DEVELOPING COMPUTATIONAL THINKING THROUGH GAMES IN SCRATCH

Irena Nančovska Šerbec, Špela Cerar, Alenka Žerovnik

University of Ljubljana, Faculty of Education, Ljubljana, Slovenia
irena.nancovska, spela.cerar, alenka.zerovnik@pef.uni-lj.si

***Abstract**: Computational thinking (CT), as a paradigm for learning computer science among young students, has seen a large increase in popularity during recent years. Mainly it is described as a problem-solving process that includes logical, analytical, algorithmic thinking and dispositions, such as the ability to confidently deal with complexity and open-ended problems. The goal is to inspire and engage the learners in such a way that they get an idea of concepts on a concrete operational level. Its development could be supported by different activities, including variety of curriculum approaches, workshops, summer schools, seminars etc. In the paper we analyse the projects developed by primary school pupils during the constructionist workshops and the projects developed by students, prospective computer science teachers. We were interested in how the students understand computational concepts, as a dimension of CT, reflected in their projects. We quantitatively assessed projects with Dr. Scratch tool, which checks the presence of the previously mentioned concepts and offers a feedback report with ideas and tips how to improve the code, aiming to encourage students' desire to keep on developing their programming skills. We compare the learning outcomes of two different projects: Maze and Escape Room. Our focus is on differences in thinking between pupils and students.*

***Keywords**: computational thinking, Scratch, evaluation.*

1. Introduction

Computational thinking (CT), a term coined by Papert in 1996, and reused in 2006, by Wing, has seen a large increase in popularity during recent years. There is no agreed upon definition of CT, but it could be considered as a problem-solving process that includes a number of characteristics, such as logical, analytical, algorithmic thinking and dispositions, such as the ability to confidently deal with complexity and open-ended problems. The goal is to inspire and involve the learners in such a way that they get an idea of computer science concepts on a concrete operational level.

In the paper we described how pupils and students develop CT through projects in educational programing environment, Scratch. Scratch, developed at MIT Media Lab, is the world's leading coding platform for kids aged 8 to 16. It already has over 15 million registered users and over 22 million shared projects (Roque, Rusk & Resnick, 2016).

We used two project activities: Maze and Escape Room game programing. In our research are involved pupils aged 8 to 12 and students, prospective computer science teachers. By means of qualitative analyses we outline the differences in pupils and prospective teachers thinking during solving programming problems.

2. Theoretical background

CT has been widely accepted and promoted both as the skill set that programmers develop, and as the general thinking skills computer scientists should develop as they learn the discipline. The first approach is based on the idea that CT skills should ultimately be evident in programs written.

In the current section we describe the meaning and dimensions of the CT assessed through programming projects in Scratch. We describe foundations of project-based learning.

2.1. Computational Thinking

CT was first mentioned by Papert (1996) to suggest a way to efficiently apply novel approaches to problem solving. Papert talked about students using computers as instruments for learning, developing creativity, innovation, and learning skills of CT. The explanation of the term came later in the year 2006, when Jannette Wing wrote an article describing the term of CT.

CT is very wide and comprehensive term that includes many fields. Wing (2006) defines it as a range of mental tools that involves solving problems, designing systems, and understanding human behaviour, using the fundamental concepts of computer science. She predicts that CT will be taught in schools as basic skill such as reading, writing, and arithmetic.

Brennan and Resnick suggest the following key dimensions of CT framework: computational concepts (iteration, parallelism, etc.), computational practices (e.g., debugging projects or remixing others' work) and computational perspectives (reflective thinking about coding) (Brennan & Resnick, 2012). The focus of our research is on the computational concepts.

Brennan and Resnick (2012) separate seven CT concepts.

- Sequences: In programming it is very important that the activity or task is expressed as a sequence of individual steps or commands that can be performed by the computer. They should follow gradually and in the correct order, like a cooking recipe.
- Loops: Loops are a kind of mechanism that enables us to execute multiple sequences and/or execute sequences indefinitely or until some condition is met.

- Events: Events are an important element of interactive media as a trigger of one sequence. As an example, we can take the start button, which triggers the playback of music. In Scratch, we can trigger the action by pressing a green flag, a space bar, etc.
- Parallelism: Parallelism is the implementation of several different sequences simultaneously. There are two types of parallelism: parallelism between sprites and parallelism within a single object. The parallelism between sprites involves the implementation of several different sprites so that each sprite has its own sequence, triggered by the same event. If one sprite has a few different actions that are triggered by the same condition, a parallelism within a sprite is applied.
- Conditionals: They provide the ability to include branching in the process, based on some conditions.
- Operators: They allow programmers to perform mathematical operations (such as addition, subtraction, division, and multiplication), operations on strings (such as bundling) and execution of functions.
- Data: Data involves storing, retrieving and updating values. In Scratch we can use variables. A variable can be a number, a string or a list that may contain a set of numbers or strings.

2.2. Project based learning

Project-based learning (PBL) is a model that organizes learning around projects. According to the definitions found in PBL handbooks for teachers, projects are complex tasks, based on challenging questions or problems, that involve students in design, problem-solving, decision making, or investigative activities; give students the opportunity to work relatively autonomously over extended periods of time; and culminate in realistic products or presentations (Jones, Rasmussen, & Moffitt, 1997; Thomas, Mergendoller, & Michaelson, 1999). We think that future teachers need to experience project-based learning through the same activities, as they will work on with their students in classroom. Therefore, we compare and analyse projects made by students and pupils.

3. Empirical research and analysis

In the current section we describe the sample and online project assessment tool.

3.1. Sample

We analyse Maze and Escape Room games programmed by students, prospective teachers of computer science, and pupils aged 8 to 12 years at different courses and workshops. Maze games were created by 16 pupils at a workshop and

8 by students of pedagogical programme of computer science. Escape Room games were created by 8 pupils at a workshop and 11 students. Some pupils already had previous knowledge in Scratch. Students had experiences with introductory programing in Python.

3.2. Project assessment in Scratch

For quantitative analysis we used open-source web application called Dr. Scratch (Moreno-León, Robles and Román-González 2015).

Dr. Scratch allows learners to evaluate their projects and receive a CT score, so they can discover their own degree of development of this ability. In addition, the tool offers a feedback report with ideas and tips how to improve the code, aiming to encourage students' desire to keep on developing their programming skills. Dr. Scratch also supports teachers in the assessment tasks, offering a complete report that could be used to identify specific issues in an automatic way (Moreno-Leon, Robles & Roman-Gonzalez 2016). Dr. Scratch's big potential is in its ability to detect bad habits of programming or potential errors and common misconceptions, such as unimportant sprite names, repetition of code, code that is never executed and the incorrect initialisation of object attributes (Moreno-León and Robles 2015). Dr. Scratch has limitations in assessing fundamental CT skills as debugging and remixing. Also, Dr. Scratch could not evaluate functionality or creativity.

CT score is assigned based on next seven criteria: flow control, data representation, abstraction, user interactivity, synchronisation, parallelism and logic. Analysed project can get from 0 up to 21 points, that is 3 points for each of the seven criteria. The feedback given by Dr. Scratch depends on the score assigned to a project. With score increasing, the information we get is more sophisticated, intended for advanced users. Depending on the score, there are three possible stages: basic, developmental and master as the highest stage.

3.3. Analysis by Dr. Scratch Criteria

For evaluation of projects we chose Dr. Scratch. The projects are evaluated by the criteria presented below (Moreno-Leon, Robles, and Roman-González 2015).

Flow control is related to the behaviour of the controlled sprite. It involves certain blocks that are repeated for a number of times or until the situation arises.

Available points:

- 1 point: the project uses the most basic way to control the behaviour of a sprite.
- 2 points: the project uses repetition, which is achieved with a repeat of instruction blocks.
- 3 points: the project uses repetition, which depends on a given situation.

The concept of data flow can be related to the concept of sequences and loops described by Brennan and Resnick (2012). According to them, sequences are a key concept in programming, expressed as a series of individual steps of instructions that can be executed by a computer.

Data representation criterion refers to the set of information about the sprite to run properly.

Available points:

- 1 point: if in a project each sprite has a number of attributes, such as position, orientation, costume and visibility of a sprite.
- 2 points: the project uses variables that can store different types of data.
- 3 points: the project uses lists to store project information.

The CT concept of the so-called data defined by Brennan and Resnick (2012) can be related to the Dr. Scratch data representation criterion.

Abstraction criteria reflects an ability to break a problem into smaller parts that are easier to understand, program and debug.

Available points:

- 1 point: different scripts control sprite's behaviour, where each of these scripts deals with a particular issue.
- 2 points: the project contains new user-defined blocks that consist of a sequence of instructions.
- 3 points: the project uses clones.

User interactivity means that, the person who is running the project can perform actions that provoke new situations in the project.

Available points:

- 1 point: the project uses "when green flag clicked" to start interactivity.
- 2 points: the project uses blocks, which allow user to interact with sprite.
- 3 points: the project uses a webcam or a microphone to create interaction.

Synchronisation criterion refers to a concept, which allows our sprites to organise things to happen in the order we want.

Available points:

- 1 point: the project uses "wait" block to synchronise the behaviour of the sprite.
- 2 points: the project uses messaging or broadcasting.
- 3 points: the project uses synchronisation to make things happen in order we want.

Parallelism is in Dr. Scratch defined as a possibility that several things occur simultaneously, similarly to Brennan and Resnick (2012.

Available points:

- 1 point: the project has several scripts that start with "When green flag is clicked".
- 2 points: in the project several things occur when the user presses a certain button or clicks on an object.
- 3 points: the project uses more complex blocks, such as "when backdrop switches to", "when loudness", and alike.

Logic criterion, according to Dr. Scratch website explanation, refers to the project's ability to behave differently depending on the situation.

Available points:

- 1 point: the project uses simple conditionals, such as "if-then".
- 2 points: the project uses blocks, such as "if-then/else".
- 3 points: the project uses logical operators in term of combining the conditions.

Brennan and Resnick (2012) define this concept as operators.

Assessment of projects

In Table 1 and Table 2 we compare the results of projects evaluated with Dr. Scratch online tool.

Table 1: Maze projects evaluated with Dr. Scratch and the differences between students' and pupils' results.

		average	average difference	stdev	median	mode
Flow control	pupils	1.9	0.3	0.5	2	2
	students	2.1		0.4	2	2
Data representation	pupils	1.3	0.3	0.6	1	1
	students	1.6		0.5	2	2
Abstraction	pupils	1.0	**-0.2**	0.4	1	1
	students	0.8		0.5	1	1
User interactivity	pupils	1.9	0.1	0.5	2	2
	students	2.0		0.0	2	2
Synchronisation	pupils	1.0	**0.9**	1.3	0	0
	students	1.9		1.6	3	3
Parallelism	pupils	1.4	**0.6**	1.0	1	1
	students	2.0		1.4	3	3

Logic	pupils	0.9	**1.1**	0.3	1	1
	students	2.0		1.1	2	1
Score	pupils	9.4	**2.9**	3.6	9	8
	students	12.4		4.7	14.5	16

Table 2: Escape room projects evaluated with Dr. Scratch and the differences between students' and pupils' results.

		average	average difference	stdev	median	mode
Flow control	pupils	2	**0.6**	0.5	2	2
	students	2.6		0.7	3	3
Data representation	pupils	1.6	0.3	0.7	2	1
	students	1.9		0.5	2	2
Abstraction	pupils	1	0	0.0	1	1
	students	1		0.0	1	1
User interactivity	pupils	2	0.1	0.0	2	2
	students	2.1		0.3	2	2
Synchronisation	pupils	3	0	0.0	3	3
	students	3		0.0	3	3
Parallelism	pupils	2.9	0.1	0.4	3	3
	students	3		0.0	3	3
Logic	pupils	1.5	**1**	1.1	1	1
	students	2.5		0.8	3	3
Score	pupils	14	**2.1**	2.1	14	16
	students	16.1		1.9	17	17

4. Analysis

From comparison between Table 1, Table 2 and Figure 1 we deduct some basic differences and some expected similarities between pupils and students.

The results of the Maze activity, presented in Table 1, show that the biggest difference is in the category of logic, which we assume is due to the differences in reasoning among pupils and students. Instead of using "if-then/else" conditionals pupils used "if-then" conditionals. Big difference also occurs in the category of synchronisation (Figure 1). We attribute the differences to the greater complexity

(broadcasting and messaging) of students' games. We also detected notable difference in the category of parallelism as shown in Table 1, which can be explained by the expected complexity of students' understanding of simultaneous events. In the other categories the differences are not notable which was expected due to the nature of the task that was given to them. Creating a labyrinth requires the use of interactivity between user's sprite and other objects, e.g. an interaction between a sprite and the wall, or with any other objects. That's probably why all pupils and students have achieved the same amount of points, regarding this criterion. The concept of user interactivity can be related to Brennan and Resnick (2012) conditionals sentences. The small negative difference in the abstraction category is attributed to the fact that one pupil defined his own blocks in his game, which was awarded by Dr. Scratch.

In general, the differences between pupils and students are smaller within the Escape Room activity. The results of the Escape Room activity, presented in Table 2, indicate that the biggest difference is in the category of logic, where students used nested conditionals. The only other difference in Escape Room project is shown in the category of flow control, which means that students were using "repeat until", and not just "repeat" as pupils did. All pupils and students scored top marks in the categories parallelism and synchronisation, which can be attributed to the type of the activity and its requirements, for example changing the scenes between levels.

Before the analysis, we have assumed that the concept of abstraction would make a gap between students´ and pupils' programming skills but our expectations didn't come true. Nearly all the projects were evaluated with 1 point in the category of abstraction. Dr. Scratch assumes the use user defined blocks for 2 points and cloning to evaluate the abstraction with 3 points. As user defined blocks and cloning are an advanced concept and our objective was to prepare students for teaching primary school pupils, we encouraged the students to use mainly the concepts that can be developed by an average pupil. If we relate our findings to Brennan and Resnick (2012), we would realise that pupils do not develop abstraction as a concept but as a practice, which they characterise as building something large by putting together collections of smaller parts.

Conclusion

How to assess CT is open research problem. Denning (2017) argues that it should be assessed as a skill, rather than through knowledge frameworks (Sentence and Csizmadia, 2017). Skills and conceptual computer science knowledge intertwine with each other and they are reflected in CT during programming. We suppose that having a clear conceptual knowledge of a discipline can also support the development of related skills.

Our approach is to directly assess evidence of computer science concepts in programs. If programming is reflecting the way of thinking, then the idea is that CT can be assessed by the quality of the programs that a student produces.

In the paper we compared the results of CT assessment between primary school pupils, who attended two workshops in Scratch, and students, prospective teachers of computer science. For CT assessment we used Dr. Scratch tool (Moreno-Leon, Robles & Roman-Gonzalez 2016). The purpose of our analysis was to compare the way of thinking of both groups during development of typical projects in Scratch, such as Maze and Escape Room.

Escape Room projects are more advanced and complex then Mazes (Figure 1). The Escape Room problem enables students to develop conceptual knowledge about the ordering (things happen in the order we want), reflected in the criteria of Synchronisation, and concept of Parallelism for both groups: pupils and students.

Students have more developed CT in comparison with pupils. In general, there is 2 to 3 points difference in the total score between the pupils and the students, which was expected. For deeper understanding of CT development among the participants of the both groups pupils and students we need to use more instruments or at least more sophisticated qualitative analysis of the projects.

According to Brennan and Resnick (2012), the presence of described concepts in the projects' code is reflecting the development of CT.

Acknowledgements

This research is supported by project "Pedagogical and technological aspects of educational computer games", funded by the Bulgarian National Scientific Fund, Contract DN-05/10, 2016.

Bibliography

1. Brennan, K. and Resnick, M. (2012) "New frameworks for studying and assessing the development of computational thinking", AERA, Innovation and technology in computer science education, (pp. 161-165), Pariz, France.
2. Denning, P. (2017) Remaining trouble spots with computational thinking, Communications of the ACM. 60(6): 33-39.
3. Howland, K. and Good, J. (2015) "Learning to communicate computationally with Flip: A bi-modal programming language for game creation", Computers & Education, Vol 80, pp 224-240.
4. Jones, B. F., Rasmussen, C. M., & Moffitt, M. C. (1997). Real-life problem solving.: A collaborative approach to interdisciplinary learning. Washington, DC: American Psychological Association.
5. Kalelioğlu, F. (2015) "A new way of teaching programming skills to K-12 students: Code.org", Computers in Human Behavior, Vol 52, pp 200-210.
6. Lye, S. Y. and Koh, J. H. (2014) "Review on teaching and learning of computational thinking through programming: What is next for K-12?", Computers in Human Behavior, Vol 41, pp 51-61.

7. Moreno Leon, J. and Robles, G. (2014) "Analyze your Scratch projects with Dr. Scratch and assess your Computational Thinking skills", Scratch Conference, 2015, pp 12-15.
8. Moreno Leon, J., Robles, G. and González, R. (2015) "Dr. Scratch: Automatic Analysis of Scratch Projects to Assess and Foster Computational Thinking", RED-Revista de Educación a Distancia, No. 46.
9. Moreno-Leon, J., Robles, G, & Roman-González, M. (2016). "Comparing computational thinking development assessment scores with software complexity metrics", In Global Engineering Education Conference (EDUCON), 2016 IEEE, Abu Dhabi, pp. 1040 – 1045.
10. Pinto, A. and Escudeiro, P. (2014) "The Use of Scratch for the Development of 21st Century Learning Skills in ICT", 9th Iberian Conference on Information Systems and Technologies (CISTI), IEEE.
11. Roque R., Rusk N., Resnick M. (2016) Supporting Diverse and Creative Collaboration in the Scratch Online Community. In: Cress U., Moskaliuk J., Jeong H. (eds) Mass Collaboration and Education. Computer-Supported Collaborative Learning Series, vol 16. Springer, Cham
12. Sentance, S. and Csizmadia, A. (2017) "Computing in the curriculum: Challenges and strategies from a teacher's perspective", Educ Inf Technol, Vol 22, pp 369–495.
13. Su, A. Y., Huang, C. S., Yang, S. J., Ding, T. J. and Hsieh, Z. Y. (2015) "Effects of Annotations and Homework on Learning Achievement: An Empirical Study of Scratch Programming Pedagogy", Journal of Educational Technology & Society, Vol 18, No. 4, pp 331– 343.
14. Thomas, J. W., Mergendoller, J. R., and Michaelson, A. (1999). Project-based learning: A handbook for middle and high school teachers. Novato, CA: The Buck Institute for Education.
15. Touretzky, D. S., Gardner-McCune, C. and Aggarwal, A. (2017) "Semantic Reasoning in Young Programmers", Proceedings of the ACM SIGCSE Technical Symposium on Computer Science Education, pp 585-590, Seattle, Washington, USA.
16. Wing, J. M. (2006) "Computational thinking", Communications of the ACM, pp 33-35.
17. Wing, J. M. (2011) "Research Notebook: Computational Thinking: What and Why?", The magazine of Carnegie Mellon University's School of Computer Science, Pittsburg, USA.