SCHOOL MATHEMATICS NEWSLETTER

ISBN 978-988-8581-17-7

政府物流服務署印

學校數學通訊 第二十六期 教育局

SMN

SCHOOL MATHEMATICS NEWSLETTER

ISSUE 26

Published by
Mathematics Education Section, Curriculum Development Institute,
Education Bureau, Government of the Hong Kong Special Administrative Region.
香港特別行政區政府教育局課程發展處數學教育組出版

ISBN 978-988-8581-17-7

Foreword

Welcome to the 26th issue of the School Mathematics Newsletter (SMN).

The School Mathematics Newsletter (SMN) is for mathematics teachers. It serves as a channel of communication on mathematics education in Hong Kong. This issue involves various articles written by academics, curriculum officers and teachers in different areas, including academic insightful views on mathematics education, suggestions of effective strategies in learning and teaching of mathematics about STEAM education, computationally enhanced mathematics education, online interactive mathematics lessons and articles related to application of mathematics, algorithms and artificial intelligence, etc.

In the existing education system, mathematics teachers are faced with the tremendous challenge of teaching students of very different abilities, motivations and aspirations. To meet this challenge, mathematics teachers need to equip themselves with necessary mathematical skills and teaching strategies to cope with different teaching situations. We do hope that all readers will find the content of this issue informative and stimulating.

The Editorial Board of SMN wishes to express again its gratitude to all contributors, and also to the fellow colleagues of the Mathematics Education Section who have made great efforts in publishing the SMN Issue 26.

SMN provides an open forum for mathematics teachers and professionals to express their views on the learning and teaching of mathematics. We welcome contributions in the form of articles on all aspects of mathematics education. However, the views expressed in the articles in the SMN are not necessarily those of the Education Bureau. Please send all correspondence to:

The Editor, School Mathematics Newsletter,
Mathematics Education Section
Curriculum Development Institute
Room 403, Kowloon Government Offices
405 Nathan Road
Yau Ma Tei, Kowloon
email: math@edb.gov.hk

1. Integration of Programming, Problem Solving and Recreational Mathematics for a Computationally Enhanced Mathematics Education

Oi-Lam Ng[1], Alvin Chan[2], Tom Ho[3], Don Tsoi[4], Angel Liu[5], Marco Law[1], Biyao Liang[6]

[1]The Chinese University of Hong Kong, [2]Good Hope School, [3]St. Paul's College, [4]Tsung Tsin College, [5]CCC Ming Yin College, [6]University of Hong Kong

Introduction

Computational thinking (CT) is a powerful cognitive tool for solving problems, designing systems, and understanding human behavior by drawing on concepts fundamental to computer science (Wing, 2006). It is helpful not only in maintaining competence in a technological society but also in supporting development in higher-order skills such as critical thinking, analysis, and scientific inquiry for the Science, Technology, Engineering, Mathematics (STEM) disciplines. Surrounding this, calls for incorporating CT into mathematics education are rapidly increasing. However, the mere presence of computers in the classroom does not ensure their effective use or quality education. Structural changes in the curriculum and instructions are needed to take full advantage of using CT to teach mathematics and problem solving.

Our work builds on the first author's previously developed conception of "learning as Making" (Ng & Chan, 2019； Ng & Ferrara 2020) to envision a computationally enhanced mathematics education. In particular, we were interested in digital making as a form of artefact construction through screen-based programming to support mathematical problem-solving in fundamental ways. Digital Making involves students' active creation of digital artefacts through block-based programming. It promotes 21st century learning skills and transforms mathematical problem-solving from merely using formulae and performing arithmetic calculations procedurally (Ng & Cui, 2021； Weng et al., 2022a). Rather, CT concepts and practices such as sequences, variable-naming, abstraction, algorithmic thinking, decomposing, and iterating are highlighted during problem-based digital making activities, through which scientific inquiry, mathematical thinking, and engineering design can also be exhibited as integrated STEM learning.

In the past two years (2020-2022), the research team as led by the first author has made significant efforts and progress in designing problem-based digital making tasks with content appropriate to senior primary and junior secondary mathematics curricula in Hong Kong. These lessons have been implemented in one public education institution (i.e., The Hong Kong

Academy of Gifted Education) and two secondary schools totaling over 120 students. In this paper, we share some detailed lesson plans in which we integrated programming, problem solving and recreational mathematics for a computationally enhanced mathematics education in a class of Secondary 1 to Secondary 3 students. Our goal is to reflect on the success and challenges in the implementation of the lessons, as well as to provide pedagogical suggestions for educators for future implementations of integrating CT in mathematics learning.

We begin with describing our lesson design and rationale in terms of teaching CT in the context of recreational mathematics and problem solving, followed by providing our observations on what students did and learned during the lesson implementation, drawing on some student solutions and projects as examples. We end with discussing some challenges experienced gathered through the year as well as pedagogical suggestions for better integrating the elements of programming, problem solving and (recreational) mathematics for integrated STEM learning in future practices.

Lesson Design and Rationale

In this article, we describe our implementation of five lessons with the following themes: Introduction, Numbers & Algebra, Data Handling, Geometry, and Finale. Throughout the five lessons, we chose to adopt Desmos Classroom to collect students' responses and to monitor their progress whenever necessary. In each lesson, one of the authors would be the main speaker and the rest would be teaching assistants, in the hopes of gaining insightful observation from multiple perspectives. To achieve "low floor, high ceiling" learning, we also included extension problems in the design to achieve differentiated instruction.

Tryouts

Since not all students have programming background, our first lesson aimed at equipping students with the necessary skills. We started by introducing some mini-games to demonstrate the function of each command, such as "if … then …", the use of "variables", and the concept of "events". For example, students will learn how to move the sprite (i.e., characters in the program) using an "if... then..." statement with arrow keys, and command the cat to fire balls at the bat in order to gain points through the use of "variables".

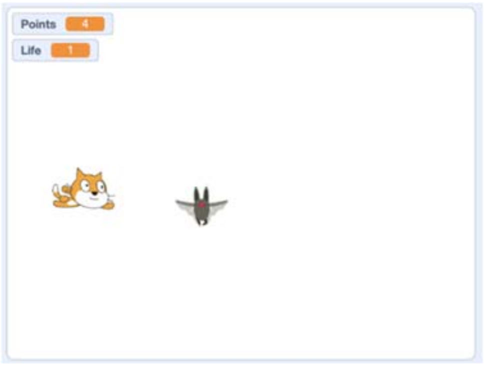
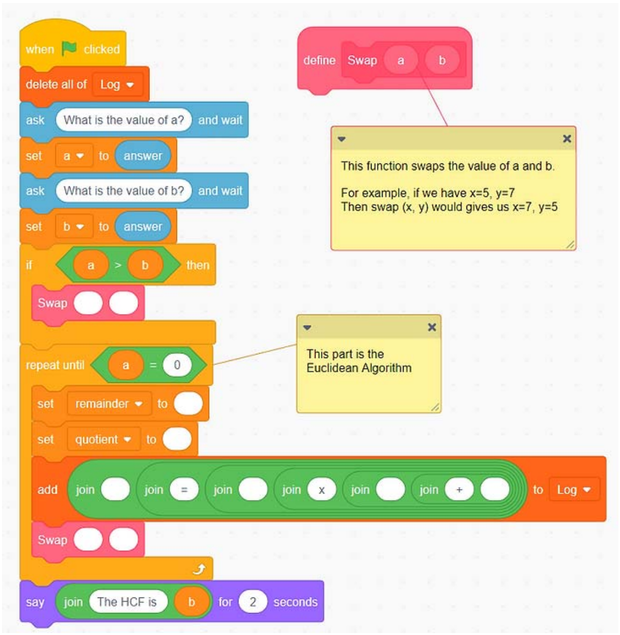


Figure 1: Shooting Game

After getting a sense of Scratch programming, the students customized their shooting games by adding personal elements to the games. Through this, they demonstrated the open-ended nature of digital making in offering opportunities for the students to apply their programming skills in different ways and creatively (see also Weng et al., 2022b).

Numbers and Algebra

The next lesson introduced three activities: Prime Number Tester, Euclidean Algorithm, and the Count from 21 Game. The first one tests whether an input integer is a prime number； and the second one utilizes an algorithm to find the greatest common divider (GCD). The concept of Arrays (or Lists) was also established in this lesson. The overall teaching strategy for the first two activities was to guide students in understanding the program specifications, decompose the problem into smaller steps, and be able to read codes that were already given to them in the program. We also designed opportunities for learning by leaving some code blocks blank and provided suitable guidance when students tried to fill in the blanks on their own (see Figure 2).



```
when 🏴 clicked
delete all of Log ▾
ask (What is the value of a?) and wait
set a ▾ to (answer)
ask (What is the value of b?) and wait
set b ▾ to (answer)
if < (a) > (b) > then
    Swap ( ) ( )
repeat until < (a) = (0) >
    set remainder ▾ to ( )
    set quotient ▾ to ( )
    add (join ( ) (join (=) (join ( ) (join (x) (join ( ) (join (+) ( ))))))) to Log ▾
    Swap ( ) ( )
say (join (The HCF is) (b)) for (2) seconds
```

```
define Swap (a) (b)
```

This function swaps the value of a and b.

For example, if we have x=5, y=7
Then swap (x, y) would gives us x=7, y=5

This part is the Euclidean Algorithm

Figure 2: An illustration of missing parts of a program

For the third activity, we decided to adjust our teaching strategy. Again, using game-based learning, we guided the students to create a program related to a game called Count from 21. Count from 21 is a game where two players take turns to cross out one or two numbers (starting from 21, see Figure 3), and whoever crosses '1' wins the game. In this task, the students were prompted to design programs in which a user plays the game with it, but it's always the computer who wins.

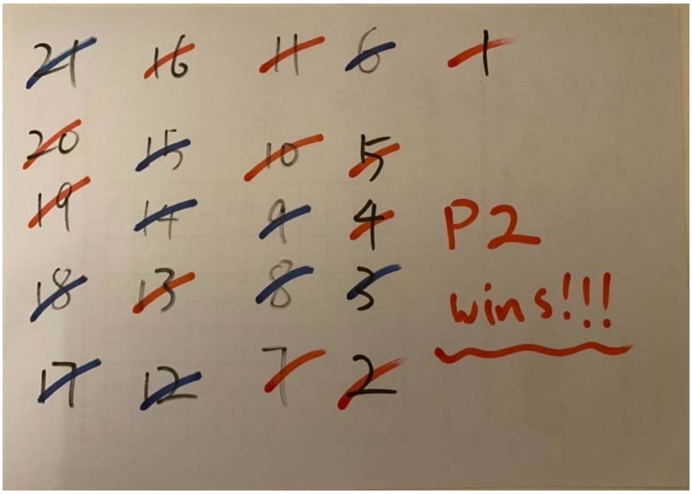


Figure 3: Count from 21

To understand the game thoroughly, we let students play with their groupmates and come up with a winning strategy on their own. In doing so, they would discover the mathematics behind the winning strategy, which was always to call a number which is a multiple of three. Then, they would have to complete the master program on their own with a few specifications. Such design is very engaging because the students first physically experienced the game played and then were challenged to devise a way that would always win. Overall, the three tasks called upon the students to use both their knowledge and discovery about numbers, as well as their developing programming skills to solve problems.

Data Handling

We then turned our focus on simulation in this lesson. Specifically, we chose three very well-known games to address the concept of experimental probability, namely Rock-Paper-Scissors, Dice Rolling, as well as Dart Throwing.

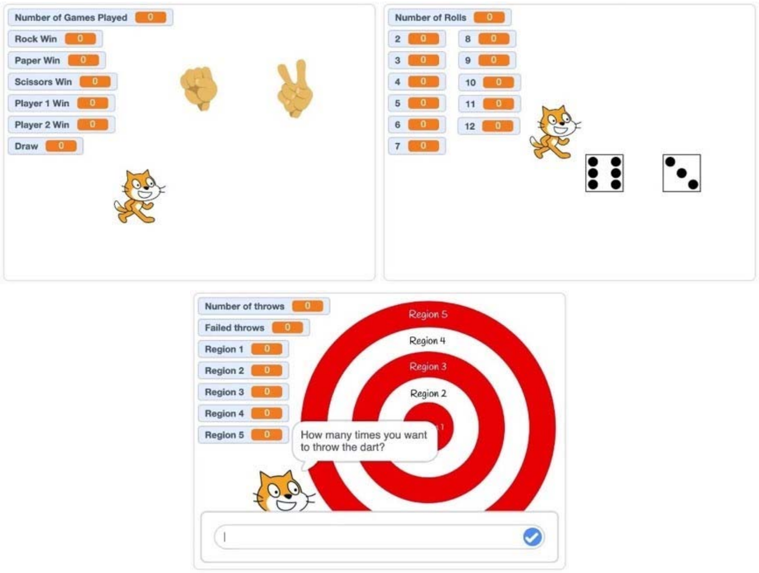


Figure 4: Snapshots of Simulation Games

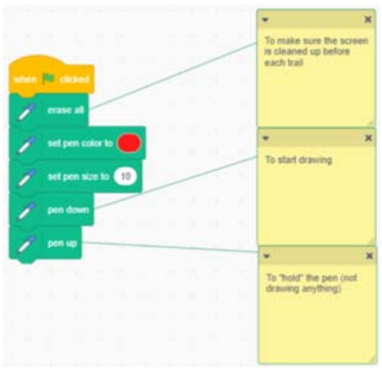
To reinforce their programming and problem-solving skills, students discussed with their partners and designed the simulations based on the requirements given. In addition, we asked some extended questions to enhance their statistical and probability-related conceptions, such as the highest occurrence of events, expected values, etc. The following are some examples of questions used:

- 'If we play the rock-paper-scissors game 1000 times instead of 100 times, what do you expect to observe?', and
- 'If we roll 6 dice instead of 2 dice, describe what you think the distribution graph will look like?'.

Through these questions, students were encouraged to make further observations and uncover the statistical characteristics such as the distribution behind the events. Therefore, we intended not only to use programming for solely generating and recording the results, but we also aimed at developing students' statistical sense and thinking throughout the lesson. Ultimately, the students gained an understanding about the advantages of using the "random" function for simulating random events in programming contexts versus hands-on simulations in real-life. They became appreciative to computational means of solving problems related to random events.

Geometry

This lesson consists of three activities: Rotational Symmetry, Tessellation and Sierpiński Triangle. During the first two activities, geometric properties were introduced in a traditional sense and then applied in a programming manner. To begin, students learned to draw figures using a "Pen" extension in Scratch program (Figure 5). It was expected that students could draw an assortment of shapes with different repetition and number of sides using the "repeat" and "change by 1" codes. Indeed, the students were quick to learn how to draw in Scratch, while visualizing different geometric figures by the way (i.e. distance and angle of turn) the "Pen" moved in their designed program. As such, the students interpreted the meaning of geometrical concepts from the program, such as which kinds of regular polygon can tessellate and which cannot (Figure 6), and the degree of rotational symmetry of a figure.



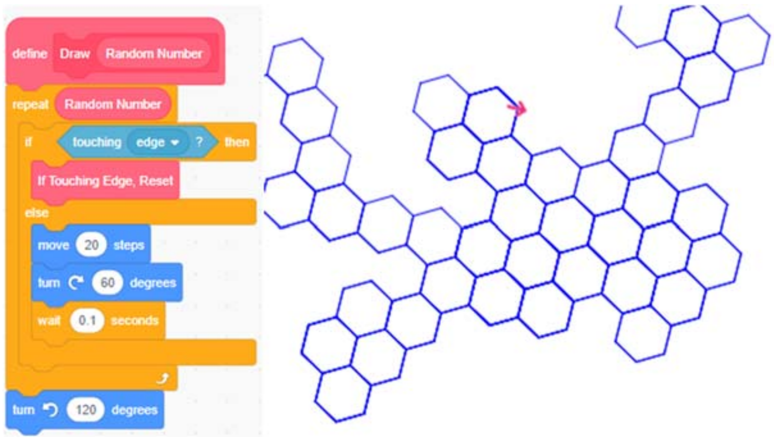
```
when 🏴 clicked
erase all
set pen color to 🟥
set pen size to 10
pen down
pen up
```

To make sure the screen is cleaned up before each trail

To start drawing

To "hold" the pen (not drawing anything)

Figure 5: A snapshot of Pen function



```
define Draw (Random Number)
repeat (Random Number)
    if <touching (edge ▾) ?> then
        If Touching Edge, Reset
    else
        move (20) steps
        turn ⟳ (60) degrees
        wait (0.1) seconds
turn ⟲ (120) degrees
```

Figure 6: Program for Tessellation

The third program serves to develop an additional knowledge beyond the school curriculum for students, namely the idea of recursion, in the sense of "a function that calls itself". This is a challenging task for students to apply what they have learnt in this and previous lessons in order to create a program with a "myBlock" and "nested loop". Given that this task was challenging, the students tested and debugged their programs with their partners. Through solving this task, we intended for the students to make sense of "myBlock" as a function mathematically in the sense of "a machine with an input and output" and to advance their programming knowledge of "nested loop" by putting one "repeat" within another.

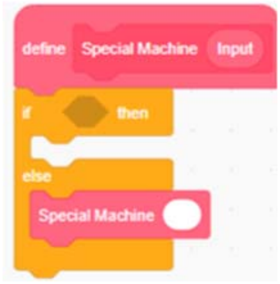


Figure 7: An example of recursion in Scratch

The Finale

For the final lesson, we hope to (1) consolidate the students' CT skills, and (2) immerse them in the real-world mathematical problem-solving in a programming context. Hence, we designed and adopted a more challenging task, the Water Jug Problem, with the following set up.

Suppose there are 3 jugs which do not have any measurement indicators. It is given that the volume of Jug A and B are m and n respectively with $gcd\ (m, n) = 1$, and that of Jug C is $m+n$. Initially only Jug C is filled with water. Suppose a desired volume is input by the user (an integer from 1 to $m+n$), students were required to create a program which informed the user how to obtain the desired volume by pouring the water among the three jugs.

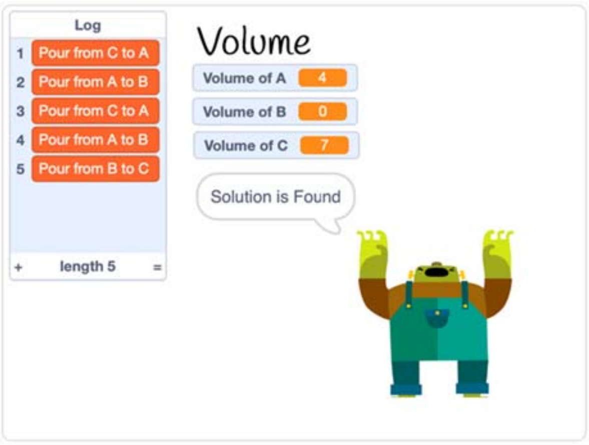


Figure 8: A snapshot of a sample program

We first let students formulate the algorithms on their own by playing with our "trial program", in which they could easily observe the behaviors of the volumes. This goal-oriented problem in which students were first shown the digital artefact, or the end product, was a pedagogical design very useful for teaching programming, for the students would be well informed what their programs should behave in the course of programming. Then, upon some mini-presentation, students started to work on the program. Extension questions were adopted to guide students in revising their program as well as their mathematical and computational thinking.

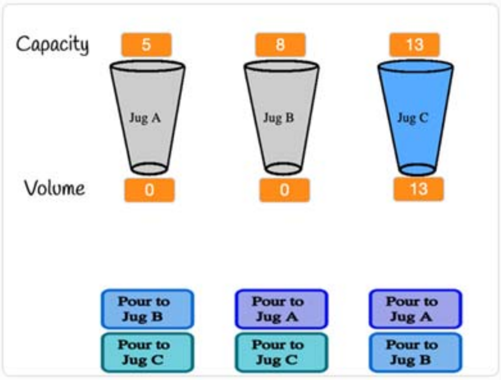


Figure 9: A snapshot on our "trial program"

Observation and Students work

Based on students' performance in the five lessons and their classwork, we made the following observations. First, as Scratch programming enables rapid generation of numerical and geometric examples, it helps students to visualize mathematical relations and patterns. For example, in non-programming environments, students often find it difficult to hypothesize the probability distribution graph of throwing two dice at one time and finding its sum. With the help of Scratch, students were able to observe the results of 100 or more throws, thus uncovering that the event of obtaining the sum of "7" has the highest occurrence and hence highest probability among all possible events. These activities would not be possible in conventional mathematics classrooms. In other words, Scratch facilitates generation of mathematical examples, thus enabling observation of mathematical relations and patterns.

Second, the use of Scratch also enhances students' problem-solving skills. As mentioned, CT enables one to understand and solve problems by drawing on concepts that are fundamental to computer science (Wing, 2006). In the Water Jug problem, for instance, students were given opportunities to understand and solve the problem from the perspective of programming. Aligned with previous studies, they performed CT as a mode of problem solving while applying various Scratch commands, such as variables, my blocks, to express their solutions in the programming environment (Ng & Cui, 2021), and they developed positive problem-solving disposition such as willingness to persist in solving the problem (Ng et al., 2021). As a result, students develop better understanding of the logical relationship between various parts of the solution and various mathematical concepts, such as the concept of function.

Third, we observed that the students were highly engaged and motivated compared to conventional classrooms. With appropriate lesson and task design, the students experienced hands-on, personal, and goal-oriented learning experiences in the "learning as making" environment (Ng & Chan, 2019; Ng & Ferrara 2020). This resulted in the phenomenon that almost all students were actively engaged throughout the whole program. Meanwhile, students were able to understand and solve more difficult and open-ended mathematics problems with the use of Scratch than the ones they were familiar in a traditional mathematics classroom. Therefore, our problem-based digital making instructional design enabled teachers to introduce mathematical problems with multiples entry and exit points, into the mathematics classroom. With such hands-on experiences, we observe that students were more willing and able to engage in each lesson, as well as participate in the activities collaboratively.

However, we also note that some students would get confused by the programming syntax easily. In those tasks where multiple loops, conditionals (e.g., “repeat”, “if… then… else…”, etc.), variables or multiple sprites were needed (e.g., Euclidean Algorithm and Rock-Paper-Scissors), students might not be well equipped to solve the problem by thinking of how it can be decomposed into smaller steps and by designing an algorithm that executes the steps, hence resulting in errors. For example, some students programmed their solution to allow an input of 0 for the task of Count from 21, which is not acceptable since the game only accepts 1 or 2 from each player. In spite of that, it should be noted that once guidance was provided by teaching assistants, such issues were resolved, and students were becoming more and more confident in applying CT to participate in the activities, as evidenced by their responses in the extension questions.

Reflection and Future Directions

In this era of technological advancement, there has been a call for nurturing students' capabilities of solving complex real-world problems. To achieve so, students are expected to develop higher-order skills such as critical thinking, analysis, and scientific inquiry for the STEM disciplines. In particular, CT enables one to understand and solve problems from the perspectives of computer science (Wing, 2006). In other words, students equipped with CT can implement information technology into problem-solving with higher efficiency and precision when compared to conventional mathematics learning and teaching. Furthermore, programming enables students to observe patterns and simulations that cannot be done in conventional mathematics classrooms, thus creating personal and mathematical meanings to them. As such, we believe these lessons could serve as an alternative pedagogical approach in the teaching of secondary mathematics in the future.

We also see that the current curriculum for STEM in Hong Kong are not yet mature such that most teachers find teaching STEM difficult. Prior research evidenced that the barriers of developing STEM are availability of corresponding lesson designs, the support and resources, the tasks and processes of implementing STEM education, and teachers' professional development (Geng et al., 2019; Lin et al., 2021). To further improve the design and practice of STEM curriculum in Hong Kong, it is essential to explore lesson designs and instructional plans enabling teachers' effective integration of elements from the STEM disciplines into mathematics classrooms. This includes lessons that effectively incorporate the interplay between mathematical and computational thinking (Cui & Ng, 2021). However, not all mathematics teachers had received relevant training in related fields; let alone having a background on programming. To this end, we are encouraged by the fact that despite some of the co-authors had little background related to programming initially, all of us were able to learn, develop, and implement engaging lessons for this project. Therefore, we believe that it is worthwhile for in-service teachers and panels to consider the possibility of integrating some elements of CT into mathematics classrooms, so that students can experience the importance of CT and information technology in the process of learning mathematics and problem-solving, as well as to appreciate the importance of STEM education.

References

[1] Cui, Z., & Ng, O. (2021). The interplay between mathematical and computational thinking in primary students' mathematical problem-solving within a programming environment. Journal of Educational Computing Research, 59(5), 988–1012.

[2] Geng, J., Jong, M. S. Y., & Chai, C. S. (2019). Hong Kong teachers' self-efficacy and concerns about STEM education. The Asia-Pacific Education Researcher, 28(1), 35-45.

[3] Lin, P. Y., Chai, C. S., & Jong, M. S. Y. (2021). A study of disposition, engagement, efficacy, and vitality of teachers in designing science, technology, engineering, and mathematics education. Frontiers in Psychology, 12, 661631.

[4] Ng, O., & Chan, T. (2019). Learning as Making: Using 3D computer-aided design to enhance the learning of shapes and space in STEM-integrated ways. British Journal of Educational Technology, 50(1), 294-308.

[5] Ng, O., Cui, Z. (2021). Examining primary students' mathematical problem-solving in a programming context: Toward a computationally enhanced mathematics education. ZDM Mathematics Education, 53, 847–860.

[6] Ng, O., & Ferrara, F. (2020). Towards a materialist vision of 'learning as Making': The case of 3D Printing Pens in school mathematics. International Journal of Science and Mathematics Education, 18, 925–944.

[7] Ng, O., Liu, M, & Cui, Z. (2021). Students' in-moment challenges and developing maker perspectives during problem-based digital making. Journal of Research on Technology in Education.

[8] Weng, X., Cui, Z., Ng, O., Jong, M., & Chiu, T. K. F. (2022a). Characterizing students' 4C skill development during problem-based digital making. Journal of Science Education and Technology, 31(3), 372-385.

[9] Weng, X., Ng, O., Cui, Z., & Leung, S. (2022b). Creativity development with problem-based digital making and block-based programming for Science, Technology, Engineering, Arts, Mathematics learning in middle school contexts. Journal of Educational Computing Research.

[10] Wing, J. M. (2006). Computational thinking. Communications of the ACM, 49(3), 33-35.