

Programming Approaches to Computational Thinking: Integrating Turtle Geometry, Dynamic Manipulation and 3D Space

Chronis KYNIGOS, Marianthi GRIZIOTI

*National and Kapodistrian University of Athens, Educational Technology Lab,
School of Philosophy, Faculty of Pedagogy, and CTI-Diophantus, Greece
e-mail: kynigos@ppp.uoa.gr, mgriziot@ppp.uoa.gr*

Received: March 2018

Abstract. During the last decade, coding has come to the foreground of educational trends as a strong mean for developing students' Computational Thinking (or CT). However, there is still limited research that looks at coding and Computational Thinking activities through the lens of constructionism. In this paper, we discuss how the knowledge we already have from other thinking paradigms and pedagogical theories, such as constructionism and mathematical thinking, can inform new integrated designs for the cultivation of Computational Thinking. In this context, we explore students' engagement with MaLT (Machine Lab Turtle-sphere)¹, an online environment of our design that integrates Logo textual programming with the affordances of dynamic manipulation, 3D graphics and camera navigation. We also present a study on how the integration of the above affordances can promote constructionist learning and lead to the development of CT skills along with the generation of meanings about programming concepts.

Keywords: Logo geometry, computational thinking, programming, dynamic manipulation, 3D graphics.

Introduction

In the past 50 years or so programming has been considered not only as a skill in itself but also as an expressive medium, as a means to generate meanings around powerful ideas (Papert, 1980). However, the importance given to learning to program has been seriously fluctuating, depending on a complex web of issues not excluding techno-social trends. Back in the 80's, many schools showed great enthusiasm in teaching their students computer programming, while by the mid-1990s schools turned away from pro-

¹ <http://et1.ppp.uoa.gr/malt2/>

programming, mainly because of a lack of subject-matter integration and a lack of qualified instructors (Noss and Hoyles, 1996). When software tools with dynamic manipulation and icon-driven technology appeared, programming was seen as a kind of unnecessary noise to doing interesting things with digital media (Kynigos, 2015). However, during the last decade, we are facing a big comeback in attention and value attributed to programming (Kafai, Bruke and Resnick, 2014) in a new context in which it is considered as an integral part of a new perception of computational literacy. Recently, renewed perceptions about the value of computer programming have become popular; that every child can and should learn to write code as a way to develop their computational thinking. The idea of computational thinking that Wing described in 2006 as ‘a set of thinking skills, habits and approaches that are integral to solving complex problems using a computer and widely applicable in the information society’, is now considered as a necessary expressive skill for 21st century students (Barr, 2011; Kafai, Bruke and Resnick, 2014; Lee *et al.*, 2012). However, there is great need for exploring pedagogical approaches and strategies for the acquisition and improvement of such skills through programming and for understanding their nature and potential uses and manifestations (Brennan and Resnick, 2014).

In this paper we argue that the challenge remains, both for designers and educators, to consider what kinds of learning processes may emerge and require encouragement and to what end students may engage with the new programmable media. This is an ongoing challenge since each new affordance providing potential for expressivity poses a need to study the ways in which it may be put to use. On top of this the endless combinations of this new affordance with other existing affordances creates a complex need for research which never has the time to provide solid information before the new affordance appears. Consider this: how much do we actually know about the expressivity potential of textual programming? Are we so sure block-based and visual programming are more powerful as means to engage students? Should we pursue studies in textual programming especially now that there are diverse ways in which it can be integrated with other affordances such as dynamic manipulation, intelligent feedback, a huge diversity of things to program?

Constructionism is a learning theory and a framework for design of programmable expressive media, based on Papert’s generic vision and ideas for learning (Papert, 1980). Constructionist design approaches for educational tools aim to engage users of all ages in the construction and mediation of personally meaningful artefacts through programming (diSessa, 2001; Papert and Harel, 1991). In this paper we describe a textual programming tool which we call MaLT2 (Machine Lab Turtleworlds) as a means to elaborate an example of a process to make pedagogical decisions on a specific configuration of affordances made available at different times and for different reasons. We attribute special value in this process of integrating affordances based on pedagogical principles (Kynigos, 2004). We thus propose the approach of *integrated constructionist designs* in which programming is combined with other affordances forming a meaningful and enriched computational environment. The affordances are selected and implemented in a way that they engage students with scientific concepts and at the same time promote the development of wider skills, like these included in CT. In integrated designs, students

use the different affordances, including coding, as means for self-expression, communication of ideas and design of personal artefacts. In MaLT2, we address the integration of textual programming with dynamic manipulation and 3D modelling. This paper is an extension of a paper written for the Constructionism 2018 conference held in Vilnius, Lithuania.

Rethinking Programming Approaches to Computational Thinking

All this recent enthusiasm around Computational Thinking and ‘coding for all’, has led to a wide development of new educational software and applications that aim to engage students of all ages with programming. Block-based programming tools such as Scratch, Alice, ToonTalk and apps like Kodu and LightBot have become very popular and have attracted many young people, but also teachers and adults, to write simple programs with them. At the same time campaigns like the ‘Hour of Code’ and ‘EU code week’ are organized every year to introduce young students to programming.

Research on computational thinking points out the importance of active engagement, construction and exploration in programming activities (Brennan and Resnick, 2014; Resnick, 2014; Stager, 2014). In many cases though (with the exceptions of Scratch and Alice), students’ engagement with programming is restricted since they get given but a series of closed quizzes and puzzles with the “making” and the social elements being limited. In addition, block-based programming can also have significant limitations for older or more experienced students (Weintrop and Wilensky, 2015). On the contrary, former pedagogical designs like textual programming are left aside regarded as obsolete. This is confirmed by the fact that there is still limited research focusing on the Computational Thinking skills that can be promoted with textual programming, with Logo-like languages for instance.

The question that arises is how these ‘obsolete’ but yet deeply studied approaches, can contribute to the design of new coding tools by providing a strong pedagogical background. For instance, a large number of studies have shown the benefits of Logo textual programming and the effectiveness of turtle geometry in offering rich mathematical experiences and encouraging the construction of meaning (Clements *et al.*, 2007; Kynigos, 1995; Papert and Harel, 1991). **We argue that if coding is seen as a new kind of expression and mediation of meaning, then it may be worthwhile investing on designing ways to make formalism functional and meaningful through its connection to other representational forms in a dependent way (Kynigos, 2015).** New educational tools for programming could for instance maintain the basic design principles of conventional Logo-based designs and be enhanced with new technologies in parts which are meaningful for the learning process such as dynamic manipulation, a variety of what is programmed (e.g. robots, devices, digital objects with properties, behaviours and interactions in diverse fields), 3D representations etc. In such tools, the development of learner’s skills would be achieved through authentically creative and constructionist activities and coding will be seen a vehicle to enhance them.

Towards an Integrated Approach

Computational Thinking is strongly connected with other known thinking paradigms such as mathematical thinking, engineering and scientific thinking and algorithmic thinking (Sinclair and Moon, 1991; Papert, 1972; Wing 2008; Hu 2011). Thus, for the development of the integrated designs described above, it is quite important to investigate how grounded implementations and approaches from these paradigms may contribute.

Let's take for example the element of abstraction. Abstraction is considered by many researchers an important CT skill and the 'mental tool of computing' (Wing, 2006). However, the concept of abstraction and abstract thinking is not new at all. Abstraction and generalization are core concepts of mathematics (Noss and Hoyles, 1996). Despite the differences that mathematical and programming abstraction may have (Wing, 2008), the long year research and the theoretical constructs from mathematics education, can be a strong basis for the development of tools and methods that foster students' abstraction skill. An interesting method that comes from the world of the Dynamic Geometry Environments (DGE's) is the dynamic manipulation of geometrical constructions (Laborde *et al.*, 2006). Dynamic manipulation helps students to understand the properties of geometrical objects and generalize their rules and the relationships between them (Goldenberg and Cuoco, 1998; Psyharis and Kynigos, 2009). This is particularly interesting with respect to thinking about abstraction as a process of defining or approaching a construction as 'how it behaves when manipulated dynamically'. We argue that dynamic manipulation could also be beneficial in the context of programming abstractions and Computational Thinking, even though it has not yet been studied enough.

Another example from mathematics that requires a number of mental skills is that of creating and manipulating 3D geometrical shapes. When students interact with 3D models they have to use spatial thinking which includes complex mental processes like the capability of understanding and recognizing the location and shapes of the objects, their relations to each other and their movement in space (Kynigos and Latsi, 2007). To **achieve that they usually apply skills like pattern recognition or decomposition** which are also included in CT (Lee *et al.*, 2011). Thus the programming and manipulation of 3D geometrical objects in a digital environment could also be a technique for supporting the cultivation of such skills.

The above examples indicate that the integration of digital affordances and concepts from other fields like mathematics can possibly be beneficial for CT skills. In our research, we try to investigate and reveal the common elements between mathematical, algorithmic and computational thinking paradigms and study their possible development through such integrated designs. In the presented study we focused on four skills which are considered by many researchers across the literature, as an important part of a learners' skill set (Lee *et al.*, 2011; Krauss and Prottzman, 2017). These are:

- a) Decomposition.
- b) Pattern recognition.
- c) Abstraction.
- d) Algorithm design.

MaLT2: Programming to Create 3D Dynamic Models

Meshing Diverse Affordances

In order to investigate the benefits of such integrated approaches to both programming knowledge and CT skills, we organized a study with MaLT2 (Machine Lab Turtle-sphere), an online Logo-based application of our design that allows the creation, exploration and dynamic manipulation of 3D geometrical models with textual programming. MaLT2 provides a learning environment that endorses the benefits and the strong theoretical basis of Logo programming but also extends it with new technological features in such ways that could enhance the development of CT skills. It can be found at <http://et1.ppp.uoa.gr/malt2>

MaLT2 implements the classic Turtle Geometry of Papert (Abelson and DiSessa, 1980) with Brian Harvey's Berkeley Logo (Harvey, 1997), extending it with the integration of the following three affordances:

- (1) The turtle becomes a sparrow. The designed Logo language supports the creation of 3D drawings by navigating a sparrow (instead of a turtle) in a 3D spherical space. This feature broadens the range and the complexity of objects that can be programmed with the Logo language. It is based on an elegantly concise two new primitives, 'roll' and 'pitch', coined a long time ago by Horacio Reggini (1985) turning the turtle on two perpendicular planes to the traditional 2D version.
- (2) The drawings become alive. MaLT2 implements a dynamic manipulation system, allowing the animation of any 3D model that has been created by a parametric procedure (i.e. *squares :a :n :b*, see Fig. 1). To use the dynamic manipula-

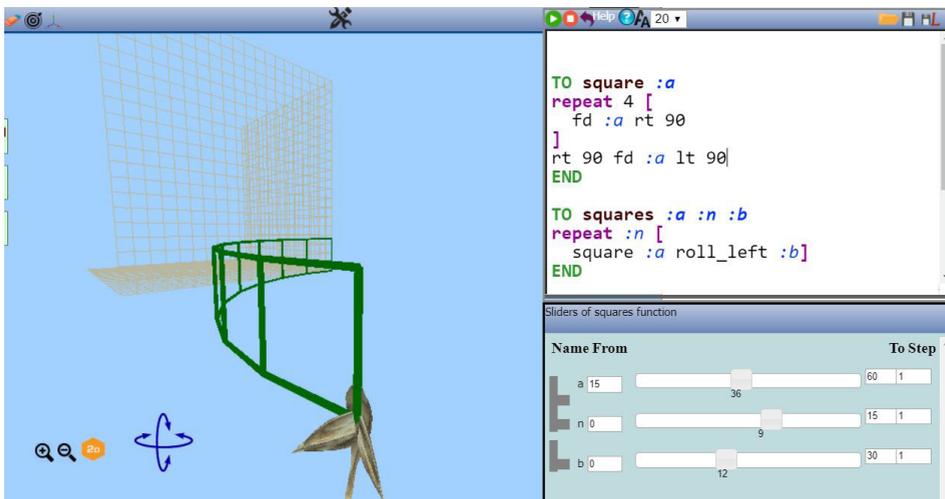


Fig. 1. Squares on a polygon curve.

tion, the user clicks on any model created by executing a parametric procedure with a set of values to its variables and activates the variation tool. This tool produces a number of sliders, one for every parameter of the procedure that created the clicked model. Each slider can be varied between an editable max and min limit. The variation of each slider, will result in an immediate re-execution of the procedure with the new input resembling a DGS-like dynamic behaviour. The aim of dynamic manipulation in MaLT2 is to reinforce the process of abstraction by means of kinaesthetically causing the continuous transformation of a structure described formally to make better sense of how this may represent a generality, such as e.g. a property of a geometrical figure.

- (3) The view becomes periscopic. The application allows navigation in 3D space where the avatar moves with a periscopic camera. The navigation in 3D space requires a number of skills like perceptual constancy (the ability to recognize some properties of an object regardless of its size, position or colour), spatial orientation (the ability to realize how an object would seem from a different point of view and recognize it even if it has been i.e. rotated or dispositioned), visual discrimination (the ability to compare multiple objects in space and recognize similarities and differences between them) and the perception of spatial relationships (Latsi and Kynigos, 2011; Lohman, 1988).

There are three basic elements that comprise the MaLT2 environment are: The “command editor”, the “3D scene” and the **uni-dimensional and bi-dimensional “variation tool”** (Fig.1). In the command editor, the user can write programs in order to navigate the avatar on the scene. The programming language of MaLT2 is based on MSW Berkeley Logo expanded with the feature of 3D graphical representations, including special commands for changing sparrow’s direction in the 3 dimensions (e.g. *up 90 and right roll 50*), importing 3D objects while moving in space (e.g. *cube 100*), changing pen colour and thickness (e.g. *setpenwidth 4*) etc. The 3D scene is where the sparrow moves as the commands are executed, with the same logic as Papert’s turtle moves and draws on a 2D canvas. The scene contains a periscopic camera, with which the learner can navigate in the 3D scene and observe the created models from different points of view. With the variation tool, the user can provoke dynamic changes to shapes created by the execution of a parametric command (e.g. *pyramid x w*). The tool provides a number of sliders, one for every parameter of the specific procedure, which can be varied between a max and min limit. The above features make MaLT a rich computational environment which extends turtle geometry beyond classic 2D designs and uniquely leads it to the modern world of online sharing, 3D graphics and animation.

Restructurations and Large-Scale Uses of MaLT

New representational media may provide different kinds of access to powerful ideas (diDessa, 2001) as well as new ways in which these ideas can be put to use and manifested as properties in the construction of digital objects. At ETL we resemble the

process of tinkering with a digital artefact, making it public in order for a discussion to take place leading to further changes to improve it, in other words considering an artefact as fallible to the process of doing mathematics described a long time ago by Imre Lakatos in his book ‘proofs and refutations’(1985). Lakatos claims that mathematical knowledge progresses when a mathematician considers themselves as an eternal learner and proposes a new idea, proof or lemma in order for peers to attempt to refute it. This process may end up in a refutation or not it doesn’t matter that much. What matters are the ideas generated and exchanged in the process. It is thus worthwhile to think outside the box with respect to traditional mathematics and computer science curricula and to come up with ideas rich in potential for meaning making. Not necessarily to question whether the concepts and structure of traditional curricula invite meaning making, but to look respectively for new ideas and idea clusters now enabled by digital media and the dynamic, constructionist representations within. In our research at ETL we have been using and elaborating four theoretical frames for action, helping us to design constructs for students to tinker with. It is necessary to briefly discuss these in order for the reader to be able to follow the subsequent study and ensuing analysis of students’ meanings.

The first is Willensky and Papert’s ‘restructurations’ (2010), a frame which is originally coined to describe the process of suggesting new ways and epistemologies to address issues like mathematical complexity through probabilistic thinking instead of looking for rules to fully explain the phenomenon. At ETL, we found this frame useful in our on-going attempts for look for constructions not easily imagined and represented with other means or constructions built with mathematics otherwise inaccessible to students. Consider for example the ‘biangle’ and the ‘twisted rectangle’ in Fig. 2 and Fig. 5. These are both figures not easily represented with traditional media and at the same time embed mathematics hardly accessible particularly to students with other means. With this medium, the ideas become relatively easy to grasp, involving simple curvature definitions in the first case and trigonometric functions in the latter.

The second frame is Vergnaud’s ‘conceptual fields’ (2009). Although this construct was coined a long time ago within the context of cognitive psychology, for us is it particularly powerful in our process of designing digital artifacts and projects for students to work with. Vergnaud suggested that in education, it is pointless to think about math-

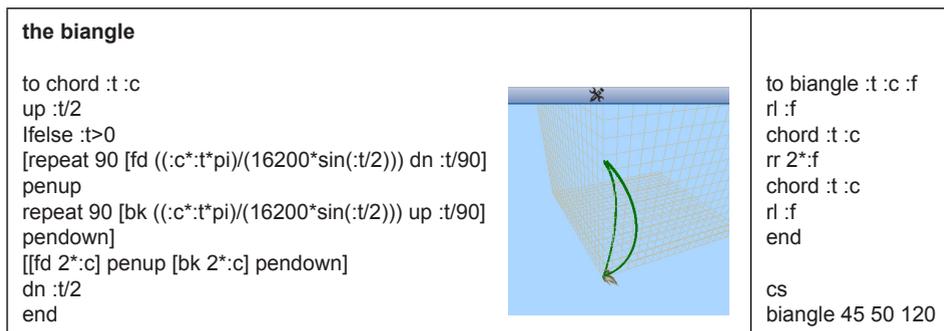


Fig. 2. A ‘Biangle on a sphere’.

ematical concepts in isolation with a silo approach. Each concept can be considered as lying at the centre of a conglomeration of tightly related concepts, a set of relevant representations and a set of problem situations where this concept plays a central role to a solution or an explanation. Depending on the problems at hand and the available representations, the same concept can be considered as being at the centre of diverse conglomerations of concepts. For instance, cosine can be thought as connected with right triangle properties or alternatively with periodic functions. A linear function can connect to real life problems, say, to do with shopping, or alternatively can be thought of as a relationship between angular values in a geometrical object like a parallelogram. In Fig. 3 for example, 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. In Fig. 4, angular properties of a geometrical figure become tightly connected to linear functions since the focus of the investigation is whether and what kind

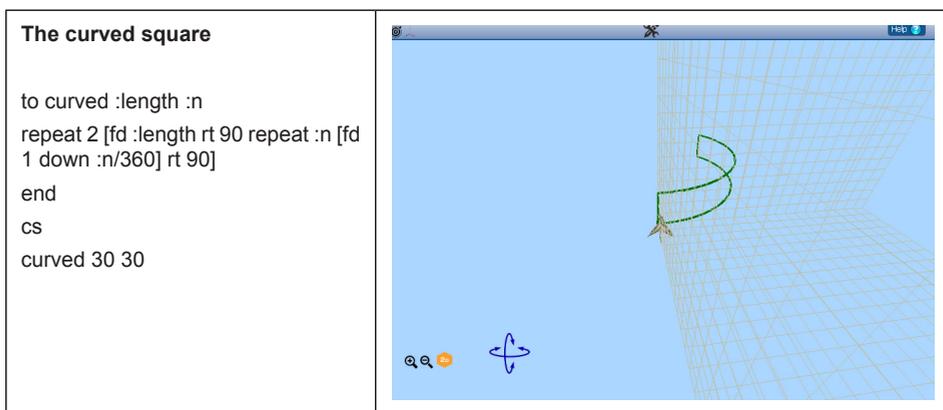


Fig. 3. A 'curved square'.

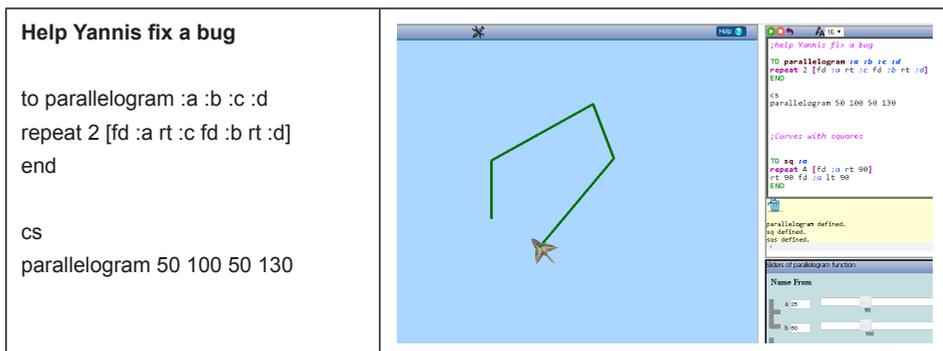


Fig. 4. Debugging a faulty parallelogram.

of relationship there is between consecutive turns for the sparrow to create a generalized rectangle on a plane.

Kynigos' 'engineered transparency' leading to black and white box designs, 'half-baked microworlds' and 'micro-experiments' (Kynigos, 2015; Kynigos, in press). In the constructionist community transparency of a digital medium is key in the sense that users have access to the powerful ideas with which the medium was constructed and are thus invited to construct from scratch and to re-construct any artefact build with the respective medium (Resnick *et al.*, 2000). However, lowering the threshold and having users begin building things with generic building blocks often ends up with them stuck on 'plateaus' (Noss and Hoyles, 1996), never having a sense of constructing something complex and interesting, their constructions thus only reach the level of simple games or postcards. Constructionism can nevertheless be considered as building with primitives of a diverse granularity. Such was the design idea for instance behind E-slate (Kynigos, 2004), where building blocks could be as complicated as whole pieces of software, ending up of example, with MaLT2' predecessor, E-slate Turtleworlds, comprised of simply joining up three distinct components in order to achieve the affordance integration discussed earlier. Half-baked microworlds are 'fallible artefacts' pedagogically engineered so as to have one or more bugs built into them (Kynigos, 2007). Students are asked to investigate, detect and correct the bugs and then to build something with their fixed artefact as a building block. In Fig. 4 for example, Yannis tried to construct a parallelogram and thought he had succeeded since this is the figure constructed with inputs of 100 30 50 150. Students are asked to inspect the code, identify the bug and fix it, thus looking deeply into the properties expressed with the buggy code. This piece of engineered transparency has been the design principle behind a large scale use of MaLT2 in the Greek Education system. In order to reach all teachers and their students, we coined the construct of a 'micro-experiment', rather than a microworld, to signify a much more focused question or problem comprising of a set of closed questions and only ending up with an open question leading to an investigation. The micro-experiment in Fig. 4 asks

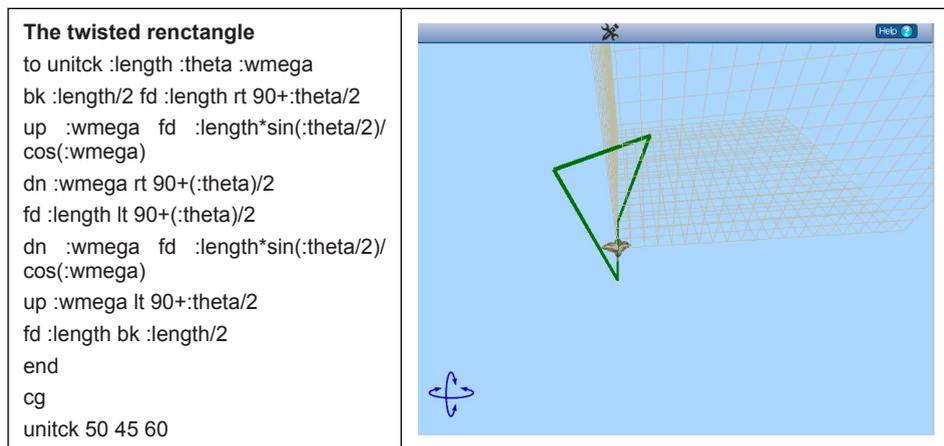


Fig. 5. The 'twisted rectangle'.

students to identify that there is a bug first by manipulating the sliders, then points them to the code to look for the bug posing some leading questions. In the end it invites them to build something of their own with the corrected parallelogram as a building block. There are 240 such artefacts amongst 1800 for mathematics in the Ministry of Education digital portal infrastructure (<http://dschool.edu.gr>), one of the most visited sites in the country.

Noss and Hoyles' (1996) perspective of reciprocal shaping of ideas between user and tool. In their foundational discussion of mathematical meaning-making with digital media Noss and Hoyles showed evidence of students' generation of mathematical understandings in connection to their interactions with a respective tool. They showed that a digital artefact originally designed to embed a mathematical idea invites students to generate meanings tightly connected to the activity they are engaged with in interaction with the tool. They called these 'situated abstractions'. At the same time, as students manipulate, tinker with and construct new models, the ideas originally embedded by the designers change and acquire an idiosyncratic specificity in relation to the ways the original artefact changes. This will be also shown in the description of the study which follows.

The Pilot Study

The integration of the above affordances can provide a new constructionist context for the development of computational thinking skills through the engagement with programming and mathematical concepts. To further investigate this we organized a study in a Greek junior high school. The aim of the study was to answer the following questions:

- How the integration of the three affordances with Logo programming contributes to students' development of computational thinking skills?
- What meanings do students generate about programming concepts while they collaboratively create animated models with the three affordances?

The total duration of the study was 9 hours divided into three 3-hour sessions and it was organized as part of the school's mathematical club. In the study participated 9 students aged 13–15 years with small previous experience in Logo programming, high experience and interest in mathematical problems and no previous experience in 3D geometry. The students worked collaboratively in 4 mixed gendered groups of 2–3 persons each, in the computer laboratory using one computer per group.

The research method we used is that of Design-Based Research which is evolved from design experiments (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. The research was carried out through repeated cycles of design and implementation, utilizing every implementation as an opportunity for data collection, evaluation and review for the next design. Design-Based Research focuses mainly on the collection and analysis of qualitative data as the objective is to identify the main characteristics and the different facets of the designed intervention when implemented with students.

Description of the Activities

For the purposes of the study we designed a set of activities with MaLT which were divided into three parts.

The first simply aimed at an introduction to the MaLT environment through simple Logo coding tasks. The first task was to program a 2D square with a variable size and manipulate it dynamically with the sliders tool. Then students were asked to use this square as a basis for programming a) a cube that would also have a variable size and b) a static cube net.

For the second part of activities students were asked to design and program a parametric cube model that would be able to animate from a 2D cube net to a cube with dynamic manipulation. There was no specific model of the cube net given to the students for this task. This activity was designed to expose students to ideas concerning the transition from two to three dimensions. At the centre of our design focus for instance was the integration of dihedral angle (the angle between two cube edges) and rotation in space. We also wanted to see if the students would engage in the computational process of generalisation and the role of parameters in procedural programming.

For the third activity students were asked to design a 3D animated drawing of their choice. In the beginning we provided the teams with three Logo procedures which they could use as building blocks for their drawings. The procedures were: a) **a procedure that created a cube with variable size named, cube :a**, b) **a procedure that created a pyramid with animated height and basis, named pyramid :x :w**, c) **a procedure that created a parallelepiped with animated height and width, named box :h :w**. **During this activity we were hoping that students would express and discuss their personal ideas about the computational and mathematical concepts used in the previous phases.**

Data Collection and Analysis

During the study, a set of data was collected for analysis that included screen capturing files, audio recordings of each group, teacher and researcher observation notes and files of students' code at the middle and the end of each session. A qualitative analysis method was followed for the evaluation of all the data. More precisely, the transcribed audio files and observation notes were analyzed in Atlas.ti tool with a set of codes related to a) the computational thinking skills of abstraction, pattern recognition, decomposition and algorithm design, b) mathematical and programming concepts of the activities (variable, parameter, angle etc) and c) students' interaction with the three affordances (i.e. slider use, creation of new function, camera rotation etc). From the analysis clusters of critical episodes emerged. As a critical episode, we consider representative moments of the student's interaction with MaLT affordances and of their engagement with CT skills. Moreover, we analyzed the Logo codes of students by comparing their progression from the first to the last session with respect to abstraction, pattern recognition and use of variables and repetition structures.

Findings

The analysis revealed that students applied the skills of abstraction, pattern recognition and decomposition especially during the second and third parts of the study. Table 1 shows a summary of how student activities were mapped to the four CT practices according to the results of the data analysis. All of the teams recognized and formed new patterns related to the 3D shapes both in the scene and in the Logo code. During the design process, they deconstructed 3D complex models to smaller parts or to 2D shapes (i.e. the cube to squares) which then used again to create other objects (i.e. a square pyramid). In addition they programmed and designed abstract models which, with the help of the sliders, could generate a number of successive instances creating an animation on the scene.

During the activities students used and discussed programming concepts, with the most common being the variable as a parameter, the procedures and sub-procedures and the error handling of different inputs. They made an extended use of the sliders tool for debugging their code but also for experimenting with the graphical outcomes of multiple parametric procedures on the scene. In this section, we present some characteristic examples from the analysis regarding both the emerging CT skills and also the applied programming concepts. In the presenting episodes we don't use the real names of the students but randomly selected aliases.

Uses of Abstraction

In the context of computational thinking, abstraction is described as the process of defining patterns, generalizing from specific instances and capturing of common characteristics or actions into one set that can be used to represent all other instances (Kafai, Bruke and Resnick, 2014; Krauss and Prottzman, 2017). Noss and Hoyles show how although this is a process of abstraction it is yet often situated, i.e. tightly connected to the activity and the tools at hand (Noss and Hoyles, 1996). In our case, this type of situated abstraction was detected mainly in the following cases:

- a) When students added a new parameter of animation to their drawing.
- b) When students used the sliders to animate a parameter of their model.

We posited that the creation of a parametric Logo procedure can be considered as computational abstraction, because it implements the generalized idea of the variable as

Table 1
Summary of student implementations of CT skills

CT Practice	How students implemented it
Pattern Recognition	Graphical & Coding patterns between 3D objects
Decomposition	Break down complex 3D structures
Abstraction	Animated models, Logo parametric (sub)procedures
Algorithm Building	Variable as parameter, repetition, procedural programming, debugging

parameter; a symbol (or word) that represents any numerical input. The students seemed to attribute to their use of the sliders some added value to the process of generalizing from a specific instance (a static model/non-parametric procedure) to an abstract model (animated model/parametric procedure).

One characteristic example occurred during part 2 when students were trying to construct a procedure for a dynamic cube net that would be able to transform from 2D to 3D space by dragging the sliders. At that moment all groups had already programmed non-parametric 3D cubes and 2D cube nets. Thus, they had to find a solution that would combine the 3D and 2D model to one by means of animating the cube net to transform into a cube.

The two students in group two, characteristically started by comparing the Logo procedures and graphic representations of the static 3D cube and the 2D cube net, aiming to identify common patterns and elements between them, the number and sequence of the squares and the size of the edges. They also observed that the two models differed at one command; the ‘up’ command that says to the sparrow to pitch up a number of degrees. As Helen mentioned “*we should add an up command right before every square so that the bird would turn up some degrees before it draws the next square*”. Helen and her teammate decided to modify the cube net procedure by adding the command ‘up 45’ before each square, so that it created a new, third instance (Fig. 6). In the dialogue below they explained this choice to the teacher.

T: Why did you write up 45?

H: Because like this we can see the model better. It is like in the middle of the animation

A: And then we will replace 45 with a variable so that the slider can take all the numbers for the animation.

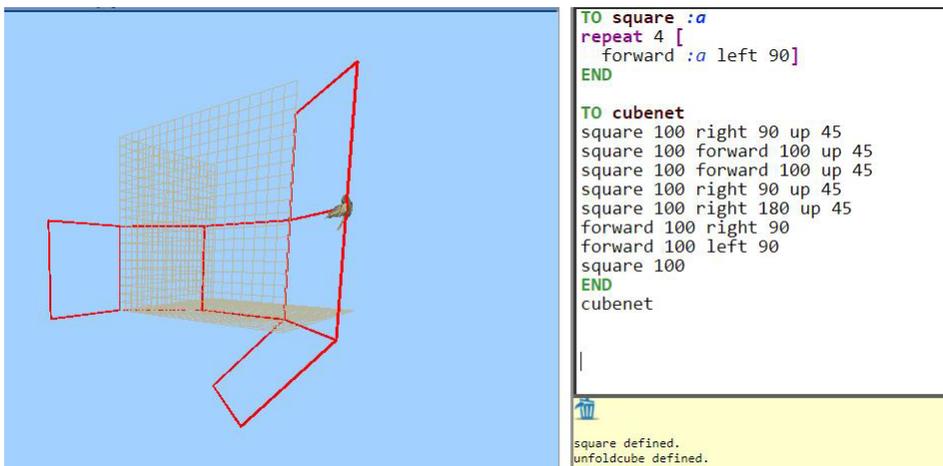


Fig. 6. The third instance of the cube model that uses the command up 45, created by group 2.

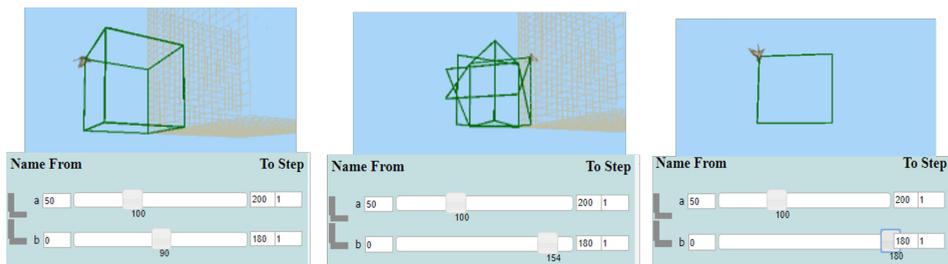


Fig. 7. Student experiments with the sliders led them to unexpected results.

Here, students expressed the idea of the animated cube as a set of cube instances that differ in one varying parameter, that of the dihedral angle. So they programmed representative static instances (0 degrees, 45 degrees and 90 degrees) which then, as they explained, they would generalize by replacing the number 45 with a variable, so that they can create an animation of all instances with dynamic manipulation.

The role of dynamic manipulation in the above process of abstraction was crucial. Students approached the cube as an object that has specific properties and they explored how it behaves when these properties are manipulated dynamically. In addition, they used the sliders to test and debug their code with unlimited number of inputs which led them to the conclusion that “the outcome of a parametric procedure must be correct for any possible input” (Helen’s quote to the teacher).

Not surprisingly, the dynamic manipulation affordance was used by the students to come up with unexpected models which they would not have been able to explore with other means. For instance, during the previous example, the two students experimented with dihedral angles greater than 90 degrees (Fig. 7). To their surprise they observed that for 180 degrees the cube model becomes again a 2D square without losing its properties, and then it ‘unfolds to the opposite direction.

Pattern Recognition

Pattern recognition as a skill of computational thinking is the ability to identify and match similarities (patterns) between different items as a way of gaining extra information. In our study students recognized and used patterns with respect to the objects’ properties and behaviours in the 3D space.

After Helen and Alice had finished with their dynamic cube net model their teacher asked them to decide what they wanted to do next.

H: “I was thinking to make a pyramid... I tried to create one at home but it was difficult with the angles etc”

A: “We could try to make the net of a pyramid first which is easier and then put a variable to fold it like we did with the cube”

In the above conversation, Alice found common patterns between the two 3D objects: their dihedral angle that transforms them from 2D to 3D space.

The presence of pattern recognition was more present during the third activity when students were given different 3D objects as building blocks to create a more complex 3D model. They saw patterns in the form of functional relations between parameters. We will use as an example a 3D house that was created by student group 3. For the creation of the procedure *house*, they used as building blocks the given procedures “*pyramid :x :w*”, with *x* being the size of pyramid base and *w* the pyramid height, and “*cube :a*”, where *a* was the size of cube edge.

At the beginning they used dynamic manipulation and camera perusal to identify common elements between the cube and the pyramid aiming to collate them in a common procedure (Fig. 8). For instance as it is presented in the dialogue of Table 2, they observed that the cube’s parameter *a* and pyramid’s parameter *w* should be the same variable in the new procedure because the base of the pyramid was attached to the top side of the cube. They also added a variable for the height of the house so that they could create a house of any size in their drawing.

In the above process, we can see the presence of both abstraction and pattern recognition skills. Students observed the common properties of cube and pyramid models (patterns) and then generalized them to the form of variables in order to create a relation



Fig. 8. Experimentation with the variation tool for the pyramid function.

Table 2
Dialogue and code of group 3 when they program their first parametric procedure of house

<p>G: What is the first parameter of the pyramid procedure? C: Move the sliders to see what each parameter does George moves the sliders of <i>pyramid</i> procedure C: So, the first <i>x</i> parameter is the base and <i>w</i> is the height G: Ok.. so... the pyramid’s base, which is parameter <i>x</i> must be the same with the side of the cube which is.. Christina looks at the Logo code of the non-parametric house C: Here it is <i>cube 70 and pyramid 70 50. They are both 70</i> G: Yes that’s what I’m saying. We have to use the same variable for both cube and pyramid. Lets name it <i>width</i>.</p>	<p>The 1st <i>house</i> parametric procedure of group 3 To <i>house1 :width :height</i> cube :width fd :width pyramid :width :height end</p>
--	--

between the two 3D models. This is particularly interesting. Instead of typically calling each procedure and giving it the same value of 70, they created a top-level variable for their house procedure and used it as input for both cube and pyramid procedures. In the presented dialogue George recognized the common elements by looking at the drawings in the 3D scene (“the pyramid’s base must be the same with the side of the cube”) and Christina **concluded the same by looking only at the code. This indicates that the exploration of a computational model with diverse affordances may enhance abstract thinking and the recognition and implementation of patterns.**

Patterns as Functional Relations between Parameters

An interesting outcome during the activity described above, was the creation of patterns through the programming of functional relations between parameters of different functions. For instance, George and Christina of group 3 (see above), decided to extend their house by connecting the parameters of the sub-procedures pyramid and cube with a functional relation.

C: Oh! What about instead of having two parameters in the house to have only the width one? And then we call the pyramid with both inputs as :width?

G: Or...Maybe the one :width and the other :width/2

*C: Yes, and then the roof would be related to the cube like...the double of it width*2 or the half width/2, width/3 etc ... And it would make a cool animation!*

G: And if we do this we can create a series of different houses for our neighborhood

In the above dialogue, the two students expressed their ideas around the concept of parameter in procedural programming. They seemed to situate the concept of passing variables as inputs between procedures and how variables maintain their numerical value through the sub-procedures.

Based on this idea, they programmed four house procedures that had the exact same code with the only difference being the parametric inputs they passed to the sub-procedure *pyramid* (Fig. 9). Then they used the four procedures to create a neighborhood the houses of which followed a specific pattern i.e. each house would have half the height of the previous one.

The role of both dynamic manipulation and 3D navigation to that process was crucial as it supported the direct experimentation with the graphical outcome of different functional relations. It also urged students to come up with and test various ideas about possible relations between two parameters and led them to the construction of creative patterns.

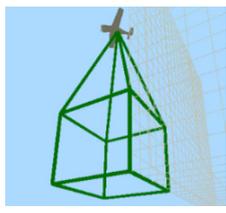
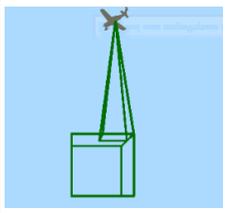
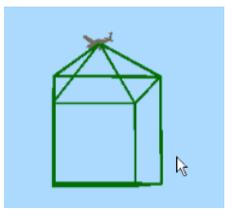
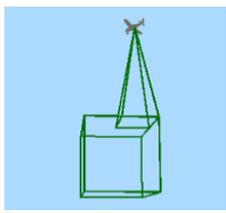
			
<p><i>2 Variables</i></p> <pre>To house1 :width :height cube :width pyramid :width :height end</pre>	<p><i>2 Variables</i></p> <pre>To house2 :width height cube :width pyramid :width/4 :height end</pre>	<p><i>1 Variable</i></p> <pre>To house3 :width cube :width pyramid :width : width/2 end</pre>	<p><i>1 Variable</i></p> <pre>To house4 :width cube :width pyramid :width/4 :width end</pre>

Fig. 9. Functional relations between parameters for the creation of four different house models.

Discussion and Conclusion

The analysis of the data showed that by using the three affordances of MaLT in conjunction, the students applied computational thinking skills in an authentic context and connected them with different interdependent representations. For instance, during the design of the animated cube net, they engaged with the concept of abstraction not only in their code (replacement of constants with variables) but also by imagining the animation of the 3D model with the sliders. They themselves thought of their task as to create a procedure where by moving any slider the model should animate smoothly between the instances of a cube. The dynamic manipulation feature of the sliders offered a strong tool for code debugging, as it allowed students to experiment with an unlimited number of inputs and immediately connect the algorithm result to the real-time animated model. From the point of view of task and artifact design, we gave them two static procedures which we thought embedded the need for them to understand dihedral angles. We observed the students consider the net as a model which could change to allow for animation, shaping the embedded ideas to integrate variable as a tool for animation.

Furthermore, we observed some students' meanings generated through the use of the tools we gave them and the specific problems at hand. For instance, they perceived and built a pyramid by building a 2-dimensional net of a pyramid first. After noticing that the values of two distinct variables should be the same, they created a super-procedure executing the respective sub-procedures with the same variable name, passing the variable at the higher procedure level instead of just providing an equal numerical input. This led them to later try out procedures with functional relations between variables handling generalized number as an object to think with. We argue that it was the combination of thinking outside the box with respect to traditional curricula in our designs, creating conceptual fields dense in opportunities for meaning making and keep-

ing an open mind to support students to generate their own meanings which resulted in these unexpected thoughts expressed and put to action by our students. The design for and study of meanings related to computational thinking and programming skills could thus draw from our growing knowledge from research in mathematics education and vice-versa.

New mathematical ideas and old ideas newly accessible to students should be a central facet of research in the design of expressive digital media. For this to come about, the exercise of restructuration and the building of conceptual fields not necessarily attached to traditional curricula is key and the process of looking for such new artefacts as in Fig. 2–Fig. 5 and then pedagogically engineering them to set students off with the right kind of focus is on-going in our team at ETL.

However, apart from the use of variables as parameters, there were limited results related to other programming concepts such as iteration or conditionals. In addition, even though students used programming concepts in all of the three activities, the majority of critical episodes related to CT skills emerged during the third activity in which students freely designed their personal 3D drawing. This fact indicated the need for including more open and creative activities to our next study. This is also an interesting outcome for other designs and activities that aim to reinforce students CT skills and reinforces the limitations of closed tasks and quizzes. Our future plans include a large-scale study which will focus on the emerging CT skills from students' engagement in a variety of activities with MaLT.

We believe that the integration of different affordances can be a strong vehicle for the development of CT but also a new way to combine programming and mathematics in a meaningful context. The results of this study may inform other integrated designs, contributing to the ongoing seek of strategies for promoting Computational Thinking for all students.

Acknowledgements

This research has been financially supported by the General Secretariat for Research and Technology (GSRT) and the Hellenic Foundation for Research and Innovation (HFRI) (Scholarship Code: 531)

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007–2013) under grant agreement n° 610467 – project “M C Squared”, <http://mc2-project.eu>. The c-book technology is based on the widely used Freudenthal Institute's DME portal and is being developed by a consortium of nine partner organizations, led by CTI&Press 'Diophantus'.

This work has been supported by the Greek National project “Digital School Platform, Interactive Books, and Learning Object Repository” (Contract No 296441/2010–2015) that is co-financed by the European Union (ESF) and National funds in the context of Operational Programme “Education and Lifelong Learning” of the Greek National Strategic Reference Framework (NSRF), and is being implemented by CTI “Diophantus”.

References

- Abelson, H., DiSessa, A. (1981). *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. Cambridge, MA: MIT Press.
- Barr, V., Stephenson, C. (2011). Bringing computational thinking to K-12: what is Involved and what is the role of the computer science education community?. *Acm Inroads*, 2(1), 48–54.
- 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, Vancouver, Canada*. 1–25.
- Clements, D.H., Battista, M.T., Sarama, J., Swaminathan, S. (1997). Development of students' spatial thinking in a unit on geometric motions and area. *The Elementary School Journal*, 98(2), 171–186.
- Cobb, P., Confrey, J., diSessa, A., Lehrer, P., Schauble, L. Design. (2003). Experiments in educational research. *Educational Researcher*, 32(1), 9–13.
- diSessa, A. (2001). *Changing Minds: Computers, Learning and Literacy*. USA: MIT Press.
- Goldenberg, E.P., Cuoco, A. (1998). What is Dynamic Geometry? In: R. Lehrer, D. Chazan (Eds.), *Designing Learning Environments for Developing Understanding of Geometry and Space* (351–368). Mahwah, NJ: Lawrence Erlbaum Associates.
- Harvey, B. (1997). *Computer Science Logo Style*, MIT Press.
- Hu, C. (2011). Computational thinking: what it might mean and what we might do about it. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (pp. 223–227). ACM.
- Kafai, Y.B., Burke, Q., Resnick, M. (2014). *Connected Code: Why Children Need to Learn Programming*. MIT Press.
- Krauss, J., Prottsman, K. (2017). *Computational Thinking and Coding for Every Student: The Teacher's Getting-Started Guide*. Corwin Press.
- Kynigos, C. (1995). Programming as a Means of Expressing and Exploring Ideas in a Directive Educational System: Three Case Studies. In diSessa, A., Hoyles, C. and Noss, R. (Eds) *Computers and Exploratory Learning*, Springer Verlag NATO ASI Series, 399–420.
- Kynigos, C. (2004). A Black and White Box Approach to User Empowerment with Component Computing, *Interactive Learning Environments*, Carfax Pubs, Taylor and Francis Group. 12(1–2), 27–71.
- Kynigos, C. (2007). Half-baked Logo microworlds as boundary objects in integrated design, *Informatics in Education*, 6(2), 1–24.
- Kynigos, C. (2015). Constructionism: Theory of learning or theory of design? In: Sung Je Cho (Eds), *Selected Regular Lectures from the 12th International Congress on Mathematical Education*. Springer International Publishing Cham Heidelberg New York Dordrecht London, Switzerland, 417–438.
- Kynigos, C., Latsi, M. (2007). Turtle's navigation and manipulation of geometrical figures constructed by variable processes in a 3d simulated space. *Informatics in Education*, 6(2), 1–14.
- Kynigos, C. (in press). The Challenge of Infusing Constructionism in Education in Greece. In: Holbert, N., Berland, M., Kafai, Y. (Eds), *Constructionism in Context*. MIT Press.
- Laborde, C., Kynigos, C., Hollebrands, K., Strasser, R. (2006). Teaching and learning geometry with technology. In: A. Gutiérrez, P. Boero (Eds.), *Handbook of Research on the Psychology of Mathematics Education: Past, Present and Future*. Boston: Sense Publishers, 275–304.
- Lakatos, I. (1976). *Proofs and Refutations: The Logic of Mathematical Discovery*. New York: Cambridge University Press.
- Latsi, M., Kynigos, C. (2011). Meanings about dynamic aspects of angle while changing perspectives in a simulated 3d space. In: *Proceedings of the 35th Conference of the International Group for the Psychology of Mathematics Education* (Vol. 3). Ankara, Turkey: PME, 121–128.
- Lee, I., Martin, F., Denner, J. et. al. (2011). Computational thinking for youth in practice *Acm Inroads*, 2(1), 32–37.
- Lohman, D. (1988). Spatial abilities as traits, processes and knowledge. In: R.J. Sternberg (Ed.), *Advances in the Psychology of Human Intelligence* (Vol. 4). Hillsdale, NJ: LEA.
- Noss, R., Hoyles, C. (1996). *Windows on Mathematical Meanings: Learning Cultures and Computers*. Dordrecht: Kluwer.
- Papert, S., Harel, I. (1991). Situating Constructionism. In: Papert, S, Harel, I. (Eds), *Constructionism* Norwood, NJ: Ablex Publishing.
- Papert, S. (1972). Teaching children to be mathematicians versus teaching about mathematics.
- Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

- Psycharis, G., Kynigos, C. (2009). Normalising geometrical figures: Dynamic manipulation and construction of meanings for ratio and proportion. *Research in Mathematics Education*, 11(2). 149–166
- Reggini, H.C. (1985). *Ideas y formas: Explorando el Espacio con Logo*. Buenos Aires: Galápago
- Resnick, M. (2014). Give P's a chance: Projects, peers, passion, play. In: *Constructionism and Creativity: Proceedings of the 3rd International Constructionism Conference*. Austrian Computer Society, Vienna, 13–20.
- Resnick, M., Berg, R., Eisenberg, M. (2000). Beyond black boxes: Bringing transparency and aesthetics back to scientific investigation. *Journal of the Learning Sciences*, 9(1), 7–30.
- Sarama, J., Clements, D. (2002). Design of microworlds in mathematics and science education. *Journal of Educational Computing Research*, 27(1), 1–3.
- Sinclair, K., Moon, D. (1991). The philosophy of LISP. *Communications of the ACM*, 34(9), 40–47.
- Stager, G. (2014). This is Our Moment In: *Constructionism and Creativity: Proceedings of the 3rd International Constructionism Conference*. Austrian Computer Society, Vienna.
- Wilensky, U., Papert, S. (2010). Restructurations: Reformulations of knowledge disciplines through new representational forms. In: *Proceedings of the Constructionism 2010 Conference*. Paris, France, 97.
- Vergnaud, G. (2009). The theory of conceptual fields. *Human Development*, 52, 83–94. DOI: 10.1159/000202727.
- Weintrop, D., Wilensky, U. (2015). To block or not to block, that is the question: students' perceptions of blocks-based programming. In: *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 199–208.
- 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: Mathematical, Physical and Engineering Sciences*, 366(1881), 3717–3727
- Zantzou I., Kynigos C. (2012). Differential approximation of a cylindrical helix by secondary school students. In: Kynigos, Clayton, Yiannoutsou (Eds.), *Proceedings of the 'Constructionism 2012' International Conference*. Athens, 136–145.

C. Kynigos is professor at the National and Kapodistrian University of Athens and director of the Educational Technology Lab at the School of Philosophy, <http://etl.ppp.uoa.gr>. He teaches courses in educational technology and mathematics education at undergraduate and postgraduate level. He had led the design and development of two programmable web-based authoring systems, MaLT2 and ChoiCo. MaLT2 is a Logo-based 3D turtle geometry system integrated with dynamic manipulation tools and camera perusal. ChoiCo is a game designer based on a choice with consequences paradigm addressing complex multifaceted socio-scientific issues. His research is on practices generated by innovative interventions in education based on a socio-constructionist paradigm. He has led 3 and taken part in 8 European related design and development projects and has been an active member of the Constructionist community since 1989.

M. Grizioti is a PhD student in the field of computational thinking through programming and modifying digital games at National and Kapodistrian University of Athens and a member of the Educational Technology Lab (<http://etl.ppp.uoa.gr>). She has a bachelor degree in Computer Science and a Masters of Education in digital technologies in education. Her research interests include Computational Thinking, ICT education, design of educational software, game-based learning and child computer interaction.