Journal of Theoretical and Applied Information Technology
31st December 2020. Vol.98. No 24

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

UNDERSTANDING STUDENTS' PERCEPTIONS AND MOTIVATIONS TOWARDS THE LEARNING OF PROGRAMMING IN MALAYSIAN HIGH SCHOOLS

[1]RODZIAH LATIH, [2]AHGILAN PEREMOL, [3]NORLEYZA JAILANI

[1,3] Faculty of Information Science and Technology, University Kebangsaan Malaysia, 43600 Bangi, Selangor, Malaysia

[2]Information Management Unit, Ministry of Education, Putrajaya, Malaysia.

E-mail: [1]rodziah.latih@ukm.edu.my, [2]ahgilan@gmail.com, [3]njailani@ukm.edu.my

ABSTRACT

Computer programming as a taught subject was added to the Malaysian National School Curriculum in stages starting from 2017. This is seen as a positive step in light of the nation's future challenges, especially with the advent of the Fourth Industrial Revolution. However, a wealth of research has indicated that programming's teaching and learning are wrought with challenges and pedagogical issues. It is thus imperative that studies are done to investigate issues regarding the acceptance and motivation in learning programming among local school children. With this objective in mind, a study was done in several schools in Selangor, Malaysia. A total of 166 form-four students were surveyed to uncover their perceptions of the materials and approaches used in the teaching of programming and their motivations in learning programming. According to the results, the materials and teaching approach were important factors to motivate students to like the subject and make them perform better.

Keywords: *Coding, Computational Thinking, Algorithmic Thinking, Computer Science Curriculum, Turtle Graphic*

1. INTRODUCTION

Programming skill has been identified as the required skill for the current job market [1]. It is also an essential job skill for a future job because many businesses rely on computer apps and systems. Learning to program lets students get a better understanding of the software, which is an essential part of the computer system and current technology devices. Programming is a process of writing, testing, debugging/ troubleshooting, and maintaining the source code of computer programs [2]. Programming is, in fact, a much broader topic than pronounced by definition. Programming exposes students to computational thinking, which involves problem-solving using a computer [3]. Thus, programming is an ideal way of developing computational thinking [4], and computational thinking may be applied to various kinds of problems that do not directly involve programming [5].

Computational thinking will let students articulate a problem and think logically. It can be used not only for mathematically well-defined problems of which solutions are completely analyzable; nevertheless, real-world problems whose answers might be in the form of large, complex software systems [6]. Students competent in computational thinking are knowingly better prepared for the daily tasks and the professional work that anticipates them in their future [7]. Thus, computational thinking should be considered as an attitude and skill for everyone and not just computer scientists [8].

Accordingly, many countries have introduced computer programming courses in schools. In the UK, the new computing curriculum was introduced in 2014 and incorporated Computer Science teaching as compulsory from ages 5-16 [9]. The curriculum was programmed in four stages throughout their formal K-12 education. For each stage, students are expected to develop aspects of computational thinking skills progressively. For example, in Key Stage 1 (age 5–7 years old), the students create and reason about the simple program. At this stage, the objective is to develop

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

deeper computational thinking and problem-solving skills, and the exercises take place without program a physical computer. In Key Stage 2 (age 7–11 years old), they will create and debug more complicated programs with specific goals and get to grips with basic programming concepts like variables and control structure. In Key Stage 3 (age 11–14 years old), the students are expected to learn two programming languages, at least one of which is a textual programming language where they have to create their programs. While the requirements of Key Stage 2 can be fully satisfied in non-textual programming like Scratch, the Key Stage 3 syllabus deliberately ensures that students move to full-textual programming. At Key Stage 4 (age 14–16), the students have an opportunity to study aspects of Information Technology and Computer Science at sufficient depth to allow them to progress to higher levels of study or a professional career.

Japan also has a plan to make programming compulsory for all primary school (age 6 – 12 years old) by 2020, followed by a junior high school (age 12 – 15 years old) in 2021 and high school (age 15 – 18 years old) by 2022 [10]. In primary school, the students will learn logical thinking through programming experiences. Since there is no subject to teach computing, the students will learn programming in other disciplines like arithmetic and science. In junior high school, the students will learn programming through the subject Technology, of which they will discover two types of programming, “Measurement and Control” and “Network Communication." In high school, the students will learn to program in two elective courses of subject “Informatics”; Information I and Information II.

Singapore introduced a computing curriculum to develop computational thinking and programming skills from pre-school to tertiary education level in 2014 [11]. Unlike countries like England and Korea, Singapore does not include computing or computational thinking as compulsory education. Instead, Singapore’s approach provides opportunities for students to develop their interests in programming and computing skills through touchpoint activities at various ages. At the pre-school level, the Playmaker Program was introduced where electronic, robotic, or programmable toys are used to engage young children in play while developing computational thinking skills such as algorithmic thinking. In primary school, the Code for Fun Enrichment Program was introduced to expose the students to computational thinking concepts and programming; and develop a workforce equipped with basic programming and computational thinking skills. The students learn to program using a visual programming language, such as Scratch, combined with a robotic kit such as the MoWay or microcontrollers such as the micro:bit. For secondary school, students are taught to program using Python programming language. The students will develop computational thinking and programming skills to create solutions with technology to solve problems.

Malaysia had also introduced programming as a subject at the school level in 2017. The curriculum has been designed to integrate the subject, starting from primary school until upper secondary school. The objective is to equip students with computational thinking and problem-solving skills. Subsequently, this study investigates the perceptions of Malaysian lower secondary students towards learning computer programming and what motivates them to learn the subject. We also introduce an engaging visual approach to learning basic programming as an alternative to the current textual approach. The aim is to investigate whether the approach can help them understand the topics and motivate them to learn computer programming.

Therefore, the rest of the paper is structured as follows: Section 2 explains the research background, while Section 3 highlights the research methods applied in this study. This is followed by sections 4 and 5, which respectively describe the results obtained and the discussion. Finally, section 6 concludes this paper.

2. RESEARCH BACKGROUND

2.1 Malaysia Education System

In Malaysia, education is divided into four stages: pre-school education (age 4-6 years old), primary school (age 7 – 12 years old), secondary school (age 13 – 17 years old), and tertiary education (college or university) (Fig. 1). The primary school takes six years (year-1 (Y1) till year-6 (Y6)), and for secondary school, it takes five years (form-one (F1) till form-five (F5)). Only

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

primary school is compulsory, where else are optional. The secondary school stage is divided into two levels; lower secondary school (F1-F3) and upper secondary school (F4-F5). After each stage, there is a national examination named Primary School Achievement Test (UPSR), Form Three Assessment (PT3), and Malaysian Education Certificate (SPM). After the PT3 examination, students are streamed according to their PT3 results and interest. They typically attend one of the three types of school; academic (art or science), technical and vocational, and religious. At the end of upper-secondary school, students from all streams take the SPM examination. Those who want to go for tertiary education can choose either to go to post-secondary school (form-six (F6)), Matriculation, or Diploma at any institution, like polytechnic. After form-six (a two-year program), they have to sit for another national examination called the Malaysian Higher Education Certificate (STPM). Generally, the admission requirement for all Malaysian universities is either STPM, matriculation, or Diploma. Figure 1 summarizes the stages in Malaysia Education System.

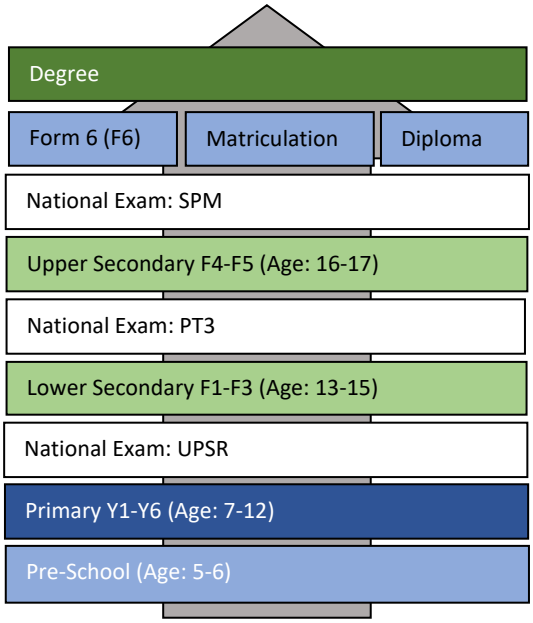

Figure 1: Malaysia Education System

In 2017, Malaysia officially introduced computer programming subjects in the national school curriculum to equip students with logical thinking and problem-solving skills. At primary school (Y4-Y6), programming is taught through the Design and Technology subject. Students learn to create algorithms (pseudocode and flow chart) and develop simple programs. The students are also learned to program hardware such as Arduino and Micro-Bit, which is an introduction to Robotics and Artificial Intelligence. Computer programming is also taught to lower-secondary school students (F1-F3) through two different subjects (the school can choose either one); Basic of Computer Science subject, and Design and Technology subject.

Further, for upper-secondary school students (F4-F5), computer programming is taught via Computer Science subject. At the lower-secondary school level (F1-F3), students are exposed to simple programming using Scratch, while upper-secondary school (F4-F5) students are using Microsoft Visual Basic, Java, PHP, Phyton, and HTML (also the school can choose which appropriate). The objective is to equip the students with a good foundation in computational thinking, problem-solving skill, and programming language for future digital economy jobs. In the post-secondary level (F6), programming is taught through Information and Communication Technology subjects where they will learn C Programming. Table I in the following shows the subjects taught for each level from primary school to post-secondary level.

Table 1: Computer Science Subjects

Level	Subject
Post-Secondary (F6)	Information and Communication Technology
Upper-Secondary (F4-F5)	Computer Science
Lower-Secondary (F1-F3)	Basic of Computer Science
	Design & Technology
Primary School (Y4-Y6)	Design & Technology

2.2 The Challenges of Teaching Computer Programming at School Level

The main aim of teaching programming is to teach how to program. The fundamental of programming is about two things; solve the problem, and create a program as a solution [2]. In the first case, the programmer analyses the problem, produces an algorithmic solution, and then

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

translate this algorithm into a program code. The second case, which is to create a program, involves two knowledge; program generation and program comprehension. To generate a program, students need to know the type of data, the process, and how to construct the command. As for program comprehension, the students should understand how their program works. Another aspect, similarly important, is the semantics of a program, also referred to as the program's meaning. A semantically correct program is a program that performs the required task. Programs written with different syntax can perform the same semantic task.

Computer programming is challenging to both students and teachers as it is required to fluent the programming language and structure the overall logical flow of instructions. The concepts in computer programming are also abstracts and challenging to learn. Thus, it is recommended for introductory programming to teach problem-solving and algorithm development before the language [12]. The algorithm is a detailed step-by-step instruction for solving a problem or completing a task. Children should be teaching to think algorithmic because it is crucial for other subjects as well as mathematics and science.

Masura et al. [13] identified six significant challenges in teaching and learning computer programming; suitable materials, teaching approaches, learning approaches, problem-solving skills, time management, and self-confidence. The students who excel in programming have worked very hard; they do many exercises, discuss with teachers and peers, and find other resources such as from the Internet [13]. Bubica and Boljat [14] also highlighted factors that influence the success of students in programming such as mathematical knowledge, spatial map sketching, sense of comfort, attribution of success, learning style, the choice of the first programming language, students' behavior while programming, learning strategies, gender, and previous programming experience, the total number of years spend in programming, a number of introduced programming languages, using viable mental models, solving problem abilities, greater number spent in playing computer games, exercising languages skills, English as a native language, students interest, and their course expectations, and systemizing Quotient (SQ) - empathy quotient (EQ) value.

The method of teaching in a classroom can be the reason for students' low performance. Using an outdated way of teaching to the current generation is not an option. Students born and grown up with interactive media and smart mobile devices do not find programs based on examples of the form 'Hello, World' motivating. They are more interested in visual programming tools like Scratch [15], Alice [16], Pencil Code [17], Kojo [18], and a real robot [19], [20]. However, using programming tools in the classroom does not impact the students' problem-solving skills; nevertheless, it improves their self-confidence in problem-solving skill [21] and their perception toward programming [22].

Lack of motivation among the students in learning computer programming may contribute to some known problems in computer science education, such as lower achievement, programming incompetence, low level of interest, and negative perceptions towards computer science. Nevertheless, using the right programming language, course content, and teaching approach can affect students' self-efficacy [23]. On the other hand, self-efficacy positively impacts intrinsic motivation, and this intrinsic motivation can engage students in learning activities [24]. Intrinsic aspects focus on individuals rather than the environmental setting. It generally includes individual attitude and expectation, goals, and emotions. Studies show that students who have intrinsic motivation disclosed excellent performance and perceived high programming skills compare with other types of motivation [22], [25], [26].

2.3 Java Turtle Graphics

Turtle Graphics is a programming language learning approach for children introduced by Wally Feurzig and Seymour Papert in the late 1960s [27]. Initially, Turtle Graphics was a small robot that looks like a turtle and could hold like a pen for drawing. It is a learning tool used to learn the Logo programming language. At that time, Logo was well accepted because of its simplicity and features for quickly show the graphical results. Papert then developed a graphic library for the Logo programming language based on the original Turtle Graphics concept.

The following Figure 2 shows an example of a program written in Java using the Turtle

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Graphics approach. Turtle Graphics is a vector graph with three attributes: location, direction, and pen. The pen has three attributes; colors, widths, and open/close conditions. The turtle on the screen moves with commands relative to its position, such as "move forward 100 spaces" and "turn right 90 degrees". Using Turtle Graphics, students can see the program's immediate visual output, and they can edit the program according to the envisioned output.

```
public class MyTurtleApp {
  public static void main(String[] args) {
    MWorld myWorld = new MWorld("My First Turtle World");
    MTurtle turtle = new MTurtle(myWorld);

    // draw a square
    turtle.forward(100);
    turtle.turnRight(90);
    turtle.forward(100);
    turtle.turnRight(90);
    turtle.forward(100);
    turtle.turnRight(90);
    turtle.forward(100);
  }
}
```

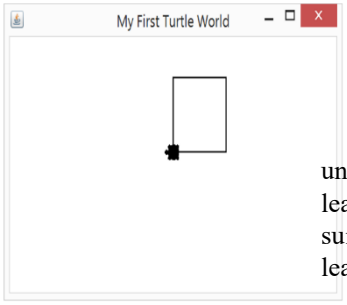

Figure 2: Example of Turtle Graphics program

Turtle Graphics can be used to learn the basic concepts of programming like selection, looping, procedure, and arrays. Figure 3 shows an example of a graphical pattern that generates using repetition structure in the Turtle Graphics approach. Other patterns also can be generated through students' imagination and creativity (Figure 4).

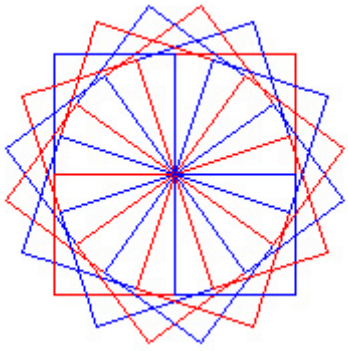

Figure 3: Example of a graphical pattern generated using the repetition structure.

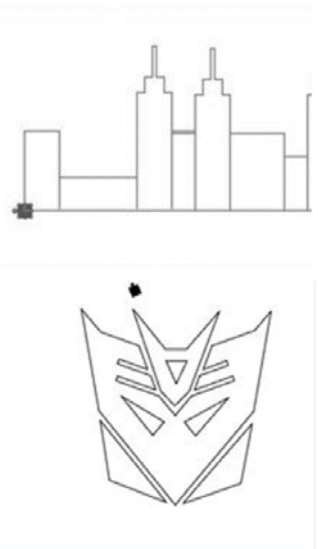

Figure 4: Examples of students' work.

The Turtle Graphics approach is easy to understand and enhances students' motivation to learn to program [26], [28]. Therefore, it is very suitable for the use of programming language learning at the school level [29].

3. METHODOLOGY

This study was carried out to investigate the perceptions and motivations of Malaysian lower secondary school students toward teaching and learning computer programming and their perceptions of using Turtle Graphics in computer programming class. The study was conducted using a survey method, and it took seven weeks to collect the data.

The teachers teach using the given Turtle Graphics module that consists of 10 topics; Introduction to Java Programming, Introduction to the Programming Environment, Debugging, Algorithm, Class and object, Variable, Data type, Input and output, Control Structure, and Repetition Structure. The teachers were provided with guidelines. At week seven, a hardcopy of questionnaires was distributed to 166 students with different socio-economic backgrounds were selected from six secondary schools across Selangor.

The respondents were form-four (F4) students who registered for Computer Science as an elective subject. The questionnaire consists of four

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

parts; the demographic, Students' perceptions towards their Computer Programming class, students' perceptions of the Turtle Graphics Module, and students' motivation toward the Computer Programming Course.

The respondents' profile, as obtained from their demographic information, is revealed in Table 2. The majority of the respondents are female, which is 70.45%, while male respondents are 29.55%. 24.40% of the respondents are from the science track, while 75.60% of the respondents are from the technical and vocational track. Most of the respondents (76.29%) have a background in the Basic of Computer Science subject, which they studied during form-one (F1).

Table 2. Profile of Respondents

Demographics	Category	Percentage
Gender	Male	29.55%
	Female	70.45%
Track	Science	24.40%
	Technical & vocational	75.60%
Background in CS	Yes	76.29%
	No	23.71%

The questions are based on the Likert scales ranging from 1(Strongly Disagree), 2(Disagree), 3(Slightly Agree), 4(Agree) to 5(Strongly Agree). We grouped the respondents' responses into three groups; negative, neutral, and positive. The Positive category refers to the 'Strongly Agree' and 'Agree' answers, the Neutral category refers to the 'Slightly Agree' answer, and the Negative category refers to the 'Strongly Disagree' and 'Disagree' answers.

4. RESULTS

4.1 Students' Perceptions of Computer Programming Class

The students were asked to give perceptions of their computer programming class. Table 3 shows the results. The majority of the students (69.28%) agreed that the programming language's logic and syntax is difficult to understand. Besides, 63.25% of the students said that the programming language used is not relevant. However, most of them (41.57%) agreed that the teacher's examples are easy to understand, and 43.98% agreed that the given materials help them understand the subject.

Table 3 also indicates that the students satisfied the teacher's teaching methods (item No. 3). 52.41% of the students said they liked the teacher's approach very much compared to only 4.22% loathing it, while 43.37% on the neutral side.

Table 3: Students' perception Of Programming Class

No.	Item	Mean	Percentage		
			-ve	Slightly Agree	+ve
1	I feel the logic and syntax of programming language easy to understand	1.34	69.28	27.71	3.01
2	I feel the programming language used in the class is relevant	1.67	63.25	6.02	30.72
3	I love the teaching pedagogical from teacher	2.48	4.22	43.37	52.41
4	The examples provided by the teacher are easy to understand	2.21	20.48	37.95	41.57
5	The reference material provided by the teacher helped me to understand the programming language	2.27	16.87	39.16	43.98

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.2 Students' Perceptions of Turtle Graphics Module

Table 4 shows the result of part three. Generally, most students accept the module and rate it as neutral (69.87% for item No. 6, and 57.23% for item No. 7). They agree that the contents of the module are relevant and in line with the technology developments. The students also accept the contents, examples, and exercises appropriately (item No. 8 to 12). The contents of the module and the examples provided are sufficient for the students to understand the topics. The majority of the students (which is 61.45% rate it as neutral) appreciate the examples given as it helps them understand the topics better.

The students also agree with the module's presentation, the style of presentation, distribution of topics, and language used (items No. 13, 14, 15, 16, and 17). However, most students said that the explanation in the module is hard to understand.

The students also agree that the module helps them improve their knowledge and programming skills (items No. 18, 19, and 20). 39.16% of the students said that the module could improve their programming skills, 2.42% disagree, and 58.43% are unbiased. Besides, the proposed module can be used as self-learning exercises, where the students can practice on their own.

Table 4: Students' perception Of Turtle Graphics Module in General

No.	Item	Mean	Percentage		
			-ve	Slightly Agree	+ve
6	Generally, do you agree with this module	1.85	22.44	69.87	7.69
7	The contents of this module are relevant and in line with technological developments	1.57	42.77	57.23	0
8	This module achieves its objectives	1.71	33.73	61.45	4.82
9	The contents of this module are sufficient for me to understand the subject of learning	1.80	27.11	65.66	7.23
10	The contents of this module are related to the course evaluation	1.82	22.89	72.29	4.82
11	The examples included are sufficient	1.84	17.47	80.72	1.81
12	The examples included help to understand this module	2.08	15.06	61.45	23.49
13	The distribution of topics in this module is reasonably long	1.90	10.24	89.16	0.6
14	The style of presentation of this module is interesting	1.80	19.88	80.12	0
15	The explanation in this module is easy to understand	1.31	68.67	31.33	0
16	The language used in the module corresponds to my level of mastery	1.75	25.9	73.49	0.6
17	The objectives of the module are clearly stated	2.07	12.05	69.28	18.67
18	This module can improve my programming skills	1.63	39.16	58.43	2.41
19	This module can improve my knowledge	1.73	31.93	62.65	5.42
20	This module allows me to practice individually	1.55	45.18	54.82	0

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

4.3 Students' Motivation Toward Learning Computer Programming

We also asked students what motivated them to learn computer programming subjects—the result in Table 5. The majority of the students (52.73% rate as neutral and 6.06% rate it as positive) said they feel excited to learn computer programming. 40.96% of the students said that they choose to be in the computer programming class compared to only 9.04% of the students that opposed it. The majority of the students (41.57% rate as neutral and 27.11% rate as positive) said they would like to study computer science further after they finish school.

The majority of the students (54.22%) said that they would ask the teachers or friends if they need an explanation on the subject. Sometimes they also try to find the answer on the Internet.

Table 5: Students' Motivation Towards Learning Computer Programming

No.	Item	Mean	Percentage		
			-ve	Slightly Agree	+ve
21	I feel excited when learning to program	2.35	6.06	52.73	41.21
22	Be in this programming class is my choice	2.22	19.28	39.76	40.96
23	If I do not understand any programming topics, I will ask a teacher or a friend or search the Internet	2.45	9.04	36.75	54.22
24	I would like to further my studies in computer science after SPM	2.04	27.11	41.57	31.33
25	The module design helped me in planning my study	2.12	4.22	79.52	16.27

5. DISCUSSION

The study investigates the perceptions and motivations of Malaysian lower secondary students towards the teaching and learning of computer programming. The students have a negative perception of the programming subject. The students alleged that the logic and syntax of the programming language is difficult to understand. Even though they agreed that the teacher's examples are easy to understand, and the given materials help them understand the subject.

We also introduced an exciting approach, via a module, to teach basic programming, and we would like to know the students' opinions on this module. The new module adopts the Turtle Graphics approach to teach computer programming. We asked the teachers to use this module to teach computer programming as a supplement to the preexisting module. We then asked the students' perceptions of this module, and the majority of them like it. The provided module has helped the students to understand the subject, and it improved their self-efficacy.

In the last part, we asked the students their motivation to study computer programming. Amazingly, most students said that their liking for the subject had motivated them to study computer programming subjects. They also plan to further studies in computer science during their tertiary education level. This type of motivation is known as intrinsic motivation, where the engagement in an activity mainly for pleasure and satisfaction [30].

The results from this research show some similarities with the research done by [25], [26], [31], where intrinsic motivation leads to self-motivation in pursuing the learning. It is also discovered that the new module introduced did manage to spark some motivation in learning computer programming. The teachers' teaching approach has also grabbed the students' attention and improves students' motivation through engaging learning experiences.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

6. CONCLUSION

This study provides some convincing evidence of common problems in learning programming. The material and the teaching approach are essential factors in giving confidence, satisfaction, and enjoyment in learning programming. One of the exciting findings shows that students who learn fundamental concepts of computer science at a young age most probably will pursue computer science. They enjoy learning computer programming; nevertheless, they said it is difficult to understand the subject.

Some limitations of this study include the limited number of schools involved in the study. This is because we only consider schools that teach Java programming. Thus, it is essential to realize that this study's results may not be extrapolated to other geographical regions or to schools that adopt other programming languages. In such scenarios, further studies will have to be carried out. The authors are currently trying to develop other Turtle Graphics modules using Python, C, and C++ programming language.

We hope these findings will help teachers strategize their teaching approach to nurture positive computer programming perceptions and motivate them to grip their skills. Programming helps give a better overall understanding of the rapidly changing technology that affects our daily life. Learning programming at the school level can help children develop these skills and play a part in the change. Learning programming, however, is not easy and notoriously hard for school level students. The success is strongly dependent on student motivation, interest, and vital foundational skill. It is essential to understand the factors influencing students' motivation, confidence, satisfaction, and enjoyment of studying computer programming. We also hope the findings will give overall views on the Computer Science curriculum, particularly in programming. The information can be used to prepare on which computer programming language can be taught.

CONFLICT OF INTEREST

The authors declare that there is no conflict of interest regarding the publication of this paper.

ACKNOWLEDGMENT

This work was supported by the Ministry of Higher Education, Malaysia, under research grant fund FRGS/1/2016/ICT01/UKM/02/3 and University Kebangsaan Malaysia under grant GGPM-2020-026.

REFERENCES:

[1] S. Hargrave, "Rise Of The Machines: Why Coding Is The Skill You Have To Learn," *The Guardian*, 2018. [Online]. Available: https://www.theguardian.com/new-faces-of-tech/2018/oct/25/rise-of-the-machines-why-coding-is-the-skill-you-have-to-learn. [Accessed: 23-Sep-2020].

[2] A. Luxton-Reilly *et al.*, "Introductory Programming: A Systematic Literature Review," in *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, 2018, pp. 55–106.

[3] F. Buitrago Flórez, R. Casallas, M. Hernández, A. Reyes, S. Restrepo, and G. Danies, "Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming," *Rev. Educ. Res.*, vol. 87, no. 4, pp. 834–860, 2017.

[4] M. Webb *et al.*, "Computer Science in K-12 School Curricula of the 2lst Century : Why , What and When ?," *Educ Inf Technol*, vol. 22, pp. 445–468, 2017.

[5] J. M. Wing, "Computational Thinking and Thinking About Computing," *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.*, vol. 366, no. 1881, pp. 3717–3725, 2008.

[6] D. Hickmott, E. Prieto-Rodriguez, and K. Holmes, "A Scoping Review of Studies on Computational Thinking in K–12 Mathematics Classrooms," *Digit. Exp. Math. Educ.*, vol. 4, no. 1, pp. 48–69, 2018.

[7] X. Basogain, M. A. Olabe, J. C. Olabe, R. Ramirez, M. Del Rosario, and J. García, "PC-01: Introduction to computational thinking: Educational technology in primary and secondary education," *2016 Int. Symp. Comput. Educ. SIIE 2016 Learn. Anal. Technol.*, pp. 1–5, 2016.

[8] T. Bell, "Computer science in K-12 education: The big picture," *Olympiads in Informatics*, vol. 12, pp. 3–11, 2018.

[9] N. C. C. Brown, S. Sentance, T. Crick, and S. Humphreys, "Restart: The Resurgence of Computer Science in UK Schools," *ACM Trans. Comput. Educ.*, vol. 1, no. 1, pp.

366–374, 2010.

[10] S. Kanemune, S. Shirai, and S. Tani, "Informatics and programming education at primary and secondary schools in Japan," *Olympiads in Informatics*, vol. 11, no. April 2016, pp. 143–150, 2017.

[11] Peter Seow, C.-K. Looi, M.-L. How, B. Wadhwa, and L.-K. Wu, "Educational policy and implementation of computational thinking and programming: Case study of Singapore," in *Computational Thinking Education*, S.-C. Kong and H. Abelson, Eds. Springer, 2019, pp. 345--361.

[12] A. Pears *et al.*, "A Survey of Literature on the Teaching of Introductory Programming Arnold," vol. 1846, pp. 161–180, 2018.

[13] M. Rahmat *et al.*, "Major problems in basic programming that influence student performance," vol. 59, pp. 287–296, 2012.

[14] N. Bubica and I. Boljat, "Predictors of Novices Programmers ' Performance," *ICERI2014 Conf.*, no. November, pp. 1536–1545, 2014.

[15] I. Ouahbi, F. Kaddari, H. Darhmaoui, A. Elachqar, and S. Lahmine, "Learning Basic Programming Concepts by Creating Games with Scratch Programming Environment," *Procedia - Soc. Behav. Sci.*, vol. 191, pp. 1479–1482, 2015.

[16] E. Aivaloglou and F. Hermans, "How kids code and how we know: An exploratory study on the scratch repository," *ICER 2016 - Proc. 2016 ACM Conf. Int. Comput. Educ. Res.*, no. Section 5, pp. 53–61, 2016.

[17] D. Bau and D. A. Bau, "A preview of Pencil Code a tool for developing mastery of programming," *Promot. 2014 - Proc. 2nd Work. Program. Mob. Touch, Part SPLASH 2014*, pp. 21–24, 2014.

[18] F. Heintz, L. Mannila, K. Nygårds, P. Parnes, and B. Regnell, "Computing at school in Sweden - Experiences from introducing computer science within existing subjects," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 9378, pp. 118–130, 2015.

[19] T. Sapounidis, S. Demetriadis, and I. Stamelos, "Evaluating children performance with graphical and tangible robot programming tools," *Pers. Ubiquitous Comput.*, vol. 19, no. 1, pp. 225–237, 2015.

[20] N. F. A. Zainal, R. Din, N. A. A. Majid, M. F. Nasrudin, and A. H. A. Rahman, "Primary and secondary school students perspective on Kolb-based STEM module and robotic prototype," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 8, no. 4–2, pp. 1394–1401, 2018.

[21] F. Kalelioğlu and Y. Gülbahar, "The effects of teaching programming via Scratch on problem solving skills: A discussion from learners' perspective," *Informatics Educ.*, vol. 13, no. 1, pp. 33–50, 2014.

[22] N. F. A. Zainal, S. Shahrani, N. F. M. Yatim, R. A. Rahman, M. Rahmat, and R. Latih, "Students' Perception and Motivation Towards Programming," *Procedia - Soc. Behav. Sci.*, vol. 59, pp. 277–286, 2012.

[23] A. Forte and M. Guzdial, "Motivation and nonmajors in computer science: Identifying discrete audiences for introductory courses," *IEEE Trans. Educ.*, vol. 48, no. 2, pp. 248–253, 2005.

[24] R. M. Ryan and E. L. Deci, "Intrinsic and Extrinsic Motivations : Classic Definitions and New Directions," vol. 67, pp. 54–67, 2000.

[25] K. M. Y. Law, V. C. S. Lee, and Y. T. Yu, "Learning motivation in e-learning facilitated computer programming courses," *Comput. Educ.*, vol. 55, no. 1, 2010.

[26] M. A. Bakar, M. Mukhtar, and F. Khalid, "The Effect of Turtle Graphics Approach on Students' Motivation to Learn Programming: A Case Study in a Malaysian University," *Int. J. Inf. Educ. Technol.*, vol. 10, no. 4, pp. 290–297, 2020.

[27] M. E. Caspersen and H. B. Christensen, "Here, there and everywhere - on the recurring use of turtle graphics in CS1," pp. 34–40, 2000.

[28] A. Perumal, R. Latih, and M. Abu Bakar, "MyJavaSchool: Students' Perceptions and Motivation for Computer Programming," *Asia-Pacific J. Inf. Technol. Multimed.*, vol. 08, no. 02, pp. 71–78, 2019.

[29] K. J. Mackin, "Turtle Graphics For Early Java Programming Education," *Artif Life Robot.*, vol. 24, pp. 345–351, 2019.

[30] V. Gopalan, J. A. A. Bakar, A. N. Zulkifli, A. Alwi, and R. C. Mat, "A review of the motivation theories in learning," *AIP Conf. Proc.*, vol. 1891, no. October 2017, 2017.

[31] S. Mohanarajah, "Increasing Intrinsic Motivation of Programming Students: Towards Fix and Play Educational Games," *Issues Informing Sci. Inf. Technol.*, vol. 15, pp. 069–077, 2018.