Criteria for evaluating digital technology used to support computational thinking via inquiry learning. The case of two educational software for mathematics and physics.

Aikaterini Bounou[1]*, Nikolaos Tselios[2], George Kaliampos[3], Konstantinos Lavidas[4], Stamatis Papadakis[5]

[1] PHD candidate, Department of Educational Science and Early Childhood Education, University of Patras, 26504 Patras, Greece aikabounou@sch.gr (https://orcid.org/0000-0001-8467-6371)

[2] Professor, Department of Educational Science and Early Childhood Education, University of Patras, 26504 Patras, Greece, , nitse@ece.upatras.gr (https://orcid.org/0000-0002-4454-2499)

[3] Assistant Professor, Department of Educational Science and Early Childhood Education, University of Patras, 26504 Patras, Greece, gkaliampos@upatras.gr (https://orcid.org/0000-0001-9902-144X)

[4] Assistant Professor, Department of Educational Science and Early Childhood Education, University of Patras, 26504 Patras, Greece, lavidas@upatras.gr (https://orcid.org/0000-0003-2225-1137)

[5] Assistant Professor, Department of Preschool Education, University of Crete, Greece, stpapadakis@uoc.gr (https://orcid.org/0000-0003-3184-1147)

* Correspondence: aikabounou@sch.gr

Abstract

There is an ongoing need to evaluate whether commonly used educational software effectively supports both inquiry-based learning and the development of computational thinking skills, which are key objectives in secondary STEM curricula. This research aims to establish criteria for characterizing digital technologies, such as modeling and simulation software, virtual laboratories, and microcosms, to ensure their suitability in supporting students' computational thinking through inquiry-based activities in STEM courses. The main criteria focus on six key areas (a) production of meaning, (b) support in problem formulation, (c) ability to easily manage processes, (d) support in expressing solutions, (e) support in executing and evaluating solutions, and (f) ability to articulate and reflect on processes and solutions. Using this evaluation framework, two widely used software tools, Tracker and GeoGebra, commonly employed in high school physics and mathematics, were assessed. The trial evaluation results are discussed, with recommendations provided for improving the software to better support these educational objectives.

Keywords: inquiry-based learning, computational thinking, digital technology, GeoGebra, Tracker

Introduction

Inquiry-based learning, and consequently inquiry-based teaching, is one of the challenges of recent years in STEM education. One of the privileged areas in which inquiry learning and teaching can be applied is experimental teaching in STEM subjects (Gormally, et.al., 2009; Harlen, 2013; Abdi, 2014; Lazonder, & Harmsen, 2016; Pedaste, et.al., 2019. Inquiry is mainly considered the cornerstone of science education (National Research Council US, 2000; 2011; Barrow, 2006), and inquiry activities and processes provide the context in which students can engage with and understand scientific concepts, acquire knowledge, and apply that knowledge. Inquiry teaching supports inquiry-based learning through which students approach and ultimately internalize the way scientific research is conducted. Inquiry-based teaching and learning can be divided into three main categories: guided, structured and open, based on the level of student and teacher involvement and students' experience of inquiry-based working methods (NRC 2000; 2011; Chin & Chia, 2004; Zion & Shedletzky, 2006; Sadeh & Zion, 2009; Zion & Mendelovici, 2012).

At the dawn of the 21st century, so-called 21st century skills have become the focus of education reform in many countries. In education, these skills are mainly summarized in the '4Cs': critical thinking and problem solving, communication, collaboration, creativity and innovation. In February 2018, an article by S. Grover is published entitled "The 5th 'C' of 21st Century Skills; Try Computational Thinking (Not Coding)", in which she argues that: "… computational thinking (CT) needs to be …the '5th C' of 21st century skills - taught to all students". But what is CT? There is general confusion among researchers and educational theorists about the definition of CT. However, all agree that key components of CT include abstract thinking, decomposition, synthesis, analysis, generalization, hierarchy, algorithmic thinking, among others (Bar & Stephenson, 2011; Aho, 2012; Grover & Pea, 2013; Yadav et.al., 2014; Shute et.al., 2017; Tang et.al., 2020). The above data has led several countries to attempt to introduce CT into primary and secondary school curricula. As far as Greece and the new curricula for high school education are concerned (http://iep.edu.gr/el/nea-programmata-spoudon-arxiki-selida retrieved October 23, 2023), CT is envisaged as the second core component, in addition to inquiry learning, which runs cross-cuttingly through all of STEM courses.

Several researchers (Pedaste, et.al., 2019; Hambrusch, et.al., 2009; Sulistiyo & Wijaya, 2022; Saad & Zainudin, 2022) have highlighted the importance of inquiry-based learning in the development of computational thinking skills through their studies. Skills such as strategy development, algorithmic thinking and problem solving are critical in inquiry learning processes. Furthermore, as the above researchers argue, computational thinking contributes to a better understanding of STEM courses and, conversely, these courses provide a framework for the development of computational thinking. Indirectly teaching computational thinking through inquiry processes such as projects and problem solving can enhance learning and improve students' skills (Pedaste, et.al., 2019; Saad, 2020; Saad & Zainudin, 2022).

There is a consensus in STEM education about the importance of providing students with learning experiences that require the development of higher order cognitive skills. Such learning experiences can occur when digital technology is used in the context of inquiry-based learning, and in particular in problem-solving processes. In general, there is a view among many researchers that STEM education is a privileged field for developing students' CT because, as Li et al. (2020) argue, CT is more relevant to specific thinking processes than to programming per se. CT involves the search for appropriate ways of processing information that are progressively improved in terms of their efficiency, correctness and elegance. The desired improvement requires the use of various strategies and skills such as decomposition, analysis, synthesis, practice, prioritization, algorithmic thinking, abstraction and modelling. According to Li et.al. (2020), it is imperative that school curricula and teaching integrate CT into the curriculum of courses, not only in computing but also in other STEM disciplines beyond computing.

The thinking process described by Li et al. (2020) is an integral part of inquiry learning. In the context of STEM education, inquiry processes can be introduced in many ways. One of these is the use of digital technology such as modelling and simulation software, microcosms and virtual laboratories. In the context of inquiry-based learning processes (Schwab & Brandwein, 1962; Barrow, 2006; Pedaste, et.al., 2015; 2019) and when teaching STEM courses, one of the ways to promote learning through inquiry and at the same time promote CT skills is to assign students to solve a problem that is first closed and then open, using digital technology. It should be noted that it would be advisable for the problem to be taken from the real world and to be as relevant as possible to the learners' experience. This inquiry-based learning process is delineated into three primary stages, as outlined in Table 1. The structure of this table is based on the work of Wagh et al. (2017), who developed a matrix aligning the elements of inquiry learning with students' interactions with code. In our adaptation, we created a matrix that maps the inquiry processes to the actions of trainees, highlighting the computational thinking (CT) practices employed during experimental activities or problem-solving tasks using digital media technology (see Table 1).

Table 1 Mapping of students' inquiry processes and actions using CT practices

Element of the inquiry process	Description of the inquiry process	CT elements in the inquiry process (student actions)
Trying to solve a problem (open or closed)	Formulate questions to be answered or hypotheses to be confirmed Plan and carry out an inquiry by following specific steps/procedures (algorithmic thinking)	Decomposition of the main question/problem and formulation of sub-questions Abstract choice of the representation of the object / phenomenon to be studied

		Strategic selection (prioritization) of variables to be studied or changed. Reason for the choice of variables (decomposition) and prediction of the effects of their changes (subtraction). Ignore variables that do not affect the problem (prioritization)
Making sense of exploration	Analyze data and synthesize results to develop claims and formulate evidence-based explanations, observe results, formulate evidence-based claims, propose theoretical mechanisms or conceptual models to help predict and explain results.	Change variables and re-run process (algorithmic thinking) Observe and record the results for each re-run (algorithmic thinking). Compare results for each variable change (cause and effect) Extract conclusions about software results from each iteration (analysis) Formulate a general claim (synthesis) Predict underlying (physical) law or meaning (generalization)
Engagement within the learning community	Student teamwork Develop a reasoned argument based on evidence about the claims made (articulation and reflection).	Collaborative exploration Mutual assistance Set up a working team Share findings/conclusions

Educational software such as simulations, virtual laboratories, microcosmos, and programming environments, can be used to support CT based on inquiry learning (Chen, et.al., 2010; Lewis, 2014; Potkonjak, et.al., 2016; De Jong, 2019). There are several software and virtual laboratories that are commonly used by secondary school teachers worldwide, such as Tracker (physics, mathematics), GeoGebra (mathematics, physics), Chem Collective's VLab (chemistry), Algodoo (physics) and many more. In light of the above, the question arises as to whether these digital tools are suitable for inquiry-based learning, while at the same time promoting the computational thinking skills that are key objectives of the secondary school curricula in STEM subjects in many countries.

Initially, the present study is an attempt to answer the above question, focusing on structuring a set of criteria that should characterize digital media technology in order to be considered suitable for exploratory inquiry-based learning and the promotion of CT skills, with the help of two important publications by Quintana et al. (2004) and Reppening et al. (2016; 2017). The Quintana et.al. (2004) software evaluation framework is limited to evaluating the ability of software to be used in inquiry processes in STEM education, while the Reppening et.al. (2016) framework is primarily concerned with evaluating programming environments in terms of their ability to promote students' computational thinking through their use.

According to the above, we aimed to synthesize and extend these two assessment frameworks to evaluate the ability of digital technologies to support both inquiry-based learning and the development of students' computational thinking. This aim arose in light of the suggestion by many researchers that computational thinking is not an exclusive subject of computer science, but that it would be appropriate to teach it as a way of thinking in the context of STEM courses (Li, et.al., 2020; Saad, 2020; Tang, et.al., 2020; Poulakis, & Politis, 2021).

Using the software evaluation tool we produce, an attempt was made to evaluate software GeoGebra and Tracker, which are widely used in both mathematics and physics in secondary education. In this phase, the researcher interacted with GeoGebra and Tracker software by addressing scenarios or problems derived from curriculum materials, as specified in the physics and mathematics syllabi. The researcher was responsible for executing the implementation of these scripts and problems. During this process, comprehensive records were maintained, documenting the actions performed by the researcher in solving each scenario or problem.

Software Evaluation Tool

Over the last twenty years, the use of digital technology, especially in STEM courses, has increased due to the improvement of school infrastructure, the large supply of digital technology, and the higher level of training of teachers in new technologies (Kim, et.al., 2018). But what is the purpose of using digital technology? First of all, it enables teachers to visualize phenomena that they would probably not be able to introduce in the classroom in any other way, except through the now-obsolete lecture. And as far as the science subjects (physics, chemistry, and biology) are concerned, even phenomena that are relatively easy to study in the science laboratory can be introduced in an even easier and more manageable way through digital technology (DeHaan, 2005; Ernerfeldt, 2008; De Jong et.al., 2013; De Jong et.al. Kim, et.al., 2018; Da Silva et.al., 2014; De Jong, et.al., 2014; Euler, et.al., 2020). A similar phenomenon can be observed in mathematics in terms of problem-solving practice, which is a practice of increasing interest in mathematics education (Hiebert, et.al., 1996; Gravemeijer & Doorman, 1999; Polya, 2004). In addition, digital technology allow the teacher to involve the learner directly in the study of the phenomenon in question. Through the use of appropriate media, the learner interacts directly with the phenomenon, can intervene in its evolution, can study the phenomenon at the level of the factors on which it depends, can return to earlier stages of the evolution of the phenomenon to study them in-depth, can intervene in the mechanisms of the evolution of the phenomenon and, most importantly, can receive feedback from the medium itself in terms of its actions. In this context, it is now argued that the use of digital technology can promote students' computational thinking (Hu, 2011; Arastoopour, et.al., 2020; Apiola & Sutinen, 2021; Kallia, et.al., 2021) by enhancing key skills that are components of computational thinking, such as analysis, synthesis, abstraction, hierarchy, algorithmic thinking, debugging, decomposition as well as problem synthesis, data manipulation, and problem-solving.

One of the key issues is the use of appropriate digital technology both to support the inquiry teaching of STEM subjects and to promote basic computational thinking skills through them in the most appropriate and targeted way. It would therefore be necessary to structure a set of criteria by which even a non-specialist in digital technology could judge whether the particular tool he or she is using can best meet the above requirements. Attempts to structure such a set of criteria are rare in international literature. In the literature review we found two evaluation frameworks related to the evaluation of digital technology relevant to STEM courses and the promotion of computational thinking through them. These evaluation frameworks have been concerned either with the ability of the media to support inquiry-based learning (Quintana et.al., 2004) or with the ability of programmatic environments to support computational thinking (Reppening et.al., 2016; 2017). The aim of this study is to synthesize and at the same time update these evaluation frameworks so that they can be used to simultaneously evaluate digital technology in terms of their ability to support inquiry-based learning on the one hand, and to promote computational thinking on the other. This objective arose from the fact that, as many researchers argue, STEM courses are a privileged field for potentially enhancing students' computational thinking and, in addition, a privileged field for applying the principles of inquiry learning.

The first evaluation framework is found in the research of Quintana et.al. (2004), which presents a well-structured framework of criteria that must be met in order to design software that supports inquiry processes in STEM courses. Quintana et.al.'s research is based on the theoretical approach of analyzing the principles that should underline the software under study and is not simply limited to the observations that can be derived from inductive analysis/investigation of software. Their research has resulted in three main strands which form the framework of criteria they have structured. According to them the software should:

- Be meaningful to learners
- Enable learners to manage research-related processes in STEM courses; and finally
- Enable learners to articulate and reflect.

Each of these axes is broken down into individual guidelines, which are composed of specific constructivist/research strategies.

The second evaluation framework is found in the research by Reppening et.al. (2016; 2017), according to which the three stages in the process of computational thinking that should be supported by the computational systems in question are:

- Problem formulation (abstraction), i.e. making sense of a problem using either spoken/written discourse or visual representations.
- The articulation/expression of the solution (automation) by the software in an unambiguous way.
- Executing the solution in such a way that the consequences of the learner's thinking become apparent and evaluating the solution.

In the article by Reppening et.al. (2016; 2017), these stages are specified in detail by individual processes to be supported by the software.

The aim of this paper is to synthesize the criteria mentioned in the above works in order to structure a framework of assessment and design criteria for digital technology that relate to STEM courses and, at the same time, support the promotion of basic components of computational thinking and inquiry-based learning.

In a first step, our aim was to synthesize the main axes of the evaluation frameworks of Quintana et.al. (2004) and Reppening et.al. (2016; 2017). After a thorough exploration of the components of these axes, we concluded that the axes set out in the framework of Quintana et.al. are of primary importance, as they represent the main aspirations of both software and simulation and programming environments through which computational thinking can be fostered. Figure 1 illustrates the axes of the evaluation framework, divided into primary and secondary axes, and the relationships between them.

The overriding requirement (primary axis A) of software, simulations and programming environments in the context of STEM education is the ability to generate meaning from their use and especially while using them (Quintana, et.al, 2004). The desired outcome for students/users who encounter and use these tools in STEM courses is both to understand and internalize the basic skills of scientific inquiry and inquiry-based learning, and to develop basic computational thinking skills. It is therefore obvious that at each step in their use of these computational tools they should, on the one hand, identify the scientific tools/concepts they are using, for example, why these tools/concepts and not some others, and, on the other hand, be able to use these tools/concepts to identify the necessary scientific practices they should use to make sense of their investigation.

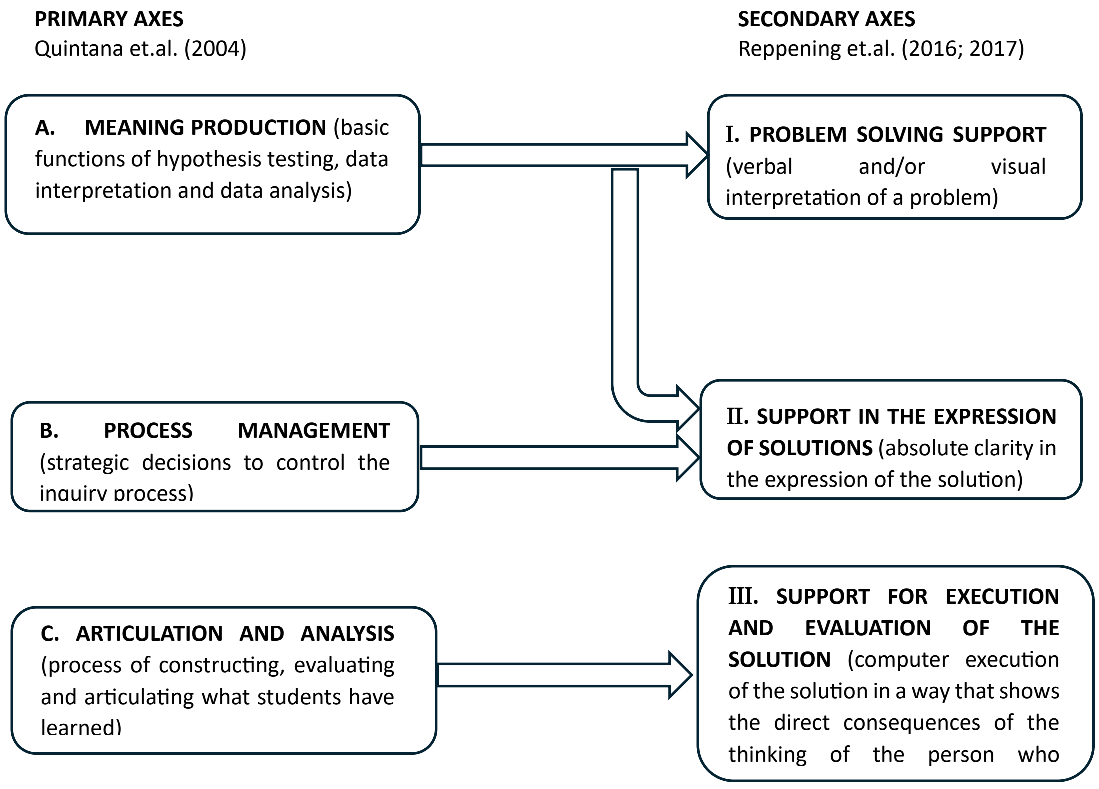

PRIMARY AXES
Quintana et.al. (2004)
SECONDARY AXES
Reppening et.al. (2016; 2017)
A. MEANING PRODUCTION (basic functions of hypothesis testing, data interpretation and data analysis)
I. PROBLEM SOLVING SUPPORT (verbal and/or visual interpretation of a problem)
B. PROCESS MANAGEMENT (strategic decisions to control the inquiry process)
II. SUPPORT IN THE EXPRESSION OF SOLUTIONS (absolute clarity in the expression of the solution)
C. ARTICULATION AND ANALYSIS (process of constructing, evaluating and articulating what students have learned)
III. SUPPORT FOR EXECUTION AND EVALUATION OF THE SOLUTION (computer execution of the solution in a way that shows the direct consequences of the thinking of the person who

Fig. 1 Primary and Secondary Axes of the evaluation framework according to Quintana et al. (2004) and Reppening et al. (2016, 2017))

A secondary objective (secondary axis I), but not of secondary importance and directly related to primary axis A, especially about the development of basic computational thinking skills, is to support students/users in formulating the problem to be investigated (Reppening et al. 2016; 2017). Recognizing that most computational thinking skills are at the same time the skills required for the problem-solving process, students/users should be supported by the computational environment in formulating, conceptualizing, and defining the problem to be solved. The software or programming environment needs to provide users with appropriate support tools to help them define the problem to be solved, as many problems in STEM have several different aspects that can be distracting for students/users. In addition, the student/user should be supported in the process of transferring the problem to the computational environment to choose those abstractions that will lead to the best possible analysis of the problem into its components and thus to the success of its solution.

The second axis in the framework for evaluating computing environments for STEM courses, as listed in the study by Quintana et al. (2004), concerns the process management mechanisms that the computing environment should provide to the learner/user. These are mechanisms that should assist the learner/user in constructing a research plan for the problem to be solved. Once the user has passed the stage of formulating and defining the problem, he or she must draw up an action plan to efficiently implement the processes that will lead to a clear expression of the solution to the problem. In this phase, the user is asked to develop an appropriate strategy to express the solution to the problem as clearly as possible, minimizing his involvement in routine tasks that increase the complexity of the solution and are not intended either to solve the problem correctly or to promote the user's computational thinking. This is where the secondary axis II of supporting the representation of solutions according to Reppening et.al. (2016; 2017) comes in. In the framework we are trying to synthesize, this axis (secondary axis II) is directly related to the mechanisms of process management (primary axis B) and also to the need to produce meaning (primary axis A) as defined by Quintana et.al. (2004) and as we have analyzed when referring to them. Supporting the expression of solutions consists in supporting the learner/user with the appropriate structure of the computing environment, to have easy access to the tools needed to work on the problem to be solved, and in supporting the writing of the code that the user will be asked to structure in terms of programming environments.

Finally, to achieve the twofold objective of computational environments in STEM education, i.e. the promotion of inquiry-based learning and the cultivation and improvement of computational thinking, it is necessary to support the continuous articulation of the user's findings and constructs and the reflection on these findings (primary axis C) during the problem-solving process, as well as the execution of the solution and its evaluation (secondary axis III). The mentioned axes C and III are directly related to each other as they both refer to the last stage of the use of the computer system, so that the user should now have completed all the necessary steps to arrive at the answers he is looking for, i.e. the solution to the problem. To make this possible, the computer system should provide reminders where needed, guidance and facilitation on the scientific concepts and practices involved in the user's constructs and findings, and assistance in achieving productive reflection by the user on them. At the same time, both computer systems and programming systems should allow the user/learner to directly observe the consequences of any intervention in the process of solving the problem, i.e. the use should be supported by direct feedback on the processes followed to solve the problem on the one hand, and to produce meaning through it on the other.

Table 2 below lists the combined guidelines of the above axes (Quintana, et.al., 2004; Reppening et.al., 2016; 2017) and the strategies that should be followed by the computing systems to align them with the guidelines and axes of the evaluation framework.

Table 2 Guidelines and Strategies of the Digital Technology Evaluation Framework according to Quintana et.al. (2004) and Reppening et.al. (2016; 2017)

AXES	GUIDELINES	STRATEGIES
Axes A - I Meaning Production – Problem Solving Support	1. Supporting conceptualization through verbal and/or visual thinking	1a. Provide visual memory representations to enhance visual thinking
		1b. Provide common STEM problems with short texts or titles
	2. Support for the transfer of data via explicit abstractions	2a. Support in the selection of explicit abstractions
		2b. Provide details that the user can focus on, and guide the user with feedback
	3. Use of representations / language to bridge prior knowledge with understanding of the new topic	3a. Functionality through visual conceptual organization
		3b. Descriptions for complex concepts
		3c. Scientific guidance for the implementation of scientific content
	4. Organizing tools and objects around the meaning and terminology of the scientific object and problem	4a. Explicit scientific strategies in student-to-tool interactions
		4b. Clear and explicit scientific strategies used by students to construct artefacts

AXES	GUIDELINES	STRATEGIES
Axes A - II Meaning Production – Support in the Expression of Solutions	5. Representations that students can explore to reveal important properties of the underlying data	5a. Representations that can be examined to reveal underlying data properties
		5b. Students can explore multiple aspects of the same object or data
		5c. Flexible representations that can be directly manipulated by students
Axes B - II Process Management – Support in the Expression of Solutions	6. Accessible programming environments	6a. Help with writing the program using visual programming
		6b. Focus the programming environment on the meaning of the code and help correct syntax errors
	7. Minimize complexity	7a. Reducing the complexity of the computational space in which the investigation takes place, through procedures that are invisible to the user.
		7b. Guiding the user, perhaps using AI, to reduce the random complexity that can arise from their choices.
	8. Provide the necessary structure for complex tasks and functionality	8a. Limiting complex work by setting useful boundaries for students
		8b. Describe complex operations using ordered and unordered decomposition operations
		8c. Activity space limitation through use of functional modes
	9. Integration of expert guidance on scientific practices	9a. Integrate expert guidance to clarify the characteristics of scientific practice
		9b. Integrate expert guidance to highlight the rationale for scientific practices
	10. Automatic handling of unattractive routine work	10a. Automation of non-obvious parts of tasks to reduce cognitive demands
		10b. Facilitate the organization of work products
		10c. Facilitate navigation between tools and activities

AXES	GUIDELINES	STRATEGIES
Axes C - III Articulation and Analysis – Support for Execution and Evaluation of the Solutions	11. Support for debugging and modification of misconceptions in programming environments	11a. Reminders and guidance for code debugging
		11b. Live programming capability
	12. Enable users to immediately see the impact of any change	12a. Guidance on how to run the application/program immediately after changing
		12b. Helping to focus on and evaluate the differences in results after each change
	13. Facilitate ongoing articulation and reflection during research	13a. Reminders and guidance to facilitate production planning
		13b. Reminders and guidance to facilitate productive monitoring
		13c. Reminders and guidance to facilitate articulation and reflection during the meaning making process
		13d. Identifying the scientific characteristics of scientific practices and products

Methodology

This study seeks to evaluate the effectiveness of a software assessment tool by applying it to two widely used educational software programs in secondary education: GeoGebra and Tracker, in the fields of physics and mathematics. GeoGebra is an educational software designed to support the teaching and understanding of mathematical and engineering concepts, as well as to aid in solving complex problems. Tracker, on the other hand, is a video analysis and modeling software utilized for addressing engineering challenges found in secondary school physics curricula.

The evaluation process involved using scenarios and problems derived from established mathematics and physics curricula in secondary education. The researchers implemented these scenarios, documenting in detail the participants' actions as they engaged with the tasks to reach solutions. These actions were then analyzed and compared with the criteria set forth by the evaluation tool to assess the software's capabilities. The results of this evaluation, including detailed findings for both GeoGebra and Tracker, are presented in the appendix of this study.

Results

The results of the evaluation of GeoGebra and Tracker using the evaluation tool are summarized in Table 3. The evaluation tool demonstrated its effectiveness in assessing the software, as evidenced by the processes outlined in the Appendix. It yielded clear results regarding the software's potential to foster both inquiry-based learning and computational thinking. According to this evaluation, GeoGebra is evaluated as highly effective in supporting students' abstract thinking, in its structure and functionality, in reducing complexity, and in providing flexible representations. However, it has two significant shortcomings: it does not offer supplementary texts or representations to aid in the understanding of problems, and it does not provide support for scientific practices. Although these gaps can be mitigated by the teacher, the most notable shortcoming is the software's inability to support students in writing code. The evaluation of Tracker software shows that it meets key assessment criteria, being effective in supporting abstract thinking, organizing information, reducing complexity, and offering flexible representations. However, it lacks support for scientific principles and practices, like GeoGebra.

Table 3 Strategies fulfilled (✓) or not fulfilled (×) by GeoGebra and Tracker software

Strategies	GeoGebra	Tracker
1a. Provide visual memory representations to enhance visual thinking	×	✓
1b. Provide common STEM problems with short texts or titles	×	✓
2a. Support in the selection of explicit abstractions	✓	✓
2b. Provide details that the user can focus on, and guide the user with feedback	✓	✓
3a. Functionality through visual conceptual organization	✓	×
3b. Descriptions for complex concepts	×	×
3c. Scientific guidance for the implementation of scientific content	×	×
4a. Explicit scientific strategies in student-to-tool interactions	✓	✓
4b. Clear and explicit scientific strategies used by students to construct artefacts	✓	✓
5a. Representations that can be examined to reveal underlying data properties	✓	✓
5b. Students can explore multiple aspects of the same object or data	✓	✓
5c. Flexible representations that can be directly manipulated by students.	✓	✓
6a. Help with writing the program using visual programming	×	×
6b. Focus the programming environment on the meaning of the code and help correct syntax errors	×	×
7a. Reducing the complexity of the computational space in which the investigation takes place, through procedures that are invisible to the user.	✓	✓
7b. Guiding the user, perhaps using AI, to reduce the random complexity that can arise from their choices.	×	×
8a. Limiting complex work by setting useful boundaries for students	✓	✓
8b. Describe complex operations using ordered and unordered decomposition operations	✓	✓
8c. Activity space limitation through use of functional modes	✓	✓
9a. Integrate expert guidance to clarify the characteristics of scientific practice	×	×
9b. Integrate expert guidance to highlight the rationale for scientific practices	×	×
10a. Automation of non-obvious parts of tasks to reduce cognitive demands	✓	✓
10b. Facilitate the organization of work products	✓	✓
10c. Facilitate navigation between tools and activities	✓	✓
11a. Reminders and guidance for code debugging	×	×
11b. Live programming capability	×	×

Strategies	GeoGebra	Tracker
12a. Guidance on how to run the application/program immediately after changing	×	×
12b. Helping to focus on and evaluate the differences in results after each change	×	×
13a. Reminders and guidance to facilitate production planning	×	×
13b. Reminders and guidance to facilitate productive monitoring	×	×
13c. Reminders and guidance to facilitate articulation and reflection during the meaning making process	×	×
13d. Identifying the scientific characteristics of scientific practices and products	×	×

Conclusions

Inquiry-based teaching and learning have been introduced into secondary education in numerous countries for several years. However, the integration of Computational Thinking (CT) into the secondary curriculum remains a topic of debate in many countries, whether as a standalone subject, a component of Computer Science, or a cross-curricular topic. Research indicates that the use of appropriate software to facilitate inquiry processes through projects or problem-solving in STEM subjects (Barrow, 2006; Pedaste et al., 2015; 2019; Li et al., 2020) can significantly enhance the development of fundamental CT skills. A key prerequisite is that the digital medium employed must implement as many of the previously synthesized strategies as possible, according to the research of Quintana et al. (2004) and Reppening et al. (2016; 2017), to serve the dual purpose of fostering both inquiry-based learning and CT. This study combines and expands two assessment frameworks to evaluate how well digital technologies support inquiry-based learning and the development of students' CT. Using the developed evaluation tool, GeoGebra and Tracker software, commonly used in secondary math and physics, were assessed.

Concerning the GeoGebra software, it is important to highlight that the evaluation results align with the broader findings of Echeverría et al. (2019) and S. van Borkulo et al. (2021) on this software. Both the aforementioned researchers and the evaluation tool assess GeoGebra as highly effective in supporting students' abstract thinking, in its organizational structure and functionality, in reducing overall complexity—by relegating complex processes to the background—and in providing a wide range of adaptable representations. However, the GeoGebra software exhibits two significant deficiencies. Firstly, it does not furnish students or users with auxiliary texts or representations to facilitate understanding, articulation, and reflection on the problem at hand. Secondly, it lacks support in providing information on the scientific features and practices involved. While these gaps can be mitigated by the teacher's guidance, the most notable shortcoming is the software's inability to assist learners in writing code when required. A possible recommendation would be to enhance GeoGebra by incorporating live programming capabilities, supported by artificial intelligence, to enable secondary school students to write code and instantly observe the outcomes.

Similarly, the evaluation of Tracker software revealed that it generally satisfies the key criteria of the assessment tool. It is particularly effective in supporting abstract thinking, organizing information, reducing complexity, and offering a diverse set of flexible representations. However, like GeoGebra, Tracker does not provide users with supporting information on the scientific principles and practices utilized. In contrast to GeoGebra, Tracker offers memorable representations and a repository of STEM-related problems, yet it does not permit users to write code to influence the software's functionality. These observations are corroborated by the research of one of Tracker's creators (Brown, 2008; Brown & Cox, 2009) as well as by other scholars (Wee, 2014; Wee et al., 2012).

It would be prudent to apply the framework of criteria to evaluate other digital technologies used in STEM education at the secondary level, to identify potential shortcomings and reinforce these areas. By doing so, educators and developers can better understand the specific areas where these technologies may need reinforcement. Additionally, it is recommended to conduct in vivo trials with students engaging in various scenarios using the respective digital technologies. These trials would provide direct insights into how students interact with the technologies, helping to determine whether the tools effectively support inquiry-based learning and CT as outlined in the framework. Moreover, observing students' problem-solving processes and their ability to apply abstract concepts during these trials would offer critical feedback for improving the design and functionality of the tools. These observations could also highlight how different technologies facilitate or hinder students' cognitive development in STEM disciplines.

Finally, since CT is a cognitive skill applicable across all fields, particularly STEM, the established criteria could be used as a standard for designing new digital technologies. These benchmarks would guide the development of tools aimed not only at improving inquiry-based learning but also at fostering essential CT skills, ensuring that students are better prepared for the demands of modern education and future STEM careers.

Bibliographic References

References to International Literature

Abdi, A. (2014). The effect of inquiry-based learning method on students' academic achievement in science course. *Universal journal of educational Research*, 2(1), 37-41.

Aho, A. V. (2012). Computation and computational thinking. *The computer journal*, 55(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

Apiola, M., & Sutinen, E. (2021). Design science research for learning software engineering and computational thinking: Four cases. *Computer Applications in Engineering Education*, 29(1), 83-101. https://doi.org/10.1002/cae.22291

Arastoopour Irgens, G., Dabholkar, S., Bain, C., Woods, P., Hall, K., Swanson, H., Horn, M., & Wilensky, U. (2020). Modeling and measuring high school students’ computational thinking practices in science. *Journal of Science Education and Technology*, 29, 137-161. https://doi.org/10.1007/s10956-020-09811-1

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?. *Acm Inroads*, 2(1), 48-54.

Barrow, L. H. (2006). A brief history of inquiry: From Dewey to standards. *Journal of Science Teacher Education*, 17(3), 265-278.

Brown, D. (2008). Video modeling: combining dynamic model simulations with traditional video analysis. *In American Association of Physics Teachers* (AAPT) Summer Meeting.

Brown, D., & Cox, A. J. (2009). Innovative uses of video analysis. *The Physics Teacher*, 47(3), 145-150. https://doi.org/10.1119/1.3081296

Chen, X., Song, G., & Zhang, Y. (2010). Virtual and remote laboratory development: A review. Earth and Space 2010: *Engineering, Science, Construction, and Operations in Challenging Environments*, 3843-3852. https://doi.org/10.1061/41096(366)368

Chin, C., & Chia, L. G. (2004). Problem-based learning: Using students' questions to drive knowledge construction. *Science education*, 88(5), 707-727.
https://doi.org/10.1002/sce.10144

Da Silva, S. L., Da Silva, R. L., Junior, J. T. G., Gonçalves, E., Viana, E. R., & Wyatt, J. B. (2014). Animation with Algodoo: A simple tool for teaching and learning physics. arXiv preprint arXiv:1409.1621. https://doi.org/10.48550/arXiv.1409.1621

DeHaan, R. L. (2005). The impending revolution in undergraduate science education. *Journal of Science Education and Technology*, 14, 253-269. https://doi.org/10.1007/s10956-005-4425-3

De Jong, T., Linn, M. C., & Zacharia, Z. C. (2013). Physical and virtual laboratories in science and engineering education. *Science*, 340(6130), 305-308. https://doi.org/10.1126/science.1230579

De Jong, T., Sotiriou, S., & Gillet, D. (2014). Innovations in STEM education: the Go-Lab federation of online labs. *Smart Learning Environments*, 1(1), 1-16. https://doi.org/10.1186/s40561-014-0003-6

De Jong, T. (2019). Moving towards engaged learning in STEM domains; there is no simple answer, but clearly a road ahead. *Journal of computer assisted learning*, 35(2), 153-167. https://doi.org/10.1111/jcal.12337

Echeverría, L., Cobos, R., & Morales, M. (2019). Improving the students computational thinking skills with collaborative learning techniques. *IEEE Revista Iberoamericana de Tecnologias del Aprendizaje*, 14(4), 196-206. doi: 10.1109/RITA.2019.2952299.

Ernerfeldt, E. (2008). Phun-2d physics sandbox (Unpublished master's thesis) Umeå University, Sweden

Euler, E., Prytz, C., & Gregorcic, B. (2020). Never far from shore: productive patterns in physics students' use of the digital learning environment Algodoo. *Physics Education*, 55(4), 045015. DOI 10.1088/1361-6552/ab83e7

Gormally, C., Brickman, P., Hallar, B., & Armstrong, N. (2009). Effects of inquiry-based learning on students' science literacy skills and confidence. *International journal for the scholarship of teaching and learning*, 3(2), 16.
https://doi.org/10.20429/ijsotl.2009.030216

Gravemeijer, K., & Doorman, M. (1999). Context problems in realistic mathematics education: A calculus course as an example. *Educational studies in mathematics*, 39(1-3), 111-129. https://doi.org/10.1023/A:1003749919816

S. Grover, The 5th 'C' of 21st Century Skills? Try Computational Thinking (Not Coding), Retrieved March 28, 2023 from https://www.edsurge.com/news/2018-02-25-the-5th-c-of-21st-century-skills-try-computational-thinking-not-coding , 2018

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, 42(1), 38-43. https://doi.org/10.3102/0013189X12463051

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *ACM Sigcse Bulletin*, 41(1), 183-187. https://doi.org/10.1145/1539024.1508931

Harlen, W. (2013). Inquiry-based learning in science and mathematics. *Review of science, mathematics and ICT education*, 7(2), 9-33.
https://doi.org/10.26220/rev.2042

Hiebert, J., Carpenter, T. P., Fennema, E., Fuson, K., Human, P., Murray, H., Olivier, A., & Wearne, D. (1996). Problem solving as a basis for reform in curriculum and instruction: The case of mathematics. *Educational researcher*, 25(4), 12-21.
https://doi.org/10.3102/0013189X025004012

Hohenwarter, M. (2002). GeoGebra-a software system for dynamic geometry and algebra in the plane (Unpublished master's thesis) University of Salzburg, Austria.

Hu, C. (2011, June). Computational thinking: what it might mean and what we might do about it. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education* (pp. 223-227).

Kallia, M., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Characterising computational thinking in mathematics education: a literature-informed Delphi study. *Research in mathematics education*, 23(2), 159-187. https://doi.org/10.1080/14794802.2020.1852104

Kim, N. J., Belland, B. R., & Walker, A. E. (2018). Effectiveness of computer-based scaffolding in the context of problem-based learning for STEM education: Bayesian meta-analysis. *Educational Psychology Review*, 30, 397-429. https://doi.org/10.1007/s10648-017-9419-1

Lazonder, A. W., & Harmsen, R. (2016). Meta-analysis of inquiry-based learning: Effects of guidance. *Review of educational research*, 86(3), 681-718.
https://doi.org/10.3102/0034654315627366

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

National Research Council. (2000). *Inquiry and the national science education standards: A guide for teaching and learning*. National Academies Press.

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking.* National Academies Press.

National Research Council US. (2011) *Report of a workshop on the pedagogical aspects of computational thinking*, Washington DC, National Academies Press.

National Research Council. (2011). *Successful K-12 STEM education: Identifying effective approaches in science, technology, engineering, and mathematics*. National Academies Press.

Pedaste, M., Mäeots, M., Siiman, L. A., De Jong, T., Van Riesen, S. A., Kamp, E. T., Manoli, C.C., Zacharia, C.Z. & Tsourlidaki, E. (2015). Phases of inquiry-based learning: Definitions and the inquiry cycle. *Educational research review*, 14, 47-61. https://doi.org/10.1016/j.edurev.2015.02.003

Pedaste, M., Palts, T., Kori, K., Sõrmus, M., & Leijen, Ä. (2019, July). Complex problem solving as a construct of inquiry, computational thinking and mathematical problem solving. In *2019 IEEE 19th International Conference on Advanced Learning Technologies* (ICALT) (Vol. 2161, pp. 227-231). IEEE. https://doi.org/10.1109/ICALT.2019.00071

Polya, G. (2004). *How to solve it: A new aspect of mathematical method* (Vol. 85). Princeton university press.

Potkonjak, V., Gardner, M., Callaghan, V., Mattila, P., Guetl, C., Petrović, V. M., & Jovanović, K. (2016). Virtual laboratories for education in science, technology, and engineering: A review. *Computers & Education*, 95, 309-327. https://doi.org/10.1016/j.compedu.2016.02.002

Poulakis, E. & Politis, P., Computational thinking assessment: literature review, *Res. on E-Learning and ICT in Educ.* (2021) 111-128. https://doi.org/10.1007/978-3-030-64363-8_7

Quintana, C., Reiser, B. J., Davis, E. A., Krajcik, J., Fretz, E., Duncan, R. G., Kyza, E., Edelson, D., & Soloway, E. (2018). A scaffolding design framework for software to support science inquiry. In *Scaffolding* (pp. 337-386). Psychology Press.

Repenning, A., Basawapatna, A. R., & Escherle, N. A. (2017). Principles of computational thinking tools. *Emerging research, practice, and policy on computational thinking*, 291-305. https://doi.org/10.1007/978-3-319-52691-1_18

Repenning, A., Basawapatna, A., & Escherle, N. (2016, September). Computational thinking tools. In 2016 *IEEE Symposium on Visual Languages and Human-centric Computing* (VL/HCC) (pp. 218-222). IEEE. https://doi.org/10.1109/VLHCC.2016.7739688

Saad, A. (2020). Students' computational thinking skill through cooperative learning based on hands-on, inquiry-based, and student-centric learning approaches. *Universal Journal of Educational Research*, 8(1), 290-296. DOI: 10.13189/ujer.2020.080135

Saad, A., & Zainudin, S. (2022). A review of Project-Based Learning (PBL) and Computational Thinking (CT) in teaching and learning. *Learning and Motivation*, 78, 101802. https://doi.org/10.1016/j.lmot.2022.101802

Sadeh, I., & Zion, M. (2009). The development of dynamic inquiry performances within an open inquiry setting: A comparison to guided inquiry setting. *Journal of Research in Science Teaching: The Official Journal of the National Association for Research in Science Teaching*, 46(10), 1137-1160. https://doi.org/10.1002/tea.20310

Schwab, J. J., & Brandwein, P. F. (1962). *The teaching of science*. Harvard University Press.

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational research review*, 22, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003

Sulistiyo, M. A. S., & Wijaya, A. (2020, July). The effectiveness of inquiry-based learning on computational thinking skills and self-efficacy of high school students. *In Journal of Physics*: Conference Series (Vol. 1581, No. 1, p. 012046). IOP Publishing. DOI 10.1088/1742-6596/1581/1/012046

Tang, X.; Yin, Y.; Lin, Q.; Hadad, R.; Zhai, X. Assessing computational thinking: A systematic review of empirical studies. *Comp. Educ.* 2020, 148, 103798. https://doi.org/10.1016/j.compedu.2019.103798.

van Borkulo, S., Chytas, C., Drijvers, P., Barendsen, E., & Tolboom, J. (2021, October). Computational thinking in the mathematics classroom: Fostering algorithmic thinking and generalization skills using dynamic mathematics software. In *The 16th Workshop in Primary and Secondary Computing Education* (pp. 1-9). https://doi.org/10.1145/3481312.3481319

Wagh, A., Cook-Whitt, K., & Wilensky, U. (2017). Bridging inquiry-based science and constructionism: Exploring the alignment between students tinkering with code of computational models and goals of inquiry. *Journal of Research in Science Teaching*, 54(5), 615-641. https://doi.org/10.1002/tea.21379

Wee, L. K. (2014). Open Educational Resources from Performance Task using Video Analysis and Modeling-Tracker and K12 science education framework. arXiv preprint arXiv:1408.5992. https://doi.org/10.48550/arXiv.1408.5992

Wee, L. K., Chew, C., Goh, G. H., Tan, S., & Lee, T. L. (2012). Using Tracker as a pedagogical tool for understanding projectile motion. *Physics Education*, 47(4), 448. DOI 10.1088/0031-9120/47/4/448

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education* (TOCE), 14(1), 1-16. https://doi.org/10.1145/2576872

Zion, M., & Mendelovici, R. (2012). Moving from structured to open inquiry: Challenges and limits. *Science Education International*, 23(4), 383-399.

Zion, M., & Shedletzky, E. (2006). Overcoming the challenge of teaching open inquiry. *The Science Education Review*, 5(1), 8-10.

References to Websites

http://iep.edu.gr/el/nea-programmata-spoudon-arxiki-selida IEP Institute for Educational Policy (retrieved October 23, 2023)

https://joint-research-centre.ec.europa.eu/computational-thinking-study_en (retrieved June 20, 2024)

http://edu-computational-thinking.eu/ (retrieved June 20, 2024)

https://www.oecd-ilibrary.org/ (retrieved June 20, 2024)

Appendix

GeoGebra Evaluation

GeoGebra is a dynamic mathematics software that combines geometry, algebra and calculus. It was developed by Markus Hohenwarter (2002) and is widely used at all levels of education and in particular at the secondary level. GeoGebra allows you to create and dynamically modify points, lines, vectors, surfaces, equations and functions. In addition, GeoGebra allows the use of iteration lists and conditions, which makes it possible to use it also in computer science classes (S. van Borkulo, et.al., 2021).

The next step will be the evaluation of the specific educational software in accordance with the criteria we have defined...

Axes A - I (Guidelines 1, 2, 3, 4)

As mentioned above, the aim of the software is to support the user in formulating a problem and at the same time to produce meaning during this process. The present software does not allow the user to structure the problem freely within the software. On the contrary, its use presupposes the existence of a given problem. Of course, this problem can be of an open type, as defined in the teaching of mathematics and physics. Open problems are those that are open and can be solved in several ways, or even those that have many different solutions. Open-ended mathematics or physics problems invest in the benefits of the inquiry processes that students follow as they wrestle with the problem and search for solutions. In this way, students can master and gain a deeper understanding of the knowledge they are building (Hiebert, et.al., 1996). When problems of this type are given to be solved within the context of software such as GeoGebra, we would also expect the development of computational thinking skills, provided that the software has the appropriate features as previously defined.

Therefore, if the student/user is given an open-ended problem to solve via the GeoGebra software, then this software will support the user in selecting clear abstractions with details on which the user can focus where and when needed (strategies 2a and 2b). For example, in the problem shown in the adjacent figure (Figure 2), a ball is shot with an initial velocity u_O from the top of the left building, height h, while a second building is located at distance D. h and D are given. If the aim is to investigate the possible values of u_O for which the ball hits the opposite building or the ground, the pupil/user, thinking abstractly, is asked to express the equation for the trajectory of the ball, a secondary equation y=f(x), in the software (Figure 3). This process is supported by the fact that the software allows the user to write functions in the format that secondary school students are used to, not necessarily in Latex (strategy 2a). In this expression of the required secondary equation, the software allows the student/user to focus on the details necessary to solve the problem, such as the coefficient of x^2. The software can guide the user to some extent through the Help option (strategy 2b).

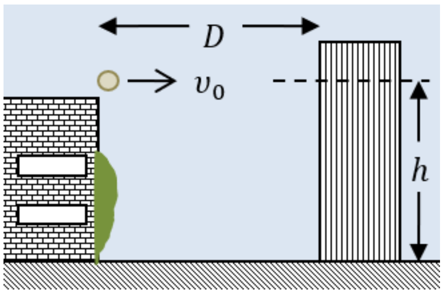
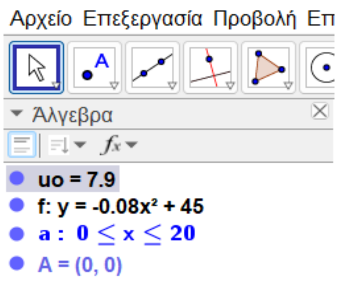


Fig. 2 Horizontal shot problem



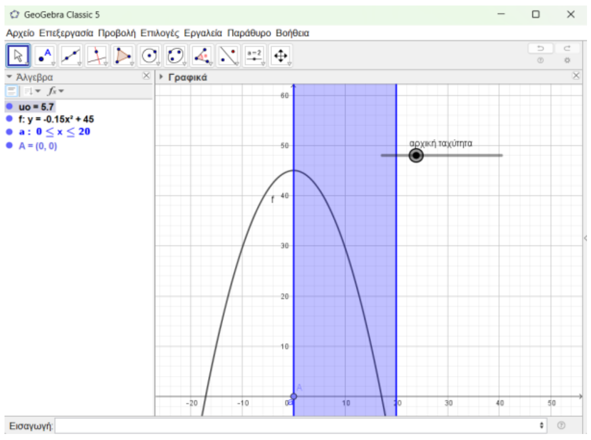
uo = 7.9
f: y = -0.08x² + 45
a : 0 ≤ x ≤ 20
A = (0, 0)

Fig. 3 Algebra image for the horizontal shooting problem (from GeoGebra software)

With regard to strategy 3a, it should be noted that the GeoGebra software is characterized by a high level of functionality, with a very good visual organization of the sections and the different representations of the software, which helps the student/user to work continuously on the problem to be solved. It is a particularly ergonomic interface, as pointed out by Echeverría et.al. (2019), where each section of the software, algebra, plane graphics, 3D graphics, spreadsheets and probabilities, has its place in the interface and can be displayed, or not, according to the user's choice, as well as the position and dimensions chosen by the user each time according to his needs. Furthermore, in the version of GeoGebra that the user can install on his computer, there is the possibility of working simultaneously on any of the different representations of the software in different windows. With regard to the above problem of horizontal shooting, the student can reproduce the path of the body for different values of the initial velocity and determine whether the body hits the ground or the opposite building (Figure 4).

On the other hand, one of the disadvantages of GeoGebra is that the descriptions of complex concepts provided by the software are particularly poor, as is the scientific guidance for the application of scientific content (strategies 3b and 3c). This is where the teacher's support is essential to ensure that the student gets the maximum benefit from the use of the software, both in terms of understanding the subject matter taught and in terms of the process of developing computational thinking skills.

As can be seen in Figure 4, the scientific strategies used by the students/users to define and visualize the trajectory of the ejected body are evident in the algebra part of the software, which is common to all the problems that a user has to solve with GeoGebra. This satisfies guideline 4 and the corresponding strategies 4a and 4b, which require the software tools to be organized in such a way as to facilitate both the conceptualization of the problem and the use of the appropriate scientific terminology to solve it. In any open problem of high school algebra or engineering, the user has the possibility to enter the elements of the problem in the algebraic field and to see the results they give in the graphic field. Any modification of the elements entered in the algebraic field, always in relation to the requirements of the problem, is automatically visible in the graphics field.

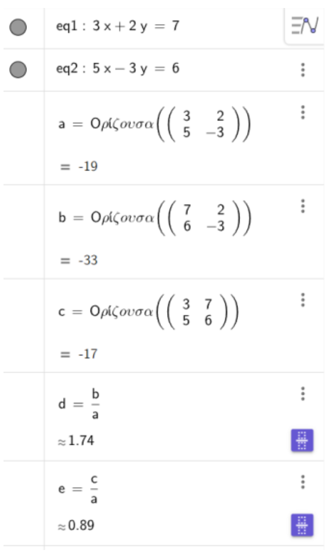

GeoGebra Classic 5
Αρχείο Επεξεργασία Προβολή Επιλογές Εργαλεία Παράθυρο Βοήθεια
Άλγεβρα
Γραφικά
αρχική ταχύτητα
f
A
Εισαγωγή:

uo = 5.7
f: y = -0.15x² + 45
a : 0 ≤ x ≤ 20
A = (0, 0)

Fig. 4 *The GeoGebra graphics window for the horizontal shot problem (image from GeoGebra software)*

Axes A - II (Guideline 5)

We could characterize the GeoGebra software as satisfactory in terms of the possibility for the student/user to explore multiple representations of the same data through its different fields. For example, the solution to a mathematical system can be solved within the software either algebraically or graphically (Figure 5), depending on the user's choice.



$\text{eq1}: 3x + 2y = 7$

$\text{eq2}: 5x - 3y = 6$

$a = \text{Ορίζουσα}\left(\begin{pmatrix} 3 & 2 \\ 5 & -3 \end{pmatrix}\right)$

$= -19$

$b = \text{Ορίζουσα}\left(\begin{pmatrix} 7 & 2 \\ 6 & -3 \end{pmatrix}\right)$

$= -33$

$c = \text{Ορίζουσα}\left(\begin{pmatrix} 3 & 7 \\ 5 & 6 \end{pmatrix}\right)$

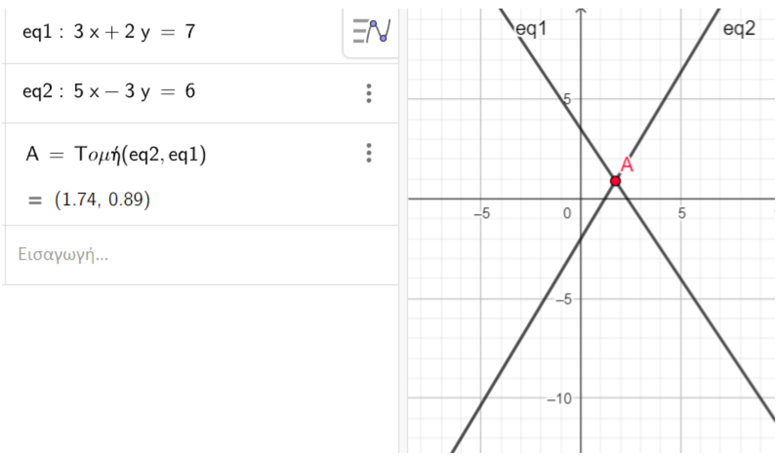
$= -17$

$d = \frac{b}{a}$

≈ 1.74

$e = \frac{c}{a}$

≈ 0.89



$\text{eq1}: 3x + 2y = 7$

$\text{eq2}: 5x - 3y = 6$

$A = \text{Τομή}(\text{eq2}, \text{eq1})$

$= (1.74, 0.89)$

Εισαγωγή...

***Fig. 5** Solving the system algebraically (left) or graphically (right) (image from GeoGebra software)*

Of course, the representational possibilities offered by the software are rather limited, which is to be expected since the present software has been designed as purely mathematical software, regardless of the fact that it can easily be used in engineering problems. For example, the previous solution of a 2x2 system could involve finding the position and meeting time of two bodies performing a smooth linear motion on the same line, given the necessary constraints. The representations provided are limited to the fields of algebra and graphics or field geometry (strategy 5a). However, these representations are easily inspected by the student/user within the software, allowing the student/user to explore the problem in at least two ways, algebraically and graphically. In addition, these representations may prove to be highly malleable and relatively easy to manipulate. Of course, whether or not these representations are malleable depends on the choices made by the student/user during the problem definition phase. For example, if the problem is to investigate the conditions under which an intersecting line is transformed into a tangent to the graph of a secondary function, like the average and momentary speed of a moving body, the student/user is asked to define two moving points of the function through which the line passes and then, by moving these points on the graph of the function, to move from the intersecting line to the tangent (Figure 6). At this point the student/user's analytical and deductive skills need to be activated.

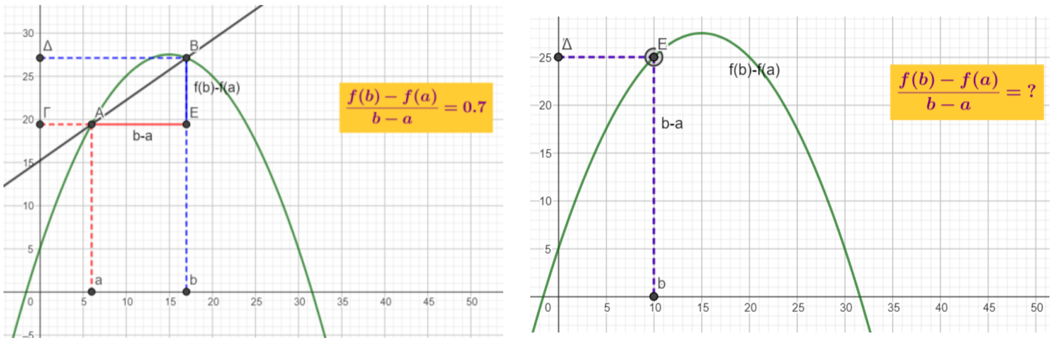
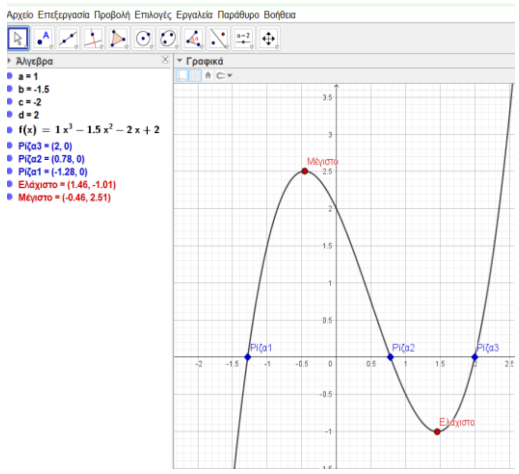


Fig. 6 *Intercept (left) and tangent (right) of a function for introducing the differential concept (image from GeoGebra software)*

Axes B - II (Guidelines 6, 7, 8, 9, 10)

The GeoGebra software allows the user to program by supporting two programming languages, GGBScript and Javascript. Although GGBScript can be considered easier than Javascript, writing a piece of code when the needs of the software require it is in any case rather difficult for the student/user. This means that guideline 6 is not met, as there is no assistance in writing the code (strategies 6a and 6b).

The reduction of the apparent complexity of the GeoGebra software in the software user interface is achieved to the maximum extent possible. Numerous procedures necessary for the different requirements that may exist during the solution of a problem are performed invisibly, without creating additional difficulties for the student/user when attempting to solve a problem with the software. For example, the roots, maxima and minima of a function can appear in the graphical representation of the function (Figure 7) without any special effort on the part of the user (strategies 7a and 8a).


Αρχείο Επεξεργασία Προβολή Επιλογές Εργαλεία Παράθυρο Βοήθεια
Άλγεβρα
Γραφικά
Μέγιστο
Ρίζα1
Ρίζα2
Ρίζα3
Ελάχιστο

a = 1
b = -1.5
c = -2
d = 2
$f(x) = 1x^3 - 1.5x^2 - 2x + 2$
Ρίζα3 = (2, 0)
Ρίζα2 = (0.78, 0)
Ρίζα1 = (-1.28, 0)
Ελάχιστο = (1.46, -1.01)
Μέγιστο = (-0.46, 2.51)

Fig. 7 *Roots and local function extremes (image from GeoGebra software)*

The GeoGebra software also allows the student/user to describe the complex tasks required to solve the problem with ordered or unordered decomposition tasks that the user has to define, thus activating the skills of analysis, decomposition, subtraction and finally synthesis (strategy 8b). The following figure shows the calculation of the average and instantaneous speed between time differences in the time-displacement graph of a moving object. The student/user has defined two moving points on the graph x=f(t) and the corresponding line and a dynamic text box where the average speed is calculated each time as the slope of the line. In the case of the tangent, the instantaneous velocity, the student/user is asked to work out how to calculate the slope using the possibilities offered by the software and basic computational thinking skills (Figure 8).

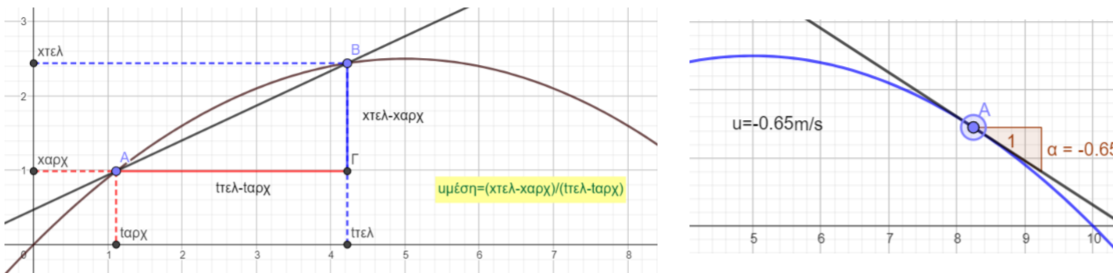


Fig. 8 Calculation of average and instantaneous velocity (image from GeoGebra software)

On the other hand, during the student/user's actions to address the problem to be solved, complex situations often arise where the software does not provide any kind of support or assistance (strategy 7b). At these stages, the teacher's assistance is needed to help the student/user solve problems of unwanted complexity. The same problem is presented in guideline 9 (strategies 9a and 9b). The GeoGebra software does not contain built-in guidance on the scientific concepts used. Its use requires either the student/user's knowledge of the scientific concepts being worked on, or the active support of the teacher by teaching these concepts beforehand or by supporting and guiding the student/user during the problem-solving phase. Which of the two strategies is chosen by the teacher depends on the problem to be solved that he/she has set for his/her students.

Guideline 10 is directly related to the complexity mentioned above and is met to a significant extent by the GeoGebra software. A significant number of tasks are performed invisibly and automatically by the software, thus reducing the cognitive demands on students/users. For example, if the problem to be solved is to investigate the existence of some basic factoring rules by discovering a pattern in the first derivative of a set of functions (Figure 9), the software helps by calculating the first derivative implicitly (strategy 10a).

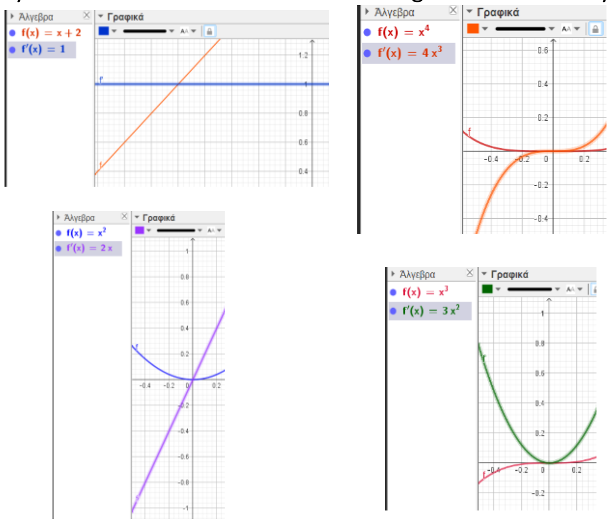


Fig. 9 First derivative of various functions (image from GeoGebra software)

At the same time, the multiple fields provided by the software to the user are easy to use and help to better organize and check the student/user's output, either by displaying them simultaneously in the interface, especially in the online version of the software, or by displaying them in separate windows in the desktop version (strategies 10b and 10c).

Axes C - III (Guidelines 11, 12, 13)

Guidelines 11 and 12 refer to programming environments and therefore it is not possible to evaluate their application or not in the GeoGebra software, in which it is possible to write code, as already mentioned, but this is on a small scale and, moreover, it cannot offer anything to the student/user in the field of learning a STEM subject and in the cultivation of computational thinking skills.

Finally, with regard to the continuous articulation and reflection on the products of the student/user's work while working on the problem to be solved, as set out in guideline 13 (strategies 13a, 13b, 13c and 13d), we should note that they are not actually provided by the GeoGebra software. Nevertheless, the continuous articulation of the problem and the reflection on the data generated by the software, following the actions of the user, are part of the inquiry process that the student/user must follow in order to successfully model the problem to be solved and lead to the answering of any questions and/or the discovery of new data related to the problem. More specifically, by introducing the problem elements into the algebraic domain, the student/user is required to cultivate data analysis, subtraction and synthesis skills in order to model and ultimately visualize the problem in the graphical domain. Thinking abstractly, the student/user should mathematise the problem, often taken from everyday life, identify the variables and define them as such through the software, observe the existence of patterns, if any, and identify and interpret the variations observed in each case in the graphical domain. Ultimately, the process of making sense as the student/user attempts to model and solve a problem with GeoGebra is a dynamic process of constant user interaction with the software.

Tracker Evaluation

Tracker is a free video analysis and modelling tool based on the Open Source Physics (OSP) Java framework. It was developed by Douglas Brown, Robert Hanson and Wolfgang Christian for use in physics education. Video modelling with Tracker is a process of combining video with computer modelling (https://physlets.org/tracker/ accessed 1 April 2023).

Tracker software is used throughout the world to teach mechanics, kinematics and beyond. Tracker can be used to study spectroscopic images and diffraction phenomena. The Tracker software also has a number of applications in secondary biology in relation to images taken using a microscope. Particularly in the study and teaching of engineering phenomena, the use of the Tracker software is a very dynamic process with direct user involvement at all stages of the study. We believe that it is a process that can significantly promote the computational thinking skills of the user through the actions that the user is required to perform in order to succeed in the study and ultimately in the solution of a problem.

Axes A - I (Guidelines 1, 2, 3, 4)

It is a software that helps the user to formulate the problem by allowing him to upload videos that he can find in the relevant library/repository of the software, or even the video that he has produced, in an attempt to define a problem to be solved in the context of the subject that he is studying (strategy 1b). In this way, the teacher or even the student can, for example, find a free fall video or even make their own free fall video. Thus, we first have a conceptualization of the problem to be solved by visualizing it (strategy 1a). This process helps the user to clarify exactly what he wants to study and even allows him, after a thorough study of the video, to define new features of the phenomenon that he might have missed and that he could study.

In a second step, the software supports the necessary abstractions (strategy 2a) that need to be made in order to be able to monitor the motion and record key quantities that could help to study the phenomenon. For example, if the task is to study the motion of a ball or a laboratory vehicle or even a real vehicle, the user should use the software to record the position of the moving body at any given time. Since the body has non-zero dimensions, he is asked to think abstractly and, with the help of the software, to define a point on the moving body whose position the software monitors during the video. In case the moving body turns during its movement, the user has to choose a suitable point of the body to follow its movement, so that the software "sees" it during the whole movement (strategy 2b). For example, a yellow tennis ball with black writing on its surface. In this case, after watching the video, the user should choose to set the reference point to, for example, the yellow surface of the ball that is visible throughout the video. In addition, depending on the movement that the user wishes to study with the help of the video, he/she should choose a coordinate system for tracking it. The software allows the user to define the coordinate system, e.g. Cartesian or polar coordinates, that he wishes to use or that is imposed by the requirements of his study with respect to the motion that he is studying. Finally, in order for the measurements to be reliable, the software assists the user in selecting the appropriate scale, which is defined by the user using the tools provided by the software. Figure 10 below shows the tracking, axes and scale selected by the user when processing a video of a body oscillation. From the above, it can be seen that the tracker software, in terms of movement study, supports the user in creating explicit abstractions in order to study in depth the movement of a body through a video recording of it.

All of the above features can certainly promote computational thinking skills such as abstraction, analysis, synthesis and modelling (strategies 1a, 1b, 2a and 2b), provided that the student is given an open-ended problem where he or she is asked to work within the software and edit the video, either alone or in groups. However, even the problems where the video is initially edited by the teacher, while not providing the above opportunities for students to develop computational thinking skills, provide similar opportunities at later stages, as will be discussed below.

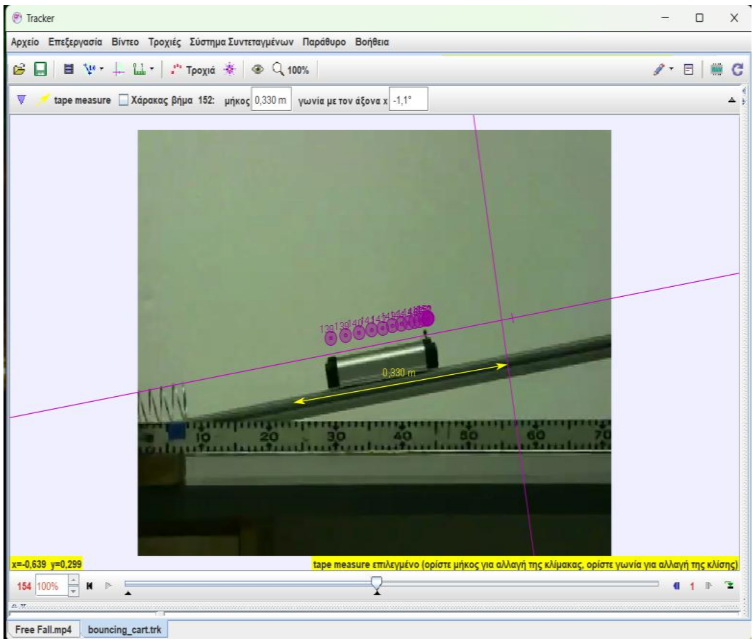
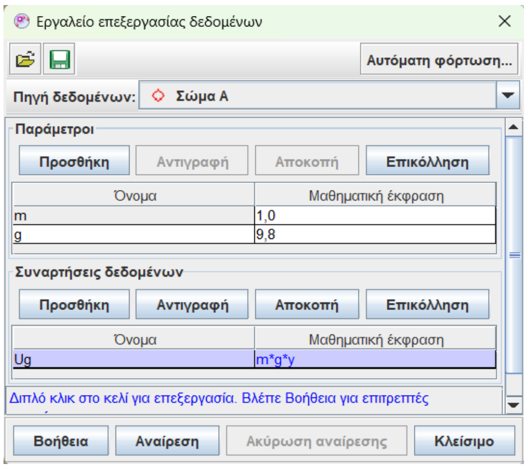

Tracker
Αρχείο Επεξεργασία Βίντεο Τροχιές Σύστημα Συντεταγμένων Παράθυρο Βοήθεια
Τροχιά
100%
tape measure
Χάρακας βήμα 152:
μήκος 0,330 m
γωνία με τον άξονα x -1,1°
0,330 m
x=-0,639 y=0,299
tape measure επιλεγμένο (ορίστε μήκος για αλλαγή της κλίμακας, ορίστε γωνία για αλλαγή της κλίσης)
154 100%
Free Fall.mp4
bouncing_cart.trk

Fig. 10 *Tracking, axes and scale calibration in a video recording of a bouncing cart (image from Tracker software)*

Regarding the ability of the software to provide descriptions of complex concepts and scientific guidance for the application of scientific content (strategies 3b and 3c), it should be stressed that the software is slow, but this is not a disadvantage. The aim of the software is not to teach new concepts directly, but to enable students to investigate the phenomena previously taught in class and to confirm the validity of the relevant theory. In addition, the tracker allows students to solve new open problems related to these phenomena, activating prior knowledge and computational thinking skills in the way described above.

Finally, for strategies 4a and 4b, which concern the organization of tools and objects around the meaning and terminology of the scientific object and the problem, it should be stressed that part of this organization is already prepared by the software itself, from the moment the user selects an axis system and calibrates it. The software measures position coordinates, velocities, momentums and kinetic energies. The rest of the organization, however, is left to the user to structure according to the requirements of the problem to be solved, using the tools provided by the software.


Εργαλείο επεξεργασίας δεδομένων

Πηγή δεδομένων: Σώμα Α

Παράμετροι

Όνομα	Μαθηματική έκφραση
m	1,0
g	9,8

Συναρτήσεις δεδομένων

Όνομα	Μαθηματική έκφραση
Ug	m*g*y

Διπλό κλικ στο κελί για επεξεργασία. Βλέπε Βοήθεια για επιτρεπτές

Fig. 11 *Parameter and data function input window (image from the Tracker software)*

For example, when a student studies the free fall of a ball by modelling a video, he is asked to define the quantities he is interested in studying (Figure 11).Thus, if the aim is to study the energy of free fall, the software can, by definition, calculate the kinetic energy of the falling body through the velocity of the body, but the user must define the functions of the dynamic and mechanical energy and also define the necessary parameters, such as the acceleration of gravity. Similarly, the user must define the forces acting on the moving object, such as gravity and air resistance, if the aim is to study the dynamics of the movement. The user's autonomy at this level requires, on the one hand, knowledge of the necessary theory and, on the other hand, analytical and deductive skills during the phase of organizing the study of an open problem using the software, for example, the variation of the mechanical energy of a body in free fall or the dynamic study of the motion of a free-falling object.

Axes A - II (Guideline 5)

The Tracker software fulfils strategies 5a, 5b and 5c to a very satisfactory degree. The representations provided by the software can be considered particularly flexible (strategy 5c), since it allows the user to trace the movement (trajectory marking) in any part of the video that interests him, to trace the center of mass in problems involving body systems and to study the movement of the center of mass, but it also offers the possibility of displaying a stroboscopic image of the movements taking place in the video, an image that is particularly useful in the study of body impacts. In addition, the representations provided can be considered flexible, multiple and also with the possibility of controlling them (strategies 5a, 5b and 5c), since the user is provided with the possibility of creating the graphs of different quantities and through them to determine the dependence or not of the quantities and the nature of this dependence. For example, when studying free fall, students have been taught and intuitively understand that it is an accelerated motion. The software allows students to establish and ultimately prove that it is a straight line, smoothly accelerating motion. One qualitative way is by tracing the trajectory of the body, where, by noting the successive positions of the moving body, frame by frame, the student finds that in equal periods of time the body travels longer and longer distances. Another way is to look at the velocity vectors in each frame of the video, where the student will see that these vectors become larger and larger as the body descends. Another way is to examine the vertical velocity versus time matrix, where an increase in the vertical velocity measure is evident. Finally, it is possible to use the software to draw a graph of vertical speed versus time, from which the software, using statistical analysis, gives the user the function that relates the two physical quantities. In this way, the user can confirm the time-velocity relationship known from theory.

Of course, it should be stressed that all the above representations are not provided a priori to the user, but it is up to the user to choose the appropriate representation depending on the problem to be solved. For example, if the user is asked to compute the displacement of a car seen moving in a video, or the vertical displacement of the Falcon 9 rocket until its crash during its landing in 2016 (video available in the Tracker library, Figure 12), the user will have to find the way to calculate this displacement using the multiple representations that can be created with the software, following problem solving procedures, computational thinking procedures, which will lead to the best and most accurate option, which is to calculate the area on the vertical velocity-time graph. Exactly the same applies to the moving car, since the user can easily see, using multiple representations, that this motion cannot be approximated by any of the motions taught in high school physics.

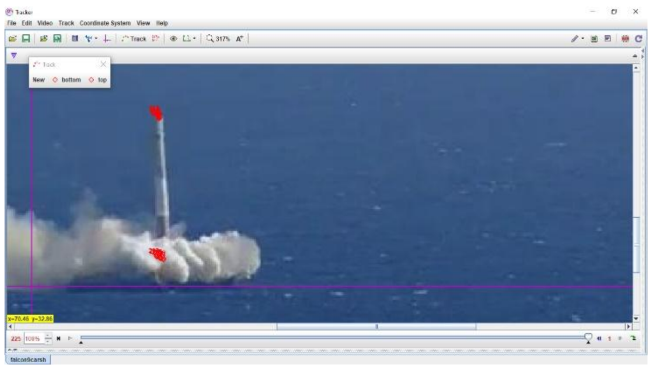
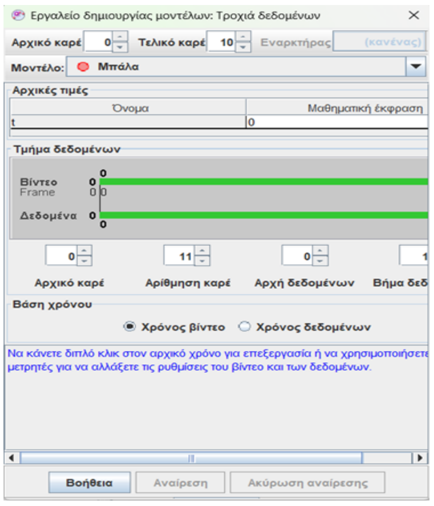


Fig. 12 *Falcon9 Landing (image from Tracker software)*

Axes B - II (Guidelines 6, 7, 8, 9, 10)

Obviously, the Tracker software is not a programming environment and therefore cannot be evaluated at the level of Guideline 6, as it is exclusively concerned with this type of environment.

Guideline 7 on minimizing software complexity has been met to a large extent. The user is not involved in the complex procedures for calculating the velocities and accelerations of the moving bodies under investigation, which are performed in the background (strategy 7a, 10a) using the method of least differences between the positions of the moving bodies and the time instants. The software also allows the user to fill any gaps in the measurement file. This is done by means of a simple command that leads to linear correlation procedures of the existing values in the background (strategy 7a, 10a) in order to find and fill in the values relating to intermediate positions of the trajectory. This procedure is particularly useful in cases where the user wishes to process and visualize data from an external source, for example an Excel file (Figure 13).


Εργαλείο δημιουργίας μοντέλων: Τροχιά δεδομένων

Αρχικό καρέ: 0 | Τελικό καρέ: 10 | Εναρκτήρας: (κανένας)

Μοντέλο: Μπάλα

Αρχικές τιμές

Όνομα	Μαθηματική έκφραση
t	0

Τμήμα δεδομένων

Βίντεο 0 | Frame 0 | Δεδομένα 0

Αρχικό καρέ: 0 | Αρίθμηση καρέ: 11 | Αρχή δεδομένων: 0 | Βήμα δεδ: 1

Βάση χρόνου

[✓ Χρόνος βίντεο] Χρόνος δεδομένων

Να κάνετε διπλό κλικ στον αρχικό χρόνο για επεξεργασία ή να χρησιμοποιήσετε μετρητές για να αλλάξετε τις ρυθμίσεις του βίντεο και των δεδομένων.

Fig. 13 *Modelling tool (Tracker software image)*

In this case, the student can study and visualize a move for which there is no video, such as a Formula 1 Grand Prix move for which there is a speed file. There are many such videos on YouTube. After recording the speed values of the car for each moment in an Excel file, the user can give this file to the tracker, model the movement and study it. The student can do the same with an Excel file containing measurements from a real experiment. In this way they can model the data they have and visualize it by creating a simulation of the real-world motion, thus gaining the ability to make comparisons. The above process requires a variety of computational thinking skills, the most important being the user's recognition that accurate modelling/simulation in a 100fps (frames per second) video requires a time step in the measurements of 0.01s. In such cases, it is not uncommon to have missing values, which, as explained, the user can generate following a command in the software that runs in the background.

It follows from the above that the software provides the necessary structure for complex tasks such as the previous one, and at the same time is highly functional (strategies 8a, 8b, 8c, 10b and 10c), which is reinforced by the fact that in the initial window the user can, if he/she chooses, simultaneously have an image of the video, the table of values and the graphical representations, or focus on what he/she is interested in at any given time (Figure 14). In addition, the handling of the video, the table of values and the graphs are very accessible. Of course, the choices the user has to make in these operations in order to solve any open or closed engineering problem they face requires them to develop computational thinking skills.

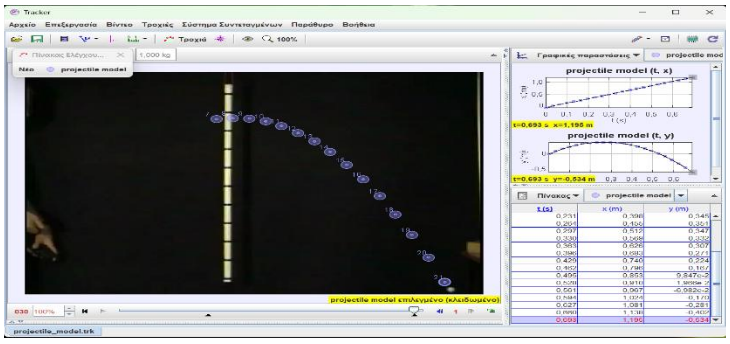


Fig. 14 *Video - Data Table - Graphs (image from Tracker software)*

Finally, regarding guideline 9 on the integration of expert guidance on scientific practices, the software itself does not provide any guidance on the scientific practices used. The user, with relevant prior knowledge of physics, is invited to work with the software on open or closed engineering problems at an inquiry level, with or without the guidance of the teacher. The more the teacher intervenes, the less the students' computational thinking skills are activated and vice versa. It is therefore necessary to find the necessary balance between teacher intervention and student self-direction in order to activate and cultivate students' computational thinking skills as much as possible, without getting them 'lost' in processes of identifying software functions.

Axes C - III (Guidelines 11, 12, 13)

Guidelines 11 and 12 relate to programming environments and therefore we cannot comment on them in relation to the Tracker software. Of course, it should be mentioned that Tracker, like most software, provides help to the user for all its functions. This help is particularly useful for students when they are asked to tackle an open engineering problem without any particular guidance from the teacher. The help provided is always clear and to the point. In addition, the Tracker allows the user to take basic notes on the problem bodies being studied and save them for later use when and if they need to refer to them. It is also possible to save the entire experiment, so that it can be continued and/or developed at a later stage, and of course to access the results at any time.

Finally, with regard to guideline 13 on continuous articulation and reflection on the problem that users are asked to solve, it should be stressed that this process is not included in the software but is part of the inquiry process that students are asked to follow in order to model and study the problem. The key components of the computational thinking that students should activate in order to be successful in studying the phenomenon and answering any question posed to them about it are the productive planning of the modelling, the productive monitoring of it and, finally, the production of meaning through this process. Of course, the software helps in this direction with its multiple and malleable representations and with the possibility for students to focus each time on a different feature of the phenomenon, either through the video itself, the data table or the graphs. But it is the students who decide where to focus, whether to define new quantities to study or new graphs, whether they have done proper tracing, proper calibration, proper use of axes. And therein lies the potential for students' computational thinking skills to emerge and be cultivated through the software, while at the same time exploring engineering issues.