THE IMPACT OF USING VISUAL PROGRAMMING ENVIRONMENT TOWARDS COLLEGE STUDENTS ACHIEVEMENT AND UNDERSTANDING IN PROGRAMMING

Azniah Ismail, Siti Sakinah Mohd Yusof and
Nor Hasbiah Ubaidullah

Computing Department, Faculty of Arts, Computing and Creative Industry,
UniversitiPendidikan Sultan Idris, Perak, Malaysia

ABSTRACT

This study aimed to identify the impact of using a visual programming environment on college students' achievement and understanding when learning computer programming. In this quasi-experimental study, 91 students were divided systematically into an experimental group (53 students) and a control group (38 students). The experimental group were exposed with a visual programming environment while the control group were using an ordinary text-based programming environment. Data was collected using pre-test and post-test, then analysed using paired t-test, independent sample t-test and thematic content analysis. A significant increase in the students' achievement was recorded during the paired t-test for both groups. However, there is no significant difference in the students' achievement between the groups. Surprisingly, the thematic analysis showed that students' understanding in the experimental group were improved relatively better than in the control group. Thus, we conclude that visual programming environment have better impact to the students' understanding.

KEYWORDS

Visual Programming Environment, Learning Computer Programming, Achievement, Understanding, College Students

1. INTRODUCTION

Programming is an essential skill that must be mastered by anyone interested in studying computer science [1]. According to Lye and Koh [2], programming is more than just coding because it exposes students to problem-solving using computer science concepts like abstraction and decomposition. Problem solving is a part of cognitive skills that is required among students. In Miliszewska and Tan [3], complex cognitive skills such as planning, problem solving and analytical thinking are said to play strong roles when learning computer programming. The skills will be very useful when it comes to analyzing the scenario of a problem given and eventually will help students to come up with good solutions. During the process of learning, students are required to understand problems given, to design some possible solutions, to build the codes and implement them.

According to Sarkawi and Bakary [4], previous literature on learning programming was quite scarce but there were some research emphasized on the importance of mental model in programming. Hence, another complex cognitive skill required when learning programming is the algorithmic thinking skill. Algorithmic thinking is a key ability in informatics that can be developed independently from learning programming [5]. By having good algorithmic thinking,

students can be expected to be able to visualize the problems and their possible solutions with some algorithmic concepts such as correctness, termination, determinism and parallelism. Algorithmic thinking is a synonym to the computational thinking or programming thinking [6].

1.1. PROBLEMS ASSOCIATED WITH LEARNING COMPUTER PROGRAMMING

Learning computer programming has commonly been associated with conceptual misunderstanding problems that appear to be frequently discussed among many researchers as in Sarkawi and Bakary [4], Kohlit [7], Tie and Umar [8], Ismail et al. [9], and Abdullah and Abbas [10], especially in the computer science education. Ismail et al. [1] emphasized that the problems may have resided in knowing syntax or understanding of concepts but most definitely in the program planning. For example, students may know and understand a syntax but have problems to apply it in their program solutions.

Table 1. Problems in computer programming found in literature

Problem solving phase		**Implementation phase**
Analysis	**Solution (general)**	**Solution (detail)**
Lack of problem-solving skills Lack of analytical thinking skills Lack of logical and reasoning skills Lack of programming planning Lack of programming conceptual understanding Lack of algorithmic skills	Inefficient tools used in representing problem solution Do not understand and unable to explain semantics actions in a program Ineffective design and testing problem solution	Do not understand and master the programming syntax and functions Unable to apply correct rules of syntax when programming Unable to use semantic knowledge of programming to write program Ineffective code and testing program to solve novel problem

(*Source: Ismail et al. [1]*)

Ismail et al. [1] divided the problems they found from the literature into problem solving phase and implementation phase as in Table 1. They conducted further investigation and managed to identify four main problems as in Table 2. They then suggested tackling the critical part of the programming process which starts at the analysis of the problem solving as it will affect the next phase of the programming sequence.

Table 2. Four main problems with learning computer programming in [1]

Problem Type
1. Lack of skills in analysing problems.
2. Ineffective use of problem representation techniques for problem solving.
3. Ineffective use of teaching strategies for problem solving and coding.
4. Do not understand and master the programming syntax and constructs.

Without or with slow improvements of the required skills, students will highly likely find difficulties to solve programming problems they have in hand. Hence, programming becomes extremely hard and very challenging to many students [7]. Ismail et al. [1] seem to agree and mentioned further that with programming as the basic skill required of computer programmers, it may have given negative consequences in the learners' attitude towards the field. Thus, teachers or lecturers have to face real challenges in teaching programming and promoting the required skills [1], [2]. Teaching through conventional method, i.e. traditional lectures in a lecture hall or a computer lab may be less effective [2], [6], [10].

1.2. VISUAL-BASED SOLUTION APPROACHES

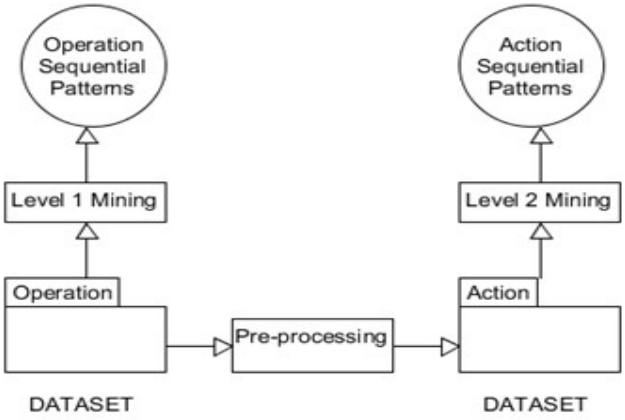

Figure 1. A two-level sequential pattern mining in Futschek and Moschitz [15]

We are interested to look for solutions or alternatives that may help teachers or lecturers to promote the required skills. One of the popular approaches is by introducing tools that develop games such as in Yau [11], Fronza et al. [12], Chau [13], and Ismail et al. [14]. However, we would like to perceive the game tools in different view, not as a game but more as a visual way of looking at the programs. Futschek and Moschitz [15] shared an interesting insight of physical, tangible object as transition to a virtual programming environment, whilst Shih [16] shared the use of Blockly Game to help students visualize the sequential pattern in programming (see the sequential pattern mining method they used in Figure 1). Further discussion on visual programming framework can be found in Idrees et al. [17].

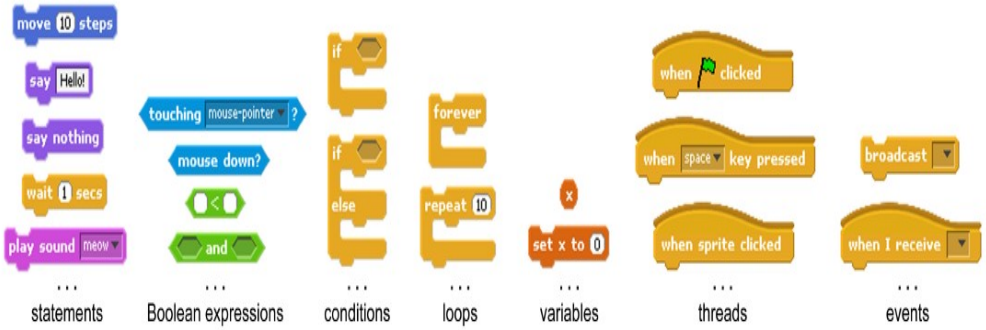

Figure 2.Programming blocks in Scratch visual programming

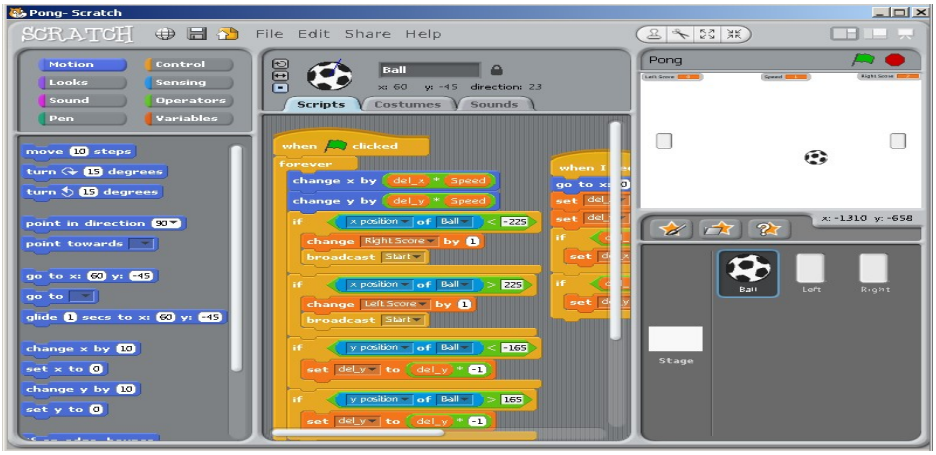

Figure 3.Scratch visual programming environment

In this article, we discuss the impact of visual-based programming environment on students' achievement and their understanding, especially when related to sequential, selection, and iteration in programming. We have chosen Scratch *(see* Figure 2 *and* Figure 3*)* to provide the visual programming environment at one of colleges in Perak.

2. RESEARCH METHODOLOGY

The methodology that was used in this study is briefly described in this section.

2.1. RESEARCH SCOPE

The study was conducted at a college that offers a Computer Science course to the college students. One of the topic in this course is programming. The study focused on achievement and understanding in learning programming when given the visual programming environment (*see* Figure 4*)*. The population was the two-year program students which consists of 114 students. The sample was then reduced to 91 students using their pre-test scores as the primary base for the selection.

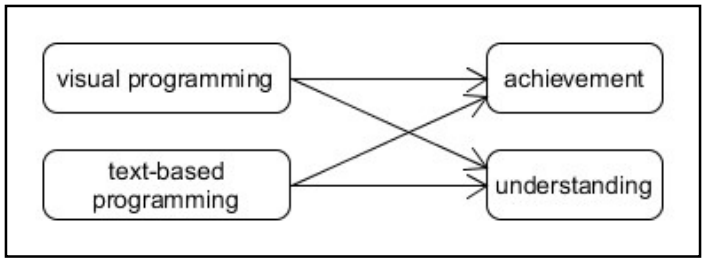

Figure 4.Research scope

2.2. RESEARCH DESIGN

The study follows a quantitative quasi-experimental research design (*see Figure 5).* Basically, there were two different groups formed in this study, i.e., one control group and one treatment group.

The following are parts of the study:

i) In general, all students took a pre-test during the initial stage. Based on the pre-test score, only students with pre-test score equal or less than 65 points will be considered as sample of the study.

ii) Those selected will be divided into two groups based on their original classes.

iii) The control group used a text-based programming environment whilst the experimental group was exposed to the use of visual programming environment, i.e. the Scratch

iv) Both groups took a post-test once the teaching and learning process for both groups were completed.

v) The pre-test and post-test scores for both groups were then compared and tested against the hypothesis using a t-test. Further qualitative tests using thematic content analysis were also conducted. All tests required are listed as in Table 3.

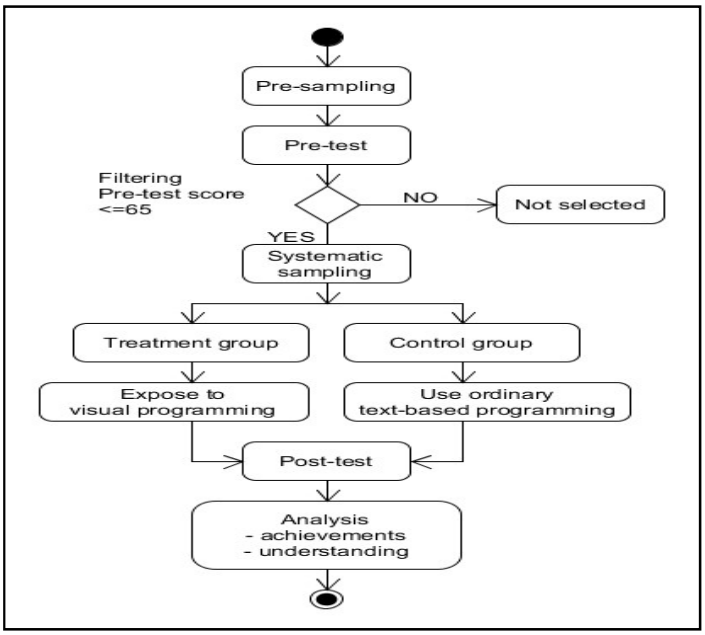

Figure 5.Research design

The t-test was chosen because it can be used to measure an increment (or decrement) in the students' achievement using the pre-test and the post-test scores. Thematic content analysis were used to observe improvements in the qualitative data i.e. the students' answers in both pre-test and post-test for both treatment and control groups.

Table 3. Data analysis design.

No	Independent var.	Dependent var.	Test name	Data required
1	Visual programming environment	(a) Achievement	Paired T-test	Mean score of pre-test and post-test (quantitative)
		(b) Understanding	Thematic content analysis	Student answers in the pre-test and post-test (qualitative)
2	Text programming environment	(a) Achievement	Paired T-test	Mean score of pre-test and post-test (quantitative)
		(b) Understanding	Thematic	Student answers in the pre-

			content analysis	test and post-test (qualitative)
3	(Comparison)	(a) Achievement	Compare 1(a) with 2(a)	Independent sample t-test on the mean scores of post-tests
		(b) Understanding	Compare 1(b) with 2(b)	Thematic content analysis results

2.3. RESEARCH INSTRUMENT

Main instruments used to collect data in this study were a set of a pre-test and a post-test. The description are as follows:

i) Pre-test: a pre-test session was conducted for all 114 students who were taking the Computer Science course at the college. The pre-test questions contained 6 questions and the structure was as in Table 4. All questions must be answered within 2 hours.

ii) Post-test: a post-test session for both student groups (the control and the treatment groups) were conducted after the teaching and learning sessions ended. The structure of the post-test questions was similar to the pre-test questions. All questions must be answered within 2 hours.

Table 4. The structures of test questions (in similar manner for pre-test and post-test).

No	Control structure	Number of questions	Question No.	Solution type
1.	Sequential	2	1a	Pseudocode
			1b	Pseudocode
2.	Selection	2	2a	Pseudocode
			2b	Flowchart
3.	Iteration	2	3a	Flowchart
			3b	C++ Coding

2.4. RESEARCH PROCEDURE

2.4.1. PILOT STUDY

In one semester prior, 50 senior students were selected at the same college (*see* Table 5) to undergo a pilot study to ensure the reliability of the pre-test and the post-test questions. High reliability means high consistency of scores that students receive on the pre-test and the post-test. The structure of the two tests was similar, both covering similar material and equal in what they measure, thus it was preferred that students' scores to be similar. The more comparable the scores are, the more reliable the test scores are. Therefore, we expected a high consistency of both tests.

Table 5. Descriptive data about the participation during the pilot study.

Group	Freq. (N)	Percentage (%)
Group 1	50	100

The content validity of both pre-test and post-test was checked by two subject experts at Universiti Pendidikan Sultan Idris. The tests were then administered with the help from the Computer Science lecturers at the college. Each test contained six items and must be answered within two hours. The students were given two hours to answer the pre-test questions followed with a short break.The post-test was then administered for another two hours.

We used Cronbach's alpha to test the reliability. The analysis results showed that the pre-test had a reliability coefficient of 0.765 while the post-test had a reliability coefficient of 0.725. Thus, both tests had high consistency of the cronbach's alpha values (*see* Table 6). Thus, the instruments were adequate for our study.

Table 6. Cronbach's alpha values for both pre-test and post-test.

Test Name	Number of questions	Cronbach's alpha value
Pre-test	6	0.765
Post-test	6	0.725

2.4.2. SAMPLING PROCEDURE

All 114 students that were taking Computer Science course that semester, were considered to take the pre-test. From the 114 students, the number reduced to 91 students because we decided that only students with pre-test score equals or less than 65 points were taken in for further examination in the study.

The 91 students were separated into two groups based on their existing classes. Therefore, one group formed 53 students and the other group formed 38 students. We decided that the first was considered as the treatment group and the latter was the control group (*as shown in* Table 7).

Table 7. Number of students in the control and the treatment groups.

No	Group	Freq. (N)
1.	Treatment	53
2.	Control	38
	TOTAL	91

3. FINDINGS AND DISCUSSION

In this section, we present the results and discuss the findings based on the hypothesis that formulated for this study.

Effects of using visual programming environment on the students' achievement

HO_1: There is no significance improvement on students' achievement when they were exposed to the use of visual programming compared to students using the text-based programming

A paired t-test was first used to compare the students' achievement in the pre-test and the post-test in the same group. The test was conducted to the treatment group's test scores. Based on the results presented in Table 8, the improvements recorded among students in the treatment group were significant (at p-value < 0.001) with mean score of 50.14 in the pre-test and mean score of 88.99 in the post-test.

Table 8. Mean scores of pre-test and post-test for the treatment group.

Group	N	Mean Score (Std. deviation)		p
		Pre-test	Post-test	
Treatment Group	53	50.14 (10.34)	88.99 (9.76)	< 0.001

We also conducted another paired t-test to compare the students' achievement in the pre-test and post-test among students in the control group. The findings showed in Table 9that they had also

recorded some significant improvement (at p-value < 0.001) with mean score of 39.97 in the pre-test and mean score of 84.28 in the post-test.

Table 9. Mean scores of pre-test and post-test for the control group.

Group	N	Mean (Std. deviation)		p
		Pre-test	Post-test	
Control Group	38	39.97 (14.66)	84.28 (12.45)	< 0.001

Given that both groups have recorded significant improvements in their achievement, the first hypothesis null might be rejected. There was only a slight difference between mean scores of post-test of both groups. We conducted a two-tailed independent sample t-test and had set to reject the null hypothesis if p-value < 0.05. We obtained results as in Table 10. Thus, we failed to reject the null hypothesis. This means although both groups have recorded significant improvement but the comparison of achievement between the two groups was statistically non-significant.

We concluded that learning programming can actually take place whether using a visual programming environment like scratch or just by using ordinary text-based programming environment only.

Table 10. T-test results on mean scores of post-test for both control and treatment groups.

Group	N	Post-test Mean (Std. deviation)	df	t	p
Treatment Group	53	88.99 (9.76)	89	2.02	0.9706
Control Group	38	84.28 (12.45)			

EFFECTS OF USING VISUAL PROGRAMMING ENVIRONMENT ON THE STUDENTS' UNDERSTANDING

HO_2: There is no significance improvement on students' understanding when they were exposed to the use of visual programming compared to students using the text-based programming

Table 11 shows a summary of thematic content analysis values on the students' answers in the pre-test and the post-test from the treatment group. In the pre-test, many students made large mistakes with logic errors and syntax errors. However, the understanding has certainly improved after they were exposed to the use of visual programming during their lessons.

We also conducted content analysis on the students' answers from the control group. Table 12 shows the analysis results for the control group. In general, we observed that the students' understanding in the control group also had largely improved despite the facts that they learned only using the ordinary text-based programming language. Given that both groups have recorded significant improvements in their understanding, the second hypothesis null was also rejected.

Table 11. A summary for the qualitative content analysis results for treatment group.

Sub Topic	Question No.	Solution Type	Pre-test	Post-test
Sequential	1a	Pseudocode	1 student succeed, 2 failed to print *name*, 1failed to print *name* and the required text.	7 students scored full marks, 1 print text but forgot to print variable.
	1b	Pseudocode	6 students did not use sequential.	All scored full marks.
Selection	2a	Pseudocode	6 students failed to answer, 3 did not use	4 students scored full marks.

			selection.	
	2b	Flowchart	5 students gave logic errors, 3 incomplete answers, 1 failed to answer	1 scored full marks, 6 still have logic errors.
Iteration	3a	Flowchart	All failed. 3 gave incomplete answers, 3 had logic errors	6 scored full marks, 2 kept doing logic errors
	3b	C++ coding	1 student succeed, 1 failed, 5 had syntax error and 1 logic error.	2 scored full marks. 5 students had minor errors.

However, when we look closely into the details of answers between both groups, we had observed some slight differences on the improvements between the two groups. For example, the answers given by the treatment group had largely improved for one of the sequential question (question number 1a) and one of the selection question (question number 2a). None of the student in the treatment groups has problem to build the solutions requested compared to the control group. This might be a working evidence of visual programming environment's effects that can aid in promoting algorithmic thinking among students in the treatment group especially when the problem-solving was related to sequential and selection.

No significant difference was observed when compared between the treatment and the control groups for question number 1b. Question number 1b might be relatively easy for both groups. On the other hand, the control group had shown slightly better improvements compared to the treatment groups when it comes to the iteration questions. Further investigations should be carried out to see if visual programming environment does help in learning certain programming structures only, as the cases in our study, the sequential and the selection questions.

Table 12. A summary for the qualitative content analysis results for control group.

Sub Topic	Question No.	Solution Type	Pre-test	Post-test
Sequential	1a	Pseudocode	9 students failed to include *name* at all	7 failed to print *name,* 1 kept doing the same mistakes
	1b	Pseudocode	2 students did not know average calculation.	All scored full marks.
Selection	2a	Pseudocode	6 students failed to answer	Only 6 students scored full marks, 2 students still did not understand the questions.
	2b	Flowchart	All failed to answer	3 scored full marks. Others still have logic errors.
Iteration	3a	Flowchart	All failed.	Only 4 scored full marks.
	3b	C++ coding	All had syntax errors, 1 had logic error.	3 scored full marks. 4 students had minor errors.

4. CONCLUSION

In this study, we discuss the effects of using visual programming environment among college students in learning programming. In this study, quantitative data of pre-test and post-test scores were tested against the formulated hypothesis using t-test. Although significant improvements in term of students' achievement were observed, similar improvements were also recorded with students that only used ordinary text-based programming language. Qualitative analysis using thematic content analysis had shown similar findings.

However, when we looked deeper into the qualitative data, there were slight differences in the

understanding between students that used visual programming environment and students that used text-based programming environment. There were some evidences that students in the treatment group have relatively better understanding about the problems and solutions compared to the students in the control group. The improvements were quite obvious for questions related to selection and sequential. These findings are interesting. Clearly, further investigation is required.

ACKNOWLEDGEMENTS

The authors would like to thank all colleagues and friends from UniversitiPendidikan Sultan Idris who provided insight and expertise that greatly assisted the research.

REFERENCES

[1] Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010a). Instructional strategy in the teaching of computer programming: a need assessment analyses. TOJET: The Turkish Online Journal of Educational Technology, 9(2).].

[2] Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: what is next for K-12?. Computers in Human Behavior, 41, pp. 51-61.

[3] Miliszewska, I. & Tan, G. (2007). Befriending computer programming: a proposed approach to teaching introductory programming. Informing Science and Information Technology, 4, pp. 278-289.

[4] Sarkawi, S., & Bakary, S.A. (2011). Kesilapfahaman konsep semantik dan sintaktik pengaturcaraaan dalam kalangan pelajar Kolej Matrikulasi Perak. Presented in the Multimedia and creative content symposium 2011.

[5] Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. In the proceedings of The International Conference in Informatics in Secondary Schools - Evolution and Perspectives, ISSEP 2006, Vilnius, Lithuania, November 7-11, 2006

[6] Silva, L. R., da Silva, A. P., Toda, A., &Isotani, S. (2018, July). Impact of teaching approaches to computational thinking on high school students: a systematic mapping. In 2018 IEEE 18th International Conference on Advanced Learning Technologies (ICALT), pp. 285-289. IEEE.

[7] Kohlit, M. (2014). Hubungan antara persepsi dan tahap kefahaman pelajar kolej matrikulasi perak terhadap topik pengaturcaraan C++. In the proceedings of Seminar BMKPM 2014.

[8] Tie, H. H., & Umar, I. N. (2011). Does a combination of metaphor and pairing activity help programming performance of students with different self regulated learning level? The Turkish Online Journal of Educational Technology, 10 (4).

[9] Ismail, M. N., Ngah, N. A., & Umar, I. N. (2010b). The effects of mind mapping with cooperative learning on programming performance, problem solving skill and metacognitive knowledge among computer science students. Journal of Educational Computing Research, 42(1), pp. 35-61.

[10] Abdullah, A. J., & Abbas, M. (2004). Kesan pembelajaran koperatif terhadap prestasi dalam mata pelajaran matematik berbantukan koswer multimedia Kementerian Pendidikan Malaysia di kalangan pelajar-pelajar tingkatan dua. In the proceedings of Konvensyen Teknologi Pendidikan ke-17, pp. 32 – 40.

[11] Yau, M. (2013). Engaging Hispanic/Latin (a) youth in computer science: an outreach project experience report. Journal of Computing Sciences in Colleges, 28(4), pp. 113–121.

[12] Fronza, I., El Ioini, N., and Corral, L. (2015). Students want to create apps: leveraging computational thinking to teach mobile software development. In Proceedings of the 16th Annual Conference on Information Technology Education, pp. 21–26. ACM.

[13] Chao, P. Y. (2016). Exploring students' computational practice, design and performance of problem-solving through a visual programming environment. Computers and Education, 95,pp. 202–215.

[14] Ismail, A., Yatim, M. H. M., Sahabudin, N. A., & Zain, N. Z. M (2016). Keupayaan murid sekolahrendahmempelajaridanmenerokaibahasapengaturcaraan visual. Journal of ICT in Education (JICTIE), 3(1), pp. 89-97.

[15] Futschek, G., &Moschitz, J. (2011, October). Learning algorithmic thinking with tangible objects eases transition to computer programming. In International Conference on Informatics in Schools: Situation, Evolution, and Perspectives, pp. 155-164. Springer, Berlin, Heidelberg.

[16] Shih, W. C. (2017, June). Mining learners' behavioral sequential patterns in a blockly visual programming educational game. In Industrial Engineering, Management Science and Application (ICIMSA), 2017 International Conference, pp. 1-2. IEEE.

[17] Idrees, M., Aslam, F., Shahzad, K., & Sarwar, S. M. (2018). Towards a universal framework for visual programming languages. Pak. J. Engg. Appl. Sci., pp. 55-65.

AUTHORS

Dr. Azniah Ismail is a senior lecturer at the Department of Computing, UniversitiPendidikan Sultan Idris, TanjongMalim, Perak, Malaysia. She has more than 15 years of teaching experience. She had obtained her first degree (BSc.Hons.IT) from Universiti Utara Malaysia, and her master's degree (M.IT) from UniversitiKebangsaan Malaysia. She has a PhD in Computer Science from the University of York UK, an IEEE member of Malaysia Section, and has research interest in IT and software engineering education.

SitiSakinahbinti Mohd Yusof is a lecturer in KolejMatrikulasi Perak, Gopeng, Malaysia. She has more than 11 years of teaching experience in Computer Science and English. She has a Master (Education) in Information Technology from UniversitiPendidikan Sultan Idris and has an interest in IT and Computer Science. She is pursuing her PhD in IT education, particularly focusing on school children.

Dr. Nor Hasbiah Ubaidullah is working as an Associate Professor at UniversitiPendidikan Sultan Idris. She received degree in Computer Science from UniversitiKebangsaan Malaysia and master of Science in Information System from University of Salford UK. She obtained her PhD degree in Information Technology from UniversitiKebangsaan Malaysia. Her research areas include Courseware Engineering, Information System, Data Management and Dyslexia.