Introducing a Paper-Based Programming Language for Computing Education in Classrooms

Aditya Mehrotra*
aditya.mehrotra@epfl.ch
Mobots Group EPFL

Christian Giang*
christian.giang@epfl.ch
Mobots Group EPFL
TME Lab, SUPSI-DFA

Noé Duruz
noe.duruz@epfl.ch
Mobots Group EPFL

Julien Dedelley
julien.dedelley@epfl.ch
Mobots Group EPFL

Andrea Mussati
andrea.mussati@epfl.ch
Mobots Group EPFL

Melissa Skweres
melissa.skweres@epfl.ch
LEARN Center for Learning Sciences
EPFL

Francesco Mondada
francesco.mondada@epfl.ch
Mobots Group EPFL

ABSTRACT

Past research has shown that the use of tangible programming platforms in computing education can enhance students' interest, engagement, and collaboration within workgroups. However, to this day, the adoption of such interfaces in classrooms has remained relatively scarce. This is possibly due to the expenses and efforts necessary to acquire, set up and maintain such platforms. In this context, the use of paper as a principal means of interaction represents an inexpensive and versatile solution, that additionally harnesses the prevalence of paper in classrooms. This work, therefore, introduces PaPL, an easily reproducible platform for paper-based programming languages. The platform was evaluated in two exploratory user studies. The first study aimed at investigating the interaction of over 100 senior year high school students with the platform under varying conditions of group size and usage constraints. In the second study, the platform was tested with 32 sixth-graders and 2 teachers to evaluate its usage in an authentic context. The results indicate that group size may affect active discussion and error count, while usage constraints may affect active discussion of students interacting with the platform. Moreover, the classroom study shows promising results with regard to the use of PaPL in formal education.

CCS CONCEPTS

• **Applied computing** → **Collaborative learning**; **Interactive learning environments**; • **Human-centered computing** → **User interface design**; *Empirical studies in interaction design.*

*Both authors contributed equally to this research.

ITiCSE '20, June15–19, 2020, Trondheim, Norway

ACM ISBN 978-1-4503-6874-2/20/06…$15.00
https://doi.org/10.1145/3341525.3387402

KEYWORDS

Computing education; educational robotics; group collaboration; tangible programming

ACM Reference Format:
Aditya Mehrotra, Christian Giang, Noé Duruz, Julien Dedelley, Andrea Mussati, Melissa Skweres, and Francesco Mondada. 2020. Introducing a Paper-Based Programming Language for Computing Education in Classrooms. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20), June 15–19, 2020,Trondheim, Norway.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3341525.3387402

1 INTRODUCTION

Recently, more and more countries have recognized the importance of integrating digital literacy, computational thinking, and related concepts into their school curricula. These competencies have been considered essential skills for future generations and therefore several national education systems have placed emphasis on their integration into compulsory schooling [1–4, 6]. To introduce children to fundamental computer science concepts at an early stage, usually graphical programming languages such as Scratch [19], Blockly [8], or Alice [5] are used. These interfaces aim at facilitating the learning process by providing a more structured and playful environment. Programs can be created by simple drag and drop actions, combining different blocks to control either educational robots or virtual agents on the screen.

Although such interfaces may indeed facilitate the introduction of basic computer science concepts, they also have certain limitations. In the classroom, children usually work in groups and share a device to solve programming tasks. This is both, to teach and invoke social norms, and due to limited resources. However, computers usually provide single user input devices, often resulting in unbalanced participation opportunities among group members. As shown previously [12], providing equal opportunities for multiple children to interact with a digital environment, can positively impact their levels of engagement and motivation.

In this regard, the use of tangible programming languages (TPLs) appear to be an interesting approach to address the issue. TPLs usually consist of blocks that can be assembled in the physical world

to control virtual agents or educational robots. Previous work has demonstrated that compared to purely screen-based graphical programming languages, TPLs may improve collaboration in a group [11], increase situational interest [11, 14, 20] and have a positive impact on learning [14, 24]. By enhancing the dynamics within workgroups [16], they may also better exploit socio-constructivist learning approaches. This is especially interesting for educational robotics activities, in which students usually work in small groups to solve a given problem situation [9]. Moreover, TPLs provide opportunities to implement more playful ways of learning, especially to engage the interest of younger children. By involving tangible objects, TPLs represent a more physical alternative to introducing computer science in classrooms. This might be valuable to address the concerns of many teachers and parents, who are still reluctant about the use of screens in classrooms. Despite these affordances, the use of TPLs in formal education settings is still relatively scarce. Considering the existing expenses for computers, tablets and/or educational robots, the additional effort necessary to acquire, set up and maintain such platforms can represent a key argument against their use. Although teachers have expressed appreciation for TPLs, it seems like the current drawbacks have hindered a more widespread use of such interfaces.

To promote the use of tangible programming in classrooms, this work, therefore, introduces PaPL, a platform for paper-based programming languages. Based on computer vision algorithms, it provides an inexpensive and customizable solution, that does not require schools to purchase any extra accessories. Instead, it aims at better utilizing existing technological resources such as computers and tablets with standard webcams. Moreover, by integrating paper as a principal means of interaction, it provides teachers and students with the possibility of self-fabrication and customization of the programming blocks. To this day, paper is a ubiquitous material in classrooms and it incorporates five important design principles for classroom orchestration: control, visibility, flexibility, physicality, and minimalism [7]. Considered a "generic tangible medium which can carry any kind of representation" [26], paper has been used to create user interfaces for different domains, such as office applications [25], interactive slideshows [17], physical 3D modeling for CAD [23] and vocational training of logisticians [26]. The physicality of paper-based interfaces can enable new ways to structure learning activities. For instance, it allows instructors to easily adapt usage constraints, such as limiting the set of available commands to reinforce reflection [7]. However, there has not been much work investigating the potential of paper-based interfaces for computing education, particularly in classroom settings. Moreover, though tangibles have been proven beneficial to improve collaboration, it was suggested that varying group sizes may alter the ways learners interact with tangible interfaces [21]. Indeed, the effect of group size on learning with computer technology has been widely discussed before [13]. However, this question has so far not been addressed for tangible programming interfaces. This work, therefore, aims at contributing to the current body of literature through two exploratory user studies addressing the following research questions: (1) How do group size and usage constraints affect the learners' interaction with a paper-based programming language? (2) What are the potential benefits of using a paper-based programming language in classrooms?

2 METHODS

2.1 The PaPL platform

PaPL is a platform developed to implement paper-based programming languages. In order to ensure its appropriateness for classroom use, the design of the platform was guided by development heuristics for educational robotics systems [10]. Using standard webcams of computers and tablets for computer vision algorithms, it provides a framework for developing paper-versions of existing programming languages.

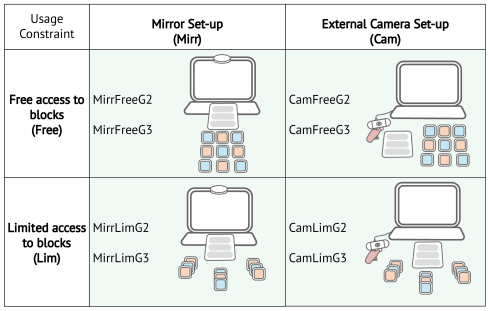

Figure 1: Schematic overview of all set-ups

The platform interprets paper blocks arranged on a programming sheet currently in two possible configurations for the camera set-up (Fig. 1). One option is to attach a mirror to the integrated webcams of laptops and direct it at the programming sheet with the paper blocks, which is placed on the laptop's keyboard (*Mirr* configuration). As an alternative, an external webcam can be connected to the laptop, which then has to be manually directed at the programming sheet with the paper blocks every time a new program needs to be captured (*Cam* configuration). On one hand, the latter acts as a usage constraint, while on the other, it also allows for more spatial freedom and may, therefore, represent a simulation for the use of tablet cameras.

For the current study, the PaPL platform was used to create a paper version of the programming language Thymio VPL [22]. Thymio VPL is an event-based, graphical programming language that was developed for the educational robot Thymio [15]. For Thymio PaPL, the event-based programming paradigm has been preserved: programs are created by associating one of the orange Event blocks with one or more of the blue Action blocks (Fig. 2). Multiple Event-Action sets, placed on different lines of the programming sheet, can be combined to realize more complex behaviors of the robot (e.g. a line follower). A detailed description would go beyond the scope of this paper, we refer the interested reader to the related work set out below. Conceived as an introductory platform preceding Thymio VPL, the programming blocks of Thymio PaPL were redesigned to make them more simple and intuitive. Programs can be loaded on the robot using a keyboard shortcut.

2.2 User study with high school students

2.2.1 Study objective. Addressing the first research question presented in the Introduction, the main objective of this user study was to explore the effects of the group size, the imposed usage

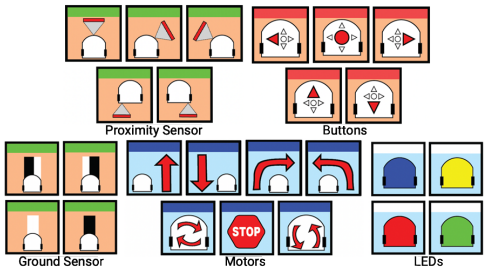

Figure 2: Thymio PaPL programming blocks

constraints and the configuration of the camera set-up on the learners' interactions with Thymio PaPL. The *Cam* configuration also represents a usage constraint, since this set-up, in contrast to *Mirr* configurations, does not facilitate a smooth sequence to load the program instructions. Thus, a three-way two-level full factorial design was implemented which involved students taking part in a programming activity using Thymio PaPL under eight different experimental conditions (Fig. 1). The levels of the three influencing factors are described hereafter. (1) The number of participants in the group: two (*G2*) or three (*G3*). (2) Usage constraint imposed on programming blocks: free access to use all blocks for any participant (*Free*) or an equal number of blocks distributed amongst the participants (*Lim*). (3) Usage constraint imposed by the configuration of the camera set-up: an integrated webcam with mirror attachment directed at the program sheet and a continuous video feed (*Mirr*) or an external webcam that has to be manually directed at the program sheet (*Cam*). To facilitate a valid comparison, all other aspects of experimental design—tasks, aesthetics, feedback, and so forth—remained constant.

2.2.2 Participants and Experimental Protocol. The study took place on two consecutive days with over 100 high school students aged between 18 and 19. Students were recruited from a university orientation event based on voluntary participation. All students gave their informed consent to participate in the study. Prior to the experimental sessions, the participants were asked about their familiarity with the Thymio robot and its programming languages. Although a few participants had heard about the robot, none of them had actively programmed it before. With six sessions on each day, a session hosted around 8-12 students for 20 minutes to conduct the activity. All sessions were organized in the same conference room, which included four experimental set-ups, each representing one of the four configurations described in Fig. 1. The set-ups were configured back-to-back so a group working on one system could not easily see the other groups. Each set-up was provided with a Thymio, a set of 22 Thymio PaPL blocks, a programming sheet and the task descriptions (Fig. 3).

An experimental session started with a five-minute introduction to the Thymio robot given by the same researcher. After the introduction, the participants were randomly grouped into dyads (*G2*) or triads (*G3*) and allotted one of the four set-ups. They were then presented the tasks and asked to solve them one after the other. The activity consisted of six tasks with increasing difficulty and a bonus task in case of completion before the end of the allotted time. The first three tasks introduced basic algorithm building concepts

Figure 3: Students interacting with the *CamFree* set-up

and were formulated using explicit instructions (e.g. Task 1: *"When I press the forward button, Thymio lights up in blue and moves forward. When I press the central button, Thymio lights up in yellow and stops"*). In contrast, the last three assignments were more complex and less explicit (e.g. Task 6: *"Thymio automatically follows a black line. At the end of the line, it turns around and continues to follow the line in the opposite direction"*). During the programming tasks, each set-up was followed by a researcher taking the role of an observer in monitoring the group's interaction with the platform and among participants. The observers did not help the groups in completing the tasks but provided support in case of technical issues. Moreover, they approved correct solutions, allowing groups to advance to the next task. After each session, the observers switched to a different set-up to reduce observational bias.

2.2.3 Measures. After the completion of the first two sessions, some modifications were applied to the experimental protocol. Specifically, it was decided to include an identical demonstrative example prior to the presentation of the tasks for all four set-ups. Moreover, all observers shared their experiences gathered so far in order to streamline the observation protocols. Due to these adjustments, it was decided that these sessions would be considered pilot runs to adjust experimental protocols and train observers for objective monitoring. The data from these groups were therefore not considered for analysis. Additionally, in order to allow for a balanced comparison between the different experimental conditions, groups from conditions with more samples were randomly excluded. Hence, the analysis was based on data from 32 groups (4 for each condition) representing 80 participants.

All programs loaded by the groups were logged for later analysis. The log files were then used to analyze the amount and type of errors made by the groups. Group collaboration was measured by the time spent in active discussion and monitored by the observers through direct observation. An active discussion was classified as any situation where one or more people of the group would be talking about the task at hand. The observations were recorded on a tablet using the mobile application Actograph (SymAlgo Technologies, Paris, France). The observers also kept track of the tasks completed by each group and gathered qualitative observations by taking written notes during the activity.

2.3 Classroom study

2.3.1 Study objective. To test the platform in an authentic context, the second experimental study focused on its use in a classroom environment. Thus, the objectives were to determine (1) the interaction of the target audience with Thymio PaPL and (2) the ability of the platform to facilitate the teacher's supervision of the programming activity.

2.3.2 Participants and Experimental Protocol. This study with 32 students and 2 teachers from two sixth-grade classes lasted around two hours. The students were 10-11 years old and parental consent was obtained prior to the study, allowing for the students' participation. Likewise, both teachers gave their informed consent to participate in the study. None of the students or teachers reported prior experience with Thymio or its programming interfaces. The experiment started with a plenary session, during which all students were given a five-minute introduction to the robot. Subsequently, the students were divided into groups of 2-3 (within their class) and given 25 minutes to explore the robot's pre-programmed modes under the supervision of one researcher. Meanwhile, 6 laptops with Thymio VPL were prepared in one room, while in the other room, 6 Thymio PaPL set-ups were provided in the *CamFree* configuration (Fig. 1). Another researcher briefed both teachers on the use of the Thymio PaPL and Thymio VPL interfaces. The teachers were also presented the solution sheets for the subsequent activities.

The exploration activity ended with a Parson's Puzzle [18] containing five Thymio programming questions. Each question described a program specification, for which the students had to determine the correct combination of Thymio programming blocks. The puzzles were attempted by each student individually. While the students of one class had to solve the questions using Thymio PaPL blocks (class A), the other class was presented the same questions with Thymio VPL blocks (class B). The questions were similar to the three easy ones from the first study (see Section 2.2.2). After this, the teachers and their students were sent to the classrooms with the set-up corresponding to their Parson's puzzle (i.e., class A to the Thymio PaPL set-ups and class B to the Thymio VPL set-ups). In the next five minutes, the students were given a short introduction to the respective interface by a researcher. This was followed by 20 minutes of programming activity, during which the students were asked to solve the same five tasks as presented in the Parson's Puzzle.

The whole activity was administered by the teachers, who were asked to conduct this as a regular class. In both classrooms, one researcher took the role of an observer and another, the role of technical support, whenever required. Following the programming activity, both classes were asked to complete the same Parson's Puzzle as before. Finally, in the last part of the experimental session, both classes switched set-ups, i.e. class A now worked with Thymio VPL and class B with Thymio PaPL. The students were given two new tasks, with a difficulty comparable to the last three questions of the first study (see Section 2.2.2). Again, both classes were introduced to the new platform and then given 20 minutes to solve the tasks.

2.3.3 Measures. The scores obtained in the Parson's Puzzles were used as a measure for the intuitiveness of the Thymio PaPL/VPL platforms. Each correct answer was awarded one point, leading to a maximum possible score of sixteen points. Parson's Puzzle scores were compared before (Pre) and after the programming activity (Post) to analyze improvements in both groups. The activity of the teachers was tracked by an observer on Actograph. Specifically, observers monitored the interaction of the teacher with the groups during the whole duration. The teachers were not informed about these observations beforehand to prevent any change in behavior. Five days after the experimental study, both teachers met with the researchers for a 20-minute interview.

Table 1: Task completion for each condition

Set-up condition	*G2 (Dyads)*		*G3 (Triads)*	
	Tasks 1-5	Task 6	Tasks 1-5	Task 6
MirrFree	4/4	2/4	4/4	4/4
MirrLim	4/4	2/4	4/4	4/4
CamFree	3/4	2/4	4/4	2/4
CamLim	4/4	2/4	4/4	3/4

3 RESULTS

3.1 User study with high school students

Table 1 summarizes the number of groups in each condition that were successful in finishing the presented tasks. While tasks 1-5 were completed by all groups except for one in the *CamFreeG2* condition, the sixth task proved itself as a challenge. This more complex assignment, in which the students had to program a line follower with Thymio, required the participants to transfer their learnings from the first five tasks. On progressing to the sixth task, most groups needed multiple trials to test their programs. The conditions for dyads (*G2*) showed a similar success rate for each condition with only two groups providing the correct solution. This, however, was more diverse for the triads (*G3*). All four groups in *MirrFree* and *MirrLim* and three groups in *CamLim* condition were successful in completing all tasks. Though, only two groups in *CamFree* condition finished all tasks.

Further to quantify performance, the errors made by the groups were analyzed using the program log files saved during the activity. Following this analysis, errors were classified into six categories based on the frequency of occurrence. (1) EA: Event-Action association errors (e.g. an Action block without any associated Event block), (2) EEA: Use of multiple Events for a single Action (e.g. *ground sensors detect white* block and *ground sensors detect black* block with *move forward* block), (3) EAA: Multiple Actions of the same category associated with a single Event (e.g. *center button* block with *move forward* block and *turn right* block), (4) GR: Errors related to the incorrect use of the infrared ground sensor, (5) PGR: Errors related to misinterpretation of proximity for ground sensor, and (6) INC: Loading an incomplete solution on the robot. From the error analysis, it was observed that dyads tended to commit more errors than triads (Fig. 4).

While groups in the *G2* conditions committed 21-26 total errors, *G3* groups only made 16-20 errors overall. Some conditions appeared to be particularly favorable for *G3* groups. In *MirrFree*

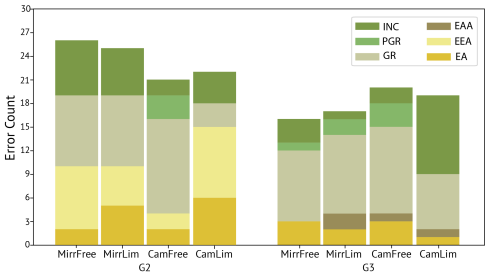

Figure 4: Error count by category

Table 2: Active discussion in percentage

Set-up condition	*G2 (Dyads)*		*G3 (Triads)*	
	Mean	SD	Mean	SD
MirrFree	81.4	21.2	69.9	37.4
MirrLim	89.7	5.7	91.9	9.2
CamFree	83.1	18.3	82.4	20.2
CamLim	89.1	11.3	95.3	5.8

and *MirrLim* settings, *G3* groups made around one-third fewer errors than *G2* groups in the same settings. In contrast, *CamFree* and *CamLim* settings did not show strong differences between *G2* and *G3* groups. Remarkably, a significant number of "EEA" errors were found for all *G2* groups, while this type of error was not found for *G3* groups.

Based on the observation data, the active discussion state during the activity was analyzed for all groups in the eight conditions (Table 2). On average, groups in *MirrLim* and *CamLim* tended to have more discussions for both dyads and triads. In contrast, groups in *MirrFree* and *CamFree* settings appeared to be less communicative. Moreover, the variance among groups in these conditions was much larger compared to the other two conditions.

3.2 Classroom study

Parson's Puzzle scores from students in both classes are shown in Fig. 5. Class A reached much higher scores (15.4±1.2 points) than Class B (9.6±4.6 points) before the programming activity. A Mann-Whitney U test validated a statistically significant difference ($p<0.01$). After performing the tasks with the robot, the performance of both classes improved. While all students of Class A were able to achieve a perfect score (16 points), students in Class B still had varying results (12.6±5.0 points). Though the difference between both groups decreased, it was still statistically significant ($p<0.01$).

The distribution of time spent with each group was evaluated for both teachers based on observational data (Fig. 6). Teacher A, starting with the Thymio PaPL platform, spent around half of their time attending to particular groups. While some groups needed more attention (e.g. group 2), others needed less (e.g. group 5). Throughout the 20 minutes of activity, the teacher followed all groups at least once. Moreover, the other half of the time, the teacher was not engaged with any specific group but monitoring the classroom in general. When switching to the Thymio VPL platform, the teacher's behavior changed drastically. More than half of the time was now

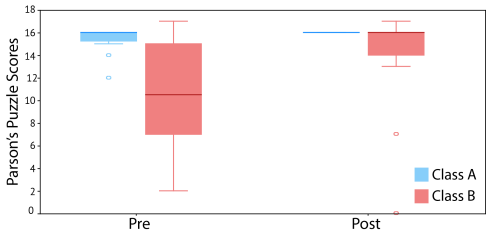

Figure 5: Parson's Puzzle scores before (Pre) and after (Post) the programming activity

spent attending to a specific group. In this activity, only two groups had direct supervision of the teacher and the time spent in overall classroom monitoring decreased with respect to the first activity. Teacher B, who first performed the activity with Thymio VPL showed a similar trend in behavior. In the VPL set-up, the teacher spent almost all of their time with particular groups and only a small fraction of their time on overall classroom monitoring. Yet only 5 out of 6 groups benefited from direct supervision. When switching to the PaPL platform, the teacher now spent significantly more time on overall classroom monitoring and in addition, all groups were now receiving teacher attention.

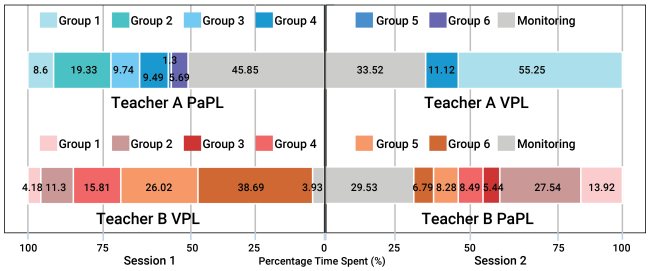

Figure 6: Proportion of time teachers spent with particular groups during the programming activities

Five days after the study, the two teachers met the researchers again for a 20-minute interview. During the interview, the teachers were asked to reflect on the activities with both platforms and share their thoughts. The group discussion was moderated by a researcher, who prepared the questions beforehand. As a main result, both teachers agreed that the PaPL platform was easier to understand and therefore particularly suited for beginners. Teacher A suggested that introducing PaPL first, could help students build a more concrete foundation before going over to the more complex VPL platform. Moreover, they acknowledged that PaPL has more potential to promote collaboration and communication. They stated that when using PaPL, students had to argue more before programming and it would encourage more experienced students to explain things to the less experienced ones. The teachers appreciated the fact that their activity was monitored and showed great interest in the analysis of their behavior. Teacher B stated that during the programming activities they applied a "fire-fighting" strategy, always addressing the next group that asked for support. Whenever the teacher felt that all of their students could work independently, they changed

to the overall classroom monitoring state. Finally, the teachers also acknowledged that all students should get an appropriate, however, not necessarily equal, amount of the teacher's attention. In this context, they agreed that the PaPL platform allowed them to be more balanced since it provided a well-framed environment. In contrast, the VPL platform, providing more functionalities, may appear more overwhelming, especially for teachers who have less experience with computer technology.

4 DISCUSSION AND FUTURE WORK

The goal of this work was to study the usage of a paper-based programming language in computing education. Two exploratory studies examined different aspects of the devised platform.

In the first study, participants were asked to perform programming tasks using the platform under different conditions. The results showed that both group size and usage constraints may influence the way students interact with the platform. Triads appeared to be more successful in the completion of difficult tasks compared to dyads. Moreover, they also committed fewer errors during programming activities. This result is in line with the findings of Lou et al. [13] who suggested that groups of three to five members achieve better task performance in comparison to dyads. However, the authors also highlighted that group size larger than two did not have significant positive effects on individual learning. Instead, for students learning in pairs, there was a small but significant positive effect. Lou et al. argued that "the difference may be due to the physical constraints associated with computer use". It might be that the physicality of the PaPL platform helps to overcome these limitations, promoting both group collaboration and individual achievement for groups larger than two, as seen from their performance. Capitalizing on socio-constructivist learning approaches, the PaPL activities may encourage students to compare alternative solutions and correcting each other's misconceptions. The analysis of the errors showed that the "EEA" category occurred in all settings for dyads. This was not the case for triads in any setting. This suggests that the inclusion of another participant to the group provides more thorough scrutiny to this specific error type, the slip-up of which is assumed to be more probable for the given tasks.

The tangibility of the interface may also favor exploratory behavior which can often lead to certain group members taking charge and leading others. In some cases, introverted or less experienced students tend to lose interest and fall short in individual learning. Therefore, one of the constraints imposed on the students in the high school study was access to the blocks. Putting usage constraints on paper-based user interfaces has been applied before to encourage reflection in the training of logistics apprentices [7]. In the present study, imposing fixed assignments of programming blocks to students appeared to increase active discussion time among the group. In contrast, the behavior of groups, where no constraint was imposed on the blocks, was more varying and thus, less predictable. This condition could, therefore, represent an interesting approach to foster collaboration and reflection in computer education, especially in cases when some group members are more introverted or less experienced. PaPL supports the practice of such constraints, which teachers cannot easily implement with purely screen-based solutions.

As a way to explore another usage constraint, two different camera set-ups were tested. Interestingly, *Mirr* configurations seemed to be especially effective for triads leading to higher task completion rates and lower error count. The configurations, however, did not have the same effect on dyads. Therefore, group size may also have an effect on how different configurations are utilized and should be carefully considered when tangible programming interfaces are used. On the other hand, the difference in error count and completion rate in *Cam* configurations is not remarkable which suggests that the constraint may also flatten the effects of group size.

The results of the second study demonstrated strong potential in regards to the use of paper-based programming languages in classrooms. Even before the programming activities, students presented to the Thymio PaPL blocks were able to attain high scores in the Parson's Puzzle tests. After the activity, they further improved and all students achieved maximum scores. On the other hand, students working with the screen-based Thymio VPL blocks reached lower scores in both tests. Providing an intuitive platform with "low floors" was argued to be an essential design element in computing education [19]. This might be particularly important to promote motivation, interest, and self-efficacy in novices. The interview with the teachers suggested that introducing Thymio PaPL prior to Thymio VPL could help students build better foundations and may, therefore, represent a smoother entry point to computing education. The teachers further acknowledged the potential of the PaPL platform to promote collaboration and communication. Thanks to its simplicity and intuitiveness, the platform may also support inexperienced teachers in running and orchestrating computing education lessons. The observations of the teachers' activities showed that both teachers provided more balanced support to all groups when they were working with the PaPL platform. Furthermore, they also had more time for overall classroom monitoring. This allows them to diagnose the progress of all the groups and provide more support where necessary.

Nevertheless, the results of the studies presented in this work are subject to the limitations of the small sample sizes and particularly the brevity of the interventions. In order to draw more substantial conclusions, future studies should include larger samples, longer interventions as well as methods to evaluate the effective learning gains in the long-term. Such studies are needed to provide real evidence about the advantages of paper-based programming languages for computing education. Moreover, it should be investigated to what extent such approaches can be favorable and when should the transition to more complex systems be considered. Finally, this work has explored the PaPL framework for event-based programming. However, it could also be studied how PaPL can be used for sequential programming paradigms involving control structures such as loops and conditions.

ACKNOWLEDGMENTS

The authors would like to thank all students and teachers that participated in the study. This work has been partially supported by the Swiss National Science Foundation through the National Centre of Competence in Research Robotics, Gebert Rüf Stiftung through "CTTS: Computational Thinking Tools for Schools" and Hasler Stiftung.

REFERENCES

[1] G Berry, G Dowek, S Abiteboul, JP Archambault, C Balagué, GL Baron, C de la Higuera, M Nivat, F Tort, and T Viéville. 2013. L'enseignement de l'informatique en France Il est urgent de ne plus attendre. *Rapport de l'Académie des sciences* (2013).

[2] Stefania Bocconi, Augusto Chioccariello, and Jeffrey Earp. 2018. The Nordic approach to introducing Computational Thinking and programming in compulsory education. *Report prepared for the Nordic@ BETT2018 Steering Group. doi: https://doi. org/10.17471/54007* (2018).

[3] Royal Society (Great Britain). 2012. *Shut down or restart?: The way forward for computing in UK schools.* Royal Society.

[4] European Commission. 2016. A new skills agenda for Europe: Working together to strengthen human capital, employability and competitiveness.

[5] Stephen Cooper, Wanda Dann, and Randy Pausch. 2003. Teaching objects-first in introductory computer science. In *ACM SIGCSE Bulletin*, Vol. 35. ACM, 191–195.

[6] Bundesrat der Schweizerischen Eidgenossenschaft. 2017. Bericht über die zentralen Rahmenbedingungen für die digitale Wirtschaft. *Schweizerische Eidgenossenschaft. Abgerufen von https://www. newsd. admin. ch/newsd/message/attachments/46892. pdf* (2017).

[7] Pierre Dillenbourg. 2013. Design for classroom orchestration. *Computers & Education* 69 (2013), 485–492.

[8] Neil Fraser. 2015. Ten things we've learned from Blockly. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)*. IEEE, 49–50.

[9] Christian Giang, Morgane Chevalier, Lucio Negrini, Ran Peleg, Evgeniia Bonnet, Alberto Piatti, and Francesco Mondada. 2018. Exploring escape games as a teaching tool in educational robotics. In *International Conference EduRobotics 2016*. Springer, 95–106.

[10] Christian Giang, Alberto Piatti, and Francesco Mondada. 2019. Heuristics for the Development and Evaluation of Educational Robotics Systems. *IEEE Transactions on Education* (2019).

[11] Michael S Horn, Erin Treacy Solovey, and Robert JK Jacob. 2008. Tangible programming and informal science learning: making TUIs work for museums. In *Proceedings of the 7th international conference on Interaction design and children*. ACM, 194–201.

[12] Kori M Inkpen, Wai-ling Ho-Ching, Oliver Kuederle, Stacey D Scott, and Garth BD Shoemaker. 1999. This is fun! we're all best friends and we're all playing: supporting children's synchronous collaboration. In *Proceedings of the 1999 conference on Computer support for collaborative learning*. International Society of the Learning Sciences, 31.

[13] Yiping Lou, Philip C Abrami, and Sylvia d'Apollonia. 2001. Small group and individual learning with technology: A meta-analysis. *Review of educational research* 71, 3 (2001), 449–521.

[14] 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.

[15] Francesco Mondada, Michael Bonani, Fanny Riedo, Manon Briod, Léa Pereyre, Philippe Rétornaz, and Stéphane Magnenat. 2017. Bringing robotics to formal education: The thymio open-source hardware robot. *IEEE Robotics & Automation Magazine* 24, 1 (2017), 77–85.

[16] Andrea Mussati, Christian Giang, Alberto Piatti, and Francesco Mondada. 2019. A Tangible Programming Language for the Educational Robot Thymio. In *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*. IEEE, 1–4.

[17] Les Nelson, Satoshi Ichimura, Elin Rønby Pedersen, and Lia Adams. 1999. Palette: a paper interface for giving presentations. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*. ACM, 354–361.

[18] Dale Parsons and Patricia Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. Australian Computer Society, Inc., 157–163.

[19] 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.

[20] Theodosios Sapounidis, Stavros Demetriadis, and Ioannis Stamelos. 2015. Evaluating children performance with graphical and tangible robot programming tools. *Personal and Ubiquitous Computing* 19, 1 (2015), 225–237.

[21] Bertrand Schneider, Patrick Jermann, Guillaume Zufferey, and Pierre Dillenbourg. 2010. Benefits of a tangible interface for collaborative learning and interaction. *IEEE Transactions on Learning Technologies* 4, 3 (2010), 222–232.

[22] Jiwon Shin, Roland Siegwart, and Stéphane Magnenat. 2014. Visual programming language for Thymio II robot. In *Conference on Interaction Design and Children (IDC'14)*. ETH Zürich.

[23] Hyunyoung Song, François Guimbretière, Chang Hu, and Hod Lipson. 2006. ModelCraft: capturing freehand annotations and edits on physical 3D models. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*. ACM, 13–22.

[24] Amanda Strawhacker, Amanda Sullivan, and Marina Umaschi Bers. 2013. TUI, GUI, HUI: is a bimodal interface truly worth the sum of its parts?. In *Proceedings of the 12th International Conference on Interaction Design and Children*. ACM, 309–312.

[25] Pierre Wellner. 1991. The DigitalDesk calculator: tangible manipulation on a desk top display. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*. ACM, 27–33.

[26] Guillaume Zufferey, Patrick Jermann, Aurélien Lucchi, and Pierre Dillenbourg. 2009. TinkerSheets: using paper forms to control and visualize tangible simulations. In *Proceedings of the 3rd international Conference on Tangible and Embedded interaction*. ACM, 377–384.