H. Chad Lane
Susan Zvacek
James Uhomoibhi (Eds.)

Communications in Computer and Information Science 1473

# Computer Supported Education

12th International Conference, CSEDU 2020
Virtual Event, May 2–4, 2020
Revised Selected Papers

Springer

*Editors*
H. Chad Lane
University of Illinois
Urbana-Champaign, IL, USA

Susan Zvacek
University of Denver
Denver, CO, USA

James Uhomoibhi
School of Engineering
University of Ulster
Newtownabbey, UK

ISSN 1865-0929 ISSN 1865-0937 (electronic)
Communications in Computer and Information Science
ISBN 978-3-030-86438-5 ISBN 978-3-030-86439-2 (eBook)
https://doi.org/10.1007/978-3-030-86439-2

# Contents

# The Code.org Platform in the Developing of Computational Thinking with Elementary School Students

Rolando Barradas[1,2,4](✉), José Alberto Lencastre[3], Salviano Soares[1,4], and António Valente[1,2]

[1] School of Sciences and Technology, University of Trás-os-Montes and Alto Douro, Quinta de Prados, Vila Real, Portugal
rolando.barradas@inesctec.pt
[2] INESC TEC, Porto, Portugal
[3] CIEd - Research Centre On Education, Institute of Education, University of Minho, Campus de Gualtar, Braga, Portugal
[4] IEETA, UA Campus, Aveiro, Portugal

**Abstract.** Computational thinking is the thinking process involved in formulating problems to admit a computational solution. This article describes a study in which the code.org platform was used to develop computational thinking with Elementary school students. After proper introduction and contextualization, we describe the 198 students from 4th grade involved in the study, following the process of collecting and analyzing data from the code.org platform. We conclude with the evaluation carried out by the students. The main conclusion of this study is that code.org is a valid option for developing computational thinking with Elementary school students. Also, a reliable way for students to start solving real-life problems, stimulating the capacity for abstraction through simulated and experienced practice.

**Keywords:** Computational thinking · Code.org platform · Technology-enhanced learning · Elementary school students

## 1 Introduction

Computational thinking is used in the design and analysis of problems and their solutions. The most crucial thought process in computational thinking is abstraction [1, 2]. Abstraction is used in determining patterns, generalizing from instances, and parameterization. It is used to let one object stand for many. It is used to capture essential properties common to a set of objects. For instance, an algorithm is an abstraction of a process that receives input, performs a sequence of steps, and produces an output. An abstract data type defines an abstract set of values and operations for manipulating those values, hiding the actual representation of the benefits from the user of the abstract data type. Designing efficient algorithms inherently involves creating abstract data types. Abstraction gives us the power to scale and deal with complexity [3].

H. C. Lane et al. (Eds.): CSEDU 2020, CCIS 1473, pp. 118–145, 2021.
https://doi.org/10.1007/978-3-030-86439-2_7

## 2 Contextualization

Today we can find computers everywhere. Regardless of the physical form in which we find them, they are used daily in fields like industry, science, education, and entertainment. First in our desktops, and now in our pockets. Nowadays, most of us carry smartphones millions of times more capable than all the computing power NASA (National Aeronautics and Space Administration) had in 1969 when Apollo 11 landed on the Moon [4].

From 1946 to present day, we watched an exponential evolution that drove us from the 27 ton and roughly 2,4 m × 0,9 m × 30 m, military-developed Electronic Numerical Integrator and Computer (ENIAC) to the smallest computer in the world, capable of fitting on a grain of rice, a small cube with 0,3 mm per side [5].

The use of computers is a constant in our daily lives and the processing power of computers has continuously increased. Yet we rarely think about this evolution, since for most users today all this technology and processing power it's taken for granted as they've always lived surrounded by it.

Generation Z was the one of true digital natives. To them nothing exists without Google or a smartphone and commanding devices by voice it's part of everyday life. But their children already belong to Generation Alpha, the first generation who will be immersed in technology their whole lives. They are more comfortable using a touchscreen device or speaking to a voice assistant than most of their adult relatives. Technology is somehow intuitive to them and they are massive content consumers. Also, soft skills entered their learning plans as they concentrate on things like problem-solving, multi-tasking, and quick thinking [6], essential skills for the 21st century [7].

Regardless of age, educational background, or professional occupation, computers are easily usable by anyone, At the same time, it is now almost imperative for everyone to have, at least, basic computer skills, that include Internet and email, word processing, graphics and multimedia, and spreadsheets. Thanks to increasingly user-friendly interfaces, applications and computers can now be used without expert knowledge to solve complex problems or technical tasks as well as for the most varied situations of everyday life. Yet, according to Resnick, Maloney, Monroy-Hernández, Rusk, and Astmon [2, 8], one of the biggest challenges that users face nowadays is the need to stop from being mere content consumers of programs and games to become creators of such contents. To do so, one needs only to be aware of a small number of basic applications and tools and sufficiently curious to search for information on the Internet or be able to do so through a trial and error process.

Since the beginning of the millennium, ICT classes (Information and Communication Technologies) have become mandatory in schools. However, until recently, the study of ICT focused on the transmission of knowledge about computer tools and a basic set of software and hardware, and how to use them to solve everyday tasks. Things like writing a text, browsing the internet, and learning how to use it to communicate efficiently are some of the knowledge that was transmitted in ICT classes. A natural evolution of the contents would be to make users understand how these tools work and how they are built.

Coding is a way of developing creative activities with children from Generation Alpha as it allows them to gain a broader view of computer uses, creatively solving real-world problems, by focusing primarily on design, planning, and implementation of a project.

It's in this context that it becomes necessary to mention another essential competence for the 21st century: computational thinking [9].

## 2.1 Computational Thinking

Computational thinking can be defined as the set of processes involved in formulating a problem and its solutions so that a human or machine can effectively solve it [10, 11], and it's more connected to conceptualization than to coding itself [9].

The development of computational thinking implies developing other skills such as (i) abstract thinking – understanding problems and solving them using different levels of abstraction -, (ii) algorithmic thinking – expressing a solution to find the most effective way to solve a problem - (iii) logical thinking - formulating and excluding hypotheses - and (iv) measurable thinking - breaking up a big problem into small parts or joining small parts to attain a more complex solution [2].

Brennan and Resnick's studies on computational thinking and the creation of interactive media products allowed them to create a framework of reference for studying and evaluating the development of computational thinking. This framework consists of three dimensions: **(i) computational concepts**, **(ii) computational practices**, and **(iii) computational perspectives**.

Concerning **computational concepts (i)**, Brennan and Resnick [3] were able to identify seven:

**Sequences** – A set of steps or instructions that can be executed to complete a coding task. In a sequence of instructions, it is important to define the correct order of execution since changing the order of one of them could lead to completely different results;
**Loops** – Mechanisms that allow the execution of the same sequence a given number of times. In certain types of problems, it is possible to identify patterns of repetition and use them to simplify the Code;
**Events** – An event is an occurrence of something that causes an action to execute;
**Parallelism** – The solution to certain problems implies that several sequences take place at the same time;
**Conditionals** – Conditions allow a program to make decisions and, through the use of decision structures, execute, or not, some piece of Code;
**Operators** – Used to express and solve mathematical and logical operations;
**Data** – Data structures are used to store, retrieve, and update values needed for the execution of a program.

The **computational practices (ii)** associated with the act of programming, are focused on the construction process [3]. They also target the processes of thinking and learning by changing the focus from what is learned to how it is learned. In their studies, Brennan and Resnick identified four sets of practices:

**Being Incremental and Iterative** – Is the practice by which children develop and check whether a project works and continue to develop new approaches to the solution, in case it's not the final one;

**Testing and Debugging** – Is the practice by which, through trial-error processes and the analysis of previously solved problems, children check what is wrong and does not work and correct it;
**Reusing and Remixing** – Is the practice in which children learn by building something new, using their old projects or projects that others have already done and shared;
**Abstracting and Modularizing** – Is the practice by which it's possible to reach a big solution by joining sets of smaller parts since complex problems can be divided into smaller and simpler problems, easier to solve.

While studying interactive media products, Brennan and Resnick identified three major **computational perspectives (iii)** of children in their relation to computing, and categorized them as:

**Express** - as computing is a means of creation and self-expression, it allows students to start to see themselves not only as consumers but also as builders;
**Collaborate/Connect** - computing gives freedom of creation with and for the others. Therefore, it allows the development of a critical spirit by making one's creations an inspiration for new projects and those of others;
**Questioning** - Trying to understand how certain problems were solved can lead to questioning the functioning of other real-world problems [12].

Therefore, analyzing the execution of projects and activities, taking into account the three dimensions described above, it will be possible to evaluate the development of computational thinking in young people.

Using coding as a way to develop computational thinking also stimulates students' creativity by making them solve real-world problems, tangible or not. According to Jonassen [13], the most relevant educational activities that students can perform are problem-solving activities because the knowledge built during the process is better understood and more easily retained. Also, by using the problem-solving method, students "learn how to learn" [14] because while looking for a solution for a problem instead of waiting for an answer they develop their domain of the procedures [15]. Using these teaching methods tends to increase the motivation of students. They become the main agent in the learning process.

## 3 Code.org

Code.org (Fig. 1) is a nonprofit organization and website, founded in 2013, dedicated to expand access to computer science in schools all over the world and to increase the participation of women and underrepresented minorities.

Their learning platform maintains an extended set of educative resources and tools that can be used in almost every platform, smartphones, and tablets included, thus being very flexible and easy to use.

Its vision is that every student in the world should have the opportunity to learn computer science, just like they learn biology, chemistry, or algebra [16].

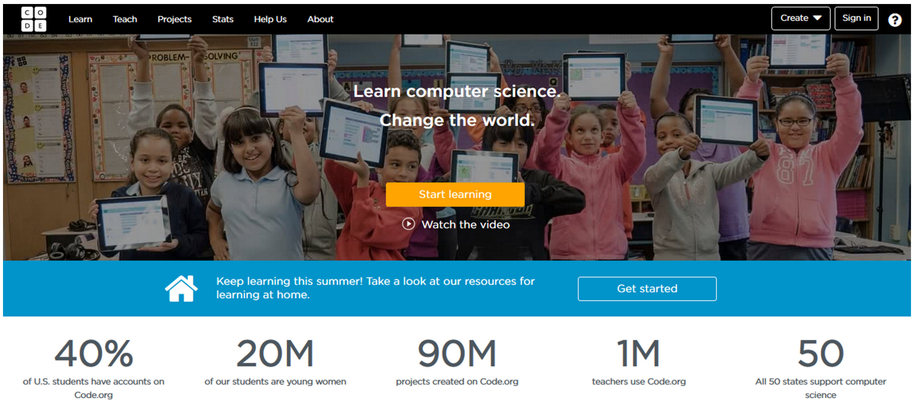

C O D E
Learn
Teach
Projects
Stats
Help Us
About
Create
Sign in
Learn computer science.
Change the world.
Start learning
Watch the video
Keep learning this summer! Take a look at our resources for learning at home.
Get started
40%
of U.S. students have accounts on Code.org
20M
of our students are young women
90M
projects created on Code.org
1M
teachers use Code.org
50
All 50 states support computer science

**Fig. 1.** Code.org homepage (adapted from [19]).

But what explains such impressive numbers, as the ones observed in Fig. 1? What makes the platform so attractive to young students? In our analysis we conducted to the platform we identified several details that we consider important and might explain its success.

## 3.1 Assessing the Content

Everyone can try to Code in Code.org. Students can even start testing the platform without creating an account. Nevertheless, access to the platform has been built to respond to almost any existing combination today. If students already have a pre-existing account from other platforms, such as Google, Facebook, and Microsoft, they can use it to access the contents. If not, they can create an account on the platform (see Fig. 2).

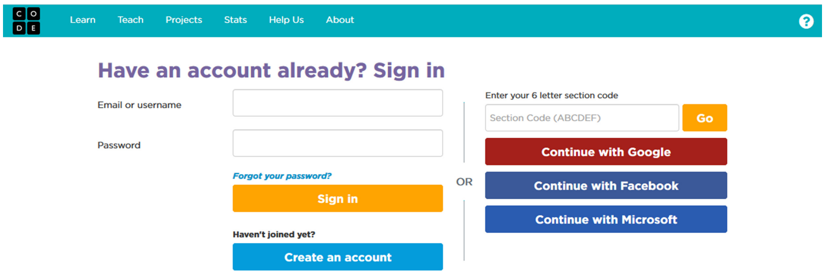


**Fig. 2.** Login page from the Code.org platform (adapted from [19]).

Any situation in which students log in, allows them to keep track of their progress and Sign in at a later time so that they can continue solving tasks.

Notice that, for a teacher to be able to monitor the learning process of each one of his students individually, they must have an account and the teacher should have created a class and assigned students to it.

Also, the fact that the platform is built as a web application, running entirely in a web browser allows you to start working more rapidly as there is no need to pre-install or configure the devices that will be used. All the learning process is done in a graphical environment through drag-and-drop of instruction blocks that students use to build their programs and solve each of the challenges (see Fig. 3).

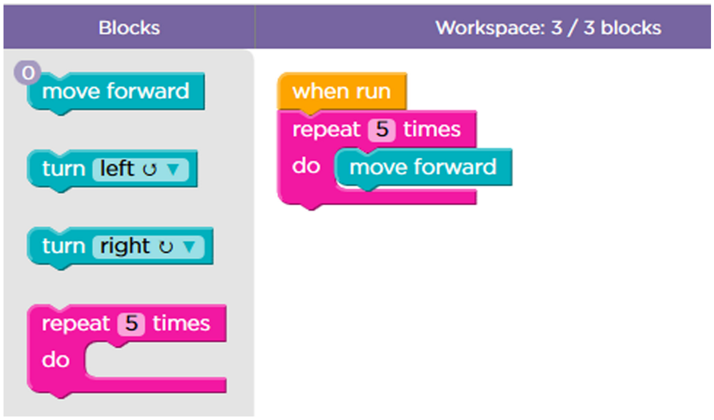


**Fig. 3.** Coding by blocks.

## 3.2 Course Organization and Self-efficacy Control Tools

The Computer Science Fundamentals, Course 2, the object of this study, is organized in 19 Lessons, online and offline, grouped by computing concepts and with increasing difficulty. Note that for this study, only online lessons finishing with Lesson 16 - Flappy Bird were considered. Every lesson starts with an introductory video that explains the lesson's objectives and gives students some insight into the type of problems they will encounter. The course includes Lessons about sequences, loops, and all the other computational concepts identified by Brennan and Resnick [3], and ends with slightly more complex concepts such as nested loops and functions.

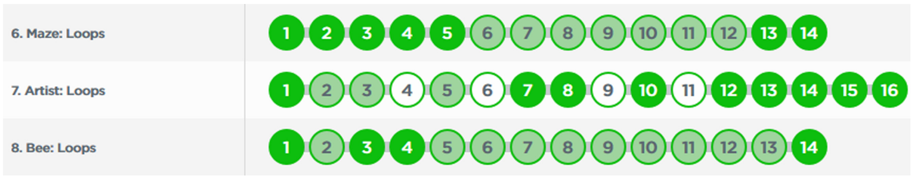


**Fig. 4.** Viewing progress for self-efficacy control.

In each coding task, students have information about the maximum number of blocks that they can use to solve the problem with the optimal solution. If a student can solve a problem using the optimal solution, the problem number will be painted in green, on their progress overview. If they manage to solve it but not with the optimal solution, the problem number will be painted with a light green color, as seen in Fig. 4. This way, students can easily see the problems that need attention and retry that problem at a later time to find the optimal solution.

The course overview is a tool that allows students to monitor their effectiveness, and to compare their progress with their peers, serving as motivation for task solving.

## 3.3 Well-Chosen Characters

When designing the platform challenges, the designers had in mind the target audience.

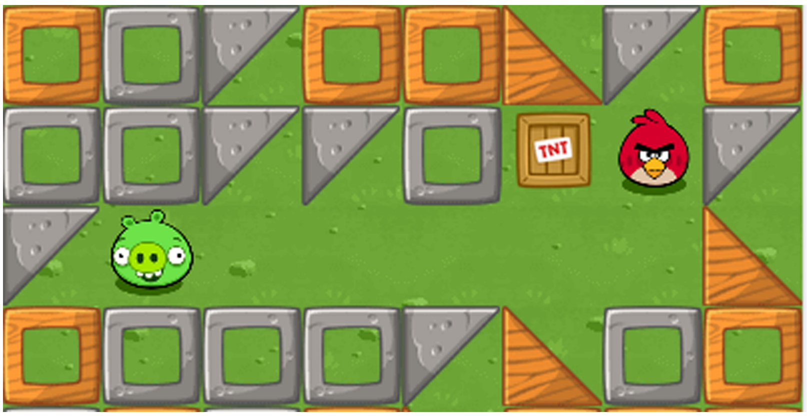


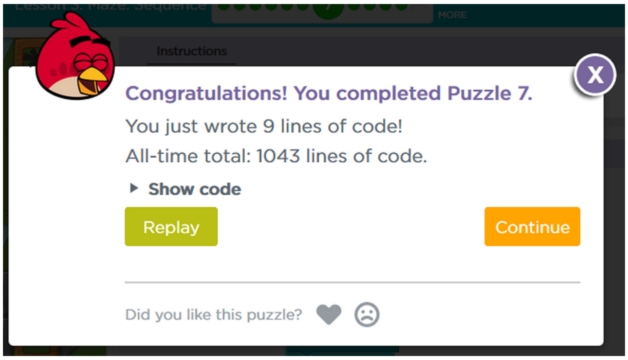
**Fig. 5.** Example of a challenge involving familiar student characters.

All the challenges were created using animations, sounds, and characters according to the age of the target group. Angry Birds, pictured in Fig. 5, and Plants vs. Zombies are some of the themes and characters used in many of the challenges, turning them even more attractive to students.

## 3.4 Instant Feedback

Talking about gamified instruction, Kapp, Blair, and Mesch [17], stated that instant feedback was one of the most important elements in a Gamification system. Gamification can be described as the use of characteristic elements of games in non-game situations, for example, the existence of reward systems, levels of difficulty, scoring tables, time limits, resource limits, definition of clear objectives, variety of game type and a narrative that contextualizes them [18].

Code.org created a gamified system that maximizes feedback. At the end of each challenge, students are notified about what they have already completed and what mistakes they have made (See Fig. 4). When students are successful in their task they are immediately rewarded with a message and sound that indicates success and joy (see Fig. 6).


Congratulations! You completed Puzzle 7.

You just wrote 9 lines of code!

All-time total: 1043 lines of code.

▸ Show code

Replay Continue

Did you like this puzzle?

**Fig. 6.** Instant feedback associated with positive reinforcement.

On the other hand, when the solution found is not the right one, students are informed immediately with messages of encouragement (see Fig. 7).

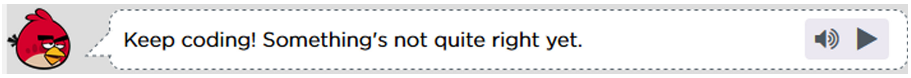
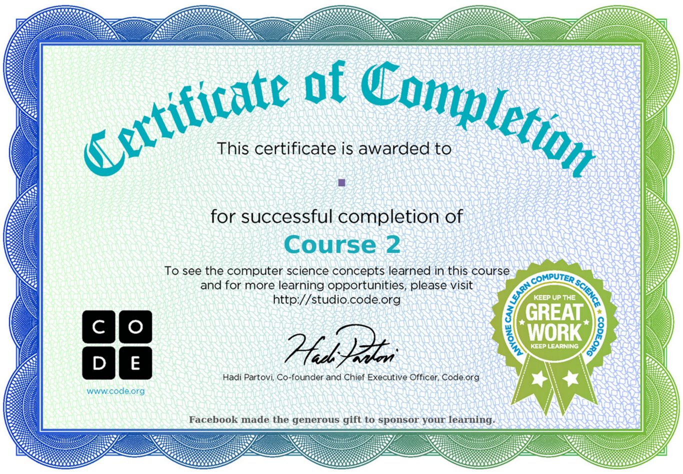

Keep coding! Something's not quite right yet.

**Fig. 7.** Instant feedback associated with non-success.

This simple process of instant feedback proves itself very effective by allowing students to immediately check their progress and compare it with others.

## 3.5 Reward System


Certificate of Completion

This certificate is awarded to

for successful completion of

Course 2

To see the computer science concepts learned in this course and for more learning opportunities, please visit http://studio.code.org

Hadi Partovi, Co-founder and Chief Executive Officer, Code.org

www.code.org

Facebook made the generous gift to sponsor your learning.

C O D E
ANYONE CAN LEARN COMPUTER SCIENCE
KEEP UP THE GREAT WORK KEEP LEARNING
CODE.ORG

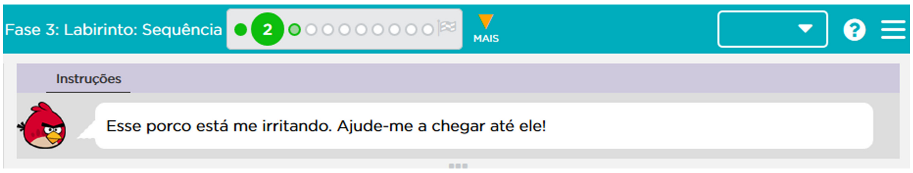
**Fig. 8.** Certificate of completion of course 2 (adapted from [19]).

As part of their implemented reward system, and in addition to the already mentioned elements of motivation in a gamified environment [18], Code.org implemented a *Certificate of Completion*, allowing students to print one with their name, at the end of each course they complete (see Fig. 8), serving as another motivation tool.

## 3.6 Points to Improve

Despite all these success factors, as the students got to work, we found that the platform also has some points to improve, namely regarding the translations into national languages, in our case, Portuguese, both in its Portuguese and Brazilian Portuguese versions.

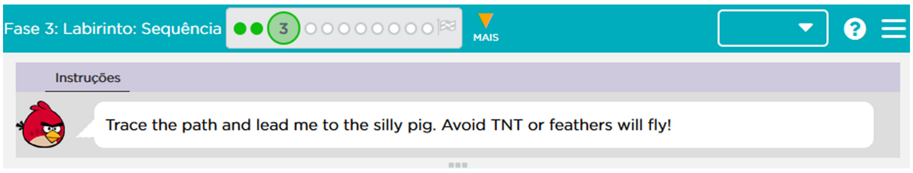
Since it is designed to be used by small children, the platform offers translations into several languages. Despite this, and it the fact that is possible to choose between Portuguese and Brazilian Portuguese, among others, as already mentioned, the translations are not always the most correct. Typically, students in this study used the platform with the Portuguese translations. However, several exercises had their statements written in Brazilian Portuguese, as can be seen in the message shown in Fig. 9.



Esse porco está me irritando. Ajude-me a chegar até ele!

**Fig. 9.** Example of a message in Brazilian Portuguese.

On the other hand, some exercises in the Portuguese version (and also in the Brazilian Portuguese) are displayed in English, although the programming interface is set to Portuguese, as shown in Fig. 10.

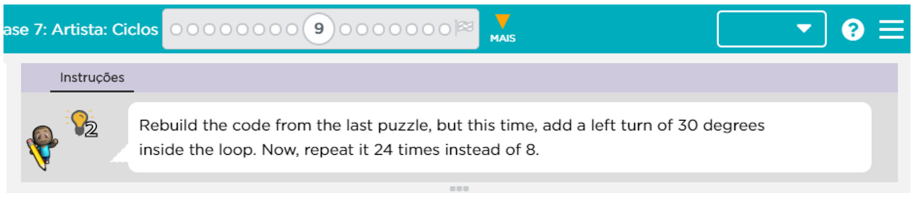


Trace the path and lead me to the silly pig. Avoid TNT or feathers will fly!

**Fig. 10.** Example of a message in English.

In other cases, such as the example shown in Fig. 11, in addition to being written in English, the language level is relatively complex for young, non-English native children. These cases required the teacher to translate the questions for them.

Another language question was the fact that some translations to Portuguese were better understood when read in Brazilian Portuguese than when done in Portuguese from



Rebuild the code from the last puzzle, but this time, add a left turn of 30 degrees inside the loop. Now, repeat it 24 times instead of 8.

**Fig. 11.** Example of a message in English that students had difficulties reading.

Portugal. This forced the students to constantly change the language of the interface in an attempt to better understand certain questions.

In addition to these language issues, students also reported some issues related to the platform freezing in certain browsers and platforms but, given the small number of occurrences, this fact was considered residual and of no greater importance to the results of the study.

## 4 Method

The participants were 198 students from eight 4th grade (9–10 years old) over three school years, divided as follows: 2017/18, 28 students from Class 1 and 28 students from Class 2; 2018/19, 25 students from Class 3, 26 students from Class 4 and 23 students from Class 5; 2019/20, 24 students from Class 6, 23 students from Class 7 and 21 students from Class 8. Of those, 99 were females and 99 were males. It's important to note that the first partial results of this study have already been published in 2020, without important data from school year 2019/20 [19].

The study was conducted over three different realities, concerning the way classes were taught. In the school year 2017/18, the classes were taught in regular classroom sessions in a pedagogical pair regime, with the ICT teacher always assisted by the headteacher of the classes, because the students were very young and it was the first time that most of them worked with computers and platforms such as Code.org. In the 2018/19 school year, it was not possible to maintain this organization, so they were taught in regular classroom sessions but only by the ICT teacher. In the school year of 2019/20, schools closed in mid-march due to the COVID-19 pandemic, so classes were taught using remote education methods, with the ICT teacher being supported by the parents at home. Additionally, students from school years 2017/18 and 2018/19 had previously taken offline coding classes [20] based on the CS unplugged book [21].

Initially, the basic concepts of the Code.org platform were taught to all students. They learned how to grad-and-drop the blocks, how to fit them together, the workspace locations, and interface details such as the action stage and execution buttons. They also learned how to use their credentials to enter the platform, consult their progress, and access the challenges. Then, they worked on a hands-on laboratory with problem-solving exercises [22] that allowed them to develop their computational thinking. The exercises/challenges involved computational concepts like sequences, loops, parallelism, events, conditionals, operators, and data, which would allow students to create their first Flappy game and share it with their relatives.

Data from these activities was collected through the platform's automatic records for statistical processing. For this statistical treatment, only data from the online exercises of the first 16 lessons of Code.org's Computer Science Fundamentals, Course 2 was used as the goal set for the course was for students to create their Flappy Bird game. Also used as instruments for data collection were an online questionnaire for the students (two questionnaires for the students of the school year 2019/20) and the notes of the ICT teacher on how the regular classes and online interaction (with the students of the school year 2019/20) worked.

## 5 Results

The results of the 115 different problems were evaluated for each of the 198 students involved in the study, in a total of 22770 problems. The global results of the study are summarized in Table 1.

**Table 1.** Global results (adapted from [19]).

| | Different problems/total analyzed | Conclusion rate (%) | Conclusion rate, non-optimal solution (%) |
|---|---|---|---|
| Global Results | 115/22770 | 86,5% | 3,99% |

In the first analysis, we found that the percentage of problems solved is 86,5%. Of these, only 3,99% of them did not get the optimal solution to the problem, which is a very positive result.

Examining the results according to the three dimensions of Brennan and Resnick's framework [3], (i) computational concepts, (ii) computational practices and (iii) computational perspectives, we obtained the following results:

(i) **Computational Concepts.**
The number of different problems indicated in Table 2 refers to the number of exercises in which each one of the concepts was covered. Also, the same problem could address more than one concept. Also, it is possible to notice a disproportion between the number of problems covered by each concept but this fact was already expected since this is an introductory course and complex concepts like Events, Parallelism, and Data are only little addressed and only near the end of the course. Sequences were the concept that students had less trouble acquiring, with a completion rate of 89,6%. Of these, only 2,4% of the results were not an optimal solution. On the opposite side, the Events, Parallelism, and Data concepts were those in which students had the most difficulties. Despite this, the completion rate is very positive, at 73,9%, of which only 0,1% did not reach the optimal solution. In global terms, the obtained average completion rate per concept was 78,7%.

**Table 2.** Global test results, grouped by computational concepts.

| Concept | No. of different problems/total analyzed | Completion rate | Completion rate with non-optimal solution |
|---|---|---|---|
| Sequences | 46/9108 | 89,6% | 2,4% |
| Loops | 67/13266 | 86,3% | 5,8% |
| Events | 10/1980 | 73,9% | 0,1% |
| Parallelism | 10/1980 | 73,9% | 0,1% |
| Conditionals | 15/2970 | 77,2% | 3,9% |
| Operators | 25/4950 | 75,9% | 2,4% |
| Data | 10/1980 | 73,9% | 0,1% |

(ii) **Computational Practices.**
All computational practices mentioned by Brennan and Resnick [3] were covered, although, like in the case of computational concepts, due to the type of problems present in the course, not all computational practices were given equal emphasis.

**Table 3.** Global test results, grouped by computational practices.

| Practice | No. of different problems/total analyzed | Completion rate | Completion Rate with non-optimal solution |
|---|---|---|---|
| Being incremental and iterative | 82/16236 | 89,6% | 4,2% |
| Testing and debugging | 23/4554 | 80,6% | 4,6% |
| Reusing and remixing | 67/13266 | 86,3% | 5,8% |
| Abstracting and modularizing | 10/1980 | 73,9% | 0,1% |

It is important to note that the number of problems indicated in Table 3 refers to the number of exercises in which each practice was addressed and that some problems addressed more than one computational practice.
Analyzing the results grouped by computational practices, it is possible to conclude that the practices of Being incremental and iterative were the most addressed throughout the course with 82 different problems. It was in these practices that students showed less difficulty, obtaining a completion rate of 89,6%. The students showed to be comfortable with the practices that involved Reusing and Remixing. In these problems, they achieved a completion rate of 86,3%, with only 5,8% of these not reaching the optimal solution. The obtained average completion rate per practice was 82,6%. The practices that involved Abstracting and modularizing were those in which students obtained lower results. However, it was not possible to determine the real reason for those results. They may be due to actual abstraction

difficulties or simply lack of time to solve the problems given the slow pace of some students.

(iii) **Computational Perspectives.**
Although not objectively measured, the three computational perspectives - Express, Collaborate, and Question - were cross-sectional throughout the process of problem-solving. Although the students had guidelines to solve most of the tasks, in some cases, they also had the freedom to create something new and to personalize something already existing through the inclusion of personal elements in the provided scenarios - Express. Most of their work was done individually, but, typically as soon as a student ended a lesson, they tried to help their most delayed colleagues by doing peer work - Collaborate. Also, curiosity about the processes and the different problem-solving methods led them to Question the technology, to try to solve problems with different levels of abstraction, and even to suggest some improvements that could be made in the existing games or challenges.

## 5.1 Flappy Bird Challenge

Programming a Flappy bird game was the final objective considered for this study.

**Fig. 12.** Generic scenario used in the Flappy Lesson (adapted from [19]).

Starting with a generic scenario represented in Fig. 12, students created their own personalized Flappy Bird games (see Fig. 13).

The results for the final task on Lesson 16 were very positive, with a 69,7% completion rate of the challenge.

They were also encouraged to share their games with family and friends, via email, by sending a link that could be executed on all platforms where code.org works. In statistical terms, in the 1463 problems solved by the students in Lesson 16, only 1 of those was not solved with the optimal solution. However, regarding the data analysis, this fact might not be considered of much relevance, since, in the case of a work of creation, personalization, and sharing, almost all the solutions presented by the students could be considered optimal.

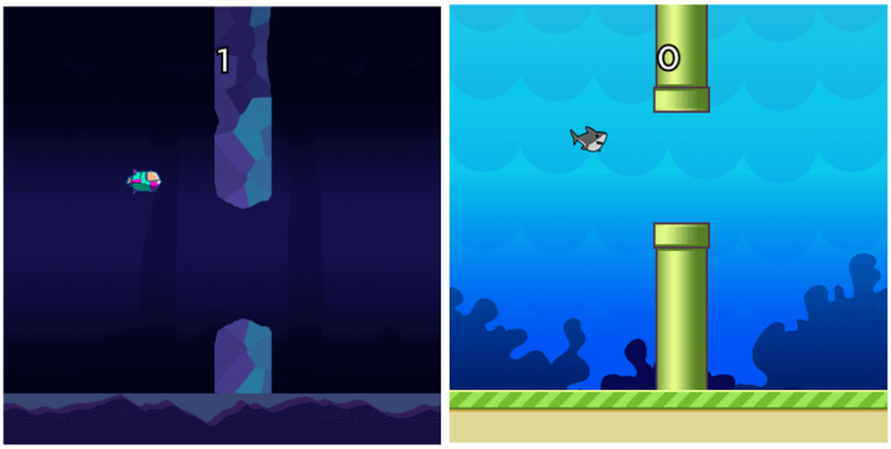


**Fig. 13.** Student-customized flappy bird examples.

## 5.2 Working @ Home

The three classes from the year 2019/20 had a completely abnormal year. As we are aware, the Covi-19 pandemic, locked us all at home and, in the particular case of Classes 6, 7 and 8, all classes since March, 13th 2020 were taught remotely via Microsoft teams Meetings, for Tasks, chatting and synchronous learning activities, and via YouTube videos, for asynchronous activities and tasks. This fact makes it important to analyze the results of these 3 classes in more detail. As previously referred, data from task completion of Classes 1–5 has already been published and analyzed thoroughly [19].

Using data from Microsoft Teams Insights, it's possible to summarize the interactions of each one of the classes with the teacher during this period (see Table 4).

**Table 4.** Online interaction between students and teacher.

| Interaction | Class 6 | Class 7 | Class 8 |
|---|---|---|---|
| Posts | 201 | 210 | 146 |
| Answers | 621 | 163 | 209 |
| Reactions | 649 | 953 | 839 |
| Answers per post (Avg) | 3,1 | 0,8 | 1,4 |
| Reactions per post (Avg) | 3,2 | 4,5 | 5,7 |
| Tasks delivered on time | 79% | 88% | 73% |

Although the contents covered, the main posts from the teacher and the tasks to perform were the same for the three classes, the interaction levels are quite different from class to class. It is possible to observe that, for example, Class 6 has more than doubled the answers per post of the other two classes. However, in that same class, only

79% of the Tasks were delivered within the specified time whereas, in Class 7, the one with the lowest interaction levels, the tasks delivered on time reached 88%.

Specifically, on Code.org, the data collected from the platform allowed us to build a table of results per Class (Table 5).

**Table 5.** Test results of classes 6–8, grouped by computational concepts (in percentage).

| Concept | Class 6 | Class 7 | Class 8 | Average |
|---|---|---|---|---|
| Sequences | 96,7 | 81,4 | 93,8 | 90,6 |
| Loops | 93,9 | 73,1 | 90,8 | 85,9 |
| Events | 82,1 | 77,8 | 86,7 | 82,2 |
| Parallelism | 82,1 | 77,8 | 86,7 | 82,2 |
| Conditionals | 88,3 | 78,3 | 90,5 | 85,7 |
| Operators | 85,8 | 78,1 | 89,0 | 84,3 |
| Data | 82,1 | 77,8 | 86,7 | 82,2 |
| Average per class | 87,3 | 77,8 | 89,1 | |

Observing the averages per Computational Thinking Concept, it is possible to verify that the minimum value obtained is 82,2% which represents a very high value, close to the overall test results of 86,5% (see Table 1) and above the average global results per concept of 78,7%, leading us to believe that the fact that these particular Classes were working at home did not interfere with the results of the test.

Visually comparing the three Classes (see Fig. 14), it is possible to see that Class 7's results were always below the results from the other two Classes.

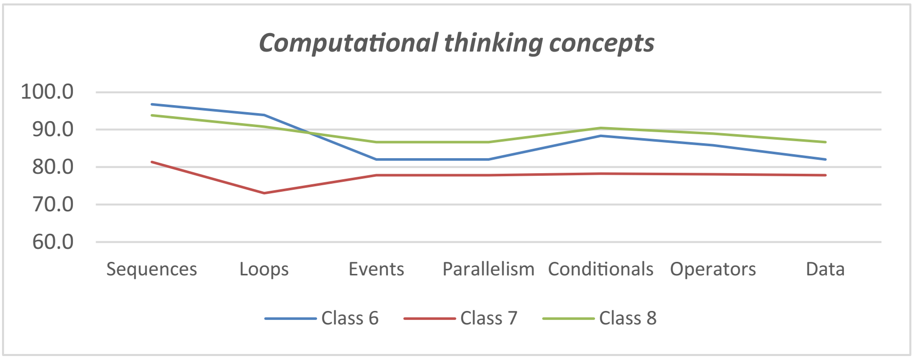


**Fig. 14.** Test results of classes 6–8, grouped by computational concepts (in percentage).

It was also possible to obtain data to analyze the computational practices by Classes. In these results (see Table 6), it is possible to observe that the minimum average value per class was 75,1%, a very positive result. Also, the minimum average value per practice was 83,9%, a value higher than the average of the entire study of 82,6% (see Table 3).

**Table 6.** Test results of classes 6–8, grouped by computational practices (in percentage).

| Practice | Class 6 | Class 7 | Class 8 | Average |
|---|---|---|---|---|
| Being incremental and iterative | 94,7 | 81,7 | 92,5 | 89,6 |
| Testing and debugging | 93,5 | 67,7 | 90,7 | 83,9 |
| Reusing and remixing | 93,9 | 73,1 | 90,8 | 85,9 |
| Abstracting and modularizing | 82,1 | 77,8 | 86,7 | 86,7 |
| Average per class | 91,0 | 75,1 | 90,2 | |

Observing the graphic of the results per practice (see Fig. 15), it is also possible to verify that, although very positive, in this dimension it was also Class 7 that obtained the lower results of the three classes. It is possible to observe that, despite some fluctuations in the values, in the case of the practice of Abstracting and modularizing, the values for all three classes are very close to each other. The greatest disparity in values occurred in the Testing and debugging practice of Class 7. Data shows that the real reason for this lower result is not that the problems were badly resolved, but that they were not resolved at all. According to the data collected from the platform, the students from Class 7 did not resolve 32% of the exercises related to that practice, hence the poor results. However, it was not possible to identify the reason why students in Class 7 did not try to solve this particular set of problems.

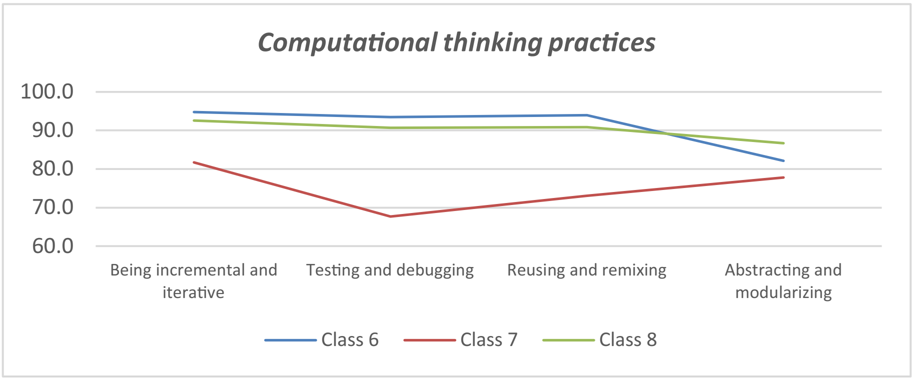


**Fig. 15.** Test results of classes 6–8, grouped by computational practices (in percentage).

Although not objectively related in this study, the results of the 2019/20 school year seem to point that lower levels of interaction on the communication platforms used by teachers lead to lower results as, in this case, could be observed by the data related to Class 7.

### 5.3 Daily Classes

As previously mentioned, this study is divided into three different learning realities: regular classroom sessions, in a pedagogical pair regime; regular classroom sessions but with only the ICT teacher; and children working at their homes, with the help of the ICT teacher, in synchronous and asynchronous activities, and their families for closer support.

The overall results of the study were excellent but, given these pedagogical changes, we found it relevant to analyze the partial results per school year.

Observing Table 7, it is possible to verify the evolution of the average results of classes, by Computational Thinking Concepts, in the three different learning realities.

**Table 7.** Results by computational thinking concepts (in percentage).

| Computational thinking concept | Average (Classes 1–2) | Average (Classes 3–5) | Average (Classes 6–8) |
|---|---|---|---|
| Sequences | 96,5 | 83,6 | 90,6 |
| Loops | 96,2 | 79,4 | 85,9 |
| Events | 89,3 | 55,6 | 82,2 |
| Parallelism | 89,3 | 55,6 | 82,2 |
| Conditionals | 89,5 | 60,8 | 85,7 |
| Operators | 89,4 | 58,7 | 84,3 |
| Data | 89,3 | 55,6 | 82,2 |
| Average per school year | 91,4 | 64,2 | 84,7 |

Reading the radar chart in Fig. 16, representing the same results, it is possible to better understand these values. It is possible to notice that, compared to Classes 1–2 and Classes 6–8, the lines referring to Classes 3–5 are always closer to the center of the chart, moving away from the optimum results.

Also, when analyzing the results by Computational thinking practices, it is possible to verify very different levels from one school year to another (see Table 8).

On the radar chart in Fig. 17, representing the same results by practice, it is possible to notice that, once again, compared to the other results, the lines referring to Classes 3–5 are always closer to the center of the chart, moving away from the optimum results.

Objectively, and comparing the results by Computational thinking concepts and practices, it's possible to verify that Classes 3–5, although with positive results, were the ones whose results were further away from the optimum objectives.

Also, when analyzing the completion results of the final task of Lesson 16, in the school year of 2017/18, (Classes 1–2), only 14,28% of the students did not complete the last challenge of the programmed course. On the other hand, in the school year of 2018/19, (Classes 3–5), this number rose to 55,4%. In the school year of 2019/20, (Classes 6–8) the value decreased again to 19,1%.

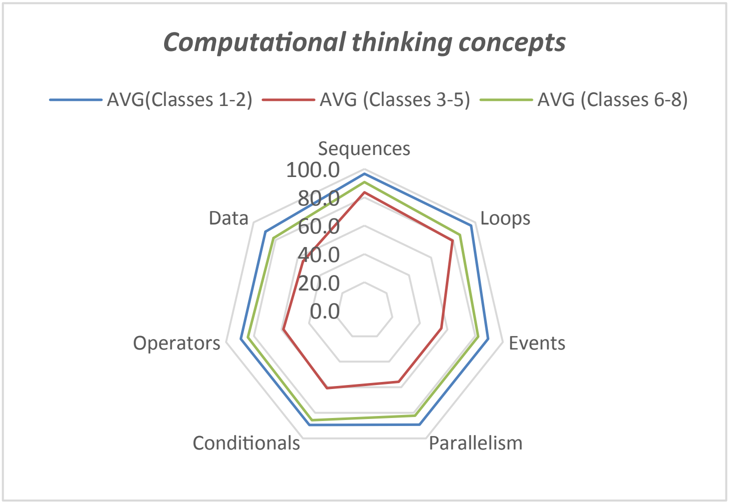


**Fig. 16.** Results by computational thinking concept (adapted from [19]).

**Table 8.** Results by computational thinking practices (in percentage).

| Computational thinking practice | Average (Classes 1–2) | Average (Classes 3–5) | Average (Classes 6–8) |
|---|---|---|---|
| Being incremental and iterative | 96,9 | 84,4 | 89,6 |
| Testing and debugging | 93,2 | 68,6 | 83,9 |
| Reusing and remixing | 96,2 | 79,4 | 85,9 |
| Abstracting and modularizing | 89,3 | 55,6 | 82,2 |
| Average per school year | 93,9 | 72,0 | 85,4 |

Although the students were not the same, pedagogically speaking, the main variable that changed throughout the study was the fact that the students had more support in some school years than in others. In all three school years, whether in regular classroom sessions or remotely, the doubts that most students had were solved with explanations of specific problem-solving methods to the entire class at the same time. Individual doubts were explained to each of the students in their place in the classroom or via chat/video meetings when working remotely. Yet, when working in a pedagogical pair regime, the fact that there were always two teachers in the classroom allowed a faster response to problems that could arise with both the computers and the platform which left more time

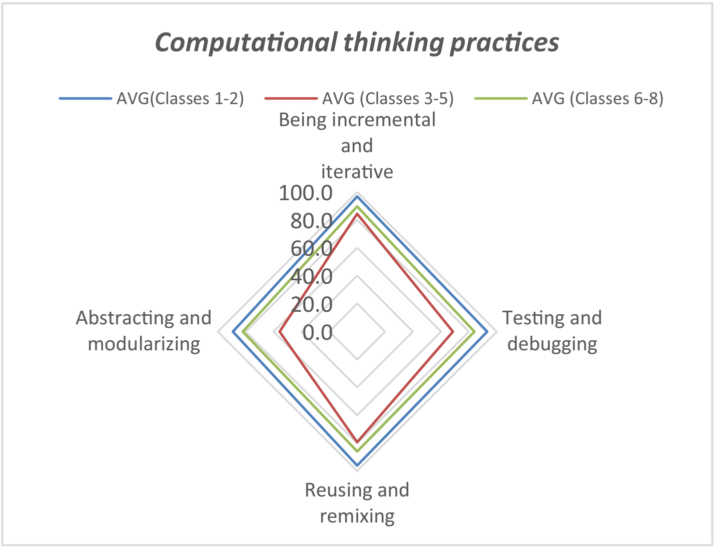


**Fig. 17.** Results by computational practices (adapted from [19]).

for students to focus on problem-solving, thus obtaining better results. When working at home, that same support was provided by parents with the help of the ICT teacher, via Microsoft Teams. This kind of extra support was not possible in 2018/19 with Classes 3–5.

Despite its effects were not objectively measured, one should not underestimate the importance of that extra support when working in this type of subject given the number of students in the classroom and their age.

## 5.4 Evaluation by Students

To allow students to evaluate their work, we used an adapted questionnaire based on previous studies [23]. This questionnaire was made available to students in an anonymous online form, with some free text questions, allowing students to express their feelings about the work they developed. This questionnaire contained the following questions:

- **Question 1.** Did you have difficulties with the process of solving the problems? If you answered Yes or Some, say what were the difficulties.
- **Question 2.** Was it possible to solve the various problems in several different ways? Did you try to do it?
- **Question 3.** Do you think the Code.org platform is easy to use?
- **Question 4.** Did you have any problems with the platform?
- **Question 5.** What was your favorite Lesson of Code.org course 2?
- **Question 6.** And what Lesson did you like least in Code.org course 2?
- **Question 7.** What do you think were the benefits (what did you learn) of coding using Code.org?
- **Question 8.** Do you like coding?

- **Question 9.** What would you like to learn more about coding?
- **Question 10.** Would you like to improve your knowledge?

In each school year, after the end of classes, students involved in this study were invited to answer the online questionnaire. We collected 146 valid responses and canceled 4 due to repeated submission and inconsistent data in the responses, such as "I had difficulties" but at the same time "I had no problems".

Examining the obtained answers, it was possible to retain the following conclusions:

When asked about **(1) difficulties in the problem-solving process**, 41 students reported that they had no problem in the problem-solving process. Only 6 students considered that they had problems. 99 students reported that they had some minor problems with this process.

However, when asked to refer the actual difficulties, we observed that only 4 of the 146 students reported real problems with the process and the platform - "I had difficulties in executing it"; "Had to reset to drag the blocks"; "At the beginning, I had some difficulties dragging the blocks". The remaining responses referred to problems that aren't related to the resolution process, but rather to some issues of previous knowledge, namely, notions of laterality ["to know if I needed to turn left or right", "I had difficulties in those problems that asked to turn right or turn left"], mathematical questions ["I had difficulties calculating the angles", "I had difficulty finding the angles and number of pixels needed", "calculating the pixels and degrees."], or even problems interpreting or solving the problems ["Sometimes it was more difficult to pass the level because I didn't quite understand how it was supposed to be done", "Some exercises were harder than others"]. Interpretation issues could be related to the previously mentioned fact concerning some of the incomplete or partially wrong translations to the local language.

When asked about **if (2) it was possible to solve the proposed problems in several different ways**, 87,7% of the students reported that they tried to do it, and 90,4% of those reported that they were able to do it in different ways. This fact was proved by the analysis of the results provided by the platform, where 3,99% of the problems were solved with non-optimal solutions, compared to the total number of exercises solved. This shows that despite the students realized that it would be possible to solve the problems differently, as they appeared signaled in a different color, not all of them were able to do so, to obtain the optimal solution.

Regarding the **(3) ease of use of the Code.org platform**, 88,4% of the students who answered the survey found the platform to be easy to use ["Code.org is very easy to use and I had no problem with the platform. It is very easy, it is practical because it is suitable for all electronic devices and to enter it you just have to enter the password and enter, and you have a solution if you forget your password", "Yes I think it is easy to use Code.org platform"]. Interestingly, some of the students who considered that the platform is not easy to use, consider that they had no problems in its use because when asked if they had had (4) problems with the platform, 80,8% of the students reported that they did not have problems.

When referring to their **(5) favorite Lesson of the course**, the survey obtained 131 valid answers. 15 of the answers were considered null because students referred to Lessons not considered in the study or to other courses on the code.org platform. Considering the valid answers, the students elected first place, Lesson 16, FLAPPY, the last considered for this study, with 31 votes, followed by Lesson 3, MAZE: SEQUENCE, the first they worked on, with 23 votes.

Lessons 7: ARTIST: LOOPS, 8: ARTIST: SEQUENCE LOOPS and 13: BEE: LOOPS, were also the ones that most pleased the students, with 12, 12 and 11 votes respectively. Although many students chose as favorite the Lessons that focused on simpler concepts, 64,1% of the students who answered the survey chose as a favorite Lesson one of the ones that already involved more complex computing concepts such as Loops, Events, Parallelism, Conditionals, Operators, and Data. This result is in line with the general results of the study, and it demonstrates that students appreciated the complexity brought to the problems by the application of computational concepts with a higher level of difficulty. They enjoyed using their recently developed computational thinking.

On the other hand, when they referred to the **(6) Lesson that they liked least**, students who had preferred lessons with more elaborated concepts tended to like less the first lessons of the course with only basic concepts and the opposite was also noted. Students who preferred the lessons with simpler concepts tended to dislike the more complex lessons. Also, 10,3% of the students who answered the questionnaire said that they could not enumerate one, because they loved all the Lessons.

When asked about what they **(7) learned through the Code.org platform**, the students gave the most diverse answers. Table 9 highlights the main categories that emerged from the content analysis after a first floating reading [24].

**Table 9.** Excerpt from the free-text responses about what students learned while working in Code.org.

| Category | Evidence (examples) | Frequency |
|---|---|---|
| Learn to work with computers | "Now I can easily use the computer and keyboard"; "Learn how to work with computers"; "They helped me to be more comfortable working with the computer"; "I learned to work better on the computer" | 21 |
| Programming | "I learned what gives rise to programs and that through Code we can simplify some programs."; I learned "how to program games."; "The benefits were learning to program."; "I learned to program very well."; "I learned how to create some things" | 20 |
| Autonomy | "I learned to be more patient and not ask for help right away."; "I learned to solve problems on my own"; "I with Code.org learned that we can never give up."; "I learned to work alone."; "I learned that one should never give up and that we have to be patient" | 17 |

(*continued*)

**Table 9.** (*continued*)

| Category | Evidence (examples) | Frequency |
|---|---|---|
| Think faster and better to solve problems | "At Code.org I was able to: (…) think faster to execute problems"; "I learned to think twice before doing it"; "I learned to solve problems"; "By coding at Code.org: I learned to think better (…)."; "I learned to use the brain more easily."; "I was amazed to do things in different ways and with creativity"; "I learned how to solve problems" | 17 |
| Learn Math | "I was also able to look at a certain angle and identify it"; "I learned a lot of math."; "I learned to use angles"; "To take it easy and learn math" | 14 |
| Play | "I learned to play Code.org games."; "play and at the same time learn."; "I learned to play games on the site."; I learned "to work, to play, and to have fun" | 11 |
| Problem Solving Skills | "I was amazed to solve problems in different ways and with creativity"; "I learned how to solve problems on my own" | 6 |
| Notions of Laterality | "I learned right and left"; "I used to swap left with right" | 5 |

Analyzing the collected data, it was possible to verify that, in general, students learned that they "should never give up" and "must-have patience" to solve the problems, in a clear reference to the development of autonomy and resilience inherent to the problem-solving process. They also learned "to think better" and "to solve problems in different ways", by themselves, and to do things "in different ways", with "creativity", facts also related to the development of problem-solving skills.

In terms of acquired knowledge, students generally refer to "better understanding the computer Codes" and to "be more comfortable working with the computer". They also use the term "Programming" and phrases like "I learned to do logical reasoning" or "I learned how to develop programs" a significant number of times. Since it was important not only to learn to program but also to learn while programming [25], it is worth mentioning the fact that some students consider that they have perfected their notions of laterality and Mathematics while solving exercises that involved mathematical concepts such as angles and drawing with pixels.

Also important is that the students were learning while having fun since, as they said, it was 'play and learn at the same time'.

To the question **(8) Do you like to program?** we obtained 135 positive responses (92,5%) confirming that students were enjoying programming, at least in the visual form they have known so far. We also obtained 8 responses from students (5,5%) who said they like to program, but only certain types of problems. As for negative responses, we obtained only three.

Regarding **(9) improving/deepening knowledge**, students tended to answer that they would like to learn more about building and coding robots and coding games. Content analysis was also performed to these free-text answers [24] and the two main categories found were replicated in Table 10.

**Table 10.** Excerpt of categories for the content analysis of the answers about deepening knowledge.

| Category | Evidence (examples) | Frequency |
|---|---|---|
| Coding robots | "In terms of coding I liked to control robots and many more things about coding and would like to improve my knowledge."; "I would like to build robots"; "I would like to know if the robots are programmed the same way we program the Code.org characters and yes I would like to improve my knowledge."; "I would like to assemble robots" | 36 |
| Coding games | "I liked to learn how to make my own games"; "What I liked to learn most in programming, was programming games"; "I would like to make other games"; "Program games like this."; "Make games. Yes, I would like to know more."; "I would like to invent my games"; "I would like to know how to program my own game" | 22 |

Regarding **(10) learning more/improving knowledge**, we obtained 98,6% of affirmative answers, with phrases similar to "I would like to improve my knowledge, as I still know very little about the world of programming", "I would love to learn more" or "Yes, I would like to". As for negative responses, we obtained only one - from the same student who said he does not like to program - and another student stated that he would like to learn "more or less".

Additionally, to evaluate the work developed by the students at home, during the distance education period, the students from Classes 6, 7, and 8 (school year of 2019/20) were invited to answer a second online questionnaire in which we collected 70 valid responses.

This questionnaire was available in an anonymous online form, with some closed answer, multiple-choice and Likert scale questions and contained the following questions:

- **Question 1.** Do you think it was more difficult to work at home with their own devices than with computers from the classroom?
- **Question 2.** Regarding the functioning of the classes… Do you think that the one-week period for the tasks was sufficient?
- **Question 3.** Did you enjoy having synchronous classes?
- **Question 4.** Would you like to have had more synchronous classes? Answer this question only if you answered Yes to the previous question.

- **Question 5.** How much time did you dedicate to solving Code.org problems? If you don't remember, ask the family at home… they may have a better idea than you…
- **Question 6.** What devices did you use to work with?
- **Question 7.** How do you rate the help of your family members to USE THE TECHNOLOGY in this Distance education phase?
- **Question 8.** How do you rate your family members' help to SOLVE Code.org PROBLEMS?

Analyzing the obtained answers, it was possible to retain the following conclusions:

Regarding the fact that the students spent a great part of the school year at home and **comparing the (1) difficulty to work at home with their own electronic devices, with the work at school**, the opinions were well divided: 50% of the students referred that it was harder to work at home than at school whereas the other half of the students stated that it was not more difficult to work at home.

To cope with the distance between teacher and students, and using the fact that the Course 2 of Code.org is already divided in Lessons, students were given minimum objectives for each week, in the form of Microsoft Teams Tasks. **Question (2)** intended to assess whether they thought that the **one-week period for the tasks was sufficient**. This question was in the form of a 5-points Likert scale [26] question where 1 meant "No, the time wasn't enough" and 5 meant "Yes, I made it all very quickly". Examining the responses, we obtained a mode of 5, considered a measure of central tendency [27], which points to the fact that students were very satisfied with the time for each one of the tasks.

Although students were comfortable with the work developed at home, mostly in the form of asynchronous tasks, they also enjoyed the synchronous classes. Based on **question (3)** answers, 67 of the 70 students (95,7%) that answered the questionnaire, said they" enjoyed the synchronous classes and doing the tasks at the same time as the other colleagues". Also, in this matter, 77,1% of the students referred that they would have liked to have more synchronous classes **(Question 4).** By comparing these two values where 95,7% liked the synchronous classes but only 77,1% would like to have more, we can perceive that despite liking the synchronous instructional mode in which they contacted via videoconference with the teacher and their colleagues, some of the students were not very comfortable with it and would prefer to do the tasks on their own, at their own pace.

**Question 5** was used to try to assess the **time that the students spent working at home**, to compare it to the other classes from previous school years. 37 of the 70 students (52,9%) that answered the questionnaire stated that they spent less than 1 h per week working on Code.org, 32 of the students (45,7%) spent between 1 and 2 h per week and only 1 of the students (1,4%) spend more than 2 h per week. Comparing these figures with previous school years is somehow difficult. Typically, students in Classes 1 to 5, worked on Code.org for 1 h when they attended the regular classroom sessions and were instructed to work on Code.org only in those sessions. Of course, at this time, it's impossible to know if they worked at home, after class. However, with the results obtained, a weighted average of 1,2 h per week, we believe that the results of the school year 2019/20 are comparable with the ones from previous years.

One of the computer skills that might have been developed in the case of these students, confined at home, is the ability to use different electronic devices to work. Typically, when they start ICT classes at school, most of them have never or rarely worked with personal computers because at home, they work only with smartphones and tablets. It was surprising to analyze the results of **Question 6**, to find out that only a small number of students used Tablets and Smartphones to perform their tasks. The results are summarized in the following Table 11 .

**Table 11.** Devices used to perform tasks at home.

| Device used | Number of references |
|---|---|
| Laptop/Notebook computer | 44 |
| Tablet | 20 |
| Desktop computer | 17 |
| Smartphone | 6 |

Is was also interesting to observe that 15 of the 70 students used more than one device to perform their tasks and attend synchronous classes. In some cases, a simple "computer setup" was created at home, with the help of their parents, so that children could perform their tasks in the computer but used their smartphones or tablets to attend synchronous classes and share video and audio at the same time, without interrupting their work.

Regarding **Question 7, we used it to evaluate the technological aptitude level of the students**. As it was the first time that all of them were involved in this kind of learning activities, it is important to know if they had difficulties in the adaptation to the new methods. The question was in the form of a 5-points Likert scale [26] question where 1 meant "I never needed help. I already knew or learned to do everything myself." and 5 meant "I needed a lot of help. It was all new and different than usual.". Examining the responses, we obtained a central tendency mode of 3 [27] which points to the fact that a great number of the students were already "computer fluent" and had some, but not many problems adapting to the tools they needed to use while accessing the class contents. This fact might be explained by the fact that they already had 5 months of regular classroom sessions, they had already developed the necessary skills to work with these tools.

The last question of this questionnaire, **Question 8, intended to evaluate the level of help that students had from their families while working at home**. This question was also in the form of a 5-points Likert scale [26] question where 1 meant "I did everything myself" and 5 meant "I had help with many of the problems". Examining the obtained responses, we obtained a mode of 2 [27] which points to the fact that most students have developed a high level of autonomy and problem-solving skills that don't require adult help or supervision at all times. Also, it's important to refer that 13 of the 70 students that answered the questionnaire mentioned that they did everything on their own. Only 5 of the students considered that they had a lot of help from their families. This fact may also have been determined by the fact that when they started working at home, the students already had 3 regular classroom sessions with their ICT teacher, in which the basics of Code.org and the problem-solving methods have been explained, in a similar way to what happened in previous school years.

## 6 Conclusions

Children live in a technology environment, marked by access to an abundance of information, rapid changes in technology tools, and the ability to collaborate and make individual contributions on an unprecedented scale. Capable children of the 21st century must be able to exhibit a range of functional and critical thinking skills related to information, media, and technology. Learning and innovation skills increasingly are being recognized as those that separate students who are prepared for more complex life in the 21st century, and those who are not. A focus on creativity and computational thinking are important to prepare students for the future.

The study described in this article used the code.org platform to analyze the development of computational thinking with Elementary school students. To assess the development of computational thinking, we use a framework that identifies computational concepts, practices, and perspectives. As all tasks proposed to students included the search for the solution of a problem, concepts, practices, and computational perspectives, it can be said that computational thinking was promoted. Students attained very positive results, while training problem-solving skills, building and retaining knowledge better.

Code.org is a handy tool to use in introductory coding classes. The fact that students have to solve different types of unfamiliar problems in creative and innovative ways makes them ask meaningful questions that clarify various points of view and lead to better solutions. The use of gamification strategies like narratives, trophies, and instant feedback, works as an engagement factor for students. Also, the fact that some exercises have clues that help children understand them, the possibility of partially solving the exercises and being able to return to complete them at a later time, makes code.org a very flexible and appropriate tool for the age group under study. It was also possible to perceive the involvement of the students with some of the characters used in the challenges, as they already knew them from the games they played. Also significant is the fact that they were learning computer science concepts while having fun.

There are, however, other conclusions to be drawn from the experience, namely concerning the importance of the support that students should have. When working in standard classrooms, the organization of this type of class should be favored in terms of pedagogical pairs or smaller groups of students for better results. The proposed tasks require very close monitoring, which is extremely difficult to be performed by a single teacher in a classroom. Doing it in remote learning, due to the COVID-19 pandemic, increased the difficulties, which may have somehow affected the final results.

However, this study made it possible to obtain positive conclusions and create working methods, both for students and teachers, which could be used in the future.

**Acknowledgments.** This work was partially financed by the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia, through national funds, and co-funded by the FEDER, where applicable.

This work was partially funded by CIEd – Research Centre on Education, project UID/CED/01661/2019, Institute of Education, University of Minho, through national funds of FCT/MCTES-PT.

We would like to thank the Colégio Paulo VI (Gondomar, Portugal) the authorization to carry out this study on its premises, and students of the 4th grade of the school years of 2017/18, 2018/19 and 2019/20 by their collaboration.

## References

1. Wing, J.M.: Computational thinking. Commun. ACM **49**, 33–35 (2006)
2. Resnick, M.: Point of view: reviving Papert's dream. Educ. Technol. **52**(4), 42–46 (2012)
3. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Annual American Educational Research Association Meeting, Vancouver, BC, Canada, pp. 1–25 (2012). https://doi.org/10.1.1.296.6602
4. Puiu, T.: Your smartphone is millions of times more powerful than all of NASA's combined computing in 1969. https://www.zmescience.com/research/technology/smartphone-power-compared-to-apollo-432/. Accessed 21 Feb 2019
5. Popular Mechanics Website. https://www.popularmechanics.com/technology/a22007431/smallest-computer-world-smaller-than-grain-rice/. Accessed 01 Aug 2020
6. BizzCommunity Website. https://www.bizcommunity.com/Article/196/423/195991.html. Accessed 01 Aug 2020
7. Partnership For 21ST Century Skills: Framework for 21st Century Learning (2009). http://www.p21.org/storage/documents/docs/P21_framework_0816.pdf
8. Resnick, M., et al.: Scratch: programming for all. Commun. ACM **52**, 60–67 (2009). https://doi.org/10.1145/1592761.1592779
9. Wing, J.M.: Computational thinking (2007). https://www.cs.cmu.edu/afs/cs/usr/wing/www/Computational_Thinking.pdf. Accessed 01 May 2019
10. Wing, J.: Computational thinking's influence on research and education for all. Ital. J. Educ. Technol. **25**(2), 1–12 (2017). https://doi.org/10.17471/2499-4324/922
11. Cuny, J., Snyder, L., Wing, J.M.: Demystifying computational thinking for non-computer scientists (2010). http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf
12. Brennan, K., Chung, M., Hawson, J.: Scratch curriculum guide draft. Nature **341**(6241), 73 (2011)
13. Jonassen, D.: Learning to Solve Problems. A Handbook for Designing Problem-Solving Learning Environments. Routledge, New York (2011)
14. Papert, S.: The Children Machine. BasicBooks, New York (1993)
15. Echeverría, M., Pozo, J.: Aprender a resolver problemas e resolver problemas para aprender. In: Pozo, J. (ed.) A Solução de Problemas: Aprender a Resolver, Resolver Para Aprender. Artmed, Porto Alegre (1998)
16. Code.org Website. https://Code.org/about. Accessed 05 Apr 2019
17. Kapp, K.M., Blair, L., Mesch, R.: The Gamification of Learning and Instruction Fieldbook. Wiley, San Francisco (2012)
18. Barradas, R., Lencastre, J.A.: Gamification e game-based learning: estratégias eficazes para promover a competitividade positiva nos processos de ensino e de aprendizagem. In: Revista Investigar em Educação (Issue Mundo digital e Educação), pp. 11–37. Sociedade Portuguesa de Ciências da Educação, Porto (2017)

19. Barradas, R., Lencastre, J.A., Soares, S., Valente, A.: Developing computational thinking in early ages: a review of the code.org platform. In: Chad Lane, H., Zvacek, S., Uhomoibhi, J. (eds.) Proceedings of the 12th International Conference on Computer Supported Education (CSEDU2020), vol. 2, pp. 157–168. SCITEPRESS – Science and Technology Publications, Prague (2020)
20. CS Education Research Group Website. http://csunplugged.org. Accessed 7 Sept 2015
21. Bell, T., Witten, I.H., Fellows, M.: CS unplugged. University of Canterbury, NZ (2015). http://csunplugged.org/wp-content/uploads/2015/03/CSUnplugged_OS_2015_v3.1.pdf
22. Jonassen, D.: Learning to Solve Problems - An Instructional Design Guide. Pfeiffer, São Francisco (2004)
23. Kalelioğlu, F.: A new way of teaching programming skills to K-12 students: code.org. Comput. Hum. Behav. **52**, 200–210 (2015)
24. Bardin, L.: Análise de conteúdo, p. 70. Edições, Lisboa (1979)
25. Resnick, M.: Learn to Code, Code to Learn (2013). https://www.edsurge.com/news/2013-05-08-learn-to-Code-Code-to-learn. Accessed 07 Feb 2019
26. Likert, R.: A technique for the measurement of attitudes. Arch. Psychol. **140**, 1–55 (1932)
27. Jamieson, S.: Likert scales: how to (ab) use them. Med. Educ. **38**(12), 1217–1218 (2004)