Discovering children's competences in coding through the analysis of Scratch projects

Paper published In *Global Engineering Education Conference (EDUCON), 2018 IEEE* (pp. 1127-1133)

Sofia Papavlasopoulou
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
spapav@ntnu.no

Michail N. Giannakos
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
michailg@ntnu.no

Letizia Jaccheri
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
letizia.jaccheri@ntnu.no

***Abstract*— Computational thinking and coding has received considerable attention over the past several years. Considerable efforts worldwide suggest the need for more empirical studies providing evidence-based practices to introduce and engage children with coding activities. The main goal of this study is to examine which programming concepts students use when they want to develop a game, and what is the interrelation among these concepts. To achieve our goal, a field study was designed and data were collected from coding activities. In detail, during a two-week period, one-day workshops were organized almost every day on which 44 children participating in, with ages between 8-17. The workshops follow a constructionist approach and comprise of two parts. First the children interact with robots, and then develop a game using Scratch. The findings provide a deeper understanding on how children code by showing the use of specific programming concepts to develop their projects and their correlations. Hence, we improve our knowledge about children's competences in coding.**

Keywords—Computer Science Education; maker movement; coding; computantional thinking; constructionism

I. INTRODUCTION

Currently, more and more worldwide efforts point out the importance of all children, from early age, to acquire digital competences and computational thinking skills. The rise in importance of computational thinking skills with respect to STEM (science, technology, engineering, and mathematics) fields has been recognized both by those within the STEM education communities and CS education organizations [1]. It is indeed critical to increase children's engagement with computational thinking, as this may lead to increased enrolments into computer science education studies, as a shortage of computer science professionals exists [2] [3]. During the last decade, organizations like Computer Science Teachers Association (CSTA), National Math and Science Initiative, ACM and code.org share the goal of fostering computer science education and the need of promoting educational settings around problem solving, computational thinking and coding [4]. In addition, countries like United Kingdom, Finland, and Israel have integrated coding into their school curriculum.

Due to a wide range of digital tools, such as Scratch, Alice, Kodu, App inventor among others, children can experience an enjoyable and engaging experience in coding. These kinds of tools are built on block-based programming and have been used in different contexts both in formal and informal education [5]. Various studies combine coding and making activities, grounded on constructionist learning, to introduce coding [6]. Also, one of the most common activities when using a block-based programming environment like Scratch, is the creation of a game [7]. Children of different ages (8-18 years old) understand concepts like loops and variables [8] at a clubhouse activity. There is a growing interest and a variety of ways to assess coding knowledge and learning gains [9]. However, it is still challenging to assess computational thinking [10].

We designed a one-day coding workshop and conducted multiple sessions of it, with various groups of children from the age of 8 to 17 years old. In this study, the aim is to investigate the extent to which children used specific programming concepts in their final projects. This allows us to gain insights into children's competences in coding, after our very short timescale coding workshop. In addition, our purpose through the analysis of the projects, is to use the results of this study as future reference in the collection of children's artifacts in a systematic way, not only at the end but also during the workshop. Also, it will help us define the prominent measures of programming concepts children can use in our coding workshop. We address the following two research questions:

- Which programming concepts are mostly used in children's games?
- How are the programming concepts in children's games related?

The rest of the paper is structured as follows: first, we present the related work and then we describe our methodology which includes the description of the coding workshop, data collection and analysis. In the fourth section, we present the findings. Finally, we discuss the results providing limitations and future work.

II. Related work

Papert's [11] constructionist approach demonstrates that children can learn by building their own projects and actively get involved in discovering their knowledge. Educational activities based on constructionist learning can be described as learning by doing, project-based learning, inquiry-based learning among others. Coding and especially game creation using technology tools is considered as another type of constructionist learning which enhances computational thinking, critical thinking and under specific contexts, collaboration and innovation.

Various studies use visual programming languages in coding activities to engage children and increase their ability to code [6] [12] [13]. Moreover, other studies have shown the value of combining physical fabrication to engage students with complex programming concepts (e.g., loops, conditionals, events) and practices (e.g., remixing, testing, debugging).

After nine years of coding summer camps, Adams and Webster [14], analyzed Scratch programs created from the students, in order to examine the relation between the choice of specific blocks with the project-types. Denner et al. 2012 [15], analyzed 108 games created from girls after a design and coding activity, showing evidence that the understanding of computer science concepts can be supported. In another study, in a one year period of using Scratch, students' projects show an increased number of programming concepts (loops, variable, Boolean logic and random numbers) [8]. Using Scratch but in the context of a classroom writing workshop, Bruke and Kafai 2012 analyzed the projects in regard to their programming blocks and the frequency of programming concepts used. Their results show that middle school children learned fundamental concepts. Also, in e-textiles projects with Lilypad Arduino, Kafai 2013 [16] examined the final artifacts and discovered the use of a variety of concepts and practices.

Many studies, which examined Scratch artifacts, used the framework developed by Brennan and Resnick 2012 to analyze the complexity of the programming concepts [17] [18] [13] [19]. In general, as someone designs interactive media with Scratch, computational concepts (mapping to Scratch blocks) refer to elements like sequences, loops, parallelism, events, conditionals, operators, and variables structures that also exist in many programming languages.

III. Methodology

A. Description of the Coding Activity

For the purpose of this study, coding workshops have been organized at the Norwegian University of Science and Technology, in Trondheim, Norway. The workshops have been designed following the constructionist approach and its main principle, learning-by-doing as done by previous efforts. Coding workshops are out-school activities, in which students from K12 education participate. Our participants are children from 8 to 17 years, and during the workshop they interact with digital robots, using Scratch for Arduino (S4A), and then, they code their own game using Scratch programming language. This activity lasts about four hours and the children are organized in pairs or in teams of three. Also, five assistants participate in the study. The assistants have previous experience with similar workshops and their main duty is to instruct the children and facilitate the process. The same process is followed on every workshop. The workshop comprises of two parts, that is; Interaction with the robots and Creating games with Scratch.

Interaction with the robots: During that part, children interact with digital robots, which are built by an artist using recycle materials mainly from computer parts. Interaction with digital robots provides a smooth start to the workshop and familiarizes children with coding. By showing a connection to the physical world helps children to handle STEM subjects and manage difficult problems [20].

First, the children are welcomed into the room by the assistants, and are divided in teams, sitting next to one robot. The assistants provide a brief explanation of the workshop's activities and point the children to a worksheet placed on their desks. The goal is for the children to become familiar with the robots by answering simple questions about the exact place and number of the sensors and lights on the robots. Then, the children should follow a paper tutorial with detailed instructions, on how to make the robots respond to the physical environment. This is done by visual effects using simple loops of Scratch for Arduino (e.g. to make the tail of the turtle robot move when there is more light at a sensor). The teams work collaboratively and independently to complete this task. The first part lasts, approximately, from forty-five minutes to one and a half hour. When everyone has completed their task, the teams take a short break. Then, the second part begins. During this part of the workshop, the children receive an introduction to coding activities including tangible objects. The interaction with digital robots offers a better understanding of STEM subjects by showing the connection with the real world, as it can help children to handle difficult problems [20]. Children are introduced to coding by playfully interacting with the robots while getting motivation and inspiration (Fig. 1).

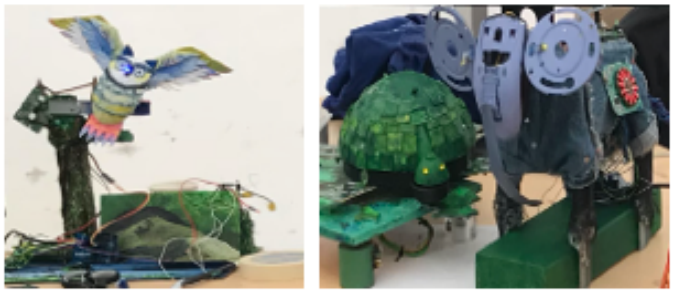

Fig. 1. Some of the robots used created from an artist

Creating games with Scratch: This section is the main activity of the workshop and its duration is about three hours, without robots being present. The goal during the second part is for the children to develop a simple game, coding in Scratch. First, the children have to think and decide the story of their game, and then create a draft storyboard. Then, they start creating their game by coding using Scratch. To achieve this goal, another paper tutorial is given to the children, which contains examples about all the basic CS concepts and possible loops they may need to use to develop their own game. The activity is based on children's self-exploration with the constant help of the assistants when asked. The assistants facilitate this process and give advice to the children on how to manage the game development while collaborating with each other. That was one of the challenges the assistants had to face; their try to

explain the programming concepts as being asked from the children and also find the simplest way for them to understand and code. Help provided, varies depending on each team's needs. For example, the first level of help is to give confirmation; then, going levels up, assistants could give the name of the concept needed to complete a task or show in detail the concept and its execution. Sometimes, more complex CS concepts are introduced only if needed. Finally, after the games have been developed, children reflect and play each other's games (Fig. 2).

Fig. 2. Children working collaboratively to complete the activities of the coding workshop

B. *Procedure and Data Collection*

The study of our coding workshop lasted for two weeks during Autumn 2016 with forty-four children in total. The participants ranged in age from 8 to 17 years old and were recruited from the local coding clubs and schools. We conducted workshop sessions almost every day with different groups of participants who had approximately the same age. The structure of the activity was the same for each session, with slightly different adjustments specially in terms of time, when the participants were at the young age of eight to twelve years old.

Our upper goal is to examine how children learn to code. We employed a variety of instruments to collect data from the learning process of the coding workshop. In this paper, we focus only on the children's final projects which were collected at the end of the activity. Each team of children, after the four hours of the workshop, had to deliver a functional game and demonstrate competence in coding. Fig. 3 presents two different scenes of games created by the teams.

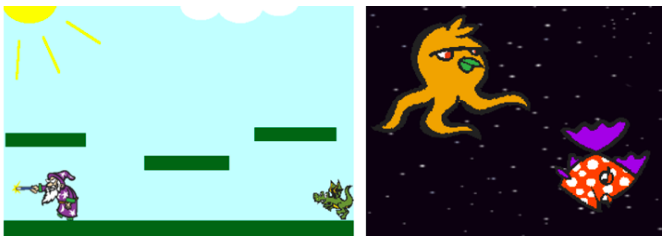

Fig. 3. Example of two scenes from the games created.

C. *Data Analysis*

The final projects collected, are the games created from the participants, using Scratch. Finally, we collected nine games in total from the participants, which links approximately to twenty-seven participants.

The analysis of the projects started with checking that the code was present and operational in all the games. For each game, a researcher manually searched on the code of the projects seeking for the blocks that indicate the concepts used in the game. After an initial testing, the games were analyzed by the researcher, based on the programming concepts inspired from Brennan and Resnick [21]. We identified six basic concepts mapping to Scratch programming blocks which are also common to other programming languages [22] [21]. These concepts are useful in a wide range of Scratch projects and agree with our previous experience with the workshop's games. Thus, to complete our artifact analysis, we based on the following measures: event handling, iteration/looping, conditional statements, threads (parallel execution), synchronization, operators, variables (see Table 1). All the previous measures fit together conceptually.

In detail, event handling refers to a key concept of coding and describes the series of individual instructions to produce actions. To create a program using Scratch, you need to think systematically about the order of the steps responding to events triggered by the user or another part of the program. Iteration/looping is the repeating action of a series of instruction Loop is a mechanism of running the same sequence multiple times. Conditional statements indicate the check for a condition (e.g. if, if-else). Threads (parallel execution) show the launching of two stacks at the same time that creates two independent threads executing in parallel. Synchronization happens when the multiple the actions of multiple sprites can coordinate. Operators provide support for mathematical, logical, and string expressions, enabling the programmer to perform numeric and string manipulations. Variables, involve storing, retrieving, and updating values. In general, Scratch currently offers two containers for data: variables (which can maintain a single number or string and is what we looked for in children's projects) and lists (which can maintain a collection of numbers or strings).

TABLE I. PROGRAMMING CONCEPTS AND THEIR EXPLANATIONS

Programming concept	**Explanation**
Event handling	Responding to events triggered showing when actions should occur
Iteration (looping)	Repeating series of instructions
Conditional statements	Checking for a condition
Threads (parallel execution)	Two independent threads that execute in parallel
Synchronization	Coordinate actions of multiple sprites
Operators	Support for mathematical, logical and string expressions enabling numeric and string manipulations
Variables	Storing, retrieving and updating values

For example, Fig. 4 shows a script form a game in which a character moves using the keyboard trying not to touch specific parts of the designed Scratch scene for the game.

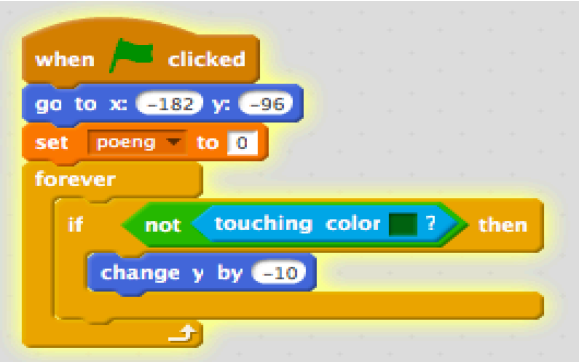

Fig. 4. Scratch script from a game.

This script uses event handling, conditional statements and iteration (looping). In each game, we counted the frequency of the blocks used regarding the concepts we have chosen as measures. Then, we performed descriptive statistics and used Pearson's correlation coefficient which determines the relationship between the variables.

IV. Findings

All the projects we analyzed were operational and completed successfully. Table 2 shows the results of the statistical analysis using IBM SPSS statistics for the minimum, maximum, mean and standard deviation of the measures we used. As we can see from the results, the mostly used concepts on children's games is the sequence/event handling and conditionals, followed by threads and operators. Synchronization was present only two times in the sum of games and only two games were using it once each.

TABLE II. Statistics for Children's Games

Programming Concept	Min	Max	Mean	Sum	Std. Deviation
Event handling	3	14	7.33	66	3.808
Iteration (looping)	0	8	3.33	30	2.739
Conditional statements	1	18	6.44	58	6.729
Threads (parallel execution)	0	15	5.78	52	4.790
Synchronization	0	1	0.22	2	0.441
Operators	0	12	6.56	49	6.069
Variables	0	14	3.33	30	4.796

The results from the Pearson's correlation (see Table 3) showed significant correlation at the 0.01 level between threads (parallel execution) and iteration (looping), variables and event handling, variables and iteration (looping) and finally variables and threads. All the significant correlations scored more than 0.7 showing a strong relation among the variables.

V. Discussion

The findings show that variables and event handling are strongly correlated. Event handling is the basic programming concept and is present in all projects. It can be considered as a prerequisite to have operational projects. On the other hand, variables are a complicated concept. Variables are present in four of the nine games in total and we saw that it is the only concept correlated with three out of the six other concepts. We notice that these games have also many blocks that indicate sequences. That can be explained from the fact that in order to use variables in the "correct" place of the code, needs a good understanding of the systematic order of steps and action you create. As such, only the games with more complex interactions with stories, characters and backgrounds had more complicated sequences and use of variables. In their study with youth 8-18 years old, Maloney et al. [8] show that all analyzed projects had sequential execution and most of them had threads; while the rest of programming concepts like loops conditional, synchronization, are not needed in every project, that was reflected also in our analysis. An interesting result is that event handling is not correlated with other concepts that variables. This is possibly related to the fact that the need of using event handling and sequences in the projects does not give any prominent correlation with the other concepts. Similarly, strongly correlated are variables and iteration as well as variables and threads. When children use the "forever" and "repeat" loops (iteration) add complexity to their games, therefore using variables inside the forever loops shows knowledge of how these loops can be used. Threads (parallel execution) is a concept that can be found from simple "when flag is pressed" to more complicated actions. In the games who had variables we find the use of multiple sprites, when something is repeatedly used. This action indicates the need for more variables, if possible. Threads and iteration are the higher correlated programming concepts. Indeed, when you initiate two stacks at the same time that execute in parallel requires the use of more scenes and characters. That is exactly the case in the games we analyzed. A surprising result is the fact that threads are not correlated with synchronization. In Scratch, synchronization helps to control timing between the sprites which run as parallel threads, therefore the use of broadcast/receive blocks (synchronization) are essential to create communication between them. Although, in the analyzed projects synchronization was very rarely used.

Finally, the descriptive statistical analysis of the projects and the Pearson correlation results provide useful insight about the prominent measures of the programming concepts. In detail, the programming concepts with the highest frequency (i.e., used often by the children), are not correlated with each other. On the other hand, the concepts that are correlated have low frequency. This finding suggests that further work is needed in the area, in order to define the prominent measures of the programming concepts. Identifying such concepts will help teachers in improving their learning materials and in developing better tools which are more appropriate for use by children.

VI. Limitations and Future Work

As with all empirical studies, our study has some limitations. First, this study was conducted in a specific context under certain circumstances, thus generalization should be done with caution. More specifically, the circumstances refer to that fact that the games were collected from a wide range of children's age and also the limited number of games analyzed at the end.

TABLE III. CORRELATIONS AMONG PROGRAMMING CONCEPTS

	Event handling	Iteration (looping)	Conditional statements	Threads	Synchronization	Operators	Variables
Event handling	1						
Iteration (looping)	0.647	1					
Conditional statements	0.374	0.608	1				
Threads (parallel execution)	0.573	0.921**	0.356	1			
Synchronization	0.174	0.242	0.257	0.204	1		
Operators	0.556	0.674*	0.564	0.568	0.316	1	
Variables	0.821**	0.876**	0.514	0.798**	-0.39	0.709*	1
** $p < 0.01$, * $p < 0.05$							

Furthermore, the workshop was performed as a structured activity, thus it would be interesting to conduct similar workshops as regular activities at school. Finally, the present study is missing some qualitative insights which would offer significant details on how the children decided to use their coding competences. Future studies should take also a qualitative approach in order to complement and extend our findings.

It should be pointed out that this project is still at a preliminary stage, as are the findings that have been presented. Nevertheless, the results strongly indicate the need to further investigate children's learning trajectories. Monitoring how students use and combine constructs, patterns, and mechanics in their games we can gain insights on children's computational thinking and learning. That will result in better designed coding activities for children.

Regarding the next steps in our project, we plan to perform mixed methods analysis. By combining qualitative and quantitative analyses we will be able to improve children's experience with such workshops. Finally, to better understand the children and the way the code, complexity theory should be combined with fuzzy-set qualitative comparative analysis (fsQCA) as it offers deeper insight into the data and can identify asymmetric relations within a sample [2, 23].

REFERENCES

[1] A.J.T. Force, Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science, in, Technical report, Association for Computing Machinery (ACM) IEEE Computer Society, 2013.

[2] I.O. Pappas, M.N. Giannakos, L. Jaccheri, D.G. Sampson, Assessing Student Behavior in Computer Science Education with an fsQCA Approach: The Role of Gains and Barriers, *ACM Transactions on Computing Education (TOCE)*, **17** (2017) Article No. 10.

[3] S. Zweben, Computing Degrees and Enrollment Trends from the 2012-2013 in, Computing Research Association Taulbee Survey, Washington, DC, 2014.

[4] S. Grover, R. Pea, Computational Thinking in K–12 A Review of the State of the Field, *Educational Researcher*, **42** (2013) 38-43.

[5] S. Papavlasopoulou, M.N. Giannakos, L. Jaccheri, Empirical studies on the Maker Movement, a promising approach to learning: A literature review, *Entertainment Computing*, **18** (2017) 57-78.

[6] J.-M. Sáez-López, M. Román-González, E. Vázquez-Cano, Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools, *Computers & Education*, **97** (2016) 129-141.

[7] O. Meerbaum-Salant, M. Armoni, M. Ben-Ari, Learning computer science concepts with scratch, *Computer Science Education*, **23** (2013) 239-264.

[8] J.H. Maloney, K. Peppler, Y. Kafai, M. Resnick, N. Rusk, Programming by choice: urban youth learning programming with scratch, ACM, Proceedings of the 39th SIGCSE technical symposium on Computer science education 2008.

[9] S. Papavlasopoulou, K. Sharma, M. Giannakos, L. Jaccheri, Using Eye-Tracking to Unveil Differences Between Kids and Teens in Coding Activities, in: Proceedings of the 2017 Conference on Interaction Design and Children, ACM, 2017, pp. 171-181.

[10] S. Grover, Assessing Algorithmic and Computational Thinking in K-12: Lessons from a Middle School Classroom, in: Emerging Research, Practice, and Policy on Computational Thinking, Springer, 2017, pp. 269-288.

[11] S. Papert, Mindstorms: Children, computers, and powerful ideas, Basic Books, Inc., 1980.

[12] E.-S. Katterfeldt, N. Dittert, H. Schelhowe, Designing digital fabrication learning environments for Bildung: Implications from ten years of physical computing workshops, *International Journal of Child-Computer Interaction*, **5** (2015) 3-10.

[13] Y.B. Kafai, E. Lee, K. Searle, D. Fields, E. Kaplan, D. Lui, A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles, *ACM Transactions on Computing Education (TOCE)*, **14** (2014) 1.

[14] L. Johnson, S. Adams Becker, V. Estrada, A. Freeman, The NMC Horizon Report: 2015 Museum Edition, ERIC, 2015.

[15] J. Denner, L. Werner, E. Ortiz, Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?, *Computers & Education*, **58** (2012) 240-249.

[16] Y.B. Kafai, K. Searle, E. Kaplan, D. Fields, E. Lee, D. Lui, Cupcake cushions, scooby doo shirts, and soft boomboxes: e-textiles in high school to promote computational concepts, practices, and perceptions, in: Proceeding of the 44th ACM technical symposium on Computer science education, ACM, 2013, pp. 311-316.

[17] Y.B. Kafai, K.A. Peppler, R.N. Chapman, The Computer Clubhouse: Constructionism and Creativity in Youth Communities. Technology, Education--Connections, ERIC, 2009.

[18] Y.B. Kafai, V. Vasudevan, Constructionist gaming beyond the screen: Middle school students' crafting and computing of touchpads, board games, and controllers, in: Proceedings of the Workshop in Primary and Secondary Computing Education, ACM, 2015, pp. 49-54.

[19] E. Lee, Y.B. Kafai, V. Vasudevan, R.L. Davis, Playing in the arcade: Designing tangible interfaces with MaKey MaKey for Scratch games, in: Playful User Interfaces, Springer, 2014, pp. 277-292.

[20] S. Bakker, E. van den Hoven, A.N. Antle, MoSo tangibles: evaluating embodied learning, in: Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction, ACM, 2011, pp. 85-92.

[21] M.R. Karen Brennan, New frameworks for studying society computer science exemplification project, *Paper presented at AERA*, (2012).

[22] C. Kelleher, R. Pausch, Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers, *ACM Computing Surveys (CSUR)*, **37** (2005) 83-137.

[23] I.O. Pappas, S. Papavlasopoulou, M.N. Giannakos, D.G. Sampson, An Exploratory Study on the Influence of Cognitive and Affective Characteristics in Programming-Based Making Activities, in: 17th International Conference on Advanced Learning Technologies (ICALT), IEEE, Timisoara, Romania, 2017.