Proceedings of the Informing Science + Information Technology Education Conference

An Official Publication of the Informing Science Institute
InformingScience.org

InformingScience.org/Publications

June 30 – July 4, 2019, Jerusalem, Israel

INTRODUCING COMPUTATIONAL THINKING PRACTICES IN LEARNING SCIENCE OF ELEMENTARY SCHOOLS [RESEARCH-IN-PROGRESS]

Gilad Shamir*	Tel Aviv University, Tel Aviv, Israel	giladshamir@mail.tau.ac.il
Dina Tsybulsky	Technion, Haifa, Israel	dinatsy@ed.technion.ac.il
Ilya Levin	Tel Aviv University, Tel Aviv, Israel	ilia1@tauex.tau.ac.il

*Corresponding author

ABSTRACT

Aim/Purpose	Science is becoming a computational endeavor therefore Computational Thinking (CT) is gradually being accepted as a required skill for the 21st century science student. Students deserve relevant conceptual learning accessible through practical, constructionist approaches in cross-curricular applications therefore it is required for educators to define, practice and assess practical ways of introducing CT to science education starting from elementary school.
Background	Computational Thinking is a set of problem-solving skills evolving from the computer science field. This work-in-progress research assesses the CT skills, along with science concepts, of students participating in a science program in school. The program pertains learning science by modeling and simulating real world phenomenon using an agent-based modeling practice.
Methodology	This is an intervention research of a science program. It takes place as part of structured learning activities of 4th and 5th grade classes which are teacher-guided and are conducted in school. Both qualitative and quantitative evaluations are parts of the mixed methods research methodology using a variety of evaluation technique, including pretests and posttests, surveys, artifact-based interviews, in class observations and project evaluations.
Contribution	CT is an emerging skill in learning science. It is requiring school systems to give increased attention for promoting students with the opportunity to engage in CT activities alongside with ways to promote a deeper understanding of science. Currently there is a lack of practical ways to do so and lack of methods to

Accepting Editor: Eli Cohen | Received: December 16, 2018 | Revised: February 9, 2019 | Accepted: February 10, 2019.
Cite as: Shamir, G., Tsybulsky, D. & Levin, L. (2019). Introducing computational thinking practices in learning science of elementary school [Research-in-Progress]. *Proceedings of the Informing Science and Information Technology Education Conference*, *Jerusalem, Israel,* pp. 187-205. Santa Rosa, CA: Informing Science Institute.
https://doi.org/10.28945/4327

	assess the results therefore it is an educational challenge. This paper presents a response to this challenge by proposing a practical program for school science courses and an assessment method.
Findings	This is a research in progress which finding are based on a pilot study. The researches believe that findings may indicate improved degree of students' science understanding and problem-solving skills.
Recommendations for Practitioners	Formulating computer simulations by students can have great potential on learning science with embedded CT skills. This approach could enable learners to see and interact with visualized representations of natural phenomena they create. Although most teachers do not learn about CT in their initial education, it is of paramount importance that such programs, as the one described in this research, will assist teachers with the opportunity to introduce CT into science studies.
Recommendation for Researchers	Scientific simulation design in primary school is at its dawn. Future research investment and investigation should focus on assessment of aspects of the full Computational Thinking for Science taxonomy. In addition, to help teachers assess CT skills, new tools and criteria are required.
Impact on Society	STEM related professions are lacking the man power required therefore the full potential of the economy of developed countries is not fulfilled. Having students acquire computational thinking skills through formal education may prepare the next generation of world class scientists and attract larger populations to these fields.
Future Research	The inclusion of computational thinking as a core scientific practice in the Next Generation Science Standards is an important milestone, but there is still much work to do toward addressing the challenge of CT-Science education to grow a generation of technologically and scientifically savvy individuals. New comprehensive approaches are needed to cope with the complexity of cognitive processes related to CT.

Keywords	computational thinking, science education, agent-based modeling and simulation, computational science, computational practices

BACKGROUND

The importance of simulating models goes back to Aristotle who claimed that imitation is a means to know nature through representations which can be valid and acceptable. In fact, Aristotle introduced the cognitive term phantasy as the necessary intermediary between the senses (particularly vision) and the intellect. Along these lines, imagination with regards to scientific learning can be defined as the disciplined and informed use of mental simulation for envisioning a system's behaviors and drawing testable inferences (Landriscina, 2014).

However, there are the difficulties that students might encounter in the process of building, simulating and updating their own mental models. This frequently occurs when students must mentally integrate multiple and dynamically changing representations of information, while carrying out complex tasks, such as with scientific hypotheses. Usage of models and simulations has facilitated science learning in which students were taught to use computational simulations (Lee, Martin, & Apone, 2014). The release of the Next Generation Science Standards (NGSS, 2013) has opened the gate to a new form of interaction between students and computational simulations. This time not as mere users rather also as simulation creators (Sneider, Stephenson, Schafer, & Flick, 2014).

NGSS introduced a new challenge to science teachers. Out of the eight NGSS practices one stands out - the practice of "using computational thinking". With it comes science teacher's uncertainty of how to integrate it into existing classroom routines. With the growing importance of computation in science, it seems appropriate that there will be a curriculum accompanied by teaching methods that coincide with the emergence and use of creational technologies which are those that enable to create additional technologies. Practices collected under the term "Computational Thinking" have been displayed in a detailed CT-Science taxonomy (Weintrop et al., 2016). Nevertheless, their practical implementation in terms of educational pedagogy and science content in the classrooms is still vague.

Beyond the inclusion of CT as a central scientific practice, as defined by the NGSS, there are other important reasons to introduce CT-Science. CT includes a set of skills that are applicable to a broad range of problems and settings. While this is a strength of the skills, it also presents challenges to teaching them, as they are not tied to a specific domain. By fusing CT instruction with science discipline, students can explore and apply CT skills within a more established and accessible science context. In this way, science can enhance CT learning. Research has also suggested that the reverse is true; CT and the use of computational tools has been shown to enable deeper learning of science content areas for students (National Research Council, 2011; Wilensky & Reisman, 2006).

Another benefit of embedding CT in STEM classrooms is that it fosters CT reaching a wider audience than would be possible if it was taught independently. One reason for that is since all schools are having courses covering science discipline. In contrary, as of 2018 in Israel only 3% of the students learn computer science (CS) in elementary schools. This is largely due to lack of resources, mostly lack of teaching hours which are historically allocated to other subject matters. Programming is a common way to introduce CT skills to students but 97% are not exposed to CS therefore are not exposed to CT skills learning. By embedding CT in STEM curricula issues of schools lacking the resources to have a course dedicated solely to CT or even to CS is addressed. Finally, embedding CT in STEM course can address the issues of practicality of implementation, especially with teachers' comfort with the material. In this approach, the CT skills that are new to teachers are embedded within concepts that teachers already have mastery over, instead of requiring the teachers to learn entirely new concepts.

A facilitator for introducing CT-Science courses is the software development environments which increasingly are becoming easier to use and are created with elementary school students in mind. Researchers see software development as a mean to promote CT with science studies (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013). One case of programming is simulations construction using an Agent-based modelling and simulation (ABMS) paradigm. ABMS is an approach to modelling systems composed of interacting, autonomous agents. The agents have behaviors, often described by simple rules, and interactions with other agents, which in turn influence their behaviors. By modelling agents individually, the full effects of the diversity that exists among agents in their attributes and behaviors can be observed as it comprises the behavior of the system (Macal & North, 2010). By modelling systems from the 'ground up', agent-by-agent and interaction-by-interaction, self-organization can often be observed in the models. Patterns, structures, and behaviors emerge that were not explicitly programmed into the models but arise through the agent interactions. The emphasis on modelling the heterogeneity of agents across a population and the emergence of self-organization are two of the distinguishing features of agent-based simulation as compared to other simulation techniques such as discrete-event simulation and system dynamics.

A computation-based approach using ABMS enables students to investigate the connections between different phenomenon levels. Using agent-based modeling tools, students model the micro-rules underlying a science phenomenon and observe the result dynamics. Since this simulation design and implementation activity requires learners to design and make a simulation themselves, not just learn from an existing ready-made one, then this approach fits well within the constructionism theory known to be highly a motivator for students learning (Papert, 1980).

CT-Science can contribute to learning in an additional form as well. There is a sharp contrast between the picture of the field of science as studied in school settings and the picture that emerges from the practice of current research. Although the two pictures are linked by similar content and the objects of study are recognizably the same. With school settings, typical instruction emphasizes the memorization of classification schemas and established theories (Jona el al, 2014). Even when students are exposed to research techniques in laboratory work, the emphasis is on following a prescribed procedure rather than reasoning from the evidence gathered in the procedure. This picture contrasts sharply with the picture that emerges from science research literature (Keeling & Gilligan, 2000; Marion, Renshaw, & Gibson, 2000). In this picture, the participants are active theorizers. They gather new evidence and devise methods to test their theories. Students form beliefs that science is a discipline in which observation and classification dominate and reasoning about theories is rare. Furthermore, they believe that learning biology consists of absorbing the theories of experts and that constructing and testing their own theories is out of reach.

In recent years, several educational research projects (Lehrer & Schauble, 2000) have used computer-modeling tools in science instruction. The approach taken herein differs from these approaches in its use of agent-based modeling paradigm that enable students to model singular elements opposed to aggregates modeling paradigms. This technical advance in modeling paradigms enables students to use their knowledge of the behavior of individual entities in the construction of theories about the behavior of a phenomenon.

There are a few programs aiming to teach Science with inherent CT:

- EcoScienceWorks (ESW) (Stelar, n.d.) is a program in Maine, USA, engaging students with environmental simulations as part of ecology topic in the science curriculum. This project exposes students to simple programming challenges as a way of introducing them to the computational thinking that underlies the simulations. Through guided experimentation, ESW deepens students understanding of both ecology and computer modeling. Contrary, the program described in this research emphasizes a full model and simulation design and implementation process therefore differs from ESW and differs in the CT practices each foster.
- CTSiM agent-based modeling environment uses an expert-based modeling method in which the simulation results are compared to an existing expert's model such as using overlays. The advantage of this method is that the end project can be validated. The disadvantages are that this hides a possible problem if the expert`s model has a flaw (Weintrop, 2016) or when there can be several different correct solutions (e.g., different computational models in CTSiM). This results in big behavioral differences between individuals, which means that an expert model is insufficient for capturing the learning behaviors of all learners (Dong, 2018). The environment described in this research emphasizes a fully uncontrolled simulation process creation therefore differs from CTSiM.
- "Middle school CS in science" of the Code.org organization is yet another program for modeling and simulation in science learning. It consists of instructional modules and professional development for the introduction of computer science concepts into science classrooms within the context of modeling and simulation. CS in Science is based on a crosswalk identifying areas of overlap between the NGSS and Computer Science Teachers Association K-12 Computer Science Standards. The difference between CS in science program and the program described by the research is that the former focuses on CS in middle schools while the latter focuses on CT in elementary school.

SCRATCH AS AN ABMS DEVELOPMENT ENVIRONMENT

The researchers selected a programming environment for the ABMS creation student activity - It is Scratch of MIT. Scratch can be described as a general-purpose, domain-independent, modeling lan-

guage and integrated development environment (IDE). It works by providing the modeler with a framework to represent the basic elements, the smallest parts, of a system and then provides a way to simulate the interactions between these elements. (See the Appendix for a list of elements.)With scratch, students write rules for few of these basic elements, specifying how they should behave and interact with one another. These individual elements are referred to as Sprites. They can also be referred to as agents or in programming language terms they are 'Classes'. Sprites can be instantiated having multiple instances of a specific Sprite. Sprite instances (SI) are situated on a two-dimensional grid called Stage on which they can move around. Some typical commands for a SI are move in a given direction, change color, set a variable according to some value, clone new SI, or broadcast message to other Sis and react on messages received from other SIs. SIs can also generate random values, so that they can, for example, execute a sequence of commands with a fixed probability.

The wide range of commands executable by SIs makes it possible to use them to represent many different systems. Dynamic modeling tools, such as scratch, are used to represent changes in the states of systems over time. In Scratch, time is represented as a discrete sequence of "clock-ticks." At each clock-tick, each SI and stage is called on to execute the rules that have been written for it. Students need not write separate rules for each sprite. A power of Scratch comes from the fact that all SIs can execute the very same set of rules at each clock-tick. If all SIs are executing the same rules, will their collective behavior not be repetitive and uninteresting? To see why this is not the case, it is important to take note of the fact that even though two SIs might be following the same rules, their behavior could be markedly different. This possibility exists because the two SIs may have quite different internal properties and may be situated in different environments. For example, the SIs may be following the rule "If you touch a consumable SI, consume it and move forward. Otherwise, turn around." If one SI is in the vicinity of another sprite, it will consume it and move forward, the other SI, far from the consumable SI, will turn around. It is this diversity in internal states and in surrounding environments that enables the collective SI behaviors to admit a surprising degree of variance. The modeling approach we describe—instantiating the individual elements of a system and simulating their interactions—is not unique to Scratch.

The Scratch environment was chosen for several reasons. It is the preferred programming environment by the ministry of education in Israel for CS studies in elementary school (MOE, 2016). It is used in teacher's professional development therefore students use it for CS studies. Scratch Support encapsulates low-threshold (i.e., easy to program), wide walls (i.e., students should be able to design a wide range of artifacts) and has multi lingual support (for Hebrew or Arabic speaking schools).

Computational Thinking in Science Taxonomy

The frameworks used by the researchers as a basis for the study include first and foremost the Computational thinking in science taxonomy (CT-Science) (Weintrop, 2016) which defines and describes the set of CT practices relevant to science which students are required to learn. (See Figure 1.)

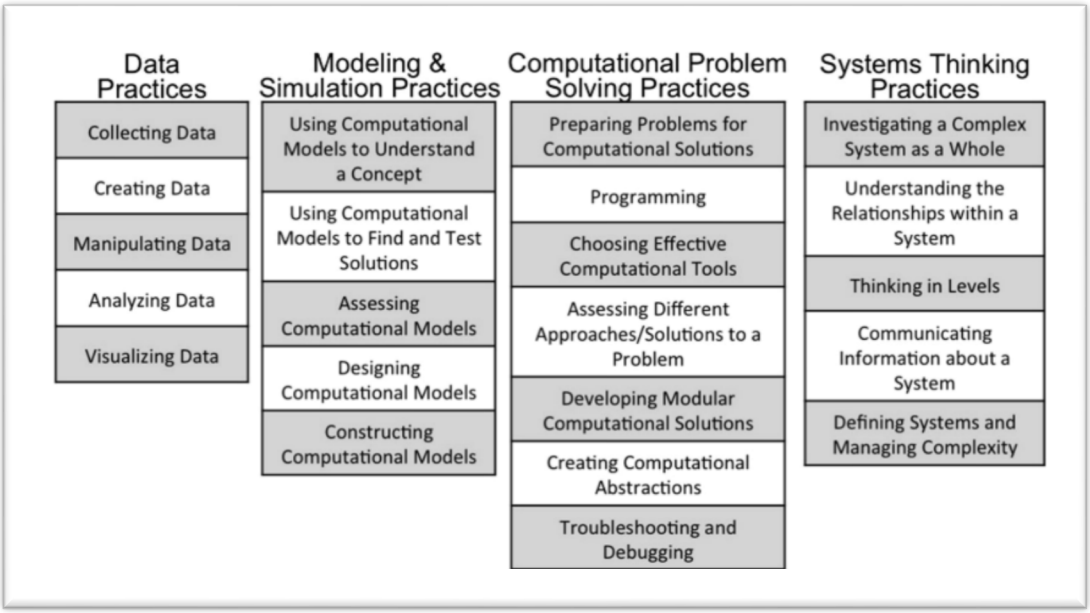


Figure 1. CT-Science taxonomy

The focus of this research is on constructionist learning of CT by creating scientific models and simulations as opposed to a traditional teacher approach of using simulations, meaning students will create simulations rather than use them. Therefore, the researchers chose to focus on a subset of 3 practices described here forth:

Preparing Problems for Computational Solutions – A Problem Solving practice

While some problems naturally fit to computational solutions, more often, problems must be re-framed so that existing computational tools such as software environments can be utilized. In the sciences, a vast array of computational tools can be employed for a given pursuit; the challenge is to map problems onto the capabilities of the tools. Strategies for doing this include decomposing problems into subproblems, reframing new problems into known problems for which computational tools already exist, and abstracting which is simplifying complex problems so the mapping of problem features onto computational solutions is more accessible. Students who have mastered this practice will be able to employ such strategies toward reframing problems into forms that can be solved.

Designing Computational Models - A Modeling and Simulations practice

Part of taking advantage of computational power in the scientific disciplines is designing new models that can be run on a computational device. The process of designing a model is distinct from implementing it. Designing involves making technological, methodological, and conceptual decisions. There are many reasons that might motivate designing a computational model, including wanting to better understand a phenomenon under investigation, to test out a hypothesis, or to communicate an idea or principle to others in a dynamic, interactive way. When designing a computational model, one is confronted with a large set of decisions including defining the boundaries of the system, deciding what should be included and what can be ignored, and conceptualizing the behaviors and properties of the elements included in the model. Throughout the design process, one must ensure that the resulting model will be able to accomplish the goal that initially motivated the model design process. Students who have mastered this practice will be able to design a computational model, a process

that includes defining the components of the model, describing how they interact, deciding what data will be produced by the model, articulating assumptions being made by the proposed model, and understanding what conclusions can be drawn from the model.

CONSTRUCTING COMPUTATIONAL MODELS – A MODELING AND SIMULATIONS PRACTICE

Note, While the CT-Science taxonomy names the program created a `model`, the researchers of this study use the term `simulation` for the constructed artifact which implements the model.

An important practice in scientific pursuits is the ability to create new or extend existing computational simulations. This requires being able to encode the model features in a way that a computer can interpret. Sometimes this takes the form of conventional programming. Being able to implement modeling ideas is critical for advancing ideas beyond the work done by others and complements the previous practice of designing computational models. Students who have mastered this practice will be able to implement new simulation behaviors, either through extending an existing simulation or by creating a new one.

RESEARCH GOALS & QUESTIONS

Computational thinking (CT) for science studies was added to students` required skills as part of NGSS definitions (NGSS, 2013). Following that, a CT for science taxonomy was created (Weintrop et al., 2016). Two of the taxonomy's categories are "Modeling and Simulation Practices" and "Computational Problem-Solving Practices" which are the focus of this research. The method of acquiring those practices is another emphasis of this research - a genre of computational programming and modeling called Agent-Based Modeling and Simulation (ABMS). In the agent-based paradigm, the student programs the behaviors of one or more agents which are computational actors, by using rules, which are then executed or simulated in steps over time to generate an evolving set of behaviors.

The researchers of this study have created a 360 approach for an intervention of embedding CT in science based on using a constructionist ABMS paradigm. It includes 1) Teacher training on ABMS principals within the context of the Scratch software development environment. 2) Creation of learning and teaching materials for elementary school students .3) An assessment tool for validating the students CT practices. Accordingly, the research questions are:

1. What are the reported outcomes of the intervention learning program? Specifically, its effect on the students CT practices called "Modeling and Simulation" and "Computational Problem-Solving"?

2. What gains in the relevant science curricular goals had the intervention learning program achieved?

METHODOLOGY

This paper describes an intervention research in elementary school settings. It is consisting of a school program focusing on students' acquired computational thinking skills and science practices along with an assessment tool. The researchers of this study have created a curriculum which embeds interdisciplinary learning of CS, CT and science based on using a constructionist approach which consist of creation of an Agent-Based Modeling and Simulation (ABMS) by the students.

Data was collected and analyzed from using several approaches:

1) Comparison of pre-course and post-course questionnaires about science.

2) Post-course self-descriptive questionnaire.

3) Classroom observations.
4) Artifact based interviews held with a selection of interested students.
5) An analysis of the ABMS projects the students have created.

Both qualitative and quantitative evaluations are essential parts of the mixed methods research methodology using a variety of evaluation technique, including pre-tests, post-tests, artifact-based interviews and project evaluations.

The programming environment used by the students is called Scratch which was created by MIT. Scratch allows computer programming to be understood relatively easily by everyone due to the basic design features and block-based programming. It allows individuals, who are beginning programming education, to understand and acquire programming logic and algorithm thinking skills.

The number of 4th and 5th grade students participating in the research group is 100 while another control group consists of the same number of students. Each group entails several whole classes chosen randomly among schools having Scratch programming as part of their curriculum. The experimental group learned using the intervention methodology of interdisciplinary study of CT and Science together by creating artifacts of the ABMS genre. The control group learned Scratch and science as two different disciplines.

A pilot study was carried out with eleven 5th grade students. All students had previous experience with the Scratch programming environment, and all had learnt about a specific science phenomenon in school - Sink hole formation. Before conducting the research, the students CT level and Science level were compared to examine if the groups are suitably homogeneous.

For the teachers to be able to lead the program they were trained on ABMS principals within the context of the Scratch software development environment. They were introduced to basic concepts in modeling complex systems through hands-on activities and participatory simulations. A scaffolded series of highly-engaging design and build activities guided them through development of a computer model in a modeling and simulation environment developed by the researches.

The science pre-test included questions about Sink hole formation which they have studied at school. The test was formed by their science teacher. Five questions were in the form of statements on the subject matter. The students answered whether they believed them to be True or False statements. This type of questions aimed to understand about the student's science knowledge. An additional question was an open-ended question which required students to state their opinion. This question was used to indicate about their ability to do a transformation of knowledge from a specific case to a different case baring the same phenomena but consisting of different characteristics.

To test the CT levels of students a computational thinking test was performed. It is based on the principles of common testing methods of various programming competencies (Bers 2018,). A test environment was created by the researches. It is based on the "Program to play" genre in which a student is asked to program in a pre-created game, also known as microworld, consisting of existing procedures in order to advance in the game levels. The microworld has existing stages which are programming puzzles, meaning the microworld supplies feedback to the player upon his successful or ill successful attempts to finish a state. It also has an ability to create additional stages by the game player using his knowledge of programming. The microworld was created in the scratch environment.

The players goal is to program the white square to move to the light green tile without reaching the blue zone in the background. Several stages had an additional sub task which increased the level of difficulty. In those game levels, for a successful completion, the student was required to move a small red square to a red tile prior to reaching the light green tile (See Figure 2).

The "Program to play" test environment and associated instruction materials were tested for clarity and functionality on over 200 students in a prior year therefore it is regarded as a well-functioning reliable tool.

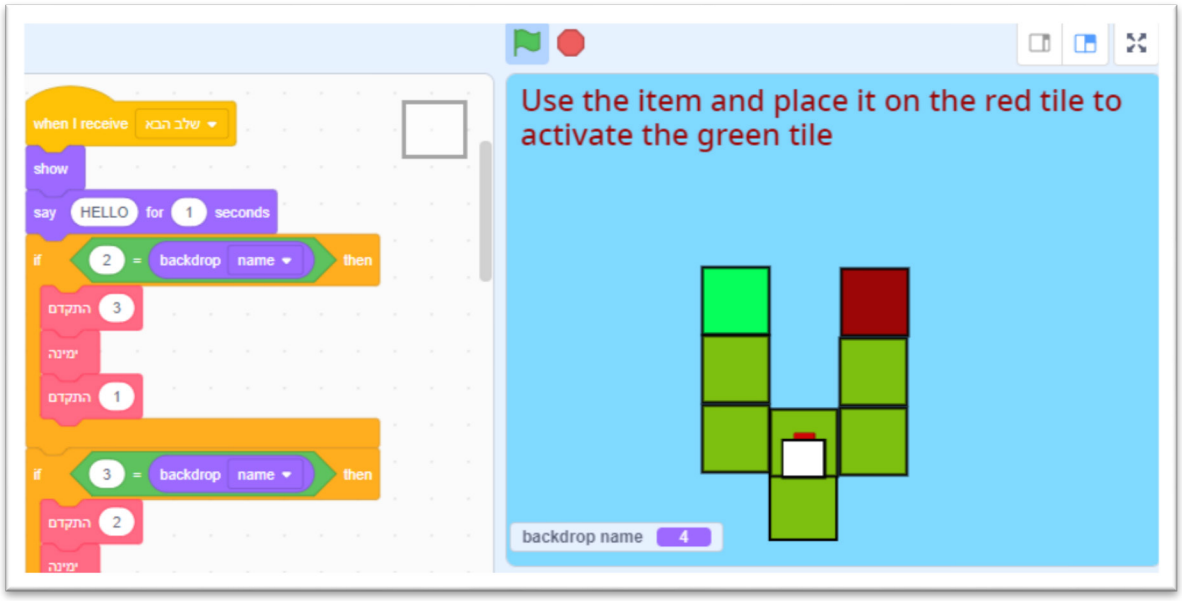


Figure 2. CT-test microworld

A single session was used for the test. It entailed 4 parts: the first being an introduction consisting of verbal explanation by the teacher and specially created videos used to validate the student`s understanding of the microworld and test requirements. The second part was a free-play construction session in which the student was encouraged to try out the programing environment of the microworld and to get acquainted with it. The third part consisted of a time-based requirement to advance in the game levels. Finally, the session had a construction challenge in which the students were asked to add, using higher level skills, additional stages of their own. A simple curriculum was created to guide the test session with the goal of collecting enough information to permit an assessment of computational thinking ability based on review of non-expert raters. A scoring sheet was developed to help standardize scoring by different raters.

The pilot study pre-test results indicated that there were no significant differences between the two groups. Moreover, since the classes follow the same CS curriculum and same Science curriculum their previous experience in learning these subject matters is similar. Therefore, we deemed the two groups to be suitably homogeneous.

The program implementation consisted of 10 classroom hours per class, spread over 5 weeks, for both the experimental and control groups. Each group consists of 5 classes of twenty 4th grade students. Prior to the intervention the classes learned programming with MIT scratch environment for 1 semester and previously, when they were in 3rd grade, they had a full year of CS studies. The entire CS studies prior to the intervention provided students with the opportunity to learn CS concepts and essential programming skills. Both groups were taught by teachers who had experience teaching Scratch programming for at least one year.

The conceptual framework for the design and development process approach is an iterative process for working in different abstractions (See Figure 4). It is accompanied by several specific learning principals:

- The use of rich computational environment. Rich computational environments are ones in which the underlying abstractions and mechanisms can be inspected, manipulated and customized. The rich computational environment is one in which the user can develop CT skills and transform from user to creators (Lee et al., 2014).
- Support low-threshold and wide walls learning activities. They included Scratch template projects for remixing purposes, videos and digital written guide books. These have three advantages: Different groups could work on different projects, teachers are free to assist in the project creation and It also helped the young students to focus on the science project rather on basic agent functionality. Therefore, building appropriate scaffolding and feedback is vital to the success of such projects.
- Incorporate multiple checkpoint for frequent feedback. In a fast pace digital world, a student is used to frequent indications of what he creates. For this, the Computational thinking basic tasks model (see Figure 3a) is used by the teachers in which the simulation is constructed in iterations (steps) each resulting in a working testable abstraction. For instance the task of problem decomposition is a practice that is central in scientific modeling (Nersessian 1992) and by doing it in iterations can facilitate processing the complex task.
- Constructivist sequence of learning activities. In the constructivist pedagogical approach scientific understanding can be developed by building upon and refining existing intuitive knowledge (Dickes & Sengupta 2012). For example, the initial learning activities leverage a naive conceptualization of the domains, and progressively refine them. In the ABMS creation process students begin with programming the behavior of single agents and gradually increase the programs complexity by modeling the behavior and interaction of multiple agents within the ecosystem.
- Use-Modify-Create learning progression model (see Figure 5). The model facilitates scaffolding the student's progression of ABMS in science using a three-stage progression for engaging in CT within computational environments. This progression, called Use-Modify-Create, describes a pattern of engagement that was seen to support and deepen acquisition of CT (Lee et al., 2011). It is based on the premise that scaffolding increasingly deep interactions will promote the acquisition and development of CT. For example, to support the "Use" part of the model, the researchers have created a Scratch simulation of science context which students can use to learn its design before embarking on their own journey for simulation design.

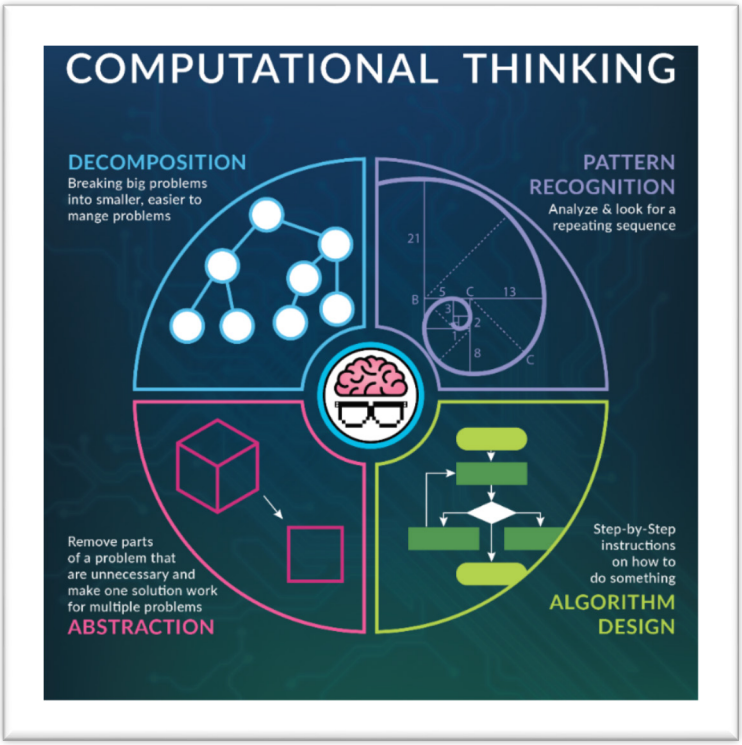


Figure 3a: Computational thinking tasks model in English

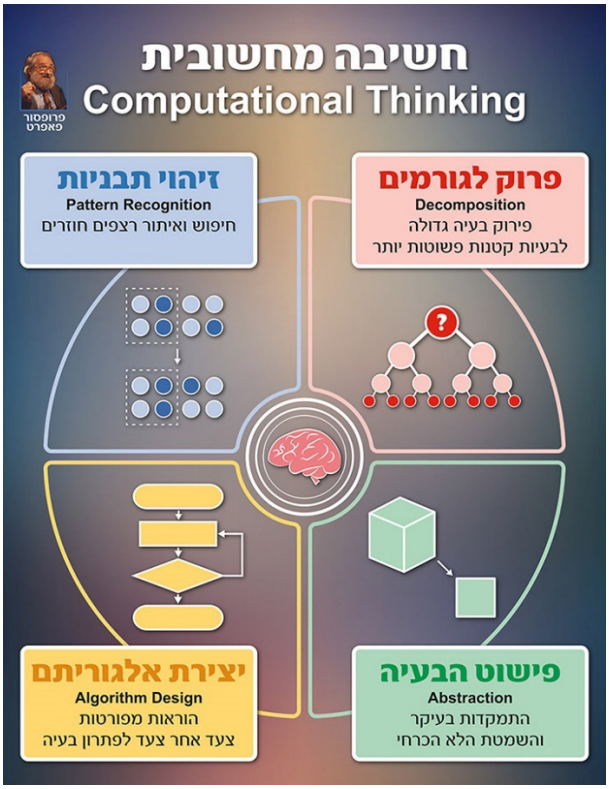


Figure 3b: Computational thinking tasks model in the student native language

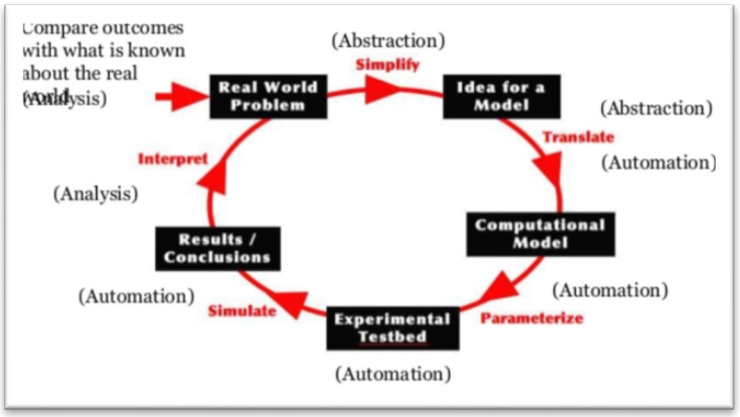


Figure 4: Progression model for integrating computational thinking with science education

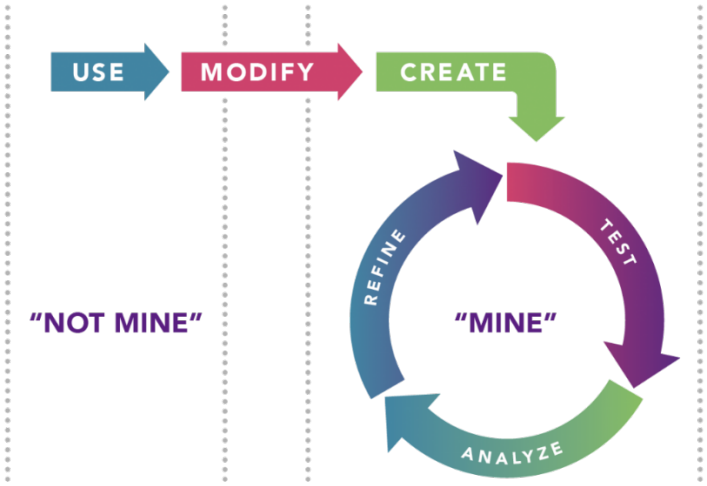


Figure 5: Use, Modify, Create learning progression

At the end of the course a science test was conducted. It consisted of the same structure, subject matter and difficulty level as the science pre-test but had different questions. This contributed to the science aspect of the research and enabled to compare between the student's level of science understanding prior to the course and post the course.

To learn about the students computational thinking level two approaches were used. One being a project analysis and the other a Post-course self-descriptive questionnaire about computational thinking perception. Together they allow the researchers to get a better understanding of the student's capabilities in "Modeling and simulation practices" and "Computational problem-solving practices" which are two of the pillars of the CT-Science taxonomy described in an earlier chapter.

Pilot Study

A pilot study was carried out with eleven 5th grade students. All had previous experience with the Scratch programming environment, and all had learnt about a specific science phenomenon in school. The intervention consisted of a single day, 5 hour long, in school, learning event during school hours. The science topic, Sink Hole Formation, was chosen by their teachers due to it being part of the national science curriculum. The teachers stated that from their experience this topic is known to be challenging for students to grasp, especially the gradual creation of an underground hole and its relation to the decreasing level of water in the nearby sea.

The 1 day event was split into five sessions:

1. Introduction of the event and pre-test. Its purpose is to gather data about the students' knowledge prior to the learning event.
2. Review the science phenomenon during which students reflected on what they have learnt. Teachers focused the discussion on the chemical interaction between salt and sweat water and the sea evaporation causing sweat water to reach the salted ground layers.
3. Introduction of the concept of Agent Based Modeling Simulation (ABMS) followed by a short time to use and remix an existing ABMS infrastructure created by the researches.
4. Students were separated into 3 groups according to their own choice. Each group designed their own ABMS on the topic of sinkholes. After getting the teachers approval of the design they were permitted to start programing by remixing the ABMS infrastructure displayed to them in the previous session.
5. Closure of the event and post-test. Its purpose is to gather data about the students` knowledge and perception of science and computational thinking post the event.

RESULTS OF THE PILOT STUDY

This is a research-in-progress therefore the results section addresses the pilot study carried out. It consisted of eleven students which developed an ABMS of a science phenomenon.

A comparison of the pre and post science questionnaire showed students increased their science subject matter score. With the maximum score being a 100 the students averaged in the pre-test 62 and in the post-test the average score was 80.

Several approaches were designed in this study to understand the courses effect on the students' CT-Science skills. They were:

1) Comparison of pre-course and post-course questionnaires about science.
2) Post-course self-descriptive questionnaire about computational thinking perception.
3) Classroom observations.
4) Artifact based interviews held with a selection of interested students.
5) An analysis of the ABMS projects the students have created.

As part of an online questionnaire the students filled out at the end of the course, the students were asked to express their views using 5 category Likert scale. These were the results:

- When looking at the 2 top categories of the scale, it is apparent that the students had a high satisfactory level of 91. Meaning, 91% of the students marked that they `highly` or `very highly` agree with the statement: 'I like the ABMS development activity'.
- Another question in the questionnaire addressed the student's perception of the course's contribution to their science subject matter understanding. When looking at the 2 top categories of the scale, it is apparent that the students had a high appreciation of 80. Meaning, 80% of the students marked that they `highly` or `very highly` agree with the statement: 'Learning science with the ABMS development activity contributed to my science knowledge of the subject matter'.
- With regards to the question which attempts to understand the computational thinking contribution, the students gave the 2 top categories of the scale a score of 100. It consisted of

36% stating 'highly' and 64% stating 'very highly' with regards to their perception of the affect the activity had on their CT understanding.

- A question in the questionnaire addressed the student's perception of the course's contribution to their problem-solving skills. When looking at the 2 top categories of the scale, it is apparent that the students had a high appreciation of 91. Meaning, 91% of the students marked that they `highly` or `very highly` agree with the statement: 'Computer programming activities help students acquire problem solving skills'.

An open-ended question aimed to assess their ability to transform their acquired knowledge of the science subject matter to a different case. Results showed 64% of the students did transform and supplied an adequate explanation, 18% wrote a partial explanation for their answer and 18% were not able to answer the question in a reasonable manner.

The last question was for the students to mention another science topic they have learnt this year which can be learnt using an ABMS constructionist methodology. 54% of them were able to do that while 46% either did not answer the question or named the interventions activity as the topic that can be learnt. This could mean that perhaps the question was not clear enough.

During the event the groups were observed by the researchers. They were asked to engage in a design process of their ABMS prior to commencing its coding. They were also asked to show the design to the teacher and have the teacher`s approve the design before the advancement from the design phase to the implementation phase. The classroom observation indicated that the process of designing the ABMS was, methodology wise, difficult for the students. 2 groups have hastily and superficially done the design therefore, the teacher asked to add to the written description a graphical visualization of the ABMS screen, drawn on paper. This additional visualization phase of the design proved useful because it led to a more detailed design. For instance, the location of the agents was more detailed, and the number of agent types increased. The 3rd team showed a different pattern of behavior. They were too detailed on the drawing consuming too much time, so the teacher had to encourage them to abstract their visualization which they did. The teacher used throughout the course a computational thinking tasks model (See figure 3b) which was displayed on a large poster. Its purpose was to correlate the students required and actual actions to the basic tasks of computational thinking showing them the models applicability to problem solving of different types, meaning: programming, science and general problem-solving cases.

The artifact-based interview was held with 3 students. It was brief due to lack of available school time resources. The students verbally expressed they highly liked the ABMS activity and their desire to have more such opportunities. They also stated very clearly their desire to show their class mates the simulations they have created and use it as a tool for themselves to teach others about the science subject matter that they created the simulation for.

The 11 students have created 3 simulations during the activity. They were reviewed by the researchers. The simulation had a mandatory theme – they were to create a sink hole creation process. The students were instructed to use a remix approach. The basis for the remix was a simulation which the researchers have created for them. One group stated that it prefers to start from a new empty project and were given the permission to do so. After a short while they decided to comply with the suggestion to do a remix instead. This was suggested by their teacher because the teacher wasn't sure she could guide them to a working simulation if they start from an empty project and due to the short time frame allowed for this session. All 3 groups successfully finished their simulations (see figures 6-8) and were very proud of their project.

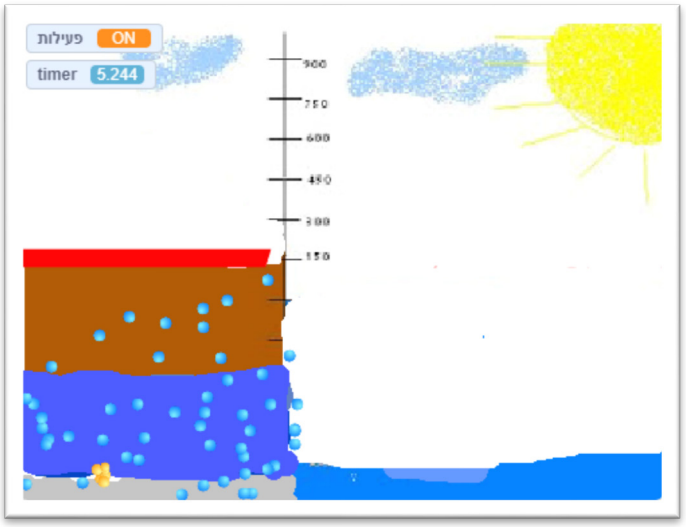


Figure 6: Students simulation #1

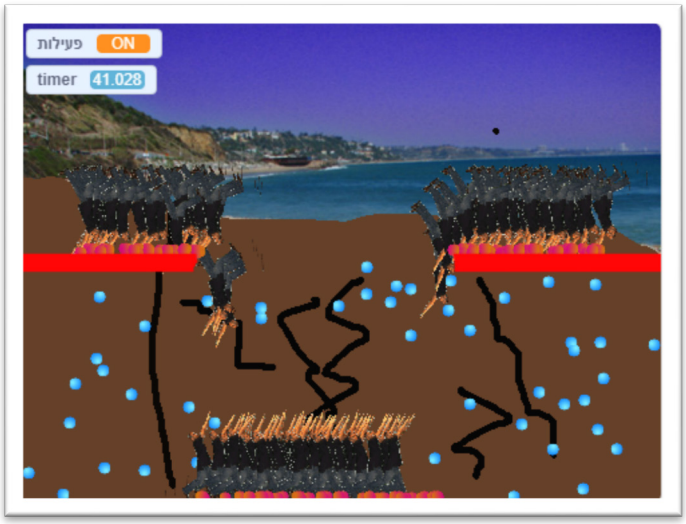


Figure 7: Students simulation #2

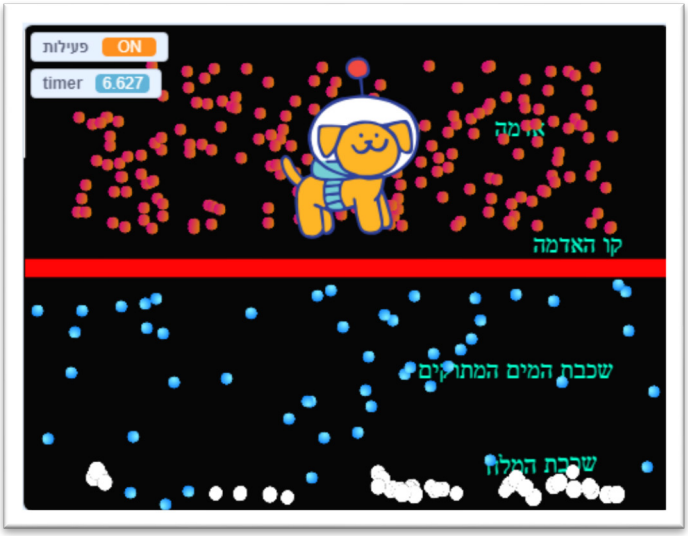


Figure 8: Students simulation #3

CONCLUSION

This paper describes a work-in-progress research. It assessed a pilot study of a Computational Thinking Science intervention program in which students work in groups. In which, they formulated an agent-based module and simulation (ABMS) of a real life phenomenon which is part of their science curriculum.

Results of the pilot study indicated that constructing ABMS improved students CT-Skills, specifically problem-solving skills. Alongside with that, the program enhanced science understanding. Furthermore, the ABMS approach connected directly to student's interests, enabled extended investigations as well as deeper understanding.

CT is an emerging skill in learning science. It is requiring school systems to give increased attention for promoting students with the opportunity to engage in CT activities alongside with ways to promote a deeper understanding of science. Currently there is a lack of practical ways to do so and lack of methods to assess the results therefore it is an educational challenge. This paper presented a response to this challenge by proposing a practical program for school science courses and an assessment method. Further research is due to see if this approach can consequently enable advanced topics to be productively introduced into the elementary school curriculum.

The inclusion of computational thinking as a core scientific practice in the Next Generation Science Standards (NGSS) is an important milestone, but there is still much work to do toward addressing the challenge of CT-Science education to grow a generation of technologically and scientifically savvy individuals. New comprehensive approaches are needed to cope with the complexity of cognitive processes related to CT.

APPENDIX

Table 1. Terms and abbreviations

No.	Element	Description
1	CT	Computational thinking
2	CS	Computer science
3	ABMS	Agent based module and simulation
4	SI	Sprite Instance
5	EWS	Eco Science Works program
6	CTSiM	A framework for constructing simulations
7	IDE	Integrated development environment

REFERENCES

Bers, M. U. (2018). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge. https://doi.org/10.4324/9781315398945

Dickes, A. C., & Sengupta, P. (2013). Learning natural selection in 4th grade with multi-agent-based computational models. *Research in Science Education*, *43*(3), 921-953. https://doi.org/10.1007/s11165-012-9293-2

Dong, Y. (2018). *Modeling students' learning behaviors in open ended learning environments.* (Doctoral dissertation, Vanderbilt University).

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). Embedding computational thinking in science, technology, engineering, and math (CT-STEM). In *Future Directions in Computer Science Education Summit Meeting, Orlando, FL*. https://doi.org/10.1007/978-3-319-08189-2_3

Keeling, M. J., & Gilligan, C. A. (2000). Metapopulation dynamics of bubonic plague. *Nature*, *407*(6806), 903. https://doi.org/10.1038/35038073

Landriscina, F. (2015). The role of mental simulation in understanding and in creating scientific concepts. *Innovazione nella didattica delle scienze nella scuola primaria e dell'infanzia: al crocevia fra discipline scientifiche e umanistiche*, 141.

Lee, I., Martin, F., & Apone, K. (2014). Integrating computational thinking across the K--8 curriculum. *ACM Inroads*, *5*(4), 64-71. https://doi.org/10.1145/2684721.2684736

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., ... & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32-37. https://doi.org/10.1145/1929887.1929902

Lehrer, R., & Schauble, L. (2000). *Modeling in mathematics and science* (pp. 101-159). New Jersey, NJ: Lawrence Erlbaum.

Macal, C. M., & North, M. J. (2010). Tutorial on agent-based modelling and simulation. *Journal of Simulation, 4*(3), 151-162.

Marion G., Renshaw E., & Gibson G. (2000). Stochastic modeling of environmental variation for biological populations. *Theoretical Population Biology, 57*, 197–217. https://doi.org/10.1006/tpbi.2000.1450

MOE (Israeli Ministry of Education) (2016). *Computer science and robotics educational program.* Retrieved Jan 3, 2019 from https://sites.education.gov.il/cloud/home/tikshuv/Documents/tochnit_limudim_machsevim_robotika.pdf

National Research Council. (2008). *Taking science to school: Learning and teaching science in grades K-8*. National Academies Press.

National Research Council. (2010). Committee for the Workshops on Computational Thinking: *Report of a workshop on the scope and nature of computational thinking*. Washington, DC: National Academies Press

National Research Council. (2011). Committee for the Workshops on Computational Thinking: *Report of a workshop of pedagogical aspects of computational thinking*. Washington, DC: National Academies Press.

Nersessian, N. J. (1992). How do scientists think? Capturing the dynamics of conceptual change in science. *Cognitive Models of Science*, *15*, 3-44.

NGSS Lead States. (2013) *Next generation science standards: For states, by states*. Washington, DC: The National Academies Press.

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

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351-380. https://doi.org/10.1007/s10639-012-9240-x

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Exploring the science framework and the NGSS: computational thinking in elementary school classrooms. *Science and Children*, *52*(3), 10. https://doi.org/10.2505/4/sc14_052_03_10

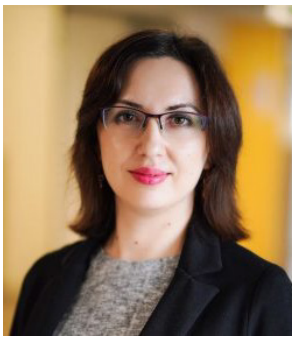
Stelar. (n.d.).*EcoScienceWorks: Exploring and Modeling Ecosystems Using Information Technology*. Retrieved Jan 3, 2019 from http://stelar.edc.org/projects/12151/profile/ecoscienceworks-exploring-and-modeling-ecosystems-using-information

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Wilensky, U., & Reisman, K. (2006). Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories—An embodied modeling approach. *Cognition and Instruction*, *24*(2), 171-209. https://doi.org/10.1207/s1532690xci2402_1

BIOGRAPHIES

Gilad Shamir is a technological-pedagogical lecturer on practical constructionism and computational thinking He is an in-service and pre-service teachers instructor for the ministry of education in Israel. He has worked in the Israeli Hitech industry for twenty years and is working on his PhD at the Tel Aviv University. His interest is researching computational thinking in different subject matters of education, and practical computational thinking assessment methods and has presented papers at conferences. He is a pedagogical manager for a company that supplies technology learning content of the ministry of education in his country. In this role he is the author of over a hundred technology educational hand books used by students and teachers alike.



Dr. Dina Tsybulsky is an Assistant Professor and head of the Biology Education research group in the Faculty of Education in Science and Technology at the Technion Israel Institute of Technology. Dr. Tsybulsky serves at the NARST Association as a member of the Program Committee and as a co-chair of the History, Philosophy and Sociology of Science research strand. Currently, she is a PI of the research grant of the Israeli Science Foundation. Dr. Tsybulsky's research interests include: inquiry-based biology learning; nature of science and scientific practices; pre- and in-service teachers' worldviews, beliefs and attitudes.

lya Levin received his Ph.D. degree in Computer Engineering in 1987 from the Institute of Computer Technologies, Latvian Academy of Science. From 1987 he was the Head of the Computer Science Department in the Leningrad Institute of New Technologies (in the former USSR). He moved to Israel 1990. Between 1993 and 1997, Ilya Levin was the Head of the Computer Systems Department in Holon Institute of Technology. In 1997, he worked as a Research fellow in the Computer Science Department of University of Massachusetts. During four years between 2003-2006, Ilya Levin was an Associate Professor of the School of Engineering in the Bar Ilan University. In 2014, he was a visiting professor in Ca' Foscari University of Venice. Presently, Prof. Ilya Levin is a Full Professor in the School of Education of Tel Aviv University. His recent research interests include: Computer Design, Cultural Studies of Information Society, Science and Technology Education. Ilya Levin is the author of around 160 research papers both in Computer Engineering and in Humanities.