Debugging on GeoGebra-based Mathematics+Computational Thinking lessons

Wahid Yunianto[1], *Theodosia Prodromou*[2], *Zsolt Lavicza*[3]
yunianto.wah@gmail.com, theodosia.prodromou@une.edu.au, zsolt.lavicza@jku.at
[1,3]Johannes Kepler University Linz, Austria; [2]University of New England, Australia

Abstract: *Computational thinking (CT) has become a buzzword recently and gained more attention from countries and researchers. Researchers realize the importance of CT and integrate it into school subjects such as mathematics, science, language, and others. Our research tries to contribute to the plugged CT activities under mathematics subjects. Collaborating with mathematics teachers, principals, and a teacher trainer, we developed a sequence of lessons in GeoGebra. Our lessons integrate CT's facets in the topic of the area of a circle. The development of the GeoGebra-based Mathematics-CT lessons incorporated educational design research methodology. We improved our lessons and implemented them for a few students. In this paper, we focused only on the debugging skill being supported by GeoGebra. Our findings show that fixing commands can be challenging as students have been through several debugging, and it can be complicated if the errors are many. This paper shows the power of GeoGebra to learn integrated CT in mathematics lessons through creating objects and debugging the program.*

1. Introduction

This paper is part of a larger study of integrating computational thinking in mathematics education using readily available mathematics software for mathematics teachers. Many countries have taken action to make Computational Thinking (CT) available in the school curriculum [1] and prepare their in-service and pre-service teachers with CT knowledge and skills [2], [3]. The Ministry of Education, Culture, Research, and Technology (MoERT) of Indonesia also followed the movement and developed the computer science (CS) school subject that was unavailable previously. Students will learn CT from this CS subject. Wing [4] argued that CT could be developed not only in CS courses. To contribute to this movement, we support CT through mathematics subjects. GeoGebra is a free open-source mathematics software that allows teachers and students to use it online or offline.

As CT is relatively new in education, in order to be successfully implemented, mathematics teachers need support with exemplary activities of integrating CT into mathematics lessons [5]. Our research is to develop GeoGebra-based mathematics-CT lessons for junior high school students. GeoGebra is a powerful, interactive geometry, algebra, statistics, and calculus software, intended for learning and teaching mathematics and science from primary. There are a few studies that used GeoGebra to integrate CT into mathematics lessons [6], [7] Van Borkulo et al. [6] found that GeoGebra could support CT's algorithmic thinking and generalization aspects. We want to investigate this issue more and enrich our understanding of how GeoGebra can support mathematics and CT.

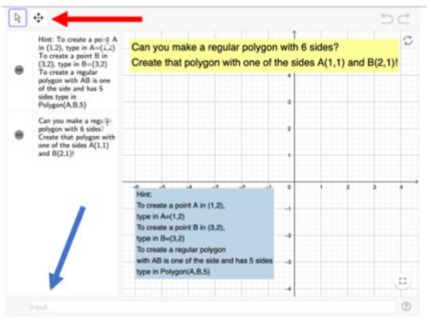
Through the educational design research approach [8], the researchers collaborated with mathematics teachers, principals, and a teacher trainer to develop GeoGebra-based mathematics-CT lessons. We refer to the framework by Shute et al. [9] which improved the previous existing framework by other researchers and gave us more operationalised CT facets to be developed in our lessons.

We developed the lessons based on the Indonesian curriculum mathematics content, determining the area of a circle and its related problems. The area of a regular polygon with many sides can be used to approach the area of a circle as per Archimedes' method of exhaustion see [10]. Hence, we developed two lessons which consisted of constructing a regular polygon to inscribe in a circle. Unlike the activity proposed by King [10] that explored the circle's circumference and its formula, our study focuses on the area of a unit circle by approaching it with an inscribed polygon. Students will program to construct a manipulatable polygon inside the unit circle. We intentionally allow our students program to construct an inscribed polygon on a circle on GeoGebra and debug errors. The debugging facet will be the focus of this paper.

2. Theoretical Framework

Our study is based on the constructionist theory by Papert [11] who introduced computational thinking in education, especially in mathematics education. His idea of computational thinking is about interacting with a learning environment in the computer to learn or access knowledge. Papert proposed a design framework that encompasses activity engagement, ownership of ideas and learning style, and exposure. The GeoGebra-based mathematics and CT lessons were designed for students to construct objects (engagement), arrange the commands based on their ideas (ownership of ideas) and work on similar tasks or sequences (exposure).

Papert was inspired by the way Computer scientists solved difficult problems using LISP programming and he developed a similar tool for young learners to do mathematics with it [12]. This inspired Papert and his colleagues to develop LOGO, a similar tool mentioned earlier that later evolved into Turtle Geometry. Young learners used this tool to construct geometric figures. Our study incorporated GeoGebra to develop an environment where students can input their commands to construct objects and learn mathematical concepts or solve problems. GeoGebra can be used as a programming tool by its input box feature and scripts or commands. We intentionally hid the drawing or construction tools by displaying only the cursor/pointer or move tool (Figure 2. 1). We also developed the modified input box and hidden 'Algebra View' on the left side of the GeoGebra window. Students could edit, delete, or insert a command if they needed to do it.


Hint: To create a point A in (1,2), type in A=(1,2) To create a point B in (3,2), type in B=(3,2) To create a regular polygon with AB is one of the side and has 5 sides type in Polygon(A,B,5)

Can you make a regular polygon with 6 sides? Create that polygon with one of the sides A(1,1) and B(2,1)!

Can you make a regular polygon with 6 sides?
Create that polygon with one of the sides A(1,1) and B(2,1)!

Hint:
To create a point A in (1,2),
type in A=(1,2)
To create a point B in (3,2),
type in B=(3,2)
To create a regular polygon
with AB is one of the side and has 5 sides
type in Polygon(A,B,5)

Figure 2. 1 The GeoGebra environment setting for Math-CT activity

We allowed students to interact with this GeoGebra environment and create objects, in this case, a polygon step by step, creating points and then the number of the sides of the polygon. Later, we expect them to explore the slider to observe the change in the polygon's area. In the end, students would create a unit circle with a manipulatable polygon inside it to approach the area of the circle.

3. Methodology

We use the educational design research (EDR) approach [8] in order to understand whether our design activity works or not and why it works or does not work. Through the iterative process of the EDR, we collaborated with mathematics teachers, principals, and a teacher educator to develop GeoGebra-based mathematics-CT lessons. We refined our activities based on our online meetings, and reflection from the pilot study. We tried our lessons with seventeen junior high school students to see how our lessons worked.

The framework. In this section, we refer to the CT framework by Shute et al. [9] involves six facets, which are decomposition, abstraction, algorithms, debugging, iteration, and generalisation. These facets and their operationalised definitions helped us to design our lessons. We will only describe the debugging facet in this paper due to page limitations. Debugging requires students to do systematic testing and modification. It can be seen when students can detect and identify errors, and then fix the errors. Additionally, when the solution or program does not work, they can identify, and fix the errors to make it work.

Our study promoted debugging by fixing the program to construct objects (polygon, circle, or inscribed polygon on a circle). In the debugging problem, we created a fictional character Andi who has written a program. The students should detect and identify whether Andi's command can be executed to successfully construct the intended objects. We allow students to learn how to properly use the commands or syntax to construct objects (point, polygon, slider, circle, etc.). For instance, to construct a circle with a center in A(1,1) and a radius of 3, the correct commands should be A=(1,1), Circle(A,3).

If they do not follow these commands, students will not be able to construct the mentioned circle. After they learned how to program using the correct commands or syntax, they would be exposed to incorrect commands. Later, they were challenged to recognise if the provided commands contained errors, and they had to fix the errors. Therefore, our debugging tasks followed the debugging facet.

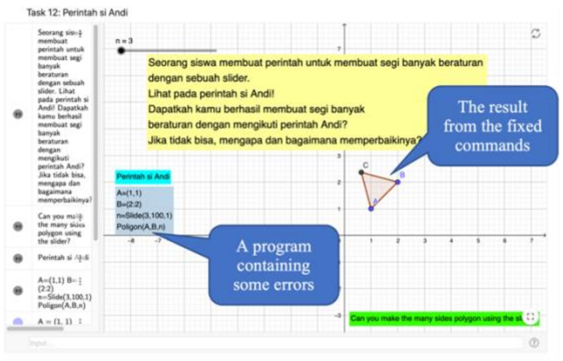
Debugging. Our study wants to contribute to debugging practices as in [13] that debugging needs more attention from researchers. Debugging can form a remedial activity after writing codes or a purposively designed activity for students to explain, find, and fix bugs [14]. Our debugging tasks were more into the second form of debugging as students were required to find the errors of the existing commands (Andi's command). Visual programming tends to provide little space for debugging as it was not designed to do so [15], and block programming prevents syntax errors from happening [16]. Thus, our GeoGebra-based mathematics+CT, non-visual programming, could allow students to make and correct errors.

To detect errors, the programming tools notify the programmers directly and specifically to which part causes the errors. Robertson et al. [17] differentiate these notifications as styles of interruption (Table 3. 1). They are negotiated-style interruption and immediate-style interruption [17]. They describe that negotiated-style interruption when the programmer gets a pending message of the errors so that they will know it later. Meanwhile, the immediate-style intervention; such as a pop-up window, informs the programmer to take action immediately.

Table 3. 1 Styles of interruption in debugging

No	Command	Types of Interruption	Description
1	A=(1:1)	negotiated-style interruption	Students will not see this as error as GeoGebra will run this and create a slider.
2	Circle(A,3)	immediate-style interruption	Students will get a pop-up where the command 'Circle' is not known. Thus, GeoGebra cannot make a circle as the command missed the letter 'r'. Students must immediately revise the command.

The context. There are 33 tasks in the GeoGebra lessons that we developed. The lessons can be found here *https://www.geogebra.org/classroom/g7agtjdh*. The tasks that involve GeoGebra manipulations are Task 1, 5, 8, 12, 15, 17, 20, 23, 27, 31, and 33. The debugging tasks are tasks 12, 17, and 23. The goal of all the tasks is to let the students understand the approximation of pi using the area of a polygon inscribed in a circle and the area formula of a circle.


Task 12: Perintah si Andi

Seorang siswa membuat perintah untuk membuat segi banyak beraturan dengan sebuah slider.
Lihat pada perintah si Andi!
Dapatkah kamu berhasil membuat segi banyak beraturan dengan mengikuti perintah Andi?
Jika tidak bisa, mengapa dan bagaimana memperbaikinya?

Perintah si Andi

```
A=(1,1)
B=(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
```

Can you make the many sides polygon using the slider?

n = 3
The result from the fixed commands
A program containing some errors
C
B
A

Figure 3. 1 Task 12 display

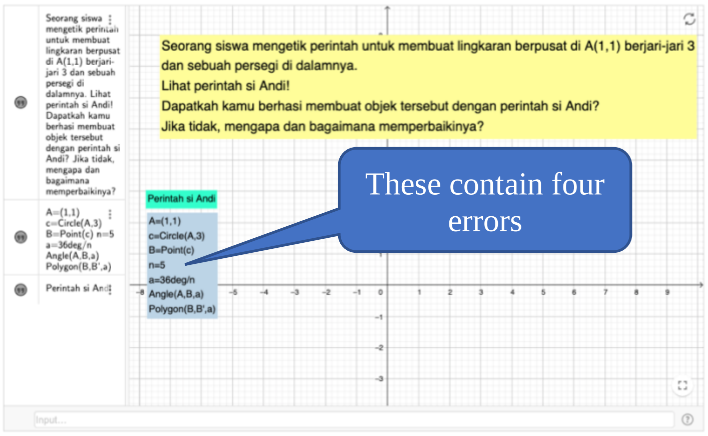
Task 12 was designed to promote debugging skills for students on the manipulatable polygon construction. Students were provided with a list of commands which included some errors. The errors started with up to three script errors. We coded the errors from what students did to solve the debugging tasks. In task 12, regarding the debugging activity, (Figure 3. 1), students must determine whether they can follow Andi's command to make a manipulatable polygon. If students cannot follow the command, then they must fix it. Three (3) errors are contained in Andi's command for this task. The errors are namely, using a dot instead of a comma for making a point, the missing letter, and the typographical error when he used the letter 'i' instead of the letter 'y'. There are four rows or lines on Andi's command, and the three are incorrect (Table 3. 2).

Table 3. 2 Errors on Task 12

Line	Correct Command	Andi's command	Description
1	A=(1,1)	A=(1,1)	
2	B=(2,2)	B=(2:2)	The use of colon : instead of comma ,
3	n=Slide(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

The errors are in the rows 2, 3, and 4. Students had to investigate whether Andi's command was correct or not. They also had to fix the command to make it right. Students could produce a manipulatable polygon from an equilateral triangle to a 100-gon by fixing the commands' errors. The slider named n can be moved from 3 to 100 to construct the desired polygon.

Task 23 required students to do debugging from more complicated commands (Figure 3. 2). It had more errors found in rows 4,5,6,7 and errors in mathematical concepts such as the internal angle. Students had to debug n=5, a=36deg/n, Angle(A,B,a), and Polygon(B,B',a) to be n=4, a=360deg/n, Angle(B,A,a), and Polygon(B,B',n) respectively.



Seorang siswa mengetik perintah untuk membuat lingkaran berpusat di A(1,1) berjari-jari 3 dan sebuah persegi di dalamnya.
Lihat perintah si Andi!
Dapatkah kamu berhasi membuat objek tersebut dengan perintah si Andi?
Jika tidak, mengapa dan bagaimana memperbaikinya?

Perintah si Andi

```
A=(1,1)
c=Circle(A,3)
B=Point(c)
n=5
a=36deg/n
Angle(A,B,a)
Polygon(B,B',a)
```

Seorang siswa mengetik perintah untuk membuat lingkaran berpusat di A(1,1) berjari-jari 3 dan sebuah persegi di dalamnya. Lihat perintah si Andi! Dapatkah kamu berhasi membuat objek tersebut dengan perintah si Andi? Jika tidak, mengapa dan bagaimana memperbaikinya?

A=(1,1) c=Circle(A,3) B=Point(c) n=5 a=36deg/n Angle(A,B,a) Polygon(B,B',a)

Perintah si And

Figure 3. 2 Task 23 display

Data Analysis. The data were collected through video screen recording when students worked on the lesson activities as well as on students' GeoGebra files. The GeoGebra files might not show all the processes, as students might have deleted what they typed in. Thus, we could see it from the video recording. We used content analysis by Krippendor [18] to analyse the videos by making, categorising, and concluding the codes. We limit the result by using a screen video recording of one student from the pilot study. Additionally, due to page limitations, we can only show some tasks. The tasks that we presented here are related to debugging only.

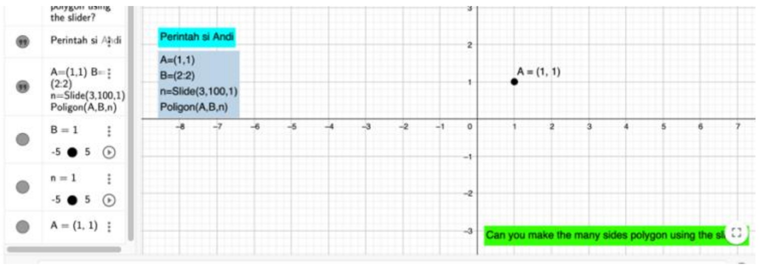
4. Result

For task 12, the majority (12 students) could answer it successfully. It seems that at least students could fix the command for creating a point (Table 4. 1). However, in further investigation, the point command could be troublesome for some students, and they would have noticed it in the immediate-type of interruption (see Figure 4. 5).

Table 4. 1 Coding for students' responses

Description	Code	Number of students
Inputted the correct commands	IC	12
Only made point A	P	3
Empty	E	1
Made some correct commands and did not continue	SC	1

These are some mistakes we found in our students (Table 4. 1). They could successfully make point A and failed to make point B. If students followed Andi's command, then when they made a point B, they would not produce the correct point B. They created a slider B instead (Figure 4. 1). We coded this as making only point A."P" stands for Point, as the students produced no other objects visible on the grid.



```
the slider?
Perintah si Andi
A=(1,1) B=
(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
B = 1
-5 ● 5
n = 1
-5 ● 5
A = (1, 1)
```

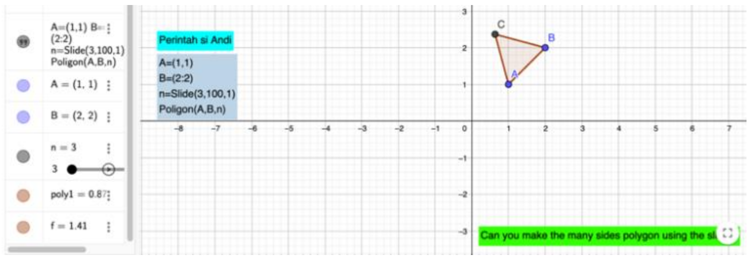
```
Perintah si Andi
A=(1,1)
B=(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
```

Figure 4. 1 A student made a B as a slider instead of a point

Other students also made mistakes by trying another way, such as B=(2 2) or B=(2<space>2), resulting in a number 4. In the end, students did not produce point B or other objects as they failed to debug the next errors. We coded this as making only a point A, "P", as the students had no other objects visible on the grid.

Some students stopped at a certain command after fixing or inputting the corrected commands. They successfully made some corrections, and the last command for the polygon was missing. This code belongs to making some correct commands and discontinuing to complete other commands, "SC" stands for some correct.

The successful students could make a desired object not by following Andi's command. They fixed the incorrect commands and resulted in the manipulatable polygon (Figure 4. 2). We coded this as an inputted correct command, "IC" stands for Inputting Correct. Other students did not do anything. We coded it as "E".

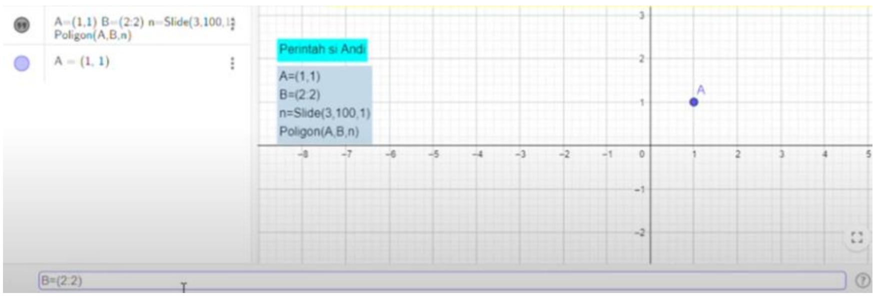


```
A=(1,1) B=
(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
A = (1, 1)
B = (2, 2)
n = 3
3
poly1 = 0.87
f = 1.41
```

```
Perintah si Andi
A=(1,1)
B=(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
```

Figure 4. 2 A student successfully fixed and ran the program

The following paragraphs describe how one student in our pilot study worked on Task 12. This target student used a laptop to solve the task. When this student typed in B=(2.2), it worked but produced a slider B (Figure 4. 3). The student should have paid attention if B is a point or not, as it needs to appear on the grid, such as point A. This student then continued to type in n=Slide(3,100,1), and then it did not work; instead, the pop-up error showed up. Then, he decided to answer "No" as he could not follow Andi's command to make the object successful on Task 13.



```
A=(1,1) B=(2:2) n=Slide(3,100,1
Poligon(A,B,n)
A = (1, 1)
```

Perintah si Andi

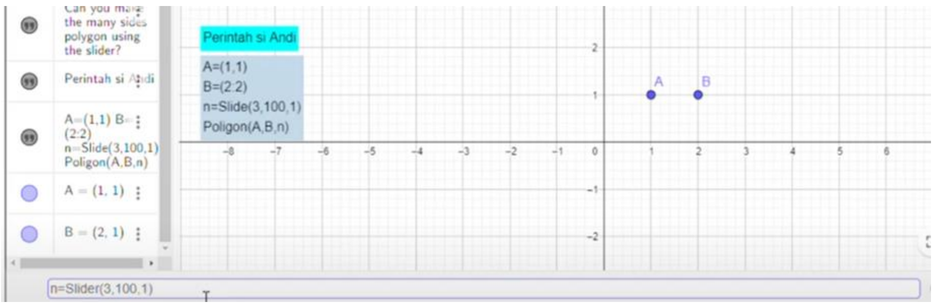
```
A=(1,1)
B=(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
```

```
B=(2:2)
```

Figure 4. 3 Typing in B=(2.2) as in Andi's command as a negotiated-style interruption

He deleted point B and then typed in the correct command B=(2,2). Thus, point B appeared on the grid. He continued to type in n=Slide(3,100,1), and again, the pop-up error appeared. He deleted the n command and then typed in B=(2,1) which was not the correct command.

He then typed in n=slide(3,100,10), and again, the pop-up error showed up due to the small letter s and the missing letter "r". He then edited the letter "s" to be "S", but again, the pop-up error showed up due to the missing "r". He deleted n and went to the previous task to help him. After seeing the command in the previous task, he continued with the correct n=Slider(3,100,1).



Can you make the many sides polygon using the slider?

Perintah si Andi

```
A=(1,1) B=(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
A = (1, 1)
B = (2, 1)
```

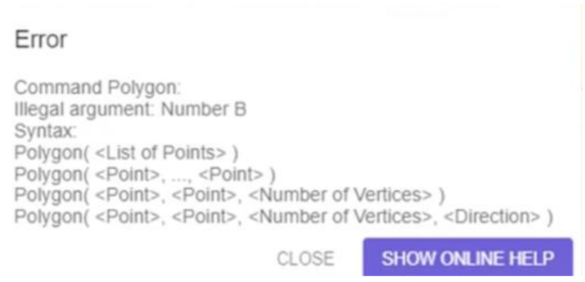
Perintah si Andi

```
A=(1,1)
B=(2:2)
n=Slide(3,100,1)
Poligon(A,B,n)
```

```
n=Slider(3,100,1)
```

Figure 4. 4 A student made a point B incorrectly

Surprisingly, he deleted the command 'n=Slider(3,100,1)' to edit B as B=(2,2). But he typed in B=(2,1) which was incorrect (Figure 4. 4). He continued to make n=Slider(3,100,1) successfully. Later, he revised B into a slider as he typed in B=(2:2). Next, this student made the polygon by typing in Poligon(A,B,n), and the pop-up error showed up. He realized his error and revised the letter i into y to become Polygon(A,B,n). After typing in the correct polygon command, the pop-up error showed up to notify that B was a problem (Figure 4. 5). B is considered an illegal argument: Number B. It should be a point, not a number

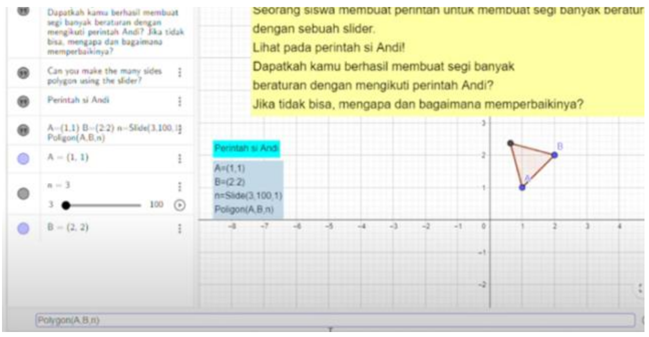


Error

```
Command Polygon:
Illegal argument: Number B
Syntax:
Polygon( <List of Points> )
Polygon( <Point>, ..., <Point> )
Polygon( <Point>, <Point>, <Number of Vertices> )
Polygon( <Point>, <Point>, <Number of Vertices>, <Direction> )
```

Figure 4. 5 An immediate-style interruption for the polygon to revise the B

So, he deleted the polygon command, then deleted the incorrect B and typed in B=(2,2). He typed in again Poligon(A,B,n) and a pop-up error showed up again and edited the command into the correct one ‘Polygon(A,B,n)’.



Dapatkah kamu berhasil membuat segi banyak beraturan dengan mengikuti perintah Andi? Jika tidak bisa, mengapa dan bagaimana memperbaikinya?

Can you make the many sides polygon using the slider?

Perintah si Andi

```
A=(1,1) B=(2,2) n=Slide(3,100,1)
Poligon(A,B,n)
A = (1, 1)
n = 3
B = (2, 2)
```

Seorang siswa membuat perintah untuk membuat segi banyak beratur dengan sebuah slider.
Lihat pada perintah si Andi!
Dapatkah kamu berhasil membuat segi banyak beraturan dengan mengikuti perintah Andi?
Jika tidak bisa, mengapa dan bagaimana memperbaikinya?

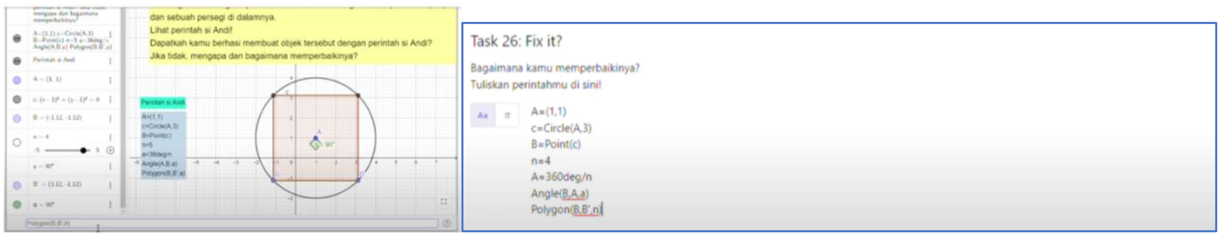
Perintah si Andi

```
A=(1,1)
B=(2,2)
n=Slide(3,100,1)
Poligon(A,B,n)
```

```
Polygon(A,B,n)
```

Figure 4. 6 A student successfully fixed the program

This student could make the commands run and produce the manipulatable polygon (a polygon that can be changed its number of sides by moving the slider). They could write down what they fixed. When a student engaged with task 23, he arrived at the correct commands and objects after a long process. This student spent more time accomplishing this task due to the order. The order here is the letter sequenced inside the command, for this case Angle(A,B,a). It confused this student to fix it until he checked the previous task to see the correct order. He then successfully made a square inside the circle (Figure 4. 7).



mengapa dan bagaimana memperbaikinya?

```
A=(1,1) c=Circle(A,3)
B=Point(c) n=5 a=36deg/n
Angle(A,B,a) Polygon(B,B',a)
```

Perintah si Andi

```
A = (1, 1)
c: (x - 1)² + (y - 1)² = 9
B = (-1.12, -1.12)
n = 4
a = 90°
B' = (3.12, -1.12)
a = 90°
```

dan sebuah persegi di dalamnya.
Lihat perintah si Andi!
Dapatkah kamu berhasi membuat objek tersebut dengan perintah si Andi?
Jika tidak, mengapa dan bagaimana memperbaikinya?

Perintah si Andi

```
A=(1,1)
c=Circle(A,3)
B=Point(c)
n=5
a=36deg/n
Angle(A,B,a)
Polygon(B,B',a)
```

```
Polygon(B,B',n)
```

Task 26: Fix it?

Bagaimana kamu memperbaikinya?
Tuliskan perintahmu di sini!

```
A=(1,1)
c=Circle(A,3)
B=Point(c)
n=4
A=360deg/n
Angle(B,A,a)
Polygon(B,B',n)
```

Figure 4. 7 Recognising where the errors are and the correct commands

5. Discussion

Papert argues that “learning consists of building up a set of materials and tools that one can handle and manipulate” [11, p. 173]. Thus, we let the students work individually on their own devices to complete all the tasks. We will discuss how our GeoGebra applet allowed students to handle and manipulate their programs and commands.

We let students have a learning sequence: learn to program, debug, and create the program. When students gained experience in debugging, they could recognise common errors (clichés) [19]. In line with this, most students could solve the debugging tasks as they get accustomed to how to fix the errors. This study witnessed students’ engagement in debugging as in [11].

Historically, programming to develop mathematical skills started in the 1970s with LOGO programming [20]. Some factors hinder LOGO's success and sustainability in education, as students and teachers have difficulty typing syntax in [21]. As we used a similar approach to LOGO, using syntax to create objects on GeoGebra, students experienced such difficulties. LOGO provided limited feedback to help learners find and reflect on their errors. Meanwhile, GeoGebra has a pop-up "debugging feature" that will appear if the syntax or command is incorrect. This has helped students in our pilot study to fix the errors.

The pop-up window in our study is related to types of interruption. Our error notifications belong to negotiated-style interruption and immediate-style interruption [17]. GeoGebra will not notify if A=(1:1) is an error immediately as it will be a slider, not a point A(1,1). Later it will become an immediate-style interruption when students cannot make a circle with the A, as it is not a point, and then a pop-up window notifies them that A needs to be fixed to be a point. Robertson et al [17] showed that the immediate-style interruption is less effective than the negotiated one. In our pilot study, students benefited from the pop-up window to fix the errors as they were new to this programming and debugging tasks.

The constructionist design could foster students' creativity and meaning-making in mathematical concepts when students create digital artefacts [22], [23]. In this case, students could utilize the mathematical concepts and observe the behaviour or characteristics. Our GeoGebra-based mathematics+CT tasks contain concepts of a Cartesian coordinate, a regular polygon, a circle, an angle, and an area of 2D shapes. It seems that the student in our pilot study could not connect the concept of Cartesian coordinate as a pair of x and y written in a standard form. It is relevant to a study by Ginat and Shmallo in [24], who found that in text-based programming, the student's lack of understanding of the underlying concepts could lead them to errors instead of syntax. It took our students some time to realize the universal form of a coordinate point as (x,y).

Acknowledgements The first author would like to thank the OeAD scholarship for the Ph.D. scholarship at Johannes Kepler University Linz.

References

[1] S. Bocconi *et al.*, "Developing Computational Thinking in compulsory education," in *Proceedings EdMedia 2016*, 2016, doi: 10.2791/792158.

[2] W. K. Ho and K. C. Ang, "Developing Computational Thinking through Coding," in *Proceedings of the 20th Asian Technology Conference in Mathematics*, 2015, pp. 73–87, [Online]. Available: http://atcm.mathandtech.org/.

[3] G. Gadanidis, R. Cendros, L. Floyd, and I. Namukasa, "Computational thinking in mathematics teacher education," *Contemp. Issues Technol. Teach. Educ.*, vol. 17, no. 4, pp. 458–477, 2017.

[4] J. M. Wing, "Computational thinking," *Communications of the ACM*. 2006, doi: 10.1145/1118178.1118215.

[5] V. Barr and C. Stephenson, "Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?," *ACM Inroads*, vol. 2, no. 1, 2011, doi: 10.1145/1929887.1929905.

[6] S. P. van Borkulo, C. Chytas, P. Drijvers, E. Barendsen, and J. Tolboom, "Computational Thinking in the Mathematics Classroom: Fostering Algorithmic Thinking and Generalization Skills Using Dynamic Mathematics Software," *ACM International Conference Proceeding*

Series. 2021, doi: 10.1145/3481312.3481319.

[7] H. Ye, B. Liang, O.-L. Ng, and C. S. Chai, "Integration of computational thinking in K-12 mathematics education: a systematic review on CT-based mathematics instruction and student learning," *Int. J. STEM Educ.*, vol. 10, no. 3, pp. 1–26, Jan. 2023, doi: 10.1186/s40594-023-00396-w.

[8] S. McKenney and T. C. Reeves, *Conducting Educational Design Research*. 2018.

[9] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educ. Res. Rev.*, vol. 22, pp. 142–158, 2017, doi: 10.1016/j.edurev.2017.09.003.

[10] A. King, "Mathematical Explorations: Finding Pi with Archimedes's Exhaustion Method," *Math. Teach. Middle Sch.*, vol. 19, no. 2, 2013, doi: 10.5951/mathteacmiddscho.19.2.0116.

[11] S. Papert, *Gears of My Childhood:Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books, 1980.

[12] C. Kynigos, "Constructionism: Theory of Learning or Theory of Design?," in *Selected Regular Lectures from the 12th International Congress on Mathematical Education*, 2015.

[13] I. Vourletsis, P. Politis, and I. Karasavvidis, "The Effect of a Computational Thinking Instructional Intervention on Students' Debugging Proficiency Level and Strategy Use," in *Research on E-Learning and ICT in Education*, 2021.

[14] J. M. Griffin, "Learning by taking apart: Deconstructing code by reading, tracing, and debugging," in *SIGITE 2016 - Proceedings of the 17th Annual Conference on Information Technology Education*, 2016, doi: 10.1145/2978192.2978231.

[15] Z. Liu, R. Zhi, A. Hicks, and T. Barnes, "Understanding problem solving behavior of 6–8 graders in a debugging game," *Comput. Sci. Educ.*, vol. 27, no. 1, 2017, doi: 10.1080/08993408.2017.1308651.

[16] M. Resnick *et al.*, "Scratch: Programming for all," *Commun. ACM*, vol. 52, no. 11, 2009, doi: 10.1145/1592761.1592779.

[17] T. J. Robertson *et al.*, "Impact of interruption style on end-user debugging," in *Conference on Human Factors in Computing Systems - Proceedings*, 2004, doi: 10.1145/985692.985729.

[18] K. Krippendor, *Content Analysis An Introduction to Its Methodology Second Edition*. 2004.

[19] M. Ducasse and A. M. Emde, "Review of Automated Debugging Systems: Knowledge, Strategies and Techniques," in *Proceedings - International Conference on Software Engineering*, 1988, doi: 10.1109/icse.1988.93698.

[20] S. Papert, "Teaching Children Thinking," *Program. Learn. Educ. Technol.*, vol. 9, no. 5, 1972, doi: 10.1080/1355800720090503.

[21] M. Resnick, "Reviving Papert's Dream," *Educ. Technol.*, vol. 52, no. 4, 2012.

[22] L. Healy and C. Kynigos, "Charting the microworld territory over time: Design and construction in mathematics education," *ZDM - Int. J. Math. Educ.*, vol. 42, no. 1, 2010, doi: 10.1007/s11858-009-0193-5.

[23] M. Karavakou and C. Kynigos, "Designing periodic logos: a programming approach to understand trigonometric functions," in *10th ERME Topic Conference (ETC10) Mathematics Education in the Digital Age (MEDA)*, 2020, pp. 223–230.

[24] D. Ginat and R. Shmallo, "Constructive use of errors in teaching CS1," in *SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education*, 2013, doi: 10.1145/2445196.2445300.