



Computational thinking and mathematics using Scratch: an experiment with sixth-grade students

José Antonio Rodríguez-Martínez, José Antonio González-Calero & José Manuel Sáez-López

To cite this article: José Antonio Rodríguez-Martínez, José Antonio González-Calero & José Manuel Sáez-López (2020) Computational thinking and mathematics using Scratch: an experiment with sixth-grade students, *Interactive Learning Environments*, 28:3, 316-327, DOI: [10.1080/10494820.2019.1612448](https://doi.org/10.1080/10494820.2019.1612448)

To link to this article: <https://doi.org/10.1080/10494820.2019.1612448>



Published online: 02 May 2019.



Submit your article to this journal [↗](#)



Article views: 4038



View related articles [↗](#)






View Crossmark data [↗](#)



Citing articles: 50 View citing articles [↗](#)



Computational thinking and mathematics using Scratch: an experiment with sixth-grade students

José Antonio Rodríguez-Martínez ^a, José Antonio González-Calero ^a and José Manuel Sáez-López ^b

^aDepartment of Mathematics, School of Education of Albacete (Edificio Simón Abril), University of Castilla-La Mancha, Albacete, Spain; ^bDepartament de Didactics, Faculty of Education, Spanish National University of Distance Education (UNED), Madrid, Spain

ABSTRACT

The potential benefits from the introduction of programming environments such as *Scratch* for learning mathematics has reactivated research in this area. Nonetheless, there are few studies which attempt to analyse their influence at the stage of Primary Education. We present the results of a quasi-experimental piece of research with sixth-grade students which studies the influence of *Scratch* both on the acquisition of mathematical concepts, and on the development of computational thinking. The experiment consisted of two different phases, a programming phase linked to the instruction in *Scratch* and focused on the acquisition of basic concepts of computational thinking (sequences, iterations, conditionals, and events-handling), and a mathematical phase completely oriented towards the resolution of mathematical tasks. In particular, the mathematical phase focused on word problems whose resolution involves the use of the least common multiple and the greatest common divisor. In order to evaluate the aims of the study, results from tests before and after instruction, both in computational thinking and in the mathematical standards, were compared. The results seem to indicate that *Scratch* can be used to develop both students' mathematical ideas and computational thinking.

ARTICLE HISTORY

Received 15 December 2017
Accepted 19 December 2018

KEYWORDS

Scratch; computational thinking; problem solving; mathematics; elementary school; programming

Nowadays, society demands citizens with extensive technological knowledge in areas that evolve at a great rate. Consequently, there is no doubt that technological knowledge and skills will become a fundamental element in the future promotion and development of current students' careers. Additionally, technology has also produced new ways of relating, communicating, learning and working (Cabero & Llorente, 2010). The introduction of programming languages is essential for the complete acquisition of digital competence and the appropriate use of information and communication technology (hereinafter, ICT) by our children (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008).

Although programming languages are currently more accessible to elementary students than the earliest languages were, other issues need to be overcome to fully exploit their pedagogical capabilities. For instance, a common error consists of designing programming activities highly disconnected from the students' real-life context, hence, they are not found motivating (Resnick et al., 2009). These disadvantages, in addition to technical limitations over past decades, have eclipsed the didactic potential that the field of programming may promote during early educational stages (Garneli & Choriantopoulos, 2018). In this context, the present work aims to study the effect of programming

activities on both the development of sixth-grade students' computational thinking and mathematical learning.

Antecedents

On computational thinking

Wing (2008) predicted that by the middle of this century computational thinking (hereinafter, CT) would be critical in students' upcoming competences. Although CT is a concept currently in vogue, the idea of developing CT in K-12 education was first introduced by Papert (1980). Basically, CT implies the recognition of the computational aspects of the world, and the use of tools and techniques of Computer Science, for the understanding of both natural and artificial processes and systems (Furber, 2012). More specifically, CT is defined as the ability to solve problems, to design systems and to understand human behaviour based on different computer concepts and processes (Wing, 2008). DiSessa (2000) underlined one of the most important perspectives on CT when arguing that it can be seen as a new form of literacy with the potential of be pervasive across different subjects, contexts, and domains (Weintrop et al., 2016).

In order to constitute a useful operational framework, there is a need to delimit the constituent components of CT. Although, as Wing (2008) suggests, CT is more complex than programming, programming requires the use of CT and is often employed to reach it (Lye & Koh, 2014). Thus, Román-González, Pérez-González, and Jiménez-Fernández (2017) considered the fundamental concepts of computing, and the logic-syntax of programming, with the aim of assessing students' acquisition of computational thinking. As a result, different computational concepts were taken into account: basic sequences, loops, iteration, conditionals, functions and variables.

The emergence and lack of maturity of the field of CT may lead to an even more delicate ambiguity when looking at areas of education. Thus, teachers are usually unfamiliar with CT and, as a result, are not able to find connections between educational curricula and CT (Shute, Sun, & Asbell-Clarke, 2017). This fact leads to little consensus on how CT should be integrated into educational processes. As a consequence, different visions are translated into a set of unplanned actions and policies that make the development of CT extremely difficult (Lye & Koh, 2014). Besides, the absence of a standard definition of the use of CT leads to research studies where measurements vary greatly across each piece of research, which makes the results less appropriate and certainly extremely difficult to compare (Shute et al., 2017).

Regardless of the difficulties of properly integrating CT into teaching sequences, different studies have produced evidence of the beneficial results that the learning of computational concepts may entail (e.g. Garneli & Chorianopoulos, 2018; Lye & Koh, 2014). So, the acquisition of CT is associated with an improvement in the ability to reason and solve everyday problems related to practically all areas of learning (Chen et al., 2017). In addition, Wing (2014) highlights the educational benefits of being able to think computationally due to the use of abstractions and reasoning skills, which enhance and reinforce intellectual abilities and, therefore, are transferable to different domains. Each element within CT, such as loops, conditionals or algorithmic development may be related to the student's effectiveness in solving any typology of problem (Chao, 2016). In a study with nine-year-old students, Manila et al. (2014) found an association between the completion of activities aimed at developing students' CT and a relevant improvement in the participants' ability to divide a problem into easier steps, or sub-problems, or the ability to create algorithms with an increasing level of abstraction. However, these authors stressed the need to include programming activities in the classroom on a daily basis for better assimilation. Different authors (e.g. Garneli & Chorianopoulos, 2018; Lye & Koh, 2014) think that it is necessary to develop new studies aimed at analysing how recent methodologies can promote CT in different traditional areas within student classrooms. In this regard, mathematics instruction seems a promising field since understanding of mathematic concepts are particularly prone to be enhanced using CT (Hickmott, Prieto-Rodriguez, & Holmes,

2018). However, these authors claim that currently there is a lack of empirical studies designed to include specific practices for K-12 that explicitly connect CT and the learning of mathematics.

Programming languages for teaching in K-12

The first programming language used with educational aims at elementary levels was LOGO (Feurzeig, Papert, & Lawler, 2011). Papert (1990) believed that the use of LOGO would help students to develop their understanding since he considered this programming language as “an instrument designed to help change the way you talk about and think about mathematics and writing and the relationship between them”. However, although it is important to underline the existence of a large amount of research (Bar-On, 1986; Howe, Ross, Johnson, Plane, & Inglis, 1982; Layman & Hall, 1988; Pea, Kurland, & Hawkins, 1985) that indicates eventual pedagogical advantages derived from the use of LOGO, most of the systematic studies was conducted exclusively with high school students. In turn, Buitrago Flórez et al. (2017) point out that skills related to CT must be taught not just at secondary and upper education levels, but from the elementary stages in order to initiate the cognitive development of students at an earlier age.

After LOGO's appearance, a large number of programming languages, many modelled after seminal LOGO, started to emerge with the aim of introducing coding activities into teaching settings in K-12 education. Regarding mathematical instruction in elementary levels, different environments have been employed to study the eventual impact of programming activities on mathematical learning. Thus, for example, a Logo-like programming environment has been successfully used with kindergarten students to work problem solving strategies and different mathematical skills related to numbers and geometry (Fessakis, Gouli, & Mavroudi, 2013), or an animated programming world, *ToonTalk*, has proved to be effective to work deep mathematical ideas with young students (Kahn, Sendova, Sacristán, & Noss, 2011). Among these environments, *Scratch* stands out as one of the most popular and studied programming languages. It was created to help young people learn to think creatively, reason systematically and work collaboratively (Brennan, Balch, & Chung, 2014). *Scratch* provides a visual programming tool that lets pupils create programmes by manipulating sequence elements graphically, rather than by coding them textually. This kind of language helps younger students work more easily and focus on activities. Related to this, Funke, Geldreich, and Hubwieser (2017) carried out a study with fourth-grade students and observed that all the participants were able to learn basic programming using *Scratch* in just three days.

Several studies point to the benefit in the development of mathematical thinking as a result of working with *Scratch* (Calao, Moreno-León, Correa, & Robles, 2015; Calder, 2010; Marmolejo & Campos, 2013). According to de Moraes, Basso, and Fagundes (2017), CT is a field which is especially appropriate to foster students' mathematical learning due to the overlapping between both domains. In addition, the *Scratch* coding environment may encourage young students to seek new representations of mathematical ideas and relationships (Hughes, Gadanidis, & Yiu, 2017). Besides, mathematical activities with *Scratch* allow teachers to adapt their teaching style to the students' individual characteristics and, at the same time, to deal not only with mathematical knowledge but also with CT (Benton, Hoyles, Kalas, & Noss, 2017). In the same way, Calder (2010) describes how students improved mathematical competence during the construction of applications in *Scratch*. In particular, the use of *Scratch* promoted the use of critical, meta-cognitive and reflexive skills, which are closely related to mathematics.

According to the above, it should be pointed out that although some studies offer evidence in support of the use of programming activities when teaching mathematics, there is a scarcity of works that, up to now, have addressed how these activities affect learning in specific areas of mathematics. Regarding this, the present work focuses on mathematical concepts whose, to the best of our knowledge, learning has not previously been addressed by means of programming activities in a visual language. In particular, concerning the field of mathematics, this work focuses on the divisibility concepts of the greatest common divisor (GCD) and the least common multiple (LCM), and

the use of these concepts in solving word problems. Problem solving in situations that involve the use of GCD and LCM is one of the mathematical topics at elementary levels where students find the most difficulties (Bintaş & Çamlı, 2009). Indeed, several studies underline the relevance of these difficulties, and point to the fact that even pre-service elementary teachers do not have a good understanding of these concepts (Brown, Thomas, & Toliaş, 2002; Gutiérrez-Gutiérrez, Gómez Guzmán, & Rico Romero, 2015; Noblet, 2013).

Objectives

The main aim of this study is to evaluate the potential of programming activities with *Scratch* to foster sixth-grade students' acquisition of mathematical ideas and CT. Specifically, to address this objective we try to answer the following research questions:

RQ1. Does explicit teaching of computational concepts in a *Scratch* environment during mathematical instruction have a significant effect on sixth-grade students' computational thinking?

RQ2. Does the use of a programming language during mathematical instruction have a significant effect on the mastery of sixth grade students in the resolution of problems that involve the use of GCD and LCM?

Method

Participants

We performed a between-groups teaching intervention with 47 students from two Primary Education sixth grade classes from an urban public school in the region of Castilla-La Mancha (Spain). Each class constituted a different condition in the study (either experimental or control). The experimental group and the control group were composed of 24 students (10 boys and 14 girls) and 23 students (9 boys and 14 girls), respectively. There was no evidence that the students had previous training in programming activities.

Research design

The study is based on a quasi-experimental design that consisted of two phases for both the experimental and the control group (EG and CG, respectively) (Figure 1). The first stage, called the *programming phase*, was a teaching sequence aimed at providing students with the basic knowledge and skills needed to create programmes using *Scratch*. This phase was composed of five sessions (60 min each) and was completed by both the EG and the CG. It was designed to explicitly work different computational concepts, namely: (i) sequences (the sequence structure of instructions that should be followed to complete a goal); (ii) iteration or loop (the control structure that makes it possible to repeat one or more sequences multiple times), (iii) event handling (instructions that

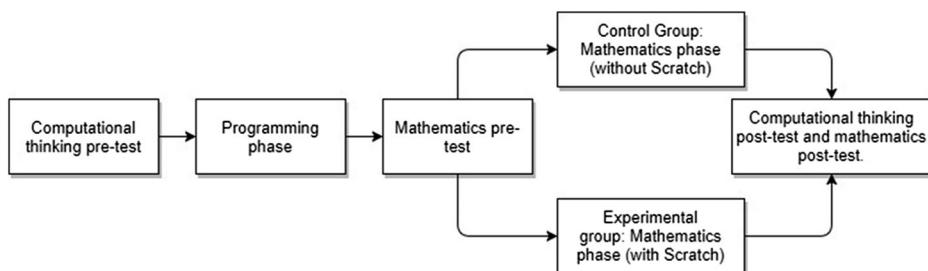


Figure 1. Phases of the experiment.

makes it possible to interact with objects in the programme environment); and, (iv) conditionals (instructions that either perform an action or not, according to a condition). Although programming consists of more computational concepts, some of them, such as functions and variables, were not explicitly considered in this study because these elements are more appropriate for secondary students. Indeed, the choice of the computational concepts addressed in this study, which can be seen as basic elements of CT, is based on the work of Román-González et al. (2017). These authors underline that these concepts are aligned with some of the CT framework (Brennan & Resnick, 2012) and with standards for students in grades 6–9 from the CSTA Computer Science Standards (Seehorn et al., 2011).

Additionally, a second phase, called the *mathematics phase*, focused on the teaching of mathematical learning standards using *Scratch* as a pedagogical tool. In particular, students completed three sessions of 60 min aimed at working a mathematical learning standard from the sixth-grade Spanish curriculum: calculate the least common multiple (LCM) and the greatest common divisor (GCD), and solve problems involving these concepts. As opposed to the first phase, different conditions were encountered by the CG and the EG during the second stage. While the EG used *Scratch* as a tool to solve the different tasks proposed, the CG worked in a classic paper-and-pencil environment. All the sessions were held once a week.

Before the programming phase, both groups completed a pre-test in order to assess their level in the field of computational thinking before any intervention. Likewise, at the beginning of the mathematics phase, they also completed a pre-test in which the students' mathematical level in relation to the learning standard addressed in the study was measured. All the tests carried out in this study lasted 60 min and were conducted at the same time in both groups. At the end of the second phase, both groups of students completed two post-tests with the aim of assessing the student's level in computational thinking and the acquisition of the mathematics learning standards, respectively. The comparison between pre- and post-tests will allow us to analyse to what extent the acquisition of computational concepts of CT and mathematical knowledge evolved for each experimental condition.

Instruments

Computational thinking test

The *Computational Thinking Test* (CTT) was composed of ten items aimed at evaluating four computational concepts: sequences, iterations or loops, handling of events and conditional sentences. The test was based on the instrument proposed in Román-González et al. (2017), where a computational thinking test aimed at 5th to 10th grades is provided. The specific structure within the questionnaire responds to a less-to-greater difficulty ordering according to the computational concepts previously presented. In each item, the different answers are presented through a set of symbols which must be associated with a succession of visual orders in the exercise (e.g. see [Figures 2](#) and [3](#)).

[Table 1](#) lists the different items used in the questionnaire organised according to the main computational concept of CT addressed. Every item involves at least one of the computational concepts previously mentioned, although due to the nature of these concepts the combined use of elements is usually necessary as the difficulty increases. The post-test used to measure the students' level of acquisition of CT (CTT_{post}) is a version of the pre-test (CTT_{pre}) with the same number of items, but with some variations in the items used with the objective of exactly measuring the same objectives. Both tests related to computational thinking (CTT_{pre} and CTT_{post}) were designed as online questionnaires and, as a result, participants completed them in the computer classroom.

Mathematical knowledge test

The second instrument, the *Mathematical Knowledge Test* (MKT), used in the experiment was a test designed with the aim of evaluating the selected mathematics learning standard. As in the case of

1. What succession of letters will draw a square under the triangle?

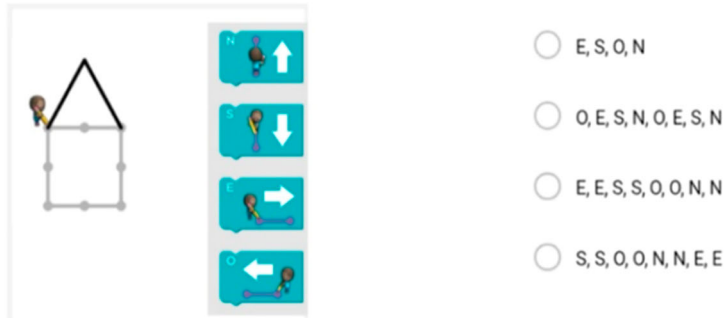


Figure 2. Example – Item from CTT (Sequences).

6. How many times will you have to repeat the orders (E,N,E,S) to complete the castle?

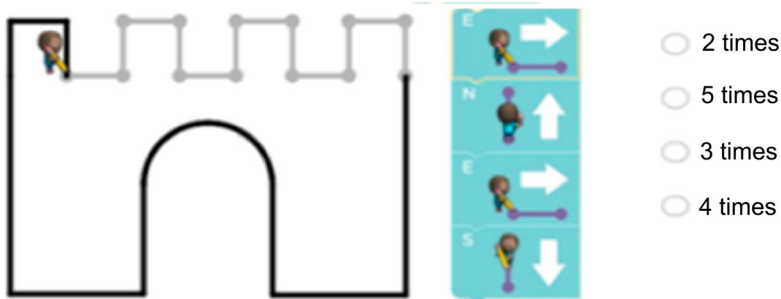


Figure 3. Example – Item from CTT (Iterations).

CT, two versions of the same test with the same structure and features were employed for the pre-test (MKT_{pre}) and the post-test (MKT_{post}), respectively. The test was made up of four items, taken from a specific instrument employed in González-Calero, Martínez, and Sotos (2016). Each item is assessed, depending on the degree of acceptance of the questionnaire. In Table 2, a brief justification of each item is presented, together with its literal statement.

Procedure

As we said earlier, during the programming phase the sessions were focused on the *Scratch* environment. The design aimed to address the computational concepts and, in turn, offer basic instruction

Table 1. Computational concepts in computational thinking test.

Item	Computational concepts involved	Computational concept evaluated
1	Sequences	Sequences
2	Sequences	Sequences
3	Sequences /Event handling	Event handling
4	Loops/Event handling	Event handling
5	Sequences / Loops	Loops
6	Loops	Loops
7	Conditional/ Event handling	Conditional
8	Sequences/Conditional/Event handling	Conditional
9	Sequences/ Event handling / Loops	Loops
10	Sequences /Iteration /Conditional	Conditional

Table 2. Classification and description of items of MKT.

Item	Description	Statement
1	Word problem whose solution implies calculating the greatest common divisor. The word <i>maximum</i> is used in the statement.	Two ropes, 18 and 24 cm long, respectively, must to be cut into equal pieces. Find out the maximum length of these pieces.
2	Word problem whose solution implies calculating the greatest common divisor. The word <i>minimum</i> is used in the statement.	We have two wires, one 42 m long and the other 35 m long. If we want to cut them into the minimum number of equal pieces, what will be the length of the pieces?
3	Word problem whose solution implies calculating the least common multiple. The word <i>maximum</i> is used in the statement.	Philip and Albert study at UCLM, and coincide from time to time in some classes, practical classes, etc. In addition, whatever happens, Philip goes to the cafeteria every 18 days and Albert every 15. If today they coincide at the cafeteria, work out the maximum number of days it would take for them to meet again.
4	Word problem whose solution implies calculating the least common multiple. The word <i>minimum</i> is used in the statement.	One beacon flashes its light every 12 s, and another one every 18 s. If they have just flashed simultaneously, find out what the minimum time to flash together again will be.

that allowed students to use *Scratch* as a tool to solve mathematical problems. After a first session aimed at the students becoming familiar with the *Scratch* environment, each session was devoted to specifically working each of the computational concepts. Furthermore, following the recommendations of Brennan et al. (2014), each session was organised in two different periods: firstly, where group activities were carried out, and in which the component of the computational thinking in question was introduced in tasks where *Scratch* was not yet used. After this, a second part was developed, where the concept was transferred to activities with *Scratch*. Additionally, this design is aligned with suggestions presented in Benton et al. (2017), who support activities in which the students themselves play the role of a programming object as this constitutes an adequate basis to later identify actions in the programming language. For example, in one of the activities the group was divided into pairs and one of the members watched a video showing a dance. Then, this student, using just a sequence of words, instructed his/her partner to do the dance. This type of activity works as a bridge to help students understand and assimilate computational concepts and to practice them more easily than in a real programming environment.

In the mathematics phase, the EG used *Scratch* to solve word problems. In particular, students worked in pairs using a computer. It was decided to group the students this way in order to facilitate student interactions and discussions concerning the solution to the different problems. Each pair was asked to develop a *Scratch* project in which they had to solve a problem related to GCD and LCM. At the beginning of the first session, the teacher (one of the authors) showed different examples of how the situations depicted in the word problems may be modelled using *Scratch* (e.g. see Figure 4). Some of the instructions (set of blocks) used in the example projects were highlighted; for instance, the different use of control blocks (loops and conditionals) or a sequence order that students should use in their projects. Different recommendations were made, such as the possibility of copying parts of the blocks that were already developed in all the previous exercises. After presenting these examples, each pair of students was given a list of word problems, completely different from the examples used in the programming phase, and were asked to use *Scratch* to solve them. In particular, they were asked to code a programme for each problem that could be of use in order to find the solution.

On the other hand, the CG worked on exactly the same problems as the EG, but using a traditional approach in a paper-and-pencil setting. In order not to introduce differences between conditions, students were also grouped in pairs in the CG, and they were invited to discuss their solutions with their partners. In addition, the same examples presented in the EG were also offered at the beginning of each session to the CG as an example of models that can be employed to represent these mathematical situations. As students in the EG had to code a programme for each problem, the pace of solving problems was slower in comparison to the CG. In fact, while students from the EG completed three



Figure 4. Screenshot of a word problem modelled using Scratch.

problems (one per session) in this phase, participants from the CG solved six problems (two per session).

In relation to mathematical knowledge, the only feedback that students from both the CG and EG were given is that, at the end of each session, they were informed about the numerical result of each problem so that they could check if they had solved them properly.

Results

RQ1. Computational thinking

In order to test the first aim of the study, differences between CTT_{pre} and CTT_{post} are compared. On both tests, correct answers were given a score of 1, and incorrect ones 0. Finally, each student was assigned a global score on each test as a result of adding the scores for each question. Before analysing eventual gains on students' computational thinking during the intervention, an independent t -test was employed to assure that the students' prior level in computational thinking between conditions was homogenous. Both groups scored no significantly different in the CTT_{pre} ($t(44.04) = -0.92$, $p = .3612$, $r = 0.14$). After checking that both groups were initially comparable, a paired-sample t -test was conducted to examine if there were significant differences in the CTT before and after the intervention. Prior to conducting the analysis, we checked that the assumptions for parametric tests were satisfied. On average, students showed better results in the CTT_{post} ($M = 7.15$, $SD = 1.93$) than in the CTT_{pre} ($M = 4.68$, $SD = 1.78$), $t(46) = -6.23$, $p < .001$, $r = .68$).

At the same time, students from CG showed similar gains in their computational thinking between pre-test and post-test ($M = 2.48$, $SD = 0.63$) compared to students from EG ($M = 2.46$, $SD = 0.51$). An independent t -test confirmed the absence of statistically significant differences between groups ($t(42.77) = 0.02$, $p = .9804$, $r = .004$).

The assumptions for parametric tests were not fulfilled when studying gains between CTT_{pre} and CTT_{post} for each computational concept. Consequently, we used a non-parametric test, the Wilcoxon signed-rank test. Results showed statistical differences after the intervention for all the components: sequences ($p = .0007$, $r = 0.35$), loops ($p = .0285$, $r = 0.23$), conditionals ($p < .001$, $r = 0.41$) and events-handling ($p < .001$, $r = 0.37$).

Regarding RQ1, the results seem to indicate a significant effect of explicit instruction in computational concepts on the level of the participants' computational thinking. In addition, no differences in CT were detected between the EG and CG, despite the fact that participants in the EG worked with Scratch during the mathematics phase, while students in the CG just worked in a paper-and-pencil environment.

RQ2. Mathematical knowledge

As in the case of the CTT, each student was assigned a global score in the MKT_{pre} and MKT_{post} as a result of adding the scores for each question. Similarly, completely correct solutions were scored as 1, and incorrect ones 0. Since the assumption of normality was not satisfied, a non-parametric test, the Wilcoxon rank-sum test, was used to analyse the students' prior level in the mathematical learning standard within the scope of the study. No statistical differences were observed between both groups in the MKT_{pre} ($W = 285.5$, $p = .8349$, $r = .03$).

Again, since assumptions for parametric tests were not fulfilled, Wilcoxon signed-ranked tests were conducted to examine if there were differences in the students' mathematical performance before and after the intervention. No significant differences were shown in the CG between MKT_{pre} ($M = 1.96$, $SD = 1.07$) and MKT_{post} ($M = 2.43$, $SD = 1.44$), although it did represent a medium-sized effect ($p = .0522$, $r = 0.29$). On the other hand, the tests revealed a significant improvement in the EG between MKT_{pre} ($M = 1.92$, $SD = 1.44$) and MKT_{post} ($M = 2.54$, $SD = 1.22$), but also with a medium-sized effect ($p = .0224$, $r = 0.33$).

With regard to RQ2, the data gathered reveals a relevant improvement in the sixth-grade students' proficiency in solving word problems related to LCM and GCF after completing programming activities with *Scratch*. Although the differences between MKT_{pre} and MKT_{post} are only statistically significant in the EG, the comparisons of the effect sizes reveal that learning gains between both conditions are very similar. As a result, although students from the EG solved a lower number of problems than students from the CG during the intervention, we can affirm that the use of *Scratch* does not produce a negative effect on the students' learning of the mathematical concepts addressed in the present work in comparison with students instructed using a traditional approach.

Conclusions

The main objective of this piece of research was to analyse the potential of programming activities using *Scratch* for both the learning of mathematical ideas and the acquisition of computational thinking (CT) in sixth-grade students. First, our results make it clear that the students' prior experience with CT was scarce. This fact indicates that the development of computational concepts does not incidentally take place over the elementary education stage. On the contrary, it may be an indicator that explicit instruction is necessary in order to develop basic computational concepts. As a consequence, modifications should be introduced to official curricula to appropriately tackle the question of CT, which is considered to be a critical skill for citizens in future years (Wing, 2008). Related to this, our results underline the effectiveness of specific instruction in developing computational concepts. In particular, after just five sessions a significant increase in the students' level of CT was detected, whose effect size, according to Cohen (1988), may be considered as very large.

Concerning the students' proficiency in solving word problems related to the concepts of LCM and GCD, a statistically significant improvement has been detected in the participants who solved problems using *Scratch*, while no significant differences occurred in the control group. Besides this fact, a more detailed analysis shows that gains in both conditions, although slightly higher for the experimental group, may be classified as medium-sized. Although previous studies underlined the benefits of programming activities with *Scratch* in the field of problem-solving (e.g. Benton et al., 2017; Calao et al., 2015; Calder, 2010), our results do not indicate a substantial improvement in comparison with a more traditional approach. However, our data seems to confirm that no detrimental effects are produced due to the teaching of problem solving through *Scratch*. This fact opens the door to cross-disciplinary approaches in which computational and mathematical concepts can be addressed simultaneously.

There are some limitations that can lead us to recommending future studies. First, the work addresses a specific typology of word problems, and the duration of the mathematical phase is short. Supported by our results, longitudinal research could be designed to deal with concepts

and skills from other mathematical areas (e.g. geometry or algebra). In addition, this study was not designed to analyse other interesting variables, such as the students' attitude towards mathematics or the student's engagement over time. However, longitudinal studies with a larger sample could investigate such variables to further enrich research and practice. Similarly, it should be underlined that the present work has evaluated the sixth-grade students' acquisition of a specific subset of basic computational concepts. Nonetheless, other computational concepts, such as operators and expressions, have not formally included in this study. Additional proposals and research studies are necessary to evaluate the learning of more abstract computational concepts by Primary school students.

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

José Antonio Rodríguez is a primary education teacher. He is member of the research group *Labintic*, which is focused on the analysis of the integration of technology in teaching and learning. His main research interests include educational mathematics, computational thinking, and learning analytics.

José Antonio González-Calero is currently a lecturer in the School of Education of Albacete at the University of Castilla-La Mancha. He is a member of the Spanish Society of Research in Mathematics Education (SEIEM). His research interests include educational mathematics, word problem solving, and interactive learning environments.

José-Manuel Sáez-López is a lecturer in the Faculty of Education at the National University of Distance Education (UNED). His lines of research are the integration of educational technology, methodological strategies, gamification, and programming in K-12. It has been recognised as Microsoft Expert Educator 2014 and Microsoft Innovative Expert in 2015.

ORCID

José Antonio Rodríguez-Martínez  <http://orcid.org/0000-0001-8151-676X>

José Antonio González-Calero  <http://orcid.org/0000-0003-0842-8151>

José Manuel Sáez-López  <http://orcid.org/0000-0001-5938-1547>

References

- Bar-On, E. (1986). A programming approach to mathematics. *Computers & Education*, 10(4), 393–401. doi:10.1016/0360-1315(86)90015-1
- Benton, L., Hoyles, C., Kalas, I., & Noss, R. (2017). Bridging primary programming and mathematics: Some findings of design research in England. *Digital Experiences in Mathematics Education*, 3(2), 115–138. doi:10.1007/s40751-017-0028-x
- Bintaş, J., & Çamlı, H. (2009). The effect of computer aided instruction on students' success in solving LCM and GCF problems. *Procedia - Social and Behavioral Sciences*, 1(1), 277–280. doi:10.1016/J.SBSPRO.2009.01.050
- Brennan, K., Balch, C., & Chung, M. (2014). *Creative computing*. Retrieved from <http://scratched.gse.harvard.edu/guide/>
- Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. Annual American Educational Research Association Meeting, Vancouver, BC, Canada, 1–25. <https://doi.org/10.1.1.296.6602>
- Brown, A., Thomas, K., & Toliaş, G. (2002). Conceptions of divisibility: Success and understanding. In S. Campbell & R. Zakis (Eds.), *Learning and teaching number theory* (pp. 41–82). Westport, CA: Ablex.
- Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research*, 87(4), 834–860. doi:10.3102/0034654317710096
- Cabero, J., & Llorente, M. d. C. (2010). Comunidades virtuales para el aprendizaje. *EduTec. Revista Electrónica de Tecnología Educativa*, 34, 1–10. doi:10.21556/EDUTEC.2010.34.419
- Calao, L. A., Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with *Scratch*. In G. Conole, T. Klobučar, C. Rensing, J. Konert, & E. Lavoué (Eds.), *Design for Teaching and Learning in a Networked World. Lecture Notes in Computer Science*, vol 9307 (pp. 17–27). Cham: Springerdoi:10.1007/978-3-319-24258-3_2

- Calder, N. (2010). Using *Scratch*: An integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom*, 15(4), 9–14. doi:10.1007/s10857-012-9226-z
- Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. *Computers and Education*, 95, 202–215. doi:10.1016/j.compedu.2016.01.010
- Chen, G., Shen, J., Barth-Cohen, L., Jiang, S., Huang, X., & Eltoukhy, M. (2017). Assessing elementary students' computational thinking in everyday reasoning and robotics programming. *Computers and Education*, 109, 162–175. doi:10.1016/j.compedu.2017.03.001
- Cohen, J. (1988). *Statistical power analysis for the behavioral sciences*. New York, NY: Routledge Academic.
- de Morais, A. D., Basso, M. V. d. A., & Fagundes, L. d. C. (2017). Educação Matemática e Ciência da Computação na escola: aprender a programar fomenta a aprendizagem de matemática? *Ciência & Educação (Bauru)*, 23(2), 455–473. doi:10.1590/1516-731320170020011
- DiSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge: MIT Press.
- Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, 63, 87–97. doi:10.1016/J.COMPEDU.2012.11.016
- Feurzeig, W., Papert, S. A., & Lawler, B. (2011). Programming-languages as a conceptual framework for teaching mathematics. *Interactive Learning Environments*, 19(5), 487–501. doi:10.1080/10494820903520040
- Funke, A., Geldreich, K., & Hubwieser, P. (2017). Analysis of *Scratch* projects of an introductory programming course for primary school students. In 2017 *IEEE global engineering education conference (EDUCON)* (pp. 1229–1236). IEEE. doi:10.1109/EDUCON.2017.7943005
- Furber, S. (2012). *Shutdown or restart? The way forward for computing in UK schools*. London: The Royal Society.
- Garneli, V., & Chorianopoulos, K. (2018). Programming video games and simulations in science education: Exploring computational thinking through code analysis. *Interactive Learning Environments*, 26(3), 386–401. doi:10.1080/10494820.2017.1337036
- González-Calero, J. A., Martínez, S., & Sotos, M. A. (2016). La tendencia a restar en la resolución de problemas de m.c.d. en alumnos de primaria. In J. A. Macías, A. Jiménez, J. L. González, M. T. Sánchez, P. Hernández, C. Fernández, ... A. Berciano (Eds.), *Investigación en Educación Matemática XX* (pp. 295–304). Málaga: SEIEM.
- Gutiérrez-Gutiérrez, A., Gómez Guzmán, P., & Rico Romero, L. (2015). Conocimiento matemático sobre números y operaciones de los estudiantes de Magisterio. *Educación XX1*, 19(1). doi:10.5944/educxx1.15581
- Hickmott, D., Prieto-Rodríguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education*, 4(1), 48–69. doi:10.1007/s40751-017-0038-8
- Howe, J. A. M., Ross, P. M., Johnson, K. R., Plane, F., & Inglis, R. (1982). Teaching mathematics through programming in the classroom. *Computers & Education*, 6(1), 85–91. doi:10.1016/0360-1315(82)90016-1
- Hughes, J., Gadanidis, G., & Yiu, C. (2017). Digital making in elementary mathematics education. *Digital Experiences in Mathematics Education*, 3(2), 139–153. doi:10.1007/s40751-016-0020-x
- Kahn, K., Sendova, E., Sacristán, A. I., & Noss, R. (2011). Young students exploring cardinality by constructing infinite processes. *Technology, Knowledge and Learning*, 16(1), 3–34. doi:10.1007/s10758-011-9175-0
- Layman, J., & Hall, W. (1988). Logo: A cause for concern. *Computers & Education*, 12(1), 107–112. doi:10.1016/0360-1315(88)90063-2
- Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, 41, 51–61. doi:10.1016/j.chb.2014.09.012
- Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice. In *Proceedings of the 39th SIGCSE technical symposium on computer science education - SIGCSE '08* (Vol. 40, pp. 367–371). New York: ACM Press. doi:10.1145/1352135.1352260
- Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference - ITICSE-WGR '14* (pp. 1–29). New York, NY: ACM Press. doi:10.1145/2713609.2713610.
- Marmolejo, J. E., & Campos, V. (2013). Pensamiento lógico matemático con *Scratch* en nivel básico. *Revista Vínculos*, 9(1), 87–95. doi:10.14483/issn.2322-939X
- Noblet, K. (2013). Preservice elementary teachers' understanding of greatest common factor story problems. In *Proceedings of the 16th annual conference on research in undergraduate mathematics education* (pp. 219–225). Denver: Sigmaa.
- Papert, S. A. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York: Basic Books.
- Papert, S.A. (1990). A critique of technocentrism in thinking about the school of the future. Cambridge, MA: Epistemology and Learning Group, MIT Media Laboratory.
- Pea, R. D., Kurland, D. M., & Hawkins, J. (1985). Logo and the development of thinking skills. In M. Chen & W. Paisley (Eds.), *Children and microcomputers: Research on the newest medium* (pp. 193–317). Norwood, NJ: Ablex Publishing Corp.
- Resnick, M., Silverman, B., Kafai, Y., Maloney, J., Monroy-Hernández, A., Rusk, N., ... Silver, J. (2009). *Scratch: Programming for all*. *Communications of the ACM*, 52(11), 60–67. doi:10.1145/1592761.1592779
- Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the computational thinking test. *Computers in Human Behavior*, 72, 678–691. doi:10.1016/j.chb.2016.08.047

- Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., ... Verno, A. (2011). *CSTA k–12 computer science standards: Revised 2011*. New York, NY: ACM.
- Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, 22, 142–158. doi:10.1016/j.edurev.2017.09.003
- Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1), 127–147. doi:10.1007/s10956-015-9581-5
- Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3725. doi:10.1098/rsta.2008.0118
- Wing, J. M. (2014). *Computational thinking benefits society* [Blog post]. Retrieved from <http://socialissues.cs.toronto.edu/index.html%3Fp=279.html>