Journal of Information Technology Education: Innovations in Practice

An Official Publication of the Informing Science Institute
InformingScience.org

JITEiip.org

Volume 23, 2024

A Constructionist Approach to Learning Computational Thinking in Mathematics Lessons

Wahid Yunianto*	Johannes Kepler University, Linz, Austria	yunianto.wah@gmail.com
Houssam El-Kasti	Qatar University, Doha, Qatar	helkasti@qu.edu.qa
Zsolt Lavicza	Johannes Kepler University, Linz, Austria	zsolt.lavicza@jku.at
Rully Charitas Indra Prahmana	Universitas Ahmad Dahlan, Yogyakarta, Indonesia	rully.indra@mpmat.uad.ac.id

* Corresponding author

Abstract

Aim/Purpose	This study presents some activities that integrate computational thinking (CT) into mathematics lessons utilizing GeoGebra to promote constructionist learning.
Background	CT activities in the Indonesian curriculum are dominated by worked examples with less plugged-mode activities that might hinder students from acquiring CT skills. Therefore, we developed mathematics and CT (math+CT) lessons to promote students' constructionist key behaviors while learning.
Methodology	The researchers utilized an educational design research (EDR) to guide the lesson's development. The lesson featured 11 applets and 22 short questions developed in GeoGebra. To improve the lesson, it was sent to eight mathematics teachers and an expert in educational technology for feedback, and the lesson was improved accordingly. The improved lessons were then piloted with 17 students, during which the collaborating mathematics teachers taught the lessons. Data were collected through the students' work on GeoGebra, screen recording when they approached the activities, and interviews. We used content analysis to analyze the qualitative data and presented descriptive statistics to quantitative data.

Accepting Editor Martin D Beer │ Received: May 7, 2024 │ Revised: July 17, August 7, 2024 │ Accepted: August 15, 2024.
Cite as: Yunianto, W., El-Kasti, H., Lavicza, Z., & Prahmana, R. C. I. (2024). A constructionist approach to learning computational thinking in mathematics lessons. *Journal of Information Technology Education: Innovations in Practice, 23,* Article 11. https://doi.org/10.28945/5365

Contribution	This study provided an example and insight into how CT can be enhanced in mathematics lessons in a constructionist manner.
Findings	Students were active in learning mathematics and CT, especially when they were engaged in programming and debugging tasks.
Recommendations for Practitioners	Educators are recommended to use familiar mathematics software such as GeoGebra to support students' CT skills while learning mathematics. Additionally, our applets are better run on big-screen devices to optimize students' CT programming and debugging skills. Moreover, it is recommended that students work collaboratively to benefit from peer feedback and discussion.
Recommendations for Researchers	Collaboration with teachers will help researchers better understand the situation in the classroom and how the students will respond to the activities. Additionally, it is important to provide more time for students to get familiar with GeoGebra and start with fewer errors to debug.
Future Research	Further research can explore more mathematics topics when integrating CT utilizing GeoGebra or other mathematics software or implement the lessons with a larger classroom size to provide a more generalizable result and deeper understanding.
Keywords	computational thinking, mathematics, constructionist, GeoGebra

INTRODUCTION

Computational thinking (CT) has received significant interest from scholars across the globe (Bocconi et al., 2016, 2022). The research on integrating computational thinking in mathematics education has increased significantly in the last five years (Subramaniam et al., 2022). Several countries have incorporated CT into their educational curricula, and CT is taught in various ways across countries. Some include it as a specific subject, some include it in other disciplines, and yet others include it as a component of a subject (Bocconi et al., 2016). Some countries, like France, Finland, Sweden, and Norway, have made it explicit in their compulsory education to integrate CT into the mathematics subject (Bocconi et al., 2022). The initiatives reported from their research show the significance of CT in education and for 21st-century skills.

Although often taught as a term in computer science, CT has applications beyond this field. It is a problem-solving approach that can be applied to a variety of disciplines (Wing, 2006). The impact of CT is crucial in various subject fields, including science, mathematics, languages, and others. Researchers have integrated CT into other school subjects such as science (Sneider et al., 2014; Weintrop et al., 2016), mathematics (Chan et al., 2020; Hickmott et al., 2018; Ho & Ang, 2015; Ho et al., 2018, 2019, 2021; Khoo et al., 2022; Subramaniam et al., 2022; Ye et al., 2023) and language (Rottenhofer et al., 2021; Sabitzer et al., 2018) and those integrations promote CT and the embedded subjects. It proved that CT is implementable in other subjects, including mathematics. Additionally, learning CT and mathematics as an integration could benefit both, as they could co-develop and support each other (Ye et al., 2023).

Based on literature reviews, the researchers found studies on the integration of CT in mathematics education focus on students, design activities, and preservice teachers (Khoo et al., 2022), tools used, and countries (Subramaniam et al., 2022), students' levels, mathematics domains, tools, as well as CT framework (Ye et al., 2023). These different literature reviews supplement each other, such as the tools used to teach CT in mathematics lessons (Subramaniam et al., 2022; Ye et al., 2023). A summary of CT research in mathematics education will help other scholars take a stance and conduct further research. From those studies, we learned about mathematical software and approaches that can potentially integrate CT into mathematics lessons.

Inevitably, with the introduction to new innovations in education, teachers might face challenges to integrate them. Furthermore, due to the curriculum reform that encourages teachers to infuse CT in possible subject areas in European countries (Bocconi et al., 2016, 2022), teachers might encounter challenges in designing and accessing CT activities or lessons. Some teachers have reported that they felt uncertain about their understanding of CT despite having it in their school curricula, and they find it difficult to teach CT to their students (Nordby et al., 2022). Teachers need to be trained due to their limited understanding of CT and teaching approaches (Kravik et al., 2022). This indicates that the presence of CT in the curriculum in their schools does not necessarily ensure that teachers can easily instruct it, and they need support. To overcome the challenges faced by teachers in Singapore, teachers are prepared with CT activities, exemplars to teach mathematics and CT, and the design Math+CT principles (Ho et al., 2019) – such CT activities for integrating CT in mathematics act as a reference for teachers in their lesson implementation.

Meanwhile, in Indonesia, the curriculum document promotes the integration of CT in computer science (CS) or informatics disciplines. It shows that Indonesia realized the importance of students developing their CT skills. However, CT integration in other disciplines is rarely found. If non-CS teachers, such as mathematics teachers, could contribute to enhancing students' CT skills, they first require access to learning activities that would aid in incorporating CT into mathematics. Additionally, in Indonesia, students primarily learn CT from worked examples and in unplugged mode (not using machines). Learning with worked examples could reduce cognitive load and impose less challenge (Chen et al., 2023). However, it might inhibit the students' CT development as it does not reflect Papert's (1980) strategy to develop students' CT skills by letting children explore the codes to program mathematical objects and gain knowledge from their interaction with the computer. Therefore, this paper provides an alternative approach and examples of learning mathematics using computational thinking with the constructionist approach. We propose to use mathematics software called GeoGebra instead of other general programming software. The research question we aim to answer is: How do the mathematics CT tasks utilizing GeoGebra align with constructionist constructs?

Literature Review

There is a growing interest in integrating computational thinking (CT) in education and mathematics education (Subramaniam et al., 2022). In this section, the researchers will elaborate on the conceptualization of CT, its components, how CT is integrated with mathematics, and the tools for students to develop CT skills.

Computational Thinking (CT)

Papert (1980) first introduced computational thinking (CT), describing it as a way of accessing knowledge through computer interaction. Papert and his fellow researchers created a program called 'turtle geometry' to let students program mathematical objects. Wing (2006) later revived CT and defined it as the ability to solve problems by processing information agents, humans, or a mix of the two, arguing that it is a core skill alongside reading, writing, and arithmetic. However, there isn't a consensus definition for CT, and there are different ways to teach it. In this study, we combined Papert's (1980) key idea and Wing's (2006) concept of problem-solving skills.

CT can be delivered in two different ways: unplugged or plugged. The term "unplugged," introduced by Bell et al. (2009), refers to conducting CT activities without using computers or information processing agents. Meanwhile, the plugged CT utilizes computers and comparable tools to help students improve their CT skills (Hermans & Aivaloglou, 2017). Students are able to participate in both types of CT activities and gain benefits from their involvement.

CT FRAMEWORK IN MATHEMATICS EDUCATION RESEARCH

The researchers investigated the existing frameworks that have been utilized in this domain during the development of our math+CT lessons. Other scholars have employed various frameworks to investigate the incorporation of computational thinking (CT) in mathematics instructions (Ho et al., 2021; Hoyles, 2015). Several frameworks have developed from their predecessors (Shute et al., 2017; Zhong et al., 2016). One potential explanation is that the framework adapts or conforms to the requirements of the CT implementation. For instance, Shute et al. (2017) further improved the framework of CT proposed by Weintrop et al. (2016) by extending grade-level specifications and the use of programming tools. The enhanced CT framework will assist us in devising our CT tasks in mathematics lessons, as this framework is not limited by grade level or specific tools. Consequently, in this study, the Math+CT tasks were grounded in the framework established by Shute et al. (2017).

In this section, the researchers will elaborate on the six CT facets described by Shute et al. (2017) and the connection to Math+CT lessons. Table 1 presents the six facets of CT, their indicators proposed by Shute et al. (2017), and the connections corresponding to our mathematics activities. These facets suggest how a task designer can infuse CT activities into mathematics lessons. However, most of the GeoGebra applets developed in this study exhibit mainly algorithms and debugging. Other facets were less apparent, but they were there.

Table 1. CT facets and indicators and corresponding mathematics activity

No.	Facet	Indicators	Connection to Math+CT Tasks
1	Decomposition	Students dissect a complex problem into smaller components. The smaller components are functional elements that, when combined, constitute the entire problem; they are not arbitrary portions.	To construct an inscribed regular polygon in a circle, students will learn how to construct a circle and then the polygon or vice-versa.
2	Abstraction	Students extract the crucial information of a problem. Abstraction can be divided into three subcategories: (a) *Collecting and analyzing data collection and analysis:* students gather information that is most useful and important from different sources and understand the connections of different data. (b) *Pattern recognition:* students should recognize patterns or principles that underlie the data. (c) *Modeling:* students construct models or simulations to depict the operation of a system or its prospective functionality.	Data collection and analysis: students will get some measurements such as the area of the circle, radius of the circle, area of the regular polygon, angle measure, and number of vertices of the polygon. Students will observe the pattern of how the number of vertices influences the area of the polygon and how the area is approaching the area of a circle. When the circle has a radius of 1, the area of the polygon inscribed in it will be approaching 3.14. Students would model the simulation of a dynamic regular polygon inscribed in the circle with a slider.
3	Algorithms	Students create a set of coherent and systematic instructions for effectively presenting a solution to a problem. The instructions can be executed by either a human or a computer. There are four subcategories:	Students will learn how to input commands in GeoGebra and later need to construct geometrical figures by themselves. Students can run some commands simultaneously. Students can

No.	Facet	Indicators	Connection to Math+CT Tasks
		(a) *Algorithm design:* students devise a sequence of systematic procedures to resolve a problem. (b) *Parallelism:* students simultaneously execute a particular number of tasks. (c) *Efficiency:* students aim to minimize the number of steps required to solve a problem by eliminating redundant and superfluous actions. (d) *Automation:* students can automate the implementation of the method if necessary to solve comparable problems.	input fewer commands to construct the objects. Students can use sliders to automate the polygon with different vertices.
4	Debugging	Students engage in the process of detecting, identifying, and fixing errors when a solution fails to function as intended.	Students will learn how to debug the work of a fictional character, Andi, who programmed incorrectly. Students must find the errors and fix them.
5	Iteration	Students iterate design procedures to enhance solutions until the optimal outcome is attained.	In the final task, students will construct a specific object, and the process should be repeated until the object comes as close as possible to the determined success criteria.
6	Generalization	Students use their CT skills in various contexts and areas in order to solve problems effectively and efficiently.	Students are expected to use their experiences from the activities to extend it to different inscribed polygons.

GEOGEBRA IN MATHEMATICS AND CT

GeoGebra is an interactive geometric software that is widely used for teaching geometry and algebra. Geogebra has three key features – modeling, visualization, and programming – allowing users to tackle the importance of different mathematical fields (Ziatdinov & Valles, 2022). Researchers can use GeoGebra to teach mathematics in demonstration, interaction, or creation (Lavicza et al., 2010).

Teachers can use GeoGebra to create an applet by inputting commands and uploading them to the website. A study by Kushwaha et al. (2013) shows that, with limited programming skills, teachers can make attractive mathematics applets using simple GeoGebra script commands. Velikova and Petkova (2019) trained preservice mathematics teachers to create GeoGebra applets, resulting in satisfying programming skills and the basic ability to use commands or codes. The researchers believe that it could be possible for junior or high school students to do the same things in more straightforward activities like the applets in our study.

GeoGebra is accessible without charge to mathematics instructors both offline and online. The implementation of GeoGebra in mathematics education facilitates students' mathematical learning (Hamzah & Hidayat, 2022; Juandi et al., 2021; Velikova & Petkova, 2019; Yohannes & Chen, 2021). Hence, our study employed this software to aid students' development of mathematical and computational thinking skills with the gadgets they could afford.

There is limited research on utilizing GeoGebra for incorporating computational thinking (CT) into mathematics (Subramaniam et al., 2022; van Borkulo et al., 2021). van Borkulo et al. (2021) implemented GeoGebra into their design lessons for academic research and reported that the lessons improved students' development of algorithmic thinking and generalization. Their study utilized GeoGebra, accompanied by a paper worksheet, and involved around 15 Grade 12 students in the Netherlands. Students inputted the variables and/or equations on the input box, resulting in the computations and representations. In this manner, it was reported that students' algorithmic thinking (AT) was enhanced by allowing them to work on steps of commands and iterations that they proceeded with GeoGebra. The "If" command was used by these students to generate a perpendicular line bisecting a given line segment. In addition, students enhanced their generalization ability by replacing letters or variables in GeoGebra. The given example used specific numbers, and then students inputted letters to replace the numbers so that it resulted in a more general form. Despite the benefit of utilizing GeoGebra to enhance CT concepts, students also experienced challenges, such as some difficulties in writing down the instructions for GeoGebra (i.e., the equation could be troublesome to input). Another recent study utilizing GeoGebra in integrated CT in mathematics lessons also improved students' CT skills (Chytas et al., 2024). Due to the limited study on the utilization of GeoGebra to incorporate computational thinking (CT) into mathematics education, the researchers contributed to this field by creating mathematics and computational thinking (CT) lessons utilizing GeoGebra. These lessons had the purpose of promoting constructionist learning to students.

Comparing Traditional and Math+CT Approaches

We expected that in our Math+CT lessons, students would learn computational thinking skills while learning mathematics through active participation in a way that Papert (1980) intended to have with his idea of constructionist 'to access knowledge.' Meanwhile, traditional approaches to teaching mathematics usually entail a teacher-centered approach, where concepts are presented, and problems are solved in a systematic way, ensuring clarity but frequently neglecting active involvement and the cultivation of critical thinking abilities (Boaler, 1998; Schoenfeld, 1988). The mathematics + computational thinking (Math+CT) innovation integrates computational thinking with a constructionist approach, using tools such as GeoGebra to encourage interactive and student-centered learning. This approach not only improves involvement but also nurtures analytical thinking and practical problem-solving abilities (Weintrop et al., 2016). For instance, students who integrate the Math+CT lessons have the ability to visually represent and investigate geometric figures in an interactive manner, resulting in a more profound comprehension of principles and enhanced problem-solving skills. In general, the Math+CT integration has been shown to have clear benefits in terms of student engagement, critical thinking skills, and the ability to apply knowledge to real-world situations. This is supported by the improved learning outcomes and positive feedback from students in the study conducted by Grover and Pea (2013) and Bers et al. (2014).

Methodology

The researchers employed the educational design research (EDR) methodology by McKenney and Reeves (2018). We worked with eight mathematics teachers and one specialist to develop Math+CT lessons. As a result, the lessons underwent development through discussion and modification from the initial Drafts I to II to III. Draft II was implemented for 17 junior high school students. This paper focuses on the implementation of Draft II.

The EDR is helpful for our study since its iterative nature helps us design the mathematics lessons that integrate CT. Figure 1 shows that EDR has three stages in the cycle, namely, (1) exploration and analysis, (2) design and construction, and (3) evaluation and reflection (McKenney & Reeves, 2018, p. 83).

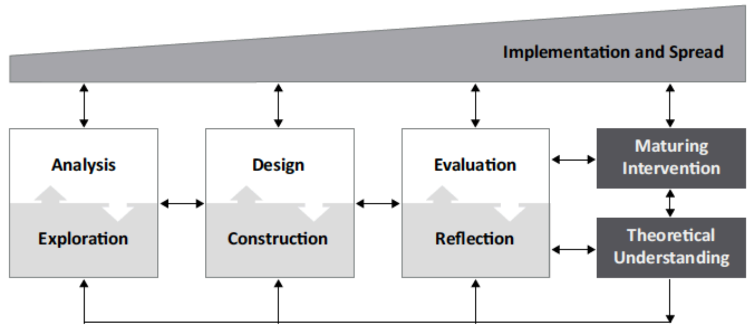

Figure 1. Generic model for conducting design research in education

In the development of Math+CT lessons, EDR provides a structured framework to design, implement, and evaluate instructional approaches that integrate computational thinking with mathematics topics. This approach incorporates iterative cycles of planning, implementation, and evaluation, starting with identifying educational goals and devising interventions associated with these goals. We utilized tools like GeoGebra to facilitate interactive exploration of mathematical topics, ensuring computational thinking abilities are promoted within authentic situations. The evaluation would focus on evidence of student learning through activities, reflections, and quantitative measures, permitting ongoing improvement based on research findings and feedback from students.

At Freudenthal Institute Utrecht University, mathematics education students utilized EDR principles to create and assess educational interventions that improve students' mathematics understanding by providing them with contextualized and engaging learning experiences (FIsme, 2015). By employing EDR in the development of Math+CT lessons, researchers cannot only revolutionize their didactical approach but also make valuable contributions to educational research by resulting in local instructional theories (ILT) on how CT could be integrated into mathematics lessons and what has worked successfully in this integration (Chytas et al., 2024; van Borkulo et al., 2021, 2023). This integration of computational thinking into mathematics education helps to provide students with the necessary skills (CT as a problem-solving skill) and knowledge to tackle future challenges in Science, Technology, Engineering, and Mathematics (STEM) domains and beyond.

The first researcher examined the national curriculum of Indonesia during Stage 1. In-depth interviews were undertaken with practitioners and experts involved in the development of the CS curriculum and CT teacher training. The researchers found some resistance to accepting new tools or software introduced by the trainers, and the participating teachers were more comfortable with spreadsheets or Ms. Excel to teach CT. Moreover, the first researcher has been unable to locate any resources pertaining to the use of CT for mathematics education within the documents of Indonesian government institutions. Consequently, the development of plugged mathematics+CT lessons is necessary.

During Stage 2, the researchers created the instructional materials. The first researcher developed the activities (Draft I) for the lessons and then distributed them to the mathematics teachers and practitioners who were collaborating with us. The mathematics topic is derived from the Indonesian curriculum intended for students in lower secondary schools. The topic of the lesson is a circle and related problems connected to area measurement. The mathematics in this study aims to teach students about the area of a unit circle using Archimedes' exhaustion method (King, 2013), which involves considering the area of an inscribed regular polygon. After the draft was ready, it was shared with the collaborating mathematics teachers (T1, T2, T3, …, T8) and the practitioner (P), who offered suggestions for development based on online discussions and participation in activities. Consequently, Draft I progressed into Draft II (Figure 2).

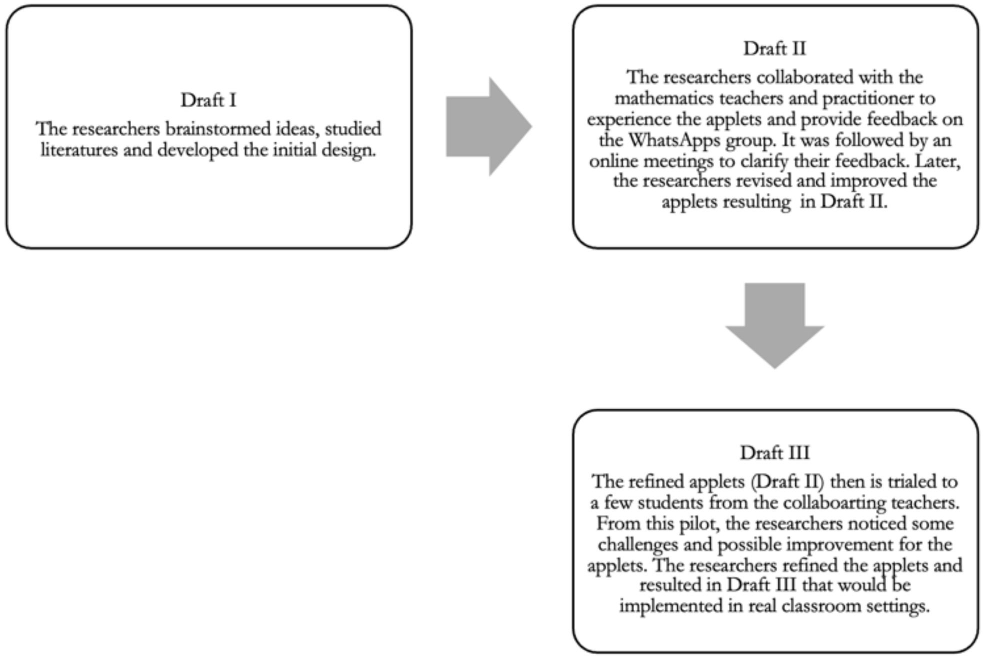

Figure 2. The improvement processes from Draft I to Draft III

In stage 3, the researchers trialed Draft II, which involved a small group of 17 students. During this stage, the researchers assessed and analyzed the outcomes and experiences in the classrooms. The students were affiliated with mathematics teachers who were actively involved in the development stage. The researchers delegated the responsibility of selecting the participating students to the teachers. Seventeen junior high school students participated in this pilot project, with each group consisting of three to four students led by five mathematics teachers. They engaged in the completion of the lessons within the GeoGebra applets. Their problem-solving methods enabled us to assess the efficacy of our design. From this pilot, the researchers gained insights to enhance the lessons for Draft III.

DEVELOPING TOOL

This section presents the tools used to integrate CT into mathematics lessons. First, existing studies (e.g., Subramaniam et al., 2022; Wang et al., 2022; Ye et al., 2023) were reviewed to list the tools used in integrating CT into mathematics. Based on our review, we have found that certain tools, such as GeoGebra, Spreadsheet, Microsoft Excel, R, and MATLAB, have been reported to support the learning of computational thinking in mathematics education (Subramaniam et al., 2022; van Borkulo et al., 2021, 2023; Wang et al., 2022; Ye et al., 2023). In this study, we have chosen to utilize GeoGebra due to its ability to facilitate commands inputting to construct mathematical objects, editing commands, and online classrooms. Additionally, GeoGebra is a familiar resource for mathematics teachers, as it has millions of users located in nearly every country (GeoGebra, 2024).

Figure 3 displays an illustration of the math+CT activities in GeoGebra. After inputting the correct commands, the 13-sided polygon was constructed. However, by only following the hint, students would not construct the intended object(s) on the yellow box, and this will be considered an incorrect response.

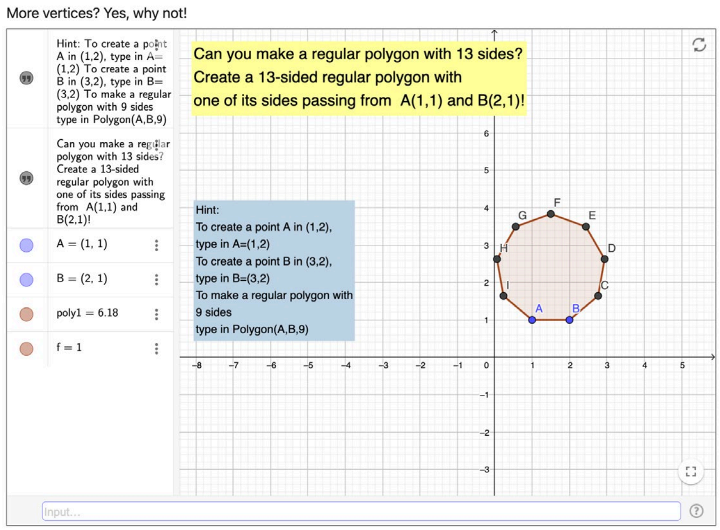

Figure 3. GeoGebra feature:
Input box and algebra view for commands to construct objects

Constructionist Approach

The development of the activities in this research is grounded in Papert's (1980) constructionist theory. Papert pioneered the use of computational thinking in the classroom, particularly in the teaching of mathematics, and conceptualized the constructionist theory. Papert's concept of computational thinking focuses on the use of digital artifacts in conjunction with a digital learning environment for the purpose of acquiring or accessing knowledge. Consequently, this form of engagement helps students build their knowledge. Papert developed a tool based on LISP programming, which computer scientists used to tackle difficult issues and help young learners explore mathematics (Kynigos, 2015). Young learners interacted with this tool to construct mathematical objects.

Papert (1980) proposed not only the definition of the constructionist theory of learning but also put forth a key principle for learning in a constructionist approach that was based on three main keys: engagement in the activity, ownership of one's ideas and learning style, and exposure. The mathematics+CT lessons in our research were developed with a focus on these principles, utilizing the GeoGebra platform. Students were encouraged to program geometric objects and fix command errors (bugs), use their ideas by organizing the instructions based on their personal preferences, and discuss their solutions with their peers.

First, the engagement describes students actively constructing or reconstructing the digital artifacts. Instead of introducing students to artificial mathematical pictures and trying to understand abstract mathematical activity, Papert (1972) preferred students to do mathematics. What Papert designed in 'Turtle Geometry' allowed students to work with the programming language and explore the geometrical shapes independently. Similarly, GeoGebra has the potential to do similar things to allow students to program mathematical objects. To some extent, the environment could shape or trigger the engagement. Figure 4 depicts how a student engaged in our applet to explore the area of a unit circle by programming a model that can visualize the changing of the vertices of the inscribed polygon in a unit circle using a slider, and this student had moved the slider into 100. This means that this student successfully programmed this simulation and played with it, depicting engagement. If students had just created the simulation without exploring the slider to observe the changes in the vertices and the polygon' area, the engagement would not been fully incorporated by students. However, if the slider remained on n=3, it might be that students have moved the slider back to n=3 after exploring it or students did not move the slider so that it remained at n=3, showing an equilateral triangle inscribed in a unit circle.

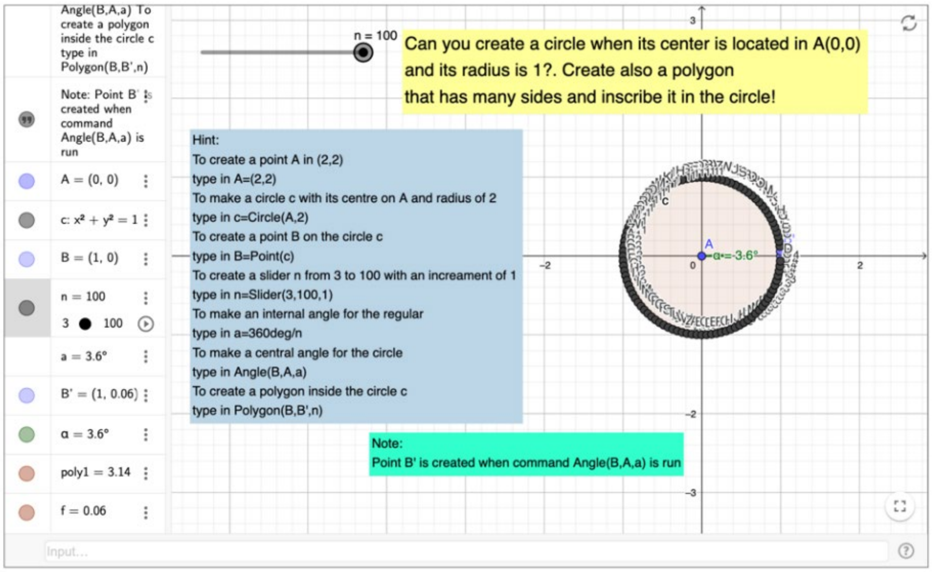

Figure 4. A student's work on creating a model for investigating an inscribed polygon in a unit circle

Additionally, specific digital environments could support students in accomplishing meaningful projects, keep them motivated, and encourage them to develop and share their projects (Lodi et al., 2019). The projects could be video games, animated stories, or simulations of simple real-world phenomena. It is essential to design an environment that allows students' engagement to construct, modify, and tinker with digital artifacts (Kynigos, 2015). In our study, the researchers provided students with GeoGebra applets, which they could use to program, construct, and simulate mathematical objects such as polygons, circles, angles, and points to understand mathematical concepts (area of circles).

The second key to learning is to facilitate students' ideas and learning styles. This means that students use their ideas to construct the artifacts and are free to use their own ways or approaches. Teachers should avoid a ready-made environment that students are prescribed to follow. An alternative is to provide 'half-baked' artifacts, as proposed by Kynigos (2007), where students can change and improve unfinished artifacts as if they were engineers. The half-baked task is intended to let students gain ownership of the techniques and ideas of the artifacts' construction (Kynigos, 2015). In the tasks developed in this study, the researchers used somewhat different half-baked artifacts by providing students with unfinished command(s) to construct geometrical objects or commands containing errors for students to fix to construct the artifacts successfully. The students can utilize their strategies to construct objects, even though some commands should follow orders. They can deviate from the example commands as long as the object is constructed. This, the researchers believe, will instill in students a sense of ownership over the strategies.

Lastly, the key to learning is exposure, which could mean that students can present their ideas, work with their peers, and demonstrate how they created the objects (Kynigos, 2015). However, GeoGebra lesson feature has yet to let students collaborate or work in groups within one task. Therefore, this key to learning is facilitated outside the learning environment. Teachers could provide rooms for students to share their creations or artifacts and how they developed them with their peers. Students were allowed to discuss their strategies with their friends. Based on the explanation of constructionist constructs, the researchers would contrast those constructs to the actual learning when students approached our Math+CT tasks.

The constructionist theory has evolved and considered social dimensions, resulting in social constructionist theory. This could enrich our perspective on the way knowledge is shaped in this manner.

Mackrell and Pratt (2017) discussed the importance of social interaction in constructionism and shed light on how social dimensions could contribute to the ‘spaces for reasons’ in their proposal to view constructionism. Brady et al. (2016) examined a cooperative project on programable badges (digital name tags) and provided evidence in favor of the possibility of incorporating social dimensions into learning in a constructionist manner. Students worked together to program the badges, and they benefited from this environment as they took their roles to contribute. Ultimately, they combined their experiences to create new knowledge or understanding.

In summary, the researchers adhered to Papert’s constructionist perspective and core learning concepts, allowing student social interaction during the artifact construction process. The researchers agree that we should consider the social component of knowledge construction; in our environment, students are free to converse or engage in discussion with one another. A more thorough explanation of this will be provided in the methodology section.

CONTEXT

In 2019, Indonesia introduced computer science (informatics) subjects in schools’ curricula from primary level to high school level that focus on computational thinking (CT) skills, in accordance with the recent update on the significance of CT in education (Suprayitno, 2019). Currently, some schools have implemented computer science (CS) subjects. However, the books or learning resources for learning CT are still dominated by unplugged CT activities. Despite the isolation of learning CT in CS lessons, Indonesian researchers have attempted to integrate CT into mathematics lessons following global trends (Irawan et al., 2024). Irawan et al. (2024) found an increase in Indonesian researchers investigating CT integration into mathematics, and it revealed limited studies incorporating the plugged CT activities when integrating it into mathematics lessons. Therefore, it is necessary to investigate further plugged Math+CT lessons in Indonesia, especially with mathematics software.

Before elaborating on the Indonesian national mathematics curriculum, the researchers provide a synopsis of the instructional methods and learning approaches employed in these five schools. Teachers participating in our study are not fully acquainted with GeoGebra. According to our conversation with the teachers, they have incorporated GeoGebra into their mathematics lessons primarily for demonstrating concepts and only occasionally for creating artifacts. The GeoGebra tasks were prepared in partnership with the teachers from these schools, as previously stated. Their astonishment stemmed from the fact that GeoGebra could be configured to utilize programming techniques to generate mathematical objects.

Making a connection to the existing mathematics curriculum is essential to sustain the implementation of the lesson. The researchers developed the lessons (activities) based on the Indonesian new mathematics curriculum content, namely determining the area of a circle and its related problems. Usually, students were directly given the area formula of a circle. Students practiced cutting the circle into segments and arranged it into a parallelogram or a triangle to find its area. Pi (3.14) is also given directly to students, or if the teachers innovated with circular objects, they used the proportion of its diameter and circumference. To learn the area of circles, this study provided a different approach, simulated using a GeoGebra applet (Figure 5). The researchers tried to provide students with an investigation to measure area of a circle through the approximation method while interacting with GeoGebra and relate to the formula they have learned.

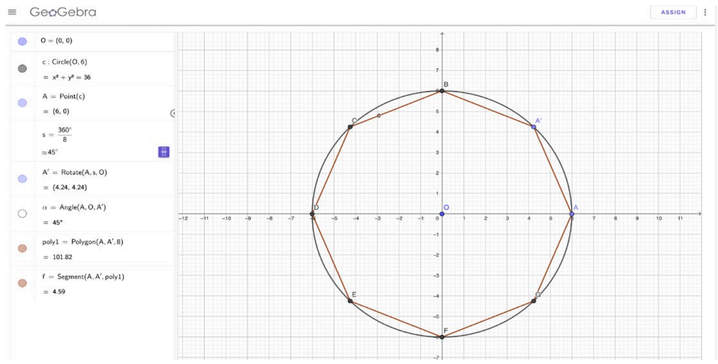

Figure 5. Approaching the area of a circle with an area of a polygon in GeoGebra

The area of a regular polygon (later, the researchers call it a polygon) with many vertices can be used to approach the area of a circle. This approach is usually called Archimedes' exhaustion method (King, 2013), which approaches the circumference of the circle by the perimeter of the polygon inscribed in it. Rather similar to King's (2013) activity that examines the circumference and formula of a circle, our study specifically focused on the area of a circle through the utilization of an inscribed polygon. Students will utilize programming skills to create a regular polygon that can be manipulated inside the unit circle. The manipulatable regular polygon is capable of altering its number of vertices, increasing or decreasing them, by the use of the slider command on GeoGebra. Consequently, as additional vertices are added, the approximation of the unit circle's area becomes increasingly accurate. Thus, the researchers developed two lessons. Lesson 1 consisted of mastering how to create regular polygons through GeoGebra's script (Figure 6). Lesson 2 continued creating a circle and then inscribing a regular polygon on a circle.

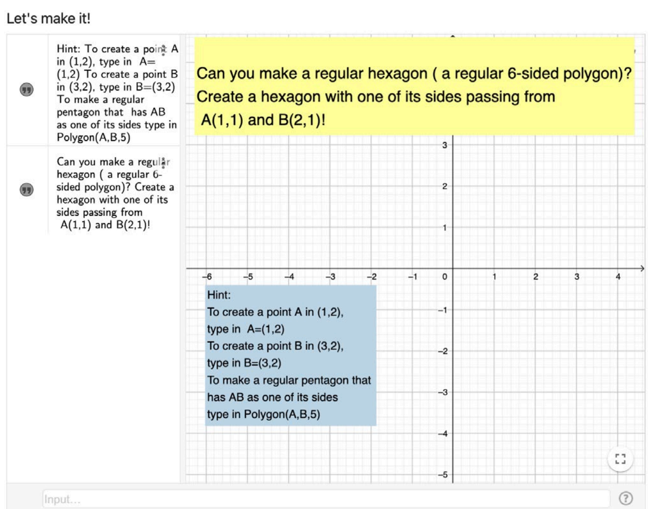

Figure 6. A task to construct a hexagon in lesson 1

The researchers have developed 33 GeoGebra tasks consisting of 11 GeoGebra applets and 22 open and follow-up questions. The tasks are divided into two main sections: section 1 consists of 4 GeoGebra applets (https://www.geogebra.org/m/st4g9kph and https://www.geogebra.org/m/ffq7xusc), and section 2 consists of 7 GeoGebra applets. Each applet has its own prominent CT components, which mainly deal with algorithms and debugging (Table 2). The yellow box indicated the main task or the goal that students needed to accomplish. In Figure 6, students had to create a hexagon with one of its sides passing from A(1,1) and B(2,1). Meanwhile, the grey box indicates a hint or an example of how to use the GeoGebra commands to create similar objects on the yellow box.

Table 2. The applet and its prominent CT components

Applet	Description	CT component
1	Constructing a hexagon	Algorithms
2	Constructing a 13-sided polygon	Algorithms
3	Constructing a polygon with a slider	Algorithms
4	Debugging a polygon construction	Debugging
5	Constructing a unit circle	Algorithms
6	Debugging a unit circle construction	Debugging
7	Constructing an inscribed polygon in a circle	Algorithms
8	Debugging a polygon in a polygon	Debugging
9	Estimating pi	Algorithms
10	Programming without Algebra View	Algorithms and debugging
11	Advanced programming without Algebra View	Algorithms and debugging

Data Collection and Analysis

The researchers requested support from the collaborating teachers to carry out the lessons. Therefore, the role of the researchers was to monitor the execution of the teaching implementations by providing guidelines, such as hypothetical learning trajectories and instructions on collecting data from video screen recording and storing it in a folder on Google Drive. The researchers could later access the recording files in the folder and the students' work in GeoGebra classrooms. To carry out the lessons, students could use the school's computer laboratory laptops, tablets, or mobile phones (Figure 7). This study gathered data by recording the screen as students completed GeoGebra tasks and examining the students' work saved on GeoGebra. Some students had challenges while attempting to capture the screen on small devices like tablets and mobile phones because of limited memory capacity. This condition probably impacted the devices' speed in accessing GeoGebra. Out of seventeen students, only twelve screen videos were collected. Not all the twelve videos recorded audio of the students and teachers.

Figure 7. Pilot implementation in different schools

Students' works on GeoGebra were analyzed quantitatively by counting the number of students who could successfully accomplish the task. The video that captured students' processes was used to support the quantitative data by providing evidence of how students proceeded with the tasks. Additionally, the types of accomplishments that students have submitted were categorized. The researchers employed the content analysis methodology outlined by Krippendorff (2004) to analyze the data specifically from applet 11, with three steps: making, categorizing, and concluding the codes. First, the researchers looked at the students' works one by one to see if any patterns or differences existed amongst students' works. Then, researchers noticed that some students worked correctly, partially correctly (incomplete), incorrectly, or even left it empty, using slider command or not.

In some tasks, especially the last applet (applet 11), the researchers set the RUN button and CLEAR or DELETE button to record how many of these students have pressed (Figure 8). This would inform us of the differences amongst students in accomplishing the task, whether they tried it once or more. For instance, when students could construct the inscribed octagon in a circle without using a slider command and only executing the run button one time without pressing the clear button, we coded this as SNsR1C0. This code denotes "Successful with No slider, pressing 1 Run and 0 Clear." Another code is FR10C0, which means Failed with pressing the run button ten times and not pressing the clear button. The E stands for empty, meaning that the student did not put any commands. The result of this will be tabulated and presented in the Finding section.

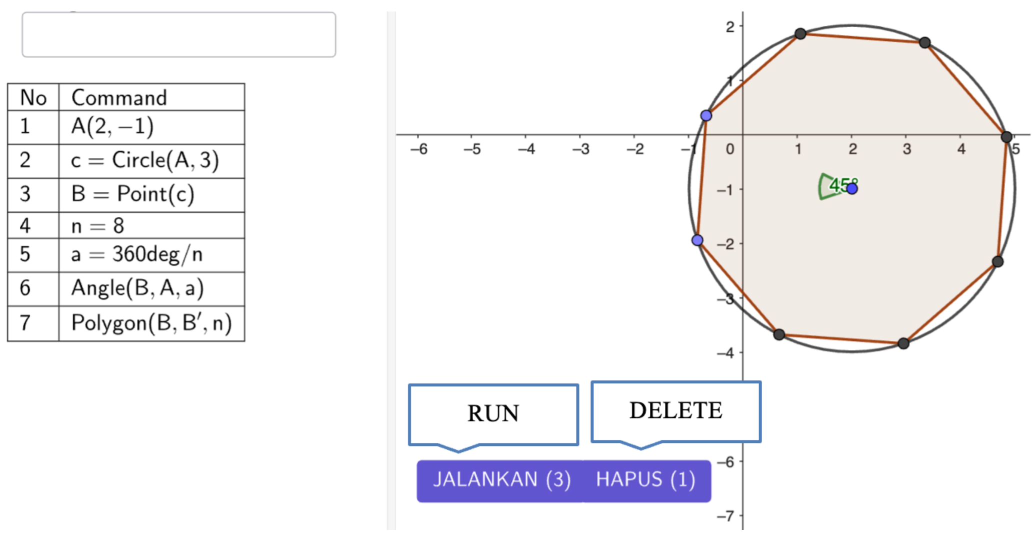

Figure 8. The DELETE and RUN buttons feature

Additional data was from an impromptu interview after students finished the lessons. One of the teachers interviewed one student on how they experienced the lessons. As it was not planned, the teacher asked some questions related to the student's experiences while working on the activities. The interview was voice recorded, and later, it was transcribed and analyzed. This data would add more information on how students perceive our tasks.

FINDINGS

This section presents sequential GeoGebra tasks for students to do. As previously mentioned, most of the students in our study did not learn mathematics using GeoGebra through programming, and they often studied GeoGebra for simulation or demonstration.

Applet 1 requested students to construct a hexagon with given points. To understand the tasks, students had to read the instructions on the yellow box as the primary goal to create requested objects (Figure 9). There are hints on how to do it (grey box). However, it uses different numbers (coordinates, sizes, radius, variables). Following the hints directly would just construct the objects on the hints but not the intended objects. Thus, students need to pay attention to this hint carefully; otherwise, students will create the object from the hint. This supported the pattern recognition in relation to the CT sub-skill. Only six students could respond correctly to this first task. The remaining students followed the hint fully or partially. This could be because it was the first time these students had worked on this kind of task.

For some tasks on the left side, the researchers made the 'Algebra View' visible purposely so that when students input mistakes on the commands, they could edit them here (Figure 9). In more challenging tasks, the researchers purposely hid this Algebra View to let students think before inputting the commands. The correct commands to create the requested objects are A=(1,1), B=(2,1), and Polygon(A,B,6). Then, a hexagon with one of the vertices lying on points A and B would be constructed (Figure 9).

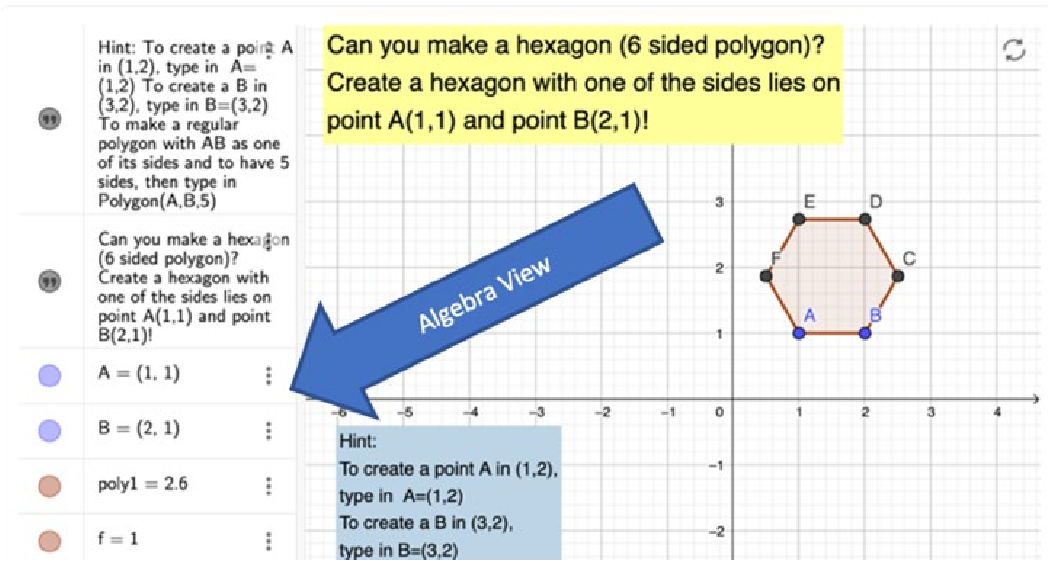

Figure 9. A successful hexagon creation on applet 1

Later, students had to confirm whether those two commands, A=(1,3) and A(1,3), could be used to create a point or not. It could help them decide which commands are more suitable for them or faster. Ten students responded that they could use those two commands to create a point. Other commands can only be inputted uniquely; no other commands can do the same. For instance, to create a regular polygon on GeoGebra, students need to input "Polygon(<Point>,<Point>,<Number of Vertices>)" and cannot be swapped the order to be "Polygon(<Number of Vertices>,"<Point>,<Point>)." Students could learn pattern recognition from this. A follow-up question was asked: What happens if the point is reordered? In this case, the command Polygon(A,B,5) becomes Polygon(B,A,5). The researchers expected that students could observe the difference between these two.

Next, students would do a similar task to create a polygon with a different number of vertices (13) (Figure 10). This was intended to help them practice and familiarize themselves with the commands as they are relatively new to them. For this task, students were asked to create a 13-sided regular polygon instead of 6 sides. Most students (14 out of 17) could successfully do this task.

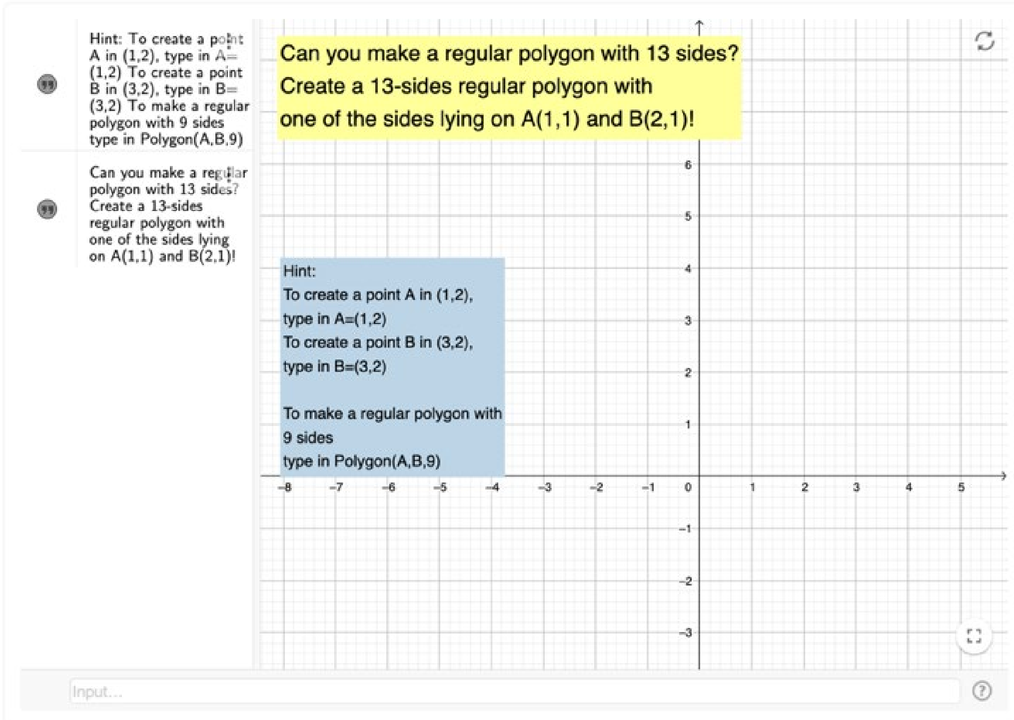

Figure 10. Applet 2: creating a 13-sided polygon

Next, students worked on an animated polygon by using the slider. The task asked students whether it was possible to change the slider to 100 (Figure 11). They could modify the slider by changing the maximum number from 10 (hint) to 100 or more. They could edit it on Algebra View or re-input it again. This required pattern recognition skills to be able to modify the variable. Most students (14 out of 17) responded that they could make a 100-sided polygon.

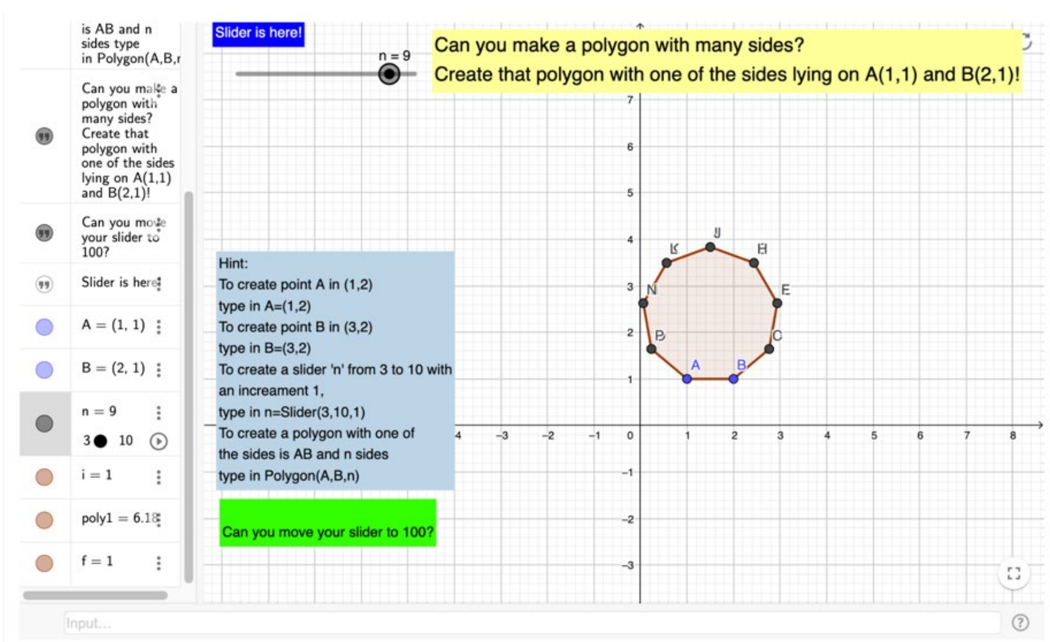

Figure 11. Applet 3: creation with a slider

Consecutively, students had to write a command for a slider with a start of 100 and an end of 1000 with an increment of 10. Most students (14 out of 17) could correctly answer this question by writing down n=Slider(10,100,10). The problem with using big numbers would slow down GeoGebra and possibly lead to a computer 'hang' or the screen freeze. This finding is significant as it suggests using fewer numbers in future studies.

When students were familiar with the commands and programs, the researchers introduced them to debugging. The task let them debug or fix a program created by a fictional name, "Andi." Andi wrote commands that contained errors that students should find and fix. Table 3 describes the errors and the correct commands.

Table 3. Errors for students to debug

Line	Correct Command	Andi's command	Description
1	A=(1,1)	A=(1,1)	Correct
2	B=(2,2)	B=(2:2)	The use of colon (:) instead of comma (,)
3	n=Slider(3,100,1)	n=Slide(3,100,1)	The missing letter: r
4	Polygon(A,B,n)	Poligon(A,B,n)	Case-sensitivity. The use of i instead of y

Andi had four commands to the GeoGebra, and students had to find the incorrect errors by Andi (Figure 12).

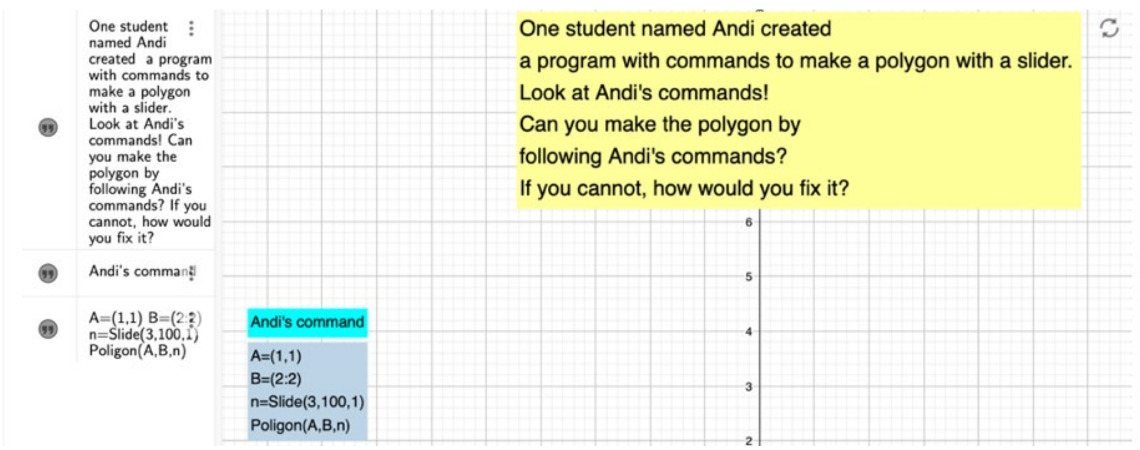

Figure 12. Applet 4: debugging when creating a polygon

In this task, most students (12 out of 17) could fix the errors. It might be challenging as this task proposed many errors (3 errors) when introducing debugging. Therefore, a few students needed help fixing the errors. The worst case was students only created a point A. The researchers would consider this in the subsequent implementation to initialize with one or two errors in the debugging task.

Another experience the researchers found was when a student (Q) discussed with her fellow student (Z) sitting at the computer desk next to her. On the debugging task (Figure 12), this student wrote down the list of commands' revisions and ensured she had listed all the revisions to her peer. Her peer replied to her that she missed one command, which is the point B command "B=(2:2)". In Figure 13, we can see that her peer peeped (her head is partially visible on this student screen recording) while helping.

Q: They (the revisions) were only the slider and the polygon. (While writing the revisions)
Z: The B!
Q: What is it with the B?
Z: The B used colon ":"!

Q: (Scrolling up to the GeoGebra task to check) This is right?
Z: The B is incorrect!
Q: Oh (realizing the B was not correct because it used colon ":")

This student (Q) could fix Andi's commands, and later, they had to write down what revision she made on the following question (Figure 14). From this conversation, students helped each other confirm the errors that needed to be fixed. One student missed one command that needs to be fixed on the question of what revision you made to Andi's commands. This showed a kind of exposure to the idea that a peer could improve the ideas of others by noticing friends' missing details or errors.

Figure 13. Two students discussed the debugging task (Applet 4)

Task 14: Bagaimana memperbaiki perintah si Andi?

Tuliskan (ketik) perintahnya di sini!

B=(2:2) menjadi B=(2,2), n=Slide(3,100,1) menjadi n=Slider(3,100,1), Poligon(A,B,n) menjadi Polygon(A,B,n)

Translation: B=(2:2) to B=(2,2) n=Slide(3,100,1) to n=Slider(3,100,1) Poligon(A,B,n) to Polygon(A,B,n)

Figure 14. The student could write down what she fixed on the error commands

After getting familiarized with polygon construction and debugging, students moved to work on constructing circles in lesson 2. This lesson could be carried out on the same day as the polygon lesson or another day, depending on the teacher's decision. In Figure 15, the researchers initialized to create a unit circle and then with a polygon inscribed on the circle. The task asked students to create a circle with a center on A(0,0) with a radius of 1 unit.

A simple command to create a circle is to use a point as a center and a number for the radius, "Circle(<point>,<radius>)." Again, this command cannot be reordered as "Circle(<radius>,<point>)." Students could get the area of the unit circle by using the area command, Area(Object). In the Algebra View, the area, Area(c), is shown as 3.14 (2-digit rounding up was set on the GeoGebra). Here, students encountered data, a measurement for an area. Students could observe the area of a circle with a radius of 1 produced an area of 3.14. At this point, students might not understand fully how the area formula for the circle "Area(c) could result in 3.14. Later, the task would go deeper into how this estimation would be derived from an area of a regular polygon inscribed to that circle.

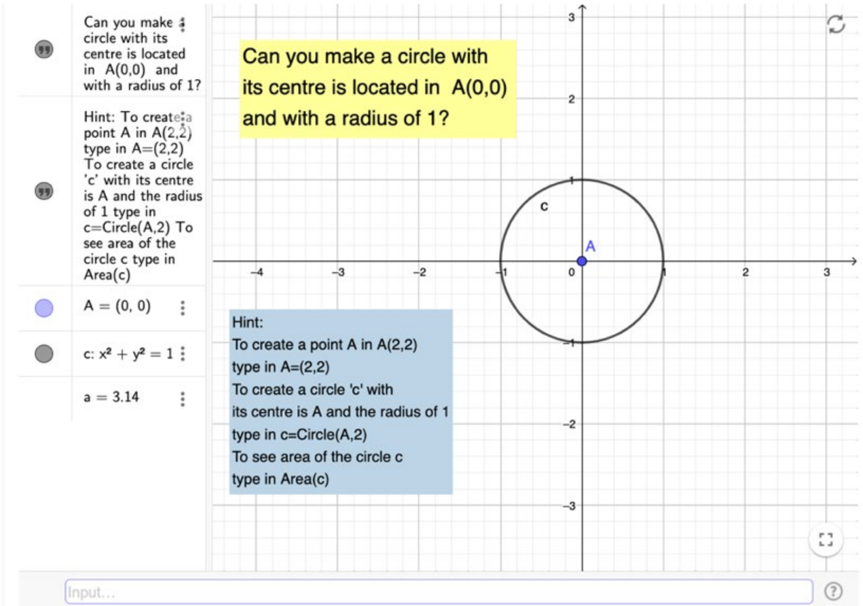

Figure 15. Applet 5: Creating a unit circle

Students had to answer to the area of a circle with a radius of 1 by observing the Algebra View. Eleven students performed well on this task, while some made mistakes on the radius as two instead of 1. This task was more straightforward than the polygon, using only three commands. However, students still needed to pay attention to the yellow box or the main objective of the task. After creating the circle, the next activity was to debug Andi's commands (Figure 16).

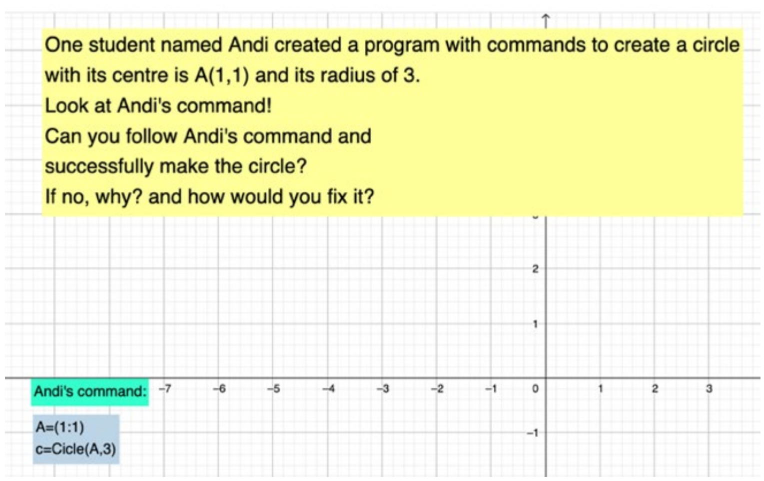

Figure 16. Applet 6: debugging a unit circle creation

In Table 4, the commands were shorter than on the polygon and contained two errors (all are errors).

Table 4. Errors for students to debug a circle creation

Line	Correct Command	Andi's command	Description
1	A=(1,1)	A=(1:1)	The use of colon (:) instead of comma (,)
2	c=Circle(A,3)	c=Cicle(A,3)	The missing letter r

Thirteen students could solve this task except those who left this task empty. All students who inputted commands on this task got the correct answer. Compared to previous debugging tasks, this task was less demanding as it consisted of fewer errors and was more straightforward. This presumably allowed students to successfully debug this task. After accomplishing this task, students constructed a regular polygon inside the circle (Figure 17). The instruction was more extended than previous programming tasks and rather complicated.

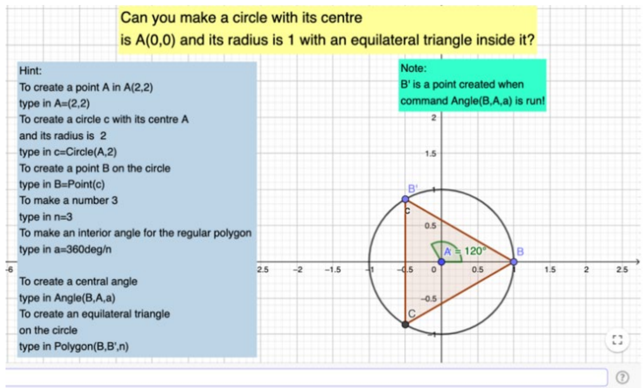

Figure 17. Applet 7: inscribing a polygon on a unit circle

This task consisted of commands to create a point on the circle, the polygon interior angle, and the circle's central angle. What made it more complicated was the new dependent point B', which was the result when running the command Angle(B,A,a). Point B' was required to construct the n-gon as BB' is a vertex of the polygon to be created in Polygon(B,B',n). This resulted in eight students who could correctly accomplish this task.

Later, students would proceed with another debugging task (Figure 18). In this task, there were many errors (4) to be found and fixed. The result showed that only seven students could respond to this task correctly. The remaining students could not finish this task or did it incorrectly. It could be that many errors were still troublesome for students in our study.

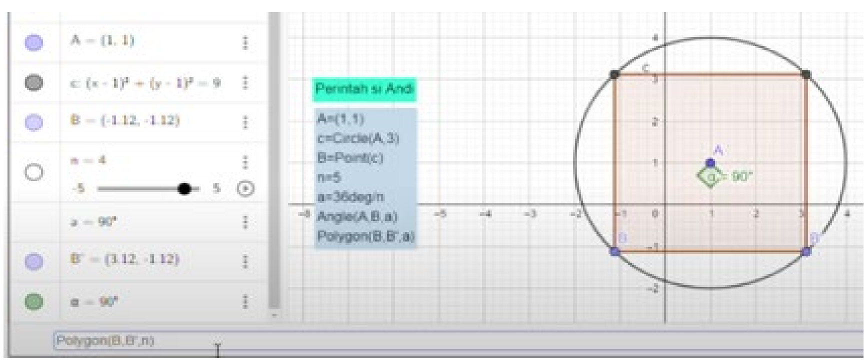

Figure 18. Applet 8: debugging task to create an inscribed square

Reflecting on this, for future implementation, the researchers reduced the errors to two (Table 5). Some students struggled a lot when debugging too many things at once, and they might not understand the pop-up notification informing them of the error. This pop-up notification was in English, and our research subjects were not English-speaking students. Although seven students could perform well, the researchers would reduce the number of errors for our next cycle.

Table 5. Errors command to create an inscribed polygon

Line	Correct command	Andi's command	Description
1	n=4	n=5	number of vertices is 4 for a square
2	a=360deg/n	a=36deg/n	The angle to be divided should be 360
3	Angle(B,A,a)	Angle(A,B,a)	Incorrect order
4	Polygon(B,B',n)	Polygon(B,B',a)	Incorrect variable for the number of vertices

In the next task (Figure 19), students would learn how to estimate pi (3.14) using an inscribed polygon on a unit circle utilizing a slider. Again, students encountered data collection, pattern, and simulation (modeling), which are part of the abstraction CT skill. The more vertices, the closer the area of the polygon is to the area of the unit circle, which is close to 3.14. In this task, when the slider was moved to 56, it created a 56-sided polygon with an area of 3.14 square units (2-decimals rounding up set-up on GeoGebra).

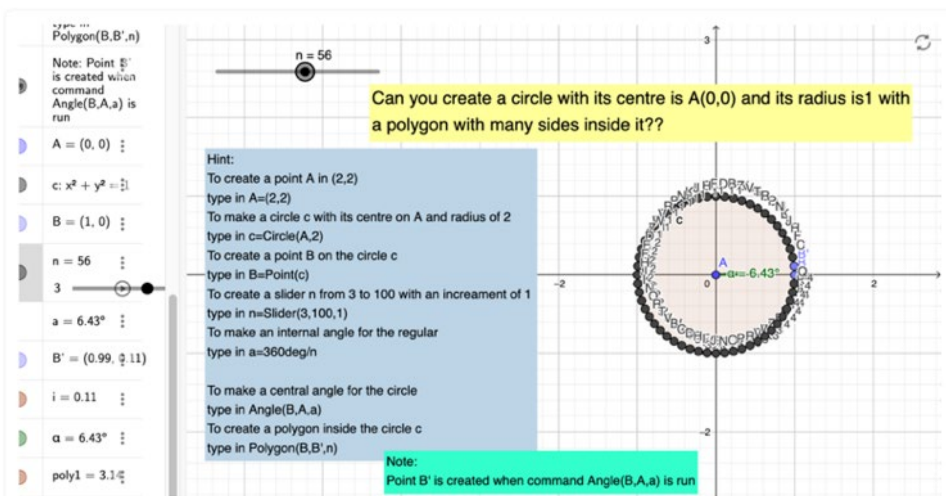

Figure 19. Applet 9: estimating pi

Eleven students could make the inscribed polygon with a slider in a circle. After completing this task, students had to answer whether the area of the polygon changes when the slider is moved. This resulted in 12 out of 17 students responding that the area changes. Additionally, students had to answer, 'What is the area of the polygon if it has 300 sides?'. Students should modify the slider to move it up to 300 or more. They could create n=Slider(3,300,1). This resulted in 15 out of 17 students who could get the correct answer (3.14) and only one answered by 0.14.

The next task was more challenging as the Algebra View was no longer visible (Figure 20). The Algebra View was hidden purposely to let students experience just entering the commands line by line and become more careful to input commands.

Students must be able to use the programs or commands they have learned to create the requested object. Seven lines or rows were the minimum commands to be inputted if students learned from the previous task (Figure 20 left). Most students (11 out of 17) could accomplish this task. Although this task allowed students to have fewer commands, it seems that students followed the pattern from the previous task. None of the students in our studies has used five commands. The researchers assume it was not because they did not know, but we did not trigger them to have as few commands as possible. In future implementations, the researchers will stress that students will have fewer commands to input.

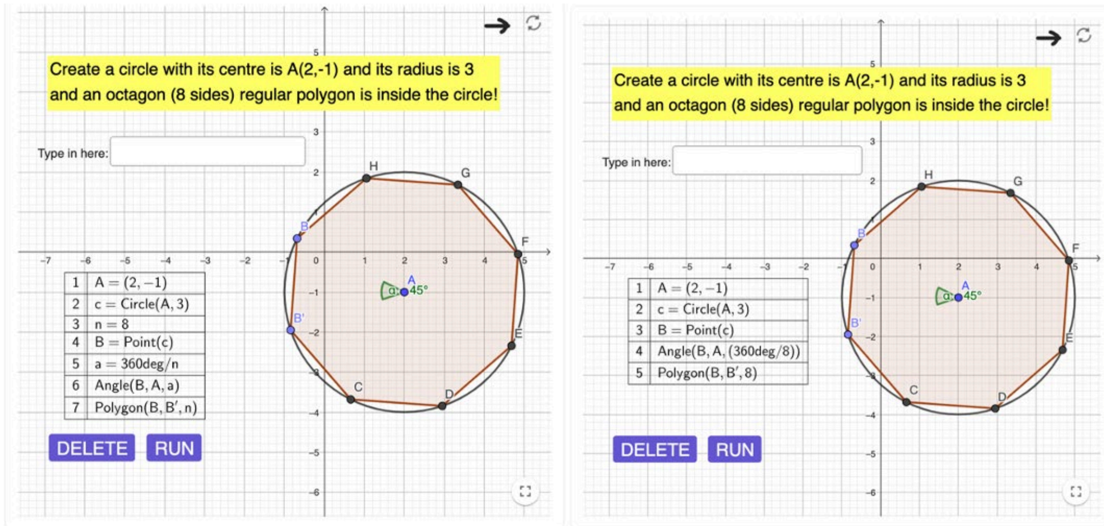

Figure 20. Applet 10: programming without algebra view

The last GeoGebra task (applet 11) requested students to perform something similar, but students could edit and replace the commands they had inputted when they made mistakes (Figure 21). The grid was hidden, and the number of runs and clears would be counted.

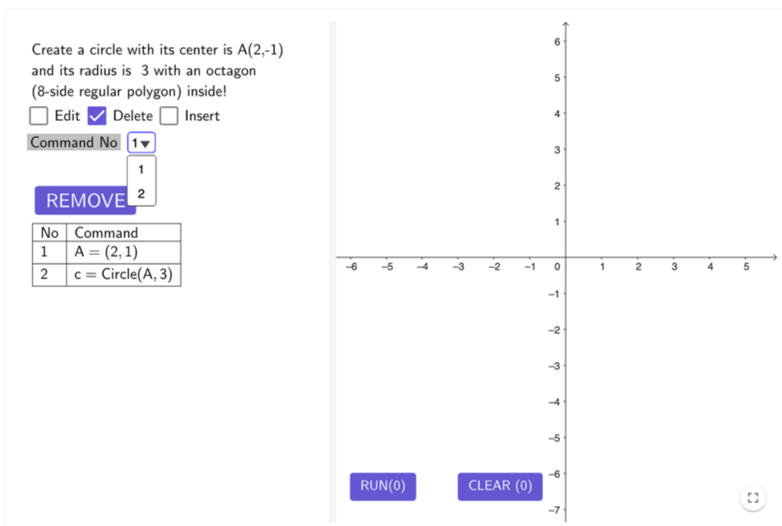

Figure 21. Applet 11: advance coding

The researchers listed students' works and coded them in Table 6. Most students could input the correct commands, 14 students. Only two students left the task empty. Only one student ran the program 16 times (SSR16C0). It seems that this student did not give up easily and kept on trying more and more. It depicted the engagement, keep trying to approach the task. Seeing that most students were successful, the researchers were convinced they learned how to construct the objects during the tasks they experienced previously. Applet 11 allowed students to use a slider or without a slider as it accommodated students' ideas. Depending on what they remembered or thought easy to use, they could decide whether to use a slider command or not. Additionally, to students' ownership of ideas, they could press run after each command they inputted, after some commands (many runs), or after all commands are inputted (1 run). Applet 11 facilitated students on what they thought was easier or more comfortable for them.

Table 6. Type or errors on the applet 11

No	Code	Description	Response	Run	Clear
1	SNsR1C0	Successful No-slider Run 1 and Clear 0	6	1	0
2	SNsR2C0	Successful No-slider Run 2 and Clear 0	2	2	0
3	SNsR5C0	Successful No-slider Run 5 and Clear 0	1	5	0
4	SNsR6C0	Successful No-slider Run 6 and Clear 0	1	6	0
5	SSR16C0	Successful Slider Run 16 and Clear 0	1	16	0
6	SSR1C0	Successful Slider Run 1 and Clear 0	2	1	0
7	SSR0C0	Successful Slider Run 0 and Clear 0	1	0	0
8	FR10C0	Failed Run 10 and Clear 0	1	10	0
9	E	Empty	2	0	0
	Total		17		

One of the teachers interviewed a student regarding what they had learned and how they perceived the tasks. The following is a conversation between the teacher (T1) and a student (S).

T1: What did you learn from today's lesson?
S: Making the polygon is easier. There is no need to make points one by one, but I could use the commands available here, using "polygon," and the same goes for the slider. No need to create one by one, but I could directly make many (vertices).
T1: What else did you experience? Do you think it is interesting?
S: Yes, attractive. Because it saved my life, without this I had to make them one by one, so this makes my work easier.
T1: What are your challenges in doing the tasks?
S: In the beginning, I was confused, the main point is to read the instructions carefully, and then you could do it successfully.

From the conversation, this student did not need to work hard on creating a polygon with a different number of vertices one by one, as instead of using the 'slider' command, it will create the manipulatable polygon. This could help to focus on the concept instead of spending time constructing polygons one by one, like on paper and pencil drawings.

The summary of the accomplishment of each applet (task) can be seen in Table 7. This has provided us with an overview of how most students could successfully work on the applets.

Table 7. Summary of each applet and its supporting CT components

Applet	Description	Successful rate
1	Constructing a hexagon	58.8%
2	Constructing a 13-sided polygon	82.4%
3	Constructing polygon with a slider	82.4%
4	Debugging polygon construction	88.2%
5	Constructing a unit circle	64.7%
6	Debugging a unit circle construction	76.5%
7	Constructing an inscribed polygon in a circle	47.1%
8	Debugging a polygon in a polygon	41.2%
9	Estimating pi	64.7%

Applet	Description	Successful rate
10	Programming without Algebra View	64.7%
11	Advanced programming without Algebra View	82.4%

Reflection from the Feedback

Besides the relatively high success rate, our applet is still open for improvement. This can be seen from the student's suggestion to make the applet better. This student thought that animations would make it more interesting and argued that many students do not like to read.

T1: What do you suggest for the applet to make it better?
S: Providing it with animations to make it more interesting, not only texts, as there are many students who do not like to read.
T1: This needs animations, doesn't it?
S: Yes
T1: To make it more interesting?
S: Yes

Additionally, from the WhatsApp group conversation, the researchers asked the teachers and the experts how to improve the initial design. The following are some of their feedback to improve the design:

> "I like the idea of a circle with a regular n-sided polygon inside it. The circle's area is calculated based on the area of the regular n-sided polygon." (T2)
>
> "I just cannot believe it, solving problems through coding ... insane ... May I confirm my understanding? This applet be conducted online, for the commands that are being used, they are in English, while for Chrome set up into Indonesian, commands in Indonesian can be used, cannot they?" Note: It seems like using a laptop provides a smooth experience with good visuals, according to students. On the Samsung 7 tablet, the visuals are still good, but there are concerns about the input bar response being slow and difficulties in deleting incorrect objects. Additionally, there was an incident where a participant accidentally deleted a question in Activity 1 (yellow box)." (T3)
>
> "Please try designing the applets so that they can be run on a mobile phone, as most students already have an Android phone at home." (T4)
>
> "The position of the input bar is not the same as in other applets." (P)

The teachers and practitioners have provided us with valuable insight that we were not aware of, such as the setting of the browser and the position of the input boxes, which were not consistent in different tasks. Based on the feedback, our applet design has improved. We became certain that the language setting on the browser would not translate the GeoGebra commands into the respective language, and we revised our input boxes to be consistent and located at the bottom of each task. Additionally, teachers have pointed out the type of devices used to execute the applets. From this, using small devices such as tablets or mobile phones could be more challenging in inputting the GeoGebra commands, and this leads us to suggest teachers to use bigger devices such as laptops or computers to carry out our Math+CT lessons.

Discussion

This section discusses our findings with related theories and studies. The researchers organized it into five parts: engagement, ownership of ideas, and exposure based on the constructionism theory by Papert (1980), as well as algorithms and debugging based on the CT framework by Shute et al. (2017).

Engagement

From the result, the researchers have witnessed that most students could accomplish the GeoGebra tasks we designed. It was not surprising that they did not do well on the first task, but they performed well on the later tasks, especially on applet 11. They might have been familiarized with what the tasks want them to do. For the engagement aspect, it shows students' involvement in participating in the tasks. The researcher collected how many clear and run buttons had been pressed, and it showed how students kept trying to accomplish the last task. They also sometimes revisited the previous tasks to help them solve the task they were working on.

Instead of introducing students to geometrical figures and giving them their properties, students constructed the points, polygons, angles, and circles. The researchers viewed that students have accomplished meaningful projects, as described by Lodi et al. (2019). Students could connect how the area of the regular polygon changes when the number of vertices increases. Additionally, students could reinvent the estimation of pi by exploring the slider on the polygon inscribed in the unit circle. This differs from the traditional teaching of the area of a circle and the pi, giving students the formula and the number directly. From this, the researchers showed that students could learn mathematics concepts through programming and debugging. This aligns with Kaufmann and Stenseth (2021) study that learning programming in mathematics lessons could enhance both programming and mathematics.

Programming could help students to work less to construct the object and focus more on mathematics. From the interview, one student utilized the slider command, making it faster for her instead of repeating the commands. This could be interpreted as attractive for this student, making her work easier. Kilhamn et al. (2021) found that programming could also engage students. This study also showed that the accomplishment rate for all the activities is quite high. Even though they struggled with the challenging debugging task, students did not easily give up and could still proceed with the following activities.

As Kynigos (2015) proposed to let students construct, modify, and tinker with digital artifacts, students in our study inputted the commands. They edited the mistakes on the Algebra View and did it on the edit, insert, and delete features that the researchers set on Applet 11. This shows that our design in GeoGebra has provided a learning environment for students to construct, modify, and tinker with digital artifacts. Even though the researchers have not yet used the half-baked artifact, instead, a program containing errors, students could tinker with that as a debugging activity. Nonetheless, the researchers should provide only fewer errors so that students would be able to handle this, as this was novel for them. The researchers have shown that fewer errors were more doable for students to handle as an amateur. The researchers also suspect the language used in the GeoGebra commands were in English would hinder students from seeing the mistakes (missing letters and typological) for "Cicle" or "Poligon" and understanding the pop-up notification. It might be different if the commands or syntaxes and notifications are in the student's native language. Related to debugging, Kaufmann and Stenseth (2021) found many cases in which students could change the code to make the program run properly. Our study has a similar finding that students could fix the program containing bugs or errors and successfully create the requested objects.

Ownership of Ideas and Learning Style

In terms of ownership of ideas and learning style, students can use any method to accomplish the tasks. Some restrictions might apply, such as the construction order or variables. However, the tasks still provided some space for flexibility and creativity. For instance, students have their own ways of finishing the last two GeoGebra tasks. The last tasks let students start differently and use any strategies to finish. For instance, in Table 6, some students applied a slider, and others did not. Students also could press the run button depending on their strategies. Our design tasks are open for using fewer commands if students could see this opportunity. However, students used only seven commands similar to the commands they previously worked on.

Exposure

Lastly, our pilot study has no discussion or presentation after students have completed all the lessons. Therefore, the researchers have yet to witness how exposure has emerged in our study. Students mostly worked individually on their gadgets, but some asked their friends if they faced difficulties. However, the researchers could listen to students' conversations. The two students discussed the missing error that had almost been forgotten. Students should write down what revisions they made to Andi's commands. It showed that a friend's ideas could be improved by listening to other friends. Additionally, it aligns with what Ackermann (2001) argued about shared knowledge being constructed through engaging in conversation around their or another's artifact. Moreover, this is relevant to epistemological plagiarism by Turkle and Papert (1990), which validates different ways of knowing and thinking. Additionally, purposefully designed peer feedback and discussion could encourage students to articulate their thinking and to understand their friends' perspectives (Butler & Leahy, 2021). Our two students have shared knowledge from the fruitful conversation and the peeping of a friend's artifact.

Algorithms

Most of our applets were dominated to support students' algorithms skills. van Borkulo et al. (2021) researched integrated CT in mathematics lessons utilizing a workbook (paper-based) and GeoGebra (technology-based) for pre-college students in the Netherlands and resulted in positive effects on students' algorithmic thinking (AT). Additionally, a recent study also supports the finding that students could develop their algorithmic thinking (AT) while learning calculus lessons in integrated CT and mathematics lessons utilizing a workbook and GeoGebra (Chytas et al., 2024). In our study, students worked only on the GeoGebra applets to construct geometrical objects to learn mathematics concepts (area of a circle and estimation of pi) by inputting GeoGebra commands. The differences between the previous two studies and our study were that students observed the patterns from the hints to create geometrical objects and requested them to construct similar objects with different sizes and coordinates. Therefore, despite supporting students' algorithmic thinking (AT), students also enhanced their pattern recognition sub-skill of abstraction skills by Shute et al. (2017).

Similar to Chytas et al. (2024), at the beginning, students struggle to input the commands on GeoGebra. As we can see in the first task, the success rate is rather low. Later, after getting familiar with the tasks and the tool, they performed better. Encountering longer commands to observe and construct similar problems (Applet 7) would be challenging for our students. In the last task, most students could construct the requested object successfully without any available hints. It has shown the progress of students in managing the order of the commands and variables to construct an inscribed hexagon on a circle in GeoGebra.

Debugging

Some applets in this study support students in developing their debugging skills. This skill requires students to detect and identify errors and then fix the errors. This study designed learning opportunities specifically for students to develop debugging skills by providing them with a fictional character, 'Andi,' with his GeoGebra commands containing some errors. Other studies that specifically provided students with debugging tasks are Liu et al. (2017) with their debugging BOTS game and Arfé et al. (2020) with their inhibition tasks. Both studies revealed positive effects on students' debugging skills. This is in line with our finding that students could detect errors, especially the debugging tasks, with few errors after they learned some commands to construct geometrical objects. However, according to Liu et al. (2017), the BOTS game did provide a pop-up message that informed the students of the errors, and the program still ran but did not successfully accomplish the goal. It is in line with the weakness of block programming, which provides little space for debugging or error notifications as it was not designed to do so (Liu et al., 2017), and block programming prevents syntax errors from happening (Resnick, 2012). Our study utilized GeoGebra, which provided students with a pop-up notification to detect the errors in two ways, namely immediate style interruption and negotiatedstyle interruption, coined by Robertson et al. (2004). These interruptions have supported students to debug the command errors on GeoGebra (Yunianto, Bautista, et al., 2023; Yunianto, Prodromou, et al., 2023). Additionally, the English language could be a barrier for non-English speaking students to understand the meaning of the pop-up.

Despite the specific tasks to develop debugging skills, when students inputted commands to GeoGebra, they made mistakes, and they had to edit or delete their incorrect commands. Applet 11 provided students with edit, delete, and insert features for fixing their commands. They could utilize the edit feature to amend their inputted commands or insert a feature to add a missing command. This study has not recorded how many times students used these features as we have not developed a program to record this behavior on GeoGebra. Additionally, we noticed that students' mathematics concepts might be related to students' ability to debug. For instance, if students have understood the universal way to write a point on a Cartesian coordinate as (x,y) with a comma as a separator, they might be able to easily detect Andi's command A(1:1), which used a colon as the separator. This is in line with mathematics, and CT skills could develop together and co-develop (Ye et al., 2023), but when students' mathematical concepts are poor, it would hinder students' ability to debug. In the last task, most students could construct the requested object successfully by inputting commands without pressing the clear button to start over.

Reflection on Our Math+CT Lessons

After implementing the pilot study with a few students, we learned from the experiences and reflected on those experiences to improve the applets and the subsequent implementations. First, our GeoGebra applets will not work optimally if they are run on small devices such as smartphones, tablets, and iPads, as the keypad will differ from the keypad on laptops or personal computers (PCs) (Figure 22). The keypad differs from the QWERTY keyboard, and this locates numbers, functions, alphabets, and symbols in different tabs, making students struggle to input a simple command like A=(1,1).

Figure 22. Keypad composition on a smartphone while inputting GeoGebra commands

Our applets require an internet connection to be run. Therefore, schools that have stable internet connections may apply or adapt our applet for their students. Technical errors, such as not showing the task, might happen when a poor internet connection occurs. Moreover, our applet can be used in different countries as they can be modified and translated into different languages. We also experienced our students struggling with debugging too many errors. It would be wise to provide students with fewer errors to fix and gradually progress to more errors to be fixed.

If educators intend to deliver Math+CT lessons, they should take into account the specific recommendations provided by this study. We can start by incorporating computational thinking (CT) skills into mathematics learning by utilizing tools such as GeoGebra to actively involve students in working with mathematical inquiries from computational perspectives. Additionally, bigger devices are recommended for the implementation of our GeoGebra applets. Moreover, we should promote cooperative problem-solving and inquiry-based learning methods in which students utilize computational thinking skills to solve real-world problems, with support from peer discussions and feedback.

Additionally, we developed our Math+CT lessons collaboratively, and educators who wish to develop Math+CT lessons should work with other educators or seek help from experts. Moreover, we could integrate computational thinking into the current math curriculum by including essential CT skills throughout different mathematics topics. We can employ technological tools such as GeoGebra and other mathematics software such as spreadsheets to generate interactive learning experiences in which students customize variables and emulate real-world problems.

CONCLUSION

The summary of the paper offered evidence of students' learning in a constructionist manner with our math+CT lessons spread across eleven applets. The engagement and the ownership of ideas were observable. Even though the researcher did not set up classroom presentations for students, two students in this study had a fruitful discussion on their artifacts and debugging process. They built shared understanding by working and checking on one another's work. In terms of CT skills supported in this study, the researchers witnessed that our designed lessons have facilitated students to be active and engaged in algorithms and debugging with good performances. Our GeoGebra applets are relatively easy to use and adjust. Therefore, interested teachers might apply our applets directly or modify them for students. However, we realized that our design is imperfect and has much room for improvement. Our study involved only a few students, with three to four students from five different schools. Therefore, the generalization of this study is limited. Expanding the intervention to a larger sample size could yield more comprehensive findings about the effectiveness of Mathematics + Computational Thinking (Math+CT) lessons across different student demographics. This strategy would entail broadening the study to include a larger number of participants with diverse educational backgrounds, providing a more thorough insight into how various student groups gain benefits from the integrated approach. Furthermore, performing subsequent investigations in different schools, mathematics topics, and task sequences will enhance the external validity of the results. The different task sequencing might result in different findings, and researchers can further investigate task sequencing and its effect.

Additionally, examining the effectiveness of Math+CT lessons in various educational settings with diverse student demographics or technological tools may highlight subtle differences in implementation and results. This comparison research not only confirms the intervention's transferability but also provides guidance for making the necessary adjustments to optimize its success in other learning contexts. Further research also could investigate the changes in students' attitudes toward mathematics and examine the effect of Math+CT lessons in a longitudinal study.

REFERENCES

Ackermann, E. (2001). Piaget's constructivism, Papert's constructionism: What's the difference?. *Future of learning group publication*, *5*(3), 438. http://learning.media.mit.edu/content/publications/EA.Piaget%20_%20Papert.pdf

Arfé, B., Vardanega, T., & Ronconi, L. (2020). The effects of coding on children's planning and inhibition skills. *Computers and Education*, *148*, 1–16. https://doi.org/10.1016/j.compedu.2020.103807

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *Journal of Applied Computing and Information Technology*, *13*(1), 20–29.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education*, *72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020

Boaler, J. (1998). Open and closed mathematics: Student experiences and understandings. *Journal for Research in Mathematics Education*, *29*(1), 41–62. https://doi.org/10.2307/749717

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implications for policy and practice.* European Commission. https://doi.org/10.2791/792158

Bocconi, S., Chioccariello, A., Kampylis, P., Dagienė, V., Wastiau, P., Engelhardt, K., Earp, J., Horvath, M., Jasutė, E., Malagoli, C., Masiulionytė-Dagienė, V., Stupurienė, G., Giannoutsou, N., Inamorato dos Santos, A., Punie, Y., & Cachia, R. (2022). *Reviewing computational thinking in compulsory education.* Publications Office of the European Union. https://doi.org/10.2760/126955

Brady, C., Weintrop, D., Anton, G., & Wilensky, U. (2016). Constructionist learning at the group level with programmable badges. *Proceedings of the Constructionism Conference*, *Bangkok, Thailand,* 61–68.

Butler, D., & Leahy, M. (2021). Developing preservice teachers' understanding of computational thinking: A constructionist approach. *British Journal of Educational Technology*, *52*(3), 1060–1077. https://doi.org/10.1111/bjet.13090

Chan, S.-W., Looi, C.-K., Ho, W. K., Huang, W., Seow, P., Wu, L., & Kim, M. S. (2020). Computational thinking activities in number patterns: A study in a Singapore secondary school. In H.-J. So, M. M. Rodrigo, J. Mason, & A. Mitrovic (Eds.), *Proceedings of the 28th International Conference on Computers in Education* (Vol. I, pp. 171-176). Asia-Pacific Society for Computers in Education. https://api.elsevier.com/content/abstract/scopus_id/85099434625

Chen, O., Retnowati, E., Chan, B. K. Y., & Kalyuga, S. (2023). The effect of worked examples on learning solution steps and knowledge transfer. *Educational Psychology*, *43*(8), 914–928. https://doi.org/10.1080/01443410.2023.2273762

Chytas, C., van Borkulo, S. P., Drijvers, P., Barendsen, E., & Tolboom, J. L. J. (2024). Computational thinking in secondary mathematics education with *GeoGebra*: Insights from an intervention in calculus lessons. *Digital Experiences in Mathematics Education*, *10*, 228–259. https://doi.org/10.1007/s40751-024-00141-0

FIsme. (2015, August). *Master theses of the group of 2010-2014*. IMPOME. http://www.fisme.science.uu.nl/en/impome/

GeoGebra. (2024). *What is GeoGebra?* https://www.geogebra.org/about

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

Hamzah, N. A. H., & Hidayat, R. (2022). The role of Geogebra software in mathematics education: A systematic literature review. *Jurnal Pendidikan Sains Dan Matematik Malaysia*, *12*(1), 24–38.

Hermans, F., & Aivaloglou, E. (2017). To scratch or not to scratch? A controlled experiment comparing plugged first and unplugged first programming lessons. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 49–56). Association for Computing Machinery. https://doi.org/10.1145/3137065.3137072

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education*, *4*, 48–69. https://doi.org/10.1007/s40751-017-0038-8

Ho, W. K., & Ang, K. C. (2015). Developing computational thinking through coding. *Proceedings of the 20th Asian Technology Conference in Mathematics*, *Leshan, China,* 73–87.

Ho, W. K., Looi, C. K., Huang, W., Looi, C. K., & Longanathan, S. (2018). Can secondary school mathematics students be taught to think computationally? *Proceedings of the 23rd Asian Technology Conference in Mathematics*.

Ho, W. K., Looi, C. K., Huang, W., Seow, P., & Wu, L. (2019). Realizing computational thinking in the mathematics classroom: Bridging the theory-practice gap. *Proceedings of the 24th Asian Technology Conference in Mathematics*, *Leshan, China,* 35–49.

Ho, W. K., Looi, C. K., Huang, W., Seow, P., & Wu, L. (2021). Computational thinking in mathematics: To be or not to be, that is the question. In T. L. Toh, & B. H. Choy (Eds.), *Mathematics – Connection and Beyond* (pp. 205–234). World Scientific. https://doi.org/10.1142/9789811236983_0011

Hoyles, C. (2015, June 19). *Revisiting programming to enhance mathematics learning.* Presentation at Math + Coding Symposium, Western University, London. https://researchideas.ca/coding/

Irawan, E., Rosjanuardi, R., & Prabawanto, S. (2024). Advancing computational thinking in mathematics education: A systematic review of Indonesian research landscape. *Jurnal Teori Dan Aplikasi Matematika*, *8*(1), 176–194. https://doi.org/10.31764/jtam.v8i1.17516

Juandi, D., Kusumah, Y. S., Tamur, M., Perbowo, K. S., & Wijaya, T. T. (2021). A meta-analysis of Geogebra software decade of assisted mathematics learning: What to learn and where to go? *Heliyon*, *7*(5), e06953. https://doi.org/10.1016/j.heliyon.2021.e06953

Kaufmann, O. T., & Stenseth, B. (2021). Programming in mathematics education. *International Journal of Mathematical Education in Science and Technology*, *52*(7), 1029–1048. https://doi.org/10.1080/0020739X.2020.1736349

Khoo, N. A. K. A. F., Ishak, N. A. H. N., Osman, S., Ismail, N., & Kurniati, D. (2022). Computational thinking in mathematics education: A systematic review. *Proceedings of the 1st International Conference of Mathematics Education, Learning and Application*. https://doi.org/10.1063/5.0102618

Kilhamn, C., Bråting, K., & Rolandsson, L. (2021). Teachers' arguments for including programming in mathematics education. *Proceedings of NORMA 20, The Ninth Nordic Conference on Mathematics Education, Oslo, Norway*.

King, A. (2013). Mathematical explorations: Finding Pi with Archimedes's exhaustion method. *Mathematics Teaching in the Middle School*, *19*(2), 116–123. https://doi.org/10.5951/mathteacmiddscho.19.2.0116

Kravik, R., Berg, T. K., & Siddiq, F. (2022). Teachers' understanding of programming and computational thinking in primary education - A critical need for professional development. *Acta Didactica Norden*, *16*(4), Article 9. https://doi.org/10.5617/adno.9194

Krippendorff, K. (2004). *Content analysis: An introduction to its methodology* (2nd ed.). Sage. https://doi.org/10.1103/PhysRevB.31.3460

Kushwaha, R. C., Chaurasia, P. K., & Singhal, A. (2013). Creating dynamic webpage for GeoGebra quiz applet. *International Journal of Information and Computation Technology*, *3*(3), 175–180.

Kynigos, C. (2007). Using half-baked microworlds to challenge teacher educators' knowing. *International Journal of Computers for Mathematical Learning*, *12*, 87–111. https://doi.org/10.1007/s10758-007-9114-2

Kynigos, C. (2015). Constructionism: Theory of learning or theory of design? In S. J. Cho (Ed.), *Selected regular lectures from the 12th International Congress on Mathematical Education* (pp. 417–438). Springer. https://doi.org/10.1007/978-3-319-17187-6_24

Lavicza, Z., Hohenwarter, M., Jones, K., & Dawes, M. (2010). Establishing a professional development network around dynamic mathematics software in England. *International Journal for Technology in Mathematics Education*, *17*(4), 177–182.

Liu, Z., Zhi, R., Hicks, A., & Barnes, T. (2017). Understanding problem solving behavior of 6–8 graders in a debugging game. *Computer Science Education*, *27*(1), 1–29. https://doi.org/10.1080/08993408.2017.1308651

Lodi, M., Malchiodi, D., Monga, M., Morpurgo, A., & Spieler, B. (2019). Constructionist attempts at supporting the learning of computer programming: A survey. *Olympiads in Informatics, 13*, 99–121. https://doi.org/10.15388/ioi.2019.07

Mackrell, K., & Pratt, D. (2017). Constructionism and the space of reasons. *Mathematics Education Research Journal*, *29*(4), 419–435. https://doi.org/10.1007/s13394-017-0194-6

McKenney, S., & Reeves, T. C. (2018). *Conducting educational design research* (2nd ed.). Routledge. https://doi.org/10.4324/9781315105642

Nordby, S. K., Bjerke, A. H., & Mifsud, L. (2022). Primary mathematics teachers' understanding of computational thinking. *Künstliche Intelligenz*, *36*, 35–46. https://doi.org/10.1007/s13218-021-00750-6

Papert, S. (1972). Teaching children thinking. *Programmed Learning and Educational Technology*, *9*(5), 245–255. https://doi.org/10.1080/1355800720090503

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas* (1st ed.). Basic Books.

Resnick, M. (2012). Reviving Papert's dream. *Educational Technology*, *52*(4), 42–46.

Robertson, T. J., Prabhakararao, S., Burnett, M., Cook, C., Ruthruff, J. R., Beckwith, L., & Phalgune, A. (2004). Impact of interruption style on end-user debugging. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 287–294). Association for Computing Machinery. https://doi.org/10.1145/985692.985729

Rottenhofer, M., Sabitzer, B., & Rankin, T. (2021). Developing computational thinking skills through modeling in language lessons. *Open Education Studies*, *3*(1), 17–25. https://doi.org/10.1515/edu-2020-0138

Sabitzer, B., Demarle-Meusel, H., & Jarnig, M. (2018, April). Computational thinking through modeling in language lessons. *Proceedings of the IEEE Global Engineering Education Conference, Santa Cruz de Tenerife, Spain,* 1913–1919. https://doi.org/10.1109/EDUCON.2018.8363469

Schoenfeld, A. H. (1988). When good teaching leads to bad results: The disasters of "well-taught" mathematics courses. *Educational Psychologist*, *23*(2), 145–166. https://doi.org/10.1207/s15326985ep2302_5

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

Sneider, C., Stephenson, C., Schafer, B., & Flick, L. (2014). Computational thinking in high school science classrooms. *The Science Teacher*, *81*(5), 53–59. https://doi.org/10.2505/4/tst14_081_05_53

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

Suprayitno, T. (2019, September 18-19). *Opening remarks*. The international conference on embedding artificial intelligence (AI) in education policy and practice for Southeast Asia. Jakarta, Indonesia. https://www.seamolec.org/article/the-international-conference-on-embedding-artificial-intelligence-ai-in-education-policy-and-practice-for-southeast-asia

Turkle, S., & Papert, S. (1990). Epistemological pluralism: Styles and voices within the computer culture. *Signs: Journal of Women in Culture and Society*, *16*(1), 128–157. https://doi.org/10.1086/494648

van Borkulo, S. P., Chytas, C., Drijvers, P., Barendsen, E., & Tolboom, J. (2021). Computational thinking in the mathematics classroom: Fostering algorithmic thinking and generalization skills using dynamic mathematics software. *Proceedings of the 16th Workshop in Primary and Secondary Education* (Article 19). Association for Computing Machinery. https://doi.org/10.1145/3481312.3481319

van Borkulo, S. P., Chytas, C., Drijvers, P., Barendsen, E., & Tolboom, J. (2023). Spreadsheets in secondary school statistics education: Using authentic data for computational thinking. *Digital Experiences in Mathematics Education*, *9*(3), 420–443. https://doi.org/10.1007/s40751-023-00126-5

Velikova, E., & Petkova, M. (2019). Analysing students' creativity in integrating GeoGebra applets in solving geometrical problems. *Baltic Journal of Modern Computing*, *7*(3), 419–429. https://doi.org/10.22364/bjmc.2019.7.3.08

Wang, C., Shen, J., & Chao, J. (2022). Integrating computational thinking in STEM education: A literature review. *International Journal of Science and Mathematics Education*, *20*(8), 1949–1972. https://doi.org/10.1007/s10763-021-10227-5

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

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

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

Yohannes, A., & Chen, H. L. (2021). GeoGebra in mathematics education: A systematic review of journal articles published from 2010 to 2020. *Interactive Learning Environments*, *31*(9), 5682–5697. https://doi.org/10.1080/10494820.2021.2016861

Yunianto, W., Bautista, G., Indra Prahmana, R. C., & Lavicza, Z. (2023, November). GeoGebra applet to learn programming and debugging in mathematics lessons. *Proceedings of the International Symposium on Computers in Education*, *Setúbal, Portugal*. https://doi.org/10.1109/SIIE59826.2023.10423707

Yunianto, W., Prodromou, T., & Lavicza, Z. (2023, December). Debugging on GeoGebra-based Mathematics+Computational Thinking lessons. Proceedings of the *28th Asian Technology Conference in Mathematics*, *Pattaya, Thailand,* 322–331.

Zhong, B., Wang, Q., Chen, J., & Li, Y. (2016). An exploration of three-dimensional integrated assessment for computational thinking. *Journal of Educational Computing Research*, *53*(4), 562–590. https://doi.org/10.1177/0735633115608444

Ziatdinov, R., & Valles, J. R. (2022). Synthesis of modeling, visualization, and programming in GeoGebra as an effective approach for teaching and learning STEM topics. *Mathematics*, *10*(3), 398. https://doi.org/10.3390/math10030398

AUTHORS

Wahid Yunianto is experienced in training mathematics teachers in Southeast Asia in utilizing technologies for teaching and learning of mathematics. He has been working in several research projects such as utilizing scientific calculators for learning mathematics and developing STEAM+CT modules. His research interest is to support mathematics teachers and students to experience meaningful mathematics activities especially when utilizing digital technologies. He is currently pursuing his PhD study at Johannes Kepler University Linz supported by OeAD Ernst Mach Grant Scholarship. His research focuses on integrating computational thinking into mathematics lessons.

Dr. Houssam El-Kasti, from Beirut, Lebanon, is a distinguished academic currently serving as a Mathematics lecturer at Qatar University's foundation program. With a wealth of experience in the field of mathematics education, Dr. El-Kasti specializes in equipping educators to impart conceptual mathematics using technology. His pedagogical journey spans various esteemed institutions, including the American University of Beirut (AUB) and the Lebanese University, where he has adeptly taught both undergraduate and graduate mathematics education courses.

In his most recent international collaboration, Dr. El-Kasti has joined forces with Johannes Kepler University Linz (JKU), Austria, focusing on advancing mathematical education theories, innovative design, and the promotion of STEAM education initiatives. Embodying his personal ethos, "Never stop learning, never stop teaching," Dr. El-Kasti epitomizes a lifelong commitment to education and knowledge dissemination.

Univ-Prof. Dr. Zsolt Lavicza, PhD. Linz School of Education, STEAM Education Centre, Johannes Kepler University, Linz, Austria. Zsolt Lavicza has worked on several research projects examining technology and mathematics teaching in classroom environments at the Universities of Michigan and Cambridge. In addition, Zsolt has greatly contributed to the development of the GeoGebra community and participated in developing research projects on GeoGebra and related technologies worldwide. Currently, Zsolt is a Professor in STEAM Education Research Methods at Johannes Kepler University's Linz School of Education. From JKU he is working on numerous research projects worldwide related to technology integration into schools; leading the doctoral programme in STEAM Education; teaching educational research methods and coordinating research projects around the world.

Rully Charitas Indra Prahmana is a Professor in Mathematics Education at Universitas Ahmad Dahlan, Yogyakarta, Indonesia. His research interests lie in the Etnomathematics, Design Research, and Realistic Mathematics Education.