

## **THE IMPACT OF USING VISUAL LEARNING ENVIRONMENT ON STUDENT PROGRAMMING COURSE LEARNING ACHIEVEMENT: A CASE STUDY OF UNIVERSITAS AIRLANGGA**

NASA Z. DINA<sup>1,\*</sup>, WILDA I. SABILLA<sup>1</sup>, KARTONO<sup>2</sup>

<sup>1</sup>Faculty of Vocational Studies, Universitas Airlangga, B Campus, Jalan Dharmawangsa Dalam, 60268, Surabaya, Jawa Timur, Indonesia

<sup>2</sup>Faculty of Science and Technology, Universitas Airlangga, C Campus, Jalan Dr. Ir. Soekarno, 60115, Surabaya, Jawa Timur, Indonesia

\*Corresponding Author: nasazadina@vokasi.unair.ac.id

### **Abstract**

This study investigates the relationship between the use of visual learning environment and student achievements in programming courses at the Faculty of Vocational Studies of Universitas Airlangga. This study involves 200 first year students in the information system department. The 200 students were divided into two groups. Each group consisted of 100 students. Group one was the experimental group and group two was the control group. The lecture in-group one was delivered using visual learning environment programming tools while the lecture in-group two was delivered using traditional methods, which were a presentation, discussion technique. The t-test of the two samples and descriptive statistics were used to study the relationship between visual learning environment and learning achievement in programming courses. This study observed a significant relationship between the use of visual learning environment and student academic achievement, suggested by the statistically significant differences between each group. The significance level value is 0.05. The conclusion is that visual learning environment usage might produce more significant increases in academic achievement than non-usage.

Keywords: Academic achievement, Educational technology, Programming courses, Teaching methods, Visual learning.

## **1. Introduction**

Programming course in the first semester is considered as the most challenging course [1] because first-year students have never written code before so they face difficulties in debugging and writing code. Programming is offered in the first year because it is the pre-requisite of other courses. It is a fundamental course in information technology basis and is still the most difficult course for students.

The lecturer's major concern is the difficulty of teaching algorithm and programming course [2]. The problems do not only come from students but also from lecturers. Poor teaching methodologies, low interaction levels with students and lack of interest contribute to the student difficulties in learning programming. A preliminary study reported the factors affecting learning programming courses in university. Meanwhile, several students need longer time to grasp the algorithm as the primary topic of the programming courses [3, 4]. Some students also need more detailed information from the lecturers than other students. To overcome this problem, algorithm-centric computer science educational tools are built [5]. Students are expected to study the algorithm by themselves outside the class hour. It is very challenging for both students and lecturer, as it requires a lot of practical learning based on case studies. It does not only need intelligence, but also a lot of commitment [2].

There are many tools built to help students deal with difficulties in learning programming skill. The visual learning environment was built to support the teaching and learning process. The use of visual learning environment in this study means software-learning tools such as Green foot and Alice, which are software tools to help students and lecturers to understand programming course better. Greenfoot and Alice are fully animated software learning for programming. Using these programmes, students are expected to obtain a better understanding.

Greenfoot could be used for 14 years old upwards. A graphical and interactive output is combined with the Java programming language as an object-oriented programming language [5]. The main strengths of Greenfoot are a good illustration of object-oriented Theory, good scaling up and easy to get started. The programming language used in Greenfoot is Java. The weakness of Greenfoot is tinkering for younger students because younger students often find it hard to debug if there is a syntax error. In addition, Greenfoot is only designed as 2D system.

Alice is a 3D Interactive graphics programming environment that facilitates the creation of interactive, animated 3D worlds. Alice seems to provide about a solution for the difficulty of teaching object-oriented programming languages. Cooper et al [6] and Bishop-Clark et al. [7] are among several researchers and computing scholars who reported the benefits of using Alice as an object-first tool; students' confidence, which had increased in their programming ability and understood basic programming concepts.

Students were able to understand that each object has its own methods and attributes. The real-life experience of dealing with objects and directing their behaviours through programming constructs and structures enables a smooth transition from Alice to high-level languages such as Java. The similarities between Greenfoot and Alice are making use of visual, sounds at the same time

learning fundamental programming course. Both also use java programming as the basis.

Other than using Greenfoot and Alice, researchers suggested students build simple games using the Scratch game environment [4, 8-10]. Scratch is a visual programming environment that allows younger learners who are primarily ages 8 to 16 to learn computer programming [11]. Usually, it produces animated story projects and games projects. The aims of designing in Scratch are for self-learning and collaborating with peers. Students were expected to learn programming basics. It gives students a specific capability to think widely and innovatively. But Scratch is not suitable for college students because it is too simple and also it uses Pascal programming. Greenfoot and Alice used Java programming [11] so they are more suitable for college students. Syntactically, there are many actual free-texts typing in Scratch so it is not easier to learn more facilitating and enabling than Greenfoot and Alice [12].

Hijon-Neira et al. [13] and Yan [14] used Greenfoot as an interactive programming exercise to teach their Computer Science students and the acceptance was very positive. Meantime Al-Bow et al. [15] used Greenfoot to High school students to learn programming and high school students were more comfortable and interested in technology after attending the class. In the other hand, Alice was implemented too but for a different level.

Kelleher and Pausch [16] and Werner et al. [17] introduced Alice to middle school students. They used a different way to teach programming using Alice. Kelleher and Pausch [16] taught the middle school students to create an animated movie with Alice and Werner and taught them to make a more complicated project like games [17]. Kelleher and Pausch [16] and Werner et al. [17] found that it increased the number of students who were keen to learn computer science and successfully introduced computer science concepts to a broad population.

Here, based on the experiences in using Alice [15-17] and Greenfoot [13, 14] the purpose of this study was to report on how is the impact of using visual learning environment on student programming learning achievement? In more detail, we used Alice and Greenfoot together as a visual learning environment. This study is significantly important because even though some reports [13-17] have shown the result using each method but the studies related to teaching programming using both Alice and Greenfoot in programming course subsequently are hard to find.

Therefore, in the programming course, we choose both Greenfoot and Alice as a visual learning environment to learn programming for first-year students in information system study program to overcome the challenges for both lecturer and students. The results were measured to see whether it is suitable or significantly beneficial for both parties.

## **2. Universitas Airlangga's Information System Study Program**

University is a place where students are expected to obtain an advanced level of education [18] such as level five until level eight. Level five or commonly known as diploma level are qualification designed to provide students with the skills and knowledge needed for work. All the diploma programs offered in the university are usually under a vocational school or faculty of vocational studies.

Universitas Airlangga is a public university and located in Surabaya, Indonesia. It was established in 1954. Until 2018, it has 12 faculties. The youngest faculty is faculty of vocational studies where information systems diploma exists. Since 2000, information systems diploma study program Universitas Airlangga has been meeting the demand of junior programmers in Indonesia. Programming course offered at the university is named "Logic and Programming". It is intended for first-year diploma students. The aim of this course is to introduce the concept of programming and to introduce the logic in solving computational problems. The course is divided into two forms, which are a theoretical and practical lecture.

### **3. Greenfoot and Alice as Learning Tools**

In the Alice environment, learners can select world (the place where they started to drag the statements/expressions) and add 3D actors from the gallery. This actor will be placed in the world. The learner can manipulate the actors using methods, which has already existed in Alice environment. The methods are the same in the traditional programming methods such as if-else statement, loops and recursive.

In the Alice environment, a program is created by dragging the statements/expressions from an exciting list of statements/ expression into the code window; learners are not required to write any code. This ensures that all the code dragged to the code window is syntactically correct as it is all taken from a predefined entry. This help directing the learners' efforts towards understanding the concepts and mechanisms of programming rather than language syntax.

The result of the Alice program can be known directly after the learners run the program. The example of the result such as the implementation of the looping concept to the object than objects moves its body repeatedly according to how many loops learners assigned. It could walk straightforward or backwards or even jump. Figure 1 shows the interface of Alice environment.

Greenfoot is almost the same as Alice, only Greenfoot is more complex than Alice. Greenfoot uses 2D but Alice in 3D. It is also made for novice programmers to develop games, animation, and simulation. The main concept is the same, there are world and actors. In Alice, learners begin to understand the concept and logic, while in Greenfoot learners are requested to understand about class, inheritance and all associated with object-oriented programming etc.

In Greenfoot, there are no drag and drop, learners should write the code in the world. Then the learners could run the program and see how does the program work. The interface of Greenfoot's world can be seen in Fig. 2 and the animation after the program is run in Fig. 3. If it is taught by the complexity then it can be concluded that Greenfoot is taught after Alice. The former focuses on the logic and the latter handled the syntax and object-oriented program.

Both of Greenfoot and Alice have their display whether 2D or 3D but in the traditional method it has none. Figure 4 illustrates the interface of the traditional method. It affects the motivation of novice learners because they never handled programming. The interface is not attractive and they already confused enough about the syntax, the logic, etc.



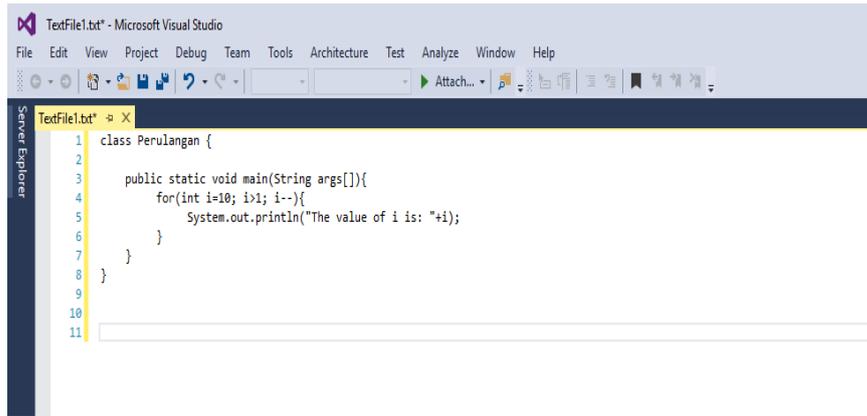
Fig. 1. Alice interface.



Fig. 2. Greenfoot script interface.



Fig. 3. Grenfoot interface.



```

1 class Perulangan {
2
3     public static void main(String args[]){
4         for(int i=10; i>1; i--){
5             System.out.println("The value of i is: "+i);
6         }
7     }
8 }
9
10
11

```

**Fig. 4. Traditional method.**

## 4. Research Methodology

### 4.1. Research subject

The participants of the study are male and female students from information study program, Faculty of Vocational Studies, Universitas Airlangga who are enrolled in the programming course. The programming courses start at the beginning of the first semester of the university year. The first semester usually starts in August each year. Universitas Airlangga is used as a model because this research is expected to help the lecturers to redesign the curriculum of Programming course in Information Systems study program.

The participants consisted of 200 male and female students, which were divided into two groups. Each group consisted of 100 students with the same ratio of male and female students: one group is experimental and the other is the control group.

The experimental group was given lectures in the experimental group were delivered using Greenfoot and Alice as a visual learning environment. Meanwhile, in the control group, the lectures were delivered conventionally. The control and experimental groups were all in their first semester and they were divided based on the class they were registered in. The experimental group and the control group are from different cohorts but the data was taken when they were in the first semester.

### 4.2. Teaching condition

Both groups (control and experimental) were taught by the same senior lecturer to avoid confounding impacts of individual bias and also to put into the judgment that using visual learning environment in teaching will not influence the traditional teaching method to avoid the misinterpretation result of the study of the two groups.

This is also to avoid the effect of the location variable in the study. To ensure the quality of the students' academic achievement of giving lectures in both groups. To minimise the standard deviation and the value of the variables: participants in both groups were from the same faculty and same study program.

### 4.3. Experimental demonstration

The independent variables in this study are the use of Greenfoot and Alice and the dependent variable is students' academic achievement. The control variables are the faculty to which, students belong, the students' level, the class' location, which is used and the lecturer who teaches. The experiments were done four times since the experimental group and the control group were from a different year from 2015 until 2018, which they are registered. The data was taken when they were in the first semester. The first semester was usually started in January until May each year.

The measure of students' performance-based achievement in a programming course included final exam 35%, mid-year exam 25%, quizzes 20% and exercises 20%. In a semester, there were 14 weeks. Every week students were given an exercise and the students should submit the exercises every week then the quizzes were given twice in a semester. The quizzes were usually given a week before the mid-year exam/final exam. The mid-year exam was held in the 7<sup>th</sup> of the 14<sup>th</sup> week and final exam was held in the 14<sup>th</sup> week.

The example of quizzes, which the students had to take, is displayed in Fig. 5. The number of questions is about 50 questions. All the questions are multiple choices and students had to answer it in 100 minutes. All the quizzes questions were about the concept of programming. Meanwhile, in mid-exam and final exam, students were given a mini project. The example of the mini-project for the exam can be seen in Fig. 6. There was only two questions in each exam while the time given to finish it around 100 minutes. Students worked individually for each test in the computer laboratory.

```
Given the following variables char c = 'c'; int i = 10; double d =
10; long l = 1; String s = "Hello"; Which of the following will
compile without error?
a. c=c+i;
b. s+=i;
c. i+=s;
d. c+=s;
```

**Fig. 5. Example quizzes question.**

Make a method / function to solve the problem below and add it to the Matrix class (change the data type to int and just select the required method). Suppose that given (inputted) matrix A. Please read the following instruction:

- a. Check the number of multiples of n (inputted and in the form of integers) in the matrix A. If there is a number multiplied by n then print "ADA" and if it is not there then print "NOTHING" and print the index number position multiples of n

|   | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 9 | 5 | 7 | 4 |
| 1 | 6 | 4 | 8 | 9 |
| 2 | 7 | 5 | 1 | 2 |
| 3 | 6 | 8 | 3 | 1 |

- b. Calculate the multiplication of all elements per line then save in array v1. Display array v1 and print if multiplication results are multiples of five, then print " MULTIPLE FIVE " if not then print "NOT MULTIPLE FIVE"

**Fig. 6. Example mid and final exam question.**

#### 4.4. Research questions

Thus, the study problems of this article are what is the impact of using visual learning environment in teaching programming courses on students' academic achievement? So that, the purpose of this study is to discover the impact of using visual learning environment for students' academic achievement who took a programming course by using Greenfoot and Alice as a visual learning environment and the students who did not use any tools or only been taught using traditional methods. The following research hypothesis was expanded from study problems:

*H<sub>0</sub>*: There are no statistically significant differences between the average grade of the experimental group and the control group of the students' academic achievement in a programming course.

*H<sub>a</sub>*: There are statistically significant differences between the average grade of the experimental group and the control group of the students' academic achievement in a programming course.

The importance of this study is a result of the some problems like: improving teaching methods for the students, motivating students on learning programming course, increasing students' academic achievement in a programming course, understanding the ability of students in learning programming course, developing lecturers' ability of using visual learning environment in teaching programming course, encouraging the use of visual learning environment in the university environment for theoretical and practical bases.

#### 4.5. Pre and post survey

The pre-survey and post-surveys were done to fulfil several purposes such as focusing the minds of the students, assessing the level of satisfaction, and the level of retention of knowledge [19]. Students were asked to complete pre-survey containing questions whether or not they have previously used Greenfoot and Alice as a visual learning tool in the programming course.

Meanwhile, at the end of the semester, students were asked to complete post-survey containing questions whether or not Greenfoot and Alice are helpful as a visual learning tool in the programming course. The printout questionnaires were distributed when pre and post surveys were conducted.

### 5. Results and Discussion

After conducting the experiment, the researcher analysed the study outcomes using a t-test also called student's t-test to measure the ratio between two groups and the differences. The t-test generates t-value. The larger the t-value, the more difference there is between groups. When t-test is run, the bigger the t-value, the more likely it is that the results are repeatable. Every t-value has a p-value to go with it. A p-value is a probability that the results from the data sample occurred by chance. P-values are from 0% to 100% but it is usually written as a decimal. For example, a p-value of 5% is 0.05. Low p-values are good because they indicate the data did not occur by chance. For example, a p-value of 0.01 means there is only a 1% probability that the results from an experiment happened by chance. In most cases, a p-value of 0.05 (5%) is accepted to mean the data is valid.

There are three main types of a t-test: Independent Samples t-test, which compares the means for two groups; Paired sample t-test, which compares the means from the same group at different times; A one sample t-test tests the mean of a single group against a known mean [20].

The significance is a statistical term that tells how sure are that a difference or relationship exists. We might be very sure that a relationship exists, but is it a strong, moderate, or weak relationship. After finding a significant relationship, it is important to evaluate its strength. Significant relationships can be strong or weak. Significant differences can be large or small. It just depends on the sample size. If t-test is lower than the critical value from the t-table then the finding is not significant. It fails to reject the null hypothesis. The probability is high that the difference or relationship happened by chance, and  $p$  is greater than the critical alpha level ( $p > \alpha$ ) [20].

### 5.1. Students' demographic data

Every programming class environment in the first semester consists of students that includes a range of ages from 18-19 years old students. There is no student with disabilities in the class. All of the students came from a science major in high school.

### 5.2. Pre and Post Survey

#### 5.2.1 Pre-survey

Figure 7 illustrates the result of pre-survey. The students were asked whether or not they have heard of or used Alice and Greenfoot before programming courses. The result was only 4.17% has used Greenfoot and Alice. They studied Greenfoot and Alice when they were in high school. Most students (about 95.83%) never knew about Greenfoot and Alice. This was because the students were only in the first semester in the early stages of a learning programming course.

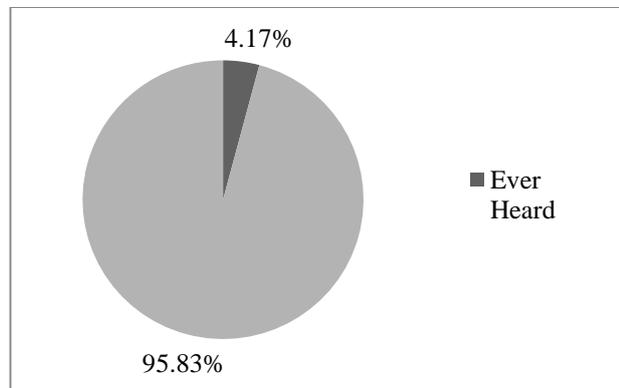
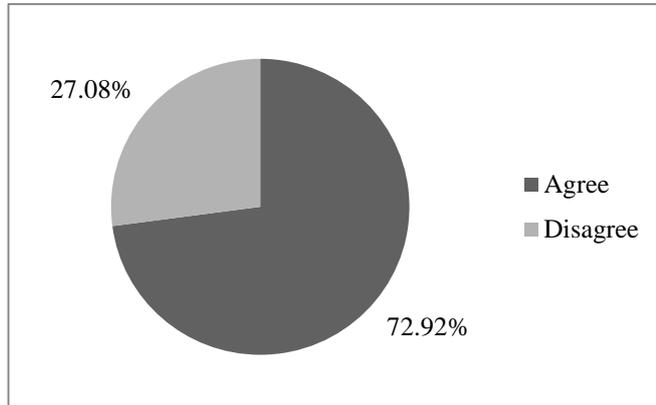


Fig. 7. Pre-lecture.

#### 5.2.2. Post survey

After performing a programming course using Greenfoot and Alice, students were asked to fill in the survey to assess their understanding. The result can be seen in Fig. 8, which indicates that 72.92 % of students agree that Greenfoot and Alice

helped them to understand the concept of programming course while the rest disagreed. It is understandable because every student has their own method to understand lecture material.



**Fig. 8. Post-lecture.**

### 5.3. Results related to hypothesis

From the t-test statistical results, there are significant differences between both groups at the significance level 0.05 and it indicates the unequally the two groups. The two groups were tested by independent t-test because those groups are different subjects [20]. The result of independent t-test was t-value is greater than t-table ( $2.95 > 1.65$ ). It proves that there are significant differences then it means that  $H_0$  is rejected. Statistically, significant differences mean a result is unlikely due to chance. Further results are shown in Table 1. After the significant test was conducted then we compare the academic achievement by grading. Grading practices in higher education have been driven by educational goals such as providing feedback to students, motivating students, comparing students, and measuring learning [21]. In this article, grading used as a comparison between learning achievement from each group. The grading system that the university used consist of seven scores. Those are “A” (excellent), “AB” (very good), “B” (good), “BC” (almost good), “C” (satisfactory), “D” (poor) and “E” (failed).

The average score from the control and experimental group are shown in Fig. 9, the number of “AB” and “B” score in the experimental group are doubled or almost tripled than the control group. In the opposite, the number of “BC”, “C” and “D” score are lesser than the control group. While “A” score is slightly lower by one and “E” score from both group are the same. The average score of the former group is “AB” while the latter group is “B”. The number of students from the experimental group mostly obtain “B” score about 34% while students from the control group mostly obtain “C” score about 29%.

After getting the statistical result of the experimental and control group, the positive impact of using Greenfoot and Alice as a visual learning environment was observable. The students’ grade of the experimental group is significantly better than the control group. This suggests that using visual learning environment in programming course in information system study program was effective to obtain better learning and better grades.

**Table 1. Result of experimental and control group.**

|                     | Number | Median | Standard deviation | Student t-value | Pearson correlation |
|---------------------|--------|--------|--------------------|-----------------|---------------------|
| <b>Control</b>      | 100    | 2.50   | 0.10               | 2.95            | 0.003               |
| <b>Experimental</b> | 100    | 3.00   | 0.94               |                 |                     |

After delivering a course using Alice and Greenfoot, we found that:

- Programming practicum in computer laboratory is required because programming is not a course, which cannot only be memorized. It demands active participation from students. One way teaching is not suitable, students are not active in traditional based teaching programming classes [18].
- Semester credit hour for programming course should be added because for the last four years, it was only two semesters credit hour. At least three credit hours because classroom experience needed more time to conduct.
- The suitable teaching technique is classroom experience. Students were given a challenge about a case study and they were expected to solve the case study but before the complexity increased, they should be taught how they can work on simple parts of a more complex program and see from the start how it is the interaction of many objects that make a program work. Nevison and Wells [22, 23] suggested that the use of case studies from the very beginning of the first course can be a successful strategy.
- In the initial course, students had to finish the exercise given in the previous class. The exercise was short answer questions. Therefore, students prepared with the material, which they will discuss in the next class. The lecturer should check the list of students who had done the exercise and those who were not. It was part of the assessment. It motivated students to finish exercises before attending the class. Goetz and Barber [24] have done an experiment related to pre-session homework; the result shows that students who completed pre-session homework activity have better performance in classroom experience compared with students who did not complete pre-session homework. The students were highly motivated, very relevant, that they felt moderately prepared to perform discussion in class. They thought that attending the session was moderate to very worthwhile.
- The additional simple demonstration of the case study in the class by the lecturer helped to boost the students' confidence level because if the lecture gave none demonstration then the students had no idea how to solve the problems. As a result, the student's motivation decreased. In addition, as the students observed lecturer, they will learn how the lecturer understand, diagnose and solve the problems. The students will know how to correct the mistakes from the lecturer [20]

From the above result, the teaching process required a special technique. Specifically, teacher needs to give an attractive case study to be solved by the students using software-learning tools provided. It helps students to visualize the problem and find the solution needed. Nandiyanto et al. [25] suggest that teachers play major role because if they want to keep the level of students understanding, they have to give an attractive method for attracting student concentration and focuses. It is their decision whether to use software learning tools in class or use

none. In this research, it has been proved that the use of visual learning environment helps to increase the level of students understanding based on their score in Fig. 9.

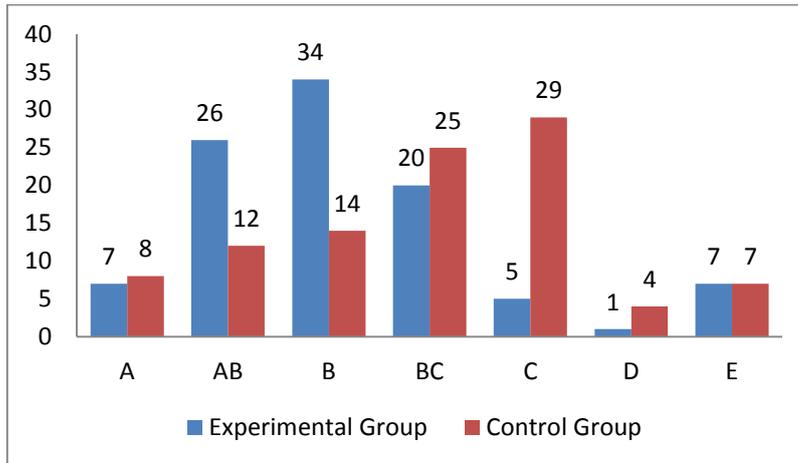


Fig. 9. Students' score from experimental and control group.

### 6. Conclusions

It can be concluded that the progress of the academic achievement for the experimental group, which use programming learning tools is greater than the control group, which only used the traditional teaching method. This suggests the effectiveness of visual learning environment using Greenfoot and Alice in presenting programming course. According to the study result, the study recommends the following: using visual learning environment in theoretical and practical studies and emphasizing the use of them as an learning programming tool in teaching, giving training courses to programming course lecturers regarding the use of visual learning environment, doing more studies on using visual learning environment in the academic curriculum in Universitas Airlangga.

#### Nomenclatures

|         |   |
|---------|---|
| f-test  | Statistical test in which, the test statistic has a F-distribution under the null hypothesis                                      |
| $H_a$   | Alternative hypothesis  |
| $H_0$   | Null hypothesis   |
| $p$     | Probability   |
| p-value | Probability value   |
| t-table | A table showing probabilities areas under the probability density function of the t distribution for different degrees of freedom |
| t-test  | Statistical hypothesis test   |
| t-value | Value on the t-test   |

#### Greek Symbols

|          |                      |
|----------|----------------------|
| $\alpha$ | Critical alpha level |
|----------|----------------------|

**Abbreviations**

|    |                         |
|----|-------------------------|
| 2D | Two-dimensional space   |
| 3D | Three-dimensional space |

**References**

1. Mckee-Scott, J. (2015). *A study of an educational game for learning programming*. Master Thesis. The University of British Columbia, Vancouver, Canada.
2. Begosso, L.C.; Begosso, L.R.; Goncalves, E.M.; and Goncalves, J.R. (2012). An approach for teaching algorithms and computer programming using greenfoot and python. *Proceedings of the Conference on Frontiers in Education*. Seattle, United States of America, 1-6.
3. Zumaytis, S.; and Karnalim, O. (2017). Introducing an educational tool for learning branch and bound strategy. *Journal of Information Systems Engineering and Business Intelligence*, 3(1), 8-15.
4. Law, K.M.Y.; Lee, V.C.S.; and Yu, Y.T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers and Education*, 55(1), 218-228.
5. Kolling, M. (2010). The greenfoot programming environment. *ACM Transactions on Computing Education*, 10(4), 21 pages.
6. Cooper, S.; Dann, W.; and Pausch, R. (2003). Teaching objects-first in introductory computer science. *Proceedings of the 34<sup>th</sup> SIGCSE Technical Symposium on Computer Science Education*. Reno, Nevada, United States of America, 19-23.
7. Bishop-Clark, C.; Courte, J.; and Howard, E.V. (2006). Programming in pairs with Alice to improve confidence, enjoyment, and achievement. *Journal of Educational Computing Research*, 34(2), 213-228.
8. Kobsiripat, W. (2015). Effects of the media to promote the scratch programming capabilities creativity of elementary school students. *Procedia-Social and Behavioral Sciences*, 174, 227-232.
9. Ferrer-Mico, T.; Prats-Fernandez, M.A.; and Redo-Sanchez, A. (2012). Impact of scratch programming on students' understanding of their own learning process. *Procedia-Social and Behavioral Sciences*, 46, 1219-1223.
10. Saez-Lopez, J.-M.; Roman-Gonzales, M.; and Vazquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school : A two year case study using "scratch" in five schools. *Computers and Education*, 97, 129-141.
11. Maloney, J.; Resnick, M.; Rusk, N.; Silverman, B.; and Eastmond, E. (2010). The scratch programming language and environment. *ACM Transactions on Computing Education*, 10(4), 15 pages.
12. Utting, I.; Cooper, S.; Kolling, M.; Maloney, J.; and Resnick, M. (2010). Alice, greenfoot, and scratch - A discussion. *ACM Transaction Computer Education*, 10(4), 11 pages.
13. Hijon-Neira, R.; Velazquez-Iturbide, A.; Pizarro-Romero, C.; and Carriço, L. (2013). Improving students learning programming skills with progames - programming through games system. *Proceedings of the 14<sup>th</sup> International*

- Conference on Human-Computer Interaction (INTERACT)*. Cape Town, South Africa, 579-586.
14. Yan, L. (2009). Teaching object-oriented programming with games. *Proceedings of the Sixth International Conference on Information Technology: New Generations*. Las Vegas, United States of America, 969-974.
  15. Al-Bow, M.; Austin, D.; Edgington, J.; Fajardo, R.; Fishburn, J.; Lara, C.; Leutenegger, S.; and Meyer, S. (2008). Using greenfoot and games to teach rising 9<sup>th</sup> and 10<sup>th</sup> grade novice programmers. *Proceedings of the ACM SIGGRAPH Symposium on Video Games*. Los Angeles, California, 55-59.
  16. Kelleher, C.; and Pausch, R. (2007). Using storytelling to motivate programming. *Communications of the ACM - Creating a science of games*, 50(7), 58-64.
  17. Werner, L.; Campe, S.; and Denner, J. (2012). Children learning computer science concepts via Alice game-programming. *Proceedings of the 43<sup>rd</sup> ACM SIGCSE Technical Symposium on Computer Science Education*. Raleigh, North Carolina, United States of America, 427-432.
  18. Pratheesh, N; and Devi, T. (2015). Necessity of learning analytics in software engineering education. *Journal of Engineering Science and Technology (JESTEC)*, 10(3), 269-281.
  19. Al-Atabi, M.; Shamel, M.M.; Chung, E.C.Y.; Padmesh, T.N.P.; and Al-Obaidi, A.S.M. (2013). The use of industrial visits to enhance learning at engineering courses. *Journal of Engineering Science and Technology (JESTEC)*, Special Issue on Engineering Education, 1-7.
  20. Lind, D.A.; Marchal, W.G.; and Wathen, S.A. (2014). *Statistical techniques in business and economics*. New York: McGraw-Hill.
  21. Schinske, J.; and Tanner, K. (2014). Teaching more by grading less (or differently). *Life Sciences Education*, 13(2), 159-166.
  22. Nevison, C.; and Wells, B. (2004). Using a maze case study to teach object-oriented programming and design patterns. *Proceedings of the Sixth Australasian Conference on Computing Education (ACE)*. Dunedin, New Zealand, 207-215.
  23. Nevison, C.H.; and Wells, B. (2003). Teaching objects early and design patterns in java using case studies. *Proceedings of the 8<sup>th</sup> Annual SIGCSE Conference on Innovation and Technology in Computer Science Education*. Thessaloniki, Greece, 94-98.
  24. Goetz, J.E.; and Barber, C.R. (2015). Evaluating a pre-session homework exercise in a standalone information literacy class. *Communications in Information Literacy*, 9(2), 176-185.
  25. Nandiyanto, A.B.D.; Asyahidda, F.N.; Danuwijaya, A.A.; Abdullah, A.G.; Amelia, N.; Hudha, M.N.; and Aziz, M. (2018). Teaching “nanotechnology” for elementary students with deaf and hard of hearing. *Journal of Engineering Science and Technology (JESTEC)*, 13(5), 1352-1363.