Scratch and Google Blockly: How Girls' Programming Skills and Attitudes are Influenced

Mazyar Seraj[1,2] Eva-Sophie Katterfeldt[1] Kerstin Bub[1] Serge Autexier[2] Rolf Drechsler[1,2]
[1]Institute of Computer Science, University of Bremen, 28359 Bremen, Germany
[2]Cyber-Physical Systems, DFKI GmbH, 28359 Bremen, Germany
{seraj,evak,kerstin.bub,drechsler}@uni-bremen.de
{mazyar.seraj,serge.autexier,rolf.drechsler}@dfki.de

ABSTRACT

Block-based programming has become popular to teach programming to young students in introductory programming environments. Nevertheless, in most western countries, girls show a lack of interest in computer science, including programming. This paper presents the results of two user studies with 24 female German secondary school students in two programming workshops. We use and compare two environments based on Scratch and Google Blockly in fostering the students' programming skills and changing their attitudes towards programming. The two block-based programming editors have been chosen as they are popular in the current educational use of block-based programming. The results support the usage of Scratch-based environment in order to support the acquisition of programming skills. However, those students who used the Blockly-based environment showed greater interest in future programming learning opportunities. The contribution of this paper is showing the different impacts of Scratch and Google Blockly on young female students' interest in programming and the acquisition of programming skills in extra-curricular programming workshops.

CCS CONCEPTS

• **Human-centered computing → Visualization**; • **Computers and Education → Computer-assisted instruction**; • **Social and professional topics → Computer science education**;

KEYWORDS

block-based programming, young female students, programming workshop, acquisition of programming skills, attitudes towards programming

ACM Reference Format:
Mazyar Seraj[1,2] Eva-Sophie Katterfeldt[1] Kerstin Bub[1] Serge Autexier[2] Rolf Drechsler[1,2] . 2019. Scratch and Google Blockly: How Girls' Programming Skills and Attitudes are Influenced. In *19th Koli Calling International Conference on Computing Education Research (Koli Calling '19), November 21–24, 2019, Koli, Finland*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3364510.3364515

Koli Calling '19, November 21–24, 2019, Koli, Finland

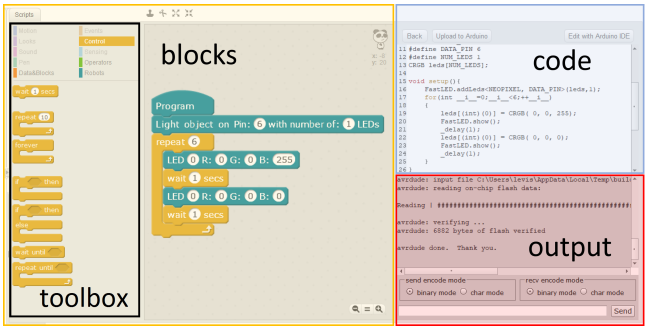
ACM ISBN 978-1-4503-7715-7/19/11…$15.00
https://doi.org/10.1145/3364510.3364515

1 INTRODUCTION

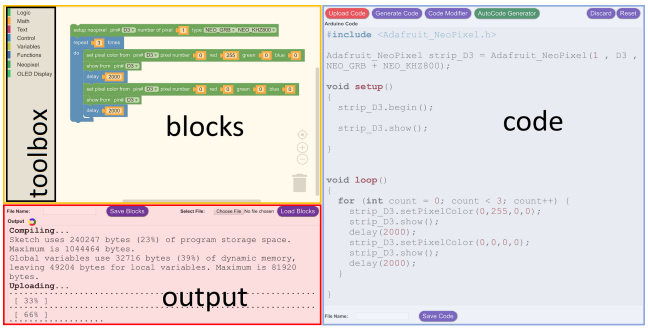
In recent years, the ability to program is increasingly becoming an important skill in our high-tech world [31, 33]. Therefore, many introductory programming environments are designed and developed in order to enable a wider range of audiences to start programming. Block-based programming is known as an approach that is widely used in the design of introductory programming environments. Using block-based programming allows inexperienced learners and novices to program through visual block-shaped programming elements such as variables, loops, conditional statements, logical operators, and functions. These elements can be dragged, dropped, and snapped together like puzzle pieces [10, 31]. Acknowledging that young students can be motivated through programming activities, the visual block-based approach is becoming significantly important for introducing basic programming concepts and eventually lead to develop an interest in computer science in general [11, 14, 31].

The low number of women compared to the high number of men in higher computer science education is a well-known problem in most western countries [5, 6, 13]. Fewer than 1 in 5 computer science graduates are women across 35 European countries [5]. Likewise, Ertl et al. [6] mentioned that approximately 25% of females are pursuing a career in STEM (Science, Technology, Engineering, Mathematics) in the EU, and this number is even lower in Germany with approximately 18% (where our study took place). The lack of computer science interest among girls or women has been shown to emerge partly from technological aspects of computer science [3, 12]. In this regard, block-based programming is introduced as an alternative to conventional text-based programming languages that appears less technical and brings new dimensions such as creativity to the understanding of programming among young students [26, 32, 34].

Since programming was taught to young students via block-based programming, many block-based programming editors such as Scratch [25], Alice [4], Snap! [9], and Google Blockly [7] have been used for introduction to programming in the context of computer science education [16, 18, 34]. Much work has been done investigating what type of editor is more beneficial for young students to have a better understanding of programming [1, 29, 31, 34]. Despite the fact that these editors are widely adapted in the design of introductory programming environments [10, 33], an open question remains about how well such programming editors can enable *young female students* to acquire basic programming skills and, at the same time, to improve their attitudes towards programming. More specifically, one challenge that computer science educators still face is the lack of studies investigating empirical evidence in using block-based programming editors among young female students. The empirical evidence is subject to describe under what circumstances a given block-based programming editor is the better



(a)



(b)

Figure 1: A view of the user interface for the (a) mBlock and (b) web-based programming application (WPA); both translated to English.

choice to foster interest and programming skills among the students in order to motivate them to join future computer science education.

In this paper, we compare two block-based programming environments (BBPEs) that are based on two popular block-based programming editors, dominating the current educational use of block-based programming [10]: MBlock [15] which is based on Scratch [25] (see Fig. 1a) and a web-based programming application (WPA) which is based on Google Blockly [7] (see Fig. 1b). We claim that these two BBPEs have different influences with respect to the students' attitudes towards programming and the acquisition of basic programming skills. Thus, this paper seeks to answer the following research question:

How do two widely used block-based programming editors – Scratch and Google Blockly – perform in order to foster young female students' programming skills and positive attitudes towards programming?

In order to examine the acceptance of introductory programming and the experience with the BBPEs among young female students, two user studies were conducted with 24 beginners (10 to 14 years old) in total. The students used the BBPEs to program a micro-controller (Arduino or WeMos boards) to control LED lights. An empirical quantitative evaluation of the two BBPEs with respect to the young female students' attitudes and perceptions of programming as well as their acquisition of basic programming skills, was conducted. Collecting and analyzing data in this study refined prior results that have shown the acceptance of programming among young students [18, 32] where they were able to learn programming via block-based programming.

2 BACKGROUND AND RELATED WORK

Over the past years, block-based programming has become a popular approach for the design of introductory programming environments for young learners. Block-based programming editors are designed to help young learners get started with programming with a low threshold [10, 19, 34]. Using block-based programming allows developers and educators to reduce the difficulty of initializing basic programming concepts (e.g., programming structure and principles) among young learners [1, 14, 29]. Block-based programming was used in a variety of studies (e.g., [1, 14, 16, 19, 23, 27, 31]) to facilitate the accessibility of learning programming, especially for young students and novices. Literature reports that block-based programming makes programming pleasant, engaging, and motivating for young students, and thus, it can leverage their interest in programming and computer science in general [29, 33, 34]. However, relatively little work has been done with a special focus on using block-based programming to promote basic programming activities among young female students in extra-curricular programming workshops.

The literature also reports that in most western countries the women's lack of interest in programming and experience with computers are two of the reasons to have a low number of women in computer science [12, 13]. In that respect, using block-based programming is reasonable to introduce young female students to programming and basic computational skills [1, 14]. In contrast to the studies which are questioning the suitability of block-based programming (e.g., [17, 22]) in order to motivate young students and prepare them for future learning programming opportunities, block-based programming is recommended as the first choice in introductory programming and computer science courses [19, 31]. Several studies show that the use of block-based programming in formal [32, 34] and non-formal [8, 29] educational context has a positive influence on young students' programming skills and interest. Studies also report successes of teaching programming concepts to young students and foster their interest and motivation in learning programming using block-based programming compared to text-based programming [27, 32, 34]. However, understanding the impacts of block-based programming in extra-curricular learning environments remains an active area of research, with a focus on how best to utilize BBPEs to foster young female students' programming skills and leverage their interest in programming and computer science.

With respect to teaching programming, two popular block-based programming editors – namely Scratch and Google Blockly – have made significant contributions to the current educational use of block-based programming [1, 10, 11, 30]. For instance, Scratch which is known as one of the most successful editors has been used to investigate the affordances of block-based programming in comparison with text-based programming environments in order to teach basic programming concepts to inexperienced high school students [32, 34]. As a result, using block-based programming, the majority of students gained more programming knowledge, had a better perception of programming, and they were more interested in continuing programming in the future. Furthermore, according to [16, 20], researchers are using Google Blockly to encourage young students to start programming robots [20] and microcontrollers [16]. Paramasivam et al. [20] took advantage of "*CustomPrograms*" which is based on Google Blockly to design a BBPE in order to enable young students with disabilities (e.g., Attention Deficit Disorder, Asperger's Syndrome, and other autism spectrum

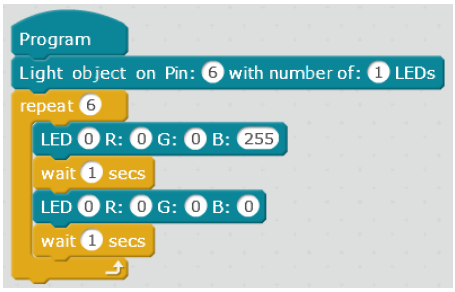
disorders or learning disabilities) to program Clearpath Turtlebot. Likewise, Martínez et al. [16] took advantage of "*BlocklyDuino*" to design a BBPE in order to enable preschool and elementary school children to program and control the behavior of Arduino boards. The results reported by these studies are encouraging, as they were successful in establishing confidence among young students that programming is interesting. Nevertheless, how to best support young female students by BBPEs to develop basic programming skills and encourage them for being interested in programming is still a growing research area, where the open question remains.

In contrast to a large number of previous studies, we seek to investigate the differences between two widely used block-based programming editors: Scratch and Google Blockly in terms of acquisition of basic programming skills and improving young female students' attitudes towards programming. Similarly, we aim to foster programming skills and interest among the students in order to increase their understanding of the programming side of computer science and motivate them to take part in future computer science education and digital society. To this end, we set up two extra-curricular programming workshops independent of the students' regular curriculum and outside their schools. We aim to provide opportunities for young female students in order to begin with basic programming activities and program micro-controllers to control LED lights using two BBPEs which are designed based on Scratch and Google Blockly.

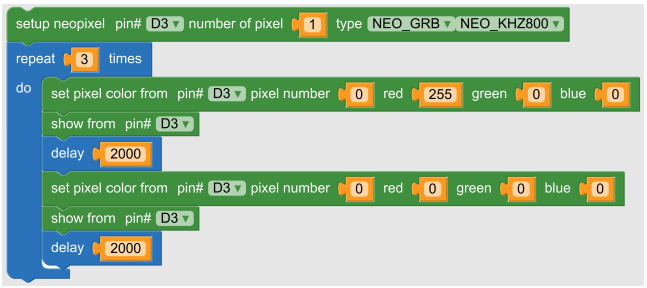
3 OVERVIEW OF THE BLOCK-BASED PROGRAMMING ENVIRONMENTS

In this paper, two BBPEs were utilized to enable young female students to learn and create programs for micro-controllers (Arduino or WeMos board) to control LED lights. In that respect, mBlock (see Fig. 1a) and the WPA (see Fig. 1b) were used in two extra-curricular programming workshops. MBlock was used together with Arduino boards, while the WPA was used together with WeMos boards. Different programming language features like variables, data types, loops, conditional statements, functions, and operators are included as block-shaped elements in both programming environments. Students author programs in these environments by dragging-and-dropping blocks. In addition to the blocks representing programming features, primitive Arduino behavior is wrapped in a set of predefined blocks. Students create programs simply by snapping blocks together. The Arduino code is generated from the blocks in the background (visible in another panel), ready for execution. The possibility of zooming in and out on blocks is given, meaning that the scale of blocks can be changed by the mouse scroll wheel or the zoom gesture on a track-pad. This enables students to see the whole sequence of blocks if needed. Here, an overview of the design of both programming environments is provided.

MBlock is based on Scratch [25], and it is primarily designed for inexperienced learners and children to learn and write programs for micro-controllers and robots. In micro-controller mode (Arduino mode), mBlock allows young students to use a visual block-based interface that comprises a full vision of the blocks (Block Panel), block categories (Toolbox), code syntax (Code Panel), and output of the code (Output Panel). Students can track the compile and upload process of the code into the micro-controller as well as the errors and Serial Monitor output in the Output Panel. In addition to blocks representing the programming features, other blocks are designed and developed to enable the students to work with LED lights, which we discuss in the following. The starting block is



(a)



(b)

Figure 2: A sample of execution blocks for the (a) mBlock, and (b) WPA; both translated to English.

labeled as `program` that always needs to be the first block. The code syntax which is nested in other blocks only appears in the Code Panel when they are connected to the `program` block. Furthermore, the students can define how many LED lights are connected to a pin via the `light object` block. It is labeled as `light object on Pin <X> with number of <X> LEDs`. The two `<X>` are input field numbers, and they are filled in with default arguments to support the understanding of the block for the students. The color of LED lights can be changed through the `LED` block that is labeled as `LED <X> R <X> G <X> B <X>`. Similarly, the four `<X>` are input field numbers, and they are filled in with default arguments. The LED number filled in with 0, which always refers to the first LED, and the color is set to be white as default. This block also includes the show LED command in order to colorize the LED light (see Fig. 2a).

The WPA is based on BEESM [28], which is primarily designed as an educational block-based programming tool for inexperienced users and novices. It is built with the Blockly library [7], and it enables users to learn and create programs for smart environments, micro-controllers, and mobile robots one at a time and in combination with each other. The design and additional features of BEESM can be found in greater detail in [28]. For this study, we manipulated the BEESM user interface to enable our target students to have a full vision of the blocks (Block Panel), block categories (Toolbox), code syntax (Code Panel), and output of the code (Output Panel). Similar to mBlock, the Output Panel shows the compile and upload process of the code into the micro-controller as well as all return values and errors for debugging purposes. To enable the students to work with LED lights, we designed and developed other blocks in addition to blocks that represent the programming features which are discussed as follows. The first block which is needed in order to define how many LED lights are connected to a pin is called `setup neopixel`. It is labeled as `setup neopixel pin# <X> number of pixel <X>`. The color of LED lights can be changed through the `set pixel color` block which is labeled as `set pixel color from`

`pin# <X> pixel number <X> red <X> green <X> blue <X>`. Similar to the mBlock, all `<X>` values are input field numbers which are filled in with default arguments to support the understanding of the block for the students. The pixel number filled in with 0, which always refers to the first LED, and the color is set to be white as default. Furthermore, in order to colorize the LED light, the show LED command is encapsulated into another block, which is called `show color`, and it is labeled as `show from pin# <X>`. The pin number is always filled in with 1, which refers to the first pin in micro-controllers (see Fig. 2b).

For a usability analysis, see Holwerda and Hermans [10] for a discussion on the differences between the two popular block-based programming editors: Scratch and Blockly. This usability analysis aimed to identify generic aspects of their user interface, and if they effectively fulfill their purpose to facilitate programming for young learners and novices. In this respect, the authors mentioned that a larger section for blocks could improve the visibility of finding and reading blocks in the program. However, the Blockly-based tool which is used (ArduBlockly [21]) does not support zooming via mouse scroll wheel or the zoom gesture on a track-pad, and it is only possible through the zooming buttons in Block Panel. It is also addressed that dragging a block out of a sequence of blocks will move all other blocks below with it in both editors. This will make manipulating of code structure more difficult for novices. Furthermore, it is suggested to have a search option to enable users looking and finding the right block in both Blockly and Scratch. However, the authors discuss that additional editor features may clutter the interface (both visually and cognitively) for adult novices and more specifically, for young learners in school. The main remaining differences between our two BBPEs are (see also Fig. 1 and Fig. 2) as follows:

(1) *The Block Panel:* the WPA contains a smaller Block Panel and larger Code Panel than mBlock.
(2) *The Code Panel:* the WPA enables students to modify the code that generates from the blocks directly in the Code Panel while in mBlock they need to open the code in Arduino IDE in order to modify it.
(3) *The Toolbox:* the Block Panel in the WPA contains the Toolbox, like a menu, that displays different categories for blocks. A set of blocks within a category is displayed temporary when students click on the category, while in mBlock, blocks within a category are displayed lasting when they click on the category.
(4) *How the blocks are shaped:* the structure of blocks can be changed using a pop-up panel (e.g., adding an `else-if` to an `if` block) in the WPA while we do not have this feature in mBlock.
(5) *How text codes are encapsulated in different blocks:* for instance, students need to use a `program` block that includes all libraries to start the program in mBlock, while in the WPA, necessary libraries are included in the corresponding blocks. Furthermore, in the WPA, a `display` block is needed in order to colorize the LED light, while in mBlock, it is nested in an *LED* block.

4 METHODS

The data presented in this paper are part of a larger study using blocks-based programming and tangible smart objects in extra-curricular programming workshops in order to foster young female students' programming skills and positive attitudes towards programming in the north of Germany. In this respect, the students are welcomed to different research centers to learn programming in extra-curricular programming workshops that are outside their school environments and not part of the regular school curriculum. In order to understand the impact of the two block-based programming editors – namely Scratch and Google Blockly – on young female students' programming skills and their attitudes and perceptions of programming, we conducted two user studies comparing mBlock to the WPA. This section begins with the study design and strategy for collecting and analyzing data, then information about the participants is presented; this section concludes with the procedure of the study.

4.1 Study Design and Data Collection Strategy

In this study, we use and compare two BBPEs in two extra-curricular programming workshops with two groups of young female students (i.e., aged between 10 and 14 years old). The workshops were held in the premises of the University of Bremen. All used equipment – computers, micro-controllers, and LED lights – were provided by the German Research Center for Artificial Intelligence (DFKI), and the group of Cognitive Neuroinformatics (CNI). The BBPEs enable students to focus on "*Programming Structures and Principles*", the main concept which was taught and exemplified through the BBPEs. In general, 24 students attended the two programming workshops (12 students each). In one workshop, 12 students used mBlock (mBlock-group), and the WPA was used in the second workshop (app-group) by the other 12 students.

In both programming workshops, a pre- and post-questionnaire was used in order to collect data with respect to the young female students' attitudes and perceptions of programming, their prior programming experience, and their age group. The acquisition of basic programming skills among the students was assessed, using a pre- and a post-programming question.

In the following, we describe the pre- and post-questionnaire as well as the pre- and post-programming questions in both programming workshops.

Pre-questionnaire. In each workshop, students received a pre-questionnaire. Four attitudinal questions were asked in order to find out the students' attitudes and perceptions of programming. These questions are based on the attitudinal questions which were used in Weintrop and Wilensky [34] and were adapted for the needs of this study. Students' confidence, enjoyment, perceived difficulty, and interest in future programming learning opportunities were evaluated using these questions. In that respect, students were asked to rate the questions "*do you think you are good at programming?*", "*do you think programming is fun?*", "*do you think programming is difficult to understand?*", and "*would you like to learn how to program?*", using a 5-point Likert scale (with 1 "*no, not at all*", 5 "*yes, very much*", and 0 "*I do not know*"). Furthermore, they were asked to determine their prior programming experience with BBPEs using the "*yes*" or "*no*" question "*have you ever worked with a block-based programming environment?*". Then, we asked them to indicate whether they can program on a scale of 1 to 5, with 1 "*no, not at all*", and 5 "*yes, very good*", using the question "*can you program?*".

Post-questionnaire. At the end of each workshop, students took the post-questionnaire. It was composed of the same attitudinal questions as the pre-questionnaire, just with different words for two questions; "*do you think programming is difficult to understand?*" changed to "*do you think programming is difficult?*", and "*would you*

like to learn how to program?" changed to "*would you like to learn better how to program?*".

In addition to the attitudinal questions, five questions were asked in order to measure the students' experiences with the two BBPEs in terms of their ease-of-use. The Students were required to rate the question "*I think the programming environment is easy to use.*", using a 5-point Likert scale (with 1 "*strongly disagree*", 5 "*strongly agree*", and 0 "*I do not know*"). The question "*do you find it easy to program with blocks?*" was also asked, using a 5-point Likert scale (with 1 "*no, not at all*", 5 "*yes, very much*", and 0 "*I do not know*"). They were then asked to rate (i) if they paid attention to the code that is generated matching the blocks, (ii) if they think the function "edit code" is helpful to better understand their program, and (iii) if they find the "output" panel helpful to understand their program. The scores for these three questions were calculated, using a 5-point Likert scale (with 1 "*no*", 5 "*yes*", and 0 "*I do not know*").

Two additional questions were added to the post-questionnaire for this study. The students were asked, "*do you think it's helpful if you program a real object? E.g. the LED light and the micro-controller*", to be answered on a 5-point Likert scale (with 1 "*no, not at all*", 5 "*yes, very much*", and 0 "*I do not know*"), and they were asked about their preference of programming with blocks or direct with code syntax, using a 5-point Likert scale (with 1 "*direct with code*", 5 "*with blocks*", and 0 "*I do not know*").

Programming questions. To validate the students' answers with respect to their prior programming experience and to analyze the acquisition of basic programming skills, in both workshops, students were asked to complete two programming questions. In this respect, a pre-programming question right after the pre-questionnaire, and a post-programming question right after the post-questionnaire were completed. The programming concepts are extended by introducing these programming questions to the students. In each pre- and post-programming question, block-shaped elements were designed independent of the two BBPEs in order to test how well the students acquire the basic programming skills which were taught during the programming workshop. For instance, see Fig. 3a, and Fig. 3b for the block-shaped elements in pre- and post-programming questions in the app- and mBlock-group, respectively. However, in each workshop, one block (block number 13) was designed similar to what the students saw and used in the corresponding programming environment. In this regard, students needed to use a `Program` block in order to start the program in mBlock, and a `display` block in order to colorize the LED light in the WPA (see Fig. 3).

The pre-programming question in the app-group was to program the micro-controller to make one LED light blink in red for 3 times with 2 seconds delay in between when the light is connected to the micro-controller on *Pin D3*. The post-programming question in the app-group was to program the micro-controller to make one LED light blink in blue for 6 times with 1 second delay in between when the light is connected to the micro-controller on *Pin D3*. In the mBlock-group, apart from the pin number, which is *Pin 6*, similar pre- and post- programming questions were asked from the students. In each programming question, students were asked to select a set of blocks and identify the order of them in a correct logical way based on the question. Students were also notified that some blocks might appear more than once and some may not even be needed in their program. The pre- and post-programming questions are slightly different from each other in both workshops. This counter-balance design of questions ensures that students read the questions

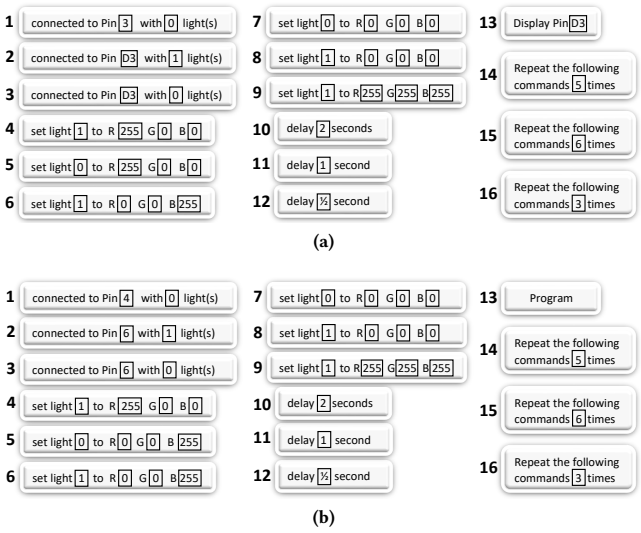


Figure 3: Block-shaped elements in (a) pre-programming question in the app-group; (b) post-programming question in the mBlock-group; both translated to English.

carefully and identify the order of blocks based on the question. Furthermore, these questions represent realistic programming problems for a micro-controller and an LED light, as colorizing the light is core to the function and use of micro-controller together with one LED light.

For each programming question, we collected the solution made by the student using the blocks. A 10-point grading rubric [24] was created to evaluate the students' performance (see Table 1). Each solution was scored by two researchers in order to ensure consistent grading.

4.2 Participants

Two user studies were conducted with 24 young female students without any prior programming experience (10 to 14 years old) from several German secondary schools. The schools' headmasters and teachers were contacted and informed about our programming workshops. Students and their parents were then announced by their school to register for one workshop, meaning the students who participated in this study were self-selected and were interested in having programming activities and learning programming.

A total of 12 female students participated in each programming workshop. In the mBlock-group, the average age of the participants was 12.67 years ($SD = 0.78$). In the app-group, the average age of the participants was 12.58 years ($SD = 1.24$). Although participants in the app-group are younger than participants in the mBlock-group, an analysis of variance (ANOVA) showed no significant difference between the groups, $F < 1$.

With respect to the students' prior programming experience, all students indicated that they have no experience in working with any BBPEs. Furthermore, when the students were required to rate whether they can program or not, only two of them in the app-group answered, "*no*" and the rest of the students answered, "*no, not at all*". Nevertheless, no significant difference was observed ($F(1, 22) = 2.20$, $p = 0.15$). Thus, the level of prior experience was not included in our analyses.

Table 1: 10-point Grading Rubric Scale

10	8	6	4	2	0
Connection	**Analysis**	**Summary**	**Incomplete**	**Attempted**	**Not Attempted**
Answer shows mastery of content and a deeper understanding of it.	Answer shows mastery and understanding of content, but it has a minor issue.	Answer shows some understanding of essential content, but it has a lack of greater evidence.	Answer does not show understanding of basic content, or it shows that mastery of the general content is missing.	Answer does not address the programming question or is off-topic.	Student did not attempt to solve the programming question.

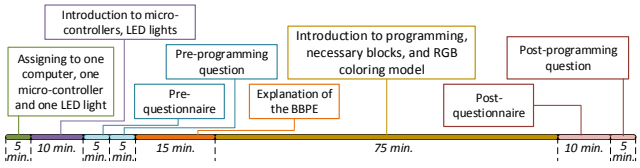


Figure 4: Procedure of the programming workshop.

4.3 Procedure

In both programming workshops, we followed the same procedure (see Fig. 4). The duration of each workshop was 130 minutes. At the beginning of both workshops, each participant was assigned to one computer, one micro-controller (Arduino or WeMos board) and one LED light in order to minimize the distraction of participants. Each computer had an installed version of the corresponding programming environment. In the mBlock-group, Arduino boards were used together with mBlock, and in the app-group, WeMos boards were used together with the WPA.

All participants were introduced to micro-controllers, LED lights, and they were shown how to connect them for 10 minutes. Afterward, the participants received the pre-questionnaire; they were asked to determine their prior programming experience and rate their perception of programming using the four attitudinal questions. Each participant was then asked to complete the pre-programming question, which required them to select a set of blocks and write their numbers in a correct logical way based on the question. The participants were then trained according to the interface of each BBPE, different panels, buttons, and their functionalities for 15 minutes. Then, during the allocated time of 75 minutes, all participants were introduced to general features of programming (e.g., variables and loops), necessary blocks to control an LED light, and the RGB coloring model. One female instructor led each workshop. In this regard, in each workshop, an oral explanation was given, using prepared slides based on each BBPE. Additionally, we used supplementary documents – including an explanation of necessary blocks and RGB coloring model – in order to help our students to work with the corresponding BBPE. This also allows us to minimize and control the instructor effects. Both female instructors have a computer science background and experience in working with young students. During the allocated time, participants were enabled to program their micro-controller, using the corresponding BBPE in three learning steps. These steps respectively were (i) colorizing one LED light with either red, green or blue color, (ii) colorizing one LED light with two arbitrary colors and write down the correct value of red, green and blue colors, and (iii) letting one LED light blink for a random number of times and seconds delay in between. The instructor helped participants at their desk during the 75 minutes to ensure that the participants had faced no major problems. At the end of each workshop, the post-questionnaire was given to the participants in order to ask them to rate their perception of programming and to rate their experience with the corresponding BBPEs. Each workshop ended with the post-programming question.

5 RESULTS

The results section is divided into three parts. First, results from an analysis of the programming part of the study are presented, reporting the students' performance on each pre- and post-programming question. Second, results from an analysis of the pre- and post-questionnaires are presented, looking at young female students' confidence, enjoyment, perceived difficulty, and interest in future programming learning opportunities. Finally, we report on results from an analysis of the post-questionnaire, focusing on students' experience with respect to ease-of-use of corresponding BBEPs. Additionally, in this part, students' preferences of programming with blocks or direct code, as well as their thought of being able to see the impacts of their programs on a real object is reported.

The following analyses were computed as one-factor analysis of variance (ANOVA), with the factor *mBlock vs. app*. Paired-samples t-test was also used to show the differences within each group of students from the beginning towards the end of each workshop.

5.1 Acquisition of Programming Skills

To understand how students' programming skills are influenced by the two BBPEs (mBlock and the WPA) in our extra-curricular programming workshop, students' performance on the pre- and post-programming questions were analyzed.

With respect to the programming questions (see Fig. 5), in the pre-programming question the students in mBlock-group performed better than the students in app-group, $M = 2.00$, $SD = 0.00$, and $M = 1.33$, $SD = 1.30$, respectively. However, no significant difference occurred, $F(1, 22) = 3.14$, $p = 0.09$. In the post-programming question, an ANOVA showed that the students in mBlock-group ($M = 6.83$, $SD = 1.80$) performed significantly better than the students in app-group ($M = 3.17$, $SD = 1.59$), $F(1, 22) = 28.02$, $p < 0.001$. Descriptively, although the students in the mBlock-group indicated a higher level of performance before working with the programming environment, they performed significantly better after working with the programming environment compared to the students who were in the app-group.

Focusing on the average performance of the students in pre- and post-programming questions shows that their performance increased in both mBlock-group and app-group. A paired-samples t-test showed that in mBlock-group, students performed significantly better in the post-programming question than in the pre-programming question, $t(12) = 9.30$, $p < 0.001$, $MD = 4.83$. Similarly, in app-group, students' performance significantly increased in the post-programming question compared to pre-programming question, $t(12) = 4.75$, $p = 0.001$, $MD = 1.83$.

5.2 Attitudes and Perceptions of Programming

To understand how students' attitudes and perceptions of programming are affected by the two BBPEs (mBlock and the WPA) in our extra-curricular programming workshop, we analyzed scores from the pre- and post-questionnaires. All answers were coded with 1 "*no, not at all*", 5 "*yes, very much*", and 0 "*I do not know*". Please note

Table 2: Students' Attitudes and Perceptions of Programming

Questions	Pre-questionnaire *M (SD)*	*ANOVA Results*	Post-questionnaire *M (SD)*	*ANOVA Results*
Confidence in mBlock-group	2.40 (0.55)	F <1	3.63 (0.74)	F(1,17) = 3.16, p = 0.09
Confidence in app-group	2.60 (1.52)		2.73 (1.27)	
Enjoyment in mBlock-group	4.64 (0.50)	F(1,20) = 1.87, p = 0.19	4.58 (0.51)	F(1,22) = 1.38, p = 0.25
Enjoyment in app-group	4.18 (0.98)		4.17 (1.11)	
Difficulty in mBlock-group	3.30 (0.67)	F(1,19) = 3.54, p = 0.08	2.33 (1.15)	F(1,22) = 2.70, p = 0.12
Difficulty in app-group	3.91 (0.83)		3.08 (1.08)	
Interest in mBlock-group	4.75 (0.44)	F(1,22) = 1.16, p = 0.29	4.33 (0.49)	F(1,21) = 1.54, p = 0.23
Interest in app-group	4.92 (0.29)		4.64 (0.67)	

M: Mean **SD**: Standard Deviation **F**: F-distribution **p**: p-value

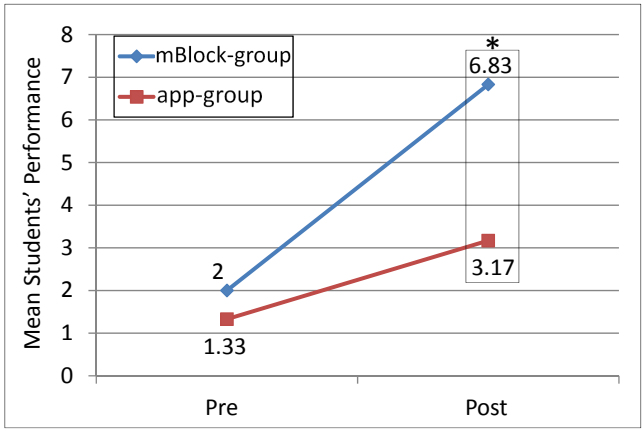


Figure 5: Students' performance on the pre- and post-programming questions.

that students who responded "*I do not know*", are not included in our analysis.

Concerning the students' confidence (see Table 2), we assessed their responses to the following question in the pre- and post-questionnaires: "*do you think you are good at programming?*". Seven students in each app- and mBlock-group responded, "*I do not know*" in the pre-questionnaire. In the post-questionnaire, this number decreased to one in the app-group and four in the mBlock-group. In the pre-questionnaire, students in the app-group indicated a slightly higher level of confidence compared to the students in the mBlock-group. However, in the post-questionnaire, students in the mBlock-group indicated (descriptively) a higher level of confidence compared to the students in the app-group. In both mBlock-group and app-group, the level of confidence in programming is higher for students in the pre-questionnaire compared to the post-questionnaire. With respect to the students who responded, "*I do not know*", we focus on the changes in their level of confidence between the pre- and post-questionnaire. In the app-group, six out of seven students had an idea in the post-questionnaire, and in general, they indicated a medium level of confidence, $M = 3.33$, $SD = 1.21$. Similarly, in the mBlock-group, four out of seven students had an idea in the post questionnaire, and in general, they indicated a higher level of confidence, $M = 4.00$, $SD = 0.82$. However, no significant difference was observed, using an ANOVA ($F < 1$).

With respect to the enjoyment of programming (see Table 2), the responses to the following question in the pre- and post-questionnaires were assessed: "*do you think programming is fun?*". Only one student in the app-group and one student in the mBlock-group responded "*I do not know*" in the pre-questionnaire. In both pre- and post-questionnaire, the level of enjoyment of programming is slightly higher for the students in the mBlock-group compared to the students in the app-group.

The third attitudinal question is whether the students think programming is difficult or not (see Table 2). To calculate a measure of the difficulty of programming, students responded to the following question: "*do you think programming is difficult to understand?*" (or "*do you think programming is difficult?*" in the post-questionnaire). Only one student in the app-group and two students in the mBlock-group responded "*I do not know*" in the pre-questionnaire. Descriptively, in the pre-questionnaire, the programming difficulty level is higher for students in both mBlock-group and app-group. Furthermore, in both pre- and post-questionnaire, the programming difficulty level is higher among students in the app-group. We focus on the score of difficulty of programming within the app- and mBlock-group, among the students who answered the question in both pre- and post-questionnaires. A paired-samples t-test showed that in the mBlock-group, students found programming significantly less difficult in post-questionnaire in comparison with pre-questionnaire, $t(9) = -2.70$, $p = 0.024$, $MD = -1.10$. However, no significant results occurred in app-group, $t(10) = -1.70$, $p = 0.12$, $MD = -0.82$.

The last attitudinal question is whether the students' interest in learning programming in the future is affected or not (see Table 2). In order to calculate a measure of it, students responded to the following question: "*would you like to learn how to program?*" (or "*would you like to learn better how to program?*" in the post-questionnaire). Only one student in the app-group responded, "*I do not know*" in the post-questionnaire. Descriptively, in both pre- and post-questionnaire, students in the app-group indicated a higher level of interest compared to students in the mBlock-group. Furthermore, in both mBlock-group and app-group, students showed a higher level of interest in the pre-questionnaire compared to the post-questionnaire. Focusing on students' interest within the app- and mBlock-group, a paired-samples t-test showed that the students' willingness to learn programming dropped in mBlock-group towards the end of the workshop; accordingly, it was just barely missed the level of significance, $t(10) = -2.16$, $p = 0.054$,

$MD = -0.42$. However, the test showed that the decrease in students' interest in the app-group was not significant towards the end of the workshop.

5.3 Programming Experience

The post-questionnaire included five questions asking students to reflect on how they perceive the ease-of-use of corresponding BBPEs (see Table 3). Furthermore, students were required to answer another question, concerning their preference for programming with blocks or with code. Finally, they were asked to indicate whether being able to see the impacts of their program on a real object is helpful or not. Please note that students who responded "*I do not know*", are not included in our analysis.

With respect to the ease-of-use of the two BBPEs, on average, the students showed broad approval to mBlock, while the students were undecided about the WPA. Accordingly, an ANOVA yielded a significant result (see Table 3). Only one student in the app-group responded, "*I do not know*" to this question. With respect to finding programming easy with blocks, on average, the students found programming with blocks significantly easier in mBlock compared to the students in the WPA (see Table 3). Similarly, one student in the app-group responded, "*I do not know*" to this question. Furthermore, the students were undecided and indicated a low level (especially in mBlock-group) of paying attention to the code which was generated matching the blocks. Six students in the mBlock-group and nine students in the app-group responded to this question. The students found that being able to use the "edit code" function and to see error messages in the Output Panel is helpful to understand their own program (see Table 3). However, only 11 students answered to each of these two questions. Seven students in the mBlock-group and four students in the app-group responded to the question with respect to the helpfulness of being able to use the "edit code" function. Six students in the mBlock-group and five students in the app-group responded to the question regarding the helpfulness of being able to see error messages in the Output Panel. Although the students in the app-group rated all the three questions higher than the students in the mBlock-group, no significant results occurred, all $F < 1$.

Concerning the preference for programming with blocks or directly with code, on average, the majority of students indicated an opinion strongly towards programming with blocks. The students in mBlock-group indicated a slightly higher preference toward blocks ($M = 4.92$, $SD = 0.29$) compared to the students in app-group ($M = 4.44$, $SD = 0.88$). No significant difference were obtained, using and ANOVA ($F(1, 19) = 3.05$, $p = 0.10$). Only three students in the app-group responded, "*I do not know*" to this question, and thus, they are excluded from our analysis.

With respect to the question of whether or not students think it is helpful to program a real object (e.g., LED light and microcontroller), they found that being able to see the impacts of their program on a real object is helpful. In this regard, students in the app-group indicated the higher level ($M = 4.55$, $SD = 0.52$) compared to students in the mBlock-group ($M = 3.73$, $SD = 1.27$). However, an ANOVA shows that it just barely missed the level of significance, $F(1, 20) = 3,89$, $p = 0.062$. Only two students (one student in each group) responded "*I do not know*" to this question, and thus, they are excluded from our analysis.

6 DISCUSSION

We now discuss the results which were presented in the previous section. In this respect, we begin with our findings based on the research question presented in Section 1. Then, we review the limitations and discuss possible future works that should be taken into account.

6.1 Findings

One of the main contributions of this study is showing the potential of using two popular block-based programming editors (Scratch and Google Blockly) in extra-curricular programming workshops in order to support the acquisition of basic programming skills among young female students. In this respect, one hypothesis we had in this study was that different BBPEs (mBlock and the WPA) have different influence on young female students' performance who have no prior programming experience. Results show that in both mBlock-group and app-group, the performance of students was significantly higher in the post-programming question compared with the pre-programming question. This supports the results from prior research that showed by designing introductory programming environments based on Block-based programming, we could help young students to have better performance [18, 32]. Results also show that in both pre- and post-programming questions, the performance of students who worked with mBlock (which is based on Scratch) was higher than those who used the WPA (which is based on Google Blockly). Having a closer look into the performance on post-programming question reveals that in mBlock-group, students highly tended to solve the programming question and their performance highly improved in comparison to the students in app-group. This finding supports the idea that designing introductory programming environments based on Scratch could help young female students to gain basic programming skills, in particular, when they are indeed new to programming.

This study also reports on young female students' attitudes and perceptions of programming, where the findings were less clear. The results show that before and after the workshop, students who used mBlock indicated (descriptively) a higher score for the enjoyment of programming. Likewise, students in the mBlock-group showed a higher level of confidence in programming after the workshop, while it was slightly lower before the workshop compared with the students in the app-group. Furthermore, students in the app-group indicated a higher level of interest in taking part in the future programming opportunities before and after the workshop compared with the students in mBlock-group. In the mBlock-group, the students' level of interest was decreased after the workshop compared with before the workshop. With respect to the difficulty of programming, our findings show that the difficulty level of programming was higher before the workshop compared with after the workshop, particularly, in mBlock-group. Our findings also show that in the app-group, students found (descriptively) programming harder both before and after the workshop.

With respect to the ease-of-use of the two BBPEs, the results showed that the students in mBlock-group found blocks significantly easier to program compared with the students in app-group. Furthermore, when asked to indicate how easy was the use of corresponding BBPEs, the students in mBlock-group found mBlock significantly easier to use in comparison to the students in app-group who used the WPA. This result is in line with results from [10] that students found programming easier with Scratch-based environments, as the visibility of finding and reading blocks is higher than Blockly-based environments. In contrast, when the students asked specific questions about the demonstration of error messages

Table 3: Students' Experiences of Using the Block-based Programming Environments

Questions	mBlock-group *M (SD)*	app-group *M (SD)*	*ANOVA Results*
I think the programming environment is easy to use. [1 = "*strongly disagree*", and 5 = "*strongly agree*"]	4.58 (0.67)	3.82 (0.98)	F(1,21) = 4.85, **p = 0.039
Do you find it easy to program with blocks? [1 = "*no, not at all*", 5 = "*yes, very much*"]	4.00 (0.74)	3.18 (0.75)	F(1,21) = 6.93, **p = 0.016
Did you pay attention to the code that is generated matching the blocks? [1 = "*no*", and 5 = "*yes*"]	2.83 (2.04)	3.22 (1.56)	F <1
Do you think that the function "edit code" is helpful to better understand your program? [1 = "*no*", and 5 = "*yes*"]	3.43 (0.79)	3.75 (1.26)	F <1
Do you find the error messages in the "output" panel helpful? [1 = "*no*", and 5 = "*yes*"]	3.33 (0.52)	3.40 (0.89)	F <1

M: Mean **SD**: Standard Deviation **F**: F-distribution **p**: p-value **$^{**}p < 0.05$**: Significant Difference

in the "output" panel, usage of "edit code" function, and paying attention to the generated code syntax matching the blocks, students in app-group rated them (descriptively) higher than the students in mBlock-group.

Concerning the subjective questionnaire data, in line with findings from [34], the students largely prefer working with blocks compared to programming code syntax. Additionally, they found it helpful to code and see the impacts of their program in a real object. This is in line with findings from [2, 18] that showed real objects could stimulate students' interest, and motivate them to begin with programming activities.

All in all, the findings show that young female students who used a BBPE based on Scratch (in this case, mBlock) performed better on programming questions and showed a higher level of ease-of-use in programming with blocks in mBlock. This suggests a Scratch-based design for a productive environment for supporting female students who have no prior programming experience to gain basic programming and computational skills. At the same time, the findings that the students using a BBPE based on Google Blockly (in this case, the WPA) show a higher level of interest in taking future programming opportunities. This indicates a gap between what the students view themselves in programming with different types of BBPEs and the future programming experience with these environments.

6.2 Limitations and Future Work

While we tried to make the conditions across the two extra-curricular programming workshops as similar as possible, there were some differences which can be introduced as limitations of this study. For instance, we used mBlock together with Arduino boards, but due to technical reasons, the WPA is used together with WeMos boards. Thus, it introduces a difference that may influence the findings of this paper. However, there was no evidence that this difference has contributed to significantly differing our target students' experiences and change their level of acceptance for programming.

Another limitation of this study is related to the number of programming tasks and period of each workshop. For example, findings of this study are limited to the diversity of the programming tasks that young female students were required to perform to a larger scale of programming activities and computational skills. Using the micro-controllers and LED lights that are used in this study, students can perform larger and more complicated tasks. For example, they can use more LED lights and sensors to create colorful and animated LED picture frames. While we intend on introducing young female students who have no prior programming experience to begin with programming activities in a extra-curricular learning environment, this is relatively narrow functionality for a micro-controller, and thus, for the programming tasks. In this regard, there is still work to be done to verify the outcome of this study when the programming workshop is longer (including more number of training sessions) and when programming tasks become more diverse and complicated among the students with and without prior programming experience. Furthermore, we explored how Scratch and Google Blockly perform to foster young female students' programming skills and leverage their interest in programming. However, there is still an open question we would like to explore in the future: what are the reasons behind the students' preferences for using Scratch-based environments, and for using Blockly-based environments.

The final limitation of this study relates to the students' prior programming experience, socio-economic status, age, and the number of participants. In that respect, we would like to emphasize that our results might be affected by a lack of geographic and socio-economic diversity of students. Furthermore, our sample includes 24 female students without any prior programming experience (age between 10 to 14), which is a relatively small sample size to generalize findings of this study to a larger scale. Thus, we look at these as a major concern, and we seek to address them in future iterations of this work. A second similar limitation is related to the control group. This is another avenue of future work to find out the impacts of visual block-based programming environments on students' attitudes and programming skills when young male students are targeted for such extra-curricular programming workshops.

7 CONCLUSION

As the number of women in higher computer science education and society is lower than the number of men in most western countries, using block-based programming is an active area of research to make the programming side of computer science more interesting and engaging for girls. In this paper, we presented a extra-curricular programming workshop and a comparative study of how different types of BBPEs impact young female students' attitudes towards programming and their acquisition of programming skills. In that

respect, we explored how young female students use mBlock and a WPA which are based on Scratch and Google Blockly, respectively. Our findings indicate how young female students' performance, attitudes, and perceptions of programming can be affected by different types of BBPEs. This finding supports the idea of using Scratch in introductory programming environments in order to motivate young female students, in particular, those without prior programming experience to solve programming problems and gain basic programming skills. Furthermore, it shows that those students who used the WPA, which is based on Google Blockly indicated greater interest in future programming learning opportunities. Thus, it supports the claim that different BBPEs have a direct impact on young female students' performance and their attitudes towards programming. By studying under what condition, what type of BBPE has a better influence on young female students with different level of prior knowledge, we enhance our understanding to design introductory programming environments for them.

Given the decreasing presence of women in computer science society in most western countries, findings from this study are essential to ensure we are providing exposure to programming activities, preparing female students for future learning programming opportunities and motivating them to join the computer science society in the future. While many questions still remain on how to best introduce programming to female students, the findings of this study can help to inform other researchers and educators about the relationship between BBPEs, programming activities in extra-curricular learning environments and young female students' experience and acceptance of programming.

ACKNOWLEDGMENTS

This work was funded by the German Federal Ministry for Education and Research (BMBF) within the project SMILE under grant number 01FP1613. The authors would like to thank for this support.

REFERENCES

[1] Efthimia Aivaloglou and Felienne Hermans. 2016. How kids code and how we know: An exploratory study on the Scratch repository. In *Proceedings of the 2016 ACM Conference on International Computing Education Research*. ACM, 53–61.

[2] Sally R Beisser. 2005. An examination of gender differences in elementary constructionist classrooms using Lego/Logo instruction. *Computers in the Schools* 22, 3-4 (2005), 7–19.

[3] J McGrath Cohoon. 2002. Recruiting and retaining women in undergraduate computing majors. *ACM SIGCSE Bulletin* 34, 2 (2002), 48–52.

[4] Stephen Cooper, Wanda Dann, and Randy Pausch. 2000. Alice: a 3-D tool for introductory programming concepts. In *Journal of Computing Sciences in Colleges*, Vol. 15. Consortium for Computing Sciences in Colleges, 107–116.

[5] Microsoft Corporation. 2017. Why Europe's girls aren't studying STEM. (2017). retrieved July 15, 2019 from http://hdl.voced.edu.au/10707/427011.

[6] Bernhard Ertl, Silke Luttenberger, and Manuela Paechter. 2017. The impact of gender stereotypes on the self-concept of female students in stem subjects with an under-representation of females. *Frontiers in psychology* 8 (2017), 703.

[7] Neil Fraser. 2014. Google blockly-a visual programming editor. *URL: http://code.google.com/p/blockly. Accessed Sep* (2014). Now available at https://developers.google.com/blockly/; accessed 10-July-2019.

[8] Francisco J Gutierrez, Jocelyn Simmonds, Nancy Hitschfeld, Cecilia Casanova, Cecilia Sotomayor, and Vanessa Peña-Araya. 2018. Assessing software development skills among K-6 learners in a project-based workshop with scratch. In *Proceedings of the 40th International Conference on Software Engineering: Software Engineering Education and Training*. ACM, 98–107.

[9] Brian Harvey and Jens Mönig. 2010. Bringing âĂIJno ceilingâĂİ to scratch: Can one language serve kids and computer scientists. *Proceedings Constructionism* (2010), 1–10.

[10] Robert Holwerda and Felienne Hermans. 2018. A Usability Analysis of Blocks-based Programming Editors using Cognitive Dimensions. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 217–225.

[11] Yerika Jimenez, Amanpreet Kapoor, and Christina Gardner-McCune. 2018. Usability Challenges that Novice Programmers Experience when Using Scratch for the First Time. In *2018 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 327–328.

[12] Vivian Lagesen. 2005. *Extreme Make-over?: The Making of Gender and Computer Science*. Ph.D. Dissertation. STS-report 71/2005. Trondheim: Centre for Technology and Society, Norwegian University of Science and Technology.

[13] Vivian Anette Lagesen. 2008. A cyberfeminist utopia? Perceptions of gender and computer science among Malaysian women computer science students and faculty. *Science, Technology, & Human Values* 33, 1 (2008), 5–27.

[14] Anand Mahadevan, Jason Freeman, and Brian Magerko. 2016. An interactive, graphical coding environment for EarSketch online using Blockly and Web Audio API. (2016).

[15] Makeblock. 2019. mBlock - The educational programming software. http://www.mblock.cc. [accessed 17-July-2019].

[16] Cecilia Martinez, Marcos J Gomez, and Luciana Benotti. 2015. A comparison of preschool and elementary school children learning computer science concepts through a multilanguage robot programming platform. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, 159–164.

[17] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. 2011. Habits of programming in scratch. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. ACM, 168–172.

[18] Alexandros Merkouris, Konstantinos Chorianopoulos, and Achilles Kameas. 2017. Teaching programming in secondary education through embodied computing platforms: Robotics and wearables. *ACM Transactions on Computing Education (TOCE)* 17, 2 (2017), 9.

[19] Lauren R Milne and Richard E Ladner. 2018. Blocks4All: Overcoming Accessibility Barriers to Blocks Programming for Children with Visual Impairments. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 69.

[20] Vivek Paramasivam, Justin Huang, Sarah Elliott, and Maya Cakmak. 2017. Computer Science Outreach with End-User Robot-Programming Tools. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 447–452.

[21] Carlos Pereira Atencio. 2019. Ardublockly. *URL: https://ardublockly.embeddedlog.com. Accessed July* (2019).

[22] Kris Powers, Stacey Ecott, and Leanne M Hirshfield. 2007. Through the looking glass: teaching CS0 with Alice. In *SIGCSE*, Vol. 7. Citeseer, 213–217.

[23] Thomas W. Price and Tiffany Barnes. 2015. Comparing Textual and Block Interfaces in a Novice Programming Environment. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research (ICER '15)*. ACM, New York, NY, USA, 91–99. https://doi.org/10.1145/2787622.2787712

[24] Y Malini Reddy and Heidi Andrade. 2010. A review of rubric use in higher education. *Assessment & evaluation in higher education* 35, 4 (2010), 435–448.

[25] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[26] Ralf Romeike. 2007. Applying Creativity in CS High School Education: Criteria, Teaching Example and Evaluation. In *Proceedings of the Seventh Baltic Sea Conference on Computing Education Research - Volume 88 (Koli Calling '07)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 87–96. http://dl.acm.org/citation.cfm?id=2449323.2449333

[27] Alexander Ruf, Andreas Mühling, and Peter Hubwieser. 2014. Scratch vs. Karel: Impact on Learning Outcomes and Motivation. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*. ACM, New York, NY, USA, 50–59. https://doi.org/10.1145/2670757.2670772

[28] Mazyar Seraj, Serge Autexier, and Jan Janssen. 2018. BEESM, a block-based educational programming tool for end users. In *Proceedings of the 10th Nordic Conference on Human-Computer Interaction*. ACM, 886–891.

[29] Mazyar Seraj, Cornelia S Große, Serge Autexier, and Rolf Drechsler. 2019. Look What I Can Do: Acquisition of Programming Skills in the Context of Living Labs. In *Proceedings of the 41th International Conference on Software Engineering: Software Engineering Education and Training*. IEEE.

[30] Mazyar Seraj, Cornelia S Große, Serge Autexier, and Rolf Drechsler. 2019. Smart Homes Programming: Development and Evaluation of an Educational Programming Application for Young Learners. In *Proceedings of the 18th ACM International Conference on Interaction Design and Children*. ACM, 146–152.

[31] David Weintrop and Nathan Holbert. 2017. From blocks to text and back: Programming patterns in a dual-modality environment. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 633–638.

[32] David Weintrop and Uri Wilensky. 2015. Using Commutative Assessments to Compare Conceptual Understanding in Blocks-based and Text-based Programs.. In *ICER*, Vol. 15. 101–110.

[33] David Weintrop and Uri Wilensky. 2017. Between a Block and a Typeface: Designing and Evaluating Hybrid Programming Environments. In *Proceedings of the 2017 Conference on Interaction Design and Children*. ACM, 183–192.

[34] David Weintrop and Uri Wilensky. 2017. Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 3.