The Impact of Guidance and Feedback in Game-Based Computational Thinking Environments

Sven Manske [1], Alexia Feier, Philip Frese, Pia Hölzel, Maurice Iffländer Rodriguez, Joshua Körner, Aron Lichte, Lena Otto de Mentock, Melinda Kocak, Natalia Szymczyk, Dilan Temel, Mathis Haefs, Nina Kersting, Rebekka C. Liewald, Daniel Bodemer [2],

Abstract: In this paper we investigated the impact of feedback and guidance on the development of computational thinking skills. To achieve this, we extended a game-based learning environment that aims to foster computational thinking by teaching programming in self-regulated learning scenarios. The learning environment has been enriched with multiple mechanisms to guide learners and provide feedback that is directed towards the development of computational thinking skills, particularly specific abstractions in programming among algorithmic thinking. To assess the impact of guidance and feedback, we conducted an empirical study with 57 participants. The findings indicate that feedback on the logical artifacts can reduce certain code smells and increase the motivation on the part of the learners.

Keywords: Computational Thinking, Game-Based Learning, Programming, Guidance.

1 Introduction

In her fundamental work, Wing [Wi06] defined Computational Thinking (CT) as an "fundamental skill for everyone, not just for computer scientists." According to Wing [Wi14], it is the "thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer ... can effectively carry out". Due to the increasing popularity of CT as a concept in education throughout the last decade [PSB17, GP13], visual block-based programming emerged as the predominant paradigm in teaching CT skills [Gr17], with Scratch being one of the most prominent representatives of this domain [Re09]. In the tradition of the educational programming language Logo [Pa80], Scratch has many facilities for learners to express themselves through programming and to create logical artifacts that solve real problems without the difficulty to learn a big set of syntactical constructs. Still, it conveys relevant concepts of computer science and programming, such as events, scene graphs, abstractions such as loops, and message passing. Scratch projects consist of a microworld ("stage") with sprites that can be programmed by the learners. However, prior research has shown that Scratch might even foster bad programming habits [Me11], which can be improved with well-designed feedback and assessment mechanisms [Mo15]. Particularly for Scratch projects, the Dr.

[1] University Duisburg-Essen, Duisburg, sven.manske@uni-due.de, https://orcid.org/0000-0002-5098-1682

[2] University Duisburg-Essen, Duisburg, daniel.bodemer@uni-due.de, https://orcid.org/0000-0003-2515-683X

Scratch system has been developed to assess CT skills based on the automatic analysis of learners' programming artifacts, namely the scratch projects created [Mo15]. The importance of guiding learners has been emphasized since the early years of Logo which is underpinned by research in promoting constructivist learning [LT97, Ma04, Da01].

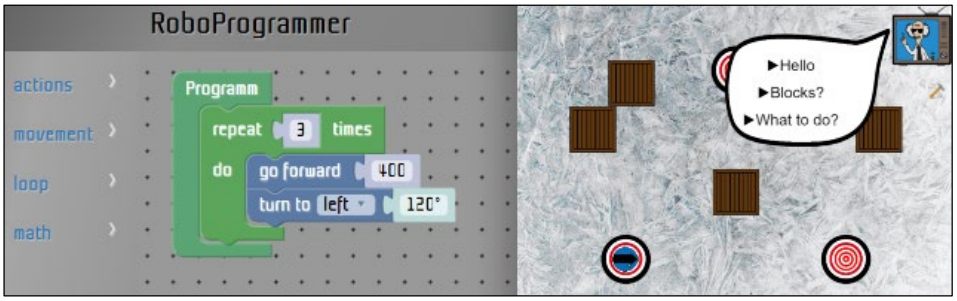

Fig. 1: The game environment of ctGameStudio consists of a microworld and a programming tool.

However, the last year of distant teaching in the COVID-19 situation has shown that self-regulated learning environments can overcome some of the burdens in the educational system. The game-based environment *ctGameStudio* aims at introducing CT concepts in a playful and guided way based on visual block-based programming [WMH18, We18]. In contrast to Scratch, the environment is not an open sandboxed, but it guides learners through an already fixed scenario with a pre-structured learning trajectory. Each stage focuses systematically on building a specific abstraction (such as variables, conditions, loops, events, or functions). Figure 1 shows the environment that is split into the code editor with the block-based visual programming language and the microworld with a virtual robot and a virtual companion on the right-hand side. This follows Logo's turtle graphic regarding the "ego-body-syntonic approach" [Pa80, AD86].

For this work, we extended *ctGameStudio* with a feedback component to support reflection on and evaluation of the logical artifacts created by the learners. Following the approach of Dr. Scratch, we developed a set of automated indicators that are used to assess computational thinking skills and to support learners' reflection. The automated assessment is based on the analysis of code artefacts, whereas the guidance is intended to support evaluation processes to foster computational thinking skills. Finally, we present the results of an empirical study with 57 participants.

2 Guidance and Feedback in ctGameStudio

The conceptualization of the guidance and feedback in the extension of ctGameStudio for this work is based on the (1) introduction of the environment with an interactive tutorial, (2) the feedback tool, and (3) prompts to support learners when they are stuck (i.e., cycles, restarting on "overshooting" the goal).

The *feedback tool* appears as soon as a level has been successfully completed. It consists of three components with performance- and CT-oriented indicators. The first component is the overview component which is presented after successfully completing a level. A CT score as an aggregative measure comprised of the different metrics was developed to give an overall understanding of the performance and to provide comparability between learners for the study. The metrics include among others the number of code smells detected, e.g., dead code or duplicated blocks. In addition, the feedback contains dynamic measures that are evaluated at runtime, such as the time to solve a level, the number of tries, or the number of blocks visited. The latter indicates the runtime behaviour in analogy to the vLOC metric ("visited lines of code") [Ma2014]. To support the evaluation of the own artefacts, each of the metrics is connected to the logical artefact. Thus, the learner might explore the metrics and get immediate feedback on the specific portion of the code through highlights and (textual) explanations. Fig. 2 shows the connection between metrics (left), highlighted code (middle) and the explanations (right).

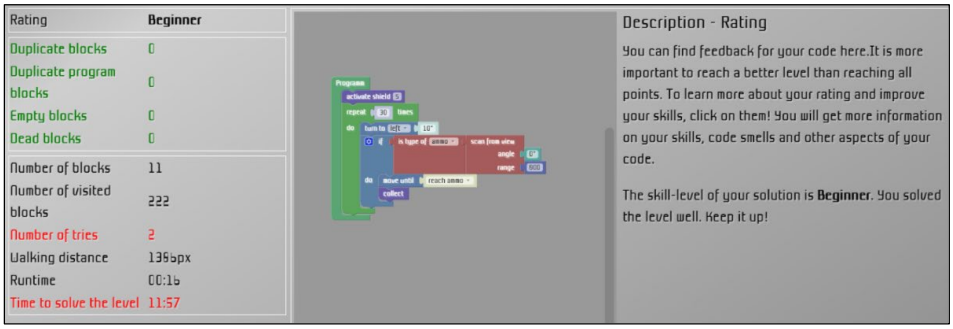

Fig. 2: The feedback tab supports learners' reflection and introspection on their code artefacts.

3 Evaluation

The aim of this exploratory evaluation is to measure the effect of the feedback and guidance components described above on certain performance parameters. The experimental group (labelled "1"), had access to the feedback component and prompts, while for the control group (labelled "0"), all guidance components were disabled. We hypothesize, that the learners of the guided version of *ctGameStudio* show increased performance parameters compared to those of the control group.

The experiment was conducted in an online setting, where 57 people participated in this experiment (21 men; 36 women; mean age: $M = 22.23$, $SD = 3.98$). Participants were given the task to interact with the learning environment for 45 minutes. Each participant had the possibility to contact the test instructor if they encountered technical problems during the study. However, additional hints on how to solve the tasks were not given. For the control group, all the guidance components were removed. However, participants of

both groups had the opportunity to get additional information about the individual programming blocks and predefined methods in the block lexicon.

To measure the performance of the learners, the following features were extracted from each code artefact: *number of dead blocks*, *number of duplicate blocks*, *number of levels completed*, *time to solve a level*, *runtime*, *number of tries to solve a level* and the *ctscore*.

4 Results

The analysis of the mean differences of the different parameters were determined with a t-test. As seen in table 1, it is noticeable that the measured parameters mostly performed better in the group with adjusted feedback and guidance components. The score *number of duplicate blocks* is significant *(t*(28.88) = 2.60, *p* = .015) with a mean difference of 0.44 blocks. The mean differences between the groups are particularly small in the categories *ctscore* (*t*(55) = -.35, *p* = .727, *MD* = -1.10) and *runtime* (*t*(53) = 1.12, *p* = .268, *MD* = *1.03 seconds*). The scores of *time to solve* (*t*(53) =.19, *p* = .850, *MD* = *12.03 seconds*) and *level won count* (*t*(38.92) = 1,27, *p* =.213, *MD* = 0.78) were better in the control group but could not reach significance.

Tab.: 1

Outcomes Performance Parameters Analysis

User Statistics	Exp_version	*M*	Mean	*p*
ctscore	0	83.71	-1.10	.729
	1	84.81		
Number of dead blocks	0	0.80	0.55	.141
	1	0.25		
Number of duplicate blocks	0	0.49	0.44	.015
	1	0.05		
Level won count	0	7.03	0.78	.213
	1	6.25		
Runtime	0	9.10	1.10	.265
	1	8.00		
Time to solve	0	173.20	-12.00	.850
	1	185.20		
Tries	0	6.62	0.11	.896
	1	6.51		

Note. Time is given in seconds.

5 Conclusion

In this paper we presented an extension of the game-based learning environment *ctGameStudio*. The new version includes feedback and guidance components that analyse the learner's code and provide appropriate guidance that improves learning, particularly the inspection, reflection, and evaluation of own logical (code) artefacts. We hypothesized that this would result in higher performance regarding computational thinking.

The results regarding the differences in the performance parameters between two groups showed no significant improvement in most performance parameters. However, the significant mean difference in *number of duplicate blocks* shows that the concepts are well explained to the participants and that the acquired knowledge can be transferred. In addition, there is a tendency for the feedback group to take a bit more time and complete slightly fewer levels, which can be explained by the fact that the participants had to deal with the feedback and interact with the component. However, these differences are not significant. Additionally, a significant improvement of the flow could be observed in the feedback group. Flow is repeatedly described in literature as a particularly important factor in the learning process, so that this result is an indication that the changes to the game experience were effective.

Since there is a considerable amount of literature on the benefits of formative and summative feedback in gaming-based learning environments, e.g. [Gr18, LK11], the reason for the small group differences regarding the performance of the learners may be due to the implementation of the feedback system and the design of the study. Even though there were small performance improvements in the feedback group, there were no significant differences in the *ctscore*. The ratios of the parameters for the calculation of the score may need to be revised. Although, *ctGameStudio* is created for first-time game-based learning of CT the participants of the study had a high heterogeneity in terms of programming experience, which may have led to mixed results. Since the game was only played for 45 minutes and it takes extra effort to read the feedback dialogs, the benefits might only be realized with longer observation.

6 Bibliography

[AD86] Abelson, H.; diSessa, A.: Turtle Geometry: The Computer as a Medium for Exploring Mathematics, 1st ed., MIT Press, Cambridge, 1986.

[Da01] Dalgarno, B.: Interpretations of constructivism and consequences for Computer Assisted Learning. In: *British Educational Research Association*, pp.183-194, 2001.

[Gr18] Groff, J. S.: The potentials of game-based environments for integrated, immersive learning data. In: *European Journal of Education*, 53(2), pp. 188-20, 2018.

[GP13] Grover, S.; Pea, R.: Computational Thinking in K–12. In: *Educational Researcher*, 42(1), pp. 38–43, 2013.

[Gr17] Grover, S.: Assessing Algorithmic and Computational Thinking in K-12: Lessons from a Middle School Classroom. In (Rich, P. J.; Hodges, C. B. eds.): Emerging Research, Practice, and Policy on Computational Thinking. *Educational Communications and Technology: Issues and Innovations*, Springer, Chalm, pp. 269-288, 2017.

[LK11] Lee, M. J.; Ko, A. J.: Personifying programming tool feedback improves novice programmers' learning. In: *Proceedings of the seventh international workshop on Computing education research* - ICER '11, pp. 109-116, 2011.

[LT97] Lee, M. O. C.; Thompson, A.:Guided Instruction in Logo Programming and the Development of Cognitive Monitoring Strategies among College Students. *Journal of Educational Computing Research*, 16(2), pp. 125–144, 1997.

[Ma04] Mayer, R. E.: Should there be a three-strikes rule against pure discovery learning? The case for guided methods of instruction. *The American Psychologist*, 59(1), pp. 14-19, 2004.

[Ma14] Manske, S., & Hoppe, H. U. (2014). Automated indicators to assess the creativity of solutions to programming exercises. In *2014 IEEE 14th ICALT* (pp. 497-501).

[Me11] Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011, June). Habits of programming in scratch. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 168-172).

[Mo15] Moreno-León, J., & Robles, G. (2015, November). Dr. Scratch: A web tool to automatically evaluate Scratch projects. In *Proceedings of the workshop in primary and secondary computing education* (pp. 132-133).

[Pa80] Papert, S.: Mindstorms: Children, Computers and Powerful Ideas. Basic Books, New York, 1980.

[PSB17] Pugnali, A.; Sullivan, A.; Bers, M. U: The impact of user interface on young children's computational thinking. *Journal of Information Technology Education: Innovations in Practice*, 16, pp. 171-193, 2017.

[Re09] Resnick, M.; Maloney, J.; Monroy-Hernández, A.; Rusk, N.; Eastmond, E.; Brennan, K.; Millner, A.; Rosenbaum, E.; Silver, J.; Silverman, B.; Kafai, Y.: Scratch: programming for all. In: *Communications of the ACM*, 52(11), pp. 60-67, 2009.

[We18] Werneburg, S.; Manske, S.; Feldkamp, J.; Hoppe, H. U.: Improving on Guidance in a Gaming Environment to Foster Computational Thinking. In: *Proceedings of the 26th International Conference on Computers in Education*, Philippines, pp. 676-685, 2018.

[WMH18] Werneburg, S.; Manske, S.; Hoppe, H. U.: ctGameStudio - A Game-Based Learning Environment to Foster Computational Thinking. In: *Proceedings of the 26th International Conference on Computers in Education*, Philippines, pp. 543-552, 2018.

[Wi06] Wing, J. M.: Computational Thinking. In: *Communications of the ACM*, 49(3), pp. 33-35, 2006.

[Wi14] Wing, J. M.: Computational Thinking Benefits Society. In: *Social Issues in Computing 40th Anniversary Blog*, New York, 2014.