Block-Based Visual Programming as a Tool for Learning the Concepts of Programming for Novices

Sukirman, Dias Aziz Pramudita, Aziz Afiyanto, and Utaminingsih

***Abstract*—*Programming* is a course that prosecutes skills in critical thinking and problem-solving. However, learning programming for novices like junior high school students is no easy task. Moreover, the strategy uses text-based programming, which requires writing source code correctly to avoid the potential of syntax errors. This study aims to investigate a tool of block-based visual programming (BBVP) for learning the programming concepts. The method used in this study consists of five phases: analysis, design, development, experiment, and evaluation. Participants were 60 students of middle school age 14-15 years old divided into two groups, experimental and control groups. Data was collected through questionnaires, pretest-posttest, and analysis of codes in the project created by students. To compare the results of both two different groups, we employed independent samples t-test. The result shows that the significance score is less than 0.05 (p-value < 0.05), which means that learning programming concepts using BBVP significantly impacts novices. The concepts of programming that can be learned include variables, sequence, iteration or looping, and conditional statements.**

***Index Terms*—Block-based visual programming, learning programming, ScratchJr, programming concepts, visual programming.**

I. Introduction

Learning programming concepts for novices is one challenging endeavor [1]. Moreover, they are students still aged at the middle school level. Several years ago, programming was for university students only, but today, many schools have implemented this course at any level. It is not only challenging for teachers but also for the students. Teachers try to provide the best strategy to make students understand and quickly receive the presented learning material contents of programming while the students struggle to comprehend [2]. Technically, programming consists of instructions and logical thinking that make students learn a script for running computer commands. Many instructional learning strategies focused on using specific programming languages and tools; as a result, the understanding of the computational concepts was lacking [3]. Accordingly, students presume that programming is slightly difficult to learn since it is slightly different from other subjects. Globally even, programming has a universal problem in the computer science curriculum with high dropout and failure rates [4].

Many countries have started or are in progress to introduce programming encapsulated in computer science (CS) subject or related in their national curriculum to response to the increasing demand for IT experts in industries, like England, Finland, Sweden, and the US [5]. Programming even has been introduced at the primary education level [6]. In Portugal, the government made a policy to make CS and programming compulsory for students, from middle school until elementary [7]. It means that the trend of "learning to program" has become popular in educational institutions around the world [8], [9]. In Indonesia, programming has also been introduced, which is a part of a computer science subject named "Informatika" [10]. However, the implementation is still partially applied to several schools. Only schools with CS teachers and computer facilities are considered ready to implement the Informatika subject appointed by the government [11].

Studies show that learning programming can facilitate students to think creatively and improve their skills in problem-solving [8], [12]. Additionally, learning to program also may encourage students to engage in logical thinking and computational thinking (CT) [13]. CT itself is described as the thought processes involved in formulating problems and the solutions represented in an effective form to be carried out [14]. CT lays the conceptual foundations from programming notions that may construct programming capabilities [15]. Even [16] formulated that CT is the core amongst prevalent topics include computing, programming, coding, and problem-solving. Therefore, several researchers stated that CT and programming skills should be learned by students of future generations of the 21st century as the new literacy [8], [12], [15], [17].

Besides improving problem-solving skills and logical thinking, learning to program can also improve the students' analytical capabilities [18]. For example, when they face a case in programming, they must find a solution with strategies or imitate and modify the existing one. The process of debugging a program carried out iteratively may lead them to find a better solution. Students will think creatively and employ many strategies to tackle the problem with a better solution from this situation. On the other hand, someone who has programming skills will have more opportunities to get a job, since a software developer is one of the most wanted in this era [19]. Therefore, preparing students with programming skills not only prepares them to tackle a problem but also means provide them for getting and competing for future jobs.

However, learning programming for novices is no easy

Manuscript received October 11, 2021; revised December 7, 2021. This work was supported by Universitas Muhammadiyah Surakarta (UMS), Indonesia, through scheme of Hibah Integrasi Tridharma (HIT), Lembaga Riset dan Inovasi (LRI) UMS.

The authors are with the Department of Informatics Engineering Education, Universitas Muhammadiyah Surakarta, Indonesia (e-mail: sukirman@ums.ac.id, dap207@ums.ac.id, a710160038@student.ums.ac.id, a710160027@student.ums.ac.id).

doi: 10.18178/ijiet.2022.12.5.1628

task. Sometimes, both instructors and learners confront several problems in their learning. Several factors contributing to the difficulties are lack of understanding the programming concepts at the early stage [20], problem-solving skills, ineffective teaching strategy, and personal traits & attitude [4]. It is no longer adequate and relevant in teaching strategy since the teaching methods still employ traditional approaches, such as, text-based programmings like Pascal, C/C++, Java, or Python [8]. Text-based programming insists students use the syntax of programming language that may make them inconvenient and feel frustrated. The syntax requires writing a source code correctly since if it is mistakenly written, an error will occur to the program, making the program run unexpectedly. Instead of learning the actual programming concepts, students are even busy dealing with syntax errors [21]. Hence, it will bring down their enthusiasm and motivation for learning how to program. In comparison, programming is not just about writing a code but more than tackling a problem and finding the solution. Even [22] stated that the concept of programming itself is more important than coding language. Perhaps the most challenging part is understanding how to create an algorithm or step-by-step instructions to solve a task and the best way to write this in the code.

One strategy that can be used to learn programming concepts more friendly for novice students is using block-based visual programming (BBVP). Cárdenas-Cobo et al. (2021) stated that the use of BBVP to learn programming has several benefits, for example, reducing syntax errors since do not use text-based code but use logical block-based featured drag-and-drop visually [23]. BBVP can also be performed with various activities like creating games, animations, or even robotic programming [12]. This is why the BBVP environment has become a popular tool and powerful for introducing coding and programming for students who have just learned programming concepts [24]. BBVP environments contain elements providing useful visual cues on how to use the commands and develop programs to make users easy to understand. The program can easily be created only by dragging and dropping the elements that may focus on logical thinking and avoid syntax errors. Additionally, visualization helps students to comprehend the semantics of introduced constructs, explain the principles of program structure and execution, and degrade misconceptions [25].

Many BBVPs are available on the internet, for example, Scratch, ScratchJr, Alice, Kodu, Construct, StarLogo, App Invertor, Greenfoot, and Blockly. However, the BBVP tool that is frequently used to learn programming is Scratch [16]. This study aims to investigate another BBVP for learning the programming concepts besides Scratch, namely ScratchJr. ScratchJr allows children to create animated games and stories by putting a sequence of graphical coding blocks like features provided in BBVP Scratch. The differences are that the features of ScratchJr is more simple and can be installed on smartphone and tablet so that allowing users to learn in formal or informal settings or even at home and school. The article is organized into four sections: introduction, methodology, results & discussion, and conclusion. The method used consists of five phases, and the results will be discussed below.

II. Methodology

A. Research Phases

This study is divided into five phases as depicted in Fig. 1: analysis, design, development, experiment, and evaluation. In the analysis phase, we conducted a literature review to find related references from journals or proceedings indexed by reputable databases. Documents the from lesson plan, book references, and curriculum were also analyzed to determine and adjust the appropriate learning topics that should be selected and relevant to the available subject. Afterward, a conceptual framework was formulated to organize the research and illustrate the big picture of the study [26]. The conceptual framework of this study is illustrated in Fig. 2.

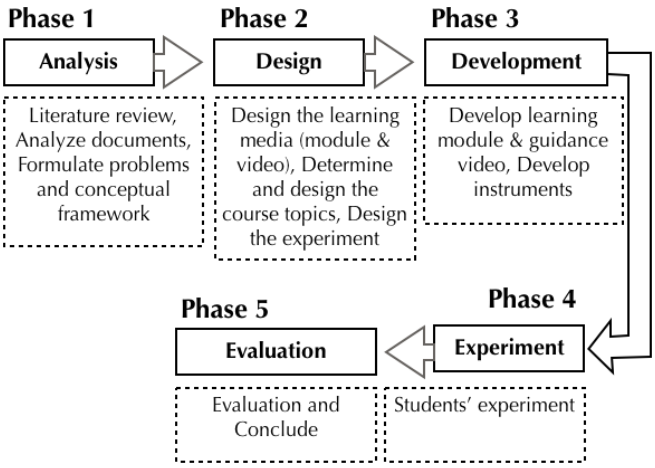

Fig. 1. Research phases of this study.

The second phase was design; they are 1) designing the learning media, including e-module and guidance video; 2) designing and determining the course topics; and 3) designing the experimental settings. The design phase has an essential role in managing and determining the course topics and the experimental design. Then, in phase three, we developed the needs determined in the previous design phase. Afterward, the experiment was carried out based on the design. In this phase, we also investigated and collected the data employed by the developed instruments. In the last phase, we evaluated all activities from the first phase until the last phase and then concluded to summarize our study.

B. Conceptual Framework

A conceptual framework is a visual representation or big picture of the expected relationship among examined variables in research [27]. We formulated the conceptual framework based on the literature review, aims, and the variables that affected the study's goals, as depicted in Fig. 2. In the experiment phase, we employed project-based learning (PjBL) approach to trigger the learning processes. We hypothesized that the students' understanding of programming concepts, including variables, sequence, iteration/ looping, and conditional statements, can be influenced by the BBVP tool and learning media encapsulated in the PjBL strategy.

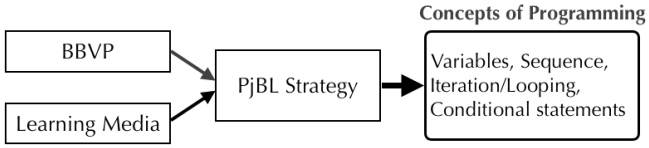

Fig. 2. Conceptual framework of the study.

C. Experimental Design

Fig. 3 shows this study's experimental design, which grouped to intervention and control, namely group A and group B, respectively. Both groups are separated by a dashed line indicating they have different treatments. The intervention or experimental group was treated by implementing the strategy of PjBL using BBVP to learn the concepts of programming. In contrast, the control group had no particular intervention, which means that the students had no distinctive handling and the learning processes were running like usual using the standard method. To find out the initial knowledge of participants, they were given a pretest. In the end, participants were given a posttest to find out the condition between before and after.

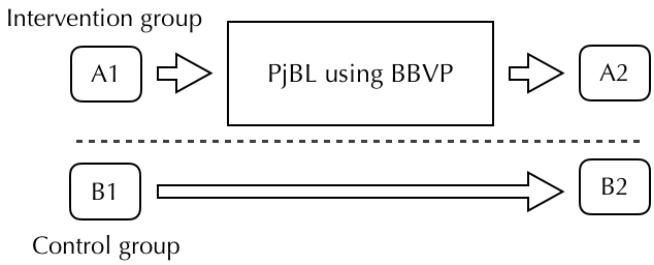

Fig. 3. Experimental design of the study.

The project employed in the PjBL settings was making a game in the course' final according to their creativity. The game project is selected because its features may introduce and reinforce procedural abstraction that is very important in the course of programming [28]. The software used to create the game project is ScratchJr, an online block-based visual programming language that offers an easy-to-implement structure and algorithm of the programs [29]. Before students carried out the final project, they obtained courses on using the software technically for five meetings. They were guided on installing and accessing the software, creating an account, and using the available features in the ScratchJr. Additionally, they were also given a module to accompany learning activities. The module contains five parts: a basic explanation of programming concepts, the introduction of ScratchJr, its features, the step-by-step guide on how to create a game project technically, and project-based instructions. The learning module was also completed by a video, so that they may re-learn afterward.

D. Participants

Participants in this study were students in grade 8 of state middle school. They were divided into two groups, intervention class and control class, consisting of 30 students in each group. The total number of participants involved in this study is 60 students. The age between 14-15 years and only four students have ever used similar BBVP, but not for learning the concepts or programming. The selected participants were students who studied in one of the schools appointed by the education ministry of Indonesia to implement the Informatika subject so that it was an appropriate place to conduct the implementation in this school. Subject matters and the guidance were delivered through Google Meet since the learning activities were conducted online as instructed by the Indonesian Minister of Education and Culture to limit the covid-19 spread [30].

E. Instruments and Data Analysis

Instruments used to collect data in this study are questionnaires, pretest, and posttest. The questionnaires were used to evaluate the usability of the learning strategy in this study. We adopted the System Usability Scale (SUS) proposed by Brooke [31] to evaluate the usability of BBVP software and the learning media we have. The answers were represented in 5 Likert scale statements: one strongly disagrees, and five strongly agree. The SUS questionnaire was adopted since it can be used to appraise the usability of a given product or service [32]. Table I shows the questionnaire related to BBVP software used to teach the concepts of programming, namely ScratchJr. Meanwhile, Table II shows the questionnaire related to learning media that we have developed to aid students in learning the concepts of programming; they are a learning module in e-book format and guidance video.

TABLE I: QUESTIONNAIRE RELATED TO BBVP SOFTWARE USED

No.	Statements
1.	I think that I would like to use this software frequently to learn programming
2.	I found the software unnecessarily complex to learn programming
3.	I thought the software was easy to use to learn programming
4.	I think that I would need the support of a technical person to be able to use this software to learn programming
5.	I found the various functions in this software were well integrated to learn programming
6.	I thought there was too much inconsistency in this software to learn programming
7.	I would imagine that most people would learn to use this software very quickly to learn programming
8.	I found the software very awkward to use to learn programming
9.	I felt very confident using the software to learn programming
10.	I needed to learn a lot of things before I could get going with this software to learn programming

To find out the differences between before and after the experiments, we used to pretest and posttest. We analyzed the results using a t-test with SPSS software. Besides, we also analyzed the game projects created by students whether they implemented the programming concepts like sequence, looping, conditional statements, and related.

III. RESULTS AND DISCUSSION

A. Project Analysis

The concepts of programming taught to the students in the intervention class of this study included sequence, iteration (looping), conditional statements, variables, lists (arrays), event handling, and Boolean logic. The concept of programming is just like a human telling to computer to do something. To understand the meaning and purposes that humans want when communicating, the computer has to understand the same language as well. The language here is

similar to the commands or instructions for computers to do something. In ScratchJr, the instructions are expressed in a block instead of text-based script code so that students can easily understand how to write instruction with visual code.

TABLE II: QUESTIONNAIRE RELATED TO THE LEARNING MEDIA USED

No.	Statements
1.	I think that I would like to use this learning media frequently to learn programming
2.	I found the learning media unnecessarily complex to learn programming
3.	I thought the learning media was easy to use to learn programming
4.	I think that I would need the support of a technical person to be able to use this learning media to learn programming
5.	I found the various information in this learning media were well designed to learn programming
6.	I thought there was too much inconsistency in this learning media to learn programming
7.	I would imagine that most people would learn to use this learning media very quickly to learn programming
8.	I found the learning media very awkward to use to learn programming
9.	I felt very confident using the learning media to learn programming
10.	I needed to learn a lot of things before I could get going with this learning media to learn programming

```
if (tap on character)
   {
      character moves up 5 grid
      then down again
    }

reset character's location
to its starting
```

Fig. 4. Example of a game project created by a student.

After receiving learning materials and guidance, students were allowed to create a game project that accomplished in groups. The group consists of 5 students so that the final game project consists of 6 games. Besides learning the concepts of programming, students also learn to collaborate in a team. Fig. 4 shows an example of a game project created by a group of students and its block coding. When the game is played by pressing the green flag symbol at the top right, the game system will be running. The gameplay is similar to the phenomenal game of flappy bird; the game object of pilot sprite has to avoid game objects of basketball and ball.

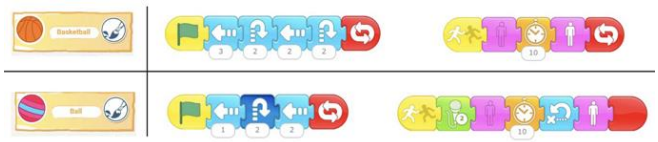

Fig. 5. Example of block code created by students.

Fig. 5 shows the block code of the pilot sprite in more detail at the bottom of the stage by selecting the sprite. It consists of three block codes, they are started on top, hop or jump, and go back to the initial position. When the pilot sprite is touched, it will jump and go back to the initial position repeatedly. From this block code, students learn the concepts of sequence instruction and conditional statements as illustrated in the pseudocode. Meanwhile, from the block code depicted in Fig. 5, students may learn looping, timer, and importing sounds. We can see that basketball and ball sprites are moving to the left repeatedly. When the sprites are bumping the pilot, they are going to hide for 10 seconds then re-appear repeatedly.

Besides the project depicted in Fig. 4, students also created other games with different sprites and instructions; for example, they do experiments to add sprites like a monkey, rabbit, pig, and other sprites. They also conducted experiments to add the available sounds to the game, and even they tried to add sound with their own voice. These explorations allowed students to think and create projects more creatively.

TABLE III: GAMES CREATED AND THEIR IMPLEMENTATIONS

Team	Game features	Programming concepts implementation
1.	The pilot has to avoid balls	Sequence, looping, conditional statements
2.	The rabbit jumps up and avoids fruits	Looping, timer, variables, using voice sounds
3.	Pig runs and avoid balls	Timer, sounds, hide and appear, variable
4.	Monkey catch the fruits	Looping, sequence, timer, hide and appear
5.	Cat play basketball	Sequence, looping, timer, hide and appear
6..	Monkey chase the fruits	Sequence, timer, pop up say, sequence

Table III shows the analysis of game projects and their features created by students based on their teams and the implemented concepts of programming. Almost all the teams employed concepts of looping and sequence since the created game projects have genre platformer games. Uniquely, each group has its characteristics, and they are all different. For example, team 3 employed recorded voice when bumping the main sprite, while team 6 used pop up say if colliding the sprites. From here, students may learn the main concepts of programming with no confront to a syntax error and complicated programming text code. Therefore, they may focus on learning the concepts of programming, logical algorithms, and problem-solving.

B. *Statistical Analysis*

Once the experiments were accomplished, the posttest was carried out to compare the intervention or experimental and control groups. The results were analyzed using SPSS statistics software. Table IV shows the descriptive statistics of pretest-posttest scores in the experimental and control groups. It can be seen that the mean score of the pretest in the experimental group is 55.33, and the posttest score is 81.00. Meanwhile, the mean score of the pretest in the control group is 57.67, and the posttest score is 63.67. It means that the experimental group has a higher improvement score than the

control group.

TABLE IV: DESCRIPTIVE STATISTICS OF PRETEST-POSTTEST

CLASSES			PRETEST Statistic	PRETEST Std. Error	POSTTEST Statistic	POSTTEST Std. Error
Experimental	Mean		55.33	2.235	81.00	1.878
	95% Confidence Interval for Mean	Lower Bound	50.76		77.16	
		Upper Bound	59.90		84.84	
	5% Trimmed Mean		55.19		81.11	
	Median		60.00		80.00	
	Variance		149.885		105.862	
	Std. Deviation		12.243		10.289	
	Minimum		30		60	
	Maximum		80		100	
	Range		50		40	
	Interquartile Range		13		20	
	Skewness		.038	.427	-.212	.427
	Kurtosis		-.253	.833	-.322	.833
Control	Mean		57.67	2.382	63.67	2.559
	95% Confidence Interval for Mean	Lower Bound	52.79		58.43	
		Upper Bound	62.54		68.90	
	5% Trimmed Mean		57.96		63.70	
	Median		60.00		65.00	
	Variance		170.230		196.437	
	Std. Deviation		13.047		14.016	
	Minimum		30		40	
	Maximum		80		90	
	Range		50		50	
	Interquartile Range		23		30	
	Skewness		-.434	.427	.012	.427
	Kurtosis		-.901	.833	-1.213	.833

A normality test was employed to find out the data distribution, as shown in table 5. Since the degree of freedom (df) for each category is 30, we used the Shapiro-Wilk normality test. It can be spotted that all the significance scores of the Shapiro-Wilk test are more than 0.05 (Sig. > 0.05), which means that the data is normally distributed.

TABLE V: NORMALITY TEST OF THE DATA

	CLASSES	Kolmogorov-Smirnov[a] Statistic	df	Sig.	Shapiro-Wilk Statistic	df	Sig.
PRETEST	Experimental	.215	30	.001	.926	30	.038
	Control	.204	30	.003	.894	30	.006
POSTTEST	Experimental	.195	30	.005	.915	30	.021
	Control	.202	30	.003	.907	30	.013

a. Lilliefors Significance Correction

To compare the results of two unpaired groups, we employed independent samples t-test, and the result is presented in Table VI. It can be seen that the Sig. score of Levene's test for equality of variances is 0.09 (p-value > 0.05), which means that the data variance is homogeneity so that each independent sample group is comparable and/ or equal. The score of Sig (2-tailed) is .000 (p-value <0.05), which means that it has both two groups have differences. Therefore, it can be concluded that the hypothesis is accepted; learning the concepts of programming using BBVP has a significant impact on novices.

C. Usability Analysis

Besides statistical analysis, we also evaluate the usability of the learning strategy that we have designed. To appraise it, we measured the BBVP software and learning media by employing questionnaires as provided in Table I and Table II. Table VII shows the results of questionnaires distributed to the students after the experiments were accomplished. The results are presented in two types of scores, namely "RAW" and SUS score. The "RAW" score refers to raw data obtained from the original mean scores, while the SUS score is a standard calculation of SUS in format ranges 0-4 and 0-100 [33]. These all types of scores are closely related to tone statements of positive and negative to distinguish the odd and even questions provided in the questionnaire.

TABLE VI: INDEPENDENT SAMPLES T-TEST

	Levene's TEV		t-test for Equality of Means						
	F	Sig.	t	df	Sig. (2-tailed)	MD	SED	95% CID Lower	95% CID Upper
EVA	7.275	.009	5.460	58	.000	17.333	3.174	10.979	23.688
EVnA			5.460	53.22	.000	17.333	3.174	10.967	23.700

EVA: Equal variances assumed
AvnA: Equal variances not assumed
Levene's TEV: Levene's Test for Equality of Variances
MD: Mean Difference
SED: Std. Error Difference
CID: Confidence Interval of the Difference

TABLE VII: QUESTIONNAIRE RESULTS RELATED TO BBVP AND LEARNING MEDIA

Item No.	BBVP RAW	BBVP SUS score 0-4	BBVP SUS score 0-100	Learning Media RAW	Learning Media SUS score 0-4	Learning Media SUS score 0-100
1	3.73	2.73	68.33	3.53	2.53	63.33
2	1.47	3.53	88.33	1.73	3.27	81.67
3	3.43	2.43	60.83	3.63	2.63	65.83
4	2.13	2.87	71.67	1.73	3.27	81.67
5	3.5	2.5	62.5	3.6	2.6	65
6	2.13	2.87	71.67	2.13	2.87	71.67
7	3.73	2.73	68.33	3.63	2.63	65.83
8	1.9	3.1	77.5	2.07	2.93	73.33
9	3.57	2.57	64.17	3.7	2.7	67.5
10	1.87	3.13	78.33	1.67	3.33	83.33
Average		2.85	71.17		2.88	71.92

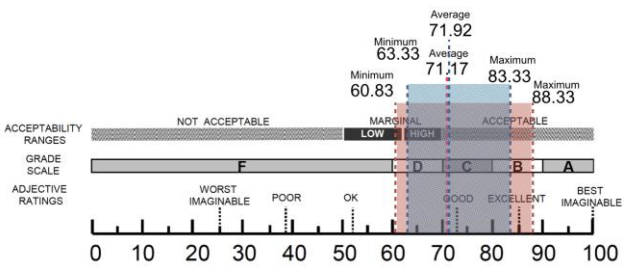

Fig. 6. SUS score categories based on the questionnaire results.

Fig. 6 depicts the SUS scores of BBVP and learning media, illustrating a comparison of acceptability score, quartile ranges, and adjective rating scale, of which the overall limit score of mean is 68 [34]. The red shading block refers to the average score for BBVP; it is 71.17, while the blue shading block is the learning media score, it is 71.92. Based on the adjective rating scale, both BBVP and learning media include good ratings, categorized as good on the grade scale. Meanwhile, based on the acceptability ranges, they include to acceptable category. Fundamentally, these scores try to

assess the even statements (question no. 2, 4, 6, 8, and 10) into positive responses, while the odd statements (question no. 1, 3, 5, 7, and 9) try to adjust the negative responses.

The SUS score is provided in a single value obtained from the calculation of the total score on average, but it still needs to look into the detailed score for each statement [32]. To inspect this more deeply, [33] conducted research to measure the usability of learning media by verifying the shape of a "five-pointed star" with a radar chart. They stated that the more "five-pointed star" looks like, the more positive result of the usability will get. The radar chart is shaped from the ten items of questions in the standard SUS questionnaire. In this study, the items are simplified become 1) frequently used, 2) complexity, 3) easy to use, 4) need support, 5) well integrated, 6) inconsistency, 7) quickly learn, 8) awkward, 9) confidently, and 10) need to learn. However, the assessment tends to subjective measures based on the participants involved.

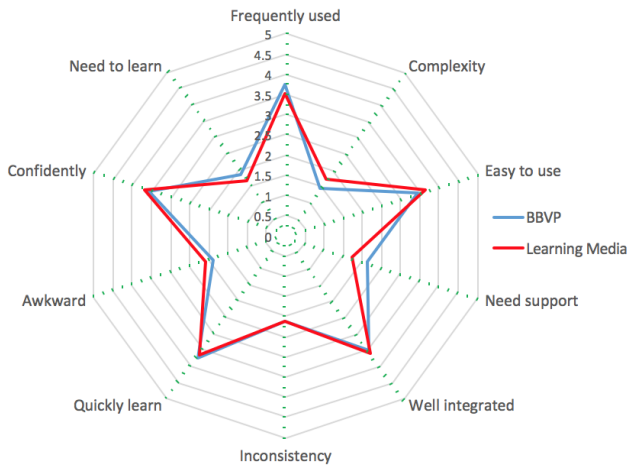

Fig. 7. SUS score categories based on the questionnaire results.

Fig. 7 describes the radar chart converted from the RAW score of SUS in Table VII. The blue line refers to the BBVP score, while the red line points out the learning media. It can be seen that the radar chart is similar to the "five-pointed star", but the top end of the star is not the maximum. The best maximum score for positive tone is at the point of 5, while for negative tone is at point 1. The highest score for positive tone in this study is 3.73 for BBVP and 3.7 for the learning media. The highest score for BBVP consists of two item questions, namely in item number one and number 7, they are frequently used and quickly learned. Meanwhile, the best score for learning media is confidence. Therefore, it can be stated that the students are going to use the BBVP frequently and also learn BBVP quickly. Related the learning media that we have developed, students stated that they felt confidently using the BBVP to learn the concepts of programming.

IV. Conclusion

Based on the results and discussions, it can be concluded that learning teaches the concepts of programming for novices can be handled by BBVP, like ScratchJr. The learning can be designed in PjBL settings by employing a game project to improve students' performance since they learn through the game project they created. By designing and developing a game project, students may learn many programming concepts, such as variables, sequence, iteration or looping, conditional statements, and many more. Even they may instantly learn how to implement a timer and play sounds to trigger the conditional statements in a game project they created so that the game results be more interesting. Further, learning programming may also enhance the skills of problem-solving since the students must be encountered some problems and have to finish them.

Based on the usability evaluation, students stated that they are going to use the BBVP frequently and also stated that learn they are quick to learn BBVP. Related to the learning media that we have developed, students stated that they felt confidently using the BBVP to learn the concepts of programming. However, although the radar chart shows the "five-pointed star" shape, it indicates that each item of questions is not lead to the maximum top end so that each item needs to be enhanced.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

Sukirman managed the project and designed the research and experimental design. Dias Aziz Pramudita constructed the programming course and its instruments for evaluation. Aziz Afiyanto and Utaminingsih administered students survey and aided the teaching/ learning processes. All authors had approved the final version of the paper.

Acknowledgment

This research is fully funded by Universitas Muhammadiyah Surakarta (UMS), Indonesia, through scheme of Hibah Integrasi Tridharma (HIT), Lembaga Riset dan Inovasi (LRI), UMS. Thanks also to the Faculty of Teacher Training and Education and the Department of Informatics Engineering Education that has given us a chance to conduct this research.

References

[1] S. I. Malik, M. Shakir, A. Eldow, and M. W. Ashfaque, "Promoting algorithmic thinking in an introductory programming course," *Int. J. Emerg. Technol. Learn.*, vol. 14, no. 1, pp. 84–94, 2019.

[2] T. Y. Sim, "Online supported learning and threshold concepts in novice programming," *2017 IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, 2017, pp. 85–90.

[3] F. B. 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] C. S. Cheah, "Factors contributing to the difficulties in teaching and learning of computer programming: A literature review," *Contemp. Educ. Technol.*, vol. 12, no. 2, p. ep272, 2020.

[5] J. Nouri, L. Zhang, L. Mannila, and E. Norén, "Development of computational thinking, digital competence and 21st century skills when learning programming in K-9," *Educ. Inq.*, vol. 11, no. 1, pp. 1–17, Jan. 2020.

[6] J. Fagerlund, P. Häkkinen, M. Vesisenaho, and J. Viiri, "Computational thinking in programming with Scratch in primary schools: A systematic review," *Comput. Appl. Eng. Educ.*, vol. 29, no. 1, pp. 12–28, Jan. 2021.

[7] P. João, D. Nuno, S. F. Fábio, and P. Ana, "A Cross-analysis of block-based and visual programming apps with computer science student-teachers," *Educ. Sci.*, vol. 9, no. 3, 2019.

[8] Y. Hu, C.-H. Chen, and C.-Y. Su, "Exploring the effectiveness and moderators of block-based visual programming on student learning: A meta-analysis," *J. Educ. Comput. Res.*, vol. 58, no. 8, pp. 1467–1493, Jul. 2020.

[9] X. Wei, L. Lin, N. Meng, W. Tan, S.-C. Kong, and Kinshuk, "The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy," *Comput. Educ.*, vol. 160, p. 104023, 2021.

[10] S. Sukirman, L. F. M. Ibharim, C. S. Said, and B. Murtiyasa, "A strategy of learning computational thinking through game based in virtual reality: Systematic review and conceptual framework," *Informatics Educ.*, 2021.

[11] D. Kemdikbud, *Kesiapan Sekolah Menengah Pertama dan Sekolah Menengah Atas dalam Menerapkan Informatika sebagai Mata Pelajaran pada Tahun 2019/2020*, Indonesia, 2019.

[12] H. Y. Durak, F. G. K. Yilmaz, and R. Yilmaz, "Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities," *Contemp. Educ. Technol.*, vol. 10, no. 2, Apr. 2019.

[13] W. Deng, Z. Pi, W. Lei, Q. Zhou, and W. Zhang, "Pencil code improves learners' computational thinking and computer learning attitude," *Comput. Appl. Eng. Educ.*, vol. 28, no. 1, pp. 90–104, Jan. 2020.

[14] J. Cuny, L. Snyder, and J. M. Wing, "Demystifying computational thinking for non-computer scientists," *Unpubl. Manuscr. progress, Ref. http//www. cs. C. edu/~ CompThink/resources/TheLinkWing. pdf*, 2010.

[15] I. Fronza, L. Corral, and C. Pahl, "Combining block-based programming and hardware prototyping to foster computational thinking," in *Proc. the 20th Annual SIG Conference on Information Technology Education*, 2019, pp. 55–60.

[16] L. Zhang and J. Nouri, "A systematic review of learning computational thinking through Scratch in K-9," *Comput. Educ.*, vol. 141, p. 103607, 2019.

[17] H. Y. Durak, F. G. K. Yilmaz, and R. Yilmaz, "Computational thinking, programming self-efficacy, problem solving and experiences in the programming process conducted with robotic activities," *Contemp. Educ. Technol.*, vol. 10, no. 2, pp. 173–197, 2019.

[18] R. Scherer, F. Siddiq, and B. S. Viveros, "A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions," *Comput. Human Behav.*, vol. 109, p. 106349, 2020.

[19] Y. A. Uly and B. P. Jatmiko, "Berdasarkan riset linkedin, ini 10 pekerjaan yang paling dicari saat pandemi," *Kompas.com*, 04-Jul-2020.

[20] P. Piwek and S. Savage, "Challenges with learning to program and problem solve: An analysis of student online discussions," in *Proc. the 51st ACM Technical Symposium on Computer Science Education*, 2020, pp. 494–499.

[21] P. Denny, A. Luxton-Reilly, E. Tempero, and J. Hendrickx, "Understanding the Syntax Barrier for Novices," in *Proc. the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, 2011, pp. 208–212.

[22] S. Vaidyanathan, "Teaching coding to kids: What programming language should we use?" *EdSurge*, 2019.

[23] J. Cárdenas-Cobo, A. Puris, P. Novoa-Hernández, Á. Parra-Jiménez, J. Moreno-León, and D. Benavides, "Using scratch to improve learning programming in college students: A positive experience from a non-weird country," *Electronics*, vol. 10, no. 10, p. 1180, May 2021.

[24] C. Chen, P. Haduong, K. Brennan, G. Sonnert, and P. Sadler, "The effects of first programming language on college students? computing attitude and achievement: A comparison of graphical and textual languages," *Comput. Sci. Educ.*, vol. 29, no. 1, pp. 23–48, 2019.

[25] C.-F. Chiu, "Facilitating K-12 teachers in creating apps by visual programming and project-based learning," *Int. J. Emerg. Technol. Learn.*, vol. 15, no. 01, pp. 103–118, 2020.

[26] J. Moon, J. Do, D. Lee, and G. W. Choi, "A conceptual framework for teaching computational thinking in personalized OERs," *Smart Learn. Environ.*, vol. 7, no. 1, p. 6, Dec. 2020.

[27] Y. Jabareen, "Building a conceptual framework: Philosophy, definitions, and procedure," *Int. J. Qual. Methods*, vol. 8, no. 4, pp. 49–62, Dec. 2009.

[28] S. P. Rose, M. P. J. Habgood, and T. Jay, "Designing a programming game to improve children's procedural abstraction skills in scratch," *J. Educ. Comput. Res.*, vol. 58, no. 7, pp. 1372–1411, Jun. 2020.

[29] D. Topalli and N. E. Cagiltay, "Improving programming skills in engineering education through problem-based game projects with Scratch," *Comput. Educ.*, vol. 120, pp. 64–74, 2018.

[30] Hadriana, - Mahdum, - Isjoni, D. Futra, and I. Primahardani, "Online learning management in the era of covid-19 pandemic at junior high schools in Indonesia," *J. Inf. Technol. Educ. Res.*, vol. 20, pp. 351–383, 2021.

[31] J. Brooke, "SUS: A 'quick and dirty' usability scale," *Usability Eval. Ind.*, vol. 189, no. 194, pp. 4–7, 1996.

[32] A. Bangor, P. T. Kortum, and J. T. Miller, "An empirical evaluation of the system usability scale," *Int. J. Human–Computer Interact.*, vol. 24, no. 6, pp. 574–594, Jul. 2008.

[33] D. Hariyanto, A. C. Nugraha, A. Asmara, and H. Liu, "An asynchronous serial communication learning media: Usability evaluation," *J. Phys. Conf. Ser.*, vol. 1413, p. 12018, 2019.

[34] A. Bangor, P. Kortum, and J. Miller, "Determining what individual SUS scores mean: Adding an adjective rating scale," *J. Usability Stud.*, vol. 4, no. 3, pp. 114–123, 2009.

Copyright © 2022 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).

Sukirman is a staff of Informatics Engineering Department, Faculty of Teacher Training and Education, Universitas Muhammadiyah Surakarta (UMS), Indonesia. His research interests are computational thinking, teaching and learning programming, game-based learning, virtual and augmented (mixed) reality for educational purposes.

Dias Aziz Pramudita is a staff of Informatics Engineering Department, Faculty of Teacher Training and Education, Universitas Muhammadiyah Surakarta (UMS), Indonesia. His research interests are ICT for education, multimedia, and game education.

Aziz Afiyanto is a student at Informatics Engineering Department, Faculty of Teacher Training and Education, Universitas Muhammadiyah Surakarta (UMS), Indonesia. He also active in many activities related to learning education and multimedia development for educational purposes.

Utaminingsih is a student at Informatics Engineering Department, Faculty of Teacher Training and Education, Universitas Muhammadiyah Surakarta (UMS), Indonesia. She also active in many student club activities such as learning programming, game development, and mobile application development.