Impacts of Creating Smart Everyday Objects on Young Female Students' Programming Skills and Attitudes

Mazyar Seraj[1,2] Eva-Sophie Katterfeldt[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,drechsler}@uni-bremen.de
{mazyar.seraj,serge.autexier,rolf.drechsler}@dfki.de

ABSTRACT
In computer programming education, learning to program tangible objects has become a common way to introduce programming to young students. In an effort to address this intervention, scientific research has been done on the effectiveness of using tangible hardware platforms such as robots and wearable products to teach basic programming concepts to children. However, there is a lack of research on how young students' attitudes and programming skills are influenced over time, when they learn to program tangible objects and make them smart. In this paper, we investigate the impacts of using a tangible everyday object and making it smart on young female students' attitudes towards programming and the acquisition of basic programming skills. During a 4-day non-formal programming workshop with 12 6^{th} grade students, they were introduced to basic programming concepts, and learned how to apply them to turn a *houseplant* into a smart object. In a pilot study, we employed a block-based programming environment and analyzed the students' trajectories of attitudes towards programming and performance based on repeated open-ended qualitative questionnaires and programming questions throughout the workshop. The results show that all students had high confidence regarding programming skills, regardless of creating smart objects. Furthermore, it indicates that experienced students highly valued the programming of tangible everyday objects compared with inexperienced students. The findings of this work contribute to our understanding of how making tangible everyday objects smart can support the development of a positive attitude and keep up of interest throughout a programming workshop among girls.

CCS CONCEPTS
• **Applied computing** → *Computer-assisted instruction*; • **Social and professional topics** → **Computer science education**;

KEYWORDS
tangible objects, young female students, acquisition of programming skills, attitudes towards programming, smart objects

ACM Reference Format:
Mazyar Seraj[1,2] Eva-Sophie Katterfeldt[1] Serge Autexier[2] Rolf Drechsler[1,2] . 2020. Impacts of Creating Smart Everyday Objects on Young Female Students' Programming Skills and Attitudes. In *The 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20), March 11–14, 2020, Portland, OR, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3328778.3366841

SIGCSE '20, March 11–14, 2020, Portland, OR, USA

ACM ISBN 978-1-4503-6793-6/20/03…$15.00
https://doi.org/10.1145/3328778.3366841

1 INTRODUCTION
The number of formal and non-formal computer programming courses and workshops that aim to introduce programming and computer science to young students is growing. In computer programming education, the application of computing in reality, tends to be shown to students. In particular, allowing the students to learn the general purpose of programming and write code for tangible hardware platforms such as robots [18, 20, 23], smart homes [14, 28, 29], and wearable products [12, 15, 24] were considered in previous works. Furthermore, visual block-based programming environments are widely used in the design of introductory programming courses [12, 18, 29, 35]. These environments are beneficial for learning to code and starting with programming activities, especially for young students [32].

Despite the growing use of tangible objects and block-based programming, relatively little empirical study has been done to understand the impacts of smart objects together with block-based programming on young students' interest in programming and computer science in general. More specifically, it is not clear how teaching basic programming concepts to young students, and letting them implement these concepts in a tangible object and make it smart can improve their attitudes towards programming over time. Previous research addressed that introducing young students to new technologies supports learning programming [18, 20, 23] and stimulates interest in computer science [28, 29]. Although research on teaching programming to young students is vast (e.g., [4, 6, 11, 13, 20]), less is known about the students' trajectories of performance and attitudes towards programming in the context of smart objects. In addition, most western countries, such as European countries have significant problems with the number of female graduates in the field of computer science. It was addressed that approximately 25% of females pursuing a career in STEM (Science, Technology, Engineering, Mathematics) in the EU [7], and fewer than 1 in 5 computer science graduates are female across 35 European countries [5]. Thus, research is needed to offer insights into the impacts of embedding the creation of smart objects in programming courses, and into female attitudes towards programming and computer science more broadly. This paper seeks to answer the following question in order to address the gap in previous research:

How do young female students' programming skills and attitudes towards programming change over time in the context of creating smart everyday objects?

To answer this question, we present the result of a 4-day non-formal programming workshop with 12 6^{th} grade German female students (between 11 and 12 years old). We investigate the influence of using tangible everyday objects and making them smart on the development of a positive attitude towards programming among the students and on the improvement of their programming performance in a pilot study. A block-based programming environment was employed to reduce the complexity of programming and facilitate it for the students to learn basic programming concepts

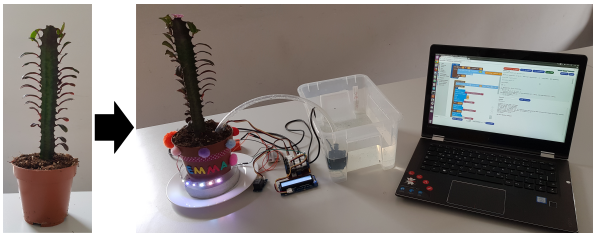


Figure 1: An example of one houseplant, at the beginning, and at the end of the workshop.

and author programs. A *houseplant* was provided as an appropriate stimulus that enables the students to connect a micro-controller, different sensors (e.g., light, temperature, humidity, sound) and actuators (e.g., LED light, water-pump, mp3-player, RGB LCD) to it. The students can program the sensors and actuators in a way that they react to each other, and build a *smart houseplant*. For instance, students can use a micro-controller (in this case, Arduino) and connect a humidity sensor, a relay, and a water-pump to it. Then, the students can program the micro-controller using block-based programming in order to enable the water-pump to pump water into the houseplant as soon as the sensor measures the humidity of the flower soil is below a certain degree (see Fig. 1). It is a special feature of this study that we examine the path of students' attitudes towards programming and their performance based on repeated open-ended qualitative questionnaires and programming questions at the beginning, in the middle and at the end of the workshop.

The paper begins with a review of previous work and how they employ tangible objects and hardware platforms to teach basic programming concepts together with block-based programming environments. Then, we describe the study design. We continue with our findings, followed by a discussion of the implications of these findings. The paper closes with conclusions and future works.

2 BACKGROUND AND RELATED WORK

The importance of learning computer programming has been shown, and it has been established as an area of research in computer science discipline [20, 33, 35]. Young students become familiar with the use of technologies (e.g., smartphones, tablets, computers, etc.), while they do not have programming skills [11, 20]. In the computer science education community (CSE), a large number of studies in the field of computing education highlighted the need to engage young students to learn basic programming concepts [18, 20, 34, 35]. In particular, they aim to motivate young female students learning the basics of computer programming and to enable them writing computer code [20, 28, 31]. However, there is limited consideration of how to improve young female students' attitudes and computer programming performance when it is channeled through appropriate stimuli, such as creating a smart object.

Tangible objects and block-based programming have been used [2, 26], and their benefits in learning programming have been shown, especially for young students [19, 20, 29, 30]. Findings show that their technological confidence benefited from block-based programming environments and tangibles, such as robotic computing platforms [18, 20] and computational textiles [12, 24]. Merkouris et al. [20] explored the benefits of learning to author programs for tangible hardware platforms such as robots and wearable computers in comparison to programming for desktop computers among young students. For this purpose, the authors used similar block-based programming environments (all based on Scratch [17]) in order to measure attitudes and programming performance in formal classrooms. It was shown that students' performance in learning basic programming concepts were not affected by the tangibles, although they showed a higher intention of learning programming when they program the robots compared with the desktop computers. Concerning gender, the girls performed better in the programming tests than the boys, although they felt less confident. Nevertheless, no information was provided on how the students' performance and attitudes changed over time from the beginning of the course towards the end of it.

The literature also reports that children learn better when they are engaged in designing and creating visible objects such as interactive applications, animations, robots, and computational textiles [20]. With respect to gender, according to [3, 15, 20] computational textiles activities make use of soft everyday materials (e.g., design bag with colors and LED lights), which are meaningful and give forms of expression to young students who are not primarily interested in technology. In addition, finding shows that girls underestimate their computer abilities and they struggle with assembling and programming materials (e.g., motors and gears) in robotic courses. Therefore, they enter programming courses with less confidence than boys [9, 10, 21]. However, Nourbakhsh et al. [21] found that girls' confidence increased more than boys by the end of the robotic courses. Furthermore, Kelleher and Pausch [16] show that performance and interest in programming highly depend on time spent in programming activities and prior programming experiences but not on gender. Nevertheless, research has not yet been conducted on the effectiveness of creating smart everyday objects together with block-based programming on young female students' programming skills and attitudes towards programming.

Most interventions to teach computer programming with block-based programming environments and tangible objects achieved high success to establish confidence and engagement among young students. However, it is still required to understand more how the use of these objects during a programming course together with block-based programming fosters programming skills, as well as promotes positive attitudes towards programming and computer science in general. In this work, we investigate the impacts of programming a *houseplant* as a tangible object and making it smart on female students' attitudes, performance, and level of interest in programming over time.

3 METHODS

The goal of this study was to experimentally investigate the impacts of tangible objects and block-based programming environments on young female students' programming skills and attitudes towards programming. We conducted a pilot study with 12 6^{th} grade female students (11-12 years old) in a 4-day non-formal programming workshop. A visual block-based programming environment based on Google Blockly, and a *houseplant* as the tangible everyday object were used. Three dimensions of students' attitudes were considered: confidence, enjoyment, and interest in future programming learning opportunities. The students' perception of using block-based programming and creating a smart object was measured with three questionnaires. Furthermore, the performance of the students was assessed with three programming questions. The questionnaires and programming questions were given to students: (i) at the beginning of the workshop (pre questions), (ii) at the end of the second day when the students had learned programming concepts (intermediate questions), and (iii) at the end of the workshop, when the students had implemented their newly learned programming skills in the houseplant and made it smart (post questions).

3.1 Study Design and Data Collection Strategy

In this study, we used a block-based programming environment in order to enable students to program a tangible everyday object. The programming environment is based on BEESM [27], which is built with the Blockly library [8] (see Fig. 2). BEESM is primarily designed

Table 1: 10-point Grading Rubric Scale

10-points (Connection)	8-points (Analysis)	6-points (Summary)	4-points (Incomplete)	2-points (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.

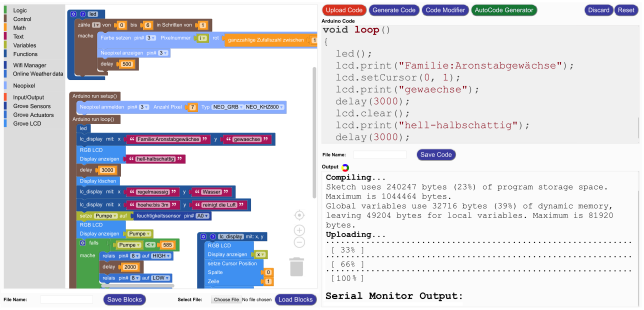


Figure 2: Screenshot of the programming environment, including the final program for a group; translated to English.

for inexperienced users and novices to learn and write code for smart environments, mobile robots, and micro-controllers one at a time and in combination with each other. The design and additional features of BEESM can be found in more detail in [27] and [29]. We changed the BEESM user interface to allow our students to use three different panels and have a full vision of blocks (Block Panel), code syntax (Code Panel), and output of the code (Output Panel). Furthermore, the tangible object in this workshop was a *houseplant*. This was used as an appropriate stimulus to enable the students to connect a micro-controller, as well as different sensors and actuators, program them, and build a smart object (see Fig. 1).

Pre, intermediate, and post questionnaires were employed to collect data concerning the students' attitudes and perceptions of programming, prior programming experience, and age group. The acquisition of basic programming skills was assessed, using a pre, an intermediate, and a post programming question.

Pre questionnaire (preQ). PreQ, which was distributed before the programming activities, consists of five open-ended questions to find out the students' attitudes towards programming and the programming workshop. With this regard, students' confidence, enjoyment, and interest in future programming learning opportunities were recorded. The students' confidence was asked through, "*how do you rate your programming skills?*" (Q1), and "*do you think you will be successful in this workshop?*" (Q2). The enjoyment was recorded using two questions "*I find programming...* (Q3)", and "*what are you looking forward to in this workshop?*" (Q4). The interest of students in learning programming was asked via the question "*how would you like to learn programming? why?*" (Q5). Furthermore, the students were required to determine their prior programming experience with block-based programming environments using the "*yes*" or "*no*" question "*have you ever worked with a block-based programming environment?*" (Q6).

Intermediate and post questionnaires (intermediateQ and postQ). IntermediateQ was distributed after learning basic programming concepts and activities in order to measure the students' attitudes towards programming. The students' perception of using block-based programming and creating a computer system consists of micro-controller, sensors, and actuators were also considered. This questionnaire was composed of the same questions as the pre questionnaire, just with different words for two questions; Q2 changed to "*do you think you were successful in this workshop?*", and Q4 changed to "*what did you like/dislike about the workshop?*". Furthermore, the students were required to answer two additional questions. These questions were about the block-based programming, "*how do you like programming with blocks?*" (Q7); and programming the sensors and actuators, "*what do you think about programming a computer system? (e.g., sensors and actuators)*" (Q8). In postQ, Q8 was changed to "*what do you think about programming a real smart object? (e.g., smart houseplant)*". All other questions remained the same as they were in intermediateQ.

Figure 3: Block-shaped elements in the pre programming question (prePQ); translated to English.

Programming questions (prePQ, intermediatePQ, and postPQ). In order to evaluate the students' prior programming experience and measure the acquisition of programming skills, they were asked to perform a pre programming question (prePQ), an intermediate programming question at the end of the second day of the workshop (intermediatePQ), and a post programming question at the end of the workshop (postPQ). In each pre, intermediate and post programming question, block-shaped elements were designed independent of the block-based programming environment in order to test how well the students acquire the basic programming concepts which were taught during the workshop (e.g., see Fig. 3).

PrePQ asked to program the micro-controller to get the data from a connected sensor, write the sensor's value into a variable and show it in an RGB LCD for 2 seconds. We added control-flow statements in intermediatePQ; therefore, we asked this time that `if` the value of the sensor is less than 20, `then` the RGB LCD should show the value for 5 seconds in the second row and fifth column. PostPQ contains all previous concepts plus loops. This time, we asked that `if` the value of the sensor is more than 30, `then` the RGB LCD should show the value for 3 seconds in the first row and the fourth column. In addition, the LCD should then blink in green for 3 times with 1 second delay in between.

In each programming question, students were asked to answer the question via selecting a set of blocks and identifying the order of them in a correct logical way. It was noted that some blocks might not be needed and some may appear more than one time in their answers. All programming questions are slightly different from each other, and they are getting more advanced from the beginning towards the end of the workshop. This counterbalance design of questions is to ensure that students read the questions carefully and identify the order of blocks based on the question. Furthermore, these questions represent realistic programming problems for a micro-controller (e.g., Arduino), a sensor (e.g., light, temperature) and an actuator such as RGB LCD. The solution given by the student were collected for each programming question and evaluated by a 10-point grading rubric [25] (see Table 1). Each solution was scored independently by two researchers to ensure consistent grading.

3.2 Participants

A total of 12 6^{th} grade female students (between 11 and 12 years old) of a German secondary school participated in this study. The school teacher was contacted regarding our programming workshop. Then, students and their parents were informed by their school to register for it. Therefore, the students who participated in this study were self-selected, and interested in learning programming and having programming activities. None of the students had received teaching in programming as part of their regular school curriculum. However,

we employed a question to record their previous programming experiences, and six students indicated that they had worked with block-based programming environments and tangibles in the past.

3.3 Procedure

The duration of each daily session was five hours, with one hour break. One female and one male instructor led the whole workshop. Both instructors had a computer science background with experience in working with young students. In this study, students worked in pairs on each of the activities. Students with prior programming experience were paired together, and those without experience were paired with each other. All students answered pre, intermediate, and post questionnaires and programming questions individually. The questionnaire was filled first each time, followed by the programming question. Each day, an oral explanation was given, using prepared slides. Additionally, we used supplementary documents—including an explanation of all materials and necessary blocks for programming and activities—in order to help the students, as well as minimize and control the instructor effects. The topics and activities were covered during each day as follows:

Day 1. First, the students filled in preQ and prePQ. They were then informed that they are going to have a set of programming activities in each group, and that these activities help them to program and design a smart *houseplant*. Then, pairs of two students (2 experienced or 2 inexperienced) were assigned to one computer. This session was followed by an introduction to the block-based programming environment. The programming concepts introduced in this session were variables, loops, and RGB coloring model, using Arduino and RGB LCD. Students also learned how to show string and numerical values on the RGB LCD in different courser positions, and how to change the LCD color. We asked the students to explore the corresponding blocks in the programming environment.

Day 2. First, we continued with how variables and loops function. Then, students were introduced to sensors (light, temperature, humidity, sound, etc.) and actuators (LED light, water-pump, mp3-player, RGB LCD, etc.). They also learned how to get data from a sensor and put it into a variable. The programming concepts included in this session were the definition of control-flow statements, such as conditions and logical operators. Students were required to execute and understand the blocks. In this respect, they started to program actuators to react to sensors data. At the end of the second day, students filled in intermediateQ and intermediatePQ.

Day 3. At the beginning of this session, each group of students was asked to present and share with others how the Arduino, sensors, actuators, variables, loops, and control-flow statements work and are executed. Then, each group chose a *houseplant*. We asked them to give a character to the houseplant and think of how the sensors and actuators in their desire houseplant should communicate and react to each other. This session followed by the implementation of the programming concepts in Arduino and start programming it based on the character of the houseplant.

Day 4. In this session, students continued with programming and designing the houseplant. At the end of this session, postQ and postPQ were given to the students to find out the changes in their performance and attitudes from the beginning of the workshop towards the end of it. The workshop ended with the presentation of the character and the functionality of each houseplant.

4 RESULTS

All participants filled out preQ and intermediateQ, as well as prePQ and intermediatePQ. One participant did not show up on the last day, and thus, postQ and postPQ were filled by eleven students. Please note, this student showed a negative attitude in preQ and intermediateQ. This case is not addressed in the description of results, but it is included in the diagrams and we refer to it in Section 5. The written responses to the questionnaires were coded independently by two researchers and then discussed in order to find an agreement on final categories. In each diagram (Fig. 4 to Fig. 6), P1 to P6 are students with prior experience, and P7 to P12 are students without prior experience in programming.

Table 2: Students' Programming Performance

Questions	Experienced *M (SD)*	Inexperienced *M (SD)*	*ANOVA Results*
PrePQ	5.00 (2.45)	2.00 (1.26)	$F(1,10) = 7.11$, $^{**}p = 0.024$
IntermediatePQ	5.33 (2.73)	3.33 (1.03)	$F(1,10) = 2.81$, $p = 0.12$
PostPQ	5.67 (2.94)	2.40 (0.89)	$F(1,9) = 5.63$, $^{**}p = 0.042$

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

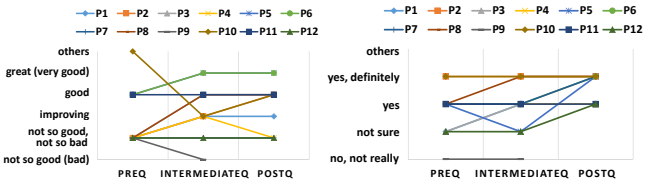


Figure 4: (a) Students rate their programming skills (Q1); (b) Students' thoughts on their success in the workshop (Q2).

4.1 Acquisition of Programming Skills

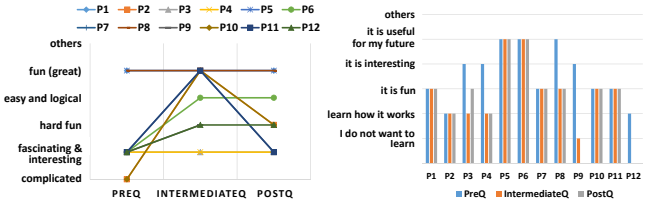
The students' performance was assessed three times, at the beginning, in the middle and at the end of the workshop (described in Section 3). The experienced students performed significantly better than the inexperienced students in prePQ and postPQ, and their performance improved (descriptively) from prePQ towards the postPQ (see Table 2). Furthermore, the performance of inexperienced students improved (descriptively) in intermediatePQ, where no significant difference was obtained compared with the performance of experienced students. However, their performance dropped in postPQ (see Table 2). No significant difference occurred within each group of experienced and inexperienced students from prePQ towards postPQ.

4.2 Attitudes and Perceptions of Programming

4.2.1 Confidence. Concerning the students' confidence, they were asked to rate their programming skills (Q1). After coding their responses, we had the following categories: "*not so good (bad)*", "*not so good but not so bad*", "*improving*", "*good*", "*great (very good)*", and "*others*" (see Fig. 4a). We saw a positive trend in intermediateQ for two-third (8) of the students while it remained the same for the other participants. It decreased for one student who did not show up on the last day (P9). No specific difference was observed between inexperienced and experienced students. In postQ, we saw an increase in confidence for two participants (P2 and P10). One (P4) rated her skills less positive than in intermediateQ, but equal to preQ. The remaining participants rated their skills the same as in intermediateQ.

The students were also asked whether they think that they would be successful in the workshop (Q2). The answers were categorized as "*no, not really*", "*not sure*", "*yes*", "*yes, definitely*", and "*others*" (see Fig. 4b). At the beginning, three students were unsure (P1, P3, and P12), and one said "*no*" (P9). The confidence increased or remained the same (positively) for all students, except one (P9), towards the end of the workshop. At the end, no unsureness was seen, and all students rated their success with "*yes*" or "*yes, definitely*".

4.2.2 Enjoyment and Interest. With respect to the enjoyment of programming, students were asked to indicate how they find programming (Q3). The answers were categorized in "*complicated*", "*fascinating and interesting*", "*hard fun*" (inspired by [22]), "*easy*



(a) (b)

Figure 5: (a) How students found programming (Q3); (b) Why students like to learn programming (Q5).

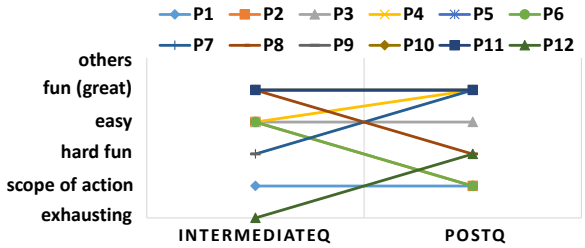


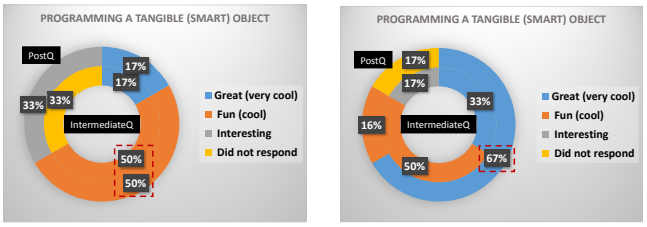
Figure 6: How students like to program with blocks (Q7).

and logical", "*fun (great)*", and "*others*" (see Fig. 5a). All participants liked programming in intermediateQ and postQ more than in preQ. However, four students (P2, P7, P10, and P11) changed their mind from intermediateQ to postQ; for instance, they realized that the programming is not only "*fun*", but fun and, at the same time, challenging ("*hard fun*"), or just "*interesting*".

The students were also required to indicate why they would like to learn programming (Q5). This question aimed to find out about their general interest in learning programming. The answers were categorized as "*I do not want to learn*", "*learn how it works*", "*It is fun*", "*It is interesting*", "*It is useful for my future*", and "*others*" (see Fig. 5b). Four students (P1, P7, P10, and P11) mentioned that programming is "*fun*" from the beginning of the workshop towards the end of it. One student (P3) finally found that the programming is "*fun*". Furthermore, three students found it "*useful for their futures*" (P5, P6, and P8). However, P8 changed her idea and found programming is "*fun*" during the workshop. For seven participants, the reason to learn programming remained the same throughout the workshop. In particular, the motivation of inexperienced students to learn programming was (expecting) "*fun*", while experienced students mentioned the learning of how technical things work.

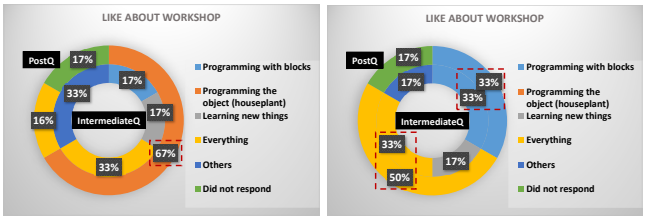
4.2.3 Block-based Programming and Smart Objects. In intermediateQ and postQ, participants were required to respond to how they liked programming with blocks (Q7). The answers were categorized as "*exhausting*", "*scope of action (e.g., opportunities to be creative)*", "*hard fun*", "*easy*", "*fun (great)*", "*others*". Here, the answers changed from intermediateQ to postQ for half of the participants (see Fig. 6). Experienced participants often mentioned that "*it is easy*", or appreciated the "*scope of action*", while inexperienced students found it "*fun*" or "*hard fun*". Please note that all experienced students mentioned in Q6 (in preQ) that they had worked with other block-based programming environments before.

We also distinctly asked how they liked programming a tangible (smart) object ((Q8) in intermediateQ and postQ). Categories were "*great (very cool)*", "*fun (cool)*", "*interesting*", "*no response*" (see Fig. 7). We saw a difference between experienced and inexperienced students. Half of the experienced students indicated that it was "*fun (cool)*" in both intermediateQ (P2, P4, and P6) and postQ (P2, P3, and P6). P4 found it "*interesting*" in postQ, and P3 did not answer this question in intermediateQ. Inexperienced students indicated



(a) Experienced students (b) Inexperienced students

Figure 7: How students like to program a tangible object (Q8).



(a) Experienced students (b) Inexperienced students

Figure 8: What students like about the workshop (Q4).

that it was "*great (very cool)*" (P8 and P11) or "*fun (cool)*" (P7, P9 and P10) in intermediateQ; P12 found it "*interesting*" to program a tangible object in intermediateQ. Their enthusiasm grew towards the postQ with a shift of two students (P10 and P12) to "*great (very cool)*". In general, half of the students did not change their minds between intermediateQ and postQ.

Students were also asked (Q4) to answer what they look forward to in the workshop (preQ) and what they liked about the workshop (in intermediateQ and postQ). Categories are "*teamwork*", "*learning programming/technical things*", "*program a real object*", "*others*", and "*no response*" in preQ. In addition, categories in both intermediateQ and postQ are "*programming with blocks*", "*programming the object (houseplant)*", "*learning new things*", "*everything*", "*others*", and "*did not respond*" (see Fig. 8). In preQ, students were mostly (5) looking forward to "*learning new and technical things*". For inexperienced students, this was about two-third (P7, P8, P10, and P12). However, experienced students were more differentiate, and two of them specified "*programming a tangible object*" (P4 and P6). In intermediateQ, the distribution of categories changed, and we can see a clear difference between experienced and inexperienced students. With this regard, experienced participants mostly mentioned: "*programming with blocks*" (P2 and P3), and "*everything*" (P1 and P4) in intermediateQ. In postQ, two-third (4) of them mentioned: "*programming the object (houseplant)*" (P3, P4, P5, and P6). Inexperienced participants mostly answered, "*everything*" (P11 and P12) and "*programming with blocks*" (P7 and P8) in intermediateQ. In postQ, three of them (P10, P11, and P12) mentioned "*everything*", and two of them (P7 and P8) mentioned "*programming with blocks*".

5 DISCUSSION

In this study, the students' performance in the programming questions did not correlate with their confidence concerning perceived programming concepts. Therefore, although girls had some difficulties in understanding the subject, they still felt positive and confident about their programming skills and success. This is in line with the results presented in [1, 20, 21] that girls' confidence level increases by visual programming environments and experience with interactive tangible objects.

With respect to the confidence, enjoyment and interest, a clear difference between experienced and inexperienced students was

not observed. However, some differences were seen in the students' responses to the questions regarding the workshop activities and items. The majority of experienced participants mentioned that they liked creating a smart object in this workshop, while the inexperienced students remained rather vague and mentioned "*everything*" or only "*programming*". The results showed that especially the experienced students appreciated working with the houseplant (as a tangible everyday object) and, due to their prior experience, they were able to articulate clearly what they like and why it was fun for them. Furthermore, the opportunity of applying their programming skills to a tangible object and making it smart was more meaningful and interesting for them. In contrast, the inexperienced students did not mention the tangible everyday object (in this case, houseplant) but they were mostly impressed by programming with a visual block-based programming environment. In addition, the findings showed that programming by itself was interesting for the inexperienced students and applying it to the tangible object did not stand out. We can assume that they did not yet have the terminology to distinguish between programming with and without tangibles. From these results, we can draw implications on designing courses for experienced and inexperienced (female) students. For experienced students, it is indicated that having a meaningful application area for programming such as a tangible everyday object, as well as making it smart is an important area that needs to be taken into account. This supports the results from the programming questions that experienced students performed significantly better than inexperienced students in the postPQ, while it was not significantly better in intermediatePQ. This result is in line with findings from [28], which showed that experienced students had more benefits from tangible objects and platforms than inexperienced students.

Our initial research question was how programming skills and attitudes change over time in the context of programming and creating smart everyday objects. This also includes finding out whether using a tangible everyday object and providing the possibility to make it smart changes young female students' attitudes towards programming. Concerning the confidence in programming, the biggest changes were observed between preQ and intermediateQ. After the introduction to the block-based programming environment, the confidence in programming increased. Except for one (the dropout student), all students felt that they performed equal or better than what they had expected. The findings showed that the students' confidence in programming was not affected by the implementation of the programming concepts in a tangible everyday object and the experience gained by that time. This is in line with their opinion about programming. After initial excitement in intermediateQ, their confidence dropped. This indicates that they realized programming is not just fun but also challenging after programming and creating the smart object. However, the feeling of being successful increased after working with the tangible object and making it smart. One reason could be that "*smart houseplant*" was the general topic and the objective of the workshop, which our participants felt that it was achieved.

Enjoyment of programming increased in intermediateQ, and it had a decrease for a few participants after using the tangible object. We assume, one reason is that the complexity of the tasks increased. Nevertheless, working with tangibles did not indicate a distinct effect on enjoyment but helped to keep it up.

As mentioned in Section 4, one student dropped out. Although she had excused herself to the workshop instructor in advance that she had another commitment on the last day, she showed negative confidence and attitude in her responses to some questions in preQ and intermediateQ. While the results which are obtained from the other students are promising, we would like to highlight that we can probably learn a lot more from dropout participants. Thus, we argue for looking more in-depth into these cases in future iterations of this work; for instance, using ethnographic methods.

5.1 Limitations and Future Work

While we tried to provide insights into the relationship between creating smart objects and improving young female students' attitudes and programming performance, this study has limitations which are addressed in the following.

The first limitation of this study relates to the number of programming tasks and the period of the workshop. For instance, the findings of this study are limited to the diversity of the programming tasks that the students performed and how they speak to more diverse programming activities and computational skills. A second similar limitation is that the intermediate questionnaire and programming question might have had an influence on students' performance in learning basic programming concepts and their motivation. Further work is needed to address these limitations and to generalize the findings beyond the specifics of this study, such as the period of programming workshop and type of programming tasks.

Another limitation of this pilot study is related to the number of participants. We would like to emphasize that the relatively small sample size (12 female students) lowers the power of findings to be generalized on a large scale. Thus, we look at this as a major concern that needs to be addressed in future directions when we expand the scope of this work with a larger sample size.

A final limitation of this study relates to the control group. This is another pathway of future work to find out the impacts of programming courses on students' attitudes and programming skills over time without using tangible objects as well as when young male students are targeted for such programming courses.

6 CONCLUSION

As the presence of women in computer science discipline is lower than men in most western countries, raising girls' interest in the programming side of computer science is an active area of research. In this paper, we present a pilot study investigating the impacts of programming and creating a smart everyday object (smart houseplant) on girls' programming skills and attitudes. Programming performance, enjoyment, interest, and confidence were not only assessed with programming questions and open-ended questionnaires before and after a non-formal programming workshop but also intermediately before starting to implement programming in the tangible everyday object (a houseplant). Our findings indicate the girls' confidence did not match their actual performance. Their confidence increased after introducing the block-based programming environment, and it remained high after creating the smart object. However, being able to program the houseplant was perceived differently by experienced and inexperienced participants. Data shows that block-based programming was interesting "*enough*" for inexperienced students, while experienced students appreciated more to apply their skills to make the houseplant smart. Furthermore, our study supports the claim that creating smart objects has a direct impact on young female students' performance and attitude towards programming. By studying the influence of creating smart everyday objects as an application of computing in reality, we enhance our understanding of designing appropriate programming courses concerning the students' prior experience. While many questions still remain on how to best introduce programming to girls, the findings of this study are essential to inform other researchers and educators about the relation between tangible objects together with block-based programming, and girls' programming performance and attitudes towards programming.

7 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] 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.

[2] Paulo Blikstein. 2015. Computationally Enhanced Toolkits for Children: Historical Review and a Framework for Future Design. *Foundations and Trends in Human–Computer Interaction* 9, 1 (2015), 1–68. https://doi.org/10.1561/1100000057

[3] Leah Buechley, Mike Eisenberg, Jaime Catchen, and Ali Crockett. 2008. The LilyPad Arduino: using computational textiles to investigate engagement, aesthetics, and diversity in computer science education. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 423–432.

[4] John W Coffey. 2017. A Study of the Use of a Reflective Activity to Improve Students' Software Design Capabilities. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 129–134.

[5] Microsoft Corporation. 2017. Why Europe's girls aren't studying STEM. (2017). retrieved August 10, 2019 from http://hdl.voced.edu.au/10707/427011.

[6] Inés Friss de Kereki and Areti Manataki. 2016. "Code Yourself" and "A Programar": a bilingual MOOC for teaching Computer Science to teenagers. In *Frontiers in Education Conference (FIE), 2016 IEEE*. IEEE, 1–9.

[7] 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.

[8] 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-August-2019.

[9] Elena Gorbacheva, Jenine Beekhuyzen, Jan vom Brocke, and Jörg Becker. 2019. Directions for research on gender imbalance in the IT profession. *European Journal of Information Systems* 28, 1 (2019), 43–67.

[10] Denise Gürer and Tracy Camp. 2002. An ACM-W literature review on women in computing. *ACM SIGCSE Bulletin* 34, 2 (2002), 121–127.

[11] Yasmin B Kafai. 2016. From computational thinking to computational participation in K–12 education. *Commun. ACM* 59, 8 (2016), 26–27.

[12] Yasmin B Kafai, Eunkyoung Lee, Kristin Searle, Deborah Fields, Eliot Kaplan, and Debora Lui. 2014. A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles. *ACM Transactions on Computing Education (TOCE)* 14, 1 (2014), 1.

[13] Filiz Kalelioğlu. 2015. A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior* 52 (2015), 200–210.

[14] Eva-Sophie Katterfeldt and Nadine Dittert. 2018. Co-designing Smart Home Maker Workshops with Girls. In *Proceedings of the Conference on Creativity and Making in Education*. ACM, 100–101.

[15] Eva-Sophie Katterfeldt, Nadine Dittert, and Heidi Schelhowe. 2009. EduWear: smart textiles as ways of relating computing technology to everyday life. In *Proceedings of the 8th International Conference on Interaction Design and Children*. ACM, 9–17.

[16] Caitlin Kelleher and Randy Pausch. 2005. Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys (CSUR)* 37, 2 (2005), 83–137.

[17] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE)* 10, 4 (2010), 16.

[18] 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.

[19] Edward F Melcer and Katherine Isbister. 2018. Bots & (Main) Frames: exploring the impact of tangible blocks and collaborative play in an educational programming game. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 266.

[20] 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.

[21] Illah R Nourbakhsh, Emily Hamner, Kevin Crowley, and Katie Wilkinson. 2004. Formal measures of learning in a secondary school mobile robotics course. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, Vol. 2. IEEE, 1831–1836.

[22] Seymour Papert. 2002. Hard Fun. *Bangor Daily News (Bangor, Maine)* (2002). retrieved August 1, 2019 from http://www.papert.org/articles/HardFun.html.

[23] 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.

[24] Kanjun Qiu, Leah Buechley, Edward Baafi, and Wendy Dubow. 2013. A curriculum for teaching computer science through computational textiles. In *Proceedings of the 12th International Conference on Interaction Design and Children*. ACM, 20–27.

[25] 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.

[26] Mitchel Resnick, Fred Martin, Robert Berg, Rick Borovoy, Vanessa Colella, Kwin Kramer, and Brian Silverman. 1998. Digital manipulatives: new toys to think with. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, New York, NY, 281–287. https://doi.org/10.1145/274644.274684

[27] 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.

[28] 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.

[29] 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.

[30] Jaekwoun Shim, Daiyoung Kwon, and Wongyu Lee. 2016. The effects of a robot game environment on computer programming education for elementary school students. *IEEE Transactions on Education* 60, 2 (2016), 164–172.

[31] Amanda Sullivan and Marina Umashi Bers. 2016. Girls, boys, and bots: Gender differences in young children's performance on robotics and programming tasks. *Journal of Information Technology Education: Innovations in Practice* 15 (2016), 145–165.

[32] David Weintrop. 2019. Block-based programming in computer science education. *Commun. ACM* 62, 8 (2019), 22–25.

[33] 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.

[34] 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.

[35] 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.