Smart Homes Programming: Development and Evaluation of an Educational Programming Application for Young Learners

Mazyar Seraj[1,2] **Cornelia S. Große**[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,cornelia.grosse,drechsler}@uni-bremen.de
{mazyar.seraj,serge.autexier,rolf.drechsler}@dfki.de

ABSTRACT

In light of the complexity of introductory programming for young learners, visual programming has become more and more popular. In particular, block-based educational programming systems have emerged as an area of active research. This paper introduces an educational block-based programming application, enabling young learners to learn and make programs in the context of smart homes. In this application, smart objects have a set of primitive behaviors which can be integrated in the general features of programming languages like variables, conditionals, loops, and functions. The programming language is shown in a graphical interface to enable young students to program with the application. The development and implementation of this application, along with helping features for the students are described. In a pilot study with 20 7^{th} grade students, the application's effectiveness and ease of use are evaluated. The results show that students can fairly solve programming problems and make real programs in the context of smart homes. Feedback of the learners is presented and discussed.

CCS CONCEPTS

• **Human-centered computing** → **Visualization**; *Interactive systems and tools*; • **Applied computing** → *Interactive learning environments*;

KEYWORDS

Educational Block-based Programming; Smart Homes; Young Learners; Visual Programming; Google Blockly

ACM Reference Format:
Mazyar Seraj[1,2] Cornelia S. Große[1] Serge Autexier[2] Rolf Drechsler[1,2] . 2019. Smart Homes Programming: Development and Evaluation of an Educational Programming Application for Young Learners. In *Interaction Design and Children (IDC '19), June 12–15, 2019, Boise, ID, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3311927.3323157

IDC '19, June 12–15, 2019, Boise, ID, USA

ACM ISBN 978-1-4503-6690-8/19/06…$15.00
https://doi.org/10.1145/3311927.3323157

1 INTRODUCTION

Young learners often have difficulties with respect to designing, integrating, compiling, executing, and debugging in introductory programming [11, 24]. These difficulties experienced by young students are related to (i) students' syntactical knowledge (e.g., syntax errors), and (ii) students' conceptual and strategic knowledge (e.g., errors when assembling and manipulating code structure) [24]. Educational programming tools generally either support students to achieve results quickly, or introduce them to real programming development environments used by professionals [2]. However, we face the lack of a settlement between a pure programming development environment, e.g., Integrated Development Environment (IDE) and a simple interface designed for the students, allowing them to learn programming and to achieve results quickly. In this respect, visual block-based programming is designed to allow the students to learn programming and overcome the obstacles of syntax and manipulation of code structure [1, 2, 5]. Furthermore, visual block-based programming environments have the advantage to be more independent of young students' native language and can be employed in different languages [10, 31]. Besides these programming environments, the existence of a motivating context is necessary. In the present contribution, the context is a real life-size smart home that reflects the programming activities of young learners. Smart homes are contexts that perfectly match the educational programming purpose and maintain the motivation of the learners as (i) they can easily see the consequences of their programming activities (e.g., as soon as someone enters the bathroom, the lights turn on), and (ii) they can experience the latest technologies and learn about the future.

We developed an educational block-based programming application to program smart homes. This approach enables young learners to learn and make programs which can be applied in smart homes. The application provides a graphical interface (see Fig. 1), and allows the learners to have a short time span between the development of ideas and the transformation and integration into a smart home. Learners have access to the standard programming language and can make

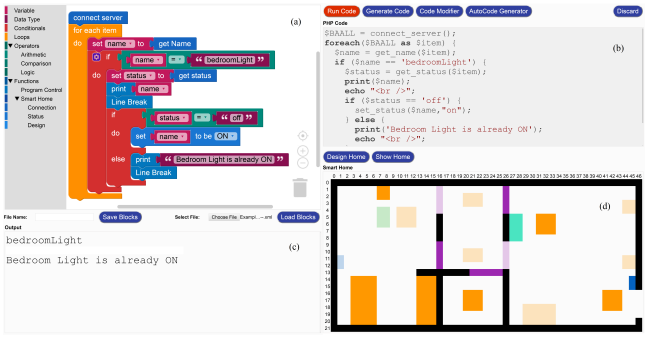

Figure 1: Screenshot of the programming application interface. (a) Blockly UI panel (translated to English); (b) Code panel; (c) Output panel (translated to English); (d) 2D graphical view of the smart home.

modifications. Furthermore, they are able to control the logic and flow of their programs. In this respect, the application provides several opportunities for them: (i) creating, editing and running programs in a real life-size smart home (and not just a toy robot or a doll house), and (ii) programming a real smart home in the German Research Center for Artificial intelligence (DFKI) that is also used by researchers, i.e., learners can actively participate and experience latest research efforts on a smart home and learn about the future.

This paper describes the development and evaluation of this educational block-based programming application in order to facilitate programming for young learners. Results on the application's effectiveness and ease of use are provided through a pilot study with 20 7^{th} grade German secondary school students. Furthermore, as introductory programming is difficult for young students [2, 24], visual block-based programming environments may still not be fully intuitive for them. Thus, they need help in order to solve programming problems using these environments. Based on instructional supports presented in [18, 26, 33], it was addressed that young learners can be supported by worked examples and instructional explanations in order to solve programming problems. Therefore, by using these supplementary documents, we helped our young students to work with the programming application and solve programming problems. The main contributions of this paper are as follows:

- Development of an educational block-based programming application in the context of smart homes in order to support young learners to learn and make programs as well as learn about the future.
- Evaluation of the application's ease of use and effectiveness in order to help young learners to solve programming problems.

2 BACKGROUND AND RELATED WORK

In this section, first, approaches using visual programming in educational tools and environments are introduced. Then, primitives which are designed and used in order to control the behavior of smart objects and smart environments are presented.

Visual Programming

Visual (graphical) programming has been used to make programming problems easier to understand and solve. Young learners can create complex programs with little training in visual programming environments [21, 22]. Visual programming is used in block-based programming editors such as Scratch [25], Snap! [12], Alice [6] and mBlock [17]. Scratch and Alice are generally used to make animations, games and interactive applications. Snap! is an extended reimplementation of Scratch, letting students to build their own blocks. Using mBlock allows young students to program robots and micro-controllers. Pencil Code is a block-based coding tool developed based on Droplet [1] to help young students work with JavaScript, CoffeeScript and HTML [2, 31]. Students are enabled to toggle between text code and blocks freely. This approach enhances familiarity with syntax while transferring from blocks to text code [2, 30, 31]. Visual programming, in particular Google Blockly, has been used as a client side of web-based environments in a number of educational and commercial applications and tools. MIT App Inventor [14], CodeIt [23], CustomPrograms [13], CoBlox [29], BEESM [28], and MakeCode [15] take advantage of Google Blockly in order to enable young learners and inexperienced users learning and making programs.

In contrast to a large part of this previous work, we seek to explore impacts of visual block-based programming environments with educational focus on young learners. We wish to support them in order to learn the general purpose of programming as well as rapidly prototype and customize ideas in the context of smart homes. Furthermore, we would like to introduce learners to new technologies which provide possibilities to tightly connect computer science to reality. In other words, we aim to introduce the future to young learners. In this respect, a visual block-based programming application is provided which enables learners to learn programming and implement their ideas into the development and control of real smart homes in order to motivate them to take part in that future.

Primitives and Smart Environments

In previous research, capabilities of smart devices were organized into *Primitives*. CustomPrograms [13] and CARMEN [20] use this method to implement robots' behavior and capabilities into primitives. Each primitive robot behavior can be called through a function such as navigating the robot to a location. These primitives are used in order to design a visual block-based application which enables students with disabilities (e.g., deafness, muscular dystrophy, and attention deficit disorder) to program Clearpath Turtlebot, capable of delivering items and autonomously navigating its environment [23]. Furthermore, BlocklyDuino which is used in [16] takes advantage of this method to enable preschool and elementary school children to program and control the behavior of Arduino boards.

Smart Environments are composed of a set of smart and controllable household appliances. Web socket communication protocol [9] and Remote Procedure Call (RPC) technology [3, 8] are fairly used as a Web-based interaction to

control them. In this way, HTTP GET and POST requests that return JavaScript Object Notation (JSON) [4] responses are used to communicate between user and Web server. BEESM takes advantage of using primitives in order to enable inexperienced users and novice programmers to program smart environments as well as mobile robots (e.g., Turtlebot) and micro-controllers (e.g., Arduino or WeMos boards) one at a time and in combination with each other [28].

3 OVERVIEW OF THE APPLICATION

The proposed educational block-based programming application is based on BEESM, enabling young learners to learn and make programs in the context of smart homes. Different features of Hypertext Preprocessor (PHP) programming language [27] like variables, data types, conditionals, loops, predefined functions, and operators are included in the application.

In this section, the application and its primitives are presented. First, the smart home is introduced. Second, the capabilities of smart objects are provided, referred to as primitives. Third, the method of using visual programming is explained. Finally, the method of compilation and execution of the code which is generated by blocks is addressed.

Smart Home

Our smart home which is part of the DFKI is a 60 m^2 ambient assisted living lab including bedroom, living room, kitchen and bathroom (see Fig. 2). The living lab contains various actuators (e.g., doors and lights) and sensors (e.g., lighting and temperature). Furthermore, two RGB lamps are available with an HTTP interface for color setting. Our application can be applied to any other smart home using web socket communication protocol and RPC technology to control smart objects.

The smart living lab's main educational use here is letting young students program different objects for different purposes in a real life-size smart environment which is also used by researchers. Additionally, it provides a unique opportunity for them to experience latest innovation and to learn about the future. In this respect, students are enabled to program different real objects in the living lab and to observe reactions of these objects to the program in real-time.

Figure 2: A view of the smart home.

Primitives

Connecting to the server (contains all objects names and status) and each primitive object's behavior is wrapped in a function that can be called by the students. Primitives are explained in the following.

Objects' attributes. Three primitives related to server connection and reading object name and status (objects' attributes) were designed. The **connect_server** primitive enables students to connect to the webserver and have names and status of all real objects in an array. The **get_name** is a simple block to return the name of all objects, while using the **get_status** block allows students to access the status of each object. These blocks allow students to search and access the name and current status of each object.

Changing status. Learners can change the status of each object using their names. Primitives were included not only to change the status of each object which is inserted as an input by the students, but also to show the name, new status and real-time changes. The **set_status** block was included for changing the status of switchable devices and objects which have a string or number type of value. The status of all dimmable devices can be set using the **set_status_dimmer** block. The **set_statusRGB** can be applied for RGB lights.

Application Interface

We used visual programming, specifically Google Blockly [10] to facilitate programming. Blockly includes programming elements such as variables, loops, conditional statements, logical operators and functions. Each element is represented as a block shaped that they can be snapped together like puzzle pieces. Blockly allows custom blocks by defining a block's appearance, inputs, outputs, and type of connections. The code generator for each block which is written by developers enables Blockly to generate the code syntax of the programming language of our choice. In our application, PHP code is generated out of the blocks. PHP programming language was chosen as it is a powerful server side scripting language to interact with web servers. Furthermore, PHP is a widely-used, free and efficient programming language.

A set of custom blocks was defined for all primitives with inputs, outputs and type of connections. Standard PHP programming language elements are also provided. In addition, some PHP functions were designed such as *print_r()* to print an array and *sleep()* to delay the program execution. These functions are used in order to provide more options to control the program flow under a category called *Program Control*.

With this regard, four panels are designed in the application (see Fig. 1) as follows:

(a) *Blockly UI* includes a workspace and a toolbox containing both predefined and customized blocks, where learners can find and assemble blocks (see Fig. 1a).
(b) A *Code* panel demonstrates the generated code and enables learners to edit the code syntax (see Fig. 1b).
(c) An *Output* panel shows all return values and errors for debugging purposes (see Fig. 1c).
(d) A *2D graphical view* displays how the status of each object changes based on the program (see Fig. 1d).

Compilation and Execution

Each block generates PHP code based on their inputs and outputs. The code syntax is generally created by generating code for the blocks which are at the top level of the program.

When the student runs the program, the generated PHP code is sent to the web-server. The application checks whether the *Code Modifier* mode is enabled or whether the code generated by the blocks should be compiled and executed. In this application, we offer Boolean success values for error handling: **connect_server**, **get_name** and **get_status** return false if the connection is not established; **set_status**, **set_statusRGB** and **set_status_dimmer** return false if the status is not changed and/or if the new status is not applied to the real object.

4 PILOT STUDY

Our application was evaluated in a training session in order to find out whether it was supportive for young learners to learn and make programs. Specifically, the effectiveness and ease of use of the application were assessed, and it was analyzed whether it was beneficial for the learners in order to solve programming problems. A pilot study was conducted with 20 7^{th} grade students of a German secondary school (7 girls, 13 boys, age: $M = 13.15$, $SD = 0.75$). The duration of the training session was 150 minutes.

Design

In this study, in addition to an oral introduction to programming and the programming application, we used supplementary documents – namely worked examples and instructional procedures – in order to help our learners working with the application. As introductory programming is difficult for young learners [2, 24], and they might not listen carefully to the oral explanations or might not remain fully concentrated, supplementary documents were provided in paper form. These documents supported the students while solving the programming tasks with the application. In this respect, the students were randomly divided into two groups: 10 students received worked examples, and the other 10 students received instructional procedures.

All students were inexperienced. At the beginning of the training session, they were asked to rate their prior programming knowledge on a scale from 1 to 5, with 1 "*no prior knowledge*", and 5 "*with programming experience*". All students rated their prior programming knowledge 2 or below ($M = 1.35$, $SD = 0.59$). Although prior programming knowledge was rated slightly higher by the students who had worked examples ($M = 1.50$, $SD = 0.71$) compared to those who had instructional procedures ($M = 1.20$, $SD = 0.42$), no significant difference was observed, $F(1, 18) = 1.328$, $p = 0.264$.

Tasks

In this study, the students worked on two tasks. In the first task, focusing on learning variables and iterative logics, the students were asked to fetch and show the name and status of each object in the *Output* panel after connecting to the server. In this task, differences between variables were demonstrated, and the students were introduced to use a *foreach* loop in order to iterate through the list of all objects. In the second task, differences between operators were demonstrated, and the students worked with loops and conditional statements. They were introduced to use *for* loops as well as several *if* statements to change the color of RGB lights based on random integers for several times.

Procedure

At the beginning of the training session, the students received a pre-questionnaire. At the end of the session a post-questionnaire was given to them in order to assess their view on ease of programming with the application. The students were randomly divided into two groups. While one group answered the pre-questionnaire, the other group was introduced to the smart home. Afterwards, the first group was introduced to the smart home, while the second group answered the pre-questionnaire. All objects and their functionalities in the smart home were explained for 20 minutes per group. Then, within each group, pairs of two students were assigned to one computer. Each computer showed a real-time full vision of the smart home during the session using three IP cameras. All students were introduced to general features of programming and the programming application for 50 minutes. Variables, data types, conditional statements, iterative logics, loops, and logical operators, as well as customized functions and primitives were explained. Before starting to work on the programming tasks, the two types of supplementary documents – namely worked examples and instructional procedures – were given to the students. Furthermore, before working on each task, they were presented a program introducing the corresponding task. The two programs respectively were (i) demonstrating the names of all available objects in the smart home, and (ii) changing the status of a dimmer light for one time. Afterwards, the students performed and completed the tasks described in the previous section. Then, they went to the smart home to see the changes in reality. Finally, they worked on the post-questionnaire.

Measures

An objective measure of the application's effectiveness is the students' success in performing the tasks. Each task consisted of several steps. Performance was operationalized by the rate of steps completed without errors. The blocks which were generated by the students were checked after the session. We labeled each block with a value and gave a final rate to the whole program based on the blocks which were correctly used and placed. Errors occurred with respect to the students' syntactic knowledge [24]; for instance, missing variable names or typing errors while using blocks. In addition, errors happened during assembling and manipulating code structure using blocks; for instance, using an *if* statement block to check a condition without using a *foreach* loop block to get all the objects names; thus, demonstrating flaws in the students' conceptual and strategic knowledge [24].

In addition, subjective data was collected in the post-questionnaire. At the end of the training session, students were asked to rate the items "is it easy to program with blocks?" and "do you think that you can easily implement ideas by programming with blocks?" using a 5 point Likert scale (with 1 "*no*", and 5 "*yes*"), and they were asked

about their preference of programming with blocks or with code syntax using a 5 point Likert scale (with 1 "*definitely with code*", and 5 "*definitely with blocks*"). Furthermore, the students were asked "do you think that it is helpful to be able to see directly in reality whether the program works as desired?", to be answered on a 5 point Likert scale (with 1 "*no*", and 5 "*yes*"). Additionally, at the end of the post-questionnaire, the students were asked to provide feedback using two questions, namely "what did you particularly like about the training session?", and "what did you particularly dislike about the training session?" in an open-ended format.

5 FINDINGS

The programming application was evaluated with respect to two aspects; accordingly, in the following, first, results concerning the ease of use are reported; second, students' performance is analyzed. Additionally, the students' feedback about the training session is presented.

Ease of Use

The results with respect to the ease of use shows that on average, the students found programming with blocks easy ($M = 4.15$, $SD = 0.75$), and they found it easy to implement ideas in block-based programming ($M = 4.15$, $SD = 0.81$). When asked about their preference of programming with blocks or with code syntax, the majority of students had a positive attitude towards programming with blocks, $M = 4.00$, $SD = 1.21$. Furthermore, the students found that being able to see the impacts of their program in reality is helpful, $M = 4.20$, $SD = 1.06$. Regarding the ease of programming with blocks depending on the type of supplementary documents, an ANOVA yielded a significant result, $F(1, 18) = 5.44$, $p = 0.031$, *partial* $\eta^2 = 0.232$, indicating that students who had worked examples ($M = 4.50$, $SD = 0.71$) found programming with blocks significantly easier compared to those who had instructional procedures ($M = 3.80$, $SD = 0.63$) of how to use the programming application. With respect to the other three items (concerning implementing ideas with blocks, preferring blocks or code syntax, and seeing impacts of the program in reality), no significant differences were obtained, all $F < 1$.

Effectiveness

Overall, the students performed 65% of the first task and 84% of the second task without errors. It means that in average, 74.5% of both tasks were completed without errors. Descriptively, in the first task, the students with worked example performed better compared to the students with instructional procedure, 77% and 53%, respectively. However, due to the small sample size ($n = 10$ dyads) no inferential statistics can be provided. With respect to the second task, both groups achieved approximately the same solution rate: 81% for students with worked example, and 87% for students with instructional procedure. In both tasks, the most common errors were setting different values to a same variable, typing errors while defining variables, and using blocks outside of loops and conditional statements where they should be placed within.

Students' Feedback

We assessed the students' feedback concerning the training session using the following open-ended questions: "what did you particularly like about the training session?", and "what did you particularly dislike about the training session?". These two questions were answered by 15 and 4 students, respectively. Table 1 shows what the students liked and disliked about the training session as well as the number of students who mentioned these aspects.

Table 1: Students' Feedback on Training Session

Students' Feedback	Number of Students
+ program by ourselves	3
+ tryout by ourselves	3
+ programming / learning about programming	4
+ liked "everything" in the training session	2
+ blocks / learning how to program with blocks	2
+ programming the smart home was very cool, smart home was great	1
– explanation at the beginning of the session was difficult to understand	3
– it was a bit complicated	1

6 DISCUSSION

In this study, the capabilities of a smart home were organized into primitives. These primitives can be applied to other smart homes using web socket communication protocol and RPC technology as a web-based interaction to control household appliances. Primitives were implemented in the application to work with all devices such as dimmable and switchable devices as well as sensors. Inexperienced, young students worked with the primitives to program and observe the impacts of their programs on a real life-size smart home. Furthermore, this application has the advantage of helping students to learn general features of programming and allowing them to program the DFKI's smart home which is also used by researchers. Generally, the students found block-based programming easy, and largely they preferred working with blocks compared to programming text code. This is supporting the results from a prior study that compared block-based with text-based programming [32]. Furthermore, students thought that they can easily implement new ideas while using blocks and found it helpful to observe the impacts of their programs in reality. These data indicate that the application is useful for young learners without prior programming knowledge. Using the proposed application, inexperienced, young students were motivated to learn and make programs to control different objects in a real life-size smart home. This is in line with findings from [7] which showed that by providing opportunities for children to change and modify the environment, they could gradually build an understanding and make rapid changes.

Google Blockly offers a framework for developers to make programming easier for inexperienced users and young learners. However, keeping in mind that introductory programming is difficult for inexperienced and young students to solve programming problems [2, 24], the visual block-based programming application may still not be fully intuitive for them. In this respect, an oral introduction to general

features of programming and the programming application was presented to the students. Furthermore, in the present work, two supplementary documents were implemented in order to help the students to work with the application and solve programming problems, namely worked examples and instructional procedures. As the students might not listen carefully to the oral explanation or might not remain fully concentrated, these documents were given to them in paper form while using the programming application.

With respect to subjective questionnaire data, the results showed that the students who had worked examples found blocks easier to program compared to the students who had instructional procedures of how to use the programming application, and the students working with examples also performed (descriptively) better in the first task compared to those working with instructional procedures. With respect to the second task, no large difference between the two groups was observed. Overall, in both groups, the performance rate in the second task – which was more complicated than the first task – was better than the performance rate in the first task, indicating substantial learning progress. Thus, when interpreting students' performance, this indicates a higher effectiveness of the application when students become more familiar with it. In line with this interpretation, errors such as setting different values to a same variable, typing error while defining variables, and using blocks outside of loops and conditional statements where they should be placed within occurred less frequent in the second task compared to the first task. This result is in line with results from [19] that young learners are capable of testing and debugging their programs when they can physically change and modify the environment.

With respect to the students' feedback about the training session, it can be noted that the students liked the training session because (i) they could program and tryout by themselves, and (ii) they learned about programming and how to program with blocks. However, some of them found it difficult to understand the information provided at the beginning of the session, indicating further need to find ways how to make the very start and the first contact with programming more accessible for young students.

Implications

Being well aware that many teachers and educators do not have immediate access to smart environments, we aim to introduce young learners to an innovative, unique environment which provides possibilities of how computer science can be connected to real-world environments. We support learners and motivate them to learn general purposes of programming. To this end, a visual block-based programming application was developed which enables learners to program and see impacts on a real life-size smart home. The results show that with this application, learners are able to program successfully by themselves. In this respect, visual block-based programming environments are suitable to simplify programming and to remove difficulties, and smart homes can provide a motivating and fascinating context for young learners.

We would like to emphasize that in our study, we directly asked school math teachers to come to the DFKI with their interested students. In this respect, teachers need to allocate a time slot in order to come to us with a group of students, limiting the possible number of students learning with the application.

Future studies with a larger sample size should focus on possible influences of young learners' gender and competence levels such as prior programming knowledge with respect to visual block-based programming environments and real life-size smart homes. In future work, based on the results obtained in this study, we plan to better adapt our training session to the students' needs in order to make it easier to understand and follow. Furthermore, students should be enabled to practice self-paced and to perform more creative tasks following their own ideas in the smart home. Including self-paced and creative tasks will also help to better adapt the training sessions to different levels of students' prior knowledge, and it becomes easier to capture the interests of the individual learners. In this respect, for instance, students shall be allowed to work with micro-controllers (e.g., Arduino or WeMos boards) and mobile robots (e.g., Turtlebot) in combination with smart homes in order to design and create their own innovative ideas. Thus, they can be supported in understanding how robots and micro-controllers can be integrated into smart homes and work with available objects.

7 CONCLUSION

This paper presents the development and evaluation of an educational block-based programming application and its primitives, enabling inexperienced, young students to program a real life-size smart home. The results show that students can fairly complete programming tasks and make real programs. Common difficulties are revealed which should be addressed in future work; mostly occurring due to different variables as well as sets of blocks which needed to be placed in loops and conditional statements.

The present study shows that inexperienced, young students have a positive attitude toward working with a visual block-based programming environment and are able to program and tryout by themselves. Furthermore, students' feedback shows that having access to a smart home is interesting. It can help them to see how the environment reacts to their program when it is following the programming principles and structures.

Still, it remains an open question whether different competence levels significantly affect the young learners' attitude towards using a visual block-based programming environment in the context of smart homes. More detailed knowledge about learners' performance and their feedback is necessary in order to adapt the learning environment to individual learning prerequisites in an optimal way. However, in order to enable young learners to experience new technologies, learn programming and see its impacts in a real life-size smart home, the present study establishes a sound basis.

ACKNOWLEDGMENTS

This work was partially 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.

SELECTION AND PARTICIPATION OF CHILDREN

Twenty students, ages 12 to 14 years old, participated in this study. Math teachers suggested the group of interested students and came with them to the German Research Center for Artificial intelligence (DFKI). Necessary permissions were taken from the school. Students' parents were informed and participation consent was obtained.

REFERENCES

[1] David Bau. 2015. Droplet, a blocks-based editor for text code. *Journal of Computing Sciences in Colleges* 30, 6 (2015), 138–144.

[2] David Bau, D Anthony Bau, Mathew Dawson, and C Pickens. 2015. Pencil code: block code for a text world. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 445–448.

[3] Andrew D Birrell and Bruce Jay Nelson. 1984. Implementing remote procedure calls. *ACM Transactions on Computer Systems (TOCS)* 2, 1 (1984), 39–59.

[4] Tim Bray. 2017. *The javascript object notation (json) data interchange format.* Technical Report.

[5] K-12 Computer Science Framework Steering Committee. 2016. K-12 computer science framework. (2016). Retrieved from http://www.k12cs.org; [Online; accessed 07-January-2019].

[6] 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.

[7] Stefania Druga, Randi Williams, Hae Won Park, and Cynthia Breazeal. 2018. How smart are the smart toys?: children and parents' agent interaction and intelligence attribution. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*. ACM, 231–240.

[8] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM computing surveys (CSUR)* 35, 2 (2003), 114–131.

[9] Ian Fette. 2018. The websocket protocol. https://tools.ietf.org/html/rfc6455. [Online; accessed 07-January-2019].

[10] 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/; [Online; accessed 10-January-2019].

[11] 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.

[12] Brian Harvey and Jens Mönig. 2010. Bringing "no ceiling" to scratch: Can one language serve kids and computer scientists. *Proc. Constructionism* (2010), 1–10.

[13] Justin Huang, Tessa Lau, and Maya Cakmak. 2016. Design and evaluation of a rapid programming system for service robots. In *The Eleventh ACM/IEEE International Conference on Human Robot Interaction*. IEEE Press, 295–302.

[14] MIT App Inventor. 2018. MIT App Inventor Homepage. http://appinventor.mit.edu/explore/. [Online; accessed 10-January-2019].

[15] MakeCode. 2018. MakeCode Homepage. https://www.microsoft.com/en-us/makecode. [Online; accessed 10-January-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] Mblock. 2018. Mblock Homepage. http://www.mblock.cc. [Online; accessed 10-January-2019].

[18] Bruce M McLaren, Tamara van Gog, Craig Ganoe, David Yaron, and Michael Karabinos. 2014. Exploring the assistance dilemma: Comparing instructional support in examples and problems. In *International Conference on Intelligent Tutoring Systems*. Springer, 354–361.

[19] David Mioduser and Sharona T Levy. 2010. Making sense by building sense: Kindergarten children's construction and understanding of adaptive robot behaviors. *International Journal of Computers for Mathematical Learning* 15, 2 (2010), 99–127.

[20] Michael Montemerlo, Nicholas Roy, and Sebastian Thrun. 2003. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In *Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on*, Vol. 3. IEEE, 2436–2441.

[21] Brad A Myers. 1986. Visual programming, programming by example, and program visualization: a taxonomy. In *ACM sigchi bulletin*, Vol. 17. ACM, 59–66.

[22] Brad A Myers. 1990. Taxonomies of visual programming and program visualization. *Journal of Visual Languages & Computing* 1, 1 (1990), 97–123.

[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] Yizhou Qian and James Lehman. 2017. Students' misconceptions and other difficulties in introductory programming: a literature review. *ACM Transactions on Computing Education (TOCE)* 18, 1 (2017), 1.

[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] Ron JCM Salden, Kenneth R Koedinger, Alexander Renkl, Vincent Aleven, and Bruce M McLaren. 2010. Accounting for beneficial effects of worked examples in tutored problem solving. *Educational Psychology Review* 22, 4 (2010), 379–392.

[27] Chris Scollo and Sascha Shumann. 1999. *Professional PHP programming*. Wrox Press Ltd.

[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] David Weintrop, Afsoon Afzal, Jean Salac, Patrick Francis, Boyang Li, David C Shepherd, and Diana Franklin. 2018. Evaluating CoBlox: A Comparative Study of Robotics Programming Environments for Adult Novices. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 366.

[30] 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.

[31] 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.

[32] 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.

[33] Rui Zhi, Nicholas Lytle, and Thomas W Price. 2018. Exploring Instructional Support Design in an Educational Game for K-12 Computing Education. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. ACM, 747–752.