Active Learning by Visual Programming: Pedagogical Perspectives of Instructivist and Constructivist Code Teachers and Their Implications on Actual Teaching Strategies and Students’ Programming Artifacts

Avital Kesler¹, Tamar Shamir-Inbal¹, and Ina Blau¹

Abstract
The integration of visual programming in early formal education has been found to promote computational thinking of students. Teachers’ intuitive perspectives about optimal learning processes - "folk psychology" - impact their perspectives about teaching "folk pedagogy" and play a significant role in integrating educational technologies, such as visual programming, within the formal curriculum. This study was conducted based on the mixed method research paradigm. First, a folk pedagogy questionnaire was distributed to 89 teachers who integrate differing technologies in their classroom in order to identify the teachers’ pedagogical perspectives:

¹Department of Education and Psychology, The Open University of Israel, Ra’anana, Israel

Corresponding Author:
Ina Blau, Department of Education and Psychology, The Open University of Israel, 1 University Road, P.O.B. 808, Ra’anana 43107, Israel.
Email: inabl@openu.ac.il
constructivist versus instructivist. Then, semi-structured interviews were conducted with 24 teachers who teach Scratch in order to gain a deeper understanding of their instructivist/constructivist perspectives and actual pedagogical practices and strategies. Finally, we analyzed 96 students’ programming artifacts to explore differences, if any, in students’ outcomes related to the pedagogical perspectives of their teachers. Findings revealed that pedagogical perspectives are reflected in teaching strategies and assessment practices employed in a visual programming environment. It is promising that teaching visual programming promoted constructivist pedagogy even among instructivist teachers and was consequently reflected in student perspectives and expressed in their programming artifacts. We discuss theoretical and educational implications of these findings.

Keywords
visual programming, constructionism, computational thinking, folk psychology and folk pedagogy, instructivist and constructivist pedagogical perspectives

Introduction

Visual Programming – Scratch Environment

In recent years, integration of visual programming through code and robotics studies has been a growing trend around the world (Rich et al., 2019; Wei et al., 2020) and in the Israeli education system (Israeli Ministry of Education, 2019). The reason for this is experts’ claim that visual programming is a powerful tool for promoting and developing computational thinking (Jacob & Warschauer, 2018; Kafai, 2016; Shute et al., 2017; Wei et al., 2020). Following this world movement, Israeli elementary and middle schools incorporated a standard new curriculum – “Code and Robotics”, which focuses on visual programming (Hadad et al., 2020) and includes a high-stake standardized test for middle school students. This reflects a desire to promote students’ computational thinking – a complex system of thinking skills, such as data collection, programming, and data analysis, which are necessary for success in the digital era (Rodríguez-Martínez et al., 2020; Wing, 2008). Practicing visual programming allows the learner to experience skills such as abstraction, decomposition and problem analysis, critical thinking, algorithmic design, evaluation, and generalization (Jacob & Warschauer, 2018; Kafai, 2016). In addition, previous research indicates that engaging in visual programming may promote the computational thinking of elementary school students (Olmo-Muñoz et al., 2020).

The original idea for teaching coding in a creative way through visual programming is based on Papert’s (1980) constructionist approach. According to
this theory, optimal knowledge construction can be empowered by building and constructing tangible artifacts in a physical or digital environment. In the last decade, recognizing the importance of constructionist learning has been strengthened in advanced technological platforms that allow intuitive coding through visual programming in environments such as Newgrounds, Alice, and Kodu (Kafai, 2016). These environments support project-based learning, which is found to be an effective way of promoting computational thinking (Hsu et al., 2018).

To date, most research on the contribution of visual programming to the computational thinking of school-age students has been conducted in informal learning settings (e.g., Luo et al., 2020; Zuckerman et al., 2009). A few studies, which examined the integration of visual programming in formal education, have shown a contribution to the development of creativity and the promotion of computational thinking (Calder, 2010). It was also found (Ke, 2014) that integrating visual programming in the context of a standard school curriculum may increase learning motivation and accentuate the relevance of the content taught in school to the students’ world (So et al., 2020). A recent literature review (Hu et al., 2021) reported a small to medium positive effect of visual programming on students’ achievements. Thus, it seems that integrating visual programming in the education system may benefit both students and the learning processes and outcomes. In addition, it can promote interdisciplinary learning when programming artifacts is related to the curriculum in math, language, history, geography, etc. (Hadad et al., 2020). This promise is seldom realized, as a literature review conducted by Hsu et al. (2018) found that most of the studies focused on programming skills training and mathematical computing, while only a few were interdisciplinary.

The visual programming environment examined in this study is “Scratch Online” (Figure 1), which is widely used in countries around the world in order to develop programming skills and promote computational thinking of students (Rich et al., 2019; Zhang & Nouri, 2019). This environment was developed at MIT in the United States in order to enable young learners to learn programming in an intuitive and creative way in a digital collaborative environment. Programming in the Scratch environment takes place as a process of assembling “building blocks” of various types. The environment allows students to experiment coding independently or with peers in producing creative and meaningful artifacts such as animated videos, interactive quizzes, games, and more (Resnick, 2012). The artifacts designed by users can be shared on the Scratch online platform and thus be accessible to others for learning or leisure activities. We refer to them in the paper as the “end users” of the artifacts.

The quality of technology-enhanced learning processes in the education system is related to the role of the teacher (Ching et al., 2018). This study focuses on the role of the teacher in a visual programming classroom.
The Role of the Teacher

Olson and Bruner (1996) found that teachers hold intuitive perspectives about teaching-learning processes. These folk perspectives were divided into four models, representing different types of learner consciousness. Each model is represented by folk psychology, which characterizes the concept of the learning process, and folk pedagogy, which describes how to teach in accordance with a particular folk psychology. Figure 2 shows the relationship between folk psychology and folk pedagogy.

As shown in Figure 2, the four models of “theory of mind” belong to two opposite pedagogical perspectives of learning (Peled et al., 2015). The first and second models reflect the instructivist perspective, while the third and fourth models reflect the constructivist perspective to learning. According to the instructivist perspective, knowledge is objective, and there is only one correct answer to each question. In contrast, the constructivist perspective perceives learning as a complex process, in which there are several correct answers and more than one solution. Thus, each individual may contribute his or her unique experience to help create extensive knowledge (Peled et al., 2015).

The different pedagogical perspectives of teachers in the context of teaching-learning processes and the adoption of innovation have a crucial impact on the perception of the role of the teacher and how he/she uses digital tools in teaching (Thibaut et al., 2018). Unfortunately, these perspectives are one of the most difficult factors to change (Tondeur et al., 2017). Studies have shown (Avidov-Ungar et al., 2020; Blau & Shamir-Inbal, 2021; Peled et al., 2015; Tondeur et al., 2017) that the instructivist perspective among teachers leads to a use of technology that reinforces the “teacher-centered” approach. These teachers will make extensive use of pedagogical strategies such as demonstration or explanation. It is
reasonable to assume that the artifacts of a learning process guided by instructivist teachers will be homogeneous, since, based on the teacher’s perspective, there is one particular way to program an optimal artifact. In contrast, teachers who hold a constructivist pedagogical perspective may lead students to use technology for active learning in the “student-centered” approach (Blau et al., 2018; Shamir-Inbal & Blau, 2020). These teachers will encourage their students to experiment independently and to rely on peer learning in order to acquire new knowledge. It can be assumed that the learning artifacts guided by constructivist teachers will usually be more diverse, since, depending on the teacher’s perspective, there are many ways to program an optimal artifact.

In the context of visual programming, previous studies claimed that constructivism is a proper approach for teaching and learning (Bakar et al., 2019; Hadad et al., 2020). However, research findings that followed the process of creating games using the Scratch environment (Ke, 2014) found that most of the educational digital games created by students under the guidance of computer science teachers in schools were instructive in nature and may have reflected the pedagogical perspective of the teachers. The literature review conducted by Hsu et al. (2018) found that such activities as aesthetic experience, design-based learning, and storytelling have rarely been adopted when teaching visual programming. That is, it can be assumed that the experience of a visual programming environment such as Scratch in the education system will not necessarily lead to pedagogical change, but will reflect the folk psychology and folk pedagogy of teachers who integrate it in their classroom.

| **Folk psychology** | **Folk pedagogy** |
|---------------------|-------------------|
| **Instructivist**   | **Instructivist** |
| 1. Learning is the ability to internalize a desired, demonstrated behavior or skill. | 1. Learning depends on the demonstration and practice of a required skill. |
| 2. Learning is the transmission of knowledge from the external environment to the learner. | 2. Knowledge is transmitted to the learner by a teacher through presentation and explanation of the topic. |

| **Constructivist** | **Constructivist** |
|-------------------|-------------------|
| 3. Learning is a discovery of personal knowledge through the unique experiences of each person. | 3. One must help a student discover the knowledge that is naturally found in his cognition. |
| 4. Learning is building personal knowledge while relying on existing scientific or cultural information. | 4. It is important to complement personal knowledge and experience by scientific or cultural information. |

**Figure 2.** Four Models of “Theory of Mind” in Learning and Teaching (Olson & Bruner, 1996).
In order to explore these issues, this study examines the following research questions:

1. What are the pedagogical perspectives of instructivist and constructivist teachers regarding the way they design the teaching of visual programming in the classroom?
2. How, if at all, are the pedagogical strategies of teachers with instructivist and constructivist perspectives reflected in student artifacts?
3. How, if at all, are the pedagogical strategies of teachers with instructivist and constructivist perspectives reflected in the degree of active learning conducted by the end users of student artifacts?

Methodology

This study was conducted using the mixed-method approach. On the one hand, we used qualitative methods to examine in depth the pedagogical perspectives that guide teachers of visual programming using the Scratch environment. On the other hand, qualitative methods were employed to categorize teachers’ “folk pedagogy” (Olson & Bruner, 1996). Based on this categorization, we further explored the connection between teachers’ perspectives and the characteristics of programming artifacts created by their students, as well as the level of active learning conducted by the end users of these artifacts.

Participants

In the first phase of this study, participants were 89 teachers from elementary and secondary schools across Israel who integrate different technologies, including visual programming, in their teaching. These teachers filled in a folk pedagogy questionnaire detailed in the following section. In the second phase, 22 teachers were selected for interviews aimed at a deeper understanding of their pedagogical perspectives and classroom practices. The interviewees are representative of the education system in Israel in terms of teaching experience (2-30 years), experience in teaching visual programming (half year - 7 years), gender (13 women and 9 men), sector (14 Hebrew speakers and 9 Arabic speakers), and the age of the learners (13 elementary and 9 secondary school teachers). These teachers completed a professional development course for basics of visual programming offered by the Ministry of Education (60 hours). The main focus of their training was on technological aspects of using Scratch visual programming in the classroom with only a few references to methods of its integration.

In addition, the teachers who were interviewed were asked to send a link to examples of representative artifacts (at different levels) created by their students and published on the Israeli Scratch Community website (http://www.scratch.org.il/) or on the international Scratch community website (https://scratch.mit.edu).
According to the teachers, these 3-4 artifacts provided by each teacher (in total, \( n = 109 \)) reflected different levels of student work. There were 78 artifacts created by students from Hebrew-speaking schools and 31 by students from Arabic-speaking schools.

**Research Tools and Procedure**

In this study, we combined several tools that helped us characterize the pedagogy that guides teachers, identifies their teaching strategies, and analyzes student artifacts:

- **Digital questionnaire**: In the first stage, an online questionnaire on folk pedagogy created with the Google forms app (Folk Pedagogy questionnaire; Blau & Pieterse, 2015, based on Olson & Bruner, 1996 framework) was distributed through a national Facebook group of teachers that are interested in discussing integrating ICT in general and visual programming teaching in particular. The questionnaire included four statements listed in Table 1. The purpose of the questionnaire was to associate pedagogical perspectives of the participating teachers with one of the four models of “theory of mind” (Olson & Bruner, 1996). Participants were characterized as having an instructivist pedagogical perception (if they chose answers A-B to the questionnaire statements) or constructivist pedagogical perception (if they chose answers C-D). At the end of the questionnaire the teachers who teach visual programming were asked to indicate whether they would be willing to be interviewed and if so, left relevant contact details on the form. Of the teachers who volunteered to continue, interviewees were chosen based on the representative criteria detailed in the previous section.

Analysis of factors confirmed with Warimax rotation showed that all items in the questionnaire were pooled into a single factor (Table 1). Reliability as an internal consistency of the questionnaire was good, Cronbach’s \( \alpha = .714 \), and all statements were included in the index that was normally distributed (range: 1-4, average: 2.80, standard deviation: 0.76, median: 2.75, skewness: 0.281).

- **Interviews with teachers**: In the second phase of the study, 22 teachers were interviewed in a semi-structured interview (inspired by Brennan & Resnick, 2012), which lasted about 45 minutes and was conducted by phone. During the interview teachers were asked to describe how they shaped the teaching process to support the learning and designing of projects which promote computational thinking in their students (Zhang & Nouri, 2019). For example, teachers were asked questions such as: Please describe your experience of teaching visual programming in the Scratch environment. Which successes and challenges do you encounter in the teaching process? How do you overcome these challenges? Please describe what strategies you use to teach visual programing and why. What characterizes the type of projects you guide your students to create?

The interviews were recorded, transcribed, and analyzed bottom-up based on the Grounded Theory approach (Charmaz, 2014), which presents participants’
perspectives and practices in their natural context. The coding yielded 629 statements, and 25% of the data was re-coded by two additional raters. A few statements, which were coded differently, were discussed between the raters, and the findings presented in the Results section reflect full agreement between the three raters.

### Table 1. Factor Analysis of the Folk Pedagogy Questionnaire (Blau & Pieterse, 2015, Based on Olson & Bruner, 1996 Framework).

| Items (1–4) and multiple choice (a/b=instructivist, c/d=constructivist) | Factor |
|---|---|
| 1. When I am taught something new, the best way for me to learn is...  
   a. to mimic and practice the execution of a desired action as demonstrated by someone who knows how to do what I want to learn.  
   b. to listen to an explanation/a lecture given by an expert on a topic I want to learn.  
   d. to think and discover independently (alone or with friends) what I want to learn.  
   e. to rely on existing scientific or cultural information in order to think and discover independently (alone or with friends) what I want to learn. | .819 |
| 2. When I learn something new, the best way to do it is...  
   a. to have the action demonstrated to me.  
   b. to explain the relevant concepts and principles directly and explicitly to me.  
   c. to guide me (in an individual or group process) so that I can gain insights about the topic being studied.  
   d. to guide me (in an individual or group process) on how to rely on existing information concerning science or culture in order to think about and discover what I want to learn. | .635 |
| 3. When I want to teach others something new, the best way to do it is...  
   a. to demonstrate to others the desired action and allow them to emulate and practice it.  
   b. to deliver the topic being studied through direct instruction of the concepts, principles and phenomena.  
   c. to guide others through dialogue to enable them to formulate insights and present their own arguments.  
   d. to help others rely on scientific/cultural knowledge in order to enable them to formulate insights and present their own arguments. | .752 |
| 4. I think a successful lesson is a lesson where...  
   a. I demonstrate the task in such a way that most students will perform it successfully.  
   b. I succeed in explaining the learning content in such a way that most students will understand what I have explained.  
   c. I manage to inspire my students to discuss the topic being studied, so that they gain insights and formulate conclusions in the context of the topic.  
   d. I succeed in stimulating my students to discuss and link their personal opinions to the existing universal knowledge about the subject when using academic or cultural sources. | .783 |
Student artifacts analysis: Among 109 artifacts provided by the participants, 53 artifacts were conducted under the guidance of instructivist teachers, while 56 were under the guidance of constructivist teachers. Note that the number of teachers holding instructivist and constructivist pedagogical perspectives was similar in both elementary and secondary schools.

Student artifacts provided by the interviewees as representative artifacts were analyzed by raters according to criteria used in previous studies (Blau & Benolol, 2016; Kafai, 2016; Peppler & Warschauer, 2011). A number of parameters were examined on a scale between 1-5. These parameters were: the level of programming of the project (i.e., the degree of its technological sophistication; Blau & Benolol, 2016; Peppler & Warschauer, 2011), the degree of clarity of the project idea, the degree of clarity of the user interface (i.e., the degree to which the project’s interface is clear and understandable for users and allows them to make efficient and convenient use of the project; Blau & Benolol, 2016; Peppler & Warschauer, 2011), the degree of originality and creativity of the project (e.g., the extent to which the visual design or concept of the project is innovative and special, relative to other projects examined; Peppler & Warschauer, 2011), and the level of aesthetics of the project design. Beyond these six criteria from previous studies, we used four criteria that emerged bottom-up from interviews with our participants: target user’s activity (i.e., to what extent users of the students’ projects become active learners), connection to the curriculum (yes/no), number of scripts, and number of sprites, as counted by the Scratch interface for each project.

Additionally, assessment of the programming level and complexity was conducted using the Dr. Scratch app (http://www.drscratch.org/; Moreno-León et al., 2015). The app evaluates each project on a scale from 0 to 21. Each project is rated according to levels: 0–7 – basic level, 8–14 – developing level, and 15–21 – mastery level. The projects are evaluated according to the following seven parameters, with each parameter measured on a scale from 0 to 3: flow control, data representation, abstraction, user interactivity, synchronization, parallelism, and logic. The combination of two analysis methods enabled a comprehensive assessment of student artifacts.

The artifacts’ coding, based on the criteria from previous studies, was conducted by two raters trained by the researchers. The first rater assessed the entire data set, while 25% of the artifacts were independently assessed by a second rater, a visual programming expert who was unaware of the research questions. The inter-rater reliability between the raters was high, Cohen’s kappa = .88, and the average rating was used for the analysis presented in the next section.

Results

Pedagogical Perspectives and the Design of Teaching Visual Programming

The aim of the first research question was to present the relationship between teachers’ stated pedagogical perspectives and how they actually expressed their
perspectives in visual programming teaching in the classroom. Based on the Folk Pedagogy questionnaire (Blau & Pieterse, 2015, based on Olson & Bruner, 1996 framework) presented in the previous section, 9 instructivist and 13 constructivist teachers were identified among the participants of the interview phase. The pedagogical design described in detail by the teachers in interviews was consistent with their categorization as either instructivist or constructivist teachers. Yet, despite belonging to two different pedagogical perspectives, teachers from both groups combined instructivist and constructivist teaching strategies in their teaching, but to a different extent and in accordance with their pedagogical perspectives. Tables 2 and 3 present the teaching strategies described by the teachers from the two groups addressing the combination of the two approaches.

The findings presented in these tables show that out of 446 content units from the interviews which relate to the various teaching strategies, the implementation of teaching strategies of the constructivist type is significantly higher ($\chi^2 = 22.101, p < .001$) among teachers with constructivist perspectives, compared with their colleagues with instructivist perspectives. It was also found that teachers with instructivist perspectives apply, almost equally, strategies which reflect both instructivist and constructivist perspectives in their teaching of visual programming (Table 2). In contrast, teachers who were characterized as having a constructivist perspectives apply mostly constructivist strategies (Table 3). The differences between applying instructivist versus constructivist strategies are presented in Figure 3.

The findings in Figure 3 show notable differences between instructivist and constructivist teachers with respect to the use of different strategies. The most prevalent instructivist strategy in teaching visual programming was demonstration, and the most prevalent constructivist strategy was allowing for learning from experience. A comparison of the degree of implementation of these strategies within each group of teachers shows that teachers with an instructivist perspective tend to use a demonstration strategy 1.5 times more than constructivist teachers and allow fewer independent learning experiences for their students. In contrast, teachers with a constructivist perspective tend to allow students to experiment independently in a Scratch environment almost three times more than with the demonstration strategy. In addition, teachers with an instructivist perspective tend to explain verbally and use more practice and repetition than constructivist teachers. In contrast, teachers with a constructivist perspective allow their students to acquire knowledge through learning by peer instruction and direct their students to work independently to problem-solve in a visual programming environment.

Another interesting finding is illustrated in Figures 4 and 5, which show percentages of the use of different teaching strategies by each of the teachers interviewed for the study. From the diagrams, it can be concluded that, as expected, all teachers with a stated constructivist perception tend to use significantly more constructivist-type strategies (Figure 5). Surprisingly, some
Table 2. Strategies Used by Teachers Who Were Categorized as *Instructivist*.

| Instructivist strategies                                      | Representative citations                                                                 | N  | %  |
|---------------------------------------------------------------|------------------------------------------------------------------------------------------|----|----|
| Encouraging creation of meaningful artifacts                  | "I told them, 'From now until the end of the year, you are doing a final project. The    | 11 | 6.3% |
|                                                              | class website has examples of interesting projects, games, and dungeons, so anyone can  |    |     |
|                                                              | take the ideas to create whatever they want, their dream project.'" (N.S.)              |    |     |
| Code demonstration by teacher                                 | "I *demonstrate* in advance rather than using the 'flipped classroom' method. . . I do  | 44 | 25% |
|                                                              | not let them discover things on their own, I do not work like this." (N.S.)             |    |     |
| Practice and repetition                                       | "I always say: 'Practice!' I explain to them that practice is like eating and drinking,   | 14 | 8%  |
|                                                              | you do it every single day. A *student has to practice* both in class and at home.      |    |     |
|                                                              | That's how it is in Scratch. You have to practice and work until you know how it        |    |     |
|                                                              | works." (A.A.)                                                                            |    |     |
| Illustration                                                  | "I try to *illustrate* the concepts I talk about - *draw* them on the floor and use     | 4  | 2.3%|
|                                                              | *different aids.*" (R.S.)                                                                  |    |     |
| Creating work procedures                                      | "I'm very strict about *programming rules* . . . and also about writing comments on     | 5  | 2.8%|
|                                                              | the side. They have to be very, very neat — to know exactly what they are doing."       |    |     |
|                                                              | (A.H.)                                                                                   |    |     |
| Oral and written explanations                                 | "I *explain*, and everything is written in the lesson plan. I tell them: 'I *explained*,  | 22 | 12.5%|
|                                                              | *Anyone* who did not understand can repeat the written material found on my class     |    |     |
|                                                              | website.'" (N.S.)                                                                         |    |     |
| Total N=89                                                    |                                                                                         | 50.6%|   |

| Constructivist strategies                                     |                                                                                         | 26 | 14.8%|
| Learning through challenging experiences                      | "Something I started this year . . . *seven minutes of challenge*. After they know how   |    |     |
|                                                              | to work with a sprite that collides with another sprite (and know how to work with     |    |     |
|                                                              | codes of movement of different types), I set seven minutes on the timer. Suddenly they  |    |     |
|                                                              | understand, they know how to do it, and they start creating." (S. W.)                  |    |     |

(continued)
Table 2. Continued.

| Strategy                                      | Representative citations                                                                                                                                                                                                 | N  | %  |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|
| Learning relevant to students                 | “We learned the axis of the numbers . . . I demonstrated the idea by using an elevator. For example, in a mall, if I enter on the ground (0) floor and want to get to the dining floor, I go up to the next floor (1). If I want to go down to the parking lot, what should I do? . . . They immediately answered: ‘We should press minus 1!’ This is an illustration of the axis of numbers that is relevant to students’ lives.” (H.A.) | 6  | 3.4% |
| Demonstration and explanation with the participation of students | “To draw a square, I do not just explain what to do, but also invite them to think together by asking questions: ‘Let’s look at command bricks. Is there a brick that can repeat the ‘move’ and ‘turn’ commands? How do I know how many times the cat needs to repeat these commands so that a square comes out? etc . . . ’ ‘Oh, teacher, we created a fan!’ Thus, by asking questions, the explanations came from them!” (H.A.) | 16 | 9.1% |
| Brainstorming                                 | “During self-experimentation all kinds of questions arise, and we discuss questions like the following . . . ‘I was able to make the sprite I created shoot, but I want to make the sprite shoot again when I press ‘enter’. I ask all of them to share suggestions how (in their opinion) this can be done.” (S.W.) | 3  | 1.7% |
| Peer teaching                                 | “Students work in pairs. They want to and love working together. There is always someone who knows more. It helps a lot to program in couples - they help each other.” (A.A.)                                                                 | 11 | 6.3% |
| Strategies for developing computational thinking | **Encouraging pre-planning:** “I write instructions about what should be in each step: Step One: Create a Script, Step Two: Choose 2 sprites, Step Three: Create an event for a sprite, Step Four: Set Variables for the code. They learn that you should always start with an event, write where the sprite will stand on a screen, and, only after that, act according to additional steps. There is a logical order in programming things. After a while they no longer need the instructions. They know the order of the planning steps and are used to the process.” (R.S.) | 8  | 4.5% |
**Table 2.** Continued.

| Representative citations                                                                 | N  | %    |
|-----------------------------------------------------------------------------------------|----|------|
| Analysis in collaboration with the learner: "If he says, 'I can't program this', I cannot help him. I help them analyze and focus on the problem: 'You can't do what? What did you try? Where did you get to? Where did you get stuck?' It is important that students are able to guide themselves and explain where the programming became difficult and where they needed help." (H.A.) | 6  | 3.4% |
| Total N = 87                                                                             |    | 49.4%|
| Total number of citations N=176                                                          |    |      |
Table 3. Strategies Used by Teachers Identified as Constructivists.

| Instructivist strategies                                      | Representative citations                                                                 | N   | %   |
|---------------------------------------------------------------|----------------------------------------------------------------------------------------|-----|-----|
| Code demonstration by teacher                                | “Sometimes, if the student wants something specific, and it’s really beyond his level, then I tell him, ‘Let’s do it this way: I’ll give you an example, and then you’ll create a code based on that example.’” (H.L.) | 26  | 9.6%|
| Practice and repetition                                       | “If it is really difficult, I tell them: ‘Go back to learning what we have already learned about these commands, because this basis is lacking in your dealing with this challenging code.’” (I.B.) | 3   | 1.1%|
| Illustration                                                  | “... when I talk about ‘variable’ command blocks I illustrate this to them as boxes inside the computer. I keep the variable inside a kind of box...” (T.S.) | 7   | 2.6%|
| Creating work procedures                                      | “Sometimes... I give those who do not feel comfortable in this environment a kind of framework—step by step—of how to act in order to program a script.” (H.L.) | 3   | 1.1%|
| Oral and written explanations                                 | “There is a stage when the students have to listen to my explanation; for example, when I show new commands, then I just talk. I explain to them what the command is needed for and in what situation it should be used, etc.” (Y.T.) | 18  | 6.7%|
| Total N=57                                                    |                                                                                       |     | 21.1%|
| Constructivist strategies                                     |                                                                                        |     |     |
| Learning through challenging experiences                      | “... I give them a problem, and tell them, ‘Let’s find the solution... you will look for a suitable command block and find out how it can be used.’ Then I give them a quarter of an hour to tackle the problem and try to program a small part in a complex script.” (A. E.) | 74  | 27.4%|
| Learning relevant to students                                 | “In the first lessons, when we learned about pen and drawing, we constantly made a comparison between the real world, which is relevant to the students, and the virtual world.” (I.B.) | 8   | 3%  |
| Demonstration and explanation with the                        | “I remind them that I want the sprite to move between random points and also to move to a different point every half minute. I ask, ‘So let’s see what you think the script should be so that the sprite jumps from point to point every half minute? Now, if we want to | 39  | 13.2%|

(continued)
| Instructivist strategies                      | Representative citations                                                                 | N  | %  |
|---------------------------------------------|------------------------------------------------------------------------------------------|-----|-----|
| participation of students                   | set a time that will be exactly half a minute, which command block can be used?" (M.O.)  | 5   | 1.9%|
| Brainstorming                               | “At the beginning of the lesson we (the teacher and the students) usually conduct a whole-class discussion about the project we are going to do. They share their ideas and we have a conversation about how it can be done – to identify what they know and what they do not know. We do a sort of alignment.” (T.S.) | 5   | 1.9%|
| Encouraging creating meaningful artifacts    | “The summary task was a challenging project that everyone had to perform. Each student created a script based on his/her own idea. I asked them not to repeat the examples I gave in class in any way, but rather think what is interesting to them... so one created a script with a cat and a mouse - another with a car and a motorcycle.” (E.E.) | 19  | 7%  |
| Peer teaching                               | “I encourage my students to help each other. Teamwork is an important principle. The whole world is moving to the concept of sharing! Collaboration is something that flows in the classroom. If they work in high tech in the future, then they will have to work in teams. That’s why it’s good to develop this practice in the classroom.” (M.K.) | 41  | 15.2%|
| Strategies for developing computational thinking | Encouraging pre-planning: ”Scratch forces children to stop and think about something in an orderly, step-by-step fashion. I encourage them to create a flow chart (before creating a code) in order to exercise logical thinking. That’s how they understand exactly what each step in creating a script means.” (Y.E.) | 9   | 3.3%|
|                                            | Analysis in collaboration with the learner: ”When a student with a specific problem approaches me, I ask him to explain what exactly his difficulty is... I sit with him... we think logically. For example, if he does not know how to choose a suitable block, I tell him: ‘What kind of command do you think you need - movement, control?’ I want to just give him hints, so that he/she will come up with a solution on his/her own and will understand the logic of the script he/she is coding.” (A.B.) | 18  | 6.7%|
| Total N=213                                 |                                                                                          |     |     |
| Total number of citations N=270              |                                                                                          |     |     |
teachers with a declared instructivist perspective also adopt more constructivist than instructivist-type strategies in their teaching in a visual programming environment (Figure 4). These four teachers were defined as having mixed pedagogical approaches and thus, artifacts of their students (n = 13) were removed from the comparative analysis reported in Tables 4 and 5.

**Figure 3.** Differences Between Instructivist and Constructivist Teachers in Pedagogical Strategies Applied for Teaching Visual Programming.

**Figure 4.** Rating of Instructivist Teachers: Using Strategies That Characterize Different Pedagogical Perspectives.
The second research question discusses the way in which different teacher pedagogical perspectives are reflected in the characteristics of student artifacts created in the visual programming environment. In doing so, the parameter “Target user’s activity” – a finding related to the third research question - was examined according to whether there are differences in this parameter among students studying with a different pedagogical guidance. In the evaluation criteria presented in Table 4, criteria 1–6 were taken from previous studies (Blau & Benolol, 2016; Kafai, 2016; Peppler & Warschauer, 2011), whereas parameters 7–10 were included in this study based on the issues that emerged during interviews with teachers. To test differences in artifact quality, an independent samples t-test was performed. The test was performed with reference to a comparison between quality measures according to various assessment measures and to teacher perspectives (instructivist versus constructivist). Table 4 shows test results performed on all artifacts sent by the teachers.

As can be seen in Table 4, the “quality of programming” parameter in the assessment by an external expert rater was found to be higher in artifacts created under instructivist guidance ($p = .012$). In contrast, the parameters of “clarity of the idea” ($p = .001$), “creativity and originality” ($p = .006$), and “aesthetics of design” ($p = .074$) were found to be higher in artifacts created under constructivist guidance.
We have presented the analysis of differences between students of instructivist versus constructivist teachers in all the artifacts. In addition, we used an independent samples t-test to explore the differences in artifacts which were created after six months or more of studying visual programming, i.e., by students who could be defined as “became an expert” in learning in a Scratch environment. Table 5 shows the results of these comparisons.

Table 5 shows that all parameters indicate a higher quality of student artifacts created under guidance of constructivist teachers. Similar to the analysis of all artifacts (presented in Table 4), the artifacts of students defined as experienced were also found to be significantly higher than artifacts created under instructivist teacher guidance in parameters of clarity of idea ($p = .002$) and creativity and originality ($p = .000$). In addition, the level of aesthetics of design was found to be significantly higher ($p = .010$) in artifacts created under constructivist teacher guidance. It should also be noted that, unlike the analysis of all the

Table 4. Differences Between Quality Parameters of Student Artifacts as a Function of Teacher Pedagogical Perspective (Instructivist Versus Constructivist).

| Parameter | Teacher type | N   | Mean | SD  | t-test |
|-----------|--------------|-----|------|-----|--------|
|           | Constructivist | 50  | 10.22| 13.403|        |
| 1. Quality of programming (Dr. Scratch evaluation) | Instructivist | 46  | 11.00| 4.201 | $t(94) = 1.050, p = .297$ |
|           | Constructivist | 50  | 10.14| 3.828 |        |
| 2. Quality of programming (expert evaluation) | Instructivist | 13  | 13.77| 2.279 | $t(21) = 2.758, p = .012$ |
|           | Constructivist | 10  | 10.60| 3.239 |        |
| 3. Clarity of the idea | Instructivist | 46  | 3.37 | 1.404 | $t(94) = -3.370, p = .001$ |
|           | Constructivist | 50  | 4.20 | 0.99  |        |
| 4. Use of medium | Instructivist | 46  | 4.11 | 1.303 | $t(94) = 1.507, p = .135$ |
|           | Constructivist | 50  | 3.66 | 1.586 |        |
| 5. Creativity & originality | Instructivist | 46  | 3.02 | 1.022 | $t(94) = -2.801, p = .006$ |
|           | Constructivist | 50  | 3.62 | 1.067 |        |
| 6. Aesthetics of design | Instructivist | 46  | 3.41 | 1.107 | $t(94) = -1.808, p = .074$ |
|           | Constructivist | 50  | 3.80 | 0.99  |        |
| 7. Target user’s activity | Instructivist | 46  | 0.96 | 0.729 | $t(94) = 0.228, p = .820$ |
|           | Constructivist | 50  | 0.92 | 0.829 |        |
| 8. Connection to the curriculum | Instructivist | 45  | 0.40 | 0.495 | $t(92) = 0.947, p = .346$ |
|           | Constructivist | 49  | 0.31 | 0.466 |        |
| 9. Number of scripts | Instructivist | 46  | 53.85| 127.441| $t(94) = 1.043, p = .300$ |
|           | Constructivist | 50  | 33.42| 52.086|        |
| 10. Number of sprites | Instructivist | 46  | 15.52| 41.179| $t(94) = 0.862, p = .391$ |

Bold values represents statistically significant.
artifacts, the analysis of “expert” student products did not show significant differences between the two groups in the level of quality of programming.

Moreover, in the artifacts of the “expert” students under constructivist teacher guidance, the level of target user’s activity was significantly higher ($p = .033$) compared to those that were created under instructivist teacher guidance. In the context of the third research question, this finding may suggest that guidance of teachers holding a constructivist perspective has an impact on the pedagogical perspective of students, who express this perspective by their artifacts created in the environment for visual programming. Students of these teachers tend to create artifacts that allow learners, i.e., the end users, to be active and construct their knowledge more actively in accordance with constructivist pedagogy (Olson & Bruner, 1996).

Figure 6 illustrates the difference between artifacts created under two pedagogical perspectives. On the right an artifact of the “Quiz” type, created under instructivist guidance, is displayed. This type of artifact is found to be very...
common and allows the learner a relatively low level of activity, namely, choosing between correct/incorrect answers. In contrast, on the left is an "Escape room" type product created under constructivist guidance. This is an original and rarely found artifact type, that requires the learner – the end user - to be very active in the game and demonstrates thinking “out of the box”. In this way, students may construct their knowledge in a better way.

**Discussion**

This study examined teaching-learning processes in the Scratch visual programming environment from the perspective of teachers who integrate visual programming in various formal frameworks. Unlike previous studies, which focused mostly on learners in non-formal learning environments (e.g., Luo et al., 2020), the current study focused on teaching strategies and on the perspectives of teachers. Moreover, this is a pioneering study that explores teaching perspectives and strategies in terms of Bruner’s “folk psychology” and “folk pedagogy” framework (Olson & Bruner, 1996). In this context, the study examined teachers’ pedagogical perspectives and the expression of these perspectives in teaching visual programming. In addition, this study goes beyond previous works, in that we explored differences in the quality of student artifacts. Also, the activity of the end users of these artifacts was examined as a function of teacher perspective. Finally, we examined possible implications of teachers “folk pedagogy” on their students’ “folk pedagogy”.

**Concerning the first research question:** It was found that there is a correlation between self-reporting of teaching-learning perspectives in the folk pedagogy questionnaire and the characterization of teacher strategies employed in the classroom and detailed in the interviews. The coding of the interviews showed
that both perspectives were reflected in the accepted teaching strategies presented by teachers with different perspectives. But the implementation of constructivist strategies, such as encouraging learning from experience, peer learning, and active participation of the learner in the learning process, was significantly higher ($\chi^2 = 22.101 \ p < .001$) among teachers with a constructivist perspective compared to their colleagues with an instructivist perspective. This finding is consistent with findings from previous studies (Olson & Bruner, 1996; Peled et al., 2015), which show that teachers apply their pedagogical perspective even when the teaching-learning process takes place in an innovative digital environment, such as a visual programming environment. However, it seems that teaching in the visual programming environment allows teachers with an instructivist perspective a relatively extensive use (about 50%) of constructivist strategies. These findings expand and add to the findings of previous studies (Lawrence & Tar, 2018; Peled et al., 2015), which show that innovative technology alone cannot change current pedagogy. However, a visual programming environment, such as Scratch, is unique in its openness to learning through experience and active construction of knowledge (Kong et al., 2020; Resnick & Rosenbaum, 2013; Zuckerman et al., 2009). We assume that it also enables teaching through active and constructivist ways and helps assimilate diversity in teaching strategies, as found in this study.

In this study we documented the stages where teachers with instructivist perspectives use constructivist pedagogy and articulate this in the interviews. However, their perspectives, i.e., cognitive schema, through which teachers approach learning processes, remain instructivist. The process documented in this study, in which a teacher uses strategies of current constructivist teaching when his/her declared pedagogical perspective is still traditional teaching, is consistent with Vygotsky et al.’s (1994) claim. According to their claim, the new cognitive schema is assimilated into cognition only after the individual uses the new term learned by him in practice at an applied performance level and over time.

Our findings suggest that teachers with an instructivist perspective first make changes in their actual pedagogy, while their stated perspective (folk psychology) remains stable. Following a change in pedagogy, folk psychology eventually may also be refined. Only then may teachers refine their perspective of evaluation, which requires a deep pedagogical understanding and a good acquaintance with the environment in which they teach (Nevo, 2002). Figure 7 presents this process, which shows a gradual transition from a traditional instructivist pedagogical approach to a constructivist pedagogical approach.

It seems that integrating visual programming environments in schools may increase the use of constructivist pedagogy, even among teachers with traditional instructivist perspectives, and it is possible that in doing so they will, over time, change their pedagogical perspective accordingly. The latter change is still waiting to be empirically demonstrated in future longitudinal studies, but, based
on the pedagogical shift found in this study, we recommend encouraging teaching in the visual programming environment in formal education.

**Concerning the second research question**, this study shows that there are significant differences between artifacts of students who study in an environment for visual programming under the guidance of teachers with instructivist versus constructivist pedagogical perspective. This finding is consistent with both the assumption of the four models of theory of mind (Olson & Bruner, 1996) that differences in teacher perspectives lead to different learning outcomes among students, and also with Papert’s assumption that the way visual programming is used as a learning tool in formal education depends, to a considerable extent, on the teacher’s approach (Papert, 1996).

From this study it seems that in the beginning of the process, artifacts of students guided by instructivist teachers were found to be higher in their programming level quality, compared to the products of students guided by constructivist teachers. It can be assumed that teachers with an instructivist perspective tend to explain explicitly and demonstrate to their students how to create code, and thus, the programming level of the artifacts they produce in the early stages of learning is higher than those of students who are guided by constructivist pedagogy and are required to experiment and create codes independently. This is consistent with the perspective underlying the first and second models of theory of mind, according to which optimal learning occurs through demonstrations and explanations (Olson & Bruner, 1996).

Nevertheless, it seems that students who learn by constructivist guidance and over time gain experience in visual programming, close the gap with students learning under instructivist guidance in their ability to create an artifact at a good programming level. It can be assumed that ongoing constructivist guidance emphasizing active learning, encourages students to choose high standards...
for their projects (Nouri et al., 2020; Resnick, 2012;) and, as a result, create artifacts with a high quality of programming. This finding is consistent with Olson and Bruner’s (1996) claim that constructivist pedagogical approaches, resulting from the third and fourth models of theory of mind, will ultimately lead students to overall and broader success in the learning process, although this does require additional time. Also, in this finding there is evidence that continuous learning through personal and independent experience, which is widely expressed in constructivist guidance, improves the quality of programming and results in artifacts that require rich programming knowledge (Resnick, 2012).

Equally important, artifacts created by constructivist guidance were found to be more original and creative, designed more aesthetically, and expressed an idea more clearly in comparison to the artifacts of students learning under instructivist guidance. After six months of experience under constructivist guidance, students were able to program with a base of wider personal knowledge and experience and thus arrived at more original and creative artifacts than students learning mostly from teacher demonstrations. This finding suggests that teachers with a constructivist perspective are able to take advantage of the nature of the Scratch environment, enabling their students to create artifacts in a personally meaningful and unique context (Maloney et al., 2010; Nouri et al., 2020; Resnick, 2012).

The third research question explored whether and how the pedagogical strategies of teachers with instructivist and constructivist perspectives are reflected in the degree of active learning conducted by the end users of student artifacts. It was found that the artifacts of “expert” students who are guided by teachers with constructivist perspectives, not only reflected programming quality as well as artifacts of students guided by instructivist teachers, but also allowed the end users to learn actively. This finding can be shown by previous results that teachers with constructivist perspectives make maximum use of the active learning which is afforded by the Scratch environment (Resnick & Rosenbaum, 2013). Our findings demonstrated that constructivist teachers promote active learning of their students to a larger extent compared to instructivist teachers (see Figure 3). It seems that teachers function as role models, implicating that continuous active learning in visual programming may lead to students’ understanding that this is an appropriate way to learn in such an environment (Hadad et al., 2020). Thus, when they design their programming artifacts, they project this understanding of learning on the end users, making them actively construct their knowledge. In other words, these students actually adopt the third and fourth models of folk psychology and pedagogy, i.e., the models belonging to the constructivist conception of the theory of mind (Olson & Bruner, 1996; Peled et al., 2015) held by their teachers. This finding may support what has already been hinted at in a previous study (Ke, 2014; Nouri et al., 2020) and suggests that guidance of teachers with a constructivist perspective affects students’ pedagogical perceptions, which are expressed in artifacts created in a Scratch environment.
Conclusion, Implications, Limitations, and Future Directions

This study found that the pedagogical perspectives of teachers have an impact on both the teaching strategies of these teachers in a visual programming environment and on the artifacts of students learning in this environment under different pedagogical guidance. In this study, we also found evidence that the pedagogical perspective of teachers may have an impact on the pedagogical perspective of their students and thus, even improve the artifacts they create. Moreover, teachers’ pedagogical perspectives are reflected in the activity level of students’ programming artifacts and thus, have implications on the active learning level of the end users. At the same time, it has been found that continuous experience in teaching in an environment for visual programming makes teaching approaches more flexible; namely, it increases the use of constructivist teaching strategies in the formal learning system.

This study adds to research literature by providing an empirical analysis of the “four models of theory of mind” approach (Bruner & Olson, 1996) in a new context of visual programming. Moreover, the study contributes to our understanding of the connection to and transfer of what Brunner calls “folk pedagogy” from teachers to learners (which is reflected in the quality parameters of programming artifacts created by students and implied by the activity level of the end users of these learning artifacts).

Regarding educational implications, the study reveals which teaching strategies are used by teachers in the classroom when teaching in a visual programming environment, which strategies promote computational thinking, and which strategies are recommended for the training of teachers who plan to work in the field of visual programming, in order to enable them to design the best learning practices in this environment. As for students, we recommend exploring in future studies whether computational thinking in general and specific parameters of students’ artifacts, such as creativity and originality which are developed in the context of visual programming, can be transferred to a different context.

This study took place six months after the initiation of the code curriculum based on virtual programming environments in Israeli schools and focused on the Scratch online environment. We recommend performing follow-up studies that will examine diverse environments for visual programming and include additional research methods. These studies should include observations of teaching strategies and students’ learning processes in the classroom with reference to pedagogical perspectives held by teachers and assessment of student programming artifacts.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.
Funding
The authors disclosed receipt of the following financial support for the research, authorship, and/or publication of this article: This study was supported by the Research Authority Foundation, The Open University of Israel.

ORCID iD
Ina Blau https://orcid.org/0000-0001-5695-7221

References
Avidov-Ungar, O., Shamir-Inbal, T., & Blau, I. (2020). Typology of digital leadership roles tasked with integrating new technologies into teaching: Insights from metaphor analysis. *Journal of Research on Technology in Education*, 1–16. https://doi.org/10.1080/15391523.2020.1809035

Bakar, M. A., Mukhtar, M., & Khalid, F. (2019). The development of a visual output approach for programming via the application of cognitive load theory and constructivism. *Development*, 10(11), 305–312.

Blau, I., & Benolol, N. (2016). Can designing self-representations through creative computing promote an incremental view of intelligence and enhance creativity among at-risk youth? *Interdisciplinary Journal of e-Skills and Lifelong Learning (IJELL)*, 12, 267–278.

Blau, I., Grinberg, R., & Shamir-Inbal, T. (2018). Pedagogical perspectives and practices reflected in metaphors of learning and digital learning of ICT leaders. *JComputers in the Schools*, 35(1), 32–48.

Blau, I., & Pieterse, E. (2015). *Teacher’s beliefs about learning questionnaire: Measuring folk psychology and fold pedagogy* [Unpublished manuscript]. Department of Education and Psychology, The Open University of Israel.

Blau, I., & Shamir-Inbal, T. (2021). Writing private and shared annotations and lurking in Annoto hyper-video in academia: Insights from learning analytics, content analysis, and interviews with lecturers and students. *Educational Technology Research and Development*. https://doi.org/10.1007/s11423-021-09984-5

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* (pp. 1–25).

Calder, N. (2010). Using Scratch: An integrated problem-solving approach to mathematical thinking. *Australian Primary Mathematics Classroom*. 15, 9–14.

Charmaz, K. (2014). *Constructing grounded theory*. Sage.

Ching, Y. H., Hsu, Y. C., & Baldwin, S. (2018). Becoming an online teacher: An analysis of prospective online instructors’ reflections. *Journal of Interactive Learning Research*, 29(2), 145–168.

Hadad, S., Shamir-Inbal, T., Blau, I., & Leykin, E. (2020). Professional development of code and robotics teachers through Small Private Online Course (SPOC): Teacher centrality and pedagogical strategies to promote computational thinking of students. *Journal of Educational Computing Research*. Advance online publication. https://doi.org/10.1177/0735633120973432
Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296–310.

Hu, Y., Chen, C. H., & Su, C. Y. (2021). Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis. *Journal of Educational Computing Research, 58*(8), 1467–1493.

Israeli Ministry of Education. (2019). *Educational cloud*. [http://sites.education.gov.il/cloud/home/tikshuv/Pages/olimpprogram.aspx](http://sites.education.gov.il/cloud/home/tikshuv/Pages/olimpprogram.aspx) [in Hebrew].

Jacob, S. R., & Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration, 1*(1), 1–21. [https://doi.org/10.26716/jcsi.2018.01.1.1](https://doi.org/10.26716/jcsi.2018.01.1.1)

Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM, 59*(8), 26–27.

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education, 73*, 26–39.

Kong, S. C., Lai, M., & Sun, D. (2020). Teacher development in computational thinking: Design and learning outcomes of programming concepts, practices and pedagogy. *Computers & Education, 151*, 103872. [https://doi.org/10.1016/j.compedu.2020.103872](https://doi.org/10.1016/j.compedu.2020.103872)

Lawrence, J. E., & Tar, U. A. (2018). Factors that influence teachers’ adoption and integration of ICT in teaching/learning process. *Educational Media International, 55*(1), 79–105.

Luo, F., Antonenko, P. D., & Davis, E. C. (2020). Exploring the evolution of two girls’ conceptions and practices in computational thinking in science. *Computers & Education, 146*, 103759. [https://doi.org/10.1016/j.compedu.2019.103759](https://doi.org/10.1016/j.compedu.2019.103759)

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education (TOCE), 10*(4), 16. [http://doi.acm.org/10.1145/1868358.1868363](http://doi.acm.org/10.1145/1868358.1868363).

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia, 46*, 1–23. [https://www.um.es/ead/red/46/](https://www.um.es/ead/red/46/)

Nevo, D. (Ed.). (2002). *School-based evaluation: An international perspective*. Emerald Group Publishing Limited.

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. *Education Inquiry, 11*(1), 1–17. [https://doi.org/10.1080/20004508.2019.1627844](https://doi.org/10.1080/20004508.2019.1627844)

Olmo-Muñoz, J. del, Cózar-Gutiérrez, R., & González-Calero, J. A. (2020). Computational thinking through unplugged activities in early years of primary education. *Computers & Education, 150*, 103832. [https://doi.org/10.1016/j.compedu.2020.103832](https://doi.org/10.1016/j.compedu.2020.103832)

Olson, D. R., & Bruner, J. S. (1996). Folk psychology and folk pedagogy. In D. R. Olson & N. Torrance (Eds.), *Handbook of education and human development: New models of learning, teaching, and schooling* (pp. 9–27). Blackwell.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.
Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95–123.

Peled, Y., Blau, I., & Grinberg, R. (2015). Does 1:1 computing in a junior high school change the pedagogical perspectives of teachers and their educational discourse. *Interdisciplinary Journal of e-Skills and Lifelong Learning-IJELL, 11*, 257–271.

Peppler, K. A., & Warschauer, M. (2011). Uncovering literacies, disrupting stereotypes: Examining the (dis) abilities of a child learning to computer program and read. *International Journal of Learning and Media, 3*(3), 15–41.

Resnick, M. (2012). Point of view: Reviving Papert’s dream. *Educational Technology, 52*(4), 42.

Resnick, M., & Rosenbaum, E. (2013). Designing for tinkerability. In M., Honey, & D., Kanter (Eds.), *Design. Make. Play: Growing the next generation of STEM innovators* (pp. 163–181). Routledge.

Rich, P. J., Browning, S. F., Perkins, M., Shoop, T., Yoshikawa, E., & Belikov, O. M. (2019). Coding in K-8: International trends in teaching elementary/primary computing. *TechTrends, 63*(3), 311–329.

Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020). Computational thinking and mathematics using Scratch: An experiment with sixth-grade students. *Interactive Learning Environments, 28*(3), 316–327.

Shamir-Inbal, T., & Blau, I. (2020). Micro-learning in designing professional development for ICT teacher leaders: The role of self-regulation and perceived learning. *Professional Development in Education, 1–17*. https://doi.org/10.1080/19415257.2020.1763434

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

So, H. J., Jong, M. S. Y., & Liu, C. C. (2020). Computational thinking education in the Asian Pacific region. *The Asia-Pacific Education Researcher, 29*, 1–8. https://doi.org/10.1007/s40299-019-00494-w

Thibaut, L., Knipprath, H., Dehaene, W., & Depaepe, F. (2018). The influence of teachers’ attitudes and school context on instructional practices in integrated STEM education. *Teaching and Teacher Education, 71*, 190–205.

Tondeur, J., Van Braak, J., Ertmer, P. A., & Ottenbreit-Leftwich, A. (2017). Understanding the relationship between teachers’ pedagogical beliefs and technology use in education: A systematic review of qualitative evidence. *Educational Technology Research and Development, 65*(3), 555–575.

Vygotsky, L. S., van der Veer, R. E., Valsiner, J. E., & Prout, T. T. (1994). *The Vygotsky reader*. Basil Blackwell.

Wei, X., Lin, L., Meng, N., Tan, W., & Kong, S. C. (2020). The effectiveness of partial pair programming on elementary school students’ computational Thinking skills and self-efficacy. *Computers & Education, 160*, 104023. https://doi.org/10.1016/j.compedu.2020.104023

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717–3725.
Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607

Zuckerman, O., Blau, I., & Monroy-Hernández, I. (2009). Children’s participation patterns in online communities. *Interdisciplinary Journal of E-Learning and Learning Objects, 5*(1), 263–274.

**Author Biographies**

Avital Kesler holds a Master’s degree in Educational Technologies and Learning Systems. She is a teacher educator and a coordinator of teacher training and professional development programs. Her research interests include technology-enhanced pedagogy, gamification in learning, and development of computational thinking among students.

Tamar Shamir-Inbal holds a PhD in Technology and Science Education. She is a faculty member in the Department of Education and Psychology at the Open University of Israel and a member of the Teaching and Design (TeLTaD) research group at the University of Haifa, Israel. As a pedagogical coordinator and academic consultant for the Israeli Ministry of Education, she leads ICT professional development programs for school principals, ICT coordinators, and teachers. Her research interests focus on integration of innovative technologies in K-12, teachers’ professional development, and the role of design in technology-enhanced teaching and learning.

Ina Blau is a Professor (Full) of Educational Technology and Cyber-Psychology. She is the Dean of Academic Development and Learning Technologies and the Head of the Graduate Program in Educational Technologies and Learning Systems at the Open University of Israel. Her research interests include integration of innovative technologies in K-12 and academia, digital literacy competencies, computational thinking in visual programming and educational robotics, social aspects of e-communication, and psychological ownership in e-collaboration.