected to influence at least response times. One reason for the insignificance might be that the effect size of the influence of indentation is probably very small, because there are other factors that presumably have a greater impact on program comprehension (i.e. the calculation performed by the code). The influence of indentation could have been increased by carefully balancing those side effects. Finding significant

differences for small effects also demands a high number of participants, which was not given in this study. Finally, it is also possible that the advantage of indentation comes more into effect, when the code is longer and more complex. Then, dividing the code into coherent parts can become more important, in order to understand the code's function as a whole. A larger depth of embedding (such as 4 or 5 nested loops have) may also increase the need for sensible indentation.

6.2 Subjective Assessment

The subjective assessment was not significantly affected by the level of indentation the rated codes had in the previous trials. This was found when the codes were presented with an equal level of indentation, as well as with their actual indentation from the tasks. Although it was expected that the participants were especially sceptical about the unusual non-indented code, their classification of difficulty was not significantly influenced by indentation. It is possible that they were mainly focused on the calculation performed by the codes instead of style. In contrast, the codes could not have been complex or long enough in order for the participants to feel that indentation supports the understanding of the code's function.

6.3 Visual Effort

Although the differences in median fixation duration between the different levels of indentation were significant, this result has to be handled with caution: The effect size is extremely small, and due to the numerous threats to validity the findings for this effect are negligible. Additionally, the tests for fixation and saccadic rate showed no significant effect of indentation on those measures. The absence of evidence of a correlation between indentation and visual effort could originate from the high subjective factor of eye-tracking data, "meaning that one person's parameters are different from another person's, irrespective of task" [HNA+11]. The small number of participants might (next to other aspects) thus hinder the finding of significant results for differences in visual effort. Furthermore, the data obtained by the eye-tracker may be too inaccurate to find significant effects in gaze behavior with the materials and setting of this study. As there are no other studies specifically examining the effect of indentation on visual effort that we are aware of, it could be possible that this effect is non-existent. However, we assume this to be unlikely, because the level of indentation changes the spatial arrangement of the code. Hence, it probably affects the process of reading code. Similar to program comprehension, this effect could even be increased, when the code is longer and alone the number of indentations is greater.

6.4 Differences to original study

In contrast to the study by Shneiderman, this study found no significant effect of indentation on program comprehension. This may be due to the various changes to the original design. We suspect that the length and complexity of code have a huge impact on the support that indentation possibly offers for comprehension. We used much shorter codes with decreased complexity in comparison to the program

used by Shneiderman, possibly resulting in smaller effects. They also measured comprehension with a wider range of questions, whereas we solely focused on mental execution. It may be that other questions capture the effects of indentation on comprehension better. Aside of design and material, one of the greatest differences is the number of participants: While Shneiderman evaluated 86 participants, we statistically tested only 22 and had therefore a reduced chance of finding significant effects.

7 Threats to Validity

There are numerous threats to the validity of the results of this study. Section 7.1 explains, which error sources could have compromised the influence of indentation on the dependent variables. Section 7.2 accounts for problems with the measurements chosen for answering the research questions, and Section 7.3 discusses the generalizability of the findings in this study to the real world.

7.1 Internal Validity

While all participants declared that they were not distracted during the tasks, the construction noise as a prime example for a confounding factor and other sounds from the corridor and the surrounding offices might have had an influence on the concentration of the participants. Of course, the sheer presence of the tracker may have impacted the participants' behavior. Although a small pilot study was conducted to discard snippets, for which participants needed more time for comprehension, it could be that differences between the finally selected snippets had a significant effect on the response time. They are also not completely normalized: the number of variables, the number of statements, identifier names, and many more attributes differ.
This also has an impact on the eye tracking data, as the different layouts of the codes even with the same levels of indentation automatically result in different gaze patterns and attributes.

7.2 Construct Validity

Measuring code comprehension only by mental execution tasks does not take all aspects of comprehension into account. It is not suitable to measure the participants' overall understanding of the code. The problem handled by a program does not have to be extracted in order to calculate the right output. However, if a participant understood the code's semantics, he/she is likely to give the correct answer regarding its output. The tasks in this study therefore only measured one aspect of program comprehension.
Rating the codes' difficulty by sequencing them does not depict absolute values. Participants were not able to adequately express their subjective assessment, when they thought all codes to be equally easy or difficult. Nevertheless, the relative differences can be measured this way.
The gaze attributes examined in this study are only a subset of possible aspects for measuring visual effort. It may be that fixation duration, fixation rate and saccadic amplitude are less informative about visual effort, than ,for example, blink rate or pupil diameter.

7.3 External Validity

The number of participants was very small and they were also very homogeneous group (over half of them were students). Thus, the lack of evidence for effects in this study cannot be transferred to the generality of programmers.
The snippets used in the tasks were rather short (17 lines of code) as they had to be of appropriate size for the eye tracker. Thus, and due the unnatural naming used, the codes do not represent everyday code.

8 Conclusion

Encouraged by the findings of Shneiderman, and the visual influence of indentation on the layout of code, we estimated that indentation affects program comprehension, subjective assessment and visual effort. We conducted an empirical study, in which participants had to calculate the output of Java code and rate its difficulties while their gaze was tracked. Contrary to our expectations, we found no significant effects The question of this thesis, *"Are indentations a simple matter of style or do they support code comprehension?"*, can therefore not be answered by our study and we are not able recommend a specific level of indentation.
With the following ideas, the methods and materials we used could be further refined to address the issues raised in the discussion.

Future Work

In accordance with the findings of Shneiderman, we expect that a replication of this study with a higher number of participants could show a tendency towards moderate levels of indentation as recommendable for program comprehension. The usage of larger code snippets could amplify the effect of indentation, as the differentiation between the logical units of code becomes more important. In addition, a more precise eye tracker would be needed, if measuring visual effort is desired. Dividing the stimulus in areas of interest and calculating the number of fixations for them, could shed more light on how gaze behavior is affected by the level of indentation. Further research of this aspect could for example investigate, whether participants switch more between nesting levels, when faced with different indentations. In contrast, completely omitting the eye tracker open up new ways to use longer and more complex codes. Without an eye tracker, a laboratory set up becomes less important, which would make it possible to conduct a replication online.
An additional factor worth looking at might be experience. It presumably plays an elementary role when it comes to structuring processes, problem-solving, and debugging. Experience could also help, when encountering unstructured or unusually formatted code.

A Appendix

A.1 Codes

```
public class WarmUp {
        public static void main(String[] args) {
                int test = 3;
                if (test > 3) {
                        System.out.print("k");
                } else {
                        System.out.print("r");
                }
        }
}
```

Listing A.1: Warm up code

```
public class Do {
    public static void main(String[] args) {
        int[] array = { 5, 6, 11, 0, 2 };
        int integer = 0;
        int number1 = 0;
        int numberA = 0;
        while (integer < array.length) {
            if (array[integer] % 2 == 0) {
                number1 += array[integer];
            } else {
                numberA += array[integer];
            }
            integer++;
        }
        System.out.print(numberA - number1);
    }
}
```

Listing A.2: Code 'Do'

```
public class Main {
    public static void main(String[] args) {
        int[] values = { 3, 0, 1, 0, 2 };
        StringBuilder result = new StringBuilder();
        int variable = 3;
        for (int value : values) {
            if (value == variable) {
                result.append("x");
            } else if (value < variable) {
                result.append("m");
            }
            result.append("o");
            variable--;
        }
        System.out.print(result);
    }
}
```

Listing A.3: Code 'Main'

```
public class Program {
    public static void main(String[] args) {
        String input = "1-3,10-11";
        int output = 0;
        for (String part : input.split(",")) {
            String[] numbers = part.split("-");
            int left = Integer.parseInt(numbers[0]);
            int right = Integer.parseInt(numbers[1]);
            int number = left;
            while (number <= right) {
                output += number;
                number++;
            }
        }
        System.out.println(output);
    }
}
```

Listing A.4: Code 'Program'

```
public class Test {
    public static void main(String[] args) {
        int variable = 0;
        String string = "3 21 4 2 55 0 13";
        int start = 2;
        int end = 18;
        String[] keys = string.split(" ");
        for (int i = 0; i < keys.length; i++) {
            int key = Integer.parseInt(keys[i]);
            boolean check = (key >= start && key <= end);
            if (check) {
                variable += 1;
            }
        }
        System.out.print(variable);
    }
}
```

Listing A.5: Code 'Test'

A.2 Questionnaire

Question	Answer Options
In which year were you born?	Integer
What is your gender?	Man, Woman, Prefer not to say
What is your main job?	Student, Employed in a private enterprise, Employed in university (no student assistant), Entrepreneur, No job, Other
What is your highest degree?	Abitur / vocational baccalaureate, Apprenticeship / technical training, Bachelor, Master / Diploma, Lateral entrant / no degree, Other degree
Are you wearing glasses?	Yes, No
Do you have a visual disorder or misaligned eyes?	Yes, No

Table A.1: General Questions

Question	Answer Options
What is your semester in your field of study?	Integer
Do you work besides studying the field of informatics?	No, Student assistant, Working student, Intern, Entrepreneur, Employee, Other
How many hours per week do you spend on own projects?	Integer

Table A.2: Student Questions

Question	Answer Options
How many years of experience with Java do you have?	Integer
How many years of overall programming experience do you have?	Integer
How familiar are you with Java?	No Experience, Basic Knowledge, Project Experience, Regular, Expert
Optional: Which other programming languages do you know (Basic Knowledge or more)?	Free Text
How many lines of code does the biggest project you ever worked on have?	Free Text
How many lines of code does the biggest project you created in a team have?	Free Text
How many lines of code does the biggest project you created on your own have?	Free Text
When did you last program something in Java?	This week, Last Month, Last Year, Longer than 1 year ago, Longer than 5 years ago

Table A.3: Programming Questions

Question	Answer Options
Did you take the study seriously?	Yes, No
Were you distracted during the study?	Yes, No
Did you have fun doing the study?	Yes, No

Table A.4: Final Questions about Study

Question	Answer Options
Do want the collective results to be send to you?	Yes, No
Do you want to take part in the lottery of the Amazon gift card?	Yes, No
Do you want to get informed about other studies?	Yes, No

Table A.5: Contact questions

Bibliography

[BDLM09] Dave Binkley, Marcia Davis, Dawn Lawrie, and Christopher Morrell. To camelcase or under_score. In *Program Comprehension, 2009. ICPC'09. IEEE 17th International Conference on*, pages 158–167. IEEE, 2009. (cited on Page 4)

[Foua] Python Software Foundation. The python language reference. Website. Available online at https://docs.python.org/3/reference/lexical_analysis.html; visited on October 8th, 2017. (cited on Page 1)

[Foub] Python Software Foundation. Style guide for python code. Website. Available online at https://www.python.org/dev/peps/pep-0008/; visited on October 9th, 2017. (cited on Page 4)

[Fri37] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937. (cited on Page 25)

[Gooa] Google. Google c++ style guide. Website. Available online at https://google.github.io/styleguide/cppguide.html; visited on October 9th, 2017. (cited on Page 4)

[Goob] Google. Google java style guide. Website. Available online at https://google.github.io/styleguide/javaguide.html; visited on September 18th, 2017. (cited on Page 2)

[Gooc] Google. Google javascript style guide. Website. Available online at https://google.github.io/styleguide/jsguide.html; visited on October 9th, 2017. (cited on Page 4)

[HNA+11] Kenneth Holmqvist, Marcus Nyström, Richard Andersson, Richard Dewhurst, Halszka Jarodzka, and Joost Van de Weijer. *Eye tracking: A comprehensive guide to methods and measures*. OUP Oxford, 2011. (cited on Page 5 and 28)

[HSH17] Johannes Hofmeister, Janet Siegmund, and Daniel V Holt. Shorter identifier names take longer to comprehend. In *Software Analysis, Evolution and Reengineering (SANER), 2017 IEEE 24th International Conference on*, pages 217–227. IEEE, 2017. (cited on Page 13)

[JC80] Marcel A Just and Patricia A Carpenter. A theory of reading: From eye fixations to comprehension. *Psychological review*, 87(4):329, 1980. (cited on Page 5)

[KUMA+84] Thomas E Kesler, Randy B Uram, Ferial Magareh-Abed, Ann Fritzsche, Carl Amport, and Hubert E. Dunsmore. The effect of indentation on program comprehension. *International Journal of Man-Machine Studies*, 21(5):415–428, 1984. (cited on Page 2)

[Lug07] AJP Lugtigheid. *Distributions of fixation durations and visual acquisition rates*. PhD thesis, Erasmus University, 2007. (cited on Page 21)

[Mau40] John W Mauchly. Significance test for sphericity of a normal n-variate distribution. *The Annals of Mathematical Statistics*, 11(2):204–209, 1940. (cited on Page 25)

[Mic] Microsoft. C-sharp coding conventions. Website. Available online at https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions; visited on October 9th, 2017. (cited on Page 4)

[MKW+90] James G May, Robert S Kennedy, Mary C Williams, William P Dunlap, and Julie R Brannan. Eye movement indices of mental workload. *Acta psychologica*, 75(1):75–89, 1990. (cited on Page 5)

[MMNS83] Richard J Miara, Joyce A Musselman, Juan A Navarro, and Ben Shneiderman. Program indentation and comprehensibility. *Communications of the ACM*, 26(11):861–867, 1983. (cited on Page 2, 7, 8, and 12)

[NTS02] Minoru Nakayama, Koji Takahashi, and Yasutaka Shimizu. The act of task difficulty and eye-movement frequency for the'oculo-motor indices'. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 37–42. ACM, 2002. (cited on Page 5)

[PE08] Matthew H Phillips and Jay A Edelman. The dependence of visual scanning performance on search direction and difficulty. *Vision research*, 48(21):2184–2192, 2008. (cited on Page 5)

[Rat93] Roger Ratcliff. Methods for dealing with reaction time outliers. *Psychological bulletin*, 114(3):510, 1993. (cited on Page 19)

[SM] Inc. Sun Microsystems. Code conventions for the java programming language. Website. Available online at http://www.oracle.com/technetwork/java/codeconvtoc-136057.html; visited on September 18th, 2017. (cited on Page 2, 4, and 14)

[SM10] Bonita Sharif and Jonathan I Maletic. An eye tracking study on camelcase and under_score identifier styles. In *Program Comprehension (ICPC), 2010 IEEE 18th International Conference on*, pages 196–205. IEEE, 2010. (cited on Page 4)

[SW65] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52(3/4):591–611, 1965. (cited on Page 19)

[TIO] TIOBE. Tiobe index. Website. Available online at https://www.tiobe.com/tiobe-index//; visited on October 6th, 2017. (cited on Page 12)

[Tob] Tobii. Tobii eyex tracker. Website. Available online at http://tobiigaming.com/product/tobii-eyex/; visited on September 12th, 2017. (cited on Page 17)

[Wer23] Max Wertheimer. Untersuchungen zur lehre von der gestalt. ii. *Psychologische forschung*, 4(1):301–350, 1923. (cited on Page 3)

[Zü] Universität Zürich. Einfaktorielle varianzanalyse (mit messwiederholung). Website. Available online at http://www.methodenberatung.uzh.ch/de/datenanalyse/unterschiede/zentral/evarianzmessw.html; visited on September 12th, 2017. (cited on Page 25)

Zusammenfassung

Diese Arbeit untersucht den Effekt von Einrückungen auf Programmverständnis basierend auf den positiven Ergebnissen einer Studie von Shneiderman et al. Zusätzlich werden das subjektive Schwierigkeitsempfinden in Betracht gezogen und das ursprüngliche Design erweitert, um zusätzliche Erkenntnisse über den Einfluss von Einrückungen auf das visuelle Verhalten zu gewinnen. Das Ziel dieser Arbeit war es, eine empirische Begründung für die Empfehlung von Einrückungstiefe in Coding Conventions zu liefern. Im Laufe unserer Studie wurden 22 Teilnehmer darum gebeten, die Ausgabe von Java-Programmen mit verschiedenen Einrückungstiefen zu bestimmen, während ihre Blickbewegungen mittels eines Eye-Trackers aufgezeichnet wurden. Obwohl kein signifikanter Nachweis für die untersuchten Effekte gefunden wurde, kann diese Arbeit ein Ausgangspunkt für weitere Studien in diesem Bereich sein.

Eidesstattliche Erklärung:

Hiermit versichere ich an Eides statt, dass ich diese Bachelorarbeit selbständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe und dass alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, als solche gekennzeichnet sind, sowie dass ich die Bachelorarbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt habe.

Jennifer Bauer

Passau, den 12. Oktober 2017