Computational thinking in the service of mathematics education: The case of animating programmable 3D models in MaLT2

Pensamento computacional ao serviço da educação matemática: O caso da animação de modelos 3D programáveis em MaLT2

Marianthi Grizioti
Educational Technology Lab, Department of Educational Studies, National and Kapodistrian University of Athens
Greece
mgriziot@eds.uoa.gr

Chronis Kynigos
Educational Technology Lab, Department of Educational Studies, National and Kapodistrian University of Athens & Department of Computer Science and Media Technology, Linnaeus University
Greece, Sweden
kynigos@eds.uoa.gr

Marcelo Milrad
Department of Computer Science and Media Technology, Linnaeus University
Sweden
marcelo.milrad@lnu.se

Abstract. The scientific community has recognized that CT goes beyond computer science and should be meaningfully integrated across the curriculum. However, regarding math education most approaches, studies and tools are focusing on programming concepts in math-covered tasks, leaving high-level mathematical processes and competences in the background. Moreover, most research focuses on primary education, leaving gaps in understanding CT's role in secondary mathematics. This paper proposes an approach that leverages programming to enhance Computational Thinking in the service of mathematical learning rather than the other way around. Through an empirical study with secondary students using MaLT2, an online 3D Turtle Geometry modeler, we investigate how CT can support mathematization, offering insights into the intersection of CT and mathematical reasoning. We explore different computational solutions to math problems, given by secondary students, discussing whether they are following a math-oriented or CS-oriented approach. The results showed that CT practices could be used in different ways for solving the same problem

Quadrante: Revista de Investigação em Educação Matemática 33(2) 110-129
https://doi.org/10.48489/quadrante.37379
Received: September 2024 / Accepted: December 2024

computationally and depending on the approach they could promote either mathematical meaning making or computer science meaning making.
Keywords: computational thinking; math education; constructionism; secondary education; 3D modeler; logo-based programming; computational practices.

Resumo. A comunidade científica reconheceu que o Pensamento Computacional (PC) vai além da ciência da computação e deve ser integrado de forma significativa em todo o currículo. No entanto, no que diz respeito à educação matemática, a maioria das abordagens, estudos e ferramentas estão focadas em conceitos de programação em tarefas matemáticas, deixando processos matemáticos de alto nível e competências em segundo plano. Além disso, a maior parte das pesquisas concentra-se nos primeiros anos, deixando lacunas na compreensão do papel do PC na matemática do ensino secundário. Este artigo propõe uma abordagem que utiliza a programação para desenvolver o Pensamento Computacional ao serviço da aprendizagem matemática, em vez do contrário. Por meio de um estudo empírico com estudantes do ensino secundário utilizando o MaLT2, um modelador online de Geometria da Tartaruga em 3D, investigamos como o PC pode apoiar a matematização, oferecendo insights sobre a interseção entre o PC e o raciocínio matemático. Exploramos diferentes soluções computacionais para problemas matemáticos, dadas por estudantes do ensino secundário, discutindo se elas seguem uma abordagem orientada para a matemática ou para a ciência da computação. Os resultados mostraram que as práticas de PC podem ser usadas de diferentes maneiras para resolver o mesmo problema computacionalmente e, dependendo da abordagem, podem promover a construção de significado matemático ou de significado em ciência da computação.
Palavras-chave: pensamento computacional; educação matemática; construcionismo; ensino secundário; modelador 3D; programação baseada em logo; práticas computacionais.

Introduction

Since around two decades before writing this paper, the development of student's Computational Thinking (CT) had been increasingly considered and promoted as a transdisciplinary 21st century skill, connected to agendas and policies mainly addressing the need for digital literacy and citizenship. In recent years, the inclusion of CT in mathematics curricula has appeared as policy in some cases, mostly as an adjunct curriculum element, with little clarity on what mainstream mathematics education has to gain from this.

Paradoxically, the idea of thinking computationally to solve mathematical problems is not that new. It originated in the context of programming to generate mathematical meaning cultivating deep experiential mathematical activity for learners. Seymour Papert's disruptive approach to learner engagement with mathematical thinking through constructionist programming activity was firstly elaborated as early as in the 60's (Papert, 1972). Papert described the mental processes that students develop when they do mathematics with a List processing (LISP)-derived programming language he called 'LOGO' (from the Greek word 'logismos') using the term 'algorithmic thinking' (Papert, 1980), later elaborating this kind of activity with his group at the MIT Media Lab under the term

'Constructionism' (Papert & Harel, 1991; Kynigos, 2015). So, how was the connection between CT and mathematical thinking lost over the years?

In the late 80's, 90's and early 00's, attention waned from uses of expressive digital media for programming and was rather diverted to uses of the awe-inspiring advent of multi-media, the Internet and the web. Programmability re-emerged as an element of something broader, termed 'computational thinking', this time in a much more generalized sense, i.e. that of CT being an integral part of computational literacy and citizenship in the digital era (diSessa, 2001; Jansen et. al., 2018; Wing, 2006,). DiSessa (2001) made a clear distinction between the three pillars of this new literacy: a) digital technology (material pillar) which is the medium for exploration, b) mental processing, and interpretation of what is explored to personal knowledge (cognitive pillar) and c) social communication of knowledge (social pillar). In 2006, Wing, with her analysis of computational thinking, expanded the ideas of Papert and diSesssa beyond computer science or mathematical problem-solving. However, this new wave did not have much focus on constructionism, i.e. Papert and his group's idea of learning though tinkering with models and developing a language to engage in discourse around this activity (Kafai & Resnick, 1996). Wing (2006) approached computational thinking as "a set of skills, strategies and behaviours", that draws on fundamental concepts of computer science, but it can be implemented for solving problems across all disciplines as well as in everyday life. Today, many researchers agree that CT should be seen as a new kind of literacy which encompasses different scientific fields including mathematics, science and arts (e.g., Israel-Fishelson & Hershkovitz 2022; Kite et al. 2020; Li et al., 2020;). This literacy involves not only coding knowledge, but a set of skills, strategies and behaviors connected to a wider value of cultivating computational thinking for the 21st century citizen.

So, the rhetoric and the whole approach now seems to have turned inside out. Rather than expanding from the generic idea of constructionist mathematical leaning with expressive media, to address transdisciplinary skills (with exceptions such as e.g. Kafai & Resnick, 1996), the task now is to consider how generalised CT can be employed in or added to domain – specific curricula including mathematics. Following that view, there have been efforts to integrate CT across the K-12 curricula as an important skill for all scientific fields and it is in this wake that CT has appeared as important enough to solicit inclusion in mathematics curricula.

We feel that pertinent questions now need to be addressed. What has mathematics education to gain from the integration of CT concepts and practices in its curricula? How can we think of CT *in the service* of mathematics education? How should the CT paradigm be shaped in ways that bring added value to mathematical learning? In the wake of the CT movement, the potential value of learning to program has mostly been considered as an end in itself, without much thought on how it can be put to use by students for expressing mathematical concepts, creating digital objects and behaviors and solving problems.

According to recent reviews (Subramaniam et al., 2022) most studies that explore the integration of CT in mathematics education are focusing on the understanding of algorithmic and/or CT concepts in math-covered tasks, e.g. create a square with iteration to learn the repeat structure, but not the other way around, e.g. understand and use the square property using variables in a parametric procedure. In that way though, students' mathematical reasoning and meaning making are overshadowed by the process of coding which simplifies or hides the mathematics behind the solution. In addition, reviews show that CT and mathematics has mainly been studied in low-level mathematics classes in primary schools and researchers highlight the need for further research on the integration of secondary school mathematics with CT (Chan et al. 2023; Lv et al., 2023).

In this paper we suggest that there is a need to invent, implement and study approaches that focus on CT *in the service* of mathematics education, i.e. the case of engaging in mathematical thinking through the use of programming as a means of expressing mathematical ideas while tinkering with digital models (Noss & Hoyles, 1996). Such approaches could contribute to the transformation of mathematics education using digital media and computational thinking. They would involve the learning of programming and mathematics in conjunction, providing students with experiences in mathematizations made possible through programming. Through an empirical study we explore possible connections and differences between mathematical and computer science learning through a CT activity in MaLT2; a freely available online 3D Turtle Geometry modeler based on Logo, the language originally created for programmable mathematics (Papert, 1980), and affording dynamic manipulation of variable procedure values and 3D camera perusal. As part of a design-based research project we used MaLT2 in an empirical study with secondary education students. The aim was to answer the following question:

A) Which CT practices students implement while engaged in mathematical activities in MaLT2?

B) Which are the math-oriented and CS-oriented solutions that students implement in MaLT2 activities, and what are their differences?

Theoretical Framework

Computational Thinking for Math Education

In the last decade, several studies have explored the integration of Computational Thinking to mathematics education. Recent studies on the field show its growing importance, and the various methods and challenges associated with its integration. Their results indicate that mathematical activties in which CT-based mathematical activities can encourage students to approach mathematical problems systematically, breaking them down into smaller, manageable parts and using algorithms to find solutions (Chan et al. 2021; Cui et al., 2023;

Shumway et al. 2021). Through coding and experimentation with computational mathematical artifacts students can develop mathematical problem-solving and problem formulation, and develop meanings of mathematical concepts, such as geometrical properties or numerical operations. Such activities can also offer a rich context for supporting the transdisciplinary integration of STEM education (Ng et al., 2023).

Despite the growth in relevant studies and tools, there are still significant challenges and gaps to be explored for exploiting CT for mathematical learning in meaningful ways. According to recent reviews (Subramaniam et al., 2022; Ye et al., 2023), most relevant studies are curried out in primary or early secondary educational contexts. As such, they focus on simple or introductory mathematical concepts, leaving the connection of CT with high-level math concepts an understudied area. Moreover, researchers have highlighted that CT should not be introduced as an add-on to mathematics learning or as another context to learn programming (Ye et al., 2023). On the contrary, it is crucial to bring mathematical learning to the foreground of CT-based math activities and conduct more research on understanding and maximizing the ways CT can support mathematical learning. In that context, we argue that it is worth investigating and highlight the different mental processes in which students percieve and implement different CT elements in CT-based mathematical activities, i.e. in math-oriented or CS-oriented processes. Such exploration and destinction could help educators to design math activties that exploit CT to laverage mathematical problem solving and meaning making in ways that where not accessible before the integration of the two domains. Next, we will elaborate on two theoretical constructs that framed our approach of utilizing CT *in the service* of Math education.

The first concerns the so-called "CT practices" and their strong connection to mathematical problem solving. Several CT Frameworks agree that CT involves a set of concepts, practices and perspectives that come from computer science but can be applied for solving problems computationally in different scientific fields and in everyday life (Tikva & Tambouris, 2020; Wing, 2008). The practices refer to certain processes for dealing with complex problems including, amongst others, abstraction, pattern recognition, decomposition and algorithm creation (Brennan & Resnick, 2012; Grover & Pea, 2018). However, similar practices have been described in mathematical problem solving and reasoning long before the definition of CT. For example, the practice of abstraction, which is central in most CT frameworks, is a core concept of mathematical problem solving. In programming, abstraction is usually approached through the creation of abstract data types or structures while in mathematics through the use of variables and functions in geometry and algebra (Kynigos, 1995; Mitchelmore, 2002). Another example is the design, manipulation and analysis of 3D geometrical shapes in geometry that involves pattern recognition and decomposition practices (e.g., break down the cube into squares, repeat a simple shape to construct a more complex one) which are also central to programming

(Tikva & Tambouris, 2021). In that sense, computational thinking frameworks could act as boundary objects between mathematics and computer science education (Ng et al., 2023). However in most CT-based mathematical activities is still quite vague in which context students implement and develop these practices and whether they focus more on the Computer Science or the Mathematical part. It would worth investigating and describe the different forms of CT practices that can be experienced in such activities, in a way that teachers and researchers could focus on the field of interest.

The second point concerns the importance of student's self-expression and communication of ideas through technology in mathematical learning and its support by constructionist CT tools. It is common that in many CT-based mathematical activities students are engaged in closed tasks that are solved through the creation of an algorithm (e.g. connect the given dots). However, in such tasks, students' exploration is limited, and they tend to develop concept-specific fragmented knowledge. On the contrary, open-ended environments that enable students learn through making and sharing of digital artifacts, can be quite beneficiary for meaning generation and the development of problem solving practices. The idea of learning through tinkering models has its roots on Constructionist learning theory (Harel & Papert, 1991), a special kind of fallibilist mathematical activity (Ernest et al., 1991, Kynigos & Diamantidis 2021) which argues that learning occurs naturally when students take agency while making and sharing tangible digital artefacts. Constructionism comprises a strong educational design element (Kynigos, 2015), where powerful mathematical ideas are embedded by pedagogical designers in special kinds of artifacts accompanied by construction units and tools, even a construction language in some cases of digital media. Such construction kits are thus designed to provide dense opportunity for students to concretize and express their ideas by designing themselves, building and engineering (Healy & Kynigos, 2010). Thus, in digital constructionist learning environments learners' agency is encouraged since they become designers using technology to build and modify artifacts which become public entities when shared with peers, while teachers act as facilitators of the process (Kynigos & Diamantidis, 2021). This view highlights the importance of social participation in the learning process, as well as the emergent productions with usually low social impact range (Papert, 1980).

From the two points raised above we can see that Computational Thinking practices have a strong connection with mathematical thinking and with the appropriate tools can become a strong vehicle in mathematical learning .However, we think that there is still a gap in the research, where the question is what kind of tools and activities can afford this kind of engagement with mathematics through coding and sharing of digital artifacts in a way that brings mathematical meaning making to the foreground of CT activity rather than the opposite.

Meshing computational and mathematical affordances in digital learning tools

Despite the wide range of existing digital tools for CT in computer-science education there are still very few prioritizing mathematical learning through programming. As a result, most empirical studies for CT in math education are using tools originally designed to learn programming, such as Scratch (Benton et al., 2017). These tools meet the needs for early-stage mathematics of primary education (Nordby et al., 2022), but usually fail to address the representation and exploration of more complex mathematical concepts of secondary and higher education. One solution to that issue can be the integration of diverse affordances in digital environments, which been proven beneficial for learning of otherwise complex and abstract concepts usually inaccessible before that integration (diSessa, 2001; Morgan & Kynigos, 2014). The affordances of a digital environment refer to the opportunities or restrictions it causes to the learning process by enabling actions between the digital artefact and the student (Calder, 2012).

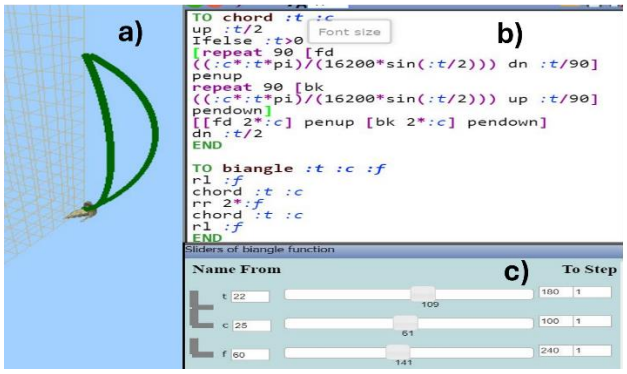

Figure 1. A dynamic 'Biangle on a sphere' in MaLT2 environment, showing the 3 affordances a) the 3D scene with camera perusal b) the text-based programming and c) the dynamic manipulation tool

In our approach we use an environment that integrates three affordances for math-oriented computational thinking called MaLT2, shown in Figure 1 (Kynigos & Diamantidis, 2021; Kynigos & Grizioti, 2018), accessible at http://etl.ppp.uoa.gr/malt2/. The affordances were carefully selected so that they bring mathematical reasoning to the foreground and connect math with coding in a meaningful way for math education. These are a) Logo-based programming of figural 3D models with a language that extends Berkley Logo (Harvey, 1997) with commands for 3D state movements, b) Dynamic manipulation of any figural models created by a parametric procedure. A 'variation tool' with sliders, allows for the instant variation of the parameter values of the executed parametric procedure, resulting in the animation of the model on the scene. c) 3D navigation with a periscopic camera in the 3D scene that allows for examination of 3D models from different angles and scales. The three integrated affordances can allow hard, elusive, or even unattainable mathematical concepts to become understandable and useable by young learners (Kynigos & Grizioti,

2018). Consider for example the "biangle" on a sphere shown in Figure 1 constructed in MaLT2. This is a figure not easily represented with traditional media and at the same time embeds mathematics hardly accessible particularly to students with other means, such as curvature, geodesic and spherical geometry.

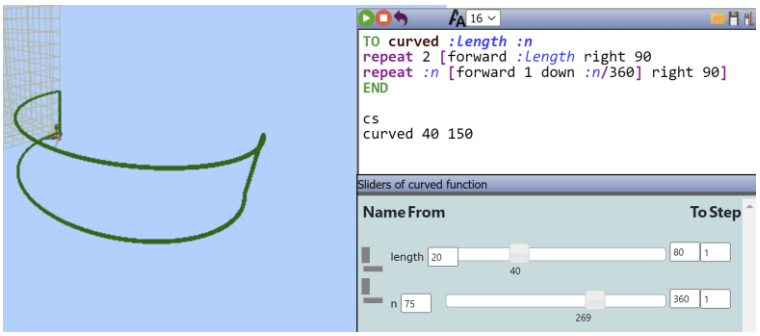

Figure 2. A dynamic square that curves in space in MaLT2 environment

Another example is the dynamic curved square in Figure 2 in which we can think of properties of a cylindrical segment as generated by curving the two opposite segments of a square. This places concepts related to cylinder, like the height, at the centre of a conglomeration of concepts not usually connected with cylinder in traditional curricula such as linear and angular properties of a square.

Methodology and implementation context

To answer the research questions presented earlier we organized an empirical study with students in a school context. The aim was to study in depth how students interact with the tool and with each other while performing certain CT-based mathematical activities in MaLT2 environment.

Research methodology

Since the aim of this study was to identify new forms of CT practices through an approach that has not been studied before, it was important to analyze students' activity and interactions throughout the implementation. Thus, a qualitative methodology was necessary. The research method we used is that of Design-Based Research which evolved from design experiments (Barab & Squire, 2004; Cobb et al., 2003) and includes the design of a pedagogical intervention and its evaluation in real classroom settings with the aim to refine the initial pedagogical design and to develop new theories. We organized a two-year design-based project with two repeated cycles of design, implementation in school context, evaluation, and review. In the first year, a pilot study with secondary education students was implemented that led to the main exploratory study in the second year. This paper presents and discusses the results of the latter study.

Participants, context and process

The main study was implemented as an after-school activity in a junior high school in Greece during the school year 2021-2022. It had a duration of 15 hours divided into three sessions. The participants were 12 students aged 14-15 years old who participated voluntarily. The students worked collaboratively in four groups of three members using one laptop per group. The groups were formed by the mathematics schoolteacher aiming to have mixed groups regarding their math performance. In the first session, they constructed 2D and 3D geometrical objects in MaLT2, including a square, a cube and a pyramid. In that phase, apart from free exploration of the tool, they were also given 2 short math-oriented tasks: a) "create a parallelogram that can never become a square" and b) "create a procedure that draws all the diagonals of any polygon". The aim of the tasks was on one hand for students to become familiar with MaLT2 affordances (e.g. dynamic manipulation of the square, procedural programming) and on the other hand for research to study the solutions that students would develop for such tasks. In the second and third sessions they freely used these objects as building blocks to design an animated scene of their choice. These two phases aimed to the expression of creative ideas by the students and the study of the CT practices they may use in their implementation. Regarding the mathematical level of the students, they had been taught the geometrical properties of 2D objects such as the square and the parallelogram as long as linear functions. They had no experience with 3D geometry, as it is not part of middle school curriculum. They also did not have previous experience with the MaLT2 tool and had little experience in Logo-based programming from Computer Science classes.

Data collection and analysis methods

To better understand student learning activity, we collected a set of qualitative data throughout the study. This involved screen and audio recordings from each group with a total duration of 35 hours and 21 digital artifacts created in MaTL2 by all groups. We did a comparative thematic analysis of the collected data using the Atlas.ti tool. First, we analyzed the transcribed audio recordings of student dialogues looking for critical incidents (Tripp, 2011), i.e. interesting moments of students' interactions with the tool or with each other that can relate to the research question, which we coded with analysis codes. For the coding of the critical incidents we used the abductive coding approach (Tavory & Timmermans, 2019), that is to start the analysis with an initial coding scheme which is then modified and enhanced with new emergent codes leading to a final scheme after several iterations. This approach allowed the researcher to remain open to emergent findings. Then we triangulated the incidents with the screen recordings and the digital artifacts validate the coding scheme and to identify themes of similar codes. For the first RQ we focused on CT practices drawn from relevant CT frameworks found in the literature (Tikva & Tambouris,

2021) and whether participants used them in mathematical problem solving. For the second RQ we further analyzed the episodes of the detected CT strategies of RQ1 to identify CS-oriented or math-oriented approaches, creating new codes where necessary. The final coding scheme was reviewed by an external researcher from the field of digital media and programming in mathematics education.

Results

In the analysis for the first RQ we identified three CT practices implemented by the students in the two phases of the study: abstraction, pattern recognition and decomposition (Table 1-column a). Abstraction code referred to processes of generalizing a specific solution to a general one that could be applied in different cases. Pattern recognition code involved processes of recognizing similarities (patterns) between entities. These entities were not only algorithmic commands but also geometrical objects or different problems. Decomposition code was related to processes of breaking down a complex entity (i.e. a problem, an algorithm, an object) into smaller parts that were more manageable to handle and understand. In the second part of the analysis for RQ2, for each practice we identified different forms in which students implemented it in their solutions, which were coded as 'CS-oriented' or 'math-oriented' depending on the concepts and competences they involved (Table 1). For instance, regarding pattern recognition practice, we detected 3 different forms: "code patterns, visual patterns and numerical patterns". In table 1 we give some examples of codes of both categories (CS-oriented and math-oriented) for each CT practice and a short description explaining the rationale behind the coding. Next, we present three representative cases of students' activity that show the difference between CS-oriented and math-oriented implementation of CT practices. We further discuss how CT in each implementation may foster or hamper down the process of mathematical meaning making.

Example 1: The case of a non-Square Parallelogram

One of the tasks given to the students was to create a parametric model of a parallelogram that would never become a square, i.e. with any value that are executed. After activity and deliberation leading to the production of a generalised parallelogram, adjoining segments independent, opposite segments equal, consecutive turns supplementary. But in one class a group of students mistakenly thought that this definition was too general because as they said, it did not only create parallelograms but also squares (Kynigos & Diamantidis, 2021). Their teacher did not intervene to provide information about this error. He appreciated the initiative and the agency and allowed this to become a classroom project - how to create a parallelogram which can never become a square.

Table 1. Examples of codes for each CT practice and their description

CT Practice	Example of Codes	Description
Abstraction	Code abstraction (CS-oriented)	Implement abstraction in their code e.g. add parameters to a Logo procedure, use variables instead of numbers
	Geometrical abstraction (math-oriented)	Express abstract rules for the 3D model on the scene e.g. generalize a 3D objects' properties
	Algebraic abstraction (math-oriented)	Create relations between variables using mathematical functions e.g. create periodic animation of two 3D objects, create proportional animation between two moving parts
Pattern Recognition	Code patterns (CS-oriented)	Implement/find a pattern in the Logo algorithm. E.g. repeat a set of commands, repeat a function
	Visual patterns (math-oriented)	Create/find patterns in the 3D model e.g. sequence of shapes, create a pattern in the animation of an object
	Numerical patterns (math-oriented)	Use the sliders to create patterns while changing the numerical input of a procedure (e.g. change the slider step to alter the animation, change the slider limits)
Decomposition	Code decomposition (CS-oriented)	Break down the code to different procedures with a focus on the commands
	Visual decomposition (math-oriented)	Break down the code based on the different geometrical elements of the visual result e.g. create one procedure for the house roof and one the main part.
	Problem decomposition (CS-oriented)	Break down a design problem into smaller parts and solve them separately. E.g. approach the design of the house parts as separate problems and then combine them together.

In the data analysis we observed that all given solutions could be grouped into two types: a math-oriented solution that uses a distorted functional relation between the two consecutive sides of the rectangle, and a CS-oriented solution that uses conditional statements for avoiding the creation of a square (i.e. turn 90 degrees). Figure 3 shows two such examples given by two groups, while critical incident 1 in Table 2 shows a representative moment of group 2 students dialogue, while they decide to implement the math-oriented solution. The students first express their ideas about the properties of the square and those of the rectangles. When S2 says that in order to reassure that the sparrow would never draw a square they have to be sure that the angle is never 90 degrees, S1 explains that rectangles have 90-degree angles, but they are not squares, as they don't have equal sides. Through the discussion they decide to focus on sides' length rather than the

angles of the parallelogram. Later, S1 suggests having two variables, one for each pair of sides. At that moment S2 express the idea of creating a functional relation between the non-equal sides for having only one slider to manipulate the shape. In the end they decide to use two parameters, :x for the side length and :c for the angles between the sides. In the code (Figure 3a) they have used unequal functional relations between the consecutive sides of the parallelogram by telling the hummingbird to move forward *x* steps, turn right *c* degrees and then move forward *x+20* steps, resulting in a non-square shape even when the angle (:c) is 90 degree. In this case students thought about the definition of a rectangle and that of a square and created a design that fits the one but not the other. This solution involves mathematical reasoning about both the quadrilaterals' properties and about functional relations (replace y with x+20 for the second pair of edges). We can say that this is a math-oriented computational solution as CT concepts (repetition, variables) are being used to achieve the target by *doing* mathematics.

Table 2. Critical Incident 1

St.	**Transcription**	**Code**
S1	Ok so to exclude all squares we need the ensure that the two opposite sides are never equal. [*changes the slider to animate the shape for different input values for the side lenght*]	Mathematical meaning making→ quadrilaterals
S2	And that the angle is not 90 degrees.	
S1	No this doesn't matter. If the angle is 90 and the two opposite sides are unequal it would be a rectangle, not a square. Look [*puts the slider for the angle to 90 and changes the other two to create a rectangle*]	
S2	You are right.	
S1	Let's try to tell the bird to move forward x steps for the two opposite sides and y steps for the two other	Code patterns
S2	Or we can have only one variable x and the other can be x plus a number, like x + 30. In that way we will have only one slider!	Mathematical meaning making→ functional relations

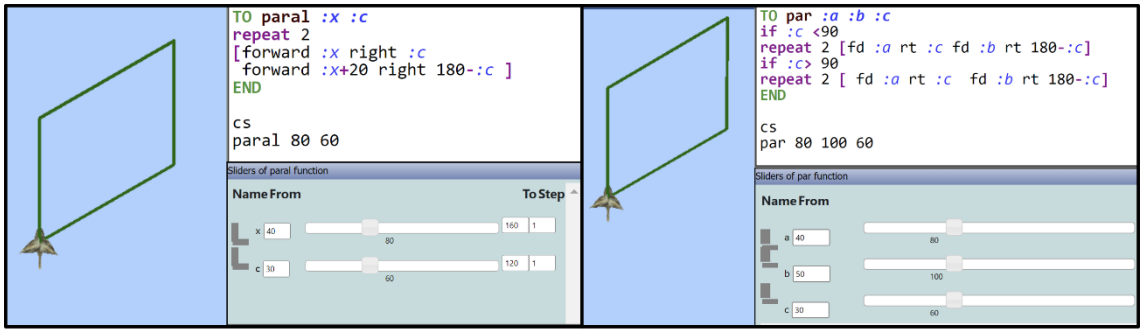

Figure 3: Two student solutions for the problem of non-square parallelogram. a) math-oriented using functional relations b) CS-oriented using if structure

The other solution developed by group 1 students (Figure 3b) uses three parameters, :a for the two opposite sides length, :b for the other two opposite sides length, and :c for the angle turn. The code draws a rectangle with no functional relation between its sides. Students used the conditional structure "if" to check whether the input number for the angle parameter (a) is 90. In that case the algorithm does not do anything, avoiding drawing a square, but also all other rectangles that are not squares and should be drawn according to the task. Regarding mathematical meaning making, this solution avoids the use of functional relations and follows a more CS-oriented approach.

Example 2: The case of n houses

In the phase of creative construction, a group of students decided to create an animated neighborhood consisting of a variable number of 3D models of consequent houses defined by a variable 'a'. The number of houses would change with the slider of the 'a' parameter creating an interesting animation. Their first attempt was to repeat the procedure "house" 'a' times, which however resulted to the model shown in figure 4a. The cause was that the ending position of the hummingbird (the equivalent of the 'turtle' avatar in Turtle Geometry), after drawing the house, was the top of the pyramid, rather than the bottom of the cube. Note that to create the house in the first place, students developed and used the trigonometric and Pythagorean properties to get the hummingbird to the center of the square base and then send it up adding the height of the pyramid rooftop to the cube's length.

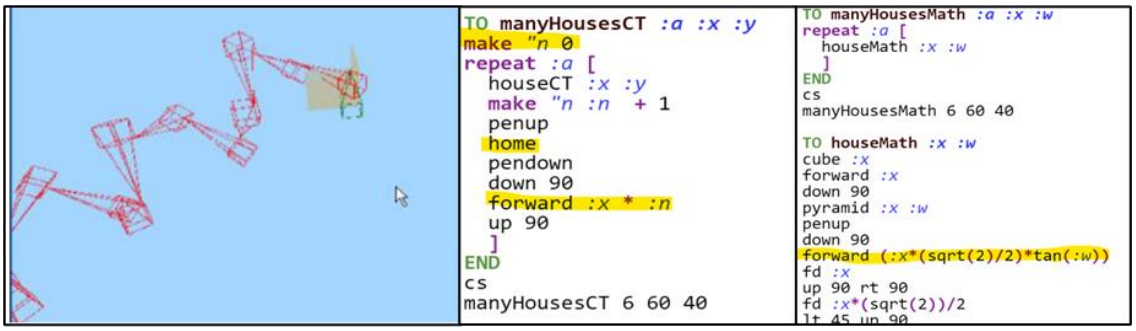

Figure 4: From left to right: a) the wrong construction of n houses, b) the CS-oriented solution, c) the math-oriented solution

To solve the problem of taking the hummingbird from the rooftop pyramid vertex to a point appropriate in order to build an adjoining house, the students developed two solutions that were both functional and with the required result. However, when looking at the two codes and the meanings students developed through their dialogues, we can see two quite different approaches from a conceptual point of view. The first, coded as "CS-oriented", involved concepts and practices from computer science (Figure 4b). After drawing one house the hummingbird goes to position [0.0.0] with the 'home' command,

orients down and then moves forward x*n (where x is the length of the cube and n the number of houses drawn so far in the repetition).

In the second solution, coded as "math-oriented", the hummingbird moves to the correct position of the pyramid's base using relatively complex mathematical calculations originally used to get it to the rooftop (Figure 4c). This involved adding the cube's height to the pyramid's central height calculated with Pythagorean theorem and then going to the corner of the cube basis. Critical incident 2 (Table 3) is a representative moment of this process, in which students express mathematical ideas for solving the problem of n-houses. After visually decomposing the house to smaller geometrical shapes they discuss how they could apply mathematics (Pythagorean theorem, calculation of angles) to bring the hummingbird to the correct position before drawing the next house. In this solution students considered the hummingbird's end position as part of the geometrical figure and thus it is calculated inside the 'house' procedure, and it is independent from any external repetition. On the contrary, in the first solution the end position of the hummingbird was approached as part of the repeat structure and was consequently calculated outside the 'house' procedure where a local variable is created (n) counting the repetition number. As in the previous case, here CT concepts (repetition and home command) were used to skip some mathematical ideas that were developed in the first solution.

Table 3. Critical Incident 2

St.	**Transcription**	**Code**
S3	So, after drawing the house we want the bird to go to this corner before drawing the next house [*points the mouse pointer to the front down right corner of the cube*]	
S4	Yes but there is no such command. Do we have to find the co-ordinates and say go to x y?	
S3	No no. Let's think. We need to take it to this point of the pyramid and then it is just go forward the length of the cube	Visual decomposition
S4	Oh, you are right. So, we only need to calculate this edge of the pyramid! How much is this? [*Turns the camera to look the shape from different angles*]	Visual decomposition
S3	Maybe we could use the Pythagorean theorem for this triangle? [*shows the triangle with the mouse while turning the camera around*]	Mathematical Meaning Making
S4	But we don't know this angle	
S3	Ok. Let's look at the commands that create the pyramid and maybe we can find the angles there. And we can do the opposite calculations.	

Example 3: The case of polygon diagonals

Rather than only juxtaposing CT and mathematical solutions it was interesting to us to consider cases where CT in fact helped to structure the mathematical properties in a unique way. In the second task of the first session, students tried to create a parametric procedure

that would draw all the 'diagonals' of any polygon. One interesting solution is the one shown in Figure 5. The code follows the CT practice of decomposition in a way that makes complex mathematics easier to express and to understand. It consists of one main procedure (diagonals :n :s) and four sub-procedures. The sub-procedure draw_diagonals :n :s :k (Figure 5b) uses the CT concept of recursion to draw :n-2 diagonals of a n-sided polygonwith only four commands (a turn right, a move forward and a move backward). The mathematical calculations for a) the length of the diagonal and b) the turn required from the respective polygon vertex to create it are expressed in two separate sub-procedures called 'diameter' and 'length_diagonal' (Figure 5a), highlighting the mathematics of the solution in a neat way. The solution can also be found online in this url: 'http://etl.ppp.uoa.gr/malt2/?diagonals'.

This example utilizes the feature of structured procedural programming, offered by the Logo-based language of MaLT2, to decompose the algorithm in an easy-to-follow structure. This is a case where the CT practice of decomposition along with CT concepts of repetition and recursion are being used to create accessible solutions that make mathematical ideas easier grasp, rather than traditional solutions with long algebraic expressions and "spaghetti" code. Here, students also made a distinction between what we could called 'mathematical' procedures, that are those focusing on do the math calculations needed for the algorithm, and 'computational' procedures designed to do the step-by-step solution of the problem in a computationally efficient way (e.g. by using recursion).

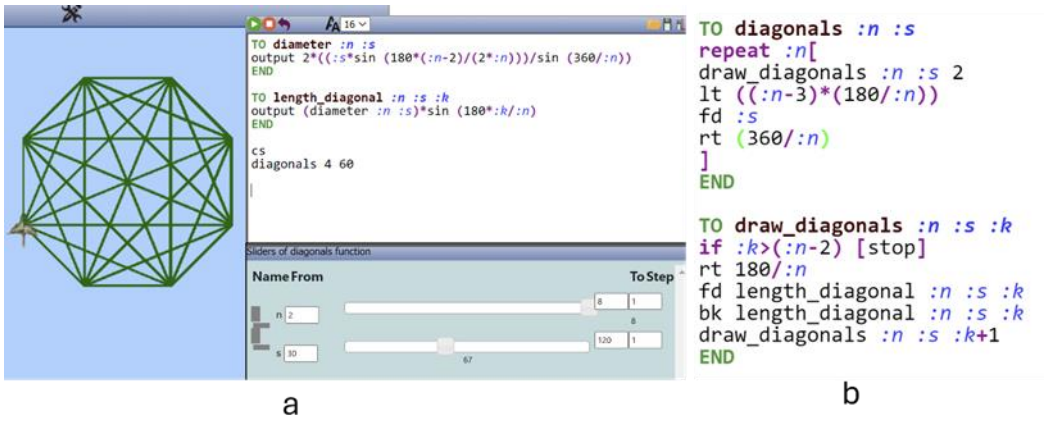

Figure 5: A solution for the polygon diagonals a) on the left, two sub-procedures focusing on the mathematics b) two procedures that call the 'mathematical' procedures

Discussion

The results of the presented study show that CT concepts and practices may be used in different ways for solving the same mathematical problem in constructionist coding tools like MaLT2. Depending on the approach that is implemented in the solution, CT could be utilized to promote either mathematical meaning making or computer science meaning

making, like in the example of n houses. The different forms of CT practices shown in table 1, indicate that CT can have a wide range of implementations and approaches in which programming is there but not with the traditional role. This finding is in alignment with recent studies and reviews which highlight the need of taking CT beyond programming and Computer Science learning (Kite et al. 2021). The different solutions presented in the three examples, show how CT concepts (e.g. recursion, repetition, procedural programming) and CT practices (e.g. decomposition, abstraction) can be used in diverse ways for mathematical meaning making by the students. For instance, in order to solve the problem with the repetitive houses students used decomposition in a visual way, that was to break down the different shapes of the house and then to focus on the specific triangle and its geometrical properties in combination with the respective commands. In the example of the diagonals students used decomposition in their code by breaking down their algorithm to smaller procedures making the mathematical calcuations more manageble.

The study results they also revealed that in some cases the use of CT concepts for solving a mathematical problem could conceal the mathematics expected to be used by students in a given task. For example, the use of conditionals in the first example (the non-square parallelogram) was computationally correct but prevented students from thinking about functional relations and design a more general and mathematically correct solution for the given task. This finding highlights the importance of a thorough didactical engineering of CT activities by math teachers in order to bring math competences and concepts to the foreground. There is need for more studies and new frameworks that would focus on the design process of activities that utilize CT in the service of mathematical learning rather than the other way around.

Finally the results indicate that the integration of math-oriented affordances in educational programming tools, such as 3D design and dynamic manipulation, can enhance the development of CT practices in a mathematical context. Dynamic manipulation allows for animations affording students with a sense of normalization but also a tangible experience of what changes and what stays the same in a mathematical construction. Considering models in 3D space greatly enhances the mathematics accessible to students for mathematizations in relation to what was possible with non-digital means. The notion of embedding mathematical concepts and properties in a parametric algorithm creating a figural animation provides teachers with the potential to generate situations where students may focus and use a specific field of concepts. Also, the students might develop experiences with mathematical generalizations such as the idea of a property of a class of figures or constructs or the idea of generalized number and variable. A point worth mentioning is that in the three examples, students' collaboration and communication with each other played a key role in the meaningful exploitation of these affordances during the learning activity. Other studies have shown as well that collaborative activities with digital

media can enable students to critically reexamine their claims and build on each other's ideas, leading to more sophisticated reasoning and understanding of mathematical concepts (Francisco, 2013; Noss & Hoyles, 1996; Kynigos et al., 2020). In this study we further showed the importance of collaboration in the exploitation of the computational affordances of the learning environment in favor of mathematical meaning making. In both critical incidents that we presented, students made use of the affordances (e.g. the camera, the sliders) while arguing about their ideas on the solution, leading them to generate meanings about mathematical concepts (e.g. the rule about rectangles and squares) through computational practices. It was their disagreements and different perspectives that lead them to utilize computational concepts for supporting their arguments (like in the example of the non-square parallelogram) or for contributing to the solution of the other team member (in the example of the n-houses).

The presented study had also some limitations that should be mentioned. The small number of participants does not allow for generalizability of the results. However, it was necessary for the qualitative analysis that resulted to a first coding scheme that can be used and evaluated in future studies with a larger number of participants. In addition, the implementation needed to be longer since many students did not have the time to complete their constructions, which would probably provide us with more data for their learning activity.

Conclusion

This paper presented the results of an empirical study that explored the different ways in which secondary school students implement CT practices to solve mathematical problems in the MaLT2 online tool. The results from the qualitative analysis of 12 students' dialogues and interactions, showed that they implemented the CT practices abstraction, pattern recognition and decomposition in their solutions. In the results we further identified and described math-oriented and CS-oriented solutions of students' CT practices. Abstraction was identified in three forms, i.e. as "code abstraction" (cs-oriented), "geometrical abstraction" (math-oriented) and "algebraic abstraction" (math-oriented). Pattern recognition was identified in the forms of "code abstraction"(cs-oriented), "visual patterns" (math-oriented) and "numerical patterns" (math-oriented). Decomposition was detected in the forms of "code decomposition" (cs-oriented) , "visual decomposition" (math oriented) and "problem decomposition" (cs-oriented). The three examples that were presented explore the differences of these solutions in terms of the meanings students develop and their connection to computer science or mathematical learning. Based on the results and discussion we suggest that there is a need to re-think ways in which CT can come back to serve mathematics education by means of providing students with a representational medium to engage in mathematization experiences through programming. We could imagine a whole curriculum with the multitude of option to embed, math-oriented CT

models and the wealth of student solutions and constructions of such models in discursive collectives enhancing both mathematical thinking and vocabulary and most of all bringing back a love for mathematics and mathematical thinking in young people. Further research could shed light on the mapping between Math and CT and lead to a CT framework for math education, which is currently missing from the literature.

Acknowledgments

This work was complementrarily funded by the followign European Projects:

- 2023-2025: TransEET Transforming Education with Emerging Technologies, HORIZON Coordination and Support Actions, HORIZON-WIDERA-2021-ACCESS-03-01 – Twinning. Project: 101078875.
- 2022-2025: Exten.D.T.2 Extending Design Thinking with Emerging Digital Technologies, HORIZON-RIA Proposal number: 101060231, Activity: HORIZON CL2-2021-TRANSFORMATIONS-01-05.

References

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The Journal of the Learning Sciences*, *13*(1), 1–14.

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, 115–138. https://doi.org/10.1007/s40751-017-0028-x

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, 1.* https://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Calder, N. (2012). *Processing mathematics through digital technologies*. Springer Science & Business Media.

Chan, S.-W., Looi, C.-K., Ho, W. K., & Kim, M. S. (2023). Tools and approaches for integrating computational thinking and mathematics: A scoping review of current empirical studies. *Journal of Educational Computing Research*, *60*(8), 2036–2080. https://doi.org/10.1177/07356331221098793

Cobb, P., Confrey, J., diSessa, A., Lehrer, P., Schauble, L. Design. (2003). Experiments in educational research. *Educational Researcher*, *32*(1), 9–13.

Cui, Z., Ng, O. L., & Jong, M. S. Y. (2023). Integration of computational thinking with mathematical problem-based learning. *Educational Technology & Society*, *26*(2), 131–146.

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school*, *19*(1), 19–38.

diSessa, A. 2001. *Changing Minds: Computers, Learning and Literacy*. MIT Press.

Ernest, P., Skovsmose, O., Van Bendegem, J. P., Bicudo, M., Miarka, R., Kvasz, L., & Moeller, R. (1991). *The philosophy of mathematics education*. Springer Open.

Francisco, J. (2013). Learning in collaborative settings: Students building on each other's ideas to promote their mathematical understanding. *Educational Studies in Mathematics*, 82, 417–438. https://doi.org/10.1007/S10649-012-9437-3

Harel, I. E., & Papert, S. E. (1991). *Constructionism*. Ablex Publishing.

Harvey, B. (1997). *Computer science logo style*. MIT Press.

Healy, L., & Kynigos, C. (2010). Charting the microworld territory over time: Design and construction in mathematics education. *ZDM – Mathematics Education*, *42*, 63–76. https://doi.org/10.1007/s11858-009-0193-5

Israel-Fishelson, R., & Hershkovitz, A. (2022). Studying interrelations of computational thinking and creativity: A scoping review (2011–2020). *Computers & Education*, *176*, 104353. https://doi.org/10.1016/j.compedu.2021.104353

Jansen, M., Kohen-Vacs, D., Otero, N., & Milrad, M. (2018). A complementary view for better understanding the term computational thinking. In *Proceedings of the International Conference on Computational Thinking Education 2018*, (pp. 2–7). CoolThink@JC. The Hong Kong Jockey Club, The Education University of Hong Kong.

Kafai, Y. B., & Resnick, M. (Eds.). (1996). *Constructionism in practice: Designing, thinking, and learning in a digital world*. Routledge.

Kite, V., Park, S., & Wiebe, E. (2021). The code-centric nature of computational thinking education: A review of trends and issues in computational thinking education research. *SAGE Open*, *11*(2), 1-17. https://doi.org/10.1177/21582440211016418

Kynigos, C. (1995). Programming as a means of expressing and exploring ideas: Three case studies situated in a directive educational system. In A. diSessa, C. Hoyles, R. Noss, & L. D. Edwards (Eds.), *Computers and exploratory learning* (pp. 399–419). Springer Berlin Heidelberg.

Kynigos, C. (2015). Constructionism: Theory of learning or theory of design? In S. J. Cho (Eds.), *Selected Regular Lectures from the 12th International Congress on Mathematical Education* (pp. 417–438). Springer.

Kynigos, C., & Diamantidis, D. (2021). Creativity in engineering mathematical models through programming. *ZDM-Mathematics Education*, *54*, 149–162. https://doi.org/10.1007/s11858-021-01314-6

Kynigos, C., Essonnier, N., & Trgalova, J. (2020). Social creativity in the education sector: The case of collaborative design of digital resources in mathematics. In V. Glăveanu, I. Ness, & C. Saint Laurent (Eds.), *Creative Learning in Digital and Virtual Environments*, (pp. 30–49). Routledge.

Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating Turtle geometry, dynamic manipulation and 3D Space. *Informatics in Education*, *17*(2), 321–340. https://doi.org/10.15388/infedu.2018.17

Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020). Computational thinking is more about thinking than computing. *Journal for STEM Education Research*, *3*(1), 1–18. https://doi.org/10.1007/s41979-020-00030-2

Lv, L., Zhong, B., & Liu, X. (2023). A literature review on the empirical studies of the integration of mathematics and computational thinking. *Education and Information Technologies*, *28*(7), 8171–8193. https://doi.org/10.1007/s10639-022-11518-2

Mitchelmore, M. C. (2002). The role of abstraction and generalisation in the development of mathematical knowledge. Paper presented at the *2nd East Asia Regional Conference on Mathematics Education (EARCOME) and the 9th Southeast Asian Conference on Mathematics Education (SEACME)*, Singapore, May 27-31.

Morgan, C., & Kynigos, C. (2014). Digital artefacts as representations: Forging connections between a constructionist and a social semiotic perspective. *Educational Studies in Mathematics*, *85*(3), 357–379. https://doi.org/10.1007/s1064 9-013-9523-1

Ng, O. L., Leung, A., & Ye, H. (2023). Exploring computational thinking as a boundary object between mathematics and computer programming for STEM teaching and learning. *ZDM–Mathematics Education*, *55*(7), 1315–1329. https://doi.org/10.1007/s11858-023-01509-z

Nordby, S. K., Bjerke, A. H., & Mifsud, L. (2022). Computational thinking in the primary mathematics classroom: A systematic review. *Digital Experiences in Mathematics Education*, *8*(1), 27–49. https://doi.org/10.1007/s40751-022-00102-5

Noss, R., & Hoyles, C. (1996). *Windows on mathematical meanings: Learning cultures and computers* (Vol. 17). Springer.

Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics. *International Journal of Mathematical Education in Science and Technology*, *3*(3), 249–262.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books Inc.

Shumway, J. F., Welch, L. E., Kozlowski, J. S., Clarke-Midura, J., & Lee, V. R. (2021). Kindergarten students' mathematics knowledge at work: the mathematics for programming robot toys. *Mathematical Thinking and Learning*, *25*(4), 380–408. https://doi.org/10.1080/10986065.2021.1982666

Subramaniam, S., Mahmud, M. S. & Maat, S. S. (2022). Computational thinking in mathematics education: A systematic review. *Cypriot Journal of Educational Sciences*, *17*(6), 2029–2044 https://doi.org/10.18844/cjes.v17i6.7494

Tavory, I., & Timmermans, S. (2019). Abductive analysis and grounded theory. *The SAGE handbook of current developments in grounded theory*, 532–546. Sage. https://doi.org/10.4135/9781526485656.n28

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature review. *Computers & Education, 162*, 104083. https://doi.org/10.1016/j.compedu.2020.104083

Tripp, D. (2011). *Critical incidents in teaching (classic edition): Developing professional judgement.* Routledge.

Ye, H., Liang, B., Ng, O. L., & Chai, C. S. (2023). Integration of computational thinking in K-12 mathematics education: A systematic review on CT-based mathematics instruction and student learning. *International Journal of STEM Education*, *10*(1), 3. https://doi.org/10.1186/s40594-023-00396-w

Wing J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A, 366*(1881), 3717–3725.