Informatics in Education, 2020, Vol. 19, No. 4, 521–541

DOI: 10.15388/infedu.2020.23

Development of Program Comprehension Skills by Novice Programmers – Longitudinal Eye Tracking Studies

Magdalena ANDRZEJEWSKA, Paweł KOTONIAK

Pedagogical University of Krakow, Institute of Computer Science, Cracow, Poland
e-mail: magdalena.andrzejewska@up.krakow.pl, pawel.kotoniak@student.up.krakow.pl

Received: June 2020

Abstract. The article discusses the findings of longitudinal studies (three stages spanning 6 months) which were to investigate the process of acquiring the ability to comprehension program code by the computer science students having started to learn to program. The studies were conducted with the use of a knowledge measurement test, the diagnostic survey, and eye tracking technology that enabled the recording of movement of the subjects' eyes and an analysis of the patterns of information processing during solving programming problems. The obtained results have shown that the students solved the tasks most effectively in the last stage of the research during which they obtained the highest indicator of correct answers in the significantly shortest time. In the last stage of the research the dominant form of the algorithmic problem analysis was code, in two previous it was flowchart. The eye tracking data have shown that regardless of the research stage the code analysis was definitely connected with a greater number of fixations, with very near values of time devoted to solving those two forms of the algorithm. The participants who increased their competences in a scope of the program code analysis had a significantly greater saccade amplitude average (SAA) and a significantly shorter fixation duration average (FDA) in the last stage of the research comparing to previous ones. The results suggest that the FDA and SAA are parameters sensitive to the development of program comprehension skills.

Keywords: novice programmers, program comprehension, eye tracking.

1. Introduction

Programming, the essential competence in computer science, is commonly considered one of the most demanded but also difficult skills to master. The complexity of the process of learning programming poses many challenges to teaching methodology. It is shown that the main sources of difficulties regarding the programming teaching process include: the didactic methods used by teachers, ways of learning by students, an insufficient level of base skills of students and their motivation, a multi-dimensional character of programming skills and psychological-social factors connected with the perception of programming (Gomes and Mendes, 2007). The phenomenon has long been the subject of

interest of a broad range of researchers because the problems occurring in this field reoccur and appear before each subsequent generation of students of information technology and related disciplines (incl. Mc Cracken *et al.*, 2001; Lister *et al.*, 2004; Pears *et al.*, 2007; Clear *et al.*, 2011; Konecki, 2014).

In psychological terms, programming is a cognitive activity requiring the utilisation of various kinds of mental models (Brooks, 1983). A model of fundamental importance is one related to solving problems, constructing and representing algorithms. Learning programming, however, is not just about acquiring or developing one's skills in the domain of problem-solving. It also requires the learner to become familiar with many abstract notions connected with programming mechanics, with the syntax of the programming language, and with the semantics of the developed code (Robins, 2003).

Many studies show that not only creating new algorithms but also analysing ready solutions – and especially implementing them in the programming language – pose a great challenge to learners (Govender and Grayson, 2006; Moström, 2011; Mendes *et al.*, 2012). According to Lahtinen *et al.* (2005), the biggest difficulty for programmers-beginners is not the understanding of basic concepts but the ability to use them in practice. In turn, in their research, Lister *et al.* (2004) found that students often lack the skills and abilities essential to the process of code reading. According to researchers, it is exactly this area that should be developed in the first place, as a domain preceding the ability to write one's own code and solve problems.

1.1. *Flowchart Versus Code*

One of the first skills acquired by novice programmers is algorithm analysis – the ability to read and write them failure to master the competences connected with a representation and a structure of algorithms is widely recognized as the basic reason for difficulties accompanying programming learning (Gomes and Mendes, 2007). An important subject matter at this stage is the answer to the question of which teaching strategy used to present algorithms is more effective: a code-based one or a flowchart-based one. Does the integration (simultaneous presentation) of the two algorithm forms affect the process of learning programming? If so, in what way? A number of studies have been conducted in this field, with different – sometimes even contradictory – results obtained. For instance, the research conducted by Scanlan (1989) suggested a clear advantage (in the process of algorithm comprehension) of a flowchart over a pseudo-code. Likewise, Carlisle *et al.* (2005) found that the majority of students wrote algorithms successfully with the use of flowcharts and that the utilisation of a programming language diverted their attention from the algorithm concept due to the difficulties related to the application of the syntax of the language. The above conclusions stand in opposition to the findings of Ramsey *et al.* (1983) and Shneiderman *et al.* (1977), according to whom flowcharts were just an alternative of representation of language syntax and did not help in understanding algorithms, especially in the case of experienced programmers. Modern research techniques which expand our knowledge about brain function and enable to have an insight in the neurobiological aspects of the learning process and

such experimental methods as eye tracking, which allow following cognitive mechanisms and can provide objective information about the programming learning process, can be helpful while resolving such dilemmas

1.2. *Eye Tracking*

Eye tracking is a technique consisting in recording of a visual activity of a human being. It is believed that eye tracking, which makes it possible to examine the functional performance of the human eye, also provides the means to analyse the visual perception and the cognitive processes related thereto, including, in particular, visual attention (Duchowski, 2003). The main parameters measured by eye trackers during studies are saccades and fixations. Fixations are motor actions of the eye, interpreted as fixing one's gaze on a certain location. We usually measure the duration and the number of fixations. It has been found, for instance, that three fixations occur per second on average during reading, which means they last about 250–300 milliseconds each on average (Rayner, 1998). Studies of attentional processes assume that fixations may act as a basis for objective measurements of cognitive processes. It is thought that the average duration of a fixation is the indicator of the engagement of one's visual attention or the depth of processing of data, the source of which is visual stimuli (Just, and Carpenter 1976). Saccades are rapid movements of the eyes, shifting the gaze from one point to another point of a visual scene. The main characteristic of a saccade is its amplitude related to the distance between subsequent fixation points. It is believed that an analysis of saccade movements can make it possible to draw conclusions regarding decision-making processes. Also, the length of a saccade is connected with the strategy of searching through the visual scene. Saccades longer than 1.6° are typical of a global strategy. Those shorter than 1.6° are considered part of a local strategy. Moreover, it has been found that experts tend to use global strategies of image search, while beginners choose local strategies by default (Francuz, 2013). Eye tracking technology is a set of objective measurement tools that make it possible to identify the visual patterns of information processing. The possibility to interpret them is a new theoretical-cognitive contribution to the research into the learning process and allows the technology to be applied more often in the exploration of many different education-related matters. These matters include: studies of the effectiveness of learning aids, verification of the existing theories on cognitive processes and learning strategies, or exploring individual differences in the process of learning and analysing the patterns of information processing – including those typical of experts and beginners (Lai *et al.*, 2013).

1.3. *Program Comprehension and Eye Tracking*

The current studies conducted with the use of eye tracking technology, which – thanks to the analysis of the relevant cognitive process (concerning visual attention) – help us understand the process of acquisition of the ability to program, pertain to a range of aspects of program code analysis (Obaidellah *et al.*, 2018).

Crosby and Stelovsky (1990) compared the behaviour of beginner and experienced programmers in order to check how experience affected the code scanning patterns in the process of code comprehension. Also Bednarik and Tukiainen (2006) identified differences in program comprehension strategies between expert and novice programmers when reading a program in conjunction with an execution visualization tool. Uwano *et al.* (2006) noticed that in the process of searching for bugs in programs, most of their research subjects first read an entire code and only later focused on selected parts thereof. The time spent on the initial scanning of the code had an impact on the effectiveness of bug detection. Sharif *et al.* (2012) proved that the said time affects the visual effort required to identify bugs, and experienced programmers spent less time than beginners on initial code scanning before proceeding with an actual search for errors.

Results of eye tracking studies have also shown that the process of reading a source code differs fundamentally from reading a text written in a natural language. These differences disappear, however, if the program code becomes similar to a text written in a natural language (Busjan *et al.*, 2011; Binkley *et al.*, 2013) and experts read the code less linearly than beginners (Busjahn *et al.*, 2015). The study of gaze behaviour when reading program code was also dealt with during the First International Workshop on Eye Movements in Programming Education (Bednarik *et al.*, 2014), as a result a scheme for illustrating the differences between reading code and reading natural text was developed.

There have also been studies aimed to answer the question of how code formatting, including the colour scheme adopted for the syntax, affects the effectiveness of analysis (e.g. ability to find syntax errors) and comprehension (e.g. determining the outcome of program execution) of the code. It has been found that coloured syntax shortens the time to perform a task and facilitates its analysis (Dimitri, 2015; Beelders and Plessis, 2016) and that the effect weakens with the growth of the level of experience with programming (Sarkar, 2015). Also, a right spatial structure of the source code makes it easier to understand it quicker, regardless of the adopted naming style for its identifiers (Binkley *et al.*, 2013). Studies on solving algorithmic problems with the use of eye tacking have been rare so far. Andrzejewska *et al.* (2016) have confirmed that the use of a formal notation of a programming language to present algorithms is becoming a challenge for beginners in the process of solving even relatively simple programming problems.

2. Method

2.1. *Aim of the Study and Research Questions*

The eye tracking studies conducted so far have not yet explored the development of the ability of logical analysis of program source code within the first months of learning programming, which seems to be a crucial period, during which significant progress in code reading and comprehension should be made. It is interesting to see whether the application of eye tracking technology makes it possible to extend and supplement the other methods diagnosing the issue in question.

The goal of this study was to examine how students developed their ability of logical analysis of program source code (reading and understanding) at the initial stage of learning programming. The paper aims, in particular, to answer the following main research questions:

- **(RQ1)** *Are there significant differences in the behavioural, subjective, and eye tracking factors when doing program comprehension tasks between the stages of the research?*
- **(RQ2)** *Does the form of algorithm presentation (code vs. flowchart) significantly affect the performance of the task and the eye tracking parameters during the two stages of the research?*
- **(RQ3)** *(a) Does the selection of a flowchart signify a smaller progress in the development of the ability to understand the code, and (b) What eye tracking measures are sensitive to the development of students' program comprehension?*

2.2. *Experiment Design*

The independent variables were: *a research stage* – three stages spanning six months (Stage 1 using only traditional materials, Stage 2 and Stage 3 using an eye tracking device; *task* – three types of programming problems; *a task form* – two forms of presenting a programming problem in Task 3 (a code vs. a flowchart); *a group* – the research subjects were assigned to one of three groups depending on the chosen form of solving the Task 3 in Stage 2 and Stage 3.

The dependent variables were divided into behavioural, subjective and eye tracking variables. The behavioural indicators of program comprehension are: achievement, which relates to the *rate of correct answers* (0 – incorrect, 1 – partly correct, 2 – correct); *time* (ms), which refers to the number of milliseconds spent answering each task; *exam test results* (%) obtained by students for the C programming language after the first and second learning semesters. The subjective indicator is the students' perception of the *difficulty of the tasks* measured in a post-survey study. Students rated the difficulty level related to the analysed tasks on the Likert scale from 1 (very easy) to 5 (very difficult).

With regard to eye tracking parameters, we focused on: *fixation count* (FC: number of fixations); *fixation duration average* (ms) (FDA: the sum of duration of all fixations divided by the number of fixations); *saccade amplitude average* (°) (SAA: the sum of all saccade amplitudes divided by the number of saccades); *scanpath length* (px) (SL: the sum of the lengths (distance from start to end) of all saccades); *dwell time* (ms) (DT: the sum of the duration of all fixations and saccades that hit the area of interest – AOI); revisits (the number of glances towards the AOI if saccades came from outside).

The analyses of the collected data were conducted mainly by means of a repeated or mixed-design model of ANOVA (analysis of variance), followed by a pairwise comparison with Bonferroni correction (post-hoc tests).

2.3. *Research Stages and Participants*

The research was carried out at intervals in three stages. The participants were recruited from the population of first-year computer science students. The first stage of the research was conducted at the beginning of the semester in which the subjects started learning to program (i.e. after about 6 weeks). Its important goal was to select those students who had never programmed before. 51 students participated in the stage in question. This stage required participants fill in a questionnaire and to solve the tasks. The questions included in the questionnaire made it possible to determine who had already learned programming in C or another language before taking up the studies. Such persons were excluded and the 35 remaining participants were included into the further stages of the research. Among those, 31 decided to take part in the second stage of the research, which was conducted with the use of an eye tracker about 3 weeks later.

The last – third – stage of the research took place six months later, near the end of the second semester of the course in programming. Unfortunately, in the meantime 7 research subjects did not pass the first semester. In addition to that, 3 other persons resigned from taking part in the last stage of the research project. Ultimately, the study group consisted of 21 participants, 17 men and 4 women, aged 19 to 24 (M = 19.80, SD = 1.20). The measurement data collected in both stages of eye tracking studies were analysed for their quality and no artefacts were found that would require excluding further research subjects.

2.4. *Eye Tracking Device and Procedure*

Apparatus. Our studies were conducted with the use of the iViewX Hi-Speed eye tracker manufactured by SensoMotoric Instrument (SMI). It is an apparatus designed to carry out non-invasive high sampling rate (i.e. 500/1,250 Hz) studies, categorised as a high-performance stationary device, used mostly in laboratory conditions.

The workstation includes a computer used to manage the entire conducted experiment, a computer screen, and an eye tracking module. The device has been designed to allow the person using it to keep their head still without their field of vision being limited.

During the experiment, the images were presented on an LCD screen on a 23” diagonal screen with a full HD resolution of 1920 x 1080. A 9-point calibration was performed before each session. The experiment was performed with the use of software called SMI Experiment Suite™ 360. The experiment scheme was designed using the SMI Experiment Center™ 3.4, and the data was recorded with the use of SMI iView X™. The results were processed using SMI BeGaze™ 2.4

The eye tracking session. Each eye tracking session was conducted individually. At the beginning, the participant became familiar with the course of the research procedure. Next, the device was calibrated, and the participant was presented a chart with a set of instructions and a number of charts shown one after another, each with a task to be performed. The final part of the session involved a validation procedure aimed at verifying the correctness of the collected results. After leaving the workstation, students filled in a short questionnaire

2.5. *Data Collection Instruments*

During the first stage the students filled in a questionnaire which included questions about their demographic details (age, sex) and about their perceived level of skills and experience in programming. Additionally, they solved 3 programming tasks, almost identical to those which were used in the further stages of the research. On the answer sheet, they noted the start and finish times of each task, and assessed the difficulty of each of them at the end. In the second and third stage of the eye tracking studies the students solved 3 tasks, and then they filled a short questionnaire with questions asking them about the problems they encountered during the analysis of the programs and about their perceived level of difficulty of the tasks. Both eye tracking studies involved the same tasks and questionnaires, and the tasks were presented in the same order. The examination test results (expressed as a percentage) which the students obtained after the first and second semester of two subjects learning conducted in the C programming language (Introduction to programming, Procedural programming) were also used as the data.

Programming tasks. Research subjects analysed and determined the result of execution of 3 programming tasks. The first two tasks (Task 1 and Task 2) were short but complete codes of programs developed in the C language. Task 3 presented the same algorithm expressed using a flowchart (right side of the task chart) or a complete program code (left side of the task chart). This task required the research subjects to first choose one of the two methods of algorithm presentation and then analyse the chosen option. Task 1 required an analysis of a `for` loop code; the loop made 3 iterations. Each step of the loop required the research subjects to calculate a simple expression based on the multiplication and subtraction of the current value of the variable in the loop. Task 2 required an analysis of a `while` loop; the loop made 5 iterations. Each step of the loop executed a conditional instruction, which calculated a simple conditional expression verifying if the value of the variable in the loop was even. The levels of difficulty of tasks 1 and 2 were similar. Both tasks required remembering a modified value of only one variable at each step of the loop. Task 3 required an analysis of a `while` loop, which made 3 iterations for the provided input data. Each step of the loop involved executing two simple instructions which modified the value of two variables. The task was the most difficult of all three as it required remembering intermediate values of the said 2 variables and the value of the input data.

3. Results

3.1. *(RQ1) Are there significant differences in the behavioural, subjective, and eye tracking factors when doing program comprehension tasks between the stages of the research?*

The answer to this question involved the use of the following metrics: time, the rate of the correct answers, the perceived task difficulty, the fixation count (FC), the fixa-

tion duration average (FDA), the saccade amplitude average (SAA), and the scanpath length (SL). With regard to behavioural factors (time and the rate of correct answers), a repeated measures ANOVA with two within-subject independent variables (3 stages x 3 tasks) was performed.

As for the time, the analysis of variance revealed significant differences between the stages [$F(2, 40) = 8.803$, $p = 0.001$]. The time spent to solve tasks in the first two stages was similar in the case of both stages and much longer than in the case of Stage 3 (Stage1 M = 148.57 s, Stage 2 M = 142.32 s, Stage 3 M = 115.05 s). Post-hoc comparisons showed that the differences between Stage 3 and Stage 1 ($p=0.001$) and between Stage 3 and Stage 2 ($p = 0.008$) were significant. The analysis of variance revealed neither a significant main effect of the performed tasks [$F(2, 40) = 2.125$, $p = 0.133$] (Task1 M = 131.26 s, Task 2 M = 128.53 s, Task 3 M = 146.15 s) nor an effect of interaction between the tasks and stages [$F(4, 80) = 0.867$, $p = 0.488$)].

Task 3 was analysed the longest, which is the expected result since the task required the largest number of activities. Yet, Table 1 shows that at Stage 2, Task 3 was analysed slightly shorter than Task 2, which might be an effect of fatigue and a decision not to analyse the task in so much detail at the stage in question.

When it comes to the students' achievements, the data revealed a significant main effect of particular stages [$F(2, 38) = 8.24$, $p = 0.001$]. The rate of correct answers was significantly higher for Stage 3 of the research (Stage 3 M = 1.55) compared to the two previous stages (Stage 1 M = 1.23, Stage 2 M = 0.92). A post-hoc analysis showed significant ($p = 0.001$) differences to exist between Stage 2 and Stage 3. Better results obtained in Stage 1 as compared to Stage 2 can be explained by the fact that in the case of Stage 1, the students could solve the problem using a sheet of paper to write down their calculations. In the case of Stage 2, they had to "memorise" all intermediate values of the variables and make calculations in their heads.

In addition the type of the task had a significant main effect on the accuracy of the answers to the performed tasks [$F(2, 38) = 6.48$, $p = 0.004$]. Pairwise comparisons proved that the lowest indicator obtained for Task 2 (M = 0.92) differed significantly from both Task 1 (M = 1.37, $p = 0.017$) and Task 3 (M = 1.42, $p = 0.007$). But the stages x tasks interaction effect was not significant [$F(4, 76) = 1{,}22$, $p = 0.309$]. The most difficult task (one with the lowest rate of correct answers at each stage (see Table 1) appeared to be Task 2, which was expected to be the easiest task. It seems that the most likely reason behind the above was the subjects' wrong interpretation of the conditional

Table 1

Means of the behavioural dependent variables for interaction effect (3 stages and 3 tasks)

Task	Time			Correct Answers		
	Stage 1	Stage 2	Stage 3	Stage 1	Stage 2	Stage 3
1	140.00	147.63	106.16	1.40	1.10	1.60
2	137.14	136.74	111.71	1.05	0.40	1.30
3	168.57	142.60	127.28	1.25	1.25	1.75

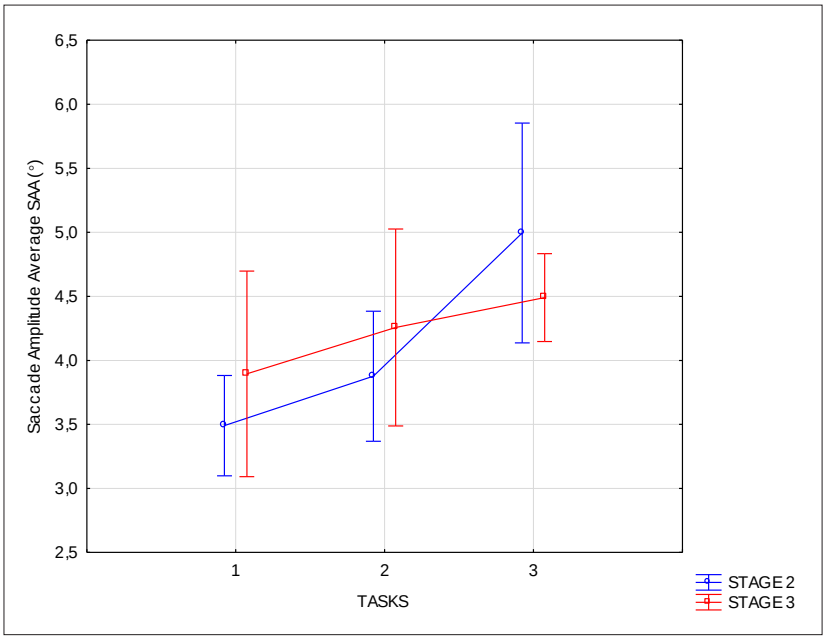

Fig. 1. Saccade Amplitude Average SAA (2 stages and 3 tasks).

expression in the instruction, i.e. `if (!(i%2))`. Students often find it difficult to calculate the value of expressions in the C language using logical operators combined with arithmetic operators.

To examine the subjective factor, a repeated measures ANOVA with one within-subject independent variable (3 stages) was performed. The analysis revealed significant differences with respect to the students' perception of the difficulty level of tasks between the particular stages [$F(2, 38) = 14.671$, $p < 0.001$]. According to the research subjects, the performed tasks were easiest at Stage 3 ($M = 1.95$); a post-hoc test showed that the opinion differed considerably in the case of both Stage 1 ($M = 2.60$, $p < 0.001$) and Stage 2 ($M = 2.65$, $p < 0.001$).

To analyse the eye movements parameters, a repeated measures ANOVA with two within-subject independent variables (2 stages x 3 tasks) was performed. The analysis focused mainly on parameters connected with fixations and saccades: fixation count (FC), scanpath length (SL), fixation duration average (FDA), and saccade amplitude average (SAA).

The data in Table 2 shows that in the case of the fixation number and scanpath length, there occurred a significant main effect between the particular stages. The research subjects displayed a significantly greater ($p = 0.003$) number of fixations and covered a significantly longer ($p = 0.005$) scanpath in Stage 2 (see Table 3). A significant main effect (for SL) or a main effect on the border of significance (for FC) can also be seen between the particular tasks. Post-hoc tests showed that the SL in Task 3 differed significantly from the SL in both Task 1 ($p < 0.001$) and Task 2 ($p < 0.001$). The obtained finding is

Table 2

Repeated ANOVA for the eye movements dependent variables (2 stages and 3 tasks), main and interaction effects

Effect	Scanpath (px)		FDA (s)		SAA (°)		FC	
	F	p	F	p	F	p	F	p
Stages	9.752	0.005	< 1	> 0.05	< 1	> 0.05	11.730	0.003
Tasks	21.011	0.000	7.843	0.001	17.218	0.000	3.031	0.059
Stages x Tasks	< 1	> 0.05	< 1	> 0.05	3.679	0.034	1.551	0.225

Table 3

Means of the eye movements dependent variables for main effect (2 stages and 3 tasks)

		Scanpath (px)	FDA (s)	SAA (°)	FC
		M	M	M	M
Stages	2	59188.16	298.97	4.12	408.21
	3	47805.14	294.19	4.21	335.89
Tasks	1	44438.24	311.78	3.69	350.55
	2	48562.45	296.86	4.07	359.95
	3	67489.26	281.09	4.74	405.64

not surprising. Both these parameters (FC and SL) are usually positively and strongly correlated with the duration of task performance (see Table 1).

Significant main effects for tasks were also discovered in the case of the fixation duration average (FDA), which was the longest for Task 1, and of the saccade amplitude average, with the highest value for Task 3 (see Table 3). A post-hoc analysis of the FDA revealed a significant ($p = 0.001$) difference between Task 3 and Task 1, and in the case of SAA, it showed significant differences to occur between Task 3 and both Task 1 ($p < 0.001$) and Task 2 ($p = 0.002$).

The longest FDA in Task 1 occurred in the case of both Stage 2 and Stage 3 of the research (see Table 4), which is most likely related to the greatest involvement of the research subjects in this task – which can be referred to as the "first task effect".

We suppose that the highest values of the SAA in Task 3 stem from the nature of the task, i.e. the feature of a flowchart, whose component arrangement translates into greater

Table 4

Means of the eye movements dependent variables for interaction effect (2 stages and 3 tasks)

Task	Scanpath (px)		FDA (s)		SAA (°)		FC	
	Stage 2	Stage 3	Stage 2	Stage 3	Stage 2	Stage 3	Stage 2	Stage 3
1	51042,05	37834,43	314,46	309,10	3,49	3,90	403,81	297,29
2	53380,81	43744,10	297,73	295,99	3,88	4,26	399,48	320,43
3	73141,62	61836,90	284,71	277,47	5,00	4,49	421,33	389,95

saccade amplitudes. The significant stages x tasks interaction effect, which can also be seen in Fig. 1, comes from the fact that at Stage 3, a greater number of research subjects solving Task 3 chose the version expressed in the form of a code, and at Stage 2, most chose a flowchart. This resulted in significantly lower SAA values in the case of Task 3 at the last stage of the research. It is reasonable to assume that the difference in the SAA values as found between Task 1 and Task 2 (see Table 4) also result from the fact that the graphic form of the code written with the use of the for-loop syntax (Task 1) is more concise compared to the syntax of the while loop (Task 2).

3.2. *(RQ2) Does the form of algorithm presentation (code vs. flowchart) significantly affect the performance of Task 3 and the eye tracking parameters during the two stages of the research?*

During Stage 1 and Stage 2, the number of the research subjects who decided to solve Task 3 with the use of either a code or a flowchart was the same, amounting to: code N = 8 (38%), flowchart N = 13 (62%). During Stage 3, 7 students from among those who opted for a flowchart before, decided to solve the task on the basis of a code, hence the populations of groups opting for either a code or a flowchart at that stage were as follows: code N = 15 (71%), flowchart N = 6 (29%) (see Table 7).

The rate of correct answers, dwell time (DT), revisits, and fixation count (FC), meaning eye tracking metrics connected with the highlighted areas of interest representing the code and the flowchart, were used to answer question RQ2. A mixed-design model ANOVA (2 stages x 2 forms) was performed.

When analysing the data given in Table 5, we can see a significant main effect occurring between particular stages within the body of the solutions of Task 3. The research subjects arrived at significantly ($p = 0.013$) higher rates of correct answers in Stage 3 of the research (see Table 6).

The students who opted for a code, performed, in general, worse. But the values for both presentation forms were similar for Stage 3. A significant ($p = 0.028$) difference in achievements was revealed between the particular stages in the scope of the code-based analysis (see Table 7).

No significant main effect was discovered in the case of the dwell time (DT) spent on solving Task 3 (see Table 5). Looking at Table 6, we can notice that the students

Table 5
Mixed-design ANOVA for the dependent variables (2 stages and 2 forms), main and interaction effects

Effect	Correct Answer		Dwell Time (s)		Fixation Count		Revisits	
	F	p	F	p	F	p	F	p
Stages	7.001	0.012	< 1	> 0.05	1.260	0.269	< 1	> 0.05
Forms	1.056	0.311	< 1	> 0.05	4.073	0.051	< 1	> 0.05
Stages x Forms	1.788	0.189	< 1	> 0.05	< 1	> 0.05	< 1	> 0.05

Table 6
Means of the dependent variables for main effect (2 stages and 2 forms)

		Correct Answer	Dwell Time (s)	Fixation Count	Revisits
		M	M	M	M
Stages	2	1.13	88.32	271.00	25.20
	3	1.70	82.13	243.18	23.15
Forms	Code	1.30	85.86	282.10	22.88
	Flowchart	1.53	84.59	232.08	25.47

Table 7
Means of the dependent variables for interaction effect (2 stages and 2 forms of algorithm)

Stage	N (%)		Correct Answer		Dwell Time (s)		Fixation Count		Revisits	
	FC	Code	FC	Code	FC	Code	FC	Code	FC	Code
2	13(62)	8(38)	1.38	0.88	85.75	90.87	241.00	301.00	26.77	23.63
3	6(29)	15(71)	1.67	1.73	83.42	80.84	223.17	263.20	24.17	22.13

analysed their task during Stage 2 generally longer, but at that stage, unlike in the case of Stage 3, they spent more time on analysing the code than the flowchart (see Table 7).

Also in the case of the eye tracking parameters connected with AOI, meaning FC and Revisits, there were no significant main effects of interaction between the stages and the forms of Task 3 (see Table 5). But it is important to bear in mind that the main stages effect for FC was on the border of significant, and the number of fixations was much higher in Stage 2.

Regardless of the study stage, a much lower number of fixations was observed for persons who used a flowchart to solve the task (see Table 7). The difference stems probably from the mechanism of code analysis, which is read quite like written natural languages, with frequent fixations. Analysing an algorithm in the form of a flowchart determines the direction of the gaze (we make a saccade). It seems that following the structure of a flowchart does not require the same number of fixations as in the case of processing a code. An important thing to mention here is that in the case of Stage 3, the research subjects analysing the code made more fixations and spent less time solving the task compared to those analysing the flowchart (see Table 7).

The Revisits parameter pertains to the number of return saccades, which in the case of this task are related to the need to return to the area of instruction, to the task containing the numerical data (values of input variables) necessary to solve the problem. The value of the parameter was two times higher in Stage 2, and always slightly lower for the code-based format, regardless of the stage. Those solving the task based on a flowchart analysis returned probably more often to the content of the problem to check the values of the input variables.

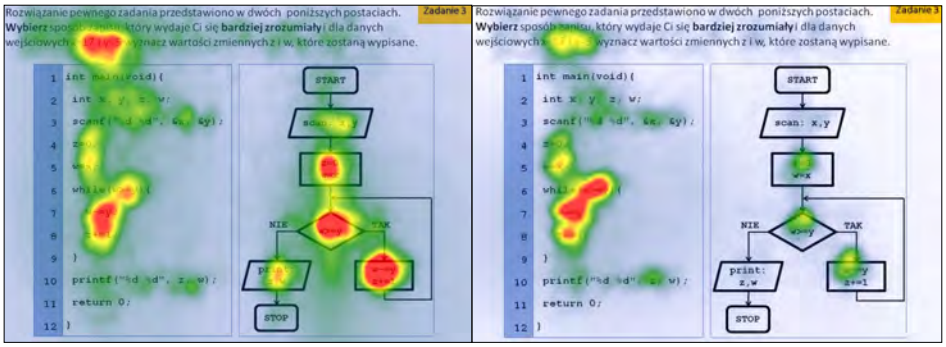

Fig. 2. Heat maps Task 3 (left: Stage 2, right: Stage 3).

The spatial distribution of visual attention during the analysis of Task 3 is also illustrated by the heat maps included in Fig. 2, where we can see the discussed differences between Stage 2 and Stage 3 of the research project.

3.3. *(RQ3) (a) does the selection of a flowchart signify a smaller progress in the development of the ability to understand the code, and (b) what eye tracking measures are sensitive to the development of students' program comprehension?*

Based on the choices the research subjects made regarding the preferred form of solving Task 3, they were divided into 3 groups: Group A (N = 8) – persons who chose the code at every stage of the research, Group B (N = 7) – persons who chose the flowchart at Stage 1 and Stage 2, and the code at Stage 3, Group C (N = 6) – persons who always chose the flowchart. It is likely that the flowchart was chosen by persons who find it easier to interpret algorithms presented in this form, meaning persons who did not made any considerable progress in learning between the particular stages of the research. Group B made probably the biggest – and Group C – the smallest – progress in the development of their program comprehension skills between Stage 2 and Stage 3.

The C programming language examination results (test results) obtained by the students after the first and second semesters of learning, the rate of correct answer, and the subjective difficulty assessment were used to answer question RQ3 (a).

A repeated measures model ANOVA with one within-subject and one between-subject independent variable (2 stages x 3 groups) was also performed.

Table 8

Repeated ANOVA for the dependent variables (2 stages and 3 groups), main and interaction effects

Effect	Test Results		Correct Answers		Difficulty		FDA (ms)		SAA	
	F	p	F	p	F	p	F	p	F	p
Groups	< 1	> 0.05	8.988	0.002	< 1	> 0.05	1,188	0.328	2.077	0.154
Stages x Groups	1.098	0.355	< 1	> 0.05	< 1	> 0.05	5.833	0.011	3.476	0.053

When analysing the data in Table 8, we can see that a significant main effect occurred only for the rate of correct answer. In general and at each stage, Group A performed better than the other two groups (see Table 9). Post-hoc tests pointed to the significance of that difference compared to Group B ($p = 0.004$) and Group C ($p = 0.010$). But another important fact is that in Stage 3, Group B performed better than Group C, unlike in Stage 2 (see Table 10).

Similar relationships surfaced for the test results variable. According to Table 10, Group B's test results were worse after the first semester of learning than those of both other groups (performing only slightly worse than Group C; the difference was bigger compared to Group A). Yet, the distribution of the test results after the second semester was different. Firstly, Group B performed better than in the first semester. In a situation in which both Group A and Group C performed worse than before, Group B managed to match the performance of Group A. The relationships are also illustrated in Fig. 3.

When analysing the subjective difficulty, it is necessary to notice that all groups considered their tasks easier at Stage 3 than at Stage 2, with Group B reporting the biggest difference in the perceived difficulty of the tasks (which amounted to -0.86) (see Table10).

The fixation duration average (FDA) and the saccade amplitude average (SAA) metrics were taken into consideration to answer question (RQ3) (b).

A repeated measures model ANOVA with two within-subject and one between-subject independent variable was performed for the following dependent variables: FDA (2 stages x 3 tasks x 3 groups) and SAA (2 stages x 2 tasks x 3 groups). The analysis of the SAA variable considers only 2 tasks because Group B changed their preferences at Stage 3 and solved the other form of the task.

Table 9

Means of the dependent variables for main effect (3 groups)

		Test Results	Correct Answers	Difficulty	FDA (ms)	SAA (°)
		M	M	M	M	M
I	A	0.54	1.55	2.29	322.1	3.53
	B	0.51	1.05	2.43	326.1	4.66
	C	0.47	1.08	2.17	359.3	3.45

Table 10

Means of the dependent variables for interaction effect (2 stages and 3 groups)

I	Stage	Test Results	Correct Answers	Difficulty	FDA (ms)	SAA (°)
A	2	0.58	1.14	2.57	315.22	3.45
	3	0.51	1.95	2.00	328.93	3.60
B	2	0.50	0.71	2.86	350.36	4.14
	3	0.51	1.38	2.00	301.75	5.17
C	2	0.51	0,89	2.50	352.10	3.46
	3	0.43	1.28	1.83	366.43	3.42

In the case of the discussed eye tracking parameters, a significant stages x groups interaction effect was observed for the FDA variable (see Table 8). When analysing Table 10, we can see that Group A displayed the shortest and similar FDAs in both Stage 2 and Stage 3. Group B displayed a significantly ($p = 0.044$) longer FDA in Stage 2 than in Stage 3. Group C displayed the longest and almost equal FDAs in both stages.

The effect of stages x tasks x groups interaction [$F(4, 36) = 1.269$, $p = 0.300$] did not appear to be significant, but if we look at Fig. 4, we will notice that Group B displayed shorter FDAs for each task in Stage 3 compared to Stage 2.

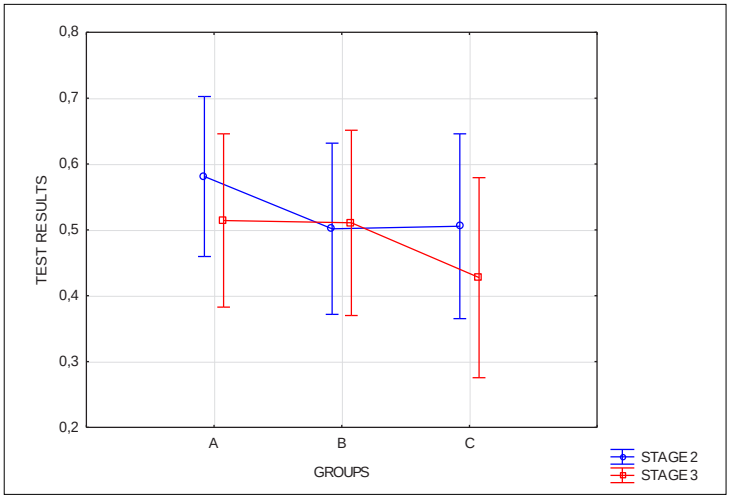

Fig. 3. Test results (2 stages and 3 groups).

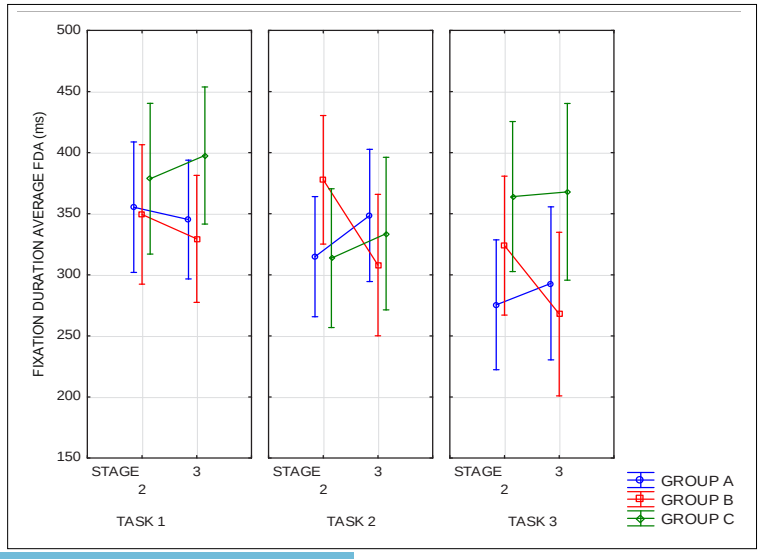

Fig. 4. Fixation Duration Average FDA (2 stages and 3 tasks and 3 groups).

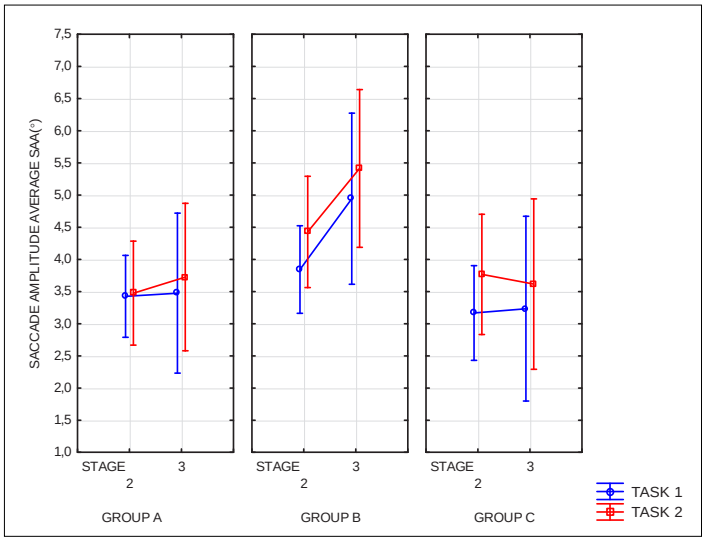

Fig. 5. Saccade Amplitude Average (2 stages and 2 tasks and 3 groups).

In the case of the SAA variable, the effect of the stages x groups interaction was on the border of significance. Group B displayed much higher values of the parameter than the other two groups (see Table 9) – in general and at both stages of the research. The performed post-hoc test showed that the results for Group B differed significantly between the particular stages ($p = 0.047$).

The effect of stages x tasks x groups interaction [$F(2, 18) = 0.462$, $p = 0.637$] was not significant, but if we analyse Fig. 5, we will notice that Group B displayed much higher SAAs for both tasks in Stage 3 compared to Stage 2. We do not see such a change in the case of the two other groups.

The analysis of our behavioural data shows that Group B made the biggest progress in their program comprehension ability and appeared to be the group with a significant increase in the SAA value, which may imply that this parameter is sensitive to the development of one's skills in the field in question.

4. Discussion

As for RQ1, we found that students solved tasks in the third stage of the research more effectively compared to the other two stages. A significant difference occurred between Stage 2, where the research subjects had the lowest score in the correct answer category, and Stage 3, where they scored the highest in the said category. Moreover, the time spent to solve tasks in Stage 3 was significantly shorter than in the case of the other two stages, where the task solving time was similar. In the final stage, the subjective perception of the difficulty of the tasks was also the lowest. The obtained results confirm that the students improved the level of their skills and those are findings which were to be expected.

A preliminary analysis of eye movement parameters showed that the values of these parameters were related to behavioural indicators (e.g. the time spent to solve tasks) or the graphic presentation of tasks. A greater number of fixations and a longer scanpath length can be seen in the case of Task 3 and at Stage 2 of the study as it was more time-consuming. The longest FDA, regardless of the research stage, was recorded for Task 1, and most likely suggests the greatest level of involvement of the research subjects in solving the task in question. The highest value of SAA, in turn, were recorded for Task 3, and it seems that this results from the feature of the flowchart, whose structure of components translates into greater amplitudes of saccades with respect to the program code. Our results are consistent with the reports from other studies which suggest that the parameters of glances are strongly connected with a character of tasks and even more sensitive to task- related aspects than to levels of expertise (Sharma, 2011).

As for RQ2, we can see that the most often chosen form of analysis of algorithmic problem at Stage 3 of the research is the code, with the flowchart selected mainly in the two previous stages. At Stage 2, the choice of the flowchart was connected with a higher rate of correct answers, unlike in the case of Stage 3, where the values for both forms were very similar. Also, we can see a significant increase in the rate of correct answers during code analysis at the said stage comparing to Stage 2. Thus like in the case of the studies conducted by Scanlan (1982), Carlise *et al.* (2005) and Andrzejewska *et al.* (2016) the flowchart turned out a preferred and more effective form of the algorithm presentation at the initial phase of learning to program.

The obtained eye tracking data showed that regardless of the stage of research, code analysis involved a much greater (bordering on significance) number of fixations, with very similar values of the time spent by the research subjects to solve the two forms of the presented algorithm. Our conclusion is that it is connected with the nature of the action of code processing. A code is read like a text written in a natural language, especially by programming beginners. This explanation is coherent with the results of Busjahn *et al.* (2015) study, who notice that novices read the code in a more linear way than experts.

We have also noticed that in the case of both Stage 2 and students who chose a flowchart, there have been more frequent revisits to the area of task which contained the initial values of variables, required to make calculations in Task 3. We suppose that it results from cognitive overload connected with a need to store those data in the memory at simultaneously monitoring the control flow and executing operations in the successive loop steps, which is more difficult for students with lower programming skills. Similar to our findings also Obaidellah *et al.* (2020) observed an increase in the number of fixation regressions that were associated with transitions between the pseudocode and problem description areas, as the difficulty of the problems increases.

As for RQ3 (a), it seems that the choice of the form of analysis of Task 3 is an objective indicator of the development of competence in the field of programming language syntax comprehension. The performed data analysis showed that the individuals who opted for a code in each of the eye tracking studies (Group A) did actually

achieve a higher rate of correct answers compared to other research subjects. They also performed best in the examination test. An important fact is that those individuals who changed their preferences regarding the chosen form of solving Task 3 and opted for a code at Stage 3 (Group B) displayed the biggest growth differences in the area of the discussed variables, and reached the performance level of Group A in the examination test at Stage 3. In the case of individuals who decided to opt for a flowchart (Group C), we see the lowest growth in the rate of correct answers and a deterioration of their examination performance. In the light of the above, it seems to us that Group B has made the biggest progress in the development of their program comprehension skills.

As for RQ3 (b), Group A and Group C, who did not display any considerable changes in the level of their program code analysis ability, did not display changes in the field of the FDA between the particular stages either. Furthermore, the parameter in question assumed the lowest values in Group A, and the highest values in Group C. In the case of Group B, it displayed a significantly shorter FDA at Stage 3 compared to the previous stage of the research. Similar relationships were found for SAA. Group B displayed a significantly higher SAA in Stage 3 of the research, and no changes were observed in the case of the other two groups. The obtained findings suggest that FDA and SAA are parameters sensitive to the development of program code understanding. This observation is consistent with reports from the studies where patterns of reading of the program code by novice and non-novice programmers were compared and it has been found out that non-novices have more transitions that span multiple lines, so their patterns are characterized by longer saccades (Peterson *et al.*, 2019).

5. Threats to Validity and Study Limitations

Our study suffers from some limitations that should be taken into consideration. Firstly, the population of our sample at the third stage of the experiment was rather small. Although the results of some eye tracking measurements revealed significant trends, a larger sample would certainly increase the statistical power of the study. The size of the sample was small mainly because some persons qualified for the initial stages of the research discontinued their education. The potential changes of the sample size are a factor that needs to be always taken into account when designing longitudinal studies. In the case of this research project, however, the situation was all the more difficult because its participants were only persons who had just started learning programming and were strongly exposed to failure in their education. The small sample size translated into a low statistical power of the results, where the research subjects were divided into 3 groups. But the three-conditional research projects did produce a number of interesting findings. In the case of eye tracking studies, the final sample size is usually also determined by the difficulties related to the quality of the obtained measurement data, affected e.g. by the precision of calibration of the testing device. Fortunately, it was possible to avoid discarding any such cases in this experiment.

Secondly, we could also have formulated some remarks about the design employed in this study. In the case of experiments utilising eye tracking technology, given the design of the user interface, it is hard to process a multi-level code, meaning a code whose size makes it stretch beyond a single screen. Our study made use of very short programs – in a limited quantity. This is because experience shows that such studies, while being non-invasive, are very exhausting for research subjects and thus if they last long, the risk of errors being made grows significantly. This could be of crucial importance to this experiment because the task charts were displayed in the same order and the research subjects, after performing the first two tasks, proceeded to the analysis of the following task someone tired, especially during Stage 2. In the case of Task 3, it should be also noticed that both areas of interest – the code and the flowchart – were featured on one chart, which could have caused some artefacts regarding the eye tracking measurement data.

Thirdly, when it comes to the perceived subjective difficulty of tasks, it would be a good idea to provide for a more precise and therefore more reliable measurement and apply a more sensitive scale – e.g. 1 to 10 instead of the current 5-point Likert scale. Moreover, we assessed the difficulty of all tasks on a one-off basis after the end of each stage. It would be more interesting to see the opinions on each task individually, which would also let us control the answer to the question about the most difficult task as perceived by the research subjects.

Moreover the results of two exam test that were considered the dependent variable (Stage 2 and Stage 3 of the research) have taken place approximately 1 month after the eye tracking session. Perhaps it would be better to carry out such an extended test at exactly the same time instead.

6. Conclusions

The main objective of our research project was to answer the question of how students develop their ability of logical analysis of program source code (reading and understanding) at the initial stage of learning programming. We employed an eye tracking approach that provided objective information on how the research subjects processed program codes, and – more importantly – how the eye movement data linked with the behavioural data, which allowed us, in turn, to draw conclusions regarding the sensitivity of eye movement parameters in the context of developing one's code analysis skills.

This issue could be explored further. Future studies would benefit from the utilisation of more complex codes – offering more difficulty levels, as well as from a more numerous sample, one more diversified in terms of the previous experience with programming. This would make it possible to obtain a fuller view of the analysed issue.

The obtained findings contribute to the exploration of the cognitive processes behind learning to program and expand the body of knowledge in this domain in the theoretical aspect, but may also find application in the field of computer science instructional design.

References

Andrzejewska, M., Stolińska, A., Błasiak, W., Peczkowski, P., Rosiek, R., Rożek, B., Sajka, M., Wcisło, D. (2016). Eye-tracking verification of the strategy used to analyse algorithms expressed in a flowchart and pseudocode. *Interactive Learning Environments*, 24(8), 1981–1995.

Bednarik, R., Busjahn T., Schulte, C. (Eds.) (2014). *Eye Movements in Programming Education: Analyzing the Expert's Gaze*. Technical report, University of Eastern Finland, Joensuu, Finland.

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

Beelders, T.R., du Plessis, J-P.L. (2016). Syntax highlighting as an influencing factor when reading and comprehending source code. *Journal of Eye Movement Research*, 9(1), 1–11.

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.

Brooks, R. (1983). Towards a theory of the comprehension of computer programs. *International Journal of Man-Machine Studies*,18(6), 543–554.

Busjahn, T., Bednarik, R., Begel, A., Crosby, M., Paterson, J., Schulte, C., Sharif, B., Tamm, S. (2015). Eye movements in code reading: Relaxing the linear order. In: *Proceedings of the IEEE 23rd International Conference on Program Comprehension*. 255–265.

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.

Carlisle, M. C., Wilson, T. A., Humphries, J. W., Hadfield, S. M. (2005). Raptor: a visual programming environment for teaching algorithmic problem solving. In: *Proceedings of the 36th SIGCSE Technical Symposium on Computer Science Education*. St. Louis, Missouri, USA, 176–180.

Clear, T., Whalley, J., Robbins, P., Philpott, A., Eckerdal, A., Laakso, M., Lister, R. (2011). Report on the final BRACElet workshop. *Journal of Applied Computing and Information Technology*, 15(1).

Crosby, M.E, Stelovsky, J.(1990). How do we read algorithms? A case study. *Computer*, 23(1), 25–35.

Dimitri, G.M. (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.

Duchowski, A.T. (2003). *Eye Tracking Methodology: Theory and Practice*. Springer-Verlag, London.

Francuz, P. (2013). *Imagia. W kierunku neurokognitywnej teorii obrazu*. Wydawnictwo KUL, Lublin.

Gomes, A., Mendes, A. J. (2007). Learning to program – difficulties and solutions, *Proceedings of the ICEE 2007 – International Conference on Engineering Education*.
`http://icee2007.dei.uc.pt/proceedings/papers/411.pdf`

Govender, I., Grayson, D., (2006). Learning to program and learning to teach programming: A closer look. In: (Eds.) Pearson, E., Bohman, P., *Proceedings of ED-MEDIA 2006-World Conference on Educational Multimedia, Hypermedia & Telecommunications*, Orlando, 1687–1693.

Just, M.A., Carpenter, P.A. (1976). Eye fixations and cognitive processes. *Cognitive Psychology*, 8, 441–480.

Konecki, M. (2014). Problems in programming education and means of their improvement. In: Katalinic, B. (Eds.), *DAAAM International Scientific Book 2014*, DAAAM International, Vienna, 459–470.

Lahtinen, E., Ala-Mutka, K., Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14–18.

Lai, M.L., Tsai, M.J., Yang, F.Y, Hsu, C.Y., Liu, T.C., Lee, S.W., Lee, M.H, Chiou, G.L., Liang, J.C, Tsai, C.C. (2013). A review using eye-tracking technology in exploring learning from 2000 to 2012. *Educational Research Review*, 10, 90 –115.

Lister, R., Adams, S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., McCartney, R., Moström, J.E., Sanders, K., Seppälä, O., Simon, B., Thomas, L.(2004). A Multi-National Study of Reading and Tracing Skills in Novice Programmers. *SIGCSE Bulletin*, 36(4), 119–150.

McCracken, M., Kolikant, Y.B-D., Almstrum, V., Laxer, C., Diaz, D., Thomas, L., Guzdial, M., Utting, I., Hagan, D., Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin,* 33(4),125–140.

Mendes, A. J., Paquete, L., Cardoso, A., Gomes, A. (2012). Increasing student commitment in introductory programming learning. *Frontiers in Education Conference Proceedings*, 1–6.

Moström, J.E. (2011). *A Study of Student Problems in Learning to Program*. Department of Computing Science Umea University, Umea.

Obaidellah, U., Blascheck, T., Guarnera D.T., Maletic, J.I. (2020). A Fine-grained Assessment on Novice Programmers' Gaze Patterns on Pseudocode Problems. In: *Proceedings of the ACM Symposium on Eye Tracking Research and Applications (ETRA)*, Article No.: 56, 1–5.

Obaidellah, U., Al Haek, M., Cheng, P. C.-H. (2018). A survey on the usage of eye-tracking in computer programming. *ACM Computing Surveys*, 51(1), 1–58.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Devlin, M., Paterson, J. (2007). A survey of literature on the teaching of introductory programming. *SIGCSE Bulletin*, 39(4), 204–223.

Peterson, C.S., Saddler, J., Blascheck, T., Sharif, B. (2019). Visually Analyzing Students' Gaze on C++ Code Snippets. In: *Proceedings of the 6th International Workshop on Eye Movements in Programming* (EMIP 2019). IEEE, Piscataway, NJ, USA.

Ramsey, R., Atwood, M., Van Doren, J. (1993). Flowcharts versus program design languages: An experimental comparison. *Communications of the ACM*, 26(6), 445–449.

Rayner, K. (1998). Eye Movements in Reading and Information Processing: 20 Years of Research. *Psychological Bulletin*, 124(3), 372–422.

Robins, A., Rountree, J., Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, 13,137–172.

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.

Sharma, K., Jermann, P., Nüssli, M.A., Dillenbourg, P. (2011). Gaze Evidence of Different Activities of Program Understanding. In: *Proceedings of the 24th Psychology of Programming Interest Group Conference (PPIG 2011)*.

Scanlan, D.A. (1989). Structured Flowcharts Outperform Pseudocode: An Experimental Comparison. *IEEE Software*, 6(5), 28–36.

Sharif, B., Falcone, M., Maletic, J. I. (2012). An eye-tracking study on the role of scan time in finding source code defects. In: *Proceedings of Symposium on Eye-Tracking Research and Applications (ETRA)*, 381–384.

Shneiderman, B., Mayer R., McKay, D., Heller, P. (1977). Experimental investigations of the utility of detailed flowcharts in programming. *Communications of the ACM*, 20(6), 373–380.

Uwano, H., Nakamura, M., Monden, A., Matsumoto, K. I. (2006). Analyzing individual performance of source code review using reviewers' eye movement. In: *Proceedings of Symposium on Eye-Tracking Research and Applications (ETRA)*, 133–140.

M. Andrzejewska – doctor of pedagogical sciences, master in computer science, an assistant professor at the Institute of Computer Science and head of Department of Educational Research and New Media at Pedagogical University of Krakow. Her research interests are in the areas of learning and teaching programming, using eye tracking technique to study of cognitive processes in learning and problem solving.

P. Kotoniak – graduate student of computer science at Pedagogical University of Krakow. He has research interests within human-computer interaction and eye tracking methods. Every day he works as a full stack developer in Bravelab company.